[HSKLSt06] *Oliver Heckmann, Marc Gomez Sanchis, Aleksandra Kovacevic, Nicolas Liebau, Ralf Steinmetz;* **A Peer-to-Peer System for Location-based Services;** Peer-to-Peer Paradigm (PTPP) Track at AMCIS 2006, Acapulco, August 2006, S.

Americas Conference on Information Systems
AMCIS-12 2006 Acapulco

A Peer-to-Peer System for Location-based Services

Oliver Heckmann
Technische Universität Darmstadt
heckmann@kom.tu-darmstadt.de

Marc Gómez Sanchís
Technische Universität Darmstadt
marcgom@kom.tu-darmstadt.de

Aleksandra Kovacevic
Technische Universität Darmstadt
kovacevic@kom.tu-darmstadt.de

Nicolas Liebau
Technische Universität Darmstadt
liebau@kom.tu-darmstadt.de

Ralf Steinmetz
Technische Universität Darmstadt
steinmetz@kom.tu-darmstadt.de

ABSTRACT

Location-based services promise to offer highly personalized content based on the users current geographical position. A typical example is a restaurant finder service that returns a list of restaurants in specified proximity of the users' current position (via GPS or mobile phone triangulation). Building the infrastructure necessary to offer competitive global location based services of different kinds can be a challenging task for an Internet or mobile phone service provider due to the amount of necessary infrastructure and information. Peer-to-Peer (P2P) systems, however, offer the possibility to deploy new services without the need for centrally managed infrastructure because they use resources available at the participating end-users' systems.

In this paper, we describe a new P2P system that can be used to offer location-based service, combining the advantages of location-aware services with the low costs and good scalability properties of modern P2P systems. The system is a structured P2P system based on a novel overlay network described in this paper. The main advantage of this overlay structure is that contrary to existing DHTs it supports location-based area search in a more efficient and reliable way. As proof of concept we built a network that offers location-based search and metadata based search for webcams that are connected to our network.

Keywords

Peer-to-Peer, Overlay, Location-based Services, Network Applications

INTRODUCTION

Peer-to-Peer (P2P) technology is currently rapidly expanding into application scenarios beyond filesharing. P2P offers many advantages over traditional client-server architectures. For example, a P2P system has no single point of failure and is therefore more robust against denial of service attacks. It will typically have cost advantages for the provider as it allows the provider to make use of the idle resources of the peers using the network. The self-organizing nature of the P2P network also makes the administration easier and cheaper. And P2P has growth advantages as the amount of resources grows automatically with newly joining peers bringing their own resources. The voice over IP (VoIP) provider Skype for example is extremely successful by using P2P technology. P2P allows Skype to use the resources of the participating peers to offer free telephony services with minimal investment in technical infrastructure[1]. Instead of running their own gateways and directory

[1] Skype internal calls can be offered for free as - besides authentication - practically all infrastructure is provided transparently by the participating peers and not the company Skype itself. For the Skype Out service that connects the Skype VoIP network with the plain old telephone systems, however, Skype has to provide and run a considerable amount of hardware resources which is one reason why this service is not offered for free.

servers, Skype is using a subset of the peers connected to the Skype P2P network to take over as many of these tasks as possible.

Looking at these promising advantages, it is therefore highly interesting to look for new application scenarios where P2P technology can be applied. We believe that location-based services can be such an application scenario. Location-based services offer context-aware content based on the users' current geographical position (Ljungstrand, 2001). A typical example is a restaurant finder service that returns a list of restaurants in specified proximity of the users' current position (via GPS or mobile phone triangulation). With the Google Local service (known as Google Maps, see (Google Local Service, 2006)), location-based services became also popular for so-called web 2.0 applications, see e.g. (Google Maps Mania, 2006).

In this paper, we investigate how location-based services can profit from the P2P communication paradigm. As we show in the related work section, traditional P2P systems are not well-suited for location-based services. Therefore, we present a novel structured overlay in this paper (Section "Overlay Design"). It is not a distributed hash table, instead it uses a distributed B-tree as core structure and additional links for robustness and efficiency reasons. Like a B-tree, it can therefore efficiently support location-based area search. As a proof of concept, we present an application called *camNet* which uses our system. It interconnects webcams and allows users to locate webcams in certain areas and regions. How *camNet* uses our overlay mechanism is discussed in Section "Application Scenario", before the conclusions.

Our solution could be used by commercial providers to build their infrastructure for location-based services harnessing the power of P2P and it could be used to form an autonomous overlay network independent of commercial providers where the costs for running the systems are automatically completely distributed among the participating nodes.

RELATED WORK

Peer-to-peer systems can be divided into structured and unstructured systems depending on whether peers are aware of the resources (e.g. files for filesharing or webcams in our application example) available at neighbouring peers or not. Location information is based on geographic coordinates that are well defined and highly structured. Therefore, it makes sense to this information as the basis for a structured P2P system.

Most of today's structured P2P systems are based on a DHT (distributed hash table), see e.g. Chord (Stoica, Morris, Karger, Kaashoek and Balakrishnan, 2001). They use hash functions to assign keys to resources and nodes to uniquely identify them. Then, these keys are mapped to the nodes responsible for them. When a search is performed, distributed look-up algorithms route the queries through some of the nodes, using the key itself as the routing information, until the node which administers the desired key is reached. The ID of a node is determined by the hash function and is completely independent of the location or powers of the node. DHTs are efficient with respect to the number of routing steps to look up one key in the overlay. They are not optimised for searching for all nodes/keys that fall into a certain area – and that exactly is required for location-based services and the main feature of our approach.

In a location-aware overlay, each node (peer) and each resource/service knows its location (e.g. GPS coordinates). We use the location plus a random number to uniquely identify a node. This has advantages over the hash function: For example, the geographical distance between two nodes in our system can easily be determined from their IDs. It strongly correlates with the number of hops in the Internet and the propagation delay between the two nodes. We exploit this additional information to optimise the structure of the overlay so that it takes the underlay structure into account. For example, a resource (a service) associated with a certain location will most probably be managed by the peers that are geographically very close to it. Such a procedure is typically not supported by most other P2P systems. Obviously, this promises a highly efficient peer-to-peer network.

CAN (Content Addressable Network, Ratnasamy, Francis, Handley, Karp and Shenker, 2001) is a DHT. The search space is represented as a d-dimensional Cartesian coordinate space on a d-torus. This space is partitioned among all the nodes, so that each one stores and administers its individual, distinct zone within the overall space. Every node is connected to each of its immediate neighbours in the coordinate space. On the first look, it might seem that a two-dimensional CAN has some similarities to the system presented in this paper. However, there are many differences. The most important one is that CAN uses a logic coordinate space where the keys are assumed to be uniformly distributed. Our system uses a physical coordinate space which directly models the surface of the Earth with the keys representing the physical location of each resource. Thus, they are not going to be evenly distributed in the geographical space, since they are distributed extremely unevenly on the surface of the Earth. We have to be prepared for the case that a small area like e.g. Manhattan offers more location-based services than all the oceans or the whole African continent. Our system is able to deal with hotspots and large empty areas very well. This issue is addressed in Section "Overlay Design", Subsection "Load Balancing".

ContextCast (Heutelbeck, 2002) is a system aimed to provide geographical addressing and resource discovery. It supports

two basic operations: search for and multicasting to all the objects located in a spatial target region *T*. As DHT, ContextCast stores (key, value) pairs, but with the peculiarity that in this case the value is a mobile object in the space and the key represents the point in the space where it is located. The ContextCast network is based on the recursive decomposition of the context space into a hierarchical structure of identical sized clusters, administered by special nodes (cluster heads) in the case they are not empty. This context space does not need to be restricted to the surface of the Earth, since additional, non-physical dimensions can be added. The nodes in ContextCast establish connections following the hierarchy tree and create deep tree hierarchies to balance the load which is the main drawback of this approach. Our system is overcoming this issue by dynamically sizing the clusters according to their load.

RectNet (Heutelbeck, 2005) is another P2P system that implements a DSTP (Distributed Space Partitioning Tree). RectNet is similar to ContextCast but has extended capabilities. RecNet is also based on a tree based network topology which dynamically adapts to the geographical distribution of the workload caused by the storage of (location, object) pairs and the processing of queries. This tree has a binary structure which simplifies recovery of the structure in the case of node failures but significantly reduces the search performance.

OVERLAY DESIGN

A P2P overlay network forms the basis of any P2P system as it manages and locates resources (and/or peers) in a decentralized fashion. In our system, a resource represents a service associated with one peer and one specific location like e.g. a restaurant or a webcam. Each node is aware of its own geographical location and the location concatenated with a random number uniquely identifies one node. Our overlay is structured as a hierarchical tree of the superpeers with interconnections (shortcuts) presented later in this section.

Hierarchical Approach

Peers offer and use services of the network and they have different capabilities. Some of them might be mobile end devices with little storage space and processing power with poor network connection while the others might be powerful desktop machines or even server-like machines with lots of RAM and good network connectivity. Furthermore, peers differ in their online behaviour. Some might be online for very long durations while others are not. To learn about the online behaviour, a mechanism like the burn-in optimization of (Darlagiannis, 2005) can be used.

Our overlay supports the heterogeneity of the participating peers by using powerful peers, with good network connectivity that tend to stay online for a long time as superpeers. Superpeers are responsible for indexing all peers/services in one clearly defined geographical area. Detailed explanations of how peer become superpeer and how their zone of responsibility is determined is presented below.

Tree-based Core

The core structure of our overlay is a tree formed by the superpeers. Peers are connected to the superpeer responsible for their location. The world is divided into disjoint zones and connected in the following way: *node A is the parent of a node B when B's zone is inside A's zone*. Figure 1 shows this idea graphically.

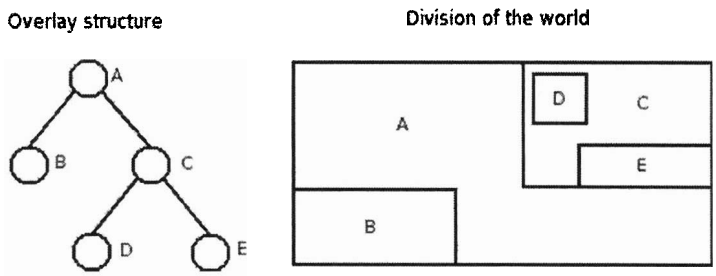

Figure 1: Core overlay structure and division of the world.

Mapping Geographical Data and Division in Zones

In order to be able to perform the required calculations and processing, a mathematical abstraction of the surface of the Earth has to be made.

Our overlay uses the Plate Carée projection to represent the two-dimensional curved surface of the Earth on a plane. This projection plots directly latitude-longitude points on a regular X, Y graph assuming the Earth is a sphere. The longitude lines on the graph are spaced using the same scale as the latitude lines, forming a grid of equal rectangles. Figure 2 shows a world map using a Plate Carée projection with 15° graticule for the latitude and 30° for the longitude.

All map projections introduce some kind of distortion, because an ellipsoid can not be mapped without stretching, tearing or shrinking to a plane. The distortion introduced by the Plate Carée grows with the latitude. For zones lying on the equator there is little distortion but zones far away from it are strongly distorted.

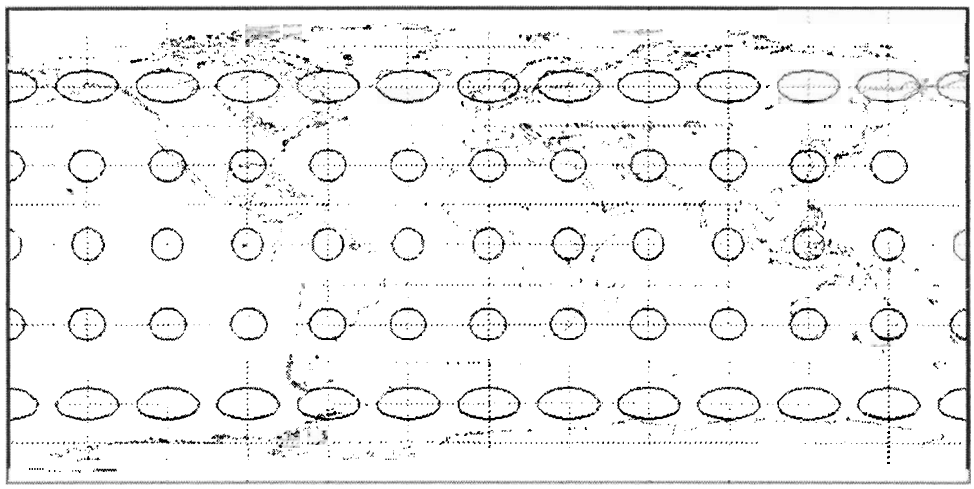

Figure 2: Plate Carée projection.

This distortion has to be taken into account when performing geographical calculations. If we want to search for zones lying within a specified radius of a point on the surface of the Earth, this circle is transformed into an ellipse on the overlay's flat projection. To visualize this, the mapping of the same circle but on different locations is shown in Figure 2.

Location-based Area Search

Our overlay is optimized for the efficient search for all services within a certain area (e.g. radius) of the Earth. The search area is specified by a simple geometric shape like a circle or rectangle. When performing a search, a peer sends a search message to its superpeer. Each superpeer knows exactly the area it is responsible for and the areas of its direct children.

When a superpeer receives a search message, it performs the following steps:

1. From the message, the superpeer calculates the shape resulting from mapping the search area onto the map projection used as explained above.
2. Then the superpeer checks if this shape is fully contained in its own zone. If not, the message is forwarded to the parent (or to the node closest to the full area if the appropriate shortcuts are available; shortcuts are introduced below as an optimization).
3. If the shape is partly overlapping with its zone, the node sends a search answer with all peers it is directly administrating and that fall into the overlapping shape. It also checks for each child if the shape intersects with the child's zone. If that is the case, it forwards the message to the child as that child might administer peers that fall into the search area.

Figure 3 shows a simple example of the look-up query routing in the core overlay (without shortcuts). Our mechanism makes sure that a search message is not forwarded and processed by nodes that do not administer matching peers as soon as the first node that overlaps the search area is reached. It also makes sure that the resulting search answer is complete; this means it contains all peers that match the search in the whole network and if no results are returned it is guaranteed that at no peers in the complete network match the search criteria. This is an important property for reliable location-based services.

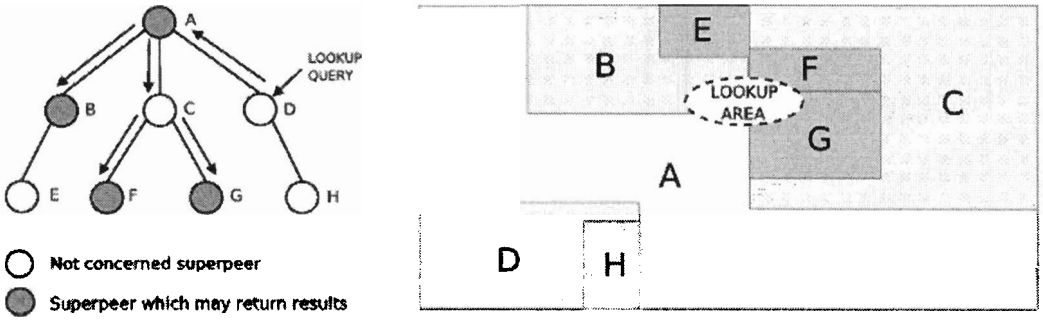

Figure 3: Example of the look-up query routing.

Load Balancing

Common P2P systems like Chord (Stoica et al., 2001) or CAN (Ratnasamy et al., 2001) rely on the consistency of the hash functions they use in order to guarantee that the keys are evenly spread between the nodes or they rely on additional load balancing mechanisms like those in (Byers, Considine and Mitzenmacher, 2003), (Godfrey and Stoica, 2005). Since the geographical location is the information used in the overlay to map resources to nodes, we can not rely on the fact that they will be uniformly distributed. Actually, they cannot be expected to be evenly distributed, since the population of the world itself is highly irregularly distributed. Their will be clear hotspots with many services/peers in the large cities and vast areas with very few peers, e.g., on the oceans.

Zones

In order to achieve a balanced distribution of load, the superpeer's zones are adapted to the load situation. In this way, areas with a high density of resources are administered by many superpeers with small zones, and areas with a low density of peers are administered by few superpeers with large zones.

Apart from the physical distribution of the resources on the Earth, another fact to take into account is their demand. In this way, superpeers administering peers with resources located in very popular zones should administer fewer peers (which intrinsically means having a smaller zone) than superpeers with not so popular zones.

Placing New Superpeers

As new superpeers join the overlay, new zones have to be generated and assigned. There is a trade-off between locality and load balancing: A superpeer should be responsible for an area close to his geographical location because this way the distance that a large amount of network traffic has to travel through the Internet is reduced. Further, a new superpeer preferably will have assigned zones from a highly loaded superpeer due to load balancing reasons.

This trade-off between locality and load balancing is resolved by the following protocol steps:

1. When a superpeer joins the overlay, it sends a NEW SUPERPEER message to a random active superpeer obtained with a bootstrapping method.
2. The message is routed through the overlay towards the node that administers the zone where the new superpeer's geographical coordinates fall into.

3. Each superpeer has a load status that is either "normal" (below a threshold L_1), "overloaded" (above L_2 but below a threshold L_2), and "critically overloaded" (above L_2).

   a. If on this way a critically overloaded superpeer is encountered, this superpeer generates a new area from his zones (details below) for the connecting superpeer and hands over the responsibility for it to the new superpeer. If an (non-critical) overloaded peer is encountered, the same process is started only if the geographic distance between the overloaded superpeer and the new superpeer is acceptably small.

   b. If no overloaded superpeer is encountered, the new peer is assigned zones around his physical location by the superpeer currently responsible for that area.

4. Next, the new superpeer is contacted and handed over information about its assigned zone and the peers therein. This is done by sending a SUPERPEER OK message. The new superpeer is thus added as a child of the other node.
5. Finally, the new superpeer sets the replying superpeer as its parent and begins administering the assigned zone and peers.
6. As we describe below, the new superpeer rewires itself with other superpeers automatically within the first minutes after being included in the overlay. This rewiring is done for performance and robustness reasons.

Clustering

To determine the new zone, a superpeer divides its rectangular zone in a specified number of equally sized rectangular clusters and keeps track of the distribution of load among the clusters. The calculation of the load of a cluster involves basically two parameters:

1. The amount of resources lying in the cluster.
2. The demand of these resources, calculated as the amount of search queries involving one of them per unit of time.

Having clustered its zone, the superpeer knows how the load is distributed within. Therefore, it is able to decide which subzones are the most loaded ones (hotspots) and create a new rectangular area out of some of these subzones that is then assigned to the new peer. This system is far more flexible than just splitting a zone into half as for example CAN is doing; see (Ratnasamy et al., 2001).

All peers lying in this newly generated zone are transmitted to the new superpeer, who starts indexing them. Figure 4 shows an example of this process.

Disconnecting a superpeer

Superpeers should not leave the overlay without previous notification of the other superpeers since they administer important information. Thus, before leaving the network a superpeer hands over to another superpeer his peer list and children. This other peer is either the parent peer of the leaving node or one of its children, depending of the load situation of these peers.

An issue in that context is the broken cluster problem illustrated in Figure 5. When superpeer B leaves, it may have also assigned some sub zones to new superpeers (its children). One problem is that the zones administered by B's children do not follow the clustering pattern of B's parent, making it difficult to add them as children. Therefore, either the zones are resized or one of the lower level superpeers is promoted to take the role of superpeer B, administering its former zone clustered with the same clustering pattern in order to be able to accept the others of B's children.

The latter is done in the current version of the system. When the parent receives a leave message from a superpeer, it sends a message to a randomly chosen child of this leaving superpeer, indicating the departure. This message is then randomly forwarded down the tree, until it reaches a leaf superpeer. This leaf superpeer replaces the departing superpeer, leaving his former zone to be administered by its former parent.

If a leaving superpeer does not have any children, the parent superpeer takes over the leaving superpeer's zone.

Formatiert: Englisch (Großbritannien)

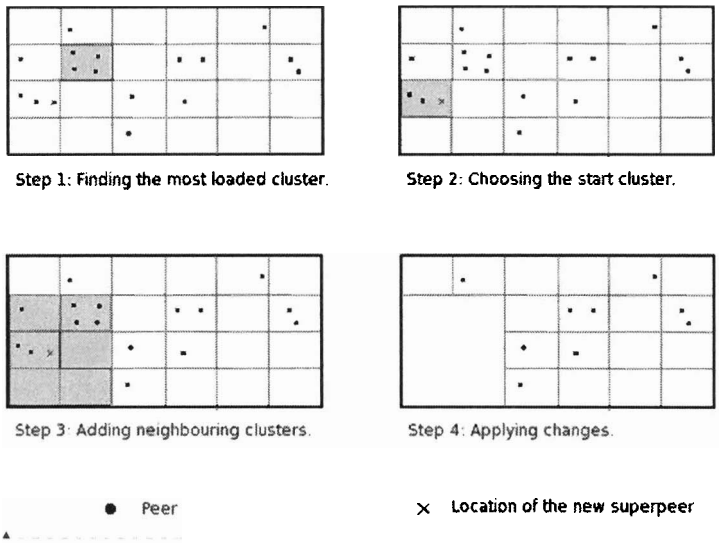

Figure 4: Example of a zone assignment.

Formatiert: Englisch (Großbritannien)

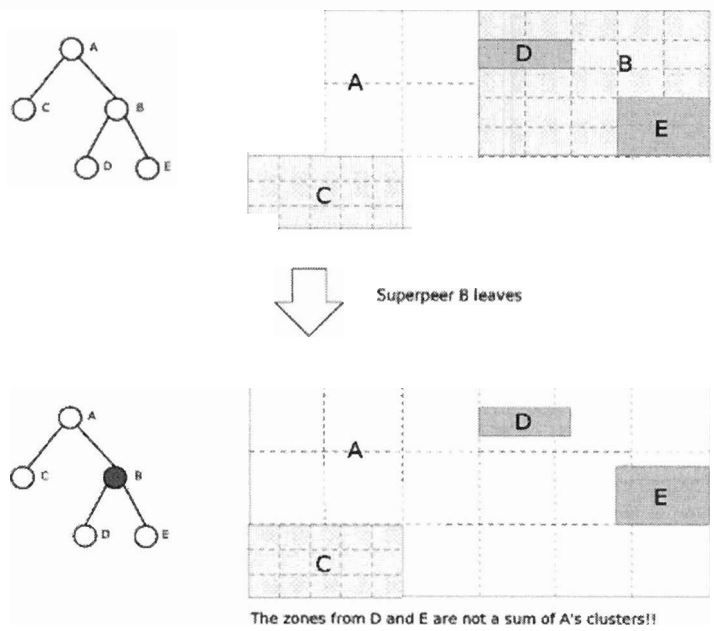

Figure 5: The broken cluster problem.

To summarize, the disconnection process follows these steps:

1. The departing superpeer informs its parent it wants to leave, sending him its zone, peers and children list. This is done by sending a REMOVE SUPERPEER message.
2. The parent contacts a randomly chosen child from the departing superpeer, and sends him a BECOME SUPERPEER message, which indicates the departure.
3. The message is forwarded down the tree until a leaf superpeer is reached.
4. This leaf superpeer informs its parent it is going to leave by sending a REMOVE SUPERPEER message to it. This message only informs about its former zone and peers (as it is a leave superpeer, it doesn't have any child).
5. The leaf superpeer contacts the departing superpeer's former parent, indicating it is going to replace it. This is done by sending a BECOME SUPERPEER REPLY message.
6. The necessary rewiring is made to have the initial core tree-based structure again. That involves sending PEER OK messages to all the new peers and SUPERPEER OK messages to all the new children. The peer administered by the leaving node also has to be contacted and informed about the address of the new node responsible for them.

If a superpeer leaves the overlay without sending the REMOVE SUPERPEER message the tree structure must be reconstructed. This is explained below in the "Robustness and Efficiency" section.

A special procedure is necessary when the root superpeer leaves. In that case, a BECOME SUPERPEER is directly sent by the root superpeer randomly to one of its children.

Figure 5 shows an example of the whole process.

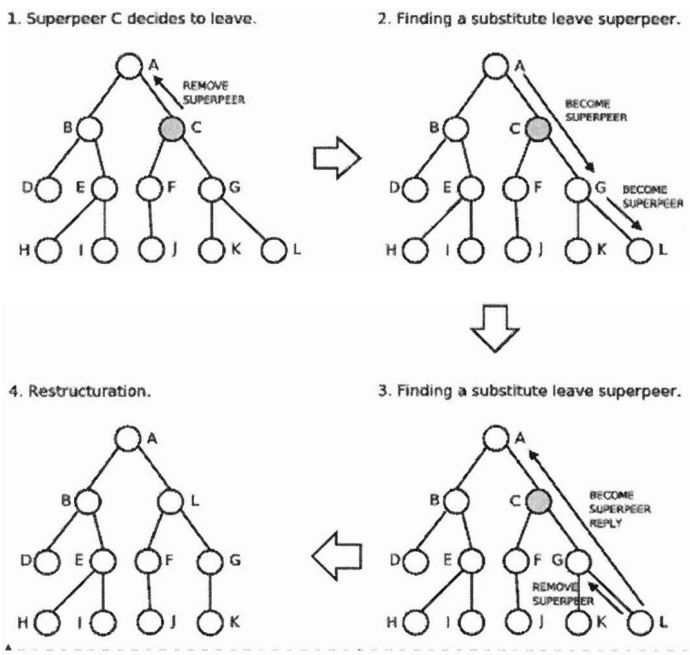

Figure 6: Example of the superpeer leaving process.

Formatiert: Englisch (Großbritannien), Rechtschreibung und Grammatik nicht prüfen

Robustness and Efficiency

Fingers and Shortcuts

As our core overlay uses a tree-based structure, scalability and robustness issues arise. To increase robustness, some rewiring is performed in order to flatten the overlay structure (see Figure 7). This decreases the average search duration and makes the overlay resistant to node failures. Each node in the overlay will have some additional connections – so-called *shortcuts* or *fingers* - to some chosen nodes which are not its parent or children. These additional connections can be used as shortcuts to reach any destination node in the overlay faster and using less communication resources of the nodes higher in the hierarchy.

For creating the finger / shortcut table (rewiring), information of search messages passing through a node is used as this does not create any additional communication overhead. For example, if node E performs a search for information administered by node B, node E will learn automatically from the search answer via nodes C and A of node B. After the search, node E can store a shortcut to node B for a limited (but typically rather long) time in its cache. Future searches can be directed directly towards node B if they fall into B's area. That way, nodes obtain shortcuts to frequently accessed areas automatically. A node tries to obtain shortcuts to at least one other part of the tree for each level of the tree hierarchy. If a node does not obtain these shortcuts automatically within a certain time, the node actively sends discover messages to random other parts of the tree to make sure that it obtains enough shortcuts. This way, the tree will not fall apart if nodes high in the hierarchy go offline unexpectedly.

Accidental Disconnection of Superpeers

In case a superpeer disconnects without performing the required steps (due to a system or network failure for example), a recovery process is started. Due to the rewiring, under normal operation no superpeer becomes isolated from the rest of the overlay. Therefore, once a superpeer detects that his parent node has gone offline without notification, it sends a BECOME SUPERPEER message along its shortcuts to the other parts of the tree to find a replacement for the missing node.

Normal (non super-) peers that are administered by a superpeer that has gone offline also get disconnected from the system when that superpeer goes offline. They use a standard keep-alive mechanism: A peers sends in regular intervals T a small UDP message to its superpeer indicating that it is still alive and expects an answer. If it does not get answer n times in a row, it assumes that the superpeer went offline without notification. It then uses the bootstrapping mechanism to connect one random node of the network, searches for the peer currently responsible for its location and then joins the network again at that superpeer.

The keep-alive mechanism is also used vice versa: If a superpeer does not receive a message from a connected peer for nT minutes, it assumes that the peer crashed and is no longer available.

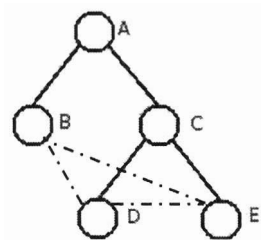

Figure 7: Full overlay structure

PERFORMANCE ANALYSIS

An important metric to track the performance of the system is the amount of traffic caused by a search query. We now present a worst case assessment. In the worst case, no shortcuts can be used and the search starts at the opposite side of the core tree. Let's assume that the overlay is structured very deeply with each superpeer having only two direct children. In that case, the maximum number of hops for a search query is $N_{hops} = 2t - 1$ where t is the depth of the tree. The total number of

superpeers n in the network is

$$n = \sum_{i=0}^{l-1} 2^i$$

In our implementation, each search query has payload of 346 bytes, the total volume of traffic generated by all nodes to forward the query to its destination assuming we have an overlay formed by 255 superpeers would in the worst case just be 4.4 KB.

APPLICATION SCENARIO

As proof of concept, we developed *camNet*, a peer-to-peer network of webcams. It is using the presented overlay for location-based area searches. The general idea of *camNet* without the location-based search functionality is described in (Liebau, Heckmann, Hubbertz, Steinmetz, 2005). Since that paper, we developed and integrated the overlay network described in this paper.

In *camNet*, individual peers are computers that can have one or more webcams connected to them. A metadata based search described in (Liebau et al., 2005), allows the users to search for keywords that describe webcams, like "coffee", "white house". A second type of search allows the location-based area search with our overlay network described in this paper. With this, users can detect all webcams in a certain area.

The limited space of this paper does not allow discussing more implementation issues. Instead, we point the interested reader to http://sourceforge.net/projects/camnet where we made the source code of the prototypical application available.

Figure 8 shows a screenshot of the application after the start. If a location-based search is performed (see Figure 9), the results are visualised in a map as shown in Figure 10. From the found webcams a picture can than be requested. Figure 11 shows the metadata based search and a picture of a found webcam.

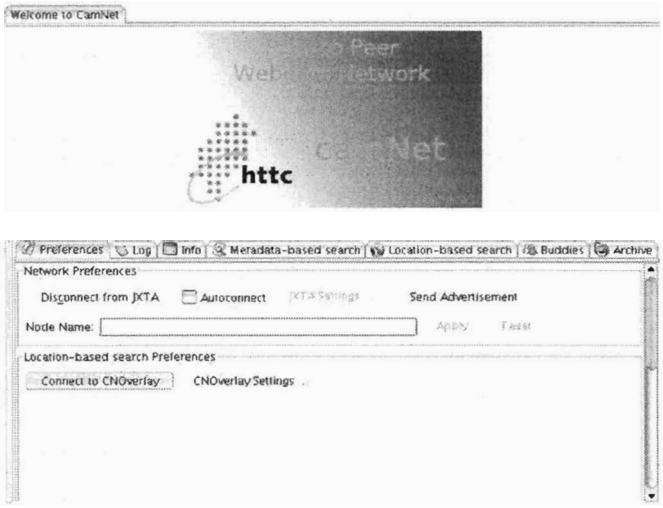

Figure 8: camNet GUI screenshot

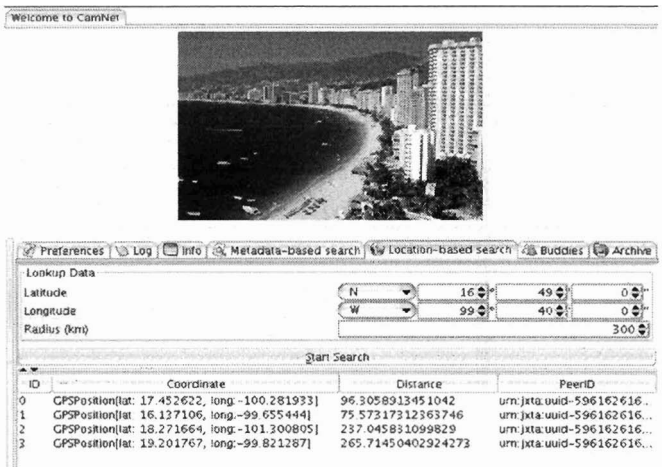

Figure 9: Screenshot after having found a nice picture from Acapulco.

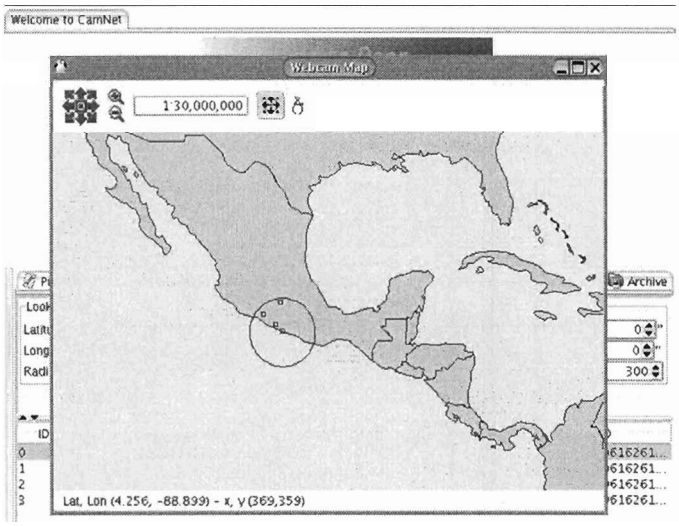

Figure 10: Webcam map showing the location of the look-up results.

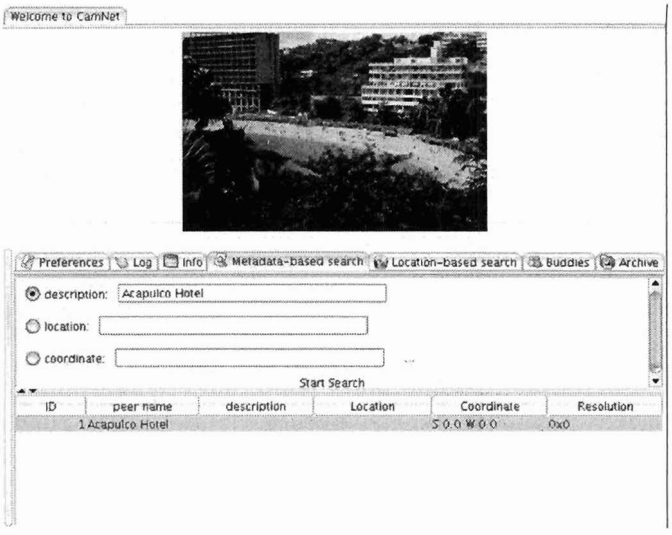

Figure 11: Finding a picture of a hotel in Acapulco with the metadata-based search.

CONCLUSIONS AND OUTLOOK

In this paper, we described a novel peer-to-peer overlay that is specialised for location-based search. It is based on a distributed and interconnected B-tree-like overlay structure. It supports load balancing directly at the design level.

As proof of concept, we integrated it network in our P2P webcam application. As future work, we continue to improve and test the robustness, stability and efficiency of the overlay network and its usage for other types of services. So far, the overlay network was shown to run stable with 1000 peers in emulation mode, further experiments in a full scale simulator are currently under development.

ACKNOWLEDGEMENTS

The work on this project is partly financed by the HMWK (Hessisches Ministerium für Wissenschaft und Kunst / Hessian Ministry for Science and Art).

REFERENCES

1. Byers, J., Considine, J., and Mitzenmacher, M. (2003) Simple Load Balancing for Distributed Hash Tables. In *Proceedings of Peer-to-Peer Systems II: Second International Workshop*, IPTPS 2003 Berkeley, Lecture Notes in Computer Science, Volume 2735 / 2003, 80 – 87.
2. Darlagiannis, V. (2005) Overlay Network Mechanisms for Peer-to-Peer Systems. *PhD thesis*, TU Darmstadt, Computer Science Department, Darmstadt, Germany, 14.07.2005.
3. Google Local Service (2006) Homepage, http://maps.google.com/
4. Google Maps Mania (2006) Homepage, http://googlemapsmania.blogspot.com/
5. Heutelbeck, D. (2002) Context Spaces - Self-Structuring Distributed Networks for Contextual Messaging and Resource Discovery. In *Proceedings of the Tenth International Conference on Cooperative Information Systems (CoopIS'02)*. University of California Irvine, USA.
6. Heutelbeck, D. (2005) Distributed Space Partitioning Trees and their Application in Mobile Computing. *PhD thesis*, Fernuniversität Hagen, Hagen, Germany.
7. Ljungstrand, P. (2001) Context Awareness and Mobile Phones, Personal & Ubiquitous Computing 5(1), 58-61.

8. Liebau, N., Heckmann, O., Hubbertz, I., and Steinmetz, R. (2005) A Peer-to-Peer Webcam Network. In *Proceedings of Peer-to-Peer Systems and Applications Workshop associated with KiVS'05*, 151-154.

9. Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. (2001) A Scalable Content-Addressable Network. In *Proceedings of the 2001 ACM SIGCOMM Conference*, San Diego CA, USA, 161-172.

10. Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H. (2001) Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, San Diego, CA, USA, 149-160.