Scalable Star-Topology Server-Array Based P2P Overlay Network Testbed

Otso Kassinen, Erkki Harjula, and Mika Ylianttila

MediaTeam Oulu, Computer Science and Engineering Laboratory
P.O.BOX 4500, FIN-90014, University of Oulu, Finland
firstname.lastname,@ee.oulu.fi

Abstract. We describe a scalable server-array based testbed for simulating various usage scenarios of peer-to-peer (P2P) overlay networks. Each server is responsible for a subset of the simulated peer processes, managed by the mechanisms presented herein. The system follows a star topology, where one master server acts as a point of control for a set of slave servers. We present both the structure of the system and the activities needed before, during, and after a simulation run in order to accomplish automated simulations, where each interesting combination of the variable parameters of the overlay network is evaluated. The functionality of the control scripts is explained in detail. Among other things, the system sets up the required start conditions for a P2P overlay simulation, manages the online-time and the specific P2P activities of each simulated peer, and facilitates the handling of the generated log files, from which the result statistics are derived.

Keywords: Network simulation, P2P overlay networks, testbed control logic.

1 Introduction

The tremendous popularity of peer-to-peer (P2P) networking technologies has led to the need for suitable, efficient testing and simulation tools for different kinds of P2P networks. Simple and small-scale tests can sometimes be run with ad-hoc setups, but advanced tests require a more organized approach.

Structured i.e. distributed hash table (DHT) based P2P networking algorithms have contributed to the scalability of the present-day P2P networks, but at the same time these advancements have inevitably led to the need for more capable underlying simulation environments in the design, proof-of-concept, and parameter-tweaking phases during the development of novel P2P networking systems. Special challenges in P2P simulations include the appropriate control of peer actions, provision of a realistic simulation environment, and the observation and analysis of the state of the distributed system and the results of actions in the network. The most significant issue in the design and deployment of new P2P networking technologies is scalability, which is – not surprisingly – also a key motivation in the development of P2P-oriented network simulation tools, as indicated by the recurring mentions in the literature [1-3]. Simply the ability to evaluate entirely new P2P protocols or algorithms, which might not be feasible with generic simulation tools, is the other main motivation for the creation of P2P-oriented network simulation systems [2, 4].

Literature on P2P simulations can be roughly divided into two categories. Firstly, several P2P simulation tools have been proposed [1-4]. Secondly, in addition to research on the design of entirely new P2P simulator systems, there are a large number of studies about using an existing simulation system, P2P-oriented or generic, for conducting a specific kind of P2P-related experiment. Examples of the second category include: using the explicitly P2P-oriented OverSim simulator for evaluating various properties of protocols for P2P overlay operation [5]; using the explicitly P2P-oriented P2PSim simulator for comparing the efficiency of iterative, recursive, and semirecursive routing schemes [6]; and using the generic NS-2 simulator for studying the delay distribution that is related to data recovery [7].

In this paper, we describe a simple yet functional P2P overlay network simulation setup, called *Scalable Star-Topology P2P Testbed* (SSTPT). This testbed system is originally intended for, but not restricted to, the simulation of DHT-based P2P protocols in scenarios, where data resources are published and searched in the distributed database formed by the P2P overlay network.

The particular technical requirements and the design goals of the testbed were derived from the simulation objectives explained in [8] and [9], for evaluating the performance of an experimental P2P protocol with interesting parameter ranges in the presence of churn and other varying network conditions. The simulated environment had to provide neutral starting conditions for the simulations (for example, making sure that a sufficient number of peers are online before starting the actual activities in a simulation), manage the flow of the actual simulations according to the selected peer behavior rules, handle the situations after simulations with any required clean-up activities, store the result logs for later analysis, and provide a bootstrap service for new peers.

As for experiment-related terminology, we use the terms *treatment* and *trial* as defined in [10], when referring to different conceptual parts of simulation experiments. A treatment is a combination of *settings* of *variables*. In our system, variables are the variable parameters of simulations, and their settings are values from the ranges explained in [8] and [9]. A trial is a collection of treatments. In our system, a trial is formed by the treatments which are conducted in series one after another until the set of interesting combinations of variable settings has been exhausted.

The structure of the paper is as follows. In section 2, the system structure is outlined. In section 3, the operation of the testbed system in different phases of the simulations, sub-dividing the conduction of one complete trial into logical steps, is explained. In section 4, the results are briefly discussed.

2 System Structure

The physical setup of the system is straightforward. There are S slave units (SU) and one master unit (MU), which are usually dedicated server machines. The machines are arranged in a star topology. The N simulated peers of an overlay network are allocated to the SUs. Because of the desired symmetrical nature of testbed load division, it is beneficial if every SU has identical components and configuration, including identical operating system installations. In practice, this uniformity requirement of the SUs is typically fulfilled by employing an organization's server arrays, which typically consist of a large number of similar rack or blade server machines acquired in a mass purchase.

The SUs are connected to each other using a high-speed Ethernet switch, which also connects them to the Internet and to the MU. Because the network connections between the SUs are critical for the fluent operation of large-scale, high-activity P2P simulations, this kind of setup is good, providing small SU-to-SU network latencies and largely isolating the non-simulation-related network traffic from the testbed.

The physical setup of the main components of the testbed is illustrated in Fig. 1. As an important addition to the system, the simulations can also employ mobile devices, allowing for simulation setups where measurements can be performed using actual mobile devices that participate in a large P2P overlay using, for example, a cellular Internet connection [9]. These mobile peers are, however, not directly part of the SSTPT system built specifically for simulations in a laboratory setting.

Linux-based open-source software infrastructure for the servers is easy to obtain and install, provides excellent facilities for control scripting including inter-process communication (IPC) and data analysis tools, and is usually well supported by the staff in any technical organization.

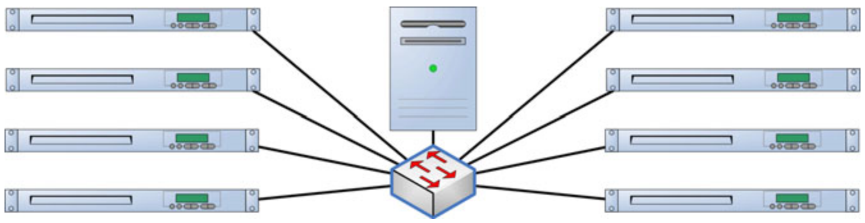

Fig. 1. Master and slave units with $S = 8$

The functionality of the testbed is largely realized using Bash shell scripts, which control all aspects of a simulation. In Table 1, an overview of the scripts and their deployment on the different machines is provided. The operation of the scripts during different phases of a trial is described in detail in section 3.

Table 1. Overview of the functionality and deployment of the simulation control scripts

Script name	Functionality	Deployment
Master Control	Controls all other scripts	MU
Slave Reconfigure	Installs updated scripts to all SUs	MU
Central Catalog	Manages dynamic global control data	MU
Slave Control	Controls actions within one SU	SU
Peer Control	Controls the actions of one peer	SU
Distr. Engine	Generates online/offline time values following a given distribution function	SU

3 SSTPT in Operation: Conducting a Trial

A state diagram providing a high-level picture about the conduction of a trial with SSTPT is shown in Fig. 2. The activities are explained in more detail in the following sub-sections.

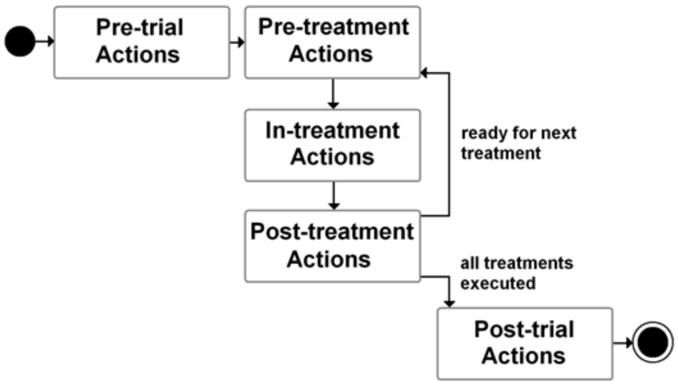

Fig. 2. High-level state diagram of a trial

3.1 Pre-trial Actions

The MU and SU machines are up and running and do not contain any unwanted processes consuming the computing resources required for the testbed operation. The P2P protocol software (the peer implementation to be run as several concurrent instances) has been installed on all machines. The basic preconditions, such as a user account capable of running the relevant applications and being authorized to use a sufficient amount of resources such as disk space, have been prepared on the machines. The clocks of the servers have been synchronized to have a clock difference of at most a few seconds; the Network Time Protocol (NTP) system provides accurate enough synchronization.

On the MU, the human operating the simulation experiment sets the Master Control script to contain the correct parameters in two categories: 1) the ranges of the actual variable parameters (and any required constant parameters), which constitute the basis for different treatments and are most interesting from the viewpoint of the experiment's outcome, and 2) the supporting parameters which affect the flow of the treatment or trial as a whole and typically have constant settings during an entire trial. The category (2) includes at least the pre-treatment wait time $t_{pre\text{-}wait}$ [minutes], the treatment duration $t_{treatment}$ [minutes], and the post-treatment wait time $t_{post\text{-}wait}$ [minutes].

The Slave Reconfigure script is run to propagate an instance of the Slave Control, Peer Control, and Distr. Engine scripts with any needed trial-specific modifications to each SU. The modifications to the content of Slave Control are made at a single location, i.e. on the MU, and Slave Reconfigure makes sure that each SU contains the same version. The transfer of any files (such as updated scripts or generated logs) between the MU and SUs is done with the scp application; however, the direct control channel between the MU and each SU is realized using a socket connection from the Bash scripts with help of the nc network scripting tool. The MU knows the IP addresses or DNS names of the SUs.

While all simulated peers are intended to be run on the SUs or possibly on separate mobile devices, the bootstrap service – which is a specialized peer instance – for the peers is installed on the MU. It should be noted that while the actions on the server-array based peer instances are controlled with scripts, the actions of the optional mobile peers must be either manually controlled (including, for example, the manual startup of the peer process on the mobile device) or scripted with methods that are outside of the scope of this paper.

3.2 Pre-treatment Actions

The Slave Control scripts on each SU are initiated and they start waiting for operation instructions from the MU. When every SU contains one running instance of the Slave Control script, the Master Control script on the MU (looping through the desired parameter combinations) starts preparations for the new treatment.

The Master Control script resets the bootstrap service and the database of the Central Catalog on the MU, thus these entities' state from any previous treatments does not affect the new treatment. Based on the known $t_{treatment}$, Master Control calculates the starting timestamp T_{start} (MM:SS).

Then, the MU prepares and passes an execution ticket (ET) to each SU over the control channel. The ET is a brief message – a string of whitespace-separated values – containing all the information a SU needs for accomplishing its responsibilities in cooperation with the other SUs during one treatment. This information is provided in the form of values for specific Bash script variables in Slave Control.

The ET for a SU contains, among other things, the number of peers that this specific SU is responsible for. If there is known asymmetry in the capabilities of the SUs, the configuration of the MU must take this into account and divide the responsibilities (how many peers are allocated for a given SU) according to that.

3.3 In-treatment Actions

When a Slave Control script has received the ET, it prepares the starting state of the P2P overlay network nodes that it is responsible for. This may, for example, involve bringing online exactly half of the peers and let them already exchange some routing information in order to achieve a neutral starting state for the overlay simulation. When the clock of each SU, approximately at the same moment, reaches the timestamp T_{start}, the Slave Control scripts start the actual simulation, which lasts until the clock reaches the timestamp $T_{start} + t_{treatment}$. Slave Control brings each new peer online by instantiating the Peer Control script. Slave Control manages the unique peer IDs and their transport protocol ports on the localhost in order to prevent conflicts between different instances of Peer Control. Slave Control also provides crucial pieces of information such as the bootstrap service IP address and port and the number of publishable resources per peer; all of these parameters originate, of course, from the ET passed from the Master Control.

During its lifetime, a peer will publish and lookup data resources, and do other specified actions. The actions of a peer are defined by the Peer Control script, which received its parameters from the Slave Control script. The Peer Control script loops, in steps of one second, through a loop where counters for different P2P action types are incremented upon every iteration. When the counters reach a value that was specified for the given P2P action type (for example, resource lookup once every 30 seconds), the P2P action is invoked: Peer Control issues the corresponding command to its P2P software instance, which then carries out the specified action. The Peer Control script uses named pipes (identified by the peer instance's peer ID or its Unix process ID in the SU) for passing the commands to its corresponding P2P protocol software instance.

The Central Catalog exists, because often there is need for storing some centralized information that is not known by the P2P software under test but is needed by the simulation framework itself. In [8] and [9], the Central Catalog was used for remembering, which resource IDs (DHT keys of data resources) were currently in use, so that the Slave Control was able to issue resource lookup requests for which the "not found" response indicated a real failure of the overlay to locate the resource and not a situation where the searched resource simply was non-existent. The Central Catalog is accessed over a HTTP interface, which allows the actions for resetting the database of the Central Catalog and inserting, querying, or removing a key-value pair (for whatever usage the simulation control requires this service).

If the online/offline times of a peer are required to follow a specific statistical distribution, the Distr. Engine is used for generating a value that serves as the duration of the next online (or offline) time period of the given peer. If, for example, the desired distribution is the exponential distribution, a value t following that distribution can be created from a random real number r $(0 < r < 1)$ and the expected mean value λ^{-1} as follows:

$$t = -\lambda^{-1} \cdot \ln(r) \tag{1}$$

At the end of a treatment, the Slave Control script kills all currently running Peer Control instances and their P2P software instances, and begins the post-treatment wait time $t_{post\text{-}wait}$ in order to allow for possibly slow operations (such as log copying or any process clean-up actions).

3.4 Post-treatment Actions

When the protocol activity has stopped, the log files are compressed using BZIP2 and transferred to a log repository on the disk of the MU for centralized storage and, later, analysis. The name of the compressed file is created from the used parameter values, thus it is easy to see which logs correspond to a specific treatment. In another possible approach, the log messages are transferred already in real-time during the run to the MU or another storage site, but obviously this can skew the results as the network connection, used also for the P2P protocol messaging, may become a bottleneck and also lost messages can be a problem.

3.5 Post-trial Actions

The logs on the MU are decompressed and analyzed with suitable tools. The analysis is facilitated if the logs are in an easily parseable form and, for example, each line contains the complete information about one observable action and is easily linkable to interrelated actions (such as the initiation of the lookup of a resource and the eventual finding of that resource).

The observation of the simulation results is based on the premise that the outcome of a treatment is determined by examining the log files or similar data entities that were generated during the simulation. This, of course, requires the P2P protocol software under test to have a logging facility and the log entries must be accurate enough to determine what operations were run and what were their end results or intermediate results, such as the re-routing hops of messages between several peers of the DHT overlay.

4 Discussion

The feasibility of SSTPT has been demonstrated by two medium-sized overlay network simulations, which have been conducted with it. In [8], the performance of an experimental P2P protocol was evaluated. The variable parameters in the trial were the overlay size (N = 200 and N = 2000), the churn rate (i.e. the online and offline times of the peers) and the frequency of resource lookup requests. In addition to the variable parameters, several constant parameters were used: some of them were assigned in the ET, and some were set during compilation time in the P2P protocol software. The log files of the trial were parsed using grep-based scripts, which counted the occurrences of specific types of log lines in the files, thus making it possible to determine values for performance metrics such as lookup request success ratio. In [9], the overlay contained also mobile peers. The aim was to measure the battery life of the mobile devices in P2P usage. The variable parameters in the trial were a subset of the variable parameters used in [8].

The SSTPT system is a simple-to-configure, rather flexible tool for examining the operation of P2P overlay network protocols. Future work on SSTPT may include creating more refined control functions for the actions of the peers; these can, for example, be based on more realistic modeling of the behavior of P2P network users.

Acknowledgments. The authors thank Jari Korhonen for his valuable contribution in the design and implementation of the testbed system.

References

1. Binzenhofer, A., Hossfeld, T., Kunzmann, G., Eger, K.: Efficient Simulation of Large-Scale P2P Networks: Compact Data Structures. In: 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 467–474 (2007)

2. Dinh, T.T.A., Theodoropoulos, G., Minson, R.: Evaluating Large Scale Distributed Simulation of P2P Networks. In: 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, pp. 51–58 (2008)
3. Dinh, T.T.A., Lees, M., Theodoropoulos, G., Minson, R.: Large Scale Distributed Simulation of P2P Networks. In: 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing, pp. 499–507 (2008)
4. Shi, G., Long, Y., Gong, H., Wan, C., Yu, C., Yang, X., Zhang, H.: A High Scalability P2P Simulation Framework with Measured Realistic Network Layer Support. In: IEEE International Performance, Computing and Communications Conference, pp. 311–318 (2008)
5. Munoz-Gea, J.P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J.C., Martinez-Rojo, A.M.: Simulation of a P2P Application Using OverSim. In: First International Conference on Advances in Future Internet, pp. 53–60 (2009)
6. Cheng, Y., Wen, X., Sun, Y.: Simulation and Analysis of Routing Schemes in Structured P2P System. In: ISECS International Colloquium on Computing, Communication, Control, and Management, pp. 524–527 (2008)
7. Dandoush, A., Alouf, S., Nain, P.: Simulation Analysis of Download and Recovery Processes in P2P Storage Systems. In: 21st International Teletraffic Congress, pp. 1–8 (2009)
8. Kassinen, O., Harjula, E., Ylianttila, M.: Analysis of Messaging Load in a P2PP Overlay Network under Churn. In: IEEE Global Telecommunications Conference, pp. 1–5 (2010)
9. Kassinen, O., Harjula, E., Korhonen, J., Ylianttila, M.: Battery Life of Mobile Peers with UMTS and WLAN in a Kademlia-based P2P Overlay. In: 20th Personal, Indoor and Mobile Radio Communications Symposium, pp. 1–4 (2009)
10. Bock, P.: Getting It Right: R&D Methods for Science and Engineering. Academic Press, London (2001)