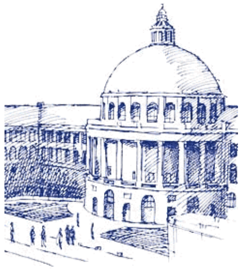
eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System

Thomas Locher, ETH Zurich
Stefan Schmid, ETH Zurich
Roger Wattenhofer, ETH Zurich



Distributed Computing Group

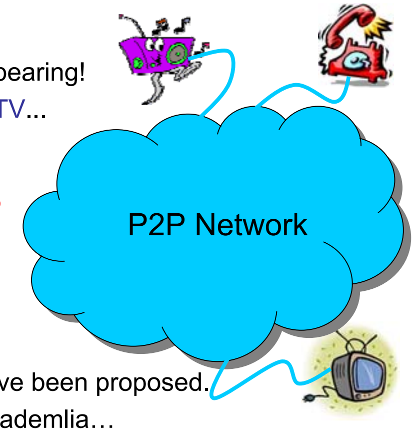
6th IEEE Int. Conference on Peer-to-Peer Computing (P2P)
Cambridge, UK, September 2006

Motivation: Ubiquitous P2P Systems

P2P systems are can be used for many different purposes.
→ File sharing, fast data dissemination, data backup…

More and more applications are appearing!
→ P2P telephony, P2P radio, P2P TV...

Many applications become *possible* because of the paradigm shift to P2P systems! → P2P TV!

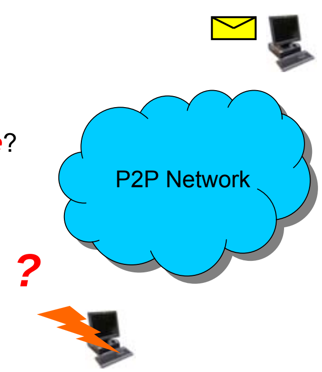


Several structured P2P systems have been proposed.
→ Chord, Pastry, Tapestry, CAN, Kademlia…

Motivation: DHTs Are Not Robust...

All those DHTs provide only one primitive operation:
Map a data item to a key. Peers responsible for the key can be found *efficiently*.

What if the peers stops operating?
→ Peers have to know about it!
What if several peers fail *at the same time*?
→ Structure might break!
Peers join and leave all the time ("churn")!
→ Hard to maintain the structure!

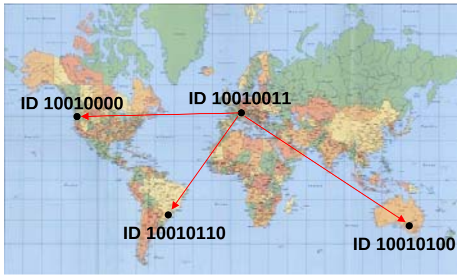


Fault-tolerance has to be *added* to the system!
How is this done?

Motivation: Only Heuristics Applied...

A common technique to introduce fault-tolerance:
Replication of data information across peers with similar IDs.

This replication has to be *repeated* continuously!



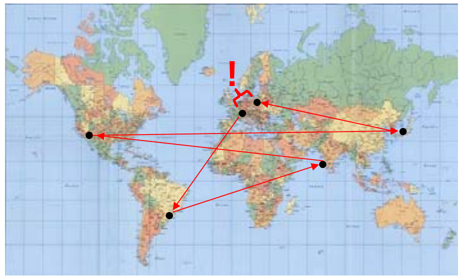
What if the replicating peers are *far away*?
➢ Updating becomes a time-consuming operation!
➢ Slow responses from other peers → Harder to maintain replication!

Motivation: Lack of Locality-Awareness...

Problem: No correlation between peer IDs and distance (no locality-awareness)!

- Only O(log n) hops in lookup paths, but paths might be long.
- No bounds on the *stretch*!

Maximum ratio between length of a path to the direct distance



Consequences:

- Inefficient queries → Long lookup times!
- Inefficent routing table updates → Harder to maintain robustness!

Motivation: More Heuristics...

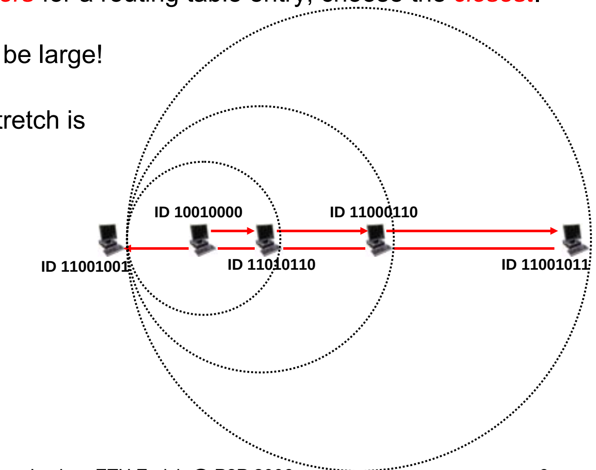
A common technique to introduce some form of locality-awareness: Among *all suitable peers* for a routing table entry, choose the *closest*.

The *stretch* might still be large!

In this example, the stretch is

$2^{O(\log n)} = n^{O(1)}$!

The number of hops!



Motivation: Goal of eQuus

We want a P2P system that has all the typical properties such as a small peer degree, small network diameter, good load balancing etc. and also meets the following requirements:

- Fault-tolerant and resilient to the permanent joining and leaving of peers („churn“).
- The lookup paths should not be much longer than the direct paths to the target peers (small stretch).
- Maintaining the desired network structure does not induce a large message overhead.

Outline

I. Motivation

II. System Overview

III. Results

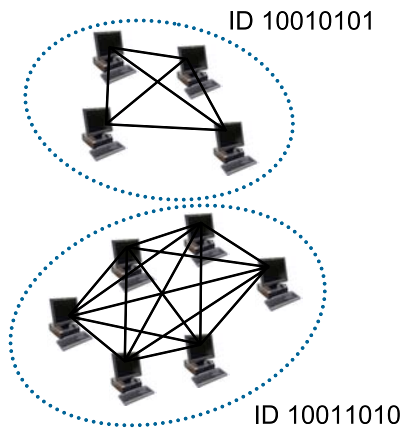
IV. Outlook / Conclusion

System Overview: Cliques

In eQuus, groups of peers form *cliques*!

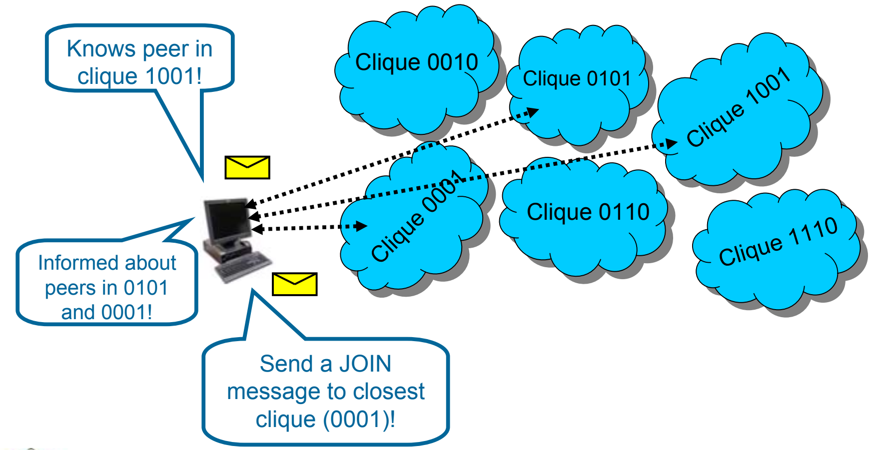
All peers know each other!

- Each clique has a unique ID $\in \{0,1\}^d$ shared among the peers of this clique → Robustness & redundancy!
- New peers always join the closest clique in the network and get the same ID → Locality-Awareness!



System Overview: Join

A joining peer iteratively searches for the closest clique!



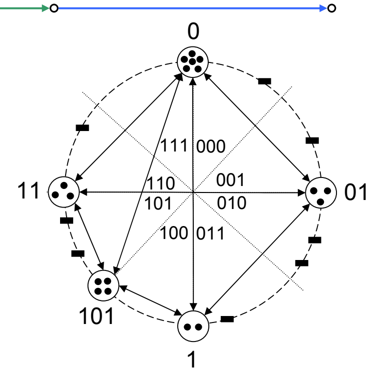
System Overview: Link Structure

Clique interconnections:

Based on *prefix-routing*: For all other $2^b - 1$ combinations of each b (= base) bit block in the routing table, store links to a suitable clique!

Additionally: Links to the *predecessor* and *successor clique*! → Peers are responsible for keys in the range:

[ID of own clique, ID of successor clique)



Example:

Routing table of clique with ID 10001101 (b = 2):

Block	Prefixes
1	00, 01, 11
2	1001, 1010, 1011
3	100000, 100001, 100010
4	10001100, 10001110, 10001111

System Overview: Link Structure

Structure can be maintained *efficiently*:

A constant number to each clique!

- Peers have links to the same cliques, but not to the same peers within those cliques → No synchronization!
- Information about joining peer can be broadcast quickly! → Peers in other cliques are not affected!

Bounds on the clique size are needed:

Lower bound $\mathcal{L} \in O(\log n)$ on the clique size: Avoid data loss!!!

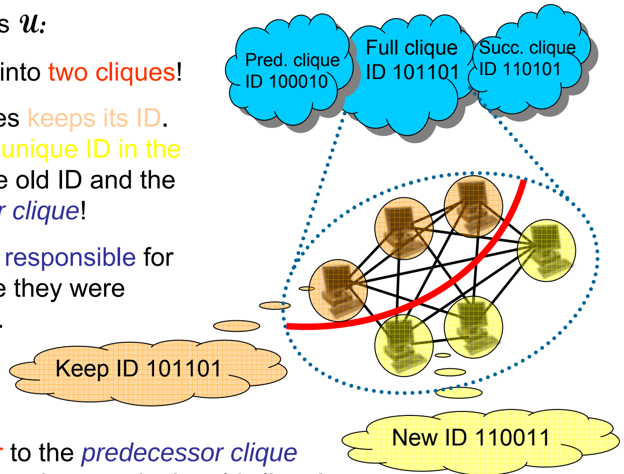
Upper bound $\mathcal{U} \in O(\log n)$ on the clique size: Limit the size of the routing tables (peer degree)!

Additional operations are necessary!

System Overview: Split

Clique size reaches $\mathcal{U}$:

- Clique is SPLIT into two cliques!
- One of the cliques keeps its ID. The other gets the unique ID in the middle between the old ID and the ID of the *successor clique*!
- Both cliques are responsible for half of the ID space they were responsible before.



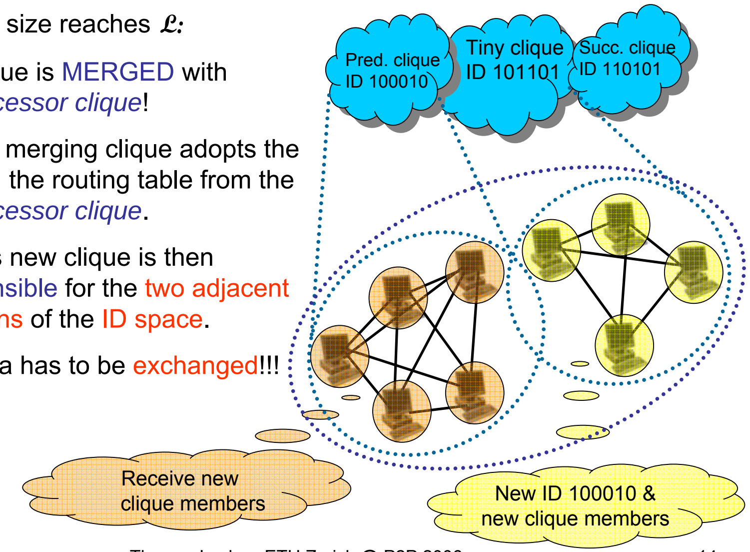
- The closest peer to the *predecessor clique* determines the peers that stay in the old clique!

System Overview: Merge

Clique size reaches $\mathcal{L}$:

- Clique is MERGED with *predecessor clique*!
- The merging clique adopts the ID and the routing table from the *predecessor clique*.
- This new clique is then responsible for the two adjacent fractions of the ID space.

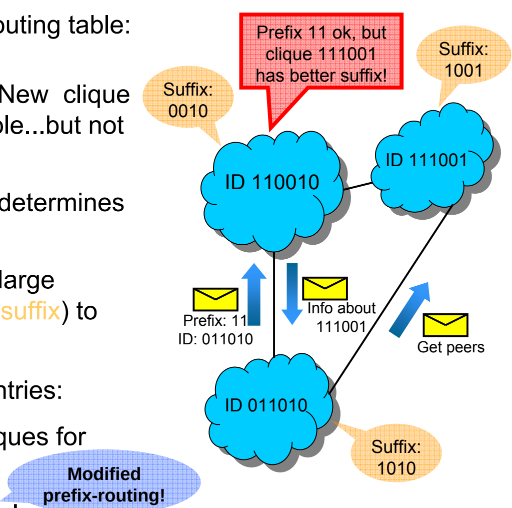
→ Data has to be exchanged!!!



System Overview: Link Maintenance

Building and updating the routing table:

- When a clique is split → New clique has to rebuild its routing table...but not from scratch!
- # bits shared with old ID determines # entries that can be kept!
- Danger of cliques with a large indegree → Use rest of ID (suffix) to solve this problem!
- Two ways to find fresh entries:
  - Ask peers in other cliques for clique information!
  - Use the lookup routine!



Outline

I. Motivation

II. System Overview

III. Results

IV. Outlook / Conclusion

Results: Model

Peer distribution:

- There are n peers in the system.
- Peers are *uniformly distributed* in a *two-dimensional Euclidean* space!

Failure:

- Each peer has the same probability to fail in a specific period of time.

Distance metric:

- The distance between two peers u and v is the *Euclidean distance* between those peers: $d(u,v) = || u - v ||_2$

Results: Fault-Tolerance

Two crucial properties have to be guaranteed:

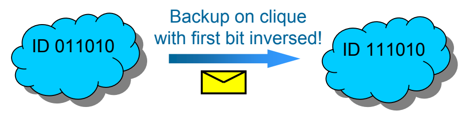
Values clearly depend on the model...

- No data is ever lost!
- The structure does not break even if there is a lot of churn!

Probability of data loss is very small if the minimum clique size is sufficiently large and the link update frequency is large enough!

However, the system is more vulnerable to *correlated failures*: If a large set of close-by peers (= peers in the same cliques!) fail at once (network failure), data will be lost!

Simple solution: Backup all data on clique that is far away!



Results: Churn

Theorem: *If all n peers are uniformly distributed, then* Ω(n) *JOIN/LEAVE events are required in expectation before either a MERGE or SPLIT operation has to be performed.*

- The more peers there are in the network, the better the system can handle churn!!!
- Intuition: More peers results in more cliques where peers can join → Always a large number of peers has to join *somewhere* before *any* clique has to split or merge!
- „Catch": This holds (only?) if peers are uniformly distributed...

Results: Locality-Awareness

Theorem: *The expected stretch of a lookup call in eQuus is at most* $(2^{b/2+1})/(2^{b/2}-1)$ *for a particular base b.*

- Example: Base b = 4 → The expected stretch is at most 8/3 ≈ 2.67!
- Building locality-aware cliques clearly results in a topology with efficient lookups!
- Furthermore, simulations show that this result is conservative!

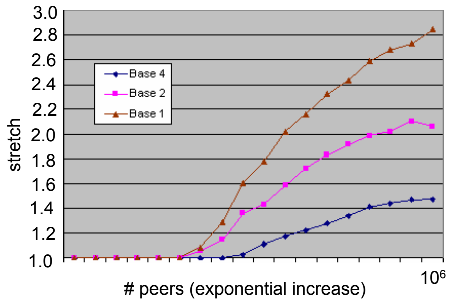
Results: Locality-Awareness

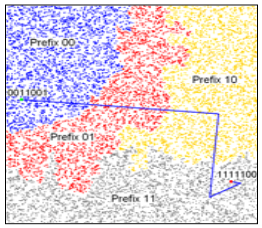
Simulations show that the stretch is much lower in expectation!

➢ If b = 4: The stretch stabilizes between 1.4 and 1.5!

➢ If b = 1: The stretch is less than 3 with 10^6 peers!

A typical simulation result with 10,000 peers and a lookup path (b = 1):





Outline

I. Motivation

II. System Overview

III. Results

IV. Outlook / Conclusion

Outlook: Realistic Model!

The most obvious improvement:

Change the model to a more realistic one!

How?

How are peers distributed on the Internet?

How are JOIN/LEAVE events distributed in a world-wide P2P system???

→ Real world implementation!

Outlook: Load Balancing!

Another crucial problem:

Ensure load balancing among all cliques (peers)!

If peers are *uniformly distributed*, load balancing is not an issue:

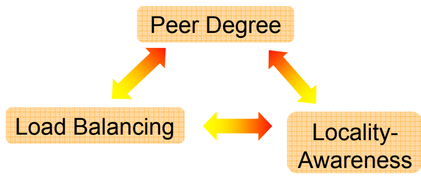
***Theorem**: If all n peers are uniformly distributed and there are D data items, each peer is responsible for at most* **$O(D \log^2 n / n)$** *data items w.h.p.*

Peers in Europe responsible for *half* of all data items?

What can be done in a more realistic model?

Outlook: Load Balancing!

Two different approaches:



- Peer Migration
  - Peers are moved to *predecessor clique* (or *successor clique*), if this clique is responsible for a large fraction of the ID space, but does not contain enough peers!
  - Preserves locality-awareness, but is expensive...
- Key Reassignment
  - Part of the assigned key space is reassigned to other, nearby cliques that have less responsibility!
  - Easier to handle („forward pointer“), but might damage the locality-property...

Conclusion

- eQuus has several desirable properties
  - Resilient to failures & churn
  - Locality-awareness
  - Low message overhead
- Several improvements possible
  - Load balancing
  - Trust issues, incentives...
- Real world implementation
  - PlanetLab study as a first step

Questions and Comments?

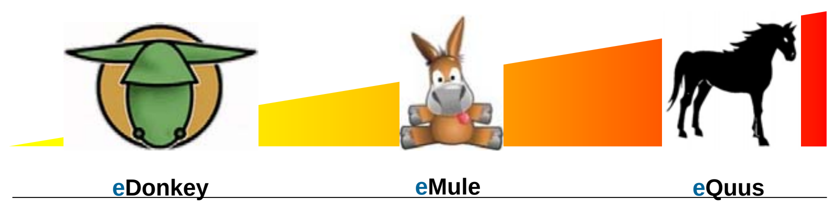
Thank you for your attention!

Thomas Locher
Distributed Computing Group
ETH Zurich, Switzerland
lochert@tik.ee.ethz.ch
http://dcg.ethz.ch/members/thomasl.html

Additional Slides: Name?

Popular P2P systems are traditionally named after animals.....

The protocols evolve (and the animals change...)

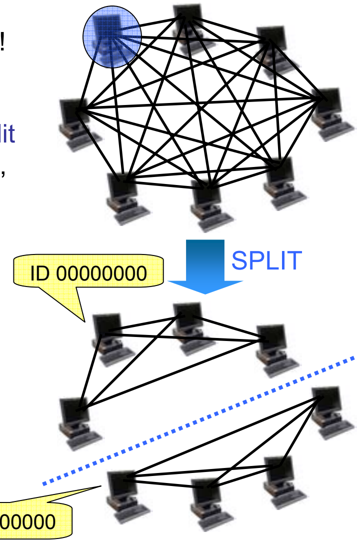


- **„Equus“ is latin for „horse“.**
- **A horse is a stronger and faster animal than a donkey or a mule...**
- **Horses band together, comparable to how robustness is established in eQuus!**

Additional Slides: First Clique

- The first clique has the ID $0^d = \underbrace{000...0}_{d}$!
- As soon as it contains $\mathcal{U}$ peers it is split into the two cliques with IDs 0^d and 10^{d-1}, each containing $\mathcal{U}/2$ peers!
- The peer with the maximum sum of distances to all other peers in the clique keeps the ID 0^d together with the $\mathcal{U}/2 - 1$ closest peers in the clique!

„Push the other clique away!"



Additional Slides: Lookup

Algorithm:

- Find clique ĉ with longest matching prefix in routing table
- Clique ĉ has a longer matching prefix: (Normal case!)
  - Forward lookup request to peer in clique ĉ!
- There is no such clique with a longer matching prefix:
  - Search key > own ID:
    - Forward to the clique with the numerically largest ID among all cliques whose matching prefix is not shorter!
  - Search key < own ID:
    - Forward lookup request to peer in the predecessor clique!

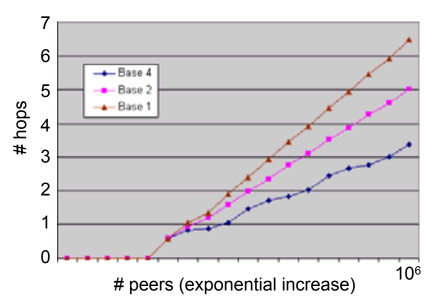
Lookup terminates here!

Additional Slides: Lookup Results

Theorem: *If all n peers are uniformly distributed, then a LOOKUP terminates successfully after at most* $\lceil \log_{2^b} n \rceil + o(1)$ *hops w.h.p., if the routing tables are accurate.*

All cliques know of all other cliques that are relevant for its routing table!

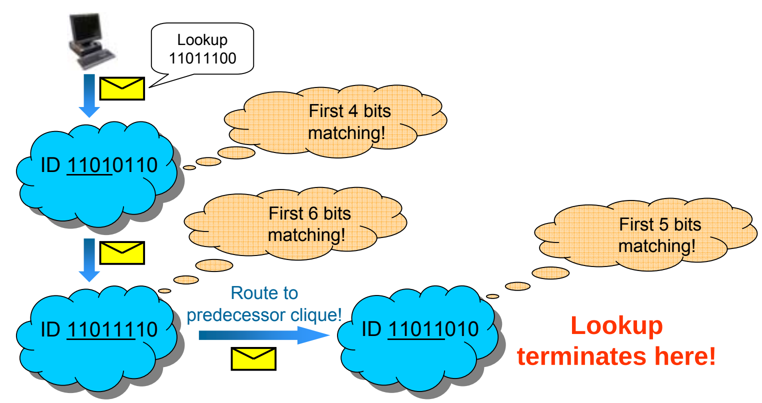
Simulations show that the average # hops is lower than $\lceil \log_{2^b} n \rceil$!



Additional Slides: Lookup Example I

Example:

Peer in clique 10010110 is looking up key 11011100:



Additional Slides: Lookup Example II

Example:

Peer in clique 10111010 is looking up key 01011101: