Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems

Kalman Graffi, Dominik Stingl, Christian Gross, Hoang Nguyen, Aleksandra Kovacevic and Ralf Steinmetz
Technische Universität Darmstadt, Multimedia Communications Lab (KOM)
Rundeturmstr 10, 64283 Darmstadt, Germany. Email: {graffi,stingl,gross,sandra}@kom.tu-darmstadt.de

Abstract— **The peer-to-peer paradigm shows the potential to provide the same functionality and quality like client/server based systems, but with much lower costs. However, the resources, e.g. storage space, CPU power and online time, provided by the peers are unreliable due to churn. In order to enable churn resistant reliable services using the resources in p2p systems, we propose in this paper a distributed mechanism termed $P^3R^3O.KOM$. The mechanism allows to reserve, monitor and use resources provided by the unreliable p2p system and maintains long-term resource reservations through controlled redundant resource provision. Evaluation shows that using KAD measurements on the prediction of the lifetime of peers allows for 100% successful reservations under churn with very low traffic overhead. This approach marks a first step for the building of a reliable p2p-based SOA and future p2p-based clouds.**

I. Introduction

The distribution of computational efforts in form of server farms, service oriented architectures (SOA), GRIDS or cloud platforms gain importance in the presence of the increasing demand of users for resources and high-quality services for low costs. Resources like CPU power are utilized for distributed computations, bandwidth and storage space for quick distribution of data and online time for protocol purposes. A functional component using these resources to provide a functionality, we term *service*. The distribution of services (e.g. in a SOA or a GRID), allows for the provision and interaction of various service providers and consumers, creating a marketplace for services. In both concepts, SOA and GRID, services are provided by their owners and service guarantees are enforced through service level agreements and contracts. The upcoming cloud paradigm follows a similar approach by providing a platform for services, which is centrally managed and decentrally implemented. The operation models of a cloud can be classified in three layers on which resources and services are provided. The cloud may offer software as a service, i.e. applications and web-tools, platform as a service, i.e. service deployment and hosting, or infrastructure as a service, i.e. storage, computing and network capabilities. Essential characteristics of a cloud are the on-demand resource provisioning, access over the Internet, resource pooling, rapid elasticity to adapt to availability of resources and monitored (and billed) quality. These characteristics are similar to those of the peer-to-peer (p2p) paradigm. Combining both approaches may lead towards user-hosted clouds that are cheap to operate and to use, thus to the free exchange of resources as we envisioned in [1]. In this paper, we aim at p2p systems providing a reliable infrastructure as a service.

[1]This work has been partially funded by the DFG research group QuaP2P, EU SmoothIT and BMBF Premium Services

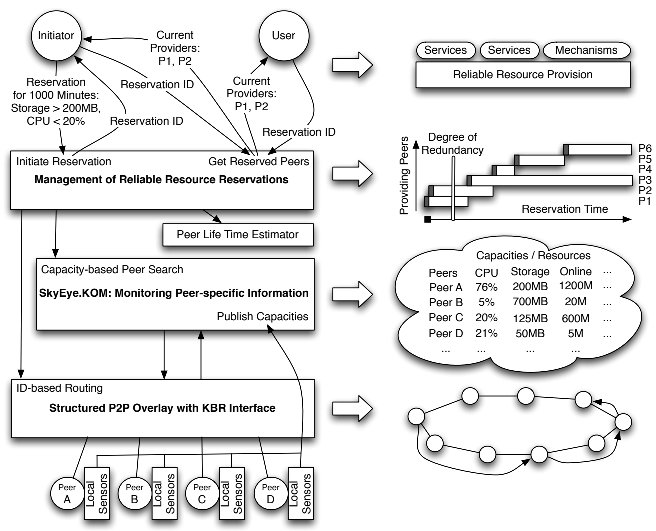

Initiator
Current Providers: P1, P2
User
Reservation for 1000 Minutes: Storage > 200MB, CPU < 20%
Reservation ID
Current Providers: P1, P2
Reservation ID
Reservation ID
Initiate Reservation
Get Reserved Peers
Management of Reliable Resource Reservations
Peer Life Time Estimator
Capacity-based Peer Search
SkyEye.KOM: Monitoring Peer-specific Information
Publish Capacities
ID-based Routing
Structured P2P Overlay with KBR Interface
Peer A
Peer B
Peer C
Peer D
Local Sensors
Services
Services
Mechanisms
Reliable Resource Provision
Degree of Redundancy
Providing Peers
Reservation Time
P6
P5
P4
P3
P2
P1
Capacities / Resources
Peers CPU Storage Online ...
Peer A 76% 200MB 1200M ...
Peer B 5% 700MB 20M ...
Peer C 20% 125MB 600M ...
Peer D 21% 50MB 5M ...
... ...

Fig. 1. Overview on Reliable Resource Reservation

The p2p paradigm, which became popular through file sharing and video streaming applications, promises to provide platforms for low costs by deploying tasks on the user devices. The users benefit from the free use of the infrastructure. In p2p systems, the participating peers organize themselves in a p2p overlay and act synchronously as consumer and provider of mechanisms and resources. P2P networks typically comprise thousands or millions of nodes with sufficient resources to provide an infrastructure for various kinds of services comparable to a SOA or a p2p cloud. However, these resources cannot be used directly for services as p2p systems are also characterized by the unreliability of the participating peers, thus the allocated resources are unreliable as well, as discussed in [2]. This issue is even more critical, as long-term resource reservations which last longer than a typical lifetime of a peer cannot be served reliably with a simple p2p overlay. For high quality services and applications based on the p2p paradigm, a reliable resource reservation is needed.

In this paper we motivate, present and evaluate a p2p protocol for reliable resource reservation and offering ($P^3R^3O.KOM$) especially with focus on long-term service level agreements for reliable resource reservation in unreliable p2p systems. In Fig. 1, the architectural interfaces for the management of resource reservations are depicted.

In Section II, we give the problem statement, functional requirements for a solution as well as the assumptions used in our solution. In Section III, we present our solution $P^3R^3O.KOM$ for reliable resource reservations in unreliable p2p systems and discuss its features and behavior under churn. Every resource reservation is managed by a small set of peers in the p2p overlay, which act as a self-monitoring service management group. This group picks peers providing the desired functionality using the functionality of capacity-based peer search in a quantity that the probability of all of the providing peers failing is very low. Through the redundancy of the resource allocation and the managed re-nomination of failed resource providers, the desired long-term reservation is met. The evaluation of the proposed mechanism is given in Section IV. There we show that with reasonable redundancy and low service maintenance overhead a reservation fulfillment ratio of 100% is reached. Section V discusses related work and in Section VI, we summarize the approach and the results.

II. Problem of Reliable Resource Reservation

One main characteristic of p2p systems is the unreliability of the peers. They come online and go offline autonomously at their will. However, the applications and services on top of the unreliable infrastructure require a reliable layer on which they can operate. Reliable resource reservation allows higher layers to reserve a number of peers with desired resource capacities for a dedicated time. The reservation time is expected to be longer than the lifetime of the peers, e.g. weeks.

Let $Cons_i$ be a constraint on the resources Res_i of the peers in terms of an upper or lower bound. A reliable resource reservation is defined by the resource specifications, reservation time and reliability:

- Resource constraints, $Cons_i$: The amount of resources to allocate, e.g. 500 MB storage space and an upload bandwidth of 128 KB/s.
- Reservation time, $RsvTime$: The time period for which the resources should be allocated, e.g. 200 days.
- Degree of Redundancy, $RsvDR$: A ratio of redundancy for the resource reservation which describes the probability that not all of the resource providers fail at the same time.

The goal of the mechanism for reliable resource reservation, $P^3R^3O.KOM$, is to provide the functionality to reserve and offer resources for a given time:

- $ReservationID$ $\leftarrow$ initiateReservation($Cons_1$, $Cons_2$,...,$Cons_i$, $RsvTime$, $RsvDR$) - reserves a peer set with given capacities for a time $RsvTime$ and a given redundancy degree $RsvDR$
- PeerID-list $\leftarrow$ getReservedPeers($ReservationID$) - returns the list of the peers providing the reserved resources at the current time

The mechanism, providing reliable resource reservations, takes care that the desired amount of capacities is constantly provided by the system during the reservation time (resource infrastructure as a service). Please note, that with a reliable resource reservation we do not require that the resources are provided by the same peers over the whole time or that the reservation initiator is online for the whole reservation time. This work marks a first step towards p2p clouds with resource reservations providing a reliable (resource) infrastructure to host services, however, further mechanisms such as usage accounting or service relocation are not incorporated. To support the relocation of services, we consider an explicit relocation time buffer while choosing new resource providing peers for the reservation. Based on the list of resource providing peers, a reliable resource reservation can be established.

In order to provide the desired functionality of reliable long-term resource reservation, we require the following mechanisms and functions. We assume a *structured p2p overlay* complying with the KBR interface [3]. This allows for ID-based routing in the overlay, a functionality which can be used to assign roles and data to given IDs in the p2p overlay. With the introduction of roles, resource reservations may be stored, managed and enforced in a deterministic and reliable manner.

One main component of the solution for reliable resource reservation is the functionality of *capacity-based peer search*. With this, a desired quantity of peers can be found in the p2p overlay with specific resource capacities. Let Res_i be a variable describing one specific resource capacity of a peer, e.g. CPU power, upload bandwidth capacity or its online time. Let $Res(p)$ be the set of attributes a peer p offers. And let $Cons_i$ be a constraint on Res_i in terms of an upper or lower bound, e.g. $Cons_1 : Res_1 < 50$, describing for example that the CPU load should not exceed 50%. Then following function is provided by the mechanism for capacity-based peer search:

- PeerID-list $\leftarrow$ capacity-based-peer-search(n, $Cons_1$, $Cons_2$,...,$Cons_i$) - query for n peers fulfilling a set of resource constraints

We assume the existence of such a mechanism and use for that our monitoring solution, SkyEye.KOM, as it was presented in [4] and [5]. It provides the desired functionality of capacity-based peer search and is applicable on any KBR compatible structured p2p overlay. Regarding the peer capacities that are monitored, we assume that the peers only offer those capacities that they are willing to share to the system and that they are actually contributing during the reservation process. Thus, variations of the offered resources over time should be considered (or filtered) by the peers themselves before they advertise them.

A *function to estimate the peer lifetime* based on its current lifetime is needed to prepare for peer failures and churn. The peer lifetime estimator is a function, that gives the probability $P_{fail}(p, t_{on}(p), t_R)$ for a peer p with current lifetime of $t_{on}(p)$ that the peer p will fail in the upcoming time period denoted by t_R. We introduce a short form for P_{fail}, as the second parameter is always depending on the first:

$$\begin{aligned} &t_{on}(p) : P \rightarrow T \\ &P_{fail}(p, t_{on}(p), t_R) : P \times T \times T \rightarrow [0,1] \\ &P_{fail}(p, t_R) : P \times T \rightarrow [0,1] := P_{fail}(p, t_{on}(p), t_R) \end{aligned} \tag{1}$$

For the determination of the the remaining lifetime of a peer, based on its current lifetime, various approaches exist. In [6]

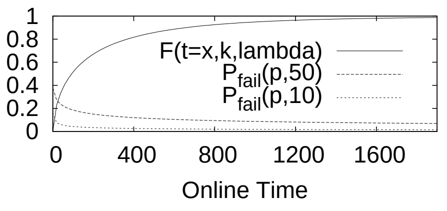


Fig. 2. Peer Lifetime Estimation

the authors present the results of having crawled contentiously the KAD network for about 6 months. The report regards the geographical distribution of peers, session times, peer availability and peer lifetime. The results in [6] show that the peer lifetime is Weibull distributed with following parameters:

$$mean = 266.5358,\ standard\ deviation = 671.5063,$$
$$scale = 169.5385,\ shape = 0.61511$$

One important result of the study is that the long peer uptime leads to a higher probability of staying online than a short uptime. As a result, the probability for staying online increases with every minute a peer stays online.

Let $F(t, k, \lambda)$ be the cumulative distribution function for the Weibull distribution with parameters k and λ. It gives for a given time span t the probability of being offline. Thus, we can calculate for all peers $p \in P$ the probability to fail in the next t_R minutes based on the their lifetime $t_{on}(p)$.

The probability $P_{fail}(p, t_R)$ that a peer p will go offline in the next t_R minutes can be calculated according to Eq. 1 as the difference of the offline probabilities at $t_{on}(p)$ and $t_{on}(p)+t_R$ in relation to the probability of having survived until $t_{on}(p)$:

$$P_{fail}(p, t_R) = \frac{F(t_{on}(p) + t_R, k, \lambda) - F(t_{on}(p), k, \lambda)}{1 - F(t_{on}(p), k, \lambda)} \quad (2)$$

In Fig. 2, we depict the distribution $F(t, k, \lambda)$ and $P_{fail}(p, t_R)$ with $t_R = 50m$ and $t_R = 10m$. With this assumptions in mind, next we present our solution for reliable long-term resource reservation.

III. $P^3R^3O.KOM$ - A P2P Protocol for Reliable Resource Reservation and Offering

The goal of the mechanism for reliable resource reservation is to provide the functionality to reserve and provide resources for a given reservation time. We propose $P^3R^3O.KOM$, a mechanism that offers reservations with specified resource constraints, a reservation time and a degree of redundancy.

The main challenge for reliable resource reservations is to reserve the desired amount of resources for a long time, which is expected to exceed the lifetime of each peer in the network and especially the lifetime of the reservation initiator. However, the resource provision must be enforced and reliably fulfilled. Churn is the main threat for long-term resource reservations. In order to provide the desired resources with a specific availability, resources are allocated and provided redundantly. In this case, individual peer failures are not critical as other peers in the resource providers' set remain active. For the resource reservation and provision process, two steps are considered:

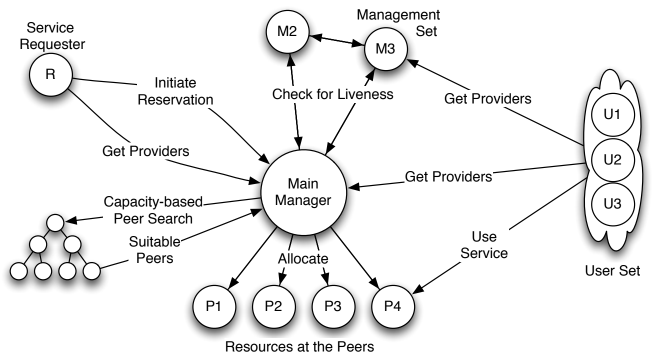


Fig. 3. Components Interaction Overview

- *Reservation Management*: A set of peers supervises the status of the resource provision, i.e. the set of resource providing peers, and adapts the redundancy level in this set in case of churn. For that, the reservation management peers predict the lifetime of the current resource providers and estimate the success ratio for at least one of the peers staying alive until the next round, t_R. These peers continuously manage the set of resource providing peers.
- *Reservation Enforcement*: A set of peers provides redundantly the desired resources for the given reservation. The peers all fulfill and contribute the resource requirements stated in the resource reservation.

An overview on the components of $P^3R^3O.KOM$ is given in Fig. 3. Next, we describe the protocol used to initiate, maintain and serve a resource reservation:

a) Setting up a Reservation: A peer in the network creates a reservation request describing the resource constraints $Cons_i$, reservation time $RsvTime$ and degree of redundancy $RsvDR$. It also derives locally a corresponding Reservation ID, e.g. the hash of the reservation parameters. This ID is mapped to the peer responsible for the specific ID in the structured p2p overlay. The resource reservation is stored on this peer in a reliable manner, e.g. in a replicated form. The peer responsible for the Reservation ID is the first peer in the management set for this reservation and termed *main manager*. The main manager additionally adds further information to the reservation request. It adds the reservation starting and finishing time and a list of variable information. The variable information contains information about the current reservation status, i.e. the probability to succeed, and a list of the current resource providers.

b) Maintaining the Management Set: Task of the management set is to reliably store and manage the status of the reservation. We add redundancy to the reservation management set for the case, that the main maintainer leaves the network. Then, the new peer responsible for the Reservation ID takes over the management role for the reservation. The main manager makes sure that it has a fixed set of backup peers, e.g. 2, that keep a local copy of the reservation status and update a new main manager in case of churn. In this step of the protocol, the main manager checks periodically the liveness of the backup peers or chooses new ones. New management

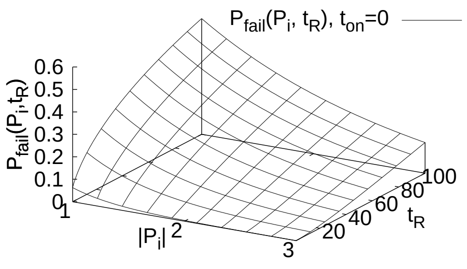


Fig. 4. Peer Failure Probability

peers are chosen using the functionality of capacity-based peer search and looking for peers that promise to stay long online, e.g. by already having a long online time. Also the backup peers check the liveness of the main manager and update the new peer responsible for the *Reservation ID* in case of churn.

c) Deriving the Success Probability for a Round: The second task of the management set is to provide the function of *PeerID-list* ← *getReservedPeers(ReservationID)*. The peer list contains peers providing resources for this specific reservation. The main manager performs following steps to ensure that enough peers are allocated, so that the desired success probability is reached. As a first step, the main manager checks the liveness and load of the resource providers in order to derive an up-to-date list of resource providers. Next, it calculates based on the current lifetime of the peers the probability that the current resource providers will be online until the next round. For that it uses the function to estimate the peer lifetime in order to retrieve a probability for liveness in the next round.

According to Eq. 2 we can calculate the probability of one single peer to fail in a given time span t_R, as depicted in Fig. 2. Now, we calculate the probability for a set of peers to fail. Let $P_i \subseteq P$ be a subset of the peers with cardinality i. The probability $P_{fail}(P_i, t_R)$ denotes for a set P_i the probability that all of the i peers fail in the next t_R minutes. The probability for successfully "surviving" a time span of t_R with the given set P_i of i resource providers is:

$$P_{succ}(P_i, t_R) = 1 - P_{fail}(P_i, t_R) = 1 - \prod_{p \in P_i} \{P_{fail}(p, t_R)\} \tag{3}$$

Thus, the mechanism for reliable resource reservation must aim at keeping the number of resource providing peers, i, so high, that the probability $P_{succ}(P_i, t_R)$ is above the degree of redundancy $RsvDR$. In Fig. 4, we depict $P_{fail}(P_i, t_R)$ with varying t_R and $|P_i|$ and fix $t_{on} = 0$.

d) Checking the Resource Provider Set: The success probability $P_{succ}(P_i, t_R)$ is matched to the degree of redundancy required in the resource reservation, $RsvDR$. In case of an endangered success probability, the main manager calculates the desired number of peers to add to the resource provider set. It picks suitable peers for the reservation in a quantity that pushes the expected success probability $P_{succ}(P_i, t_R)$ above the degree of redundancy $RsvDR$ specified in the reservation for the next time span t_R .

e) Identifying the Number of Missing Resource Providers: The number of resource providing peers must be sufficient to keep all the time at least one provider online even under churn. Next, we calculate the amount of providers for a reliable resource reservation. Let $Num_{exact} : [0, 1] \to \mathbb{N}$ be a function calculating for a given success probability $RsvDR$ the number of peers needed, so that $P_{succ}(P_{Num_{exact}}, t_R) = RsvDR$. As this number of peers is calculated exactly, in case of churn, the probability for all peers staying alive in the round drops below the probability $RsvDR$. In order to avoid this, we present two possible approaches for deriving the desired quantity of resource providing peers Num_{suff}, with $P_{succ}(P_{Num_{suff}}, t_R) \geq RsvDR_{new}$ even under churn.

f) Probability Buffer Approach (PBA): We pick the number of resource providers with a slightly increased degree of redundancy, i.e. by adding a probability buffer of $P_{buff} \in [0, 1]$. Thus, we pick an amount of peers, such that we surpass $RsvDR$ with a higher aliveness probability and use a buffer to address the case of churn:

$$Num^{PBA}_{suff} = Num_{exact}(RsvDR + P_{buff}(1 - RsvDR)) \tag{4}$$

Redundant Peer Approach (RPA): The redundant peer approach adds to Num_{exact} an additional amount of peers, N_{buff}. It assumes that the N_{buff} peers that contribute most to the resource reservation will leave and need to be replaced within a time t_R:

$$Num^{RPA}_{suff} = Num_{exact}(RsvDR) + N_{buff} \tag{5}$$

Picking new Resource Providers: In order to add new peers to the resource provider set, the main manager uses the function of capacity-based peer search of SkyEye.KOM. It defines the desired capacities and requests an amount of peers which push the probability of at least one peer staying alive until the next round above the desired threshold for the degree of redundancy. SkyEye.KOM provides a list of Peer IDs with suitable capacities. Please note, that any other mechanism providing the function of capacity-based peer search may be used. The main manager contacts the new peers in order to allocate their resources for the given reservation. The contacted peers allocate the requested resources for the reservation and acknowledge the allocation to the main manager. We also consider during the picking of new peers, a specific time to deploy and start a given service on the allocated resources. Thus the new peers are usable only after a delay.

g) Preparing the Next Round: The steps mentioned above are repeated periodically in a round-based manner with a period of t_R. In each round, the main manager updates the management set as well as the set of resource providers. The set of reserved peers is now up to date and the function *PeerID-list* ← *getReservedPeers(ReservationID)* is served by the main manager which is identifiable by the Reservation ID.

With these steps, the reservations are set up and maintained for the whole reservation time in a totally distributed fashion.

IV. Evaluation

The goal of the evaluation is to measure the quality of functionality, performance and costs of the proposed mechanism for reliable resource reservation and the capacity-based peer search. We discuss the influence of the configuration parameters for both approaches, the probability buffer approach (PBA) and the redundant peer approach (RPA). For the evaluation, first we introduce the relevant metrics and the simulation setup and present then evaluation results, related work and conclusions.

In the following, we present the metrics to validate the function of reliable resource reservation and to measure the performance and costs of the approach: The reservation success ratio, S_{Rsv}, is the ratio of successful reservations over the reservation time. The redundancy success ratio, S_{RsvDR}, is the ratio of services which were successfully provided with a continuous uptime probability higher than the threshold given by the degree of redundancy $RsvDR$ of the reservation. Regarding the costs, we consider the total number of resource providing peers, $Prov._{sum}$, that were needed to accomplish a resource reservation. The average number of peers, $Prov._{avg}$, providing resources during the resource reservation phase sets the total time for resource provisioning of all involved peers in relation to the total reservation time. Further, we consider the traffic overhead for reservation management, measuring the reservation maintenance overhead in terms of traffic. The total number of queries for capacity-based peer search used to complete a reservation indicates the utilization ratio of the assumed capacity-based peer search functionality.

A. Simulation Setup and Workload

We simulated our approach on PeerfactSim.KOM [7], which allows for the simulation of layer-based p2p systems. As a workload model, we use a churn model based on KAD measurements [6] and the underlay model uses measurement data on real-world round-trip times [8]. In Table I, we depict the setup for the evaluation. We simulated a p2p system with 1000, 2000 and 4000 peers in total with KAD churn for 5000 minutes. We varied the number of nodes to observe the impact of the scale of the network on the reservation results. First, all peers join the idealized DHT overlay, CDHT, which offers the KBR-functionality. It provides reliable and consistent ID-based routing in the presence of churn. A fraction of 5% of the peers initiate, after joining the p2p overlay, a reservation request with a reservation time of 3000 minutes. This results in 50/100/200 reservations in the network with 1000/2000/4000 peers. Specifically, we analyzed long-term reservations which had a reservation time significantly longer than the average online time of a peer (266 minutes). We used $t_R = 50m$ and $RsvDR = 0.9$, meaning that $P^3R^3O.KOM$ takes care that the probability for at least one resource providing peer to survive a period of 50 minutes is higher than 90%.

B. Evaluation Results

We show the evolving of the number of peers in the p2p network under churn in the setups with $N = 1000$, $N = 2000$ and $N = 4000$ in the Figures 5(a), 5(b) and 5(c). In these figures, we also depict the number of capacity information at the root of the SkyEye.KOM tree and its support peer (SP) which is available to use for the capacity-based peer search. SkyEye.KOM provides a fresh monitoring view on the capacity of more than 80% of the peers in the p2p network as a basis for the picking of the resource providers in Step h in a fully decentralized manner with no peer being overloaded.

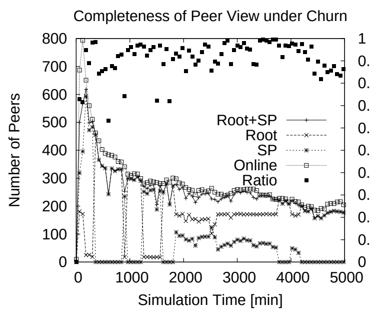
The main metric for the performance of $P^3R^3O.KOM$ is the reservation success ratio, S_{Rsv}. For the cost, it is the number of average reservation providers during the reservation time, $Prov._{avg}$. First, we discuss the performance of the solution and subsequently the overhead of the mechanism. In Fig. 6(a), we depict the performance related metrics of the reservation success. The x-axis lists the various setups for RPA and PBA for a variation of network sizes. Here, the setup number 3, for example, describes the PBA approach with $P_{buff} = 50\%$ and the RPA approach with $N_{buff} = 3$. With increasing setup counter, i.e. with a higher security buffer, the success ratio increases both for the whole resource reservation as well as for maintaining the redundancy level. The RPA approach clearly outperforms the PBA approach in terms of reservation success and reaches with $N_{buff} = 4$ and $N_{buff} = 5$ a reservation success ratio of 100%. We notice that the scale of the network size does not affect the reservation quality, in the case that sufficient suitable providers exist.

In order to maintain the resource reservations, different types of overhead occur, which we depict in Fig. 7. The main metric depicting the costs, $Prov._{avg}$, measures the average number of parallel resource providers for a reservation. As in an optimal case, only one single peer provides the resource for the whole time, we can compare and benchmark the reservation mechanism against this cost metric. The question is, how many resource providers are needed in parallel in order to provide 100% of the resource reservations successfully.

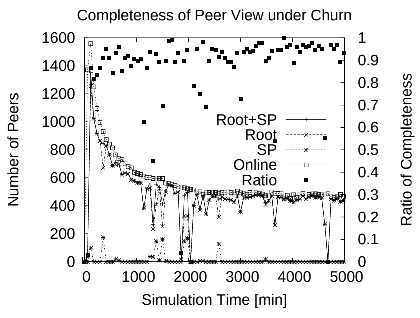
Fig. 7(a) presents the total and average number of resource providers for a reservation. We observe that both metrics increase linearly with setups for RPA, i.e. in specific with N_{buff}. The PBA approach on the other hand allocates much less resource providers for the resource provisioning. Due to this, the performance of the approach is also worse. Regarding the total number of providers, we see that for a resource reservation that last for 3000 minutes approximately 50 to 60 peers are used in total to provide all reservations (e.g. RPA with $N_{buff} = 4$). Although this number seems large, it

TABLE I
Setup for the Simulation of $P^3R^3O.KOM$

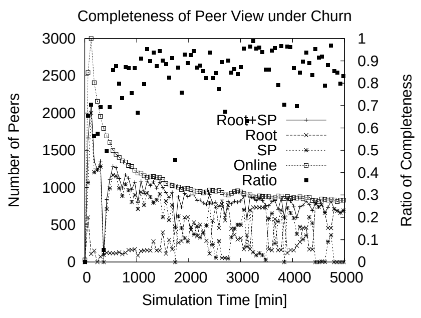
General		**Network and User**	
Simulation time	5000 min	Churn Model	KAD [6]
Number of nodes, N	1k,2k,4k	Avg. t_{on}	266 min
Number of Resv.	5% of N	$RsvTime$	3000 min
SkyEye.KOM		**Resv. Management**	
Attr. Update Freq.	60 s	Management backup	4
Tree degree	4	Backup check freq.	10 min
Resv. Complexity		**Approaches**	
Round time, t_R	50 min	PBA ($\times$ 10%)	1,3,5,7,9
$RsvDR$	0.9	RPA (+ peers)	1,2,3,4,5



(a) Completeness of Peer View, N=1000

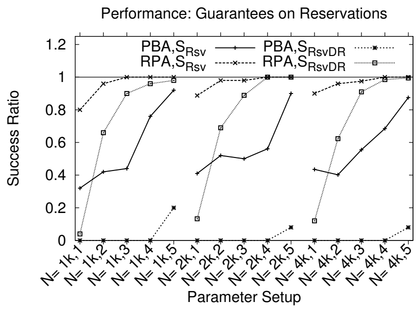


(b) Completeness of Peer View, N=2000

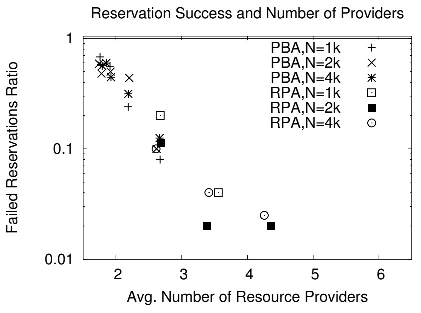


(c) Completeness of Peer View, N=4000

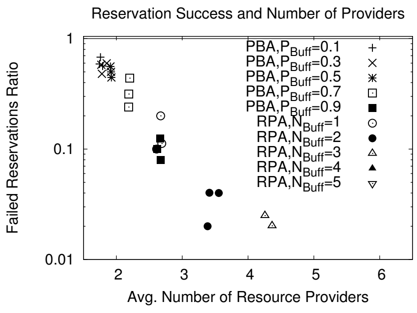
Fig. 5. Peer Count and Completeness of Attribute Information View



(a) Success Ratio of Reservations



(b) Success Ratio, Varying Number of Peers



(c) Success Ratio, Varying Approach

Fig. 6. Performance of Reservation Maintenance

characterizes the main purpose of $P^3R^3O.KOM$, to allocate resources for long term reservations that cannot be provided by single peers due to churn.

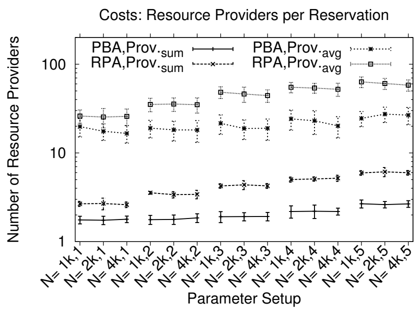
The corresponding traffic for the reservation maintenance is shown in Fig. 7(b). The maximum average traffic overhead of reservation management is reached with RPA and $N_{buff} = 5$, the average reservation management overhead, $Traffic_{avg}$, is in this case 1745 KB over a time period of 3000 minutes, i.e. slightly more than 2 days. With periodic checks every $t_R = 50m$ over 3000 minutes, in total 60 checks have been conducted by the main manager, each with 30kb overhead in average of maintenance messaging. This overhead is very low and underlines the practical usability of the approach.

In Fig. 7(c), we present the capacity-based peer search related queries that are initiated by $P^3R^3O.KOM$ in SkyEye.KOM in order to find suitable peers. Please note that any other mechanism providing capacity-based peer search can be used, the access pattern on that mechanism is the same. The number of queries is directly related to the total number of resource providers per reservation. Periodically in an interval of $t_R = 50m$, the main manager checks whether the amount of resource providers is sufficient or not. In the case that not enough resource providers are there or that they are expected to leave soon, a capacity-based peer search query is initiated for the missing amount of resource providers. The peers in the reply are instantly allocated for resource provisioning.

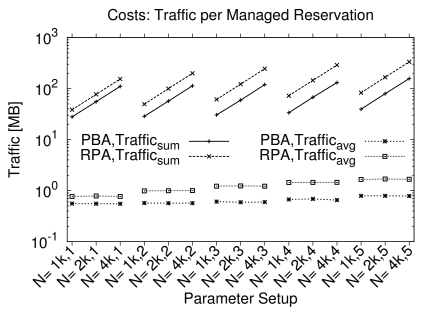
The startup delay for the reservations is in average less then 15 seconds for any setup for $P^3R^3O.KOM$ (no figure). The time is used to contact the main manager, which itself emits a query for suitable peers using the capacity-based peer search functionality of SkyEye.KOM. The query traverses the SkyEye.KOM tree and is replied to the main manager with a set of suitable peers, whose resources are then individually reserved. However, a reservation setup delay of several seconds is to be seen in relation to the reservation time of 3000 minutes.

Having presented the evaluation results regarding the performance and costs of the approaches RPA and PBA of $P^3R^3O.KOM$, next, we investigate in Fig. 6(b) and Fig. 6(c) the direct relation between the number of redundant resource providers and the resulting reservation success probability. The metrics regarding the average and total number of providers and traffic overhead are directly linked to the reservation success ratio. The more redundantly the PBA and RPA approaches pick and provide resource providers, the less probable it becomes that the desired degree of redundancy is missed or the whole reservation fails. The PBA approach allocates in maximum 2.67 resource providers which is not sufficient to fulfill all reservation guarantees. The redundant peer approach, RPA, provides with $N_{buff} = 3$ in the network with 1000 peers a 100% reservation success ratio, but fails in the network with 2000 and 4000 peers in two cases. With $N_{buff} = 4$ and $N_{buff} = 5$, 100% of the reservations are successfully fulfilled. For that, in average 5.10 and 6.02 peers respectively are invoked simultaneously in average for resource provisioning.

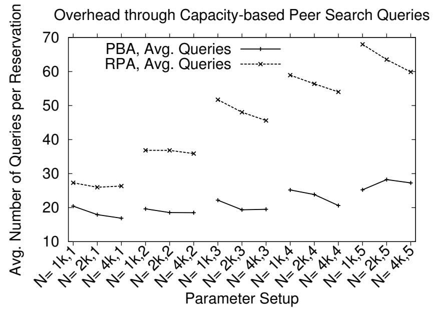
To conclude the evaluation on the reservation management solution $P^3R^3O.KOM$, we present a series of reservation requests taken from the simulations with 4000 peers in which 200 reservations were initiated. In Fig. 8, we depict these



(a) Providers per Service



(b) Traffic Overhead



(c) Query Overhead per Service

Fig. 7. Costs related to the Resource Reservations

series using the PBA approach with $P_{buff} \in \{30\%, 90\%\}$ and the RPA approach with $N_{buff} \in \{2, 5\}$. The rectangles in the figures depict individual peers providing resources for the reservation. It is obvious that the average lifetime of a peer is much shorter than the reservation time. Without $P^3R^3O.KOM$, the desired resources could not be used over the long reservation time in the presence of churn. Fig. 8(a) shows that for PBA with $P_{buff} = 30\%$, the aimed success probability of $RsvDR + P_{buff} \cdot (1 - RsvDR) = 0.9 + 0.03 = 0.93$ is not enough to maintain the success probability higher than $RsvDR$ and to maintain the reservation. In the time from minute 1321 to 1324 no providers exist and the reservation breaks. Although the main manager picks then new providers, the interruption of the reservation is unacceptable. Fig. 8(c) shows a service example using PBA with $P_{buff} = 90\%$. The resources are continuously provided and the reservation is successful. However, the aimed degree of redundancy is missed three times, at minutes 3425, 3491 and 4278. In Fig. 8(b), we depict an example using RPA with $N_{buff} = 2$. The behavior is similar to the PBA approach with $P_{buff} = 90\%$, however the result is better due to the increased average number of providers. The RPA approach adds 2 additional peers for providing the resources and in order to strengthen the reliability, thus at least 3 providers in parallel are aimed at. In Fig. 8(d), we show an example with RPA and $N_{buff} = 5$. In this case, the approach aims at 6 peers in parallel leading to a success ratio for the reservations of 100%.

We evaluated the costs and performance of $P^3R^3O.KOM$ and with a parameter study the effect of the maintenance effort, $Prov._{avg}$, on the reservation success ratio S_{Rsv}. We showed that a redundancy level of 5-6 is needed in a p2p network with KAD churn to provide a 100% reservation success ratio.

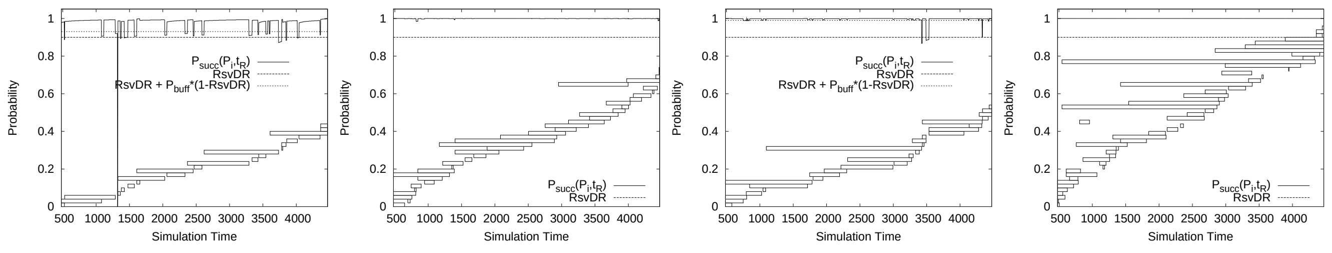
V. Related Work

Regarding p2p-based cloud or GRID platforms, several approaches have been published. In [9], the authors portray an approach to deploy services on peers, aiming at low delays to customers. The approach stated in [10] delegates and uses services on peers in a p2p network. Both approaches put the reservation initiator in place to supervise the execution. We aim at long-term resource reservation which allow the initiator to leave the system in the meanwhile, while the main manager of the reservation as well as its backup peers maintain the reservation. In [11] and [12] a p2p-based approach for service discovery in GRIDs is given. The integration of self-organizing elements in GRIDs is presented in [13]. The authors of [14] hypothesize a p2p-based job scheduler for GRIDs, allowing peers in the GRID to derive their own schedules. The authors of these papers focus on interconnecting and improving existing GRIDs, while we fully utilize the resources of the user devices. The authors of [15] design a new overlay for managing resources, which ignores the many very good p2p overlays existing in the community. We proposed a dedicated mechanism that relies on existing, well evaluated structured p2p overlays. With them, $P^3R^3O.KOM$ provides 100% successful resource reservations. In [16], a portal layer is introduced on top of the p2p overlay and resources are communicated with roaming agents. This information is hardly to be complete and does not aim at providing reservations on these resources. In [17], a p2p GRID for distributed resource management in unstructured p2p systems is discussed. The authors emphasize approaches for service registration and discovery, but do not address the issue of churn. The authors of [18] focus in their paper on the retrievability of resource information in p2p networks, they do not focus on their long-term reservation.

Considering related work in the field of reliable resource reservation, we state the claim that only few work has been conducted by the community. This is to our believe based on the fact, that for reliable resource reservation a mechanism is needed to find suitable peers and this field is sparsely investigated as well. P2P systems mainly focused on data-centric application scenarios or on short term resource consumption at the edge. With the upcome of service oriented architectures and clouds, however, the topic of a reliable p2p service infrastructure is emerging as well. We address this field with our approach for reliable resource reservation in unreliable p2p systems.

VI. Conclusion

In this paper, we discussed the issue of unreliable resource reservations in p2p systems. Reliable (reserved) resource provision allows for the hosting of services and the provision of a distributed service oriented platform. However, peers in p2p systems may leave the network anytime and thus interrupt any hosting task using the reserved resources. We



(a) PBA, $P_{buff} = 30\%$ (b) RPA, $N_{buff} = 2$ (c) PBA, $P_{buff} = 90\%$ (d) RPA, $N_{buff} = 5$

Fig. 8. Example Reservations managed with PBA and RPA in $P^3R^3O.KOM$

aimed with our solution, $P^3R^3O.KOM$, especially at the provision of long-term resource reservations, which outlast the average lifetime of a peer by far. For that, we proposed a twofold mechanism, termed $P^3R^3O.KOM$, with a distributed reservation management set and a set of resource providing peers. The management set acts as contact group for reservation initiations and access to the list of resource providing peers. This set of resource providers is controlled by the resource manager and supplemented with a sufficient number of additional peers in the case, that the peers in the set are not expected to stay online until the next control. To find suitable peers, we used our monitoring approach SkyEye.KOM [5] to gather peer specific information and to provide the functionality of capacity-based peer search, which is challenging in large-scale p2p networks.

We proposed two approaches to identify the required number of redundant peers for resource provisioning, the probability buffer approach, and the redundant peer approach. Evaluation shows, that the proposed solution in combination with the redundant peer approach fulfills the resource reservations in 100% of the cases with $N_{buff} = 4$ and $N_{buff} = 5$ with a service cost of 5.10 and 6.02 times more resource invested than requested. We evaluated the investment costs with a redundancy of 1-5 peers and found that 100% successful reservations require this amount of redundancy in the presence of churn behavior that was measured in real networks like KAD. The traffic overhead for maintaining the reservations by the resource managers is maximum 1745 Kb in 2 days, for which the reservation was lasting. Thus, the maintenance costs are considered being very low while the resource reservation is provided with 100% success ratio. In addition, the cost for the resource management and provision is independent of the network size, it is only influenced by the reservation complexity and reservation time.

This solution allows for reliable resource reservation in unreliable p2p networks with continuously joining and failing peers. Through the creation of a reliable infrastructure, service may be deployed in a distributed fashion, setting a mark towards a p2p-based cloud or service oriented architecture.

In the future, we plan to extend $P^3R^3O.KOM$ with a sensor considering the access patterns for reservations. Thus, we enable the solution to handle flash crowd access patterns by adding more redundancy to the resource providers in the case of an increased access frequency for a given reservation. Further, we plan to add a service deployment and relocation function to $P^3R^3O.KOM$ in order to create a p2p-based service platform. We believe, that the proposed solution helps in enabling a p2p-based cloud infrastructure providing users a platform to share and use their resources (for free) and to better utilize the available capacities at the edge of the Internet in a reliable manner.

References

[1] K. Graffi *et al.*, "From Cells to Organisms: Long-Term Guarantees on Service Provisioning in Peer-to-Peer Networks," in *Proc. of ACM SIGAPP NOTERE '08*, 2008.

[2] K. Graffi *et al.*, "Overlay Bandwidth Management: Scheduling and Active Queue Management of Overlay Flows," in *Proc. of IEEE LCN '07*, 2007.

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

[4] K. Graffi *et al.*, "SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems," in *Proc. of IEEE ICPADS'08*, 2008.

[5] K. Graffi *et al.*, "Monitoring and Management of Structured Peer-to-Peer Systems," in *Proc. of IEEE P2P '09*, 2009.

[6] M. Steiner, T. En-Najjary, and E. W. Biersack, "Analyzing Peer Behavior in KAD," Institut Eurecom, Tech. Rep. EURECOM+2358, 2007.

[7] A. Kovacevic *et al.*, "Benchmarking Platform for Peer-to-Peer Systems," *it - Information Technology*, vol. 49, no. 5, 2007.

[8] T. S. E. Ng and H. Zhang, "Global Network Positioning: A New Approach to Network Distance Prediction," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 1, 2002.

[9] P. Bocciarelli, A. D'Ambrogio, and M. Angelaccio, "QShare: QoS-Enabled Description and Discovery of Services in SOA-Based P2P Applications," in *Proc. of IEEE WETICE*, 2007.

[10] M. Amoretti *et al.*, "SP2A: Enabling Service-Oriented Grids using a Peer-to-Peer Approach," in *Proc. of IEEE WETICE*, 2005.

[11] B. Labno, M. Bubak, and B. Balis, "A P2P Approach to Resource Discovery in On-Line Monitoring of Grid Workflows," in *Proc. of Euro-Par*, 2008.

[12] M. Gharzouli and M. Boufaïda, "A Generic P2P Collaborative Strategy for Discovering and Composing Semantic Web Services," in *Proc. of ICIW*, 2009.

[13] F. Schintke, T. Schütt, and A. Reinefeld, "A Framework for Self-Optimizing Grids Using P2P Components," in *Proc. of IEEE DEXA Workshops*, 2003.

[14] J. Cao *et al.*, "A Peer-to-Peer Approach to Task Scheduling in Computation Grid," in *Springer GCC (1)*, 2003.

[15] D. Li *et al.*, "IPBGA: A hybrid P2P based Grid Architecture by using Information Pool Protocol," *Springer Journ. of Supercomp.*, vol. 49, no. 2, 2009.

[16] W. Dan and Z. Rongjuan, "A Layered Resource Management Model in P2P System," in *Proc. of IEEE PDCAT '05*, 2005.

[17] P. Uppuluri *et al.*, "P2P Grid: Service Oriented Framework for Distributed Resource Management," in *Proc. of IEEE SCC '05*, 2005.

[18] A. Rajput and S. Rotenstreich, "Making a Case for Resource Management in a P2P Environment," in *Proc. of CSREA IKE '04*, 2004.