
PEER-TO-PEER SYSTEMS

Abstract

Peer-to-peer technology has become very popular for various applications such as file sharing or live streaming. Today, a large fraction of the Internet traffic is due to peer-to-peer applications. However, there are still many theoretical and practical challenges. In our research project, we mainly focus on

two such challenges: dynamics and cooperation. Besides designing and analyzing algorithms in theory, we have built several software prototypes such as the file-sharing tool BitThief, the live and on-demand streaming tool Pulsar, or the Wuala application.

Section

Distributed Computing Group

Coordinator

Prof. Dr. Roger Wattenhofer

TIK Participants

Thomas Locher
Remo Meier
Stefan Schmid

Funding

HaslerStiftung ManCom
HaslerStiftung DICS

Web

www.dcg.ethz.ch

Literals

PEER-TO-PEER SYSTEMS

Theory

In contrast to client-server architectures, the p2p computing paradigm seeks to harness the computing power of all machines in the network: A peer-to-peer system leverages resources of all clients such as bandwidth, storage space, or CPU cycles. A crucial advantage of such distributed systems is their scalability: As more and more peers join the network and the demand on the system increases, the total capacity of the system increases as well. Moreover, p2p systems are often fault-tolerant as data is replicated and there is no single point of failure.

However, there are also challenges. While a server is available most of the time, p2p systems often experience frequent membership changes: A user typically joins the network only for a short period of time, e.g., in order to download a file, and then leaves the network again. Hence, a p2p system in practice must guarantee seamless operation despite the ongoing joins and leaves. In peer-to-peer parlance, these rapid membership changes are called churn.

Another challenge in peer-to-peer computing comes from the fact that the participants – and therefore the resource contributors – do not always follow the suggested protocol but are rather uncooperative or selfish. Game theory can be used to study such behavior and to estimate the resulting damage.

Dynamics

Most p2p systems in the literature are analyzed against an adversary who can crash a functionally bounded number of random peers. After crashing a few peers the system is given sufficient time to recover again. Our research differs from this in two aspects. First, we assume that joins and leaves occur in a worst-case manner. We think of an adversary which can remove and add a bounded number of peers. The adversary cannot be fooled by any kind of randomness. It can choose which peers to crash and how peers join. Second, the adversary does not have to wait until the system is recovered before it crashes the next batch of peers. Instead, the adversary can constantly crash peers while the system is trying to stay alive.

We propose a solution which renders a system resilient against an adversary who continuously attacks the weakest part of the system. Indeed, our system is never fully repaired but always fully functional. Such an adversary could for example insert a crawler into the p2p system, learn the topology of the system, and then repeatedly crash selected peers, in an attempt to partition the p2p network. Our system counters such an adversary by continuously moving the remaining or newly joining peers towards the sparse areas.

Non-Cooperation

The power of peer-to-peer computing arises from the collaboration of its numerous constituent parts, the peers. If all the participating peers contribute some of their resources – for instance bandwidth, memory, or CPU cycles –, highly scalable decentralized systems can be built which significantly outperform existing server based solutions. Unfortunately, in reality, many peers are selfish and strive for maximizing their own utility by benefiting from the system without contributing much themselves. Hence the performance of a p2p system crucially depends on its capability of dealing with selfishness. A well-known mechanism designed to cope with this free-riding problem is the tit-for-tat policy which is for instance employed by the file-distribution tool BitTorrent.

However, selfish behavior in peer-to-peer networks has numerous important implications even beyond the peer's unwillingness to contribute bandwidth or memory. For example, in unstructured p2p systems – the predominant p2p architectures in today's Internet –, a peer can select to which and to how many other peers in the network it wants to connect. With a clever choice of neighbors, a peer can attempt to optimize its lookup performance by minimizing the latencies – or more precisely, the stretch – to the other peers in the network. Achieving good stretches by itself is of course simple: A peer can establish links to a large number of other peers in the system. Because the memory and maintenance overhead of such a neighbor set is large, however, egoistic peers try to exploit locality as much as possible, while avoiding storing too many neighbors. It is this fundamental trade-off between the need to have small latencies and the desire to reduce maintenance overhead that governs the decisions of selfish peers.

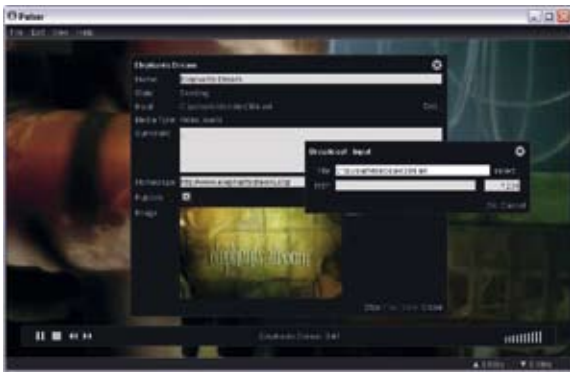


Fig. 1: Screenshot of Pulsar streaming tool

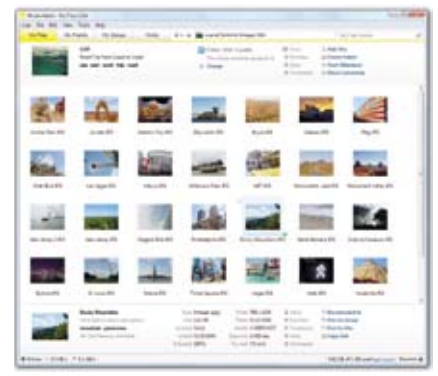


Fig. 2: The Wuala file sharing application

We have investigated the impact of selfish neighbor selection on the quality of the resulting network topologies by making a game-theoretic analysis. In particular, we studied the Price of Anarchy of p2p overlay creation, which is the ratio between an optimal solution obtained by perfectly collaborating participants compared to a solution generated by peers that act in an egoistic manner, optimizing their individual benefit. The importance of studying the Price of Anarchy in peer-to-peer systems stems from the fact that it quantifies the possible degradation caused by selfishness.

Selfishness is not the only challenge to the performance of distributed systems. Often, these systems have to cope with malicious adversaries who seek – independently of their own cost – to degrade the utility of the entire system, to attack correctness of certain computations, or to cause endless changes which render the system unstable. Aware of these threats, many researchers especially in the area of security and distributed computing have devised solutions to defend against such possible attacks. We initiated the study of a system consisting of both selfish and malicious participants. We ask: What is the impact of the malicious players on a selfish system's efficiency?

In order to capture these questions formally, we introduced the Price of Malice of selfish systems. The Price of Malice is a ratio that expresses how much the presence of malicious players deteriorates the social welfare of a system consisting of selfish players. More technically, the Price of Malice is the ratio between the social welfare or performance achieved by a selfish system containing a number of Byzantine players, and the social welfare achieved by an entirely selfish society.

Systems

Our theoretic work is often complemented by real implementations. In order to demonstrate that today's p2p systems cannot prevent users from free-riding, we implemented our own BitTorrent client BitThief which downloads entire files without uploading anything at all. (The client can be downloaded from <http://dgc.ethz.ch/projects/bitthief/>.) Although this client is rather a proof-of-concept and has a simple graphical user interface, it has attracted quite some attention and has become popular. In the following, two sample applications are presented in more detail.

Today, we are witnessing an explosion of online video content provided on websites such as YouTube. It is likely that in the near future, the Internet will also revolutionize television. Due to its scalability, peer-to-peer technology is an appealing paradigm for providing live TV broadcasts over the Internet. Live p2p streaming is not only an active field of research, but there are already commercial products emerging, e.g., JumpTV, PPLive, SopCast, among others, which provide television to thousands of viewers. Live streaming faces several challenges that are not encountered in other p2p applications such as file sharing. The stream-

ing content is required to be received with respect to hard real-time constraints, and data blocks that are not obtained in time are dropped, resulting in a reduced playback quality. Additionally, a live broadcast ought to be received by all users simultaneously and with minimal delay. Moreover, as video streams often already demand a high transmission rate themselves, redundant transmissions of the protocol itself must be minimized. Finally, any successful live streaming system has to be robust to peer dynamics.

We have designed and compared various algorithms and have built a streaming system called Pulsar. The key feature of Pulsar is its combination of push and pull operations: Peers forward a packet to their neighbors explicitly without preceding requests (push). This yields low latencies. This push phase fuels the subsequent pull phase where peers request the remaining packets from their neighbor. Pull operations are attractive because of the low overhead (no redundant transmissions).

Apart from real-world tests such as the broadcast of the IPTPS 2007 conference, we have performed extensive simulations of our protocol. Pulsar can be used both for live streaming and on-demand streaming.



Fig. 3: Our robust and locality-aware distributed hash table eQuus

We have also built more classic p2p systems, e.g., for file-sharing. Farsite is a secure, scalable file system that logically functions as a centralized file server but is physically distributed among a set of untrusted computers. Farsite provides file availability and reliability through randomized replicated storage; it ensures the secrecy of file contents with cryptographic techniques; it maintains the integrity of file and directory data with a Byzantine-fault-tolerant protocol; it is designed to be scalable by using a distributed hint mechanism and delegation certificates for pathname translations; and it achieves good performance by locally caching file data, lazily propagating file updates, and varying the duration and granularity of content leases.

Recently, we have started project Wuala (code-named Kangoo): Wuala is a new way of storing, sharing, and publishing files on the Internet. It is a free desktop application for Windows and Mac that brings its users a convenient and secure online storage. Unlike traditional online storage systems, Wuala is decentralized and can harness idle resources of participating computers to build a large, secure, and reliable online storage. Wuala has become a commercial product and is developed and run by Caleido AG.

Selected Publications

Thomas Locher, Remo Meier,
Stefan Schmid and Roger Wattenhofer
Push-to-Pull Peer-to-Peer Live Streaming
DISC 2007

Thomas Locher, Stefan Schmid
and Roger Wattenhofer

Rescuing Tit-for-Tat with Source Coding
P2P 2007

Thomas Locher, Patrick Moor, Stefan Schmid
and Roger Wattenhofer

Free Riding in BitTorrent is Cheap
HotNets 2006

Dominik Grolimund, Luzius Meisser,
Stefan Schmid and Roger Wattenhofer
**Cryptree: A Folder Tree Structure
for Cryptographic File Systems**
SRDS 2006

Thomas Locher, Stefan Schmid
and Roger Wattenhofer
**eQuus: A Provably Robust and Locality-Aware
Peer-to-Peer System**
P2P 2006

Thomas Moscibroda, Stefan Schmid
and Roger Wattenhofer
**When Selfish Meets Evil: Byzantine Players in
a Virus Inoculation Game**
PODC 2006

Thomas Moscibroda, Stefan Schmid
and Roger Wattenhofer
On the Topologies Formed by Selfish Peers
PODC 2006

Fabian Kuhn, Stefan Schmid, Joest Smit
and Roger Wattenhofer
**A Blueprint for Constructing Peer-to-Peer
Systems Robust to Dynamic Worst-Case Joins
and Leaves**
IWQoS 2006

Dominik Grolimund, Luzius Meisser,
Stefan Schmid and Roger Wattenhofer
**Havelaar: A Robust and Efficient Reputation
System for Active Peer-to-Peer Systems**
NetEcon 2006

Fabian Kuhn, Stefan Schmid
and Roger Wattenhofer
**A Self-Repairing Peer-to-Peer System Resilient
to Dynamic Adversarial Churn**
IPTPS 2005

Adya, W. J. Bolosky, M. Castro, G. Cermak,
R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch,
M. Theimer and R. Wattenhofer
**FARSITE: Federated, Available, and Reliable
Storage for an Incompletely Trusted
Environment**
OSDI 2002

Software

www.dcg.ethz.ch/projects/bitthief
www.getpulsar.com
www.pstreams.com
www.wua.la