ISPRP: A Message-Efficient Protocol for Initializing Structured P2P Networks

Curt Cramer, Thomas Fuhrmann
System Architecture Group
Universität Karlsruhe (TH)
76128 Karlsruhe, Germany
{curt.cramer|thomas.fuhrmann}@ira.uka.de

Abstract

Most research activities in the field of peer-to-peer (P2P) computing are concerned with routing in virtualized overlay networks. These overlays generally assume node connectivity to be provided by an underlying network-layer routing protocol. This duplication of functionality can give rise to severe inefficiencies. In contrast, we suggest a cross-layer approach where the P2P overlay network also provides the required network-layer routing functionality by itself. Especially in sensor networks, where special attention has to be paid to the nodes' limited capabilities, this can greatly help in reducing the message overhead.

In this paper, we present a key building block for such a protocol, the Iterative Successor Pointer Rewiring Protocol (ISPRP)*, which efficiently initializes a P2P routing network among a freshly deployed set of nodes having but link-layer connectivity. ISPRP works in a fully self-organizing way and issues only a small per-node amount of messages by keeping interactions between nodes as local as possible.*

1 Introduction

Peer-to-peer (P2P) applications have attained huge popularity in the last years. Although they were mainly used for sharing files over the Internet, other application areas are becoming more and more important, too. We believe that P2P protocols can serve as an important building block for various application scenarios in ad-hoc networks and sensor networks. There, they can be used to provide efficient, self-organizing, scalable, and reliable means for storing, aggregating, and retrieving data. Distributed hash tables (DHTs) like Chord [7], Pastry [6], and others especially lend themselves to these tasks.

From an architectural point of view, DHTs are overlay networks, i.e. they leverage a pre-existing communication infrastructure for establishing a virtualized communication structure among the participating nodes. As a result of the virtualization, in this structure, messages are routed in an application-specific way, independently of the routes chosen in the underlying network. The ratio of the path length (between two arbitrary nodes) as induced by the overlay routing algorithm to that of the shortest path in the underlying network is called *routing stretch*.

Naturally, overlay protocols also yield a control message overhead in establishing and sustaining their virtual structures. Research, thus far, has mostly been concerned with the long-run characteristics of these systems. E.g., it has been studied how to improve on routing stretch and how DHTs behave under frequent node arrivals and departures ("node churn"). In this paper, however, we focus on the issue of *initializing* a DHT-based overlay routing network among a set of nodes. Moreover, we assume *no underlying network protocol* to be available. (Other studies of P2P protocols often assume some underlying routing protocol like AODV whose overhead then is neglected.)

We believe that both aspects, the sudden and simultaneous deployment, as well as the lack of a prior routing infrastructure, reflect the specific requirements of sensor networks that distinguish them from P2P applications on the Internet. Especially, in this paper, we show that a cross-layer design of an initialization protocol for a DHT-based routing overlay, that meets all requirements of a network-layer routing protocol, is in fact feasible and can provide very good performance.

We envision an application scenario where some 1,000 limited-capability devices (e.g. small home appliances like power switches, lamps, motion detectors, etc.) autonomously form a P2P routing network on top of their physical topology. On top of that, higher-level protocols like service discovery can then be instantiated. The network shall not require any pre-defined structure, which simplifies network planning. Since it would be cumbersome to manually configure addresses and other parameters in an ever-growing network like this, the organization of the network thus should be fully autonomous. Moreover, since the devices might have reduced processing capabilities and memory capacities, the messages sent between them, and the state to be maintained by them should be kept at a minimum.

The *Iterative Successor Pointer Rewiring Protocol (ISPRP)*, which we present in this paper, is a first step towards

implementing a P2P routing overlay for scenarios like the one outlined above. It is able to build a basic ring overlay 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. Note that we do not cover optimizations after the initialization of the basic ring. These optimizations are feasible, yet outside the scope of this paper. The key concern with ISPRP was to limit the amount of messages required for the initalization. This is done by locally restricting interactions and message exchanges to places where inconsistencies occur, ISPRP produces only very low overhead.

This paper is structured as follows: The design goals of ISPRP and the construction of the basic solution, including the specification of ISPRP are presented in section 2. In section 3, results of a simulation study comparing ISPRP with a simple flooding protocol are discussed. They show the good viability of our protocol. Previous work which is related to ISPRP is reviewed in section 4. We conclude our findings with an outlook on future research directions in section 5.

2 Autonomous Initialization of Structured P2P Routing Networks

The basic problem to be solved by ISPRP is to build the seed for a network layer on top of a "bare" (i.e., freshly deployed) network with only link-layer connectivity available. We term this problem the *initialization* of the network, since no protocols or structures above the link layer are given prior to the execution of ISPRP. The remainder of this section describes the problem definition in more detail, and presents our solution approach.

2.1 Problem Definition

The network that has to be initialized is comprised of nodes among which only link-layer connectivity is assumed to be given (i.e., no routing protocol is available to send messages from a node i to an arbitrary node $j \neq i$). Every node has to have link-layer knowledge (i.e. 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 issuing hello beacons via radio broadcast.

As in *ad-hoc* [4] and *sensor networks*, and unlike the Internet, no distinction is made between end nodes and intermediate nodes, i.e. all nodes are peers being equal in their functionality. However, we do not consider node mobility for the time being. For the first design of ISPRP, we also assume the network to be static, i.e. no nodes join or leave during the initialization period.

ISPRP implements a P2P routing protocol at the network layer. Nodes choose their addresses by themselves, e.g. by randomly selecting them or by generating hash values of their MAC-layer addresses, or public cryptographic keys.

Addresses are elements of a large linear integer space, i.e. $[0, 1, 2, \ldots, s-1]$, that wraps around at the borders where s typically is a power of a base b. By making the address space large enough, address collisions become very unlikely. As in other P2P routing networks like Chord [7], the basic network structure needed to consistently route messages from one node to another is *a ring*. In this ring, the nodes are arranged or sorted according to their unique addresses. Each node maintains a virtual link or pointer to its *successor*, which is the node directly following it in the thus ordered node set, taking into account the wrap-around.

The virtual links are composed of multiple physical links combined at the link layer. It is insufficient for the nodes to only know the address of their respective successor. A *path* to this successor has to be available, 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.

In our cross-layer design and the sensor-network scenario, however, there are no dedicated routers, and the nodes themselves take responsibility for routing messages. Here, storing the above-mentioned information would lead to huge memory requirements, as addresses are randomly assigned and thus, next-hop information cannot be hierarchically aggregated, which is a common practice in fixed networks as 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 want to communicate with.

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, i.e. by exchanging as few messages as possible in order to account for the nodes' limited (engergy, processing, and memory) resources.

It has already been noted elsewhere [7] 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 is the number of nodes in the network (called "network size" in the following). If not efficiently, routing yet works consistently in the basic case of the nodes being arranged in the ring structure.

In Chord, nodes also are arranged into a ring. However, in contrast to ISPRP, a working routing protocol is required as a *prerequisite* for the construction there. It is unique to ISPRP that *no routing protocol is required prior to constructing the ring*. Each node in the network only needs to have link-layer connectivity to its surrounding nodes (its one-hop neighbors).

2.2 Flooding

A simple and straightforward way to disseminate routing information is *flooding*. Many protocols for routing in ad-hoc networks employ some form of flooding, e.g. the popular *Ad hoc On-Demand Distance Vector (AODV)* protocol [4]. Flooding can also be used to create the ring network in our problem setting. For this, each node first selects its tentative successor from its local one-hop neighbourhood. Subsequently, each node floods the whole network with a message containing its own address. Upon reception of these messages, nodes check if the originating node precedes their current tentative successor in the virtual address space and change their pointer whenever appropriate. After all nodes have flooded the network, every node will have seen the identifiers of all other nodes on the network and thus have chosen its correct successor.

From a flooded message, nodes can only learn the addresses of the message source and the last hop which forwarded the message. As message headers only contain (*source address*, *destination address*) pairs, intermediate nodes have to determine the next hop to which they forward messages to. Therefore, each node has to store the source and last hop addresses *of every message it received during the flooding phase* in order to guarantee correct routing, as it otherwise will not be able to route messages towards an arbitrary destination.

This corresponds to distance-vector routing schemes where each node has to store (*interface*, *destination*) pairs. Only as a result of the administratively and hierarchically assigned node addresses, nodes can aggregate and thereby reduce memory requirements below $O(n)$. In our scenario, however, the random address distribution requires each node to store $O(n)$ state information. Moreover, flooding leads to a huge volume of messages, i.e., $O(n^2)$ messages in total. Consequently, memory requirements and energy consumption as a result of the huge transmission volume render flooding protocols an unfavourable choice.

Memory requirements for storing state information which scale linearly with the network size quickly become prohibitive, since our goal is to build the routing overlay among limited-capability devices. As mentioned above, one way to reduce memory consumption is to employ *source routing*, where each node only has to memorize the path to its current tentative successor.

2.3 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.

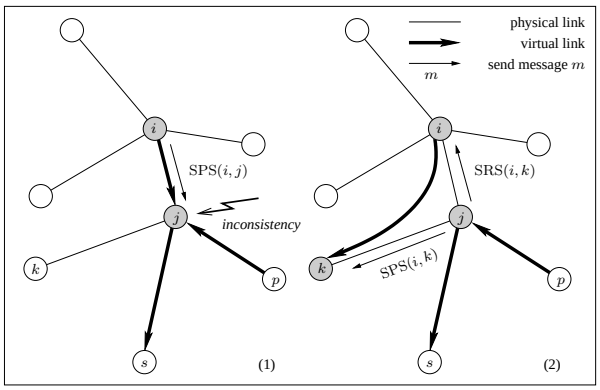

Figure 1: One iteration in applying the ISPRP

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 $\mathrm{SPS}(i,j)$.

Upon reception of an $\mathrm{SPS}(i,j)$ 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, i.e. 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 (i.e. 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 $\mathrm{SRS}(i,k)$. In order to reduce the number of control messages, j directly sends the $\mathrm{SPS}(i,k)$ message required in the next step to k itself (fig. 1, step (2)). As i receives the $\mathrm{SRS}(i,k)$ 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*.

The structure formed by executing the ISPRP might not always be a ring. As a result of wrapping the address space around at its borders, nodes could organize themselves into two intertwined rings, e.g. $0 \rightarrow 2 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 0 \rightarrow \ldots$. This global inconsistency cannot be detected by ISPRP since the wrap-around point leads to ambiguities here. They can, however, be easily detected by the nodes attempting to route messages to themselves [7].

In rare cases, the wrap-around could also lead to the formation of two independent rings. This happens if the network can be partitioned by breaking only a few links. In a situation like this, ISPRP might not find a path from one part of the network to the other one. It is sufficient for an arbitrary node to find its successor in the other network part. In each ring, there is exactly one node whose successor pointers crosses the wrap-around point of the address space. Thus, once the execution of ISPRP should yield independent rings, by each of those nodes flooding the network, paths between them are found and the rings are joined, leading to a globally consistent state. Note that this is significantly different from each node having to flood the network. With our solution, the increase in the total per-node message amount equals the number of independent rings formed, in the example 2.

To enable routing between nodes and their successors, source routes have to be recorded in the headers of messages (cf. sections 2.1 and 2.2). As a result of ISPRP's local interactions and its iterative nature, path lengths will be sub-optimal. To reduce path lengths, nodes shorten their successor paths using Dijkstra's algorithm. It is likely that prior to this, cycles in the paths exist, since with the iterative rewiring procedure, messages may travel over the same part of a virtual link several times. (Note that no additional messages are exchanged for this clipping procedure.)

3 Evaluation

To evaluate the efficiency of ISPRP in different network environments, and to compare its performance to flooding, we have implemented and simulated the protocols as described in section 2 using the language C.

3.1 Evaluation Methodology and Performance Indicators

For being able to evaluate their performance, the protocols are run in networks of the sizes n from 100 to 1000. Different random graph models have been chosen in order to study their effect on protocol behavior: 1. The *Erdős-Rényi* random graph is a simple model in which edges are added between two arbitrary nodes with fixed probability p (here, 0.1). 2. A random graph model for fixed networks, or wireless networks where some nodes have transmission ranges higher than most other nodes is the *power-law* random graph. Here, node degrees are distributed according to a power law: Few nodes are highly connected ("hubs"), whereas the majority of the nodes only are connected to few other nodes. Shortest path lengths in power-law random graphs commonly are small. 3. The *unit-disk* model is convenient for homogeneous wireless networks like sensor networks. A transmission radius of one unit is set for each node. Nodes are dispersed over a square area according to a uniform random distribution. By varying the edge length of the square, node density can be controlled (here, each node has 13 one-hop neighbors on average).

To ensure statistical significance, ten distinct simulation runs have been made for each of the network configurations. We assured connectedness of each network graph generated in order to avoid misleading results. This is no restriction as ISPRP also works in partitioned networks (cf. sec. 2.3).

Each simulation run starts with an uninitialized network and terminates once a consistent state has been reached. For the flooding protocol, this means that each node floods the whole network before the created overlay is consistent. For ISPRP, execution is stopped once the nodes cannot detect any more local inconsistencies.

Note that in the early state of the work as presented here, we did not take into account sophisticated MAC layer protocol models or link delays.

Efficiency of ISPRP is measured in two different terms: 1. The average path length from a node to its overlay successor. 2. The average number of messages sent per node in the time spanning from the uninitialized network to the correctly initialized overlay. In our scenario, the primary figure of merit is message overhead, as sending and handling messages translates into radio transmissions and computational effort, i.e. ultimately the depletion of the nodes' energy.

In our scenario, packets are broadcast via radio and a transmission from a node to all of its one-hop neighbors in the network graph is thus counted as one message.

3.2 Results and Discussion

Figures 2–4 depict the simulation results. For each random graph model, we plot the successor path lengths achieved with flooding versus those of ISPRP (left hand side) and the respective per-node number of messages sent on average (right hand side). (Note the logarithmic scale of the vertical axis for easy comparison of the respective factors.)

Note that in any scenario, flooding incurs a high message overhead. For each node flooding its address into the network, each other node has to forward the message once, leading to a total overhead of n messages sent per node.

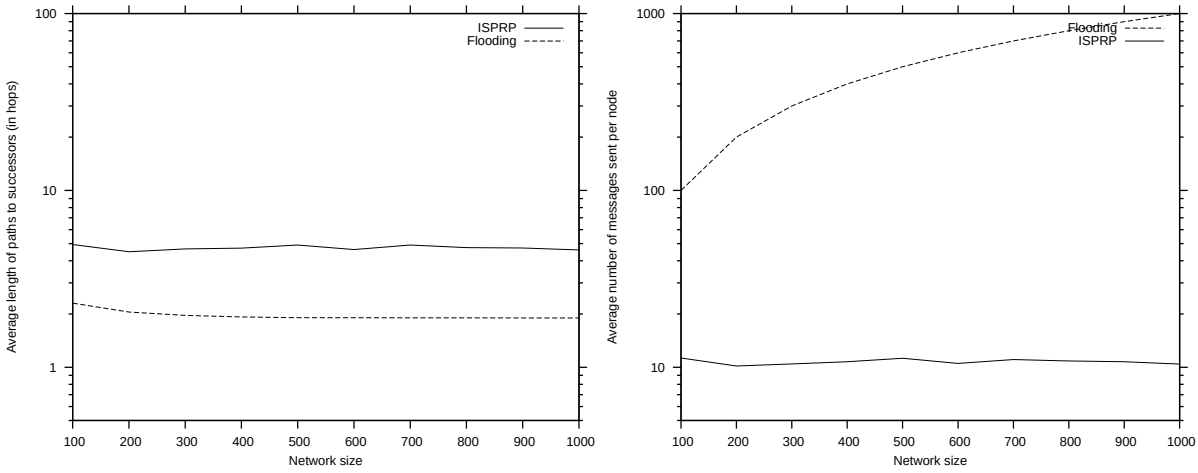

Figure 2: Erdős-Rényi random graph

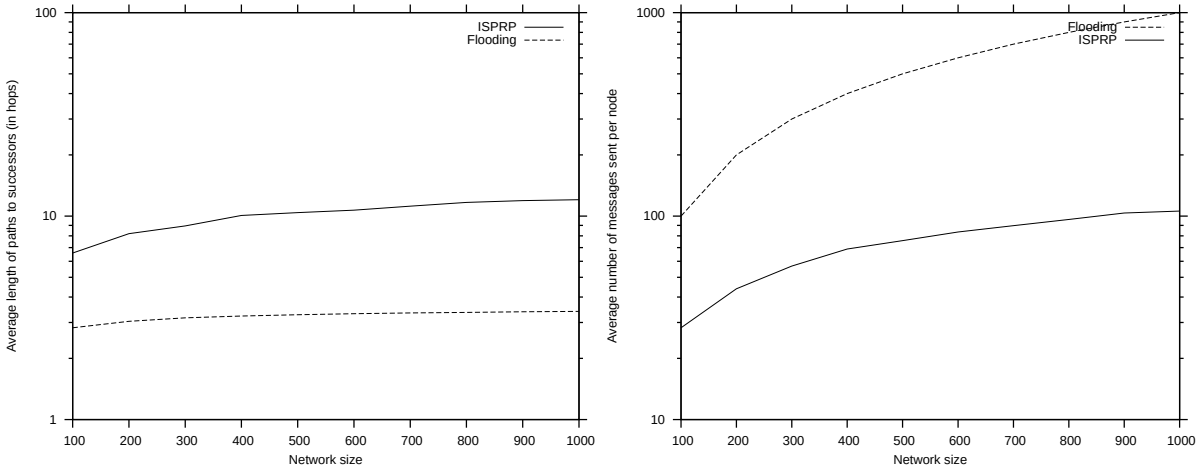

Figure 3: Power-law random graph

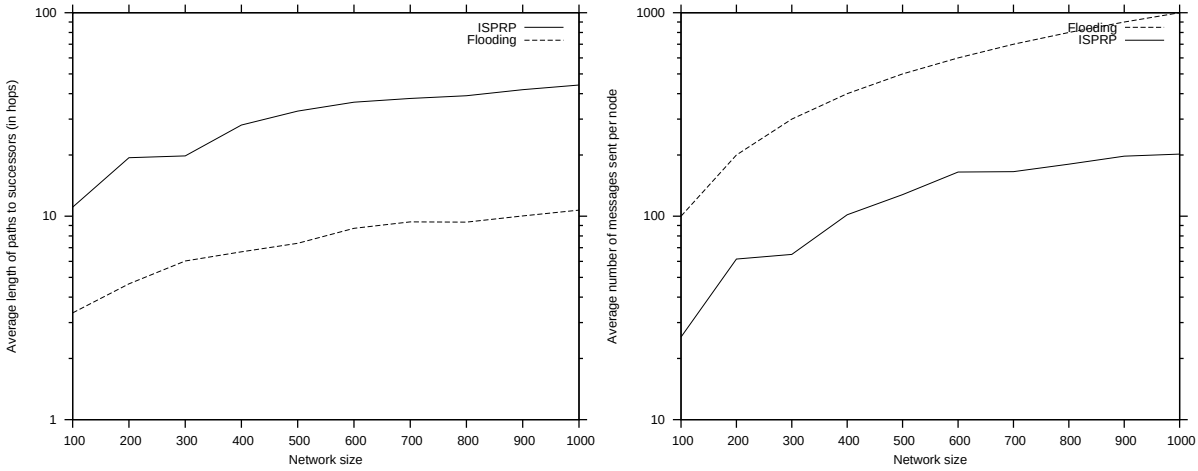

Figure 4: Unit-disk random graph

This is regardless of the network topology.

ISPRP, however, performs very well and issues only 10 messages on average in the Erdős-Rényi scenarios, independent of network size. ISPRP clearly achieves the efficiency goal, even in this case of a sparsely connected network. A similar discussion applies to the evaluation in power-law random graphs. There, message overhead increases slowly with network size, but still is several orders of magnitude lower than with flooding. This is also the case with the unit-disk graphs, where ISPRP's overhead is higher than in the others as a result of the different path length scaling properties.

With ISPRP, there is a trade-off between the achieved path length from a node to its successor and the message overhead. As discussed in section 2.3, the reason for this is that ISPRP only performs local corrections to the successor pointers. Flooding marks the bottom line of achievable path length – with it, the shortest successor paths are always taken. For all scenarios, the increase in path length is outweighed by the reduction in control traffic. In the unit-disk graph, both factors are roughly the same. As stated before, the reduction of control traffic is our primary interest since the goal of ISPRP is the efficient initialization of the network. In the Erdős-Rényi scenarios, there is a stretch (defined as the ratio of the length of the path found to that of the shortest path) of about 2.4. In the power-law scenario, stretch varies between 2.3 and 3.5, depending on the network size. In the unit-disk scenario, stretch is between 3.3 and 4.1.

Although ISPRP yields path lengths that exceed those of the shortest possible paths by a factor of about 4, we consider the trade-off between the longer path lengths and the huge reduction in control traffic overhead to be excellent. Further simulation studies on which we cannot report here indicate that further improvements are possible using the aforementioned refinement mechanisms. They can be applied once ISPRP has constructed the basic ring.

Altogether, we thus conclude that in the evaluated scenarios, ISPRP achieves both of its design targets, efficiency and simplicity.

4 Related Work

Other authors have also considered building a network-layer routing protocol using P2P techniques [1, 2]. 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 target at the Internet.

Hu et al. propose to use the Pastry DHT for unicast routing in ad-hoc networks, where routes between virtual addresses are discovered using the DSR routing protocol [3, 5]. Their work differs from ours in that we do not employ flooding to discover routes.

5 Conclusions and Future Work

In this paper, we have presented the *Iterative Successor Pointer Rewiring Protocol (ISPRP)*. It 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 overlay among limited-capability nodes which assign addresses to themselves. Simulation studies presented here have shown that ISPRP meets its goal and creates the P2P overlay with very low control message overhead.

The ISPRP is the beginning of further research steps as outlined throughout the paper. Although the establishment of a ring-structured routing network by ISPRP is fundamental to correct routing, the ring must be extended by classical DHT mechanisms to achieve full functionality.

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, we are currently extending the protocol to have it handle dynamics, i.e. nodes joining and leaving the network, as well as mobility. There, route breaks have to be detected and repaired, e.g. similiar to the way AODV [4] maintains them.

References

[1] 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.

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.