8th Netties Conference
Technische Universität Ilmenau
September 30th to October 2nd 2002

Thorsten Strufe / Dietrich Reschke

Efficient Content Distribution in semi-decentralized Peer-to-Peer-Networks

1. INTRODUCTION

Sharing content such as scientific papers or multimedia data in locally distributed organisations or between a group of co-operating users still poses a couple of problems.
Existing systems like centralized client-server solutions and distributed peer-to-peer applications both suffer from their characteristic disadvantages.
Centralized systems on the one hand have advantages in their good manageability while lacking fault tolerance, load balancing and scalability, especially when the shared content consists of many large files. Peer-to-peer systems [1] on the other hand are fault tolerant by design, offer much easier ways to implement load balancing and are simple to extend as new participants can join wherever they are and whenever they come up. The drawbacks are that they usually are hard to manage, require intelligent lookup strategies and by nature do not scale to large numbers of participating servants [2]. The last few years have seen a huge number of systems, topologies, paradigms and models, which in one way or the other were called peer-to-peer. In this paper the basic definition is used: the peer-to-peer model describes a system or virtual network consisting of equal servants, which perform or at least are able to perform the same tasks. Each servant may act as either a server or a client in different communication relations. This model defines the opposite of the traditional client-server model as the definition leaves space to introduce some loose structure but fundamentally describes a decentralized system.
Another drawback of existing systems is the fact that global searches usually do not produce very satisfying results, as it is still impossible to group the hosts by interest and hard to determine sources of good quality.
No system, apart from mail and messaging, offers the possibility to actively distribute, or "push" content to a broadly distributed set of users or implements satisfying secure multicast today. Multicast, or group communication, stands for the communication between disclosed nodes where any node belonging to the group may send data, which every other node of the group receives.
A general lack of applications can be observed which enable users to share resources in closed groups and to make them searchable and locatable.
The aim of this paper is to outline an efficient, fault-tolerant content distribution system which introduces simple secure group communication and which remains scalable, even to large files as well as to a large number of resources or participants.

2. CHALLENGES OF THE DOMAIN

The system consists of very heterogeneous participants; its domain can be described as a highly dynamic ad-hoc network. Hosts may have a broadband connection or might be connected through unreliable links or narrowband connections. Different nodes have different uptimes,

some nodes might be on workstations or servers, which are up most of the time while other hosts might use dial-up connections or be up for a short period of time.
The only way to maintain service in this domain is through a decentralized topology, hence the system needs to be peer-to-peer. These systems have a couple of disadvantages as mentioned in the first section: it is almost impossible to determine their structure and as a consequence they do not scale well to high number of participants. These characteristics lead to a couple of problems:

1) Due to the lack of structure group communication is not easily implemented;
2) The break down of one performant servant can cause the fragmentation of the virtual network into separated parts;
3) Search in peer-to-peer networks does not scale well;
4) File transfer in peer-to-peer networks frequently imposes a high amount of unnecessary traffic on the underlying networks.

The reason that it is impossible to implement satisfying group communication in completely decentralized organized systems lies in the fact that there is no apparent possibility to locate, notify and safely authenticate members of a group.
The servants in a peer-to-peer network communicate only with the hosts in their direct neighbourhood. In a network with nodes of different performance many nodes rely on few performant servants, which as a result become hubs in the network. Fragmentation can occur if one of these hubs, which previously was the only connection between two otherwise separate partitions of the network is shut down or subject to a failure.
Searching in peer-to-peer networks is usually done by flooding the network with a query. Each node that receives a request performs a local lookup and again broadcasts the requests to its neighbours. Time-to-live fields, which lead to a limited search horizon, are introduced to avoid stray requests. If a search does not produce any hits a new search is performed with an increased TTL-counter to extend the number of reached hosts. This method generates a lot of traffic on every node in the search-vicinity and hence has a huge impact on the scalability.
As the servants have no information of the structure of the net, peer-to-peer networks generate a lot of unnecessary load on the underlying networks. Servants usually just locate resources and start downloading them from whichever source responds first. This might be a servant next door or a servant on the other side of the globe. Subsequent hits are generally ignored. If the connection to the source breaks down or if the source servant ceases service the whole procedure has to be started from the beginning.

3. RELATED WORK

Current work on scalable search in peer-to-peer systems can be classified into three categories: Message chaining [3,4], distributed hash tables (DHT) as lookup indexes [5,6,7,8,9] and the introduction of servers in assisted peer-to-peer [10,11,12].
The concept of the Message Chain, which, slightly altered, in recent papers [4] is sometimes called "Random Walker", follows the idea of avoiding to broadcast queries by joining current requests in a node and sending them on random trails through the network. In some approaches, each query in such a joined message chain has its own TTL-field to stop it from staying on the network forever. Other approaches introduce queries, which talk back to the node where they originate, to find out if a different trail has already produced results.
Both approaches fail to address the fact that in certain periods, many queries are placed for the same resources and that caching of results can lead to a much more efficient search.
The idea to introduce a distributed hash table as an index stems from the fact that every node in the network has a unique ID, which can be used to build a ring, ordering the hosts by increasing IDs. When a servant publishes a resource it hashes the resource's name and places it on the servant with the ID closest to the hash value. To locate resources, a servant sends the hash value of the requested resource, on the ring towards a servant with an ID close to the value. Every servant on the way forwards the request towards the servant with the matching ID, where a

lookup is performed. The reply is sent back over the ring to the requesting servant. Systems using distributed hash tables scale a lot better than broadcasting systems but are unable to perform any other than exact matching of the queried string. Other downsides of DHTs are the facts that they are not fault tolerant by design and that resources have to be published regularly to keep the index up to date.
Assisted peer-to-peer either relies on well-known central servers or on servants, which are configured to be superpeers. The first approach, of which Napster probably is the best known, is not a pure peer-to-peer system and depends on central lookup servers, which support the servants in locating resources. It is by design not fault tolerant as the server represents a single point-of-failure. The same applies for superpeer approaches: the emerging network in these systems can handle failures better but are prone to fragmentation.

4. CONCEPTS

The proposed system introduces some strategies to overcome the mentioned problems and to make secure group communication possible.

4.1 MAKING PEER-TO-PEER SCALABLE

One of the key issues of the system is the introduction of some sort of locality. To evade unnecessary load in the underlying networks nodes always try to copy replicated chunks from the nearest source. While locally available data is downloaded from local nodes, locally unavailable data is copied from remote sources in parallel to maintain an equal distribution of the resources over the system. This behaviour causes a far less long distance traffic on the network.
The peer-to-peer model is slightly altered due to the introduction of supernodes [13], which provide lookup and group services. This leads to a loosely structured semi-decentralized system of participating servants. The servants are equal in the sense that every node is capable of performing the same tasks. They are self-aware, have information about the network and can dynamically reconfigure themselves to supernodes and back whenever needed. The supernodes provide the basis for caching mechanisms as well as for simple group services. Nodes are relieved from a high volume of traffic as lookup is performed between the supernodes.
To keep the load down for supernodes, message chains are used in supernode communication.
To enhance fault tolerance and quicker downloads the shared files are fragmented into chunks to enable parallel download and to avoid additional traffic in the case of a broken connection.

4.2 GROUPS

To enable active content distribution in decentralized systems it is necessary to have the ability to group servants and a means to notify them. Otherwise the push of a resource would lead to a network wide broadcast of the file. Servants would not have the possibility to protect themselves from receiving every data pushed by any other servant of the system.
A very simple group model can be achieved through the extension of the description that nodes publish at their lookup services by the names of open groups they belong to. This model is only suitable for open groups without the feature of group communication and reliable notification.

4.2.1 Group model

More sophisticated models are needed to accomplish closed groups with the opportunity to implement group communication. Closed groups require features for identification, authorization and authentication. Every servant needs a key pair and a group-ID as the group services of the system are relying on public key cryptography. The servant's fingerprint is used as its group-ID as it should be non-ambiguous and short.
Authorization can be organized flat or role-based. In flat group authorization all members of the group are trusted equally and have the right to authorize any other interested party to become a

member of the group. In flat group authorization it is not possible to exclude a participant from the group. Role based group authorization offers access control over the membership of participants. Usually only the group owner or specially authorized members can authorize interested parties to join the group or exclude participants from further membership. The proposed system uses role-based authorization. If a servant creates a group it becomes the owner of it. The whole group management is owner-centred. To relieve the creator of the group from having to authorize every member it is possible for the owner to appoint substitutes, who then are considered owners of the group themselves. Simple authorization is done by adding the group-ID of a servant to the group repository. A way to authenticate a servant as being member of a group is to identify it and to look up its group-ID in the group repository.

4.2.2 Secure Groups

Security is hard to implement in systems of an untrusted decentralized topology. Every node has to rely on the information it is given by its peers, as there are no well-known central security services at hand. However, with public key cryptography it is possible to build a basis for trust in untrusted environments. If the public key of a peer is signed by a trusted servant it can be assumed that the peer is who it claims to be.

To enforce that only members of the group can access group information a group key is needed.

Reliable authorization of a new member is done through signing the public-key of the member by the owner, adding the group-ID to the member repository and notifying authentication services of the updated repository. To make sure that no unauthorised servant tampers with the group repository it has to be public key encrypted by the owner of the group.

Exclusion of servants can be done by marking the corresponding group-ID in the member repository and notifying every authentication service of the update. The group key additionally has to be changed to keep the excluded servant from reading group messages.

For secure authentication the servants need to know if the peer is who it claims to be and to lookup if it is a member of the group. The latter is done easily by looking the peer up in the group repository. With the group-ID it is possible to check the propagated public-key, as it has to be the fingerprint. To further authenticate a requesting peer the Needham-Schroeder protocol is user: A random string encrypted with the requesting peer's public key is sent to it, which it decrypts, encrypts with the public-key of the requested servant and sends it back. After this procedure the requested servant can be sure of the identity of the peer, as both the private- and the public-key are needed for it.

Group information has to be placed on trusted servants, which in a decentralized environment can only be servants belonging to the group.

5. SYSTEM MODEL

The system is composed of locality-aware servants and organized in a loosely structured peer-to-peer topology.

5.1 LOCALITY

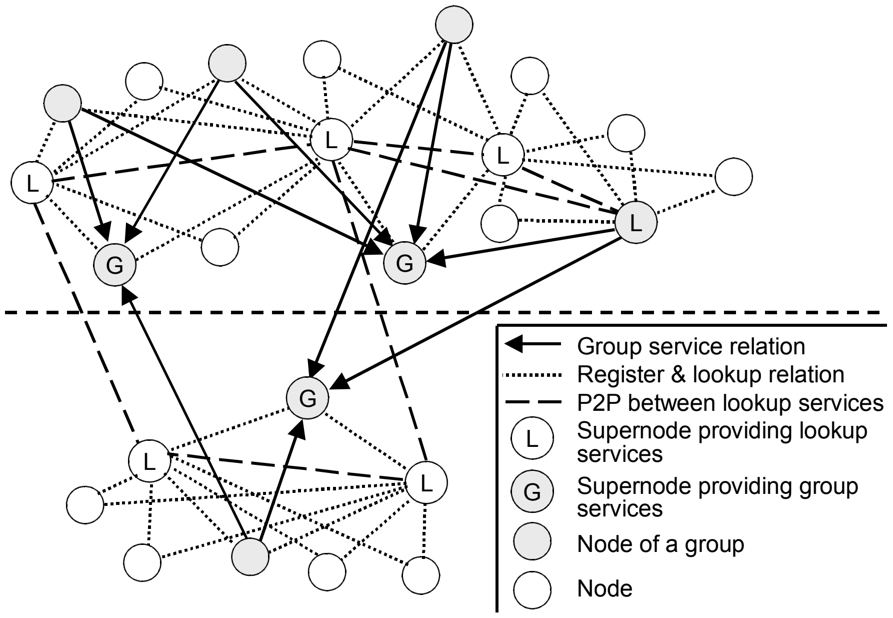
Without a global positioning system or the information in the routers of the backbone geographical locality is very hard to determine. As the system relies on locality, a relative distance, calculated from IP-Hops and response times, is introduced. Each node calculates its own relative distance to other servants and defines which servants it consider local and which remote.

5.2 ROLES

Servants can act in different roles, either simple nodes or supernodes. Supernodes provide either lookup or group services. Every servant acts as a simple node on start-up and can dynamically be reconfigured to become a supernode offering additional services.

5.2.1 Nodes

Nodes register at local lookup services and provide them with a list of its shared content.
Every node keeps a peer-list, a list of known other servants (sometimes called host-cache or warehouse), which contains local and remote lookup and group services as well as nodes with which it was in contact. If it is communicating with a formerly unknown node that it considers local the two nodes swap information about their local lookup services. On discovery of a new local lookup service a node registers itself at it, determines which lookup services have the highest uptime and which lie in the closest distance and uses them.


L
G
Group service relation
Register & lookup relation
P2P between lookup services
Supernode providing lookup services
Supernode providing group services
Node of a group
Node
Fig 1: System sketch including nodes, lookup- and group services

If the number of supernodes, providing either lookup services or group services for a certain group falls below a lower bound simple nodes are triggered to reconfigure [13] themselves and to offer the needed services.

5.2.2 Lookup Supernodes

Lookup services act as search gateways for their local nodes. Any messaging dealing with the lookup of resources is done between the lookup services.
Lookup services can be divided into a naming and a lookup, or localization service. Naming is used to find the identifiers of resources with a certain name; lookup is used to locate instances of a resource on the network. Supernodes offering lookup services hold extended peer-lists. These in addition to the ID, address and distance to the nodes contain the information about their shared content. Search results as well as information about remote servants are cached in lookup services. Servants in peer-to-peer networks tend to be either highly volatile or very reliable [14] and hence are replaced in the cache by inverse uptime: the servants with the shortest uptime are replaced first. Search results are replaced using a LRU-strategy.

5.2.3 Group Supernodes

Group services comprise of notification, key distribution, authentication and secure naming. The information needed to perform these services is an up-to-date group key and a trusted list of current members of the group. To relieve the servants belonging to a group from having to search and download this data any time a new member joins or a member is expelled from a group the system contains group supernodes, which offer these services and publish them at their local lookup services.

Every node belonging to a group registers itself in at least two group services of its group to make notification possible and to obtain a dependable way to distribute group keys. The group subnetwork can be considered reliable as group keys are updated – and distributed – regularly and nodes notice if they are contacted by only one of their group services or not at all. If a node observes that a group service has failed it searches for different group services and registers itself with them.
To enable secure naming of classified files in a group, a method of secure naming is needed. Lookup services that do not belong to a group, are unable to perform a different search for encrypted names than exact match. To realize a secure naming service, nodes register their encrypted content at the group nodes. These then implement a simple secure naming service analogous to the lookup services.

6. CONCLUSION

The major drawbacks of current file sharing and content distribution systems are its inability to send data to a certain group of peers, unsatisfactory scalability and communication overhead. The created system solves these problems through introducing relative locality and specialised services.
Some additional advantages are achieved including firstly, possibility to search for content on hosts, which are known to belong to a certain interest group only, which leads to a much better set of hits. Secondly the system integrity is ensured and load balancing as well as fault tolerance are integral part of the system.
We expect that such data sharing networks are going to provide the basis for large scalable internet applications.

References:
[1] *The Gnutella Protocol Specification V. 0.4*; Available from: http://rfc-gnutella.sourceforge.net/.
[2] J. Ritter. *Why Gnutella can't scale. No, really*, 2001. Available from http://www.darkridge.com/ jpr5/doc/gnutella.html.
[3] M. Wulff, P. Kropf, and H. Unger. *Message Chains and Disjunct Path for Increasing Communication Performance in Large Networks*. In P. Kropf et al., editor, Distributed Communities on the Web 2000.
[4] Qin Lv, Pei Cao, Edith Cohen, Kai Li, Scott Shenker: *Search and Replication in Unstructured Peer-to-Peer Networks.* In Proceedings of 16th ACM International Conference on Supercomputing (ICS'02), New York, USA, June 2002
[5] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan: *Chord: A scalable peer-to-peer lookup service for Internet applications.* Technical Report TR-819, MIT, March 2001.
[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker: *A Scalable Content-Adressable Network* SIGCOMM01, August 27-31, 2001, San Diego, CA, USA
[7] A. Rowstron and P. Druschel: *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems.* IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), November, 2001.
[8] P. Harrison: *The Circle: Some first steps towards decentralizing internet services.* 2002. Available from http://www.csse.monash.edu.au/~pfh/circle/talk.pdf
[9] P. Maymounkov and David Mazières: *Kademlia: A Peer-to-Peer Information System based on the XOR Metric.* 2002. Available from http://www.cs.rice.edu/Conferences/IPTPS02/109.pdf
[10] Napster http://www.napster.com/.
[11] FastTrack http://www.fasttrack.nu/.
[12] EDonkey2000 http://edonkey2000.com/overview.htm
[13] T. Strufe: *Effizientes Peer-to-Peer-Distributionssystem für multimediale Inhalte.* Net.Objectdays2002, Workshop "Multimediale Informations- und Kommunikationssysteme" to appear.
[14] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. *A Measurement Study of Peer-to-Peer File Sharing Systems*. In Proceedings of Multimedia Computing and Networking, 2002.

Authors:
Thorsten Strufe
Prof. Dr. Ing. habil. Dietrich Reschke
Technische Universität Ilmenau, Postfach 100 565
98693 Ilmenau
E-Mail: {thorsten.strufe,dietrich.reschke}@tu-ilmenau.de