Building Adaptive Peer-To-Peer Systems

Evangelia Kalyvianaki and Ian Pratt
University of Cambridge Computer Laboratory
11 JJ Thomson Avenue, Cambridge, UK
{firstname.lastname}@cl.cam.ac.uk

Abstract

Most peer-to-peer systems are built upon the assumption that their running environment is homogeneous. However, for Internet-scale applications this assumption could lead to performance limitations. We propose a framework for building peer-to-peer systems which use performance monitoring techniques to adapt to changes in their environment.

1. Introduction

During the last few years, peer-to-peer (P2P) computing has gained new significant attention, through a variety of applications. P2P systems are running on thousands of nodes across the planet with different characteristics such as network capacity, processing power and disk storage and latency. However, early P2P systems both unstructured such as Gnutella and structured do not account any of the dynamics and the heterogeneity of their running environment.

Recent unstructured systems in particular KaZaA and GIA [2], do take heterogeneity into account. In KaZaA peers are divided into two categories: preassigned nodes with high bandwidth network access, known as supernodes and regular peers. Queries are forwarded only to supernodes, which maintain a list with the file names on their connected regular peers and therefore avoiding overloading all peers of the system. Saroiu *et al.* [5] report that both Gnutella and Napster hosts vary in terms of bandwidth, latency, lifetime and shared data. These observations led to the design of GIA, a Gnutella-like system which aims to respond to high aggregate query rates. For this purpose, each peer calculates the maximum number of queries it can handle per second and thus the maximum number of possible neighbours. Through this mechanism and other optimisations GIA achieves a three to five order of magnitude improvement in overall system capacity. In the above systems, the metric that shows the heterogeneity of the nodes is calculated based either on instant measurements or user provided information and maintained unchanged throughout the life of the application. However, peers' capacity as perceived by the application (e.g. availiable processing power) and available network bandwidth depend not only on their hardware characteristics, but also on their usage load. Since peers' capacity is dynamic a P2P system must be periodically updated on peer and network performance.

Existing structured P2P systems are not designed to take into account the heterogeneity of the peers [3]. We argue that these systems can be extended towards this direction without severe modifications. Several structured P2P systems incorporate topology awareness into their design. One example is Pastry's [4] proximity metric. A lookup for a particular nodeID may match several nodes in the routing table. Each match is tagged with a proximity metric which shows the distance, in terms of IP hops, of the current node from the given IP address. Such metrics can be extended to incorporate more information about the matched node such as processing capacity, disk latency and expected load.

In this paper we propose a framework for building adaptive P2P systems. Adaptation is based upon observations of the system's behaviour resulting from monitoring the nodes of the system.

2. Architecture

The framework is based on tracing kernel level and network traffic events on the nodes of the P2P network during the life of the application. Events are analysed in two different levels. Firstly, events are processed locally on nodes in order to produce statistics of the nodes' behaviour. Secondly, events across different nodes that participate in the same P2P operation, e.g. join, are combined into a single path. These paths are later used for analysing the behaviour of the P2P operation across the system. The framework consists of three stages: **gathering information**, **information analysis** and an **adaptation stage** described below.

1. **Gathering Information**
   The nodes of the P2P system generate traces during the execution of the application. The kernel-level op-

erations performed during the execution along with the network traffic associated with the application are stored in trace files. The tracing mechanism is the Linux Trace Toolkit (LTT) [6], a dynamic recording tool which logs events in the Linux kernel. A few of the more important event types are: system calls, socket communications, scheduling changes and file system operations. We have added a new event to capture network traffic in kernel-level.

2. **Information Analysis**
   This stage focuses on analysing performance across the P2P system by processing events generated from the previous stage and is divided into local and combined analysis. The local analysis aims to find node's capacity as perceived by the P2P application (e.g. processing load and available disk bandwidth) through processing raw events such as frequency of schedule changes, number of running processes and duration of file system operations. The combined analysis aims to study the performance of an application operation across different nodes. Different operations are identified from network packets across traces. The outcome of this analysis is the performance characterisation of an operation in terms of performance of individual nodes across the path of the execution of the operation as defined previously and network availability between successive nodes.

3. **Adaptation**
   Periodically, information of the current state of the P2P network resulting from the previous stage is published to the P2P applications running on different nodes. This information can be used in different ways by the application for adaptation purposes as paths or nodes with better overall performance can be preferred over others. An example is given in the next section.

3. Discussion

We are currently developing a prototype framework of the proposed architecture. As an example application we study the Scribe [1] multicast infrastructure, which is built upon the Pastry routing substrate. Scribe creates a tree in order to send messages to members of the same group. Members occupy the leaves of the tree, while messages are sent by the root. We use our prototype to study the performance of the paths from the root of the tree to the leaves when a message is sent. We aim to discover whether a path suffers from poor performance, when for example one or more participant nodes are highly loaded or the available network bandwidth across some parts of the path is limited. This information can be later used when a new member joins the tree. These experiments will be conducted with the prototype developed on the PlanetLab[1] testbed.

The framework is designed in a way that it does not significantly perturb the resources (e.g. nodes' CPU and network bandwidth) of the P2P network. Initial experiments suggest that worst case performance impact of running the modified LTT is not severe. Enabling tracing on a highly loaded web server reduced its performance[2] by 6%, while 3% of the events were lost. During the experiments all possible events produced by all processes are stored. Since in the proposed framework only the events related to the P2P application are captured, the measured impact can be further improved.

4. Conclusions

Internet scale applications such as P2P systems are running in a heterogeneous and constantly evolving environment. Such systems can attain better overall performance if they can adopt to these environmental factors. For this reason we propose a framework that monitors the P2P system and provides information regarding the nodes and network performance for adaptation purposes.

5. Acknowledgements

We are grateful to Intel Research at Cambridge for funding the work of Evangelia Kalyvianaki. We also thank the anonymous reviewers for their helpful comments.

References

[1] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 2002.

[2] Y. Chawathe, S. Ratnasamy, L. Breslay, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *ACM SIGCOMM*, Germany, Aug. 2003.

[3] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. In *IPTPS*, 2002.

[4] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[5] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Multimedia Computing and Networking 2002 (MMCN '02)*, 2002.

[6] K. Yaghmour and M. R. Dagenais. Measuring and Characterizing System Behavior Using Kernel-Level Event Logging. In *2000 Usenix Annual Technical Conference*, San Diego, CA, June 2000.

1 http://www.planet-lab.org

2 SpecWeb99 was used for these experiments