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

DELIVERABLE D4.2

First report on quantitative and qualitative architecture validation

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	First report on quantitative and qualitative architecture validation
Date of preparation	29.04.2009
Author(s)	Janne Riihijärvi (RWTH), Dirk Trossen (BT), Giannis Marias (AUEB), Trevor Burbridge (BT), András Zahemszky (LMF), Jukka Ylitalo (LMF), Dmitrij Lagutin (HIIT), Konstantinos Katsaros (AUEB), George Xylomenos (AUEB), Jarno Rajahalme (NSNF), Kari Visala (HIIT), Mikko Särelä (LMF), Borislava Gajic (RWTH), Christian Esteve (Unicamp), Somaya Arianfar (LMF), Pekka Nikander (LMF), Teemu Rinta-aho (LMF), Jari Keinänen (LMF), Kristian Slavov (LMF)
Responsible of the deliverable	Janne Riihijärvi (RWTH) Phone: +49 2407 575 7034 Email: jar@mobnets.rwth-aachen.de
Reviewed by	Dirk Trossen (BT), Janne Riihijärvi (RWTH), Jarno Rajahalme (NSNF), George Xylomenos (AUEB), George Polyzos (AUEB), Petri Jokela (LMF), Giannis Marias (AUEB)
Target Dissemination Level	PU
Status of the Document	Completed
Version	1.0
Document location	http://www.psirp.org/publications/
Project web site	http://www.psirp.org/

Revision History:

Revision	Date	Issued by	Description
0.6	2009-04-27	Janne Riihijärvi	Preliminary MS Word draft
0.7	2009-04-27	Dirk Trossen	Added conclusions
0.71	2009-04-29	Janne Riihijärvi	Added Section 4.1 content from LMF
0.9	2009-04-29	Janne Riihijärvi	Integrated review comments
0.91	2009-04-29	Dirk Trossen	Final integration of comments
1.0	2009-04-30	PMC	Approved

This document has been produced in the context of the PSIRP Project. The PSIRP Project is part of the European Community's Seventh Framework Program for research and is as such funded by the European Commission.

Table of Contents

1 Introduction

The PSIRP project is an EU FP7 funded project with a 30 month lifetime. Its ambition is to investigate major changes to the IP layer of the current Internet, finally replacing this layer with a new form of internetworking architecture. Evaluating solutions for such ambitious goal is a challenge in itself. This document presents our initial results of addressing this challenge. More specifically, we report results from the qualitative and quantitative evaluation activities related to the PSIRP architecture.

Following our evaluation plan laid out in deliverable 4.1, evaluation activities have been divided into two categories. Qualitative evaluation focuses on validation in terms of both security and socio-economic aspects, whereas quantitative evaluation consists of evaluation of the behaviour and performance of the architecture as a whole as well as selected individual components in a number of scenarios of different scales. The basis of the evaluation work consists of the architecture design described in deliverables D2.2 and D2.3, as well as the prototype implementation resulting from activities described in deliverables D3.1 and D3.2. Since the architecture work has very wide scope and design of the various components has reached different levels of completeness, the evaluation work described here has not attempted to cover all the aspects of the documents mentioned above. Instead, focus has been placed on key technologies and solution candidates to assist in the architecture design process.

The remainder of the document is structured as follows. In Section 2 an overview of the socio-economic validation activities is given. Given the still preliminary status of this work, its presentation remains on a slightly higher level. In Section 3, the security related evaluation of the PSIRP architecture and various individual components is discussed in detail. Results from the quantitative evaluation activities are then given in Sections 4 and 5, with the first of these sections focussing on various technical solutions and evaluation methods, and the latter covering the evaluation of the current rendezvous system design. The rendezvous evaluation is discussed separately not only due to its key importance to the project but also because the evaluation carried out will be used as a foundation for subsequent inter-domain level evaluation activities. Section 6 finally concludes the deliverable.

2 Overview of the Socio-Economic Validation

This section presents the current results of the socio-economic evaluation of major PSIRP architecture components and aspects. We first outline the objectives of the evaluation before presenting the used methodology. The current status of applying this methodology forms the majority of the section before describing the next steps of our work.

2.1 Objectives of the Evaluation

Designing an inter-domain architecture with the scope and objectives of the PSIRP project, i.e., widely replacing the currently deployed IP technology with something new and largely different (in operations and technologies), is likely to cause a similar (radical) change in socio-economic aspects. Examples for these changes are the creation of new markets, the emergence of new players, new ways of creating and maintaining communities, and the development of new and radically different business models.

While we appreciate that we cannot and will not shed light on all of these questions within the lifetime of the project, our work intends to start the necessary understanding in this space. For this, we intend to focus on major *architectural components* of PSIRP, components likely to impact the abovementioned questions. We further intend to *devise a methodology* of understanding and evaluating a large-scale (technology) design, such as the one proposed in PSIRP, from a market and value chain dynamics perspective. Both, the results of the evaluation as such as well as the methodology used, are the major objectives of our work.

Concretely, our results presented in this deliverable focus on the *rendezvous* component of our architecture, more specifically the solution presented in [Psi2009]. A simple use case is applied to examine potential market and business questions in a set of to-be-developed system dynamics (SD) models. These models are expected to enable us defining and evaluating market scenarios, identifying new players to emerge due to the design chosen and devising example business models for potential players in these markets. The focus in this deliverable is on the first two aspects, while business models are left for future considerations.

We must however note that the work is still in its infant steps. Most of the work shown in this section focuses on the methodology and the current results. The evaluation is not (yet) at the stage of providing insightful answers into market questions. This will remain our focus in the near term future work.

2.2 Methodology Overview

This section outlines the methodology (also referred to as *value chain analysis toolkit*) that is used for the following examination. It is directly based on the one laid out in [CFP2008], aiming for the development of SD models. However, we clarified a number of steps and geared the methodology more directly towards the development of SD models, while the original CFP model targeted a wider initial (qualitative) understanding of the created value chain.

The methodology is devised in so-called **steps**, which an *evaluator* (i.e., a person chartered with the evaluation of a particular design and proposition) follows. It is important to understand however that the steps only provide guidance. Certain steps can be omitted or executed in different order, e.g., in order to merely aim for SD models (for which certain steps are not directly necessary) rather than target the development of business models. Each step is elaborated in the following subsections based on the chosen rendezvous example, i.e., explaining the methodology itself but also the findings with respect to our evaluated architecture component.

To begin, we refer to Figure 1 which presents the overall methodology in a mind map format. We use mind map formats throughout the presentation of the following steps as a form of

presentation—the original methodology in [CFP2008] used a loose set of presentation slides as a guide through the steps. We felt that a more focused (mind map) type of execution of the steps is beneficial in conducting the evaluation.

The methodology is divided into two major parts. The first one aims at specifying and dissecting the case at hand, increasing the understanding of control points, value and influences that occur in the wider value chain. It forms a first stage of (qualitative) evaluation that is crucial in the development of analytical (quantitative) models for evaluation. These models are developed in the second stage of the methodology, which are not part of this deliverable. The results of the first stage directly feed into this construction of (SD) models. We outline at the very end of this section how our current results laid the foundation for future work in this area.

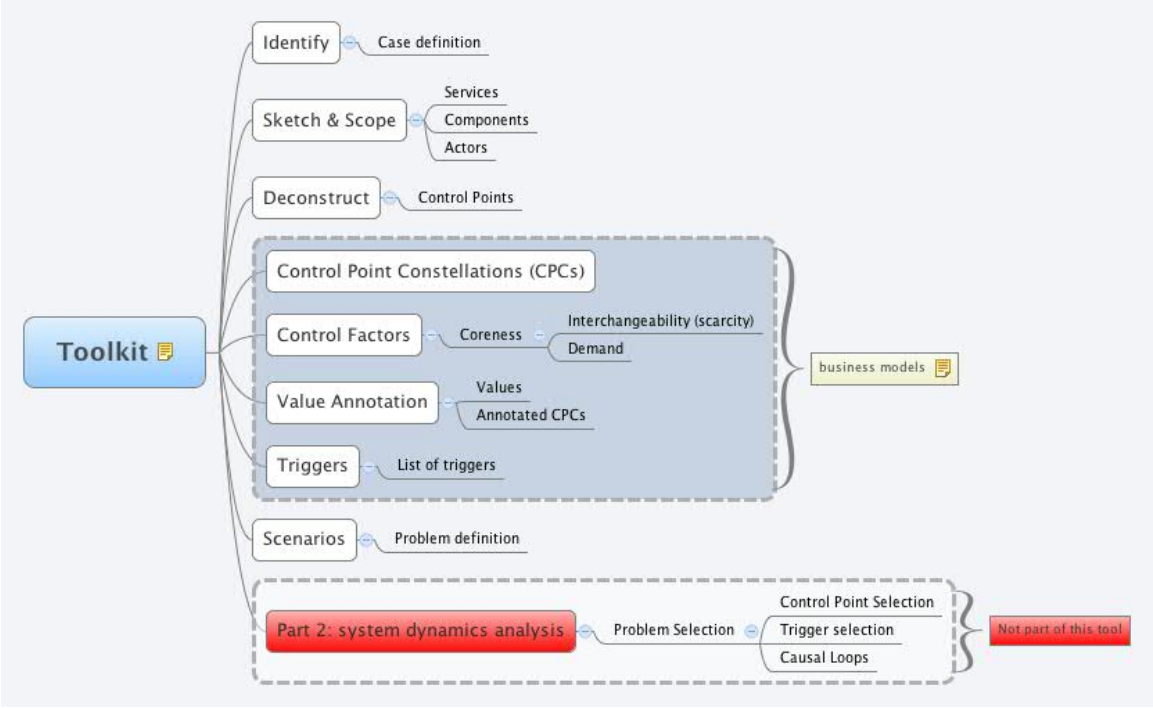

Figure 1: Methodology: Value chain analysis toolkit.

We now refer to Section 2.3 and its subsections to further explain the steps of the first part of our methodology alongside the conducted use case for the rendezvous component evaluation.

2.3 Current Steps of the Methodology

The following section presents the status of the conducted steps of the methodology. We use the evaluation of the rendezvous solution, as presented in [Psi2009], as an example for also outlining the methodology itself.

2.3.1 Identify

The purpose of the **Identify** step is to focus the evaluation towards a very particular use case, easing the application of the following steps. It is not uncommon that several use cases are evaluated for a single proposition or architecture components. Each of these use cases would apply the steps of the methodology, although resulting SD models can be joined up at later stages.

Core parts of the step are the description of the use case at hand, to be written in plain text. Furthermore, *assumptions* are outlined. These assumptions serve the purpose of narrowing down the field to be considered, i.e., the technical components to be considered. Assumptions are to be chosen in a way that the evaluated market is seen to be largely independent from the omitted pieces of the evaluation. However, one can also imagine evaluating the particular cross-effects through dedicated use cases later, i.e., focusing on the particular aspects of the original omission. The *focus* part outlines the intended scope of the use case in plain text. This gives the reader an idea of the mind set and intention of the evaluator when conducting the evaluation.

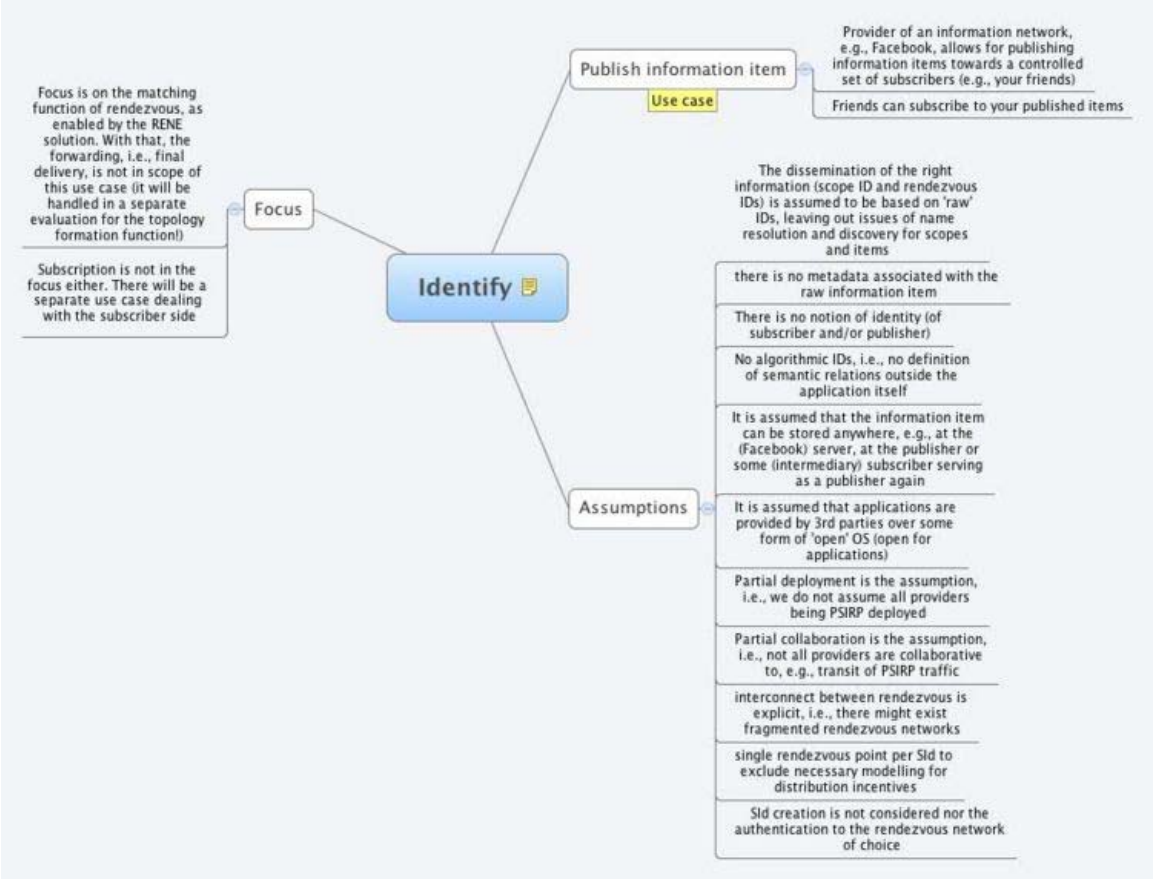

Figure 2: Identify step.

Figure 2 presents the application of this step for the rendezvous case. We chose a very simple use case of matching when an information item is published to a particular information network (see [Psi2009]), e.g., a user publishing a picture to a social networking site such as Facebook today, with the intention that others (e.g., his friends) could subscribe to these items. Hence, the focus is solely on the rendezvous component of the PSIRP architecture [Psi2009], largely omitting aspects of (inter-domain) forwarding. While outlining subscribers in the following, the subscription process is initially left out, too.

We define the following assumptions for our use case.

- We leave out aspects of discovery and resolution of information items and networks (scopes). We intend to focus on the potential combination of rendezvous and discovery solutions in later use cases.

- Aspects of metadata and algorithmic identifiers [Psi2009] are left out to eliminate control points around metadata and algorithm definition, standardization, and management.
- In order to eliminate any particular considerations for the end device market (e.g., application bundling to mobile devices), we assume 3rd party application deployment on end devices within some form of open (for applications) operating system. We however recognize that such potential bundling is an important aspect in today's market place. We therefore include the aspects in the control point considerations, but we will leave them out of original SD model development (with the potential to easily include them at a later stage).
- We assume storage of information items to be independent from the actual act of publishing, i.e., the use case is not an exact replication (within the PSIRP architecture) of today's social networking sites, which usually upload information items to the site, apart from performing the matching of publisher and subscribers. This will allow for a separate consideration of storage providers (including the end user directly) at later stages.
- Aligned with our rendezvous design considerations, partial deployment of the PSIRP components as well as partial collaboration of forwarding providers is assumed. This is important for market considerations and the overall viability of the architecture.
- We assume that the interconnection of rendezvous networks is explicit, i.e., there exist explicit policies to interconnect rendezvous networks through an *interconnection overlay* with the possibility to have several of these interconnection overlays. This lends itself to the consideration of fragmentation scenarios.
- A single rendezvous point is considered for each scope to exclude (for now) distribution incentives for having different rendezvous points per SId.
- SId creation is excluded from the considerations for now, i.e., authorization to the rendezvous network is out of scope. SId creation becomes more important when combined with the issue of discovery.

The assumptions and the focus outlined above now allows for following the next steps more concisely.

2.3.2 Sketch & Scope

The **Sketch&Scope** step dissects the use case under the light of the underlying technical architecture being used for its implementation. For this, the view is taken that *actors* are providing *services* via certain technical *components* under an assumed technical architecture. The reason for this dissection is threefold. Firstly, it allows for defining the *control points* (see Section 2.3.3) within our use case. Secondly, it allows for defining *value annotations* (see Section 2.3.6) and therefore for investigating the potential value flow within the value chain. Thirdly, it allows for defining actor-specific business models for a subset of services being provided with the value chain (not considered in this deliverable).

Figure 3 presents the sketch&scope step for the rendezvous solution.

The presented components can directly be derived from the current PSIRP architecture, specifically the rendezvous solution, as presented in [Psi2009]. The services are divided into the act of publishing and subscribing. Different services within the architecture are provided by different actors to others. For instance, a high-level publishing service is implemented between user and rendezvous point (e.g., Facebook). This service is implemented as a sequence of publishing services between the intermediary actors (see also Section 2.3.4). Finally, the actors focus on the particular case at hand, i.e., we differentiate an interconnection overlay (IO) provider from a rendezvous network (RENE) node provider (i.e., a node provider within a particular rendezvous network that is interconnected via an interconnection overlay). In addition, a rendezvous point (RP) provider is assumed as an actor. At the transit level, a variety of local ISPs are assumed (for the publisher, the IO provider, the RENE node provider

as well as the RP provider), while transit ISPs are folded into a single actor. The reason for dividing the local access ISPs into the different ones is that local ISPs might have different incentives for bundling functionality, such as RENE node functionality, into their local ISP offering.

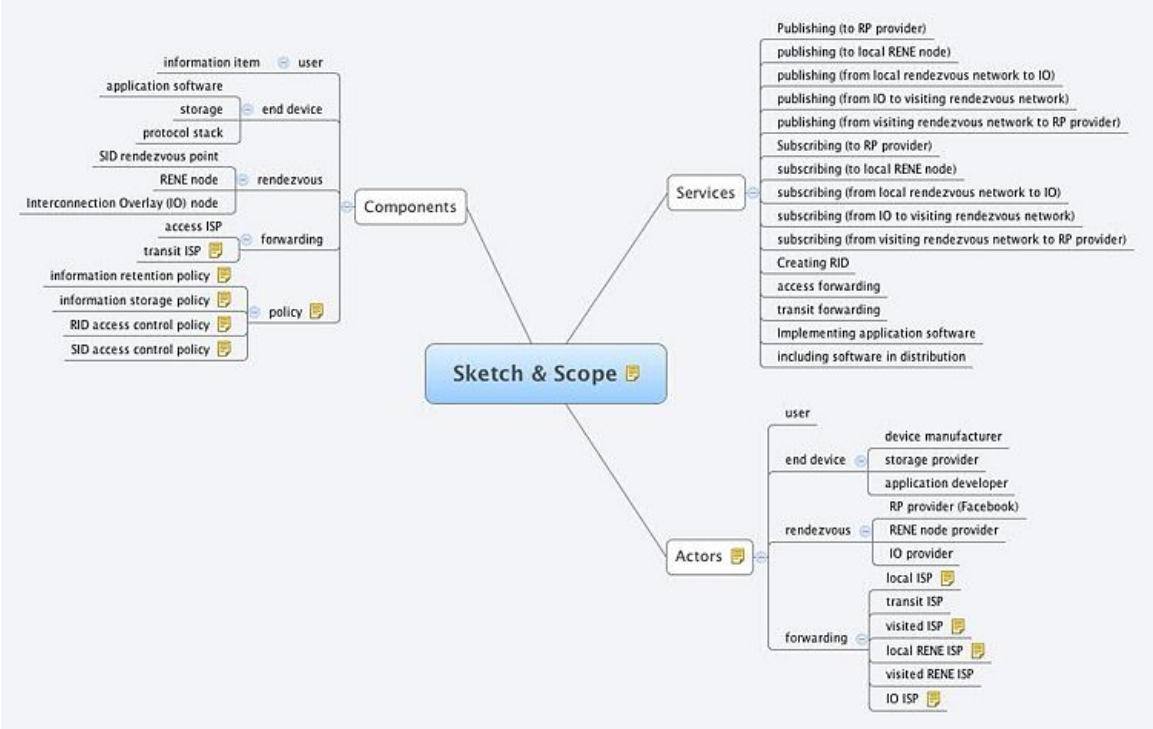

Figure 3: Sketch & Scope step.

At the device level, we assumed a simple division of storage provider, application developer, and device manufacturer. With this, storage providers are assumed to be edge-based, focusing on high-level information item storage, excluding for now any lower-level temporary storage by network components.

2.3.3 Deconstruct

In this step, we deconstruct the use case into a set of control points under several dimensions beyond pure technological ones. For this, we use the actors, services, and components information from the previous subsection.

But first, let us define a control point as

Definition: A *control point* is a point at which management can be applied. Control points can be rooted in business, regulatory, or technical regimes.

It is obvious from this definition that control points do not only reflect technical components (the so-called *functional* control points), although they will eventually be implemented by the technical components and the underlying architecture(s). Hence, the control points are usually a superset of the (technical) components, including additional *non-functional* control points.

Control points usually hold some form of value, which we will address in the control factors and value annotation steps (Sections 2.3.5 and 2.3.6).

The deconstruct step forms an important step towards the development of SD models for the dynamics of the underlying value chain, since the control points, holding a particular value, are under constant stress—this stress being imposed by triggers (identified in Section 2.3.7) and expressed in the SD models, based on certain characteristics of particular scenarios.

Figure 4 presents the control points for the considered rendezvous use case.

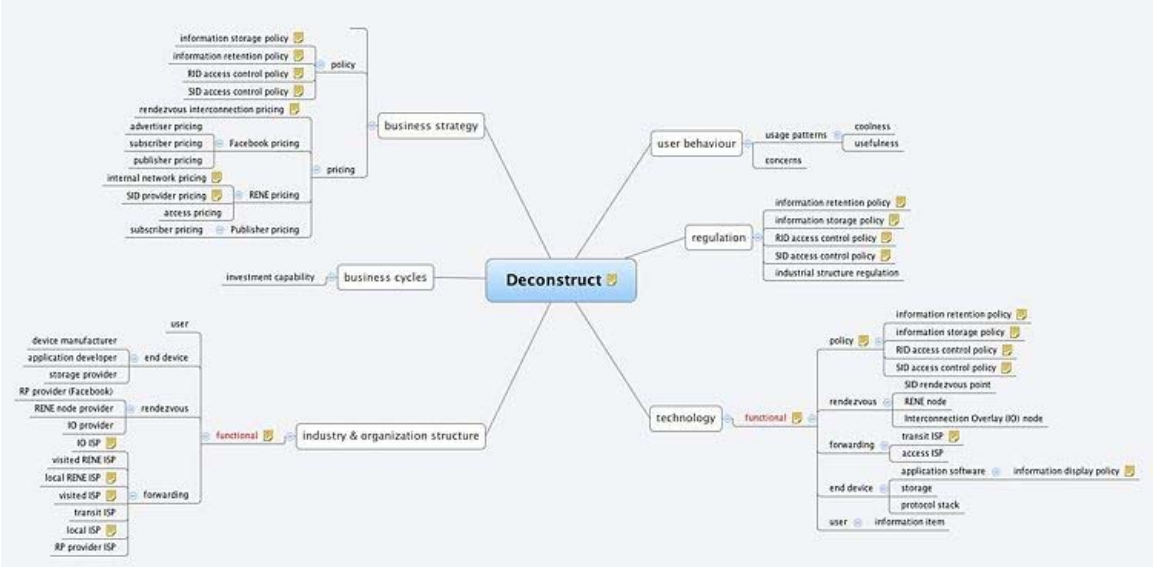

Figure 4: Deconstruct step.

One can see that the actors and components of the sketch&scope step map onto functional control points along the dimensions of *technology* as well as *industry & organization* structure. There are a number of control points with regards to *business strategy*, mainly around different pricing opportunities (e.g., advertiser or publisher pricing). The *user behavior* control points reflect early adoption related ones, such as coolness and usefulness, but also concerns of different kind (not listed specifically). The *regulation* related control points mainly act on the different policies of the underlying architecture, enabling different specific markets. For instance, particular regulation on RID access control policies would control the (regulated) openness of the service offered to end users by the social networking site. Last but not least, investment capability in the dimension of *business cycles* is an important control point, given the often cyclic character of long term investment decisions.

2.3.4 Control Point Constellation

A control point constellation (CPC) represents a particular (technical) implementation of the considered use case, using the functional control points as defined in the previous step (i.e., within the dimension of technological and industry structure control points). The CPC is constructed based on some assumption of the technical architecture that defines the implementation of the particular CPC. The underlying technical architecture ought to be able to implement many/certain CPCs (each of which includes a number of business models per player) although they can be specific to a particular (narrow) architecture.

CPCs are used in the methodology to describe the assumed technical implementation of a given use case under a particular underlying architecture. They give a first outline of potential value flow in the given architecture and assist the creation of SD models by outlining the functional part of the architecture that is considered.

The diagram uses (mind map) relationships to show the services, i.e., they map onto the sketch&scope step in Section 2.3.2. Graphs can intersect, building networks rather than sequential service transactions only, in cases where service transactions happen in parallel.

It is important to note that a CPC is not a business model in itself. It is merely a technical implementation of the underlying architecture, i.e., a representation of a value chain in which several business models can be implemented. It therefore includes a variety of business models in the sense that a business model is embedded within a particular CPC as an attempt

of players to extract value in certain control points. These attempts of extracting value at certain points can lead to conflicts/tussles, potentially making the CPC, i.e., the technical implementation, break at certain relationships (i.e., service transactions).

With this mind, it is crucial to understand that the CPC step does not focus on the individual business models but the ability of an architecture to implement certain cases and therefore individual business models as well as the ability to accommodate the value generation of the players involved.

Usually, there can be one or more CPCs. One CPC could, for instance, focus on different markets emerging from one particular technical solution, i.e., how value shifts from one or more players to others. Several CPCs could, for instance, focus on how players position different technical solutions for market creation, devising strategies to shift from one business model (enabled by one CPC) to another (in turn enabled by the other CPC).

Figure 5 shows the CPC step of our rendezvous example.

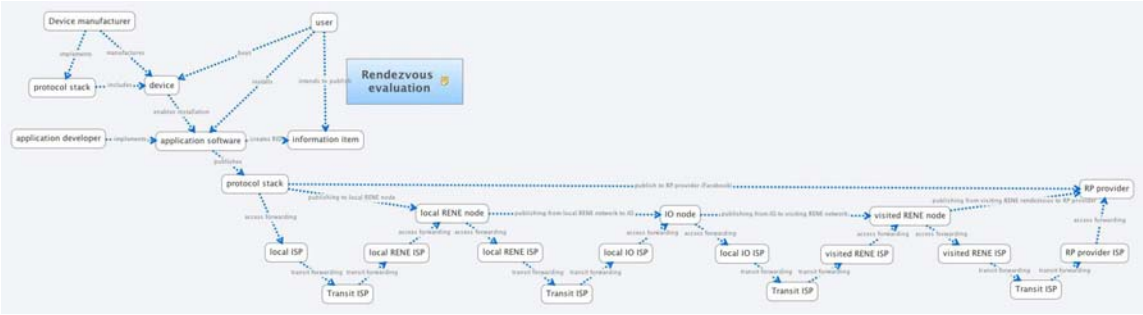

Figure 5: Control Point Constellation for publishing information item.

While the left side of the diagram focuses on the end device part for potential future consideration (e.g., bundling of services with end devices), the right side of the diagram shows the service transactions between players in the rendezvous solution of PSIRP. For more information, refer to [Psi2009]. The sequence of transactions on the right side shows the transit of different publishing actions between the involved ISPs.

2.3.5 Control Factors

The step of assigning **control factors** to the (functional) control points of Section 2.3.3 serves the purpose of qualitatively evaluating the potential performance of certain control points as to

- *interchangeability*, i.e., the ability to interchange a particular actor implementing this control point with another one in the market place, and
- *demand*, i.e., the ability to capture a certain demand of service consumers.

This consideration is important when devising business models since it (qualitatively) assesses the different control points with regards to their ability to extract value for a particular actor, focusing on the implementation of this control point.

It is important to understand however that this step merely attempts to narrow the scope of consideration for business modeling by understanding the potential of the various control points. If (a) business modeling is not the focus of the consideration (e.g., in cases where larger market understanding is the focus) or (b) the potential control points are well understood, or even fixed (e.g., by technology or places in current value chains), this step can be skipped.

In our example of evaluating the PSIRP rendezvous solution with respect to its market creation potential, we decided to skip this step.

2.3.6 Value Annotation

As stated in the definition of Section 2.3.3, a control point is a point where management can be applied. While regulatory control points serve the purpose of correcting or imposing particular market constellations, management in functional control points is mostly applied for extracting value (economic benefit) for the actor involved in the management of that control point. The step of value annotation captures this economic benefit in our methodology, i.e., it annotates service transactions and control points with expected or historically observed value. Such value can be expressed in direct monetary terms but also in indirect terms, such as subscription numbers.

Such value annotation serves two purposes. Primarily, it is used for the development of business models for particular actors trying to extract a particular value in their business proposition. The methodology of this toolkit then aims at evaluating the probability of success for such business model, investigating the triggers and influencing control points that potentially sustain or destroy certain propositions. Secondarily, the value annotation is important to understand potential points of contention in terms of control, i.e., which high valued control points are likely to be embattled? Such conflict must then be captured in the evaluation of triggers (see next section) and their influence on the control points.

In our example of evaluating the PSIRP rendezvous solution, we skipped this step since we did not focus on the enabled business models for the market creation evaluation. But a good understanding of the potential value is important in the development of SD models so that we will need to execute this step relatively soon.

2.3.7 Triggers

In our methodology, value chain dynamics are represented as a set of triggers acting on control points within a certain system design (underlined by the assumed technical architecture). This is similar to forces in a physical system straining a complex physical system. The current step collects these potential triggers along the same dimensions as the control points itself, i.e., beyond pure technological forces. Given the dimensions of the control points, one can expect several tens of triggers in a complex system.

Figure 6 shows the collections of these triggers for the rendezvous evaluation.

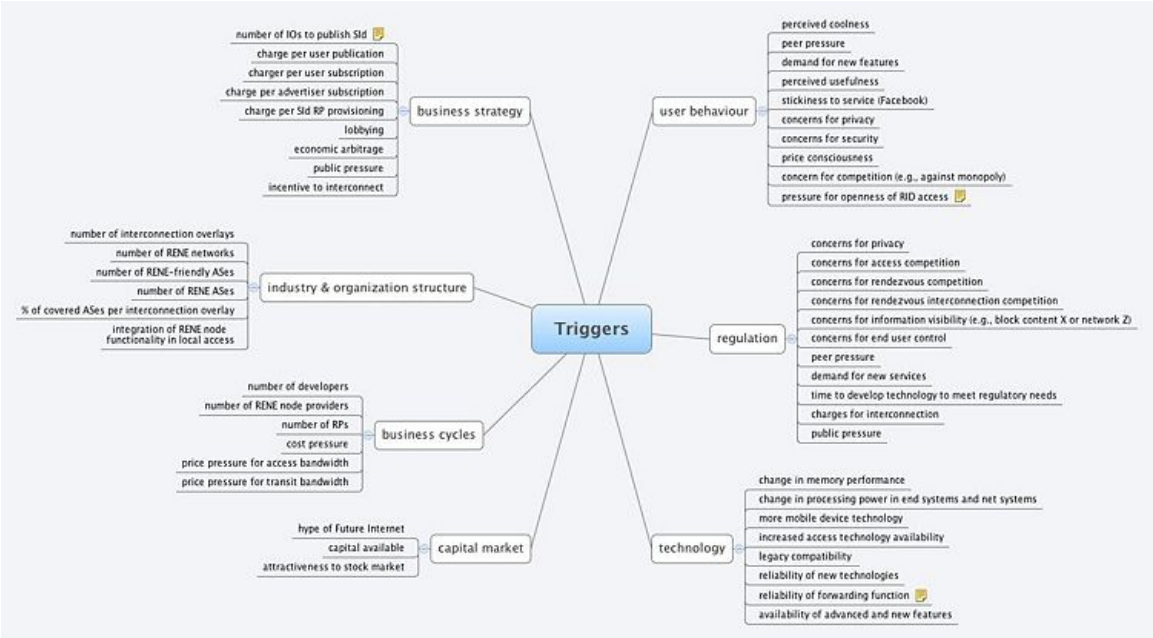

Figure 6: Triggers step.

On the *technology* side, relevant and expected advances of technology are considered, such as memory performance and availability of new access technologies. The incumbent character of the proposed architecture is considered in the legacy compatibility and reliability triggers. *Regulation* triggers address a variety of concerns like privacy, competition in various areas as well as innovation type of charters (e.g., demand for new features) that are implemented by certain regulatory bodies. On the *user behavior* side, triggers on perceived coolness and usefulness address typical early adopter and user concerns, while concerns for privacy and security are likely to influence adoption, too. Important for our considerations are also user concerns for fair competition, e.g., acting as a social counter-movement to rising monopolies in certain areas.

Important triggers on the *industry and organization structure* are the numbers for interconnection overlay and RENE node providers. This will directly feed into our SD models for the considered scenarios (see next subsection). But also the *coverage* of interconnection overlays plays a role from the perspective of rendezvous point providers, influencing the publication of a given SId in a number of overlays to increase overall coverage (and therefore reachability by the wider Internet community). Price pressure and competitor triggers constitute most of the *business cycle* triggers, while we included attractiveness of Future Internet investments (even to the point of hype) in the *capital market* triggers.

2.3.8 Scenarios

As the last step of the first part of our methodology, we need to outline scenarios for the overall value chain dynamics evaluation. Similar to the use case itself, these scenarios should be specific. A good test for these problems is the ability to implement them well in an SD model that can be analyzed. Although it seems odd that this clear problem specification comes so late in the step sequence, it needs to be understood that the steps executed so far do not necessarily target the development of SD models, while this last step clearly aims at being the basis for SD models developed in the second part of the methodology.

Figure 7 shows the current scenarios for the rendezvous solution.

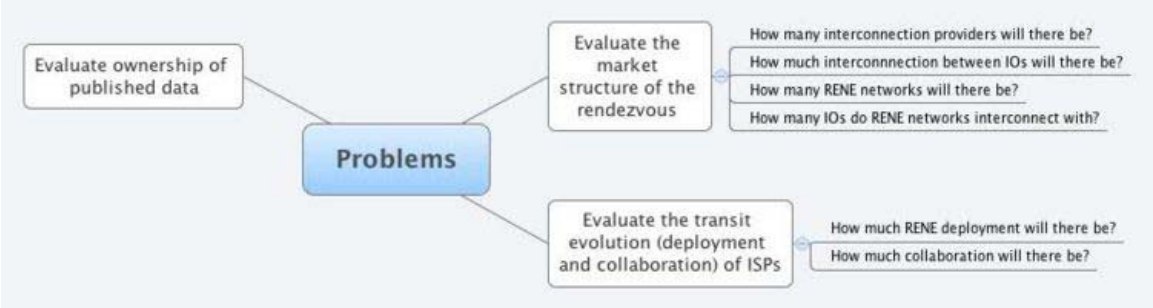

Figure 7: Scenarios step.

We can see two major problems to be evaluated with the created material:

- **Evaluate the market structure of rendezvous providers and interconnection overlays**: here, important are questions such as the number of interconnection overlay(s) providers, the number of rendezvous networks, the degree of IO interconnection, and the intended coverage of rendezvous networks when it comes to interconnection overlays. This problem area leads us to a number of potential market scenarios which we would like to shed more light on in our future work:
  - *One Google*: there is only one IO provider (the future Google)
  - *Few dominant Googles*: there are several but still rather dominant IO providers
  - *Google everywhere*: no dominant IO providers, virtually everybody interconnects with everybody else

  - *Fragmented Googles*: fragmentation of the interconnection market with controlled exposure of information networks (SIds) to the different fragments. Questions on control of this exposure are of interest.
- **Evaluate the transit evolution**: here, the evolution of the deployment and collaboration aspect in our model is of interest. In other words, how do we see the deployment of PSIRP-compliant components progressing, what drives this evolution (technology, regulation, user demand)? Furthermore, how is the collaboration between PSIRP-compliant and legacy ISPs progressing? Do we see trends of 'negative' collaboration, e.g., through blocking PSIRP traffic? These scenarios will give insight into the importance and strategy for migration.

The left side of Figure 7 hints at potential other problems of interest for future evaluation, e.g., the ownership of published data (as it has arisen in the recent UK case against Facebook in this matter).

2.4 Conclusions and Future Work

In this section, we presented a methodology and early results on applying this methodology for a particular component within our architecture, namely the inter-domain rendezvous solution. While being based on existing MIT work, in which one of the authors of this deliverable was instrumental in realizing, we significantly extended the methodology specifically towards the development of system dynamics models that will allow us to quantitatively evaluate different scenarios and problems with respect to the rendezvous solution. For this, we presented the different steps of the methodology along the rendezvous evaluation case, based on a simple publishing use case. We decomposed the problem space into services, actors and components as well as into control points and triggers acting on these control points. We furthermore outlined the problems to be evaluated in more depth within to-be-developed system dynamics models. All this is to be seen as important ground work to apply the developed methodology deeper within the current use case but also within future use cases and components.

For future work, we will continue our work laid out in this deliverable. As indicated in our methodology (in Section 2.2), the first part of the steps aims at the development of SD models for a deeper analytical evaluation of components within our architecture. The control points, triggers and problems outlined in the material presented here, directly lend themselves to the development of such models in our future work through the development of appropriate causal loops. While we focused on a simple publishing use case, it is relatively easy to extend this towards a subscribing use case but also towards other considerations, such as integration of different policies etc. These deeper SD models will be developed in the near future to deepen our understanding of the markets created by the rendezvous solution as well as devise business models from the perspective of existing and potential new players.

In addition to evaluating an already designeded component within the PSIRP architecture, we also intend to apply our methodology in supporting the design of a still-to-be-developed component. More specifically, we intend to investigate design choices for the inter-domain topology formation function within PSIRP (see D2.3 [Psi2009]). It is the aim to understand the control structures enabled by different design choices in terms of executing, as well as initiating the topology formation process.

3 Security Evaluation of the PSIRP Architecture and Components

Our security evaluation targets main concepts and components of our architecture, based on the status as reported in D2.3 [Psi2009]. This section addresses first the used methodology before presenting the analysis of individual concepts and components.

3.1 Methodology

In our security work, the focus is on evaluating security aspects of multiple architectural components, therefore protecting the system from misbehaving nodes and players. The main components consist of Identifiers, Forwarding, Node-internal Architecture, Helper Functions, Rendezvous, Topology Management and Formation, and Network Attachment. The project has been looking for countermeasures to potential security vulnerabilities. This results in a set of research problems as well as in a set of conflicting goals. We need to consider the right engineering balance at the intersection of the different security and architectural component requirements. Finally, the various kinds of security concepts and architectural components are combined together in a novel and seamless way, providing a sort of base line security for our pub/sub architecture. In this, an essential part of the research methodology is to test and verify the architecture using simulations and real code.

Additionally, several threats that have been reported in the current Internet paradigm (e.g., identity spoofing, amplification and reflection attacks, time analysis attacks, clogging and resource consumption attacks, replay attacks, flooding, etc) are analyzed in terms of their applicability to the PSIRP functional components. These attacks are studied to evaluate these components mainly for confidentiality, integrity, and availability (DoS resistance) of the information. For the purpose of our analysis, specific adversary models are assumed. Depending on the evaluation scenario and the functional component that is under evaluation, adversaries are assumed to be capable to eavesdrop a particular link on the forwarding path, transmit bogus or dummy packets to the link, capture and replay transmitted packets, modify packet fields, sign the content of packets, etc.

3.2 Forwarding

The PSIRP architecture describes forwarding as the process of sending packets along a fast-path between a publisher and one or more subscribers. The route is described by a tree constructed from forwarding identifiers (Fids) that temporarily describe a link or partial tree. It is clear from the architecture description that these FIds must be transient to prevent the sending of unwanted traffic to subscribers. That is, when the subscriber no longer wishes to receive data, one or more of the FIds that form the delivery tree must change.

The current architecture in [Psi2009] is not very specific on the operation of the rendezvous and topology formation functions during the unsubscribe operation. This is largely due to the ongoing work in this area of forming an inter-domain delivery topology. We however intend to provide some recommendations from a security perspective. An alternative of changing the mapping state between the FIds and physical links that form the delivery tree is not discussed, primarily due to the preferred choice of source controlled in-packet soft-state. In this model the source cannot be trusted to use alternative FIds in order to remove the leaving subscriber. Instead, the FId must be rendered inoperative by a party trusted by the subscriber (potentially through a chain of trust). Ultimately we expect that the subscriber's forwarding network will be trusted by the subscriber and this will be sufficient to prevent the subscriber receiving unwanted information. If the traffic is stopped earlier in the network this could be done through the trust relationship between forwarding providers (potentially through the inter-domain topology function).

To meet security requirements the unsubscribe operation should result in a change in the FIds sufficient to drop the leaving subscriber from the delivery tree immediately. Ideally the change

in FIds should result in the dropping of traffic as far back into the network as possible to relieve cost from the forwarding provider. If the same FId(s) are also used by other subscribers (for the delivery of the same or related data to an RId) then the new FIds must be communicated to the party constructing the forwarding packet (e.g. the publisher) to ensure that communication continues between the publisher and the remaining subscribers. To avoid a period of disconnection it thus seems necessary to create and disseminate the new FIds before the old ones are discontinued.

Since the forwarding network to which the subscriber is attached to is capable of changing Fids so as to drop only the departing subscriber (e.g. by removing the FId nearest to the subscriber, and replacing it for remaining subscribers, if necessary), the architecture may wish to consider that this is done immediately in response to local functions under the same authority (e.g. the local rendezvous and intra-domain topology functions for the forwarding network). This would lower the burden of trust management between the subscriber and remote rendezvous or inter-domain topology functions and result in lower latencies for the unsubscribe operation. If even lower latency for the unsubscribe operation is desired, the architecture may consider changing immediately the mapping between the last-hop FId and the link to which the subscriber is attached (or just dropping the last-hop FId if it only serves the leaving subscriber), although this does conflict with the desire of changing FId mapping further back in the network for cost reasons; a balance which needs to be studied further.

3.2.1 Protecting the Forwarding Network

Forwarding security should also consider how the forwarding network is protected from the actions of subscribers, inter-domain topology formation functions etc. It is assumed that the forwarding network is protected from the actions of subscribers through the network attachment process, which may include authentication and accounting mechanisms. Subscribers can attack the forwarding network (and therefore rendezvous and topology functions) through the subscribe and unsubscribe operations. The forwarding network may thus consider restricting such operations, or linking these operations directly to an accounting mechanism. The forwarding network may also receive requests to change either FIds or forwarding state in order to prune paths to subscribers who have left and protect the downstream forwarding network. It is not currently fully specified how such requests are made (potentially via the inter-domain topology formation function), and how these requests are authenticated and authorised.

3.2.2 Forwarding Incentives

It has also not been discussed yet what incentives could or should be offered for the correct operation of the forwarding function. The architecture mentions that subscribers may bring their own resources to the forwarding function, which we have interpreted as operating as forwarding nodes. This is not a general solution since the correct incentives must exist for forwarding networks serving end subscribers. Also it is not clear that a subscriber will trust another subscriber to operate part of the forwarding network for their traffic, and how the presence of such untrusted forwarding nodes might impact signalling and how they could be controlled through the topology formation functions.

3.2.3 Packet Layer Authentication

Along with ending valid forwarding paths (when subscribers do not need them anymore) through the alteration of FIds, the network architecture also provides protection through Packet Level Authentication (PLA). Although the forwarding path to a subscriber may be valid the network may still reject the packet if it fails authentication or authorisation checks.

In the following we present some potential attack scenarios against PLA functionality. Throughout the discussion the reader can refer to Figure 8 for illustrations.

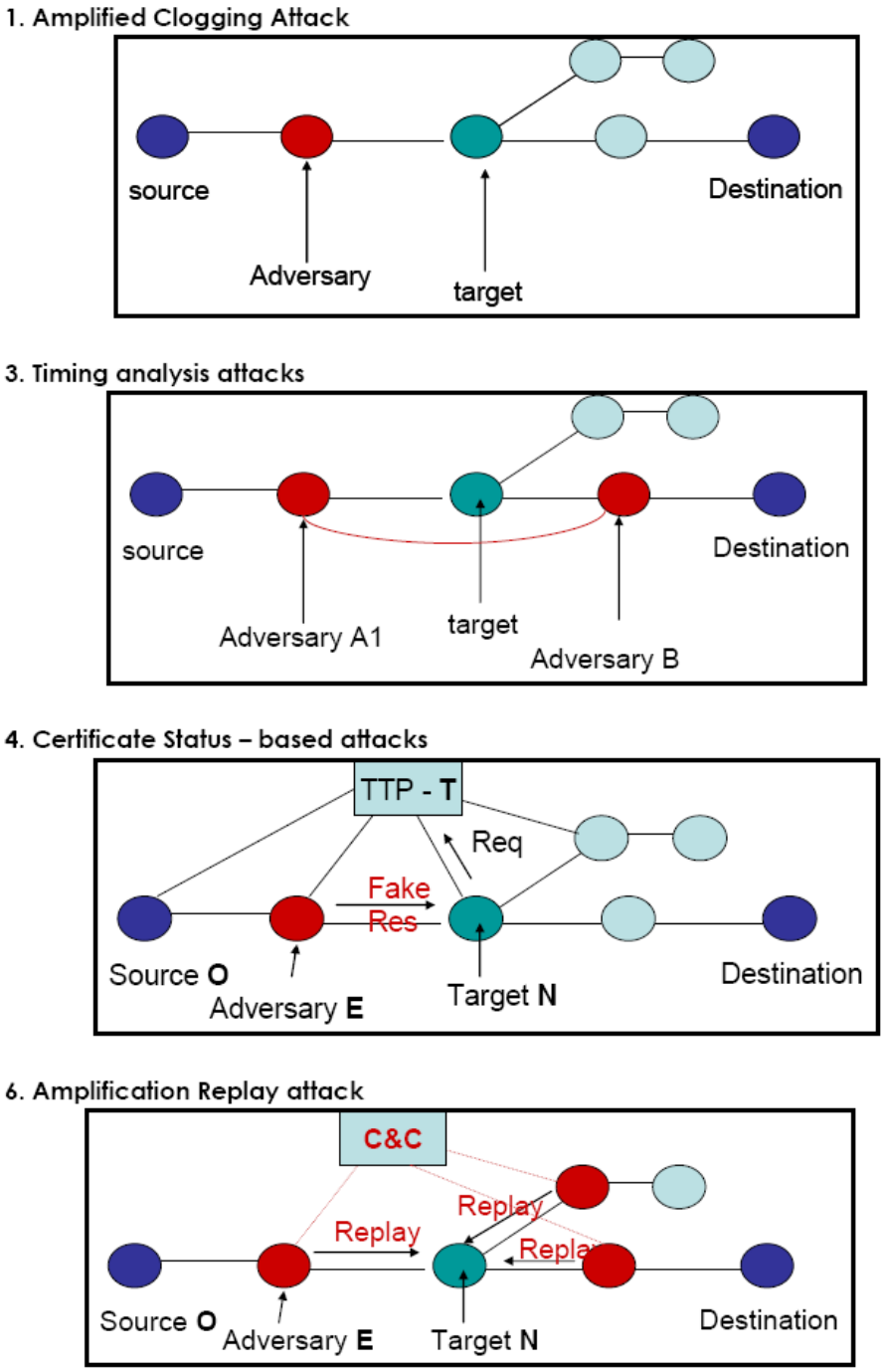

Figure 8: Illustration of various potential attacks towards PLA.

I. Amplified Clogging Attack: This attack can be deployed by an adversary that eavesdrops and has access to sent packets in the link that forwards packets towards the receiver – target (forwarding link). The adversary might:

1. Capture a valid PLA header and attach it to a new, fake packet
2. Produce a sequence of fake packets with fake authentication info (signature does not verify the content)

3. Produce a sequence of bogus packets with worthless payload and valid authentication info (signature verifies the rubbish content)

In collusion with other nodes it is possible to amplify these attack scenarios, as we explain below. The effectiveness of these scenarios depends on how many packet verifications per second the receiver can support, the receivers' buffer limits and the number and transmission rate of the adversaries. If there are only a few adversaries that maintain access to the links of a particular authentication point or receiver, then this attack might be mitigated through some verification calculations without buffer overflows. But is it possible to amplify or aggregate these attacks? The answer is yes, in particular for the 3rd scenario.

In the current Internet the adversary can determine the unique identifier (e.g. IP address) of the target. It informs a Control centre, which in turn commands several nodes to realize a clogging attack against the target (probably using different forwarding paths). This amplifies the attack at a scale that is proportional to the number of colluding nodes (i.e., the cardinality of the attackers set). If we assume that the target is a central "sprouter", then the impact might be considerable. In the PSIRP architecture, such an attack would be harder since each attacker would require a valid forwarding path to the target. Distributed adversaries cannot share valid paths with each other, and thus must acquire there own valid forwarding path from observing local traffic (or applying through the network rendezvous and topology functions). In doing so it would be hard for each adversary to determine that the forwarding path intersected the same target as its peers.

As previously mentioned, only the third attack scenario can be amplified effectively. This is because nodes running PLA towards the edge of the network will not discard the bogus packets, since they verify their integrity. On the other hand, scenarios 1 and 2 cannot be propagated in the network, since access nodes (at the edge of the network) will discard these two types of the fake packets. However, in the case that multiple adversaries are located on the same segment or shared link (for example with very large reach passive optical networks) then these attack scenarios might be credible.

II. Spoofing and Loop-back attacks: Let us assume a packet with proven integrity. The attached certificate is valid and the signature is valid as well. The packet includes an unused layer 3 address (e.g., a layer 3 unique identifier). What happens when this packet reaches a target? The target node either drops the packet, relays the packet and forwards this packet on a specific interface, or it broadcasts the packet to any interface? The last option sounds more reasonable. In PSIRP we might not use source-addresses (like IP does). If within the packet some information exists to denote the next forwarding element in the path, or the destination address the attacker can try to forge the next forwarding id or the destination address with the address of the node under attack. This produces a loop-back. If an Fid is used to identify and resolve every element that it is used in the fast path to transfer a steam of packets, the attacker can eavesdrop, identify which Fids are used to send packets to the node under attack (target) as a destination point (end of path), and use the aforementioned scenario 2 to send unused (dummy) packets to this Fid (end eventually to the target).

III. Timing analysis attacks: Let us assume two colluding nodes, the node A before the target on the path, and node B located after the target. A sends a legitimate packet to B via the (intermediate) target and determines the delay threshold that is used by the target in order to drop packets (based on the timestamp filled in the packet).

The attacker node A uses the estimated delay threshold to send some legitimate packets to the target with a particular latency disclosure, which forces these packets to be dropped by the target. In time the target may conclude that there is congestion in the network and increase the delay threshold. This then provides an increased opportunity for the attacker to replay previous legitimate packets from other hosts.

IV. Certificate Status–based attacks: Let us assume that intermediate routers need to be aware about the status of public keys and corresponding certificates (revoked, valid, etc). Even if each certificate is configured to have a short lifetime, it is possible to use the same

certificate for the duration of a session, which might be extended to several hours (e.g., video or radio service), or even days (i.e., broadcasting a conference through the Web). This point is critical since it might be used to enforce the following DoS. Consider a node N that sends an certificate status requests to a TTP entity T in order to check the validity of the certificate C that is attached to a packet sent by the originator O. The enemy E (located close to N, e.g., in the same subnet or closed enough with respect to the range wireless access) realizes the request, prepares and signs a response that updates the status of the certificate C of the originator O to invalid (revoked). Node E signs the certificate status information (C) with its own private key, but includes in the response message the identity of T (spoofs the unique IP locator, or unique FId of T) as the originator of the status information, and sends this response to N. At the same time, entity T (the real TTP) signs the certificate status information (C) with its own private key and includes in the response message its identity T as the originator of the status information and sends this response to node N. Node N receives two messages that declare the status information of C. One is fake (from E) and one is valid (from T). Node N cannot distinguish which is the correct one. If it assumes that E's response is the valid one, then certificate C will be discarded, as well as the corresponding packet and all the subsequent packets of O that are based on the same public key of O. This will enforce the discarding of all the packets arriving to N from O. This scenario mandates each PLA node to check the signature part of the attached TTP certificate with the public key of the TTP certificate (verify the TTP signature) when it receives a packet or exchanges CSIs.

V. Non-existent TTP attack: The question here is what happens if some adversaries flood the network with packets that attach certificates signed by nonexistent or unknown TTPs. In this case, each node in the network that performs PLA functions will ask to establish a trust path towards these TTPs. Such queries might introduce significant overhead, since they might be initiated and orchestrated simultaneously by many attacking nodes. The problem here is that the networking (forwarding) nodes are fast enough to produce demands for TTPs. TTPs are the bottleneck of this scenario, since they significantly delay any single response to verify status or establish trust paths. The architecture, in particular the PLA solution, may need a solution to detect and mitigate such attacks without referring malicious requests to the TTP.

VI. Amplification Replay attack: Here the adversaries capture packets and retransmit these packets to the target. This attack might be coordinated by central adversary nodes (this is the case today in the Internet, where command and control (C&C) centres manage several bots) to amplify the attack and increase the impact. Consider the scenario where an adversary E, located near to the target (on the path or within the range of wireless access), sniffs the communications and captures one or more legitimate packets that N will eventually receive. A covert channel (sinkhole) is established towards a C&C centre.

Through this channel, E sends the legitimate packet to the control centre. The centre asks some or the majority of the zombie clients to replay the packet towards the target N. The target N receives multiple copies of the packet and verifies their validity. If the state (timestamp and/or sequence number) of the packet is still maintained in N, then all the subsequent ones will be dropped. Otherwise, only one can be forwarded to the appropriate outgoing interface, and the others will be dropped, since the first one will update the state that N keeps for the packet. Depending on the cardinality of the adversaries that are controlled by a central server, this scenario might force intermediate nodes to exhaust their networking queues.

3.3 Secure Identifiers and Access Control

This chapter describes the authorization design for rendezvous and scope identifiers and evaluates the security solution. For the purpose of security evaluation, the chapter contains more detailed description of the security design than D2.3. The focus remains on authorization while the evaluation of algorithmic identifiers is for further study.

3.3.1 Design of Authorization

3.3.1.1 Public-key-based Identifiers

Rids and Sids are described in D2.3 in the following way. "The most basic notion of PSIRP is that everything is information by virtue of identifying a bucket of data with a statistically unique label. This label is used in the process of rendezvous to create the transient relation between the publisher of this information and the subscribers to this information. For this reason, we call the identifier being used for information the **rendezvous identifier (RId)**." Moreover, "Each information network represents information in itself and therefore requires an identifier in order to be addressed. To reflect the specific purpose of information networks, i.e., to scope the reachability of information, we introduce the **scope identifier (SId)** as a specific subclass of rendezvous identifiers."

An essential reason to use public-key-based Rids and Sids is to provide an authorization mechanism for scope "owners". One promising approach is to use DONA kind of identifiers that consist of two parts: public-key and label (P:L). In other words, the same public-key can be re-used for multiple identifiers by simply changing the label part. In PSIRP, we use ECC-based public-keys that are short enough to be directly included into the identifiers (P-part). This means that the entity who has access to the corresponding private-key "owns" the identifier. In the Sid case, the entity is called Trusted-Third-Party (TTP). This TTP authorizes other public-key owners to publish and subscribe in a specific scope. The main idea behind authorization is twofold. Firstly, it prevents malicious nodes from publishing unwanted traffic in a scope. Secondly, unauthorized parties cannot subscribe to content related to a target scope. In practice, rendezvous nodes are responsible for verifying that a public-key associated to a Rid is authorized by a scope-owner for publishing anything in the scope.

3.3.1.2 Authorization of Publishers and Subscribers for Scopes

The flow chart below illustrates four roles that are publisher, subscriber, rendezvous and TTP. Initially, the publisher, subscriber and TTP create public-key pairs for themselves. Px illustrates the public-key and Px.priv denotes the corresponding private-key. TTP creates a scope and defines a Sid for it, effectively becoming the owner of this scope (it is worthwhile noting that the TTP could be the publisher or subscriber directly, in which case no ‘third’ party would be involved. In other words, the TTP must be seen as a logical component that potentially maps onto existing actors like publisher or subscriber). The Sid consists of a public-key (P2) and a label. In addition, TTP creates a security sub-scope (sec-SID) for the original Sid. The sec-SID is used to transmit authorization events related to main scope. The security sub-scope can be derived from the original Sid by adding a security mask (a bit in the first byte) to the label part. The first byte of the label part should be reserved for network management, including security, purposes.

One essential reason to have separate security sub-channel is to minimize events on the original traffic channel. On-the-path forwarding nodes may subscribe to listen to authorization events on the sub-channel but they are not interested in receiving other kinds of traffic. By using a security sub-channel it is possible to send revocation information and minimize the additional traffic overhead on the already heavily loaded nodes. In practice, the security sub-channel is used to publish certificates. In addition, the PSIRP versioning-model can be used to publish updated versions of certificates that revoke previous versions.

Initially, the TTP generates a self-signed certificate (CER2) to be able to publish anything in its own security sub-scope. A publisher that wants to provide content in a scope requests initial authorization from the TTP. The authorization can be done in several ways depending on the type of scope and network topology. In a basic case, the publisher and TTP may establish a trust-relationship via an out-of-band mechanism. The TTP can use the established trust relationship during network attachment phase to authorize the publisher's public-key (P1). In this kind of approach, the scope and the access provider belong to the same trust domain. Another approach is to implement SSH-like opportunistic authorization where TTP does not

have a priori knowledge of the publisher. In this case, TTP authorizes the publisher's public-key (P1) without an initial trust relationship. It is good to notice that typically the publisher knows beforehand the public-key of the scope because it is implicitly included in the Sid. A promising alternative is to apply reputation management mechanisms for expressing trust relationships between public-keys of publishers, subscribers and the TTP. However, the approach is for further study.

A common point within the presented trust models is that the TTP makes a decision whether the publisher is authorized or not for publishing content in the scope. The authorization is expressed with a certificate (CER1). The authorization certificate contains the scope identifier, the authorized public-key and a signature of the scope-owner. The certificate is itself a publication that has an own cer1-Rid created by TTP. The label-part of the cer1-RID is hash of Sid and P1. In this way, the subscriber and forwarding node in the network are able to subscribe to the certificate just knowing the Sid and Rid (carrying P1). TTP registers the provision of the certificate that causes the certificate to be published in the rendezvous system under the security sub-scope (sec-SID). Later on, the TTP does not need to be on-line and does not constitute a single point of failure in the system.

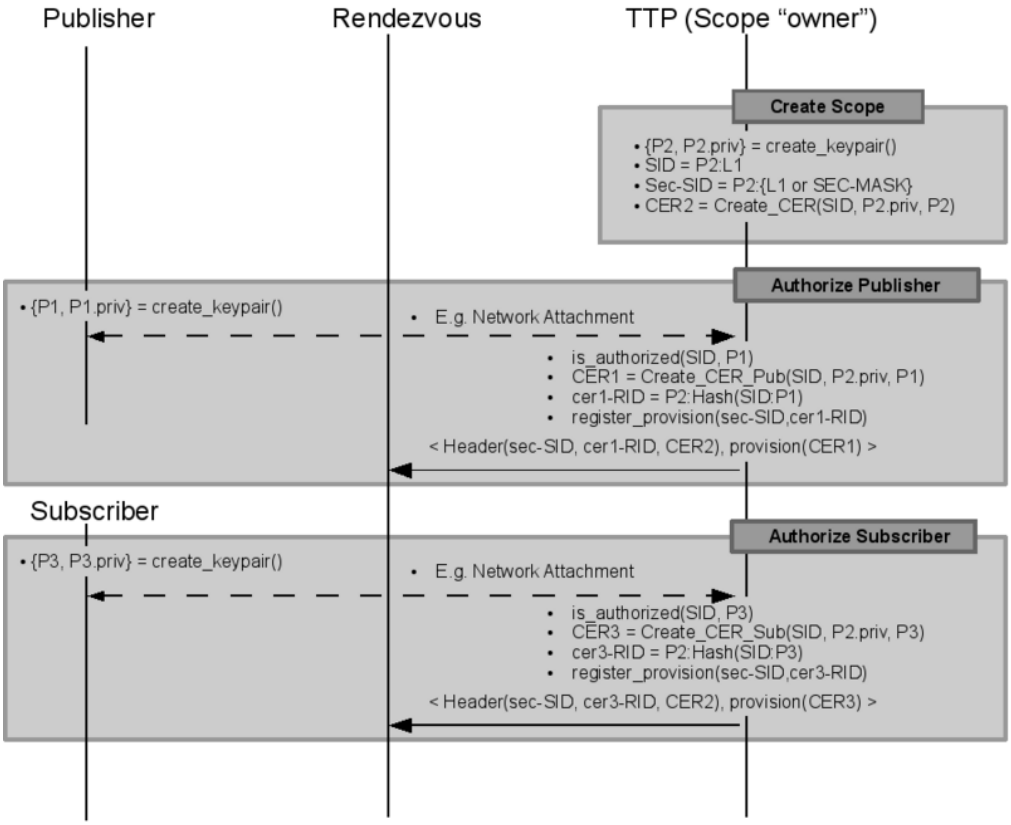

Figure 9: Trusted-third-party, i.e., scope-owner, authorizes a publisher and a subscriber.

From a subscriber point of view, authorization happens in a similar way as in the publishing case. It is good to notice that while the subscriber is a role of an entity that is willing to receive events 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 an publication. Thus, the authorization procedure for the subscriber is similar to the publisher role. The main difference is that the certificate contains attributes that defines either the right (associated with the public key) to send or receive publications in the scope. Basically, we have a set of 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 drops or processes the packet after verifying the carried signature.

3.3.1.3 Publishing Provision and Interest in Scopes

The flow chart below illustrates two roles, i.e., a publisher and a rendezvous point. Initially, the publisher creates some content that it wants to publish in the scope. For this purpose, the publisher calls 'syscall' that firstly registers an interest for subscribing certificate CER1 in the security sub-channel (in the earlier phase, TTP authorized with CER1 the public-key P1 to provide content in the scope). In the presented security model, the message containing the request for CER1 is a self-signed to avoid the chicken-and-egg problem. The main idea behind this approach is that anybody can publish an interest in the security sub-scope. However, in order to be able to publish anything in the original scope, the publisher or subscriber must obtain a valid certificate from security sub-scope. As a result, the rendezvous provides the certificate (CER1) for the subscriber that can be used to publish provision in the original scope. It is also good to notice that the public key pair associated with the Rid that is carried in the header must always be stored by the node that computes the header for packet. Finally, the publisher is able to publish provisioning for the actual content in the scope. The rendezvous verifies validity of the authorization based on the certificate included in the packet.

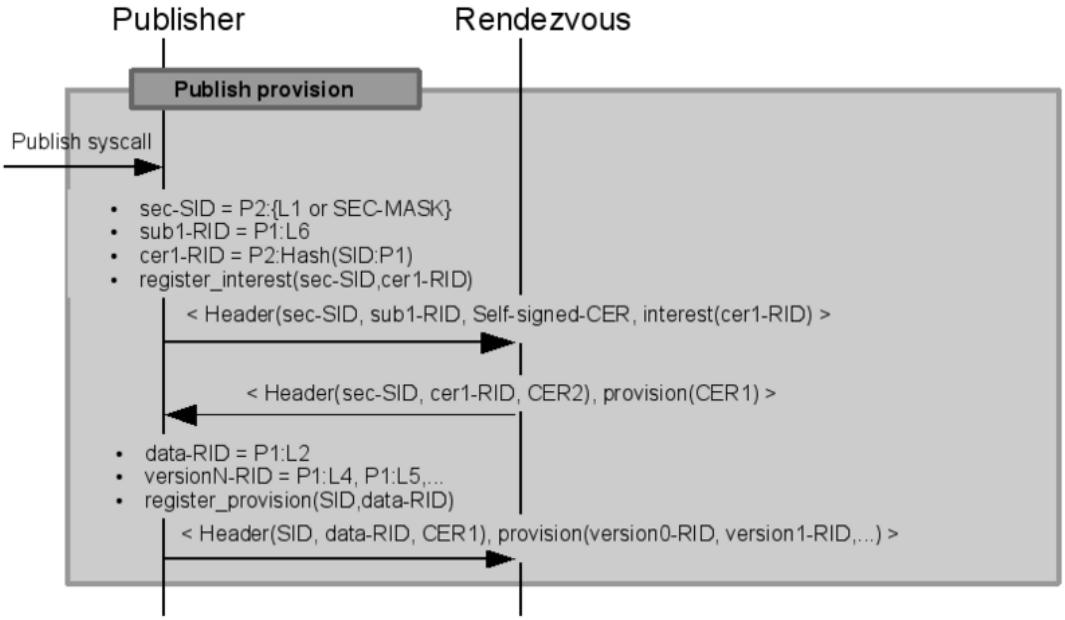

Figure 10: Publisher is authorized to publish provision in a scope.

The flow chart below illustrates two roles, i.e., a subscriber and a rendezvous point. From the subscriber point of view, the approach is almost similar with the publisher's case. The main difference is that the certificate (CER3) authorizes the subscriber to only publish interests in the scope, not provisioning. During the first round-trip the subscriber receives the certificate from the rendezvous. It is good to notice that the subscriber authorisation, like that for the publisher, may cache the certificates for future usage. The second round-trip below illustrates the procedure of receiving the content associated with subscribed (and earlier published) data-RId.

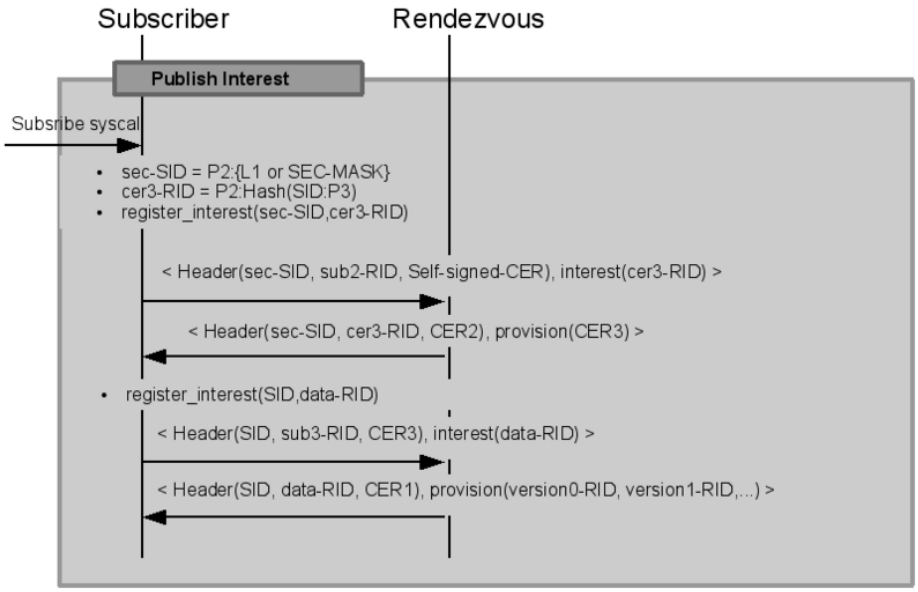

Figure 11: Subscriber is authorized to publish interest in a scope.

3.3.2 PLA-based Authorization

This section describes how the authorization can be implemented with Packet Level Authentication (PLA). 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 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.

Header format

Figure 12 describes the header format of the proposed solution. We omit other potential PSIRP header fields like header type or metadata. The first two bold boxes contain SId and RId respectively. SId and RId consist of public keys concatenated 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 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.

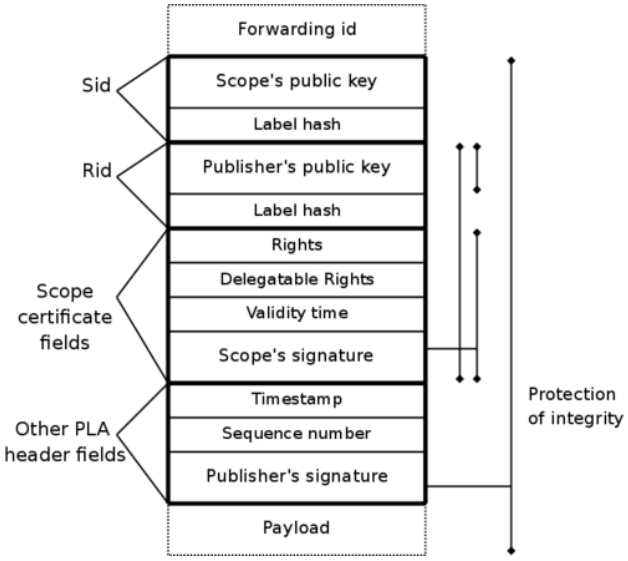

Figure 12: The overview of the header consisting of SId, RId, security attributes and signatures.

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 chosen optimizations.

1. Check sequence number and timestamp to detect duplicated and delayed packets.
2. Check the scope certificate validity time and rights.
3. Verify the scope certificate's signature.
4. 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.

There are at least two possibilities to handle revocation. The simplest solution is to use short term certificates with validity times of hours or minutes. Therefore, if the scope wants to revoke publisher's access, it will simply not renew the publisher's certificate. Another alternative is that each certificate has 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.

Other features

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

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

In some cases we want RIds to have 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 (P). The authority would authorize the publisher (P1) to utilize the RId through the certificate mechanism: (certificate (Issuer P) (Target P1) (Rights Rid)). If the publisher loses his private key, it is enough to request a new certificate from the authority for

publisher's new identity. As a downside, this would increase computational and bandwidth overhead since multiple signatures would be present in packets.

3.3.3 Evaluation

3.3.3.1 Creation of the scope certificate to authorize the publisher

One important question should be considered: should the scope owner authorize the publisher's public key or the RId? If the scope owner authorizes the publisher's public key, then the publisher can claim through the certificate that any arbitrary RId derived from his public key belongs to that scope, even if the scope itself has not specifically accepted the use of the RId. Authorizing RIds would give much more fine-grained access control, however, this would require a high amount of signaling traffic when the same publisher publishes thousands of RIds in the same scope, since the scope must give a separate authorisations for each RId. Supporting both options would be the most flexible solution even though it would complicate packet handling in routers.

3.3.3.2 Revocation of scope certificates

If the routers subscribe to certificate revocation notifications, then the security of the system will be very good, since traffic with the revoked certificate can be stopped immediately. However, this approach places a heavy burden on routers and on the rendezvous system, since the amount of different certificates passing through the router can be very high.

Using short-term unrevocable certificates creates a window of vulnerability since routers are not aware of the revocation immediately. It also places slightly higher load on the scope since it must frequently renew publisher's certificates.

The optimal approach depends on policy and types of traffic. Should intermediate routers or end-users, such as the scope and publisher, bear the overhead for certificate revocation system? If the amount of certificates is not very large, and data is transmitted always along the same short paths, then subscribing to revocation notifications would be an efficient solution. Since in this case only a limited amount of routers would need to be notified of the revocation.

However, if the number of certificates is very large, or data is delivered to various parts of the network, then using short term certificates would be the best alternative, since it removes the need for routers to subscribe for revocation notifications.

Since the Internet's traffic falls mostly to the latter case, and since it is more logical for the publisher and scope to be responsible for their traffic, using short-term certificates would be a best solution in most cases. Subscribing to revocation notifications could be an optional feature, which would be used, e.g., if the router wants to maximize the security.

3.3.3.3 Computational requirements and bandwidth overhead

The most time consuming operation for ECC signature verification is the elliptic curve point multiplication. By default, PLA uses implicit certificates where both the packet's signature and the TTP certificate is verified using three point multiplications. Traditional certificate mechanisms outlined here use two point multiplications per signature. Therefore verifying both the packet's signature and scope certificate requires four ECC point multiplications.

Efficiency can be increased by caching scope certificates in routers, in which case only the packet's signature would need to be verified. However, colluding scope owners and publishers can pollute such caches by sending a large amount of packets using different scope certificates and filling caches of routers. Authorizing RIds instead of publisher public keys in scope certificates would have a similar risk, since the amount of used certificates would be very high.

Additional certificates, e.g., a certificate from the operator or a certificate for delegating rights to utilize RId, would each require two point multiplications for verification and roughly 600 bits space in the packet header.

Overall, the bandwidth and computational overhead is not a significant issue, assuming that a dedicated hardware is used for signature verifications and long certificate chains are not used frequently.

3.4 Node-internal Architecture

This section contains the threat analysis for the node-internal architecture based on description of the blackboard architecture in D2.3.

3.4.1 OS security

The current architecture relies on the processor's memory management unit together with the operating system to provide security at the lowest level. Applications running on a modern operating system do not have a complete view of computer's memory. In fact, techniques like page table and virtual memory make it possible for operating systems to restrict application's capability to address non-authorized memory areas. As a result, an application cannot write into another application's memory, unless explicitly shared by the application "owning" the memory. In the prototype, all publications utilize the shared memory concept. In addition, the memory is marked as copy-on-write. Therefore, every write operation to the memory results in the operating system creating a copy of the original memory for the changes.

From a security perspective, this means that other applications cannot modify the existing publications in the blackboard. However, it is vital to remember that an administrator has access to all physical memory, swapped memory, and operating system internals. Therefore, a compromised system is able to access all data regardless of whether or not encryption and/or other information hiding techniques are used. This is a problem in current operating systems, and is not related to the blackboard architecture as such. We note that the blackboard architecture does not worsen (nor improve) the situation.

3.4.2 Scope access rights

Scopes provide limited access control. Information in scopes should be subjected to policy rules. Potential enforcement operations are restricting:

- who is allowed to publish in the scope
- who is allowed to subscribe to the publications in the scope
- temporal aspects: persistent subscription request vs. immediate return with or without data
- who is allowed to examine the content of the scope (i.e. list all publications)

The scopes must not leak information. The objective is that an attacker cannot gather information about scopes and what has been published in them. Therefore, a publication should only be accessible from the scope in which it was published. However, a publication may be published in many scopes, which could even have conflicting security policies.

Publications have metadata, including the list of scopes within which the publication exist. In case of such multiple scopes however, this metadata must not be leaked, unless the requester has the appropriate privileges to access that information. The blackboard architecture should enforce this on the operating system level.

3.4.3 Authorization for versioned publications

It is possible for an application to update the publication it has previously placed on the blackboard. To help link the two versions together (on a mental level), the blackboard architecture allows both versions to share the same high-level identifier. The problem that arises from this feature is that an attacker may republish the original publication with his own changes as an updated revision of the original publication, without the original author ever noticing or consenting. A solution is to require a proof of ownership when trying to publish a

revision of an existing publication. However, since it is conceivable for publications to be done in cooperation, the other authors should have a way of revising the publication. Access control mechanisms may have to be applied to versioned publications as well. How exactly this is done is left for further study.

3.4.4 Internal and external scopes

From an architectural point of view, scopes can be internal or external. An internal scope is one that exists only in the local host. The external scope exists in the networked hosts (in many cases including the local host). Both have different security requirements, and both provide different assurances of security. Clearly, if a scope is used to store company secrets, information about the scope and its contents cannot be allowed to leak to unrelated hosts, in case those hosts are compromised at some point in future. Therefore, some external scopes cannot be present in all hosts. Vice versa, host local information published in an internal scope, must not leave the host. From a security point of view, mapping internal scopes to external ones and vice versa remains a future work item.

3.4.5 Parallel usage of helpers

CPUs today have an ever increasing number of processing cores. The industry trend seems to be in increasing processing concurrency rather than pure speed.

Helper applications are applications that interface with the blackboard and act once suitable publications or subscriptions appear. Ideally, all these helpers would be able to perform parallel data processing. What this means, for example, is that transport processing would be done at the same time as the integrity of the packet is being verified. It is quite obvious that the biggest drawback is the case where the integrity check actually fails. Since the transport processing has already processed the packet, it would have to rollback the changes. However, even with these problems, the parallel handling of networked data appears as an interesting prospect and an avenue for innovation.

An alternative is to use a chain of helpers, much like what the current operating systems do. Each step is processed separately in sequence, passing information from layer to layer. This would partially void the benefits of the blackboard architecture in network operations.

3.4.6 Resource usage

At the application level, the operating system needs to keep track of the publications that the application has published and subscribed to. Some sort of accounting for usage and resource sharing or limiting mechanisms is required to prevent misbehaving applications from stealing all the system resources. Today, operating systems impose various limits on user processes, e.g., how many files an application may hold open, mass storage space, amount of memory, etc. What should be limited, and how much?

3.4.7 Application level security

Applications and higher-layer transport and content management services can offer their own security features such as TLS. How much should the applications trust the publish-subscribe architecture, especially compared to the current Internet? What kind of "Security-of-Service" can we offer to applications?

3.5 Helper functions

Helper functions in D2.3 were divided into three main categories, network management, remote service and host service functions. These categories are evaluated below.

3.5.1 Network Management Functions

These functions are used to gather information from the network, adjust the network's parameters and for other management tasks. The most commonly used management protocol on the current Internet is the Simple Network Management Protocol (SNMP).

Having a good security is essential for network management functions, since incorrect configuration can bring down parts or even the entire network or at minimum reduce its capacity. Manipulation of the network can also enable attacks against other users of the network (such as diversion of traffic). The most important requirement is that all network management control messages, which include information gathering and configuration messages, must be authentic and sent by trusted nodes. Otherwise, the attacker may feed incorrect information to the network management server or routers. In some cases confidentiality may be necessary to prevent outsiders from gathering information about the state of the network.

PLA would be suitable solution for providing data origin authentication and integrity protection. It would operate as follows: the network has an authority which all nodes trust – the trusted third party. Such an authority will issue two kinds of certificates, one set for sending configuration messages and another for providing network information. Routers would send information gathering messages to the management server using the latter certificate, while the network management server would use the former certificate to configure network parameters. All parties would be configured to drop control messages unless they have a correct certificate attached and the message's PLA signature is valid. Scalability would not be a significant issue since it can be assumed that nodes participating in the network management within the same network will have trust relationships with the network's authority. Therefore, key distribution or certificate management will not produce a significant overhead.

Alternatively, other solutions based on shared secrets or other approaches may also be used to provide integrity protection and authentication. Higher level security solutions can be used for confidentiality protection, when needed.

3.5.2 Remote Service Functions

Remote service functions include among others MTU (maximum transmission unit) discovery, error correction, content re-encoding, and application caches.

Security, especially the integrity protection and authentication, is essential for such functions. Unlike network management functions, these functions span across several networks making the necessary security mechanisms more complex.

The aim of the MTU discovery is to determine the minimum MTU size on the forwarding path in order to divide the publication into smaller pieces in an efficient way. Nodes participating in the MTU discovery must somehow prove that they belong to the specific forwarding tree, to prevent forgery of the MTU information by an external attacker.

For application caches there are two important issues. First, the cache must verify the authenticity of data in order to prevent attackers from polluting caches with incorrect data. Additionally, the receiver must be able to guarantee the authenticity of data received from the cache. Therefore, the cached data should contain, e.g., the signature of the original publisher.

Content re-encoding is the most difficult function from a security point of view, since the actual content is modified by third parties. The subscriber must be able to determine that nodes performing the re-encoding are adhering to the publisher's and subscriber's wishes and are not changing the actual content (semantics) in any way. It is very difficult for the subscriber to determine authenticity of re-encoded data in a scalable way since it is assumed that the subscriber does not have a full access to the original data. Instead the subscriber may establish a trust relationship with the encoding service and check the authentication of any received data. Policers may be used to check that encoding services are performing correctly by injecting known content into the re-encoder.

Certificate mechanisms, where the publisher authorizes a few nodes to perform re-encoding on its behalf, may be utilized here. However, it is difficult to achieve good security and scalability at the same time with such approach. If the amount of re-encoding nodes trusted by the publisher in the network is low, then only parts of the network can utilize them. As the amount of re-encoding nodes grows, the security may weaken since the publisher will not have a direct control or trust relationship with all nodes performing the re-encoding operation.

3.5.3 Host Service Functions

Host service functions are network management related functions that are used mostly by end hosts. Current examples of such functions are ping, traceroute, tcpdump, and others.

While security is important for host service functions in some cases, it is not as critical as for network management or remote service functions. Host service functions are located above the network layer, therefore they may utilize own higher level solutions to provide additional security when necessary.

3.6 Rendezvous Architecture Security Evaluation

The inter-domain rendezvous system is explained and evaluated in Section 5. In this section, we focus on the security evaluation of the rendezvous function. The security design work for the rendezvous system is still ongoing, and because of that, we concentrate here on analyzing the possible threats to the system and stating the goals of our security design against which we will eventually evaluate the solution. We are mostly interested in the security of the architecture and not vulnerabilities particular to specific implementations.

The rendezvous system is aimed at a global, inter-domain environment with a multitude of potentially malicious agents and it is a critical component of a PSIRP network. Therefore, the architecture should assume the minimum amount of trust for other stakeholders of the system.

Our methodology for rendezvous security evaluation is going to include two complementary approaches: firstly, we gather here attacking methods against all components of the implementation and later can analyze them by using attack trees starting from them. Secondly, we can try to formally prove that the architecture has some desirable properties related to our security goals. The first approach should give us realistic scenarios with few assumptions but the problem with the enumeration of attack vectors is that we can never be able to completely exhaust all possible methods of attack. Formal proofs do not leave security holes, but they may require unrealistically strong assumptions to go through which does not accurately reflect the real world.

3.6.1 Evaluation Criteria

On a very general level, the security related goals of our architecture can be categorized into the following requirements:

- Confidentiality of
  - publications and their content
  - network topology and rendezvous network information
  - subscriptions
- Anonymity of users
- Integrity of publication and scope information
- Authenticity between the digital information and external world
- Fair resource allocation and compensation
- Inter-domain policy compliance

- Admissibility
- Availability
- Utility
- Accountability of
    - users
    - rendezvous networks
- Recoverability
- Robustness

The rendezvous architecture should also provide rudimentary support for implementing an access control and authorization mechanism. Basically, it is impossible to prevent information from spreading after it has been leaked and application level solutions like data encryption and encapsulation of information inside trusted domains are probably needed to achieve high level of confidentiality. However, some use cases also benefit from the separation of data rather than relying on encryption for confidentiality. This may form a lightweight alternative (without encryption), or a second layer of defence for applications that are concerned that encryption may be broken (potentially years later). With policy compliant routing, the information can be routed via trusted domains while being unencrypted in some cases.

It should be noted that some high-level goals can be partially in contradiction with each other, and it is unclear whether an implementation exists that has all the desirable properties at the same time. For example, confidentiality is often at odds with efficiency and scalability: if information is stored inside a capsule like a traditional web server, each operation can be preceded by an access control operation on a trusted platform on the boundary of the encapsulation. On the other hand, if we want to disseminate the information to a large number of subscribers, the reliance on such protection boundaries and trusted platforms can become a bottleneck. A similar potential mismatch exists between anonymity and accountability. In a global system, guaranteeing the locality of failures can go a long way towards robustness and fault tolerance of the whole system but typically this makes the system harder to maintain. Generally, the more powerful operations the system makes available, the more destructive attacks may be enabled by the same operations.

3.6.1.1 Challenging Use Cases

Some of the most challenging individual use cases for a data-oriented system seem to be related to access control:

- Popular streaming of paid content, where the system should support basic access control and still be scalable to millions of subscribers constantly joining and leaving,
- extremely valuable content like a new movie release,
- private scopes that are globally accessible, and
- user accessing her private data from an untrusted computer.

Private scopes are stored at a user chosen rendezvous network that the user trusts, for example, her own computer. If global access to such scope is required, an indirection pointer to the scope is stored in the rendezvous network interconnection architecture that makes it possible to locate the scope remotely. The scopes may have some external meaning attached to them. For example, "company confidential scope" could represent a distribution strategy for information that limits the set of allowed subscribers to the employees of the company. With private scopes we mean cases where even the very existence of such scope should be confidential.

3.6.1.2 Scopes and Resources

In a PSIRP network, scopes represent distribution strategies for data that is published in the network. Each publication must be published inside at least one scope but a single publication can belong to multiple scopes simultaneously and can be moved from one scope to another. The scope determines among others the QoS semantics, policies, access control, persistence, reachability, and location of the published data. Scopes can also be thought of as the means to map the publications to the real-world resources since the scope controls the access to the publication (for example the ability to subscribe).

3.6.2 Threat Model

In this subsection we try to list categories of potential attack vectors against the rendezvous system by considering where the attack is launched and what methods and targets might be chosen.

- Attack from inside by
  - a malicious operator building part of the network that
    - could try to change the topology or affect topology 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 communication between nodes by adding, removing, changing or eavesdropping messages.
- Attack from outside by
  - end-user computer that
    - might attack the system by using it in a legitimate way through its API/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 by
  - producing Byzantine failure modes or
  - information warfare
    - where "government hidden functionality" in router processors designed or manufactured in other countries may be activated by secret command strings.

3.6.2.1 Attack Objectives

Next we try to consider what objectives the attackers might have related to the rendezvous functionality. Possible aims include:

- unauthorized deletion of a publication or a scope,
- unauthorized modification of publication contents or metadata,
- unauthorized publish or subscribe (possibly with a specific RId or SId),
- gaining knowledge of the existence of a certain publications or a subscriptions,
- unauthorized advertisement of an existing scope,
- determining the identity of a publisher or a subscriber,
- affecting the forwarding route formed for a specific publication so that the new route is not policy compliant,

- gaming the route selection process to get personal advantage,
- gathering an unfair amount of resources/control of/to the rendezvous system (for example, Sybil attack), and
- preventing the normal operation of the rendezvous system (for certain publishers, subscribers, domains, or rendezvous networks/scopes).

3.6.3 Initial Solutions

We try to follow known good principles of design for security in our architecture. For example, it is important to avoid single points of failure in a distributed system. Another principle is that there should be multiple methods of security at each layer of the system so that breaking a single security mechanism does not render the entire system unusable.

3.6.3.1 Access Control

The PSIRP rendezvous architecture provides a basic access control mechanism by implementing scopes inside rendezvous networks (a single machine can also implement its own local rendezvous "network") that are trusted by the owners of the scopes. If a publisher now trusts the scope and publishes a publication inside the scope, the rendezvous system guarantees that every rendezvous operation by a new subscriber involves a node from the rendezvous network that implements the scope. This node can then be used to control access to the publication since there exists a trust chain from the publisher to the node. The rendezvous point can, for example, send a signed certificate containing a shared secret used as a capability addressed to the new subscriber that enables her to join the forwarding tree. This capability is then checked at some router currently on the forwarding tree against the public key of the publisher to determine whether the new subscriber can join the tree.

For popular content, such a control point in every rendezvous operation may become a bottleneck, but fortunately this is only a problem for confidential publications. For public publications, it is possible to cache the rendezvous information completely using the scalable, untrusted rendezvous network interconnection mechanism (Canon hierarchical DHT in our current design). We believe that public publications should be the norm and if high-level of confidentiality is needed, then the reachability of the publication should be limited to trusted domains by using publication-specific routing policies. Alternatively, publishers can split the publication into multiple parts that are distributed via different routes which makes it more unlikely that malicious domains on the forwarding route can get access to the data published. The actual mechanism of performing the access control check can vary between different rendezvous networks.

We do not specify the format for the information that the subscribers use in their subscription messages to provide their credentials to the rendezvous network responsible for the scope requested. The information should, however, be encrypted using scope's public key to prevent anybody else from eavesdropping the message and using the credentials. Different rendezvous networks may use different mechanisms and subscriber credentials need to be compatible with them. For example, both role-based access control to scopes based on identification of the subscriber and capability-based techniques are possible.

Similarly, rendezvous networks may require some credentials from publishers who wish to use the resources of the rendezvous network to store publications. In typical cases, we would assume the publisher either to control the rendezvous network or to compensate for the use of its resources by some out-of-band means.

The data can also be encrypted on the application level to provide confidentiality, but this leaves the problem of key dissemination to the applications.

3.6.3.2 Integrity

The integrity of the data is guaranteed by signing the data with the public key associated with the RId of the publication (or a chain of certificates starting from that public key). Secure association with a specific scope can be done by signing the RId with the public key of the scope. See section on identifiers for a more detailed description about how to use cryptographic signatures and certificates to achieve this.

3.6.3.3 Availability and Utility

The current rendezvous solution uses a Canon hierarchical DHT [Gan2004] as the rendezvous network interconnection mechanism that allows publishers and subscribers to find the rendezvous network containing a given scope (and therefore its rendezvous point). This is achieved by storing scope pointers in the Canon's Chord ring address space and guaranteeing the integrity of the pointers by signing them with the public key of the scope that is pointed to. Because the SIds are distributed on the Chord ring using one-way hash function, it is assumed that they are evenly distributed among all nodes of the DHT (at each level of the Canon hierarchy) and that the load is evenly balanced between the nodes. However, as the number of nodes in the interconnect is several orders of magnitude lower than the size of the address space, this still leaves the possibility of generating enough random SIds so that a number of SIds are hashed to be stored by the same node. With a naive implementation, this leaves the possibility to attack a particular node in an otherwise scalable system by depleting its resources and this way target an attack against particular existing scopes. This type of attack can be prevented by periodically changing the hashing function used (in the whole Canon hierarchy, to keep the invariants of the system intact).

As the inter-domain rendezvous network interconnection is built by multiple parties providing their own resources to the system, there needs to be redundancy on both nodes and their connections. Also, in the current Internet, it has been shown that non-transitivity of reachability between the nodes of a DHT is a real problem [Fre2005]. The redundancy provides fault-tolerance against node and link failures and at the same time protects the system against DDoS attacks targeted to specific nodes. The hierarchical structure of Canon should also provide the system with good failure locality. If the nodes of the interconnection Chord structure can only be reached via the Chord protocol following the Canon topology, then it also can be used to limit the effects of a DDoS attack mostly to those parts of the network that contain the attacking nodes. This can be achieved by load balancing the requests originating from different child hierarchies in the Canon structure at each level.

3.6.4 Analysis

A DDoS attack (see Section 3.9) against a particular scope/rendezvous network using a large botnet and trying to render the scope unavailable is probably the single most difficult attack to prevent as it is difficult to separate legitimate users from compromised ones and, for example, give their packets a higher priority. However, this problem applies only to subscriptions to an access controlled scopes and remote publication operations that requires every rendezvous operation to reach the target rendezvous network. With public scopes, it is possible to cache the results near the clients and merge several subscription operations into a single subscription upstream in the Canon hierarchy. Also, popular, access controlled scopes require large rendezvous networks to handle the number of subscribers joining and leaving. The rendezvous function is only a control plane operation and therefore attacks against it do not affect already formed forwarding trees. Trying to fill the RId/SId address space is not a feasible attack as identifiers are 256 bits long.

Our current design does not yet have a solution for how to share the resources of the rendezvous network interconnection fairly between the participating rendezvous networks. We have assumed that each rendezvous network provides a set of nodes to the Canon hierarchy for storing scope pointers and routing subscription and publication operations towards their rendezvous points. Each node uses storage space for scope pointers of other rendezvous

networks and routes subscriptions/publications originating from other domains. Basically, the interconnection should statistically provide storage and routing resources to each rendezvous network and their customers proportional to their investment in the Chord structure. This problem mainly stems from the fact that we want the interconnection architecture to be completely distributed, dynamic, and scalable. The number of scope pointers stored in the system is likely to be much larger than the number of resource records in the Domain Name System (DNS) of the current Internet.

3.6.5 Conclusion

Generally, it appears that data-orientation is a more natural fit for public information where the efficient dissemination is the most important requirement and the same data can be multicasted to a large number of subscribers. If confidentiality of information is the most important concern, then the more traditional message-oriented remote object/server or actor-based communication model with possible end-to-end encryption for each receiver individually can be secured in a more straightforward manner.

We will continue to work on the security architecture of the inter-domain rendezvous function. Especially, we need to solve the fair resource usage issue of the rendezvous interconnection and try to find new ways to mitigate DDoS attacks against rendezvous networks.

3.7 Topology Management and Formation

The topology management and formation functions are not fully specified in the current architecture [Psi2009], so assumptions have to be made concerning the information each node has about the overall topology and the topology management messages exchanged. Below we outline some potential attacks that should be considered in the final design of the topology management functions, including the specification of virtual links.

Path failure attack. To recover from a path failure, every node along the path is allowed to modify zFilters. However, this provides the potential to an attacker to modify a zFilter in such a way that packets will follow additional routes (sinkhole). A countermeasure can be either to not allow nodes along the path to modify zFilters, or to somehow inform the source that a specific node modified the zFilter. This could potentially be achieved by signing the zFilter and requiring that the first node to finds an invalid signature to report this to the sender for further investigation.

Loop control attack. In order to avoid loops each node must know the neighbouring nodes' outgoing Link ID and LITs towards the node itself – called incoming Link ID and LITs. For each incoming packet, the node checks the incoming LITs of its interfaces, except the one from where the packet arrives, and compares them to the zFilter. A match means that there is a possibility for a loop and the node caches the packet's zFilter and the incoming Link ID for a short period of time. In the case of a loop, the packet returns over a different link than the cached one. A node A can create zFilters containing node B's incoming LITs; this way it will force B to maintain state. Moreover, suppose node A wants to send a packet to C through B. A malicious node that is next to B may name its outgoing interface towards B in a way that it will match the zFilter used for the communication between A and C. Then it will send a packet to B that will have the same zFilter with the one used between A and C; this way when B receives the packets that originated in A it will think that a loop has occurred and it will drop the packet. If the malicious node is not attached to B it may still be possible to perform the attack by creating a virtual link with B. A solution for this can be to use a TTL field in each packet or to store a data hash as well to detect the loop.

Eavesdropping. Suppose a malicious node named B wants to eavesdrop on packets that A sends to C. It can create a virtual link between itself and any node along the path from A to C and assign an ID to this virtual link that will match the zFilter used between A and C.

3.8 Network Attachment

The architecture is clear that once a network attachment point is selected, authentication and authorisation can take place between the network attachment point and the host. In this section, we consider the security risks before the attachment takes place – that is during the network attachment process. These risks may compromise the confidentiality, integrity or availability of either the network attachment point or the user.

From the user perspective, they are accountable for traffic sent into the network through the network attachment point. We assume that the user is authenticated and the confidentiality and integrity of this traffic is maintained once network attachment is completed. It should not be possible for other users to inject traffic into the network without being accountable. Similarly, the user needs to authenticate that their traffic is being sent through the expected network attachment point, regardless of whether or not the payload of the traffic is encrypted. There seems to be little to prevent a rival attachment point from also subscribing to the user traffic, however, this should not prevent the traffic reaching the chosen attachment point.

Also of concern is the traffic that a user receives which does not originate from the chosen network attachment point. It should not be possible for other local users to pretend to originate traffic from the attachment point (for example using the same FIds). However, in the current attachment process the FIds will be freely available to anyone listening on the local network, and little prevents their use by other users or rogue attachment points. It is also possible that another user can participate as a man-in-the-middle by relaying traffic between users and attachment points (that are perhaps out of direct range of the user). Although such behaviour may be desired to construct multi-hop networks, the user must be aware that a man-in-the-middle may be altering the traffic that is sent or received. Although prior distribution of cryptographic keys used in packet-level authentication can provide a solution, it is not clear that a user can securely connect to networks for which they do not have a prior relationship.

The user must allow a certain amount of traffic to be received for the attachment (or re-attachment for mobility) process to take place. This traffic should be limited to legitimate attachment process traffic and not provide an opportunity for other parties to attack the user. It is possible to imagine that other local attackers may overwhelm their peers or perform other attacks through the network attachment process. Such an attack might, for example, result in the attacker being the sole user of a network attachment point with limited bandwidth, while denying service to other users. Alternatively a rival network may send spam on the FId used by another attachment point (although it is expected that such content will be dropped by the user's network attachment component).

Other concerns are privacy and business confidentiality and espionage. Although no permanent network identity is used (such as a MAC or IP address) the user traffic may be identified by a set of identities used during the attachment authorisation or packet authentication. Thus, mobile devices and users may be tracked by neighbouring attachment points or users. The architecture, and the related packet authentication technology, may wish to consider allowing one-off pseudonyms to identify the user. In the case of espionage, neighbouring rival attachment points may analyse the identities and usage patterns of customers of other networks.

Hereafter we present in brief some potential attack scenarios for the Network Attachment procedure.

Attach-based flooding attack. If any node can be attached to any NAP without any kind of control and limitation, then an arbitrary number of adversaries can be attached to a NAP. This might cause overload on some NAPs, depending on the state that each NAP maintains per attached node. Adversaries might target NAPs (using eavesdropping techniques that can be applied to zFilters) that either are already overloaded, or control central attachment facilities on subnetworks, such as those that serve rendezvous points. These scenarios might force DoS on any networking function that relies on network attachment.

Attachment toggling attacks. Along with exhausting the attachment state capability of the NAP, an attacker may also perform toggling attacks by attaching to and then leaving the NAP, with an intent to exhaust the computation power of the NAP or create onward signalling traffic. The NAP may consider removing attachment state at its own timescales and batching de-attachment requests, similar to leaving an IGMP multicast session.

Handshaking-based flooding attack. The attachment procedure includes a handshaking phase between two NA points. Similar to a SYN Flooding attack, it is possible to have some adversaries that locate a particular NAP and produce a sequence of messages for handshaking, but never close or terminate this handshaking (half-open attachments). Pending requests should not exhaust resources or maintain states that are unmanageable.

Spamming. The well known IDs for the network attachment scope (SNA) might open spamming opportunities to the clients. Clients might receive advertisements and solicitations from many NAP (or prentend NAP) nodes, before deciding to attach on a particular one. Such solicitations might contain legitimate information about attachment points and networks, or spam content.

3.9 DDoS resilience

This section considers the evaluation criteria for DDoS defence according to which the architectural design after D2.3 will be evaluated.

Distributed denial of service (DDoS) attacks consist of large number of hosts deliberately trying to overwhelm the targets storage, processing, or communications capabilities (e.g. bandwidth between it and the rest of the internetwork). The basic protection mechanisms utilized in the existing literature are traffic classification (e.g. filtering [Ion2002, Arg2005, Hui2007, Liu2008] and proof of right to send (i.e., capability [And2004, Wen2006, Yaa2004, Yan2005] mechanisms), replication and diffusion (e.g., overlay mechanisms), and hiding (e.g., overlay mechanisms such as secure-i3). Liu et al. [Liu2008] show that neither reactive filtering, nor proof of right to send mechanisms have optimal resiliency characteristics in IP-based networks. Reactive methods tend to be slow, it may take hundreds of seconds for the filters to propagate, whereas capability based systems are vulnerable to denial of capability attacks [Arg2005, Liu2008]

Massive replication works by amassing resources greater than the attacker. However, it is so costly to implement that only few actors can afford it. It may still be useful, or even necessary, in case of services that can serve many or all participants in the network. As none of the solutions are perfect, the PSIRP architecture should be designed to employ multiple techniques to prevent and mitigate effects of DDoS attacks, without sacrificing performance or increasing costs too much. Firstly, the system should be built around a 'default off' assumption. Both the publisher and the subscriber need to indicate their wish to communicate to the rendezvous system before the publisher is given forwarding identifiers. These FIds specify a source route and should contain a mechanism for invalidating them, e.g. either by changing them periodically or requiring a cryptographic capability (e.g. based on PLA) ensuring that the permission to send is valid.

Locally, it should also be simple to apply reactive filtering. If an attacker manages to circumvent the forwarding security mechanism and send data to the target with a false FId, the target with the help of the network should be able to stop the traffic for the FId in question. This can be done, e.g., with reactive filtering by propagating the FId upstream to those routers that would otherwise forward a packet based on that FId towards the target. However, care must be taken in order to not allow such filtering requests to be used to mount a denial-of-service attack on someone else.

Finally, the rendezvous service is a natural target of an attacker due to the central service it provides to all nodes in the network. It needs to be protected. Replication, hiding, and resource control economics are potential avenues for DDoS resistance. If over-provisioning is

the main solution, then special care should be taken in analyzing business incentives to invest in the service.

The system both as a whole and as individual components needs to be analyzed against DDoS attacks. Attackers can utilize various techniques to achieve their target and care should be taken in trying to enumerate the potential vulnerabilities and dealing with them.

4 Performance Evaluation of Architectural Solutions

In this section we provide results from the quantitative evaluation activities carried out in the project up to this point. We first provide some results from a performance evaluation of the prototype implementation which was jointly performed with WP3 in Section 4.1. The evaluation of the zFilter solution for intra-domain forwarding is given in Section 4.2, followed by a discussion on the performance of network coding solutions in Section 4.3. A performance evaluation of the overlay PSIRP variant discussed in D2.3 is provided in Section 4.4. In Section 4.5 our early work towards the integration of network testbeds with packet level simulations is described, before outlining the currently open issues and drawing some conclusions on the evaluation work up to this point in Section 4.6.

4.1 Evaluation of the prototype implementation

We have set up two different environments for evaluating the performance of the current PSIRP prototype. The hardware used in the evaluation consists of laptops and desktops fitted with NetFPGA cards to test the forwarding node implementation. The laptops have 4 GB of DDR2 RAM, an Intel Core 2 Duo T9600 processor running at 2.8 GHz and a gigabit Ethernet interface. The operating system is fixed to FreeBSD 7.1- BETA2 with GENERIC kernels. Two of the NetFPGA desktops have 8 GB of DDR2 RAM, an Intel Core 2 Quad processor running at 2.5 GHz with a gigabit Ethernet interface. One of the NetFPGA desktops has 2 GB of DDR2 RAM, an Intel Core 2 Duo E6320 and two gigabit Ethernet interfaces on motherboard. The NetFPGA source used is version 2.1.

4.1.1 End Node Efficiency

We start evaluating our prototype in terms of its internal efficiency. Once we establish a baseline for efficiency, we investigate the interactions from the kernel point of view via page fault tests. We expect that a shared memory strategy in a pub/sub architecture would heavily depend on demand paging. Finally, we conclude by pushing the prototype at its limits and trying to clock the maximum performance in a common file transfer test. While the inherent speed will not be achieved at this rather early stage of the prototype, we can still establish rough bounds of the overall performance capabilities.

Measurement I: Local Stack Performance

Our first measurement investigates the local stack (end node) performance. The FreeBSD implementation consists of three components: a loadable kernel module, a shared library, and a user space daemon. These are used for basic publish/subscribe operations (both node-locally and between nodes). To show the performance of this implementation, we have measured the time used for different operations (publish and subscribe).

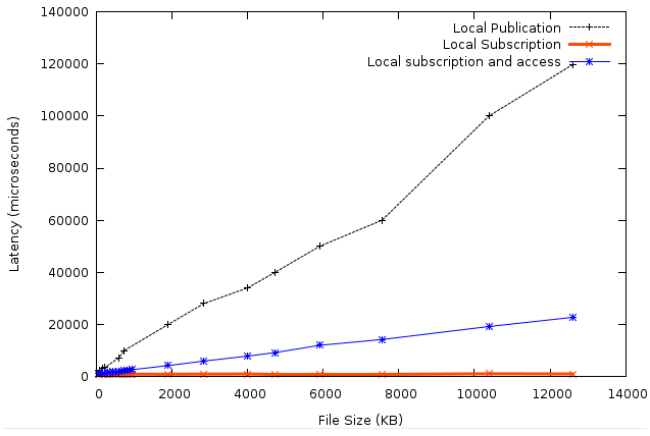

Figure 13: Reply latency of local operations.

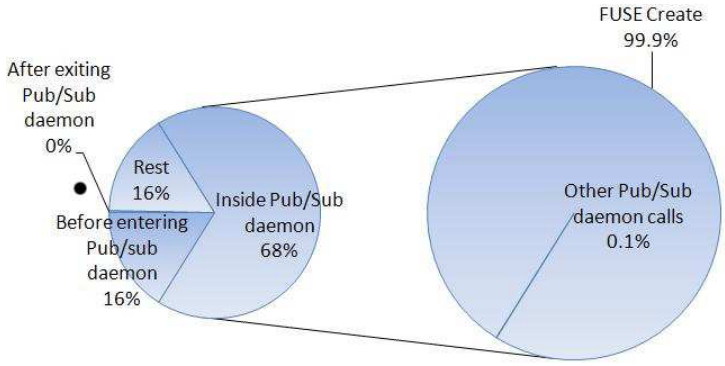

Figure 14: Time usage analysis for the different code components during publishing.

The average latency of replying to a local publish/ subscribe request is shown in Figure 13. The figure shows a constant time for subscription of different locally cached publications (in this case, files with various sizes). A subscribe operation just returns the metadata of the publication to the subscriber. For local publications this metadata is available locally, so the lookup operation does not take much time. Subscribing to such a publication causes a simple memory mapping operation and is thus independent of the data size. The initial subscription is equivalent to a simple availability check. Therefore, even if the publication metadata is not available locally, there is still no difference from the application point of view. In that case, the publications availability would be checked through the network, and the result would be returned to the application before a timeout. Most of the time during a subscribe operation is spent in the Pub/Sub daemon from which different *open* and *read* calls are requested. When the application wants to access different parts of the publication, the situation changes slightly. In this case, when the publication is completely available locally, the time required to read the memory area is linear in the size of the data. Each attempt to access different segments of the publication causes a local access to the memory pages of the publication. Again, if some pages of the publication are not available locally, the system will try to reach the through the network with the help of previously received metadata. There is again no difference in the application API when the publication is available locally or somewhere in network.

It can be seen that the most time consuming local operation is publish, with a linear increase by data size. For large files, the time consumed inside the Pub/Sub daemon is much higher than any other operation. Figure 14 gives a detailed breakdown of the time consumed in various phases. As can be seen, most of the time used in the Pub/Sub daemon is for creating the publication, which requires creating and storing metadata for the publications. Currently, the time consuming metadata operations are the creation of the RId, which is a hash of the data, and the creation of a bitmap which keeps track of locally cached parts of the data. For example, the delay to publish a 12 MB file is around 90 ms, which is acceptable, because this delay is caused by hashing of the content and is done only once during the lifetime of the publication. Figure 14 also shows the time spent in the syscalls handling between the I/O module and libpubsub and the delays between different pub/sub daemon event handling calls which also affect the latency. Outside the pub/sub daemon, buffering the file from the disk can cause a major effect on latency. The rest of the time spent here is caused by interactions between the kernel module and the Pub/Sub daemon. Inside the stack, the time spent between different pub/sub daemon event handling calls is used to check the output queue for possible packets waiting to be handled.

Measurement II: Page Fault

The second measurement in the efficiency category is to test the time to serve a page fault. In this test, we measure the time it takes to handle a page fault utilizing our prototype as the transporter of the data. For this test, we instrumented the segmentation fault trap handler.

More specifically, we timed the execution of the vm_fault() function call. For acquiring timestamps we used the nanouptime() function.

The test file we used to evaluate publish/subscribe and mmap() performances was a 2.1 MB file which required 535memory pages. We caused page faults on pages 1, 50 and 400, to simulate an application accessing publication data.

Table 1: Summary of page fault measurements.

Test	Latency (in μs)
First run	
Pub/Sub: 1st page access	103
Pub/Sub: 2nd page access	88
Pub/Sub: 3rd page access	74
mmap: 1st page access	6.0
mmap: 2nd page access	5.9
mmap: 3rd page access	5.8
Subsequent run	
Pub/Sub: 1st page access	83
Pub/Sub: 2nd page access	64
Pub/Sub: 3rd page access	57
Remote	
Pub/Sub:	
- Total	500 ($\pm$110)
- Request-to-Receive	$\approx$300
mmap + NFS:	
- Total	300 ($\pm$5)
- Request-to-Receive	$\approx$280

Our initial approach was to get one figure that best describes the speed of our prototype, but we ran into problems due to the multiple levels of caching. In FreeBSD, caching is spread out to numerous places. There are caches ranging from disk buffers to on-die caches in CPUs. Therefore, we measured the performance when data was first accessed and then when data was later accessed again from another process.

Dissecting the results presented in Table 1, we can clearly see that we are quite far from the performance of mmap(). Most of the performance deficit was already analyzed in previous measurements. The average time spent in user space handling the page fault was measured to be around 50 µseconds. We experienced an increase of speed of 10 to 20 percent if the page was already accessed before. The improved speed is due to kernel caching. However, the fact that mmap manages to perform the subsequent runs without page faults, indicates that our kernel module is not yet optimal.

As for the network results, we have started and analysis for the mmap + NFS case. By default, NFS would have fetched two pages, and our prototype up to 16 pages. To have comparable results we instructed the test programs to fetch exactly one page during a page fault. From the results we can see that we are almost on a par with NFS for network handling. Our weakest link is currently the processing of the data.

Measurement III: Throughput

Finally, we have measured the throughput achievable by the present prototype implementation. In order to establish the baseline network performance of our prototype, we measured the maximum transfer speed. As our prototype operates by demand paging, the file transfer is realized by causing page faults on all pages. Therefore our test application subscribes to a remote file, and causes page faults on all pages. Each page fault is caused by a single memory read operation. We test the throughput with different effective payload sizes. We were hoping that the results would reveal saturation points, in which the processing of data consumes more time than the latency caused by the network data transfer.

We ran a file transfer test with different payload sizes ranging from 512 bytes to 4096 bytes. Since we are using directly connected PCs, we can use Ethernet jumbo frames, and do not have to resort to fragmentation. The transferred file used in these measurements was sized at 6.1 MB. The file was cached at the sender.

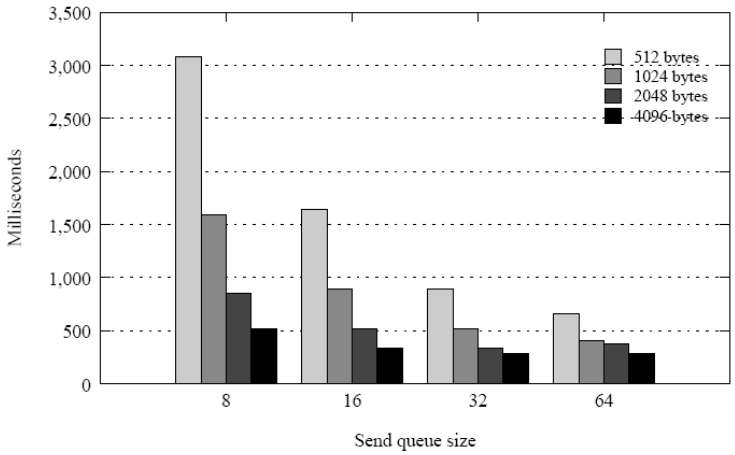

Figure 15: Data transfer latencies with various payload sizes.

Results, as depicted on Figure 15, suggest that the maximum payload throughput with our setup is of the order of 21 MB/s. The effect of send queue size can be easily observed. Sending out packets with send queue set to one, our prototype manages to emit two consecutive packets in 1.7ms. When the send queue size is increased, two packets can be sent within 0.1ms.

4.1.2 Forwarding Node Efficiency

The evaluation of the forwarding node implementation is mainly based on various latency measurements of single packets. This is because the main improvements from our implementation compared to standard IP router stem from the fast switching that should reduce latency significantly. The performance of the forwarding node was evaluated by software that runs in the FreeBSD kernel to have more accurate timestamps for measurements. We also set the receiving interface to polling mode so as to get more accurate results [Wan2005].

Table 2: Results of latency measurements of forwarding node implementation in microseconds

Path	Average latency	Standard Deviation	Latency/ NetFPGA
Wire	16	1	x
1 NetFPGA	19	2	3
2 NetFPGA	21	2	3
3 NetFPGA	24	2	3

Netgraph [Cob2000] was used to communicate with the network interface (NIC). In addition to the measurement machine, three NetFPGA machines were used in the measurements setup. Additional machines were added later on to introduce background traffic. The topology is shown in Figure 16.

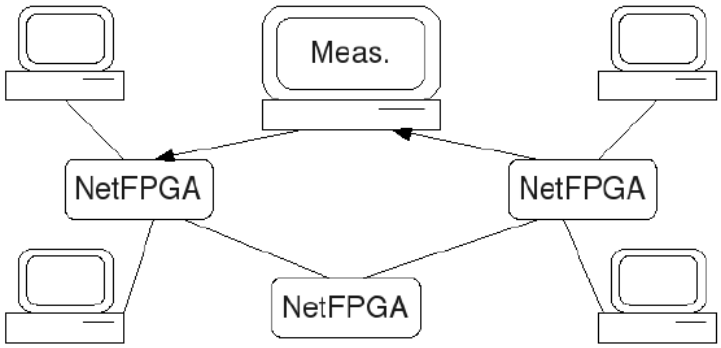

Figure 16: Topology of measurement setup.

This setup was used to measure the latency between two NICs in various conditions. The first set of measurements (shown in Table 2), were on the latency of the forwarding code with very low load. In each case, the latency of 10000 packets was measured. Each packet contained a MAC header, a Bloom filter, and a time stamp. Packets were sent at rate of 25 packets per second and were forwarded out from two ports on each node.

The usage of Bloom filters creates 0.056 μsec latency for each packet. That is, in fact, α quite insignificant amount of time compared to the measured 3 μsec that is used by the whole NetFPGA implementation. To verify if this is actually true, it was worthwhile to measure an implementation that does not use Bloom filters but just forwards packets on predefined ports and see if there is actually a visible difference. Also cases where traffic is forwarded to one port only and to all ports, except to the one that it came from, were measured. The results are shown in Table 3 and those were consistent with our expectations. For the implementation to be feasible, it needs to also perform well under high load, so the same measurements were performed with added background traffic. Background traffic is added on the first forwarding node. The background traffic consists of 1Kbit packets.

Table 3: Results of latency measurements of implementation with no Bloom filtering, forwarding to one port, forwarding to three ports in microseconds.

Path	Average latency	Standard Deviation	Latency/ NetFPGA
3 NetFPGAs fixed ports	25	4	3
3 NetFPGAs one port	25	4	3
3 NetFPGAs to 3 ports	26	4	4

Table 4: Latency through 3 NetFPGAs with background traffic in microseconds.

Background traffic	Average latency	Packet loss	Latency/ NetFPGA
0.93Gbit/s	30	0	5

As can be seen from the results shown in Table 4, additional traffic seems to increase latency noticeably. This can be partially explained by the fact that if a 1Kbit packet arrives just before the timestamp packet, it has to wait until the previous packet has gone through the input arbiter. This does not reduce throughput because the data path is faster than all inputs combined. If necessary this can be countered by adding parallel data paths.

One important point was to compare the implementation to IP based routing. This was measured by using 100000 ICMP echo request packets through the plain wire, a Bloom filter (BF) implementation and a reference IP router implementation. The IP router implementation had only five entries on its forwarding table and it could be expected that the latency would increase as entries increased. That does not happen with the BF implementation so the results compare almost the best case for the IP router to the average case for the BF. The average from a high number of samples was taken to compensate a quite high mean deviation. From the results shown in Table 5, it can be seen that the BF implementation adds less latency than the IP router.

Table 5: Ping through various implementations.

Path	Average latency (ms)	Mean Deviation
Plain wire	0.094	0.028
IP router	0.102	0.044
bloom filter	0.096	0.028

4.2 Analysis of the zFilter intra-domain forwarding mechanism

The main objective of this section is to explore the scalability limits of the zFilter forwarding approach in an intra-domain setting. First, we present the metrics that can help us evaluate the forwarding architecture. As Bloom filters form the basis of the forwarding method, we extensively study the false positive probability of the Bloom filters used for different design parameters and optimizations. We complement this analysis with packet-level ns-3 simulations providing false positive measurements for realistic intra-domain topologies so as to gain further insights on the forwarding efficiency of this mechanism. We then summarize the lessons learned and the economic impacts of the design parameters using a basic system dynamics model and, finally, present our conclusions.

For the sake of readability, we briefly summarize here the zFilter-based forwarding architecture. In the basic design, each directed link has a Link ID with a couple of bits set to one. If we want to create a path/tree, we simply take the bitwise OR of the corresponding Link IDs (i.e. insert them into a Bloom filter), which will result in a so-called *zFilter*. This zFilter will be included into the packet header, describing the path/tree the packet should traverse through. When making forwarding decisions, the routers check which of their Link IDs are present in the zFilter. This decision can be easily made via bitwise AND operations. An optimization is for all links to have d different Link Identity Tags (LITs) (and d different forwarding tables), thus allowing the topology manager to compute d different zFilter-candidates describing the same path/delivery tree so as to then select the best one based on some criteria. To make this technique work, the id of the forwarding table chosen for the zFilter should also be present in the packet header.

In the basic mechanism, the forwarding table of a node should contain one entry for each of its neighbours. In order to enable the creation of *virtual links/trees* that span multiple physical links, we introduce so-called *virtual states*,. In this case, the same Link IDs/LITs will be present in the forwarding tables of the corresponding nodes; if the topology manager decides to use that virtual link/tree, it may include only a single element into the zFilter instead of all the elements representing the physical links on the path/tree. For a detailed architectural

presentation of the zFilter-based forwarding architecture, we refer to our earlier deliverable D3.2 [Psi2009].

4.2.1 Performance indicators

False positive rate: A fundamental metric is the *false positive rate* of the in-packet Bloom filter (BF). Recall that due to the heavy compressing nature of Bloom filters, sometimes false positive answers are given to a membership query, i.e. an element was never added to the BF, but a positive answer is given when testing whether this element is a member of the BF. This metric shows how often we get false positive answers when querying an elements' membership in the BF. Assume that Link ID Tags (LITs) are already in the form of *m*-bit vectors, with *k* bits set to one, and further assume that *n* of them are added to a candidate Bloom filter. Assuming that the hash functions for generating the LITs are uniform, for a given set of system parameters *m*, *n*, *k*, it is easy to show that the a priori false positive estimate before adding the actual elements is the following [Hao2007, Lum2009]:

$$fpb = \left[1 - \left(1 - \frac{1}{m}\right)^{k \cdot n}\right]^k$$

Setting the partial derivative of *fpb* with respect to *k* to zero, we get the value for *k* that minimizes the false positive probability. This is attained when $k = \log2(m/n)$, and is rounded to an integer that determines the optimal number of ones in the LIT representation.

Note that the calculation of *fpb* does not require knowledge of the number of bits set to one in the Bloom filter *after* inserting the LITs. An accurate estimate of the basic false positive rate can be provided once the fill factor of the Bloom filter is known; the fill factor is the fraction of bits set to one after all the required LITs are inserted. The posterior false positive estimate *fpa* is the expected false positive estimate after BF construction:

$$fpa = \rho^k$$

In practice, the prior and posterior estimates are usually very close due to their concentration around the mean. In our small bit vector scenario, every bit counts and fewer false positives will be key to system performance. The *fpa* metric has a very important role at zFilter-creation time, as the topology manager may select the zFilter that has the minimal *fpa* among all candidates; i.e. it selects the candidate filter which is expected to perform best by considering the amount of bits set to one in the resulting zFilters and the amount of ones in the single LITs (*fpa optimized* selection).

Finally, the *observed false positive rate* is the actual *false positive rate* (*fpr*) when a set of membership queries are made on the Bloom filter:

$$fpr = \frac{\text{Observed false positives}}{\text{Tested elements}}$$

Note that the *fpr* is an experimental quantity and not a theoretical estimate. The minimum observed *fpr* of the candidate Bloom filters provides a reference lower bound for a specific Bloom filter design. Recall that an *fpr optimized* selection of the LIT-based zFilters can be made when the topology module bases the selection of the zFilter on the lowest observed *fpr* after checking for false positives against a set of LITs expected in the network path.

Forwarding efficiency: The false positive rate describes the overall network performance only indirectly as it gives a detailed picture about the Bloom filter as a data structure; it does not capture the topology-dependent characteristics of the zFilters. In practice, a false positive answer to a membership query results in sending the packet over a link was not originally added to the zFilter, so some network resources are wasted. Consequently, we need another metric to capture more accurately the actual bandwidth consumption caused by false

positives. Hence, we introduce *forwarding efficiency* as a metric to quantify the bandwidth overhead caused by sending packets through links according to a false decision:

$$fwe = \frac{\#\text{Links on shortest path tree}}{\#\text{Links during delivery}}$$

In other words, forwarding efficiency is 100% if the packets strictly follow the optimal tree for reaching the subscribers (defined in any appropriate metric), otherwise it shows how much of the total traffic is not requiredto reach all the interested subscribers. This metric is more informative than the false positive rate when larger subscriber sets are served by multiple delivery trees and, therefore, identical packets would traverse through some links more than once. Though the forwarding efficiency in each tree may be high, the overall forwarding efficiency for the publication can be low if the trees are not perfectly disjoint. Forwarding efficiency can help decide whether it is better to use one big tree to reach all subscribers or it is more beneficial to use multiple trees for publication delivery. Another example is when virtual links are used in the network, where false positives may be more costly, as they may easily result in deliveries over multiple hops (as a packet is matched at one node to a virtual link, it will unavoidably match again in the following nodes participating in the virtual link).

4.2.2 Analysis of Bloom filter performance

We carried out a set of simulations to get a practical understanding of the actual performance and the design space of the BF-based data structure for compact representation of Link IDs/Link Identity Tags. In order to have small overhead in the packet headers we set the size of the zFilter to 256 bits (248-bit BF + 8 additional bits, e.g., for the forwarding table selector), which compares fairly to a pair of source and destination IPv6 address fields (2 * 128 bits).

Our goal is to evaluate how many links can be inserted in the zFilter for a certain upper bound of false positive probability *Pmax* = 5%. Recall that false positives lead to data packets delivered over neighbouring link IDs that were not included in the zFilter. However, the probability that a false positive propagates over further links is multiplicative at each hop by *Pmax*, given the large space of link IDs *m*!/(*m* - *k*)!. Recall that in our data-centric inter-networking scenario with massive *opportunistic caching*, such false positives are even less harmful; if there is space available, the data may be cached at these locations thus making network usage more efficient.

We simulated many system parameters (*k* in [4,5,6], *d* in [2,4,8,16,32]) and for pair-wise combinations we ran 500 tests counting the false positives by querying for the presence of 1000 link IDs of a randomly generated disjoint *Tset*. Figure 17 shows the false positive ratio results for *d* = 8 degrees of choice, *k* = 5 for the standard BF (where *d* = 1) and a *kdist* = 3,3,4,4,5,5,6,6 (nr. of bits set to one in different LITs) to build the 8 candidate BFs. The points in the graph are the mean values of the experiments and the error bars denote the standard error deviation.

As expected, the best performing BF (*fpr optimization*) is the one we can select when the Tset is known. The BF selected after observing its fill factor (*fpa optimization*) shows also better performance than a standard BF with no choice (d=1). After inserting 32 elements, an *fpa*-optimized BF reduces on average the *fpr* by around 0.5% (from 2.75% to 2.25%) and the best performing candidate brings the *fpr* another 0.5% down to under 1.8%. The performance gains increase with the number of elements inserted in the filter. The best BF candidate supports up to 43 elements before reaching the 5% false positive rate. In comparison, the standard BF reaches this *fpr* after inserting around 38 links. In practice, these extra 5 links mean that more subscribers can be reached without losing forwarding efficiency.

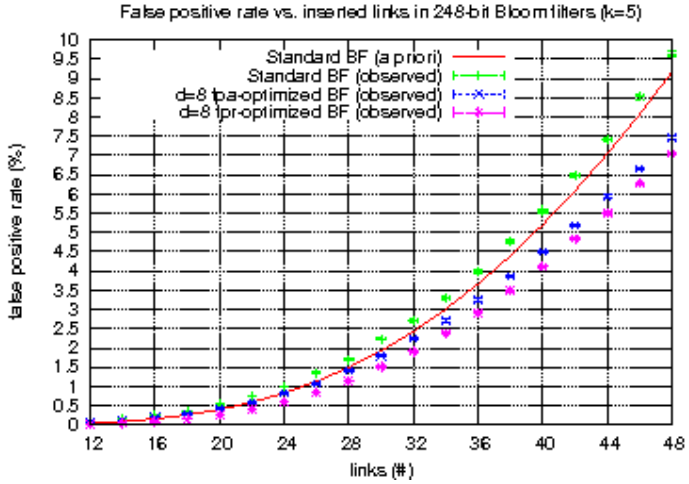

Figure 17: Having more choices when constructing BFs leads to improved BFs: approximately 1% better *fpr* in our region of interest (30-40 links or below 5% *fpr*).

Figure 18 and Figure 19 illustrate the performance gains of having 8 choices when constructing BFs. The percentage boxes illustrate the distribution of *k* among the experiments when the BF is selected following the minimal *fpa* criteria and when the actual best performing BFs are considered. The percentage curve highlights how often the BF used performed the best among the 8 candidates in the standard BF scenario (Figure 19) and when the candidate BF was selected after computing the lowest *fpa* (Figure 18). One can also appreciate the adaptive selection characteristic of *k_opt* (the optimal number of ones in the link identities) depending on the *fill factor*. With approximately 32 inserted elements, almost every second time the best performing BF was selected by just looking at the *fill factor*.

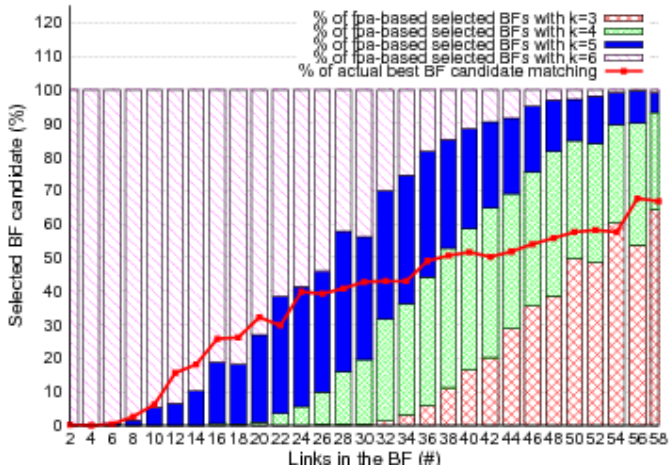

Figure 18: Performance gains of d-LIT constructed BFs and the distribution of k_opt with fpa-optimization.

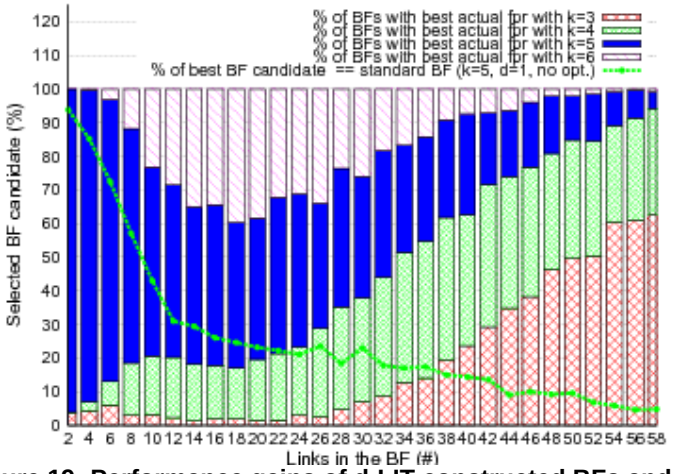

Figure 19: Performance gains of d-LIT constructed BFs and the distribution of k_opt with fpr-optimization.

4.2.3 Packet-level simulations

Another important part of our evaluation studies was to implement our line speed forwarding scheme in ns-3 [Hen2008]. First, it serves as a quick proof-of-concept implementation of the zFilter idea. Second, it investigates its behaviour on different topologies and provides a detailed picture about the effect of the different parameters in the network settings by complementing the previous analysis on basic Bloom filter performance.

Simulator implementation: Before we discuss the results, we briefly introduce our simulation platform. We implemented a zFilter-based forwarding layer as an extension to ns-3. We decided to implement the forwarding layer on top of the existing (simulated) Ethernet implementation. This was purely an implementation choice, as Ethernet or any other Layer 2 MAC is not required by the zFilters. The simulated forwarding layer supports multiple parallel forwarding tables and implements a basic loop prevention approach by caching packet headers in the forwarding elements that may cause loops by arriving on different incoming interfaces. The forwarding layer makes forwarding decisions by checking only the FId (the zFilter) and does not currently check the Rendezvous Identifiers. Over the forwarding layer, we implemented a simplified pub/sub-based internetworking layer to send publication data to a given rendezvous identifier. There is a separate topology module that has all the information about the network topology (network map and link identities) and can thereby construct the zFilter for any required publisher-subscriber set by computing the shortest path tree first and then bitwise ORing the appropriate link identities. It is able to select fpa-optimization by selecting the candidate with the lowest fill factor (while fpr-optimization is currently implemented by trying all the candidate zFilters in the network).

Topologies: Choosing realistic AS topologies to validate new forwarding algorithms is a challenging task, mainly due to the inaccuracies of the available data sets. One set of data we used are intra-domain AS topologies from Rocketfuel [Mah2002]. Though not completely accurate, they are a common (best) practice to approximate new forwarding schemes to real world scenarios. A second useful data set of router-level topologies is SNDlib [Orl2007], where contributors worldwide have published their network maps to the research community. For ultimate scalability analysis, we also experimented by generating topologies with 3000 nodes with the Brite tool (according to the Barabasi-Albert and Waxman models). Recent studies [She2008] have pointed out the limitations of the Rocketfuel data, suggesting that the number of actual physical routing elements is less than the one inferred by their measurement technique. We observed that Rocketfuel graphs are more complex than the largest topologies in SNDlib. For our purposes this fact (potential inaccuracy) is actually favourable, as it further stresses our forwarding mechanisms and investigates its performance in worst-case

scenarios. We are also aware of the situation that switching elements are not present in topology maps, something that could have an impact on the number of link IDs required as the number of network element hops increases. However, we believe that routers should be regarded as the significant networking units on which link IDs are defined, leaving layer 2 elements hidden from the topology function. The characteristics of the subset of the topologies used can be seen on Table 6.

Table 6: Some characteristics of the topologies used for the simulations.

AS	1221	3257	3967	6461	*TA2*
Nodes (#)	104	161	79	138	65
Links (#)	151	328	147	372	108
Diameter	8	10	10	8	8
Radius	4	5	6	4	5
Avg (Max) degr.	2 (18)	3 (29)	3 (12)	5 (20)	3 (10)

Stateless forwarding: In our experiments we investigated different settings for the number of forwarding tables (*d*), as well as for the number of users interested in a publication (*n*, where we always assume a single-publisher scenario, and, therefore, (*n* - 1) subscribers). Furthermore, we investigated the effect of different LIT-sets for the nodes (constant *k* = 5, and variable *k* in [3,3,4,4,5,5,6,6] distribution), and used different BF selection algorithms (*fpr*/*fpa* optimization). We carried out our tests on these well-referenced topologies and performed 500 runs for each parameter setting.

Table 7: ns-3 simulation results, for d=8, variable k distribution.

Users	AS	Links (#)		Efic. (%)		fpr (%)	
		mean	95%	mean	5%	mean	95%
4	*TA2*	8.6	12.7	99.92	100	0.02	0
	1221	9.7	13.6	98.08	88.89	0.37	2.13
	3257	9.6	13.5	99.83	100	0.02	0
8	*TA2*	15.6	20.0	99.6	94.12	0.2	1.59
	1221	16.8	21.3	97.78	90.89	0.54	2.02
	3257	17.9	22.9	98.95	91.3	0.28	1.25
16	*TA2*	25.7	30.9	97.92	91.67	0.83	2.67
	1221	27.4	31.0	95.51	88.22	1.28	3.17
	3257	31.3	36.7	92.37	79.58	1.76	3.86
24	*TA2*	34.1	38.8	95.2	87.18	1.95	4.63
	1221	36.1	41.0	92.06	83.33	2.65	5.19
	3257	42.2	48.1	82.27	67.69	4.17	6.96
32	*TA2*	41.4	46.0	92.04	84.31	3.46	6.46
	1221	44.0	48.3	88.22	78.95	4.32	7.45
	3257	52.2	57.9	71.47	59.34	7.3	10.41

Table 8: ns-3 simulation results for different configurations (standard d=1, fpa and fpr optimizations).

Users	AS	links	fpr_{fpa} (%)		fpr_{fpr} (%)		Stdrd
		mean	k_c	k_d	k_c	k_d	$k=5$
8	TA2	15.6	0.12	0.2	0	0	0.18
	1221	16.83	0.44	0.54	0.26	0.26	0.55
	3967	17.72	0.28	0.33	0.03	0.03	0.48
	6461	17.18	0.32	0.39	0.06	0.07	0.36
16	TA2	25.7	0.54	0.83	0.01	0.03	0.8
	1221	27.37	1.17	1.28	0.36	0.45	1.57
	3967	29.04	1.13	1.29	0.24	0.34	1.48
	6461	29.31	1.55	1.57	0.71	0.83	1.89
24	TA2	34.1	1.65	1.95	0.38	0.58	2.03
	1221	36.14	2.48	2.65	1.21	1.33	3.55
	3967	37.65	2.55	2.78	1.31	1.48	3.22
	6461	39.60	3.72	3.79	2.81	2.86	4.86

We present the gist of our simulation results on Table 7 and Table 8. Table 7 contains sample results using the *fpa* selection criteria with a variable distribution of *k*. The performance appears adequate in all of the topologies with up to 23 subscribers (approximately 32 links); forwarding efficiency is still above 90% in the majority of the test cases. This result is much better than multiple unicast, where the same links would be used multiple times by the same publication. For example, in AS3257 the unicast forwarding efficiency is only 43% for 23 subscribers.

Table 8 sheds light on the difference between the *fpa* and *fpr* algorithms. There is an interesting relation between the distribution of *k* and the optimization strategies: in our region of interest, *k_c* = 5 performs better than the variable *k* distribution (*k_d*). As expected, *fpr*-optimization successfully reduces the false positive rate and outperforms the non-optimized (*d* = 1) approach by 2-3 times in the scenarios with 16 users. The gain of using *fpa* instead of the non-optimized algorithm is clear, although not as significant as with *fpr*. Usage of *fpa* still holds some performance advantages and is very attractive because of its ease of implementation and low processing overhead.

The improvements can be also observed in the sample results of AS6461, see Figure 20, where the false positive rates are shown. For the sake of completeness, the number of links required in the delivery tree so as to reach the different sized subscriber sets is also shown. As already pointed out, for characterizing the *forwarding performance*, the forwarding efficiency metric reflects more accurately the bandwidth overhead. Huge differences may occur if there is a significant presence of high-degree nodes. Intuitively, the same false positive rate will yield lower forwarding efficiency in topologies with huge amounts of high-degree nodes than those without them. This is because the false positive rate shows how many times we are likely to get false positives when testing a Bloom filter, and in the case of the presence of a huge amount of high-degree hubs, we will simply test the Bloom filter inserted in the packet header more times per node. As each false positive leads to unnecessarily forwarding a packet on a link, the bandwidth overhead is increased.

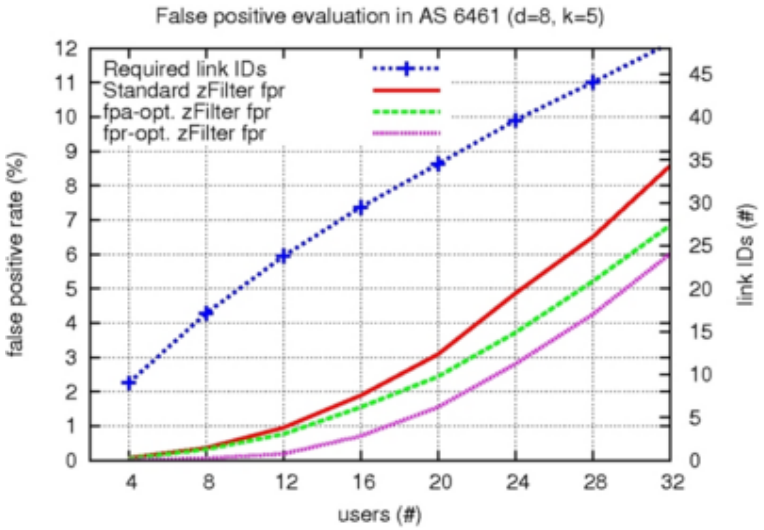

Figure 20: False positive evaluation for AS6461.

We can classify the Rocketfuel topologies into 2 different groups. One of them has only a few high-degree hubs (less than 10 neighbours), while in the other group the ratio of the high-degree hubs is at least 10% among all the nodes. For example, AS1221 belongs to the first category and AS3257 belongs to the latter. The corresponding false positive rate and the forwarding efficiency values (for different numbers of forwarding tables) are shown on Figure 21 and Figure 22 (we used 5 bits set to one in the LITs and the quickly computable *fpa*-optimization). To further point out the differences between the false positive rate and forwarding efficiency metrics, we also carried out tests in large topologies consisting of 3000 nodes (one was generated according to the Barabasi-Albert, while the other one according to the Waxman graph-model).

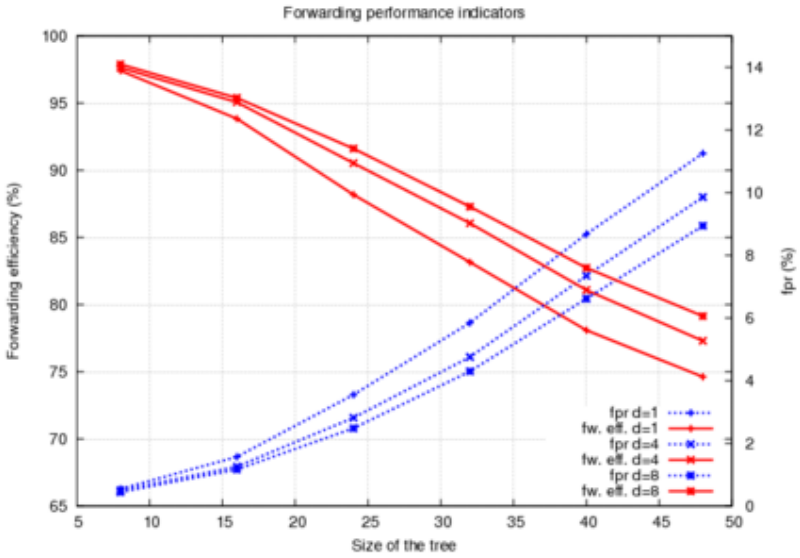

Figure 21: The connection between false positive rate and forwarding efficiency in AS1221.

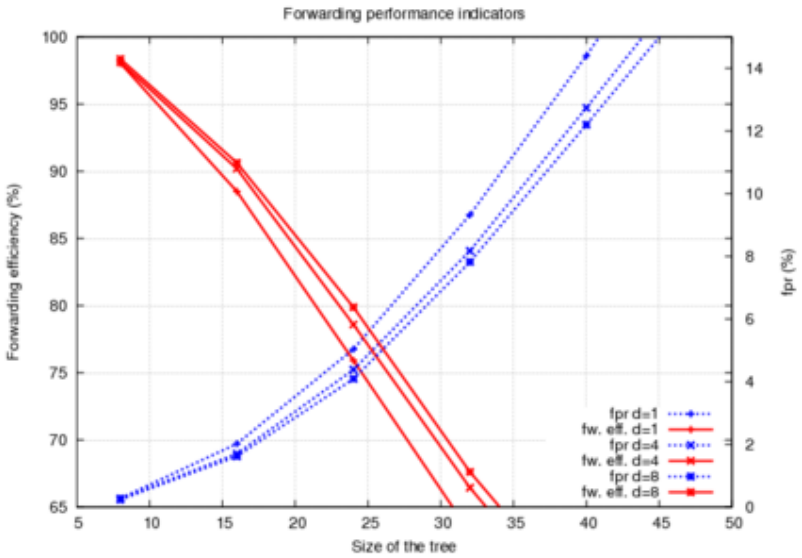

Figure 22: The connection between false positive rate and forwarding efficiency in AS3257.

The results showed that when the false positive rate was 1% the forwarding efficiency was already only 90%, and only 8 entities were covered by the delivery trees with these characteristics (1 publisher and 7 subscribers). This phenomenon is due to the fact that these topology models have some nodes with very high degrees (scale-free characteristics). These results helped us reveal a new design consideration (applicable also for virtual links, see later): it is beneficial to select the link identifiers so that those connected to high-degree nodes have more bits set to one than those connected to low-degree nodes. The exact tuning of this consideration, however, is left for further study.

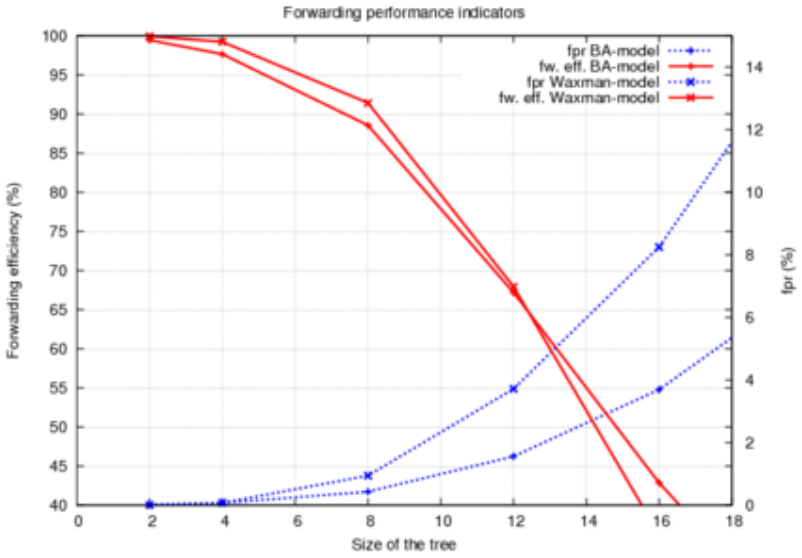

Figure 23: Forwarding efficiency and false positive rate in large scale-free topologies.

We must also mention another very important property of zFilters. Readers may expect tests revealing how many trees can co-exist in the network simultaneously. However, this is not constrained by the forwarding functionality (only by the processing capabilities of the tree-computing module in the topology manager). As the links are inserted into the zFilters, trees are only present in the packet headers, and therefore are completely independent from each other. Hence, the number of simultaneous active trees can be arbitrarily high without affecting network performance.

Stateful forwarding: In networks with scale-free properties, a large part of the traffic flows between high-degree hubs. We experimented with the effects of installing virtual trees covering different parts of the network. We built virtual trees from the publisher towards the core and between the high-hubs, increasing the performance only slightly, as virtual trees substituted only a couple of physical links.

Significant performance enhancements can be achieved if we install virtual trees rooted at (high-degree) core nodes and covering a set of subscribers, thereby avoiding the presence of many LITs in the zFilter. To avoid harmful false positives with the introduction of virtual trees, we may use LITs with slightly more bits set to one than for physical links (e.g., 10 bits instead of 5 bits). The results on Figure 24 show that dense multicast is supported with more than 95% forwarding efficiency even if we need to cover more than 50% of the total nodes in the network (see Table 7 for comparison).

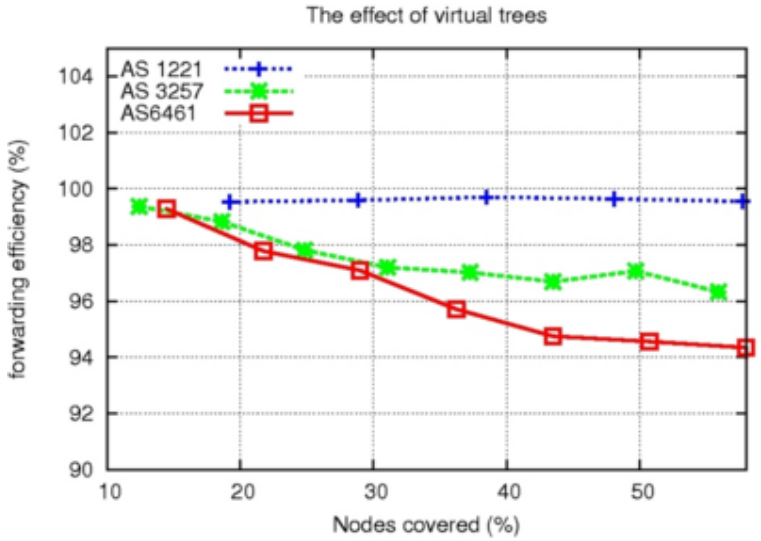

Figure 24: forwarding efficiency achieved by installing virtual trees into the network (d=8, k=5, fpa-optimization).

4.2.4 Forwarding table size

Assuming that each forwarding node maintains *d* distinct forwarding tables, with an entry consisting of a Link ID and the associated output port, we can estimate the amount of memory needed by the forwarding tables:

$$FT_{mem} = d \cdot \#Links \cdot [size(LIT) + size(P_{out})]$$

Considering *d* = 8, 128 links (physical and virtual), 248-bit LITs and 8 bits for identifying the outgoing port, the total memory required would be 256 Kbits, which easily fits on-chip.

The *d*-LIT enhancement comes at a price of a factor of *d* in fast memory size. Although this memory size is small due to the source-based routing approach, there is room to design a more efficient forwarding table. Since LIT entries will only contain *k* bits set to one, a sparse representation would only require storing the *k* positions of the bits to be ANDed in the test

membership on packet arrival. Thereby, the size of each LIT entry is now reduced to only k*log2(LIT) and the total forwarding table size is thereby reduced to 48 Kbits. Furthermore, if we use a segmented hash approach where each hash to generate the LIT is limited to a region of m/k bits, each LIT entry would require just k*log2(LIT/k) bits, which in our example translates to a total forwarding table size of 36 Kbits. Our evaluations on the NetFPGA implementation suggest that the forwarding performance is not penalized when adopting the sparse representation of the LIT entries.

4.2.5 Network cost

The optimal balance between the amount of virtual links and false deliveries can only be found by considering network economics. The direct prices can be abstracted through the *average price of sending a packet over a link*, the *price of one virtual link per interface*, and *the price of caching memory on forwarding nodes*. The ratio of these, the distribution of the number of subscribers, the structure of the network graph, the typical locations of publishers, the number and location of in-network caches, and subscriber churn are all input parameters needed for cost optimization.

Given that the different network architecture proposed by the PSIRP project is likely to change traffic patterns compared to those seen today, many of these parameters are almost impossible to estimate. We merely point out that as memory prices are likely to fall faster than communication prices also in the future, the optimal structure is likely to increase the demand for caching and explicit state over time.

4.2.6 Discussion

To support more simultaneous subscribers than what can comfortably fit into a single zFilter, two choices can be considered. First, we can create virtual trees to maintain the fill factor and to keep the redundant deliveries under control. This comes at the price of control traffic and the increase of state in forwarding nodes. Second, we can send multiple packets: instead of building one large multicast tree we can build several smaller ones, thereby keeping the zFilters' fill factor reasonable. The packets will follow the desired route with acceptable false delivery rates, but exact copies will pass through some links multiple times. Depending on the scenario specifics, this can result in more bandwidth waste than in the case of a single larger tree.

Assuming a Zipf distribution of the subscribers [Liu2005], with m = 248, our forwarding fabric requires no state for the large majority of topics, requiring virtual links or multiple sending only for the few most popular topics. Furthermore, as we can easily combine the stateful and stateless methods, we can readily accommodate a number of changes in the popular topics before needing to signal a state change in the network.

Our current understanding of the dynamics of the operation of zFilters (and the effect of the different parameters of the design space) is shown on the loop diagram of Figure 25. The diagram shows the forces present in the zFilter-based forwarding design, and shows how different parameters affect each other (+ means that the increase in the value of one parameter increases the other, while - means that the increase in one parameter causes the other parameter to decrease). Until now, we mainly explored the limits of the original zFilter-approach (without virtual states) and have preliminary understanding that how virtual states may increase the performance. However, the usage of caching and quantifying its benefits is left for further study. This loop diagram is also an important step towards understanding the cost structures of the network operation (see previous section).

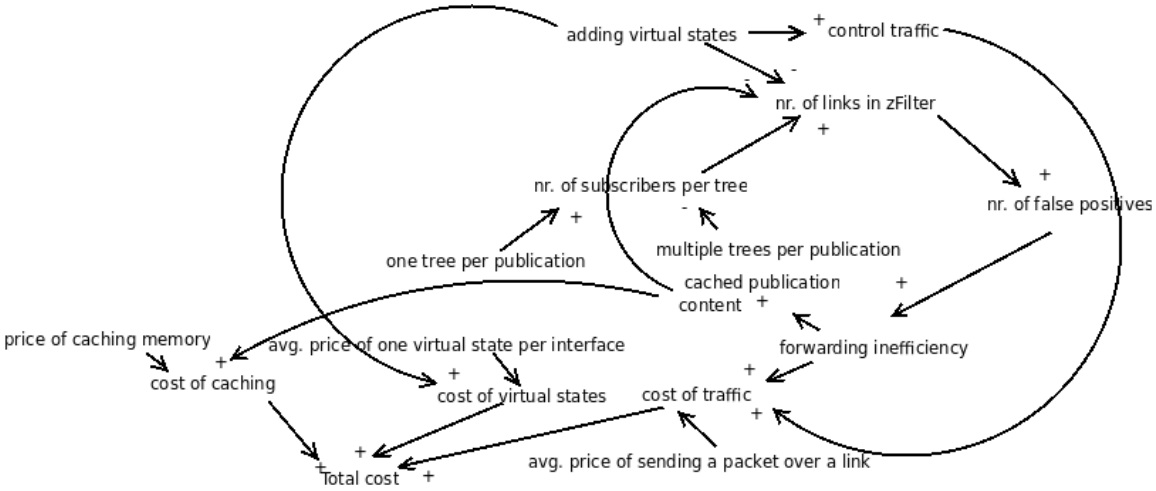

Figure 25: Exploring the design space of zFilters.

4.3 Network Coding Performance Evaluation

Due to its generality and potential, network coding has received a large amount of interest. It has been envisioned as a potential solution for increasing throughput and enabling higher data rates than the conventional unencoded communication or the application of source coding alone. Network coding has also been discussed in the context of the PSIRP architectural work, but concrete decisions on its application have been postponed until more information on the expected performance issues and related architectural concerns are available. In this section we provide first evaluation results towards that direction.

While the majority of existing work has focused on analyzing wireless network coding mostly using algebraic tools [Ahl2000], [Koe2003], [Ho2006], [Lun2008], [Kat2005], [Cho2003] our goal is to evaluate network coding behaviour in middle sized networks under more realistic circumstances, including point to point connections. We focus on two commonly considered network coding approaches, namely XOR and linear network coding (see deliverable D2.1 for a thorough background discussion) and evaluate the performance of these approaches under different conditions using the ns-3 network simulator [Hen2008]. We specifically study and compare the decoding performance of these two coding schemes assuming, in a sense, optimal packet combination, as discussed in the following.

For simple topologies and packet distributions, the decision upon the XORed packet combination which provides the optimal decoding gain is rather straightforward. In more complex scenarios of XOR network coding, such as the one shown in Figure 26, a router having packets *p1p2p3p4p5p6...* will optimize decoding from the viewpoint of the receivers, in the sense that all of them will be able to decode missing packets, combining *p1p2p3p4* together. This leads to a simple optimization rule followed in our implementation: a router will maximize coding gain by making *n* packet combinations, if all recipients already have n-1 packets of the same combination. We assume this information is available either via overhearing or some type of explicit signalling.

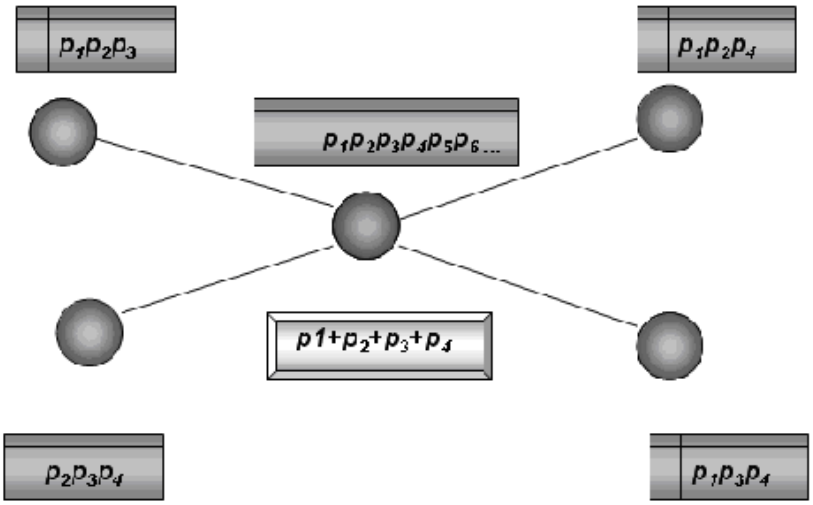

Figure 26: Maximizing coding gain by making combinations of n packets if all receivers already have n-1 packets of the same combination.

4.3.1 Implementation scenario

In our implementation, the XOR coding/decoding process is inserted above the network layer, since the network layer header contains information about a packet's source, which is necessary for coding/decoding. Ns-3 has the notion of a PointToPointChannel as a communication link and a PointToPointNetDevice representing the network interface card and its device driver. In order to facilitate the connection between PointToPointChannel and PointToPointNetDevice ns-3 introduces the PointToPointHelper functionality. We exploit ns-3's helper functionalities to build point-to-point links between nodes in the network, with desired data rates and channel delays.

The implementation of the ns-3 node class was extended by adding buffers to store packets of interest. Every node keeps track of received and sent packets, based on which it gains knowledge about packets distributed in the neighbourhood. Thus, during coding/decoding every node relies on local information about the overall packet distribution, without contacting other nodes. This information is important for optimization, since not every packet combination will lead to successful data recovery and network coding gains. In our scenario, we aim to maximize this benefit by following the rule shown in Figure 26 provided that this opportunity exists. Therefore, each packet has to be created in such a way that every neighbour, after receiving the packet combination and performing a XOR operation with the packets it possesses, can retrieve the missing packet. This is possible only if the node on receiving the *N* packet combination, already has buffered the *N*-1 packets of the received combination.

In our case, a node having *M* point to point interfaces (neighbours), in order to benefit from broadcast will make *M* packet combinations, ensuring that every neighbour already has *M*-1 native packets of that combination. Before combining, the node performs a look up in its received packet buffer to find all packets with *M*-1 sources (packets with the same unique ID but with different source addresses). Such packets, having the same ID but *M*-1 different source addresses denote the same packet received from *M*-1 different sources, thus it is one possible packet to be put into the final combination. Finally, the node makes all possible M (number of neighbours) combinations out of packets that have *M*-1 sources, and broadcasts it. Information about the combined packets is stored in the packet metadata combination to facilitate the decoding process. Upon receiving the packet, the node performs the look-up in the buffer one more time to find the combination of packets that has to be XORed with the

received combination in order to retrieve missing packet. The validity of the resulting packet is checked by performing a checksum computation over the payload [Bra1988].

In our implementation, XOR coding/decoding at the sender/receiver side is an auxiliary process to "common sending/receiving". Thus after (say) receiving the packet, the node first checks if the incoming packet is part of a combination of packets. If not, it performs an ordinary packet processing as in the case where no coding/decoding mechanisms are present. In the case of sending the packet, the node first checks if there is a possibility to send a packet combination based on the information it has about the packet distribution. Otherwise, it sends packets following the regular pattern. In other words, it is opportunistic since it performs the coding operation only when such a possibility exists.

On the other hand, for random linear network coding, packet generation is carried out probabilistically. Our implementation of this model relies on the ns-3 random number generators for creating coefficients and selecting packets to be combined. A node generates an output packet as a linear combination of randomly chosen packets from its buffer and the random coefficients *g1,g2,...,gn*. The structure of the packet combination is recorded in the form of a vector of coefficients and packet IDs. The vector is then stored in the packet metadata. Based on this information, the decoding side is aware of the composition of each packet and performs Gaussian elimination after a sufficient number of combination packets of the same encoded structure is received (*n* packets with linearly independent vector of coefficients).

Our network coding scenario resembles simple datagram service communication between nodes. The capacity of point to point channels is initially set to 4.5Mbps, with the delay of 5ms. The network topology varies based on the number of nodes involved, ranging from 25 to 121 nodes, and the type of connectivity (grid or butterfly networks). We exploit the case of randomness in building a network topology by applying a certain probability of link existence between two nodes, making our conclusions based on the two extreme cases, that is, setting the connection probability between two nodes to 10% and 90%. We extend our observations to the case of a 55% connection probability. Additionally we are monitoring the influence of different traffic patterns. The basic traffic model relies on the OnOff Application of ns-3, which switches transmission and idle states according to predefined On and Off intervals. During the On interval, CBR traffic with a data rate of 448Kbps and packet size of 210 bytes is generated, while the Off interval denotes no traffic. By changing the parameters of the PointToPoint helper we are varying data rate and packet delay. Different traffic patterns are obtained by altering the transmission probability for every node in the network. As in the case of the variable topology pattern, we examine transmission probability 10% and 90% for every node. The efficiency of network coding is evaluated in the terms of packets additionally decoded at the receiver side compared with the case without coding.

4.3.2 Results

Simulation results show that the benefits of both types of network coding highly depend on the different parameters governing the network topology and the coding process. In both cases, XOR and linear network coding results rely on the packet distribution and the number of packets that each node stores in its buffer. One of the important concerns of the network coding implementation is the mean packet latency that coding and decoding introduce. Increasing the size of packet buffers increases the amount of data that each node has to be aware of and leads to higher decoding delays. For XOR coding, having a relatively small number of packets P with M-1 sources in the buffer, the combination of N (number of interfaces) over P becomes very large (due to P!/N!(N-P)!). The task of finding a combination which results in benefits for all neighbours becomes a time consuming process. On the other hand, if the size of the packet buffer is not sufficiently large, the node will not have enough information based on to perform beneficial coding. Results further show that latency is not heavily influenced by the underlying topology, but varies primarily based on the channel parameters, that is, data rate and, of course, the delay it introduces. For the fixed channel

delay, latency added by XOR coding decreases when increasing the data rate of transmission. Overall increase in packet latency due to the XOR network coding/decoding in our implementation is negligible for most of the applications as illustrated in Figure 27. The figure also shows the changes in packet latency based on applied packet data rate.

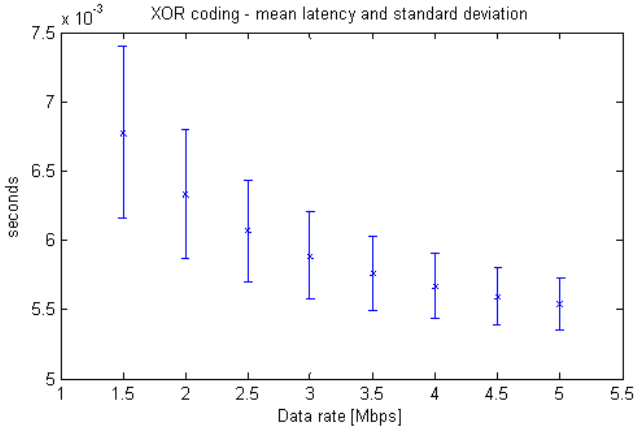

Figure 27: XOR network coding - mean packet latency and standard deviation (36 nodes network).

In order to compare the latency introduced by XOR network coding and linear network coding we fix the channel parameters, and vary only the number of packets in the linear network coding packet combination. Our results show that linear network coding performs slightly better than XOR coding in terms of packet latency for the same network settings. Figure 28 shows the linear network coding packet latency for the same network setting as in the case of XOR coding, with a fixed data rate of 4.5Mbps.

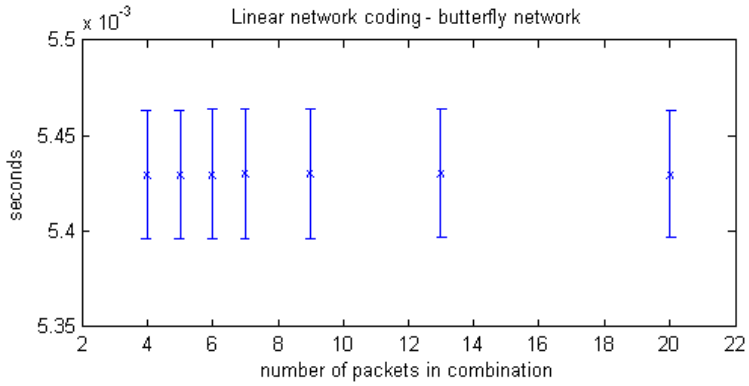

Figure 28: Linear network coding - mean packet latency and standard deviation (36 nodes network, 4.5Mbps data rate).

Moreover, Figure 28 illustrates the influence of the packet combination size on latency. For relatively small packet combinations Gaussian elimination does not introduce additional computational delay when the number of packets combined is less than 20. On the other hand, the number of packets combined as well as the underlying topology have an impact on linear network coding performance in terms of decoding gain as illustrated in Figure 29.

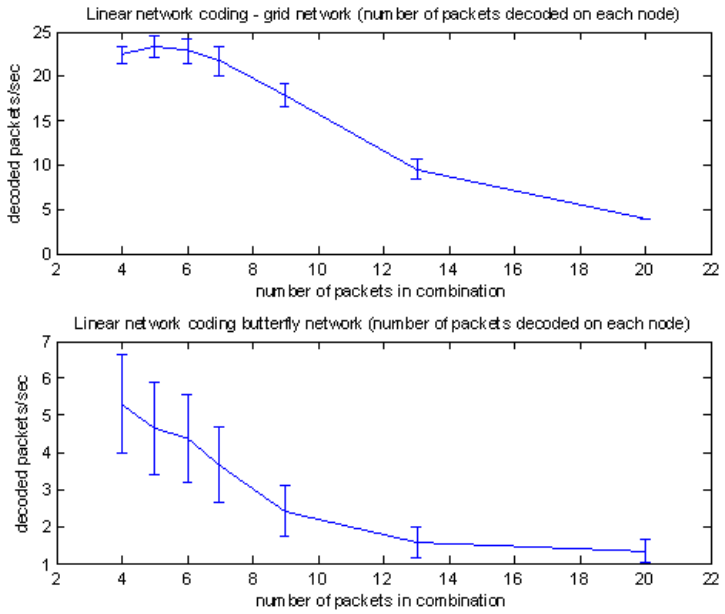

Figure 29: Linear network coding - mean number and standard deviation of packets decoded on each node (36 nodes grid and butterfly network).

In the cases of grid and butterfly networks, increasing the number of packets put into the combination might lead to a performance degradation due to the higher complexity and longer time needed for a sufficient number of additional packets to be received. Network coding performance under different traffic patterns is another important property we evaluate. Defining a certain probability for each node to broadcast data determinates its participation in overall traffic as well as in network coding. In our implementation we examine the relation between a given broadcast probability and network coding performance. Results show that changing the traffic pattern influences the overall performance of network coding in terms of efficiency, while the packet latency remains the same.

Figure 30 shows the results for a 90% broadcast probability where the mean number of decoded packets is proportional to this predefined probability (approximately 90% of network coding benefit compared to the case where all nodes broadcast).

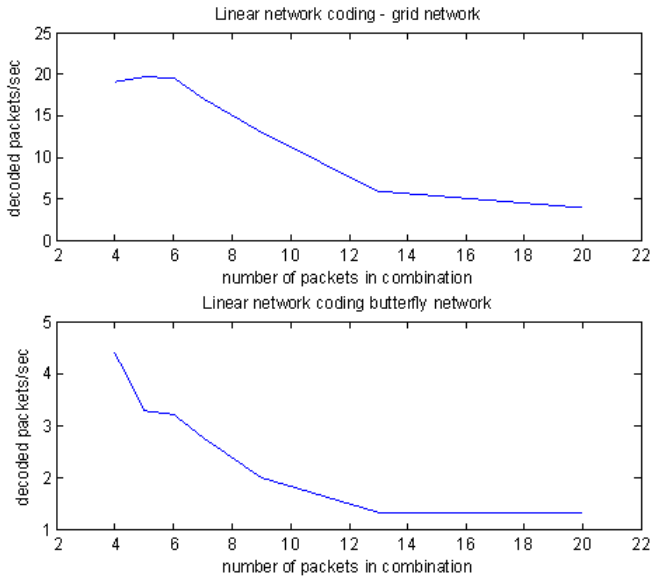

Figure 30: Random traffic pattern (node broadcasts with 90% probability) mean number of decoded packets, 36 nodes network.

The standard deviation of the number of decoded packets is high due to the large possible variations of number of packets sent and received on the same node (even if the node is not sending, it is able to receive packets from neighbours and to decode). Setting the broadcast probability to be very low, e.g., 10% in our implementation, leads to negligibly low network coding benefit.

Another illustration of system randomness applied to our implementation is topology oriented. We generate random topologies by defining the probability of link existence between nodes and monitor network coding behaviour under these circumstances. The results are similar to the case of random traffic patterns, in the sense that the network coding benefit is proportional to the given link existence probability, and it degrades dramatically, and can even be disregarded, for low probabilities. In contrast to the random traffic pattern case, the standard deviation of number of packets decoded is relatively small, since there is no possibility for a node to send/receive to/from existing links as shown in Figure 31.

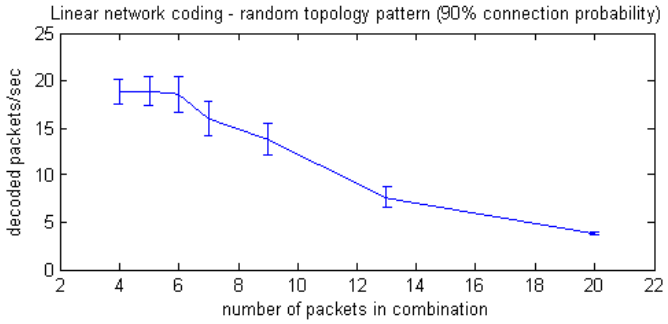

Figure 31: Linear network coding, random topology pattern (90% probability of link existence between two nodes) mean number and standard deviation of decoded packets, 36 nodes network.

In order to better understand the dependency of the decoding benefit on network connectivity, we extend our observations to larger networks and "critical" connection probabilities between nodes. We choose a 55% link existence probability to avoid cases of obvious network disconnections due to low probability, e.g. 10%, and cases of high node connection probability, e.g. 90%, where circumstances are rather the same as in completely connected networks. Moreover, we apply this connection probability to 8 grid networks of different sizes i.e. 25, 36, 49, 64, 81, 100 and 121 nodes. The decoding gain is not anymore proportional to the connection probability as it was in the case of high connection probability. Results show that the decoding curve has the same shape as for completely connected networks, but the network coding gain is much lower due to network disconnections and nodes lacking sufficient information to perform better coding/decoding as shown in Figure 32.

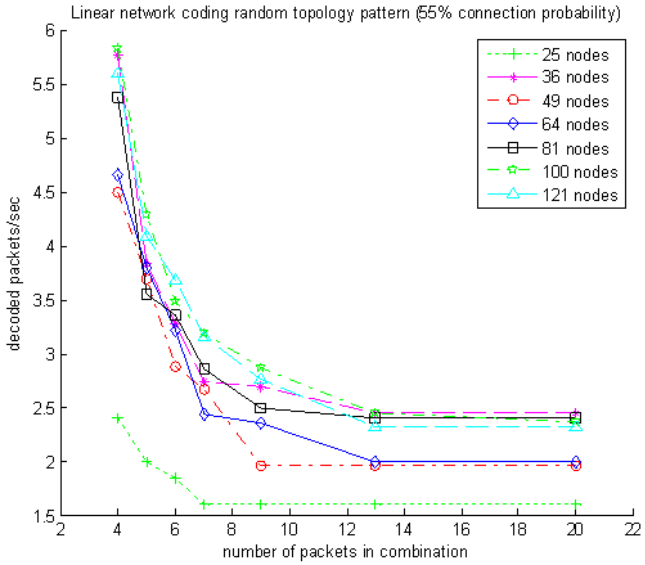

Figure 32: Linear network coding, random topology pattern, mean number of decoded packets (55% probability of link existence between two nodes).

The size of the network is another important parameter influencing the decoding results. Increasing the number of nodes involved in communication leads to increasing the number of packets decoded, for the same traffic and topology parameters applied. Figure 33 illustrates the impact of network size on number of packets decoded on each node for grid networks built upon 25, 36, 49, 64, 81, 100 and 121 nodes.

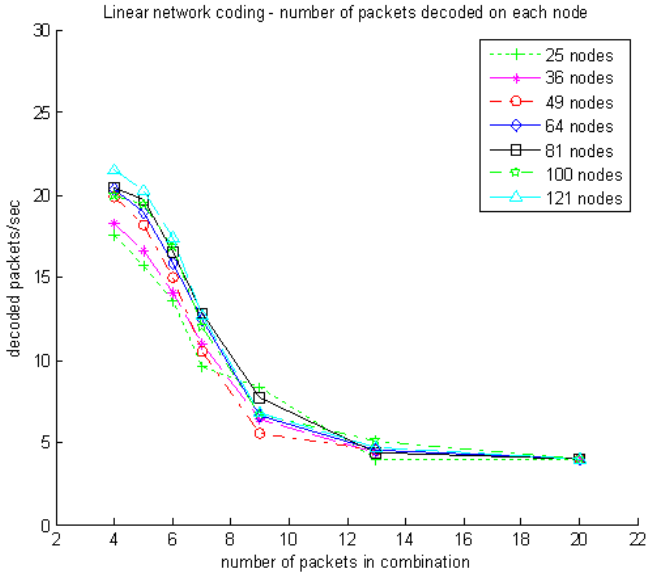

Figure 33: Linear network coding, influence of network size on network coding gain.

4.3.3 Conclusions

The parameters and topologies that have been analyzed above are just one small portion of network coding design space. A similar evaluation could be performed in the wireless network case as one of potential extensions to the current work. Moreover, in our implementation we have assumed that the nodes have infinite buffers for storing packets and performing lookup for decoding. Thus, one of the interesting objectives to investigate would be the influence of limiting the buffer size on overall network coding performance. There are many other potentially interesting network settings to be investigated in the future in order to get a more complete picture of network coding manners and its applicability in specific use cases.

4.4 Performance evaluation of the overlay PSIRP variant

4.4.1 Introduction

One alternative approach for the investigation of the intrinsic characteristics of publish/subscribe and multicast network architectures focuses on an overlay variant of the PSIRP paradigm based on Scribe [Cas2002], an overlay publish/subscribe system providing multicast routing, which in our case is based on the Pastry DHT [Row2001]. This approach presents two significant advantages: it explores the potential of an overlay implementation with clear deployment advantages, while, at the same time, it allows the investigation of the aforementioned inherent properties based on protocols already implemented in publicly available evaluation tools.

In our work, we used OverSim [Bau2007], an overlay network simulation framework for the OMNeT++ simulation environment [Var2009]. The OverSim framework provides implementations of several overlay schemes and applications, among which are the implementations of Pastry and Scribe that our study is based on. The framework also provides a variety of underlying network structures varying from simplistic ones (SimpleUnderlay) where no actual routing is performed, to more complicated ones (IPv4Underlay) where the complete protocol stack, provided by the INET protocol framework [Var2009], is in operation.

In this context, we first investigated the potential benefits of network assistance in the provision of overlay multicast services for massive content distribution. To this end, we paid special attention to the development of a scalable and realistic simulation environment, primarily exploring the underlying topology models. Based on this testbed, we studied the

graph oriented properties of the overlay trees and demonstrated the benefits of the network support for overlay multicast.

However, since the multicast tree properties are insufficient to characterize the dynamic performance of actual applications, we further turned our attention to the specification of a realistic content distribution application model to serve as a benchmark for the benefits envisioned by the PSIRP architecture. Based on the popularity of P2P content distribution applications, we chose BitTorrent as the main application model for benchmarking. In order to be able to perform a comparison study against the performance of our overlay multicast scheme regarding content distribution, we implemented the BitTorrent suite of protocols for the OMNeT++ simulation environment and further created a churn generator module for the OverSim environment based on an analysis of real BitTorrent traces.

Finally, based on the OMNeT++,/INET/OverSim simulation environment we explored the potential benefits of this overlay PSIRP variant in the support of node mobility via widespread support for multicast.

In the following we provide a detailed description of the performance evaluation tools developed, as well as the results regarding the performance of the proposed overlay-based PSIRP variant.

4.4.2 Developed Tools

4.4.2.1 Topologies

Our initial concern in building a simulation environment related to the underlying network topology. OverSim's SimpleUnderlay model provides a scalable routing substrate since no network protocols are actually in operation. In this model, packets are directly sent to end hosts by simply using a global routing table, with packet delivery delay being determined by the two communicating ends' distance in the Euclidean space. Furthermore, each end host can be assigned to a logical access network for which the access delay, bandwidth and packet loss characteristics may be set. Though it has been shown that this model provides a scalable solution for the simulation of large numbers of overlay nodes [Bau2007], it suffers from serious limitations: the lack of protocol functionality and step-by-step routing are major drawbacks of the model, since important aspects of a real system are neglected, such as the queuing of packets in intermediate nodes (routers) and therefore the packet delays, and even losses, that arise due to network congestion. As a result, this model cannot be used to evaluate the dynamic performance properties of realistic applications.

On the other hand, the IPv4Underlay model provides a good approximation of real networking conditions by incorporating the operation of almost all widely deployed networking protocols. We therefore focused on this model for our work, exploring its memory and processing time requirements for the simulation of very large network topologies. Apart from scalability, the other major drawback of this model is that it lacks support for routing policy weights, such as those produced by the Georgia Tech Internet Topology Model (GT-ITM) [Zeg1996] that has been used for previous studies of Scribe performance [Cas2002], [Cas2003a]. Instead, the model employs a non-weighted shortest path algorithm (Dijkstra's) which calculates the shortest paths between any pair of network nodes, regardless of their placement on the network. Hence, in contrast with reality, it is possible for a path between two routers in a single stub network to pass through several transit routers, something very likely to influence the results produced, especially when it comes to routing issues. By not taking routing policy weights into account, routing paths may become shorter than in reality, and since Pastry employs proximity metrics in the selection of overlay neighbours, this could result in the selection of the wrong node as an overlay neighbour.

In order to avoid routing inaccuracies, we constructed a conversion tool that allows the use of GT-ITM topologies within the OverSim platform. Our tool was implemented as an extension to the BRITE topology generator export tool [Var2003] which already allows the parsing of GT-ITM topologies and the conversion of BRITE topologies into OMNeT++ format. However, the

existing tool did not provide any support for weighted topologies, nor did it make a distinction between transit and stub routers, producing flat topologies. We solved these problems by piggybacking the routing weights inside the channel definition of the produced OMNeT++ topology, using fields whose values are not provided by the GT-ITM model, but are instead later read from configuration files. Furthermore, we incorporated the distinction between transit and stub routers in the tool and translated it into IPv4Underlay's distinction between backbone and access routers. The support for routing policy weights was completed by employing a weighted shortest path algorithm, leading to a platform that captures all the intricacies of GT-ITM and is therefore comparable to earlier simulation studies based on it.

Despite these important enhancements, the resulting GT-ITM based IPv4Underlay model presents two other significant limitations. The first is revealed when considering the locality properties (with respect to the consumption of ISP-specific resources) of data exchanges in P2P content distribution applications, such as BitTorrent. It has been shown that BitTorrent's network-agnostic peer-wire protocol has an adverse impact on capacity related ISP costs by allowing downloads from peers residing in external domains even though the desired data are already present locally [Kar2005]. This has triggered several research efforts (e.g. [Xie2008], [Bin2006]) which would benefit from a simulator providing the flexibility to study ISP level aspects of the protocol performance. Hence, as OverSim's IPv4Underlay model provides no access to such information, we further enhanced our topology conversion tool to also preserve the unique Autonomous System numbers [IAN2009] produced by BRITE and to export them to a separate configuration file so that each router, as well as each attached end host, can be assigned the corresponding AS number. Direct access to this information is provided to the simulation programmer, facilitating the investigation of inter-domain routing issues both for our overlay multicast routing scheme and the BitTorrent protocol.

The second limitation is that OverSim does not make any distinction between the uplink and downlink characteristics of access links (e.g bandwidth). However, this distinction is important in providing a realistic networking environment, since current typical access technologies (e.g. ADSL) do present this asymmetry. This issue is further signified by the fact that bandwidth heterogeneity results in systematic unfairness of the peer-wire protocol's download rate based tit-for tat mechanism [Bha2005]. Hence, we further enhanced OverSim's IPv4Underlay model to support a range of channel characteristics that can be different for the two directions of each access link. Specifically, the simulation programmer is able to specify different channel options for the uplink and downlink of each access link type, along with the fraction of the total access links across the entire network that each channel type is assigned to. Table 9 shows an example setup for the access links of a simulation scenario. The fraction values indicate that, for example, 40% of the participating end hosts can download data with a maximum rate of 8 Mbps while they can upload data with a maximum rate of 1 Mbps

Table 9: Bandwidth distribution of access links.

Uplink (Mbps)	Downlink (Mbps)	Fraction
1	4	0.20
1	8	0.40
2	16	0.25
2	24	0.15

4.4.2.2 BitTorrent Simulation Module

As mentioned above, implementing BitTorrent for the selected simulation environment was a decision dictated by the need to have a point of reference for content distribution performance, so as to be able to compare against it the the performance of the overlay PSIRP variant The specific selection of BitTorrent as a reference content distribution application was driven first by its popularity, rooted at the perceived low download times. Most importantly though, BitTorrent emerged as an efficient solution for large scale content distribution which alleviated

content providers from the burden of the centralized content distribution, a burden that is further exacerbated by the lack of network multicast support on the Internet. On the other hand, however, in BitTorrent the redundant unicast transmissions of the same packet are not avoided, they are just distributed between the peers. While efficient from the content provider and end user perspective, BitTorrent imposes significant overhead for the network providers. Hence, with the emergence of PSIRP, a multicast content distribution scheme re-emerges as a potential solution for mass content distribution, further promising a more efficient use of network resources, so that a straight comparison of both resource utilization and user perceived performance with BitTorrent becomes more reasonable.

However, this comparison was not considered to be a straightforward procedure with the currently available tools, as current BitTorrent simulators either consider coarse-grained representations of the underlying network, thus reducing the realism of the simulation, or omit many important features of the BitTorrent protocols. Therefore, we engaged in the implementation of a full featured and extensible BitTorrent protocol module for the OMNeT++ simulation environment. Our implementation was primarily based on the unofficial BitTorrent Protocol Specification [Bit2009a] and secondarily on the only authoritative document available [Bit2009b]. The reason for this choice was that the latter describes the entities involved in the protocols, the basic concepts, and the rudimentary transactions among them, but it lacks the behavioural and implementation details of the entities.

Our implementation includes both the Tracker and the Peer-Wire protocols. Regarding the latter, which is the corner stone of BitTorrent functionality, we implemented all basic features including several optional ones, such as super-seeding and the end-game mode. Our target was to create a highly configurable implementation providing the capability of tuning several protocol aspects from the configuration files. Table 10 shows the list of configuration parameters available and Figure 34 provides an overview of our implementation architecture.

Table 10: BitTorrent peer-wire protocol parameters.

Parameter	Default Value
file size (MB)	700
piece size (KB)	256
block size (KB)	16
DHT port	-1
pstr	BitTorrent protocol
pstrlen	19
keep alive (sec)	120
have supression	true
choking interval (sec)	10
downloaders	4
optUnchokedPeers	1
optUnchoking interval (sec)	30
seederDownloaders	4
seederOptUnchokedPeers	1
rarest list size	5
minNumConnections	30
maxNumConnections	55
timeToSeed (sec)	0
request queue length	5
super seed mode	false
end game mode	true
maxNumEmptyTrackerResponses	5
newlyConnectedOptUnchokeProb	0.75
downloadRateSamplingDuration (sec)	20

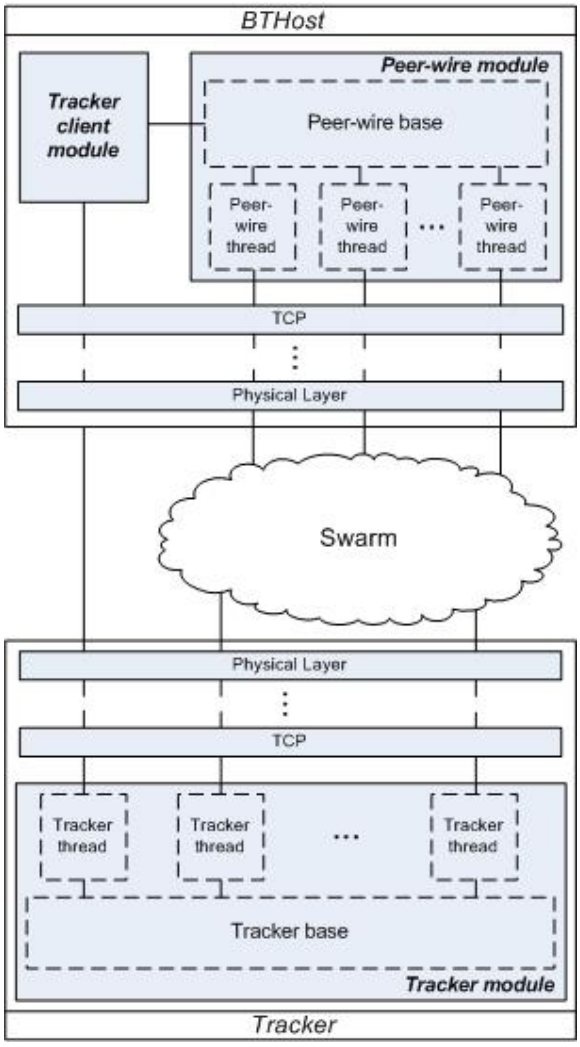

Figure 34: BitTorrent module architecture.

4.4.2.3 BitTorrent Churn Model

In order to simulate the dynamic behavior of a content distribution application, end hosts need to be randomly introduced into the network both in a topological and in a chronological sense (i.e., they need to be placed at random nodes in random time points). The churn models provided by the OverSim platform, together with the underlying IPv4Underlay configuration mechanism, constitute a flexible mechanism for dynamically deploying end hosts in the network. However, the churn models available in OverSim were not designed to reflect the arrival processes of real applications. Instead, they provide a generic mechanism for the arrival process, and several distributions describing the duration of a peer's presence in the network. Since in BitTorrent this duration depends on protocol operation rather than on a predetermined distribution, we focused on the arrival process. Using the OverSim churn generator mechanism, we implemented the BitTorrentChurn model that reproduces the arrival process of BitTorrent clients presented in [Guo2007]. In this study, based on the analysis of BitTorrent user traces, it was observed that the peer arrival rate for a torrent follows an exponential decreasing rule with time *t*:

$\lambda(\tau)=\lambda(0)e^{-t/\tau}$,

where λ(0) is the initial arrival rate when the torrent starts and τ denotes the file popularity. Based on this distribution it can be shown that N_{all}=λ(0)t, where N_{all} is the total population size. Hence, by retrieving values for N_{all} and λ(0) from the configuration files, our model can generate random arrival times for each BitTorrent peer. It is stressed that this process can also be followed in the case of the overlay multicast assisted content distribution alternative application to be developed by PSIRP, or indeed, with any application.

4.4.3 Overlay Multicast PSIRP variant

4.4.3.1 Router Assisted Overlay Multicast

This absence of IP multicast support has led to the emergence of overlay solutions that do not require network support, such as those based on DHT substrates. In DHT substrates like Pastry [Row2001] a uniform identifier space is distributed among the participating nodes; these are regular end hosts that use the underlying IP transport transparently to the routers. These nodes co-operate to efficiently route data tagged with a specific identifier to the node assigned with that part of the identifier space. To facilitate this process, each DHT node maintains overlay routing state that allows it to relay a received packet to another node whose part of the identifier space is closer to the packet's identifier, until the node actually responsible for that identifier is reached. The advantage of such schemes is that both the amount of routing state required per node and the maximum number of hops required to reach any other node scale logarithmically with the number of nodes, a critical feature in the context of mass content distribution.

On the other hand, packets following overlay routes no longer follow the shortest path towards their destination node. Unlike other DHT schemes (for example, Chord [Sto2003]), Pastry attempts to minimize this side-effect, a property that motivates its use in our scheme. By employing proximity metrics, such as the number of IP hops or the round trip time, Pastry takes network locality into account: among the possibly many DHT nodes that are closer to a packet's identifier, and which could thus continue relaying the data, Pastry chooses the closest one with respect to the employed proximity metric.

The Scribe [Cas2002] system achieves multicast distribution over any DHT substrate, not necessarily Pastry [Row2001], by mapping the name of each group to an identifier and making the node responsible for that identifier the RV point of the group. Receivers join the group by sending a join message towards the group identifier; as the message propagates towards the RV point, reverse path routing state is established until a node already in the tree is found, thus forming a multicast tree rooted at the RV point. A sender simply routes data towards the group identifier, so that the RV point may then propagate it over the established tree. An important characteristic of Scribe is that multicast routing state is maintained in a decentralized fashion: each node in a tree is only aware of its immediate ancestors and descendants. This is a significant scalability advantage over other overlay multicast schemes (for example, Bayeux [Zhu2001]) as it means that Scribe does not require excessive signalling traffic in order to gather global state information.

The reliance of Scribe on end hosts may however lead to inefficiencies. An end host that is an interior node in some trees will limit the bandwidth available to all those trees to that of its access link. This can be avoided by exploiting the properties of the underlying DHT to create a set of trees such that each node will be an interior node for only one of them [Cas2003b]. However, this solution is tied to a specific overlay routing scheme (in this case, Pastry). In addition, an end host that is an interior node even for a single tree, may still be a bottleneck: as shown in Figure 35(a), data in transit has to enter and exit the RV point (peer b) and the other two internal end nodes c and d, only one of which (peer c) is also a receiver, via their access links. If these access links are asymmetric, the tree bandwidth will be limited by the, typically lower, uplink bandwidth. Another issue is that neighbouring end hosts may download the same content via separate tree branches, thus incurring unnecessary network load. For example, in Figure 35(a) the two peers e and f receive separately the content from their parent in the tree (peer d). Note that in this example peers b and d act as intermediate tree nodes

without being receivers. This arises when nodes participating in various multicast groups share the same DHT substrate, so as to amortize DHT maintenance costs among different groups and improve DHT routing performance by increasing the available overlay paths.

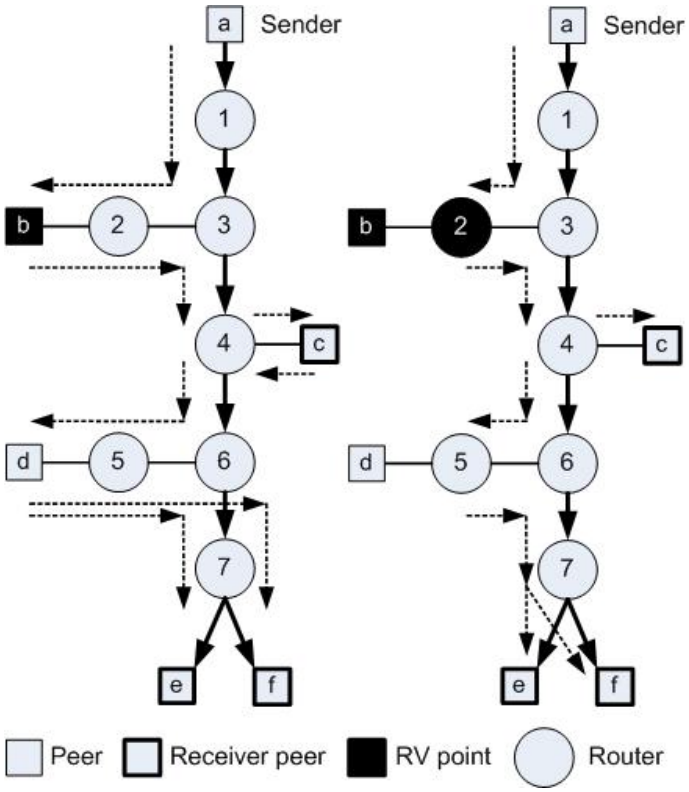

Figure 35: Overlay multicast: (a) non router assisted (b) router assisted.

To investigate the intrinsic characteristics of a multicast supporting, publish-subscribe network architecture and at the same time avoid these aforementioned overlay multicast problems, we consider using the access router of a peer as its proxy in the DHT substrate and overlay multicast scheme. This means that the access router participates in the DHT on behalf of the attached peer. If multiple peers are attached to the same access router, a single place will be held by it in the DHT, that is, the access router will be assigned a single portion of the identifier space, regardless of the number of directly attached peers. An access router will only enter the DHT and act as a proxy if at least one of its attached end hosts is a peer, therefore access routers are not burdened with the signalling overhead of maintaining the DHT unless there is a reason to do so. In the same manner, the access router also acts as a proxy for the peer in the Scribe trees. This means that the access router is responsible for joining the multicast groups indicated by the attached peers and forwarding the incoming traffic to them. The access router may also participate in a multicast tree as an interior node, subject to its position in the identifier space and the operation of regular Scribe, that is, if in regular Scribe its attached hosts were interior nodes of that tree. In this case, it forwards the incoming traffic to its tree descendants, as well as to any interested attached peers.

The proposed proxy role of access routers presents some significant advantages regarding the characteristics of the created distribution trees. First, as shown in Figure 35(b), data do not need to cross the access links of interior tree nodes at all, only crossing the, typically faster, downlink direction of those access links leading to nodes that are members of the group. For example, data will not cross the access link between peer d and router 5 at all, and it will only cross the access link between peer c and router 4 in the downlink direction so as to deliver the content to peer c. Second, multiple tree branches towards end hosts attached to the same access router can be aggregated in a single branch leading to that access router (in our

example router 7). For the entire distribution tree of Figure 35, router assistance means that a packet transmitted to the group will only cross 12 instead of 20 links with regular Scribe (or 8 with an optimal IP multicast tree), avoiding the uplink direction of access links. Therefore, in router assisted overlay multicast the paths through the distribution trees become shorter and faster, while redundant transmissions over the access links of intermediate nodes are prevented, something especially important in an environment where asymmetric access links are prevalent, as it prevents the (typically slower) uplinks from becoming bottlenecks.

In order to thoroughly investigate the potential gains of the proposed router assisted overlay multicast scheme, we have performed an extensive set of realistic simulations. One of the main concerns in simulating large scale content distribution is related to the scalability of the simulation. The original performance evaluations of Scribe [Cas2002], [Cas2003a] were focused on scalability so as to investigate the structure of the produced trees and their properties, such as the distribution of the forwarding load in an Internet like environment. In addition to investigating the performance of our router assisted overlay multicast scheme with respect to the structure of the produced multicast trees, we also strive to create a simulation environment that can be used for detailed application simulations, with applications running on top of the routing substrate, being also scalable enough to support realistic studies over Internet like network topologies.

4.4.3.2 Simulation scenarios

In our simulations we considered a wide range of network topologies ranging from 650 routers to 5050 routers in total. Five different topologies were generated for each network size and all presented results express the average values over those instances. Table 11 shows the GT-ITM parameters used to generate the topologies [Zeg1996]. In all topologies, the default link establishment probabilities were used, that is, a link between two transit routers was established with a probability equal to 0.6 and a link between two stub routers was established with a probability equal to 0.42. The target number of end hosts were then randomly attached to the stub routers. In order to stress the limits of our simulation environment, we first investigated the memory requirements for loading each topology in the simulator, and then tried to strike a balance between the memory requirements of the larger topologies and the limitations they imposed on the number of end hosts participating in the experiments. Most of the results here refer to Topo1 with a variable number of end hosts in the network. Specifically, we simulated three different scenarios with 500 (Sparse), 1000 (Medium) and 4000 (Dense) end hosts, respectively, so as to explore the impact of host density to the performance of the suggested overlay multicast scheme.

Regarding multicast groups and their sizes, a Zipf-like distribution was used for the size of each group, that is, the r-th group had a size equal to floor($Nr^{-1.25}$+0.5), where N was the total number of overlay nodes, as in [Cas2002]. The first group included all overlay nodes, while the last group had a size equal to 11 nodes, a size typical of instant messaging applications [Cas2002]. Based on this lower bound and the number of participating hosts, the number of groups in the Sparse scenario was 21, in the Medium scenario it was 36 and in the Dense scenario it was 110. The members of each group were randomly selected from the entire end host population, meaning that each end host may have participated in many groups. For each group, a random identifier was chosen and a non-member end host was randomly selected as the sender. In all scenarios, end hosts first join the overlay (Pastry) network; in our scheme, this means that proxy routers join the overlay network on behalf of their attached end hosts. After the initialization of the overlay network has completed, participating nodes start joining, and therefore forming, the multicast trees. When all nodes have joined the respective trees, the senders send a data packet towards the RV point of each tree to inspect the path between the sender and the RV point. Note that in our measurements unlike in earlier studies [Cas2002], [Cas2003a], we explicitly take into account not only the tree formed between the RV point and the receivers, but also the path from the sender to the RV point, since application performance is determined by the entire path between sender and receivers.

Table 11: Network topology parameters.

	Topo0	Topo1	Topo2	Topo3
Transit domains	5	7	9	10
Avg. routers per transit domain	5	4	4	5
Stub domains per transit router	5	7	9	10
Avg. routers per stub domain	5	7	9	10
Stub routers	625	1372	2916	5000
Transit routers	25	28	36	50
Total routers	650	1400	2952	5050

4.4.3.3 Scalability

As shown in [Bau2007], OverSim enables the simulation of scenarios with even 100.000 overlay nodes. However, in the simulations presented in that paper, the underlying network was either too simplistic (SimpleUnderlay) or too small (IPv4Underlay with 20 backbone and 20 access routers). If the same number of overlay end hosts is used with both models, then the memory requirements of the IPv4Underlay model will be larger due to additional network elements (routers and links) and their operation. It is also expected that the memory footprint of routers will differ from that of overlay end hosts.

The simulator memory requirements for the considerably larger network sizes described in Table 11 are presented in Figure 36: the x-axis indicates the size of the network in terms of the total number of participating routers, while the curves indicate the memory footprint of scenarios either without any end hosts or with 1000 end hosts in the proposed router assisted (Proxies) or in the regular Scribe scheme (No Proxies). It is clear that the memory footprint of the networking topology increases dramatically with the number of participating routers, in a non-linear fashion. This increase is due to the increasing number of links between the participating routers. The memory requirements of the network topologies have a severe impact in the feasibility of large scale scenarios, since, for example, a topology with 5000 access routers and 50 backbone routers, already requires approximately 1800 MB of memory; for a realistic simulation, we also need to add end hosts, along with their access links.

It is important to point out why our proposed router assisted scheme requires less memory than the regular Scribe scheme: as all overlay functionality is provided by the access routers, we did not create the actual end hosts at all, so as to reduce the memory footprint of the simulation. Furthermore, our scheme requires fewer messages for the establishment of the overlay, since each router joins the overlay only once, regardless of the number of end hosts that it is a proxy for. Note also that these messages travel smaller distances since they do not need to cross the access links; only a single message is required for an end host to initially ask its access router to be its proxy.

Figure 37 shows the processing (CPU) time for the simulation of each scenario, which includes constructing the network topology with 1000 end hosts, running Pastry to establish a DHT substrate, running Scribe to establish multicast trees and sending a single message to the RV point of each multicast tree. Again, the resource requirements of the Proxies scenario are considerably lower than in the No Proxies scenario. This can be attributed to the lower number of running simulation modules as well as to the avoidance of multiple messages for overlay maintenance when many hosts are attached to the same router.

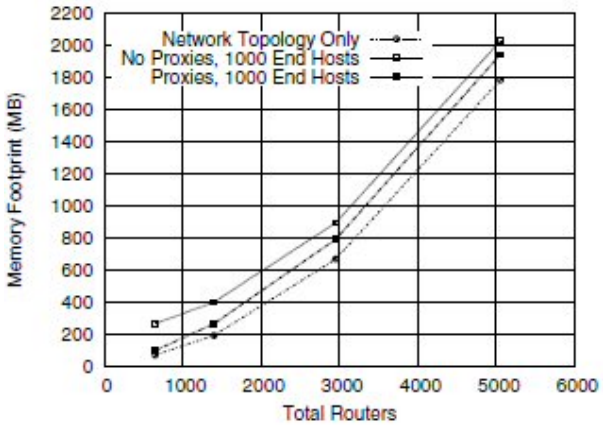

Figure 36: Simulation memory requirements.

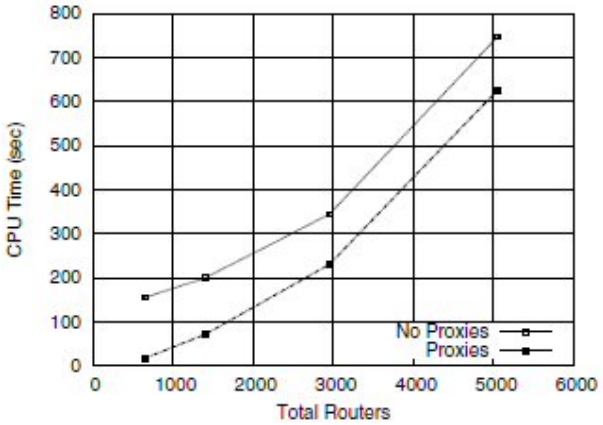

Figure 37: Simulation processing time.

4.4.3.4 Performance

As the proposed scheme alters the structure of the multicast trees produced, we examine below the performance of each scheme in terms of the properties of the resulting trees.

Path stretch

The most obvious metric for evaluating overlay routing schemes is the overhead incurred by not strictly following the shortest path IP routes. This is usually expressed by the term stretch which is defined as the ratio between the overlay path length (or delay) and the length (or delay) achieved by IP routing. In essence, stretch expresses the degree to which the performance achieved is worse than the performance of the underlying IP substrate, a penalty paid in return for the scalability of the overlay solution. Multicast complicates this metric, not only because we need to take into account the entire distribution tree, but also because the Internet does not provide multicast routing in the first place. The usual convention, also followed here, is to assume that IP multicast would use the tree formed by merging the optimal unicast paths between the sender and each receiver. In the absence of multicast, a sender desiring to reach all receivers would have to send duplicate copies of each packet over all those paths.

We define path stretch as the ratio between the number of IP hops comprising the path from the sender to a receiver in the multicast overlay tree to the number of hops that comprise the shortest unicast path between these two nodes, averaged over all trees and paths. The overlay paths are comprised of two segments, from sender to RV point and from RV point to

receiver. For the first segment we use the shortest path between these two nodes: after the initial message sent to locate the RV point for a group, the sender can cache its IP address so as to avoid the overlay path for subsequent data transmissions. For the second segment we use the path constructed by the overlay scheme. Figure 38 shows the path stretch achieved for the Sparse, Medium and Dense scenarios in Topo1; by definition the stretch of IP multicast is 1. It is clear that our approach yields significantly lower stretch values in all scenarios (40-45% decrease), regardless of end host density.

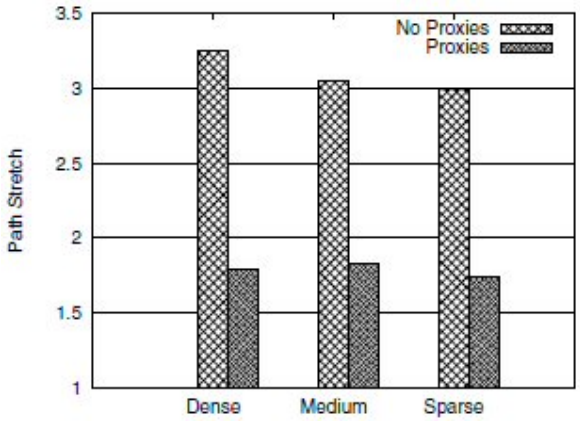

Figure 38: Path stretch.

It should be pointed out that our results diverge from those shown in the papers evaluating Scribe [Cas2002], [Cas2003a] in two ways. First, we measure the entire path between the sender and each receiver, as opposed to the path between the RV point and each receiver, since it is the entire path length that applications have to cope with. Second, we use network hops rather than delays to calculate stretch, since in a static environment we can only calculate the propagation delay. Since this delay may only represent a small part of the actual delays faced by applications in a dynamic environment where queuing delays due to congestion may be dominant, we believe that defining stretch using these delays would be misleading.

Transmission Stretch

Another important performance metric is the efficiency of the multicast trees produced in terms of the total load imposed on the network for data distribution. We define as the total transmission load the number of hop-by-hop transmissions required in order for a single packet originating from the sender of each group to reach all group members of the corresponding group. In order to provide a normalized metric, we define transmission stretch as the ratio between the total transmission load imposed by each overlay multicast scheme to the total transmission load imposed by IP multicast.

Figure 39 shows the transmission stretch achieved for the Sparse, Medium and Dense scenarios in Topo1; by definition the stretch of IP multicast is again 1. Exploiting router assistance results in a decrease of the transmitted packets in all cases (7 to 19% decrease). This is due to the fact that the proposed router assisted multicast overlay scheme leads to the formation of multicast trees with fewer IP hops compared to the regular Scribe trees, leading to a more efficient utilization of the network. Recall also that, as mentioned above, our router assisted scheme eliminates packet transmissions in the uplink direction of access links leading to intermediate nodes, thus removing a bottleneck imposed by asymmetric access links.

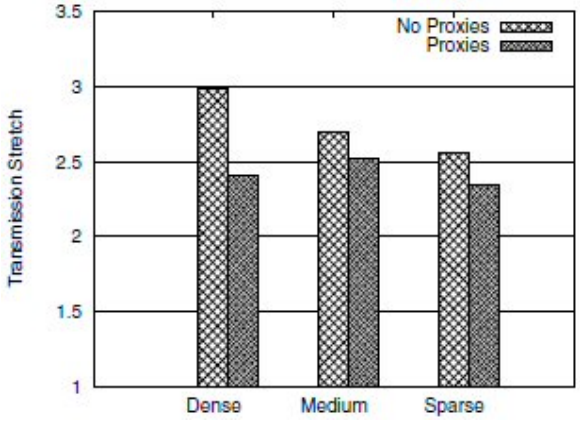

Figure 39: Transmission stretch.

Node Stress

A critical metric of a multicast scheme's performance is the forwarding load it imposes on participating nodes. In a multicast scheme forwarding load can be expressed by the branching factor of each node, that is, the number of descendants it forwards traffic to; in a multi-tree scenario, attention must also be paid to the number of groups served by a node. In Scribe each node maintains forwarding information in a separate children table per group, with each table's entries referring to the node's children for that group. Hence, we calculate two metrics of node stress, as in [Cas2002], [Cas2003a]. The first metric is the number of children tables maintained by each node, which corresponds to the number of multicast groups the node is forwarding traffic for. In our router assisted scheme, this includes the groups for which a router is not forwarding data to other overlay nodes, but is only acting as a proxy on behalf of one or more attached receivers. The second metric is the number of children entries maintained by each node, which corresponds to the number of nodes it is forwarding traffic to, across all multicast groups. Again, in our scheme this includes the group recipients attached to an access router, for which the access router acts as a proxy.

Figure 40 and Figure 41 depict the two node stress metrics for the Sparse, Medium and Dense scenarios in Topo1, aggregated over all nodes in the network; results from other topologies indicate that both metrics are mostly dependent on end host density. The proposed router assisted scheme incurs a significant forwarding state overhead increase, especially considering that this state is concentrated on the access routers serving end hosts participating in the overlay, rather than being distributed among the participating end hosts as in regular Scribe. This increase however is largely an artefact of our definition of the node stress metrics, as both metrics include the proxy entries in the access routers, which do not require the same maintenance overhead as regular Scribe entries.

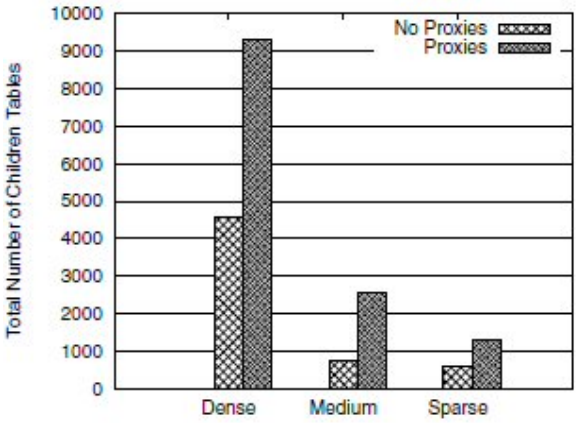

Figure 40: Node stress: children tables.

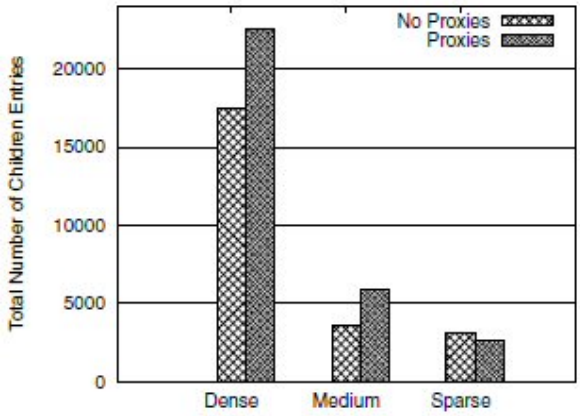

Figure 41: Node stress: children entries.

In order to assess how the amount of forwarding state affects the nodes participating in the overlay multicast routing process, Figure 42 and Figure 43 show the number of overlay nodes per tree, as a function of the number of end hosts participating in the DHT substrate, either directly or via their access routers; for clarity, the figures show only the ten most popular groups in each scenario. As expected, with more end hosts (Dense) and fewer access routers (Topo1), Figure 42 shows that the forwarding load in the router assisted approach is concentrated on fewer nodes than in regular Scribe, while with fewer end hosts (Sparse) and more access routers (Topo3), Figure 43 shows that the forwarding load of the router assisted approach is distributed in roughly the same manner as in regular Scribe.

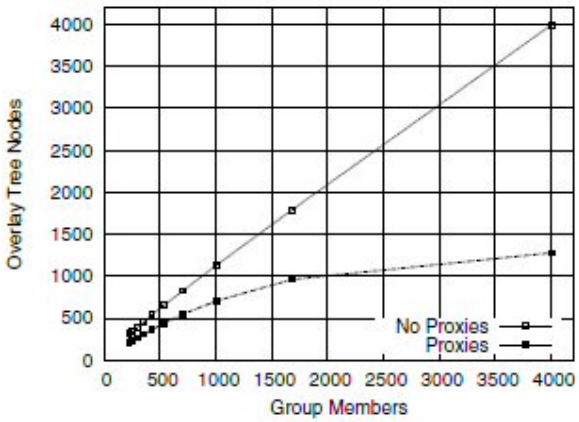

Figure 42: Number of overlay nodes (Topo1, Dense).

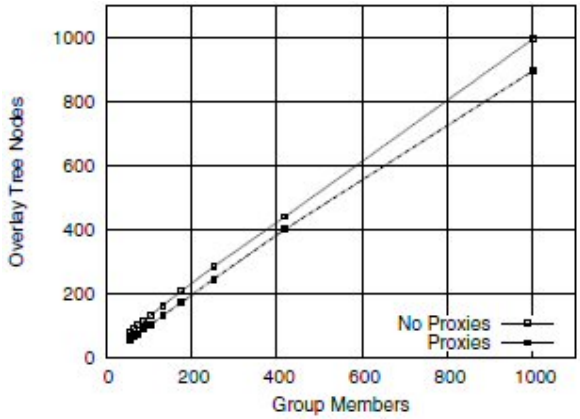

Figure 43: Number of overlay nodes (Topo1, Sparse).

On the other hand, this reduction in the number of overlay participants also reduces the number of overlay signalling messages exchanged to establish the DHT substrate (Pastry). As demonstrated in Figure 44 for the Sparse, Medium and Dense scenarios in Topo1, the router assisted approach greatly reduces (by 45%) the signalling overhead for DHT maintenance in the Dense scenario, as in this case the access routers participate in the overlay on behalf of many end hosts. In the Sparse scenario each router hardly ever serves more than one end host however, thus we see only a slight reduction (4%); in the Medium scenario the reduction lies between these extremes (25%). Note that in our router assisted scheme these messages travel shorter distances since they do not need to cross the access links to and from the end hosts. Therefore, there exists a tradeoff between placing the burden of forwarding and tree maintenance on the access routers, and reducing path stretch, link stress and DHT related signalling.

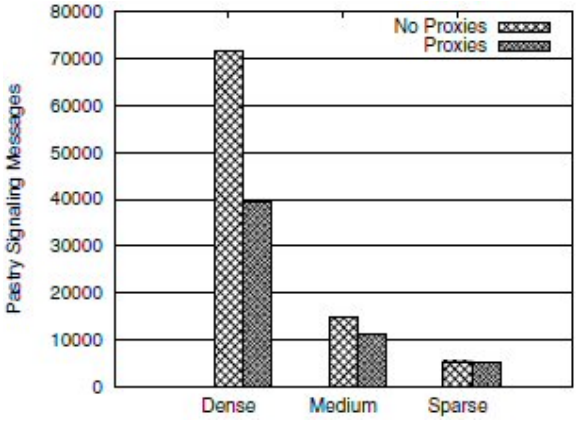

Figure 44: Pastry signaling overhead.

4.4.4 Overlay Multicast Assisted Mobility (OMAM)

In D2.3 we considered the case of mobility support provided by our overlay multicast bsed PSIRP variant. This support is provided by the multicast nature of the routing substrate: the creation of multicast delivery trees enables the localization of routing state updates with respect to node movement. At the same time, the Route Convergence property of Scribe is expected to further shorten the distances travelled by routing update messages as end hosts move.

In the following we evaluate the performance of the proposed architecture in the context of streaming services such as Mobile TV. Our target is to investigate the impact of node movement on the experienced service quality. We compare the performance of our mechanism to that of Mobile IPv6, a standardized solution that can also serve as a basis for indirect comparisons with other mobility schemes. In this comparison the Route Optimization feature of MIPv6 is enabled through the standardized Return Routability procedure [Per2004]. We did not consider multicast assisted mobility schemes based on IP multicast due the limited deployment of IP multicast itself. The following performance metrics are investigated:

Resume time: The elapsed time between the association of the Mobile Node (MN) with a new Access Point (AP) and the reception of the first packet by the MN in its new position. It indicates the efficiency of the signalling mechanism used to divert traffic towards the new position of a MN.

Packet loss: The percentage of transmitted packets not delivered to the MN due to mobility. It is a coarse grained indication of the service disruption incurred.

End-to-end packet delay: The elapsed time between the transmission of a packet by its source and the reception of the packet by the MN. It indicates the impact of stretch on the delivered service.

4.4.4.1 Signalling analysis

In the following we provide an analytical comparison of the proposed architecture and Mobile IPv6 in terms of signalling overhead. We chose Resume Time as the primary performance metric as it directly reflects the efficiency of a mobility support scheme in terms of communication disruption. Obviously, this time is heavily affected by the signalling required in order for the network entities involved to be informed about the MN's change of position, that is, the time required for the routing substrate to adapt to the movement of the MN. Figure 3 shows a generic network topology in which a mobile node (MN), initially residing at its home network, moves around the network while communicating with a Corresponding Node (CN). Table 12 outlines the signalling messages of the Return Routability procedure which is required for Route Optimization to work.

Table 12: Return Routability procedure.

#	Message Type	Source	Destination
1	Binding Update (BU)	MN	HA
2	Binding Ack. (BA)	HA	MN
3	Home Test init (HoTi)	MN	HA → CN
4	Care-of-Test init (CoTi)	MN	CN
5	Home Test (HT)	CN	HA → MN
6	Care-of Test (CT)	CN	MN
7	Binding Update (BU)	CN	MN
8	Binding Ack. (BA)	MN	CN

In the following we use $d_{x \to y}$ to denote the distance in number of hops between the network entities x and y. Since steps 3 and 4 and steps 5 and 6 of the Return Routability procedure take place in parallel, we can express the Resume Time of Mobile IPv6 as follows:

$RT_{MIPv6} = 4d_{MN \to HA} + 2d_{MN \to CN} + 2d_{HA \to CN}$

In our overlay multicast assisted mobility (OMAM) scheme, a re-subscribe message is sent over the wireless medium by the MN towards the newly visited Overlay Access Router (OAR) which in turn generates a Scribe JOIN message towards the lowest Common Ancestor (CA) in the multicast tree. Hence:

$RT_{OMAM} = d_{MN \to OARk} + d_{OARk \to CA}$

The following equation expresses Pastry's route convergence property.

$d_{OARk \to CA} = a\, d_{OARk-1 \to OARk}$, $a \to 1$

By neglecting the distance cost of wireless medium transmissions in both scenarios and assuming, for simplicity, that OAR_{k-1} belongs to the path between the HA and OAR_k, our scheme results in a smaller RT value when:

$RT_{OMAM} < RT_{MIPv6} \Leftrightarrow a < 4 + 2(2d_{OARk-1 \to HA} + d_{OARk \to CN} + d_{HA \to CN})/(d_{OARk-1 \to OARk})$

According to the route convergence property $a \to 1$, thus it is evident that our architecture results in a reduced RT. Moreover, $RT \to 0$ when OAR_k is already a member of the publication's multicast tree. This may happen due to another MN attached to OAR_k that has expressed interest for the same publication, or because OAR_k is acting as a forwarding node, that is, an intermediate tree node, for another OAR_j.

4.4.4.2 Evaluation

To evaluate the performance of our scheme under dynamic conditions, we used the OMNeT++ simulation framework [Var2009] enhanced with the xMIPv6 implementation of Mobile IPv6 [You2008] and OverSim [Bau2007]. In our simulations we considered a simple network topology comprising multiple OARs deployed in a grid-like topology. All OARs run the full TCP/IP protocol stack, as well as Pastry and Scribe in the case of the proposed architecture. All wired connections are implemented with Ethernet links. One IEEE802.11b AP is directly connected to each OAR, with neighbouring APs operating in disjoint channels. We were restricted by the xMIPv6 model structure which necessitates the deployment of a single home network with a single MN in any topology.

Both the HA and the CN are connected to randomly chosen, disjoint OARs of the topology. The AP in the Home Network of a MN is directly connected to the HA rather than to the respective OAR. In the case of the proposed architecture and in order to produce comparable scenarios, we placed an extra OAR between the AP and the randomly chosen OAR acting as the initial point of attachment of the MN to the network. Furthermore, an extra OAR was also connected to a randomly chosen OAR in the grid topology to serve as the originator of the data flow. Figure 45 shows an example topology. The circles around each networking entity

denote the transmission range of its wireless interface. The square bounded area denotes the part of the topology actually accessible by the MN. We have chosen a topology providing full wireless coverage in order to restrict the anticipated service disruption to each protocols' operation.

Our scenarios consider a single MN initially connected to its home network. Upon initialization, the MN starts to move following the mobility model described in [Per1999]. In this model a MN moves in a straight line, makes a turn and starts over. Its speed and direction are updated every x seconds, with x following a normal distribution with a mean of 10 seconds and a standard deviation of 0.1 seconds. The speed of the MN is normally distributed with a mean of 1.39 meters/sec (approximate walking speed of 5 Km/h) and a standard deviation of 0.01 meters/second. The change in direction also follows a normal distribution with an average of 0 degrees (no turn) and a standard deviation of 5 degrees. Whenever the MN reaches the limits of the simulated area it bounces with the same angle and speed.

Our measurements were based on a simple application scenario in which a stationary node sends a sequence of UDP datagrams (CBR traffic) towards the MN. The UDP stream resembles a H.264, Level 1 SQCIF video stream with 30.9 frames per second [ITU2007]. In the case of MIPv6, the CN is initially only aware of the MN's home address and uses it as the destination of its data. While moving, the MN is responsible for updating its bindings with its HA and the CN in the case of Route Optimization. In the case of the proposed architecture, the CN simply sends its UDP packets towards a rendezvous point whose position in the networks is determined by a randomly generated key and Pastry's functionality. The complete set of parameter values used in our simulation environment is provided in Table 13.

Table 13: Simulation scenario parameters.

Parameter	Value
Grid size	30 x 30
Number of MNs	1
Number of CNs	1
Wired connections type	100 Mps Ethernet
Propagation delay (ms)	0.5
Data rate (Kbps)	64
Packet size (bytes)	26
Total number of packets sent	556200

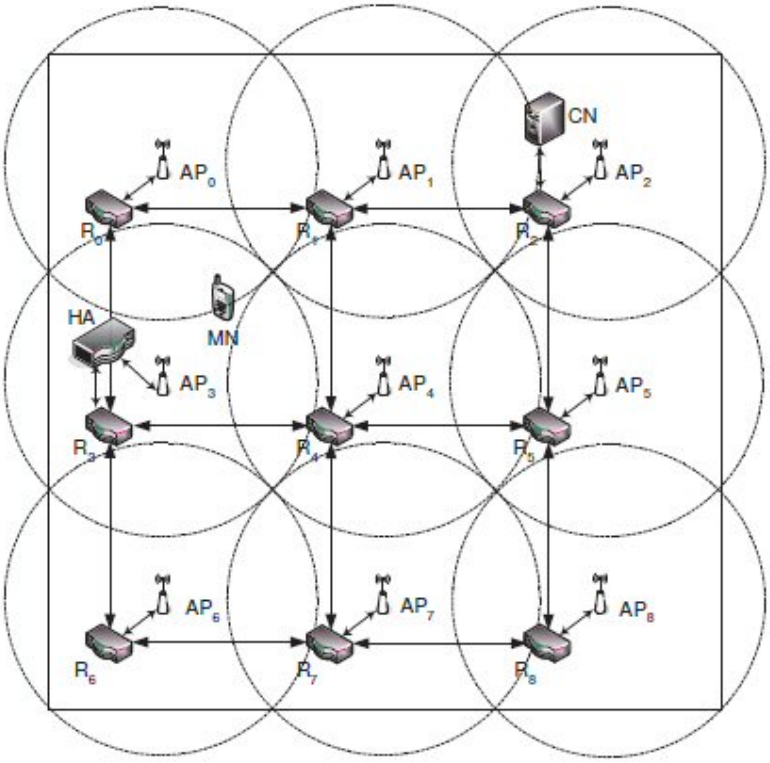

Figure 45: Grid example topology.

We now present our initial results derived from the simulation scenarios described above. These results constitute the first step towards a thorough evaluation of the proposed architecture with respect to mobility. Table 14 shows the values derived for the above described metrics.

Table 14: Simulation results.

	Mobile IPv6	Overlay Multicast
Resume time	1.208 sec	0.007 sec
Packet loss	2.002%	1.059%
End-to-end delay	12ms	17ms

As expected, our scheme yields a significantly lower Resume Time than Mobile IPv6. This is justified by the localized handling of mobility. Upon each change of network position Mobile IPv6 launches the Return Routability procedure in order to apply Route Optimization. As shown in the previous section, this causes an exchange of signalling packets both with the HA and the CN. Considering the fact that these nodes may be located in distant parts of the network, this signalling may considerably delay the establishment of new routes for the traffic destined to the MN. On the other hand, in our architecture this signalling overhead is reduced since only the CA node in the multicast tree is notified about the MN's change of position. The significance of the reduced Resume time is depicted in the Packet loss metric where we see a noticeable difference between the two considered approaches. Furthermore, it must be noted that by localizing routing updates our architecture enables the actual delivery of packets in transit in the scenarios where the CN resides in a distant area of the network. However, these improvements come at the cost of end-to-end delay. Indeed, as we see in Table 14, our approach results in a higher end-to-end delay. As explained earlier, this is due to stretch imposed on the routing due to the reliance on a DHT substrate. We must note however, that this increase could be acceptable for non-interactive streaming applications as the ones considered here.

4.5 Network emulation: integration with the prototype

Network emulation is a work area where the Evaluation and Implementation work packages meet, as it means connecting a packet-level simulator to the physical prototype implementation. Network emulation is beneficial because of two reasons: 1) it is useful for detailed protocol testing, i.e. a large number of use cases can be "played" to the prototype and 2) it allows testing of the protocol code in medium-scale environments via code re-use. In our earlier project deliverable D3.2 we outlined two possible future plans of network emulation: 1) implement an interface between the simulator and the prototype and build a simplified protocol in the simulator for large-scale testing of the protocol and 2) porting as much code of the real implementation as possible into the simulator. The second goal is obviously out of scope for the NetFPGA-based hardware implementation. But with the NetFPGA implementation, where only a few physical machines are available for the partners, the network emulator can be even more important to make the machine believe that it is part of a bigger network.

For implementing the network emulation pieces, we decided to use ns-3, as modules that allow network emulation have been designed to the simulator architecture from the very beginning (and not as an afterthought in ns-2), and additionally, we also chose this platform because of its promising and clear design for the quantitative analysis of the zFilter-based forwarding architecture.

At the date of writing this deliverable (when ns-3.4 is out), ns-3 already supports two emulation modes in the stable release. Both of these modes are realized by NetDevices (the simulation implementation of network interfaces). The first mode of operation is implemented by the Emulated NetDevice. In a nutshell, this means that the network device in the simulator is connected to a real interface, but looking at it from the "top" in the simulator it looks like any other simulated network device, therefore it can be part of the simulated nodes. The other mode is the opposite: real host softwares can be attached to the simulation and communicate with simulated nodes via ns-3's simulated channels. In the project our decision was to explore the usability of the first mode, as it can work both with the BSD and the NetFPGA implementations.

So far, we have created a partial/preliminary implementation that recognizes the packets belonging to the pub/sub protocol, and is able to detect the packet header that contains the forwarding information. Currently the forwarding decision it makes is simplified: it copies the packet to all of the outgoing interfaces (broadcast). Note, however, that the goal of the current implementation exercise was only to perform a quick feasibility study and to evaluate whether ns-3 will be appropriate for our purposes. Current experiences are promising, as a first interoperability test was carried out, where two BSD nodes were connected through a simple ns-3 network.

In order to have some first scalability analysis, our plans were to estimate the latency of our implementation by different sized topologies. To achieve this, we implemented a simple measurement tool on FreeBSD, using a PC with two separate Gigabit Ethernet NICs. The tool sends a timestamped packet through one NIC and waits for the packet to be returned through the other one. We measured the average packet delay over 10000 packets over differently sized topologies. The topology model was the one shown on Figure 46. Its parameters are the number of nodes in the "chain" between the two real interfaces (n), the number of side branches (s), and the number of nodes per side branche (b). The delay on the simulated links was set to zero, as at this time we were interested only in the processing delay caused by the simulator. The simulator was running on a PC with quad-core Intel Xeon E5420 processor at 2.50 GHz with 3.9 GiB of memory.

The results are shown on Figure 47. (with parameters n=6 b=6. s=0,1,2,4,8,16,32 and 64). The results tell us that with huge trees there is a linear relation between the real time delay and the number of side nodes, however, the delay is close to constant when there is a maximum of ~30 nodes in the tree. We can also conclude that the measurements did not reveal any exponential explosion in our region of interest. We also tested the scenario when

there is only one simulated node connected with to emulated netdevices to the sending machine. For that, we measured 244.7 microseconds, which is a reference lower bound for the performance of any further protocol implementation and functionality that we will integrate to our emulator based on ns-3.

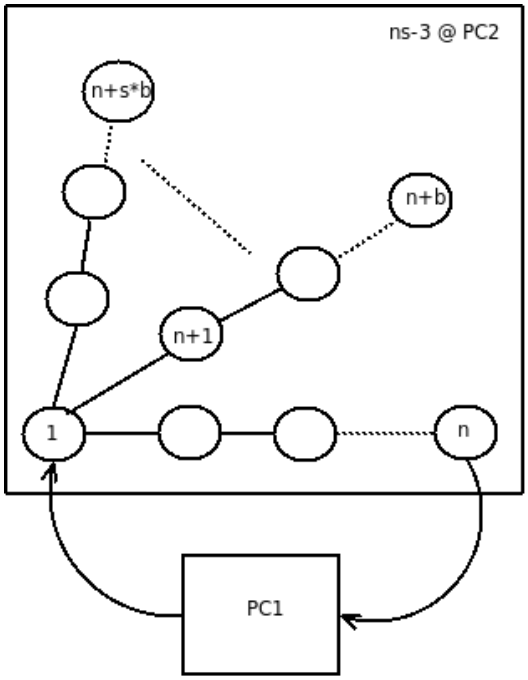

Figure 46: Measurement setup and the simulated topology model.

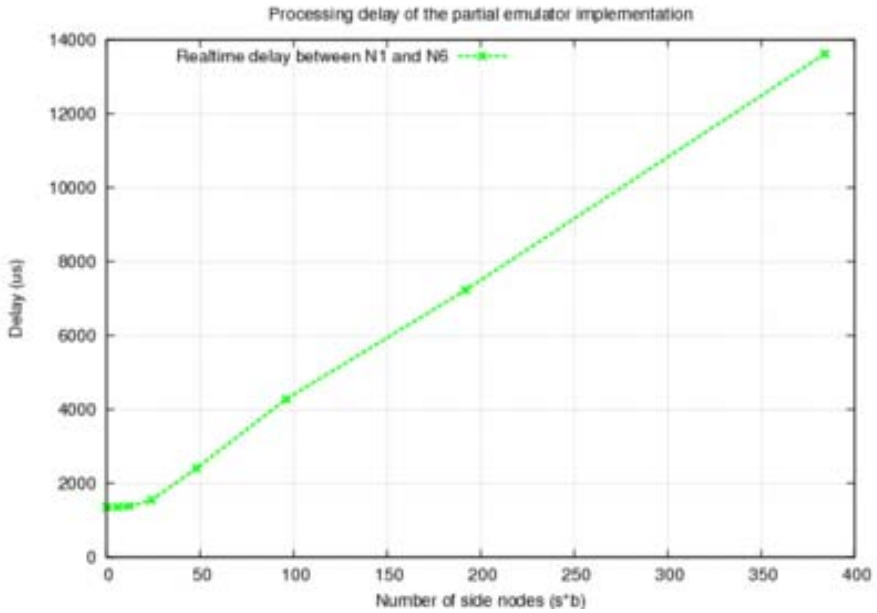

Figure 47: Realtime delay through ns-3.

4.6 Conclusions and Open Issues

While substantial progress has been made in evaluation work as described in the previous sections, much of course remains to be done. The present results have been fed back into the architecture and implementation work which have simultaneously continued their work towards the next design cycle. New component designs are emerging, focussing on issues such as inter-domain topology formation, algorithmic IDs, blackboard-centric node internal architecture, helper functions, mobility, and network attachment. Each of these will be subject to evaluation and validation in the future, with results being reported in the upcoming deliverables D4.5 and D4.6.

Evaluation of the prototype implementation will be revisited as the blackboard-based implementation revision is completed. Once this is done we expect the evaluation of the implementation in network testbeds to at least partially converge with the packet level simulation activities. As shown by results given above, ns-3 can support emulated networks that are seamlessly connected to prototype implementations to provide additional scalability for the detailed evaluation activities. Also the focus of the work will shift from performance evaluation of individual components towards a more system-level view. This will require further work on various simulation models and tools for performance evaluation, such as different type of traffic generators. Inter-domain evaluation work will also be continued, in part extending the current rendezvous evaluation work (discussed in the following section) and partially to include new architectural components, such as the inter-domain topology formation. Finally, as a short-term goal, our experiences with various validation and simulation tools will be documented in deliverable D4.4.

5 Evaluating the Rendezvous Architecture

The main purpose of the rendezvous functionality in the PSIRP architecture is to provide publish/subscribe signalling reachability from publishers and subscribers to data or service specific rendezvous points.

The main requirements for this functionality are:

1. **Efficiency of operation**: measured in signalling overhead and overall latency,
2. **Feasibility**: scalability to Internet-like networks and data space sizes,
3. **Deployability** and maintainability, and
4. **Security**, fairness, and recoverability.

The first two requirements are usually considered for inter-networking designs, but in many cases the third requirement, deployability, is often not explicitly addressed. In this section we will evaluate the design along these dimensions.

Other important goals for the rendezvous architecture are fault tolerance, flexibility via modularity and orthogonality, openness, and extensibility of the architecture. However, we do not cover these in detail in our system evaluation.

The rendezvous architecture is briefly explained in the following subsection but a more comprehensive description can be found in the PSIRP deliverable D2.3 [Psi2009].

5.1 Introduction to the Architecture

The role of the rendezvous function in the PSIRP architecture is to match the interests of publishers and subscribers. Rendezvous implements the function of determining the set of subscribers and publishers for a given publication and instructs the underlying network machinery to perform the needed data delivery.

Figure 48 presents an overview of rendezvous within local, intra-domain, and inter-domain network environments. The rendezvous entities involved in each of these settings are necessarily determined via scopes. The inter-domain rendezvous is further subdivided to rendezvous within and between rendezvous networks.

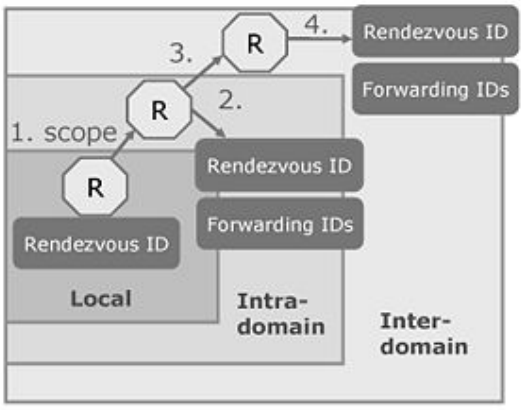

Figure 48: Different levels of rendezvous.

Rendezvous state is created to allow the subscriptions and publications to meet within a rendezvous node hosting the specified scope. Subscriptions and pre-publication advertisements are utilized by the rendezvous system to create partial forwarding state from

publishers towards subscribers. When a publication becomes active, i.e., when there is both an actively sending publisher and one or more active subscribers, the rendezvous system instructs the topology formation function to complete the forwarding path by mapping the rendezvous identifier to intra-domain and inter-domain forwarding identifiers.

5.1.1 Rendezvous Networks

Rendezvous networks are the unit of deployment in the PSIRP rendezvous system. The rendezvous networks are formed by rendezvous nodes (RNs) organizing into a BGP-like inter-domain hierarchy (Figure 49). Rendezvous networks reflect organizations and routing policy boundaries in a similar way to ASes in the current Internet.

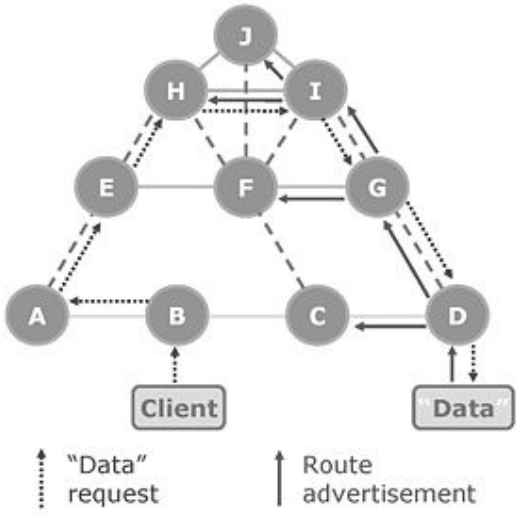

Figure 49: Inter-domain rendezvous within a rendezvous network.

Rendezvous points (RPs) are logical meeting places of the pub/sub system. There is one (potentially distributed) RP for each <scope identifier, rendezvous identifier> pair. RPs are hosted by (one or more) physical rendezvous nodes.

5.1.2 Rendezvous Network Interconnection

Given the description of rendezvous networks above, the next challenge is the interconnection of such networks to form a globally reachable rendezvous solution.

The participating rendezvous networks in such virtual interconnection overlays need to share a degree of mutual trust. The level of this trust needs to be the highest amongst the networks interconnecting directly, and may be lower between entities further away from each other. This gives a rise to a virtual hierarchy-like structure, where rendezvous networks join into virtual subdomains, which in turn are joined in higher level subdomains. In the end all participating rendezvous networks are part of the same, overall virtual interconnection structure.

One possible structure for providing the above is the Canonical Chord hierarchical DHT [Gan2004]. The overlay structure is virtual, and all the interaction happens between the root-level rendezvous nodes of the rendezvous networks.

It should be noted that this interconnection structure is only used for the rendezvous signalling; the actual data being transmitted can still follow optimal policy-compliant forwarding paths, even when the signalling paths may be longer to gain the needed Internet-wide scalability.

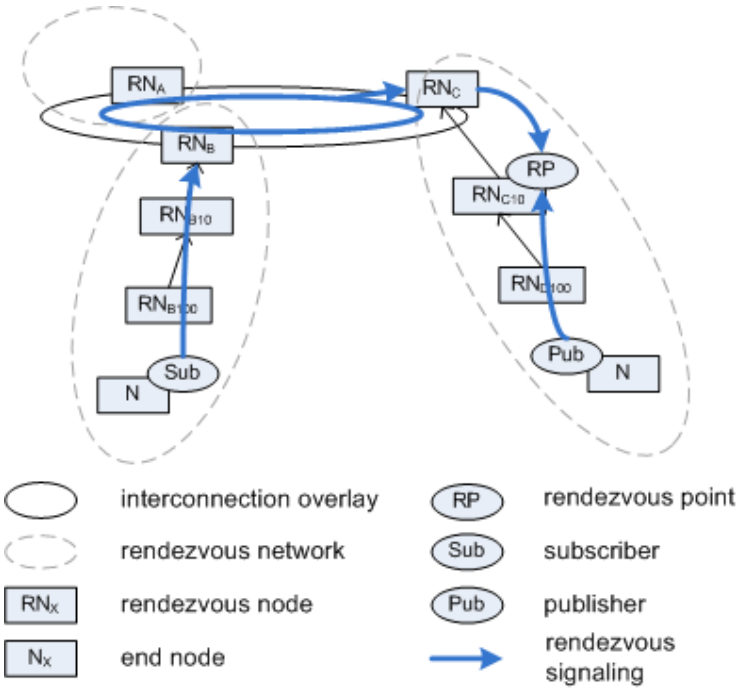

Figure 50: Reference rendezvous topology.

Figure 50 represents an example inter-domain rendezvous system containing three rendezvous networks; A, B and C (dashed ellipses), which are interconnected through an interconnection overlay (the solid ellipse). In the system we have two types of physical entities, namely *rendezvous nodes* (RN) and *end-nodes* (N), and three types of logical entities: *rendezvous point* (RP), *subscriber* (Sub) and *publisher* (Pub).

5.2 Evaluation Goals and Methods

Our main goal is to get a realistic view on how the architecture fits in the current Internet inter-domain topology and on the scalability of the system, and estimate what kind of experience it can provide to end-users at what cost to the stakeholders involved.

In particular, we show that assuming the present-day autonomous system structure and traffic patterns, scopes can be resolved to their respective rendezvous nodes with acceptable latencies and loads. We are also interested in the network stretch of subscription operations that form the bulk of the operation of the rendezvous system, in order to get an understanding of how efficiently the underlying network resources are used.

We shall now explain in detail the assumptions behind our simulation-based evaluation. We simulate the system operation at the level of the inter-domain structure of the Internet. Packet-level simulation on the host level would obviously not be feasible on this scale, and it is also not necessary for the level of detail and realism we are seeking. In terms of evaluation metrics, our focus is on the delay and, to a lesser extent, on the load induced to the system. From the point of view of the user of the rendezvous system the most important variable is the latency of the rendezvous operation, because it must be typically performed before the actual payload communication can take place. The rendezvous system is part of the *control plane* of the network and its bandwidth consumption is not as important as it is assumed that the payload traffic forms the bulk of the total traffic. The main advantages of using a high-level, Internet-scale simulation is that the model has realistic amount of content and nodes, end-user observed latencies can be measured directly, and our assumptions about the incentives of the operators can be tested, as well as how they affect the rendezvous network interconnect formation.

Our evaluation method with respect to deployability is mainly analytical, as the question is highly complex and open.

5.2.1 Network Model

We use CAIDA's AS relationships dataset [Cai2008] as our Internet inter-domain topology model (see Figure 51 for an illustration). The dataset consists of a full AS graph derived from a set of RouteViews BGP table snapshots. While it gives a good picture of the tier-1 connectivity and provider-customer links between ASes, it is known to lack many of the peer links [Oli2008]. In the case of large content provider networks, it is reported that as much as 90% of peer links can be missing, because they are invisible to the set of available route monitors. Naturally, this inaccuracy of the model affects our results, but nevertheless gives us a conservative estimate of the scalability of the system as some existing links have not been utilized.

The total number of ASes in the dataset used is 25881 and number of (bidirectional) links is 52407. The links are annotated as being either peer-peer, provider-customer, or sibling-sibling ones.

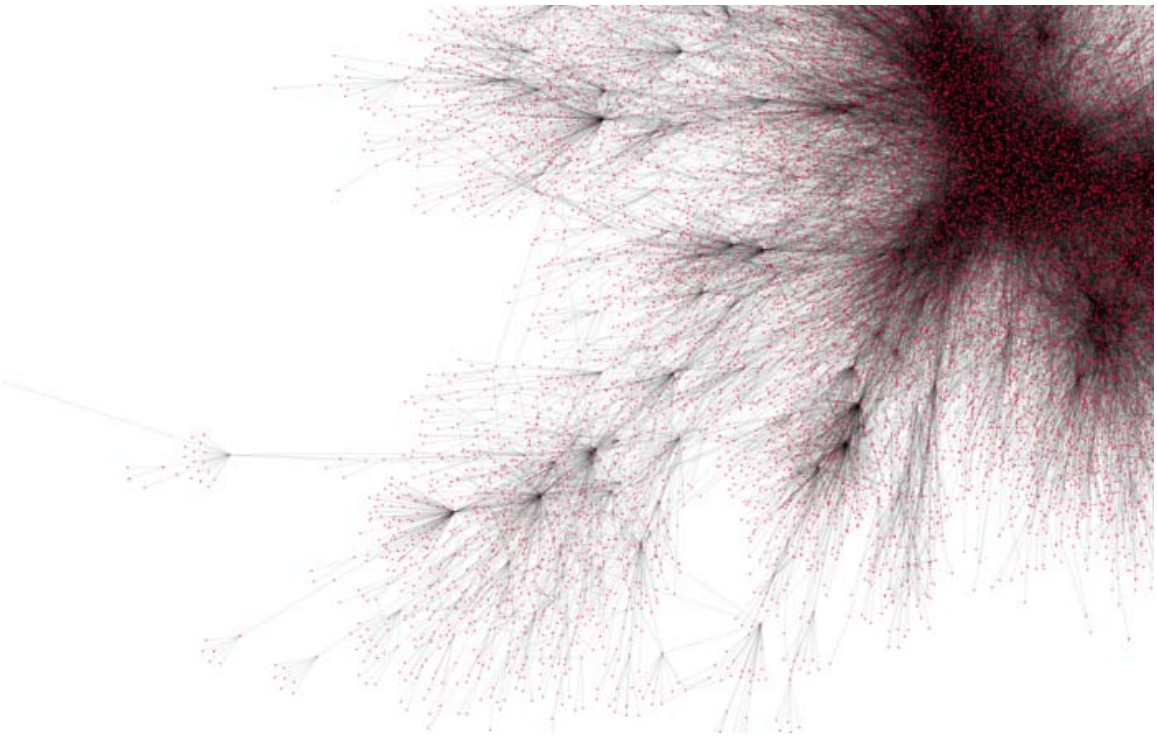

Figure 51: Part of the AS graph visualized.

5.2.2 Traffic Model

We form the traffic model for the system by categorizing ASes into different types, each playing a different role in the system both in terms of participation in rendezvous network operation and in generating traffic. The categorization used is given in [Cha2005]: each AS is characterized in terms of the traffic volumes of three types of network usage, called utilities. The three types used are web hosting (Uweb), residential access (Ura), and business access (Uba). Business access models the cumulative transit provided by the AS for all of its customers and their customers etc. Each of these utilities follows a power-law or a Zipfian distribution of the type Uweb ~ $Rweb^{c}$, where the power-law exponent *c* is approximately -0.9 for North America and Europe and -1.1 for Asia-Pacific (the amount of web traffic was inferred from the web content in search results), and Rweb denotes the rank of the AS in terms of its

web hosting utility. Similar results hold for the other utilities. The authors also present estimates for the rank correlations of the utilities. Based on these results, we annotate the AS graph by generating random variates from the Zipf distributions for different utilities, and assigning these to the ASes so that the observed rank correlations are obtained. Details of this process are given in [Cha2005]. We assume that the target scopes of queries are distributed to ASes proportional to Uweb + α Ura. In the simulation we used the value 0.5 for parameter α, that is, web hosting generates more scopes than home users do. The queries themselves are originated using the Ura distribution only. The popularity of scopes is assumed to follow again a Zipfian distribution in line with several studies in content delivery networks and other such services ([Cha2007], [Gil2007]).

To obtain estimates for the link latencies, we use the following delay values: 34ms for inter-AS node-node hops and 2ms for intra-domain router hops [Zha2006]. The number of intra-domain router hops used between Crescendo nodes residing in the same AS is 1 + floor(log D) where D is the degree of the AS. This is based on findings in [Tan2001], where a strong correlation between the number of routers in an AS and the degree of the AS is found (May 2001 coefficient of correlation was 0.959) and the assumption that the routing topology is efficiently designed.

5.2.3 Modelling Rendezvous Networks

We assume that individual rendezvous networks are implemented as DONA [Kop2007] systems, where the topmost ASes of the rendezvous network store the information of all SIds stored in the rendezvous network. Formation of the rendezvous networks in the simulation is based on a simple model.

As a starting point, each AS is considered as a potential rendezvous network. We simulate the evolution of the rendezvous networks by each AS finding potential rendezvous customers and providers. For incentive compatibility reasons the stub ASes prefer to join rendezvous networks provided by their transit providers. For scalability reasons the transit providers may decline to serve other large transit providers. This choice is affected by two conditions: First, customers having at most 4 rendezvous customers in their network are accepted. Second, larger transit networks with relatively small amount of content are accepted, the limit being 1% of the potential total load on the formed rendezvous network.

This represents the upper scaling limit for the amount of rendezvous state a participant is willing to manage without direct compensation. If there are multiple possible roots for a joined rendezvous network fulfilling the aforementioned requirements, then the largest possible rendezvous network is created.

After forming the individual rendezvous networks, they are joined together in a hierarchical DHT fashion similar to Crescendo/Canon ([Gan2004], [Sto2003]). The essential properties we are interested in are the locality of intra-subhierarchy paths and the convergence of inter-domain paths. Good potential locality properties can also be found in other DHT-based technologies like for example in SkipNet [Har2003], Cyclone [Art2005], Kademlia [May2002], Tapestry [Zha2001] and its new implementation Chimera, and Pastry [Row2001] and the Bamboo system built on top of it. Canon was chosen here mainly because of its clear mapping to the hierarchical structure of the AS topology.

Good locality in the formation of the Canon hierarchy is achieved via observing the underlying AS interconnection topology, while avoiding stand-alone stub ASes for policy-compliancy reasons. In the simulation we laid down the virtual overlay hierarchy to closely follow the underlying AS topology formed by customer-provider links between ASes by first removing cycles stemming from multihoming relationships and then favouring smaller transit providers for a more localized and balanced hierarchy. The minimum number of Chord rings to be merged into a new sub-hierarchy in the Canon structure is five. In the real world, multihoming could be utilized by, for example, splitting the underlying hierarchy into multiple virtual trees to balance the traffic. It follows that the fan-out of the hierarchy can greatly vary at different levels

of the tree. The top layers especially are very flat as tier-1 ASes can provide access to hundreds of customer networks. This is in sharp contrast to the Canon paper evaluation, where a fixed fan-out of 10 in internal nodes of the tree was used. All tier-1 domains are joined into a single tree with a single virtual domain added on top of them.

Each Chord ring stores a record for each local scope, at each level of the hierarchy, consisting of the scope id, home location pointer, and potentially additional forwarding information. The scope id is used as the key in the hash table. When the Canon hierarchy is formed by joining rings at the same level, the scope pointers are copied to new nodes in the combined ring when there is a new node that has its id closer counter-clockwise to the scope id than the node storing it in the local ring. Basically, this means that there is often a copy of each scope pointer at each level of the hierarchy. Therefore the total storage space requirements are n log n where n is the number of scopes and assuming a balanced Canon hierarchy. This way it is possible to guarantee that the whole rendezvous operation is local when the home location of the SId and the subscriber are in the same branch of the hierarchy.

Each rendezvous network reserves a number of (virtual) nodes proportional to the number of hosted SIds in that network for virtual overlay use. For simplicity of analysis, we assume that all nodes forming the virtual overlay have similar amounts of storage and processing capacity. The virtual overlay is only used to store and locate scope pointers that contain information about the current location of the scope. Each rendezvous network is allowed to store a number of scope pointers proportional to the number of nodes they provide to the global network. In this way the costs incurred to each rendezvous network are on a par with the costs they cause to the system.

In addition to persistent scope pointer storage, each node contains βk amount of storage for caching the most recent scope pointers queried via them by their customers. Here k is the amount of storage used for storing scopes at the node and the parameter β describes the relative amount of memory used for publishing and subscribing. An analytical model of the cache performance in steady state was used to estimate the relevant hit probabilities. Assuming that each node caches the n most popular scopes, then, on average a scope with a popularity rank pr is found cached at a node x on level a when

$$pr < \left(\frac{\beta \cdot s \cdot (A/N)}{(A_{x+1,a} - A_{x,a}) mod A}\right)$$

where $A_{i,j}$ is the Chord node identifier of the i-th node at level j and N is the total number of nodes, s is the total number of scopes and A is the size of the whole address space. This assumes that identifiers are evenly distributed in the entire identifier space.

5.2.4 Simulation Results

We find that our simulation generated ~1100 rendezvous networks with a total of ~1200 Canon nodes when the total number of scopes was 1010. The maximum depth of the generated hierarchy turned out to be 7, and the average ~4. Each Canon node had ~4GB memory for storing the scope pointers. As we rely on the power-law distribution of the scope popularity, we expect caching of the scope pointers to be effective. The following figures show the cumulative distribution functions (CDF) of the measured values.

In Figure 52 we examine the AS-level hop count stretch of the path taken by rendezvous messages to a scope's rendezvous node (relative to the shortest valley-free policy-compliant route) with different scope pointer cache sizes. The cache memory is proportional to the memory reserved for scope pointers (β). The Zipf exponent is -1.0, which represents the traditional Zipf distribution. Four cache sizes are used: a) no cache at all, b) 1% (~40MB), c) 10% (~400MB), and d) 100% (~4GB).

We find that for the typical Zipf distribution a 1% cache size is very effective, and that additional cache memory usage might not be justified.

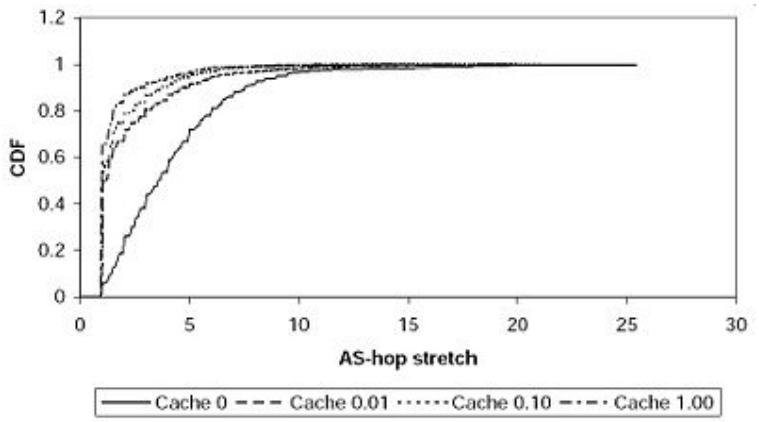

Figure 52: AS-level stretch with cache sizes (Zipf -1.0).

In Figure 53 we examine the sensitivity for different degrees of Zipfian distributions with the cache size of 1%. The range of exponent values is adopted from ([Cha2007], [Gil2007]). The smaller absolute values are very conservative.

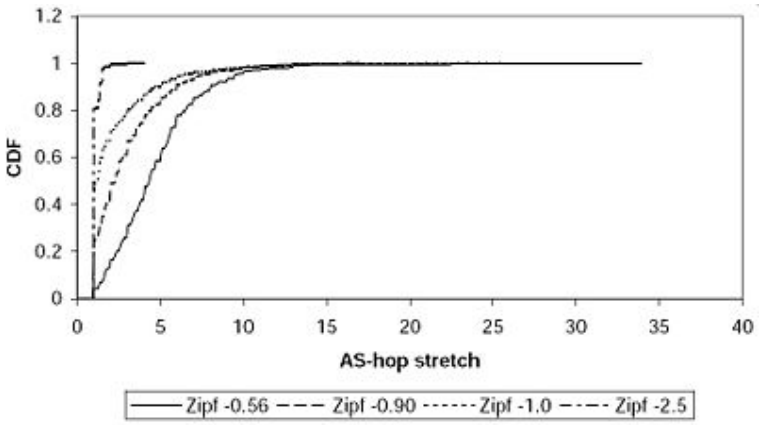

Figure 53: AS-level stretch with Zipf exponent (1% cache).

In Figure 54 the distribution of subscription latencies in seconds is given (with the same parameters as in Figure 52), showing the time to reach the Canon node with the scope pointer from the user's location with different cache sizes. The latency model used is described above in Section 5.2.2.

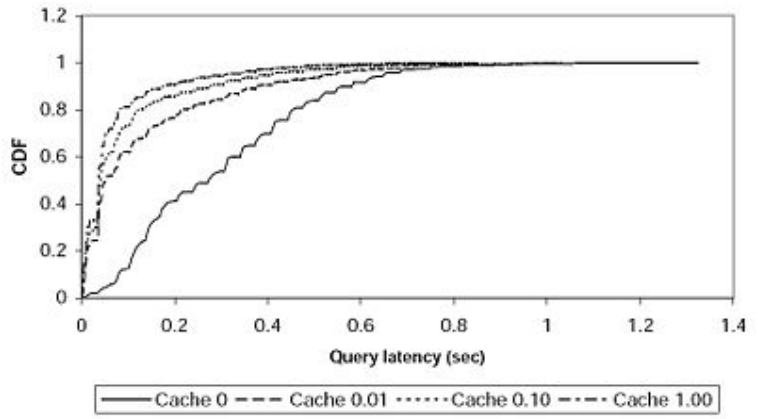

Figure 54: Query latency with cache size (Zipf -1.0).

5.3 Feasibility

The communication, memory and processing resources required to implement the rendezvous networks must scale to the growing global Internet namespace sizes with reasonable cost, considering the likely technology development trends [ITR2008]. Poor scalability implies higher costs; these will limit the scope of the system, the types of objects the system will support and therefore cut off the long tail [And2006] of the system.

The feasibility of implementing the rendezvous networks themselves is straightforward to establish based on the earlier work in the literature. Let us begin by considering the individual rendezvous networks, in which the highest storage and processing requirements occur for the root nodes. Since inside the rendezvous networks we utilize DONA-like mechanism for storing the registration, our discussion will parallel that of the DONA evaluation [Kop2007].

Assuming 160-bit scope ids together with two bytes of next-hop information leads to storage requirement of 22 bytes/id. Adding some reserve for storing the subscription state, cached forwarding information, and the overhead of the data structures involved leads to a rough estimate of a 30-40 bytes/entry. Given the capabilities of a typical server nowadays, we can estimate that a typical rendezvous node could thus store around 10^8 such records in memory.

Now supposing that the total number of scopes in the system would be even a couple of orders of magnitude higher than the number of hosts in the Internet, we would end up with around 10^{10} scopes in total. This would imply that even with such a high total scope count, around 10^3 servers would suffice from the storage point of view to support the collection of all rendezvous networks. The exact number of servers needed for the individual rendezvous networks would then, of course, depend on the distribution of scopes into them and on the total number of rendezvous networks.

From the computational complexity viewpoint, the feasibility analysis of DONA [Kop2007] indicates that the number of servers needed to support the incoming subscriptions would be in the range 10^2-10^3 for the aggregate traffic of a tier-1 ISP. However, since by design individual rendezvous networks see in our framework only a fraction of the total aggregate traffic, we expect the requirements in our case to be significantly lower, perhaps by another order of magnitude. These figures are comparable to other major infrastructure services, such as the DNS, showing that the proposed rendezvous network architecture is indeed feasible to adopt and deploy.

The second component affecting the feasibility is the added overhead from interconnecting rendezvous networks. The exact increase in the overheads and its scaling properties depend on the technical solution adopted. Use of rendezvous providers would extend the above discussion in a close analog to the DONA model. Establishing virtual interconnection overlays using, for example, the Canon structure would lead into a small addition in the storage requirements scaling logarithmically with the number of participating domains. This would not, however, influence the above order-of-magnitude estimates on the total number of nodes required.

5.4 Deployability

Deployability of a global, inter-domain technology depends on the interplay of several factors, including at least incentives and the functional and economical feasibility of the system. It is important to understand which parts of a multi-participant architecture can be designed in advance and which will depend on the deployment and run-time incentives of the participants.

As it is not very likely that all parties in the global Internet will deploy the new system in the foreseeable future, a deployment strategy that requires everyone to adopt the system in order for it to operate, is doomed from the start. It seems desirable to aim for an incremental deployment requirement: Any party deploying the system by itself should get some utility out of the system. Additionally, any parties who have deployed should be able to interconnect their systems so as to gain any possible benefits.

Deployment is determined by stakeholder incentives. If a party is required to invest in the system, that party should also be able to expect benefits at least in proportion to the (required) investment. Requiring investment without any obvious benefits is a recipe for technology failure. Even the chances of operational disincentives (like possible loss of future revenue) are enough to make the system a hard sell. Dependence on third party deployment should be kept to a minimum in order to lessen the effect of operational disincentives. However, the other major objectives (scalability and efficiency) may require some level of functional specialization within the overall system.

From the above we can deduce that when applied to the current internetworking market, we cannot expect all ASes to participate in or even support a rendezvous solution [Rat2005]. Therefore, while clustering deployment along AS adjacencies is beneficial for efficiency reasons, it cannot be assumed to happen universally. Especially, as argued in [Raj2008], we should not expect large transit providers (the so called tier-1 ASes) to support a system that may take traffic away from their networks. Some less obvious disincentives may apply at any AS, so the architecture should be prepared to bypass any AS, and only rely on support of the ASes who actually want to take part in the solution.

5.4.1 Analysis of Interconnection Incentives

The faith-sharing principle [Cla1988] suggests that the local ISPs are natural candidates for rendezvous provision. They also have a built-in incentive to provide the rendezvous service, as it will enable a high degree of localization of data-oriented traffic [Xie2008].

The rendezvous service has the quality of a two-sided market [Roc2004] with network effects, and with the providers and users of the objects in their respective sides of the market. The number of providers in the rendezvous service affects the value users place on the service and vice versa, i.e. there are cross-market network effects that define two-sided markets. This, combined with the socio-economic complexity of the Internet, makes deployment challenging.

As an example of economic uncertainties, let us assume that the rendezvous service has network effects, i.e. the utility grows super-linearly, say O(N log N), where N is the size of the rendezvous network [Bri2006]. If there are two rendezvous network providers of sizes N_i and N_j, should they co-operate by interconnecting their networks or not?

The utility that each operator gains from settlement-free interconnection compared to the status quo is on the order of $N_i \log ((N_i + N_j)/N_i)$. But in a competitive market, one may not expect the status quo to remain the same. If either (or both) operators expect to gain market share or even to capture the full market itself, it may be beneficial to refuse interconnection or to offer it only on a monetary settlement basis. The beliefs that market participants hold about their future with or without interconnection shape their responses and evolution of the system. Interconnection cannot be forced, and thus takes place only when both participants believe they gain from it, compared to their beliefs of the future without interconnection.

With multiple rendezvous providers and the AS structure of the Internet, the situation becomes more complicated. As an example, consider rendezvous being deployed by ASes in a transit relationship. The transit customer and provider can utilize their combined resources more efficiently if the provider also offers a rendezvous transit service, i.e. only the provider makes the full rendezvous state explicitly available and the customer relies on its provider for those items that do not lie within its own area. Similarly, peers in the AS structure may benefit by directly interconnecting their rendezvous state, instead of using their transit AS as rendezvous transit between them. This means that we can utilize parts of the AS structure in the manner of [Kop2007] as a model for forming rendezvous networks.

5.4.2 Comparison to Alternative Solutions for Interconnection

There are many possible approaches to the problem of rendezvous network interconnection. It seems that none of them can be prescribed, as the intricate preferences of the network stakeholders are not known in advance and also develop over time.

The most obvious alternative is that a central entity in the whole global network takes the responsibility of managing rendezvous signalling reachability between all rendezvous networks. However, this option is fraught with challenges relating to e.g. trust, incentives, competition, etc.

Some of these challenges could be addressed by competition of multiple such entities. The providers would compete in global rendezvous reachability coverage and thus would invite all the rendezvous networks to deliver their reachability state to them. Additionally, such competing rendezvous providers could then proceed to exchange rendezvous state between themselves, in practice shifting the interconnection problem up one level. Proactive state exchange between such providers is one possibility, but that again leads to explosion of the state management overhead.

Finally, we observe a possibility for a fully distributed mechanism for rendezvous network interconnection which is the solution that we have chosen for the PSIRP architecture. The participating rendezvous networks self-organize to interconnect without any third party rendezvous infrastructure.

As global mutual trust is out of the question, multiple such interconnection overlays will be necessary in practice. While the virtual structures can fully operate without 3rd party rendezvous infrastructure, it may be beneficial to incorporate the virtual hierarchies into legal entities for contractual reasons. This would decrease the number of needed contracts from the maximum peer-wise number of N^2 to something closer to a linear function of the number of rendezvous networks, as each participant would need to contract only with the interconnection-related legal entity.

Our basic design criteria for the virtual interconnection structure is to enable efficient on-demand distribution of rendezvous state. This would leave most of the existing scope-related reachability state local to their rendezvous networks, as only a small fraction of all scopes are expected to gain global popularity. We also aim to utilize the expected power-law scope popularity distribution to efficiently utilize caching to keep the average rendezvous routing latency low [Ram2004].

The Canon hierarchy provides two important properties for efficiency. The first of these is locality, that is, making the communication within a subdomain stay within that subdomain, rather than leaking out to rendezvous networks outside that subdomain. This property hinges on domain-wise interconnection of the Chord rings maintained at the rendezvous network root nodes, and per subdomain-level registration of the scope reachability state. This distribution of the reachability state also provides robustness against node failures. The second feature is that at each subdomain level, there is a given exit node to the next level for each scope identifier. This makes caching the scope identifier reachability state very efficient, as the other nodes in transit need not cache the returned scope reachability state.

The Canon structure requires the participants to share a common view of the virtual hierarchy. This hierarchy could be informed by the underlying inter-AS topology, which reduces the signalling stretch, but can also be orthogonal to it. This latter option makes it possible to form the structure based on degrees of mutual trust or business interests, regardless of the underlying physical connectivity.

However, as the rendezvous related traffic will only represent a small fraction of the overall traffic, other issues, such as scaling costs, dependency on the transit provider, etc. may hinder the above arrangement. Therefore, we do not expect the whole network to form one homogeneous rendezvous network, but assume a market-based evolution to determine the economical sizes of rendezvous networks, and the type of interconnectedness between them.

5.5 Conclusions on Rendezvous Evaluation and Future Work

We have argued here that our solution steers clear of several potential obstacles for deployment. We have shown that the proposed architecture can be built scalably and securely and is feasible for Internet operators to implement. Also, the latency of the rendezvous operation for end-users is low enough for most use cases.

The evaluation work carried out so far should be seen only as a first step towards comprehensive evaluation of the proposed architecture. Our reliance on the present-day structure of the Internet as the foundation of the evaluation work is a natural starting point, but can by no means be considered to be the final word on the subject. First, the structure of the Internet itself is heavily evolving, and issues such as IPv4 address space exhaustion have the potential to dramatically change the way the forwarding infrastructure of the Internet is organized. Second, the proposed introduction of a global rendezvous service is an example of a potentially disruptive technology that has the potential to affect and create market structures that are essential to modelling the creation of rendezvous networks. For example, it would certainly be interesting to revisit economically driven models for the evolution of the AS graph ([Cha2006], [Cor2007]) from this viewpoint.

There are many ways for continuing the evaluation work presented here. We list here some possible refinements to the high-level simulation model and the additional measurements that may be performed:

- Link and node loads simulation,
- improved network model that takes into account the multitude of physical connections between ASes in different Internet Exchange Points represented here as a single logical connection,
- modelling the locality of the traffic patterns and different link latencies based on physical distance,
- modelling bursts and network failures,
- trying new usage scenarios like CDN, pub/sub, home storage, and cloud computing and understanding future network evolution,
- comprehensive sensitivity analysis, and
- packet-level simulations of a subset of the system and testing of the prototype implementation.

6 Conclusions

This deliverable presented the first major evaluation of the PSIRP architecture and its underlying technologies, as laid out in [Psi2009]. Given the nature of the PSIRP project, it is clear that such evaluation is broad so that this deliverable must be seen as a first step and beginning only. It will be continued in further technical reports and deliverable D4.5, including more detailed evaluation studies and considering an even broader range of technologies. The scope of the evaluation in this deliverable ranged from socio-economic evaluation over security design and evaluation of specific technologies in important architectural areas such as rendezvous and intra-domain forwarding.

The socio-economic work presented a methodology for evaluating the PSIRP architecture from a value chain dynamics perspective, also lending itself to the development of business models for our architecture. The work clearly needs to be seen as the beginning of a larger effort to better understand the markets created by technological choices, such as the rendezvous solution in PSIRP, and identify future players that will be enabled by the choices we made during our design. For this, we identified the set of control points and triggers influencing these control points – both of which will be used in the development of system dynamics models for particular problems that we formulated. The work also laid ground for future architecture design work, e.g., in the topology formation area, where we intend to understand the different design choices possible from a socio-economic perspective.

The security work provided a comprehensive security evaluation of major architectural components. Threat scenarios and ways to overcome potential threats, better securing our solutions, were presented for areas like rendezvous, the node-internal architecture or the topology formation process. We based the work on the current understanding of our architecture, as expressed in D2.3 [Psi2009]. Given our overall design methodology, outlined in [Psi2009], the results of the security work are directly fed back into the overall design.

On the quantitative side, several architectural solutions and mechanisms were evaluated, including the intra-domain forwarding solution based on zFilters and the inter-domain rendezvous solution. We provided methodologies for evaluation, scenarios, and results that showed the viability of our solutions. For instance, the developed zFilter-based forwarding solution was evaluated for scalability in metro-sized networks with moderate sized multicast groups. In parallel, an overlay multicast based option for supporting PSIRP in general and mobile nodes in particular was evaluated in terms of routing efficiency and state maintenance overhead. We also outlined scalability and stretch evaluations of the rendezvous solution [Psi2009]. For this, we developed a topology model based on current topology data and a classification that lends itself specifically well to data-centric scenarios.

In conclusion, the initial evaluation work for the PSIRP architecture and technologies yields very promising results. Our socio-economic understanding is growing with promising insights into the control and markets to be created. The security work shows that our red team approach works very well, feeding back results into the overall design to strengthen the output. And the first quantitative results show promising scalability and viability of crucial components like rendezvous and intra-domain forwarding. Concretely, we can conclude that

- The socio-economic evaluation provides the first step to a better understanding of the markets enabled by our architecture components.
- The security work shows that the security design of the PSIRP architecture results in better protection of critical elements of the architecture.
- Our first quantitative results for the crucial components of intra-domain forwarding show an impressive scalability for metro-size networks while the rendezvous evaluation shows promising scalability results in terms of achieved stretch.

References

[Ahl2000] R. Ahlswede, N. Cai, S.-Y.R. Li and R.W. Yeung, “Network Information Flow,” IEEE-IT, volume 46, number 4, pages 1204–1216, July 2000.

[And2004] T. Anderson, T. Roscoe, and D. Wetherall, “Preventing internet denial-of-service with capabilities,” Proceedings of Hotnets II, pages 39–44, 2004.

[And2006] C. Anderson, “The Long Tail,” Hyperion Books, 2006.

[Arg2005] K. Argyraki and D. Cheriton, “Active internet traffic filtering: Real-time response to denial-of-service attacks,” Proceedings of Usenix, 2005.

[Art2005] M. Artigas, P. Lopez, J. Ahullo, and A. Skarmeta, “Cyclone: A Novel Design Schema for Hierarchical DHTs,” Proceedings of IEEE P2P'05, 2005.

[Bau2007] I. Baumgart, B. Heep, and S. Krause, “OverSim: A flexible overlay network simulation framework,” Proceedings of the IEEE Global Internet Symposium, 2007, pp. 79–84.

[Bha2005] A. Bharambe, C. Herley, and V. Padmanabhan, “Analyzing and improving BitTorrent performance,” Technical Report MSR-TR-2005-03, Microsoft Research, 2005.

[Bin2006] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, “Improving traffic locality in bittorrent via biased neighbor selection,” Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, page 66, Washington, DC, USA, 2006.

[Bit2009a] BitTorrent development community, “BitTorrent Protocol Specification v1.0,” available at http://wiki.theory.org/BitTorrentSpecification.

[Bit2009b] BitTorrent.org, “BitTorrent Protocol Specification,” available at http://www.bittorrent.org/beps/bep_0003.html.

[Bra1988] C. P. R. Braden and D. Borman, "Computing the Internet Checksum," RFC 1071, 1988, available at http://www.ietf.org/rfc/rfc1071.txt

[Bri2006] B. Briscoe, A. Odlyzko, and B. Tilly, “Metcalfe's Law is Wrong,” IEEE Spectrum, volume 43, number 7, pages 34-39, 2006.

[Cai2008] CAIDA, “The CAIDA AS Relationships Dataset, August 18th, 2008,” available at http://www.caida.org/data/active/as-relationships/

[Cas2002] 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.

[Cas2003a] M. Castro, M. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast using peer-to-peer overlays," Proceedings of IEEE INFOCOM'06, vol. 2, 2003, pp. 1510–1520.

[Cas2003b] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," in Proceedings of the Symposium on Operating Systems Principles, 2003, pp. 298–313.

[CFP2008] Communications Futures Program, "The Value Chain Dynamics Methodology", Whitepaper of the Communications Futures Program consortium at MIT, available at http://cfp.mit.edu, 2009.

[Cha2007] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System", Proceedings of the ACM Internet Measurement Conference, October 2007.

[Cha2005] H. Chang, S. Jamin, Z. Mao, and W. Willinger, "An Empirical Approach to Modeling Inter-AS Traffic Matrices", Proceedings of the ACM SIGCOMM IMC'05, pp. 139-152, 2005.

[Cha2006] H. Chang, S. Jamin, and W. Willinger, "To Peer or Not to Peer: Modeling the Evolution of the Internet's AS-Level Topology," Proceedings of the IEEE INFOCOM'06, April 2006.

[Cho2003] P. A. Chou, Y. Wu and K. Jain, "Practical Network Coding", Proceedings of the Allerton conference, 2003.

[Cla1988] D. Clark, “The Design Philosophy of the DARPA Internet Protocols,” ACM SIGCOMM Computer Communication Review, vol. 18, issue 4, pp. 106-114, August 1988.

[CFP2008] Communications Futures Program, "The Value Chain Dynamics Methodology", Whitepaper of the Communications Futures Program consortium at MIT, available at http://cfp.mit.edu, 2009.

[Cob2000] Cobbs, A, “All about netgraph,” available at http://www.daemonnews.org/200003/netgraph.html, March 2000.

[Cor2007] J. Corbo, S. Jain, M. Mitzenmacher, and D. Parkes, "An Economically-Principled Generative Model of AS Graph Connectivity", Proceedings of NetEcon+IBC, 2007.

[Fel2007] Anja Feldmann, "Internet Clean-Slate Design: What and Why?", ACM SIGCOMM CCR, volume 38, number 3, pp. 59-64, 2007.

[Fre2005] M. Freedman, K. Lakshminarayanan, S. Rhea, and I. Stoica, "Non-Transitive Connectivity and DHTs." Proceedings of USENIX WORLDS 2005, December 2005.

[Gan2004] P. Ganesan, K. Gummadi, and H. Garcia-Molina, “Canon in G Major: Designing DHTs with Hierarchical Structure,” Proceedings of ICDCS, March 2004.

[Gil2007] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube Traffic Characterization: A View From the Edge", Proceedings of ACM SIGCOMM IMC'07, pp. 15-28, 2007.

[Guo2007] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of BitTorrent-like peer-to-peer systems," IEEE Journal on Selected Areas in Communications, volume 25, number 1, pp. 155–169, 2007.

[Har2003] N. Harvey, M. Jones, S. Saroiu, M. Theimer, and A. Wolman, "SkipNet: A Scalable Overlay Network with Practical Locality Properties", Proceedings of USENIX USITS'03, 2003.

[Hao2007] F. Hao, M. Kodialam, and T. V. Lakshman, "Building high accuracy bloom filters using partitioned hashing", Proceedings of ACM SIGMETRICS '07, pages 277-288, 2007.

[Hen2008] T.R. Henderson, M. Lacage, M. and G. F. Riley, "Network Simulations with the ns-3 Simulator", Demo paper at ACM SIGCOMM '08, 2008.

[Ho2006] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi and B. Leong. A Random Linear Network Coding Approach to Multicast. IEEE/ACM Transactions on Networking, volume 52, number 10, pages 782–795, 2006.

[Hui2007] F. Huici and M. Handley. An edge-to-edge filtering architecture against DoS. "ACM SIGCOMM Computer Communication Review", volume 27, number 2, pp. 39-50, 2007.

[IAN2009] IANA, “Autonomous system numbers,” available at http://www.iana.org/assignments/as-numbers.

[Ion2002] J. Ioannidis and S. Bellovin, “Implementing pushback: Router-based defense against DDoS attacks,” In Proceedings of NDSS, 2002.

[ITR2008] ITRS, "International Technology Roadmap for Semiconductors", 2008, available at http://www.itrs.net/

[ITU2007] ITU, “H.264 : Advanced video coding for generic audiovisual services,” available at http://www.itu.int/rec/T-REC-H.264, 2007.

[Kar2005] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet service providers fear peer-assisted content distribution?" In Proceedings of the Internet Measurement Conference, pages 63–76, 2005.

[Kat2005] S. Katti, D. Katabi, W. Hu, H. Rahul and M. Medard, ”The Importance of Being Opportunistic: Practical Network Coding for Wireless Environments,” Proceedings of the Allerton conference, 2005.

[Koe2003] R. Koetter and M. Medard, “An Algebraic Approach to Network Coding,” IEEE/ACM Transactions on Networking, volume 11, number 5, pp. 782–795, 2003.

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

[Liu2005] H. Liu, V. Ramasubramanian and E. G. Sirer, "Client behavior and feed characteristics of RSS, a publish-subscribe system for web micronews", Proceedings of the ACM Internet Measurement Conference, 2005.

[Liu2008] X. Liu, X. Yang, and Y. Lu, “To filter or to authorize: network-layer DoS defense against multimillion-node botnets,” Proceedings of the ACM SIGCOMM 2008 conference on Data communication, pp. 195–206, 2008.

[Lum2009] S. Lumetta and M. Mitzenmacher, "Using the Power of Two Choices to Improve Bloom Filters", Internet Mathematics, vol 4 number 1, p. 17-33, 2009.

[Lun2008] D. S. Lun, M. Medard, R. Koetter and M. Effros, “On coding for reliable communication over packet networks,” Physical Communication, volume 1, number 1, pages 3–20, 2008.

[Mah2002] R. Mahajan, N. Spring, D. Wetherall and T. Anderson, "Inferring Link Weights using End-to-End Measurements", Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement, pp. 231-236, 2002.

[May2002] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric", Proceedings of IPTPS02, 2002.

[Oli2008] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and Lixia Zhang, "In Search of the Elusive Ground Truth: The Internet's AS-level Connectivity Structure", SIGMETRICS Perf. Eval. Rev., volume 36, pp. 217-228, 2008.

[Orl2007] S. Orlowski, M. Pióro, A. Tomaszewski, R. Wessäly, "SNDlib 1.0-Survivable Network Design Library", Proceedings of the 3rd International Network Optimization Conference (INOC 2007), 2007.

[Per1999] C. Perkins and K.-Y. Wang, "Optimized smooth handoffs in mobile IP," Proceedings of the IEEE International Symposium on Computers and Communications, pp. 340–346, 1999.

[Per2004] D. Johnson, C. Perkins, and J. Arkko, "Mobility support in IPv6," RFC 3775 (Proposed Standard), June 2004, available at http://www.ietf.org/rfc/rfc3775.txt

[Pou2005] J. Pouwelse, P. Garbacki, D. Epema, and H. J. Sips. "The bittorrent p2p file-sharing system: Measurements and analysis," Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS), volume 3640, pages 205–216. LNCS, February 2005.

[Psi2009] M. Ain (ed.), "PSIRP Deliverable 2.3: Architecture Definition, Component Descriptions, and Requirements", 2009.

[Raj2008] J. Rajahalme, M. Särelä, P. Nikander, and S. Tarkoma, "Incentive-Compatible Caching and Peering in Data-Oriented Networks", Proceedings of ReArch'08, 2008.

[Ram2004] V. Ramasubramanian and E. Sirer, "The design and implementation of a next generation name service for the Internet," Proceedings of ACM SIGCOMM'04, October 2004.

[Rat2005] S. Ratnasamy, S. Shenker, and S. McCanne, "Towards an Evolvable Internet Architecture", ACM SIGCOMM'05 Proceedings, pages 313-324, 2005.

[Roc2004] J. Rochet and J. Tirole, "Defining Two-Sided Markets", IDEI, January 2004.

[Row2001] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems", Proceedings of Middleware 2001, November 2001.

[She2008] R. Sherwood, A. Bender and N. Spring, "Discarte: a disjunctive internet cartographer", SIGCOMM Comput. Commun. Rev., vol. 38, pages 303-314, 2008.

[Sto2003] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications", IEEE/ACM Trans. Netw., volume 11, pages 17-32, 2003.

[Tan2001] H. Tangmunarunkit, J. Doyle, R. Govindan, W. Willinger, S. Jamin, and S. Shenker, "Does AS size determine degree in as topology?", SIGCOMM Comput. Commun. Rev., volume 31, number 5, 2001.

[Wan2005] Wana, T, “Improving time measurement in the test traffic measurements project,” Master’s thesis, University of Applied Sciences Wiener Neustadt, 2005.

[Wen2006] D. Wendlandt, D. Andersen, and A. Perrig, “Fastpass: Providing first-packet delivery,” Technical report, CMU cylab, 2006.

[Xie2008] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications", Proceedings of ACM SIGCOMM'08, pp. 351-362, August 2008.

[Yan2004] A. Yaar, A. Perrig, and D. Song, “SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks,” Proceedings of the IEEE Symposium on Security and Privacy, pp. 130–143, 2004.

[Yan2005] X. Yang, D. Wetherall, and T. Anderson, “A DoS-limiting network architecture,” Proceedings of ACM SIGCOMM, 2005.

[Zha2006] B. Zhang, T. Ng, A. Nandi, R. Riedi, P. Druschel, and G. Wang, "Measurement based analysis, modeling, and synthesis of the internet delay space", Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pp. 85-98, 2006.

[Zha2001] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing", April 2001.