The P-Grid Peer-to-Peer System

http://www.p-grid.org/

Manfred Hauswirth
Laboratoire de Systèmes d'Information Répartis
École Polytechnique Fédérale de Lausanne

Credits: The P-Grid Team

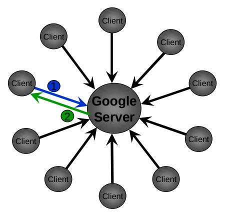
Karl Aberer	**technology and concepts, all subprojects**
Philippe Cudré-Mauroix	**caching, economic concepts, semantics**
Anwitaman Datta	**replication, updates**
Zoran Despotovic	**trust**
Manfred Hauswirth	**technology, concepts, impl., updates, semantics**
Magdalena Punceva	**simulations**
Roman Schmidt	**implementation**
Jie Wu	**application: distributed search engine**

Overview

- What is P2P?
  - Levels of decentralization
  - Self-organization
  - P2P properties
  - Related approaches
- P-Grid basic concepts
  - Searching
  - Building up a P-Grid
- Research issues in P-Grid
  - Request load balancing
  - Trust
  - Updates
  - Semantic gossiping
  - Economic models
  - Threats to P-Grid in the real world

Centralized Information Systems

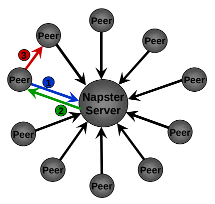
- Web search engines
- Example: Google
  - April 2001: 8000 servers (22-40 GB each), 11 million visitors/month, 1-2 Terabytes information



- Pros
  - Fast response time
  - Global ranking of web pages
- Cons
  - Probably OK to start a company for Web search engine
  - But do we want this for every major application ?
  - Infrastructure, administration, cost

(Semi-)Decentralized Information Systems

- P2P file sharing
- Example: Napster
  - Feb 2001: 2.79 billion songs traded, 220 songs per user, 2 Mio songs (avg. file size 5 MB, approx. 10 TeraByte of data), 1.57 Mio. users
  - Infrastructure: about 100 servers

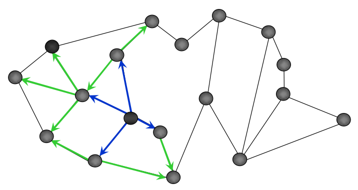


Lessons Learned from Napster

- Pros
  - exploit (unused, replenishable) resources at nodes
  - decentralization of cost and administration
  - avoids the most serious scalability bottleneck by decentralization (i.e., file transfer)
  - take advantage of the users annotating music (search)
  - set up a very large scale information system without heavy investment (as e.g., Google)
  - keeping content where it is created
  - exploit positive externalities resulting from copying of information
- Cons
  - copy copyrighted material
  - Still Napster is logically centralized since the location of data objects relies on a server ⇨ could be shut down

Fully Decentralized Information Systems

- P2P file sharing
- Example: Gnutella
  - August 2000: 40.000 nodes, 3 Mio files
  - Infrastructure: no servers



- Pros
  - Fully decentralized
  - Little global knowledge
- Cons
  - Requests are broadcasted
  - Free-riding
  - No mathematical model

Decentralization – Self-Organization

- Decentralization
  - strategy to avoid performance bottlenecks (scalability), single points of failure, points for legal attack
  - no central coordination, no central database, no peer has a global view of the system
- Self-organization
  - global behavior emerges from local interactions
  - cooperation/coordination without central control
  - Decisions based on local (or missing) information (autonomy or non-determinism)
- P2P: Towards symmetric system architectures with some desired (or observed) global behavior

(Non-technical) Motivations for using P2P

- Exploiting free resources
  - Famous example: SETI@Home
- Sharing costs
  - New economic models (also of interest to large companies)
- Autonomy
  - No dependence on central server
- Self-maintenance
- Anonymity
  - No registration at central server
  - Though non-negligible security problems (open environment)
  - Legal protection, but …

P2P is not new !

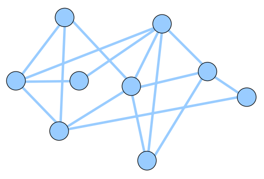
- The original Internet was designed as a P2P system
  - any 2 computers could send packets to each other
    - no firewalls / no network address translation
    - no asymmetric connections (V.90, ADSL, cable, etc.)
  - the back-then "killer apps" FTP and telnet are C/S but anyone could telnet/FTP anyone else
  - servers acted as clients and vice versa
  - cooperation was a central goal and "value": no spam or exhaustive bandwidth consumption
- Typical examples of "old-fashioned P2P":
  - Usenet News
  - DNS
- The emergence of P2P can be seen as a renaissance of the original Internet model

What is new ?

- Clay Shirkey (The Accelerator Group):
  - "Peer-to-peer is a class of applications that take advantage of resources—storage, cycles, content, human presence—available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, peer-to-peer nodes must operate outside the DNS and have significant or total autonomy of central servers."
  - P2P "litmus test:"
    - Does it allow for variable connectivity and temporary network addresses?
    - Does it give the nodes at the edges of the network significant autonomy?
- P2P ~ an application-level Internet on top of the Internet

What is P2P ?

- Every participating node acts as both a client and a server ("servent")
- Every node "pays" its participation by providing access to (some of) its resources
- Properties:
  - no central coordination
  - no central database
  - no peer has a global view of the system
  - global behavior emerges from local interactions
  - all existing data and services are accessible from any peer
  - peers are autonomous
  - peers and connections are unreliable



Types of P2P Systems

- E-commerce systems
  - eBay, B2B market places, B2B integration servers, …
- File sharing systems
  - Napster, Gnutella, Freenet, …
- Distributed Databases
  - Mariposa, …
- Networks
  - Arpanet
  - Mobile ad-hoc networks

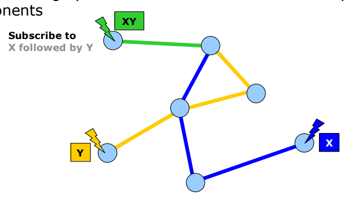
Related Approaches

Related distributed information system approaches:

- Event-based systems
- Push systems
- Mobile agents
- Distributed databases

Event-based (publish/subscribe) Systems

- System model
  - Components (peers) interact by generating and receiving events
  - Components declare interest in receiving specific (patterns of) events and are notified upon their occurrence
  - Supports a highly flexible interaction between loosely-coupled components

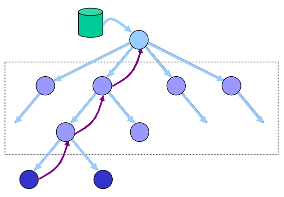


Event-based vs. Peer-to-Peer

- Common properties:
  - symmetric communication style
  - dynamic binding between producers and consumers
- Subscription to events ~ "passive" queries
  - EB: notification
  - P2P: active discovery
- Subscription language supports more sophisticated queries and pattern matching (event patterns with time dependencies)
- Event-based systems typically have a specialized event distribution infrastructure
  - EB: 2 node types, P2P: 1 node type
  - EB infrastructure must be deployed

Push Systems

- A set of designated broadcasters offer information that is pre-grouped in channels (weather, news, etc.)
- Receivers subscribe to channels of their interest and receive channel information as it is being "broadcast" (timely distribution)
- Receivers may have to pay prior to receiving the information (pay-per-view, flat fee, etc.)
- Pull → push

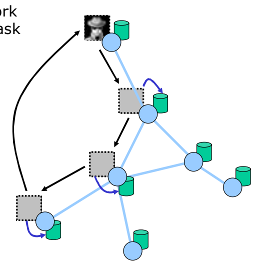


Push Systems vs. Peer-to-Peer

- Asymmetric communication style (P2P: symmetric)
- Focus is on timely data distribution not on discovery
- Filtering may be deployed to reduce data transmission requirements
- Subscription to channels is prerequisite
- Producer/consumer binding is static
- Push systems require a specialized distribution infrastructure
  - Push: 3 node types, P2P: 1 node type
  - Push infrastructure must be deployed

Mobile Agents

- A mobile agent is a computational entity that moves around in a network at its own volition to accomplish a task on behalf of its owner
  - can cooperate with other agents
  - "learns" ("Whom to visit next?")
- Mobility (heterogeneous network!)
  - Weak: code, data
  - Strong: code, data, execution Stack

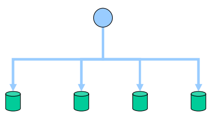


Mobile Agents vs. Peer-to-Peer

- Very similar in terms of search and navigation
  - P2P: the peers propagate requests (search, update)
  - MA: the nodes propagate the agents
  - Mobile agent ~ "active" query
- Mobile agent systems require a considerably more sophisticated environment
  - mobile code support (heavy)
  - security (protect the receiving node from malicious mobile agents and vice versa)
- In many domains P2P systems can take over
  - more apt for distributed data management
  - less requirements (sending code requires much bandwidth, security, etc.)

Distributed Databases

- Fragmenting large databases (e.g., relational) over physically distributed nodes
- Efficient processing of complex queries (e.g., SQL) by decomposing them
- Efficient update strategies (e.g., lazy vs. eager)
- Consistent transactions (e.g., 2 phase commit)
- Normally approaches rely on central coordination



Distributed Databases vs. Peer-to-Peer

- Data distribution is a key issue for P2P systems
- Approaches in distributed DB that address scalability
  - LH* family of scalable hash index structures [Litwin97]
  - Snowball: scalable storage system for workstation clusters [Vingralek98]
  - Fat-Btree: a scalable B-Tree for parallel DB [Yokota99]
- Approaches in distributed DB that address autonomy (and scalability)
  - Mariposa: distributed relational DBMS based on an underlying economic model [Stonebraker96]
- P2P Data Management has to address both scalability and autonomy

Usage Patterns to Position P2P

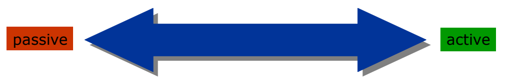
Discovering information is the predominant problem

- Occasional discovery: search engines — P2P, MA
  - ad hoc requests, irregular
  - E.g., new town — where is the next car rental?
- Notification: event-based systems — push
  - notification for (correlated) events (event patterns)
  - E.g., notify me when my stocks drop below a threshold
- Systematic discovery: P2P systems — search engines, MA
  - find certain type of information on a regular basis
  - E.g., search for MP3 files of Jethro Tull regularly
- Continuous information feed: push systems — event-based
  - subscription to a certain information type
  - E.g., sports channel, updates are sent as soon as available

The Interaction Spectrum

Event-based systems
Push systems

Mobile agents
Peer-to-peer systems



State-of-the-Art Systems

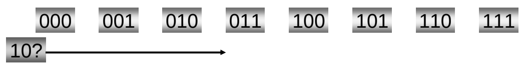
- Napster
- Gnutella
- Freenet
- OceanStore
- Farsite
- FastTrack
- Tornado
- Chord
- CAN
- P-Grid

Main P2P Design Requirements

- Resource discovery
- Managing updates
- Scalability
- Robustness and fault tolerance
- Trust assessment and management

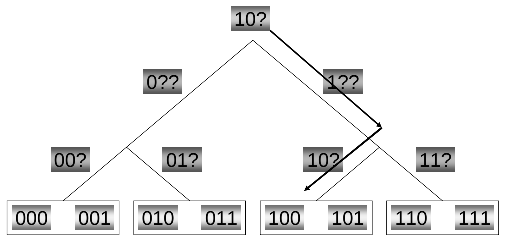
Data Access Structures (List)

- Given: a search request (e.g., a name)
- Find all data objects that correspond to this request quickly (e.g., having the name)
- Sequential search does not scale
  - N objects: N steps



Data Access Structures (Tree)

- Search tree (Prefix Tree)
- Scales: N objects: $\log_2(N)$ steps

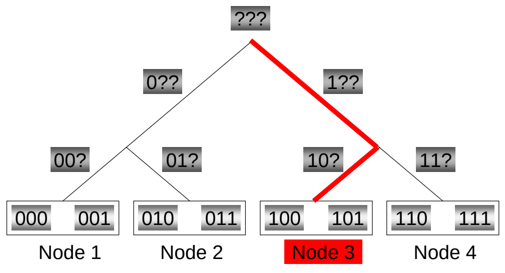


Scalable Data Access Structures – 1

- Assume number of data objects >> storage of one node
  - Distributed storage
- Given a data access structure
  - Size of data access structure = number of data objects
  - Therefore: Size of data access structure >> storage of one node
- Problem: where to store ?

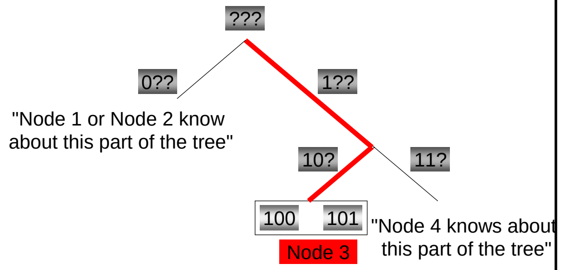
Scalable Data Access Structures – 2

- Trick: each node remembers its own path

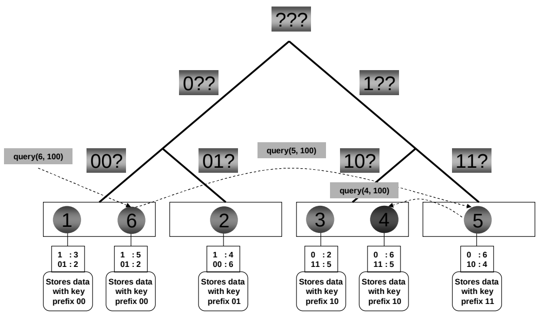


Scalable Data Access Structures – 3

- Trick: each node remembers its own path


???
0??
1??
"Node 1 or Node 2 know about this part of the tree"
10?
11?
100
101
Node 3
"Node 4 knows about this part of the tree"

P-Grid Queries


???
0??
1??
query(6, 100)
00?
01?
query(5, 100)
10?
11?
query(4, 100)
1
6
2
3
4
5
1 : 3
01 : 2
1 : 5
01 : 2
1 : 4
00 : 6
0 : 2
11 : 5
0 : 6
11 : 5
0 : 6
10 : 4
Stores data with key prefix 00
Stores data with key prefix 00
Stores data with key prefix 01
Stores data with key prefix 10
Stores data with key prefix 10
Stores data with key prefix 11

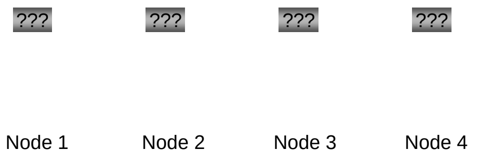
SDAS Discussion

- Scalable Data Access Structures
  - Require only $Log_2(N)$ storage at one node
  - Support $Log_2(N)$ search
  - Are therefore scalable in N (beyond $N=10^{10}$ as in Google)
- Idea found in
  - OceanStore
  - DNS
  - Parallel and distributed DBMS

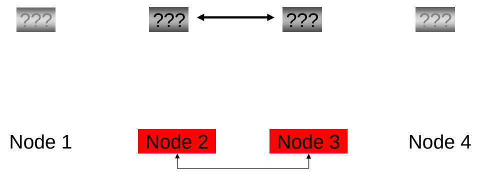
Construction of SDAS ?

- Standard methods
  - Each node has an identifier, compute position in the tree from the identifier
  - Structure of tree depends on distribution of identifier. What's the connection ?
    - Nodes can no more choose which data they want to store and which requests they want to answer (autonomy)
  - Each node asks a coordinator for its position in the tree
    - Coordinator is the bottleneck
- Better idea
  - Replace the coordinator by a random process

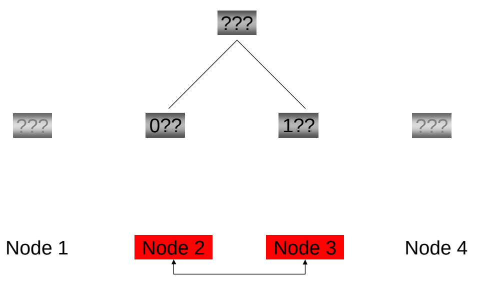
Random Meetings



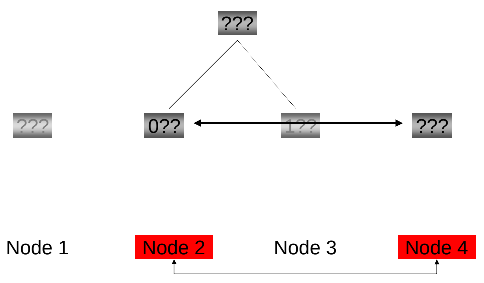
Random Meetings



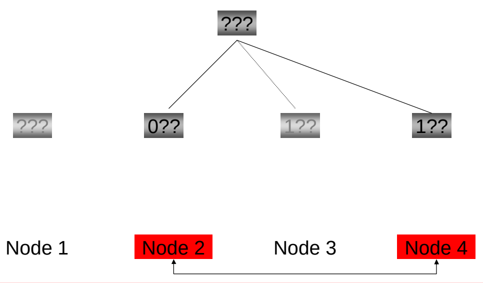
Random Meetings



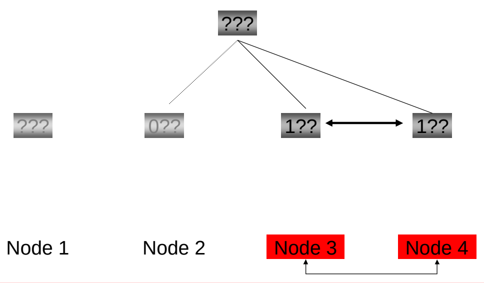
Random Meetings



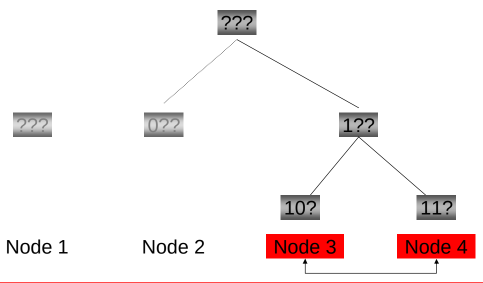
Random Meetings



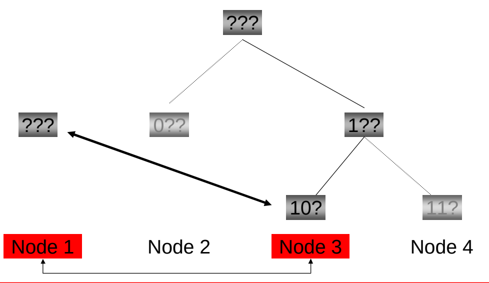
Random Meetings



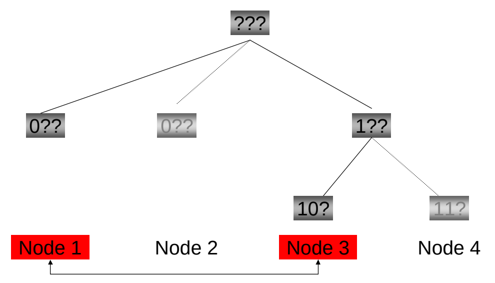
Random Meetings



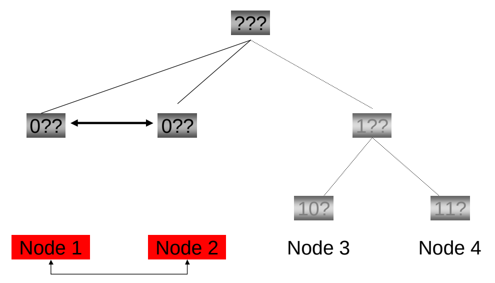
Random Meetings



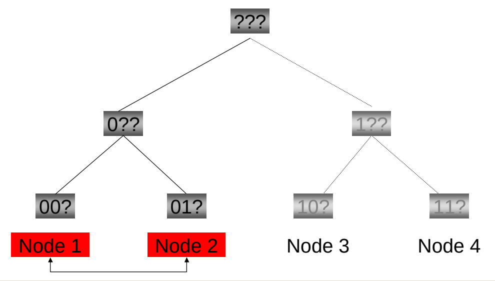
Random Meetings



Random Meetings

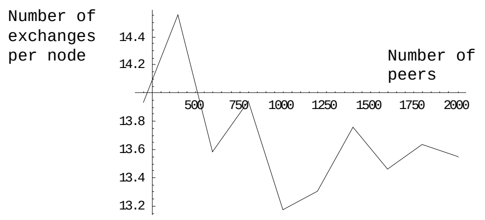


Random Meetings



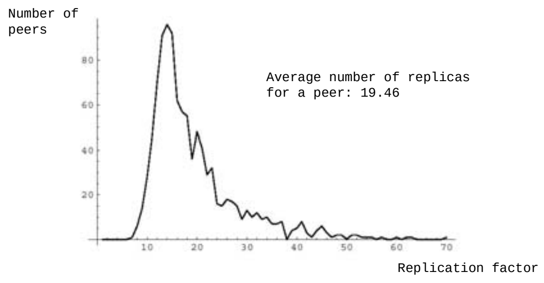
Convergence ?

(n = 200…2000, k = 6, recmax =2, refmax = 1)



Replication ?

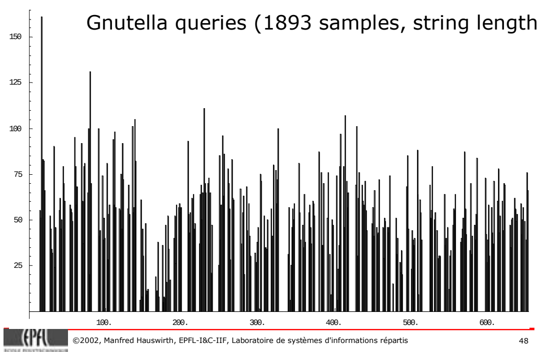
(n = 20000, k = 10, refmax = 20)



Average number of replicas for a peer: 19.46

Non-uniform Data Distribution

Gnutella queries (1893 samples, string length 4)

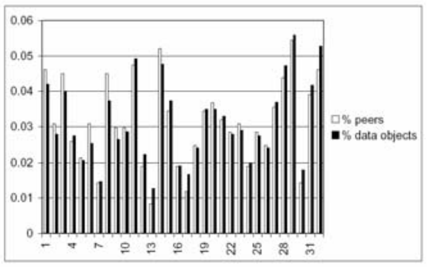


Algorithm for Balancing Storage Load

- With skewed data distributions peers would store different amounts of data items ⇨ uneven distribution of load
- Every peer stores some data initially
  - Representative sample
- Modified "meeting algorithm"
  - Node extends its path only if #data items > ε
  - Otherwise only data exchange (duplicate generation)
  - Maintain data distribution in exchange (requires care)
- Result
  - Each leaf node supported by a comparable number of nodes
  - Each node gets the same load (depending on ε)

Result

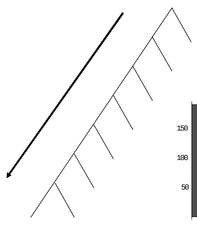
- … and still converges quickly

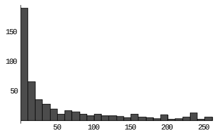


Unbalanced Trees

- Problem: with the algorithm for balancing the storage load, the P-Grid tree will become unbalanced, i.e., deeper ⇨ more work to do for answering search requests

Worst case: N steps !

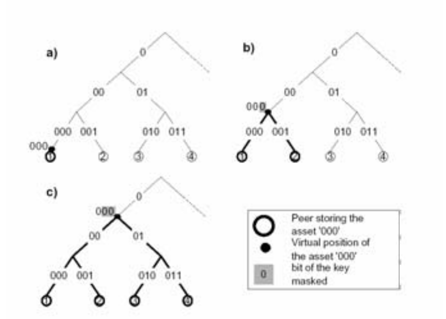




Analysing required messages

- Depth of tree does not matter, number of messages when searching is important
- Theorem: No matter what shape the P-Grid tree has, if the probability for a reference to another node occuring in the reference lists is constant, then the number of messages required is O($\log_2$(N))
- Constant probability can be achieved by merging the reference lists of peers whenever possible

Request Load Balancing – 1


a)
b)
c)
0
00
01
000
001
010
011
Peer storing the asset '000'
Virtual position of the asset '000'
bit of the key masked

Request Load Balancing – 2

- Gamma: popularity threshold
  - if access measure > gamma a new popularity level is granted
- Optimum: 0.2 (< 0.2: too much competition)

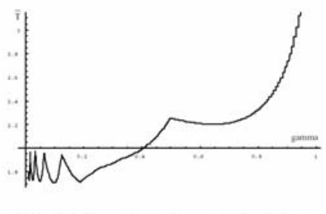


Figure 5: Average Response Time, Finite Capacity

Aspects of Self-Organization in P-Grid

- Nodes decide locally
  - on their position in the search tree when they meet
  - whether to deepen a search tree based on storage load
  - whether to actively replicate a data item based on request load
- Nodes balance through local operations
  - probability of reference distribution (required for search efficiency)
  - probability of replication of data objects
- Only global "agreements" are on
  - What are the search requests
  - P-Grid organisation

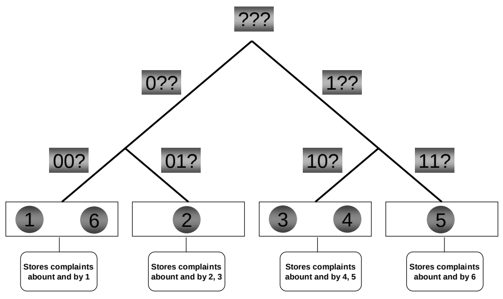
Practical Aspects of P-Grid

- Implementation exists: feasible
- Analysis shows that overhead is reasonable for typical setting (replication, tree structure)
- An efficient update mechanism based on gossiping has been developed
- More complex queries can be supported (regular expressions, paths)

Trust Management based on Reputation

- Approach
  - Record complaints by peers
  - Build a decentralized data warehouse based on P-Grid
  - Compute average number of complaints
  - Retrieve from the data warehouse all complaints on (and by) a peer
  - Also assess the trustworthiness of the peers reporting theses numbers
  - Apply a weighting formula and decide
- Result
  - Even with a large fraction of cheaters (25% are cheating 25% of the time) they can be reliably recognized

Using P-Grid to store Trust Data


???
0??
1??
00?
01?
10?
11?
1
6
2
3
4
5
Stores complaints abount and by 1
Stores complaints abount and by 2, 3
Stores complaints abount and by 4, 5
Stores complaints abount and by 6

Updates in P-Grid

- Most P2P systems consider data to be read-only
- The goal is not to achieve complete consistency but rather to know what is the probability of a correct answer given certain model parameters
- Scenarios:
  - A query occurs during an update
  - A peer is online while an update is processed
  - A peer is offline while an update is processed
  - A peer crashes or fails
  - The communication with a peer is temporarily disrupted

Types of Updates in P-Grid

- New peer joins P-Grid: all the peer's data must be communicated to the responsible peers and their replicas
- New data item is inserted
- Existing data item is updated
- Due to a re-organisation in the P-Grid a new peer becomes responsible for a certain data item

P-Grid's Update Algorithm (Push Phase)

- *p* receives an update request *update(K, V, p_f, R_f)* with *K*: key of the data to update, *V*: new value, *p_f*: requesting peer, *R_f* replica list of requesting peer
- *p* propagates the update request to *R_p* \ *R_f*
- *p* sends *R_p* \ *R_f* to *f* so that *f* learns about new replicas
- *p* discovers additional replicas from *R_f* \ *R_p*, contacts them and updates its replica list: *R_p* = *R_p* ∪ *R_f*
- If *R_p* did not change for the last *L* update requests (or time period *T* or self-tuning parameter), then ascend the P-Grid search tree to find new replicas and contact them.
- If a peer *r_p* cannot be contacted then retry:
  - Check whether the ping counter has exceeded its maximum or the the maximum offline period of replica peers has expired.
  - If one of these 2 conditions is true remove *r_p* from *R_p*.
  - If *r_p* becomes online again and requests a state update of or other peers notify the peer that did the unsuccessful ping, then *r_p* is put into *R_p* again.

P-Grid's Update Algorithm (Pull Phase)

- Scenario 1: If a peer has been off-line it must try to get a consistent view of the data again.
  - *p* contacts some of its replicas randomly and asks for all updates for the time it had been offline.
- Scenario 2: A peer is online but communication is temporarily disrupted.
  - This is more complicated because the peer may not recognize that its communication with the rest of the P-Grid network is broken.
  - To get around: contact other peers (specifically its replicas) at regular intervals
    - Randomly ping a replica *r_p* from the list of active replicas *R_p*
    - Check whether the ping counter has exceeded its maximum or the the maximum offline period of replica peers has expired.
    - If one of these 2 conditions is true remove *r_p* from *R_p*.
    - If *r_p* becomes online again and requests a state update of or other peers notify the peer that did the unsuccessful ping, then *r_p* is put into *R_p* again.

P-Grid's Update Algorithm (Discussion)

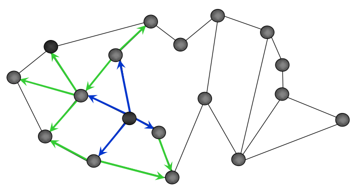
- Availability of peers is modelled as a Poisson process
- Low online probability (10%-30%)
- Tries to balance consistency with effort
- Overhead of additional messages is reasonably low

Semantic Bottlenecks

- Physical scalability is nice (and can be solved), but the really hard problem is semantic interoperability
- Reminder: Napster
  - not only centralized data lookup
  - but also the schema to describe data objects (bottleneck)
  - decentralized the actual annotation of data (good)
- Can we decentralize the task of agreeing on a common schema ?
  - This would solve one of the hardest problems in information systems: semantic interoperability

Peer-to-Peer Networks (Gnutella)

- Content: mainly music files
- Schema: file name
- Query: keywords



Semantics as a Network Phenomenon

- Generate semantics locally
  - Communication among nodes
  - Decentralization, scalability
- Study evolution of semantics globally
  - Emergent properties of (social) networks
- Study the problem for well-accepted applications using standard platforms
  - Experimental approach

Proposal: Semantic Gossiping

• Content:	Media, publications, scientific data
• Schema:	XML DTD or schema
• Query:	XPath or XQuery
• Network:	Broadcast of search requests (gossip)

"From P2P to Semantic P2P"

Scenario Semantic Gossiping: Local View

```
Q1=
<Title>$s/Title</Title>
FOR $s IN /Song
WHERE $s/Size<1000000
```

Schema 1

```
<Song>
  <Title>
    21st century
    schizoid man
  </Title>
  <Interpreter>
    King Crimson
  </Interpreter>
  <Size>
    23456789
  </Size>
</Song>
```

```
T12=
<Song>
<Title>$t/Name</Title>
<Interpreter>$t/Author
  </Interpreter>
  <Size>$t/Bytes</Size>
</Song>
FOR $t IN /Title
```

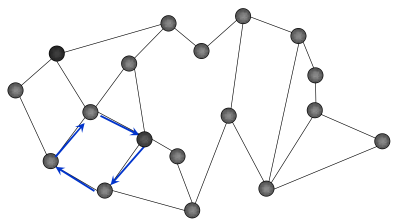
Schema 2

```
Q2=
<Title>$s/Title</Title>
FOR $s IN T12
WHERE $s/Size<1000000
```

```
<Title>
  <Name>
    21st century
    schizoid man
  </Name>
  <Author>
    King Crimson
  </Author>
  <Bytes>
    23456789
  </Bytes>
</Song>
```

How to detect an Agreement ?

- Start with query Q
- Translate it along the known "semantic connectors"

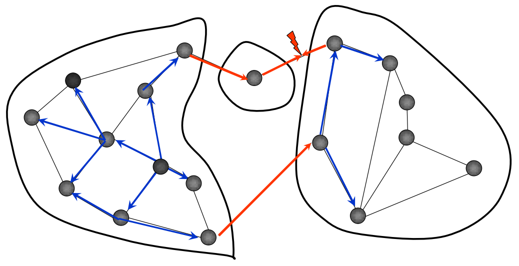


Semantic Kernels

- After a few translations the query returns
- T1(T2(T3(T4(Q))))
- In general: T1(T2(T3(T4(Q)))) != Q
- But there always exists Q' sucht that
- T1(T2(T3(T4(Q'(Q)))))= Q'(Q))
- Therefore an agreement exists on this query !

- Research:
  - How to efficiently detect the "agreements"
  - How to quickly improve the level of agreement
  - How to use agreements for seatching across semantic domains

Scenario Semantic Gossiping: Global View



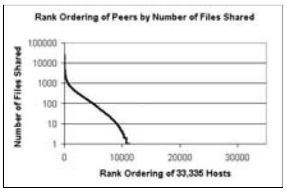
Economic Models

- Introduction of economic models to encourige resource sharing
  - Example: Mojo Nation
  - Each peer has to provide some of its resources to be eligible to search and download in the P2P system
  - Prevent "backbone" P2P systems (e.g., Gnutella)
  - Challenges
    - Security (authenticity)
    - Common "currency"
- Study effects of economic models on the self-organization of P2P systems (besides the algorithmic organization principles)

Free-riding on Gnutella [Adar00]

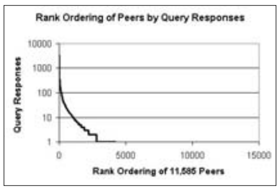
- 24 hour sampling period:
  - 70% of Gnutella users share no files
  - 50% of all responses are returned by top 1% of sharing hosts
- A social problem not a technical one
- Problems:
  - Degradation of system performance: collapse?
  - Increase of system vulnerability
  - "Centralized" ("backbone") Gnutella ⇔ copyright issues?
- Verified hypotheses:
  - H1: A significant portion of Gnutella peers are free riders.
  - H2: Free riders are distributed evenly across domains
  - H3: Often hosts share files nobody is interested in (are not downloaded)

Free-riding Statistics - 1 [Adar00]



- H1: Most Gnutella users are free riders
- Of 33,335 hosts:
  - 22,084 (66%) of the peers share no files
  - 24,347 (73%) share ten or less files
  - Top 1 percent (333) hosts share 37% (1,142,645) of total files shared
  - Top 5 percent (1,667) hosts share 70% (1,142,645) of total files shared
  - Top 10 percent (3,334) hosts share 87% (2,692,082) of total files shared

Free-riding Statistics - 2 [Adar00]

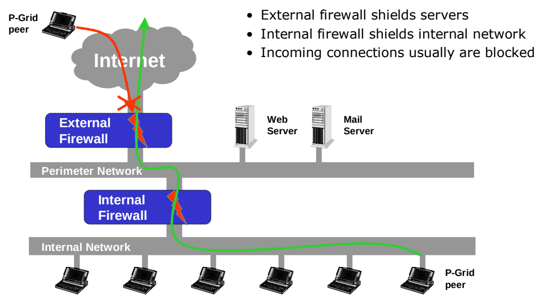


- H3: Many servents share files nobody downloads
- Of 11,585 sharing hosts:
  - Top 1% of sites provide nearly 47% of all answers
  - Top 25% of sites provide 98% of all answers
  - 7,349 (63%) never provide a query response

Dynamic IP Addresses/Mobility

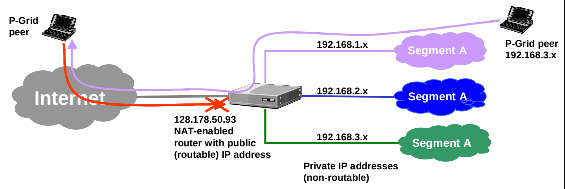
- Typically hosts have changing IP addresses
  - Dynamic Host Configuration Protocol (lease time)
  - Host mobility (physical mobility)
- No problem for pull-based P2P systems
  - New peer initiates a "permanent" connection to other peer(s) that route(s) requests to the new peer via this connection (for example, Gnutella).
  - No "permanent" connection ⇨ problem
- BIG problem for pushed-based P2P systems
  - Peers actively try to connect via a new connection (for example, P-Grid)
  - What if the IP address has changed in the meantime?
  - Location transparency? Migration transparency?

Firewalls



- External firewall shields servers
- Internal firewall shields internal network
- Incoming connections usually are blocked

Network Address Translation (NAT)



- NAT translates private (non-routable IP) addresses into public (routable) ones
- Unidirectional concept (from Intranets to Internet)
- Bi-directional possible, but difficult and thus usually not configured
- Many protocols are not NAT-friendly: VoIP, RTP, RTCP, IPSec, P-Grid, etc.

Authenticity and DOS attacks

- Scenario:
  - P-Grid is operational
  - Some peers have dynamic IP addresses
- Problems:
  - How to find out that old address has become invalid?
    - No response ⇨ Network problem? Peer got new address?
    - Response ⇨ Is it still you, John? (authenticity, replay, man-in-the-middle)
  - DOS attacks are very simple:
    - Assume peers report back their new IP address
    - EvilHacker.org participates in P-Grid and thus finds out IP addresses
    - EvilHacker.org reports all IP addresses it finds pointing to random hosts or itself

Problems to tackle for P-Grid

- IP addresses (hostnames) are everywhere
  - Routing tables
  - Index
- Peer authenticity
- Rate of IP address changes may be crucial (thrashing)
- NAT must be addressed for applicability of P-Grid for end-users

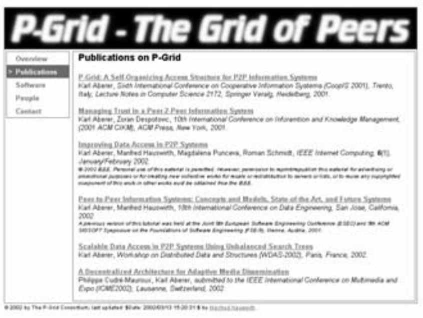
Dynamic IP addresses in P-Grid (proposal)

- Each peer is uniquely identified by a Universal Unique Identifier (UUID)
- UUIDs are mapped onto IP addresses via a directory services, i.e., P-Grid itself
  - Routing tables/index: UUIDs instead of IP addresses
  - Hen/egg problem: works only if probibility of follow-up queries is < 1
- Upon coming online again each P-Grid peer inserts its new IP address into the P-Grid mapping UUIDs onto IP addresses
- Authenticity of mappings
  - Public key schemes: too heavy and too much administrative effort
  - Use zero-knowledge-based scheme instead

Conclusions

- P2P is a valid new paradigm for Internet applications beyond mere file sharing
- P-Grid P2P approach proven to work (theory, implementation, simulation)
- Further research necessary to make P-Grid a platform for general-purpose applications

P-Grid Website (www.p-grid.org)



Publications on P-Grid

P-Grid: A Self Organizing Access Structure for P2P Information Systems
Karl Aberer, *Sixth International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy, Lecture Notes in Computer Science 2172, Springer Verlag, Heidelberg, 2001*

Managing Trust in a Peer 2 Peer Information System
Karl Aberer, Zoran Despotovic, *10th International Conference on Information and Knowledge Management, (2001 ACM CIKM), ACM Press, New York, 2001*

Improving Data Access in P2P Systems
Karl Aberer, Manfred Hauswirth, Magdalena Punceva, Roman Schmidt, *IEEE Internet Computing*, **6**(1), *January/February 2002*
[illegible]

Peer to Peer Information Systems: Concepts and Models, State of the Art, and Future Systems
Karl Aberer, Manfred Hauswirth, *18th International Conference on Data Engineering, San Jose, California, 2002*
[illegible]

Scalable Data Access in P2P Systems Using Unbalanced Search Trees
Karl Aberer, *Workshop on Distributed Data and Structures (WDAS-2002), Paris, France, 2002*

A Decentralized Architecture for Adaptive Media Dissemination
Philippe Cudré-Mauroux, Karl Aberer, *submitted to the IEEE International Conference on Multimedia and Expo (ICME2002), Lausanne, Switzerland, 2002*

© 2002 by The P-Grid Consortium, last updated: $Date: 2002/03/13 15:20:31 $ by [illegible]

The End

Questions?