MULTILAYER GNUTELLA P2P Resource Sharing with an Efficient Flexible Multi-Keyword Search Facility

Sebnem Oeztunali Steffen Rusitschka Alan Southall

Siemens AG, Corporate Technology
Intelligent Autonomous Systems
firstname.lastname@siemens.com

Abstract— **In Peer-to-Peer networks, multi-keyword searching with wildcard support represents a difficult problem in which many trade-offs between efficiency and flexibility must be made. In Distributed Hash Tables, algorithms such as Squid must overcome the complexity of mapping a multi-dimensional address space onto a one-dimensional address space whilst limiting or avoiding the amount of traffic and hot-spots that are generated. Unstructured P2P systems, such as Gnutella, support multi-keyword searches with wildcards, but sacrifice bandwidth and fail to reliably locate rare resources. In this paper, we will present a Peer-to-Peer protocol which supports flexible, wildcard enabled, multi-keyword queries which reliably find both rare and popular resources in an efficient manner.**

I. Introduction

In resource sharing Peer-to-Peer (P2P) applications, the search facility is one of the most important aspects of the user interface. Users commonly query the system with keywords, often including wildcards, describing the resource they want to retrieve, e.g. first name and last name, and expect reasonable results within an acceptable period of time. For the acceptance of the system it is crucial that multi-keyword searches are carried out efficiently.

During the processes of developing P2P based Voice over Internet Protocol (VoIP) systems, we created Multilayer Gnutella which uses simple textual keywords to describe and search for the resources in the network. The Multilayer Gnutella protocol creates an overlay topology which represents the keyword associations between resources in the system. Peers position themselves in multiple keyword layers via virtual links to other peers which share one or more keywords. Search queries are propagated through the appropriate keyword layers in order to find the intersection of peers which match all of the keywords contained in the query. By organizing the network, and hence how queries are routed, according to keyword associations we are able to reduce the amount of traffic generated by searches and locate both rare and popular resources in a reliable manner.

The rest of this paper is organized as follows. In Section II, we introduce the design of our keyword associating overlay and explain the Multilayer Gnutella protocol in detail. In Section III we will describe our simulation environment and present the results of our experiments. In Section IV, we compare our work to state of the art P2P protocols which support multi-keyword, wildcard enabled searches. In Section V our conclusions and a description of our future work are given.

II. MULTILAYER GNUTELLA – PROTOCOL DESCRIPTION

In Multilayer Gnutella, peers are responsible for their shared content, or resources, and each peer maintains an inverted index of descriptive keywords for the content it is sharing. The overlay network is created by peers forming links to other peers which share a common keyword, i.e. they form virtual keyword layers as shown in figure 1. As the layers are organized in this manner, we view the layers in Multilayer Gnutella as being partitioned by keyword. Layer intersections are formed by peers sharing more than one resource, or resources having multiple keywords, and therefore being members of multiple layers. By using this overlay topology, multi-keyword queries such as ”ajax AND web2.0 ” are not forwarded to peers which do not belong to the responsible layers.

In addition to the keyword layers, the Multilayer Gnutella protocol builds a weak link layer consisting

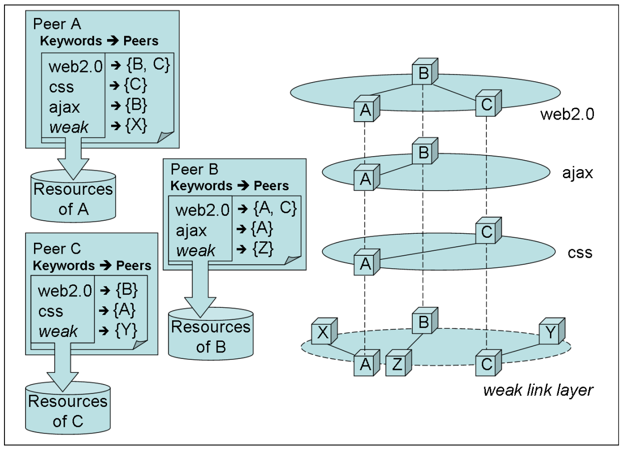


Fig. 1. Virtual structure of multiple keyword layers associating the keywords that the peers share

of randomly discovered peers which do not have any keywords in common. The weak link layer prevents search queries from being trapped in keyword layers that are not useful. For example Peer Y, shown in figure 1, will only receive successful results for the query "ajax AND web2.0" by propagating the search through the weak link layer.

All communication between peers in a Multilayer Gnutella network is carried out via query messages marked by a time to live (TTL). The TTL of a query determines how often the query will be forwarded to linked peers. The three main Multilayer Gnutella queries are: join, stabilize and search. The join query enables peers to discover other peers, stabilize queries are used to detect changes in the network and search queries are used to search for resources in the Multilayer Gnutella network. A description of the three main query types is given below.

a) Join Query: In order to join keyword layers, Multilayer Gnutella peers must extend their local inverted index with the addresses of other Multilayer Gnutella peers that are sharing resources with one or more common keywords. A peer discovered by the joining peer which has a keyword in common to the joining peer is called a keyword link.

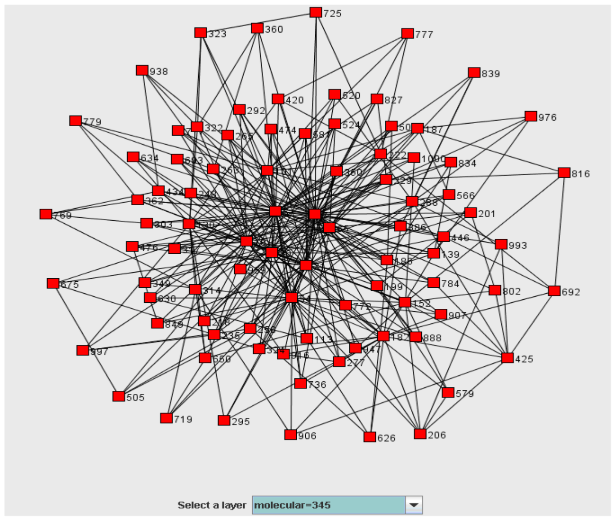
When a peer wishes to join the system, it must send a join query containing all unique keywords in its reverse index to the Multilayer Gnutella network. In order to do this, the peer must first send the join query to a rendezvous peer, or well-known host, which will return a list of candidate peers. After receiving the candidate list, the joining peer sends the join query to one or more of the candidate peers. A receiving peer consults its keyword links list for keyword links matching the join query's keywords. If matches are found, the join query is forwarded to those peers. If the receiving peer has no matching links, the query will be forwarded to all unique peers in the keyword links list. The join query is propagated in this manner until its TTL expires.

Peers contacted during the join process decide whether or not to respond to a join query based on the query's remaining TTL, their current linkage and most importantly if they have keywords in common with the joining peer. A join response message is sent directly to the joining peer and contains all matching keywords from the join query. By using the remaining TTL, a receiving peer only replies if the TTL has the maximum or minimum value. When the TTL has the maximum value, the receiving peer is one of the first peers to receive the join query. When the TTL has the minimum value, the joining peer has spread its links in one keyword layer as far as the TTL of the join query allows. This form of link management attempts to ensure that peers in a keyword layer are not successively linked to links of their links, because these peers can already be reached when forwarding queries with the given TTL. Join queries which are received more than once are discarded and join queries are not propagated to peers from which a join query was received.

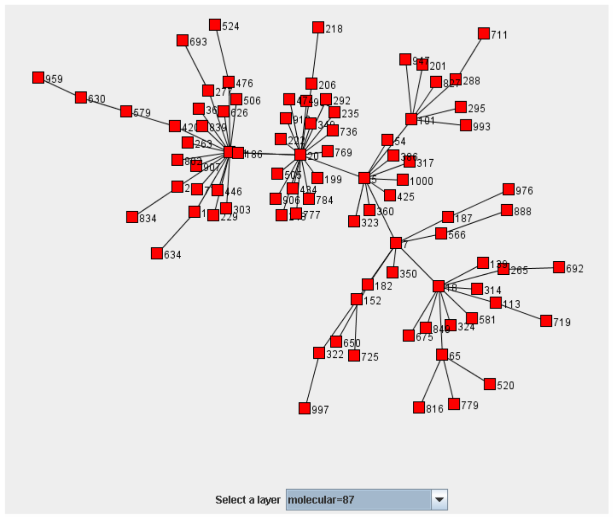
Figure 2(a) shows the resulting topology in a keyword layer when peers have a maximum of four links. When compared to figure 2(b) which shows the same keyword layer with a maximum of one link, we can clearly see that link management with four links per keyword provides a more resilient topology with shorter routes between peers. It should be noted that, currently, all links in Multilayer Gnutella are unidirectional and that Multilayer Gnutella peers do not maintain open connections to peers in the keyword links list, i.e. Multilayer Gnutella connects to others peers on demand.

b) Stabilize Query: Stabilize query messages are routed in the same manner as join queries, with the only difference being that a stabilizing peer has already established matching keyword links with other peers and therefore does not require any candidate peers from a rendezvous peer. During the stabilize process, the stabilizing peer detects: unavailable peers, peers that recently joined the same keyword layer or keyword changes in the currently linked peers due to resource deletions and additions.

c) Search Query: Search queries in Multilayer Gnutella consist of multiple keywords, which may contain wildcards. When a peer receives a search query, it



(a) Peers are allowed 4 links per keyword which are outspread as far as the TTL allows



(b) Peers are allowed 1 link per keyword

Fig. 2. Effect of link management on the topology of keyword layers

builds a matching layers list from its local keyword links list. The matching layers list contains those layers with keywords matching the query keywords. Based on this a ranked peers list is calculated which will be sorted in descending order by the number of occurrences of a peer inside the matching layers list. If the occurrence of the first peer in the layers list is greater than one, the link ranking mechanism is used to forward the query to the first three peers in the ranked peers list. Otherwise, the layer ranking mechanism is used to forward the query to three random peers in the most populated matching layer, i.e. the matching layer that contains the most peers. When forwarding queries to other peers, the query TTL is decremented and forwarding only takes place if the TTL has not expired.

If no known peers are members of a layer which matches a keyword in the query, it gets forwarded to the weak link layer. The pseudo code for Multilayer Gnutella query routing, link ranking and layer ranking can be seen in the listing below.

Link ranking is used whenever peers exist that are linked with more than one matching keyword. Since the peers which match the query locally best are used to forward the query, the search gets narrowed down to peers that match the query globally better with each hop. If no peer was found which is at least in two layers for a query, layer ranking is used to forward the query to the most populated layer in order to spread it in one layer as far as possible.

```
MAX_FORWARD = 3
Layer = struct (keyword:String, peers:list of Peer)
Query = list of String
// default value for count is 0
PeerCount = struct (peer:Peer, count:Int = 0)
layers : list of Layer

func matching_layers(query:Query) : list of Layer
  ls : list of Layer = []
  for each l in layers
    for each kw in query
      // match with wildcard support
      if kw matches l.keyword and not (l in ls)
        ls << l
  return ls

func route_query(query:Query) :void
  ls = matching_layers(query)
  // count occurrence of a peer in matching layers
  pcs : list of PeerCount = []
  for each l in ls
    for each peer in l.peers
      pc = find or insert into pcs by peer
      pc.count += 1

  if pcs is empty
    forward query to three peers in weak layer
    return

  pcs = sort pcs by descending PeerCount.count

  if pcs[0].count > 1
    for i in 0..MAX_FORWARD-1
      forward query to pcs[i].peer
  else
    ls = sort ls by descending Layer.peers.length
    // ls[0] is the most populated layer
    for i in 0..MAX_FORWARD-1
      peer = ls[0].peers[i]
      forward query to peer
```

III. Simulation Results

For our simulation, we used a collection of research papers from [11] as resources. From each resource, we extracted the author last names and words from the title of the paper as descriptive keywords. Each peer was randomly assigned between 3 and 10 papers which it could share with the network. In the distributed papers some authors published more than others, and some keywords appear more often in titles than others.

For comparison purposes, we built the simulator so that it worked with both a Gnutella v0.4 protocol implementation [4] and a Multilayer Gnutella implementation. In the Gnutella network, the simulated peers had a maximum of 15 connections with randomly discovered peers, and search queries were flooded through the network. The Multilayer Gnutella protocol was implemented as described in Section II and each simulated peer had a maximum of 4 links per keyword. In both networks the TTL of a query was set to 3.

During our simulation runs, we let the networks grow stepwise and took snapshots of each network at sizes of 100, 250, 500, 750 and 1000 peers. Each peer searched 3 times for different popular and then rare resources. For each search a random paper was selected from either the popular or rare resource set. Three of the keywords of the paper were selected as the search keywords. A search result matching all of the keywords was only counted as a hit if the paper id was the same as the one we picked for that search. Success rates were calculated as the ratio of number of total searches to number of searches that returned minimum one hit. The success rates of the search simulation in the growing networks were plotted against each other in figures 3(a) and 3(b) for popular and rare resources respectively.

A well-known problem of Gnutella is that it cannot reliably find rare resources, which is apparent in figure 3(b). One can also see that the problem of finding rare resources increases with the size of the network.

An interesting property of Multilayer Gnutella on the other hand is that it shows the same, very high levels of success independent of whether the search is for a popular or a rare resource in the system. Once the search query reaches peers which have matching keyword links the link and layer ranking mechanism help peers to align their local routing decisions. The peers forward the query along to the most populated matching keyword layer, or to peers that have two or more keywords matching the search query. When the search query is forwarded to peers that are linked in other keyword layers than the search query's keywords, their weak links help reach the appropriate keyword layers with the next hop.

IV. Related Work

The overlay topology in P2P systems must match the method used to describe and retrieve resources in order to efficiently route queries through the system, e.g. hash keys or meta-data. Examples of P2P networks, which demonstrate this are Chord [10], Semantic Overlay Networks (SONs) [3] and Shark [6]. Chord provides efficient key lookups by creating an overlay topology that can efficiently route keys. SONs use classifications as meta-data to describe resources. Each resource, each peer and each search query is categorized using the same classification scheme. The classified search query can then be efficiently routed through categories of peers. Shark on the other hand uses multidimensional meta-data hierarchies to both describe the resources and create the overlay network. Neighboring peers represent the same meta-data level. Search queries that are categorized using the same meta-data can then be efficiently routed through appropriate overlay levels.

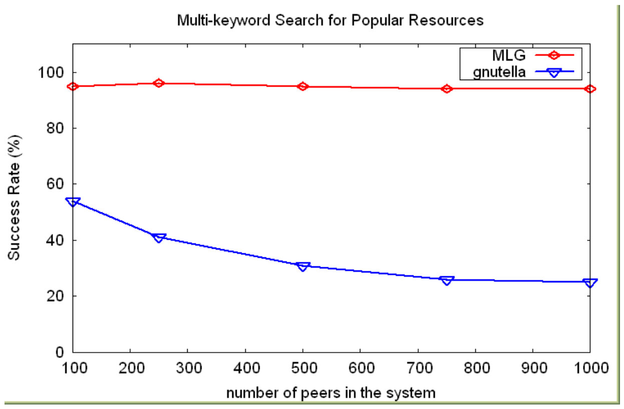
In all of the above systems, the efficiency and degree of flexibility of the search is determined by the overlay topology, which conforms to the meta-data used to describe, or reference, the resources. Multilayer Gnutella exposes the same characteristic, but uses simple textual keywords, as the basis for both building the overlay topology as well as the meta-data for describing and addressing resources. In contrast to categories, textual keywords are application independent.

Wildcard searches in DHTs are a challenging problem. Squid [9] uses space filling curves in order to map multiple dimensions of keywords and substrings onto the one-dimensional key space of a DHT such as Chord. The complexity of the algorithm and the inherent rigidness of keys as meta-data make it difficult to support wildcard enabled multi-keyword searches efficiently [7].

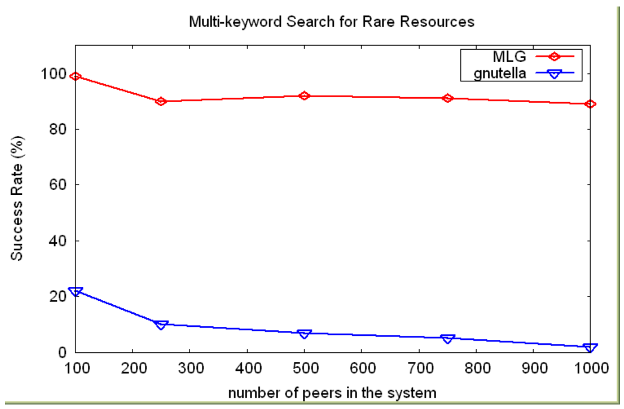
Hybrid P2P systems make an effort to combine the flexibility of unstructured systems with the provable efficiency of key lookups in DHTs [5]. The rationale behind this proposal is that unstructured networks are fast and feasible at locating popular resources in the network, whereas DHTs can guarantee search results for rare items. However, in order to use a hybrid system efficiently, the popularity of resources in the network must be known in advance.

V. Conclusion and Future Work

Multilayer Gnutella avoids imposing rigid or categorization structures on the overlay and maintains the



(a) Popular resources



(b) Rare resources

Fig. 3. Success rate comparison of flexible multi-keyword searches in Gnutella and Multilayer Gnutella (MLG) networks

flexibility and resilience properties of pure P2P networks. By using keywords to create the topology of the Multilayer Gnutella network, searches are routed towards the intersection set of peers that have all of the keywords in the query, which means that the scope of the propagation is limited to those peers which have relevant links and data. The typical problem of locating rare and popular resources equally well is also addressed in Multilayer Gnutella.

In the Web2.0 application domain, tagging has become a popular way of labeling resources with descriptive, simple textual keywords. In the flickr [2] community, users tag their photos so that photos from other users which are tagged with the same keywords can be found easily. Bloggers tag their postings in order to share knowledge and ideas with other interested users. A considerable amount of Web2.0 systems [1] are interlinked by users that share content and their keywords. We believe that Multilayer Gnutella can be used to support tagged resource sharing in a P2P manner. To this end, our future work will attempt to enable a P2P search systems for blogs.

A full implementation of Multilayer Gnutella has been integrated into the Siemens Resource Management Framework [8].

References

[1] Best web2.0 web sites. http://arbitraryreadings.blogspot.com/2006/10/best-web-20-sites_9358.html, 2006.

[2] http://www.flickr.com, 2006.

[3] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P systems. Technical report, Stanford University, 2003.

[4] RFC-Gnutella v0.4. http://rfc-gnutella.sourceforge.net/developer/stable/, 2003.

[5] Boon Thau Loo, Ryan Huebsch, Ion Stoica, and Joseph M. Hellerstein. The case for a hybrid P2P search infrastructure. In *Proceedings IPTPS*, 2004.

[6] J. Mischke and B. Stiller. Rich and scalable Peer-to-Peer search with SHARK. Autonomic Computing Workshop Fifth Annual International Workshop on Active Middleware Services, Seattle, Washington, USA., 2003.

[7] D. Olpp, S. Rusitschka, and A. Southall. Practical experience with publishing real names in a Chord DHT using space-filling curves. not published yet, 2006.

[8] S. Rusitschka and A. Southall. The Resource Management Framework: A system for managing metadata in decentralized networks using Peer-to-Peer technology. In *Agents and Peer-to-Peer Computing*, volume 2530, pages 144–149. LNAI, 2003.

[9] C. Schmidt and M. Parashar. Enabling flexible queries with guarantees in P2P systems. *IEEE Internet Computing*, 8(3):19–26, May/June 2004.

[10] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-to-Peer lookup service for Internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[11] UCI KKD Archive. Text data sets: NSF research awards abstracts [1990-2003]. http://kdd.ics.edu/databases/nsfabs/nsfawards.data.html, 2004.