J Supercomput
DOI 10.1007/s11227-017-2082-y

FLAPS: bandwidth and delay-efficient distributed data searching in Fog-supported P2P content delivery networks

Mohammad Shojafar[1,2] · **Zahra Pooranian**[3] · **Paola G. Vinueza Naranjo**[1] · **Enzo Baccarelli**[1]

Abstract Due to the growing interest for multimedia contents by mobile users, designing bandwidth and delay-efficient distributed algorithms for data searching over wireless (possibly, mobile) "ad hoc" Peer-to-Peer (P2P) content Delivery Networks (CDNs) is a topic of current interest. This is mainly due to the limited computing-plus-communication resources featuring state-of-the-art wireless P2P CDNs. In principle, an effective means to cope with this limitation is to empower traditional P2P CDNs by distributed Fog nodes. Motivated by this consideration, the goal of this paper is twofold. First, we propose and describe the main building blocks of a hybrid (e.g., mixed infrastructure and "ad hoc") Fog-supported P2P architecture for wireless content delivery, namely, the Fog-Caching P2P architecture. It exploits the topological (possibly, time varying) information locally available at the serving Fog nodes, in order to speed up the data searching operations performed by the served peers. Second, we propose a bandwidth and delay-efficient, distributed and adaptive probabilistic

✉ Mohammad Shojafar
mohammad.shojafar@cnit.it; mohammad.shojafar@uniroma1.it

Zahra Pooranian
zahra@math.unipd.it

Paola G. Vinueza Naranjo
paola.vinueza@uniroma1.it

Enzo Baccarelli
enzo.baccarelli@uniroma1.it

1 Department of Information Engineering, Electronic and Telecommunication, Sapienza University of Rome, Rome, Italy

2 Center of National Consortium Inter-universities in Telecommunication (CNIT), University of Rome – Tor Vergata, Via del Politecnico, 1, 00133 Rome, Italy

3 Department of Mathematic, University of Padova, Padova, Italy

Published online: 07 June 2017

search algorithm, that relies on the learning automata paradigm, e.g., the Fog-supported Learning Automata Adaptive Probabilistic Search (FLAPS) algorithm. The main feature of the FLAPS algorithm is the exploitation of the local topology information provided by the serving Fog nodes and the current status of the collaborating peers, in order to run a suitably distributed reinforcement algorithm for the adaptive discovery of peer-to-peer and peer-to-fog minimum-hop routes. The performance of the proposed FLAPS algorithm is numerically evaluated in terms of Success Rate, Hit-per-Query, Message-per-Query, Response Delay and Message Duplication Factor over a number of randomly generated benchmark CDN topologies. Furthermore, in order to corroborate the actual effectiveness of the FLAPS algorithm, extensive performance comparisons are carried out with some state-of-the-art searching algorithms, namely the Adaptive Probabilistic Search, Improved Adaptive Probabilistic Search and the Random Walk algorithms.

Keywords Fog computing · Content delivery networks (CDNs) · Fog-Caching P2P (FCP2P) · Adaptive probabilistic data search (APS) · Learning automata (LA) · Reinforced Q-learning · TCP/IP overlay networks

1 Introduction

A recent forecast by Cisco points out that mobile multimedia content delivery will cover around 72% of the overall mobile network data traffic by 2020 [1]. The main feature of this service is that the requested contents are densely concentrated in the space so that some of them are asynchronously requested several times by multiple (e.g., possibly, collaborative) clients equipped with bandwidth and computing resource-limited wireless devices. Motivated by this consideration, wireless Fog-Caching is gaining momentum under the umbrella of the emerging 5G networking paradigm [2,3]. By design, the Fog-Caching paradigm (also referred to as Femto-Caching) aims at avoiding resource-wasteful content duplication by locally caching the most popular data at the proximate serving Fog nodes. This paradigm is, indeed, gaining extensive attention, due to its intrinsic feature of reducing both content acquisition latency and network backhaul data traffic [3]. However, since Fog-Caching aims at moving locally popular contents to Fog nodes and Fog nodes are, by design, small-size data centers, a direct utilization of the client–server paradigm for content delivery may overload the Fog nodes, especially when the spatial distribution of the querying clients is unbalanced [4]. In such cases, the workload of the caching Fog nodes may be alleviated by allowing proximate clients to act as peers, in order to balance the distribution of the queried data over both collaborating peer nodes and serving Fog nodes.

1.1 Application scenario and main contributions of the paper

Motivated by these considerations and by referring to the application scenario of Fig. 1, the objective of this paper is threefold. First, we propose a hybrid architecture (e.g., the FCP2P architecture) for wireless Fog-supported CDNs. This architecture

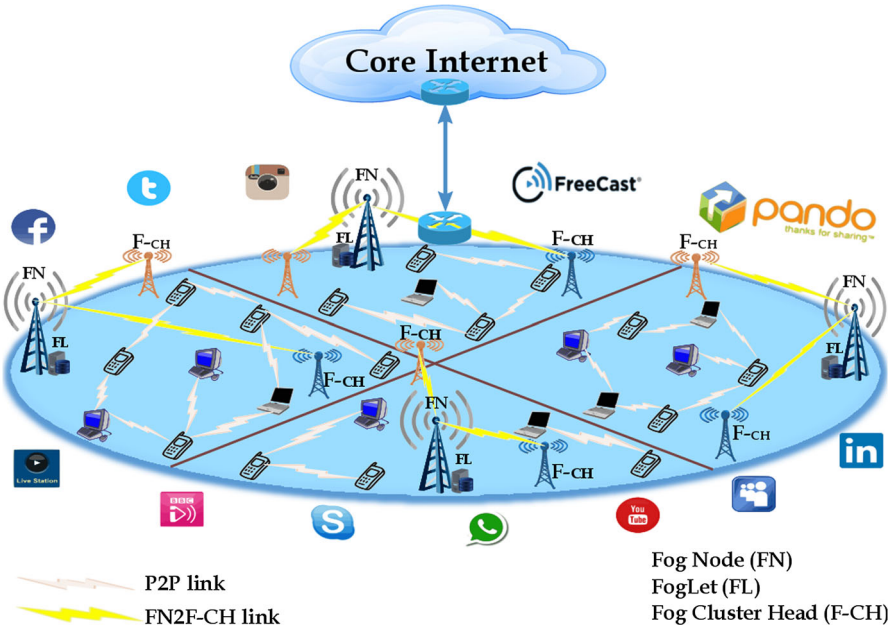

Fig. 1 The proposed FCP2P technological platform

suitably integrates the complementary paradigms of the (infrastructure) Fog-Caching and the ("ad hoc") P2P, in order to sustain a transport-layer overlay network built up by end-to-end reliable TCP/IP connections. Second, by exploiting the local information provided by the serving Fog nodes about the inter-peer distances and peer-to-peer link gains, we design and test a bandwidth and delay-efficient distributed route-discovering algorithm, namely, the Fog-supported Learning Automata Adaptive Probabilistic Search (FLAPS) algorithm. It aims at dynamically discovering minimum-hop P2P and F2P routes over the currently built up overlay network. The goal is to forward the query messages generated by the requiring peer nodes to suitable caching nodes at the minimum bandwidth cost. Specifically, the main distinguishing features of the proposed FLAPS are the following ones: (i) it empowers each network node by a Learning Automata (LA) agent, in order to implement the routing process in an adaptive and distributed way; (ii) it implements a suitable reward-penalty mechanism for adaptively improving the actions performed by each LA; and (iii) it accounts for the current values of the success rate, message duplication factor, response delay and per-query message hit, in order to adaptively tune the reward and penalty parameters. Third, we numerically test the performance of the proposed FLAPS over a number of benchmark CDN topologies. Finally, we compare the obtained FLAPS performance with the corresponding ones of some state-of-the-art search algorithms, namely, the Adaptive Probabilistic Search (APS) [5], the Improved APS (IAPS) [6], and the Random Walk (RW)-based [7] algorithms.

1.2 Related work and comparisons

An examination of the (recent) related work corroborates the conclusion that research on CDNs is till now pursued by moving along two main parallel (e.g., nearly disjoint) directions.

Specifically, a first (more recent) research direction focuses on the architectural design of emerging infrastructure (e.g., structured) virtualized Fog-Caching networks for low-latency content delivery [8–12]. Toward this end, the authors of [8] introduce an architecture for the optimized placement of device clones (e.g., Virtual Machines (VMs)) over multiple spatially distributed serving nodes. The target is to improve the Quality of Experience (QoE) perceived by the clients while achieving load balancing over the available serving nodes. Similar VM-allocation problems are also considered in [9], in order to reduce node energy consumptions and in [10] under different VM scheduling disciplines. More recently, the authors of [11] afford the joint optimization of data caching over multiple computing nodes and data retrieving by clients. The resulting joint algorithm in [11] minimizes both the data acquisition delay and the data placement cost. In [12], the authors consider a multi-tier network of computing nodes, in which multiple heterogeneous remote servers serve multiple clients, while each client can offload/retrieve data to/from multiple proximate Fog nodes. The algorithm proposed in [12] maximizes the network-wide revenue under both storage and communication constraints. Overall, like our contribution, all the mentioned papers promote the utilization of the Fog paradigm as a viable means for local data caching. However, unlike our contribution, these papers do not consider the integration of the Fog infrastructure with the P2P one.

A second (more consolidated) research direction focuses on the design of distributed search algorithms for data retrieving over "ad hoc" (e.g., unstructured) P2P CDNs [5–7, 13–21]. Specifically, the (somewhat traditional) approach pursued by Gnutella [13] assumes that each peer is joint to an arbitrary set of neighbors and the search process relies on a blind (e.g., context and content oblivious) uncontrolled flooding of query messages over all neighbors. Although quite simple to implement, the uncontrolled flooding is prone to storm phenomena, and this triggered further research on improved context and content-aware search strategies for P2P applications. They moved along several directions, that exploit the statistical and topological properties of the random walks [7], small-world graphs [14] and peer heterogeneity [15], in order to suitably driving the search process. "Ad hoc" solutions for coping with churn phenomena induced by peer mobility are presented in [16] and [17]. After recognizing that searching for the best path over a direct graph is typically an NP-hard problem, some biology-inspired meta-heuristics have been also applied for performing data search over P2P CDNs, namely, the Ant Colony-based (AntNet) meta-heuristic in [20] and the distributed query routing by Ant Colony (SemAnt) in [21]. Roughly speaking, both these approaches use local information and reputation-based learning methods, in order to bias the path-search process toward the most promising candidates. However, the space of the solutions spanned by these approaches is large, and, then, these methods typically suffer by slow convergence. Lastly, some adaptive probabilistic search approaches have been proposed (e.g., the APS in [5] and the IAPS in [6]), in order to reduce the search space. The common feature of the APS and IAPS approaches is that they drastically

reduce the search space by implementing and updating suitable score functions, in order to store the most promising searching paths. Overall, like our contribution, all these papers deal with the common general problem of the design of resource-efficient distributed data search algorithms over "ad hoc" P2P overlay networks.

However, we point out that the main goal of this paper is to design and test the performance of a self-tuning distributed route discovery algorithm, that exploits the Q-learning reinforcement paradigm, in order to adaptively build up minimum-response-time P2P transport routes over heterogeneous Fog-supported time-varying wireless CDNs. Specifically, the proposed learning-based route discovery algorithm (e.g., the FLAPS one) self-selects (in an iterative way) the best set of actions to be implemented to build up the target P2P routes by using some suitable learning functions. These last dynamically reward/penalize the previously performed actions on the basis of the feedback messages received from the neighbors of the nodes that participate to the current routes. Interestingly, in the FLAPS algorithm, the feedback messages carry out information about: (i) the inter-peer distances and (ii) the round-trip delays and throughput of the inter-peer TCP/IP transport connections, and this information is gathered and broadcast by the supporting Fog nodes.

The rest of the paper is organized as follows. In Sect. 2, we present the main functional blocks of the proposed hybrid FCP2P architecture for wireless content caching and retrieving. Afterward, in Sect. 3, we detail the proposed route discovery FLAPS algorithm, while Sect. 4 investigates about some related implementation aspects. After detailing the tested application scenarios and performance metrics in Sect. 5, Sect. 6 presents the performance of the FLAPS algorithm and compares it against the corresponding ones of the APS, IAPS and RW algorithms. Finally, in Sect. 7, we summarize the main attained results and give some hints for future research.

2 The proposed FCP2P hybrid networking architecture for data searching over CDNs

The basic building blocks of the proposed FCP2P system architecture is sketched in Fig. 1. It leverages Device-to-Device (D2D) and Fog-to-Peer (F2P) end-to-end TCP/IP connections, in order to dynamically build up P2P and F2P transport-layer links, respectively. So doing, FCP2P gives rise to (possibly, multiple and heterogeneous) hierarchical overlay networks, that are composed by (see Fig. 1): (i) clusters of nearby collaborating peers; (ii) Fog-Cluster Heads (F-CHs), that act as local gateways for the connected peers and (iii) serving Fog nodes. Each Fog node acts as an Internet gateway for the served F-CHs. Furthermore, since it is equipped with a small-size data center (e.g., a Fog-Let (FL); see Fig. 1), it provides also structured caching and computing support to the connected F-CHs and peers. A Fog node may also directly acts as a cluster header when it is surrounded by a number of nearby peers. According to the P2P paradigm, each peer node may simultaneously act as server and client for the nearby peers and may forward both query messages and queried data.

In Fig. 1 each peer may connect to other proximate peers by dynamically setting up D2D connections [4]. According to the emerging 5G vision [3] and Fog paradigm [22], these connections may employ different (possibly, heterogeneous) transmission tech-

nologies at the physical layer, like, for example, short-range Bluetooth technologies, medium-range WiFi technologies, or even long-range cellular 3G/4G technologies.

After performing node clustering (see Fig. 1), the resulting overlay network is described by an undirected graph: $\mathcal{G} \triangleq <\mathcal{V}, \mathcal{E}>$, where $\mathcal{V}$ (resp., $\mathcal{E}$) is the set of the graph nodes (resp., graph edges). Each node $v \in \mathcal{V}$ represents a peer node, an F-CH node, or a Fog node, while each edge $e \in \mathcal{E}$ indicates a transport-layer TCP/IP bi-directional connection. Furthermore, a node label (resp., edge label) indicates the overall up/downstream bandwidth available at the node (resp., the transport capacity of the corresponding TCP/IP two-way connection). In the sequel, the set $\mathcal{V}_i \subseteq \mathcal{V}, i \in \mathcal{V}$, denotes the set of peer neighbors of the i-th wireless peer node wp_i.

2.1 File tables and learning automata

Each peer node may directly connect to other peer nodes or send its request to the nearest Fog node, in order to forward the received request to other Fog nodes or F-CH nodes (see Fig. 1). For this purpose, each F-CH node serves some peers and supports the Fog node traffic when the traffic load increases. An F-CH node also performs synchronization functions. The FLAPS algorithm is carried out by the F-CHs in a distributed manner, in order to increase the response rate and decrease latency such as elaborated in [23]. Besides, each FL of Fig. 1 is equipped with the list of the neighbor peers. In order to increase the response rate and decrease the latency, the FLAPS algorithm enumerates the generated per-FN and peer-file type queries and broadcasts this information over the network. For this purpose, the FLAP algorithm sorts the score probability of each active node in a not increasing fashion.

Each FN is equipped with the list of the peers that are successfully traversed, in order to find the searched file. We introduce a simple assumption here: we select the nearest peers (i.e., in terms of hop number from the query requester) that host the requested file. We named these nearest peers as the *responder* peers. By definition, a query answer consists of sending a file to the requester peer. In order to forward the query and reach the most suitable FN, we resort to a learning automata approach. It implements an adaptive Q-reinforcement learning mechanism, (e.g., the FLAP), that can select the best route actions on the basis of the feedback messages received from the FCP2P environment. For this purpose, each peer is equipped with a list of the stored information (or files) as in Table 1. It can deliver these files to its neighbors. Table 1 reports the instance of a *Query Table (QT)*. It stores the file information which is available at the hosting FN. Specifically, each query table comprises four columns, that report the file type, format, number of file copies, and probability scores. These last indicate the storing probabilities of file types at time slot n and at each peer.

Table 1 Query table of the status of each node at time slot n

File type	m	N_{km}	Score probability (sp)
Video	$*$.mp4	100	sp_{k1}
Audio	$*$.amc	200	sp_{k2}
Text	$*$.txt	300	sp_{k3}

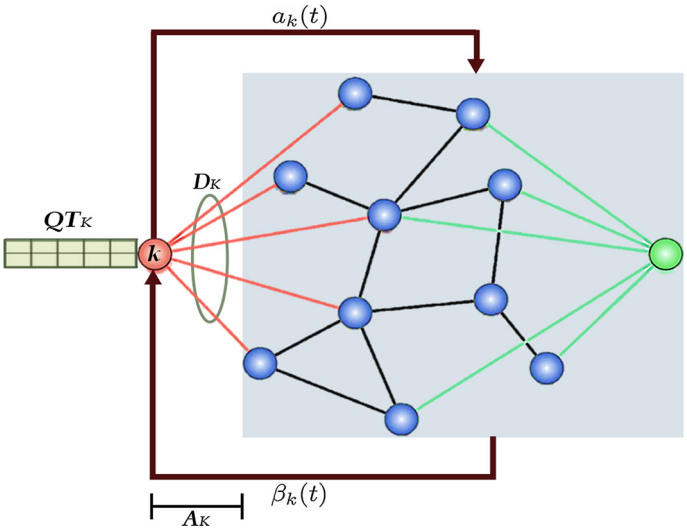

Fig. 2 The relationship between LA at the k-th peer node A_k and the rest of the network. QT, Query table; D_k, degree of the k-th peer node; $a_k(t)$, set of actions of k-th peer node at slot t; and $\beta_k(t)$, set of the feedback messages from the network in response to the action $a_k(t)$ at time slot t

As is shown in Fig. 2, each LA performs a set of actions toward its neighbors and receives their feedback, in order to decide the set of actions at the next iteration. The set of learning automata is denoted by: $<A, a>$, where $A \equiv \{A_1, A_2, \ldots, A_N\}$ is the set of learning automata corresponding to the set of nodes (e.g., N is the nodes number), and $a \equiv \{a_1, a_2, \ldots, a_N\}$ denotes the set of actions, so that $a_k \equiv \{a_k^1, a_k^2, \ldots, a_k^{r_k}\}$ is the list of actions available at the learning automata A_k, while r_k is their number. The actual value of r_k depends on the connection degree of the k-th node. Specifically, we consider an LA at the k-th node A_k that runs over n time slots (e.g., $t \in \{1, \ldots, n\}$) and produce r_k actions a_k and r_k feedback messages. At each iteration t, LA can select one of the action for the next iteration based on their probabilities and punishes/rewards them for the next iteration. In the designed LA algorithm, the value of a reward a (if a positive feedback is received) or penalty b (if a negative feedback is received) is constant and unique for each action. The two relationships used for updating the reward and penalty probability values are detailed in Eqs. (1) and (2), respectively. Specifically, the reward rate increases in Eq. (1) the probability to take the action. In a dual way, the punishment rate Eq. (2) reduces the probability of the action.

$$
\begin{aligned}
& p_{kj}(t+1) = (1-a) \times p_{kj}(t), \\
& p_{ki}(t+1) = p_{ki}(t) + a \times [1 - p_{ki}(t)] + \mathrm{sp}_{km}, \\
& \forall k, j; \quad j \neq i, \quad i \in \{1, \ldots, r_k\}, \quad k \in \{1, \ldots, N\};
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
& p_{kj}(t+1) = \frac{b}{r_k - 1} + (1-b) \times p_{kj}(t), \\
& p_{ki}(t+1) = (1-b) \times p_{ki}(t) + \mathrm{sp}_{km}, \\
& \forall k, j; \quad j \neq i, \quad i \in \{1, \ldots, r_k\}, \quad k \in \{1, \ldots, N\}.
\end{aligned} \tag{2}
$$

In Eqs. (1), (2) p_{ki} is the probability of the i-th action at LA A_k. In the FLAPS framework, the set $\beta_k(t)$ of the environmental feedback received by the k-th node of Fig. 2 at slot t comprises the current numbers of hops (or, if available, the current Round-Trip-Times (RTTs)) and the current throughput of all TCP/IP connections sustained by the neighbors of node k, while the corresponding action set $a_k(t)$ collects the routing actions currently available at node k. The components of the sets $\{\beta_k(t)\}$ are periodically gathered by the serving FNs of Fig. 1 and broadcast to the served nodes. The set $a_k(t)$ is periodically self-tuned by the k-th node by applying the reward/punishment mechanism of Eqs. (1) and (2). Therefore, in order to traverse a routing path, we need to select its next hop based on the aforementioned probabilities and punish the other possible paths.

At the beginning, the table entries are populated by the same number of files [such as N_{k1} number of file type 1 (e.g., *.mp4), N_{k2} number of file type 2 (e.g., *.amc), etc.]. The score probability sp_{km} is the number of occurrences of the file type m at node k, e.g., the chance that node k hosts file type m. Two events that may happen in FCP2P networks are *leaving* and *joining* nodes from/to the network. A uniform and equalized distribution of entries in the query table is assumed before the separation of a node from the network. This guarantees that the attenuated bloom filters (i.e., data structures that exist in each node of the system) [24] in its query table remain accessible, while the node is offline. The node should be synchronized to the current network status when it rejoins the FCP2P network. For this purpose, it is needed to firstly find the set of its active neighbors. For accomplishing this task, each node may use handshaking messages [25], so to update its query table by sending updating requests to its neighbors. Without loss of generality, let d_k be the number of active neighbors of the k-th peer node, with d_k less than the node degree D_k. Then, the k-th peer begins to send the query messages over d_k routing paths using the highest rewarded actions. One of the tasks of A_k is to update the d_k at k-th peer, before starting to send the query messages. Besides, in FLAPS, the action set of each learning automata consists to select one of the neighbors based on their probabilities. The starting probability of finding the j-th file type for each active path that can be traversed from k-th peer node is defined as: $p_{kj}(1) = \frac{1}{d_k}$, at $t = 1$. Furthermore, the score probability of k-th peer node at time slot n for the m-th file type is evaluated on the basis of the previous probabilities of the neighbors that host the m-th file type. These values can be calculated by normalizing the received queries and their LA probabilities as follows:

$$\mathrm{sp}_{km}(n) \equiv \begin{cases} \sum_{t=1}^{n} \sum_{l=1}^{N_{km}} p_{km}(t)/N_{km}, & \text{if } N_{km} \neq 0, \quad \forall k = \{1, \ldots, N\}, \\ 0, & \text{if } N_{km} = 0, \quad \forall k = \{1, \ldots, N\}, \end{cases} \tag{3}$$

where: (i) N_{km} is the number of m-th file type copies at node k; (ii) n is the number of time slots for each A_k; and (iii) $p_{km}(t)$ is the action probability at the k-th peer node at time slot t for the m-th file type. In addition, the FLAPS algorithm stores the score of each file based on the results of the former query searches. The normalization of Eq. (3) forces the learning algorithm to be consistent over all nodes and suitably redirects. It forces to give priority to some queries to reach the data and decreases the missed data and increases the hit rates. Hence, by proceeding backward from the *responder* node to the *requester* one, each learning automata at the visited nodes punishes the own

not-visited neighbors. If there exist multiple paths from the responder to the requester, the shortest path is selected and all nodes on the shortest path are rewarded by the same quota, while the other not-visited nodes will be punished by the same quota. However, if the requested file is not found or the selected node has been previously visited, FLAPS punishes by b units the not-visited neighbors.

3 The architecture of the FLAPS algorithm

The proposed FLAPS algorithm comprises two phases, namely, the search phase and the neighbor selection phase. The latter also includes the mentioned reward/punishment mechanism, in order to promote inter-peer collaboration.

3.1 The search phase

During the search phase, when a node receives a request message for a file, it searches for the requested file in its local database. If the node finds the file in the own database, it sends a response message to the requiring node by forwarding a routing table. In order to limit the response delay, we need to keep the response rate high. For this purpose, FLAP updates the score probability of m-th file type and reward/punishes the actions of each visited peers by applying the LA rules of Eqs. (1), (2). If the requested file is not found, the search phase for the file will continue by selecting the k neighbors (e.g., k walkers) with the highest probability values over all neighbors. If at least one neighbor retrieves the file in own query table, the neighbor node will be awarded. The search ends when the file is found or the Time-To-Leave (TTL), (e.g., is the maximum number of hops crossed by each query message) expired. FLAP uses the Internet Control Message Protocol (ICMP) signaling messages to escape network crowding and avoid a requested service not being available or neighbor peers being unable to be reached. Therefore, only the available peers are guaranteed to be visited by the FLAP during the search phase. As a consequence, if the requested file does not exist, the node forwards the request to its neighbors through its query table (Table 1). So doing, FLAP by-passes the occurrence of infinite loops.

3.2 Neighbor selection phase

Selected neighbors propagate the query messages, in order to locate the requested file. After receiving the feedback from the environment, if $k/2$ of selected neighbors find the file, the probability values of the selected neighbors increase, otherwise a punished probability value is updated by the FLAPS. The following mechanism is used to better balance the query traffic. Before successfully answering a query, a peer p checks if any of its neighbors has the requested file. In the affirmative case, it delegates the responsibility to answer the query to the peer with the smallest in-degree over those that host the queried file. Otherwise, the peer directly sends the file to the requester. However, there is a good chance that some of the neighbors of a peer also have the same files. Therefore, the less loaded peer is forced to serve a part of the load. Neighbor selection for request forwarding is based on the information stored in the node query

tables (see Table 1), e.g., the number of effective demands (file score probabilities), file types and file type probabilities. The requiring node selects the neighbors with the highest scores for the requested file type. Lastly, since each node has limited counting capacity and input bandwidth, FLAPS needs to implement a control policy, in order to manage the incoming traffic and limit the per-node received query messages. For this purpose, a least-recently-used (LRU) policy is used by FLAP to regulate the input traffic at each node. This policy avoids the chunk of some meaningless messages by ICMP and, thus, enables the network's capacity to serve meaningful new messages in the near future. Algorithm 1 reports the pseudo-code of the FLAP algorithm.

4 Implementation aspects and complexity

In this section, we focus on the main system issues that may impact on the actual implementation of the proposed FLAPS algorithm and the supporting FCP2P networked architecture of Fig. 1.

4.1 Medium access

The FLAPS algorithm runs atop an overlay network built up by transport-layer TCP/IP connections, and this leaves room for various options at the MAC layer. However, since in the considered application scenario of Fig. 1: (i) the peer nodes may be mobile and they are typically bandwidth and energy-limited; and (ii) the Fog nodes could be exploited for performing peers' coordination, we believe that the reservation-based cognitive Orthogonal TDMA (OTDMA) medium access protocol recently presented in [26] could be a viable choice. Figure 3 reports the basic elements of the corresponding frame structure.

In a nutshell, the OTDMA protocol in [26] operates as follows:

(i) the time axis is partitioned into alternating up and downstream frames. In each cluster, the time duration of each frame is adaptively set by the corresponding Fog node on the basis of the traffic (e.g., number of query messages and volume of required data) to be transported;

(ii) at the beginning of each up frame, the acceding peers send in upstream to the corresponding coordinating Fog nodes Request-to-Access (RA) messages. Each RA message specifies the volume of the data to be sent, the residual energy of the peer node and the needed average energy per transmitted bit (e.g., the average level of the fading of the utilized link). Since the RA messages sent in upstream may collide, a suitable cognitive scheduling approach is developed in

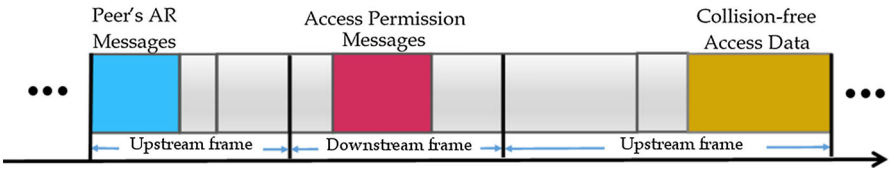

Fig. 3 Up/downstream frame structure of the reservation-based OTDMA protocol in [26]

[26], in order to guarantee a per-peer maximum collision rate and, then, a per-peer minimum access throughput;

(iii) after gathering all the (collision-free) RA messages, the coordinating Fog node allocates to the requiring peers non-overlapping (e.g., orthogonal) transmit time windows. The duration of each time window is optimized on the basis of the information carried out by the corresponding RA messages, in order to maximize the overall network-wide bandwidth usage. The notification of the allocated time windows is broadcast in downstream to all connected peers;

(iv) finally, during the next upstream frame, each peer transmits its data over the assigned time window without experiencing data collision.

Before proceeding, we point out that, in the considered FCP2P scenario of Fig. 1, there are three main reasons for adopting the (aforementioned) reservation-based OTDMA protocol in place of pure contention-based Carrier Sensing Multiple Access/Collision Avoidance (CSMA/CA) MAC protocols. First, CSMA/CA-based MAC protocols cannot guarantee a per-peer minimum access throughput. Second, CSMA/CA-based protocols are prone to data collisions, that, in turn, waste both bandwidth and energy resources. On the contrary, the reservation-based OTDMA in [26] guarantees collision-free data access and limits also the collision rate of the RA messages. Third, the hybrid channel control access working mode of the legacy *IEEE802.11e* standard already supports reservation-based TDMA scheduling mechanisms, so that it already provides the basic functions for the implementation of the OTDMA protocol in [26].

4.2 Node clustering and construction of the overlay network

The proposed FLAPS algorithm runs atop an overlay network whose (possibly, time varying) topology is obtained by performing a suitable clustering of the peer nodes. For this purpose, the recently proposed Prolong Stable Election Protocol (P-SEP) in [27] for node clustering in Fog- supported wireless sensor networks looks as an attractive solution. Shortly, P-SEP implements a suitable iterative greedy-type clustering algorithm, whose target is the maximization of the peer lifetime. In order to attain this goal, P-SEP exploits the information on both the inter-peer distances and peer-to-peer link gains that is broadcast by the serving Fog nodes. At the core of P-SEP, there is the selection of a number of F-CHs, in order to attain load balancing, reduction of the energy consumed by the peer nodes and stretching of the lifetime of the (energy limited) peers. In a nutshell, the main features of P-SEP may be so summarized [27]: (i) P-SEP accounts for the sizes of the built up clusters by optimizing the average F-CH-to-Peer distances; (ii) P-SEP accounts for the energy budget of each peer, in order to maximize the network-wide lifetime; (iii) P-SEP directly outputs the (aforementioned) graph: $\mathcal{G} \triangleq <\mathcal{V}, \mathcal{E}>$ of the resulting overlay network composed of the build up TCP/IP connections.

4.3 Peer time synchronization

In our framework, the goal of the Peer Time Synchronization (PTS) is twofold. First, it allows the proposed FLAPS algorithm to proceed in a round basic way [see the n

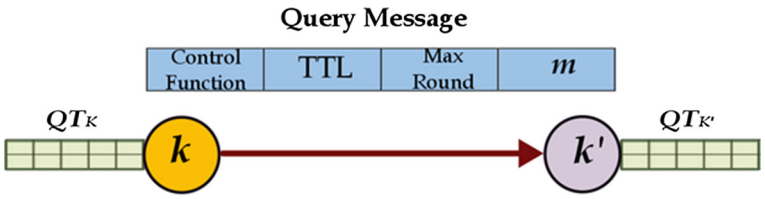

Fig. 4 Implementation of the message passing mechanism under FLAP. *QT* Query table

index in Eq. (3)]. Second, it allows implementing the framing structure of Fig. 3 for TDMA-based MAC protocols. For this purpose, as in [28], the distributed Timing Synchronization Function (TSF) of the (aforementioned) hybrid channel control access working mode of the *IEEE802.11e* could be used as a basic building block. Specifically, the approach developed in [28] does not require the utilization of network-wide master clocks, but it exploits the periodic transmission of beacons performed by the *IEEE802.11e* protocol, in order to allow the inter-peer tuning of the timing information. Hence, according to the *IEEE802.11e*-compliant approach pursued in [28], beacon packets may be directly included at the beginning of each frame of Fig. 3, in order to attain distributed cluster-wide peer synchronization.

4.4 Format of the query messages and message passing

According to Fig. 2, a requester peer node (e.g., node k) sends the query message to the selected destination. Figure 4 shows the inter-node message passing mechanism implemented by FLAP. The query message includes the TTL, the MaxRound, the ordered list of the nodes visited by the query message, and the requested file identification.

At this regard, we point out that FLAP exploits the Round-Trip-Time (RTT) measurements and SYN/SYNACK signaling segments already provided by state-of-the-art TCP suites, in order to attain the node-to-node synchronization needed by the message passing mechanism of Fig. 4 without introducing further protocol overhead.

4.5 Distributed implementation of the designed reinforcement learning algorithm and implementation complexity

In this subsection, we focus on the implementation of the reinforcement learning algorithm implemented by the proposed FLAP algorithm.

According to Algorithm 1, at first, FLAP equips the CDN graph with the list of LA for each active node. Then, it uniformly distributes the files over the nodes and gives primary score probability to each file type of the peer nodes and names them as sp^0 and N_{km}^0, respectively. Then, FLAP starts the main loop that performs the query search for the specific file $q(m)$ over the CDN. In the searching phase, FLAP needs to search, at first, in the requester local database $\mathrm{QT}_{\mathrm{req}}$, and, if the file is found, it sends back it to the requester. Otherwise, FLAP starts the requester LA A_{req} by a recursive search over its neighbors and updates A_{req} and $\mathrm{sp}_{\mathrm{req\,m}}$. The requester gives

specific TTL and, while traversing, FLAP puts the visited peers into the queue qq. If FLAPS retrieves the file before TTL expiring, it rewards all the actions and updates their score probabilities for that specific file type, while it penalizes the other paths. Finally, FLAPS punishes the double visited peers and updates the corresponding score probabilities.

Algorithm 1 FLAPS pseudo-code

INPUT: $G =< V, E >$, TTL
OUTPUT: Found node of $q(m)$
1: **while** TTL and MaxRound are not attained **do**
Search $q(m)$ inside req database;
2: **if** $q(m) \in \mathrm{QT}_{\mathrm{req}}$ **then**
Update A_{req}, $\mathrm{sp}_{\mathrm{req\,m}}$;
3: **else** Find d_{req} neighbors by using Eqs. (1) and (2);
req=found(neighbor to search);
Recursively search QT_{req} for $q(m)$ and update A_{req}, $\mathrm{sp}_{\mathrm{req\,m}}$;
4: **end if**
5: **end while**
6: **if** $q(m)$ retrieving occurred within MaxRound and TTL **then**
Send back the found node;
7: **else** Send message $q(m)$ not found in CDN and punish all qq nodes using their LAs;
8: **end if**
9: **return** found node of $q(m)$.

In order to analyze the performance-versus-complexity trade-off, let Δ be the maximum number of degree of nodes in graph G (e.g., $\Delta \triangleq \max_{k=\{1,\ldots,N\}} D_k$) and let N be the number of nodes to be traversed by each query. If the time needed to check the local query table for the specific file is negligible, the time complexity of the query search in the FLAPS is on the order of $O(1)$. The overall storage requirement of the FLAPS algorithm is the order of MaxRound $\times$ N $\times$ $\log_2(\Delta)$. Table 2 reports the memory requirement and computational complexity of the proposed FLAP and compare them against the corresponding ones of the IAPS [6], APS [29] and k-RW [7] algorithms. An examination of Table 2 points out that the computational complexities of the IAPS, APS and RW algorithms are Δ times larger than that of the FLAPS, while the memory requirement scales as $\log(\Delta)$.

Table 2 Storage requirements and computational complexities $R :=$ Maxround

	FLAPS	IAPS [6]	APS [29]	RW [7]
Storage requirement	$O(R \times N \times \log_2(\Delta))$	$O(R \times N \times \Delta)$	$O(R \times N \times \Delta)$	$O(R^2 \times N \times \Delta)$
Computational complexity	$O(1)$	$O(\Delta)$	$O(\Delta)$	$O(R \times \Delta)$

We conclude this Sect. 4 by pointing out that the implementation of the proposed FLAPS algorithm is distributed, and, being structured in rounds (see Eq. (3)), its convergence rate does not depend on the diameters (in the number of hops) of the underlying peer clusters.

5 Simulated scenarios and considered performance metrics

In this section, we describe the considered test scenarios and the adopted performance metrics.

5.1 Setup description

We resorted to random graph models in order to simulate the overlay P2P transport-layer network and used the PeerSim software [30] for the simulations. All simulations are carried on a desktop equipped with Intel core 2 dual 2.6 GHZ CPU and 4.0 GB RAM. In our simulations, the number of peers is set to 1000 with 20% of failure rate. In addition, the peer-to-peer average path delay is 200 (ms) [31]. The average node inter-request time for query searches is set to 3000 (ms), and, each node has an average degree of 8. Furthermore, we set the maximum number of walkers to 15 and the TTL is set to 6 hops.

Table 3 summarizes the main simulated parameters and their default values. In the simulated scenarios, 150 files are used. The adopted inquiry and repetition strategies follow the protocols described in [32]. Nevertheless, in the simulations carried out in this paper, we adopt a less skewed distribution, where the files in the 90th percentile of the ranking are about 40% of the total number of stored files. Furthermore, three file extensions are employed, e.g. .mp4, .amc, and .txt, with work sizes less than 5 MB. For each file type, 1000 files are evenly scattered over the nodes of the network [31]. The used values for the reward and punishments b coefficients are 0.05 and 0.01, respectively.

Table 3 Main simulated parameters

Parameters	Value
Number of nodes	1000
P2P model	Pure
Graph model	Random
Average node in/out degrees	8
Number of walkers (k)	15
TTL (time-to-leave)	6
Number of files	150
Replication distribution	Zipf ($\alpha = 0.82$)
Query distribution	Zipf ($\alpha = 0.9$)
Number of requester nodes	100
Number of queries per requester node	300

5.2 Definition of the adopted performance metrics

In the carried out simulations, the following five performance metrics have been numerically evaluated:

(i) *Success rate (SR)*: it is the ratio between the number of successful queries and the total number of queries;
(ii) *Hit-per-query (HpQ)*: it is the per-query number of successfully retrieved files;
(iii) *Message-per-query (MpQ)*: it is the per-query number of sent messages;
(iv) *Response delay (RD)*: it is the time elapsed from sending the query request to receiving the corresponding response; and,
(v) *Messages duplication (MD)*: it is the per-query average number of duplicated messages needed for completing the search process.

6 Performance evaluation and comparisons

In this section, we test and compare the performance of the proposed FLAPS algorithm against the corresponding ones of some competitors, namely, the APS [29], IAPS [6] and k-RW based [7] search algorithms. Shortly, we point out that: (i) the APS algorithm uses the feedback from the previous searches to efficiently perform future searches. Also, it uses k walkers out of Δ connections at each node and updates the probability functions which are used to increase the probabilistic forwarding to reach the destination; (ii) the IAPS algorithm relies on an ant-based multi-agent system and utilizes independent walkers that are distributed, over a P2P network. It uses a reward/punishing mechanism that is driven by the ants' movements along the routing paths; and, (iii) the k-RW algorithm randomly chooses one neighbor and continues this process until the query destination is reached. Although the k-RW decreases the overhead of the query messages, its average response delays are typically quiring large [7], and the reported results confirm, indeed, this conclusion.

6.1 Success rate

The first group of carried out tests aims to evaluate and compare the SR of the FLAPS, APS, IAPS and k-RW search algorithms. The obtained numerical results are reported in Fig. 5. An examination of this figure leads to two insights. First, the SR performance of the FLAPS algorithm improves for increasing values of the number k of walkers and approaches 85% at $k = 15$. This value is around 39, 9 and 3% better than the corresponding ones attained by the k-RW, APS, and IAPS, respectively. These results support the effectiveness of the learning mechanism implemented by the FLAPS algorithm. Second, the SR curve of the FLAPS algorithm quickly increases for values k of the walkers up to 10 and, then, it takes a quasi-flat behavior. Hence, values of k as small as 8–9 suffice for attaining the most part of the final SR value.

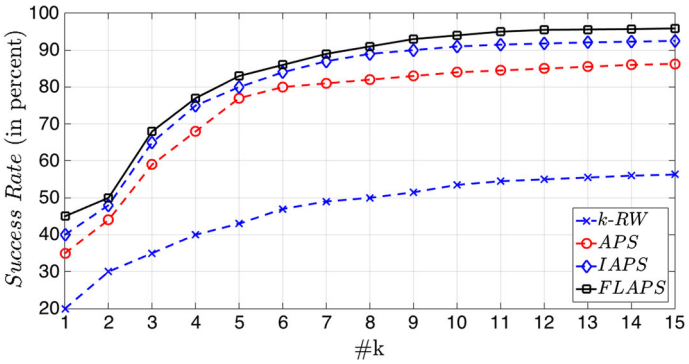

Fig. 5 Success rate-versus-number of walkers k

6.2 HpQ and MpQ performances

The second group of numerical tests focuses on the numerical evaluation of the HpQ and MpQ performance metrics of the FLAPS, APS and IAPS algorithms normalized with respect to the corresponding ones of the k-RW algorithm. An examination of the plots of Fig. 6 leads to the following three main conclusions. First, since the k-RW algorithm tends to generate large volumes of query messages, the HpQ performances of the APS-type algorithms are better than the corresponding one of the k-RW. Second, for values of k larger than 5, the HpQ performance of the FLAPS outperforms the corresponding ones of the APS and IAPS of about 12 and 7%, respectively. Third, although the values of the MpQ metric increase for increasing values of the performed hops under all the tested algorithms, the rate of increment of the FLAPS algorithm is less than the corresponding ones of the APS and IAPS algorithms. Roughly speaking, this is due to the fact that the punishing/rewarding policy implemented by the learning phase of the FLAPS algorithm increases the probability of the selected walkers to participate in the file search, and this reduces, in turn, the per-query average number of generated messages. At this regard, an examination of the plots of Fig. 6b corroborates the conclusion that, for values of k larger than 7–8, the FLAPS MpQ values are less than the corresponding ones of the k-RW, IAPS and APS of about 2, 12 and 23%, respectively.

6.3 Normalized response delay

The third group of simulations aims at evaluating the average per-query normalized response delay and the corresponding per-hop normalized average delay. The obtained numerical results (expressed in terms of multiple of the round time) are reported in Fig. 7a and b, respectively.

They refer to a TTL value of 6 hops. An examination of these figures leads to two main conclusions. First, the quasi-flat behavior of the lowest plot of Fig. 7a supports the conclusion that the response delay of the k-RW algorithm is not sensitive to the considered number of hops. However, the corresponding behaviors of the response

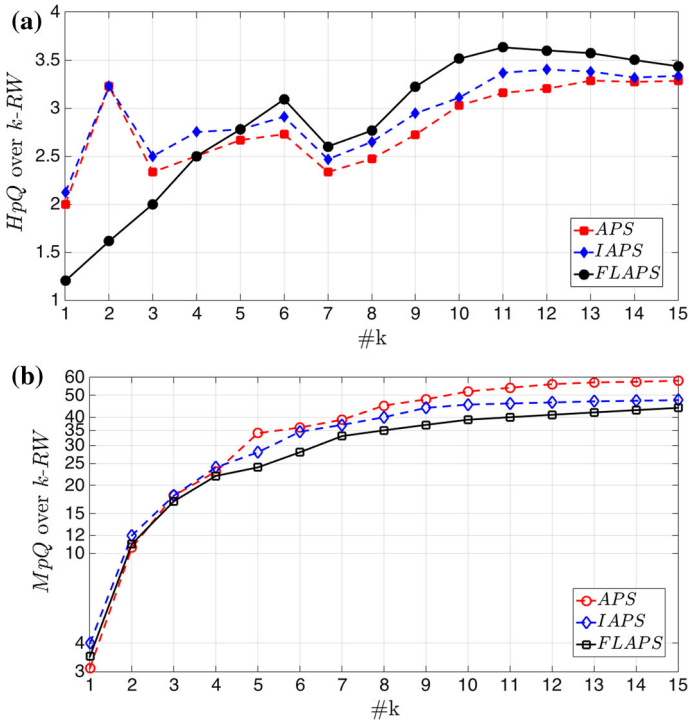

Fig. 6 k Number of walkers; *HpQ* hit-per-query; *MpQ* message-per-query. **a** HpQ-versus-number of walkers normalized to k-RW, **b** MpQ-versus-number of walkers normalized to k-RW

delays of the APS, IAPS, and FLAPS algorithms are bell-shaped and picked around $k = \text{TTL}/2$ (e.g., $k = 3$ in Fig. 7a). Hence, in terms of response delay, we have that (i) the APS-based algorithms are effective for values of k below $k = \text{TTL}/2$ and (ii) due to its self-learning feature, the FLAPS algorithm outperforms the APS and IAPS ones at $k \leq \text{TTL}/2$. Second, the numerical results of Fig. 7b supports the conclusion that the APS-based algorithms outperform the k-RW in terms of per-hop average delay. Roughly speaking, this is due to the fact that the k-RW algorithm tends to generate numbers of per-hop messages larger than the corresponding ones of the APS-based algorithms. Furthermore, due to the exploitation of its learning capability in the neighbor selection phase, the FLAPS algorithm outperforms the IAPS and APS ones of about 5 and 13%, respectively (see the right-most bars of Fig. 7b).

6.4 Message duplication

The fourth set of simulations aims at evaluating and comparing the MD performance, in order to test the actual effectiveness of the FLAPS algorithm in reducing the collision rate of the utilized walkers. The obtained numerical results are reported in Fig. 8, and their examination leads to two main conclusions. First, the APS-based algorithms (largely) outperform the k-RW one in terms of the MD metric. Roughly speaking,

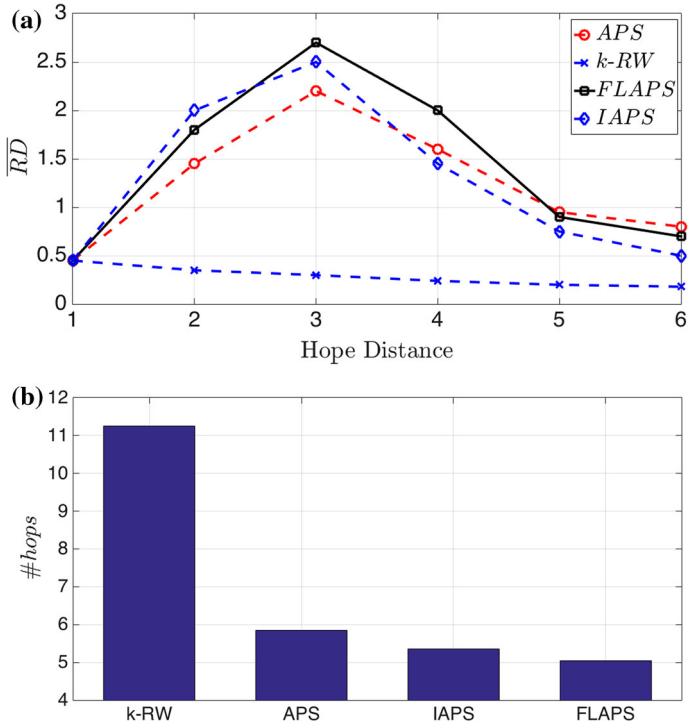

Fig. 7 **a** Walker distance (in number of hops) at each query, **b** average number of crossed hops

this is due to the fact that the APS family of algorithms enhances its search capability by jointly exploiting peer priority, and query tables. Second, the FLAPS algorithm is capable to effectively exploit its self-learning capability, in order to reduce the occurrence of message failures and, then, lower message duplications. As a consequence, a comparison of the lowest plots of Fig. 8 shows that MD performance of the FLAPS algorithm is better than the corresponding ones of the IAPS and APS algorithms of about 18 and 25%, respectively.

6.5 Performance effects of the size of the overlay network

The fifth set of simulations aims at investigating the performance sensitivity of the considered search algorithms on the sizes and topology features (like the average in/out node degree) of the considered overlay networks. For this purpose, as in [6], pure and uniform distributions have been used for the random generations of the overlay networks tested in this section. The obtained average results are reported in Table 4, and they refer to a replication ratio of 1%.

An examination of Table 4 leads to three main conclusions. First, the values of the performance metrics of the FLAPS algorithm do not significantly vary, even when the sizes of the simulated networks increase by a factor 5 (compare the first four columns of Table 4). Second, under all tested configurations, the performance metrics

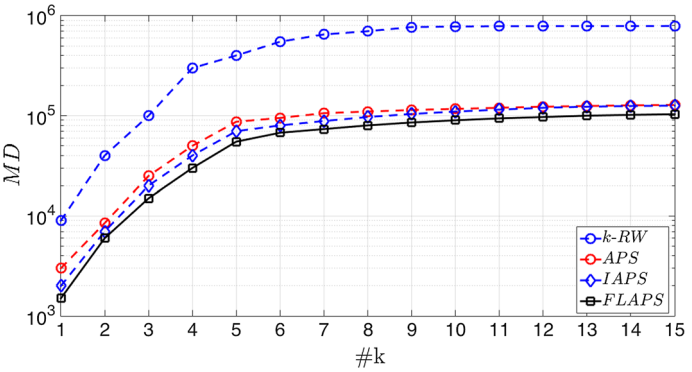

Fig. 8 MD-versus-number of walkers

Table 4 Average performances, *U* uniform distribution; *N* number of peers. The reported values are averaged over 15 walkers

(N, Topology)	SR	HpQ	MpQ	DM	SR	HpQ	MpQ	DM
	FLAPS				IAPS			
(10^3, Pure)	83.5	4.5	30.6	6.7×10^4	80.6	4.3	34.7	8.2×10^4
(5×10^3, Pure)	80.2	4.0	33.1	6.3×10^4	76.2	3.8	36.8	7.9×10^4
(10^3, U)	88.4	4.7	40.6	6.9×10^4	84.2	4.6	44.9	8.4×10^4
(5×10^3, U)	85.9	4.6	46.7	6.8×10^4	79.6	4.4	47.6	8.2×10^4
	APS				k-RW			
(10^3, Pure)	74.6	4.1	39.4	8.9×10^4	46.4	1.5	31.5	5.4×10^5
(5×10^3, Pure)	72.0	3.1	44.1	8.5×10^4	49.2	1.1	28.3	5.0×10^5
(10^3, U)	78.2	4.3	47.7	9.6×10^4	38.3	0.9	45.5	5.6×10^5
(5×10^3, U)	75.5	3.5	50.0	9.3×10^4	40.0	0.8	42.2	5.3×10^5

of the FLAPS algorithm are uniformly better than the corresponding ones of the considered competing algorithms. This confirms the performance scalability of the proposed search algorithm, even under random network settings.

6.6 Performance effects of the number of cluster headers

The last set of carried out simulations aims at giving insight into the performance sensitivity of the considered search algorithms on the number of selected F-CHs. Roughly speaking, we expect that the average cluster size decreases and the average number of inter F-CH messages increases when the number of selected F-CHs grows. In order to evaluate the relative effects of these two phenomena, we have performed simulations in which $k/2$ randomly selected peers act as F-CHs (i.e., same as the setup defined in [22]). Table 5 reports the obtained numerical results. They refer to *three*

Table 5 Effects of the CDN density

	Scenario 1	Scenario 2	Scenario 3
Peers	64	256	1024
F-CHs	8	32	256
Requests	165	640	2750
SR (%)	85.5%	83.3%	82.1%

different scenarios, namely scenarios 1, 2 and 3. In scenario 1 (resp., scenarios 2 and 3), the average number of nodal degree is 3 (resp., 6 and 10). All the simulated parameters are as in Table 3. Roughly speaking, we expect that the average success rate decreases for increasing values of the CDN density, e.g., by passing from scenario 1 to scenario 3. This is due to the fact that the total number of query requests quickly increases by passing from scenario 1 to scenario 3 (see the third row of Table 5). However, the numerical results reported by the last row of Table 5 show that the decrement of the success rate suffered by the FLAP algorithm is limited up to 3.4%. This support the conclusion that the FLAP performance well scales with the density of the underlying CDN.

7 Conclusions and future work

This paper focuses on the problem of file search in (possibly, heterogeneous) Fog-supported P2P CDNs. Its main goal is to leverage the local (possibly, time varying) information about the inter-peer distances and states of the inter-peer TCP/IP connections provided by the serving Fog nodes, in order to reduce the network-wide message duplication (e.g., bandwidth consumption) and search latency. Toward this end, a hybrid (e.g., mixed infrastructure—"ad hoc") architecture for the Fog- supported CDN has been presented (namely, the FCP2P architecture), and the functionalities of its main building blocks have been described. Afterward, the FLAPS algorithm has been proposed, in order to implement latency-efficient file searching over Fog-supported P2P overlay networks. Remarkable features of the FLAPS algorithm are that: (i) its implementation is distributed over the available peer nodes; and, (ii) it is capable of adapting to the (possibly, time varying) topological properties of the available overlay network. Both these features are attained by equipping each peer node by an LA that acquires context information by the environment (e.g., proximate peers and serving Fog nodes) and, then, processes it through a suitable Q-learning reinforcement algorithm. The FLAPS performance is numerically tested and compared to the corresponding ones of some state-of-the-art algorithms, namely, the APS, IAPS and k-RW algorithms. The obtained numerical results corroborate the conclusion that the proposed FLAPS algorithm outperforms the considered competing ones in terms of latency and success rate.

In principle, the presented work could be further expanded along three main research directions. First, the presented version of the FLAPS algorithm operates on single-indexed search tables (see Table 1). Hence, according to the emerging paradigm of the information-centric networking [12], it could be of interest to develop a generalized

version of the FLAPS algorithm for the efficient data retrieving over multi-indexed file tables. Second, current placement algorithms for Fog-Caching do not still account for the client preference, client experience, Fog storage and bandwidth resources. Hence, it could be of interest to develop data placement algorithms that account for both spatial and temporal popularity of the cached data. Finally, emerging Virtual Reality-based applications involves intensive data analytic operations that, in turn, demand for largely duplicated databases. Hence, a third research direction may concern how intelligently to perform Fog-caching, in order to maximize the database reuse and, then, speed up the corresponding file retrieving operations.

Acknowledgements This work has been supported by the project "GAUCHO—A Green Adaptive Fog Computing and Networking Architecture" founded by Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2015, and by the project "V-FOG: Vehicular Fog for energy-efficient QoS mining and dissemination of multimedia Big Data streams" founded by Sapienza University of Rome Bando 2016.

References

1. CISCO (2016) Cisco visual networking index: global mobile data traffic forecast updated, 2015–2020. White paper. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-forecast-qa.pdf
2. Wang X, Chen M, Taleb T, Ksentini A, Leung V (2014) Cache in the air: exploiting content caching and delivery techniques for 5G systems. IEEE Commun Mag 52(2):131–139
3. Bastug E, Bennis M, Debbah M (2014) Living on the edge: the role of proactive caching in 5G wireless networks. IEEE Commun Mag 52(8):82–89
4. Li Y, Sun L, Wang W (2014) Exploring device-to-device communication for mobile cloud computing. In: Communications (ICC), 2014 IEEE International Conference on. IEEE, pp 2239–2244
5. Tigelaar AS, Hiemstra D, Trieschnigg D (2012) Peer-to-peer information retrieval: an overview. ACM Trans Inf Syst (TOIS) 30(2):9
6. Shojafar M, Abawajy JH, Delkhah Z, Ahmadi A, Pooranian Z, Abraham A (2015) An efficient and distributed file search in unstructured peer-to-peer networks. Peer-to-Peer Netw Appl 8(1):120–136
7. Gkantsidis C, Mihail M, Saberi A (2004) Random walks in peer-to-peer networks. In: INFOCOM 2004. Twenty-third annual joint conference of the IEEE computer and communications societies, vol 1. IEEE
8. Li B, Li J, Huai J, Wo T, Li Q, Zhong L (2009) Enacloud: An energy-saving application live placement approach for cloud computing environments. In: Cloud computing, CLOUD'09. IEEE International Conference on. IEEE, pp 17–24
9. Gao Y, Guan H, Qi Z, Hou Y, Liu L (2013) A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 79(8):1230–1242
10. Lucas-Simarro JL, Moreno-Vozmediano R, Montero RS, Llorente IM (2013) Scheduling strategies for optimal service deployment across multiple clouds. Future Gener Comput Syst 29(6):1431–1441
11. Yang L, Cao J, Liang G, Han X (2016) Cost aware service placement and load dispatching in mobile cloud systems. IEEE Trans Comput 65(5):1440–1452
12. Cui Y, Wu Y, Jiang D (2015) Analysis and optimization of caching and multicasting in large-scale cache-enabled information-centric networks. In: Global Communications Conference (GLOBECOM), 2015 IEEE. IEEE, pp 1–7
13. Gnutella forum (2017). http://www.gnutellaforums.com/
14. Merugu S, Srinivasan S, Zegura E (2003) Adding structure to unstructured peer-to-peer networks: the role of overlay topology. In: Group communications and charges. Technology and business models. Springer, pp 83–94
15. Chawathe Y, Ratnasamy S, Breslau L, Lanham N, Shenker S (2003) Making gnutella-like p2p systems scalable. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. ACM, pp 407–418

16. Chowdhury F, Kolberg M (2013) Performance evaluation of structured peer-to-peer overlays for use on mobile networks. In: Developments in eSystems Engineering (DeSE), 2013 Sixth International Conference on. IEEE, pp 57–62
17. Maymounkov P, Mazieres D (2002) Kademlia: A peer-to-peer information system based on the xor metric. In: International Workshop on Peer-to-Peer Systems. Springer, pp 53–65
18. Thampi SM, Sekaran KC (2007) Autonomous data replication using q-learning for unstructured p2p networks. In: Sixth IEEE International Symposium on Network Computing and Applications (NCA). IEEE, pp 311–317
19. Yang B, Garcia-Molina H (2002) Improving search in peer-to-peer networks. In: Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on. IEEE, pp 5–14
20. Di Caro G, Dorigo M (1998) AntNet: distributed stigmergetic control for communications networks. J Artif Intell Res 9:317–365
21. Michlmayr E (2006) Ant algorithms for search in unstructured peer-to-peer networks. In: 22nd International Conference on Data Engineering Workshops (ICDEW'06). IEEE
22. Shojafar M, Cordeschi N, Baccarelli E (2016) Energy-efficient adaptive resource management for real-time vehicular cloud services. IEEE Trans Cloud Comput PP(99):1–14
23. Dabbagh M, Hamdaoui B, Guizani M, Rayes A (2016) An energy-efficient VM prediction and migration framework for overcommitted clouds. IEEE Trans Cloud Comput PP(99):1
24. Rhea SC, Kubiatowicz J (2002) Probabilistic location and routing. In: INFOCOM, Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3. IEEE, pp 1248–1257
25. Hayajna T, Kadoch M (2017) Analysis and enhancements of hello based link failure detection in wireless mesh networks. Telecommun Syst. doi:10.1007/s11235-017-0293-4
26. Cordeschi N, Amendola D, Shojafar M, Baccarelli E (2015) Distributed and adaptive resource management in cloud-assisted cognitive radio vehicular networks with hard reliability guarantees. Veh Commun 2(1):1–12
27. Naranjo PGV, Shojafar M, Mostafaei H, Pooranian Z, Baccarelli E (2017) P-SEP: a prolong stable election routing algorithm for energy-limited heterogeneous fog-supported wireless sensor networks. J Supercomput 73(2):733–755
28. Mustafa M, Papatriantafilou M, Schiller EM, Tohidi A, Tsigas P (2012) Autonomous tdma alignment for vanets. In: Vehicular Technology Conference (VTC Fall), 2012 IEEE. IEEE, pp 1–5
29. Tsoumakos D, Roussopoulos N (2003) Adaptive probabilistic search for peer-to-peer networks. In: Peer-to-Peer Computing, P2P 2003. Proceedings. Third International Conference on. IEEE, pp 102–109
30. Peersim: A peer-to-peer simulator (2016) http://peersim.sourceforge.net/
31. Marti S, Ganesan P, Garcia-Molina H (2004) DHT routing using social links. In: International Workshop on Peer-to-Peer Systems. Springer, pp 100–111
32. Ripeanu M, Foster I (2002) Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems. In: International Workshop on Peer-to-Peer Systems. Springer, pp 85–93