PSIRP
PUBLISH-SUBSCRIBE
INTERNET ROUTING
PARADIGM

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

DELIVERABLE D4.4

Description of validation and simulation tools in PSIRP context

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	02.08.2009
Authors	Janne Riihijärvi (RWTH), Dirk Trossen (BT), András Zahemszky (LMF), Dmitrij Lagutin (HIIT), George Xylomenos (AUEB), Jarno Rajahalme (NSNF), Kari Visala (HIIT), Borislava Gajic (RWTH), Pekka Nikander (LMF)
Responsible of the deliverable	Janne Riihijärvi (RWTH) Phone: +49 2407 575 7034 Email: jar@mobnets.rwth-aachen.de
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/

SEVENTH FRAMEWORK PROGRAMME

European Commission
Information Society and Media

Table of Contents

1 Introduction

This document describes the tools used and developed for the evaluation activities in the PSIRP project. As the objective of the project is the development of a new Internetworking architecture based on the use of publish-subscribe communication directly on the network layer, the domain of the evaluation work is very broad. Technical questions regarding scalability and performance need to be studied for networks of different sizes, ranging from small local area networks to Internet scale. Additionally, due to its scope, the architecture work requires support from further evaluation activities related to issues such as security and the potential socio-economic impact of the new architecture.

All of these evaluation activities require proper tools. In deliverable D4.1, we outlined the foreseen types of tools to be used, and the criteria to be used in the selection process. The main conclusions of that document have been found to be valid, and the present document describes the outcomes of the selection processes alongside with descriptions and accounts of our experiences on working with the selected tools. In some cases, no appropriate existing tools were found, and new ones were developed as a part of the evaluation activities. The rationale and outcome of such development work is also discussed in this document.

The rest of the document is structured as follows. In Section 2, tools for socio-economic validation activities are discussed. Section 3 focuses on packet-level simulation tools, namely ns-3 and OMNeT++, both of which have been used and extended extensively in the project. In Section 4, we discuss the challenges and solutions for simulating Internet-scale networked systems, focusing on the performance evaluation of the PSIRP rendezvous solution. Finally, in Section 5, platforms for testing and evaluating different prototype implementations produced in the project are discussed, before concluding the document in Section 6.

2 Tools for Socio-Economic Validation

D4.2 describes the overall toolkit [Rii2009] that was established for performing the socio-economic evaluation. This toolkit executes a number of steps to determine use cases, actors, control points, triggers etc. The intention is to support the development of system dynamics models for particular use cases and problems associated with these use cases.

There are three areas of evaluation that require tools for performing the described socio-economic evaluation, namely:

- **toolkit**: the operation of the steps described in D4.2
- **reference modes**: the development of so-called reference modes for the stocks within the system dynamics (SD) models
- **SD models**: the particular system dynamics models that are derived from the toolkit work

In the following, we will give a brief outline of the various tools being used in the abovementioned areas.

2.1 Toolkit

The toolkit, i.e., the collection of steps as outlined in D4.2, is implemented using *mind mapping technology*. In other words, the relations of the different steps and the information within each step are visualized utilizing mind mapping concepts. For instance, *actors*, *components* as well as *services* in the **sketch & scope** step (see [Rii2009]) are visualized as *floating topics* within a mind map *sheet*. Relations between topics and sub-topics are shown with different mind map visualizations, mostly utilizing the *tree* or *map* concepts.

Embedding the steps into such mind mapping concepts allows the person performing the evaluation as well as the stakeholders involved in the evaluation to grasp the right level of detail while still providing the ability to insert more information, e.g., notes.

Mapping the toolkit steps onto such common mind mapping concepts allows the use of essentially any mind mapping software that is available. No specific concept is used in order to make portability easy. In our case, we use the (free) open source software *XMind* [Xmind2009] on Macintosh as well as Windows machines. The software allows the exporting of the results to HTML and the creation of various image formats, which allows the inclusion of the visualized concepts in other documents.

The basis for each particular use case is a *base mind map*, visualizing the concepts of the toolkit, according to D4.2. This base mind map is then adapted to the particular case at hand through creating a copy and modifying the relevant steps, including removing or skipping the irrelevant ones. Currently, there are three base mind maps for the following three cases:

- **design evaluation**: allows the evaluation of a particular architectural design, as currently performed for the PSIRP rendezvous solution (see D4.2)
- **design choice evaluation**: allows the evaluation of a variety of architecture design choices, as currently performed for the PSIRP inter-domain topology formation function (see D4.2)
- **business modelling**: allows the development of a variety of business models for a particular use case.

The different base mind maps essentially differ by having removed unnecessary steps or elaborating on particular steps that are of importance for the evaluation case.

2.2 Reference Modes

Reference modes are an essential component for any system dynamics modelling process [Ster2000]. A reference mode for a *stock* in an SD model describes the expected behaviour of the stock on a system level, i.e., under the influence of the various causal loops that will eventually be developed. Usual factors appearing in reference modes are *take-up*, *death*, *exponential growth* and alike.

While SD modelling tools (see next section) usually allow the drawing of such reference modes, we decided to utilize basic curve drawing capabilities in Microsoft PowerPoint. In other words, the basic s-curve as well as the declining curves in reference modes are drawn with the curve functionality in PowerPoint. Boxes and normal text functionality is then used for the annotation of the various reference zones.

For this, a basic reference mode has been created with PowerPoint. This basic reference mode serves as a template for the reference modes to be created in our evaluation. Additional declining curves can be added through copy-and-paste or existing ones can be deleted. The major reference zones are already marked and need mere revision for the case at hand. This allows for a simple yet appealing visualization of the reference modes which can also quickly be used in presentations.

2.3 SD Models

We utilize the *Vensim* modelling software [Ven2009] for capturing the stock/flow models that are derived from the toolkit steps (see Section 2.2.1). The causal loops are drawn, starting off with the auxiliary variables identified in the toolkit steps. Vensim allows the drawing of stand-alone causal loops or complex stock flow models with causal loops (the latter being the usual case for our evaluations).

Vensim further allows for a variety of pre-defined functions to be defined for each causal loop variable. Different forms of simulations are supported, including a real-time mode in which variables can be changed with a slider while observing the overall simulation output.

As mentioned above, we chose to not use the Vensim reference mode functionality since it was deemed unacceptable in terms of usability.

While we appreciate that the Vensim software is not free or open source software, the free license version for educational purposes still allows the user to perform most of the necessary SD modeling steps. More complex operations, such as the real-time mode, require a professional license. However, the Vensim software has been evaluated as being superior to other software packages in terms of supported functionality.

3 Packet Level Simulation Tools

For rapid evaluation of different networking solutions in terms of performance packet level simulations are the standard approach. They are actively used in PSIRP as well. In this section we give an overview of the simulation platforms selected in D4.1, and discuss our experiences in applying them in a project of a clean-slate nature.

3.1 Using ns-3 for simulation and network emulation

3.1.1 General overview

Ns-3 [Hen2008] is an open source discrete-event network simulator intended mainly to provide an advanced simulation tool for networking research and education. It is publicly available, free software licensed under the GNU GPLv2 license, making it an attractive platform for simulator-based performance evaluation in PSIRP. In the following we shall give a short overview of the ns-3 design, and discuss its applications within the project.

One of the ns-3 design objectives is to replace the ns-2 network simulator, even though it is not compatible with it. Although both simulators are written in C++, and some code has been successfully ported from ns-2 to ns-3, ns-3 does not support ns-2's APIs. While ns-2 is typically scripted in OTcl and uses nam (Network Animator) for visualization, ns-3 code is completely written in C++ with optional Python bindings and weak nam support. Ns-3 uses the WAF Python based building system. For tracing, ns-3 provides pcap file support for every device in the simulation. A common tool for displaying pcap files and visualizing the results is Wireshark, which provides a nice user interface as well as various analysis and statistical tools. This allows the unification of performance analysis tools for simulation and experimental testbeds reducing the overall development effort in the project.

For a device that connects to a network ns-3 uses the general term *Node* representing the basic computing device abstraction. The Internet nodes are constructed in a way that more faithfully represent real computers embracing key network abstractions, e.g., sockets, network devices, multiple interfaces for nodes, using IP addresses. A node is then connected to the network via an instance of the *Channel* class. A channel in ns-3 can model simple point-to-point connection, i.e., PointToPointChannel, but can also model more complex wireless or Ethernet channels, e.g., WiFiChannel, CsmaChannel, BridgeChannel.

In real life, hardware components do not work properly without suitable device drivers. Ns-3 provides such an abstraction through *NetDevices* which play the role of software drivers and simulated hardware at the same time. Thus, the Node is able to communicate with more than one other Node using different Channels by installing proper NetDevices. Applications are the basic abstractions, which drive the execution of the ns-3 simulation, e.g., OnOffApplication generating constant bit rate traffic or UdpEchoClientApplication/ UdpEchoServerApplication representing a client/sever set for generating UDP echo traffic. Additionally, ns-3 provides the *Helper* functionality to ease creating and connecting the network components, i.e., Nodes, NetDevices and Channels in proper way. Each ns-3 packet consists of byte buffer, tags and metadata. The byte buffer contains a serialized representation of the packet's headers. Tags are responsible for linking particular subset of bytes in the packet with information stored in the tag. Metadata contains ancillary information of headers and trailers related to the packet.

Ns-3 supports a wide range of different advanced network features, e.g., Wi-Fi models, IPv6 addressing, bridging different LAN segments and a network simulation cradle allowing real world TCP/IP stack to be used in simulations. The integration of ns-3 simulations with real world traffic is done with the tap device allowing real hosts to communicate via simulated networks and network emulation devices allowing simulated nodes to exchange information through real links. Currently supported operating systems are Linux x86, and x86_64, MacOS X ppc and x86, and Windows under the cygwin and mingwin runtime environments.

Due to its object oriented nature, extending ns-3 is intuitive, via the creation of entirely new classes, class inheritance and class aggregation. Creating a new protocol's headers and trailer is performed by inheriting from the default basic Header and Trailer classes of ns-3 and redefining their main methods: Serialize, Deserialize, GetSerializedSize and Print. Therefore, ns-3 is appealing for evaluating the PSIRP architecture since it provides a comfortable way of realizing completely new protocols, not relying on any existing TCP/IP implementation. Moreover, the integration of the created protocols into the present ns-3 structure is rather simple because of its C++ character.

Ns-3 has already been used in PSIRP as a simulation tool for analyzing specific aspects of network protocols, e.g., in the network coding performance evaluation where we studied network coding behaviour under various conditions, or for the analysis of the zFilter-based forwarding. Furthermore, PSIRP utilizes the ns-3 emulation facility as a foundation of preliminary interoperability testing by connecting BSD nodes through a simple ns-3 network [Rii2009].

3.1.2 Extending ns-3 with zFilter-based forwarding

To get a deeper understanding of the zFilter-based forwarding mechanism's performance, we implemented the protocol as an extension to ns-3. While until now not all advantages of ns-3 have been fully exploited, the plan is to continue to extend the protocol with additional details (z-Formation, and possibly transport models), where packet-level analysis will be more beneficial. The detailed analysis of the performance of the zFilters has already been discussed in a previous deliverable, while this report is focusing merely on the implementation details of the simulator. The architectural overview of the zFilter-based forwarding mechanism was presented first in D2.3.

Our extensions to ns-3 are all written in C++. The implemented model is fine-tuned for *zFilter multicast data delivery*; node-internal publish/subscribe and signalling messages are not used. Also, at some points the simplified model may be a bit different to the conceptual architecture. The implementation follows a layered model (instead of the Component Wheel). The *Upper Layer* is only partially implemented and very much simplified, used mainly for keeping track of the publication interests of the nodes. The *Forwarding Layer* is on the other hand better detailed, and contains the basic multicast forwarding method and some additional features (forwarding optimization, fast reroute, and loop prevention). In the following, the two layers are discussed, as well as helpers for different purposes (topology, statistics etc.).

3.1.2.1 Upper Layer

In ns-3, in order to define a new protocol, at least two new classes need to be implemented. One is responsible for the protocol behaviour and instances in all nodes using the protocol, while the other is the representation of the protocol's packet header. For the Upper Layer, the header -in this simplified model- only contains the Rendezvous Identifier. The nodes keep track of the publications (simply referred to by Rendezvous Identifiers) they are interested in. The Upper Layer's API is used directly by the code that defines the simulation scenario. With this simple API, the Upper Layer can be programmed to listen to a publication specified by a Rendezvous Identifier. This results in a registration in its internal table for publications, allowing the node to know whether a packet is meant for it (i.e., it has subscribing applications), or if it is merely a forwarding node or has received the packet as a consequence of a false positive in the forwarding decisions. With the other feature of the API, the scenario defining code can request the Upper Layer to send out a data publication with the specified Rendezvous Identifier (and other information that needs to be put to the Forwarding Layer's fields). The Forwarding Layer delivers to the Upper Layer all received packets as well as additional information about the processing of the packet (see later). The Upper Layer then checks the Rendezvous Identifier and decides whether or not it has a subscriber application. Finally, it reports all the information of the packet processing and the Rendezvous Identifier to the statistics-collector module.

This layer is relatively thin and needs to be revised once the Rendezvous and Topology functions will be modelled. The reason for its existence is to make the Forwarding module as independent from the other parts as possible, meaning that the Forwarding Layer implementation is not containing any unnecessary functionality that might be part of a different module in a real-world implementation.

3.1.2.2 Forwarding Layer

The heart of this extension to ns-3 is the system under test, i.e., the Forwarding Layer. It acts on the so-called *Forwarding Header*, which contains a Bloom Filter and 8 bits of flags as additional information, reserved for further extensions. Currently, the index of the forwarding table (for forwarding performance optimization), the backup bit (showing that the packet is on its backup path or its primary path in fast reroute mechanisms) and the time-to-live value (a method for loop prevention) is put into that field. The API it provides for the Upper Layer is simple: it accepts a request to send a packet with an encoding (Bloom filter) of certain Forwarding Identifiers. Such Forwarding Identifiers may be individual Link Identifiers (LIDs) or sub-paths across multiple links (e.g., other Bloom filters). The use of a Bloom filter in this context is referred to as a zFilter. From the other end, it is notified when a packet arrives on one of the Ethernet interfaces (for easier extendibility, the zFilter Forwarding Layer was implemented above Ethernet, with only broadcast address being used). Finally, the topology module has the capability to fill the forwarding tables (including physical and virtual links) and information about the backup paths.

The most important structure is the Forwarding table, which contains *{index of the FId, interface ID}* pairs. Furthermore, it has a loop prevention cache with *(zFilter, expected interface)* pairs. The latter is important to stop looping packets by registering all those zFilters that might arrive from more than one interface. Another way of stopping looping packets is to use time-to-live and the usage of the cache. The TTL can be selected when starting the simulator. Generally, time-to-live has worse results regarding false positives (the forwarding of packets on links that are not required), but does not require extra processing and memory (state) in the forwarding nodes.

Once an Ethernet interface captures a packet, it is handed to the Forwarding Layer through a callback function. The Forwarding Layer then performs the following steps:

- It checks whether the packet is looping either by:
  - checking the Loop Prevention cache
  - decreasing the TTL field and comparing it to zero
- It makes the forwarding decision by matching the appropriate FIds in the Forwarding table with the packet's zFilter. The output is a list of unique interfaces.
- It adds the packet's zFilter to the loop prevention cache, if needed (and if used)
- It calls the Ethernet's Send function in the appropriate interfaces
- It hands the packet to the Upper Layer, along with information about packet processing (e.g. number of matching interfaces etc.)

Later, the simulator was extended to support rerouting around a failed link or a failed node. The fast reroute mechanisms are activated when the packet needs to be sent out on a link that is down. In this case, either the backup path FId (a zFilter of the backup path) is ORed to the zFilter of the packet, or a virtual link is activated and the zFilter is not touched. Other forwarding nodes on the backup path perform simple LIT to LIT forwarding without examining the packet zFilter until the packet arrives at the other side of the broken link and the backup path flag is removed. The backup paths require additional forwarding state consisting of LIT to LIT mapping for intermediate nodes in the backup path, along with LIT to virtual LIT mapping on the originating node and virtual LIT termination information in the destination node.

3.1.2.3 Helpers

There are some global modules that are mainly used for driving the simulations and logging. The topology module is responsible for collecting the information about the network topology and the LITs (through reading a file). It then fills in the nodes' Forwarding Tables. Finally, it is responsible for providing zFilters for the requested delivery trees. The statistics module creates an additional log file, which has information about each Rendezvous Identifier, how many links they traversed through, how many LITs were added to the zFilter, how many interfaces were checked, how many times a loop had to be cut, and how many nodes added an entry to the cache for loop prevention. From this information, the false positive rate and the forwarding efficiency can be calculated, as well as some insight can be gained on loop prevention performance.

The simulation scenario code allows a wide range of parameters to be set. First, the topology file should be fed, which specifies the links and the link identities. Other parameters are: number of publications, number of subscribers per publication, number of parallel forwarding tables, loop prevention method, fast reroute method. After requesting the topology module to create the needed delivery trees, the simulator initiates data delivery by contacting the appropriate node's Upper Layer.

3.1.2.4 Network emulation

With ns-3's network emulation feature, it is possible to make the simulator part of a real network. With this technique, the actual prototype code can be tested in much larger environments than prototyping usually allows. We conducted a small test to evaluate the emulation mechanism’s potential. We used *Emulated NetDevices*; this mode of operation means that the network device in the simulator is connected to a real interface, but viewing from the “top” in the simulator it looks like any other simulated network device so it can be part of the simulated nodes. To achieve this, we had to adjust the simulator's Forwarding Header to the one that the prototype uses. Then, we implemented a simplified forwarding decision: a node copies the packet it receives to its other interfaces. Finally, this simplified implementation was connected to the old BSD prototype, interoperability was verified and some early measurement results were produced. Our previous deliverable D4.2 has more details on ns-3's emulation feature.

3.2 Simulating overlays with OMNeT++

3.2.1 Introduction

A medium term alternative to a native implementation of PSIRP, directly on top of the network hardware, is an overlay implementation on top of IP. An overlay variant of PSIRP should be easier to deploy as it could initially execute only at end hosts. Should it become successful, routers would gradually join the PSIRP overlay so as to offer improved performance, thus allowing networks to eventually drop support for IP.

Our work in the overlay PSIRP variant has so far focused on efficient support for multicast, which is seen as the main enabler for providing the entire PSIRP functionality in the overlay. We have therefore concentrated on overlay schemes supporting efficient multicast routing, investigating their properties and abilities in the PSIRP context. In particular, we have focused on the Scribe overlay multicast scheme [Cas2002] which has been designed to operate on top of the Pastry DHT based content routing scheme [Row2001]. Pastry maintains a locality aware DHT which provides low routing stretch compared to simpler overlays like Chord, while Scribe uses the DHT to implement in a simple but efficient manner multicast trees. In addition to the performance of overlay multicast for distributing publications to subscribers, we have also studied its usefulness for supporting mobility in the PSIRP context. This is a well known idea that was originally proposed for mobility support in IP but never caught on due to the lack of widespread support for IP multicasting.

As part of our work in overlay multicast we have explored the concept of hierarchical DHTs as proposed by the Canon scheme [Gan2004] with the purpose of making the routing scheme better conform to the way actual networks are interconnected on the Internet. This work can also be used for the evaluation of the inter-domain rendezvous network interconnection mechanism, which allows publishers and subscribers to locate the rendezvous network containing a given scope (and therefore its rendezvous point), as the current solution to this problem employs a hierarchical DHT.

For our work on DHT based overlays we have used OverSim [Bau2007], an overlay network simulation framework for the OMNeT++ simulation environment [Omn2009]. The OverSim framework provides implementations of several overlay schemes and applications, including Chord and Pastry, as well as overlay multicast schemes, such as Scribe. 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 [Omn2009], is in operation. As a result, one can either run small simulations on top of realistic networks, or large simulations on top of abstracted networks.

As part of the initial performance evaluation of overlay multicast reported in D4.2, we made multiple enhancements to the IPv4Underlay model provided by the INET protocol framework. In particular, we have extended the model to support access links with asymmetric characteristics (e.g. bandwidth, error rate, etc.), hierarchical networks imported from GT-ITM topologies, routing policy weights and Autonomous system numbers [IAN2009] in the simulated network. Our enhancements improve the realism of the simulated networks and facilitate the investigation of inter-domain routing issues.

Using this simulation environment, we have already investigated the potential benefits of network assistance, i.e. router participation in the overlay, for the provision of multicasting in the context of content distribution. By studying the graph properties of the overlay multicast trees we have demonstrated the benefits of network assistance in terms of routing stretch and multicast tree efficiency. We also utilised this platform to evaluate the performance characteristics of an overlay multicast based scheme for supporting host mobility.

Since static multicast tree properties are insufficient to characterize the dynamic performance of actual applications, we have also worked on a realistic content distribution application model that will serve as a benchmark for the benefits envisioned by the PSIRP architecture. Due to 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 between our overlay multicast based BitTorrent alternative and the regular BitTorrent of today's Internet, 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.

Based on our initial experience with the performance of the overlay multicast scheme with and without network assistance, we decided to extend our simulation platform in two directions. The first direction is related to the hierarchical character of networks and its apparent contradiction with the flat paradigm adopted by DHTs. In this direction we have implemented Crescendo, the Chord based realization of the Canon paradigm for hierarchical DHTs [Gan2004] on the OverSim platform and also designed a Pastry based realization of this paradigm, which is expected to provide the benefits of proximity aware neighbour selection in the DHT identifier space. The second direction is related to the deployment of the overlay multicast based PSIRP variant. In this direction we focused on studying the evolutionary path towards the adoption of the proposed architecture in an incremental fashion, starting with only the end hosts participating in the overlay and gradually adding routers to the overlay either in a single or multiple network domains. To complement these directions, we further enhanced the platform by providing support for the collection of statistics related to inter-domain and intra-domain traffic. In the following we provide a detailed description of the performance evaluation tools developed.

3.2.2 Hierarchical DHTs

Distributed Hash Tables (DHTs) have been proposed as flat, non-hierarchical structures, in contrast to most scalable distributed systems of the past. The primary advantage of DHTs is that they allow participating nodes to operate in a homogeneous manner and provide a uniform distribution of load among them, thus avoiding single points of failure, while maintaining logarithmic scaling properties. Their flat design however does not provide the inherent advantages of hierarchical structures, i.e. fault isolation and security, effective caching and bandwidth utilization, adaptation to the underlying physical network, hierarchical storage, and hierarchical access control [Gan2004].

The Canon paradigm aims to bridge the two worlds of flat and hierarchical network design. It essentially establishes regular DHTs in each participating administrative domain (hereafter simply named domain) and recursively merges them at higher levels in the physical network hierarchy, thus leading to the creation of a hierarchy of overlay structures. Canon offers the same routing state vs. routing hops trade-off provided by standard DHT designs. The essential properties we are interested in are the locality of intra-domain paths and the convergence of inter-domain paths. According to the former property, paths between nodes of the same domain should never leave the domain. Hence, routing stretch is reduced and fault isolation is achieved by not allowing failures in external domains affect intra-domain overlay routing. In the context of the overlay variant of PSIRP and the investigation of content distribution scenarios, this property is especially important since it means that data transfer within an administrative domain will never involve nodes outside the domain, thus reducing traffic over costly peering links between administrative domains. The latter property is expected to further reduce unnecessary inter-domain forwarding of data and to provide better control over outgoing and incoming traffic for each domain. This is due to the fact that in a Canon structure, all paths from an administrative domain towards a node in another domain, will exit the former domain through a single, common node. This means that multicast trees based on reverse path forwarding, as used in Scribe, will enter each domain via a single node, thus simplifying caching, reducing state maintenance and improving tree properties.

Canon can be applied to many different proposed DHTs to construct their Canonical versions. We have implemented Crescendo, the Canonical version of the Chord DHT [Sto2003], in the OMNeT++/INET/OverSim simulation platform, based on the provided implementation of Chord. Our implementation provides a full-fledged simulation environment built on top of hierarchical GT-ITM topologies. Following the transit-stub model of the topologies, Crescendo builds a single Chord ring for each stub domain. It is noted that each such ring follows a reverse assignment of keys' responsibility than regular Chord, so that inter-domain path convergence can be achieved. On top of this multi-ring structure, Crescendo creates a global Chord ring by merging the appropriate links from each stub network. Essentially, extra overlay links are inserted in each stub domain ring, enabling routing towards destinations in different administrative domains. These links are used once a local (i.e. inside a single stub domain) routing query has reached the local node closest to the target identifier.

The Crescendo implementation provides an important simulation tool both for the rendezvous network interconnection mechanism and for the overlay PSIRP variant. Regarding the latter, it completes a full-fledged simulation platform providing the entire protocol stack, including the aforementioned routing enhancements and the hierarchical DHT functionality. It is noted however that it can be adapted to more abstract topologies, with deeper domain hierarchies, that trade realism for scalability. In this case, the simulator can be used to extend the graph-theoretic investigations of the Canon paradigm that have been performed for the rendezvous network interconnection mechanism.

Despite the importance of the intra- and inter-domain properties of the Canon paradigm, the Crescendo implementation is based on the network agnostic Chord DHT. This means that the design of Chord still allows for inefficient routing, since Chord is single dimensioned along the identifier space, completely neglecting network space proximity during neighbor selection. Our performance evaluation indicates that there are clear benefits due to the adoption of the Canon paradigm with respect to routing delay and stretch, however there is ample room for further improvements. It is therefore anticipated that a Canon realization on top of a network-aware DHT scheme will provide a more efficient overlay routing mechanism. Therefore we have focused on the Pastry DHT [Row2001], which provides a two-level structure, taking into account both identifier and network proximity. To this end we have designed a hierarchical version of Pastry which promises better adaptation to the network and we are currently in the implementation process.

3.2.3 Incremental overlay deployment

To this end, we have engaged in the investigation of the potential benefits of an incremental deployment process in which enhanced network entities are gradually deployed on top of the existing IP infrastructure. As a baseline we consider the case where only end hosts participate in the overlay, and then consider the performance benefits when selected subsets of network routers also participate in the overlay, acting as proxies of the end nodes.

In this direction, we have extended our simulation platform to allow the investigation of the potential benefits of the proposed architecture with respect to two dimensions in the deployment process, i.e. intra-domain and inter-domain deployment. In the first dimension, the simulator supports dynamic deployment of an increasing number of overlay multicast enabled network entities inside a single administrative domain. Effectively, the potential gains of the deployment can be studied as a function of the extent of the deployment within that domain, which essentially reveals the size of the investment required by a network operator. Apart from a random deployment of the proposed network entities inside an administrative domain, the simulator also allows the definition of more informed deployment policies, e.g. selecting network location for the deployment based on the fan out or the position of each router inside the domain (for instance, routers in the border of the domain, interfacing with the backbone).

In the second dimension, the simulator allows the incremental deployment of the proposed functionality across administrative domains, i.e. it enables the specification of the number (or percentage) of the administrative domains actually engaging in the deployment process. In this manner, the potential gains can be revealed for individual domains selecting to invest on infrastructure for the proposed functionality, against domains choosing otherwise. It is noted, that the simulation platform allows the concurrent tuning of both dimensions, yielding a rich variety of deployment scenarios.

4 Tools for Inter-Domain Level Simulations

While the packet-level simulation tools discussed in the previous section can be effectively used to study protocol performance on networks with some thousands of nodes, simulating Internet-scale networks is not feasible with them. However, since some components in the PSIRP architecture, such as the rendezvous system, need to scale to such large networks, alternative simulation techniques are required. In this section, we discuss the development of a simulation framework for enabling such Internet-scale simulations by operating on a higher level of abstraction compared to the typical packet-level simulation tools.

4.1 Internet-scale high-level simulation

Systems like the inter-domain rendezvous architecture [Raj09] can have a complex behaviour and their comprehensive validation without building and running the actual system is only possible via simulation. Packet level simulations can be used to test individual rendezvous nodes or small scale installations, but these experiments may not reveal emerging phenomena stemming from scaling the system to its real-life size or unnoticed properties of the actual environment that are not captured by the smaller scale simulator. On the other hand, network simulation should be at least an order of magnitude cheaper to implement than the actual system.

It might be possible to run full size model by scaling the time dimension and running the simulation at a slower speed than the actual system, but in our case, the amount of state stored in the working rendezvous system makes this kind of approach infeasible. Therefore, the only possible approach is to carefully approximate details in the simulation and concentrate on selected aspects of the whole system and measure only them. This approach requires showing that the abstractions do not affect in a significant way the measured properties. Also, the level of detail of the different aspects of the simulation should be in balance: there is no point in implementing some part accurately if approximation elsewhere makes the results uncertain.

In the case of the rendezvous architecture, we both slowed down the time, simulated simultaneous operations sequentially, and modelled the state of the system (caches and queries) only statistically without actually storing any state. In addition, we modelled the network relatively accurately only on the level of domains and their logical links. For intra-domain topology only latencies were considered, which were taken from a distribution that is a function of the size of the domain. This way, we were, among others, still able to have a realistic number of nodes, realistic locations for nodes on AS-level granularity for latency and stretch calculations, and had a more realistic traffic matrix that would have been possible on any packet-level model of the system. In addition to the actual architecture, the simulation also modelled the evolution of the rendezvous networks based on the AS-level topology of the Internet and inferred proximity of the domains and how this affects the system formation. This is also something that is impossible to represent in a smaller scale model.

4.2 PSIRP high-level simulator

An AS-level Internet simulator was developed from scratch in the PSIRP project specifically for the needs of the evaluation of the inter-domain rendezvous architecture. The simulator was originally written in Python and then ported to GNU Octave with small modifications. The simulator is a simple program that is run from the command line, performs a batch run of simulated operations and finally prints and plots the results of the simulation. Many parameters of the run can be easily changed like

- the settings for Chord,
- number of simulated subscriptions,

- inter- and intra-net hop latency estimates,
- domain type distributions, and
- shape of the popularity distribution of RIds.

Although the implementation of the simulator was built around simulating the rendezvous interconnect architecture with little time to re-factor the software design of the simulator, the code is relatively modular. As a consequence, the general valley-free inter-domain routing algorithm and domain type modeling code together with its associated traffic matrix can be easily separated from the more specific code simulating the rendezvous nodes and the evolution of the rendezvous networks. In fact, we are already working on modifying the simulator to be used in the simulation of inter-domain topology formation. The code is also well documented for easy modification by third parties.

4.2.1 Network model

The simulation uses CAIDA's AS relationships dataset [Cai2008] as the Internet inter-domain topology model. The dataset consists of a full AS graph derived from a set of RouteViews BGP table snapshots. While it gives a good picture of tier-1 connectivity and provider-customer links between ASes, it is known to lack many 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.

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

The simulator program reads its input directly from CAIDA's dataset file format that is a single long text file enumerating all known relationships between ASes. This data is enough to calculate all possible logical valley-free paths between domains and estimate the preferred path by using the one with the smallest number of AS-level hops for relatively accurate model of inter-domain routing in the Internet. Therefore, it should be easy to use alternative models of AS-level connectivity for different types of future internet technologies as long as there is an evolution model for forming the business relationship between the domains.

Currently, one of the main drawbacks of the simulator is that we have not yet had the time to implement any kind of network failure model. However, the simulator has to deal with the apparent non-connectivity between some of the domains in the CAIDA datasets. For example, this affects the overlay link selection, where the connectivity between prospective nodes is first tested with the network model, and links are formed only if connectivity exists between the nodes.

4.2.2 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], characterizing each AS in terms of the traffic volumes of three types of network usage, called utilities. The three types used are web hosting (U_{web}), residential access (U_{ra}), and business access (U_{ba}). 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 $U_{web} \sim R_{web}^{c}$, where the power-law exponent c is approximately -0.9 for North America and Europe and -1.1 for Asia-Pacific (they inferred the web traffic amount from web content in search results), and R_{web} 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 $U_{web} + \alpha\ U_{ra}$. 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 U_{ra} 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.

4.2.3 Modeling the evolution of 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: Customers having at most 4 rendezvous customers in their network are accepted. Also, 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 convergence of inter-domain paths. 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 connectivity in the underlying AS interconnection topology, while avoiding stand-alone stub ASes for policy-compliancy reasons. In the simulation we used hop-distance based clustering to figure out which domains would likely become members of a joint Canon subhierarchy. In the real world, multihoming could be utilized by, for example, by splitting the underlying hierarchy into multiple virtual trees to balance the traffic. It follows that the fanout of the hierarchy can greatly vary at different levels of the tree. Especially, the top layers are very flat where tier-1 ASes can provide access to hundreds of customer networks. This is in sharp contrast to the Canon paper evaluation, where a fixed fanout of 10 in internal nodes of the tree was used. Canon hierarchies are joined together via prefix clustering based overlay to provide universal connectivity within the network.

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. 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 $O(n \log n)$ where n is the number of scopes, 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 the 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 the 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 β k 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 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 whole identifier space.

4.2.4 Basic operation and output

The simulator works by simulating random individual subscription operations originating from locations and targeting SIds based on distributions given by the model described above. Messages are followed in the Chord nodes and routed between nodes based on the network model. The core Chord algorithm is fully simulated except that the success of cache operations is based on the high level cache model described above. The limitations of the system are that it is assumed that individual subscription operations do not in any way interact with each other and can be just superimposed to form a complete picture of the system. Also, network errors (and error handling in Chord nodes) are not modelled in the current version. Then all relevant information is stored from the individual run and a new random subscription operation is started. In the end, the results are scaled based on assumption how many subscriptions are performed per second in real-time in the actual system and scaled distributions can be calculated.

The program is relatively efficient and in the later GNU Octave version the routing algorithm was even more optimized which makes it possible to simulate thousands of subscriptions in the time scale of minutes with a typical laptop computer.

At the end, the simulator program outputs latency, stretch, and node load distributions generated by the set of subscription operations run based on the simulation parameters given. It also outputs some general information about the formed rendezvous network and the shape of the interconnection overlay. These measurements are directly useful in evaluating the most important technical properties of the system like the average latency of subscriptions. By running the simulation multiple times and adjusting the simulation parameters, we can also measure the effect of parameters on the relevant evaluation metrics and find their optimal values, like the efficient size for caches in nodes.

4.3 Future work

By the end of 2009 the goal is to package the simulator and release the code as open source. This means that the topology, domain type-, traffic-, evolution-, intra-domain topology-, and system node models must be clearly separated in the code to make it easier for 3rd party developers to use and extend the code for new uses. Also, implementing a network failure model and a more accurate intra-domain topology model would make the code applicable to an even larger set of use cases.

5 Evaluation Platforms and Testbeds

In addition to simulation-based evaluation studying the performance of the actual prototype implementations developed is a major item in the PSIRP evaluation work. In this section we discuss some of the platforms and testbeds used in these activities.

5.1 NetFPGA

5.1.1 Evaluation through hardware-based implementation

The PSIRP project aims to create a networking system that is scalable on the Internet level. Therefore, the forwarding solution must be scalable to the high speeds used by core network routers.

To evaluate the feasibility of our approach, we have implemented the forwarding functionality on a field programmable gate array (FPGA) based hardware. This has been done for several reasons. First, the PSIRP project is developing new networking protocols that are not supported by the current hardware. Furthermore, purely software based prototype can not reach an optimal performance, since all incoming data must be forwarded to the main CPU for processing. In the FPGA-based hardware prototype, the incoming data can be processed in-place at line speed, without a need for interaction with the main CPU or the operating system. Results from the hardware based prototype will give an important information regarding the overall efficiency and scalability of the forwarding solution.

Additionally, experimenting with an FPGA-based hardware prototype makes it easier to design application specific integrated circuits (ASICs) in the future that are optimized for the PSIRP forwarding function. Such ASICs can be utilized in the future routers making PSIRP easier to deploy.

5.1.2 Stanford NetFPGA as a prototyping platform

NetFPGA [NetFPGA] is a flexible, and open hardware platform for research and classroom experimentation in terms of networking and traffic processing. Designed at Stanford University, and manufactured by Digilent Inc., it ships in the form of a PCI extension card with four gigabit network interfaces, and can easily be plugged in a typical workstation.

Unlike the typical networking hardware, where functionality is strictly defined once it leaves the manufacturer's assembly line, NetFPGA is fully programmable and can process the incoming and outgoing data in any way possible. NetFPGA utilizes the Virtex II-pro 50 FPGA chip, which contains 2 PowerPC CPU cores and over 50,000 programmable logic cells.

The biggest advantage of NetFPGA is the significant community behind it. NetFPGA is widely used in research projects and there is a good support available for it. There also exist several networking implementations for NetFPGA, against which our own implementation can be compared for evaluation purposes. NetFPGA is basically a de-facto standard for network hardware implementations, just like ns-2 and ns-3 are standard network simulators.

The downside is that the hardware inside NetFPGA is somewhat old. The network interfaces are limited to 1Gbps speed, and NetFPGA is attached to the host computer using a standard PCI bus, which becomes a bottleneck if data is sent from the host to several NetFPGA network interfaces. Additionally, the FPGA chip has limited programmable resources and is manufactured following an old 0.13um process. While it is big enough to handle most of the common processing tasks, the limits of the FPGA may be reached in more advanced designs.

5.1.3 z-Filters implementations

The zFilter forwarding algorithm [Jok2009] has been implemented on the NetFPGA. The implementation was based on the Stanford reference switch implementation, which was modified to create a simple zFilter switch. According to early measurements, the efficiency of the NetFGPA-based zFilter forwarding is good with an additional latency of only 3-5µs per hop. The zFilter forwarding on the NetFPGA had actually a lower latency than a reference IP router implementation.

Our experiences with NetFPGA as the prototyping platform are overall positive. It is a solid development platform that is available in a complete package. As the downside, its documentation is somehow lacking, and this made the development of FreeBSD drivers more difficult. Still, the NetFPGA platform is suitable for clean-slate development.

5.1.4 Future work

As a result of ongoing research related to NetFPGA cards, FreeBSD kernel driver and user utilities are being developed. This allows for the usage of the FreeBSD operating system for NetFPGA interaction and programming. Once FreeBSD support is completed, the PSIRP Blackhawk prototype will be modified to utilize NetFPGA for zFilter forwarding.

Hardware implementation for z-Formation, caching and transport protocol support will be considered in the future. The latter may fall beyond the scope of the project.

Currently, the built-in Power PC CPUs on the NetFPGA card are not enabled. Enabling them, with the NetFPGA community's support, would increase NetFPGA's efficiency and performance.

5.2 The BT-Essex network testbed

Experimental testing of the technology solutions developed in PSIRP is crucial for evaluating the viability of the overall proposition of the project, namely to develop a viable alternative to the current IP paradigm. Hence, efforts have been undertaken to establish a campus-scale infrastructure that can be gradually expanded for our testing purposes. The following section outlines this infrastructure and the current status of its build-out.

5.2.1 Infrastructure

The infrastructure is based on a heterogeneous IP infrastructure that was built under the recently finished UK TSB (Technology Strategy Board) funded project HIPnet [HIP2009]. It provides a variety of access technologies in the wireless and wireline domain, e.g., WiMax, WiFi, and all-optical fixed infrastructure. The infrastructure spans the local campus at Essex University, located at the edge of Colchester (UK). The university's facilities hold about 2500 students in their dorms, with access to their infrastructure. The wireless coverage has recently, in June 2009, been extended to full campus coverage with a single SSID. The WiMax coverage spans most of the campus, using a rotating antenna. The physical connectivity on optical level extends to Cambridge University as well as to the BT facilities at Adastral Park (UK). Figure 3 outlines the current facilities.

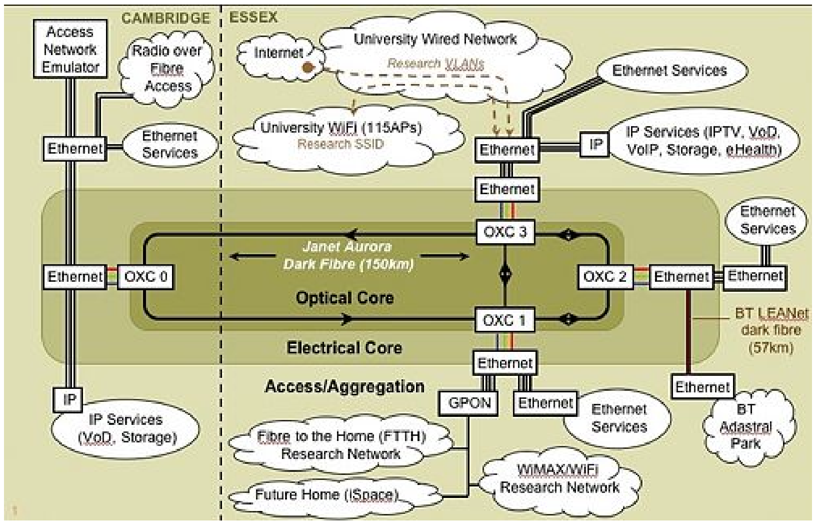

Figure 1: Testbed between BT and Essex University (UK).

As can be seen, various service facilities like storage and video services are available. These could be used for a variety of PSIRP-based services later on.

5.2.2 Current Integration with PSIRP

Efforts are currently underway to extend the current IP facilities towards PSIRP. Given the largely Ethernet basis for the infrastructure, this is relatively easy for the wireline part. Specifically, there are currently three machines being installed for a simple PSIRP network setup. Two of these machines are publisher and subscriber, respectively, currently running the latest release of the PSIRP node architecture (BlackHawk). The third node is a forwarding node, utilizing the current implementation of the Bloom filter approach developed in PSIRP [Jok2009].

Work has started to install a third end node, serving as a second subscriber in a simple scenario. Furthermore, a second forwarding node will be installed soon for demonstrating the intra-domain forwarding capability. The forwarding elements will be co-located with the Ethernet switches at Essex University (see Figure 3). The publisher will reside at Essex University together with one subscriber. The second subscriber will reside in the BT premises at Adastral Park. We expect the test bed and its PSIRP nodes to be gradually expanded towards a set of some 10 nodes within a growing campus infrastructure.

5.2.3 Potential Usage

Apart from the apparent demonstration appeal of such networked setup, serving our dissemination and demonstration to potentially interested stakeholders, the test bed will also serve evaluation purposes. For instance, real network load performance experiments can be conducted for evaluating (a) end node architecture performance and (b) forwarding node performance.

But we also plan to utilize the test bed for extensions on technological level. One such extension is the development of a topology management module (see [Tro2009]) which demonstrates the applicability of the PSIRP information concepts for optimizing resource utilization on optical level. This work will be conducted in partnership with Essex University and is a very good example of the engagement with external partners.

5.3 RWTH testbed for large-scale network emulation

Simulation tools discussed above allow for evaluation of the performance of selected mechanisms, such as rendezvous networks and their interconnection mechanisms, on very large scale. However, there is also a clear need for carrying out large scale tests of the prototype implementations developed in the project. The BT-Essex testbed described in the previous section is one of the platforms enabling this. For tests and evaluation activities calling for a larger number of nodes in a controlled environment we plan to utilize the network emulation testbed installed at RWTH Aachen University.

The testbed consists of a collection of powerful servers equipped with multi-core CPUs and four gigabit Ethernet network interfaces each. All of the servers are connected to a high-performance switch allowing for different network topologies to be set up for the experiments. For scaling up the number of nodes available for the experiments virtualization techniques will be used. Each of the servers is capable of hosting a substantial number (10-20) of virtual machine (VM) instances running the developed prototype implementations. Individual VMs can either host communication endpoints, or nodes normally associated with the network infrastructure such as forwarding nodes, rendezvous servers or topology management nodes.

Experiments with larger numbers of forwarding nodes, including experiments with multiple domains, can be realized by combining the use of VMs with network emulation techniques based on the use of ns-3 as discussed in Section 3.1. In such a setup individual VM instances can emulate complete forwarding infrastructures within individual domains, while other VMs connected to those emulated forwarding domains can act as traffic sources or as nodes offering rendezvous or topology management services.

Towards the end of the project we will evaluate the possibilities to connect some of the testbeds deployed over tunnels running over the existing Internet to further increase the scale available for prototyping and demonstration purposes.

6 Conclusions

In this document we have discussed the simulation and evaluation tools selected and developed for the validation and performance evaluation of the PSIRP architecture. We have seen that the decision to favour well-established open source tools has served the project well, with very few commercial or closed-source tools being necessary. In addition to using and extending existing tools, the development of new ones has also taken place. This has been especially so for socio-economic validation and performance evaluation of the rendezvous network architecture. We strongly believe that the methods and tools developed for these activities are of independent value, and can be modified to serve as a foundation for the evaluation of other large-scale network architecture related proposals as well.

Future work in the project regarding these tools will consist of finalizing those still under development, and using the remainder of the project lifetime to carry out evaluation activities, feeding the results back into the architecture and implementation workpackages. The first round of this evaluation work has already been completed, with results reported in deliverables D4.2 and D4.3. During the second round the scale of the evaluation activities will be extended, and more emphasis will be placed on the validation and evaluation of the developed prototypes.

References

[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.
[Cai2008]	The CAIDA AS Relationships Dataset, August 18th, 2008, see 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.
[Cha2005]	H. Chang, S. Jamin, Z. Mao, and W. Willinger, "An Empirical Approach to Modeling Inter-AS Traffic Matrices", ACM SIGCOMM IMC'05. Proceedings, pages 139-152, 2005.
[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", ACM Internet Measurement Conference, October 2007.
[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", ACM SIGCOMM IMC'07. Proceedings, pages 15-28, 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.
[HIP2009]	HIPnet, http://gow.epsrc.ac.uk/ViewGrant.aspx?GrantRef=EP/E002382/1, 2009
[IAN2009]	IANA, “Autonomous system numbers,” available at http://www.iana.org/assignments/as-numbers.
[Jok2009]	P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, C. Esteve, “LIPSIN: Line speed Publish/Subscribe Inter-Networking”, Proc. of ACM SIGCOMM, 2009.
[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, pages 181-192, October 2007.
[NetFPGA]	NetFPGA. Available at: http://www.netfpga.org/.
[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, pages 217-228, 2008.
[Omn2009]	OMNeT++, "OMNeT++ Simulator ", available at http://www.omnetpp.org, 2009.
[Raj09]	Jarno Rajahalme, Mikko Särelä, Kari Visala, Janne Riihijärvi, "Inter-Domain Rendezvous Service Architecture", Submitted to CoNext'09 on June 20th 2009.
[Rii2009]	J. Riihijärvi (ed), “First report on quantitative and qualitative architecture validation”, PSIRP D4.2, 2009.
[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.
[Ster2000]	J. Sterman, Business Dynamics: Systems Thinking and Modeling for a Complex World, McGraw-Hill, 1008 pages, March 2000.
[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.

[Tro2009] D. Trossen (ed), "Architecture Definition, Components Descriptions and Requirements", PSIRP D2.3, 2009.

[Ven2009] Vensim/Ventana Systems, Inc., available at http://www.vensim.com/, July 2009

[Xmind2009] XMind - Social Brainstorming and Mind Mapping, available at http://www.xmind.net, July 2009.

[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", IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pages 85-98, 2006.