Scalable Data Access in P2P Systems

Karl Aberer

EPFL

Computer & Communication Systems

Institute for Core Computing Science

Distributed Information Systems Laboratory

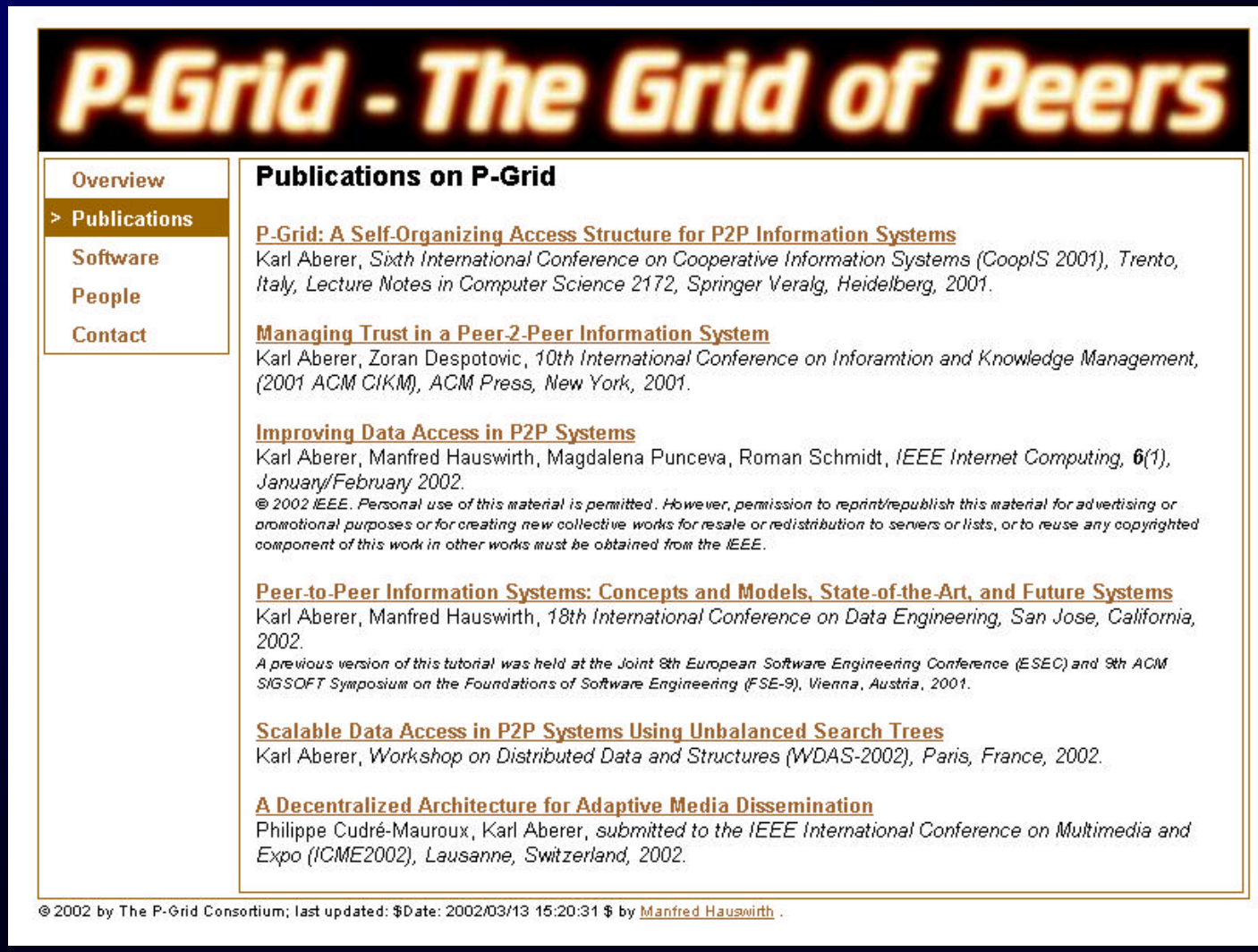
karl.aberer@epfl.ch

lsirwww.epfl.ch

July 31, 2002

CERN

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



Overview

1. P2P Systems : Decentralizing Information Systems

2. P-Grid : Efficient Decentralized Search

3. Semantic Gossiping : Emergent Global Semantics

Overview

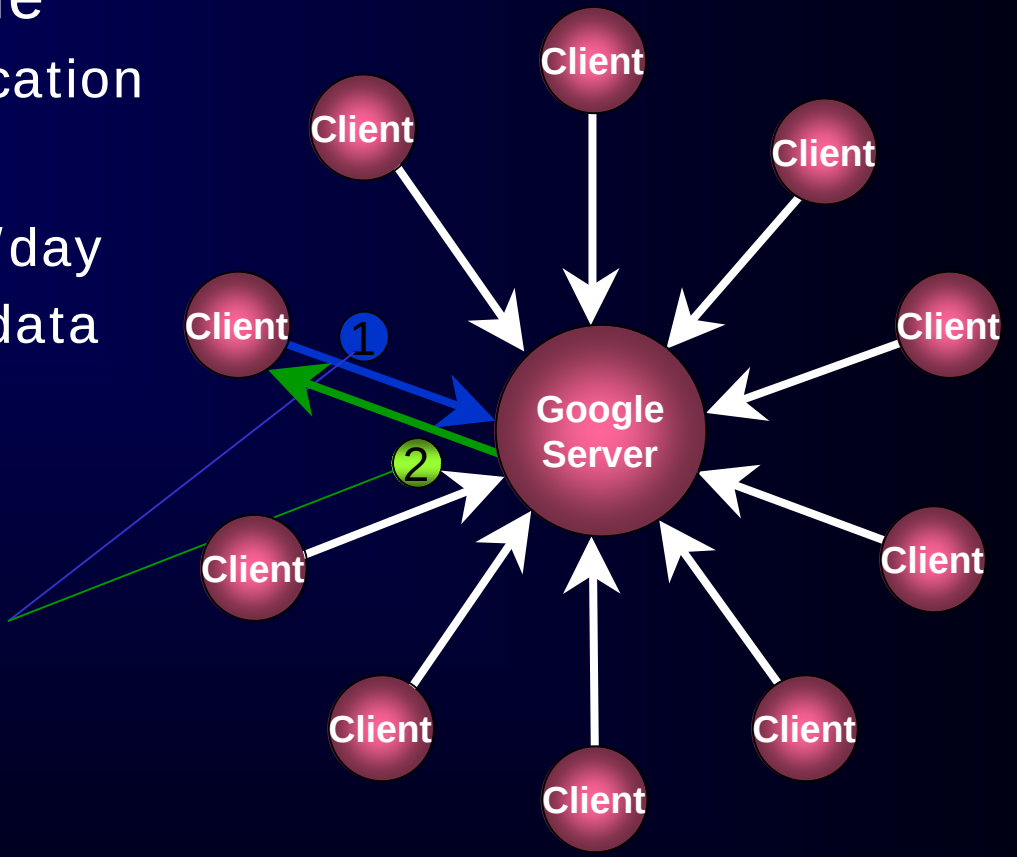
1. P2P Systems : Decentralizing Information Systems

2. P-Grid : Efficient Decentralized Search

3. Semantic Gossiping : Emergent Global Semantics

Centralized Information Systems

- Web search engine
  - Global scale application
- Example: Google
  - 150 Mio searches/day
  - 1-2 Terabytes of data (April 2001)

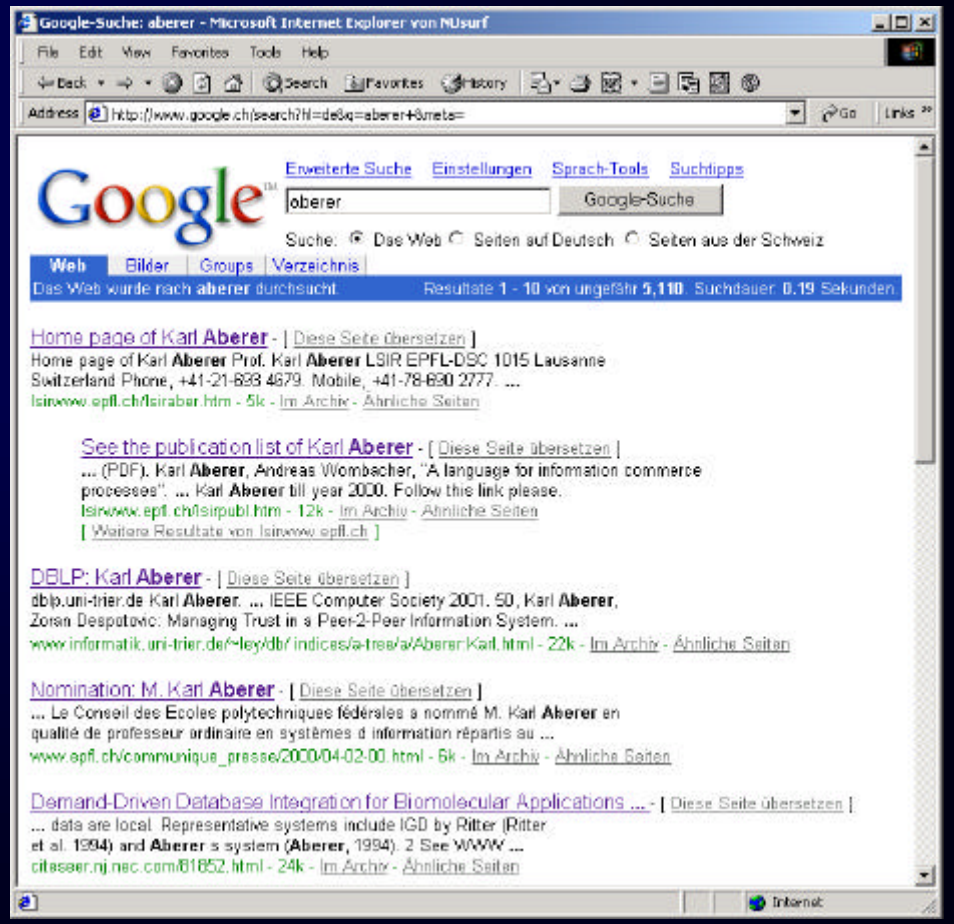


Result
home page of Karl Aberer …

Google: 15000 servers

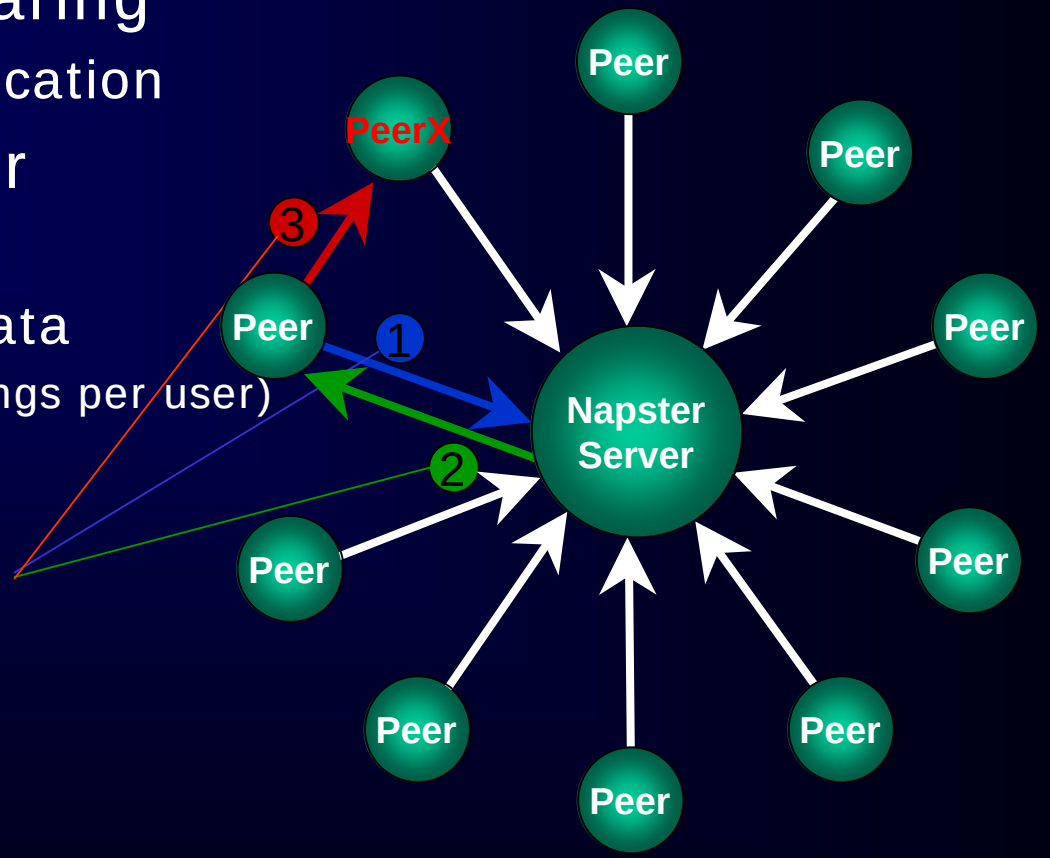
Google Assessment

- Strengths
  - Global ranking
  - Fast response time
- Weaknesses
  - Infrastructure, administration, cost
  - A new company for every global application ?



(Semi-)Decentralized Information Systems

- P2P Music file sharing
  - Global scale application
- Example: Napster
  - 1.57 Mio. Users
  - 10 TeraByte of data
    (2 Mio songs, 220 songs per user)
    (February 2001)



Request and transfer file f.mp3 from peer X directly

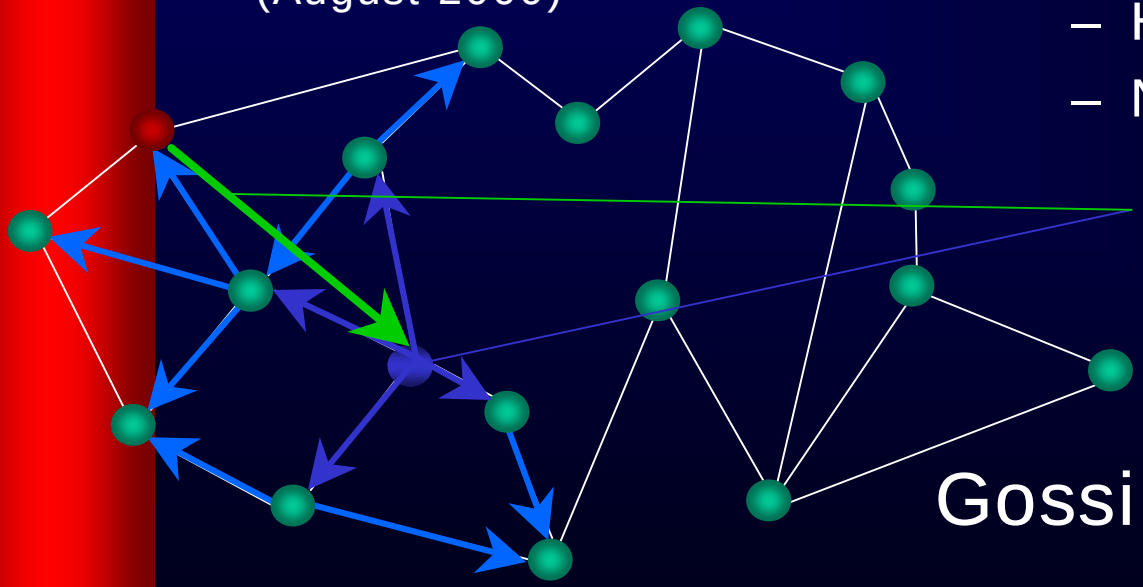
Napster: 100 servers

Lessons Learned from Napster

- Strengths
  - global information system without huge investment
    - exploit unused resources at nodes (space)
    - exploit users knowledge at nodes (annotation)
  - decentralization of cost and administration
- Weaknesses
  - business model: copyrighted material
  - server is single point of failure
  - therefore it can, for example, be shut down

Fully Decentralized Information Systems

- P2P file sharing
  - Global scale application
- Example: Gnutella
  - 40.000 nodes, 3 Mio files (August 2000)



- Strengths
  - Good response time
  - No infrastructure
- Weaknesses
  - High network traffic
  - No structured search

I have "brick_in_the_wall.mp3"
....

Gossiping

Gnutella: no servers

Napster vs. Gnutella

	Napster	Gnutella
Resources: search	**central**	**decentral**
Resources: file exchange	**decentral**	**decentral**
Knowledge: schema	**central**	**trivial**
Knowledge: annotation	**decentral**	**decentral**
	Partially decentralized	Self-Organizing

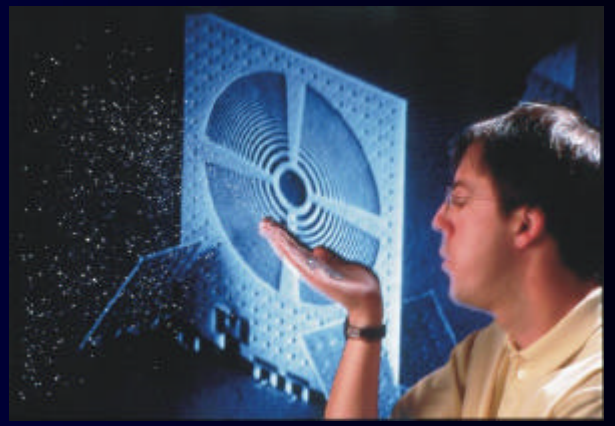
Decentralization – Self-Organization

- Decentralization avoids bottlenecks
  - Resources (cost, administration)
    ➔ avoid performance bottleneck
  - Design (knowledge and control)
    ➔ avoid single point of failure

- Full decentralization requires self-organization
  - Local information, local operation, local decision
  - Global behavior emerges from local behavior

Motivations

- Technical
  - Too many nodes
  - Mobile ad-hoc networks



- Business
  - content distribution (e.g. Bertelsmann-Napster)
  - knowledge management
  - scientific data and resource sharing (e.g. seti@home)

Original Internet designed as decentralized system
P2P ~ application-level Internet on top of the Internet

Self-Organization and Efficiency

- Self-organization
  - Can be costly if done wrong
- Example: Search Efficiency in Gnutella
  - Search requests are broadcasted
  - Anectode: the founder of Napster computed that a single Gnutella search request (18 Bytes) on a Napster community would generate 90 Mbytes of data transfers

Overview

1. P2P Systems : Decentralizing Information Systems

2. P-Grid : Efficient Decentralized Search

3. Semantic Gossiping : Emergent Global Semantics

Overview

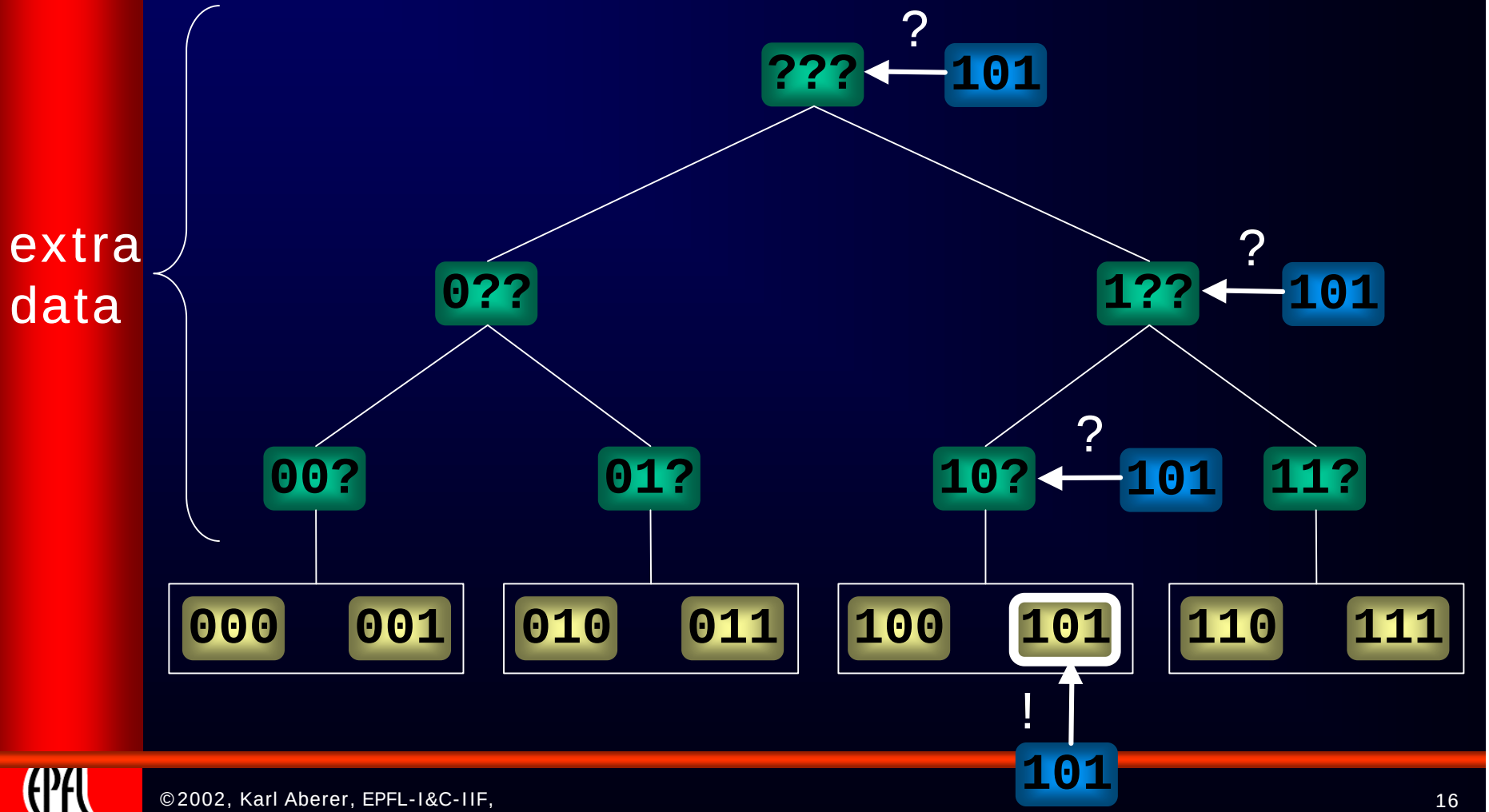
1. P2P Systems : Decentralizing Information Systems

2. P-Grid : Efficient Decentralized Search

3. Semantic Gossiping : Emergent Global Semantics

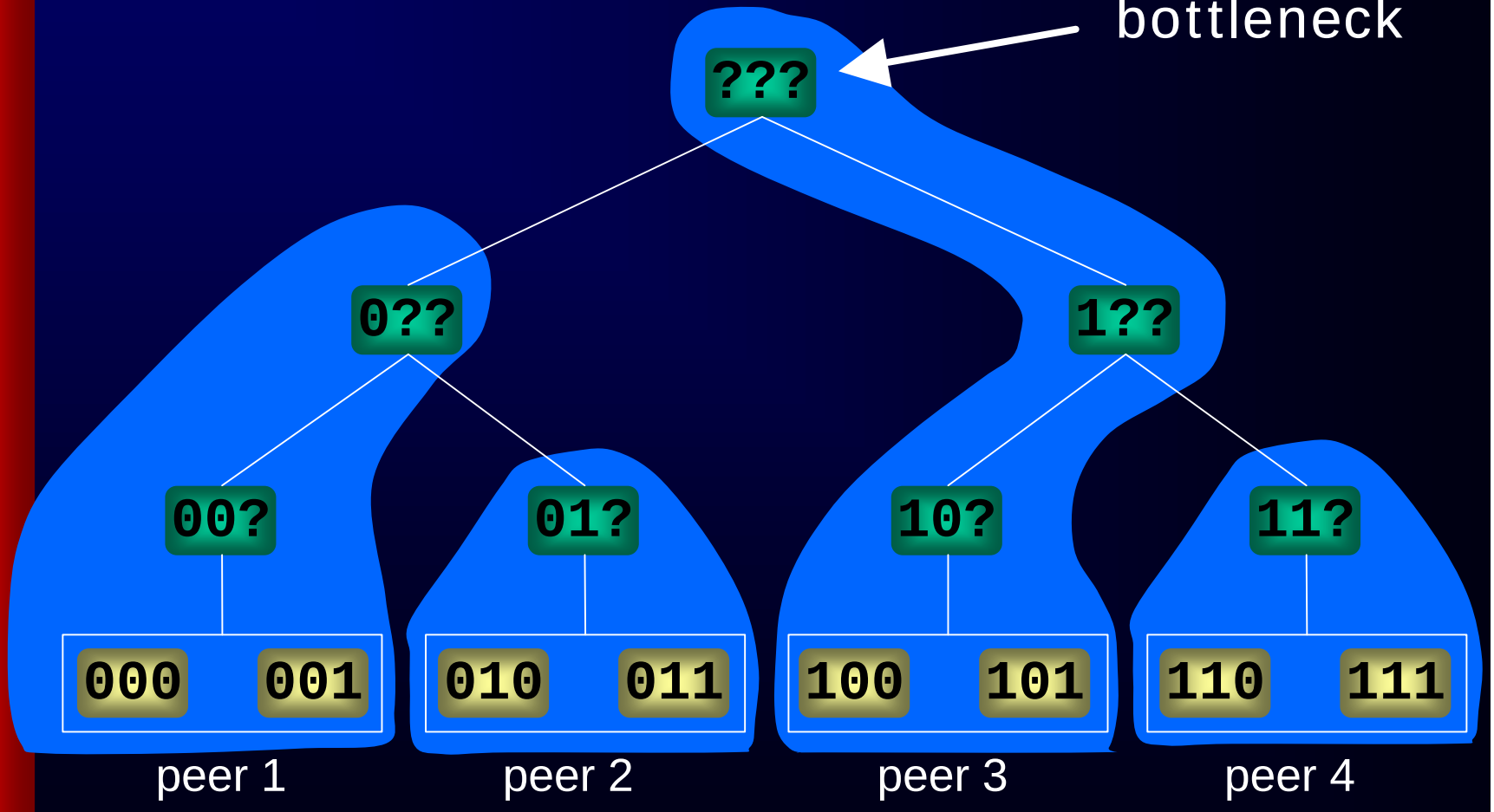
Data Access Structures

- Search tree (Prefix Tree)

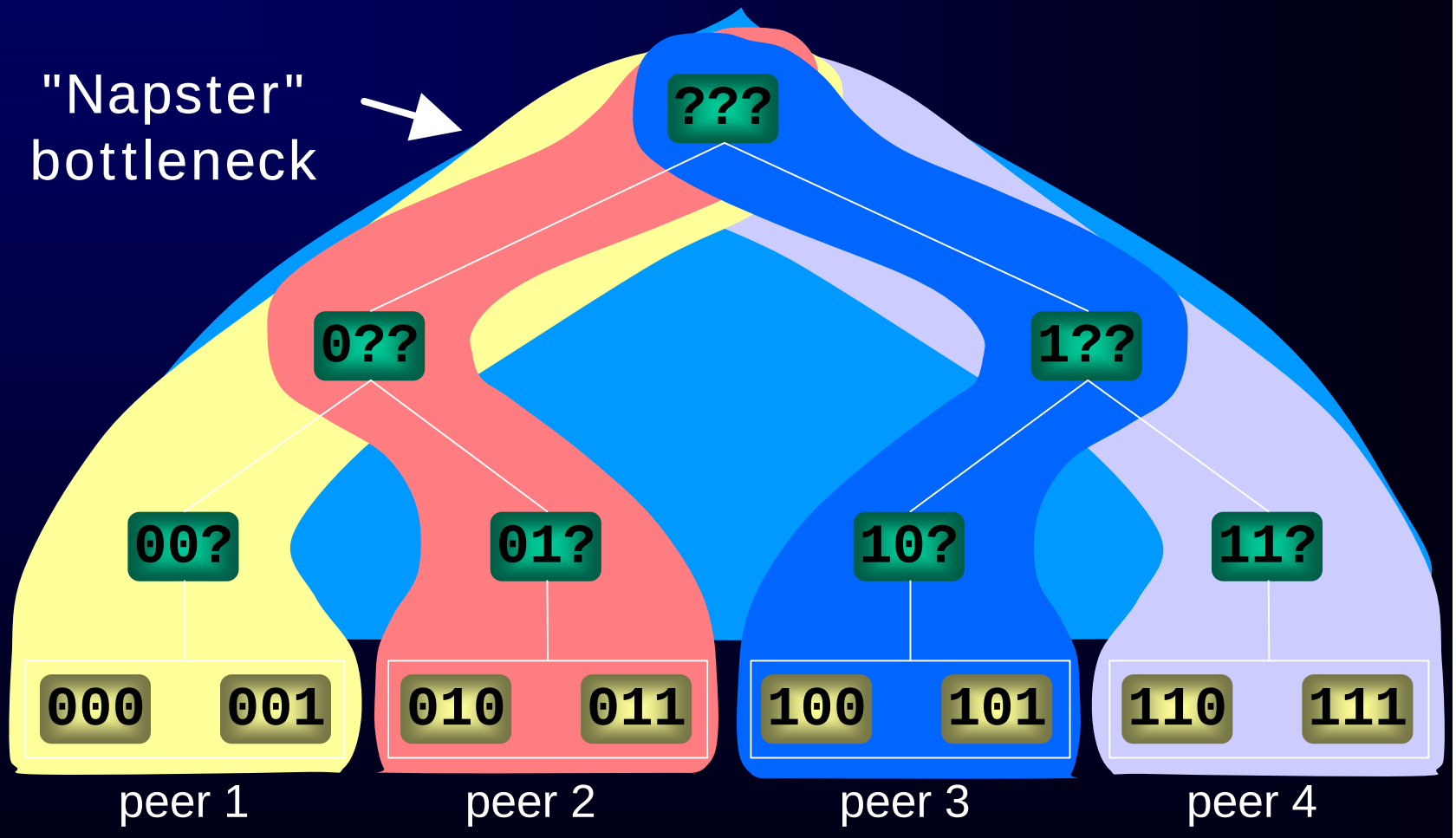


Scalable Data Access Structures (SDAS)

- Distribute search tree over peers

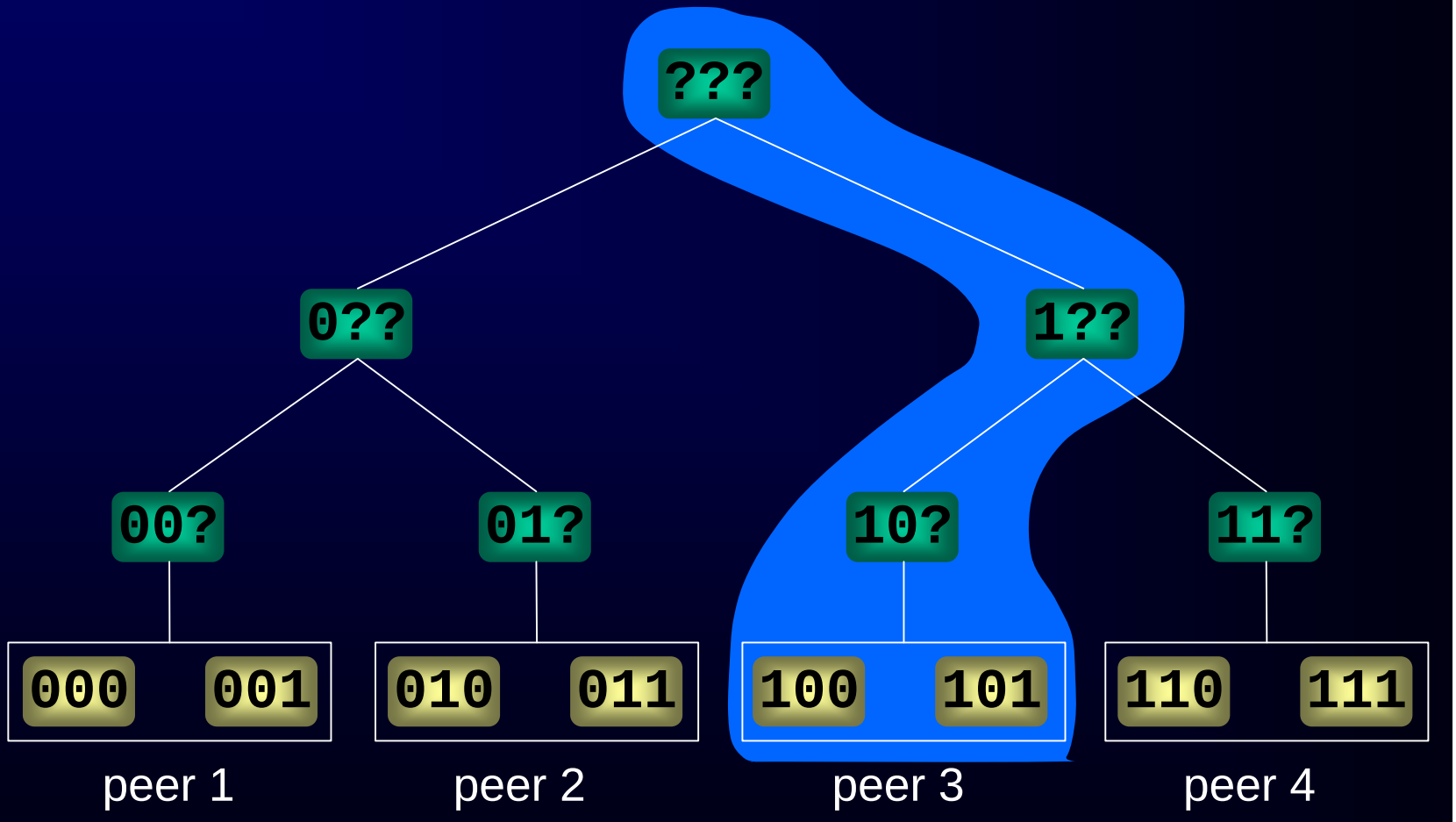


Scalable Data Access Structures



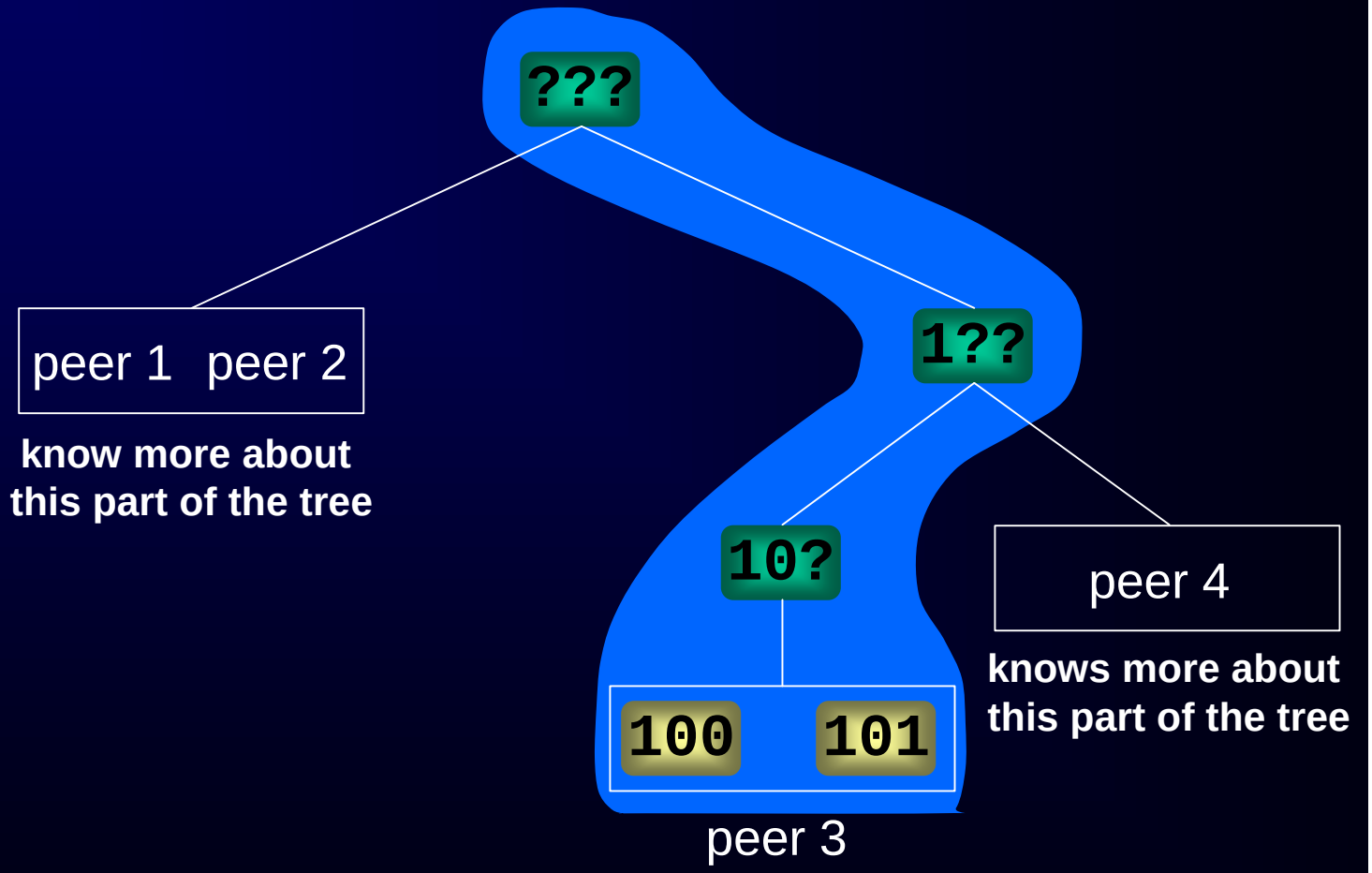
Scalable Data Access Structures

- Associate each peer with a complete path



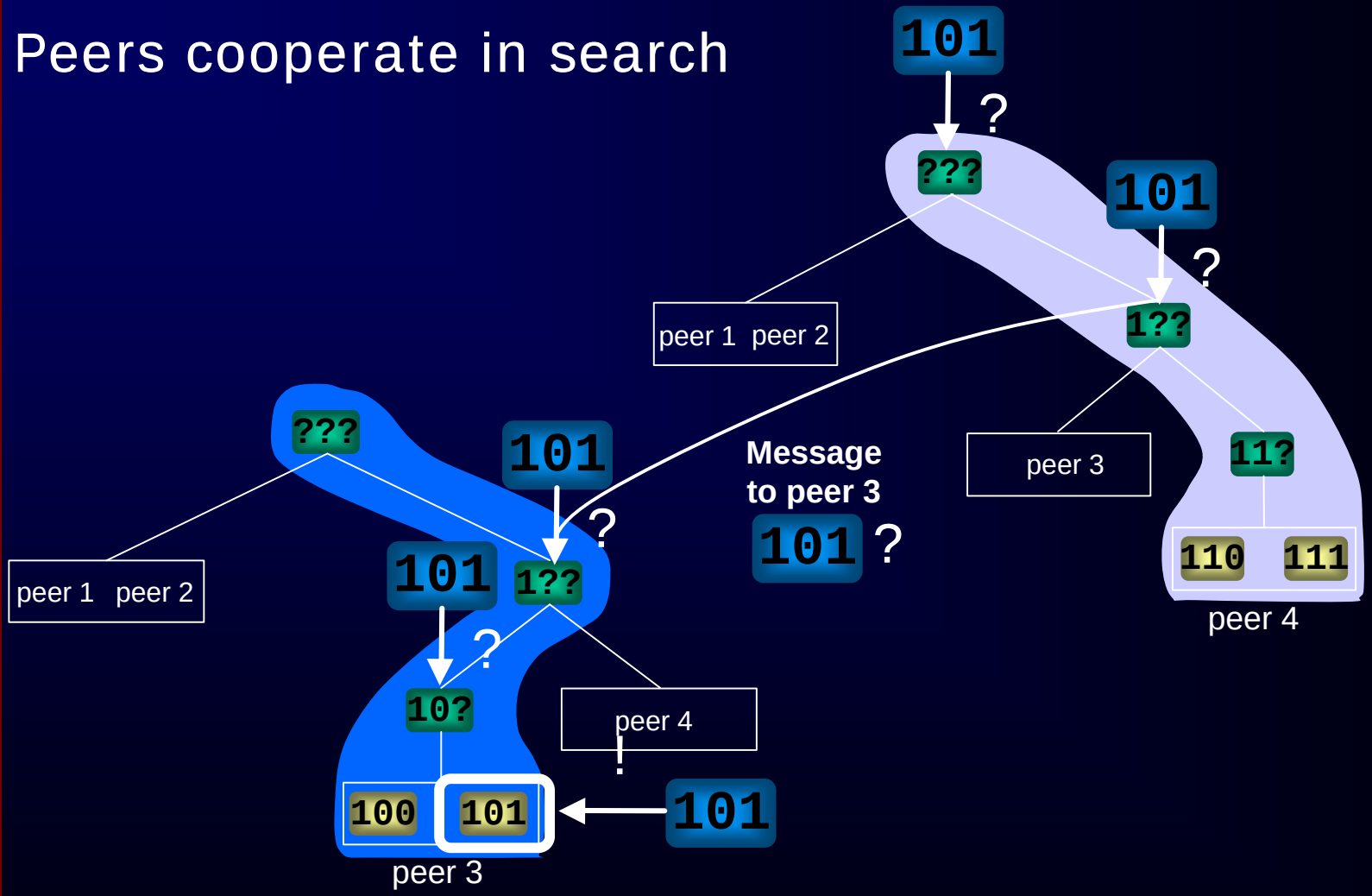
Scalable Data Access Structures

- Associate each peer with a complete path



Result: P-Grid [Coopis 2001]

- Peers cooperate in search



P-Grid Efficiency

Table 1. Performance comparison of Gridella and Gnutella.

Peers	Gridella messages	Gnutella messages
20,000	61	8,744
40,000	63	26,240
60,000	65	26,240
80,000	65	78,728
100,000	68	78,728
120,000	69	78,728
140,000	68	78,728
160,000	69	78,728
180,000	69	78,728
200,000	72	78,728

Construction of P-Grid ?

- SDAS similar to P-Grid
  - distributed and parallel DBMS, scalable storage systems (OceanStore, Pastry, Plaxton etc.), DNS
- Standard construction methods

1. Each node has an identifier - compute from the identifier the position in the tree
   - Structure of tree depends on identifier distribution
   - Nodes cannot decide which data they want to store and which requests they want to answer (autonomy)
2. Each node asks a coordinator on its position in the tree
   - Coordinator is a bottleneck

P-Grid Construction

- Idea
  - Replace the coordinator by a random process
  - P-Grid construction algorithm [Coopis 2001]
  - distributed, decentralized, randomized

```
when two peers meet
    if a maximal path length is not reached
        try to extend "their" path in tree
        do the necessary bookkeeping
```

- Requires some care in order to work efficiently

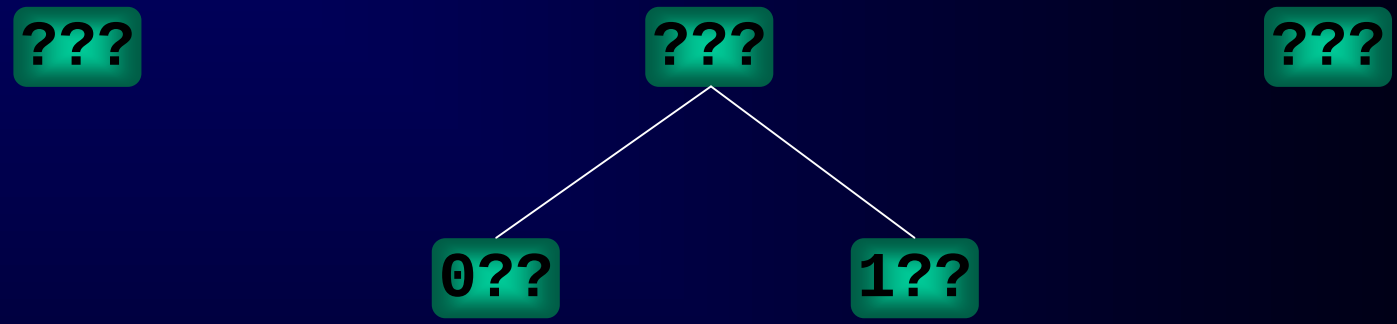
P-Grid Construction Example

peer 1 peer 2 peer 3 peer 4

P-Grid Construction Example

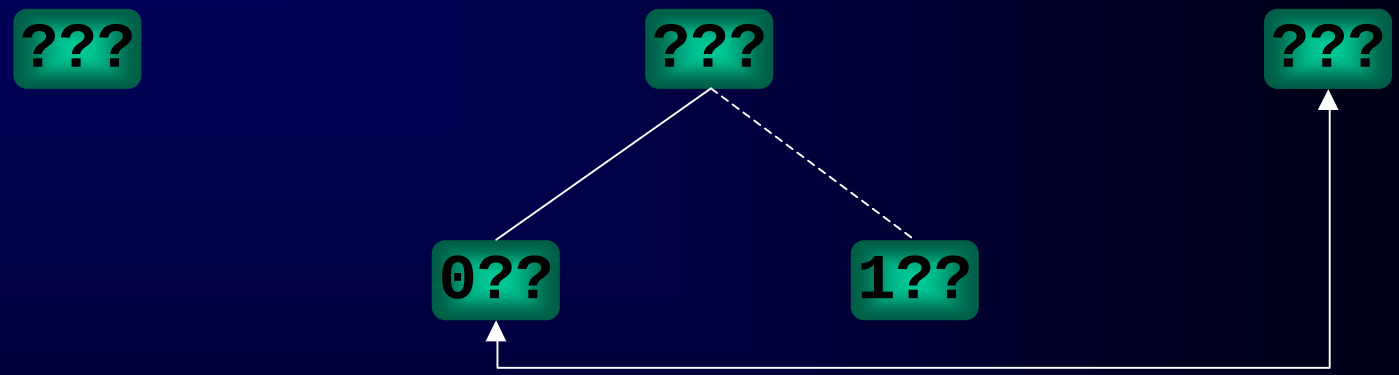
peer 1 peer 2 peer 3 peer 4

P-Grid Construction Example



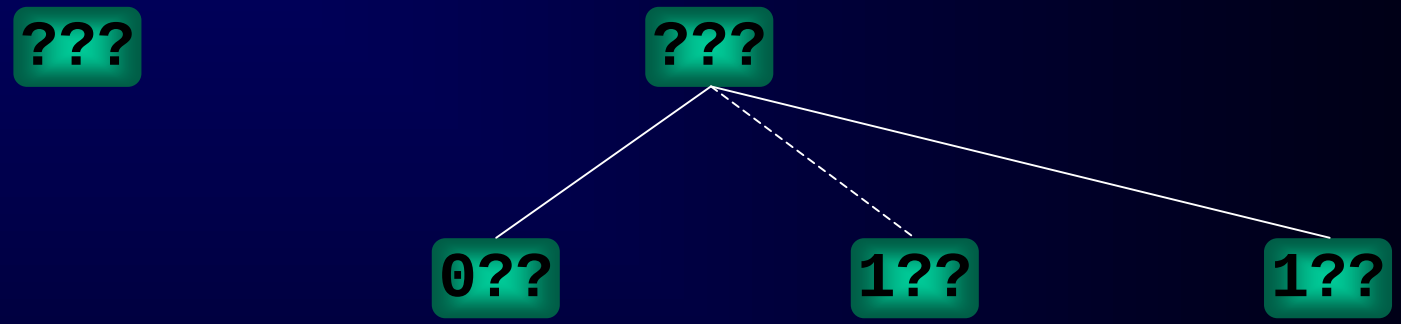
peer 1 peer 2 peer 3 peer 4

P-Grid Construction Example



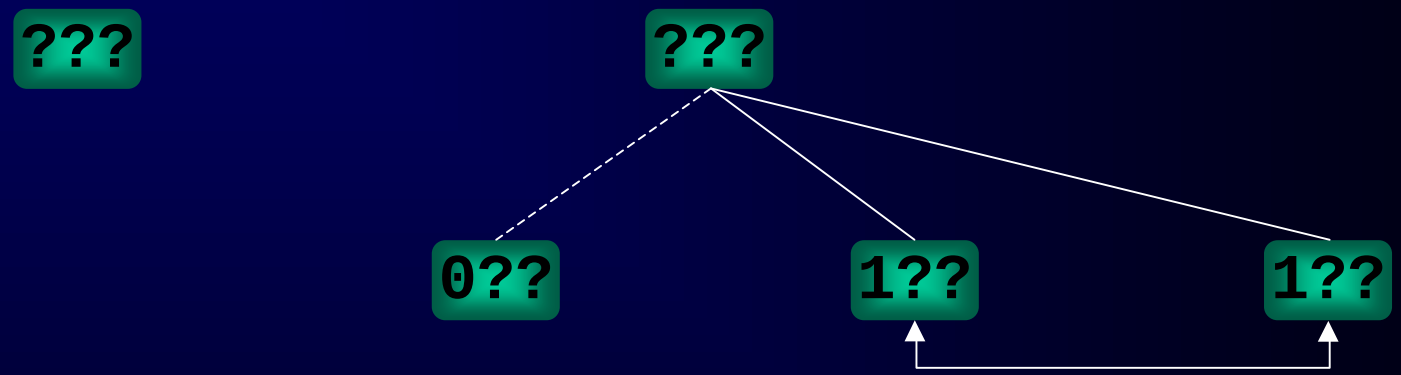
peer 1 peer 2 peer 3 peer 4

P-Grid Construction Example



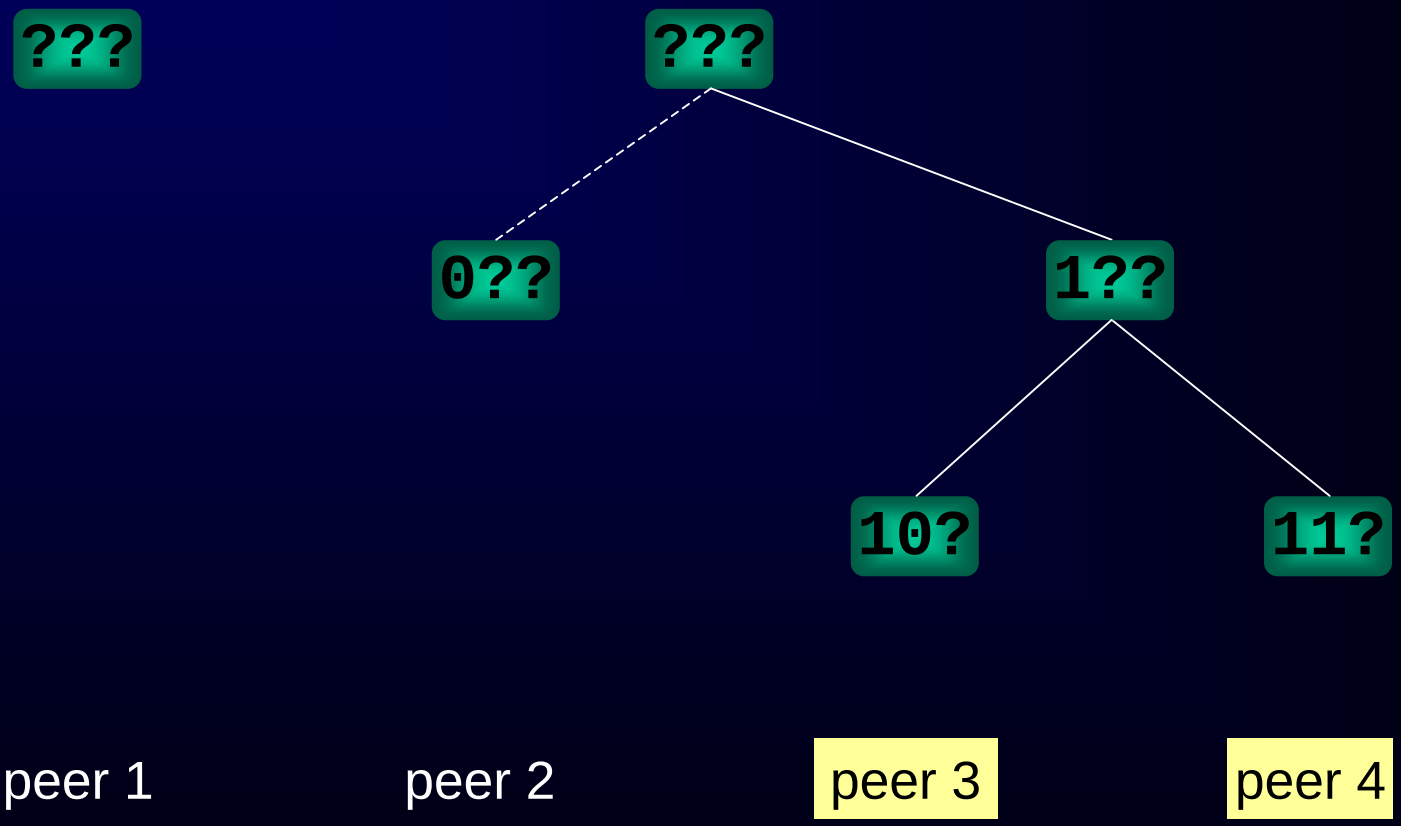
peer 1 peer 2 peer 3 peer 4

P-Grid Construction Example

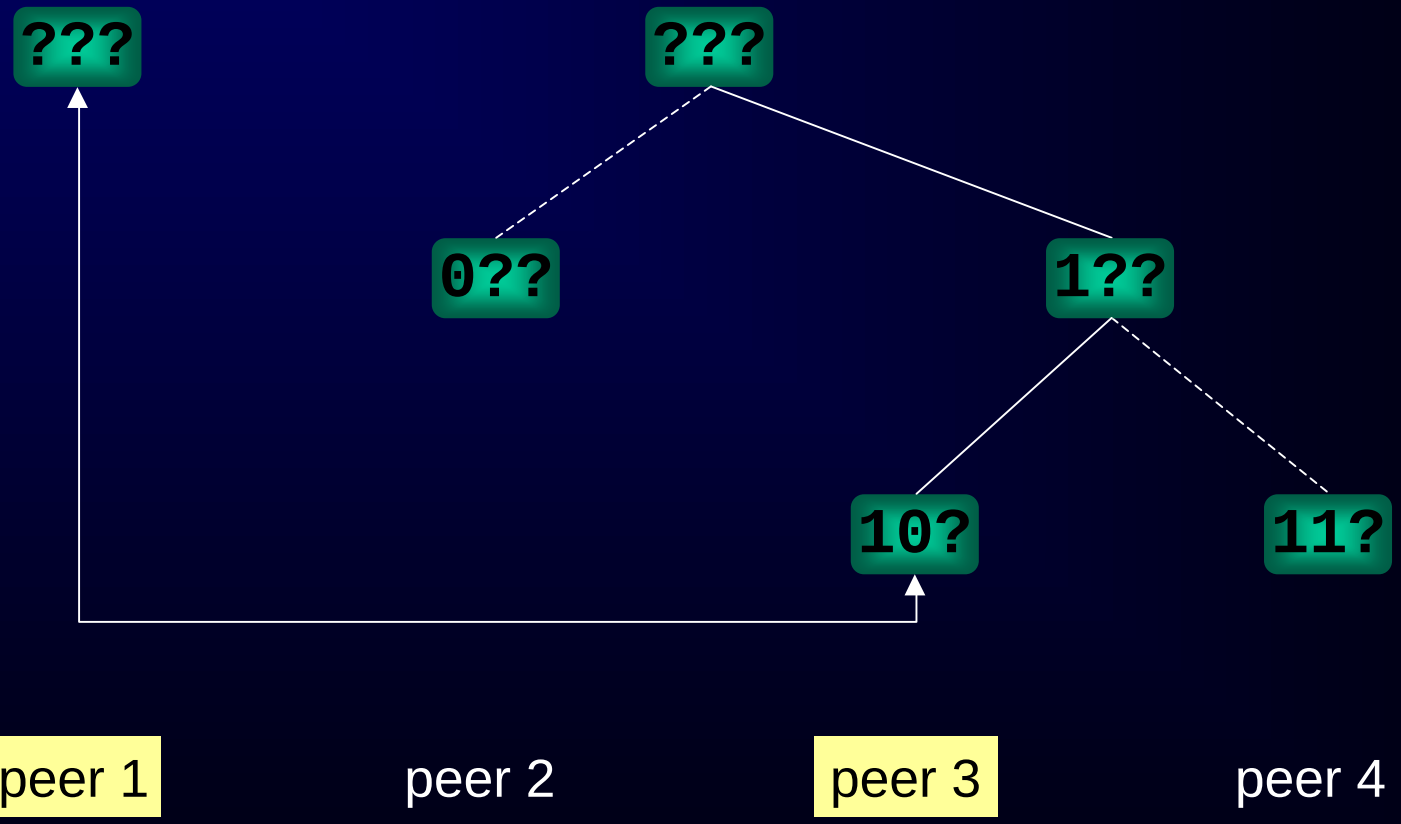


peer 1 peer 2 peer 3 peer 4

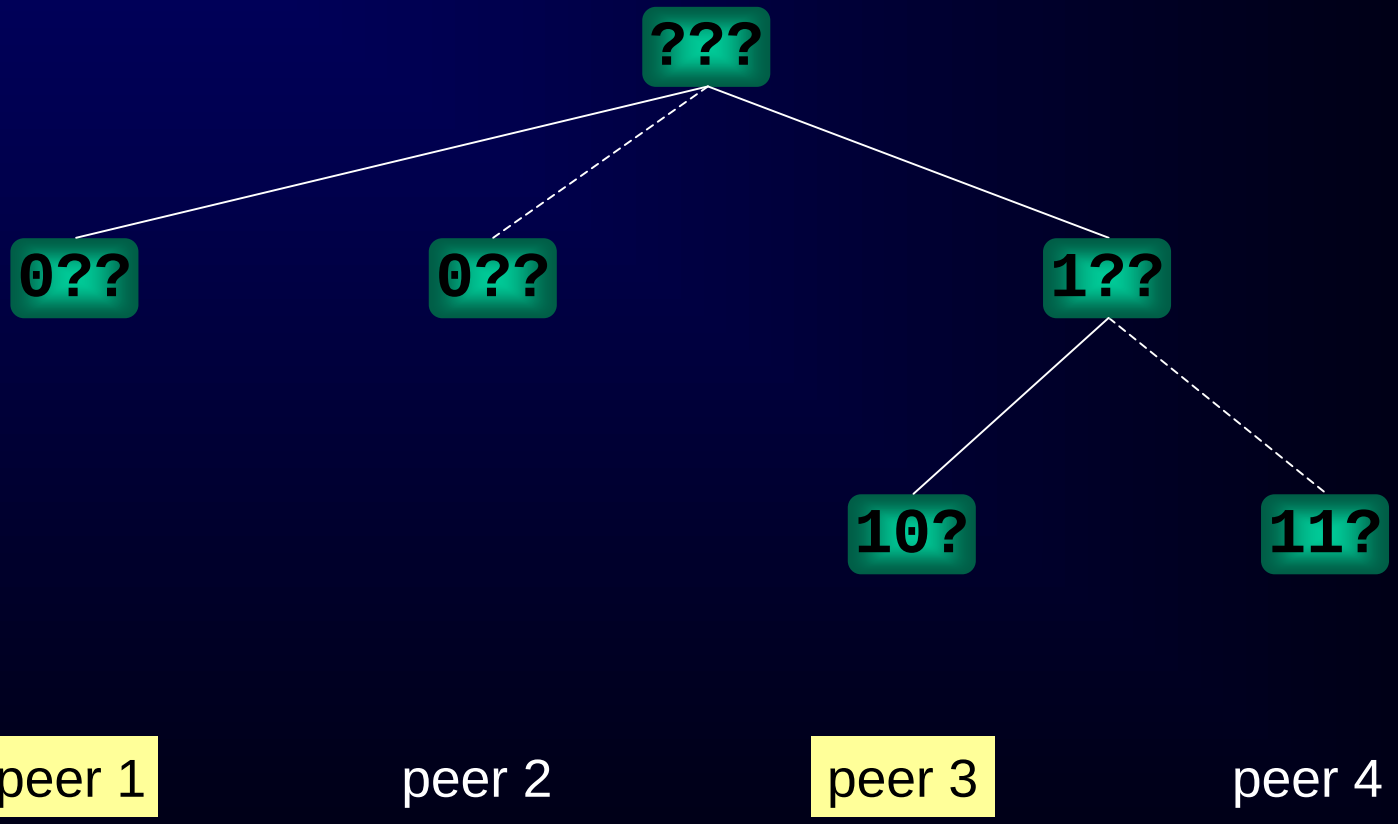
P-Grid Construction Example



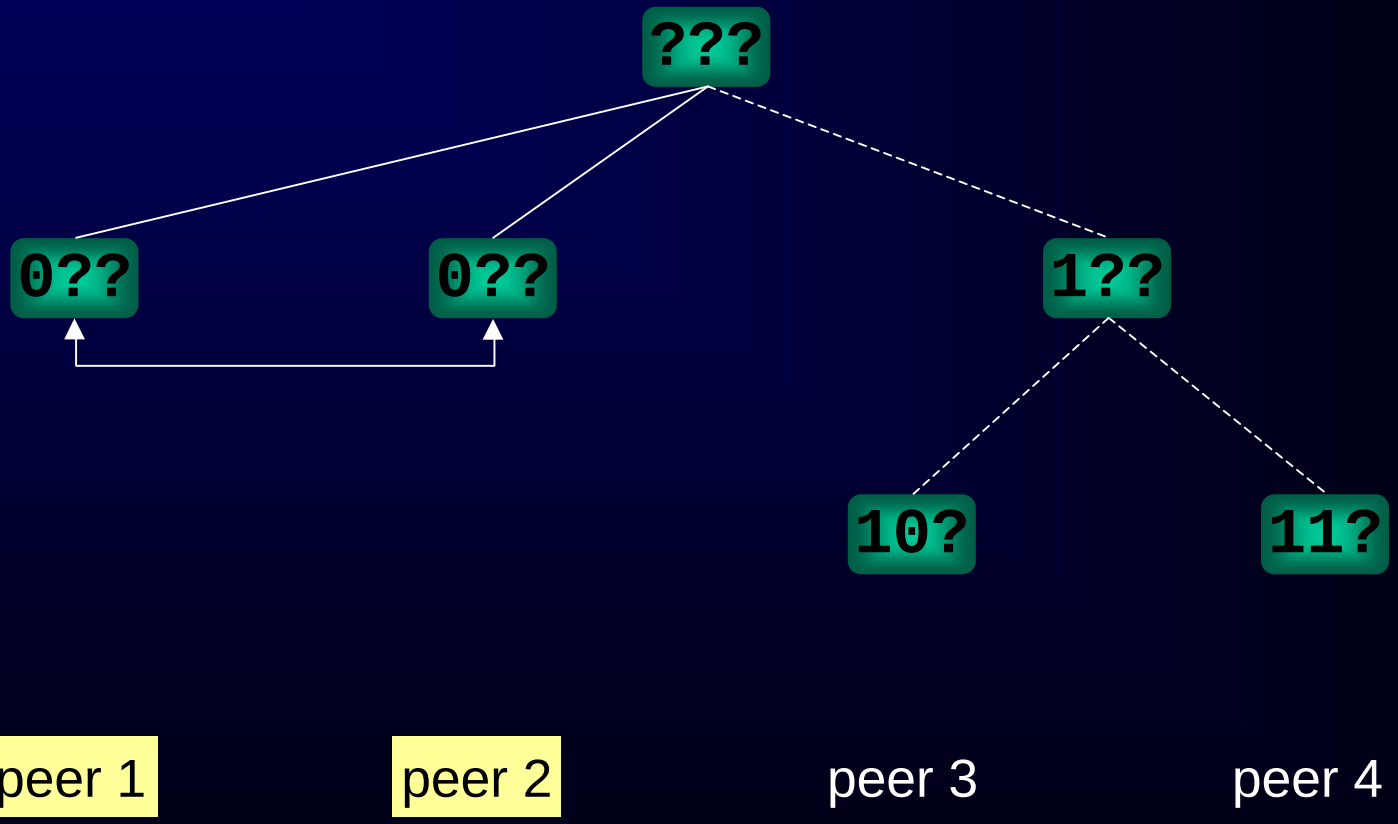
P-Grid Construction Example



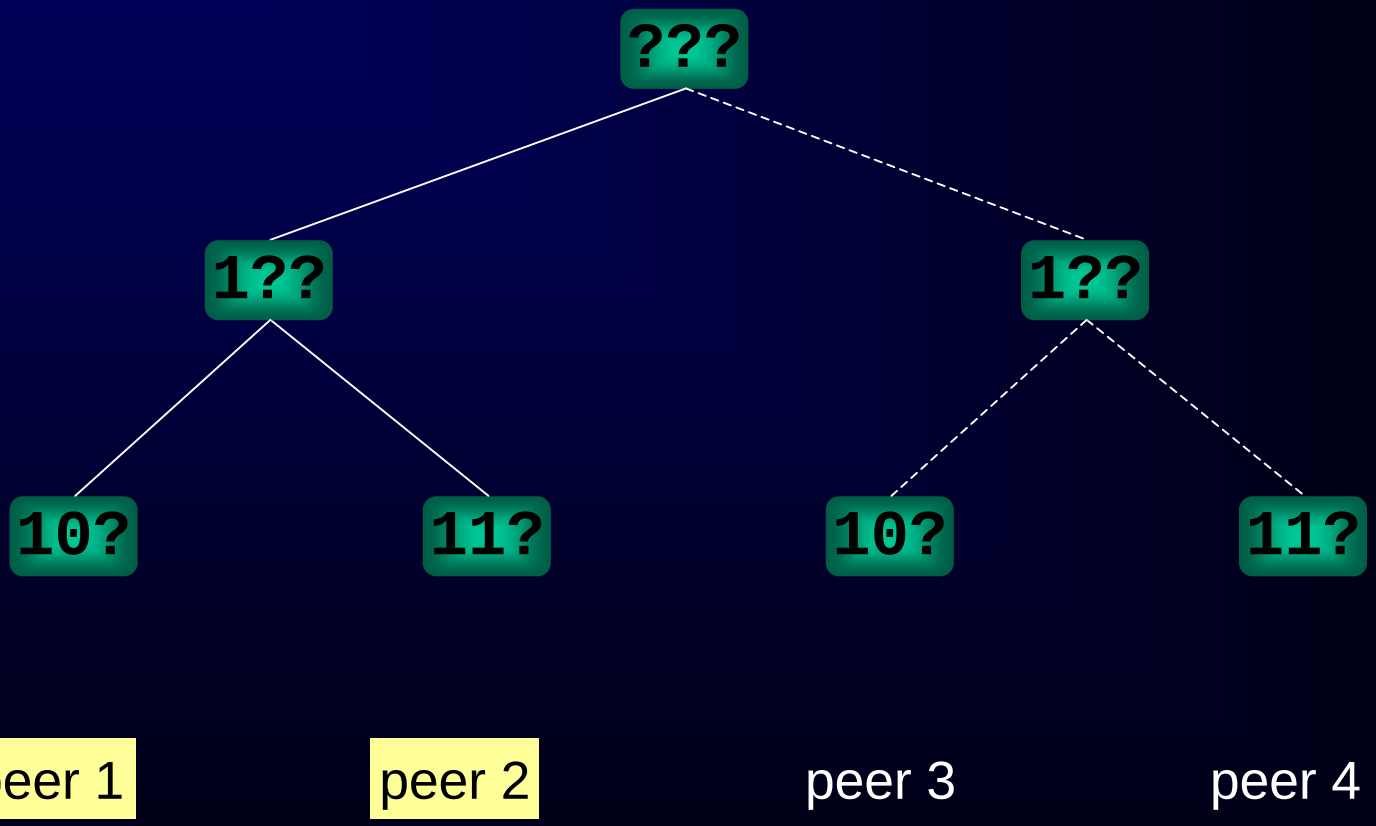
P-Grid Construction Example



P-Grid Construction Example



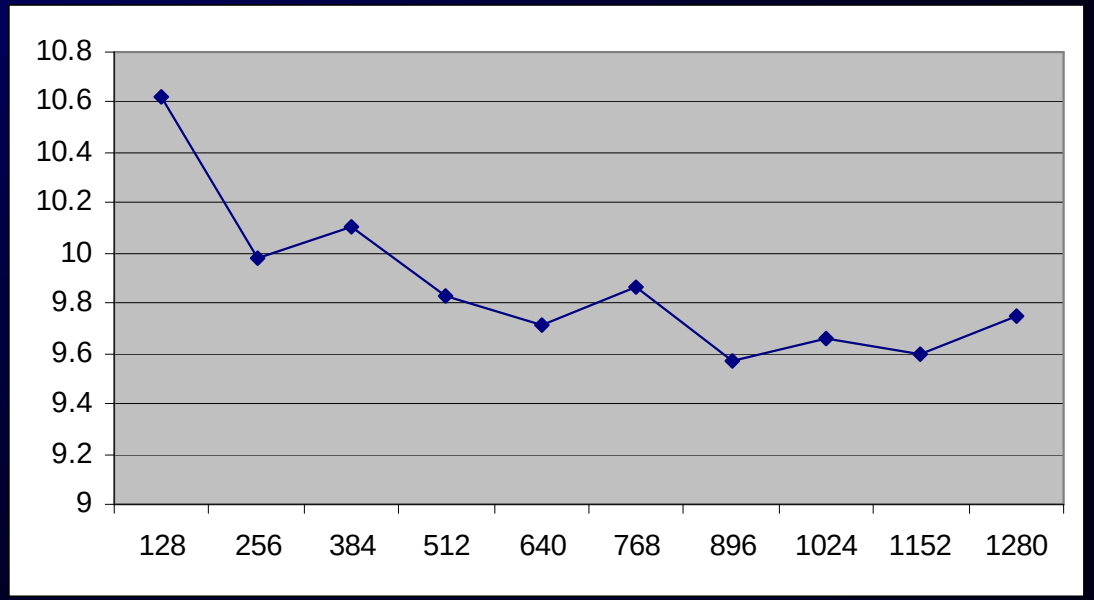
P-Grid Construction Example



Efficiency of P-Grid Construction

- Constructing a tree of depth 6 (64 leaves)

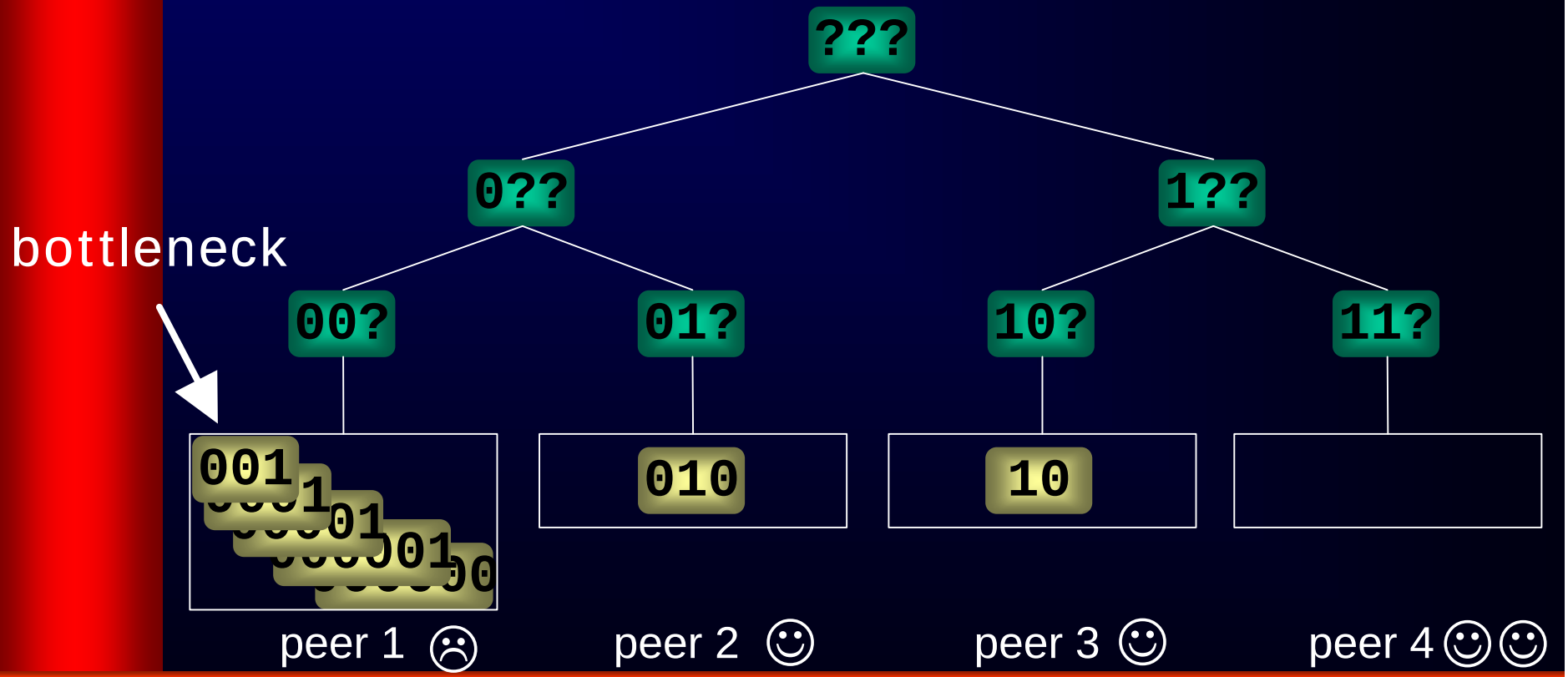
Number of "meetings" per peer



Number of peers

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

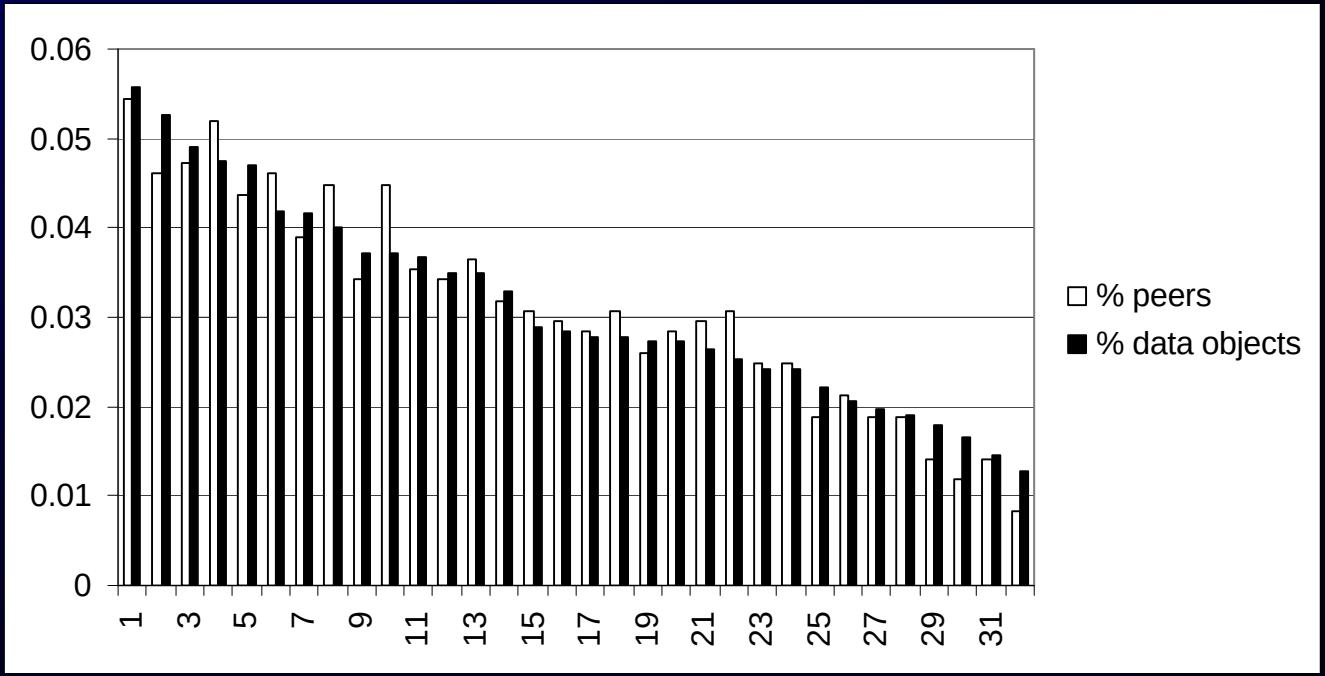
- It makes no sense to create leaves in the tree for data occuring rarely
- Every peer stores initially some data

```
when two peers meet
        peer extends path only if #data items > e
                do the necessary bookkeeping
        otherwise data exchange (duplicate 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

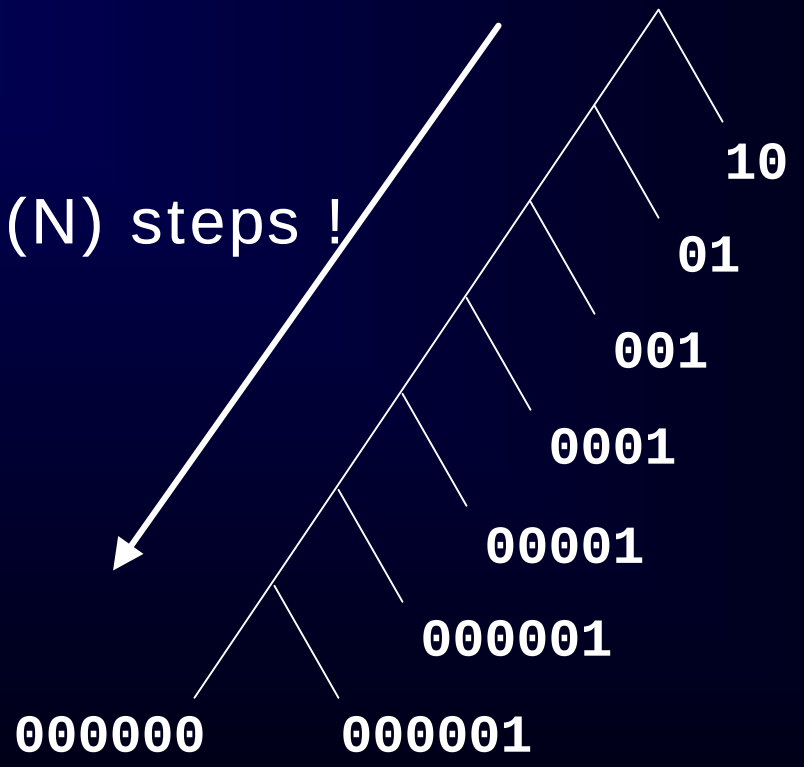


All binary strings of length 5 sorted by frequency

Unbalanced Search Trees

- Problem:
  tree will be deeper where more data items
  ⇒ more work for answering search request

worst case: O(N) steps !



Analysing Required Messages

- Assume only the number of messages required for a search is relevant
  - Multiple nodes in the tree can be traversed without sending a message

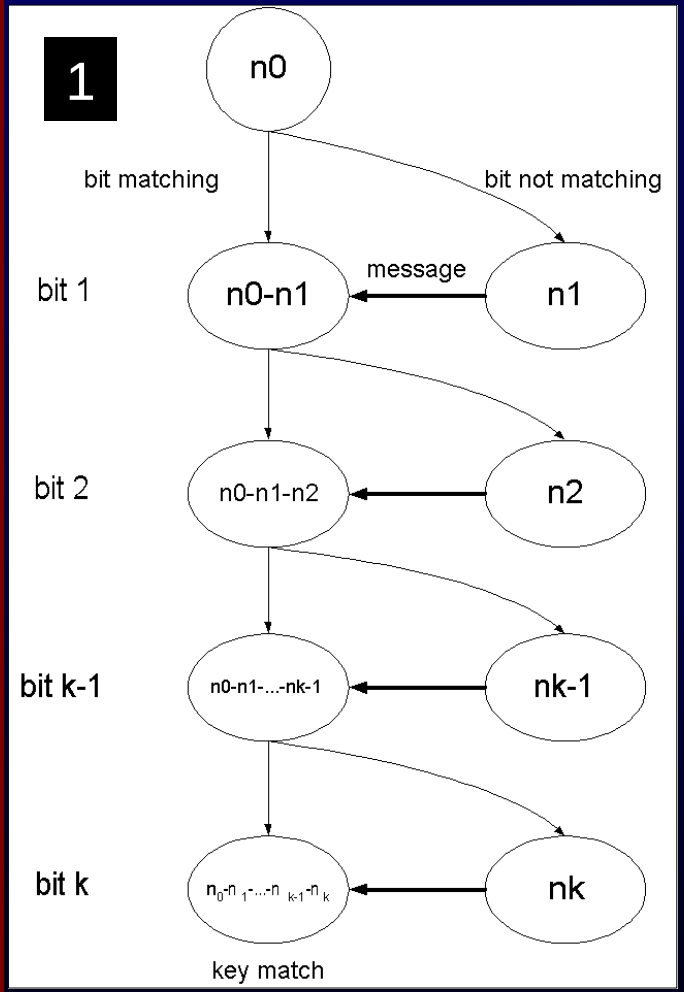
Theorem [WDAS 2002]

IF the probability for a reference to another node occuring in the reference lists is uniform for all references that possibly can occur,
THEN the number of messages required for a search is $O(\log_2(N))$ no matter what shape the P-Grid (tree) is.

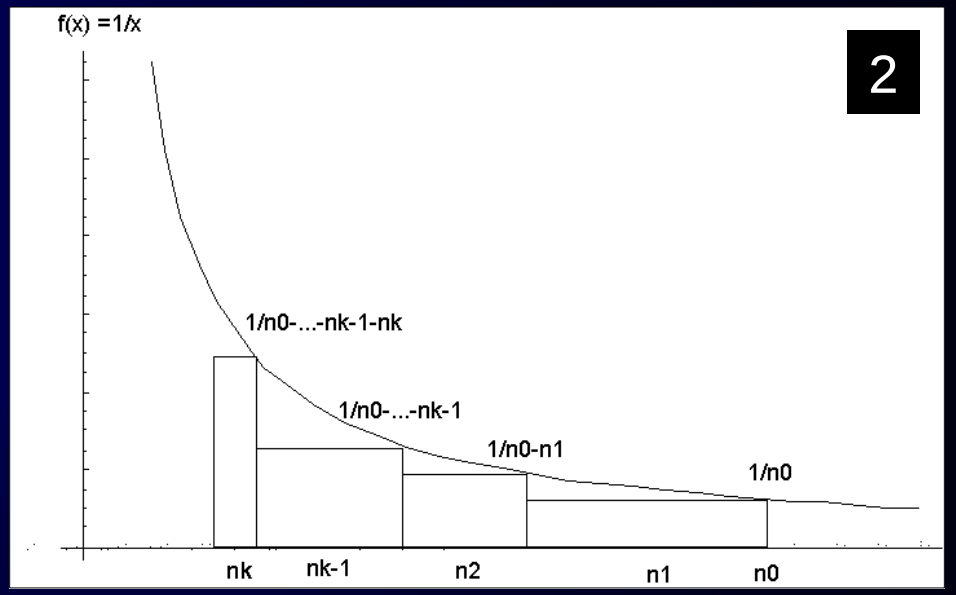
- Equal probability can be achieved by systematically merging the reference lists

Proof of Theorem

1



2



3

$$\sum_{i=1}^{k} \frac{n_i}{n_0 - \ldots - n_{i-1}} < \log n_0 = \log 2 \ \log_2 N.$$

messages

Self-Organization in P-Grid

- Nodes decide through local agreement
  - on their position in the search tree when they meet
  - whether to deepen a search tree based on storage load
- Nodes balance data through local operations
  - Reference distribution
    - required for search efficiency
  - Replica distribution of data objects
    - required for search reliability
- Global "agreements" are only on
  - Type of search requests
  - P-Grid organisation

Practical Aspects of P-Grid

- Implementation exists: feasible
  [IEEE Internet Computing 2002]
- Analysis shows that indexing overhead is reasonable for typical setting
  [Coopis 2001]
- Algorithms for additional replication of more frequently requested data objects
  [ICME 2002]
- Update mechanism based on gossiping
  [EPFL-TR 2002]
- Application for storing reputation data
  [CIKM 2001]
- More complex queries can be supported
  (regular expressions, paths, joins)

Freenet: System Architecture

- Adaptive P2P system which supports publication, replication, and retrieval of data
- Protects anonymity of authors and readers
  - infeasible to determine the origin or destination of data
  - difficult for a node to determine what it stores (files are sent and stored encrypted)

  $\Rightarrow$ nobody can be sued
- Requests are routed to the most likely physical location
  - no central server as in Napster
  - no constrained broadcast as in Gnutella
- Files are referred to in a location independent way
- Dynamic replication of data

OceanStore: System Overview [Rhea01]

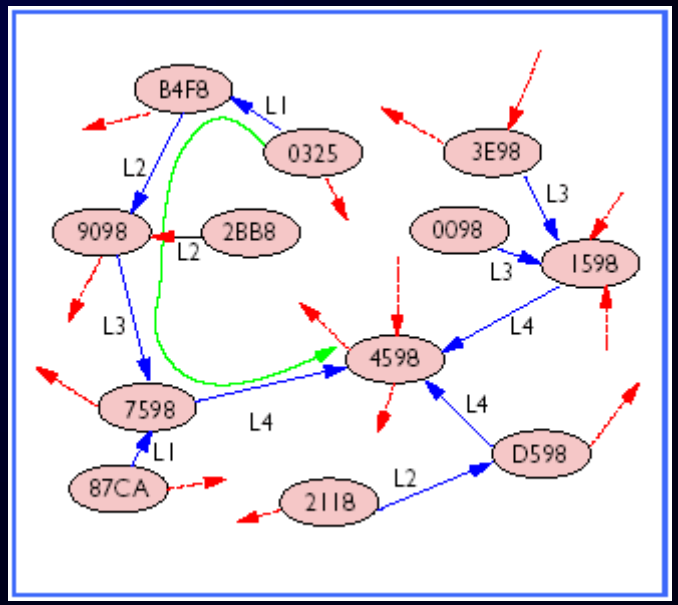
- Internet-based, distributed, global storage infrastructure
- Consists of possibly millions of cooperating servers
- Data is split up in fragments which are stored redundantly on servers
- Sophisticated replication to ensure data availability and performance (introspection-based self-tuning)
- High fault tolerance: monitoring, erasure (m-of-n) coding, self-verifying fragments, automatic repair, byzantine update commitment, etc.
- Automatically recovers from server and network failures, incorporates new resources, and adjusts to usage patterns

OceanStore: Routing Infrastructure

- Data identified by a key must be located => all fragments must be located and verified
- Tapestry subsystem
  - Self-organizing routing and object location system
  - Routing: hashed suffix routing

Tapestry routing example. A potential path for a message originating at node 0325 destined for node 4598.Tapestry routes the message through nodes **** ? ***8 ? **98 ? *598 ? 4598, where asterisks represent wildcards. The role each hop plays in the path is marked with a level number.

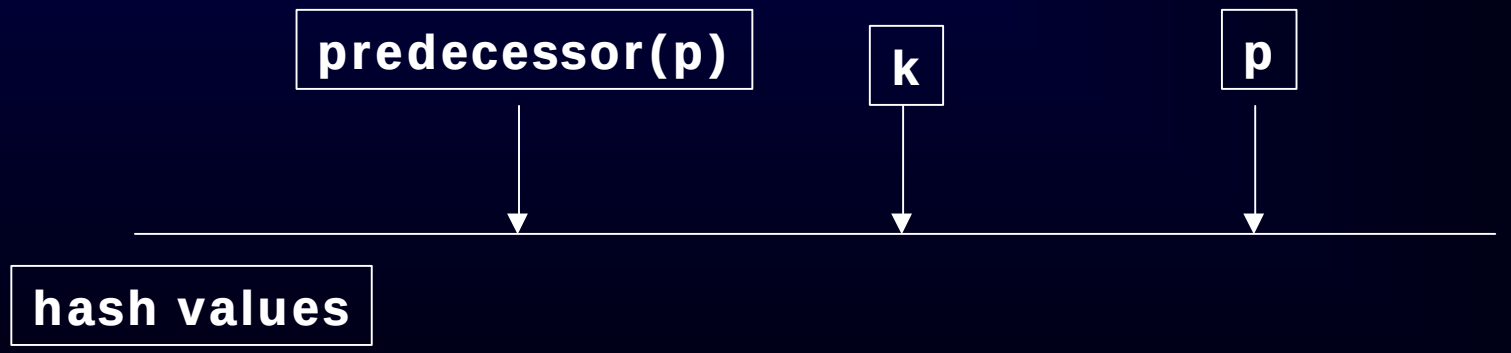
[Rhea01]



CHORD Consistent Hashing

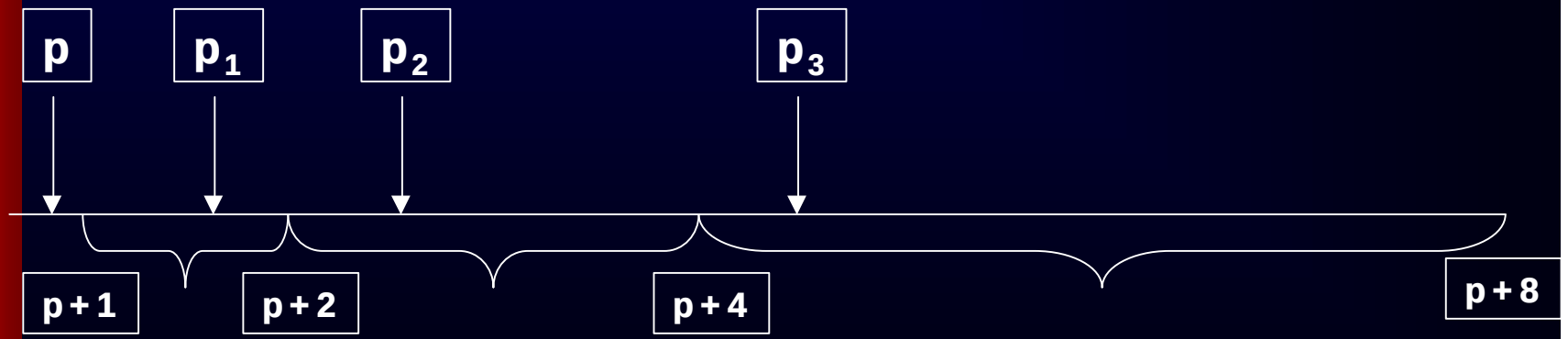
- Based on a hashing of search keys and peer addresses on binary keys of length **m**
- Each peer with hashed identifier p is responsible (=stores values associated with the key) for all hashed keys k such that

$$k \in]\ \text{predecessor}(p),\ p\ ]$$



CHORD Routing Table

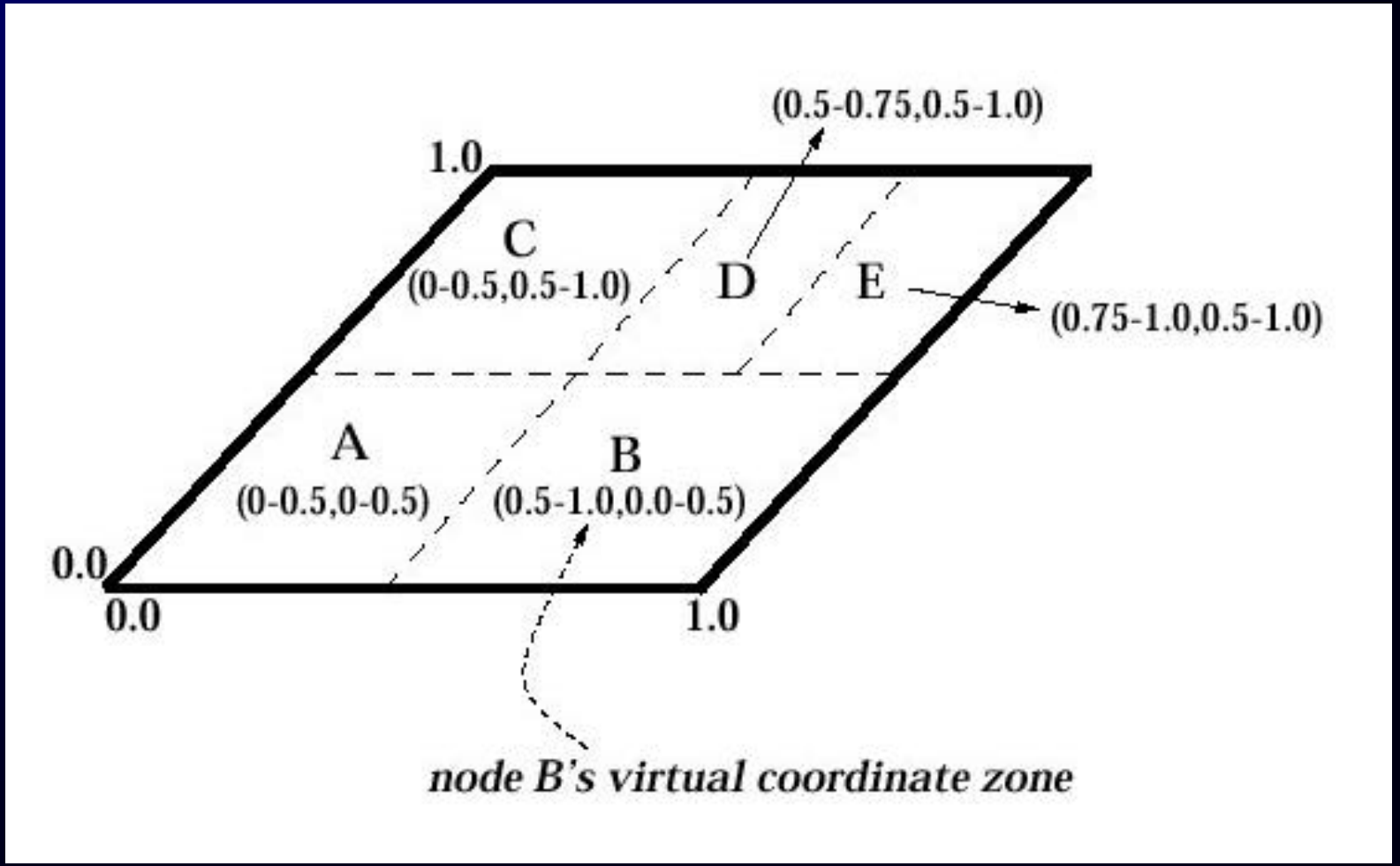
- Each peer p stores a “finger table” consisting of the first peer with hashed identifier p_i such that
  $p_i = \text{successor}(p + 2^{i-1})$ for $i = 1,..,m$
- We write also $p_i = \text{finger}(i, p)$
- A search algorithm ensures the reliable location of the data
  - Complexity O(log n), n nodes in the network



Content-Addressable Networks (CAN)

- Based on hashing of keys into a **d-dimensional space (a torus)**
- Each peer is responsible for keys of a subvolume of the space (a zone)
- Each peer stores the peers responsible for the neighboring zones for routing
- Search requests are greedily forwarded to the peers in the closest zones
- Assignment of peers to zones depends on a random selection made by the peer

CAN Zones (2D Space)



Comparison of Approaches

	Paradigm	Search Type	Search Cost (messages)	Fixed Data Assignment	Common Network Origin
Gnutella	Breadth-first search on graph	String comparison	$2*\sum_{i=0}^{TTL} C*(C-1)^i$	no	no
Freenet	Depth-first search on graph	Equality	O(Log n) ?	no	no
Chord	Implicit binary search trees	Equality	O(Log n)	yes	yes
CAN	d-dimensional space	Equality	O(d n^(1/d))	no	yes
Tapestry	Prefix trees	Equality	O(Log n)	yes	no ?
P-Grid	Binary prefix trees	Prefix	O(Log n)	no	no

Overview

1. P2P Systems : Decentralizing Information Systems

2. P-Grid : Efficient Decentralized Search

3. Semantic Gossiping : Emergent Global Semantics

Overview

1. P2P Systems : Decentralizing Information Systems

2. P-Grid : Efficient Decentralized Search

3. Semantic Gossiping : Emergent Global Semantics

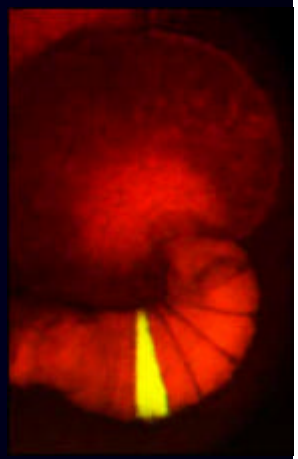
Decentralization: Napster vs. Gnutella

	Napster	Gnutella
Resources: search	**central**	**decentral**
Resources: file exchange	**decentral**	**decentral**
Knowledge: schema	**central**	**trivial**
Knowledge: annotation	**decentral**	**decentral**
	Partially decentralized	Self-Organizing

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: ...MNTS**LEECH**MPKGD...



- 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>** Û **<Species>**
  - **<Author>** Û **<Authors>** etc.
- But what about
  - **<AaMutType>** Û **<DnaMutType>**
  - **<FtKey>** Û **<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>$sp/ID</ID>
FOR $sp IN /SP_entry
WHERE "anglerfish" IN $sp/organism
```

```
Q2=
<ID>$sp/ID</ID>
FOR $sp IN T12
WHERE "anglerfish" IN $sp/organism
```

SwissProt
(known schema)

EMBLChange
(own schema)

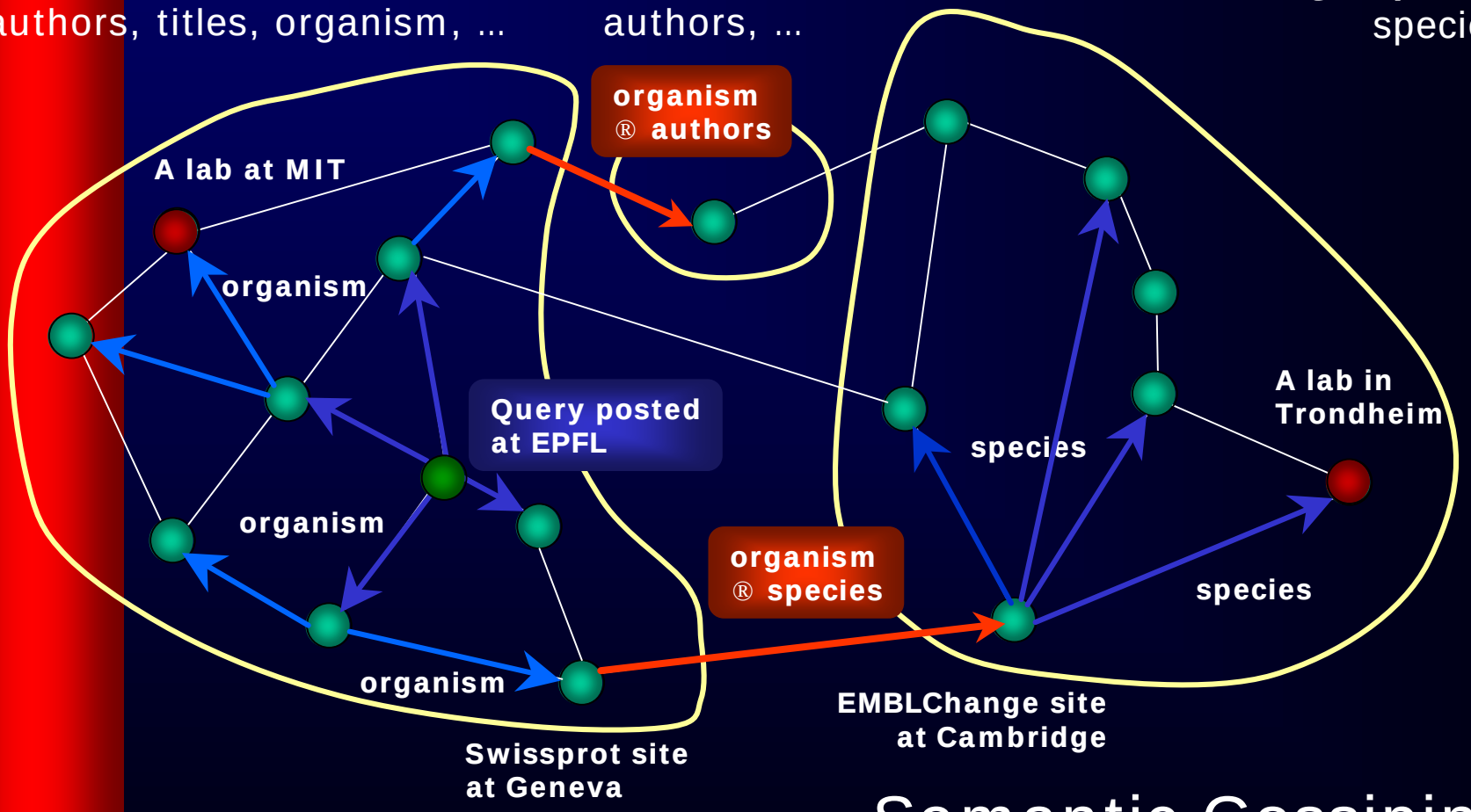
```
<SP_entry>
  <ID>CBPH_LC
  <Authors>Roth
  <Organism>
    Lophius americanus
    (American goosefish)
    (Anglerfish).
  </Organism>
  <Sequence>
    MKQICSIVLL ...
  </Sequence>
</SP_entry>
```

```
T12 =
<SP_entry>
  <ID>$ec/ID</ID>
  <Organism>
    $ec/Species
  </Organism>
  </SP_Entry>
FOR $ec IN /EC_entry
```

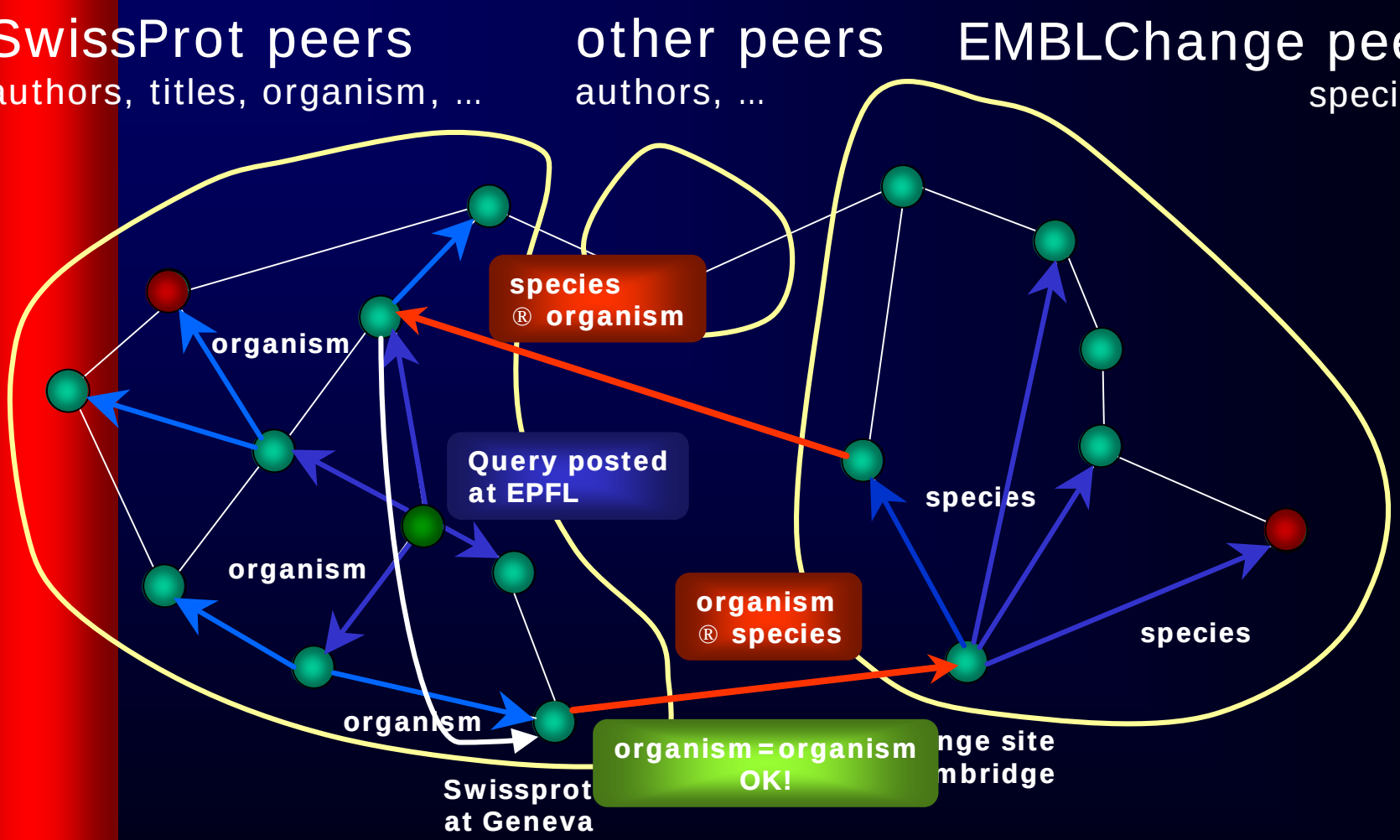
```
  LAJAFGI_5; VRT
</ID>
<Species>
  Lophius americanus
  (anglerfish)
</Species>
SQ>
Sequence 1 BP
/SQ>
/EC_entry>
```

Computer-processable languages: XML, XQuery

Global Semantic Interoperability



How to Detect a Semantic Agreement ?



Check what is preserved in cycles (semantic kernels) !

Research Questions

- Many fundamental problems
  - Erroneous agreements
  - Agreement on schema but not on data
  - Complex data types and mappings
  - Overlapping of data collections
- Approach: algorithms and tools
  - to automatically generate, detect and use local translations
  - identify which are correct with a high probability (via semantic kernels)
  - control of global search (via semantic gossiping)

Conclusion

- New information services without requiring new infrastructure
  - Sharing of existing resources
  - Sharing of existing knowledge
- Technical perspective
  - Self-organization is feasible, efficient, ...
- Social perspective
  - Self-organization preserves autonomy
  - Self-organization requires trust
  - Global behavior based on local agreements
- Models from sociology, economy and biology