Churn and Selfishness: Two Peer-to-Peer Computing Challenges

Stefan Schmid
Distributed Computing Group
ETH Zurich, Switzerland
schmiste@tik.ee.ethz.ch

Invited Talk
University of California, Berkeley
380 Soda Hall
March, 2006
Outline of this Talk

- Current research of our group at ETH
  - Based on our papers at IPTPS 2005 and IPTPS 2006

- Two challenges related to P2P topologies
  
  **CHALLENGE 1: Churn**
  - dynamics of P2P systems,
  - i.e., joins and leaves of peers ("churn")
  - our approach to maintain desirable properties in spite of churn

  **CHALLENGE 2: Selfishness**
  - impact of selfish behavior on P2P topologies
  - How bad are topologies formed by selfish peers?
  - Stability of topologies formed by selfish peers?
CHALLENGE 1:

Fast and Concurrent Joins and Leaves ("Churn")
Dynamic Peer-to-Peer Systems

• Properties compared to centralized client/server approach
  – Availability
  – Efficiency
  – Etc.

• However, P2P systems are
  – composed of unreliable desktop machines
  – under control of individual users

=> Peers may join and leave the network at any time!
Churn:

Permanent joins and leaves

How to maintain desirable properties such as
- Connectivity,
- Network diameter,
- Peer degree?
Challenge 1: Churn

- Motivation for adversarial (worst-case) churn
- Components of our system
- Assembling the components
- Results and Conclusion
Motivation

• Why permanent churn?

Peers join system for one hour on average

Hundreds of changes per second with millions of peers in the system!

• Why adversarial (worst-case) churn?

E.g., a crawler takes down neighboring machines (attacks weakest part) rather than randomly chosen peers!
The Adversary

• Model worst-case faults with an adversary $ADV(J,L,\lambda)$

• $ADV(J,L,\lambda)$ has complete visibility of the entire state of the system

• May add at most $J$ and remove at most $L$ peers in any time period of length $\lambda$

• Note: Adversary is not Byzantine!
Synchronous Model

- Our system is synchronous, i.e., our algorithms run in **rounds**
  - One round: receive messages, local computation, send messages

- However: Real distributed systems are **asynchronous**!
  - Algorithms can still be used: local synchronizers

- Notion of time necessary to **bound the adversary**
  - E.g. 1 round = max. RTT
A First Approach

- Fault-tolerant hypercube?
- What if number of peers is not $2^i$?
- How to prevent degeneration?
- Where to store data?

Idea: Simulate the hypercube!
Simulated Hypercube System

Simulation: Node consists of several peers! Such a hypercube can be maintained against ADV(J,L,λ)!

Basic components:

• Route peers to sparse areas
  Token distribution algorithm!

• Adapt dimension
  Information aggregation algorithm!
Components: Peer Distribution and Information Aggregation

Peer Distribution
• Goal: Distribute peers evenly among all hypercube nodes in order to balance and adversarial churn
• Basically a token distribution problem

Counting the total number of peers (information aggregation)
• Goal: Estimate the total number of peers in the system and adapt the dimension accordingly
Peer Distribution (1)

Algorithm: Cycle over dimensions and balance!

Perfectly balanced after $d$ steps!
Peer Distribution (2)

- But peers are not fractional!

- And an adversary inserts at most $J$ and removes at most $L$ peers per step!

Theorem 1: Given adversary $ADV(J,L,1)$, discrepancy never exceeds $2J+2L+d$!
Components: Peer Distribution and Information Aggregation

Peer Distribution
• Goal: Distribute peers evenly among all hypercube nodes in order to balance biased adversarial churn
• Basically a token distribution problem

Counting the total number of peers (information aggregation)
• Goal: Estimate the total number of peers in the system and adapt the dimension accordingly
Information Aggregation (1)

- Goal: Provide the same (and good!) estimation of the total number of peers presently in the system to all nodes
  - Thresholds for expansion and reduction

- Means: Exploit again the recursive structure of the hypercube!
Algorithm: Count peers in every sub-cube by exchange with corresponding neighbor!
Information Aggregation (3)

- But again, we have a concurrent adversary!

- Solution: Pipelined execution!

Theorem 2: The information aggregation algorithm yields the same estimation to all nodes. Moreover, this number represents the correct state of the system \(d\) steps ago!
Composing the Components

- Our system permanently runs
  - Peer distribution algorithm to balance biased churn
  - Information aggregation algorithm to estimate total number of peers and change dimension accordingly

> But: How are peers connected inside a node, and how are the edges of the hypercube represented?

> And: Where is the data of the DHT stored?
Distributed Hash Table

- **Hash function** determines node where data item is replicated

- Problem: Peer which has to move to another node must replace all data items.

- Idea: Divide peers of a node into core and periphery
  - Core peers store data,
  - Peripheral peers are used for peer distribution
Intra- and Interconnections

• Peers inside a node are completely connected.

• Peers are connected to all core peers of all neighboring nodes.
  – May be improved: Lower peer degree by using a matching.
Maintenance Algorithm

- **Maintenance algorithm** runs in *phases*
  - Phase = 6 rounds

- In phase $i$:
  - **Snapshot** of the state of the system in round 1
  - One exchange to estimate number of peers in sub-cubes (*information aggregation*)
  - Balances tokens in dimension $i \mod d$
  - Dimension change if necessary

---

All based on the snapshot made in round 1, ignoring the changes that have happened in-between!
Results

• Given an adversary $ADV(d+1,d+1,6)$...
  => Peer discrepancy at most $5d+4$ (Theorem 1)
  => Total number of peers with delay $d$ (Theorem 2)

• ... we have, in spite of $ADV(O(\log n), O(\log n), 1)$:
  – always at least one core peer per node (no data lost!),
  – peer degree $O(\log n)$ (asymptotically optimal!),
  – network diameter $O(\log n)$. 
Discussion

• Simulated topology: A simple **blueprint** for dynamic P2P systems!
  – Requires algorithms for **token distribution and information aggregation** on the topology.
  – Straight-forward for **skip graphs**
  – Also possible for **pancake graphs**!  
    ( Diameter = Degree = O(log n / loglog n) )

• A lot of future work!
  – A first step only: dynamics of P2P systems offer many research challenges!
  – E.g.: Other **dynamics models**, **self-stabilization** after larger changes, etc.!
  – E.g.: **Selfishness** => see CHALLENGE 2
  – E.g.: also **measurement studies** are subject to current research:
    • Churn in file sharing systems?
    • Churn in Skype? (=> IPTPS 2006)
eQuus: An Alternative Approach with Low Stretch (1)

• eQuus
  – Optimized for random joins/leavs rather than worst-cae
  – Hypercube too restrictive
  – Token distribution is expensive
  – Adding locality awareness!

• “Simulated Chord”
  – Local split and merge only
  – According to constant thresholds
  – Split operation according to latencies!
eQuus: An Alternative Approach with Low Stretch (2)

- Split and merge happen seldom
  - If joins and leave \textit{uniformly distributed}: balls-into-bins
  - \textbf{Small stretches} if nodes are \textit{uniformly distributed} (= roughly direct paths used)
CHALLENGE 2:

Selfish Peers
Challenge 1 -> Challenge 2

• Simulated hypercube topology is fine…

• … if peers act according to protocol!

• However, in practice, peers can perform selfishly!
Motivation

Power of Peer-to-Peer Computing =
Accumulation of Resources of Individual Peers

- CPU Cycles
- Memory
- Bandwidth
- ...

Collaboration is of peers is vital!

However, many free riders in practice!
Motivation

• Free riding
  – Downloading without uploading
  – Using storage of other peers without contributing own disk space
  – Etc.

• In this talk: selfish neighbor selection in unstructured P2P systems

• Goals of selfish peer:

  (1) Maintain links only to a few neighbors (small out-degree)

  (2) Small latencies to all other peers in the system (fast lookups)

What is the impact on the P2P topologies?
Challenge 2: Road-Map

- Problem statement
  - Game-theoretic tools
  - How good / bad are topologies formed by selfish peers?
  - Stability of topologies formed by selfish peers
  - Conclusion
Problem Statement (1)

- $n$ peers $\{\pi_0, \ldots, \pi_{n-1}\}$

- distributed in a **metric space**
  - Metric space defines distances between peers
  - triangle inequality, etc.
  - E.g., Euclidean plane
Problem Statement (2)

• Each peer can choose…
  – to which
  – and how many
  – … other peers its connects

• Yields a directed graph $G$
Problem Statement (3)

- Goal of a selfish peer:

  (1) Maintain a small number of neighbors only (out-degree)

  (2) Small stretches to all other peers in the system

    - Fast lookups!
    - Shortest distance using edges of peers in G...
    - ... divided by shortest direct distance

- Only little memory used
- Small maintenance overhead
Problem Statement (4)

• Cost of a peer:
  – Number of neighbors (out-degree) times a parameter $\alpha$
  – plus stretches to all other peers
  – $\alpha$ captures the trade-off between link and stretch cost

\[
\text{cost}_i = \alpha \text{outdeg}_i + \sum_{i \neq j} \text{stretch}_G(\pi_i, \pi_j)
\]

• Goal of a peer: Minimize its cost!
Challenge 2: Road-Map

Problem statement

- Game-theoretic tools
- How good / bad are topologies formed by selfish peers?
- Stability of topologies formed by selfish peers
- Conclusion
Game-theoretic Tools (1)

• **Social Cost**
  – Sum of costs of all individual peers:

  \[
  \text{Cost} = \sum_i \text{cost}_i = \sum_i (\alpha \text{ outdeg}_i + \sum_{i \neq j} \text{stretch}_{G}(\pi_i, \pi_j))
  \]

• **Social Optimum OPT**
  – Topology with minimal social cost of a given problem instance
  – \( \Rightarrow \) “topology formed by collaborating peers”!

What topologies do selfish peers form?

\( \Rightarrow \) Concepts of **Nash equilibrium** and **Price of Anarchy**
Game-theoretic Tools (2)

- **Nash equilibrium**
  - “Result” of selfish behavior => “topology formed by selfish peers”
  - Topology in which no peer can reduce its costs by changing its neighbor set
  - In the following, let NASH be social cost of worst equilibrium

- **Price of Anarchy**
  - Captures the impact of selfish behavior by comparison with optimal solution
  - Formally: social costs of worst Nash equilibrium divided by optimal social cost

\[
\text{PoA} = \max_i \{\text{NASH}(I) / \text{OPT}(I)\}
\]
Challenge 2: Road-Map

- Problem statement

Game-theoretic tools

- How good / bad are topologies formed by selfish peers?
- Stability of topologies formed by selfish peers
- Conclusion
Analysis: Social Optimum

• For connectivity, at least $n$ links are necessary
  – $\Rightarrow$ OPT $\geq \alpha \cdot n$

• Each peer has at least stretch 1 to all other peers
  – $\Rightarrow$ OPT $\geq n \cdot (n-1) \cdot 1 = \Omega(n^2)$

Theorem: Optimal social costs are at least

$$\text{OPT} \in \Omega(\alpha \cdot n + n^2)$$
Analysis: Social Cost of Nash Equilibria

• In any Nash equilibrium, no stretch exceeds $\alpha + 1$
  – Otherwise, it’s worth connecting to the corresponding peer
  – Holds for any metric space!

• A peer can connect to at most $n-1$ other peers

• Thus: $\text{cost}_i \leq \alpha \ O(n) + (\alpha + 1) \ O(n)$
  => social cost $\text{Cost} \in O(\alpha \ n^2)$

Theorem:

In any metric space, $\text{NASH} \in O(\alpha \ n^2)$
Analysis: Price of Anarchy (Upper Bound)

• Since $\text{OPT} = \Omega(\alpha n + n^2)$ ...

• … and since $\text{NASH} = O(\alpha n^2)$,

• we have the following upper bound for the price of anarchy:

Theorem:

In any metric space, $\text{PoA} \in O(\min\{\alpha, n\})$. 
Analysis: Price of Anarchy (Lower Bound) (1)

- Price of anarchy is tight, i.e., it also holds that

\[ \text{PoA} \in \Omega(\min\{\alpha, n\}) \]

- This is already true in a 1-dimensional Euclidean space:

Peer: \( \pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \ldots, \pi_{i-1}, \pi_i, \pi_{i+1}, \ldots, \pi_n \)

Position: \( \frac{1}{2}, \alpha, \frac{1}{2} \alpha^2, \alpha^3, \frac{1}{2} \alpha^4, \ldots, \frac{1}{2} \alpha^{i-2}, \alpha^{i-1}, \frac{1}{2} \alpha^i, \ldots, \frac{1}{2} \alpha^{n-1} \)
Price of Anarchy: Lower Bound (2)

\[ \pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \ldots, \pi_{i-1}, \pi_i, \pi_{i+1}, \ldots, \pi_n \]

Peer: \[ \frac{1}{2}, \alpha, \frac{1}{2} \alpha^2, \alpha^3, \frac{1}{2} \alpha^4, \ldots, \frac{1}{2} \alpha^{i-2}, \alpha^{i-1}, \frac{1}{2} \alpha^i, \ldots, \frac{1}{2} \alpha^{n-1} \]

Position:

Note: Social optimum is at most \( O(\alpha \ n + n^2) \):

To prove:

1. “is a selfish topology” = instance forms a \textit{Nash equilibrium}
2. “has large costs compared to OPT”
   = the social cost of this instance is \( \Theta(\alpha \ n^2) \)

Note: Social optimum is at most \( O(\alpha \ n + n^2) \)
Price of Anarchy: Lower Bound (3)

- Proof Sketch: Nash?
  - Even peers:
    - For connectivity, at least one link to a peer on the left is needed
    - With this link, all peers on the left can be reached with an optimal stretch 1
    - No link to the right can reduce the stretch costs to other peers by more than $\alpha$
  - Odd peers:
    - For connectivity, at least one link to a peer on the left is needed
    - With this link, all peers on the left can be reached with an optimal stretch 1
    - Moreover, it can be shown that all alternative or additional links to the right entail larger costs
Price of Anarchy: Lower Bound (4)

• Idea why social cost are $\Theta(\alpha n^2)$: $\Theta(n^2)$ stretches of size $\Theta(\alpha)$

• The stretches from all odd peers $i$ to a even peers $j > i$ have stretch $> \alpha/2$

• And also the stretches between even peer $i$ and even peer $j > i$ are $> \alpha/2$
Price of Anarchy

Theorem: The price of anarchy is
\[ \text{PoA} \in \Theta(\min\{\alpha, n\}) \]

- PoA can grow linearly in the total number of peers
- PoA can grow linearly in the relative importance of degree costs \( \alpha \)
Challenge 2: Road-Map

• Problem statement

• Game-theoretic tools

How good / bad are topologies formed by selfish peers?

• Stability of topologies formed by selfish peers

• Conclusion
Stability (1)

- Peers change their neighbors to improve their individual costs.

How long thus it take until no peer has an incentive to change its neighbors anymore?

Theorem:

Even in the absence of churn, peer mobility or other sources of dynamism, the system may never stabilize (i.e., P2P system never reaches a pure Nash equilibrium)!
Stability (2)

- Example for $\alpha=0.6$

- Euclidean plane:

\[
\begin{align*}
\pi_a & \quad 2.14 & \pi_b & \quad 1 & \pi_c \\
\pi_1 & \quad 2 & \pi_2 & \quad 2 & 2+\delta \\
1.96 & \quad 2-2\delta & & & \delta \text{...arbitrary small number}
\end{align*}
\]
Stability (3)

- Example sequence:

  Again initial situation
  => Changes repeat forever!

- Generally, it can be shown that there is no set of links for this instance where no peer has an incentive to change.
Stability (4)

- So far: no Nash equilibrium for $\alpha=0.6$

- But example can be extended for $\alpha$ of all magnitudes:
  - Replace single peers by group of $k=n/5$ very close peers on a line
  - No pure Nash equilibrium for $\alpha=0.6k$
Challenge 2: Road-Map

• Problem statement

• Game-theoretic tools

• How good / bad are topologies formed by selfish peers?

Stability of topologies formed by selfish peers

• Conclusion
Conclusion

• Unstructured topologies created by selfish peers

• Efficiency of topology deteriorates linearly in the relative importance of links compared to stretch costs, and in the number of peers

• Instable even in static environments

• Future Work:
  - Complexity of stability? NP-hard!
  - Routing or congestion aspects?
  - Other forms of selfish behavior?
  - More local view of peers?
  - Mechanism design?
Churn and Selfishness: Two P2P Challenges

Thank you for your attention!

Questions? Comments?

Further reading: