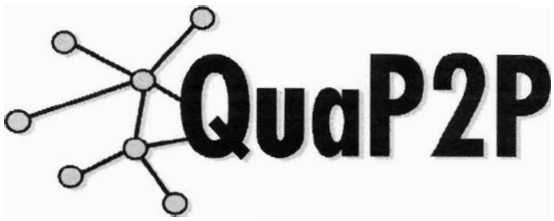
[GKS08] *Kalman Graffi, Aleksandra Kovacevic, Ralf Steinmetz*; **Towards an Information and Efficiency Management Architecture for Peer-to-Peer Systems based on Structured Overlays.** no. KOM-TR-2008-2, March 2008.



Technische Universität Darmstadt

Department of Electrical Engineering and Information Technology
Department of Computer Science (Adjunct Professor)
Multimedia Communications Lab
Prof. Dr.-Ing. Ralf Steinmetz

Towards an Information and Efficiency Management Architecture for Peer-to-Peer Systems based on Structured Overlays

Technical Report KOM-TR-2008-02

By

Kalman Graffi, Aleksandra Kovacevic, Ralf Steinmetz

Contact: Kalman.Graffi@kom.tu-darmstadt.de
http://www.kom.tu-darmstadt.de

First published: 3. March 2008
Last revision: 3. March 2008

For the most recent version of this report
please send a mail to Kalman.Graffi@kom.tu-darmstadt.de

DFG Research Group 733: Improving the Quality of Peer-to-Peer Systems

Towards an Information and Efficiency Management Architecture for Peer-to-Peer Systems based on Structured Overlays

Kalman Graffi, Aleksandra Kovacevic, Ralf Steinmetz
Technische Universität Darmstadt, Multimedia Communications Lab KOM
Merckstraße 25, 64283 Darmstadt, Germany
{graffi,sandra,steinmetz}@kom.tu-darmstadt.de

March 4, 2008

Abstract

Peer-to-peer applications are becoming more and more complex, a modular p2p system design consisting of various functional layers (e.g. FreePastry and add-ons) eases the development of new applications, as developers can rely on existing, efficient components. In order to use optimized strategies in these p2p functional layers, additional information on other peers and states in the p2p system is needed. We present in this paper an architecture, applicable on structured P2P overlays, which enables interested peers and parties to monitor the state of the P2P network and further provides the functionality of finding a queried set of peers fulfilling a requested attribute state. With this, functional layers in the P2P application can query for peers fulfilling specific requirements, which are then addressed to fulfill a layer-specific role. Our solution is scalable (leveraging the underlying DHT), easy to deploy (simple add-on to existing DHTs), efficient (O(log N) hops per query and update) and proposes a valuable component in future's modular component-based P2P applications.

1 Introduction

The field of peer-to-peer (P2P) research is broadening in recent years [1], ranging from classical overlays and content distribution, to P2P streaming, replication management, security issues and many other functional layers. With the grow of application areas for the P2P paradigm, more and more mature solutions are presented (e.g. BitTorrent [2] instead of Napster).

These optimized solutions often benefit from additional information they gather individually, for example in BitTorrent each peer measure individually the link quality to its neighbors. In replication management the replication layer is responsible for finding appropriate candidates to store replicas on them.

Future P2P applications will combine various existing P2P functional layers. Imagine an application in which you can search (unstructured overlay) or lookup (structured overlay) specific content, which you can download (content distribution) or directly stream (P2P streaming). After consuming the content, you may add a comment to the specific content, that is then replicated (replication) and synchronized (versioning) according to specific criteria. All these steps require individual specific P2P functional layers. In order to build modular and optimized P2P applications each functional layer in the P2P system needs to be optimized individually. Optimization requires information about the P2P network, that more valuable choices can be made. Up to now, each functional layer individually tries to gather and process information about the P2P network. Having various functional layer combined in an P2P application this method becomes inefficient and infeasible. We argue that the task for obtaining the required informa-

tion for each functional layer in a P2P system should be done by an additional information and efficiency management layer.

Contribution In this paper we present an optimized information and efficiency management layer for P2P systems, that fulfills the function to gather, aggregate and store the meta-information (i.e. attributes) about peers in the P2P network. Further it has the role of a cross-layer information system, that answers to queries of the various layers in a P2P system. A typical query asks for n peers in the network fulfilling specific requirements R. The information management layer provides as result a list of contact information to n peers, that fulfill these requirements, if there are n in the network. With an additional information and efficiency management layer in P2P systems, the load of information gathering can be taken from the various P2P layers, so that focus shifts from *how to obtain the information* to *how to use the information.*

Organization of the paper The rest of the paper is structured as follows. In Section 2 we present the problem and its requirements to a solution. Our solution, is presented in 3 in detail. In Section 4 we conclude our work. Please note, that this work is presented in a stripped form (no related work, evaluation not shown), as the complete work is currently under submission.

2 Problem Statement

In this section we summarize the key aspects of the problem statement for building an information and efficiency management layer. Efficiency, defined as quality provided in relation to corresponding costs, gains more impact in more and more modularized P2P systems. In order to come to decisions, which result in the same quality but come with less costs, a wider set of alternatives has to be known. These wider set of alternatives is then considered for the decision by ranking the alternatives according to a quality measure.

Current State

Individual layers in the P2P system gather information on other peers by their own. The information is then processed, analyzed and used by the specific peer (at the specific layer). E.g. the bandwidth capabilities of peers a BitTorrent system are measured and used individually by each peer. The load of information gathering and the benefit of information using is not shared.

Desired State

An information and efficiency management layer is to be seen orthogonal in the P2P system layer model. It is to be used by the individual functional layers to announce layer specific information to other peers. This information is then spread by the information management architecture and can be addressed by queries of all peers. The information management layer is built by the peers in the system (load sharing) and can be used by all peers (benefit sharing).

Functionality Goals

We state two goals for the functionality of an information and efficiency management layer. First, overlay independence. It should be applicable on any structured P2P overlay, which is compliant to Key-based Routing [3]. Dabek et al. proposed in [3] a common API for structured P2P overlays, which can be used to be overlay independent. Second, we define the query type which addresses the announced information. A solution should enabled peers to ask the information and efficiency management layer for the contact information of n peers fulfilling a list of requirements, which can be combined. A peer may ask e.g. for the contact information of 5 peers which have at least 200kb/s upload capacity in average, have been online for 5 hours, and have at least 10Mb available storage space. The information management architecture responds then with 5 peers fulfilling this criteria, that can be used by the querying layer for further processing.

Quality Aspects

Building an architecture for gathering peer attributes states various requirements on the quality of the solution. These quality requirements do not influence the functional requirements, but show which design goals should be aimed.

Scalability Scalability of the information management architecture, both in regard of the number of peers and the number of attributes is a key quality aspect for information and efficiency management in peer-to-peer systems.

Flexibility The query and information update processing should adapt to the read and write patterns of the corresponding attributes in order to minimize the traffic overhead.

Robustness The information management architecture should cope with churn, failing peers and apply mechanisms to overcome peer failure.

Load Balancing and Load Maximum The load for maintaining the information management architecture should be balanced on all peers participating in the P2P network.

Heterogeneity Individual peer capabilities should be taken account, by allowing each peer to specify a maximum load to tolerate. With this, stronger peers can contribute more, and weaker peers are not overloaded.

Assumptions

We state following assumptions for the information and efficiency management layer. We assume that the P2P host maintains a structured overlay which is compliant to the Key-Based Routing (KBR) specification [3]. An information management architecture which builds on an existing structured overlay needs utilizes this to send information and efficiency management layer specific messages using the function route

void route(key → K, msg → M, nodehandle → hint).

With the function *route* a node in the information management architecture is able to send a message to a node which is responsible for a specific key in the DHT (which may represent a role in the p2p network). We further assume that the DHT layer provides information on the keys a peer is responsible for in the DHT. A peer should know in a DHT, whether it is responsible for a specific key or not.

All peers in the structured overlay apply the information and efficiency management layer in their implementation and participate in the information management architecture by providing their specific attributes. In our solution we do not discuss security issues resulting from malicious or selfish peers, we assume in this first step protocol-compliant behavior of the peers.

Quality Measurment

The efficiency and profit of an individual information and efficiency management layer can be measured for an individual peer and for the whole system. Comparing the traffic costs and quality of the information for each functional layer individually gathering information with an dedicated information and efficiency management layer reveals the benefits for having a dedicated component.

3 Our Solution

3.1 Main Idea

As core of our information management architecture we define intervals (Domains) in the ID space each with a deterministically chosen responsible peer (Coordinator). Peers in the Domain address their information updates at their Coordinator using the overlay-independent Key Based Routing interface. Having a B-tree of Coordinators, information updates are passed periodically one level higher. In order to relieve the load on Coordinators, Support Peers are chosen from the corresponding Domain based on their capabilities. The system supports queries for a desired amount of peers fulfilling specific (quality)

criteria (e.g. *Give me 10 peers with at least 500kb/s upload bandwidth and 20Mb storage space*). Queries are stated at the responsible Coordinator of the peer and passed up the tree until one Coordinator can provide a valid result, while going up the tree, the considered Domain size increases. Further monitoring information is propagated upwards the tree and aggregated at each level in the tree.

3.2 Definitions

3.2.1 ID space

Object IDs and Peer IDs are elements in this continuous set of identifiers. The notation for the ID space is S_{ID}. Please note, we use the term ID and key synonymously.

3.2.2 Responsibility function

Let $p \in S_{ID}$ be a peer ID, then there exists a subset $S_p \subseteq S_{ID}$ so that peer p is responsible for all (object) IDs/keys in that set. Following counts

$$\forall p, q \in S_{ID} \ : \ p \neq q \ \rightarrow S_p \cap S_p = \{\} \qquad (1)$$

We define the responsibility function *res* as follows:

$$res: \ S_{ID} \rightarrow S_{ID} : o \rightarrow p \ with \ o \in S_p \qquad (2)$$

3.2.3 Domain

Continuous interval $D_.^i$ in the key space. Domains at the same level i in the tree do not overlap. Let $p \in S_{ID}$ and D_i^{ID} be a Domain sequence. Then following counts

$$\forall p \in S_{ID} \ : \ D_p^0 = S_{ID} \qquad (3)$$

$$\forall i \in \mathbb{N} \ \forall p \in S_{ID} \ : \ p \in D_i^p \qquad (4)$$

$$\forall i \in \mathbb{N} \ \forall p \in S_{ID} \ : \ D_p^{i+1} \subseteq D_p^i \qquad (5)$$

$$\exists k \in \mathbb{N} \ \forall i \in \mathbb{N} \ with \ i \geq k \ \forall p \in S_{ID} \ : \ D_p^{i+1} = D_p^i \qquad (6)$$

3.2.4 Key for Domain

A key in the Domain, whose correspondent peer defined as responsible for the Domain. The Domains build a b-tree structure. A Domain of level l (e.g. $[i_a, i_b]$) is partitioned in b (Sub-)Domains of level $l+1$ (e.g. $[i_a, i_1], [i_1+1, i_2], [i_2+1, i_3], ..., [i_{b-1}+1, i_b]$). Let K be the function mapping a Domain (subset of S_{ID}) to an ID in S_{ID}. Let K_p^i be the key of the Domain D_p^i, then following holds

$$K: \ \wp(S_{ID}) \rightarrow S_{ID} : \qquad (7)$$

$$K_p^i := min(D_p^i) + \frac{max(D_p^i) - min(D_p^i)}{2}$$

$$\forall p \in S_{ID} \ \forall i \in \mathbb{N} \ : \ K_p^i \in D_p^i \qquad (8)$$

3.2.5 Coordinator of a Domain

A peer responsible for the specific Domain. Let D_j^i be a Domain, then its Coordinator C_j^i is a peer and defined as

$$C_j^i \ = \ p \in S_{ID}, \ with \ p = res(K_j^i) \qquad (9)$$

This means, the Coordinator peer for the Domain D_j^i (which is in the i^{th} level, and contains the ID j) is defined as the peer which is responsible for the key K_j^i of the corresponding Domain.

3.3 Design decisions

3.3.1 Function of the Information And Efficiency Architecture

An information management architecture is required in a P2P system to gather, process and provide information of the peers. An important design issue is, how the peers want to benefit from the processed information. The structure of the query to an information architecture decides on its usability. We state that there are to key questions in today's and future P2P applications:

1. Which peers in the system can fulfill some specific task / role (optimization).
2. What is the status of the network (monitoring).

The first question is relevant for the function of a complex system as specific tasks has to be assigned (e.g. replication storing, multi-cast streaming, ...). The second question is relevant for the overlay service providers (OSP) (e.g. Skype) providers and Internet service providers (ISP) (e.g. AT&T). OSPs want to minimize the load on their supporting servers and to keep the network running. Future OSPs aim at providing complex functionality based on an architecture with unreliable peers. For this they need to have statistics on the P2P system.

We aim at both functionality goals, monitoring the whole system and providing a set of peers with desired attributes. Whereas monitoring allows for aggregation of information (e.g. addition of the number of peers in two halves of an ID range), the second function requires to keep the information of each individual peer, this information cannot be merged.

3.3.2 How to find a Coordinator

In order to send periodically information updates to the responsible peer (Coordinator) for a Domain, its peer ID has to be known. Two approaches exist: using stateless allocation to a specific peer (ID) (like in [4]) or dynamic assignment (like in [5]), which requires information exchanges for maintaining the architecture. Stateless solutions are mapping the ID of a peer to the ID of its corresponding Coordinator. This comes with no costs, but does not take the quality of the coordinating peer into account. It may be, that weak or overloaded peers are chosen to be Coordinators. On the other hand, choosing peers based on their characteristics as Coordinators results in a load-balanced information overlay, but also in additional costs for advertising the chosen Coordinators.

In our solution we use a stateless deterministic function for providing information on the Coordinator responsible for a peer ID. As the information management architecture is designed as an addon for current strucutured overlays, traffic overhead minimization is a crucial goal.

3.3.3 Information ID space for overlay independence

Each overlay comes with its own ID space. We introduce a general unifying ID space in order to build an information management architecture, which is independent of any specific (structured) overlay. The unified information ID space (S_{IID}) ranges from 0 to 1 and is parameterizable in its granularity. Any ID used in current structured overlays can be mapped to this interval. Furthermore the mapping can be optimized by putting stronger peers on more important positions.

Let S_{OID} be the overlay ID space, and $IDmap$ the function mapping from S_{OID} to S_{IID}, then the mapping function should retain convexity of identifier subsets. This means that the mapping function should be linear in relation to the responsibility function

$$\forall oid \in S_{OID} IDmap(resp(oid)) = resp(IDmap(oid)) \quad (10)$$

3.3.4 How to calculate the Domain Keys

One specific ID in a Domain is marking the responsibility for the Domain. In combination with the ID space mapping, the calculation of the Domain Key, can be used to load balance the architecture in advance, by putting stronger peers on more important positions. The Domain Key has to be chosen deterministically by any function fulfilling Equation 7 and 8. Domain Keys are chosen deterministically as the ID in the middle of a Domain interval. Optimization of the ID mapping and the Domain Key function is not considered in this paper, but is part of the future work.

3.4 Load balancing using Support Peers

Deterministically chosen Domain Keys may put weak peers into charge of being responsible for Domains. Choosing aiding Support Peers, that help the Coordinator of a Domain to receive the information updates and to answer the information queries, resolves the

limitation of a deterministic role assignment. Support Peers are chosen based on their qualities using an election function (see Subsection 3.5.7).

3.5 Protocols

3.5.1 How to find a Coordinator

Let p be the ID of the peer. Calculate Domain Keys K_i^p for the specific peer l levels deep, with

$$l = min(i \in \mathbb{N} \; with \; D_p^i \subseteq S_p) \tag{11}$$

The Coordinator C_p of peer p is then C_p

$$C_p = res(K_{l-1}^p) \tag{12}$$

Peers send update messages periodically to their Coordinator. Coordinators of level k (e.g. peer p) identify the Coordinators they have to propagate the updates to by identifying the node one level higher in the tree, which is the peer $res(K_{k-1}^{K_k^p})$.

3.5.2 Query - what and how

The information architecture provides the function to resolve queries of the type: Give me n peers fulfilling a set of requirements on the known peer attributes (e.g. a minimum storage space, a maximum load, ...). Queries contains a field identifying the requester, defining the number of requested peers and a list for requirements on peer attributes (and how they are connected: AND, OR, $\leq$, $\geq$). Peers address their queries to their responsible Coordinators. The Coordinator checks locally whether he has information about n peers fulfilling the desired requirements. Then it either replies with n peers fulfilling the criteria or it redirects the query one level higher in the tree. If no Coordinator in the tree can respond to the query, the root of the tree responds with the list of peers fulfilling the criteria (less than n). Please note, that peers do not have to agree on a set of valid attributes.

3.5.3 Monitoring

For monitoring purposes, peers send besides regular updates, containing their peer attributes, additional information which can be aggregated. These information has to be predefined in order to apply aggregation functions. All peers have to agree on a set of attributes that can be aggregated and is interesting for monitoring. Peers send then these information to their Coordinator, which aggregates them, and forwards the compressed information to its Coordinator one level higher. At the root of the tree, the monitoring information is complete and can be used. On lower levels, snapshots of the tree can be seen.

3.5.4 Joining the Information Management Architecture

To join the information network, peers send a regular update to their Coordinator. No specific join or keep-alive maintenance is required (as we rely on the *route*-functionality of the KBR compliant DHT). The tree is constructed from the bottom up.

3.5.5 Decreasing the height of the tree

Each Coordinator should be responsible for at least C_{Min} and at most C_{Max} peers (parameters may be peer specific). Being responsible for more than C_{Max} peers requires to keep up to many connections, whereas being responsible for less then C_{Min} peers leads to a high number of levels in the tree. A high tree results in many update hops until information of the peers reach the top of the tree, the information may get too old.

In order to decrease the height of the tree, Coordinators check upon receiving an update, whether the number of peers they know to be responsible for is between C_{Min} and C_{Max}. If a Coordinator C_p^i receives an update of peer p, and C_p^i is responsible for less peers than C_{Min}, then C_p^i advises peer p to send its next n updates to C_p^{i-1}. Coordinator C_p^{i-1} may advise peer p to send its updates to C_p^{i-2} and so on. Peer p may even maintain a history to determine which Coordinators are best to address the updates to, so that they neither redirect nor refuse these updates. Addressing of updates and queries is not strict, beginning at a deeper part of the tree only disburdens peers at higher levels of the tree, that are responsible for more information. If updates or

queries are addressed "too" deep or high in the tree, the information is anyways included and queries are resolved.

If a Coordinator of a Domain receives updates of more than C_{Max} peers of its Domain in a specific time interval, it contacts the Coordinators of its SubDomains (see Eq. 7) to ask them to take over the peers. Further the peers of the Domain are contacted to address the Coordinators of the Subdomains with their updates and queries.

3.5.6 How Supporting Peers support the Coordinator

The Coordinator of a Domain may decide that itself is incapable to burden the load of a Coordinator. Coordinators have to store the information of the peers they are responsible for, process information updates and react on queries. Support Peers may be chosen by the Coordinator to share the load. The Coordinator announces to the peers it is responsible for its Support Peers in a reactive manner. The peers, address then for a given time period their updates and queries to the Support Peers. The Coordinator and the Support Peers synchronize their information periodically in order to keep themselves up to date with the information.

3.5.7 Support Peer election algorithm

Each Coordinator appoints the best m Support candidates (SC) in its own Domain (according to some metric): SC_1 to SC_m. Support Peers for its own Domain are chosen from the peers $SC_{\frac{m}{2}+1}$ to SC_m. The information about the best $\frac{m}{2}$ Support Peer candidates (SC_1 to $SC_{\frac{m}{2}}$) is passed to one level higher, so that in this larger Domain more valuable candidates are available. The number of Support Peers is increased with a high number of queries and high churn, but decreased in a scenario with a lot of update and synchronization overhead. Following parameters are used to determine the required number of Support Peers per Domain: frequency of updates, frequency of queries, churn, Domain size, number Of peers in the Domain, tree level.

3.5.8 Information synchronization with the Support Peers

Regular synchronizations between the Coordinator and its Support Peers are necessary, as all of them receive updates from peers they are responsible for. These information has to be shared so that the Coordinator and its Support Peers can process queries properly. For load balancing, Support Peers have been introduced. For bounding the load on individual peers, peers may set a limit of load to accept, both Coordinators and Support Peers may do so. Information that can be aggregated is not influenced by this limitation, as it has to be processed and forwarded in any case. However, information of individual peers may be dropped. However, this is not critical, as the query type this information is used for, asks in general for a small number of peers fulfilling specific criteria. Keeping the information of thousands of peers enables the Support Peers or Coordinators still to provide valuable results.

3.5.9 Coping with failures of Coordinators

The failing of a Coordinator C_p^i is detected by its Support Peers and peers addressing update messages to this peer. As soon as a Support Peer identifies that the Coordinator failed it starts a lookup for the peer now being responsible for the Domain Key $res(K_{l-1}^p)$. This peer is then the new Coordinator and it is synchronized by the Support Peer. Peers do the same, by detecting the lost connection and identifying the new Coordinator with starting a lookup for the corresponding Domain Key.

4 Conclusion

In this paper we discussed motivation and requirements for building an information and efficiency management layer as dedicated module for peer-to-peer systems. This layer gathers, aggregates, analyzes and provides information from and to the peers in the P2P networks. The other functionality layers in the P2P system are disburdened from this load and focus on the question *what* information is needed for layer-specific optimizations and not *how* to obtain

this information. With our component the development of complex multi-layer P2P application can be eased and the efficiency of the system is increased.

We presented in this paper an architecture, applicable on structured P2P overlays, which enables interested peers and parties to monitor the state of the P2P network and further provides the functionality of finding a queried set of peers fulfilling a requested attribute state. With this, functional layers in the P2P application can query for peers fulfilling specific requirements, which are then addressed to fulfill a layer-specific role (e.g. storing replicas addressed by the replication layer).

In the architecture a peer identifies using a deterministic function to which peer (Coordinator) to send information updates (leveraging DHT-functionality). By this a virtual tree is build from bottom to top. Upon overload, Coordinators may choose Support Peers from the set of peers they receive information updates from in order to share the load. As the architecture builds on the route-functionality describe in the Key-based Routing Layer [3], the tree needs no further maintenance messages. Having only short term information transmitted in the tree, every peer may fail without the need for (information) recovery mechanisms.

Our solution is scalable (leveraging the underlying DHT), easy to deploy (simple add-on to existing DHTs), efficient (O(log N) hops per query and update) and proposes a valuable component in future's modular component-based P2P applications.

Acknowledgment

The first two authors of this paper are supported by the German Research Society (DFG) Research Group 733, "QuaP2P: Improvement of the Quality of Peer-to-Peer Systems" [6].

References

[1] K. Graffi *et al.*, "Peer-to-Peer-Forschung - Überblick und Herausforderungen," *it - Information Technology (Methods and Applications of Informatics and Information Technology)*, vol. 46, no. 3, 2007.

[2] BitTorrent, http://www.bittorrent.com.

[3] F. Dabek *et al.*, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Proc. of IPTPS '03*, 2003.

[4] Z. Zhang, S. Shi, and J. Zhu, "SOMO: Self-Organized Metadata Overlay for Resource Management in P2P DHT," in *Proc. of IPTPS '03*, vol. 2735. Springer, 2003, pp. 170–182.

[5] R. van Renesse and A. Bozdog, "Willow: DHT, aggregation, and publish/subscribe in one protocol," in *Proc. of IPTPS '04*. Springer, 2004, pp. 173–183.

[6] DFG Research Group 733, "QuaP2P: Improvement of the Quality of Peer-to-Peer Systems," http://www.quap2p.de.