Self-organizing Distributed Data Access

Karl Aberer
School of Computer and Communication Sciences, EPFL
karl.aberer@epfl.ch
lsirwww.epfl.ch

TU Berlin, October 31, 2003

joint work with:
Anwitaman Datta, Manfred Hauswirth, Roman Schmidt, Philippe Cudre-Mauroux, Zoran Despotovic
Outline

1. **Self-organizing Information systems**
2. Introduction of P-Grid
3. Identity management
4. Semantic interoperability
5. Conclusion
1. What is Self-organization?

- Informal characterization (physics, biology, cybernetics,… and CS)
  - distribution of control
    (= decentralization = symmetry in roles = P2P)
  - local interactions, information and decisions
    (= autonomy)
  - emergence of global structures
  - failure resilience and scalability

- Formal characterization (Francis Heylighen)

"The basic mechanism underlying self-organization is the noise-driven variation which explores different regions in a system's state space until it enters an attractor"
Self-organizing Data Access

• The most studied system: Gnutella and descendants
  - A group membership protocol (ping/pong) to construct an overlay network
  - A search protocol (query/queryhit) to locate information in the overlay network

• What is Gnutella used for?
  - content sharing (music, video, etc.)
  - studying self-organization in computer science

• Why is Gnutella self-organizing?
Resulting Network Structure

- Specific network structure emerges from a self-organizing process
  - state space = graphs with constant outdegree
  - noise = time of node joins, latency, decisions on connectivity
  - attractor = approx. powerlaw graphs

- Powerlaw graphs
  - Preferential attachment + growing network [Barabasi, 1999]
  - However uniform distribution at tail (minimal connectivity, adds to robustness)
  - Short diameter [Ripenau et al, 2002]
Searching the Network

- **Gnutella = unstructured network**
  - no information about what data neighboring nodes store
  - Broadcast to d neighbors with maximal time-to-live TTL
  - TTL > network diameter
  - duplicate message detection: < dN messages

- **Reducing message bandwidth**
  - Random walkers [Lv et al, 2002]
  - Percolation search [Sarshar et al 2002]
  - Assumptions on network structure (random, powerlaw, small world graph etc.)

Conclusion: Network construction and search protocols are orthogonal issues
1. Network construction is a self-organizing process
2. Search is "simply" a randomized, distributed algorithm

Problem: Search in "unstructured networks" requires high bandwidth
Outline of the Presentation

1. Self-organizing Information systems
2. Introduction of P-Grid \[\text{Coopis01, IEEE ICO2, ...}\]
3. Identity management
4. Semantic interoperability
5. Conclusion
Efficient Distributed Data Access

- **FULL REPLICATION**
  - High update cost
  - Search cost low
  - Maximal bandwidth low

- **SCALABLE DISTRIBUTED DATA ACCESS STRUCTURES** (e.g., prefix routing)
  - Low update cost
  - Search cost low
  - Maximal bandwidth low

- **BROADCAST** (e.g., Gnutella)
  - High update cost
  - Search cost high
  - Maximal bandwidth low

- **SERVER** (e.g., Napster)
  - High update cost
  - Search cost low
  - Maximal bandwidth high
Data Access Structures

- Search tree (Prefix Tree)
Scalable Data Access Structures (SDAS)

- Distribute search tree over peers
Scalable Data Access Structures

"Napster" bottleneck

peer 1 peer 2 peer 3 peer 4
Scalable Data Access Structures

• Associate each peer with a complete path
Scalable Data Access Structures

- Associate each peer with a complete path

```
peer 1  peer 2
know more about this part of the tree

peer 3

100  101
knows more about this part of the tree

peer 4
```
Result: P-Grid [Coopis 2001]

- Peers cooperate in search
ID peer identifier

2,3 data keys (2=0010 etc.)

1 : 12, 13 routing table entry

query(101) @ 7

Prefix Routing

ID peer identifier

2,3 data keys (2=0010 etc.)

1 : 12, 13 routing table entry

query(101) @ 7

Prefix Routing
## P-Grid Efficiency

### Table 1: Performance comparison of Gridella and Gnutella.

<table>
<thead>
<tr>
<th>Peers</th>
<th>Gridella messages</th>
<th>Gnutella messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>20,000</td>
<td>61</td>
<td>8,744</td>
</tr>
<tr>
<td>40,000</td>
<td>63</td>
<td>26,240</td>
</tr>
<tr>
<td>60,000</td>
<td>65</td>
<td>26,240</td>
</tr>
<tr>
<td>80,000</td>
<td>65</td>
<td>78,728</td>
</tr>
<tr>
<td>100,000</td>
<td>68</td>
<td>78,728</td>
</tr>
<tr>
<td>120,000</td>
<td>69</td>
<td>78,728</td>
</tr>
<tr>
<td>140,000</td>
<td>68</td>
<td>78,728</td>
</tr>
<tr>
<td>160,000</td>
<td>69</td>
<td>78,728</td>
</tr>
<tr>
<td>180,000</td>
<td>69</td>
<td>78,728</td>
</tr>
<tr>
<td>200,000</td>
<td>72</td>
<td>78,728</td>
</tr>
</tbody>
</table>
Construction of a Prefix Routing Structure

• Standard method (Plaxton, Pastry, Chord, Tapestry, …)
  - static assignment of peers to data keys:
    fully defines the distributed data structure
  - distributed, deterministic algorithms insert nodes correctly into the network
  - Problems: load balancing, robustness
  - Alternative: coordinator (bottleneck)

• Self-organizing method (P-Grid [CoopIs 2001, IEEE IC 2002])
  - replace the coordinator by a random process
  - peers agree via local interactions on their data key
  - requires some care in order to work efficiently

when two peers meet
peer extends path if maximal path length not reached
do the necessary bookkeeping
P-Grid Construction Example

peer 1  peer 2  peer 3  peer 4
P-Grid Construction Example
P-Grid Construction Example

peer 1  peer 2  peer 3  peer 4
P-Grid Construction Example

peer 1
peer 2
peer 3
peer 4
P-Grid Construction Example

peer 1
peer 2
peer 3
peer 4
P-Grid Construction Example

peer 1  peer 2  peer 3  peer 4

©2003, Karl Aberer, School of Computer and Communication Sciences

TU Berlin
31. October, 2003
P-Grid Construction Example
P-Grid Construction Example

peer 1  peer 2  peer 3  peer 4
**P-Grid Construction Example**

```
peer 1  peer 2  peer 3  peer 4
```

```
0??  0??  1??
```

```
10??  11??
```
P-Grid Construction Example

peer 1          peer 2

peer 3          peer 4
P-Grid Construction Example

peer 1

peer 2

peer 3

peer 4
Efficiency of P-Grid Construction

- Constructing a tree of depth 6 (64 leaves)
Non-uniform Data Distribution

• Construct a tree of depth 2 for the following data:
  10, 01, 001, 0001, 00001, 000001, 000000
P-Grid Construction Balancing Storage Load

- It makes no sense to create leaves in the tree for data occurring rarely
- Every peer stores initially some data

when two peers meet
peer extends path only if #data items > ε
    do the necessary bookkeeping
otherwise data exchange (replica generation)

- Requires some care in order to work efficiently AND to correctly balance the storage load
Result [WDAS 2002]

- Each node has same storage load
- Algorithm still converges quickly

All binary strings of length 5 sorted by frequency
Balancing Data Distribution

- The following problem has been studied to understand load balancing properties during construction:
  - Initially each peer holds 1 item marked 0 and 2 items marked 1.
  - Peers interact bilaterally, decide to hold in the future only data items marked 0 or 1 and exchange their data accordingly.
  - Goal: 1/3 of the peers should decide for 0, 2/3 for 1.
  - Possible interactions:
    - $(12, 12) \rightarrow (20, 04)$ with probability $p/2$ (peer 1 specializes to 0, peer 2 to 1).
    - $(12, 12) \rightarrow (12, 12)$ with probability $1-p$ (nothing happens).
    - $(04, 02) \rightarrow (03, 03)$ (load balancing).
    - $(12, 20) \rightarrow (02, 30)$ (redistribution).

- Stochastic process (Markov process) approximated by diff. equations:
  - For $p=1$ (always specialize immediately) 8% of peers stay in state 20, thus no perfect balancing possible.
  - If $p \rightarrow 0$ the fraction of peers staying in state 20 tends towards 0.
  - Conclusion: during P-Grid construction avoid eager specialization in order to avoid imbalance.
Unbalanced Search Trees

- Construct the tree for the following data:
  10, 01, 001, 0001, 00001, 000001, 000000

  worst case: $O(N)$ steps!

query(000000)

different choice of references

$\leftrightarrow$
different P-Grids for the same tree shape

**Theorem**: If we select randomly a P-Grid corresponding to a given tree shape, then the expected number of messages in a search is $\log(N)$, independent of the tree shape. 

[WDAS 2002]

1. P-Grid construction is a self-organizing process
2. Prefix search is an (efficient) randomized, distributed algorithm
Outline of the Presentation

1. Self-organizing Information systems
2. Introduction of P-Grid
3. Identity management [DEXA MDM03, ...]
4. Semantic interoperability
5. Conclusion
Self-healing Directory

- Any distributed access structure (such as P-Grid) requires dynamic mapping of a logical ID (associated data key) to physical ID (IP address)
  - This mapping cannot be static in the presence of dynamic IP addresses
  - A very important problem for the implementation of any P2P system

P-Grid

lookup IP address

routing based on logical address

lookup IP address in case of failure

directory implemented by P-Grid

directory (logical ID <-> IP address)
Approach and Result

• Basic Approach
  1. Store the mapping in a P-Grid access structure
  2. Incoming peers register and the mapping is updated
  3. If they reuse a logical ID they proof their identity using a secret key
  4. P-Grid nodes cache mapping in their routing tables and use it for routing
  5. In case of invalid cache entries recursively searches on the directory are initiated to repair the invalid cache

• Repair strategies
  - Eager strategy: repair each stale entry encountered immediately
  - Lazy strategy: repair a routing table when all references at one level become stale
query(01*) @ 7
  ...query(0101) @ 7 (for stale entry 5, cycle -> abort)
  ...query(1110) @ 7 (for stale entry 14, forward to 12 or 13)
  ...query(1110) @ 12 (is offline)
  ...query(1110) @ 13 (for stale entry 2)
  ......query(0010) @ 13 (forward to 5)
  ......query(0010) @ 5 (forward to 7)
  ......query(0010) @ 7 (forward to 9)
  ......query(0010) @ 9 (new entry for 2 found !)
  ...query(1110) @ 2 (new entry for 14 found !)
query(01*) @ 14 (finally 😊)
Analysis of Lazy Repair Strategy

- System is in dynamic equilibrium if rate of changes due to changing IDs and rate of repairs is equal
  - $r_{up}$ rate of changes
  - $r$ number of references maintained at each level
  - $p_{stale}$ probability that references at a level are all stale
  - $p_{dyn}$ probability that any reference is stale
  - $N_{rec}$ number of additional recursive queries initiated

- $r_{up} (1-p_{dyn}) \log_2(n) r = (1-r_{up}) N_{rec}$

- $A_h = r + r p_{stale} A_{rec}, A_{rec} = \log_2(n) A_h, N_{rec} = (A_{rec}-1) / (r \log_2(n)/2)$

- relationship between $p_{dyn}$ and $p_{stale}$ obtained from modelling a probabilistic process
Comparison Analytical vs. Simulation

- and comparing lazy vs. eager (case peers always online, but change ID)

messages $n=128,256,512,1024$

(a) Message vs. $r_{up}$, eager algorithm

(b) Message vs. $r_{up}$, lazy algorithm
Dependency on Offline Behavior
Outline of the Presentation

1. Self-organizing Information systems
2. Introduction of P-Grid
3. Identity management
4. Semantic interoperability [WWW03,JoSemWeb03]
5. Conclusion
Why are Schemas Important?

- Example: Searching biological databases
  - Without schema (like Google, Gnutella)

- Searching for data on "anglerfish"
  - Results will be precise

- This seems easy, but the same for "leech"
  - Organism leech
  - Authors: "Bleech", "Leechman", ...
  - Protein sequences: ...MNTSLEECHMPKGD...

- Search for "257" ...
Schema Heterogeneity

• Different databases - Different schemas
  - SwissProt: Find <Species> leech </Species>
  - EMBLChange: Find <Organism> leech </Organism>

• Standardization (global schema)?
  - Music files: clear scope, simple semantics ☺
  - Scientific databases: different scope, distributed knowledge, little agreement, etc. 😐

• Hardest problem in information systems: semantic interoperability
Translating Heterogeneous Schemas

• A non-expert may be able to relate
  – <Organism> ⇔ <Species>
  – <Author> ⇔ <Authors> etc.
• But what about
  – <AaMutType> ⇔ <DnaMutType>
  – <FtKey> ⇔ <FtKey>
in Swisschange and EMBLChange?

• The answers can only be given by the experts ...
  ... sometimes only by the data owners!

**Approach**: ask them to provide their translations from some "known" schema to their "own" schema (local step)
Local Semantic Interoperability (Translation)

Q1 =
$id$s p/ID$</ID>
FOR $sp$ IN /SP_entry
WHERE "anglerfish" IN $sp/organism

Q2 =
$id$s p/ID$</ID>
FOR $sp$ IN T12
WHERE "anglerfish" IN $sp/organism

SwissProt
(known schema)

EMBLChange
(own schema)

Computer-processable languages: XML, XQuery

©2003, Karl Aberer, School of Computer and Communication Sciences

TU Berlin
31. October, 2003
Global Semantic Interoperability

SwissProt peers
authors, titles, organism, ...

other peers
authors, ...

EMBLChange peers
species, ...

A lab at MIT
organism
organism
organism
Swissprot site at Geneva

A lab in Trondheim
species
species
species

Query posted at EPFL
organism → authors
organism → species

Semantic Gossiping
How to Detect a Semantic Agreement?

SwissProt peers
authors, titles, organism, ...

other peers
authors, ...

EMBLChange peers
species, ...

Check what is preserved in cycles!
Establishing Semantic Agreement

- Network of peers - share a finite set of similar concepts
  - e.g. biological databases
- But - they refer to concepts using different names
- Global interoperability created by applying semantic gossiping techniques
  - Using purely pair-wise, local transactions
- Semantic Analysis
  - Cycle Analysis: check whether mappings of names along cycles are consistent
  - Results Analysis: check whether content of query matches concepts (assuming peers can perform content analysis)
  - determine likelihood that for a given set of observations a given mapping is correct (probabilistic analysis)
- Adaptation
  - peers modify mappings which are likely to be false
  - Eventually (on a long run) method repairs all erroneous mappings since peers can base their decisions on numerous and meaningful feedback cycles
Experimental Evaluation

• Creation topology of $n$ peers
  - Peers share $|C|$ concepts, but use distinct names to refer to them
  - Each peer connected through translation to other peers (e.g. using small world graphs)

• Generation of mappings for every translation link
  - Correct mappings
  - Erroneous mappings ($eRate$)

• Semantic gossiping techniques applied iteratively to detect and rectify erroneous translations
  - At every step peer randomly selects one name and issues a query about it
  - Query propagated in Gnutella-like fashion with TTL value

• Peers evaluate the correctness of current mapping (maximum-likelihood techniques)
  - Peer adopts most probably correct mapping if probability of being correct is above 50%
Cycle analysis – Sensitivity to the number of outgoing edges

- \( N=25; \) eRate = 0.1; \( \| \mathbf{C} \| = 4; \) TTL=5.
Cycle analysis - Sensitivity to the TTL

- $N=25$; $eRate = 0.1$; $\|C\| = 4$; $l=4$. 
Cycle analysis - Sensitivity to the initial error rate

- \( N=25; \| C \| =4; \) TTL=5; \( l=4 \)
Cycle analysis - Scalability

- $eRate = 0.1; ||C|| = 4; TTL = 5; l = 4;$
Results Analysis – Sensitivity to initial error rate

- N = 50 peers
- 100 shared documents
- l = 2 outgoing links
- \|C\| = 4 concepts
- TTL=3
- Initial error rate 10%
- Probability of misclassifying documents 10%
Results Analysis – Sensitivity to misclassification rate

% wrong mappings

$\# \text{ steps}$

- $r_{\text{res}} = 0.4$
- $r_{\text{res}} = 0.3$
- $r_{\text{res}} = 0.2$
- $r_{\text{res}} = 0.1$
- $r_{\text{res}} = 0.05$
Results Analysis – Scalability
Results Analysis – Sensitivity to number of documents
Combined results

- Every peer first performs a result analysis step
  - Modifying a few mappings depending on results returned
- Cycle analysis step
  - Reaching local agreement on mappings based on cycle feedback
- Simulation
  - 25 peers
  - 4 concepts
  - 2 outgoing links
  - TTL = 3 (results) TTL=6 (cycles)
Combined results
Outline of the Presentation

1. Self-organizing Information systems
2. Introduction of P-Grid
3. Identity management
4. Semantic interoperability
5. Conclusion
Conclusion

Self-organizing Distributed Data Access
1. Construction of distributed data access structure as self-organizing process
2. Scalable, randomized, distributed algorithm for search, update exploiting 1.

Self-organizing Applications (Multi-Level Self-organization)
- Bootstrapping identity
- Reputation and trust, relevance and ranking

- Data management issues (in particular with semantically meaningful data)
  - skewed data distributions, updates, complex search predicates, semantics

- P-Grid implementation under way
  - Java, nice GUI, Gnutella/JXTA compatible
Publications on P-Grid

**P-Grid: A Self-Organizing Access Structure for P2P Information Systems**

**Managing Trust in a Peer-to-Peer Information System**

**Improving Data Access in P2P Systems**
Karl Aberer, Manfred Hauswirth, Magdalena Puncor, Roman Schmidt, *IEEE Internet Computing*, 6(1), January/February 2002
© 2002 IEEE. Personal use of this material is permitted. However, permission is required for re-publication, republication, or redistribution to servers or lists, or to reuse any copyrighted material from this work in other works.

**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.
A previous version of this tutorial was held at the Joint 9th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-9), Vienna, Austria, 2001.

**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