Karlsruhe Institute of Technology (KIT)
Institute of Telematics

TELEMATICS TECHNICAL REPORTS

Realistic Underlays for Overlay Simulation

Ingmar Baumgart, Thomas Gamer,
Christian Hübsch, and Christoph P. Mayer
Institute of Telematics, Karlsruhe Institute of Technology (KIT), Germany
{baumgart,huebsch,mayer}@kit.edu, gamer@tm.uka.de

January, 25th 2011

TM-2011-1

ISSN 1613-849X

`http://doc.tm.uka.de/tr/`

Institute of Telematics
Karlsruhe Institute of Technology (KIT)
Zirkel 2, 76131 Karlsruhe, Germany

Realistic Underlays for Overlay Simulation

Ingmar Baumgart, Thomas Gamer, Christian Hübsch, and Christoph P. Mayer
Institute of Telematics, Karlsruhe Institute of Technology (KIT), Germany
{baumgart,huebsch,mayer,}@kit.edu, gamer@tm.uka.de

***Abstract*—Overlay networks have become an enabler for innovation in today's Internet through cost-efficient and flexible deployment of novel services. The self-organization and scalability properties that peer-to-peer-based overlay networks provide have created real-world large-scale systems like Kad, or Amazon's Dynamo. Due to their distributed behavior, developing and evaluating large-scale overlay networks has a much higher complexity in contrast to centralized systems. To cope with this complexity, simulation has proven indispensable for design and evaluation of overlay networks. Building upon the OMNeT++ simulation environment, the OverSim framework provides widely used simulation of a large and growing set of overlay networks. Realistic environments for evaluation of such networks are crucial to obtain meaningful results, yet complex to develop and validate. The ReaSE topology and traffic generator allows to create Internet-like network topologies, background traffic, and attack traffic. In this work we integrate ReaSE with OverSim, therewith allowing for evaluation of overlay protocols upon realistic underlays and realistic background traffic. This integration provides an important step for design and evaluation of overlay-based systems and allows for meaningful results. We provide insights into runtime and memory consumptions of overlay simulations on the new ReaSE-based underlay on the one hand, and show effects on overlay protocols caused by the realistic underlay on the other hand.**

I. Introduction

Overlay Networks have become an enabler for developing and deploying novel services in today's Internet through the unintrusive and cost-efficient concept of virtual networks. In contrast to services deployed inside the network, overlay networks do not require changes inside the network infrastructure, nor deployment of costly equipment. Rather, overlay networks are deployed on client end-systems—called *peers*—that provide resources for the overlay network, therewith also called *peer-to-peer* (P2P) network. When designed carefully, overlay networks exhibit beneficial properties like scalability, or self-organization which further ease maintenance and deployment. However, there are only few real-world overlay networks deployed in large scale like *Kad* [1], and Amazon's *Dynamo* [2]. Further efforts to bring P2P-based overlay networks into reality are Adobe's Flash P2P technology *Stratus* [3] or Wikipedia's effort to handle the large volume of video data through the BitTorrent-driven and browser-based *Swarmplayer* [4].

To foster real-world deployments of overlay networks, exhaustive evaluation is crucial to understand their distributed behavior in terms of scalability, self-organization, controllability, load, and behavior under failure or attack. Evaluating behavior of overlay protocols in real-world deployments has immense administrative overhead and huge cost and is therefore not economical. Simulation has therefore become the *de facto* approach for overlay protocol design and evaluation. Here, the *Overlay Simulation Framework* (OverSim) [5] provides a strong foundation for the design, evaluation, and comparison of P2P overlay networks through a large number of building blocks and existing protocols (cf. Section II).

To achieve meaningful results, realistic models are required for simulation. OverSim e. g. provides realistic churn models based upon stochastic observations of real-world systems [6]. Underlay latencies are modeled in OverSim based upon data from the CAIDA Skitter project [7][8] which have been gathered through real-world measurements. This so called *SimpleUnderlay* provides great simulation performance due to abstraction from the underlying network topology and paths. However, this underlay model does not obey underlay effects like cross-traffic, influence of AS-level and router-level topology, or router queuing effects that result in jitter.

The project *Realistic Simulation Environments* (ReaSE) [9] provides a simulation model that allows for the generation of underlay topologies and background traffic, based upon characteristics that also have been analyzed through real-world Internet observations [10], [11], [12], [13]. These characteristics are, for instance, a powerlaw-distribution in a topology's node degrees or background traffic showing self-similarity. ReaSE has been developed as a simulation model for OMNeT++ and is based on the protocols implemented by the INET framework. ReaSE provides standalone topology generation through GUI-based tools as well as traffic generation during simulations based on different traffic types and network services.

In this paper we perform an integration of OverSim and ReaSE to allow for meaningful evaluation of overlay networks on realistic underlays. We see this integration of OverSim and ReaSE as an important step towards:

1) Support the community in the design and evaluation of overlay protocols and distributed systems.
2) Increase acceptance of overlay networks in real-world applications through a better understanding of the effects and behavior overlay networks expose in real-world deployments.

This paper is structured as follows: we given an introduction to OverSim and ReaSE in Sections II and III, respectively. The design decisions and actual integration of OverSim and ReaSE are explained in detail in Section IV. Evaluation results with respect to simulation performance and overlay behavior are presented in Section V and compared to results obtained through existing underlays. Guidelines for selection of an

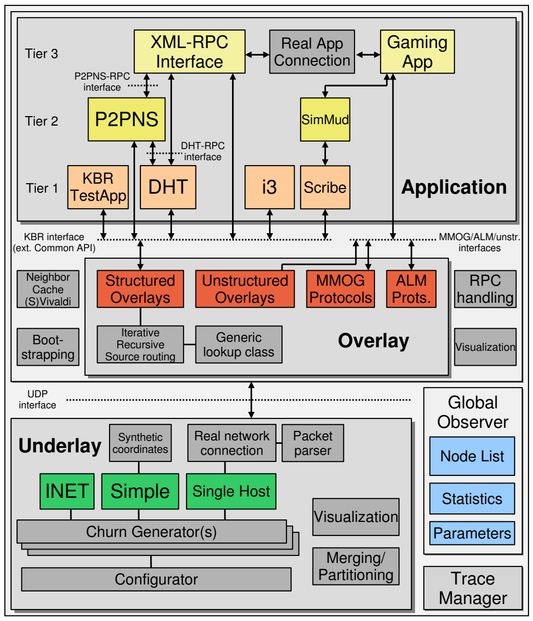

Fig. 1. OverSim's modular architecture

underlay are given in Section VI. We compare our approach in Section VII against related work. Conclusion and future work are given in Section VIII.

II. OverSim

A fundamental problem in studying peer-to-peer networks is the in-depth protocol evaluation, commonly involving both simulation in a large-scale network as well as testing in real networks (e. g. PlanetLab). The *OverSim* [5] simulation framework facilitates these tasks flexibly, being designed to fulfill a number of requirements partially neglected by existing simulation approaches. OverSim comprehensively includes many (un-)structured peer-to-peer protocols, several event distribution protocols and applications based on these protocols. All protocol implementations can be used for both simulation as well as real world networks. Additionally, OverSim provides several common functions for structured peer-to-peer networks to facilitate the implementation of additional protocols and to make them more comparable.

OverSim's architecture shown in Figure 1 allows the modularized modeling of all components in a P2P network in easily exchangeable or extensible manner, thus facilitating code reuse. Several exchangeable underlay network models allow to simulate complex heterogeneous underlay networks as well as simplified networks for large-scale simulations (up to 100 000 nodes have been simulated successfully).

A. Underlay Abstraction

The framework provides different underlay abstraction models differing in complexity and accuracy, being the *SimpleUnderlay*, support of the *INET Framework* as well as the *SingleHostUnderlay*.

The SimpleUnderlay is the default underlay model for OverSim. It combines a low computational overhead with high accuracy, making it a good model for simulating large overlay networks. Nodes are placed into a n-dimensional Euclidean space, determining mutual delays based on their euclidean distance (with n=2 or n=3 depending on the data set). Nodes' positions are chosen to match the measurements from the *CAIDA/Skitter* project. Additionally, each node is assigned to a logical access network characterized by bandwidth, access delay, jitter and packet loss parameters to allow the simulation of heterogeneous access networks. Mobility can be achieved by changing coordinates, access network characteristics and the IP address of a node. To model bandwidth effects, each node contains a logical sending queue. The SimpleUnderlay allows for simulation of underlay network partitioning and merging.

For simulation of heterogeneous access networks, backbone routers and terminal mobility, OverSim provides an underlay model based on the INET framework. Here, the IP stack is completely modeled and even routers can be part of the simulated overlay. INET also contains several MAC protocol implementations, which e. g. allow to model wireless IEEE 802.11 interfaces.

The SingleHostUnderlay provides real network support for OverSim. It acts as a middleware to support deploying overlay protocols developed for OverSim on real networks.

Since all underlay abstraction models share a consistent UDP/IP interface to the overlay protocols, using a different model is fully transparent to the overlay layer.

B. Protocols, Applications & User Models

To facilitate the implementation of overlay protocols in OverSim, several common overlay protocol functions have been identified and integrated into the simulation framework. Examples are overlay message handling (e. g. RPCs), a generic lookup with support for different routing modes, node failure discovery and routing table recovery. Furthermore, the framework offers *Common API* [14] support, bootstrapping support, and proximity awareness (e. g. Vivaldi, GNP).

All of these features allow for rapid overlay protocol prototyping and make protocol implementations comparable and less error-prone.

For overlay applications either a layered or a component-based architecture is feasible (see Figure 1). Applications can use the *Common API* interface, *Application Layer Multicast* interface, or *Virtual World* interface provided by the overlay. Additionally, OverSim makes use of an XML-RPC interface to provide overlay services (e. g. distributed data storage) to external applications similar to the interface provided by the OpenDHT service. User behavior can be scripted by using the *trace manager* which parses scenario or trace files containing application events.

C. Churn Modeling

OverSim provides several models for generating churn, including a lifetime-based churn model supporting different distribution functions (e. g. Weibull, Pareto or Exponential). Alternatively, a scenario or trace file containing join and leave events can be used to model churn behavior. It is possible to use more than one churn generator at the same time to simulate groups of nodes with different churn behaviors. For each churn generator, different node configurations and overlay parameters can be specified, allowing easy generation of complex scenarios with heterogeneous node behavior.

D. Real Network Support

All protocol implementations can be employed without code modifications in real networks. This can be accomplished in two different ways: With the SingleHostUnderlay introduced in Section II-A an OverSim instance emulates a single overlay host, which can be connected to other protocol instances over a real network like the Internet. With the INETUnderlay, in contrast, OverSim can simulate an arbitrary number of overlay hosts. For real network support, OverSim's simulation time can be synchronized with the real time. Using the Linux TUN interface, the hosts in the INETUnderlay can communicate with an external network. This can be used to demonstrate overlay protocols and applications with a limited number of physical devices by connecting them to a large number of emulated overlay nodes.

III. ReaSE

ReaSE, which has been first introduced in [9], features easy and repeatable creation of simulation scenarios with characteristics close to reality in multiple aspects. This, on the one hand, facilitates a meaningful evaluation of Internet-like systems and protocols. On the other hand, this ensures that results of different research activities are comparable due to the usage of equal simulation premises. In the following, the basic characteristics and design decisions of ReaSE are shortly summarized.

With the objective of allowing for a meaningful simulative evaluation of Internet-like systems and protocols we identified three important basic aspects that must be modeled as realistically as possible:

- Internet-like topologies,
- Self-similar background traffic, and
- Large-scale attacks.

Creation of Internet-like topologies is divided into two hierarchical levels, as indicated in Figure 2: First, a topology of *Autonomous Systems* (AS) is generated. In a second step, each AS gets a separate router-level topology. The router-level topology, in turn, is structured hierarchically: it consists of core, gateway, and edge routers as well as actual host systems.

The routing process itself is divided into these two levels, as well. This means, that communication between different AS is only handled by core routers taking part in the inter-domain routing. Every router-level node, on the other hand, takes part in the intra-domain routing of its specific AS. Thus, gateway and edge routers are not aware of routers outside their own AS. Routing within ReaSE currently is done by statically determining all routers' routing tables at simulation startup.

As stated by various publications in recent years, e. g. [15], [10], [11], today's topologies—on AS-level as well as on

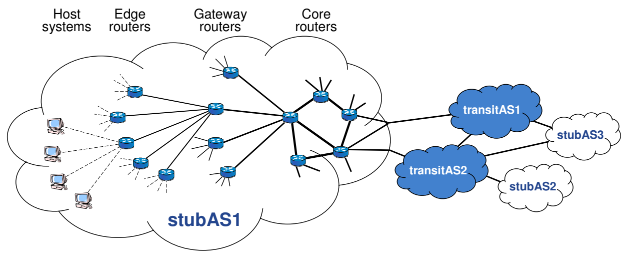

Fig. 2. Hierarchical topologies created by ReaSE

router-level—show, amongst other characteristics, a powerlaw distribution in node degree. Therefore, the topologies generated by ReaSE also show this characteristic, on AS-level as well as on router-level. In case of the router-level additional aspects like market demands, link costs, and hardware constraints have to be considered [11]. Therefore, router-level topologies are modeled as hierarchical topologies as shown in Figure 2 whose link bandwidth increases gradually towards core routers due to traffic aggregation and increasing cost of high-bandwidth routers. The fact that ReaSE-generated hierarchical topologies actually reproduce the mentioned realistic characteristics well has been validated in numerous simulation runs by calculating the three powerlaw values defined by Faloutsos et al. [15] for generated topologies of various different sizes and comparing these values to reference values of real networks. Further details on this validation can be found in [16].

For achieving realistic simulations not only traffic for the specific evaluated protocol or system has to be generated and examined. Indeed, it is necessary to create additional *background traffic* showing the same characteristics as normal traffic in real networks like the Internet—as such traffic usually has a heavy influence on the observed protocol or system. Replay of traffic traces is infeasible in large-scale simulations [17] due to the necessity to record traces at multiple different spots in the Internet, and due to time dependencies and context between the traces. Thus, ReaSE provides means for generation of background traffic *within* the simulation.

Therefore, ReaSE extends the INET framework, which provides basic protocol functionality of the TCP/IP stack for OMNeT++, with special client and server entities for traffic generation. Generation thereby includes various traffic types such as web, streaming, mail, and ping traffic. The generation of background traffic relies on traffic profiles that define the basic behavior of those different traffic types. A traffic profile hereby specifies representative parameters of the corresponding service—like waiting times between consecutive request and reply packets, packet lengths, and number of requests and replies per flow. During a simulation, these specified values serve as configuration for independent random variables based on pareto and normal distributions, respectively. Again, it has been extensively validated by simulation that the resulting traffic generated by the client and server entities features complex characteristics such as self-similarity—as traffic in the Internet does [12]. Furthermore, different publications observed that protocol shares in the Internet remained stable over the last ten years despite various innovations like voice-over-ip or video-on-demand [13]. These protocol shares—about 80 % of the communication relies on the transport protocol TCP, about 20 % on UDP and only about 1 % of the packets observed in the Internet are ICMP error messages—are reproduced realistically by ReaSE-generated traffic, too.

Bearing in mind security-related topics, a further aspect is considered by ReaSE: the generation of *attack traffic* showing characteristics close to reality. ReaSE therefore provides means to generate, for example, *Distributed Denial-of-Service* attacks based on the mechanisms of *Tribe Flood Network* [18]—a notorious DDoS tool used in the wild.

IV. Integration

For integrating OverSim with ReaSE we developed a new underlay model in OverSim that allows to load topologies generated with ReaSE and enables to run background traffic generation. This approach allows to easily generate new topologies and traffic models with ReaSE and requires no adaptations on the side of OverSim. We took special care in a loose coupling of OverSim and ReaSE so that they can be developed independently and are not mutually required, rather OverSim integrates ReaSE as an optional component. This is important as ReaSE employed new modules for implementing the topology, routing, and traffic generation. From the architectural perspective the new underlay for ReaSE-based topologies is similar to OverSim's InetUnderlay.

An implementation view of the new `ReaSEUnderlay` model is given in Table I. The `ReaSEUnderlay` main module does not (in contrast to e. g. the `InetUnderlay`) include routers and channel definitions, but rather the `TerminalConnector` (being defined in the `ConnectReaSE` module described below). Its main purpose is bundling together all required underlay components in a single network module. `ReaSEInfo` extends the class `PeerInfo` and is used mainly by churn generators. It assists in mapping the correct underlay structure to a given node in OverSim's `GlobalNodeList`. The `ReaSEUnderlayConfigurator` also serves churn generators as an interface to add or delete nodes in the network. Here, especially in case of AS topologies, nodes also have to be added to the corresponding submodule. `ConnectReaSE` defines the functions of the `TerminalConnector` which is being used to create overlay nodes in the correct submodule and afterward connect them to an edge router. Therefore, the `TerminalConnector` has to determine an appropriate edge router and create routing table entries considering the added overlay node. Similarly, a node and its routing entries may be deleted. The `ReaSEOverlayHost` module directly describes an overlay participant, while `RUNetworkConfigurator` configures the nodes right before the actual simulation starts. Finally, ReaSE-generated topologies are provided in a specific folder.

Initializing the `TerminalConnector` consists of two steps, the first being determining the module's parameters and setting the corresponding variables, while the second is transferring the topology to appropriate structures. Doing the latter has to consider connecting overlay nodes to edge routers in *Stub Autonomous Systems* (SAS) exclusively. In case the topology has no AS, the routers are considered as one single AS. Also, router modules have to be classified as core routers, gateway routers or edge routers, before shortest paths are calculated to all edge routers employing Dijkstra's algorithm. For adding a new node, a random edge router in the topology is chosen, considering possible constraints (like a maximum number of possible connections or IP address range limitations). After edge router determination, an overlay node is created in the corresponding AS, links are created and routing table entries are added. Deleting an overlay node requires two steps: First, the specific node is deleted from the `GlobalNodeList` and the churn generator. Then, it has to be determined if the node leaves the topology gracefully or not. Finally, the node gets disconnected from its edge router and is removed. Figure 3 shows a screenshot of a running OverSim simulation based on a ReaSE-generated underlay. Multiple transit and stub AS can be seen, being interconnected and abstracted in the view. An internal view of one such single AS—consisting of terminals and router—is shown in Figure 4. Each of the terminals here really takes part in the overlay structure.

The ReaSEUnderlay provides some data structures and mechanisms to increase efficiency in use. To avoid routing entries being created from scratch for every new node added, routing information is being held in an `edgeRoutes` structure for all nodes connected to the same edge router. This allows efficient reuse if applicable. Also, since ReaSE divides the network topology into different AS, every system in an AS and every overly node connected to it obtains an IP Address of the AS's own IP range. Consecutive IP addresses may not be used here since possible node fluctuations would lead to address range consumption. Thus, the ReaSEUnderlay provides the `stubSystem` structure that helps keeping track of addresses currently in use or freely available.

V. Evaluation

For evaluating the ReaSE-based underlay we are interested in two categories: First, performance and memory requirements for the new ReaSE-based underlay which we present in Section V-A. Second, the behavior of overlay protocols when running on the ReaSE-based underlay in comparison to the other OverSim underlays and in comparison to real-world measurements, which we present in Section V-B. Furthermore, we used an adapted version of the GT-ITM [19] topology generator for OverSim based on [20] for generating a comparison underlay. General simulation settings are shown in Table II. The following plots show the average values and 98% confidence intervals of 10 simulation runs with different seeds.

TABLE I
OVERVIEW OF REASEUNDERLAY IMPLEMENTATION

Name	Files	Functionality
`ReaSEUnderlay`	*.ned	Main underlay module that integrates the `ReaSEUnderlayConfigurator`, `ChurnGenerator`, `ConnectReaSE`, and `RUNetworkConfigurator`.
`ReaSEInfo`	*.cc, *.h	ReaSE-specific information attached to an overlay node.
`ReaSEUnderlayConfigurator`	*.cc, *.h, *.ned	Configurator module for the ReaSEUnderlay.
`ConnectReaSE`	*.cc, *.h, *.ned	Connects overlay terminals to the ReaSE edge routers.
`ReaSEOverlayHost`	*.ned	Decription of a host that participates in the overlay.
`RUNetworkConfigurator`	*.cc, *.h, *.ned	Configures the nodes belonging to the topology before starting actual simulation.
`topologies/`	folder, *.ned	Contains ReaSE generated topologies in ned-files.

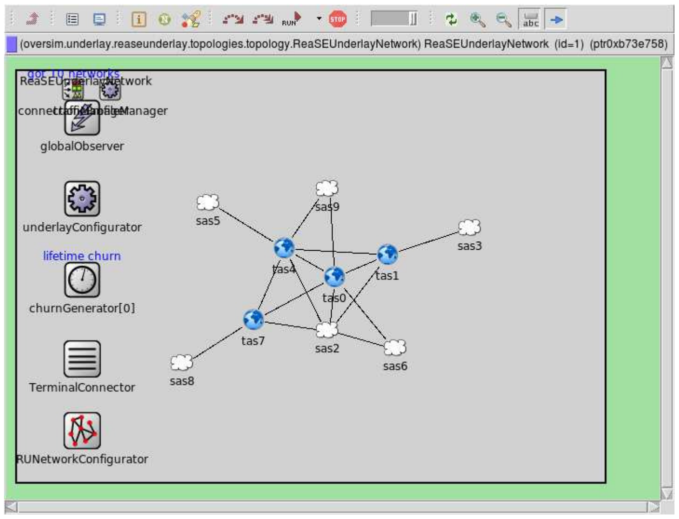

Fig. 3. OverSim with ReaSEUnderlay showing the whole network

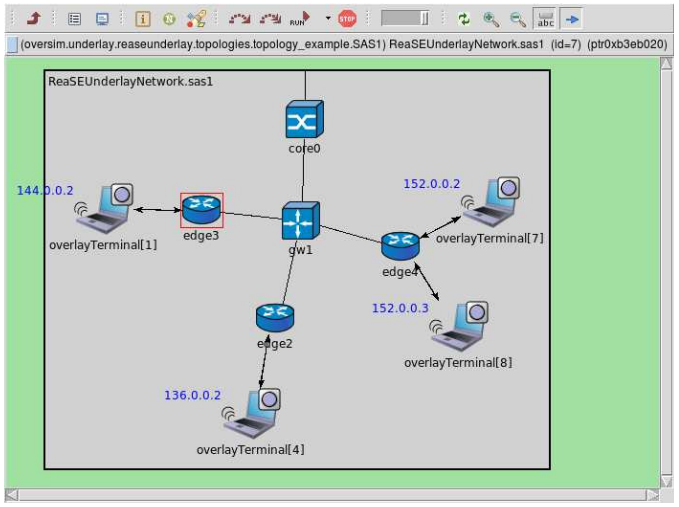
(oversim.underlay.reaseunderlay.topologies.topology_example.SAS1) ReaSEUnderlayNetwork.sas1 (id=7) (ptr0xb3eb020)
ReaSEUnderlayNetwork.sas1
core0
144.0.0.2
152.0.0.2
overlayTerminal[1]
edge3
gw1
overlayTerminal[7]
edge4
152.0.0.3
edge2
overlayTerminal[8]
136.0.0.2
overlayTerminal[4]

Fig. 4. OverSim with ReaSEUnderlay showing a single AS

TABLE II
SIMULATION SETTINGS

Group	Parameter	Value
Environment	CPU	Dual Core 2.8 GHz
	RAM	1 GB
	OS	Ubuntu 9.10
	OMNeT++	Version 4.1
	OverSim	Version 20101113
	ReaSE	Version 1.23
General	Number of nodes	500 and 1000
	Simulation time	3 hours
Chord	Routing mode	Iterative
	Successor list	8
	Stabilize interval	20 s
	Fixfinger interval	120 s
sVivaldi	Dimensions	2
	Height vector	no
ReaSEUnderlay	server2edge link	5 ms, 100 Mbps
	host2edge link	5 ms, 2 Mbps
	edge2host link	5 ms, 16 Mbps
	edge2gateway link	1 ms, 100 Mbps
	gateway2core link	1 ms, 1000 Mbps
	core2core link	1 ms, 1000 Mbps

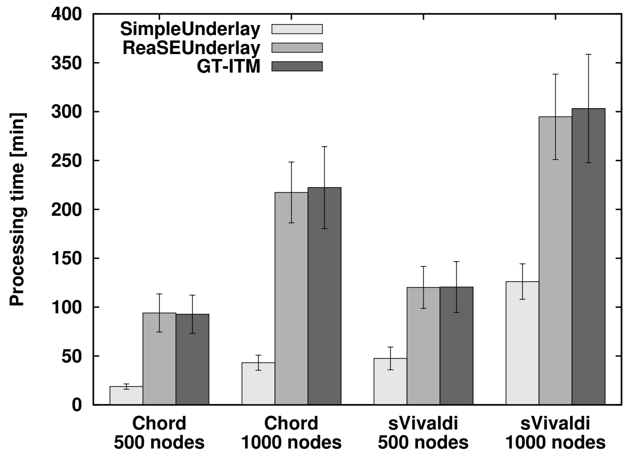

Fig. 5. Processing time depending on underlay model

A. Simulation performance

For evaluating large-scale overlay networks the simulation duration is of interest as it is the limiting factor for overlay size and number of seeds that can be generated with a given scenario. We choose the well-known Chord [21] protocol for our comparisons. In one setup we run Chord alone with 500 overlay nodes, and with 1 000 overlay nodes. Then, we additionally employ the sVivaldi [22] Internet coordinate system with Chord. Figure 5 shows an overview of simulation duration for the different scenarios for three different underlays SimpleUnderlay, ReaSEUnderlay, and GT-ITM-based underlay. The SimpleUnderlay generally provides the best performance and fastest simulations as it does not employ complex routing or intermediate systems. The GT-ITM-based underlay and ReaSEUnderlay have a comparable simulation duration which is several orders of magnitude higher than the SimpleUnderlay. An interesting effect is the SimpleUnderlay's performance decrease and resulting simulation time increase when employing sVivaldi. While the SimpleUnderlay's simulation time more than doubles, GT-ITM and ReaSEUnderlay have a much smaller relative increase in simulation time. This is due to the fact that protocols such as sVivaldi pose a constant overhead that is independent of the underlay. In the case of sVivaldi and Chord, the constant overhead of sVivaldi outweighs the more underlay-specific overhead of Chord.

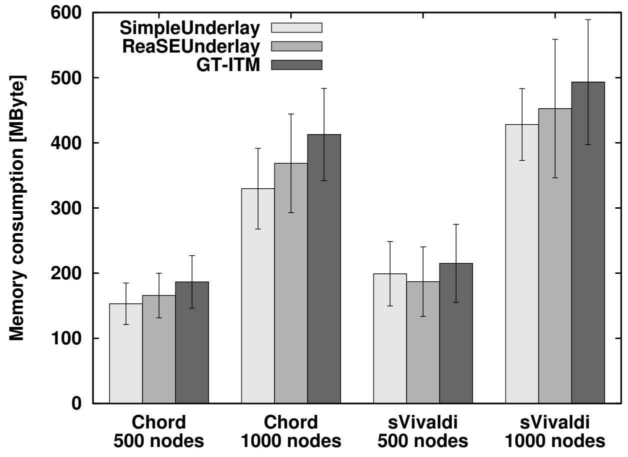

Fig. 6. Memory consumption depending on underlay model

Besides simulation runtime, the memory requirements of the simulation pose a limiting factor in size of the overlay simulation due to hardware constraints. We again use the same set of scenarios as in the prior runtime evaluation. Figure 6 shows the memory requirements for the three underlays SimpleUnderlay, ReaSEUnderlay, and GT-ITM-based underlay. Generally, the SimpleUnderlay provides the smallest memory requirements, ReaSEUnderlay the second smallest memory, and GT-ITM-based underlay the largest memory requirements. The differences are, however, not so extreme so that we argue that in terms of memory the realistic properties gained with the ReaSEUnderlay outweigh the higher requirement in terms of memory.

B. Overlay behavior

To evaluate how overlay protocols behave differently depending on the employed underlay, we use the sVivaldi latency estimation mechanisms. We are especially interested in whether distributed protocols exhibit a behavior similar to real-world deployments. The sVivaldi latency estimation system provides a good way to evaluate the behavior, especially as there exist real-world measurements of the Vivaldi system on the PlanetLab testbed [23]. Figure 7 shows the relative error of the sVivaldi simulation on the three underlays, and results from Vivaldi real-world measurements from [23]. The SimpleUnderlay results in an unrealistic behavior as latencies are modeled through an ideal 2D coordinate system.

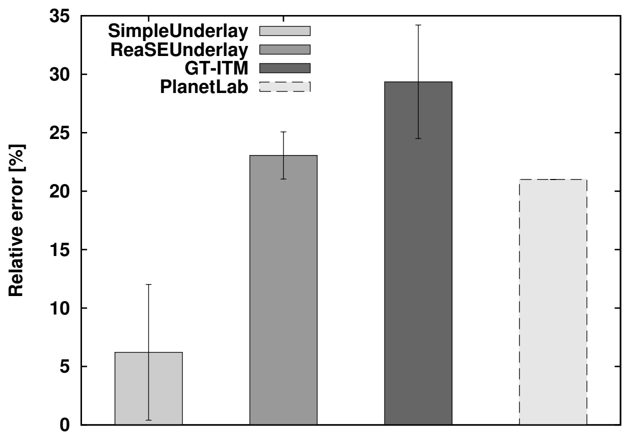

Fig. 7. Relative error of sVivaldi's latency estimation. PlanetLab measurement from [23]

ReaSEUnderlay provides the best estimation error in terms of similarity to the real-world estimation error reported in [23] from PlanetLab.

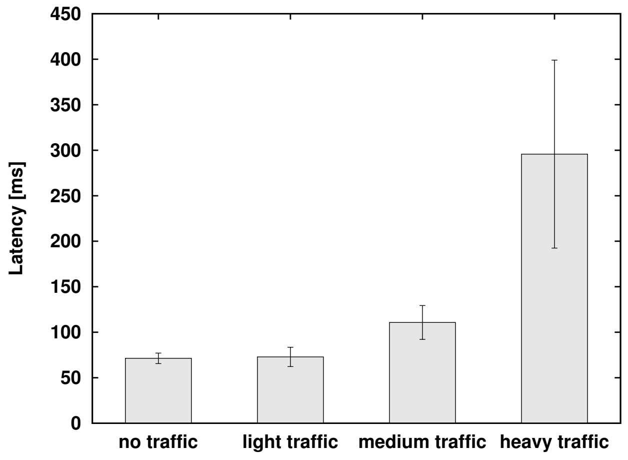

Fig. 8. Effects of background traffic on overlay routing latencies

Besides generating real-world topologies, ReaSE allows generation of self-similar background traffic at simulation runtime. We use this traffic generation feature to evaluate the effect of overlay *Key-based Routing* (KBR) latencies with different traffic profiles. Figure 8 shows Chord running on the ReaSEUnderlay with different intensities of ReaSE-generated background traffic. It can be seen that such background traffic results in non-negligible impact on the overlay routing behavior. This is due to router queue effects that result in network congestion in the routers. Therewith the ReaSEUnderlay can be used to evaluate overlay protocols with additional background traffic to get an insight into the overlay protocol stability and resilience. While we just present initial results to show that the new ReaSE-based underlay allows to evaluate overlay protocols under the influence of background traffic, an in-depth evaluation of overlay protocol behavior is future work.

VI. Choosing an underlay

Based on our evaluations and experience with the different underlays in OverSim we give guidelines on selecting the correct underlay based on requirements for the simulation.

a) SimpleUnderlay: The SimpleUnderlay provides the best performance in terms of memory consumption and simulation run-time. It is therefore best suited if a very large simulation is aimed for. Using the SimpleUnderlay OverSim allows to run simulations with 10 000 nodes on commodity hardware in real-time. Large-scale scenarios of 100 000 nodes have been successfully simulated on specialized hardware. Typical Internet latencies based on CAIDA/Skitter[7][8] data allow good approximation of real-world behavior in simple settings.

b) InetUnderlay: The InetUnderlay aims at use of custom topologies that can be built with heterogeneous protocols, e. g. for support of wireless access. Such underlays are on the one hand built through a simplified topology algorithm inside OverSim and can be easily adapted to run overlays over custom topologies and networks.

c) ReaSEUnderlay: The ReaSEUnderlay presented in this paper allows to run overlay networks on realistic topologies that can be easily generated through ReaSE tools and expose topological properties found in today's networks. As such they pose more complex scenarios but exhibit latencies and jitter that result from router queuing effects like found in the real world. Specifically we found that Internet coordinate systems like sVivaldi pose performance similar to PlanetLab measurement when run on the ReaSEUnderlay. The ReaSEUnderlay allows for scaling of the underlay and overlay separately, and has different link latencies on different links in the topology. It therefore can be used to perform P2P traffic engineering, and focus on link stress, e. g. in *Application Layer Multicast* (ALM) protocols.

VII. Related Work

Different topology and traffic generators exist that could be integrated with the OverSim simulation environment. We briefly review the topic of topology and traffic generation and argue for our decision to integrate ReaSE with OverSim.

d) Topology Generation: Early topology generators like the Waxman model [24] are based on random graphs that do not take real-world properties of Internet topologies into account. More advanced generators like GT-ITM [19]—which is used for comparison in our evaluations—or TIERS [25] perform a first step in taking structural properties of Internet topologies into account. Based on the findings of Faloutsos et al. [15] that Internet topologies exhibit a power-law distribution, newer generators like Inet-3.0 [26], or BA [27] have been developed. They, however, can only generate AS-level topologies and do not take special properties of the router-level topology into account. Another well-known topology generator is BRITE [28] that provides a large set of topology generation functionality. BRITE, however, lacks traffic generation and is no longer developed.

e) Traffic Generation: An important property of Internet traffic is its self-similarity [12]. Traffic generators that focus on simulation environments and produce suitable traffic mixes are e. g. BonnTraffic [29] and TrafGen [30]. Existing traffic generators, however, require exhaustive configurations for all end-systems taking part in the traffic generation. This makes them unsuitable for creation of large-scale topologies. Furthermore, they mostly focus on a single application that generates traffic, we however require a realistic mix of Internet background traffic.

Our choice for selecting ReaSE to be integrated with OverSim is threefold:

- First, ReaSE is cleanly integrated with the OMNeT++ environment.
- Second, ReaSE is based on state-of-the-art algorithms for topology and traffic generation.
- Third, ReaSE integrates multiple functionality that otherwise would require integration of multiple tools (topology generation, background traffic generation, attack traffic generation).

VIII. Conclusion

Overlays present an important technology for implementing and testing of novel services and application in the Internet. To ease evaluation of distributed overlay protocols, simulation has proven to be of major importance. For providing realistic simulation environments, in this paper we integrated the ReaSE topology and traffic generator with the well-established OverSim overlay simulation framework. While simulation time is a critical factor where the new underlay shows strong negative impact, memory requirements that pose actual hardware constraints increase only slightly with our newly developed ReaSE-based underlay. We have shown that distributed systems like sVivaldi expose more realistic behavior on the new underlay. Further, background traffic is an important characteristic of real-world networks that has been neglected in simulations. The integration of ReaSE with OverSim finally provides means on the way towards realistic overlay simulations. We provide the new ReaSEUnderlay in the latest OverSim release for the research community as open source.

There are open issues, especially in dimensioning of the underlay network, and in modeling of link latencies on the different topology hierarchies that we aim to tackle in future work. Furthermore, the evaluation of overlay protocols in face of background traffic is an important topic that must be analyzed in more detail.

ReaSE: http://www.tm.kit.edu/rease
OverSim: http://www.oversim.org

Acknowledgments

The authors would like to thank Bernhard Müller for his valueable contribution in initial implementation and evalua-

tion.

References

[1] M. Steiner, T. En-Najjary, and E. W. Biersack, "Long Term Study of Peer Behavior in the KAD DHT," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1371–1384, Oct. 2009.

[2] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's Highly Available Key-value Store," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205–220, Dec. 2007.

[3] Adobe Systems Incorporated, "Stratus," http://labs.adobe.com/technologies/stratus/, Sep. 2010.

[4] P2P-Next, "Swarmplayer," http://swarmplayer.p2p-next.org, Sep. 2010.

[5] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proceedings of IEEE Global Internet Symposium (in conjunction with IEEE INFOCOM)*, Anchorage, AK, USA, May 2007, pp. 79–84.

[6] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," in *Proceedings ACM SIGCOMM IMC Workshop*, New York, NY, USA, Oct. 2006, pp. 189–202.

[7] B. Huffak, D. Plummer, Daniel, D. Moore, and K. Claffy, "Topology Discovery by Active Probing," in *Proceedings of Symposium on Applications and the Internet Workshops (SAINT-W)*, Washington, DC, USA, Jan. 2002, pp. 90–96.

[8] P. Mahadevan, D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitropoulos, K. Claffy, and A. Vahdat, "Lessons from Three Views of the Internet Topology," Cooperative Association for Internet Data Analysis (CAIDA), University of California, San Diego, CA, USA, Technical Report tr-2005-02, Aug. 2005.

[9] T. Gamer and M. Scharf, "Realistic Simulation Environments for IP-based Networks," in *Proceedings of International Workshop on OMNeT++(in conjunction with SIMUTools)*, Marseille, France, Mar. 2008, pp. 83:1–83:7.

[10] S. Zhoua, G. Zhang, G. Zhang, and Z. Zhuge, "Towards a Precise and Complete Internet Topology Generator," in *Proceedings of International Conference on Communications, Circuits and Systems (ICCCAS)*, Guilin, China, Jun. 2006, pp. 1830–1834.

[11] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A First-Principles Approach to Understanding the Internet's Router-level Topology," in *Proceedings of ACM SIGCOMM*, Portland, Oregon, USA, Sep. 2004, pp. 3–14.

[12] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and Possible Causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, Dec. 1997.

[13] W. John, S. Tafvelin, and T. Olovsson, "Trends and Differences in Connection-Behavior within Classes of Internet Backbone Traffic," in *Proceedings of International Conference on Passive and Active Network Measurement (PAM)*, Cleveland, OH, USA, Apr. 2008, pp. 192–201.

[14] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, USA, Feb. 2003, pp. 33–44.

[15] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-law Relationships of the Internet Topology," *Computer Communication Review*, vol. 29, no. 4, pp. 251–262, Oct. 1999.

[16] T. Gamer and C. P. Mayer, "Simulative Evaluation of Distributed Attack Detection," *Simulation: Transactions of the Society for Modeling and Simulation International*, May 2011, to appear.

[17] V. Paxson and S. Floyd, "Why we don't know how to simulate the Internet," in *Proceedings of Conference on Winter Simulation (WSC)*, Atlanta, GA, USA, Dec. 1997, pp. 1037–1044.

[18] D. Dittrich, "The Tribe Flood Network Distributed Denial of Service Attack Tool," Oct. 1999, http://staff.washington.edu/dittrich/misc/tfn.analysis.

[19] K. L. C. u. S. B. Zegura, Ellen W., "How to Model an Internetwork," in *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, Mar. 1996, pp. 594–602.

[20] K. Katsaros, N. Bartsotas, and G. Xylomenos, "Router Assisted Overlay Multicast," in *Proceedings of Euro-NGI Conference on Next Generation Internet networks (NGI)*, Aveiro, Portugal, Jul. 2009, pp. 106–113.

[21] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, F. M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[22] C. de Launois, S. Uhlig, and O. Bonaventure, "Scalable Route Selection for IPv6 Multihomed Sites," in *Proceedings of NETWORKING*, Waterloo, ON, Canada, May 2005, pp. 1357–1361.

[23] J. Ledlie, P. Gardner, and M. I. Seltzer, "Network Coordinates in the Wild," in *Proceedings of Symposium on Networked Systems Design and Implementation (NSDI)*, Cambridge, MA, USA, Apr. 2007, pp. 299–311.

[24] B. M. Waxman, "Routing of Multipoint Connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[25] M. B. Doar, "A Better Model for Generating Test Networks," in *Proceedings of of Global Telecommunications Conference (GLOBECOM)*, London, UK, Nov. 1996, pp. 86–93.

[26] J. Winick and S. Jamin, "Inet-3.0: Internet Topology Generator," University of Michigan, Technical Report UM-CSE-TR-456-02, Jul. 2002.

[27] S.-H. Yook, H. Jeong, and A.-L. Barabasi, "Modeling the Internets Large-scale Topology," *PNAS*, vol. 99, no. 21, pp. 13 382–13 386, Oct. 2002.

[28] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," in *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)*, Cincinnati, OH, USA, Aug. 2001, pp. 346–354.

[29] B. Roemer, "BonnTraffic: A Modular Framework for Generating Synthetic Traffic for Network Simulations," http://web.informatik.uni-bonn.de/IV/BoMoNet/BonnTraffic.htm, Nov. 2005.

[30] I. Dietrich, "OMNeT++ Traffic Generator," http://www7.informatik.uni-erlangen.de/isabel/omnet/modules/TrafGen/, Sep. 2006.