
# PSIRP
# Publish-Subscribe Internet Routing Paradigm
# FP7-INFSO-IST-216173

# DELIVERABLE D2.2

## Conceptual Architecture of PSIRP Including Subcomponent Descriptions

| | |
|---|---|
| Title of Contract | Publish-Subscribe Internet Routing Paradigm |
| Acronym | PSIRP |
| Contract Number | FP7-INFSO-IST 216173 |
| Start date of the project | 1.1.2008 |
| Duration | 30 months, until 30.6.2010 |
| Document Title: | Conceptual Architecture of PSIRP Including Subcomponent Descriptions |
| Date of preparation | 8.8.2008 |
| Authors | Mark Ain (TKK-HIIT) (editor), Sasu Tarkoma (TKK-HIIT) (editor), Dirk Trossen (BT) (editor), Pekka Nikander (LMF) (editor), Trevor Burbridge (BT), András Zahemszky (ETH), Jarno Rajahalme (NSNF), Dmitrij Lagutin (TKK-HIIT), Mikko Särelä (LMF), Janne Riihijärvi (RWTH), Teemu Rinta-aho (LMF) |
| Responsible of the deliverable | Sasu Tarkoma<br>Phone: +358 50 384 1517<br>Fax: +358 9 694 9768<br>Email: sasu.tarkoma@hiit.fi |
| Reviewed by: | Dirk Trossen (BT), George Xylomenos (AUEB), Jarno Rajahalme (NSNF), Pekka Nikander (LMF), Trevor Burbridge (BT), Janne Riihijarvi (RWTH), Petri Mähönen (RWTH) |
| Target Dissemination Level: | Public |
| Status of the Document: | Completed |
| Version | 1.1 (revised 8.8.2008) |
| Document location | http://www.psirp.org/publications/ |
| Project web site | http://www.psirp.org/ |


# Table of Contents

# Executive Summary

This document presents the conceptual architecture developed within the Publish-Subscribe Internetworking Routing Paradigm (PSIRP) FP7 project. The ambition of the project is to investigate major changes to the internetworking layer of the current Internet, potentially leading to its replacement by some new scheme based on information-centric publish/subscribe networking.

The architectural work within PSIRP is a design exercise that combines bottom-up experiences with top-down rationalization approaches, and with this, establishing a design methodology that is to develop, implement, and evaluate design choices for fulfilling our project ambition. Following this design methodology, this report leads us to develop the architectural principles that are fundamental for our design and its implementation choices. These principles allow us to formulate the major concepts that are driving our overall design:

- The notion of *data & metadata*, establishing an information-centric view of our architecture.
- The notion of *scopes* as a way to delimit reachability of information, therefore building information networks.
- The *identifier* concepts surrounding the information-centric view and allowing for building the desired information networks.
- The concept of an underlying *publish-subscribe communication model* that supplies to the receiver only the information demanded by it, such provisioning being largely controlled by the information receiver.

Based on these concepts, we formulate our conceptual architecture and its major components, consisting of the network protocol machinery structure within the nodes (“stack” design), the network architecture, and the service model. The first defines how network components interact in a layer-less fashion and the second defines how the distributed network is realized using the network protocol components. Finally, the service model outlines the interfaces between applications and network services and the network itself.

We further present initial design proposals that may potentially implement the conceptual ideas that we have developed so far. These design proposals reflect the bottom up part of our design process and are therefore instrumental in guiding our design. Furthermore, we outline application considerations, representing the view that application developers will have on our solution. Important issues surround the construction of overlay structures on top of our solution and the outline of potential applications, including currently known applications, such as e-mail, and their implementation in a PSIRP world.

It is important to note that this document represents the first iteration in our project macro-cycle of design, implementation and verification. Given this, the conceptual ideas as well as the resulting architecture are likely to evolve both through debate and implementation, but in particular through evaluation of the concepts. Therefore, clearly formulated revisions will follow this document, and this report should be viewed as the beginning of a sequence of continual refinements to the PSIRP conceptual architecture.

# 1 Introduction

This document presents a conceptual internetworking architecture developed within the Publish-Subscribe Internet Routing Paradigm (PSIRP) project [PSI2008a]. The PSIRP project is an EU FP7-funded project with a 30 month lifetime. Its ambition is to investigate major changes to the network layer of the current Internet, leading to the replacement of this and other low-level layers for the purpose of adopting a new form of internetworking.

For almost 30 years, the Internet has been coping with ever increasing traffic and new applications, including voice and video, while retaining its original architecture, drafted almost 40 years ago. The current dominant internetworking solution, the Internet Protocol suite, works reasonably well for most existing demands but suffers from a number of limitations. The most notable of the design aspects that have turned detrimental is the imbalance of powers in favour of the sender of information, who is overly trusted. The network accepts anything that the sender wants to send and will make a best effort to deliver it to the receiver. This has led to increasing problems with unsolicited traffic (e.g. spam e-mail) and distributed denial of service (DDoS) attacks, forcing companies and users to conceal their e-mail addresses and place their systems behind firewalls. Further challenges include those related to efficient support for mobility, efficient global multicast, and multi-homing. In addition, reconciliation of end-to-end reachability with other networking requirements that arise from the scarcity of IP addresses and an untrustworthy environment (e.g. firewalls, network address translation (NAT), and other middleboxing techniques), is a much studied, albeit hard to solve problem.

In the PSIRP project, we see the main reason for the shortcomings of current IP-based internetworking being deeper embedded in its underlying communication paradigm than in its operational shortcomings. The endpoint-centric communication paradigm that underpins the current Internet (and its predecessor, the telephony network) places rather arbitrary topological constraints on the delivery of *information*. With the observed increase of information-centric services, such as the World-Wide Web or newer contemporary applications such as sensor networks, the inflexibility of endpoint-oriented topologies increasingly places a burden on solution developers that needs circumvention by virtue of ever increasing number of overlays. This leads to a lack of flexibility and increasing rigidity.

The worst consequence is that the full range of possibilities offered by the Internet is not being exploited and trust in its proper operation has been lost.

Experts all over the world are beginning to agree that a fundamental reform of the Internet's paradigms and core technologies is needed to cope with the challenges presented in the new millennium. The PSIRP project aims towards resolving these issues with the investigation of a new information-centric communication paradigm. For this, we consider any communication scenario as being constituted of the production, retrieval, and consumption of information, all of which are surrounded by concerns of the different parties involved in a particular situation. The notion of *intention* becomes crucial in the establishment of communications between two parties, where the match of intentions, and not the reachability of endpoints, is the key to successful communication. With that, our approach moves the balance away from the currently sender-driven IP model and towards a receiver-controlled "publish-subscribe" (pub/sub) like methodology.

In such receiver-controlled pub/sub networking, senders "publish" what they want to send and receivers "subscribe" to the publications that they want to receive. In principle, no one receives any material to which they have not explicitly expressed an interest by way of subscription. The result is a powerful yet flexible infrastructure with a high degree of resiliency.

One can observe that a large share of the Internet's usage is already essentially pub/sub in nature:

- Dissemination of software updates
- Delivery of breaking news announcements
- General media broadcasting (e.g. audio/video)
- Periodic and aperiodic messaging

…and many more!

In addition, contemporary areas of research such as sensor networks and context awareness rely on pub/sub communications to provide services to end users.

It seems promising to derive a new Internet architecture based on such an information-centric pub/sub paradigm, leading to a redesign of all Internet communication layers. In such a new Internet, multicast and caching will be the norm and security and mobility will be designed into the architecture, rather than added as after-thoughts.

Placing information in the centre of our design considerations has many impacts on the architecture we present in this document. For instance, the support for multicast and anycast primitives for data delivery becomes a native mode of operation in an information-centric world. In addition, many traditional problems, such as mobility and multihoming, become easier to solve, some even trivial. With clients indicating their willingness to receive data by subscribing to it, it is the aim of the network to act as a substrate for the ensuing data delivery process from distributed data sources. However, the scalability of data and interest-oriented networking is still a major challenge. The conceptual architecture presented in this report is the first step towards addressing some of these challenges.

This report presents the considerations for major parts of our design, ranging from the basic communication model over the build-up of information hierarchies to the considerations surrounding the reachability of information. Its structure is laid out as follows. First, in Section 2, we present the general methodology for the architecture work in the project. Then, in Section 3, we outline our vision of a future Internet. Section 4 summarizes the goals of the conceptual architecture and the constraints that have to be taken into account. Section 5 presents the set of foundational principles for the architecture with subsequent sections building on these principles. Section 6 examines design considerations and patterns that are important for our architecture, prior to presenting the conceptual architecture in Section 7. The high-level architecture is divided into components, which are considered in more detail in Section 8. Two concrete proposals to implement our conceptual architecture are then presented in Section 9. Lastly, application considerations are overviewed in Section 10, and concluding remarks are presented in Section 11.

# 2 Methodology

PSIRP is an architecture design project and therefore establishes a highly structured design process that will allow for achieving our ambitious mission. Our approach involves both ‘’evolutionary’’ and ‘’revolutionary’’ design goals, complemented by a combination of bottom-up work and top-down rationalization.

### The Role of "Clean Slate"

Often, the term *clean slate* is used to describe what PSIRP intends to do. We do not intend to focus on whether or not PSIRP is indeed clean slate. The focus of our view on clean slate is that of undergoing a rigorous design process, leading us towards a set of possible solutions for laying the ground for the Future Internet. That view steers us towards the direction of formulating this design process before continuing in particular directions of design choices. This section intends to formulate this process.

### Combining Bottom-Up and Top-Down

The major challenge in PSIRP is to create the right combination of *bottom-up ideas* and *top-down rationalization* (which can be seen as forming a somewhat rigorous design process). We do recognize that the bottom-up approach is of utmost importance to push the limits within the goals of our project. Different ideas are taken, tested, and integrated, and some are also discarded. The team structure of PSIRP encourages this bottom-up approach and it is in particular expected that design choices are pushed forward in this manner.

We also recognize the importance of rationalizing our work in an attempt to clearly formulate our ideas against a sensible, well-founded design process. This is regarded as a helpful basis by which to carve out the essence in our propositions, steer towards areas of necessary investigation, and also document the rationale by which certain decisions and choices were made.

This view on our approach, as a combination of bottom-up work and top-down rationalization, leads to the following methodology for our architecture design, illustrated in Figure 2.1.

Our intention of following a clean slate design approach can be recognized by the formulation of a clear ‘’vision’’ (one which is larger than the scope of the project) and the definition of ‘’goals’’ for our design. We then follow suit by clearly outlining the ‘’principles’’ for our design from which we derive ‘’design considerations’’ and ‘’patterns’’ to be seen throughout the overall system. This will lead us to a conceptual architecture and its ‘’components’’, to be mapped onto (several) ‘’design choices’’ to be implemented and evaluated against the particular requirements of the choices, the requirements being derived from the governing principles of our design.

While this approach seems rather top-down and sequential at first sight, one can recognize the intention to permanently adapt the design process by virtue of allowing any step of the design to be questioned, leading to refinements at any level. In particular, issues such as *scalability* – often hard to grasp as a top-down design constraint – is seen to be driven by this bottom-up experience when it comes to the particular fields that require attention in this regard. This questioning indicates the integration of the bottom-up approach that will guide our daily work on the particular choices we make.

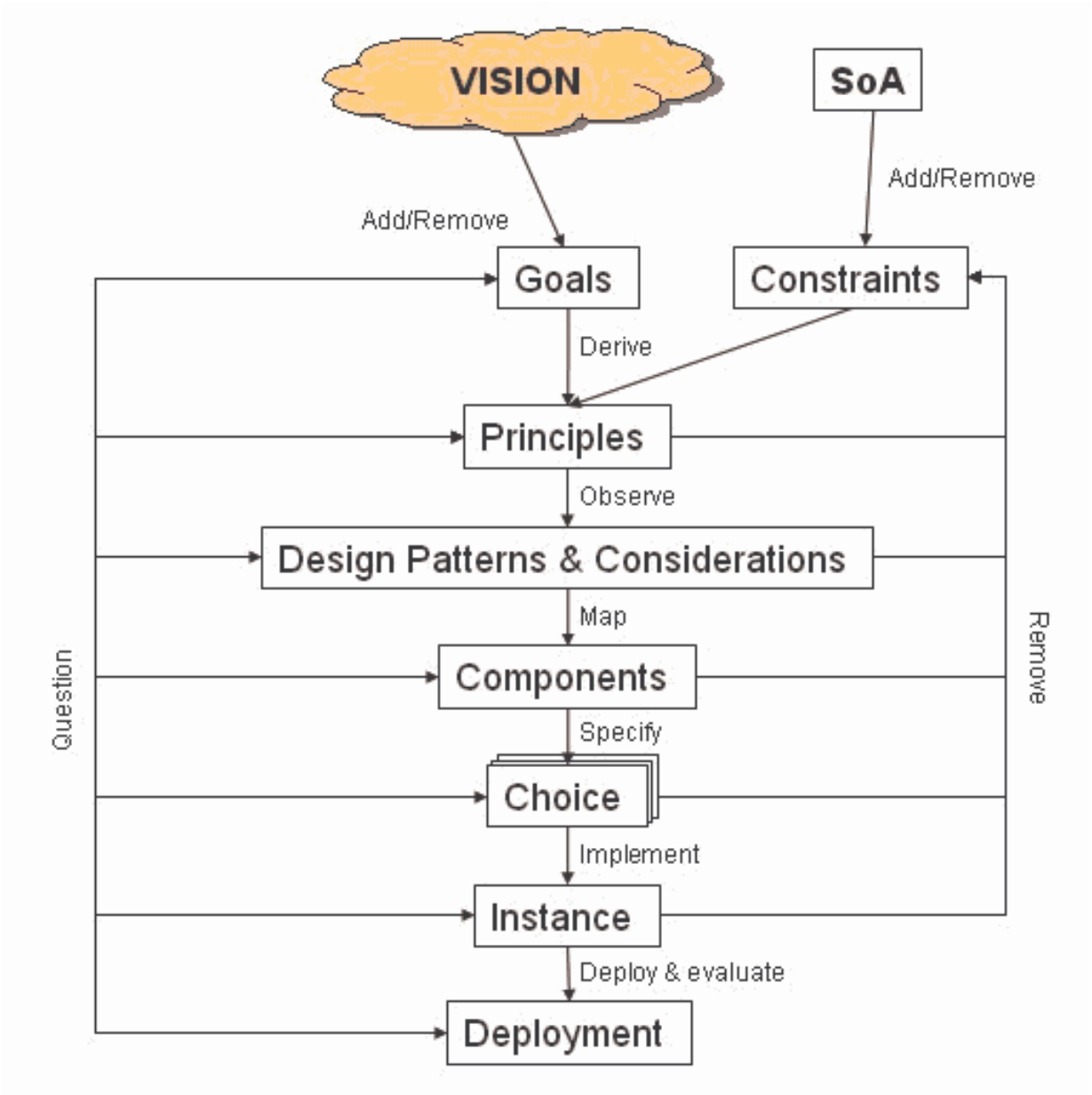


**Figure 2.1 – System design process**

It is worthwhile noting that one might not recognize a typical top-down *requirements engineering* process in this approach. This is intentional since strict requirements at the very beginning of our design are seen as being too restrictive. Instead, we base our design on a visionary approach with design principles that will allow for different design choices to emerge, all implementing our vision and principles but also potentially following very different detailed approaches in achieving them. This is complemented by our lessons learned from the implementations (the combination of top-down and bottom-up), leading to a more evolutionary and holistic approach where specifications of components are integrated rather late on a component level while the overall systemic view is preserved, guided by rough design principles as well as questioned, and therefore evolved by our learning and progress.

This very schematic view of our design methodology can be recognized in the structure of this document, which intends to cover the first six steps of this process, i.e., starting from the vision towards a first set of choices, with the content for these sections largely derived from the bottom-up work we performed so far.

### Life Cycle Within our Methodology

While the above methodology describes micro-cycles of development throughout the design process for an architectural solution, PSIRP also implements a macro cycle, as outlined in Figure 2.2.

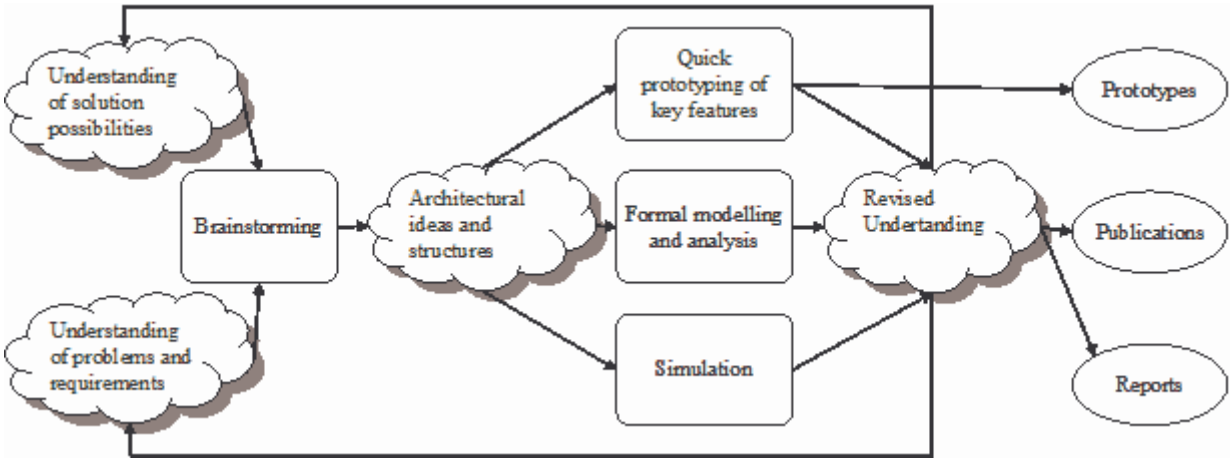


**Figure 2.2 – (Macro) life cycle of our design process**

It can be seen that we follow an iterative approach from brainstorming (around principles, design considerations, and choices) over the identification of structures and patterns to rapid prototyping and evaluation of ideas, leading to a refinement of the original design. Hence, this process implements a single macro life cycle (and feedback loop) for our design methodology in Figure 2.1 and will effectively lead us to a collection of refinements and choices throughout the project's lifetime. This macro life cycle has been the basis for our project planning and is therefore the foundation of our project structure, reflected in the team structures and the timing of the deliverables.

# 3 Vision

Our starting point is end-user centric. From the PSIRP viewpoint, any particular communication scenario is essentially a form of ‘’production, dissemination, and/or consumption of information’’. This communication is immersed within the users’ and society’s ‘’concerns surrounding the information’’ and ‘’tasks embedded in this information’’. These concerns include, for example:

- *Who* to share information with?
- *Where* to deliver the information?
- *What* to receive in return?
- *What* is the information used for?
- *Where* and for *how long* is the information stored?
- *How* to receive exactly what is desired and required for one’s purpose?
- *How* to choose community members or explicit business partners?

etc.

The concerns of different parties, such as individuals, communities, organizations, or societies, within a given set of communication experiences, are often in conflict (e.g. lawful interception vs. privacy). This leads to conflicts (a.k.a. ***tussles***) between the parties involved.

Our desire also involves considering many of these questions deeply from a technical point of view. For example, how to define the “who” (to share with) or “where” (to deliver) in technical terms? In today’s Internet, the IP address and associated domain name or extension thereof such as a URL or e-mail address seem like the only real means of identifying both actors (who) and locations (where). In future networks, such “network-oriented” mechanisms seem inadequate if we want to properly support intelligence and intentionality. It seems desirable to design future networking solutions so as to better reflect the intentions of their users, and not the other way around, as is the case today.

### From Designing Systems for Tussle...

Tussles already exist in today’s end-point oriented communication paradigm, as exemplified by both voice telephony and the Internet. For instance, emergency call (911 in the US) regulation is in conflict with the operator’s desire to reduce costs, while lawful interception stands in conflict with individuals’ desire and rights to privacy.

We assert that the increasing push towards information networking (i.e. application scenarios centred on information dissemination, such as IPTV, Web usage, sensor networks, and many others) will increase the number of tussles due to the attached information and its surrounding tussles, such as Digital Rights Management (DRM), general security and privacy, controlled sharing of confidential information, and many more.

It can be observed that, for the resolution of tussles foreseen by designers of a system, parallel architectures and solutions are developed and deployed, inherently implementing a specific set of tussle constraints. As a consequence, in many deployments today the architecture itself reflects the tussles of the scenario the architecture was designed for (and therefore the involved parties). “Design for Tussle” by Clark et al. [Cla2002] offered a novel insight for the proper design of such systems, i.e., providing guidance as to how a system ought to be designed so as to withstand and even incorporate a wide range of tussles. In the paper, the authors (Clark, Wroclawski, Sollins, and Braden) laid out principles for architectural

design that would allow for tussles to commence within the particular architectural solution. It is therefore a rather architecture-driven view of the problem.

The principles detailed in [Cla2002] have been criticized for being held relatively informal, in that hard metrics for successful tussle design were missing. It was further recommended that a separation of concerns should be possible, although it is hard to specify how this can be achieved. The authors did separate the problem into design and runtime phases, i.e., resolving tussles at design time (through means of standardization and proper requirements engineering) and at runtime, although the latter was not underpinned by any particular recommendations on the architectural level. Given the Design for Tussle methodology, we can observe the following reality in system design and development:

- Today, a potential tussle in the marketplace finds its entry into the architectural solution mostly at the design phase, if even then, and is typically represented more as a set of "nuts & bolts" in a technical solution (e.g. firewalls) rather than as a conceptual solution (e.g. middlebox control).
- Tussles that become apparent after the original design are often incorporated into the architectural solution by means of evolutionary extensions to the original design using the same design means (e.g. standardization). In other words, tussles that the designers and standards-holders of an architecture fail to recognize upfront often seep into the architecture through incremental, non-standardised features or network elements, leading to gradual erosion and eventual ossification of the architecture; e.g., consider the history of Network Address Translation (NAT).

This reality results in the appearance of multiple parallel architectural deployments, each of which represents a set of tussles for the set of parties that were involved in the design of the particular architecture. An example of this is the area of VoIP deployments with parallel IETF SIP compliant deployments and 3GPP deployment, both underpinned by a highly similar conceptual architecture, yet implemented in parallel deployment solutions. The major issues raised by this reality in system design are:

- Architectural rigidities through feature interactions of the "nuts & bolts" type of extensions occur increasingly.
- Parallel architectures are costly to maintain. For instance, developing and maintaining two VoIP stacks (catering for two different markets) can be expensive for device manufacturers.
- Parallel architectures are costly to design and deploy due to the increased costs for standardization and development.
- Parallel architectures typically lead to diminished networking externalities, thereby reducing social welfare as compared to a system using a single common architecture.
- Parallel architectures are often specific to certain business cases (e.g. 3GPP) with little flexibility to changes. Should such changes occur, the architectural solution can often prove to be costly or impossible to modify for the new conditions (in [Cla2002], Clark et. al. refer to the ability of a solution to "bend" under changing conditions).

### ...to Tussle Networking

Given this state of affairs in architectural development and deployment, it seems desirable to remove the need for parallel developments and deployments as a result of tussle delimitation, moving instead towards a design of the network itself as an execution environment for tussle policies. These tussle policies express the concerns of the players within a particular tussle space (individuals, organizations, public bodies, etc.). Within such an execution model, tussles are resolved on the basis of runtime policies, avoiding the need for design and deployment of parallel architectures, i.e., the resolution of tussles moves from the currently predominant

design phase to the runtime phase. It is worth noting that a vision for a single execution environment might sound rather far fetched and unrealistic, but even a small reduction in the number of potentially emerging parallel architectures through increased runtime ability to adapt to new tussles is likely to yield some improvement over the current situation.

Our vision, as shown in Figure 3.1, assumes a basic plane for provisioning information at a large scale, delimited by the surrounding concerns of the particular communication scenario (bottom plane of our architectural vision) [Psi2008b].

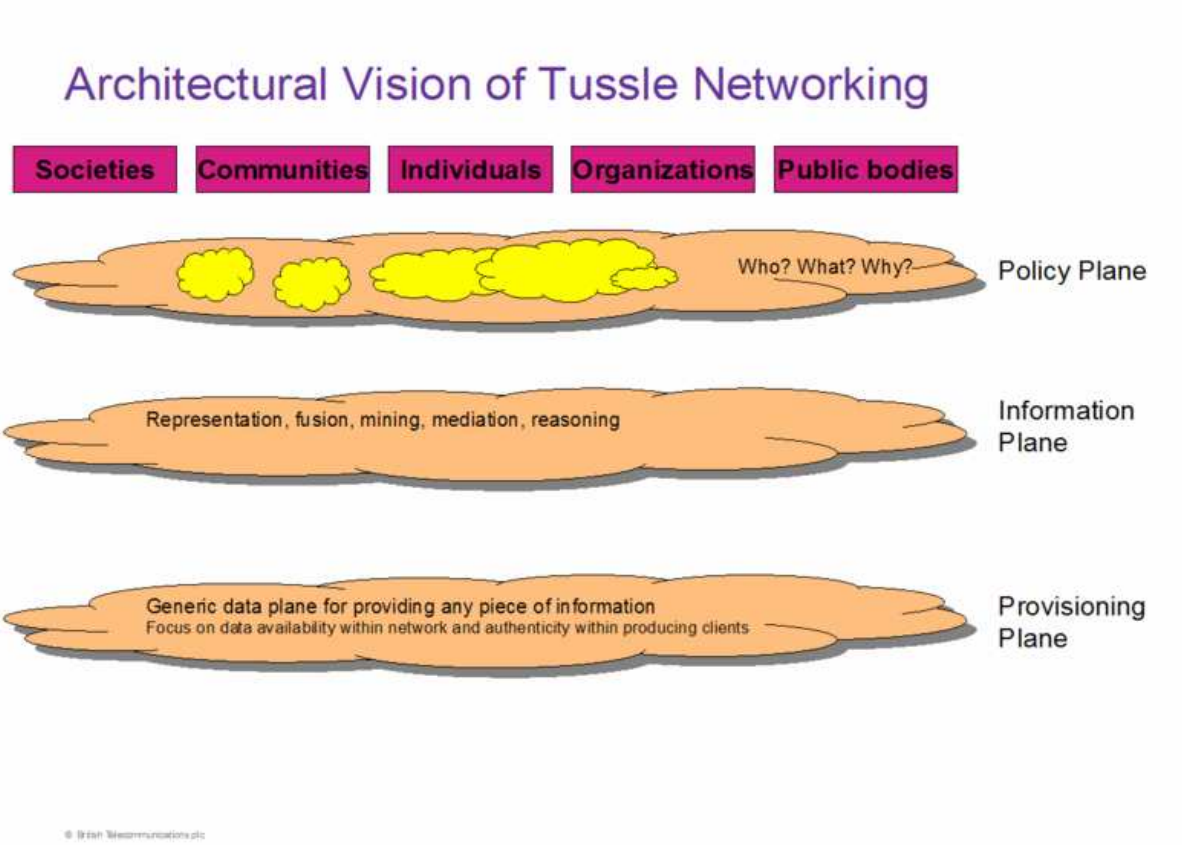


**Figure 3.1 – Architectural vision**

It further assumes another plane dedicated to information representation, mediation, reasoning and mining, as a kind of toolbox for building information systems (middle plane in Figure 3.1). The policy plane (upper plane in Figure 3.1), is envisioned as putting the communication scenario in the required context of concerns, expressed as policies that govern the scenario itself. Means for negotiating and mediating conflicting concerns are envisioned. This can lead to the isolation or partial overlap of communication scenarios, depending on the given set of concerns that governs them.

Many parts of our architectural vision have their counterparts in contemporary fields of research. Figure 3.2 illustrates how work on the Semantic Web [Ber2001] or policy runtimes, as well as in agent technology, fits into the architectural implementation of our vision. While the provisioning plane could conceivably be considered as already being in place in the form of today's (IP-based) Internet, we see a publish/subscribe internetworking based plane as more suitable to implement our vision, not least due to its more apparent information-centrism.

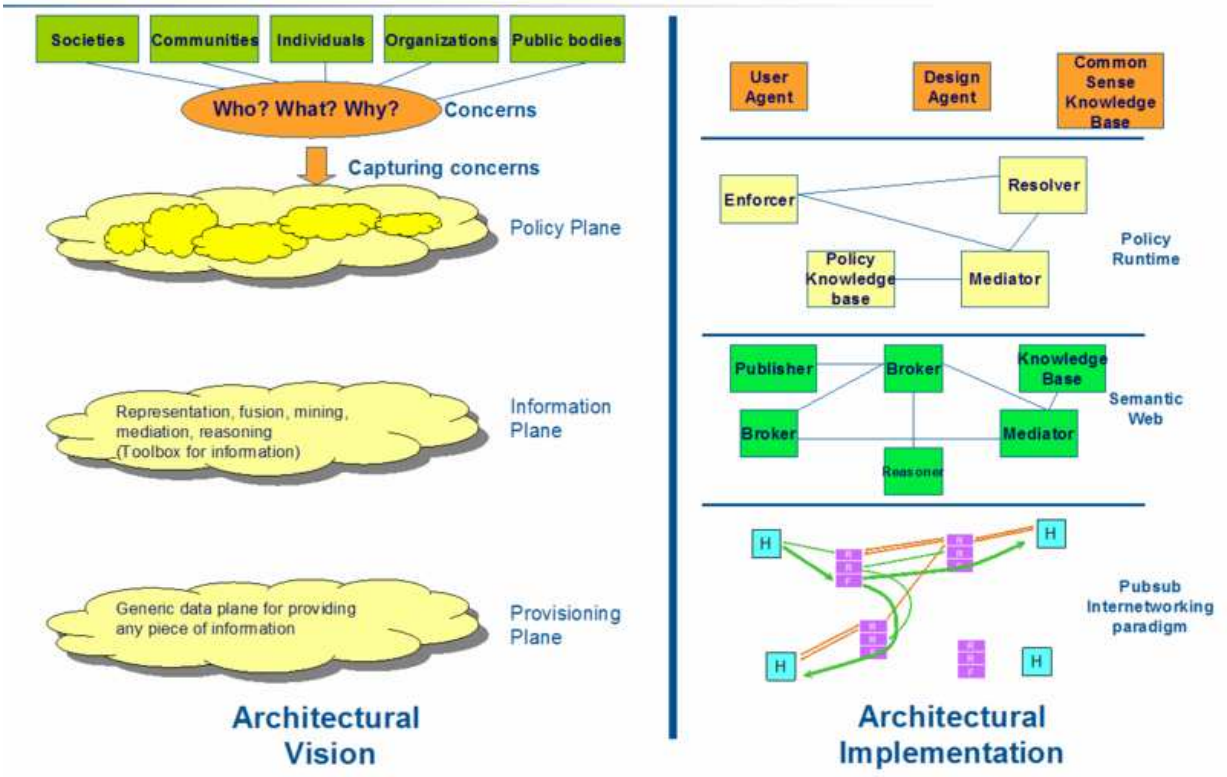


**Figure 3.2 – From vision to implementation**

# 4 Goals and Constraints

The vision laid out in Section 3, and presented in more detail in the PSIRP vision document [Psi2008b], sets the aspirations for our work in a very wide context, going far beyond the capabilities of this project. However, it is clear that our design, its principles, and the implementations themselves potentially do go beyond the lifetime of this very project. Hence, the design exercise of this project is driven by the very ambitious goals that are derived from our vision; these goals need to be scoped within the particular work items of the project, while still preserving the wider context of our vision and its far-reaching future ambitions.

Looking at our vision carefully, we can observe that it revolves around the following major concepts:

- Everything is *information*, building up from simple forms of information to very complex knowledge on application levels.
- Different forms of information *reachability* exist throughout all levels of the design, and this reachability can be changed and adapted in real-time.
- *Control* is relegated to the receiver by virtue of a communication model that allows for a choice of reception without needing to receive everything.

The PSIRP project will present a redesign for the Internet architecture from the pub/sub point of view, taking nothing (not even IP) for granted. The work focuses on the intersection of security, routing, wireless access, architecture design, and network economics, in order to design and develop efficient and effective solutions. A goal for the new pub/sub-based internetworking architecture is to restore the balance of network-economics incentives between the sender and the receiver and to be well suited to meet the challenges of future information-centric applications and use modes.

With the aforementioned concepts and high-level goals, we can define the following main objectives for our design process:

- Devise methods for building information hierarchies as a layered graph.
- Devise methods for flexible information reachability.
- Devise a basic communication model which allows for increased receiver control.

These objectives are complemented by supplementary goals that revolve around the project as a whole:

- Make “information” the centre of attention and remove the "identity-location merge" and even the need for “identity-location split” that plague current networks.
- Implement innovative multicasting and caching features to optimize performance and efficiency.
- Implement baseline security functionality as a native core component of the architecture.
- Implement and validate the new architecture under realistic operational scenarios.
- Collaborate with key industry experts and promote their ideas in the wider community through publications, presentations, workshops, and involvement with other projects at the European and international levels.
- Disseminate the results.

This formulation of our goals gives us the chance to outline a first very high-level concept architecture presented in Figure 4.1.

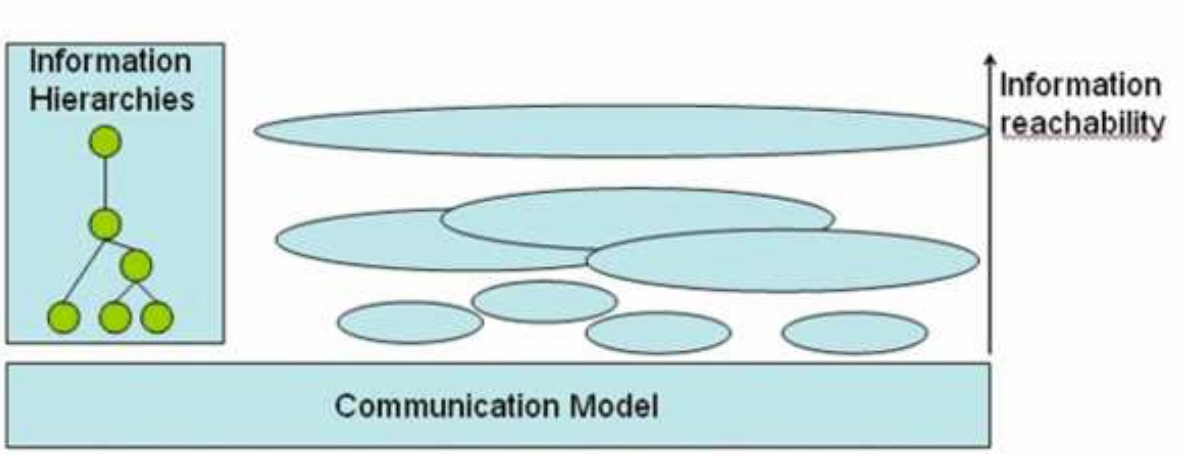


**Figure 4.1 – High-level architecture: bringing together our main concepts**

As outlined in Figure 4.1, *information hierarchies* exist on many levels as layered graphs, and the supply and demand for information varies considerably within these different hierarchies. These information hierarchies are built by virtue of different identifier concepts, which we will introduce later in Section 6.3.

*Information reachability* is about sustaining a multitude of information hierarchies whose *demand* and *supply* relations might evolve over time. We will introduce concepts for delimiting information reachability, implementing these demand/supply relations, in Section 6.2.

The supply and demand of information is satisfied by an underlying and omnipresent *communication model*. This communication model, according to our vision, is intended to shift the balance from sender to receiver control. The *publish-subscribe model*, as devised later in Section 6.4, allows for such increased receiver control by virtue of the demanding receiver of information specifically subscribing to information supplied to the communication substrate by a publisher of said information.

Following our design methodology (see Section 2), we will attempt to derive more fine-grained concepts and components that will eventually lead us from the high-level architecture in Figure 4.1 to our conceptual architecture in Section 7 and the description of the individual components in this architecture in Section 8.

### Constraints

In our design methodology of Section 2, we outlined constraints as being central to the design of our system. In our first iteration of the project lifecycle, we intend to derive potential constraints largely from our evaluation of the state-of-the-art. Constraints that might occur as a result of deployment options, scalability, early implementation experiences, and other factors, are intended to be taken into account within the second cycle of the project.

### Scope Within our Work

The scope of this project is focused, within our vision, in the provisioning plane (see Figure 3.1) and therefore concentrates our efforts on aspects that relate to the more traditional "routing and forwarding fabric", as understood in light of the vision. The current model of IP, used in the Internet, establishes a routing fabric centred on a topological notion of the network, in which packets are delivered end-to-end between explicitly named endpoints. Changes in this topology, as well as changes in forwarding behaviour, are difficult, albeit not impossible, to incur and ascertain (e.g. firewalls). This leads to inflexibility not only on

technical but also on business (e.g. inter-provider peering) levels. Hence, the ability of the present architecture to adapt to changing needs on the technical, social, and business levels is limited and usually achieved with a more evolutionary extension model rather than with a fundamental conceptual model for a revolutionary adaptive network.

**We aspire to change the routing and forwarding fabric of the global inter-network so as to potentially operate entirely based on the notion of information (associated with a notion of *labels* to support fabric operation) and its surrounding concerns, explicitly defining the scope of the information.** While we do not directly embed the higher level semantics of information into the network, we intend to devise means that will enable the higher levels to embed concerns and social structures surrounding this information deeply within the architecture. This will be reflected by selected work items in our project (e.g. forwarding and rendezvous).

It is worthwhile pointing out that the envisioned operation on information, as formulated above, indicates an important move away from the current endpoint-centric (data) networking model. While such a model, as embedded in IP networking, enables the transmission of a stream of data between two explicitly addressed end-points (with total transparency as to the information represented in this data and the nature of the communication surrounding this exchange of data), the model envisioned in our aspiration emphasizes information-centric operation in the sense that data pieces are explicitly addressed and therefore directly embedded into the network. This operating methodology is obviously disparate as compared to the endpoint-centric operation observed in current IP and IP-like networks.

This explicit addressing of data instead of identifying endpoints consuming data, can be seen as the most significant step on this level of the overall architecture of moving from a data to an information view, i.e., establishing correlations between data through explicit (data) addressing. This is aligned with, e.g., Van Jacobson's view (as formulated in [Jac2006]) on the evolution of communication from connecting wires (public telephone system) over connecting end-points (current Internet) to connecting information (our aspiration described earlier). It connects well with our above-mentioned goal of enabling information hierarchies by setting the foundation for these hierarchies on a rather low level in the overall architecture.

# 5 Principles

A **principle** signifies a point (or points) of probability on a subject (e.g., the principle of creativity) which allows for the formation of a rule, norm, or law, by (human) interpretation of the phenomena (i.e. events) that could be created with regard to that subject. Reducing a rule to its underlying principle says that, for the purpose at hand, the principle will not / cannot be questioned or further derived (unless you create new rules). Hence, a principle has an inherent degree of minimality and indivisibility when contrived as a governing characteristic underlying a rule of design.

In this section, we present the guiding principles for the PSIRP design, as currently established in the project. We first outline the process used to derive the principles, and then proceed to describe the principles that seem to apply to most if not all inter-network designs. Finally, in the third subsection, we present a number of principles that appear specific to PSIRP goals and aspirations.

## 5.1 Methodology for Deriving our Principles

The process of *pruning* (evaluation and refinement according to rules that maintain some agreed base consistency) an original set of suggested principles was very important in deriving our basic principles within PSIRP. We underwent a team exercise to formulate our base principles through the use of collaborative (Wiki) tools. The resulting initial set of principles had an expected larger number of (often) overlapping constituents and was relatively detailed. From this initial set, we derived a pruned set of principles, shown below, through the following methodology:

- An original principle could be removed or placed under another principle (i.e., secondary principle) if and only if the vision of our design (the phenomena at hand) was not destroyed by the pruning (i.e. the maintenance of consistency).

Through this, we were able to determine a minimal set of principles that describes our architecture, potentially with a set of secondary principles. For the pruning, any team member was allowed to prune one or more principles with an explanation as to why this specific pruning was possible. We preserved the initial list of principles for archival purposes but omit it from this document due to length limitations.

We divided the final set of principles into two completed groups, namely what we could call the **PSIRP principles** (principles that are rather unique to the category of architecture we intend to build) and the **general principles** (principles that apply to a wider range of architectures - almost all, in fact - whose breadth is comparable to PSIRP). The PSIRP principles are driven by the vision we formulated. Hence, dropping one or more of our final PSIRP principles would almost certainly lead to a failure to realize our vision, i.e., we intend to reduce the initial collection of principles to a set of principles that describe the crucial essence of our vision. The general principles are more related to the design process itself; i.e., they drive the particular formulation and selection of general design and implementation choices.

## 5.2 General Principles

With the above said in mind, we have formulated the following general design principles for our internetworking architecture:

- **G1** - T2T Principle: Functions are implemented at points of the network that can be considered trustworthy from the user's (of the function) perspective. For this, means to evaluate the trustworthiness of functions and their placement must exist in order to

fully enable choice based on the evaluation of trustworthiness. Effectively evaluating trustworthiness requires a proper balance of privacy and attribution, the former required to foster trust and the latter required for the just punishment of un-cooperative parties (which is necessary for fostering an environment of trust).

- **G2** - Market Creation: The architecture supports potentially multi-dimensional metrics of compensation that will enable effective market creation. Negative market externalities should be resolved by applying Coasean-like principles to assign new, artificial property rights in such a way that 1) a new market place is formed which is able to internalise the externality and 2) that the thereby released utility is divided fairly between the involved parties.
- **G3** - Multi-dimensionality: The effectiveness of any architectural solution is evaluated under a (multi-dimensional) metric along the dimensions of financial, social, environmental, and economical benefits.
- **G4** - Evolvability: Any solution should enable proper evolution beyond itself. This evolution can be considered part of migration and deployment scenarios.
- **G5** - Minimality: No entities are introduced beyond the necessity to implement the design principles of the architecture.

## 5.3 PSIRP Principles

Information is central in the internetworking architecture we intend to create. While IP decomposes an information-centric view of the world by introducing topological constraints on the delivery of information (i.e. the subnet-structure of IP networks), we intend to primarily build the system architecture around information from the viewpoints of ‘’semantics’’ (i.e. meaning) and ‘’scope’’ (i.e. breadth of coverage).

In this information centrism, two concepts are fundamental:

- the recursive nature of *information semantics*, and
- the notion of *information scope* as a concept of reachability.

The concept of recursive information semantics is expressed by basing our design on the notion of information, starting from very low level forwarding information and leading up to complex semantics with certain specific contexts. For that, we assume that ‘’flat labels’’ are used to identify communication relations between entities which are concerned with a particular class of information. Note that the particular topology for delivering this information is not our concern at the moment and will be addressed in later deliverables. We further assume that any higher layer semantics will be recursively built on top of these low-level forwarding semantics. With this, we build a bridge between higher level concepts like ontologies and the lower level information that is used to deliver these exact higher level concepts, without specifically introducing the topological constraints in reachability that we see in IP-based networking. The result is envisioned to be a flexible set of delivery mechanisms whose operation is based on segregated and hierarchical meaning within the relevant information that is involved.

Given the concept of recursive information semantics, we now turn towards the issue of reachability. Information scoping replaces the concept of network graph topologies in IP networks, i.e., it represents the (limitation of) reachability of information by virtue of its comprehensiveness. Given the outlined recursion on the semantic level, scoping information and therefore the reachability of the said information in itself is recursive. Hence, we assume that for certain semantic levels, certain mechanisms exist for scoping the information related to the given semantic level. There exists a variety of mechanisms to scope information on various levels. Hence, for example, while we can see a Generalized Multi-protocol Label

Switching (GMPLS) [Ban2001] service manager as a means to scope forwarding labels in an all-optical network, rendezvous mechanisms are usually used for scoping information at the routing level, and search or discovery mechanisms are expected to be applied on higher semantic levels. With the notion of scope defining the reachability of information, we aim for the architecture to be neutral to the semantics and structure of the data.

With these concepts of recursive information semantics and information scopes for certain semantic levels, we further assume a communication model that places the control over reception of data in the hands of the receiver (provided that the permission to receive has been granted in the first place). Hence, a model similar to publish/subscribe, in which data is only received once subscribed to it, is the foundation for our system architecture. However, the exact implementation of this communication model will vary depending on the level of semantics we are operating on.

We now take these concepts and translate them into principles (the essence for defining rules to build architectural concepts for our vision):

- **PS1** – Information is multi-hierarchically organised: Higher-level information semantics are constructed in the form of ‘’directed acyclic graphs’’ (DAGs), starting with meaningless forwarding labels and moving towards higher level concepts (e.g. ontologies), or vice versa. It is also possible to view the situation as the higher level concepts being realised in terms of lower level concepts etc., until one ends up with individual data items and semantically meaningless forwarding labels. This can be illustrated by Figure 3.2, where abstract information hierarchies are realized by the supply and demand of information over some concrete topology. Hence, an important part of the network is to be able to map high-level hierarchies to lower-level hierarchies, and ultimately to forwarding labels for actual data delivery.
- **PS2** – Information scoping: Mechanisms are needed that allow for scoping information on different levels of semantics in the architecture (e.g. via means of rendezvous, discovery, search, and others). With this, information scoping limits the reachability of information to the parties having access to the particular mechanism (e.g. rendezvous) that implements the scoping.
- **PS3** – Scoped information neutrality: Within each scope of information, data is forwarded based on the given (scoped) identifier, i.e., the architecture is neutral with regard to the semantics and structure of data.
- **PS4** – The architecture is receiver-driven: No entity shall be delivered data unless it has agreed to receive it beforehand through appropriate signalling methods (i.e., excluding physical means of receiver choice), wherever this can be implemented in the conceptual solutions for our architecture. Limitations to this principle can apply, for instance, at the physical level.

In the remainder of this document, we focus on the impact of these principles on design considerations for our conceptual architecture and the formulation of particular design choices. Although we restrict our considerations mainly to the equivalent of the internetworking (IP) layer, our principles are formulated to apply to all layers throughout the system.

# 6 Design Considerations and Patterns

In this section, we examine foundational design considerations, and both design and architectural patterns for the PSIRP architecture. As mentioned before, we aim to support information internetworking. A base assumption for our work is that information is structured into hierarchies (**PS1**). According to this principle, higher-level relations between objects and entities translate into lower-level relations and ultimately to forwarding labels. In the following, we first outline the concept of data and metadata before introducing our approach to information reachability. We then present the different identifiers in our system, how they relate to each other and how they resolve, before considering the pub/sub communication model as the base paradigm for exchanging information in a PSIRP system. Finally, in the end of this section, we briefly consider the role of network coding within the architecture.

## 6.1 Data and Metadata

The information-centric view of our architecture places **data** in the centre of attention. An information set, represented at the network level by a specific rendezvous identifier (see Section 6.3), may relate to one or more discrete items of data at the application level. In other words, we can envision individually addressed data items to be delivered throughout our network rather than individually addressing the end-points dealing with the data items. Each data item represents a piece of information that is related to some application level task, such as the transport of a particular figure or a piece of mail.

Each piece of information on the application level, however, often relates to other information. For instance, information on picture size, date of picture taken or others, relate to the actual picture data at hand. Such data is often referred to as **metadata**, although such data merely represents itself as plain data to the network level. This concept of metadata is crucial for our architecture and is used for application level implementation of certain tasks as well as on network level to simplify management and core functions of our architecture.

On application level, data, such as our figure example, is represented and individually addressable within our architecture to enable delivery of this data item. Application-level metadata that relates to this data is, again, represented as individually addressable data on network level. In other words, any item of information may be considered as metadata for other items of information. All items of information are directly referred to by their individual identifiers or resolvable by means known to the application. Such mechanisms may be used, for example, by a content-delivery application to indicate sub-fragments of the content, or to indicate a reply communication channel in a client-server relationship. There are no limits to what such metadata–data relationships may mean within the application scope.

We also consider metadata that is understood within the network itself. Such network-level metadata might be concerned with how the communication for a particular data item may be conducted. This network metadata may be found as (soft) state within the network, carried as part of the communication header, or referred to by separate identifiers.

Any network metadata will be limited by the network functions that it may control. Such functions may include access control, flow control, error notification, congestion notification, etc. We will introduce different usages for this metadata concept in the following sections, underlining the importance of this issue for our design.

## 6.2 Concept of Scope

The PSIRP principle PS2 outlines the existence of a **scoping** mechanism as to *limit the reachability of information to the parties having access to the particular mechanism (e.g.*

*rendezvous) that implements the scoping* (see Section 5.3). In this section, we outline the concept of scope more closely from the PSIRP architecture point of view, focussing on the role of the network in implementing the concept. We should also note that application functions, above the network, also perform scoping according to our definition, such as the restriction of records retrieved from a directory service. We deliberately leave out these higher levels mechanisms within this discussion to focus on the scope of dissemination of information across the network.

As PS2 describes, scoping mechanisms are implemented to limit the reachability of information with respect to a particular set of senders and receivers of a particular piece of information. Therefore, scoping can be seen to represent a logical equivalent to the concept of *topologies* (such as link-local or site-local) in the endpoint-centric IP world. Given the information-centrism of our architecture, however, a **scope** (identified through a specific *scope identifier*, see Section 6.3) is naturally attached to EVERY item of information that is fed into the network (although we can consider the case of 'no scope' being attached to a data item as being equivalent to the notion of 'localhost' in the IP world - in other words, the information would not leave the local computer). In other words, scoping allows for building *information inter-networks* as opposed to topological inter-networks.

We can consider that the scope of a piece of information is controlled by several mechanisms. The publisher of a specific piece of information must place the piece within an explicitly identified scope during the publication operation. The rendezvous identifiers themselves can be used to implement simplistic scoping by restricting their distribution to specific parties. Furthermore, since no information will flow without subscriptions, the subscriptions may also be considered to define a (temporally dynamic) scope of information. Security permissions and routing policies, associated with a rendezvous identifier, may also be combined with publication and subscription operations to determine the final information dissemination.

However, here we introduce the notion of a separately defined scope that acts in combination with the rendezvous identifier during the publication and subscription operations to determine the potentiality of flow of information across the network. There are a number of reasons for maintaining a separately controlled scope for each publication:

1. To allow the publishers and subscribers to select a scope and to migrate information between scopes such that interests are grouped together. This provides a key benefit to the network as subscriptions may be aggregated into more scalable scopes, particularly for inter-domain management.
2. Allow aggregate operations on the scope, such as access control and other metadata controlled operations; e.g. defining a scope-level caching policy. This can be used by the applications to manage sets of information in a scalable fashion.
3. Allow separate publications to have shared control of the dissemination through the scoping mechanism, yet allowing each publication to be separately authenticated. For example, multiple publishers all publishing to a single scope, can still be separately identified.

A certain piece of information, however, is not restricted to a single scope. Instead, certain information can *reside* within different scopes, as also expressed in our use cases [PSI2008b], e.g., a particular video can reside in a user's personal scope AND the user's *friends* scope, both scopes defined (in this case) by the end user.

Scopes, and therefore the reachability of information within said scopes, are governed separately with respect to, for instance, set of senders and receivers allowed to operate with each scope, the admission of new senders/receivers (i.e., access control), and other issues. Such governance rules can be interpreted as *metadata* associated with the scope.

There may also be advantages in allowing multiple levels of scopes. This may provide further benefits for the management and scalability of the information dissemination across the

network. In this approach, scopes may be included within further scopes and migrated between parent scopes.

Scopes define a powerful concept that can construct social relations between entities, representing consumers and providers of information, and the information itself. This is illustrated in Figure 6.1, where certain information (e.g., a picture) is available to family and friends, while other information is merely visible to colleagues. Each scope is attached with a governance policy, represented as metadata, which for example includes authentication information for potential receivers of the information.

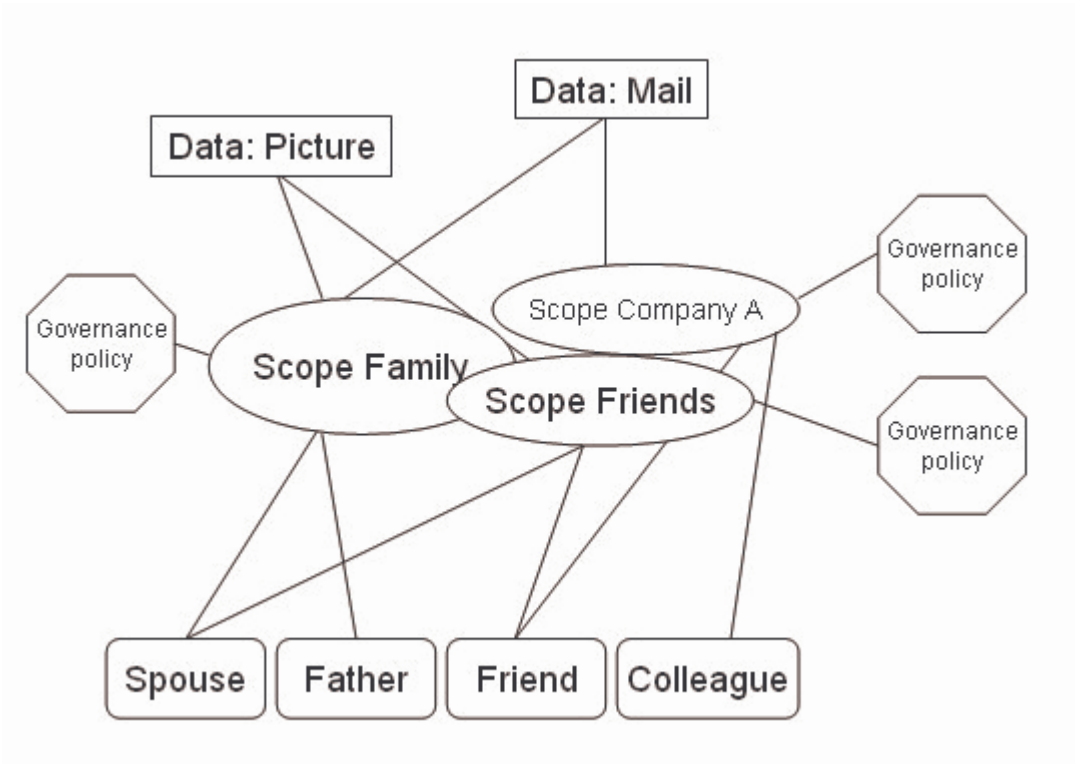


**Figure 6.1 – Concept of scope**

Scopes can easily be re-constructed, removing certain parties from the scope (through changing the governance policy), adding new pieces of information to the scope (through publication within that scope) and information can easily be assigned to new scope (through publication within the new scope). This ability works towards our stated ambition to enable networks to reflect social structures and information grouping in a dynamic way.

It can be seen that the concept of scoping enables a dynamic assembly of providers and consumers of information in a way independent from the actual forwarding of said information in the network. It is important to note, however, that neither the senders nor the receivers possess any knowledge of any restricted peering agreements between network operators. Peering and routing policies between network operators should not restrict the visibility of information that is sent or received; instead, they can be used by operators to differentiate economic and quality-of-service conditions.

## 6.3 Identifiers

We can view the global network of information as an acyclic *graph* of related pieces of data, each identified by some identifiers. In the PSIRP architecture, identifiers define the relationships between the pieces of information in our system on the different levels, such as

the application or networking level. In other words, we assume that any relationship between two pieces of information is realised by (at least) one piece of information containing some identifier, referring to the other pieces of information.

Figure 6.2, below, illustrates the different identifier spaces in our architecture. From the very high application level, we see different data items, embedded in their application context, mapped onto the PSIRP architecture on the rendezvous level, before finally being mapped onto the forwarding level for delivery to the end systems. For simplicity reasons, we have left out issues such as data segmentation (which might result in several publications – depending on the application semantics) and other application-specific issues.

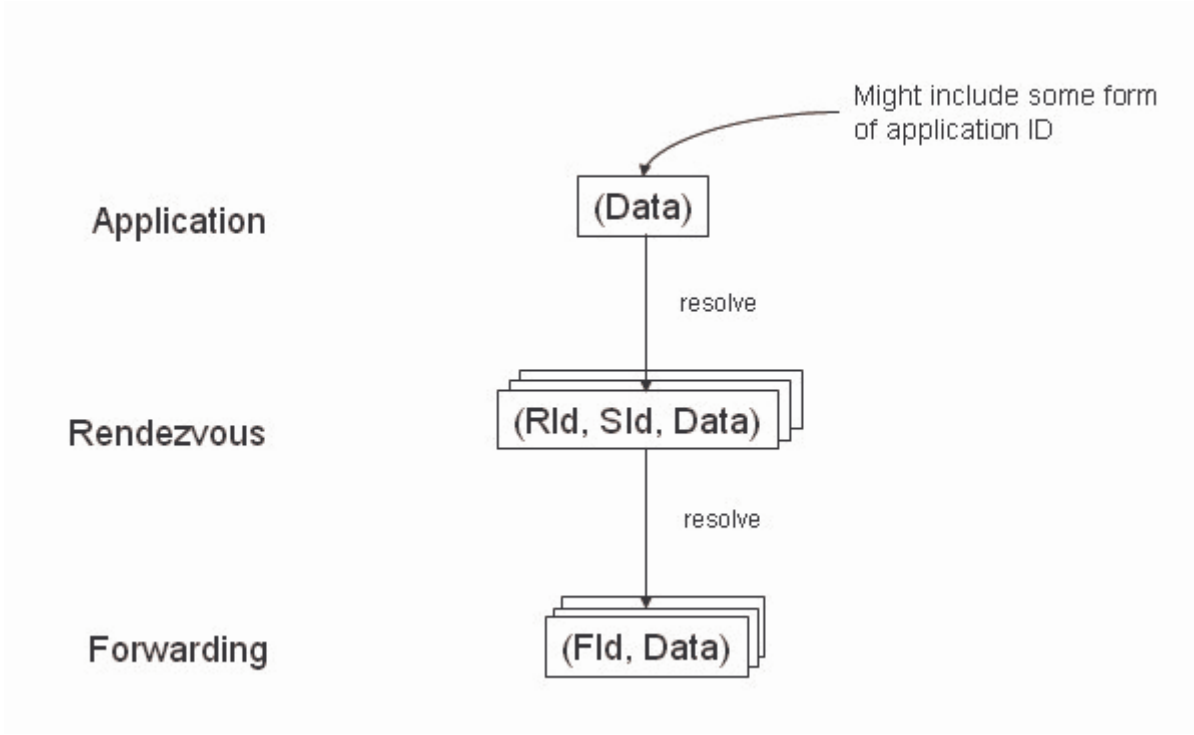


**Figure 6.2 – Identifier structures**

In the following, we examine the meaning and usage of these identifiers. From this, we suggest potential horizontal layers within the network and comment on the required resolution services (Section 6.3.2) that are required to provide links between identifiers residing in different layers.

With this in mind, we propose the following classes of identifiers:

- **Application identifiers**, used by publishers and subscribers.
- **Rendezvous identifiers**, used to bridge higher level identifiers with lower layer identifiers.
- **Scope identifiers**, used to delimit reachability of given information.
- **Forwarding identifiers**, used to transport publications across networks.

### Application Identifiers

The concept of an identifier can be applied to many different areas within the context of applications. Traditionally, identifiers have been used to categorize, distinguish, and track devices (domain names), products (product/serial barcodes), people (employee numbers, driving license numbers, full names), content (film titles, web page addresses), locations (grid

references, building numbers/addresses), services (web service addresses, phone numbers, email addresses), and many other items.

There are several interesting points to note from the above examples. Firstly, identifiers are not required to be either universal or unique. An identifier is only meaningful given some knowledge of its semantics and how to apply it usefully within a process. The same identifier may mean different things to different applications, and be applied within wholly different processes. For example, a company employee named “John Smith” and the “John Smith” identified within the UK Inland Revenue system may be entirely different people. The identifier “John Smith” is certainly used within different processes.

The second point to note with these application identifiers is that many of them have a meaningful internal structure. Indeed, there is no hard boundary between an identifier, a structured name, or richer structured content (such as a postal address or a search query). An identifier (or name, or address, or …) is merely some item of information that is applied within the context of a specific process to establish a relationship to “something”. We observe that the *context* in which an identifier is created and used determines the interpretation of the identifier to any entity that processes it.

We consider that application identifiers and other descriptors used by applications are defined using application specific semantics and contexts, and they are not as such understood by the network. Therefore, a separate class of identifiers, resolved from application identifiers and understood/controlled by the network, is needed.

### Rendezvous Identifiers (RId)

The network in itself is only concerned with routing and forwarding data; the network should be agnostic towards application identifiers. They are not suitable for direct use in the network because they are not unique and lack universal properties (such as the ability to authenticate senders). In addition, their lack of a common length or structure would cause obvious difficulties building suitable routing tables, and the non-random nature of the application identifiers could lead to hot-spots within a rendezvous system. Furthermore, the decoupling of application and network semantics allows the same network identifier to be used by multiple applications (for example, if an identifier is used for all traffic to a user group), although such multiple use of rendezvous identifiers needs to be properly considered in any application scenario in order to avoid ambiguity.

To be able to connect applications over the network, we rely on a special class of identifiers, called *rendezvous identifiers*, which are used by the network to connect different application identifiers. When an application joins or sends to a rendezvous identifier, it should be clear what information it has elected to receive, or what distribution group it is sending to. This is easier if the identifier is unique. The quality of uniqueness must be balanced against a desire to obtain an even spread of identifiers across the identifier space (for scalable rendezvous) and the ability to dynamically self-generate such identifiers.

Such rendezvous identifiers can be created by any application and shared (directly or indirectly) with other applications who the application wishes to communicate with. Hence, rendezvous identifiers may be originally created by a sender or a receiver of network data, and indeed their role may change over their lifetime within the network.

In addition to routing, rendezvous identifiers may be used by several other network operations. Such operations could include network authentication and access control of senders or access control of receivers who are allowed to join a given rendezvous identifier.

Rendezvous identifiers are meaningful within both an application context and a network context. To the applications, they represent a communication channel through which other applications can be reached. The application understands the expected application audience

(e.g. the voice-over-IP service for John Smith). The network uses the same identifier to establish communications between all senders and receivers that are joined to the identifier.

### Scope Identifiers (SId)

Rendezvous identifiers enable the network to route and forward requests appropriately, e.g., forwarding a subscription request to a publisher of data matching the given rendezvous identifier. In order to delimit the reachability of information, we use specific *scope identifiers* for this purpose. A scope identifier is attached to each rendezvous identifier and associated data (see Figure 6.2). A set of (RId, SId) pairs is used by the rendezvous system to determine the appropriate set of forwarding identifiers for delivery of this data to the set of receivers. It is worthwhile pointing out that such relationship from a set of (RId, SId) pairs to a particular set of forwarding identifiers (i.e., a particular forwarding tree) is often not unique, i.e., it is often likely to share common forwarding trees for a larger set of data items, each of which is identified by a particular RId within a given scope SId. One reason for such tree sharing is the ability to manage forwarding trees in a scalable manner, in particular within the core of the network. Also note that a single RId can be associated to a set of SIds, e.g., a publication of data can be associated to different scopes of reachability, as outlined in Section 6.2.

More formally, we can define the relation of RIds and SIds as follows:

Let $S$ denote the scope space of finite size and let $R$ denote the rendezvous identifier space of finite size. The identifier space $R$ is independent of the scope space $S$. There exists a mapping $M$ from each element of the scope space to the rendezvous space. This mapping is from an element of $S$ into a set containing elements of $R$. In other words, each scope contains a well-defined set of rendezvous identifiers. More formally, $M$ is a set-valued function from the set $S$ to the power set of $R$, $M$: $S \rightarrow 2^R$.

Note that we leave out the implementation detail as to how the set of scopes and the association to a particular RId is communicated within an API. Furthermore, it is left out in this section how the scopes are identified. Possibilities for this include dedicated rendezvous identifiers included in the scope set itself or the inclusion of a distinct scope identifier in some metadata information attached to the data.

### Forwarding Identifiers (FId)

Where rendezvous identifiers are universal amongst all applications using the network, each and every possible forwarding tree or tree segment within the network may also have a unique identifier. Another concept that determines the forwarding trees in use within the network is the concept of scope (see Section 6.2), expressed by a particular SId, in addition to the RId of the data item, bearing in mind that forwarding trees are likely to be shared for scalability purposes.

Each active pair of rendezvous and scope identifiers (i.e., each such pair presently in use by at least one active sender and at least one receiver) will have a mapping to at least one forwarding identifier, though such a mapping may be quite complicated, as discussed above. The network may also construct logical forwarding trees that use the same forwarding identifier over multiple hops. In this case, the network has the option for further resolving such logical routes into finer grained (and ultimately single hop) routes. Also, the network will typically perform aggregation, pruning and optimization operations for trees that are common for many (RId, SId) pairs, e.g., in cases where a set of information (expressed by a set of RIds) is shared between similar sets of senders/receivers (e.g., due to being different media streams within a single application, or being distinct items within a larger information set), typically leading to an overlap in physical delivery infrastructure. Especially in the middle of the network such overlap is expected to take place almost all the time, in particular from the perspective of achieving scalable tree management.

Forwarding identifiers are not expected to be used or understood by applications (except by network management applications).

### 6.3.1 Selection of Identifiers

For a rendezvous identifier to be useful, it must be known to at least one sender and one receiver. There are two key ways in which this can be achieved:

- either the sender acquires the identifier and informs any potential receivers, or
- the receiver selects an identifier and informs potential senders.

A third and perhaps less implicit option exists, where a third party application or service initially acquires the identifier and consequently shares it with a sender and a receiver. Where the choice is at the sender's end, we expect that the identifier will often be chosen in a manner similar to those described in the Host Identity Protocol (HIP) [Mos2008] or the Data Oriented Network Architecture (DONA) [Kop2007], i.e., the identifier is based on a public key whose corresponding private key is in the sender's possession. This allows the sending host to choose an identifier that can be used by receivers to authenticate the data payload and verify that it was in fact sent by the party that possesses the associated private key.

Identifiers may also be chosen in accordance with other allocation schemes, as will typically (but not necessarily) be the case for the "by-receiver-acquired" identifiers. We can imagine variations of the DONA approach that allow multiple senders to publish over the same identifier. This could be achieved, for example, by using a shared group key, identity-based encryption, or where such peer agreement is not possible, by using a third party to issue proxy re-signature keys.

From the content point of view, there seems to be three different cases for choosing identifiers:

1. choosing an identifier for *pre-existing* data that will remain static in the future,
2. choosing an identifier for data that does *not exist yet*, but will not be changed once it has been created, and
3. choosing an identifier for dynamic data that needs to be *versioned*, i.e., it may be updated at some time in the future.

#### Identifier Persistence

We suppose that each application instance will typically use and distribute many identifiers. Depending on what is represented by each of these identifiers, some identifiers may reference the same concept for decades, whereas others may be used for only a single transient packet. Although we desire identifiers to be uniquely selected by or allocated for applications, it may be possible re-use short-lived identifiers in certain select cases. Only the application is in a position to know the reasonable lifetime of such identifiers or the risk of re-using the identifier when it may still be in existence through other applications.

#### Multi-Sender and Multi-Receiver

Since identifiers can relate to many possible items, it does not seem sensible to restrict their use to a mere single sender and/or receiver. For example, a group of many senders may be interested in transmitting data about an item of information (for example, the sightings of migratory birds). Conversely, many receivers may also be interested in obtaining the same data. An obvious technical choice to address this situation would therefore be to allow multiple

senders and multiple receivers to use the same identifier. However, there are a couple of reasons why we might wish to restrict the technical model.

Using a single-sender multicast model allows the network "stack" to authorise the sender without requiring further analysis of the payload. This can be useful for efficiently reducing the amount of unwanted or unauthorised traffic on a per-principal basis. In addition, it may also mean that the network-level forwarding trees can be optimised for each sender without resort to an arbitrarily located rendezvous point. The downside of this approach is that each receiver that is interested in receiving related information from multiple senders must join additional network identifiers. However, this seems to be a relatively small price to pay since we expect a single application to manage many independent identifiers. We still maintain all of the benefits of multicast bandwidth sharing (and potentially more, due to optimisation of each source tree).

However, strict single-source multicast cannot be our only model, as we also want to be capable of distributing receiver-chosen identifiers to multiple potential senders (for example, for fan-in of sensor information, or contact with an application or web server). This requires a "concast" model with multiple senders and a single receiver. Note that, unlike multicast, there is no bandwidth benefit in providing a concast model over multiple unicast channels. However, it seems to provide a significant identifier management benefit to the receiver applications and reduces the overall number of identifiers. The trade-off is that sender authentication must be performed by examination of the payload and the receiver must coordinate the senders' generation of signatures so that they are valid over the chosen identifier (using group keys, identity-based cryptography, or re-signature schemes). Otherwise, the potential problem of unauthorised senders must be solved through some other, perhaps completely novel means.

### 6.3.2 Resolution of Identifiers

As shown in Figure 6.2, a process of *resolution* must take place in order move from initial application identifiers or content descriptions, to rendezvous identifiers, and finally to the forwarding identifiers that are used to propagate data between network elements.

#### Resolution within Application Identifiers

Many applications will perform their own versions of identifier resolution, whereas others may rely on shared services within the network to perform these roles. Such services exist today in the form of search engines and directories, such as Google and *ENUM*. These services simply accept search criteria or inputs as names/addresses/identifiers, and return alternative descriptors or identifiers. The resolution operation can therefore take on many different meanings. For example, resolving an actor's name into possible film titles, or resolving a person's name into a telephone number.

Of course, using an external resolution service implies that *the application has already resolved the correct network rendezvous identifier to communicate with the resolution service*. That is, at some point, an application will wish to communicate over the network, either as a sender or receiver, and it is at this point that it must identify which network rendezvous identifiers to use. Once again, this information can be resolved by the application itself, or through shared network services. For example, an application may query a directory service to determine the correct rendezvous identifiers that it must join in order to participate in a teleconference.

#### Resolution of Application Identifiers to Rendezvous Identifiers and Scope Identifiers

Similarly to resolution within application identifiers, at some point the application will resolve its information requirements into one or more rendezvous identifiers. Scope identifiers should also be resolved by the application if these are not pre-configured. Since the resolution of

rendezvous identifiers is conducted by the application, the number of resolution methods is not restricted. Examples are likely to include application functions for determining rendezvous identifiers from content hashes, using directories, and receiving rendezvous identifiers embedded within data from other applications.

#### Resolution within Rendezvous Identifiers and Scope Identifiers

Since the application interacts with rendezvous identifiers, it is entirely possible for the application to resolve rendezvous identifiers directly. For example, on receipt of some data over the network, the application may extract a rendezvous identifier to be used for return communications. Alternatively, the data may contain only an application identifier (such as a device name) that must be resolved to a new rendezvous identifier via other means.

Most of the aforementioned "linking" between rendezvous identifiers is performed by applications using application-specific knowledge. However, there is also a class of rendezvous identifier resolution that may be performed within the network. Since all applications, and hence all network domains, will use the same rendezvous identifier name space, there is no need to translate between different rendezvous identifiers for the purpose of establishing communications paths between all senders and receivers. However, a rendezvous identifier that is in use may need to be further resolved into a subsequent rendezvous identifier over which error notifications or congestion information is sent. The information with which to perform this resolution may be contained, for instance, within the header of the data sent over the network or in special network control traffic; cf. the previous discussion about the network functions of metadata, in Section 6.1.

If scope identifiers are recursive, then the network also has to perform resolution between scope identifiers. For example, if the scope identifiers form a hierarchy, then a scope identifier higher in the tree may be resolved to lower scope identifiers.

#### Resolution of Rendezvous Identifiers and Scope Identifiers to Forwarding Identifiers

The network must resolve the rendezvous identifier, along with any scope identifier, into a set of forwarding identifiers over which the publisher (or a cache) can send data to the subscribers. This function is performed primarily by the intra-domain and inter-domain rendezvous functions. These function have the responsibility for resolving subscriptions, publications, and pre-publication advertisements where they to exist, into forwarding identifiers. This process is described in further detail later in Section 8.1.

#### Resolution within Forwarding Identifiers

The forwarding capability within the network is responsible for taking traffic from one network interface and transmitting it onto another. The incoming traffic may be on a different forwarding identifier, and hence the forwarder can be seen to perform resolution (switching) between forwarding identifiers. This process may be performed based on forwarding tables, or at the instruction of forwarding directives within the data. The rendezvous systems will have an intrinsic role in setting or calculating such forwarding identifier resolution, for example, from an inter-domain path to an internal path.

## 6.4 Pub/Sub Communication Model

As stated in the previous sections, PSIRP places the notion of information in the centre of attention through addressing information directly (via rendezvous identifiers) and allowing for building networks of information via the concept of scopes (see Section 6.2). In addition to this

information-centric view of our architecture, the underlying communication paradigm plays a crucial role in our architecture, as also outlined in our first high-level architecture in Figure 4.1.

For this, we take the approach that the receiver has control over what it receives and we cascade this approach throughout the multiple layers and elements of the PSIRP architecture. A receiver **must elect** to join (i.e. subscribe) to an identifier before it can receive any information. Sending (i.e., publishing) as well as receiving operations are thus decoupled between the senders and the receivers (or sender and receiver network locations). Thus, data is sent to a (rendezvous & scope) identifier rather than a dedicated network location.

It may be more appropriate to state that we send data *tagged with an identifier* rather than to an identifier. It may, of course, be understood that this identifier means a particular receiver or a network location (although we question whether many applications will ever need to communicate with network locations), or it may represent some more abstract concept, such as a service or a (storage) item of information. Extending this concept through the layers of the PSIRP architecture leads us to avoid using network locations even for identifying network elements, such as routers. Instead, each element is configured to subscribe, or join, the communication identifiers required to route data through to the next network elements and eventually to the receiving applications. If desired, we can still maintain the direct management of each element through its subscription to one or more management identifiers.

Hence, PSIRP intends to move the functionality of many existing publish/subscribe systems (e.g., [Eug2003b]) onto the internetworking layer but also base the entire communication, throughout the architecture, on this paradigm.

An overview of the prior concepts as applied using such a publish/subscribe methodology is shown in Figure 6.3. This example illustrates the presence of a potentially unlimited number of nodes interacting with the network via publish and/or subscribe operations using a potentially unlimited number of rendezvous identifiers and scopes. The rendezvous system is responsible for linking associated publications and subscriptions within the network and properly routing data amongst the pertinent nodes. In this case:

- Node 1 publishes within $scope_1$ to $RId_1$ which has no subscribers, and subscribes to $RId_2$ within $scope_2$.
- Node 2 publishes using $RId_2$ within both $scope_1$ and $scope_2$, with node 3 and node 1 subscribing, respectively.
- Node 3 is subscribed to $RId_2$ within $scope_1$, and $RId_3$ within $scope_2$ (which has no publishers and is therefore considered inactive), and publishes to $RId_m$ using $scope_n$.
- Node x is subscribed to $RId_m$ within $scope_n$.

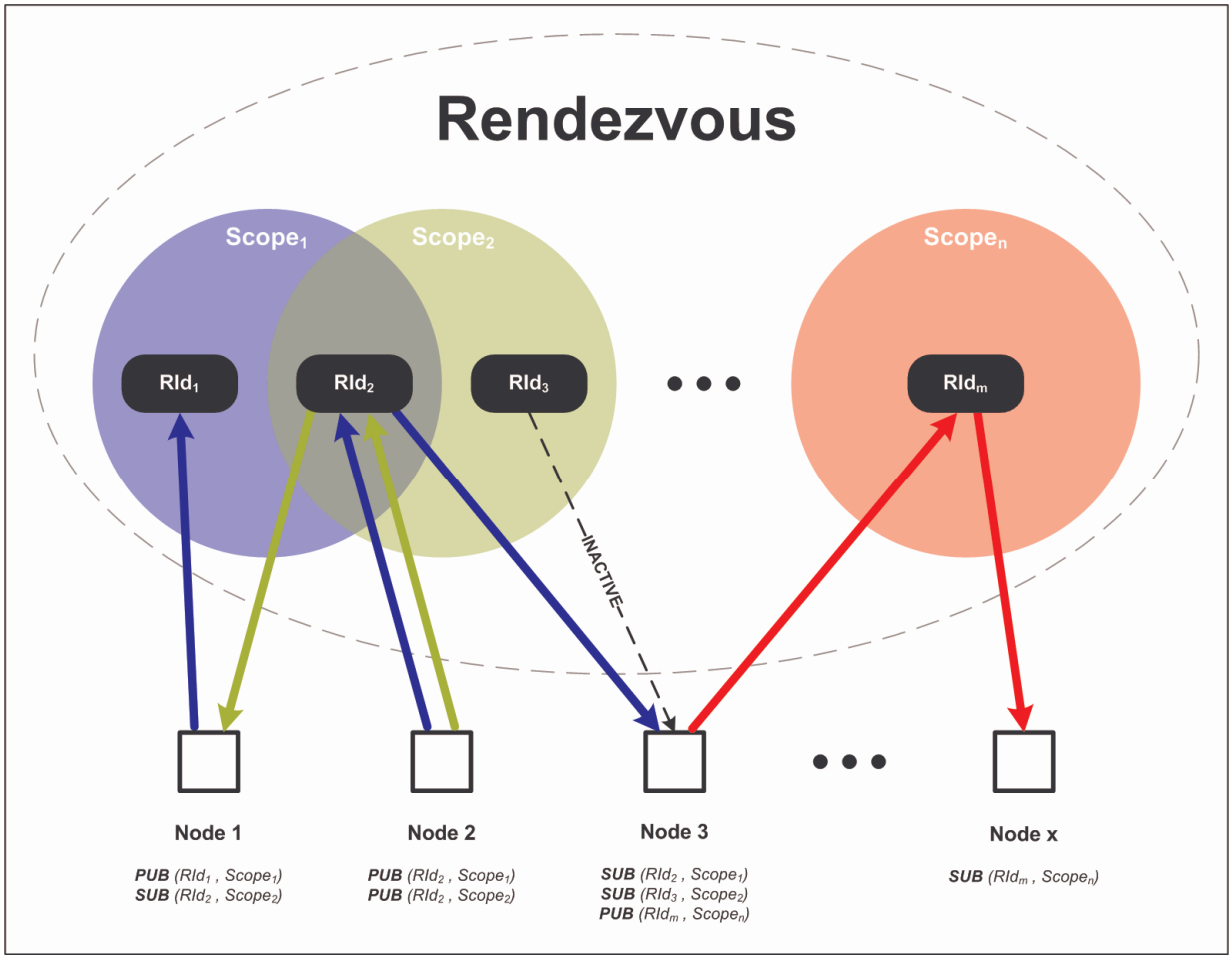


**Figure 6.3 – Pub/sub communication model overview**

## 6.5 Network Coding

Network coding is an umbrella term for a diverse family of techniques involving codification/compression of transmitted packets in order to increase reliability and/or channel capacity. Some of the key goals that underlie PSIRP's architectural design schema include the use of foundational implementations to address reliability, robustness, and performance. Network coding concepts offer tremendous potential to achieve these goals within several facets of the architecture. A key concern is the proper level at which network coding should be integrated. At the present time, we see three somewhat orthogonal possibilities for applying network coding in PSIRP:

1. **Utilize coding at the application level without strict integration into the network.** Architecturally, this only places requirements on the metadata, where the applied coding scheme must be expressed and multiple publication streams (corresponding to the different coding options available) have to be correlated.
2. **Employ simple network coding (such as the XOR-coding [Kat2006]) to increase reliability in specific network segments that meet pre-defined environment conditions (e.g. high error rates, delivering multicast-type traffic, congestion etc.).** Traditionally, wireless networks and especially Wi-Fi networks exemplify these types of environmental factors. Such network coding applications can be accomplished directly on top of the link layer, thus remaining transparent to PSIRP lower layer implementations.
3. **Integrate network coding into the topology and low-level forwarding functions.**

The first two options can almost certainly be exercised. Research is needed to select the appropriate solutions, e.g., to optimally solve the reliability issues related to publish-subscribe traffic delivery in wireless networks, but little specific considerations are necessary at the architectural level.

The third option is the most challenging one, in an architectural sense. While potential gains from the integration of forwarding and coding (and potentially caching) appear to be significant, it would also impose limitations on the level of manipulation that can be exercised over information that is being transmitted in the network. For example, applying policies on publications may become significantly harder. Such an application of network coding could also raise various security concerns, ranging from hampering packet inspections to "information leaks" between domains, which might be unacceptable for policy reasons. At minimum, domain-level policies need to be taken into account in setting up coding schemes, and at worst, packets or chunks would have to be decoded at border nodes connecting separate domains. The high traffic levels typically present within such nodes might severely curtail the design space for realistic network coding schemes. On the other hand, other considered features, such as universal caching, may pose similar challenges and require solutions that are likely to be compatible with those required here. Hence, a proper economic analysis may be required, in order to consider the potential benefits vs. any drawbacks and other complications.

In summary, the different varieties network coding are attractive technologies which will be seriously considered for inclusion within the PSIRP architecture as the project progresses. However, beyond applications that are either below or above the core PSIRP components in traditional layered terms, the architectural considerations may become quite complicated. It is also important to realize that network coding is not a "magic bullet" for solving reliability and capacity problems. While the potential benefits associated with proper network coding implementations are theoretically tremendous, a number of roadblocks remain that impede practical usage. A key component of the project's future work will be dedicated to critical investigation and evaluations of these considerations.

# 7 Conceptual Architecture

This section presents the PSIRP conceptual architecture. The architecture is based on the general and PSIRP principles, and defines the key entities and processes of the system.

## 7.1 Overview

The PSIRP conceptual architecture consists of three crucial parts, namely the protocol "stack" architecture (called the *component wheel*), the *networking architecture*, and the *service model*. These are not independent parts, but rather together they form the basis for different PSIRP networks and network applications. The first part defines how the protocols are organized within a single node and their components. The second part defines how the PSIRP network functions as a collection of entities based on the in-node protocol organisation. The third part defines the network interfaces towards data subscribers, publishers, and network services.

## 7.2 PSIRP Component Wheel

The PSIRP conceptual architecture is based on a modular and extensible core, called the **PSIRP component wheel**. The architecture does not have the traditional stack or layering of telecommunications systems, but rather components that may be decoupled in space, time, and context. The idea of such a layerless network stack has been proposed before, for example, in the Haggle architecture [Hag2007]. The novelty of the PSIRP proposal is to use publish/subscribe style interaction throughout the conceptual architecture, and thus support a layerless and modular protocol organization.

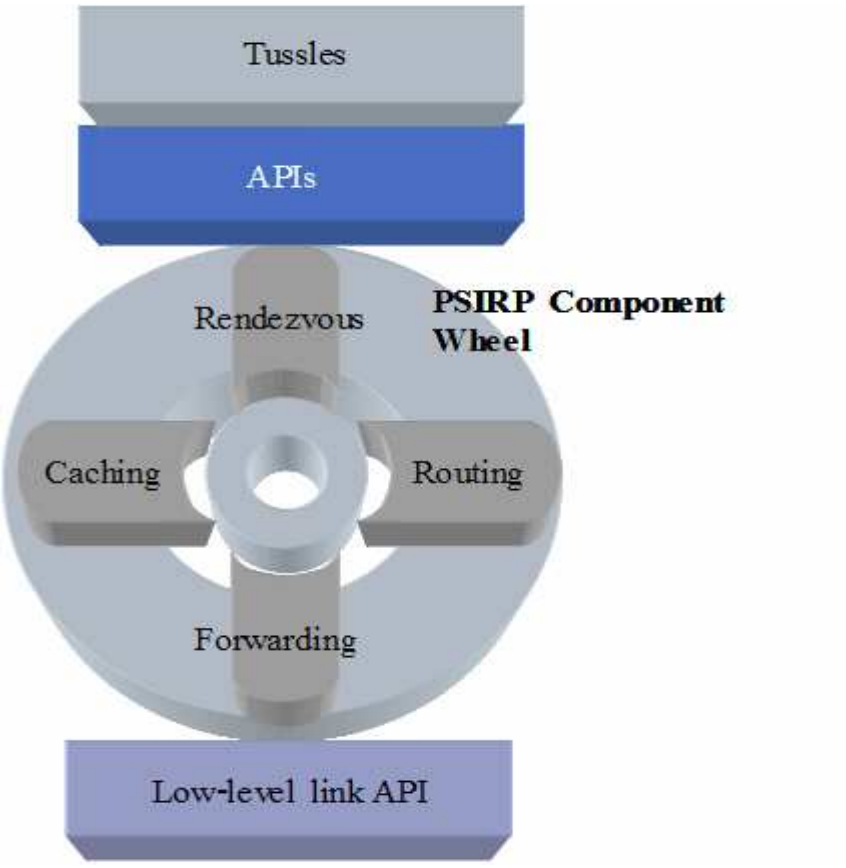


**Figure 7.1 – PSIRP component wheel**

Figure 7.1 presents an outline of the conceptual architecture with the PSIRP component wheel in the middle. Above the wheel, we have APIs that allow for using different networking features available in the system. The figure illustrates the typical components needed in the wheel for inter-domain operation, namely *forwarding, routing, rendezvous,* and *caching*.

## 7.3 PSIRP Network Architecture

The **network architecture** is a collection of entities that are based on the PSIRP component wheel architecture. The entities use common syntax and semantics for publications, identifiers, and processing rules. In this section, we first define the *entities* that are central to the conceptual network architecture, and then present the key network *processes* which are needed for distributed operation both within managed domains (intra-domain), and between managed domains (inter-domain).

The conceptual architecture consists of the following entities:

- *Identifiers*. As previously discussed, there are three or four classes of identifiers, namely forwarding identifiers, scope identifiers (which may end up being a subset of rendezvous identifiers), rendezvous identifiers, and application identifiers. Application identifiers are resolved into rendezvous identifiers, and ultimately to forwarding identifiers, also called *labels*. This resolution is done by the protocol wheel and applications are mainly concerned with the application identifiers, using rendezvous identifiers to interface with the network.
- *Data and metadata*. A rendezvous identifier is implicitly associated with a well-defined (but not necessarily fixed) data set, consisting of one or more publications. The data sets may also have associated metadata, which may include, e.g., scoping information and other useful information either for ultimate receivers or for network elements. The metadata can also be considered data. It may be provided to the network within a publication to which it relates, or it may be provided as a separate publication to a rendezvous identifier. For example, it may be provided as a publication on a separate rendezvous identifier which can be linked to the rendezvous identifier of the publications which it affects.
- *Scoping information* is associated with a publication, e.g., as part of metadata or through a scope identifier, see Section 6.2. Scopes determine the elements of the rendezvous system that act on published data and therefore define the (information) *network* the information belongs to. A publication may be associated with one or more scopes and they can be interpreted at various levels of abstraction.
- *Subscribers and publishers*, which create publications, and consume publications, respectively.
- *Domains*, which are administrative network areas that can be connected using the inter-domain forwarding architecture.

Figure 7.2 presents an example overview of the above entities and highlights their relationships; especially the relationships are subject to change based on implementation experience (see Section 2). As discussed before, data packets and their associated metadata are published and subscribed in a domain. Typically, the data is associated with one or more application identifiers and one or more scopes. At this writing, there are multiple opinions where to include the scoping information; for example, the scoping information may be included in the metadata part of the packet or it may be represented as a separate identifier in the packet header. As discussed in Section 6.3, each application first resolves application identifiers to rendezvous identifiers and then hands the rendezvous identifiers to the network, using the scopes to properly map each rendezvous identifier to one or more forwarding identifiers, both within a domain and between domains. Within a domain this is called intra-

domain routing and forwarding, and between domains this is called inter-domain routing and forwarding.

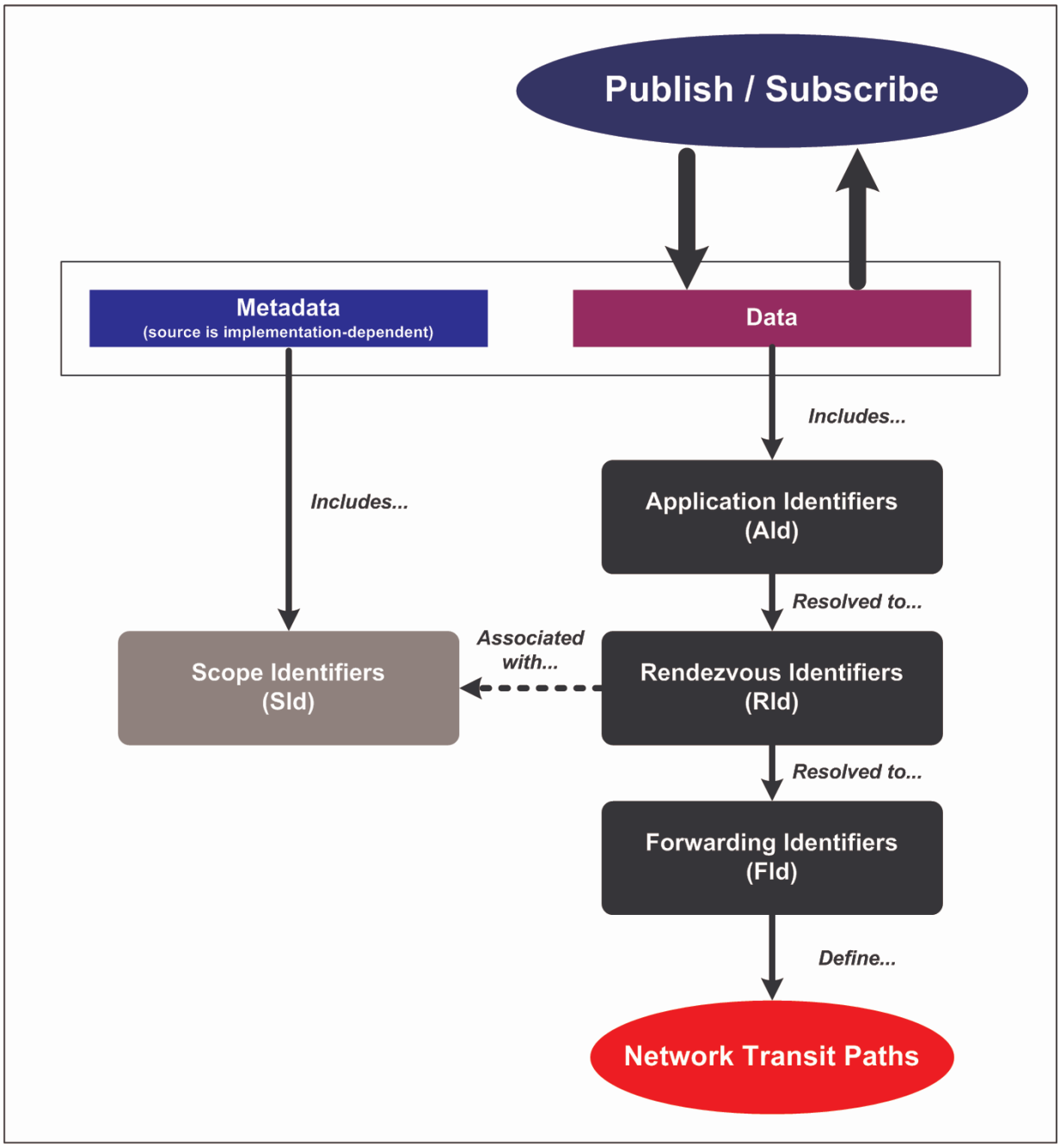


**Figure 7.2 – Key entities of the conceptual architecture**

We note that it is beyond the scope of the current conceptual design how, exactly, metadata is provided. Above, we discussed two examples: *in-flow* provisioning or provisioning as *separate publication*, linked through rendezvous identifiers.

The architecture consists of the following six key processes:

- *Rendezvous,* which is the process of resolving rendezvous identifiers into forwarding identifiers within a given scope. The scope determines the part of the rendezvous system that is used by the network. The three simplistic, topology-oriented cases, reflecting the current usage, are link-local, intra-domain, and inter-domain scopes. However, we expect that future applications will use more semantically-based scopes

instead of such topological scopes, implementing, e.g., scopes based on social networking structures. The rendezvous and scope identifiers are handles to the rendezvous system. Together they uniquely identify the given publication and the associated policies and metadata.

- *Intra-domain routing and forwarding* pertains to data delivery in an administrative domain. Intra-domain routing is concerned with local policies.
- *Inter-domain routing and forwarding* pertains to data delivery in the global network, typically spanning several domains. The inter-domain routing system is configured through the rendezvous process and takes into account any inter-domain policies in effect.
- *Forwarding and transport,* which pertains to data transfer between subscribers and publishers.
- *Caching* is a network process offered by either the local system or any system on the communication graph over the network.
- *Network attachment* is responsible for discovering network attachment points and configuring components in such a way that communication becomes possible.

These processes are presented and discussed in Section 8.

## 7.4 Service Model

The PSIRP *service model* determines the information flow and semantics that are supported by the network. The service model includes the interface between the network and applications, and also the interface between the network and network management tools. We divide the service model into three parts, namely *publisher/sender*, *subscriber/receiver*, and *network services*. The first part defines how data can be sent to and over the network and what primitives are offered by the network for this. The second part defines how data can be received from the network and defines an interest-registration service and the necessary upcalls for data reception. The third part defines the monitoring and controlling points offered by the network for management purposes.

### Publisher/Sender

The publisher/sender interface supports publication of data. Each publication has an associated flat label, called the rendezvous identifier, and an optional metadata part. Such metadata could, for instance, specify the *usage policy* for a particular publication. The following table gives an overview of possible features of the publisher/sender model. Which of these features are technically feasible to implement and economical to support is still being discussed within the project. For example, even the concept of congestion is problematic given the data-oriented nature of networking and the multicast and caching features.

| Feature | Description |
|---|---|
| *Metadata* | Publisher includes metadata for the network or for the eventual receivers. Such metadata may be represented as a separate "meta"-publication, or through other means. |
| *Publisher anonymity* | Network ensures publisher anonymity upon request by the publisher. Publisher remains (locally) accountable, though. |
| *Multicast* | Publisher can send a publication to many subscribers through publishing the publication only once. |
| *Data correlation* | Publisher can indicate that certain publications are likely to be subscribed together. This is most likely to be implemented using metadata. The network is not guaranteed to understand correlations, i.e., some implementations may not care about such correlation information. |
| *Caching* | Publisher can indicate caching preferences to the network, e.g., through to-be-specified metadata fields. |
| *Anycast* | Publisher can send anycast publications that are delivered to only one subscriber. This feature is meant to be mainly used in functions like local discovery. At this writing it is unclear whether global generic anycast can be supported, or only local specialized anycast-like services. |
| *Scoping* | Publisher must indicate one or more scopes for a publication. To publish a publication in several scopes may require multiple operations. |
| *Accountability* | Publisher is authorized by the network. The network operator may be later able to prove that the publisher has indeed published a given publication. However, at this writing it is unclear what such proof exactly means or what are the expected semantics for publisher identity. |

### Subscriber/Receiver

A subscriber initiates a receiver-driven communication through a rendezvous identifier, specified in an act of subscription. Similar to the publisher, the subscriber can specify additional metadata surrounding the request (e.g., define a *usage context* in which the subscription is supposed to happen). The table below gives an overview of the possible features of the subscriber/receiver model. The same caveats that apply for the publisher/sender model apply here, too.

| Feature | Description |
|---|---|
| *Subscription state removal* | Network may offer different techniques for removing subscription state, for example, explicit leave, implicit leave, auto leave, and expiration. |
| *Publisher authentication* | Subscriber requests that the network authenticates publisher. However, as indicated above, at this writing it is unclear what exactly is meant with publisher identity, and therefore the exact semantics of publisher authentication must also be left open. |
| *Data integrity* | Subscriber requests that the network delivers only data that has known origin and is integrity protected. |
| *Accountability* | Subscriber is authorized by the network. The network operator may be later able to prove that the subscriber has indeed subscribed to given publication. However, at this writing it is unclear what such proof exactly means or what are the expected semantics for subscriber identity. |

### Network Services

The network services part of the service model is responsible for supporting various administrative services, such as network management and measurement tools. Central to this part is the support for rudimentary directory services, which allow network devices to bootstrap communications. The directory services are realized within the rendezvous system.

From the management point of view, there are two key phases in communications: first, we have the rendezvous phase, which sets up or re-uses existing forwarding state from publishers towards subscribers. Second, we have the forwarding phase, in which data is delivered using the forwarding state from publishers (or caches) to subscribers. The rendezvous phase is slower, because it needs to manage policies and process metadata associated with a subscription or a publication. The forwarding phase is faster given that it is done using existing forwarding trees and using the forwarding identifiers.

Network management features, such as policy decision making and policy enforcement, are expected to happen in the rendezvous system. This means that changes to policies are first processed by the rendezvous system and they may or may not imply changes to the forwarding states. Features specific to certain publications or scopes are expected to be expressed as metadata linked to the respective identifiers.

# 8 Components

This section presents the key PSIRP network components, namely the rendezvous system, intra-domain and inter-domain routing and forwarding architectures, forwarding and transport, caching, security policies and access control, and network attachment.

## 8.1 Rendezvous

### 8.1.1 The Role of Rendezvous

When moving from an end-point-oriented internetworking architecture to an information-oriented internetworking architecture, the traditional *routing* function of packet networks becomes divided into several functions. Perhaps the most important of these is rendezvous, whose role is to match the interests of senders and receivers. That is, rendezvous implements the function of determining the set of subscribers and publishers for a given publication and instructing the underlying network machinery to perform the needed data delivery.

Within the PSIRP architecture, rendezvous is accomplished through specialized functions that operate between physical network devices known as *rendezvous points* (RPs), which can be viewed as being either fixed or non-fixed indirection points for network communications. The collection of network RPs together form the foundation of the PSIRP *rendezvous system*, which is responsible for associating data subscribers and publishers to some scope implementing the PSIRP design principle PS2. In this context, rendezvous is inherently related to the notion of publication scoping in that a scope (amongst other things) determines the subset of RPs that are responsible for a given publication and/or subscription. The importance of the services offered by the rendezvous system cannot be understated; rendezvous is crucial in providing a control plane for connecting publishers and subscribers in a policy-compliant fashion both within and between administrative domains.

### 8.1.2 Rendezvous Points

While the rendezvous system is a logically centralized service, its functionality is distributed over a physical set of interconnected RPs, both within any given domain and across domains. Selecting the appropriate devices and domain locations to house RP services is a somewhat arduous task. Rendezvous represents a cornerstone feature in PSIRP's pub/sub implementation, and great care must be exercised to ensure its reliability and robustness in the face of every foreseeable operating scenario.

A refinement of the original end-to-end (E2E) principle by Clark et al [Cla2007] outlines the importance of **trustworthiness** of execution rather than particular placement of functions in endpoints. This so-called *Trust-to-Trust* principle is captured as our design principle G1 and considered in our design in the sense that **rendezvous should occur at locations within the network that are trusted to operate correctly in terms of communal, economical, and functional requirements**. Bearing this in mind, the process of determining the optimal network locations for RPs must account for various environmental factors as well as inter-domain traffic policies and pub/sub scoping mechanisms. The specific nature of these concerns is obviously highly implementation dependent, and further research and development of the PSIRP architecture is required in order to arrive at a conclusive set of evaluation criteria. It is expected that the scoping mechanism, with different policies associated with different scopes, may help in achieving a scalable solution.

### 8.1.3 Rendezvous at Different Internetworking Levels

Due to its importance in policy enforcement and defining (often user-created) information scopes in various situations, the rendezvous system constitutes a relatively well-defined environment where tussle is likely to commence [Cla2002]. The rendezvous system is therefore a policy-enforcement point in the architecture and a mechanism for supporting freedom of choice for network end points. Similar rendezvous functionality has been used in many distributed systems, for example HIP [Mos2008] [Egg2004], IP multicast (RFC 2362), i3 [Sto2002] and Hi3 [Nik2004], FARA [Cla2003], PASTRY [Row2001], and HERMES [Pie2004].

Figure 8.1 considers the many faces of rendezvous. In addition to its involvement in basic packet delivery, rendezvous plays a role in higher level network activities including inter-domain policies and communal aspects like those observed in services such as Facebook and Flickr. These aspects of rendezvous are discussed in more detail in Section 9.

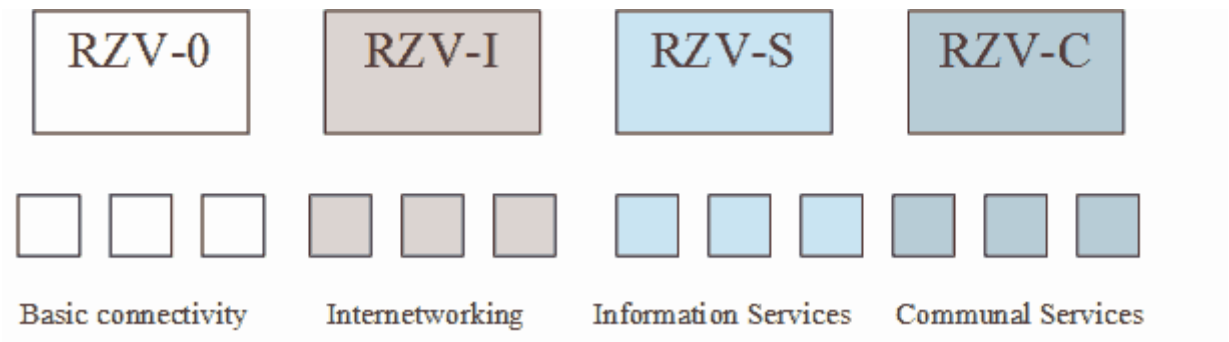


**Figure 8.1 – The many faces of rendezvous**

Figure 8.2 presents an overview of rendezvous within local, intra-domain, and inter-domain network environments. As discussed in the previous section, the rendezvous entities involved in each of these settings are necessarily determined via publication scopes (see Section 6.2).

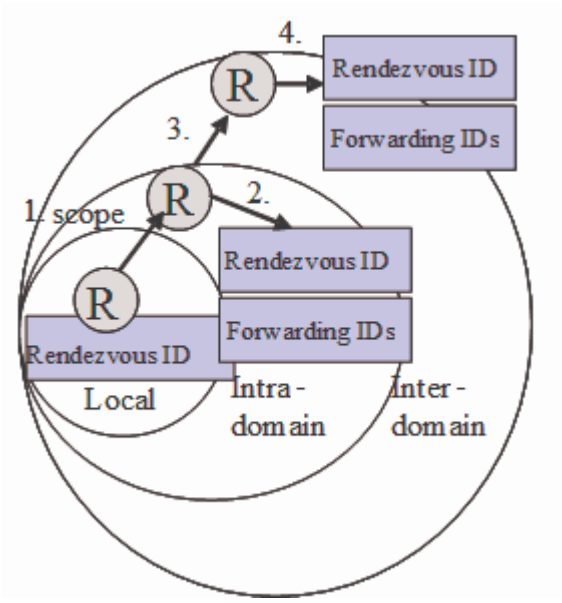


**Figure 8.2 – Network architecture with rendezvous**

A *rendezvous identifier* (RId) is associated with policy-compliant data dissemination graphs for publication delivery, both in the local domain (intra-domain) and between domains (inter-

domain). The rendezvous identifiers are chosen from within a large enough set to provide a probabilistic guarantee of uniqueness without a central allocation authority. Applications may resolve *application identifiers*, which are contained within published data, into rendezvous identifiers. It is then the responsibility of the rendezvous functions to find suitable data transit and delivery paths in the network and denote them with *forwarding identifiers* (FIds). This resolution from a rendezvous identifier to a set of forwarding identifiers is based upon the rendezvous identifier in conjunction with scoping (as identified through the *scope identifier*, Sid) and policy mechanisms. The breadth of reference of FIds is variable, potentially limited to single hops or dynamically expandable to encompass full multicast trees.

Rendezvous state is created to allow the subscriptions and publications to meet within a specified scope. Subscriptions and pre-publication notifications (or *advertisements*), possibly included in metadata or data-correlation notifications, may be utilised by the rendezvous system to create partial forwarding state from publishers towards subscribers. When a publication becomes active, i.e., when there is both an actively sending publisher and one or more active subscribers, the rendezvous systems are used to complete the forwarding path by mapping the rendezvous identifier to intra-domain and inter-domain forwarding identifiers. This late mapping can be used to implement dynamic policies, such as inter-domain routing and caching.

The rendezvous system ensures that neither traffic policies nor publication/subscription policies and scopes are violated. The rendezvous system is also responsible for ensuring that overall network overhead remains within reasonable bounds; the system can dynamically adjust its operational characteristics and control tradeoffs between configuration latency and data delivery overhead.

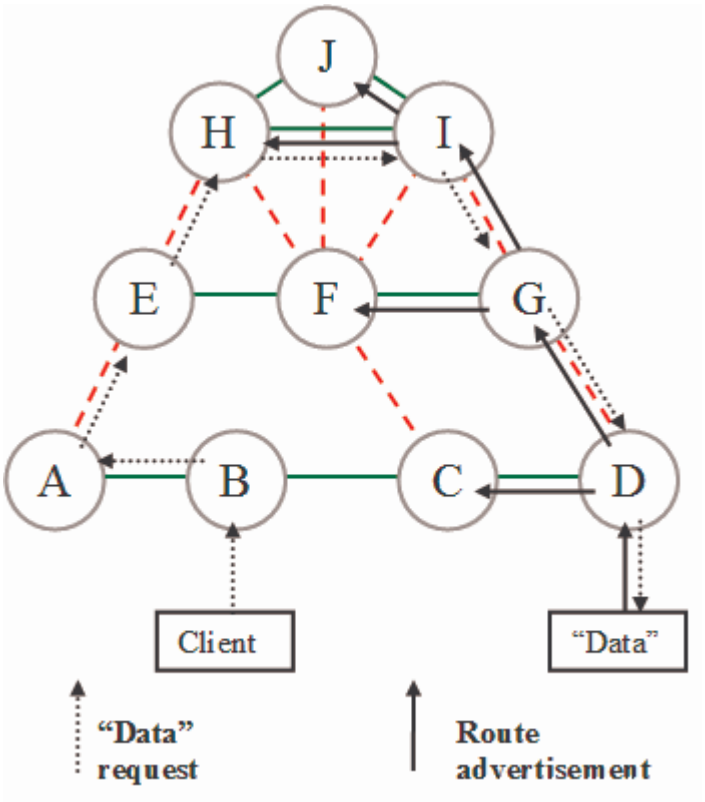


**Figure 8.3 – Inter-domain rendezvous example**

In practice, inter-domain traffic policies impose effective limitations over the distribution of rendezvous state between different network domains. Figure 8.3 illustrates a sample inter-domain rendezvous scenario involving a network with ten domains. Those RPs that are

selected to handle inter-domain coordination are likely responsible for providing a view of the entire local domain to neighbouring domains. Here, the dashed lines represent transit relationships between domains and solid lines represent peering relationships. In this example, advertised publication routing state is distributed to all immediate peers and providers (solid arrows). Subscriptions are routed to siblings and providers (dashed arrows), and follow advertised publication routing state when encountered (in domain H in this example). This structure expressed in this example denotes one possible inter-domain methodology that arises from existing business relationships amongst separate administrative domains.

## 8.2 Inter-domain Forwarding Architecture

The PSIRP inter-domain forwarding architecture defines the publication forwarding functions at the domain-level, i.e., between administrative domains. The structure and size of the inter-domain forwarding tables are crucial for system scalability, since the forwarding functionality needs to be implemented using line-speed hardware (the so called *fast path*). In this section, we briefly consider several alternatives for inter-domain forwarding and then consider the implications for the PSIRP architecture. The rendezvous system is responsible for configuring and maintaining the inter-domain state in a policy compliant way.

All forwarding in the PSIRP system happens via *stacks of forwarding identifiers*, carried in publications (or fragments thereof) while in transit. The forwarding identifiers are unique only within a set of domains, and sometimes only within a single domain or even a single link. The forwarding identifiers are removed from the stack before entering domains where they are not valid any more. A publication is forwarded to the neighbouring domain if (and only if) such forwarding is compliant with the domain's inter-domain traffic policy.

Forwarding identifiers can be assigned for various purposes:

- *Per publication:* In this case, a domain assigns a unique forwarding identifier for each active rendezvous identifier visible within the domain. This is simple but will not scale to a large number of rendezvous identifiers. Hence, a domain may decide to assign unique forwarding identifiers for some critical entities which the PSIRP architecture is built on (e.g. rendezvous entities, domain sprouters, neighbouring domains), as well as the most popular rendezvous identifiers within the domain. Having unique forwarding identifiers for some publications will allow flexible and optimal multicast delivery.
- *Shared forwarding trees, tree segments, or paths:* The forwarding identifier is shared among many publications, i.e. many publications map to the same forwarding identifier. This will save forwarding state, but makes mapping from rendezvous identifiers to forwarding identifiers more complicated.

After the last identifier in a stack has been consumed, the publication should be at the nodes where it has been subscribed to. Due to the possibility of using shared delivery trees, the rendezvous identifier of the publication needs to be utilized to deliver the publication only to the applications entities that have subscribed to it.

At the inter-domain level, an encapsulating forwarding option is needed so that domains that do not have any subscribers for the publication in question are relieved from the burden of state maintenance (at least at the forwarding level) for such publications. This is crucial for the load that the PSIRP architecture places on central transit operators (e.g. Tier-1 ISPs). Due to the fact that arbitrary policy-compliant domain-level paths are usually rather short (e.g. [Gao2001] finds that the longest existing AS path is 13 hops), a publication header length need not increase significantly due to the inclusion of an encapsulating path-vector component. For example, *domain-level path-based forwarding* is a well-known encapsulation technique, recommended for Internet evolution also for security reasons [Han2004], and used in many of the recent clean-slate architecture proposals, such as ROFL and DONA. In PSIRP,

the domain-level paths would be built during the rendezvous process: A domain-level path would be recorded as a side effect of routing a rendezvous identifier from domain to domain.

The choice between various delivery trees, per-publication forwarding identifiers, and domain-level paths is a local policy decision of the rendezvous system at each domain through which the publication and subscription requests are routed. This rendezvous phase is responsible for producing policy-compliant paths, but nonetheless the forwarding functionality may need to be capable of performing policy enforcement and dropping all traffic not compliant with the domain's inter-domain traffic policies.

The differently assigned forwarding identifiers can be used together to construct arbitrary content distribution hierarchies. Typically, an overall end-to-end delivery tree is formed of multiple path-segments joined together by publication-aware forwarding junctions. These junctions are natural positions for, e.g., content caching.

### 8.2.1 The Inter-domain Interface

For the forwarding functionality, each domain needs to announce their presence at the inter-domain links. These announcements need to be flooded into the neighbouring domains so that each router in each domain knows how to forward packets to all their neighbours. This can happen by the receiving domain assigning unique forwarding identifiers to the neighbouring domain (Figure 8.4), and the rendezvous entities in the receiving domain maintaining a mapping between the neighbouring domain identifier and the assigned forwarding identifier. Then, when the subscription or publication advertisement messages are routed by the rendezvous entity, the corresponding forwarding identifiers can be added to the forwarding path vector associated with the rendezvous state.

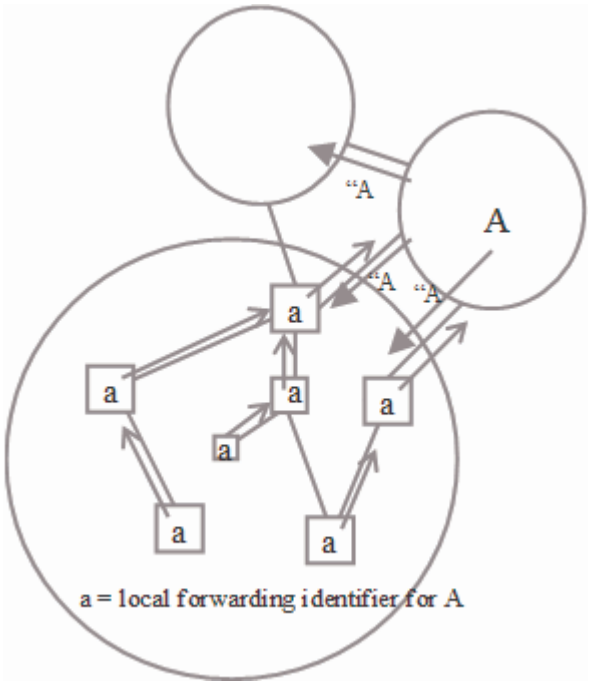


**Figure 8.4 – Domain A advertises its presence to its neighbours**

### 8.2.2 Inter-domain Packet Format

Each packet arriving over an inter-domain interface has a *path-vector* component. The elements in this vector are forwarding identifiers*, some of which are assigned to individual neighbouring domains*. The exact format for the forwarding identifier is outside the scope of this document while the nature of the path vector is the concern of this section.

One design goal for the path vector is robustness against denial-of-service attacks. Thus it would be desirable if only the designated node to which the path vector is given could make use of it. There are a number of possible ways to attain this goal:

- Use of *capability tokens*: The user of the path vector is given a cryptographic token, which is verified before the packet is forwarded.
- Use of *non-global forwarding identifiers*: The forwarding identifiers used in the path vector need not be globally unique. Each domain may locally name their neighbouring domains with random numbers and insert these local labels into the path vector. If the path vector is used from an incorrect source, the path becomes essentially random, and will likely be soon cut due to inter-domain policy violation.
- Each domain could use *multiple forwarding identifiers* to signify the same neighbour, thus making it hard to compare two different path vectors and find correlations between them.
- *Limited lifetime forwarding identifiers*: Each local identifier used for a neighbouring domain could be associated with a limited lifetime. Thus the corresponding path vectors are of limited time use. The rendezvous process will need to calculate the path vector composite lifetime as the path is recorded (the minimum of the lifetime of any of the vector components). The subscription request could also carry a desired lifetime so that the domains would know which labels they can affix to which path vector.

Another design goal for the path vector would be compactness. As the network grows, the number of the path components may slightly increase from the current maximum of 13 [Gao2001]. Some domains have only a few neighbours, while some have hundreds. A scheme where the forwarding identifier encoding can be of variable length is therefore an option. In any case, each domain on the path will need to be able to decide the number of bits it adds to the path vector by itself. We envisage that with a proper scheme a typical path vector will use less space than, e.g., the source and destination IPv6 addresses combined.

## 8.3 Intra-domain Forwarding Architecture

#### Introduction

The goal of the intra-domain routing and forwarding in PSIRP is to finally deliver the publications to all interested subscribers. In one possible case the publisher lies in the same domain as some subscribers and in the second case the publication arrives from a remote domain; in this latter case the intra-domain forwarding starts at the domain entry point.

In the intra-domain model for PSIRP at least three kinds of different functions are distinguished: active network elements, called sprouters, with the task of connecting publishers and subscribers through subscription and publication routing and forwarding; end-nodes that represent the clients in the architecture, i.e., information producers (publishers) and consumers (subscribers); and rendezvous entities that allow subscriptions and publications to converge and form forwarding state. Note that this section is not meant to give a detailed picture about rendezvous, but it should deal with this functionality as routing and forwarding are partly implemented through and highly depend on the rendezvous functionality, as we will see later.

**Viewpoints for Routing and Forwarding**

This section aims to present the concepts that help to explore the design space in the intra-domain routing and forwarding architecture. One of the key goals is to find a scalable architecture and this also means a scalable intra-domain forwarding mechanism. It has been identified that there are at least three viewpoints in connection with scalable routing in PSIRP.

The first dimension is the forwarding efficiency. This means the usage of optimal distribution trees in the domain for publication delivery; i.e., the subscribers get the publication on the shortest paths and duplicates are only created if needed. The shortest path can mean the path with the lowest hop count, minimal delay, lowest congestion, lowest risk in case of failure etc.

Another viewpoint is the amount of the state in the network entities. The size of the routing tables is more and more becoming a problem in the current Internet. In a pub/sub based model the number of publications is several orders of magnitude higher than the number of prefixes in the current Internet, and hence a major challenge of the PSIRP project will be to find a scalable solution to the problem of network state. It is clear that even in the intra-domain case it is not scalable to have a separate entry for each rendezvous identifier in the sprouters on the fast path. This is one of the reasons why forwarding identifiers were introduced in Section 6, as multiple rendezvous identitifiers can be aggregated onto the same forwarding path.

The third viewpoint is the overhead in the packet headers, i.e., the amount of information carried in the packets for routing and forwarding purposes. As an example from the current IP world, one can imagine strict source routing as a design choice with a large amount of routing information in each packet, while traditional destination-address-based hop-by-hop routing needs a large amount of entries in the routing tables in the routers.

The next paragraphs discuss some high-level options in the context of the three scalability dimensions we have just outlined. The following major choices are introduced:

- Flooding
- On-path Rendezvous
- Forwarding trees
- DHT-based concept

### 8.3.1 Flooding and On-path Rendezvous

**Flooding**

A trivial intra-domain solution is to hold subscriptions at the edge of the network and to flood publications across the network. This approach is taken in the gossip type communication protocols [Eug2003a] and results in very high bandwidth utilisation as publications are delivered along forwarding paths where subscribers do not exist. This approach is most efficient when there are very high numbers of dynamic subscriptions and relatively few publications.

The classical approach to publish/subscribe is to instead flood subscriptions throughout the network. A publication at any point can then be routed efficiently towards the subscriber. Although efficient in terms of publication bandwidth, this method suffers from subscription state and subscription churn overheads. The introduction of a third category of message, or advertisement, can be used to only propagate subscriptions to where the may be relevant. This approach is used by SIENA [Car2001].

### On-path Rendezvous

Let us assume that there is one sprouter providing rendezvous functionality for each scope. This sprouter has previously subscribed to a well-known identifier so that others can send publications and subscriptions (requests for publications) toward it within the particular scope. This will mean that every subscription and publication will traverse through the Rendezvous Node. (As an implementation note we state that there can be many Rendezvous Nodes in the network subscribing to the same scope identifier which would help scalability and prevent them from being the bottleneck of the network.)

As the subscription flows from the end-point (and its connected sprouter) towards the Rendezvous Node either a forwarding directive can be collected by the subscription that identifies the return path or states can be created in the sprouters meaning where to forward the publication. This second option has the advantage that it offers state merging in the sprouters on the path.

This process is illustrated by Figure 8.5 in which two hosts subscribe to the same rendezvous identifier (P1). Note that as the subscriptions are propagated towards the Rendezvous Node, the subscriptions are aggregated, i.e. Sprouter C in the figure does not have to forward the subscription twice toward the rendezvous node (which is Sprouter D in the example).

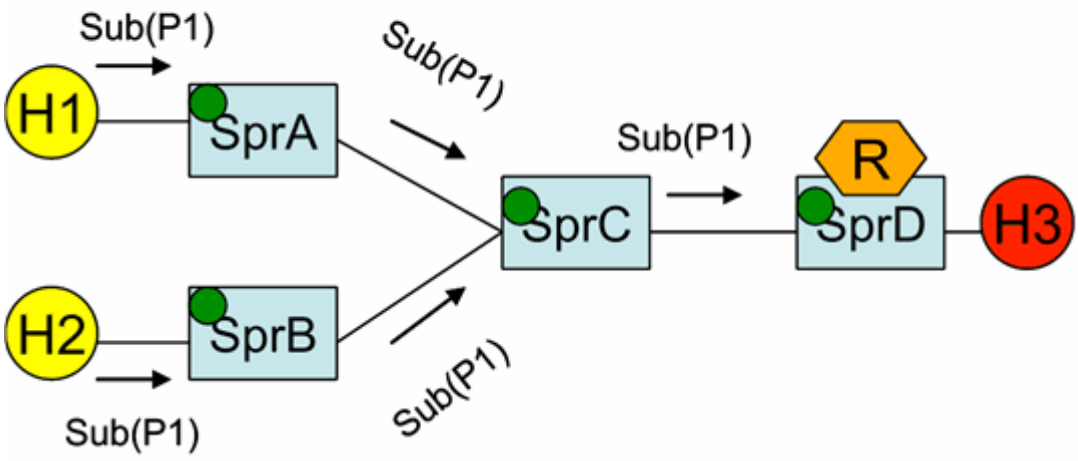


**Figure 8.5 – Subscription state maintained in the sprouters (subscription phase)**

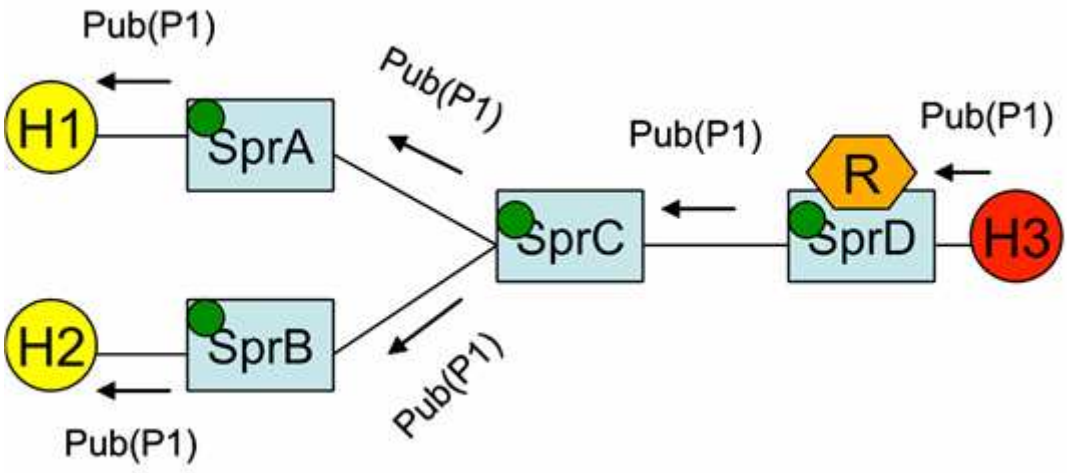


**Figure 8.6 – Publication directed by subscription state (publication phase)**

Modifications of this approach can be discussed that alleviate the amount of state in the network. One approach that can be used is for the subscriptions to collect forwarding directive information as they are routed towards the rendezvous node. Instead of installing state in intermediate sprouters, this forwarding directive can be used to route publications to the end subscribers. Multicast dissemination of publications can still achieved by aggregating the forwarding directive information for a single rendezvous identifier. The trade-off is that both subscriptions and publications have higher packet overheads due to carrying forwarding directives and that state is moved from the intermediate sprouters into the rendezvous node.

Another approach, discussed in detail in the next section, is to form forwarding trees across the network and perform an early match of publications to a set of forwarding trees that cover all subscribers.

### 8.3.2 Forwarding Trees

This solution aims to keep the number of states in the system low, while offering a compromise between the number of states and transport efficiency. The operator maintains some forwarding trees in the domain, one of which covers all the sprouters residing in the domain, while the others cover different subsets of the sprouters. (Consider the example of Figure 8.9 where there exists a global spanning tree and another tree covering 3 sprouters in the domain.) Therefore every router residing on a tree will store forwarding states specifying the neighbours on the particular tree. It is the responsibility of the rendezvous functionality to map a publication specified by a rendezvous identifier to a tree specified by a forwarding identifier. This mapping is basically determined by active subscriptions, but it can also be based on policy decisions, scope checking, QoS requirements etc.

An extreme case of this solution is to have only one global spanning tree with all publications mapped to this single tree. This is a more efficient solution of the publication flooding mechanism presented above. Again, the disadvantage is that publications are delivered for which no subscriber exists at a sprouter. Another extreme sub-case is for each destination sprouter to have its own concast forwarding tree connecting all rendezvous nodes to itself. In this case the rendezvous node would perform an early match of the publication against subscriber interest and forward the publication to the trees for every sprouter to which subscribers are attached. The disadvantage of this approach is that the ability to multicast is

lost, since a single publication may be sent multiple times across a single network link to different destination sprouters. This approach is shown in Figures 8.7 and 8.8.

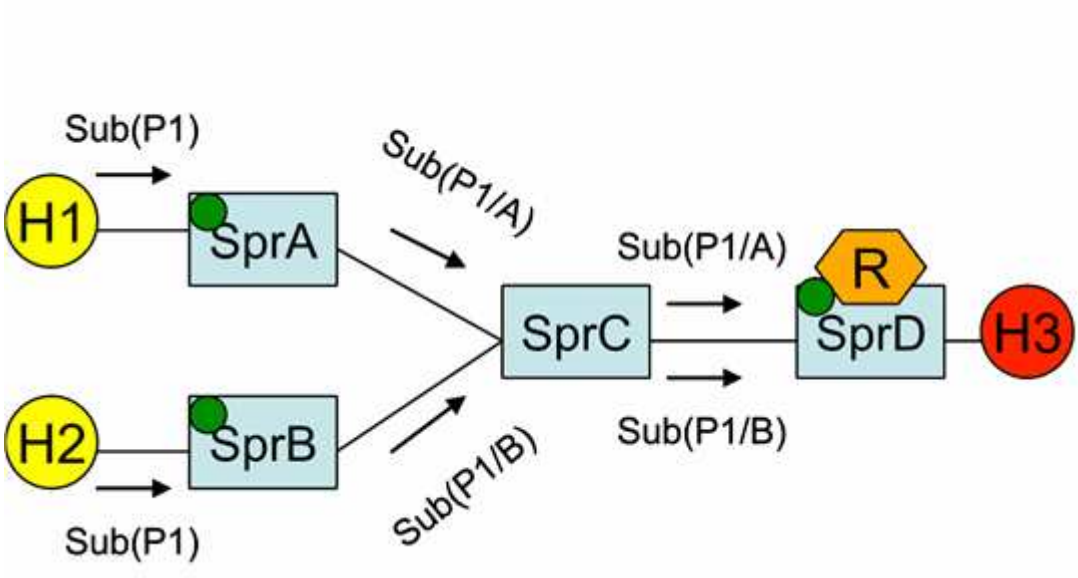


**Figure 8.7 – Individual sprouter concast tree (subscription phase)**

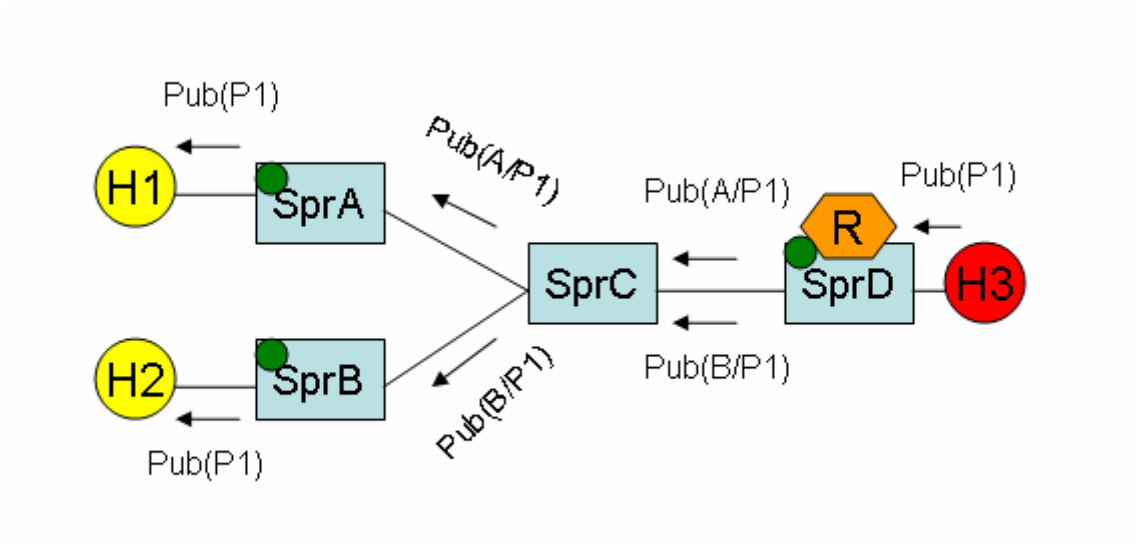


**Figure 8.8 – Publication to each individual sprouter concast tree (publication phase)**

Another approach is to have one tree for each subset of sprouters in the domain. This will result in a huge number of forwarding trees, however it is ensured that the best possible tree for the particular publication always exist and therefore can be used for publication delivery. Apart from the transit sprouters lying on the paths connecting the publisher and the subscribers, no entities will get unwanted content. Some examples of forwarding trees are shown in Figure 8.9.

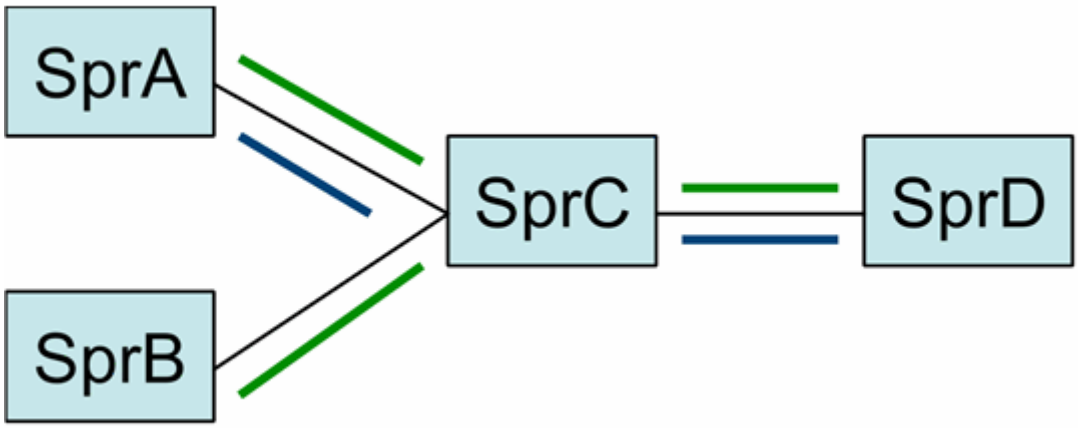


**Figure 8.9 – Forwarding trees in the network**

### 8.3.3 DHT-based Forwarding Concept

The next discussed mechanism is based on the realization that there are some places in the network where per-publication states cannot be avoided and these are the subscribers. Therefore an important concept may be to exploit this and head towards a solution where only the endpoints (subscribers) store states for the publications.

In many application-level multicast proposals, applications keep track of some other members of the same multicast group. This means that the routers in the network do not need to store multicast-related states. One protocol that inspired our solution is the application-level multicast implemented over CAN [Rat2001]. In this proposal a new mini-CAN is formed for every multicast group, and only the group members of G are the members of the mini-CAN for G. The delivery of packets destined to group G is achieved via a flooding mechanism in the appropriate mini-CAN. Although this paper introduces a mechanism using CAN it is clear that other DHTs can form the basis of this mechanism, e.g. Chord.

Apart from application-level multicast using CAN, the presented solution is based on Routing on Flat Labels [Cae2006]. In ROFL nodes have random unique identifiers. The routers represent their connected hosts by creating virtual nodes acting on behalf of the hosts. In the intra-domain case the virtual nodes form a Chord-like ring by storing source routes to their successors and optionally, to their predecessors and additional nodes. As an optimization, routers may cache all the source routes they are part of. The packets destined to a particular node id are simply forwarded to the id that is numerically closest but not past the destination id.

Now we describe shortly the basics of a clean-slate design that is based on Routing on Flat Labels [Cae2006] and the application-level multicast approach. We assume that the hosts in the domain form a global ROFL-ring, which is needed for building the mini-rings for the different publications. We also assume that the network is in a stable state, and the rendezvous bootstrapping process has successfully terminated, i.e. each sprouter in the domain knows where to forward subscribe and publish messages by knowing a next hop to a rendezvous node in the different scopes. One possible implementation of the rendezvous can be that it stores some members of each publication's ring and after receiving a subscription and checking if the publisher is allowed to get the publication it replies with a sprouter identifier where the ring can be joined. This particular sprouter will help the joining subscriber to build its forwarding table corresponding to the ring by finding source routes to its successor

and optionally to its predecessor and additional nodes (remember that the nodes have different neighbours in the different rings). Finding the paths can be implemented via sending control messages similar to those used in ROFL's network joining phase. One example is shown on Figure 8.10, where two different publications are assumed. The arrows on the illustration represent the knowledge of strict source routes.

This DHT-inspired solution does not utilize the transport efficiently, as publications do not traverse the shortest path. Also, as it uses source routes, there is some overhead in the publication's header. However, it effectively distributes the per-publication states among the interested subscribers and publishers, which makes this solution very attractive.

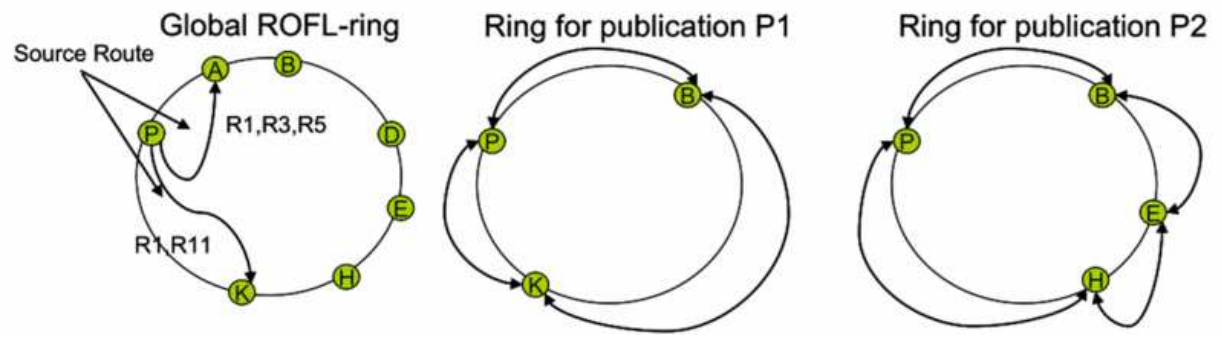


**Figure 8.10 – The Global ROFL-ring and per-publication rings**

## 8.4 Forwarding and Transport

The transport layer is traditionally responsible for providing communication services to end hosts and applications while using the basic services offered by the network. Usually, the transport layer is implemented on the end hosts only, much like the TCP and UDP transport services offered in today's Internet, while the network and lower layers are implemented by network routers and other specialized network elements. This design consideration of implementing the transport functions in the end hosts is largely driven by the end-to-end design principle ([Sal1984]; see also [Bel1976]) that lays an important foundation of today's Internet. We can usually observe that there will be multiple transports tailored to the requirements of different application sets.

More recently, the end-to-end principle has received criticism by e.g. Moors [Moo2002] and even Clark himself [Cla2007]. Hence, given today's more complex environment, one can fairly question how much the end-to-end principle applies any more. In this Section we consider different transport components and options for the PSIRP network architecture.

### On Transport in PSIRP

While traditional layering implies that the network and transport layers are implemented in different parts, such as the network and the end hosts, it is not clear that this model of network and transport would easily and naturally translate onto our PSIRP system. Endpoints in PSIRP have a very different role compared to traditional IP networks, moving away from dedicated communication (and transport) endpoints to rather temporary information providers and consumers, potentially impacting the implementation of transport functionality.

Furthermore, many of the design considerations in implementing transport layer functionality are driven by the end-to-end principle [Sal1984]; i.e., functions are largely implemented in the end hosts. Already our shift from E2E towards T2T ([Cla2007] and principle **G1** of our design) indicates that end host implementation of transport functions is not a given but instead

depends on the ability to perform certain functionality appropriately and in a trustworthy manner. Also, the observed potential recursive nature of information layering and reachability functions (such as implemented in the RTFM design choice [Sar2008] presented later in this document) questions the more traditional layering approaches and therefore the traditional transport-network division.

Hence, we focus our discussion on transport more on the proper implementation of the functionality itself without being bound to traditional layering considerations. This is aligned with our intention to follow a clean slate design, i.e., a design that questions all fundamentals, even the ones manifested in well accepted OSI models.

### On Transport Functions

The traditional transport layer can be characterized by a set of transport functions expected to be implemented. The goal of this discussion is to outline these traditional transport functions and consider a number of questions:

- Can such a transport function be provided by the PSIRP publish/subscribe architecture?
- What support for these functions needs to be provided by the network (e.g. congestion marking)?
- What particular features of our envisioned PSIRP network can support the implementation of particular transport functions?
- Should any of these functions be directly provided by the PSIRP network?
- What might be the reasons for providing these functions by the PSIRP network (rather than by the end points)?

This discussion is only the beginning of finding answers to these questions and it is expected that these questions will drive a substantial part of our design and implementation considerations.

### Error Detection & Recovery

Today's Internet is a best effort packet delivery system. As such, error detection and recovery are left to the transport functions in the end hosts. This decision allows the network to provide only basic common functionality, whereas reliability can be customized by different transport layers since different applications will have different requirements. In TCP error notification occurs through the receiver end host detecting missing packets, whereas UDP has taken the decision to not provide packet loss detection and retransmission since its intended applications are less sensitive to packet drops. Both provide a checksum for detection of packet corruption, but do not provide redundant encoding for automated error correction.

Although PSIRP could provide network level detection and correction of both packet corruption or dropped packets, the rationale for keeping these functions in the end hosts remains the same as for the original Internet, namely that applications will have different requirements, and to provide a generic customizable function within the network would be burdensome on applications that do not require high reliability. Considering a multicast transmission, it would be undesirable to retransmit packets lost by only one receiver to the entire set of subscribers. However, in this case an end host reliability protocol can perform redundant encoding to allow receivers to recover (in cases where errors are common or feedback from a large audience is inadvisable) or send the missing data direct to the receiver using a unicast transmission. It is also possible to consider shared retransmission channels which receivers can join to receive lost or corrupted data.

The outstanding problem with a publish/subscribe model is that the sender and receiver are decoupled. That is to say that the sender does not know whether to expect acknowledgment (or negative acknowledgement) from a receiver, while the receiver does not know when to expect data from a sender. For streams of data this is not so much of a problem since the receiver can often detect transmission losses in a timely manner through the subsequent delivery of further data. For the transmission of short or irregular data items it is often necessary to include some continuous heartbeat information so that the receiver can detect a loss of communication. This can be extremely expensive when considering sensor or event driven applications where the data may never be transmitted (for example an electric motor, monitored by a sensor, never overheats).

Some transports may be able to aggregate applications so that each may act as a heartbeat for the other applications, or provide a common heartbeat when this is not sufficient [Sop2003]. This is only possible with knowledge of the receiver population and the identifiers to which they are subscribed [Nek2003]. Hence, for some applications there remains an argument for having an error notification capability within the PSIRP network layer. A scenario that could justify such a capability would be a critical sensor-based application.

### Source Quenching

Multicast transport protocols sometimes include the ability for the publisher to detect when there are no longer any subscribers to their data. This allows the publisher to stop sending data to the network. To perform quenching just using the end hosts requires careful estimation and controlled feedback of audience numbers or the replication of subscriber join and leave messages to the sender. The problem with the former approach is the amount of traffic created and the risk of reply implosion. The problem with the latter approach is that the subscriber count information will undoubtedly become out of synchronization with the actual network subscriptions. There is also the additional problem of not being able to feedback the join notification until some data has been sent with the sender details. This is especially a problem if the join notification is for the first subscriber and intended to start the publication process. In this case some common subscriber registry can be used to provide a decoupled back-channel. Furthermore, network load may also be a problem if a large number of subscribers are joining and leaving regularly.

A solution to these problems is to provide a quench notification facility within the network. This makes some sense since the subscriptions already exist within the network and all that is required is the final subscription notification to any potential publisher. Network load can be controlled since the subscriptions can be aggregated into a single some/none subscriber trigger.

### Multiplexing & Ordering

Other services that may be considered are the multiplexing of data from different applications, and the preservation of packet ordering. If multiple applications on an end host are sending to the same rendezvous identifier in PSIRP, an end host transport facility can combine such communications into longer transmission units (or fragment communications that are too long). This appears to be a natural function between the application and the network on the end host. However, the network may perform further multiplexing, especially when the data is transmitted over different link and physical mediums that use different transmission lengths. The network may also multiplex data from different end hosts going towards the same receiver, or even along the same network path, de-multiplexing these flows at a later point.

The current Internet uses the concept of Ports in addition to the IP address. This allows an application to identify a set of applications at the network location. This explicit multiplexing by the applications should not be required within PSIRP, since each application is free to use its own set of identifiers (or share them as desired).

Transport functions such as TCP also provide delivery of packets to an application in the same order as they were transmitted, regardless of whether packets have been delayed in the network, or retransmitted due to corruption or loss. Again, the argument is that not every application requires such capability, and it is easier to achieve on the end host (especially in combination with retransmission).

Considering a concast communication from multiple hosts the network could attempt to provide an ordered delivery service. However, it remains unclear how many applications might benefit. Such a service could also be provided by a host transport function using synchronized clocks, although this may not be as accurate as ordering provided within the network (checked at intermediate nodes).

### Duplex

The transport can use a sender address to set up bi-directional communication. For PSIRP, instead of a sender IP address, the sender may include a further identifier for return communications. This has the flexibility that the return identifier may be subscribed to by additional or alternative hosts than the original sender. If this return identifier is used only by the receiving host, then it may be considered as part of the transport function.

Alternatively the network may support duplex communication directly. One approach to achieve this is for the network to mark the outbound packets with return path routing information. This approach avoids building state within the network routers which would make them vulnerable to overloading and Denial-of-Service attacks.

### Connection Orientation

The end host transport may provide a communication model that is connection oriented (over a packet switched network) or vice-versa. Once the connection is established the application may omit much of the original control data. An end-host transport for PSIRP could also provide similar functionality.

### Flow Control & Congestion Avoidance

Flow control can be achieved by duplex communication between the receiver and the sender. In this manner the receiver can ask for the flow to be reduced either due to its own inability to process the data, or due to congestion in the network (that is notified downstream to the receiver explicitly through mechanisms such as ECN, or through noticing packet loss). Work on re-feedback [Bri2005] allows the sender to control the congestion they are prepared to cause, which is estimated from the receiver feedback.

Although this approach works well for longer unicast sessions, both very short communications and multicast pose their own problems [Bri2004]. Very short sessions (e.g. single packets) are penalized since the sender does not have any accurate congestion information. For multicast there is the problem of how to allocate congestion across multiple receivers, combined with the signaling of which receiver should send feedback to the sender (to avoid implosion and receive regular and timely congestion updates).

There is also the underlying economic problem of just who is responsible for the congestion. Is it the sender (who is clearly responsible for sending the data), or the subscriber (who is responsible for the routing path)? This question determines who needs to receive congestion information in order to terminate or reduce the flow. It also determines the attacks that we are likely to see. For example, if the sender is responsible for congestion, but is getting feedback from the receiver, there is clearly an opportunity for a malicious receiver to cause the sender to generate congestion. This particular example is somewhat mitigated by the re-feedback technique of requiring the sender to declare the expected congestion.

Some of these problems can be countered with a network congestion pushback service [Ram2001]. One advantage of this system is that it can potentially provide congestion estimation even before the sender publishes to a particular identifier. However, this approach has its own set of problems. Any implementation must find a way to communicate congestion information back towards the sender. If information is tunneled the origin will be hidden from the intermediate network nodes. Thus congestion notification will need to be fed back hop-by-hop incurring significant delays.

There is also a concern over the amount of congestion notification traffic that may be produced, particularly during times of congestion or Denial-of-Service attacks. Host based solutions can provide feedback in their usual acknowledgement messages, which potentially (combined with multi-path routing) can flow over different routes. Finally, the reliability of the congestion notification service must be considered if the network does not also offer a native reliability service. This is not a problem for end hosts who can provide reliable feedback.

#### Multi-path Selection

A dominant recent interest in future internet forums has been around the concept of multi-path routing, and the combination of path selection with congestion control algorithms [Key2006]. To perform path selection the end hosts and any other routing control points need to be aware of available paths and the congestion on each path. To improve the results of the path selection, each session should not be considered separately, but as part of a combined optimization problem.

Considering a multicast network it is possible that subscriptions to a rendezvous identifier form multiple delivery trees with some (probabilistic) guarantee of separate routing paths. This path redundancy can be applied both for the routing of subscriptions and potential publisher advertisements over the rendezvous system, along with the forwarding paths that are set up in the forwarding network as a result of such subscriptions and advertisements.

## 8.5 Caching

In this section, we examine two caching models that are our starting points for a PSIRP-aware caching service. We discuss the realization of caching both in the PSIRP network, and also as an external service that is executed on top of the network. The first model primarily consists of a content-aware item caching service, where the client application requests specific items of content. The second model implements a replay service for subscribers to an identifier. In each case we describe an application overlay example with minimal impact on the network interface and service, and a second model with cache support within the network.

### 8.5.1 Content Caching

#### A) Overlay Content Caching Using Multicast or Anycast Publish/Subscribe

A typical content delivery overlay can be implemented through the original source(s) of content subscribing to a content advertisement and obtaining a PSIRP rendezvous identifier. Any client interested in the data represented by the identifier can send a request to the sources of the content. This can be achieved through an anycast transmission (for example to the nearest content server). Alternatively a multicast transmission can be used with a limited Time-To-Live or probabilistic reply function (to avoid implosion). In the anycast model the network is responsible for selecting the best content server, while in the multicast model the

client receives invitations to deliver the content, and is then responsible for selecting one or more content servers to engage in the actual content delivery.

If the identifier is associated with a unique item of content, then a simple trigger request is sufficient to indicate that the client is interested in receiving the content in return. If the identifier is more general, perhaps representing a set of content items or a series of versions of the same content item, then a more expressive request can specify exactly what the client is interested in receiving. If this more expressive content request is used then typically a multicast communication model should be used since any single server may not have exactly the required items of content.

The selected content can be returned back to the client through an indicated return identifier, or alternatively the network can establish a return path.

Since the client is in direct two-way communication with specialised content servers or caches, the client/server can implement specialised protocols for the delivery of the content (for example with flow control, real-time or reliability properties). If the anycast communication pattern is used, either a session needs to be established to ensure that future packets from the client reach the same server, or an alternative identifier unique to that content server needs to be communicated to the client and used on subsequent (e.g. flow control) packets to the server. In the multicast model this is not necessary since initially only the ability to serve the content is returned, and then direct communication can take place with the chosen content server.

To achieve caching the client request must be directed through a local caching proxy. As the content is returned to the client, the cache will maintain a copy of the content and register with the same identifier that the client specified in the content request. Each caching proxy should subscribe to an identifier that represents all caches corresponding to a certain function (for example BBC video content cache). The client knows the category of information it is interested in, and retrieves the appropriate cache function identifier. It then tunnels its content request within a communication to the cache proxy. This communication can be performed using anycast to select the nearest cache, or alternatively caches can be polled using multicast.

Within the cache the content request is extracted and submitted towards content servers and caches already serving the content. The content is delivered to the cache proxy that can maintain a copy as it serves the content onwards to the client application. If the client does not choose to route their request through a caching proxy, then the content will not be cached unless the content servers or other caches directly push content to other servers. There is a conflict of interest since the client is unlikely to request the same content and benefit from the caching of that item. Although it desires to use available caches, it may do so directly without the use of the local cache proxy (unless these are secured).

### B) PSIRP Network Content Caching

As before the content servers subscribe using a content advertisement and obtain PSIRP rendezvous identifiers for each unique item of content. The client obtains the identifier of the item of content that it is interested in receiving and sends this as an anycast publication to the identifier. The network anycast capability routes the request to the best (for example nearest) content server or previously subscribed cache.

The request collects a return routing directive as it is forwarded across the network. This is used by the content server to send the specific item of content back towards the client. Alternative a reply identifier can be used.

Since there is no guarantee that any cache sits along the return path, the content server needs to signal to the network that the communication is a cacheable item, along with the identifier that the item is to be subscribed to. The cache indicator is used by domains along

the reply path to redirect the communication through caching points in the network. As the data flows past the cache attachment point it is mirrored into the cache. The cache reassembles the content item, and then subscribes to the identifier indicated in the caching directive.

Common content server and cache protocol functions such as flow control can be achieved by forming a session to ensure that future flow control packets are routed to the same server via the anycast mechanism. Alternatively an alternative flow control identifier can be returned to the client in a packet prior to the content delivery.

More specific caching intelligence for particular types of content or ranges or clients is difficult, but not impossible to achieve. For example a domain can implement more specialised cache functions, and the caching directive can indicate the nature of the content being delivered.

The caching directive can be considered metadata that forms an instruction to both the network routing and cache functionality. Such metadata may be delivered to the network "in band" along with the content. Alternatively the network may be pre-configured. However, since the delivery routing path may be configured dynamically, or be used for multiple content items (for example when a return rendezvous identifier is used by the client), the in band approach appears to be more suitable to control caching behaviour.

### 8.5.2 Subscription Channel Caching

#### C) PSIRP Channel Caching and Replay

Models (A) and (B) both provide content replication ability. In contrast this model caches transmission units or packets delivered over a rendezvous identifier. Hence this service model is very different. It is not used to more efficiently retrieve items of content that are available through other means (e.g. from the content servers). Instead it provides a replay of communications previously sent over the rendezvous identifier. Since these communications can be from any application, they are not necessarily retrievable by other means (e.g. a sensor value). Although this could be considered a more general ability, this model is also more restrictive in other ways. Notably the packets are not associated to a content identifier separately from the delivery mechanism. The packets are instead only available through requests concerning the same rendezvous identifier over which they were originally delivered.

In this model there is no concept of a client requesting a content item. Simply, any data published over a rendezvous identifier may be cached. This can result either from requests by the publisher, or by the subscriber. For example, the subscriber may indicate that it wishes a certain time period or number of packets to persist within the network. Alternatively a publisher may indicate such instructions since it has knowledge about its subscriber base (i.e. the application that is served by publishing to the rendezvous identifier). Ultimately the decision is up to the network cache.

Since the cache is only ever applied to the rendezvous identifier delivery channel, the metadata that controls the cache may be configured for longer durations than single or multiple packet sequences (such as an item of content). Such cache metadata may be delivered dynamically in band from a publisher, for the lifetime of a subscription from the subscriber, or for some arbitrary period by the cache operator.

When a client subscribes to a rendezvous identifier, it is electing to receive data published in the future to that rendezvous identifier. Additionally the subscriber can ask to 'catch up' on information it has recently missed previous to its subscription. This catch-up request is relayed back into the network as part of the subscription. Where the subscription joins the existing subscriber multicast tree, or traverses parts of the network where a previous subscription existed (for example due to subscriber drop out), a cache may exist. The first cache to see the

catch-up request will start to transmit the cache data to the subscriber in addition to any current publications sent over the rendezvous identifier. Since not all downstream subscribers will desire to see this catch-up information (since many subscribers will have seen it during the first transmission), the cache information should be sent over an additional rendezvous identifier specified by the subscriber during the catch-up request.

It is still possible in this model for the publisher to indicate to the network meaningful items of information for caching. This may be achieved through cache metadata indicating which packets form items of content. Unless all packets within an item are seen, the packets are not held within the cache. Note that the metadata will only identify packets sent over the same rendezvous identifier. The problem with this approach is that subscribers will not be able to retrieve selected packets that have been missed during handovers or connection drop outs.

**D) Overlay Channel Caching and Replay**

If we do not wish to add functionality into the network as part of publish and subscribe operations then the sort of caching described in (C) can be provided by application level servers. Either the client will tunnel its subscription through such application caches, or alternatively caches may simply subscribe to rendezvous identifiers irrespective of client requests. Any client wishing to receive catch-up data will identify a relevant cache for the rendezvous identifier (using application level search or directory services) and contact the cache directly on a cache application identifier to retrieve the missed data. In this case any publisher cache instructions are contained within the application payload.

## 8.6 Security Policies and Access Control

We have already discussed that the final dissemination of the information across the network is controlled by several mechanisms. The publication and subscription concerning a rendezvous identifier and scope form the primary mechanism. This may be influenced by other policies expressed as network metadata. For example the choice of caching will determine that the traffic flows past an appropriate caching facility, whereas a network peering policy will determine which inter-network paths are used. Security policies may also be expressed by several parties to determine the overall information dissemination. The publisher, subscriber, network operator and other parties to whom authority is recognized (e.g. government) may determine security policies that affect the flow of the information in the network.

Security policies may be used to achieve confidentiality goals - for example to stop business information leaking to competitors. Although information can only be received by subscribers to the correct identifier, for many applications this will not provide adequate security. As a simple method of controlling access, any application is also free to encrypt the payload of the data it sends over the network. Only receivers possessing the correct cryptographic keys will be able to decode and subsequently access the information. This form of scoping is controlled by applications and inherently external to the network. We may also consider that the network may participate in the control of subscribers by requiring some form of authentication and access control. The policy itself may be transmitted along with the information or provided via separate control metadata.

In terms of integrity, subscribers to information will want assurances that the information they receive has not been tampered with and comes form the expected source (not necessarily the publisher to the network). Again, end-to-end cryptographic mechanisms can be used by the application to sign the data. However we can also consider network level mechanisms to authenticate the publisher and potentially control access to the publication operation for a given rendezvous identifier or scope.

Subscribers will also want assurance that the information sent to a particular rendezvous identifier has a particular well understood meaning and format, and thus will want to restrict the communication channel to those parties that it trusts to adhere to such agreements. These receiver-controlled mechanisms can also provide a layer of defence against denial-of-service attacks where restricted knowledge of rendezvous identifiers and migration between rendezvous identifier will not suffice.

We can imagine three valid control options for sender access control:

1. where all senders are authenticated and access-controlled,
2. where no access control is implemented by the network, and
3. where access control is implemented at the request of each receiver.

In case #3, it would be acceptable for different receivers to select different options. This may result in the same information being transmitted over multiple channels: some where we trust that upstream access control has been performed, and others where we have to implement access controls closer to any receiver that has requested them.

At the present stage of the project, it is unclear which of the above options are actually implementable in a scalable manner. We expect that our implementation and simulation experiences within the next phases of the project will limit our choices to a reasonable degree of convergence.

## 8.7 Network Attachment

This section provides information about the process of network attachment in the context of PSIRP. We identify design considerations and requirements related to this area, and also present an example of a network attachment protocol for PSIRP.

When a node joins a PSIRP network, its goal is to use (PSIRP) network services on top of a data link layer connection. Essentially this means publishing to and subscribing information in the network. Therefore, the node needs to bootstrap communication using an attachment procedure, which is based on the basic functions provided by the underlying (publish-subscribe based) PSIRP network.

Network attachment may include the following functions (from the perspective of a node that joins a PSIRP network):

- Detecting another node, presumably a *sprouter* (a PSIRP router), that can be used as a point of attachment, and finding out about its capabilities. Based on e.g. knowledge about the services offered and policies mandated by the other node, the joining party can make the decision of whether to attach to it or not.
- Obtaining authorization to use the network. This may require that the node is able to offer compensation for the services it intends to use. Options here include that a node negotiates a contract for using services, or that it authenticates itself to the attachment point and refers to an existing, possibly implicit, contract.
- Exchange of configuration information, e.g. rendezvous identifiers and security related parameters.
- Establishing a control channel with the attachment point. Such a channel can, if needed, be used for further negotiations and data updates.

We anticipate having at least one concrete, generic network attachment mechanism, including necessary protocols, designed for PSIRP. In designing this kind of architecture, issues such as security and mobility, among other things, need to be addressed.

Figure 8.11 shows a draft of a network attachment protocol with three partially overlapping phases. It provides an example of how some of the functions outlined above can be

implemented. It also shows how the network attachment functionality can be designed as a special application module on top of the RTFM architecture [Sar2008]. The protocol example is explained below.

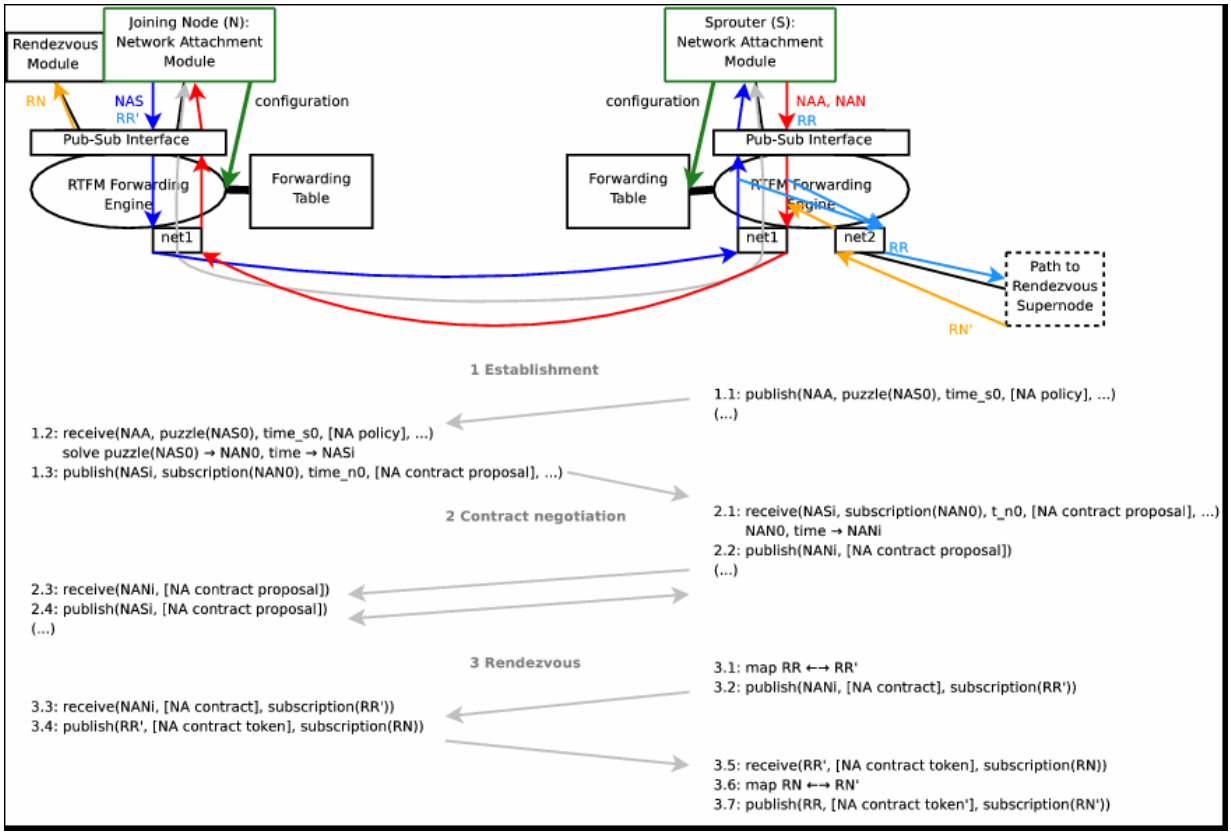


**Figure 8.11 – Network attachment**

In the first phase the communication is bootstrapped. Initially a new node (N) that joins a network starts listening for link advertisement messages (known to have a static identifier *NAA*) on the link (i.e. it subscribes to NAAs). These advertisements are broadcast (i.e., published) by sprouters (S) on the same link. An advertisement contains a cryptographic puzzle, which requires some amount of computational resources (possibly none) of N in order to be solved before the sprouter creates a state for the client. The solution of the puzzle contains parameters for generating dynamic rendezvous identifiers (NASi) that S subscribes to. When N learns NASi, it can publish a message that contains a subscription to an identifier NANi. Thus a communication channel, used for further negotiations, has been established between N and S.

In addition other data, such as cryptographic keys and authentication data, as well as service and compensation policy information, can be provided in the advertisement and other initial messages exchanged between N and S.

In the second phase N and S need to negotiate a contract, which enables S to provide services to N based on a compensation scheme. This negotiation happens in an *offer-answer manner*, i.e., N proposes a contract to S, which either accepts the proposal or creates a counter-proposal. Either party may also terminate the negotiation if an agreement is not reached. When S accepts a contract proposal, it also authorizes N to use services in the network on the terms of the contract.

Finally, N and S start the process of intradomain rendezvous. N learns a rendezvous identifier (RR') that a supernode subscribes to, or one that S may have switched (RR → RR').

Additionally, N may also indicate that it itself subscribes to a certain rendezvous identifier (RN), which may also be subject to switching at S (RN → RN').

# 9 Design Choices

In this section we present two realizations of the conceptual architecture, namely the RTFM and Black Box designs. We consider the different parts of the designs and how they relate to the conceptual architecture.

## 9.1 RTFM

In this part, we will describe the essential parts of a pub/sub network. How a host can join a network, publish data, and subscribe to publications, the requirements for the pub/sub routers (sprouters), how to find a route to a publication in the case of the subscribe primitive, and what happens when a host decides to publish using the publish primitive.

To accomplish this, a pub/sub network gives each host only two network primitives: *publish* and *subscribe*. When a node publishes data, no data transfer actually takes place (the rendezvous system is informed of its existence). Only when a node subscribes to a named piece of data, the network finds the publication and creates a delivery path from the publisher to the subscriber.

Publications have two identifiers: a *private identifier* and a *public identifier*. The architecture does not restrict how the identifiers are formed, but it is a good idea for the identifiers to be cryptographically bound together. The idea is that the private identifier is known only to the publisher(s) of the publication, and the public identifier is used to subscribe to the publication. The private identifier can be created by e.g. hashing the content of the publication data or metadata and a signature created with the private key of the publisher. The public identifier can be e.g. a one-way hash of the private identifier. Having these two identifiers per publication enables a) the publisher to prove it "owns" the publication b) the subscriber to check the validity of a received publication.

Obtaining the identifier calls for a *directory service*, where long lived publications may have an entry which maps a human readable name into the public identifier. This is similar to DNS which maps domain names into IP addresses. A notable difference to the current architecture is that the directory service lists names and IDs of *data* objects and not endpoints or locations of data. It is the job of the rendezvous system to match the subscription to a publication, and this can happen in more than one network location.

The network architecture is composed of three modules: Rendezvous, Topology, and Forwarding (and the physical Media). Everything in the network, starting from network attachment, and including how the above mentioned modules are built up, is done using these primitives. The following discussion describes the system in a steady state situation.

### 9.1.1 Forwarding

Forwarding is used to actually deliver data from one location to another. It is based on label switching, i.e. each packet has a label (or a stack of labels) and a forwarding table as shown in Table 9.1. Label is a bit string in the packet that is used by the routers to make forwarding decisions. Port is a local numbering (internal to the host) of the different next hops that the packet can be forwarded to, including the wired and wireless network interfaces to the next hop sprouter, as well as other software modules (such as applications) internal to the host in question.

Labels can be stacked as shown in Table 9.1. The first row shows a packet labeled X will be forwarded on port 1 and a label A is appended to it. The second row shows the branching of a multicast tree Y. The last row shows a packet from which the label Z is popped from the stack

and the packet forwarded to port 3. * denotes any label. Port 3 can for example be a loopback, i.e. returning the packet to be processed by the forwarding table after Z has been popped from the label stack.

| Incoming label | Outgoing port(s) | Outgoing label(s) |
|---|---|---|
| X | 1 | AX |
| Y | 1<br>2 | Y<br>Y |
| Z* | 3 | * |

**Table 9.1 – Forwarding table**

The system may, thus, deliver a given packet labeled X in one location to multiple destinations. We call the route that the packet travels from the insertion of a label to the destinations a delivery tree (and chose against using the term multicast tree, because of its IP architecture connotations).

### 9.1.2 Topology

The main purpose of the topology module is to create and maintain delivery trees used for forwarding traffic. It acts both proactively and reactively. It proactively creates delivery trees that may be needed and reactively constructs new trees, when existing delivery trees do not provide the necessary connectivity, paths break down, or the combined multicast tree is "too" suboptimal.

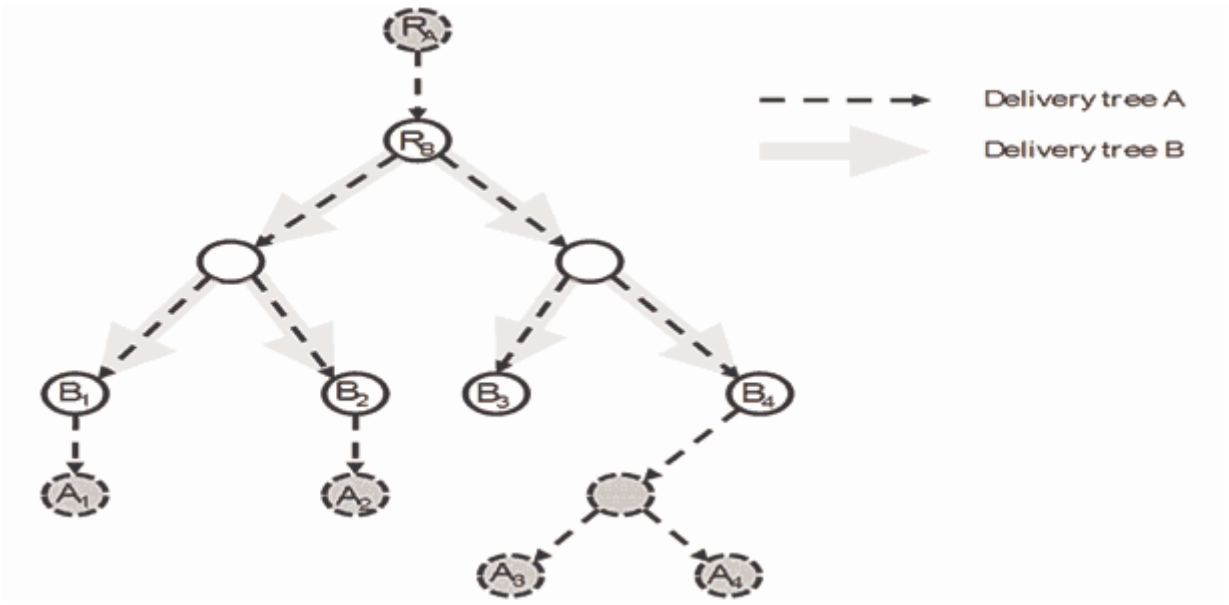


**Figure 9.1 – Combined delivery trees**

The delivery trees are constructed both via individual lower layer links and by combining existing "lower layer" delivery trees. An example is illustrated in Figure 9.1 above, where delivery tree B is being used by delivery tree A to forward data from the tree's root node $R_A$ to the leaves $A_1..A_4$. It can be noted that the multicast tree B leaf node $B_3$ also receives data sent in multicast tree A. This example presents a suboptimal solution as $B_3$ receives data that might not be of interest to it. This could be optimized by creating a new delivery tree C that $B_3$ does not subscribe to. However, creating new delivery trees requires new entries in the forwarding tables, and it may be more optimal in terms of resource use to use an existing suboptimal delivery tree than to always create a new optimal one.

### 9.1.3 Rendezvous

When a node subscribes to a publication, the network must first find a copy of it. The rendezvous system is a distributed structure that provides this function. It can be e.g. a Distributed Hash Table, or a semi-hierarchy, such as Data Oriented Network Architecture (DONA) [Kop2007]. It uses the distributed structure to route to a copy of the wanted data and gathers enough information on the way for the topology system to identify the delivery trees needed to forward the actual data to the subscriber.

## 9.2 Black Box

We develop a conceptual view of a potential information-centric system architecture that is based on a **black box** which is built recursively upon to assemble the higher level complexity in semantics and scope.

*In a sense, a black box represents a subnetwork which hides its structure from other subnetworks, implementing the recursive nature of reachability scope.* [Tar2008]

As an analogy, we may consider a network that performs Network Address Translation (NAT) at each router, thus hiding the network topology, as being such black box based system.

An important consideration has to be placed on the point where rendezvous functions, in particular, are implemented in our design. In Clark's refinement to the original E2E principle, he outlines the importance of trustworthiness of execution rather than particular placement of functions in endpoints [Cla2007]. This so-called Trust-to-Trust principle (captured as our design principle G1) is considered in our design in the sense that rendezvous should happen where the network is trusted to operate correctly in terms of communal, economical, and functional requirements.

We argue that this kind of black box network design has favourable properties, namely support for flexible network deployment, and enhanced security and privacy. On the other hand, additional network management tools are needed. We note that certain diagnostic packets can be implemented that tag through which routers a packet was delivered.

### 9.2.1 Internetworking

Internetworking is a crucial feature that is responsible for bridging the various black boxes and connecting the subscribers and publishers of information. From the view point of a single publisher or a subscriber, we can define two important directions for data, namely the direction of control messages originating from a receiver (subscriber), and the direction of data originating from sources that is routed towards interested receivers. We note that control messages can also be used for the publisher to optimize routing. It is necessary that any correlated subscriptions and data advertisements from publishers meet in order to be able to

build distributed forwarding state over the network. We call this meeting in the middle rendezvous and it is a core component of the proposed architecture.

Each black box has a number of rendezvous points, which are managed control points for the black boxed network. Each network exposes certain advertisements, metadata, and enforces policies on inbound and outbound traffic. Before we define the role of the RPs further, we briefly consider the NIRA architecture.

NIRA (A New Inter-Domain Routing Architecture) empowers users with the ability to choose a provider and domain level end-to-end path [Yan2007]. The motivation for this is that only users know whether a path works or not. This model creates competition between paths that different ISPs offer, because users can choose the most suitable paths. This work emphasizes policy-based routing and defines the valley free domain model. In this model, the network comprises of three parts for each sender and receiver, namely the core region (tier-1), the uphill region that covers all possible paths from the sender to the core, and the downhill region covering all possible paths from the core down the receiver. Each region can have its own routing protocols.

We borrow the notion of the valley free domain model from NIRA and propose that forwarding paths between subscribers and publishers are calculated between the RPs and the core region.

Our model is recursive in the sense that a black box can contain black boxes. Hence we can have nested RPs. RPs have a full mesh towards the core network (towards the previous level of recursion). The key idea is to use RPs as control points to build distributed routing and forwarding tables. RPs also act as policy enforcement points. They can be characterized as intradomain RPs and interdomain RPs, respectively. Later in this report, we show how the completeness of subscriptions or data advertisements to RPs can be used to characterize the system and optimize its behaviour. Figure 9.2 illustrates the recursive nature of black boxes and presents the rendezvous point mesh.

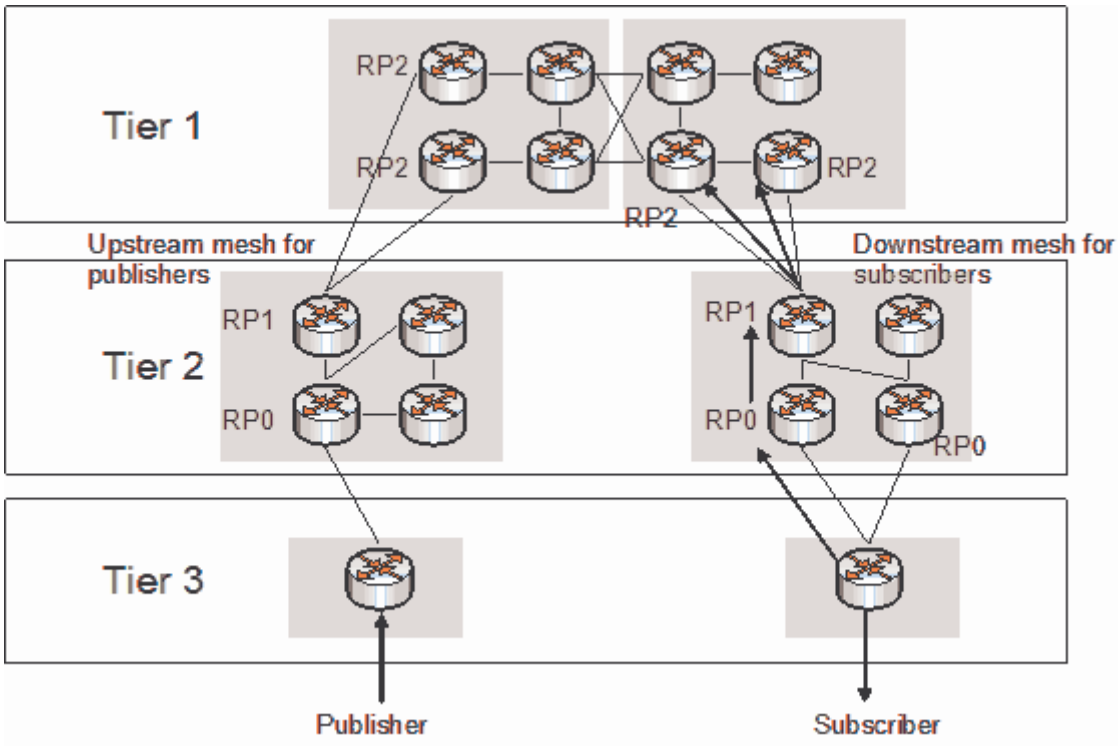


**Figure 9.2 – Valley-free model using rendezvous points**

### 9.2.2 Rendezvous Function

Rendezvous is about offering the flexibility of defining (information) reachability. This is achieved through a cascade of rendezvous functions, each for a specific (semantic) abstraction level. Each rendezvous function resolves into an RP, which provides a convenient way to be able to support extensibility of the system, for example in terms of networking policies.

Our definition of the rendezvous function (*RZV*) is as follows:

```
RZV(X,D,P) {
Let set X ← resolve(X)
Let set D ← rewrite(D)
sendToRP(X,D,P,RZV)
}
```

Here **X** is the network address of the next RP, and **D** is the data label or description which is the basis of routing and forwarding decisions. Both **X** and **D** are sets, and **P** is the packet payload. A packet is a triplet **(X,D,P)**. The *resolve* function resolves to the next rendezvous points. The *rewrite* function performs necessary rewriting of the data label **D** for sending the packet comprising of the triplet **(X,D,P)**. The *sendToRP* primitive is a unicast, anycast, or multicast function that transfers the packet to the next logical RP **X**, where it is processed by an *RZV* function.

Since our work assumes data and interest centric operation (according to principle **P4**), we assume that the data packet description **D** is a label or a semi-structured descriptor. Our aim is to be able to support various data centric communication mechanisms using the same rendezvous function approach. The system must have a base rendezvous layer, which we denote *RZV_0* that ensures basic connectivity for data packets. Then building on top of this, the idea is to introduce label-based forwarding, and ultimately content-based forwarding.

### 9.2.3 Flat Labels and Scopes

Flat self-certifying [Gir1991] labels seem to be a natural choice for a data oriented architecture. Recently, many systems based on globally unique flat namespaces have been proposed, for example UIP [For2004], TRIAD [Che2000], Nimrod, i3 [Sto2002], DOA, ROFL [Cae2006], and DONA [Kop2007].

We propose to utilize flat labels in such a way that each packet identifier has two parts, namely *scope* and *data description* both being flat labels, equivalent to the notion of scope and rendezvous identifiers as introduced in Section 6. With this, the scope allows for the definition of subscription and publication context, effectively implementing our design principle **PS2**. It also enables better aggregation of labels for recursive scoping. As an analogy, a scope could be equivalent to a domain name, and the data description an identifier of a particular web page.

Labels need to be long (e.g. 256 bits or more) to ensure statistical uniqueness and to be able to embed security features. Most labels should be self-certifying in order to ensure that the source of the data is reliable. Metadata may be associated with labels and should be self-reflecting, in other words include information on how it should be interpreted. Some control traffic will have metadata. Metadata is extensible and not all parts need to be understood by network entities.

### 9.2.4 Completeness

The completeness of data subscriptions and data advertisements is given by Definition 1. This formulation is flexible enough to be useful for various routing protocols. Completeness may be used to characterize the whole routing system. In addition, it may also be used to characterize a part of the routing system, such as a *path*. If a graph, subgraph, or path is not complete, then it is *incomplete* [Tar2007].

*Definition 1: A data advertisement **A** is complete in a network system **PS** if there does not exist a router **r** with a data subscription that has not processed a matching **A**. Similarly, a data subscription **S** is complete in **PS** if there does not exist a router **r** such that **r** has a data advertisement that matches with **S** and **S** is not active on **r**.*

### 9.2.5 Rendezvous and Completeness

We now extend the rendezvous function in such a way that it is easy to check whether or not part of the rendezvous function has successfully completed. This is accomplished by assuming that the rendezvous function maps to a tree of rendezvous points (RPs). This is given by Definition 2. Figure 9.3 illustrates packet and message exchanges in the rendezvous process.

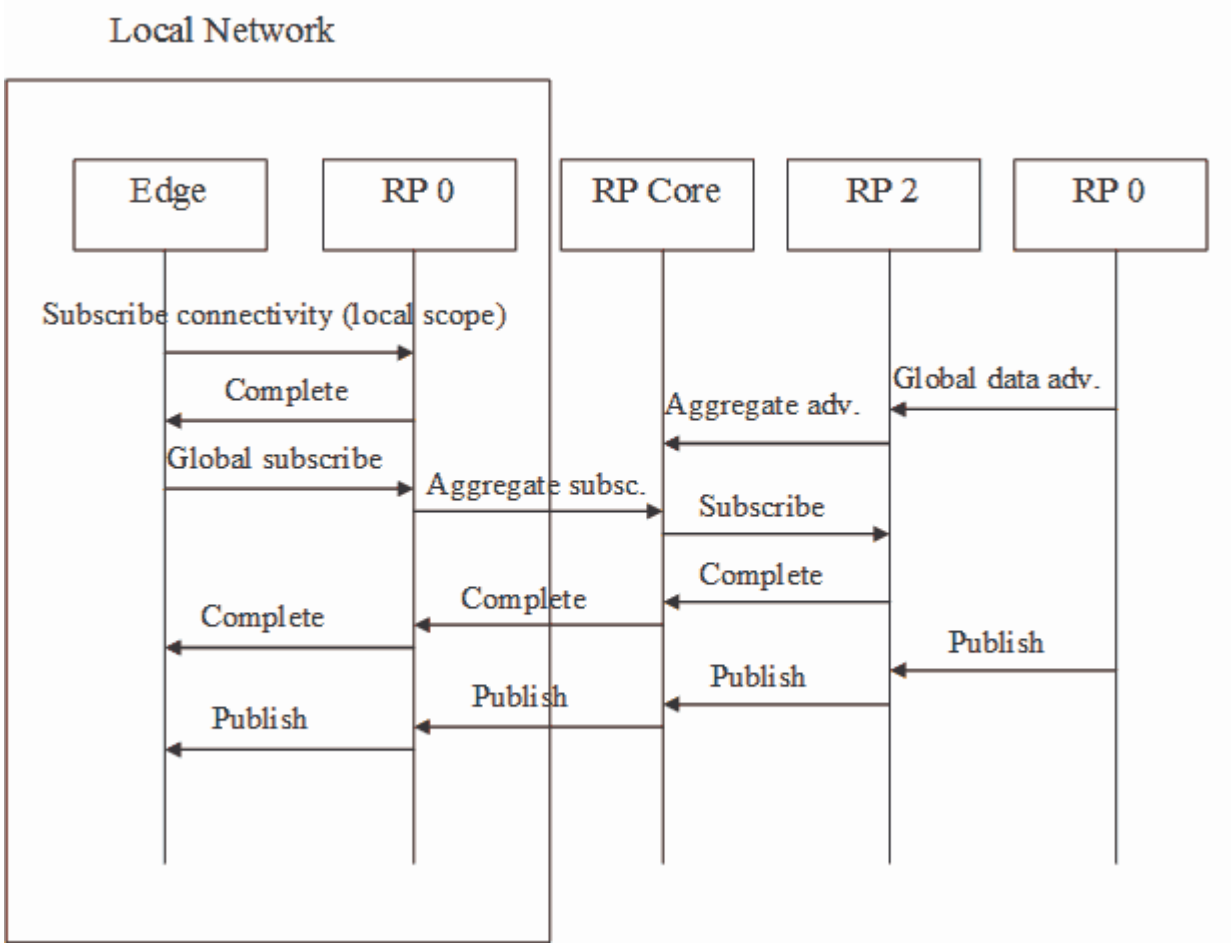


**Figure 9.3 – Example interactions**

*Definition 2: A distributed rendezvous function is complete if the signaling paths to the RPs identified by the function are complete according to Definition 1.*

Our definition of the rendezvous function with completeness checking (RZVC) is as follows:

```
RZVC(X,D,P) {
Let set X ← resolve(X)
Let set D ← rewrite(D)
Let set I ← incomplete(X,D)
Forall x ∈ I: sendToRP ({x},D,P, RZVC)
}
```

The function *incomplete* is defined as follows:

```
incomplete(X,D) {
Let set I be initially empty
Forall x ∈ X: If D is active on x and
completeness ack has
not been received then I ← {x}
Return I
}
```

If a part of the rendezvous process is complete with respect to the input parameters (and any modified parameters during the process), it is not necessary to continue the rendezvous process further.

We observe that the changes in complete parts do not require any interaction with the client; the incomplete parts are the most interesting from the network topology viewpoint.

Moreover, the number of nested RPs gives a natural metric for the topology update process for a particular data item **D**. This number, denoted by $h(\mathbf{D})$, can be used to gauge the level of completeness of a logical space of the information network. The localhost is defined as $h(0)$ and the local area network as $h(1)$. In addition to completeness, it is possible to determine the Round-Trip-Time (RTT) of $h(\mathbf{x})$, for some **x**. This can be seen as a basis for network diagnostic functions.

```
CMPL (X,D) {
Let set X ← resolve(X)
Let set D ← rewrite(D)
Let set I ← incomplete(X,D)
If I ≠ Ø then return INCOMPLETE
return COMPLETE
}
```

### 9.2.6 Impact of Mobility and Topology Updates

Mobility support can be realized on different levels in the network protocol stack and for different types of endpoints. In this section, we consider the physical mobility of publishers, subscribers, and networks. Such mobility support, implemented in a handoff mechanism, needs to be supported by the network. This requires that the routing and forwarding topology is updated accordingly during and after handoffs.

A simple way to cope with a handoff on the subscriber and publisher level is for subscribers to re-subscribe to those publications they are subscribing to, and for publishers to re-establish their publications at the new location. The rendezvous service is responsible for maintaining the proper set of rendezvous points and multicast forests.

A similar mechanism can be used for router and network mobility, although in this case the function of re-subscription and re-publishing becomes a function of aggregating a set of publications and subscriptions. With this, the mobile router that receives a subscription from its mobile network becomes a rendezvous point for the nodes it serves. When the mobile router moves, it simply updates the publish and subscribe requests to the rendezvous system, which updates the forwarding tables in the network to deliver data to the new location.

The main problem then becomes how to optimize the rendezvous system for this. We observe that if the data labels are already subscribed or published in the new network, the handoff is complete.

### 9.2.7 Discussion

The novelty of the proposal stems from the combination of black box based network model and the use of recursive rendezvous processes on multiple logical layers in combination with publish/subscribe primitives. The aim of this mechanism is to better support current data intensive network applications, such as YouTube, Flickr, many “mashups,” and many web pages that are frequently updated. The approach is motivated by the need to support many different kinds of network technologies, including DHT-based overlays and new kinds of networking solutions, such as all-optical GMPLS core networks that are based on label switching. The label-based approach might yield performance improvements also in broadcast networks such as wireless access networks.

Further work is needed to understand the scalability properties of data-centric networks with recursive rendezvous. We note that various probabilistic techniques can be used to aggregate flat labels, which typically increase the false positive rate. It has been shown that the Internet follows power-law distributions both at the router level and AS level. This means that the physical fabric of the Internet and the business interconnection of networks can both be considered to be scale-free networks.

# 10 Application Considerations

## 10.1 Communication Models

Although the network presented within PSIRP is based upon a publish-subscribe model, a wide number of different communication paradigms have to be supported to meet the needs of diverse applications (including all those that currently operate over today's IP-based Internet). In this section we describe several common communication paradigms and show how these can be implemented using subscriptions and publications to network level rendezvous identifiers. It should be noted that there is no single way to implement the communication models discussed below. Our intention is to show that such models are possible to implement in principle, not to suggest the best way to implement them. This will vary between applications and we expect many different implementations to co-exist.

### Request-Response

Implementing a request-reply communication is straightforward, and explained in previous works such as I3 (Internet Indirection Infrastructure) [Sto2002]. The server application will subscribe to a rendezvous identifier over which it expects to receive requests. This identifier may be uniquely used by one server, or may be shared across multiple servers. In these latter cases, requests may be multicast to many potential servers, or may be anycast to the best (nearest) server.

The rendezvous identifier over which requests are to be sent is shared with potential application clients. This can be achieved through previous application communications which embed the request rendezvous identifier within their communications, or more specifically through dedicated directories or similar discovery services (e.g. search engines).

The client application will send the request to the indicated rendezvous identifier, which will be received by one or more previously subscribed servers. Any servers wishing to reply must send a response back to the client. One way to achieve this is for the client to embed a response rendezvous identifier within the payload of the request. The client will have already subscribed to this response rendezvous identifier and thus will receive any response from the server(s).

The following figure outlines a potential request-response overlay.

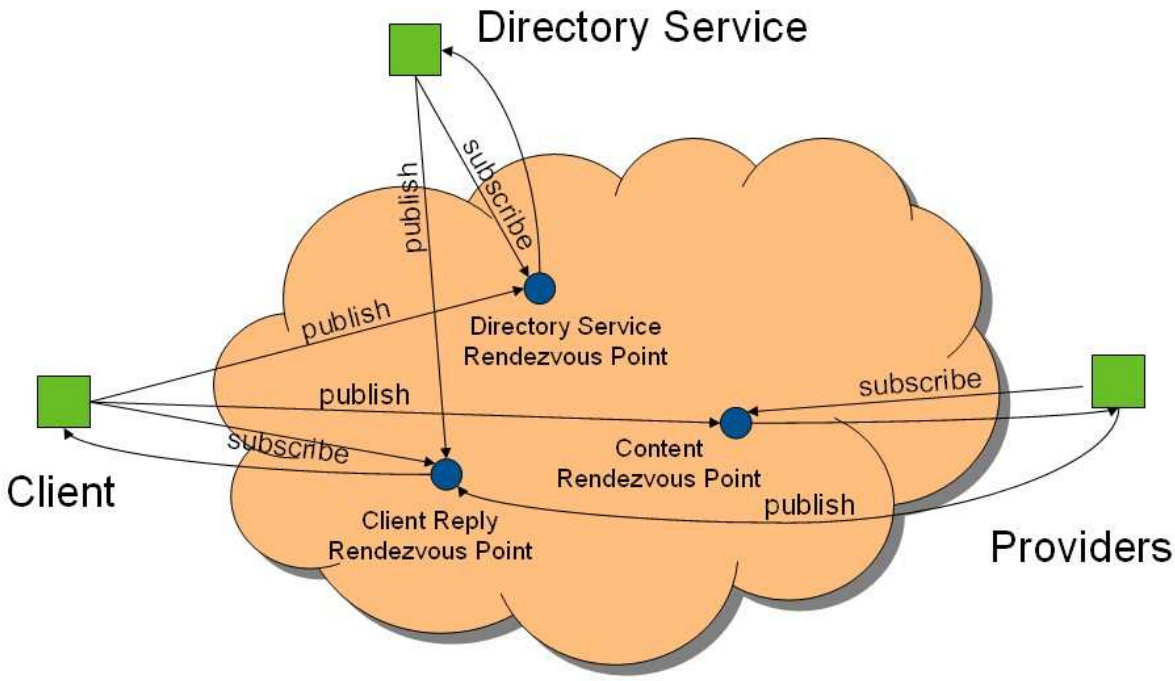


**Figure 10.1 – Request-response overlay in PSIRP**

### Channel-Based Publish-Subscribe

This communication model provides one-to-many delivery to all subscribers of a particular communication channel. It is also referred to as *identifier-based*, *address-based* or *topic-based* publish-subscribe, the latter typically providing structured hierarchical communication channels and the means to subscribe to a section of the hierarchy with a single operation.

Such publish-subscribe systems can be easily implemented over a PSIRP network by converting the identifier/address/topic into a rendezvous identifier, provided such an identifier supports multi-sender multicast for delivery. If only single-source multicast is supported by the PSIRP rendezvous identifier, then the receiver must subscribe to a rendezvous identifier for each potential sender for each channel. This has the advantage of allowing the receiver to establish trust with each sender, and easily reject an individual sender from the overall communication.

To implement hierarchical topic structure, each leaf node in the hierarchy can be represented by a rendezvous identifier. Subscribing to a higher level topic within the hierarchy is implemented by subscribing to all of the rendezvous identifiers for leaf topics that are below the higher level topic node. If a one-operation subscription is required, this can be implemented by edge middleware between the application and the network, along with providing storage and filtering capabilities.

The following figure outlines a potential overlay solution for this model.

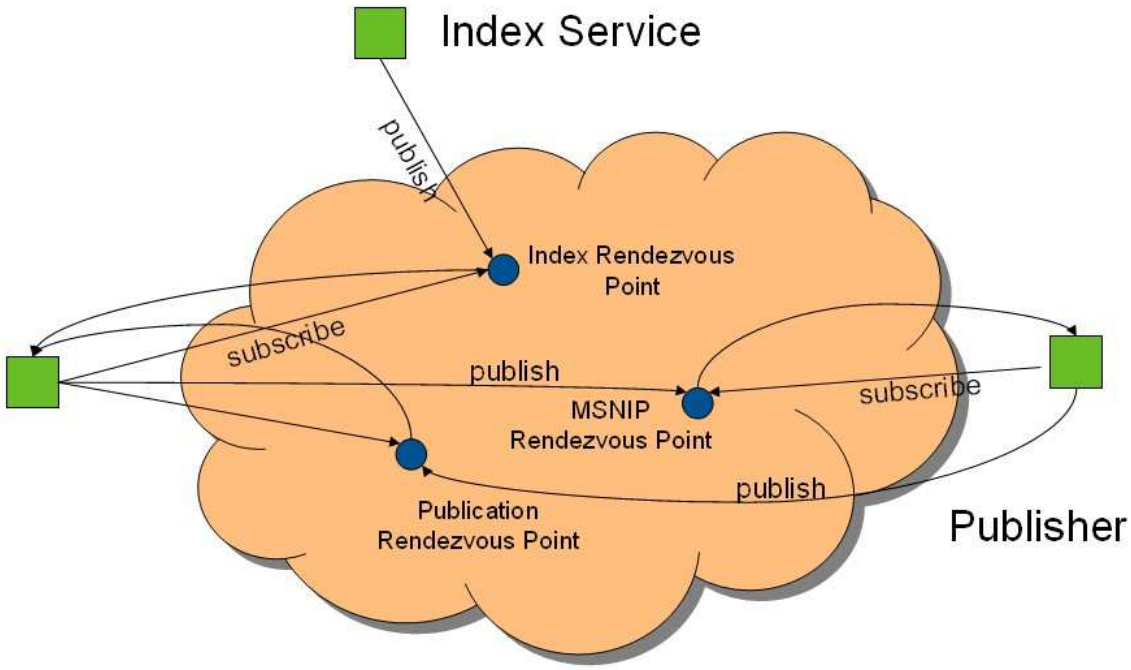


**Figure 10.2 - Channel-based pub/sub overlay in PSIRP**

### Content Filters

Many channel-based publish-subscribe systems allow the subscriber to employ a content filter to restrict the data they receive on a particular channel. This filtering capability requires some understanding of the data at the filtering points. Thus a filter might specify particular attributes of an XML message. For a general network such as the PSIRP rendezvous network, providing such capability would be difficult. Filtering XML messages requires different functions to filtering a voice session. If such plug-in filter capability is not provided by the network, then it can be applied more flexibly at the network edge. The disadvantage of this approach is that filters cannot be efficiently aggregated nearer the source of the communication. Overlay networks can be constructed for such purposes, but they are unlikely to provide efficient communication paths.

### Content-Based Publish-Subscribe

By content-based publish-subscribe we mean a network that routes structured content based upon routing policies that can apply to any part of the content. This type of network is classically implemented by SIENA [Car2003] [Ros2001], which also introduces the concept of content advertisements to improve subscription state scalability. This is dissimilar to channel-based systems where the routing occurs on only a single identifier/address/topic, but provides additional content filters at specific points (normally near the receiver edge). These two concepts blur together in systems that provide access controlled communication groups that provide content-based routing within the group. In this case the group can be viewed as analogous to a channel or topic.

Any existing content-based publish-subscribe system can be implemented as an overlay on the PSIRP network by using a rendezvous identifier for communication to each content router. In this example, the rendezvous identifier simply replaces the IP address used when the system runs over an IP network. This is also true for any other existing IP overlay or application.

Approaches that utilise multicast communication channels between content routers towards the edge of the network have the most to benefit by the implementation of network-level multicast provided by the PSIRP rendezvous and forwarding capabilities. One such approach is MEDYM (Match-Early with DYnamic Multicast) [Cao2005]. This work uses complex content matching nodes at the edge, but uses simple address based multicast routing between the match nodes. In this respect it is similar to edge deployed content filters for channel-based systems, but the multicast addresses remain hidden to the end application and are optimized for the delivery of content between the match nodes. To implement such a system over the PSIRP network, each match node can subscribe to a shared rendezvous identifier. A client may then send a content subscription using anycast to this rendezvous identifier, to reach the nearest available match node. In a simple system, without edge multicast, this content subscription would also include a further rendezvous identifier where the client can be sent any content that matches their content subscription. Internally the match nodes will create a set of rendezvous identifiers for distributing advertisements, subscriptions and content publications between themselves. When a publisher wishes to advertise available content or publish content to the network, this is again sent as an anycast communication to the rendezvous identifier for the match nodes and delivered to the nearest available node. It is then communicated to other remote match nodes which have subscriptions by choosing the appropriate internal rendezvous identifiers. Finally it will be delivered to the clients using the supplied rendezvous identifier for client notification.

It is also possible to extend the concept by locally matching clients to multicast edge delivery channels. If this is desired, the client notification rendezvous identifier is used to notify additional rendezvous identifiers that the client should join to receive the content that best matches their content subscription.

## 10.2 Common Applications

An important aspect of PSIRP design revolves around accommodating the demands of common Internet tasks. We divide such applications into two main categories: *PSIRP applications* and *legacy applications*. The former seek to harness the characteristics of pub/sub networking to provide new functionalities for both end-users and the underlying network infrastructure that are either infeasible or inefficient in the current Internet. However, the Internet as it exists today commands a high level of dependence from its users both as a result of its size and ubiquity. As a result, we still need to support upper-layer legacy applications and protocols in initial pub/sub implementations so that transitioning to the new architecture will be seamless and minimize service loss.

In this section, we contemplate innovative uses for pub/sub networking and also briefly consider how routine tasks such as WWW browsing, messaging, multiplayer gaming, file sharing etc., using the methods prescribed by current legacy systems, can be mapped to function in pub/sub environments. We also examine several relevant scenarios where these pub/sub applications may be useful.

### 10.2.1 E-mail

E-mail can be realized efficiently using the pub/sub primitives and the rendezvous system. Here, the idea is that a specific scope defines the rendezvous system for e-mail functionality corresponding to the SMTP protocol. The rendezvous system would then allocate a publication identifier and subscription identifier for the user that are globally reachable.

One important security issue for the e-mail service is: how to prevent users from sending e-mails using forged e-mail addresses? If we assume that a publisher can include a cryptographic signature and it's own public key to sent packets, then the security of e-mail can

be improved by using a simple principle where the e-mail address is derived from the user's public key, for example: e-mail address = Hash(public_key).

Thus, the e-mail service (or any other node in the network) could verify that the sender's e-mail address included in the e-mail message corresponds to the hash of the sender's public key (and the sender's public key must correspond to the signature over the packet). Such a method would offer a lightweight way to solve this issue.

This mechanism could also be utilized to authenticate users when retrieving their e-mail messages. The e-mail service would send messages only to the holder of the corresponding public key.

There are also many other issues in the PSIRP-based e-mail service, for example, what mechanism will be used to deliver e-mail messages to recipients, which must be considered separately.

### 10.2.2 World-Wide-Web Browsing (HTTP & DNS)

The anatomy of modern WWW browsing is relatively simple when examined at a high level. The use of URLs (i.e. domain name + content path) as "friendly names" simplifies addressing for users, who do not have to remember 32-bit IP addresses in order to reach desired content. This of course mandates DNS translation as a sort of middleware between the user and the Internet, reinforcing the fact that IP addresses serve as both identifiers for content and a hierarchical addressing scheme to define content locations. Once the desired domain has been resolved to the proper IP address, HTTP invokes simple client-server requests/responses to obtain the requisite information from the desired path. The involvement of advanced techniques, such as caching and intermediary forwarding, is beyond the scope of this document.

Unlike current IP-based networks, PSIRP applies a flat, possibly self-certifying, labeling scheme using information scoping and semantics that separates the identifier for a particular piece of data from its topological location (Section 3). The label size will be considerably larger than an IPv4 address in order to maintain unique significance across the network (i.e. total avoidance of NAT), mandating some sort of name resolution feature for-end users. However, the logical separation of identity and addressing (or the identifier-locator split) may allow resolution to be performed locally by the client. As discussed earlier in Section 3 of this document, the label scope is analogous to a URL domain name and the data description can be compared to the content path.

The following illustrates a potential mapping between current HTTP/DNS browsing and pub/sub implementations:

| | HTTP/DNS | PSIRP |
|---|---|---|
| **1** | Client enters a URL | Client enters friendly name (dynamics as of yet undecided). |
| **2** | URL is sent to DNS server which resolves corresponding IP address and returns result to the client. | Client performs a cryptographic hash of the friendly name and obtains the label scope OR Client publishes the friendly name to the local node's label, whereupon the corresponding label scope is resolved (either by this node or by forwarding to a rendezvous node). |
| In either of the above cases:<br>The local node subscribes on behalf of the client.<br>Since the data descriptor for the desired content may not always be known, publishers could implement a globally-standardized "broadcast" data description. Subscribing to a label containing the publisher's scope and this broadcast identifier could allow clients access to the publisher's "main page," whereupon specific data descriptions (i.e. linked content paths) can be determined and subscribed to. | | |
| **3** | Client sends HTTP request to the server. | The client's subscription is forwarded through the rendezvous network until it reaches either a node with the requisite cached contents or the node nearest the publisher (i.e. web server). |
| **4** | Server responds to the request and returns data to the client. | The node receiving the client's subscription adds the client to the list of label subscribers and forwards the current iteration of the data to the client. Subsequent updates to the content are forwarded automatically to the client. |

### 10.2.3 Voice-over-IP (VoIP)

Generally, a VoIP-type of communication service can be accomplished as follows:

| | VoIP | PSIRP |
|---|---|---|
| **1** | Client initializes their application and connects to the VoIP network, allowing them to receive calls from other users. | Client subscribes to their own label, a cryptographic hash of some friendly name and possibly the client's public key. This allows other users to contact the client at will. There should be methods available for the client to actively deny publications to its label from unwanted callers. |
| **2** | Client obtains the identifier for a desired peer and calls them. The distributed VoIP network resolves the client's identifier to its location and establishes a connection. Depending on the type of service, the call transit may be either packet-switched (packets take different routes to the destination) or circuit-switched (a single physical circuit is established and used by all packets). | To establish a call, the client determines the friendly name of its desired peer and obtains its label. This can either be accomplished locally through a cryptographic hash or via lookup through the rendezvous network. The client then publishes a "hello" message to this label. |
| **3** | Upon the establishment of this circuit, the client's peer is notified of the incoming call (i.e. their phone rings) and they can immediately communicate with the client. On call completion, the transport circuit is destroyed. | The receiving peer is provided a notification that they are receiving a call. They can either reject the call, upon which the calling client is informed and their subscription is removed, or accept the call. In the latter case, a subscription to the client's label is immediately performed to allow for return traffic. Both subscriptions are "torn down" when the call is completed, but label records may remain in rendezvous-nodes for a preset time period in order to improve connection speeds in the case of repeated calls in the future. |

A clear advantage of PSIRP in this regard is the effective use of multicast labels with the rendezvous network, allowing for conference calls without the need for a dedicated centralized call centre. Moreover, redundant connections can be achieved by using switched virtual circuits (SVCs) and traffic engineering (TE) techniques.

Another interesting possibility involves the use of public keys to define client labels, which could afford built-in bilateral authentication and encryption to calling parties.

As opposed to SS7's complex centralized network and dumb endpoints, a pub/sub VoIP implementation should attempt a distributed design which places more functionality in the hands of clients. SIP may potentially be a good modelling point in this regard.

**SIP in a PSIRP world**

A central technology in today's VoIP implementations is the Session Initiation Protocol (SIP) [Ros2002b] and its extension, the SIP event mechanism [Roa2002]. The SIP architecture is in many ways similar to a pub/sub architecture and is therefore easy to map onto a PSIRP internetworking architecture. The standard initiation of a VOIP communication is implemented via a request-response scheme (SIP INVITE [Ros2002b]). Given the request-response model presented in Section 10.1, this can be implemented as follows:

- the SIP AOR (address-of-record), which represents the client's identifier, is published to the directory service (see Figure 10.1)
- the SIP contact address of the inviting party represents the client reply rendezvous identifier
- the SIP contact address of the invited party represents the content rendezvous identifier
- the SIP INVITE is sent to the content rendezvous identifier with the client reply rendezvous identifier given in the invite
- the communication takes place in similar request-reply (i.e., offer-answer) fashion as in standard SIP

For SIP event extensions [Roa2002], such as for SIP registrations [Ros2002a] or SIP-based presence applications, the PUBLISH/SUBSCRIBE methods of the SIP event extensions can be implemented as a topic-based pub/sub overlay on top of PSIRP. Given the description of this overlay in Section 10.1, the SIP extensions can be implemented as follows:

- the SIP event package is the index in Figure 10.2. Through the index service, the SIP environment could be extended towards dynamically created events. The identifiers for these events would then be published by their creators in the index service. When implementing the current standard SIP event system, these event identifiers would be defined in standards and a lookup would therefore not be necessary.
- the SIP event server subscribes to the publication rendezvous identifier (for SIP SUBSCRIBE)
- the SIP publisher subscribes to the MSNIP rendezvous identifier for the reply channel
- the SIP event publication (SIP PUBLISH) takes place towards the publication rendezvous identifier, carrying the MSNIP identifier for the reply channel
- the SIP event server sends any reply information (SIP REPLY) to the obtained MSNIP identifier.

### 10.2.4 Wireless Sensor Networks (WSNs)

Sensor networks are an excellent application area for information-centric architectures like the one developed in PSIRP. As also argued in [Bri2004], most wireless sensor networks (WSNs) rely on diffusion methods that allow for replying to sensor data requests in a distributed manner. In other words, requests are answered by "relevant" nodes that might be able to contribute to the required response as opposed to being directed to and answered by specific nodes. For this, addressing is applied to information rather than to nodes, which closely relates these environments to PSIRP.

In contrast to the other application areas presented so far, there is no pre-dominant pub/sub solution for WSNs. However, we can largely divide the efforts in WSNs into two categories, namely:

- ad-hoc WSNs
- infrastructure-based sensor networks

The former category operates in many cases based on some ad-hoc routing protocol (in which the routing metric is based on the particular constraints of the environment, such as battery power, distance, etc.) with a data diffusion technique. Approaches such as TinyDB [Mad2005] rely on a view of the *network as a database* with diffusion requests to the nodes within the network. Diffusion techniques can be implemented within the PSIRP system as an anycast primitive (the anycast metric being closely aligned with the diffusion technique at hand).

Conversely, infrastructure-based sensor networks might operate in wired and wireless environments that are based on more traditional endpoint-centric infrastructures, such as IP. For instance, the Nokia Remote Sensing (NORS) platform presented in [Tro2007] operates over standard IP, implementing a pub/sub overlay for subscribing to sensor data in local sensor networks via IP-enabled sensor gateways. For the latter, the platform employs standard Java-enabled mobile devices. The applied pub/sub protocol is similar to the SIP events standard [Roa2002] and is therefore readily ported onto a PSIRP API. In addition, the ability to directly address information rather than the sensor gateway, as it is currently done in NORS, generates new application possibilities for this type of platform, in particular in participatory sensing scenarios, one of the focus use cases for NORS. Such possibilities, for instance, can comprise the provisioning of certain sensor information within a certain location by a number of sensor gateways. This effectively implements a sensor diffusion scheme in an infrastructure-based sensor network. Local sensor integration is accomplished either via proprietary methods or via connecting to local potentially PSIRP-based diffusion networks.

### 10.2.5 Discussion

Pub/sub can be especially useful to initiate push-based notifications when web sites are updated. The current Internet is based on client polling, which is very inefficient and consumes a lot of network resources. An alternative, RSS, is also burdensome to content servers and may be problematic to users who are located behind NAT proxies when service providers limit RSS polls by source address to constrain network resource consumption. On the other hand, PSIRP's pub/sub architecture can eliminate client polling, provides abundant content-based addressing space to eradicate the need for NAT, and allows for efficient data multicasting that optimizes one-to-many transmissions.

Labels could be defined to include public keys so that all pub/sub operations are forced to employ authentication and encryption services. Moreover, entities could generate their own public/private key pairs for use in situations where anonymity is desired.

# 11 Conclusions

This report presented the first results of the architectural design process within the PSIRP project. It can only be seen as the first step in the desired clean slate (re-)design of the Internet. As we outlined in our methodology (see Section 2), we envision a process of bottom-up lessons learned and top-down rationalization, the first results you can see in this report.

Following this methodology, the conceptual architecture and components presented in this report is only part of our progress so far in the project. The clarification of concepts, presented in the design considerations, as well as the formulation of new questions pushing forward our future development, is the rather hidden progress that we made. Hence, many of the issues addressed in this report, such as identifiers, the concept of scope, rendezvous, caching, forwarding and transport as well as our inter-domain routing solution, will see further progress in the near future, based on the discussions and work that led to this current report.

In particular, the novel aspects of our thinking in the direction of information-centrism and scoping of information reachability is likely to fuel this progress. Also our considerations on the role of transport and caching are likely to see deepening in our near future work. These considerations and their results will find entry in the more detailed technical architecture to be delivered by this project in M14, together with dedicated evaluation of the overall architecture and particular components, such as forwarding or rendezvous.

# 12 Terminology

| | |
|---|---|
| *Application identifier* | Any higher-level identifier that applications may use. It is usually mapped onto a set of rendezvous identifiers for the data items relating to such application identifier. |
| *Caching* | Process of temporarily storing data that is expected to be useful in the future in intermediate network elements. |
| *Component wheel* | The PSIRP layerless network "stack" design, in which a number of network components communicate internally within a node using the publish/subscribe paradigm. |
| *Data* | Published data. Data is published using a rendezvous identifier together with a set of scope identifiers. |
| *Domain* | A managed network that is analogous to an autonomous system. |
| *Forwarding identifier* | An identifier used to associate a publication with a forwarding multicast tree or forest. |
| *Inter-domain routing* | Inter-domain routing pertains to data delivery in the global network, typically spanning several domains. The inter-domain routing system is configured through the rendezvous process and takes into account any inter-domain policies in effect. |
| *Intra-domain routing* | Intra-domain routing pertains to data delivery within an administrative domain. Intra-domain routing is concerned with local policies. |
| *Metadata* | Data may also have associated metadata, which includes scoping information and other useful information either for ultimate receivers or network elements. This metadata in itself is data, i.e., it is associated with another rendezvous identifier. Metadata may be provided within a publication or as a separate data element with a separate rendezvous identifier. |
| *Network attachment* | The process of obtaining connectivity with a PSIRP network. |

| | |
|---|---|
| *Publication* | A self-contained data unit that has been made available using the publish primitive. |
| *Publisher* | An entity that uses the publish primitive to make data available. |
| *Receiver Scope* | Scope that determines the parts of the rendezvous system that are responsible for updating subscription state for a network entity. A scope can have many interpretations and they may be associated with each other. It is the responsibility of the rendezvous system to ensure that scopes follow set policies. |
| *Rendezvous* | Rendezvous is the process of matching publishers and subscribers according to given rendezvous identifiers and initiating the transfer of data over the network within a given scope. |
| *Rendezvous identifier* | An identifier given to an entity by the rendezvous system in order to mediate between high-level identifiers, namely application identifiers, and lower-level identifiers, namely forwarding identifiers. |
| *Scope* | Scope determines the part of the rendezvous system that is used by the network. The three typical low level cases are link local, intra-domain, and inter-domain. But scope can also be used to map higher-level concepts, like social networks, onto particular parts of the rendezvous system. |
| *Sender Scope* | Scope that determines how the network rendezvous system should interpret a publication. A scope can have many interpretations and they may be associated with each other. It is the responsibility of the rendezvous system to ensure that scopes follow set policies. |
| *Service model* | A model that is used by network elements to interface with the network. The PSIRP service model that includes three parts: subscriber, publisher, and network service parts. |
| *Sprouter* | A PSIRP Publish/Subscribe Router |
| Subscriber | An entity that uses the subscribe primitive to request certain pieces of data. |
| *Tussle* | Tussle is defined as a conflict of interests, these interests being brought forward by players within a given communication scenario, either expressed |

as explicit constraints, requirements or other concepts of concerns. Design for Tussle [Cla2002] offered a novel insight for the proper design of such systems, i.e., providing guidance as to how a system ought to be designed so as to withstand and even incorporate a wide range of tussles.

# 13 References

[Ban2001] A. Banerjee, J. Drake, J.P. Lang, B. Turner, K. Kompella and Y. Rekhter, "Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements," *IEEE Communications Magazine*, vol. 39, issue 1, pp. 144-150, Jan. 2001.

[Bel1976] D. Belsnes, "Single-Message Communication," *IEEE Transactions on Communications*, vol. 24, issue 2, pp. 190-194, 1976.

[Ber2001] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American Magazine*, May 17, 2001, available at: http://www.sciam.com/article.cfm?id=the-semantic-web [Accessed on 15 July, 2008].

[Bri2004] R. Briscoe, "The Implications of Pervasive Computing on Network Design," *BT Technology Journal*, vol. 22, issue 3, pp. 170-190, July 2004.

[Bri2005] R. Briscoe, A. Jacquet, C. Di Cairano-Gilfedder, A. Salvatori, A. Soppera, and M. Koyabe, "Policing Congestion Response in an Internetwork Using Re-feedback," *ACM SIGCOMM Computer Communications Review*, vol. 35, issue 4, pp. 277-288, Sept. 2005.

[Cae2006] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "ROFL: Routing on Flat Labels," *ACM SIGCOMM Computer Communication Review*, vol. 36, issue 4, pp. 363-374, Sept. 2006.

[Cao2005] F. Cao and J. P. Singh, "MEDYM: Match-Early and Dynamic Multicast for Content-based Publish-Subscribe Service Networks," *25th IEEE International Conference on Distributed Computing Systems Workshops*, Colombus, OH, June 2005, pp. 370-376.

[Car1996] B. Carpenter, *Architectural Principles of the Internet*, IETF RFC 1958, June 1996.

[Car2001] A Carzaniga and A.L. Wolf, "Design and Evaluation of a Wide-area Event Notification Service", *ACM Transactions on Computer Systems (TOCS)*, vol. 19, issue 3, pp. 332-383, Aug. 2001.

[Car2003] A. Carzaniga and A. L. Wolf, "Forwarding in a Content-based Network," in *Proc. ACM SIGCOMM 2003*, Karlsruhe, Germany, Aug. 2003, pp. 163-174.

[Che2000] D. R. Cheriton and M. Gritter, "TRIAD: A New Next-Generation Internet Architecture," July 2000, available at: http://www-dsg.stanford.edu/triad/ [Accessed on 15 July, 2008].

[Cla2002] D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet," in *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, New York, NY, Aug. 2002, pp. 347-356.

[Cla2003] D. Clark, R. Braden, A. Falk, and V. Pingali, "FARA: Reorganizing the Addressing Architecture," in *Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Karlsruhe, Germany, Aug. 2003, pp. 313-321.

[Cla2007] D. Clark and M. Blumenthal, "The End-to-End Argument and Application Design: The Role of Trust," in *Proc. Conference on Communication, Information, and Internet Policy (TPRC)*, Arlington, VA, Sept. 2007.

[Egg2004] L. Eggert and J. Laganier, *Host Identity Protocol (HIP) Rendezvous Mechanisms*, IETF Internet draft, work in progress, Oct. 2004.

[Eug2003a] P.T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov and A. M. Kermarrec, "Lightweight Probabilistic Broadcast", in *ACM Transactions on Computer Systems (TOCS)*, vol. 21, issue 4, pp. 341-374, 2003.

[Eug2003b] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, issue 2, pp. 114-131, 2003.

[For2004] B. Ford, "Unmanaged Internet Protocol: Taming the Edge Network Management Crisis," *SIGCOMM Computer Communication Review*, vol. 35, issue 1, pp. 93-98, 2004.

[Gao2001] L. Gao, "On Inferring Autonomous System Relationships in the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, issue 6, pp. 733-745, Dec 2001.

[Gir1991] M. Girault, "Self-Certified Public Keys," *Advances in Cryptology (EUROCRYPT)*, pp. 490-497, 1991.

[Hag2007] Haggle partners, "Deliverable D1.2: Specification of the CHILD-Haggle," *HAGGLE*, Aug. 2007, available at: http://www.haggleproject.org/deliverables/D1.2_final.pdf [Accessed on 15 July, 2008].

[Hue2003] R. Huebsch, "Content-Based Multicast: Comparison of Implementation Options," Technical Report, 2003.

[Jac2006] V. Jacobson, "If a Clean Slate is the Solution What Was the Problem?," *Stanford "Clean Slate" Seminar*, Feb. 2006, available at:

http://cleanslate.stanford.edu/seminars/jacobson.pdf [Accessed on 15 July, 2008].

[Kat2006] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in The Air: Practical Wireless Network Coding," *ACM SIGCOMM Computer Communication Review*, vol. 36, issue 4, pp. 243-254, 2006.

[Key2006] P. Key, L. Massoulié, and D. Towsley, "Combining Multipath Routing and Congestion Control for Robustness," in *Proc. 40th IEEE Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, Mar. 2006, pp. 345-350.

[Kop2007] T. Koponen, M. Chawla, B. G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A Data-Oriented (and Beyond) Network Architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, issue 4, Oct. 2007, pp. 181-192.

[Mad2005] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An Acqusitional Query Processing System for Sensor Networks", in *Proc. ACM TODS*, 2005.

[Moo2002] T. Moors, "A Critical Review of 'End-to-end Arguments in System Design'," *International Conference on Communications*, vol. 2, pp. 1214-1219, 2002.

[Mos2008] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, *Host Identity Protocol*, IETF RFC 5201, Apr. 2008.

[Nek2003] M. Nekovee, A. Soppera, and T. Burbridge, "An Adaptive Method for Dynamic Audience Size Estimation in Multicast," in *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, vol. 2816, 2003.

[Nik2004] P. Nikander, J. Arkko, and B. Ohlman, "Host Identity Indirection Infrastructure (Hi3)", in *Proc. Second Swedish National Computer Networking Workshop (SNCNW 2004)*, Karlstad, Sweden, 2004.

[Pie2004] P. R. Pietzuch, "Hermes: A Scalable Event-Based Middleware," doctoral dissertation, Computer Laboratory, Queens' College, University of Cambridge, Feb. 2004.

[PSI2008a] PSIRP, *PSIRP Homepage*, available at: http://wiki.psirp.org [Accessed on 15 July, 2008].

[PSI2008b] D. Trossen (ed.), "PSIRP Vision Document", available at http://www.psirp.org/ [Accessed on 15 July, 2008].

[Ram2001] K. K. Ramakrishnan, S. Floyd, and D. L. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, IETF RFC 3168, Sept. 2001.

[Roa2002] A. B. Roach, "Session Initiation Protocol (SIP)-Specific Event Notification", IETF RFC 3265, June 2002.

[Rat2001] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level Multicast using Content-addressable Networks," *Lecture Notes in Computer Science*, 2233, 2001, pp. 14-29.

[Ros2001] D. Rosenblum, "A Tour of Siena, an Interoperability Infrastructure for Internet-scale Distributed Architectures," *Ground System Architectures Workshop*, Feb. 2001.

[Ros2002a] J. Rosenberg, *A Session Initiation Protocol (SIP) Event Package for Registrations*, IETF RFC 3680, Mar. 2004.

[Ros2002b] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, IETF RFC 3261, June 2002.

[Row2001] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," in *Proc. of Middleware 2001*, Heidelberg, Germany, Nov. 2001, pp. 329–250.

[Sal1984] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," *ACM Transactions on Computer Systems*, vol. 2, issue 4, pp. 277-288, Nov. 1984.

[Sar2008] Mikko Särelä, Teemu Rinta-aho, Sasu Tarkoma, "RTFM: Publish/Subscribe Internetworking Architecture," in *Proc. ICT-Mobile Summit*, Stockholm, Sweden, June 2008.

[Sop2003] A. Soppera, T. Burbridge, B. Briscoe, and M. Rizzo: "GAP: The generic announcement protocol for event messaging," in proc. *London Communication Symposium (LCS'03)*, UCL, London, UK, Sept. 2003.

[Sto2002] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proc. 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pittsburgh, PA, Aug. 2002, pp. 73-86.

[Tar2007] S. Tarkoma and J. Kangasharju, "On the Cost and Safety of Handoffs in Content-based Routing Systems," *Computer Networks*, vol. 51, no. 6, Apr. 2007.

[Tar2008] S. Tarkoma, D. Trossen, M. Särelä, "Black Boxed Rendezvous Based

Networking," *The 3rd ACM International Workshop on Mobility in the Evolving Internet Architecture (MobiArch 2008)*, Aug. 2008, unpublished.

[Tro2007] D. Trossen and D. Pavel, "NORS: An Open Source Platform to Facilitate Participatory Sensing with Mobile Phones", in *Proc. Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous 2007)*, Philadelphia, PA, Aug. 2007.

[Yan2007] X. Yang, D. Clark, and A. Berger, "NIRA: A New Inter-Domain Routing Architecture," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 775-788, Aug. 2007.