Towards Autonomic Networking Using Overlay Routing Techniques

Kendy Kutzner, Kurt Cramer, and Thomas Fuhrmann

System Architecture Group, Universität Karlsruhe (TH), Karlsruhe, Germany
{kendy.kutzner,curt.cramer,thomas.fuhrmann}@ira.uka.de

Abstract. With an ever-growing number of computers being embedded into our surroundings, the era of ubiquitous computing is approaching fast. However, as the number of networked devices increases, so does system complexity. Contrary to the goal of achieving an "invisible computer", the required amount of management and human intervention increases more and more, both slowing down the growth rate and limiting the achievable size of ubiquitous systems.
In this paper we present a novel routing approach that is capable of handling complex networks without any administrative intervention. Based on a combination of standard overlay routing techniques and source routes, this approach is capable of efficiently bootstrapping a routable network. Unlike other approaches that try to combine peer-to-peer ideas with ad-hoc networks, sensor networks, or ubiquitous systems, our approach is *not* based on a routing scheme. This makes the resulting system flexible and powerful with respect at application support as well as efficient with regard to routing overhead and system complexity.

1 Motivation and Introduction

The Autonomic Computing vision outlines a future in which computing systems are self-managing. By this, the systems' administrative complexity is expected to be greatly reduced. Autonomic Computing techniques therefore lend themselves to the implementation of large-scale ubiquitous systems. There, networking plays a fundamental role as the systems' complex functionality only emerges from the *cooperation* of the otherwise less powerful embedded devices. When comparing the anticipated traits of Autonomic Computing with those achieved by peer-to-peer (P2P) networking, striking similarities can be discovered. We therefore see great opportunities in developing autonomic networking systems targeted at Ubiquitous Computing by employing P2P techniques.

In this paper, we present an important building block for the successful deployment of that idea, a novel routing scheme based on the combination of peer-to-peer overlay routing and source routing in the underlay. This combination is capable of pooling tens of thousands of small, network-enabled, autonomous devices – each with limited processing and memory capabilities – into a routable

M. Beigl and P. Lukowicz (Eds.): ARCS 2005, LNCS 3432, pp. 222–235, 2005.

network. The protocol is fully self-organizing and self-optimizing, has a low per-node control message overhead, and yields efficient path lengths as compared to globally optimized shortest-paths networks. Moreover, being based on distributed hash-table addressing, the resulting network also efficiently supports higher-layer functionality like look-up services, etc.

To our knowledge such an approach has not been studied well in the literature so far. Few authors already proposed to replace the Internet Protocol (IP) by a peer-to-peer overlay [4],[3]. But no definite results have been achieved by them yet. Hu et. al. proposed in [8] to combine Dynamic Source Routing with a DHT, however they were not able to eliminate the potential flooding of the whole network.

1.1 The Benefit of Overlays

One key aspect in the use of overlay networks is their capability of introducing an application specific addressing that abstracts from the physical necessities of the underlying network [2]. While in the Internet addresses should reflect the structure of the physical network, applications need to address logical entities. This is best seen with distributed hash tables (DHTs). There, logical addresses (identifiers, keys) are obtained by hashing the name of an object into a very large key space. The overlay network (e.g. CAN [12], Chord [16], Pastry [13], or Kademlia [9]) then maps these logical addresses to physical network nodes, thereby hiding the physical structure of the network. Applications can use this abstraction to, e.g., distributedly store files. Although this is probably the most popular use of overlay networks by now, other uses might be even more powerful. An example is the Internet Indirection Infrastructure [15], that employs the logical address space to create network services for mobility support, group communication, and the like.

This *separation of concern* is most useful in the area of pervasive and ubiquitous computing. For this reason many authors are combining peer-to-peer concepts with mobile ad-hoc networks, sensor networks, etc. ([7], [6], [14] and references therein) In general, these approaches employ a two-layered approach, where the basic connectivity is provided by some off-the-shelf routing protocol, like fish-eye routing [10], ad-hoc on-demand distance vector routing [11] or the ones compared in [1]. On top of this already fully routable underlay network, a separate overlay network of peer-to-peer mechanisms is established. While this provides the full benefit of the logical overlay addressing, this two-layered approach gives rise to inefficiencies. E.g., the nodes have to use their scarce resources for *two* different routing protocols, overlay *and* underlay. Moreover, the network is burdened with the control message overhead of *two* routing protocols. These inefficiencies are avoided by our approach, that inherently combines overlay and underlay routing. Before describing this combination, we briefly review those key aspects of overlay and underlay routing that form the ingredients for our novel protocol.

1.2 A Brief Review of Underlay and Overlay Routing

Routing can be viewed as being the task of applying general knowledge of a network's structure to a concrete packet (frame, message, datagram). Such a packet originates at one node of the network (source) and is destined to another node of that network (destination). Routing schemes can, among other properties, be classified according to *where* the knowledge of the network structure is stored:

With many dynamic routing approaches, especially with *distance vector routing*, but in essence also with *link state routing*, all the nodes in a network build up their specific part of the global knowledge that allows them to route a packet *towards* its destination. Each node stores its knowledge in a routing table. Thereby, these dynamic routing schemes benefit greatly from a well-organized address space that allows the nodes to aggregate large chunks of the address space into a single routing table entry. But with real networks, especially with organically growing ubiquitous networks, the aggregability of the address space deteriorates quickly. Especially the low-resource devices targeted in pervasive computing are not capable of storing and handling large routing tables.

Overlay routing with distributed hash tables, on the other hand, circumvents the aggregation problem by choosing peers that allow a perfect routing table. A simple algorithmic rule determines which peer is responsible for which address. Hence, the routing table has exactly p entries when a node has p peers. However, to work properly, a DHT overlay must be able to connect to all the peers prescribed by the algorithmic rule. This is no problem in overlays where an underlay routing mechanism provides full connectivity in the network. But it seems to be impossible to achieve in physical networks, where the neighbors of a node cannot be chosen at will.

Contrary to this seemingly obvious statement, we will show how a simple extension of the standard overlay approach can indeed create a DHT-like routing scheme in arbitrary networks. To this end, we combine a typical DHT routing (Chord) with two well-known underlay routing approaches: mutual state exchanges (as with distance vector routing protocols, DVRP) and source-routing. Both approaches are known to have severe drawbacks: DVRP scales badly to larger networks, while source-routing can only be employed in combination with some other routing means. But, as we will show, both obstacles are also overcome by the combination with the DHT-routing mechanism. (For the time being, we will not address other problems often associated with source routing, like e.g. security issues. These are severe problems in open networks with malicious users, but they are much less severe in our target scenario as described in the following section.)

1.3 Target Scenario Description

Before we describe our proposed routing approach in section 2 and report first results on efficiency with respect to traffic overhead and achievable path length in section 3, we briefly describe the scenario for which we designed this routing

protocol. We also envisage our protocol being applied outside this specific scenario. Nevertheless, we also believe this particular application to be of growing importance in the near future.

Our novel routing approach targets networks of some tens or hundreds of thousand autonomous networked nodes that are connected by some physical communication links, be it wired or wireless. A typical example is a network of sensors and actuators in a large office building, a large factory plant, or a large municipal utility. The nodes have limited memory and processing capabilities. They are deployed redundantly, so that individual nodes may fail at any time, but we assume that they do so only at a moderate rate. We will, however, address the case that the entire system is shut down and re-started, as might happen with a power outage.

Individual nodes might be moved in the network (i.e. be detached and re-attached elsewhere), but we do not assume all the nodes to be truly mobile to a large extent. Moreover, we assume the nodes to be fully cooperative, i.e. we do not consider malicious nodes in this paper. This important issue is to be addressed in future work.

We assume the nodes to be redundantly linked to other nodes, typically in their vicinity. Communication requirements are assumed to be repetitive and of small data-rate. E.g. a light-switch will mostly signal to its associated lamps, the temperature sensor will mostly signal to the air-conditioning in the same room.

Additionally, many typical applications in the field of pervasive computing will benefit from the logical address space that the overlay aspects bring into our routing scheme. E.g. nodes can employ indirection mechanisms for service discovery, as well as for multicast, anycast, or concast. Moreover, addresses may be randomly assigned numbers, i.e. they can be hashes of device names, public cryptographic keys, etc. all of which greatly simplifies many applications in such a network (see [4] for a praise of these benefits).

We believe that the described network properties are typical for the described scenario of a large, but closed system. Although here centrally administered approaches would be applicable in principle, we do not believe that they are practically feasible in the in the light of a significant roll-out of such small autonomous embedded communication devices. To the contrary, we strongly believe, that only truly self-organizing systems will be able to pave the way to the large-scale deployment of distributed computing devices in all kinds of appliances. A technician installing such a device will well understand that it needs to be connected to the communication network, but he will probably not know the network topology, respect its original layout, register the device with a central facility, etc. Experience shows that in practice, people tend to apply quick hacks, i.e. they want to simply attach a device and expect it to work right away, and this is exactly what our approach accomplishes.

2 An Overlay-Underlay Routing Combination

We consider a network of randomly connected nodes with randomly assigned addresses. (The respectively employed definition of "randomness" is explained in section 3 where the simulation results are discussed.) Such a fully random network is the worst case assumption for a scenario like ours. In order to not complicate the description of our protocol, we begin with the assumption that the links are reliable point-to-point links. (Link and node failures as well as shared media will be addressed in section 2.3.) For simplicity, we also assume the network to be always connected and the addresses to be unique within this network. Provided sufficient redundancy, the former will hold for *almost every* network graph in the studied scenario. The latter is a typical assumption made with overlay networks where the address space is much larger than the number of nodes. Even if the address space was populated so densely that a collision became likely, conflict detection and resolution would be simple with standard overlay means (e.g. by having the neighbors of a newly attached node query for the node's address before accepting it as a neighbor).

Now the proposed protocol works as follows:

2.1 Bootstrapping Phase

Upon powering up a node, the node sends a HELLO_1 message to all its physical neighbors. They reply with a HELLO_2 message. If a new link is introduced between two existing nodes, both will send a HELLO_2 message. Both types of hello messages carry the respective node's address. Thus, after all hello messages have been exchanged, all nodes know the addresses of all their physical neighbors. This mechanism is known as *Neighbor Discovery*. There may be hardware technologies which support neighbor discovery and where therefore this mechanism is not necessary. We included it for completeness.

The neighbors' addresses are inserted into a constant size routing table. For an address space of 2^m unique addresses, each node's routing table contains m regular entries and one special purpose entry (see below). The organization of this routing table is such that the ith (regular) entry at a node with address a points to a node with address b whose distance $d(a,b) = (b-a) \bmod 2^m$ is at least 2^i and less than 2^{i+1}. In other words, the address space is circularly connected with a defined orientation so that distances are *asymmetric*. The larger i, the larger the chance that addresses fall into the respective interval. (Note that the 0'th entry can only be populated in the unlikely case that another node's address is the immediate successor of the node's address.) If more than one node is to be stored in the same entry of the routing table, one of them has to be chosen. An extension is to keep more than one up to k entries. The tradeoffs included are described in section 2.3. Upon start-up, when only physically neighboring nodes are known, this choice which nodes to keep has to be made at random. Later on, when the node has accumulated more "knowledge" about the network, the "better" node is chosen, where the term "better" is yet to be defined (see below).

Each entry in the routing table points to one node in the network. The entry itself is a string of node addresses, which describes a source route to the respective node. Upon start-up, all address strings trivially have length one. Later on, the strings will grow in size. The simulation results presented in section 3 show that for networks with 10 000 to 100 000 nodes, the length of these paths is on the order of the network diameter.

As soon as all physical neighbors have been entered into the routing table, the node sends all the nodes in the routing table an UPDATE message containing the full table. (It can do so, since by definition of the routing table it has either a source route to these nodes, or the destination is a direct neighbor.) Upon reception of an UPDATE message, the respective receiver takes that source route from the received table that leads to itself, inverts it and prepends the inverted source route to every other entry in the received table. If this address string concatenation leads to loops in the resulting path, these loops are cut so that the shortest possible path is formed. Finally, all the obtained paths are compared to the node's own routing table. Besides a new entry in the routing table there is one other reason to send an UPDATE message. Upon reception of an UPDATE message, the receiving node answers with such an message. It is important that these in turn don't lead to further messages. To avoid such an UPDATE-storm, every UPDATE message carries a flag whether the receiver should reply with another UPDATE message.

If a received path is better than the locally stored path for a given regular entry, the respective entry in the routing table is replaced. Here, a new entry is considered to be better, either if so far no path was available locally for the respective entry of the routing table, or if the new path is shorter. (Note that by replacing an entry in the routing table, that entry now points to a different node!)

The special purpose entry is used to store the path to the node's successor in the address space, i.e. the node with the smallest distance. This is necessary since otherwise the path to the successor might be replaced by a slightly more distant node (in the logical address space) which happens to have a shorter path (in physical network hops). Although this case is rare, it would lead to not being able to route to some of the nodes in the network (e.g. the successor).

After the routing table has been updated, the node sends a new UPDATE message to all the nodes currently stored in the routing table. (No update is sent if updating left the routing table unchanged.) In order to avoid an inflation of update messages, generation of the update is delayed by a timeout T_{update}. During this time potential further update messages can be processed. In a stable network without node churn, after a while, consistency will be achieved and no further updates will be sent. In real-world networks where nodes come and go, there will always be new update messages to be processed. Nevertheless, even there a sufficient level of consistency will permanently prevail in the network. Because newer and shorter routes are preferred by the selection algorithm, the network is constantly optimizing itself without the need for a central authority. Note that there are no flooded messages necessary to discover all routing table

entries, which is a clear improvement for the scalability of the system compared to previous aproaches.

Very quickly after start-up of the entire network, the update mechanism fills the routing table. (Clearly, due to the sparsely populated address space, the low-index entries of the routing table typically remain empty.) In the further course of operation the length of the stored paths shortens until an (almost) stable state has been reached. But even before that time, the system is able to route to arbitrary destinations.

2.2 Routing Phase

The routing table that results from the bootstrapping phase has all properties required by the standard Chord routing algorithm. Accordingly, routing can now be done in almost exactly the same manner as described with Chord. The only difference is that we are using the source route paths that are stored in the routing table instead of the transport addresses of a regular Chord. The distinction between overlay and underlay present in Chord directly maps to the use of two different packet types for routing in our proposed system: Type 1 packets contain a source route and can thus be directly forwarded towards the respective destination. Type 2 packets only bear the destination address. In order to be forwarded, a node must look up a source route. (In the Chord analogy, type 1 packets are TCP/IP packets and type 2 packets are overlay packets.)

Consider the following example: A node with address a has to forward a type 2 packet to a destination node with address b. To this end, it looks up that source route path from the lower $\lfloor \log_2 d(a,b) \rfloor$ entries of the routing table that has the shortest path length. This selection algorithm equals PRS with path length as metric. It then encapsulates the type 2 packet into a type 1 packet with that source route. (From a networking point of view this is in fact exactly the same as what happens in normal overlay routing schemes. There, the overlay messages are "encapsulated" into TCP segments and IP packets.)

As has been shown before [5], such an approach leads to a highly efficient routing with path lengths that are only slightly larger than those of the optimal paths. Unlike the classical use as an overlay routing protocol *on top of an already* existing network, our protocol does not require any underlying routing mechanism. This is a large benefit since it enables routing in scenarios like those described in section 1.3. Nevertheless, we want to also investigate the performance of our protocol as compared to a hypothetical optimal shortest path routing. (Note, that neither full-topology nor distance vector approaches are feasible in our target scenario. Hence, this is only a benchmark indicating what could be possibly achieved with much more resources in the individual nodes.)

2.3 Extensions for Practical Use

So far, we have presented the principles of our proposed new approach. Before we present simulation results indicating that this approach indeed has the required properties, we briefly sketch some extensions to the protocol that will be required

for practical use or allow tradeoffs between memory, number of messages and efficency. These extensions are subject of our current research efforts.

In real networks, packet loss is common due to node and link failures, queue overflows or problems on the link layer. Moreover, we expect in practice a moderate rate of node churn, i.e. nodes being attached and detached from the network. Given that the rate of such events is sufficiently low, our protocol is able to handle these situations. The argument is as follows: As will be demonstrated in the next section, our protocol converges rather quickly even for large networks. As long as the churn rate is below the convergence time, there will be no problem at all. If the churn rises, a growing number of inconsistencies will prevail in the network. At the same time, update messages do not cease any more, but perform constant maintenance of the network. I.e. routing table entries become soft-state. The actual thresholds which churn or packet loss rates lead to which rate of update messages and which level of correctness, are, however, to be determined by future simulation work.

The protocol described above maintains at most one source route per address space interval. An obvious extension is to keep more routes per interval, or to increase the number of intervals (provided the interval lengths are increasing logarithmically). This increased redundancy can be expected to reduce the achievable paths lengths and improve convergence. But it might also increase the amount of update traffic. The investigation of this trade-off, too, is subject of our ongoing work.

Another objection to the practical use of our protocol as described above might be that we described it only with point-to-point links. We believe this to be a minor technical detail. When the underlying hardware is based on a shared medium, as is typically the case in wireless scenarios, both the neighbor discovery and the exchange of update messages are accelerated since now several nodes can learn from each HELLO or UPDATE message. Since the protocol does not contain acknowledgments, etc., both message types can be considered as point-to-multipoint without further change. Moreover, the concept of source routes that always explicitly addresses the next hop during packet forwarding immediately carries over to shared media.

Finally, to reply to yet another potential objection with regard to the practical use of our proposed approach, we briefly mention the protocol overhead implied by using source routes. Consider a network with, e.g., a diameter of 20 hops and an address space size of 64bit. Then the source route can be expected to lead to a 160byte header for each packet. This is definitely not suitable for the low-resource devices of our target scenario. But, as stated in section 1.3, we expect communication to be repetitive and typically limited to a small set of destinations. Hence it is worth while to generate compressed headers, where the source route does not contain globally unique addresses but only labels that are locally unique to the respective nodes along the path. (We cannot elaborate the technical details here. See text books on ATM or MPLS for an explanation of this standard idea.) In a network with a node degree of $2^4 = 16$, this will lead to a packet header of only 4bit per source route hop.

3 Simulation Results

In order to be able to judge the achieved efficiency of our proposed routing protocol we implemented it as a simulation, focusing on the core aspects: the number of update messages and the length of the created source routes. The simulation program's source code is available from the authors on request.

The network's topology was modeled as random graph with constant node degrees of 3, 5, 10 and 20. (In detail: To ensure connectedness of the whole graph, first, a random permutation of all nodes was connected in form of a ring. Then additional links were entered between randomly drawn nodes until all nodes had acquired the respective degree.) Node addresses were drawn uniformly random from a 32bit address space, and duplicate address assignments were explicitly excluded. For each node degree various network sizes were simulated, ranging from 100 to 20 000 nodes.

At the beginning of each simulation run, all nodes were equipped with a routing table, each containing the node's physical neighbors (as it would result from exchanging the HELLO messages). Then the set of all nodes was traversed in consecutive rounds, according to the random permutation of the initial ring. For each node all the pending update messages were processed, and the resulting routing table sent to the nodes stored in the table.

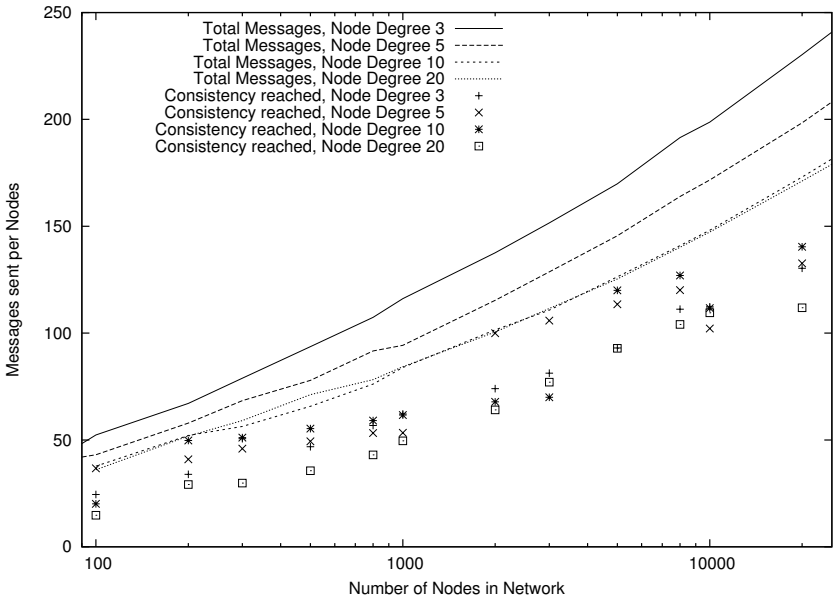

Fig. 1. Messages per node until consistency is reached

Figure 2 shows the average number of UPDATE messages per node that were exchanged during the various simulation runs as (normal, dotted, dashed) lines. Graphical approximations show that the number of UPDATE messages grows with $O(\log^2(n))$. A network of 10 000 nodes produces between 140 and 200 messages per node, depending on the degree of the nodes.

Long before all UPDATE messages have been exchanged, the system has already reached global consistency, where global consistency means that all nodes have stored their correct successor. The respective number of UPDATE messages needed for global consistency is shown as cross (star, box, plus) in figure 2. E.g., the 10 000 node networks reach global consistency after about 100 messages per node.

Once global consistency is achieved, the system is capable of correctly routing all packets. However, since global consistency can only be detected if global knowledge is available, the nodes have to keep sending UPDATE messages until these messages do not cause further changes in the regular routing table entries.

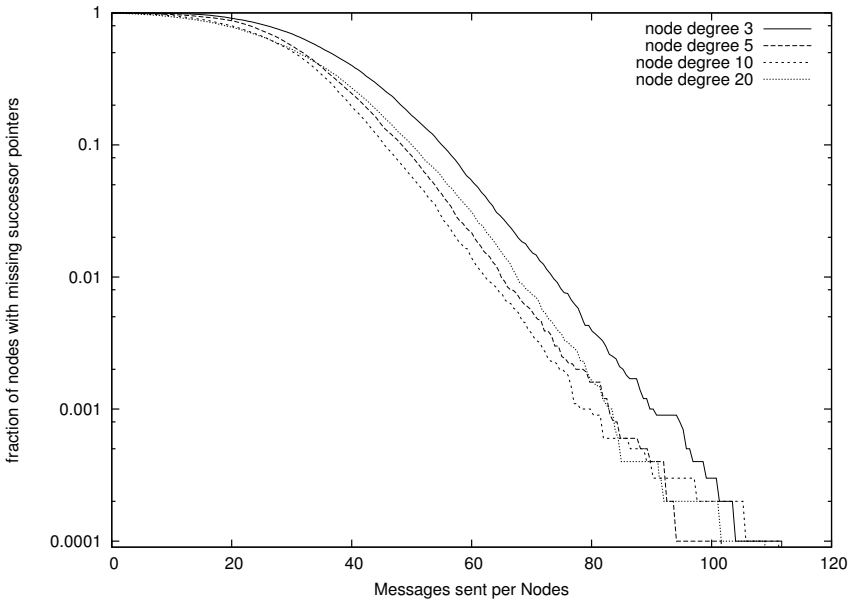

Fig. 2. Convergence to Consistency in a Network of 10 000 nodes

But even before reaching global consistency, the system is already able to correctly route most of the packets. Figure 2 shows how consistency proceeds during the simulation runs for the 10 000 node networks with node degrees of 3, 5, 10 and 20. At the beginning, nodes only know their physical neighborhood. Since for larger networks, it is highly unlikely that in this neighborhood there is a correct successor, almost 100% of the nodes lack their correct successor.

After a few rounds of update message exchange, the nodes learn more and more about nodes with similar addresses. (Note that the protocol is constructed to spread this knowledge in the address-neighborhood, not the physical neighborhood. Hence the quick exponential convergence.) The vast majority of the nodes (90%) have correct successor routes after about 40 – 50 exchanged messages. The rest of the time and update messages is spent to correct very few mistakes.

This entire bootstrapping process assumes the entire network to be powered up at once, as might be the case after a power outage. Such an extreme scenario is typically not studied in the field of overlay computing, since there one assumes a gradual extension of the network. However, we consider it a rare, but important "worst" case for our target scenario. Hence, the emphasis.

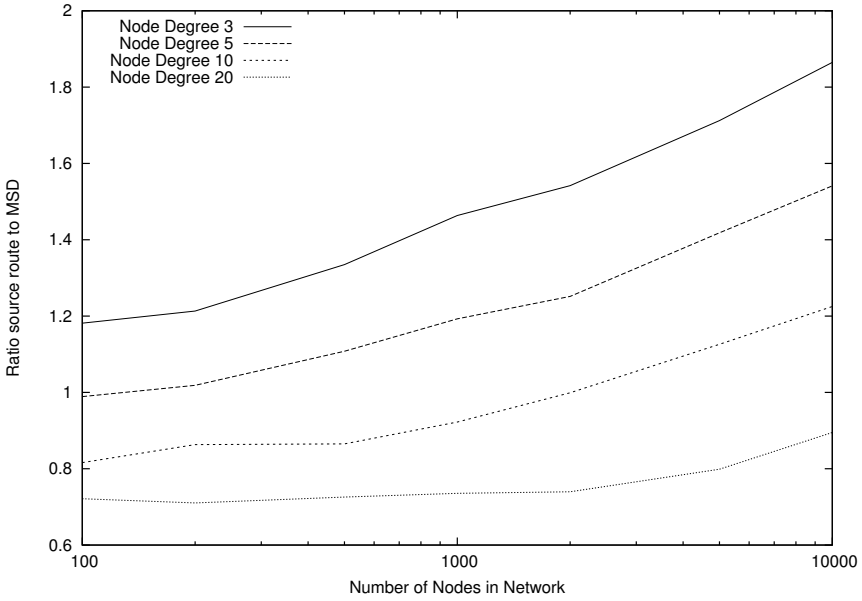

Fig. 3. Ratio of discovered source routes to the mean shortest distance

Our protocol does not aim at finding perfect shortest routes, however we strive to find good ones. Therefore, it is worthwhile to look at how good the selected routes are. Figure 3 shows the routing stretch for a single routing table entry, i.e. the ratio of the length of the source route and the mean shortest distance (MSD) of the graph. MSD is defined as the average of the shortest distance between two randomly selected nodes in the graph. Note the logarithmic scale. Since our algorithm is biased for smaller source routes (*proximity node selection*), it can select source routes shorter than the MSD. As can be seen from the graph, these source routes are typically in the order of magnitude of MSD, growing only logarithmically with the network size.

Note that, like with Chord, for routing to arbitrary nodes, $O(\log_2 n)$ of these source routes have to be traversed consecutively, where n is the number of nodes in the network. But, also like with Chord, there is a freedom of choice which source routes a packet should follow. [5] has shown that this *proximity route selection* (PRS) mechanism typically reduces the overall path length to about the length of the longest individual hop. (We are currently investigating the effects of PRS on our approach. First results are promising.)

4 Conclusion and Outlook

In this paper, we have presented a novel routing approach that is capable of efficiently routing messages in a random graph network where the nodes bear randomly assigned addresses. The latter is an important essence to the implementation of distributed hash tables (DHTs) on which many currently proposed middleware services are based. Like with DHTs, our approach distributes routing knowledge in the entire network so that a full route to an arbitrary destination can be found within $O(\log n)$ steps. It also limits the size of the routing table in each node to $O(\log n)$ entries, despite the random address assignment. Unlike other approaches, that promote the use of overlay techniques in mobile or pervasive network environments, we *do not require any underlying routing protocol at all.* Thereby, we are able to efficiently address much larger systems.

We motivated a target scenario for our propose protocol by considering a network of tens or hundreds of thousands of small embedded sensor actuator nodes that is extended with new nodes by and by. We pointed out that such a scenario highly benefits from a routing algorithm like ours, that is fully self-organizing even with the considered large random network topologies that could not be handled with conventional ad-hoc network or sensor network routing protocols. Moreover, since our protocol provides a DHT-like service to the application running on top of it, typical applications like look-up services, etc. are easy to be implemented efficiently.

We showed with the help of simulations that our approach leads to globally consistent routing tables after $O(\log^2 n)$ update messages where n is the network size. We also showed that partial correctness is achieved rather quickly, and that the number of routing inconsistencies decays exponentially. Furthermore, our simulations indicate that the resulting path lengths are on the order of the network diameter.

Nevertheless, our research towards the use of overlay networking techniques for autonomic networking is still at its beginning. Many interesting questions are to be resolved by future research. We see, e.g., considerable potential for further improving the efficiency of our approach, especially concerning overall path lengths. Additionally, we are currently studying the efficiency of our algorithm in a network with node churn, i.e. where nodes come and go or change their physical position in the network at a moderate rate. Besides these ongoing research activities, we still need to address many important questions like security and accounting, that would yield our approach applicable for open networks, too.

Altogether, we believe that the here proposed combination of overlay and underlay routing techniques is an important step towards the realization of truly autonomic computing systems.

References

1. Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, October 1998.
2. Curt Cramer and Thomas Fuhrmann. On the Fundamental Communication Abstraction Supplied by P2P Overlay Networks. *European Transactions on Telecommunications*, 16:1–9, 2005.
3. Jakob Eriksson, Michalis Faloutsos, and Srikanth Krishnamurty. PeerNet: Pushing Peer-to-Peer Down the Stack. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Claremont Hotel, Berkeley, CA, USA, February 2001. Springer Verlag.
4. Bryan Ford. Unmanaged Internet Protocol. *ACM SIGCOMM Computer Communications Review*, 34(1):93–98, January 2004.
5. P. Krishna Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proceedings of the ACM SIGCOMM Conference*, pages 381–394, Karlsruhe, Germany, August 2003.
6. Tobias Heer, Heiko Niedermayer, Leo Petrak, Simon Rieche, and Klaus Wehrle. On the Use of Structured P2P Indexing Mechanisms in Mobile Ad-Hoc Scenarios. In *Proceedings of the 34. Jahrestagung der Gesellschaft für Informatik*, volume 2, pages 239–244, Ulm, Germany, September 2004.
7. Hans-Joachim Hof, Erik-Oliver Blaß, Thomas Fuhrmann, and Martina Zitterbart. Design of a Secure Distributed Service Directory for Wireless Sensornetworks. In *Proceedings of the 1st European Workshop on Wireless Sensor Networks*, Berlin, Germany, January 2004.
8. Y Charlie Hu, Himabindu Pucha, and Saumitra M Das. Exploiting the synergy between peer-to-peer and mobile ad hoc networks. In *Proceedings of HotOS-IX: Ninth Workshop on Hot Topics in Operating Systems*, May 2003.
9. Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer-Verlag, 2002.
10. Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing in mobile ad hoc networks. In *ICDCS Workshop on Wireless Networks and Mobile Computing*, pages D71–D78, 2000.
11. Charles E. Perkins and Elizabeth M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.
12. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of the SIGCOMM 2001 conference*, pages 161–172. ACM Press, 2001.
13. Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware) 2001*, Heidelberg, Germany, November 2001.

14. Rüdiger Schollmeier, Ingo Gruber, and Florian Niethammer. Protocol for Peer-to-Peer Networking in Mobile Environments. In *Proceedings of the 12th International Conference on Computer Communications and Networks*, Dallas, Texas, USA, October 2003.
15. Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet Indirection Infrastructure. In *Proceedings of the 2002 ACM SIGCOMM conference*, pages 73–86, Pittsburgh, PA, USA, 2002. ACM Press.
16. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the SIGCOMM 2001 conference*, pages 149–160. ACM Press, 2001.