HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

Winter semester 2014

Jun.-Prof. Dr.-Ing. Kalman Graffi

Heinrich Heine University Düsseldorf

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

Controlling the Quality of Peer-to-Peer Systems
– Control Loops in P2P Systems

▶ Management of Peer-to-Peer Systems

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Management

- Administration of distributed systems
- Here: we focus on controlling the quality of service

Scenario:

- Large scale p2p application (VoIP, social network, game)
- Monitoring tells:
  - Response time is too slow
- Question: how to influence the response time?
- Answer: adaptation of peer configuration helps

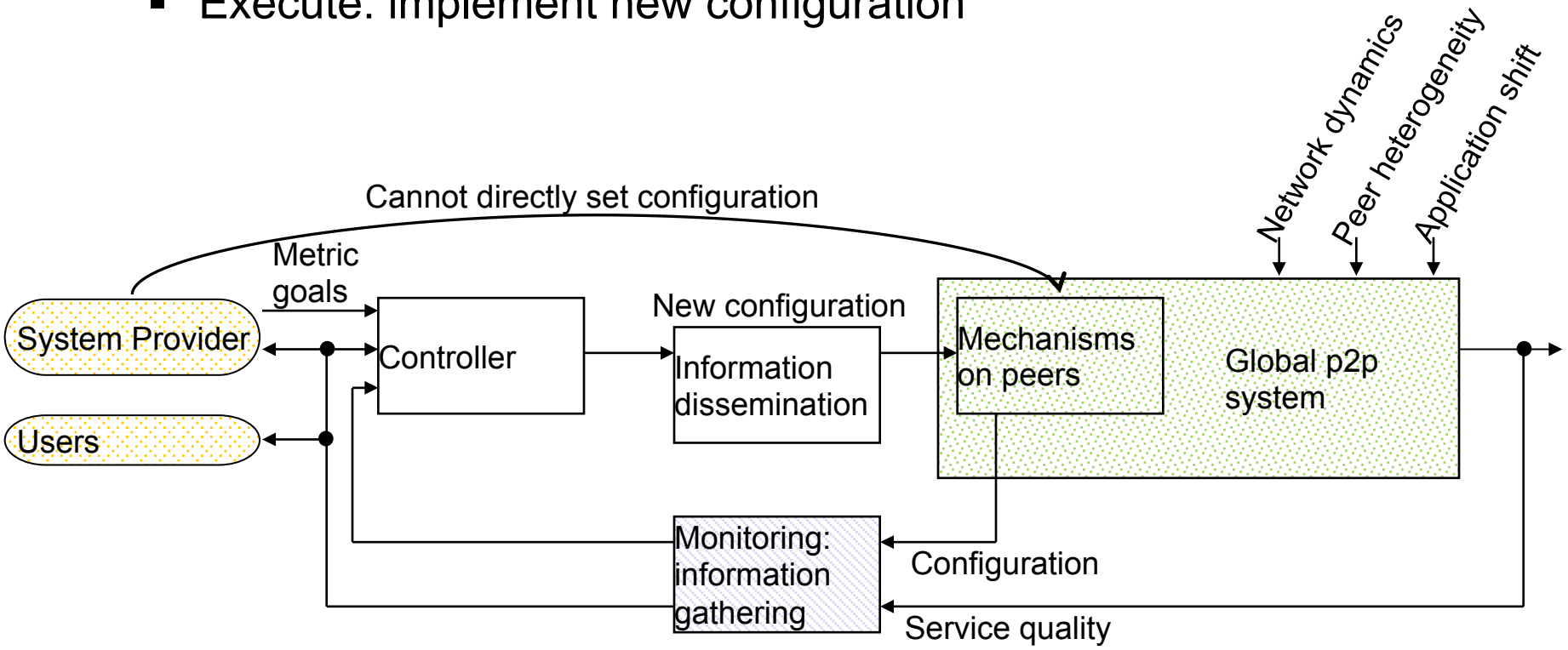
Management enables

- Access to peer configuration
- Automated re-configuration of all peers
  - Coordination and initiation of actions
  - Analysis and planning

Excursion: Control Theory

Control loop:

- Monitor: obtain information on status
- Analyze: decide on quality of status
- Plan: decide on new configuration
- Execute: implement new configuration



Elements for Controlling P2P Systems

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

System operator

- Gives input: defines quality goals
- Could be also the system itself

P2P system

- Needs to be parameterizable
  - All configurations open to change
  - Examples
    - Stabilization intervals, neighbor selection strategy, routing table size
    - Replication rate, tree degree, monitoring update interval
    - ...
- Needs to be monitorable
  - All relevant quality measure must be obtained
  - Examples
    - Up/down bandwidth consumption, response time, hop count
    - Lookup success probability, node load, peer count, peer uptime
    - ...

Towards Self-Configuring P2P Systems

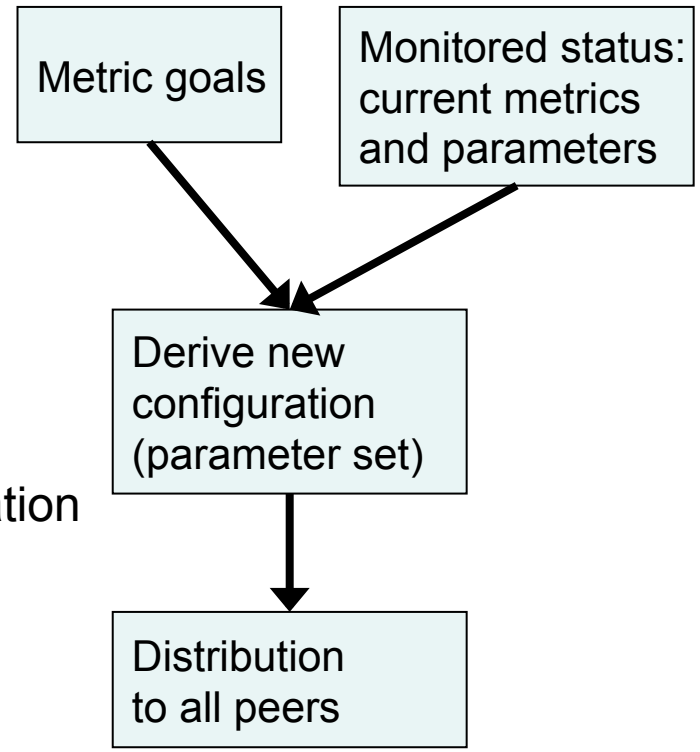
HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Steps prepared

- Predefined quality goals given
- P2P overlay parameterizable
- Monitoring reveals current system state

Steps needed

- Derive new parameter configuration
- Based on
  - Predefined metric intervals
  - Current metrics and parameter configuration
- Distribute new configuration timely to all peers

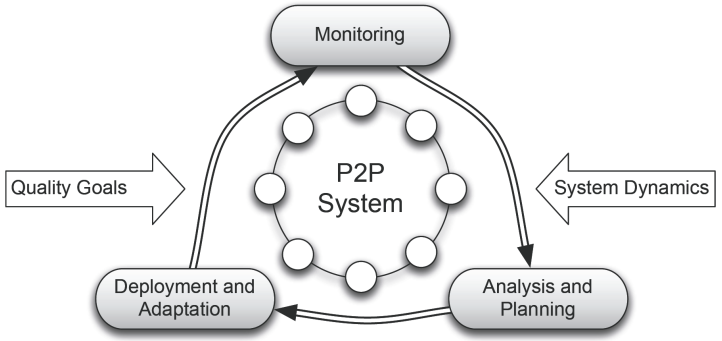


Goal: System reaches and holds predefined metric intervals

Simple Planning

Analysis and planning step

- Detects metric violation
- Step-wise adaptation of configuration
- Approaching target metric interval

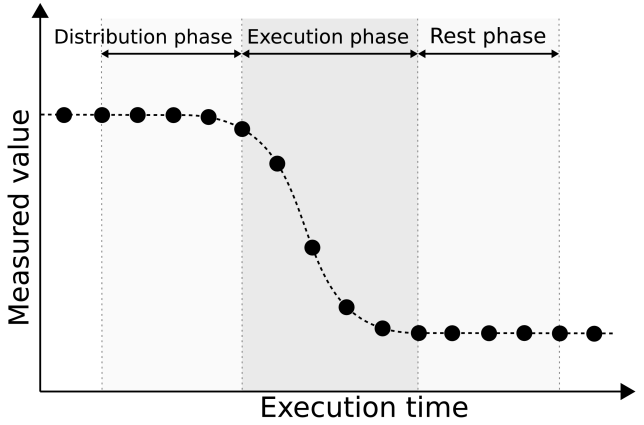


Changes need time

- Analyze slope of metric history
- Act only if changes settled
- Prevent configuration oscillation

Planning:

- Who initiates changes?
- Who decides on their execution?



Option 1: Decentralized Planning

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Scenario:

- Unstructured p2p overlay with gossip-based monitoring

Idea:

- Every node monitors and plans locally
  - Local monitoring
  - Globally accepted planning rules
  - Every node re-configures himself

Pro:

- No coordination effort needed
- Quick adaptation

Cons:

- How to adapt globally accepted planning rules
- Behavior might diverge (divergent monitoring history)

Option 2: Central Planning

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Scenario:

- Structured p2p overlay with tree-based monitoring

Idea:

- Root of monitoring tree manages
  - Monitoring tree gathers also configuration information
  - In root available: system status and configuration
  - Root analyzes and plans re-configuration
  - Pushes re-configuration command with global status to all peers

Pro:

- Unified decision
  - Effort of considering long history trails justified
- Quality goals centrally tunable

Cons:

- Trustworthiness of root?
- Induces more overhead

Re-Configuration of the System

Management protocol

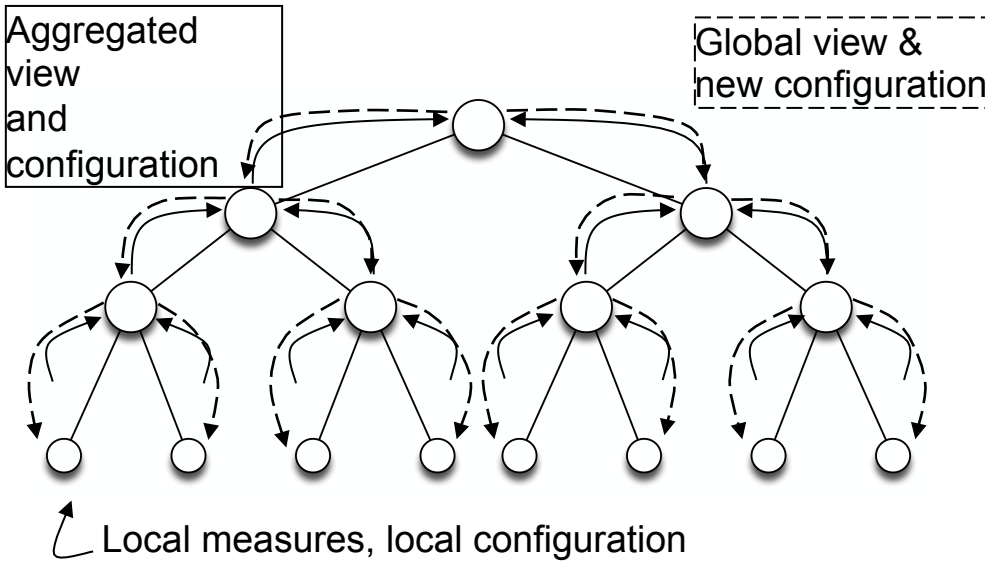
- Gather current system parameters
- Distribute new parameter configuration

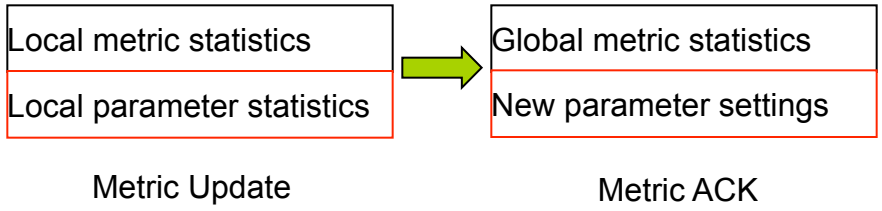
Approach

- Extension of SkyEye.KOM
  - Gather statistics on parameter settings
- Root:
  detects violation, plans new configuration
- New configuration:
  - Spread with SkyEye.KOM in ACKs
  - Applied by peers locally

Planning step

- Various approaches possible
- Closing the control loop

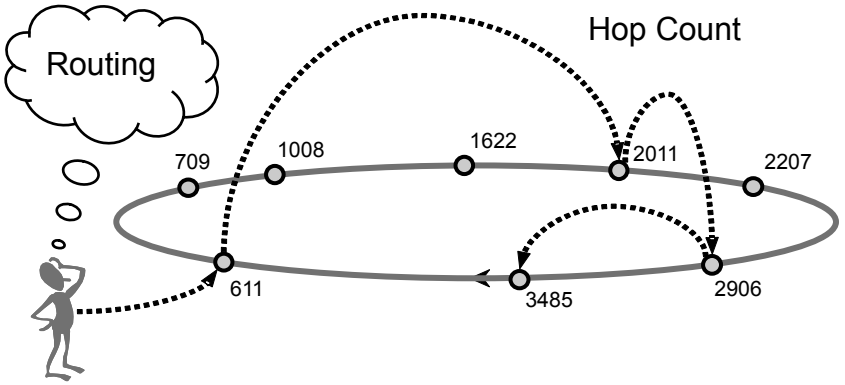




Management of the Hop Count in Chord

Adapted p2p overlay “Chord”

- Classic DHT
- Adapted:
  - Parametrizable finger table size
  - To consider new configuration

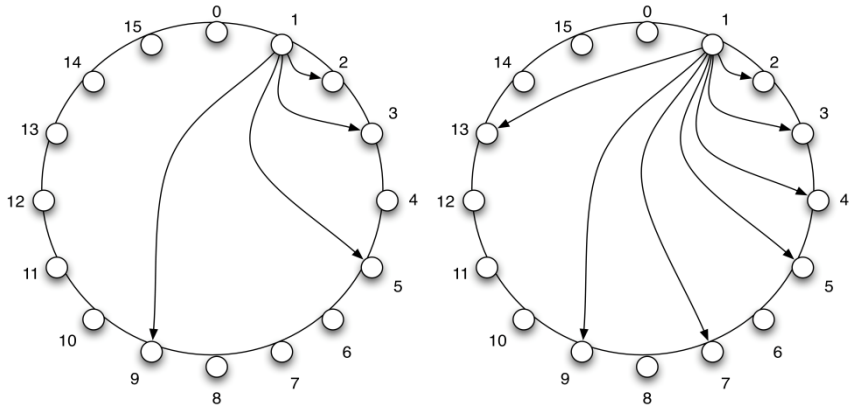


Managing the behavior of Chord

- Main metric: hop count
- → Target metric: [7, 10]
- Parameter: finger table size

Evaluation goal

- Target metrics are reached and held
- Feasibility of control loop



Simulation Setup

- N=10000, UI=30s, β=4

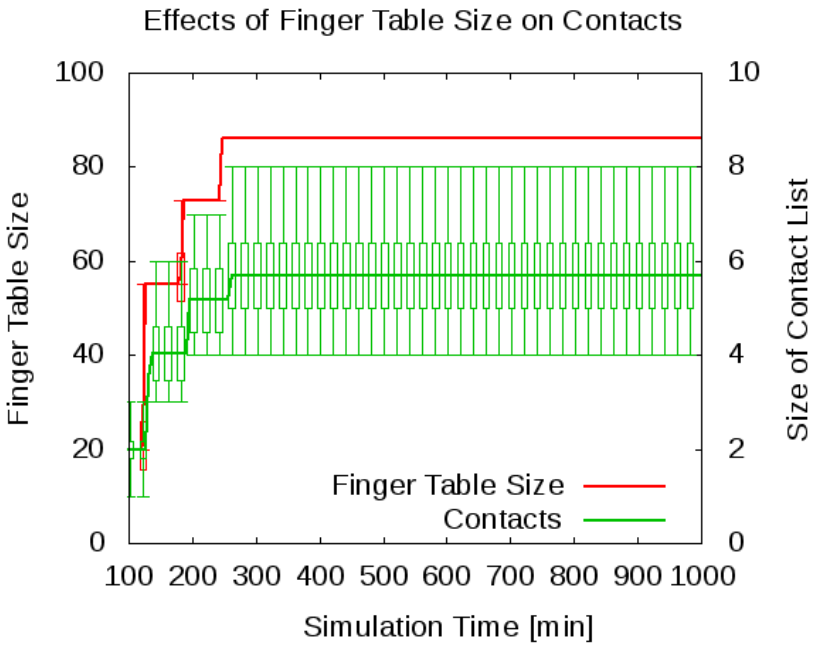
Interdependencies in Chord

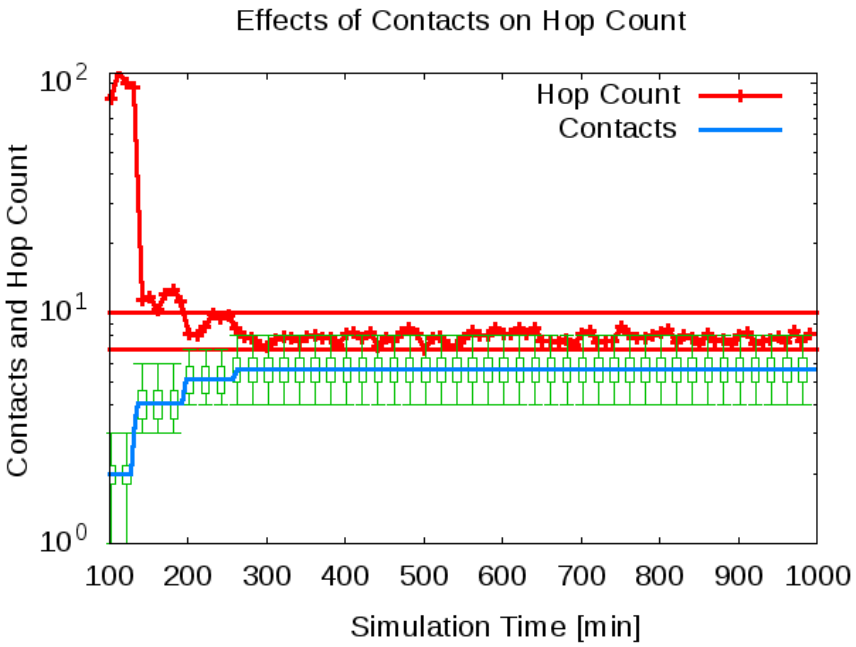
Contacts ~ log(FingerTableSize)
Hop Count ~ 1 / logContacts(N)
Approach:

- Modify FingerTableSize → effect Contacts
- Modify Contacts → effect on Hop Count

Rules for violation of the goal metric

Static Rule Approach	Finger table size
Hop count too small	- 10 %
Hop count too large	+ 100 %



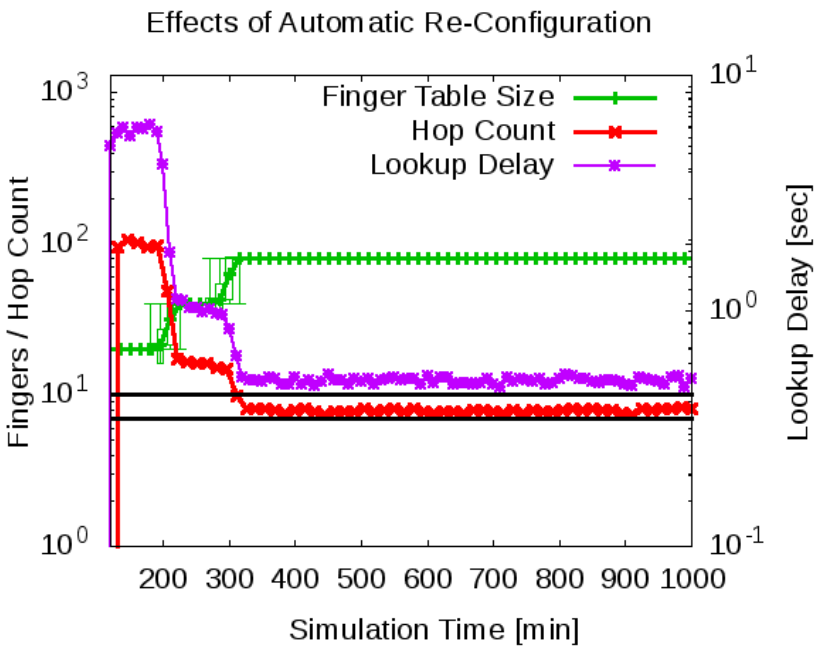


Evaluation Results

Results

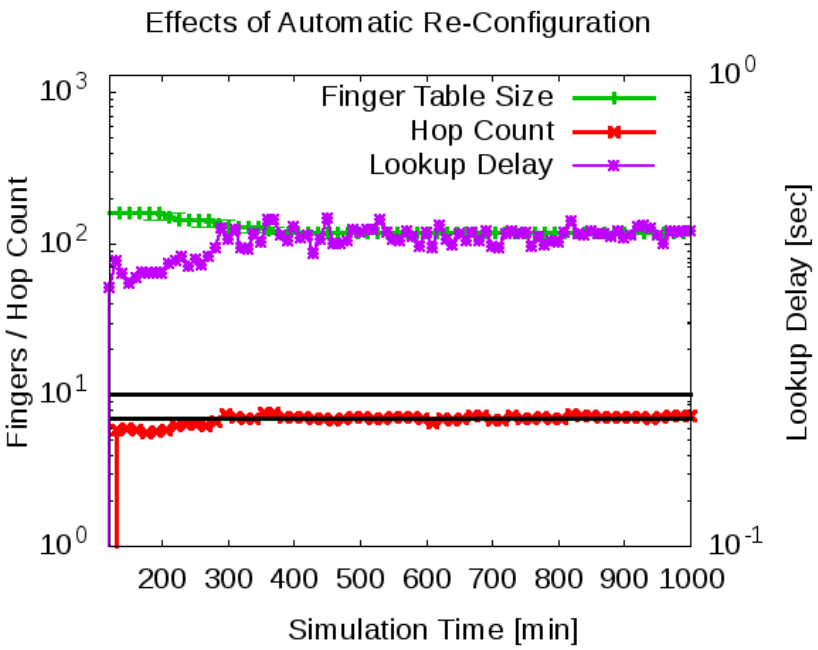
- Metric interval [7,10] is reached and held
- Management cycle is closed

Start: too large	Initial	2 Steps
Finger Table Size	20	80
Hop Count	100	8



- Target metric approximation from 2 sides
- Corresponding effects on lookup delay

Start: too small	**Initial**	**3 Steps**
Finger Table Size	**160**	**117**
Hop Count	**5.7**	**7.1**



Evaluation of Control Loops in P2P Systems

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Motivation

- Monitoring and control of quality of service
- Necessary for running a commercial p2p system

Further research questions

- Long-term storage of monitored status
- Automated distributed learning
  - Interdependencies of configuration and quality
  - Relates quality demands to configuration demands
- Security issues

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

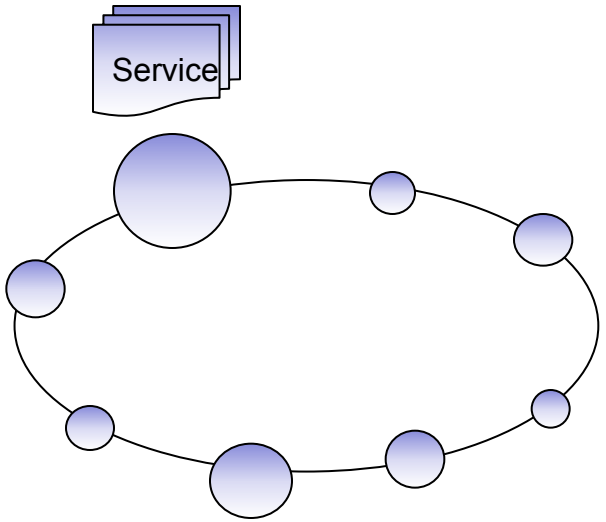
Peer-to-Peer Cloud Systems

– Monitoring Peer-specific Information

OTF Scenario / Issus of P2P

On the Fly Computing

- Service market enables flexible service composition
- Services offered in p2p network
- Services run on commercial OTF compute centers
- Variation: Services run on suitable peers



Challenge 1:

- Peer capacities are heterogeneous
- How to find suitable peers?

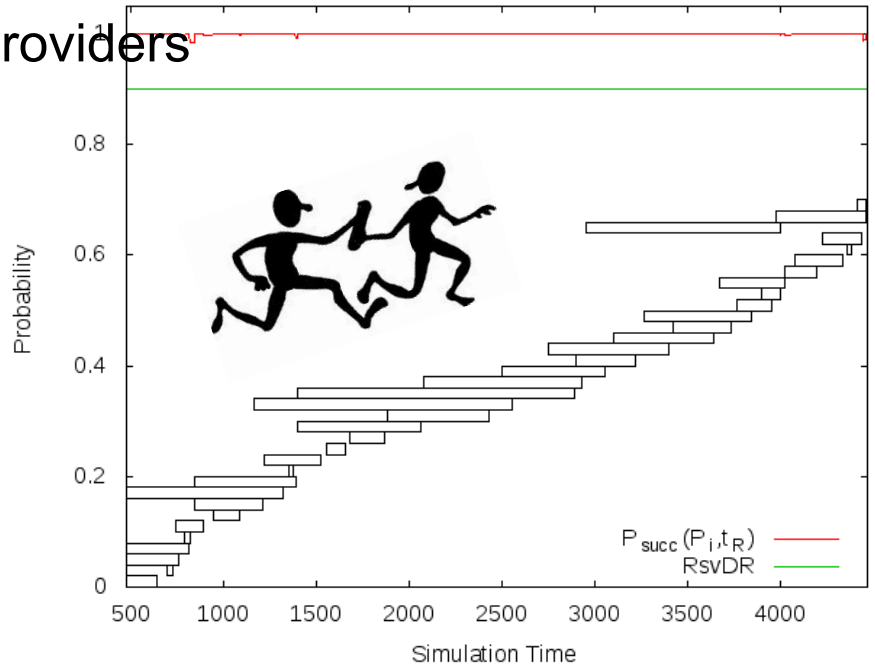
Challenge 2:

- Lifetime of peers is limited
- How to support long-term “reservations”?

Towards Reliable Resource Usage

Reliable p2p services

- Require replication of resource providers
  - FIND suitable peers
    - that provide desired resources
  - Allocate them in a resource pool
    - for the specific service

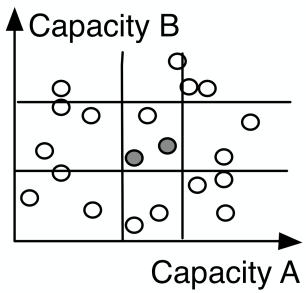


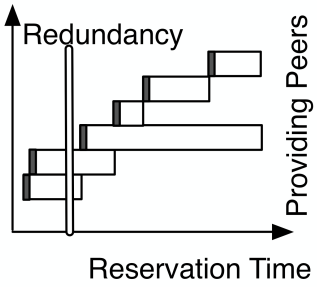
Reliable

- Resource identification
- Resource allocation
- Service hosting

Implications

- Allows for p2p-based cloud services
- With service-level agreements





Management of Reliable Resource Reservations

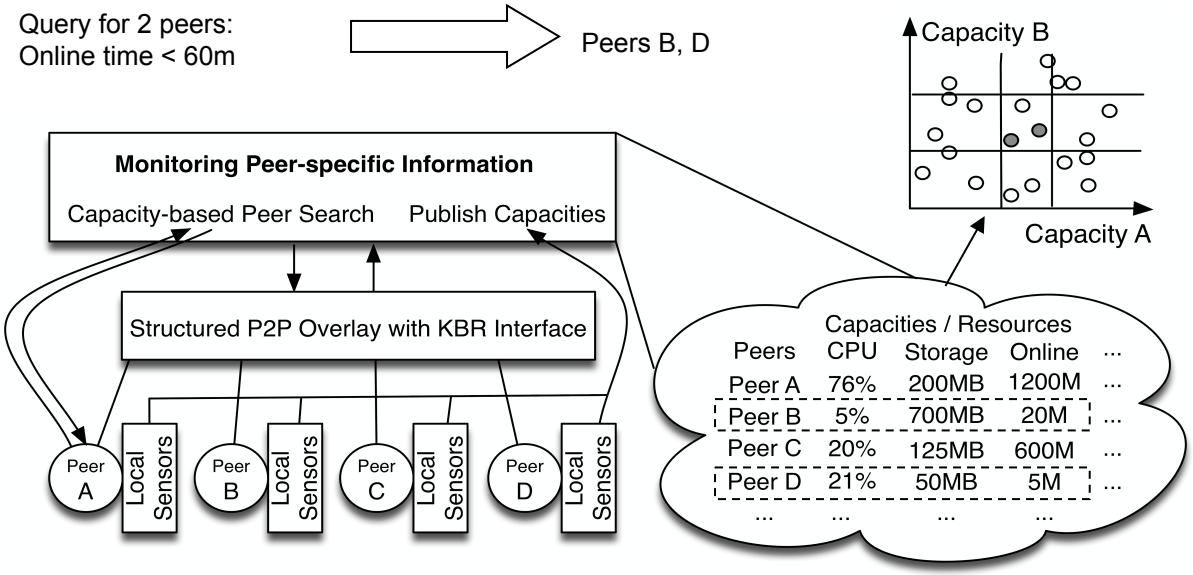
Monitoring Peer-specific Information

Peer-specific information

- Capacities:
  - Max / current bandwidth
  - Operating System, Java version
  - CPU power
  - Free disk space
  - Responsibility range
  - Parent coordinator
  - …

Information is NOT aggregatable:

- Size of information grows with number of peers
- Leads to overhead issues
- List-based concatenation
  E.g. peer 101, up bandwidth 27kb/s,
  …



Typical List of Attributes

X	X	Parent PeerID - ID of the Parent-Coordinator	ID
X	X	Current tree level of the node	#
X	X	Lifetime of the peer	#
Offered Capacities			
X	X	Offered download capacity	KB/s
X	X	Offered upload capacity	KB/s
X	X	Offered CPU cycles	#
X	X	Offered main memory	MB
X	X	Offered storage space	MB
Evaluation-specific Attributes			
X		GNP-Location Coordinates	5 Doubles
	X	The operating system of the peer	String
	X	The system architecture of the node	String
	X	The current version of the node's Java VM	String
	X	The lower bound of the node's responsibility range	ID
	X	The upper bound of the node's responsibility range	ID
Attributes relevant for the Tree Extension with Support Peers			
X	X	Maximum capacity for receiving attribute entries	#
X	X	Maximum capacity for sending attribute entries	#

Gathering Peer-specific Information

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Type of information
- Individual Peer ID and peer specific information:
  - Free storage space, CPU power, bandwidth capabilities, online time, …
  - Responsibility range, node degree, Coordinator ID, …

Desired query
- Capacity-based peer search:
- Find N peers with e.g. node degree > 20, free storage space > 10MB, online time > 10h

Design decision: proactive
- Constantly gathering peer information in the tree
- Query directly accesses prepared data
- Better for scenarios with frequent queries

Design decision: incomplete information
- But 100% of queries should be solved

Challenge:
- Information cannot be aggregated → grows in size
- Costs may overload the Coordinators

Solution idea:
- Basic tree structure as monitoring topology
- Replace weak peers in tree with strong Support Peers

SkyEye.KOM: Tree Topology

Concept of new layer

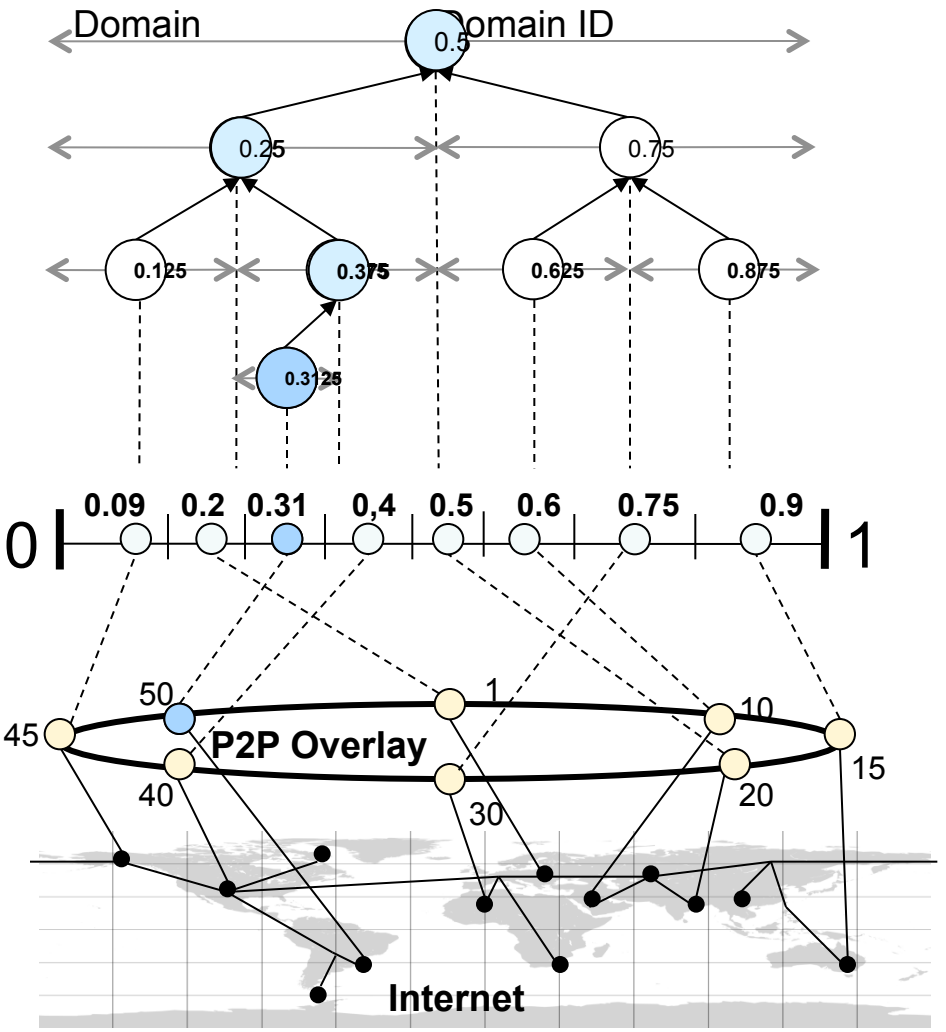
- Decouples from specific p2p overlay
- Unified ID space [0,1]

Tree of information domains

- Domain: ID range for monitoring
- Domain size split in β parts per level
- Domain IDs build tree topology

Peers to Domain ID assignment

- Peer ID p, level l → Domain IDs
- If peer is responsible: position defined



Gathering Peer-specific Information

Supporting Peers for Load Balancing

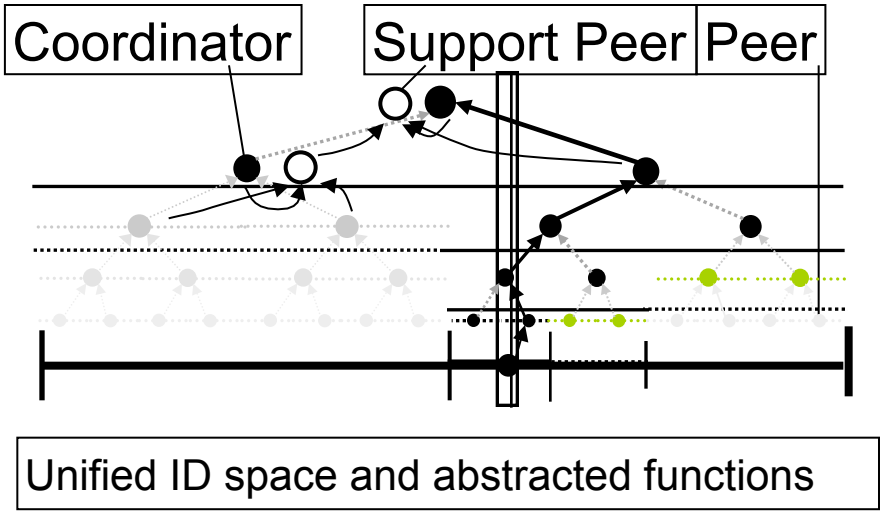
- Each peer defines max. load
- Coordinator may choose strong Supporting Peers
- Workload delegated to supporting peer

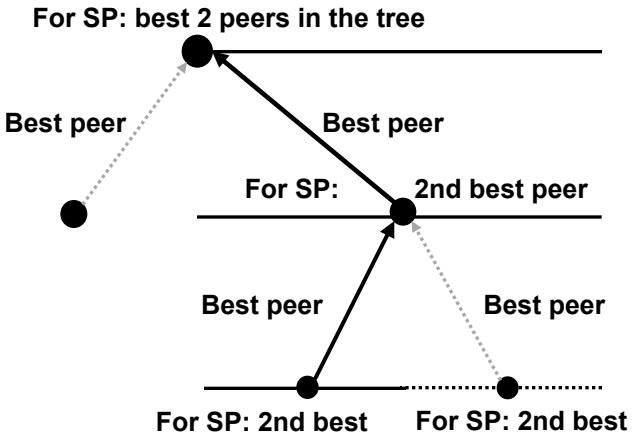
Good peers chosen

- Pick 2 best peers in the domain
- Best peer advertised one level up
- Second best peer used

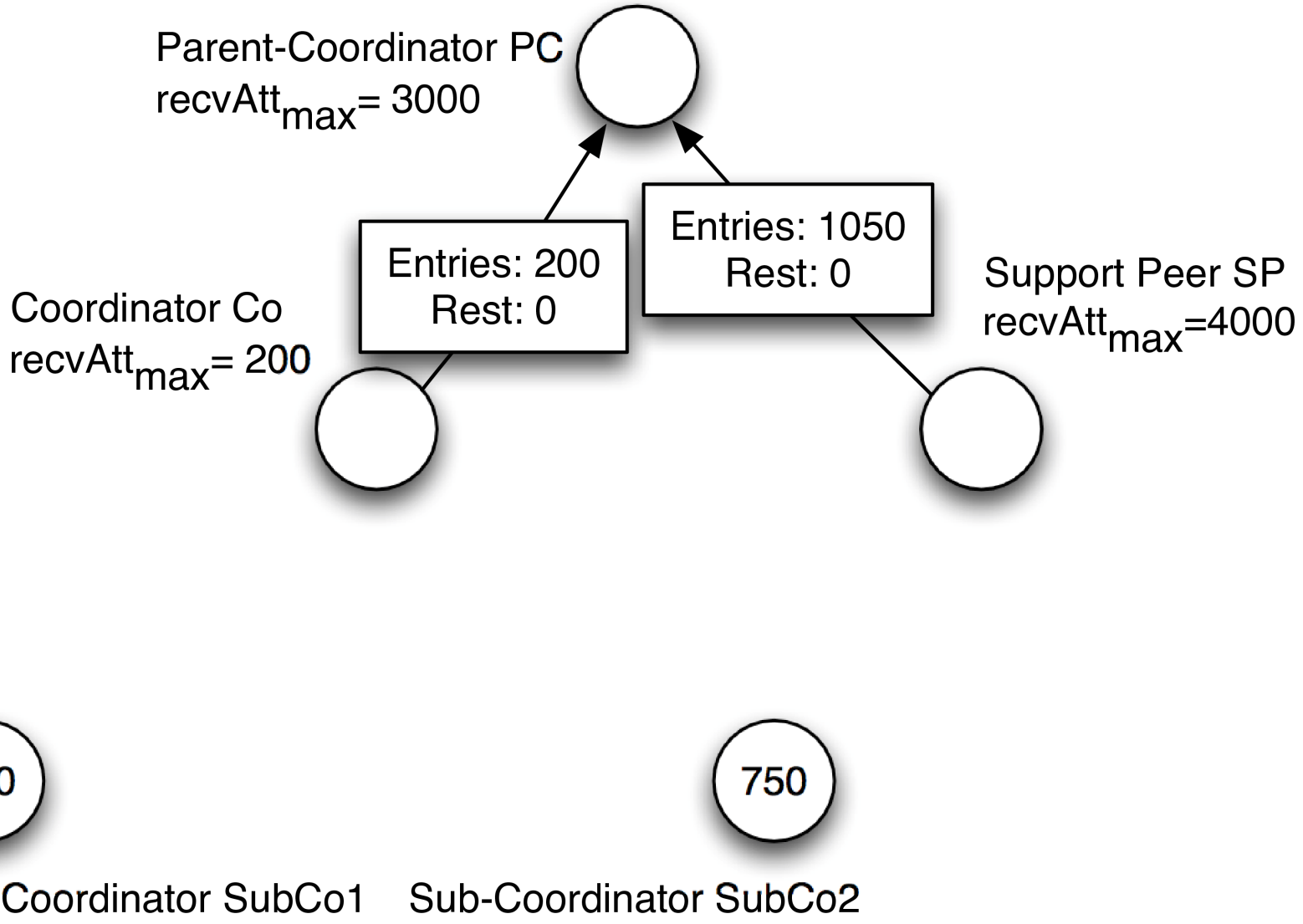
Results

- In a tree with strong peers
- Best peers at the top, carrying most of the load
- No peer is overloaded

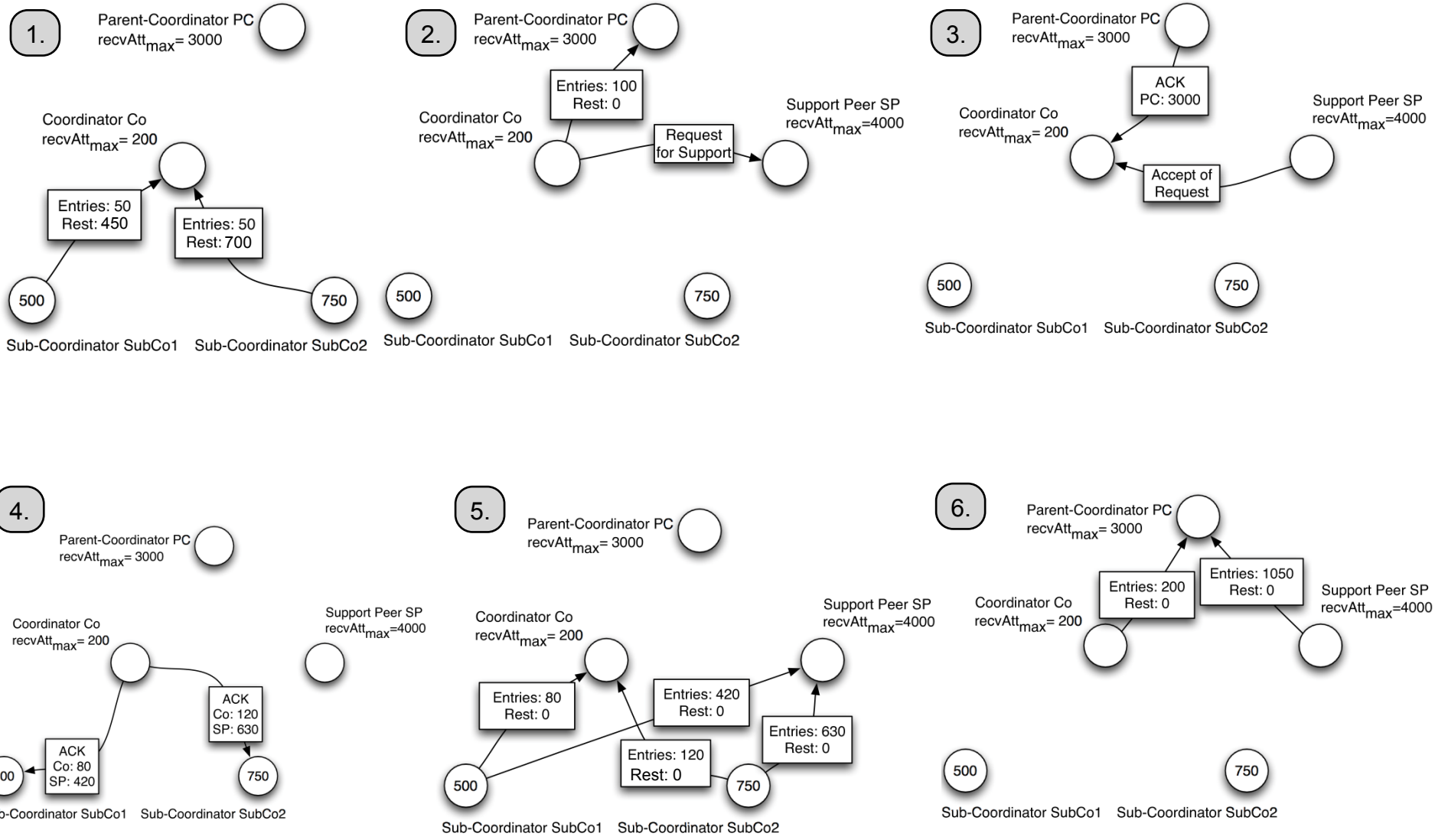




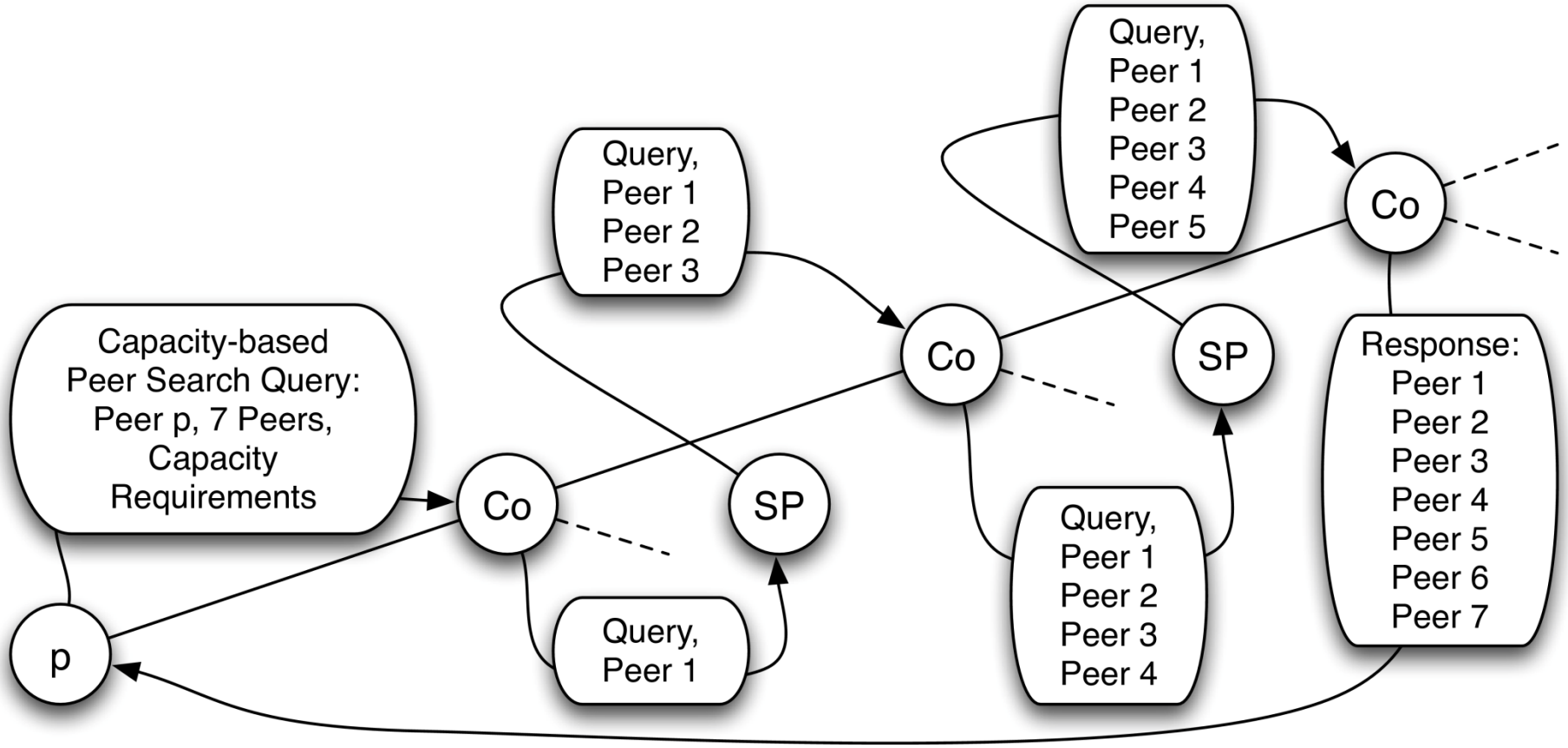
Protocol for Peer Capacity Indexing



Protocol for Peer Capacity Indexing – for Printing



Query Protocol



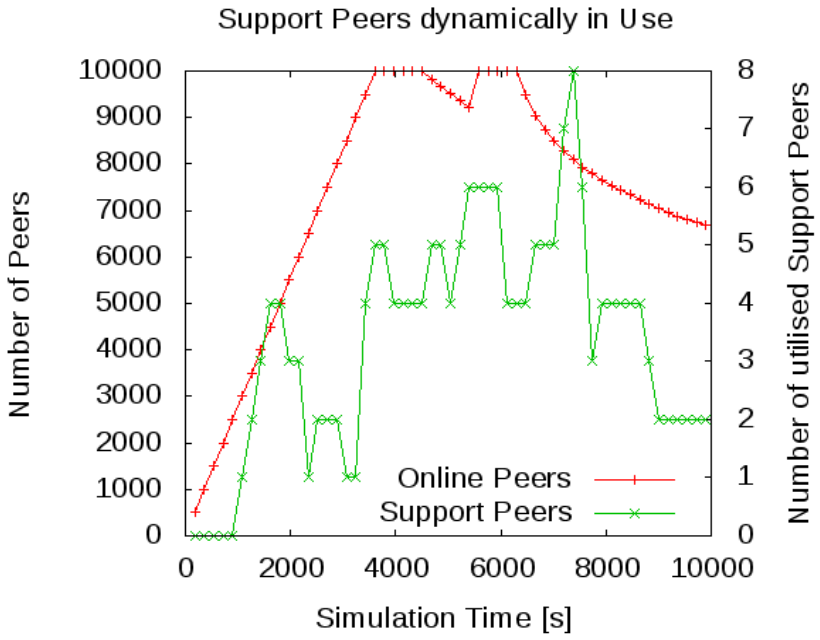
Evaluation

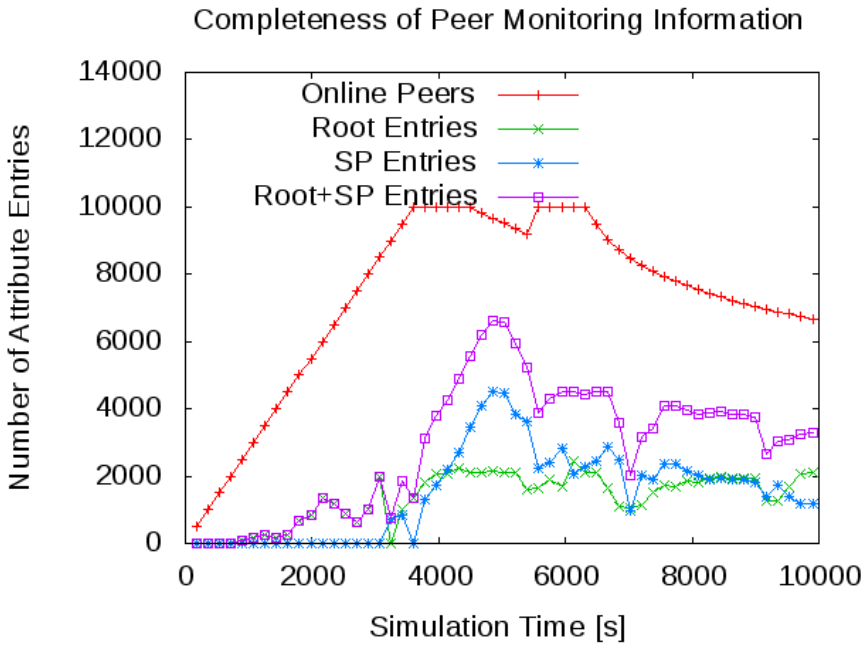
Setup:

- Nodes: 10,000
- Protocols: Chord, SkyEye.KOM
- Heterogeneous peer capacities
- Churn: join, exponential churn, KAD churn

Coverage

- Scope limited by the load limits of the individual peers
- Higher tree levels supported with strong Support Peers





SkyEye.KOM: Peer-specific Information

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Query – originators and solvers

- Most peers around level 10
- Most queries solved between root and peers at level 5

Effect of query complexity

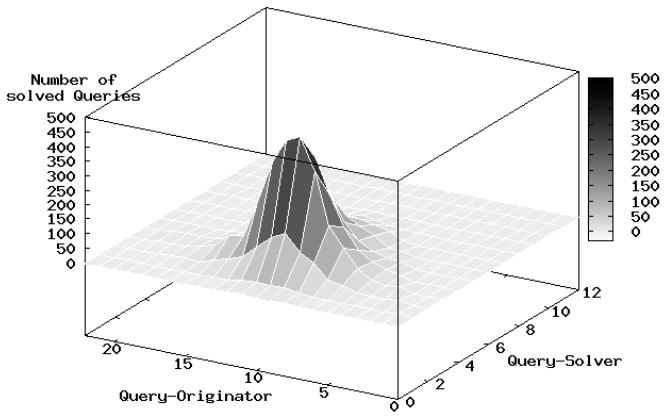
- Queries demanding better resources
  are solved higher in the tree
- "Good" peers bubble up in the tree
- Hard queries solved near root

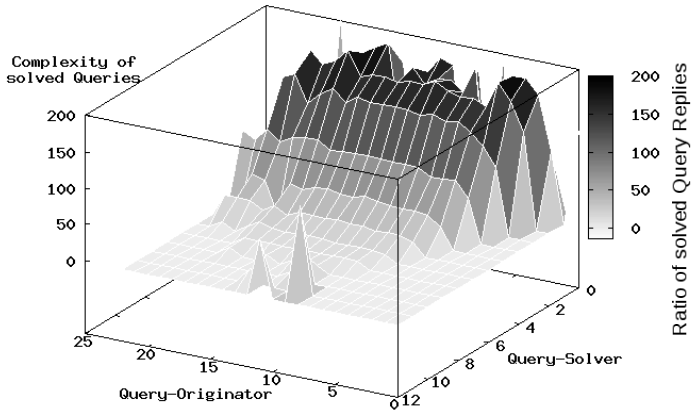
Quality of query results

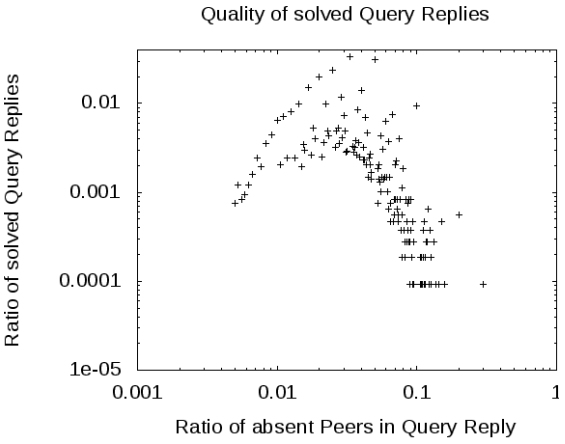
- Reply contains correct results

In only very few cases:

- Peers in reply already left







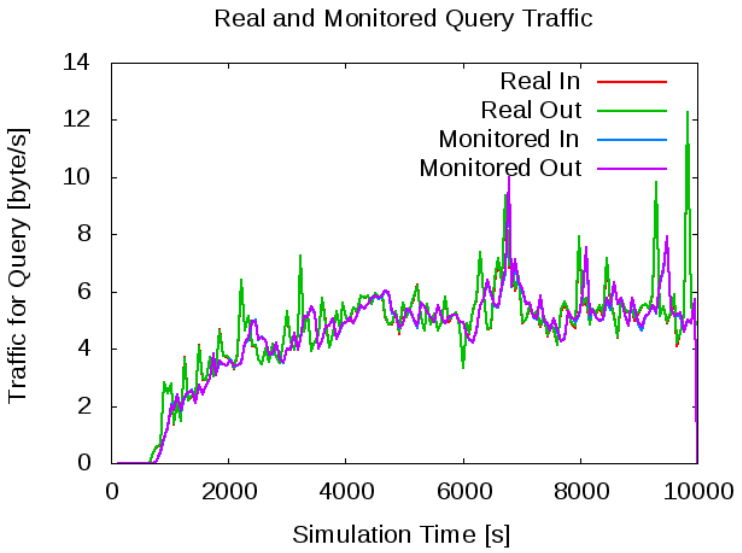
Traffic Overhead

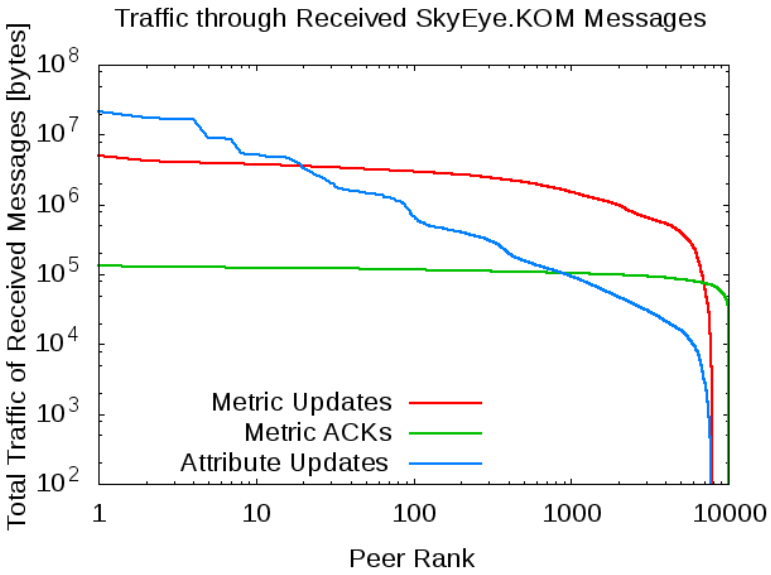
In average:

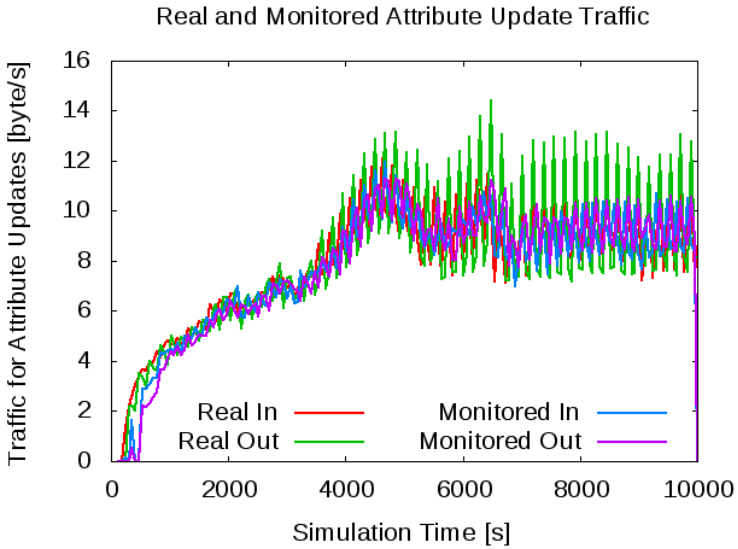
- Index traffic small (10b/s)
- Query traffic small (6b/s)

But:

- Strong imbalance
- Load matches peer capacity







Summary on Monitoring Solution SkyEye.KOM

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Monitoring scope:

- System-specific information: statistics on system-wide metrics
- Peer-specific information: detailed view on capabilities of individual peers

For all structured P2P overlays

- Covered by KBR-function:
  route(msg, key), lookup(key)
- Usable by all functional layers in the P2P system

Features:

- Overlay-independency
- Robustness, churn resistance
- No overloaded peer
- Supporting peer heterogeneity
- Low overhead

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

Peer-to-Peer Cloud Systems

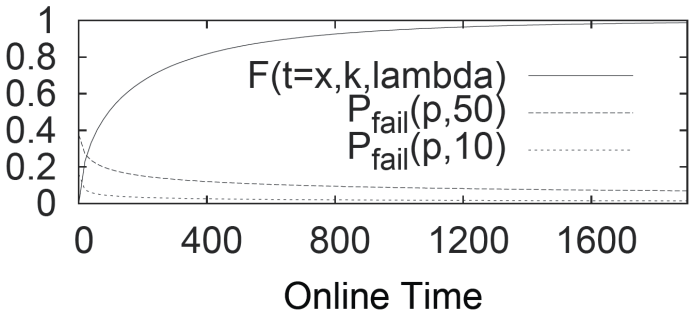
– Reliable Resource Reservation in P2P

▶ Reliable Reservations in Unreliable P2P Systems

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

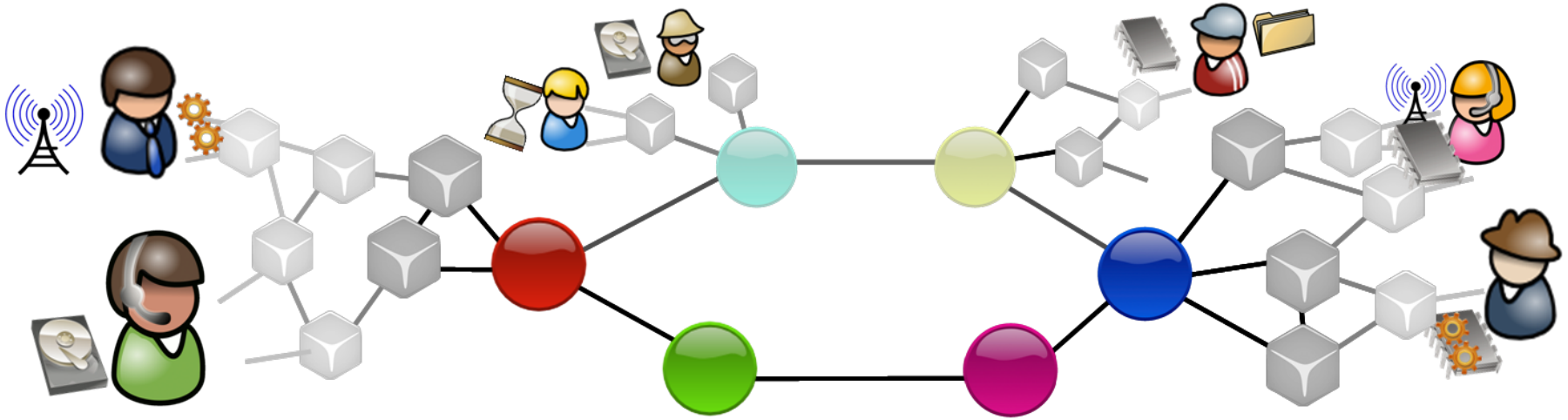
Goal:

- Reliable resource provision (e.g. 2 months)
- Use the resources of unreliable nodes
  - Average lifetime 2 days

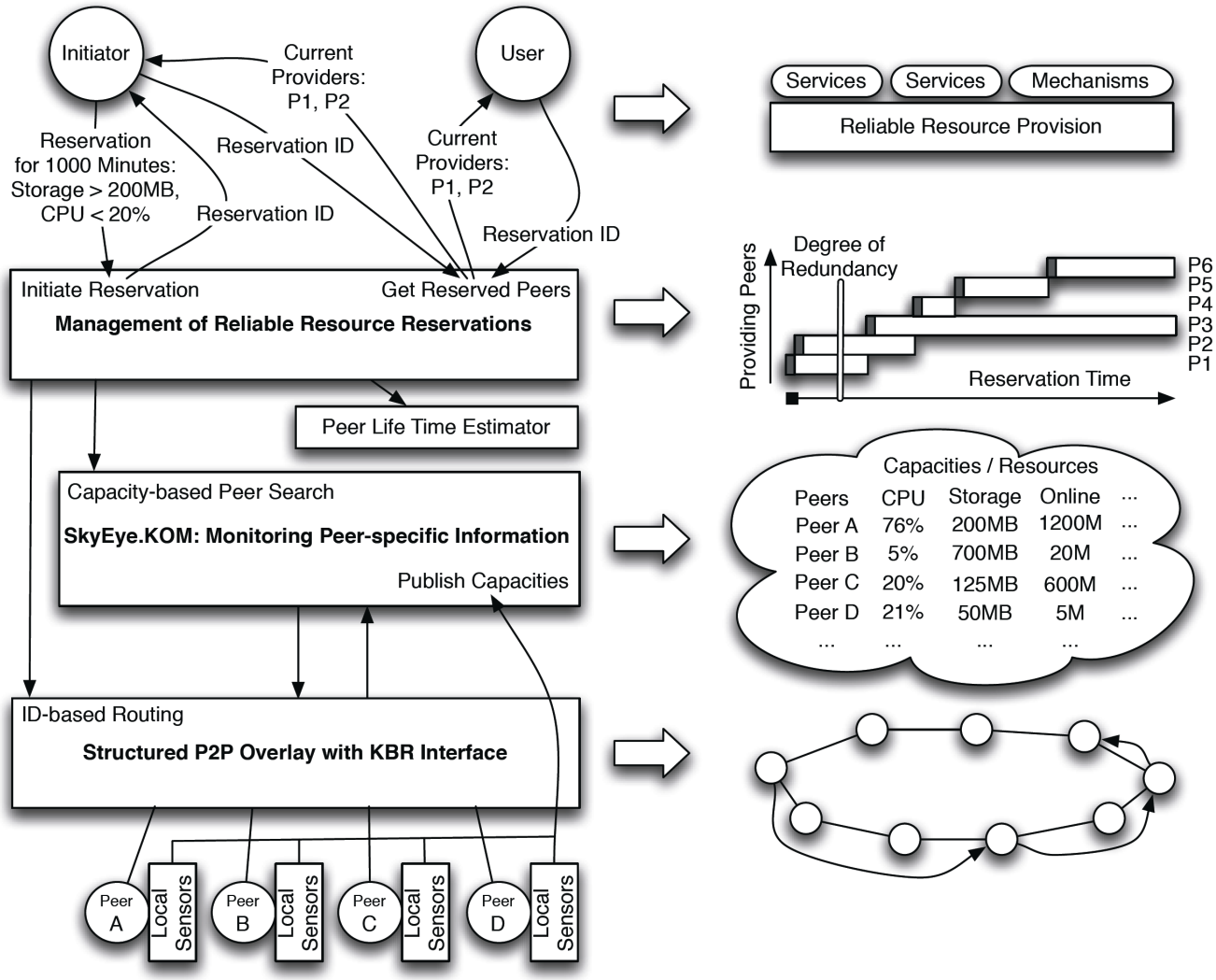


Issue:

- Finding peers that provide resources in desired quantity
- P_fail(p,10): probability that a peer will fail in the next 10 minutes
  - Depending on its current life time
- Peers are short living



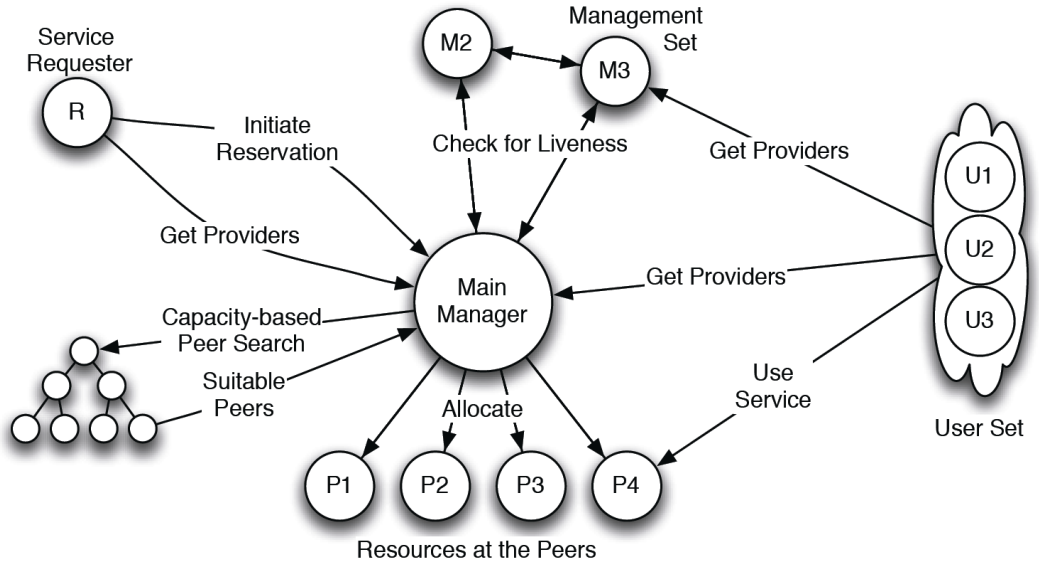
P3R3O: P2P Protocol for Reliable Resource Reservations



See K. Graffi et al., "Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems" In: IEEE International Parallel and Distributed Systems '10 (IEEE ICPADS'10), December 2010.

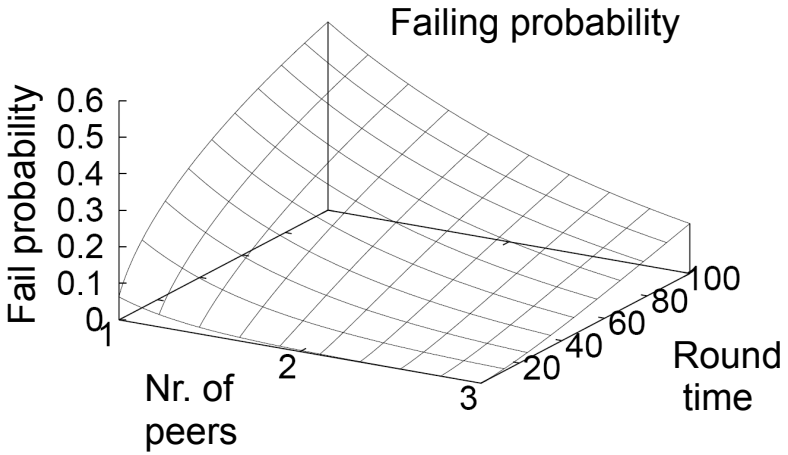
Elements of the Solution

1. Setting up a reservation (by service requester)
   - Reservation: resource constraints (Cons_i), reservation time (RsvTime) and degree of redundancy (RsvDR)
   - Responsible peer (main manager) identified by ID
2. Maintaining the Management Set
   - Main manager supervised by backup peers
   - Manages se

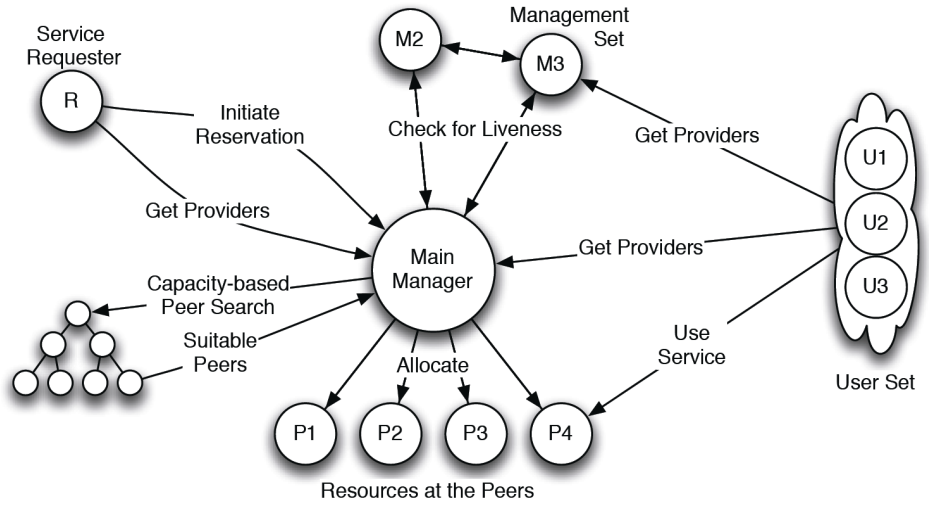


Tasks of the Main Reservation Manager

3. Deriving the Success Probability for a Round
   - Assuming that Resource Provider Set is not empty
   - Check their lifetimes, calculate the round survival probability
     - If nr. of surviving peers > RsvDr → wait until next round, step 3
     - If too less peers surviving → step 4



4. Add new resource providers
   - Determine number of missing peers
   - Find new suitable peers (with SkyEye)
   - Use in Resource Provider Set



Evaluation of P3R3O

Main question:

- Can a p2p system host long services
- although peers fail after a short time?

Reservation

- Resource request:
  - 500 MB storage, 200 kb/s upload
  - Reservation time: ca. 200 days
- Success prob.: RsvDR (here 0.9)

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

Approaches to determine missing peers

- Redundant Peer Approach (RPA):
  - add more redundancy
  - i.e. add 1,..,5 more peers than needed
- Probabilistic Buffer Approach (PBA):
  - decrease the tolerated failure risk
  - i.e. risk of failure: 0.09 – 0.01

Simulation environment

- Chord
- N=1000/2000/4000 nodes
- Num. of reservations: 5% of N
- SkyEye.KOM used for finding peers

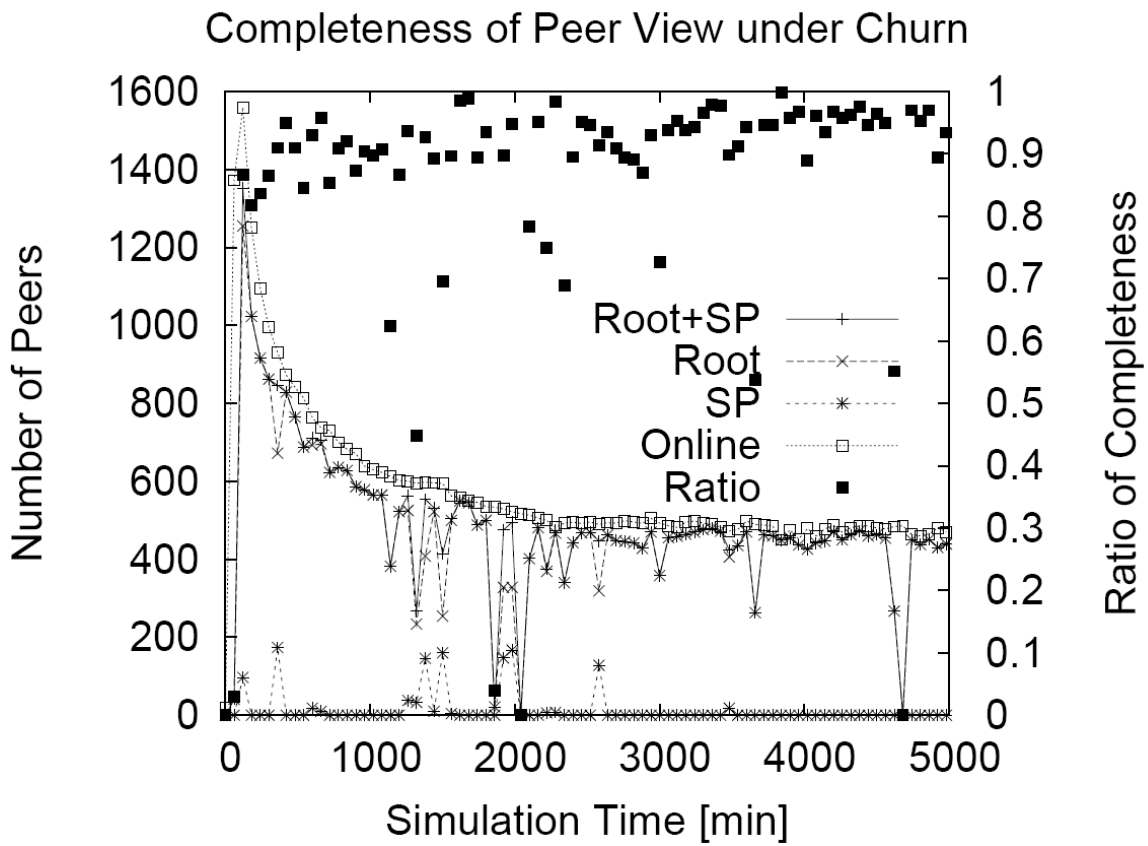
Completeness of Monitoring View

Completeness of monitoring view

- Limited due to node capacities
- Monitoring load might be larger

Observation

- Support peers carry the load
- Good matching of load to strong peers
- Ratio of Completeness in avg. > 0.9

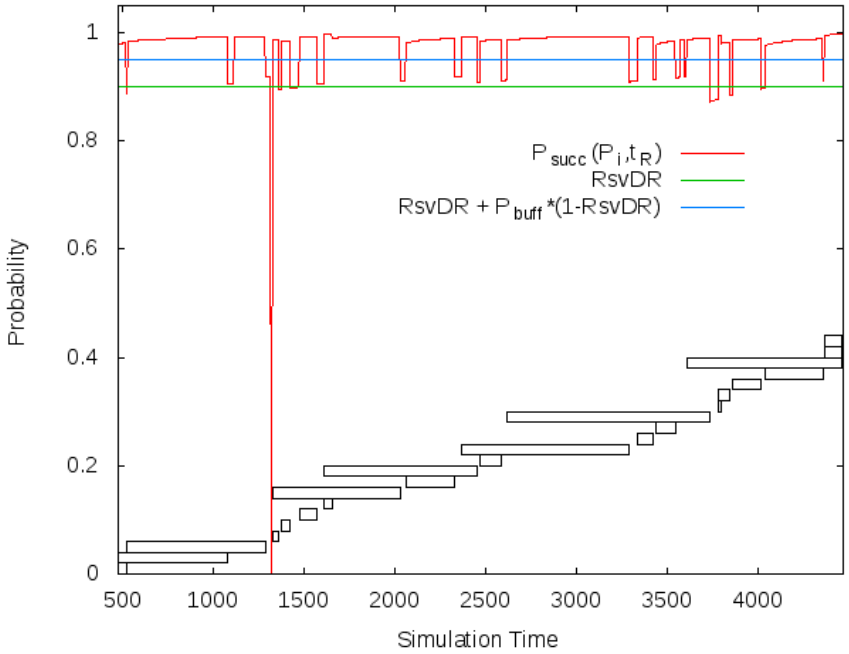


Some examples

PBA, success probability 0.93

Both goals are missed:

- Success probability of 0.93 not held
- Total reservation fails

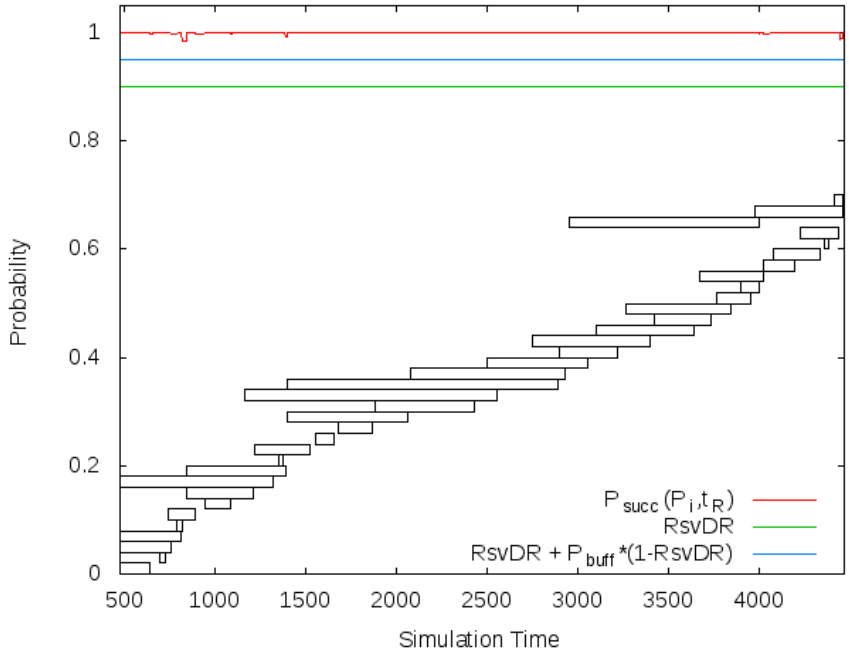


RPA, N=2

Success probability 0.90

Both goals reached:

- Success probability 0.9 held
- Reservation successful

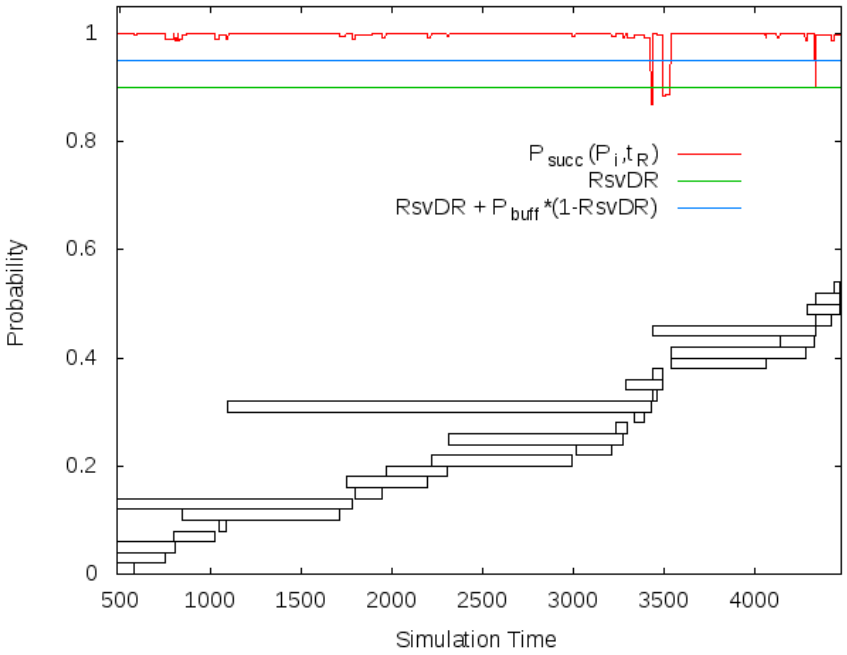


Some more examples

PBA, success probability 0.99

Both goals are missed:

- Success probability of 0.99 not held
- However, reservation successful

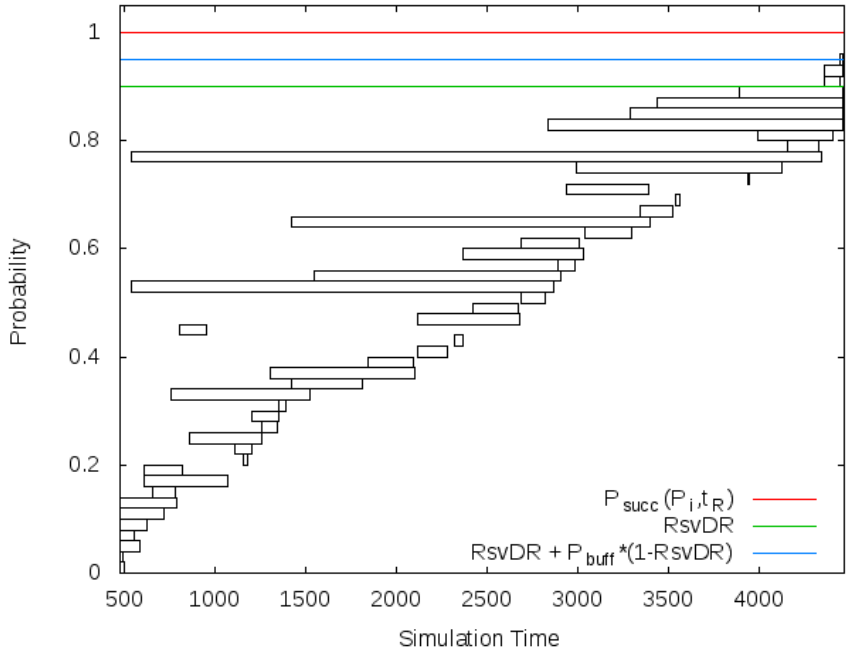


RPA, N=5

Success probability 0.90

Both goals reached:

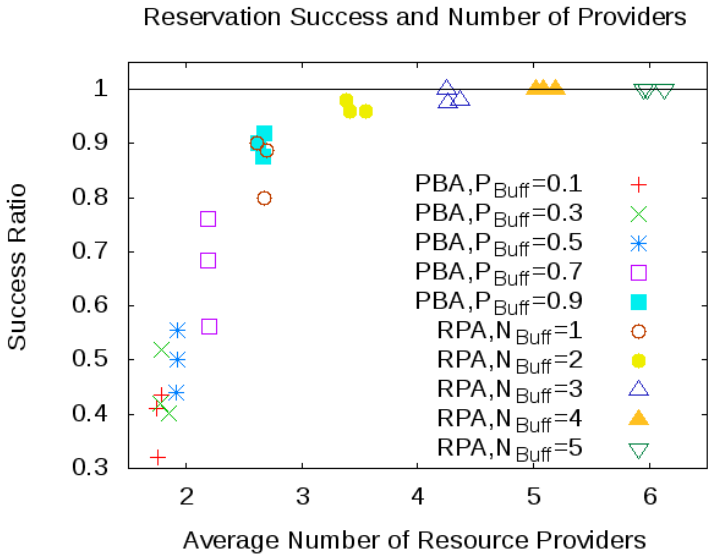
- Success probability 0.9 held
- Reservation successful



Reservation Performance and Costs

Success ratio of reservations

- Redundant Peer Approach (RPA) better than Probabilistic Buffer Approach (PBA)
- With redundancy 5 or 6:
  - 100% success of reservations

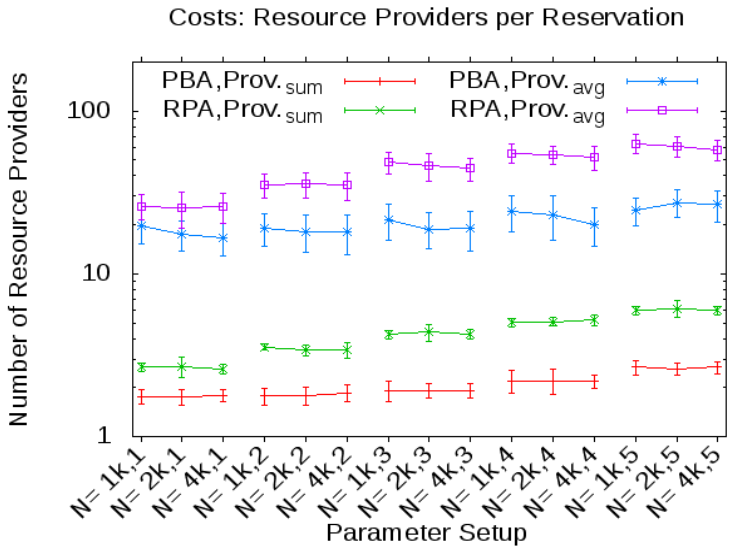


Nr of providers per reservation

- With RDA, redundancy = n (n=5)
  - In average n+1 → 6
  - In total 10 * (n+1) → 60-70
- With PBA: far less

Conclusion:

- Additional redundancy leads to better results



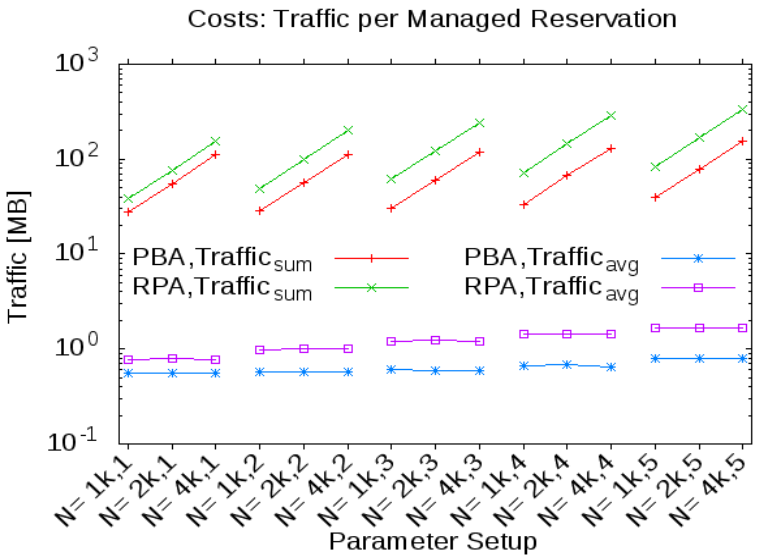
Time and Traffic Costs

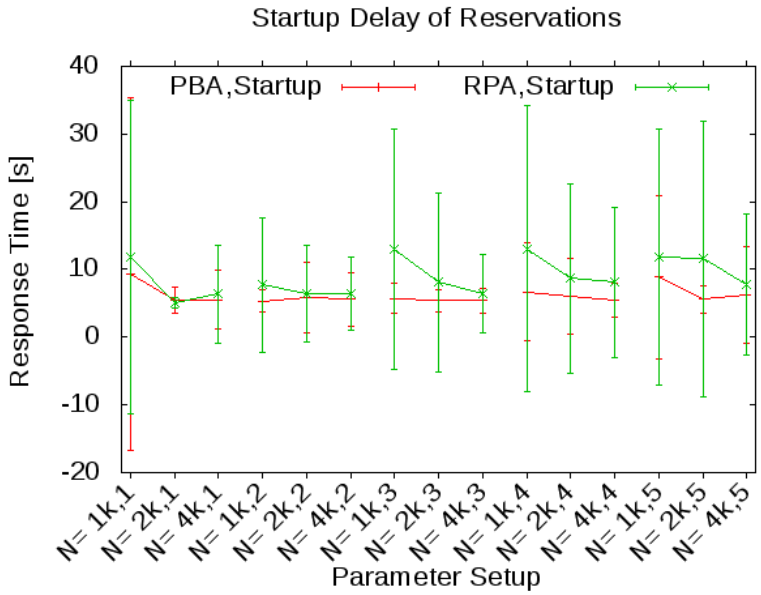
Traffic overhead

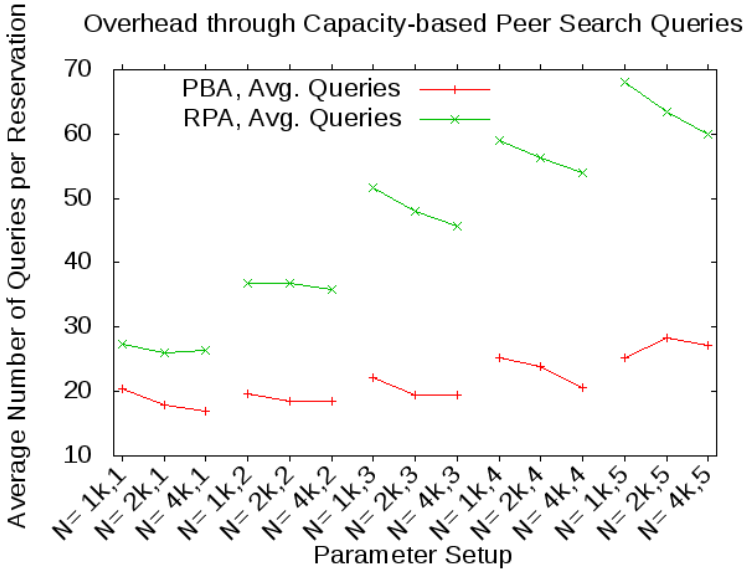
- In average per service
- RPA n=5: ca 2 MB over 200 days
- PBA, success 0.99:
  0.8 MB over 200 days

Starting delay

- Finding first provider set
- In average < 15 sec







Summary Reliable Resource Reservations

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Monitoring peer-specific information

- Fat tree structure
- Enables capacity-based peer search

Reliable resource reservation

- Distributed reservation management
- With high redundancy (> 5) and sufficient resources: 100 % success

Research questions

- Stateful services – how to handle state?
- Security – how to cope with attackers
- Adaptivity – monitoring load should adapt to request patterns