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

DELIVERABLE D2.5

Final Updated Architecture

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	33 months, until 30.09.2010
Document Title:	Final Updated Architecture
Date of preparation:	19.7.2010
Authors:	Dirk Trossen (UCAM) (editor), Pekka Nikander (LMF), Kari Visala (AALTO-HIIT), Trevor Burbridge (BT), Paul Botham (BT), Chris Reason (BT), Mikko Sarela (LMF), Dmitrij Lagutin (AALTO-HIIT), Ventzislav Koptchev (IPP-BAS)
Responsible of the deliverable:	Dirk Trossen (UCAM) Phone: +44 7918 711695 Email: dirk.trossen@cl.cam.ac.uk
Reviewed by:	Dirk Trossen (UCAM), Trevor Burbridge (BT), Pekka Nikander (LMF), Janne Riihijarvi (RWTH), Paul Botham (BT), George Parisis (UCAM), George Xylomenos (AUEB)
Target Dissemination Level:	Public
Status of the Document:	Completed
Version	1.0
Document location	http://www.psirp.org/publications/
Project web site	http://www.psirp.org/

Table of Contents

1 Introduction

The ambition of the PSIRP project is to define a new architectural waist for a Future Internet in which information is at the very heart of the network's operation. This new waist of an inter-networking architecture is defined and specified through a series of deliverables that addresses a variety of issues related to the project's ambitions. These issues range from design principles over design considerations to outlining design and implementation choices. This produces a broad set of material that is considered to be helpful for the project as well as for the wider community in order to further develop the basic ideas and concept of PSIRP into a workable prototype of this Future Internet.

This document should be approached in this evolving work context. It is an update and extension to ideas presented throughout the other documents of the project, most notably the first and second architecture deliverable, D2.2 and D2.3 respectively. Hence, one should not expect a final specification of our architecture that can be handed to a system engineer for implementation. Instead, this document sharpens and continues our work in several key areas. Firstly, we present a section that positions our previous work on design principles and concepts in the context of defining invariants of our architecture, i.e., properties that we see as inherent for any design choice based on our thinking. Secondly, we provide an update on design considerations and choices for major components and areas in our architecture, ranging from inter-domain functions over identifier considerations to security and transport considerations. Last but not least, we summarize the key points that we see as the major contributions of the project's efforts in an attempt to set out an agenda for future work.

2 Architecture Invariants

In [PSI09], we outlined the design principles for developing the PSIRP architecture. Throughout our work, these principles have laid the foundation for a set of invariants of our architecture, i.e., properties that do not change throughout the implementation and execution of said architecture. We believe that the formulation of such invariants is an important step in our development since it constitutes an understanding as to what is fundamental not only for the design of the architecture but also in its realization. Hence, while the principles focus on the design, we now conclude that the following properties are fundamental to the actual implementation of what we have designed so far.

Given that the PSIRP architecture provides a replacement for the IP waist in the current Internet hourglass, we see the following invariants being provided at the waist level:

- The **first invariant** is that of flat label identifiers as being the means to identify information items as the main entities of the architecture. An information item is generally any collection of data that is relevant within a given application context. It can represent a principal of a transaction, a policy rule acting on another piece of data, or a pointer to some imaging data. In our new waist, each information item is identified using a statistically unique flat label identifier. These identifiers are self-generated and the associated semantic of the information is only known to the applications generating the said identifiers. As an example, a video publisher might generate an identifier through hashing a human-readable name of a video into a suitable identifier[1].
- The **second invariant** is that of scoping as a means for hierarchically ordering information along certain application structures. Each information item is placed in at least one scope. There may be one or more global scopes, which make information items reachable to anybody. A scope, therefore, becomes nothing more than a special information item, holding other information items within an application-specific structure. An example of a scope could be a set of pictures or a group of friends in a social network.
- The **third invariant** is the service model of the waist, which consists of publishing of and subscribing to individual information items within a scope. We argue that this model is conceptually broader than traditional request-response service models that are largely present in the current Internet although such models can be implemented through, e.g., including identifiers for response information into the original request publication with the client subscribing to the response identifiers and the server publishing to it.
- The **fourth invariant** is that of providing functions for finding information, determining a valid delivery graph, and forwarding information (packets) along that graph. It is to be noted that these functions are often separated in implementation but they may also be merged as a result of (often local) optimization.

Although the goal of our work is not to show the application of these invariants to ANY information-centric architecture, we believe that a fundamental architectural discussion is required around possible properties (and even invariants) of a set of architectures with goals similar to those of PSIRP. We acknowledge and highly appreciate the contribution of John Wroclawski (USC-ISI) and Karen Sollins (MIT CSAIL) to this discussion. The formulation of the PSIRP invariants and the discussion of their potential central role as inherent properties of

[1] Not an invariant but an important design tool is the notion of algorithmic identifiers. Here, a common algorithm is utilized among a set of network elements to logically bind information items to a collection of items. Through such a common algorithm, network elements can determine, for instance, parents or sibling relationships among information items. Functions like error control, caching, sequencing and others can utilize these mechanisms but upper layer applications can also effectively order the relations of items through such algorithmic relations and utilize lower network functions for improved dissemination of such collections.

a certain class of information-centric architecture is a direct consequence of intensive and ongoing discussion and collaboration with both of them.

2.1 Design Considerations

From our invariants outlined above, a set of design considerations follows that are directly related to the invariants. More specific implementation and design considerations are presented in Section 3.

2.1.1 On Layering

The invariant of utilizing labels and scoping for structuring information goes further than attempting to provide application developers with a more natural way to access the network. Together with the other invariants, it leads to a concept of layering that describes a new way to build up a layered architecture – defining a new Internet hourglass.

Referring to Figure 2.1, the invariants of information items within scopes are utilized above the waist to implement *scopes of discourse* through the composition of scopes. These composed scopes can be used as constraints in the pub/sub operations that act upon a particular information item. With this, we assert, concepts of context, scope of information reachability and other social constructs can be implemented through recursively applying a scoping operation.

For instance, a high-level service such as Facebook might constitute a very large scope, exposed in the global scope(s) for universal reachability towards the members of Facebook. This larger scope can be further constrained by individual group or friend scopes, eventually limiting the reachability of the information items residing in these scopes of discourse. The reachability of the information items to given sets of users, e.g., your friends on Facebook, can be limited through realizing access control mechanisms for particular scopes. Hence, with this set of constraining scopes, various communication patterns within social networking applications can be implemented.

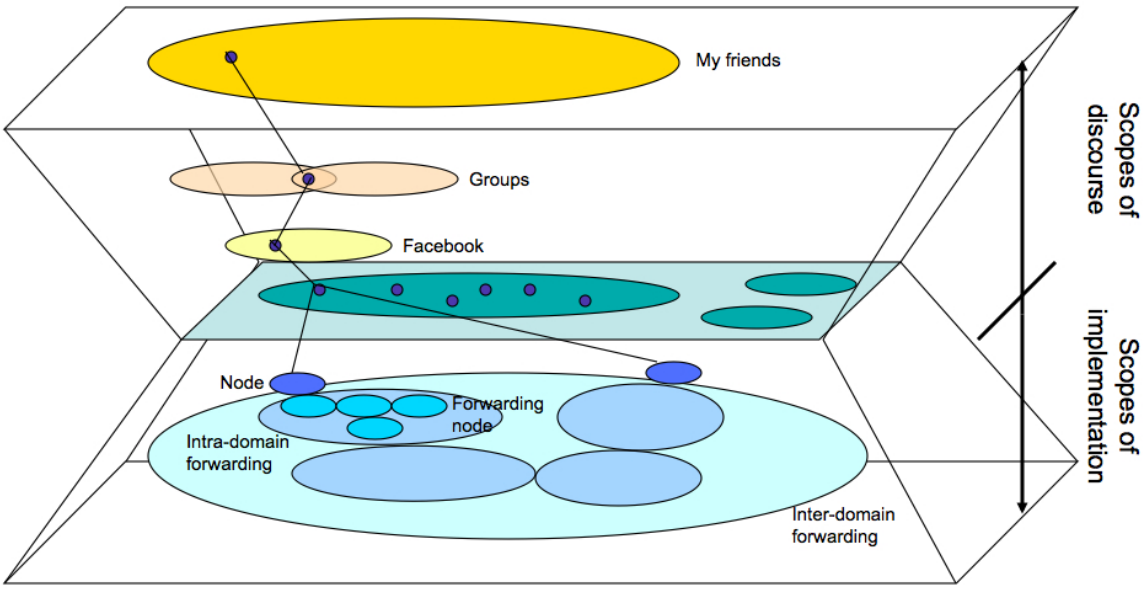

Figure 2.1: A New Hourglass

In another example, one can represent an organizational structure, in which a corporation is reflected in the highest scope (within the organization) with further scopes being used to constrain information to, e.g., business units, departments, groups, or individuals. It is likely

that a resolution mechanism will exist for resolving human-readable concepts onto the scopes of discourse and the labelling within each of these scopes.

At the level of the waist, a new API is exposed to the application developer. Proposals have been made for such a new API, e.g., in [PSI09]. Common to these proposals is the higher level of abstraction where individual information items are requested through a pub/sub-like service model, following our third invariant outlined above.

While we utilize the scoping concept above the waist to implement social structures through a composition of scopes of discourse, scoping is utilized below the waist, too, as *scopes of implementation*. Here, the discourse is that of realizing the delivery of information across actual transport networks. Scopes are utilized to define the boundaries for a functional model of network functions that determine the dissemination strategy for the information items residing within a particular scope. Hence, it provides a region of consistency to implement the necessary functions for dissemination.

As stated in our fourth invariant, these major functions relate to the *finding of information*, the *formation of an appropriate delivery topology,* and the final act of *delivery along the formed topology*. As indicated in Figure 2.1, such boundaries can be thought of as node-internal strategies, link-local strategies, strategies within single domains, or across domains. For instance, the implementation of the information-centric protocol stack of a PSIRP node provides a private, node-local scope for inter-process communication as well as scopes for intra- and inter-domain network functions, utilized for local forwarding, topology management, rendezvous and alike.

The techniques described in [Jok09] outline an intra-domain forwarding solution, which effectively implements a series of overlapping link-local scopes within a single intra-domain scope. In this case, the information is being disseminated as a series of packets transmitted from a publisher (or domain ingress) node. This level of implementation is possibly several "layers" under that of the application developer's original publication since additional network functions such as segmentation and error control can also be supported as separate scopes of implementation. This effectively leads to extending a high-level API that is exposed towards the application developer with functions for memory-like access, as outlined in [PSI09].

The lesson learned here is that having information as a common thread provides an appealing layering concept where functions of information finding, topology formation and delivery (forwarding) recursively appear throughout all layers. This enables a commonality in design that can be utilized for developing a rigid design framework for an overall architecture. While we can see examples for such layering in work such as [PSI09, Fot09], its benefits are still open to evaluation. Only the development of a coherent design framework based on such layering is likely to provide the insight needed for assessing the potential benefits of this design.

2.1.2 On Separation of Functions

We stated above that the existence of functions for finding information, building a delivery graph, and forwarding information along this graph is a crucial invariant of our architecture. This brings up the issue of separating these functions or merging them for implementation (and often optimization) reasons. It is vital to understand that this issue is important in the nature of implementing a scalable information-centric architecture. The following example is constructed to underline this point.

Let us think of a social networking very much akin to today's solutions, like Facebook. A large-scale social network such as Facebook is likely to be distributed all the over the world in terms of publishers and subscribers. In other words, one can think of it as a social construct that is unlikely to be locally constrained. We assume that there is at least one scope of discourse that is hosting the information space of the social network.

There are now two options for implementation. First, let us consider one that is similar to today's Facebook in which the identifier “facebook.com” points to a (set of) server(s) that “host” the service Facebook. Hence, all publications to the Facebook scope are in fact uploads, i.e., any subscription to a named data piece is in fact routed to the Facebook server farm. In this case all information has in fact a location by virtue of the upload operation to a set of dedicated servers, whether one wanted it or not.

Let us then consider another approach that builds on the power of storing the data at the publisher or at any other node. In this case, the social network is represented by the grouping through the discourse scope representing the social network. This is appealing to a company like Facebook since it still allows control over the data by virtue of possible access control and profiling of usage patterns while relieving Facebook from the burden of hosting the actual data, and therefore reducing overall costs of their operations. Any entity that happens to have a particular information item (such as a status update or photo) can provide the information to the interested subscriber.

In this form of a social network, what would happen if functions of finding and delivery were not separated? For that, we assume a similar operation as implemented in CCN [Jac09]. An interest in a specific social network information item is broadcasted within a single domain with one or more nodes replying with the requested information. If the content is not available in the domain, a content router [Jac09] forwards the interest request to any domain that hosts the information. When a content router receives an interest request, it broadcasts the request locally in order to possibly retrieve the item. Hence, in this implementation, finding information and routing along a delivery graph are folded into a single operation.

What does this mean for our scenario? In CCN terms, the discourse scope would be represented by a single domain, say “facebook.com”, with content within that domain being represented, e.g., as an XPath representation. Given the widespread geography of publishers in our social network, this would require that almost ANY content router in ANY domain would need information about 'facebook.com'. But what would this information be since there is no single domain that hosts the data anymore, i.e., some facebook.com data is likely to exist in ANY domain worldwide? As a result, ANY status update of ANY social network member is likely to be spread over many, if not all, domains in the Internet! If we couple this with the local broadcast of interest requests upon reception of such a request, the operation amounts to a global flooding of status updates in any network that might hold viable information about this social network.

What is the lesson to be learned here? It is that, if information is location-less (which is often the case), finding the information needs to be separated from the construction of an appropriate delivery graph, in order to optimize the operation of each of these functions. This motivates the introduction of an explicit (global) rendezvous service in the PSIRP architecture [PSI09]. However, it does not exclude solutions for (implementation) scopes in which functions are merged for optimization reasons. The choices of implementing the functions (either separated or optimally joined) is realized within the layering concept of Section 2.1.1 in which such implementation scope provides a region of consistency for implementing the functions.

3 Update on Architecture Components

In this chapter, we present updates of the designs of various architectural components. These include inter-domain functions and various problems that need to be tackled in operations as well as design and security considerations for various components and operations. As outlined in the introduction, this material should be read in the context of existing work that is presented in the previous architecture deliverables D2.3 and D2.4.

3.1 Inter-Domain Topology Formation

Within the PSIRP architecture, the *Inter-domain Topology Formation* (ITF) function essentially provides "high-quality" routes to customers, including both end-users and ISPs. The technical challenge is then to optimally match the various and sometimes conflicting requirements this implies. Ideally, we seek to guide ITF technical design choices with regard to wider socio-economic factors including technology supply, market conditions, user concerns (e.g. security/privacy) and regulatory actions.

In this section, we summarise our progress in this field. Firstly, Section 3.1.1 employs modelling techniques to consider likely Internet evolution scenarios and identify any consequences for ITF architectural design. Section 3.1.2 then emphasises the crucial importance of ITF security/privacy considerations, considering a variety of security-related design choices for implementing an ITF function within PSIRP. Lastly, Section 3.1.3 details specific (preferred) options for ITF implementation and considers the likely consequences in practice.

3.1.1 Design Considerations

In the current ITF context, we have identified four scenarios as particularly relevant:

1. Finance: focus on the reliance on highly resilient links for financial transactions
2. Technology: focus on a possible breakdown of technology cycles
3. Routing: focus on possible routing choices
4. Privacy: focus on possible backlash against privacy evading technologies

Details are discussed below but we note that the first two cases are relatively simple (serving largely to calibrate and validate the models) while the remaining two scenarios are more PSIRP-specific. To help understand the broader socio-economic issues, we employed a socio-economic evaluation approach that is based on "Systems Dynamics" modelling and simulation techniques. Previous work [PSI10] has described our general approach and methodology, identifying key variables ("stocks") and their dependencies ("flows"), as summarised in Figure 3.1 below.

With regard to the model itself, the fundamental behaviour is a flow of technology solutions to ITF providers, controlled by the regulator and subject to various feedback loops. The regulator responds to user concerns and (typically) seeks to limit market growth to ensure reasonable competition. PSIRP's success will obviously be dependent on both perceived usefulness (i.e., market demand) and compatibility of technical design choices with the prevailing socio-economic climate. Positive market feedback may be countered by negative regulatory feedback, depending on the specific scenario.

The central role of the *number of ITF providers* is emphasised by its strong connectivity to other variables in the model. In particular, industry/end-user concerns and the effect of hype were assumed to be triggered by appropriate threshold parameter-values for the *number of ITF providers*. A delay between regulatory decisions and actions was also introduced to provide additional flexibility and realism. From a modelling perspective, the current trend

towards collaborative regulation [Hui02] may also be captured by allowing parameters to assume flexible values (in order to perform a "sensitivity analysis").

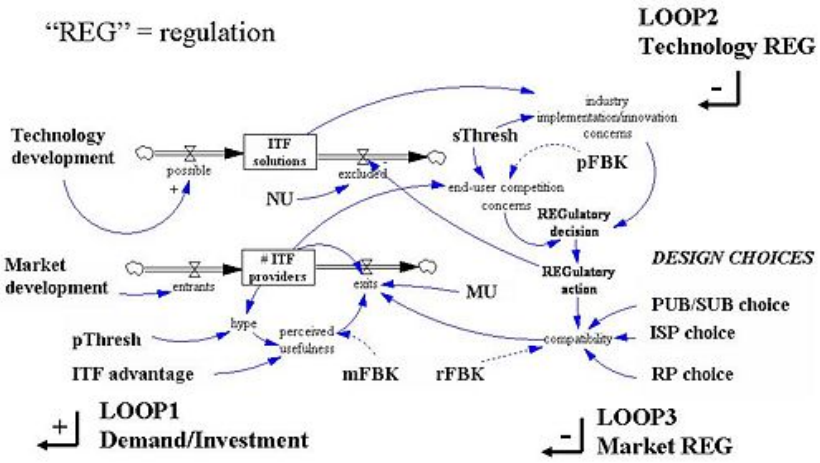

Figure 3.1: ITF Stocks and Flows

For definiteness, as a "base case" for model calibration and validation, we assigned parameter-values for the rate of both technology and market development, together with other variables, as shown below.

Quantity	Value	Units
Technology development rate	30	solutions/year
Market development rate	20	providers/year
Threshold for industry/end-user concerns	30	number of providers
Threshold for hype	50	number of providers
Regulatory delay	3	months

Table 3.1: Base Case Parameter-Values

3.1.1.1 Model Calibration

For Scenario 1, we envisaged a situation where the available capital investment changes markedly over a 20-year timescale, expecting a differential impact on PSIRP take-up relative to more conventional network offerings. In view of recent global economic upheavals, fluctuations in the wider economy (e.g., telecom bubbles vs. global recession) could obviously strongly affect PSIRP's viability/success; this is particularly true for a new technology seeking to establish a foothold against incumbent competitors. In terms of our model, we allowed market development to oscillate from "boom to bust" over 5-year cycles and compared behaviour with the base case.

As expected, results in Figure 3.2 demonstrate that the number of ITF providers is very sensitive to such fluctuations, varying in sympathy with investment fluctuations due to a strong feedback via hype to perceived usefulness. Our model suggests that the corresponding effect on ITF solutions is much less marked. Varying investment only influences ITF solutions via secondary feedback through end-user competition concerns to the regulator. At a deeper level, this emphasises that judgements on how investment affects providers and solutions (simultaneously) will be crucial to both modelling (where the coupling between such influences must be parameterised) and ultimate PSIRP success/failure.

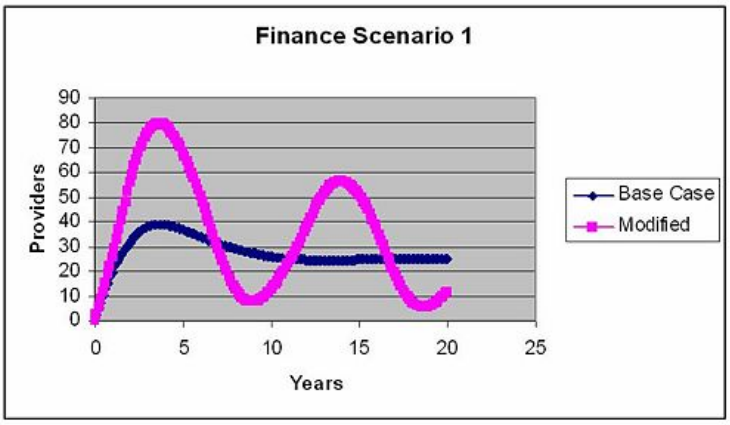

Figure 3.2: Finance Scenario

With regard to Scenario 2, both memory and processing limits might represent serious technological causes for concern [Mey07] in rolling out any new technology. Historically, Moore's Law tended to ensure technology scaled at rates surpassing the growth rate of information but the Law (arguably) does not apply to building high-end routers; growth in resources available to routers could eventually slow down and may even stop, while network demand continues to grow. It seems intuitive that, as a new entrant, PSIRP will be particularly vulnerable to any sustained technological slow-down. To explore likely consequences within our model, we modified technology development to simulate a progressive breakdown of Moore's Law in 5-year discontinuous bursts, once again comparing with the base case.

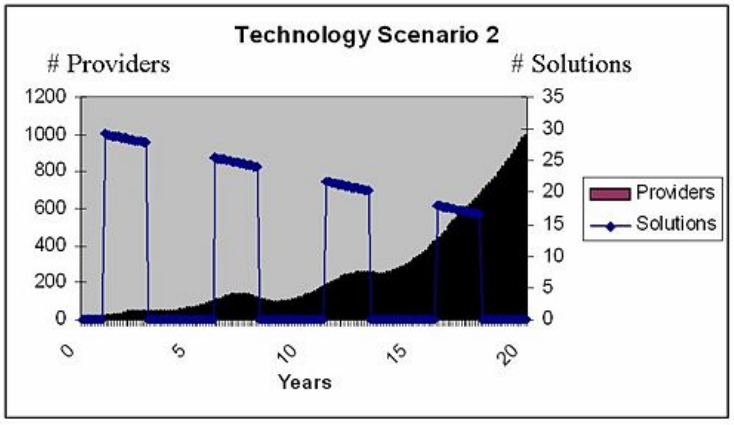

Figure 3.3: Technology Scenario

Results in Figure 3.3 show a strong tendency for the number of ITF providers to rise (exponentially), modulated by the periodic burstiness in technology flow. In our model, decreasing the flow of ITF solutions lessens regulatory feedback control exercised over ITF providers, which experience a surge in growth due to the investment stimulus. This emphasises the critical interaction between positive market feedback and negative regulatory feedback, as mentioned above. In a practical situation, it would be necessary for the regulator to react more quickly to changing socio-economic conditions [Vai09].

Our experience with the Finance and Technology studies provides strong confidence that the model successfully captures system behaviour in an intuitive fashion. We now consider the more PSIRP-specific Routing and Privacy scenarios.

3.1.1.2 (Inter-domain) Routing Scenario

With regard to inter-domain routing and connectivity, PSIRP essentially offers the possibility for providing high-quality (VPN-like) routes at a (premium) price through dedicated ITF

providers. In this scenario, we investigate the trade-off between likely performance (e.g., delay) improvement [Quo07, Raj08] and price, to better understand the market for such services and implications for ITF design decisions.

There is a general regulatory trend away from legalistic requirements towards more co-operative regimes with shared/devolved responsibility. Previous modelling work [Vai09] has tended to reflect this, focusing on how telecommunication regulation must become more flexible in the face of potentially disruptive technology changes. Failure to do so will inevitably compromise the delicate balance between regulatory control and innovation. From a provider viewpoint, we also note the trend towards partial transit, paid peering and multi-homing, which could significantly affect ITF design choices; similarly, net neutrality remains a hot topic with potentially significant implications for network access regulation.

To study these effects within our model, we introduced an ITF advantage reflecting how improved PSIRP performance would be expected to increase perceived usefulness of an ITF service. In principle, the ITF advantage is itself dependent on many other factors, as depicted in Figure 3.4. Apart from direct performance improvement, these might include user incentive to interconnect and specific PSIRP architectural design choices.

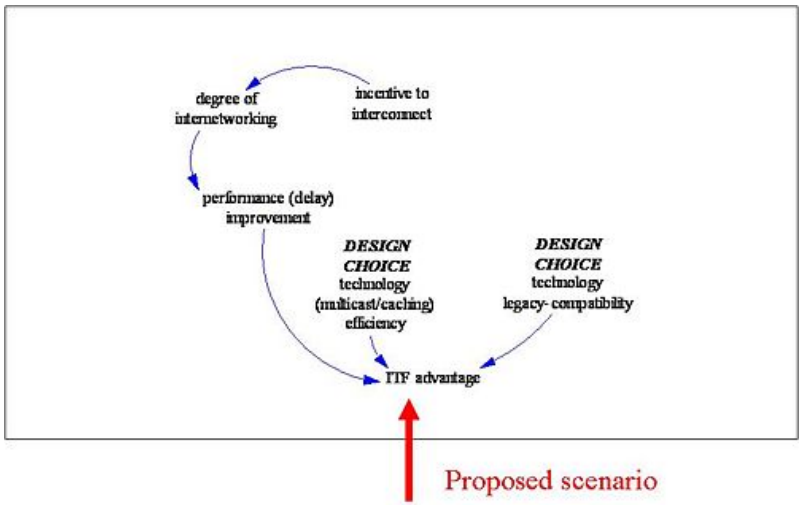

Figure 3.4: Routing Stocks and Flows

For modelling purposes, we assumed an initial take-up as users became increasingly prepared to pay more for better service, eventually levelling off and then terminating over a more extended period as users gradually came to regard such improvements as “normal service”. As shown in Figure 3.5, the number of ITF providers behaves qualitatively as in the base case (Figure 3.2), first rising then falling back towards a regulatory plateau. However, in the longer-term the market becomes increasingly sensitive to variations in ITF advantage with less evidence of a stable plateau as ITF advantage falls towards zero. Once again, the critical interaction is between positive market feedback and negative regulatory feedback, as already emphasised in Technology Scenario 2.

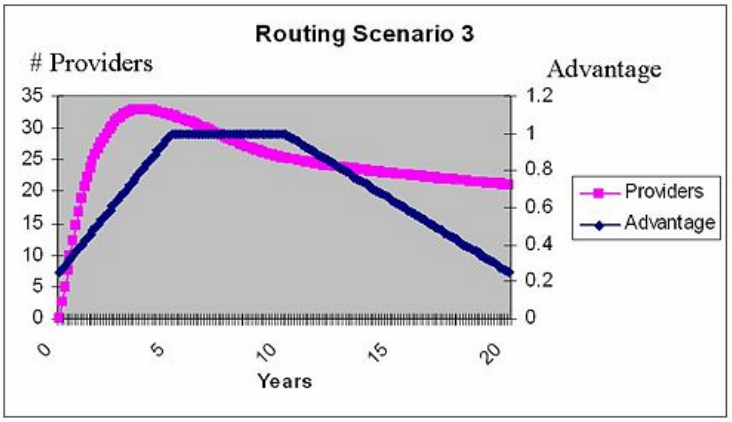

Figure 3.5: Routing Scenario

3.1.1.3 Privacy Scenario

This scenario explores tradeoffs between the importance of the various user choices and consequences for PSIRP ITF design decisions, such as the balance between source routing and topology hiding (emphasising importance of pub/sub and ISP choice, respectively). In particular, we investigated a situation where pub/sub choice was paramount due to customer anger at a series of privacy lapses, leading to adverse ITF publicity and consequent user-backlash.

In general, the dramatic increase in computing power, bandwidth and storage capacity has radically increased the ability of organisations to collect, store and process personal data. This is a potential cause for concern [Bro09]. On the one hand, new technologies like ubiquitous computing, surveillance technologies, biometrics, behavioural advertising, or social networking provide a hitherto unknown capability for eroding privacy. On the other hand, general social and political fears of terrorism or organised crime may drive both public and private authorities to make use of these possibilities. Overall, these developments are generally thought to pose a serious challenge to existing privacy laws and principles. Cybercrime remains a major issue for policymakers and law enforcement agencies. Besides problems with fraud, key concerns include malware, spam and cyberwar attacks.

As an obvious privacy-related example, we might mention eHealth and telemedicine applications. While the medical and economic benefits of integrated health information systems may be substantial, the usual public policy concerns associated with large-scale information systems apply. Given the extraordinary sensitivity of personal health data, special attention must be given to issues of privacy and IT security. A key challenge will be to make the best possible use of eHealth technologies for the benefit of the patient while complying with local privacy and security regulations.

Within our model, the significance of design choices (PUB/SUB for our chosen scenario) is measured by their compatibility with the current socio-economic situation. Once again, each choice is itself dependent on many other factors, as shown in Figure 3.6 where PUB/SUB choice will depend on a mix of user concerns for security and trust, access regulation and the user-ISP tussle over routing control [Lak04].

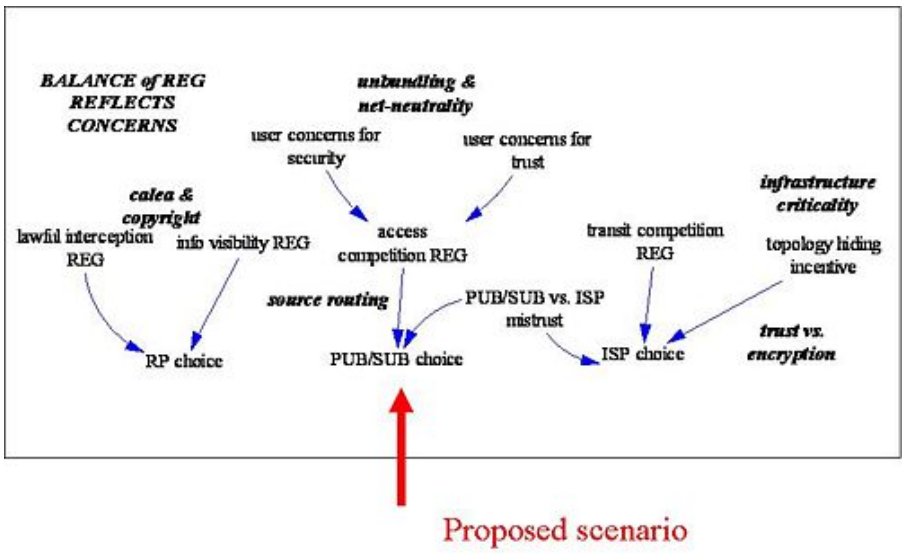

Figure 3.6: Privacy Stocks and Flows

In modelling such a scenario, we parameterised the pub/sub choice assuming a series of major privacy/security incidents occurring every few years, each lasting several months. As expected and shown in Figure 3.7, the number of ITF providers behaves similarly to the base case. However, longer-term behaviour is heavily modulated by fluctuations in regulatory pressures, caused by user-backlash. Our model suggests this is at least a 10-20% effect. If regulatory effects were actually somewhat weaker than assumed here, market forces would ultimately cause the number of ITF providers to rise exponentially, as observed in Technology Scenario 2. Again, this emphasises the critical interaction between competing positive market feedback and negative regulatory feedback.

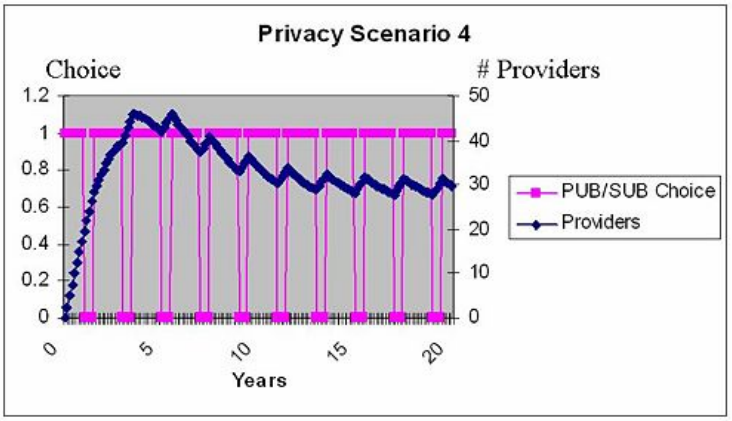

Figure 3.7: Privacy Scenario

3.1.1.4 Summary

In light of the modelling and analysis carried out, the following broad conclusions may be drawn:

- Our model successfully captures system behaviour in an intuitive fashion; in a typical scenario (Figure 3.2 base case), the number of ITF providers initially rises and then falls back towards a regulatory plateau.
- The market is very sensitive to investment fluctuations due to strong feedback via hype to perceived usefulness; our model suggests that the corresponding effect on ITF solutions is much less marked; more generally, judgements on how investment might

affect both providers and solutions simultaneously will be crucial to PSIRP success/failure.

- The competition between market and regulatory feedback is critical; our model suggests this is at least a 10-20% effect (e.g., Privacy Scenario), underlining the importance of tuning ITF architectural design choices to the prevailing socio-economic situation.
- If regulatory effects were relatively weaker than assumed here, market forces (reinforced by ITF advantage) would ultimately cause the number of ITF providers to rise exponentially.
- Similarly, the regulator must be ready to react sufficiently rapidly to changing socio-economic conditions (e.g., user-backlash)

The overall conclusion must be that the ideal ITF architectural design will represent a compromise amongst all these influences, with particular emphasis on security/privacy aspects. Accordingly, security considerations are discussed in more detail below.

3.1.2 Security Considerations

In previous deliverables, e.g., [PSI10], the role of the ITF, or Inter-Domain Topology Formation function was explained. This function takes information about the location of subscribers, together with network information and policies and preferences from different parties in order to choose a forwarding path (or tree). What is not clear is how the ITF interacts with other functions in the architecture. This section seeks to provide some preferable architecture options based upon an analysis of the information involved and potential security concerns, classified along the following dimensions:

- **Initiator**: The publisher interacts with the rendezvous function to obtain information about subscribers (such as their attachment network). This information may be returned to the publisher who then initiates the communication with the ITF. Alternatively the rendezvous system may initiate the contact with the ITF on behalf of the publisher. A final option is that the attachment network has an agent that communicates with the ITF. Such an agent could be the local network rendezvous or topology formation function, residing however within the same trust boundary as the local forwarding network.
- **Recipient**: Although we would usually assume that the recipient of information from the ITF might be the initiator of the forwarding path request, this may not be the case. For example, even if the publisher communicates with the ITF, the forwarding path may be returned only as far as the local forwarding network, with the remainder hidden from the publisher. The publisher is then given a path only as far as the forwarding network that holds the rest of the forwarding path information.
- **Control point(s)**: Where is the forwarding path actually selected? At one extreme the ITF may supply all relevant forwarding information leaving the choice and construction of the forwarding path to other parties (such as the publisher). At the other extreme the ITF will receive all concerns and policies and decide upon the best forwarding path. Intermediate ITF solutions may provide a restricted range of forwarding information or paths, leaving the final decision to another party (such as the publisher). In this case the ITF trusts the publisher (or other party) with more information, but the publisher may hide some of their concerns and eventual choices from the ITF.
- **Level of client information**: Different levels of information may be returned by the ITF. Loosely we can consider that the ITF may return a single constructed path, multiple paths, or a collection of path segments. In the latter two cases the publisher (or other entity between the publisher and the ITF) is left to make the final path decision (or even use multiple simultaneous paths).

- **Level of network information**: It is not yet clear how end to end forwarding paths are expressed. Options include each forwarding network sharing zFilter paths across their network (that can then be aggregated to form an end-to-end path) or using virtual paths for transition routes across forwarding network. Such virtual path identifiers can then be replaced/supplemented by a local network zFilter. Other options may be to use AS identifiers or finer-grained waypoints within the networks. The level of information supplied by each forwarding network (via their internal topology management) to the ITF determines what level of information may be supplied onward to the publisher or other parties and may depend on the trust between the forwarding network and the ITF.

3.1.2.1 Information Issues

We shall now briefly address the various issues that arise by passing information between components of the architecture.

Information Passed to ITF from Forwarding Networks:

- Forwarding path information; this includes potential routes along with information about resilience, QoS, congestion etc.
- Forwarding path policies; these policies reflect a bias or restriction on the use or aggregation of the forwarding paths.

The forwarding network is assumed to have a trust relationship with the ITF provider that allows the ITF to establish policy conformant paths on behalf of the forwarding network. The ITF acts as a trusted point between multiple forwarding domains that are potentially competitors (while having to work together to form an end-to-end path). The forwarding route and policy information is competitive information and should not be shared with other forwarding networks. If forwarding network information is passed beyond the ITF it is assumed that the information will be de-sensitized (e.g. through aggregation or limited route choices) so that this information is no longer commercially sensitive to other forwarding networks. Detailed forwarding information can also potentially be used by the publishers to abuse or attack the network.

Information Passed to ITF (originating) from the Rendezvous System

- Subscriber location; this location information needs to match the level of the forwarding information; if the ITF is constructing routes between ASs, then the set of ASs covering the subscribers needs to be known; alternatively a set of waypoints (nearest to the subscribers) may be required instead.

The network locations of subscribers are confidential information between the subscriber and the rendezvous system. However, the ITF does not require subscriber identities or even their number within each forwarding network – only that some subscribers exist. Although this desensitizes the subscriber information it is not necessary for any party other than those exhibiting control over the forwarding path selection to have this information. Thus if the publisher is not controlling the path selection, such information would ideally only be shared with the ITF.

Subscribers may also have preferences and policies about how they want information to be delivered. Since PSIRP is a subscriber driven architecture it is preferable to allow subscribers to not only subscribe to information, but to have some say in how they want it delivered. This is technically difficult since there may be many subscribers and meeting the interests of all subscribers may not be in the interests of the system as a whole. For example, if different subscribers were to select different Quality-of-Service classes, then should all subscribers subsidise the high quality shared links nearer the publisher? Ultimately the issue is an economic one, but it also means that such subscriber information has to be communicated to the ITF to allow it to make the routing decision and facilitate accounting between the forwarding networks.

Information Passed to ITF from the Publisher

This information will include publisher credentials (which can be matched to ITF or forwarding network policies), along with the publisher's own policies or preferences that affect the chosen forwarding path(s). For example the publisher may request a specific QoS, or request multiple redundant paths with specified isolation properties. The requests of the publisher will often be confidential in terms of both the request and the path information passed back. Such information can be used to analyse the network activities of the publisher (e.g., destination networks, degree of multicast, subscriber churn, QoS preferences etc.).

Information Passed from the ITF

The ITF may propagate selected network path and policy information, or may pre-compute partial or full paths. Such information can be used to route packets across the network. Paths may be signed by the ITF stating that they conform to specific policies and are available only to a single publisher. This can prevent unintended parties from using the path. Depending upon the representation of the path, this information may also be used to derive network topology and policies and also to isolate partial path information that could then be used to form non-compliant paths. This latter misuse may be stopped by the lack of an ITF signature.

Partial path information may be subject to more abuses since it may be used to form non-compliant paths. The ITF will be unable to sign the complete path unless conversations occur between the controller (e.g., publisher) and the ITF. Furthermore, the more detailed the path information, the more knowledge an attacker will gain about network topology, and the more sensitive business policies will have to be shared beyond the ITF.

3.1.2.2 Security Issues

How the ITF communication is arranged depends largely upon two considerations:

- Who is trusted to see the various types of information?
- Who is trusted to make control decisions?

In choosing an ITF communication design, we are balancing the security requirements of multiple parties. For example, the publisher may prefer to see all forwarding information and make its own choices. However, the forwarding network will not trust the publisher with detailed forwarding or policy information or to adhere to its business policies. Fortunately there are a number of other actors (such as the ITF, rendezvous system or ISP/attachment network) that may act as trusted intermediaries, hiding information from each party and making trusted control decisions.

Forwarding Network – ITF – Publisher Tussle

Both the forwarding networks and the publishers have some trust relationship with the ITF. The forwarding network trusts the ITF to establish policy conformant routes, yet protect the forwarding network's confidential information. The publisher also expects the ITF to establish a best/fair route complying with its own preferences (which is true even if the ITF is hidden behind, or encapsulated within, a forwarding network or rendezvous service). However, the publisher may not trust the ITF with all of their preference information, seeking to make the final forwarding decision itself. This will create a tussle between the publishers and the forwarding networks, and it seems that the forwarding networks will hold the upper hand. The ITF will have no business if the forwarding networks do not share information with it, and they will not do so if their networks and businesses are compromised by the ITF sharing information with unknown publishers. On the other hand, it seems that a large majority of publishers will have little concern that the ITF, with whom they already have a certain level of trust, has visibility over the forwarding preferences.

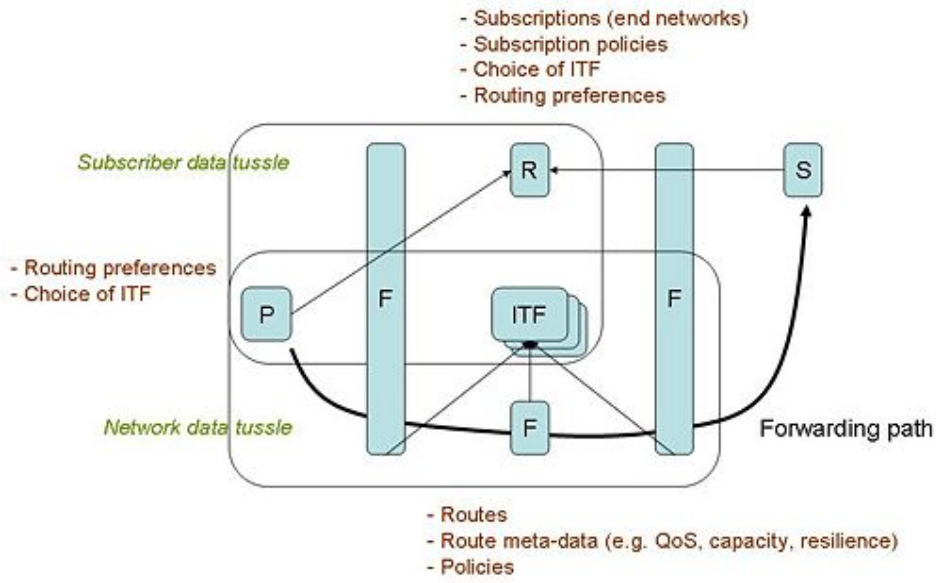

Figure 3.8: Tussles

This conflict is shown in Figure 3.8 as the 'network data tussle'. In this diagram a single ITF is shown interacting with a single publisher and subscriber. The rendezvous systems, for simplicity, are shown as a single component. In a real system the rendezvous function would be distributed and under the control of multiple rendezvous operators. The diagram also shows a limited number of forwarding network functions, under the assumption that the publisher and subscriber each have an ISP to which they perform network attachment. In reality these parties may multi-home and also operate or participate in other local networks with different degrees of trust to that of an ISP.

Perhaps a more compelling reason to share more fine-grained information with the publisher is to allow a more scalable and dynamic forwarding selection without continual recourse to the ITF.

Subscriber/Rendezvous – ITF – Publisher Tussle

Along with the tussle described above, there is a separate conflict involving the subscriber's information. This is shown as the 'subscriber data tussle' in Figure 3.8.

Ideally, subscriber information should not be shared with the publisher (or vice-versa) since a pub/sub network should provide decoupling between these parties. A publisher should be able to publish without knowing any details about the subscriber set, although a 'quench' control may be used to prevent the publisher wasting network resources if no subscribers exist.

Subscribers must share subscription information with one or more rendezvous systems, and also attach to one or more forwarding networks. It is likely that such a subscriber ISP network will also operate a local rendezvous component in order to be able to perform intra-domain routing and route inter-domain traffic to local subscribers.

At least summaries of subscriber locations (for example end network identifiers) must be communicated to the ITF in order to construct a forwarding path. This implies that the chosen ITF must be trusted by both the publisher (e.g. to provide a good forwarding path) and the subscriber/rendezvous system (not to reveal or analyse subscriber location and behaviour).

Another question is how trusted other elements, such as the publisher or publisher attachment network or ISP, may be to the subscriber (and rendezvous system on behalf of the subscriber). Such elements may be involved in conveying information from the rendezvous system to the ITF or may actually use such data (for example, if the publisher is involved in route construction/selection).

Although we have described the problem as the 'subscriber data tussle', we must also be aware of the concerns of the publisher. For example, even if the publisher did not choose the ITF it must trust it to hold confidential information on routing requests and provide routes without any subversion or prejudice. If the rendezvous system participates in the communication between the publisher and the ITF, then that rendezvous system must also be trusted by the publisher.

3.1.2.3 Design Patterns

The discussion above leads to the identification of several design patterns for the architectural components interacting with the ITF. In this section we present some of these basic patterns and discuss their merits. The first four patterns deal with the interaction between the rendezvous system and the ITF. The next three patterns then deal with the amount of information and control given away by the ITF. Thus, one of the first four patterns may be combined with one of the latter three to provide an overall design choice.

A1) **Trusted Publisher Pattern:** In this design, choice the publisher acts as the initiator and communication hub between the rendezvous system and the ITF. As shown in Figure 3.9, the publisher will interact in separate transactions with the rendezvous system, followed by the ITF. This has the obvious advantage that each transaction can be controlled separately and the publisher can detect and repeat any failed communication. In addition, no overall transaction state is held by the network (leading to potential scalability and DoS problems).

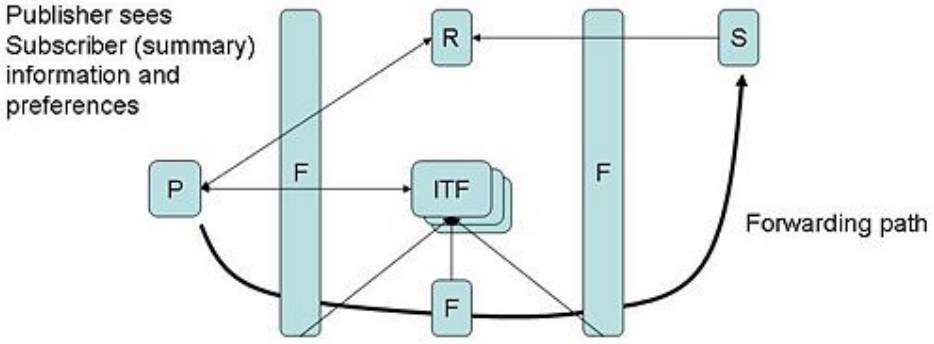

Figure 3.9: Trusted Publisher Pattern

The publisher obtains aggregate subscription information from the rendezvous system. This information is visible to the publisher (therefore, to some extent, compromising subscriber confidentiality). In addition the publisher may be expected to honour subscriber preferences (such as the choice of ITF or routing policies). This therefore implies that either the publisher is trusted to make such a decision, or that the community of ITF providers will insist on seeing aggregate Subscriber preference information signed by a trusted rendezvous provider (discrete per subscriber information signed by each subscriber would reveal too much subscriber information to the publisher).

The publisher passes the subscriber information along with its own preferences to the selected ITF. The ITF then calculates the forwarding path and passes the forwarding information back to the publisher.

A2) **Direct Rendezvous-to-ITF Pattern**: In this pattern, as illustrated in Figure 3.10, the publisher engages the rendezvous system, but the subscriber information is passed directly to the selected ITF without going back to the publisher.

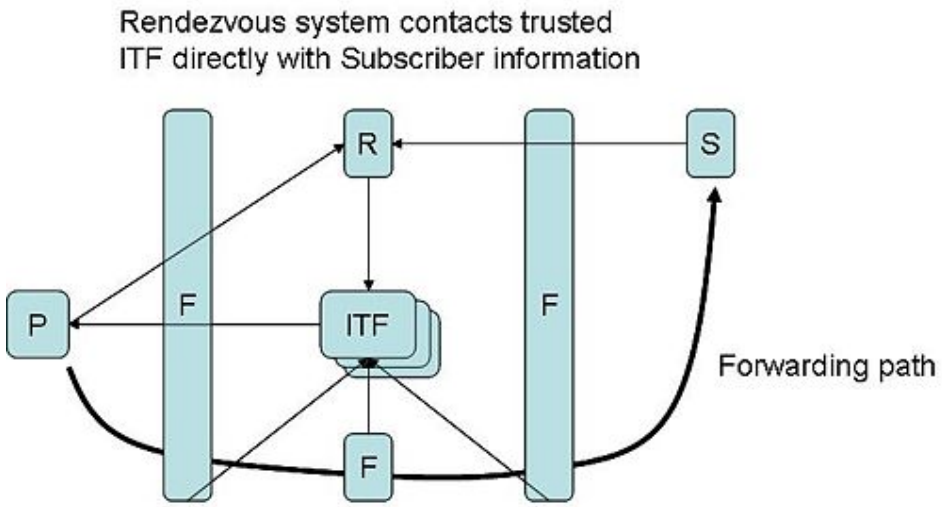

Figure 3.10: Direct Rendezvous-to-ITF Pattern

Although this obviously protects the subscriber Information from the publisher, the publisher must reveal routing preference information to the rendezvous system so that this can be passed along with the subscriber information to the ITF. In a slight variant, the publisher information may be encrypted so that it is visible only to the ITF. In any case the choice of ITF is revealed to the rendezvous system. If the publisher makes the ITF selection, then the ITF must be known and trusted by the rendezvous system (since otherwise the ITF may collude with the publisher). This pattern potentially has some problems due to the fact that the rendezvous-ITF interaction is hidden from the publisher. Thus, the publisher relies on the rendezvous system to manage communications failures to the ITF.

Return communications from the rendezvous system and the ITF can be passed directly to the publisher (through the publisher subscribing to such information).

A3) **Tunnelled-through-Publisher Pattern:** A variant on pattern A2 is to 'tunnel' the communication from the rendezvous system to the ITF via the publisher. This maintains the subscriber confidentiality but allows the publisher to control the transaction with the rendezvous system and ITF separately.

To achieve this type of interaction, the rendezvous system encrypts the subscriber information with the ITF key. Thus, the choice of ITF must still be known to and trusted by the rendezvous system. This pattern is shown in Figure 3.11.

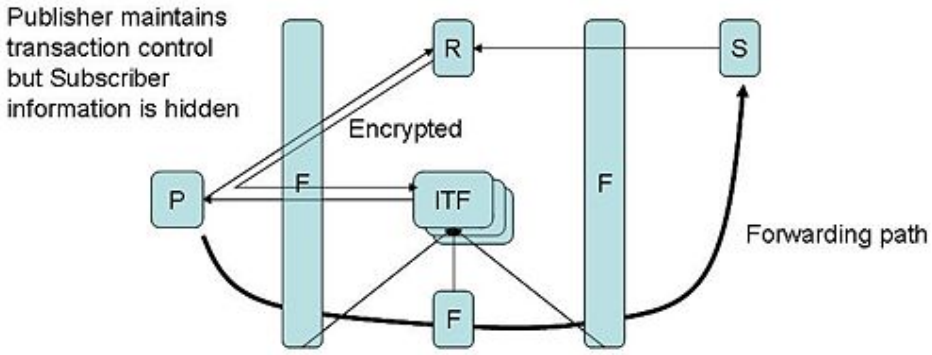

Figure 3.11: Tunnelled-through-Publisher Pattern

A4) **Forwarding Broker Pattern**: If patterns A1, A2, And A3 are all unsuitable because of the lack of trust between the publisher, rendezvous and ITF, we can introduce a broker that is trusted by all these components. Since the ISP forwarding network (potentially) already carries the communication between these components, it is a small step to involve it in the rendezvous and ITF functions.

In this pattern, a forwarding network operator introduces a rendezvous and ITF broker function. Although there may be several forwarding networks between the publisher, the rendezvous and the ITF systems, the forwarding network chosen for this task should be trusted by the rendezvous and ITF and used for communications to both of these functions. Therefore, such a forwarding network may be the publisher's ISP.

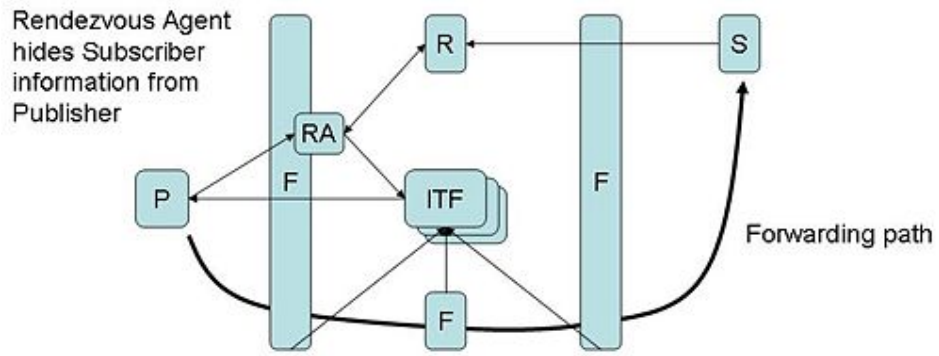

Figure 3.12: Forwarding Broker Pattern

As illustrated in Figure 3.12, the publisher is interacting with the rendezvous system via a rendezvous agent that resides in the forwarding network. This may already be a natural communication pattern since the local ISP is likely to host a local rendezvous system that can also take on the role of the broker between the ITF and "foreign" rendezvous systems.

B1) **Control at the ITF Pattern**: Once the ITF has received information about the subscribers, along with publisher preferences, it can use the information received from the forwarding networks to make route selections and construct the forwarding path (e.g., a zFilter). If the control is maintained by the ITF, then the publisher will receive a zFilter suitable for inter-domain forwarding to its subscribers.

This model will be used where the publisher is not trusted with finer grained forwarding information (for example partial paths or multiple paths).

B2) **Control at the Publisher Pattern:** In this model, the publisher is trusted to share the route selection with the ITF. The ITF will perform some initial route candidate selection from the wealth of forwarding network information. It will then pass the route candidates or partial computed paths to the publisher. The publisher is then free to construct or choose between the final paths.

This model reveals more information to the publisher, but conversely allows the publisher to hide some final route selection criteria from the ITF (e.g., choice of route to avoid particular forwarding networks). It also allows the publisher to respond dynamically by adjusting its forwarding paths without recourse to the ITF for every route adjustment (for example, allowing handover between multiple paths).

B3) **Control in the Forwarding Network Pattern**: Similarly to pattern A4, the forwarding network can operate some part of the route selection function. At one extreme this can involve not passing any route information to the publisher at all (if the publisher is not even trusted to see the inter-domain zFilter), holding the forwarding path information in a topology agent and

providing the publisher with only a link to the topology agent. In other variants, meaningless identifiers can be provided to allow the publisher to choose alternative routes without exposing the zFilter.

Such a topology agent must be trusted by the ITF to hold the route information as well as be trusted by the publisher to make the correct route decisions on its behalf. This pattern is shown in Figure 3.13.

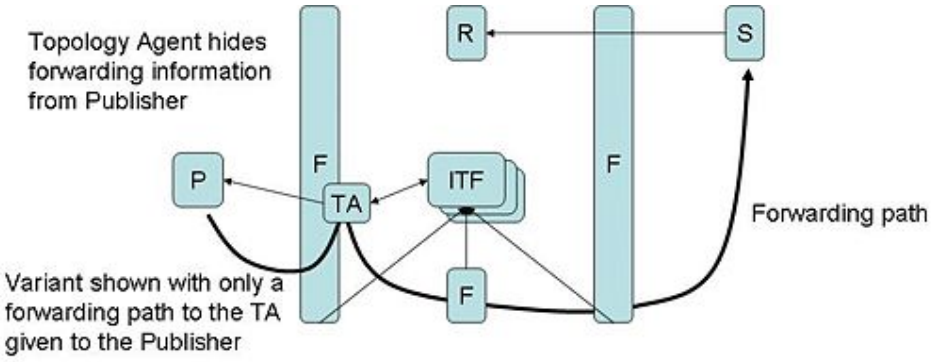

Figure 3.13: Control in the Forwarding Network

The patterns above illustrate some simple design choices in order to understand and discuss architectural options for implementing an ITF function. These are intended to serve as starting and discussion points rather than final designs. It is important to note that any final design does not have to conform to a single pattern but may provide a hybrid approach. Thus for an architecture that is designed for both trusted and untrusted publishers, we might provide an optional rendezvous and topology Agent in the forwarding network. In any hybrid approach, the discussion then focuses on who has control of which option is taken during run-time.

In the following section, we elaborate on one hybrid model that caters to a wide range of different trust relationships and forms our preferred design for a general-purpose (e.g., Internet) PSIRP architecture with both business and consumer services.

3.1.3 Design Choices

3.1.3.1 Possible PSIRP Design Candidate

As hinted above, we have developed a preferred candidate that uses the concepts of a rendezvous and topology agents operated by a forwarding network (that is trusted by the rendezvous systems and ITF). This is shown in Figure 3.14. In this example (for simplicity), we assume that a single forwarding network operator can be found to meet the trust requirements of the rendezvous systems and the selected ITF, although in reality the rendezvous and topology agent can be operated in different forwarding networks with a peering arrangement and trust relationship.

This design uses both of the options presented in patterns A4 and B3 to allow the forwarding network operator to act as an intermediary between the publisher and both the rendezvous and ITF systems. Such a forwarding network may be an ISP and is likely to have a contractual or other direct relationship with the publisher that allows the forwarding network to select the mode to operate in. It should also be a large enough commercial organisation to make the establishment of trust relationships with rendezvous and ITF providers feasible.

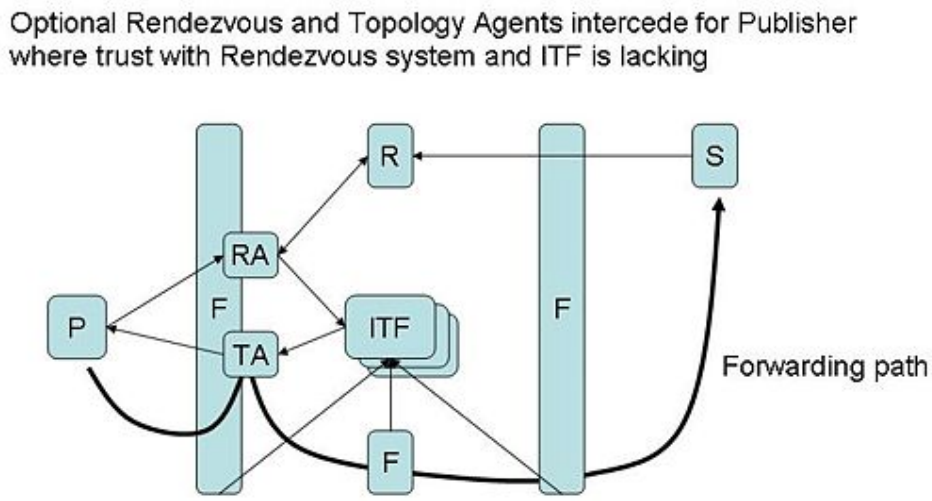

Figure 3.14: Design Candidate

Technically, the choice of forwarding network in which to operate the trusted intermediary Agents can be made at run-time in two ways. The first method involves manual selection by the publisher. If the rendezvous system or the ITF refuses to directly interact with the publisher, the publisher can route its communications through a forwarding network that has the appropriate level of trust. This forwarding network does not have to be the attachment network or ISP, but may provide rendezvous identifiers to enable remote publishers to connect to its rendezvous or topology agents. This presents a bootstrapping problem since it assumes that the publisher is able to create a forwarding path to such agents. This can be overcome by using a rendezvous system for the agents, which freely grants the locations and subscriptions of these Agents and an ITF, which likewise openly constructs forwarding paths to such agents.

The second approach uses more automation in the network. The publisher sends its rendezvous requests to a local forwarding network whose rendezvous agent makes a decision about whether it is able to handle such a request. If it believes it is not trusted to receive all subscriber information from the rendezvous systems it may forward the request to another network. This approach is not so different to the layered rendezvous architecture of PSIRP, with the exception that results will not be returned to the publisher, but instead only returned back as far as there is sufficient trust. This may then result in multiple rendezvous agents at different levels into the rendezvous network holding subscriber results on behalf of the publisher, which they are unable to pass on. These results may be sent separately to the ITF.

Both the manual and automatic modes can work together with the publisher making the first choice of forwarding network rendezvous agent, and then the request being cascaded further into the rendezvous network.

In the request to the rendezvous agent, the publisher includes a choice of ITF. Alternatively, the rendezvous agent may insert its choice if no explicit choice is made by the publisher. If the rendezvous agent is operated by the ISP, it is likely to be in a good position to choose a suitable ITF with whom they have a good trust relationship. If the request is passed to further rendezvous agents, then the choice of ITF must be respected by these agents (as otherwise subscriber information would be sent to different ITFs). In this design, we make the assumption that the publisher will have control over the ITF choice since otherwise we would have to negotiate between the preferences of multiple subscribers.

Any rendezvous agent that receives a request from a publisher will first gather as much subscriber information as possible from the available rendezvous systems. It then contacts the ITF directly and passes to it both the subscriber information and the publisher preferences. Along with this information, the rendezvous agent includes a method to contact its preferred topology agent.

The rendezvous agent can also act as a cache and aggregator. For example, if a Publisher request is already "covered" by previous publisher requests, then it may not be necessary to involve the rendezvous systems. In addition, if a topology agent already has the required forwarding information, it may not even be necessary to contact the ITF.

The ITF decides how much it trusts the nominated topology agent. It then calculates one or more forwarding paths or path segments in order to fulfil the ITF request. This information is then passed to the nominated topology agent. As mentioned above, this topology agent does not need to reside in the same operator network as the rendezvous agent. For example, the topology agent may be nearer to the publisher (for example in the ISP network) for network efficiency.

Upon receiving the path information from the ITF, the topology agent decides how much it trusts the publisher, which originated the request (publisher credentials can be carried in the messages through the rendezvous agent and ITF). It then chooses between the following options:

- Not to service the publisher and to return an error message. This is unlikely since the rendezvous agent has already accepted the publisher request.
- To hold the path information returned from the ITF itself and provide the publisher with a separate path to the topology agent (which of course can be distributed throughout the network for scaling and path efficiency)
- To provide some or all of the path information to the publisher so that it can operate its own forwarding.

It should be noted that even if a publisher interacted with the rendezvous systems and ITF directly, it may still nominate a topology agent to hold forwarding path information on its behalf.

Once a forwarding path is established, we also need to consider how the path reacts to subscriber churn and mobility. Changes to subscriber information (in terms of destination networks) will arrive via the rendezvous systems to the publisher or rendezvous agent. These parties can then re-contact the ITF to request that the inter-domain path is updated. Changes to the path information are then sent by the ITF to the topology agent of the publisher.

If a topology agent holds forwarding path information, then we must also consider the retention of forwarding information. One mechanism would be for the publisher to express how long it requires the forwarding path (in terms of time or number of packets). The topology agent will grant the publisher an initial allowance and may then require additional refresh handshakes with the publisher to ensure that the forwarding state is still required.

If a publisher is mobile, the link provided to the topology agent may be updated without changing the rest of the forwarding information held by the topology agent. If the publisher roams too far, then the inter-domain path information may be transferred to another topology agent. Ultimately, the publisher can initiate a new request if it moves between networks.

3.1.3.2 Example Operation[2]

1. The publisher sends its intent to publish to the rendezvous agent in the forwarding network (ISP) it is attached to. This request may be passed along a chain of such rendezvous agents until it reaches one trusted by the rendezvous system to which the request refers. This request contains the publisher's preferences/policies for both the rendezvous and the ITF functions.
2. The rendezvous system adds subscriber information to the request and returns the message to the trusted rendezvous agent nearest to the publisher.

[2] Note that although the terms 'send' and 'receive' are used in the example below, these actually consist of a previous subscription from the receiver and a subsequent publication.

3. The rendezvous agent(s) pass the request to the ITF.
4. The ITF adds route information to the message and returns it to the topology agent.
5. The topology agent makes the final route decision. It constructs a forwarding path from the publisher to the topology agent and returns it to the publisher.

3.2 The Problem of Un-Subscription

In the PSIRP architecture as presented in D2.3, the rendezvous and forwarding functions are cleanly separated and may be operated by different business organisations. This offers advantages, such as being able to operate very efficient forwarding paths, but presents the complication that subscriptions and un-subscriptions are sent via the rendezvous function and are not visible or enforceable by the forwarding networks. While a publisher may legitimately decline to serve information in response to a subscription, it is not permissible that a publisher ignores an un-subscription request and continues to bombard the subscriber.

In this section, we investigate the problem further and look at architectural alternatives (and complementary approaches) to the problem of un-subscription.

3.2.1 Components in the Problem Space of Un-subscription

3.2.1.1 Rendezvous

The rendezvous function is responsible for registering subscriptions and providing attachment network information to the Inter-Domain Topology Formation (potentially via the publisher or publisher attachment network). As such the rendezvous function does not have information about the forwarding topology or the link identifiers that are used in the construction of the forwarding label (zFilter). Such information is shared by the forwarding networks with the topology manager for each domain, while inter-domain links are shared with one or more ITF operators.

3.2.1.2 Forwarding

The forwarding network is responsible for forwarding publications across its network to subscribers attached to it. When a publisher has already interacted with the Topology Manager, this process is as simple as matching the zFilter against outgoing link identifiers. In this mode, the forwarding elements have no knowledge of subscribers, but are merely forwarding traffic along a pre-calculated multicast tree.

Where traffic is arriving from another network, it is likely that the ITF does not have an intimate knowledge of the subscriber's attachment network. Therefore, finer-grained subscriber location information is required in order to perform the intra-domain forwarding. In order to perform this operation it is clear that the forwarding network needs to reference a local rendezvous function for subscribers attached to it. This does not preclude the use of additional foreign rendezvous providers to which the subscriber also register its subscriptions. The local rendezvous function will operate in conjunction with the local Topology Manager to construct the intra-domain forwarding tree. Such local subscriber location and/or the constructed intra-domain path are likely to be cached to improve the transmission performance of later packets.

The operation for a local publisher to obtain a forwarding path zFilter may be identical to that for inter-domain traffic. However, it may also vary since the publisher is on hand to negotiate forwarding route preferences and operate some of the path selection algorithm. Whereas a remote publisher forwarding across multiple domains may only have knowledge of the inter-domain path, a local subscriber may have knowledge of the intra domain path to local subscribers.

3.2.2 The Problem of Un-subscribing

If a subscriber chooses not to receive further information, it may unsubscribe from the information identifier. Like the subscription, this request is sent to one or more rendezvous providers, presumably including a local rendezvous function for the attachment network. Once the subscription state is removed from the rendezvous system, new publishers will not be able to acquire forwarding trees to these ex-subscribers.

The problem is simply that previous publishers may still have valid forwarding paths. This may be because they hold a valid zFilter, or because such forwarding state is still maintained in the network after the un-subscription (for example a cached intra-domain forwarding path for inter-domain traffic). This may result in a situation where the subscriber receives information in which it is no longer interested. Obviously, the situation can be exploited by malicious publishers for denial-of-service attacks or other types of abuse.

3.2.3 Approaches to Un-subscription

There are a number of potential approaches to ensuring that the subscriber is not subject to information overload or abuse (much) beyond the un-subscription.

3.2.3.1 Time-Limited Publisher Capabilities

The network may grant the ability to send traffic into the network for a limited time period. After the period expires, the publisher (or other network function) must reference the rendezvous system to check that the subscription is still valid before extending the right to publish. These capabilities may be checked at the network edge, but can also be integrated into the Packet Level Authentication scheme and checked at multiple enforcement points in the network. This approach is shown in Figure 3.15.

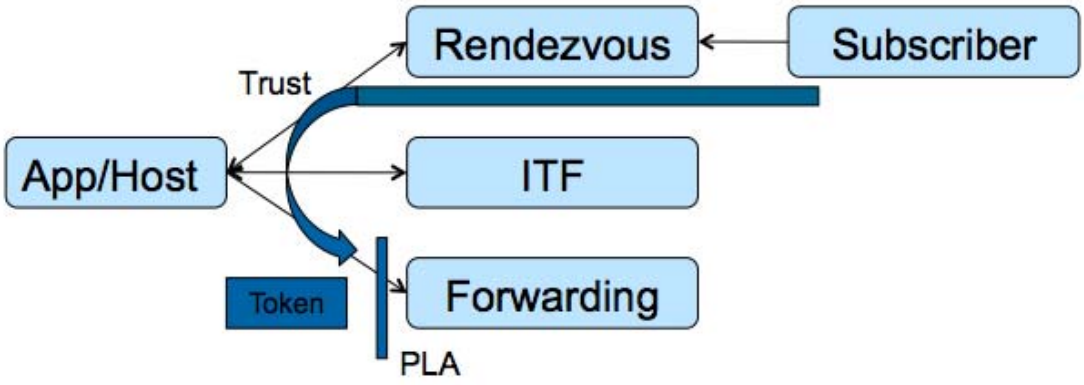

Figure 3.15: Rendezvous Network trusts Forwarding Network

The problem with this approach is that the time window may not be sufficiently small to protect the interests of the subscriber or sufficiently large to operate an efficient network. It is therefore desirable to complement a publisher capability approach with an immediate subscriber-side protection mechanism.

3.2.3.2 Time-Limited Network State

If forwarding state is held within the network (such as an intra-domain multicast path to subscribers), then this state must be refreshed at a minimal interval. This will apply, for example, to cached intra-domain forwarding tree information for inter-domain traffic as shown in Figure 3.16. It will also apply to network held inter-domain paths (for example, where the publisher is not trusted to hold or operate the inter-domain path selection).

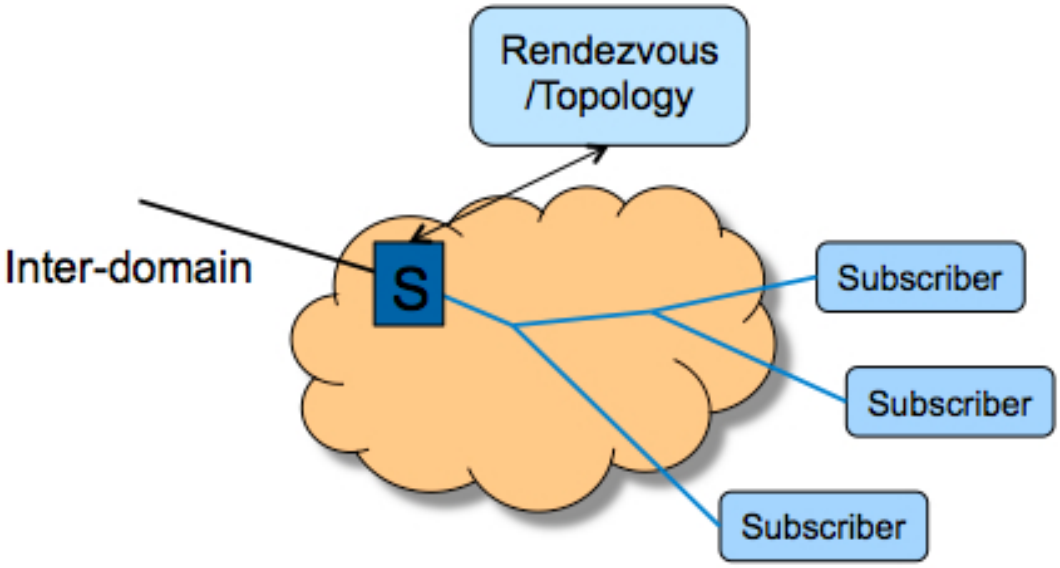

Figure 3.16: Rendezvous and Topology State Cached in Final Forwarding Network

The problems are the same as for time-limited publisher capabilities, except the result may still be that traffic is forwarded into the network before being dropped. At first it seems that this approach does not offer any benefit beyond the time-limited publisher capabilities. It should, however, be remembered that the local forwarding network is likely to have an immediate relationship with the subscriber and act on their behalf (as a direct customer). Limiting publisher capabilities closer to the publisher (such as at the publisher ISP) requires a chain of trust between the subscriber and publisher networks. We should also bear in mind that network state retention will have to be managed. Thus such a solution will have to be implemented for network state scalability.

3.2.3.3 Time-Limited Forwarding Identifiers

A slightly different solution is to change the Forwarding or Link identifiers used in the network at set (but not necessarily the same or synchronised) intervals. This will invalidate any forwarding path state held by either the publisher or by the network elements. These parties will then need to re-apply for a new forwarding path, which will not include any ex-subscribers. This approach may also be desirable to prevent attacks on the network that attempt to calculate the network topology and link identifiers.

One problem of operating this approach alone is that applications will need to detect packet loss before requesting a new forwarding path. This is wasteful in terms of network usage and may not be tolerable to certain applications. Thus it is likely to be combined with some sort of edge or publisher timer mechanism to request new forwarding paths before the old one becomes obsolete.

3.2.3.4 Triggered Un-subscription (Rendezvous)

The un-subscription request may be propagated through the rendezvous systems to those parties that requested subscriber information. These un-subscription notices need only be conveyed when the last subscriber leaves a location (for example a local broadcast network, or an attachment network for the inter-domain requests).

Trusted network components (such as a topology agent in the publisher's attachment network or the intra-domain routing agent for traffic arriving from inter-domain) will respect the 'cease

and desist' notice and recalculate the forwarding path. In most cases the publisher itself cannot be relied upon to behave appropriately when receiving an un-subscription request. Thus, as discussed in the section on Inter-Domain Topology Formation, a trusted agent can be used to hold the forwarding path state and refresh this as required by un-subscription requests. This is shown in Figure 3.17.

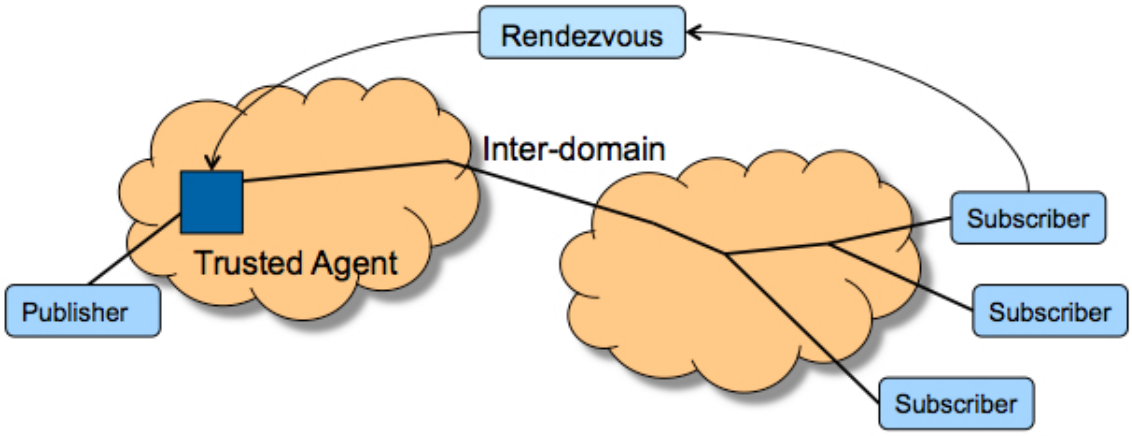

Figure 3.17: Use of Trusted Agent

Regardless of other mechanisms, un-subscription requests should always be sent to the rendezvous function in order to ensure that new publishers do not receive stale subscriber information (or that information is received in previous locations for mobile subscribers). In addition, subscriber information should be 'timed out' from the rendezvous system to protect rendezvous state since subscribers cannot always be relied upon to unsubscribe.

Problems include the holding of valid network forwarding state by untrusted parties (such as the publisher) and the delay in propagating the un-subscription notice through the rendezvous function.

3.2.3.5 Triggered Un-subscription (Forwarding)

Due to the delay in propagating an un-subscription request through the rendezvous system, we can consider that this should be combined with an un-subscription request sent to the forwarding network. This may result in either the Forwarding Identifier of a link changing (and hence traffic being dropped before reaching the subscriber) or the installation of a temporary block on a forwarding identifier (until such time as one of the other mechanisms described above takes over) as shown in Figure 3.18.

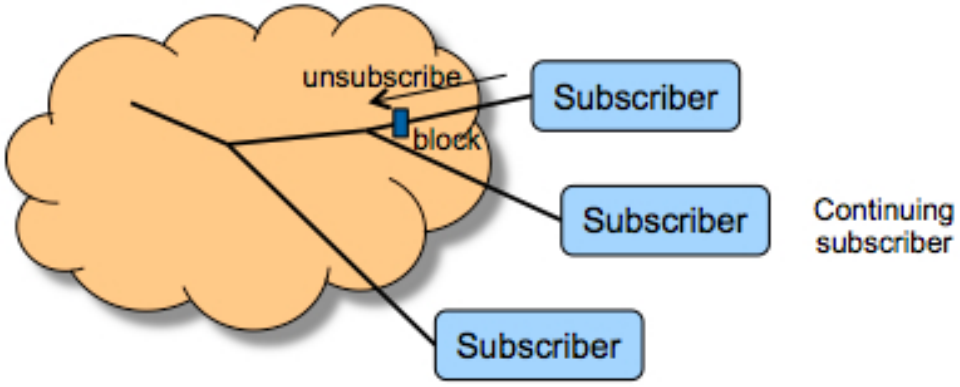

Figure 3.18: Un-subscription Request back into Forwarding Network

The advantage of this scheme is that it is immediate, although it may result in traffic being sent into the network that is later dropped (if there are no other live subscribers in that area of the network). In the next section we discuss how this may be achieved in more detail.

3.2.4 Unsubscribing Using a Reverse Forwarding Path

When a subscriber chooses to no longer receive information it will unsubscribe using the rendezvous identifier (and scope identifier) to the rendezvous function. This function should encompass all rendezvous systems, which previously held the subscription registration and should include a local rendezvous system for the local attachment network. This de-registration from the rendezvous function is required to ensure that new publishers do not include the ex-subscriber in their path formation.

At the same time the subscriber informs the local forwarding node (e.g. the attachment point) of their instruction to unsubscribe. This assumes that the unsubscribing host is not multi-homed, in which case the hosts should wait for a publication packet to arrive in order to identify to which attachment forwarding node to send the un-subscription request (although it is possible to flood the un-subscription request to forwarding nodes that may never see a publication). The local forwarding node cannot simply change the forwarding identifier used by the subscriber since other subscribers may also be present (in a broadcast network). In addition the same forwarding identifier may also be used for different items of information that the subscriber still wishes to receive.

Ideally, the forwarding identifier would become obsolete and the remaining subscriptions (for either the ex-subscriber or other hosts) would be transferred to different forwarding identifiers. However, since this would involve co-ordinating publishers and/or network forwarding state (e.g., in topology managers) this would not be any more efficient than the un-subscription via the rendezvous system. As a local (forwarding network) solution that can react more quickly to unsubscribe requests, we can consider that traffic is bridged locally to new forwarding identifiers or that information blocks are installed into the forwarding network.

If the final link is a broadcast network, there is no advantage in changing the forwarding identifier if other subscribers remain, since the ex-subscriber may gain the same benefits by not listening to the broadcast. However, if no other subscribers remain, or the host has a dedicated link, then changing the forwarding identifier will stop unwanted traffic from arriving at the host interface. In order to achieve this, the attachment forwarding node may be subscription aware, managing a tally of current subscribers against rendezvous identifiers.

Instead of storing subscription state, an alternative would be to allow subscribers to co-ordinate. Thus a subscriber who sends an unsubscribe message to the forwarding node may be overruled by another subscriber asserting that the broadcast forwarding identifier should still be maintained. This would avoid the requirement for subscription state in the forwarding elements, although it may cause problems for subscribers that have a poor or intermittent edge connection. Once it is determined that no other subscriber sharing the forwarding identifier is receiving information for the rendezvous identifier, then the forwarding identifier may be removed. If any subscribers are using the same forwarding identifier for other items of information (via different rendezvous identifiers) then they must be informed of a new forwarding identifier set to listen for their remaining subscriptions.

To do so, the edge forwarding node may replace the forwarding identifier and re-attach the subscribers to this new identifier. It must then direct all traffic not for the unsubscribed rendezvous identifier, but with a zFilter matching the original forwarding identifier over the new forwarding identifier.

A largely equivalent technique would simply be to leave the forwarding identifier but to install a block for traffic bearing the unsubscribed rendezvous identifier. One problem is that this block cannot be propagated into the network beyond the point where other continuing subscribers exist. Another problem is that new subscriptions would also need to be propagated into the network in order to remove the block (or move it to the point where it only applies to the

unsubscribed party). Where the block may be propagated along a reverse path created by received packets, there is no way for a new subscription to be propagated, other than to flood the network (to a determined number of hops) since the new subscription may not be entering the network at a point where it is known that there is an upstream block.

As an alternative to using the rendezvous identifier to block traffic we can also consider a partial un-subscription where the zFilter is used as the blocking pattern. In this case, the subscriber is electing to unsubscribe from information sent from a particular network path instead of information mapped to a rendezvous identifier. Although this seems interesting at first (e.g., as a way of mitigating against denial-of-service attacks), it is worth noting that for inter-domain traffic the zFilter may be identical for multiple publishers attached to the same ISP (if the inter-domain routing is visible) or even for all traffic arriving on the local forwarding network at the same peering point (if the inter-domain path is removed at the final domain). This approach still shares the same problems of how to ensure that such blocks do not affect other (current and future) subscribers.

Any blocks or redirects have to be maintained until either a subscriber (re)joins the rendezvous identifier, or until a time window elapses. This window should be sufficiently long so that it can support one of the other un-subscription mechanisms discussed earlier.

Blocking unsubscribed traffic at the edge of the network, although beneficial to the unsubscribed host, does not benefit the forwarding network. Ideally the traffic would be pruned as close to the publisher as possible. Thus, un-subscription notices may be propagated backwards into the network. The problem is that this cannot be performed (without wasteful flooding) until publications arrive, since there is no knowledge in the network of where publications about a particular rendezvous identifier may originate. When a publication arrives at an edge forwarding node (or end host) which currently has an un-subscription block in place (for all matching forwarding identifiers in the zFilter), the node may propagate the un-subscription request back into the network. Each forwarding node along the reverse path must check to see if it has other potential subscribers and only propagate the un-subscription request further if it has received un-subscription requests from all forwarding branches that match the zFilter. Since new subscriptions must be flooded the blocks should only be propagated in a predetermined number of hops.

3.2.4.1 Using composable and RId dependent FIds

In-packet Bloom filters (iBF) can be used to create the following two properties for forwarding identifiers:

1. The forwarding identifiers can be expressed as unicast paths and the source can combine those into a single (or multiple) multicast tree(s) by bitwise ORing them together.

2. The forwarding identifier may be tied to a particular rendezvous identifier for a specified time period.

The latter is accomplished by each forwarding element having a periodically changing key that is used, in combination with the RId, to compute the identifiers used to making local forwarding decisions. The identifiers can be created in such a way that there is a return channel on the forwarding path, which the receivers can use to replenish the subscription. This approach reduces the scope of rendezvous into a facilitator for initiating communications between publisher and subscriber. The rendezvous itself does not need to maintain state for on-going communications.

The composability of FIds ensures that the subscribers do not depend on each other, and the use of in-packet Bloom filters means that no publication/subscription related state is required in the forwarding elements. However, in the face of a denial of service attacker, the un-subscription may either take some time (for the forwarding elements to change their keys), or a helper function in the network will be needed. The purpose of the trusted network based helper function would be to gather the subscriptions and perform the composition. Then, it

would forward the composition concatenated with a set of zero's and encrypted to the publisher. Hence, once the helper function receives an unsubscribe request, it can remove the receiver from the composition before forwarding the new forwarding id to the publisher.

3.2.5 Discussion

The PSIRP network should always operate un-subscription via the rendezvous system since the purpose of the rendezvous system is to maintain rendezvous state. However, this alone is not sufficient since (a) the publisher may not be trusted to cease activity and (b) the latency may be unsuitable for some applications. The use of either time-limited publisher capabilities or time limited network state or forwarding identifiers is an effective remedy against (a). The use of time-limited publisher capabilities can be combined with the Packet-Level Authentication work and the security framework already reported in D2.4. This approach also stops the publication as early as possible without consuming network resources. Changing the forwarding identifiers provides a second line of defence and prevents targeted network attacks that require knowledge of the forwarding identifier topology while network state will have to be time-limited to avoid state scalability problems.

A simpler solution would always be to ensure that a trusted part of the network will respond to the un-subscription through the rendezvous system. This may result in the immediate re-creation of the local network forwarding path. In addition, we have already described in the section on ITF operation how local agents within the forwarding network may increase the level of trust in publisher operations. Such a topology agent can also be employed to reliably react to un-subscription requests.

The reverse forwarding path un-subscription request, described above, can be used as a short-term mechanism to provide low latency un-subscription response. It does not provide a complete solution since to do so would require the propagation of un-subscription requests through the network to the publisher and the maintenance of long-term (un)subscription state in the forwarding network. Even in this case it cannot prevent the first packet arriving from the publisher. Although we have discussed that it is not feasible to propagate information blocks very far into the network, this may not be such a drawback if the density of the remaining subscribers is high.

3.2.6 Unresolved Weaknesses

Even if all the mechanisms above are operated, there remains a limited opportunity to perform attacks. Distributed Denial-of-Service attacks may still overwhelm the ability of an end-host to unsubscribe. Although each separate attack stream may be quenched (if the host has the capacity to do so fast enough) the un-subscription blocks may not be carried deep enough into the network to provide uncongested routes for legitimate traffic. This is particularly a problem if other nearby hosts fail to react to the attack and are therefore still considered to be valid subscribers. Another concern is that any security mechanism is itself a means for attack. Thus, attackers posing as subscribers may attempt to overload the un-subscription mechanisms by selectively subscribing/unsubscribing and causing cascades of control messages through the network. Indeed, bogus subscribers may send unsubscribe notifications to the forwarding network since it has no direct knowledge of authorised subscriptions (although it has authorised attachment) in an attempt to overwhelm the edge forwarding node with un-subscription state. A malicious publisher acting in concert can then target nodes deeper into the network. Adding subscription validity checks would detract from the low latency operation of the forwarding network un-subscription mechanism and only provide an alternative opportunity for attacks.

3.3 zFormation

The zFilters are vulnerable to replay attacks, in which an attacker uses a given zFilter for traffic it was not meant for. Second, it is also possible to perform a correlation attack by combining knowledge from several zFilters to compute information about link identifiers. This

will enable an attacker to guess at least a partial path to a target it has not been authorized to send to. Finally, an attacker may be able to inject traffic to an existing zFilter if he can guess (or compute) a zFilter that leads to the path described by the zFilter.

To solve the abovementioned problems, Bloom filters can also be built using dynamic link identifiers or *edge pair labels* [Est09]. Instead of using a static forwarding table, which indicates the LIT for a given outgoing link, the router has a local secret key K and an enumeration of its neighbours. It uses these to compute an edge pair label with a function Z(K, #in, #out, F) where F is information from the packet, e.g., RId, and Z a secure cryptographic function that is fast to compute (e.g., a spreading hash function). The dynamic computation of link identifiers makes it possible to have secure forwarding identifiers that depend on the RId on the packet. Hence, a node cannot use a given FId, RId pair for sending packets with some other RId. Additionally, both incoming (#in) and outgoing link (#out) are used for computing the edge-pair label. This makes it more difficult for the attacker to perform an injection attack.

This approach still leaves a slight possibility for combining existing zFilters that describe crossing paths, assuming that the attacker is capable of getting zFilters with a chosen F (e.g. between attacking nodes). This can be prevented, if each router performs a secure bit permutation on the zFilter of every packet it forwards. The bit permutation needs to depend on F and local secret key to prevent an attacker from using a correlation attack to gain information on the permutations used. The effect of using bit permutations is to prevent an attacker from combining zFilters that have a different root in the network.

3.4 Identifiers

In this section, we address two issues related to identifiers in the context of the PSIRP architecture. The first relates to general design considerations for identifiers while the second focuses on the specific usage of so-called algorithmic identifiers [PSI09].

3.4.1 Design Considerations for Identifiers

While there is possibly a longer list of design considerations for identifiers, the following section focuses on two crucial considerations, namely those for long-term as well as variable-length identifiers.

3.4.1.1 Long-term Identifiers

Using rendezvous and scope identifiers with the P:L structure (similar to DONA, see also [PSI10]) as long-term identifiers is problematic, since the security mechanism implementation is coupled with the identifier itself in the form of a public key that can be compromised or lost.

The probability of key leakage can be reduced with delegation. In that case, the master key, which is part of the identifier, is stored offline and delegated keys are used for the actual communication. Security may also be improved by storing cryptographic keys into a secure separate hardware store inside the node. Therefore, even if the node is compromised, the attacker could not read the corresponding private key.

However, the abovementioned solutions are not completely sound, and therefore they are not suitable for applications that use the network level identifiers as the sole identity of a persistent entity. For example, in a distributed application, the identifier may be scattered in multiple systems and changing the identifier may become unmanageable. One solution is to use the application identifier as a long-term identifier, and treat SIds and RIds as more or less temporary network level secure identifiers. Even if the RId is changed, a resolution service with an orthogonal security mechanism (such as manually configured trust hierarchy) would resolve the application identifier to another RId. This is analogous to using DNS names instead of IP addresses as a way to permanently refer to the servers in the current Internet.

3.4.1.2 Variable-length Identifiers

In PSIRP, RId and SId form the identification means for the narrow "waist" of the architecture and they are used by all functions of the system to identify publications and scopes. Because the basic identifiers are fixed in length, it is not possible to encode arbitrary semantic information in them. In some cases, when the structure of the information is known beforehand, it is possible to use algorithmic ids (AlgIds) to generate the RIds on the subscriber side, but in general, variable-length identifiers are needed. Of course, it would be possible to implement the functionality of variable-length identifiers using only fixed length identifiers, but this would require additional message exchanges, adding a round-trip time to the latency and giving up the possibility of locality (using caching), which is unsuitable for some interactive applications.

For example, a map application could embed GPS coordinates in a URL that names the content associated with the map coordinates. In such cases, the data is probably stored in a sparse data structure on the server side and generated on-the-fly based on requests. However, the application is still data-centric in nature and it is possible to cache the results in the network based on the hash of the identifier. The network does not need to interpret the identifiers and the end-to-end principle is adhered to, but the full variable-length identifier must still be contained in the subscription and rendezvous messages so that the data source and scope home can construct the data based on the embedded information. Note that it is not enough to just store the hash of the identifier, as it is not possible to decode the semantic information back from the hash on the publisher side. In payload messages, the RIds can simply be replaced with short hashes.

It should be possible to have multiple naming schemas for application level identifiers (AIds), which are then mapped to RIds/SIds by some external means based on different application requirements. Some of these identifier types can have variable-length names for content. Therefore, we could specify an extended RId label that can be optionally included in the payload of rendezvous and subscription messages and is used to identify the data together with the RId contained in the message header.

3.4.2 Update on Algorithmic Identifiers

3.4.2.1 Identifiers

The PSIRP architecture is information agnostic – information may have explicit or implicit relationships, defined on any number of levels including application, end user, transport and ontologies [PSI09]. As such, it would be useful to be able to relate this information either explicitly or implicitly, to which there are two technical approaches - relationship tags and AlgID's.

3.4.2.1.1 Algorithmic Identifiers (AlgId's) and Relationship tags

The term *algorithmic identifier* (or AlgId) refers to identifiers or graphs of identifiers that can be created through automated algorithms. End hosts could use the same algorithms to generate other related identifiers to enable, e.g., information subscription.

Relationship tags are used to describe tags applied to the packet header (or payload) to form relationships between two seemingly unrelated identifiers, and may or may not be sufficient to produce other related identifiers. Given that relationship tags can be thought of as a subset of AlgIds (in terms of functionality), this document will focus only on the design and implementation of AlgIds.

Figure 3.19 illustrates the abstraction available through the use of identifiers, enabling the user to choose the most appropriate form.

Figure 3.20 gives an example instantiation of Figure 3.19, showing that either the algorithm alone, the place to find the algorithm, or an algorithm to find the place to find the algorithm can be transmitted.

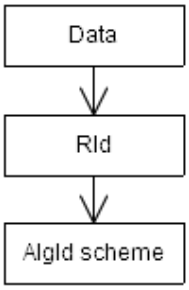

Figure 3.19: AlgId hierarchy

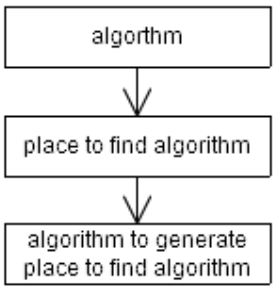

Figure 3.20: An example instantiation of the AlgId hierarchy with algorithms

Taking the general case as illustrated in Figure 3.19, algorithmic identifiers are generated using a function, which takes inputs (which could include a RId or other parameters) and outputs a Rid. It is therefore the use of an algorithm, which is the key differentiator between AlgIds and relationship tags.

3.4.2.1.2 Usage within PSIRP

As outlined in PSIRP D2.4, there is a wide range of potential network uses for algorithmic identifiers.

Subscription Management

In network architectures such as PSIRP, the subscriber selects and subscribes to individual identifiers. As these identifiers should ideally be uniformly distributed throughout the identifier space to avoid routing hotspots, PSIRP provides scopes (denoted as SIds, see [PSI09]) within which the individual information identifiers (RIds) are published. We can envision that there are several alternative semantics for publishing and subscribing to identifiers structured in a hierarchy (it is important to note that these semantics do not reflect particular semantics at certain interfaces, such as on the service level, but publishing actions throughout various levels of our architecture). Publishing to a (non-leaf) identifier within the hierarchy can result in three actions:

(1) The information is sent over the network on the specified identifier. No function is used to generate additional AlgIds for the publisher.

(2) The information is sent over all identifiers that are reachable in the hierarchy from the specified identifier (including the specified identifier). A function can be used to derive the subordinate AlgIds.

(3) The information is sent over all identifiers that are (common) antecedents in the tree (including the specified identifier(s)). A function can be used to derive the antecedent

Similarly, subscription to an identifier can result in:

(1) Subscription to information carried over the network on that identifier.

(2) Subscription to information on (that identifier or) any descendent identifiers in the hierarchy.

(3) Subscription to (information on the identifier(s) or) any (common) antecedent identifiers in the hierarchy.

If all options are available, an application of algorithmic identifiers must take care to match the publication and subscription semantics. For example, choosing the second option for both the publisher and the subscriber would result in the information being delivered multiple times over different identifiers.

Forwarding State Aggregation

Similar to subscription aggregation, algorithmic identifiers may also perform a role in the forwarding function. In such networks, links, waypoints or intermediate networks may be given identifiers that are used to control the forwarding of information. This concept is less useful in overlay identifier-routed networks where traffic is forwarded via the rendezvous point, but is applicable to networks with a separate forwarding path specification. Since the PSIRP network architecture separates a (fast) forwarding path from the (slow) rendezvous path, such techniques are applicable here. PSIRP specifies forwarding links using *forwarding identifiers* (FIds, see [PSI09]). Thus, we can consider that these can be either algorithmically generated or aggregated into longer path identifiers. For security reasons, the FIds within the PSIRP forwarding network are volatile as subscribers remove their interest in information. Thus, functions can also be provided to determine how such FIds are cycled. Trusted publishers, or topology formation components, may use secret parameters to AlgId functions in order to be able to determine how the forwarding identifiers change over time.

Caching

To perform effective caching, we must identify useful chunks of information. For example, it is probably of little use to collect a few frames of video without the associated meta-information, or at least it may be more useful to retain complete frames than frame deltas in the case a cache needs to perform selective dropping. The cache therefore needs to be able to identify a complete useful set of information to be cached. This could be achieved if such a set were identified by an identifier and the cache was aware of how many related identifiers were children of such a set identifier. For example, an instruction could be sent along with an item of information that the identifiers of related “siblings” in the useful set are generated using a sequence number *1...N*, a function *f*, and a set identifier *S*.

Coding

It is possible that the same information can be sent over the network using different encodings. Such encodings may be lossless, preserving the original information, or may transform the information (e.g., compaction to different video bitrates). Such encodings may be identified automatically by generating AlgIds. An application that wishes to adapt its bitrate, would therefore be able to automatically generate the AlgId of the required encoding and subscribe to this new information feed. Alternatively, mobility to a different device with a different screen size or audio capabilities might also result in subscription to a different encoding. This approach would also work for layered video, where each layer would be identified with an AlgId produced from the original information identifier.

Return Path

Many applications may wish to operate in a client-server mode. One communication pattern for implementing such a client-server relationship over a natively publish-subscribe paradigm (such as provided by PSIRP) is for the server to subscribe to an identifier to receive requests for information. To return the response, the client also needs to subscribe to an identifier. The identifier used for this return path may be automatically generated as an AlgId. Thus, a

request-reply transport layer operating over a publish-subscribe network might automatically subscribe to the reply AlgId before sending the request on the original identifier.

Flow Control

AlgIds could also be used to perform flow control. A simple example is that the information could be sent at different rates using different AlgIds. Alternatively, an AlgId derived from the content delivery identifier could be used for signalling information to control the sender rate (like TCP). Any application wishing to adapt the rate for a particular identifier would send requests to the AlgId automatically.

Content Fragmentation

Fragments of content (such as BitTorrent pieces) may be sent by using AlgIds. Any application wishing to receive a complete item of information can generate and subscribe to the identifiers for each fragment, instead of requiring that these are explicitly listed in content meta-data. Such fragmentation can also be structured semantically – e.g., voice, video, biography, trailer etc.

Sequence Numbering

Any application wishing to produce a sequence of information items may use AlgIds for each item in the sequence produced from a sequence identifier. For example, the temperature reading from a sensor may be identified by a single identifier. Each separate reading is then allocated an automatically generated AlgId. Any application wishing to follow the sequence must adapt its subscription ready to receive the next item in the series. This allows previous items to be repeated without burdening applications that have already received them.

Error Control and Reliability

Similarly to flow control, an AlgId can be automatically generated for any application that wishes to receive network delivery errors associated with another information identifier. Other AlgIds may then be used for the retransmission of information. Using an AlgId for error correction allows a sending application to retransmit information without burdening multicast listeners who received the information correctly. Separate AlgIds may also be generated to transmit logs of the information that is being sent over other identifiers so that applications can detect missing deliveries.

Announcements

Prior to sending information, announcements may be sent over corresponding AlgIds. These announcement channels may carry announcements for a variety of other identifiers. Thus, an application can subscribe to a few announcement AlgIds, covering its broad information interests. When an announcement is received, they can then join the correct rendezvous identifier to receive the information, reducing the average subscription state in the network and allowing receivers to pick and choose which information they receive over a rendezvous identifier.

3.4.2.2 Design Choices

In this section, we examine some of the questions along with the resulting options available for designing an AlgId scheme:

- Which AlgId scheme could the publisher use?
  - Which class of function should be used?
- Where will any AlgId relationship testing take place?
  - Will there be enough information provided in the packet for this, or will it be stored externally?
- How and when will this AlgId scheme be communicated to the subscriber?

3.4.2.2.1 Choice of class of AlgId function

When designing an AlgId scheme, there are four classes of functions available for the publisher to choose between, based upon the requirements of the scheme for the information being published.

Identifier generated from a single parent

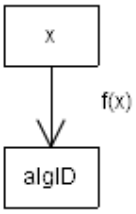

Figure 3.21: Identifier generated from a single parent (one-way function)

The first class includes one-way functions (e.g., hash functions) that generate *identifiers* from a single parent. While this limits the generation of identifiers to descendents only, in some situations this would be seen as a benefit as it would restrict the linking of parents to children only to those who knew both the parent and child identifiers in addition to the algorithm and any secret keys used.

Identifier generated from a single parent using a reversible function

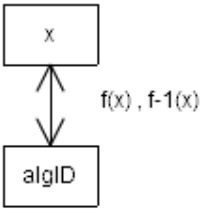

Figure 3.22: Identifier generated from a single parent (reversible function)

The second class of functions takes a single parent and uses a reversible function, for example a block cipher (when input parameters are also known). This enables the generation (and insertion) of both descendents and antecedents, offering the most flexibility to the user of the scheme.

Identifier generated from multiple parents using a one-way function

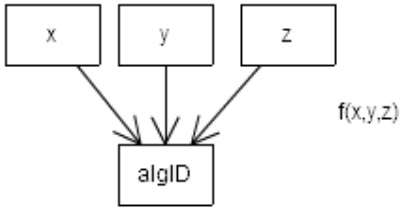

Figure 3.23: Identifier generated from multiple parents

This third class of function takes multiple parents to generate a child identifier using a one-way function, for example a bloom filter. A result of applying an algorithm from this class is that information is lost. As a result, it is impossible to generate the parent identifier using the child identifier. Whilst the child may be able to surmise a potential set of parents and an even larger set of antecedents, this will make this class of function unsuitable for some scenarios where accurately generating parent identifier(s) is a required attribute. Additionally, once an identifier has been created, no new antecedents may be added to the graph.

Identifier generate from multiple parents (reversible)

The fourth class of function takes multiple parents to generate a child, however by using a reversible function and storing function data along with every identifier, the parent identifiers can always be generated given that of the child. An example usage of this could be for news feeds where a topic can belong under multiple categories.

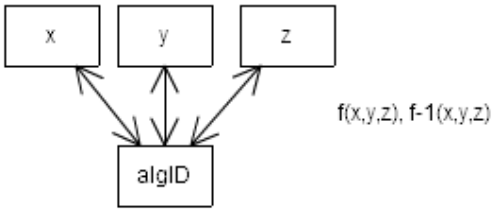

Figure 3.24: Identifier generated from multiple parents using a reversible function

As each class of functions has different attributes which may or may not be applicable for certain uses, the choice of class of function is one best left to the publisher.

3.4.2.2.2 Relationships

Given that identifiers are usually expressed using a generalised graph model, there are a handful of mathematical relationships, which we may want to be able to test:

- **Reachability**: Are two information items part of a larger data item?
- **Parent/Child**: Direct reachability (non-transitive).
- **Ordering**: Which is the antecedent and which the descendent (particularly useful for fragmentation and sequencing)?
- **Range test**: Given three identifiers does one lie in the middle of the other two (useful for subscription ranges)?
- **Common antecedent/parent**: Given two identifiers, are they related in a larger graph? This would not necessarily require knowledge of the antecedent.
- **Edge ordering**: If multiple children are produced from a node either using the same or different functions, can we apply ordering precedent to the functions and thus produce a complete ordering?
- **Prevent linkability**: Given two identifiers, we may want to prevent testing of the relationship without a third (or more) node(s). For example, to test for a common antecedent, the user would have to have knowledge of the antecedent itself.

Relationship Testing and Communication

Given the relationships above, there are several associated roles, which we may wish to be performed within the network; relationship testing, relationship generation, AlgId source and AlgId client.

A relationship testing role would be used to evaluate relationship tests and return a true or false answer, unlike relationship generation, whereby the service would be expected to return a RId or other information. The AlgId source and AlgId client will be explored in more depth in the next section.

Information structure (Relationship) communication

Depending on the role performed, the question of how to transmit the publication information structure remains: should information be transmitted along with every RId, should the

information be available as a bulk download under a separate RId, or should it only be available through a relationship testing or generation service?

If the information structure was transmitted in every frame (making each RId descriptive of its location within the overall structure), fields, which the publisher may wish to include, depending on the algorithm, include:

- A Root flag
- A Leaf flag (i.e., do not continue to generate descendent identifiers)
- Branching order
- Total number of branches off parent (child count)
- Depth (could be inferred by generating path back to the root identifier)

If the information structure was available as a bulk download, this could be available under a separate RId, which may be transmitted in the first frame, in every frame or in a selection of frames.

Lastly, a service could be provided to enable others to submit relationship test or generation requests, abstracting the complexity of this task to a potentially more capable node.

Relationship testing and generation location

Having explored how the information structure can be transmitted, there are three places in which the testing and generation could take place:

- **The local node** wishing to find out the relationship (usually the subscriber, but may be an intermediate node on the forwarding path, such as an opportunistic cache)
- **The publisher**
- **A third party**, such as a specific relationship testing service

Option 1 – the local node

As the local node will usually (but not always) be the subscriber, it should either already have the algorithm to generate related identifiers, or possess an RId from which the algorithm could be retrieved. In the latter case, this information may have been included in the first packet or a subsequent packet depending on the used communication scheme (discussed in the next section).

Given that the relationship data will be available at the local node, any relationship testing or relationship generation would be performed locally, preventing the node from making frivolous tests as this would only impact the local node's resources.

Option 2 – the publisher

Whilst being the most obvious choice, given that the publisher would have full knowledge of any relationships between identifiers (including any secrets required by the algorithm), this could result in misuse depending on the type of service offered.

The most computationally intensive service would be to offer *relationship* generation, in which users could, for example, send requests over a separate RId with a request similar to **getParent(childRId)**. This would therefore be a prime target for a DoS attack.

To somewhat mitigate this effect, a *relationship testing* service could be offered, in which the requests would have to provide the expected answer and the service would only have to return true or false, for example **testIsParent(expected_parentRId, childRId)**.

If the publisher provided a relationship testing service over a special RId over which test requests could be sent using a specific payload format.

Option 3 – A third party

Rather than requiring the publisher to handle relationship test requests, this could be "outsourced" to a helper function residing within the network, and as such would allow multiple different services to operate, potentially with different restrictions on use (public/private/paid for). This requires that the third party has a full copy of the publication information structure, or has some knowledge of the identifiers used, the function itself and any secrets required for its use. While offering the advantage that the publisher would not have to expose another potential means of attack, it would result in having to share this information with a third party.

There are two types of helper function, which could be present in the network:

- an **AlgId generating helper**, which could pass AlgIds back to the subscriber or perform the relationship tests.
- a **Value added helper**, which could, for example, generate identifiers (and test relationships as required) and then perform some action. For subscription management, this could be hiding the complexity of the data being addressed via separate RIds by performing the RId resolution and subscription locally so that the end subscriber only sees a stream of data.

Suitability

Given the options, the choice of location and information structure communication for relationship testing depends largely on the data being transmitted and thus on which and how often the relationships need to be tested.

Some of the factors influencing this decision are as follows:

- Frequency and type of relationship look-ups
- Free space within packet headers/payload
- Spare network capacity
- Sensitivity of information relationships
- Complexity of information relationships

If on one hand the subscriber or an intermediate party will only rarely require relationship testing, then, rather than including this information in every packet, it would be wise to require the subscriber to request this information separately from a third party for every relationship they wish to test, as illustrated in Figure 3.25.

1) Helper publishes intention to receive requests for relationship information
1) Subscriber subscribes to receive data relating to its relationship request
2) Helper replies with relationship information

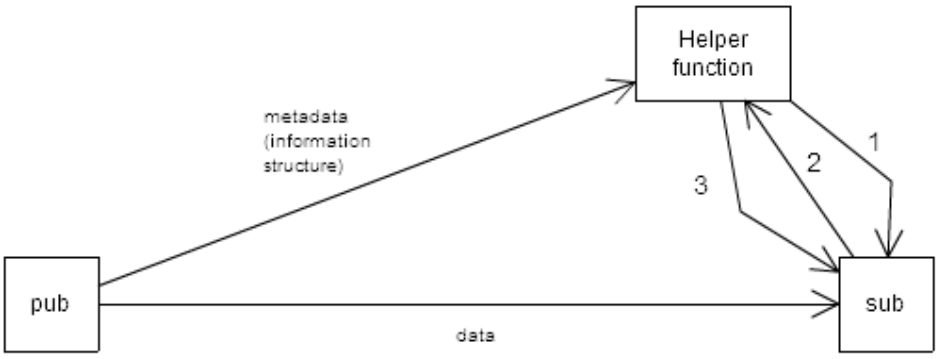

Figure 3.25: Third party relationship testing using a helper function

Conversely, if the subscriber requires frequent relationship testing, then storing this data as close to the subscriber is beneficial, either by transmitting it in every packet (Figure 3.26), or providing the information available as a bulk download, via its own RId (Figure 3.27).

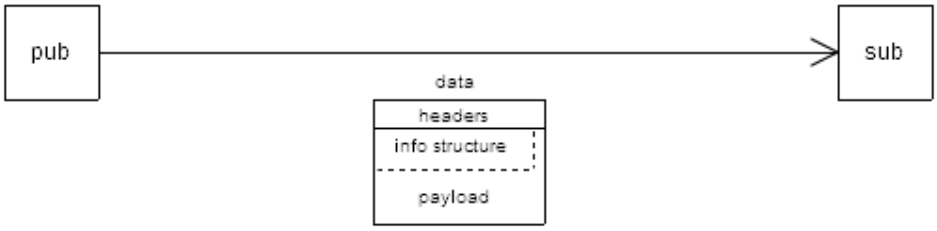

Figure 3.26: Enabling local node relationship testing by adding in-frame relationship information

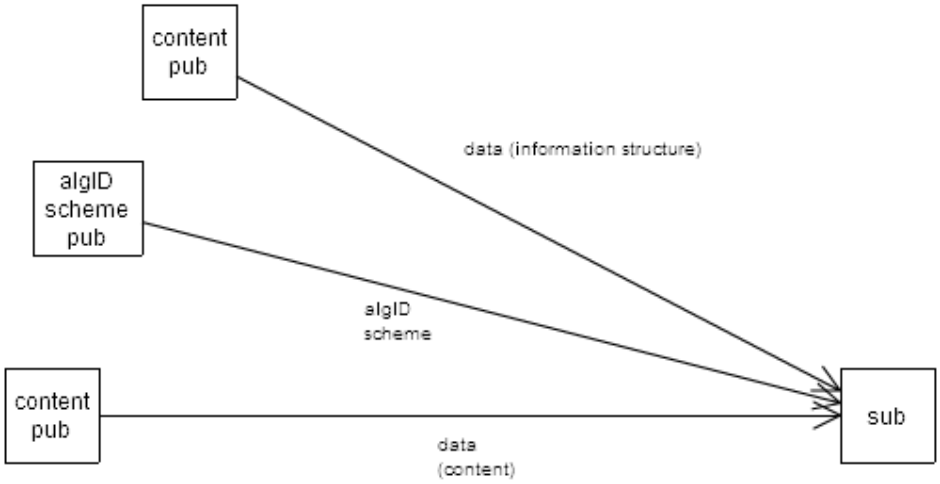

Figure 3.27: Bulk relationship structure information download

Figure 3.26 also illustrates that the content publisher does not have to be the AlgId publisher, as the subscriber first subscribes to RIdA, which includes a link to the location of the algorithm to find the relationship data, RId B. Upon subscribing to RId B, the AlgId scheme is transferred in the payload along with the parameters to generate the RId, RId C of the location for the relationship data. Lastly, the subscriber subscribes to RId C to receive the relationship information.

3.4.2.2.3 Schemas

When using algorithmic identifiers, it is likely that no scheme (or class of generating function) will perfectly suit every situation. Therefore, enabling different schema would allow the user to tailor the algorithm to suit the particular task in hand. This flexibility could be achieved through the implementation of a generic AlgId framework, which defines common methods and properties.

This framework could take any number of forms, including one of the following;

- An X bit header field referring to standardised AlgId schemes, published at a centrally trusted repository, similar to how XML and HTML schema definitions are in the current Internet. Within this scheme, setting the field to all zeros could indicate no schema, while all ones indicate a custom schema being described in the payload.

- An AlgId schema reference RId which would allow any custom scheme from any location. This would present space issues (which could be solved in a number of ways) and numerous security issues.
    - Enable upload of custom schemas to one central repository, with the scheme reference being assigned a unique identifier. An example of this, using the current Internet addressing scheme for clarity, would be http://schemaDB.org/custom/ref=#, where # is the XX bit unique identifier
    - The same as above, but using a short bit pattern prefix to enable the use of several different repositories. This could enable the use of a scope local repository.
    - Require that the schema URL to be no longer than XX bits.
    - Use a time-limited unique identifier from a pool and store the schema along the proposed route(s).

Schema Communication

Having decided upon a particular scheme, the question remains of how (and when) to communicate this scheme to the subscriber. The usual conversation between publisher and subscriber will result in the subscriber requesting information from a publisher given a RId. Assuming the RId is actually generated using an algorithm, and the subscriber is interested in this related information, there are a few possible ways to communicate the algorithm used:

- In the first frame
    - In the header in a standardised field (such as a reference to a schema repository
    - In the payload with no other content (content begins from first algorithmically generated RId)
    - In the payload, at the start or end of the actual content to be communicated
- In every frame
    - In the header in a standardised field (such as a reference to a schema repository
    - In the payload
- In a separate frame under a different RId or via a helper function

Figure 3.28, 3.29, and 3.30 illustrate how the schema could be stored under a different RId, which could either be the location of a helper function or the schema alone.

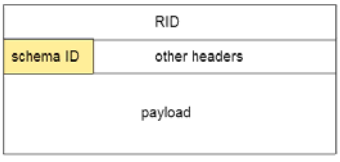

Figure 3.28: Example header structure using a schema Id

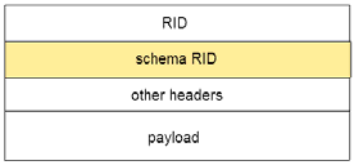

Figure 3.29: Example header structure using a schema RId

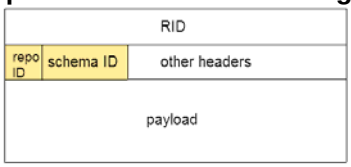

Figure 3.30: Example header structure using a repository Id and schema Id

3.4.2.2.4 Packet Addressing

When choosing which AlgId scheme to use and how to implement it, the granularity of RID addressing is also important.

Every frame has its own RId

When information is transmitted such that every frame (and frame fragment) has its own RId, the algorithm information must be transmitted within every fragment as it cannot be assumed that the subscriber has received any previous frames related to the currently subscribed RId in which the AlgId scheme is contained.

Multiple frames share a single RId

When information is transmitted over an RId in which every frame may not have its own RId, the issue of when and where to transmit the AlgId and relationship information becomes more complicated. This may occur when the application has no desire to be able to individually identify information frames – for example if an item of content is broken down for transport or if previous information frames become obsolete.

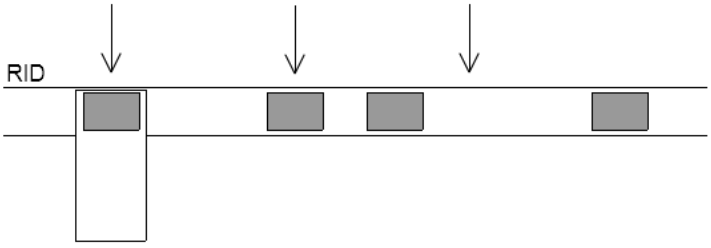

Figure 3.31: Multiple frames sharing the same RId

If the AlgId scheme was transmitted only in the first frame sent over the RId, subscribers joining the stream half way through publication would not receive this information. This leaves us with an open question of **WHEN** to transmit this information:

- Transmit the AlgId scheme in every frame
- Transmit the AlgId scheme in the first frame
- Transmit the AlgId scheme in selected frames (analogous to video files having key frames and deltas)

Another choice is whether to transmit this information combined with or independent from the content to be delivered.

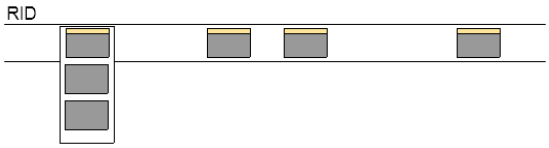

Figure 3.32: AlgId scheme combined with the content to be delivered

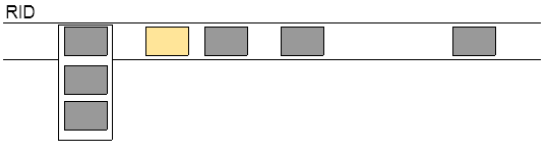

Figure 3.33: AlgId scheme separate from the content to be delivered

Suitability

Comparing the possible options, the choice primarily comes down to the space taken up for transmitting the AlgId scheme.

If the scheme were to be transmitted in full, then choosing to deliver it in every frame would result in little space left for the actual content. Transmitting it in only the first frame sent over the RId could result in subscribers who join the stream half way through missing the scheme information, meaning that the subscriber would have to find another method to receive this information (for example waiting for the content to transmitted over the RId to loop back round, if transmitted in this way by the publisher). To mitigate the effect of joining the stream half way through, the publisher could transmit the schema periodically to ensure that those joining half way through do not have to wait too long for this information. Unfortunately, finding the balance between the time period to wait for schema re-transmission is likely to have to be tweaked on a per publication basis.

If on the other hand the scheme was just an RId upon which the information was published, then all three options to transmit this information would be feasible, again based upon the amount of free space that would be left in each frame.

Finally, there appears to be no clear advantage to delivering the scheme combined with the content or in separate frames, and as such this decision should be left up to the publisher. One permutation of transmitting the scheme combined with the content which would not be advisable would be to cut the full scheme into chunks and transmit this content over a number of frames – you would have to ensure the user received every fragment as otherwise the scheme would be incomplete.

3.4.2.3 Use Case: Subscription Management

In this section, we take the example of subscription management to elaborate upon the requirements and conclude with a potential implementation for an AlgId scheme.

3.4.2.3.1 Choice of content delivery model

Definition of terms:

Publish: An operation at the network level which involves one party (namely a content publisher) to transmit (publish) data over a predefined RId to one or more subscribers

Subscribe: An operation at the network level whereby a user subscribes to receive content published by a content publisher over a pre-defined RId.

Content publisher: An entity which holds content and can make it available for others to request.

3.4.2.3.2 Basic PSIRP approach

The next diagram illustrates the PSIRP model for matching publishers and subscribers.

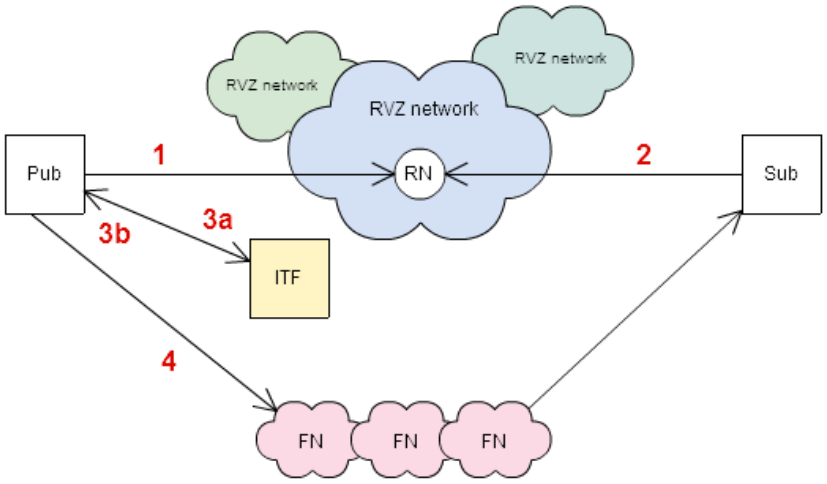

Figure 3.34: Basic PSIRP delivery model

Step 1: The publisher, P1, fetches subscription information and registers to receive updates over a known RId belonging to the RVZ node

Step 2: The subscriber, S1, subscribes to a chosen RId

Step 3a: The publisher requests a forwarding path to S1 from the ITF.

Step 3b: The ITF returns the forwarding path to P1.

Step 4: The content publisher publishes the content C1 to S1

With the basic PSIRP model, only one RId is exposed to the publisher and subscriber, although others are used internally, for example between the publisher P1 and the ITF.

3.4.2.3.3 Document delivery over the basic PSIRP model

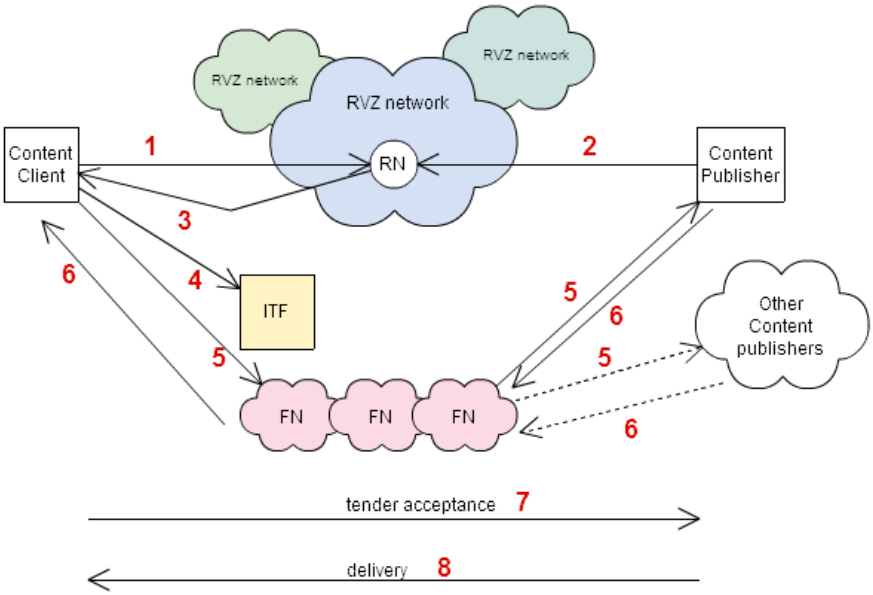

Figure 3.35: Document delivery over the basic PSIRP model

Step 1: The content client requests content C1 using RIda to the rendezvous system

Step 2: The content publisher makes available content C1 and advertises this to the rendezvous system (this availability can also be signalled before the content client requests the content in step 1).

Step 3: The rendezvous system returns a list of network providers, which contain content providers for content C1 to the content client.

Step 4a: The content client, CC1 requests forwarding paths to each of the network providers returned by the ITF system

Step 4b: The ITF system returns the forwarding paths to CC1

Step 5a: The content client sends out content provision requests over each of the forwarding paths to the various content providers of content C1, each containing a back channel id for the replies

Step 5b: The content client subscribes to each of the back channel Ids (usually simultaneously or before the content provision requests are sent).

Step 6: Using the back channel included in the content provision request, each content provider sends tenders with its own back channel Ids (to which it automatically subscribes).

Step 7: The content client accepts one or more of the tenders and responds to the content providers(s) including new back channel Ids (to which it automatically subscribes) for the content to be delivered over.

Step 8: The content provider(s) receive the tender acceptance for content C1 and publish the data as requested.

Within this model, multiple RIds are exposed and used, particularly during the tendering phase.

3.4.2.3.4 Document delivery over a specialised "Document Model"

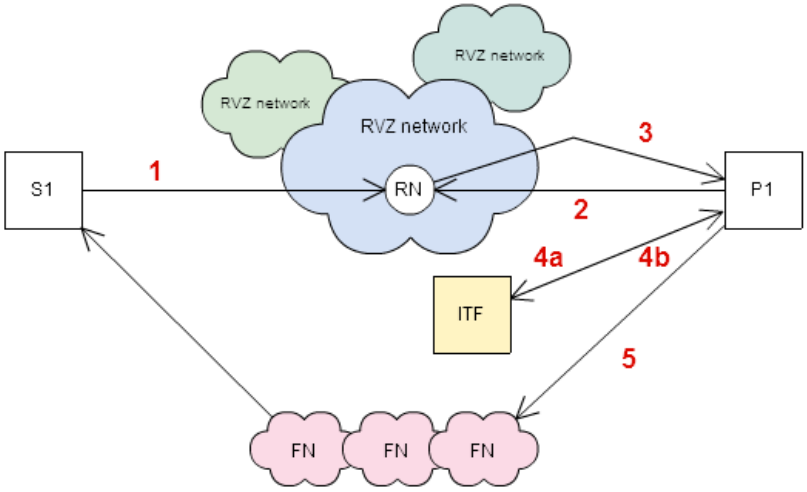

Figure 3.36: Document delivery over a specialised Document Model

Step 1: The subscriber S1 indicates its interest to request and sync content C1 to the rendezvous system.

Step 2: The content publisher, P1, makes content C1 available and informs the rendezvous system.

Step 3: The rendezvous system matches the requests locally and chooses a publisher P1 (from those it knows have made the content available) and registers S1's interest to receive that content with P1.

Step 4a: The content publisher, P1, having received the request from S1, requests a forwarding path from the ITF.

Step 4b: The ITF returns the forwarding path to P1.

Step 5: The content publisher, P1 publishes C1.

This model provides a one-to-one mapping between content identifiers and rendezvous identifiers, making this model tightly coupled. While this model uses more than one RId internally, only the one RId (based upon the content being delivered) is visible to the publisher and subscriber during communication in steps 1, 2 and at the final hop of the forwarding network (as it would be visible under various other forwarding identifiers until the last hop)

3.4.2.3.5 Comparison of document delivery models

In the channel model, there is a two step process, with the subscriber first requesting information from all matching publishers directly (matched via the rendezvous system) and then the subscriber choosing which publisher to receive the information from. This allows the subscriber to take into account various social, economic, political, security and network parameters, such as number of hops, jitter, bandwidth speed, personal preference for publishers, cost to retrieve content, geographic boundaries crossed, and secured/encrypted links.

By simply matching publish and subscribe requests, no technical or implementation restrictions are placed on the rendezvous system.

In the document model, the rendezvous system would be expected to perform the matching of subscribers to a suitable publisher (assuming there is more than one source). While this would potentially mean a reduced volume of traffic between pub and sub (pub m msgs ->Rvz->1 msg sub), it offloads the choice from the subscriber to the rendezvous node, which may or may not be desirable.

While this document will not mandate a particular model to use for subscription management, any AlgId scheme constructed should support both delivery models.

3.4.2.3.6 Requirements for an AlgId scheme

Within subscription management, there are a few key requirements, which the AlgId scheme must enable a subscriber to fulfil:

1. Calculate and subscribe to parent and children identifiers
2. Calculate how many information items comprise the collection/graph

The latter would be useful for subscribers to check they have received all of the information items in the graph.

Choice of class of function

While the choice of function should ultimately be left up to the publisher, given the requirements above, the function is most likely to belong to the "single parent reversible" category as the subscriber could independently generate parent identifiers. If however the subscriber has a means to start from or obtain the root Id then the function could feasibly be "single parent one-way". This may be because the subscriber starts by receiving the root frame, or because a prior frame has a link (e.g., RId) to the root. The use of a single parent reversible function would also provide benefits to other related use-cases, such as fragmentation or caching, as third party nodes with knowledge of the AlgId scheme could

generate the identifiers of related frames to ensure that the complete or a useful collection of information fragments was retrieved.

If delivering data using the basic PSIRP model, it is important to highlight that multiple AlgId schemes could be used during the delivery process, and in particular for generating the RIds for the request for tender channels.

In addition, the publisher has the choice between using a direct calculation function or a stepwise function for algorithm generation. In a stepwise function the subscriber would need to receive intermediate frames in order to generate the entire AlgId tree. If using a stepwise function, the subscriber would need to generate every intermediate ID in a path, for example between an Id at depth 10 and the root, which for larger trees (and non root identifiers) would become costly. Each frame would also have to indicate its position in the tree to enable the traversal.

If using a direct function, the subscriber would either need to generate the root Id and then the desired Id (based on its graph position), or calculate the Id directly (again based on its graph position).

Overall, while both stepwise and direct calculation functions are adequate, the latency introduced by a stepwise function having to work through a potentially large number of intermediate Ids makes the use of a direct calculation function preferred.

AlgId schema communication

Under the banner of subscription management, there are two extremes of content, which may be transmitted: short pieces of information addressed by only a few Ids (for example news tickers) and large bulk downloads addressed by many Ids. Regardless of the scheme used, having to include this in every frame would be a significant overhead in the latter case – by using some form of inheritance, which could be a one bit “use AlgId scheme referenced by parent”, this could be greatly reduced while providing only a minimal overhead increase in the former case.

Relationships and relationship testing

Requirements 1 and 2 above correlate to the relationships of parent/child and reachability, however the frequency of testing of each of these relationships will vary dramatically between publications. Therefore, it should be assumed that there could be a potentially large number of requests for testing both types of relationship, and thus several factors need to be considered:

Static / Dynamic information graph

The first factor the publisher needs to consider when choosing how to enable relationship testing is whether the information graph will be static (will not change over time) or dynamic (is likely to be extended/changed in the future). This is because in the latter case, the relationship information would have to be updated upon every change to the graph, and could result in the subscriber having stale knowledge of the information graph unless informed of every change.

Location of relationship testing

The second factor is the frequency and complexity of relationship testing required, and thus the location where this testing should take place. As information graph sizes will vary between publications (and thus frequency of relationship tests), any of the potential options outlined under the design choices section could be valid with one exception. When publishing dynamic information, it would be unwise to transmit relationship information in-packet to make the RIds self-describing of their relationships in the information graph, as it would require the subscriber to continuously check the validity of its internal information structure model (e.g., by maintaining all subscriptions to receive graph changes). Alternatives are to collect the relationship information together to reduce the amount of RIds that need continual subscription, or to signal changes in the graph from a reduced number of RIds (e.g., the root frame).

3.4.2.3.7 Implementation

In this section, we illustrate how an AlgId scheme could fit with both document delivery models and conclude with four example AlgId scheme implementations.

Subscription Initiation

As any algorithm used for subscription management should support both document delivery models, it is worth illustrating what are the differences in requirements, if any.

Taking Figure 3.37 as our example publication consisting of 6 fragments of content, each identified by a separate RId derived from an arbitrary AlgId scheme, it can be shown that after the tender request and accept phase, the document delivery is identical for both models.

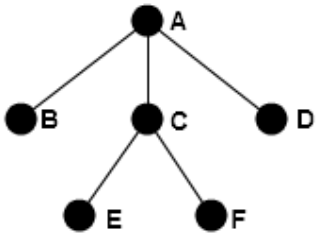

Figure 3.37: Example publication information structure

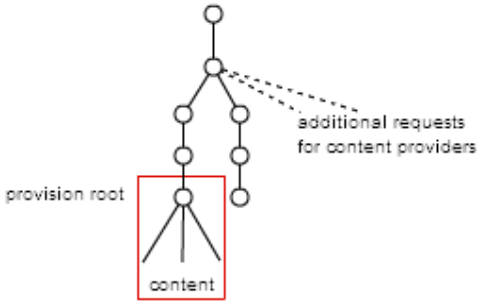

Figure 3.38: Document delivery tree using the channel model

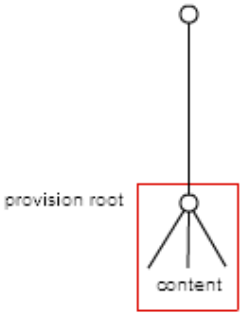

Figure 3.39: Document delivery tree using the document model

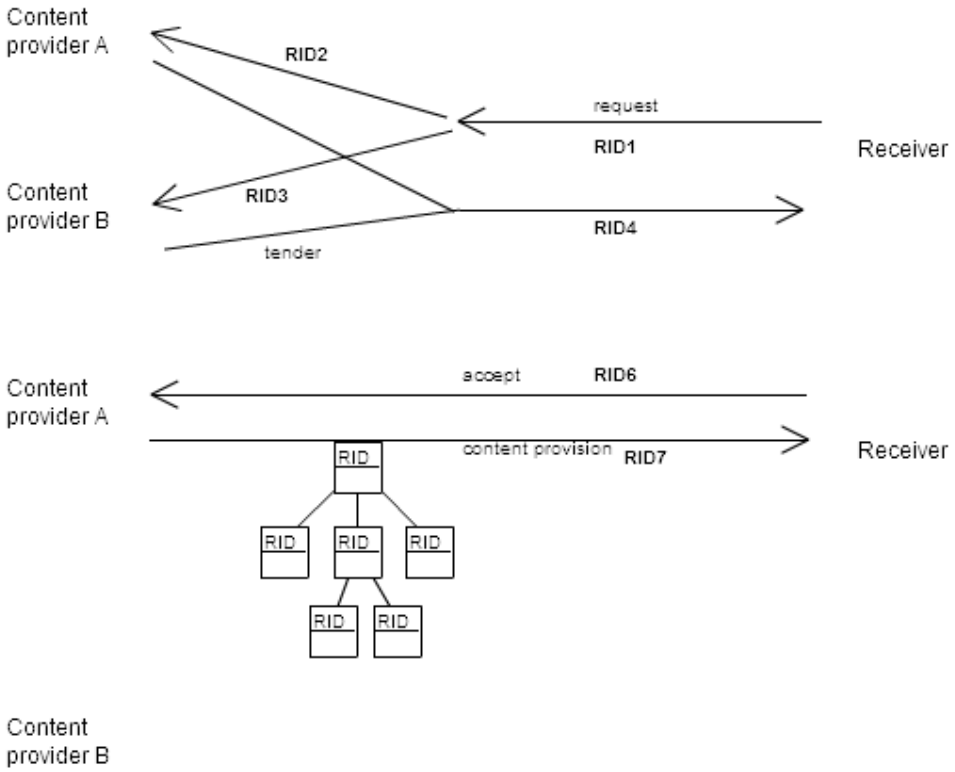

Figure 3.40: Content provision using the channel model

Following the steps required for subscribing to a document using the channel model (Figures 3.38 and 3.40), the initial request for content is sent from the receiver to the rendezvous node over RId1, which is in turn forwarded as a tender request to all matching content publishers over RId2 and RId3 respectively. Within this request, RId4 is included as the back channel for all tender responses to be returned to the subscriber. Each content publisher responding to the tender request includes their own RId, RId 5 and 6 respectively for the subscriber to communicate their acceptance over. The subscriber, having received replies from both content providers chooses the first provider and sends an accept message over RId5, including the RId (RId7) of the channel to fulfil the content provision. The content provision then begins initially over RId7, and other RIds as described within the AlgId scheme communicated by the publisher.

How to derive the tender/accept RIds

If the publishers were to use an AlgId scheme to generate the tender and accept channels (and back channels), the algorithm would have to be transmitted along in the initial content provision request. Upon the content publishers receiving this message via the rendezvous node, the reply channel RId could be generated and the response could be sent to the subscriber. For the content provider, this would mean including in the reply the inputs to the algorithm used to generate the RId for their reply channel. Unfortunately, as every content provider (of which there could be a large number) must choose a different RId from every other content provider without being able to communicate with them, the range of valid inputs to the algorithm would have to be very large to reduce this possibility. As the number of bits required transmitting these values would likely approach the size of the RId which they reference, the benefit to using an algorithm for this stage of the subscription process fades. Finally, upon the subscriber receiving all of the content provision requests, it would too have to choose an input from a suitably large range for the content publisher to provide the content over, as this RId should be kept secret from the other unsuccessful content providers for security reasons.

In summary, using an algorithm to generate tender and accept channels would result in the overhead of:

1. Transmitting the algorithm to every content provider
2. Every content provider replying with an input parameter approaching the size of the RId itself
3. The subscriber replying over the above channel with an accept message and the input parameters (which again approaches the size of an RId) for the RId to provision the content over

As such, given the space and computational complexity of using an algorithm and AlgIds, it would be sensible to include the full RIds for each channel, and allow both the content provider and the subscriber to choose what these RIds should be themselves.

Algorithm

As the extra RIds required by the channel model for tender and accept requests will be chosen without the use of an algorithm, the requirements for the algorithm for deriving the AlgId sused to deliver the content are identical. Regarding relationship testing and generation, this will be performed at the local node using the in-frame information structure data. The AlgId scheme will be linked to via a RId, and where children use the same scheme as their parents, the one bit "user parent AlgId scheme" bit will be set. By using this one bit flag, the same usually required in the header for the AlgId scheme RId can be removed, increasing the available space for the payload. Additionally, there will be a parameter length field to indicate to the receiver how many bits following this field are used for AlgId input parameters, to enable a flexible and more efficient use of the frame (and again maximising the space for the payload). This frame format is illustrated in figures 3.41 and 3.42.

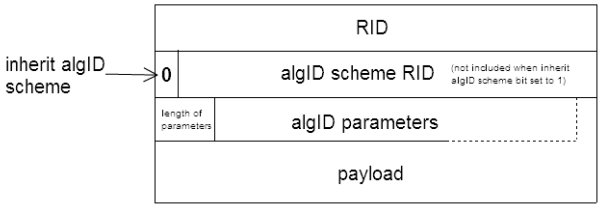

Figure 3.41: Frame structure when AlgId scheme is not inherited from parents

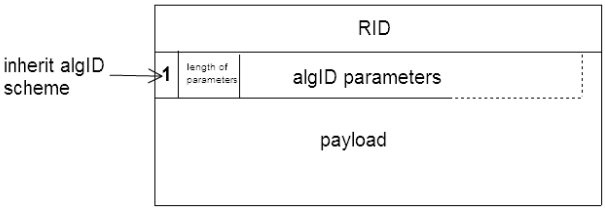

Figure 3.42: Frame structure when AlgId scheme is inherited from parents

As the choice of the algorithm will ultimately lie with the publisher depending on the content to be delivered, below are four examples of how different types of algorithm could be implemented.

Stepwise

Using the AES 192 algorithm as an example, this could be used as a stepwise function by requiring that the previously generated RId in the path be used as an input:

f(RId$_{n-1}$, key, child count, sibling number) = RId$_n$

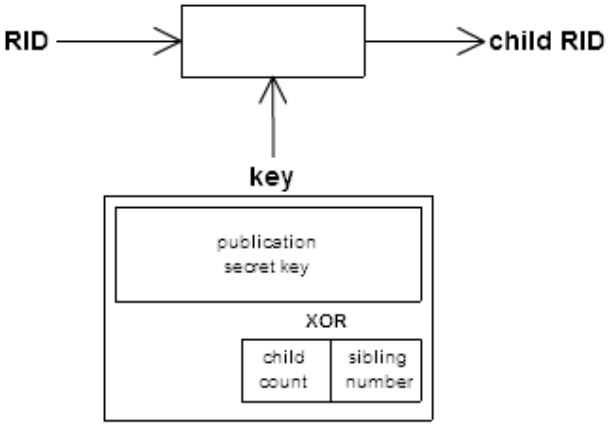

Figure 3.43: Graphical representation of the AlgId generation process for a stepwise function

To enable the subscriber to generate parent RIds, each frame would have to store the child count and sibling number of each node.

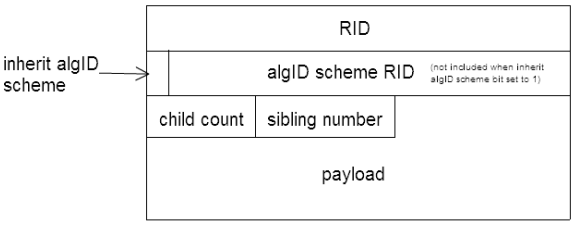

Figure 3.44: Example frame structure for a stepwise function

Direct

Given that the primary difference between a direct and stepwise function is whether the root or the previous ID used as an input, the AES192 algorithm can be used again. While it was sufficient for the stepwise function to store child count and sibling number, using the algorithm as a direct function places the requirement that the exact nodes' location must be stored in each frame.

Under the heading of direct functions, there are two types of implementation – a single step or a recursive implementation. In both cases, it is assumed that the subscriber stores the (root) ID from the initial frame delivery.

Exploring the recursive implementation first, taking a simple tree fragment as illustrated in Figure 3.45, the frame describing RID A will have to include a description of the entire tree. Depending on the type of tree, it would have to store varying amounts of information at every node:

1. Complete and balanced – branching factor and depth.
2. Balanced – branching factor, depth and pruning or growing information depending on how incomplete the tree was.
3. None of the above – the entire tree description, for example 3.3_0.0, with n different delimiters to differentiate the n different levels.

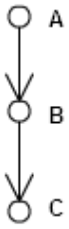

Figure 3.45: Tree fragment to illustrate recursive direct function implementation

If the quantity of information to be stored at each node became too large or difficult to describe, this would be stored to an external location and referenced within the frame.

At the root node, A, the frame structure may look like Figure 3.46, likewise with B (Figure 3.47) and C (Figure 3.48).

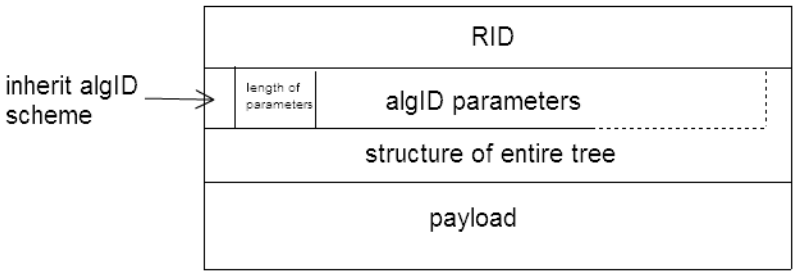

Figure 3.46: Frame structure at node A in Figure 3.45

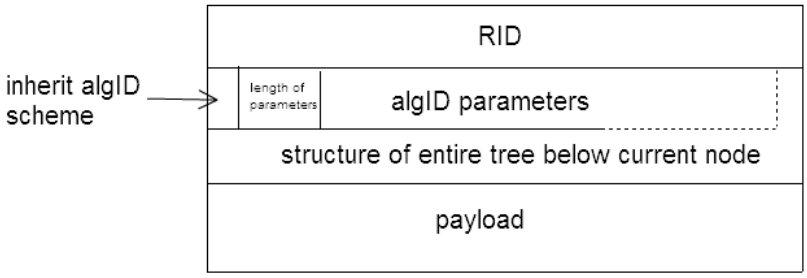

Figure 3.47: Frame structure at node B in Figure 3.45

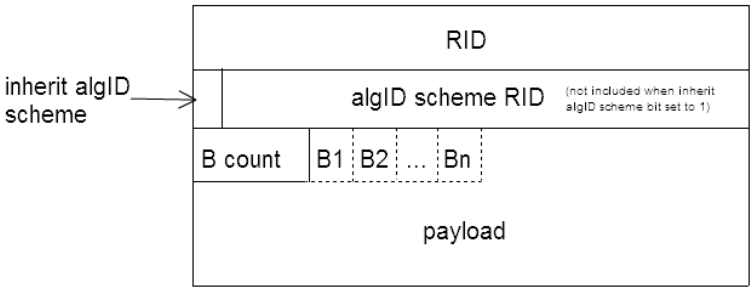

Figure 3.48: Frame structure at node C in Figure 3.45

In order to generate a child RId, starting at the root node the AES192 algorithm would have to be applied twice; once to generate the intermediate RIdB and then again to generate and subscribe to RIdC. It's important to note that the subscriber only has to generate the RId of the intermediate node, not retrieve its contents.

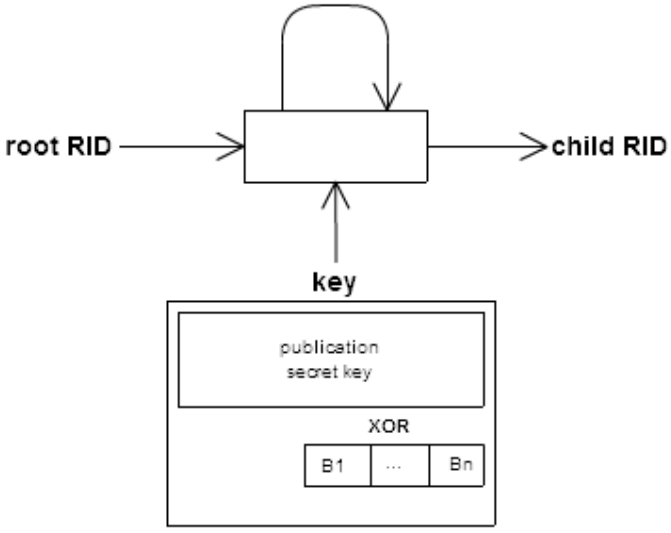

Figure 3.49: Graphical representation of the AlgId generation process for a recursive direct function

Looking at the single step implementation, as every node can be reached in one step from the root, this means the tree can be represented as a single level tree (excluding the root) and thus the only relationships to physically exist are parent and child between the root and every other node. This means that all structure information is only meta information stored at every node.

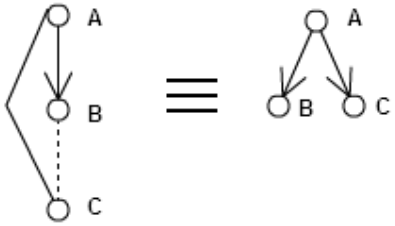

Figure 3.50: Tree fragment to illustrate single step direct function implementation

At the root node, the frame structure could look like Figure 3.51, containing the structure of the entire tree in a similar manner to that of the recursive direct function implementation, assuming the tree is not simplified to a one level tree before publishing, and nodes B and C could look like Figure 3.52.

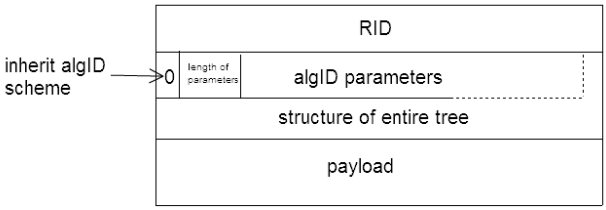

Figure 3.51: Frame structure at node A in Figure 3.50

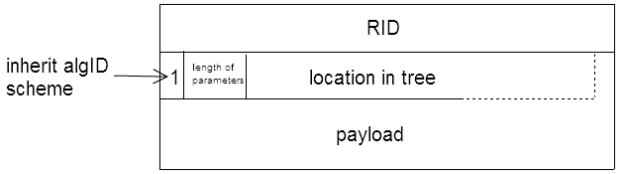

Figure 3.52: Frame structure at node B and C in Figure 3.50

One-way

Another example, this time using a one-way function would be to use the Tiger hash function (with a digest size of 192) and store the RId of the root node in the header. The inclusion of the root RId, along with the number of children and sibling number at each node in the graph would allow the user to generate the entire tree.

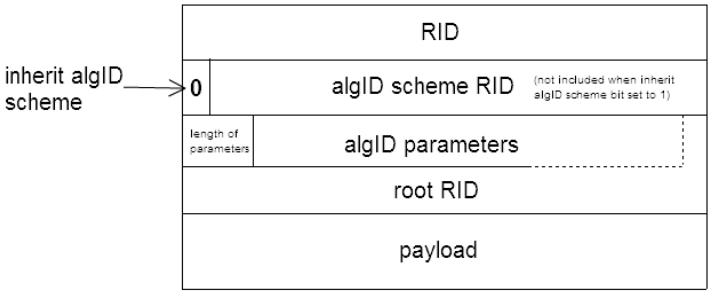

Figure 3.53: Example frame structure for a one-way function

The main disadvantage of this scheme is that in order to generate the parent ID of a given node, the subscriber would have to generate ALL Ids from the root downwards until the child node was reached again, which for large trees would result in a significant number of operations. This inflexibility could be mitigated by including extra position information in each frame, decreasing however the payload size and ultimate efficiency of the transmission scheme.

External

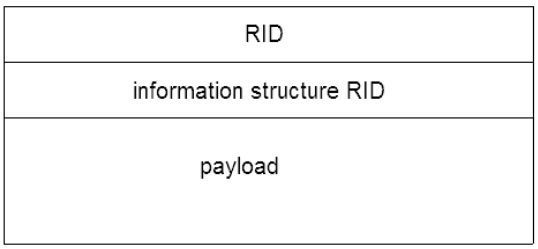

Figure 3.54: Example frame structure for externally referenced structure information

The final example is that of having the entire publication information structure published to an external RId, removing the need for the publisher to use any algorithm to generate the RIDs. By including this RId in every frame, the subscribers can download this information themselves to get the related RIds to which they may need to subscribe.

Conclusion

By examining the four potential implementations above, it is apparent that once location information needs to be stored at every frame, there is little difference between the stepwise, direct and one-way function. As the external implementation removes the need for location storage all together, it also removes the need for the use of an algorithm, turning the solution into a relationship tag example, which would offer significant in-frame space savings.

As a result of the example implementations above, no concrete recommendation can be made regarding which specific implementation to use, except that the choice of implementation should be left as a decision to be made by the publisher given the type of content to be published.

3.5 Caching

In the Internet, a large amount of content is transferred repeatedly. Most of the time, the content is retransmitted from the source to serve different requests coming across the network. The efforts to reduce the amount of repeated traffic in the current Internet can be roughly divided into two categories: application level caches, including web caches and Content Delivery Networks (CDNs), and application-independent caches, which can be often found in the so-called WAN optimization products. Today, these approaches together offer significant improvements to network performance by caching some of the content, but they all represent extra services on top of the actual network (overlays). In the PSIRP architecture, caching is considered one of the obvious pieces of functionality that a service provider or the network itself is well equipped to offer.

At the lowest level of the PSIRP architecture, every piece of content is considered to be addressable. This means that even when information is fragmented to multiple chunks, each such chunk may possibly be identified with their own rendezvous identifiers (RId). Concepts like scoping and metadata can be used to determine how the chunks fits into larger data abstractions, such as documents or streams. In any case, addressing each chunk with a separate RId allows for considering each one of them as being cache-able and request-able.

Retrieving information chunks from a cache requires subscribers to know the their RIds beforehand. In the present design choice, before the actual transfer of any data begins, the subscriber obtains a list of RIds that make up the object it desires – alternatively, such list of RIds may be computed algorithmically with approaches outlined in Section 3.4.2. It then requests each chunk (logically) separately; in practice, any node may optimistically push chunks whenever there is free capacity over its outgoing links. For this, the source constructs a meta-data object, containing the RIds of the individual chunks, and publishes it. If there is an update to an object, the source will update the meta-data object, republish it, triggering the subscriber(s) to get the new version, allowing it to construct the new version of the object, and even to republish it later.

Forwarding nodes or other caching points in the network, can easily cache the chunks passing by, with different algorithms. They can then replay them from their own cache whenever necessary. This means that in the PSIRP architecture, even information with the granularity of single chunks is independently cache-able and retrievable in the network.

3.5.1 Example Implementation

One of the caching mechanisms developed consists of a single cache store per node servicing all network interfaces. The cache stores clones of all incoming information chunks. Duplicate entries are not stored. A cache item is considered old and removed if the time interval since the item was entered or last used exceeds a predetermined constant value. The cache retrieval functionality relies on the “subdatachunk” type of request (developed at IPP-BAS), through which specific publication chunks can be requested, this request being sent through a reverse forwarding path. Upon receipt of such a request a caching node checks its cache for the requested content. Any chunks found are sent back to the requesting node and, if necessary, a modified “subdatachunk” request, including only the requested chunks that were not found in the cache, is forwarded towards the publisher along the reverse forwarding path. The process continues upstream until all requested chunks are delivered either by a caching node or by the publisher. Currently, only caching nodes along the path from subscriber to publisher are potential data sources, i.e., implementing an opportunistic caching

mechanism along the forwarding path from publisher to subscriber[3]. This leads to the following limitation (see Figure 3.55).

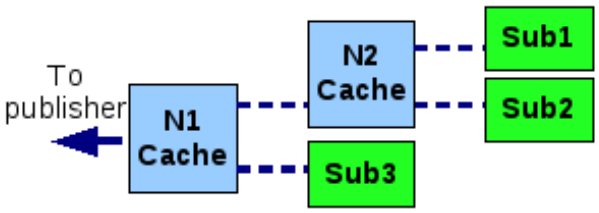

Figure 3.55: Caching Operation

If Sub1 subscribes to a publication, the publication data will be cached in N1 and N2. At some later time, before the cached data timeout is reached, Sub2 subscribes to the same publication. The entire publication will be delivered from the cache of N2 and the 'age' of the delivered cache items in N2 will be reset back to zero. Later on, the timeout interval will be reached for N1 and the publication data will be deleted. If at this point Sub3 subscribes to the same publication, the data will be fetched from the publisher even though it is still available two nodes away at N2. Data caching is implemented as a shared library written in C++ with a C interface described in detail in the 'Transport functionalities implemented in Blackhawk' section of Deliverable 3.5 - Final description of the implementation.

3.6 Transport-level Congestion Control

This section outlines the transport-level congestion control (TCC) mechanism that is currently realized within the PSIRP architecture. It resembles a flow-oriented congestion control mechanism, which can be publisher- or subscriber-controlled.

3.6.1 TCC – publisher-controlled

In this TCC version, the rate towards the congested area is controlled at the nodes which send publication data. Unlike in TCP where the communication is of an end-to-end type and the client notifies the server to reduce its sending rate, here the rate can also be modified at any intermediate node. In a PSIRP sense, these intermediate nodes act as (re-)publisher of the information items, hence the name “publisher controlled” (the term “network-assisted” is fitting, too, pointing to related work in this area).

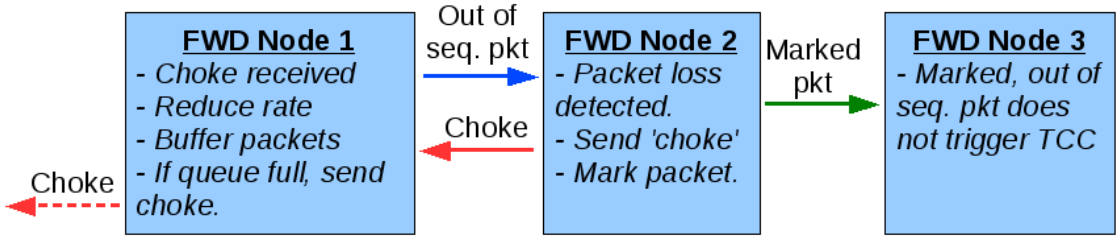

Figure 3.56: Publisher-controlled TCC

Every node divides traffic into separate flows based on packet FIds and can trigger the TCC mechanism for a given flow if it receives an out of sequence data packet, indicating a packet loss in that flow[4]. An out of sequence packet should only trigger the mechanism at the first node it is detected. For this purpose, the first node that detects the loss marks the out of sequence packet. Once marked, the packet will not trigger the TCC mechanism at subsequent

[3] It is important to realize that this solution relies on the notion of a reverse forwarding path, which has not formally been introduced in the PSIRP architecture yet. However, it underlines the potential importance for such solution.

[4] Several methods can be used to implement a sequencing mechanism, such as using algorithmic RIds or a relationship tagging mechanism (see Section 3.4.2).

nodes. A node that has reduced its sending rate due to congestion will receive data at a faster rate than its sending rate. In such conditions the data is buffered in the node's output queue. If the free buffer space gets low the node will signal the previous node, which in turn will reduce its sending rate and also start buffering data. A node communicates a congestion condition to the previous node via special choke packets. A choke packet is sent if either an unmarked out of sequence packet is received or the buffer space is getting low.

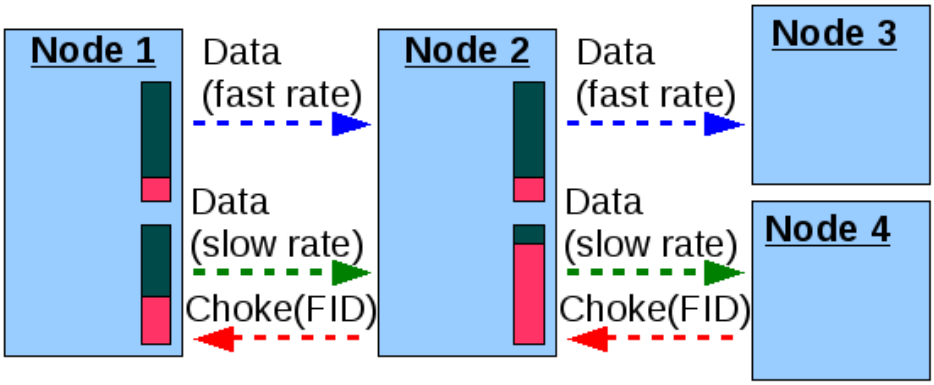

Figure 3.57: Publisher-controlled TCC – rate control

All nodes create a separate outgoing queue for each flow. This allows independent per flow rate control and avoids a situation where congestion along one route would undesirably affect the rate towards other uncongested destinations. All queues are processed in a round-robin fashion. The queue rate control is implemented as a token bucket filter. In addition all nodes have a single input queue where all incoming packets wait to be processed with the goal of using this queue as an indicator of node overload. When a node receives a choke packet for a particular flow, specified by the FId in the choke packet's payload, it reduces the sending rate for that particular flow in a single step to a predetermined value. The rate then increases multiplicatively if no other choke packets concerning this particular flow are received. Since it takes time for a node to receive and react to the choke packet, a node detecting congestion will in general experience multiple packet losses and thus send multiple choke packets. In these conditions special care is taken at the choke receiver to react only to the first choke packet of a given series. The module is implemented as a shared library written in C++ with a C interface described in an upcoming technical report outlining the implementation in more detail.

Remaining issues to be addressed:

- It takes too long for the node receiving the choke to react and reduce the corresponding sending rate, which significantly degrades TCC efficiency.
- Since nodes distinguish traffic flows based on FIds, in a situation where a subscriber subscribes to multiple publications from the same publisher, the data packets for the different publications would be part of the same data flow. For the purpose of traffic rate control this is sufficient, since a potential congestion would equally affect all data transfers between a particular publisher-subscriber pair. Lost packet detection however would not work due to the mixing of flow-sequence-numbers at the intermediate nodes (although assigning such sequence numbers through a common transport-level mechanism can prevent such mix-up). A possible solution would be for the intermediate nodes to use the meta-data to distinguish data packets with same source and destination but pertaining to different publications.

3.6.2 TCC - subscriber controlled

In this TCC version, which is TCP-like type in the sense that the TCC mechanism is always triggered by the "client" (subscriber), the rate towards the congested area is controlled indirectly through the number of chunks the subscriber requests at a time.

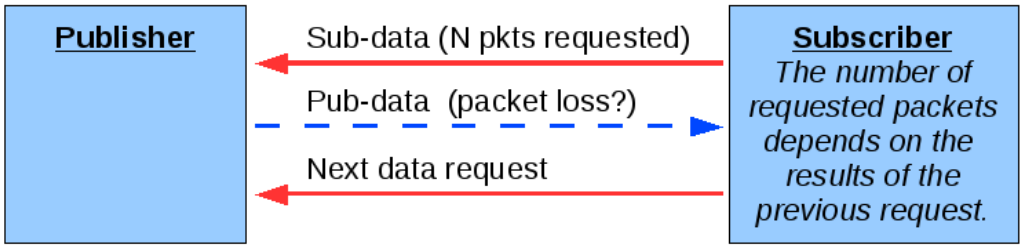

Figure 3.58: Subscriber-controlled TCC

The publication is obtained with a series of data requests containing a subset of all data chunks. A "slow start" algorithm is implemented by the subscriber starting to request two chunks. If no packet loss is detected the number of requested chunks increases by a factor of two with every consecutive request. In case of a packet loss, the number of requested chunks is reduced according to the percentage of lost packets in the previous request. Every consecutive request is triggered by the receipt of the last packet of the previous request, which incurs a round-trip-time overhead in the overall publication reception time for every request, compared to an ideal case where all chunks are requested at once and there is no packet loss. Such overhead, however, could be reduced through interleaving traffic flows on the transport level.

3.7 Rendezvous Security

This section addresses two aspects of rendezvous security, namely securing the rendezvous process itself and providing rendezvous interconnection security.

3.7.1 Securing the Rendezvous Process

An approach for securing an upgraph-based rendezvous process is presented in [Lag10]. Here, we adapt this mechanism to the more general case within the PSIRP architecture. The solution aims to prevent denial-of-service (DoS) attacks against publishers, the subscribers and the rendezvous system, while being scalable.

In the example solution, the rendezvous process operates as follows. The publisher sends a publish request to the rendezvous system. The subscriber also sends a subscription request for the publication. After matching the publication and subscription requests at the rendezvous point that serves the particular publication, a topology is formed in collaboration with the inter-domain topology formation function (see Section 3.1 for design considerations for this function). Eventually, the publisher will receive an appropriate Fid to send its publication towards the subscriber(s).

From a security point of view important questions include: whether the publisher can be trusted to serve the publication? Is a subscriber really at the network location? How to prevent DoS attacks against the subscriber? How to prevent DoS attacks against the publisher?

The chosen security mechanism is based on the traditional certificate mechanism and control messages are protected by PLA. On the forwarding layer, zFilter and zFormation mechanism are used to provide protection against DoS attacks.

In step 0, both the scope and publisher authorize themselves through certificates (C1 and C2). Mutual authorization is necessary, since without a C2 certificate a hostile scope could induce a load to the publisher by claiming that the publisher is willing to serve the publication. In step

1, the publisher sends a publish request to the rendezvous system. This request contains a certificate from the access network to the publisher (CY) along with certificates C1 and C2. In the next step (which could happen in advance of step 1), the subscriber sends a subscribe request to the rendezvous system along with the CX certificate that is given from the access network to the subscriber. In step 4, the topology formation process is initiated along with the necessary certificates. Finally, the publication is sent in step 5 along with the certificates.

The main advantage of this approach is that the inter-domain topology formation process can independently verify the validity of the subscription request and drop invalid requests immediately, before they even reach the publisher[5]. This effectively protects publishers from DoS attacks. C1 and C2 certificates state that the publisher is willing to serve the publication as well as that it is trusted by the scope. CY and CX certificates provide information about the publisher's and the subscriber's topological location. Therefore, hostile parties are not able induce load on the target network by spoofing their location.

Since publication and subscription messages utilize cryptographic identities and certificates, it is easy to limit the amount of allowed messages per time frame and per destination. This provides an additional protection, since a hostile subscriber cannot flood indefinitely valid subscription requests towards the same data source. Furthermore, by revoking or not renewing the CX and CY certificates a hostile node can be removed from the network.

The abovementioned example can be easily extended to support access control for subscriptions. In that case the subscriber would authenticate with the scope, and receive an additional C3 certificate. This certificate would be included in the subscription message and therefore intermediate nodes are able to enforce access control by verifying the validity of the C3 certificate.

The rendezvous security mechanism based on certificates and PLA is flexible, and would also work with other rendezvous solutions.

3.7.2 Rendezvous Interconnect Security

In the proposed global rendezvous system for PSIRP, presented in [PSI09], rendezvous networks are interconnected using a hierarchical Chord DHT [Gan04] implementation, which is responsible for storing global scope advertisements on behalf of the originating *rendezvous nodes* (RN). When a subscriber initiates a rendezvous and the publication cannot be found from the local rendezvous network, the subscription request is recursively routed using the Crescendo algorithm, with the modification that each message is actually a subscription operation on top of the routing layer as explained in [Vis09]. The results of the rendezvous operations can be cached in RNs and the interconnect nodes. Rendezvous interconnect operators are the organizations that provide the nodes for the interconnect architecture.

The rendezvous interconnect sub-hierarchy owner controls each sub-hierarchy of the *rendezvous interconnect* (RI). Together these owners authorize RI nodes to join that part of the hierarchy of the overlay and provide them with an address range from the Chord ring.

The availability of the rendezvous service is secured with the help of RI sub-hierarchy owners that act as trusted third parties authenticating RI nodes before they can join the DHT [Cas2002]. Each RI node is assigned an identifier range from the Chord ring using a temporary certificate signed by the RI sub-hierarchy owner in question. This prevents the Sybil attack [Dou2002] and the routing table poisoning attack. We assume that only a relatively small portion of the nodes is malicious, which makes replication an effective solution to availability. The locally optimized links in the DHT are based on the Canon hierarchy that is expected to loosely follow the underlying network topology.

The scope owner authorizes a RN to host itself with a certificate, which prevents false scope advertisements in the RI. The Canon routing algorithm guarantees that the most local

[5] It is important to keep in mind that the actual responsibility to drop malicious requests depends on the implementation of the ITF process, taking into account the design considerations in Section 3.1.

advertisement in the RI hierarchy is always found first. To avoid DDoS attacks against particular RI nodes, we utilize traffic admission control and forwarding limit at each node of the RI as presented in [Das05, p. 134]. This is possible because the SIds are uniformly hashed to the DHT nodes and honest traffic distribution should be roughly balanced when taking caching into account. As a result, the same subscriber may not continuously rendezvous with the Rids of the same scope and has the incentive to cache the results of the rendezvous operation. RI nodes also cache popular rendezvous results and store a subscription in the RI to monitor updates of the cached data. This makes popular data scalable without large investment in the home rendezvous networks (HRN) by the scope owner. The RI is further protected from attacks by the fact that it uses for communication the pub/sub model provided by the underlying routing and forwarding layers, which makes it difficult to circumvent the DHT topology. Access controlled scopes require the rendezvous message to reach a HRN trusted by the scope that can act as a policy enforcement point and encrypt the response with the public key of the subscriber. Popular access controlled scopes therefore require the rendezvous network to have capacity large enough to handle incoming subscriptions and possible DDoS subscription attacks by botnets. To address this, it is possible to replicate the scope implementation to multiple RNs. To avoid a RI node becoming a hotspot, it is possible to create multiple advertisements in the RI by adding *salt values* [Zha01] to the SId before hashing it to determine the RI node.

On the rendezvous system level, we do not have to consider the tussle for good human-readable names as the scope names are always relative to a namespace created by the public key of the SId. On the other hand, each scope advertisement consumes storage resources of the RI and a fair allocation should be enforced by the RI mechanism. This can be achieved, for example, with per-node quotas for each sub-hierarchy in the RI, which gives each sub-hierarchy the incentive to control resource usage of each of its clients. The quotas can be based on contracts between interconnect operators and their customers.

4 Major PSIRP Architecture Contributions

In this section, we attempt to summarize the main PSIRP contributions on the architectural level. It is clear that these contributions cannot be seen (and would have not been made) without the significant contributions on the implementation and evaluation levels. However, we attempt to frame these contributions in an architectural context.

4.1 From Principles to Invariants

When starting the PSIRP project, it was seen as a crucial exercise to formulate design principles upon which we could build our work and the (architectural) outcome. This led to the formulation of four major PSIRP design principles [PSI09], in addition to general design principles that can be seen as crucial for many large-scale design efforts.

Throughout our work, however, we came to formulate stronger invariants for information-centric designs that follow these principles as well as the other ideas developed in the PSIRP project. In other words, rather than loosely outlining principles for a design, we are confident enough to outline invariants for ANY information-centric architecture akin to PSIRP. This is a strong architectural result that will need constant evaluation in the future. One of the goals of such an evaluation is to determine the validity of these invariants for a wider range of architectures beyond PSIRP.

4.2 From Functions to Design Choices

The information-centric starting point of the PSIRP project naturally led to the formulation of three crucial functions to be implemented, namely the finding of information, the construction of a delivery graph for the information, and the delivery of the information along this graph.

Throughout the project, we have come to outline concrete design choices for these functions, not only implementing the functions themselves but also addressing related issues that come from their implementation, such as security considerations, considerations for identifier choices, and many more. This has created a wealth of information around these design choices from which implementation choices can be selected, evaluated, and tested.

4.3 From Scoping to the Potential for a Rigorous Design Framework

The concept of scoping [PSI09] quickly became a central theme in our architectural work and throughout the formulation of design choices for various architecture components. With the formulation of our architecture invariants, however, this concept has been extended towards a foundation for layering that not only addresses our vision of providing a mapping of social constructs and other relationship between information items onto a thin inter-networking layer [PSI08] but also provides the potential for a rigorous design framework under the inter-networking layer. This layering concept in the context of scoping within implementation regions provides a region of consistency, which can be used for applying, e.g., optimization and control theory techniques that fully utilize the heterogeneous network resources that are likely to exist below the waist of the architecture. While the project has not addressed the nature of such a rigorous design framework, we firmly believe that our thinking around scoping and layering has paved the way for such a framework to be studied and applied.

4.4 From a Vision to a Research Agenda

PSIRP has been from the beginning a vision-led effort, driven by the firm belief of a few people that such a dramatic change of the inter-networking architecture would benefit the Future Internet in many ways. While this vision was based on some concrete ideas for its realization, only the execution of the project and the wealth of information that surrounds its execution have led us to sharpen our research agenda for the future. In other words, the

project and its partners with which it collaborates so successfully has been working towards concretizing the potential work in many areas through follow-on projects (such as the PURSUIT FP7 project which will start in September 2010), national projects (such as various ICT SHOK efforts in Finland) as well as international collaborations (such as through the current US NSF efforts in the Future Internet architecture funding round). Much of the work executed in these efforts is a direct result of the groundwork that PSIRP has laid in its efforts. Some examples for such a growing research agenda are congestion control and caching solutions, multi-path resource pooling, algorithmic identification, naming solutions, rendezvous solutions beyond global rendezvous, applicability of control and optimization theory, and many more. The formulation of this research agenda would have been impossible without the collaboration among the PSIRP project partners and the growing collaboration with external partners.

5 References

[And04] T. Anderson, T. Roscoe, and D. Wetherall. “Preventing internet denial-of-service with capabilities”. Hotnets II, pages 39–44, 2004.

[Bro09] I. Brown, “Socio-economic drivers of Internet development”, 2009

[Cas02] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan Wallach, "Secure routing for structured peer-to-peer overlay networks", Proc. of the 5th Usenix Symposium on Operating Systems Design and Implementation, 2002.

[Das05] N. Daswani, “Denial-of-service (dos) attacks and commerce infrastructure in peer-to-peer networks (draft),” Ph.D. dissertation, Stanford, Jan. 2005.

[Dou02] John Douceur, "The Sybil Attack", In Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.

[Est09] C. Esteve, P. Jokela, P. Nikander, M. Sarela, J. Ylitalo, “Self-routing denial-of-service resistant capabilities using in-packet bloom filters”, in the 5th European Conference on Computer Network Defense (EC2ND), 2009

[Far06] A. Farrel, J.-P. Vasseur, and J. Ash. “A path computation element (PCE)-based architecture”, IETF RFC 4655, 2006.

[Far07] A. Farrel. “MPLS-TE Doesn't Scale”, MPLS 2007, available at http://www.olddog.co.uk/Farrel_Scaling-MPLS-TE.ppt, 2007.

[Fot09] N. Fotiou, G. C. Polyzos, D. Trossen, “Illustrating a Publish-Subscribe Internet Architecture”, Proceedings of Future Internet Architectures: New Trends in Service Architectures (2nd Euro-NF Workshop), 2009.

[Gan04] P. Ganesan, K. Gummadi, H. Garcia-Molina, “Canon in G Major: Designing DHTs with Hierarchical Structure,” in ICDCS’04 IEEE Computer Society, pp. 263–272, 2004.

[Gel82] D. Gelernter, A. J. Bernstein. “Distributed Communication via Global Buffer”, In proceedings of the first ACM SIGACT-SIGOPS symposium on Principles of distributed computing, 1982.

[Hui02] J. Huigen, “OECD reviews of regulatory reform”, 2002

[Jac09] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, R. L. Braynard, “Networking Named Content”, ACM CoNext, 2009.

[Jok09] P. Jokela, A. Zahemszky, C. Esteve, S. Arianfar, and P. Nikander, “LIPSIN: Line speed publish/subscribe inter-networking”, In Proceedings of ACM SIGCOMM’09, Barcelona, Spain, Aug. 2009.

[Lag10] D. Lagutin, K. Visala, A. Zahemsky, T. Burbridge, and G. F. Marias, “Roles and Security in a Publish/Subscribe Network Architecture,” In Proceedings of the IEEE Symposium on Computers and Communications (ISCC'10), Riccione, Italy, June 2010.

[Lak04] K. Lakshminarayanan, I. Stoica, and S. Shenker. “Routing as a service”, 2004.

[Mey07] D. Meyer, “Report from the IAB workshop on routing and addressing, RFC 4984”, 2007

[PSI08] D. Trossen (ed.), “From Design for Tussle to Tussle Networking: PSIRP Vision and Use Cases”, PSIRP Technical Report TR01-008, 2008

[PSI09] M. Ain, D. Trossen, P. Nikander et. al., “PSIRP Deliverable 2.3: Architecture Definition, Component Descriptions, and Requirements,” February 2009

[PSI10] J. Riihijarvi (ed.), “Final architecture validation and performance evaluation report”, 2010

[Quo07] B. Quoitin, “Evaluating the benefits of the locator/ID separation”, 2007

[Raj08] J. Rajahalme, “Incentive-compatible caching and peering in data-oriented networks”, ReArch workshop, 2008

[Urd09] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen, "A Survey of DHT Security Techniques", ACM Computing Surveys, 2009

[Vai09] C. Vaishnav, “The end of core: should disruptive innovation in telecommunication invoke discontinuous regulation?”, Ph.D. thesis, MIT CSAIL, 2009

[Vis09] K. Visala, D. Lagutin, and S. Tarkoma, “LANES: An Inter- Domain Data-Oriented Routing Architecture,” in Proceedings of ReArch’09 workshop, Dec. 2009.

[Wen06] D. Wendlandt, D. Andersen, and A. Perrig. “Fastpass: Providing first-packet delivery”, Technical report CMU cylab, 2006.

[Yan07] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai. “Tesseract: A 4D network control plane”, In NSDI’07, 2007.

[Zha01] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, “Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing,” UC Berkeley, Tech. Rep., Apr. 2001.