UNIVERSITY OF CALIFORNIA
RIVERSIDE

Adaptive Data Dissemination and Content-Driven Routing in Peer-to-Peer Systems

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Science

by

Thomas S. Repantis

August 2005

Thesis Committee:
Dr. Vana Kalogeraki, Chairperson
Dr. Dimitrios Gunopulos
Dr. Michalis Faloutsos

The Thesis of Thomas S. Repantis is approved:

Committee Chairperson

University of California, Riverside

Acknowledgements

I would like to express my gratitude and appreciation to my advisor Dr. Vana Kalogeraki, for the inspiration and guidance, motivation and support she has offered me.

I am also grateful to my committee members, Dr. Dimitrios Gunopulos and Dr. Michalis Faloutsos, for their time and their insightful feedback.

Finally, I would like to thank the rest of the members of the Distributed Real-Time Systems lab and the anonymous reviewers of [33] for their comments.

ABSTRACT OF THE THESIS

Adaptive Data Dissemination and Content-Driven Routing in Peer-to-Peer Systems

by

Thomas S. Repantis

Master of Science, Graduate Program in Computer Science
University of California, Riverside, August 2005
Dr. Vana Kalogeraki, Chairperson

Peer-to-Peer systems have emerged as a cost-effective means of sharing data and services and are offering fault-tolerance and self-adaptation in large-scale environments. However, the efficient location of data objects or services in a fully decentralized, self-organizing, unstructured overlay network remains a challenging problem. Most of the current solutions rely on maintaining global knowledge or generate large amounts of traffic and as a result do not scale well.

In this work we propose adaptive data dissemination and content-driven routing algorithms for intelligently routing search queries in large-scale, unstructured systems. In our mechanism nodes build and maintain content synopses of their objects and adaptively propagate them to the most appropriate peers. Based on the content synopses, a routing mechanism is being built to forward the queries to those nodes that have a high probability of providing the desired results. Through extensive simulation studies of networks of thousands of nodes and for different content synopses propagation strategies, we show that our approach is highly scalable and significantly improves resources usage by saving both bandwidth and processing power.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The explosive growth of rich-media online content, such as audio, video, news articles, images, and documents has created new challenges for real-time collaboration among multiple users in large-scale distributed environments. At the same time, advances in the networking, processing and storage capabilities of personal computers have signaled the emergence of peer-to-peer (P2P) systems as a platform for providing and receiving data and services. According to the peer-to-peer paradigm, nodes act autonomously to form large-scale distributed systems that enable the sharing of their resources. The peers form an overlay, as is shown in Figure 1.1, a logical network over the physical network, and act as both clients and servers. They employ their own location and routing mechanisms and maintain soft state information about other nodes. The peers can be geographically distributed, heterogeneous in their resource capabilities, and dynamic in their participation in the system. Peer-to-peer systems have been used with great success for storing and sharing data [18, 27, 10] as well as for per-

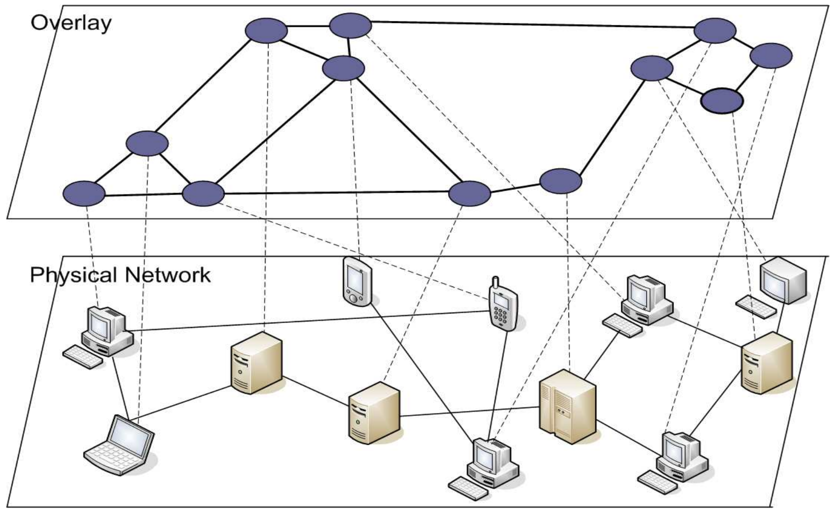

Figure 1.1: The peers form an overlay on top of the physical network.

forming distributed computations [40, 16, 12]. Some of their attractive features include cost effectiveness (by aggregating existing resources), increased autonomy (by self-organizing), improved scalability (due to the absence of a central coordinator's bottleneck), and reliability (due to lack of a single point of failure).

Two main approaches have emerged for constructing peer-to-peer networks: *structured* and *unstructured* overlays. *Structured* overlay networks are organized in such a way that objects are located at specific nodes in the network and nodes maintain some state information, to enable efficient retrieval of the objects. On the other hand, in *unstructured* overlay networks, objects can be located at random nodes, and nodes are able to join the system at random times and leave it without a priori notification. Hence, unstructured overlays are better in coping with churn [35] –the continuous process of node arrival and departure– and the heterogeneity of the peers. Furthermore, unstructured overlays have been deployed and are actively been used by millions of Internet users.

However, in an unstructured topology several design issues arise, one of the most challenging ones being the efficient search and retrieval of data or services [1]. The major issue is that no central manager can have an accurate global view of the system's contents. The problem is complicated further by the fact that the environment is dynamic and heterogeneous. Peers join, leave, and fail without a priori notification and have very different and restricted processor, storage and communication capabilities. Finally, in a large-scale peer-to-peer network, the amount of traffic generated by queries can be overwhelming.

Traditionally, search in unstructured peer-to-peer networks is performed based on keyword queries by flooding the network with messages and propagating the search query hop-by-hop until the desired answer is found. The problem with this approach is that it fails to take into account the probability of a node to be able to provide the asked object. Hence, the search messages travel a large number of hops, wasting processing power of many nodes, and producing large amounts of network traffic, while the answer to the query is delayed.

Recently proposed techniques [46] use the keywords of the queries to compute the similarity of the query message to previously seen queries, to probabilistically forward the query to only a subset of the nodes. These rely on knowledge collected locally at the peer by monitoring the messages propagated in the network.

Content summarization [28] is another technique that has been proposed to tackle the aforementioned problem. It is recently receiving a lot of attention as a means to reduce latency, balance the query load and alleviate hot spots. Nodes construct summaries of the

[1] We will be using the term "object" to refer to both data and services.

objects in their local stores and propagate them to their peers. By having access to these summaries, a node can perform a local search to determine which nodes have the requested object and thus can efficiently decide where to propagate a query, to maximize the probability for a fast reply.

However, when using the content summaries, it is important to intelligently decide to which nodes and how often to propagate them to the network. Since content summaries are passed around in messages, they introduce some performance cost. Storing the summaries of the contents of all the peers in the network in one node is impossible due to bandwidth and storage limitations and also because of the dynamic behavior of the peers. In such large-scale systems, changes to the stored data happen more often than they can be communicated to a single peer. Thus, the overlay network can greatly benefit from intelligent decisions regarding when and where content summaries are propagated.

In this work [33] we target the problem of data dissemination in unstructured, decentralized peer-to-peer networks. We propose adaptive data dissemination and content-driven routing protocols to reach the requested objects, while keeping the number of propagated messages small. In our mechanisms nodes build and maintain content summaries of their local data and adaptively disseminate them to their most appropriate peers. Peers use the *Bloom filter* data structure [3] to build a synopsis of their local content. Bloom filters allow us to answer cardinality queries with a certain probability. Nodes disseminate their content synopses to other peers, so that they can use them to efficiently route queries for objects. We investigate and compare three different techniques for adaptively selecting the most appropriate recipients of a peer's content synopsis, taking into account the number and type of queries sent by other peers in the past. The goal of the adaptive decision is to selectively propagate the synopsis to those nodes that need them the most for their routing decisions, while keeping the number of transferred synopses low. Our experimental results validate the performance benefits of our approach.

1.1 Contributions

Our major contributions are:

1. We propose a **content-driven routing mechanism** for finding objects in large-scale, unstructured peer-to-peer networks. Our mechanism propagates the queries to those peers that have a high probability of providing the desired results. The mechanism is driven by content synopses that are stored locally at the peers.

2. We propose **adaptive data dissemination algorithms** that decide to which peers to propagate the content synopses to improve the search and retrieval of the objects and make more efficient use of the bandwidth and processing power resources. The novelty of our approach is that content summaries are propagated dynamically to selected peers based on the requests and replies generated by those peers.

3. We present an extensive experimental study of large-scale networks, that illustrates that our mechanism reduces the number of messages sent, the number of peers contacted

and achieves high recall efficiency, in comparison to other popular searching techniques, even in the presence of disconnecting nodes. We compare the performance of our mechanism under different content-based propagation strategies and discuss their results.

The rest of the thesis is organized as follows: In chapter 2 we present the architecture of our adaptive data dissemination and content-driven routing mechanism in detail. In chapter 3 we describe the experimental evaluation of our approach and discuss our results. We review related work in query routing and data dissemination in overlay networks in chapter 4. Finally, we draw conclusions and explore avenues to future work in chapter 5.

Chapter 2

System Architecture

In this chapter we present our system architecture in detail. Section 2.1 presents our network model, while Section 2.2 gives an overview of our system's operation. Section 2.3 describes the content synopsis data structure our system uses, and Section 2.4 describes the content-driven routing mechanism. Section 2.5 presents the content synopses dissemination strategies, followed by Section 2.6, which describes the parameters taken into account in the adaptive synopses dissemination. Finally, Section 2.7 discusses implications of dynamic behavior in the synopses dissemination.

2.1 Overlay Model

We consider an overlay network of N nodes (peers) that store objects. The overlay is constructed on top of the physical network and the peers are linked through virtual connections. Each peer has a globally unique identifier (e.g. port:IP) and maintains connections with other peers. The network is unstructured, decentralized and self-organizing, meaning that peers make their own decisions on which peers to connect to or to query for objects. The number of connections of a peer can vary and is typically restricted by the resource capabilities of the peer. The peers of a node can be randomly selected, defined a priori based on some optimization criteria (such as round-trip delays), or dynamically established and revised in response to the node interactions or changes in the processing and networking conditions [17]. Our mechanisms aim to facilitate searching in any type of unstructured peer-to-peer network. Peers that are not directly connected communicate through relaying. In other words, peers not only exchange messages with their neighbors, but also route messages coming from other peers.

Each object stored in a peer is uniquely identified by the means of intrinsic references [14] which are generated when the object is first inserted in the system. Intrinsic references are based on the hash digest of the object's actual contents rather than its name or location and therefore allow us to create persistent, state-independent, and immutable storage. Alternatively, each object can be associated with a set of keywords to allow meta-data types of searching. The mechanisms presented in this paper are orthogonal to the type of search and therefore we just focus on searching by an object's intrinsic reference.

2.2 System Operation

Each peer uses the *Bloom filter* data structure [3] to build a synopsis of the content in its local store. Bloom filters are compact data structures that represent a set of objects stored at each peer by using an array of bits; each bit takes a binary one or a zero value. The cardinality of an object is checked by comparing the bit array generated by hashing the object by multiple hash functions, to the bit array of the Bloom Filter data structure. This allows us to answer with a certain probability whether the object is in the group or not. Each peer stores two types of filters, a *local filter* for the objects available locally at the node and *remote filters* for objects stored in remote peers. The node sends its local filter to remote peers. The recipients of the filter are selected adaptively, by taking into account the number and type of queries sent by the peers. The synopses are used to efficiently route queries for objects. The goal of the adaptive decision is to selectively propagate the synopsis to those nodes that need them the most for their routing decisions, while keeping the number of transferred synopses low.

Peers search for objects by sending query messages to their immediate neighbors. Those queries are evaluated locally in each peer and in case matching objects exist, results are returned to the searching peer. Otherwise, the query is routed to those of its peers whose synopses present a closest match. Figure 2.1 illustrates our system's operation.

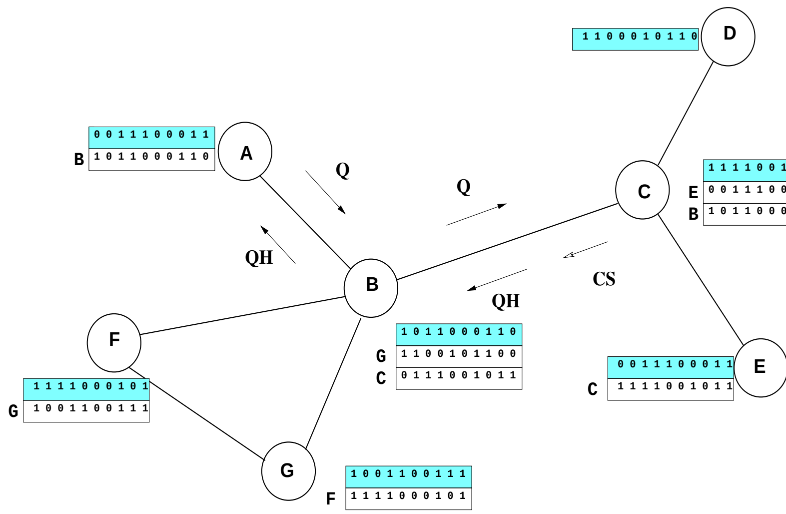

Figure 2.1: System operation example. Each node maintains a local content synopsis, as well as content synopses of remote peers. Using the synopses each node is able to route queries more efficiently. In this example, peer C propagated its content synopsis CS to peer B. B based on CS was able to route peer A's query Q only to C, and the result QH is routed back to A.

2.3 Content Synopses

In this Section we describe the data structure we use to summarize each peer's content. Assume that peer p has a group of n objects given by the set $S_p = a_1, a_2, ..., a_n$. The Bloom filter that represents the set S_p is described by a bit array BF_p of length m, all initially set to 0. We assume k hash functions, $h_1, h_2, ..., h_k$ with $h_i : X \rightarrow 1...m$. Each hash function maps each element of the set S to a value between $1...m$ in a totally random fashion. For each element $s \in S$, the bits at position $h_1(s), h_2(s), ..., h_k(s)$ are set to 1. Note though, that, a bit may be set to 1 multiple times. To determine whether a certain element x is in S, we check whether all the bits given by $h_1(x), h_2(x), ..., h_k(x)$ are set to 1. If any of them is 0, then we are certain that the element x is not in the set S. If all $h_1(x), h_2(x), ..., h_k(x)$ are set to 1, we conclude that x is in S, although there is a certain probability that we are wrong. This is the case that a Bloom filter may yield a *false positive*. After inserting n elements into a Bloom filter of size m using k hash functions and letting p_0 be the probability that a specific bit is still 0, the probability of a false positive (the probability that all k bits have been previously set) is shown [3, 15] to be: $p_{err} = (1 - p_0)^k = \left\{1 - \left\{1 - \frac{1}{m}\right\}^{kn}\right\}^k \approx \left\{1 - e^{\frac{-kn}{m}}\right\}^k$. As this equation shows, there exists a trade-off between k, m, n, and the accuracy of the objects' representation using Bloom filters. This trade-off is investigated experimentally in 3.4.1.

Our system exploits the probability that a small number of false positives does not greatly affect the performance of our searching mechanism. This fact makes the Bloom filter approach highly suitable for locating objects accurately and fast.

To support the removal of members from the sets represented by the Bloom filters we use counting Bloom filters. In this approach, a counter is added to each bit in the filter, so that the number of objects that are hashed in the same position is counted. An example of a counting Bloom filter is shown in Figure 2.2 (i).

Each peer may store content synopses for several peers connected to it, indexed by their IDs. Moreover, each of those content synopses, may contain not only the Bloom filter of the peer's local content (*local filter*), but also Bloom filters of the content of remote peers connected to it (*remote filters*). Hence, to store multiple content synopses, we use multi-level Bloom filters. Figure 2.2 (ii) shows an example of a multi-level Bloom filter. Notice that the Bloom filter of each level is not merged but appended to that of the previous level. That approach consumes more memory space to store the Bloom filters, but allows us to estimate the location of a larger number of objects more accurately.

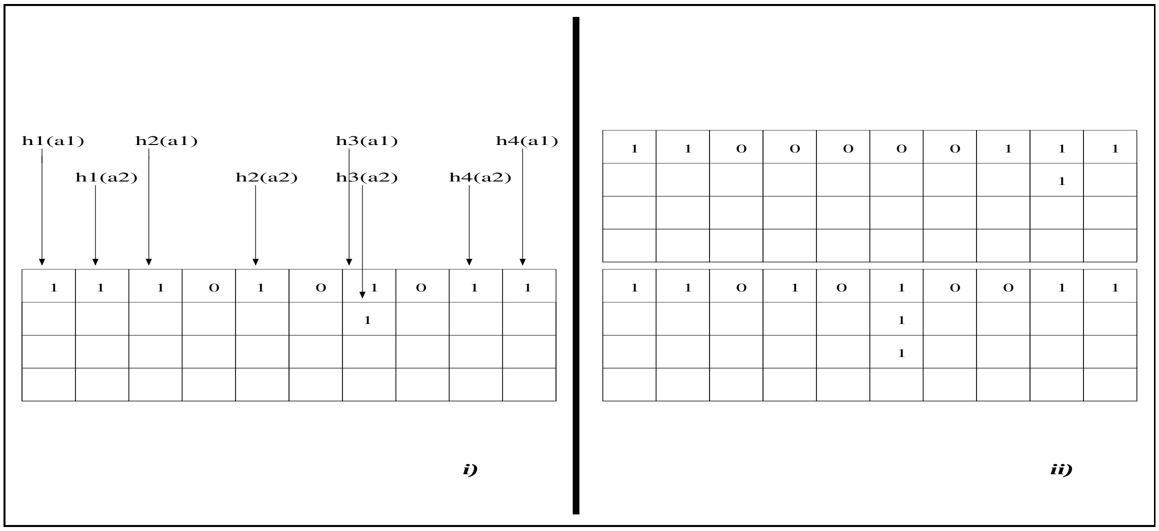

Figure 2.2: (i) Counting Bloom filter example. The counters keeps track of the number of objects that are hashed in the same position. (ii) Multi-level Bloom filter example. The filter of each level is appended to that of the previous level.

2.4 Content-Driven Routing

In our content-driven query routing mechanism each peer stores the content synopses of other peers, and utilizes that information in order to route queries more efficiently. In particular, when a peer receives a query, apart from searching its local content, it also searches the stored content synopses of other peers. If there is no match in its local content, the peer forwards the query only to its immediate peers whose synopses state that they or their neighbors contain the requested object. The pseudocode of the algorithm for checking the content synopsis of a peer is presented in Figure 2.3. Only if the object is not found in any content synopsis, is the query forwarded to a set of random neighbors.

If the query cannot be satisfied locally, the node must decide to which of the peers to propagate it next. Thus, it searches the contents of the stored synopses of remote peers and the query is propagated only to the peers whose synopses indicate that they contain the

```
public boolean checkAllRemoteSynopses(Node conn_node, Document doc) {

   ContentSynopsis remoteConSyn =
      (ContentSynopsis)content_synopsis.remoteBFs.get(conn_node);

   if (remoteConSyn.localBF.isMember(docID) == true) {
      // Local Bloom filter positive
      if (conn_node.hasContent(doc) == false) {
         // Local Bloom filter false positive }
   return true; }

   for (Enumeration e = remoteConSyn.remoteBFs.keys();
      e.hasMoreElements(); ) {
      Node node = (Node)e.nextElement();
      ContentSynopsis ConSyn =
         (ContentSynopsis)remoteConSyn.o_remoteBFs.get(node);
      if (ConSyn.localBF.isMember(docID) == true) {
         // Remote Bloom filter positive
         if (node.hasContent(doc) == false) {
            // Remote Bloom filter false positive }
         return true; } }

   // Else, we have no positives at all: neither local, nor remote.
      return false; }
```

Figure 2.3: The algorithm for choosing peers to forward a query. For each peer that its content synopsis has been stored, its local and remote Bloom filters are checked for matches.

requested object. These are the peers with the highest probability of actually containing the object. If the object is not found in any synopsis, the node forwards the query to a random subset of the immediate peers. To provide a termination condition so that messages are not propagated indefinitely in the network when no objects are found, each message is associated with a time_to_live (TTL) field that represents the maximum number of times the message can be propagated in the network. The TTL value is decreased each time the message reaches a peer. A node that receives a message with TTL zero, stops forwarding the message. Also, if a node receives the same message from two different peers, it discards the duplicate.

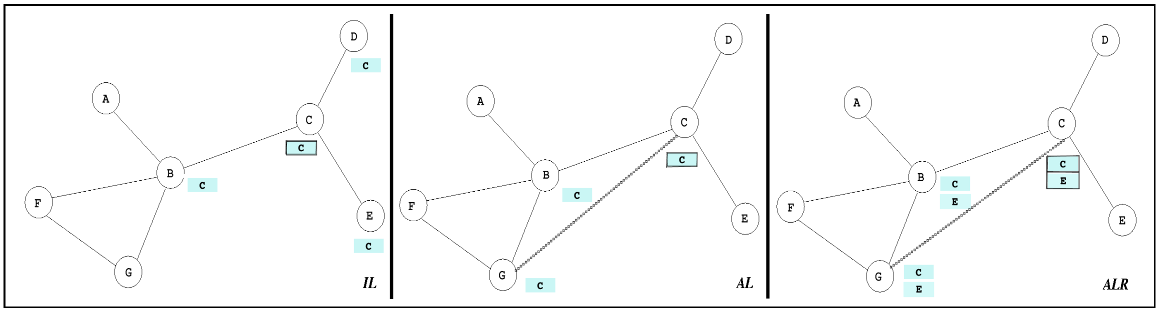

Figure 2.4: An example of the content synopses dissemination strategies IL, AL, and ALR. According to IL, node C propagates only its local synopsis to all its immediate peers (peers one hop away). In AL, C propagates its local synopsis to selected immediate and remote peers. With the latter a direct connection may need to be created. In ALR, C propagates both its local and stored remote synopses to selected immediate and remote peers.

During the system operation, the node keeps statistics about the queries and the replies generated or propagated through the peer. In particular, it keeps track of (1) the number of queries sent by the peer and the replies (query hits) received to its queries from other peers, (2) the number of queries received at the peer and the replies it generates to other peers. These are used to decide to which peers to disseminate a synopsis of the local content of the peer.

2.5 Content Synopses Dissemination Strategies

Since the bandwidth used for transferring content synopses is limited, as well as the space in nodes to store them, each peer selects only some of the other peers to propagate its content synopsis. We have implemented and compared three different strategies for content synopses propagation.

2.5.1 Propagate local content synopsis to immediate peers (Immediate Local – IL)

According to this strategy each peer sends its local content synopsis to all its immediate peers and routes queries by taking into account only the content synopses of its immediate peers. This strategy is simple, but of limited use: Since only a small number of content synopses is examined for the routing decision to be taken, a lot of the queries cannot be directed using the content synopses. The protocol then resorts to randomly choosing peers to further forward the query and thus generates a lot of traffic.

2.5.2 Propagate local content synopsis to peers selected adaptively (Adaptive Local – AL)

Using this more elaborate strategy, each peer sends its local content synopsis to a selection of peers, according to several parameters. Again the routing is done following the synopses of the local content of other peers. The recipients of the content synopsis of a peer are selected not only among its immediate neighbors, but also among remote peers. The adaptive selection of the synopses recipients aims to make the content synopses available to the peers that have a high probability of using them again in the future and yet keep the number of synopses transfers limited. The parameters used to decide to which peers to propagate the content synopses are described in Section 2.6. As the number of synopses used in routing is again limited, this strategy is also often obliged to resort to randomly forwarding queries.

2.5.3 Propagate both local and remote content synopses to peers selected adaptively (Adaptive Local Remote – ALR)

This multi-level strategy differs from the previous, in that the peers propagate and use for their routing decisions not only the synopses of the local content of their immediate peers or peers they have interacted with, but also synopses of the content of remote peers. More specifically when a peer propagates its local content synopsis to other peers, it also propagates the content synopses of remote peers it has stored. Other peers store those remote content synopses together with the local synopsis of that peer and use them to route queries to it. Since each peer stores and propagates remote content synopses of peers it is connected to, it can then easily route queries for content stored in them. Obviously this strategy enables the peers to examine a lot of content synopses before routing a query. Therefore a lot of the queries can be routed accurately and randomly forwarding queries is not used that often. The processing time spent in examining the content synopses is still little. The amount of information transferred between the nodes in order to propagate the remote and the local synopses is higher than in the previous strategies, but still restricted through the use of adaptive selection of the synopses recipients. As already mentioned, the parameters used to decide to which peers to propagate the content synopses are described in Section 2.6.

Figure 2.4 presents an example of the different content synopses propagation strategies discussed above. A point that needs to be made is that content synopses do not necessarily have to be propagated as individual messages, but can rather be piggybacked on the current usage messages (e.g. queries and replies). As already mentioned, all of the above strategies are assuming a simple network infrastructure, where peers route queries through their immediate neighbors. In AL and ALR a more advanced overlay network is built, where peers open or can potentially open direct connections to peers that provide them with good results ("share similar interests with them") and routing can also be based on content synopses of peers outside a node's current horizon. In that case, where interest locality among the peers is exploited, queries can be routed even faster and more accurately, at the cost of managing many –probably short-lived– connections and of storing, processing and propagating a large number of content synopses between many different peers. Allowing "transient" content synopses to traverse the network would be the physical continuation of this approach. Yet, even though the cost of propagating a synopsis may not be too high, in a large-scale system the cost of maintaining up-to-date information throughout the path that a transient content synopsis travels, about where it came from and about how to reach its source would be prohibitive. This would be even more the case for dynamic environments, with frequent topology changes or content updates.

The frequency with which a peer propagates its content synopsis depends on the number of queries it receives. The number of content synopsis messages propagated depends on the thresholds of the several parameters discussed in Section 2.6.

$peer\_id_i$	The peer's globally unique identifier.
$connected\_peers_i$	The list of peers currently connected to this peer.
$object\_list_i$	The list of objects stored locally at the peer.
$queries\_received_i$	The total number of queries this peer has processed.
$searh\_msgs\_received_i$	The number of search messages this peer has received, indexed by the IDs of the query originators.
$local\_hits_i$	The number of local hits generated by queries, indexed by the IDs of the query originators.
$sent\_contentSynopsis\_to_i$	The list of peers that have received a current version of the local content synopsis.

Table 2.1: Parameters of each peer i in the system.

2.6 Adaptive Synopses Dissemination Parameters

Each node in the system is associated with a list of characteristics, which are summarized in Table 2.1.

In order to decide more accurately which peers would benefit from obtaining the content synopses, adapt the selection decision to the current status of the network and thus propagate the content synopses more efficiently, each peer takes into account several parameters. These are used by the AL and ALR propagation strategies.

- The number of queries q_i a node has received by a peer, and their frequency. Peers that have sent a lot of queries to us will most probably make good use of our content synopsis in their routing decisions. A lot of forwarded queries indicate peers that route a lot of traffic. They can use our content synopsis to avoid sending us queries for content we do not have.

- The number of replies r_i a node has provided a peer with, and their frequency. This parameter identifies the popularity of our stored objects among specific peers. Peers that generated a lot of local hits and got a lot of replies by us to their requests will also most probably need our content synopsis in their routing decisions.

- The number of connections $conn_i$ other peers maintain. This parameter identifies the connectivity degree of a peer and is a factor in estimating the average number of messages per time unit this peer may route. A peer that plays the role of a hub in the network, routing many queries, will most probably need the content synopses more.

2.7 Implications of Dynamic Behavior

Since the network is dynamic and self-organizing, nodes may leave or join independently. This especially applies to mobile environments. The system must be able to propagate content synopses to reflect such changes in the connections. Moreover content synopses must be updated whenever an object is added, deleted, or changed in a node's content. Hence, updated content synopses must be generated in two cases:

- When a peer detects an update at the local repository (content changes) of objects (new objects are obtained, existing objects are deleted or new versions of existing objects are created).

- When a peer detects an incoming or withdrawn peer connection (connection establishment or drop).

2.7.1 Content Changes

When the content is updated, a new content synopsis is disseminated by the peer. To minimize the traffic in the network our approach (1) does not generate an update unless the contents of the peers have changed and (2) groups individual Bloom filter updates into group updates to propagate them to the peers. Content synopses are disseminated due to both local and remote content changes.

2.7.2 Connection establishment

According to the content synopses propagation strategy being followed, a newcomer may receive content synopses from its neighbors immediately, or adaptively during operation. The same applies to the newcomer's decision to propagate its own content synopsis.

Thus, following a push model, a peer would choose to push its content synopsis to other peers as soon as it is connected. This will result in the other peers replying with their content synopses. Since peers keep track of where they have sent their content synopsis, duplicates in synopses propagation are avoided. This is the default method of synopses propagation. Yet we also discuss a more passive method in the next paragraph, suitable for peers with very short connection times.

Following a pull model, a peer would choose to ask for the content synopses of other peers only when it needs to search for something or route a query. This approach would result in extraneous traffic for explicitly asking for the content synopses, but it might prove useful in highly dynamic environments. In the case of fast moving mobile users for example, it might make more sense to allow them to explicitly pull synopses they will need, instead of bombarding them with synopses of different neighborhoods as they move around.

2.7.3 Connection drop

When a peer permanently disconnects from the network, neither the content synopses of other peers stored in it, nor its content synopsis stored in other peers will be useful anymore. Its immediate peers will sense the disconnected peer and all the relevant content synopses will be removed after a time threshold t_r. In addition, a `DISCONNECTED` message will be sent to the non-immediate peers to remove their corresponding content synopses.

Chapter 3

Experimental Evaluation

In this chapter we present a detailed experimental evaluation of our mechanisms. Section 3.1 describes our simulation environment, while Section 3.2 summarizes the characteristics of the different query routing protocols we compared. Section 3.3 presents the performance metrics we used, Section 3.4 discusses our results, and finally Section 3.5 summarizes the benefits of our technique.

3.1 Simulation Infrastructure

To investigate the characteristics of our adaptive content-driven routing mechanism in detail we have implemented an unstructured peer-to-peer network using the Gnutella [18] P2P communication protocol. In order to be able to evaluate systems of thousands of peers, we have used the Neurogrid simulator [20]. Our implementation of the adaptive content-driven routing protocol was done in approximately 3500 lines of Java code. The parameters used

Node Parameters	Number of nodes	Varying
Network Parameters	TimeToLive of query messages	7
	Initial number of connections per node	3
	Minimum number of connections per node	3
	Maximum number of connections per node	10
	Network topology	Random
Content Parameters	Size of pool of available objects	2000
	Number of objects per node	30
	Distribution of objects over nodes	Uniform
Bloom Filter Parameters	Size of filter, in bits	10
	Number of hash functions	4
	Size of counter for each position, in bits	4
Simulation Parameter	Number of averaged measurements	20
	Number of searches per experiment	400

Table 3.1: Simulation settings.

in the simulation are presented in Table 3.1. We chose the network size to vary up to 3000 nodes, an estimate of the number of concurrently active nodes in a university campus.

In our implementation we used counting, multi-level Bloom filters. To create the hash functions, used in generating the Bloom filters, similarly to [43], we took advantage of a cryptographic message digest algorithm (SHA-1 [30]) and of its property of pseudo randomness. More specifically, we used SHA-1 to hash strings of arbitrary length, representing the peers' content, to 160 bits. We then built the hash functions by dividing the SHA-1 output into smaller sets of bits.

Our average results are derived from 20 measurements and each one of those is averaged from 20 searches. In other words, each experiment run includes 400 searches in total. The peers' content is chosen from 2000 sample objects, of which 100 are randomly selected to be search targets. To demonstrate locality of interests, different peers in the same vicinity may query for the same sets of objects.

3.2 Overview of Protocols

We ran simulations to compare the different strategies that use content synopses to route query messages. Those strategies were described in Section 2.5 and are briefly summarized in Table 3.2. We also compared our strategies to a traditional Breadth-First Search (BFS) algorithm. Even though BFS is not directly comparable to our content-driven routing protocols, we chose to present it here to illustrate the differences and the relative gain from our adaptive propagation schemes.

Protocol	Query Routing	Synopses Propagated	Synopses Recipients
IL	Content-driven routing	Local content synopses	All immediate peers
AL	Adaptive content-driven routing	Local content synopses	Selected immediate and remote peers
ALR	Adaptive content-driven routing	Local and remote content synopses	Selected immediate and remote peers
BFS	Flooding all immediate peers	–	–

Table 3.2: The query-routing protocols we compare in the experiments.

3.3 Performance Metrics

We introduced a number of metrics to evaluate both the utilization of the system's resources, and the efficiency of the query routing algorithms. Moreover we include metrics specifically for the comparison of the content-driven protocols and the accuracy of the Bloom filters.

Hence, the metrics we used to compare the searching algorithms were:

1. **Average Message Transfers.** The average number of query messages sent during a search. This metric indicates how efficiently the network bandwidth is used.

2. **Average Nodes Reached.** The average number of nodes reached during a search.

This metric shows how many nodes are contacted to provide results to a query and is therefore an indication of the efficiency of the search algorithm in terms of bandwidth and processing power usage.

3. **Average Recall Efficiency.** Recall is defined as the proportion of all possible matches to a search that were actually discovered. Recall efficiency is defined as the ratio of recall against the number of query messages that have travelled through the network during that search. Therefore the recall efficiency average is an indication of the usefulness of the query messages that are propagated.

The metric we used to measure the accuracy of the Bloom filters was:

1. **False Positives.** This is the number of incorrect reports by Bloom filters; stating that an object is stored in a peer, when it actually isn't.

Finally, the metrics specifically pertaining to the content-driven protocols were:

1. **Synopses Hits/Misses Ratio.** This is the ratio of the content synopses hits against the content synopses misses. It shows how many of the queries could be routed based on the Bloom filters and is therefore an indication of the usefulness of the content synopses.

2. **Filter False Positives/Total Positives Ratio.** This is the ratio of the Bloom filter false positives against the total number of Bloom filter positives. It indicates how many of the queries were falsely routed based on the Bloom filters, over the total number of

queries that were routed based on them. Therefore it measures the accuracy of content-driven routing.

3. **Total Content Synopses Messages.** This is the total number of messages sent for content synopses propagation. It measures the content-driven routing protocol overhead (cost).

4. **Total Query Messages.** This is the total number of query messages propagated. It measures the efficiency of the network bandwidth usage.

3.4 Analysis

In this Section we present a detailed discussion of the experimental results. In 3.4.1 we investigate the optimal values for the Bloom filter parameters, while in 3.4.2 we compare our adaptive content-driven routing to flooding-based search. In 3.4.3 we compare the different content synopses dissemination strategies to each other in more detail. Finally, in 3.4.4 we investigate the accuracy of adaptive content-driven routing in highly dynamic environments.

3.4.1 Bloom Filter Parameters

As already shown in Section 2.3, there exists a tradeoff in the representation of objects through Bloom filters. Three different parameters may affect the accuracy of the representation, in other words the number of false positives yielded: The size of the Bloom filter in bits (memory overhead), the number of hash functions used (computation overhead), and the

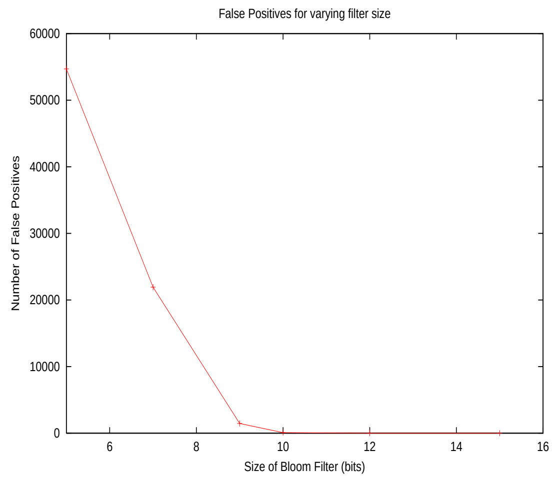

Figure 3.1: Bloom filter false positives for varying size of the filter (in bits).

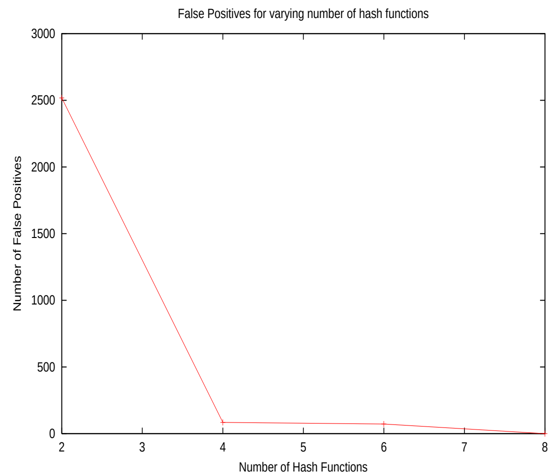

Figure 3.2: Bloom filter false positives for varying number of hash functions.

number of objects to be represented. We investigated the optimal values for those parameters in our first set of experiments, by varying each one of them while keeping the others constant. We used counting Bloom filters with 4-bit counters, the simplest content-driven routing algorithm (IL), 4000 possible objects, and 1000 nodes and focused on the number of false positives.

Effect of filter size to the number of false positives. As Figure 3.1 shows, filter size can greatly affect the number of false positives. Small filter sizes can result to thousands of false positives. However false positives are virtually eliminated above 10 bits (when representing 30 objects per filter and using 4 hash functions).

Effect of number of hash functions to the number of false positives. As Figure 3.2 shows, the number of false positives greatly decreases when using 4 hash functions or more (when representing 30 objects per filter and using 10 bits for the filter size).

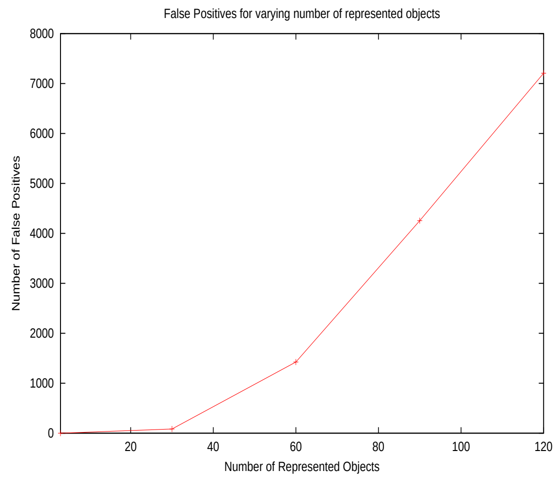

Figure 3.3: Bloom filter false positives for varying number of represented objects per node.

Effect of number of represented objects to the number of false positives. Figure 3.3 shows that when using 4 hash functions and Bloom filters of 10 bits size, not more than 30 objects can be represented by a filter without significant loss in accuracy.

Taking into account the above results, we decided to use Bloom filters 10 bits long, 4 hash functions, and 30 objects per node (chosen out of 2000 unique objects)[1] for the rest of the experiments that use content synopses.

3.4.2 Comparison of Content-Driven Routing and Breadth-First Search

In our second set of experiments we compared content-driven query routing to traditional flooding-based search.

Average message transfers during a search. Figure 3.4 shows that content-driven rout-

[1]Hence when the number of nodes ranges from 10 to 3000, the total number of objects ranges from 300 to 30000 and the replication degree ranges from 0.15 to 15.

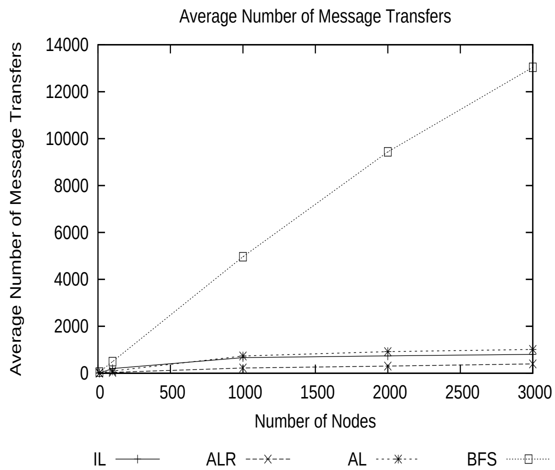

Figure 3.4: Average number of query messages sent during a search for varying network size.

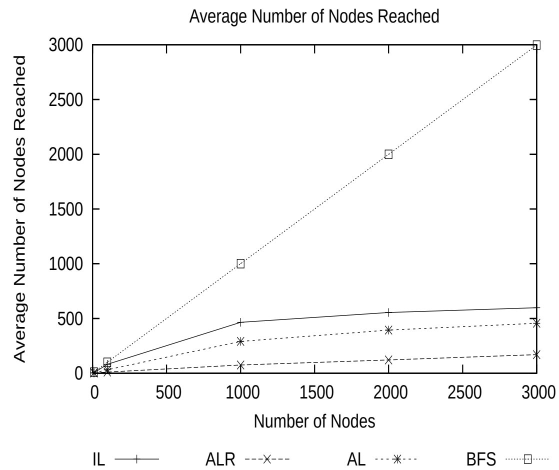

Figure 3.5: Average number of nodes reached during a search for varying network size.

ing drastically decreases the number of query messages transferred during a search. As the number of nodes increases, the number of message transfers grows dramatically in flooding-based BFS, while the content-driven routing mechanisms manage to keep the message transfers almost at a fixed level. Thus, by using the network bandwidth efficiently, content-driven routing is therefore able to scale to thousands of nodes. ALR, by propagating content synopses of both local and remote peers adaptively, achieves the minimum number of message transfers needed to answer a query. It is noteworthy that the decrease in query messages between ALR and BFS reaches 97%.

Average number of nodes reached during a search. Figure 3.5 again shows the benefits of content-driven routing in terms of bandwidth and processing power usage efficiency. All the content-driven routing techniques are able to provide query hits by contacting more than one order of magnitude less peers than BFS, which contacts a lot of peers unnecessarily.

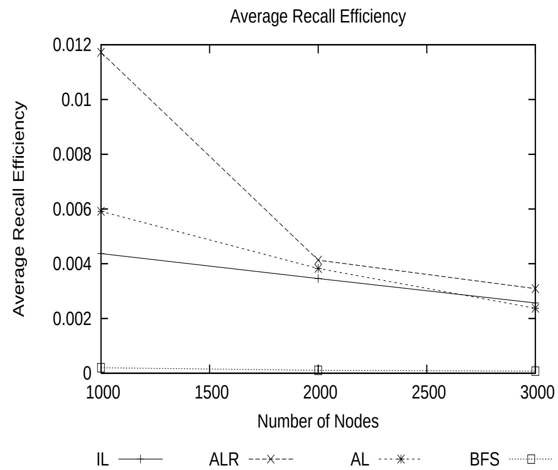

Figure 3.6: Average proportion of possible matches discovered over the average number of query messages transferred during a search for varying network size.

Moreover the content-driven routing strategies keep the number of reached nodes at an almost constant level, while the nodes that are reached with BFS grow linearly as the total number of nodes increases. The Figure shows that the adaptive AL and ALR techniques guide queries more efficiently than the simplistic IL, in which content synopses are propagated blindly to all immediate peers. ALR is again the most efficient and scalable technique of all, due to the adaptive use of the multi-level Bloom filters.

Average Recall Efficiency during a search. Figure 3.6 shows the value of the query messages that are propagated during a search, in terms of their contribution to the discovery of possible matches. Even though the flooding of BFS is able to discover a lot of matches, the cost of query messages transferred results in its low recall efficiency. ALR again has the highest recall efficiency, followed by the other adaptive content-driven routing strategy, AL. The reason is that adaptive content synopses propagation places the Bloom filters where they

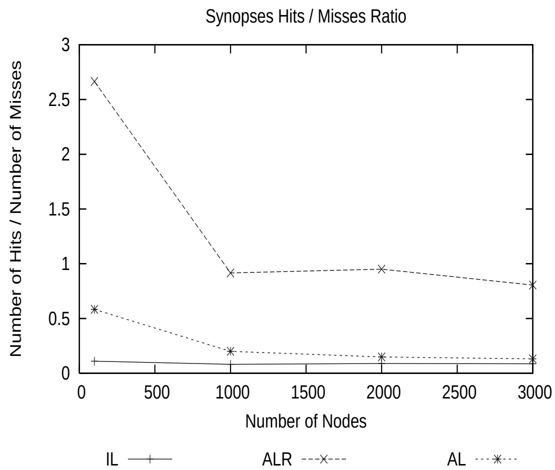

Figure 3.7: Content synopses hits over misses for varying network size.

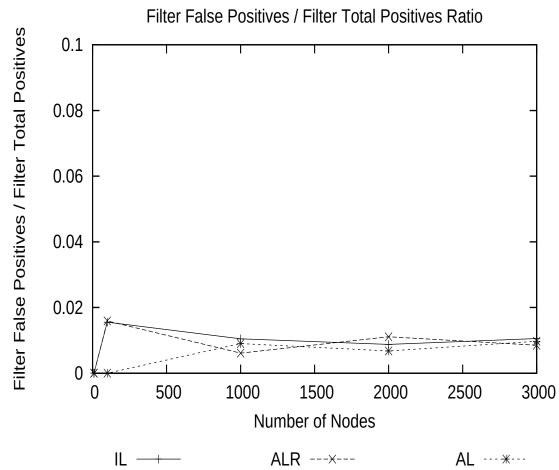

Figure 3.8: Bloom filter false positives over total positives for varying network size.

are more likely to be needed, achieving better performance than the blind IL. As the number of nodes grows, the proportion of the total matches discovered by the content-driven routing mechanisms decreases, since the queries are guided, in order to contact a small number of nodes and to produce a small number of messages.

3.4.3 Comparison of the Content-Driven Routing Protocols

In our third set of experiments we compared the different content-driven routing protocols to each other in more detail

Content synopses hits over misses. Figure 3.7 shows how much the query routing actually benefits from the use of the content synopses. We notice that simply placing content synopses of local content to immediate neighbors (IL) is useful for routing only about 10% of the queries. On the other hand, adaptively placing content synopses (AL and ALR) improves

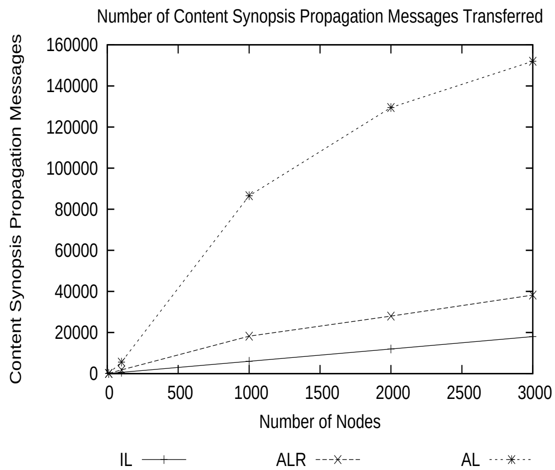

Figure 3.9: Total number of content synopses propagation messages transferred for varying network size.

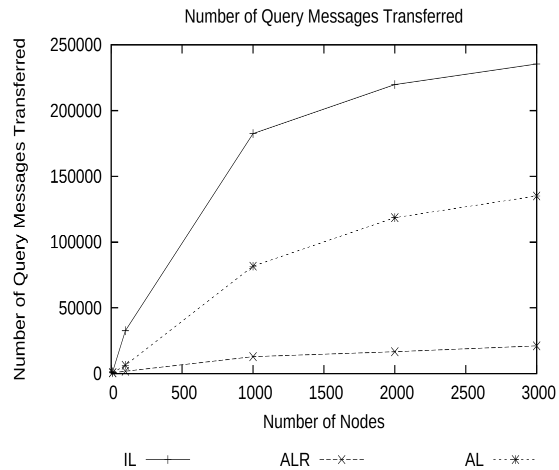

Figure 3.10: Total number of query messages transferred for varying network size.

their usefulness to 20% for AL and to 90% for ALR. By propagating local and remote content synopses, ALR manages to drastically decrease the number of Bloom filter misses and achieves a hits/misses ratio close to 1, meaning that half of the queries can be routed based on the content synopses.

False positives over total positives. Figure 3.8 shows that content-driven routing is extremely accurate. For all three routing strategies that use content synopses only a very small percentage (around 1%) of the total queries that are routed based on them is falsely routed, due to Bloom filter false positives. Thus, our choice of the Bloom filter parameters allowed us to minimize the false positives.

Total content synopses messages. Figure 3.9 shows the relative cost of the different content-driven routing protocols, in terms of content synopses propagation messages. By simply propagating content synopses only to immediate peers, IL keeps the protocol overhead low. However the usefulness of the content synopses in that approach is limited, as Figure 3.7 indicates. AL on the other hand has to propagate a lot of content synopses for them to be useful in query routing. ALR, by adaptively propagating local and remote content synopses, manages to route queries effectively and yet keep the protocol overhead at a reasonable level, even as the number of nodes increases. That overhead is acceptable, if one takes into account the drastic saving of query messages ALR achieves. Thus, combining several content synopses in one message, as ALR does, reduces significantly their dissemination overhead.

Total query messages. Figure 3.10 shows the savings in query messages adaptive strategies achieve. Especially ALR, by guiding queries through the use of local and remote content synopses, manages to keep the number of query messages low and easily scale to thousands of nodes. Bandwidth is thus used more efficiently in ALR than in any other of the content-driven routing protocols. ALR reduces the number of query messages by utilizing a lot of content synopses and placing them intelligently in the network. Notably, ALR decreases the number of query messages transferred by half an order of magnitude compared to AL and by one order of magnitude compared to IL.

3.4.4 Adaptive Content-Driven Routing in Dynamic Environments

In our fourth set of experiments we evaluated our protocols in a mobile environment, where peers leave the network dynamically. We gradually disconnected peers throughout the experiment run and we conducted experiments for disconnections reaching 10, 20, and 30% of the

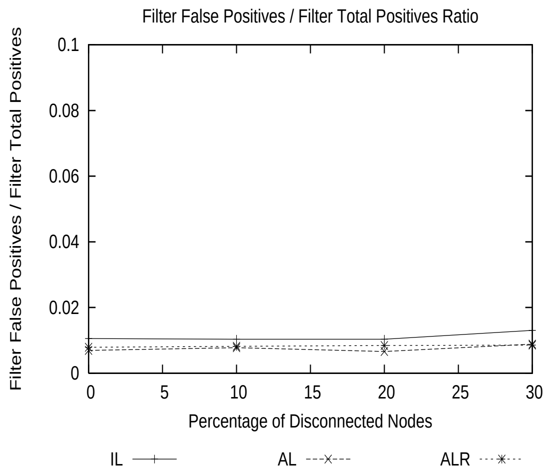

Figure 3.11: Bloom filter false positives over total positives for varying percentage of disconnected nodes.

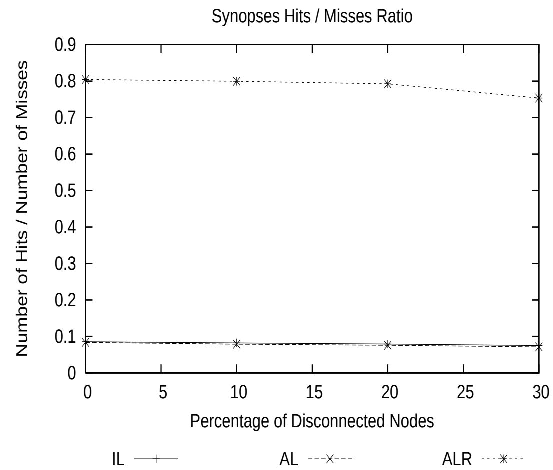

Figure 3.12: Content synopses hits over misses for varying percentage of disconnected nodes.

total number of peers, which was initially 3000. We report the effects of the disconnections on the Bloom Filter behavior.

False positives over total positives. Figure 3.11 shows that content-driven routing remains very accurate even when a lot of peers disconnect. The neighbors of a leaving peer realize the disconnection and update their summaries, while peers further away also update their synopses when they are notified by a `DISCONNECTED` message they receive from the immediate peers. Hence false positives are not increased by the peer disconnections.

Content synopses hits over misses. Figure 3.12 shows that peer disconnections do not considerably affect the success of the synopses in query routing either. ALR, which is the most aggressive mechanism in synopses dissemination, often routes queries successfully using the summaries. When a lot of peers disconnect, less synopses are available to help in query routing, hence a small degradation in the hit ratio.

3.5 Summary of Results

By comparing the different content-driven routing protocols for a variety of performance metrics, and also by evaluating their characteristics in comparison to a very different but common approach (BFS), we were able to quantify our claims regarding the advantages of adaptive content-driven routing:

1. Counting Bloom filters can provide a very accurate representation of the peers' content. Less than 1% of the queries are incorrectly routed due to false positives.

2. Content-driven routing drastically decreases the number of query messages transferred during a search, allowing very efficient use of the network bandwidth. When adaptively propagating local and remote content synopses, the savings in query messages can reach 97%, compared to a flooding-based mechanism and yet maintain high recall efficiency.

3. Content-driven routing can answer queries by contacting more than one order of magnitude less peers than a flooding-based mechanism, allowing efficient use of the nodes' processing power.

4. Content-driven routing is able to scale to thousands of peers. As the network size grows, the routing mechanism can still provide query hits, while keeping the number of query messages and nodes contacted during a search at almost constant levels.

5. Content-driven routing remains accurate and robust even in highly dynamic environ-

ments. Content synopses continue to accurately summarize peers' content and remain useful for query routing even after a lot of peer disconnections.

6. Adaptive content-driven routing enables us to make much better use of the content synopses than blind content synopses propagation. Multi-level Bloom filters that are placed strategically in the network are able to guide queries almost 5 times more often than simple Bloom filters which are propagated blindly to the immediate peers only.

7. In the adaptive content synopses propagation strategies, multi-level Bloom filters are able to guide queries approximately 4 times more often than simple Bloom filters are and yet keep the total number of content synopses propagation messages 4 times lower than when using simple Bloom filters.

8. By propagating local and remote Bloom filters adaptively query messages can be decreased by half an order of magnitude compared to adaptive propagation of just local Bloom filters and by one order of magnitude compared to blind propagation of local Bloom filters to all immediate peers. Thus, adaptive local and remote content synopses propagation offers the best performance of all compared strategies in terms of scalability, bandwidth usage, processing power usage, and recall efficiency.

Chapter 4

Related Work

In this chapter we review related efforts in query routing (Section 4.1) and data dissemination (Section 4.2) in overlay networks.

4.1 Query Routing

Several mechanisms have been proposed to facilitate searching in peer-to-peer networks [24, 27, 37]. In this Section we discuss the major approaches and their relevance to our content-driven routing architecture.

4.1.1 Unstructured Overlays

Breadth-First-Search

The initial version of Gnutella, employed a simple flooding-based query routing protocol (bounded Breadth-First-Search). Without imposing any structure on the system, peers would randomly connect to other peers and propagate queries to their neighbors within a certain radius. Building upon this protocol several efforts have focused on improving the efficiency and scalability of searching in unstructured overlays.

Super-Peers

QRP (Query Routing Protocol) of RFC-Gnutella 0.6 [18] employs ultra-peers to filter queries and only forward them to the leaf nodes that are most likely to have a match. This filtering is done by looking the query words through a hash table that is sent by the leaf node to its ultrapeer. Similar is the approach followed by FastTrack, a proprietary protocol used by KaZaA [22] and other file sharing applications: Super-peers with higher networking, storage, and processing capabilities volunteer to maintain meta-data for files located in regular peers. This way queries have to travel only through a network backbone before they reach nodes that can offer results.

Random Walks

Random walks [25] have been another suggested alternative to query flooding. In this approach, a peer randomly forwards its query to k of its neighbors. Each of these peers forwards the query to one of its neighbors and by repeating this process k random walks through the network take place. Similarly to that technique, the protocol proposed in [21] allows peers to propagate queries to k random neighbors. This way the number of walks increases exponentially. In [5] biased random walks are combined with flow control and topology adaptation to take into account the heterogeneity of the peers

Query Caching

Efforts on utilizing the previous queries and their replies have also been made. In [46] the keywords of the queries are used to compute the similarity of the query message to previously seen queries, to probabilistically forward the query to only a subset of the nodes. This technique relies on knowledge collected locally at the peer by monitoring the messages propagated in the network, while in our approach summaries of the actual content of other peers are disseminated. Caching the results of queries, while arbitrarily partitioning a network in layers is proposed in [45]. In addition to a local index, that keeps indices of local files, each peer maintains a response index, which caches the query results that flow through the peer. While this work also aims at reducing search traffic, the approach followed focuses on query caching and not on content summarization.

Routing Indices

In [8] Routing Indices are proposed as a means to guide queries towards the direction of the requested object. Each peer maintains statistics which indicate how many objects are reach-able through each neighbor (compound routing index). Improving this basic protocol, the number of hops required to reach an object (hop-count routing index) and the cost of storing different routing indices (exponentially aggregated routing index) can be taken into account. In our approach, Bloom filters provide a more precise content summarization mechanism which should enable more efficient query routing.

Depth-First-Search

Censorship resistance has been the focus of the creators of Freenet [6]. Each peer maintains a routing table of addresses of other peers and of keys of the objects they are storing. Using these routing tables a bounded Depth-First-Search takes place, combined with caching of the retrieved objects in intermediate nodes. This way anonymity and redundancy are achieved.

Centralized Indices

Similar to the original Napster [29] peer-to-peer file-sharing application, BitTorrent [2] relies on a centralized database for locating data objects. Unlike Napster though, this central location (tracker) allows a peer to retrieve pieces of the same data object concurrently from different peers. To punish free-riding, peers prefer to cooperate with peers they have received responses from (tit-for-tat).

4.1.2 Structured Overlays

Structured overlay networks [26, 32, 39, 42, 47] handle location and routing as a single problem and impose a structure in the system by mapping the objects to particular nodes. Also referred to as "Distributed Hash Tables (DHTs)", structured overlays employ different algorithms to assign object keys to nodes to guarantee key retrieval in logarithmic time. Chord [42] uses consistent hashing and places the node IDs in a virtual ring. CAN [32] suggests a multi-dimensional node ID coordinate space and maps keys in this space using uniform hashing. Tapestry [47] and Pastry [39] employ a Plaxton-style global mesh network and locate a key in steps, by matching it with the suffix or prefix of the node ID respectively. Finally Kademlia [26], when trying to match a key to a node ID, utilizes the XOR metric to calculate the distance in the key space.

Even though structured overlays achieve object retrieval in bounded time, they have traditionally been inherently limited in other ways [4, 5]: They do not support complex keyword-based queries without constraints on data placement, they do not take peer heterogeneity into account, and do not handle robustly network dynamics, like massive peer arrivals, departures, or failures. Several efforts have been made to address all of the above issues [4]. Keyword-based searching has been made feasible by maintaining inverted indices that map keywords to objects [34]. In this approach the partitioning is vertical, meaning that each node maintains pointers to all the objects that contain a specific keyword. Bloom filters are used to reduce the bandwidth required to answer "AND" queries (which need the cooperation of more peers' incremental results to be answered), and to cache object lists. To address peer heterogeneity the virtual hosts approach is used, according to which a node participates in the peer-to-peer system as several logical hosts, proportional to its request processing capacity.

In our work we focus on unstructured, dynamic, self-organizing networks, in which peers can decide locally what objects to store and can join the system at random times and leave it without a priori notification.

Similar to our approach, Rhea and Kubiatowicz [36] propose a probabilistic location protocol based on attenuated Bloom filters, which improves the latency of locating files. Again, the difference from our mechanism is that they place the objects to specific nodes based on some keys and use these keys to route the requests to the nodes. Furthermore, we investigate different algorithms for disseminating the content synopses. Aspnes et al [1] have shown that there is no need for such global coordination in the network. Our approach has the advantage that it does not impose any structure; we assume that the system is self-organizing, driven only by decisions made locally at the peers.

4.2 Data Dissemination

Our work builds upon [28] and [31]. In [28] the concept of content summarization was introduced, while in the current work we focus on mechanisms for the dissemination of the content synopses, we present more elaborate summarization techniques and discuss performance in highly dynamic environments. One of our criteria for the adaptive selection of the content synopses recipients is the notion of interests, explained in [31]. In this Section we discuss relevant research efforts on data dissemination in peer-to-peer and large-scale distributed systems.

4.2.1 Anti-Entropy Protocols

Planet-P [9] locates objects by replicating globally two data structures: A membership directory and a compact content (term-to-peer) index. Members gossip about changes to keep these data structures updated and loosely consistent. Gossiping is done by pushing rumors to random peers and by pulling information from random peers. A content ranking algorithm based on the vector space ranking model is also used, to find only highly relevant documents to a query. The set of terms in each peer's local index is summarized using a Bloom filter. The global index is used to find peers that have a term, and then the local index is used to return the specific documents. The cost of storing and maintaining the global data structures makes the system unsuitable for users with modem-speed connections, low storage capabilities, or for networks of more than some thousand peers. Our mechanism on the other hand does not rely on any global knowledge of the network and thus as minimum overhead and no need for structure.

Rumor spreading algorithms have been proposed, that offer probabilistic guarantees, instead of ensuring strict consistency [44]. For example, P-Grid [11] uses a hybrid push/pull rumor dissemination algorithm. A new update is pushed by the initiator to a subset of peers that are affected by it, because they have the original version of the data item, and is further propagated by them. Peers that have been disconnected, that have not received updates for a long time, or that have received a pull request but are not sure if they have the latest update, pull updates from one or more other peers. Two parameters, the probability of forwarding an update, and the fraction of the total replicas to which peers initially decide to forward an update, are being considered for spreading the rumors. The protocol utilizes P-Grid's network infrastructure to route messages.

4.2.2 Meta-Data Caching

CUP (Controlled Update Propagation) [38] is used for maintaining caches of meta-data for locating content. A node receives and propagates updates based on personal economic incentives. The investment return is secured when a node can answer queries using the stored meta-data, instead of having to further forward them. Each node decides whether to register for receiving and propagating updates for an item according to popularity (based on the number of queries received for that item)-based incentives, either probabilistic, or log-based, also taking into account its workload and/or network connectivity. Our Bloom filter-based approach focuses on large-scale, unstructured networks.

4.2.3 Hierarchical Data

Breadth and Depth Bloom filters have also been used for summarizing hierarchical data structures [23]. These however focus on specific data structures, such as XML documents and assume a hierarchical network organization.

4.2.4 Web-Caching

The Bloom Filter mechanism has also been used in Summary Cache [15] in the context of web-caching. The authors have shown that Bloom filter representations are economical and reduce the bandwidth consumption in the network.

4.2.5 Streaming Data

Work has also been done on filtering and disseminating streaming data [41], where data repositories are organized hierarchically according to their coherency requirements, as well as on overlay topologies for routing real-time media streams between some publishers and many subscribers [17]. In [41], in order to provide updates of highly dynamic, streaming, and aperiodic data, an organization of data repositories is proposed. The repositories are organized hierarchically, with those that have the highest coherency requirements placed closer to the data source. Data updates are pushed down that hierarchy, only to the repositories that require them (according to their coherency requirements). Repositories are placed in a way that their coherency requirements are *just* met, so that repositories with more stringent coherencies end up serving repositories with more loose coherencies. Back-up parents are used to handle repository or communication link failures. Active back-up parents deliver data with less stringent coherency, reducing the overhead of providing resiliency and enabling the detection of the failure.

4.2.6 Data Replication

Several efforts have focused on techniques for replicating data in peer-to-peer networks. In [7], different replication strategies are evaluated, and an optimal is found between two extremes, a uniform and a proportional, which offer the worst performance. In [13] load balancing in unstructured peer-to-peer networks is achieved by object replication. The Fairness Index of the distribution of the load across the peers is used to drive the load balancing decisions. The problem we consider differs in that we focus on disseminating pointers to the data instead of the actual data. Our goal is to enable efficient object retrieval rather than alleviating data serving hotspots.

Chapter 5

Conclusions and Future Work

In this work we have presented mechanisms for adaptive data dissemination and content-driven routing of queries in large-scale, unstructured overlay networks. Based on content synopses, nodes can forward queries intelligently only to their peers that are highly probable to provide replies. By propagating the synopses adaptively we have shown how they can be strategically placed in the network, where they are most probably going to be needed. We have simulated large-scale overlays of thousands of peers and also verified the robustness of our mechanism under dynamic peer disconnections. We have compared our content-driven routing mechanism to traditional flooding-based searching to find out tremendous savings in query messages. Thus, our approach is scalable and highly efficient in terms of bandwidth and processing power usage. We have compared three different synopses propagation strategies. Our results show that adaptive propagation of local and remote synopses performs much better than blind propagation to immediate peers, or just local synopses propagation.

Our future work includes the comparison of our push-based protocol to the analogous pull-based, as well as investigating the construction of overlays to efficiently propagate content synopses. Moreover, taking into account more parameters when deciding where to propagate the content synopses and experimenting with the synopses propagation depth might also be interesting. Finally, we plan to investigate in detail the behavior of content-driven routing when built on top of message routing protocols for mobile ad hoc networks [19].

Bibliography

[1] J. Aspnes, Z. Diamadi, and G. Shah. Fault-tolerant routing in peer-to-peer systems. In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing, PODC*, July 2002.

[2] BitTorrent Home Page. http://www.bittorrent.com/, 2001.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[4] M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation, NSDI*, May 2005.

[5] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *Proceedings of ACM SIGCOMM 2003*, August 2003.

[6] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009, 2001.

[7] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of ACM SIGCOMM 2002*, August 2002.

[8] A. Crespo and H. Garcia-Mollina. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems, ICDCS*, July 2002.

[9] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, HPDC*, June 2003.

[10] F. Dabek, B. Zhan, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common api for structure peer-to-peer overlays. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS), Berkeley, CA, USA*, February 2003.

[11] A. Datta, M. Hauswirth, and K. Aberer. Updates in highly unreliable, replicated peer-to-peer systems. In *Proceedings of the 23rd International Conference on Distributed Computing Systems, ICDCS*, May 2003.

[12] distributed.net Project. http://www.distributed.net/, 2005.

[13] Y. Drougas and V. Kalogeraki. A fair resource allocation algorithm for peer-to-peer overlays. In *Proceedings of the 8th Global Internet Symposium*, March 2005.

[14] K. Eshghi. Intrinsic references in distributed systems. Technical Report HPL-2002-32, HP Labs, 2002.

[15] L. Fan, P. Cao, J. Almeida, and A.Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. In *Proceedings of ACM SIGCOMM'98*, August 1998.

[16] Folding@home Project. http://folding.stanford.edu/, 2005.

[17] G. Fry and R. West. Adaptive routing of qos-constrained media streams over scalable overlay topologies. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, May 2004.

[18] Gnutella Protocol Development. http://rfc-gnutella.sourceforge.net/, 2003.

[19] H. Hu, J. Xu, W.S. Wong, B. Zheng, D.L. Lee, and W.C. Lee. Proactive caching for spatial queries in mobile environments. In *Proceedings of the 21st International Conference on Data Engineering, ICDE*, April 2005.

[20] S. Joseph. An extendible open source P2P simulator. *P2P Journal*, pages 1–15, November 2003.

[21] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *Proceedings of the 11th International Conference on Information and Knowledge Management, CIKM*, November 2002.

[22] KaZaA Home Page. http://www.kazaa.com/, 2001.

[23] G. Koloniari and E. Pitoura. Content-based routing of path queries in peer-to-peer systems. In *Proceedings of the 9th International Conference on Extending DataBase Technology, EDBT*, March 2004.

[24] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, Second Quarter 2005.

[25] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th ACM International Conference on Supercomputing, ICS*, June 2002.

[26] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems, IPTPS*, February 2002.

[27] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57, HP Labs, 2003.

[28] A. Mohan and V. Kalogeraki. Speculative routing and update propagation: A kundali centric approach. In *Proceedings of the 2003 IEEE International Conference on Communications, ICC*, May 2003.

[29] Napster Home Page. http://www.napster.com/, 1999.

[30] National Institute of Science and Technology. Secure Hash Standard (SHA1). Federal Information Processing Standard (FIPS) 180-1, April 1995.

[31] M. K. Ramanathan, V. Kalogeraki, and J. Pruyne. Finding good peers in peer-to-peer networks. In *Proceedings of the 2002 International Parallel and Distributed Computing Symposium, IPDPS*, April 2002.

[32] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[33] T. Repantis and V. Kalogeraki. Data dissemination in mobile peer-to-peer networks. In *Proceedings of the 6th International Conference on Mobile Data Management, MDM*, May 2005.

[34] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *Proceedings of the 4th International Middleware Conference*, June 2003.

[35] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. In *Proceedings of the USENIX Annual Technical Conference*, June 2004.

[36] S. Rhea and J. Kubiatowicz. Probabilistic location and routing. In *Proceedings of IEEE INFOCOM 2002*, June 2002.

[37] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: Search methods. Technical Report UNSW-EE-P2P-1-1, University of New South Wales, Sydney, Australia, September 2004.

[38] M. Roussopoulos and M. Baker. CUP: Controlled update propagation in peer-to-peer networks. In *Proceedings of the 2003 USENIX Annual Technical Conference*, June 2003.

[39] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001.

[40] SETI@home Project. http://setiathome.ssl.berkeley.edu/, 2005.

[41] S. Shah, S. Dharmarajan, and K. Ramamritham. An efficient and resilient approach to filtering and disseminating streaming data. In *Proceedings of the 29th International Conference on Very Large Data Bases, VLDB*, September 2003.

[42] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[43] The XLattice Project. http://xlattice.sourceforge.net/, 2005.

[44] R. van Renesse, K.P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, 2003.

[45] C. Wang, L. Xiao, Y. Liu, and P. Zheng. Distributed caching and adaptive search in multilayer p2p networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems, ICDCS*, March 2004.

[46] D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos. Information retrieval in peer-to-peer systems. *IEEE CiSE Magazine, Special Issue on Web Engineering*, pages 12–20, July/August 2004.

[47] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, January 2004.