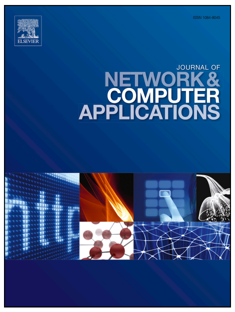


Energy-aware and adaptive fog storage mechanism with data replication ruled by spatio-temporal content popularity

Ruben Vales, Jose Moura, Rui Marinheiro

PII: S1084-8045(19)30092-X
DOI: https://doi.org/10.1016/j.jnca.2019.03.001
Reference: YJNCA 2337

To appear in: *Journal of Network and Computer Applications*

Received Date: 8 August 2018
Revised Date: 4 January 2019
Accepted Date: 2 March 2019

Energy-Aware and Adaptive Fog Storage Mechanism with Data Replication Ruled by Spatio-Temporal Content Popularity

RUBEN VALES, ISCTE-IUL
JOSE MOURA, ISCTE-IUL, Instituto de Telecomunicações
RUI MARINHEIRO, ISCTE-IUL, Instituto de Telecomunicações

Data traffic demand increases at a very fast pace in edge networking environments, with strict requisites on latency and throughput. To fulfil these requirements, among others, this paper proposes a fog storage system that incorporates mobile nodes as content providers. This fog storage system has a hybrid design because it does not only bring data closer to edge consumers but, as a novelty, it also incorporates in the system other relevant functional aspects. These novel aspects are the user data demand, the energy consumption, and the node distance. In this way, the decision whether to replicate data is based on an original edge service managed by an adaptive distance metric for node clustering. The adaptive distance is evaluated from several important system parameters like, distance from consumer to the data storage location, spatio-temporal data popularity, and the autonomy of each battery-powered node. Testbed results evidence that this flexible cluster-based proposal offers a more responsive data access to consumers, reduces core traffic, and depletes in a fair way the available battery energy of edge nodes.

KEYWORDS

Edge storage, Clustering, Replication, Distance, Spatio-temporal data popularity, Offloading, Energy awareness

1 INTRODUCTION

Mobile devices are inherently resource constrained in terms of storage, processing and energy. These resource constraints limit the number of applications that are adequate to be run on mobile devices [1][2][3]. To tackle this limitation, [4] arguments for the use of mobile cloud computing (MCC). MCC offers to mobile devices storage and processing resources located at centralized distant servers [1][3]. These clouds can provide unlimited resources; however, MCC does not easily scale, centralized servers are prone to the classic problem of “single point of failure”, and they are typically located far away from mobile users. This long distance between each user and the remote cloud inflicts a high latency, experienced by mobile users when accessing data or services stored at the remote cloud.

Due to both technological enhancements and the increase on the user demand, mobile applications are even more requiring real-time data processing. These applications include high quality multimedia dissemination, distributed interactive games, and sharing of high amounts of data with low latency [5]. In fact, the already referred high latency, added to bandwidth bottleneck, communication overhead, and location blindness experienced by mobile users when accessing remote clouds, raises a serious problem in mobile scenarios [6]. It is then very urgent to bring computing resources closer to end-users to satisfy the strict requisites imposed by emerging mobile applications. Consequently, remote MCC needs to be surpassed with alternatives that offer a more edge-oriented computing paradigm [7][8], by migrating resources, such as services and data, closer to end users and devices. This is on the genesis of the edge-oriented computing paradigm, that allows more responsive cloud services, accomplished by extending the services from the core in cloud data centers to the edge of the network, by placing intermediate nodes between the cloud and the end user, which are responsible for better serving ubiquitous smart devices, fulfilling user resource requests. In addition, we can observe the high proliferation of smartphones and the continuous evolution of their resources [9]. All these aspects make possible to use the extra resources of smartphones in novel edge services. Consequently, we envision to bring cloud services even closer to consumers, by empowering devices as providers, i.e., remote cloud services might be replaced or enriched with local fog services. Using this new paradigm, legacy functional aspects of computing, networking, and storage need to be revisited. This paper is focused on the storage service improvements that this paradigm shift brings.

The current work contributes to the area of fog storage, by designing and implementing a hierarchical hybrid file system, which uses edge devices as both consumers and providers of local data storage, as shown in Figure 1. It is a hybrid system because, on one hand, it has a centralized management and on the other hand, it enables both data storage and data dissemination on a completely distributed way.

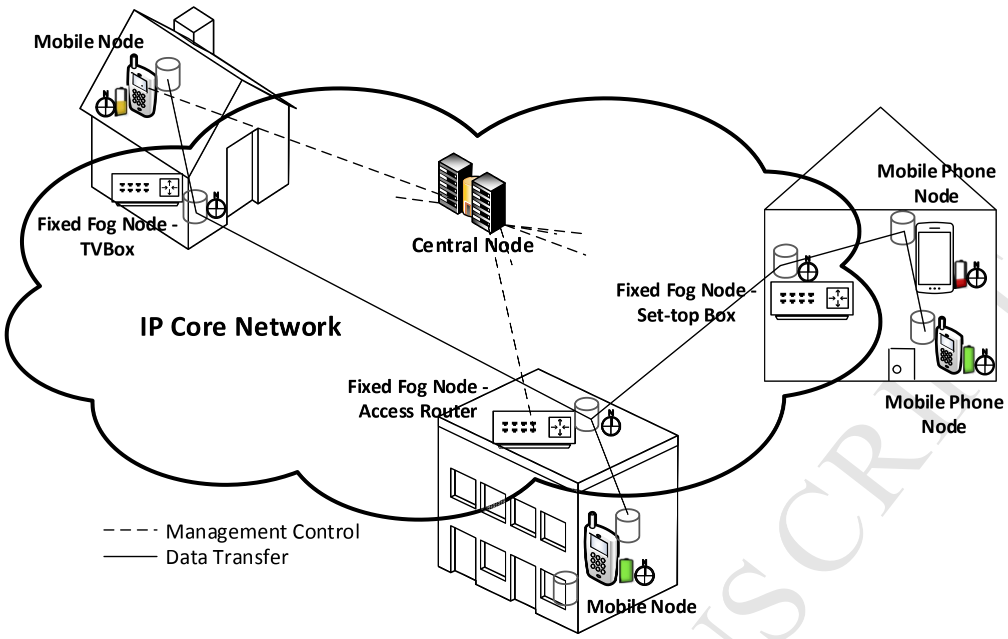


Fig. 1. Architecture of the proposed fog storage system.

Figure 1 shows two data-storing node types: mobile nodes and fog nodes. A central node manages data storage at the network edge by considering networking nodes' energy levels and localization information. The peripheral data storage also considers data's popularity. These local metrics are sent to the central node, using management control messages, shown in Figure 1. Data messages are exchanged directly among data-storing nodes in a completely distributed way.

The current proposal contributes for the research area of fog storage, as follows:

- the edge computing is useful not only for either pushing data or services closer to users or for diminishing their access latency, as vastly addressed by the literature. In fact, according to our perspective, edge computing also obliges the networking research community to holistically rethink some classical concepts such as location, distance, energy, and popularity.
- it extends HDFS [10] with Pharos [11] to obtain a cluster-based file system enhanced with the capability for evaluating distances between pairs of nodes in a scalable way; it also considers spatial-temporal data popularity [12] and nodes available battery energy [13];
- this hybrid data storage was deployed in a testbed with fog devices, to evaluate the proposed solution in a comprehensive way. The obtained results show that this proposal replicates data to edge devices based on node localization and spatio-temporal data popularity, offers data to consumers with low latency, offloads traffic from core to the edge, and enables a fair energy consumption among battery-powered nodes;
- the proposal can support emerging mobile services similar to [14][15]; it can also enhance the caching service already supplied by content distribution networks (CDNs) [16], extending the caching services from cloud datacenters to fog devices;
- our contribution is agnostic regarding where the storage is placed, being it fog devices, such as mobile devices, middleboxes or core devices such as storage servers.

The rest of the paper is organized as follows: section 2 revises the literature in topics related with our contribution; after that, section 3 explains how the proposed solution was designed and implemented; section 4 presents and analyzes testbed results; and finally, section 5 concludes the paper.

2 RELATED WORK

There are several alternatives to MCC such as edge computing [8], cloudlets [2][17][18], multi-access edge computing (MEC) [19][7], or fog computing [20][7][17]. First, the edge computing tries to overcome the latency issue of MCC by introducing an intermediate layer between the remote cloud and end-users. This intermediate layer is responsible for satisfying the mobile users' resource requests. Second, the cloudlets have been proposed as trusted stationary machines or a cluster of stationary machines with high capabilities at the neighborhood of mobile devices. In this way, cloudlets act as resource providers, most often installed along with Access Points (AP) to enable the communication with mobile devices, and in some cases both cloudlet and AP are integrated on a single entity. However, they may not always be available due to wireless short range, channel congestion or interference issues. Third, MEC, proposed by Telcos, deploys servers that offer cloud computing capabilities inside the Radio Access Network (RAN), near mobile subscribers. Mobile network operators allow the use of the access network, where low latency and high-bandwidth as well as direct access to real-time radio network information (such as subscriber location, cell load, etc.) is available. This can be used to allow content, services and applications to be accelerated, increasing server responsiveness from the edge. Additionally, MEC servers are context aware, as they manage information on end devices, such as their location and network information. But their capacity is limited, therefore deciding which and how resources can be managed at the edge can still be a trick endeavor [13]. Furthermore, MEC can become an expensive option for mobile users due to the data access is made typically via cellular networks. Fourth comes the fog computing paradigm, term introduced by Cisco Systems, whose rationale for coining this term is that a fog is nothing more than a cloud that is closer to the ground. In fog scenarios, the storage and processing requisites are ubiquitously addressed by nearby fog nodes, that fulfil the necessary resources, in a widely distributed manner [8], in the form of fog nodes [21], possibly at different node levels and densities [22]. These fog nodes are typically access routers and, in some cases, could be machine to machine (M2M) gateways [7]. Fog computing can be pertinent to support data management for smart cities [23], including the deployment of 5G mobile access technology [24] or for IoT [25]. Table 1 lists and compares the main technical aspects among these four architectures for mobile services. From these, the architecture more well-aligned with our current work is fog computing.

Table 1. Comparison Among Several Architectures for Mobile Services

Main Technical Aspects	**Mobile Cloud Computing**	**Cloudlet**	**Multi-Access Edge Computing**	**Fog Computing**
Referenced work	[26]	[27]	[28]	[17]
Distance to mobile node	High	Low	Low	Low
Latency	High	Variable	Low	Low
Backhaul load	High	Low	Medium	Low
Computational power	Ample	Ample	Limited	Medium
Storage capacity	Ample	Ample	Limited	Medium
Access	xG	WiFi	xG	Heterog
Mobility support	Good	Limited	Good	Medium
Coverage	Ample	Limited	Ample	Ample
Context awareness	No	Could be	Yes	Yes
Reliability	Low	Low	Medium	High
Hierarchy	1 tier	2 tiers	2 tiers	3+ tiers
Cooperation among mobile nodes	No	No	No	Yes
Energy efficiency	High	Medium	Medium	Low

As already discussed, mobile cloud computing is evolving to a more well-identified edge-oriented computing paradigm, by migrating services even closer to end users. Aligned with this paradigm, new computing architectures should enable mobile devices to share among them their available resources to support local services, with minimum latency, in a coordinated manner. This contrasts with the previous edge implementations, where the mobile device's exclusive role in the cloud was that of a consumer. There is a myriad of proposals, that, regarding control, follow two architectures types: centralized such as FemtoClouds [29], CACTSE [30], Hyrax [31], and MOMCC [32] or decentralized, where nodes keep track of their own resources, such is the case with EECRS [33], Phoenix [34], E-DRM [35], the proposal of Barca [36], the proposal do Lacuesta et al with their spontaneous ad hoc mobile cloud computing network [37] and the proposal of Monteiro et al. [38].

However, none of the previous proposals directly uses mobile node's energy level as a metric in their resource allocation strategies. Additionally, precise node location is not properly addressed in the Network Coordinates (NC) of their choice. Moreover, previous works do not address in a complete way how to manage the data storage closer to the final consumers, in particular, how data storage can be refined by the data popularity. Literature discussed below address some of the functional characteristics aligned with our work, such as: energy efficiency, nodes NC, and edge data caching.

Several works [39][40] addressed energy efficiency in wireless sensor networks in a similar way as we do in the current contribution. In addition, [41] proposes a solution to place edge servers in order to support both energy efficiency and diminishment of the access latency. The authors of [42] go a step further to diminish the access latency in the sense that they discuss client-storage techniques for supporting multiple consumer cloud storage on mobile devices. Nevertheless, these techniques based on mobile devices could rapidly deplete their batteries. To avoid this problem, we think that, whenever possible, most part of the data storage effort should be supported by the network edge devices (e.g. base stations, access points, routers) that have a permanent energy supply. Additionally, [43] exploits content caching and delivery techniques for 5G and [44] surveys literature about caching in information centric networks. Our proposed fog storage system integrates very well with [43][44] because the former enables energy-efficiency and the latter work is driven by efficient data discovery distributed mechanisms. But it is not only important to improve the energy efficiency. In fact, it is fundamental to minimize data traffic at the backhaul link, as suggested in [13]. This is normally designated by traffic offloading [45]. Our solution follows this approach and also offers online caching during data delivery [46].

We have also found some proposals regarding the use of node NC in a networking system, such as: landmark-based [47] or distributed [48][11][49]. A landmark-based approach is used in the Content Addressable overlay Network (CAN) [47]. It relies on a few fixed nodes, called landmarks, which are known to the cluster. In this way, clients measure their Round Trip Time (RTT) latency to each landmark and then compute their network position in a Euclidean coordinate system. Considering these steps, the node position in the coordinate system is represented with a vector with a cardinality matching the total number of landmarks being used. The number of landmarks has a very strong impact on the predicted node localization accuracy when compared with its real localization. A main drawback of this approach occurs when there is landmark failure or overload, which increases latency to the client and consequently affects the system’s accuracy. But another less obvious problem is the triangle inequality violation (TIV) [50]. This happens because a routing path between two nodes with a direct link can very often have a longer path than an alternative path through an intermediary node. Consequently, systems merely using the Euclidean distance model are in-capable of denoting TIVs in their metric space, and therefore the performance is seriously impacted. An alternative method to the previous discussed CAN is Vivaldi [48], where nodes determine their coordinates by sending ping packets to some closest nodes. The RTT between two nodes is modelled as a physical force. The nodes’ coordinates are related with the diverse existing forces among the nodes. The final goal is to reach a system state with a minimum value of energy, in the sense that, when the minimum value of system energy is achieved, the system knows the localization of all nodes with a higher accuracy. Pharos [11] is another distributed localization system. In Pharos, uses a cluster-based design, where nodes have a pair of network coordinates: global and local. These new characteristics empower Pharos with better RTT predictions than in the case of Vivaldi. Pharos also relies on the use of n-dimensional Euclidean space for their coordinates, accompanied by a height coordinate. The height coordinate models the latency penalty of network access links, such as queuing delay [51]. A similar work is available in [49]. The current work uses and compares both Pharos and CAN approaches.

Another aspect in our proposal is the caching of data at the network edge [52][4][53][13][54][55], since it can significantly improve bandwidth and energy efficiency in wireless networks when compared with caching only at infrastructure nodes. The authors of [52] argue that a considerable amount of information could potentially be used in wireless networks for achieving efficient content caching and low-latency data delivery to the consumers. Information to be processed may include: channel access; interference; users’ mobility; battery level; accurate predictions about the data demand; and even the social connections among data consumers. Similarly to our contribution, the work in [25] also deals with the delivery of data at the network edge, in particular IoT data. They have addressed the complete lifecycle of data, mainly the interchange of that between heterogeneous devices, and scalability, by adding several servicing layers. However, these works haven’t considered energy limitations of mobile devices or sensors or, a novelty in our proposal, the distribution of data in several caches within the network infrastructure depending on the spatio-temporal data popularity [52][16][54]. This data popularity is evaluated using a dynamic model [56][57][58]. Dynamic models evidence global gains on flexibility, reliability and performance in comparison with static ones. A previous contribution [59] discusses the literature on stream processing engines and mechanisms for exploiting resource elasticity features of cloud computing in data stream processing. Resource elasticity allows for an application or service to scale out/in according to fluctuating usage demands. In addition, the data stream processing is performed at the network edge before delivering data to consumers, which causes delay. Our work has a slightly different perspective. Data are directly transferred from a source node to a consumer without any intermediation or brokerage support. In [60] studies data obfuscation and encryption techniques on multi-cloud storage systems. These are out of scope of our work but one can consult [60] for further details. Other interesting aspects for studying are service migration [61] and Big IoT data analytics [62].

To sum up, this paper presents the design and implementation of a hybrid cluster-based system that supervises a fog domain. Our system collects the following metrics that previous proposals have only partially considered: distance from consumers to the data, node battery level, and spatio-temporal information popularity. It has also a replication data algorithm that decides whether to replicate or not depending on the previous discussed metrics. Both metadata and policies are centrally managed, but, in opposition, data storage and dissemination among users are both managed in a distributed way. Table 2 compares our proposal with previous work, highlighting the novel aspects integrated by our research.

Table 2. Comparison of Major Features of Our proposal with Related Work

Major Features	**Access Latency**	**Data Replication Based on Data Popularity**	**Energy Efficiency**	**Data Interplay Between Fog and Core**
[39]	No	No	Yes	No
[40]	No	No	Yes	No
[25]	No	No	No	Yes
[41]	Yes	No	Yes	No
Our current proposal	Yes	Yes	Yes	Yes

3 PROPOSED SOLUTION

This section has two parts: i) discuss the hybrid design proposal; ii) details the implementation of the proposed data storage for relevant edge computing environments.

3.1 Architecture

This paper proposes a fog computing architecture [17]. This has fog nodes which refer to devices commonly installed at user's premises, such as: switch, router, set-top box, etc. Using fog nodes to store data enhances data availability and brings data closer to consumers. We have proposed a hybrid architecture, where a central node has the next components (Figure 2): i) a master DFS instance which is responsible for managing the filesystem's metadata; ii) a Data Placement Scheme which runs resource allocation strategy; iii) a Network Topology Map augmented by rich context awareness; iv) a Data Control that receives nodes coordinates and battery level information. Both fog and mobile nodes, run server DFS instances, for transferring data directly among nodes using a decentralized operation mode. In addition, the mobile nodes run a client DFS instance to interact with the central node (Figure 2).

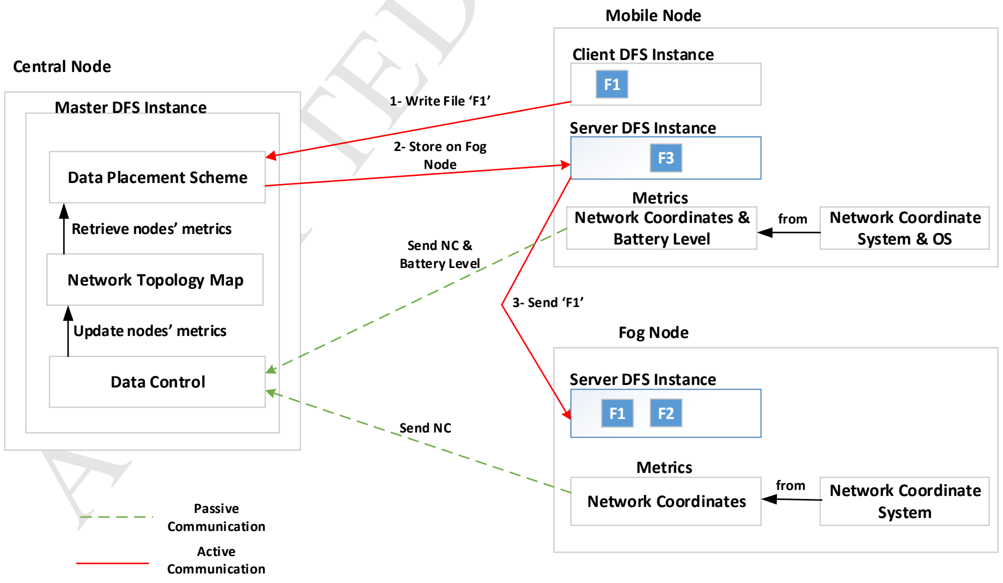
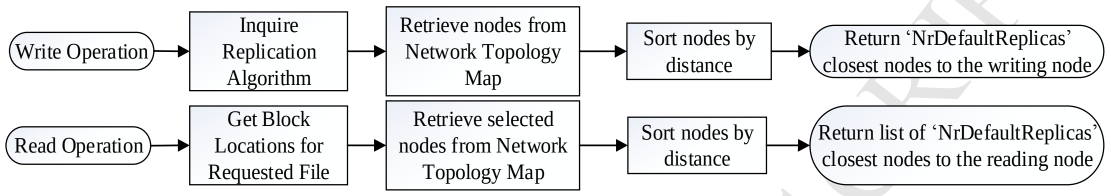


Fig. 2. Interaction between central node and data-storing nodes.

Figure 2 shows a solution design for integrating the custom metrics (node localization and node battery energy level) in our system. The mobile nodes store some part of the existing data inside the edge cloud. There are two types of communication. The first communication type is passive, and it updates the context awareness information (node's localization and energy level) on the central node. It is passive because the metrics aren't explicitly requested; they are periodically retrieved and sent by the data-storing nodes, and then received and saved on the central node. The localization of each node is retrieved using a network coordinate system that should be deployed on each data-storing node. If the data-storing node has battery, it should also retrieve the battery energy level via the local operating system (OS). The central node receives this information in a Data Control unit, which updates the network topology map with the transferred metrics. The network's topology map should be kept updated, as it is used in the data placement scheme. This decides where data get stored/retrieved from. The second communication type (active) is also present in Figure 2. It is associated to a filesystem write request from the mobile node. The data placement scheme weights the cost between distance and battery level for all the network nodes, to determine which node the data should be transferred to. We follow-up detailing how filesystem operations are used in the proposed system to perform data replication in a dynamic way.

3.1.1 Data Replication Scheme. Write and read operations follow the steps shown in Figure 3. We have a default number of replicas in the system for each file. This is a system's parameter and it is designated by 'NrDefaultReplicas'. This enhances data availability and the management of available battery energy.



Write Operation
Inquire Replication Algorithm
Retrieve nodes from Network Topology Map
Sort nodes by distance
Return 'NrDefaultReplicas' closest nodes to the writing node
Read Operation
Get Block Locations for Requested File
Retrieve selected nodes from Network Topology Map
Sort nodes by distance
Return list of 'NrDefaultReplicas' closest nodes to the reading node

Fig. 3. The interaction between central node and data-storing nodes.

A novel aspect of our solution is how distances are evaluated among nodes. In this proposal, distances between nodes are energy-aware, as it is explained below. When a write operation is requested, the central node accesses its replication algorithm, which is a part of the data placement scheme. This replication algorithm evaluates distances between nodes in a novel way by factoring in their localization and their energy level (see expression (1)). From (1) is noticeable how the replication algorithm evaluates the distance between the writing node (Node1) and each potential replication node (Node2). The weighted distance (WD) function has two terms. The first is related with the Euclidean distance between the nodes and this term is directly proportional to WD. The second term is related with the battery status of Node2. In opposition to the first term, this second term is inversely proportional in relation to WD. This means if Node2's battery has a low energy level (i.e. short lifetime), then the distance between the nodes is increased. In this case, Node2 probably would not be selected as a replicating node because Node2 will be considered far away from the writing node. This is also a novel facet of our work.

$$WD_{(Node1,Node2)} = T.\overline{Node1, Node2} + \omega.\frac{1}{BatteryLvl_{Node2}}\ (s) \qquad (1)$$

Expression (1) has two parameters. Considering the first one, T, when we assign to it the value of one, this means the replication algorithm evaluates directly the distance between two nodes according to their network coordinates. Alternatively, the second parameter, ω, it controls the level of fairness in energy depletion as the content is disseminated within the network.

When a read operation is requested, the central node verifies which nodes store the requested file, and then retrieves context awareness information from both the reader and the file-storing nodes. The central node then runs a sorting algorithm that evaluates the distances among the file-storing nodes and the reader, using (1). The central node sorts these WDs and creates a sorted node list, with the closest node in the first position, and sends this list to the reader, so the reader can retrieve the file from the closest node to itself, with minimum delay.

Additionally, the data placement scheme includes a replica management module that manages the number of replicas per file in the fog cloud storage service, based on files' popularity. The replicas should be placed in the areas where those are very popular. By high popular data within a specific area, it means that users within that area and at a quite recent time interval have made a significant number of requests for those data. We propose a reactive replica management (RM) module. Figure 4 shows the flow diagram for the RM module.

The RM module shall run whenever a 'read' request is received on the central node. The RM module calculates the files' popularity according to their near-past access frequency. The module then "draws" a circumference, with radius 'b', around the reader node's position. This radius is inversely proportional to the file's popularity. So, a high popular file has a relatively low search radius. Then, the RM module verifies if any of the nodes that store the requested file are already located within the circle area associated to the search radius. At this step, two options could occur. On one hand, if at least one node is within the circle area, then the data are considered close enough to the reader, and no replication takes place; the read operation resumes normal functionality by sending to the reader node the list of sorted closest nodes. On the other hand, if all the nodes that store the requested file are outside of the circle area, then the system registers a 'hit' for that user/file. The term 'hit' reflects the system's intent to replicate a file to a node that is closer to the reader. After a configurable amount of consecutive 'hits'– 'hit threshold', the system finally replicates the file to the closest node to the reader node using (1). The "hit threshold" is a very useful parameter to confirm in a stable way to our proposal an increase on a popularity file, avoiding undesirable system oscillations. As the file is initially stored too far apart from the potential reader node, then our proposal discovers that situation and it moves that file to the vicinity of that reader node. In this way, the reader nodes experience smaller download times than they would experience if content stay far-away from them.

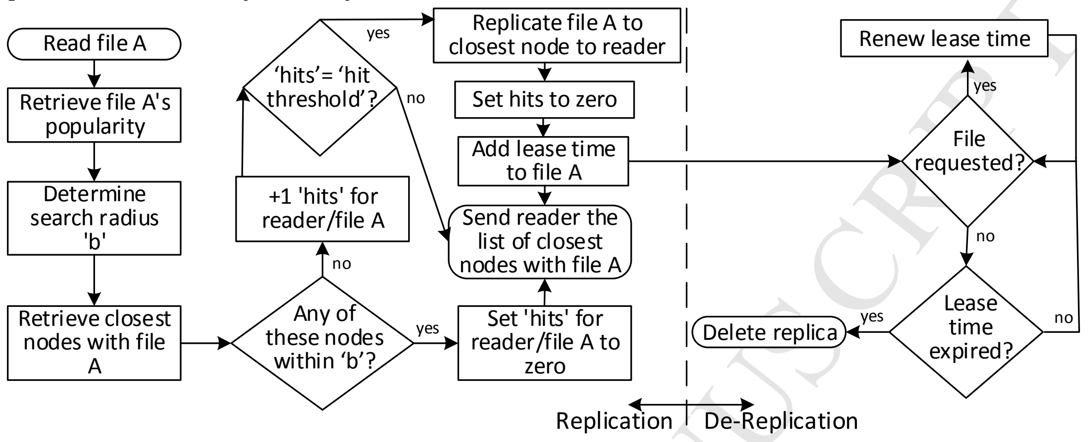

Read file A
Retrieve file A's popularity
Determine search radius 'b'
Retrieve closest nodes with file A
Any of these nodes within 'b'?
yes
no
+1 'hits' for reader/file A
'hits'= 'hit threshold'?
yes
no
Replicate file A to closest node to reader
Set hits to zero
Add lease time to file A
Send reader the list of closest nodes with file A
Set 'hits' for reader/file A to zero
Renew lease time
yes
File requested?
no
Lease time expired?
yes
no
Delete replica
Replication
De-Replication

Fig. 4. Flow diagram for the Replica Management module at Central Node's Data Placement Scheme.

Our current solution allows popular content to get consistently replicated to the network edges. Alternatively, unpopular content doesn't get so much replicated in the network, as the search radius around the reader node should be large enough to include the nodes that already store the requested file, avoiding data replication. The expression (2) evaluates the 'b' search radius around the reader node.

$$b = k.a + \beta \frac{1}{P} \quad (s) \qquad (2)$$

In (2), *P* means the file's (temporal) popularity and *a* is the minimum possible search radius (i.e. spatial popularity). Parameters *k* and *β* are configurable parameters. They regulate the replication sensitivity according to node proximity (spatial popularity) and (temporal) popularity level, respectively. By varying *k*, the intensity of replication that occurs in the system changes according to node proximity. Lower values of *k* will make the system replicate with more intensity, because it requires requested data to be placed very close to the reader node. Similarly, *β* controls the impact of popularity on the system's replication intensity. When high values of *β* are used then these values imply a low level of data replication. This also means (temporal) popularity has almost no impact on data replication.

Our proposal to evaluate files' popularity is inspired on the temporal sliding window average, which is used in TCP to estimate Ack timeouts. It uses moving average calculations to smooth out short-term fluctuations and highlight long-term trends. The popularity will reflect the trend of access frequency for any given file. The file popularity is also shared amongst all the data chunks associated to that file. The expression (3) evaluates files' popularity.

$$P = \frac{1}{Avg\ \Delta t_n} \quad (s\text{-}1) \qquad (3)$$

As visualized in (4), Δt_n is the time that has passed from the last access of the file to the present time. Parameter $\propto$ is a constant that determines the system's reaction to the current Δt_n. The Avg Δt_n of a file ends up being equivalent to its access frequency in the near past. It relates to the file popularity history. When a file is accessed more often, its Δt_n will decrease, and so will the Avg Δt_n, which will reflect a high popularity for that file, as shown in (3).

$$Avg\ \Delta t_n = \Delta t_n . \propto + (1 - \propto) . Avg\ \Delta t_{n-1} \quad (s) \qquad (4)$$

Analyzing the previous four expressions, (1)-(4), one can conclude that our proposal is based on a spatio-temporal data popularity [63] to decide how the data should be replicated within the network. The data replication is controlled by a searching radius around the reading node (see (2)). In this way, as the data popularity increases then the searching radius inversely decreases. This also increases the probability of a data replication being executed within the network infrastructure.

3.1.2 De-replicating unpopular content. It is also important to avoid over-replicated data occupying unnecessary storage, essentially in nodes with scarce available resources. Figure 4 also shows that a lease time is given to files that are replicated beyond the default 'NrDefaultReplicas'. The lease time is renewed whenever the file is requested. If the lease time expires, the file is deleted on the node furthest away from the initial writer of the file, or its last known position. This way, the system gets balanced out in a stateless mode, by keeping the file replication level correctly tuned to its instantaneous popularity.

In the following sub-section, we discuss the implementation of our proposal for a fog DFS that is aware of node localization, file popularity, and available energy on each battery-powered node.

3.2 Implementation

This sub-section starts by justifying why we have opted for Hadoop Distributed File System (HDFS) [10] to be implemented as the data sharing system as well as detailing its implementation on each node type. Afterwards, we focus on how data-storing nodes retrieve their position and battery level. Then, we describe how these metrics are integrated in the data sharing system. Finally, the implemented data replication scheme is discussed.

3.2.1 Data Sharing System. Recalling, our proposal has a central node, fog and mobile nodes. Each run different OSs, forming a heterogeneous environment. We have opted for HDFS [10] as the data sharing system. A strong reason for choosing HDFS is that it is implemented in Java. This enables running the same Java application on any machine independently of its hardware and operating system. We next discuss how HDFS has been implemented on each node type of our proposed system. First, the central node runs Ubuntu and is responsible to launch the HDFS namenode. Secondly, the fog nodes use OpenWrt, which is a specialized network OS that enables routing between wired and wireless networks. Thirdly, we have used Android for mobile nodes. Android works with a different Java implementation than the one required for HDFS. We have used the chroot method to port HDFS into Android.

3.2.2 Network Coordinate System and Battery Level. Two different network coordinate system (NCS) have been implemented on the datanode devices. They allow for collecting datanodes' positions. The first NCS is a landmark-based approach, inspired on the Content Addressable Network (CAN) topology [47], and the second NCS is a decentralized approach designated by Pharos [64]. We below detail the implementation of each of these two NCSs.

CAN implementation in each datanode is now discussed. Here, three stationary devices are known by all nodes. They will be used as landmarks. Datanode devices will ping each one of these landmarks sequentially. The ping measures the RTT latency between the datanode and each landmark associates it to a specific level. Each node has coordinates (lvl A, lvl B, lvl C), which are associated to the landmarks (A, B, C) latency levels. For example, a datanodes' position could be (0,2,1) meaning it has a 0-30ms RTT latency to the landmark A, a 101+ms latency to B, and a 31-100ms latency to C. After a namenode knows its network coordinates, it is ready to send them off to the HDFS namenode. In this way, the namenode is updated by all the datanodes. Then, the namenode can evaluate the Euclidean distances among the datanodes, using their received coordinates.

An implementation of Pharos [11] that is written in twisted python has also been used. Each datanode machine runs a client of Pharos. Using the initial Pharos implementation, when a node moves to a new area which could be considered as belonging to a different cluster, the node cluster doesn't get updated. So, every node remains in the initial cluster where it was placed at start-up, regardless of its mobility. In this way, we have changed the Pharos client implementation to offer a script that takes in consideration the node mobility and exports the updated node localization to whom could be interested in that information. By periodically running this script, the datanode device is now able to collect their updated network coordinates and send them to the HDFS namenode.

Integrating the energy level metric to be used in the DFS, was a matter of collecting that metric on the datanode with battery and sending it to the Data Control unit on the namenode, as shown in Figure 2. This has been done by sending this energy metric (and the already discussed updated node localization coordinates) in the periodic HDFS heartbeat message that each datanode sends to the namenode. The heartbeat feature is used natively by HDFS namenode to check the liveness of every datanode. The heartbeat receiving code running at the namenode has also been changed. So, it can retrieve the metrics received via the heartbeat message and update the network topology with routing paths among the datanodes. These paths have associated energy-aware distances. In this way, the distance between two datanodes enables the namenode to evaluate the delay associated to the routing path between them.

The next step is the HDFS namemode to issue via a secure shell (ssh) a remote execution in each datanode of a configuration script to allow that datanode to communicate with others, using the emulated environment but, assuming real values for both rate and delay of each routing path. As an example, one can suppose that the namenode aims to configure a routing path from datanode A ($Aip="192.168.1.2") to datanode B ($Bip="192.168.1.3") with a rate and delay values of respectively 30mbit and 100ms ($ABdelay="100ms"). To achieve this, the namenode should namely configure datanode A, issuing the next command: *ssh root@192.168.1.2 "bash –s" < config.sh $ABdelay $Bip.* The Table 3 shows the contents of the script "*config.sh*" with some commands associated to the Linux utility Traffic Control (TC). This script after being remotely executed by namenode in datanode A, it configures datanode A with real network metrics (rate, delay) associated to a flow path from datanode A to datanode B.

Table 3. TC commands to configure datanode A with real network metrics of a flow path to datanode B

```
tc qdisc add dev eth0 root handle 1: htb # define hierarchical token bucket for flow
policing
tc class add dev eth0 parent 1:1 classid 1:11 htb rate 30 mbit burst 50kb mtu 10000 #
configure flow average rate with 30mbit
tc qdisc add dev eth0 parent 1:11 netem delay $1 # configure flow delay with 100ms
tc qdisc add dev eth0 parent 1:0 protocol ip u32 match ip dst $2 flow id 1:1 # define
the forwarding rule in datanode A to enable the network flow A -> B
```

3.2.3 New Data Replication Scheme. Data placement scheme modifications were made on the HDFS namenode. Main changes are as follows: modify the replication strategy as well as the sorting strategy to evaluate expression shown in (1); log the diverse times each file has been requested; these instances of time are also used to update the file popularity; and implement the replica management module based on file popularity. We next explain the more relevant changes we have made on the HDFS namenode.

Replication Strategy

HDFS deploys its replication strategy in a class designated as block placement module. We replaced its code for the Weighted Distance (WD), evaluating expression (1). This way, whenever a file is written to the filesystem, the block placement module calculates the WD from the writer to all the other nodes in the system. It then sorts the WDs and chooses which nodes to write to, with a criterion of the closest node to the writer node first and then the second closer node and so on up the parameter "number of replicas" is reached. When a file is requested for a read operation, the system also checks the sorting strategy. Again, we replaced the code in the sorting module for our WD, following (1). In this way, it is returned a sorted list of nodes with the closest node to the reader node at its top position and the farthest node to the reader node at its bottom position.

File Popularity

The file popularity is evaluated as shown in (3). This equation expects the Avg Δtn for that file. The Avg Δtn is calculated in (4). It needs the Δtn associated to the time the read request was received, and Avg Δt_{n-1}. This has been implemented by logging read requests in a structure like: Hashmap1(File, Hashhmap2(timestamp, Avg Δtn)).

When is accounted the first request for a file read, a new entry in Hashmap1 is created for that file, which has its value pointing to Hashmap2, which saves its key as the timestamp of the last received request time, and the value as the Avg Δt_n associated to that timestamp. The Avg Δt_n is calculated using equation (4), by first evaluating Δt_n. Δt_n is the difference between the current timestamp and the last saved timestamp for that file. Second, it retrieves the previously stored Avg Δt_n, which is associated to the last saved timestamp. In this way, the RM module evaluates the file popularity, using the two terms in (3).

Replication Management

As shown in Figure 4 (left side), when a file is requested for a read operation, a sequence of system checks is performed. They verify if a file should be replicated to a node closer to the reader. Another extra mechanism was implemented in HDFS that replicates a file between two specific nodes. This has been implemented by using two HDFS features. The first feature changes the replica count on files. The second feature runs a block recovery tool periodically to feed both replication management and HDFS with the exactly same view about the system status. In this way, the block recovery tool keeps in a completely synchronized manner the files' replica counts against the total amount of replicas in the filesystem.

De-Replication Management

The de-replicating data module has been implemented, by adding, in the first place, the replicated files to a list at replication time. A thread that is launched at namenode's start-up, periodically verifies, for each file in the list, if the configured 'lease time' has expired (Figure 4, right side). If it has, then the system decreases the replica count for that file by one. Similarly, to the replicating process, the block recovery feature has another method that checks if a file has more replicas in the filesystem than it should. In this case, it calls the delete replica method in the data placement scheme. This method has been changed to find and delete the farthest replica from the last known position of the original writer.

The next section discusses obtained results from the tests made over our proposal to evaluate the network coordinate system's accuracy and the data placement scheme's performance.

4 EVALUATION RESULTS

This section presents functional and performance tests and discusses the obtained results. In the sub-section 4.1 we discuss the next aspects: the evaluation scenario; the testbed specification; and the set of configuration parameters used during our experiments. In the sub-section 4.2, we present and discuss the obtained results from the diverse experiences we have made to evaluate our current fog storage proposal.

4.1 Evaluation Scenario

The network was configured, as shown in Figure 5, which mimics a typical networking layout scenario on the Internet. It shows three edge networks, designated by cluster's A, B and C. The network topology of Figure 5 has two link types. The first type aggregates the links among nodes in the same edge network; these links have their rate set to 300Mbit/s. Each of these edge networks has a gateway, which forwards traffic to nodes on other edge networks. The second link type aggregates the links between gateways, emulating the backbone connections of current topology. These connections have their rate set to 30Mbit/s. In this way, we assume the core link capacity per user is 10 times less than edge link capacity.

Table 4 shows the VMs specifications used in the testbed. The communication between the diverse VMs was supported by network adapters "Internal network" of the Hypervisor. Then, each link of our topology was emulated using the traffic control (TC) Linux tool. To configure the latency link connecting two datanodes, a shell script was run on the namenode, that: i) retrieves every datanode's position; ii) calculates the euclidean distance between the two datanodes; iii) translates the distance into a latency value in milliseconds; and then iv) establishes a ssh connection into each datanode to issue the tc command that emulates the evaluated latency (and the bandwidth) in the communications link between the two datanodes. In this way, the latency of each link is configured accordingly the distance between the nodes that are interconnected by that network link. Figure 6 shows the node layout for test scenarios, including the delay of each link of the network topology.

Some system parameters values have been configured and remained unchanged throughout tests. The 'NrDefaultReplicas' has been set to two. The 'hit threshold' was set to three. All the obtained "average results" have a confidence interval (CI) of 95%. In the next sub-section, we present and discuss the evaluation results of our fog storage proposal.

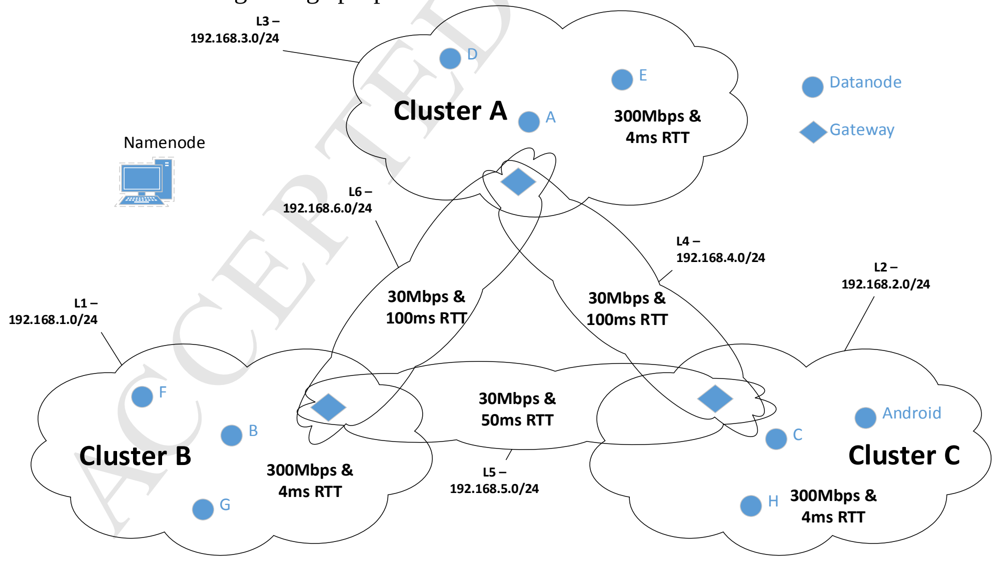


Fig. 5. Network topology for test scenarios.

Table 4. Testbed specifications

Node Type	Hypervisor	CPU	RAM	Storage	OS
Namenode	VirtualBox	Virtualized CPU	4GB	30GB	Ubuntu
Fog Datanode	VirtualBox	Virtualized CPU	512MB	34GB	Openwrt
Mobile Datanode	Genymotion	Virtualized CPU	2GB	34GB	Android
Host		Intel i5-4200M@2.50Ghz	16GB	1TB	Windows

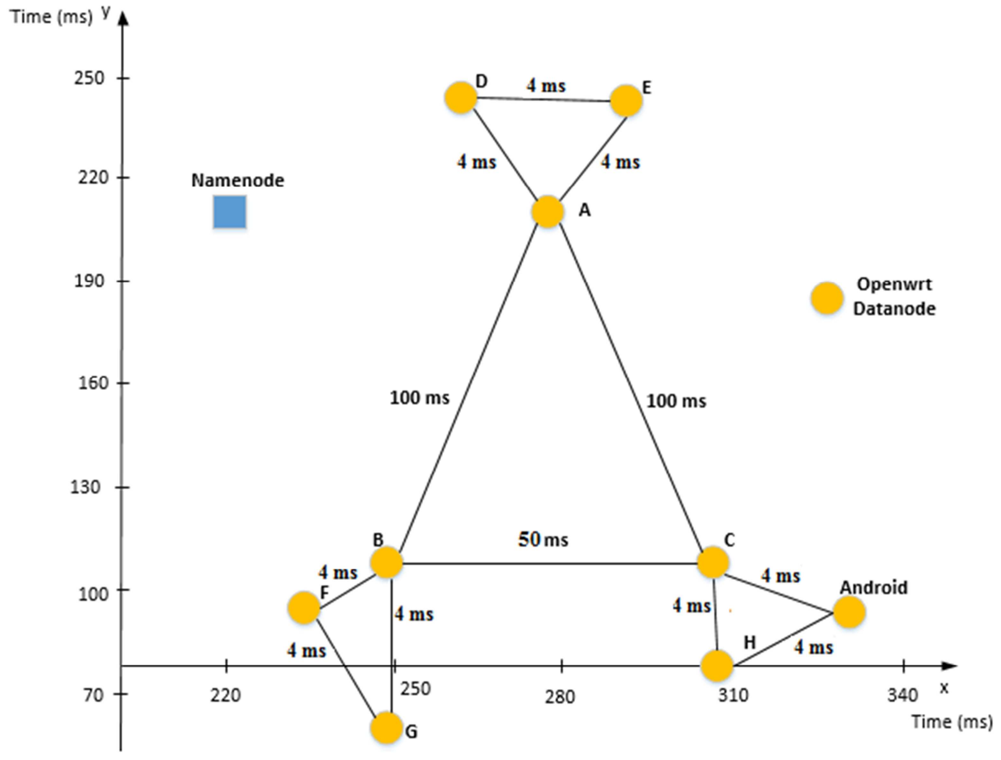


Fig. 6. Node layout for test scenarios.

4.2 Analysis of Results

The implemented prototype was initially tested, by studying the accuracy of each NCS for measuring the distances among nodes. This information is used in (1) to determine where to read or write data from. The system has also been tested regarding the data placement scheme. In addition, some performance metrics have been tested, which we will describe in the following sections.

4.2.1 Measure the Accuracy of Network Coordinate Systems for Efficient Node Localization (Test I). A test scenario has been created to evaluate the network coordinate system (NCS) localization accuracy. The test consists of instructing the Android node in Figure 5, to move around the distinct localizations within the map topology of our scenario. For each occupied position by the Android node, the system collects information about the Android node "correct" coordinates as well as the measured latency between the same Android node and each other node of our network. At the end, the collected data are processed to verify if, for each location, the discretized coordinates (and associated discretized distances among the nodes) obtained via the latency measurements, using a triangularization method, correlate well or not with the "correct" Android coordinates. In other words, we measure the accuracy level of the studied NCS.

We have experimented CAN with three landmarks fixed in the same position but changing the number of levels to discretize the latency and associate to discrete distance values (Table 5).

Table 5. CAN Topology latency discretization for different number of levels

Number of Levels	Latency discretization (ms)
3	0-30; 30-100; 100+
4	0-30; 30-50; 50-75; 75+
5	0-30; 30-50; 50-75; 75-100; 100+
6	0-30; 30-50; 50-75; 75-100; 100-120 ; 120+

The previous CAN test has been repeated for Pharos. When using Pharos, each node runs a client that continuously converges its network coordinates to an optimal solution. So, the system's accuracy has been studied for different converging times. To enable studying this convergence time, a slight modification was made to the test scenario. Now, the Android node stays in each position for a configurable amount of time – PositionDelay, before collecting the latency links and network coordinates and moving on to the next position. Furthermore, the Pharos client on each node is modified to run every six seconds. In this way, by changing the PositionDelay between tests, we can study the system's accuracy for different converging times. We cycle the PositionDelay between four, eight and fourteen seconds, which represent the 0.66; 1.33 and 2.33 iterations per position, respectively. This is evaluated by: *NrIterationsPerPosition=PositionDelay/PharosIterationTime*. We below compare Pharos and CAN.

As visualized in Figure 7, when CAN used only 3 levels of discretization, it evidences low accuracy (i.e. 61% accuracy, meaning in 39% of cases the CAN distances were not accurate). For higher "number of levels" the CAN accuracy slightly increases but it stabilizes at a maximum accuracy value around 75% for 5 and 6 levels. As shown in Figure 8, the minimum accuracy value of Pharos (i.e. 85% for 0.66 iterations/position) is still higher than the maximum measured accuracy of CAN (i.e. 75%). The maximum value we measured for Pharos was 90% for 2.33 iterations/position. This result occurs because the iterative algorithm requires some extra time to converge to the more accurate distances among nodes. So, Pharos shows an interesting tradeoff between its accuracy and nodes mobility pattern.

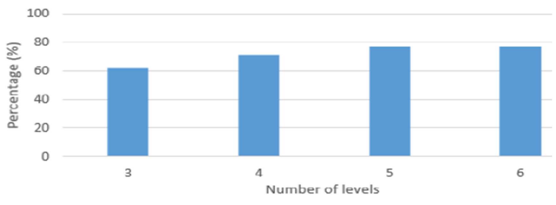


Fig. 7. CAN accuracy vs. the number of levels.

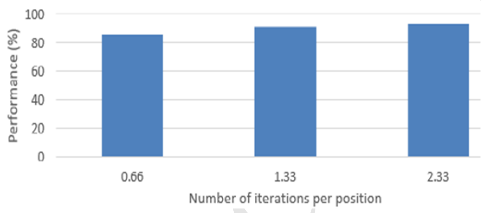


Fig. 8. Pharos accuracy vs. the iterations / position.

4.2.2 Evaluate Data Replication Scheme. We have tested the prototype for some performances aspects, such as: the effect of the "hit threshold" parameter on the proposed solution functionality; the system's fairness in energy depletion when distributing content; the processing overhead on the namenode; the system's responsiveness to different replication parameters; and the traffic offloading from the network core to edge. We discuss all these results below.

<u>"Hit threshold" Parameter Study (Test II)</u>

As already explained before, after three (i.e. the default value of "hit threshold") consecutive read requests for a specific file, the system could replicate that file from a too-far away datanode to datanodes closer to the file consumers. Once the file is replicated, the clients should benefit from lower download times. We have tested this functionality, using the topology of Figure 5. The obtained results are shown in Figure 9. From this figure, we see that at times 98s, 100.9s and 103.2s the download period is considerably long due to the reading node did not have any nearby node with a copy of the file the former node aims to consume. In addition, for each reading request, there is normally a delay of 1s between the time instant when the reading node issues the reading request and the time when the file download is initiated. A considerable amount of this delay is due to the namenode that initially is involved in the reading process, namely to process the file metadata. After three consecutive read requests sent to node F of the same file, the "hit threshold" value is reached, and the system replication process is initiated. This replication consists on initially the namenode determines the closest datanode to the current consumer node (using equation 1 in section 3.1.1). In this scenario, the nearest node from the consumer has been selected as the node E. After the closest datanode has been found (i.e. node E), node E receives the extra replica of the file, as shown in Figure 9 via the traffic between t=107.5s and t=109s. At t=110.8s, when the fourth request is issued by the reading node, the transfer time is considerably reduced because the current data source node (E) is closer to that reading node than in the case of node F being the source.

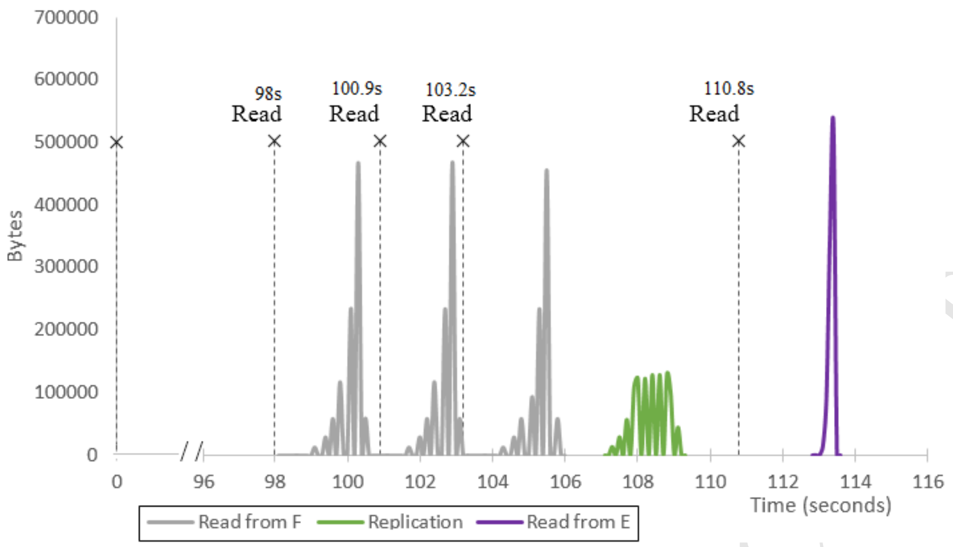


Fig. 9. Evaluation results of "Hit threshold" parameter.

Fairness in Energy Depletion (Test III)

In this test scenario, the gateway nodes in Figure 5 act as regular datanodes. All the nodes except A, B and C are instructed to write one file to the HDFS. Nodes A, B and C will each be receiving one file from each node in their edge network. Every datanode, expect for A, B and C, will then retrieve all the files in the filesystem. During this experience, we changed the parameter 'ω' of (1).

All datanodes have an initial battery level of 100%, and for every transfer between two nodes, their battery level decrease by 1%. At the end of the test, the battery level of each node has been collected. From our results (Figure 10), one can conclude that lower values of 'ω' (e.g. 25) have higher standard deviation (i.e. 19). Higher values of deviation indicate that the battery levels collected at the end of the test are very diverse. This means that some nodes finished the test with a rather low battery level, and others with high battery level. Therefore, using lower values of 'ω', some nodes that store popular data get penalized by draining out their battery. On the other hand, when 'ω'=3000, the standard deviation diminished from 19 to 10 (i.e. 47% reduction). In this way, when 'ω'=3000 there is a fairer energy depletion among the datanodes during files transfer.

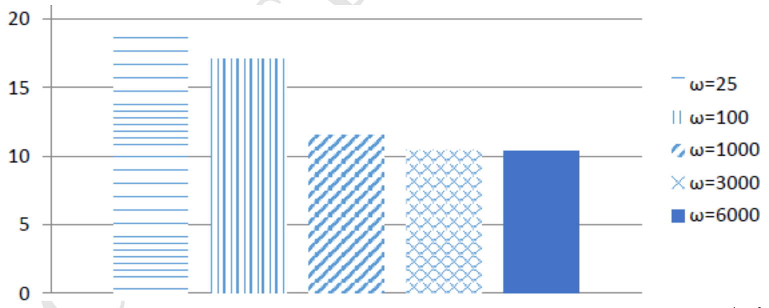


Fig. 10. Standard deviation of battery energy depletion for distinct values of 'ω'.

Processing Overhead on Namenode (Test IV)

From this test onwards, the gateway nodes in Figure 5 act as regular gateways. Each datanode is instructed to write a file of 1MB to the HDFS. Then, all datanodes retrieve all files, ten times. Each time every file is retrieved, counts as a loop attempt. This experiment used the following replication parameters: $\Delta tn=2s$; $\beta=1000$; $k=1.0$. From the obtained results (Figure 11), we have concluded the highest time (i.e. average value for the ten tries) the namenode spends on processing a file request was 10ms. In this situation, it can be neglected for the analysis on system performance. The maximum value for processing a file at the namenode occurs for the third loop attempt, essentially due to file replication.

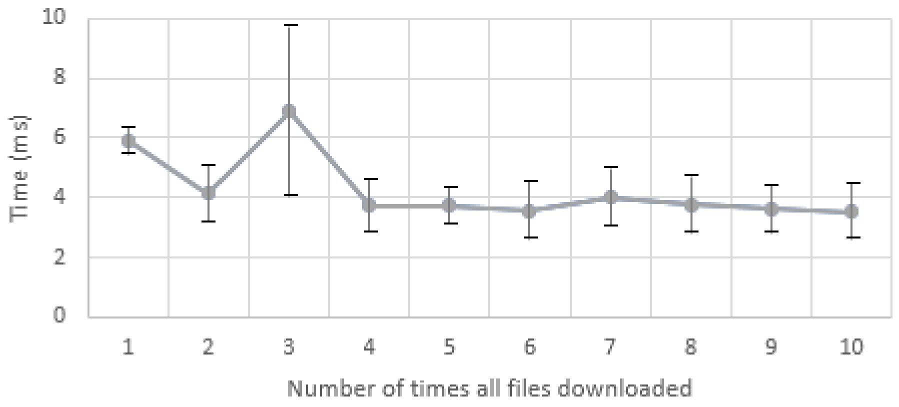


Fig. 11. Average time for processing a file per loop attempt at the namenode.

System's Responsiveness to Node Proximity (Test V)

In this experience, we want to study how the system responds in terms of speed in accessing data, when changing the node proximity in (2). We have assumed distinct values for the parameter 'k' and other replication parameters have been static, as follows: β=800, Δtn=2s and α=95%.

Figure 12 shows that for higher levels of 'k' (e.g. k=3.0) the average transfer time remained practically the same throughout the entire test. This means no file replication has occurred. This is expected, as the 'b' search radius associated to higher values of 'k', is big enough to create a circumference around each reader node that has always within that circumference the requested files. For values of 'k' smaller 2.0 it is visible a decrease on the average transfer time. This decrease on the file transfer time occurs because for smaller values of 'k' the system increases its responsiveness to node proximity. In this way, the system performs replication to closer nodes after the third reading request of each file.

System's Responsiveness to Popularity (Test VI)

This test studies how the system reacts to the popularity level. The parameter 'β' in (2) assumed diverse values, while keeping the other replication parameters as constants: k=1.4, Δtn=2s and 'α'=95%.

Figure 13 shows that lower values of 'β' give quicker access times to data. This happens because data are replicated in a more intensive way, and latency in accessing data is overall decreased. By giving higher values of 'β', the replication requirement regarding popularity are stricter, meaning data needs to have higher popularity levels to get replicated. This can be seen in Figure 13, for values of 'β'=1550, where download time remains unchanged, meaning there has hardly been any file replication.

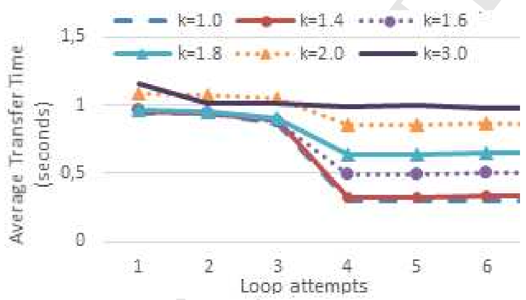


Fig. 12. System responsiveness to node proximity by varying 'k'.

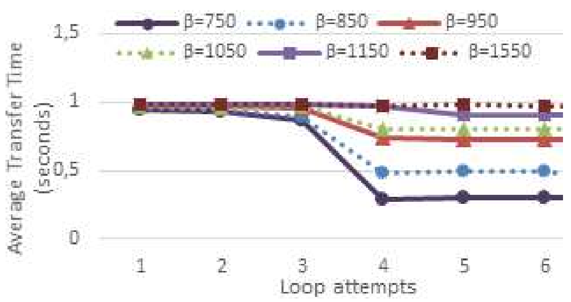


Fig. 13. System responsiveness to data popularity by varying 'β'.

Core to Edge Traffic Offloading (Test VII)

This test evaluates backhaul traffic offloading. In this test, each file has 10MB instead of 1MB, to emulate a scenario with "elephant flows". The incoming traffic has been monitored at the edge and gateways interfaces. The replication parameters were as follows: β=550; k=1.2; Δtn=2s; and α=99%.

Figure 14 shows that the core connections are heavily used up until around 360s, while the edge connections are not very used. This is expected as many file requests are for data belonging to other edge networks, so data need to pass through the core. At around 360s, data finish being replicated to the edge. From this point on, the core traffic increases very slowly, while the edge traffic increases at a much higher rate. This is because now, most data being requested are located on the same network as the reader node. Since the edge network (per user) has more bandwidth than the core, access speeds to data increase. The obtained results from this last test evidence a gradual traffic offloading from the network core to the network edge, according to the number of file requests from clients. The Table 6 presents a summary of the main obtained results from the testbed evaluation of our novel fog storage proposal.

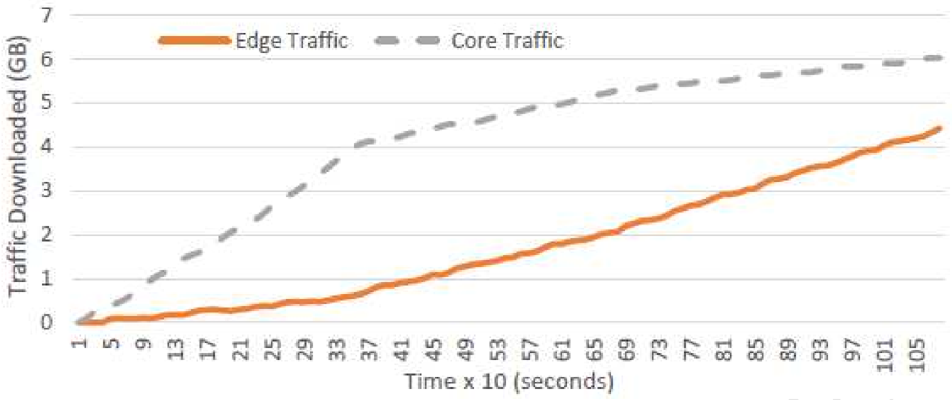


Fig. 14. Core to edge data offloading.

Table 6. Summary of the main obtained results from the evaluation tests of our novel fog storage system

Test	Main Result
I	Pharos showed better accuracy than CAN
II	Highlighted the system performance gains of using the "hit threshold" parameter and its associated replication functionality
III	Parameter 'ω' controls the fairness on the depleted energy per mobile node
IV	Namenode showed a satisfactory performance for managing the cluster-based system
V	Parameter "K" controls the system sensibility to the distance between nodes; after a parametrized value (in "hit threshold") of consecutive requests for the same file, this file is temporarily replicated to the clusters where the previous requests come from. Consequently, the file transfer time to each consumer is diminished.
VI	Parameter "β" controls system sensibility to the file (data) popularity. It decreases file transfer time between the elected data source node and each consumer.
VII	Our solution offloads traffic from the network core to the network edge organized in clusters. This offloading is made incrementally according to the data popularity.

5 CONCLUSION

We have proposed and implemented a solution that pools the storage resources of mobile devices and fog nodes for deploying an edge cloud. The content distribution is managed according to the node's energy level, node's localization, and spatio-temporal data popularity.

Results show that the proposed solution reduces the file transfer time experienced by end users. This better responsiveness occurs because data are stored closer to their consumers and are available in a more robust way. These responsiveness and robustness gains are due to novel aspects such as nodes' distance augmented by spatio-temporal data popularity as well as the available energy at nodes powered by battery. Results also evidence the backhaul links become less congested, as data get replicated to the edge, mitigating the backhaul bandwidth limitations per user that early proposals of MCC suffer from. In addition, the obtained results confirm the enhancement of both network's lifetime and data availability.

The current work can be extended to high-complexity scenarios, aggregating several network domains, where each domain has its own namenode running, as an example, within a SDN controller [17]. In this way, the migration of user files across distinct domains could be supported by a federation of SDN controllers with learning capabilities [56]. A relevant learning capability is to assess the impact of edge data processing on the quality of user experience. Other option to evolve the current work is to investigate a new edge approach that integrates storage, processing, and networking features.

REFERENCES

[1] L. Guan, X. Ke, M. Song, and J. Song, "A Survey of Research on Mobile Cloud Computing," in *2011 10th IEEE/ACIS International Conference on Computer and Information Science*, 2011, pp. 387–392.

[2] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 337–368, 2014.

[3] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the Cloud: Enabling Mobile Phones As Interfaces to Cloud Applications," in *Proceedings of the ACM/IFIP/USENIX 10th International Conference on Middleware*, 2009, pp. 83–102.

[4] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: Moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, 2016.

[5] M. Firdhous, O. Ghazali, and S. Hassan, "Fog Computing: Will it be the Future of Cloud Computing," in *Third Int Conf Informatics Appl*, 2014, pp. 8–15.

[6] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *J Netw Comput Appl*, vol. 98, pp. 27–42, 2017.

[7] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing cloudlet and mobile edge computing," in *Global Internet of Things Summit (GIoTS)*, 2017, pp. 1–6.

[8] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–34, Apr. 2018.

[9] Statista, "Number of smartphone users worldwide 2014-2020," *Web Page*, 2017. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed: 15-Sep-2017].

[10] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 2010, pp. 1–10.

[11] Y. Chen, Y. Xiong, X. Shi, B. Deng, and X. Li, "Pharos: A decentralized and hierarchical network coordinate system for internet distance prediction," in *GLOBECOM - IEEE Global Telecommunications Conference*, 2007, pp. 421–426.

[12] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M.- Thalmann, "Who, Where, when and What: Discover Spatio-temporal Topics for Twitter Users," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 605–613.

[13] F. Gabry, V. Bioglio, and I. Land, "On Energy-Efficient Edge Caching in Heterogeneous Networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3288–3298, 2016.

[14] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen, "Help your mobile applications with fog computing," in *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops, SECON Workshops 2015*, 2015, pp. 49–54.

[15] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache 'less for more' in information-centric networks (extended version)," *Comput. Commun.*, vol. 36, no. 7, pp. 758–770, Apr. 2013.

[16] F. Chen, K. Guo, J. Lin, and T. La Porta, "Intra-cloud lightning: Building CDNs in the cloud," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 433–441.

[17] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions," in *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, B. Di Martino, K.-C. Li, L. T. Yang, and A. Esposito, Eds. Singapore: Springer Singapore, 2018, pp. 103–130.

[18] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[19] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[20] Cisco, "Cisco Fog Computing Solutions: Unleash the Power of the Internet of Things," *White Paper*, 2015. [Online]. Available: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-solutions.pdf. [Accessed: 09-Jun-2018].

[21] E. Tordera, X. Masip-Bruin, J. Garcia-Almiana, A. Jukan, G.-J. Ren, J. Zhu, and J. Farre, "What is a Fog Node? A Tutorial on Current Concepts towards a Common Definition," *CoRR*, vol. abs/1611.0, 2016.

[22] E. Balevi and R. D. Gitlin, "Optimizing the Number of Fog Nodes for Cloud-Fog-Thing Networks," *IEEE Access*, vol. 6, pp. 11173–11183, 2018.

[23] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, and A. Al-Fuqaha, "Smart Cities: A Survey on Data Management, Security, and Enabling Technologies," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2456–2501, 2017.

[24] P. Neves, R. Calé, M. Costa, G. Gaspar, J. Alcaraz-Calero, Q. Wang, J. Nightingale, G. Bernini, G. Carrozzo, Á. Valdivieso, L. J. García Villalba, M. Barros, A. Gravas, J. Santos, R. Maia, and R. Preto, "Future mode of operations for 5G – The SELFNET approach enabled by SDN/NFV," *Comput. Stand. Interfaces*, vol. 54, pp. 229–246, Nov. 2017.

[25] L. Andrade, M. Serrano, and C. Prazeres, "The Data Interplay for the Fog of Things: A Transition to Edge Computing with IoT," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.

[26] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, “Heterogeneity in mobile cloud computing: Taxonomy and open challenges,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 369–392, 2014.

[27] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The Case for VM-Based Cloudlets in Mobile Computing,” *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[28] P. Mach and Z. Becvar, “Mobile Edge Computing: A Survey on Architecture and Computation Offloading,” *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[29] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, “Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge,” in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 9–16.

[30] Q. Wang, Z. Hu, M. Wang, and H. Liu, “CACTSE: Cloudlet aided cooperative terminals service environment for mobile proximity content delivery,” *China Commun.*, vol. 10, no. 6, pp. 47–59, 2013.

[31] E. E. Marinelli, “Hyrax : Cloud Computing on Mobile Devices using MapReduce,” CMU, 2009.

[32] S. Abolfazli, Z. Sanaei, M. Shiraz, and A. Gani, “MOMCC: Market-oriented architecture for Mobile Cloud Computing based on Service Oriented Architecture,” in *2012 1st IEEE International Conference on Communications in China Workshops (ICCC)*, 2012, pp. 8–13.

[33] Y. Lu, B. Zhou, L.-C. Tung, M. Gerla, A. Ramesh, and L. Nagaraja, “Energy-efficient content retrieval in mobile cloud,” in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing - MCC ’13*, 2013, pp. 1–21.

[34] R. K. Panta, R. Jana, F. Cheng, Y.-F. R. Chen, and V. A. Vaishampayan, “Phoenix: Storage Using an Autonomous Mobile Infrastructure,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1863–1873, 2013.

[35] A. Moon and H. Cho, “Energy Efficient Replication Extended Database State Machine in Mobile Ad Hoc Network,” in *IADIS International Conference on Applied Computing*, 2004, no. September 2015, pp. 224–228.

[36] C. Barca, C. Barca, C. Cucu, M. R. Gavriloaia, R. Vizireanu, O. Fratu, and S. Halunga, “A virtual cloud computing provider for mobile devices,” in *Proceedings of the 8th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2016*, 2016, pp. 1–4.

[37] R. Lacuesta, J. Lloret, S. Sendra, and L. Penalver, “Spontaneous ad hoc mobile cloud computing network,” *Sci. World J.*, vol. 2014, no. Article ID: 232419, pp. 1–19, 2014.

[38] R. Monteiro, J. Silva, J. Lourenço, and H. Paulino, “Decentralized Storage for Networks of Hand-held Devices,” in *Proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2015, pp. 299–300.

[39] A. L. Kakhandki, S. Hublikar, and Priyatamkumar, “Energy efficient selective hop selection optimization to maximize lifetime of wireless sensor network,” *Alexandria Eng. J.*, vol. 57, no. 2, pp. 711–718, Jun. 2018.

[40] H. Cheng, Z. Su, D. Zhang, J. Lloret, and Z. Yu, “Energy-Efficient Node Selection Algorithms with Correlation Optimization in Wireless Sensor Networks,” *Int. J. Distrib. Sens. Networks*, vol. 2014, no. 576573, pp. 1–14, Mar. 2014.

[41] Y. Li and S. Wang, “An Energy-Aware Edge Server Placement Algorithm in Mobile Edge Computing,” in *2018 IEEE International Conference on Edge Computing (EDGE)*, 2018, pp. 66–73.

[42] H.-S. Yeo, X.-S. Phang, H.-J. Lee, and H. Lim, “Leveraging client-side storage techniques for enhanced use of multiple consumer cloud storage services on resource-constrained mobile devices,” *J. Netw. Comput. Appl.*, vol. 43, pp. 142–156, Aug. 2014.

[43] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, “Cache in the air: exploiting content caching and delivery techniques for 5G systems,” *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[44] I. Abdullahi, S. Arif, and S. Hassan, “Survey on caching approaches in Information Centric Networking,” *J. Netw. Comput. Appl.*, vol. 56, pp. 48–59, Oct. 2015.

[45] F. Rebecchi, M. D. de Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti, “Data Offloading Techniques in Cellular Networks: A Survey,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 2, pp. 580–603, 2015.

[46] M. A. Maddah-Ali and U. Niesen, “Fundamental Limits of Caching,” *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[47] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable network,” in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM ’01*,

2001, vol. 31, no. 4, pp. 161–172.

[48] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 15–26, 2004.

[49] Y. Chen, X. Wang, C. Shi, E. K. Lua, X. Fu, B. Deng, and X. Li, "Phoenix: A weight-based network coordinate system using matrix factorization," *IEEE Trans. Netw. Serv. Manag.*, vol. 8, no. 4, pp. 334–347, 2011.

[50] C. Lumezanu, R. Baden, N. Spring, and B. Bhattacharjee, "Triangle Inequality and Routing Policy Violations in the Internet," in *Passive and Active Network Measurement*, 2009, pp. 45–54.

[51] B. Donnet, B. Gueye, M. A. Kaafar, and M. Ali Kaafar, "A survey on network Coordinates systems,design, and security," *IEEE Commun. Surv. Tutorials*, vol. 12, no. 4, pp. 488–503, 2010.

[52] S. He, H. Tian, and X. C. Lyu, "Edge Popularity Prediction Based on Social-Driven Propagation Dynamics," *IEEE Commun. Lett.*, vol. 21, no. 5, pp. 1027–1030, 2017.

[53] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, 2016.

[54] T. Yang, H. Pen, W. Li, D. Yuan, and A. Zomaya, "An Energy-efficient Storage Strategy for Cloud Datacenters based on Variable K-Coverage of a Hypergraph," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3344–3355, 2017.

[55] R. T. Kaushik and M. Bhandarkar, "GreenHDFS : Towards An Energy-Conserving , Storage-Efficient , Hybrid Hadoop Compute Cluster," in *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, 2010, pp. 1–9.

[56] G. Paschos, E. Batuˇ, I. Land, G. Caire, and M. Debbah, "Wireless Caching: Technical Misconceptions and Business Barriers," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 16–22, 2016.

[57] M. Ahmed, S. Traverso, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal Locality in Today's Content Caching: Why it Matters and How to Model it," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, May 2013.

[58] Z. Cheng, Z. Luan, Y. Meng, Y. Xu, D. Qian, A. Roy, N. Zhang, and G. Guan, "ERMS: An elastic replication management system for HDFS," in *Proceedings - 2012 IEEE International Conference on Cluster Computing Workshops, Cluster Workshops 2012*, 2012, pp. 32–40.

[59] M. Dias de Assunção, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *J. Netw. Comput. Appl.*, vol. 103, pp. 1–17, Feb. 2018.

[60] A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems," *J. Netw. Comput. Appl.*, vol. 59, pp. 208–218, Jan. 2016.

[61] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A Survey on Service Migration in Mobile Edge Computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[62] M. R. Anawar, S. Wang, M. Azam Zia, A. K. Jadoon, U. Akram, and S. Raza, "Fog Computing: An Overview of Big IoT Data Analytics," *Wirel. Commun. Mob. Comput.*, vol. 2018, pp. 1–22, May 2018.

[63] S. Ardon, A. Bagchi, A. Mahanti, A. Ruhela, A. Seth, R. M. Tripathy, and S. Triukose, "Spatio-temporal and Events Based Analysis of Topic Popularity in Twitter," in *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, 2013, pp. 219–228.

[64] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, 2009, pp. 280–293.

José André R. S. Moura received the B.S. degree for Electronics and Telecommunications from the Universidade de Aveiro (UA) – Aveiro University, in 1989. From 1989 to 2000, he worked as a project manager for assembling and testing Supervisory Control and Data Acquisition (SCADA) systems on EFACEC Sistemas Electrónica, SA. From 2000 to 2002, he worked as a researcher in the VESPER project developing the Virtual Home Environment (VHE) paradigm on INESC-Porto. Since 2001, he has been working on computer networks in ISCTE-IUL. He finished his PhD thesis at Lancaster University (UK) in 2011, focusing on managing the mobile access in heterogeneous wireless networks. Since 2011, he is an assistant professor on ISCTE-IUL. He is currently a researcher with IT-IUL, Lisbon, Portugal. His research interests include Wireless Networks, Game Theory, Virtualization, and Software-Defined Networking.