Distrib Parallel Databases (2009) 26: 181–205
DOI 10.1007/s10619-009-7046-7

Rewiring strategies for semantic overlay networks

Paraskevi Raftopoulou · Euripides G.M. Petrakis · Christos Tryfonopoulos

Published online: 9 July 2009

Abstract Semantic overlay networks cluster peers that are semantically, thematically or socially close into groups, by means of a rewiring procedure that is periodically executed by each peer. This procedure establishes new connections to similar peers and disregards connections to peers that are dissimilar. Retrieval effectiveness is then improved by exploiting this information at query time (as queries may address clusters of similar peers). Although all systems based on semantic overlay networks apply some rewiring technique, there is no comprehensive study showing the effect of rewiring on system's performance. In this work, a framework for studying the attribution of rewiring strategies in semantic overlay networks is proposed. A generic approach to rewiring is presented and several variants of this approach are reviewed and evaluated. We show how peer organisation is affected by the different design choices of the rewiring mechanism and how these choices affect the performance of the system overall (both in terms of communication overhead and retrieval effectiveness). Our experimental evaluation with real-word data and queries confirms the dependence between rewiring strategies and retrieval performance, and gives insights on the trade-offs involved in the selection of a rewiring strategy.

P. Raftopoulou (✉) · E.G.M. Petrakis
Technical University of Crete, University Campus, 73100 Chania, Greece
e-mail: paraskevi@intelligence.tuc.gr

E.G.M. Petrakis
e-mail: petrakis@intelligence.tuc.gr

C. Tryfonopoulos
University of Peloponnese, University Campus, 22100 Tripoli, Greece
e-mail: trifon@uop.gr

C. Tryfonopoulos
Max-Plank Institute for Informatics, 66123 Saarbruecken, Germany

Keywords Peer-to-peer networks · Semantic overlay networks · Rewiring strategies · Information retrieval

1 Introduction

Peer-to-peer (P2P) systems offer the potential for low-cost sharing of information, while ensuring autonomy and privacy of the participating entities. The main idea behind P2P is that, instead of relying on central components, functionality is provided through decentralised overlay architectures. In overlay networks, peers typically connect to a small set of other peers. Queries are then, propagated to the network searching for information qualifying query criteria by utilising existing connections and following a predetermined query forwarding strategy. The popularity of earlier systems like Gnutella[1] and Freenet,[2] built upon the idea of unstructured overlay networks, has propelled research in this direction while lately, the proliferation of social networking has added yet another interesting dimension to the problem of searching for content.

In Semantic Overlay Networks (SONs), peers that are semantically, thematically or socially similar are *organised* into groups. SONs, while being highly flexible, improve query performance and guarantee high degree of peer autonomy [2, 12, 20, 23]. This technology has proven useful not only for information sharing in distributed environments, but also as a natural distributed alternative to Web 2.0 application domains, such as decentralised social networking in the spirit of Flickr[3] or del.icio.us.[4] Contrary to structured overlays that focus on providing accurate location mechanisms (e.g., [21, 26]), SONs are better suited for loose P2P architectures, which assume neither a specific network structure nor total control over the location of the data. Additionally, SONs offer better support of semantics due to their ability to provide mechanisms for approximate, range, or text queries, and emphasise peer autonomy.

Peer organisation in a SON is achieved through a *rewiring protocol* that is (periodically) executed by each peer. The purpose of this protocol is to establish connections among *similar peers*. This is achieved by creating new connections to similar peers and by discarding connections that are outdated or pointing to dissimilar peers. The goal of a rewiring protocol is to create *clusters* of peers with similar interests. Queries can then be resolved by routing the query towards clusters based on their likelihood to match the query.

The management of large volumes of data in P2P networks has generated additional interest in methods for effective network organisation based on peer contents and consequently, in methods supporting information retrieval (IR) over SONs (e.g., [8, 9]). Most of these research proposals assume a standard rewiring procedure for creating SONs and search the network by exploiting certain architectural [8, 27, 28] or modelling [1, 24, 25] aspects of this organisation. The effectiveness of IR methods, supporting content-based retrievals over SONs, depends on the type of content stored

[1] http://www.gnu.org/.

[2] http://freenetproject.org/.

[3] http://www.flickr.com/.

[4] http://www.del.icio.us.com/.

by the peers, on the type of queries allowed and most importantly, on the rewiring procedure applied and on the efficiency of the query forwarding procedure implemented.

The present work goes one step further and suggests that the retrieval performance depends on the rewiring strategy applied. To show proof of concept, we adopt a generic P2P architecture that utilises widely adopted techniques from SONs, such as characterisation of peers by content, peer clustering and link rewiring. In our SON architecture, peers are characterised by documents in the spirit of [8, 10, 13, 14, 20, 23, 25]. Each peer in the network maintains links to other peers with similar content to form clusters. This is achieved through a rewiring protocol which runs independently on each peer as in [4, 12, 14, 20, 23, 25]. The methods referred to above use the peer organisation, emerging from the execution of the rewiring protocol, to efficiently route queries to the clusters of peers that are more likely to answer the query. In our setting, a number of choices in designing a rewiring strategy are considered and all are implemented and evaluated (in terms of network load and retrieval efficiency) using real-world data and queries.

This work aims at providing a better understanding on the functional issues related to the design and performance of any rewiring protocol by answering the following fundamental questions: (i) what is an appropriate forwarding strategy for rewiring messages within a SON, (ii) what information should be used to update peer connections, (iii) should peer connections be bi-directional or just one-directional, (iv) how coherent and well-connected should a cluster of similar peers be? Based on insight acquired by answering these questions, we propose optimisations to the rewiring protocol. The optimal rewiring strategy is designed based on experimental results and on the analysis of these results.

The remainder of the paper is organised as follows. Section 2 presents proposals that implement SON-like structures to support IR functionality, while Sect. 3 discusses a generic SON architecture and the related protocols. Section 4 presents the proposed rewiring strategies that aim at better peer organisation. The experimental evaluation of the strategies is presented in Sect. 5, followed by conclusions and issues for further research in Sect. 6.

2 Related work

Issues related to effective network organisation and to IR approaches implementing SON-like structures have been previously raised by many researchers. In the following, we present those focusing on architectural issues (i.e., overlay topology, along with the corresponding protocols) and those exploiting certain modelling aspects (e.g., models describing peer similarity).

Dealing with architectural issues, Merugu et al. [14] show that constructing small-world-like overlay topologies improves retrieval performance. Loser et al. [12] introduce the concept of semantic overlay clusters for super-peer networks. This work aims at clustering peers that store complex heterogeneous schemes by using a super-peer architecture. In [13], Lu et al. propose a two-tier architecture, according to which a peer provides content-based information about neighbouring peers and determines how to route queries in the network. Along the same lines, Klampanos et al. [8] propose an architecture for IR-based clustering of peers, where a representative peer (hub) maintains information about all other hubs and is responsible for query routing. Loser et al. [11] propose a three-layer organisation of peers (based both on peer content and usefulness estimators) and suggest combining information from all layers for routing queries. In a similar spirit, Hidayanto et al. [4] propose self-organising peers into communities based on their content (expert network), the issued queries (interest groups), or both the content and the queries (hybrid groups).

Emphasising on protocols, Voulgaris et al. [29] propose an epidemic protocol[5] that implicitly clusters peers with similar content. In [18], Penzo et al. propose a strategy for incremental clustering of semantically related peers. Along the same lines, Schmitz [23] assumes that peers share concepts from a common ontology and proposes strategies for organising peers into communities with similar concepts. iCLUSTER [20] extends these protocols by allowing peers with multiple interests to form clusters. Jelasity and Montresor [6] observe that aggregation (i.e., global network information as average load) may reveal useful information while self-organising in large-scale networks, and present epidemic protocols capable of exploiting this information.

Focusing on modelling issues, Schmitz and Loser [24] present a rewiring model along with evaluation metrics commonly used in SONs. In [25], Spripanidkulchai et al. introduce the notion of peer clustering based on similar interests rather than similar content. According to this work, peers are organised on the top of the existing Gnutella network to improve retrieval performance. In a similar spirit, Li et al. [10] propose creating a self-organising network based on the semantics of data objects stored locally to peers. Aberer at al. [1] introduce a decentralised process that, relying on pair-wise local interactions, incrementally develops global agreement and obtains semantic interoperability among data sources. Parreira et al. [17] introduce the notion of “peer-to-peer dating” that allows peers to decide which connections to create and which to avoid based on various usefulness estimators. Koloniari et al. [9] model peer clustering as a game, where peers try to maximise the recall for their local query workload by joining the appropriate clusters.

All the works referred to above aim at providing architectural or modelling solutions to support IR over SONs. In this work, rewiring is treated as an important system component that needs to be further investigated and optimised. We adopt a simple and general rewiring model and identify possible rewiring alternatives, the combination of which make up a range of possible rewiring strategies worth exploring. We compare rewiring alternatives in terms of efficiency and retrieval effectiveness and show the benefit resulting from optimising the rewiring mechanism.

3 Overview

SON-based systems apply a periodic rewiring protocol to cluster peers with similar interests [10, 29] and a fireworks-like technique [16, 27] is used to route queries

[5]Epidemic protocols exploit randomness to disseminate information across peers.

within the network. Peer interests may vary from descriptions of document collections [13, 20], to topics in a hierarchy [8], schema synopses [7], or ontology concepts [23]. In i*Cluster* [20], the interests of a peer are identified by using its local document collection, and a single peer has a tunable and dynamic number of interests depending on its capabilities, collection size and content diversity. i*Cluster* was the first system to overcome the specialisation assumption [15] (i.e., each peer has only one interest) common in SONs.

3.1 Architecture

We consider a P2P network, where peers are responsible for serving both users searching for information and users contributing information to the network. The peers run the rewiring protocol and form clusters based on their likelihood to contain similar content. Each peer is characterised by its information content, i.e. its document collection, which may be either automatically (by text analysis) or manually assigned to each document (e.g., tags or index terms). To identify its interests, a peer categorises its documents using an external reference system, i.e. an ontology as in [23] or a taxonomy such as the ACM categorisation system, or by clustering [3]. A peer may be assigned more than one interests. Each peer maintains a *routing index* holding information for *short-range* and *long-range* links to other peers. Short-range links correspond to *intra-cluster* information (i.e., links to peers with similar interests), while long-range links correspond to *inter-cluster* information (i.e., links to peers having different interests).

Our architecture is general enough to cover fundamental design aspects of existing semantic overlays: as typically assumed in the literature peers are categorised by their content [8, 10, 13, 14, 20, 23, 25] and links to other peers involve not only similar peers, but also dissimilar ones to support the connectivity between different clusters as in [14, 20, 23, 25]. The rewiring procedure is executed locally by each peer and aims at clustering peers with similar content, while queries are forwarded to peer clusters that are similar to the issued query.

3.2 Basic protocols

The main idea behind SONs is to let peers self-organise into clusters with semantically similar content. Then, query execution is performed by identifying the cluster of peers with similar content and by addressing a peer within this cluster. In this section, we present the basic protocols that specify how peers join a SON and self-organise into clusters and how queries are processed.

3.2.1 Joining protocol

When a peer p_i connects to the network, it has to follow the join protocol. Initially, p_i categorises its documents into one or more categories. Consequently, p_i may have more than one interests $I_{i\kappa}$, with κ denoting the number of p_i's interests, stored in its *interest list* $I(p_i)$. A peer interest is represented by the corresponding index terms (if an external reference system has been used) or by a centroid vector (if documents have been categorised by clustering).

For each distinct interest I_{in}, peer p_i maintains a separate routing index RI_{in}, which contains short-range and long-range links. Entries in the routing index are of the form $(ip(p_j), I_{jk})$, where $ip(p_j)$ is the IP address of peer p_j and I_{jk} is the k-th interest of p_j. The amount of memory required to store the necessary information in each peer is a few hundred bytes per routing index entry: a typical entry occupies 4 bytes for the IPv4 address of a node and 2 bytes for the port number, in addition to the description of a peer interest (i.e., a vector of content terms). The number of routing indexes maintained by a peer equals the number of its interests. Peers may merge or split their routing indexes by merging or splitting their interests reliant to changes in their content. A routing index is initialised (when a peer is joining the network, or upon the bootstrapping) as follows: peer p_i collects in RI_{in} the IP addresses of $s+l$ randomly selected peers. These links will be refined according to the interest I_{in} of p_i using the rewiring protocol described in the next section. The computational cost to update the index is O$(s+l)$, where s is the number of short-range links and l is the number of long-range links.

In the following, for simplicity of the presentation, we assume that each peer p_i has only one interest I_i. The same discussion applies for multiple interests, since peers maintain one routing index per interest and the rewiring strategy is applied independently for each peer interest.

3.2.2 *Rewiring protocol*

Peer organisation proceeds by establishing new connections (to similar peers) and by discarding old ones. Each peer p_i periodically (e.g., when its interests have changed) initiates a rewiring procedure (independently for each interest) by computing the intra-cluster similarity (or *neighbourhood similarity*)

$$NS_i = \frac{1}{s} \cdot \sum_{\forall p_j \in RI_i} sim(I_i, I_j), \tag{1}$$

where s is the number of short-range links of p_i (according to interest I_i), I_j is the interest of peer p_j, p_j is a peer contained in RI_i, and $sim()$ can be any appropriate similarity function. The neighbourhood similarity NS_i is used here as a measure of cluster cohesion. If NS_i is greater than a threshold θ, then p_i does not need to take any further action since it is surrounded by similar peers. Otherwise, p_i issues a FINDPEERS$(ip(p_i), I_i, L, \tau_R)$ message, where L is a list and τ_R is the time-to-live (TTL) of the message. List L is initially empty and will be used to store tuples of the form $\langle ip(p_j), I_j\rangle$, containing the IP address and interests of peers discovered while the message traverses the network. System parameters θ and τ_R need to be known upon bootstrapping.

A peer p_j receiving the FINDPEERS() message appends its IP address $ip(p_j)$ and its interest I_j to L (or the interest most similar to I_i if p_j has multiple interests), reduces τ_R by one and forwards the message to the m neighbour peers ($m \leq s$) with interests most similar to I_i. This message forwarding technique is referred to in the literature as *gradient walk* (GW) [22, 23]. When $\tau_R = 0$, the FINDPEERS() message is sent back to the message initiator p_i by using the contact information contained in the message. Figure 1 illustrates the above rewiring procedure in algorithmic steps.

```
Procedure Rewiring($p_i, I_i, \tau_R, \theta, m$)
Initiated by $p_i$ when neighborhood similarity $NS_i$ drops below $\theta$.

input: peer $p_i$ with interest $I_i$ and routing index $RI_i$
output: updated routing index $RI_i$

1:  compute $NS_i = \frac{1}{s} \cdot \sum_{\forall p_j \in RI_i} sim(I_i, I_j)$
2:  if $NS_i < \theta$ then
3:    $L \leftarrow \{\ \}$
4:    create FindPeers()
5:    $p_k \leftarrow p_i$
6:    repeat
7:      send FindPeers() to
          $m$ neighbours of $p_k$ with interests most similar to $I_i$
8:      let $p_j$ be a neighbour of $p_k$ receiving FindPeers()
9:      $L \leftarrow L :: \langle ip(p_j), I_j \rangle$
10:     $p_k \leftarrow$ every $p_j$ receiving FindPeers()
11:     $\tau_R \leftarrow \tau_R - 1$
12:   until $\tau_R = 0$
13: return list $L$ to $p_i$
14: update $RI_i$ with information from $L$
```

Fig. 1 The rewiring protocol

When the message initiator p_i receives the FindPeers() message back, it utilises the information contained in L to update its routing index RI_i by replacing old short-range links corresponding to peers with less similar interests with new links corresponding to peers with more similar interests. Remark that for the linkage of the peers the rewiring protocol takes into account peers' interests, which stand for the content (and not the number) of their documents.

Peers store long-range links in their routing indexes, which stand as short paths to dissimilar clusters. For the update of the long-range links, peer p_i uses a random walk in the network to discover peers with dissimilar interests.

3.2.3 Query processing protocol

Let us assume that a user issues a query q through peer p_i. Initially, p_i compares q against its interest I_i.[6] If $sim(q, I_i) \geq \theta$, where $sim()$ can be any appropriate similarity function between a query and a peer interest and θ is the similarity threshold, then p_i creates a message of the form Query$(ip(p_i), q, \tau_b)$, where τ_b is the query TTL, and forwards it to *all* its neighbours using the short-range links in RI_i. This forwarding technique is referred to as *query broadcasting* (or *query explosion*) [23]: the query is broadcasted to the neighbours of p_i, which (due to clustering) are similar to p_i and will (most likely) be able to answer q.

If $sim(q, I_i) < \theta$, peer p_i forwards a Query$(ip(p_i), q, \tau_f)$ message with query TTL τ_f to m peers connected to p_i (using the short-range and the long-range links in RI_i) with interests most similar to q. The query message is thus, forwarded through

[6]If p_i has multiple interests then it compares q independently against each interest.

```
Procedure Query_Processing($q, p_i, \tau_f, \tau_b, \theta, m$)
Compares query $q$ against the document collection of $p_j$, retrieves
matching documents, and forwards $q$ to the network.

input: query $q$ issued by peer $p_i$ and threshold $\theta$
output: list $R$ of documents similar to $q$

1: if $sim(q, I_i) \geq \theta$ then
2:   compare $q$ against $p_i$'s local document collection
3:   if $sim(q, d) \geq \theta$ then
4:     $R \leftarrow R :: \langle p(d), m(d), sim(q, d)\rangle$
5:   send message RetRes($ip(p_j), R$) to $p_i$
6:   $\tau_b \leftarrow \tau_b - 1$
7:   forward QUERY() to all short-range links in $RI_i$
8: else
9:   forward QUERY() to $m$ neighbours of $p_i$ with interests
       most similar to $q$
10:   $\tau_f \leftarrow \tau_f - 1$
11: repeat the above procedure for $p_i$'s neighbours
12: until $\tau_f = 0$ or $\tau_b = 0$
```

Fig. 2 The query processing protocol

distinct paths from peer to peer until a peer p_j similar to the query is reached. Then, q is broadcasted in the neighbourhood of this peer as described in the previous paragraph. This query forwarding technique is referred to as *fixed forwarding* [23], since forwarding proceeds until q reaches a cluster of similar peers. All forwarding peers execute the aforementioned protocol and reduce τ_f by one at each step of the forwarding procedure. The combination of the two parts of query routing (i.e., fixed forwarding and broadcasting) is referred to in the literature as *fireworks* technique [16, 27].

Notice that the value of the broadcasting TTL τ_b is different from the value of the fixed forwarding TTL τ_f. Typically, τ_b is smaller than τ_f since in broadcasting the QUERY() message needs to reach peers only a few hops away (i.e., in the same cluster of the message recipient). In the case of fixed forwarding the message needs to explore regions of the network that are possibly far away from the query initiator. Figure 2 presents the query processing protocol.

3.2.4 *Document retrieval protocol*

Let us assume a peer p_j similar to the query q is reached. Apart from the forwarding protocol, p_j also applies the following procedure for retrieving documents similar to q. Query q is matched against p_j's local document collection and all documents d with $sim(q, d) \geq \theta$, where $sim()$ can be any appropriate similarity function between a query and a document and θ is the similarity threshold, are retrieved and ordered by similarity to q. Subsequently, p_j creates a result list R containing tuples of the form $\langle p(d), m(d), sim(q, d)\rangle$ for each relevant document d, where $p(d)$ is a pointer to d and $m(d)$ are metadata describing d (e.g., document title, author and an excerpt of the document's text in the style of search engine result presentation). The resulting list is placed in a message of the form RETRES$= (ip(p_j), R)$ and is returned to the peer that initiated the query using the contact information contained in the QUERY() message. In this way, query initiator p_i accumulates the results obtained by different peers, merges the different lists in a single list that contains unique entries sorted by descending similarity and presents the results to the user.

4 Proposed rewiring strategies

In the following, several variants to the basic rewiring protocol are proposed corresponding to modifications in (i) the routing technique used by peers to forward rewiring messages, (ii) the amount of information available to peers for updating their short-range links, (iii) the technique for establishing links, and (iv) the strategy used by peers to update their long-range links.

4.1 Forwarding of rewiring messages

A very important component of the rewiring protocol is the routing technique used by the peers to forward the rewiring messages. We introduce four different message forwarding techniques for the FINDPEERS() message and elaborate on the peer clustering performance.

4.1.1 The Gradient Walk strategy

This strategy is used in the basic rewiring protocol, where a message recipient p_j forwards the FINDPEERS() message to the set of m peers stored in its routing index RI_j with interests most similar to I_i.

The idea behind the Gradient Walk (GW) strategy is to forward the rewiring message to peers which are likely to be similar to I_i, collect information about these peers, use this information to update p_i's short-range links and eventually position p_i among similar peers. A drawback of this strategy is encountered for the case where clusters of similar peers cannot be reached or have not yet been formed. This is usually the case at system bootstrapping and also, at periods of high churn, where peers that initiate a rewiring process have low probability to discover other similar peers clustered together. In such cases, GW is expected to lead to poor peer clustering.

4.1.2 The Random Walk strategy

Under the Random Walk (RW) strategy, a message recipient p_j forwards the FINDPEERS() message to a set of m randomly chosen peers stored in RI_j. This is implemented by modifying the line 7 of Fig. 1 as follows:

```
send FINDPEERS() to $m$ random neighbours of $p_k$
```

The idea behind the RW strategy is to explore the network for peers with interests similar to I_i (the interest of the message initiator p_i), by making no assumption on the clustering of the network. In this way, at periods where the network is not well-clustered, a random exploration will increase the probability of finding peers with interests similar to I_i. Compared to the GW strategy, RW is expected to converge to a clustered network faster. Since peer clustering is used to support IR functionality, the RW strategy is also expected to present better retrieval performance than the GW strategy.

4.1.3 The combined strategy

In the combined (GW+RW) strategy, a message recipient p_j forwards the FINDPEERS() message with equal probability either to (i) a set of m randomly chosen peers, or (ii) the set of m peers with interests most similar to the interest I_i of the initiator peer p_i. This is implemented by modifying the line 7 of Fig. 1. The GW+RW strategy presented here resembles the strategy used in [23]. This strategy aims at combining the benefits from the RW and GW strategies, as the RW strategy can be applied in unclustered networks for reaching similar peers which are far apart from each other, while the GW strategy can be applied for exploring neighbourhoods of similar peers.

The rationale of applying both forwarding solutions at the same time is not only to connect p_i to similar peers discovered by forwarding the message to similar peer clusters, but also by enabling propagation of the forwarding message to other similar peers through non-similar peers. However, the GW+RW strategy combines the two components in a naive way by invoking each component based on a random decision.

4.1.4 The informed strategy

Under the informed (inf-GWRW) strategy, a message recipient p_j forwards the FINDPEERS() message either to (i) a set of m randomly chosen peers, or (ii) the set of m peers with interests most similar to the interest I_i of the initiator peer p_i, relying on how well the network is clustered. This strategy goes one step further from the GW+RW strategy, since it invokes each component based on an informed (rather than random) decision. In inf-GWRW, p_j takes into account how well the network is organised and decides whether to use gradient or random walk for message forwarding. This is implemented by modifying line 7 of the pseudocode shown in Fig. 1 as follows:

```
if $NS_k \geq \theta$ then
 send FINDPEERS() to
 $m$ neighbours of $p_k$ with interests most similar to $I_i$
else
 send FINDPEERS() to $m$ random neighbours of $p_k$
```

Contrary to the GW+RW strategy, inf-GWRW puts emphasis on peer autonomy and peer perception of network organisation.

4.2 Updating short-range links

In the basic organisation protocol, only the initiator of a FINDPEERS() message may utilise the message contents to collect information about peers with similar interests and update its routing index. In what follows, we examine the idea of letting also peers along the forwarding path to utilise the information collected by the message. In this way, other peers, aside from the initiator peer, may exploit this information to update their routing indexes and improve clustering without incurring extra message traffic.

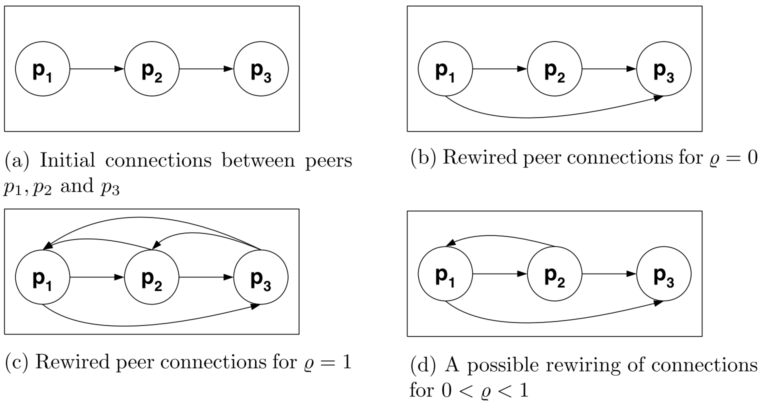
To achieve this, we utilise a new system-wide parameter coined *refinement probability* ϱ and let peers decide whether to exploit or not the information contained in FINDPEERS() message they received. This decision is taken non-deterministically by each peer in a fully decentralised way. Parameter ϱ takes values in the interval [0, 1] and is used as follows: every peer receiving a FINDPEERS() message may utilise the information contained in it (i.e., the interests of previous message recipients) with probability ϱ. When $\varrho = 0$, *no peer* apart from the message initiator use the contents of FINDPEERS() message, while when $\varrho = 1$, *all peers* participating in the forwarding of the FINDPEERS() message exploit the information contained in it to update their routing indexes. When $0 < \varrho < 1$, a peer p_j receiving a FINDPEERS() message exploits the information contained in it to update its routing index with probability ϱ. Notice that, when $\varrho = 0$ this protocol is reduced to the basic protocol. System parameter ϱ needs to be known upon bootstrapping. The use of ϱ is implemented by modifying the line 8 of Fig. 1 as follows:

```
let $p_j$ be a neighbour of $p_k$ receiving FINDPEERS()
$p_j$ generates a random number $x \in [0, 1]$
if $x \geq \varrho$ then update $RI_j$ with information from $L$
```

If peer p_j receiving a FINDPEERS() message decides to utilise the information in it, then p_j updates its routing index RI_j by replacing short-range links that are outdated or pointing to peers with dissimilar interests with links found in the message. It follows that, the initiator peer p_i of FINDPEERS() message *always* explores the information contained in the message to update its short-range links, since this peer initiated the rewiring process.

Example 1 Let us assume a network where each peer can store only 2 links to other similar peers ($s = 2$), rewiring TTL is $\tau_R = 2$, and that some arbitrary peers p_1, p_2 and p_3 that belong to the network are all interested in the same topic. Assume also, that these peers are initially connected to each other as shown in Fig. 3(a). In the following we demonstrate how peer organisation is affected by different values of ϱ.

- $\varrho = 0$. When peer p_1 decides to initialise a rewiring procedure, it sends a FINDPEERS() message with its interest to peer p_2. Remember that (by definition) when $\varrho = 0$ no peer apart from the message initiator use the contents of the message. Thus, p_2 just appends its own interest to it and forwards it to peer p_3. Similarly, p_3 appends its interest to the message and sends it back to the initiator peer p_1 (since τ_R has reached 0). Subsequently, p_1 uses the information in the message and connects to p_3 since they have similar interests. The resulting (rewired) links are shown in Fig. 3(b).
- $\varrho = 1$. In this case, all forwarding peers exploit the information contained in the FINDPEERS() message to refine their short-range links. When p_1 decides to initialise a rewiring procedure it contacts p_2. Since $\varrho = 1$, p_2 uses the information



(a) Initial connections between peers p_1, p_2 and p_3

(b) Rewired peer connections for $\varrho = 0$

(c) Rewired peer connections for $\varrho = 1$

(d) A possible rewiring of connections for $0 < \varrho < 1$

Fig. 3 An example illustrating the usage of the refinement probability ϱ

contained in the message and creates a link to p_1 as they have similar interests (remember that the message contains also the interest of the message initiator). Next, p_2 appends its interest to the message and forwards it to peer p_3, which also uses the information contained in the message and connects to both p_1 and p_2. Similarly, p_3 appends its interest to the message and sends it back to p_1 that in turn, updates its routing index. The resulting links are shown in Fig. 3(c).

Notice that, p_1, p_2 and p_3 have all their short-range links occupied with peers that have similar interests simply by using the rewiring message that p_1 initiated. To achieve this when $\varrho = 0$ would require more rewiring messages, since p_2 and p_3 would need to initiate their own rewiring procedure to collect information about similar peers and refine their links. In our example setting, $\varrho = 1$ seems to achieve a good result by organising p_1, p_2 and p_3 in a fully-connected cluster. However, this cluster (Fig. 3(c)) is isolated from the rest of the similar peers that may exist in the network, since there is no free short-range link to connect to other similar peers (remember that $s = 2$). Creating isolated peer clusters has a negative effect in recall, as a query might not reach all relevant clusters in the network.

Remark that allowing peers to store more short-range links in their routing indexes could also eliminate the isolated peer clustering problem. However, there is a trade-off related to the size of the routing index. Increasing the number of the short-range links eliminates the probability to create isolated peer clusters, but also results in increasing the number of peers belonging in a neighbourhood, which in turn entails more ambiguous peer clusters. However, the idea of peer rewiring is to cluster similar peers together, rather than let each peer has a vast knowledge of the network, and based on this clustering achieve good retrieval performance. In addition, increasing the number of short-range links stored by each peer results in increasing the message traffic in the network, since a query, according to the query processing protocol, is broadcasted in the neighbourhoods of similar peers. Consequently, the number of short-range links stored by a peer must be kept low compared to the number of peers in the network.

– $0 < \varrho < 1$. To avoid generating excess message traffic during rewiring (as with $\varrho = 0$) and also creating isolated peer clusters (as with $\varrho = 1$), ϱ should be appro-

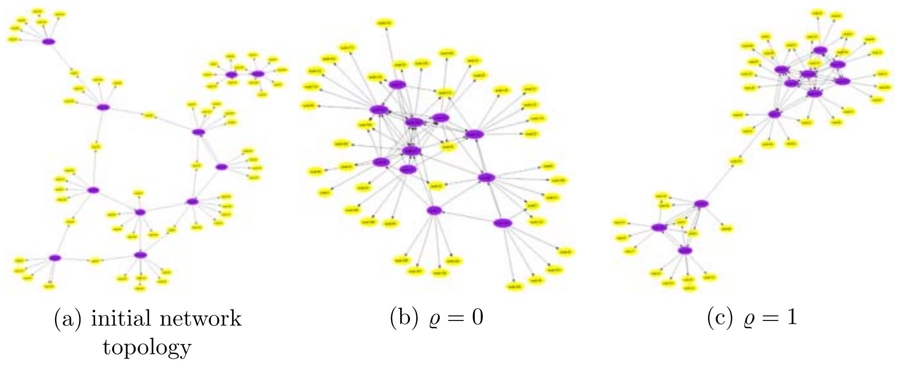

(a) initial network topology (b) $\varrho = 0$ (c) $\varrho = 1$

Fig. 4 An example illustrating network organisation when using different values of ϱ

priately tuned to a value between 0 and 1. Figure 3(d) shows a possible network organisation when $0 < \varrho < 1$ (in this case only peer p_2 utilised the information in the rewiring message initiated by peer p_1).

Figure 4 illustrates how ϱ affects peer clustering. The initial peer connections are shown in Fig. 4(a). Dark-coloured nodes represent peers of the same category, whereas light-coloured nodes represent peers from other categories that the former are connected to. Notice that, when $\varrho = 0$ most of the similar peers are clustered together, but there are also peers that are hard to reach (dark-coloured peers in the lower right part in Fig. 4(b)). When $\varrho = 1$ peers of the same category have formed two well-connected but disjoint neighbourhoods (Fig. 4(c)). This reveals an inherent weakness of rewiring: similar peers are not always clustered together, since they may form more than one clusters. Notice that this will also affect retrieval performance since queries are not always capable of locating all disjoint clusters in the network.

To recap, refinement probability ϱ can be used to tune the number of peers that update their links by exploiting the rewiring messages initiated by other peers. In general, when $\varrho > 0$ the network will converge faster to a clustered peer organisation, while with $\varrho < 1$ peers are discouraged from using the same information to refine their short-range links. This results in creating clusters where at least one path connecting all similar peers exist, which in turn increases retrieval effectiveness.

4.3 Introducing symmetric links

In the basic organisation protocol, when a peer p_i discovers that its interest is similar to that of a peer p_j, it is not entailed that p_j is aware of this similarity. We modify the basic protocol to facilitate symmetric links (SL) among peers. Under this strategy, if a peer p_i discovers another similar peer p_j through a FINDPEERS() message, p_i updates its routing index RI_i and sends a SIMPEER$(ip(p_i), I_i)$ message to p_j to inform it that they share similar interests. This is implemented by modifying the line 14 of Fig. 1 as follows:

```
update $RI_i$ with information from $L$
send SIMPEER$(ip(p_i), I_i)$ to $p_j : p_j \in L,\ sim(I_i, I_j) \geq \theta$
```

When p_j receives a SIMPEER() message, it uses the information in it to decide whether to refine or not its short-range links with a link to p_i. Notice that p_j may decide not to update its routing index RI_j if it is already connected to similar peers, i.e. $sim(I_j, I_k) \geq sim(I_j, I_i)$ for all interests I_k in RI_j.

Notice that this symmetry in link creation is not enforced by any way underlining peer autonomy. Notice also, that the SIMPEER() message is sent only when p_i updates its routing index. The only cost associated with this modification is one extra message for each RI_i update.

The idea behind this strategy is to exploit information discovered through the rewiring procedure of a peer to update links of other peers. This in turn, decreases the time required to create a clustered network. However, creating clusters of similar peers with the SL strategy can result in highly coherent peer neighbourhoods, that in turn increase the probability to create isolated cliques of similar peers (similarly to the case of $\varrho = 1$).

4.4 Updating the long-range links

Long-range links in a SON are exploited to facilitate long jumps in the network and allow reaching distant peer neighbourhoods with small communication overhead. The only requirement that needs to be met for the long-range links of a peer p_i is to point to peers that are *not* similar to p_i. We present two strategies that can be used by peers to obtain and maintain such links.

4.4.1 The Random Sampling strategy

According to the Random Sampling (RS) strategy which is used in the basic protocol, a peer p_i creates a FINDDISPEERS$(ip(p_i), I_i, \tau_R)$ message and forwards it in the overlay to search for peers with dissimilar interests. The message is forwarded in the network using a random walk strategy until message TTL τ_R reaches zero. Then, peer p_j that received the message compares its interest I_j against I_i contained in the message, if $sim(I_j, I_i) < \theta$ p_j appends its IP address $ip(p_j)$ and its interest I_j to the message and sends it back to p_i. Otherwise, p_j sends an appropriate message to notify p_i that they share similar interests. A peer p_i that receives a FINDDISPEERS() message back checks whether it already has a long-range link to a peer p_k for which $sim(I_j, I_k) \geq \theta$. If so, p_i disregards the message and re-initiates the procedure, since it already stores a short path towards the cluster that both p_k and p_j belong. Otherwise, p_i updates the long-range links of RI_i with a pointer to p_j.

4.4.2 The Biased Sampling strategy

The idea behind the Biased Sampling (BS) strategy is to use the rewiring protocol and the corresponding rewiring messages to collect information about non-similar peers. Under this strategy, a peer p_i that receives a FINDPEERS() message uses the information contained in it to update its long-range links as follows. It randomly selects a peer p_j contained in the message, for which $sim(I_i, I_j) < \theta$ and $sim(I_j, I_k) < \theta$ holds, for all its long-range links p_k. In this way, the long-range links of p_i contain

(i) peers with interests dissimilar to I_i and (ii) peers with dissimilar interests between each other.

Notice that peers do not have to explicitly initiate a rewiring procedure and the network is not loaded with extra messages. However, peers that belong in the same neighbourhood will tend to have high overlap in their long-range links, since they use the same messages to acquire these links. This *biased sampling* of the network may reduce retrieval effectiveness, since queries will follow similar routes and thus, leave parts of the network unexplored.

4.5 Combining all together

All the modifications to the basic rewiring protocol discussed above can be combined to increase the retrieval effectiveness and reduce the message overhead of any SON-based system. Summarising, the different rewiring strategies presented earlier are the following: (i) GW, RW, GW+RW, or inf-GWRW, (ii) ϱ taking values in the interval [0, 1], (iii) SL or non-SL, and (iv) RS or BS. Since the proposed strategies can be utilised independently of each other, this creates a large space of possible combinations. Notice that different (combinations of) rewiring strategies may emphasise recall, while others may be efficient in terms of network traffic. In the next section, we show how each individual strategy affects the performance of the basic protocol and also, identify interesting strategy combinations that can further improve system performance.

5 Experimental evaluation

In this section, we present our evaluation of the proposed rewiring protocols using two real-world datasets with web and medical documents.

Datasets. The first dataset contains over 556,000 web documents from the TREC-6[7] collection belonging in 100 categories and has been previously used to evaluate IR algorithms over distributed document collections (e.g., [30]). The second dataset is a subset of the OHSUMED TREC[8] document collection that contains over 30,000 medical articles from 10 different categories. The queries employed in the evaluation of both corpora are strong representatives of document categories and are issued from random peers in the network. Notice that this setting is a stress test for a SON, since we do not assume a query distribution that follows peers' interests.

Setup. We consider N loosely-connected peers, each of which contributes documents in the network from a single category. At the bootstrapping, peers are connected as described in Sect. 3.2.1. The base unit for time used in the experiments is the period t. The start of the rewiring procedure for each peer is randomly chosen from the interval $[0, 4K \cdot t]$ and its periodicity is randomly selected from a normal

[7] http://boston.lti.cs.cmu.edu/callan/Data/.

[8] http://trec.nist.gov/data/t9_filtering.html.

Table 1 Baseline parameter values

System parameter	Symbol	Value	Routing parameter	Symbol	Value
peers	N	2,000	rewiring TTL	τ_R	4
short-range links	s	8	fixed forwarding TTL	τ_f	6
long-range links	l	4	broadcast TTL	τ_b	2
similarity threshold	θ	0.9	message fanout	m	2

distribution of $2K \cdot t$, in the spirit of [20, 23]. Therefore, each peer starts (and goes over again) independently the rewiring process. We start recording the network activity at time $4K \cdot t$, when all peers have initiated the rewiring procedure at least once. We used a network size of 2,000 peers and our results were averaged over 25 runs (5 random initial network topologies and 5 runs for each topology). The average number of peers per class for the TREC-6 (resp. OHSUMED) corpus was 20 (resp. 200), with standard deviation 4.42 (resp. 68.8). The simulator used to evaluate the rewiring protocols and their modifications was implemented in C/C++ and all experiments were run on a Linux machine. The baseline parameter values used for the experiments are summarised in Table 1.

Determining the size of the routing index is an important task since it affects both retrieval effectiveness and network traffic. Small routing tables may result in a poorly organised network (and thus low retrieval performance), while large routing indexes cause high traffic at query time due to excessive broadcasting. In this work, the size of the routing index is determined by experimentation. A study that discusses how to set (some of) the parameters of Table 1 can be found in [20]. More elaborate methods for finding the appropriate size of the routing index is an important issue for future research.

Performance measures. As it is typical in the evaluation of P2P IR systems, performance is measured in terms of network traffic and IR effectiveness. The *network traffic* is measured by recording the number of rewiring (resp. search) messages sent over the network during rewiring (resp. querying). The IR effectiveness is evaluated using *recall*, i.e. the number of relevant documents retrieved over the total number of relevant documents in the network. Additionally, *recall/(search) message* is used to quantify a benefit/cost metric. Notice that precision is always 100% in our approach, since only relevant documents are retrieved. Our evaluation is goal-oriented: we are interested in measuring the system performance directly, without resorting to clustering quality measures (e.g., [5]) that might give misleading results [19]. One strategy is better than another if it presents high recall for less network traffic.

Document distribution. In the following, we examine the dependence of the system performance on the distribution of the documents over the peers. Thus, for a given rewiring strategy we measure both retrieval performance and network traffic for three different real-life document distributions. Figure 5 illustrates retrieval performance, in terms of recall and search messages, as a function of time for the RW strategy when the documents are distributed to the peers (i) uniformly, (ii) using a heavy-tailed distribution (i.e., zipf) and (iii) using the normal distribution. As shown in Fig. 5(a),

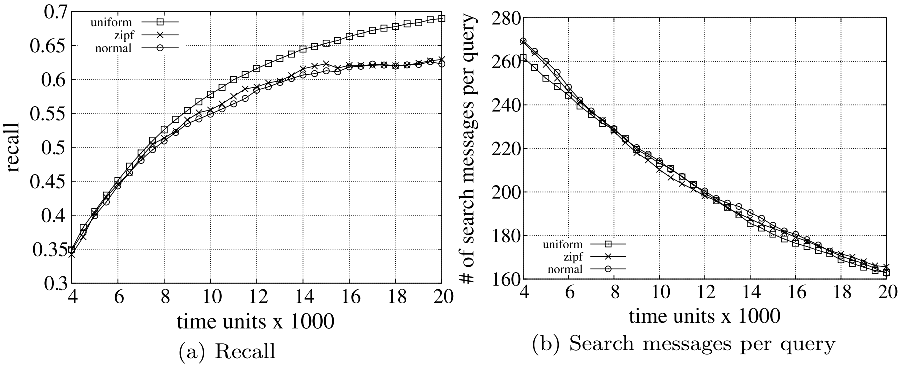


Fig. 5 Performance under different document distributions for the RW strategy (TREC-6)

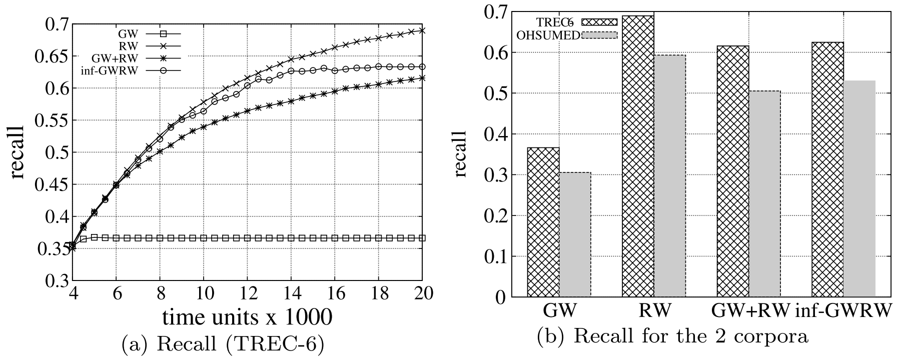


Fig. 6 Retrieval effectiveness for different forwarding strategies

recall is marginally affected by the document distribution. In terms of network traffic, Fig. 5(b) illustrates that the number of search messages is not affected by the document distribution. In the rest of this section, we use the uniform distribution to assign documents to peers.

5.1 Using different forwarding strategies

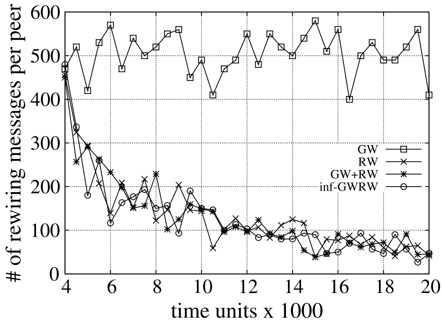
Figure 6(a) illustrates the retrieval effectiveness of the network as a function of time for different forwarding strategies. At the beginning of the rewiring procedure ($t = 4K$) peers are still randomly connected, the network is not yet organised into clusters of peers with similar interests and the values for recall are low (around 35%). After some time, when all peers have executed the rewiring protocol more than once ($t = 8K$), we can observe the effect of the different forwarding strategies to peer organisation and consequently, to retrieval effectiveness. The GW strategy improves recall only by 3%, which can be attributed to poor network organisation. The RW strategy demonstrated the best retrieval performance overall achieving up to 12% better recall than GW+RW and up to 8% better recall than inf-GWRW. In turn, the inf-GWRW strategy outperforms GW+RW achieving up to 5% better recall.

The fact that the RW strategy results in the best retrieval performance and also converges faster towards a clustered organisation of peers is attributed to the way it organises the network. The RW strategy explores the network in a random fashion increasing the probability to discover peers with similar interests, since peers are initially randomly connected. Even when similar peers start to get organised into clusters, using the RW strategy proves efficient in discovering isolated peers. The GW strategy explores peers' neighbourhoods to discover peers with similar interests and yields poor results, since in an unclustered (random) network peers are not always connected to other similar peers. The GW+RW strategy performs a random selection of a forwarding strategy at each invocation and is thus, limited by the bad performance of the GW strategy. The inf-GWRW strategy initially performs at least as good as the RW strategy. However, as the network converges towards a stable network, peers tend to choose the gradient walk for the forwarding of the rewiring messages reducing the probability to discover isolated peers.

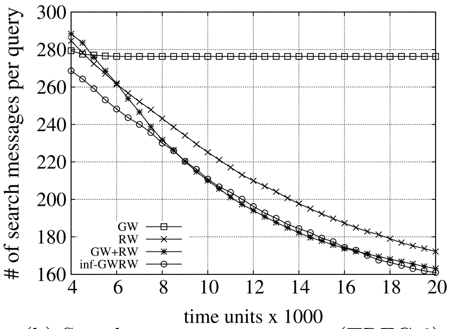
Figure 6(b) shows the recall for TREC-6 and OHSUMED corpora when the network is organised using different forwarding strategies. The values of recall presented in the figure correspond to an organised network ($t = 20K$). The RW strategy proves to be the best for both corpora in terms of recall.

The lower values of recall in the case of OHSUMED corpus are due to the different number of peers per class. Specifically, there are 200 peers per class when using the OHSUMED corpus, whereas there are 20 peers per class in the case of TREC-6 corpus. This difference in the number of peers per class affects retrieval performance in two ways: (i) The rewiring process tries to collect all similar peers in the same neighbourhood. Naturally, the smaller the number of peers per class is, the easier and the faster all similar peers are clustered together. When using the TREC-6 corpus the rewiring task is easier compared to when the OHSUMED corpus is used. (ii) Recall depends (among others) on the broadcasting TTL τ_b and on the number of short-range links s stored by each peer. Even though all similar peers are gathered together in the same neighbourhood (no matter their number), there is the case that a query may not reach all peers within the cluster if the diameter of the corresponding peer neighbourhood is bigger than s^{τ_b}. We choose to use small values for s and τ_b, since peer clusters are more cohesive and network traffic is kept low.

Figure 7 presents network traffic (rewiring and search messages) for the four strategies over time. In terms of rewiring messages (Fig. 7(a)), the network initially presents a high message overhead, which is greatly reduced (over 200%) for the RW, the GW+RW and the inf-GWRW strategies when the network starts to get organised into clusters ($t > 8K$). Apparently, the GW strategy does not manage to reach an effective peer organisation and peers continue executing the rewiring protocol to discover peers with similar interests, which leads to high message traffic. The other three strategies manage to efficiently and quickly organise the network and maintain an effective peer organisation at a small communication cost. All strategies need high number of search messages (Fig. 7(b)) when the network is not yet organised into coherent neighbourhoods (left-most points in the x-axis). However, this message overhead is decreased (65% decrease) for RW, GW+RW and inf-GWRW as the peers get organised into clusters with similar interests (right-most points in the x-axis). The GW strategy does not improve network traffic, as it does not manage to efficiently or-

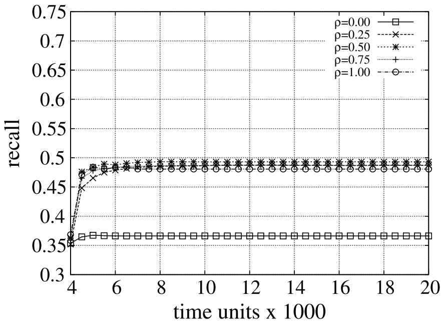


(a) Rewiring messages (TREC-6)

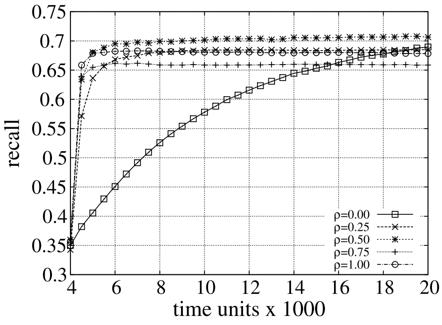


(b) Search messages per query (TREC-6)

Fig. 7 Network traffic (rewiring and search messages) for different forwarding strategies



(a) Recall when using GW strategy



(b) Recall when using RW strategy

Fig. 8 Retrieval effectiveness for different values of ϱ (TREC-6)

ganise the network. Notice that the inf-GWRW and GW+RW strategies performs slightly better in terms of overall network traffic compared to the RW strategy.

5.2 Varying the refinement probability

Figure 8 illustrates retrieval effectiveness for various values of ϱ as a function of time for two forwarding strategies: (i) GW used in our basic protocol and (ii) RW exhibiting better retrieval performance than its competitors.

Figure 8(a) illustrates the way ϱ affects retrieval performance when the GW strategy is used. When the network is unorganised ($t = 4K$) the queries cannot be routed efficiently resulting in low recall (approximately 35%). When the network starts to organise ($t = 6K$), higher values of recall are achieved. The retrieval performance of GW strategy improves 50% for $\varrho > 0$, which is attributed to the fact that more peers update their short-range links by exploiting the rewiring messages.

Figure 8(b) presents recall over time when using the RW strategy. When ϱ is low (i.e., <0.5), the network is clustered at a slow rate since more peers need to initiate the rewiring process. As ϱ increases, the network converges faster to organised clusters of peers. When ϱ is high (i.e., > 0.5), network organisation and recall have converged

Table 2 Performance for RW when the network is organised ($t = 20K$) for both corpora

ϱ	OHSUMED			TREC-6		
	recall	search msgs	recall/msg ($\times 10^{-3}$)	recall	search msgs	recall/msg ($\times 10^{-3}$)
0.00	0.53	282	1.88	0.69	172	4.01
0.25	**0.68**	282	**2.41**	0.68	142	4.79
0.50	0.55	**232**	2.37	**0.72**	**140**	**5.14**
0.75	0.54	238	2.31	0.66	141	4.68
1.00	0.58	261	2.22	0.67	140	4.79

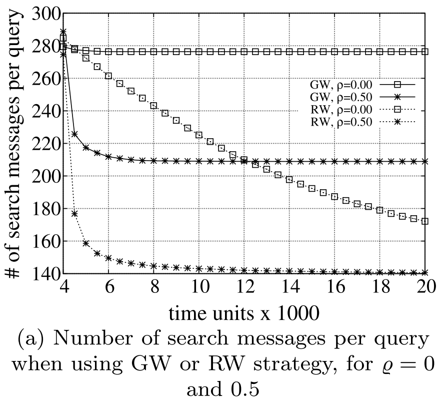
to a stable state fast and remain unchanged. In these cases, the relatively low values of recall imply the existence of isolated peers. The higher value of recall is achieved for $\varrho = 0.5$, i.e. when half of the forwarding peers use FINDPEERS() message to update their short-range links.

In general, when $\varrho > 0$ the network converges faster to a clustered peer organisation by using less messages, and when $\varrho < 1$ peers are discouraged from creating isolated clusters. The ideal value for ϱ is the one resulting in connected clusters (i.e., there is a path connecting all similar peers), while using minimum number of rewiring messages. Table 2 presents recall, search cost and recall/message for the RW strategy, for varying ϱ and over both corpora in an organised network ($t = 20K$). When $\varrho = 0$, RW presents the lowest recall/message for both corpora. When $\varrho = 0.50$, RW presents a good performance for both corpora (lowest message traffic, highest and second highest recall/message and a high recall value). The best value of ϱ in terms of recall is 0.25 when using OHSUMED corpus and 0.5 when using TREC-6 corpus.

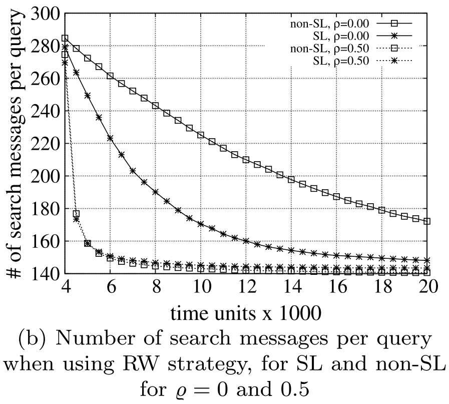
The different effect of ϱ on system performance across different corpora is attributed to the different number of peers per class. Remember that increasing the value of ϱ, the probability of creating fully-connected peer clusters is also increased. Given that the number of short-range links is much smaller than the number of peers per class, in the case of OHSUMED corpus increasing the value of ϱ turns similar peers to create many isolated clusters thus, resulting in bad retrieval performance. Conclusively, the higher the number of peers per class is, the lower the value of ϱ should be. In this way, more rewiring messages may be generated, getting though an efficient peer clustering.

Figure 9(a) shows the number of messages per query for $\varrho = 0$ (i.e., the value used in the basic rewiring protocol) and 0.5 (i.e., the value that resulted in the best retrieval performance) for the GW and RW strategies. When the network is not yet organised a higher number of search messages is needed. This message overhead is decreased (65% decrease for RW) when $\varrho = 0.5$ as peers get organised. When $\varrho = 0$, the network converges slowly to organised peer neighbourhoods and the message overhead decreases with a slower rate.

In terms of rewiring messages, the network initially presents a high message overhead, which is greatly reduced (by a factor of 15) when the network organises into coherent clusters (around moment $6K$) for $\varrho > 0$. When $\varrho = 0$, the network does not manage to quickly reach an effective peer organisation and peers keep executing the rewiring protocol presenting higher message overhead and slower decrease rate

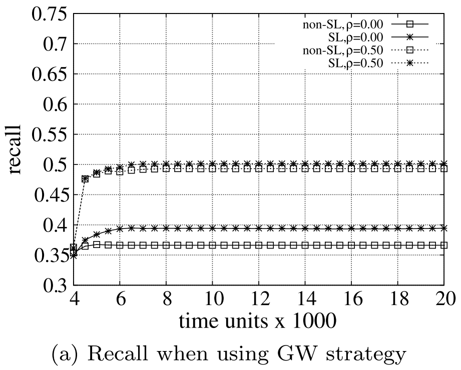


(a) Number of search messages per query when using GW or RW strategy, for $\varrho = 0$ and 0.5

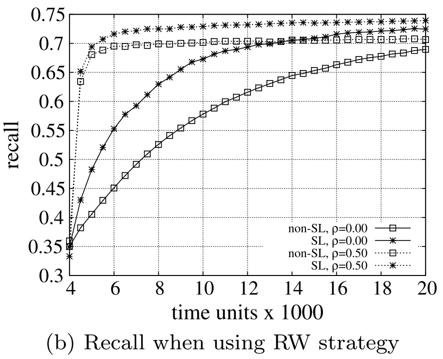


(b) Number of search messages per query when using RW strategy, for SL and non-SL for $\varrho = 0$ and 0.5

Fig. 9 Search cost for different rewiring strategies (TREC-6)



(a) Recall when using GW strategy



(b) Recall when using RW strategy

Fig. 10 Retrieval effectiveness for SL and non-SL corresponding to $\varrho = 0$ and 0.5 for different forwarding strategies (TREC-6)

when compared to the case where $\varrho > 0$. The corresponding graphs are omitted due to space constraints.

5.3 Symmetric links

Figures 10(a) and (b) show the effect of symmetric versus non-symmetric links in the retrieval performance over time for two different forwarding strategies and for two different values of ϱ. The data set used in this set of experiments is TREC-6. The SL strategy, when used with the GW strategy, achieves better retrieval performance (about 7%) compared to the basic rewiring protocol (Fig. 10(a)). The effect of the SL strategy on the retrieval performance is small in the case of $\varrho = 0.50$, since some of the links introduced by SL are also created due to ϱ (forwarding peers have already updated their links to point to previous message recipients and thus, SL has no additional effect on clustering). The effect of creating symmetric links is higher in the case of the RW strategy, since more similar peers per rewiring process are discovered (Fig. 10(b)). The SL strategy improves retrieval performance by 10% for both

Table 3 Recall for RW when the network is organised ($t = 20K$) for both corpora

ϱ	OHSUMED		TREC-6	
	non-SL	SL	non-SL	SL
0.00	0.53	0.64	0.69	0.73
0.25	0.68	0.57	0.68	0.75
0.50	0.55	0.56	0.72	0.75
0.75	0.54	0.54	0.66	0.76
1.00	0.58	0.54	0.67	0.74

values of ϱ and facilitates network converge towards organised peer clusters. Notice that when using the SL strategy along with $\varrho = 0.5$, the network organises faster than when using the SL strategy or $\varrho = 0.5$ alone. Consequently, creating symmetric links has a positive effect in retrieval performance.

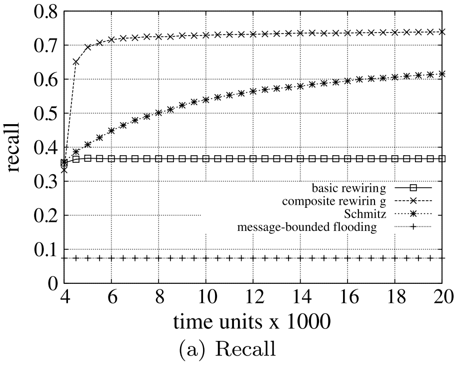
Table 3 presents recall in an organised network ($t = 20K$) for both corpora and for different values of ϱ. Notice that when $\varrho > 0$, SL does not always improve recall on the OHSUMED corpus. When the OHSUMED corpus is used, there are many peers per category and, even though similar peers are all clustered together, the retrieval performance is also affected by other parameters, as the number of short-range links s and the broadcasting TTL τ_b.

Figure 9(b) illustrates on the number of messages per query for the RW strategy and for two values of ϱ as a function of time. When $\varrho = 0$, the SL strategy decreases network traffic by 18% when compared to the case of non-symmetric links between peers. The effect of the SL strategy in network load is smaller when $\varrho = 0.50$. By cross examining the results presented in Figs. 10(b) and 9(b), we conclude that the SL strategy manages to efficiently organise peers and thus, improve retrieval effectiveness and search costs.

The SL strategy achieves better network organisation and thus, better retrieval performance with low communication cost in terms of rewiring messages. Notice that this strategy was expected to impose extra message traffic, due to the additional messages sent for supporting the symmetric links. In fact, the SL strategy decreases message traffic as it exploits existing traffic to rewire the links of the peers. In terms of rewiring, our experiments show that the SL strategy requires in general 10% less messages to organise the network and also, results in a faster clustering when compared to the non-SL strategy.

5.4 Updating long-range links

The RS and BS strategies are used for updating the long-range links. Notice that, we do not explore ϱ or SL at the same time since they do not affect the long-range links. The retrieval performance of RS shows an average improvement of about 5% over BS. In terms of message overhead, the RS strategy imposes around 15% more rewiring messages as it uses extra message traffic to discover peers with dissimilar interests and presents almost the same number of search messages per query compared to the BS strategy. From this set of experiments, we conclude that the update strategy for the long-range links has small impact on the performance, since both strategies perform similarly, with RS presenting slightly better retrieval but also higher



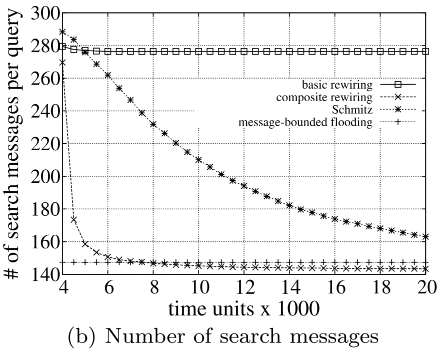


Fig. 11 Comparison of different rewiring protocols (TREC-6)

message traffic during rewiring. The corresponding graphs are omitted due to space constraints.

5.5 Composite rewiring

In the following, we combine the best strategies identified above. The corresponding strategy is called *composite rewiring* and its performance is compared against (i) the basic rewiring protocol presented in Sect. 3.2.2, (ii) the clustering approach of Schmitz [23], modified for an IR task and using the parameter values of Table 1 to obtain results comparable to our setting, and (iii) a message-bounded flooding algorithm, i.e. a modified flooding strategy that terminates when reaching a predetermined number of messages (in our case the number of messages of the most efficient strategy). Message-bounded flooding was implemented to serve as a baseline for the rest of the strategies.

Composite rewiring strategy uses the best strategies as identified in the previous sections: (i) comparing the different forwarding strategies, the RW strategy presented the best performance, (ii) experimenting with different values for ϱ, $\varrho = 0.5$ resulted in the best retrieval performance, (iii) showing the effect of symmetric links, the SL strategy achieved better retrieval performance with low communication cost, and (iv) experimenting on strategies for updating the long-range links, the RS strategy performed slightly better than its competitor. Therefore, composite rewiring consists of utilising the RW forwarding strategy, $\varrho = 0.5$, the SL strategy for creating short-range links, and the RS strategy for updating long-range links. This combination is expected, according to the results presented earlier, to perform well in terms of recall, while imposing lower message costs than competitors.

Figures 11(a) and (b) illustrate the retrieval performance and communication load as a function of time for the different rewiring protocols. The data set used in this set of experiments is TREC-6. In Fig. 11(a), we observe that composite rewiring achieves a 97% increase in recall when compared to the basic rewiring and about 21% increase when compared to the approach of [23]. In Fig. 11(b) we observe that the composite strategy is the most efficient in terms of message traffic (remember that flooding is message-bounded). Similar results have been also obtained for the OHSUMED corpus.

5.6 Discussion

The different rewiring components (as identified in Sect. 4) are: (i) GW, RW, GW+RW, or inf-GWRW strategy, (ii) ϱ taking values in the interval [0, 1], (iii) SL or non-SL strategy, and (iv) RS or BS strategy. There are many choices to combine making up a range of possibilities worth exploring. Notice that different combinations of rewiring strategies may emphasise recall and behave slightly worse in terms of message costs and vice versa.

When message traffic is in question, the inf-GWRW strategy performs better while achieving reasonably high recall values (Fig. 6). The RW strategy emphasises recall for a small increase in message traffic (Fig. 6). Setting $\varrho = 0$ leads to increased traffic and moderate retrieval performance, while a value of $0.25 \leq \varrho \leq 0.75$, achieves a good trade-off between traffic and retrieval effectiveness (Fig. 8). The SL strategy improves the overall system performance (Fig. 10), contrary to the updating strategy of long-range links that shows no significant performance differences.

6 Conclusion

We presented a comprehensive study of both existing and innovative strategies for rewiring protocols and showed how performance (in terms of retrieval effectiveness and communication overhead) depends on each component of the rewiring protocol. Subsequently, we have identified a combination of components that outperforms existing rewiring protocols. We are currently working on a rewiring protocol for highly dynamic settings, where peer churn and data dynamicity follow appropriate models that exist in the literature. We study the way churn affects the system performance and try to identify the rewiring protocol that performs well under dynamic networks.

References

1. Aberer, K., Cudre-Mauroux, P., Hauswirth, M.: The Chatty web: Emergent semantics through gossipping. In: WWW (2003)
2. Garcia-Molina, H., Yang, B.: Efficient search in peer-to-peer networks. In: ICDCS (2002)
3. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, Cluster Analysis. Academic Press (2001)
4. Hidayanto, A., Bressan, S.: Towards a society of peers: Expert and interest groups in peer-to-peer systems. In: OTM Workshops (2007)
5. Hui, K., Lui, J., Yau, D.: Small-world overlay P2P networks: Construction, management and handling of dynamic flash crowds. Computer Networks (2006)
6. Jelasity, M., Montresor, A.: Epidemic-style proactive aggregation in large overlay networks. In: ICDCS (2004)
7. Kantere, V., Tsoumakos, D., T.K.S.: Semantic grouping of social networks in P2P database settings. In: DEXA (2007)
8. Klampanos, I., Jose, J.: An architecture for information retrieval over semi-collaborating peer-to-peer networks. In: ACM SAC (2004)

9. Koloniari, G., Pitoura, E.: Recall-based cluster reformulation by selfish peers. In: NetDB (2008)
10. Li, M., Lee, W.C., Sivasubramaniam, A.: Semantic small world: An overlay network for peer-to-peer search. In: ICNP (2004)
11. Loser, A., Tempich, C.: On ranking peers in semantic overlay networks. In: WM (2005)
12. Loser, A., Wolpers, M., Siberski, W., Nejdl, W.: Semantic overlay clusters within super-peer networks. In: DBISP2P (2003)
13. Lu, J., Callan, J.: Content-based retrieval in hybrid peer-to-peer networks. In: CIKM (2003)
14. Merugu, S., Srinivasan, S., Zegura, E.: Adding structure to unstructured peer-to-peer networks: The use of small-world graphs. Parallel Distributed Comput. **65**(2), 142–153 (2005)
15. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmer, M., Risch, T.: EDUTELLA: A P2P networking infrastructure based on RDF. In: WWW (2002)
16. Ng, C.H., Sia, K.C., Chang, C.H.: Advanced peer clustering and firework query model in the peer-to-peer network. In: WWW (2002)
17. Parreira, J.X., Michel, S., Weikum, G.: p2pDating: Real life inspired semantic overlay networks for web search. Information Processing and Management (2007)
18. Penzo, W., Lodi, S., Mandreoli, F., Martoglia, R., Sassatelli, S.: Semantic peer, here are the neighbors you want! In: EDBT (2008)
19. Raftopoulou, P., Petrakis, E.: A measure for cluster cohesion in semantic overlay networks. In: LSDS-IR (2008)
20. Raftopoulou, P., Petrakis, E.: iCluster: A self-organising overlay network for P2P information retrieval. In: ECIR (2008)
21. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: ACM SIGCOMM (2001)
22. Sacha, J., Dowling, J., Cunningham, R., Meier, R.: Discovery of stable peers in a self-organising peer-to-peer gradient topology. In: DAIS (2006)
23. Schmitz, C.: Self-organization of a small world by topic. In: P2PKM (2004)
24. Schmitz, C., Loser, A.: How to model semantic peer-to-peer overlays? In: P2PIR (2006)
25. Spripanidkulchai, K., Maggs, B., Zhang, H.: Efficient content location using interest-based locality in peer-to-peer systems. In: INFOCOM (2003)
26. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup protocol for Internet applications. IEEE/ACM Trans. Netw. **11**(1) (2003)
27. Tang, C., Xu, Z., Dwarkadas, S.: Peer-to-peer information retrieval using self-organizing semantic overlay networks. In: SIGCOMM (2003)
28. Triantafillou, P., Xiruhaki, C., Koubarakis, M., Ntarmos, N.: Towards high performance peer-to-peer content and resource sharing systems. In: CIDR (2003)
29. Voulgaris, S., van Steen, M., Iwanicki, K.: Proactive gossip-based management of semantic overlay networks. Concurr. Comput.: Pract. Experience **19**(17) (2007)
30. Xu, J., Croft, W.: Cluster-based language models for distributed retrieval. In: ACM SIGIR (1999)