Computer Communications 32 (2009) 1445–1459

Contents lists available at ScienceDirect

Computer Communications

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

Enhancing peer-to-peer content discovery techniques over mobile ad hoc networks

Diego N. da Hora [a], Daniel F. Macedo [b,*], Leonardo B. Oliveira [c], Isabela G. Siqueira [a], Antonio A.F. Loureiro [a], José M. Nogueira [a], Guy Pujolle [b]

[a] Computer Science Department, Federal University of Minas Gerais, Belo Horizonte, MG, Brazil
[b] Laboratoire d'Informatique Paris VI (LIP6), Paris Universitas, Paris, France
[c] Computing Institute, Federal State University of Campinas, Campinas, SP, Brazil

ARTICLE INFO

Article history:
Received 10 October 2008
Received in revised form 18 March 2009
Accepted 27 April 2009
Available online 9 May 2009

Keywords:
Mobile ad hoc networks
Peer-to-peer
Performance evaluation
Simulations

ABSTRACT

Content dissemination over mobile ad hoc networks (MANETs) is usually performed using peer-to-peer (P2P) networks due to its increased resiliency and efficiency when compared to client–server approaches. P2P networks are usually divided into two types, structured and unstructured, based on their content discovery strategy. Unstructured networks use controlled flooding, while structured networks use distributed indexes. This article evaluates the performance of these two approaches over MANETs and proposes modifications to improve their performance. Results show that unstructured protocols are extremely resilient, however they are not scalable and present high energy consumption and delay. Structured protocols are more energy-efficient, however they have a poor performance in dynamic environments due to the frequent loss of query messages. Based on those observations, we employ selective forwarding to decrease the bandwidth consumption in unstructured networks, and introduce redundant query messages in structured P2P networks to increase their success ratio.

1. Introduction

Mobile ad hoc networks (MANETs) are frequently employed in rescue operations and in battle fields due to their independence to previous infrastructure [1,2]. The characteristics of such scenarios preclude the use of client–server architectures, since servers become points of vulnerability. The peer-to-peer (P2P) [3,4] paradigm thus is more suitable on such situations, since it mitigates task overload by distributing them among nodes [1,5].

The synergy among P2P networks and MANETs is a well-studied topic in the literature [1,6–11]. Those networks present several similarities. Both are decentralized and self-organizing, have dynamic topology, and route queries in a distributed environment. In addition, nodes have equivalent functions and capabilities, as they send and reply to requests originated from one another. MANETs and P2P networks are not only similar, but also complementary. Because nodes in MANETs usually have low computing capacity and, therefore, are unable to play the role of servers all the time, a P2P application is the ideal tool to disseminate information in this scenario.

Since a P2P network does not possess a unique service provider at a given time, the assignment of tasks among nodes prevents them to become overloaded. In addition, some applications enabled by MANETs (e.g., rescue team communication in disaster situations and exchange of information in battle fields) require users to cooperate with others, e.g., a rescue team worker might request situation reports from his nearest neighbors. Although a central server could store this information, this approach would be more expensive (the information would travel more hops) and less resilient, as the server becomes a point of failure and target of attacks.

P2P networks can be classified into two categories, based on how they organize data [3]. In *unstructured* networks, data is spread on the network without any kind of planning. Hence, a node willing to find a certain information must ask all the nodes of the P2P network if they have a copy of a given object. Example networks are Freenet and Gnutella. In *structured* networks, the information is organized following certain criteria, in order to ease content searching. Thus, when a node wishes to find a certain information, it uses distributed indexes to quickly find where the information is stored. Example networks are Chord, CAN and PASTRY. Those indexes, also known as *distributed hash tables* (DHTs), are built in a way that nodes are responsible for an homogeneous amount of data, and also allow queries to be resolved with much less messages than on unstructured approaches. While unstructured networks impose a much larger network load, they are simple to implement and nodes can easily join the network. For structured networks, on the other hand, each node that enters the network must register in the DHT each file that it will share.

* Corresponding author. Tel.: +33 1 44 27 87 86.
E-mail addresses: dnhora@dcc.ufmg.br (D.N. da Hora), Daniel.Macedo@rp.lip6.fr (D.F. Macedo), leob@ic.unicamp.br (L.B. Oliveira), isabela@dcc.ufmg.br (I.G. Siqueira), loureiro@dcc.ufmg.br (A.A.F. Loureiro), jmarcos@dcc.ufmg.br (J.M. Nogueira), guy.pujolle@rp.lip6.fr (G. Pujolle).

0140-3664/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2009.04.005

Previous work evaluated the performance of those content discovery techniques in P2P networks on wired scenarios [12–14]. Their results are not applicable to MANETs, since the wireless medium is much more dynamic due to node mobility and the frequent variations in channel quality due to interference and fading. Although the literature is ripe with P2P protocols and applications tailored to MANETs, we lack the understanding of how each of the underlying factors of a MANET (node mobility, link quality, node density) influences the performance of a P2P application. Such an understanding is useful for designing new protocols, for example to stress which physical and environmental factors should be considered in the design, and which could be left out and to identify the weight of each factor on the overall performance, allowing the creation of solutions specific to a certain deployment (i.e., low mobility, low disconnection scenarios).

This article brings two contributions. First, it evaluates existent content discovery techniques over models richer than those of previous work. Our models take into account mobility, lossy channels, collisions, number of nodes, amount of queries, node connections/disconnections and number of file replicas, providing a more realistic portrait of P2P networks over MANETs. Results show that protocols that make use of redundant query messages (i.e., unstructured protocols) are, in general, more efficient for MANETs, although they consume more energy than structured protocols. Structured protocols, on the other hand, are more suitable for static environments [15].

Our second contribution is the enhancement of structured and unstructured networks in order to counter the limitations unveiled in the performance study above by the use of redundancy in structured networks and the use of Gossiping in unstructured networks. Due to the generality of our proposals, they can be used in any P2P protocol of the two classes studied. The proposed extensions for unstructured networks reduce average energy consumption and delay, while our solutions for structured networks increase the success ratio at the expense of average energy consumption and delay [16].

The rest of this article is organized as follows. Section 2 discusses the related work. Section 3 briefly describes the two P2P protocols used in our evaluation. Section 4 presents the simulation environment and the analysis method used in this work. Section 5 presents the performance evaluation of existing content discovery techniques. Next, Section 6 describes our enhancement to P2P content discovery over MANETs, followed by their evaluation in Section 7. Finally, Section 8 draws the conclusions.

2. Related work

P2P networking is a prolific research topic of wired networks. Several protocols have been proposed (see [4] for a summary). However, there are few comprehensive performance analysis of such protocols. Lv et al. studied the performance of unstructured P2P networks in the Internet for various forwarding strategies, file popularity and query distributions [17]. The authors propose two new forwarding strategies to counter the high load imposed on the network due to the amount of query messages sent. Ge et al. used markov chains to study the performance of P2P networks for a very large number of nodes [12]. The authors evaluated centralized, structured and unstructured P2P networks under ideal channel conditions, showing the limitations of both centralized and unstructured networks due to the traffic imposed on certain nodes.

Random walks have been proposed as a solution for the poor performance of unstructured protocols [17–20]. The principle works as follows. Instead of flooding the network with queries, random walk based protocols rely on a fixed amount of messages, called *walkers*, which wander around the network. Whenever a node receives a walker, it checks if it has the requested information. If not, it forwards the walker to one of its neighbors, selected at random. Walkers have a time to live (TTL) in order to purge unsuccessful queries. The key to this approach, thus, is setting the correct value for the TTL and to know how to use topology information to bias the forwarding decision towards nodes with high probability of having the information sought. random walks in the Internet solve as many queries as flooding-based strategies and are more scalable, however the response time is higher. The protocol Adaptive Probabilistic Search (APS) is an adaptive version of random walks, where the probability of following a path changes according to previous queries [21].

Due to the strong resemblance and synergy of MANETs and P2P networks, several articles study how both interoperate. Schollmeier et al. [6] and Borg [1] discuss similarities and differences of MANETs and P2P networks. Hu et al. [9] proposed the Dynamic P2P Source Routing (DPSR) protocol, an ad hoc routing protocol that employs strategies used by DSR routing protocol [22] and the Pastry P2P protocol [23] to improve scalability. Oliveira et al. [7] studied an unstructured P2P application running over a MANET under three different ad hoc protocols (DSR, AODV, DSDV).

Studies showed that P2P protocols developed for wired networks present poor performance over MANETs, due to the unique characteristics of those networks: the harshness of the medium, frequent node connections/disconnections and the frequent creation of disconnected components on the network topology [24]. Further, a handful of P2P protocols have been specifically tailored to MANET environments. Franciscani et al. [8] concentrated on minimizing the impact of the highly dynamic topology obtained of P2P networks over MANETs by proposing algorithms for the (re)configuration of the P2P topology. Srinivasan et al. [25] proposed 7DS, which enables the exchange of data among peers not directly connected to the network by exploring peer mobility to opportunistically forward queries. Klemm et al. [11] presented ORIOM, a P2P protocol that sets up overlay paths on demand. Schollmeier et al. proposed MPP, which uses cross layer communication to interlink its routing layer to the underlying physical layer and in turn adapt its virtual topology to the physical one [26]. Besides file sharing, it also provides location aware services.

MANET deployments may differ significantly due to the diversity of the employed hardware, mobility constraints and the existence of fixed nodes. Thus, P2P protocols tailored to a specific class of network or deployment have been proposed. Lee et al. proposed Bit Torrent-like protocols to vehicular MANETS (VANETs) [27]. VANETs have high mobility and quite dynamic link qualities. The proposed solution, called CodeTorrent, uses network coding combined with a Gossiping-like broadcast strategy to allow nodes to cope with the conditions of those MANETs.

In virtual collaborative environments (VCE), a hybrid MANET-infrastructured network is used to disseminate geographical data and reports. Such a P2P network was designed to military or emergency applications, where different groups share the same networking infrastructure [28,29]. In such scenarios usually there are static nodes with stronger capabilities, allowing the use of hierarchy to increase scalability. In the Workpad project, a hybrid MANET-infrastructured network is devised [28]. The wired network acts as a backbone for several deployed MANETs. A P2P network is created at the MANET level for local communication, while another P2P runs on the backbone. Boukerche et al. built a single Gnutella network with several "channels", which act similarly as multicast groups [29]. This organization was employed in order to reduce the amount of control information that must be exchanged and stored by each P2P peer.

Although the literature is ripe with P2P protocols and applications tailored to MANETs, it is not yet clear how the properties of

a MANET, such as node mobility, link quality or node density, influence the performance of P2P applications. This knowledge would be useful for devising smarter protocols, helping in the identification of the key factors in the performance of a given scenario or deployment. Such a study has been performed on a theoretical level by Ding and Bhargava [10]. This study presented complexity results in *O*-notation. Nevertheless, they did not take into account important aspects such as mobility and channel error. Further, a comprehensive study is needed to evaluate the effect of collisions, mobility and high channel error. This article is a step towards this direction.

3. Description of the evaluated protocols

In our evaluation, we employ Gnutella and random walks to represent the unstructured P2P approach, while we employ Chord to represent the structured approach. Before presenting our results, we will briefly describe the operation of these protocols.

3.1. Unstructured protocols

Gnutella propagates queries by controlled flooding. Whenever a node receives a query, it tries to resolve it locally. If the resolution succeeds, a RESULT message is returned directly to the query source. Otherwise, the query is forwarded to all peers in a neighbors list. To avoid that queries propagate indefinitely, there is a TTL field similar to that in the IP protocol embedded in every query message, which is decremented at each hop. Messages with TTL zero or duplicated messages are discarded.

In this work, we randomly select the neighbors of a peer out of the nodes that are online at the time the peer joins the P2P network. This assignment is done offline, similar to a central server that acts as the P2P network entry point (this is usual in most Internet Gnutella clients). P2P networks have a dynamic behavior due to peers joining and leaving the network. Thus, the Gnutella protocol includes a topology control scheme for maintaining an up-to-date list of neighbors. To accomplish this, all peers periodically send PING messages to their neighbors and wait for an acknowledgment (the PONG message) in order to check if their neighbors are still online. When no answer is received from a neighbor it is replaced by a peer randomly chosen from the set of online peers.

We also evaluated a random walk based protocol. In this protocol, the sender of the query instantiates n walkers, or query messages, with fixed TTL and then sends each walker to one of its neighbors, selected with a uniform probability. Whenever a peer receives a walker, it checks if it owns the requested information. If it does, the walker is removed from the network and the peer contacts the source of the query. If the information is not found locally and the TTL of the walker is not zero, then the walker is forwarded and its TTL is decremented by one unit. As in Gnutella, we use a walker cache to avoid multiple receptions of the same query, since initial experiments (not described in this article) showed that such a cache improves the performance of the network. Further, peers use PING-PONG messages to control the availability of their neighbors.

In general, random walk and Gnutella (or other flooding-based unstructured protocols) can be represented as a single framework, where different parameter values define the operational mode: flooding-based or random walk based. Imagine a forwarding protocol where a node forwards new messages to e neighbors of its neighbor list containing n neighbors if the TTL is higher than zero. Further, if we call t the initial TTL and q the number of queries created at the node originating the request, we can see that Gnutella is a protocol where $n = e,\ e > 1,\ q = 1$ and t is very small (usually not more than 5 or 10). Meanwhile, random walk is a protocol where $n > e,\ e = 1,\ q > 1$ and t is set to a large value (usually, a function of the number of nodes). Those changes produce completely different results due to the characteristics of the resulting forwarding mechanism. In this general protocol, the amount of message forwards will be approximately equal to $\sum_{T=1}^{t}(q \times e^t)$. Replacing the values in the formula, we see that random walk produces at most $q \times t$ message forwards, while Gnutella will produce at most $\sum_{T=1}^{t} e^t = \frac{e(1-e^t)}{1-e}$ message forwards.[1] Further, the energy consumption of both will behave quite differently. Due to the almost sequential nature of forwarding in random walk, its response time tends to be much higher than Gnutella's, which has a more parallel query pattern.

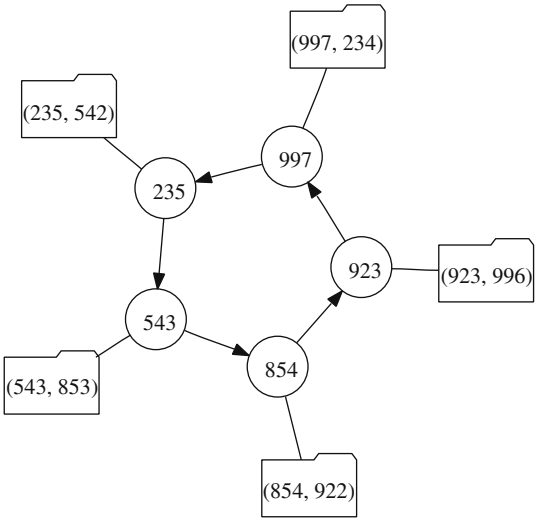

Fig. 1. Example five-node Chord network.

3.2. Structured protocol

To evaluate the performance of structured networks, we implemented the Chord protocol [30]. This choice was due to its simplicity and proved bounds.

Chord uses hashes to divide files among the peers. All peers are arranged in a ring, where the position of each peer in the ring is based on its IP address. The order of the peers is in increasing hash value, as shown in Fig. 1. Peers are responsible for files having hashes (identifiers) within the range $[hash(my\_ip), hash(my\_successor\_ip)$, e.g., in the figure, the peer with hash 543 will know the location of files with hashes from 543 up to 853 (in the figure, hash ranges are written in the folders linked to the peers). Statistically, in a large network with n peers, each peer is responsible for approximately $\frac{1}{n}$ of the hash space.

The protocol performs efficient lookup using the *finger table*, which is a carefully crafted index. Suppose that we are looking for file with hash 625, thus we have to find the node that takes care of this hash value. Initially all peers are potential candidates. Chord then checks, for each received query, which of the peers in the finger table has the closest hash value, and then forwards the query to it.

The trick of Chord is that each entry i in the finger table is the peer having the highest $hash(IP_i)$ such that $hash(IP_i) \leqslant (myhash + 2^i) mod 2^m$, where m is the number of bits generated by the hash function and *myhash* is the hash of the IP of the current peer. This allows queries to be solved at approximately $log(n)$ message forwards by

[1] The equations do not take into account losses due to failed nodes or variations in the wireless channel. They also do not model the effect of query caches, which drop already seen queries, instead of forwarding them.

Table 1
Characterization of the default simulation scenario.

Parameter	Value
MAC protocol	IEEE 802.11b
Number of nodes	50
Area size	1 km^2
Node placement	Uniform distribution
Node mobility	Random way-point, $0 \leqslant speed \leqslant 1.0$ m/s
Percentage of online nodes	50%
Number of files	250 different files in the network, 6 replicas per file
File queries	Each node queries 10% of the files, with query times following a uniform distribution
Queries per simulation	1250
Simulation time	200 s
Channel losses	0%

using a forward algorithm that implements a binary search in the hash space. Unlike unstructured protocols, which are based on flooding, structured protocols use only one query message due to the existence of a DHT. Chord is no exception, as it forwards queries using the finger tables.

The performance of Chord, as in any structured P2P protocol, relies on the accuracy of the DHT (for Chord, the finger table and the Chord ring). Thus, peers must frequently update all entries of the DHT. The dynamicity of the DHT is dictated by the amount of peers entering or leaving the P2P network. When a node leaves the network, it delegates the responsibility of its hash space to another peer. Thus, it transfers all the $[hash, IP]$ pairs to the delegated peer. Even more, all the finger tables pointing to the offline peer must be updated. A similar operation is required for peers entering the network, as those will claim a part of the hashes for themselves.

We implemented the complete set of Chord functionalities, including protocols for building and maintaining the distributed indexes. We also implemented file insertion and deletion in the network, using operations similar to those of file search.[2]

4. Network characterization

In order to evaluate the proposed protocols, we performed simulations with the NS-2 simulator, using the two-ray-ground radio propagation model. The workload simulates a search and rescue operation, where Wi-Fi devices form a P2P network over a MANET. The P2P protocols are implemented on top of the UDP protocol, since TCP does not perform well in this type of environment [31,32]. We chose AODV [33] for routing as it presents the best performance under a P2P application in most common MANET scenarios [7]. Nodes use IEEE 802.11b radios modeled upon a Cisco Aironet 350 card [34] with a transmission power of 10 dBm. In this setting, the radio consumes 1.6887 W for transmission, 0.6699 W for idle and 1.0791 W for reception modes. The interface queue length is 30 packets.

The simulations are based on a default scenario that we consider as being the closest to the target application conditions we envision. We assume a network of 50 nodes scattered in a 1500 m $\times$ 1500 m grid area following an uniform distribution. Nodes move accordingly to the random way-point mobility model (since it is frequently used for individual movement [35]) with a pause time of 0.1 s and an average speed uniformly selected from 0 to 1.0 m/s. At any given point of the simulation, 80% of the nodes are always online, while the remaining nodes join the network at some point and leave after some time. Join and leave times follow an uniform distribution. Each node provides five different files, thus there are 250 different files in the network. Each file has, in average, six replicas randomly distributed among the peers. In the default scenario we do not consider losses due to channel error. We consider only losses due to collisions, not losses due to interference or momentary medium degradation. Table 1 summarizes the parameters used in the default simulation scenario. In the following sessions, we will always employ the default scenario, varying one simulation parameter while leaving the others fixed.

We evaluated three protocols: Chord, Gnutella and random walk. The first one was chosen in order to provide results representative of the structured paradigm, while the others represent the unstructured paradigm. We picked more than one unstructured protocol because it is well-known in the literature that the performance of flooding-based protocols is quite different from random walk ones, as presented in Section 3.1. For a fair comparison, no optimizations that could improve the performance over ad hoc networks were implemented. For Chord, the *finger table* is updated every 5 s, *stabilize* function runs every 10 s and PING messages are sent every 10 s. Regarding Gnutella's simulation parameters, we assume that each node has a maximum number of four neighbors and a message cache of 100 application messages. The TTL for queries is set to 4 and the PING messages are sent every 10 s. For random walk, we defined the number of neighbors to be six. Each query spawns four walkers, which have a maximum TTL equal to one quarter of the number of nodes (e.g., 25 for a 100 node network). All packets have a fixed length of 64 bytes.

Each scenario is executed 33 times, using different seeds for the random number generator. Results are presented with a 99% confidence interval. We focus our analysis on three metrics: *hit rate* (the fraction of the queries successfully resolved in the P2P network), *response time* (delay perceived by a user requesting some content, including the time for transmitting the query to the network, locating the desired content and returning a response back to the user), *energy consumption* (the average energy consumed per node).

5. Classic content discovery over MANETs

We have analyzed the impact of the following factors on the performance of P2P networks over MANETs: network load, network size, number of replicas, channel error rate, mobility, and application dynamics. In the following, we describe each parameter and present the results obtained.

5.1. Network load

We analyzed the effect of network load over the performance of the three P2P protocols by varying the number of queries. In order

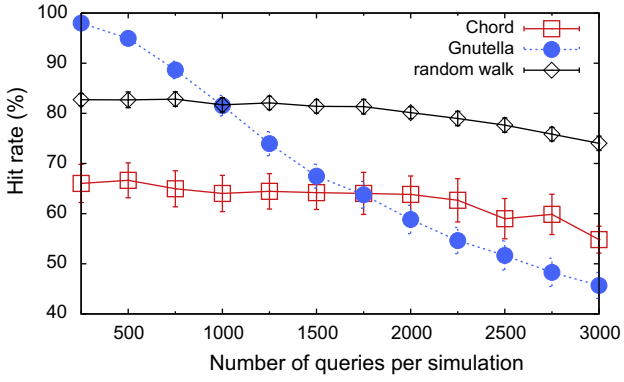

Fig. 2. Scenario 5.1: hit rate.

[2] Since the original article does not propose a mechanism to insert and delete file hashes.

to preserve the number of file replicas, we added more files to the network according to the number of queries. We varied the number of queries on the network from 450 to 3000. Queries were uniformly distributed among the simulation time.

Fig. 2 presents the hit rate of the three protocols. Gnutella presented the highest hit rate for a low query rate, achieving nearly 100%. Random walk perform the second best, maintaining a hit rate of 74% up to 82%. Chord performed the worst, resolving from 50% to 68% of the queries. Gnutella is the protocol that suffered the most with load variations, once its performance varied from 98% up to 46%. This performance is due to the number of messages sent, which defines the delay and the amount of query messages lost.

To support our conclusion, we present the response time (Fig. 3). For Gnutella, scenarios with more than 500 queries per simulation presented a response time higher than one second, which may already be considered a situation where the network is congested. For example, for 500 queries Chord and random walk presented a response time that is one order of magnitude lower. Chord and random walk also presented an exponential growth on their response time, however due to the reliance on less query messages their results were up to two orders of magnitude better than those of Gnutella.

The performance of this scenario is largely influenced by the number of messages sent. While Gnutella creates an exponential number of messages, random walk relies on only a few query messages, meanwhile Chord relies on a single query message. Fig. 4 shows that Gnutella has a significantly higher query overhead then Chord or random walk. For 3000 queries, Gnutella sends two times more messages than the other two protocols.

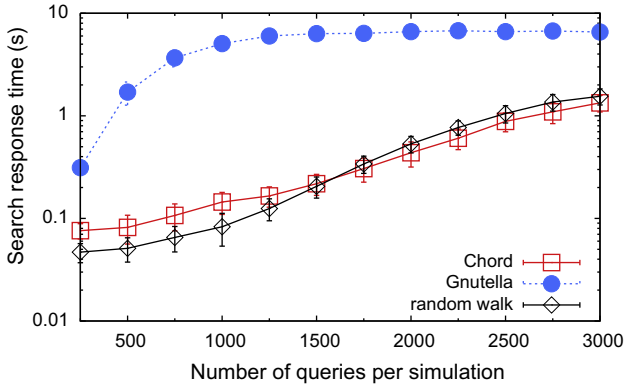

Fig. 3. Scenario 5.1: response time.

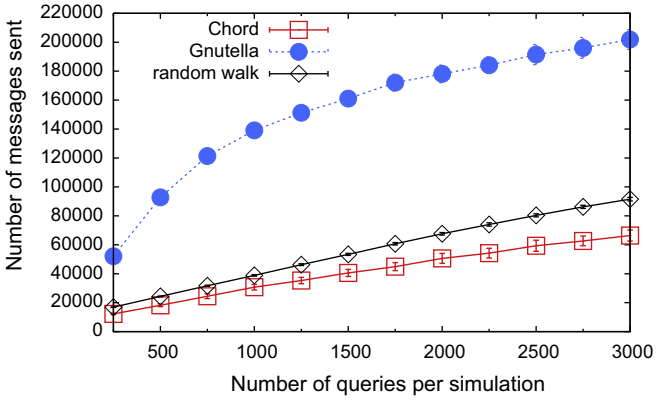

Fig. 4. Scenario 5.1: total number of msgs sent.

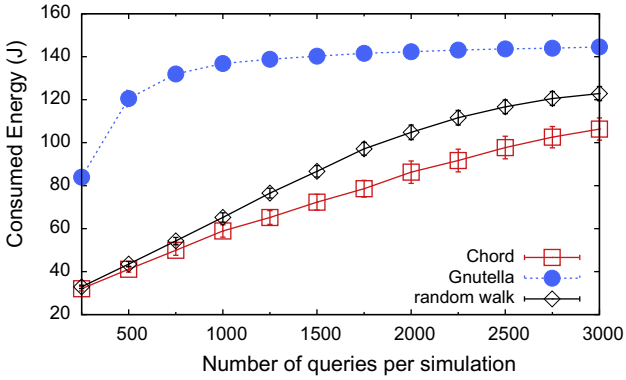

Fig. 5. Scenario 5.1: energy consumed.

Gnutella consumed from 20% up to 100% more energy than Chord and random walk, as Fig. 5 shows. Chord spent less energy than random walk for scenarios with higher loads, however on low loads both protocols tend to consume a similar amount of energy. The consumption curve of the protocols presented a tendency to flatten after a certain query rate. This occurs because, once the network is congested, the MAC layer reaches its performance limit, and discards the frames that it cannot serve. Hence the tendency for a flat energy curve and a constant reduction on the hit rate.

This scenario illustrates the impact of redundancy in queries. Unstructured protocols resolved more queries, due to the increased amount of redundancy. However, this comes at the cost of a higher response time and energy consumption. An increased redundancy was beneficial for networks with a light traffic, however more redundancy tends to saturate the network faster.

5.2. Network size

We also evaluate the impact of the number of nodes on the P2P network. The goal is to evaluate the performance of the protocols when the average hop count between two arbitrary nodes is different. The average hop count impacts search performance, as queries have to pass through more nodes before finding the desired content. We varied the number of nodes by changing the grid size, at the same time maintaining a fixed network density and number of queries.

Fig. 6 shows the hit rate. For all protocols, the number of queries solved (the hit rate) reduced with the number of nodes. The decrease in the hit rate coincides with a sharp increase in the experienced response time (Fig. 7), indicating that the source of the degradation is network contention due to more control messages from the protocol stack (i.e., routing) and from the P2P application.

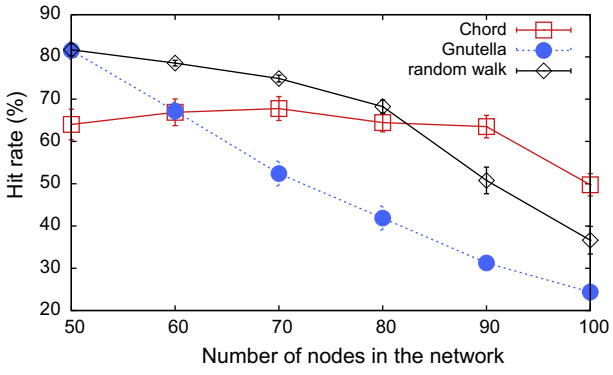

Fig. 6. Scenario 5.2: hit rate.

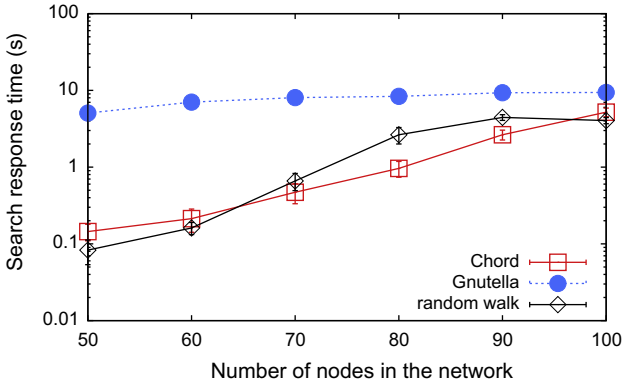

Fig. 7. Scenario 5.2: response time.

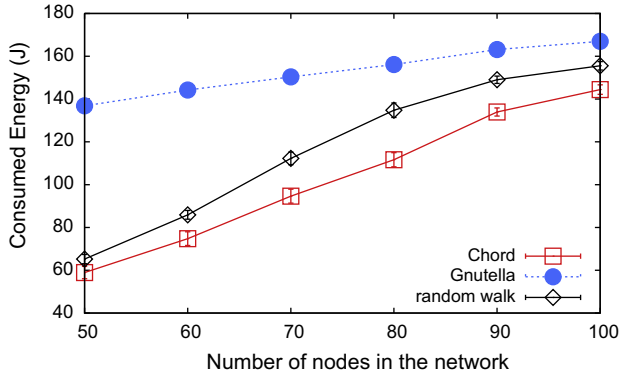

Fig. 8. Scenario 5.2: energy consumed.

For Chord, this moment occurs only at 100 nodes, while for Gnutella the network is already saturated for 50 nodes, and for random walk this happens for 80 nodes. Random walk performed the best in terms of hit rate and response time, resolving more queries than Gnutella and having a response time quite similar to that of Chord for networks of under 70 nodes.

The average energy consumption shown in Fig. 8. Gnutella is by far the most consuming protocol, consuming more than the double of energy than the other protocols for networks of 50 nodes. This difference tends to reduce with the number of nodes, due to contention. Contention can also be measured by the amount of messages dropped, shown in Fig. 9, where message drops increase by up to six times.

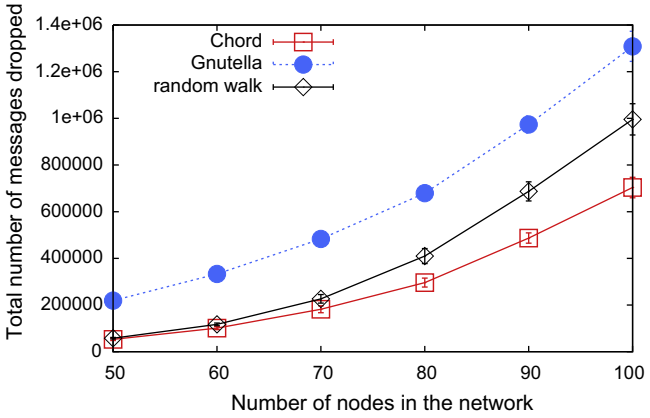

Fig. 9. Scenario 5.2: packets dropped.

5.3. Number of replicas

This scenario demonstrates the effect of the number of file replicas on the performance of the P2P network. Structured protocols do not take advantage of multiple copies of the file, since there is only one entry per unique file in the distributed index. Unstructured protocols, however, take full advantage of more replicas when searching for a particular file, since their lack of structure requires querying all the peers of the network, and thus more peers having the file will increase the probability of finding the file with less messages.

Fig. 10 shows the hit rate. The behavior of Chord is constant, as expected. On the other hand, Gnutella and random walk had significant performance improvements with the addition of replicas. In a network were each file exists only in one particular peer, Chord has a higher hit rate than random walk and even Gnutella protocols. For more than three replicas of the file, Gnutella and random walk perform better than Chord.

The response time of the queries is displayed in Fig. 11. Gnutella and random walk presented a constant response time. Meanwhile, Chord increased its response time with more replicas. We attribute this to the increased cost of maintaining more references to the files, as each file replica must be registered in the ring. This additional overhead incurred in a higher response time for the queries. The same trend can be seen in the energy consumption, as Fig. 12 shows. Still, Chord had the smallest response times and energy consumption among the evaluated protocols.

Fig. 13 shows the average number of virtual hops traversed by each solved query. This metric indicates how many peers a query has to go through before the requested information is found. Chord has the highest number of hops and, since the structure of the in-

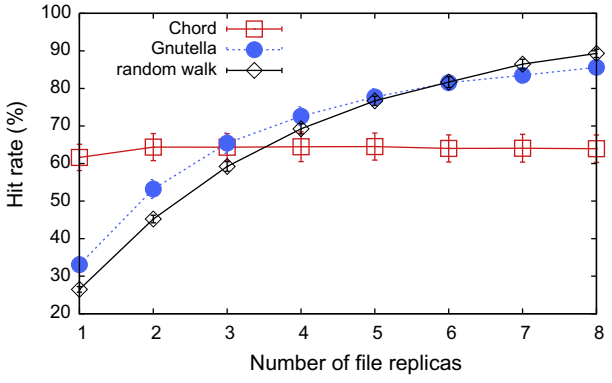

Fig. 10. Scenario 5.3: hit rate.

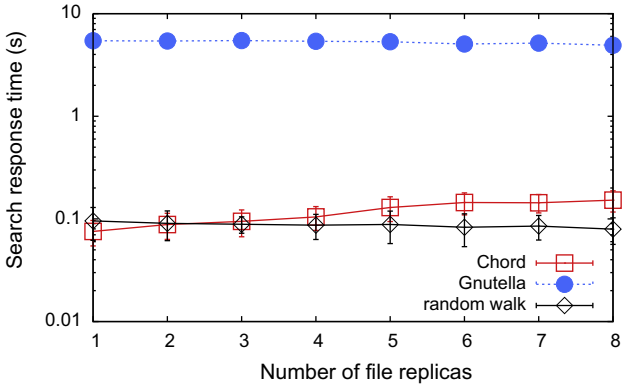

Fig. 11. Scenario 5.3: response time.

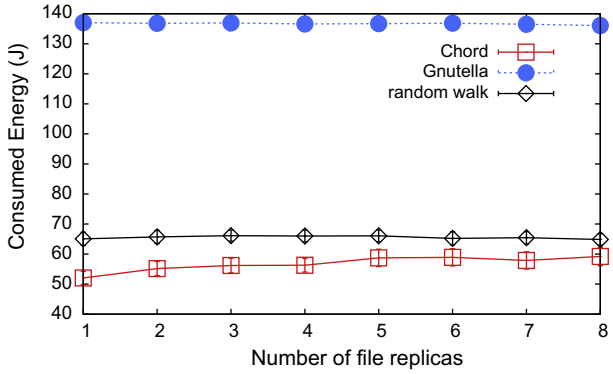

Fig. 12. Scenario 5.3: energy consumed.

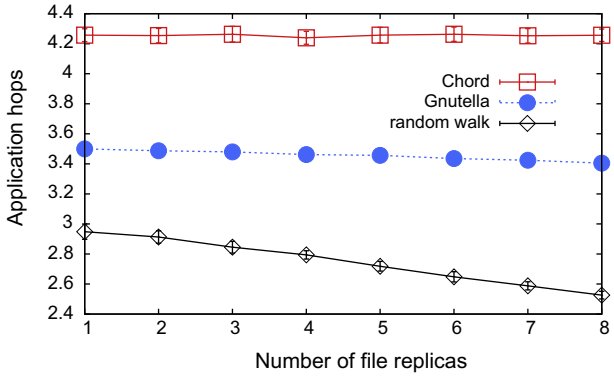

Fig. 13. Scenario 5.3: virtual hops.

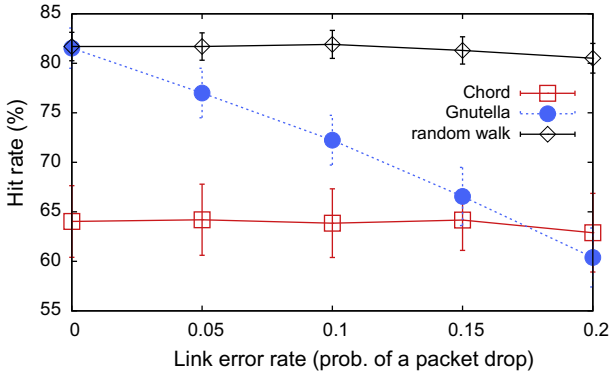

Fig. 14. Scenario 5.4: hit rate.

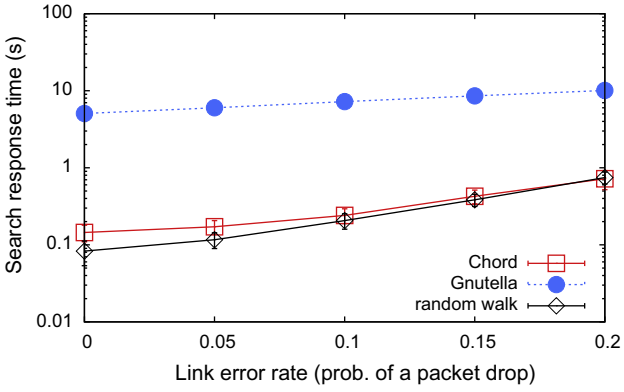

Fig. 15. Scenario 5.4: response time.

dex does not change with the number of replicas, the number of hops suffered an insignificant variation. For random walk and Gnutella, the number of query messages sent was smaller since queries finish earlier, once it is easier to find the information sought. Random walk had the biggest reduction, while Gnutella suffered a small decrease in the number of visited hops.

5.4. Channel error

Wireless networks typically face much higher bit error rates when compared to wired networks, and this leads to significant packet losses. Link quality is highly dependent to the environment and node position, varying significantly from one region to another [36,37]. As a characterization of network error rates for such environments is a daunting task and is out of the scope of our work, we varied packet loss probability uniformly over all nodes.

Fig. 14 shows the hit rate. Random walk and Chord had no significant losses in their hit rate, while for Gnutella the performance dropped by up to 27%. Although the hit rate is not quite affected, the response time suffered a significant increase, as shown in Fig. 15. This is due to the need for more retransmissions at the MAC layer. The retransmission mechanism reduces the amount of messages lost, which may explain the constant hit rate of Chord and random walk. For Gnutella, on the other hand, the number of retransmissions required is already near its maximum, since the network is congested. The sum of the two effects, high congestion and high link error rate, leads to packet drops.

More retransmissions leads to more energy consumption, as shown in Fig. 16. Random walk and Chord increased their energy consumption due to an increase in the error rate. Meanwhile, for Gnutella, the energy consumed reduced with high error rates. This

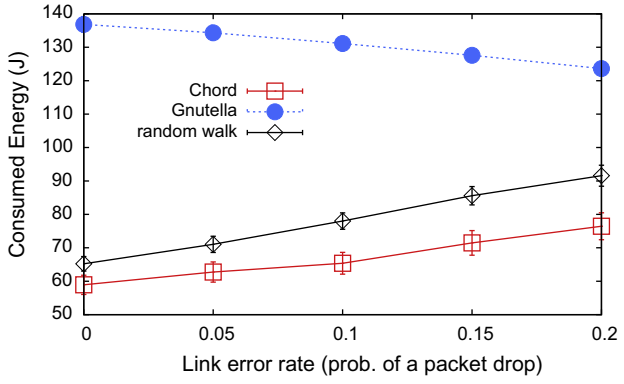

Fig. 16. Scenario 5.4: energy consumed.

is due to packet drops, as explained previously. For networks using Gnutella we identified an increase in packet drops due to routing errors, as Fig. 17 shows. Thus, without a proper route, Gnutella was unable to resolve queries.

5.5. Node mobility

A key difference between ad hoc networks and fixed networks is mobility. Due to node mobility, routes must be constantly updated. The same is true for queries in P2P networks, which expect that the destination of a query forward is reachable at a given time. P2P networks are tolerant to failures, as they accommodate node failure and disconnection, but current algorithms are designed to work in a wired environment, where disconnections are much less

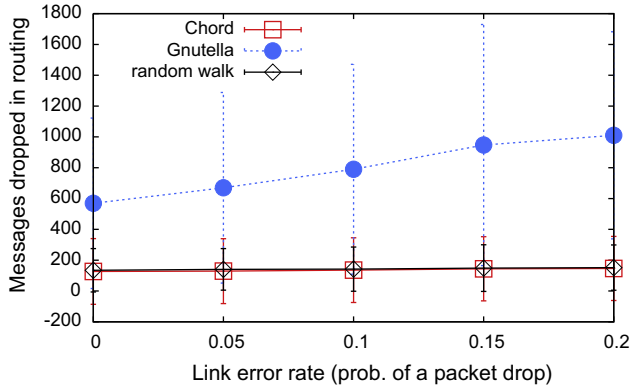

Fig. 17. Scenario 5.4: packets dropped at the routing level.

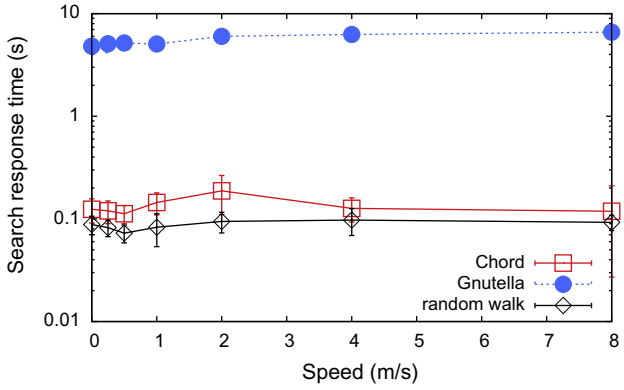

Fig. 19. Scenario 5.5: response time.

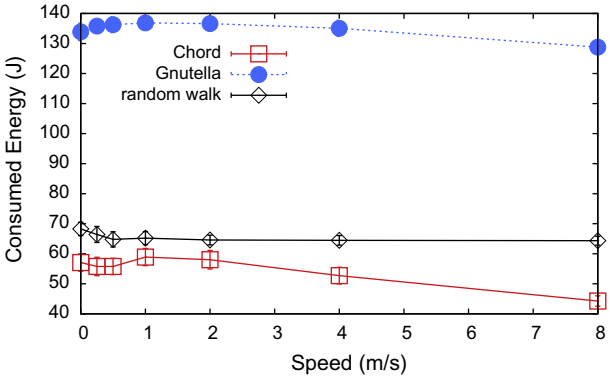

Fig. 20. Scenario 5.5: energy consumed.

frequent than in ad hoc networks. In this subsection, we evaluate the impact of physical topology changes in P2P protocols. We studied mobility by varying the average node speed from 0, 0.25, 0.5, 1, 2, 4 up to 8 m/s.

Chord suffered a significant performance degradation in terms of hit rate (60%), as Fig. 18 shows, while random walk and Gnutella had smaller losses, at the order of 10%. This occurred because unstructured protocols have some sort of redundancy in their queries, while structured protocols rely in only one message, which is repeatedly forwarded around the overlay P2P network.

The response time was roughly constant for random walk and Chord. A null hypothesis test confirmed that the variation could not be attributed to node mobility. Gnutella, on the other hand, had a small increase in the response time for larger node speeds, as Fig. 19 shows.

Meanwhile, the energy consumed by Chord, shown in Fig. 20, reduced with increasing node speeds. This is due to the smaller amount of messages to be forwarded, as they were frequently lost due to path losses. Gnutella and random walk had a smaller variation, though. Due to the resiliency of multiple query messages traveling the network, both protocols suffered less route breaks. Further, the increased mobility required more route updates, which were the most important cause of energy consumption.

Node mobility impacts the amount of losses, as shown in Fig. 21 by the number of packets sent during the simulation. This occurs because more routes are invalid, and hence less messages could be forwarded. This effect is quite pronounced for Gnutella due to its exponential behavior. The number of messages sent considers only P2P application packets, including the forwarding of queries. There is a decrease in the number of packets sent, since nodes out of range or broken routes will reduce the amount of possible forwards. This effect is more pronounced in Gnutella, again due to its exponential forwarding behavior. For example, if a neighbor does not forward a message with TTL equal to three, potentially the message will not be forwarded n^2 times, where n is the average number of neighbors.

5.6. Network dynamics

Next, we evaluate how the network dynamics, i.e., nodes joining and leaving the network, impact the performance of the P2P protocols. The dynamicity of the network requires reconfigurations in the topology, which may affect the efficiency of the protocols. Gnutella is quite simple to be reconfigured, as the nodes must only add or remove neighbors from their list of known peers. Chord, in contrast, requires a lengthy process, as nodes must rebuild their finger tables and migrate the references to their files on the DHT. We varied the percentage of online nodes from 50% to 100%.

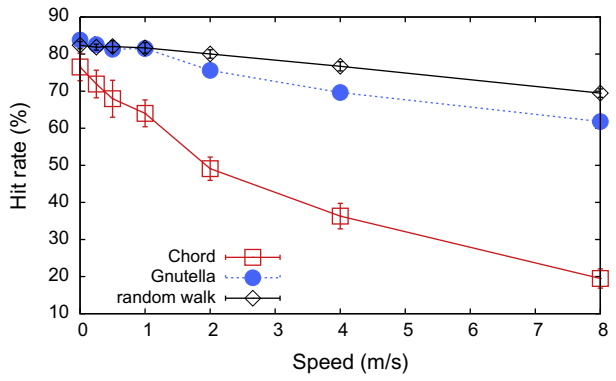

Fig. 18. Scenario 5.5: hit rate.

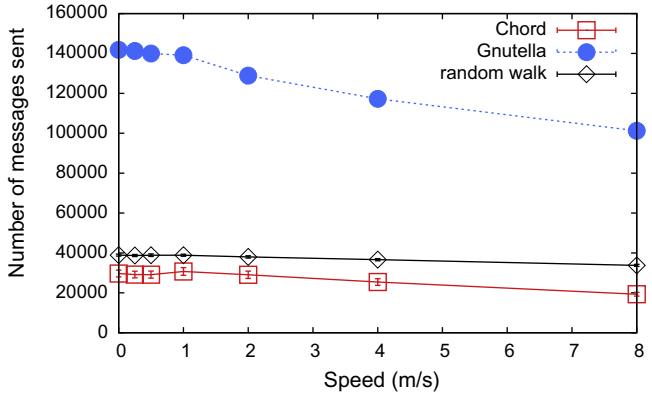

Fig. 21. Scenario 5.5: total number of msgs sent.

Fig. 22 shows the hit rate. Gnutella and random walk presented up to 10% higher hit rates with less node departures. Meanwhile, the hit rate of Chord increased from 50% to nearly 78% when no peers leave the network. Clearly, Chord has a significant performance impact due to node dynamics.

Fig. 23 presents the average response time of the protocols. This figure shows that the amount of node departures/joins does not influence much the response time of the queries for Gnutella and random walk. For Chord, on the other hand, the response time was reduced by 55% for a completely static scenario. We attribute this to the smaller amount of ring maintenance operations that have to be performed (i.e., insertion/removal of files, migration of file responsibilities), which leads to a reduced number of control messages sent over the network.

As expected, the average energy consumption increased with more active peers, since those peers will produce more traffic. Fig. 24 shows that the energy consumption of Chord increased by 50%. A similar effect was produced for Gnutella and random walk. For Chord, though, this result may be counter-intuitive since the amount of control overhead should be reduced. However, we must also consider that much more queries were solved.

Fig. 25 shows the amount of packets dropped in the simulation. The increase was linear, probably due to the addition of new peers in the simulation. A quick look at the response times show that the network is under a light traffic, hence a linear increase in drops instead of an exponential one, which would be characteristic of congestion.

It is worth mentioning that some ad hoc networks (such as the ones employed in rescue situations) will exhibit a significant amount of disconnections due to harsh environmental conditions,

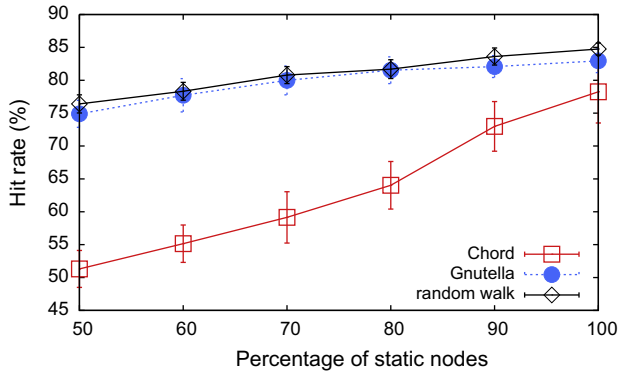

Fig. 22. Scenario 5.6: hit rate.

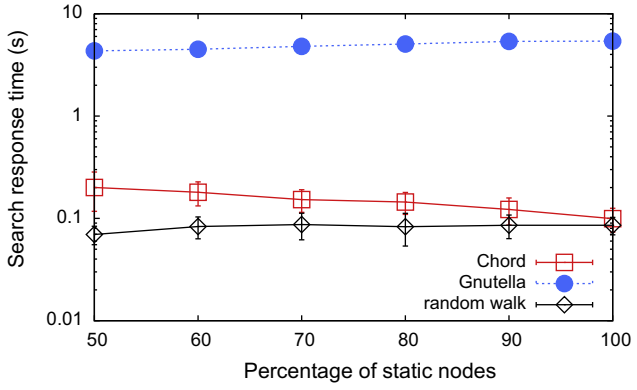

Fig. 23. Scenario 5.6: response time.

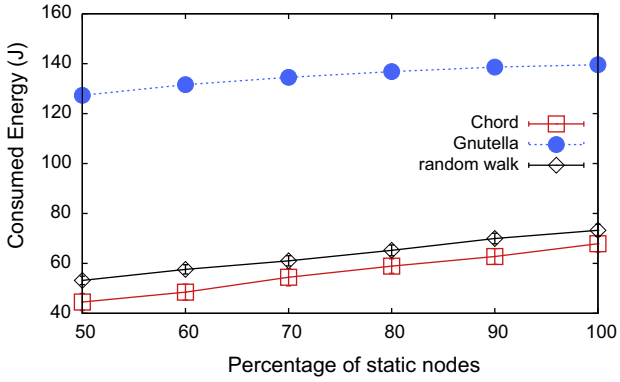

Fig. 24. Scenario 5.6: energy consumed.

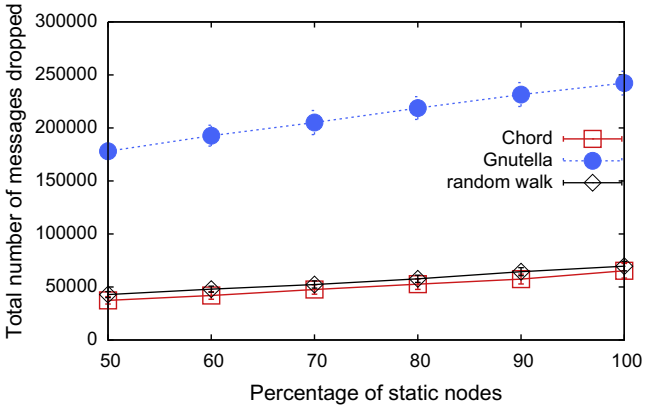

Fig. 25. Scenario 5.6: packets dropped.

but there are also more "well-behaved" ad hoc networks. This scenario shows that Chord is more suitable for less dynamic networks, requiring less energy consumption and yielding lower response times when compared to Gnutella. For applications where disconnections are frequent, conversely, Gnutella is the more robust protocol.

5.7. Discussion

Our study shows that Chord is not suitable for highly dynamic or low reliability environments. However, when the network is reasonably static and interference is low, Chord outperforms Gnutella, yielding higher hit rates with lower energy consumption. This behavior is mainly due to the highly dynamic behavior of ad hoc networks, which influence packet losses. When nodes are static, performance in Chord is mainly bound by the number of nodes always online, that is, the availability of the network.

On the whole, Gnutella presents better hit rates, however at a higher cost. Unstructured protocols are clearly the most adequate for most scenarios, as their "flooding" of queries throughout the network increases resilience. They are also dependent on the number of replicas, having better performance when more replicas are available. Structured protocols, on the other hand, are independent of the number of replicas, thus being more appropriated for querying "rare" content. They are also more suitable for controlled scenarios, where there is little or no mobility and node disconnections are rare. We must note that an implementation of structured protocols using reliable transmission may enhance the performance of such protocols over ad hoc networks, despite the increase in complexity and in the number of messages sent.

In our version of Gnutella and random walk we did not implement any neighbor discovery algorithm, as neighbors were randomly assigned. Our Chord implementation, however, captures all aspects of network formation and maintenance, thus Chord suffers with more overhead in our analysis.

6. Improving P2P over MANETs

This section presents our proposed modifications to the structured and unstructured approaches. We add redundancy on structured networks in order to increase their fault tolerance. For the unstructured approach, we propose the use of selective forwarding to decrease network utilization.

6.1. The structured approach

In the structured approach, the poor performance is due to the frequent loss of query packets. To counter this, we send redundant query messages. The node that originates the query sends more than one message to its neighbors, chosen either randomly or following some rule. One possible implementation is to modify the forwarding strategy of the protocol to choose multiple paths, e.g., defining several possible next hops instead of one. The objective is to have several messages following slightly different paths, thus increasing the probability of success of the queries. We will explain below the implementation of our optimization strategy in general, and then its implementation in Chord.

The proposed strategy can be implemented in any structured protocol. DHTs implement the mapping of a file to an n-dimensional point of a delimited space (In Chord, for example, this value space has one dimension, where values range from $[0, 2^m]$, where m is the length, in bits, of the keys). Each node is responsible for keeping a region on this space, that is, it must manage all the points (keys) that lie within the boundaries of its space. Thus, the forwarding function uses information stored on the current node to define which of its known neighbors are closest to the destination. Hence, in order to implement our modification, the forwarding function must be modified to return more than one node. We describe below how we implemented this enhancement in Chord.

In Chord, peers forward messages using the finger tables by finding their peer which is closest to the key for the requested content. Thus, we implement redundancy by requiring the peer originating the request to send messages to the n closest entries on the finger table, as Algorithm 1 shows. The change is only applied to the node initiating the query, in order to have a fixed number of redundant query messages. In Chord, consecutive entries on the finger table can point to the same peer, thus we skip repeated entries. Peers receiving a query follow the traditional algorithm, sending only one copy of the message.

Algorithm 1. Adding redundancy to Chord queries.

```
1:  procedure query_file(id)
2:    closest = closest_preceding_finger(id);
3:    sent = 0;
4:    last = null;
5:    while sent≠number_redundant_msgs do
6:      if finger[closest].node≠last then
7:        send_query(finger[closest], id);
8:        last = finger[closest].node;
9:        sent = sent + 1;
10:     end if
11:     closest = (closest − 1) mod finger.size;
12:   end while
13: end procedure
```

6.2. The unstructured approach

As mentioned before, the performance of unstructured P2P approaches is limited by the amount of messages sent. To counter this, we borrowed concepts from a classic MANET routing algorithm called Gossiping [38]. In Gossiping, each node forwards its requests to each of its neighbors according to a pre-defined probability, significantly reducing energy consumption and network load [39]. Since this technique requires only to add a forwarding probability to incoming query packets, it can be applied to any unstructured P2P protocol.

To avoid having to empirically define the best forwarding probability for each network, we propose an adaptive mechanism based on network load. In this approach, that we refer to as Gossiping-LB (from load-balancing), the forwarding probability for a given neighbor is calculated as $p \times (1 - u)$, where p is a fixed value, and u $(0 \leqslant u \leqslant 1)$ is the queue utilization of the neighbor. This process is repeated for every node that receives the query. The operation of Gossiping-LB is shown in Algorithm 2. This calculation allows Gossiping-LB to send more messages to neighbors with lower load, while less messages are sent to saturated nodes. The queue utilization is piggy-backed on the PING-PONG messages used to check if the peers are still active. Differently from routing, we propose that nodes "draw the coin" for each neighbor (line 16 of the algorithm), thus our approach sends messages to an arbitrary subset of the neighbors of a peer, avoiding the "all or nothing" behavior of Gossiping in routing.

Algorithm 2. Implementing Gossiping with load-balancing in Gnutella.

```
1:  procedure receive_pong(n)
2:    neighbors_n.utilization = n.utilization;
3:  end procedure
4:  procedure query_file(id)
5:    forward_query(id, initial_ttl);
6:  end procedure
7:  procedure receive_query(n, id, ttl)
8:    if have_file(id)
9:      send_reply();
10:   else if ttl > 1
11:     forward_query(id, ttl − 1);
12:   end if
13: end procedure
14: procedure forward_query(id, ttl)
15:   foreach n in neighbors do
16:     if random() < p × (1 − n.utilization) then
17:     send_query(id, ttl);
18:     end if
19:   done
20: end procedure
```

7. Evaluation of the improvements

This section evaluates the proposed enhancements using the same scenarios described in Section 5. We chose the scenarios where structured and unstructured P2P networks performed worse, as our objective is to improve their performance on the worst case.

7.1. The structured approach

The selected scenarios for the structured approach were node mobility (varying node speed) and network dynamics (varying number of connections and disconnections). From now on, we refer to the original implementation of Chord as *no redundancy*, while

redundancy x indicates a modified Chord that employs *x* query messages.

7.1.1. Node mobility

The hit rate decreased with higher node speeds, as Fig. 26 shows. As expected, the redundant messages increase success rate. The difference for *no redundancy* and *redundancy 2* varies from 2% up to 4% for low mobility. Note that increasing the mobility reduces this difference. This occurs because node speed increases the amount of route breaks, making it difficult to find a valid route for the nodes in the index table. However, redundancy increased the hit rate by up to 2% for node speeds of 4 m/s or more.

The performance gains provided by the redundant messages come at the expense of the response time (Fig. 28) and energy consumption (Fig. 27). The average response time increased around 0.2 s for *redundancy 4* and 0.4 s for *redundancy 8*. This difference increases for lower node speeds. Energy consumption also increased by up to 80% for *redundancy 8*. Thus, those results show that there must be a compromise of energy consumption, delay and hit rate when choosing the amount of redundancy. Fig. 29 shows the amount of messages sent. As expected, less messages could be delivered when mobility increases.

7.1.2. Dynamics

Next, we varied the percentage of dynamic peers, which are the peers that eventually leave the simulation. Fig. 30 shows that the hit ratio increases with more redundant messages by up to 5% for 8 messages. This gain varies from 2% up to 5% for redundancies from two to four. Further, when few nodes leave the network (more than 80% of the nodes are always online), there are no gains on the hit rate. Clearly, the proposed modification is suitable only to dynamic scenarios. Fig. 31 shows that the implementations with redundancy are more sensitive to the number of nodes, which changes the network load, as shown in the low dynamics scenarios. The response time grows quickly with the percentage of static peers for redundant implementations. As the network gets more dynamic, the network load decreases. Figs. 32 and 33 show the average energy consumption and the average number of messages sent per node, respectively. The energy consumed follows the tendency of the number of packets sent, thus the higher the number of packet sent the higher is the energy consumed. Energy consumption can increase by up to 40% if *redundancy 8* is employed.

Thus, the number of redundant messages must be a compromise of the hit rate, energy consumption and response time. From the results above we identify that a low number of redundant messages should be used, since the costs of sending more messages quickly out-weight the gains in the hit rate.

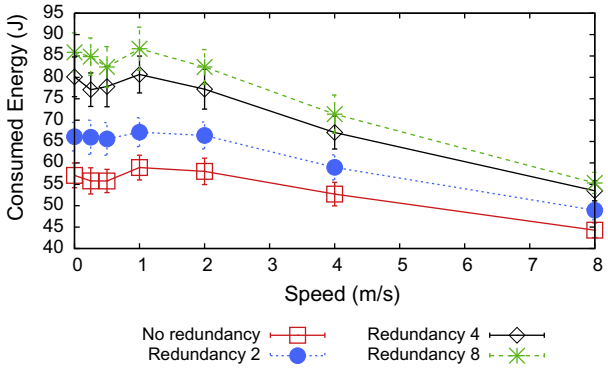

Fig. 28. Energy consumption, Scenario 7.1.1.

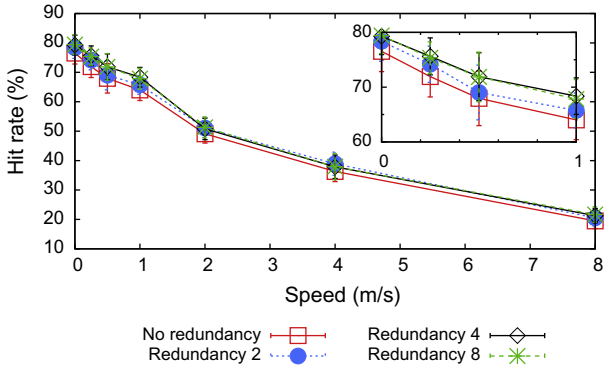

Fig. 26. Hit rate, Scenario 7.1.1.

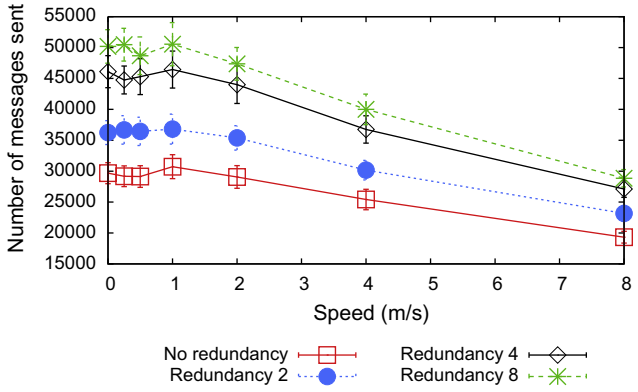

Fig. 29. Number of messages sent, Scenario 7.1.1.

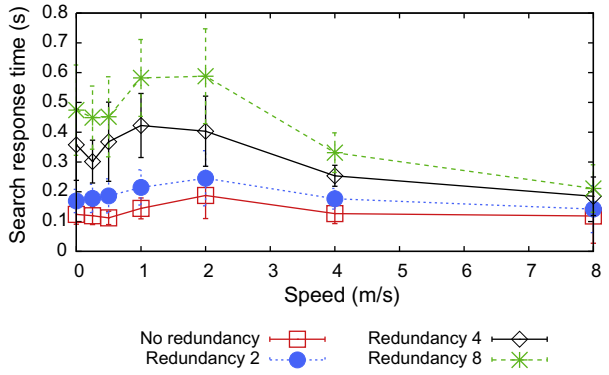

Fig. 27. Response time, Scenario 7.1.1.

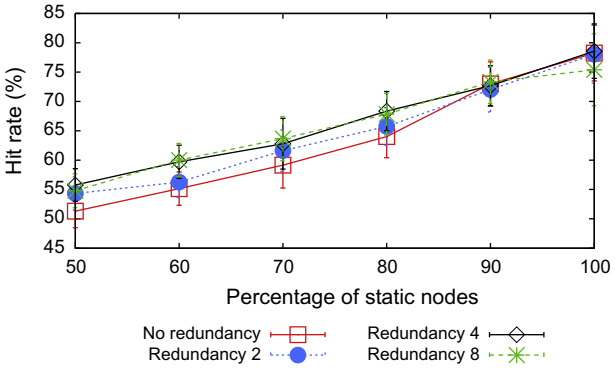

Fig. 30. Hit rate, Scenario 7.1.2.

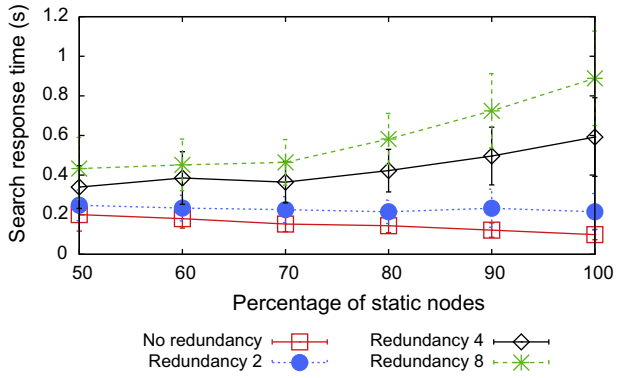

Fig. 31. Response time, Scenario 7.1.2.

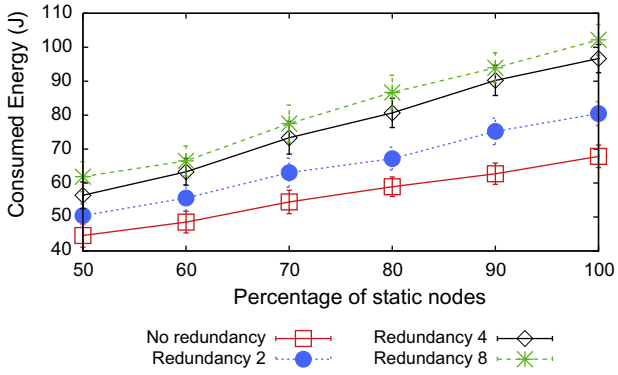

Fig. 32. Energy consumption, Scenario 7.1.2.

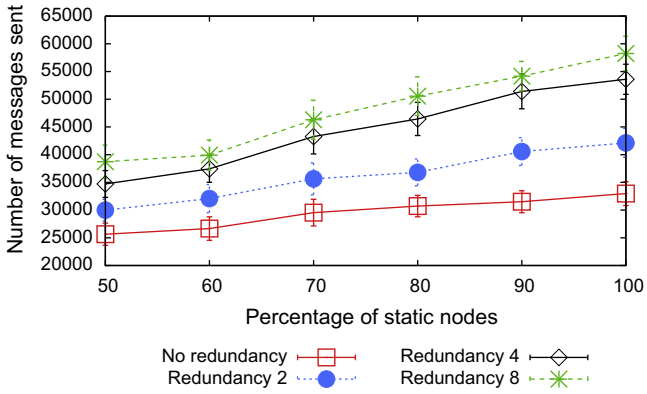

Fig. 33. Number of messages sent, Scenario 7.1.2.

7.2. The unstructured approach

For the structured approach, we identified that its worst performance occurs when network load is high, that is, when there is a big number of peers or when the peers make a high number of requests. The results are described below. In this section we refer to *Gossiping-LB* as the Gossiping with load-balancing strategy, and *Gossiping* as the selective forwarding strategy. Parenthesis after the name of the protocol indicate the forwarding probability for each neighbor and, when omitted, the forwarding probability is 1.0.

7.2.1. Number of nodes

This scenario evaluates the performance of the protocols when varying the number of nodes on the network, while keeping node density constant. Fig. 34 shows the query hit rate. As we have seen in Section 5.2, the network is already saturated for Gnutella when there are 50 nodes. Thus, all the proposed modifications outperform Gnutella since they reduce the amount of packets sent. For 50 and 60 nodes, Gossiping(0.7) has the best performance, outperforming Gnutella from 10% up to 15%. For more than 60 nodes, however, Gossiping(0.5) performs best.

The response time, shown in Fig. 35, follows the expected behavior of systems with finite queues. For example, for Gossiping(0.5), the response time is low for networks having 50 and 60 nodes, as the network is able to timely process all the packets (its queue occupation is low). For networks of more than 70 nodes, the response time grows exponentially, indicating that the system is coming to its maximum capacity and queues are building up. Finally, the response time stabilizes, indicating that the queues are already full and excess packets are dropped. All the other evaluated protocols presented a similar curve, however their saturation has already been reached (here we identify saturation as a response time superior to one second).

In stochastic systems, the optimum occurs before the response time starts to increase exponentially. In our simulations, this point coincides in all protocols with the point where the hit rate starts to decline. Further, we identified the "optimum" operation point for each proposed approach: 50 nodes for Gossiping(0.9) and Gossiping(0.8)-LB, 60 nodes for Gossiping(0.7) and 70 nodes for Gossiping(0.5). Due to the results above, we decided to stop our simulations at 100 nodes, since larger networks would have a hit

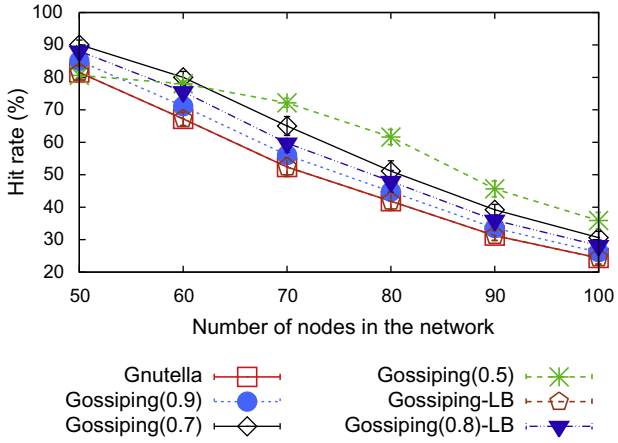

Fig. 34. Hit rate, Scenario 7.2.1.

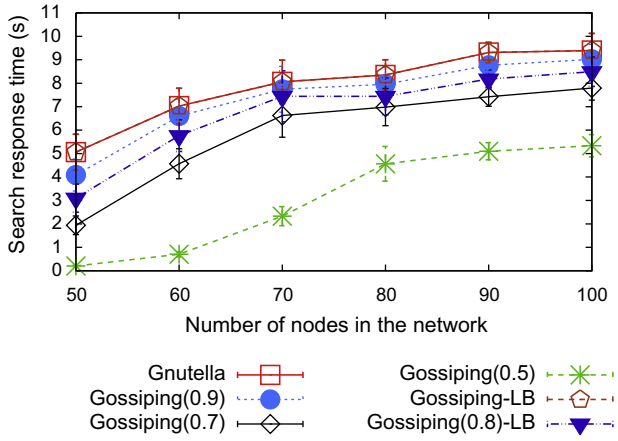

Fig. 35. Response time, Scenario 7.2.1.

rate near to zero. Moreover, the proposed protocols outperform Gnutella only after its saturation point, as in those situations the smaller amount of messages sent by the proposed approaches mitigates collisions and packet drops due to full queues.

Since the proposed approaches send less packets, energy consumption decreases, as shown in Fig. 36. For 60 nodes, for example, Gossiping(0.8)-LB presents a consumption 8% inferior to Gnutella, while Gossiping(0.7) consumes 22% less. Meanwhile, for smaller networks, the added redundancy increases the hit rate. The effect of contention can be seen in the amount of messages dropped, shown in Fig. 37. The amount of packets dropped increases with the forwarding probability employed: Gossiping(0.5) has the smallest amount of drops, while Gnutella has the highest amount. Indirectly, the amount of message drops determines the hit rate and the response time, since drops will incur in retransmissions or query losses.

7.2.2. Network load

In this scenario, we fixed the number of queries in the simulation, as described in Section 5.1. The hit rate varied up to 20% from one protocol to the other, clearly showing that the forwarding probability impacts the performance of the protocols. The hit rate is shown in Fig. 38. The forwarding probability plays a decisive role in the performance improvements, since lower probabilities tend to provide higher gains, however too low probabilities may provide a smaller hit rate (as exemplified with a 50% forwarding probability).

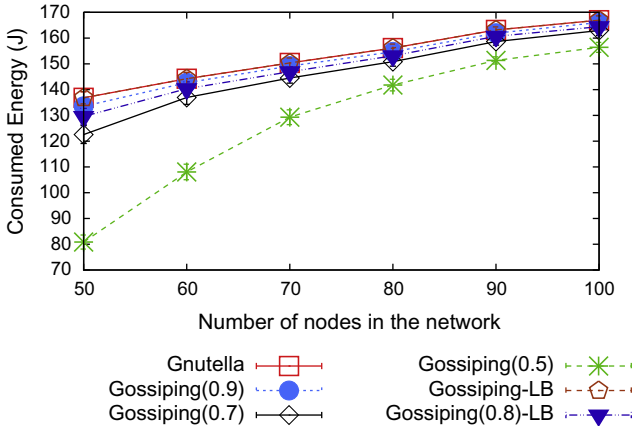

Fig. 36. Energy consumption, Scenario 7.2.1.

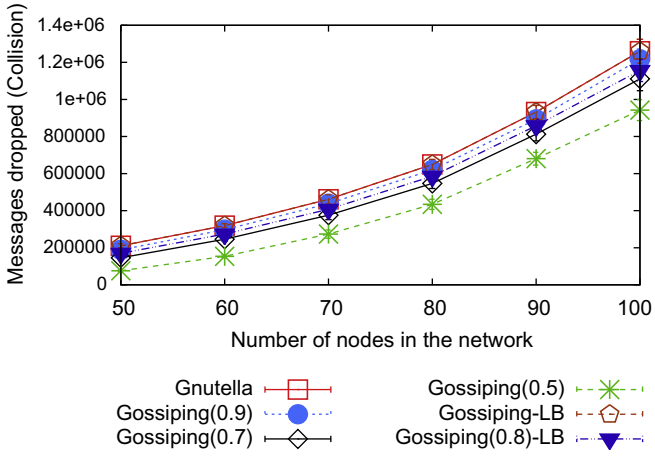

Fig. 37. Number of collisions, Scenario 7.2.1.

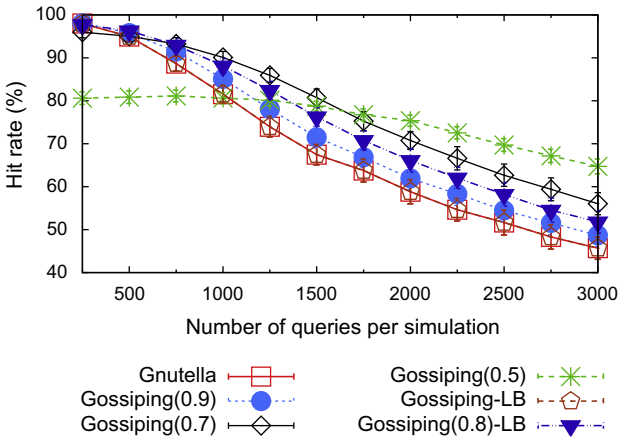

Fig. 38. Hit rate, Scenario 7.2.2.

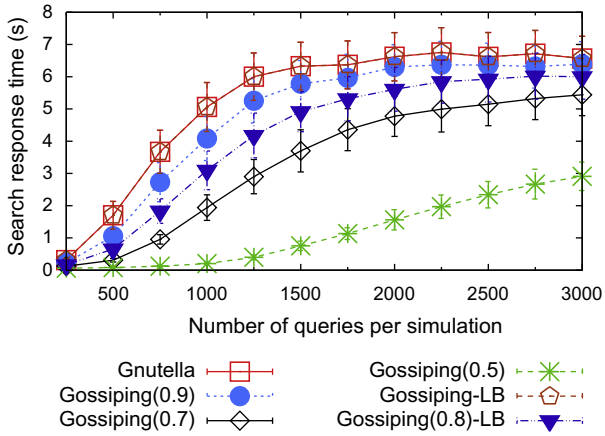

Fig. 39. Response time, Scenario 7.2.2.

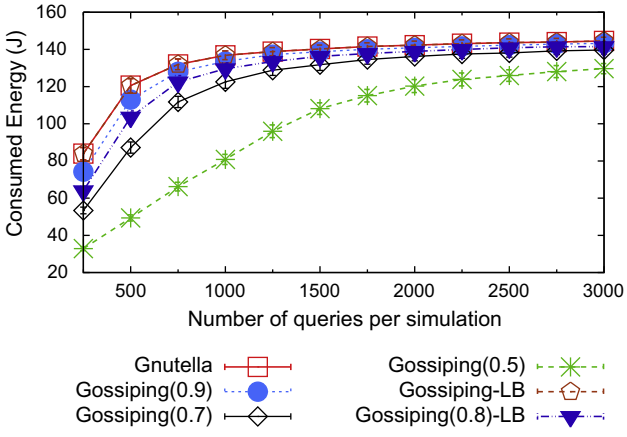

Fig. 40. Energy consumption, Scenario 7.2.2.

There were also performance gains on the average response time and on energy consumption, as shown in Figs. 39 and 40, respectively. Both curves present a negative increase in their slope, and have a tendency of settling at a fixed value. Again, this is a classic indicator of the network reaching its maximum capacity. As expected, lower forwarding probabilities will provide lower response times and energy consumption. However, the gains in energy consumption are much less pronounced than those for response time.

This occurs because the consumption for transmission and reception modes of the radio are quite similar.

The number of messages sent, presented in Fig. 41, shows the sign of contention building up. Although the number of queries increased linearly, the actual increase in the number of messages sent tends to get smaller and smaller for higher query rates. This is due to message drops, which are more frequent in higher loads. Indeed, a reduction in the forwarding probability delays the point where the network reaches its maximum contention point.

7.2.3. Number of file replicas

Next, we varied the effect of file replicas on the proposed solutions. Fig. 42 shows the hit rate. Again, the Gossiping solution with 50% forwarding probability had a poor performance, however it manages to equal that of the original Gnutella implementation. Meanwhile, lower forwarding probabilities increased the hit rate. This is due to the characteristics of the scenario, the same of Section 5.3, where the response time of Gnutella was well over one second, showing a high level of contention.

The number of packets sent is shown in Fig. 43. The number of messages sent increases with the forwarding probability. Again, since a smaller number of peers must be traversed to find a given file, the number of packets sent decreased by up to 12% for the classic Gnutella and up to 17% for Gnutella(0.5).

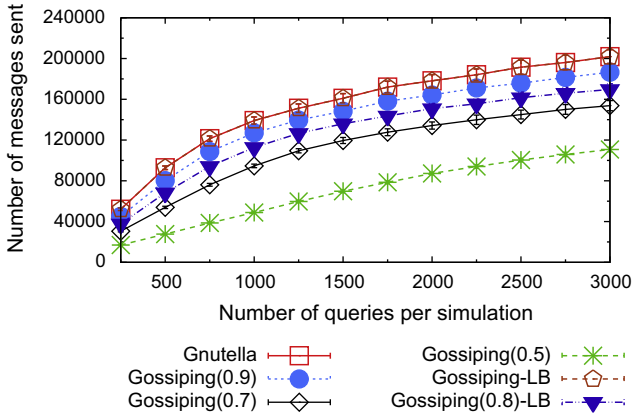

Fig. 41. Number of messages sent, Scenario 7.2.2.

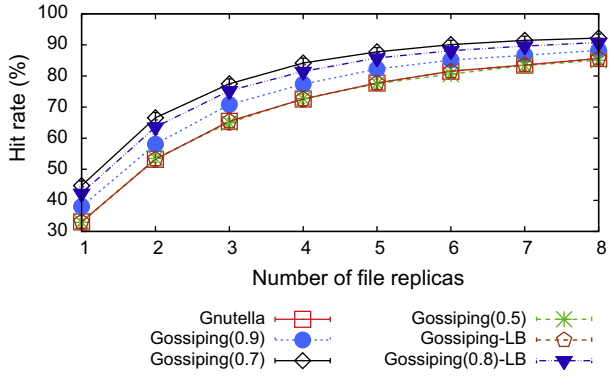

Fig. 42. Hit rate, Scenario 7.2.3.

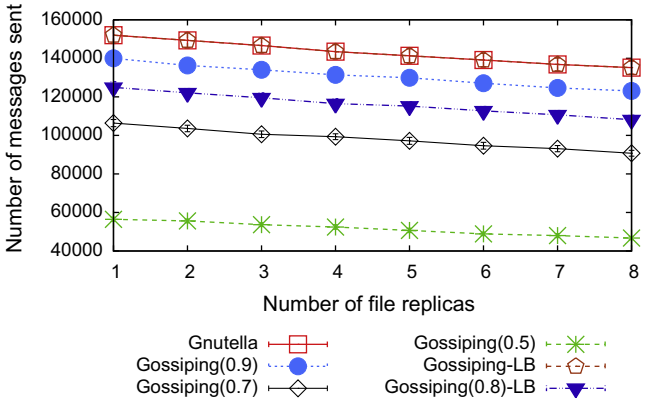

Fig. 43. Number of packets sent, Scenario 7.2.3.

8. Conclusion

Peer-to-peer networks are a perfect fit for content dissemination on MANETs due to their robustness provided by the delegation of tasks to several nodes. P2P networks are divided into two types according to the employed content distribution technique. Unstructured networks use controlled flooding, while structured networks use distributed indexes. Further, most P2P systems were designed to operate on wired networks, thus they perform poorly on MANETs.

This article evaluated the performance of structured and unstructured content discovery techniques over MANETs and proposed two enhancements to such techniques. By using simulations, our study was able to consider several characteristics of the wireless medium (collisions, interference, dynamic topologies) that usually are not taken into account by analytical models. Based on the results obtained from this study, we identified structural weaknesses in the content discovery techniques and proposed enhancements to counter those limitations. In unstructured networks, the amount of query messages quickly overloads the network, decreasing the performance of the P2P network. We proposed the use of selective query forwarding to reduce the load. The query forwarding probability is dynamically adapted according to network load, measured by the occupation of the network device. Structured networks, on the other hand, send only one message per query, which is frequently dropped due to collisions. To counter that, we introduced redundant query messages.

As future work, we will study other approaches to improve the performance of P2P networks over MANETs. One promising approach is caching the most frequently requested items. This would reduce the response time and the number of messages in the network by resolving the queries in less hops. We can further improve structured networks by using load-balancing to dynamically determine the amount of redundant messages in Chord.

To allow a fair comparison, the present study employed protocol implementations that are not optimized to MANETs. Thus, another future work is to repeat this evaluation of P2P protocols where the P2P topology is closer or similar to the physical topology. In such implementations the protocols could employ the relatively low cost of broadcast messages to improve the performance of P2P applications.

Acknowledgements

We would like to thank the following institutions for funding our research: CNPq, a funding agency from the Science and Technology Ministry of Brazil; CAPES, a funding agency from the Education Ministry of Brazil; FAPEMIG, a funding agency from the state of Minas Gerais, Brazil.

References

[1] J. Borg, A comparative study of ad hoc & peer to peer networks, Master's thesis, University College London, 2003.

[2] Z.J. Haas, J. Deng, B. Liang, P. Papadimitratos, S. Sajama, Wireless ad hoc networks, in: Wiley Encyclopedia of Telecommunications, 2002.
[3] X. Li, J. Wu, Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, Auerbach Publications, 2005 (Ch. Searching Techniques in Peer-to-Peer networks).
[4] S. Androutsellis-Theotokis, D. Spinellis, A survey of peer-to-peer content distribution technologies, ACM Computing Surveys 36 (4) (2004) 335–371. doi:http://doi.acm.org/10.1145/1041680.1041681.
[5] D. Talia, P. Trunfio, Toward a synergy between P2P and grids, IEEE Internet Computing 7 (4) (2003) 94–95.
[6] R. Schollmeier, I. Gruber, M. Finkenzeller, Routing in peer-to-peer and mobile ad hoc networks: a comparison, in: International Workshop on Peer-to-Peer Computing, 2002, pp. 172–186.
[7] L.B. Oliveira, I.G. Siqueira, A.A.F. Loureiro, On the performance of ad hoc routing protocols under a peer-to-peer application, Journal of Parallel and Distributed Computing (JPDC) 65 (11) (2005) 1337–1347.
[8] F.P. Franciscani, M.A. Vasconcelos, R.P. Couto, A.A.F. Loureiro, (re)configuration algorithms for peer-to-peer over ad hoc networks, Journal of Parallel and Distributed Computing 65 (2) (2005) 234–245, doi:http://dx.doi.org/10.1016/j.jpdc.2004.09.007.
[9] Y.C. Hu, S.M. Das, H. Pucha, Exploiting the synergy between peer-to-peer and mobile ad hoc networks, in: HotOS-IX: Ninth Workshop on Hot Topics in Operating Systems, 2003, pp. 37–42.
[10] G. Ding, B. Bhargava, Peer-to-peer file-sharing over mobile ad hoc networks, in: 2th IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp. 104–108.
[11] A. Klemm, C. Lindemann, O.P. Waldhorst, A special-purpose peer-to-peer file sharing system for mobile ad hoc networks, in: IEEE Semiannual Vehicular Technology Conference (VTC2003-Fall), 2003, pp. 2758–2763.
[12] Z. Ge, D.R. Figueiredo, S. Jaiswal, J. Kurose, D. Towsley, Modeling peer–peer file sharing systems, in: Proceedings of the 22nd IEEE Conference on Computer Communications, 2003, pp. 2188–2198.
[13] D. Tsoumakos, N. Roussopoulos, Analysis and comparison of p2p search methods, in: InfoScale'06: Proceedings of the 1st International Conference on Scalable Information Systems, ACM, New York, NY, USA, 2006, p. 25. doi:http://doi.acm.org/10.1145/1146847.1146872.
[14] B. Yang, H. Garcia-Molina, Improving search in peer-to-peer networks, in: 22nd International Conference on Distributed Computing Systems, 2002, pp. 5–14.
[15] L.B. Oliveira, I. Siqueira, D.F. Macedo, A.A. Loureiro, J.M. Nogueira, Evaluation of peer-to-peer network content discovery techniques over mobile ad hoc networks, in: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2005, pp. 51–56.
[16] D.N. da Hora, D.F. Macedo, J.M.S. Nogueira, G. Pujolle, Optimizing peer-to-peer content discovery over wireless mobile ad hoc networks, in: The Ninth IFIP/IEEE International Conference on Mobile and Wireless Communications Networks (MWCN), 2007, pp. 6–10.
[17] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, Search and replication in unstructured peer-to-peer networks, in: ICS'02: Proceedings of the 16th International Conference on Supercomputing, ACM Press, New York, NY, USA, 2002, pp. 84–95. doi:http://doi.acm.org/10.1145/514191.514206.
[18] S.D. Servetto, G. Barrenechea, Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks, in: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, 2002, pp. 12–21. doi:http://doi.acm.org/10.1145/570738.570741.
[19] C.L. Barrett, S.J. Eidenbenz, L. Kroc, M. Marathe, J.P. Smith, Parametric probabilistic sensor network routing, in: Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, 2003, pp. 122–131. doi:http://doi.acm.org/10.1145/941350.941368.
[20] C. Gkantsidis, M. Mihail, A. Saberi, Random walks in peer-to-peer networks, in: Proceedings of the IEEE Infocom, 2004, p. 130.
[21] D. Tsoumakos, N. Roussopoulos, Evaluation of ad hoc routing protocols under a peer-to-peer application, in: Third International Conference on Peer-to-Peer Computing (P2P), 2003, pp. 102–109.
[22] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: Charles E. Perkins (Ed.), Ad Hoc Networking, Addison-Wesley, 2001, pp. 139–172.
[23] A. Rowstron, P. Druschel, Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems, in: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001, pp. 329–350.
[24] M. Gerla, C. Lindemann, A. Rowstron, P2p manet's – new research issues, in: M. Gerla, C. Lindemann, A. Rowstron (Eds.), Perspectives Workshop: Peer-to-Peer Mobile Ad Hoc Networks – New Research Issues, no. 05152 in Dagstuhl Seminar Proceedings, 2005.
[25] S. Srinivasan, A. Moghadam, S.G. Hong, H. Schulzrinne, 7ds – node cooperation and information exchange in mostly disconnected networks, in: IEEE International Conference on Communications (ICC), 2007, pp. 3921–3927.
[26] R. Schollmeier, I. Gruber, F. Niethammer, Protocol for peer-to-peer networking in mobile environments, in: 8th International Conference on Computer Communications and Networks, 2003, pp. 121–127.
[27] U. Lee, J.-S. Park, J. Yeh, G. Pau, M. Gerla, Code torrent: content distribution using network coding in vanet, in: MobiShare'06: Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking, ACM, 2006, pp. 1–5.
[28] M. Mecella, M. Angelaccio, A. Krek, T. Catarci, B. Buttarazzi, S. Dustdar, Workpad: an adaptive peer-to-peer software infrastructure for supporting collaborative work of human operators in emergency/disaster scenarios, in: CTS'06: Proceedings of the International Symposium on Collaborative Technologies and Systems, IEEE Computer Society, Washington, DC, USA, 2006, pp. 173–180. doi: http://dx.doi.org/10.1109/CTS.2006.72.
[29] A. Boukerche, A. Zarrad, R. Araújo, Smart Gnutella overlay formation for collaborative virtual environments over mobile ad-hoc networks, in: Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT), 2006, pp. 143–156.
[30] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Transactions on Networking 11 (1) (2003) 17–32.
[31] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, R. Sivakumar, ATP: a reliable transport protocol for ad hoc networks, IEEE Transactions on Mobile Computing 4 (6) (2005) 588–603.
[32] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, in: Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), 2002, pp. 89–102. doi:http://doi.acm.org/10.1145/633025.633035.
[33] C.E. Perkins, E.M. Royer, Ad hoc on-demand distance vector routing, in: 2nd IEEE Workshop on Mobile Computing System and Applications, 1999, pp. 90–100.
[34] Cisco aironet 350 series client adapters. Available from: <http://www.cisco.com/en/US/prod/collateral/wireless/ps6442/ ps4555/ps5818/product_data_sheet09186a00801ebc29.html/> (March 2008).
[35] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1998, pp. 85–97.
[36] G. Gaertner, V. Cahill, Understanding link quality in 802.11 mobile ad hoc networks, IEEE Internet Computing 8 (1) (2004) 55–60.
[37] D.S.J. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, 2003, pp. 134–146.
[38] S.M. Hedetniemi, T.S. Hedetniemi, A.L. Liestman, A survey of gossiping and broadcasting in communication networks, Networks 18 (1988) 319–349.
[39] W.R. Heinzelman, J. Kulik, H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks, in: Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1999, pp. 174–185. doi:http://doi.acm.org/10.1145/313451.313529.