P2PTester: testing P2P platform performance

Bogdan Butnaru, PRiSM laboratory, UVSQ
Florin Dragan, PRiSM laboratory, UVSQ
Georges Gardarin, PRiSM laboratory, UVSQ
Ioana Manolescu, Gemo group, INRIA Futurs
Benjamin Nguyen, PRiSM laboratory, UVSQ
Radu Pop, Gemo group, INRIA Futurs, Mandriva
Nicoleta Preda, Gemo group, INRIA Futurs
Laurent Yeh, PRiSM laboratory, UVSQ

1 Introduction

Recent years have seen significant development of peer-to-peer systems (P2P). Specific distributed data structures have been proposed, mostly by recycling and improving pre-existing works on distributed shared memory or DSM, each of which seek to provide best performance for the crucial *locate* operation: locate a data item in a potentially large set of peers. Such data structures are typically called *overlay networks* [5, 6, 7, 8].

On top of overlay networks, numerous P2P content management platforms have been developed [4, 9]. The basic functionalities of a P2P content management platform are not fundamentally different from the goals of traditional distributed databases [10]: give to the user *the illusion of a centralized system* while executing complex data management operations in a distributed setting. Thus, what really makes the difference between such systems (and presumably, what will determine their success or failure) is their *performance*. Performance measures are typically based on benchmarks and/or systematic testing suites.

However, to this day, no such benchmarking and measurement tools have been proposed for P2P data management architectures. One reason is the inherent complexity of testing distributed systems; however, this complexity has been tamed in the past, even since the very first day of TPC [11] transaction performance benchmarking. Another reason lies in the apparent diversity of P2P platforms:

- The underlying communication layer ranges from sockets, to Java, through XML message-based interaction.
- The network structure can vary widely, from structured networks (among which DHTs are just a subset !) to unstructured networks [12, 13] or hybrid systems etc.
- The applications envisioned in each system also vary a lot, due to the usage of relational [9], XML [4] or RDF [13] data model. In consequence, the "query language" (or its corresponding facility in each P2P platform) also varies a lot across systems.

We propose to demonstrate *P2PTester*, the first platform to systematically measure the performance of P2P content management systems. P2PTester is a Java-based application which *wraps around*, and *interfaces with*, any arbitrary P2P system (see assumptions on the underlying system in the next section). P2PTester allows the user to:

- launch, in a supervised manner, the construction of a peer network of controlled size and complexity
- measure the space and time costs of the process of indexing the data from this set of peers, within the P2P network
- measure the space and time costs associated to processing specific queries in each system
- trace the communications spawned from the processing of each specific query or search issued by a peer. This information is useful as a hint on the communication complexity of the system, but may also prove valuable for the system implementor, by helping him to trace (and perhaps debug) his platform

The existing P2P testing projects [2, 3] offer a solution for strictly testing the overlay structures (the performances of interconnection methods in terms of failure recovery procedures, number of exchanged messages, ...). Our approach differs from these works in the sense that we propose a complex testing tool suitable for *testing, developing, and performing comparative analyses of P2P database-oriented applications*. To that purpose, P2PTester is specifically concerned with measurements regarding the management of data in a P2P system (i.e. size of exchanged messages, size of data sent by each peer involved in responding to a query, ...) rather than pure simulation. As a first feasibility study, we plan to demonstrate how P2PTester can be used to measure a target application (See Section 3) issued from the context of the eDOS RD European project [14].

2 P2PTester outline

The goals of P2PTester are as follows:

Genericity: our first and primary goal is that our system be usable to measure a wide range of P2P platforms.

Scalability: the tester application must be ready-to-deploy at a large scale, since we want to test the performance of distributed systems with a large number of "real" peers. Moreover, its own overhead should ideally be low, so as not to influence the overall performance of the system measured.

Modularity: P2PTester must allow performing fine-grained measures of various components of a P2P data management application. Thus, it may be interesting to grasp the performance of a P2P system's locating function *only*, or of its indexing component only, of its distributed query processing operations only etc. This essential feature also enables the testing of complex, hybrid systems, for instance XML or RDF data management layers deployed alternatively on a DHT or on a "neighbor-connection" network etc.

2.1 Generic P2P System Architecture

P2PTester is structured in four independent layers, schematically presented in Figure 1.

Communication. The first layer offers a trusted communication infrastructure for exchanging messages between peers. For generality, we provide a common interface that must be implemented by any module that provides basic communication functionalities. In the first tester release, we offer a basic socket-based communication module.

Application. Application-specific modules belong to this layer, which includes the peer entity, mainly composed of the indexing, routing and query processing modules. *The indexing module* is responsible for propagating the (system-specific) information used to locate data items and to process queries. *The routing module* exploits such information to locate in the P2P network peers which may be useful in answering a query. Finally, *the query* *processing module* (if it exists) performs query processing operations, and it initiates communications whose aim is to ship data between peers (these communications may not use the routing modules).

Test. The test layer is in fact interspersed with and between the other layers, in order to attain our genericity and modularity goals. Given the variety of platforms and implementation details, an effort will be made to ensure that once defined, a test is general enough to be run on several similar, yet different architectures. The test layer includes a *peer manager*, responsible for launching and stopping peers of the system tested, and a *test control interface*, which receives and processes tests to be run. To gather and interpret (distributed) test results, a *distributed logger* is present in this layer, which records the details of each event in the evolution of the network. Finally, a *test monitor* coordinates the modules in this layer, and automates the production of test results.

Test Generation. To help the user devise and run tests, we provide a (graphical) interface where users can define tests by specifying several basic parameters: the number of peers in the test, the type of peer overlay that must be used, the data sets to be indexed, the queries to be executed, the duration of the test, how the measure results are to be gathered and structured etc. The test results are presented with the help of an interactive visualizer.

As a supplementary testing help, geared more specifically towards the P2P platform *developers*, *P2PTester* provides a basic set of modules which can be plugged together with specific data management layers. For instance, P2PTester provides its own DHT implementation. The purpose is not to compete with existing DHTs, but to enable fast prototyping of a running system, and to allow testing the extent to which a system's performance depends on a specific DHT (by trying it alternatively with the one provided by P2PTester).

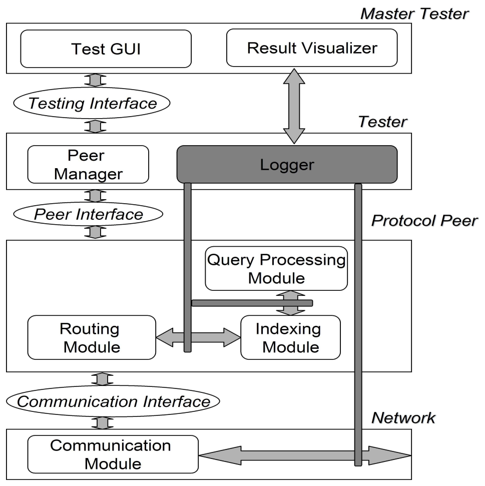

Figure 1: The Tester Architecture

2.2 P2PTester's Interactions with the System Under Test

The tester offers a common API for writing P2P test scenarios for a large spectrum of P2P system architectures. *Each method in this API corresponds to (wraps) one of the methods provided by the P2P system under test.* Thus, a call to the *join* method, which normally allows a peer to join a P2P system, is intercepted by P2PTester, which logs it, then redirects it to its rightful destination peer, all the while measuring its response time, the communications engendered by the *join* implementation provided by the system etc. Other frequent calls such as *leave* (disconnecting a peer from the network), *publish* (publishing indexing/catalog information in the network), *remove* (withdrawing published data), *leave* (disconnecting a peer from the network), and finally *query* (processing locate or more complex query requests) are intercepted and logged by P2PTester in a similar way.

P2PTester deployment is envisioned as follows. Assume the intended P2P deployment architecture (in the absence of testing) consists of

N_L *logical peers* running a given P2P data management software, deployed on N_ϕ *physical peers* (or machines), where $N_L \geq N_\phi$. Deploying the same architecture while testing it with P2PTester involves deploying N_ϕ P2PTester instances, one on each physical machine, and using each instance to start the corresponding logical peers and direct tests on them.

The parameters measured by P2PTester during a test run include the following:
• Number, and size, of messages required for: (*i*) joining/leaving, (*ii*) publishing and (*iii*) querying.
• Size of index/routing data stored at a given peer
• Size of the data currently published in the network
• Query result sizes
• Query processing time, broken down (whenever the underlying system allows it) into:
- Locate time, or the time it takes to identify the peers in the network holding useful data;
- Pre-processing time, such as the time to filter out some of the located peer and/or to chose the ones to contact;
- Processing time, spent in the data transfer and processing operations specific to query processing in the system under test;
- Post-processing time, such as the time to rank results, aggregate them etc. (in short, all operations that the query peer may perform before presenting results to the user).

3 Demo Scenario

3.1 Target Application

We plan to demonstrate P2PTester on a distributed application dedicated to the collaborative production, testing, integration and distribution of free software. This application is inspired from the eDOS R&D project currently ongoing [14].

Many kinds of users participate in such a system. Most users are *writers*: they contribute successive versions of specific software packages and/or their documentation. Other users are *testers/integrators*: they need to have up-to-date versions of software packages on their sites (possibly automatically pushed by the system as part of a subscription), in order to test and integrate these packages among them. A few sites have a *publisher profile*: periodically, they publish large-scale integrated software suits, together with their documentation etc. Other sites serve as *mirrors*: they only replicate published suits, with the purpose of making them available faster to downloaders scattered all over the world. Finally, a large majority of participants only *download* software, either integrated suites or individual packages under test.

We have chosen this application as representative mainly due to:
• the distributed nature of the application
• the dynamicity of all peers involved (everybody can get involved in such a free software development and exploitation effort, and similarly, everybody can leave at any time)
• the variety of read/write profiles of the participating peers, which should allow to test the suitability of a P2P system for a large spectrum of real-life applications.

3.2 P2P Systems Under Test

KadoP[4] is a P2P system built on the support offered by a DHT. The system can be used for publishing and querying XML documents. Published documents are indexed using an extension of the DHT API. The KadoP query language is based on tree pattern queries. For testing KadoP we use the tester DHT and we integrate the specific query processing modules in the architecture of the tester.
PIER [9] is a relational query processor adapted to a masively P2P architecture. As KadoP, PIER is based on a DHT structure

that is used for indexing and querying data. In PIER queries are expressed in a relational language (e.g. SQL) and trasformed in optimized execution plans evaluated over the DHT.

As KadoP and PIER offer similar functionalities, in the demo presentation we plan to offer an comparative evaluation by testing the presented application model implemented over each of the two systems for showing the strenghts and weaknesses of each architecture and giving some ideas regarding the state of the art in the performance of P2P query processors.

4 Conclusion

P2PTester is the first attempt to measure P2P performance in a controlled environment. Our approach is to isolate basic components present in current P2P platforms, and to insert "hooks" for P2PTester to capture, analyze and trace the interactions taking place in the underlying system. This allows developers to gather useful feedback on their system, researchers to perform competitive analysis, and P2P research and development to profit in general from a thorough, across-the-board comparative analysis.

References

[1] Carlo Sartiani, Paolo Manghi, Giorgio Ghelli, Giovanni Conforti: XPeer: A Self-Organizing XML P2P Database System. EDBT Workshops 2004: 456-465

[2] http://overlayweaver.sourceforge.net/

[3] http://planet.urv.es/planetsim/

[4] The Kadop Project: http://www-rocq.inria.fr/gemo/Gemo/Projects/KadoP/

[5] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In Proceedings of the 2001.

[6] Druschel, P., and Rowstron, A. Past: Persistent and anonymous storage in a peer-to-peer networking environment. In Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS 2001) (Elmau/Oberbayern, Germany, May 2001), pp. 65-70.

[7] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-level Multicast using Content-Addressable Networks. In Proc. of the 2nd International Workshop of Network Group Communication (NGC), 2001.

[8] H. V. Jagadish, Beng Chin Ooi, Quang Hieu Vu: BATON: A Balanced Tree Structure for Peer-to-Peer Networks. VLDB 2005: 661-672

[9] HUEBSCH, R., ET AL. Querying the internet with PIER. In Proc. of VLDB (Sept. 2003).

[10] M. T. zsu, P. Valduriez. Principles of Distributed Database Systems. Prentice-Hall, 1991

[11] http://www.tpc.org

[12] Crespo, A. and Garcia-Molina, H., 2002. Routing Indices for Peer-to-Peer Systems. Proceedings of the International Conference on Distributed Computing Solutions (ICDCS'02).

[13] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza Peer Data Management System. TKDE, 16(7), 2004.

[14] http://www.edos-project.org/xwiki/