GeoPeer: A Location-Aware Peer-to-Peer System*

Filipe Araújo
Faculdade de Ciências
Universidade de Lisboa
filipius@di.fc.ul.pt

Luís Rodrigues
Faculdade de Ciências
Universidade de Lisboa
ler@di.fc.ul.pt

July 19, 2004

Abstract

This paper presents a novel peer-to-peer system that is particularly well suited to support context-aware computing. The system, called GeoPeer, aims to combine the advantages of peer-to-peer systems that implement distributed hash tables with the suitability of geographical routing for supporting location-constrained queries and information dissemination. GeoPeer is comprised of two fundamental components: a Delaunay triangulation used to build a connected lattice of nodes and a mechanism to manage long range contacts that allows good routing performance, despite unbalanced distribution of nodes.

1 Introduction

The importance of context-aware services has grown significantly in the last decade. Context-aware computing focus on enriching applications with contextual information, like position, user activity, nearby people and devices,

*Sections of this report will be published in the Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications (IEEE NCA04), August, 2004, Cambridge, MA, USA. This work was partially supported by LaSIGE and by the FCT project INDIQoS POSI/CHS/41473/2001 via POSI and FEDER funds.

time of day or weather conditions [5]. In this paper, we are particularly interested in supporting location-aware services, as location is one of the most important parameters that can be extracted from context information.

Examples of location-aware services include querying for specific resources available in a geographic area (for instance, looking for a restaurant or a hospital in a given neighborhood), reading and integrating information collected by sensor nodes in a given region (for security purposes or environmental monitoring), and disseminating notifications to all nodes in a given region (to multicast warnings about natural or human-induced hazards, such as floods, chemical leaks, etc). Several other example applications of context-aware computing are described in [5].

A significant body of work exists on services and infrastructures for supporting location-aware mobile applications [17] but also on location-aware sensor-network services [19]. Most of these architectures rely on the existence of stationary nodes connected to the wired network infrastructure. This paper is mainly concerned with the scalability of the network of stationary nodes that provide support to very large-scale location-aware services (possibly, in cooperation with mobile nodes and wireless sensors). To the best of our knowledge, the scalability, decentralization, and dynamic aspects of the stationary infrastructure supporting location-aware computing have been overlooked in the literature.

To address the scalability problem, we propose to use a peer-to-peer system, implementing a distributed hash-table, to store context-related information in a distributed manner. Unfortunately, as we will discuss in the related work section, existing peer-to-peer systems such as Pastry [24], Tapestry [29], Chord [26], D2B [8], Koorde [13] or Viceroy [18], do not own the characteristics required to support location-aware services. Systems such as CAN [22], TOPLUS [10], eCAN [28], and the Delaunay triangulation proposed by Liebeherr et al. [16], are closer in spirit to GeoPeer but, as we will discuss later, they also lack features which are essential to support location-aware services in an efficient manner.

Faced with the limitations of previous work, we propose a novel peer-to-peer architecture that works as an overlay network on top of IP, called GeoPeer. Nodes of GeoPeer arrange themselves to form a Delaunay triangulation augmented with long range contacts, established by a mechanism called Hop Level. With the aid of this mechanism, GeoPeer achieves short path lengths. In particular, our experimental results suggest that the node degree/path length trade-off of GeoPeer is logarithmic/logarithmic, independently of node distribution, which is of crucial importance to a location-aware system.

Unlike most other systems, the use of geographical location is inherent to GeoPeer. Therefore, GeoPeer owns a number of interesting properties: it is capable of providing location-awareness in fundamental operations performed by applications, such as reads, writes or queries. For instance, inheriting from the techniques used by Liebeherr et al. in [16], we can augment multicast messages or flooded queries with scope information to limit their range, e.g., when raising an alarm after some accident. Queries containing significant regional information can also benefit from our system: the collection of information may be performed by a local proxy on behalf of the client that may be located far away from the region of interest to the query. For instance, someone in Lisbon may be searching for restaurant information in New York: instead of making all the replies traverse the Atlantic, a proxy in New York could aggregate all the replies and send back a single message to the originator of the query.

The remainder of the paper is organized as follows: Section 2 overviews previous work. The GeoPeer architecture is described in Section 3. The details of its long range contacts mechanism is described and evaluated in Section 4. Some applications of GeoPeer are referred in 5. Section 6 concludes the paper.

2 Related Work

There is a significantly wide body of research which is relevant to the GeoPeer architecture, including work on context-aware computing [12], work on wireless *ad hoc* networks, namely location-aware routing schemes [27] including Delaunay triangulations [15, 14], and work on peer-to-peer systems. Given the nature of our contributions, we limit ourselves in the following paragraphs to the discussion of previous work on peer-to-peer systems and on its suitability to support location-aware services.

In recent years, many peer-to-peer systems providing distributed hash tables holding (key, value) pairs have been proposed [21, 24, 29, 26, 8, 13, 18, 22, 10, 28, 11]. In [21], Ratnasamy *et al.* proposed a data-centric storage for sensor networks, called Geographic Hash Table (GHT). Values are stored and retrieved from the storage through a key that is hashed to a geographical location. Although some similarities exist between GHT and

GeoPeer, important differences apply, because energy, memory or processing effort concerns are of primary importance in GHT, while in wired peer-to-peer systems, like GeoPeer, routing efficiency is more important. Routing efficiency is typically characterized by a number of key properties such as path length, node degree and node congestion, which are, in fact, the most important aspects in the evaluation of a peer-to-peer system. It is known that for constant node degree, the best path length that can be achieved is $O(\log n)$, while for $O(\log n)$ node degree, the path lengths can be reduced to $O((\log n)/(\log\log n))$ [13].

Although optimal or near-optimal in terms of routing efficiency, systems such as Pastry [24], Tapestry [29], Chord [26], D2B [8], Koorde [13] or Viceroy [18] are not eligible to efficiently support location-aware services: they all use unidimensional random addresses that cannot directly represent a concrete physical location. Furthermore, the use of geographical locations as node identifiers would break the uniform distribution of the name space on which these systems rely to offer properties such as limited path lengths. Unlike these, SkipNet [11] has the ability to organize data by string names (e.g., domain names). Hence, a SkipNet is more appropriate to restrict location of a resource or scope of a query to an organization where nodes share some common domain name, like "someexistingname.com". However, it cannot handle physical locations, which means that searching for a nearby facility or placing a resource near its users, if these users are not within the same organization, is not easily addressed by a SikpNet.

On the other hand, systems like CAN [22], TOPLUS [10] or the Delaunay triangulation proposed by Liebeherr et al. [16] could, in principle, be adapted to meet our goals. However, adaptation of TOPLUS would require some mechanism to translate IP addresses into geographical locations, because TOPLUS organizes peers into groups of IP addresses. CAN would have to be converted to use real geographical positions instead of virtual ones, which would, again, break identification and load balancing. But the main problem with both, CAN and Delaunay triangulations, is the large network diameter, which is an important drawback. This inconvenience can be mitigated with the use of long range contacts. This approach has been followed in the design of eCAN [28], which extends the basic CAN architecture with a complex Long Range Contacts (LRCs) management scheme (in eCAN LRCs are called "expressways"). However, the basic CAN is not as efficient as Delaunay triangulations in the support of multicast, because the average node degree is smaller. Moreover, eCAN embodies a number of sophisticated mechanisms, not strictly required for location-aware services, that introduce a significant processing overhead when compared to the LRC scheme we adopt in GeoPeer. Furthermore, the Hop Level mechanism we use is almost oblivious to node distribution, which is a central aspect in a system where physical location dictates the virtual identification of a node.

Given that no straighforward adaptation of previous peer-to-peer systems is able to provide efficient support to location-aware services, we have opted to design a new peer-to-peer architecture, called GeoPeer, that captures the most positive aspects of previous work. This new architecture is described in the next section.

3 Architecture of GeoPeer

3.1 Overview

In GeoPeer nodes self-organize into a planar Delaunay triangulation augmented with carefully selected long range contacts (LRCs) to significantly reduce path lengths. A graph based on a Delaunay triangulation has the following desirable characteristics:

1. expected $O(1)$ node degree;
2. good nearby routing performance; and
3. simple distributed construction.

In GeoPeer, the identification of a node corresponds to its physical location. The combination of these features results in a peer-to-peer system with the following unique advantages:

– by creating a mesh of nodes identified by their physical location, support for applications that execute location-aware operations, such as queries or broadcasts, can be provided by very simple mechanisms;

– when compared with a two-dimensional CAN-like network, the node degree in a Delaunay triangulation should be greater, but still $O(1)$ in expectation (near 6 instead of 4 in perfectly balanced cases) and, therefore, nearby routing should be improved;

– due to the LRCs that augment the Delaunay triangulation, GeoPeer has logarithmic path lengths for all network sizes we tested. Moreover, our LRCs management schemes is elegant, but nevertheless, adaptive to unbalanced use of both physical and identification space.

GeoPeer may also be applied to manage arbitrary objects by using a one-way hash function to compute keys from some relevant object attribute. Keys correspond to positions in space and, therefore, hashing some value representing an object yields the GeoPeer identification of that object. Since identification and position are equivalent, the hash function returns a pseudo-arbitrary position in space. Therefore, location information attributes may be used to carefully position resources in some application dependent way, e.g., by enforcing the use of a hash function that returns some node near the clients of a service.

3.2 Main Components

In the following paragraphs, the main components of the GeoPeer architecture are described in detail. These components are:

1. an algorithm that creates and maintains the Delaunay triangulations;
2. an algorithm that ensures that any possible key is held by exactly one existing node;
3. an algorithm that performs routing of messages in the overlay network; and
4. a mechanism tho establish LRCs.

We start by describing the first three mechanimsms. The discussion of the long-range contact mechanism is postponed to Section 4.

3.3 Notation and Definitions

In the remaining of text we will use the following conventions: nodes and points will be represented by capital letters , e.g., A; edges are represented by the two nodes that define them, for instance, AB; a triangle defined by nodes A, B and C is represented as $\triangle ABC$; the circumcircle of $\triangle ABC$ is represented as $\bigcirc ABC$; an angle ($< \pi$) between line segments AB and AC defined at A is interchangeably represented as $\angle BAC$ or $\angle CAB$ — the

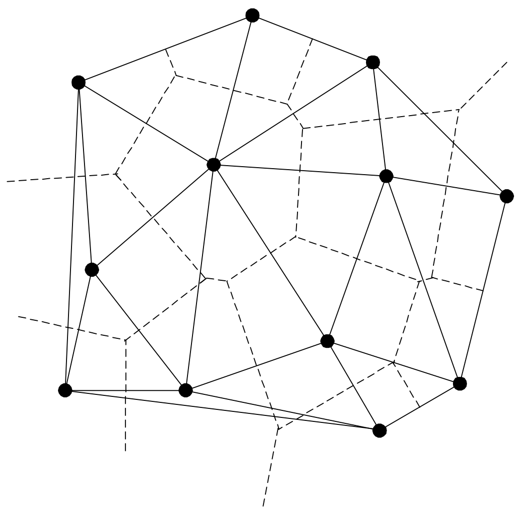

Figure 1: Voronoi cells vs Delaunay triangulation

vertex where the angle is measured stays in the middle, while the position of remaining vertices is arbitrary;

A triangulation of a node set V is called a *Delaunay triangulation* if the circumcircle of each of its triangles does not contain any node of V [15, 14]. The Voronoi cell is a dual concept of the Delaunay triangulation, defined as follows: the Voronoi cell of node P is the set of points in space that are closer to P than to any other node. If the Voronoi cells of two nodes share a common border, then a Delaunay edge exists between those two nodes. This relation is illustrated in Figure 1 where borders of Voronoi cells have dashed lines, while Delaunay edges have solid lines.

3.4 Creation and Maintenance of Delaunay Triangulations

To create and maintain the Delaunay triangulation, GeoPeer uses a scheme similar to [16] (note however that, unlike GeoPeer, [16] does not use LRCs, which we introduce in Section 4). Note that many constructions proposed for wireless *ad hoc* networks, such as [15, 9], are not applicable in this context, since they assume static settings for triangulation and they assume that nodes are provided with broadcast-capable radios.

3.4.1 Messages

To create and maintain the Delaunay triangulations, nodes periodically exchange messages with their geographic neighbors. The five message types exchanged by the algorithm are:

– the BEACON message, used by a node to inform its neighbors that it is still actively participating in the overlay network;

– the JOIN message, used to add new nodes to the network;

– the FAILURE message, used to disseminate information about the failure or departure of a node;

– the TRIANGULATE message, used by a node to propose the setup of a Delaunay triangle with its neighbors;

– the BREAKLINKS message, used to reconfigure the network in response to new joins and leaves.

The purpose and function of each of these message types will be detailed in the following paragraphs. The algorithm does not require channels to be perfect: all messages that are critical to the convergence are retransmitted periodically, if no appropriate reply is received.

3.4.2 Steps

The algorithm is decentralized, as it does not rely on any single point of control. It consists of three logical steps:

1. the *neighbor discovery* step. Node N initiates this step to enter the network. To join, node N must use some out-of-band mean to discover one node already participating in the network, say P. P will then forward a special JOIN message on behalf of N destined to N. Since N does not yet belong to the network, the JOIN message will be received by some node X. This node X will forward the JOIN message to all the Delaunay neighbors of N that X knows about (to inform them of the existence of N) and will also reply with another JOIN message to N with the list of those Delaunay neighbors (note that this step does not establish the triangulations);

2. the *neighbor maintenance* step. In this step, nodes that belong to the same triangle periodically exchange BEACON messages to inform their neighbors that they are alive and actively participating in the network;

3. the Delaunay *triangulation* step. This is naturally, the most complex step of the algorithm.

Based on the information collected in the previous steps, each node P

computes a Delaunay triangulation using its own local knowledge. As a result, P may find out that there should exist a Delaunay triangle $\triangle PN_1N_2$, between P, N_1 and N_2. In this case, for convenience of exposition, we say that the predicate $\mathit{Delaunay}\triangle_P(N_1, N_2)$ is true at P.

When $\mathit{Delaunay}\triangle_P(N_1, N_2)$ holds at P, P sends a TRIANGULATE $\triangle PN_1N_2$ message to both N_1 and N_2. When P receives a TRIANGULATE $\triangle PN_1N_2$ from N_1, if $\mathit{Delaunay}\triangle_P(N_1, N_2)$ holds then P replies to N_1 with another TRIANGULATE message, otherwise, P replies with a BREAKLINKS message including all nodes that it believes should triangulate with N_1.

Therefore, if all neighbors agree on the triangulation, they will exchange a consistent set of TRIANGULATE messages and the corresponding Delaunay triangles are set-up. Otherwise, they update their local information using the contents of the BREAKLINKS message and re-execute the local computation. Note that if there is some node inside $\bigcirc PN_1N_2$, the predicate $\mathit{Delaunay}\triangle_P(N_1, N_2)$ is immediately switched to false. A very simple way of checking this condition was presented by Sibson in [25]. Throughout the text we assume that no four nodes are co-circular (co-circularities can be easily addressed by slightly perturbing the position of involved nodes).

3.4.3 Dynamic Aspects of the Algorithm

As noted before, to cope with a dynamic topology, the algorithm must take into account the following aspects:

1. the failure of nodes;

2. the emergence of new nodes and, as a consequence, the possibility of nodes having a different view of the network topology.

Node failures and departures are detected through the absence of BEACON messages from that node (to simplify the presentation, we do not distinguish these two events, however departures allow a more gracious way to redistribute the keys). When some neighbor of F detects that node F failed, it recomputes the Delaunay triangulation and sends a FAILURE message to all its Delaunay neighbors. All nodes that are neighbors of F should resend the FAILURE messages of F. This ensures that all Delaunay neighbors of F become aware of its failure. Since network is asynchronous, nodes must store information about the failure of F. Therefore, FAILURE and BREAKLINKS messages include a list of nodes that are known to be failed (possibly empty in the case of a BREAKLINKS message). If after a TRIANGULATE message from P, N replies with a BREAKLINKS message, with indication of some node F that P knows to be failed, P sends a FAILURE message and later retries the triangulation.

It is also possible for nodes to enter the graph at any instant. Assume that P becomes aware of the presence of some new node Q and, as a result of recalculating the Delaunay triangulation, some triangles, $Delaunay\triangle_P(N_1, N_2)$ commute from true to false. In such case P sends a BREAKLINKS message to the vertices of those triangles. If $Delaunay\triangle_P(Q, N_1)$ is true for some node N_1, P will again send TRIANGULATE messages as described before.

3.4.4 Optimizations

For clarity of exposition, we deferred discussion of the following issue: BREAKLINKS and FAILURE messages should not carry indefinitely information about all nodes that failed in the past, as this could become a considerable overhead. Therefore, N only resends information that F failed to some peer node A until A acknowledges. Furthermore, to avoid storing information of some failed node F forever, nodes discard information of F after the expiration of a timeout.

3.5 Space Division

For each point in space there is one and only one responsible GeoPeer node. As depicted in Figure 2a) dividing the space into Voronoi cells might be complicated, because Voronoi cells may cross triangle borders. Therefore, the following algorithm is used to divide the space in GeoPeer: nodes must determine the circumcircle and the perpendicular bisectors for each of its Delaunay triangles, as depicted in Figure 2b). In "well behaved" triangles, where the point O lies inside the triangle, division of the space is straightforward and is done according to the figure. Areas A_A, A_B and A_C cover the entire triangle and define the set of points that are, respectively, closer to A, B and C. If the point O lies outside the triangle, such a division is still possible. However, in this case, two of the areas will not share a common border. Trivial tie-breaking mechanisms, not involving any communication, are used to define to which node belong the points in the perpendicular bisectors and points dividing two different triangles. In the borders of the plane, where no

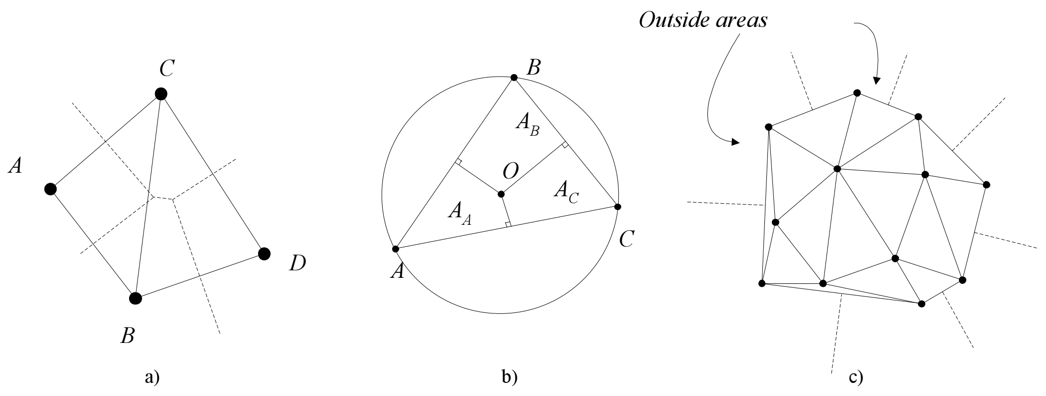

Figure 2: a) Voronoi cells (dashed lines) cross triangle boundaries, b) Circumcircle, c) Outside areas

further triangulations are possible, proximity criterion is used to determine the areas of responsibility of the nodes, as depicted in Figure 2c).

3.6 Basic Routing

Routing between GeoPeer nodes (not considering LRCs) resumes to routing in a Delaunay triangulation. A myriad of routing algorithms that are typically used in the context of wireless *ad hoc* networks can be used in a Delaunay triangulation, namely compass, randomized compass, greedy or Voronoi [27, 3]. In GeoPeer, we have opted to use the greedy algorithm to route messages. It has the following decisive advantages: *i*) it ensures convergence in a Delaunay triangulation [3], *ii*) it is efficient in most circumstances [15] (although Bose and Morin [3] showed some cases where performance is arbitrarily bad, these examples should be rare pathological cases); and *iii*) greedy routing algorithm copes with LRCs without any modification.

If routing in GeoPeer was based exclusively on the use of a greedy algorithm, the resulting network operation would be inefficient due to the large network diameter of the underlying Delaunay triangulation; namely, communication in the GeoPeer network would have a very large latency. Therefore, we need to enhance GeoPeer with a mechanism capable of reducing path lengths, by creating a series of long range contacts (LRCs). Clearly, LRCs play a critical role in GeoPeer as they can significantly reduce path lenghts, thus improving performance. In the next section we describe the LRC mechanism of GeoPeer and compare its performance with an alternative mechanisms proposed for the CAN network [28].

4 Efficient Routing and Long Range Contacts (LRCs)

The goal of establishing LRCs is to reduce path lengths, that are typically $O(n^{1/2})$ in a two-dimensional CAN or in a Delaunay triangulation. By carefully selecting LRCs, it is possible to effectively reduce path lengths while, at the same time, maintaining a limited expected node degree. If a typical, near-optimal, node degree/path length trade-off in evenly balanced overlay networks is logarithmic/logarithmic, if the node distribution in the virtual space id becomes uneven, these strong path length properties are unlikely to hold (as discussed in Section 2 one notable exception are the SkipNets [11]). This issue is particularly relevant in the context of GeoPeer, because distribution of nodes is very likely to be uneven, due to the equivalence between geographical position and virtual identification. Therefore, in this Section we will overview the two alternative mechanisms to manage LRCs that fit best to the bi-dimensional space of GeoPeer: *Hop Level* presented in [1] and expressway CAN (eCAN) [28], intended to improve CAN. In both cases, greedy routing algorithm is used, and the neighbor closest to destination, which can be either a LRC or a nearby neighbor, is used as the next hop. Our experimental evaluation shows that Hop Level mechanism is a more suitable approach for the network sizes we tested.

4.1 Hop Level Mechanism

In the *Hop Level* mechanism, nodes try to avoid doing more than a predefined number of hops, say b hops. For instance, if some node A_1 is forwarding a message from N and if A_1 is the b-th hop of the message, N should create a LRC to A_1. To do this, A_1 sends a message prompting N to create a LRC to itself. The process is repeated, this time starting at A_1: if after b hops, message reaches A_2, A_1 will create a LRC to A_2, and so on. Now suppose that the following series of b LRC is created: NA_1, A_1A_2, $\ldots$, $A_{b-1}A_b$. In this case, a new LRC from N to A_b would be created. This new LRC would be one level above of the others, i.e., while LRC NA_1, A_1A_2, $\ldots$, $A_{b-1}A_b$

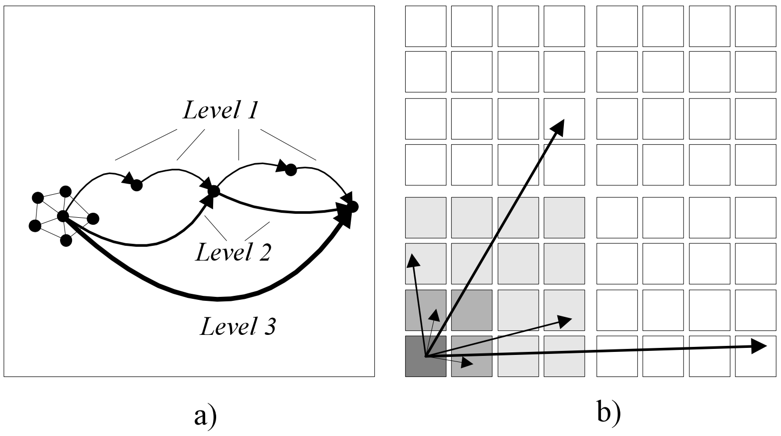

Figure 3: LRCs: a) Hop Level, b) eCAN-like

are of level 1, LRC NA_b is of level 2. The creation of LRC, possibly with increasing levels, is recursively repeated until message reaches its destination. Figure 3a) illustrates the main idea. Hence, b hops at level $l-1$, fire the creation of an LRC at level l, to ensure that an LRC of level l jumps over b^l hops. In practice we used $b = 2$, but for further implementation details reader is referred to [1].

Interestingly, it is not necessary to configure some limit to the number of LRCs admissible per node, because this number is typically limited by the dimension of the network. In fact, if the distance of the LRCs of a given level to the origin is correct, the LRCs should spread in a perimeter around the origin node (this perimeter does not necessarily look like a circle, because nodes can be unevenly distributed), until no additional LRC of that level is required. However, there are pathological cases where it might be necessary to limit the number of LRCs to ensure that this number stays within the logarithmic boundary. This issue, as well as a background mechanism to ensure that LRCs have the correct distances to the origin are still ongoing work. The experiments presented in Section 4.3 were ran in a setting with no limitation or correction of LRCs. We believe that even without such mechanisms, results achieved by Hop Level are quite satisfactory and point to an easy deployment of this mechanism in real scenarios.

4.2 eCAN-like Mechanism

To enable a comparison with eCAN, we use a mechanism that builds LRC in a way that closely resembles the expressways of eCAN. We must emphasize that this mechanism is an extremely simplified version of the complete eCAN solution, that only captures the fundamental impact of the expressway mechanism in routing, and does not attempt to reproduce other features of eCAN (such as the mechanisms that provide support for complex interaction schemes like publish/subscribe). In spite of these simplifications, we believe that our implementation of expressways mimics the eCAN LRC mechanism with enough accuracy to allow a fair comparison.

Hence, the idea is to make a first level division of the entire space in four big squares. Each node keeps LRC to two of these four squares: one to some node that is in the square above/below, the other to some node that is in the square at right/left. Then, the four big squares are further divided in other four smaller squares. This time, some of the squares in the middle may have a total number of four LRC (above, below, right and left). This process is repeated for as many levels as wanted. Figure 3b) illustrates the eCAN-like LRC scheme. In our context, we fixed the number of levels to 8, in a total of 30 LRCs. It should be noticed that, since the center of a given square will probably not correspond to any existing node, the actual LRC will be the node responsible for that center point.

4.3 Comparison

In this section we summarize a comparison between Hop Level mechanism and eCAN. The most important result of such a comparison is the average path lengths achieved by both mechanisms in balanced and extremely unbalanced scenarios, for several network sizes. To do this comparison we used networks with sizes ranging from 100 to 50000 nodes. To unbalance the nodes we used a truncated Gaussian bivariate distribution with standard deviations ranging from 1.0 to 0.01 (we only show results for the most unbalanced distribution). Results depicted in Figure 4 clearly shows that eCAN does not obtain logarithmic path lengths (which should appear as a straight line) for unbalanced scenarios. The reason for this is that 8 levels are too many in zones where node density is small, and too few near the center where node density is large. This means that the only safe way to configure eCAN is to use a brute-force approach capable of covering the entire virtual id space.

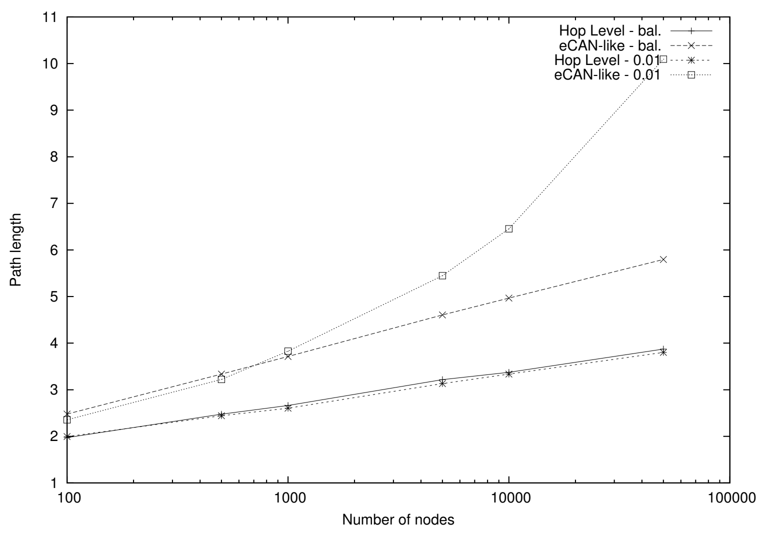

Figure 4: Comparison of path lengths

However, this will make the number of LRCs per node grow logarithmic to the space, *not to* the number of nodes. On the contrary, Hop Level is not considerably affected by node density and, as shown in Figure 5, the number of LRCs grows logarithmically, for the network sizes considered. Therefore, we believe that in the context of GeoPeer, Hop Level mechanism is the most adequate mechanism to establish LRCs, because it is oblivious to node density and thus greatly precludes the need for manually configured parameters in order to achieve a logarithmic/logarithmic node degree/path length.

5 Applications of GeoPeer

GeoPeer may, as any other decentralized peer-to-peer system, be used to support any sort of application that benefits from a scalable implementation of a distributed hash table, such as, for instance, decentralized storage services [6, 7, 23]. However, some of the characteristics of GeoPeer, like location-awareness and uneven distribution of nodes, make it specially fit for the support of location-aware services. We now illustrate the benefits of the architecture by giving some examples of context-aware services that can be trivially implemented on top of GeoPeer and that can benefit from the reduced diameter of a GeoPeer network.

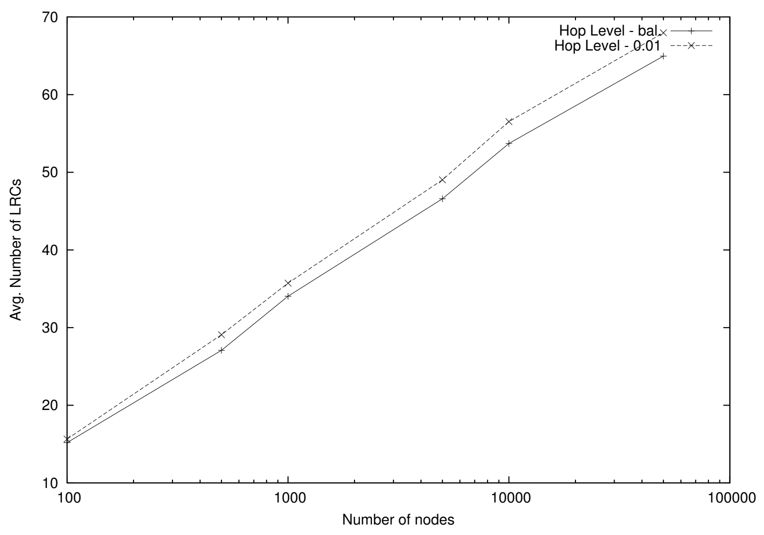

Figure 5: Average number of LRCs

Geographically-scoped multicast. This service consists in disseminating a notification to all nodes located inside a given geographic region (e.g., an alarm about some natural disaster). The service can be easily implemented by routing a notification to the GeoPeer nodes responsible for the center of that area which will, in turn, initiate a scoped-broadcast of the notification, using the technique proposed in [16]. It should be noted that, with the exception of a bi-dimensional CAN (and variations like eCAN) no other peer-to-peer system would directly support this service. Furthermore, the use of Delaunay triangulations makes GeoPeer more efficient than CAN or eCAN.

Geographically-scoped queries. This service is the counterpart of the previous one. It is used to collect information from nodes located inside a given geographic region (e.g., environmental or security monitoring of geographical areas by connection of the relevant sensors to the GeoPeer nodes). It can also be used to collect more mundane information, such as the location of cinemas or bars in the vicinity of a given location. The service works by having the node responsible for the center of the region of interest acting as an ambassador of the client. This node can efficiently query all nodes in a given diameter, collect all the replies, and send the consolidated information back to the client in a single message (this may involve computation of averages, selection of the lowest or highest values, etc.).

Other location-aware services. GeoPeer also opens new less obvious possibilities for applications that need to determine location of critical resources, like a rendezvous point in a core-based multicast tree [2] or in publish-subscribe applications [20, 4]. A complete exploration and evaluation of such solutions is beyond the scope of this paper.

All these applications require a routing scheme that allows a message to be addressed to a given geographical location. GeoPeer offers this functionality and LRCs allow routing to be achieved with a small number of hops.

6 Conclusions

We have presented a novel peer-to-peer architecture called GeoPeer, intended to provide services to location-aware applications. These services include geographically-scoped multicasts or geographically-scoped queries, which can be used by applications willing to spread some alarm within a limited region or applications searching for local facilities, like restaurants or cinemas, for instance. Additionally, location properties offered by GeoPeer can be explored by other more complex applications, like core-based multicast trees or publish-subscribe systems, to improve location of critical resources.

The fundamental core that supports location-aware routing in GeoPeer is a Delaunay triangulation of nodes augmented with a LRC mechanism called Hop Level. With the help of this mechanism, experimental results suggest that GeoPeer achieves logarithmic path lengths for logarithmic node degree, independently of node distribution in the space, a characteristic that is of crucial importance to build a scalable location-aware peer-to-peer system.

Acknowledgements

The authors are thankful to Patrick Eugster and Rachid Guerraoui for their comments on earlier versions of this paper.

References

[1] Filipe Araújo and Luís Rodrigues. Long range contacts in overlay networks with unbalanced node distribution. DI/FCUL TR 04–8, Department of Informatics, University of Lisbon, July 2004.

[2] A. Ballardie. Core based trees (cbt version 2) multicast routing. Request for Comments 2189, September 1997.

[3] Prosenjit Bose and Pat Morin. Online routing in triangulations. In *10th Annual Internation Symposium on Algorithms and Computation (ISAAC)*, 1999.

[4] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 2002.

[5] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[6] J. Douceur and R. Wattenhofer. Optimizing file availability in a secure serverless distributed file system. In *Proceedings of 20th IEEE SRDS*, pages 4–13, 2001.

[7] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *HotOS VIII*, Schoss Elmau, Germany, May 2001.

[8] P. Fraigniaud and P. Gauron. The content-addressable network D2B. Technical Report 1349, LRI, Univ. Paris-Sud, France, Jan 2003.

[9] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanners for routing in mobile networks. In *2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, 2001.

[10] L. Garcés-Erice, K.W. Ross, E.W. Biersack, P.A. Felber, and G. Urvoy-Keller. Topology-centric look-up service. In *COST264/ACM Fifth International Workshop on Networked Group Communications (NGC)*, Munich, Germany, 2003.

[11] Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Fourth USENIX Symposium on Internet Technologies and Systems (USITS '03)*, Seattle, WA., March 2003.

[12] Jeffrey Hightower and Gaetano Borriella. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[13] Frans Kaashoek and David R. Karger. Koorde: A simple degree-optimal distributed hash table, 2003.

[14] Luan Lan and Hsu Wen-Jing. Localized delaunay triangulation for topological construction and routing on manets. In *2nd ACM Workshop on Principles of Mobile Computing (POMC'02)*, 2002.

[15] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *The 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[16] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with Delaunay triangulation overlays. Technical Report CS-2001-26, University of Virginia, Department of Computer Science, 5 2001.

[17] Henning Maass. Location-aware mobile applications based on directory services. *Mobile Networks and Applications*, 3(2):157–173, 1998.

[18] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Twenty-First ACM Symposium on Principles of Distributed Computing (PODC 2002)*, Monterey, California, July 2002.

[19] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, and Mani B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM*, pages 1380–1387, 2001.

[20] P. Pietzuch and J. Bacon. Hermes: A distributed event-based middleware architecture. In *22nd IEEE International Conference on Distributed Computing Systems Workshops (DEBS '02)*, 2002.

[21] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: A geographic hash table for data-centric storage in sensornets. In *First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, September 2002.

[22] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Conference on applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[23] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz. Maintenance-free global data storage. *IEEE Internet Computing*, 5(5):40–49, 2001.

[24] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[25] R. Sibson. Locally equiangular triangulations. *The Computer Journal*, 21(3):243–245, 1977.

[26] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *ACM SIGCOMM*, San Diego, August 2001.

[27] Ivan Stojmenovic. Position-based routing in ad hoc networks. *IEEE Communications Magazine*, July 2002.

[28] Zhichen Xu and Zheng Zhang. Building low-maintenance expressways for p2p systems. Technical Report HPL-2002-41, HP, 2002.

[29] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.