Semantic Web 0 (2022) 1–0
IOS Press

Survey of Models and Architectures to Ensure Linked Data Access

Mahamadou Toure [a,b,*], Kaladzavi Guidedi [c], Fabien Gandon [a], Moussa Lo [b] and Christophe Gueret [d]
[a] *WIMMICS, Inria Sophia Antipolis, Nice, France*
E-mails: mahamadou.toure@inria.fr, fabian.gandon@inria.fr
[b] *Gaston Berger University, Saint Louis, Senegal*
E-mail: moussa.lo@ugb.edu.sn
[c] *University of Maroua, Maroua, Cameroon*
E-mail: kaladzavi@univ-maroua.cm
[d] *Accenture Labs Dublin, Dublin, Irlande*
E-mail: christophe.gueret@accenture.com

Abstract. Mobile Access to the Web of Data is currently a real challenge in developing countries, mainly characterized by limited Internet connectivity and high penetration of mobile devices with the limited resources (such as cache and memory). In this paper, we survey and compare proposed solutions (such as models and architectures) that could contribute to solving this problem of mobile access to the Web of Data with intermittent Internet access. These solutions are discussed in relation to the underlying network architectures and data models considered. We present a conceptual study of peer-to-peer solutions based on gossip protocols dedicated to design the connected overlay networks. In addition, we provide a detailed analysis of data replication systems generally designed to ensure the local availability of data on the system. We conclude with some recommendations to achieve a connected architecture that provides mobile contributors with local access to the Web of data.

Keywords: Web of Data, Gossip protocol, graph replication, limited connectivity, semantic overlay, mobile contributor

1. Introduction

The Web (or Web of Documents) is originally seen as a software architecture for making documents available, and for linking and sharing them over a network of connected machines [1]. This vision evolved very quickly. Indeed, in [2], the author shows how the vision of hypertext, i.e. the linking of documents by hypertext links, must be overcome to allow machines to automatically link Web data to real world-elements, which would at the same time allow intelligent agents to add and manipulate Web content. In 2006, Tim Berners Lee [3] clarified his semantic web vision by emphasizing that its sole aim was not to publish information on the Web, but to link them in order to allow a machine or a human to browse the Web of data. He then presented a collection of guidelines for delivering and linking structured data on the Web: the Linked Data Principles. These principles offer directions on the application of semantic web standards technologies (such as RDF, URI, SPARQL) to link data from different sources. By analogy to traditional Web (of documents), that can be explored via hyperlinks, applications that adopt the principles of linked data can browse the Web of Data made up of different data sources by following the RDF (Resource Description Framework) links to provide more relevant responses to users' needs [4].

In a number of the distributed systems such as Web applications, search engines and e-learning platforms, some network nodes provide services, and others consume these services. Internet is the common communication channel that allows the nodes that constitute data sources and services (server nodes) and consumer nodes of these services (client nodes) to be connected and interacting. However, the availability and partic-

*Corresponding author. E-mail: mahamadou.toure@inria.fr.

ularly the quality of the Internet is not always ensured in certain areas such as the African continent. Indeed, despite the significant progress made in recent years, Internet access remains a major constraint in developing countries in general and Africa in particular. According to Internet Words Stats data [5] of December 2020/June 2021, the penetration rate was estimated at 46.2% in Africa compared to 93.9% in North America, 87.1% in Europe and 63.8% in Asia. Furthermore, even when Internet access is provided, the distributed systems must cope with the dynamics of their environments. Distributed systems have to mainly consider the eventual node failure and the popular client/server model is then revealing its limitations. In such a model, the server is the only node that provides the service and in case of unavailability, it becomes the single failure point for the entire distributed system. Moreover, this centralized architecture, apart from the significant costs that may be incurred to keep it operational, does not efficiently exploit the available resources (such as storage memory and processor) of the client nodes, which participate passively to the architecture and benefit from the provided service. Nevertheless, these client nodes have the particularity of being increasingly mobile, numerous and constitute a privileged means of access and consumption of Web's services. For example, GSM Association (GSMA) 2020 report on mobile economy in Sub-Saharan Africa [6] shows that smartphone adoption continues to rise rapidly in the region, reaching 50% of total connections in 2020, as cheaper devices have become available. Over the next five years, the number of smartphone connections in Sub-Saharan Africa will almost double to reach 678 million by the end of 2025, representing an adoption rate of 65% of the population.

In recent years, the peer-to-peer (P2P) model positioned itself to be an efficient solution to meet the dynamism and scalability requirements of the distributed systems [7]. Nodes act both as clients and servers, contributing to the services in which they participate. Regardless of the specific type of application, p2p systems can be defined according to the topology connecting the nodes to each other to form the underlying network usually called overlay [7]. There are two main classes of superposed networks [7]: structured overlay networks well represented by distributed hash tables (DHT) and unstructured overlay networks. The former link their peers according to their identifiers in order to allow efficient routing between them. The identifier of each node determines its position on the network. But, this makes the architecture very vulnerable to the departure/arrival or node crash scenarios [8]. For the unstructured overlay networks, links between nodes are created randomly or based on proximity measurements. Networks based on gossip protocols are good illustrations [9–12]. These protocols aim to build and maintain an unstructured topology with random graph properties [13]. They also provide load balancing between nodes and are used as building blocks in network management applications, in particular for very dynamic environments [11].

In this paper, we aim to survey the existing solutions (such as models and architectures) that can contribute to solving the problem of intermittent access to the Web of Data by mobile contributors. These solutions are discussed here in relation to the network architectures and data models. We firstly present a conceptual study of P2P solutions based on gossip protocol dedicated to the design and management of the connected overlay networks. And secondly, we give a detailed analysis of the technical approaches to ensure local data availability on such a peer-to-peer architecture.

Our contributions in this paper are: 1) a comparison of the functional approaches of gossip protocols based on the underlying adhesion mechanism. 2) A classification of approaches dedicated to designing data sharing systems adopting an RDF data model. For each of these approaches, we also identify a set of the existing applications and the future research trends that we consider relevant.

The document is organized as follows: Section 2 presents our research methods. In Section 3, we present a motivating application scenario. Section 4 addresses unstructured architectures based on gossip protocols and the the system evolution model generated by contributors. Section 5 describes the approaches taken from these architectures to access the Web of Data despite connectivity and hardware resource constraints. Section 6 presents the conclusion with an overview of the evaluation parameters and future research directions.

2. Research Methods

2.1. Keywords and terms

We indicate here the keywords and the expressions and synonyms used to search for scientific papers to be considered in preparing this document. These queries and variations were the seeds to create the corpus of related work we studied.

Keywords: Semantic web, gossip protocol, peer-to-peer system, mobile access, Data web, graph replication, limited connectivity, semantic overlay, mobile contributor.

Search terms: "Semantic profile" or "semantic clustering" or "Semantic data exchange" and "gossip protocol" or "random peer sampling" or "membership management" and "peer-to-peer system" or "peer-to-peer membership" and "mobile access" or "local access" or "local data" or "local data replication".

2.2. Sources

We indicate here the main sources used to write this document:

- `DBLP`: dblp.uni-tier.de
- `HAL`: hal.archives-ouvertes.fr
- `ACM Digital Library`: dl.acm.org
- `IEEExplore`:ieeexplore.ieee.org
- `Mendeley`:scholar.google.com
- `CiteSeer`:citeseerx.ist.psu.edu

2.3. Document selection criteria

We now document the different criteria considered when searching for documents. Our choices are justified by the orientation of the papers and led to the following inclusion and exclusion criteria.

Inclusion criteria:

- Papers dealing with the design of gossip protocol
- Papers dealing with the problem of random sampling in gossiping
- Papers proposing data exchange systems based on gossip protocols
- Papers dealing with semantic data exchange on peer-to-peer architectures
- Papers dealing with local access to data sources
- Papers written in French or English
- Papers published in conferences or journals, short papers, workshop papers and research reports that address the points mentioned above
- Papers less than 5 years old (priority)

Exclusion criteria:

- Papers that do not deal with peer-to-peer communication or local access to data
- Slides of presentations, unpublished studies
- Papers dealing with semantic web techniques but not applied to peer-to-peer architectures
- Papers that present gossip mechanisms but are not dedicated to data exchange in physical environments
- Papers that offer local access solutions on non-mobile devices
- Papers dealing with gossip protocols on centralized peer-to-peer architectures

3. Motivation: Example of scenario

We target a scenario where users form a P2P network such that anyone can generate information and make it available to everyone else on the network. To illustrate this scenario, let us consider the real case of the International Jazz Festival of Saint-Louis in Senegal. It is an annual event during which thousands of people are meeting in different locations around the city to celebrate Jazz music and enjoy the various concerts and cultural activities held for this purpose. During this event, the constant need to access and share information related to the schedule and their contents is an important point for improving the quality of the festival.

Mobile access to the Web of Data: Khadim, Thierno and Guirane are friends who came to attend the festival. Khadim is interested in concerts at Abdoulaye Wade Square and festivities held mainly on the edge of the Senegal River that borders the city. Thierno and Guirane came specially for different quintets' show on Faidherbe Square. Their smartphones are assumed to be equipped with the application dedicated to this event. When they arrive at the Saint-louis bus station, they detect the wifi network in the area dedicated to the festival and got connected to it. Thierno and Guirane are then automatically integrated into clusters located in the area in relation to their points of interest. As for Khadim, being the only one interested in African Jazz, he is integrated in an empty cluster for this point of interest and in another cluster related to the river. Ev-

eryone wants to have information about their interests, so they query their applications. Thierno and Guirane receive identical information since they belong to the same clusters. Khadim receives information about festivities held near the river. He also receives information on concert planning. This information originates from a cluster located at Abdoulaye Wade Square. Their friend Samuel who lives in Saint Louis has also been integrated into this cluster.

Local access and collaborative modification: Samuel has set up his profile by specifying his interest for concerts at Abdoulaye Wade Square. By visualizing the schedule of these concerts, he notices that the name of one of the artists was incorrectly written on the system, he decides to modify this information. A local band decides to informally join the festival by playing at the entrance of the bridge (of the city) and they add their event to the shared data. Khadim decides a few minutes later to refresh the data presented to him and finds that the artist's name has indeed changed and that a new event was added.

4. Peer-to-peer and semantic data exchange

Research on RDF data exchange on peer-to-peer networks has made great progress. Many of these solutions, particularly those dedicated to dynamic architectures, are based on gossip-based (or epidemic) protocols. These protocols have an interesting approach in the sense that they are very resistant (having an intrinsic redundancy degree allowing to mask network and node failures) and scalable (load distribution across all system nodes) [14]. The operating principle is conceptually as follows: when a node (contributor) needs to send information via the network, it randomly selects t nodes among its neighbors and transmits them the message (t is a parameter named *fanout*). Each node repeats this process once it first receives the message.

Challenges and operating principles of these protocols are very diverse. They are very suitable for designing peer-to-peer communication systems because of their scalability, ease of deployment, but above all their resistance to network and process failures. They have been successfully applied in several areas: aggregate calculation (such mean, variance, minimum and maximum) [9–11], load balancing [12], network management [13]. The shared characteristic of such protocols is that each node periodically or reactively exchanges information with certain of its peers. A gossip protocol assumes the existence of an underlying membership mechanism, a fundamental component, which provides each node a complete or partial system knowledge. This consists of a list of node identifiers or profiles commonly called a *view*.

The costs in terms of memory and network traffic to ensure overall system knowledge are generally elevated for dynamic networks where nodes continuously leave and join. These constraints lead us to mechanisms that provide partial knowledge of the system. These mechanisms are more adapted to the dynamic characteristics of the system and are less constraining in terms of resources. Several gossip protocols were then implemented on top of these mechanisms. The latter can be classified into two families [14]: basic membership mechanisms and two-layer membership mechanisms.

4.1. Gossip protocols with basic membership mechanisms

With this type of mechanism, the view owned by each node is composed of peers dispersed across the network. No characteristics are considered on a node to integrate it into another node's view. Two system models can be identified among these protocols [14]: centralized systems that use a set of central servers whose main task is to provide each node with a random view, and decentralized systems whose main characteristic is self-organization. We will focus here on protocols that adopt decentralized architectures that avoid in particular problems related to storage memory size and central server failures (Figure 1: On each node, the outgoing arrows indicate the neighboring nodes that make up its local view. The incoming arrows represent the views of the neighboring nodes in which the node is integrated). These protocols can be divided into two groups: (1) protocols using a random selection of peers at runtime and (2) those using deterministic selection.

4.1.1. Random peer selection

These gossip protocols have the particularity of being totally based on random choices. More precisely, at runtime, the peers' selection for information dissemination is done randomly among the peers composing the node view. All the nodes composing the view have in principle the same probability of being requested for the exchange. This keeps the architecture connected and close to the random graph properties. The chal-

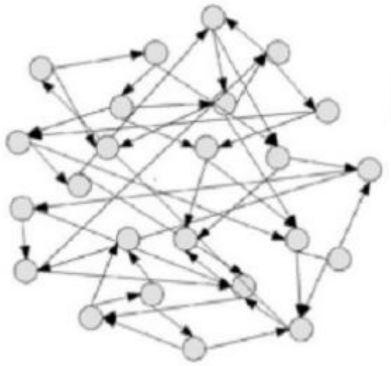

Fig. 1. Example of a basic membership mechanism [14].

lenge is therefore to build and maintain the connected graph even with node failures. Choosing the *fanout* parameter to be applied to the protocol is decisive in the sense that it has a particular impact on the speed of message propagation. Hence the importance of finding a good value for the *fanout*. The following work is based on this principle.

Ganesh et al. [15] present SCAMP, a peer-to-peer membership service that operates in a completely decentralized manner, where no node has any overall knowledge of members. The system is totally self-organized, the size of local views naturally converges to the "right" value for gossips to succeed. This value depends on the size of the system.

Jelasity et al. [12] introduce a generic scheme, generalizing gossip protocol based on sampling services. Sampling services aim to give each node a set of peers with which to exchange information. Authors show that unstructured dynamic overlay networks based on gossip protocols are natural candidates for the implementation of sampling services for their reliability and scalability. Two methods are presented in the Peer Sampling Service API: Init() and GetPeer(). Init() starts the service for a given node if it was not previously done. GetPeer() retrieves a peer address when the group includes several nodes.

Jelasity et al. [16] introduce a generic framework implementing decentralized peer-sampling service by building and maintaining unstructured dynamic architectures based on information about peer contributors. The basic principle underlying the proposed framework for designing the peer sampling service itself relies on gossip paradigm. In fact, each node manages a local small table providing a partial view on the entire set of nodes and periodically updating it by a gossip process.

Leitao et al. [17] present HyParView, a membership protocol to support gossip broadcasting that ensures elevated reliability levels, despite high node failure rate. HyParView is based on an approach based on the use of two distinct partial views: small active view, and a bigger passive one. All nodes' active views build together an architecture used for message transmission. The purpose of the passive view is to maintain a list of nodes that can be used to substitute active view's failed nodes.

Bortnikov et al. [18] present Brahms, a random peer sampling algorithm for large dynamic systems that are prone to malicious behavior. Brahms provides a view to each and also overcomes Byzantine attacks (e.g. attacks where adversaries have full control to certain authentic nodes from which they disrupt the network).

4.1.2. Deterministic peer selection

These protocols differ from random selection protocols in the deterministic nature of their peer selection procedures during data exchange. In previous protocols, although the architecture remains effectively connected despite its dynamic nature, peer-to-peer links are by default meaningless. In the case of a deterministic choice, the selection is guided by the application of mechanisms based on characteristics such as peer "age" (time spent in the view), metric distance, similarity (such shared interests, semantic proximity and profile) and scheduling (Round Robin). This allows to have virtual but relevant links between peers in the architecture, thus improving the quality of the views. In this category, the following related works have attracted our attention.

Voulgaris et al. [13] describes CYCLON, a low-cost, full membership management framework. CYCLON improves on the basic shuffling protocol [19]. Shuffling operation is a swapping process of a neighbour subset among a couple of nodes. It originates from any of these two nodes. The basic shuffle protocol guarantees that overlay connectivity remain intact until membership changes. CYCLON uses a similar design like the basic shuffle. However, nodes do not randomly select the neighbour with whom to exchange informations, they choose the oldest neighbor.

Nedelec et al. [20] proposes a random peer sampling protocol called Spray, based on Scamp and Cyclon. The protocol is designed to avoid the constraints introduced by WebRTC framework. WebRTC allows communication channels between browsers to be estab-

lished. However, it does not manage addressing and routing. Browsers connect by exchanging offers and receipts using a shared mediator like couriers, specialised signaling services or available WebRTC links. Spray avoid these constraints through its three-part connection establishment procedure, by using only neighbor-to-neighbor interactions. Spray: 1- adapts dynamically each peer's neighborhood. So, connection volume logarithmically increases with network size; 2- only uses interactions between neighbours to establish connections. Thus, connections are established in constant time; 3-quickly converges towards a topology with similar properties as a random graph. As a result, the network gains robustness against large-scale failures and efficiently disseminates information.

In **[21], Alromih et al.** proposes EEGossip, an Energy-Efficient Gossiping protocol to route data towards sink. Using a selection procedure, the protocol determines the best path for each neighbor. The selection function uses the next node residual energy, next node distance (the distance between the current node and its neighbor) and the sink distance (the distance between the sink and the next node). For the calculation of the distance, EEGossip uses the Chebyshev distance [22] which overcomes the Euclidean distance regarding both processing complexity and execution time [23].

4.1.3. Synthesis on basic membership mechanism

In summary, gossip protocols adopting a basic membership mechanism have several advantages for the decentralized peer-to-peer systems. They make it possible to maintain the architecture connected despite the mobility (arrival/departure) of the nodes. The failure resistance of nodes and network is ensured in particular by the level of data redundancy. The amount of information passing through the architecture can be controlled by choosing the size of the views, the *fanout* parameter, the type of protocol execution (cyclic or reactive) and also the type of message propagation (push/pull). WebRTC also has an important advantage for these protocols through its signaling and connection services. It enables gossip protocols to be deployed on mobile phone or tablet web browsers, enabling direct/indirect connections between mobile users. Table 1 provides a summary of the different gossip protocols surveyed in this section. The following criteria were used to compact their comparison in one table:

- `Push-propagation`: the protocol is based on a push stream. In this type of propagation, nodes transmit data to randomly selected neighbors without expecting responses from them. The mechanism is very suitable for disseminating information as it is unnecessary to have receivers responding to originators.
- `Pull-propagation`: the protocol is based on a pull stream. Pull propagation guarantees data is sent when required. This allows to reduce network load whenever data size is important.
- `Cyclic Execution` : the protocol is executed in a cyclic manner at the level of each peer.
- `Biased peer selection` : the choice of a peer when executing the protocol is made in a deterministic way. A selection mechanism is applied to select the most suitable node (oldest (age), closest (metric distance), or similarity metric)
- `Defense mechanism` : the protocol integrates one or more security mechanisms to maintain the architecture connected and/or preserve the anonymity of each peer.
- `WebRTC` : the protocol is based on the WebRTC framework which allows communication channels between browsers to be established.

4.2. Gossip protocols with two overlay membership mechanisms

A second family of protocols is designed on top of the membership protocols described in Section 3.1 by adding clustering mechanisms to form two-level architectures (Figure 2). Here, the role of basic gossip protocols is to build and maintain a connected architecture on top of which an appropriate clustering mechanism is then applied to build even more efficient overlay networks.

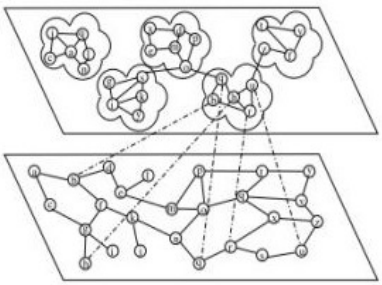

Fig. 2. Example of two overlay membership mechanisms [24].

The common objective of these architectures is to cluster nodes according to a geographical-semantic,

Table 1
Summary of basic gossip protocols. The total support of a characteristic is indicated by the sign √, the partial support by the sign ∘

	Scamp [15]	Cyclon [13]	HyParView [17]	Brahms [18]	Spray [20]	EEGossip [21]
Push-propagation	√	√	∘	√	√	
Pull-propagation			∘	√	√	√
Cyclic execution		√	∘	√	√	
Biased peer selection		√			√	√
Defensive mechanism				√		
WebRTC		√			√	

profile or network proximity criteria and take this proximity into account in order to provide participants with local neighbor lists exclusively made of members of the same cluster. Some models offer mechanisms that allow a few selected nodes in the cluster to be equipped with a remote view composed of nodes from other clusters in order to keep the entire system connected (Figure 2). Kermarrec et al. [14] show this connectivity is achieved through a limited set of links among clusters.

Several models of two overlay protocols have been proposed. These protocols have the particularity of being completely autonomous, in a way that all the nodes have the same roles. The architecture adhesion procedure for a newly arrived node does not require contact with a particular node (contact server). Each node can be used as a contact for a new one. The membership mechanisms differ from one protocol to another. The aim is to find a first peer who responds favourably to the membership request. The latter peer will transfer the information relating to his view according to the type of flow (push and/or pull) adopted by the protocol. This information will allow the node that initiated the request to initialize its view. On a number of these protocols, clusters are also formed by adopting the operating principle of the basic gossip protocols. This is the case of Gossple, Vicinity and Behave [24–26]. To form the second overlay, exchanges between peers are performed according to the gossip model guided by the clustering metric considered. In Gossple, Behave, and Cyclades for example, the cluster is represented by a second view held by each node. This second view is made up of the best neighbours selected on the metric basis. Clusters are improved as they are updated by reactive or cyclical update operations.

A number of these solutions also include caching mechanisms to speed up data access [25–27]. This type of mechanism allows relevant content to be temporarily stored on the architecture. Thus, all local caches form a common cache accessible to system contributors. A partial/total caching of the data passing through the system is performed by these contributors in order to process a large part of the requests from the local cache or from the caches of neighboring peers.

4.2.1. Overlay approaches

Voulgaris et al. [28] proposes a proactive approach to build epidemic protocol-based semantic overlay gathering same content peers. Each peer manages a semantic neighbor list and first queries its semantic neighbors to find a file. The model assumes a semantic proximity function that provides numeric semantic proximity metrics among peers regarding their lists of files.

Jelasity et al. [29] proposes T-MAN gossip protocol which allows to build a variety of network architectures. T-MAN is based on a peer sampling service [16] generating a starting network architecture of random links. T-Man primarily depends on three parameters: message size m, sampling parameter w and ranking method *RANK*. Nodes rank their descriptor set (composed of descriptors from random links) with the ranking method and pick first k as neighbours. These results in a structure named *the target graph*.

Mordacchini et al. [30] proposes a general system architecture dedicated to take advantage of collaborative peer-to-peer information exchange. The idea is to gather similar users and disseminate relevant recommendations across them. The protocol uses a clustering mechanism to group similar users. Each peer firstly independently determines which peers they are linked to. These individual connections are selected based on interest-based metric distance, among the encountered peers. Each time it meets a new peer, it learns from and communicates with potential new neighbours (i.e. similar users). When this process is stabilized, a peer may consider his neighbourhood as a representation of a community of common interest.

Bertier et al. [24] describes the construction of an anonymous social knowledge network using a gossip protocol called Gossple. Periodically, Gossple nodes exchange on their interest and calculate their interest-related distances. They propose to improve navigation in Web 2.0 systems through implicit personalization: an anonymous network of knowledge interested by similar topics is associated with each user, independently from the way they expressed their interests. There is an implied use of this knowledge by users in order to drive and refine searching actions.

Voulgaris et al. [25] present a self-organizing, generic overlay management framework, VICINITY. Given the node descriptor p and a set of node descriptors D, a select function *SELECT(p, D, k)* is applied, to return the set of k descriptors that are most closely related to the outgoing links from p in the target structure. Such function is often built on a proximity-specific measurement globally defined. The protocol relies on two layers. The base layer is the peer sampling service. It is in charge of keeping the architecture connected and periodically providing the upper layer with candidate nodes. Candidates are randomly and uniformly sampled throughout the system. The upper layer protocol, named VICINITY, determines which nodes to foster by use of the selection function.

Frey et al. [26] present Behave, a decentralized caching architecture based on behavioral positions and using gossip protocols to construct superposed clusters with similar interest peers. Behave nodes adopt a gossip protocol based on the similarities between their navigation histories to form an interest-based topology. Each node thus has a set of neighbours whose browsing history is closest to their own. From this topology, Behave's behavioral cache appears as the merging of a node's neighbors' local caches.

Boutet et al. [31] present HyRec, a scalable and cost-effective online system dedicated to customizing user-centric collaborative filtering. HyRec loads recommendation tasks on users' web browsers, while the process is driven by a server which also controls user profile relationships. Each user receives from the HyRec server a set of candidate profiles. Each browser then calculates the KNN (k-nearest-neighbor (KNN) or k-best neighbors,) of its user and the most relevant elements based on this sample. Using a sampling approach, both KNN selection and article recommendation are delegated to users' web browsers. The sampling approach follows the same principle as gossip protocols.

Carvajal et al. [32] introduce a WebRTC-based library called WEbGC. WebGC enables web browsers to communicate using gossip protocol and to interact with node-JS applications.. WebGC is also based on the SimplePeer framework, which operates as a JavaScript library and serves as a layer for WebRTC facilitating peer-to-peer data connections.

Folz et al. [27] present CyCLaDEs, a network architecture based on LDF (Linked Data Fragment) similarities. Using LDF client similarities, CyCLaDEs intends to provide a decentralized behavioral cache for LDF query processing. A predefined number of best customers is identified for each customer and a one-to-one link is established with each one. In case of a given user's process, first the local cache is checked for each sub-request triplet, followed by its neighbors' cache, and if necessary, the LDF server. The network architecture relies on random peer sampling model for member composition management and a clustered architecture to handle the k-best neighbors.

Nedelec et al. [33] present CRATE, a real-time decentralized collaborative editor. CRATE directly operates on web navigators using WebRTC. By using an optimistic replication process, CRATE ensures documents accessibility and responsiveness. For the browser network construction, CRATE uses SPRAY random peer sampling protocol with WebRTC technology. With SPRAY, a locally based neighborhood table is provided to editors allowing communication within a subset of editors. CRATE uses SPRAY protocol to evolutively broadcast any replicated sequence operations to all co-workers.

Pilet et al. [34] introduce a peer-to-peer protocol enabling user privacy protection during decentralized averaging. Many limitations of existing solutions, such as eavesdropping attacks, restrict peer exchange coordination, or use of expensive cryptographic primitives such as homomorphic encryption, are overcome by the protocol. The protocol relies on an attack-resilient Random Peer Sampling (RPS) service: Brahms. Each pair is provided with a sample of remaining network peers by the RPS. During several rounds, the protocol exchanges noise before starting to transmit actual data. This makes it hard for an honest but curious attacker to know whether a user is transmitting noise or actual data.

Meiklejohn et al. [35] present PARTISAN, an actor's application operating platform that enhances evolutivity and decreases latency. In [36], an actor is defined as a computational entity that, in response to a message it receives, can concurrently: send a finite number of messages to other actors, create a finite number of new

actors, designate the behavior to be used for the next message it receives. Actor application is one in which actors have the option to stay on a variety of nodes and be able to seamlessly interact with other nodes' actors. PARTISAN offers greater evolutivity by giving developers the ability to define the used network tier at execution point while not modifying application semantics. PARTISAN offers four overlays to developers, including a peer-to-peer overlay based on HyParView [17] and Plumtree broadcast protocol [37]. The peer-to-peer protocol allows PARTISAN to provide both membership processes used to add/delete members from cluster and sending processes for asynchronous messaging.

Leonardi et al. [38] present NAPA-WINE a P2P television system (P2P-TV) architecture. P2P-TV is defined as a system in with a source dividing video stream in pieces of data, exchanged with nodes and then distributed to all network members. The goal of NAPA-WINE is to push chunks into layer where peers collaborate to broadcast them, with no requirement for huge resources and bandwidth to sustain the service. Gossip protocols such as Newscast [12] or Cyclon [13] are used for the underlying topology management. This allows peers discovery in overlay topologies mapped on over the network.

4.2.2. Synthesis on overlay approaches

These two-overlay protocols benefit from the advantages offered by the basic protocols on top of which they are built. Therefore the architecture is connected and resistant to failures. The second overlay is particularly useful for creating links between nodes that make sense in relation to the context. It therefore reduces latency time during search scenarios. On certain solutions, the clustering algorithm is executed at the execution of the underlying base protocol. This limits the amount of information transiting on the architecture. Table 2 provides a summary of the different solutions presented in this section. The following characteristics allowed us to compare these different models:

- `Similarity metric` : the model uses a similarity metric to estimate the distance (in terms of interest, metric, semantics or content) between peers;
- `Clustering by gossip` : the clustering algorithm used is based on the same mechanism as a gossip protocol at runtime;
- `Compression of exchanged data` : a compression strategy is applied on data to maintain fluid exchanges in the system;
- `Caching` : the model integrates one or more caching techniques for data storage and thus ensures availability and local access;
- `Semantic clustering` : the clustering algorithm is based on semantic criteria; more precisely, the similarity metric used estimates the semantic distance between peers;
- `Membership Management Protocol (MMP)` : this is the base protocol (gossip protocol) used in the model to build and maintain the architecture's connectivity;
- `Contact server` : the arrival of a new peer in the architecture is done through a contact server whose main role is to store all or a part of each peer's information and to provide each newcomer with the necessary data for its integration (such as view, IP address and identifier).

4.3. Other relevant work on peer-to-peer architectures

Here we first give a brief overview of the advantages of p2p network architecture compared to client-server architecture.

In client-server architecture, tasks or workloads are partitioned between servers and services are requested by clients. Typically, clients and servers communicate via a network, but they may also reside on the same system. In peer-to-peer architecture, tasks or workloads are partitioned between peers and are deemed to form a peer-to-peer network. Peers have the same potential and the same privileges. Peers make some of their resources, such as processing power, disk storage or network bandwidth, available to other network participants.

The main difference between these two types of architectures is that in the client-server architecture there are designated clients that request services and servers that provide these services, whereas in peer-to-peer systems, peers act as both consumers and providers. In addition, client-server systems require a central file server and are expensive to implement compared to peer-to-peer systems. On the other hand, in the client-server system, a dedicated file server provides a level of access to the clients, offering increased security compared to p2p systems in which security is managed by the end users. In addition, the performance of p2p

Table 2
Summary of the two overlay membership protocols. Full support for a feature is indicated by the sign √.

	Voulgaris [28]	T-man [29]	Mordacchini [30]	Gossple [24]	Vicinity [25]	Behave [26]	HyRec [31]	Webgc [32]	Cyclades [27]	Crate [33]
Similarity metric	√		√	√	√	√	√	√	√	
Gossip clustering			√	√	√	√		√	√	
Double view	√			√	√	√		√	√	
Data compression				√		√				
Caching	√					√			√	
Semantic clustering	√	√						√		
Used MMP	Cyclon [13]	Jelasity [16]	Cyclon [13]	Brahms [18]	Cyclon [13]	Brahms [18]		Cyclon [13]	Brahms [18]	Spray [20]
Contact server							√		√	√

networks suffers as the number of nodes increases, but client-server systems are more stable and can be scaled up as much as desired. Below we present some relevant peer-to-peer approaches.

KBox (Knowledge Box) [39] is an approach dedicaded to transparently shift the query execution on knowledge graphs to the user or application (i.e., the edge of the network). This RDF publication and consumption architecutre relies on a decentralized architecture and pushes the CPU consumption from the server to the client, while making the query execution twice as fast than traditional client-server SPARQL endpoint architecture. The results of the evaluation show that in terms of disk space, RAM, network traffic and CPU, KBox is more efficient than other server side approaches. They also demonstrate that KBox is more scalable than traditional architectures based on SPARQL endpoints.

Current approaches for sharing and processing RDF datasets suffer from low data availability and query runtime performances. These approaches also present a long and resource-intensive process when consuming data. E.Marx et al. [40] propose a P2P architecture for RDF knowledge sharing and SPARQL query processing. In this model, each node can be a simple client, SPARQL endpoint, LDF-server, or LDF-client and should be able to share the data, as well as querying capabilities. The approach addresses specific challenges related to RDF dataset sharing and processing such as Finding Relevant Peers, Query-Load Balancing, Executing Federated Queries, Data Replication, Data Sharing among peers and Free Riding peers problem.

To address semantic web issues related to high query loads at the data provider's site (SPARQL endpoints) and availability of datasets, PIQNIC(P2p system for Query processiNg over semantIC data) [41] combines both client and server functionality at each peer and introduce replicas. This avoids single points of failure and the data is still available even though the original source is not. PIQNIC is a P2P-based architecture for publishing and querying RDF data. It provides a client that, in addition to providing query access to vast amounts of data, functions as a server maintaining a local datastore. Experimental results show that PIQNIC can serve as an architecture for sharing and querying semantic data, even in the presence of node failures.

FireChat [42, 43] by Open Garden is a messaging software enabling users to communicate offline regardless of Internet connectivity or phone service. FireChat relies upon open-access mesh network and is able to use phone-to-phone bluetooth signals to connect whenever mobile phone coverage is unavailable. However, FireChat does not give users the ability to query their neighbors, or to access data preceding its arrival in the community.

Solid [44] is a decentralized platform for social Web applications. The user's data is stored in a Web-accessible personal online datastore (pod). To share information with others (individuals or organisations), user gives permission to access the appropriate information in his pod. The data in the pod remain owned by the user owner. Querying several users' pods however remains an open question on Solid.

The entity registry system (ERS) [45] aims to replace the Web as a platform for publishing Linked Data when the latter is not available (i.e. in cases where internet connectivity is not guaranteed). It allows for Linked Data without using the centralised components that make up the Web infrastructure. ERS is compatible with the RDF data model. But the availability of data in case of a contributor crash is not guaranteed. When a contributor crashes, its data is no longer available to its neighbors unless it was replicated before the crash.

Snob [46] is a query execution model for SPARQL query over RDF data hosted in a network of browsers. Direct neighbours in the network are the data sources and results received from neighbours are stored lo-

cally as intermediate results. To speed up query execution, browsers processing similar queries are connected through a semantic overlay network.

5. RDF graph sharing and collaborative modification

In this section, we focus on collaborative data sharing systems. We give a theoretical analysis of the technical approaches to ensure local data availability, in particular in relation to network architecture based on gossip protocols and data model oriented on RDF graph replication on a decentralised architecture. In the following sub-sections, we do not look at approaches related to the semantic web in general, but more specifically at work related to collaborative sharing and modification.

5.1. RDF Graph

5.1.1. RDF Data Structure

In an RDF graph, different entities are vertexes in the graph and relationships between them are represented as edges (Example: Figure 3). Information about an entity is represented by directed edges emanating from the vertex of that entity (labeled by the property type), where the edge connects the vertex to other entities, or to special literal vertexes that contain the value of a particular attribute for that entity. The core feature of RDF of extensibility relies on a minimal vocabulary with predefined semantics identified by the RDF namespace and ready to be extended.

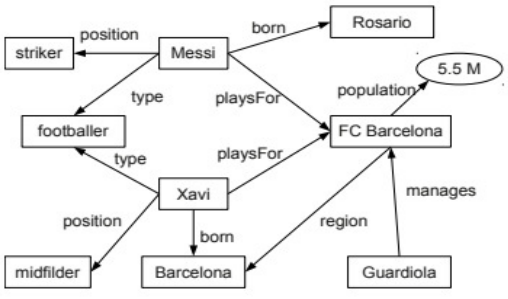

Fig. 3. Example RDF Graph Data from DBpedia [47]. Edges in the graph indicate that the entity ("Messi") is of type "footballer", was born in Rosario, and plays striker for FC Barcelona. Each of the entities that "Messi" is connected to in this graph can have their own set of connections; for example, FC Barcelona is shown to be connected to the Barcelona entity through the region relation.

RDF Schema is a special vocabulary (rdfs namespace), a meta-ontology, that supports the declaration of lightweight vocabularies by providing some elementary bases. The relationship of the data to an RDF specification's schema-defining part is particularly relevant [48]. It is common practice to distinguish an RDF Graph into a data part and a schema part. When looking at the storage [49] and querying [50] in databases, a natural distinction is made from schema definition and data statements. As an example, [48] defines a data subgraph of an RDF Graph G as a maximal subgraph G'satisfying $(\text{subj}(G') \cup \text{obj}(G')) \cap \text{pred}(G') = \phi$. The schema subgraph associated to G' is G\G'.

5.1.2. RDF Graph Storage in collaborative sharing systems

RDF has different representations such as RDF/XML, N-Triples, N3, Turtle and JSON-LD. These syntaxes are generaly used for storing and exchanging RDF data; However, we address here software storage solutions known in collaborative sharing and modification approaches [51–53].

Jena [51] is a Semantic Web Framework, offering a Java programming interface, a database system, and query languages (RDQL, SPARQL) [54, 55]. Jena's original design (Jena-1) used two alternative approaches to store an RDF Graph: (1) Three tables: one for statements, one for literals and one for resources. Here the main problem was the heavy use of joins to answer queries. (2) One statement table, with indexes by subject, by predicate and by object.

Sesame [52] is an architecture for efficient storage and expressive querying of large quantities of data in RDF and RDF Schema. The main feature of Sesame is that it provides query languages, and a subset of RQL which incorporate the RDF Schema semantics. The concrete data storage is implemented differently according to the underlying database system [52]: PostgreSQL and MySQL.

Virtuoso [53] is a multi-protocol server providing access to relational data stored either within Virtuoso itself or any combination of external relational databases. Virtuoso's initial storage solution is fairly conventional: a single table of four columns holds one quad, i.e. triple plus graph per row. Recent version offers three options for each table: partitioned, replicated or local. Partitioning is based on partition columns specified by the administrator, which are used for hash-based partitioning; partitions can also be replicated, if specified. Replication copies the full table to each machine, which can be used for query-based partitioning,

or to store a global schema that is frequently accessed by queries. Local tables are only accessible to the individual machine, and are typically used for local configuration.

Solutions have also been proposed to overcome some of the limitations of RDF. HDT (Header-Dictionary-Triples) [56] is one such illustration. HDT is an RDF publication and serialization format which addresses the limitation of traditional RDF representations such as high levels of verbosity/redundancy and weak machine-processable capabilities of large datasets. HDT decompose RDF dataset into three logical components: Header, Dictionary, and Triples. This decomposition alone leads to space savings of up to 15 times compared to the original representation. Using a specific compressor named HDT-Compress, the solution also realize a size reduction to half of the achieved by traditional compressors. Martínez-Prieto et al. [57] propose to speed up huge RDF graphs consumption by publishing and exchanging RDF serialized in HDT (Header-Dictionary-Triples) binary format. A lightweight post-process (at consumption), caled HDT-FoQ(Focused on Querying)), is then proposed to enhance the HDT representation with additional structures providing a full-index for RDF retrieval. This makes the exchanged RDF data directly and easily retrieval. HDT-FoQ is highly successful in triple pattern resolution, maintains competitiveness in middle-sized dataset joins, and has the potential to improve for larger datasets.

5.1.3. RDF Graph Querying in collaborative sharing systems

For approaches related to collaborative sharing systems, RDF data and RDF schemas can be considered at three different levels of abstraction [52]: at the syntactic level (they are XML documents), structure level (they consist of a set of triples) and semantic level (they constitute one or more graphs with partially predefined semantics).
Clearly, the requirement is for methods to query on the semantic level (i.e: querying the full knowledge and not only explicitly stated assertions). Two options exist to achieve this goal: (1) Calculate and store as the query base the given graph closure, (2) Enable inference of new statements via the query processor as required.

Much work has been done to build infrastructures consisting of centralized nodes (Napster [58], BitTorrent [59], Direct Connect [60], Google Apps like Gmail and Google talk [60]), dedicated to data sharing and capable of ensuring the availability and consistency of these data. These client-server infrastructures of the Web of Data allow the implementation of light clients for users, thus transferring processing and calculation loads to the servers. They are also very suitable (compared to decentralized infrastructures) in terms of speed when searching for information on large amounts of data. Central nodes can cooperate to share knowledge. But the major difficulty in this method is that query results may change over time, depending on node availability. If a node becomes unavailable, it is not relayed by any other node in the network. Thus, any node can be a point of failure for the system [59].

To overcome these centralized constraints, several approaches in the literature consider distributed scenarios based on peer-to-peer networks. The network provides high resistance to technical failures of nodes. These peer-to-peer systems offer many advantages of decentralized distributed systems but suffer from problems related to the availability and reliability of sources and data. To overcome these constraints, decentralized peer-to-peer systems rely on mechanisms to mitigate the impact of disconnection scenarios on data availability, but also to balance the system's processing loads on all peers. In the case of decentralized RDF data sharing systems, these mechanisms allow a graph to be partially or totally replicated, shared and modified in a collaborative way through the peers involved in the architecture. To this end, several solutions have been proposed in the literature adopting each of the following principles, which are conceptually different:

- `Graph replication`: the graph is stored on all peers in the network that can modify and/or delete portions of it independently. These addition or deletion operations will then be propagated through the architecture using variable techniques to ensure the consistency of the different copies.
- `Distributed graph and structured peer-to-peer system (case of DHTs)`: the graph is distributed across all peers on a structured peer-to-peer environment (a DHT). In this case, requests are routed directly to peers holding the requested data.
- `Distributed graph and semantic overlay`: the graph is distributed over all par-

ticipants by adopting a semantic peer-to-peer network architecture. Also here, requests are routed to neighboring nodes and nodes belonging to other clusters if necessary.
- `Cloud and fog computing`: the graph is hosted on the cloud. A collaborative cache on browsers is then set up on the architecture to overcome disconnection scenarios.

As stated in the introduction about the vulnerability of DHT architectures to mobility and crash scenarios we will not treat DHT cases in this chapter given the context targeted by our study with constraints in terms of limited resources, intermittent connectivity and mobility of contributors. Also, for having described the general conception of Distributed graph and semantic overlay approach in section 4.2, we will not treat it in this chapter. Next, we will focus on solutions based on following approaches: Graph Replication and Cloud/Fog Computing.

5.2. Graph replication

Data replication methods are gaining popularity in peer-to-peer computer systems as a way to achieve high availability and reliability. Data replication is now a relevant design principle to ensure reliability, load balancing, but mainly the high availability of data and services in distributed systems. It improves the performance and availability of information sharing in a large-scale network [61]. Data replication consists precisely of storing in separate sites several data object copies. The graph is stored on all (or part of) the peers of the network which can modify and/or delete portions of it autonomously. These addition or deletion operations will then be propagated across the architecture to ensure the consistency of the different replicas. This ensures a high availability of locally relevant data, especially in peer-to-peer architectures where data storage and processing is distributed among peers which may have dynamic behaviors.

5.2.1. Replication criteria

Replica control mechanisms can be classified according to three criteria [61]: where updates take place (single-master or multi-master), when updates are propagated to replicates (synchronous vs asynchronous) and how replicas are distributed over the network (full or partial replication).

- `Update placement`: For Single-master method, each reproduced object has only one primary copy. With this method, only one site is given reading and writing access whereas only reading authorization is granted to others [61]. This is often referred as the master-slave method for master nodes interaction with other "slave" nodes hosting a copy [61]. Centralizing updates on a single point make it easier to manage concurrent access. However, centralization introduces potential bottleneck and point of failure. Multi-master method allows several sites to hold a primary replica of a single object. These copies are all simultaneously updatable. Each site is allowed to update its copies. Multi-master provides more agility over single-master, as in case of a master crash, another one may handle replicas [61].
- `Distribution of replicas`: For replicas placement, there are two basic approaches: full replication and partial replication [62]. With full replication, a shared object is copied to each member's site. It provides load balancing simplicity as all sites offer equal capacity. It also offers maximum availability since any site can replace any other site in case of failure. With partial replication, each site holds a copy of a subset of shared objects, so that the replicated objects may be different from one site to another. This approach requires less storage space on the sites and updates only spread to the sites concerned. Thus, these updates produce a reduced load for the network and sites compared to full replication [62].
- `Update propagation`: Updates can be propagated on a replica control system using a synchronous or asynchronous approach [63]. The synchronous update propagation approach applies changes across replicas within the update producing operations context. Therefore, once operation performed, all replicas have the same state [61]. The source node of the transaction (set of update operations) propagates the update operations in the context of the transaction to all other replicas before executing this transaction. There are several algorithms and protocols to carry out this process [64, 65]. Thus, synchronous propagation requires possible consistency between replicas. With the asynchronous approach, the transaction is first executed at the local site and then updates spread to remote sites. Asynchronous propagation has the advantage that replica unavailability will not interrupt updating

processes, thus improving availability. It is possible to classify asynchronous replication solutions as optimistic or pessimistic based on their assumptions about conflicting updates [66, 67]. Pessimistic asynchronous replication is based on predictions that updating conflicts will occur and introduce propagation methods to prevent it. In contrast, optimistic asynchronous replication is based on the positive hypothesis that conflicting updates arise rarely, if at all. Compared to traditional pessimistic approach, optimistic replication promises higher availability and performance, but lets replicas temporarily diverge and users see inconsistent data. Updates are propagated in the background. Conflicting updates will be processed at the next stage, accepting a certain degree of divergences between sites. Most of these optimistic replication systems ensure possible consistency between replicas [66, 68].

The degree of replication (complete or partial), as well as the source (single or multi Master) and propagation mode (synchronous or asynchronous) of updates in the system are fundamental characteristics for data replication systems.

5.2.2. Selected Graph Replication approaches

BAYOU [69] is a mobile data search solution allowing to reproduce a database on a computer, edit it offline and then synchronize to any other replicas. In Bayou, a single main site determines actions to execute or interrupt and informs other sites.

RDFGrowth [70] focuse on semantic data sharing where a single peer can modify the shared knowledge, while the others have the right to read. RDFGrowth targets a particular scenario where peers participate in interest groups to develop their internal knowledge on one or more thematic areas.

The concept of OT (Operational Transformation) [71–73] has been developed for collaborative publishers. OT is based on the principle that operations are directly executed at the local site and commands are then forwarded to other sites. Therefore, the same operations sequence is executed by all sites, possibly in different orders. OT's objective is to maintain operations intent and ensure replicas converge. This is done by using a rewrite rule for each simultaneous operations pair.

Skaf et al. [74] present the first semantic peer-to-peer wiki, SWOOKI. SWOOKI's network is composed of a set of interconnected autonomous semantic wiki nodes that can join and leave the network dynamically. It is an unstructured and decentralized P2P system that requires no central coordination or knowledge. It is based on a symmetrical communication model where each peer can act as both server and client. Data management is based on optimal data replication where each peer hosts a copy of the wiki pages and the associated semantic data. Each peer can autonomously offer all the services of a semantic wiki server, access, search and requests are executed locally without any routing of requests on the network (full replication).

Weiss et al. [75] present Logoot-Undo CRDT (Commutative Replicated Data Type). A CRDT is a type of data whose operations, when simultaneous, give the same result regardless of the execution order. This is an emerging formalism for optimistic replication. Logoot-Undo is an algorithm that incorporates the *undo anywhere, anytime* functionality. For reasons of efficiency and fault tolerance, the network content is replicated. This replication can be either total or partial. The Logoot-Undo approach belongs to the CRDT framework whose main idea is to provide real commutability between simultaneous operations. An operation is either an insertion or a deletion. Operations are grouped into patches and sent to all other replicas for integration. Each patch is delivered once and only once for each replica.

In **[76], Spaho et al.** consider peer-to-peer systems with an architecture that includes super-peers. A peer group consists of several peers that may be geographically distant from each other but have a common objective. The work is comprised of tasks to be performed by group peers. Replicating peer group documents is a major task. Peer group has a central manager referred to as the super-pair. It assigns tasks to the group's peers and keeps track of how the work is being done. It facilitates interaction with other peer communities. Super-peer connects group peers with other network peers and super-peers. If any portion or all of a peer's document evolves, remaining peers holding the document's replica will apply changes.

Ibanez et al. [77] present SU-Set (SPARQL-Update Set), a CRDT (Commutative Replicated Data Type) for RDF graphs updated with SPARQL Update 1.1 operations. The authors design a CRDT for RDF graphs updated with SPARQL Update 1.1 operations, thus ensuring the possible consistency of a Live Linked Data (LLD) social network. The latter is considered here as a cooperative publishing system with low latency needs and no editing constraints. The CCI (Convergence, Causality, Intention prevention) coherence model of [72] is used. SU-Set extends the insertion and deletion operations of OR-Set [78] (Observed-

Removed Set) to union and difference. In OR-Set, each inserted element is labelled with a unique identifier. This ensures that the elements stored in the payload are always unique, and therefore, added and deleted only once. SPARQL operations on the triplets are performed by the user, then rewritten into SU-Set operations with pairs *(triple, id)* in a transparent manner for the user and sent forward to the other nodes, where they are re-performed at the reception.

Crate [33] is a decentralized, real-time collaborative editor that runs directly in web browsers using WebRTC. To provide document availability and responsiveness, Crate follows the optimistic replication scheme. Each publisher reproduces the document locally and performs directly its operations. Then, the publisher distributes its changes to all other participants. The system is correct if editors with the same set of changes have convergent replicas to an equivalent state, i.e. users read the same document. This property is the strong eventual consistency [79].

5.3. Distributed graph: cloud and fog computing

This sub-section discusses on the evolving paradigm of Cloud Computing, which aims to provide reliable, customized and QoS (Quality of Service) guaranteed dynamic computing environments for end-users. Although this paradigm is mainly driven by the high availability of Internet access, it presents a relevant approach for us which consists in providing power (such as storage, calculation and processing) for end-users via Internet instead of obtaining this power by acquiring hardware and software. We are particularly interested in this approach as it could be envisaged in a local environment with limited Internet access. Mobile contributors could form a local common storage and possibly benefit from power of local super-contributors (i.e more powerful contributors such as storage, calculation and processing) via a peer-to-peer connection.

The size, diversity and increased complexity of RDF reasoning makes it difficult to maintain huge RDF data volumes. Distributed data store architectures are required to overcome volume related challenges. Cloud computing has become a widely adopted paradigm for scalability, fault tolerance and elasticity features offered to many applications, facilitating distributed and parallel architectures deployment. Semantic Web community researchers focus on solving the scalability and performance problems of traditional Semantic Web tools by leveraging cloud computing technologies. The advent of Cloud Computing has paved the way for a distributed ecosystem of RDF triple stores that has the potential to provide very large scale storage and distributed query processing capabilities. The MapReduce paradigm, which is built on Google's file system [80], is the dominant parallel and distributed programming paradigm in the cloud computing community because of its high performance and fault tolerance capability [81]. MapReduce is an programming model for parallel processing of huge data volumes. It is an evolutionary technology welcomed by the scientists. This is used by Google for web indexing, data storage, social networks. Apache also implements MapReduce in the open-source framework Hadoop [82], which is successfully applied to solve data-intensive problems in different domains. This is a distributed file system in which files are stored using replication. Hadoop offers a high degree of reliability and fault-tolerance.

5.3.1. Cloud for distributed RDF data

In recent years, cloud computing provided many opportunities for companies by offering their customers a range of IT services. The current cloud computing *pay-as-you-go* model is becoming an effective alternative to owning and managing private data centers for customers facing web and batch applications processing [83]. Cloud computing frees organizations and their end users from having to plan for many details, such as storage resources, computational limitations and the cost of network communication.

Cloud computing is becoming the general Internet approach to information storage, retrieval and management. At the same time, mobile devices are emerging as the main service applications. Successful integration of cloud computing and mobile devices is therefore the key task of the next generation network. However, this integration faces several fundamental challenges: service agility, real-time response, long-term connection [84]. To address these challenges between cloud and mobile applications, fog computing has recently emerged as a more practical solution to enable seamless convergence between cloud and mobile for content delivery and real-time data processing [85]. Fog computing can address these issues by providing resources and services that are accessible to end users at the edge of the network, while cloud computing is more about providing distributed resources over the main network.

5.3.2. Fog/edge for distributed RDF data

Fog computing is a distributed computing paradigm acting between cloud data centers and devices/sensors (users) as a middle tier [86]. The fog computing concept was introduced by Cisco in 2012 to address the challenges of IoT (Internet of Things) applications in the conventional cloud computing [87]. A fog computing system consists of conventional network devices like routers, switches, decoders, proxy servers and base stations. (Figure 4) placed near peripherals/sensors [86]. These components have various computation, storage, networking, and other features, and can support the execution of service applications. As a result, functional components allow fog based services to build widespread geographic cloud-based service distributions. In addition, fog computing eases positioning support, enhanced mobility, live interaction, interoperability and evolutivity. Thus, it can operate efficiently in terms of service latency, energy consumption, network traffic, capital and operating expenses, content distribution. In this sense, Fog computing better meets the requirements of IoT applications as opposed to the single use of cloud computing [88].

Fog computing is often assimilated to Edge computing particularly due to the fact that Edge takes up the idea of Fog computing, i.e. bringing computing resources closer to end users [89]. However, there is a particular difference that is based on the location of computing resources. Fog computing is based on small data centers spread over different sites located on the periphery of the network [90]. These data centers typically have several servers and provide computing and storage resources to customers located at the edge of the network. Edge computing, on the other hand, allows data processing on the peripheral network, which consists of end devices (such as mobile phones and smart objects), peripherals (such as edge routers, decoders, bridges, base stations, wireless access points) [89].

There are similar concepts such as Mobile Cloud Computing (MCC) and Mobile-Edge Computing (MEC) that identify themselves in Fog concept [86]. MCC describes an infrastructure where the storage and processing of data take place outside of mobile equipments [86]. Mobile cloud-based services transfer processing and data storage power from smartphones to the cloud, providing mobile services and applications not only to the users themselves, but to a wider set of mobile subscribers as well [91]. MEC may be considered like a cloud service operating on a dynamic network and accomplishing particular functions not achievable using a conventional network architecture [92]. In the context of Internet of Things, fog computing appears to be a mix of MCC and MEC, while it stands out as an increasingly promising and widespread computing paradigm.

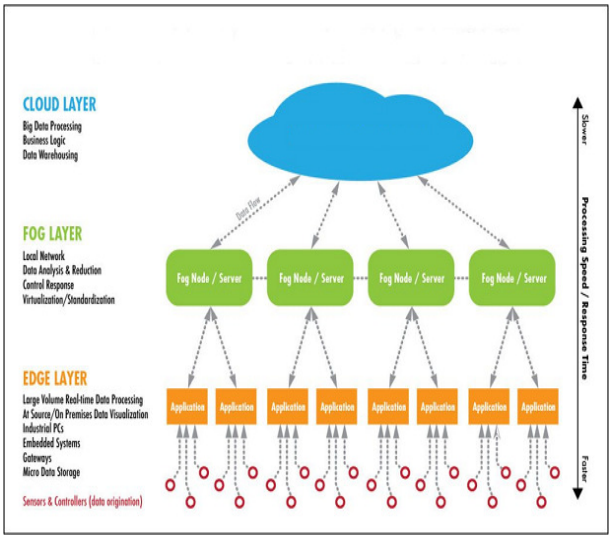

Fig. 4. Example of Fog/Edge architecture [93].

5.3.3. Selected Fog/edge approaches

Cloudlet [94] is considered as an exemplary implementation of resource-rich Fog nodes. Three layers made up its architecture. The lower layer consists of both Linux kernel and cloud data cache, the middle layer is virtualization with a set of cloud software such as OpenStack [95], and the upper layer consists of applications isolated by different virtual machine (VM) instances. Cloudlets have been specifically designed to provide services to mobile users with limited local resources that can act as lightweight clients and access cloud resources that are one-step away via a wireless network.

Zhu et al. [96] apply existing methods for web optimization in an innovative way. In the context of Fog computing, these methods can be combined with unique knowledge that is only available on Fog devices. A more dynamic adaptation to the user's conditions can also be achieved with specific knowledge on the network periphery. Consequently, the rendering performance of a user's web page is improved beyond that obtained by simply applying these methods on the web server.

Hong et al. [97] present a PaaS (Platform as a Service) programming model, called Mobile Fog, that

provides simplified programming abstraction and supports dynamically scaled applications at runtime. In this model, an application consists of distributed mobile Fog processes that are aligned with computing instances distributed over the fog and cloud, as well as various equipment at the network periphery. At runtime each process performs application-specific tasks such as detection and aggregation in relation to its location and level in the network hierarchy. Each Mobile Fog process manages the workload of a certain geospatial region.

DiploCloud [98] is an efficient and scalable distributed RDF data management system for the cloud.Three major structures constitute the system: clusters of RDF molecules considered as hybrid structures using both property tables and RDF sub-graphs, model lists (storage of literals in compact lists as in a column-oriented database system) and an efficient key index that indexes URIs and literals based on the clusters to which they belong. The system design follows the architecture of many modern distributed cloud-based systems (for example, Google's BigTable [99]), where a main node is responsible for interacting with customers and orchestrating operations performed by other nodes (Worker).

Faruque et al. [100] introduce Fog computing as an innovative platform for energy management. The proposed platform uses interoperability, scalability, adaptability and connectivity between intelligent platforms on the fog computing platform, which is a low-power, low-cost device for computing, storage and communication. Energy management or control software is implemented as a service based on Devices Profile for Web Services (DPWS) also used for discovery to provide plug-n-play functionality. This service-oriented architecture also summarizes the heterogeneity of communications and hardware. With a huge number of vehicles in urban areas, bringing underused vehicle resources into service offers an excellent opportunity and value. Hou et al. [101] thus conceive the idea of using vehicles as communication and computing infrastructures, called *Vehicle Fog Computing (VFC)*, which is an architecture that uses a multitude of collaborative customers/end users or onboard devices close to the user to perform communications and calculations, based on a better use of each vehicle's individual communication and computing resources.

Rahman et al. [102] present a fog-based distributed semantic model named Semantic-fog allowing semantic services in the vicinity of IoT devices. The semantic fog structure comprises IoT features in the perception level, fog units in the treatment level, and cloud infrastructure for the treatment and application level. The fog nodes are organized according to their functionalities. Layer 2 fog unit gathers raw sensory data and executes certain functions like filtering and clustering, and passes them on to the higher level fog unit. Layer 1 nodes will obtain this aggregated data of high quality and carry out compilation, modelling and mapping processes, and initiate suitable measures. Then, data are transmitted to remote cloud for storing, viewing or complex treatment.

Mehmood et al. [103] propose a cloud-centric IoT platform for virtual object registration and initialization. For security reasons, permission and control are required for registration procedure. Only authorized persons will be approved by the authorization mechanism to record IoT equipment and prevent unneeded IoT platform exploitation. This differs from traditional IoT platforms as they offer material and application services within a single platform and allow users to connect and use them. RDF is used for finding, exchanging and presenting data on IoT marketplace.

In **[104], Farnbauer-Schmid et al.** introduce the Semantic Edge Computing Runtime (SECR), an edge computing tool developed to provide a background for IoT peripherals. SECR combines two concepts: data integration to support the diversity of IoT applications, and edge technology to minimize data volume to be transmitted for distant processing. By using data integration, edge processing can be carried out at a greater degree of abstraction. All outputs from SECR services are available as RDF graphs allowing edge level interoperability.

5.4. Synthesis on collaborative graph sharing and modification approaches

The four concepts previously presented (Graph Replication, Distributed Graph and DHT, Distributed Graph and Semantic Overlay, Cloud and Fog Computing) constitute a set of approaches that are relevant for decentralized architectures. Their use in the design of mobile solutions for sharing and contributing Data to the Web could solve the problem of high availability of relevant data in contexts where access to remote sources is constrained by limited connectivity to Internet. The approach of intelligent replication of the graph on the architecture is particularly interesting in this context. The challenge then concerns the preservation of data consistency as update operations are produced at the contributing nodes. Opti-

mistic asynchronous replication effectively addresses this challenge with the assumption that there are few conflicting updates, which, if they occur, are processed at the end of the process. This means that the system tolerates a certain level of divergence between replicas. This compromise may be acceptable for collaborative sharing and contribution solutions. Table 3 summarizes different solutions presented. These models are compared according to the following criteria:

- `Data type`: Exchanged data types.
- `DHT`: The architecture relies on DHT.
- `Partial replica`: The system adopts a partial replication mechanism on the graph.
- `Semantic Overlay`: The system builds a semantic overlay on the nodes.
- `Total replica`: The system adopts a total replication mechanism on the graph.
- `Fog layer`: The architecture has a Fog layer.

The literature shows that the approaches identified above (subsections 4.2 and 4.3) have been successfully applied on architectures based on gossip protocols. The decentralized peer-to-peer architecture is built using the underlying gossip protocol that also ensures connectivity despite node arrivals and departures. The protocol also manages, in some cases, the exchange of update operations between nodes. The data structure is then hosted on the architecture according to one of the previous approaches. We can cite a few examples.

In Crate [33], a decentralized real-time collaborative editor that runs directly on web browsers using WebRTC, the gossip Spray protocol builds and maintains a mesh of contributing web browsers. Spray provides each contributor with a local neighborhood table (the view) that allows communication in a subset of editors. Crate adopts the graph replication approach. The graph represents a document shared by all publishers (contributors). Each publisher reproduces the document locally and performs its operations directly. To also ensure the consistency of the graph on each editor, Crate uses the SPRAY protocol to distribute all update operations to all collaborators in an evolutionary way.

Swooki [74] is a peer-to-peer semantic wiki that combines the wiki approach of ontologies such as Semantic MediaWiki [105] and a peer-to-peer wiki based on total replication and CCI (convergence, causality preservation and intention preservation) model such as Wooki [106]. On the one hand, Swooki is based on the graph replication approach. The graph here represents a semantic wiki page (combination of text and RDF data). On the other hand, update operations will be routed to collaborating nodes using the gossip protocol [107] combined with an anti-entropy protocol [108].

Voulgaris et al. [109] use the semantic superposition approach to exploit the semantic structure present in document sharing systems to improve search performances. They propose an architecture with contributor nodes that are semantically close. Each node maintains a list of semantic neighbors to which requests are submitted first, before using a default search mechanism if no semantic neighbors can respond to the request. To build and maintain the semantic neighborhood, the authors assume the existence of a peer-to-peer system supporting semantic searches. This system is then built with the SCAMP gossip protocol that generates an unstructured overlay network.

6. Discussion and Conclusion

Our objective in this paper was to survey the existing solutions (such as models and architectures) that could contribute to solve the problem of intermittent access to the Web of Data by mobile contributors. First of all, we confirmed that gossip protocols offer well adapted approaches to the design and maintenance of decentralized and dynamic peer-to-peer architectures. Consequently, our prospection was guided in its first part by the analysis of solutions based on gossip protocols dedicated to the design and management of peer-to-peer overlay networks, and then to the analysis of approaches, dedicated to data sharing systems construction according to the RDF data model.

For architectures based on a gossip protocol, we distinguished two architectures types. The first *(gossip protocols with a basic adhesion mechanism)* refers to basic systems dedicated to designing and maintaining the connectivity of the underlying architecture on which the various exchanges between peer members are based. The second type of architecture *(gossip protocols with two overlay adhesion mechanisms)* is dedicated to the automatic formation of clusters of interest or proximity aimed to group together peer members of the architecture that have one or more common interests or that are close according to a given metric of similarity. These two overlay systems are built on top of basic protocols. We also note the possibility of taking into account the location of peers when selecting neighbours [25]. The neighbourhood will then be formed by semantically close peers, having a certain

Table 3
Summary table of solutions dealing with mobile access to data: The support of a characteristic is indicated by the sign √

	Semantic-fog [102]	SECR [104]	Swooki [74]	Cloudlet [94]	Logoot-Undo [75]	Crate [33]	DiploCloud [98]
Data Type	RDF	RDF	RDF	file	file	file	RDF
DHT					√		
Partial Replica	√	√			√		√
Total Replica			√		√	√	
Fog Layer	√	√		√			√

geographical proximity, i.e. a maximum limit in terms of geographical distance is set between the peers considered as neighbours. Based on parameters such as *fanout, view size,* and *node degree,* gossip protocols are generally evaluated by comparing the different architecture behaviors at runtime. Protocol properties such as *failure resistance, convergence time,* and *load distribution, graph properties (clustering coefficient, average of shortest paths, balanced distribution)*, are analyzed and interpreted following test scenarios on simulation platforms such as PeerSim [110].

The works surveyed in this document present many trends relevant for future work on gossip protocols. Among them, we especially identified: the consideration of the geographical parameter, in particular by applying a deterministic selection during sampling and exchanges; the integration of the adaptive *fanout* to take into account the evolution of the architecture size; the implementation of JavaScript solutions to facilitate the deployment of protocols and peers inside Web browsers; and the exploitation of the advantages of explicit friends' social networks.

The structured analysis of approaches dedicated to the construction of data sharing systems is based on the layout of the architecture graph, i.e. whether it is replicated (partially or totally), or distributed, or shared and modified in a collaborative way through the involved peers in the architecture. Existing solutions are generally boosted by local caching, source indexing and synchronization mechanisms. This last element raises the issue of data consistency during system execution, in particular the divergence of states between remote and local sources when all synchronization operations are performed. To overcome this constraint, in some cases, such as [74], the CCI (Convergence, Causality Preservation and Intention preservation) model is used to ensure system coherence. This model allows the system to maintain the following three properties [72]: Convergence: when the same set of operations is executed on all sites, they will all have the same state; Causality: if an operation *O* is executed before another operation *O'*, the same execution order is respected on all sites ; Intent: for any operation *O*, the effects of the execution of *O* on all sites are the same as the intentions of *O*, and the effect of the execution of *O* does not change the effects of the independent operations. Other solutions [109, 111, 112] also rely on mechanisms for caching relevant data according to the metadata of peer requests. Replacement (or deletion) policies identify which data to move to persistent storage or permanently delete for proper cache management.

Using metrics such as complexity in time and space and the traffic effect on network architecture, the proposed solutions are compared by analyzing their performance in relation to their own properties. These include data quality metrics (exhaustiveness, conciseness, consistency), cycle number, query load, scalability. Here, we also noted some interesting points for future work. They can be summarised mainly in two directions. The first concerns the problem of synchronization between local and remote sources. It targets the design of relevant mechanisms to improve the consistency and reliability of the exchanged data. The second direction aims to ease the constraints associated with the implementation of CRDT (Commutative Replicated Data Type) by eliminating the requirement for the underlying network to ensure causal ownership.

To conclude, we consider that to achieve a connected architecture that provides mobile contributors with local access to the Web of data, several technological approaches should be combined. On the one hand, to build and maintain the connected architecture, gossip protocols with two overlay adhesion mechanisms are more efficient for building connected overlay networks of mobile contributors. On the other hand, to ensure local access to data, we believe that optimistic replication systems are suitable for environments where access to the Web of data is intermittent.

Data consistency can be ensured by following the CCI model. One additional relevant approach that we can use in future work is cooperative cache systems. These systems are also relevant to ensure local access, especially in cases where web browsers represent the contributors. We believe this survey shows that there is a very promising domain of research and an establish set of existing work and research directions to propose innovative and original new ways of sharing linked data even in technologically limited environments or with the goal of reducing our technological footprints.

References

[1] T. Berners-Lee and J. Timothy, Information management: A proposal, Technical Report, 1989.

[2] T. Berners-Lee, W3 future directions, 1994. https://www.w3.org/Talks/WWW94Tim/.

[3] T. Berners-Lee, Linked Data, 2006.

[4] C. Bizer, T. Heath and T. Berners-Lee, Linked data: The story so far, in: *Semantic services, interoperability and web applications: emerging concepts*, IGI Global, 2011, pp. 205–227.

[5] I.W. Stats, Internet World Stats, 2021.

[6] GSMA, The Mobile Economy Sub-Saharan Africa, 2020.

[7] A. Ismail, Communautés dans les réseaux sémantiques pairs-à-pairs, PhD thesis, Aix-Marseille 2, 2010.

[8] Z.G. Ives, A.Y. Halevy, P. Mork and I. Tatarinov, Piazza: mediation and integration infrastructure for semantic web data, *Journal of Web Semantics* **1**(2) (2004), 155–175.

[9] D. Kempe, A. Dobra and J. Gehrke, Gossip-based computation of aggregate information, in: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, IEEE, 2003, pp. 482–491.

[10] A. Montresor, M. Jelasity and O. Babaoglu, Robust aggregation protocols for large-scale overlay networks, in: *International Conference on Dependable Systems and Networks, 2004*, IEEE, 2004, pp. 19–28.

[11] M. Jelasity, A. Montresor and O. Babaoglu, Gossip-based aggregation in large dynamic networks, *ACM Transactions on Computer Systems (TOCS)* **23**(3) (2005), 219–252.

[12] M. Jelasity, R. Guerraoui, A.-M. Kermarrec and M. Van Steen, The peer sampling service: Experimental evaluation of unstructured gossip-based implementations, in: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer, 2004, pp. 79–98.

[13] S. Voulgaris, D. Gavidia and M. Van Steen, Cyclon: Inexpensive membership management for unstructured p2p overlays, *Journal of Network and systems Management* **13**(2) (2005), 197–217.

[14] A.-M. Kermarrec, L. Massoulié and A.J. Ganesh, Probabilistic reliable dissemination in large-scale systems, *IEEE Transactions on Parallel and Distributed systems* **14**(3) (2003), 248–258.

[15] A.J. Ganesh, A.-M. Kermarrec and L. Massoulié, Peer-to-peer membership management for gossip-based protocols, *IEEE transactions on computers* **52**(2) (2003), 139–149.

[16] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec and M. Van Steen, Gossip-based peer sampling, *ACM Transactions on Computer Systems (TOCS)* **25**(3) (2007), 8.

[17] J. Leitao, J. Pereira and L. Rodrigues, HyParView: A membership protocol for reliable gossip-based broadcast, in: *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, IEEE, 2007, pp. 419–429.

[18] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot and A. Shraer, Brahms: Byzantine resilient random membership sampling, *Computer Networks* **53**(13) (2009), 2340–2359.

[19] A. Stavrou, D. Rubenstein and S. Sahu, A lightweight, robust P2P system to handle flash crowds, *IEEE Journal on Selected Areas in Communications* **22**(1) (2004), 6–17.

[20] B. Nédelec, J. Tanke, D. Frey, P. Molli and A. Mostefaoui, Spray: an Adaptive Random Peer Sampling Protocol, PhD thesis, LINA-University of Nantes; INRIA Rennes-Bretagne Atlantique, 2015.

[21] A. Alromih and H. Kurdi, An energy-efficient gossiping protocol for wireless sensor networks using Chebyshev distance, *Procedia Computer Science* **151** (2019), 1066–1071.

[22] Wikipedia contributors, Chebyshev distance — Wikipedia, The Free Encyclopedia, 2019, [Online; accessed 26-February-2020].

[23] S. Dahal et al., Effect of different distance measures in result of cluster analysis (2015).

[24] M. Bertier, D. Frey, R. Guerraoui, A.-M. Kermarrec and V. Leroy, The gossple anonymous social network, in: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer, 2010, pp. 191–211.

[25] S. Voulgaris and M. Van Steen, Vicinity: A pinch of randomness brings out the structure, in: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer, 2013, pp. 21–40.

[26] D. Frey, M. Goessens and A.-M. Kermarrec, Behave: Behavioral cache for web content, in: *IFIP International Conference on Distributed Applications and Interoperable Systems*, Springer, 2014, pp. 89–103.

[27] P. Folz, H. Skaf-Molli and P. Molli, CyCLaDEs: a decentralized cache for Linked Data Fragments, in: *ESWC: Extended Semantic Web Conference*, 2016.

[28] S. Voulgaris and M. Van Steen, Epidemic-style management of semantic overlays for content-based searching, in: *European Conference on Parallel Processing*, Springer, 2005, pp. 1143–1152.

[29] M. Jelasity, A. Montresor and O. Babaoglu, T-man: Gossip-based fast overlay topology construction, *Computer networks* **53**(13) (2009), 2321–2339.

[30] M. Mordacchini, R. Baraglia, P. Dazzi and L. Ricci, A p2p recommender system based on gossip overlays (prego), in: *2010 10th IEEE International Conference on Computer and Information Technology*, IEEE, 2010, pp. 83–90.

[31] A. Boutet, D. Frey, R. Guerraoui, A.-M. Kermarrec and R. Patra, Hyrec: Leveraging browsers for scalable recommenders, in: *Proceedings of the 15th International Middleware Conference*, ACM, 2014, pp. 85–96.

[32] R. Carvajal-Gómez, D. Frey, M. Simonin and A.-M. Kermarrec, Webgc gossiping on browsers without a server, in: *International Conference on Web Information Systems Engineering*, Springer, 2015, pp. 332–336.

[33] B. Nédelec, P. Molli and A. Mostefaoui, Crate: Writing stories together with our browsers, in: *Proceedings of the 25th International Conference Companion on World Wide Web*, International World Wide Web Conferences Steering Committee, 2016, pp. 231–234.

[34] A.B. Pilet, D. Frey and F. Taiani, Robust Privacy-Preserving Gossip Averaging, in: *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, Springer, 2019, pp. 38–52.

[35] C.S. Meiklejohn, H. Miller and P. Alvaro, {PARTISAN}: Scaling the Distributed Actor Runtime, in: *2019 USENIX Annual Technical Conference (USENIX 19)*, 2019, pp. 63–76.

[36] Wikipedia contributors, Actor model — Wikipedia, The Free Encyclopedia, 2020, [Online; accessed 26-February-2020].

[37] J. Leitao, J. Pereira and L. Rodrigues, Epidemic broadcast trees, in: *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*, IEEE, 2007, pp. 301–310.

[38] E. Leonardi, M. Mellia, C. Kiraly, R.L. Cigno, S. Niccolini and J. Seedorf, Network Friendly P2P Streaming: The NAPA-WINE Architecture (2020).

[39] E. Marx, C. Baron, T. Soru and S. Auer, KBox—Transparently Shifting Query Execution on Knowledge Graphs to the Edge, in: *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, IEEE, 2017, pp. 125–132.

[40] E. Marx, M. Saleem, I. Lytra and A.-C.N. Ngomo, A decentralized architecture for SPARQL query processing and RDF sharing: a position paper, in: *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, IEEE, 2018, pp. 274–277.

[41] C. Aebeloe, G. Montoya and K. Hose, A decentralized architecture for sharing and querying semantic data, in: *European Semantic Web Conference*, Springer, 2019, pp. 3–18.

[42] Hong Kong Protests Propel FireChat Phone-to-Phone App, Hong Kong Protests Propel FireChat Phone-to-Phone App, 2014, [Online; accessed 22-May-2020].

[43] FireChat – the messaging app, FireChat – the messaging app, 2014, [Online; accessed 22-May-2020].

[44] A.V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga and T. Berners-Lee, Solid: A platform for decentralized social applications based on linked data, *MIT CSAIL & Qatar Computing Research Institute, Tech. Rep.* (2016).

[45] M. Charlaganov, P. Cudré-Mauroux, C. Dinu, C. Guéret, M. Grund and T. Macicas, The entity registry system: Implementing 5-star linked data without the web, *arXiv preprint arXiv:1308.3357* (2013).

[46] A. Grall, H. Skaf-Molli and P. Molli, SPARQL query execution in networks of web browsers, in: *THE 17TH INTERNATIONAL SEMANTIC WEB CONFERENCE, WORKSHOP ON DECENTRALIZING THE SEMANTIC WEB*, 2018.

[47] J. Huang, D.J. Abadi and K. Ren, Scalable SPARQL querying of large RDF graphs, *Proceedings of the VLDB Endowment* **4**(11) (2011), 1123–1134.

[48] J. Hayes, A graph model for RDF, *Darmstadt University of Technology/University of Chile* (2004).

[49] A. Matono, T. Amagasa, M. Yoshikawa and S. Uemura, An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays., in: *SWDB*, 2003, pp. 151–168.

[50] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis and M. Scholl, RQL: a declarative query language for RDF, in: *Proceedings of the 11th international conference on World Wide Web*, ACM, 2002, pp. 592–603.

[51] B. McBride, Jena: Implementing the rdf model and syntax specification, in: *Proceedings of the Second International Conference on Semantic Web-Volume 40*, CEUR-WS. org, 2001, pp. 23–28.

[52] J. Broekstra, A. Kampman and F. Van Harmelen, Sesame: A generic architecture for storing and querying rdf and rdf schema, in: *International semantic web conference*, Springer, 2002, pp. 54–68.

[53] O. Erling and I. Mikhailov, Virtuoso: RDF support in a native RDBMS, in: *Semantic Web Information Management*, Springer, 2010, pp. 501–519.

[54] J.J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne and K. Wilkinson, Jena: implementing the semantic web recommendations, in: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, ACM, 2004, pp. 74–83.

[55] K. Wilkinson, C. Sayers, H. Kuno and D. Reynolds, Efficient RDF storage and retrieval in Jena2, in: *Proceedings of the First International Conference on Semantic Web and Databases*, Citeseer, 2003, pp. 120–139.

[56] J.D. Fernández, M.A. Martínez-Prieto, C. Gutiérrez, A. Polleres and M. Arias, Binary RDF representation for publication and exchange (HDT), *Journal of Web Semantics* **19** (2013), 22–41.

[57] M.A. Martínez-Prieto, M. Arias Gallego and J.D. Fernández, Exchange and consumption of huge RDF data, in: *Extended Semantic Web Conference*, Springer, 2012, pp. 437–452.

[58] S. Saroiu, K.P. Gummadi and S.D. Gribble, Measuring and analyzing the characteristics of Napster and Gnutella hosts, *Multimedia systems* **9**(2) (2003), 170–184.

[59] L. Liu and N. Antonopoulos, From client-server to p2p networking, in: *Handbook of Peer-to-Peer Networking*, Springer, 2010, pp. 71–89.

[60] M. Rogers and S. Bhatti, How to disappear completely: A survey of private peer-to-peer networks, *networks* **13** (2007), 14.

[61] E. Spaho, L. Barolli and F. Xhafa, Data replication strategies in P2P systems: A survey, in: *2014 17th International Conference on Network-Based Information Systems*, IEEE, 2014, pp. 302–309.

[62] V. Martins, Data replication in P2P systems, PhD thesis, Université de Nantes, 2007.

[63] J. Gray, P. Helland, P. O'Neil and D. Shasha, The dangers of replication and a solution, *ACM SIGMOD Record* **25**(2) (1996), 173–182.

[64] B. Kemme and G. Alonso, A new approach to developing and implementing eager database replication protocols, *ACM Transactions on Database Systems (TODS)* **25**(3) (2000), 333–379.

[65] P.A. Bernstein, V. Hadzilacos and N. Goodman, Concurrency control and recovery in database systems (1987).

[66] Y. Saito and M. Shapiro, Optimistic replication, *ACM Computing Surveys (CSUR)* **37**(1) (2005), 42–81.

[67] E. Pacitti, P. Minet and E. Simon, Fast algorithms for maintaining replica consistency in lazy master replicated databases, PhD thesis, INRIA, 1999.

[68] M. Shapiro, K. Bhargavan and N. Krishna, A constraint-based formalism for consistency in replicated systems, in: *International Conference On Principles Of Distributed Systems*, Springer, 2004, pp. 331–345.
[69] D.B. Terry, M.M. Theimer, K. Petersen, A.J. Demers, M.J. Spreitzer and C.H. Hauser, Managing update conflicts in Bayou, a weakly connected replicated storage system, in: *SOSP*, Vol. 95, 1995, pp. 172–182.
[70] G. Tummarello, C. Morbidoni, J. Petersson, P. Puliti and F. Piazza, RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications., *P2PKM* **108** (2004).
[71] C.A. Ellis and C. Sun, Operational transformation in real-time group editors: issues, algorithms, and achievements, in: *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, Citeseer, 1998, pp. 59–68.
[72] C. Sun, X. Jia, Y. Zhang, Y. Yang and D. Chen, Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems, *ACM Transactions on Computer-Human Interaction (TOCHI)* **5**(1) (1998), 63–108.
[73] C.A. Ellis and S.J. Gibbs, Concurrency control in groupware systems, in: *Acm Sigmod Record*, Vol. 18, ACM, 1989, pp. 399–407.
[74] H. Skaf-Molli, C. Rahhal and P. Molli, Peer-to-peer semantic wikis, in: *International Conference on Database and Expert Systems Applications*, Springer, 2009, pp. 196–213.
[75] S. Weiss, P. Urso and P. Molli, Logoot-undo: Distributed collaborative editing system on p2p networks, *IEEE transactions on parallel and distributed systems* **21**(8) (2010), 1162–1174.
[76] E. Spaho, A. Barolli, F. Xhafa and L. Barolli, P2P data replication: Techniques and applications, in: *Modeling and Processing for Next-Generation Big-Data Technologies*, Springer, 2015, pp. 145–166.
[77] L.D. Ibáñez, H. Skaf-Molli, P. Molli and O. Corby, Live linked data: synchronising semantic stores with commutative replicated data types., *IJMSO* **8**(2) (2013), 119–133.
[78] M. Shapiro, N. Preguiça, C. Baquero and M. Zawirski, Conflict-free replicated data types, in: *Symposium on Self-Stabilizing Systems*, Springer, 2011, pp. 386–400.
[79] M. Shapiro, N. Preguiça, C. Baquero and M. Zawirski, A comprehensive study of convergent and commutative replicated data types, PhD thesis, Inria–Centre Paris-Rocquencourt; INRIA, 2011.
[80] S. Ghemawat, H. Gobioff and S.-T. Leung, The Google file system (2003).
[81] J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters, *Communications of the ACM* **51**(1) (2008), 107–113.
[82] K. Shvachko, H. Kuang, S. Radia, R. Chansler et al., The hadoop distributed file system., in: *MSST*, Vol. 10, 2010, pp. 1–10.
[83] A.D. JoSEP, R. KAtz, A. KonWinSKi, L. Gunho, D. PAttER-Son and A. RABKin, A view of cloud computing, *Communications of the ACM* **53**(4) (2010).
[84] T.H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei and L. Sun, Fog computing: Focusing on mobile users at the edge, *arXiv preprint arXiv:1502.01815* (2015).
[85] I. Stojmenovic, S. Wen, X. Huang and H. Luan, An overview of fog computing and its security issues, *Concurrency and Computation: Practice and Experience* **28**(10) (2016), 2991–3005.
[86] S. Kosta, A. Aucinas, P. Hui, R. Mortier and X. Zhang, Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, in: *2012 Proceedings IEEE Infocom*, IEEE, 2012, pp. 945–953.
[87] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, Fog computing and its role in the internet of things, in: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, ACM, 2012, pp. 13–16.
[88] S. Sarkar, S. Chatterjee and S. Misra, Assessment of the Suitability of Fog Computing in the Context of Internet of Things, *IEEE Transactions on Cloud Computing* **6**(1) (2015), 46–59.
[89] R. Mahmud, R. Kotagiri and R. Buyya, Fog computing: A taxonomy, survey and future directions, in: *Internet of everything*, Springer, 2018, pp. 103–130.
[90] B. Confais, Conception d'un système de partage de données adapté à un environnement de Fog Computing, PhD thesis, Université de Nantes, 2018.
[91] H.T. Dinh, C. Lee, D. Niyato and P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, *Wireless communications and mobile computing* **13**(18) (2013), 1587–1611.
[92] M. Patel et al., Mobile-Edge Computing-Introductory Technical White Paper, ETSI MEC white paper, Technical Report, V1 18-09-14, 36 pages, 2014.
[93] WINSYSTEMS, Cloud, Fog And Edge Computing – What's The Difference?, 2017.
[94] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, The case for vm-based cloudlets in mobile computing, *IEEE pervasive Computing* (2009), 14–23.
[95] R.C. Computing, OpenStack Cloud Software, 2012.
[96] J. Zhu, D.S. Chan, M.S. Prabhu, P. Natarajan, H. Hu and F. Bonomi, Improving web sites performance using edge servers in fog computing architecture, in: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, IEEE, 2013, pp. 320–323.
[97] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder and B. Koldehofe, Mobile fog: A programming model for large-scale applications on the internet of things, in: *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, ACM, 2013, pp. 15–20.
[98] M. Wylot and P. Cudré-Mauroux, Diplocloud: Efficient and scalable management of rdf data in the cloud, *IEEE Transactions on Knowledge and Data Engineering* **28**(3) (2016), 659–674.
[99] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes and R.E. Gruber, Bigtable: A distributed storage system for structured data, *ACM Transactions on Computer Systems (TOCS)* **26**(2) (2008), 4.
[100] M.A. Al Faruque and K. Vatanparvar, Energy management-as-a-service over fog computing platform, *IEEE internet of things journal* **3**(2) (2016), 161–169.
[101] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin and S. Chen, Vehicular fog computing: A viewpoint of vehicles as the infrastructures, *IEEE Transactions on Vehicular Technology* **65**(6) (2016), 3860–3873.
[102] H. Rahman and M.I. Hussain, Fog-based semantic model for supporting interoperability in IoT, *IET Communications* **13**(11) (2019), 1651–1661.
[103] F. Mehmood, S. Ahmad and D. Kim, Design and Implementation of an Interworking IoT Platform and Marketplace in Cloud of Things, *Sustainability* **11**(21) (2019), 5952.

[104] M. Farnbauer-Schmidt, J. Lindner, C. Kaffenberger and J. Albrecht, Combining the Concepts of Semantic Data Integration and Edge Computing, *INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik–Informatik für Gesellschaft* (2019).

[105] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller and R. Studer, Semantic wikipedia, in: *Proceedings of the 15th international conference on World Wide Web*, ACM, 2006, pp. 585–594.

[106] S. Weiss, P. Urso and P. Molli, Wooki: a p2p wiki-based collaborative writing tool, in: *International Conference on Web Information Systems Engineering*, Springer, 2007, pp. 503–512.

[107] P.T. Eugster, R. Guerraoui, S.B. Handurukande, P. Kouznetsov and A.-M. Kermarrec, Lightweight probabilistic broadcast, *ACM Transactions on Computer Systems (TOCS)* **21**(4) (2003), 341–374.

[108] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart and D. Terry, Epidemic algorithms for replicated database maintenance, *ACM SIGOPS Operating Systems Review* **22**(1) (1988), 8–32.

[109] S. Voulgaris, A.-M. Kermarrec and L. Massoulié, Exploiting semantic proximity in peer-to-peer content searching, in: *Proceedings. 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. FTDCS 2004.*, IEEE, 2004, pp. 238–243.

[110] A. Montresor and M. Jelasity, PeerSim: A scalable P2P simulator, in: *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, IEEE, 2009, pp. 99–100.

[111] I. Clarke, O. Sandberg, B. Wiley and T.W. Hong, Freenet: A distributed anonymous information storage and retrieval system, in: *Designing privacy enhancing technologies*, Springer, 2001, pp. 46–66.

[112] J.X. Parreira, S. Michel and G. Weikum, p2pDating: Real life inspired semantic overlay networks for web search, *Information Processing & Management* **43**(3) (2007), 643–664.

[113] W.W.W. Consortium et al., RDF 1.1 concepts and abstract syntax (2014).

[114] D. Beckett and B. McBride, RDF/XML syntax specification (revised), *W3C recommendation* **10**(2.3) (2004).

[115] Deloitte, Deloitte, 2015.

[116] R.L. Grossman, The case for cloud computing, *IT professional* **11**(2) (2009), 23–27.