Combining Virtual and Physical Structures for Self-organized Routing

Thomas Fuhrmann*

System Architecture Group, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany
Tel.: +49 721 6086730; Fax: +49 721 6087664
fuhrmann@ira.uka.de

Abstract. Our recently proposed scalable source routing (SSR) protocol combines source routing in the physical network with Chord-like routing in the virtual ring that is formed by the address space. Thereby, SSR provides self-organized routing in large unstructured networks of resource-limited devices. Its ability to quickly adapt to changes in the network topology makes it suitable not only for sensor-actuator networks but also for mobile ad-hoc networks. Moreover, SSR directly provides the key-based routing semantics, thereby making it an efficient basis for the scalable implementation of self-organizing, fully decentralized applications.

In this paper we review SSR's self-organizing features and demonstrate how the combination of virtual and physical structures leads to emergence of stability and efficiency. In particular, we focus on SSR's resistance against node churn. Following the principle of combining virtual and physical structures, we propose an extension that stabilizes SSR in face of heavy node churn. Simulations demonstrate the effectiveness of this extension.

1 Introduction

Many researchers consider routing to be a prototypical example for self-organizing distributed systems. In practice, however, large-scale routing is often not as self-organizing as one might think. For example, the Internet's scalability relies on a suitable assignment of addresses that allows address aggregation in the routers. Self-organizing network systems such as mobile ad-hoc or sensor network routing protocols, on the other hand, mostly limit the network size to a few hundred nodes or limit the routing purpose to, e. g. , data aggregation.

We aim at a related but different scenario: large unstructured networks where the nodes communicate using the key-based routing semantics of structured routing overlays. Our guiding example is a community of digital homes where inexperienced users deploy an organically growing number and variety of networked devices. These devices comprise tiny processing and communication units that interface with sensors or actuators or both. Depending of the respective application these devices shall be able to communicate across the entire network.

* This work was supported by Deutsche Forschungsgemeinschaft under grant FU448-1.

H. de Meer and J.P.G. Sterbenz (Eds.): IWSOS 2006, LNCS 4124, pp. 49–61, 2006.

The following examples shall motivate why we believe that such a scenario will become more and more important in the near future.

Example 1: With SSR, the association of a wall-mounted switch (=sensor) with a light bulb or marquee motor (=actuator) is no longer fixed by physical cabling, but can be (re-)programmed according to the current users' preferences. Since the system is self-organizing, users can arbitrarily add and remove devices without breaking the system. This is more robust than today's centralized building automation systems.

Example 2: Blinds, marquees, etc. must be retracted when heavy rain or a storm approach. Today, each house has its own sensors. With SSR, all the sensor readings from the neighborhood can be combined to yield more reliability for the entire community. This is achieved by exploiting the aggregation tree implicitly formed by proximity aware key-based routing (see below).

Example 3: Sensors in the greenhouse at the other end of the garden need not directly report to the associated display in the living room, kitchen, etc. Any infrastructure (both wired and wireless) can relay the data. If required, the data can be redirected to other locations on demand, e.g. to the person who takes care of the greenhouse when the owners are on vacation.

In the sketched scenarios, typically, wireless links are mixed with wired links. Individual sensors might use (different) radio communication technologies, but there will also be e. g. power line cabling. Using these wires to bridge larger distances reduces both, the energy required by the sensors and the electromagnetic interference problems. Thereby, communication relations can efficiently be stretched across the whole network. Unlike today's building automation networks, the resulting network will lack any simple structure: it will neither be a tree that allows an address structure that can be aggregated, nor will it be a simple 2-dimensional layout, as it is, for example, required to use greedy perimeter geographical routing.

Furthermore, these examples also demonstrate that self-organization is essential for future networks: Users will add and remove devices without taking care of the network structure. These devices may have only limited memory and computation resources. New devices need to be quickly integrated; and devices leaving suddenly and ungracefully should not harm the entire network's functionality. Small, isolated networks consisting of only a few devices may grow together to form a large network with thousands of devices covering a whole neighborhood of digital homes. Scalable source routing (SSR) has been specifically designed for such scenarios.

The SSR protocol has been described elsewhere in detail [6]; and we kindly ask you, the reader, to revert to that publication for any questions regarding the details of the SSR protocol. Here we primarily discuss SSR's self-organizing properties. By that we mean SSR's ability to provide structured routing in large, unstructured networks without central components and without the need for configuration or human interaction. Unlike many state-of-the-art routing protocols for large networks, SSR operates in arbitrary network topologies, i. e. SSR is

self-organizing not only with respect to its operation but also with respect to the physical system setup.

Especially, in this paper, we describe how a simple set of rules creates a globally consistent structure from limited local information. Furthermore, we describe how these simple rules achieve robustness in face of node churn, i. e. when nodes ungracefully drop out of the network. We hope to thereby promote the general understanding for the mechanisms of self-organizing systems, and to foster the application of similar principles in other problems in networking and distributed systems.

This paper is structured as follows. Section 2 briefly describes SSR's core rules. Section 3 discusses SSR's mode of operation in the light of self-organization. Based on this insight, section 4 describes a novel extension to SSR that improves SSR's stability in face of node churn. Finally, section 5 concludes with an outlook to future work.

2 Scalable Source Routing

Scalable source routing (SSR) is a self-organizing routing protocol that provides key-based routing in large, unstructured networks where nodes have limited resources. Its design was guided by the following assumptions that reflect the constraints and conditions of the scenario described above:

1. Nodes can afford only little memory to hold routing state. The memory is assigned statically, i. e. there is no way to acquire additional memory. (Our implementation, for example, requires only 4kB state per node.)
2. The network has no simple topology, i. e. it is neither hierarchical nor necessarily planar. (This is caused by mixing wireline and wireless links.)
3. Nodes have location independent addresses, e. g. assigned by the manufacturer.
4. Nodes may communicate with many different destinations when using the key-based routing semantic.

SSR's key engineering trick is the combination of a virtual structure, the address space, with the actual physical structure (=topology) of the network. Both structures are equipped with a metric. The virtual metric is defined as the absolute value of the numerical address difference, taking the address wrap into account. The physical metric is defined as the length of a source route (i.e. hop count) that connects the respective nodes.

As a result, SSR combines the beneficial properties of other routing approaches:

Like typical on-demand protocols such as AODV [13] or DSR [10], SSR does not need to maintain routing state for all the nodes in the network. This keeps the required routing state small and does not burden an otherwise (nearly) idle network with unproportionally high control traffic.

Like typical link state or distance vector protocols such as OSPF or RIP, once consistency is achieved, SSR can route any message without the need to acquire

or build up routing state for a previously unknown destination. This keeps the routing delay low and avoids buffering the messages.

Moreover, SSR does *not need to flood* the network with route request messages (contrary to e. g. AODV and DSR); and it does *not need to assign special node identifiers* according to the nodes' position in the network (contrary to e. g. LANMAR [12], Safari [2], and Beacon Vector Routing [4]). Thereby SSR avoids the use of additional look-up protocols which, again, reduces the control traffic associated with such look-up mechanisms. Moreover, SSR does not make any assumption about the network topology. Thus SSR can efficiently combine wireless and wireline links. This is unlike e. g. *Greedy Perimeter Stateless Routing* [11] where the network graph is assumed to be planar.

SSR shares some ideas with previous work by various authors: DPSR [9] combines the Pastry overlay with DSR. Ekta [14] enhances DPSR with indirect routing. But unlike SSR both protocols need to flood the network to discover source routes. MADPastry [17] combines Pastry with AODV to achieve indirect routing. Unlike SSR, MADPastry needs to maintain a key-to-node mapping which causes significant control traffic when nodes are mobile. Several other authors have proposed similar uses of overlay structures for the maintenance of location directories [3,5,16].

One of SSR's key advantages is that the protocol is so simple that it can be compactly implemented on resource-limited embedded controllers. Its core rule is based on the Chord [15] idea: Nodes have location independent addresses that are viewed as virtual ring.

> A message is forwarded so that its virtual distance to the destination decreases monotonically.

Clearly, this is not always possible with a node's direct physical neighbors alone. Hence, a node maintains a cache with source routes which correspond to the 'fingers' in the Chord overlay. (See below which source routes are entered into the cache.) >From that cache the node picks a so-called next *intermediate node* to which the message is then forwarded using the respective source route.

> When choosing the next intermediate node, physically close nodes are preferred over virtually far nodes.

As a result, SSR builds up a source route in the message header that spans from the message's source to its destination including the intermediate nodes. Note that, when appending a source route, loops are cut out of the source route so that a minimal spanning tree remains.

> Intermediate nodes enter the source routes that are conveyed in the messages into their cache. This cache is operated in a least-recently-used manner.

Those familiar with the Chord overlay routing protocol and its proximity route selection extension [8] will recognize that the just described procedure allows efficient *key-based routing* (KBR). KBR is a generalization of unicast routing, i. e. SSR is capable of both, KBR and unicast routing. With KBR messages

are delivered to that node whose address is numerically closest to the message's destination. KBR can serve as foundation for many distributed applications, like for example distributed hash tables (DHT).

Nevertheless, as known from Chord, correct unicast routing and consistent key-based routing is only guaranteed if each node stores correct fingers (here: source routes) to its virtual neighbors, i. e. its successor and predecessor in the address space ring.

Before we describe why SSR is in fact able to self-organizingly obtain and maintain these source routes, we briefly discuss figure 1 which illustrates SSR's mode of operation. In practice, in such a small network all nodes would quickly acquire the full topological knowledge of the entire network. For our example here, we assume that nodes have only very limited information, i. e. each node knows only its direct physical and virtual neighbors.

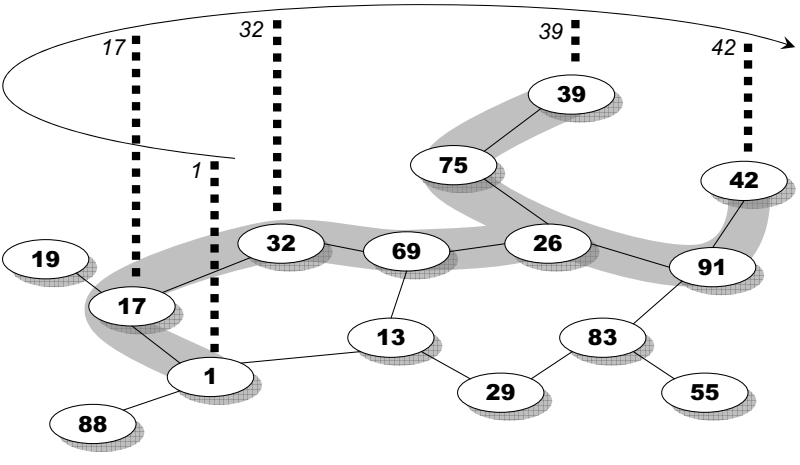

Fig. 1. Illustration of Scalable Source Routing

Assume further that node 1 wants to send a message to node 42. The message is first sent from node 1 to node 17, because among the physical neighbors of node 1 node 17 is virtually closest to the destination. (Note that $42 - 17 = 25$ is smaller than $42 - 13 = 29$ and $88 - 42 = 46$.) For the same reason, node 17 forwards the message to node 32.

So far, the routing process has only involved direct physical neighbors. At node 32, none of the physical neighbors is virtually closer to the destination than node 32, the current intermediate node. Thus, node 32 must append a non-trivial source route. According to our assumption, node 32 caches a source route to its virtual neighbor, node 39, so that the message can be accordingly forwarded to node 39. Node 39, finally, can append a source route to node 42.

As can be seen from figure 1 the result of this routing process is a spanning tree that connects the intermediate nodes 1, 17, 32, 39, and 42. (Note that for

simplicity of terminology, we include the source and destination node in the set of intermediate nodes.) This tree is conveyed in the message header, so that the intermediate nodes, especially the destination node, can enter the source routes to the previous intermediate nodes into their caches. Thereby, the nodes acquire more knowledge about the network.

In our example, node 42 has acquired a source route to node 32. When node 42 sends a reply via node 32 to node 1, node 32 will acquire a source route to node 42. Thus subsequent messages from node 1 to node 42 will not have be routed via node 39 any more. The nodes in the network are 'learning' the relevant source routes.

Note that in order to reduce the overhead in message headers, the source route tree can be pruned, so that it contains only the most recent intermediate nodes. A detailed analysis shows that these are the most relevant nodes of this self-organized cache improvement process.

3 SSR's Self-organizing Properties

The discussion of the example from figure 1 has already shown that once the nodes have source routes to their direct virtual neighbors, SSR's caching rule has the effect that the nodes also accumulate source routes to (some of) their indirect virtual neighbors. In presence of payload traffic, these source routes are frequently used and therefore remain in the caches. Nevertheless all state is soft state and can thus adapt to changes. (See below for a discussion of cache maintenance in presence of node churn.) Moreover, unlike e. g. Chord, SSR does not employ an explicit stabilization method, but self-organizingly builds the respectively required knowledge. As a consequence, SSR does not necessarily produce control messages in an idle network. This is an important feature in our target scenario.

As mentioned above, SSR only works consistently if all nodes have source routes to their respective direct virtual neighbors. In the remainder of this section, we discuss why this is in fact likely to be the case in practice. To this end, we have to review three emergent properties of SSR.

1. First, we note that due to SSR's preference for physically close nodes, each of a node's direct physical neighbors will be chosen as a first intermediate node for a particular address space interval, namely the interval covering all positions in the address space for which the respective node is virtually closest. Fig. 2 illustrates the choice for node 1 in our example of fig. 1: Node 1 sends traffic for the interval $[7, 14]$ to node 13, traffic for the interval $[15, 52]$ to node 17, and traffic for the interval $[53, 94]$ to node 88. Traffic for the interval $[95, 6]$ stays at node 1. (We assume nodes to have addresses in $[0, 99]$.)

Correspondingly, node 32 sends traffic for the interval $[51, 92]$ to node 69, and traffic for the interval $[93, 24]$ to node 17. Thus, nodes specialize for the addresses around their own address and attract the respective traffic from their physical vicinity.

This is demonstrated by the simulation results in figure 3. The plot shows nodes (black disks) in a unit disk random graph that all try to route messages

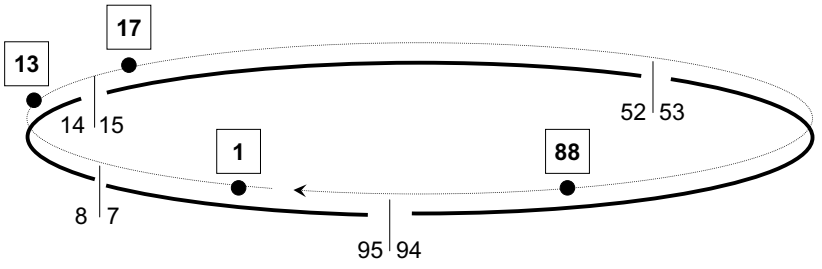

Fig. 2. Illustration of address range specialization

to the same destination. For the plot, this destination was randomly picked. The node density in the graph is chosen to be near the critical density to obtain a connected graph (approx. 10 nodes per radio range disk). The arrows indicate to which of their direct physical neighbors the nodes forward the messages.

Obviously, clusters emerge in the graph, where each cluster forwards the message to one node in the cluster (= cluster head), namely the node that has specialized in the respective destination address. Note that unlike many MANET protocols, SSR does not explicitly determine the clusters and the cluster head. The clusters rather are an emergent property of SSR that is caused by the basic routing rule. Consequently, the cluster formation is robust against node churn. Note further that the clustering will be different for different destination addresses.

Clearly, if SSR had only this first emergent property described so far, the system would be inconsistent in the sense that messages destined to the same address but originating in different clusters would not end up at the same node. Two further emergent properties resolve this inconsistency. By virtue of theses properties SSR can connect (some of) the nodes within the clusters; and it can connect nodes in different clusters via the cluster heads.

2. Upon bootstrapping, a node only knows its direct physical neighbors, and thus has to choose its virtual neighbors from this limited set of nodes. With high probability this leads to inconsistent assumptions among the nodes in the clusters. In the example of figure 1 both, node 17 and node 13, will assume node 1 to be their predecessor. If node 1 is aware of this inconsistency, it can easily resolve it. Thus, SSR requires the nodes to express their assumptions:

> When routing to a next intermediate node that is assumed to be a virtual neighbor of the previous intermediate node, indicate that assumption in the message header.

In the example, upon reception of such a message from node 17, node 1 can inform node 17 of the existence of node 13 by sending a message containing a source route that connects node 1 and node 13. In general, the cluster heads will serve as catalysts for their physical neighbors to find source routes to better virtual neighbors within the cluster. However, this process will only rarely

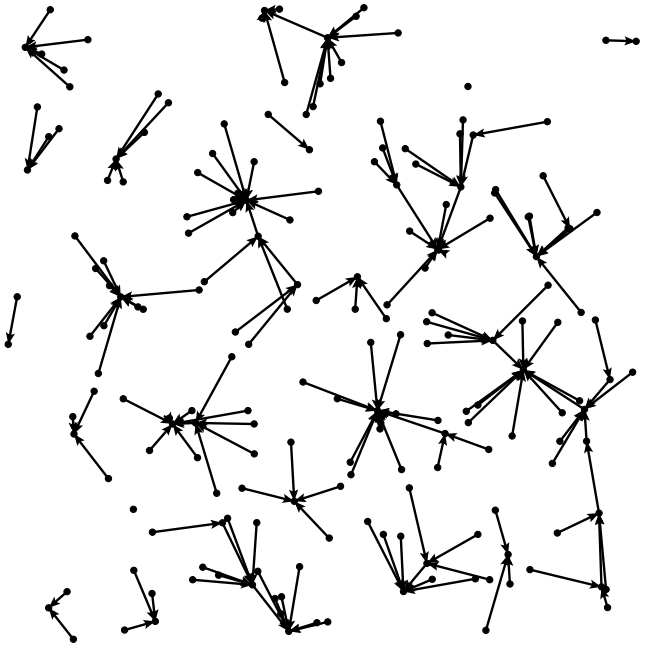

Fig. 3. Example for SSR's self-organized clusters

provide nodes with source routes to their direct virtual neighbors, because these are likely to lie in other clusters.

The observation of a third emergent property explains why this is only an apparent problem:

3. The clusters are slightly different for slightly different destination addresses. Thus, information about the virtual neighbors can be conveyed between the different clusters. In the example of figure 1 node 1 mediates the information that nodes 13 and 17 are virtual neighbors. Before that mediation, messages from node 29 or 69 to node 17 would have ended at node 13, whereas the same message originating at nodes 1 or 32 would have been correctly routed to node 17. In other words, nodes 29 and 69 belong to another cluster than 1 and 32 with respect to destination node 17. After that mediation, all the messages will be correctly routed to node 17, i. e. the clusters (for destination node 17) have become connected by the source route connecting nodes 13 and 17. Furthermore, node 17 can now mediate a source routing connecting the direct virutal neighbors 29 and 32.

Simulations show that even large networks converge quite quickly [6]. Even though a detailed analysis [1] shows that this bootstrapping mechanism cannot always guarantee global consistency, extensive simulations including node mobility and churn [7] indicate that an such inconsistency is unlikely to appear in practice. In fact, any inconsistency in the network is much more likely to be caused by nodes moving or leaving the network and thereby breaking the source

routes that connect the virtual neighbors. We will discuss this problem in the following section.

4 Stability in Face of Churn

To cope with node and link churn, overlay peer-to-peer protocols such as Chord maintain finger tables that contain not only the direct virtual neighbors, but also several indirect virtual neighbors. But with SSR the source routes to the virtual neighbors have a high probability to overlap. As explained above, in the example of figure 1, node 1 mediates the route 13-1-17 from which node 17 then mediates 29-13-1-17-32. Node 1 is thus crucial for the virtual neighborhood of nodes 13, 17, 29, and 32. In general, with SSR often a single node suffices to break the source routes to all direct and several indirect virtual neighbors of a node.

Luckily, a simple trick resolves that problem: Nodes cache not only the source routes to their virtual neighbors, but also store some of those virtual neighbors' physical neighbors. When a route is found to be broken and cannot be salvaged, it is back-propagated to the intermediate node that appended the broken path. There, the outdated source route is deleted from the cache and the message is then routed to one of the physical neighbors of the now unreachable virtual neighbor. Since with high probability that new destination induces an entirely different cluster pattern, the message is likely to follow a completely independent path. Once that new destination has been reached, the message can be trivially forwarded to the original destination because it is by definition a physical neighbor. In order to do so, the original destination has always to be kept in the message header.

We explain this again with the example of figure 1: Assume the link between 26 and 91 is broken, e. g. because node 91 has moved out of node 29's radio range. When the message in the example above cannot be forwarded to node 91, it is back-propagated to node 39, the previous intermediate node. Along the way, the nodes delete the respective link from their caches. If any of the nodes on the path back to node 39 can salvage the source route, i. e. if it knows a source route to a node in the message's source route that lies beyond the broken link, the message's source route is accordingly modified and the message is forwarded along the new source route. In order to update the caches SSR always back-propagates at least a respective control message indicating the broken link together with an alternative route. This is especially important for node 39 whose cache was the cause that the message contained an outdated link. This back-propagation mechanism ensures that the caches are quickly updated when outdated information is used for the routing process.

If the message has been back-propagated to node 39 without having been salvaged, node 39 substitutes the destination 42 with one of node 42's physical neighbors, here with node 91. According to SSR's routing rules, the message is now routed along the intermediate nodes -39-75-83-91. At node 91, the message can then be forwarded to the original destination, node 42.

As a result, SSR achieves a high delivery ratio even in face of heavy node churn. This may however be at the expense of a message being in flight for a

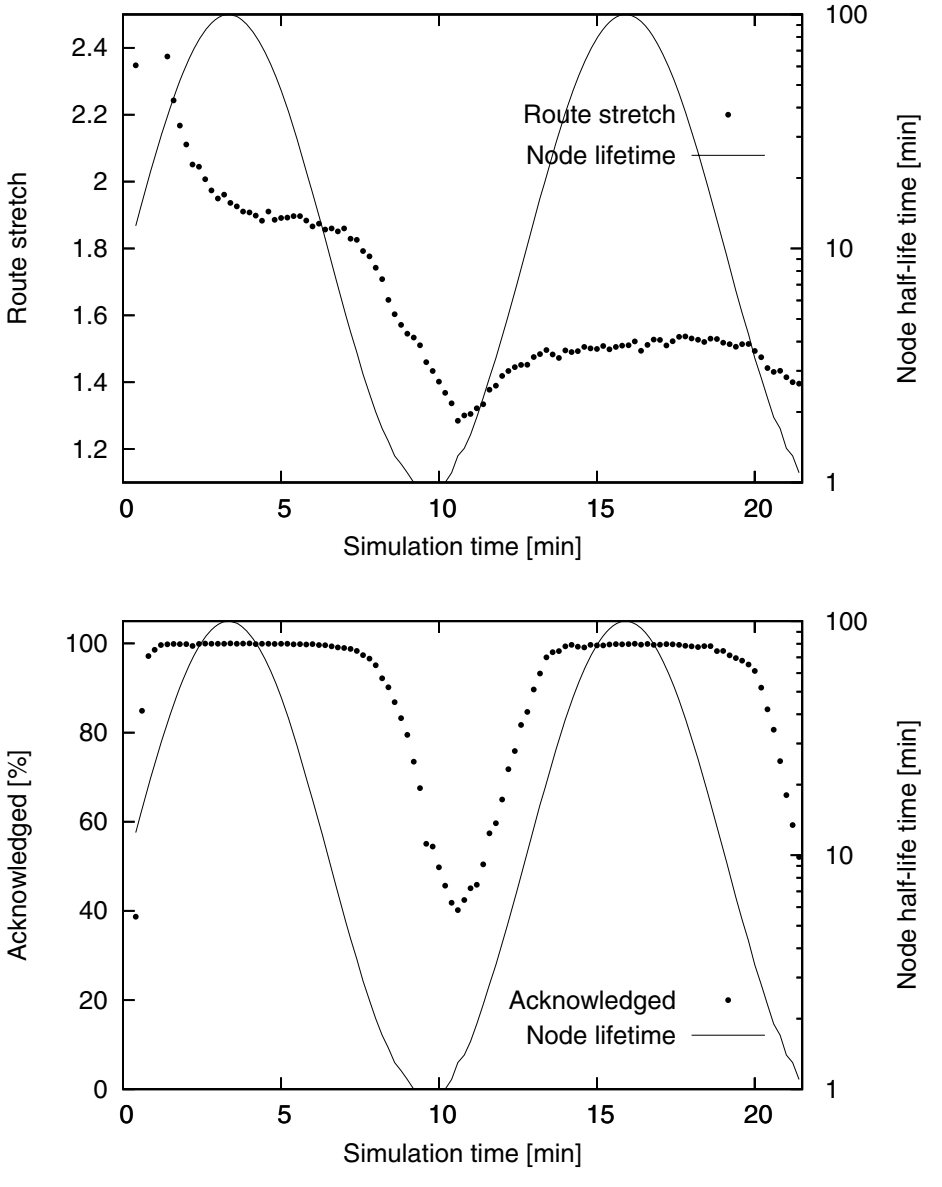

Fig. 4. SSR's performance in face of heavy node churn

potentially long time. Nevertheless, extensive simulations show that SSR is still more efficient than state-of-the-art protocols such as AODV that has to flood the network to discover routes. The comparison of SSR to MANET routing protocols such as AODV is extensively discussed in [7]. Here, we focus on the effect of node churn.

Figure 4 shows results for an 8000 node small-world network with 8.3 kBit/s links. We have chosen this topology here, because it is more affected by node churn than unit disk graph networks, which are typical for MANETS. Moreover, we expect the networks in our target scenario to be exhibit more small-world properties than typical MANETs. The low link rate reflects the capabilities of typical low-cost, low-resource embedded devices.

In our simulation, each node every 12 seconds sends a message to a randomly chosen node (equal distribution). This simulates frequent look-ups in e. g. a DHT built on top of SSR. Upon reception of such a message, the receiving node acknowledges the message by sending back a response to the message's original source. Throughout the simulation nodes are killed according to a Poisson process with a half-life time that oscillates between 60 and 6000 seconds (plotted line). Whenever a node is killed, a new node joins the network (at a random position), thereby keeping the entire number of nodes constant. Note that the oscillating node life time allows us to study the hysteresis in the routing system, an effect that is typical for many self-organizing dynamic systems.

Figure 4 shows the acknowledgment rate and the route stretch (i.e. the ratio to the shortest path determined using the Dijkstra algorithm and global knowledge of the network) for the first two oscillation periods of a typical simulation run. As expected from [6], during bootstrapping, the route stretch peaks at about 2.4 and slowly declines afterwards. The stretch is slightly smaller during heavy node churn periods (i. e. periods with small half-life time) because then longer source routes have a significant probability to break and thus do not contribute to the stretch calculation.

More importantly, the simulation shows that with this extension SSR is capable of maintaining almost 100% correct delivery as long as the nodes' half-life time is above about 10 min. Even if the life time falls below 1 min, SSR is able to correctly acknowledge almost 50% of the messages.

Note that for determining these values we have to take the hysteresis into account. To this end, we compare the rate at a given life-time value when the life time is reducing, with that value when it is increasing. Thereby, we eliminate potential short-term effects and get a good approximation for the long-term behavior for the respective life-time value.

5 Conclusion

In this paper, we have reviewed our recently proposed scalable source routing (SSR) protocol. SSR combines source routing in the physical network with Chord-like routing in the virtual ring that is formed by the address space. In particular, we have explained how SSR's combination of the virtual ring structure with the physical network topology creates a self-organizing routing protocol that is able to efficiently operate in large scale networks. We have pointed out that SSR is especially suited for organically growing networks of small embedded devices that have limited memory and computation resources.

Based on our discussion of SSR's self-organizing features we have proposed an extension to SSR, that stabilizes SSR under heavy node churn. This extension was motivated by the insight, that a single churn event can affect the source routes to several of a node's virtual neighbors, but that it is unlikely to affect the source routes to the virtual neighbors' physical neighbors. In order to confirm the effectiveness of that extension, we have briefly discussed an 8000 node simulation of SSR.

Currently, we are applying SSR's basic principle, namely the combination of virtual and physical structures to other problems in the design of self-organizing systems. These problems include various aspects of distributed systems such as distributed scheduling and replica management. We hope to be able to report on first results soon.

References

1. Curt Cramer and Thomas Fuhrmann. Self-Stabilizing Ring Networks on Connected Graphs. Technical Report 2005-05, University of Karlsruhe (TH), Fakultaet fuer Informatik, March 2005.
2. Shu Du, Ahamed Khan, Santashil PalChaudhuri, Ansley Post, Amit Kumar Saha, Peter Druschel, David B. Johnson, and Rudolf Riedi. Self-Organizing Hierarchical Routing for Scalable Ad Hoc Networking. Technical Report TR04-433, Department of Computer Science, Rice University, Houston, TX, USA, 2004.
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. Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng Tien Ee, David Culler, Scott Shenker, and Ion Stoica. Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets. In *Proceedings of 2nd Symposium on Networked Systems Design and Implementation*, Boston, MA, U.S., May 2005.
5. Bryan Ford. Unmanaged Internet Protocol. *ACM SIGCOMM Computer Communications Review*, 34(1):93–98, January 2004.
6. Thomas Fuhrmann. Scalable routing for networked sensors and actuators. In *Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, September 2005.
7. Thomas Fuhrmann, Kendy Kutzner, Pengfei Di, and Curt Cramer. Scalable Routing for Hybrid MANETs. submitted.
8. K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The Impact of DHT Routing Geometry on Resilience and Proximity. In *Proceedings of the SIGCOMM 2003 conference*, pages 381–394. ACM Press, 2003.
9. Y. Charlie Hu, Saumitra M. Das, and Himabindu Pucha. 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*, Lihue, Kauai, Hawaii, May 2003.
10. David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, 353:153–181, February 1996.
11. Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000)*, pages 243–254, Boston, MA, August 2000.

12. Guangyu Pei, Mario Gerla, and Xiaoyan Hong. Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of IEEE/ACM MobiHOC 2000*, pages 11–18, Boston, MA, U.S., August 2000.
13. 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, New Orleans, LA, USA, February 1999.
14. Himabindu Pucha, Sumitra M. Das, and Y. Charlie Hu. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. In *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*, English Lake District, UK, December 2004.
15. 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.
16. Aline Carneiro Viana, Marcelo Dias de Amorim, and Serge Fdida. An Underlay Strategy for Indirect Routing. *Wireless Networks*, 10:747–758, 2004.
17. Thomas Zahn and Jochen Schiller. MADPastry: A DHT Substrate for Practicably Sized MANETs. In *5th Workshop on Applications and Services in Wireless Networks (ASWN 2005)*, Paris, France, June 2005.