Self-Stabilizing Ring Networks on Connected Graphs

Curt Cramer, Thomas Fuhrmann
System Architecture Group
University of Karlsruhe (TH)
76128 Karlsruhe, Germany

Phone: +49 (7 21) 6 08 – 67 29
Fax: +49 (7 21) 6 08 – 76 64
e-mail: cramer@ira.uka.de

January 31, 2005

Fakultät für Informatik

Technical Report 2005-5

Abstract

Large networks require scalable routing. Traditionally, protocol overhead is reduced by introducing a hierarchy. This requires aggregation of nearby nodes under a common address prefix. In fixed networks, this is achieved administratively, whereas in wireless ad-hoc networks, dynamic assignments of nodes to aggregation units are required.

As a result of the nodes commonly being assigned a random network address, the majority of proposed ad-hoc routing protocols discovers routes between end nodes by flooding, thus limiting the network size. Peer-to-peer (P2P) overlay networks offer scalable routing solutions by employing virtualized address spaces, yet assume an underlying routing protocol for end-to-end connectivity.

We investigate a cross-layer approach to P2P routing, where the virtual address space is implemented with a network-layer routing protocol by itself. The *Iterative Successor Pointer Rewiring Protocol (ISPRP)* efficiently initializes a ring-structured network among nodes having but link-layer connectivity. It is fully self-organizing and issues only a small per-node amount of messages by keeping interactions between nodes as local as possible.

The main contribution of this paper is a proof that ISPRP is self-stabilizing, that is, starting from an arbitrary initial state, the protocol lets the network converge into a correct state within a bounded amount of time.

Keywords peer-to-peer, routing, sensor networks, self-stabilization, correctness

1 Introduction

Autonomous systems like ad hoc and sensor networks require protocols which make the system self-organizing. A routing protocol, for example, should be able to cope with self-assigned random node addresses and to scale reasonably with the network size at the same time. Routing scalability traditionally is achieved by introducing a hierarchy in the address structure. The hierarchy is dependent on the position of the nodes in the network graph. In autonomous networks, the hierarchy thus has to be established dynamically, for example using clustering protocols. Still, as nodes commonly have self-assigned random network address, the majority of proposed ad-hoc routing protocols (see for example [Per00]) performs route discovery between end nodes using some variant of flooding. This approach is limited in its scalability.

Although Peer-to-Peer (P2P) systems are mainly known from the various file-sharing applications, their main benefit is that they offer scalable routing in a virtualized address space [CF05b]. Structured overlays like Chord [SMK+01], Pastry [RD01], and others are well-known examples of P2P systems. Here, nodes also have self-assigned random addresses, however, overlay addresses. These systems assume the prior existence of a network-layer routing protocol.

We investigate cross-layer solutions where scalable routing in a virtual address space is implemented as network-layer protocol by itself. The *Iterative Successor Pointer Rewiring Protocol (ISPRP)* [CF05a] is our first step towards this. It is able to build a basic ring network on top of a network in which nodes only have link-layer connectivity. Once this ring has been established, messages can be routed between arbitrary nodes. The key concern with ISPRP was to limit the amount of messages required for the initalization as opposed to previously proposed flooding-based approaches [HDP03, PDH04]. This is done by locally restricting interactions and message exchanges to places where inconsistencies occur. Thus, ISPRP produces only very low overhead.

In this paper, we present a correctness proof for ISPRP. Moreover, we show that ISPRP is self-stabilizing [Dol00] in the sense that the protocol lets a (connected) network converge into a globally correct state starting from an arbitrary state within a bounded amount of time. Thus, ISPRP is able to initialize a routeable network among freshly deployed nodes like it is required in a sensor network.

This paper is structured as follows: We discuss related work in the next section. A detailed description of the problem is given in section 3, and ISPRP is specified in section 4. The main section is number 5, in which a proof for ISPRP's correctness is given. There, we also show that ISPRP is self-stabilizing. We conclude our findings and give an outlook on future work in the last section.

2 Related Work

Other authors have also considered building a network-layer routing protocol using P2P techniques [EFK01, For04]. However, their design is not yet complete and therefore, no experimental performance evaluation is available. In contrast to our ring-based system, their system uses hierarchical, tree-based routing. They do not consider small nodes with limited capabilites like in sensor networks, but aim at the Internet.

Hu et al. propose to use the Pastry overlay for unicast routing in ad-hoc networks, where routes between virtual addresses are discovered using the DSR routing protocol [HDP03, PDH04]. Their work differs from ours in that we do not require all nodes to flood the network in order to discover routes.

3 Problem Description

Given is a "bare" (that is, freshly deployed) network with only link-layer connectivity available. There is no routing protocol available to send messages from a node to any other node. The network can be modeled as a graph $G = (N, E)$ consisting of the nodes N and the edges $E \subseteq N \times N$. Initially, each node only has knowledge (that is, the MAC addresses) of all nodes in its direct (one-hop) vicinity, either by means of the MAC protocol itself, or by each node periodically broadcasting "hello" beacons. We assume that the graph is connected, meaning that there exists a path from $i \in N$ to any $j \in N$. This is no limitation, as in partitioned graphs, each partition can be handled as a standalone, connected graph.

The problem to be solved by ISPRP is to build the seed for a network layer on top of the given graph. We term this problem the *initialization* of the network. As in *ad-hoc* [Per00] and *sensor networks*, and unlike the Internet, no distinction is made between end nodes and intermediate nodes, that is, all nodes are peers being equal in their functionality. However, we do not consider node mobility for the time being. In our first design of ISPRP, we also assumed the network to be static and that there is no message loss.

Nodes choose their network-layer addresses by themselves, for example by randomly selecting them, from a large linear integer space $A = [0, 1, 2, \ldots, s-1]$, that wraps around at the borders. In structured P2P overlays, s typically is a power of a base b. By making the address space large enough, address collisions become very unlikely. We assume addresses to be unique in the following. This lets us identify nodes by their addresses, and $N \subseteq A$. Nodes are arranged in A according to a total ordering $<$. The node j which has the next greater address in increasing direction after i is called i's *(correct) successor*, where an address space wrap-around can occur. We denote this by $j = succ(i)$. The ordering $\prec$ is defined to allow for the address space wrap-around based on the total ordering $<$ of integers. Note that in the circular address space, $i \prec j$ *and* $j \prec i$ for all i, j. But we need $\prec$ to compare sequences of more than two nodes arranged on a ring, for example $b \prec a \prec c$, where we mean that b does not lie between a and c, and c does not lie between b and a on the circle in increasing direction.

As in other P2P routing networks using a circular address space like Chord [SMK+01], the basic network structure needed to consistently route messages from one node to another is *a ring*. In this ring, each node maintains a virtual link or pointer to its successor. The virtual links are composed of multiple physical links combined at the link layer. It is insufficient for the nodes to only know the virtual address of their respective successor. A *path* to this successor has to be known, too. Usually, the network below the overlay establishes this path. There, the routers keep enough information to determine the next message hop along a route to any given destination. E.g., Chord requires a working routing protocol as a prerequisite.

In our cross-layer design and the P2P scenario, however, there are no dedicated routers, and the nodes themselves take responsibility for routing messages. With randomly assigned addresses, storing the above-mentioned information leads to huge memory requirements and protocol overhead, as next-hop information cannot be hierarchically aggregated, which is a common practice in fixed networks like the Internet. To avoid this problem, another option is to use *source routing*. There, message headers contain an enumeration of all the nodes along the path. Hence, forwarding nodes do not have to maintain any routing information, but message sources need to know the routes to all destinations they communicate with.

If not efficiently, routing yet works consistently in the basic case of the nodes being arranged in the ring structure. It has already been noted elsewhere [SMK+01] that with a ring and the addition of $O(\log n)$ short-cut links, lookup paths with lengths of $O(\log n)$ overlay hops on average can be achieved, where $n = |N|$.

The task of ISPRP is to create the ring structure among the nodes, overlaid onto the "bare" physical network, by establishing source routes from each node to its corresponding successor node. This has to be done efficiently, that is, by exchanging as few messages as possible in

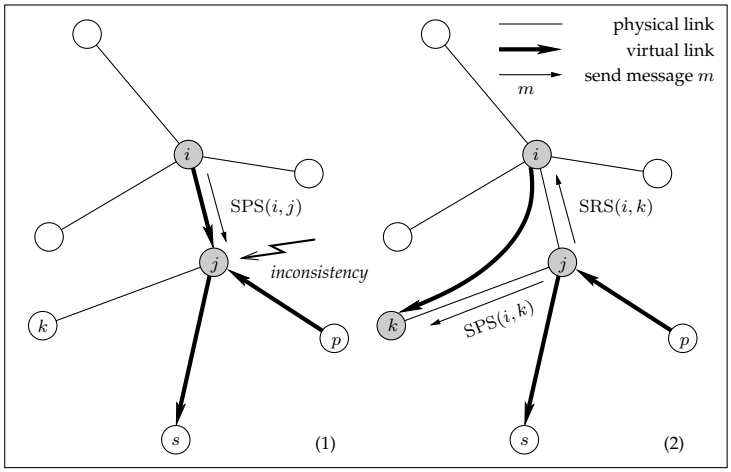

Figure 1: One iteration in applying the ISPRP

order to account for the nodes' limited (energy, processing, and memory) resources.

The desired ring can also be constructed by each node flooding its address in the network. This, however, yields a message overhead of $O(n^2)$, and requires the nodes to store $O(n)$ state information for forwarding messages to arbitrary nodes. Simulation results presented elsewhere [CF05a] show that ISPRP achieves the same goal with a significantly smaller message overhead.

4 The Iterative Successor Pointer Rewiring Protocol (ISPRP)

The ISPRP was designed with two key aspects in mind: simplicity and efficiency. By relying on the total ordering of the linear integer address space, inconsistencies can be locally detected and interactions are also locally performed, effectively reducing the control message overhead.

(The description of the protocol and the notations used here are accompanied by an example which is depicted in fig. 1.) Each node i stores a limited amount of information about the network: The addresses of its one-hop neighbors, the source route to the node j which – from i's current local view – correctly registered i as its successor, and the source route to its own current successor.

A node i initializes its successor pointer by selecting the node with the smallest address among all nodes with addresses larger than its own from its one-hop neighbourhood, taking into account the address space wrap-around. The tentative successor node j is informed about i's choice by sending it a *Successor Pointer Solicitation* message $SPS(i, j)$.

Upon reception of an $SPS(i, j)$ message, using its local data and the data from the SPS message, j checks for a local inconsistency. If no inconsistency is detected, j registers i as its current predecessor. In case of an inconsistency, there are two possible causes: (1) j has chosen a wrong successor, or (2) more than one node is pointing at i, that is, there already is a node pointing at it (this node is called its *predecessor*). In the first case, which can occur if j finds a better successor in the source route contained in the header of the message, j adjusts its pointer and subsequently issues an SPS message.

In the second case (that is, there already was a node p pointing at j), an inconsistency is detected by j (see fig. 1, step (1)). As a consequence of the total ordering property, no two nodes can have the same successor. Therefore, j checks which of the nodes pointing at it (p or i) does so incorrectly. In the example, we assume i to be this node. Node j then selects the node k which – in j's local view – is i's correct successor instead of j itself from the set of nodes whose addresses are locally cached. Node j informs i about that fact by sending it back

a *Successor Rewiring Solicitation* message $SRS(i,k)$. In order to reduce the number of control messages, j directly sends the $SPS(i,k)$ message required in the next step to k itself (fig. 1, step (2)). As i receives the $SRS(i,k)$ message from j, it may adjust its successor pointer to point at k. The reception of either a SPS or SRS triggers the execution of the steps for detecing local inconsistencies at j as described above. Inconsistencies are resolved by iteratively "moving" a node's successor pointer towards the correct one. Hence, the *Iterative Successor Pointer Rewiring Protocol, ISPRP*.

To enable routing between nodes and their successors, source routes have to be recorded in the headers of messages. As a result of ISPRP's local interactions and its iterative nature, path lengths will be sub-optimal. To reduce path lengths and eliminate cycles, nodes can shorten their successor paths using Dijkstra's algorithm.

5 Proof of Correctness

In this section, a proof of ISPRP's correctness and its self-stabilizing property is given.

Definition 1 (Network state). Network state is modeled by the function $NS : N \rightarrow N$ which maps a node i to the node it has currently selected as its successor.

In the case of $|N| > 1, \forall i \in N : NS(i) \neq i$, as each node has at least one-hop neighbor which by definition $\neq i$.

The notion of ISPRP's "correctness" has to be detailed first: As ISPRP can only detect local inconsistencies, it is not ensured by ISPRP alone that the global state of the system will be a ring. We can distinguish between *local correctness* and *global correctness*.

Definition 2 (Local correctness). The network is in a locally correct state NS if, and only if $\forall i \in N \exists_1 j \in N : NS(j) = i, \forall i \in N : NS(i)$ is defined, and $NS(i)$ is the node with smallest address known to i succeeding it in the address space.

Local correctness implies that using ISPRP, no node can locally detect an inconsistency leading to corrections. In the following, we call the change of a node's successor pointer by the protocol actions resulting from the detection of a local inconsistency a *protocol interaction*. By "termination of ISPRP's execution", we mean that no more protocol interaction is made by any node once a certain network state has been reached.

Definition 3 (Global correctness). The network is in a globally correct state NS if, and only if $\forall i \in N : NS(i) = succ(i)$.

Note that global correctness also implies local correctness. However, a locally correct network does not have to be globally correct. E.g., as a result of wrapping A around at its borders, nodes could organize themselves into two intertwined rings, for example $0 \rightarrow 2 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 0 \rightarrow \ldots$. The wrap-around could also lead to the formation of two independent rings. It is shown in section 5.1 that with ISPRP, the network will always converge into a locally correct state within bounded time.

We have devised a *flooding repair mechanism*, which together with ISPRP transforms a locally correct network into a globally correct one. The proof of this claim is given in section 5.2. The repair mechanism works as follows:

Definition 4 (Flooding repair mechanism). Each node i whose successor pointer crosses the address space border, this means $i < 0 \wedge NS(i) \geq 0$, floods G with a message containing its network-layer address.

Note that this significantly differs from each node having to flood the network. With our solution, the increase in the total per-node message amount equals the number of intertwined or independent rings formed, which usually is small.

In section 5.3, we show that ISPRP and the repair mechanism together make the network self-stabilizing. That is, starting from an arbitrary network state, like "from scratch" in an uninitialized network, the protocols ensure that a globally correct state is reached within a finite amount of time. For simplicity reasons, in all proofs, message delivery is assumed to be reliable, and thus, a message sent from i to j arrives there within a finite amount of time.

For practical reasons, we also have to prove two other features: First, ISPRP and the repair mechanism, respectively, can be run concurrently and do not require node synchronicity. Second, convergence to global correctness is not dependent on the particular sequence in which ISPRP and the repair mechanism are executed, as long as the repair mechanism is run after local correctness has been reached. This means that the repair mechanism can be periodically run by the affected nodes, as one would usually implement the protocol.

5.1 Local Correctness

First, we show that the execution of ISPRP yields a locally correct network state. More precisely, we claim that:

Theorem 1. *Starting with a network in an arbitrary state NS on an arbitrary, yet connected graph G, the execution of ISPRP terminates after a finite number of protocol interactions, and network state then is locally correct.*

Note that the requirement of a connected graph is an implicit one. It ensures that $i \in N$ is able to reach any $j \in N$ via some path in G. To prove the theorem, we employ the *variant function* proof technique [Dol00]. Network state is reflected by the value of the variant function $v(\cdot)$.

Definition 5 (Set of potential successors). We define the *set of potential successors* $PS(i)$ of a node i to be the (ordered) set of all nodes which succeed i in the address space starting with its correct successor, ending with the node $j = NS(i)$ which i currently selected as its successor. I.e., $PS(i) := \{succ(i), \ldots, j\}$ where $i \prec succ(i) \prec \ldots \prec j$.

Definition 6 (Variant function). Our variant function then is $v(G, NS) := \sum_{k \in N} |PS(k)|$.

As a node i knows its one-hop neighborhood, it can change $NS(i)$ to a more appropriate successor if necessary. If the network starts in an uninitialized state, i makes its first successor choice based on its one-hop neighborhood.

Proof of theorem 1. The proof of the theorem is divided in three parts:

1. We show that $v(G, NS)$ is strictly monotonically decreasing if a protocol interaction is made.
2. We show that $v(G, NS)$ is bounded from below.
3. We show that the termination of ISPRP's execution is equivalent to the network being in a locally correct state.

Proof of part 1. For the first part, we have to take a look at the sets $PS(\cdot)$. As $j = NS(k)$ is defined, $PS(k) = \{succ(k), \ldots, j\}$. By specification of the protocol, an interaction is made if $NS(k) = j$, and there is either another node i pointing at j at the same time, or j knows of another node i where $k \prec i \prec j$. In the first case, depending on whether $k \prec i \prec j$ or $i \prec k \prec j$, subsequently, k changes its successor pointer to i, or i changes its successor pointer to k. (Note that the second case is subsumed under the first case.) Let a be the node which changes its current successor from c to b, where $a \prec b \prec c$. Before the interaction, $PS_{pre}(a) = \{succ(a), \ldots, b, \ldots, c\}$, and after the interaction, $PS_{post}(a) = \{succ(a), \ldots, b\}$. It is implied that $|PS_{post}(a)| < |PS_{pre}(a)|$. In turn, $|v(G, NS_{post})| < |v(G, NS_{pre})|$, which means that $v(G, NS)$ is strictly monotonically decreasing if a protocol interaction is made. □

Proof of part 2. Second, the minimum size $|PS(k)|$ of every set of potential successors equals 1. That is, if k has chosen its correct successor $succ(k)$, then $PS(k) = \{succ(k)\}$. By definition, no other node $j \neq k$ with $j \neq succ(k)$, and $k \prec j \prec succ(k)$ exists, therefore, in this case, ISPRP does not change k's successor pointer any more. The argument holds $\forall k \in N$ and thus, $v(G, NS) = \sum_{k \in N} |PS(k)| \geq \sum_{k \in N} 1 = |N|$, which is the least lower bound. □

Taken together, part 1 and 2 imply the following: Starting from an arbitrary state, it is guaranteed that ISPRP's execution terminates after a finite number of protocol interactions.

Proof of part 3. However, for the complete proof of theorem 1, third and last, we still have to show "termination of ISPRP's execution" $\Leftrightarrow$ "network is in a locally correct state". Starting with the $\Rightarrow$ direction, termination of ISPRP's execution means that no more protocol interactions are made. This implies that each node k points at the closest successor $\neq k$ among its known nodes, and exactly one node is pointing at k (for $|N| > 1$). At most one node is pointing at k, as otherwise the execution would not have terminated. Assume that no node is pointing at k. Then, as each node j points at another node $\neq j$, there has to $\exists j \in N \setminus \{k\}$ which as pointed at by two different nodes. This is a contradicition, since execution would not have terminated in this case. Hence, at least one and at most one, implying exactly one node is pointing at k.

In the other direction, by definition, a locally correct state implies that each node k points at the closest successor among its known nodes, and exactly one node, which from k's view is its closest predecessor, points at k. Taken together, with ISPRP, no node detects an inconsistency, which means that no protocol interaction is made and hence, ISPRP's execution terminates. □

□

5.2 Global Correctness

Theorem 2. *Using ISPRP and the flooding repair mechanism, a locally correct network state is transformed into a globally correct one within a finite number of protocol interactions.*

Proof of theorem 2. We begin by examining the structure of a locally correct network in detail. Recall from definition 2 that local correctness is equivalent to $i \in N$ pointing at another node $j \in N$ ($j \neq i$) and being pointed at by exactly one node $k \in N$ ($k \neq i$).

Definition 7 (Correct and incorrect successor pointers). The pointer from i to j is called *correct* if $j = succ(i)$, otherwise *incorrect*.

Consider the ordering of all nodes in N along the address space by their unique addresses (as depicted in fig. 2). We now divide the nodes into subsets called *bands*.

Definition 8 (Bands of nodes). A locally correct network state is assumed to be given. Starting with the first node next to the address space border in increasing direction, follow the successor pointers as long as they are correct. After that, all nodes visited thus far, including the first and last ones, form the first band. The next band starts with the node being pointed at by the incorrect successor pointer, and so on. Stop when a pointer crosses the address space border, as a band is not allowed to cross the border.

Consider the bands and the nodes visited in this first iteration as being numbered, say with a "1". If there are any unnumbered nodes left, start with the first unnumbered node next to the address space border in increasing direction and repeat the procedure (assigning an incremented number each time the border is crossed) until no unnumbered node is left.

Definition 9 (Chains of bands). As a result of the numbering procedure leading to the division of nodes into bands (cf. definition 8), the network is furthermore structured into *chains of bands*. Since the network is in a locally correct state, each node points at another node, and is being

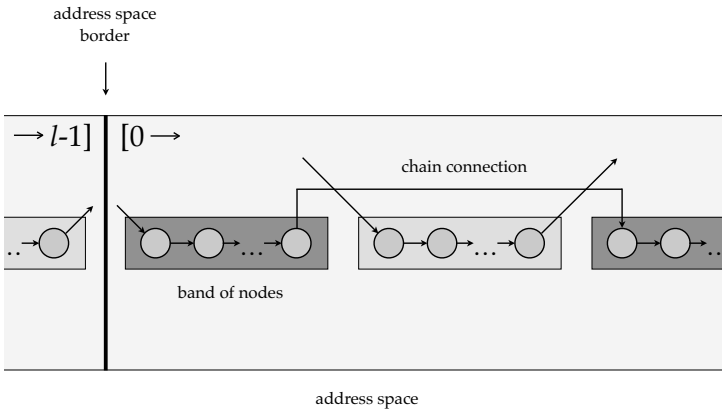

Figure 2: Bands of nodes

pointed at by exactly one node. Thus, this also applies to the bands: Each band points at another band, and exactly one band is pointing at it.

The numbering procedure ensures that, starting with the first band next to the address space border in increasing direction having a certain number, following the connected bands up to the address space border again, all bands have been assigned the same number. We call these connected bands *chains*. There are as many chains in the network as different numbers have been assigned in the numbering procedure. As a consequence of the network's local correctness, all nodes in a chain are sorted according to their unique addresses. By the *last node in a chain*, we denote the node next to the address space border in *decreasing* direction. The *first node in a chain* is the node next to the address space border in *increasing* direction.

Lemma 1. *In a locally correct network, both, the successor pointer ending at the first node in a chain, and the successor pointer originating at the last node in the same chain are crossing the address space border. All other pointers in the chain do not cross the border.*

Proof of lemma 1. The lemma is a direct result of definitions 8 and 9.

□

Definition 10 (Ordered tree). Given a directed tree, we call the tree *ordered* if an edge from node i to j implies $i < j$.

Lemma 2. *In a locally correct network, a chain minus the pointer to the first node and the pointer originating at the last node is an ordered tree.*

Proof of lemma 2. By definition of a chain, for each node i and its successor j in a chain, it holds that $i < j$. (Note that we can now use the unambiguous total ordering $<$, as we defined the chains not to wrap around the address space border.) The chain is weakly connected. From lemma 1, it follows that only the pointer to the first node, and the pointer originating at the last node cross the address space border. By removing these pointers, it is ensured that the chain is acyclic. Thus, a chain suffices the conditions for being an ordered tree with its last node being the root node. □

Now, by employing the flooding repair mechanism (cf. def. 4), each node having a successor pointer which crosses the address space border is required to flood its address in G. By lemma 1, these are the chains' last nodes. In fig. 3 (i), they are marked by an asterisk.

Lemma 3. *Protocol interactions resulting from the execution of ISPRP on an ordered tree do not deprive the tree of its ordered tree property, and the tree remains a single tree.*

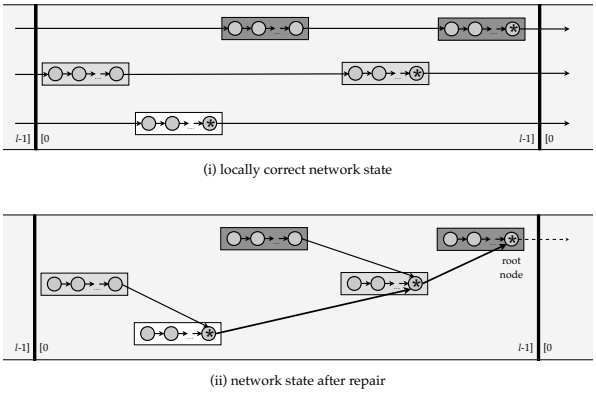

Figure 3: Locally correct network state and the flooding repair mechanism

Proof of lemma 3. Protocol interactions occur where local inconsistencies are detected in the tree. A local inconsistency is detected by k if more than one node is pointing at it. Let i and $j \neq i$ be two nodes pointing at k at the same time. Depending on whether $i < j$ or $j < i$, either i or j has to change its successor pointer, respectively. Let a be the node which changes its pointer to the other node, b (that is, $a < b < k$). Removing the pointer from a to k, we get two ordered trees, one rooted at r (which includes b as a child of k in the ordered sub-tree rooted at k), and one rooted at a. The two trees are merged by adding the pointer from a to b, which introduces no loop. Also, for the new edge from a to b, it holds that $a < b$. Thus, the graph resulting from the protocol interaction again is one ordered tree. □

Lemma 4. *After the flooding repair mechanism was executed by the last chain node with the largest address among all last chain nodes in an originally locally correct network, the network consists of exactly one ordered tree.*

Proof of lemma 4. Let L be the set of all last chain nodes in the original locally correct network. The nodes $l_k \in L$ can be ordered: $l_1 < l_2 < \ldots < l_{|L|}$. If node $l_k \in L$ floods the network with its address, all other nodes in N learn of its existence. Thus, all nodes in $P = \{l_i \in L | l_i < l_k\} = \{l_1, l_2, \ldots, l_{k-1}\}$ change their successor pointers to l_k, as l_k precedes their respective current successors, which by lemma 1 are behind the border. Additionally, by definition, for a node $e \notin P$ situated at the end of a band, it can also hold that $e < l_k < NS(e)$. Only these e and the last nodes $l_i \in P$ change their successor pointers.

From lemma 2, we know that a chain is an ordered tree rooted at its last node. If a node $e \notin P$ at the end of a band with $e < l_k < NS(e)$ changes its successor pointer to point at l_k, the ordered trees in the network are restructured and their number remains the same. The chain which contains e is being cut in two ordered trees by removing the incorrect pointer from e to $NS(e)$. Subsequently, by e changing its successor pointer to point at l_k, the ordered tree rooted at e is merged with the ordered tree containing the sub-tree rooted at l_k. As $e < l_k$, the result again is an ordered tree since no loop is introduced by adding the edge from e to l_k.

If $l_i \in P$ changes its successor pointer to end at l_k, the ordered tree rooted at l_k and the ordered tree rooted at l_i are merged into one which by definition again is ordered, as $l_i < l_k$. No loop is formed by the added edge. Thus, the ordered trees rooted at a node in $P = \{l_i \in L | l_i < l_k\} = \{l_1, l_2, \ldots, l_{k-1}\}$ are merged into one ordered tree, resulting in $|L| - (k-1) = |L| - k + 1$ ordered trees in the network in total.

By this merger, the last chain nodes which changed their successor pointers no longer are last chain nodes and do not flood the network anymore. However, as it still holds that $l_k < l_{k+1} < \ldots < l_{|L|}$, the ordered tree rooted at l_k is merged with the ordered tree rooted at l_i where $i > k$ once l_i floods.

Ultimately, if $l_{|L|}$ floods the network, all ordered trees in the network are merged into exactly one ordered tree (see fig. 3 (ii)). □

Lemma 5. *Using ISPRP and the flooding repair mechanism, a locally correct network is transformed into exactly one band of nodes within a finite number of protocol interactions.*

Proof of lemma 5. From lemma 4, it follows that the locally correct network is transformed into exactly one ordered tree within a finite number of protocol interactions, assuming that all last nodes in the chains flood the network within a finite amount of time. As a result of theorem 1, using ISPRP, the network is then made locally correct within a finite number of protocol interactions. The network then also still remains exactly one ordered tree, which follows from lemma 3. Consequently, the ordered tree is a band of nodes, that is, the successor pointers of all nodes in the network except for that originating at the single remaining last node are correct. Assume that the network does not consist of a single band. Then, there exists node i which points at $j \neq succ(i)$ and $i < succ(i) < \ldots < j$. As the tree is ordered, there cannot exist k with $i < k$ lying on the path from i to j, as otherwise i would have chosen k as its successor. Hence, there has to exist a sub-tree rooted at $l \neq i$ with $succ(i) < \ldots < l < j$, which is contradiction to the local correctness of the network. □

Lemma 6. *Once ISPRP's execution has terminated after the flooding repair mechanism was used by the affected nodes, the single remaining successor pointer crossing the address space border is correct.*

Proof of lemma 6. Let i be the node whose successor pointer crosses the address space border. Let j be the node which is pointed at by i. Assume that the pointer is incorrect. This implies that $j \neq succ(i)$. From lemma 5, it follows that after ISPRP's execution, the successor pointers of all nodes in $N \setminus \{i\}$ are correct. Thus, there exists a node $k \in N \setminus \{i\}$ which points at j, that is, $j = succ(k)$. Following from that, both nodes i and k point at j at the same time, which is contradiction to the condition that the execution of ISPRP has terminated before. Hence, i's successor pointer is correct. □

From lemmas 5 and 6, it follows that using ISPRP and the flooding repair mechanism, a locally correct network state is transformed into a globally correct one within a finite number of protocol interactions.

□

5.3 Self-Stabilizing Property

Assuming reliable message delivery and bounded delivery times, a protocol interaction has finished within a bounded amount of time. The following theorem thus is a direct result of theorems 1 and 2:

Theorem 3 (Self-stabilizing property). *ISPRP and the flooding repair mechanism together make the network self-stabilizing. That is, starting from an arbitrary network state on an arbitrary, yet connected graph, a globally correct network state is reached within a finite amount of time.*

As stated before, we also have to prove two more features in order to show that the protocols are implementable in a practical setting:

Lemma 7. *The execution of the flooding repair mechanism prior to the network's convergence into a locally correct state does not prolongate the termination of ISPRP's execution.*

Proof of lemma 7. Prolongation of the termination is equivalent to the value of the variant function being increased by the flooding repair mechanism ($v(G, NS_{post}) > v(G, NS_{pre})$). Assume l_k floods, then a node i changes its successor pointer only if l_k precedes i's current successor j in the address space ($i \prec l_k \prec j$). This implies $|PS_{post}(i)| < |PS_{pre}(i)|$. Otherwise, i does not change its successor pointer, which implies $PS_{post}(i) = PS_{pre}(i)$. Both cases taken together imply $v(G, NS_{pre}) \leq v(G, NS_{post})$. Hence, termination of ISPRP's execution is not prolongated. □

Theorem 4. *The flooding repair mechanism can be run concurrently, and does not require node synchronicity.*

Proof of theorem 4. As a result of lemma 4, the last nodes in the chains do not have to execute the repair mechanism synchronously. I.e., the mechanism does not require any means of detecting whether all nodes in the network have reached local correctness before it is run. Each of these nodes can execute it independent of the others. Once the last chain node with the largest address among all last chain nodes has flooded, global correctness will eventually be reached. □

Theorem 4 applies to ISPRP itself as well, which is an implicit result of the proof using a variant function. Thus, convergence to global correctness is not dependent on the particular sequence in which ISPRP and the repair mechanism are executed, as long as the repair mechanism is run after local correctness has been reached.

6 Conclusions and Future Work

In this paper, we have presented a correctness proof for the *Iterative Successor Pointer Rewiring Protocol (ISPRP)* [CF05a]. The protocol is our first step towards the investigation of building a network-layer P2P routing protocol on top of a network with only link-layer connectivity available. ISPRP's goal is to efficiently create a basic ring-structured P2P network among limited-capability nodes which assign addresses to themselves. Simulation results presented elsewhere show that ISPRP meets its goal and creates the P2P overlay with very low control message overhead.

The main result presented here is that ISPRP together with its accompanying repair mechanism is self-stabilizing in the sense that starting from an unitialized network, the network converges into a globally correct state within a bounded amount of time.

Thus far, we have only considered the establishment of a consistent P2P routing network in static networks after their initial deployment as a basis for the design and evaluation of ISPRP. For more practical scenarios, the protocol has to handle dynamics like node mobility. There, route breaks have to be detected and repaired, for example similiar to the way AODV [Per00] maintains them.

References

[CF05a] Curt Cramer and Thomas Fuhrmann. ISPRP: A Message-Efficient Protocol for Initializing Structured P2P Networks. In *Proceedings of the IEEE International Workshop on Strategies for Energy Efficiency in Ad Hoc and Sensor Networks 2005 (IWSEEASN '05)*, Phoenix, AZ, USA, April 2005. To appear.

[CF05b] Curt Cramer and Thomas Fuhrmann. On the Fundamental Communication Abstraction Supplied by P2P Overlay Networks. *European Transactions on Telecommunications (ETT)*, 16(1):1–9, 2005.

[Dol00] Shlomi Dolev. *Self-Stabilization*. The MIT Press, Cambridge, Massachusetts, USA, 2000.

[EFK01] 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.

[For04] Bryan Ford. Unmanaged Internet Protocol. *ACM SIGCOMM Computer Communications Review*, 34(1):93–98, January 2004.

[HDP03] Y. Charlie Hu, Sumitra M. Das, and Himabindu Pucha. Exploiting the Synergy between Peer-to-Peer and Mobile Ad Hoc Networks. In *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS IX)*, pages 37–42, 2003.

[PDH04] 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.

[Per00] Charles E. Perkins, editor. *Ad Hoc Networking*. Addison-Wesley/Pearson Education, Upper Saddle River, NJ, 2000.

[RD01] A. Rowstron and P. 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.

[SMK+01] 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 ACM SIGCOMM Conference 2001*, pages 149–160, San Diego, CA, USA, 2001. ACM Press.