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. 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.

1. 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.

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 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) [2] provides a strong foundation for the design, evaluation, and comparison of P2P overlay networks through a large number of building blocks and existing protocols. 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. Underlay latencies are modeled in OverSim based upon data from the CAIDA Skitter project [6][11] which have been gathered through real-world measurements. This *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 or packet drops.

The project *Realistic Simulation Environments* (ReaSE) [5] provides a simulation model that allows for generation of underlay topologies and background traffic, based upon characteristics that have been analyzed through real-world Internet observations [14, 10, 3, 7]. 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 the OMNeT++ simulation core 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 supporting the community in the design and evaluation of overlay protocols and distributed systems as well as increasing 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: Design decisions and actual integration of OverSim and ReaSE are explained in Section 2. Evaluation results with respect to simulation performance and overlay behavior are presented in Section 3

OMNeT++ 2011, March 21, Barcelona, Spain

DOI 10.4108/icst.simutools.2011.245526

and compared to results obtained through existing underlays. Guidelines for selection of an underlay are given in Section 4, followed by our conclusion and future work. An extended version of this paper is available as technical report [1].

2. 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.

An implementation view of the new `ReaSEUnderlay` model is given in Table 1. 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 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 determines 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.

3. 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 3.1. 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 3.2. Furthermore, we used an adapted version of the GT-ITM [13] topology generator for OverSim based on [8] for generating a comparison underlay. General simulation settings are given in Table 2. The following plots show average values and 98% confidence intervals of 10 simulation runs with statistically independent seeds.

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

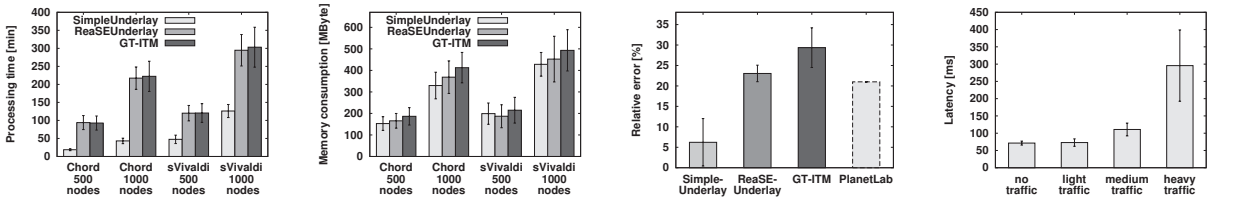
Table 2: Simulation settings

3.1 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 [12] 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 [4] Internet coordinate system with Chord. Figure 1(a) 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,

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.

Table 1: Overview of ReaSEUnderlay implementation



(a) Processing time (b) Memory consumption (c) Relative error of sVivaldi's latency estimation (d) Effects of background traffic on overlay routing latencies

Figure 1: Simulation Results: Performance of different underlays in OverSim

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.

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 1(b) shows the respective 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.

3.2 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 [9]. Figure 1(c) shows the relative error of the sVivaldi simulation on the three underlays, and results from Vivaldi real-world measurements from [9]. The SimpleUnderlay results in an unrealistic behavior as latencies are modeled through an ideal 2D coordinate system. ReaSEUnderlay provides the best estimation error in terms of similarity to the real-world estimation error reported in [9] from PlanetLab.

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 1(d) 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.

4. 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.

The SimpleUnderlay shows 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[6][11] data allow good approximation of real-world behavior in simple settings.

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.

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.

5. 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. OverSim can be found at `http://www.oversim.org/`, ReaSE at `http://www.tm.kit.edu/rease/`, and the extended version of this paper in [1].

Acknowledgments

Ingmar Baumgart is supported by the *Concept of the Future* of KIT within the framework of the German Excellence Initiative. Christian Hübsch and Christoph P. Mayer are supported by the Landesstiftung Baden Württemberg gGmbH within the BW-FIT project SpoVNet. The authors thank Bernhard Müller for his valueable contribution in initial implementation and evaluation.

6. REFERENCES

[1] I. Baumgart, T. Gamer, C. Hübsch, and C. P. Mayer. Realistic Underlays for Overlay Simulation. Telematics Technical Report (ISSN 1613-849X) TM-2011-1, Institute of Telematics, Karlsruhe Institute of Technology (KIT), Germany, Jan. 2011. http://doc.tm.uka.de/tr/TM-2011-1.pdf.

[2] 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)*, pages 79–84, Anchorage, AK, USA, May 2007.

[3] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, Dec. 1997.

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

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

[6] 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)*, pages 90–96, Washington, DC, USA, Jan. 2002.

[7] 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)*, pages 192–201, Cleveland, OH, USA, Apr. 2008.

[8] K. Katsaros, N. Bartsotas, and G. Xylomenos. Router Assisted Overlay Multicast. In *Proceedings of Euro-NGI Conference on Next Generation Internet networks (NGI)*, pages 106–113, Aveiro, Portugal, July 2009.

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

[10] 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*, pages 3–14, Portland, Oregon, USA, Sept. 2004.

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

[12] 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*, 11(1):17–32, Feb. 2003.

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

[14] 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)*, pages 1830–1834, Guilin, China, June 2006.