System Design for Large Scale

Gnutella

Early Design

- Fully distributed solution to P2P file sharing
- Partially reandomized overlay network.
    - Each node i connects to a number ki of other nodes (this number can vary across nodes, as well as allocated bandwidths)
- Bootstrapping → done by HTTP hosted host caches and by local host caches from previous sessions
    - Due to **high churn**, local host caches can quickly become **outdated**
- Routing → based on flooding and reverse path routing

Protocol

- **PING and PONG messages** - discover new nodes
    - PINGs are flooded
    - PONGs are answered by along reverse paths
- **QUERY and QUERY RESPONSES** - provide search capabilities on content textual descriptions
    - Queries are flooded and replies back propagated
    - The answer set on the requesting node slowly grows with time until the diameter, or maximum hops, is reached
- **GET and PUSH requests** - used to initiate file transfers between peers
    - PUSH used to circumvent single firewalls that woiuld block a GET in a given direction

This early design was found out not to scale, and PING/PONG traffic was dominant in the overlay.

Improved

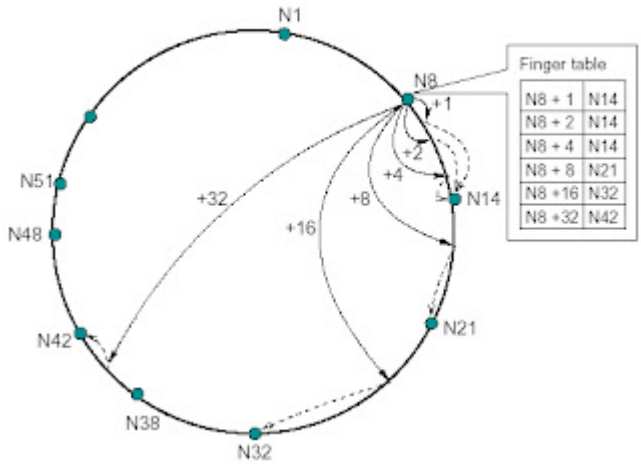
- **Super peer**
    - Act like as in the early Gnutella overlay while shielding traffic and mediating access to client peers → two-tier architecture is formed
    - Mediate search and only contact target peers with a high likelihood of having the searched for content

Distributed Hash Tables

- Provide ways of mapping keys to network nodes.
    - Node joins and leaves should be accounted for in the protocols, in order to preserve some structure in the routing supporting the DHT.

Chord

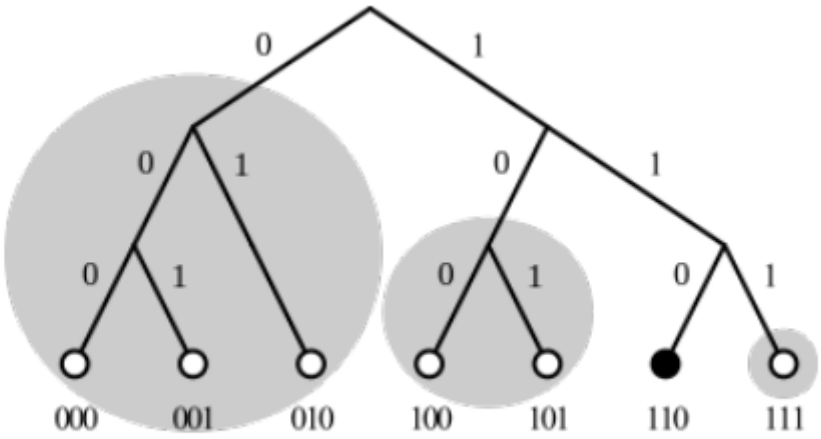
- Nodes and keys are assigned probabilistic unique ids in id space from 0 to $2^m - 1$.
    - Nodes ids (IPs) and keys are hashed and m bits are taken
- Keys and nodes are arranged in an ordered **circle modulo** 2^m
    - For a given key and a given set of nodes, it is possible to determine the successor node (nodeID >= keyID) of that key position . This node will store the key
- It must be possible to contacto an arbitraty node and ask it to find the sucessor node for an arbitrary key
- Each node knows the IP address and id of clockwise m other nodes, and r vicinity nodes in both directions
- The ith entry in node n indicates the first node s that succeeds n by at least 2^(i−1)
- Nodes keep O(log n) knowledge on other nodes and routing takes O(log n) steps.



Kademlia

- Nodes and Keys share a 160 bits space of ids. Keys are stored on “close by” nodes.
- Id distance is computed by a XOR metric. XOR is an interesting symmetric distance metric that respects the triangle property.
- Routing is symmetric and alternative next hops can be chosen for low latency or parallel routing.
- Routing tables: list for each bit of the node id.
- A node in list position i, must have bits 0 to i – 1 identical to the list owner, a different i th bit, and can differ from position i onwards. Its easy to find nodes for

the first positions.



For node 110, groups must match initial sequences: $\perp, 1, 11$

- To account for failing nodes and alternative paths in each position up to k nodes are stored. k is about 20.
- Candidate node uptimes is considered when competing for k limited positions.