Peer-to-Peer Networking and Applications (2020) 13:872–889
https://doi.org/10.1007/s12083-019-00817-0

Novel Distributed Dynamic Backbone-based Flooding in Unstructured Networks

Saeed Saeedvand[1] · Hadi S. Aghdasi[1] · Leili Mohammad Khanli[1]

Received: 15 March 2019 / Accepted: 9 September 2019 / Published online: 18 November 2019

Abstract
Resource discovery on different unstructured and dynamic networks such as grid, peer-to-peer, and cloud networks is an inevitable challenging issue. The primary method for resource discovery on the unstructured networks is flooding a query on the network. All existing flooding algorithms for unstructured networks generate almost high additional duplicated queries. This high duplication of the unstructured networks causes a lot of network traffic. This paper, therefore, proposes a novel flexible Distributed Dynamic backbone-based Flooding (DDBF) algorithm for distributed unstructured networks. This paper explores Grid middleware, Peer-to-Peer (P2P) paradigm, and cloud networks resource discovery requirements and it proposes flooding algorithm based on the P2P networks using simulation. To evaluate and prove DDBF algorithm we, first, evaluated it on four fixed network topologies along with two different query flooder distributions, then we evaluated it with one dynamic network topology. The performance of the proposed DDBF algorithm was assessed with five different metrics. The result showed a dramatic decrease in the number of engaged flooder nodes, the number of duplicated queries and consequently, network delay compared with the state-of-the-art algorithms.

Keywords Resource Discovery · Flooding · Unstructured Dynamic Networks · Peer-to-Peer Networks · Distributed Algorithm

1 Introduction

The ultimate target of any data or resource sharing networks is to make large sets of resources and make them available to their deployed applications and users. A fundamental service in these networks is resource location discovery. In resource sharing networks, after specifying the existing resources, the system returns the locations where the required resources currently exist in [1]. The ultimate goal of Grid middleware, Peer-to-Peer (P2P) networks, and microdata centers on cloud computing is the use of resources across multiple domains. These systems evolved from different communities and served different needs. Traditional grid middleware, P2P, and MDCs are mostly based on the wired network resources owned by various institutions. These infrastructures are structured in virtual organizations, which are subjected to specific sharing policies. Thus, apart from their differences, these networks have many common characteristics such as resource discovery, dynamic behavior, and heterogeneity of the involved components. Hence, resource location discovery is a key issue for these kinds of networks in which applications are composed of hardware and software resources that need to be located [2–4].

Based on the literature, we can categorize described networks into two different main architectures including structured networks and unstructured networks [5–7] as below:

Structured networks: These types of networks have some central directory servers. It means that the set of links (connections) between nodes are controlled, and resources are placed not at random nodes but at specified locations. These locations make information discovery easy to satisfy. For information discovery in structured networks, there is no need to flood a query on the network. Thus, searching for resources and information on these kinds of cloud networks are informed search [8]. It is a well-known issue that the structured cloud networks are required to have up-to-date central directory servers,

✉ Hadi S. Aghdasi
aghdasi@tabrizu.ac.ir

Saeed Saeedvand
saeedvand@tabrizu.ac.ir

Leili Mohammad Khanli
l-khanli@tabrizu.ac.ir

1 Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

which requires a very high cost. Structured networks are quite widespread in the research literature based on the different network structures such as grid middleware or cloud networks, but because of the dynamic nature (nodes enter and leave the network rapidly) most of the networks, it is hard to use such networks widely [9].

Unstructured networks: These types of networks don't use centralized directory nor any accurate control over the network topology or file placement [10]. Searching resources and information in this kind of networks is known as blind search [7, 11]. In these networks, the connectivity change and nodes can enter and leave the network rapidly. Gnutella is an example of such a network [12]. For blind search, the most typical searching method is flooding, where a query is broadcasting to all adjacent nodes within a certain radius [13]. Unstructured networks are extremely resilient to nodes entering and leaving the network. For instance, platforms such as MapReduce are increasingly popular for their simplicity, scalability, and flexibility on unstructured networks.

All of the unstructured networks use resource sharing systems, where resource discovery is an inevitable challenge. Thus, due to the lack of an underlying structure in these networks, there is no information about the location of files. Thus the prevailing resource location method is "flooding". However, in the unstructured networks the current query flooding algorithms are generating a high number of duplicated queries, which causes large loads on the network. In the last years, for flooding query in unstructured networks, different algorithms have been proposed. Random Walk [14], Query Forwarding in geographically distributed search engines [15], a lightweight information-sharing scheme [4] and some others in [16, 17] are the most important ones.

The existing flooding algorithms have some drawbacks such as generating a large amount of redundancy on the network which causes latency. Reducing latency on the unstructured networks is very important, and researchers repeatedly worked on this problem [4, 9, 18–20]. So it is necessary to decrease network load and consequently network latency on the unstructured networks. Therefore, in this paper, we propose a Distributed Dynamic backbone-based Flooding (DDBF) algorithm. DDBF algorithm is aimed at decreasing the number of engaging nodes in the flooding process to decrease load and consequently network latency in comparison to state-of-the-art flooding algorithms. DDBF algorithm proposes a reliable approach to create a novel flexible backbone on the dynamic unstructured networks for flooding queries.

In this regard, we focus on the Gnutella-like unstructured P2P systems. We do so because (1) most of the grid systems are based on the structured P2P approach [3]. Therefore, by providing flooding method on P2P systems, we can employ it on the grid middleware as well as MDCs. (2) measurement data suggests that P2P applications have a very significant and rapidly growing impact on internet traffic [21]. (3) P2P networks are actively used by a large community of users [7], (4) existing flooding-based query algorithms for discovery resources in these networks still generate a large amount of traffic and large systems quickly become overwhelmed by the query induced load [22]. (5) our proposed algorithm can be evaluated on P2P network topologies easily, and (6) P2P networks are very attractive for certain applications of unstructured networks because it requires no centralized directories and no precise control over network topology or data placement [23].

To evaluate and prove the proposed DDBF algorithm we use four fixed network topologies in our study including (i) Power-Law Random Graph (PLRG) [24], (ii) Normal Random Graph (Random) [25], (iii) Gnutella graph (Gnutella), (iv) Two-Dimensional Grid (Grid), all with fixed query distributions, and one random dynamic network with dynamic query distributions.

The rest of this paper is organized as follows. Section 2 presents a literature review for query flooding algorithms and connected dominating set algorithms. Section 3 concentrates on the problem description in the distributed unstructured networks. Section 4 explains the proposed distributed dynamic backbone-based flooding (DDBF) algorithm. Section 5 compares and discusses the results of the performance evaluation for DDBF with three flooding algorithms on five different topologies. Finally, section 6 presents the conclusion.

2 Literature Review

Resource location discovery is used on three important networks, namely Grid, Peer-to-Peer (P2P) networks, and cloud networks which we describe each briefly below. Grid middleware on the Grid networks provides important and basic services for resource discovery, data management, resource management, security and communication [26]. In this regard, grid systems interconnect storage systems, computer clusters, instruments, and existing share resources, such as data, storage, software applications, equipment, and CPU time. Over several decades P2P networks have become an effective way for distributed resources for communication and cooperation among nodes [27–29]. One of the most popular services proposed by P2P is file sharing (e.g., Gnutella). In P2P networks other applications for real-time data transfer cycle stealing, or existing collaboration (such as Skype, SETI@Home, and Groove). In these networks, participation is dynamic as users can enter, leave, and rejoin the system totally unpredictably. Cloud computing is a new computing model that makes use of pools of physical computing resources known as data centers (DCs) [30]. Cloud service providers are building out geo-distributed networks of microdata centers (MDCs) [4]. These MDCs, on-demand, can be organized into some nodes to provide different services, in which resource discovery is required [31].

The unstructured network resource discovery is known as blind search. It's because not any node is aware of the other's information and resources. Therefore, they must flood queries. It is known that the original Gnutella algorithm [13] uses a simply constrained flooding approach for search, which generates too much traffic on the network. To decrease this traffic on the network, some efforts are analyzed in this section.

In [11], authors purposed research to serve as a review of the most promising Grid systems that incorporate P2P resource location methods in order to perform a qualitative comparison of the existing approaches and to draw conclusions about their advantages and their weaknesses. In [23], the authors clearly show that the flooding-based query algorithm used in Gnutella does not scale enough and each query generates a large amount of traffic and large systems quickly become overwhelmed by the query induced load.

In [32], authors presented an Advanced Probabilistic Flooding (APF), in which a node decides to broadcast a message regarding the popularity of resources and the hop distance from the initial node. Its main drawback is its probabilistic nature. Qin Lv et al. in [7] proposed a query flooding algorithm based on multiple random walks that resolves queries almost as quickly as Gnutella's flooding method while reducing the network traffic by two orders of magnitude in many cases. In this work, peers randomly choose only a ratio of their adjacent peers to forward a query to. This reduces the average message production, while still creating a large number of redundant queries. In [33], the authors proposed a modified Breadth-First-Search (BFS) mechanism, which it was an extension of the Gnutella protocol. Modified-BFS allowed searching with keywords and was designed to minimize the number of messages that are needed to search the network. The modified-BFS mechanism still contacts a large number of peers. In [14] the authors proposed a random walk algorithm with a focus on the statistical properties of sampling performed. They quantified the effectiveness of random walks for searching and construction of unstructured peer-to-peer (P2P) networks. Also, they showed experimentally, that searching by random walks performed better than flooding. Proposed algorithm achieved a message reduction and load balancing in comparison to the flooding scheme. But, success rates greatly depend on network topology. In [34] the authors added structure to unstructured Gnutella network to improve search performance in a real-world deployment. This work shows how the importance of decreasing redundant queries on latency can be critical, which in practice even creating a new structure for the network can be affordable. In [35] the authors proposed a CSO (Capacity Sharing Overlay), as a P2P management system that enables the sharing of reusable resources and specifically network capacity. The main objective of the CSO was to provide a set of services to the interconnected networks to enable node sharing. In this research, the authors tried to address the problem of network workload fluctuations by changing the network size appropriately.

Boroumand et al. in [16] proposed optimized query forwarding for resource discovery in unstructured peer-to-peer grids. They used a genetic algorithm (GA) in which they considered both of the path length and network traffic. Boroumand approach reduced the hop numbers and prevented massive flooding of queries. This approach uses statistical tables that are obtained from the recorded history of previous queries. Then a genetic algorithm is applied to these statistical tables to find the optimum adjacent peers. This method was compared with a random walk and flooding approaches. However, it's known that GA is an evolutionary algorithm that in unstructured networks if adjacent peers' changes rapidly, the algorithm cannot act real-time enough for flood queries properly. Also collecting and updating statistical tables creates high traffic on the network rapidly, which is not considered in this approach. In addition to mentioned related works, there exist some other works that proposed almost the same approaches for flooding improvements in [15, 17, 36]. Hervé Baumanna and et al. in [37] proposed a flooding technique in dynamic graphs with arbitrary degree sequence. They established it by analyzing flooding in a sequence of graphs drawn independently at random according to a model of random graphs. However, they just focused on flooding time without considering network traffic and comparing their work with state-of-the-art works. In [4] the authors proposed a lightweight information sharing scheme in distributed cloud networks. This work developed a lightweight path flooding algorithm to improve existing flooding algorithms using hop count restriction. Lightweight path flooding algorithm is recently proposed, and its comparisons show the superiority of it against previous flooding algorithms. Although lightweight path flooding algorithm improved existing flooding algorithm, it still creates a lot of redundant messages over the network. In addition, there are some different approaches for reduction of cloud network latency for different specific purposes at [19].

Constructing Connected Dominating (CDS) algorithms some approaches that are based on the creation of a backbone for sending messages. CDS algorithms are mostly used in wireless sensor networks. Solving minimum CDS is NP-hard which can be classified as centralized algorithms, and distributed algorithms based on the network information they use. The central algorithms need global information of the network connectivity at one central base station and are aimed to reduce wireless nodes energy consumptions [38–40]. This feature makes them unsuitable for dynamic networks as there is no centralized control of the network. In [41], Wu and Li proposed a distributed CDS algorithm in which they first create a trivial CDS, then delete the redundant nodes based on two sets of pruning rules. In [42] a local algorithm is developed where any vertex on the network itself decides whether or not it is part of the dominating set depending just on the vertices that are in a constant number of hops away from it. In [43] authors presented a minimum CDS approach for wireless sensor networks using pseudo dominating set. In this paper authors through degree-based greedy approximation algorithm reduce the CDS size as much as possible. It is worth mentioning that the proposed algorithms for CDS problem are proportional to wireless sensor networks and its hypothesis. Proposed algorithms don't ensure connectivity in different situations; for instance, because of wireless sensor network nature, they don't have any strategy to add some new peers to the network after connected dominating set creation.

In [44] authors presented a protocol for 3D P2P overlay over mobile ad hoc networks (MANETs) in which each peer runs a distributed algorithm to exploits a 3D-overlay. This paper considered a scenario of a structured P2P overlay over a MANET. This algorithm's main drawback is trying to maintaining multi-paths to a destination peer, which cause higher redundancy and overload on the networks with lower dynamic nature. In [45] authors proposed a multihop Proximity aware Clustering Scheme for Mobile peer-to-peer systems (PCSM), which integrated three factors to select the cluster to join. Availability of the cluster head, number of physical hops, and the cluster size are the considered factors that integrated with a maintenance process to manage the mobility of peer. In this paper detection process for more available and stable peers in high dynamic networks is a big challenge. Then authors improved their algorithm by adding a global file popularity estimation naming clustering-based replication strategy (CRS) [46]. While the proposed algorithm has promising results, it uses global knowledge of all peers of the network which always is useful but it brings different challenges such as overload in highly dynamic networks. In [47] authors proposed a churn-resilient system in which they find alternative routing paths for balancing the query loads with high workloads. The idea in this research study is evolving the network into a cluster-like topology through resource grouping and a rewiring method to spontaneously organizing and clustering the peers which have the same resources. Then authors proposed a collaborative Q-learning algorithm to balance the query loads among the intragroup peers. In this paper, authors are used conventional Q-learning algorithm and they are not discussed the well-known memory problem of this method in which each node should dedicate a remarkable amount of memory to build the state-action Q-table in high connectivity networks.

In Table 1 we summarized the discussed research studies objectives, techniques, main approaches, and shortcomings.

3 Problem Description

In the distributed unstructured cloud network, if one of the nodes needs to discover resources on the network, it should send a new query to all of its adjacent nodes. Also, this query should be relayed by each node to all other ones on the network. Therefore, a major problem with flooding is that there are many duplicate messages introduced by flooding, particularly in high connectivity graphs. By duplicate query, we mean the multiple copies of a query that are sent to a node by its multiple adjacent nodes. Duplicate queries are pure overhead, which they incur extra network interrupt processing at the nodes receiving them but do not increase the chance of finding the object. Traditional flooding methods cause a lot of traffic because they have to send data to all the adjacent nodes. Also, duplication detection mechanisms are always needed in flooding algorithms in which detected duplicate messages are not forwarded. However, even with this duplicate suppression, the number of duplicate messages in flooding algorithms can be excessive, and the problem worsens as the TTL (Time-To-Live) increases. Consequently, these problems mean that flooding incurs considerable message processing overhead for each query, increase the load on each node as the network expands, and increase the query rate [48].

To prevent the mentioned flooding algorithms drawbacks, the flooding-style search method is designed, which discards redundant messages and does not execute the flooding again. However, since redundant messages will occur regardless of flooding style, it causes high traffic on the network. In order to alleviate that problem, some algorithms such as the random walks algorithm, restricted flooding algorithms, the light-weight path flooding scheme, etc. have been proposed in [4, 7, 14–17, 33, 34, 36]. In the latest research which named lightweight path flooding scheme, one efficient approach for decreasing the repeated query overhead is proposed in [4]. In the lightweight flooding scheme, a node does a flooding on the query that includes the list of the adjacent peers' addresses. Upon receiving the query, the nodes check if its adjacent nodes exist in the list of addresses included in the query, and it will not send the query to the nodes that exist in the address list. Lightweight path flooding scheme was compared with earlier state-of-the-art works, and it demonstrated higher performance by decreasing the number of repeated queries significantly. However, this solution reduces query duplication at one hop dramatically, thus by increasing the number of hops its performance decreases. Therefore, despite it decrease the redundant messages dramatically, a large number of duplicate messages is still introduced.

In order to solve this problem, we propose Distributed Dynamic backbone-based Flooding (DDBF) algorithm. In DDBF algorithm we present a distributed algorithm to prevent sending queries by all unnecessary connected nodes. In fact, we create a flexible backbone of active nodes, which just these

Table 1 summarized specification of existing flooding approaches

	Scheme	Objective	Techniques	Approach / tools	Shortcomings
1	APF [32]	Adjusting the forwarding probability at the time a node receives the query message to reduce the duplicate message overhead while maintaining a high probability of query success.	Node decides to broadcast a message regarding the popularity of resources and the hop distance from the initial node.	Probabilistic flooding strategy	probabilistic nature in which success rates greatly depend on network topology.
2	k-walker random walk [7]	Proposing a query algorithm based on the multiple random walks to resolve queries as fast as Gnutella's flooding method along with reducing the network traffic.	Query flooding algorithm based on multiple random walks.	A distributed technique	Peers randomly choose only a ratio of their adjacent peers to forward a query to and still creating a large number of redundant queries.
3	Modified-BFS [33]	Information retrieval in pure peer-to-peer networks aiming to minimize the number of messages that are needed to search the network and improve the scalability of the search procedure.	Extension of the Gnutella protocol and intelligent search mechanism.	Local search mechanism	Still contacts a large number of peers.
4	Random walk [14]	Proposing an improved searching approach by focusing on power and efficiency of sampling in P2P networks.	Focused on the statistical properties of sampling performed to show searching by random walks performed better than flooding approach.	A distributed technique	Success rates greatly depend on network topology.
5	Hybrid approach with adding structure to Gnutella [34]	Augment the unstructured Gnutella network with a structured overlay network and present query, maintenance, and ultrapeer election algorithms.	Adding structure to unstructured Gnutella network to improve search performance.	A distributed technique based on Gnutella	Lack of comprehensive comparisons and evaluation of dynamic networks with different topologies.
	CSOA [35]	Proposing a set of services to the interconnected networks to enable node sharing and specifically, network capacity.	Address the problem of network workload fluctuations by changing the network size appropriately.	A distributed service providing technique	CSOA supports standard blind P2P search techniques such as flooding and k-walkers and it has their shortcomings.
6	PCSM [45]	Proximity aware Clustering Scheme for Mobile peer-to-peer systems (PCSM).	Checking availability of the cluster head, number of physical hops and the cluster size are the considered factors that integrated with a maintenance process to manage the mobility of peer.	A clustering algorithm	The algorithm is relied on detecting more available and stable peers, while in highly dynamic networks is a big challenge. Also, it is missed to be evaluated on different network topologies.
7	Optimized query forwarding [16]	Optimizing query forwarding for resource discovery in unstructured peer-to-peer grids through evolutionary algorithms.	Considering both of the path length and network traffic which reduced the hop numbers and prevented massive flooding of queries.	Genetic algorithm	In dynamic unstructured networks if adjacent peers' changes rapidly, the algorithm cannot act real-time enough for flood queries properly. Also, collecting and updating statistical tables creates high traffic on the network rapidly.
8	Flooding in dynamic graphs [37]	Proposing a flooding technique in dynamic graphs with arbitrary degree sequence.	Analyzing flooding in a sequence of graphs drawn independently at random according to a model of random graphs.	Prove upper bounds on the flooding time	Approach just focused on flooding time and proving without considering network traffic and comparing their work with state-of-the-art works.
9	[4] Lightweight path flooding algorithm	Developing a lightweight path flooding algorithm to improve existing flooding algorithms using hop count restriction.	Path flooding algorithm to improve existing flooding algorithm using hop count restriction.	A distributed technique	Still creates a high number of redundant messages over the network.
10	Distributed CDS [41]	Proposing a simple and distributed algorithm for calculating connected dominating set in ad-hoc wireless networks.	Create a trivial CDS, then delete the redundant nodes based on two sets of pruning rules.	A distributed technique	It is based on the wireless sensor network characteristics and depending on the nodes' transmission radius.
11	CPDS2HI [43]	Presenting a minimum CDS approach for wireless sensor networks using pseudo dominating set, aiming to reduce the CDS size as much as possible.	At first, constructing a CDS and then reduce its size further by excluding some of the CDS nodes cleverly without any loss in coverage or connectivity.	Greedy approximation algorithm	Proportional to wireless sensor networks and its hypothesis. It doesn't ensure connectivity in different situations.
12	(CRS) [46]	Proximity aware Clustering Scheme for Mobile peer-to-peer systems (PCSM) and improving file availability aiming to enhances the search efficiency.	Adding a global file popularity estimation.	A distributed technique	Using global knowledge of all peers of the network which always is useful but it brings different challenges such as overload in highly dynamic networks.

Table 1 (continued)

	Scheme	Objective	Techniques	Approach / tools	Shortcomings
13	3DO [44]	A protocol for 3D P2P overlay over MANETs.	Each peer runs a distributed algorithm that exploits a 3D-overlay to calculate a consecutive logical identifier to a peer.	A distributed technique	Maintaining multi-paths to a destination peer, which cause higher redundancy and overload on the networks with lower dynamic nature.
14	CCLBR [47]	Proposing a churn-resilient system aiming to improve the search efficiency with finding alternative routing paths for balancing the query loads with high workloads.	Evolving the network into a cluster-like topology through resource grouping and a rewiring method with same resources and then applying a collaborative Q-learning method to balance the query loads among the intragroup peers.	Q-learning algorithm	Well-known memory problem of the Q-learning algorithm is not discussed. Thus, each node should dedicate a remarkable amount of memory to build the state-action Q-table in high connectivity networks.

active nodes are ordered as query senders. Hence the number of the query senders and the number of unnecessarily repeated queries decrease significantly on the network. As mentioned earlier, we assume that the proposed algorithm runs over the unstructured network. Therefore, we provide a reliably distributed algorithm for activating only some of the nodes as a backbone to decrease the number of the duplicated queries.

4 Proposed Distributed Dynamic backbone-based Flooding Algorithm

In this section, we describe the proposed Distributed Dynamic backbone-based Flooding (DDBF) algorithm. In the DDBF algorithm, the main idea is to activate some of the nodes as query flooders to send queries on the whole network. Therefore, we select some of the nodes as active nodes, which cover all connected nodes on the network. Therefore, when we talk about an "Active" node, we assume it as a query flooder. On the other hand, an "Inactive" node is a node that does not flood any queries. In this regard, we provide some rules for activating or deactivating nodes. We should note that when we use N_i notation it means the node that is a target in descriptions and it runs the rules. In the DDBF algorithm, all activated nodes never change their active mode to inactive mode unless they leave the network. If a node is active and leaves the network, it becomes inactive and only proposed distributed rules can make it active again.

The proposed DDBF algorithm includes two phases. The first phase is named new backbone creation, and the second phase is named query flooding. In the first phase, a flexible backbone for flooding queries is created for all existing nodes on the network. This happens by running four proposed distributed rules. Then, in the second phase when one node needs to send a query, it uses the created backbone to flood it. Figure 1 shows total state machine of DDBF algorithm.

As shown in Fig. 1, there are two main phases in the DDBF algorithm. The first phase (backbone creation phase) includes two steps which are called activation rules and assurance of connectivity. At backbone creation phase; first of all, all existing nodes on the network follow four rules to be in the active or inactive mode, then assurance of connectivity step starts to ensure the connectivity. After the backbone creation phase, the query's second phase (flooding phase) starts to work. Consequently, all nodes on the network can forward a new query but just activated nodes are able to flood queries to all of its adjacent nodes. In the query flooding phase in order to preserve flexibility over the network, there are two special situations. First, when a new node is added to the network, it follows proposed rules to be in one of the active or inactive modes (then it enters the query flooding phase). Second, if one of the activated nodes fails or is disconnected for any reason, the assurance of connectivity step will run on the network.

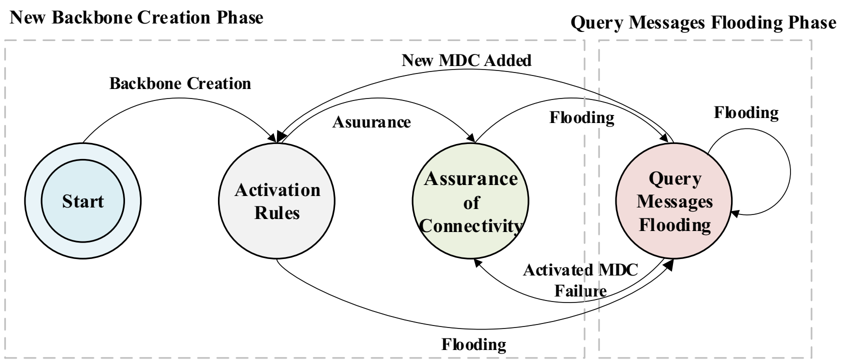

Fig. 1 The state machine of DDBF algorithm

The following section, first, describes algorithm definitions, then explains both of the phases at two different subsections in detail.

4.1 DDBF algorithm definitions

To describe the rules, we assume some notations and functions with an example in Table 2. So, let's assume a simple network in Fig. 2 as nodes' links in a network. Note that all of the examples in Table 2 is based on this simple network.

In Fig. 2, the nodes $\{2, 4, 5\}$ are considered as activated nodes, and the nodes $\{1, 3, 6\}$ are considered as inactive ones.

4.2 Backbone creation phase

As mentioned above, the backbone creation phase has two steps as follows: (i) Activation rules step, (ii) Assurance of connectivity step. So, in the rest of this section, we describe both steps in detail.

4.2.1 Activation rules step

In the activation rules step, nodes performing proposed rules to be in one of the active or inactive modes. The activation rules are described below (Fig. 3 illustrates a flowchart of rules). Note that these rules can run concurrently on different nodes with any different performing starting time at different nodes.

First of all, each node checks if its mode is active or not. If not, the following 4 rules should be performed, if yes it remains active until it leaves the network.

Rule 1: The node N_i checks its number of links ($NC\{Adj(N_i)\}$) If its number of links is lower than 2, ($NC\{Adj(N_i)\} < 2$) so it means the node has just one adjacent node. Therefore it stays inactive and sends a message to its single adjacent node to activate it (sends $AAM(N_j)$); otherwise, it goes to the second rule.

Rule 2: The node N_i checks its number of activated adjacent nodes ($NA\{Adj(N_i)\}$) If a number of activated

Table 2 Notations used in the rest of the paper; brackets are examples according to the simple network in Fig. 2

Function		Description	Example
1	$\mathbf{Adj(N_i)}$	All adjacent nodes of N_i	$Adj(3) = \{1, 5, 4\}$
2	$\mathbf{N\,C\{Adj(N_i)\}}$	Number of links of N_i	$NC\{Adj(3)\} = NC\{1, 5, 4\} = 3$
3	$\mathbf{N\,A\{Adj(N_i)\}}$	Number of active adjacent nodes of N_i	$NA\{Adj(3)\} = NA\{1, 5, 4\} = 2$
4	$\mathbf{N\,D\{Adj(N_i)\}}$	Number of inactive adjacent nodes of N_i	$ND\{Adj(3)\} = ND\{1, 5, 4\} = 1$
5	$\mathbf{A(N_i)}$	Active adjacent nodes of N_i $A(N_i) = \{a_{1i}(N_i), a_{2i}(N_i), \ldots, a_{Li}(N_i)\}$	$A(3) = \{4, 5\}$
6	$\mathbf{A_1(N_i)}$	One of the active adjacent nodes of N_i $A_1(N_i) = \{a_{1i}(N_i)\}$	$A_1(3) = \{4\}$ *or* $\{5\}$
7	$\mathbf{D(N_i)}$	Inactive adjacent nodes of N_i $D(N_i) = \{d_{1i}(N_i), d_{2i}(N_i), \ldots, d_{Ki}(N_i)\}$	$D(3) = \{1\}$
8	$\mathbf{A(A_i(N_i))}$	Active adjacent nodes of one of N_i active adjacent $A(A_1(N_i)) = A(\{a_{1i}(N_i)\})$	$A(A_1(3)) = A(\{5\}) = \{4\}$ *or* $A(\{4\}) = \{5\}$
9	$\mathbf{A(D(N_i))}$	Active adjacent nodes of inactive adjacent N_i $A(D(N_i)) = A(\{d_{1i}(N_i), d_{2i}(N_i), \ldots, d_{Ki}(N_i)\})$	$A(D(3)) = A(\{1\}) = \{2\}$
10	**AAM(Nj)**	Sending adjacent activation message to node N_j	–

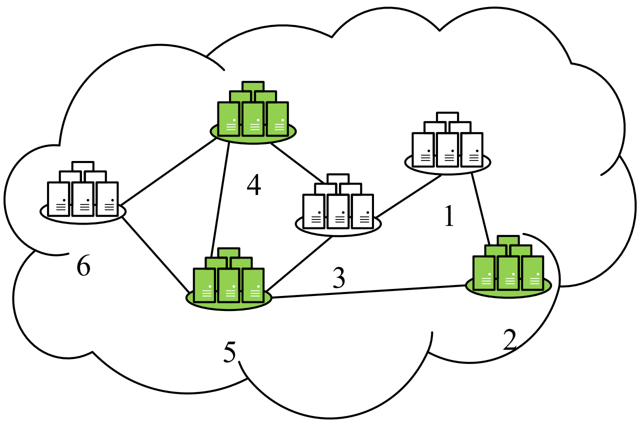

Fig. 2 An example of nodes links and modes

adjacent is lower than one ($NA\{Adj(N_i)\} < 1$) it means there are not any active adjacent nodes. Therefore, the N_i first collects its adjacent nodes' number of links, then it compares if it has a greater number of links in comparison to its adjacent nodes. If yes it activates itself. Otherwise, it sends an activation message ($AAM(N_j)$) to an adjacent node with the maximum number of links,($N_j = Max\ \{NC\{N_i\}, NC\{d_{1i}(N_i)\}, NC\{d_{2i}(N_i)\}, \ldots, NC\{d_{Ki}(N_i)\}\}$)

Rule 3: The node N_i checks if its number of activated adjacent nodes is equal to two, ($NA\{Adj(N_i)\} = = 2$) If so, it requires both activated adjacent nodes and their activated adjacent nodes identifications (for instance IP address), then it finds the subscription nodes between these two nodes. It's for finding a link between its two activated nodes; Now, if a link is found, the node N_i changes its own status to active. Otherwise, it goes to the final rule.

Rule 4: The node N_i checks if the number of its inactive adjacent count is equal to one ($ND\{Adj(N_i)\} = = 1$), which means it has just one inactive adjacent. If so, node N_i checks if there are subscription nodes between its active adjacent nodes and its inactive adjacent' active adjacent nodes or not, ($(A(D(N_i)) \cap A(N_i)) = \varnothing$) If there is no at least one subscription node, first the node N_i becomes active, then it sends an Adjacent Activation Message ($AAM(N_j)$) to its single inactive adjacent node to activate it too.

In the flowchart (Fig. 3), $A(N_i)$ means activating itself, which runs the rules, and $D(N_i)$ means activating all of the inactive adjacents of N_i.

Communicated messages in the activation rules step To perform the proposed rules in the DDBF algorithm, each node requires some information from its adjacent nodes. Thus, prior to performing rules, some information messages (IM) is requested from adjacent nodes by each node. IM contains each node's critical situations such as nodes number of links (connections count) and so on. IM only communicates between nodes once when a new node enters or leaves the network. This gathers some information maximum in two

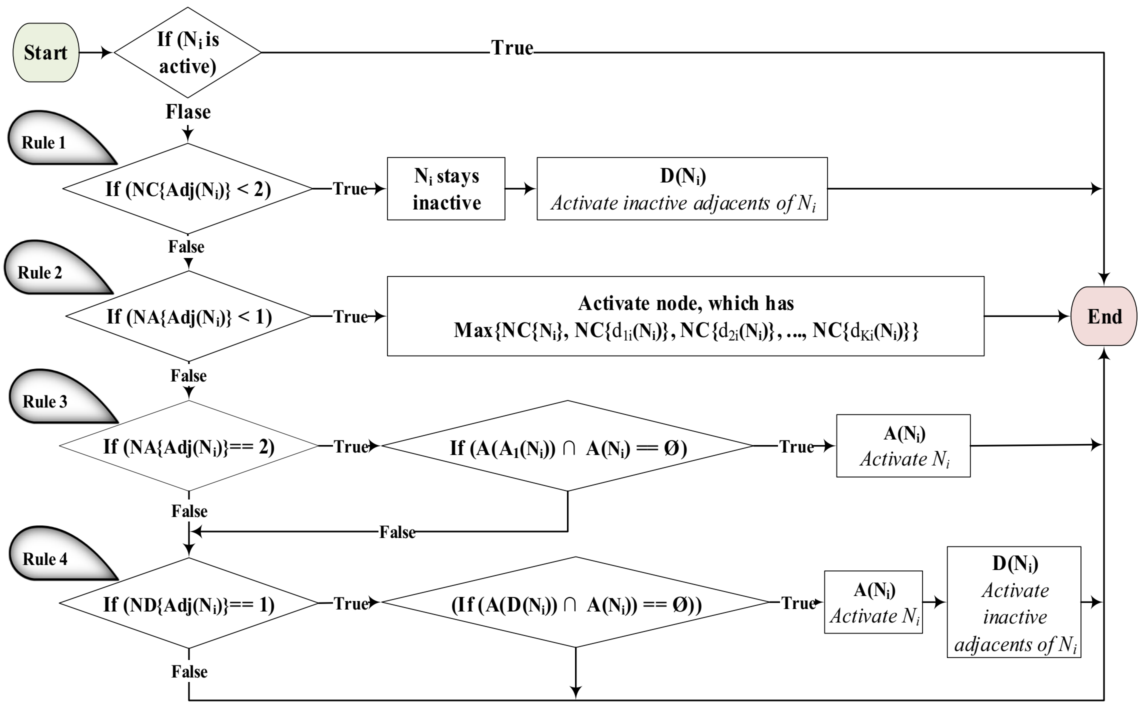

Fig. 3 Flowchart of activation rules steps in DDBF algorithm, which each node should run in the backbone creation phase

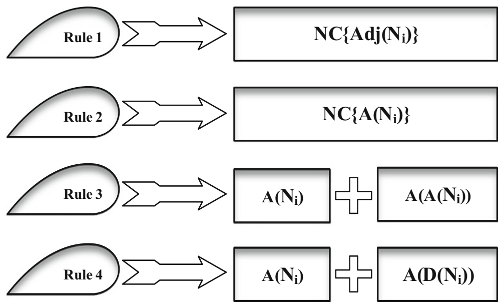

Fig. 4 The required information between adjacent nodes in the activation rules step

hops (adjacent nodes of the node N_i, and adjacent to adjacent nodes of the node N_i). We should note that right after changing node N_i status from inactive to active, an information message is sent to all adjacent nodes immediately. This is to update information of the adjacent nodes of node N_i about its status. For each proposed rule in Fig. 3 the required information for nodes is shown in Fig. 4.

Critical situation in the activation rules step In the DDBF algorithm except for one condition, simultaneous execution of proposed rules on the different nodes is safe. This critical situation is when node N_i replays its own mode as inactive to its few adjacent nodes, and if at the same time one of its adjacent nodes sends a message to change its mode to active mode. In this condition, the node N_i is active, whereas its adjacent nodes have opposite information. Therefore, it may cause bad decisions in the adjacent nodes (at worse activating additional unnecessary node), (Fig. 5).

In Fig. 5, one example of the critical situation is shown. In this example node #1 and node #2 asked node #3 for its mode and node #3 replied inactive mode to both of them (red arrows). After that, if node #1, decides to activate node #3 due to the DDBF algorithm rules, (green arrow). Then node #5 will have the wrong status of node #3. Therefore it causes bad decisions in node #5. Therefore, to avoid such a bad decision, we considered one additional rule. This rule checks and considers each of the nodes when the mode of the node is changed by its adjacent node. The pseudocode of this rule is as follow:

In this pseudocode, the rule shows that by using this supplementary rule, each node is able to update its previously reported mode to its adjacent nodes. Therefore, according to this supplementary rule, in the previous example in Fig. 5, node #3 sends an adjacent update message$AUM(N_5)$ to node #5 immediately after changing its state by node #1, (blue arrow). Thus, the mentioned problem is fixed.

An example of activation rules step In order to understand the rules better, we show one scenario for performing the rules on a simple network as an example as follows: Let's assume the nodes are linked as in Fig. 6 (a), and they are in the first step of the backbone creation phase.

In Fig. 6 (a), there are 16 nodes with some links between them, which want to create their backbone. Due to the Rule 1, the Nodes #9, #11 and #16 have less than two adjacent nodes. Therefore they will decide to be inactive, and they activate their adjacent nodes #8, #10 and #14 respectively (Fig. 6 (b)). Due to the Rule 2, each of nodes #2, #3, #4 and #7 reaches to the Rule 2 first; sends $AAM(N_3)$ to the node #3 to activate it. This is because of adjacent node numbers, i.e., node #3 has the maximum number of adjacent nodes. Also, node #1 activates node #4 for the same reason in Rule 2 (Fig. 6 (c)). Due to the Rule 3, one of the nodes #5 or #6 changes its mode to active mode. The reason is that both of them has only two different active adjacents nodes which are not connected to each other. Here, we considered node #5 as activated first, which does not make a difference if node 6 is activated; (a critical situation could occur here, see section 4.2.1.2). Due to the Rule 4, nodes #12 and #13 change their mode to active mode too. The reason is that node #12 has just one inactive node (node #13), whose activated adjacent nodes are not linked to node #12, (Fig. 6, d). Finally, none of the nodes can enter to any of the proposed rules, and the backbone creation phase rules step finishes. It's clear in (Fig. 6 (d)), one optimum backbone for query flooding with a minimum number of activated nodes is created.

It is worth noting that distributed algorithms cannot provide a global optimum solution in a large state space. Thus, in the DDBF algorithm, the created backbone may not activate nodes in a global optimum necessarily. But the results are near the optimum. In the

```
If ($AAM(N_i)$ received from one adjacent node)
  If ($N_i$ is "inactive" and its mode sent to another adjacent before)
    {
      Send an adjacent update message $AUM$ to all adjacent nodes
      with wrong information to update their information
    }
```

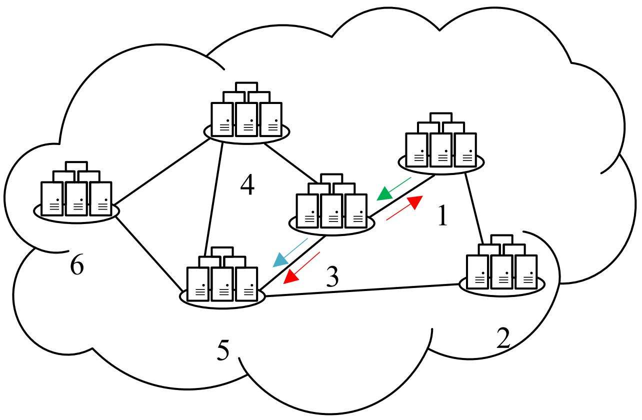

Fig. 5 An example of the critical situation; red arrows are inactive mode announcement of node 3, the green arrow is for activating node 3 message, and the blue arrow is an update message

discussed example, there can be some different scenarios of activating too. However, in Fig. 7 we show the result of some different scenarios of concurrent performing the rules, which provide suitable backbones.

As shown in Fig. 7 there are four different instance results of a different order of performed rules. These results show different outputs for rules distributed concurrently on the network.

4.2.2 Assurance of connectivity step

This step is for ensuring the connectivity in the worse situation between all activated nodes in all topologies with dynamic nature. This step performs in two situations. The first condition is after performing proposed rules and just when a new backbone is created (for the first time after a small time gap ω to allow performing rules by all nodes on the network). The second condition is when an activated node leaves the network. For assurance of connectivity, we define a Connectivity Check Message (CCHM). When one of the activated nodes is required floods a specific CCHM with its ID on the network. This node at first condition will be a random node on the network, and in the second condition will be one of the adjacent nodes of the activated and leave (we described it in Sec. 4). Consequently, CCHM floods by created backbone to whole nodes on the network. Each node that receives CCHM follows assurance of connectivity instructions illustrated in Fig. 8.

In Fig. 8 the rules run when N_i receives the CCHM. Thus, if N_i is inactive it runs the instructions as follows:

- N_i waits for $\mathcal{E}$ and asks their adjacent nodes for CCHM. If all of its adjacent nodes receive CCHM, then, the insuring of the connectivity finishes. But if even one of their adjacent nodes has not received CCHM, the related node will

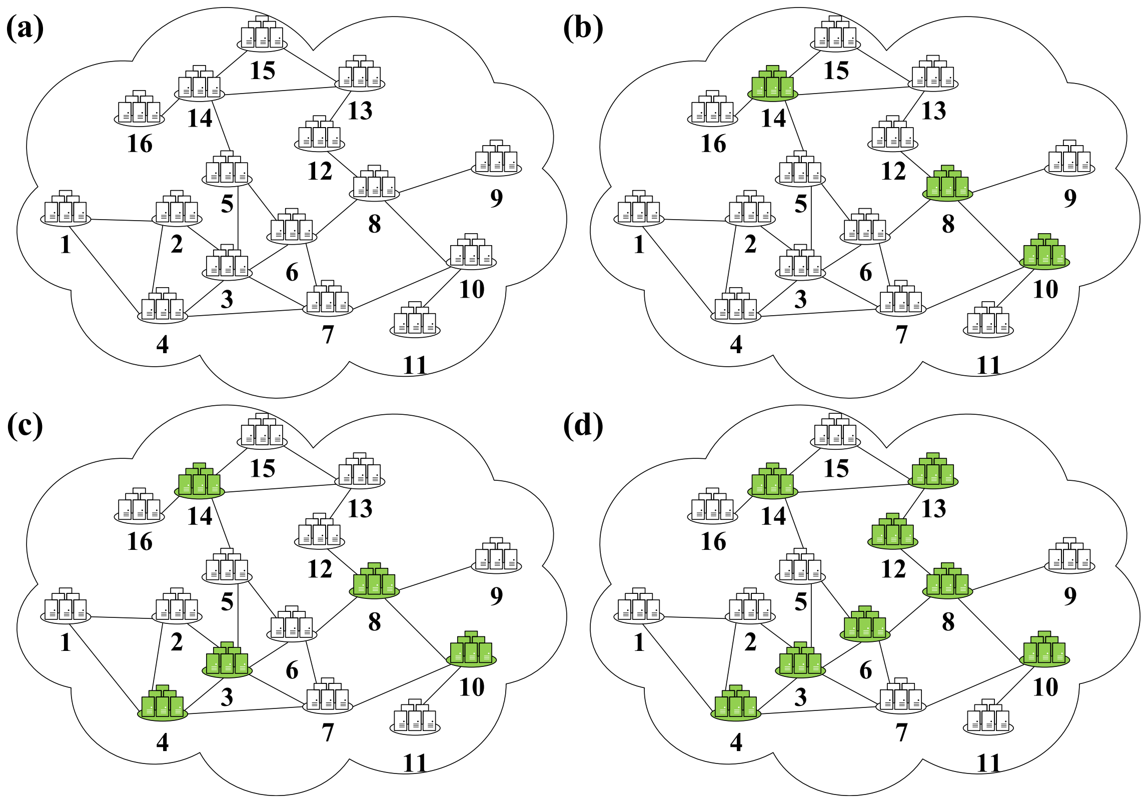

Fig. 6 The example of connected Nodes; subfigures (**a**), (**b**), (**c**) and (**d**) show the activation sequence respectively

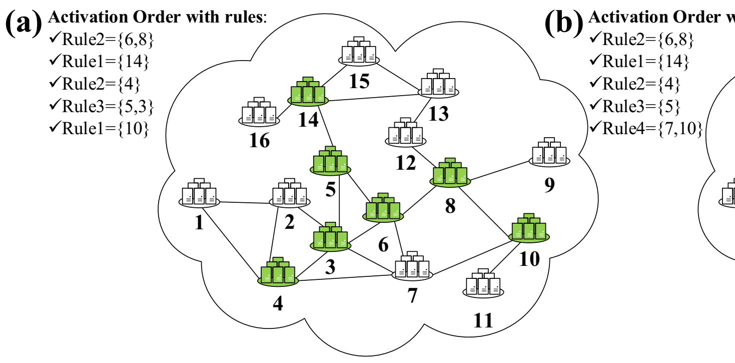
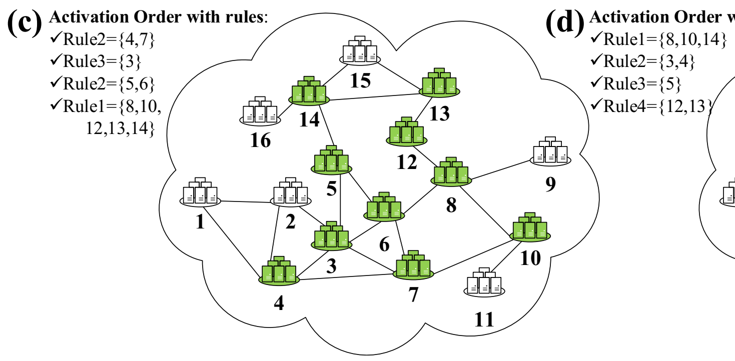

Fig. 7 Different scenarios of performing rules; figures (**a**), (**b**), (**c**), and (**d**) show the activated MDs with a different sequence of performed rules. For instance, in figure (a), Rule 2, activated nodes #6, 8; rule 1 activated node #14 and so on

activate itself and flood a new CCHM just to the nodes which have not received the CCHM. This CCHM sending routine continues until every CCHM floods on the network. After that, the connectivity between activated nodes is insured. (The waiting time ε contains a bigger worse time than a message that takes to flood on the whole network like time to live (TTL)).

Based on this step, proposed DDBF algorithm becomes a flexible backbone, which is robust in dynamic conditions for any topology.

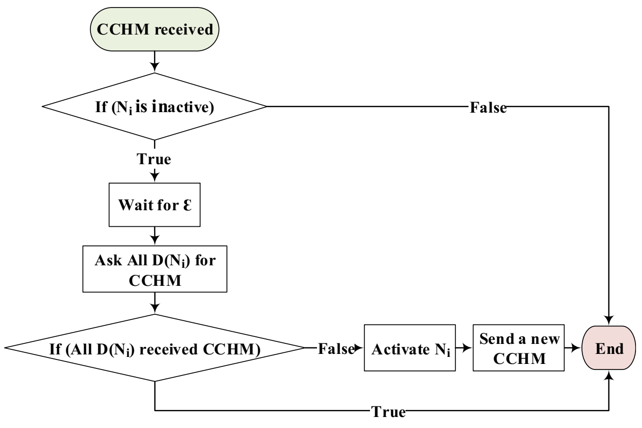

Fig. 8 Assurance of connectivity step flowchart

4.3 Query forwarding phase

In this phase, the created backbone is ready to use. Each node which has a query to flood on the network can use the created backbone. The inactive nodes which need to send a query only send one query to one of its active adjacent nodes. So, each active node sends the query to all of its adjacent nodes. By repeating this routine, the query is sending to all existing nodes on the network. As shown and mentioned in Fig. 1, if some new nodes enter the network, newly entered nodes follow proposed rules to be in one of the active or inactive modes. Second, if some of the activated nodes fail or leave the network, the assurance of connectivity step will run on the network.

5 Performance evaluation

In this section, we compare DDBF algorithm with the Traditional Flooding algorithm (TF), Random Walk algorithm (RW) [14], and Lightweight path flooding algorithm (Lightweight) [4]. As mentioned above, TF causes a lot of traffic because it sends data to all the adjacent nodes. Well-known Random walk technique, which forwards a query message to a randomly chosen neighbor at each step until the object is found, is mainly dependent on the network topology. Also, lightweight path flooding algorithm is the most recently presented algorithm, which still generates a large number of redundant queries on the network.

5.1 Simulation setup

To evaluate and prove the proposed DDBF algorithm, we use four fixed network topologies and one dynamic network with dynamic query distributions in our studies. Key statistics of these network topologies are shown in Table 3.

In Table 3, the number of nodes and their degree information for five graphs has been shown. Details of these topologies are as follows:

(i) Power-Law Random Graph (PLRG): Many real-life P2P networks have topologies that are power-law random graphs [49]. In this graph, the node degrees follow a power-law distribution in which when ranked from the most connected to the least connected, the ith most connected node has a/i^{β} adjacent nodes, where β is a constant. Once the node degrees are chosen, the nodes are connected randomly [24].

(ii) Normal Random Graph (Random): A random graph which is generated by a modified version of the GT-ITM topology generator [50].

(iii) Gnutella graph (Gnutella): The Gnutella network topology node degrees roughly follow a two-segment power-law distribution.

(iv) Two-Dimensional Grid (Grid): A two-dimension grid which is a simple graph chosen for comparison purposes. We should note that the chosen fixed networks are based on the research in [23].

(v) Dynamic graph (Dynamic): The initial structure of this graph is based on PLRG, and it changes with entering and leaving random nodes in the graph dynamically. The reason for selecting PLRG as the initial graph is that many real-life P2P networks have topologies that are like power-law random graphs [49]. Based on this dynamic graph, we can evaluate our proposed algorithm in the almost real dynamic network. Hence in this graph, we tried to consider the true dynamics of a node coming and going on the network. The maximum and minimum number of nodes and degrees in this network are shown in Table 3. Node's entering and leaving rate from the network is considered as a static value to show results in the simulations. Note that when a random node is chosen to leave the network, the whole network connectivity is preserved.

In the flooding process, distribution of the flooder's nodes can be important by assuming that n flooder nodes require discovery resources. Let q_i be the relative popularity of the ith node. The values are normalized:

$$\sum_{i=1}^{n} q_i = 1 \tag{1}$$

We investigated the following distributions for flooder nodes:

(i) Uniform**:** All resources are equally popular.

$$Uniform: q_i = 1/n \tag{2}$$

(ii) Zipf-like**:** the popularity of flooder nodes follows a Zipf-like distribution. Studies have shown that Napster, Gnutella, storage cluster and Web queries follow Zipf-like distributions [51, 52].

$$Zipf{-}like: q_i \propto 1/i^{\alpha} \tag{3}$$

For each set of simulations, we first select the topology and the query flooder distributions. Then we assume that when a node floods a query, all other existing nodes on the network must receive the query at least once. In the networks with static topology simulations, a

Table 3 Key statistics of used network topologies in the simulations

	Topology	Number of nodes	Number of total links	Nodes average degree	STD	Maximum degree	Median degree
1	**PLRG**	3830	8870	4.81	18.4	270	1
2	**Random**	3640	7568	4.75	1.50	18	4
3	**Gnutella**	3823	8907	4.92	9.03	127	3
4	**Grid**	500–3000	995–5890	3.96	0.2	4	4
5	**Dynamic**	500–3000	1256–9340	3.21–4.82	1.1–2.1	9–22	2–3

number of flooders are considered 1000, and in dynamic one 200 for both distribution types. Also, the value of α parameter is considered as 0.8. Flooder nodes are chosen based on two distribution types described above. Thus, statistics at simulations show averages. Note that, we run the simulations for each query independent of other queries because individual queries do not have any effect on each other.

5.2 Metrics

Performance issues in real P2P networks can be based on a variety of items. Therefore, we focus solely on efficiency aspects and use the following metrics in our described topologies in P2P networks. These metrics, though simple, reflect the fundamental properties of the proposed algorithm in comparison to others.

1) A number of the nodes, which are involved in queries flood in the whole network.
2) A number of the total generated queries in the whole network.
3) Worse delay (hops) for receiving query by all existing nodes in the whole network.
4) The average delay in receiving queries from all existing nodes in the whole network.

5.3 Simulation results for static network topologies

In Fig. 9, we compared DDBF algorithm with TF, RW, and Lightweight algorithms with six different number of nodes with respect to grid topology (Fig. 9 (a)), and PLRG, Random, and Gnutella topologies (Fig. 9 (b)). In this figure, the number of involved nodes for flooding on the network has been compared in which results are flooding averages per both uniform and Zipf-like query flooder distributions. By looking at results, it can be seen that the traditional flooding scheme used all of the peers for flooding query on the network. Traditional flooding, random walk, and Lightweight path flooding algorithm have almost the same results. However, lightweight is better. In contrast, DDBF algorithm, because of using a dynamic backbone for query flooding, decreased the number of engaged nodes in query flooding process significantly. In terms of engaged nodes for query flooding, the average percentage of decrease in all experiments in comparison with lightweight is 30.44%.

Figure 10 shows comparisons in terms of a number of the flooded queries on the network with a different number of nodes on the grid topology (Fig. 10 (a)), and with different topologies (Fig. 10 (b)). Results were obtained from flooding query with both uniform and Zipf-like query flooder distributions. In this figure, it is clear that DDBF algorithm has dramatic results in decreasing the number of the redundant queries in all topologies.

DDBF algorithm decreased the number of the redundant queries in a high number of nodes on the network about 49% compared to flooding scheme and 27% compared to Lightweight path flooding algorithm in all topologies. Based on results of Fig. 10 (a), we should note that by an increase in the number of the nodes, the performance of DDBF algorithm increases gradually compared to all of the other algorithms. It means DDBF algorithm outperforms that of others for more nodes on the network.

Figure 11 (a) shows a comparison in terms of worse queries delay (hops) and a number of the nodes on the grid topology. Also, Fig. 11 (b), shows a comparison in terms of worse queries delay (hops) and a number of the nodes on the three different topologies. Like previous experiments, results were obtained from flooding query with both uniform and Zipf-like query flooder

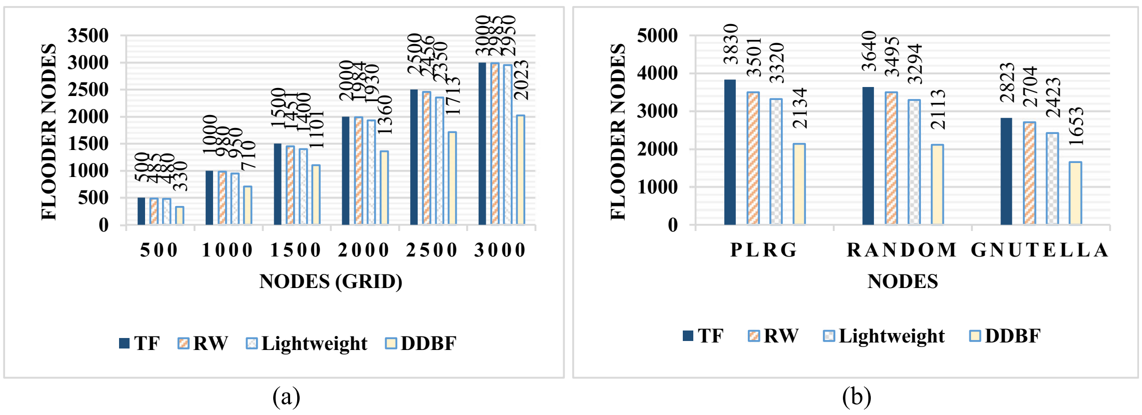

Fig. 9 Comparison in terms of the number of engaged nodes for query flooding (**a**) grid topology, (**b**) PLRG, Random, and Lightweight topologies

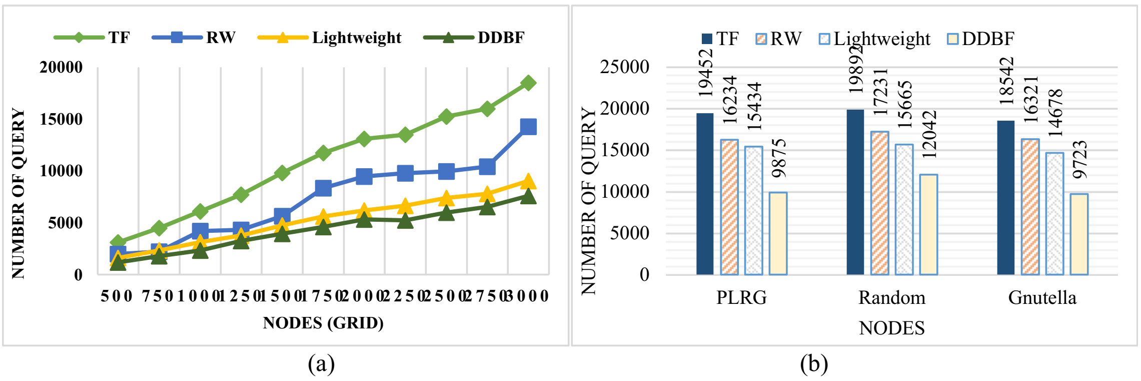

Fig. 10 Comparison the number of flooded queries on the network (**a**) grid topology, (**b**) PLRG, Random, and Lightweight topologies

distributions. TF algorithm engages all nodes on the network for flooding. Thus its worse delay time is the best. In contrast to RW and Lightweight path flooding algorithms, DDBF uses backbone to flood the queries, which path loops on the path become minimum. Thus, its worth query delay performance is better than RW, and Lightweight path flooding algorithms. By looking at these results, DDBF cannot provide the same worth delay as TF. However, it is best in comparing to RW and lightweight algorithms. Also, based on the dramatic improvements in the other important metrics DDBF provides a close worth delay to TF.

In Fig. 12, we compared DDBF algorithm with TF, RW, and Lightweight path flooding algorithms in terms of query flooding average delay (hops) on the network. Figure 12 (a) shows the results of query flooding average delay per deferent number of the peers on the Grid topology. Also, Fig. 12 (b) shows the results of query flooding average delay per deferent number of the peers on the three other topologies (see Table 3). As it is obvious that the results show DDBF algorithm results are between Flooding scheme and Lightweight path flooding algorithm like the previous worse queries delay comparison. It means DDBF algorithm improves delay against other state-of-the-art algorithms at all while it maintains a reasonable delay in comparison with traditional flooding algorithm (TF), which engages all existed nodes on the network for query flooding.

5.4 Simulation results for dynamic network topology

To evaluate dynamicity on the proposed algorithm, we present the results of query flooding on a dynamic topology (see its detail in Sec 5.1). For dynamic network topology, we propose the results of two important metrics as a number of engaged nodes on the flooding and number of queries on the network. Figure 13 (a) compares the DDBF algorithm with TF, RW, and Lightweight path flooding algorithms in terms of a number of engaged nodes. It is worth noting that in Fig. 13 first, the number of nodes on the network increasing (new nodes adding to the network randomly), then at the same network number of nodes are decreasing (nodes leaving the network randomly). As mentioned in the simulation setups, dynamic network topology per each change is 200 query flooding (100 per each flooder distribution types).

In the results obtained for dynamic network topology, average improvement percentage in terms of the number

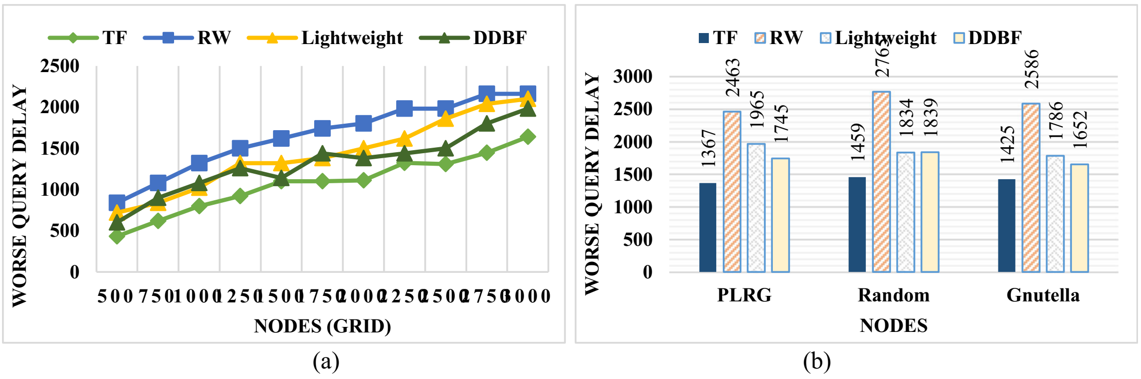

Fig. 11 Comparison the number of the worse queries delay (hops) on the network (**a**) grid topology, (**b**) PLRG, Random, and Lightweight topologies

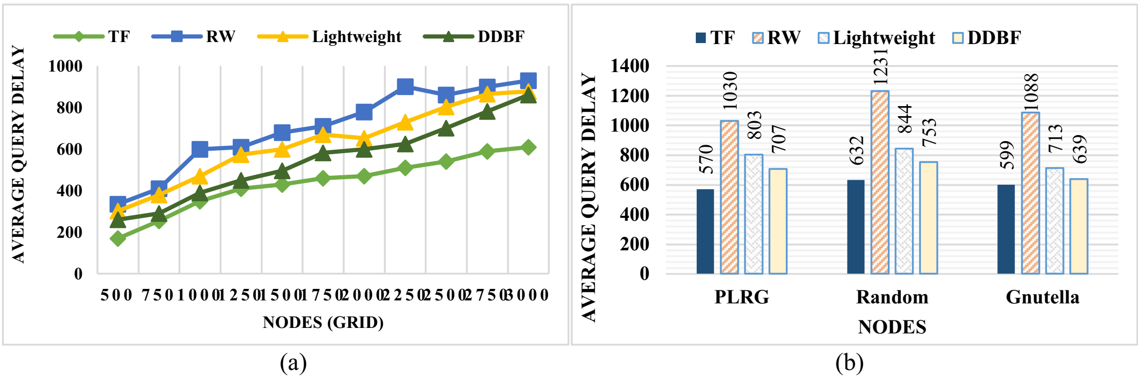

Fig. 12 Comparison, the number of the averages, queries delay (hops) on the network (a) grid topology, (b) PLRG, Random, and Lightweight topologies

of engaged nodes for flooding on the network is 42% for DDBF in comparison to the TF, and in comparison to the lightweight algorithm is 24%. Also, Average improvement percentage in terms of the number of flooded queries on the network is 67%, for DDBF in comparison to the TF and it is 34% in comparison to the lightweight algorithm. Consequently, in addition to static topologies, the simulation results in the dynamic network topology prove DDPF superiority.

6 Conclusion

Unstructured networks are becoming a promising platform and have developed rapidly in recent years. Sending queries to resource discovery in the unstructured networks such as Grid systems, Peer-to-Peer (P2P) network, and clouds networks is one of the necessities in networks. Therefore, networks' performance can be influenced by using an efficient algorithm for flooding queries. Therefore, in this paper, we developed a distributed dynamic backbone-based flooding (DDBF) algorithm to improve existing flooding algorithms in terms of redundancy and delay. We implemented and evaluated DDBF in P2P networks, which can be easily expanded to Grid and cloud networks. In the DDBF algorithm, we proposed a distributed algorithm whose nodes are able to create a reliable backbone for flooding in the cloud network. The DDBF algorithm ensures connectivity in the created backbone even by disconnecting and connecting nodes rapidly. Thus, the reliability of receiving flooded query by all of the involved nodes on the network is ensured. Finally, we compared the performance of the flooding schemes. The results of the experimental tests in five different topologies verify the proposed DDBF algorithm's improvements. Most importantly in the results obtained for dynamic network topology, in terms of the reduced engaged nodes at DDBF is 42% in comparison to the TF, and 24% in comparison to the lightweight algorithm. Also, in terms of the reduced flooded queries at DDBF in comparison to the TF is 67%, and in comparison to the lightweight algorithm is 34%. In general in all different topologies DDBF achieves better network performance in comparison to existing state-of-the-art algorithms.

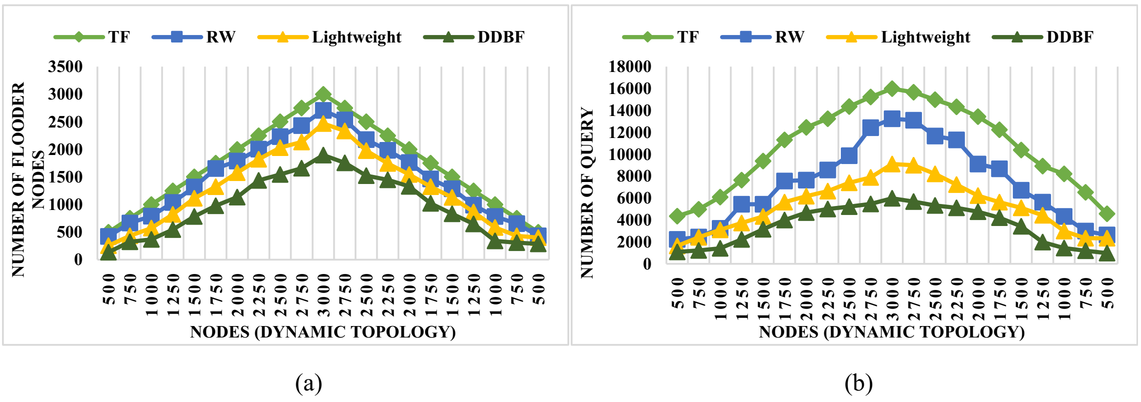

Fig. 13 Comparison TF, RW, Lightweight path, and DDBF flooding algorithms on the dynamic topology in terms of (a) number of engaged nodes for query flooding (b) number of flooded queries on the network

In the future work we will focus on the proposing one backbone based single solution for load balancing for flooder nodes on all kinds of the network's topologies.

References

1. Zarrin J, Aguiar RL, Barraca JP (2018) Resource discovery for distributed computing systems: A comprehensive survey. J Parallel Distrib Comput 113:127–166. https://doi.org/10.1016/j.jpdc.2017.11.010
2. Kryukov A, Demichev A (2018) Decentralized Data Storages: Technologies of Construction. Program Comput Softw 44(5): 303–315. https://doi.org/10.1134/S0361768818050067
3. Trunfio P, Talia D, Papadakis H, Fragopoulou P, Mordacchini M, Pennanen M, Popov K, Vlassov V, Haridi S (2007) Peer-to-Peer resource discovery in grids: Models and systems. Futur Gener Comput Syst 23(7):864–878. https://doi.org/10.1016/j.future.2006.12.003
4. Kang S, Kim T, Jeon H, Lee W, Kang S (2015) A healthcare information sharing scheme in distributed cloud networks. Clust Comput:1–6. https://doi.org/10.1007/s10586-015-0488-y
5. Hu W-C, Kaabouch N (2012) Sustainable ICTs and management systems for green computing. IGI Global,
6. Zhou J, Shi Z (2010) Unstructured P2P-enabled service discovery in the cloud environment. In: Intelligent Information Processing V, vol 340. Springer Berlin Heidelberg, pp 173–182. doi:https://doi.org/10.1007/978-3-642-16327-2_23
7. Lv Q, Cao P, Cohen E, Li K, Shenker S (2002) Search and replication in unstructured peer-to-peer networks. In: ICS, New York, NY, USA, ACM, pp 84–95
8. Crespo A, Garcia-Molina H (2002) Routing indices for peer-to-peer systems. Paper presented at the Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on
9. Madan M, Mathur M (2014) Cloud network management model - A novel approach to manage cloud traffic. Int J Cloud Comput Serv Archit (IJCCSA) 4(5). https://doi.org/10.5121/ijccsa.2014.4502
10. Daswani S, Fisk A (2002) Gnutella UDP extension for scalable searches (GUESS)
11. Trunfioa P, Taliaa D, Papadakisb H, Fragopouloub P, Mordacchinic M, Pennanend M, Popove K, Vlassovf V, Haridif S (2007) Peer-to-Peer resource discovery in Grids: Models and systems. Futur Gener Comput Syst 23(7):864–878. https://doi.org/10.1016/j.future.2006.12.003
12. Ripeanu M (2001) Peer-to-peer architecture case study: Gnutella network. In: Peer-to-Peer Computing, 2001. Proceedings. First International Conference on. IEEE, pp 99–100. doi:https://doi.org/10.1109/P2P.2001.990433
13. The gnutella protocol specification v0.4. http://rfc-gnutella.sourceforge.net/developer/stable. Accessed 01 Apr 2018
14. Gkantsidis C, Mihail M, Saberi A (2004) Random walks in peer-to-peer networks. Perform Eval P2P comput syst 63(3):241–263. https://doi.org/10.1016/j.peva.2005.01.002
15. Cambazoglu BB, Varol E, Kayaaslan E, Aykanat C, Baeza-Yates R (2010) Query forwarding in geographically distributed search engines. In: SIGIR '10 Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval New York, NY, USA. ACM, pp 90–97. doi:https://doi.org/10.1145/1835449.1835467
16. Noghabia HB, Ismaila AS, Ahmeda AA, Khodaeib M (2012) Optimized query forwarding for resource discovery in unstructured peer-to-peer grids. Cybern Syst Int J 43(8):687–703. https://doi.org/10.1080/01969722.2012.717860
17. Lee W, Kim T, Kang S, Kim H (2015) Revised P2P data sharing scheme over distributed cloud networks. In: Information Science and Applications. Lecture Notes in Electrical Engineering, vol 339. Springer Berlin Heidelberg, pp 165–171. doi:https://doi.org/10.1007/978-3-662-46578-3_20
18. Cheng L, Wang C-L (2013) Network performance isolation for latency-sensitive cloud applications. Futur Gener Comput Syst 24(9):1073–1084. https://doi.org/10.1016/j.future.2012.05.025
19. Wu D (2014) iCloudAccess: Cost-effective streaming of video games from the cloud with low latency. IEEE Trans Circuits Syst Video Technol 24(8):1405–1416. https://doi.org/10.1109/TCSVT.2014.2302543
20. Shehab A, Elhoseny M, El Aziz MA, Hassanien AE (2018) Efficient schemes for playout latency reduction in P2P-VoD systems. In: Advances in Soft Computing and Machine Learning in Image Processing. Springer, pp 477–495. doi:https://doi.org/10.1007/978-3-319-63754-9_22
21. Sen S, Wang J (2002) Analyzing peer-to-peer traffic across large networks. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment. ACM, pp 137–150. doi:https://doi.org/10.1145/637201.637222
22. Barjini H, Othman M, Ibrahim H, Udzir NI (2012) Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured P2P networks. Peer-to-Peer Networking and Applications 5:1):1–1)13. https://doi.org/10.1016/j.jpdc.2017.11.010
23. Lv Q, Cao P, Cohen E, Li K, Shenker S (2002) Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th international conference on Supercomputing. ACM, pp 84–95. doi:https://doi.org/10.1145/514191.514206
24. Aiello W, Chung F, Lu L (2000) A random graph model for massive graphs. In: Proceedings of the thirty-second annual ACM symposium on Theory of computing. Acm, pp 171–180. doi:https://doi.org/10.1145/335305.335326
25. Calvert KL, Doar MB, Zegura EW (1997) Modeling internet topology. IEEE Commun Mag 35(6):160–163. https://doi.org/10.1109/35.587723
26. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the grid: Enabling scalable virtual organizations. Int J High Perform Comput Appl 15(3):200–222. https://doi.org/10.1177/109434200101500302
27. Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on World Wide Web. ACM, pp 640–651. doi:https://doi.org/10.1145/775152.775242
28. Darlagiannis V, Mauthe A, Steinmetz R (2004) Overlay design mechanisms for heterogeneous, large-scale, dynamic P2P systems. J Netw Syst Manag 12(3):371–395. https://doi.org/10.1023/B:JONS.0000043686.04679.03
29. Budhkar S, Tamarapalli V (2018) Delay management in mesh-based P2P live streaming using a three-stage peer selection strategy. J Netw Syst Manag 26(2):401–425. https://doi.org/10.1007/s10922-017-9420-5
30. Jararweha Y, Al-Ayyouba M, Darabseha A, Benkhelifab E, Voukc M, Rindos A (2016) Software defined cloud: Survey, system and evaluation. Futur Gener Comput Syst 58:56–74. https://doi.org/10.1016/j.future.2015.10.015
31. Greenberg A, Hamilton J, Maltz DA, Patel P (2008) The cost of a cloud: Research problems in data center networks. ACM SIGCOMM Comput Commun Rev 39(1):68–73. https://doi.org/10.1145/1496091.1496103
32. Margariti SV, Dimakopoulos VV (2015) On probabilistic flooding search over unstructured peer-to-peer networks. Peer-to-Peer Networking and Applications 8(3):447–458. https://doi.org/10.1007/s12083-014-0267-1

33. Kalogeraki V, Gunopulos D, ZeinalipourYazti D (2002) A local search mechanism for PeertoPeer networks. In: CIKM '02 Proceedings of the eleventh international conference on Information and knowledge management, New York, NY, USA. ACM, pp 300–307. doi:https://doi.org/10.1145/584792.584842
34. Mark K, Manfred H (2006) Adding structure to Gnutella to improve search performance in a real-world deployment. Distributed Information Systems Laboratory
35. Exarchakos G, Antonopoulos N (2007) Resource sharing architecture for cooperative heterogeneous P2P overlays. J Netw Syst Manag 15(3):311–334. https://doi.org/10.1007/s10922-007-9069-6
36. Kim H, Kim Y, Kim K, Kang S (2008) Restricted path flooding scheme in distributed P2P overlay networks. Paper presented at the International Conference on Information Science and Security, Seoul, 10–12 Jan
37. Baumann H, Crescenzi P, Fraigniaud P (2014) Flooding in dynamic graphs with arbitrary degree sequence. J Parallel Distrib Comput 74(5):2433–2437. https://doi.org/10.1016/j.jpdc.2014.01.007
38. Purohit GN, Sharma U (2010) Constructing minimum connected dominating set: Algorithmic approach. International journal on applications of graph theory in wireless ad hoc networks and sensor networks (GRAPH-HOC) 2 (3). doi:https://doi.org/10.5121/jgraphoc.2010.2305
39. Zhang Z, Zhou J, Mo Y, Du D-Z (2016) Performance-guaranteed approximation algorithm for fault-tolerant connected dominating set in wireless networks. Paper presented at the The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016
40. Guha S, Khuller S (1998) Approximation algorithms for connected dominating sets. Algorithmica 20(4):374–387. https://doi.org/10.1007/PL00009201
41. Wu J, Li H (1999) On calculating connected dominating set for efficient routing in ad hoc wireless networks. Paper presented at the Proceedings of the 3rd In ternational Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications,
42. Nieberg T, Hurink J (2006) A ptas for the minimum dominating set problem in unit disk graphs. In: Approximation and Online Algorithms, vol 3879. Springer, pp 296–306. doi:https://doi.org/10.1007/11671411_23
43. Mohanty JP, Mandal C, Reade C, Das A (2016) Construction of minimum connected dominating set in wireless sensor networks using pseudo dominating set. Ad Hoc Netw 42:61–73. https://doi.org/10.1016/j.adhoc.2016.02.003
44. Abid SA, Othman M, Shah N (2014) 3D P2P overlay over MANETs. Comput Netw 64:89–111. https://doi.org/10.1016/j.comnet.2014.02.006
45. Rahmani M, Benchaïba M (2018) PCSM: an efficient multihop proximity aware clustering scheme for mobile peer-to-peer systems. J Ambient Intell Humaniz Comput:1–18. https://doi.org/10.1007/s12652-018-0808-1
46. Rahmani M, Benchaba M, Seddiki M A Clustering-based Replication Strategy for Mobile P2P networks. In: 2018 International Conference on Applied Smart Systems (ICASS), 2018. IEEE, pp 1–6. doi:https://doi.org/10.1109/ICASS.2018.8651946
47. Shen X-J, Chang Q, Liu L, Panneerselvam J, Zha Z-J (2016) CCLBR: Congestion control-based load balanced routing in unstructured P2P systems. IEEE Syst J 12(1):802–813. https://doi.org/10.1109/JSYST.2016.2558515
48. Hughes D, Coulson G, Walkerdine J (2005) Free riding on Gnutella revisited: the bell tolls? IEEE distrib syst online 6(6). https://doi.org/10.1109/MDSO.2005.31
49. Zang C, Cui P, Faloutsos C, Zhu W (2018) On power law growth of social networks. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2018.2801844
50. Zegura E (1996) GT-ITM: Georgia tech internetwork topology models (software). Georgia Tech," http://www.cc.gatech.edu/projects/gtitm. Accessed 01 May 2018
51. Breslau L, Cao P, Fan L, Phillips G, Shenker S (1999) Web caching and Zipf-like distributions: Evidence and implications. In: INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, IEEE, pp 126–134. doi:10.1109/INFCOM.1999.749260
52. Zhang L, Deng Y, Zhu W, Zhou J, Wang F (2015) Skewly replicating hot data to construct a power-efficient storage cluster. J Netw Comput Appl 50:168–179. https://doi.org/10.1016/j.jnca.2014.06.005

Saeed Saeedvand received his B.S. (2011) and M.S. (2014) degrees in Computer Engineering. Currently, he is Ph.D. candidate in Departement of Information Technology, Faculty of Electrical and Computer Engineering, University of Tabriz (5166616471), Tabriz, Iran. He has been worked on the humanoid adult-size, and kid-size robots since 2009. He is a member of the technical committee of IranOpen robotic competitions since 2016. His current researches includes artificial intelligence, robotics, machine learning, and computer vision.

Hadi S. Aghdasi received his B.S. (2006) from Sadjad University of Technology, Mashhad, Iran, M.S. (2008) and Ph.D. (2013) degrees From Shahid Beheshti University, Tehran, Iran. All are in Computer Engineering. He is currently assistant professor in the Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Tabriz (5166616471), Tabriz, Iran. His current researches focus on routing and medium access control, topology models, image and video transmission in Wireless Networks. He is director of the both Wireless Ad hoc and Sensor networks research Laboratory and Humanoid Robots and Cognitive Technology research laboratory in University of Tabriz.

Leili Mohammad Khanli received her B.S. (1995) from Shahid Beheshti University, Tehran, Iran, M.S. (2000) and Ph.D. degrees (2007) from Iran University of Science and Technology, Tehran, Iran. All are in Computer Engineering. She is currently associate professor in the Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Tabriz (5166616471), Tabriz, Iran. Her research interests include Cloud computing, and Quality of Service management. She is director of Cloud Computing research laboratory in University of Tabriz.