Analyzing and Implementing Peer-to-Peer Systems with the PeerSE Experiment Environment

Ludger Bischofs
OFFIS
Institute for Information Technology
Oldenburg, Germany
Email: ludger.bischofs@offis.de

Wilhelm Hasselbring
Software Engineering Group
University of Kiel
Kiel, Germany
Email: wha@informatik.uni-kiel.de

***Abstract*—This paper introduces the *PeerSE Experiment Environment (EE)* including concepts for a simulation-based analysis and implementation of P2P systems. The *PeerSE EE* supports a comparative analysis of P2P system models to identify models fulfilling given requirements. In a next step, model components are reused for implementing laboratory and real-world P2P systems. The integrated visualization for simulation and laboratory experiment results helps developers to compare the behavior of simulation models and laboratory systems. Thus, the *PeerSE EE* allows a controlled transition of model components to laboratory components. The applicability of the *PeerSE EE* is demonstrated by exemplarily analyzing and implementing a P2P system.**

I. Introduction

Peer-to-Peer (P2P) systems are distributed systems composed of up to millions of functionally equivalent entities (peers), which form P2P overlay networks on top of physical networks to communicate. The functionality of a peer is implemented by a P2P application which defines the behavior of the whole P2P system. The equivalence of peers is realized by providing client functionality as well as server functionality.

Implementing a P2P system with specified behavior is a difficult task because the behavior depends on many factors, such as the used P2P search methods and the underlying physical network. Simulation can help to estimate the system behavior in an early development phase based on system models.

The main contribution of this paper is the *PeerSE EE* including concepts for a controlled transition from P2P simulation models to laboratory and real-world P2P software systems. "Controlled" means, that the behavior of P2P system models and laboratory systems are compared based on the same metrics. Further advantages of the *PeerSE EE* allow a comparative analysis of P2P system models and the reuse of model components for implementing laboratory and real-world P2P systems.

This paper is organized as follows. Section II describes the main features of the *PeerSE EE*. In Section III, the simulation model of the *PeerSE EE* is specified. The applicability of the *PeerSE EE* is demonstrated in Section IV by exemplarily developing a P2P system. After presenting related work in Section V we conclude and outline future work.

II. PeerSE Experiment Environment

The *PeerSE EE* is an open source tool (downloadable from [1]) for developing P2P systems in Java. P2P system models can be analyzed with the integrated discrete event system simulator which implements the *PeerSE Simulation Model* described in Section III. The *PeerSE EE* supports the reuse of model components for performing laboratory experiments. In laboratory experiments, P2P applications communicate over a laboratory physical network in real time (instead of model time in simulation experiments) and are controlled by user models. Laboratory P2P applications can be reused for implementing real-world P2P applications. In contrast to laboratory P2P applications, real-world P2P applications are controlled by real-world users, so that no model components are used anymore.

Special graphs (so called *PeerSE Graphs*) are used for initializing *PeerSE Simulation Models* and for visualizing the initial system state of a P2P system model within the *PeerSE EE*. The visualization of state changes is also possible, but has not been integrated yet.

The *PeerSE EE* visualizes simulation and laboratory experiment results based on the same metrics as charts. This helps developers to compare the behavior of simulation models and laboratory systems. The comparison of simulation and laboratory experiment results allows a controlled transition of model components to laboratory components. In simulation experiments, results can be visualized already at experiment runtime to allow fast testing and debugging during the model development.

III. PeerSE Simulation Model

The *PeerSE Simulation Model* contains components for modeling and simulating P2P systems (see Figure 1). In the *PeerSE Simulation Model*, a P2P system is just an aggregation of P2P applications, which communicate over a physical network, often using an additional P2P overlay network on top of it. Each P2P application is typically controlled by a user, but it may also act stand-alone, e.g. for building and managing P2P overlay networks.

The *PeerSE Simulation Model* is initialized by means of a special graph called *PeerSE Graph*. A *PeerSE Graph* is a mixed pseudo graph, that may contain directed and undirected

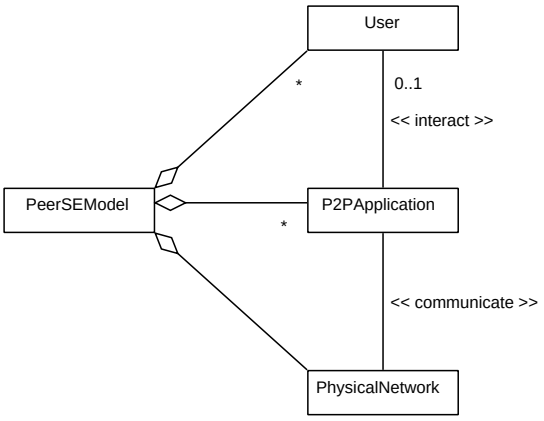

Fig. 1. Model components of the *PeerSE Simulation Model*

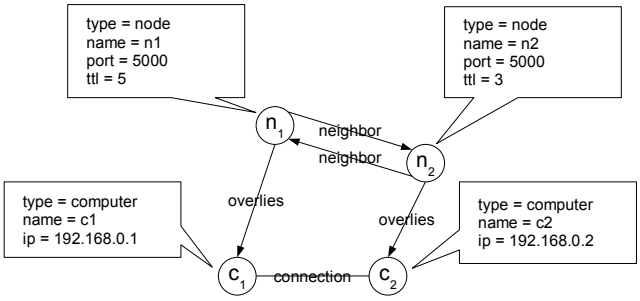

Fig. 2. Visualization example of a *PeerSE Graph*

edges, loops and parallel edges (see [2]). Additional initialization information like node or edge labels are represented by attributes. Figure 2 exemplarily depicts a *PeerSE Graph* containing the computers c_1 and c_2, that are connected with an undirected edge. The nodes n_1 and n_2 represent nodes of a P2P overlay network. Attributes of the nodes and edges are visualized as node/edge labels and in special boxes. The shown example is only one possibility for visualizing a *PeerSE Graph* and its attributes. There may be better solutions depending on the graph topology and attribute structure. In the following, we specifiy a *PeerSE Graph* in Object-Z [3].

We first define the set of attributes as a base type because its structure is not specified in detail at this point. An attribute of a node or an edge can for instance be just a number or perhaps a complex data structure (e.g. an XML document).

$[ATTRIBUTE]$

We call a node of a *PeerSE Graph PeerSEVertex*. A *PeerSEVertex* contains a finite set of attributes and is specified as an Object-Z schema:

PeerSEVertex

$attributes : \mathbb{F}\, ATTRIBUTE$

Directed edges are called *PeerSEArc* and contain the nodes *from* and *to*, which are connected via a directed edge. *PeerSEArc* also contains a finite set of attributes of base type *ATTRIBUTE*.

PeerSEArc

$from, to : PeerSEVertex$

$attributes : \mathbb{F}\, ATTRIBUTE$

Undirected edges are specified as *PeerSEEdge* and include two nodes $v1$ and $v2$. They also contain a finite set of attributes of base type *ATTRIBUTE*.

PeerSEEdge

$v1, v2 : PeerSEVertex$

$attributes : \mathbb{F}\, ATTRIBUTE$

A *PeerSE Graph* consists of a finite set *vertices* of nodes, a finite set *arcs* of directed edges and a finite set *edges* of undirected edges. A *PeerSE Graph* is specified as an Object-Z class as follows:

PeerSEGraph

$vertices : \mathbb{F}\, PeerSEVertex$

$arcs : \mathbb{F}\, PeerSEArc$

$edges : \mathbb{F}\, PeerSEEdge$

$\forall a : arcs \bullet (a.from \in vertices)$

$\forall a : arcs \bullet (a.to \in vertices)$

$\forall e : edges \bullet (e.v1 \in vertices)$

$\forall e : edges \bullet (e.v2 \in vertices)$

INIT

$vertices = \varnothing$

$arcs = \varnothing$

$edges = \varnothing$

AddPeerSEGraph

$\Delta(vertices, arcs, edges)$

$g? : PeerSEGraph$

$vertices' = vertices \cup g?.vertices$

$arcs' = arcs \cup g?.arcs$

$edges' = edges \cup g?.edges$

PeerSE Graphs can be composed of *PeerSE Graphs*. Thus, it is possible to generate a physical and overlay network graph separately to join these graphs in a further step. The procedure *AddPeerSEGraph* of the class *PeerSEGraph* specifies such an operation.

Figure 3 shows an example of a *PeerSE Graph* composed of a physical network and a P2P overlay network. The concept of *PeerSE Graphs* composition is supported by the *PeerSE EE* for creating the initial *PeerSE Graph* needed for initializing the *PeerSE Simulation Model*. This allows an independent development of *PeerSE Graph* generators for several types of physical and overlay networks. Overlay networks can be mapped on physical networks in a further step, what allows a high flexibility for creating initial *PeerSE Graphs*.

IV. Evaluation of the PeerSE Experiment Environment

The applicability of the *PeerSE EE* is shown in this Section by describing the analysis and implementation of an exemplary P2P system for distributed software development.

A tutorial experiment within the course software system engineering at the University of Oldenburg in winter semester

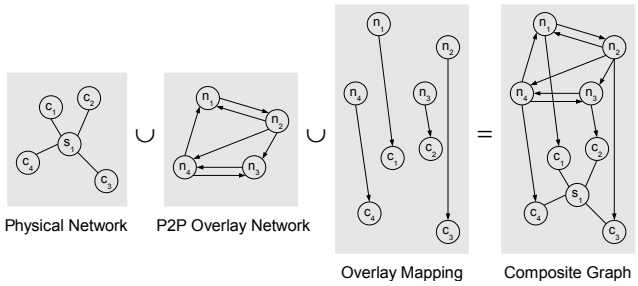

Fig. 3. Composition of a *PeerSE Graph* containing a physical network and an overlay network

2005 already indicated the applicability. The computer science students were able to develop and analyze P2P search methods in Java using the eaSim simulator [4], a previous prototype of the *PeerSE EE*. eaSim does not offer functionality for further developing the simulation model to a laboratory P2P system, wherefore the implemented search method have not been executed in a laboratory experiment setting. The further development of P2P system models to real-world P2P systems and the combined execution of model and real-world components have been demonstrated with the RealPeer framework [5] within the scope of this work.

In this Section, we show the applicability of the *PeerSE EE* by modeling and implementing a P2P system. We focus on the controlled transition of a P2P application model component to a laboratory P2P application. Accordingly, we take a look at the reuse of model components and the comparison of simulation and laboratory experiment results.

We assume that a P2P system shall be implemented which supports the self-organization of FLOSS (Free/Libre and Open Source Software) developers into projects. To this end, a method for searching for developers and projects is required. Additional requirements are that the P2P system must be resource efficient and that search queries must be answered as fast as possible.

A. Simulation Experiments

We use simulation experiments for a comparative analysis of different P2P application model components to find the one which best fulfills the requirements. The model components *User*, *P2PApplication* and *PhysicalNetwork* are required for executing simulation experiments. The *PeerSE EE* also requires an experiment configuration file, *PeerSE Graph* generators and event list generators for initializing the simulation experiments (see Figure 4).

In our case, four *PeerSE Graph* generators have been developed. The first one generates a simple physical network graph used for initializing the physical network model consisting of connected computers. The second one generates a FLOSS overlay network graph [6], the third one a pure unstructured P2P overlay network graph, and the fourth one a Chord overlay network graph (Chord ring) [7]. Simulation experiments have been designed and executed to compare

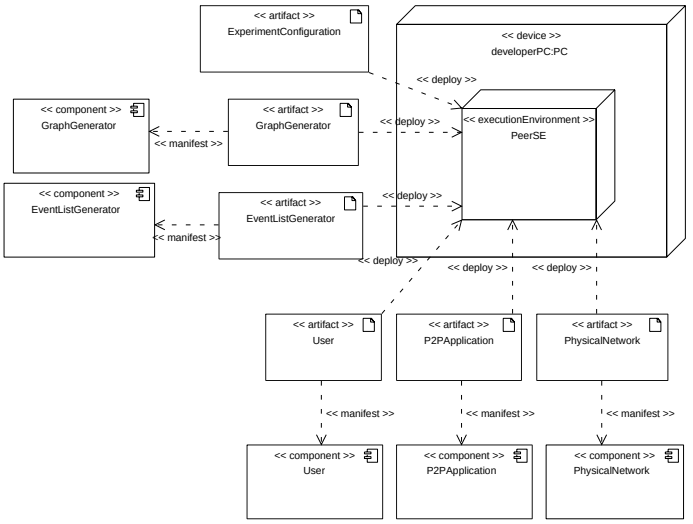

Fig. 4. Deployment diagram of a simulation experiment

experiment name	experiment duration	experiment runs	physical network	FLOSS overlay network (FLOSS)	P2P overlay network (P2P)	Chord overlay network (Chord)	P2P search method	searched overlay network	number of computers	number of search queries	node state changes (offline/online)	max. event time
FLOSS_P2P_40	60000 ms	10	●	●	●	○	Breadth-First-Search	P2P	40	100	-	59000 ms
FLOSS_Chord_40	60000 ms	10	●	●	○	●	Chord	Chord	40	100	-	59000 ms
FLOSS_P2P_40_Dynamic	60000 ms	10	●	●	●	○	Breadth-First-Search	P2P	40	100	8	59000 ms
FLOSS_Chord_40_Dynamic	60000 ms	10	●	●	○	●	Chord	Chord	40	100	8	59000 ms

Abbreviations: ● = yes, ○ = no

Fig. 5. Experiment settings with 40 computers

different P2P architectures generated with these *PeerSE Graph* generators.

Figure 5 lists the experiment settings described in this paper. Two different kinds of *PeerSE Graphs* are generated. The first one consists of a physical network, a pure unstructured P2P overlay network, and a FLOSS overlay network, and the second one contains a Chord overlay network instead of a pure unstructured P2P overlay network. Breadth-first-search is used for searching within the pure unstructured P2P overlay network and Chord is used for searching within the Chord overlay network. There are ten experiment runs for each experiment with a duration of 60000 ms. Events are processed within the first 59000 ms of the experiment runs, so that answers have time to arrive within the last second.

Several metrics have been used to compare the P2P system models. The simulation experiment results show, that search queries can be answered comparable fast in both systems. Chord is a much more efficient search strategy than breadth-first search, because it needs much fewer messages to answer queries. This is the reason, why we chose the Chord-based architecture as candidate for the controlled transition into a laboratory system.

B. Laboratory Experiments

Figure 6 explains the concept of executing laboratory experiments in a distributed environment on a physical network.

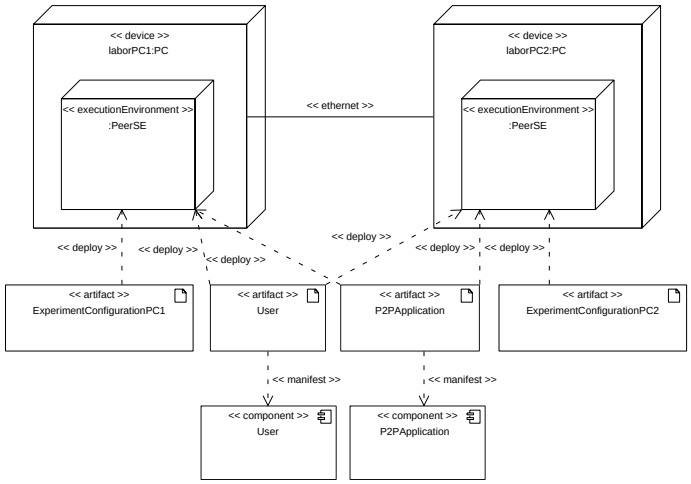

Fig. 6. Deployment diagram of an exemplary laboratory experiment

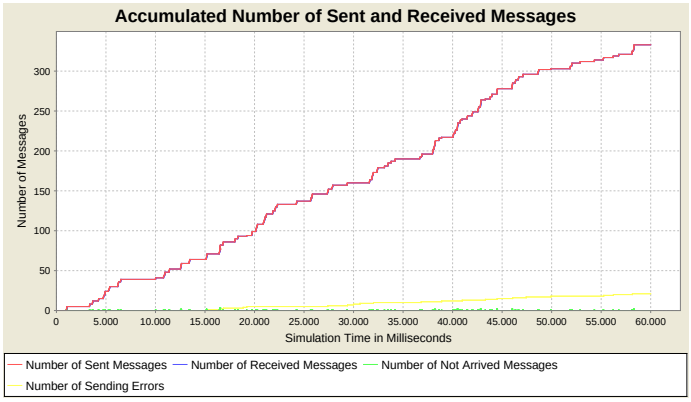

Fig. 7. Accumulated number of sent and received messages in a simulation experiment with a Chord overlay network

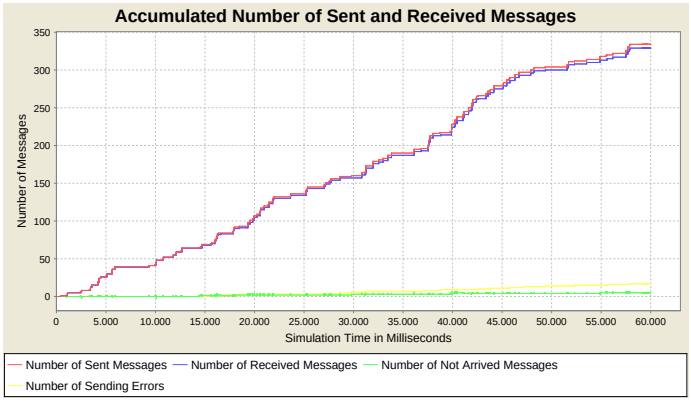

Fig. 8. Accumulated number of sent and received messages in a laboratory experiment with a Chord overlay network

It contains a deployment diagram of an exemplary laboratory experiment running on two computers, which can be easily extended to more computers. For executing laboratory experiments, the model of the P2P application is extended for running on a laboratory network. Therefore, the *PhysicalNetwork* model component is not needed anymore. Each P2P application gets an individual configuration file with initialization information and an individual event list. The event lists of the simulation experiments are reused, so that the same events occur in laboratory experiments. In the following section, we describe some laboratory experiment results and compare them with the simulation results.

C. Simulation vs. Laboratory Experiment Results

During the simulation and laboratory experiment runs, the same metrics are used for measurements to allow a direct comparison of experiment results. For example, Figure 7 depicts the accumulated number of sent and received messages during a simulation experiment run, whereas Figure 8 depicts the corresponding laboratory experiment results. Even though the same initial event list of the simulation experiment has been reused for the laboratory experiment, we observe slightly differing experiment results. In the figured laboratory experiment results for instance, one more message has been sent than in the corresponding simulation experiment. Another difference is that the number of received messages is smaller than the number of sent messages in the laboratory experiment. Five messages have not been received and not been registered as sending errors. These messages must get lost during the data transfer.

Several further metrics have been used during the evaluation of *PeerSE EE* and most compared results are similar. Deviations have been measured using the metric *Time to First Answer*. The results of the laboratory experiment have been measured on a cluster with busy and idle computers. The major part of the time values on idle computers lay between 0 and 50 milliseconds and therefore under the time values of the simulation experiment. In comparison with the simulation experiment, there have been some outliers in the laboratory experiments. The maximal time to answer on an idle cluster has been above 270 milliseconds. On a busy cluster, the measured values have been much bigger and outliers could be measured with delays of more than 2000 milliseconds. The CPU usage and network traffic have not been modelled in the simulation model, so that these values can not be recreated in a simulation experiment without extending the simulation model. The actual simple physical network model uses a pseudo-random transmission time in milliseconds in the interval of $(15, 20]$ for sending a message from one computer to another.

D. Real-World Experiments

In the completion phase, the laboratory P2P application is extended to a real-world P2P application by simply adding a main method. A simple GUI has already been implemented as part of the simulation model to control a P2P application during a simulation experiment for test purposes. Figure 22 shows the screenshots of three P2P applications, that are running on different computers in a local network and are controlled by a real user. The P2P applications use a Chord overlay network on top of a TCP/IP based physical network for answering user queries.

The comparison of simulation and laboratory experiment results with experiment results of real-world P2P systems with real-world users is out of the scope of the *PeerSE EE*, but such a comparison is possible by using the same metrics. The challenge would be to implement large experiment settings with real-world computers and real-world users.

V. Related Work

In the course of developing the *PeerSE EE*, we reviewed 37 P2P system simulators and many other tools that can be used for analyzing or implementing P2P systems. Most P2P simulators focus on special P2P architectures, are not designed for a reuse of simulation models for implementing real-world P2P systems, and do not support a validation within a laboratory network. In the following, we present tools that are comparable with the *PeerSE EE* because they allow the simulation of P2P systems and their execution on physical networks.

Neko [8] is a platform for the design and performance evaluation of distributed algorithms. The same implementation can be simulated and executed on a physical network. In contrast to the *PeerSE EE*, Neko does not include a GUI, a simulation model for P2P systems comparable with the *PeerSE Simulation Model*, and concepts for comparing experiment results from simulated and executed experiments.

MACEDON [9] is a method for specifying distributed algorithms in a compact domain-specific language which is used for generating code that can be deployed in evaluation infrastructures and in physical networks. In contrast to the *PeerSE EE*, a direct comparison of simulation and laboratory experiment results is not supported and the visualization of experiment settings and results is not part of the MACEDON method.

Jones and Dunagan report their experience with the development of running P2P systems within the Herald project [10]. Simulation and real-world experiments have been executed during the development of a P2P system. The same code base has been used for the simulation model and the real-world system. A comparative analysis of P2P architectures, a GUI for visualizing topologies or experiment results, and a direct comparison of simulation and laboratory experiment results are not described.

WiDS (WiDS implements Distributed System) [11] is an integrated toolkit for the development of distributed systems. It supports the execution of the same code within simulation experiments and as a real-world system. In contrast to this work, WiDS does not contain a GUI, nor does it support comparative analysis or a comparison of simulation and laboratory experiment results for a controlled transition from simulation models to laboratory systems.

VI. Conclusion and Future Work

We presented the *PeerSE EE* for analyzing and implementing P2P systems based on simulation. The *PeerSE EE* includes the *PeerSE Simulation Model* which can be initialized and visualized via *PeerSE Graphs*. We showed that the *PeerSE EE* allows a comparative analysis of P2P system models based on different P2P architectures. The best fitting P2P application model component has been be reused for implementing a laboratory P2P application which can be used for laboratory experiments on a laboratory network with distributed computers. Experiment results of simulation and laboratory experiments have been comparable directly because the same event lists and metrics have been used. In the evaluation example, only small behaviour discrepancies of the simulated P2P system and the laboratory P2P system occured. Therefore, we speak of a controlled transition of the simulation component of a P2P application to a laboratory P2P application. A transition to a real-world P2P application is also possible and has been implemented, but the execution of experiments with real-world P2P applications controlled by real users on a real physical network may be a complex task.

The use of existing laboratory networks like PlanetLab [12] can be useful for testing P2P systems on a large physical network. We have to consider that the model component of the physical network should model the laboratory network for a direct comparison of simulation and laboratory experiment results.

References

[1] "PeerSE - Analysing and Developing Peer-to-Peer Systems with PeerSE." [Online]. Available: http://www.peerse.eu/

[2] G. Chartrand and O. R. Oellermann, *Applied and Algorithmic Graph Theory*, ser. International Series in Pure and Applied Mathematics. McGraw-Hill, 1993.

[3] G. Smith, *The Object-Z Specification Language*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[4] "easim." [Online]. Available: http://sourceforge.net/projects/easim/

[5] D. Hildebrandt, L. Bischofs, and W. Hasselbring, "RealPeer - A Framework for Simulation-based Development of Peer-to-Peer Systems," in *Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP 2007), Naples, Italy, 7.-9. Februar*. Los Alamitos, CA, USA: IEEE, 2007, pp. 490–497.

[6] L. Bischofs and W. Hasselbring, "Generating and Visualising Organisational Structures of Free/Libre and Open Source Software Projects, Proceedings of the Third International Conference on Open Source Systems, Limerick, Ireland, 11-14 June, 2007," in *Open Source Development, Adoption and Innovation*, J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti, Eds. Springer, 2007, p. 392, poster.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord - A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, San Diego, California, USA*. ACM Press, 2001, pp. 149–160.

[8] P. Urbán, X. Défago, and A. Schiper, "Neko: A Single Environment to Simulate and Prototype Distributed Algorithms," *Journal of Information Science and Engineering*, vol. 17, no. 6, Nov. 2002.

[9] A. Rodriguez, C. Killian, S. Bhat, D. Kostić, and A. Vahdat, "MACEDON - Methodology for Automatically Creating, Evaluating, and Designing Overlay Networks," in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2004, pp. 20–20.

[10] M. B. Jones and J. Dunagan, "Engineering realities of building a working peer-to-peer system," Microsoft Research, Tech. Rep. MSR-TR-2004-54, 2004.

[11] S. Lin, A. Pan, Z. Zhang, R. Guo, and Z. Guo, "WiDS: an integrated toolkit for distributed system development," in *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*. Berkeley, CA, USA: USENIX Association, 2005, pp. 17–17.

[12] "PlanetLab - An open platform for developing, deploying, and accessing planetray-scale services." [Online]. Available: http://www.planet-lab.org/