Extensive Analysis of a Kad-based Distributed Storage System for Session Data

Jan Skodzik, Peter Danielis, Vlado Altmann, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany
Tel./Fax: +49 (381) 498-7284 / -1187251
Email: {jan.skodzik;peter.danielis}@uni-rostock.de

***Abstract*—The P2P-based system for the distributed storing of session data of Internet service providers' access nodes called P2P-based storage platform is presented. Session data is continuously changing due to customers connected to access nodes, i.e., it is highly volatile. However, it has to be stored persistently as it is required for data forwarding, traffic filtering, and deriving statistics. Failing access nodes must be able to restore their data after reentering the network. Today, the session data is stored in the access nodes' flash memory, which is limited in life time and size and intended for other purposes.**

Contrary, the P2P-based storage platform allows to store session data in the access nodes' available RAMs connected by the distributed hash table-based P2P network Kad. A Kad network consisting of 8,000 access nodes for distributed storage of session data is setup for simulation. The simulation results show the traffic overhead to be minimal, linear scalability, and prove the high availability of the stored session data.

***Index Terms*—P2P, Kad, Distributed Data Storing, Simulation, Scalability.**

I. Introduction

Session data (SD) is created at the access node (AN) the customer is connected to via a physical port. This SD provides information such as IP addresses, physical ports, MAC addresses, and lease times of IP addresses. Internet service providers (ISPs) must store the SD as it is required for operational, administrative, and control tasks. SD changes frequently due to the dynamic behavior of the customers connected to the AN. Therefore, the memory for storing the data is accessed frequently for rewriting.

A customer who connects to the Internet automatically requests an IP address via the Dynamic Host Configuration Protocol (DHCP). In addition, SD is permanently stored and accessed on a regular basis as it is needed for data forwarding and the configuration of the session filter of an AN. Session filters block traffic from non-configured participants, i.e., customers who have not requested an IP address. In the case of the rebooting or crash of an AN, data must be reloaded, i.e., the recovery of SD is necessary. The session filter is reconfigured after the restoring of SD. Only after this recovery, the customer has Internet access again. A restart of an AN happens 2-3 times a week in the worst case, on average every 4 weeks, and at best once a year [1]. The ANs' flash memories are used in order to ensure the persistence of SD even in the worst case. However, flash memory is limited in its availability and the number of write cycles and is intended for other purposes like storing configuration parameters of the AN. Flash memory typically has a durability of 10,000 write cycles. An update of SD happens every 15 min to 1 h so a write access becomes necessary very often. Thus, the flash memory could already fail after 104 days of being active [1].

Instead of flash memory, this paper proposes to use the ANs' available volatile random access memory (RAM) and computation power. Unlike flash memory, the RAM can be written almost an unlimited number of times. A distributed hash table (DHT)-based Kad network allows the system to connect and share the RAM of all ANs. Thereby, the Kad network works as semi-permanent distributed memory for a structured storage of the SD. Each AN shares its memory and computation capacity with the participants in the Kad network. After the restart or shutdown of an AN, the recovery of the SD is performed by obtaining the necessary data from the Kad network. SD must be available at a very high reliability. Typically, the availability must have a probability of 99.999%. Since data is stored in a distributed manner in the ANs' RAM, the availability must be ensured with appropriate mechanisms as ANs can fail. The redundancy is kept low while ensuring high data availability by using erasure resilient codes (ERCs) [2]. Thus, the persistence of data is ensured.

The P2P-based storage platform (PSP) has been successfully implemented as software prototype running on the ANs as they provide sufficient resources. An average AN like, e.g., Freescale Semiconductor's PowerQUICC II Pro has 40% of free computing resources available. Of its 1 Gbyte RAM capacity, 400 Mbyte are available [1]. However, it is hardly possible to build a test setup with several thousands of nodes [3]. Therefore, a simulation model has been implemented to investigate the availability of the data, scalability of the P2P network, and the network utilization. The simulation model emulates the realistic behavior of the nodes with failure events and a trace of generated traffic by PSP due to exchanged data in the P2P network. Below, the following main contributions are briefly described:

- Briefly description of PSP.

- Determination of realistic environment parameters for the simulation.
- Evaluation of simulation results regarding data availability, scalability, and traffic overhead.

The remainder of this paper is organized as follows: Section II contains a comparison of the proposed approach with related work. Section III gives a brief description of PSP. The necessary simulation parameters are derived in Section IV. Section V evaluates simulation results before the paper concludes in Section VI.

II. Related Work

[4]–[9] propose to use DHT-based solutions for the distributed storage of data. In [4], globally distributed untrustworthy servers shall be used. To ensure trustworthiness, special measures have to be taken. In contrast, PSP renounces these measures since ANs represent trusted entities.

Ribeiro et al. [5] present a DHT-based platform with a low peer availability caused by high churn to realize persistent data storage. The lookup complexity in this approach is $O(N/log_2(N))$ hops for each lookup. PSP takes advantage of the Kad network since the lookup complexity of Kad is $O(log_{2^b}(N))$, where b is the number of bits of the node ID (hash value), which can be skipped with each lookup step.

b is set to a value of 4 for Pastry and 5 for Kademlia in their original implementations. However, the average value for b in Kad is 6.98 [10]. Therefore, Kad is optimal for PSP due to its lookup performance [10]–[12].

In [6], the authors suggest a public data management system for Web applications. The main focus is on data and service integration and the enabling of functionality to process data in the databases. In particular, the focus is on efficient database queries. PSP focuses on enabling the automatic distribution and collection of data without the need for complex requests. Therefore, PSP can save additional computational effort.

This also applies to the approach in [7], which focuses on a Pastry-based scalable storage platform for the storage of recorded IP data streams. This approach mainly focuses on improving the performance of complex search operations for requests of the mentioned data streams. Druschel et al. [8] describe a global Internet-based storage system consisting of non-trusted nodes based on Pastry. To ensure high data availability, the authors use simple data replication. In PSP, Reed-Solomon (RS) codes are used to ensure high data availability with substantially less data redundancy. Additionally, PSP uses the Kad network rather than Pastry as done in [7] and [8]. Kademlia improves a design discrepancy of Pastry. The Pastry routing distance metric is not necessarily the actual numerical distance of the node IDs. Moreover, Pastry needs two routing phases, which reduces its lookup performance and requires a complicated formal analysis of the worst-case behavior. Contrary, Kademlia uses the XOR routing metric, which reduces this problem [11].

Toka et al. [13] deal with very large amounts of data with a size of several Gbytes in a network with a high peer churn rate. Additionally, the authors assume a network of heterogeneous peers so that scheduling becomes relevant. These additional scheduling algorithms have no significant influence if the nodes are homogeneous in terms of bandwidth and connectivity behavior like in PSP.

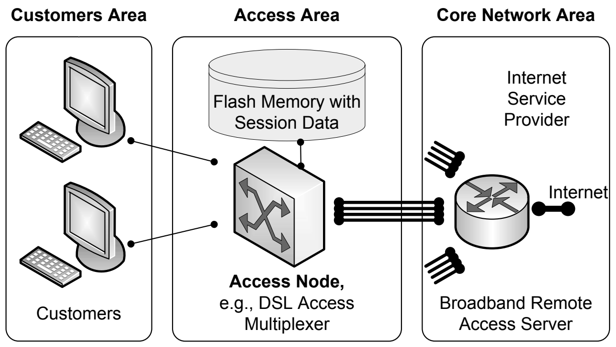
Customers Area
Access Area
Core Network Area
Flash Memory with Session Data
Internet Service Provider
Internet
Customers
Access Node, e.g., DSL Access Multiplexer
Broadband Remote Access Server

Fig. 1. Access network with PSP node.

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. This model is suited better for P2P-based file sharing applications with a high rate of migration of peers unlike the peers in PSP.

In contrast to the author's approach, none of the presented works uses available resources of trusted reliable infrastructure to provide a storage platform for general data. Ye et al. [9] provide a distributed storage platform of relatively trustworthy peers with a focus on ensuring actuality of data and data safety. PSP uses unique strings and an additional time to live (TTL) value to ensure actual data. The TTL value depends on the specific scenario and has to be modified in terms of the validity of the data. There is no further overhead and extra communication between the nodes in PSP. Ye et al. needs additional servers to enable the concept of one-hop routing. Therefore, each server stores the full routing table. PSP renounces the use of central instances like servers as single point of failures (SPoF) to achieve a fully scalable and fault-resistant network.

III. PSP in General

P2P technology in access networks: In Figure 1, a typical access network is depicted. Access networks comprise so-called ANs like IP Digital Subscriber Line Access Multiplexer (DSLAM). PSP is located on these ANs. A decentralized storage solution is desirable using available free resources instead of using the limited flash memory to store the SD in a centralized way. P2P technology has many beneficial properties to realize a decentralized system efficiently. Apart from being an application for file sharing, P2P is an innovative networking paradigm. All ANs are peers with client and server functionality and form a new logical overlay on top of the existing topology. Furthermore, high scalability and reliability are intrinsic properties of P2P networks, which can be used without additional costs. Each AN contributes its available resources to the decentralized storing platform PSP. Consequentially, the P2P network itself becomes a shared storage resource with the ANs sharing their available RAM resources. Each AN stores only a part of the entire SD. Additionally, the ANs share their computing power and the network becomes a distributed computing resource. PSP uses Kad as an implemented version of Kademlia. Kademlia uses a DHT to route the communication and data. It is a self-organizing DHT-based system, i.e., it does not require additional external maintenance. By using Kad, PSP does not show any SPoF so that high resistance against (Distributed) Denial of Service attacks (D)DoS and network failures is given [11]. Redundant data storage is supported by Kad per se - so that a higher reliability of the SD can be achieved. If one or more ANs fail the data is still available with a high probability. However, to achieve nearly 100% availability of SD in volatile RAM, additional measures are necessary.

A. Achieving data availability of 99,999 %

A crucial aspect of data storage in general and SD in particular is to ensure high data availability [2]. Especially, in the case of distributed data storage, appropriate measures have to be taken to guarantee a data availability of typically at least P_d = 99.999 % even if parts of the distributed memory system fail. The AN availability P_n is constant and can only be improved by using more reliable hardware. Therefore, increasing the data availability P_d is only possible by adding redundancy. However, this will result in a higher amount of data to be stored in the Kad network.

Data replication (DR) or ERCs are options to achieve a high availability of the SD. The memory overhead factor S represents the memory overhead regarding the original data D_{org} and the redundant data $D_{redundant}$ (see Formula 1).

$$S = \frac{D_{org} + D_{redundant}}{D_{redundant}} \tag{1}$$

Due to memory limitations in hardware, S should be kept at a minimum while ensuring a high data availability Pd. Therefore, P_d and S are compared using DR and ERCs. $P_{d,DR}$ if using DR is described in Formula 2. Complete SD copies have to be stored in the network without any encoding.

$$P_{d,DR} = \sum_{i=1}^{S-1} \binom{S-1}{i} P_n^i (1-P_n)^{S-1-i} \tag{2}$$

If ERCs are used, the data to be stored is split into m parts. Subsequently, the generation of k coded chunks containing information from all m chunks is carried out during an encoding process, which requires the time $T_{EncChunks}$. Overall, $n = m + k$ chunks are generated representing the data. With any m of these n chunks, an efficient ERC is able to restore the original data, even if some chunks are missing [2]. The availability of the data is represented by Formula 3.

$$P_{d,ERC} = \sum_{i=m}^{n} \binom{n}{i} P_n^i (1-P_n)^{n-i} \tag{3}$$

In case of DR, S equates to 6 if P_d is 99.999 % and P_n equals 90 % whereby this value for P_n would characterize very unstable nodes. Contrary, S equates to 2 if using ERC if n is 32 and m is 16. Thus, S can be reduced by the factor 3 by using ERCs. Therefore, PSP uses ERC, in particular RS codes, to realize a high data availability because of the higher efficiency in terms of the overhead. RS codes have a relatively high degree of complexity concerning the generation of coded chunks and its decoding into a complete file [15] [16]. However, ANs have enough available computing capacity to solve the computational task.

Another advantage arises from the fact that RS codes are systematic block codes. This means that data and code chunks are ordered. Data chunks are followed by coding chunks. If all m data chunks are available the whole file can be restored by simple composition of these chunks, i.e. without decoding. Thus, RS codes have proven to be extremely efficient and save computing time and energy for the decoding, which takes $T_{DecodeChunks}$.

B. Kad-based realization of PSP

PSP is realized by connecting the ANs of an ISP with a Kad network. The Kad protocol is extended to enable the transfer and deletion of data chunks. The ANs accomplish a structured storing of the SD and coded data chunks of other PSP nodes. Additionally, every AN becomes a peer respectively a PSP node and is assigned a hash value. The hash values can be created, e.g., from the IP address. The ANs are placed in the Kad network depending on their hash values. Each AN is responsible for storing its own SD in the RAM to provide all connected customers with IP addresses using the AN's DHCP functionality. Furthermore, the AN must store chunks of sessions data of other ANs. A node has to store chunks if the hash value of a file name combined with the AN ID is similar to the node hash value. A detailed description of the node architecture supporting DHCP and Kad functionality is given in [3].

C. PSP node interactions

PSP node activities can be triggered by internal or external events.

Internal events: They happen if the AN needs to perform one of the three operations read, delete, or store of its own SD in the Kad network. An AN needs to read its SD from the Kad network if it has failed before. Finding another AN is performed with $O(log_{2^b}(N)) * T_{Hop}$, where T_{Hop} is the round-trip time (RTT) between two ANs and $O(log_{2^b})$ denotes the necessary number of hops to the targets. Deleting is a basic operation, which is necessary to remove SD that have become invalid. If an AN detects the change of its SD it deletes the old SD. Changes happen due to DHCP operations or administrative commands. If the AN sends DHCP acknowledgments or a client sends a DHCP release packet the SD is changed and needs to be updated. Additionally, administrative changes in the SD result into the necessity of updating the SD. Subsequently, after having removed the old SD the node stores the new data in the Kad network, which is the third possible operation and requires the time $T_{TransData}$ to transmit the data.

$$\text{rect}(t) = \sqcap(t) = \begin{cases} 8 & \text{if } n * 120min - 25min < |t| \leq n * 120min + 25min \\ 30 & \text{if } n * 120min + 25min < |t| \leq (n+1) * 120min - 25min \end{cases} \quad n = 0, 1...\infty \tag{4}$$

External events: They occur if an administrative instance of an ISP sends commands to change SD or PSP parameters. PSP offers the possibility to directly execute operations on the AN independently of the DHCP functionality. It is possible to give the following commands:

- Storing of the actual SD.
- Deletion of an AN's own stored SD.
- Deletion of distributed SD.
- Sending of requested data parts to administrative instance.
- Modification of RS parameters.

These operations allow the administrative instance to have full control over the behavior of PSP in terms of data reliability and stored data. If data is removed, stored, or searched PSP uses the lookup operation of the Kad protocol to get in contact with responsible nodes. To enable the search and the transmission of data, PSP integrates extra search objects into the Kad protocol. For communications, UDP packets are used. The data transmission between nodes is carried out with TCP, which allows a reliable data transfer.

IV. Simulation setup

By means of simulation, the performance, scalability, and utilization of the connections by PSP are investigated. Thereby, the high availability and reliability of the SD in the Kad network are proven. To show the behavior of thousands of nodes, a simulation model based on the functionality of the working prototype of PSP has been developed [3]. A discrete C++ simulator with a time resolution of one second has been developed to enable the fast simulation of one year with several thousands of nodes. To speed up simulation, the simulation is purely functional and therefore independent of the network topology. It is able to measure the generated data volume, exchanged data chunks per second, and to validate the functionality of the PSP system. As network size for the simulation, we chose the backbone network of German Telecom, which consists of about 8,000 main distribution frames containing the DSL nodes, i.e., the ANs [17]. All ANs are connected with each other with a bandwidth of 10 GBit/s.

There are two operations an AN needs to execute. First, it stores the data in the Kad network. Second, the AN wants to get its own data back from the network.

A. Storing session data in the network

The AN needs to store its data in a distributed manner in the network. This happens if the SD on the AN changes. Changes happen if a "DHCP acknowledge" is transmitted from the DHCP server to a client or a "DHCP release" from the client to the DHCP server. To emulate realistic behavior of the DHCP traffic, the traffic of a university DHCP server was captured with the program Wireshark for one week and the "DHCP acknowledge" and "DHCP release" packets were filtered. About 200 customers were connected to the DHCP Server. The IP lease time is set to two hours.

The chronological sequence of "DHCP acknowledge" and "DHCP release" packets handled by the DHCP server can be described by the rectangular function in Formula 4. There is a phase with less "DHCP acknowledge" and "DHCP release" packets, which is called "Low Phase" lasting for the time T_{Low} of 50 min. Afterwards, a "High Phase" is characterized by a higher number of "DHCP acknowledge" and "DHCP release" packets for the time T_{High} of 70 min. The "Low Phase" and "High Phase" are repeated periodically. In Table I, the parameters to characterize the DHCP server behavior are listed in summary.

Parameter		Value
Low Phase		50 min
High Phase		70 min
#DHCP ack./releases per 5 min	Low	8
	High	30

TABLE I
Parameters of periodic rectangular function to emulate the behavior of the DHCP server

From the behavior of the DHCP server, the times for the storing of the SD in the Kad network can be derived. Every AN in the simulation starts at a random moment within the first 120 min of the simulation. Within the "Low Phase", an AN distributes its SD every 30 min and during the "High Phase" every 10 min as SD changes more frequently. These time intervals of SD storing in the Kad network correspond to the time interval of 15 min up to an hour for storing the SD in the flash memory [1].

Several parameters have to be determined to calculate the time T_{Store} for encoding and storing a SD in the Kad network (see Formula 5).

$$T_{Store} = T_{EncChunks} + T_{Hop} * \lceil Hops_{Avg} \rceil + T_{TransData} \tag{5}$$

The average time $T_{EncChunks}$ to encode SD depends on the SD size. Each AN with its DHCP functionality contains a SD volume of 4 MB. When storing the data in the network, the data is split by means of the RS codes. The parameter m and k are set to 16. This leads to 32 chunks with a size of 256 KB each resulting in 8 MB altogether. The time to encode k chunks and to create m chunks depends on the computing power of the AN hardware. An average AN achieves 1,500 - 3,000 Dhrystone MIPS of computing power when using a PowerQuicc II Pro or PowerQuicc III as typical processors [18]. $T_{EncChunks}$ is determined with a comparable PC and an average value of

7.125 s is achieved. Furthermore, it is important to know the time needed to store the chunks in the Kad network. The time for the lookup process to find a responsible node is given by the average number of hops. The number of hops for a lookup in Kad is specified by Formula 6. N is the total number of nodes in the test scenario and varies between 1,000 up to 8,000, which is the size of the German Telecom backbone. The number of nodes has been varied to show the scalability of PSP. The parameter b denotes the number of bits, which can be skipped with each lookup step and is set to 6. Therefore, the average number of hops for 8,000 nodes equates to $Hops_{Avg} = 2.16$ and is set to $\lceil Hops_{Avg} \rceil = 3$ for the worst case. Additionally, it is necessary to know the RTT of one Hop T_{Hop}. As typical RTT value in Germany, a value of 50 ms is assumed.

$$Hops_{Avg} = log_{2^b}(N) \tag{6}$$

The time for data transfer is described by Formula 7.

$$T_{TransData} = \frac{Data\,volume}{Bandwidth} \tag{7}$$

The bandwidth in the simulation is set to 10 GBits/s and the data size is set to 8 MB. This results in a transfer time $T_{Transdata}$ of 6.250 ms. In summary, the total time T_{Store} to store a data set in the Kad network is 7.281 s.

B. Restoring session data from the network

An AN needs to restore its SD from the Kad network if it is reset after a failure. The time consumption T_{Reset} of a reset procedure ranges from 10 to 15 minutes and is set to 15 min for the worst case. During the reset procedure, no interaction with the Kad network is performed. In addition to the practical worst, average, and best case behavior of the nodes, some additional theoretical failure profiles have been defined to show the scalability and performance of PSP. The number of needed resets after a failure of an AN and corresponding P_N values are given in Table II.

FAILURE PROFILE	YEARLY RESETS	P_N
Worst	130	99.629
Average	13	99.963
Best	1	99.997
Additional 1	50	99.857
Additional 2	100	99.715
Additional 3	250	99.267
Additional 4	365	98.958
Additional 5	500	98.573

TABLE II
FAILURE PROFILES OF THE ANS.

The time T_{Rest} for restoring the SD from the Kad network after a reset is described by Formula 8.

$$T_{Rest} = T_{DecodeChunks} + T_{Hop} * \lceil Hops_{Avg} \rceil + T_{TransData} \tag{8}$$

$Hops$ and T_{Hop} have the same values like when storing the data. $T_{Transdata}$ is only half as big as when storing SD as

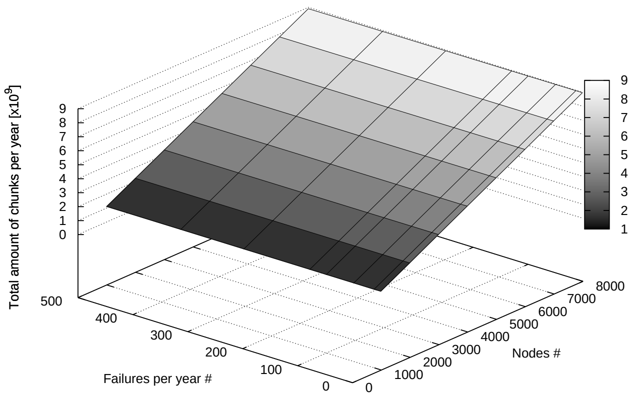

Fig. 2. Total amount of exchanged chunks per year.

only 16 chunks are necessary to restore the SD. Therefore, PSP cancels further requests for chunks as soon as 16 chunks available. The time for decoding the collected chunks varies with the available chunks. The maximum time has been determined to simulate the worst case behavior. In the worst case, it is assumed that PSP got 16 *coded* chunks. This results into a maximum time $T_{DecodeChunks}$ of 18.760 s. In summary, the time T_{Rest} equals 18.913 s for an AN, which reenters the Kad network to restore its SD.

As the simulator time resolution is set to one second, T_{Store} is rounded to 8 s and T_{Rest} is rounded to 19 s. In summary, the simulation parameters are given in Table III.

PARAMETER	VALUE	DESCRIPTION
N	1,000-8,000	Number of nodes
$Days$	365	Number of days
T_{Store}	8 s	Time to store SD
T_{Rest}	19 s	Time to restore SD
T_{Low}	30 min	Period of data storing in low phase
T_{High}	10 min	Period of data storing in high phase
T_{Reset}	15 min	Reset time of failed node

TABLE III
PARAMETER VALUES DURING SIMULATION.

V. EVALUATION OF THE RESULTS

Linear scalability: The exchanged SD volume is given in Figure 2. As apparent, the amount of transferred chunks per year linearly scales with the number of nodes N in the network. If we assume 8,000 nodes working with average failure profile, the average rate for each node equates to 9.1 Kbyte/s, which is a not a significant traffic overhead. This results in a capacity utilization of 0.0069 ‰ for the 10 Gbit/s connection, which does not impact the functionality of the ANs. Additionally, the number of transmitted chunks slightly decreases with an increasing number of failures per year. This happens due to inactivity of failed node, which do not store their SD in the Kad network until they are active again. This behavior is exemplary displayed in Figure 3 for the network sizes of 8,000, 7,000, and 6,000 nodes.

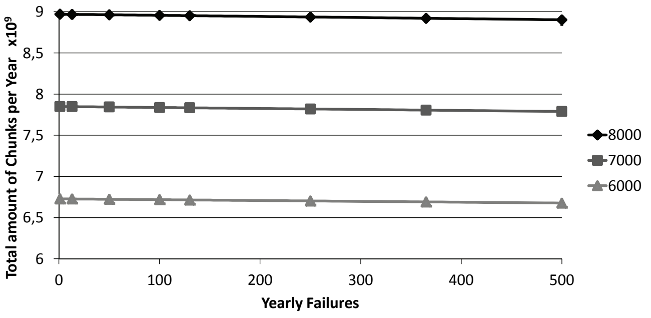

Fig. 3. Total amount of chunks per year.

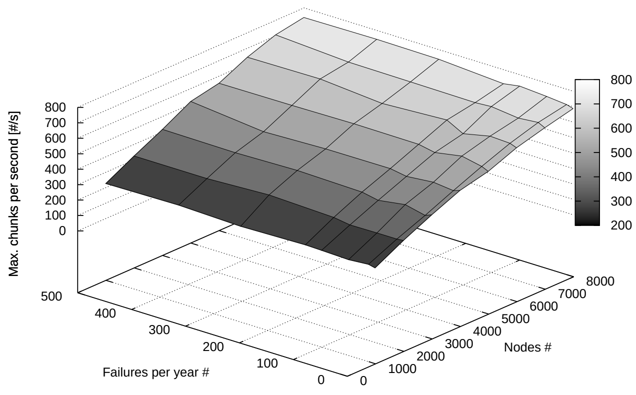

Fig. 4. Peak number of chunks per second.

Low traffic overhead: Another important value is the highest chunk number exchanged per second, which is an indicator for the maximum network load in the system. These results are depicted in Figure 4. The maximum number of chunks per second directly depends on the number of nodes N in the network. The highest number of exchanged chunks per second for 8,000 nodes running with average failure profile is 701 per second. This results in a data volume of 175,25 MByte/s in the whole Kad network. With respect to the high number of nodes, this is a relatively small data volume.

Permanent data availability: During the simulations, it never happens that PSP is unable to recover the SD from the P2P network due to high availability. This could also be guaranteed with a theoretical failure profile, which generates significant more failures than the practical worst case. The results of the simulation underline the applicability of PSP for large scale networks.

VI. Conclusion

An approach called PSP is presented to share the volatile RAM resources of ANs to form a distributed data storage with high resilience by using a Kad network. Free resources on the ANs offer the the opportunity to run PSP directly on the ANs. To characterize PSP in terms of scalability, traffic overhead, and high availability of SD, a simulation model of the PSP node functionality has been developed to enable the simulation of a large scale network. An extensive simulation with 8,000 ANs shows a linear scalability in terms of generated data volume and maximum exchanged chunks per seconds. The generated traffic volume occupies only 0.0069 ‰ of the 10 Gbit/s connections of the AN and therefore does not impact the AN functionality. Additionally, every failing PSP node was able to restore its SD during a simulation time of one year, even with a higher yearly theoretical failure rate than the specified practical worst case. Summarized, PSP offers a DHT-based distributed storing system with high availability and reliability of SD in large scale networks.

Prospectively, investigations on how to ensure a real-time behavior for data exchange in PSP will be carried out.

References

[1] M. Ninnemann, “Freescale semiconductor powerquicc ii pro - performance and utilization,” Broadband Access Division. Former Nokia Siemens Networks GmbH & Co. KG, November 2011.

[2] W. L. et al., “Erasure code replication revisited.” IEEE P2P, 2004, pp. 90–97.

[3] P. Danielis, M. Gotzmann, D. Timmermann, T. Bahls, and D. Duchow, “A peer-to-peer-based storage platform for storing session data in internet access networks,” *Telecommunications: The Infrastructure for the 21st Century (WTC), 2010*, pp. 1 –6, sept. 2010.

[4] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, “Oceanstore: an architecture for global-scale persistent storage,” *SIGPLAN Not.*, vol. 35, no. 11, pp. 190–201, 2000.

[5] H. Ribeiro and E. Anceaume, “Datacube: A p2p persistent data storage architecture based on hybrid redundancy schema,” in *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, feb. 2010, pp. 302 –306.

[6] M. Karnstedt, K.-U. Sattler, M. Richtarsky, J. Muller, M. Hauswirth, R. Schmidt, and R. John, “Unistore: Querying a dht-based universal storage,” in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, april 2007, pp. 1503 –1504.

[7] C. Morariu, T. Kramis, and B. Stiller, “Dipstorage: Distributed storage of ip flow records,” in *Local and Metropolitan Area Networks, 2008. LANMAN 2008. 16th IEEE Workshop on*, sept. 2008, pp. 108 –113.

[8] P. Druschel and A. Rowstron, “Past: a large-scale, persistent peer-to-peer storage utility,” in *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, may 2001, pp. 75 – 80.

[9] Y. Ye, I.-L. Yen, L. Xiao, and B. Thuraisingham, “Secure, highly available, and high performance peer-to-peer storage systems,” in *High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE*, dec. 2008, pp. 383 –391.

[10] R. Stutzbach, Daniel ; Rejaie, “Improving lookup performance over a widely-deployed dht.” INFOCOM, 2006, pp. 1–12.

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

[12] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the xor metric.” IPTPS, 2002.

[13] L. Toka, M. Dell’Amico, and P. Michiardi, “Data transfer scheduling for p2p storage,” in *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, 31 2011-sept. 2 2011, pp. 132 –141.

[14] Q. Xin, T. Schwarz, and E. L. Miller, “Availability in global peer-to-peer storage systems,” Dec. 2004.

[15] R. Rodrigues and B. Liskov, “High availability in dhts: Erasure coding vs. replication,” in *Fourth International Workshop on Peer-to-Peer Systems*, 2005.

[16] V. Huffman, W. C. ; Pless, “Fundamentals of error-correcting codes.” Cambridge University Press, 2003.

[17] S. Schweda, “Federal Administrative Court Rejects Competitors’ Claim for Access to Telekom’s Dark Fibre,” 2010. [Online]. Available: http://merlin.obs.coe.int/iris/2010/3/article14.en.html

[18] freescale semiconductors, “Integrated Communications Processors: MPC8360E PowerQUICC II Pro Family,” Tech. Rep., 2007.