A Peer-To-Peer-based Storage Platform for Storing Session Data in Internet Access Networks

Peter Danielis, Maik Gotzmann, Dirk Timmermann, University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany, Tel./Fax: +49 (381) 498-7272 / -1187251
Email: {peter.danielis;dirk.timmermann}@uni-rostock.de

Thomas Bahls, Daniel Duchow, Nokia Siemens Networks GmbH & Co. KG, Broadband Access Division
17489 Greifswald, Germany, Tel./Fax: +49 (3834) 555-642 / -602
Email: {thomas.bahls;daniel.duchow}@nsn.com

Abstract

Internet service providers (ISPs) have to store session data of their customers for operation, management, and control tasks. Thereby, each access node (AN) of an ISP's access network keeps track of session data (e.g., IP addresses, MAC addresses, and lease times of IP addresses) of all connected customers. Session data is highly volatile due to continuous changes. It has to be stored persistently as it is required for regular data forwarding and traffic filtering. In case of an AN's restart or crash, it needs to be reloaded. Today, session data is stored in an AN's flash memory, which is limited in its availability and rewritability and intended for other purposes.
Therefore, this paper proposes to organize ANs into a distributed hash table (DHT)-based Peer-to-Peer network to share their available RAM resources. Thereby, the DHT network serves as semi-permanent distributed memory for a structured redundant and interleaved storage of session data. In doing so, availability of session data is actually increased despite using RAM for data storage. After a restart or crash, an AN reloads session data by selectively reading required data from the DHT network.

1 Introduction

Internet service providers (ISPs) have to store session data of their customers as it is required for operation, management, and control tasks. Thereby, each access node (AN), e.g., a DSLAM (Digital Subscriber Line Access Multiplexer), a CMTS (Cable Modem Termination System), or a GPON OLT (Gigabit Passive Optical Network Optical Line Termination) of an ISP's access network keeps track of session data of all customers, which are connected to it via physical ports. Session data comprises, among other things, address information such as IP addresses, physical ports, MAC addresses, and lease times of IP addresses. The data is highly volatile due to continuous changes of its database requiring frequent memory accesses for rewriting. A customer connecting to the Internet automatically requests an IP address via DHCP. Session data has to be stored persistently and is accessed on a regular base for regular data forwarding as well as for traffic filters restricting traffic based on configured address information. In case of an AN's restart or crash, data has to be reloaded, i.e., session data recovery is needed. Today, flash memory is used to assure persistency of session data. However, this memory is limited in its availability and rewritability and is intended for other purposes such as the storage of configuration parameters of an AN.

In addition to flash memory, ANs have a certain *available* volatile RAM storage and computing capacity, which may be used at no extra costs. In contrast to flash memory, RAM memory can be rewritten almost unlimited. Therefore, this paper proposes to organize ANs into a distributed hash table (DHT)-based Peer-to-Peer (P2P) network to share their available RAM resources. Thereby, the DHT network serves as semi-permanent distributed memory for a structured storage of session data. Each AN contributes a part of its storage and computing capacity to the DHT network. After a restart or crash, an AN performs session data recovery by selectively reading required data from the DHT network so that persistency is assured.

Session data has to be available with a very high reliability, typically with a probability of at least 99.999 %, often refered to as "five nines". As data is stored in a distributed manner on all ANs, this availability has to be ensured by adequate mechanisms as ANs may fail or be unavailable. Therefore, data is stored redundantly. To keep this redundancy at a minimum while providing a high availability, erasure resilient codes are explored and compared to simple data replication [1].

Briefly summarized, the main contributions of this paper are the following:

- Options offered by P2P technology for designing a decentralized storage platform in the access network are investigated. P2P technology is used to realize semi-permanent storage of session data in volatile RAM memory. A P2P system for the realization of the P2P-based storage platform is chosen.
- Erasure resilient codes and simple data replication

are compared regarding data availability and storage overhead. An erasure resilient code is chosen and implemented.

- Architecture and functionality of the P2P-based storage platform (PSP) are described.

The remainder of this paper is organized as follows: Section 2 describes how to utilize P2P technology for realizing a distributed storage platform. Section 3 explains how to achieve high data availability for PSP. Section 4 presents the realization of PSP. Section 5 contains a comparison with related work. The paper concludes in Section 6.

2 Utilizing P2P Technology for a Decentralized Storage Platform

To avoid the utilization of flash memory for centralized session data storage, a decentralized storage solution is desirable where *available* resources can be used. Thereby, ANs, which are located in the access area (see **Figure 1**), shall serve as platform for such a solution. Each AN has a unique ID, which may be an alphanumeric string.

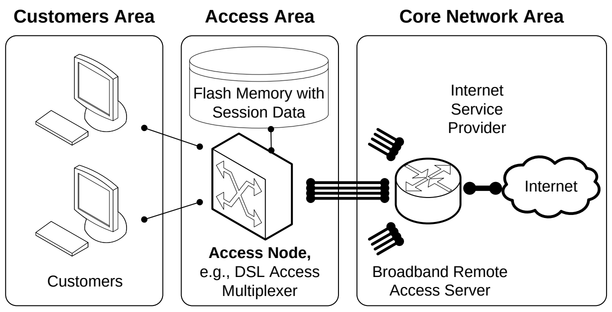
Customers Area
Access Area
Core Network Area
Flash Memory with Session Data
Internet Service Provider
Internet
Access Node, e.g., DSL Access Multiplexer
Customers
Broadband Remote Access Server

Figure 1: Access network containing ANs. Today, session data for connected customers is stored in the flash memory of each AN and requested via DHCP.

P2P technology offers many promising options to efficiently realize such a decentralized solution. Beyond its incriminated applications like, e.g., file sharing, P2P is a *new networking paradigm*. No clients and servers exist. All network nodes are peers, i.e., ANs are both client and server and form a logical overlay on top of the existing topology. Thereby, scalability and resilience are intrinsic features of P2P. There is no single point of failure in a P2P network. Therefore, it inherently provides high resilience towards DoS attacks and network faults. Moreover, it is a proven concept as it has been successfully applied in, e.g., recently developed P2P-based IPTV solutions like Zattoo and Joost [2, 3].
Each AN contributes to the P2P-based network application with its available resources. Thus, the P2P network itself is a *distributed storage resource* with ANs sharing their available memory. Thereby, each AN stores just a fragment of all session data. ANs share their computational power. Thus, the network is also a *distributed computing resource*.

A wide variety of P2P systems and alternatives exist, e.g., unstructured decentralized systems like Gnutella and structured DHT-based systems like Kademlia [4]. DHTs are chosen as they offer the best trade-off between search and storage complexity [4]. DHTs are *self-organizing*, i.e., no external maintenance is necessary. DHTs allow for deterministic lookups to avoid false negatives. Most DHT-based P2P networks support redundant data storage. This property contributes to an even higher resilience as session data is available on the DHT ring with a high probability even in case of one or more failing ANs.

2.1 Choosing a DHT-based P2P Protocol

There are manifold DHT-based P2P protocols such as, e.g., Chord, Pastry, and Kademlia [5, 6, 7]. Most of them have a logarithmic search complexity. Each DHT-based protocol provides for a routing table, which contains contacts for lookups. By reason of its flexibility of the routing table and high lookup performance as well as its analysis properties, the Kademlia protocol has been selected. Kademlia's advantageous analysis properties allow for an easy formal analysis of its worst-case behavior. Its flexibility of the routing table allows for a high lookup performance by using temporary contacts and keeps efforts for the maintenance at a minimum. For the realization of PSP, the Kademlia-based Kad protocol is chosen, which has been implemented and applied in the eMule client [8].

2.2 The Kademlia-based P2P Protocol Kad

To each peer, which is part of the Kad network, a hash value is assigned (e.g., 16 bytes calculated by MD5). Based on its hash value, a peer occupies a place in the hash address space. Based on an identifier of a fragment of session data (called chunk), a hash value from the same address space is calculated. This identifier may be the concatenation of an AN's ID and the "file name" of the chunk. A peer stores session data chunks with hash values similar to its own hash value. The distance needed to derive the similarity degree between two hash values is calculated by the XOR metric. Two hash values are similar if their distance is not greater then a defined search tolerance.
Peers always have sufficient information about other peers in their direct neighborhood. In distant regions, they always know at least one contact or more due to the flexibility of the routing table. This guarantees low latency for lookups as requests can be issued in parallel and delays caused by failing peers can be avoided.
An iterative algorithm is applied for lookups on the DHT ring. During a lookup, the searching peer first contacts peers, which are closest to the hash value of the requested part of session data, from its own routing table. In return, those peers answer the searching peer. Thereby, it learns new contacts until it knows contacts with sufficiently similar hash values. The lookup algorithm is simplified by using the XOR metric as it makes the Kad network more consistent and more efficient [8].

For the use with PSP, the Kad protocol has been modified to convey session data and extended with a deletion functionality to erase session data chunks.

3 Achieving High Data Availability

The most important and decisive criterion for the storage of data in general and session data in particular is to ensure high data availability [1]. Especially in case of distributed storage, appropriate measures have to be taken to reach a data availability P_d of typically at least 99.999 %.
On the one hand, a high value for P_d can be assured by data replication. On the other hand, data can be split into data chunks and redundancy is added in the form of interleaved coding chunks (erasure resilient coding). Data and its replicas or data chunks and coding chunks are saved on several storage systems, i.e., ANs in the authors' envisaged use case. Node availability P_n is constant as it can only be improved by the deployment of more reliable hardware. Therefore, data availability P_d can only be increased by adding redundancy. Consequently, the data volume to be stored is increased. This is considered as costs for higher data availability and shall be referred to as storage overhead factor S. Thereby, S denotes the ratio of original data volume + added redundancy divided by original data volume.
As storage capacity of ANs is limited, S should be as small as possible while providing high data availability P_d. To find out the best candidate for satisfying this requirement, simple data replication is compared to erasure resilient coding in terms of S and P_d in the following.

3.1 Simple Data Replication vs. Erasure Resilient Coding

In case of simple data replication (DR), data is copied from the node holding it to S - 1 other nodes.
Thereby, S is an integer value as complete data copies are generated. To restore the original data after a restart or crash, it has to be copied back from one of these S - 1 nodes. Consequently, P_d depends on P_n and S [1]:

$$P_{d,\,DR} \;=\; \sum_{i=1}^{S-1} \binom{S-1}{i} P_n^i (1-P_n)^{S-1-i} \qquad (1)$$

If using erasure resilient coding (ERC) data has to be split into m data chunks before storage. Then, k coding chunks are created containing information of all m data chunks. Thus, n = m + k chunks are generated, which represent the data. From *any* m of these n chunks, an efficient erasure resilient code is able to restore the original data. Thus, P_d depends on P_n, m, and n whereby n/m = S [1]:

$$P_{d,\,ERC} \;=\; \sum_{i=m}^{n} \binom{n}{i} P_n^i (1-P_n)^{n-i} \qquad (2)$$

Basically, both DR and ERC increase data availability. However, ERC provides a desired data availability P_d with much less storage overhead than DR. Let P_d be 99.999 % and a node be available with P_n = 90 %. Using DR, a value of 6 is obtained for S from Equation 1, i.e., data would have to be replicated five times. Contrary, ERC provides P_d = 99.999 % with a much lower S = 2 (with, e.g., n = 32 and m = 16). Thus, ERC decreases S by a factor of three. Consequently, ERC is used for ensuring high data availability rather than simple data replication.
ERC shows high complexity regarding the generation of coding chunks and for decoding them into a complete file if necessary. Anyway, ANs have sufficient computing power available for solving this computational task.

3.2 Reed-Solomon Codes

Reed-Solomon (RS) codes are chosen as erasure resilient code as they are very practical for achieving high availability for distributed storage [9].
One advantages is their capability of restoring data from *exactly* m *arbitrary* data chunks. Thus, RS codes optimally exploit added redundancy. Another advantage results from the property of RS codes to be systematic block codes. This means that data and coding chunks are ordered. Thereby, data chunks are followed by coding chunks. If m of n chunks are available and if these are data chunks, data can be restored by simple reassembly of them *without* decoding. This makes RS codes very efficient and saves computing time and power for decoding.

4 Kad-based Realization of PSP

PSP is realized by organizing an ISP's ANs into a Kad-based DHT. These ANs save session data chunks of other PSP nodes in a structured way (see **Figure 2**). Session data chunks include chunks session data has been split into and coding chunks generated by means of the RS code. As an AN represents a peer, i.e., a PSP node in the Kad network, a hash value is assigned to it, e.g., calculated from its IP address. Based on its hash value, an AN inserts itself on the DHT ring, which covers the whole hash address space.
Thereby, each AN

- stores its own session data in its RAM to be able to provide connected customers with IP addresses.
- is responsible for storing session data chunks with similar hash values calculated from a concatenation of AN ID and the "file name" of the session data chunks.

As proof of concept, a prototype has been developed, which emulates an AN with PSP functionality. It is implemented in C++ and currently runs under Windows.

4.1 Architecture of a PSP Node

The design of a PSP node is depicted in **Figure 3**. The main components are the memory with own session data (1), a memory with session data chunks of other PSP nodes (2), a

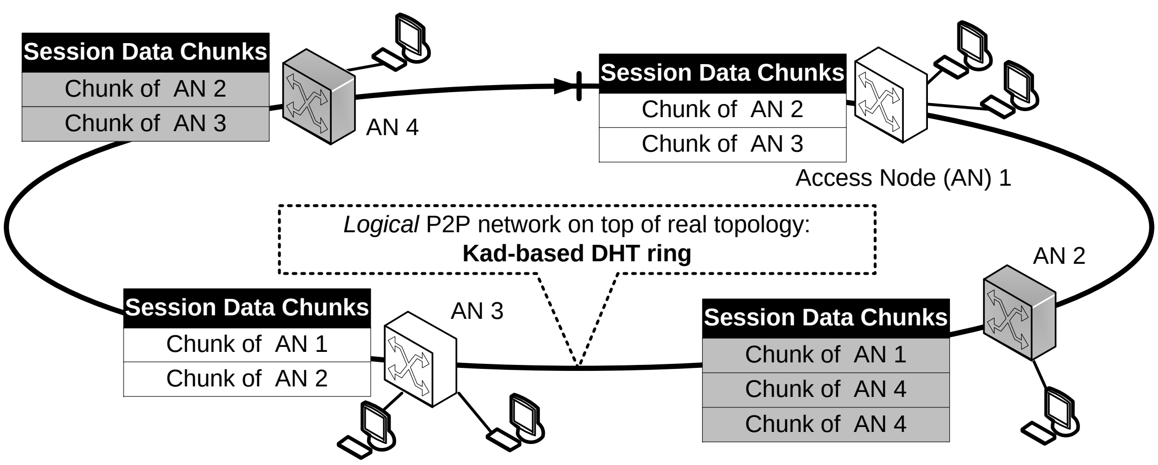

Figure 2: Kad-based DHT ring for structured storage of session data chunks.

routing table (3), the Kad block (4), and a module with functionality to process triggers (5). The memory with own session data provides session data for customers directly connected to the AN. The second memory stores session data chunks, which the PSP node is responsible for, based on its hash value. The routing table contains contact information about other PSP nodes to be able to communicate with them. The Kad block realizes the functionality of the Kad protocol. This includes bootstrapping, maintenance, and the execution of lookups and store and delete operations on the Kad-based DHT ring. It should be noted that at least one PSP node is assumed to be always available in the Kad network to be used for bootstrapping by joining nodes. Furthermore, the Kad block contains RS functionality to split and code session data into chunks and and vice versa to decode chunks into session data. These chunks are transfered via TCP in case of a lookup or a store operation on the Kad ring. If a PSP node shall execute an operation on the DHT ring, the Kad block is advised to contact PSP nodes from its routing table. A predefined number of PSP nodes closest to the hash value of a session data chunk is contacted in parallel.

4.2 PSP Node Activities

PSP node activities can be triggered by *internal* and *external* triggers. The node reacting on a trigger and executing an operation is called *triggering node*.

Internal triggers are caused by internal processes such as a change or the loss of the session data on an AN. External triggers can be received via an interfaces to the ISP, i.e., to an administrator (see **Figure 4**). Via this interface, lookups, store and delete, and configuration requests are issued and an (un-)successful processing is signaled. These requests can be either sent via packets or issued through an API between administrator and PSP node functionality. This interface allows for a flexible access to and configuration of an AN's session data by an administrative instance.

Via the interface to the Kad-based DHT ring, Kad requests and responses are received and sent, which are used for lookups and store and delete operations on the DHT ring. Packets for bootstrapping and maintenance are exchanged via this interface as well. Moreover, TCP data transfer of session data chunks is performed here.

Following, internal and external triggers and the interaction with other PSP nodes are described in detail.

4.2.1 Internal Triggers on a PSP Node

Internal triggers indicate when session data of an AN *has to* be read (lookup) or deleted from the DHT ring or stored.

If an AN is restarted after a crash or reset a lookup is automatically triggered on the Kad ring. Thereby, the restarted AN reads back its own session data by requesting appropriate chunks from other nodes. As session data is stored in the AN's RAM, it is lost after restart and therefore needs to be restored.

If an AN notices changes in its session data it initiates a delete operation on the DHT ring. Thereby, it deletes session data chunks created from the old session data. Changes can occur due to expired leases times of IP addresses or customers releasing their IP address (DHCP release) and administrative changes in the configuration of session data.

After an AN noticed changes in in its session data and deleted appropriate session data chunks from the DHT ring, session data chunks are created from the new session data and stored on the DHT ring.

4.2.2 External Triggers from an Administrator

External trigger requests may only be sent by an administrative instance of the ISP. These messages are intended to explicitly request (lookup), store, or delete session data chunks if an AN's session data has been modified or *shall* be restored. Furthermore, parameters of the RS code can be configured to achieve desired data availability P_d (configuration request).

If a PSP node receives a search request from the administrator a lookup on the DHT ring is initiated. After a successful lookup, a search response is sent back to the adminis-

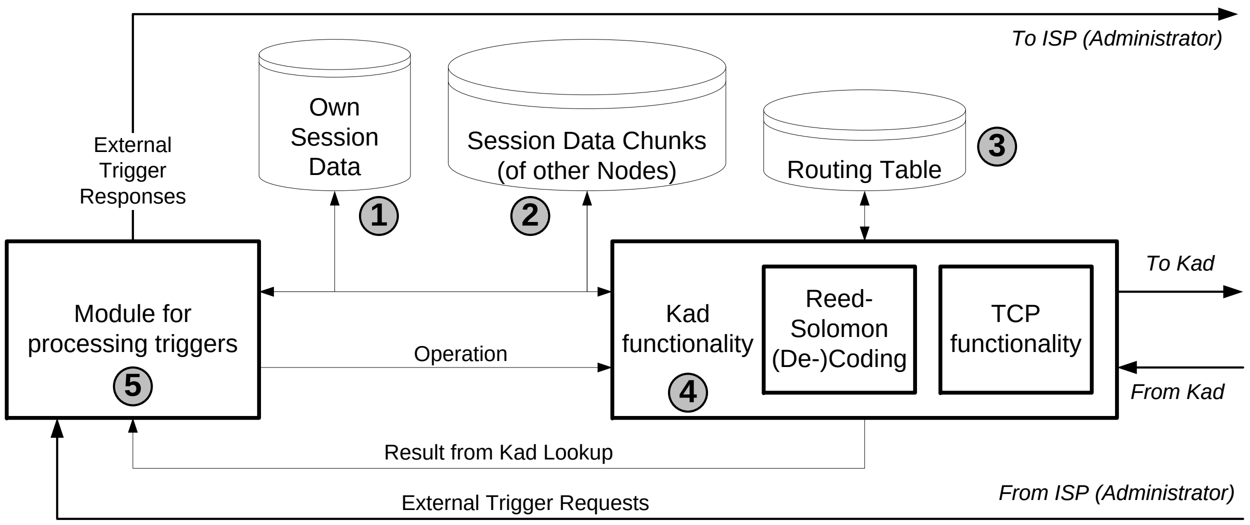

Figure 3: Block diagram of a PSP node.

trator signaling an (un-)successful processing of the search request.

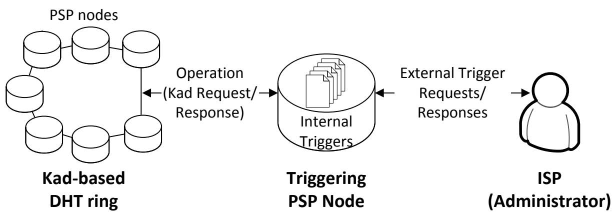

Figure 4: External and internal triggers leading to node activity and interactions with other PSP nodes.

If a node receives an update request, it splits its session data into chunks and generates coding chunks by means of the RS code. Then, a store operation on the DHT ring is initiated. Finally, the node responds with an update response.
On reception of a delete request, a delete operation on the DHT ring is initiated and a delete response is sent back. Moreover, an administrator can trigger the deletion of an AN's memory holding its own session data (delete memory request). In this case, session data chunks are *not* deleted from the DHT ring. It may be necessary to delete an AN's own session data to undo changes and then restore the original session data from the DHT ring.

4.2.3 Interactions with other PSP Nodes

Lookups on the DHT ring: Lookups are initiated by search requests from the administrator or internal triggers. During a lookup, nodes are contacted, which are closest to the hash value of the requested session data chunks. First, these contacts are taken from the routing table of the triggering node. In the course of the lookup, closer nodes are learnt from the responses of contacted nodes. Finally, nodes with a hash value similar to the hash value (within the search tolerance) of the requested session data chunks are found and contacted. Such a node tries to find session data chunks in its memory. If it succeeds it answers the triggering PSP node, which immediately terminates the lookup. If the requested session data chunks can not be found on any node, a timeout terminates the lookup. As RS codes are used to ensure high data availability P_d, the probability for *not* finding session data chunks on any PSP node is 1-P_d. Thus, timeouts terminating a lookup should rarely occur.
Store operations on the DHT ring: Store operations are initiated by update requests from the administrator or internal triggers. In the same way, a lookup is done, close nodes from the routing table of the triggering node are contacted first. When sufficiently close nodes have been learnt, they are advised to store the session data chunks. The store operation terminates if all session data chunks have been stored or through a timeout. A timeout may only occur if the store operation fails for any chunk. In this case, the operation is repeated until it is successful.
Delete operations on the DHT ring: Delete operations are initiated by update delete requests from the administrator or internal triggers. They work like lookups and store operations. Sufficiently close nodes are advised to delete the session data chunks. The delete operation terminates if all nodes have deleted the session data chunks or through a timeout. A timeout should only occur if one of the nodes holding a session data chunk to delete has gone offline.
Any of the operations contribute to increase the number of contacts the triggering node knows. Thereby, the Kad network becomes more redundant and more robust. Also, the lookup performance potentially increases as some iteration steps on the way to the target may be left out because closer contacts have been learnt.

4.3 Related Issues

Session data has memory requirements of some MBytes per AN. Therefore, the quantitative amount of memory that can be saved by using ERC compared to replicating data is moderate (given a desired data availability). However, vice versa a *significantly higher data availability* can be assured by ERC in any case (given a certain available memory capacity).

The utilization of the Kad network introduces some restrictions on PSP. As it bases on open source code of the eMule client, the exploitation of flaws by hackers is theoretically simplified. However, open source-based Apache web servers are market leader and Linux continuously increases its market shares, which argues for the high quality of open source code [10].

Communication in the Kad network introduces additional traffic into the network. But due to Kad's flexible routing table, traffic overhead through maintenance is minimal.

5 State-of-the-art in P2P-based Storage Platforms

[11, 12, 13] propose to use DHT-based solutions for distributed storage of data. In [11], globally distributed untrusted servers shall be used. In [12], the authors suggest a public data management system for Web applications. The solution in [13] focuses on a scalable storage platform for storing IP flow records. In [14], data sets are organized into redundancy groups. Thereby, the focus is on high availability of data in P2P-based applications with a high churn of nodes. In contrast to the author's approach, none of the cited works uses available resources of trusted reliable infrastructure to provide a storage platform for general data.

6 Conclusion

This paper proposes a P2P-based storage platform called PSP, which supersedes the use of expensive and rarely available flash memory for storing session data. Instead, available RAM storage and computing capacity of ANs is utilized, assuming some idle time or spare capacity being left in an average AN. In that way, the storage of session data is provided without additional costs for the ISP. Via the DHT-based P2P network Kad, ANs are organized into a logical P2P overlay on top of the existing topology. Thereby, they share their available memory and computing resources. As high scalability and resilience are intrinsic features of DHTs, PSP shows these features at no extra costs. As session data is stored interleaved by means of erasure resilient codes, all data can be restored even in case of a high number of unavailable ANs. Thereby, much less storage overhead is created than using simple data replication.

As proof of concept, a software prototype has been developed, which emulates an AN with PSP functionality. Currently, this prototype is ported to a Xilinx evaluation board.

Acknowledgement

The authors would like to thank the Broadband Access Division of Nokia Siemens Networks GmbH & Co. KG in Greifswald, Germany for their inspiration and continued support in this project. This work is partly granted by Nokia Siemens Networks.

References

[1] W.K. Lin et al., "Erasure Code Replication Revisited." IEEE P2P, 2004, pp. 90–97.

[2] "Joost - Free Online TV," 2007. [Online]. Available: http://www.joost.com/

[3] "Zattoo - TV to Go," 2007. [Online]. Available: http://zattoo.com/

[4] R. Steinmetz and K. Wehrle, *P2P Systems and Applications, Springer Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 2005.

[5] Ion Stoica et al., "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications." ACM SIGCOMM, 2001, pp. 149–160.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems." 18th IFIP/ACM International Conference on Distributed Systems Platforms, 2001, pp. 329–350.

[7] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric." IPTPS02, Cambridge, USA,, March 2002.

[8] R. Brunner, "A performance evaluation of the Kad-protocol," Master's thesis, University of Mannheim, Germany, November 2006.

[9] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.

[10] NETCRAFT, "August 2009 Web Server Survey," August 2009. [Online]. Available: http://news.netcraft.com/archives/web_server_survey.html

[11] J. Kubiatowicz et al., "OceanStore: An Architecture for Global-Scale Persistent Storage." ASPLOS, 2000, pp. 190–201.

[12] M. Karnstedt et al., "UniStore: Querying a DHT-based Universal Storage." International Conference on Data Engineering, 2007, pp. 1503–1504.

[13] C. Morariu et al., "DIPStorage: Distributed Storage of IP Flow Records." 16th IEEE LANMAN, 2008, pp. 108–113.

[14] Qin Xin et al., "Availability in Global Peer-To-Peer Storage Systems." Distributed Data and Structures, Proceedings in Informatics, 2004.