Using Linearization for Global Consistency in SSR

Kendy Kutzner[1] and Thomas Fuhrmann[2]

[1]University of Karlsruhe
Computer Science Department
Am Fasanengarten 5
76131 Karlsruhe, Germany
kutzner@ira.uka.de

[2]Technical University of Munich
Computer Science Department
Boltzmannstrasse 3
85748 Garching, Germany
fuhrmann@net.in.tum.de

Abstract

Novel routing algorithms such as scalable source routing *(SSR) and* virtual ring routing *(VRR) need to set up and maintain a virtual ring structure among all the nodes in the network. The* iterative successor pointer rewiring protocol *(ISPRP) is one way to bootstrap such a network. Like its VRR-analogon, ISPRP requires one of the nodes to flood the network to guarantee consistency.*

Recent results on self-stabilizing algorithms now suggest a new approach to bootstrap the virtual rings of SSR and VRR. This so-called linearization method does not require any flooding at all. Moreover, it has been shown that linearization with shortcut neighbors has on average polylogarithmic convergence time, only.

1 Introduction – Scalable Source Routing

The scalable source routing protocol (SSR) [4][6] is a novel routing protocol. It draws on ideas from overlay routing protocols such as Chord. Like Chord, SSR views all nodes of the network as being members of a virtual ring. The nodes' addresses determine their position on the ring. In other words, the ring is the *circularly connected address space*. This ring is purely virtual, i. e. it is completely independent from the actual physical topology of the network. SSR does *not* assume the nodes' addresses to match the actual network topology. Nodes have virtual neighbors in the virtual ring and physical neighbors in the physical network topology. In general virtual and physical neighbors of a node are independent.

Unlike Chord, SSR is a network layer routing protocol. Physically neighboring nodes are directly linked by some communication technology. For example, in the wireless case, nodes are physical neighbors when they are in reach of each other's radio links. Virtual neighbors are connected by source routes which act as virtual links. SSR builds up and maintains these source routes as part of its routing protocol (cf. sec. 3). In essence, nodes exchange messages containing source routes to other nodes. They store (some of) these source routes and may append (parts of) them to each other to create new source routes. Thereby, the nodes change the virtual links in the virtual network graph. SSR does this so that these virtual links, i. e. the sources routes, eventually connect all nodes to form the virtual ring. The details of the formation and maintenance of the virtual ring are discussed in [4].

When the virtual ring has been established, SSR can route messages to any destination. By construction the route cache of each node contains source routes to the node's neighbors in the virtual ring. Beside that the caches will contain source routes to other destinations. For example, all nodes that are part of a source route in the cache can be viewed as potential destinations, too. When routing a packet, the respective node chooses that (intermediate) destination from its cache that is physically closest to itself and virtually closest to the final destination of the packet. It appends the according source route from its cache to the packet's header. The nodes along this source route can then forward the packet using the source route in the packet. This routing step is repeated at the intermediate destination and all subsequent destinations until the packet has reached its final destination. If the virtual ring has been formed consistently, this routing algorithm is guaranteed to succeed for any source and destination pair.

Obviously, SSR is based on ideas from structured routing overlays such as Chord [9]. It combines these ideas with proposals to use source routes as edges in the overlay graph [8][10]. Recently, a proposal similar to SSR was put forth, *virtual ring routing* (VRR) [2]. Contrary to SSR, VRR does not use source routes and route caches, but builds up routing state along the paths that reflect the edges of the virtual

1-4244-0910-1/07/$20.00 ©2007 IEEE

network in the physical network.

Both SSR and VRR require a globally consistent virtual ring to guarantee correct routing. As we will discuss in section 3, this was thought to require at least one node to flood the network. Furthermore, no formal assessment about the convergence time of SSR's and VRR's consistency mechanisms has been done so far. In this paper we transfer recent results from the study of self-stabilizing algorithms to the convergence of SSR (and VRR). We describe a new ring formation algorithm that does not require flooding to guarantee consistency. Furthermore, using a result from Onus et al [1] we know this mechanism to have on average polylogarithmic convergence time for random graphs.

The rest of the paper is structure as follows: Section 2 summarizes the linearization technique from Onus et al. for arbitrary graphs. Section 3 describes how this linearization technique can be applied to the ring formation of SSR (and VRR). Section 4 details this new approach. Finally, section 5 concludes with an outlook to future work.

2 Linearization

Graph linearization is the task to link the nodes of an arbitrary graph in the order of their identifiers. Such an algorithm is called self-stabilizing if it converges to the correct state for every possible input graph.

As Onus et al. [1] showed, it is possible to linearize any given connected graph having unique node identifiers with the following simple decentralized and self-stabilizing algorithm:

Algorithm 1 (Pure Linearization) *For each node v we consider all its n neighbors u_i in the graph. These u_i can be sorted according to their node identifiers. Let $u_1 < \ldots < u_k < v < u_{k+1} < \ldots < u_n$ be the respective ordering. Then we replace the edges $\{v, u_1\}$, $\{v, u_2\}$, ..., $\{v, u_n\}$ with the new edges $\{u_1, u_2\}$, $\{u_2, u_3\}$, ..., $\{u_k, v\}$, $\{v, u_{k+1}\}$, $\{u_{k+1}, u_{k+2}\}$, ..., $\{u_{n-1}, u_n\}$. By applying this operation repeatedly the graph is transformed into a sorted linear list.*

Since *pure linearization* may require many iterations for some graphs, it is unsuitable for real world networks. Hence, Onus et al. propose *linearization with memory*: Unlike with pure linearization, edges are not replaced but added. This improves the performance of the algorithm significantly: The average runtime decreases from linear to polylogarithmic.

However, keeping all edges may require significant memory at the nodes. Therefore, Onus et al. propose *linearization with shortcut neighbors* (LSN). The idea is similar to the techniques that are applied in [9] and [7]: Every node divides its local view of the identifier space into exponentially growing intervals. For every interval at most one edge is remembered.

As with pure linearization, LSN preserves the connectedness of the input graph. Moreover, Onus et al. show that this algorithm converges quickly for regular random graphs as well as for power law graphs (e. g. a power law graph with 16.000 nodes and $\alpha = 2$ converges in less than 39 rounds). Hence, LSN is suitable for real world applications. In the following section we describe how to use the linearization technique to improve SSR and similar protocols.

3 SSR and Linearization

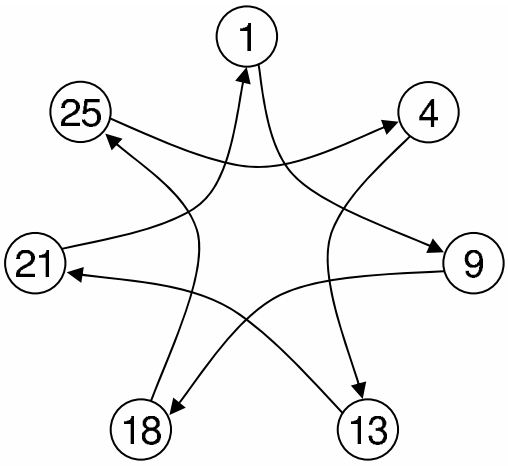
The authors of SSR originally proposed ISPRP, the *iterative successor pointer rewiring protocol*, to ensure consistency of SSR's virtual ring. ISPRP achieves local consistency of the ring by means of an iterative protocol that ensures that each node has exactly one successor and exactly one predecessor. To this end each node sends a notification message to its presumed successor. If a node, say node A, detects a *local inconsistency*, i. e. more than one node notified it to be its successor, A sends update messages to these nodes to ensure a partial ordering among these nodes. Say nodes B and C both notified A to be their successor, then either $B < C < A$ or $C < B < A$. In the former case A sends an update to B pointing it to C, and vice versa in the latter case.

As described in section 1 these messages contain source routes. In our example an update that is sent from A to B pointing it to C as better successor would carry a source route $A \rightarrow C$ that B would append to its source route $B \rightarrow A$. Thereby B obtains a source route $B \rightarrow C$.

This process continues until all nodes have pointers to their successors in a locally consistent way. ISPRP achieves *global consistency* by having one node flood the network with its identifier. Thereby a potentially remaining global inconsistency can be detected and resolved iteratively using ISPRP's normal rewiring process [3]. VRR employs a similar technique by piggy-backing the address of one node – the so called 'representative' – onto the hello beacons. Both, SSR and VRR propose to choose the node with the numerically largest address as (one) representative.

Linearization avoids this flooding step: It considers the address space as linear, not as being a ring. Thereby the address space regains its natural global ordering. Note that each iteration of the linearization process preserves the connectedness of the network. Therefore the network will not be partitioned if it was connected at the beginning. As a consequence, local consistency is equivalent to global consistency.

This is illustrated in figure 1. With respect to the successor relation of ISPRP, all nodes are locally consistent, i. e. all nodes have exactly one successor and exactly one predecessor. This is easily seen when we draw the graph as



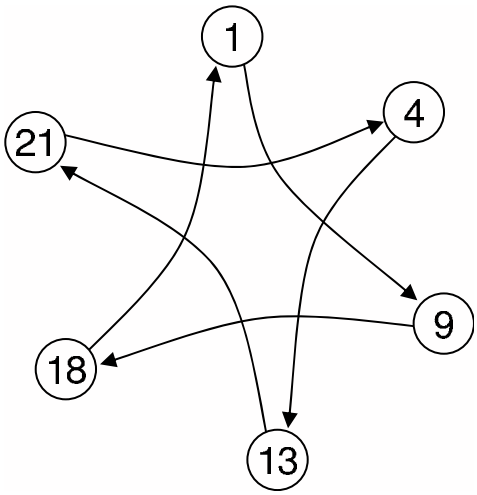


Figure 2. Separate Rings

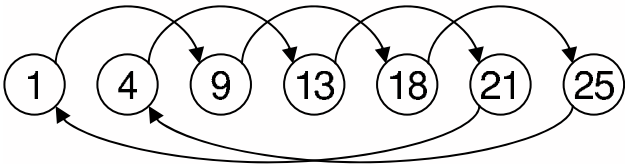


Figure 1. The Loopy State

ring (upper figure). Obviously, despite being locally consistent, the graph is not globally consistent. If we draw it linearly (lower figure), this global inconsistency is also reflected locally: Nodes 1 and 4 have two right neighbors each; nodes 21 and 25 have two left neighbors each. If all nodes – except for the leftmost and rightmost node – had at exactly one left and right neighbor, the graph would be globally consistent.

The type of inconsistency discussed so far is called *loopy state*. Another type of global inconsistency are several partitioned virtual rings. Figure 2 shows a corresponding example. Here nodes 1, 9, 18 and 4, 13, 21 form two disconnected rings. A consistency mechanism must hence not only avoid loopy states. It must also guarantee that the resulting virtual graph is one ring.

In the next section we sketch an iterative consistency mechanism for SSR that manages both: It removes local inconsistencies and preserves the connectedness of the graph.

4 Using Linearization in SSR

The insight from the previous section suggests a new algorithm for ring formation in SSR (and VRR). We describe it formally as self-stabilizing algorithm:

Let V be the set of nodes in the network. Let E_p be the set of edges in the physical network graph. Let E_v be the set of edges in the virtual network graph that is to be turned into the virtual ring. Edges in E_p are physical communication links. Edges in E_v are source routes.[1]

Unlike with ISPRP the edges in E_v are undirected, i. e. $\overline{v_1 v_2} \in E_v \Leftrightarrow \overline{v_2 v_1} \in E_v$, because linearization uses the total ordering of the nodes' addresses to distinguish left and right neighbors. Thus, $\forall v_{1,2} \in V, v_1 \neq v_2$ either $v_1 < v_2$ or $v_2 < v_1$. In contrast, ISPRP uses the directedness of the edges in the virtual graph to define its successor relation.

We define $N_L(v) := \{v' \in V : \overline{vv'} \in E_v \wedge v' < v\}$, the *left neighbor set* of v, and correspondingly N_R, the *right neighbor set* of v.

Upon initialization, the set of virtual edges is initialized to contain the physical edges, i. e. $E_v := E_p$. Then, each node linearizes its virtual neighbors. We explain this linearization procedure for the right neighbor set. It can be applied analogously to the left neighbor set. Assume $v_{2,3}$ are right neighbors of v_1 with $v_2 < v_3$ and $\neg\exists v' \in N_R(v_1) : v' < v_{2,3}$.

Then v_1 sends both v_2 and v_3 a neighbor notification message containing a pointer to v_3 and v_2 respectively. For SSR these pointers are source routes. For VRR the notification messages set up state along their forwarding path. In terms of the formal description here, this means that the edge $\overline{v_2 v_3}$ is entered into E_v (if it has not already contained that edge).

An SSR node v_2 that receives a neighbor notification message pointing it to v_3 enters the according source route into its route cache. Then v_2 acknowledges the message

[1] Note that the proposed mechanism also applies to other routing mechanisms such as Virtual Ring Routing. There the virtual edges are the paths as represented by the routing table entries.

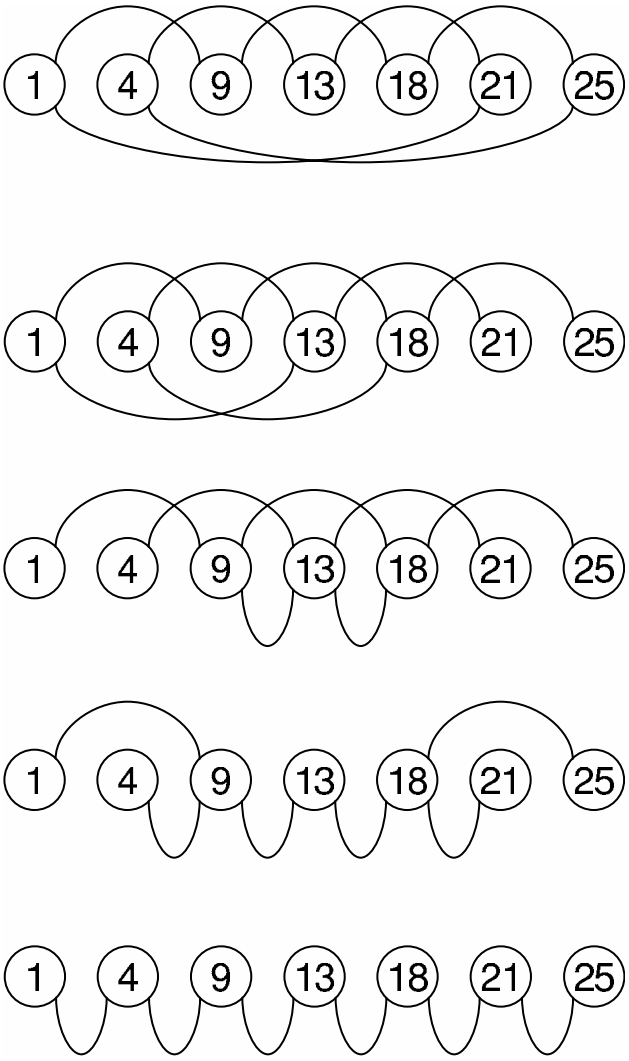


Figure 3. The Linearization Algorithm at work

to indicate that its side of the new edge has been successfully established. When v_1 has received the acknowledgment both from v_2 and v_3 it may remove the state for its edge to v_3. If it does so, it should send a tear down acknowledgment to v_3 so that v_3 can remove the according state too. In terms of the formal description here, this removes $\overline{v_1 v_3}$ from E_v.

As a result, this procedure reduces the size of v_1's right neighbor set by one. Moreover, any node v can apply it as long as $|N_R(v)| > 1$. The same holds for the nodes' left neighbor set. Thus, linearization will transform the virtual network graph into the linear graph, in which each node has at most one left and one right neighbor. Furthermore, each linearization step maintains the connectedness of the graph. Assuming trivially that the physical network graph is connected, we have thus proven that linearization builds a globally consistent connected linear virtual graph. Figure 3 illustrates the algorithm with the example from section 2.

In order to complete the virtual ring, the leftmost node must establish an edge to the rightmost node. To this end, a node with an empty left neighbor set sends a clockwise discovery message. Similarly, a node with an empty right neighbor set could send a counter-clockwise discovery message. It should do so for sake of redundancy. These messages are routed through the virtual ring using the usual greedy routing rules of SSR (or VRR respectively) until they reach a node with an empty right (or left) neighbor set. They are then acknowledged, thereby concluding the setup of the virtual ring.

As described in section 1, SSR nodes cache source routes. Each of these routes contains many nodes that can serve as intermediate destinations. As has been demonstrated in [5], a node typically caches at least one node for each of the exponentially growing intervals (shortcut node) introduced with LSN (cf. sec. 2). Thus, from the results in [1] we know that this algorithm has polylogarithmic convergence time if the nodes in this shortcut set are informed with a message containing paths to the members of the current set.

The SSR protocol as presented in [6] already guaranteed the network to converge into a consistent and connected state. However, it was necessary to flood the network at least once to guarantee these properties. With the modifications presented in this section, the protocol is now guaranteed to converge and to keep the network connected without flooding any message.

5 Conclusion and Future Work

Scalable source routing (SSR) and virtual ring routing (VRR) are novel routing algorithms that are based on ideas from structured overlay networks such as Chord. They set up and maintain a virtual ring in which the nodes are ordered according to their globally unique identifier. SSR suggested to use ISPRP to bootstrap the virtual ring. VRR provides a similar functionality in its protocol. Both require at least one node to flood the network in order to guarantee global consistency of the virtual ring. Albeit SSR and VRR have shown good performance in simulations, no formal assessment of their convergence time has been published so far.

In this paper, we applied a recent idea from self-stabilizing algorithms to SSR. This so-called *linearization* method guarantees globally consistent convergence of the virtual ring without the need for flooding. Moreover, it is known to have polylogarithmic convergence time, only.

Still, this fruitful collaboration between the more theoretical side of the research on self-stabilizing algorithms and their application to real-world challenges of routing are

at their beginning. As a next step we are about to evaluate the performance of linearization in simulations of SSR and VRR. But more thorough theoretical work needs to be done, e. g. to obtain more precise bounds on the performance of SSR and VRR, especially with respect to convergence, stability and router state.

Acknowledgments

This research was funded by Deutsche Forschungsgemeinschaft under grant number FU-448/1. The authors would like to thank Christian Scheideler for the fruitful discussions that lead to this paper.

References

[1] Melih Onus Andrea Richa Christian Scheideler. Linearization: Locally self-stabilizing sorting in graphs. In *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments*, 2007.

[2] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron. Virtual Ring Routing: Network Routing Inspired by DHTs. In *Proc. ACM SIGCOMM '06*, Pisa, Italy, September 2006.

[3] Curt Cramer and Thomas Fuhrmann. Self-Stabilizing Ring Networks on Connected Graphs. Technical Report 2005-5, Fakultät für Informatik, Unversität Karlsruhe (TH), Germany, 2005.

[4] 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*, pages 240–251, September 2005.

[5] Thomas Fuhrmann. A self-organizing routing scheme for random networks. In *Proceedings of the 4th IFIP-TC6 Networking Conference*, pages 1366–1370, Waterloo, Canada, May 2–6 2005.

[6] Thomas Fuhrmann, Pengfei Di, Kendy Kutzner, and Curt Cramer. Pushing chord into the underlay: Scalable routing for hybrid manets. Interner Bericht 2006-12, Fakultät für Informatik, Universität Karlsruhe, June 21 2006.

[7] Kendy Kutzner, Curt Cramer, and Thomas Fuhrmann. Towards Autonomic Networking using Overlay Routing Techniques. In *Proceedings of the 18th International Conference on Architecture of Computing Systems (ARCS '05) - System Aspects in Organic and Pervasive Computing*, Innsbruck, Austria, March 2005.

[8] 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.

[9] 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.

[10] 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.