Self-organizing Overlay Networks

Kai Shen
Dept. of Computer Science, University of Rochester

Some Concepts

Self-organizing networks:
- Can be formed spontaneously
- Node joining/leaving and network partitioning occur at high rate

Peer-to-peer systems:
- Non-hierarchical architecture
- All nodes are functionally equal

Overlay network:
- A logical network laid on top of another network; can be organized quickly with the help of the underlying network
- Most often overlay networks on top of the Internet

Design Objectives

- Scalability: able to support large number of nodes
  - runtime overhead at each node is slow-scaling - goes up slowly along with the system size: O(n) is terrible, O(\log n) is good.
  - runtime overhead includes processing cost, space requirement, and network bandwidth consumption
- Tolerance of long-term or short-term node failures
  - losing a few objects is fine, but node failures should not cripple the system.
  - overhead of handling each failure is non-scaling.

Self-organizing Overlay Network

- Object location lookup
  - finding a single distinct object over a sea of them
- Resource/service discovery
  - finding one of many replica resources/services over a sea of them
  - preferably finding the one nearby
- Relevance searching
  - many are relevant with different relevance
  - the more relevant, the better
-Massively-parallel interactive games
  - highly interactive - real-time performance constraints
Object/resource lookup: Gnutella

Who has P-Diddy.mp3?

I do

I do

I do

non-hierarchical, query flooding.

Query Flooding

- Object location lookup
  - Finding a single distinct object over a sea of them
    - too much overhead with uncontrolled flooding; no guarantee for flooding with limited depth
- Resource/service discovery
  - Finding one of many replica resources/services over a sea of them
  - Preferably finding the one nearby
  - Looks like a good fit, but how to stop further flood when the subject is found?
- Relevance searching
  - Many are relevant with different relevance
  - The more relevant, the better
  - Similar to the resource/service discovery

Object Location Lookup

"Find a single distinct object over a sea of them"

Design objectives:

- Scalability: slow-scaling overhead at each node
  - Lookup cost
  - Maintenance cost
  - Storage requirement
- Tolerance of long-term or short-term node failures
  - Losing a few objects is fine, but node failures should not cripple the system.
  - Overhead of handling each failure is slow-scaling

Hashing + heuristics-based routing

"P-Diddy.mp3" is hashed into a virtual location "L"
- Every node forwards each query to a neighbor heuristically closer to "L"

Can I have P-Diddy.mp3?

Here is P-Diddy.mp3. Store it somewhere.
Chord [Stoica et al. 2001]

- each data object is mapped into the first node towards the clockwise direction in the circular space
- remaining questions:
  - what each node needs to maintain?
  - how to perform the lookup?
  - how to repair the system who nodes join and fail?

Two Naive Solutions

- Solution #1: every node maintains the link to the node next in the circular ID space
  - lookup cost
  - space requirement
  - repair cost
- Solution #2: every node maintains the link to everybody else
  - lookup cost
  - space requirement
  - repair cost

Chord [Stoica et al. 2001]

- every node maintains the links to
  - the node half-way in the circular ID space
  - the node a quarter-way across the circular ID space
  - the node an eighth-way across the circular IP space
  - ...

Content Addressable Network [Ratnasamy et al. 2001]

- each data object is hashed into a virtual point in the space
- each data object is mapped into the node that owns the region
- every node maintains the links to its immediate neighbors

Questions: lookup cost? storage requirement? repair cost?