SCOM / SRSI
Overlays for Content Distribution
Aka P2P Networks

2018-19

Ana Aguiar

DEEC, FEUP

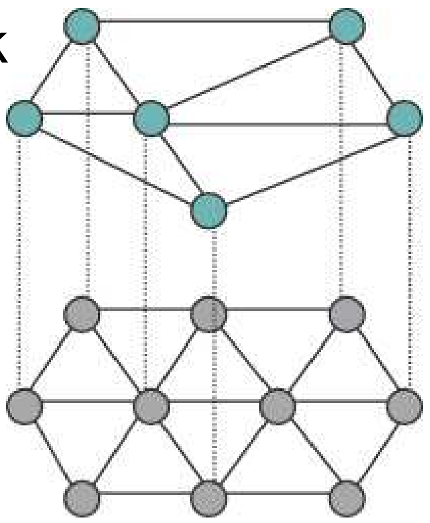
Images from “Computer Networks: a Systems Approach”, L. Peterson, B. Davie

P2P NETWORKS

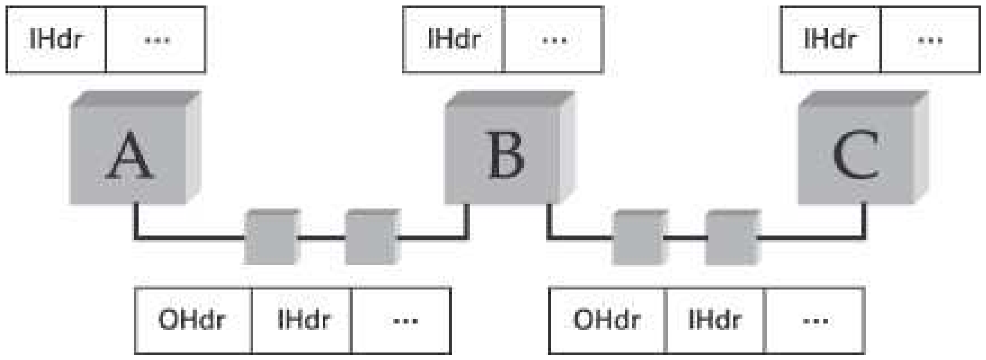
Overlay Networks

Logical networks implemented on top of physical networks

- Only some nodes of physical network belong to overlay network
- Links between overlay nodes can span multiple physical links and nodes
- Overlay networks are invisible from physical networks
- Implemented at the application level



Example Overlay Tunnel



Overlay data packets are encapsulated in physical network packets

Nodes A, B and C can see the physical and overlay addresses of overlay data packets

- They can use either to route packets

Physical network nodes can see only the physical network addresses of the data packets

- Physical network can only use physical network address to route

Peer-to-Peer Overlay Networks

"Peer-to-peer systems are distributed systems consisting of interconnected ***nodes able to self-organize*** *into network topologies* ***with the purpose of sharing resources*** *such as content, CPU cycles, storage and bandwidth,* ***capable of adapting to failures*** *and* ***accommodating transient populations of nodes*** *while maintaining acceptable connectivity and performance* ***without requiring the intermediation or support of a global centralized server or authority****."*

in A Survey of Peer-to-Peer Content Distribution Technologies, ACM Computing Surveys, 2004

Peer-to-Peer Overlay Networks

- De-centralised, self-organised networks of peer nodes on top of IP networks
    - Peers can come and go anytime
    - Peers can have both client and server roles
    - Peers share their resources
    - Neighbour peers in the overlay network may not be neighbours in the IP network
- No central coordination in finding and accessing resources
- Uses capacities that lay at the edge of the networks
    - Enables sharing CPU time, storage, contents, bandwidth
- Typical services
    - File sharing, streaming, personal communication, distributed computing

Challenges

What is necessary to offer services on top of such a network?

- Content placement
- Content search/ finding

What is necessary to set-up and maintain such a network?

- Peer joining and leaving
- Incentives to cooperation, if non-managed (not covered)
- Reputation management, if non-managed (not covered)

STRUCTURED P2P NETWORKS

19/10/18

Structured P2P Networks

Network topology is controlled

Content placement is controlled

Search uses Distributed Hash Table (DHT) or variations thereof

Advantages:

- Probabilistic search delay bounds for rare content

Disadvantages:

- Content is moved every time a peer joins or leaves network
- Higher overhead and delay for looking up popular content

Useful for controlled environments, like data centers, CDN

Distributed Hash Table (DHT)

Hash table

- Data structure that maps identifying keys to their pair values
- Identifying keys are mapped to indexes of the pair values

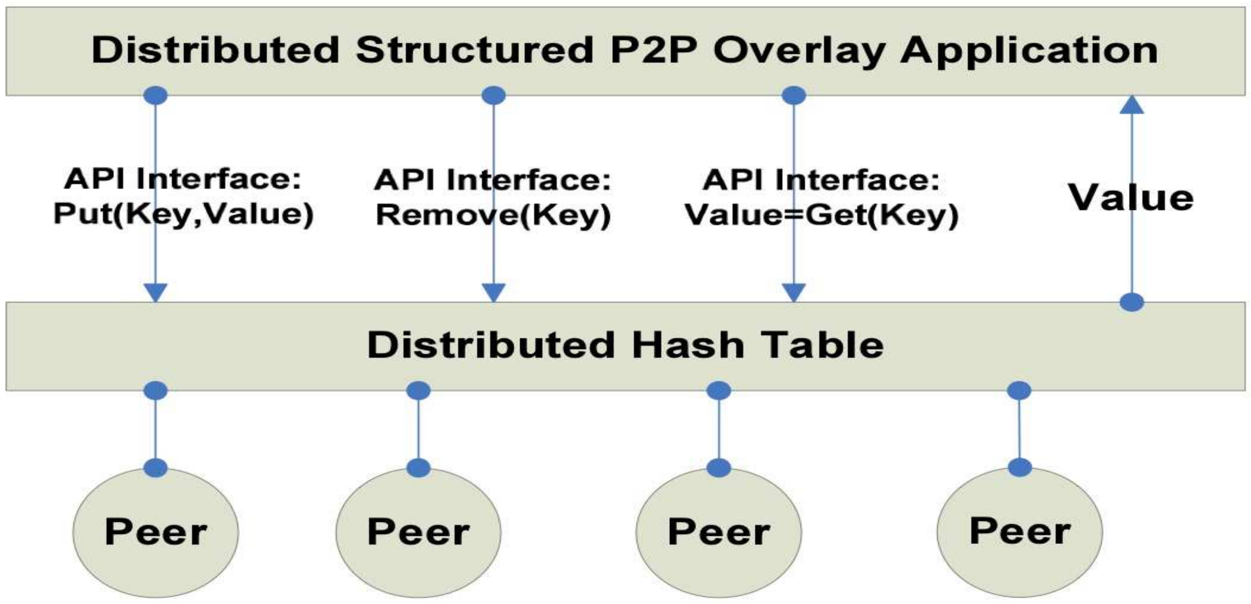
More efficient insert and lookup than other data structures

- Lookup independent of number of values in the table
- Often insertions and deletions in constant time

DHT

- Distributed computation of keys, since hash function is know to all peers.

DHT


Distributed Structured P2P Overlay Application
API Interface: Put(Key,Value)
API Interface: Remove(Key)
API Interface: Value=Get(Key)
Value
Distributed Hash Table
Peer
Peer
Peer
Peer

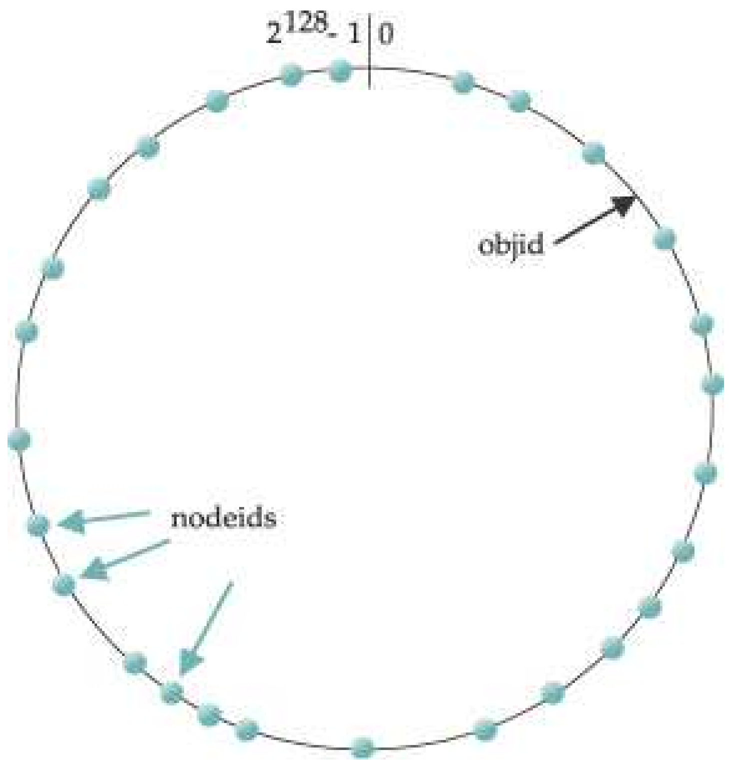
Chord Search

Map objects and nodes onto same key space

- Node: Hash(IP address)
- Content: Hash(Content name)

Object with key k is located in the successor node of k: node whose key is equal to or follows k in the ring

Low amount of contents moved when node leaves or joins



Chord Properties

Assigns keys to nodes with *consistent hashing*

- Each node holds at most (1+e)*K/N keys, e=O(logN)
  - k = #keys, N = #nodes
- When node N+1 joins the ring, at most O(K/N) keys change hands
- When Nth node joins or leaves only O(1/N) fraction of keys move
- Lookup grows as O(logN), N=#nodes

Chord Search

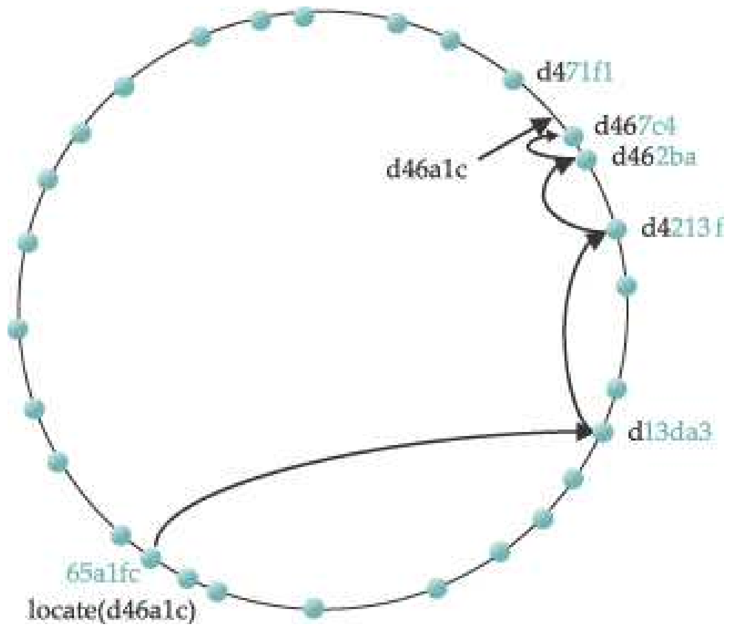
How to find the IP address of a node?

Each node knows of its successor

Query is passed around the circle until it reaches node

May not be very efficient in a network of N nodes

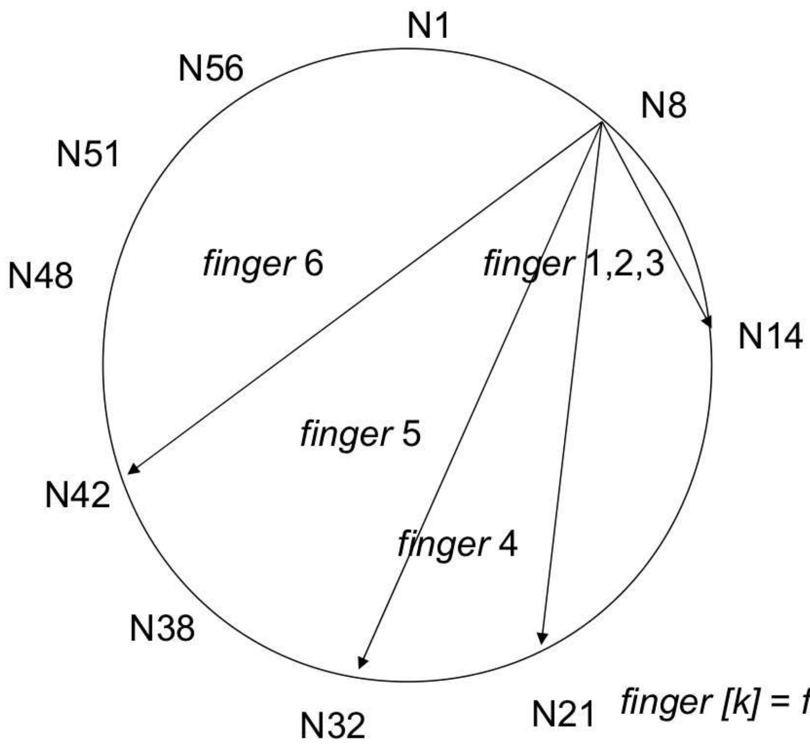
Increase efficiency through *finger table* (=routing table)



Chord Finger Table Routing

- Keep a routing table to a small number of peers
  - Enable faster transversal of the network overlay
- Chord: table of m entries for a key space of m bits
  - Entry i has the identity of the first peer that succeeds n by $2^{(i-1)}$
- Peers have more information about closer peers
- Each peer can now traverse half of the distance to the farthest node directly

Chord Finger Table



Finger Table for N8

N8+1	N14
N8+2	N14
N8+4	N14
N8+8	N21
N8+16	N32
N8+32	N42

finger [k] = first node that succeeds $(n+2^{k-1})\bmod 2^m$

Exercise

Consider a chord ring with space [0,2^5-1], and with the following set of nodes {10, 15, 21, 28, 30}.

Calculate the finger table for node 21.

Chord Stability

- Finger tables reduce the probability of loosing connectivity
  - Nodes keep info about r successors
  - If each fails with probability p, node keeps connected with probability 1-p^r instead of 1-p
- Chord connectivity relies on successor pointers
  - If successors are correct, lookup always returns correct answer
  - If finger tables are partially incorrect, lookup may take longer, but will be correct
  - If successor pointers are not up-to-date, a lookup may fail
- Protocol periodically verifies successor pointers and updates finger tables
- Join protocol guarantees that after some time all finger tables are updated

Chord Join

New node join implies

- Updating successor, predecessor and finger of new node
- Updating finger tables and predecessor of other nodes
- Update contents of new node

New node must know a node in the network

Chord Join

New node asks known node to lookup his successor, predecessor and fingers

- Successor predecessor becomes predecessor of new node
- New node becomes predecessor of successor
- Successor becomes successor of new node

New node updates finger tables of all nodes that are likely to have it in their tables

- Complexity: $O(\log N)^2$

New node contacts successor to transfer keys (=contents)

See Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01). 2001

UNSTRUCTURED P2P

19/10/18

Unstructured P2P Overlays

Loose, nearly random network topology

Peers join and leave network according to small set of loose rules

Content availability depends on node availability

No content movement on topology change

Unreliable search

Load on each peer grows with number of queries

Example: Original Gnutella Search

- Service: file sharing
- No overlay addresses
- Content queries are flooded as QUERY messages
  - TTL limits it to a number of levels
- Each node forwards QUERY to all neighbours
- If node has the content, it replies with QUERY RESPONSE to previous node
  - Specifies IP address and port for download
- QUERY RESPONSE is forwarded upwards until it reaches origin
  - Only original neighbour knows who requested content
- Actual data transfer is a regular file transfer
- Scalability
  - Guarantees location of rare contents (flooding), but no time bound
  - Very large amount of QUERY messages in the network

Example: Gnutella Maintenance

Nodes know a certain number of other nodes

- Any node needs to know one node in the network to join
- Send PING message to known nodes, which become neighbours by answering with PONG
- Any node can leave anytime
- PING and PONG used for probing neighbour presence

Fully de-centralised means no-one can be made responsible for the network

- How to bootstrap?

Example: Original Gnutella

Advantages

- Very resilient
- Low overhead for highly varying node population
- Guarantees to find contents if they are in the network
- No node carries responsibility for the network

Disadvantages

- Low search scalability
- No delay bounds on search
- No guarantee that contents stay in the network

Improve Search

How would you improve search scalability?

- Limit number of neighbours to whom QUERY is forwarded
  - Random walk, parallel random walks, highest degree nodes first
- Limit search depth
  - If no reply is returned, search deeper (expanding rings)
- Pro-active object replication:
  - Content replicas at certain locations to improve the efficiency of queries or delivery.
  - Object replication raises issues of data consistency and synchronisation
- Indices at nodes store location of nearby contents
  - Additional messages must be exchanged
  - Additional maintenance overhead (freshness)
- Add hierarchy!

Further reading: M. Ripeanu. “Peer-to-Peer Architecture Case Study: Gnutella Network”. In Proceedings of the First International Conference on Peer-to-Peer Computing (P2P). 2001

Gnutella Query Routing Protocol

- Ultrapeers and leaf nodes
  - Each leaf connects to a few ultrapeers
  - Ultrapeers connect to each other
- Leaf nodes announce contents shared to ultrapeer neighbours using hashes
- Ultrapeers
  - Append all contents they see at one hop to their own table
  - Share their table with their direct neighbours
  - Direct queries to leafs only if there is a hit on a hash
- Reduces amount of query messages in the network

http://rfc-gnutella.sourceforge.net/src/qrp.html

Using hierarchy to improve search has been proven to be advantageous in peer-to-peer networks with power law node degree, as long as the “trackers” are the most long lived nodes

Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, B. Huberman. “Search in Power Law Graphs”. PHYSICAL REVIEW E, VOLUME 64. 2001

DOI: 10.1103/PhysRevE.64.046135

Example: Bittorrent

Uses chunks to increase likelihood of keeping content available

Files split into small pieces (chunk)

- As soon as a downloader finishes downloading one piece, he becomes an uploader for that piece of the file
- Chunk is available at least while node downloads the rest of the file

Difficult to find out which peers have which parts of the file in a distributed manner without a large overhead (recall Gnutella)

- Trackers are super-nodes used to search for peers
- .torrent files have meta information about the files to transfer, including indexes of the chunks that make up a file and the responsible tracker

Bittorrent Tracker Protocol

- Peers request from trackers information about peers that have chunks of the file
  - Communication with trackers is done over HTTP
    - Alternative protocol over UDP to reduce overhead: uTP
    - uTP reduces traffic by 50%
  - Peers can repeat request periodically
- Trackers respond with IP and port of peers that have chunks of the file
- Peers inform trackers when they start, stop and finish downloading a file
- All other communication is among peers

Bittorrent Peer Protocol

- Peers contact each other using TCP
- Peer protocol is symmetric
    - Data can flow on either direction
    - Messages exchanged are the same
- Connections refer to file chunks and have two indicators
    - Interested/ not interested
    - Chocked/ unchocked: used to control upload rate of each peer
    - Data transfer takes place when one side is interested and the other is unchocked
- Peers download chunks randomly
    - Increases probability that all pieces are replicated to improve content availability
- When a peer finishes downloading a chunk, it announces it to all its connected peers

Bittorrent Fairness

- Total download rate (of all peers) must be equal to total upload rate
  - Each peer must be downloading from someone
- Fairness
  - Peers are most satisfied when their upload rate is proportional to their download rate: extremely hard to do in a distributed fashion
    - Because how much data peers download from other peers depends on the chunks that each one has
    - Peers are not always uploading to the peers they are downloading from
    - But they try to…

Bittorrent Incentives: Tit-for-tat

- Tit-for-tat: Very simple and successful strategy for autonomous agents in repetitive Prisoner's Dilemma games (Game Theory)
  1. Cooperate on the first move
  2. Repeat the previous move of the opponent on the following moves
- Bittorrent version
  - Peers reciprocate upload to peers downloading to them
    - unchoke 4 peers uploading at highest rate
    - Several connections actively transferring in both directions but without central coordination
  - Non-utilised connections are unchoked on a trial basis to probe for better transfer rates (optimistic unchoke)
    - First contact peers have higher probability of being unchoked
- B. Cohen., "Incentives Build Robustness in BitTorrent". 2003

Invited talk by António Rodrigues

A GLIMPSE AT INFORMATION CENTRIC NETWORKING

NEXT: INTERNET OF THINGS