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

DELIVERABLE D2.4

Update on the Architecture and Report on Security Analysis

Title of Contract	Publish-Subscribe Internet Routing Paradigm
Acronym	PSIRP
Contract Number	FP7-INFSO-IST 216173
Start date of the project	1.1.2008
Duration	30 months, until 30.6.2010
Document Title:	Update on the Architecture and Report on Security Analysis
Date of preparation	02.09.2009
Author(s)	Mark Ain (TKK-HIIT) (editor), Dirk Trossen (BT), Kari Visala (TKK-HIIT), Trevor Burbridge (BT), Paul Botham (BT), Petri Jokela (LMF), Jukka Ylitalo (LMF), Dmitrij Lagutin (TKK-HIIT), Jimmy Kjällmann (LMF), Giannis Marias (AUEB)
Responsible of the deliverable	Dirk Trossen (BT)
	Phone: +44 7918 711695
	Email: dirk.trossen@bt.com
Reviewed by:	Dirk Trossen (BT), Pekka Nikander (LMF), Trevor Burbridge (BT), Janne Riihijarvi (RWTH), 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/

European Commission
Information Society and Media

Table of Contents

1 Introduction 4
2 Algorithmic IDs 5
2.1 Introduction 5
2.2 Usage of Algorithmic Identifiers 5
2.2.1 Subscription Management 5
2.2.2 Forwarding State Aggregation 6
2.2.3 Caching 7
2.2.4 Coding 7
2.2.5 Return Path 7
2.2.6 Flow Control 7
2.2.7 Content Fragmentation 7
2.2.8 Sequence Numbering 8
2.2.9 Error Control & Reliability 8
2.2.10 Announcements 8
2.3 Technical Approaches 8
2.3.1 Algorithmic Identifiers 9
2.3.2 Relational Tags 9
2.4 Suitability of Technical Approaches 10
2.5 Constraining the Generation of Algorithmic Identifiers 11
2.6 Information Dissemination and Conventions for It 11
2.7 Security 12
2.8 Example: A TCP-like Transport Service 13
2.8.1 An Extended Transport Service 14
2.8.2 A Layered Multicast Transport Service 15
3 zFilter-based Forwarding 16
3.1 Background 16
3.2 zFilter Principles 16
3.3 zFilter Forwarding Mechanism 17
4 Security Architecture 19
4.1 Forwarding-level Protection 20
4.1.1 Forwarding Vulnerabilities 21
4.1.2 zFilters Work as Capabilities 21
4.1.3 z-Formation Principles 22
4.1.4 Topology and z-Formation 23
4.1.5 Binding between Rendezvous and z-Formation 23
4.1.6 Attack Mitigation 24
4.2 Authorization in the Rendezvous System 24
4.2.1 Scope Ownership 25
4.2.2 Node-Internal Communication 25
4.2.3 Main Scope and Security Sub-scope 26
4.2.4 Creating Scopes 26
4.2.5 Authorizing Publishers 28
4.2.6 In-packet Certificates 30
4.2.7 Registering Provisioning and Interest for Content 30
4.2.8 Summary of Security Properties 31
4.3 Security in Inter-connected Rendezvous Systems 31
4.3.1 Data integrity and Confidentiality 32
4.3.2 Approaches to Rendezvous Interconnect Formation 32
4.3.3 Attack Scenarios 32

4.4 Packet Authentication for Forwarding Security 36
4.4.1 Packet Verification 37
4.4.2 Other Features 38
4.5 Network Attachment 38
4.6 Preventing Spam through Notarization 40
4.6.1 Subscribers as Spam Initiators 41
4.6.2 Publishers as Spam Initiators 41
4.7 Reputation Services for Publish/Subscribe 43
5 Inter-domain Topology Formation 45
5.1 Design Goals 45
5.1.1 Design Requirements 45
5.1.2 Conceptual Components 46
5.2 Design Choices 47
5.3 Evaluating Design Choices 49
5.3.1 Socio-economic Starting Point 49
5.3.2 Methodology 50
5.3.3 Causal Loops 50
5.3.4 Reference Modes 52
5.3.5 Summary 54
6 Conclusions 55
7 Terminology 56
8 References 58

1 Introduction

This report forms the fourth deliverable in the PSIRP Architecture series. The first report D2.1 set down some high level principles for the architecture, along with a review of the state of the art. D2.2 provided an initial architecture concept, while D2.3 provided much greater detail of the architecture components and their interfaces.

In this report, we do not attempt to provide details of the overall architecture and all of the components. Readers requiring a wider technical overview should see D2.3. Instead, we provide updates on selected components; namely forwarding and the inter-domain topology formation. This report also concentrates on the key cross-component discussions of identifiers and security. The reason for the selection of these four topics is that they are all tightly related within the architecture. Inter-domain topology formation provides the route information to the forwarding components. How the route is selected, and how the information is forwarded across the network to subscribers are key security problems, and identifiers (in the form of Forwarding Identifiers or FIds) are at the heart of the network.

Being able to relate network identifiers, without understanding the information that they relate to, can be a powerful tool for the network components, transport services, and other middleware. However, such relationships can also be used to infer confidential information or to coordinate attacks. We have previously demonstrated such algorithmic identifiers (AlgIds) in the use of the zFilter in the forwarding component, where the zFilter is related to a set of Link IDs, forming the multicast tree to the subscribers. Without security it may be possible for onlookers to analyse the path of the information (such as determining the sender or receivers), or to choose unauthorised paths within the network, even deliberately setting up loops or flooding the network and receivers.

Although we have previously presented Packet Level Authentication for forwarding security, we have not shown how this lowest level security check relates to the demands of other architecture components and the end users themselves. In this report we show that all of the users, components and third party (security) services within the PSIRP architecture are connected in order to achieve their legitimate goals.

The remainder of the deliverable is organised as follows. Section 2 introduces our current work on algorithmic identifiers before presenting the updates on forwarding in Section 3. Significant focus is placed on our security architecture, presented in Section 4. The updates on inter-domain topology formation are given in Section 5, before concluding this document in Section 6.

2 Algorithmic IDs

The following section presents our current work on algorithmic identifiers. We outline usage, technical approaches as well as examples in the various subsections.

2.1 Introduction

Information delivered across a PSIRP network may, of course, have complex associations. The actual meaning of the information, and hence the subtlety of such associations, can only be understood by the applications and end users. One item of information may be a film, whereas another item is the biography of an actor or a piece of music used in the film. Such associations may only ever exist outside of the network in people's minds, or they may be facilitated by search and indexing services (e.g., IMDB [IMDB]).

Associations may also be more explicit, using semantic languages or ontologies to describe the meaning of information and hence its relationship to other items. Finally, information can include direct references to other pieces of information (such as the inclusion of a PSIRP Rendezvous Identifier (RId) or some other application identifier that can be resolved to a RId).

All of these approaches are permitted, and expected, to be used, since the PSIRP network is deliberately agnostic to the information it carries, as outlined in its design principles [PSI09a].

It is also possible for relationships to be expressed and understood by middleware, transport services or network components, in order to improve the performance of either applications or network functions. This is the subject of the work presented here. We use the term *Algorithmic Identifiers* (or AlgIds) since we originally assumed that solutions would create graphs of identifiers through automated algorithms. Subscriber-side services can then apply the same algorithms to subscribe to the related information. As we shall see later, our work has also moved beyond these applications of AlgIds to methods of tagging information in the network. In the next section, we motivate the work by exploring a number of potential applications of such schemes.

2.2 Usage of Algorithmic Identifiers

In this section, we examine a wide range of potential network uses. . It is important to note that these examples do not represent the final thinking with respect to these functions, e.g., in the error control area. They merely outline potential usages for algorithmic identifiers. Only a thorough future evaluation will shed light on their applicability within the wider PSIRP architecture. Furthermore at this stage, we do not consider whether such functions would be applied on network equipment such as routers, on the end host as part of the protocol stack, or as in-path services

2.2.1 Subscription Management

In network architectures such as PSIRP or publish-subscribe overlays, such as Scribe [Cas02a] and Bayeux [Zhu01], the subscriber selects and subscribes to individual identifiers. Since such identifiers should be uniformly distributed throughout the identifier space in order to avoid routing hotspots, the selection of contiguous identifiers through the use of wildcards or identifier ranges is not meaningful. Thus, in order to select a range of information communicated over multiple identifiers, other mechanisms are required. PSIRP provides *scopes* (denoted by SIds, see [Psi2009a]) within which individual information identifiers (RIds) are published and subscribed. It is therefore theoretically possible for a subscriber to subscribe to an SId to receive information on all subordinate identifiers (which may be RIds or other SIds arranged hierarchically). However, scopes within PSIRP are intended for a different purpose, not for replicating the hierarchical topic-based structures found in Message Oriented Middleware products. In PSIRP, a subscription to an SId is only intended to provide control information for the delivery of the underlying RIds. A component of the PSIRP rendezvous system subscribes to a SId to receive subscription messages for underlying RIds along with policies to manage the communication over these RIds. The rendezvous system component is not interested in receiving the actual data published on the RIds since this should be forwarded directly towards the end subscribers.

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 service level, but publishing actions throughout various level 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 AlgIds.

In Message-Oriented-Middleware (MOM) topic hierarchies, typically the first option is used, therefore the topic is used to carry information about the aggregate concept (for example information about a company division). In PSIRP, scopes have a similar semantic where publications to a scope provide meta-data such as policies to control publications to RIds within that scope. 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.

Typically, a topic-based MOM will implement the second semantic option for the subscriber. A subscriber to a topic will receive information relating to the aggregate concept, along with receiving all the information sent on underlying topics (providing a quick subscription to many channels). In contrast, PSIRP scopes implement the first semantic option.

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.

2.2.2 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 separate forwarding path specification. Since the PSIRP network architecture separates a (fast) forwarding path from the (slow) rendezvous path, such techniques are applicable to PSIRP. PSIRP specifies forwarding links using *Forwarding Identifiers* (Fids, see [Psi2009a]). Thus, we can consider that these can be either algorithmically generated or aggregated into longer path identifiers. One algorithm already considered within PSIRP is to aggregate separate FIds within a Bloom filter in the packet header (see Section 3).

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.

2.2.3 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*.

2.2.4 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.

2.2.5 Return Path

Many applications may wish to operate in a client-server type of 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.

2.2.6 Flow Control

In the discussion on coding, we have already touched on how an application (or transport) might adapt its receiving rate by subscribing to different compactions of the information with corresponding AlgIds. However, AlgIds could also be used to perform flow control without alteration of the information. 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.

2.2.7 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 be explicitly listed in content meta-data. Such fragmentation can also be structured semantically – e.g., voice, video, biography, trailer etc. components of a movie.

2.2.8 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 so as to receive the next item in the series. This allows previous items to be repeated without burdening applications that have already received them.

2.2.9 Error Control & 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.

2.2.10 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.

2.3 Technical Approaches

We have explored a range of solutions that can enable some, or all, of the applications described above. In brief we split such solutions into two categories:

- **algorithmic identifier** schemes where identifiers are created using functions of other identifiers;
- **relational tagging** schemes where tags are applied in the packet header to form relationships between two otherwise (seemingly) random identifiers.

We can immediately see that the two solution categories have a number of key differences.

Algorithmic Identifiers	Relational Tags
Relationship is known before the generation and use of the identifier	Identifiers can be related at any point
Relationship is fixed	Relationships can dynamically change through the removal, modification or addition or further tags.
Only the producer of the identifiers can make relationships	Any party can tag information within the network
Relationships are universal	Different publishers may assert different relationships
Potential receivers can subscribe to related information in advance	Receivers have to have current subscriptions in order to receive tags

2.3.1 Algorithmic Identifiers

As we have discussed, in the algorithmic identifier schemes, identifiers are created to form a relationship graph from other identifiers. Two specific sub-cases are especially worth considering:

- an identifier is generated from another single identifier
- an identifier is generated from multiple identifiers.

For example, a single root identifier can be used to construct a tree, representing both parent-children relationships (such as fragments of content) or sequences (either as ordered siblings or as a path from the root). Multiple root identifiers will result in multiple disconnected trees. Trees can be extended at any time, but can never be merged. Cyclic relationships are not possible, except from accidental collisions resulting from the functions used to generate the identifiers (for example hash functions).

By knowing an identifier, it is possible for a receiver to generate related identifiers (although of course such identifiers may not exist). Depending on the function used, it may be possible to generate antecedent identifiers. For example, if the function is a block cipher, and the branching order is known, then the child identifier and branch number can be used to recreate the parent identifier. If the branch number is not known, then a set of possible parents may be surmised (and an even larger set of possible grandparents, siblings etc.). If a one-way function is used, then it is only possible to generate descendent identifiers.

In the second case, the related identifier is formed from multiple parents, for example using a Bloom filter. In this case information is lost, and it is not possible for a potential receiver to generate the parent identifiers from that of the child. Since any two or more identifiers may be used, or re-used, the resulting information structure is a directed graph. Once an identifier has been created, no new antecedents may be added to the graph.

2.3.2 Relational Tags

Relational tags seem to have much to offer due to the flexibility they provide. However, the fact that a receiver cannot determine (all) their subscriptions in advance may prohibit their use for many applications. We can consider two sub-categories of relational tagging schemes:

- where tags are sufficient to produce the related identifiers
- where tags can only be used to test relationships between two known identifiers

In the first case, the tag may actually be the related identifier(s). This is perhaps the most flexible of all options, allowing the receiver to determine directly all related identifiers, yet also enabling different publishers to assert different relationships amongst already existing identifiers. However, the obvious drawback is the space that is required to list all the related identifiers (and their relationship in a graph from the original identifier). This approach has been used in previous work for the indexing of messages on multicast announcement channels [Sop03].

Alternatively, tags may be combined algorithmically with the first identifier to produce one or more related identifiers. This can be considered as a hybrid scheme combining some of the properties of both algorithmic identifiers and relational tagging. Thus, different publishers can create different related identifiers from a common root identifier, but are still unable to relate existing identifiers.

In the second case above, tags are created from two or more related identifiers. Since the tag represents a compacted form of the identifiers, it cannot be used to generate the related identifiers, but only to conduct probabilistic testing of the relationship. A technical implementation of such a scheme would be a Bloom filter formed from related identifiers. Different relationships in a graph of information can be represented in multiple tags (for example on tag for siblings, one for children). The use of zFilters in the forwarding design and implementation (see Section 3) can be considered as such a tag, signifying a relationship between the packet and a set of forwarding identifiers (FIds).

2.4 Suitability of Technical Approaches

In the previous section, we outlined a number of broad technical schemes without going into the details of possible functions. This provides us a sufficient starting point to comment on how these different approaches can meet the requirements of the application outlined earlier in this report. It is important to note that the hybrid algorithmic identifier/relational tag scheme is not considered since its applicability for the applications can be considered largely similar to the algorithmic identifier scheme from which it is extended.

Application	*AlgId (single parent reversible)*	*AlgId (single parent one-way)*	*AlgId (multiple parent)*	*Tag (identifier list)*	*Tag (test only)*
Subscription	Y	P	N	Y	N
Forwarding	N	N	Y	Y	Y
Caching	Y	P	P	Y	N
Coding	Y	Y	N	Y	N
Return	Y	Y	N	Y	N
Flow	Y	Y	N	Y	N
Fragmentation	Y	Y	N	Y	N
Sequencing	Y	Y	N	Y	N
Error & Reliability	Y	Y	N	Y	N
Announcements	Y	N	N	Y	N

Y=mostly suitable, P=Partly Suitable, N=not suitable

The above table provides only a guide to the suitability of the schemes to the various applications, and we do not have room in this report to detail our full discussions. As an example, let us consider the suitability of the schemes for subscription management. In subscription management we may need to perform the following actions:

- calculate and subscribe to sub-information categories
- calculate and subscribe to super-information categories
- calculate and subscribe to meta-information categories.

The calculation of identifiers by the receiver means that schemes providing testing of relational tags are not sufficient, whereas listing related identifiers is capable of expressing any relationships. The bi-directional ability is required to calculate and subscribe to both child and parent information categories. Algorithmic identifier schemes with multiple parents can only be used to calculate identifiers that are used to transmit information relevant to those subscribing to an exact set of existing identifiers and is considered too narrow to be useful.

2.5 Constraining the Generation of Algorithmic Identifiers

We have considered several schemes that allow a potential receiver to generate and subscribe to algorithmically generated identifiers. What we have not considered so far is how the receiver knows which identifiers (in a potentially infinite graph) are valid and useful for their requirements. For example, how does a receiver know how many fragments comprise an item of content? It may be that the application will have a constant number of related identifiers. However, in other cases, it seems clear that some information must be signalled from the generator of the algorithmic identifiers to the potential receivers. One option is to signal such information from the publisher along with the identifier(s) from which the receiver should start the graph construction. This can occur as part of the (meta-) information payload, which is received by the application and then signalled to the transport or network functions.

Another approach is to include some structure information in the packet along with the algorithmic identifier. For example, the last identifier at the end of a sequence or the leaf of a tree can have an associated flag that indicates that a receiver should not attempt to generate any further identifiers. Of course, this approach only works if the receiver has received such a packet and is not simply trying to generate and subscribe to such identifiers in advance. A more general approach would be to signal structure information along with the root or other key identifiers to which we expect the receiver has already subscribed. For example, the publisher can indicate that there are 100 fragments in a two-layer tree. Such information may be included in the packet header along with the root identifier, but can also be considered information in its own right and transmitted on a separate identifier. This identifier is, of course, generated algorithmically from the root identifier. In the next section, we briefly discuss how to disseminate information related to algorithmic identifiers.

2.6 Information Dissemination and Conventions for It

We can see that there are several types of information that need to be disseminated for an algorithmic identifier scheme to operate. These include the following knowledge:

- How algorithmic identifiers in the information graphs relate to application or transport concepts. For example, knowing that the first sibling is used for retransmission requests.
- The limits of the information graph. For example, knowing that there are only 20 fragments of an item of content. Alternatively, for actual lists of related identifiers, the knowledge of the list itself.
- The algorithmic identifier scheme being used (if not universal)
- The function used in the algorithmic identifier scheme (since different functions may be ‘pluggable’ into the same scheme).
- Any secrets such as cryptographic keys that are required to make the associations between identifiers.

There are a number of conventions for disseminating this information:

- The information may be universally true. For example, if only one algorithmic identifier scheme is implemented (e.g., structure of the packet header).
- The information may be universally true for a specific application or a transport protocol (e.g., structure of transport header, knowledge of relationship meanings).
- The information may be transmitted outside of the algorithmic identifier implementation, e.g., in the payload of previous transmissions.
- The information may be signalled in the packet or in an extended transport header. For example, a relational tag may be included in the packet header.

- The information may be transmitted using a Rendezvous Identifier to which receivers subscribe. In this case, we have the related problem that the rendezvous identifier must be obtained through one of the means listed previously. This fits with the PSIRP paradigm that all information can be separately addressed using RIds.

As an example, we consider the relational tagging scheme described earlier where the tags are actually the related identifiers. These related identifiers can be described in a format such as XML that can be used to provide ordering and structure to the listed identifiers. For a generic scheme, XML would describe relational properties in a graph, such as children, parents, siblings etc. In more specific application schemes, XML could also attribute meaning to the relationships (e.g., error control, error retransmission).

In this example, this information about related identifiers is in itself given a RId. The next problem becomes how to relate the root information RId with this new RId describing related information. One method would be to standardise an optional field in the packet header to transmit this first related RId through which all other related RIds can be obtained. The advantage of storing as much relational information at the edge is obvious since it provides the most flexibility and mitigates some of the concerns over state in the packet, especially for those who have no interest in the related information.

A drawback of such an approach might be that there will be a significant latency between receiving a packet comprising the root information and knowing about the related identifiers. Thus, this scheme may not be sufficient for transport protocols. Of course, it is possible to operate multiple algorithmic identifier schemes simultaneously. For example, some applications may use the optional related information identifier field, whereas others may apply an algorithm to the original identifier to directly calculate a smaller fixed number of related information identifiers that are required immediately. The presence of such schemes may be simply assumed by the transport or application or may be signalled using a code-point in the packet header.

2.7 Security

The use of algorithmic identifiers exposes the system to a couple of potential attacks. Any eavesdropper can attempt to use the algorithmic identifier relationships to expose confidential information. Whereas normally the payload can be encrypted and the identifier (RId) may appear random, the use of algorithmic identifiers allows the eavesdropper to identify relationships in the information that is being received. The structure of such relationships, along with other characteristics such as the size of the information (pieces), can be used to infer the information that is received.

Other potential attacks can come from illegitimate senders abusing the algorithmic identifier schemes. For example, a malicious sender might assume that subscribers are also subscribed to a range of algorithmic identifiers and use these to conduct Denial-of-Service attacks or attempt phishing attacks on identifiers on which the subscriber is expecting legitimate traffic. These attacks should not be possible if the algorithmic identifiers are subject to the same security constraints as the original information identifier. Any party incapable of sending to the original identifier should be incapable of sending to any related algorithmic identifier.

One method to maintain the confidentiality and integrity of the algorithmic identifier schemes is to use a secret key during the generation of the identifiers. If this approach is taken, the function used to generate the algorithmic identifier must have good security properties (such as a block cipher or a secure hash function). Simple bit-wise operations will not provide any protection against an attacker gaining the key since the plain and cipher text pairs (the original and algorithmic identifiers) are easily available through eavesdropping on the network near the publisher or the subscriber.

The use of relational tags should have similar security controls. It may appear sufficient that only the publisher of a rendezvous identifier should be able to insert the relational tags. These tags may be protected using Packet Level Authentication along with the rendezvous identifier and other packet header fields. However, the transport service example below shows a requirement for these relational tags to be written by parties subsequently to publication, and in theory there is no reason why identifiers should not be related at any point in their lifecycle by other parties. Thus, in some applications these relational tags may remain unprotected, or alternatively their integrity may be assured using different keys to those of the original publisher.

In general, since algorithmic identifiers and relational tags are meant to provide an extremely quick method of relating information (since a longer approach can always be provided by the application payload), security mechanisms should be as lightweight as reasonable. We can exploit techniques such as caching previously checked relational tags.

2.8 Example: A TCP-like Transport Service

In this section, we consider how some of the ideas we have introduced above can be used to support a transport service for PSIRP, providing similar properties to TCP. It is very important, however, to realize that what we present in this section is only an example of what can be done with algorithmic identifiers. In no way do we suggest that these transport protocols should actually be adopted as described. On the contrary, studies on transport protocols design and realization are only at the beginning in our project.

Whereas TCP provides a full duplex service between two hosts, in PSIRP it makes more sense to consider a simplex service from the publisher to a number of subscribers.

Removing the concept of endpoints from TCP, we can observe that there are several pieces of information that can all be separately addressed using RIds in the PSIRP architecture. The primary RId is used to transmit the application information from the publisher to the subscribers. This information, for example, may be a video stream. TCP also inserts its own information such as a sequence number and a checksum. In PSIRP, we can consider similarly packaging the application payload with a transport header without changing the RId.

The subscriber will return transport-related information, such as the window size and acknowledgement number, to the publisher. Since information flow in PSIRP is one-way from a publisher to a subscriber, this information must be carried over one or more additional RIds to which the original publisher can subscribe. Since PSIRP provides multicast forwarding paths, it is unlikely that the publisher will want to receive an unknown number of receiver responses. However, we assume that there is a set of RIds to which the publisher can subscribe to receive aggregated flow and error control information from the receivers, and a different set of RIds on which the receivers send their individual information. We assume that there is a sequence of forwarding nodes in the network and that each has an associated manager to perform feedback aggregation. Between each feedback manager, we require another set of RIds corresponding to different levels of aggregated feedback information.

The publisher will use an algorithm to automatically calculate the RIds on which flow and error control information are to be received from the nearest network feedback manager, and it will subscribe to these before publishing any information on the original identifier.

For the purposes of this example, we assume that each successive feedback manager needs to subscribe to different RIds to receive the next set of successively finer-grained feedback information. This can be achieved by the feedback manager writing a relational tag into the packet header. This tag is used as an input into the algorithmic identifier function to produce a different set of feedback RIds to which it subscribes. Each successive feedback manager then uses the received tag to calculate where it should send feedback information to, before changing the tag to receive feedback information from the next set of feedback managers, and ultimately receivers.

Once the feedback con-cast tree is created, receivers can request the retransmission of lost or corrupted information. One option is to use a single retransmission channel, again with a separate RId, which the publisher and receivers calculate automatically from the original information RId.

An alternative approach to using feedback aggregation is to selectively manage the feedback directly from the subscribers. This approach is used in TCP-Friendly Multicast Congestion Control (TFMCC) presented in IETF RFC 4654. If this approach is taken, the algorithmically generated identifiers can be used to suppress the feedback and receive congestion reports from the subscribers. In other schemes, the subscribers may use algorithmically generated identifiers to communicate with each other in order to co-ordinate their feedback.

As we have already mentioned, although this is an interesting example of how algorithmic identifiers could be used, in practice other methods of flow control are more suited to multicast networks. Later we present another transport use case, which uses layered fragmentation or coding to allow each receiver to control the information flow.

2.8.1 An Extended Transport Service

In the TCP-like transport service above, we implicitly assumed that fragmentation would occur using a transport header to provide sequencing information, while maintaining the original RId for every packet.

In this section, we explore how algorithmic identifiers can also be used to supplement the original RId for the purposes of fragmentation of information and sequencing of network packets. This is mainly a theoretical flight-of-fantasy to explore how far we can push the use of algorithmic identifiers since we do not expect the network subscription performance to allow such dynamic usage.

We start by assuming that a set of subscribers is already listening to an RId that tentatively relates to the overall content that they are interested in receiving. Since we also assume that the content is fragmented and that each fragment is assigned to a packet with a separate Rid, it will be necessary for each subscriber to identify and subscribe to these fragment identifiers in order to receive the full content that they are expecting.

A simple approach would be to utilise a scheme that lists the fragment identifiers. Each subscriber (transport service) would receive a first packet on the original content identifier. This packet would contain as a payload a structured list of identifiers that can be processed by the receiving transport. The transport will then subscribe to and receive each fragment and reassemble the content on behalf of the application. Instead of using the general error feedback identifier in the first example, the transport can now explicitly request lost or corrupted packets on error control identifiers of their own. This can be used to scale distributed content systems where different fragments are shared from different servers (which can then subscribe to their own proportion of the error control information).

Alternatively, we can use an algorithmic identifier scheme to automatically generate the identifiers of the fragmented packets. The subscriber transport would automatically generate (e.g., using a hash function) a list of fragment identifiers and subscribe to them in advance.

This highlights the requirement for some delimitation to be disseminated from the publisher to the subscriber, since otherwise the subscriber would not know how many fragments to subscribe to. One way to achieve this would be for the ‘original’ content identifier to not contain the first packet of the information, but instead convey meta-information about its delivery. This meta-information could include the first packet identifier (if the original identifier is not used as the first element if the algorithmic scheme), the function used, and information to limit the application of the function (e.g., number of steps in a hash chain, index of first and last leaf of a tree etc.).

Although the exploration of using separate RIds for each fragment is interesting, it has potentially major flaws in terms of performance, since the subscriber will potentially suffer from some delay in the many subscriptions it must create, and there are other problems in synchronising the receiver subscriptions before sending each fragment.

2.8.2 A Layered Multicast Transport Service

Another approach to flow control over multicast is to let the receiver manage their rate by providing layered multicast channels. The basic principle is that the receiver can then join the channel, or channels, that in combination deliver the information at the desired rate. One technique is simply to have different flow rates (for example different video encodings) on different multicast channels. A more sophisticated approach would be to fragment the content (allowing the subscriber to receive multiple fragment simultaneously), or in the case of real-time streams to use a layered encoding [Bye02] [McC96].

If we use this approach, then each layer must be transmitted using a separate RId so that the receivers may selectively subscribe to it. We can use an algorithmic identifier to produce the set of such RIds for the individual layers, or transmit a list of layer RIds to the subscribers.

Each subscriber can monitor the congestion they are causing (e.g., through packet loss or congestion marking) and adapt their receiving rate accordingly [ElK01][Cho09]. It is even possible that the receiver may receive such layers from different publishers and drop channels that are causing the most network congestion.

Alternatively, or in combination with receiver control, the packet forwarders in the network may apply prioritisation based upon the ordering of the algorithmic identifiers (or a separate codepoint). This can enable the network to drop the traffic that is elastic (for example, higher bit-rate enhancement layers) and maintain the flow of the essential basic service. In addition to prioritisation of packet dropping, the network may also decide to suspend the onward transmission of certain layers. This can be useful to avoid high wastage (delivery of unusable packets) that results from heavy dropping [Oou05]. In past studies, flow identifiers and priority labels have been used to indicate which layers may be dropped. Using algorithmic identifiers to identify the layers allows the forwarder to identify flows of aggregate layers and identify the ordering of the layers without such additional information.

Typically, redundant coding is used to prevent the necessity of retransmission, although it would be possible to provide additional an error control and retransmission channels using algorithmic identifiers. Error recovery coding can be published on algorithmic identifiers instead of being included with the primary information stream [Zha07]. This allows the error recovery information to be layered to meet the demands of subscribers with different error rates. The error recover information may also be available on different identifiers with different transmission delays (after the transmission of the original information). This allows subscribers to tune the latency of error information to which they are subscribed, and also to be able to subscribe to new recovery information (when their current information is not sufficient).

One potential problem with such a transport in the PSIRP architecture is the response of the network to subscription changes, if we assumed that all subscriptions would need to go through the global rendezvous system. Hence, although subscriptions should be able to be dropped locally fairly quickly, it may take longer to prune the multicast trees back to where they are no longer causing congestion. Similarly, new subscriptions (especially from the first subscriber on a network) may take some time to propagate through the (global) rendezvous system to the publisher so that the subscriber’s network can be added to the forwarding path. Thus, techniques such as using algorithmic identifiers to provide on-demand error correction for real-time media may not be workable on that level. However, one can imagine localized solutions with faster rendezvous and topology formation that would counter these drawbacks.

3 zFilter-based Forwarding

This section outlines the updates for our (zFilter-based) forwarding work, providing an overview of the zFilter mechanism that is currently in use in our architecture.

3.1 Background

In the early 1990s, the Multi-Protocol Label Switching (MPLS) protocol emerged to solve the traffic engineering problems caused by the explosive growth of Internet traffic which burdened the costly ATM switches and the slow-performing software-based IP routers. With MPLS, once a flow has been identified at the network edge and mapped onto a label, performance limiting network functions like the IP-address lookup are eliminated from the core network. With the advances in router processing power and line-speed hardware-based forwarding, in addition to traffic engineering, other attributes of MPLS such as fast protection, multi-protocol support, Virtual Private Networks (VPNs) and more recently support of Pseudo-Wires (PW) and Virtual Private LAN Services (VPLS), contribute to the compelling forces that drive MPLS adoption.

One of today's main network evolution challenges consists of scaling Ethernet to the WAN. Carrier Ethernet (CE) demands connection-oriented and carrier-centric services like Transport-MPLS or Provider Backbone Transport (PBT). Despite their technical differences, both underlying technologies adopt carrier-centric tunnels in the form of either label switching or stacked VLAN tags and MAC-in-MAC Ethernet extensions to enable explicit (source) routing with deterministic service parameters.

Routing experts have pointed out the critical growth of Internet routing tables, not only because of the implications on high-speed memory requirements but also because of the critical convergence time of the per forwarding node global-view-oriented IP routing paradigm. In the case of MPLS, it is easy to see [Far07] that MPLS-TE suffers from scalability limits when trying to extend its domain of applicability. The essential reasons for that are the maximum number of the 32-bit label switched paths (LSP) and extensive signalling related to LSP set-up.

3.2 zFilter Principles

To overcome some limitations of the existing link layer technologies, we have presented a scalable and fast *zFilter* forwarding mechanism in [Jok09]. zFilter forwarding is one alternative to implement forwarding in the pub/sub architecture. The zFilter mechanism uses in-packet *Bloom* filters (iBF) for forwarding decisions. In this kind of system, the topology system both constructs forwarding identifiers (FIds), used in a source-routing manner, and on demand installs new states at the forwarding nodes. The LIPSIN forwarding solution does not name nodes or interfaces. Instead, links are named, separately in each direction. In practise, each Link ID is an m-bits long string with k bits set to one, with k << m and m relatively large. This makes Link IDs statistically unique. For instance, with m=248 and k=5, we can derive around $m!/(m-k)! =\sim 9 * 10^{11}$ different Link IDs.

The network topology information is known by the topology layer. The division between the topology and the forwarding components is similar to those of the routing as a service proposal [Lak04] and the direct network control approaches, such as [Yan07]. Such a division can be achieved with a (distributed) topology service, similar to the Path Computation Entity (PCE) [Far06] in (G)MPLS.

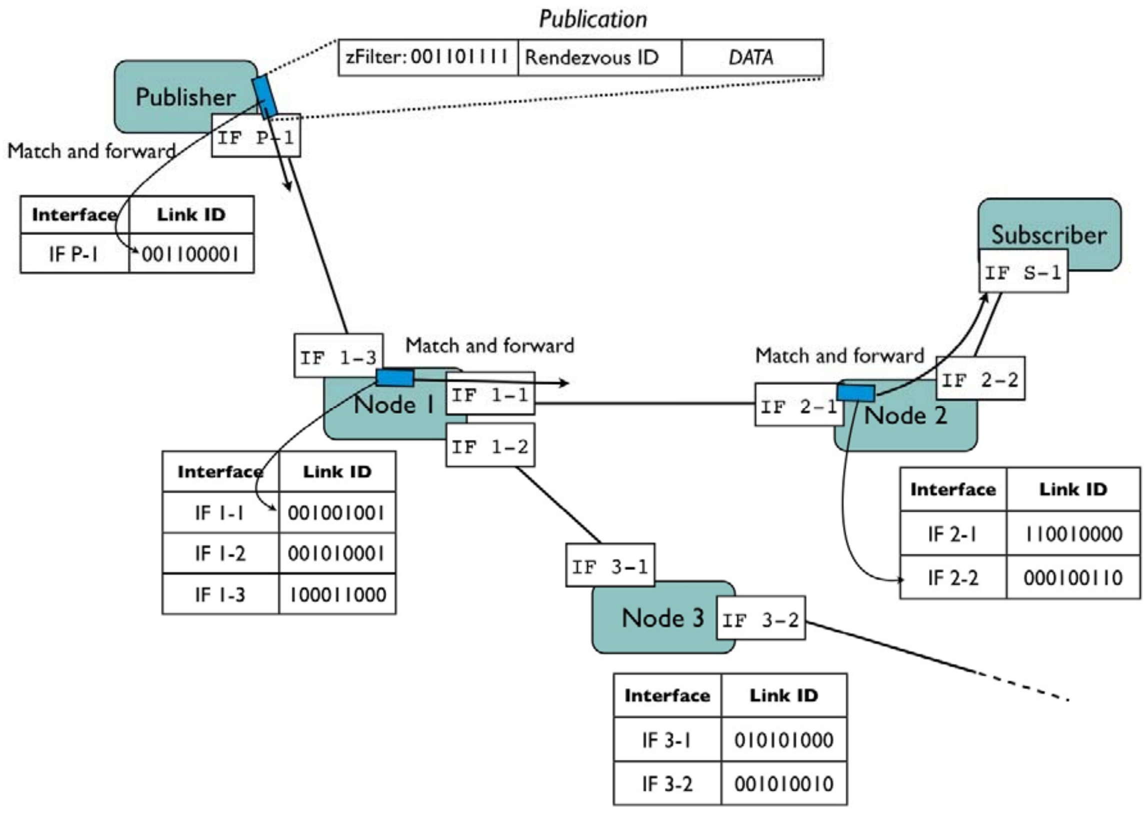
Using Link ID information, the topology manager can create a delivery tree to be used to deliver data from its current location, either from the publisher or a cache, to the set of subscribers. The set of Link IDs is combined using an OR function into the zFilter [Jok09]. There is no need to compress the Link IDs using hash functions (as in a Bloom filter) since the Link IDs are already very sparse in the number of bits set.

3.3 zFilter Forwarding Mechanism

The Link IDs are used at two distinct instances. First, to construct a zFilter for a given delivery tree T, the topology component takes a binary OR over the IDs of the links forming the tree. The resulting zFilter Z is then passed to the source, allowing it to send packets along the delivery tree using the zFilter as the forwarding identifier. With that, the forwarding fabric naturally supports multicast, while unicast is simply a special case with a single receiver.

Second, when a forwarding node receives a packet, it needs to determine where to forward the packet to. In the basic case, the forwarding tables are tiny: they contain only the outgoing Link IDs of the given node. The forwarding decision on each forwarding node is simply matching the outgoing Link IDs the node has with the zFilter in the packet header. This is illustrated in Figure 3.1. If a Link ID has been added, the Bloom filter verification returns a positive answer, and the packet is forwarded to that link. If there are multiple matches, the packet will be forwarded on multiple outgoing interfaces. This operation is done hop-by-hop on each of the forwarding nodes, and finally the packet is delivered to the subscribers. In other words, for each outgoing link *o*, the node checks if the zFilter *Z* contains 1s in those bit positions where the Link ID *L* does. If so, the node forwards the packet along that link; i.e., if *(Z AND L)* == *L*, then forward the packet over the link *o*. If the zFilter contains multiple outgoing Link IDs, then the packet is forwarded to each of them, resulting in multicast.

Using Bloom filters introduces of course the possibility of false positives; their probability rises as more links are included in the iBF. The *Link ID Tag* (LIT) mechanism, also described in [Jok09], provides control over the false positives by defining *d* different names for each outgoing link. Consequently, any given delivery tree can be described with *d* different iBFs, each of them having different bit patterns. This allows iBF selection based on different criteria, such as the smallest number of false positives.



Interface	Link ID
IF P-1	001100001

Interface	Link ID
IF 1-1	001001001
IF 1-2	001010001
IF 1-3	100011000

Interface	Link ID
IF 2-1	110010000
IF 2-2	000100110

Interface	Link ID
IF 3-1	010101000
IF 3-2	001010010

Figure 3.1 – zFilter forwarding principles

Bloom filter matching can also return false positives, i.e., the matching operation gives a positive result, even though the item was not actually added to the filter. In the zFilter forwarding solution, this means that the packet is sometimes forwarded on a link that was not part of the delivery tree. As long as the false positive ratio is small enough, we consider this harmless. Such false forwarding decisions can even be utilised to cache packets opportunistically.

While the presented method is scalable up to metropolitan area networks, for unicast and sparse multicast trees, the studies covers also the case of dense multicast trees. In that case, the number of false positives in the forwarding decisions grows higher, causing more unneeded traffic. To solve this problem, the zFilter concept introduces *virtual links*. A virtual link is a collection of single, physical links, and the virtual link is assigned a Link ID of its own. This requires a small amount of state to be created on the forwarding nodes on that virtual link. A virtual link ID can be used in a same way, as the normal Link IDs when creating the zFilter or forwarding the packet in the network.

4 Security Architecture

The PSIRP security architecture exploits publish/subscribe networking capabilities to build a security 'control plane' across the PSIRP network components, additional services and end users. In this manner, components subscribe to receive information such as security policies, public key certificates and revocation lists. Each component can establish trust relationships through which it receives such security information. This information can either be requests for control (such as security policies), or information that is required in order for it to perform some security control (such as certificates or notifications that other required steps have been conducted).

As an example, we can consider that network forwarding is controlled through Packet Level Authentication. However, packet forwarding may only take place if the publisher is authorised to attach to the local network, has permission to send to the relevant RId and SId, and has registered the content in advance with a notary service in order to police spam. The publisher may also be required to prove that they have used an authorised Inter-Domain Topology Formation Manager to construct a policy compliant route for the transmission. PLA does not need to check all of these credentials itself, since it can trust other components to have performed such authorisations in advance.

In more detail, it is expected that the publisher and subscriber will be required to follow a number of security related steps in order to achieve communication over the network. An illustration of such potential steps follows.

Network Attachment

Both the publisher and subscriber need to attach to a local network in order to begin to interact with other PSIRP network components. It is presumed that such local networks will require authentication against a valid payment account, or some other proof of ability to pay before authorising the connection. Free networks should still require some accountability from the publisher or subscriber. The account may be held with the local attachment network or may be accepted from other networks or account authorities with whom a trust relationship exists. In order to protect the authorisation mechanisms, the network attachment process may involve a precursory attachment puzzle to mitigate denial of service attacks on the authorisation systems.

Further accounts may be established once an initial account exists or may be bootstrapped by providing limited connection ability to payment (or other accountability) systems on initial connection. Accounts may also be set up outside of the network (via phone, email etc.) or carried on secure devices.

Subscription to RId, SId

Once attached, the subscriber can engage the rendezvous systems to subscribe to a particular RId and SId. The rendezvous systems are expected to perform a level of access control to protect themselves against rogue elements within the network. It is possible that the rendezvous network will authorise the subscriber or may trust that the authorisation performed by the attachment network is sufficient. The rendezvous system will also decide whether to accept the subscription. This can be based upon its own policies (coverage of certain SIds, load) or upon the policies of RId or SId owners. The concept of RId or SId ownership is flexible as subscribers, publishers or third parties can potentially desire ownership and control over the RId/SId.

The subscription will be shared across other elements of the rendezvous system which must decide whether to allow the propagation of the subscription. This will be based upon the trust of the other rendezvous service elements, along with their own local policies.

Publication to RId, SId

After attaching to the network, the potential publisher will interact with the rendezvous system to find information about subscribers. As for the subscriber, the rendezvous system will perform access control, along with checks that the publisher is authorised to send to the RId and SId in question.

The connection of the architecture components is still under discussion, so what happens next is given only as an example of how the topology formation and forwarding may operate. In the scenario below, we assume that the publisher controls the interaction.

After obtaining the attachment network information for all visible subscribers, the publisher will interact with an Inter Domain Topology Formation module to calculate a forwarding path. The ITF may again perform access control on the request, potentially even requiring that the publisher has a separate account with the ITF, or accepting that they will receive settlement from other parties (such as the attachment network). The ITF will determine one or more policy compliant routes (with policies coming from the forwarding providers, the ITF itself and any service requests from the publisher), and deliver the routes back to the publisher.

Optional Security Services

Any module in the PSIRP network architecture may decide that either the publisher or subscriber must receive prior authorisation or other security assertions from other network services. Examples illustrated in this report include an assertion from a notary service that the publisher is accountable for the content it is sending (either to prevent spam or copyright infringement), or an assertion from a reputation authority that the publisher is trusted to publish to the network (on the RId and SId). For example, the rendezvous system may enforce a policy (on behalf of the SId owner) that requires the publisher to prove that is has notarised the content before sending it (instead of actually restricting the identities of publishers that are allowed to use the scope).

Forwarding

Finally the publisher attempts to forward the information across the network towards the subscribers using the forwarding routes provided by the Inter-Domain Topology Formation. The attachment network and any subsequent forwarding network will decide to accept the packets using techniques such as Packet Level Authentication. The packet signature will not necessarily identify the publisher to each network, but will assert that the publisher is permitted by a trusted authority to send the packet. The trusted authority in this case may be the attachment network, or another party such as the rendezvous system or Inter-Domain Topology Formation provider. In any case, the presence of such an assertion may also imply that previous authorisation steps have taken place (such as the subscriber having a current subscription, the scope owner giving permission, the attachment network having a valid account etc.).

Exactly who trusts whom, and how such security services are operated remains part of the PSIRP research programme. The following sections of this report provide potential ideas and implementations for some of these security controls and services.

4.1 Forwarding-level Protection

Within the pub/sub networking model, there are three main avenues for Distributed Denial-of-Service (DDoS) attacks:

(1) An attacker may try to attack 'legitimately', i.e., through rendezvous (gaining one or more forwarding identifiers).

(2) An attacker may try to overload the rendezvous system with excess requests.

(3) The attacker may try to guess or construct a forwarding identifier so that it can overload the target, without receiving help from the rendezvous or topology components.

In this section, we focus on the last one, relying on existing work on capability-based techniques, over provisioning, and contractual relationships to solve the former two. Additionally, we exclude insider threats, leaving them for future work.

4.1.1 Forwarding Vulnerabilities

There appear to be a few vulnerabilities that our LIPSIN [Jok09] approach does not protect from. First, while a given zFilter works only from its source to its sink(s), the same zFilter can also be used for other traffic than the traffic that it was meant for. We call this a *zFilter replay attack*. Second, while the used encoding helps to hide the link identifiers, correlation between iBFs is still possible, creating a *computational attack*; see below. Third, while each zFilter is directly usable only by the source and any en-route nodes, if an attacker can determine another zFilter that passes through any of the en-route nodes, it can inject traffic to the delivery tree.

In the computational attack, an attacker collects valid, related zFilters and analyses them. Wherever the bit patterns are similar among a group of zFilters, it is likely that any reoccurring bits represent a partial graph common to those zFilters. Hence, knowledge over a large number of (source, sink(s), zFilter) triples may allow an attacker to create valid zFilters towards a target. By merging correlation pairs from multiple sites (e.g., using bots), DDoS attacks might well be possible. While the introduction of the LIT construction makes this attack computationally more expensive, especially when d is large, the attack appears to remain practical.

Z-formation [Est09] is an alternative mechanism to generate zFilters for forwarding paths. The dynamically formed and updated zfilters protect the subscribers from unwanted traffic without introducing much extra signalling between nodes in the system. In Z-formation, the zfilters works as capabilities as discussed in the following section.

4.1.2 zFilters Work as Capabilities

Network *capabilities*, as introduced by Anderson et al [And04], are architectural approaches that enable secure statements to be attached to packets, allowing routers to easily check if a packet was approved by the receiver. They are typically based on cryptographic approaches that enable routers to verify packets in a stateless way, though some statements, such as those related to maximum bandwidth, do require state. When capabilities are required, any prospective sender must first retrieve a suitable capability, either directly from the receiver (using explicit bandwidth reserved for that), out of band, or through a trusted third party [Wen06].

Using z-Formation [Est09], there is no need to have capabilities separate from forwarding identifiers; i.e., the capabilities operate as a forwarding identifiers and vice versa. Building upon LIPSIN [Jok09], a native multicast forwarding method based on in-packet Bloom filters, we introduced in [Est09] a DDoS resistant forwarding service. It prevents DDoS attacks effectively without losing any of the appealing scalability properties and resource requirements in the original proposal. We construct a system where having separate capabilities is unnecessary, as with our iBF-based forwarding identifiers it becomes computationally hard to extract path information for constructing new capabilities without insider help.

The interesting property with iBF is that it is hard to extract detailed path information from the iBF based forwarding identifier, which makes it possible to construct routing identifiers only for those source\sink combinations that are approved by both the source and the sink(s).

In particular, the Z-formation approach differs from existing label switching and source-routing-based systems in that (1) the forwarding header has a fixed size, independent of the number of hops. It is still important to notice that there is a practical upper limit of (around) 40 for the number of hops; see [Jok09] for a detailed analysis. (2) the system operates in a stateless fashion with very small forwarding tables, (3) the system is multicast friendly, and (4) the system has basic, built-in security features.

In a way, our source routing proposal is in line with the needs and evolution of carrier networks to provide flexible connection-oriented services (e.g., T-MPLS, PBT, VPLS) and the renewed interest in flow-based networking with separated control planes [Yan07]. At the cost of larger packet headers, the iBF-based forwarding approach has the attributes required to scale up to Internet-like environments. Towards this goal, network-assisted DDoS protection needs to be considered *ab initio*.

4.1.3 z-Formation Principles

Instead of maintaining a fixed forwarding table that lists a number of Link IDs (or LITs) for each outgoing interface, z-Formation uses dynamically computed link names. For each incoming packet, a function Z computes the corresponding LITs using (i) an in-packet rendezvous ID (RId), (ii) a periodically changing secret K , (iii) the incoming and outgoing interface in- dices (In, Out), and (iv) the Link ID Tag index d. This produces a dynamically computed Link ID Tag (LIT), as depicted in Figure 4.1. As in LIPSIN [Jok09], each LIT O = Z (RId, K (t), In, Out, d) is a Bloom mask of size m. As the zFilter is now constructed using these dynamic LITs instead of static LITs, the resulting zFilter becomes additionally bound to the RID, a specific time period, and the input port. Specifically, having the RId as an input parameter ties the given zFilter to only those packets carrying the specified RId, which, for example, makes reactive filtering an easier task, as it can be done based on the RId.

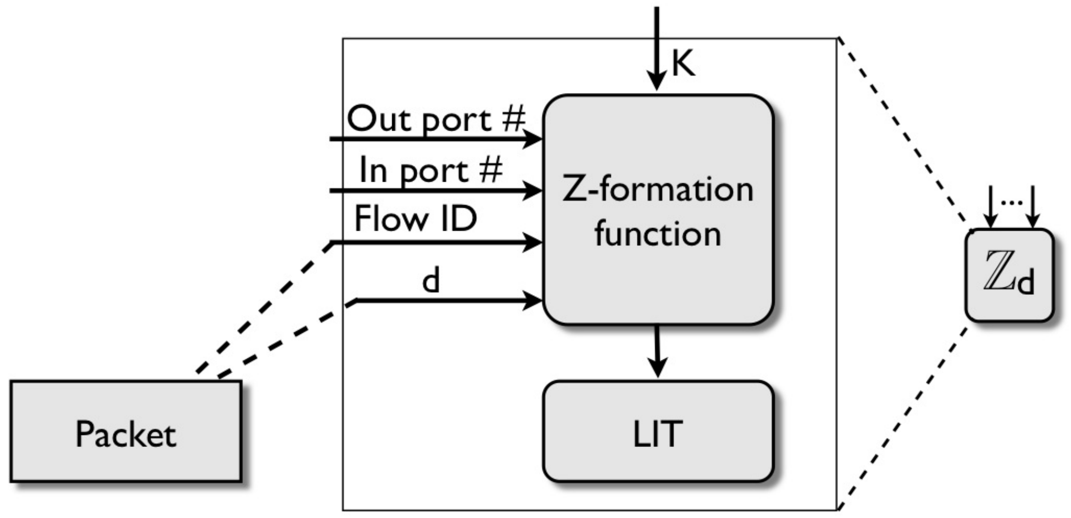


Figure 4.1 – z-Formation dynamically creates LITs

To construct the time-bound shared secrets K, each forwarding node *i* shares a master key K_i with the topology manager. For a time period *t*, $K_i(t) = F(K_i, t)$, where *F* is a cryptographically secure pseudo-random function and *t* denotes the period. For example, *t* may be a seeded counter or wall time clock at a coarse enough granularity. In either case, the forwarding nodes and the topology manager need to have loosely synchronized clocks. The topology manager always uses the newest valid value of *t*; the forwarding node also accepts *j* (one or a few previous) values. In this way, if *t* is advanced every Δt seconds, even if $K_i(t)$ is compromised for a specific *t*, the attack is limited to the single forwarding node using the key and to the maximum time of $j\,\Delta t$.

Finally, as Z takes in both the outgoing and incoming interface indices as inputs, any given zFilter is tightly bound to the corresponding forwarding path or delivery tree. That is, this feature blocks the injection attack, preventing off-path attackers from sending data towards a delivery tree even if they know both the RId and the zFilter. Additionally, including the incoming interface index as an input-parameter allows us to introduce virtual interfaces within forwarding nodes, thereby enabling on-path services.

4.1.4 Topology and z-Formation

Typically, the topology manager receives the interface information from the forwarding nodes. The RId information is received from the Rendezvous system when a subscription matches a publication. The topology manager defines a seed value for each forwarding node used for key generation and communicates the value to each forwarding node. The system is more complex than the basic zfilter solution, but on the other hand, it can achieve smaller forwarding tables and better DDoS protection. The minimum state that needs to be stored in the routers is the Z-function. The principles of Z-formation are illustrated in Figure 4.2.

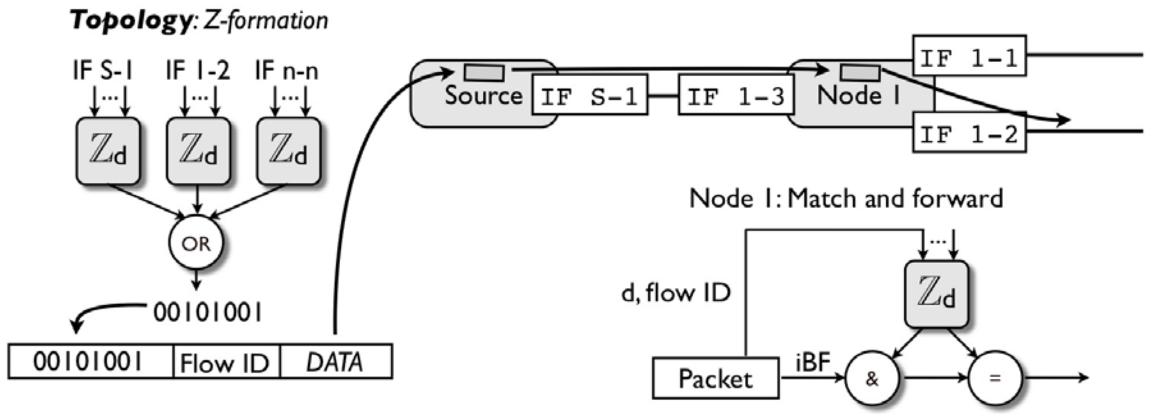


Figure 4.2 – z-Formation principles

When a data packet arrives at a forwarding node, it will pick the d-value from the zFilter, the RId from the packet, the incoming interface identifier, as well as *K*. The router calculates the Link ID for each of its outgoing interfaces using this information together with the outgoing interface identifier. The resulting Link ID is matched to the forwarding zFilter in the packet, and in case of a positive result, the packet is forwarded on the interface. While forming the Link ID, it is also possible to generate a new d-value (d'), which replaces the original d in the forwarded packet's zFilter. This is utilized to avoid routing loops in the network.

In a typical implementation, the Z-formation calculation can be done in parallel on each of the interfaces, thus optimizing the required time for the operation. It is possible to implement the function on available FPGA hardware, with the goal of not causing additional delay at the forwarding node.

4.1.5 Binding between Rendezvous and z-Formation

The Z-formation does not directly protect from attackers that replace the payload with some spoofed content. Basically, an attacker can record and copy a valid zFilter into a new packet, and use the original RId that was used to form the zFilter, thus creating a valid header. It is possible to mitigate the attack by allowing the sender to sign the packet payload with the public-key that is part of the RId. The forwarding nodes can verify the signature in the packet utilizing the public key part of the RId. This verification does not have to be done for all the packets, but only for a fraction of the packets. In case the verification fails, the forwarding node can inform the topology manager about the event and the corresponding zFilter that was carried in the packet. The topology manager can request earlier forwarding nodes in the routing path to start using a stricter signature verification policy or identify the origin of the sender of spoofed messages

To protect the subscribers from unwanted traffic, the topology manager must not create end-to-end forwarding (path) identifiers without explicit subscriptions. One approach is that the rendezvous system informs the topology system about the authorized match between a subscriber and a publisher. Based on this information, the topology fabric creates a valid zFilter for the publisher to send data to the subscriber. After the lifetime of the zFilter expires, the publisher must request a new zFilter from the topology manager in order to send content to the subscriber. Renewing the zFilter requires an existing subscription. If the subscriber unsubscribes from a rendezvous identifier, the topology system will not renew the zFilter for the publisher. The communication between the rendezvous and the topology system is for further study as it depends very much on the underlying business models. In most cases, the rendezvous and topology belong to different administrative domains.

4.1.6 Attack Mitigation

Even if we assume topology knowledge by the attackers, having a shared secret among the forwarding nodes unknown to the attackers reduces the best attack strategy to a brute force attack consisting of generating random labels and expecting that at least one of them reaches the target(s) (see [Est09]). In practice, a fast re-keying frequency (i) protects very short paths and (ii) limits the duration of DDoS attacks based on the misuse of legitimate zFilters. A zFilter expiration time in the order of a few dozens seconds is long enough to complete typical transactional traffic without requiring zFilter renewal. It is also good to notice that the DDoS protected forwarding plane only complements additional security measures at the end node at a higher level of the stack, similar to end-host firewall implementations where only solicited (subscribed) data flows are allowed and processed.

By virtue of the time-based re-keying mechanism, a forged path lasts only for $j\Delta t$ in the worst case. After that, a malicious node would need to re-initiate the attack process. As the most efficient attacks we are aware of require excessive probing by the attacker, an attack can be detected early by the sudden increase in incomplete paths (caused legitimately by false positives) caused by the falsely labelled packets injected by the attacking node(s). Hence, a blacklist mechanism can be used to block or shape down any suspicious traffic.

As our networking model assumes the existence of an in-packet RId, each attack needs to be tailored for a specific RId. This does not only limit the scope of an attack but also facilitates any blacklisting mechanisms. It becomes simple to block or restrict by invalid (RId) signature, e.g., if incomplete routes beyond a threshold are detected.

4.2 Authorization in the Rendezvous System

From the security point of view, Rendezvous acts as a capability distribution centre [Wen06]. Topmost in Figure 4.3, the Rendezvous matches publishers and subscribers according to rendezvous identifiers (RIds) within a given scope. A scope defines an information network that is labelled with a scope identifier (SId). That is, a scope determines a part of the information network. A scope may be topological, such as link local, intra-domain, and inter-domain, or content-centric, such as the Library of Congress. Scopes can also be used to map higher-level concepts, like social networks.

The rendezvous-level authorization mechanism protects the subscribers from unwanted traffic. Only authorized publishers are allowed to publish content in scopes. The different components and phases of the authorization procedure are explained in the following sections.

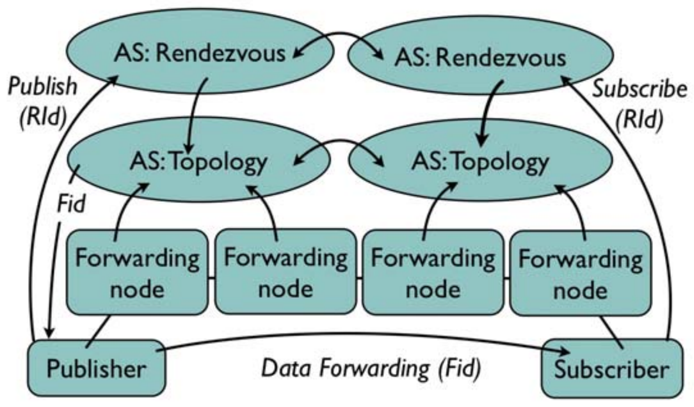


Figure 4.3 – RVS, topology, and forwarding

4.2.1 Scope Ownership

Each scope has an “owner”. This means that the entity which has access to the corresponding private-key "owns" the SId. The scope “owner” can also be called *Trusted-Third-Party* (TTP). Publishers and optionally subscribers must be authorized by the TTP in order to send or receive content within a scope. Such authorization is expressed with authorization certificates. For the purposes of this report, we provide here a set of definitions that are used in the later sections.

- *PK.x* denotes a public-key-pair that is associated with x. For example, *PK.SId* defines a public-key-pair that is associated with SId. All processes that have access to the public-key-pair are “owners” of the Sid.
- $PK_{pub}.x$ denotes the public part of the key pair associated with *x*.
- $Id = PK_{pub}.id:L$ denotes a DONA-like identifier, i.e., concatenated with public-key and label. The label *L* can have any kind of structure.
- *CER.subject* denotes an authorization certificate that consists of a set of attributes:
    - *Rights* defines the actions/resources that the subject is allowed perform, e.g., the right to register provision or interest in a scope
    - *Subject* denotes who is allowed to perform the action. This can simply be the public-key of the subject.
    - *Issuer*, states who issued the certificate in order to form chains of trust, e.g., a scope “owner”, authorizes the subject to act in a scope. The scope owner is trusted to issue such certificates for scopes that it owns.
- *SId.security* denotes the security sub-scope, which is explained in later sections.
- '|' denotes a logical OR operation, while *Hash* denotes a one-way hashing function.

The system protects subscribers from unwanted traffic at the rendezvous and forwarding levels, as the subscribers do not receive any traffic that they have not explicitly subscribed to. In practice, rendezvous nodes are responsible for verifying that a public key associated to a RId is authorized by a scope “owner”. This is described in the following sections.

4.2.2 Node-Internal Communication

Within a node, application processes and other entities can communicate with each other through a *blackboard*, which bears a resemblance to a Linda-like systems and tuple spaces [Gel82]. In this way, node-local interactions can be carried out with the same pub/sub paradigm that is used in the network. Also integral parts of the pub/sub system, i.e., functions implemented within the *component wheel* (see Figure 4.4), can utilize this host-internal rendezvous structure.

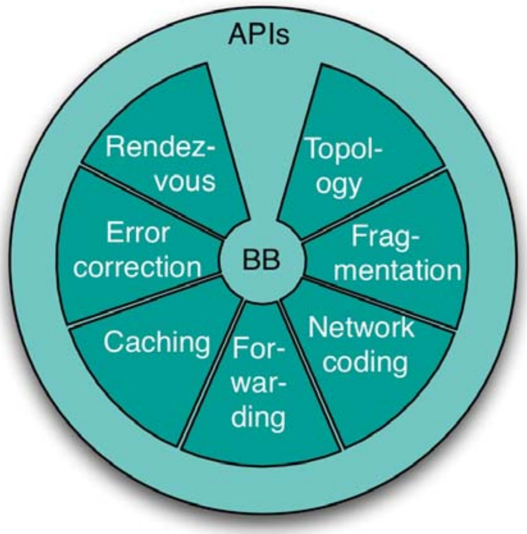


Figure 4.4 – Blackboard (BB) in the middle of the component wheel

4.2.3 Main Scope and Security Sub-scope

As mentioned earlier, rendezvous uses scopes to match content publishers and subscribers. However, to protect the scope from unwanted traffic, the scope “owner” authorizes publishers and subscribers to register events in its scope. The publishers register provision and subscribers register interest for content. To authorize publishers (and optionally subscribers) to access the main scope, we have defined a so-called *security sub-scope*.

The security sub-scope is decoupled from the main scope as a control-channel for delivering authorization certificates between the scope “owner” and other actors. The other actors consist of publishers, subscribers and forwarding nodes. It is worth noting that the scope “owner” could be the publisher or subscriber directly, in which case no ‘third’ party would be involved. In other words, the scope “owner” must be seen as a logical component that potentially maps onto existing actors like publisher or subscriber. To overcome the chicken-and-egg problem, all actors are allowed to subscribe to events in the security sub-channel but only the scope “owner” is authorized to publish certificates in the security sub-scope. Once the publishers and subscribers have received their certificates, they can start registering events in the main scope.

One essential reason to have a separate security sub-channel is to minimize events on the original traffic channel. . It also helps to preserve the PSIRP principle that different information can be separately addressed. The forwarding nodes may subscribe to listen to security related events on the security sub-channel but they are not interested in receiving other kinds of traffic. One target is to minimize the additional traffic overhead on the already heavily loaded forwarding nodes. The security sub-channel is mainly used to publish new and revoke existing certificates. The PSIRP versioning model can be directly used to publish updated versions of certificates that revoke previous versions.

4.2.4 Creating Scopes

To protect the system from a single point-of-failure and support different kinds of trust models, we have de-coupled the rendezvous mechanism from the authorization mechanism. This is illustrated in Figure 4.5. The publisher and scope “owner” may establish a trust-relationship directly or via “third” parties. The scope “owner” makes the authorization decision based on the earlier established trust-relationship.

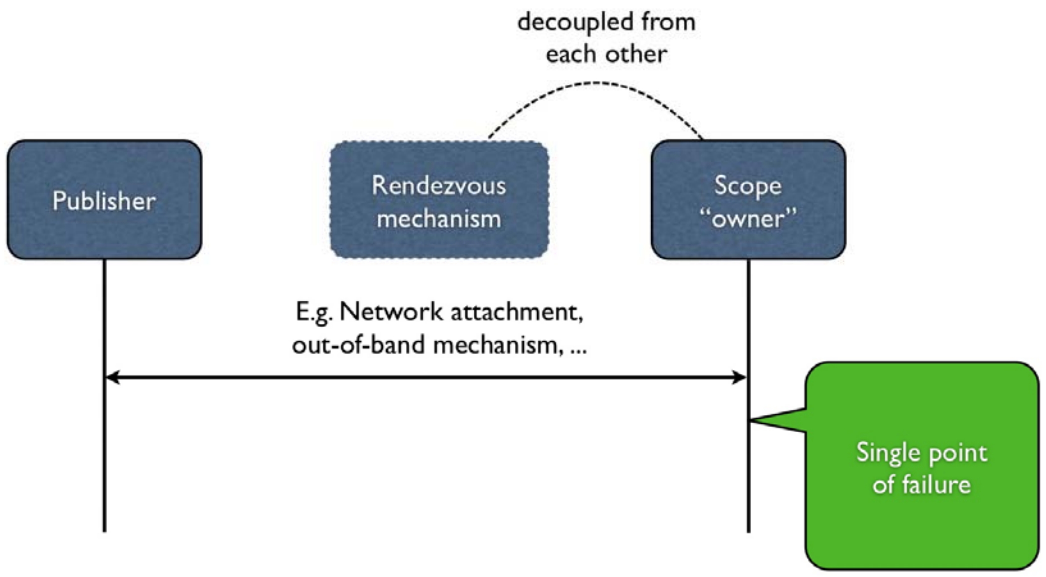

decoupled from each other
Publisher
Rendezvous mechanism
Scope “owner”
E.g. Network attachment, out-of-band mechanism, ...
Single point of failure

Figure 4.5 – Decoupling rendezvous and authorization mechanisms from each other

The authorization request can be integrated, e.g., with the network attachment exchange or delivered out-of-band. Here, we do not analyze the different mechanisms to deliver authorization requests to the scope “owner”. We only assume that the publisher is able to contact the scope “owner” and thus bootstrap the communication in the scope. Moreover, the trust models between publishers, subscribers and scope “owners” are out of scope for this discussion.

More precisely, the authorization request can be originated from a publisher or a subscriber. From the authorization point of view, the cases are similar because at the forwarding layer the subscription message is delivered as a special publication. Therefore, the scope “owner” separates the publishers and subscribers from each other by approving different kinds of publishing rights. Publishers are authorized to register provision and subscribers to register interest for content in the scope. Basically, we have different types of publications that are either sent by publisher or subscriber. The type of the publication message and the associated SId and RId define whether the rendezvous system drops or processes the packet after verifying the carried signature.

The basic assumption is that any node in the system can create a public key-pair 'PK.SId' and a label for a new scope identifier 'SId'. Initially, the owner of the public key pair defines a label for the main scope, e.g., 'SId = PK_{pub}.SId : L1', and generates the security sub-scope. In this example we assume that the security sub-scope is related algorithmically to the original SId by enabling a security flag in a reserved part of the label, e.g., 'SId.security = PK_{pub}.SId : (L1 | Sec-Mask)'. The security sub-scope is labelled by ORing the original SId with a so-called *security mask*. Each SId contains one byte in the label part that is reserved for control bits. The security mask 'Sec-mask' sets one bit on in the label part resulting in a separate security sub-scope identifier 'SId.security'. Other more complicated schemes are possible that use more advanced code-points for multiple options, or use other functions to algorithmically generate the security identifier (as explained earlier in the Algorithmic Identifiers section).

Once the scope “owner” has defined both SId and SId.security, it creates a self-signed certificate 'CER.self' for itself. The certificate authorizes the scope “owner” to publish in its own scope. The certificate is verified by the rendezvous system when the scope "owner" registers provision in the scope. The most essential parts of the certificate look like the following:

- RId of CER.self = PK_{pub}.SId : Hash(SId : PK.SId)
- Rights = Register provision in SId
- Subject = PK_{pub}
- Issuer = PK_{pub}

It is important to note that the certificate is a publication in itself. The scope "owner" defines an RId for the certificate and publishes it in the rendezvous system. The label part of the certificate is a hash of the SId and PK.SId values. The reasoning for this kind of labelling is explained later.

4.2.5 Authorizing Publishers

Initially, the publisher generates a public key pair (PK.publisher) for itself. The publisher selects a scope (e.g., SId = PK_{pub}.SId : L1) and requests authorization from the scope owner as described earlier. The authorization may take place well before the publisher actually publishes any content on the scope.

Once the scope "owner" receives an authorization request from the publisher, it verifies the publisher's credentials. Depending on the trust model the credentials can be based, e.g., on access control list, delegated credentials, opportunistic authentication or reputation in the rendezvous system. If the publisher is allowed to access the scope, the scope "owner" creates an authorization certificate for the publisher to register provisioning in the scope. The certificate is illustrated in the following:

- RId of CER.pub = PK_{pub}.SId : Hash(SId : PK_{pub}.publisher)
- Rights = Register provision in SId
- Subject = PK_{pub}.publisher
- Issuer = PK_{pub}.SId

The label part of the certificate's RId is hash of SId and PK_{pub}.publisher. In this way, the subscriber and forwarding nodes in the network are able to subscribe to the certificate by knowing the Sid and the publisher's public key. The scope "owner" registers the provisioning for the certificate in the rendezvous system under the security sub-scope (Figure 4.6). Later on, the scope "owner" does not need to be on-line and does not constitute a single point of failure in the system.

From the subscriber's point of view, authorization happens similar to the publisher case. It is important to note that while "subscriber" is a role of an entity that is willing to receive information from the network, all the messages sent by the subscriber are basically publications from the network's perspective. Even the interest for receiving content is sent to the rendezvous system as a publication.

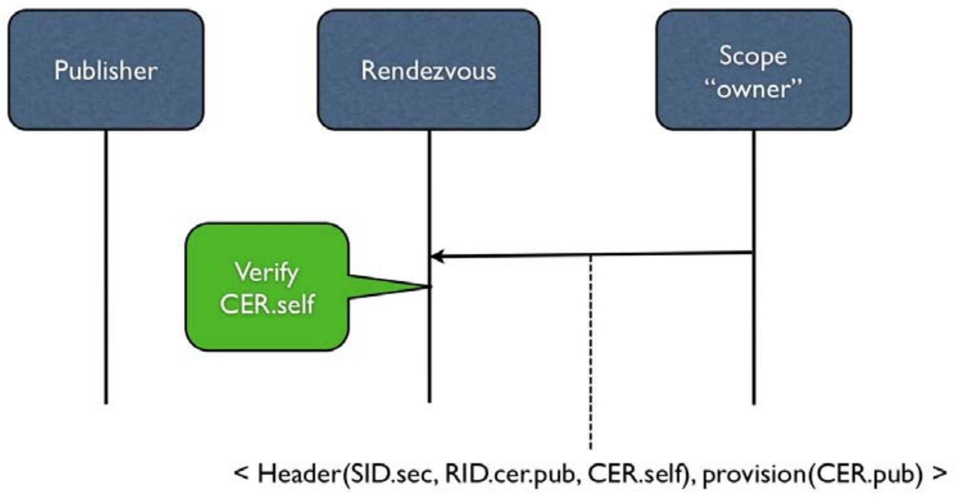

Publisher
Rendezvous
Scope "owner"
Verify CER.self
< Header(SID.sec, RID.cer.pub, CER.self), provision(CER.pub) >

Figure 4.6 – Scope "owner" publishes certificate in the security sub-scope

The publisher subscribes to the approved certificate in the security sub-scope. It is able to generate the SId.security and RId.CER.publisher based on the knowledge of PK_{pub}.publisher and SId values. The rendezvous system returns the certificate to the publisher (Figure 4.7).

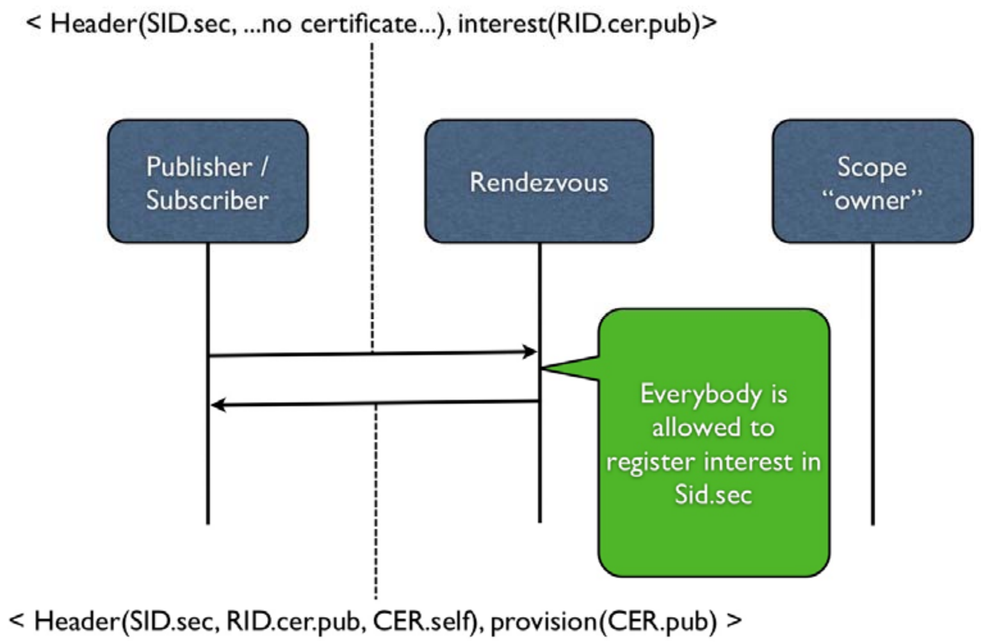

< Header(SID.sec, ...no certificate...), interest(RID.cer.pub)>
Publisher / Subscriber
Rendezvous
Scope "owner"
Everybody is allowed to register interest in Sid.sec
< Header(SID.sec, RID.cer.pub, CER.self), provision(CER.pub) >

Figure 4.7 – Publisher subscribes to certificate that was approved by the scope "owner"

In some cases, RIds should have a very long lifetime, which is problematic since private keys may be lost or leaked. One alternative is deriving the RId from some authority's cryptographic identity (PK.authority). The authority would authorize the publisher (PK.publisher) to utilize the RId through the certificate mechanism: (certificate (Issuer PK_{pub}.authority) (Target PK_{pub}.publisher) (Rights to publish to RId)). If the publisher loses his private key, he can generate a new one and request a new certificate from the authority.

4.2.6 In-packet Certificates

The earlier sections discussed the decoupling of rendezvous and authorization mechanisms. The certificates were published in the rendezvous system as individual publications. With the ECC cryptography the public keys and signatures used are short enough to be also carried in the packet headers.

Currently, we have three alternative design choices to include the authorization certificates into the packet headers. The first alternative is to include the publisher's certificate, with the right to publisher to a certain scope, in each payload packet. The other alternative is to include the in-packet certificate only in meta-data packets that carry information of the RIds of the data chunks (as described in Section 2). The third alternative is to present the authorization certificate to the topology layer and get a valid forwarding identifier (zFilter) that matches the certificate. As a result, the packet header will contain only the signature of the publisher not the full authorization certificate. The design of the in-packet certificates is for further study and it is strongly coupled with the Packet Level Authentication (PLA) mechanism.

4.2.7 Registering Provisioning and Interest for Content

The publisher creates some content and defines a rendezvous identifier for the data (RId.data = PK_{pub}.pub:L4). The publisher registers provisioning for the RId.data in the scope (SId). The registered meta-data packet contains the certificate approved by the scope “owner”. The rendezvous nodes verify the signature and time-stamp of the certificate before registering the provisioning. This is illustrated in the following Figure 4.8.

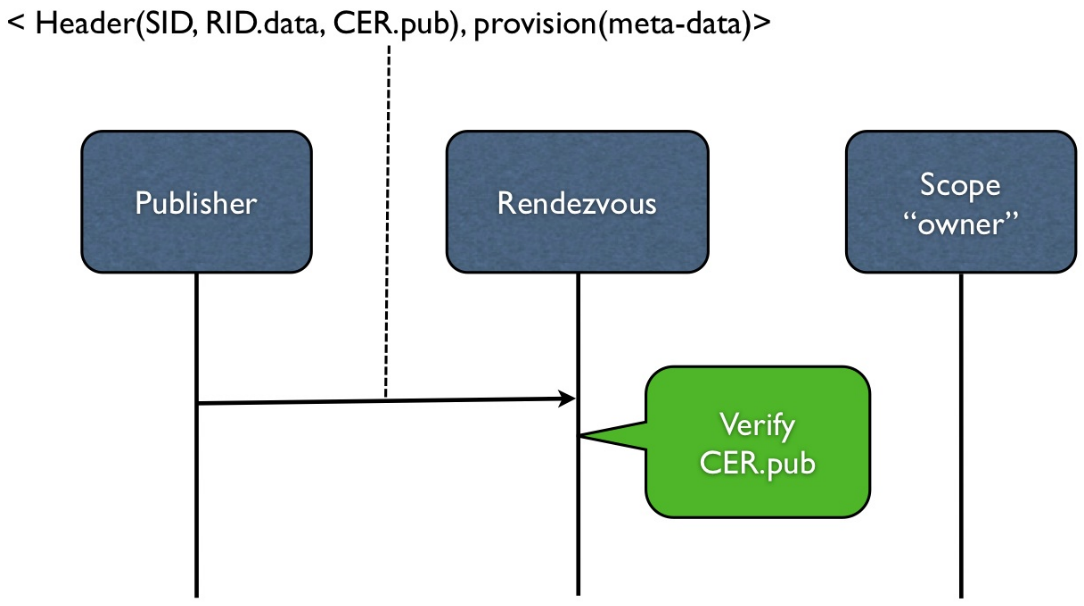


Figure 4.8 – Registering provisioning for data

From the subscriber’s point of view, the procedure is almost similar to publisher's case as illustrated in Figure 4.9 below. The main difference is that the subscriber's certificate authorizes the subscriber to only register interest in the scope, not provisioning.

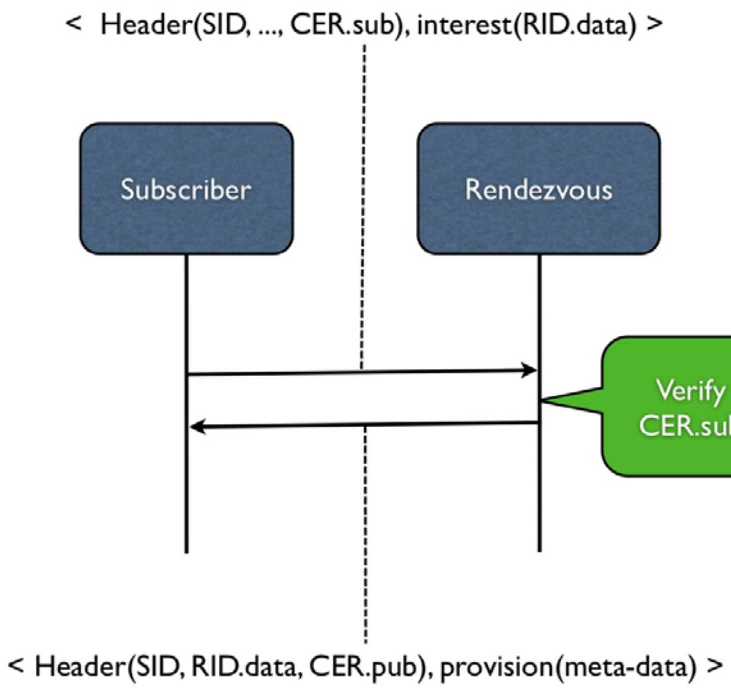

< Header(SID, ..., CER.sub), interest(RID.data) >
Subscriber
Rendezvous
Scope "owner"
Verify CER.sub
< Header(SID, RID.data, CER.pub), provision(meta-data) >

Figure 4.9 – Registering interest for data

4.2.8 Summary of Security Properties

The security properties of the system can be summarised as follows. Each scope has an owner, and the corresponding SId is derived from the owner's public key. The scope owner authorizes publishers to publish in the scope using a standard certificate mechanism. These requests and certificates are exchanged through the separate security sub-scope. In order to allow flexibility and long lived identities, the system also supports delegation of rights for both the scope ownership and publishing rights.

The main advantage of such a system is that any node in the network can verify whether the scope owner and publisher have the necessary rights, i.e., does the publisher X have a right to publish in scope Y using RId Z? Therefore, forged publication messages can be stopped before they even reach the destination. This is possible since the RId and SId are derived from cryptographic identities.

4.3 Security in Inter-connected Rendezvous Systems

Individual rendezvous networks interconnect with each other by using a hierarchical DHT that distributes the global state in a scalable way to multiple nodes with minimal duplication. We plan to use the Canon-version of Chord, Crescendo [Gan04], as the basis of our implementation. Every object, that can be located globally in a PSIRP inter-network, has a pointer to it stored in the rendezvous interconnect system. We assume that the DHT would eventually consist of thousands of nodes spanning over multiple administrative domains, which makes it impossible to assume that every node will function correctly at all times.

The rendezvous interconnect system is also a central component in the PSIRP network stack and is required in most operations, which places heavy requirements on the robustness and "inertia" of the architecture. The high-level security goals for the rendezvous interconnect are *availability*, *utility*, *integrity*, *confidentiality* (of publications, subscriptions, and scopes), *accountability*, and *fairness* of resource allocation. It is assumed that subscribers and publishers trust the scope they are using for communication. This means that false RId advertisements inside a given scope are not a problem of the network. The rendezvous system is only responsible for correctly resolving the home rendezvous network of scopes and returning the result to the subscriber with a high probability. We also assume that the hierarchical structure of Crescendo will loosely follow some administrative boundaries and that subhierarchies function as self-contained systems with possibly higher trust between their members.

In this section, we analyze the main threats against the interconnect architecture and present the initial defence mechanisms planned, but as we are still working out the details of the rendezvous system, there are some open problems while the work continues. A good survey of state-of-the-art techniques for DHT security can be found in [Urd2009].

4.3.1 Data integrity and Confidentiality

The rendezvous system stores pointers to the locations of the home rendezvous networks for every scope advertisement stored in the system. The integrity of these advertisements is guaranteed by a certificate issued by the scope owner authorizing the rendezvous network to host the scope. This certificate is stored together with the pointer/advertisement and can be checked by every node in the DHT without external help.

Confidentiality of publications and their labels can be achieved by a rendezvous operation that always reaches the home rendezvous network of the scope trusted by the scope owner to perform the access control for the scope. In this case, the rendezvous system also returns a secret key used for encrypting the labels and content that are used in topology and forwarding layers,.The secret key is encrypted using the public key of the subscriber while it travels through the rendezvous interconnect.

4.3.2 Approaches to Rendezvous Interconnect Formation

Three approaches were considered for the admission of new nodes to the DHT structure:

(1) A trusted central control for each subhierarchy that creates identities for every node under it and binds them to the real-world identities of the participating organizations

   a. The interconnect evolves slowly and is based on contracts between operators. There is no need for dynamic node id generation and the public keys of the trusted central entities can be kept offline.

   b. Still, a portion of nodes must be assumed to be malicious in this scenario too, as different organizations have different goals and levels of security internally.

(2) A peer-to-peer based system with an automatic reputation algorithm

   a. There is no central control of any kind and the system evolves freely.

(3) A hybrid solution with distributed creation of node identities but still contract-based with humans in the loop

   a. A centralized control element can be a single point of failure for security and has policy issues.

We eventually ended up with the hybrid approach as the rendezvous interconnect is an essential component of the infrastructure and a peer-to-peer system could be rendered completely unusable by a successful attack with no way of restoring it. For example, the Sybil attack [Dou2002, Cas2002c] can be impossible to protect against in a peer-to-peer network without some further assumptions like trusted topology information from underlying layers. The purpose of the DHT formation is to assign secure identities for every node that are then used to choose their positions in the Chord ring and determine their connectivity with the other nodes.

4.3.3 Attack Scenarios

The possible attacks can be categorized based on their source:

- Attack from the inside by
  - a malicious operator building part of the network that could try to change the topology or affect the DHT formation by adding or removing nodes to the rendezvous network interconnect or
  - a compromised node or link that could be used to break the inter-working between nodes by interfering in the communication between nodes by adding, removing, changing or eavesdropping messages.
- Attack from the outside by an end-user computer that might attack the system by using it in a legitimate way through its API or sending packets from local interface (for example, a botnet-based DDoS attack or trying to gain unfair amount of resources).
- A coordinated attack by multiple parties that could cause Byzantine failure modes for example in information warfare.

The attacks could try to achieve

- an unauthorized addition/deletion of a publication or a scope
- an unauthorized modification of RId or SId metadata
- gaining knowledge of the existence of a certain publication, scope, or a subscription
- activity monitoring of a certain publication, scope or subscription
- an unauthorized advertisement of an existing publication or scope
- utilizing an unfair amount of resources of the rendezvous interconnect
- preventing the normal operation of the rendezvous system (for certain publishers, subscribers, scopes, or rendezvous networks).

In the following list, we have enumerated the most prominent known methods of attack against the rendezvous interconnect, the minimal requirements for each attack, a short explanation how the attack might be implemented, and finally a list of planned countermeasures against them.

1. DDoS against a particular interconnect node

- **Requirements:** a leaked Chord hash function and a botnet
- **Method:** Using a hash function, the botnet nodes generate lots of SIds and publish the ids falling into a certain range in the interconnect, thus causing scope advertisement state to accumulate in a certain rendezvous node. Alternatively, the botnet can create a surge of subscriptions hashed to the same node.
- **Defense:**
  - Interconnect has only soft state, that is, publishers are required to refresh their scopes periodically and rendezvous nodes can purge old scopes or store them in cheaper disk storage. However, this does not offer much protection against instantaneous attacks.
  - Chord address space can be load balanced based on node capabilities.
  - The DHT can collect usage statistics about its subhierarchies and end-customers and, if the distribution of subscriptions/publications is too improbable, the suspicious subhierarchy could be slowed down.
  - Each scope advertisement can be replicated in multiple nodes in the interconnect using "salt values" added to the SId before hashing it. This way any scope does not depend on a single node in the interconnect. Attacks may hit all replicated nodes, but does provide resilience in the case of single node failure.

2. *DDoS against a particular rendezvous network*
   - **Requirements:** botnet
   - **Method:** Botnet computers flood (bogus) requests towards RIds hosted by the same rendezvous network.
   - **Defense:**
     - Public publications/scopes are affected less as the results for the rendezvous operations for such publications can be distributed and cached/multicasted in the rendezvous network. Only access control and unpopular publications require reaching the home rendezvous network of the scope.
     - Scopes can be replicated in multiple rendezvous networks and requests load balanced among them.
     - Local rendezvous networks could slow down hosts whose requests to a particular rendezvous network are denied. Again, usage statistics can be used for accountability.
3. *Sybil attack [Dou2002, Cas2002c]*
   - **Requirements:** Because the rendezvous system consists of only few thousands of nodes, the Sybil attack can be mounted even with a small number of hostile nodes.
   - **Method:** The malicious party creates a large number of pseudonymous nodes, using them to gain a disproportionately large influence in the DHT.
   - **Defense:** Authentication of nodes based on public key identities or topological information.
4. *Poisoned data*
   - **Requirements:** a malicious DHT or an external node
   - **Method:** Sending extra responses to DHT nodes or modifying the result of a subscription operation in an interconnect node.
   - **Defense:** All communication between nodes is encrypted and the payload is self-certifying.
5. *A subhierarchy or an end-host using an unfair share of resources*
   - **Requirements:** a malicious host or a DHT node(s)
   - **Method:** A subhierarchy or an end-host stores more scope pointers or originates more subscriptions than what is allocated to it.
   - **Defense:** Usage statistics are collected and resources are shared at each level of the Canon hierarchy according to the contracts between the operators of the interconnect. This gives the subhierarchies the incentive to recursively manage their share of resources to their customers. Because the statistics are collected simultaneously by multiple parties, incorrect data can be noticed.
6. *A subhierarchy providing less resources than negotiated*
   - **Requirements:** a malicious DHT node or nodes
   - **Method:** A DHT node refuses to store scope pointers or serve subscriptions as negotiated between the operators of the DHT.
   - **Defense:** Reverse statistics of provided resources could be collected by other nodes of the DHT and aggregated together. Because this process is replicated to multiple parties, incorrect data can be noticed.

7. *Selectively refusing to serve publishers and subscribers*
   - **Requirements:** a malicious DHT node or nodes
   - **Method:** An interconnect node can drop subscription messages or results, or selectively refuse to store valid scope pointer advertisements based on the publisher/subscriber location or the used SId.
   - **Defense:** Each scope advertisement can be replicated in multiple nodes in the interconnect using "salt values" added to the SId before hashing it. This way, any scope does not depend on a single node in the interconnect. The number of used replicas can depend on the service. If a certain path in the interconnect systematically fails to serve a given scope or a subscriber, the connection can be tested by a trusted third party and the malicious node removed from the system.
8. *Malicious nodes interfering with the DHT topology formation algorithm*
   - **Requirements:** a malicious DHT node or nodes
   - **Method:** A compromised or hostile DHT node or nodes send false data to their neighbors in the Crescendo ring trying to break the Chord invariants.
   - **Defense:** Nodes are assigned identities based on public keys and certified by trusted third parties or derived from underlying topology information. These identities are directly used to form the Chord ring address of the node guaranteeing that the topology for the non-compromised nodes is correct.
9. *Spoofing of a source address of a subscription*
   - **Requirements:** a botnet
   - **Method:** A subscriber tries to redirect the responses to her subscription operations towards another destination causing a DDoS attack against it.
   - **Defense:** Recursive routing in the DHT prevents changing the return address of the subscription operations.
10. *Gaining knowledge of an existence of a SId or a RId*
    - **Requirements:** a malicious subscriber or a DHT node
    - **Method:** Subscriber tries to probe the existence of random SIds or an interconnect node leaks information about stored scope pointers.
    - **Defense:** Encryption of communication between adjacent nodes in the interconnect prevents external eavesdropping. SIds used in the scope advertisements can only be the one-way hash of the original P:L identifier. This way, the SIds are just meaningless bit sequences from a very large address space, and the attacker must know both P and L beforehand. If the scope owner wants to keep the existence of given RIds completely secret, their labels can be encrypted on the topology and forwarding layers and subscription operations access controlled in the home rendezvous network of the scope.
11. *Gaining knowledge of a subscriber identity/location and/or subscribed RId*
    - **Requirements:** a malicious DHT node or rendezvous network
    - **Method:** It would be preferable if the subscribers could keep their identity and location secret from the rendezvous system interconnect and rendezvous networks storing the scope requested.
    - **Defense:** The recursive routing of the hierarchical DHT could be used to mask the accurate source of a subscription by only revealing the immediate subhierarchy where the request originates from. Encryption of communication between adjacent nodes in the interconnect prevents external eavesdropping. If the subscriber wants to keep the

subscribed RId confidential, it is possible to use a two-pass rendezvous to form a secret channel between the home rendezvous network of the scope and the subscriber, or if the scope has only a single data source, the subscription operation on the rendezvous layer could only contain the requested scope.

12. *Denial of existence of a scope*

- **Requirements:** a malicious DHT node
- **Method:** If the protocol has a message denoting a missing scope, intermediate nodes can store this response and replay it to subsequent subscriptions. On the other hand, if nodes use their public key to sign the responses dynamically, then the keys must be stored online. This problem has been analyzed in the DNSSEC design work.
- **Defense:** Online keys for nodes could be used together with temporary certificate chains or a key revocation mechanism could be designed in the system.

4.4 Packet Authentication for Forwarding Security

This section describes how authorization can be implemented with *Packet Level Authentication* (PLA). The solution can be used in a several ways. First, PLA protects the whole packet payload with a cryptographic signature and ties the publisher and scope identity to the Rid, making it a standalone solution. For additional efficiency, zFilters can be tied to an RID, which is tied in turn to the publisher's public key and packet's signature. Therefore, it will not be necessary to verify the packet's cryptographic signature at every step.

By default, PLA uses so-called *implicit certificate mechanisms* where the publisher's public key is derived from the TTP certificate. This reduces bandwidth and computational overhead. The drawback is that the publisher's public key changes every time when the certificate is renewed. Since our requirements are that the public key is long lived, standard certificate mechanisms should be used in conjunction with PLA. This means that the data packet will contain two sets of signatures and public keys. The signature of the scope owner authorizes the other public key associated with the RId to publish content in the scope. The second signature proves that the content was sent by the authentic RId owner.

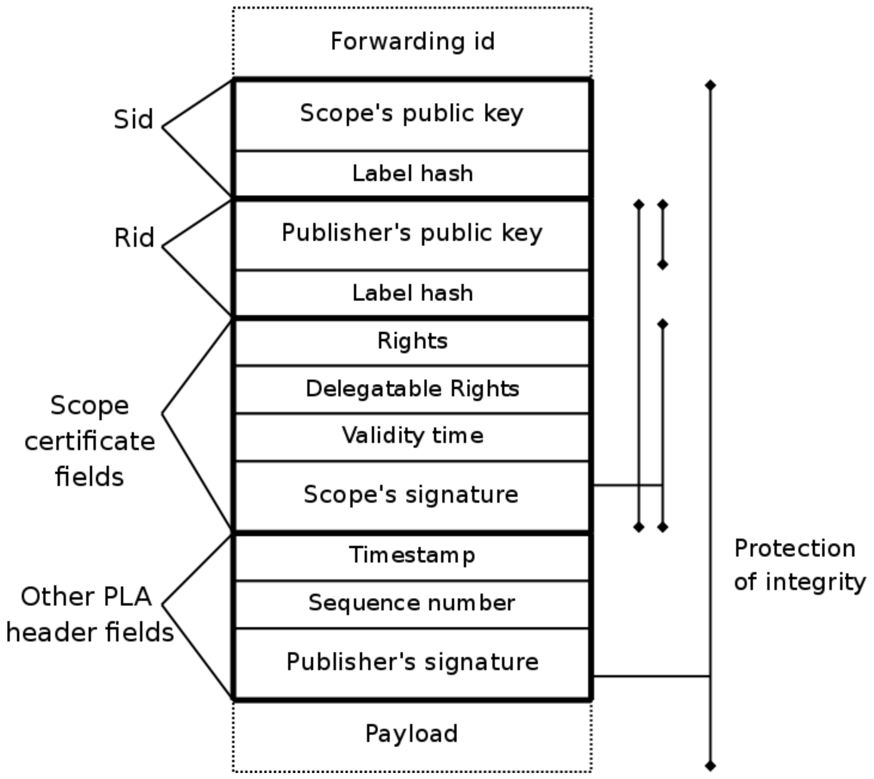


Figure 4.10 – Overview of the packet header

Figure 4.10 describes the header format of the proposed solution. It does not include all possible PSIRP header fields like the header type or metadata. The first two bold boxes contain the SId and RId, respectively. The SId and RId consist of public keys, concatenated together with a label as mentioned earlier. Further, there are other fields of the authorization certificate approved by the scope owner. They include rights fields which denote the rights that a publisher has to that scope and which rights it may delegate forward. There is also a validity time field and the signature over the certificate. As discussed previously, the signature may protect just be the publisher's public key or the whole RId. These alternatives are expressed by lines on the right side of the figure.

In the final part of the security header are other PLA related fields such as sequence number and timestamp that are used to detect duplicated and delayed packets. Finally, there is a publisher's signature over the whole packet, ignoring the FId, to protect the packet's integrity. Since the public keys of the scope and publisher are included in the SId and Rid, the size of the additional security header (scope certificate and PLA header fields) is just 832 bits with padding.

4.4.1 Packet Verification

Below, the required steps are listed for fully verifying the packet at routers or at the final destination. Steps may be performed in a different order, depending on the optimizations chosen.

- Check the sequence number and the timestamp to detect duplicated and delayed packets.
- Check the scope certificate validity time and rights.
- Verify the scope certificate's signature.
- Verify the packet's signature.

To optimize the packet verification process, forwarding nodes may cache hashes of validated scope certificates. Such a method increases the performance, since the scope signature does not need to be verified for each packet. In addition, the forwarding nodes may subscribe to the related security sub-scope to receive events related to revoked certificates. Alternatively, forwarding nodes can rely on the security properties of zFilters.

There are at least two possibilities to handle revocation of rights. The simplest solution is to use short-lived certificates with validity times of hours or minutes. Therefore, if the scope wants to revoke a publisher's access, it will simply not renew the publisher's certificate. Another alternative is for each certificate to have its own RId to which a possible revocation messages concerning this certificate will be published. This RId is derived from the certificate itself, e.g., cer1-RId. Routers that encounter new certificates will automatically subscribe to cerx-RId and therefore will be notified of the revocation of that certificate.

4.4.2 Other Features

The above approach is designed for authenticating the publisher. Signalling traffic to the rendezvous system will not necessarily use this scheme.

To enhance security, we may want to use a separate certificate that gives the publisher permission to use the network. Such a certificate would be issued by, e.g., an operator and it would improve security since traffic from misbehaving nodes could be stopped more efficiently. Additionally, it could be used for per packet or per bandwidth billing.

4.5 Network Attachment

The network attachment process (described in Section 4.7 of [PSI09a]) enables publishers and subscribers to join the pub/sub network. Network attachment nodes publish advertisements broadcasted on link interfaces. Nodes that want to get such information publish subscriptions that other nodes can respond to. These initial messages contain information about attachment points, including subscription data for establishing two-way control channels with selected nodes. These advertisements and solicitations are published with identifiers that are pre-defined and well known.

The network attachment might be subject to DoS attacks, mainly in the form of resource depletion attacks. An example of how DoS can be initiated follows. In a network attachment scenario, once a node (say *B*) has already collected information about other attachment points by subscribing to predefined scope S_{NA}, it chooses to attach to a particular node, say *A*. It does this by publishing an initiation message (P_{BA-0}) with Rid_{BA-0} in scope SId_A. Subscription data for A-to-B is included in this message, along with other information, such as an authentication request. *A* responds with message P_{AB-1} and thus the two nodes establish a two-way control channel. This handshake procedure might be used by many malicious nodes that use the same steps as *B* does to select and attach to node *A*, and establish a control channel. If attachment requests exceed a particular threshold per unit time, or if the channel establishment is associated with complex authentication primitives, then *A* (or any other target node) might suffer from a DoS attack.

Such DoS attack scenarios might be escalated when state-full network attachment is implemented. Further escalation is expected if virus-infected machines (zombies) are coordinated to attack particular targets, e.g., network attachment points belonging to a single operator.

The aforementioned attacks are expected to be launched before any authorization or authentication procedure of the initiator (i.e., publisher or subscriber) or authorization of its public keys. For that reason, it is not easy to employ other the DoS mitigation techniques already proposed in pub/sub networks. For instance, the *PSGuard*, proposed in [Sri07], is a secure mechanism for distribution of advertisements. PSGuard ciphers the updates to guarantee the confidentiality of hidden data contained in an update. This requires that the hidden data can be read only by authorized subscribers, i.e., those that have the corresponding keys. In [Min08], a secure aggregation mechanism was proposed. This enables publishers to transmit data to legitimate users without being understood by other subscribers or network elements. This scheme also requires the authentication of legitimate users.

In our scenarios, in order to prevent DoS and to avoid connection depletion attacks, we can introduce some computational obligation for attaching to the network or associating with a scope. This obligation will be easy and cheap to be answered or resolved by a computer machine, but it requires a reasonable amount of time or moderate resources to be committed.

Thus, considering that an attacker can make requests to attach without ever completing the attachment process, or multiple devices can coordinate attacks on the attachment points, then a suggested countermeasure is to use computational puzzles or challenges. The initiator is challenged to perform the computation or solve the puzzle, and pass the result back to the NAP (Network Attachment Point) or scope owner. This computation has to be something a human user cannot easily perform and something expensive enough to increase the initiator's processing or communication cost. This cost increase has to be high enough to make it prohibitively expensive for attackers but inconsequential for legitimate initiators. Actually, solving a puzzle gives the initiator of the attachment process an access, for a time interval, to a virtual channel on the NAP, i.e., to a small slice of the server's resources [Wat04].

In [Par01], cryptographic salts are used to prevent DoS attacks in the Internet. This requires that the initiator of the attachment process (i.e., logon request) should encrypt the provided nonce with its own nonce using the public key of the server. Then, the server decrypts the cipher and validates the provided nonce. In general, the server (NAP or the scope owner in our case) should pay out less effort or cost to produce or verify an individual challenge than the respondent to solve and communicate the answer.

Some important attributes for the required solution include [Jue99] [Aur00]:

- Creating a puzzle and verifying its solution is an inexpensive computer process
- The cost to produce a puzzle should be much lower than required to solve it
- The computational cost of a solution is easily adaptable
- The puzzles can be solved in most hardware configurations
- It is impossible to calculate the solution of the puzzle before its announcement, even for those people who create the puzzle
- While the respondent solves the puzzle, its creator is not obliged to store information about the respondent or the solution
- The same puzzle can also be given to many individual respondents, but if one discovers its solution that does not mean that it could help others

For the purpose of PSIRP, a variation of the scheme originally presented in [Aur00] is adopted. We use an authentication-free variation of this scheme, since public keys and corresponding certificates cannot be verified yet. Additionally, since network attachment is solicited in PSIRP (i.e., initiators ask from entities to attach), the nonce is produced dynamically, and not periodically.

Hereafter, we use the term server to indicate a NAP that receives publication requests for attachment, or a scope owner that receives public keys for signing access rights to scopes. Clients are the requestors. When a request is received, the server generates a nonce *Ns* and sends it to the client. To prevent the attacker from submitting pre-computed values, *Ns* needs to be random, for example, using a timestamp (i.e., *Ts*). The server also decides the difficulty level *k* of the puzzle. The Server sends *Ns* (or *Ts*) and *k* to the client. To solve the puzzle, the client generates a random nonce *Nc*. The main function of this client nonce is to prevent eavesdropping attackers from responding to the challenge before the client (or at least all clients). Then, it tries to find the number *X* that satisfies the following equation:

$$h(Ns, Nc, X)=00000\ldots0Y$$

where:

h is a cryptographic hash function (e.g., SHA2),

N s= the server's nonce,

Nc = the client's nonce,

X = the solution of the puzzle,

k = the puzzle difficulty level,

00000…0 = the k first bits of the hash value; must be zero, and

Y = the rest of the hash value.

The client sends the values *Ns*, *Nc* and *X* to the server, which validates the result. The difficulty of the puzzle is adapted based on the parameter *k*. Solving the puzzle depends exponentially on the required number *k* of zero bits in the beginning of the hash. Reasonable values of *k* lie between 0 and 64. We can use

- Easy puzzles (k=0-15) for fully trusted clients
- Medium (*k*=16-31) for marginally trusted clients
- Difficult (*k*=32-47) for marginally untrusted clients
- Strong (*k*=48-63) for non-trusted or unknown clients

Also the value of k can be adapted from easy to strong whenever the server is overloaded with requests. This is equivalent to the rate limitation techniques for DoS prevention.

4.6 Preventing Spam through Notarization

In order to define spam in PSIRP, we should first concentrate on the communication model and its impact on anti-spam design. Within the PSIRP architecture, rendezvous points and scope owners provide a control plane for connecting publishers and subscribers in a security policy compliant fashion. In the PSIRP model, the communication between publisher and subscriber is mediated by a rendezvous point (RP), which matches the interests of subscribers to the content or service provided by the publishers, when interests and provisions are associated with a particular scope.

An RP is not equipped with functions that self-prove the validity and accuracy of the content or service provided by the publishers, and thus, it is not able to prove that the published and advertised content is actually spam. From an economic viewpoint, this may have an impact on the AS providers that operate RPs since subscribers will not select the RPs providing inaccurate services, i.e., those that match malicious spam content to subscribers' interests.

On the other hand, to support the Trust-to-Trust design principle, it is essential to include trustworthiness in the rendezvous execution phase and associate this service with trust semantics.

Spam in the PSIRP paradigm is considered as any unsolicited, bulk communication that attempts that produce annoyance or displeasure to end-users, publishers or subscribers, and as a second-order consequence, influence the normal operation of the pub/sub network. The obvious objective for dealing with bogus entities and spam initiators in PSIRP is to define the entities that could behave like spammers and their targets, namely the victims. We investigated how each entity behaves and how it could produce spamming in the application layer, i.e., for the end-user (publisher or subscriber). Here we place emphasis on the situation where spam is injected by end user. Other studies are concentrating on spam that is produced by entities that operate within the network and have been compromised by attackers (e.g., the Botnet- Bogus Brokers in [Ta06]).

4.6.1 Subscribers as Spam Initiators

In this case, when subscribers act as spammers, they cannot have the publishers as targets, since there is no direct communication, and any signaling connection is made through the RPs or scope owners. Therefore, we consider only whether subscribers conduct spamming attacks against RPs. This is feasible since they may flood the pub/sub network with duplicated, unmatched, or workload complexity subscription requests [Wu07]. For instance, they might register interest for data (via the primitive *interest(RId.data)*) that is not included in the scope with identifier *SId*. Or they might submit registration requests using a bogus certificate (i.e., *CER.sub*). In addition, some subscribers might be curious to obtain information about publications that are not included in their interests. Concluding, even if the subscriber wishes to deliver spam in PSIRP, this will only influence the RP operations, and its impact will be equivalent to a DoS attempt. The publisher, on the other hand will be unaffected, since the RP mediates between the two end entities, and the main problem that the publisher will realize is the potential degradation of the service provided by the associated RP.

4.6.2 Publishers as Spam Initiators

If we assume that using the *advertise* or the *provision* methods, the publishers distribute promotional info that is received by subscribers, then a spamming condition might appear. This is similar to mail or instant messaging spam, and there are several prevention methods, such as authentication of the publisher, black/white lists, grey lists, Turing or computational puzzles (e.g., CAPTCHA tests), or cryptographic puzzles. A more complicated situation arises when publishers advertise content or services inaccurately, meaning that the description (or metadata) and the identification of the publication (e.g., a unrepeatable algorithmic-based hash value such as *RId.data*) does not coincide with the actual content or service that the subscribers finally receive. This type of content-based spamming is much more costly in terms of network operations than advertising spamming. Large files of bogus content may be delivered, whilst the network spends lot of its resources and processing capacity to create efficient topologies and forwarding schemes for marginal profit.

We will concentrate on this spam scenario, where the advertised or registered content does not match with the content delivered to the requesting subscribers. We assume the existence of a TTP entity that issues public key certificates to publishers, subscribers, and the RPs. Also an Arbiter entity is used as follows.

Once the RP matches a subscription (i.e., interest for some *RId.data*) and a publication with the same data id (*RId.data*), it uses a hash function $h(M_X)$ of the data (M_X) with its private key, KR_P and passes the information $V:=KR_P(h(M_X))$ to the Arbiter.

The publisher X signs the content M_X of the publication RId.data and its hash or description function $h(M_X)$ with its private key, KR_X. If the publication policy or scope defines that confidentiality should be supported, the publisher X encrypts the result using the public key KU_Y of subscriber Y. The identifier of X, i.e., ID_X, is also included, and finally the publisher with ID_X sends to the Arbiter A the message M_X'' as

$$X \rightarrow A: M_X'' := E_{KRx}[\ E_{KUy}[\ E_{KRx}[\ M_X\ ]\ ]\ ||\ ID_X\ ||\ \text{Rid.data}]\ ||\ ID_{X,}\ \text{Rid.data}$$

The Arbiter A checks if the public key of X is valid. It uses KU_X to verify M_X'' and verifies that it was indented for the publication *Rid.data* from publisher X.

Then, A produces a message M_A that consists of ID_X, the crypto part $E_{KUY}[E_{KRX}[M_X]]$ and the message V. It signs this with its private key KR_A and sends this to subscriber Y

$$A \rightarrow Y: M_A := E_{KRA}[\ E_{KUY}[\ E_{KRX}[M_x]\ ]\ ||\ V||\ ID_X\ ||\ RId.data]\ ||\ ID_{X,}\ RId.data$$

Subscriber Y applies KU_A to M_A and rebuilds $E_{KUY}[E_{KRX}[M_X]]$, retrieving V. Then it uses KR_Y, it gets $E_{KRX}[M_X]$, then applies the public key KU_X of X and gets M_X. It applies the function h and gets $h(M_X)$. It uses V and KUp to get $h(M_X)$ and if $M_X= h(M_X)$, then M_X is neither bogus nor spam.

The arbiter is used as a trust anchor. In the case where the subscriber claims that the RId.data is spam and the publisher denies that was the initiator of this, the crypto element $E_{KUY}[E_{KRX}[M_X]]$ can be used for this dispute. This element reveals which party of the communication was untruthful.

The aforementioned scheme can be used only when some scope security policies require such an arbitration service (probably included in the *SId.security* sub-scope).

Alternatively, it can be the basis for a compensation scheme as follows. Initially, all the publications of a publisher are provided to subscribers without arbitrations. When some subscribers start complaining about bogus content from a particular CER.pub and RId.data pair, then the arbitration should be enforced, probably using some payments on behalf of publishers as penalties. This penalty will eventually get back to zero, rewarding trustworthy behavior, when the service from that particular publisher reaches again satisfactory levels. A similar technique is used for the scheme Payments at Risk, introduced in [Aba03].

For the payments, we might consider a penalty scenario where the compensation policy is related to a pricing question concerning the amount of the payment a node should ask to forward packets. When the topology managers have synchronized and created the path to deliver the content *i* to matched subscribers, then each forwarding node *j* in the path estimates a cost-of-energy (c_{ij}) to forward this content *i*. This cost depends on energy consumption, bandwidth, QoS, or other semantics. The arbiter asks the topology managers to estimate forwarding costs along the paths for delivering *i*. Then, when a bogus publisher delivers the spam content *i*, this publisher is asked to pay at least the sum of the cost-of-energy (c_{ij}) values of the path. The cost-of-energy parameter is a time dependent function, which takes into account each forwarding nodes' preferences and current conditions (e.g., overloaded). We should be sure that each node's incentive is to reveal its true cost. An implicit incentive proving that it is not profitable for a node to alter the cost on purpose is the fact that this node might not be chosen for forwarding in the future, since its cost function will be too high. When a node is excluded from forwarding paths, its utilization factor is decreased, and the AS operator that manages this node might perceive a reduction of welfare gained from this node. For a successful implementation of this scheme, each forwarding node should not be aware about the costs of other nodes, and this could be secured using cryptographic techniques when these values are reported to the topology managers. Finally, each subscriber should be compensated, and, thus, a cost of damage should also be included in the model.

The aforementioned payment scheme might be applied proactively or reactively. In the former case, an amount of deposits equal to the total cost of the path are requested from the publisher. These deposits are rebated to the publisher when no bogus content is delivered to the subscribers. Otherwise, they are reserved by the Arbiter and shared among the involved ASes and the subscribers. In the latter case, the publisher is requested to compensate the forwarding nodes and the subscribers when bogus content is delivered and certified by the Arbiter.

4.7 Reputation Services for Publish/Subscribe

In pub/sub systems, publishers and subscribers are decoupled in time and space and usually they are not aware of each other. As a result, they rely on a third party for their communication, and this third party has to be trustworthy. Moreover, being information-centric, pub/sub systems boost the competition between various content providers as well as opening the ground for information location services. In such an environment, a high trust level and good reputation are necessary. On the other hand, pub/sub systems have the idiosyncrasy that each node does not have direct and reliable evidence of other nodes' reliability but have to rely on reputation values. Moreover, the trust value that a node A has about a node B is useful for A in order to execute a function on behalf of another node C. In other words, B and C will interact only if A considers B trustworthy. There are numerous examples in which trust can play an essential role in a pub/sub system's lifetime. Trust can be used for building mechanisms that will charge subscribers for receiving publications. It can be used to protect subscribers from spammers, botnets and viruses. Trust can be used for selecting an effective information search engine, or for selecting a prestigious publisher, as well as for limiting information dissemination.

To design an effective trust management framework, it is essential to model and evaluate, in terms of satisfaction or dissatisfaction, the behaviour of the key entities participating in the architecture, namely the publisher, the subscriber and the broker. Moreover, it should be determined which network elements evaluate and report behaviours.

Different trust semantics can be defined for each entity involved in the pub/sub architecture.

- *Publishers*: When a publisher does not publish the promised content, or provides only part of it, its trust diminishes over time. Moreover, bogus content produces network overhead and subscriber dissatisfaction, whilst no content consumes rendezvous point processing cycles and signalling overhead. Such conditions should decrease the publisher's reputation value. Additionally, when one publisher continuously advertises publications that are not chosen by any subscriber, the rendezvous plane will eventually serve worthless events. This condition also decreases satisfaction level. Moreover, a publisher might use tricky techniques to become attractive to the subscribers. This will increase its information dissemination and allow a publisher to potentially profit from the increased subscription. Tampering with metadata, which is a form of spamming or phishing, might also cause dissatisfaction.

- *Subscribers*: A non-trusted subscriber might continuously subscribe to the same set of publications, even for non-existing publications (although this might be interpreted as a normal behaviour in some systems). Some queries might also be considered as misbehaviours - for instance, a subscriber tries to address the whole publication space. When a subscriber attempts to reveal (re-publish) content that it has downloaded before, but on a less protective scope, then this misbehaviour can also been seen as misbehaviour. Moreover, subscribers should be monitored when trying to flood the network with subscriptions or conduct subscription toggling. Such behaviour might appear as DoS attacks to the rendezvous points. Finally, publishers might want subscribers to compensate them for the publications they received. When compensation is supported, false positive or negative rankings are possible, and these actions should also decrease the satisfaction level for the untruthful subscribers.

- *Network*: Pub/sub rendezvous services might also be part of the trust assessment, not only by supporting publisher or subscriber trust decisions, but as subjects of the evaluation as well. We can distinguish two types of brokers, namely forwarding brokers and rendezvous points. Reliable forwarding brokers shall route data to the proper destination without modifying or revealing it to unauthorized entities. On the other hand, rendezvous point satisfaction levels might also be evaluated, since, for instance, they should not give preference to specific publisher notifications or subscriber requests. Moreover, they should provide efficient and fair matching. A reliable rendezvous point should properly forward the subscription messages as well as establish the forwarding path between publisher and subscriber. Moreover, a publisher trusts the rendezvous point not to include unauthorized subscribers that are not part of the scope.

For the successful operation of a trust management system, we need to define the function by which the judging entities will evaluate the judged entities as well as the definition of the entities that needs to report regarding this function. Furthermore, a mechanism for the trust degree calculation based on direct experience and reputation values has to be agreed among the cooperating nodes. Finally, it is necessary to design a procedure for making decisions based on the trust values maintained for various nodes.

5 Inter-domain Topology Formation

This section presents an update of the work on inter-domain topology formation, as first presented in D2.3 [PSI09a]. We first outline the design goals of the work, the resulting design choices for the implementation of this functions as well as early result evaluating these design choices, including the used methodology.

5.1 Design Goals

In this section, we discuss the problem of inter-domain topology formation (ITF), our assumptions and the design requirements driving our technical considerations regarding possible solutions. In this context, the topology layer works with the rendezvous and forwarding functions in the routing phase of communication to:

- Help build routing information about the forwarding nodes and forwarding paths according to policies set by operators and users
- Store policies and network topology information
- Manage edge routers between domains that prevent policy violations and protect domain internals

We assume a PSIRP network will be divided into autonomous systems (domains) controlled (as in the Internet today) by a mix of competing, commercial operators seeking profit, as well as communities like universities and governments that may have other goals. Domain level connectivity is largely determined by the relationships between these organizations, the needs of their customers, geographical, historical and political considerations, and only indirectly guided by the technology used. The starting point for our work has been the current Internet but it is expected that the change in the underlying network paradigm, represented by PSIRP, will affect the evolution of the domain level topology.

In today's Internet, packets tend to flow along so-called *valley-free paths*, typically consisting of several transit provider links upwards, then zero or one peering link, and finally several transit customer links downwards. Policy compliance on the path is fully controlled by the operators. In the future PSIRP-based Internet, the paths must reflect policies between operators and also RId-specific policies negotiated by the publishers and subscribers in the rendezvous process, therefore they are likely to become much more flexible.

Currently, home users typically buy peak bandwidth to their local ISPs network instead of paying according to the amount of traffic consumed. This is understandable from a risk management point of view because end users want to limit total cost and do not want to actively think about it. Based on these observations, we assume that typical users will require predictable costs in the future, too.

5.1.1 Design Requirements

Previous work [PSI09a] has identified the following list of requirements for the inter-domain topology forwarding function, as a basis for subsequent design choices:

- The topology layer should allow operators to flexibly control the routing policies of the packets traversing their domain.
- It should be possible for customers to define RId-specific policies, which are then taken into account in the overall topology formation.
- A PSIRP solution should consider the costs and policies of both publishers and subscribers when building forwarding trees (in contrast to the current Internet where only sender side control of the routing exists).

- The topology layer should have enough expressive power to enable complex policies and business relationships between ASes to be described, not relying on assumptions such as a fixed set of Tier-1 operators and strictly hierarchical AS topology (e.g., multi homing and RId-specific partial transit should be easily possible).
- Operators of domains should be able to keep their intra-domain topology hidden, only being required to expose minimal information for the purposes of topology formation.
- Inter-domain topology formation should not unnecessarily limit the implementation and management of intra-domain topologies. There can be different implementations inside domains, all compatible with the inter-domain topology formation.
- Incremental deployment on top of the current Internet AS topology should be feasible.
- The topology layer should automatically and quickly adapt to changes in network topology and efficiently use available routing resources, in accordance with constraining policies.
- Topology formation should take into account the fact that large domains are linked at multiple geographically dispersed PoPs (even if they only have a single logical business association) and by exposing some intra-domain information the routes could be further optimized between the domains.
- Topology formation should allow for potentially different policies between the same domains, depending on their point of interconnection.

5.1.2 Conceptual Components

In this section, we briefly review the conceptual component architecture relating to inter-domain topology formation, as detailed previously in [PSI09a]. A *topology management* function is assumed to exist within each autonomous system (domain). This function implements the local topology management and communicates the relevant peering information to the inter-domain topology formation function.

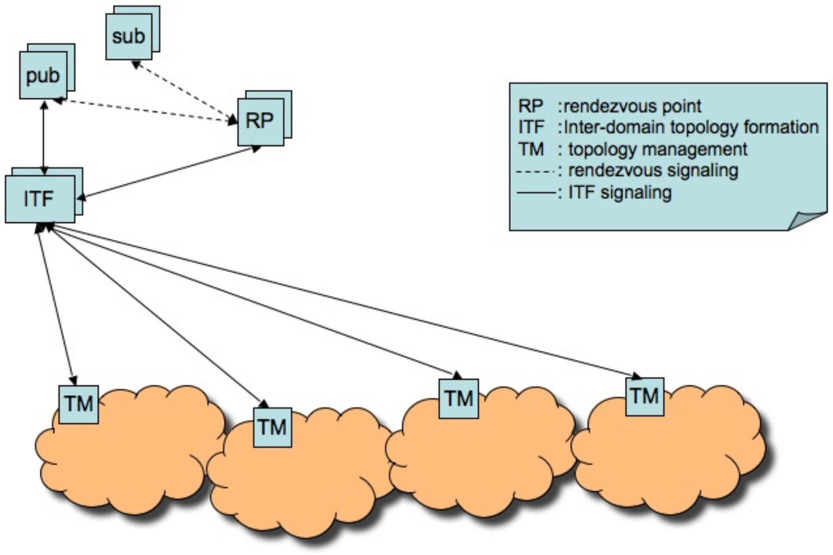


Figure 5.1 – Conceptual components for inter-domain topology formation

Publishers and subscribers come together in the *rendezvous* process within the rendezvous point representing the particular SId in which the information items (labelled via an RId) are located. The arrows in Figure 5.1 show the relations of these components and are not meant to illustrate the exact message and information exchange between them. However, dashed arrows indicate relations stemming from the rendezvous process while solid arrows show topology formation relations.

5.2 Design Choices

The above discussion has served to define:

- The inter-domain topology formation problem
- Our accompanying assumptions
- The design requirements to be satisfied by the ITF function
- The conceptual component architecture for the ITF function implementation

Previous work [PSI09a] has also outlined a series of initial considerations to guide ITF *design choices*, addressing issues such as:

- Role of the ITF component (efficiency and modularity)
- Interdomain topology information (granularity)
- Publish/subscribe approach to interdomain topology formation
- Creation of a (policy-driven) peering topology market
- Control of the formation process (which parties make decisions)
- Fault tolerance and multipath routing
- Anycast

Given these, we seek to understand the impact of PSIRP ITF *design choices* (made to encourage technology deployment) with regard to both:

- Evaluation of which design choices are possible under (evolving) socio-economic conditions
- Development of dedicated solutions

With the use cases centred on retrieval of an information item (i.e., after matching at the rendezvous point has been performed), the focus of business modelling will be on the interaction between ISPs, rendezvous provider, and intra-domain topology provider (implementing topology management for a particular AS). Modelling work to-date [PSI09b] has used the Xmind brain-storming tool to identify both:

- ITF technical implementation options (“control point constellations”)
- Primary factors (“triggers”) determining ITF behaviour/capabilities in a range of likely service scenarios (where users desire routes exhibiting various combinations of policy-compliance, trust, QoS and resilience)

A *control point constellation* (CPC, see Figure 5.2) represents a particular technical implementation of the use-case under consideration, based on specific architectural assumptions (i.e., *design choice*). It captures the ability of an architecture to support a variety of business models (and the associated value generation of the players involved).

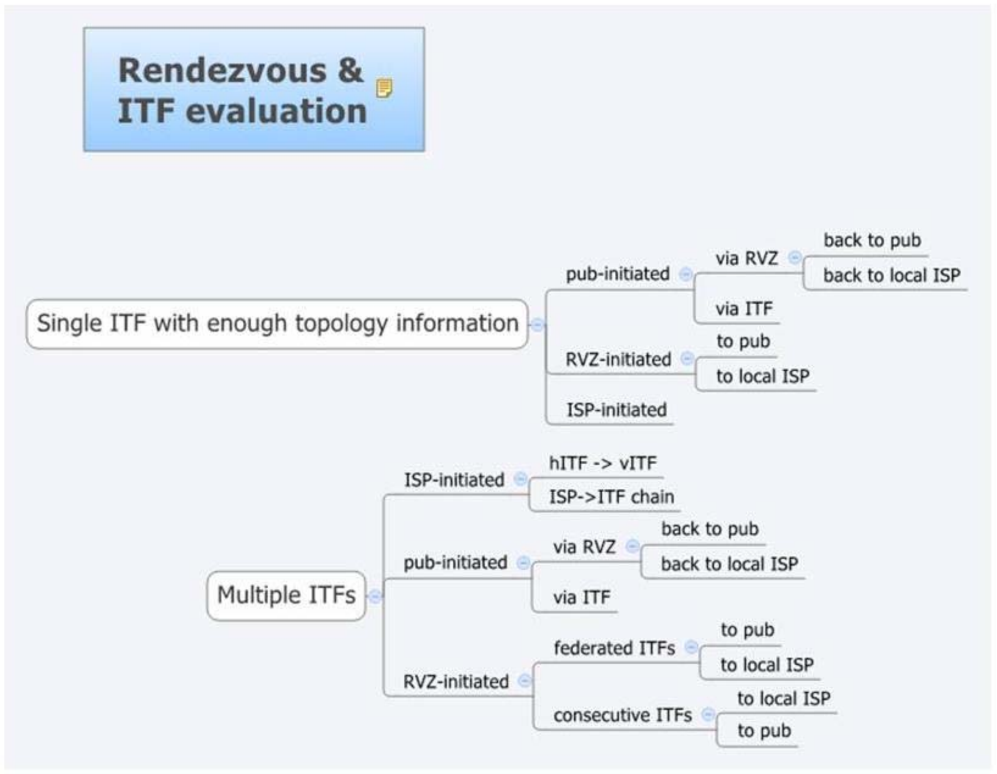


Figure 5.2 – CPC for inter-domain topology formation

We can see from Figure 5.2 that there exist a variety of potential design choices, involving the initiation of the topology formation process either through the publisher, the rendezvous point (serving the retrieval request) or the local ISP to which the publisher is connected. The variety of the design choices result from addressing the problem of exposing the necessary (topology) information to the variety of involved parties. In other words, the primary concerns that are addressed by the design choices are "*who initiates communication*" and "*how to accommodate sensitivities regarding exposure of information*" (e.g., ISP routing topology).

As outlined in D4.3 [PSI09b], the primary tool to evaluate the viability of the many design choices is the notion of *triggers*. These triggers represent the potential socio-economic force that influences particular aspects of the viability of a particular design choice. The reader is referred to D4.3 for more information on the underlying methodology to derive these triggers. Based on this methodology, the corresponding ITF triggers are outlined in Figure 5.3, arranged into various categories (technology, user behaviour, regulation etc.) reflecting their role in topology formation. On the technology side, advances in performance and availability of new features must be considered alongside legacy compatibility and reliability requirements. Regulatory pressures might range from traditional controls over access/transit competition to new concerns regarding information visibility at the individual packet content level. Apart from the obvious price concerns, user perception of coolness/usefulness will be crucial for stimulating demand, while business/industry sensitivity to exposure of information (topology hiding) must again be recognised.

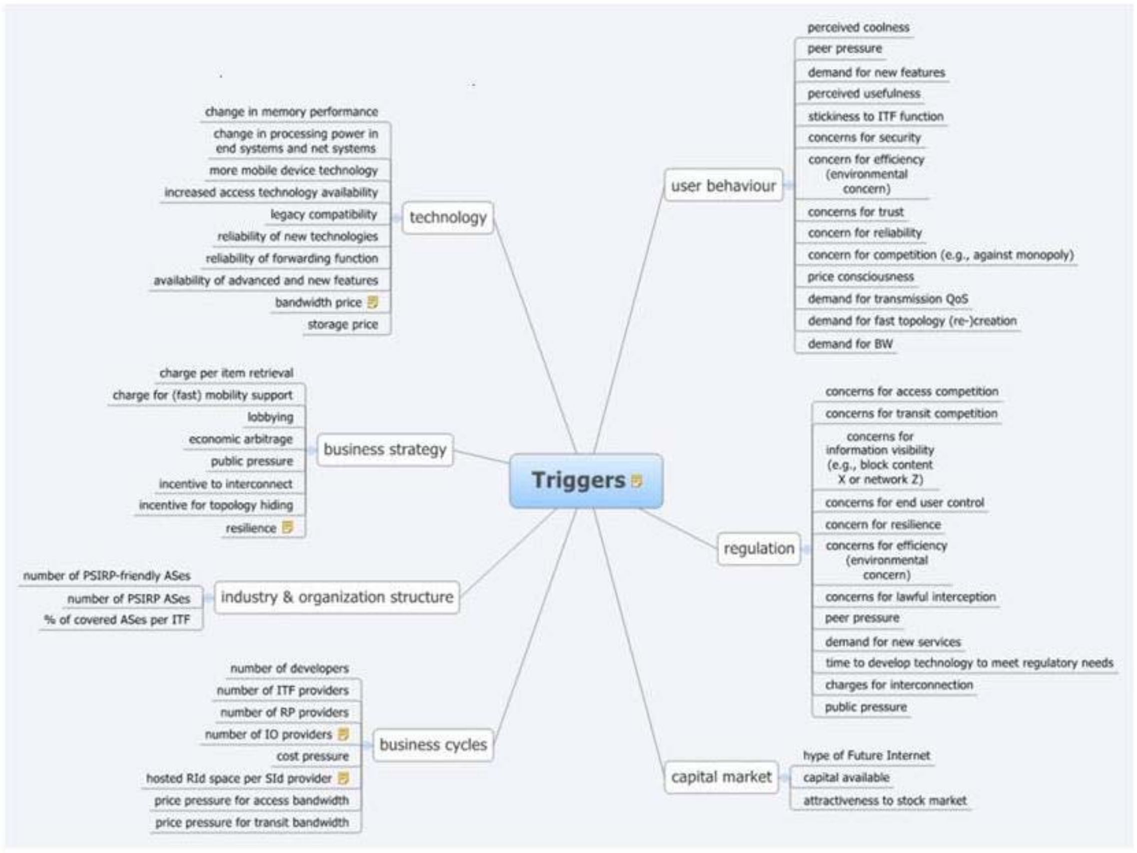

Triggers
technology
change in memory performance
change in processing power in end systems and net systems
more mobile device technology
increased access technology availability
legacy compatibility
reliability of new technologies
reliability of forwarding function
availability of advanced and new features
bandwidth price
storage price
business strategy
charge per item retrieval
charge for (fast) mobility support
lobbying
economic arbitrage
public pressure
incentive to interconnect
incentive for topology hiding
resilience
industry & organization structure
number of PSIRP-friendly ASes
number of PSIRP ASes
% of covered ASes per ITF
business cycles
number of developers
number of ITF providers
number of RP providers
number of IO providers
cost pressure
hosted RId space per SId provider
price pressure for access bandwidth
price pressure for transit bandwidth
user behaviour
perceived coolness
peer pressure
demand for new features
perceived usefulness
stickiness to ITF function
concerns for security
concern for efficiency (environmental concern)
concerns for trust
concern for reliability
concern for competition (e.g., against monopoly)
price consciousness
demand for transmission QoS
demand for fast topology (re-)creation
demand for BW
regulation
concerns for access competition
concerns for transit competition
concerns for information visibility (e.g., block content X or network Z)
concerns for end user control
concern for resilience
concerns for efficiency (environmental concern)
concerns for lawful interception
peer pressure
demand for new services
time to develop technology to meet regulatory needs
charges for interconnection
public pressure
capital market
hype of Future Internet
capital available
attractiveness to stock market

Figure 5.3 – Triggers for inter-domain topology formation

CPC, triggers and scenarios thus serve to define the modelling problem for study within the System Dynamics simulation environment, as discussed below.

5.3 Evaluating Design Choices

As outlined in [PSI09b], the socio-economic methodology we developed ultimately provides a tool to verify the viability of the design choices outlined in Figure 5.2. As a starting point for our following presentation, we take the results from D4.3, i.e., the execution of the first part of our methodology. From there, we outline the methodology to derive the results of the second part of our evaluation, which we present in more detail.

5.3.1 Socio-economic Starting Point

The PSIRP socio-economic work seeks to guide upstream architectural decisions to better take into account economic forces further downstream, particularly with regard to encouraging technology deployment. In practice, this involves recognising the broad range of likely requirements and ensuring that the technical capability exists to best meet these needs.

In particular, the ITF supplies inter-domain topologies, ultimately facilitating distribution of content via "quality" routes (for publishers and subscribers who are willing to pay for better than Internet best-effort transport services). Significant efficiencies can potentially be achieved via economies of scale, increasing ITF incentives to grow.

The market is "two-sided" because customer/provider numbers are strongly coupled by a feedback mechanism (to a greater extent than in a simple supply-demand relationship), so one/both sides of the market benefit from increasing adoption and/or consumption by the other side. In such a situation, platform pricing can be adjusted so that one group effectively subsidises another (perhaps through group discounts or bundling). The resulting pricing structures can be complex and strongly influenced by competition.

5.3.2 Methodology

Our overall methodology, as outlined in D4.3 and in D4.2 [PSI09c], takes into account this larger socio-economic environment. The methodology supports the step-wise investigation of this environment, enumerating design choices and the potential influences on the viability of these design choices. In the previous Section 5.2, we presented the results of this part of the overall methodology.

As outlined in D4.2, the second part of the methodology leads to the development of *system dynamics models* [Ste00], a computer-based technique with origins in many fields including control theory, cybernetics, organizational theory, behavioural psychology, economics and simulation. It helps building models of complex systems to aid understanding. The completed models are used to test changes aimed at improving system behaviour. These models are based on identifying the causalities among various influences in such system, modelling these causalities with the help of graphical *causal loops*, and parameterising the influences based on solid desk-based research. This, eventually, enables the simulation of various socio-economic scenarios. In our case, we expect to model various scenarios evaluating the viability of particular design choices, as outlined in Section 5.2.

The structure of a system dynamics model consists of a collection of “stocks” (system variables representing items which can accumulate over time) and “flows” (rate of change of stocks). In a system dynamics model, stock and flow structures are embedded in feedback loops (both positive and negative). Together, these stocks, flows and feedback relationships define the actual structure of a system, including any decision-making processes. The essential philosophy might be summarised as:

- Systems are fundamentally dynamic. The evolution of the system in time is its primary characteristic, rather than its state at any given instant.
- Systems themselves are defined by stocks and flows.
- Behaviour of a system is ultimately controlled by its structure, in terms of stocks, flows and positive/negative feedback loops.
- System controls are typically circular feedback relationships, rather than linear chains of cause and effect.

5.3.3 Causal Loops

In the present ITF context and within our aim to reflect upon the viability of our various design choices of Figure 5.2, we seek to define the relevant stocks, flows and feedback loops based on the CPC, triggers and scenarios identified in Section 5.2, using our current realization of our methodology iin the Xmind tool. The specific stocks selected as the most important are:

- *Number of ITF providers*
- *Importance of rendezvous provider (RP) views when choosing topology*
- *Importance of local ISP views when choosing topology*
- *Importance of publisher views when choosing topology*

The *Number of ITF providers* is an obvious measure of market size, while the remaining “importance-related” stocks reflect the balance to be struck in accommodating the different motivations and policies of the RP, ISP and information publisher. Within system dynamics, control relationships amongst the various components may be captured in a “causal loop” diagram, as shown in Figure 5.4, where items 1-14 denote selected triggers as follows:

1. RP choice importance

2. regulation (information visibility)
3. ISP choice importance
4. regulation (transit competition)
5. publisher choice importance
6. regulation (access competition)
7. incentive for topology hiding
8. user concerns (e.g. anti-monopoly)
9. industry concerns (time to develop technology to meet regulatory needs)
10. demand for BW etc./concerns for trust
11. charge per item retrieval
12. capital available
13. hype
14. perceived usefulness

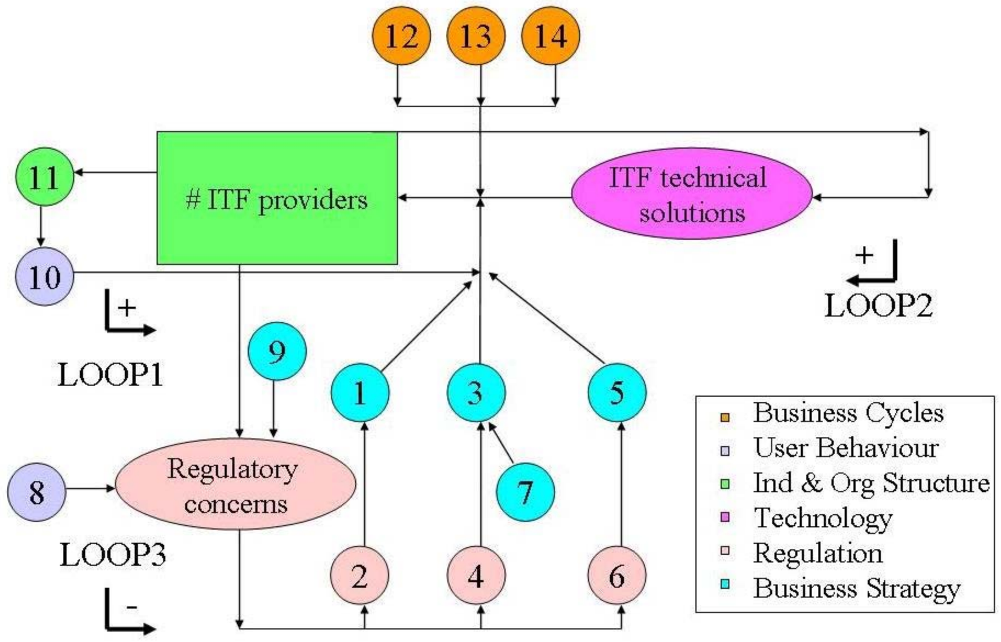


Figure 5.4 – Causal loops for inter-domain topology formation

Arrows represent influences of triggers on stocks/flows and on each other. Some triggers are essentially treated as inputs (e.g., items 12-14 related to business cycles, largely reflecting external economic conditions). However, there are also three feedback loops evident in the diagram (LOOP1, LOOP2, LOOP3):

- Supply vs. demand between ITF providers and users, generating positive feedback
- Technical solutions supporting ITF providers, leading to demand for additional improvements, generating positive feedback
- Regulatory concerns (items 2, 4, 6), tending to restrain market growth and also influencing the importance of RP, ISP and publisher topology choices (items 1, 3, 5), generating negative feedback

The causal loop description thus exposes the dynamic structure of the ITF system. It forms a natural prelude to a System Dynamics simulation, where the goal is to explore in more detail the consequences of ITF design choices for technology uptake in the wider socio-economic environment.

The currently selected triggers represent a first iteration for developing a first causal loop of our problem. With that, Figure 5.4 serves as a basis for further developing the causal loops more specifically for the problems at hand, represented through the various stock and flow models. This is likely going to lead to more complex causal loop models. This development is going to take place using commercial system dynamics tools that will directly enable simulation of desired scenarios.

5.3.4 Reference Modes

In any modelling exercise, it is crucially important to validate results and hence build confidence in the final model. The system dynamics technique approaches this via consideration of “reference modes” which are time series graphs of key system variables, showing their likely behaviour in typical scenarios (e.g., observed historically or expected in future). The specification of a system's reference modes also serves to capture mental models and suggest appropriate model structure, helping to:

- Identify important variables
- Establish the likely time scale (duration) of interest
- Highlight relevant behaviour the model must mimic (e.g., oscillation, overshoot and collapse, S-shaped growth etc.) in a particular regime

These reference modes are usually developed before the actual causal loops. They serve as a reference point of the expected behaviour so that adjustments can be made throughout the development of the underlying causal loops for the case that expected behaviour (as identified in the reference modes) and actual behaviour (as expressed through the causal loops) diverge. Such adjustment can either comprise of adjustments to the reference modes or to the causal loops. The adjustments are usually undertaken in dialogue with the involved stakeholders that were involved in the design of the original material (see D4.2).

As outlined in Section 5.3.3, we have identified four stocks as the most important variables, while ten years is here suggested as a likely order-of-magnitude estimate for time scale (based largely on historical experience of Internet peering evolution). The general behaviour each stock might reasonably follow is postulated as a traditional “S-curve”, involving a relatively slow start-up period, followed by rapid growth, terminating in a stable “plateau” at a level determined by the overall success of the technology.

Considering first the *number of ITF providers*, the market might be expected to evolve in various ways over time, subject to regulatory constraints. In particular, fragmentation into “regions” may occur, these defined according to some general notion, such as:

- Geography to help optimise network resource usage
- Local peering, conditioned by business relationships, multiple providers cooperating to provide extended coverage
- Resilience requirements (e.g., financial services provision as outlined in [PSI09b]) implying direct fragmentation by market

The reference mode for the *number of ITF providers* stock is shown in Figure 5.5. The degree of success envisaged ranges from an optimal scenario involving very strong take-up to relative failure, where Internet users/providers largely ignore PSIRP and rely mainly on BGP-type interconnect.

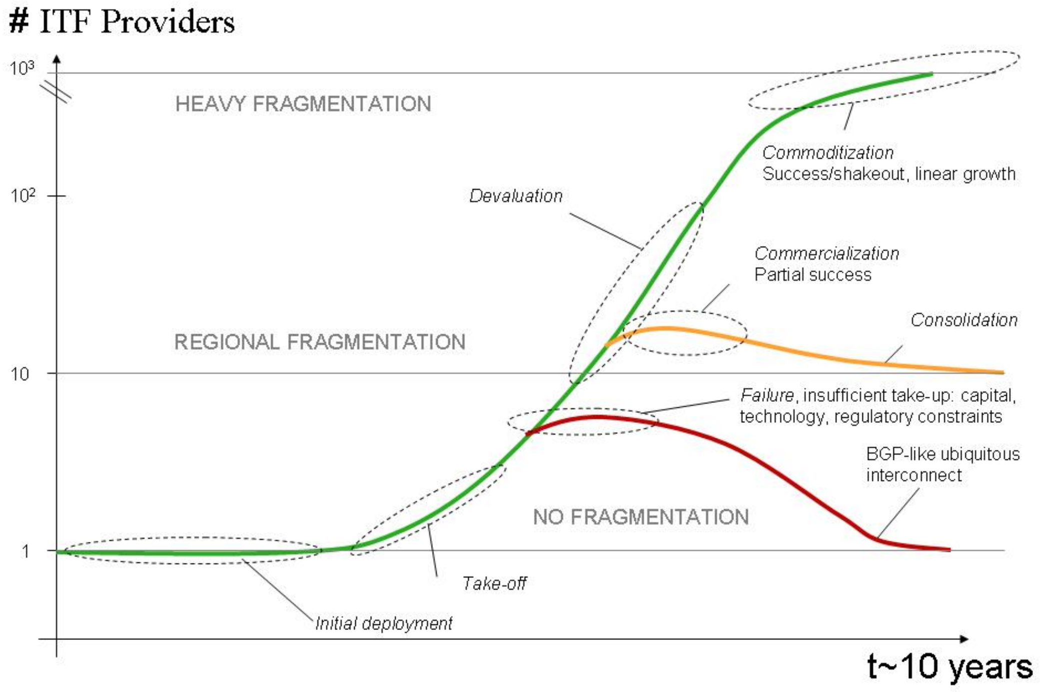


Figure 5.5 – Reference mode for number of ITF providers

With regard to the interplay between RP, ISP and publisher wishes when choosing topology, different views will dominate depending on the prevailing market and regulatory conditions (e.g., balance between legal requirements on access/transit peering and packet information-content):

- Publishers will normally be motivated by a desire for link differentiation (e.g., QoS, access regulatory compliance etc.) and may well tend to mistrust ISP behaviour, perhaps based on historically negative experiences when the peering market has been heavily biased towards providers.
- An ISP will be similarly motivated by link differentiation (as the agent legally responsible for access/transit peering regulatory compliance) and may well bias any decisions strongly towards its own interests (e.g. protecting network infrastructure vs. publishers, topology hiding vs. competitors and favouring cached content nearer to itself for resource optimisation).
- A RP will desire link differentiation with respect to *information-related* regulatory compliance (as the agent legally responsible), seeking "best" long-term compromises to any "tussle" between RP, publisher and ISP views, probably favouring cached content nearer itself rather than individual ISPs

The functional behaviour (S-curve) of *RP, ISP and publisher importance-related* stocks will probably be very similar (in terms of their evolution over time), so Figure 5.6 concentrates on the local ISP reference mode as a typical example. *Importance of ISP choice* is heavily affected by trends in regulation and the degree to which initial enthusiasm for the service (in terms of hype/coolness etc.) wanes in the face of price pressures.

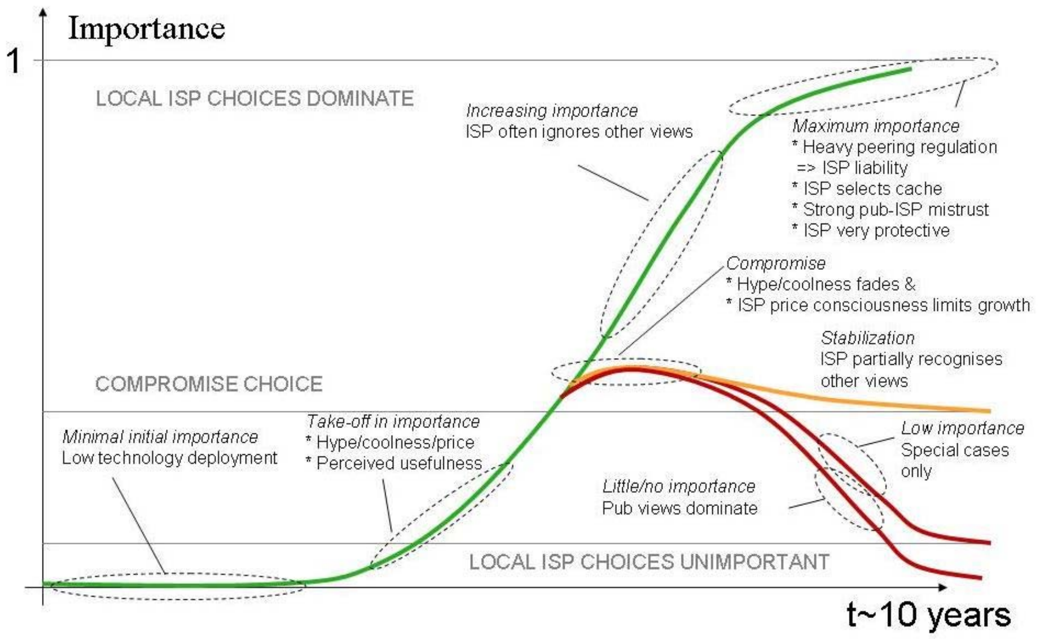


Figure 5.6 – Reference Mode for the Importance of local ISP choice

5.3.5 Summary

This overall approach defines and prioritises the functionality required by users, which must be designed into PSIRP architecturally (design choices reflecting design goals) to provide incentives for operator deployment. Analysis might include any/all of a variety of factors (many, non-technical) such as:

- Technology
- Policy
- Competition
- Regulation
- Investment hold-ups
- Business risk/uncertainty

The CPC is a specific technical implementation utilising the ITF architectural conceptual components described above, with triggers and scenarios defining the stocks/flows for the System Dynamics scheme. Causal loops capture dynamic control relationships (e.g. feedback mechanisms) between the various system components, while reference modes provide a "reality-check" on model structure.

PSIRP socio-economic work to-date has thus served to fully define the ITF modelling problem for detailed simulation within the system dynamics environment in the next phase of the project.

6 Conclusions

This deliverable provided an update of crucial parts of the architecture work. It is not an updated architecture, as compared to D2.3, rather than an elaboration of ongoing work that is directly based on our work presented in D2.3.

Such elaboration of architectural work was the presentation of the algorithmic identifiers work. This type of algorithmically determined rendezvous identifiers was originally presented in D2.3 as a promising approach to address and implement crucial functions in the network, such as fragmentation, but also provide an interesting semantic grouping of information items for applications. The work presented here highlighted various techniques to implement these promises but also shed more light of potential problems and constraints when putting this technique in place.

An important part of this elaboration is the presentation of our security architecture work. Aligned with our overall project goal, the security design team has taken the various concepts of the overall architecture, as expressed in D2.3, and has developed a growing security architecture around this. In particular techniques for rendezvous and forwarding security have been developed, the relation between scope owner and publishers/subscribers has been clarified, and packet-level security techniques have been outlined. This leads to a growing confidence that the PSIRP architecture cannot only build as a scalable but also secure system.

The presentation of the forwarding work outlined the advance the project made in this area, based on the published work of the forwarding design team. The advances target the advanced security of the zFilter approach but also the architectural integration with the future topology formation by enabling policy compliance enforcement through tying the RId to the resulting forwarding path. This is an important step forward to building our overall inter-domain forwarding solution.

Progress on the design for the inter-domain topology formation was finally presented as a connection to a major inter-domain component that remains to be finalized in our project. Given the variety of potential design choices, stemming from the various interests in building an inter-domain forwarding path, led us to choose an approach that follows a socio-economic rather than a pure technological path. We presented the current status of the various design choices and the first results of our system dynamics models that will eventually allow for simulating various socio-economic scenarios. With that, we hope to narrow down the various choices, giving us the confidence to develop relevant techniques in each of these choices.

7 Terminology

Application identifier	Any higher-level identifier that applications may use. It is usually mapped onto a set of rendezvous identifiers for the data items relating to such application identifier.
Algorithmic identifier	An identifier that is determined algorithmically. It is used for rendezvous and also scope identifiers to enable *information collections*, i.e., sets of information items.
Caching	Process of temporarily storing data that is expected to be useful in the future in intermediate network elements.
Component wheel	The PSIRP layer-less network "stack" design, in which a number of network components communicate internally within a node using the publish/subscribe paradigm.
Domain	A managed network that is analogous to an autonomous system.
Forwarding identifier	An identifier used to associate a rendezvous identifier with a forwarding path.
Information item	The lowest entities of data being used in the PSIRP network. An information item is associated with a rendezvous identifier (RId) and assigned to one or more scopes.
Information collection	A set of information items. An *information network* forms a specific kind of information collection, also called *scope*. Other information collections can be formed via metadata information items, which include pointers to other information items. Also *algorithmic identifiers* can be used to assemble information collections, the collection being defined by the algorithm being used to determine the individual rendezvous identifiers of the information items in the collection.
Information network	A collection of information items under a single scope. An information network is identified by a scope identifier
Inter-domain routing	Inter-domain routing pertains to data delivery in the global network, typically spanning several domains. The inter-domain routing system is configured through the rendezvous process and takes into account any inter-domain policies in effect.
Intra-domain routing	Intra-domain routing pertains to data delivery within an administrative domain. Intra-domain routing is concerned with local policies.
Metadata	Data may also have associated metadata, which includes scoping information and other useful information either for ultimate receivers or network elements. This metadata in itself is data, i.e., it is associated with another rendezvous identifier. Metadata may be provided within a publication or as a separate data element with a separate rendezvous identifier.

Network attachment	The process of obtaining connectivity with a PSIRP network.
Publication	A self-contained data unit that has been made available using the publish primitive.
Publisher	An entity that uses the *publish* primitive to make data available.
Rendezvous	Rendezvous is the process of matching publishers and subscribers according to given rendezvous identifiers and initiating the transfer of data over the network within a given scope.
Rendezvous identifier	An identifier given to an entity by the rendezvous system in order to mediate between high-level identifiers, namely application identifiers, and lower-level identifiers, namely forwarding identifiers.
Scope	Scope defines an information network, assembling a collection of informatior items under that scope. A scope is labelled with a scope identifier (Sid). It determines the part of the rendezvous system that is used by the network. The three typical low-level cases are link local, intra-domain, and inter-domain. But scope can also be used to map higher-level concepts, like social networks, onto particular parts of the rendezvous system.
Service model	The model that is used by network elements to interface with the network. The PSIRP service model includes a low-level API, a channel model and higher-level service models
Subscriber	An entity that uses the *subscribe* primitive to request certain pieces of data.
Tussle	Tussle is defined as a conflict of interests, these interests being brought forward by players within a given communication scenario, either expressed as explicit constraints, requirements or other concepts of concerns. Design for Tussle [Cla2002] offered a novel insight for the proper design of such systems, i.e., providing guidance as to how a system ought to be designed so as to withstand and even incorporate a wide range of tussles.

8 References

[Aba03] Abadi, M., Burrows, M., Birrell, A., Dabek, F., and T. Wobber, "Bankable Postage for Network Services", Proceedings of the 8th Asian Computing Science Conference, Mumbai, India December 2003.

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

[Aur00] T. Aura, P. Nikander, and J. Leiwo, "DoS-resistant authentication with client puzzles", In 8th International Workshop on Security Protocols, pages 170–181. Springer-Verlag, 2000.

[Bye02] Byers J.W., Luby M., Mitzenmacher M., & Rege A., "A Digital Fountain Approach to Reliable Distribution of Bulk Data", in IEEE Journal on Selected Areas in Communications, October 2002.

[Cas02a] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," IEEE Journal on Selected Areas in Communications, vol. 20, no. 8, pp. 100–110, 2002.

[Cas02c] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach, "Secure routing for structured peer-to-peer overlay networks", Proc. of the 5th Usenix Symposium on Operating Systems Design and Implementation, 2002.

[Cho09] R. R. Chodorek, A Chodorek, "ECN-capable TCP-friendly Layered Multicast Multimedia Delivery", UKSim 2009: 11th International Conference on Computer Modelling and Simulation, 2009.

[Cla02] D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet," in Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, NY, Aug. 2002, pp. 347-356.

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

[ElK01] I. El Khayat, G. Leduc, "A stable and flexible TCP-friendly congestion control protocol for layered multicast transmission", Proc. of 8th International Workshop on Interactive Distributed Multimedia Systems (IDMS'2001), September 2001.

[Est09] C. Esteve, P. Jokela, P. Nikander, M. Sarela, J. Ylitalo. "Self-routing Denial-of-Service Resistant Capabilities using In-packet Bloom Filters", Submitted to Re-Architecting the Internet (ReArch'09), 2009.

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

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

[Gan04] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in G Major: Designing DHTs with Hierarchical Structure," *ICDCS*, March 2004.

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

[IMDB] The Internet Movie Database, available at http://www.imdb.com/, 2009

[Jue99] A. Juels, and J.Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks", in Proc. 1999 Network and Distributed Systems Security Symposium (NDSS99), pp151–165, 1999.

[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.

[Lak04] Lakshminarayanan, K. and Stoica, I. and Shenker, S. "Routing as a Service", UCB/CSD-04-1327, UC Berkeley, 2004

[McC96] McCanne S., Jacobson V. & Vitterli M., "Receiver-driven layered multicast", Proc. ACM SIGCOMM'96, August 1996.

[Min08] K. Minami, A. J. Lee, M. Winslett, N. Borisov, "Secure Aggregation in a Publish/Subscribe System", Proceedings of the 7th ACM workshop on Privacy in the electronic society, 2008

[Oou05] Oouchi H., Takahashi K., Nagata H. & Kamasawa K., "Multi-rate Control Method Using Layered Content", Proceedings of the Symposium on Applications and the Internet (SAINT'05), 2005.

[Par01] D. Park, J. Kim, C. Boyd, E. Dawson, "Cryptographic Salt: A Countermeasure against Denial-of-Service Attacks", 2001.

[PSI09a] M. Ain, D. Trossen, P. Nikander et. al., "PSIRP Deliverable 2.3: Architecture Definition, Component Descriptions, and Requirements," February 2009

[PSI09b] D. Trossen, M. Särela, P. Botham and T. Burbridge, "PSIRP Deliverable 4.3: First report on deployment incentives and business models," May 2009.

[PSI09c] Janne Riihijärvi et al., "PSIRP Deliverable 4.2: First report on quantitative and qualitative architecture validation", May 2009

[Sop03] A. Soppera, T. Burbridge M. Nekovee, "Index-based event messaging", in Proc International Workshop on Large-Scale Group Communication, October 2003.

[Sri07] M. Srivatsa, Ling Liu, "Secure Event Dissemination in Publish-Subscribe Networks", International Conference on Distributed Computing Systems, 2007,.

[Ste00] J. Sternman, "Business Dynamics: Systems Thinking and Modeling for a Complex World", McGraw-Hill, 2000

[Ta06] S. Tarkoma, "Preventing Spam in Publish/Subscribe", Proceedings of the 26th IEEE Conference on Distributed Computing Systems Workshop, 2006.

[Urd09] G. Urdaneta, G. Pierre, and M. Van Steen, "A Survey of DHT Security Techniques", ACM Computing Surveys, 2009.

[Wat04] B. Waters, A. Juels, J. A. Halderman, and E. W. Felten, "New Client Puzzle Outsourcing Techniques for DoS Resistance", in Proc. CCS, 2004.

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

[Wun07] A. Wuny, A. Cheungy and H.-A. Jacobsen, "A Taxonomy for Denial of Service Attacks in Content-based Publish/Subscribe Systems", in ACM DEBS '07 , 2007.

[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.

[Zha07] Zhang Z. & Li V.O.K., "Network-Supported Layered Multicast Transport Control for Streaming Media", IEEE Transactions on Parallel and Distributed Systems, Vol.18, No.9, September 2007.

[Zhu01] Zhuang S.Q., Zhao B.Y., Joseph A.D., Katz R.H., Kubiatowicz J.D., "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination", in Proceedings of the 11th International workshop on Network and Operating Systems Support for Digital Audio and Video, 2001.