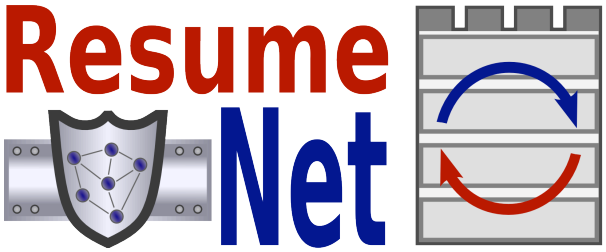
Resilience and Survivability for future networking: framework, mechanisms, and experimental evaluation



Deliverable number	3.4
Deliverable name	Overlay-based end-to-end connectivity
WP number	3
Delivery date	1/3/2011
Date of Preparation	7/3/2011
Editor	Nils Kammenhuber (TUM)
Contributor(s)	Stephan Günther (TUM), Ralph Holz (TUM), Nils Kammenhuber (TUM)
Internal reviewer	Andreas Fischer (UP), Paul Smith (ULanc)

Summary

This deliverable describes ongoing work on an overlay framework that helps to protect IP connectivity between hosts. We describe challenges against which we protect; we develop requirements and design of the overlay framework. Our main contribution is a description of Perco-Pastry, our overlay routing algorithm. Furthermore, we describe our method to derive a node's geographic location from RTT measurements.

Contents

1 Introduction

One of the key ideas of the Internet was that every host can send data to any other host. Even though middleboxes such as firewalls, NATs and other packet filters nowadays enforce policy-related restrictions on this freedom, this is still true for the general case.

However, if IP packets that are sent from one host to another systematically get lost, there normally is no way for the sending host to use a different path – routing in the Internet is destination-based[1], and the routing decisions are made subsequently and independently by the routers along the path. Although there exists an IP option that allows the sender to specify the route [Inf81], almost all routers and firewalls ignore this field due to security and various concerns: User-controlled routing eases denial-of-service attacks and may also circumvent routing policies set up by the provider.

To overcome this problem, we propose to make use of an overlay. Nodes in this overlay can be used as relays. If the connection $A \rightarrow B$ is disturbed, then it may still be possible for A to send packets via some relay nodes in the overlay, i.e., $A \rightarrow R_1 \rightarrow ... R_i \rightarrow B$. For this purpose, the packets are encapsulated as long as they are travelling through the overlay. As long as nodes will still be able to communicate via relay nodes in the overlay, this solution can greatly help to increase resilience of IP-based services. Apart from pure services, the overlay infrastructure can also be useful for IP-based network management mechanisms that are intended to enhance network resilience, such as distributed challenge detection mechanisms (e.g., deliverable D2.2a).

This document sketches an overlay design that allows to be used for these purposes. In terms of our D^2R^2+DR methodology, it functions as a remediation mechanism. We furthermore present some preliminary results of evaluating a prototype implementation.

The remainder of this document is structured as follows: Section 2 describes the challenges against which the overlay can protect. In section 3, we draw up requirements that the overlay should fulfill. These drive the overall overlay design that we describe subsequently in section 4. Section 5 presents a routing algorithm suitable for our purposes and gives more details of our prototype implementation based on the Pastry framework [RD+]. Section 6 describes our approach of determining the geographical address of an overlay node based on end-to-end measurements. We present related work in section 7, and we conclude our findings in section 8.

2 Challenge description

In this section, we will describe the challenges for which we want to provide a means of remediation in greater detail, and we relate them to the challenge categories devised in deliverable D1.1, section 3.1.

2.1 Causes of problems

We thus want to tackle the following typical scenarios that result in excessive loss of IP packets, up to a complete loss of IP connectivity from a host A to a host B:

[1] Although IP offers the option to specify a source route in the header of an IP packet, routers are practically always configured to ignore this field, due to security considerations. Therefore, the only practical solution to send a packet along a path that is different from the one prescribed by IP routing is to employ some kind of tunneling mechanism.

Failures in the routing system. For example, misconfigurations or software bugs may result in packets being lost. Reaction to these failures often is rather slow, since detection as well as eradication of the failure usually depends on human interaction.

Challenge categories: Component faults, hardware destruction, human mistakes, failure of a provider service.

Slow reaction of routing protocols to link failures. Intradomain routing protocols are comparably quick when reacting to network failures; they normally can restore full connectivity on the timescale of seconds [GRF03]. However, due to the various damping methods that are built into BGP (e.g., MRAI timer, route flap damping, etc.), this does not hold for interdomain routes – it is not uncommon that the routing is disturbed for minutes, or even longer [TSGR04].

Challenge categories: Component faults, human mistakes, failure of a provider service.

Unacceptable performance degradation on some network paths through the network. For example, this may be due to congestion events occuring on a short time scale (e.g., traffic spikes, hot spots), which routing protocols normally do not protect against. Another example is a temporal signal degradation in a wireless link, e.g., due to a thunderstorm. Note that these events usually do not result in full connectivity loss between two nodes whose connection traverses the affected link – rather, performance parameters such as packet loss or delay will deteriorate significantly. Therefore, it may be sufficient to only shift *some* traffic away from the impaired (direct) connections but not *all* traffic in these failure cases.

Challenge categories: Component faults, human mistakes, malicious attacks, unusual but legitimate demand for service, failure of a provider service.

Nodes that have turned malicious. A node may start to selectively drop traffic that it is meant to forward to others, or otherwise misuse or even manipulate that traffic.

Challenge categories: Malicious attacks.

In many cases, these causes are interleaved or cannot be distinguished. For example, in the famous case of Pakistan Telecom disabling YouTube [RIP], a grave router misconfiguration had the same effect as if Pakistan Telecom had tried to maliciously disturb YouTube's traffic.

2.2 Characteristics of failures

In general, we can consider IP connectivity failures to be unidirectional, i.e., if traffic exchange between two end hosts $a \rightarrow b$ is disturbed, this does not necessarily imply that the reverse direction $b \rightarrow a$ is affected as well: First of all, the affected link may be affected in only one direction (e.g., in the case of congestion); second, routing is often asymmetric, especially when traversing multiple ASes. Of course, bidirectional failures may be catered for by treating them as two individual unidirectional failures.

We aim to render the time scale at which our overlay system can react to routing failures as small as possible. Realistically, we expect to be able to offer an alternative route via the overlay one the scale of seconds. Although this does not help against very short disturbances in the sub-second domain (e.g., transient traffic spikes or network failures that are recovered through intradomain routing protocols or protection switching), it nevertheless can help against problems with interdomain routing, or problems with routing configuration that would require manual interaction by a network operator.

2.3 Impediments that are not failures

Although connectivity between hosts may be impaired through the presence of firewalls and NAT devices on a path between these hosts, we do not consider this to be a failure case. Therefore, levelling the effect of such middleboxes by creating a facility to obtain unrestricted IP connectivity through a firewall is not the primary concern of the overlay, since these impediments usually are introduced with a specific purpose in mind. Nevertheless, it may be a useful property of the overlay if it actually can help mitigating connectivity restrictions imposed by middleboxes.

2.4 Affected services

Trivially, impediments to IP can affect all services that depend on IP. Obviously, this includes popular services such as Web services, video streaming, or e-mail.

Furthermore, the Internet normally does not provide separate channels for signalling and management traffic (i.e., no out-of-bound signalling). For example, routing protocols like BGP, management protocols like SNMP or `telnet` used to configure network components, or other resilience-enhancing mechanisms like those developed in the context of ResumeNet like distributed challenge detection mechanisms[2] also rely on IP. Therefore, this traffic pertaining to the signalling plane is routed along the same paths as traffic on the data plane. This leads to the paradoxical situation that IP connectivity is actually needed the most (e.g., to quantify the extent of an IP service disruption) when it is available the least (i.e., due to these very disruptions).

3 Requirements

In this section, we describe the most important requirements that an overlay-based solution should fulfill.

Service: Our goal is to protect IP service against the disruptions and challenges that we described in section 2. We have to do this protection transparently to other services operating on top of IP, so that they do not require any adaptations. The easiest way to achieve this transparency is to encapsulate IP packets into overlay messages when it is determined that the IP packets are to be sent via the overlay instead of using standard IP routing (underlay).

Scale: There will be potentially many peers participating in the overlay. The system thus needs to be built such that it scales well with a large number of participating nodes. Furthermore, there is no fixed set of participants. The system should be designed such that it can cope with significant churn (i.e., a high frequency of join and leave events). This is expected to hold for at least specific types of hosts (e.g., end user nodes), whereas other hosts are expected to remain stable, i.e., rarely signing on/off (e.g., routers or servers).

Security-related assumptions: We assume that only benevolent participants take part in the overlay, i.e., no adversaries. In a later step, this scenario may be amended such that possibly malicious behaviour of overlay participants is addressed. Thus the system

[2] see e.g., deliverables D2.2a, D2.2b, D2.3a, D2.3b, D2.4, D3.2

should be designed in a way such that this secondary goal can be achieved without heavy redesign.

Having said that, it is clear that there will be neither a certification nor a trust mechanism built into the network. However, the system should be designed in a way such that it should be easy to extend the overlay framework later in this regard.

Limitations of our approach: The assumptions described in the previous section 3 and our restriction of failures which we introduced in section 2.3 naturally limit the scope and thus the applicability of our proposed overlay. Apart from these restrictions, we also should mention that an overlay that protects IP connectivity can only be seen as a means of *remediation*, i.e., protection against the most disruptive consequences of a challenge. In a reasonable networking context, routing protocols and other means should be able to restore normal operation (i.e., *recovery*, according to the $D^2R^2 + DR$ terminology) after some time.

Contrasting routing requirements: As was described in section 2.4, our overlay should protect common IP-based services, e.g., Web services, e-mail, video streaming, etc. On the other hand, we also have to provide an additional means of communication for other resilience-enhancing mechanisms (cf. section 2.4). However, these two aspects of the overlay software impose fundamentally different requirements on the connection quality that the overlay delivers: Most popular end-user services (e.g., WWW, file downloads) typically require large bandwidths and possibly small delays, whereas signalling and management protocols normally require only a small bandwidth but benefit from connections that are as stable as possible.

This more or less constitutes a dichotomy, since the most stable paths are not necessarily the most performant ones, and vice versa. Therefore, the overlay software needs to take the type of data into account when making routing decisions. We will elaborate on this point in section 4.1.

4 Design

We now describe the most important design aspects of an overlay that can handle the challenges described in section 2 while conforming to the requirements that we laid out in section 3. Note that some design questions cannot be answered through theoretical considerations; rather, they require further practical analyses. This is ongoing work; we will report our final results in a follow-up deliverable (D 3.1c).

4.1 One overlay software, two aspects

As we described in sections 2.4 and 3, requirements of typical service traffic can vastly differ from those of signalling and management traffic, which in turn needs to be reflected in routing. As with traditional networks that can be seen to be divided into a data plane and a signalling plane, therefore is useful to divide the overlay into two aspects: a traffic forwarding overlay and a management overlay.

4.1.1 Traffic forwarding overlay

The *traffic forwarding overlay* is the overlay functionality that transmits encapsulated IP packets. It corresponds to the data plane in network architecture. The encapsulated IP packets are forwarded via nodes of the traffic forwarding overlay, instead of being forwarded unencapsulated via the IP underlay, so as to overcome deficiencies of direct IP connectivity (section 2). The forwarding overlay does not necessarily form a strongly connected graph, since most connections will only be established on-demand in the face of a challenge.

In order to forward traffic, the overlay needs to apply some kind of routing mechanism. This means that a node receiving an encapsulated IP packet needs to decide what to do with the packet, i.e., either

- selecting an overlay participant to which to forward the packet to,
- or unwrapping the packet and sending out the original IP packet, using the IP underlay,
- or – in the case of an error, such as a policy violation – dropping the packet.

Traffic on the forwarding overlay is expected to be bandwidth intensive. Therefore, the routing of this overlay should be tuned so as to provide a good performance, which mainly implies a good throughput and perhaps a low delay. On the other hand, reliability aspects play a less influential role[3] This is due to the fact that the traffic forwarding overlay is intended to be a remediation and perhaps a performance enhancing facility, but not so much a recovery mechanism.

4.1.2 Management overlay

The *management overlay* is used for forwarding requests for establishing a virtual link or other routing-related messages pertaining to the traffic forwarding overlay. In other words, it manages the service overlay. Furthermore, it can be used to transfer important signalling and management data for other resilience-enhancing mechanisms. As their correct functioning may crucially depend on a working management overlay, its nodes need to be part of a strongly connected graph that is resistant to severe disruptions, i.e., massive loss of IP (i.e., underlay) connectivity between neighbours in the management overlay.

Traffic on the management overlay is expected to be low-volume. However, it is vital that the management be fully functional at all times. In comparison to the forwarding overlay, it is thus much more critical to keep up connectedness, i.e., to ensure that all participating nodes in the overlay form one big strongly connected graph component so that the overlay cannot be split. In contrast, performance is less of an issue.

Having said this, it is not necessary that the participants of the management overlay form a full mesh topology – rather, the number of connections needs to be restricted due to scalability considerations (section 3). The peer selection mechanism thus may apply reliability considerations (also see section 5.2).

[3] Although these are undeniably crucial aspects for some applications (e.g., telemedicine), bear in mind that the IP overlay is intended to be a generic solution.

4.1.3 Integration and interaction

The two aspects can be either viewed as two separate, but strongly interacting overlays, as just described. Alternatively, they can be seen as one overlay that offers two different services; one for the actual traffic, and one for out-of-band signalling. Here lies a difference between our overlay approach and traditional IP networks, which normally apply in-band signalling.

Our separation allows messages in one overlay to be routed completely differently from the other overlay (e.g., performant paths vs. reliable paths). Note furthermore that nodes that are neighbours in one overlay (i.e., they directly exchange messages via the underlay) are not necessarily neighbours in the other.

Interaction between the two overlays is twofold:

- All signalling pertaining to the forwarding overlay (e.g., routing updates or connection requests) should be performed via the management overlay and/or using direct IP connections, if available, but not via the forwarding overlay itself.
- In an ongoing challenge such as a sudden overload situation on a link, the two overlays may compete with each other for scarce network resources. Therefore, it has to be ensured that management overlay traffic gets a higher priority than forwarding overlay traffic. It is likely that an implementation of this feature needs some interaction with the operating system kernel in a way such that management overlay packets are preferred over forwarding overlay packets and other best-effort traffic when queuing, presumably using appropriate ToS/DSCP flags in combination with an appropriate setup.

  If the network allows to make use of QoS features like the ToS bits in the IP header, these also should be used to make sure that the IP underlay forwards management messages with a higher priority than forwarding overlay traffic. However, most networks do not allow to use QoS features (at least not across AS boundaries); we therefore do not consider QoS aspects further in this document.

4.2 Difference to traditional overlay routing

A fundamental design decision is how the overlay(s) should do routing, i.e., how the paths are determined along which the messages are sent. Existing overlay implementations usually assume that every node is able to contact every other node directly, i.e., via IP or a protocol on top of IP. As was laid out in section 2.1 however, in our challenge scenarios we may at best assume that most nodes can contact most other nodes, but we definitely cannot assume that each node can contact all other nodes – after all, IP failures are just what we want to protect us from. This implies that we cannot make use of existing routing protocols (e.g., OSPF, BGP). Rather, we have to employ routing algorithms that are suited to our problem, while at the same time conform to our other requirements laid out in section 3 (e.g., scalability). Our approach for doing routing in the overlay will be described in more detail in section 5.

4.3 On-demand vs. proactive path establishment

Paths taken through the traffic forwarding overlay may not be needed for a long time in many cases. In most cases, it suffices for the forwarding overlay to establish links on-demand upon detection of a connectivity failure in the underlay. This implies that the graph of the traffic

forwarding overlay is not necessarily fully connected, but may actually consist of single links or very small islands of connectivity comprising only a handful of nodes.

In contrast, the graph of nodes in the management overlay needs to form a strongly connected component that offers many alternate paths between any pair of nodes. This increases reliability of the management overlay.

4.4 Geographic diversity

Geographic diversity means that both nodes participating in overlays as well as the underlying physical links used to connect these nodes are geographically wide-spread. Groups of nodes that are located at the same position compromise the robustness of the overlay. Furthermore, nodes in close geographic proximity tend to use the same links or geographically parallel links to connect to other overlay participants, even if they are connected to entirely different IP networks [RNS09]. Thus, one single link failure might be sufficient for all nodes located at the same site to be taken offline at once. An important aspect here is that the network's topology on layer 3 (IP) may be significantly different from the topology on layer 2. Even redundant links that are installed in close proximity may fail concurrently due to some kind of disaster. For example, the Howard Street Tunnel fire destroyed two redundant fiber optic links and led to severe impacts on Internet connectivity [Fed]. A similar reasoning is also applicable to nodes located at the same site. Consequently, one method to increase the robustness of an overlay is to ensure geographic diversity of both nodes and redundant links.

To make well-founded decisions whether a node is a suitable candidate in terms of its geographic position, a thorough understanding of the underlay is essential. In particular, knowledge about the routing paths of the underlay and the geographic positions of both routers and nodes is needed. In general however, this information is hard to obtain. First, only little is known about the physical structure of the underlay – usually, service providers are very reluctant to reveal topology or routing information about their networks, since this is regarded as a trade secret. Furthermore, one cannot assume that all nodes participating in an overlay are willing to reveal their exact geographic position, e.g. due to privacy reasons. Therefore, we require techniques that reveal as much information about the underlay as possible, without any need for cooperation – neither of all nodes in the overlay, nor of routers along the paths that the IP packets travel.

To derive information about the physical layout of the underlay, *IP Geolocation* can be employed. In the past few years there has been a lot of research in this area. Existing approaches can roughly be categorized into two distinct classes depending on where positioning information is obtained from.

4.4.1 IP Geolocation based on public data

The first class relies on a static approach which infers location data from publicly available information like the fully qualified domain name (FQDN) of nodes. However, there are many drawbacks associated with this class of techniques. First, node names are not available for many routers, or they do not reveal position information in a standardized, or even obvious, way. Second, some kind of database must be maintained to associate IP addresses with locations. Inconsistent, outdated, or wrong datasets are inevitable.

4.4.2 Measurement-based IP geolocation

The second class of techniques is based on active measurements. They rely on estimates of the time that a signal needs to propagate from source to destination as basic source of information. In contrast to the widely known distributed Vivaldi algorithm [DCKM04], which derives node coordinates in a *virtual* (non-real, possibly higher-dimensional) coordinate system, we want to estimate a node's *real* position on the earth's surface. The idea is to use a set of *landmark* hosts with known geographic positions. The majority of approaches further demand that landmarks are able to measure the RTT to an arbitrary unknown node.

To estimate the position of a target, constraints for the target's position are derived. For instance, a landmark probes the target and obtains the RTT, which is mapped to a distance estimate between landmark and target. In conjunction with the landmark's position – which is assumed to be known – a feasible region can be defined wherein the target is likely to be located. This approch is used by *Constraint Based Geolocation (CBG)* introduced by Gueye et al. in [GZCF06]. There are other more complex techniques like *Topology Based Geolocation (TBG)* introduced by Katz et al. in [KBJA+06]. TBG for instance also obeys intermediate targets, i.e. the routers along the path from a landmark to a given target, hence the naming *topological*. [KBJA+06] also provides a small overview on and comparison of IP geolocation techniques.

Although measurement-based approaches are not perfect, they are very promising. Especially their ability to automatically adapt to changes makes them superior to static ones. This does not necessarily mean that position information of known nodes is ignored. If topological information can be considered as it is possible with TBG, advantages of both classes can be combined. Our approach to determine the geographic of a node on the base of distributed measurement of IP round-trip times will be explained in greater detail in section 6.

5 Routing Around Failures: Management and Forwarding

The concepts to improve IP connectivity that we presented in section 4 offer many different parameters and settings (e.g., what metric to use for routing, etc.). To find an optimal setting of parameters is thus extremely time-consuming. Therefore, only a subset of the concepts to improve IP connectivity presented in section 4 could be reasonably investigated within this task due to time limitations.

We thus decided to address the case of missing IP connectivity between two given nodes on the Internet. The most common case is failure of a link or router along the path of the IP packets. Normal recovery mechanisms for IP are assumed to be too slow (section 2.1) or have also stopped working. We thus need to route around the affected nodes.

5.1 General Concept

Management and forwarding overlay are actually not completely distinct instances. They share the same address space, i.e., id_A identifies node A both in the management and in the forwarding overlay. Different algorithms and message types can be used, depending on the required functionality, but the address space is always the same. There is one difference, however: while all nodes are required to participate in the management functionality, not all nodes are required to participate in the forwarding functionality. This means that while the

nodes in both overlays overlap to a (great) degree, different routing state has to be kept for both functionalities.

The management functionality is straight-forward to construct and define. The forwarding functionality is a bit different here. As our use case is missing IP functionality, we do not construct mechanisms for pure overlay-based forwarding. Instead, we use the management functionality to establish tunnels for IP packets.

An important part of our design is that all nodes run an instance of a measurement software. For this purpose, we use the UNISONO measurement framework (cf. 7). Measurement information can be exchanged using the management functionality. Information that is of particular value is geographic location, RTT and neighbouring AS systems. System properties like available resources can also be queried.

5.2 Using Pastry for the Management Overlay

In the remainder of this section, we describe our proposal for a combined management and forwarding overlay. This is based on the Pastry substrate for Key-Based Routing. In essence, we require all nodes to participate in this Pastry instance in order to constitute the management overlay.

In the following, we motivate our choice of Pastry. We then describe the management overlay's methods establishing forwarding functionality for missing IP links.

Our reasoning for choosing Pastry is as follows:

- Pastry is available as a mature Open Source implementation. It has a well-designed Java API. It allows to design several 'applications' that run on the same Pastry node [RD+]. One such application will be our Perco-Pastry extension (section 5.3). Others, which have not been implemented, could be signalling for SIP, for DISCo (see deliverable D2.2a), etc.
- The nodes of a Pastry network form a strongly connected graph component. This property means that Pastry already offers a significant part of the functionality that is required by the management overlay.
- Pastry supports Proximity Neighbour Selection, which allows to optimise the forwarding in the overlay network according to the underlay metric of latency. The result is that Pastry forwarding has a minimised stretch, i.e. the ratio of overlay latency to underlay latency is minimised.
- Furthermore, Pastry allows to install a custom, user-defined metric that is to be used for proximity neighbour selection. This allows to tune the Pastry-based management overlay to greater resilience. The current metric, RTT, mainly reflects the length of the cables between two hosts and therefore the likeliness of a cable cut. Other meaningful metrics that we propose to use are the number of hops, which reflects the probability for router failures along the paths; the number of ASes traversed, which reflects the probability for BGP failures; the uptime, which reflects the likeliness that a node will remain in operation due to the probability distribution of node uptimes [Fes10]; or a compound of the mentioned metrics.

We had initially also considered the Kademlia algorithm for this. Compared to Kademlia, Pastry has both advantages and disadvantages. The primary advantage is that Pastry supports

the Common API [DZD+03], i.e., it splits the functionality of routing, forwarding and DHT storage. Kademlia does not do this. Furthermore, Pastry actually forwards messages along the overlay, whereas Kademlia uses an iterative lookup to determine an underlay address for a given Kademlia identifier and then routes through the normal underlay. This makes it difficult to derive a set of nodes that could be used in a source route for overlay transport. On the whole, Pastry is more preferable.

5.3 Pastry Percolation Extension (Perco-Pastry)

We will now describe our extension to the Pastry routing algorithm that is intended to route around failed links. The algorithm is not fine-tuned yet and is currently undergoing evaluation. The name *Perco-Pastry* alludes to the intention that the algorithm eventually allows to provide IP connectivity even for a seemingly impenetrable IP network (i.e., defunct direct IP connectivity).

5.3.1 Scenario

Our algorithm is intended for the case where IP connectivity between two given hosts on the Internet has broken down. However, a Pastry overlay exists in which the two hosts and some other hosts participate. The Pastry-ID of a host H is simply the hash value of the IP address and the UDP port on which the Pastry is listening: $ID_H = hash(IP_H, Port_H)$.

5.3.2 Principles of Perco-Pastry

When direct IP connectivity breaks down, the algorithm will attempt to use the Pastry overlay to find a way to set up a tunnel IP between the hosts. The operation works in three distinct phases.

Phase 1: Normal Pastry Routing. In the first step, the source host will compute the Pastry ID of the destination host and attempt to route a normal message through the overlay.[4] This is done with a *seeker message*. If the seeker message arrives at the destination host, an IP tunnel will be set up between the source and destination hosts. This IP tunnel consists of segments: each segment corresponds exactly to one hop in the overlay.

This phase is depicted in Figure 1. Phase 1 has the advantage that the performance penalty is relatively small: For an overlay, Pastry routing is reasonably fast.

Perco-Pastry will switch to Phase 2 in the following cases:

- Phase 1 has failed to set up a tunnel.
- The failed link state continues to persist and there is a need to set up a more efficient tunnel (e.g., with lower delay).

Phase 2: Seeking Alternative Routes. Should Phase 1 not succeed in setting up a tunnel or should Perco-Pastry decide to look for more efficient tunnels, it will make a first attempt at finding an alternative route that is not entirely dependent on Pastry routing.

[4] For example, the Pastry IDs can be chosen as an MD 5 hash over the IP address of a host. This makes it very easy to determine a Pastry ID for a given IP address.

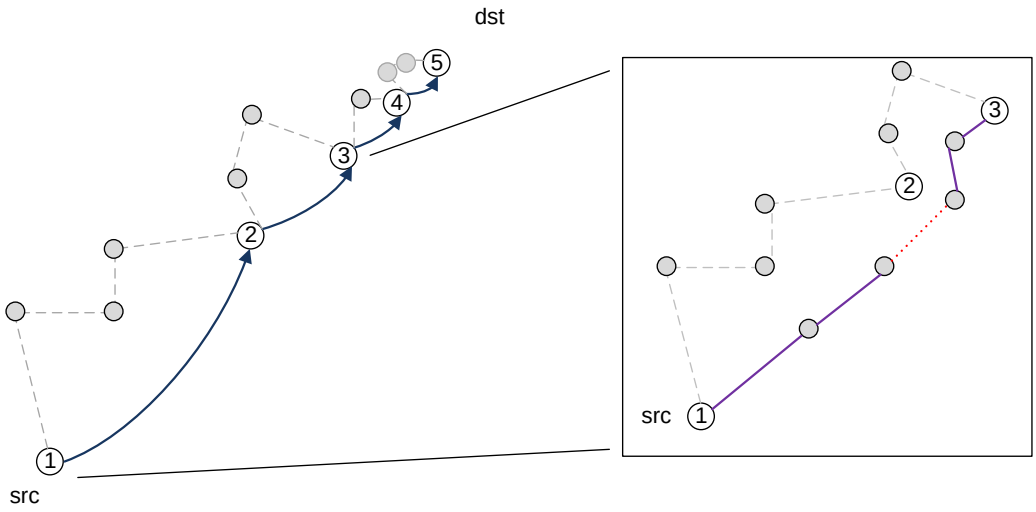


Figure 1: Phase 1 of Perco-Pastry. An IP link (right side, red dotted) has failed. Normal Pastry routing is used to route around it. With Pastry routing, the path in the underlay is different (left side, grey dashed).

The idea is again that Pastry routing will be used to forward a message. However, at every hop, a Pastry node will now conduct a traceroute to the destination host to find out if there is IP connectivity and at which cost (RTT). Every host that can successfully contact the destination host will report back to the source with a success message. It will then be chosen to set up a tunnel. If there is a choice between several hosts, the one will be chosen that offers the best efficiency. This phase is depicted in Figure 2.

An improvement here can be to implement the tunnel in such a way that a new, more efficient tunnel can be established as soon as one is found.

Perco-Pastry will switch to Phase 3 only if Phase 2 does not succeed.

Phase 3: All-Out Attempt at Finding a Route. Phase 3 is initiated if both Phase 1 and Phase 2 did not succeed in establishing an IP tunnel. The idea of this phase is a sort of 'all-out' attempt at finding hosts that have IP connectivity to the destination host.

In a first step, Perco-Pastry will route a message to the next hop on the normal path to the destination host in the overlay. This message will contain a request to initiate Phase 3 searching for IP connectivity to the destination host.

The recipient reacts by selecting k other Pastry nodes to which it has connectivity. To each, it sends a message asking it to reply whether IP connectivity to the destination host exists or not. Selection is based on a metric. At the time of writing, this can be either RTT or geographic position (low RTT is better, closer geographic position is better).

Each of the thus queried hosts will answer. This answer is going to be sent back to the source. If it is positive, an IP tunnel just like in Phase 2 will be set up.

If the answer is negative, the source will continue to search iteratively. It will choose one among the nodes that have been queried so far and ask it again to select k nodes to query on its behalf, just like before. The hosts will again reply directly to the source.

This pattern continues iteratively until either an IP tunnel can be set up or a time-out is

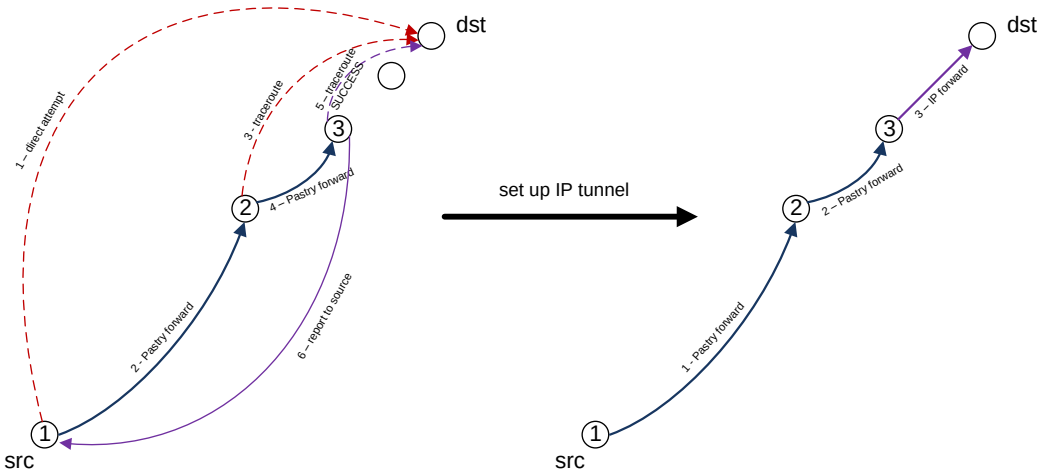
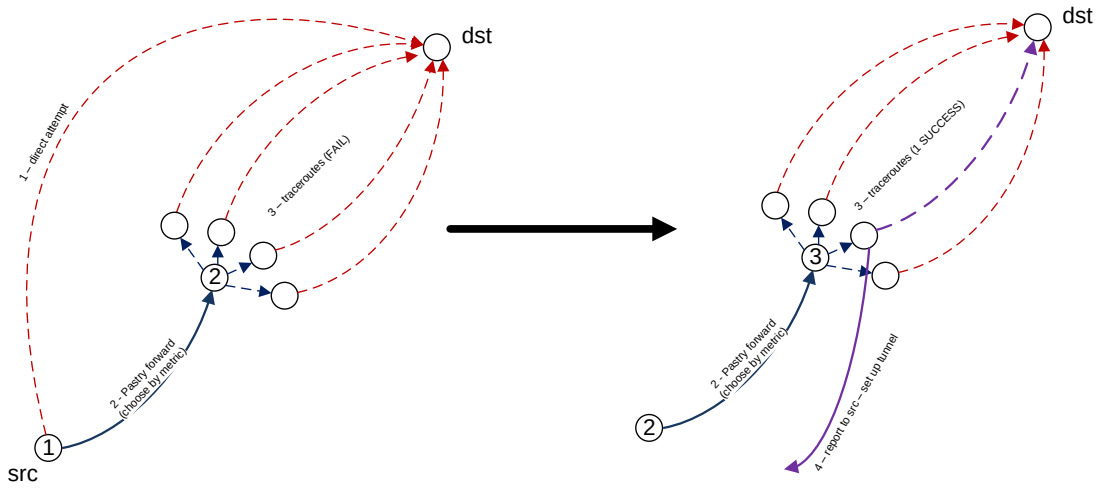


Figure 2: Phase 2 of Perco-Pastry: tracing from each hop and reporting to the source.



reached. The nodes that the source selects will be based on a metric chosen by the source, usually geographic position, position in the overlay, or RTT.

5.4 Ongoing Evaluation

Due to the delayed processing of T3.5, we are still evaluating Perco-Pastry. UNISONO, as detailed before, can be used to gain the measurement values that we need for our experiments and simulations. Our approach to evaluation is two-fold.

5.4.1 Practical Evaluation

First, we are conducting practical PlanetLab experiments. We have set up a PlanetLab configuration to test which results Phase 1 can yield. The part of Phase 1 has been implemented with FreePastry and is currently tested.

In particular, we measure which paths in the underlay packets take when they are routed via the overlay, compared to direct IP connectivity. This will yield first result regarding the efficacy of Phase 1. Currently, we are using Pastry's standard Neighbour Proximity metric for the management overlay, but this may change as our implementation and ongoing experiments mature.

5.4.2 Underlay Model and Simulations

Second, we are currently investigating a suitable underlay model that can be used in simulations. This underlay model is derived from actual measurements of the amule Kademlia network. The assumption is that, while Kademlia is a different forwarding algorithm, the statical properties of the participating nodes can be assumed to be similar to our own scenario. Phases 2 and 3 are supposed to be tested in a simulator.

6 Ensuring geographic diversity through IP geolocation

All measurement-based approaches for IP Geolocation (cf. section 4.4.2) rely on RTT measurements. If topological information can be used, not only the RTTs between landmarks and the target are needed, but also RTTs to nodes along the path to the target. Thus, the measurement phase is to a large extent independent of the actual Geolocation approach being used. The integration into the overlay software (e.g., Perco-Pastry) can therefore be implemented as outlined in Fig. 3. The Geolocation service is provided by an Unisono plugin (Section 7), which is accessible by overlay applications through the connector component Clio (Section 7). Furthermore, Unisono provides a couple of plugins that can measure the RTT to a target or along the path to a target.

Assume that $\mathcal{L} \subset \mathcal{N}$ denotes the subset of all nodes $\mathcal{N}$ participating in the overlay that are willing to provide geolocation services. We refer to $\mathcal{L}$ as the *set of landmarks*. Nodes that do not provide geolocation services still have an active GeoLoc plugin for Unisono, which is used to handle localization requests. The GeoLoc plugin maintains a list of nodes $i \in \mathcal{L}$ no matter if the node itself is a landmark or not. This can be accomplished by using a DHT storing the mapping between participating nodes and their overlay IDs. The process of geolocating a node thus looks as follows:

1. When an overlay application on node k requests localization services, its request is passed to the GeoLoc plugin through Clio. It notifies all nodes $i \in \mathcal{L}$ about the request and the target's IP address (see Figure 4a).
2. Each landmark $i \in \mathcal{L}$ probes the target x to obtain the desired RTT samples. If path information is needed, nodes along the paths (i, x), $\forall i \in \mathcal{L}$ may also be probed (see Figure 4b).
3. The results are passed back to the GeoLoc plugin of the initiator k which is thereupon able to locate node x based on the information received (see Figure 4c). The resulting position estimate is passed up to the application that initiated the request.

After a node has been geolocated, this information then can be taken into account in routing decisions, cf. section 5.2.

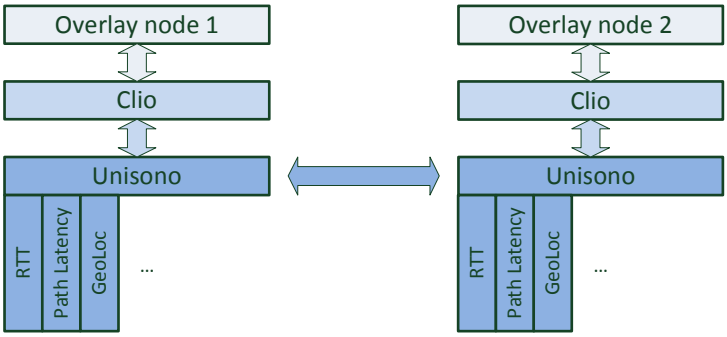


Figure 3: Integration of geolocation capabilities into the overlay.

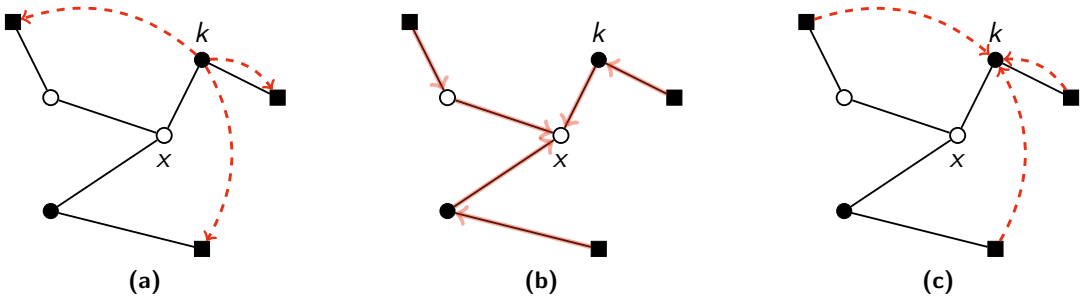


Figure 4: Overlay node k initiates a localization request for node x by notifying the landmarks $i \in \mathcal{L}$ (a). The landmarks probe node x to obtain RTT samples (b). The results are passed back to the initiator k which can perform the localization based on the information obtained (c).

7 Related work

N2N [DA08, nto] is a distributed layer-2 VPN (i.e., it can forward any type of Ethernet frames, not just IP packets). However, it only allows one-hop connections. It provides a facility for NAT traversal.

Similar to N2N, **Tinc** [tin] is a distributed layer-2 VPN (i.e., it also can forward Ethernet frames, not just IP frames). Apart from direct connections, it also allows routing via intermediate overlay nodes; the routing algorithm being used is of a link-state type.

In contrast to N2N and Tinc, **RON (Resilient Overlay Networks)** is not a distributed layer-2 VPN, but an overlay for exchanging IP traffic [ABKM01, Bea]. The seminal paper on RON was, to our knowledge, the first to introduce overlays for enhancing the resilience of IP routes. The software is intended to couple entire networks. Although routing via the overlay is possible, the algorithm in use is a link-state algorithm, which severely limits scalability. To our knowlege, an enhanced link-state algorithm with reduced signalling traffic was proposed [SZP+09] for RON, but not actually implemented into the software.

IGOR, the Internet Grid Overlay Routing network [AEHF08], is a P2P overlay network that provides a routing service similar to Chord. Unlike a distributed hash table which offers only publish/subscribe services, Igor is an overlay network that routes messages.

The **SpoVNet** project[5], in which some of the authors contributed, has produced a software package that allows the spontaneous, on-demand creation of application-specific overlays. The SpoVNet package consists of a robust implementation of an overlay creation mechanism (called ariba [BHMW08]), a multicast service (MCPO) and an event notification service (EONSON). The package also provides CLIO/UNISONO [HH09, HHNL09, HH], a comprehensive framework to determine node and link properties.

UNISONO is a measurement framework that offers acess to a multitude of measurement modules. At the time of writing, about 55 such modules are implemented; RTT, bandwidth, hop count etc. are among them, but also requests for host properties like current load, free memory, link capacities and available access technologies. UNISONO itself is a stand-alone component that can be run independently of applications or overlay networks. If a physical host participates in several overlays at the same time, information about underlay properties can be stored, accessed and shared thanks to UNISONO running only once, as a central measurement component.

CLIO is a connecting component that can be used by SpoVNet-based applications to access UNISONO and extend its functionality. It follows an Order-Reply paradigm: accessing software issues orders to CLIO. These are executed by UNISONO. This allows to aggregate orders (same orders are only executed once). CLIO can also execute so-called Remote Orders: these are requests for measurement that a node can ask another node to conduct on its behalf. Also part of CLIO's extended functionality are so-called overlay measurements, e.g. RTT or bandwidth measurements in a forwarding overlay.

In our proposal for Perco-Pastry, we make use of UNISONO and implement a CLIO-like interface to access it from Pastry.

Pastry is a P2P overlay [RD$^+$, Wik]. A fairly well-known Java implementation (FreePastry) exists. It comes with **Past**, a DHT implementation. Our overlay prototype builds heavily on FreePastry.

Tor [Tor] is an overlay network that is intended to enable online anonymity. Instead of contacting some server (e.g., , a Web server that offers incriminating contents) directly, a node running the Tor software may instead choose to contact the server via the overlay. In this case, to the sever it will seem that another computer (a so-called Tor exit node) has issued the TCP connection instead of the actual requesting node. Data within the Tor network is encrypted, with the intention to make it impossible to detect which nodes effected which traffic entering and leaving the Tor network at the exit nodes.

8 Conclusion

We have presented an overlay-based solution as a remediation strategy to protect a network against disruptions of IP and its dependent services. We described how routing in such an overlay differs, on the one hand, from routing in a physical network, as well as from traditional overlay routing. Then we described Perco-Pastry, our prototype that allows us to evaluate our novel routing concepts that we devised. The description of Perco-Pastry was followed by a description of our method to determine a host's geographic location purely based on RTT

[5]SpoVNet was funded by Landesstiftung Baden-Württemberg, Germany.

measurements at the IP layer. This allows the overlay to avoid paths that share common risks due to geographic proximity.

Evaluating and extending our Perco-Pastry and geolocation implementation is an ongoing task, the details of which will be presented in a follow-up deliverable.

References

[ABKM01] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of ACM SOSP*, Banff, Canada, October 2001.

[AEHF08] Bernhard Amann, Benedikt Elser, Yaser Houri, and Thomas Fuhrmann. IgorFs: A distributed P2P file system. In *Proceedings of the Eigth IEEE International Conference on Peer-to-Peer Computing (P2P'08)*, Aachen, Germany, September 8 – 11, 2008. IEEE Computer Society.

[Bea] Hari Balakrishnan et al. Resilient overlay networks (project homepage). `http://nms.csail.mit.edu/ron/`.

[BHMW08] Roland Bless, Christian Hübsch, Sebastian Mies, and Oliver P. Waldhorst. The underlay abstraction in the Spontaneous Virtual Networks (SpoVNet) architecture. In *Proc. 4th EuroNGI Conf. on Next Generation Internet Networks (NGI 2008)*, pages 115–122, Kraków, Poland, April 2008.

[DA08] L. Deri and R. Andrews. N2N: a layer two peer-to-peer VPN. *Lecture Notes in Computer Science*, 5127:53–64, 2008.

[DCKM04] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26. ACM, 2004.

[DZD+03] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for structured peer-to-peer overlays. *Peer-to-Peer Systems II*, pages 33–44, 2003.

[Fed] Federal Railroad Administration. Report To Congress: Baltimore's Railroad Network, Challenges and Alternatives. `http://www.fra.dot.gov/downloads/RRDev/brn1.pdf`.

[Fes10] Ali Fessi. *Resilient Application Layer Signaling based on Supervised Peer-to-Peer (P2P) Networks*. PhD thesis, Technische Universität München, Germany, 2010.

[GRF03] M. Goyal, K. K. Ramakrishnan, and Wu-Chi Feng. Achieving faster failure detection in OSPF networks. In *IEEE International Conference on Communications (ICC)*, volume 1, pages 296–300, 2003. `doi:10.1109/ICC.2003.1204188`.

[GZCF06] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-Based Geolocation of Internet Hosts. *IEEE/ACM Trans. Netw.*, 14(6):1219–1232, 2006.

[HH] Dirk Haage and Ralph Holz. Unisono (project homepage). `https://projects.net.in.tum.de/projects/unisono`.

[HH09] Ralph Holz and Dirk Haage. CLIO/UNISONO: practical distributed and overlay-wide network measurement (extended abstract). In *4th GI/ITG KuVS Workshop on The Future Internet and 2nd Workshop on Economic Traffic Management (ETM)*, Zürich, Switzerland, November 2009.

[HHNL09] Dirk Haage, Ralph Holz, Heiko Niedermayer, and Pavel Laskov. Clio – a cross-layer information service for overlay network optimization. In *Kommunikation in Verteilten Systemen (KiVS) 2009*, Kassel, Germany, March 2009.

[Inf81] Information Sciences Institute, University of Southern California. RFC 791, September 1981.

[KBJA+06] E. Katz-Bassett, J. John, T. Anderson, A. Krishnamurthy, Y. Chawathe, and D. Wetherall. Towards IP Geolocation Using Delay and Topology Measurements. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 71–84, New York, NY, USA, 2006. ACM.

[nto] ntop.org. N2N: a layer two peer-to-peer VPN (project homepage). `http://www.ntop.org/n2n/`.

[RD+] A. Rowstron, Peter Druschel, et al. Pastry (project homepage). `http://freepastry.org/`.

[RIP] RIPE NCC. YouTube hijacking: A RIPE NCC RIS case study. `http://www.ripe.net/news/study-youtube-hijacking.html`.

[RNS09] J.P. Rohrer, R. Naidu, and J.P.G. Sterbenz. Multipath at the transport layer: An end-to-end resilience mechanism. In *International Conference on Ultra Modern Telecommunications Workshops (ICUMT)*, pages 1–7, October 2009. `doi:10.1109/ICUMT.2009.5345602`.

[SZP+09] D. Sontag, Y. Zhang, A. Phanishayee, D. G. Andersen, and D. Karger. Scaling all-pairs overlay routing. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 145–156. ACM, 2009.

[tin] Tinc VPN (project homepage). `http://tinc-vpn.org/`.

[Tor] Tor. The onion router. `http://www.torproject.org/`.

[TSGR04] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of hot-potato routing in IP networks. *SIGMETRICS Perform. Eval. Rev.*, 32(1):307–319, 2004. `doi:http://doi.acm.org/10.1145/1012888.1005723`.

[Wik] Wikipedia. Pastry (DHT). `http://en.wikipedia.org/w/index.php?title=Pastry_%28DHT%29&oldid=379423745`, reviewed on 06-Feb-2011.