File Sharing to Resource Sharing – Evolution of P2P Networking

Anura P. Jayasumana

Electrical & Computer Engineering,

Colorado State University,

Fort Collins, CO 80525

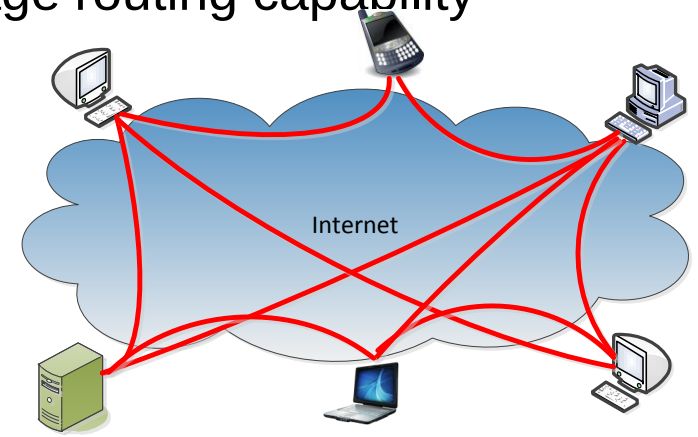
Slides by Dilum Bandara and Anura Jayasumana

Outline

- File sharing
  - Unstructured vs. structured overlays
  - Performance enhancements
    - More state, caching, replication
  - Opportunities & challenges
- Streaming
  - Tree-push vs. mesh-pull
  - Opportunities & challenges
- Resource sharing
  - Collaborative P2P
  - Resource aggregation
  - Opportunities & challenges

Peer-to-Peer (P2P) Systems

- Distributed systems without any central control
- Autonomous peers
  - Equivalent in functionality/privileges; Both a client & a server
- Protocol features
  - Protocol constructed at the application layer
  - Overlaid on top of Internet
  - Typically a peer has a unique identifier
  - Supports some type of message routing capability
- Fairness & Performance
  - Self-scaling
  - Free-rider problem
  - Peer Churn



P2P Applications

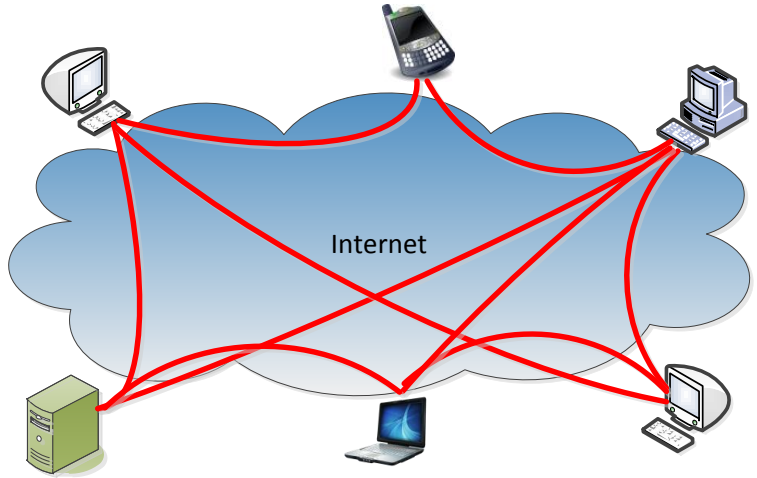
- Many application domains
  - File sharing – BitTorrent, KaZaA, Napster, BearShare
  - IPTV – PPLive, CoolStreaming, SopCast
  - VoIP – Skype
  - CPU cycle sharing – SETI, World Community Grid
  - Distributed data fusion – CASA

- Impact of P2P traffic
  - In 2008 – 50% 2009- 39% of total Internet traffic (2014-17%)
  - Today – Volume still growing
    - 3.5 Exabytes/month (4 in 2014)
  - globally, P2P TV is now over 280 petabytes per month
  - P2P traffic 20 percent of all mobile data traffic globally

[Hyperconnectivity and the Approaching Zettabyte Era, Cisco 2010]

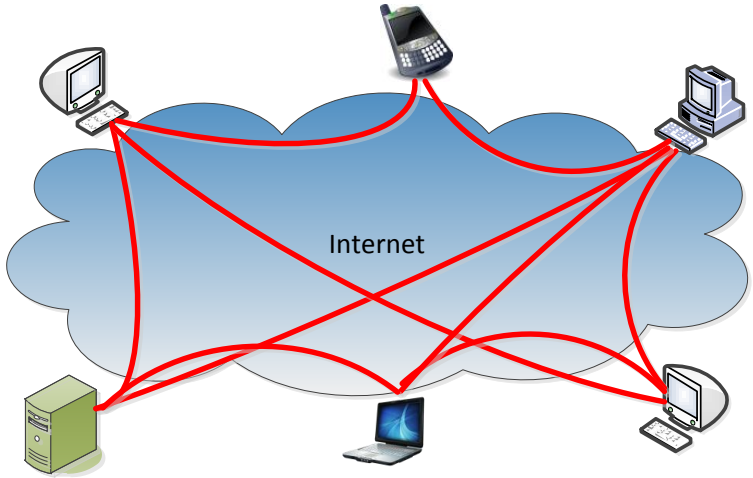
P2P Characteristics

- Tremendous scalability
  - Millions of peers
  - Globally distributed
- Bandwidth intensive
  - Upload/download
  - Many concurrent connections
    - Aggressive/unfair bandwidth utilization
    - Aggregated downloads to overcome asymmetric upload bandwidth
- Heterogeneous
  - Superpeers
  - Critical for performance/functionality



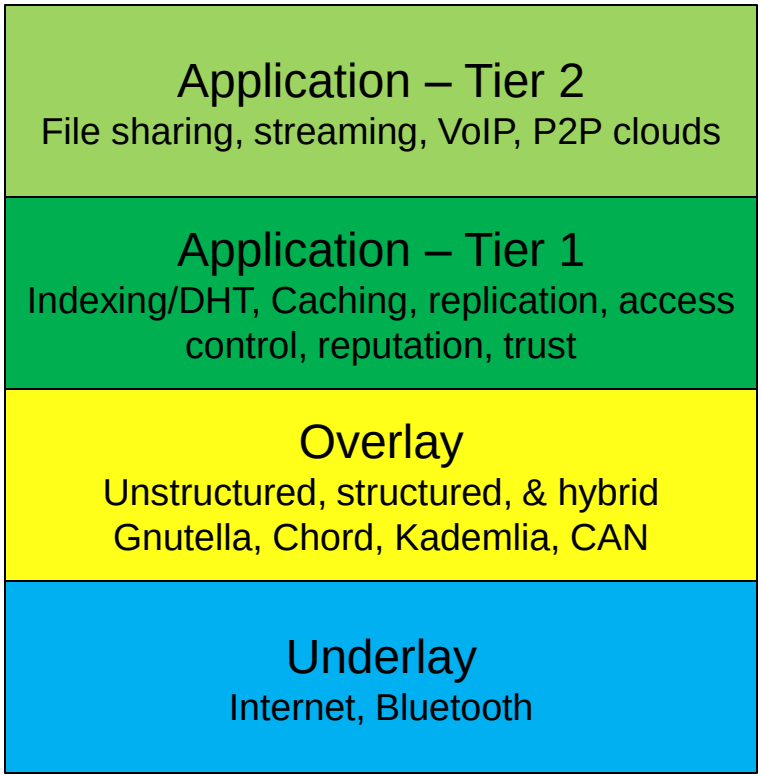
P2P Overlay

- Peers directly talk to each other, or if they are not directly connected, uses overlay routing mechanism via other peers
  - Best effort service on Internet
- Peers are autonomous
  - Determines its own capabilities based on its resources (minimum threshold of resources)
  - Decides on its own when to join, leave
- Peers have symmetrical roles (relaxed in cases such as superpeer)
- Overlay is scalable and resilient
  - In size, geography
  - Graceful degradation, ensure connectedness when nodes leave, etc.
- Overlay Maintenance
  - Overlay has to be self-organizing (overlay management is done in a distributed manner)

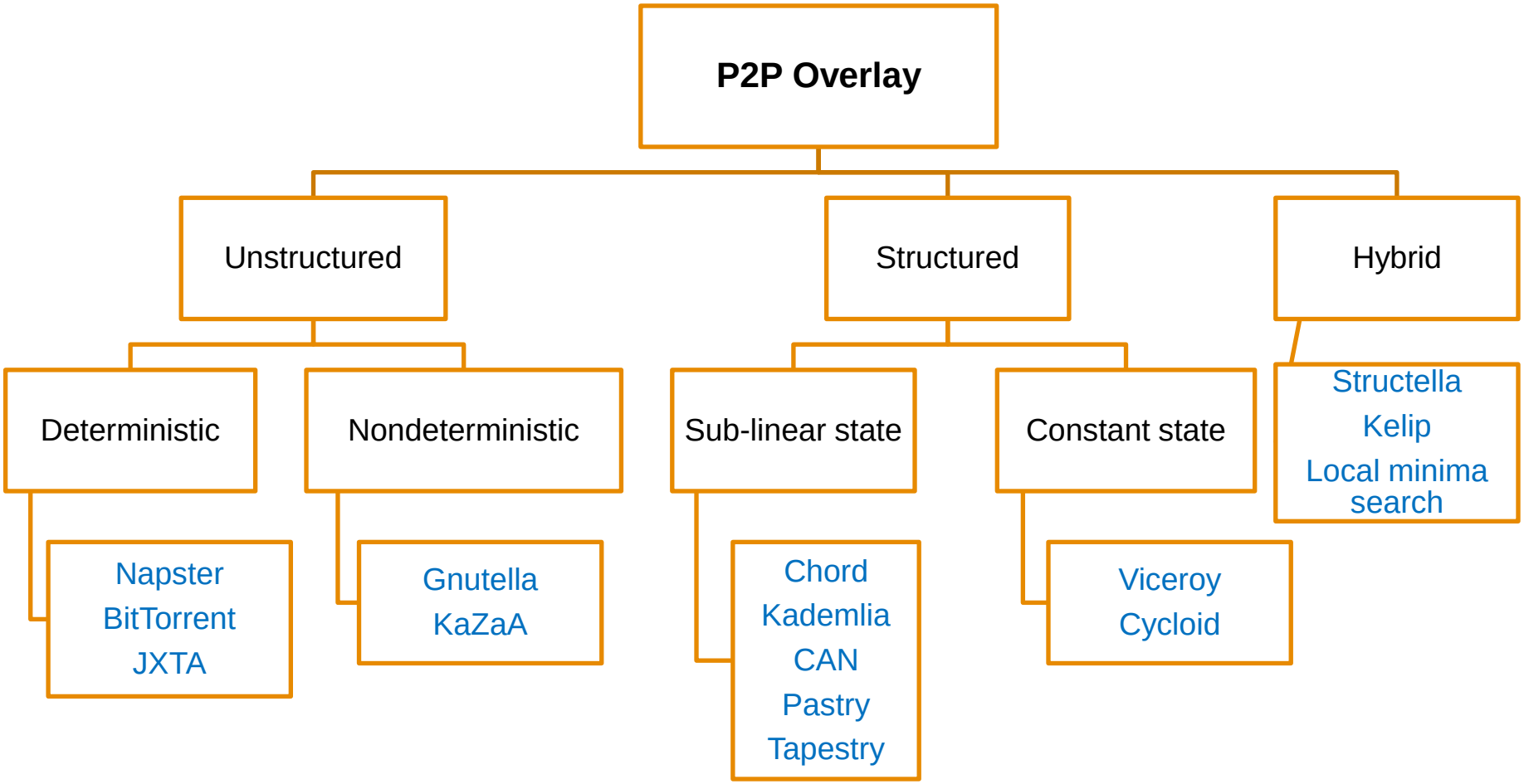


Terminology

- Application
  - Tier 2 – Services provided to end users
  - Tier 1 – Middleware services
- Overlay
  - How peers are connected
  - Application layer network consists of peers
  - E.g., dial-up on top of telephone network, BGP, PlanetLab, CDNs
- Underlay
  - Internet, Bluetooth
- Peers implement top 3 layers



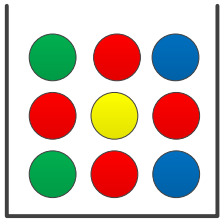
Overlay Connectivity



Bootstrapping

- How is an initial P2P overlay is formed from a set of nodes?
  - Use a well known server to register initial set of peers
  - Some peer addresses are well known
  - Use a well known multicast group address for peers to join
  - A well known domain name
  - Use a local broadcast to collect nearby peers, and merge such sets to larger sets
- Each peer maintains a random subset of peers
  - e.g., peers in Skype maintain a cache of superpeers
- An incoming peer talks to one of the known peers
- A known peer accepting an incoming peer
  - Keeps track of the incoming peer
  - May redirect the incoming peer to another peer
  - Give a random set of peers to contact
- Discover more peers by random walk or gossiping within overlay

Resource Discovery Overview

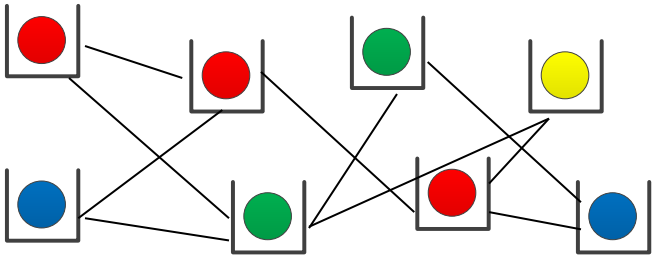


Centralized

$O(1)$

Fast lookup

Single point of failure

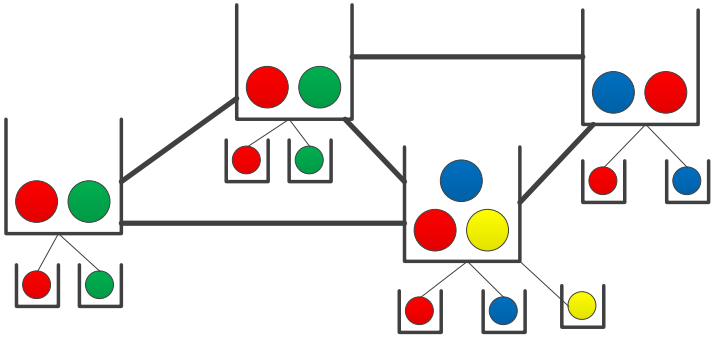


Unstructured

$O(hops_{max})$

Easy network maintenance

Not guaranteed to find resources

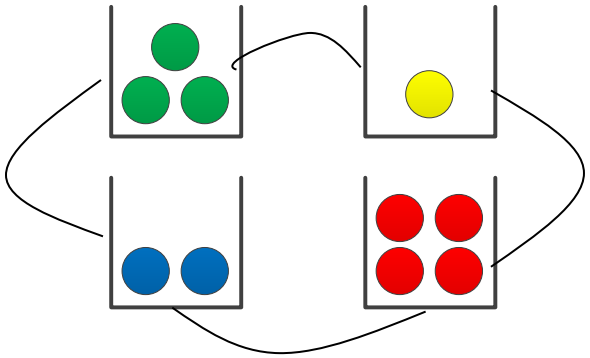


Superpeer

$O(hops_{max})$

Better scalability

Not guaranteed to find resources



Distribute Hash Table (DHT)

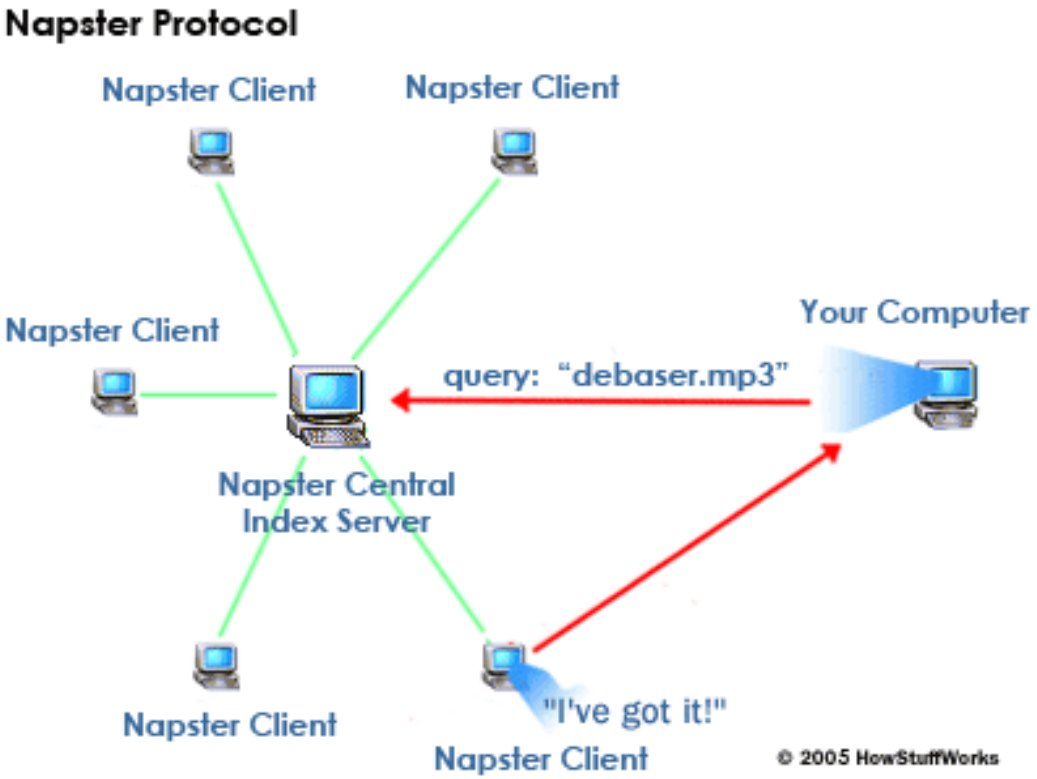
$O(\log N)$

Guaranteed performance

Not for dynamic systems

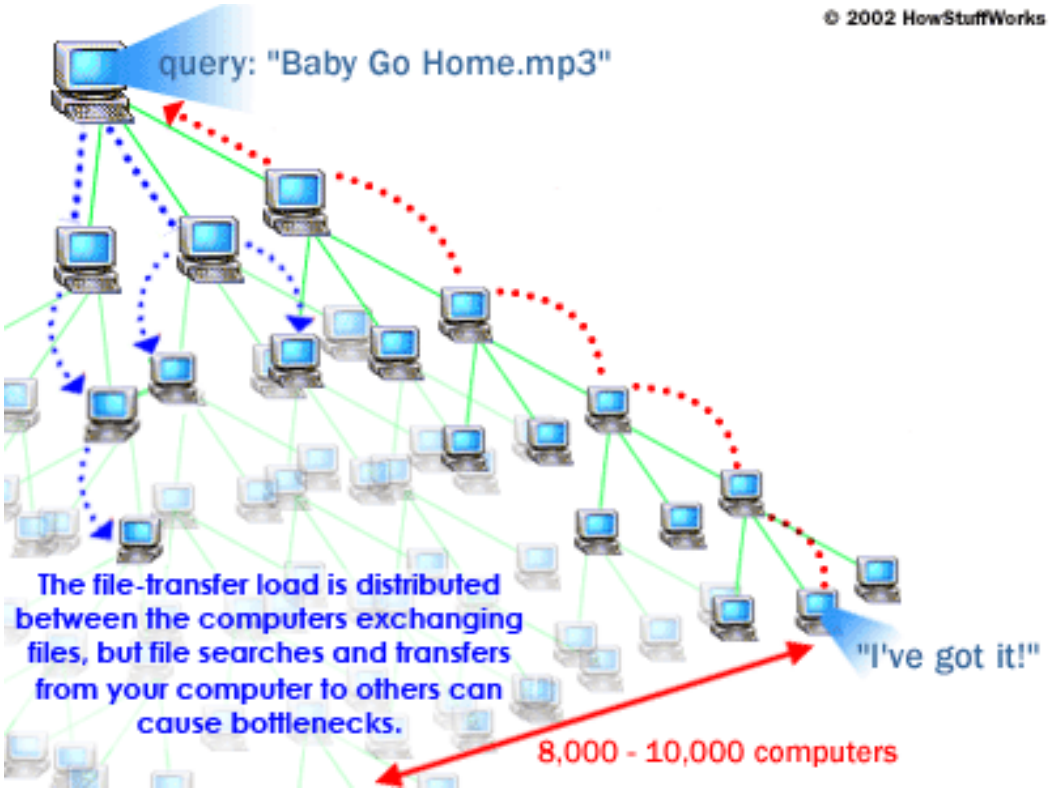
Centralized – Napster

- Centralized database for lookup
  - Guaranteed content discovery
  - Low overhead
  - Single point of failure
  - Easy to track
    - Legal issues
- File transfer directly between peers
- Killer P2P application
  - June 1999 – July 2001
  - 26.4 million users (peak)



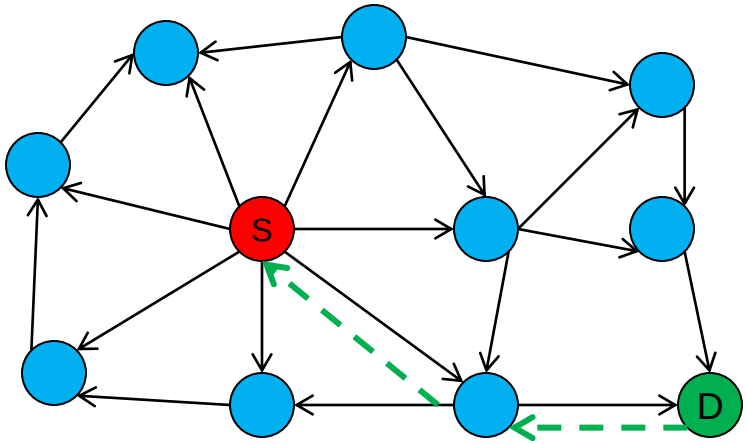
Unstructured – Gnutella

- Fully distributed
- Random connections
  - Initial entry point is known
  - Peers maintain dynamic list of neighbors
  - Connections to multiple peers
- Highly resilient to node failures

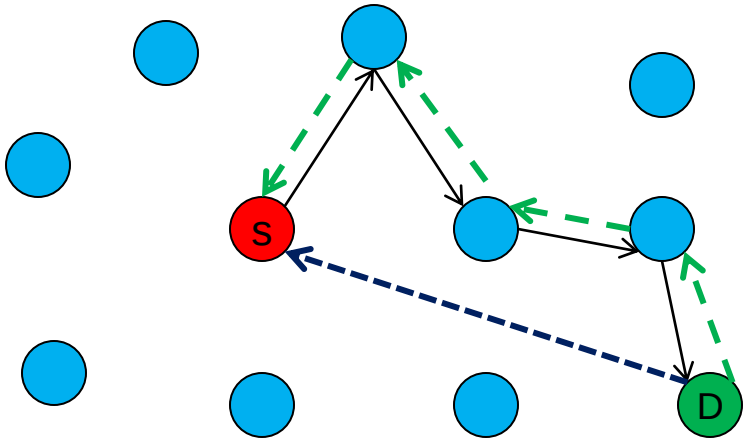


Unstructured P2P (cont.)

- Flooding-based lookup
  - Guaranteed content discovery
  - Implosion → High overhead
  - Expanding ring flooding
- TTL-based random walk
  - Content discovery is not guaranteed
  - Better performance by biasing random walk toward nodes with higher degree
- If response follow same path → Anonymity
- Used in KaZaA, BearShare, LimeWire, McAfee



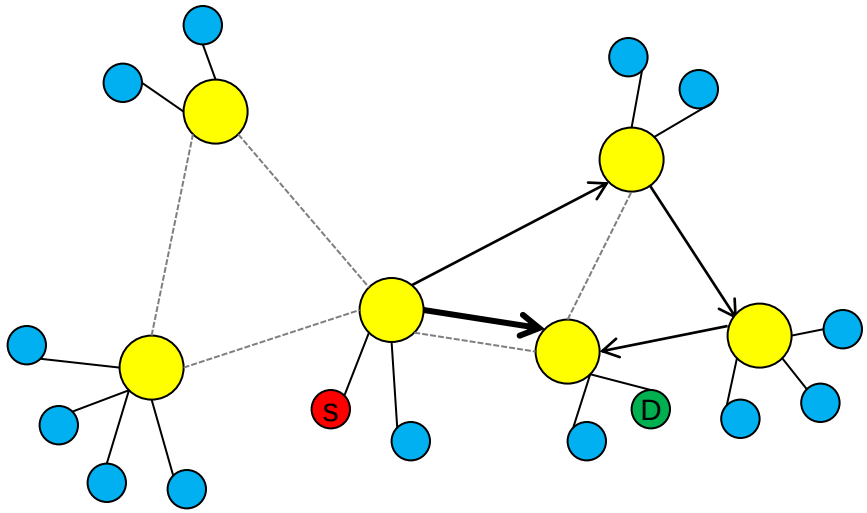
Flooding



Random walk

Superpeers

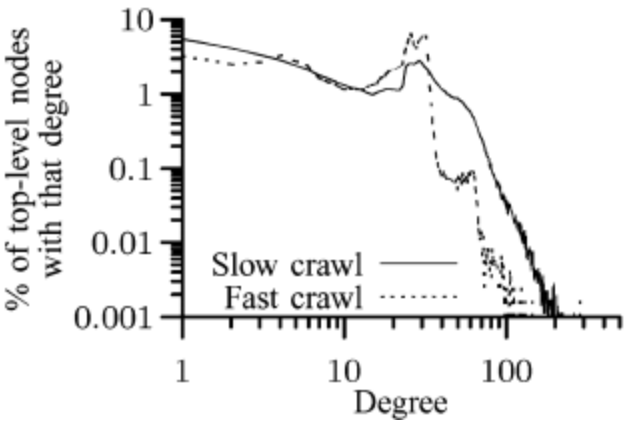
- Resource rich peers → Superpeers
  - Bandwidth, reliability, trust, memory, CPU, etc.
- Flooding or random walk
  - Only superpeers are involved
  - Lower overhead
  - More scalable
  - Content discovery is not guaranteed
- Better performance when superpeers share list of file names

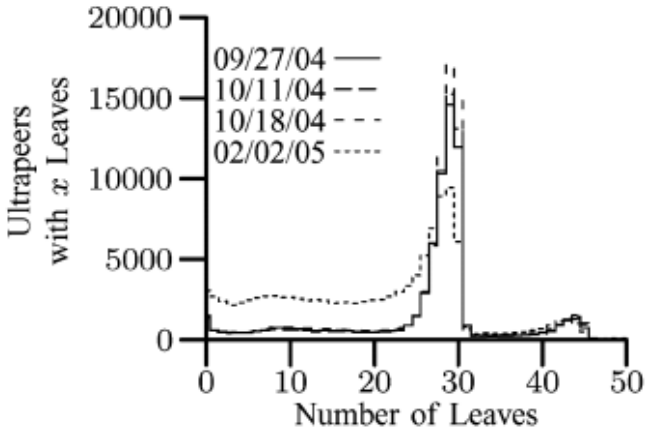


- Examples: Gnutella V0.6, FastTrack, Freenet KaZaA, Skype

Scale-Free Overlays

- Unstructured overlays can lead to scale-free networks
  - New nodes connect to
    - Existing nodes
    - Higher degree nodes
- Specific implementations may set limits on node degree
  - e.g., LimeWire maintains 27-32 connections
  - User modified code increases connectivity
- Can be used to enhance P2P lookup
  - Index at high degree nodes
  - Biased random walk towards to high-degree nodes

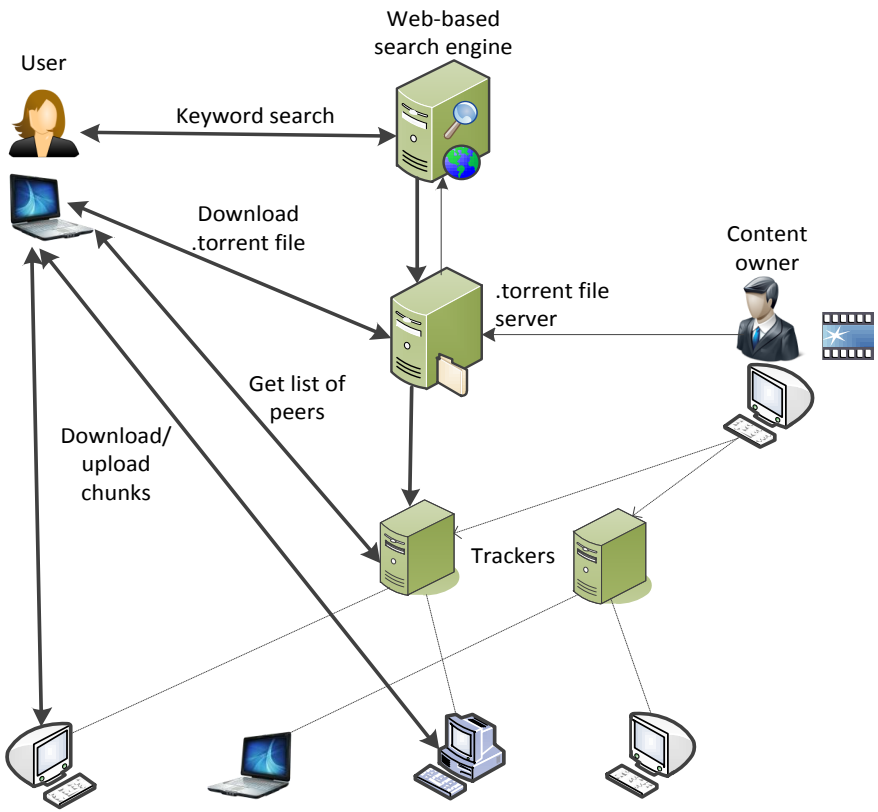




[Stutzbach, 2008]

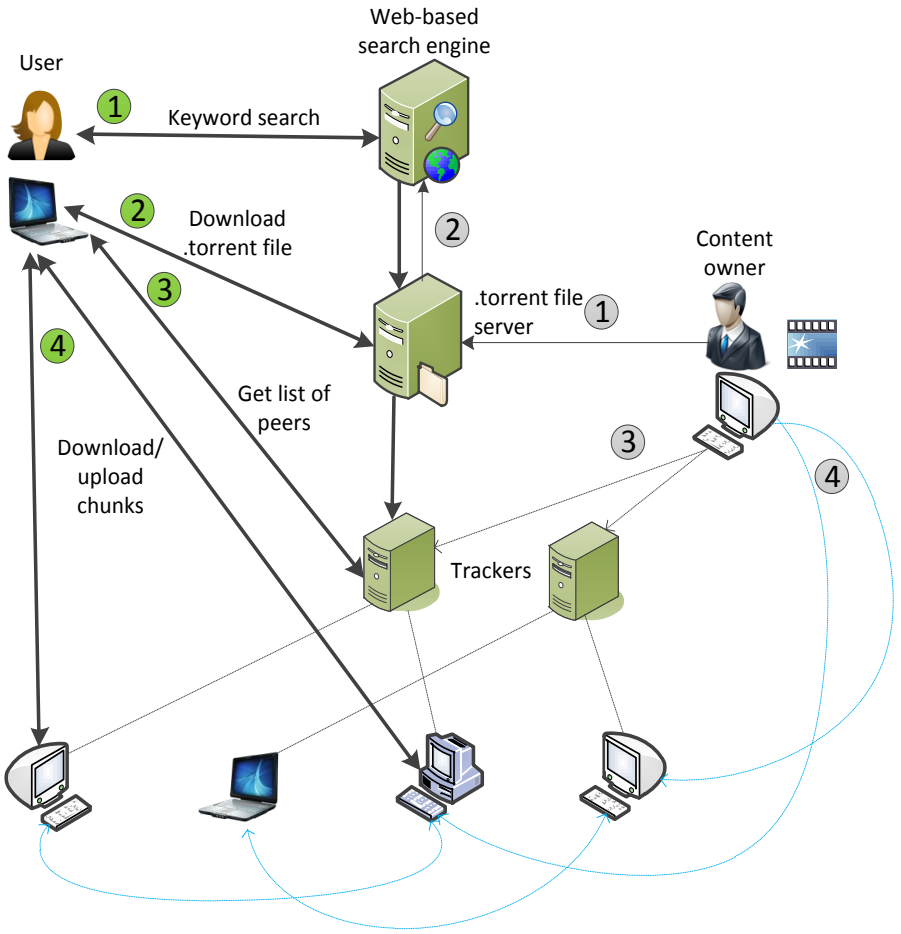
BitTorrent

- Most popular P2P file sharing system to date
- Features
  - Centralized search
  - Multiple downloads
  - Enforce fairness
  - Rarest-first dissemination
- Incentives
  - Better contribution → Better download speeds (not always)
  - Enable content delivery networks
  - Revenue through ads on search engines



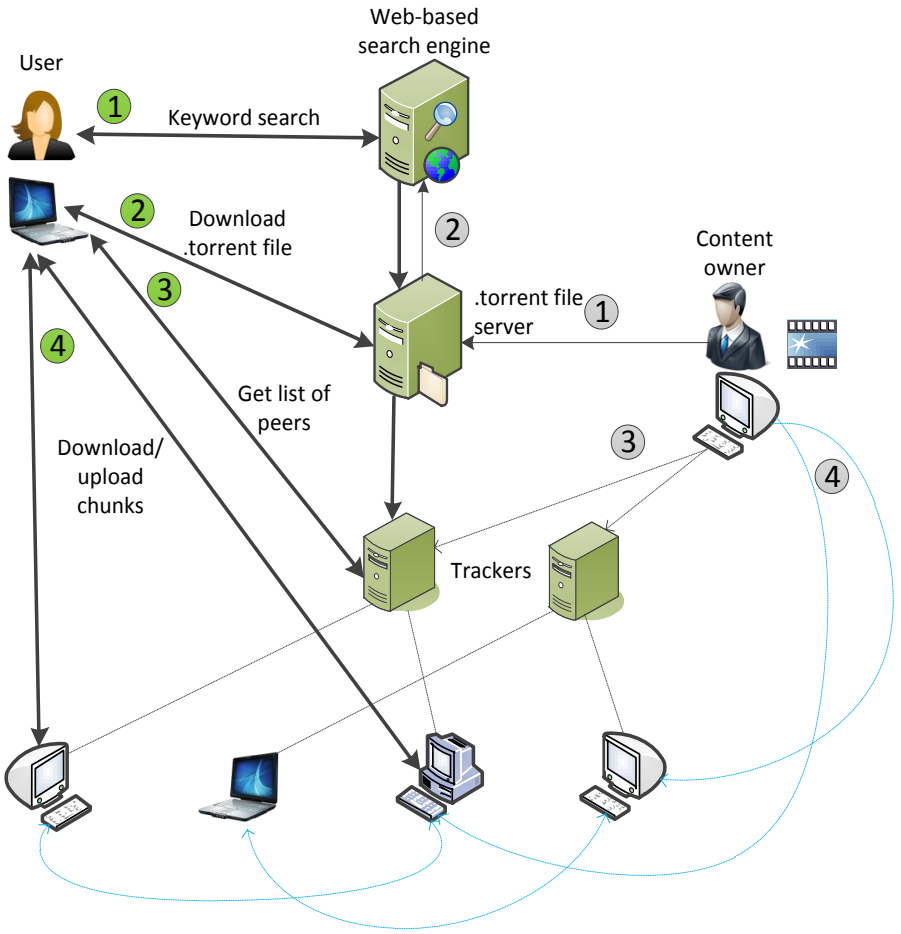
BitTorrent Protocol

- Content owner creates a *.torrent* file
  - File name, length, hash, list of trackers
- Place *.torrent* file on a server
- Publish URL of *.torrent* file to a web site
  - Torrent search engine
- *.torrent* file points to a tracker(s)
  - Registry of leaches & seeds for a given file



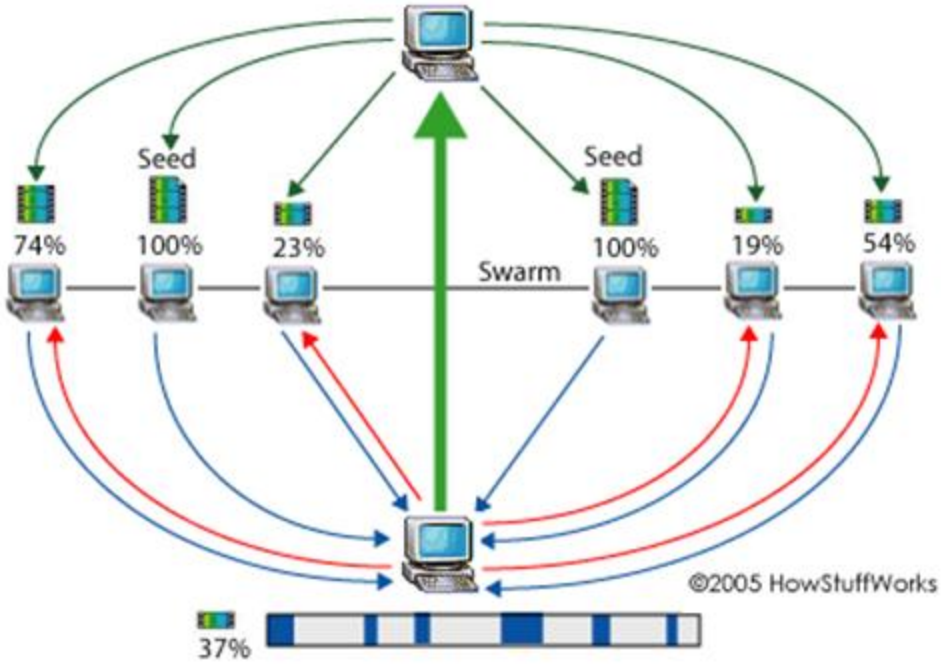
BitTorrent Protocol (cont.)

- Tracker
  - Provide a random subset of peers sharing same file
- Peer contacts subset of peers parallelly
- Files are shared based on chunk IDs
  - Chunk – segment of file
- Periodically ask tracker for new set of IPs
  - Every 15 min
- Pick peers with highest upload rate



BitTorrent Terminology

- Swarm
  - Set of peers accessing (upload/download) same file
- Seeds
  - Peers with entire file
- Leeches
  - Peers with part of file or no file (want to download)



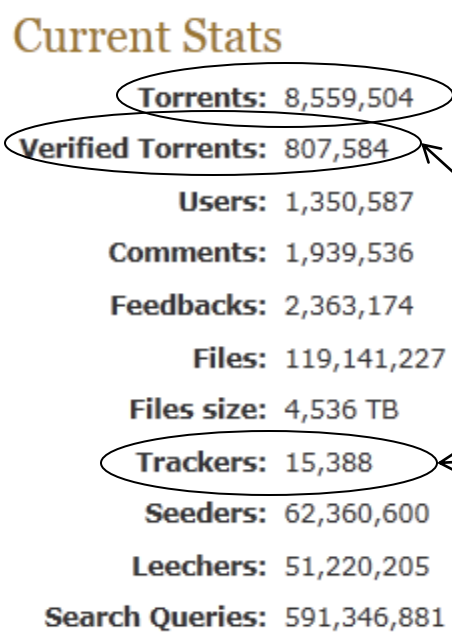
TORRENT NAME	SIZE	FILES	AGE	SEED	LEECH
The Hangover Part II 2011 TS XViD-EP1C	1.38 GB	5	1 day	14898	14595
Dawning 2011 DVDSCR 390MB XviD-Bello0076	392.07 MB	2	2 days	88	32
Dawning 2011 DVDSCR XviD AC3 SiC	1.39 GB	3	2 days	735	1108
Listen to Your Heart 2011 DVDRip AC3 DMT	701.24 MB	12	2 days	192	243
Blue Crush 2[2011]BRRip XviD-ExtraTorrentRG	702.27 MB	5	2 days	1567	2030
CLEANER 2007 * 1CD * BRRip XviD AAC * M777 M2Tv	706.08 MB	1	2 days	33	40
Dawning 2011 * 1CD * DvDScr XviD AC3 * M777 M2Tv	712.96 MB	1	2 days	36	19
Beastly 2011 DVDRip XviD AC3 BeFRee	713.23 MB	3	2 days	4024	3166

www.kat.ph

BitTorrent Site Stat

User ranking of file quality

Seedpeer.com

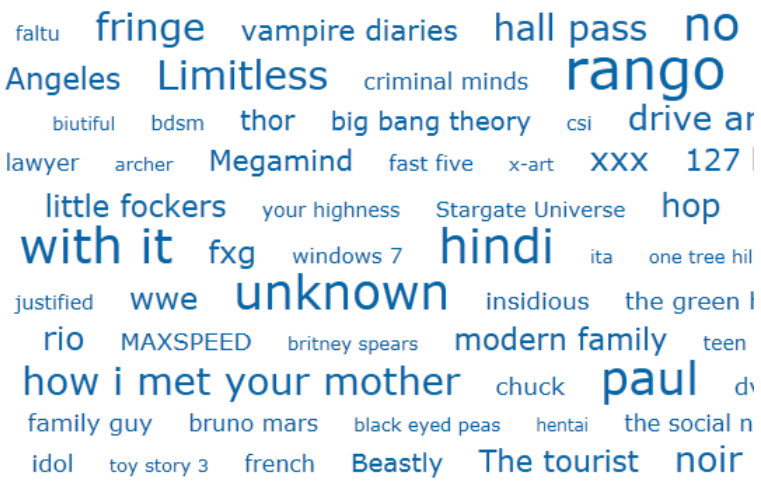


Files in search engine

User verified to be valid

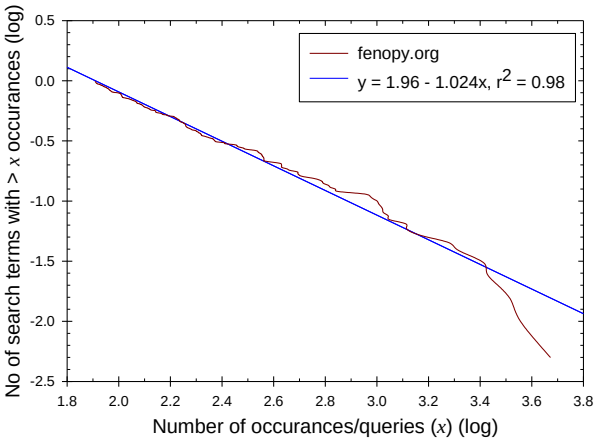
Across all files

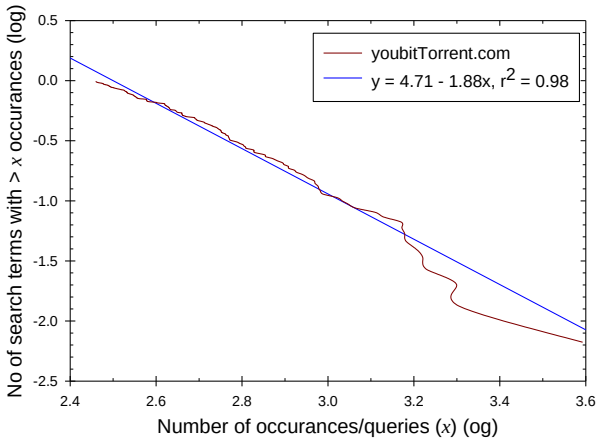
www.kat.ph/stats/

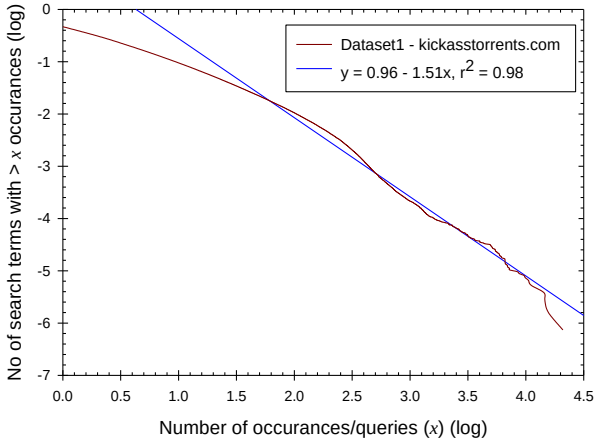


Search Cloud

BitTorrent Content Popularity

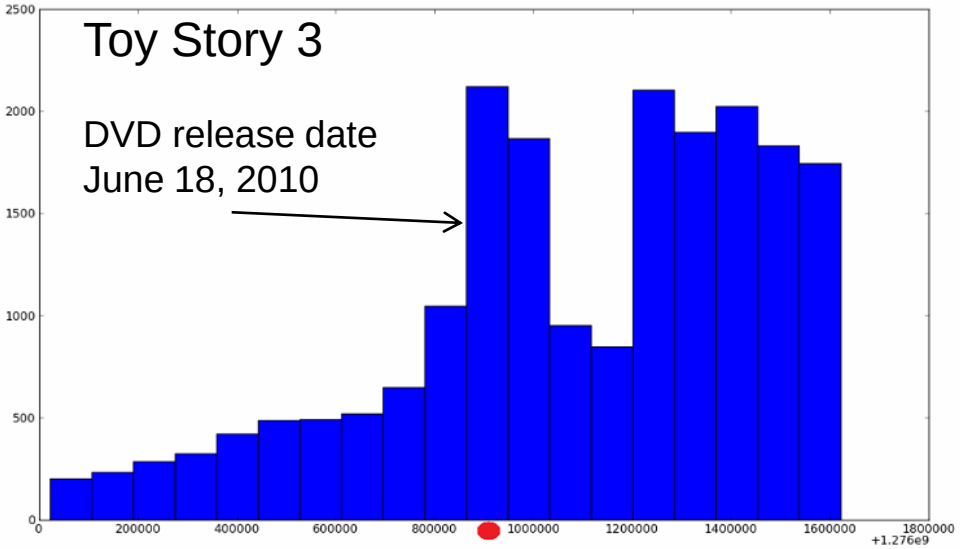




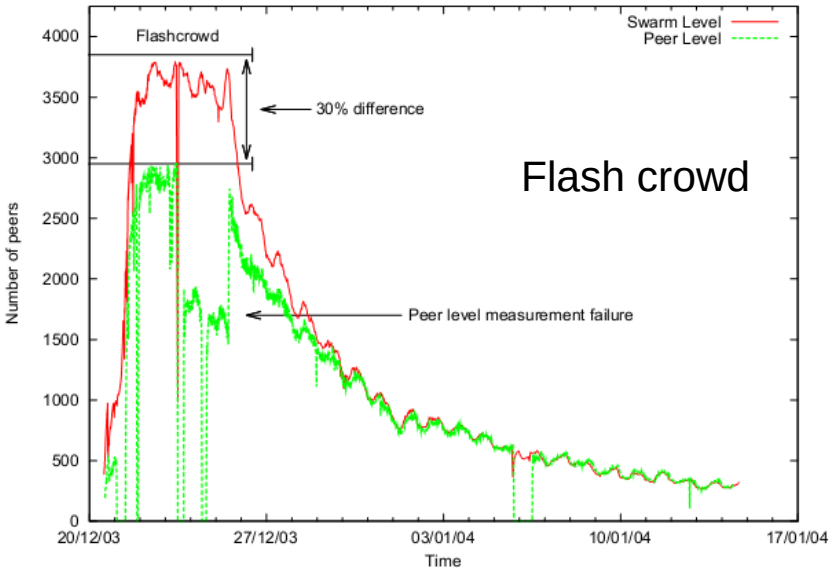


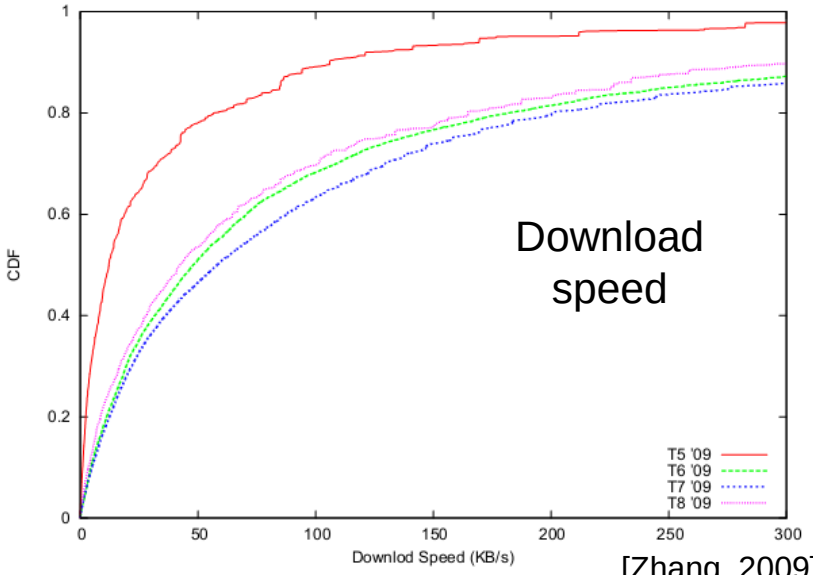
[Bandara, 2012b]

- Few highly popular content
- Moderately popular content follow Zipf's-like distribution
- Typical Zipf's parameter 0.5-1.0



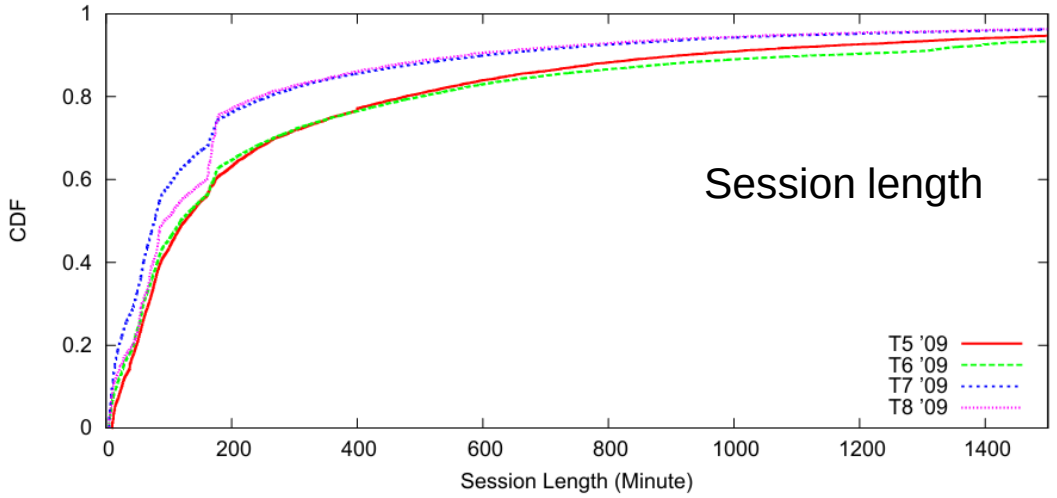
BitTorrent Characteristics



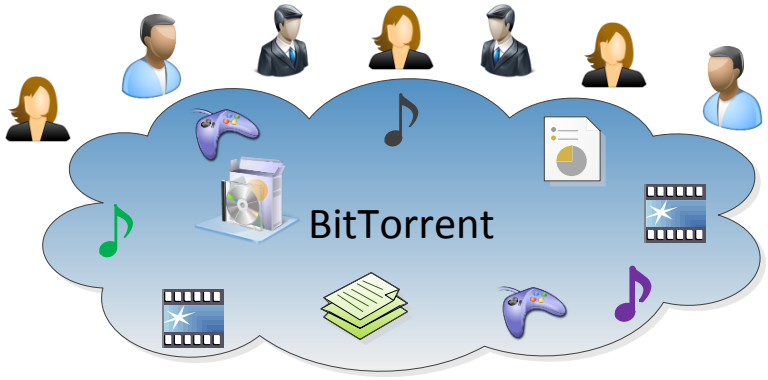


[Zhang, 2009]

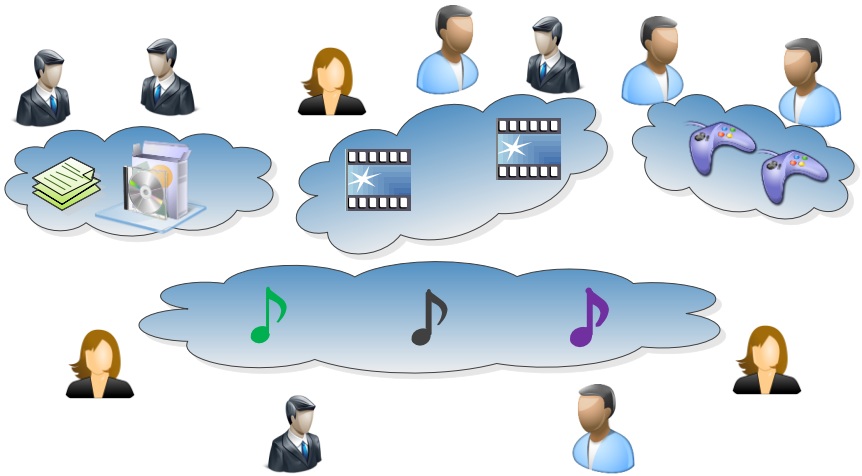
- Flash crowd effect
- Asymmetric bandwidth
- Most peers leave after downloading
- Diurnal & seasonal patterns



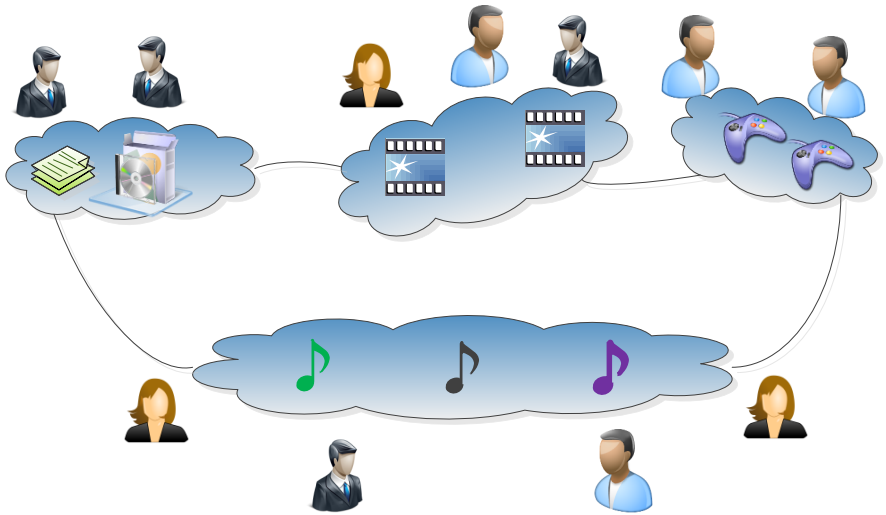
BitTorrent Evolution



Global community



Islands of communities



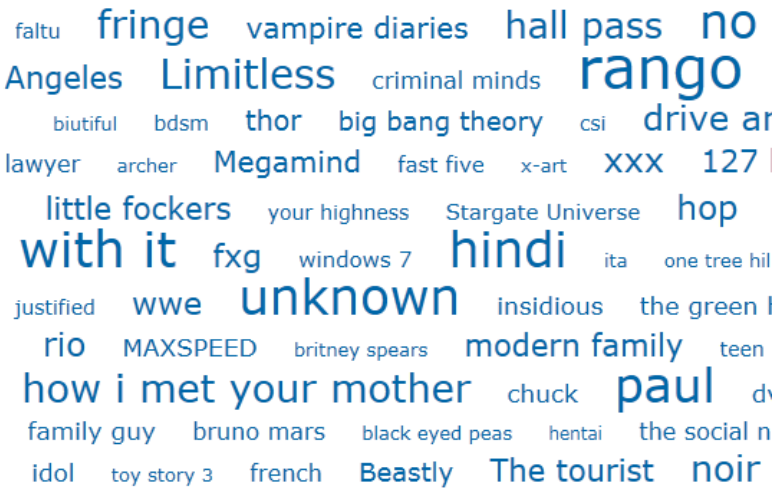
Connected islands of communities
BitTorrent ver. 4.2

BitTorrent Communities

- Many communities emerged based on similarity
  - Semantic – songs, video, games, Linux distributions
  - Geographic – China, India
  - Organizational – private communities
  - Run their own trackers
- Many islands of deployments
  - Not isolated
  - Have to search in many trackers
  - v4.2.0 – connect peers using a Distributed Hash Table (DHT)
- Many private communities
  - Require invitation to join
  - Require login
- Today, BitTorrent → Hierarchical + DHT

BitTorrent Communities (cont.)

- Similarity among communities



Community	EX	FE	SP	TB	TS	TE	TR
FE	0.38						
SP	0.00	0.00					
TB	0.40	0.29	0.00				
TS	0.48	0.33	0.00	0.48			
TE	0.53	0.23	0.00	0.31	0.25		
TR	0.10	0.08	0.00	0.06	0.09	0.06	
YB	0.36	0.35	0.00	0.29	0.42	0.20	0.04

* EX – extratorrent.com, FE – fenopy.com, SP – seedpeer.com, TB – torrentbit.net, TS – torrentscan.com, TE – torrentsection.com, TR – torrentreactor.net, YB – youbittorrent.com. Date – 24/07/2010 ~04:55 UTC.

[Bandara, 2011a]

BitTorrent Fairness/Incentives

- Tit-for-tat
- Bandwidth policy
  - Upload to 4 peers that give me the highest download bandwidth
  - 1 random peer
  - Create clusters of similar bandwidth peers [Legout, 2007]
- Chunk policy
  - Rarest first
  - Download least popular chunk
  - Initial seed try not to send same chunk twice
- Most peers leave immediately after downloading
- Modified nodes increase free riding
  - Modified policies
- Message Types
  - Choke/Unchoke
  - Interested/Not_interested
  - Have
  - Bitfield
  - Request
  - Piece
  - Cancel

Summary – Unstructured P2P

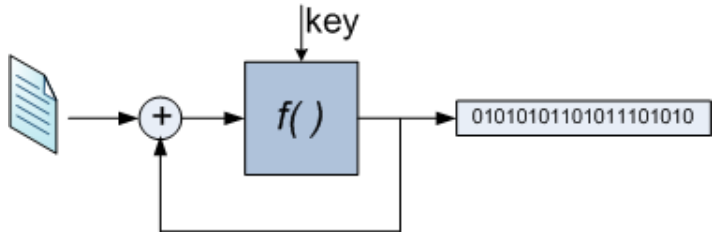
- Separate content discovery & delivery
  - Content discovery is mostly outside of P2P overlay
- Centralized solutions
  - Not scalable
  - Affect content delivery when failed
- Distributed solutions
  - High overhead
  - May not locate the content
- No predictable performance
  - Delay or message bounds
  - Lack of QoS or QoE

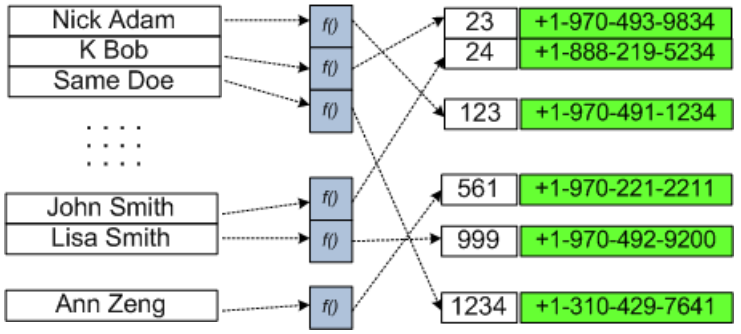
Structured P2P

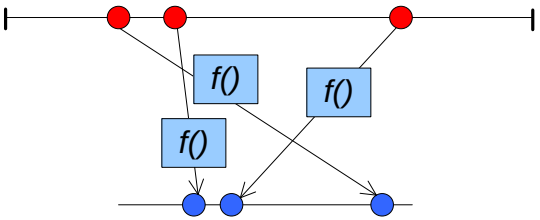
- Deterministic approach to locate contents & peers
- Locate peer(s) responsible for a given key
- Contents
  - Unique key
  - Hash of file name, metadata, or actual content
  - 160-bit or higher
- Peers also have a key
  - Random bit string or IP address
- Index keys on a Distributed Hash Table (DHT)
  - Distributed address space $[0, 2^m - 1]$
- Deterministic overlay to publish & locate content
  - Bounded performance under standard conditions

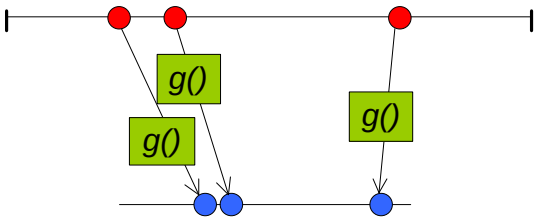
Terminology

- Hash function
  - Converts a large amount of data into a small datum
- Hash table
  - Data structure that uses hashing to index content
- Distributed Hash Table (DHT)
  - A hash table that is distributed
- Types of hashing
  - Consistent or random
  - Locality preserving



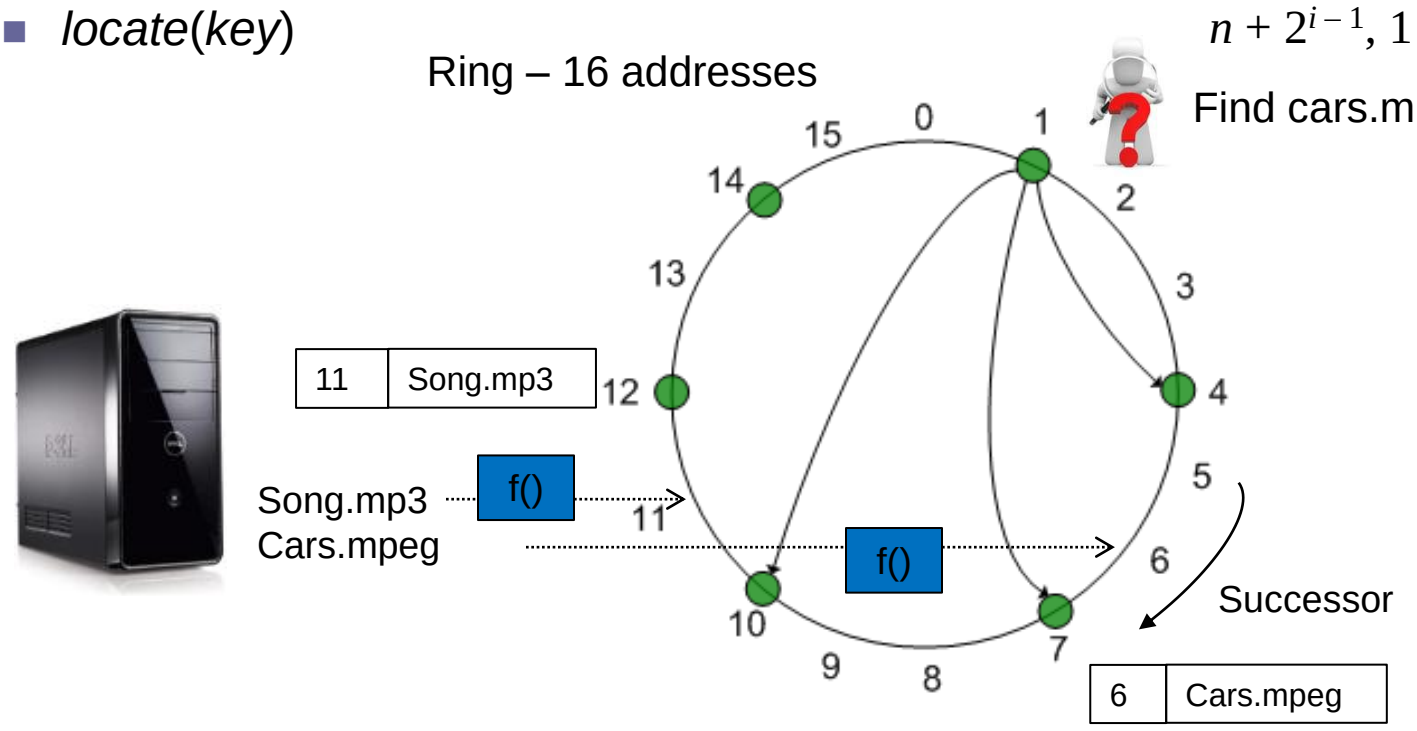






Structured P2P – Example

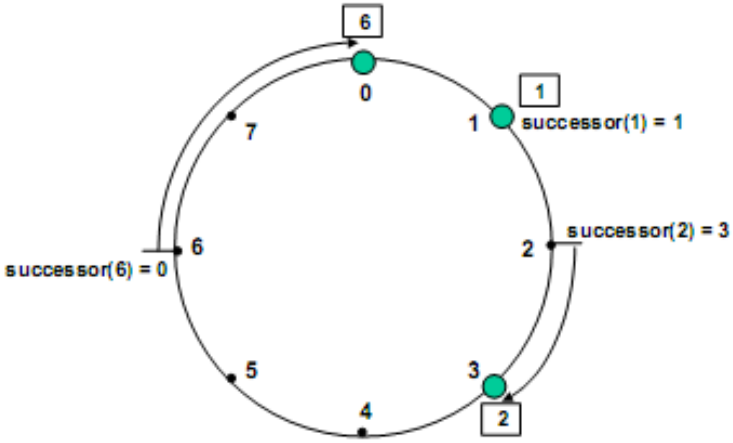
- 2 operations
  - *store*(*key*, *value*)
  - *locate*(*key*)



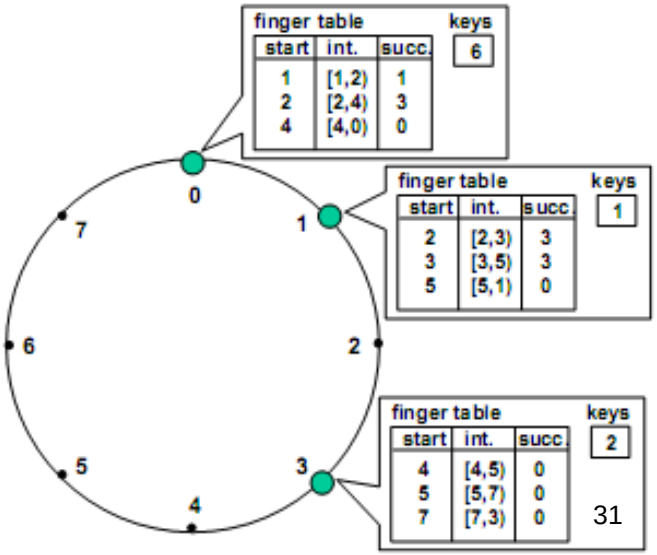
$O(log\ N)$ hops

Chord [Stoica, 2001]

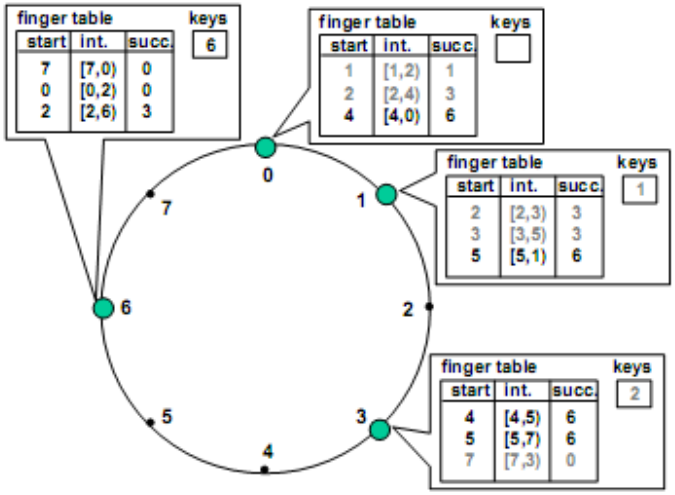
- Key space arranged as a ring
- Peers responsible for segment of the ring
  - Called successor of a key
  - 1st peer in clockwise direction
- Routing table
  - Keep a pointer (finger) to m peers
  - Keep a finger to (2^{i-1})-th peer, $1 \leq i \leq m$
- Key resolution
  - Go to peer with the closest key
  - Recursively continue until key is find
  - Can be located within $O(\log n)$ hops



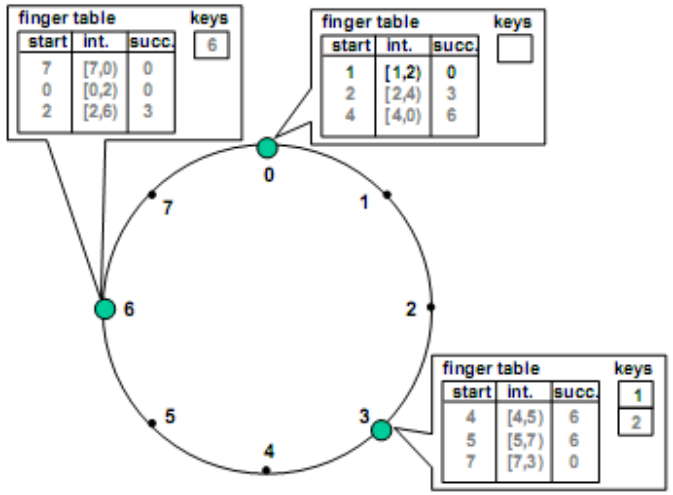
m =3-bit key ring



Chord (cont.)



New peer with key 6 joins the overlay

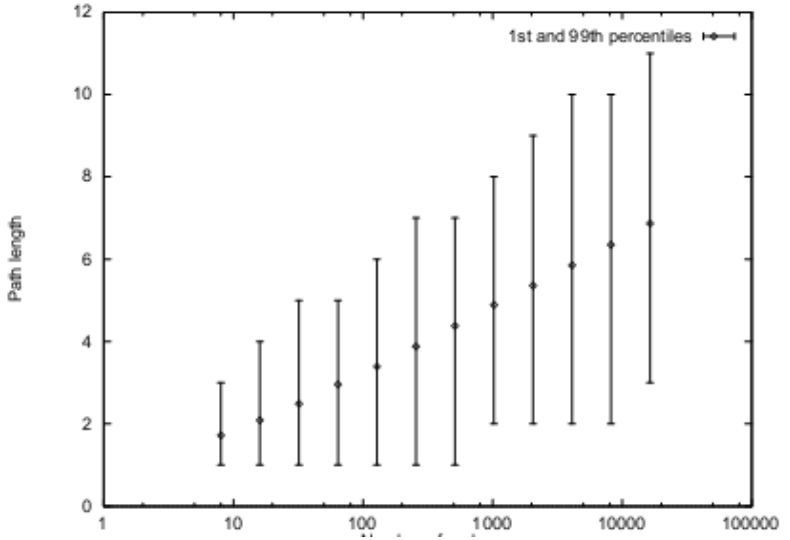


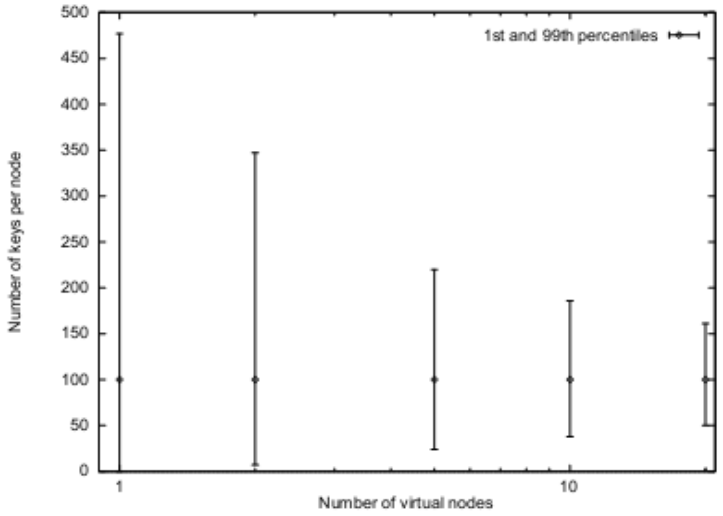
Peer with key 1 leave the overlay

- New peer entering overlay
  - Takes keys from the successor
- Peer leaving overlay
  - Give keys to the successor
- Fingers are updated as peers join & leave
- Peer failure or churn makes finger table entries stale

Chord Performance

- Path length
  - Worst case $O(\log N)$
  - Average $\frac{1}{2}\log_2 N$
- Updates $O(\log^2 N)$
- Fingers $O(\log N)$
- Alternative paths $(\log N)$!
- Balanced distribution of keys
  - Under uniform distribution
  - $N(\log N)$ virtual nodes provides best load distribution



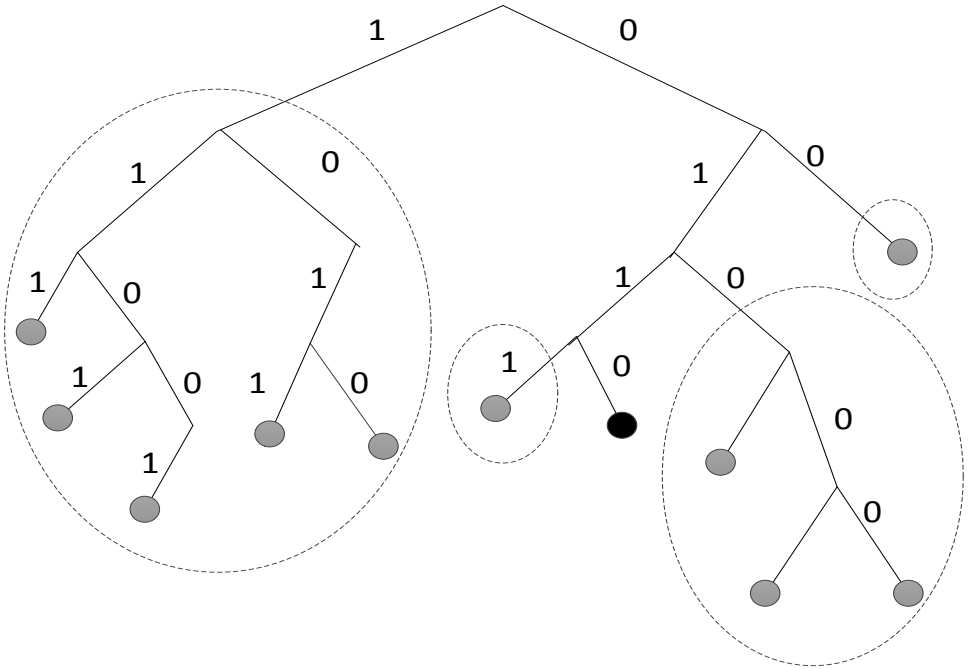


Kademlia [Maymounkov, 2002]

- Used in BitTorrent, eMule, aMule, & AZUREUS
- 160-bit keys
  - Nodes are assigned random keys
- Distance between 2 keys is determined by XOR
  - Routing in the ring is bidirectional
  - $dist(a \rightarrow b) = dist(b \rightarrow a)$
  - Enable nodes to learn about new nodes from received messages
- Keys are stored in nodes with the shortest XOR distance

Kademlia (cont.)

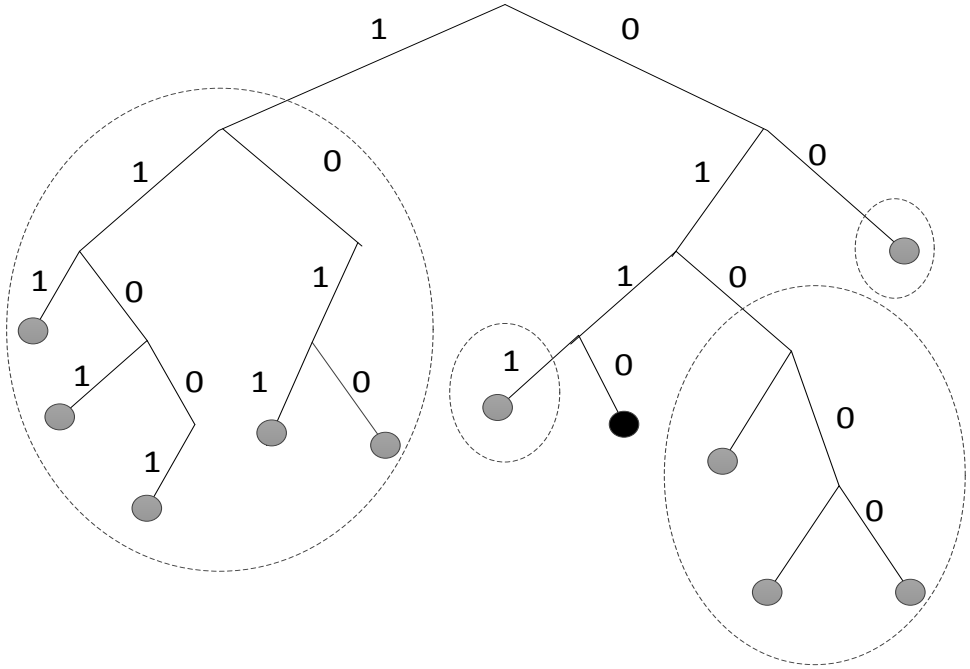
- *k*-bucket routing table
  - Store up to k peers for each $(2^i, 2^{i+1})$ distance, $1 \leq i \leq m$
  - Update bucket entries based on least-recently seen approach
  - Ping a node before dropping from a bucket
  - Better performance under peer churn & failure



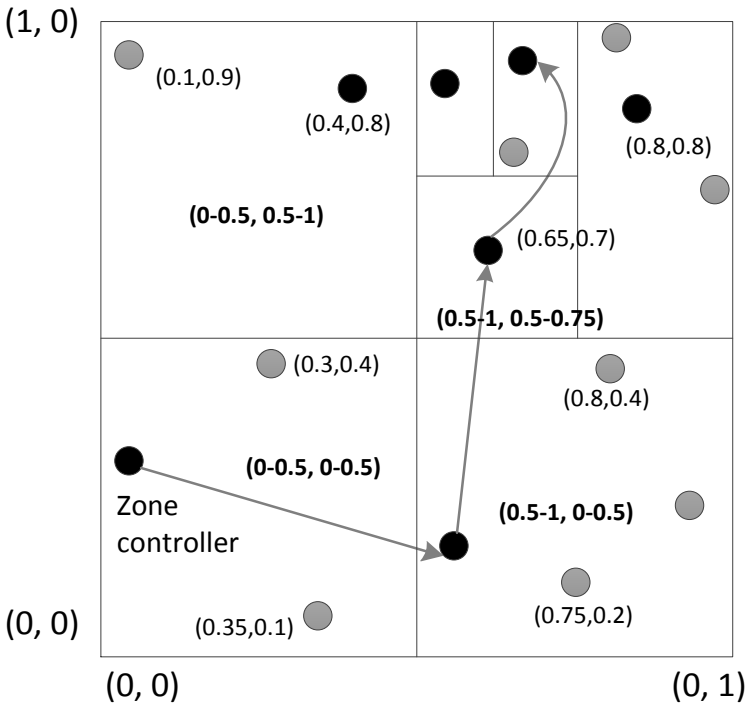
Node with key 0110 keeps k entries for
- 1xxx/1
- 00xx/2
- 010x/3
- 0111/4

Kademlia Routing

- Find set of peers with the shortest distance in routing table
  - Longest prefix match
- Concurrently, ask α of them to find an even closer peer
- Iterate until no closer peers can be found
- Then send the query to α closest peers

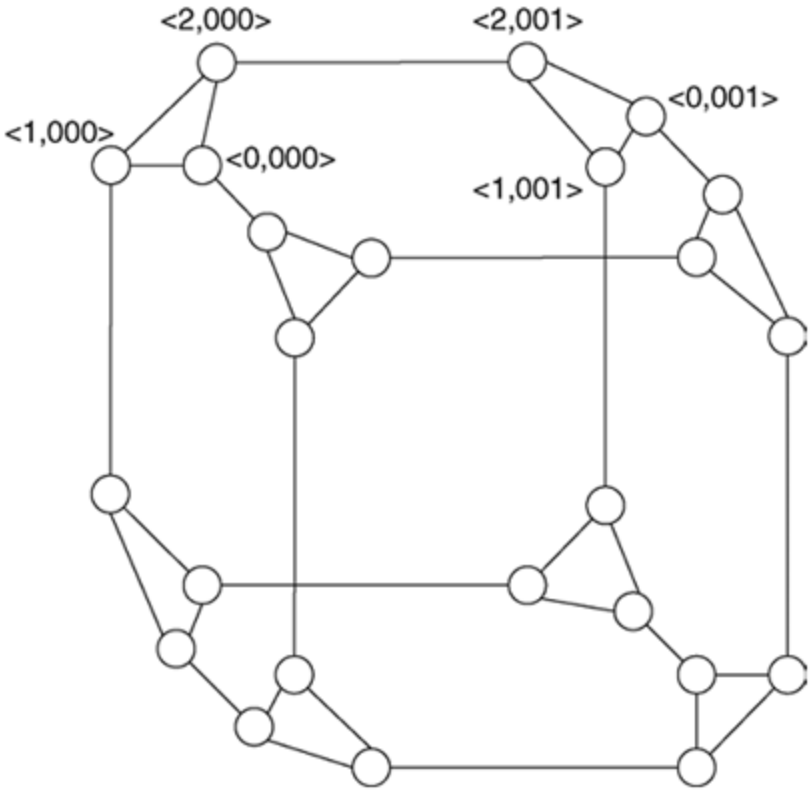


Structured P2P – Alternate Designs



d-Torus

Content-Addressable Network (CAN)
[Ratnasamy, 2001]

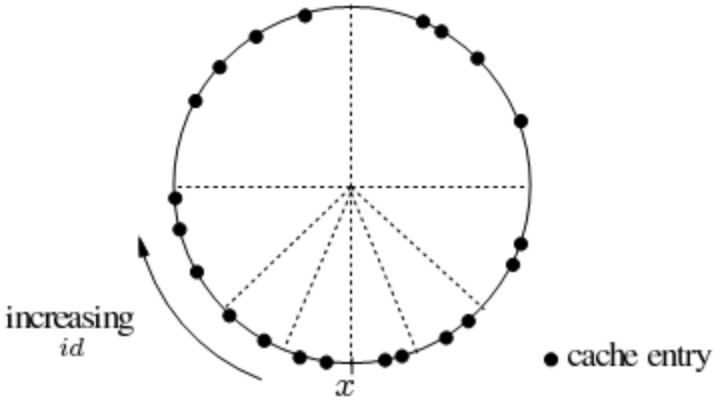


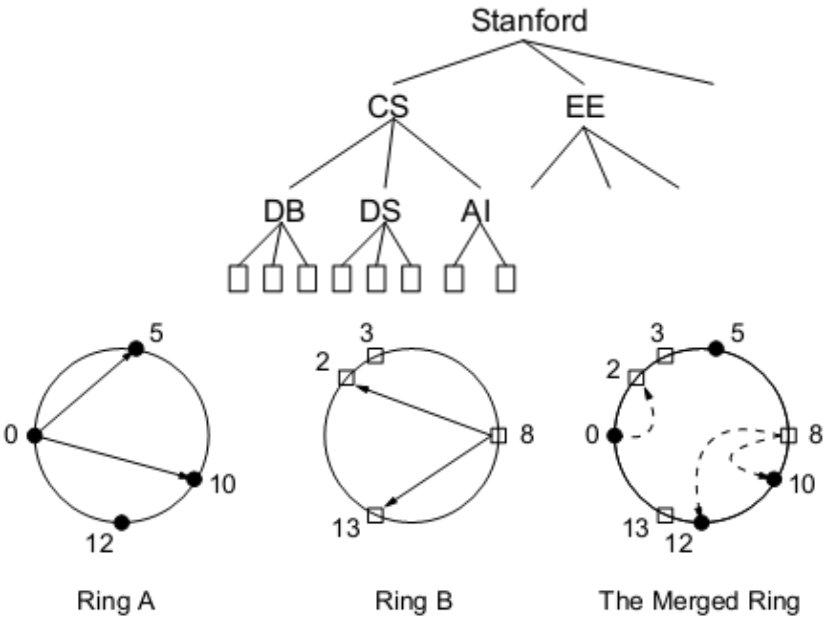
Cube connected cycle

Cycloid [Shen, 2006]

Structured P2P – Extensions

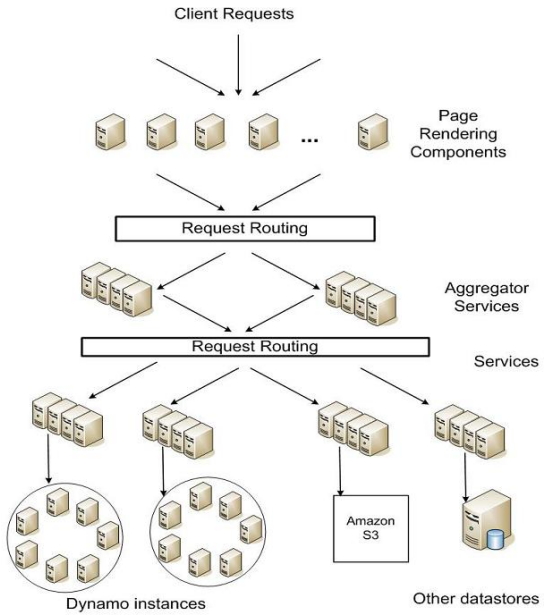
- EpiChord [Leong, 2004]
  - Use messages being forwarded to learn about new nodes
  - Cache their contact information
  - Can achieve $O(1)$ lookup
- Cannon [Ganesan, 2004b]
  - Hierarchical DHT
  - Each level in hierarchy maintains a ring
  - Merge rings at higher levels
  - Maintain original fingers as it is
  - Merging add few new fingers
- Many other designs for specific applications

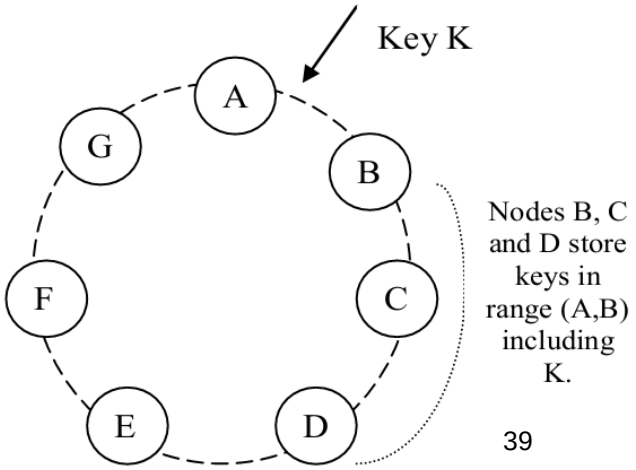




Amazon Dynamo [DeCandia, 2007]

- Highly-available key-value system
  - Many large datasets/objects that only require primary key access
    - Shopping carts, better seller lists, customer preferences, product catalogs, etc.
  - Relational databases are not required, too slow, or bulky
    - Fast reads, high availability for writes
  - Always failing servers, disks, switches
- Objects are replicated in successors
- All peers know about each other using gossiping
- Can read/write to any replica
  - Mechanisms to deal with different versions of objects





Summary – Structured P2P

- Content discovery is within the P2P overlay
- Deterministic performance
- Chord
  - Unidirectional routing
  - Recursive routing
  - Peer churn & failure is an issue
- Kademlia
  - Bidirectional routing
  - Parallel iterative routing
  - Work better under peer failure & churn
- *MySong.mp3* is not same as *mysong.mp3*
- Unbalanced distribution of keys & load

Summary (cont.)

Scheme	Architecture	Routing mechanism	Lookup overhead*	Routing table size*	Join/leave cost	Resilience
Chord	Circular key space	Successor & long distant links	$O(\log N)$	$O(\log N)$	$O(\log^2 N)$	High
CAN	d-torus	Greedy routing through neighbors	$O(dN^{1/d})$	$2d$	$2d$	Moderate
Pastry	Hypercube	Correct one digit in key at time	$O(\log_B N)$	$O(B \log_B N)$	$O(\log_B N)$	Moderate
Tapestry	Hypercube	Correct one digit in key at time	$O(\log_B N)$	$O(\log_B N)$	$O(\log_B N)$	Moderate
Viceroy	Butterfly network	Predecessor & successor links	$O(\log N)$	$O(1)$	$O(\log N)$	Low
Kademlia	Binary tree, XOR distance metric	Iteratively find nodes close to key	$O(\log N)$	$O(\log N)$	$O(\log N)$	High
Cycloid	Cube connected cycles	Links to cyclic & cubical neighbors	$O(d)$	$O(1)$	$O(d)$	Moderate

* N – number of nodes in overlay, d – number of dimensions B – base of a key identifier

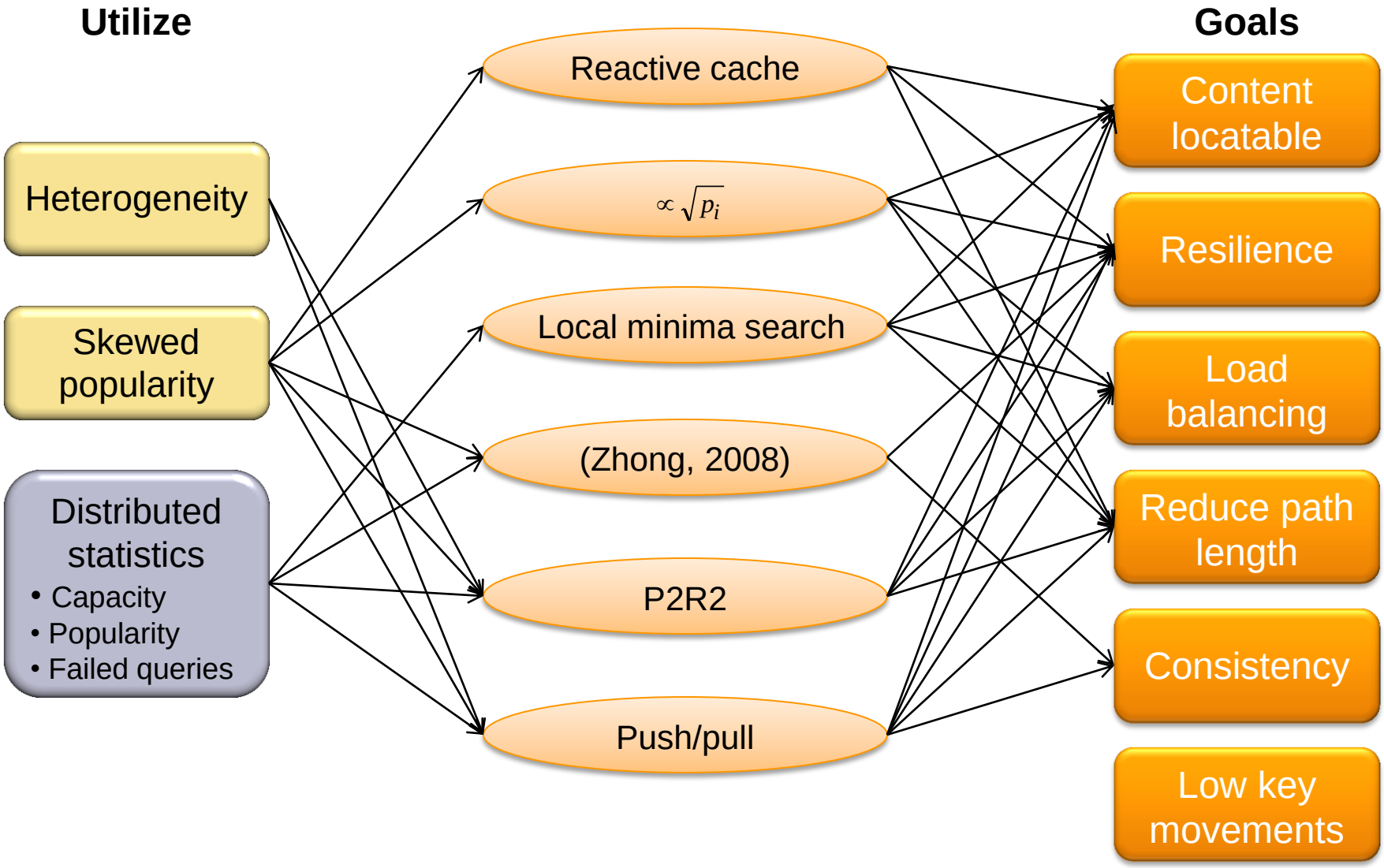
Structured vs. Unstructured

	Unstructured P2P	Structured P2P
Overlay construction	High flexibility	Low flexibility
Resources	Indexed locally	Indexed remotely on a distributed hash table
Query messages	Broadcast or random walk	Unicast
Content location	Best effort	Guaranteed
Performance	Unpredictable	Predictable bounds
Overhead	High	Relatively low
Object types	Mutable, with many complex attributes	Immutable, with few simple attributes
Peer churn & failure	Supports high failure rates	Supports moderate failure rates
Applicable environments	Small-scale or highly dynamic, e.g., mobile P2P	Large-scale & relatively stable, e.g., desktop file sharing
Examples	Gnutella, LimeWire, KaZaA, BitTorrent	Chord, CAN, Pastry, eMule, BitTorrent

Enhancing Lookup Performance

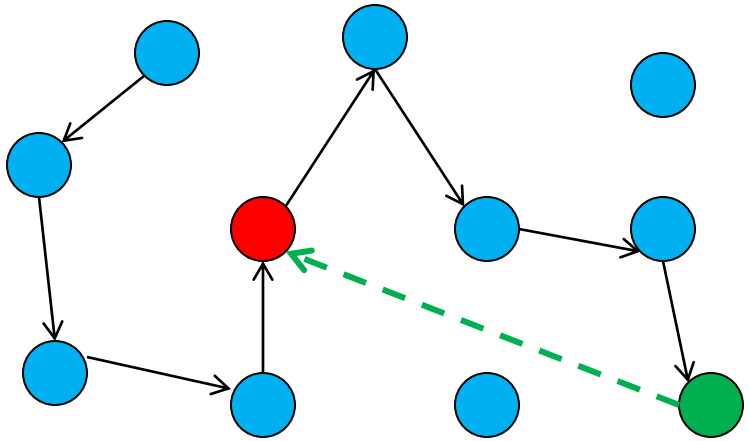
- Many fingers/pointers
- Caching
  - Skewed popularity
  - Reactive/passive
    - Cache what you receive
  - Proactive/active
    - Demand based
    - Community caching
- Replication
  - Load balancing

Unstructured P2P – Performance Enhancements

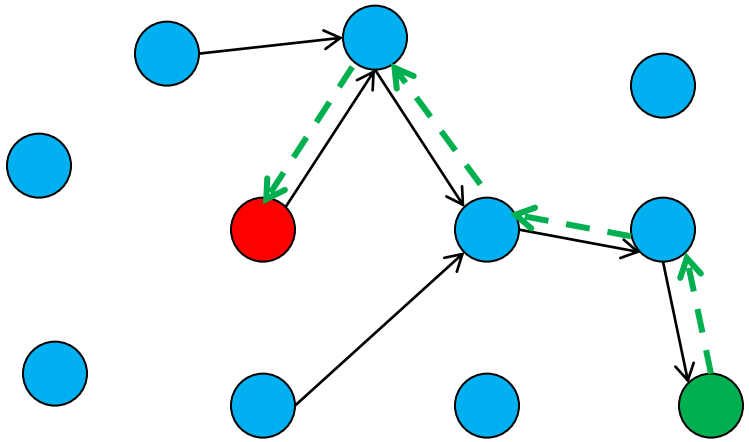


Unstructured P2P – Caching

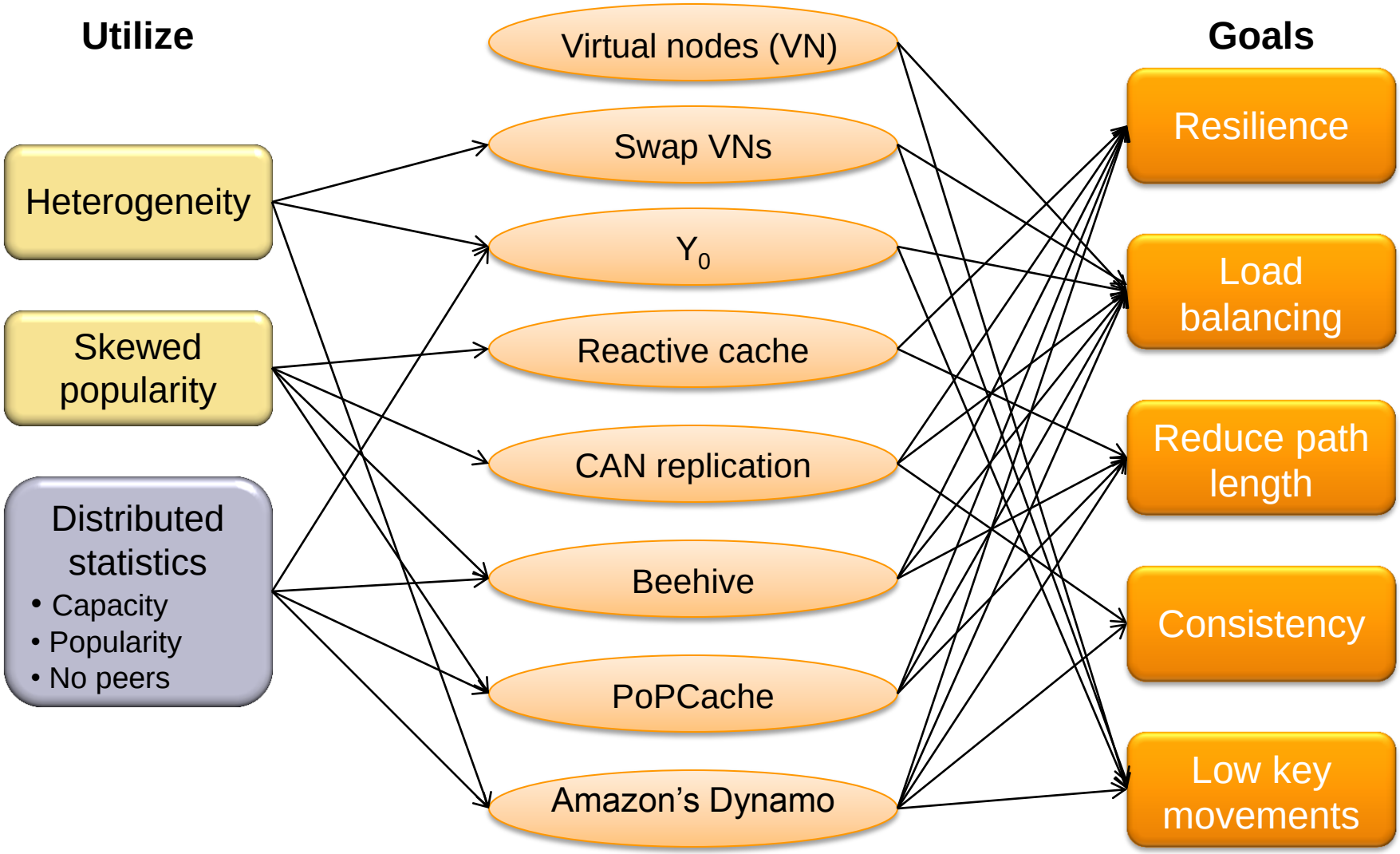
- Passive/reactive caching
  - Cache at query originator
  - Minor improvement



- Active/proactive caching
  - Cache along path
  - Leads to $(f_k)^{½}$ allocation
    - f_k – popularity of content k
  - Relatively better lookup
  - [Cohen, 2002] & [Lv, 2002]

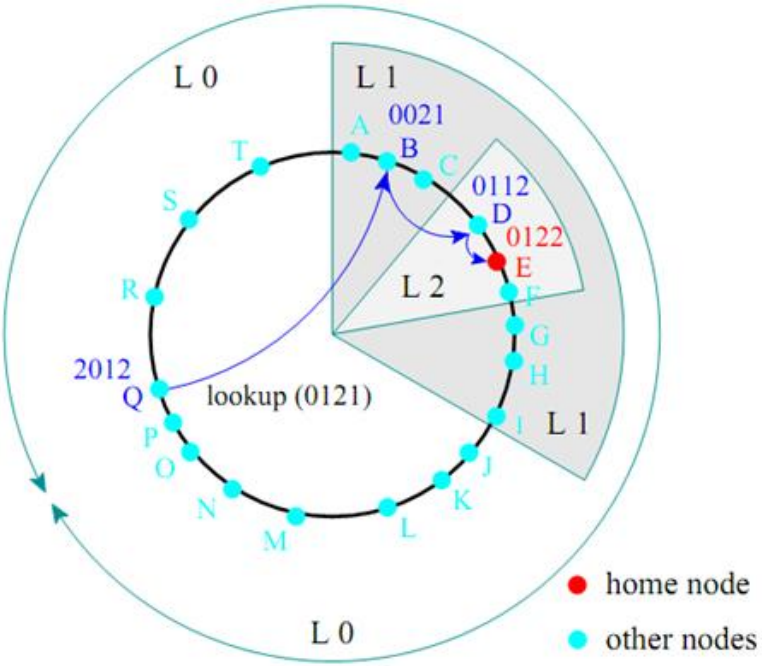


Structured P2P – Performance Enhancements



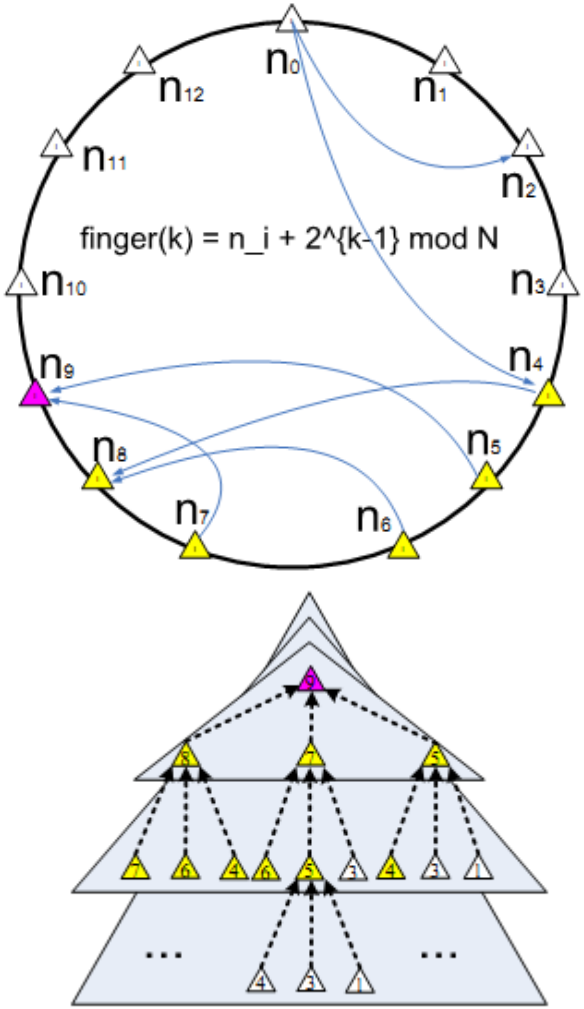
Structured P2P – Caching

- Beehive [Ramasubramanian, 2004]
  - Cache most popular keys everywhere
  - 2nd most popular at ½ of nodes
  - 3rd most popular at ¼ of nodes
  - Assume Zipf's popularity distribution
  - Global popularity estimation
- Issues
  - Unnecessary caching
    - Not all nodes are interested in most popular content
    - Not all intermediate nodes are involved in routing
  - Works only with Zipf's distribution



Structured P2P – Caching (cont.)

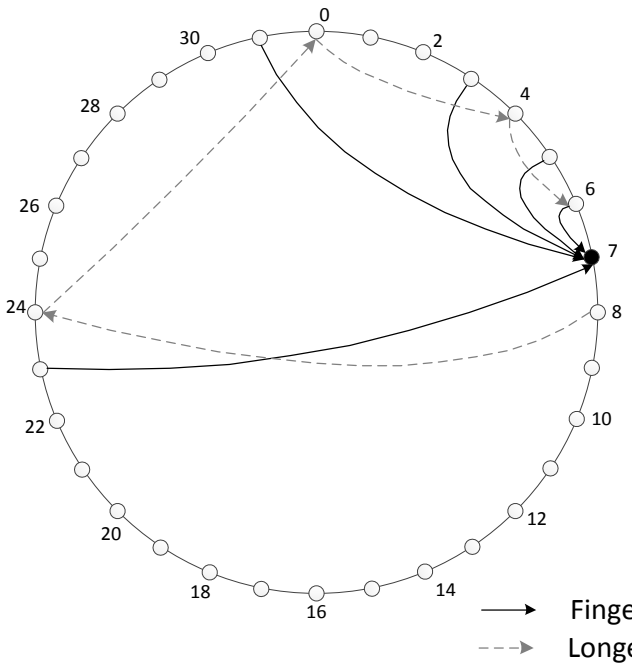
- PoPCache [Rao, 2007]
  - Use overly routing tree to place cache entries
  - $c_k = f_k B$
    - c_k – cache capacity allocated to k
  - Place cache entries from bottom of routing tree
  - Global popularity estimation
- More efficient than Beehive
- Issues
  - Overlay routing tree is not symmetric
  - c_k can exceed N
  - Use of upper bound O($log\ N$)

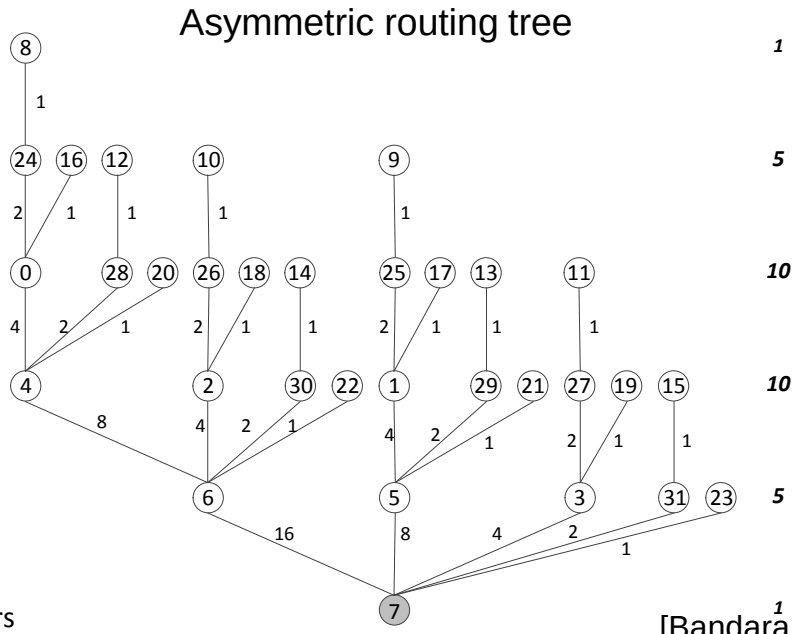


Local Knowledge-based Distributed Caching (LKDC)

- Each overlay node
  - Independently decides what keys to cache based on number of queries it forwards
  - Tries to maximize number of queries it can answer
- NP-complete [Bandara, 2011d]
- Relaxed version of problem (namely GKDC) can be used to determine
  - Where to place cache entries?
  - How many entries to place?
- GKDC says local statistics are adequate to decide what to cache at a node
  - Heuristic algorithm based on Least Frequently Used (LFU) caching

Global Knowledge-based Distributed Caching (GKDC)

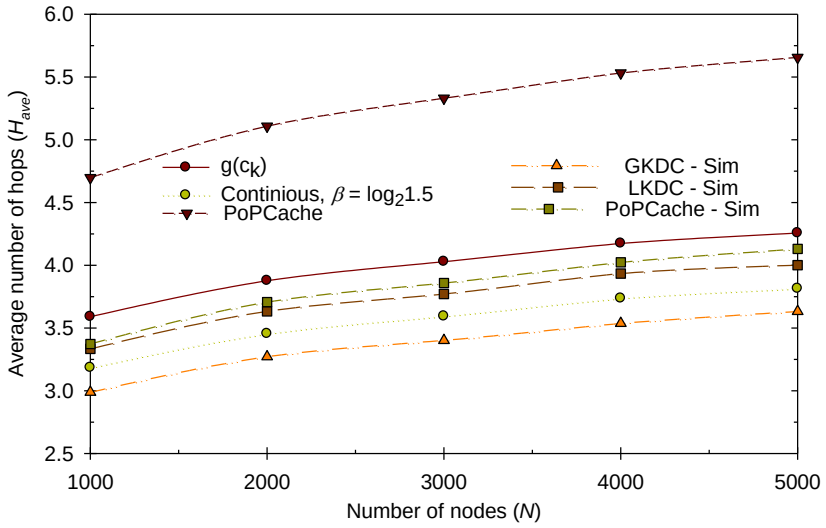


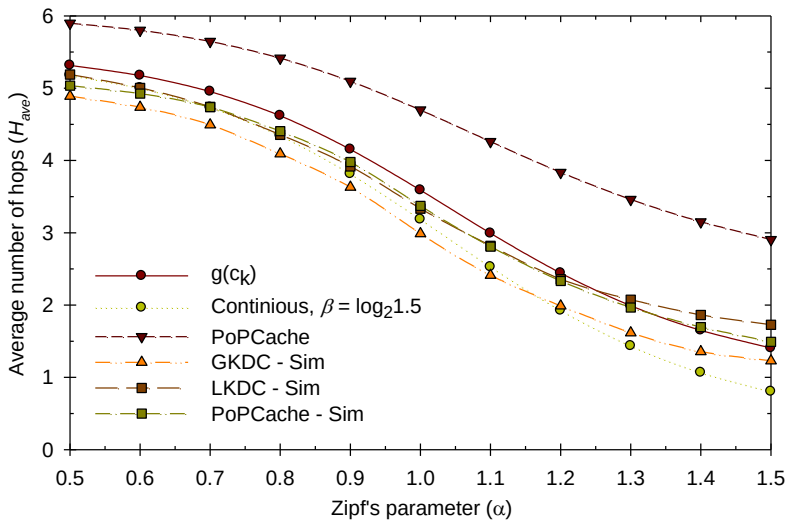


[Bandara, 2011d]

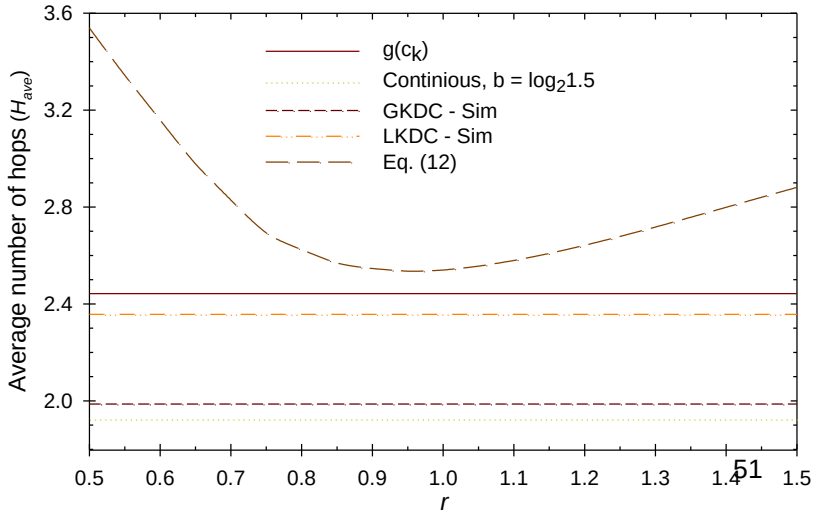
- Where to place?
  - At nodes that forward most number of messages
  - 6, (4, 5), (0, 1, 2, 3), …
- How many?
  - $$c_k = \begin{cases} N-1 & \text{if } k \le l \\ \dfrac{f_k\left(B - l(N-1)\right)}{P(l,K,\alpha)} & \text{else} \end{cases}$$
  - Place according to local popularity

Heuristic-Based LKDC – Performance



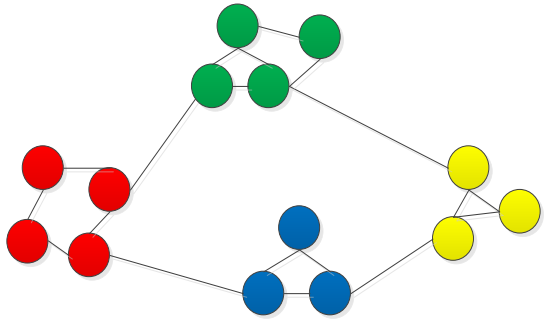


- Same performance as PoPCache using
  - Small caches
  - Local statistics only
- Works with any skewed distribution



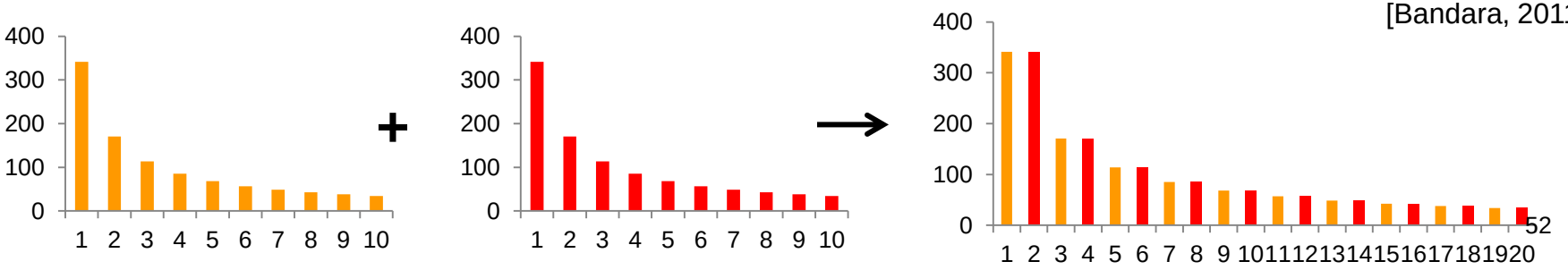
Community-Aware Caching

- Many small P2P communities are emerging
- Enhancing lookup
  - Unstructured → Restructure overlay
  - Structured → Cache only most popular resources in entire system
- However
  1. Communities are not isolated
  2. Loose popularity due to aggregation



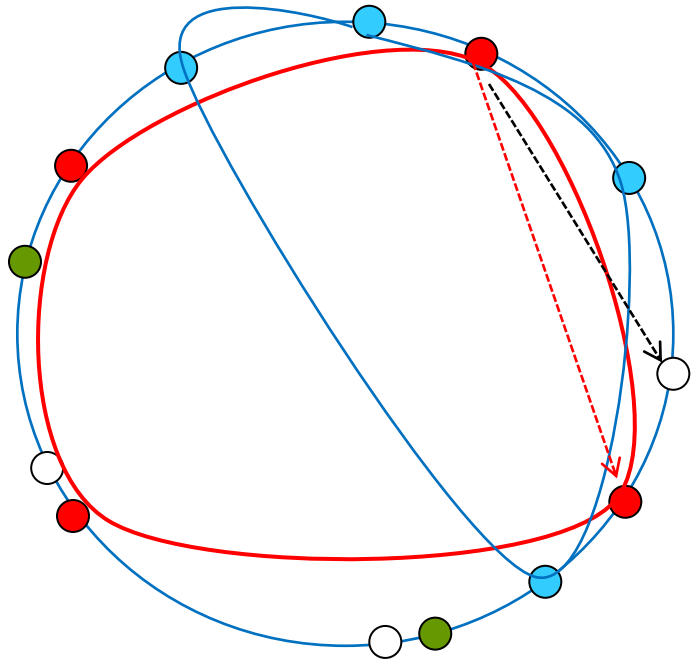
Community	EX	FE	SP	TB	TS	TE	TR
FE	0.38						
SP	0.00	0.00					
TB	0.40	0.29	0.00				
TS	0.48	0.33	0.00	0.48			
TE	0.53	0.23	0.00	0.31	0.25		
TR	0.10	0.08	0.00	0.06	0.09	0.06	
YB	0.36	0.35	0.00	0.29	0.42	0.20	0.04

[Bandara, 2011d]



Community-Aware Caching

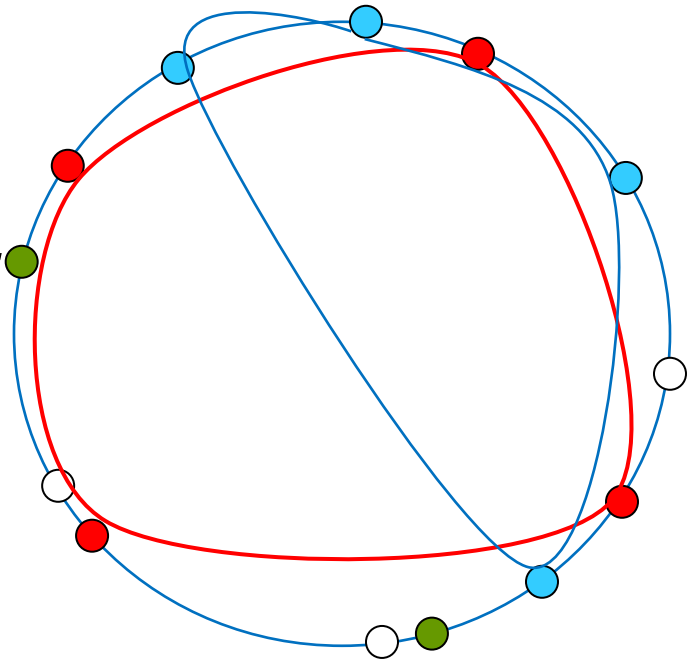
- Goal – Reduce content mixing or overlay restructuring
  - Preserve popularity
- Each community forms a sub-overlay
  - Links to community members
  - Sample nodes pointed by fingers to find community members
- Forward messages through community members
  - Nodes can identify what's popular within their community



- By probing i-th finger & its successor $2(i + 2 \log N - m) - 1$ nodes can be found
- Community of size M have $M/2^{m-i+1}$ peers in the range of i-th finger

Community-Aware Caching (cont.)

- Cache based on communities' interest
  - *"What is important to me is also important to other community members"*
  - *"They may have queries it before me"*
- Heuristic-based LKDC caching algorithm
  - Weighted LFU caching
  - Local statistics only
- Pros
  - Works with any structured overlay that provide multiple paths
  - Peers can be in any community
  - Preserves path length bound $O(\log N)$

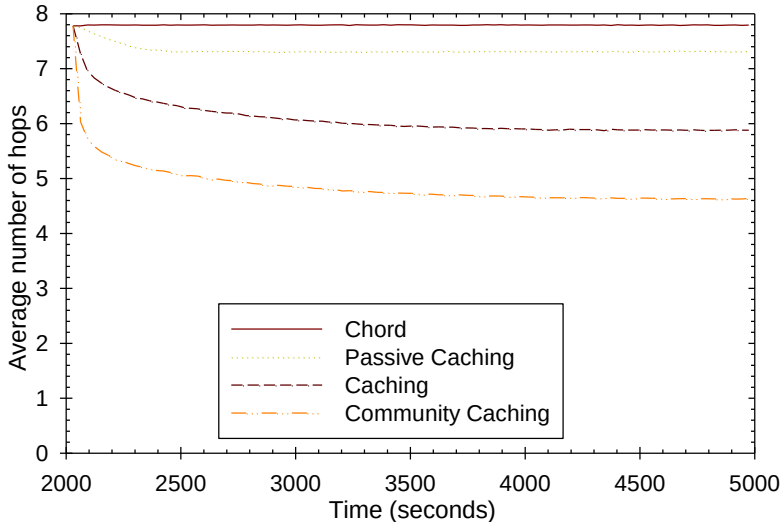


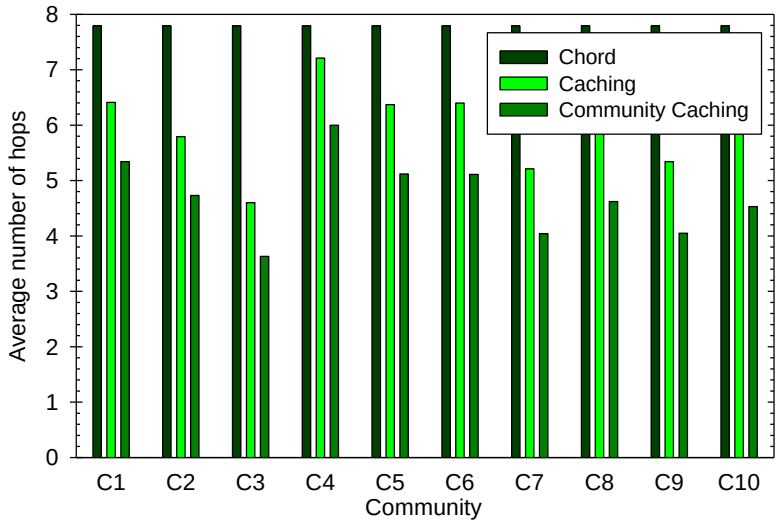
Community-Aware Caching – Simulation Setup

- 15,000 nodes
- 10 communities
- Chord overlay
- Simulated using OverSim

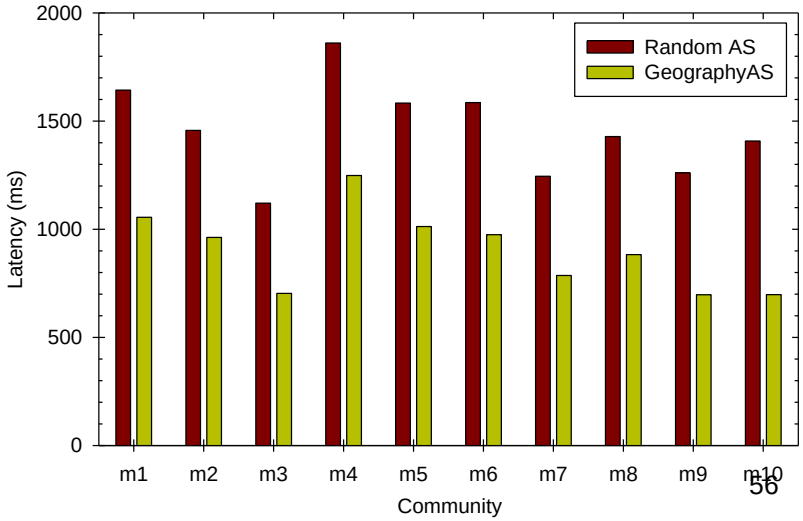
Community	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
No of nodes (apx.)	600	600	600	1,200	1,200	1,200	1,200	1,200	2,400	4,800
Zipf's parameter	0.85	0.95	1.10	0.5	0.80	0.80	1.0	0.90	0.90	0.75
No of distinct keys	40,000	30,000	30,000	40,000	40,000	40,000	50,000	50,000	50,000	50,000
Similarity with community (x)	0.2 (C_8)	0	0.1 (C_7)	0.2 (C_9)	0.3 (C_8) 0.5 (C_7)	0	0.1 (C_3) 0.5 (C_5)	0.3 (C_5) 0.2 (C_1)	0.4 (C_1) 0.2 (C_4) 0.3 (C_{10})	0.3 (C_9)
Queries for rank 1 key	4,516	8,535	17,100	603	6,454	6,454	21,059	11,956	23,911	17,030

Community-Aware Caching – Results

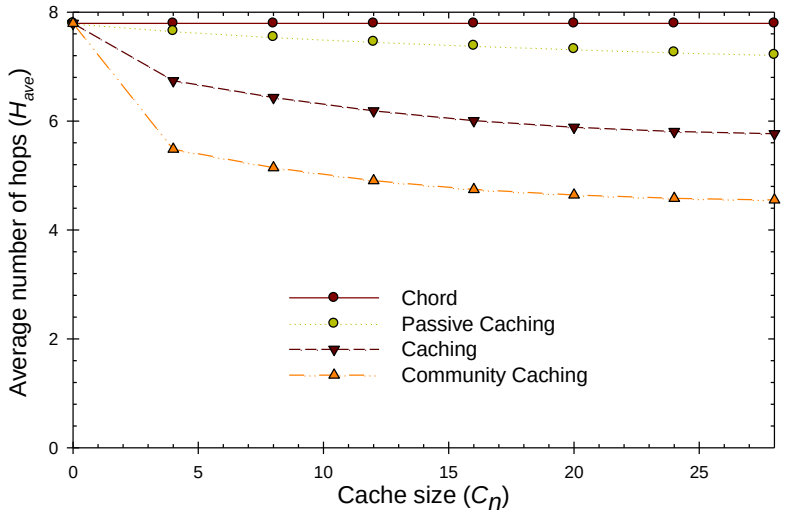


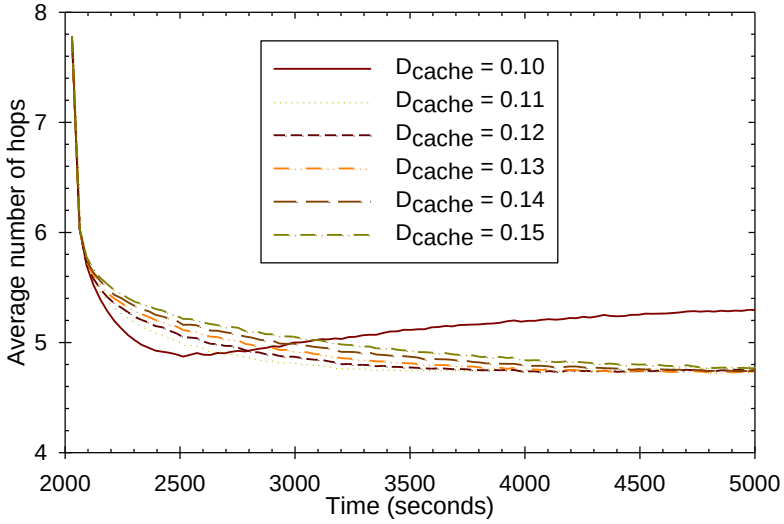


- More popular communities
  - 48-53% reduction in path length
- Least popular community
  - 23% reduction (7% with caching)
- Geographic communities
  - 48-50% latency reduction

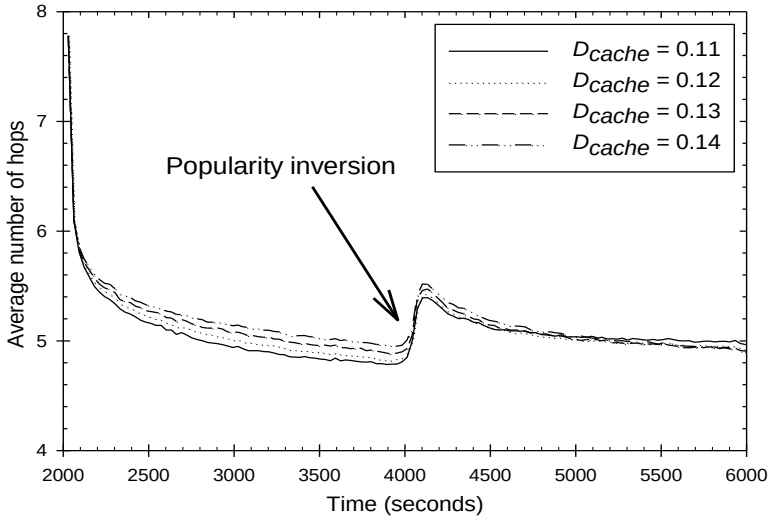


Results (cont.)





- Significant performance with small caches
- Caching threshold reduce cache thrashing, & overhead
- Fast response to popularity changes

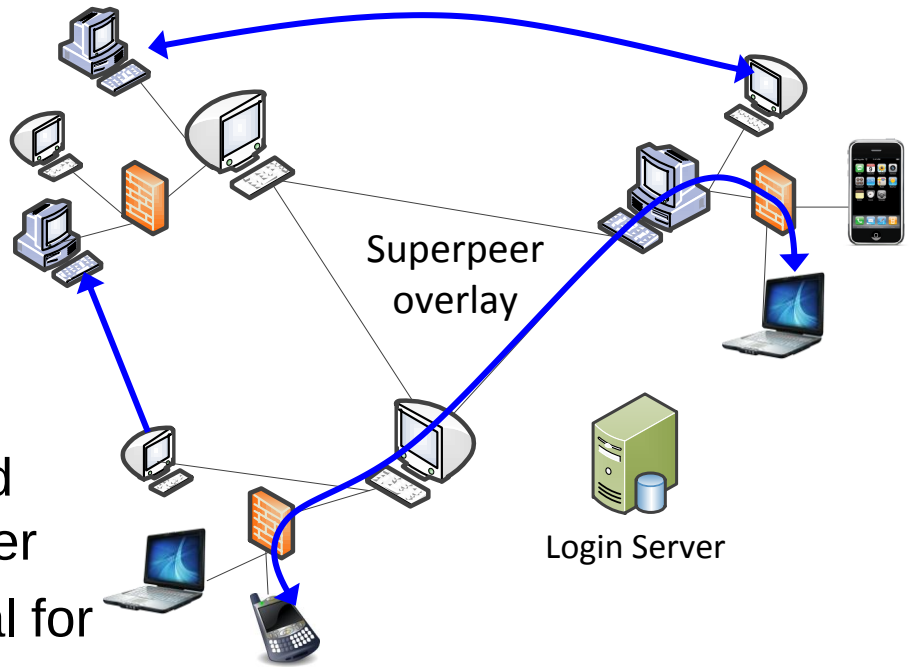


Load Balancing

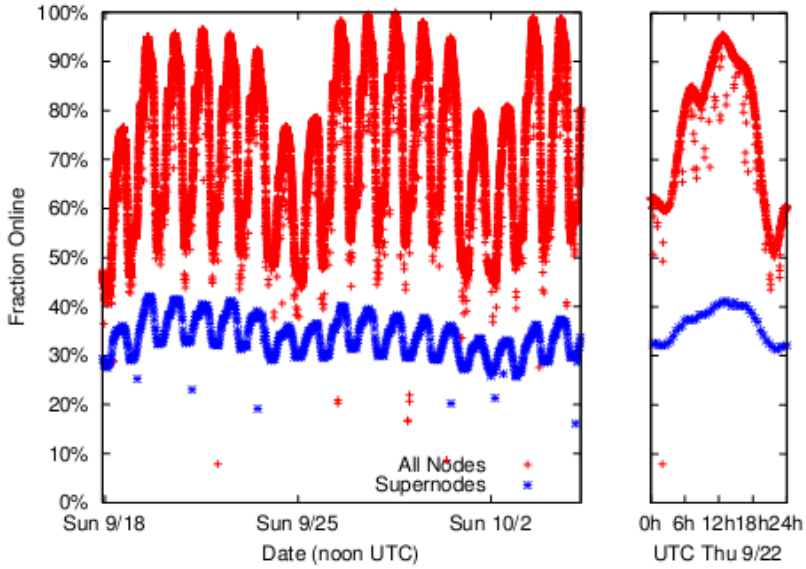
- Swap nodes
  - Overloaded node swap its overlay location with a resourceful node
  - Load may exceed capabilities or any node → Not scalable
- Virtual nodes
  - A physical node appear as several virtual nodes
  - Chord index table is balanced when there are $N \log N$ virtual nodes
  - Increase lookup cost, e.g., Chord $O\{\log (N \log N)\}$
- Replication
  - Save in multiple neighbors, e.g., Kademlia & CAN
  - Doesn't work with some overlays, e.g., Chord
- Caching
  - Nodes that cache reduce query load on indexing node
  - Doesn't work well with mutable content and/or large indexes

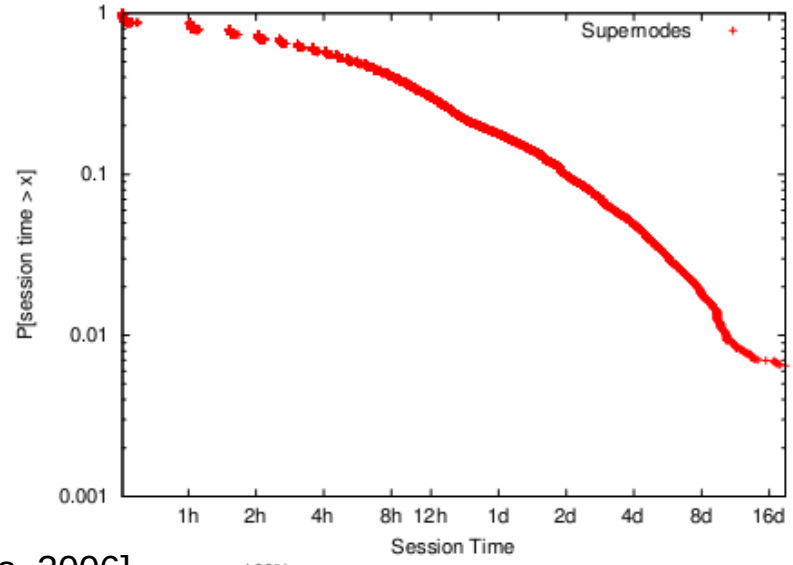
Skype

- Proprietary
  - Encrypted control & data messages
  - Many platforms
- Voice/video calls, instant messaging, file transfer, video/audio conferencing
- Superpeer overlay
  - Related to KaZaA
  - Based on bandwidth, not behind firewall/NAT, & processing power
  - Enables NAT & firewall traversal for ordinary peers



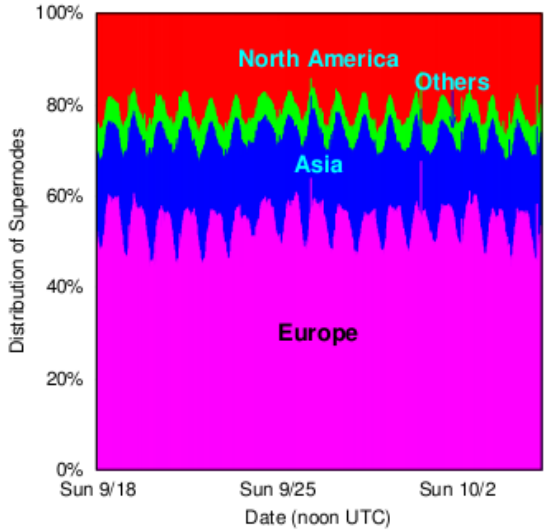
Skype (cont.)





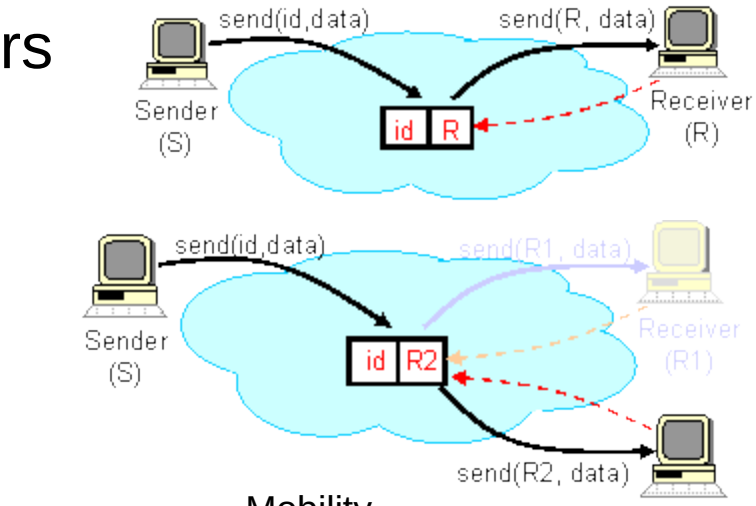
[Guha, 2006]

- 30% superpeers
  - Relatively stable
- Diurnal behavior
- Longer session length than typical P2P - Heavy tailed



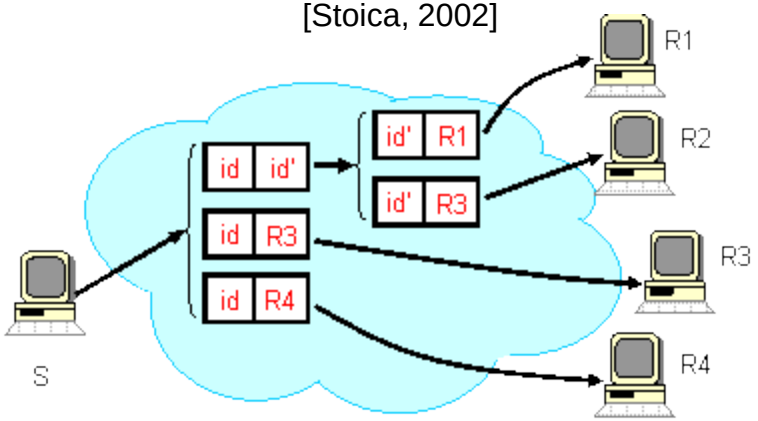
P2P as Publisher/Subscriber Services

- Rendezvous service for consumers & producers
  - File sharing, RSS feeds, mobility, multicast, data availability in sensor networks
- i3 – Internet Indirection Infrastructure [Stoica, 2002]
  - Packets have unique IDs
  - Receive requests packets from DHT
  - Unicast, anycast, multicast, mobility
- Many other application specific solutions



Mobility

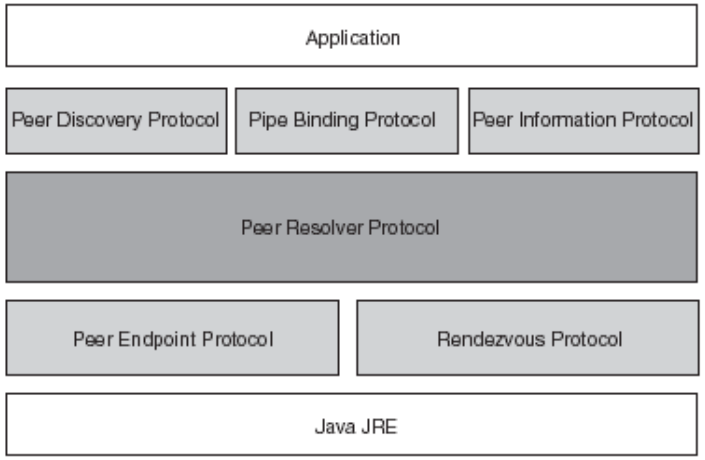
[Stoica, 2002]

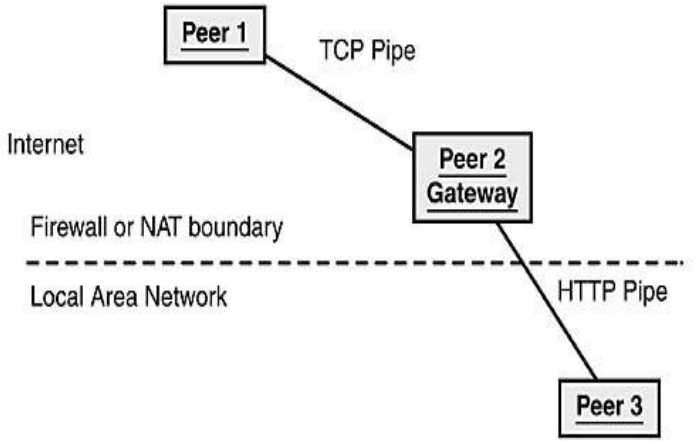


Multicast

P2P Middleware – JXTA

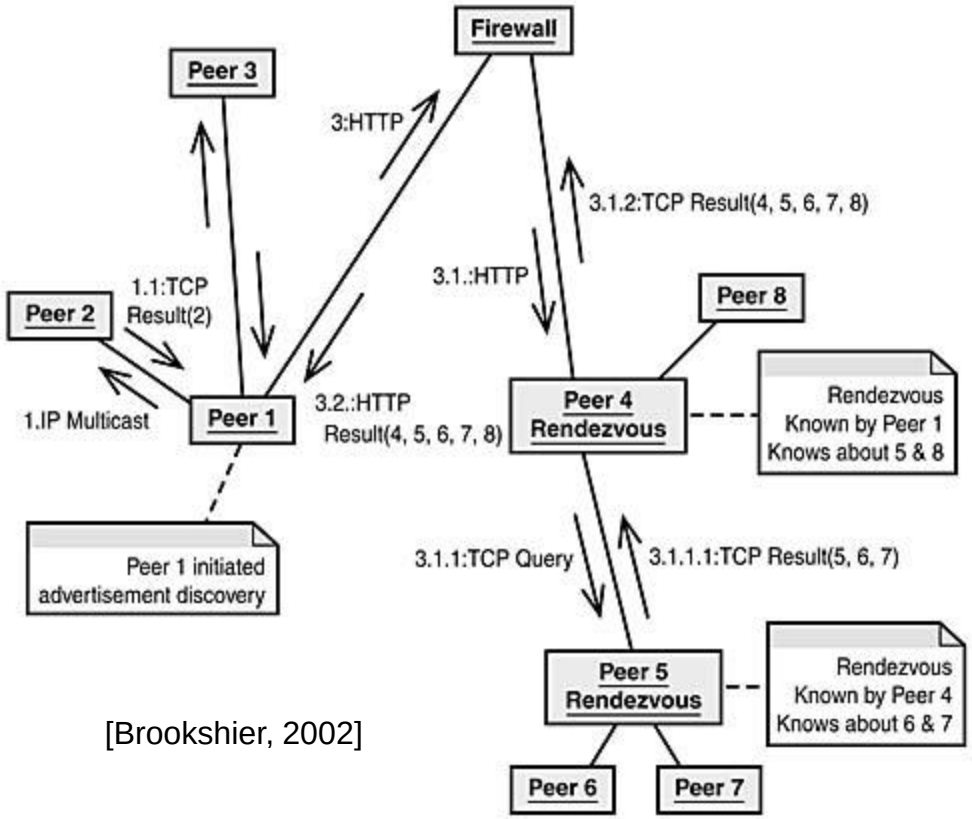
- XML-based P2P protocol specification
  - Sun Microsystem – up to 2010
  - Implementations for Java, C/C++, C#
- Many protocols
  - Peer Discovery – Resource search
  - Pipe Binding – Addressable messaging
  - Peer Information – Monitoring
  - Peer Resolver – Generic query service
  - Peer Membership – Security
  - Rendezvous – Message propagation
  - Peer Endpoint – Routing
- Use any available protocol to traverse firewalls/NATs
  - HTTP, TCP





[Brookshier, 2002]

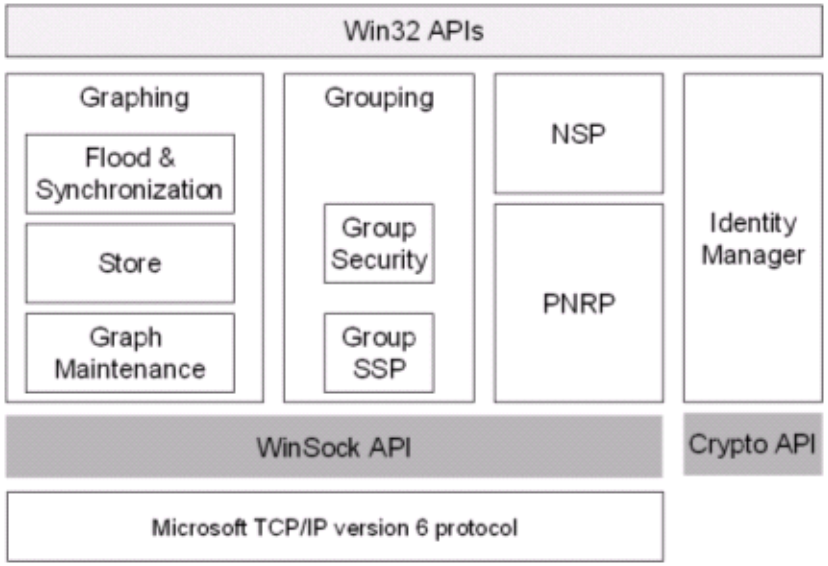
JXTA (cont.)



[Brookshier, 2002]

P2P Middleware in Windows

- Available since Windows XP
  - Supports IPv6, IPV4 needs tunneling
- Graphing
  - Overlay connectivity maintenance
  - Flooding based
- Grouping
  - Peers groups
  - Access control
- Distributed name lookup
- Peer identity management
- Applications
  - Windows HomeGroup, Meeting Space, Internet Computer Names



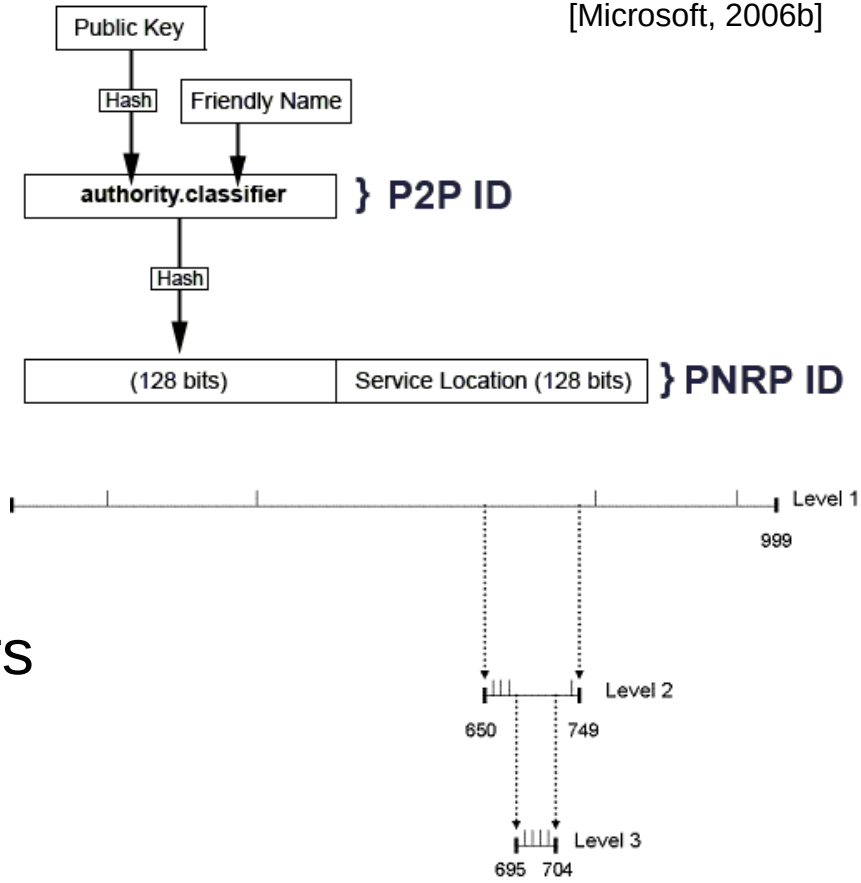
[Microsoft, 2006a]

NSP – Name Service Provider
PNRP – Peer Name Resolution Protocol

Windows Peer Name Resolution Protocol

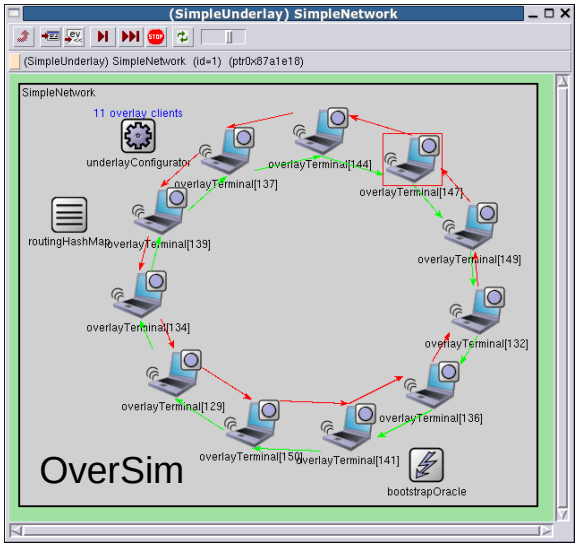
- Scalable, secure, & dynamic name resolution protocol
- P2P ID
  - End-point identifier of an application, service, user, group
  - Circular ID space
- Service location
  - Locally unique service ID
- Peers cache IDs of other peers
  - Hierarchical like Kademlia
- Iteratively search for peer(s) with shorter distance to destination

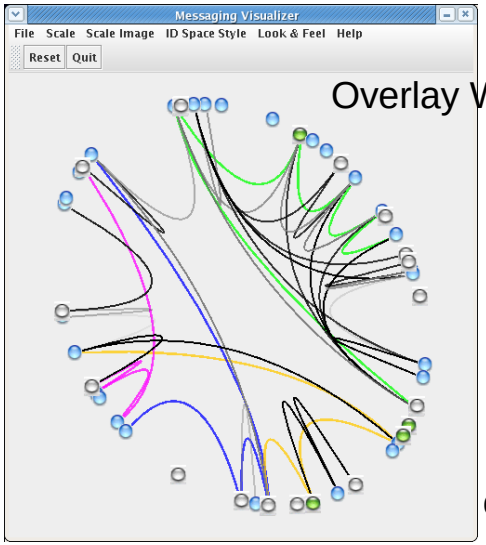
[Microsoft, 2006b]



P2P Simulators

- OverSim
  - OMNeT++ based, GUI, C++
  - Prebuilt (un)structured protocols
  - Underlay support, e.g., GT-ITM
  - *oversim.org*
- PeerSim
  - Java based
  - Fast cycle-based simulation
  - Prebuilt (un)structured protocols
  - *peersim.sourceforge.net*
- Overlay Weaver
  - Emulator
  - Java based, GUI
  - Prebuilt (un)structured protocols
  - *overlayweaver.sourceforge.net*





Challenges & Opportunities

Challenges	Opportunities
P2P communities • Identify, form, & maintain	Community-aware performance enhancements • Lookup, fast download, reputation, trust, etc. • Interest-based topology adaptation • Capturing/using social relationship among peers
Topology missmatch & ISP traffic blocking • P4P, ALTO • User's don't like ISPs to suggest	Transparent selection of local peers • User & ISP friendly designs • Incentives • Network coordinates
QoS & QoE	Best effort → deterministic performance • Predictable download times • Real-time & VoD streaming

Challenges & Opportunities (cont.)

Challenges	Opportunities
Free riding • 15% of Gnutella peers contribute to 94% of content • 63% of peers never responded to queries [Adar, 2000]	Incentives, trust, & enforcement • Revenue models • Ads, pay-per-click • Retaining users after downloads
Security • File pollution, viruses, worms • Topology worms • Anonymity • Route poisoning, sink holes, Sybil attacks	Content protection & overlay security • Signed content • Enabling/disabling anonymity • Authentication & accountability • Centralized solutions are proposed • Community support to moderate content
Copyright violation • Direct/indirect infringement	Monitoring, enforcement • Active/passive monitoring
Load imbalance	Static & dynamic load balancing

Challenges & Opportunities (cont.)

Challenges	Opportunities
Integrating P2P & social networks • Capturing social relationships • Privacy	Enhanced performance, QoE • Social graph-based P2P overlays • Better incentives • Better caching, replication, load distribution

Low resilience in structured P2P
• Sudden departures
  • Route failures
  • Loss of content index
	Maximize resilience/availability
	• Enhancing consistency of replicas

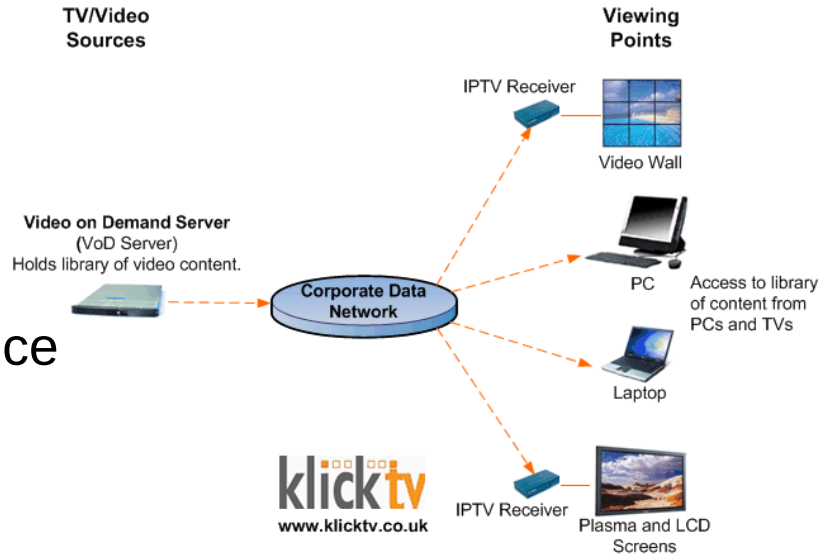
Connectivity • NAT, firewalls, & proxies • 66% of BitTorrent peers are behind firewalls [Zhang, 2009]	Connectivity services & tunneling • Performance should not depend on whether a peer is behind a NAT/firewall

Outline

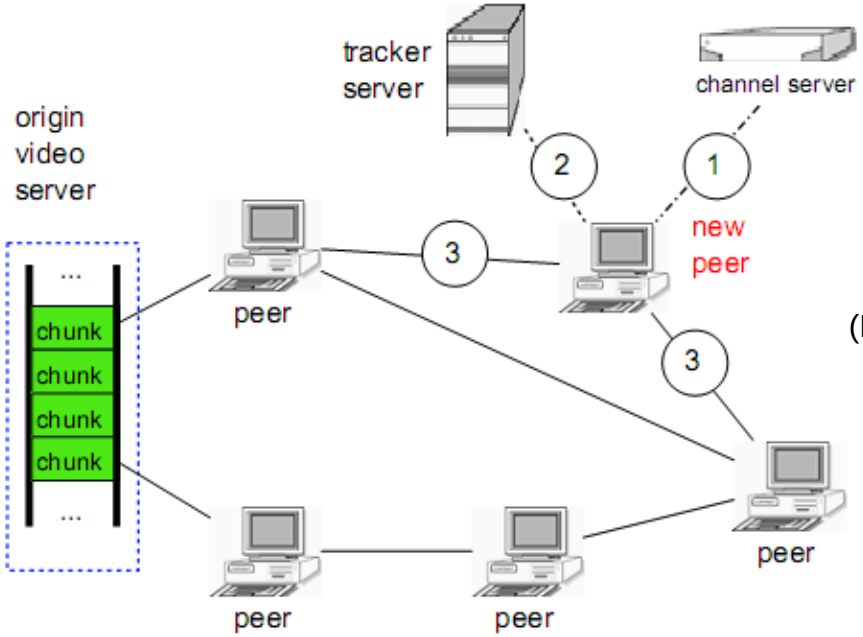
- File sharing
  - Unstructured vs. structured overlays
  - Performance enhancements
    - More state, caching, replication
  - Opportunities & challenges
- Streaming
  - Tree-push vs. mesh-pull
  - Opportunities & challenges
- Resource sharing
  - Collaborative P2P
  - Resource aggregation
  - Opportunities & challenges

P2P streaming

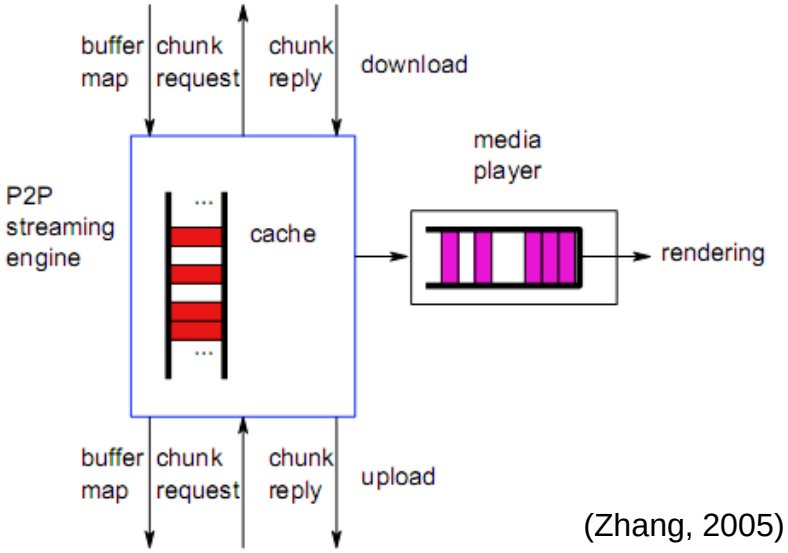
- Emergence of IPTV
  - Content Delivery Networks (CDNs) can't handle bandwidth requirements
  - No multicast support at network layer
- P2P
  - Easy to implement
    - No global topology maintenance
  - Tremendous scalability
    - Greater demand → Better service
    - Cost effective
  - Robustness
    - No single point of failure
    - Adaptive
  - Application layer



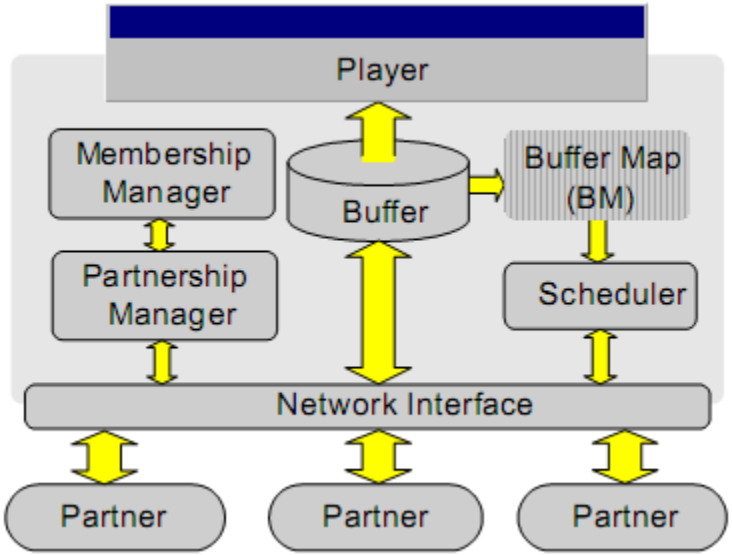
P2P Streaming – Components



(Hie, 2008)



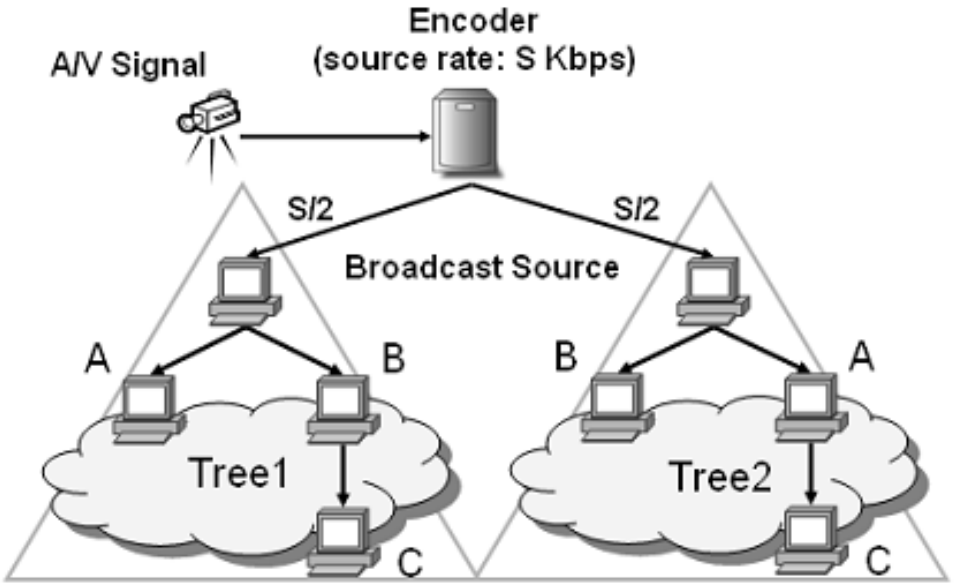
(Zhang, 2005)



- Chunk
  - Segment of the video stream
  - E.g., one second worth of video
- Partners
  - Subset of known peers that a peer may actually talk to

Tree-Push Approach

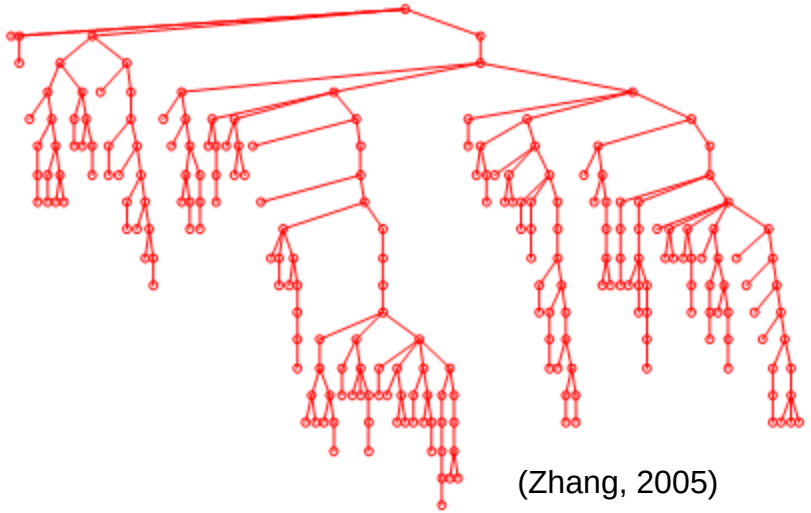
- Construct overlay tree starting from video source
- Parent peer selection is based on
  - Bandwidth, latency, number of peers, etc.
- Data/chunks are pushed down the tree
- Multi-tree-based approach
  - Better content distribution
  - Enhanced reliability



(Liu, 2008)

Tree-Push Approach – Issues

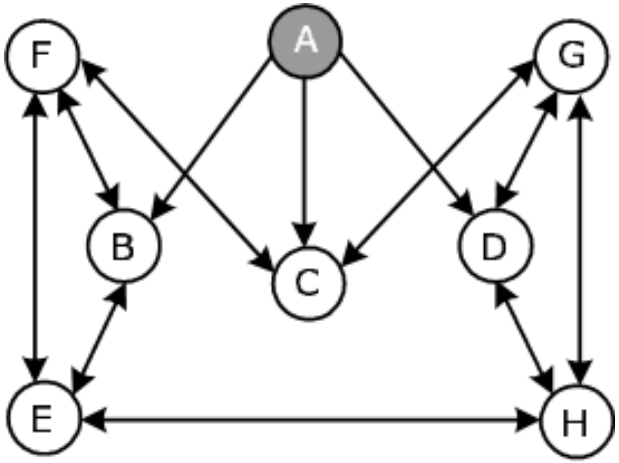
- Connectivity is affected when peers at the top of the tree leave/fail
  - Time to reconstruct the tree
  - Unbalanced tree
- Majority of the peers are leaves
  - Unable to utilize their bandwidth
  - High delay



(Zhang, 2005)

Mesh-pull approach

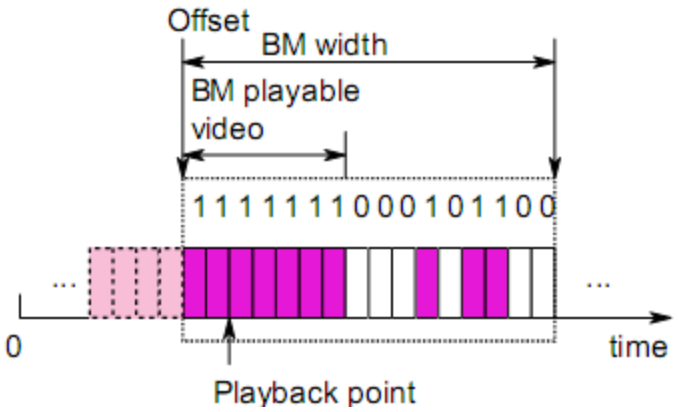
- A peer connects to multiple peers forming a mesh
- Pros
  - More robust to failure
  - Better bandwidth utilization
- Cons
  - No specific chunk forwarding path
  - Need to pull chunks from partners/peers
  - Need to know which partner has what
- Used in most commercial products



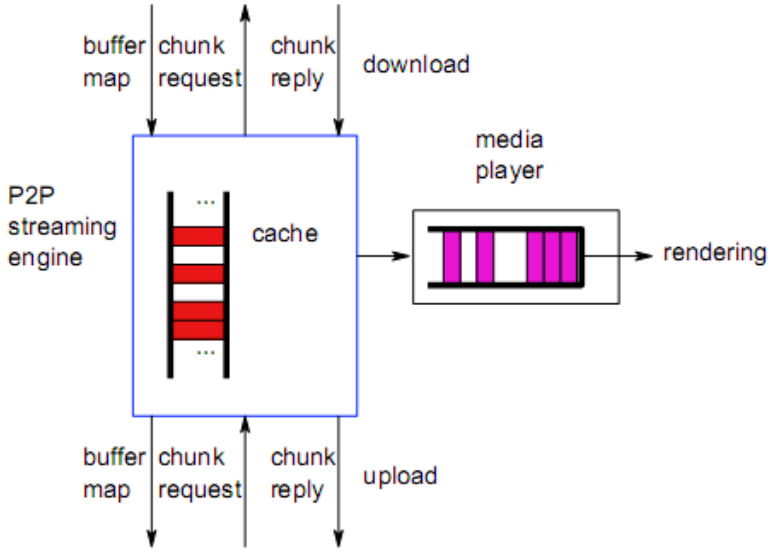
(Zhang, 2005)

Chunk Sharing

- Each peer
  - Caches a set of chunks within a sliding window
  - Shares its chunk information with its partners
- Buffer maps are used to inform chunk availability
- Chunks may be in one or more partners
- What chunks to get from whom?



(Hie, 2008)

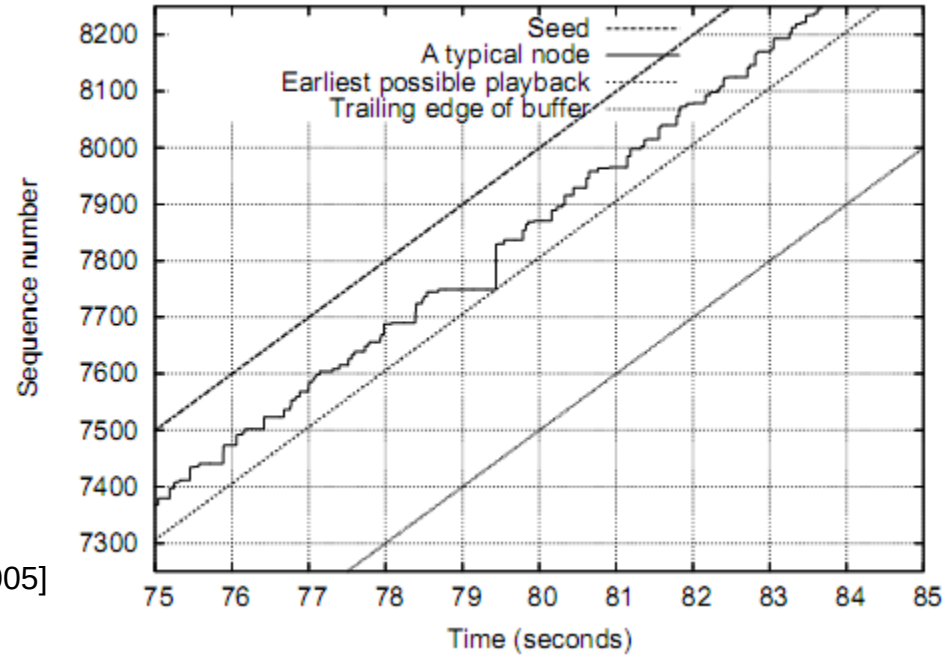


Chunk Scheduling

- Some chunks are highly available while others are scare
- Some chinks needs to be played soon
- New chunks need to be pulled from video source
- Chunk scheduling consider how a peer can get chunks while
  - Minimizing latency
  - Preventing skipping
  - Maximizing throughput
- Chunk scheduling
  - Random, rarest first, earliest deadline first, earliest deadline & rarest first
- Determines user QoE
- Most commercial products use TCP for chunk transmission
  - Control message overhead ~1-2%

Random Scheduling

- One of the earliest approach – used in Chainsaw
- Peers periodically share buffer maps
- Select a random chunk & request it from one of the partners having the chunk
- Some peers may experience significant playback delay
  - 1-2 minutes
- Skipping is possible

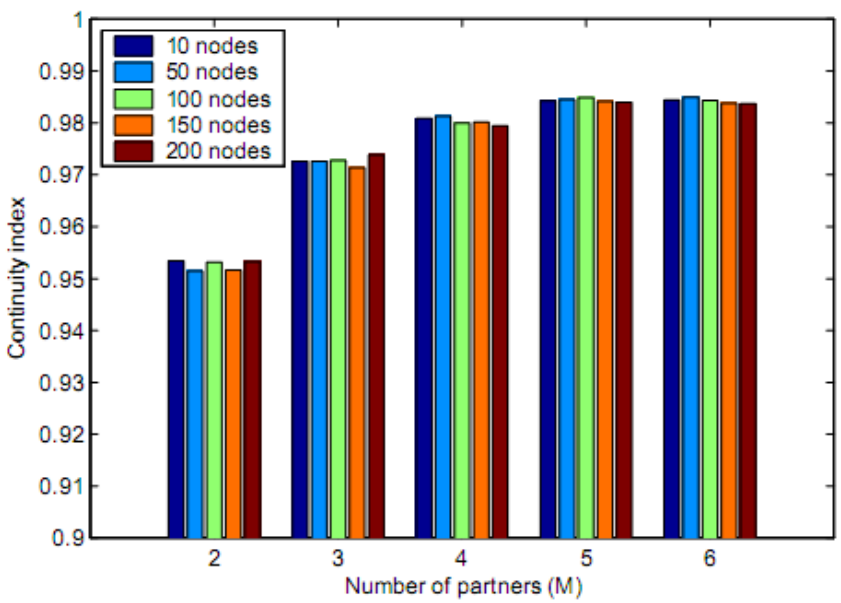


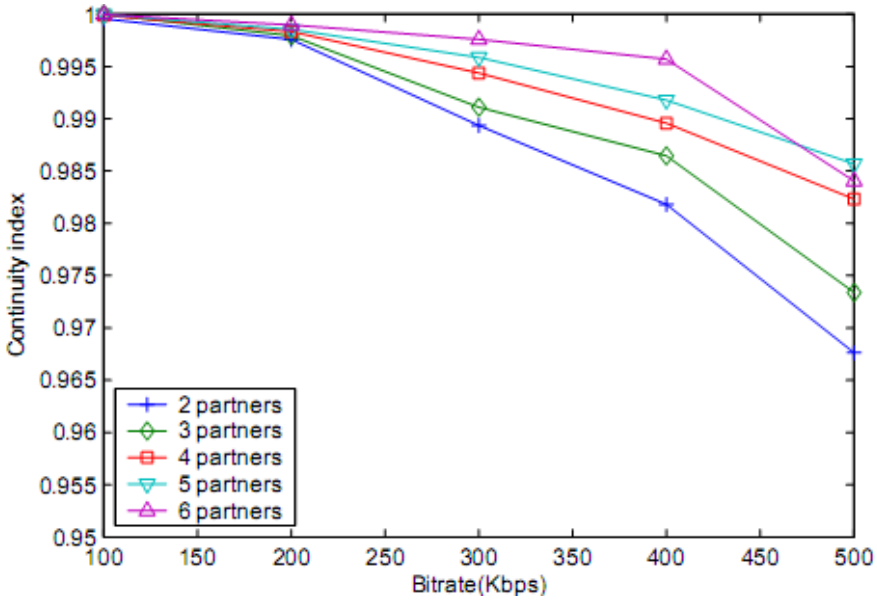
[Pai, 2005]

Rarest First Scheduling

- Used in CoolStraming
  - Chunk = 1 sec video, 120 chunk in sliding window
- A peer gets the rarest chunk so that chunk can be spread to its partners
- Steps
  1. Gather buffer maps periodically
  2. Calculate number of suppliers (i.e., partners with chunk) for each chunk
  3. Request chunks with the lowest number of suppliers
  4. For chunks with multiple suppliers, request from the supplier with highest bandwidth & free time
     - Gather application-level bandwidth data for each partner
- Request are made through a bitmap

Rarest First (cont.)

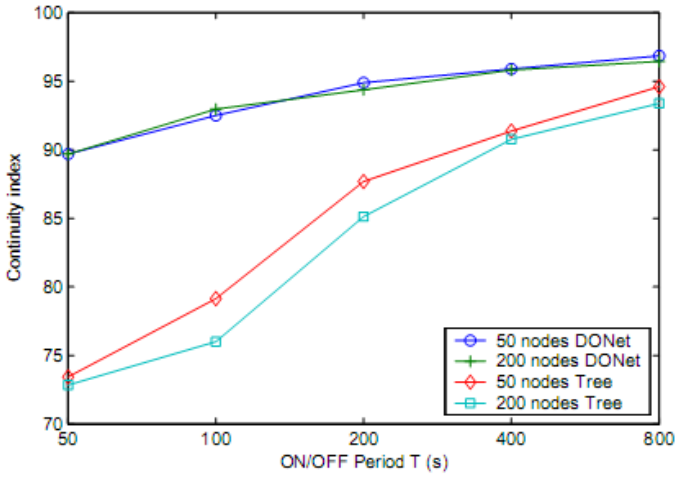




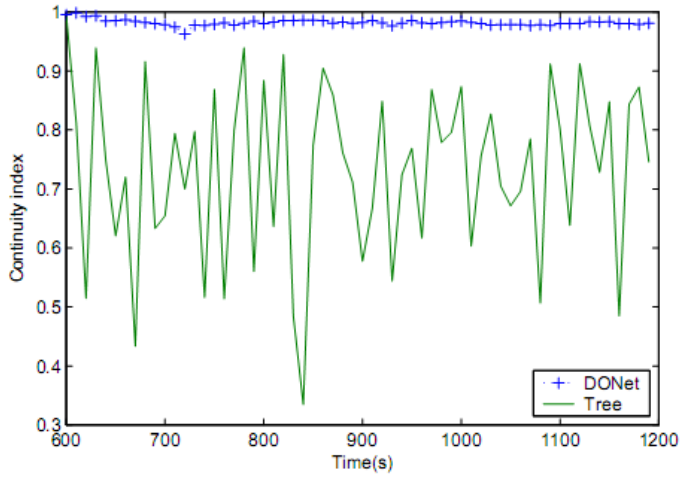
(Zhang, 2005)

- It is sufficient to maintain 4 partners
- Discover more peers overtime – use gossiping
- Keep only the partners that have sufficient bandwidth & more chunks

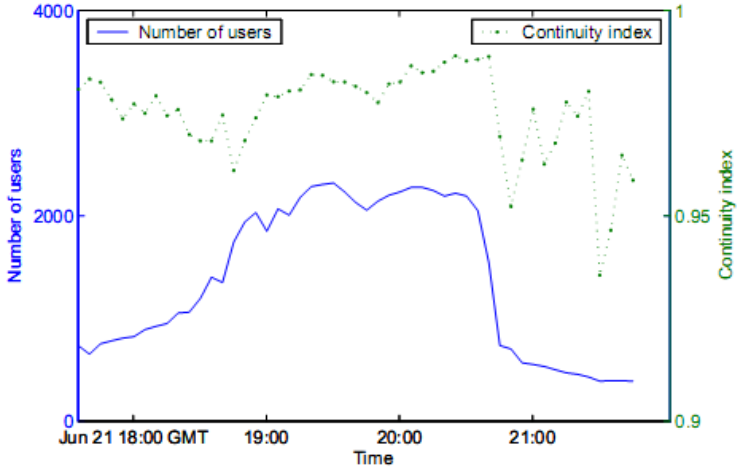
Rarest First (cont.)



(Zhang, 2005)

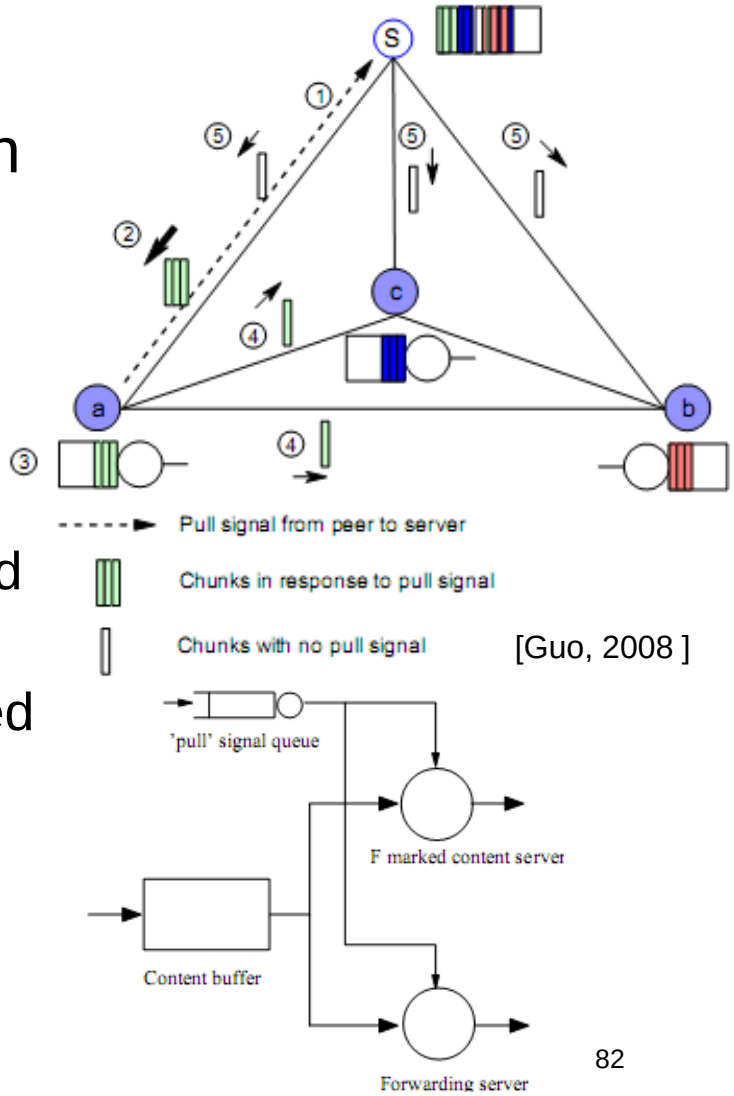


- More robust than tree-push approach
- Larger user community → Better service quality
- Most users experience < 1 min delay



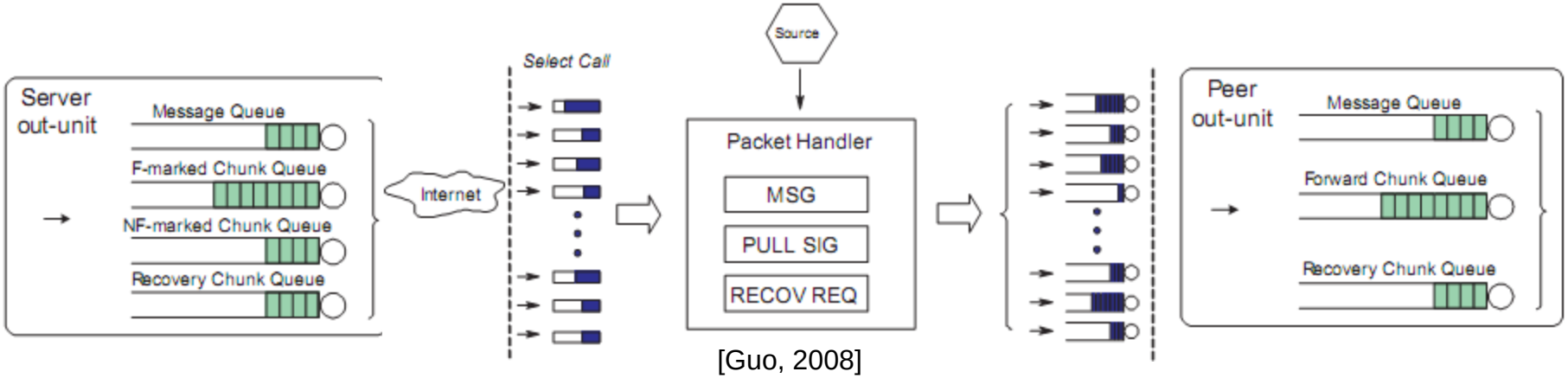
Queue-Based Scheduling

- Objectives – Continuity & quality
- Try to maximize bandwidth utilization of peers
- Available bandwidth is inferred from queue status
- Steps
  1. Peers pull chunks from source (marked as *F*)
  2. Peers push chunks to its peers (marked as *NF*)
  3. If source is not busy, it push chunks to peers (marked as *NF*)



[Guo, 2008]

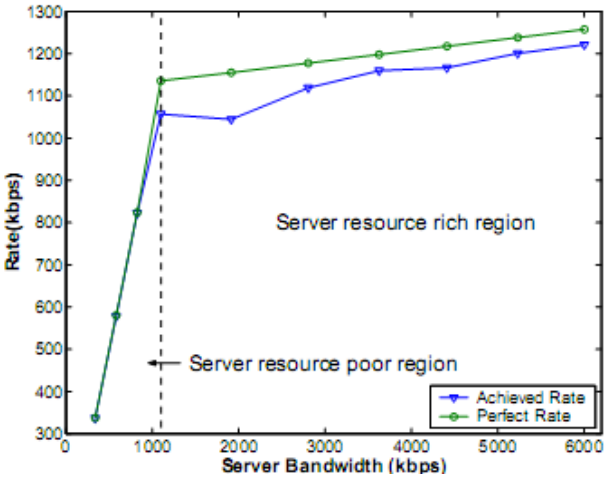
Queue Based Scheduling (cont.)

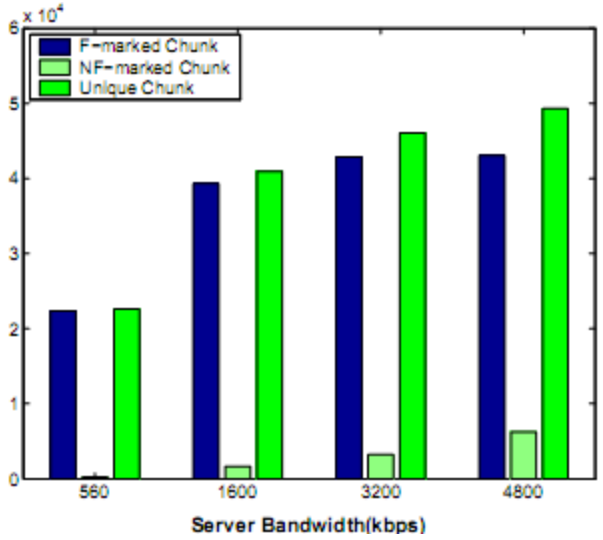


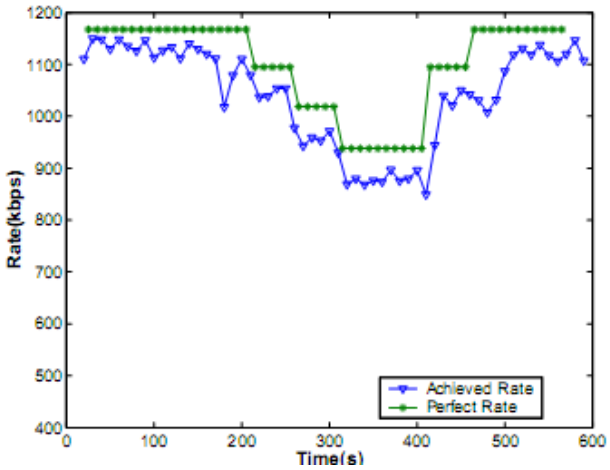
[Guo, 2008]

- Queues have different priorities
- Missing chunks can be requested from source or peers
- Maintain a separate queue & a connection to each peer
  - Prevents a slower peer from slowing down the whole system

Queue Based Scheduling (cont.)





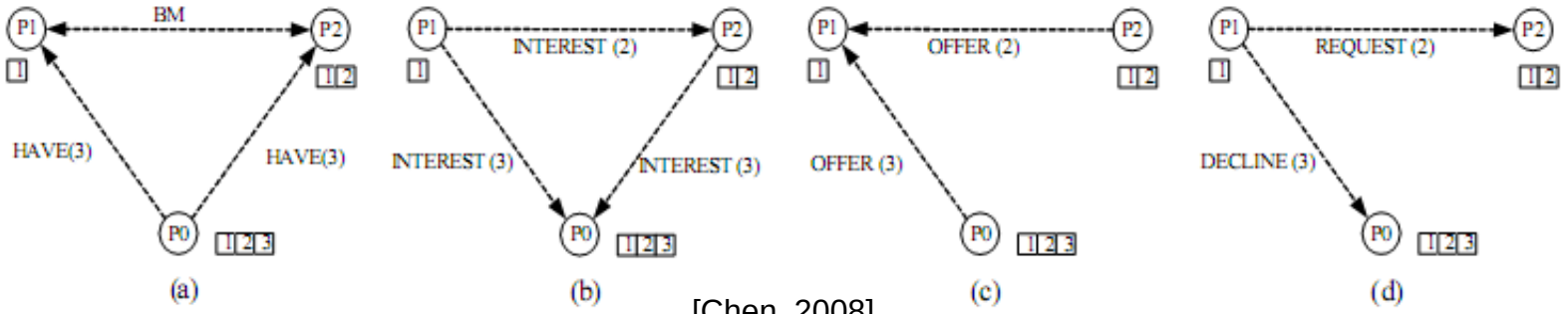


[Guo, 2008]

- Server need to contribute more bandwidth
- More suitable for on-demand video
- Lower hierarchy reduce latency
- Less scalable
- Peer churn & failure can affect the continuity

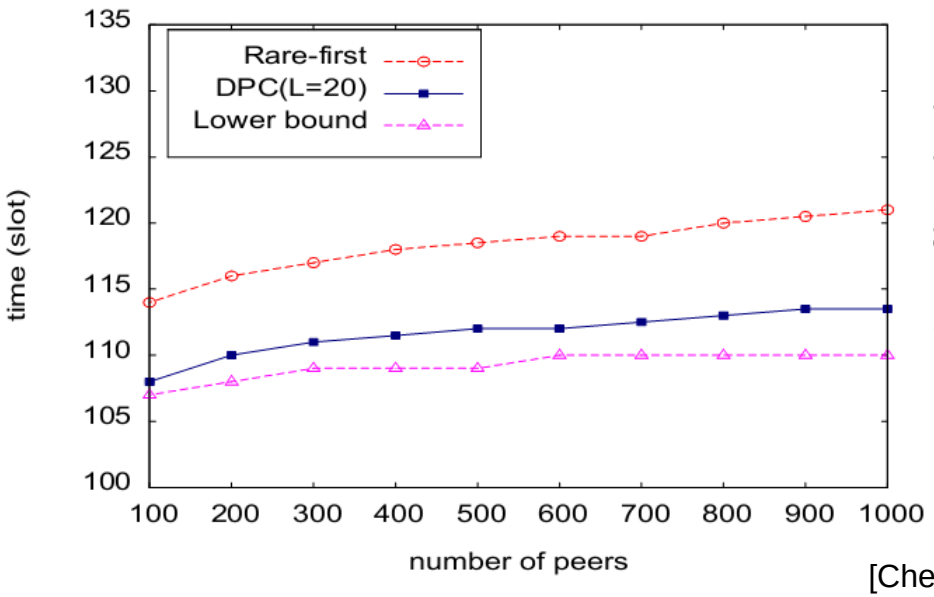
Earliest Deadline First

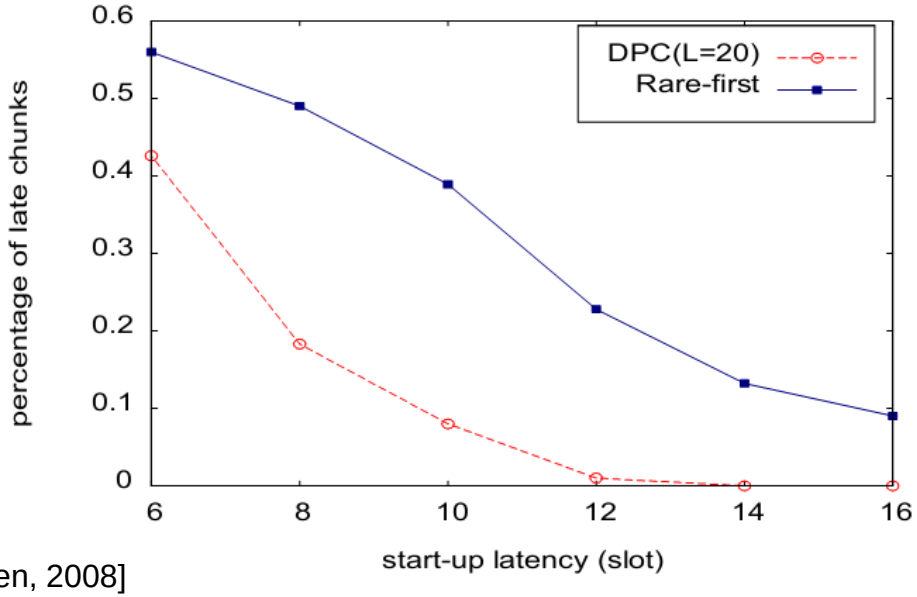
- Objectives – Minimum playback delay & continuity
- Rule 1
  - Chunk with the lowest sequence number has the highest priority
  - So request chunk with the lowest sequence number
  - Try to meet earliest deadline
- Rule 2
  - Peer with the lowest, largest sequence number in buffer map has the highest priority
  - Falling behind, so let it seed up



[Chen, 2008]

Earliest Deadline First (cont.)

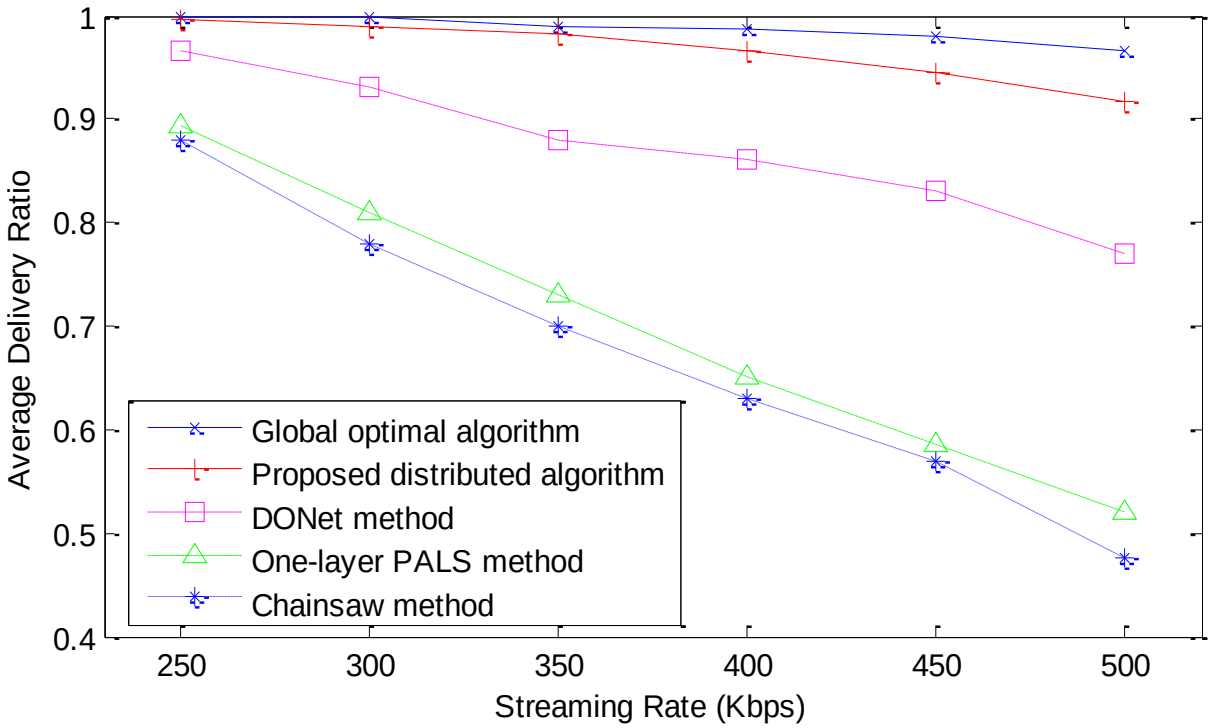




[Chen, 2008]

- DPC – Distributed Priority based Chunk scheduling
- *L* – Number of partners
- Lower playback delay
- Lower skipping

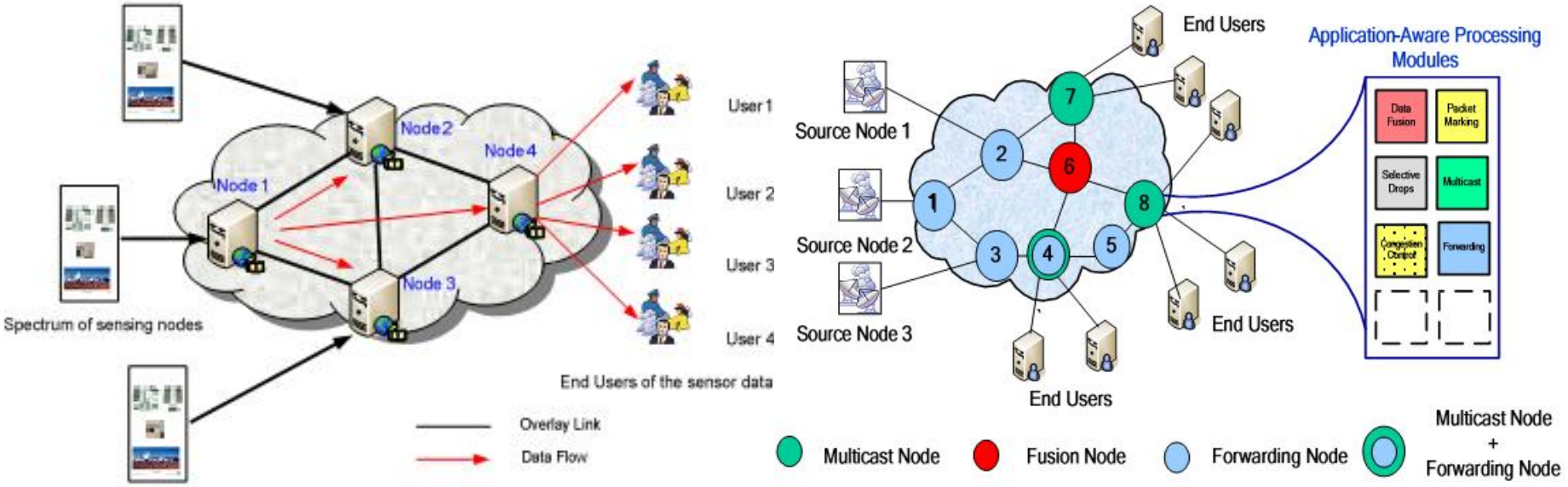
Hybrid Chunk Scheduling



[Zhang, 2006]

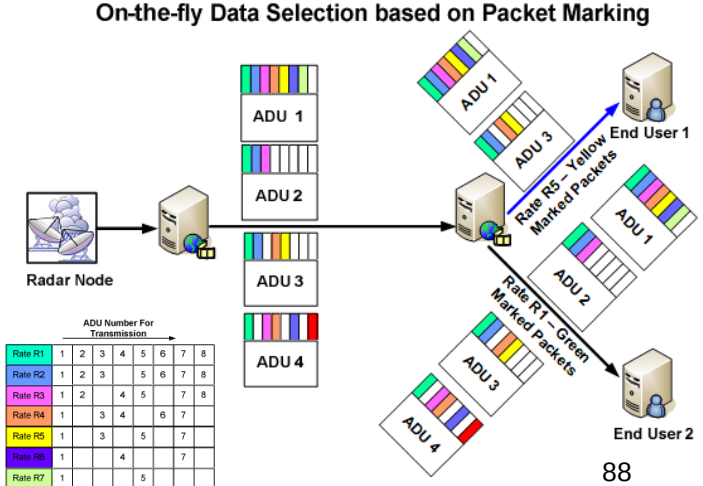
- Combine both earliest deadline & rarest first
- Lower delay than CoolStreaming & Chainsaw
- Lower skipping

Application-Aware Radar Networking



[Banka, 2007]

- Application-aware overlay networks
  - Application-aware packet marking & streaming
- In-network data fusion
- API for application-aware service deployment
- Data-fusion latency estimation



[Lee, 2006]

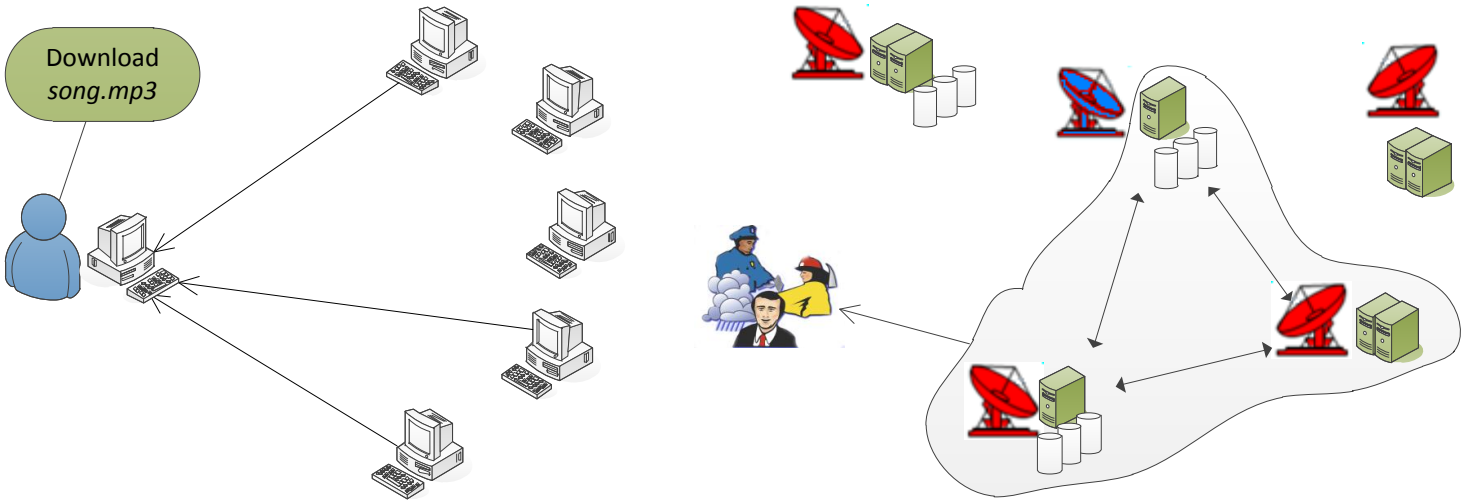
Challenges & Opportunities

Challenges	Opportunities
QoS & QoE • Peer churn & failure • 30% of users leave overlay within 3 minutes [Tang, 2007] • Asymmetric bandwidth	Best effort → deterministic performance • Real-time & VoD streaming • Minimizing skipping & start-up delay • Adaptive network formation & routing • Integrating social networks
Heterogeneous devices	Supporting • Different video qualities • Scree sizes • Processing, memory, & bandwidth • Coding designed for P2P
Digital rights management	Distributing certificates/keys • Pay-per-view, VoD, ads

Outline

- File sharing
  - Unstructured vs. structured overlays
  - Performance enhancements
    - More state, caching, replication
  - Opportunities & challenges
- Streaming
  - Tree-push vs. mesh-pull
  - Opportunities & challenges
- Resource sharing
  - Collaborative P2P
  - Resource aggregation
  - Opportunities & challenges

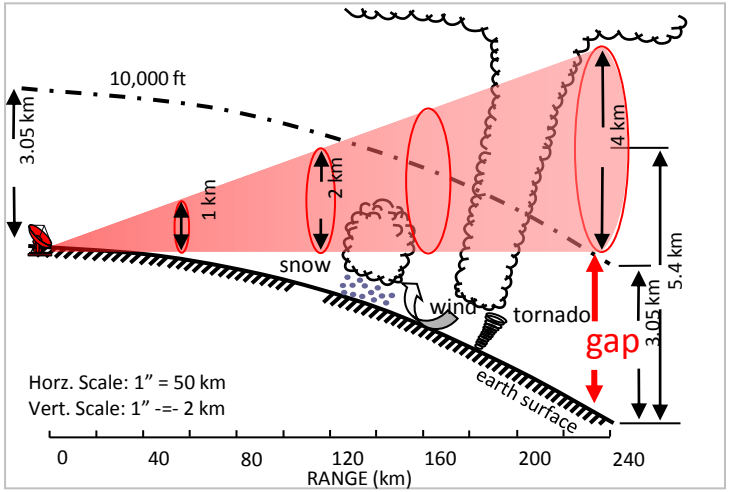
Collaborative P2P Systems

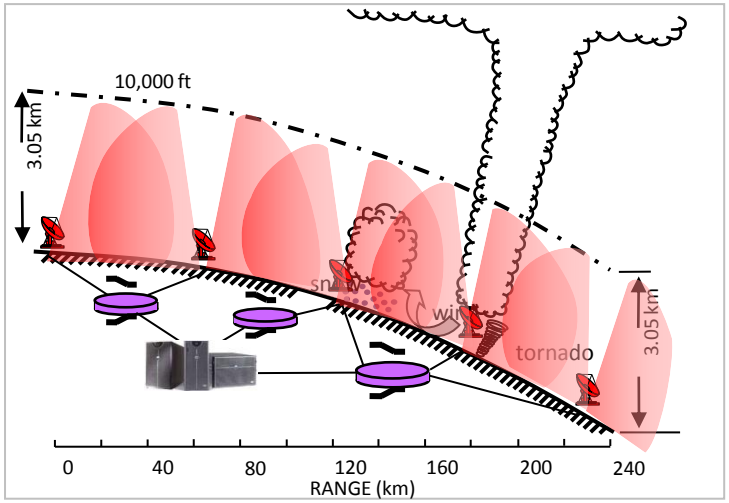


- About interaction of groups
- Aggregate group(s) of resources
  - Diversity in resources & capabilities is an asset
- Can accomplish greater tasks – beneficial to all peers
- Many applications
  - DCAS (Distributed Collaborative Adaptive Sensing), P2P clouds, GENI (Global Environment for Network Innovation), mobile P2P, social networks

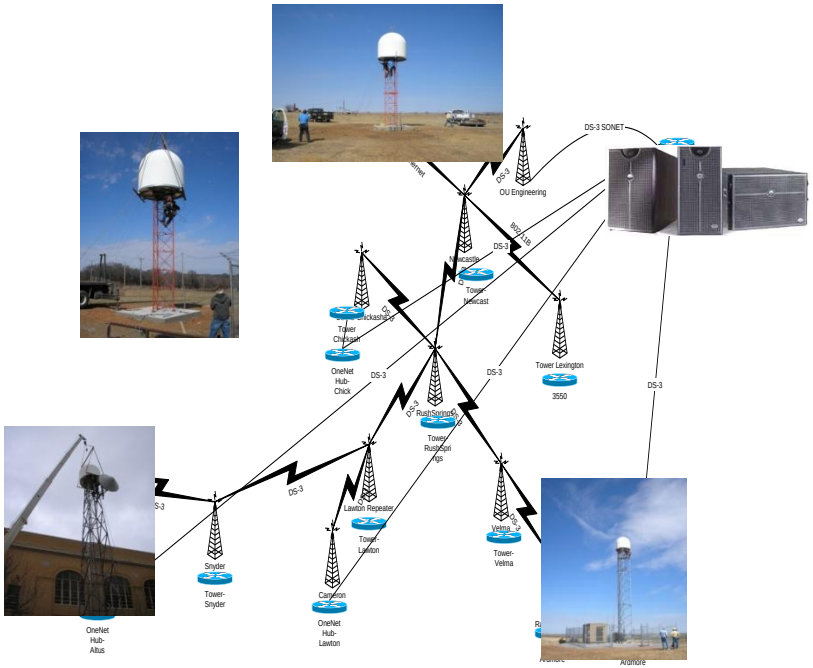
Collaborative Adaptive Sensing of the Atmosphere (CASA)

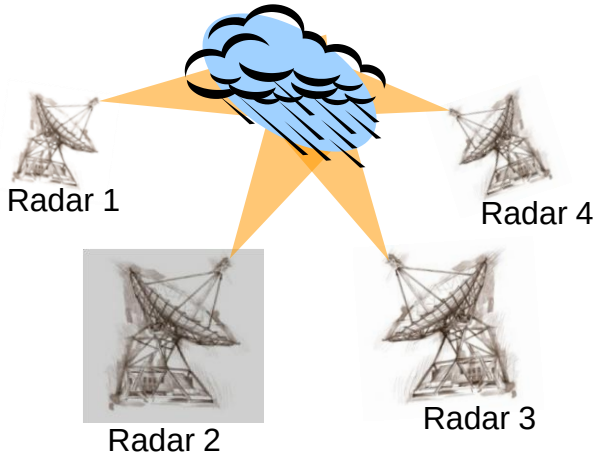
- Distributed Collaborative Adaptive Sensing (DCAS) system
- Concept
  - A network of small radars instead of one large radar
  - Sense lower 3 km of atmosphere
  - Collaborating & adapting radars
    - Improved sensing, detection, & prediction
- CASA goal
  - Improve warning time & forecast accuracy for hazardous weather

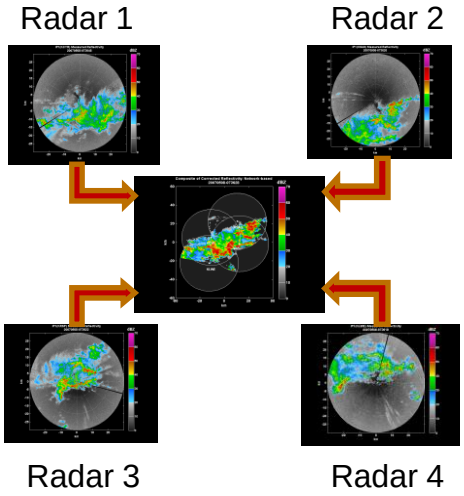




CASA Oklahoma Test Bed

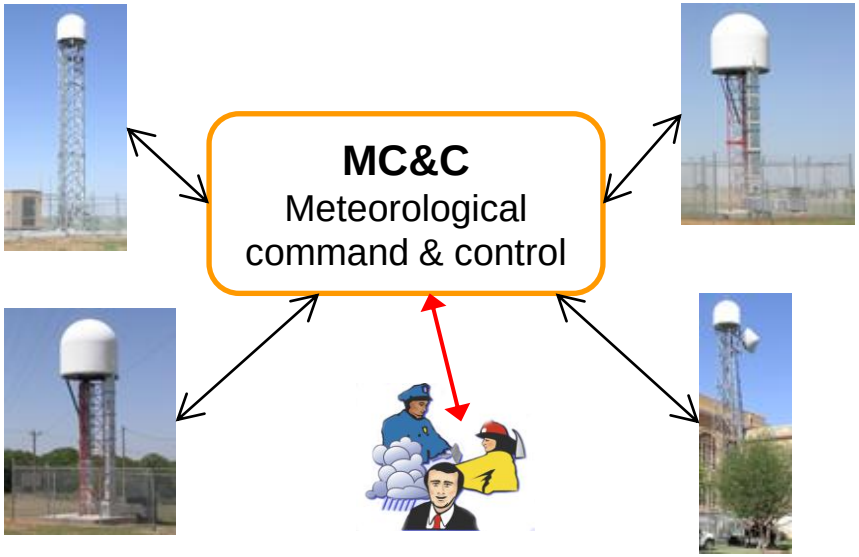


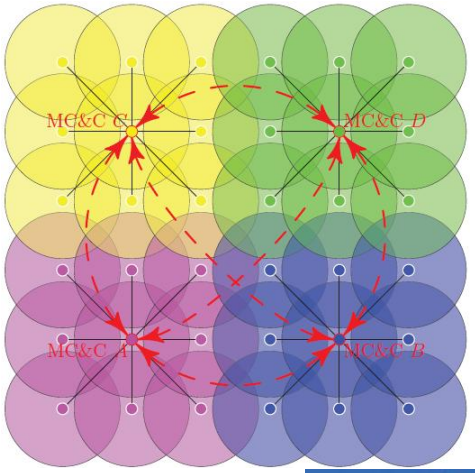




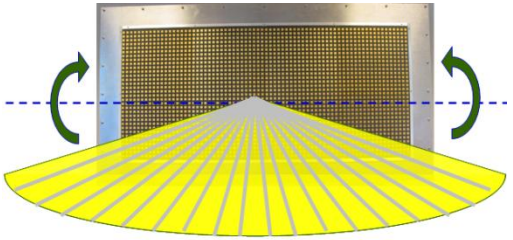
- Multiple high bandwidth streams
- Real-time communication
- Simultaneous observations by multiple radars
- Multi-sensor data fusion
- Heterogeneous infrastructure & end users
- Hostile weather conditions

Large-Scale CASA Deployments

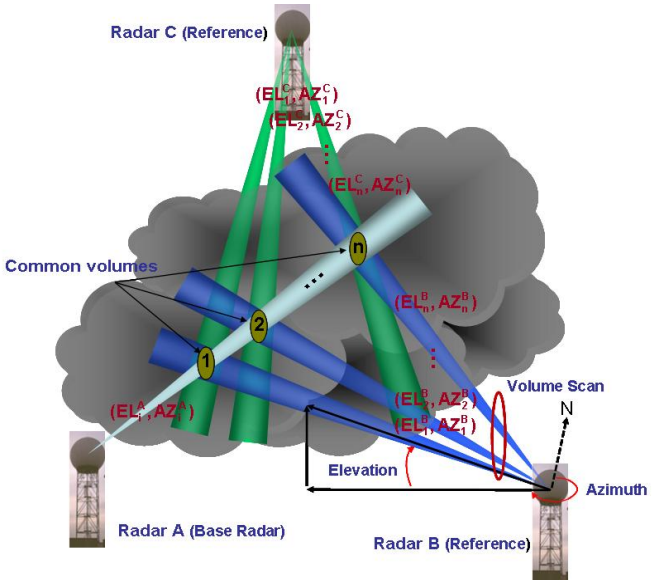


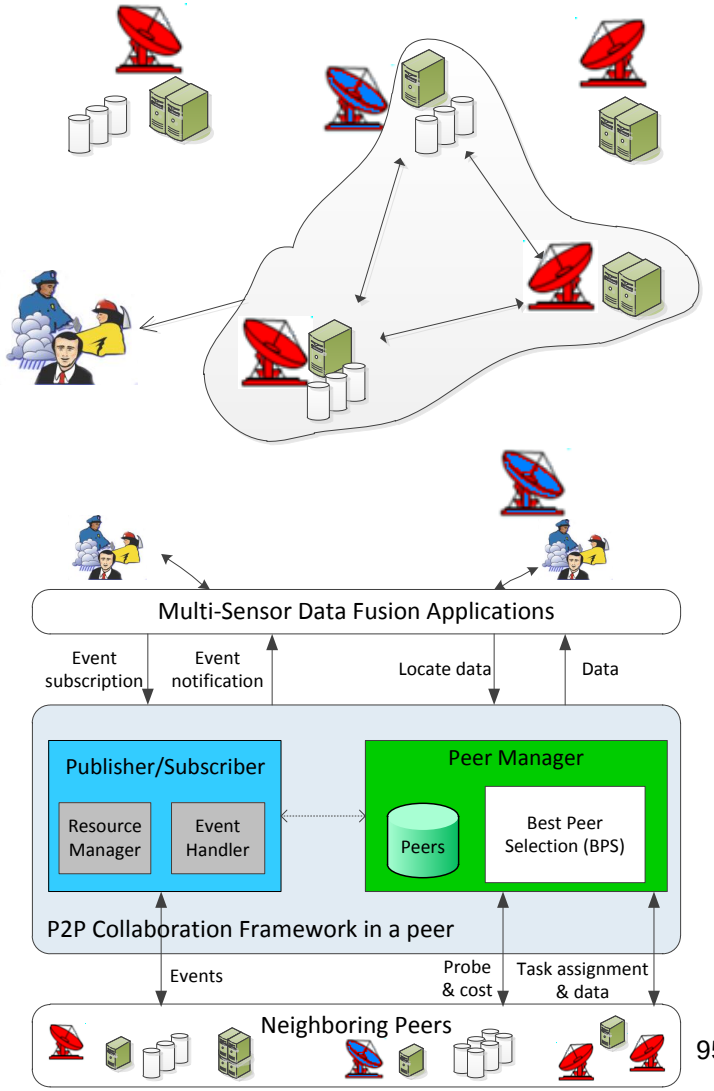


- Large-scale CASA deployments are lot more computation, bandwidth, & storage intensive
- New solid-state radar data rates in Gbps
- Distributed & heterogeneous resources
- Increased resource utilization



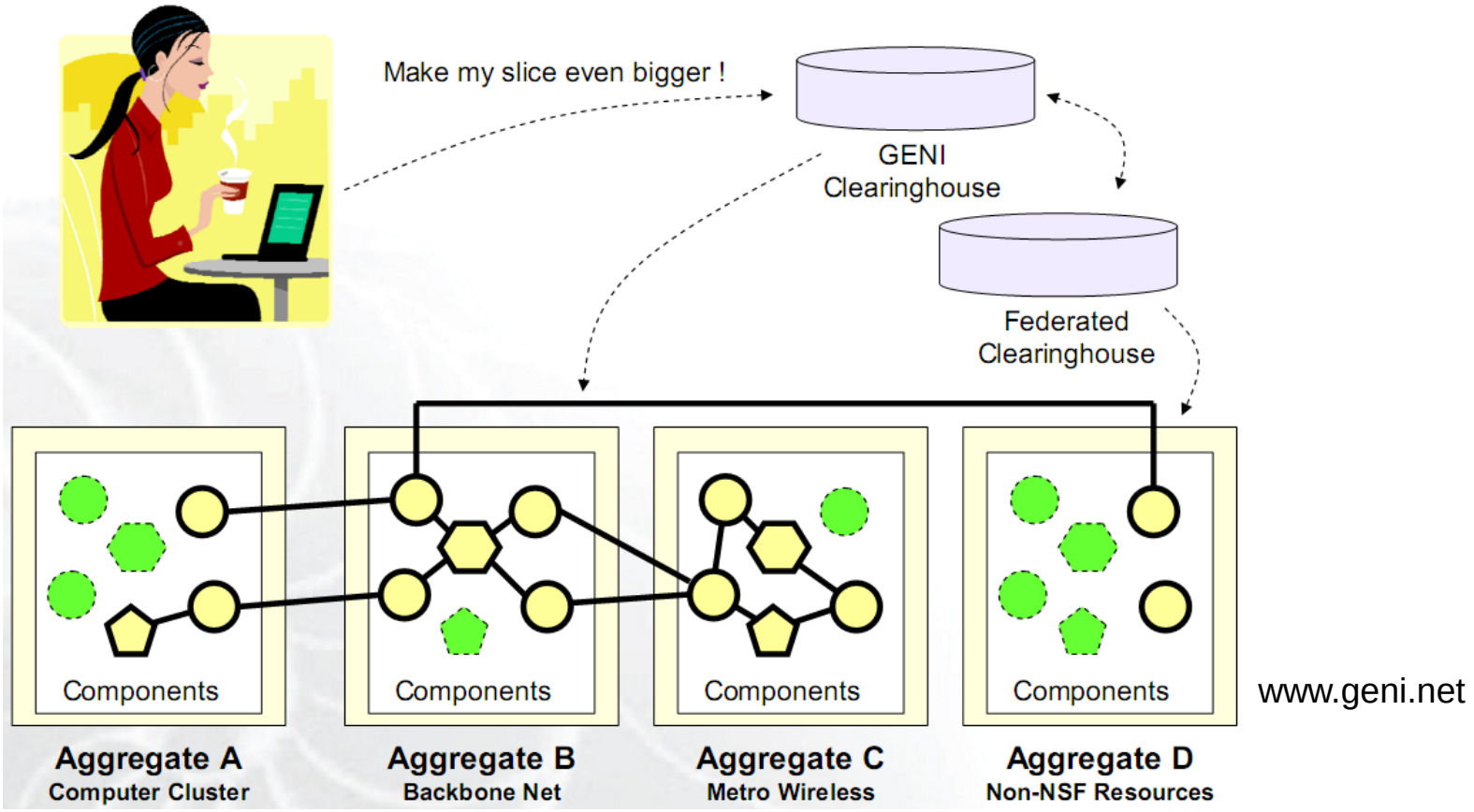
CASA (cont.)





- Groups of multi-attribute resources
  - Radars/sensors, processing, storage, scientific algorithms
  - Heterogeneous, dynamic, & distributed
- Need to aggregate groups of resources as and when needed

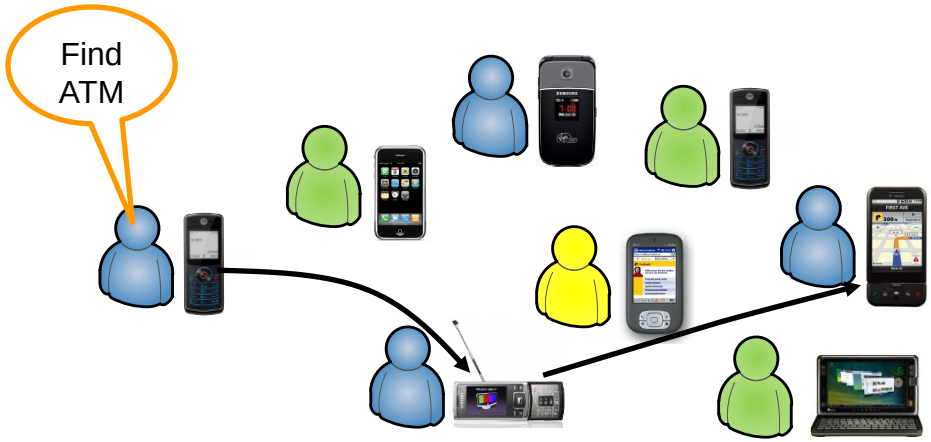
Global Environment for Network Innovations (GENI)



- Collaborative & exploratory platform for innovation
- Aggregating groups of resources across multiple administrative domains

Other Applications

Community (P2P) clouds

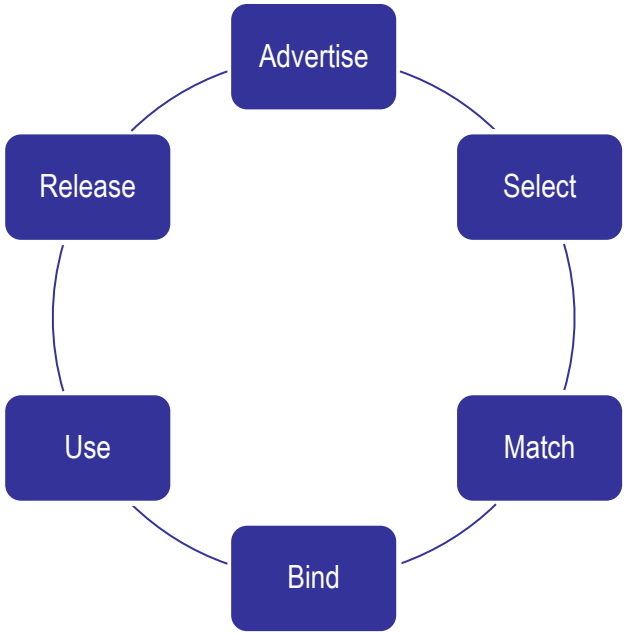


Mobile P2P

- Depends on some form of resource aggregation
  - Multi-attribute, heterogeneous, dynamic, & distributed

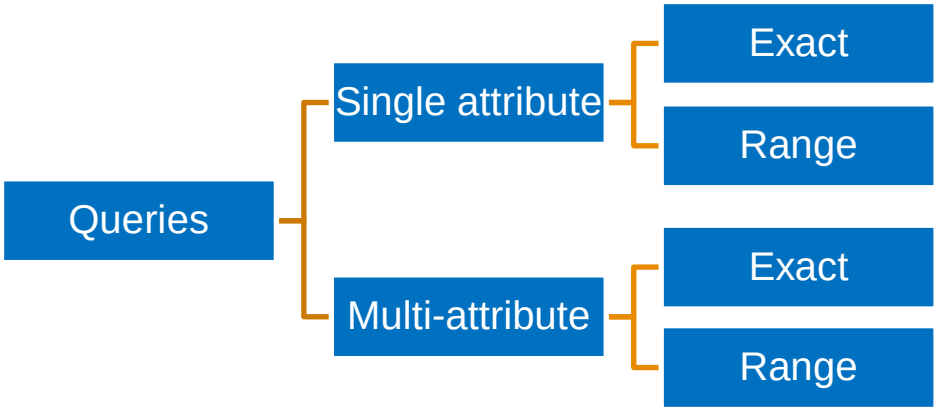
Multi-Attribute P2P Resource Aggregation

- Phases of resource aggregation
  - Advertise resources
    - Attributes & usage constrains
  - Select best resources
  - Match resources
    - Bandwidth, latency, packet loss, neighborhood (avoid ISP)
  - Bind to resources
    - Agreement between user & resource
  - Use resources
  - Release
    - Task complete or decreased demand
- Process continues
  - Demand increases
  - Overcome/use fail/new resources



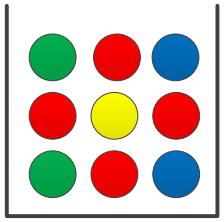
[Bandara, 2011d]

Multi-Attribute Queries



- Specify multiple attributes & range of attribute values
  - *"Find 2 nodes"*
  - *"Find 2 Linux nodes"*
  - *"Find 2 nodes with CPU ≥ 2.0 GHz and 256 ≤ Memory ≤ 512 MB and OS="Linux 2.6" and Latency ≤ 50 ms"*
- May also specify constraints
  - *"Find 2 GHz ×86 CPU: available between 12:00am-6:00am to my friends and average utilization must be ≤ 60%"*

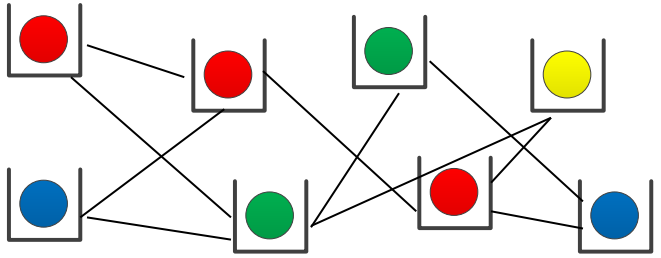
Resource Discovery – Unstructured P2P



Centralized

$O(1)$

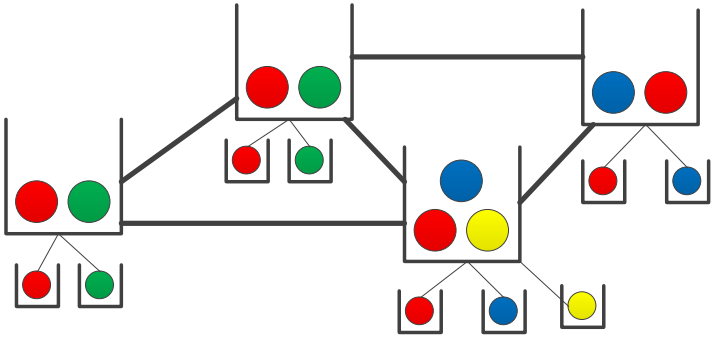
Single point of failure



Unstructured

$O(hops_{max})$

Not guaranteed to find resources



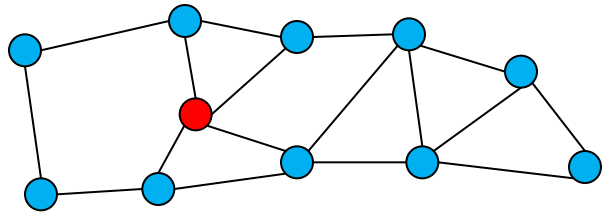
Superpeer

$O(hops_{max})$

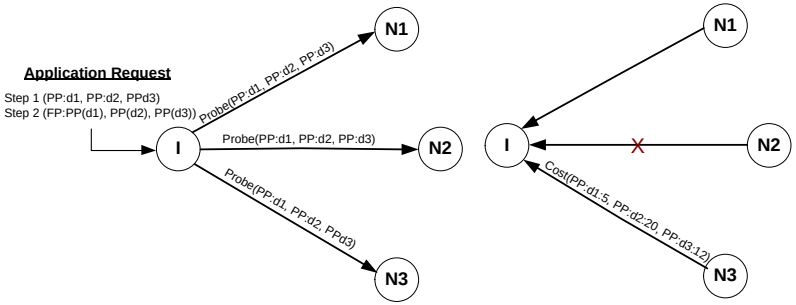
Not guaranteed to find resources

Unstructured P2P (cont.)

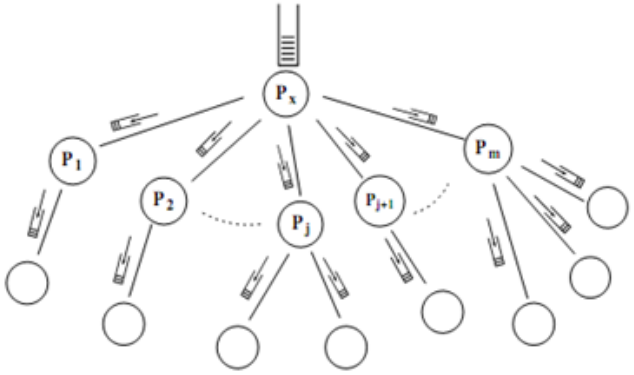
- Random walk
  - Superpeer overlay
  - Pros – low overhead, accurate state
  - Cons – no guarantees, high latency
- Broadcast
  - Best peer selection [Lee; 2007]
  - Expanding tree [Yao, 2006]
  - Pros – accurate state
  - Cons – high overhead, not scalable
- Gossiping
  - Agents carry resource information [Kwan, 2010]
  - Pros – low overhead, large coverage
  - Cons – stale data, no guarantees



Random walk



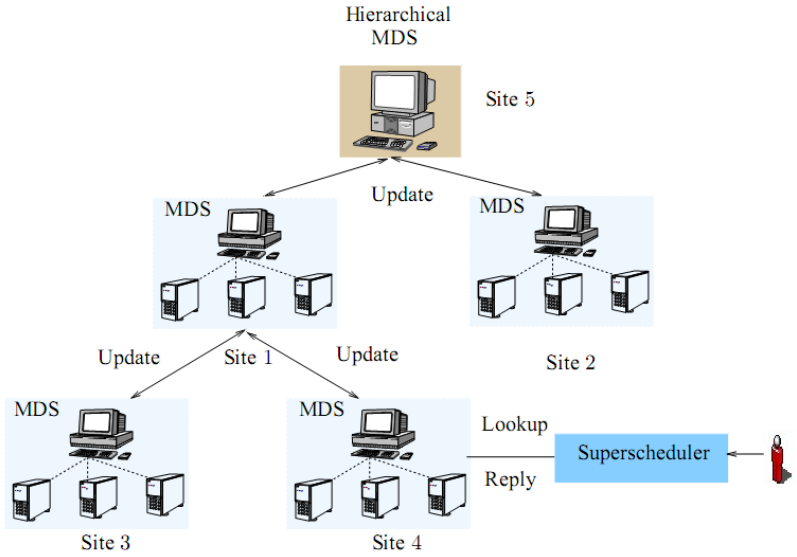
Best peer selection



Expanding tree

Unstructured P2P (cont.)

- Report to specific nodes
- Centralized
  - Report to a known location
    - E.g., GENI clearing house
  - Pros
    - Accurate state, guaranteed, low adv/query cost
  - Cons
    - Single point of failure, not scalable
- Hierarchical
  - Report to local repository
  - Which in turn report to a regional one
    - e.g., GENI federated clearing house



[Ranjan, 2008]

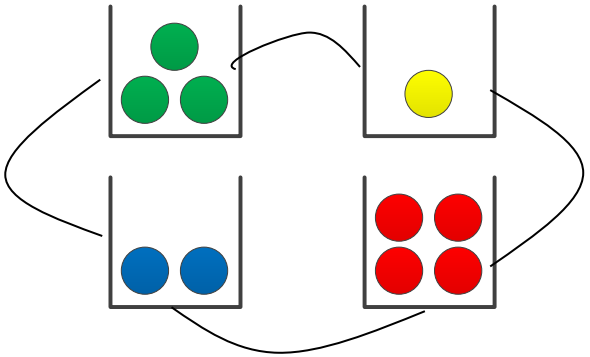
UPnP (Universal Plug & Play) – upnp.org

- Pervasive P2P network connectivity across
  - PCs, mobile phones, TVs, intelligent appliances, sensors, actuators
- Data sharing, communication & control
  - Expressed using XML
  - HTTP & TCP/IP for direct communication
- Facilitates collaborative P2P applications within
  - Home, office, & everywhere in between

wordpress.com

www.orbitmicro.com

Resource Discovery – Structured P2P



Distribute Hash Table (DHT)

$O(\log N)$

Guaranteed performance

Not for dynamic systems

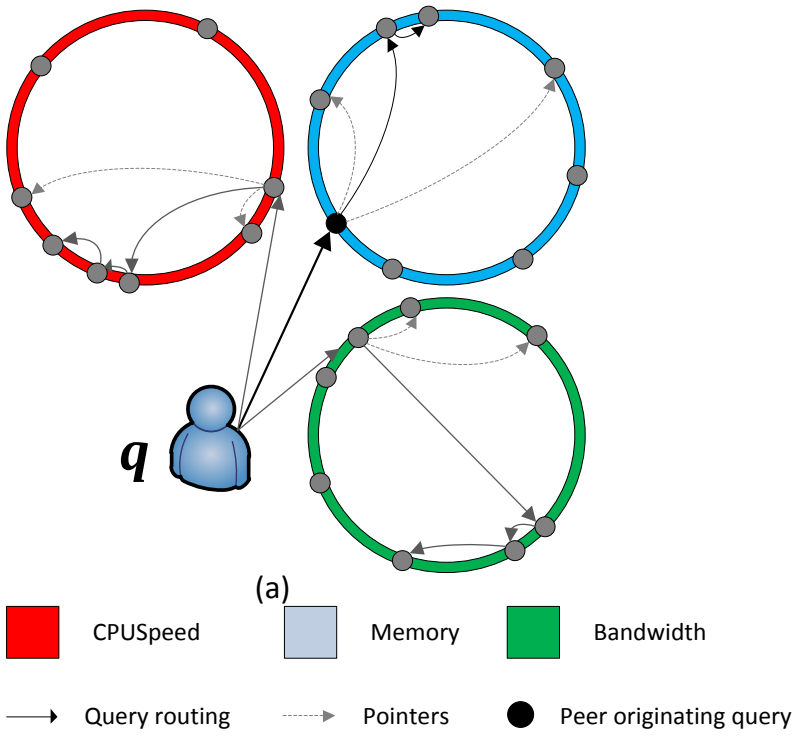
Resource Discovery – Multiple Rings

- Separate ring for each attribute
  - Mercury [Bharambe, 2004]
  - Locality preserving hashing
    - Map attribute values to nearby nodes
    - $(v - v_{min})/(v_{max} - v_{min})$

1. Multiple sub-queries
   - Go to $c_1/m_1/b_1$
   - Then go from $c_1/m_1/b_1$ to $c_2/m_2/b_2$ using successors
   - Finally, a database-like join
   - Total cost $O(N)$

$$C_{TOTAL} = \sum_{r \in R} C^r_{ADV} + \sum_{q \in Q(t)} C^q_{QUE}$$

$$C^q_{QUE} = \sum_{i \in q_A} \left(h_i + \left\lceil \frac{r^i_q}{r^i_{\max}} \times \frac{N}{A} \right\rceil - 1 \right)$$

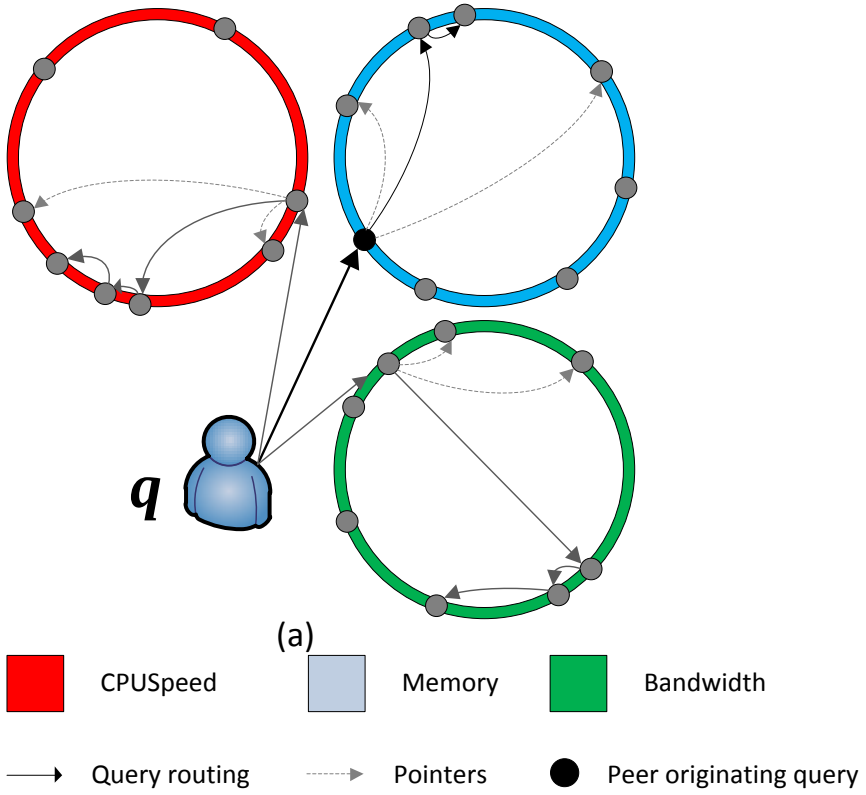


$$query = c_1 \leq CPU \leq c_2,\ m_1 \leq Memory \leq m_2,\ b_1 \leq BW \leq b_2$$

Multiple Rings (cont.)

2. Single-Attribute Dominated Queries (SADQ)
   - Advertise attributes to all rings
   - Pick $min(c_1 - c_2, m_1 - m_2, b_1 - b_2)$
   - Search that ring
   - Query stop as soon as desired no of resources are found
   - Low query cost
   - High advertising cost

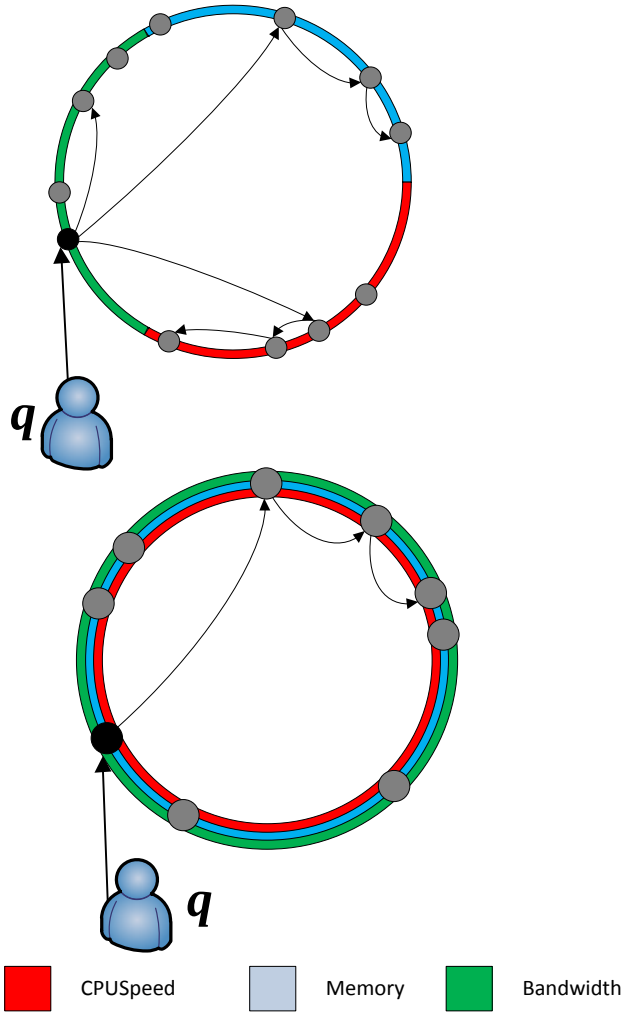
- Pros
  - Support new attributes
- Cons
  - Many routing entries
  - Load balancing issue



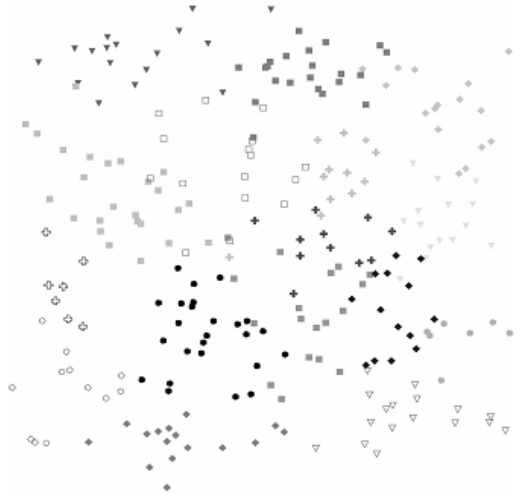
$$query = c_1 \leq CPU \leq c_2, m_1 \leq Memory \leq m_2, b_1 \leq BW \leq b_2$$

Single Ring

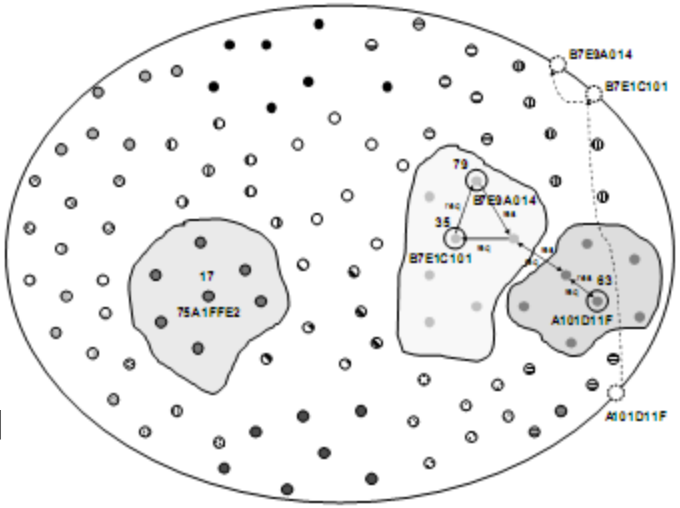
- Single-partitioned ring
  - LORM [Shen, 2007]
  - Sword [Albrecht, 2008]
  - Pros
    - Few routing entries
  - Cons
    - Hard to add new attributes
    - Load balancing issue
- Single-overlapped ring
  - MAAN [Cai, 2004]
  - Pros
    - Few routing entries
    - Relatively better load distribution
  - Cons
    - Easy to add new attributes



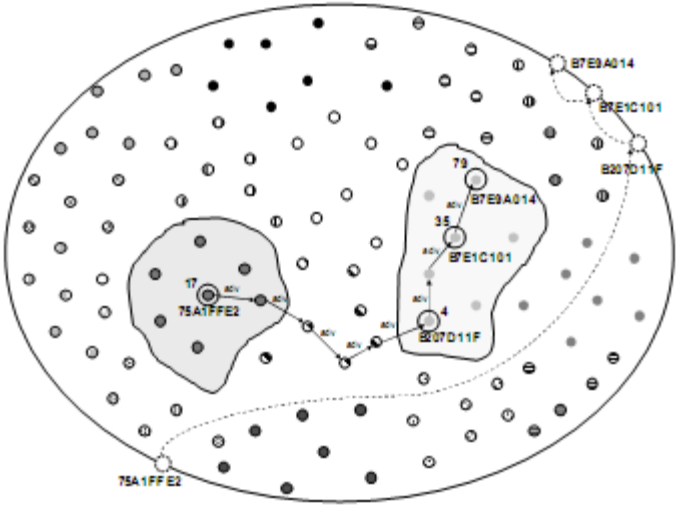
Resource Discovery in MANET



MADPastry [Zahn, 2005]

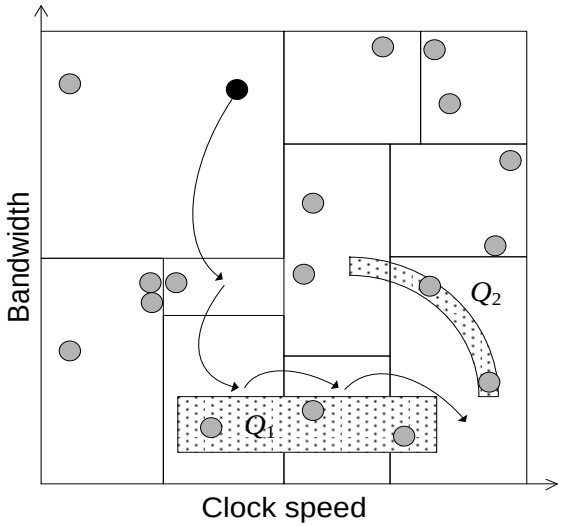


- Group nodes based on landmarks
- Mapped to a partitioned-ring
- Advertise resources to
  - Nodes within own landmark
  - Global address on ring
  - Can locate nearby resources
  - Reduce latency & hops
- Doesn't work with many attributes



D-Torus – MURK [Ganesan, 2004a]

- Map attribute values to a *d*-torus
- Partition *d*-torus to zones
  - A peer is responsible for a zone
  - Track as a kd-tree
- Index in appropriate zone
- Greedy routing of queries
  - Parallel search on neighboring zones
  - Results are send to query originator
  - Finally, database-like join
  - Cons - high query cost
- Also, mapped to Chord using space filling curves
  - Cons – loose locality, cost is $O(N)$

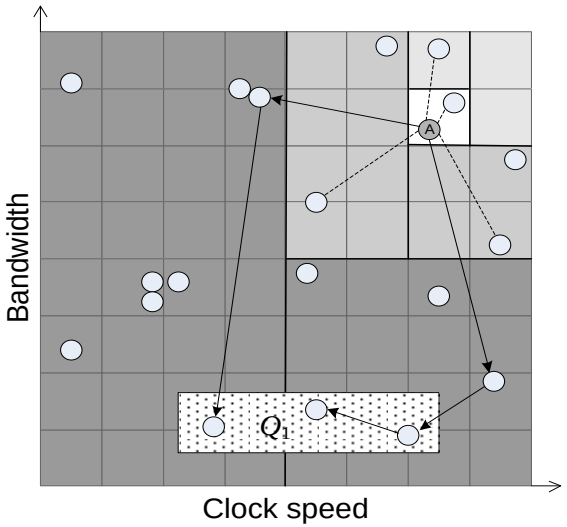


MURK - MUlti-dimensional Rectangulation with *Kd*-trees

D-Torus – Resource-Aware Overlay

[Costa, 2009]

- DHTs can't track dynamic attributes correctly
- Use only static attributes
- Form overlay by connecting peers based on
  - Partition torus into hierarchical cells
  - Keep a pointer to a node in each level in hierarchy & cell
    - Identified using gossip protocol
- Query resolution
  - Depth-first search starting from lowest level cell
- Cons – high latency & support only static attributes



Resource Selection, Match, & Bind

- Select
  - Supported by all solutions
- Match
  - Need access to multiple resources
  - Superpeers index multiple resources
  - Sword support latency & bandwidth as AS level
  - MADPastry based on locality & latency
- Bind
  - Need access to resource
  - Superpeers, unstructured P2P, & resource-aware overlay
- No single solution support all 3 requirements

Summary of Structured P2P Solutions

Scheme	Architecture	Routing mechanism	Lookup overhead (point query)*	Lookup overhead (range query)*	Routing table size*	Load balancing
Mercury	Multiple rings	Successor & long distant links	$O(^1/_k \log^2 n)$	O(n)	k + 2 per ring	Dynamic
LORM	Partitioned ring	Cycloid	O(d)	O(n)	O(1)	Static
MADPastry	Partitioned ring (locality based)	Pastry	O(log n)	O(n)	O(log l)	Static
MAAN	Single ring	Chord	O(log n)	O(n)	O(log n)	Static
MURK	d- torus	CAN with long distance links	$O(\log^2 n)$	O(n)	2d + k	Dynamic
SWORD	Partitioned ring, resource matching	Chord	O(log n)	O(n)	O(log n)	Static
Resource-aware overlay	d- torus partitioned into cells	Links to peers in other cells	O(n)	O(n)	O(d)	Static

***** *n* – number of peers in overlay, *k* – number of long distant links, *d* – number of dimensions, *D* - network diameter, *l* – number of landmarks

Summary of All Solutions

Scheme	Architecture	Advertise	Discover	Select	Match*	Bind*
Flooding	Flood advertisements or queries	Yes	N/A	Guaranteed	N/S	When queries are flooded
Gossiping	Agents share resource specifications they know	Yes	Yes	Moderate probability of success	Simple matching	N/S
Random walk	Agents carry resource specifications & queries	Yes	Yes	Moderate probability of success	Simple matching	When query agents are used
Superpeer	2-layer overlay	Yes	Yes	Relatively high probability of success	Simple matching	Yes
Mercury	Multiple rings	Yes	N/A	Guaranteed	N/S	N/S
LORM	Partitioned ring	Yes	N/A	Guaranteed	N/S	N/S
MADPastry	Partitioned ring (based on locality)	To local & neighbor partitions	N/A	Guaranteed	Latency & hop count	N/S
MAAN	Single ring	Yes	N/A	Guaranteed	N/S	N/S
MURK	d- torus	Yes	N/A	Guaranteed	N/S	N/S
SWORD	Partitioned ring, resource matching	Yes	N/A	Guaranteed	Yes	N/S
Resource-aware overlay	d- torus partitioned into cells	Static attributes	N/A	Guaranteed	N/S	Yes

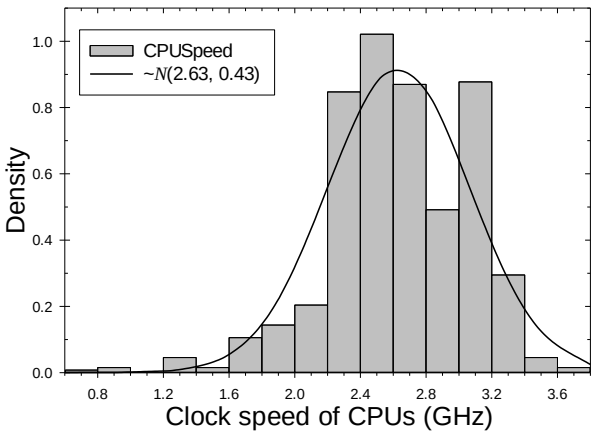
Resource & Query Characteristics

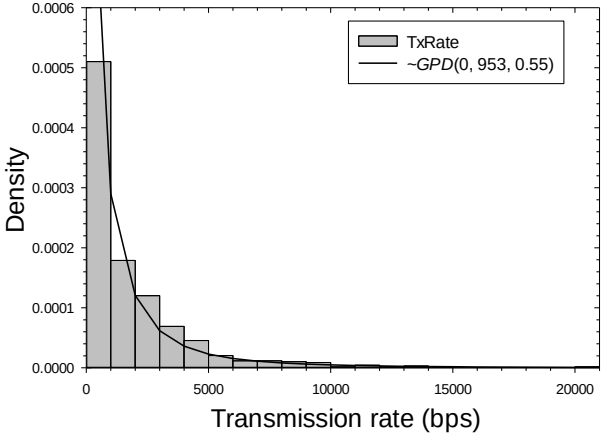
- Resources & queries are characterized by multiple attributes
  - Need detailed understanding to design, optimize, & validate
- No formal analysis → Many simplifying assumptions
  - Few attributes
  - Ignore cost of updating dynamic attributes
  - i.i.d attributes
  - Uniform/Zipf's distribution of resources/queries
  - Queries specifying a large number of attributes & a small range of attribute values
- Leads to inaccurate designs, performance analysis, & conclusions

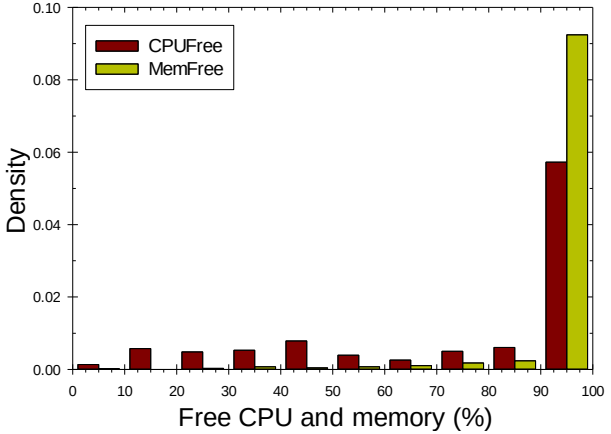
Datasets

- PlanetLab node data
  - Global research network for developing new network services, protocols, & applications
  - Reflects many characteristics of Internet-based distributed systems
    - Heterogeneity, multiple end users, dynamic nodes, & global presence
    - Used to evaluate many preliminary P2P protocols & applications
  - 12 static & 34 dynamic attributes sampled every 5 min
  - 500-700 active nodes
- SETI@home
  - Desktop grid
  - Static resources from 300,000+ nodes
  - 21 static & 4 dynamic attributes

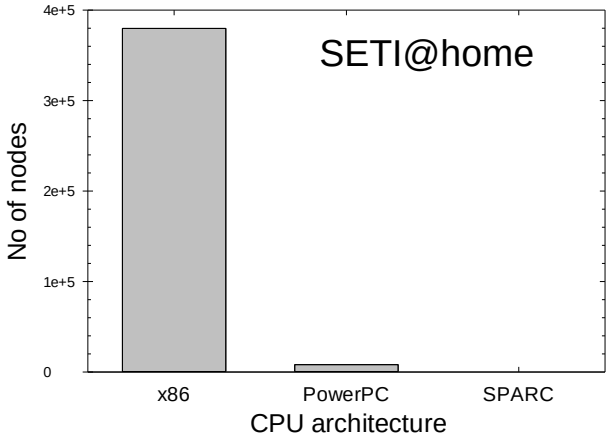
Resource Characteristics



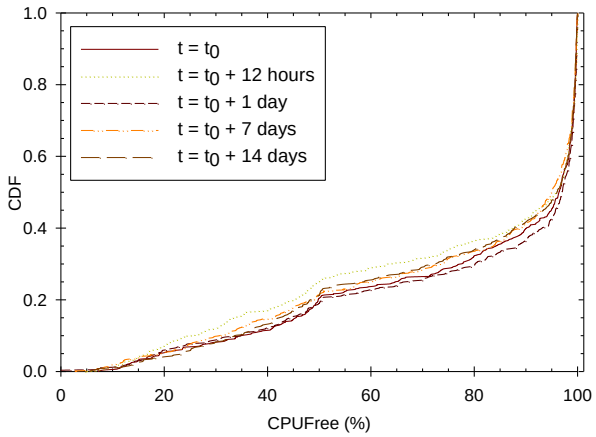


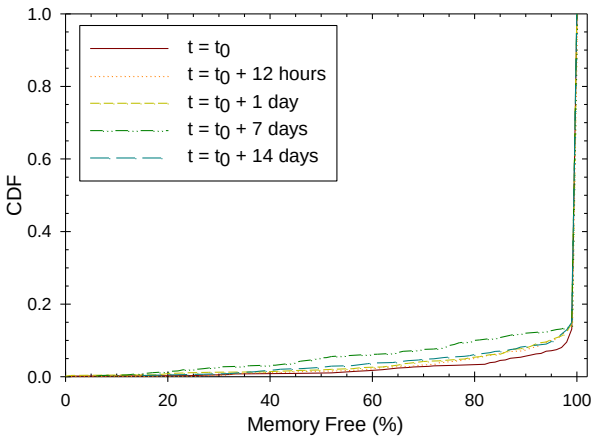


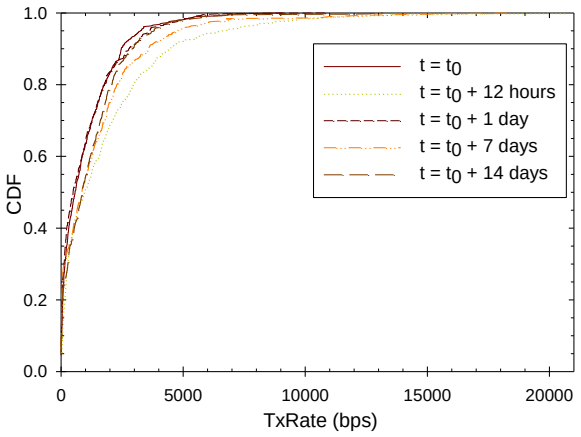
- Resources satisfy a mixture of probability distributions
  - Gaussian – CPUSpeed, MemSize, DiskFree
  - Pareto – TxRate, RxRate
  - Many identical nodes
- Highly skewed distributions
  - CPUFree, MemFree, CPU architecture



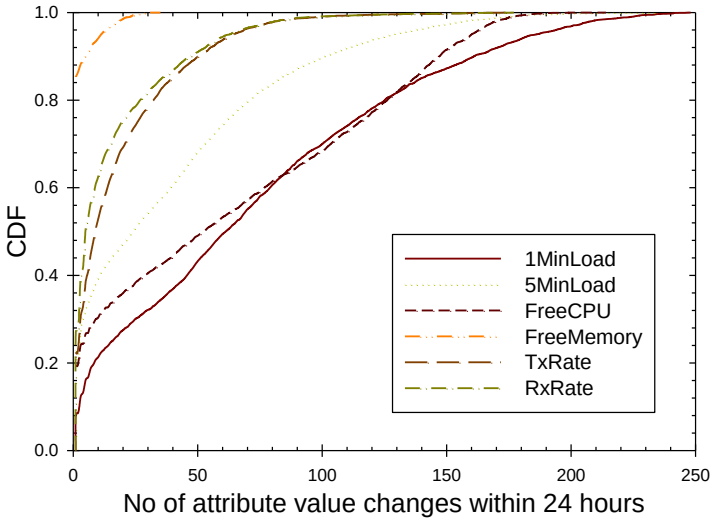
Dynamic Attributes at Different Times







- Distribution of dynamic attributes is stable over days
- Dynamic attributes & their rate of change fits Pareto
  - Same attributes/nodes change frequently
  - Many status updates



Thresholds: *CPUFree* = *MemFree* = ± 10%, *1MinLoad* = ± 2, *TxRate* = *RxRate* = ± 1 Kbps

Resource Characteristics – Correlation

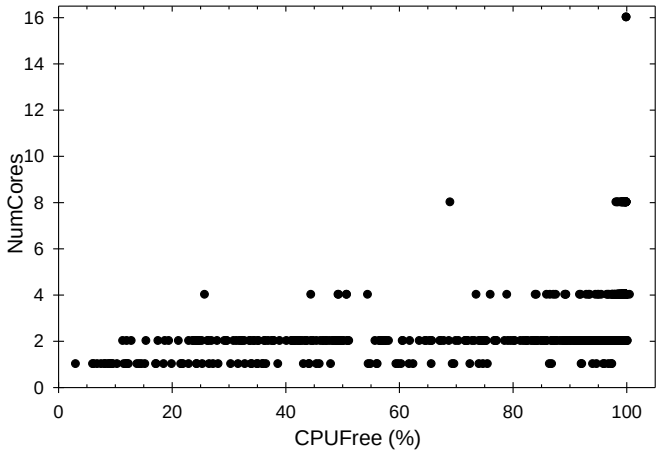
Pearson's correlation coefficient

	CPUSpeed	NumCores	CPUFree	1MinLoad	MemSize	MemFree	DiskFree	TxRate
NumCores	-0.09							
CPUFree	0.02	0.48						
1MinLoad	0.03	-0.31	-0.57					
MemSize	0.06	0.28	0.26	-0.25				
MemFree	0.13	0.21	0.31	-0.35	0.25			
DiskFree	-0.09	0.46	0.37	-0.29	0.54	0.23		
TxRate	0.08	-0.23	-0.26	0.24	-0.12	-0.17	-0.12	
RxRate	0.10	-0.23	-0.30	0.35	-0.13	-0.20	-0.16	0.85

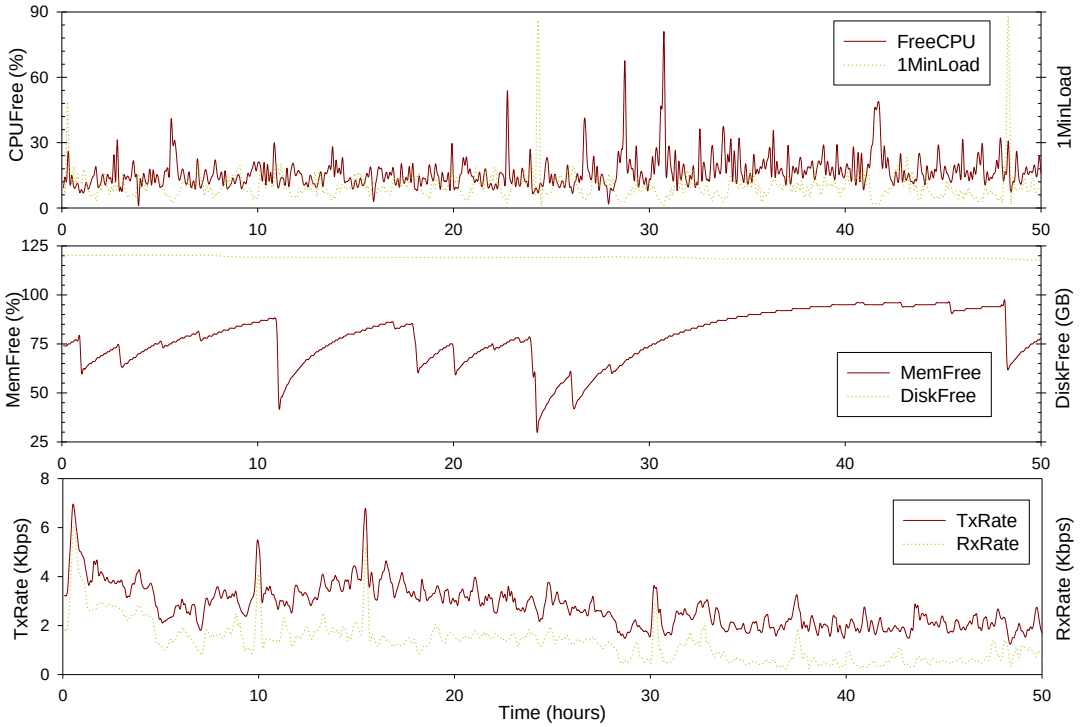
Spearman's ranked correlation coefficient ρ

	CPUSpeed	NumCores	CPUFree	1MinLoad	MemSize	MemFree	DiskFree	TxRate
NumCores	0.04							
CPUFree	-0.07	0.67						
1MinLoad	0.10	-0.42	-0.72					
MemSize	0.03	0.37	0.37	-0.33				
MemFree	-0.07	0.37	0.37	-0.38	0.53			
DiskFree	-0.20	0.60	0.52	-0.41	0.44	0.44		
TxRate	0.06	-0.35	-0.39	0.30	-0.07	-0.20	-0.29	
RxRate	0.07	-0.33	-0.42	0.41	-0.11	-0.21	-0.29	0.86

- Complex correlation among attributes
- Correlation between attributes
  - Static-dynamic
  - Dynamic-dynamic

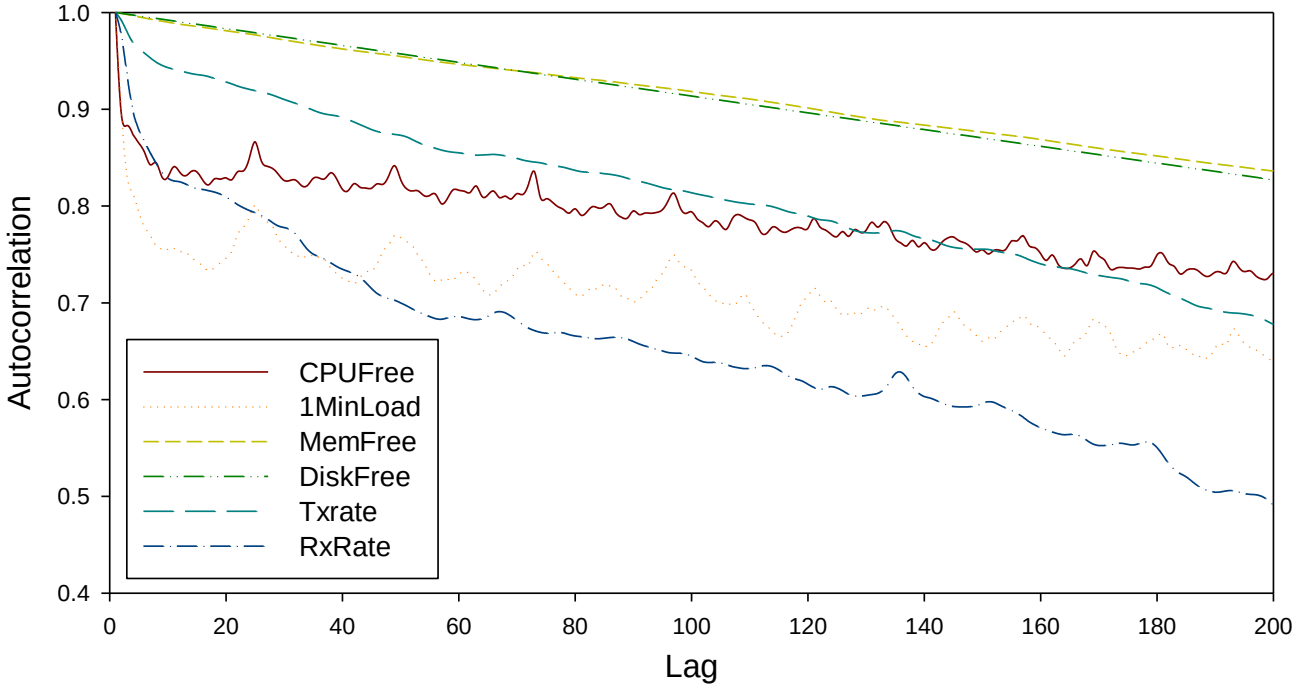


Dynamic Attributes – Contemporaneous Correlation



- Contemporaneous correlation among time series of dynamic attributes
- Specific temporal pattern in MemFree
- Temporal patterns need to be preserved

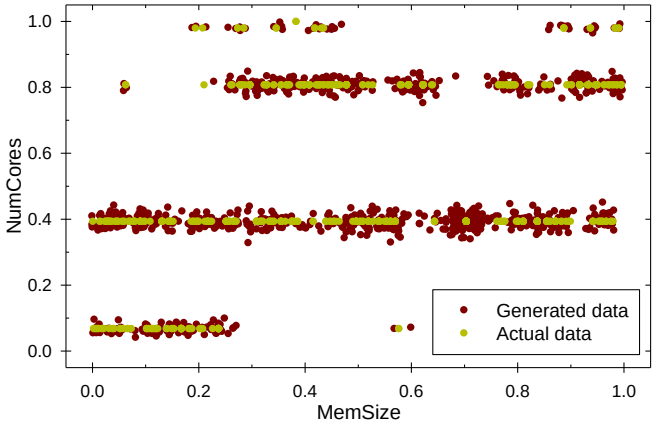
Dynamic Attributes – Autocorrelation



- High autocorrelation in DiskFree & MemFree
- No noticeable change in DiskFree
- Temporal patterns need to be preserved

Modeling Static Attributes

- Need to preserve correlation
  - Attribute values can't be randomly drawn from marginal distributions
  - Pearson's correlation matrix is insufficient
- Copulas capture complex correlations
  - Functions that couple multivariate distributions to their marginals
  - Multivariate joint distribution defined on d-dimensional unit cube s.t. marginal distribution u_i is $\sim uniform(0, 1)$
  - $F(u) = C\left(F_1(u_1), \ldots, F_d(u_d)\right)$
- Empirical copulas support complex/unknown distributions & correlations
  - $C_n\left(\frac{i}{n}, \frac{j}{n}\right) = \frac{\text{No of pairs}(x, y)\,\text{s.t.}\, x \le x_{(i)} \text{ and } y \le y_{(j)}}{n}$
  - $x_{(i)}$ ordered statistics of x
  - No need to find distribution of attributes



Modeling Dynamic Attributes

- Specific temporal patterns in time series → Can't draw values randomly
- Contemporaneous correlation → Can't draw independently
- Goal – Not to predict future behavior, but to generate nodes with similar overall characteristics
  - Not necessary to fit a model
- Build a library of time series segments
  - Pick the most distinct pattern & split according to structural changes
    - Preserve distinct temporal patterns
  - Split other time series at same position & replay segments together

Modeling Dynamic Attributes (cont.)

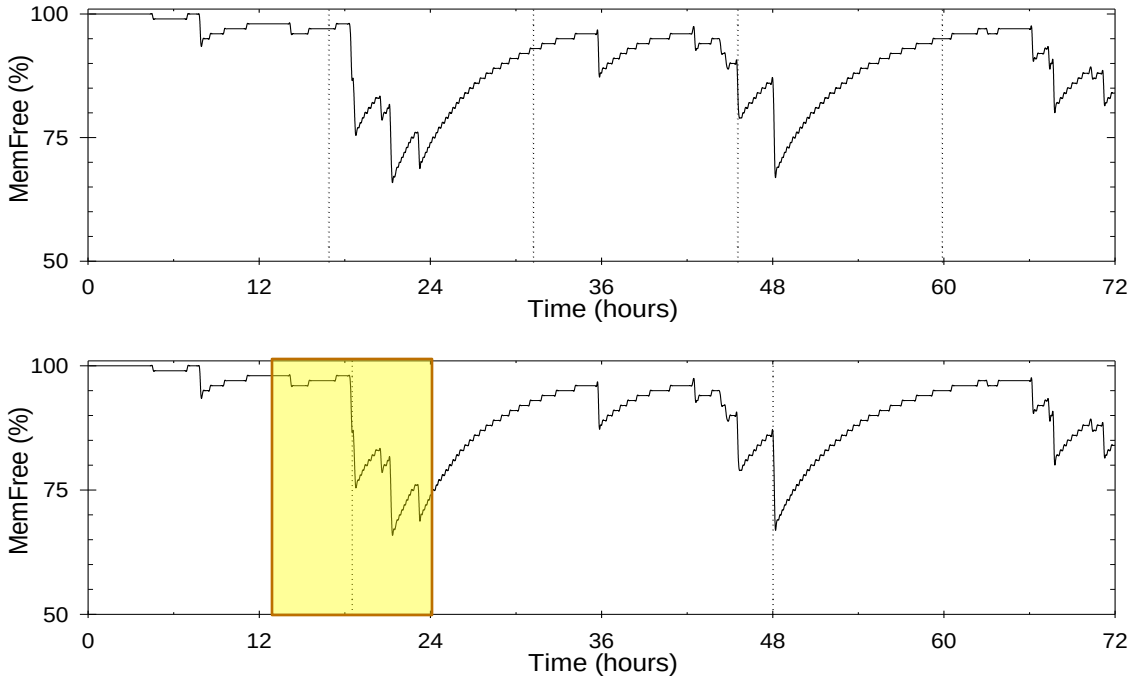
$$y_i = x_i^T \beta_i + u_i \quad i = 1, \ldots, n$$

Check for Null Hypothesis that

$H_0: \beta_i = \beta_0, i = 1, \ldots, n$

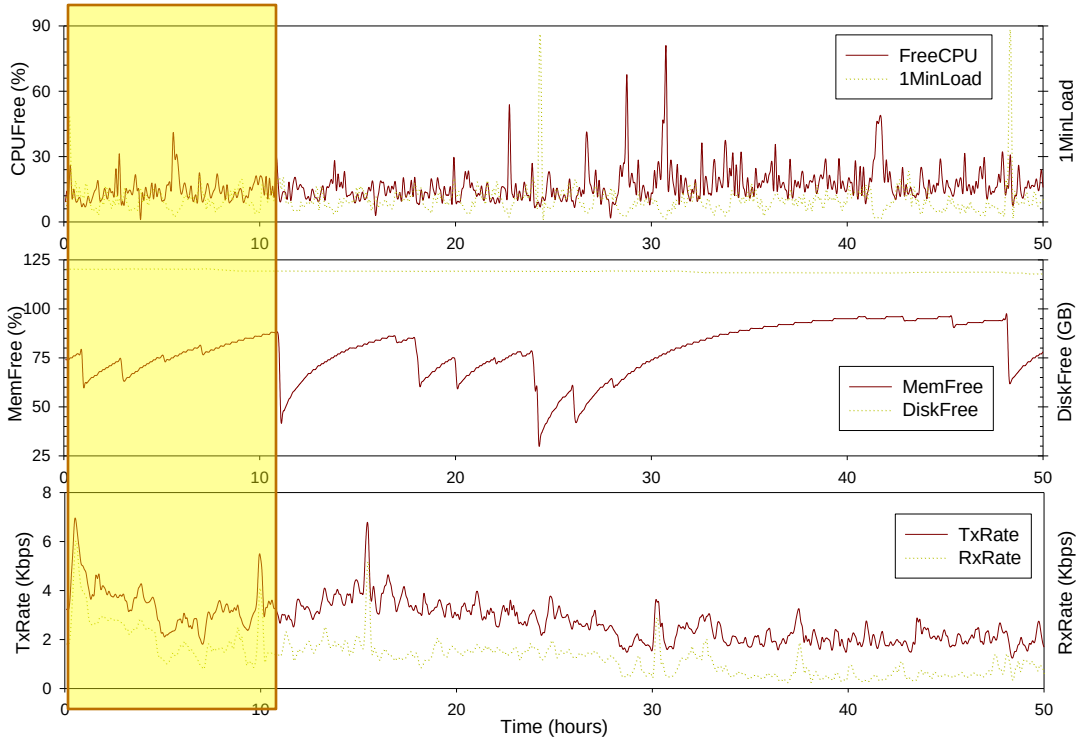
Sliding Window

w = 20, Δ = 30%



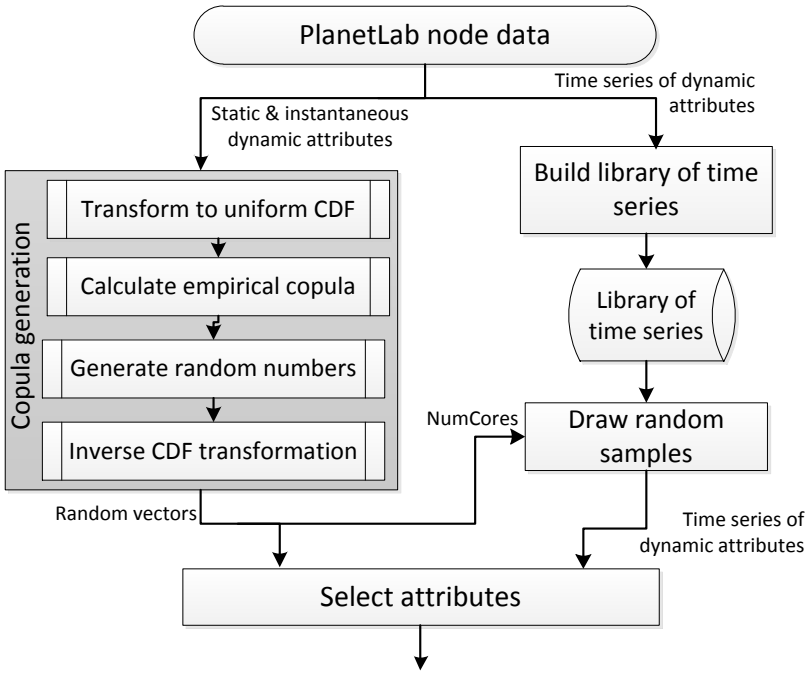
- Initial approach – R *strucchange* package
- Better approach – Sliding window (w) looking for significant change in average value (Δ) of 2 halves of the window

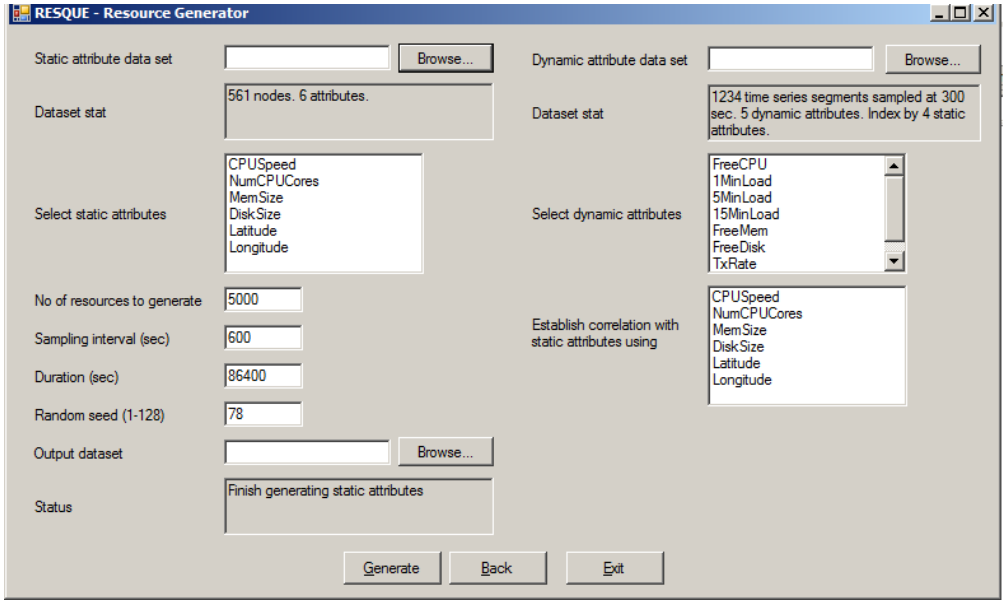
Dynamic Attributes – Contemporaneous Correlation



- Split other time series at same position & replay segments together
- Concatenate segments to form longer sequences
- Segments are index by static attributes

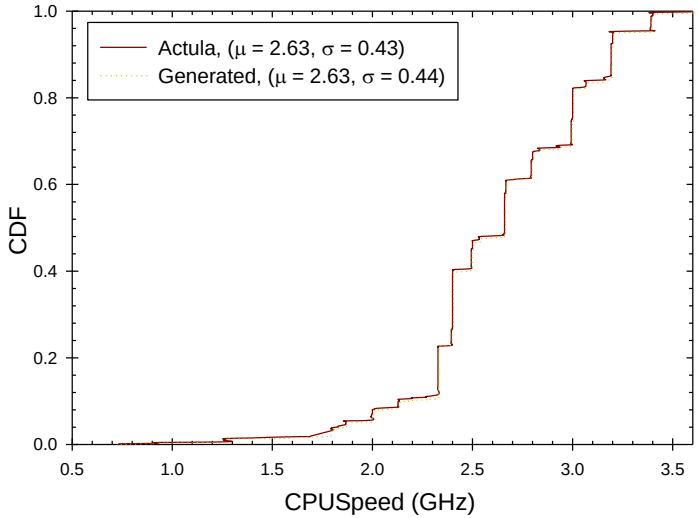
RESQUE – RESource & QUEry Generator

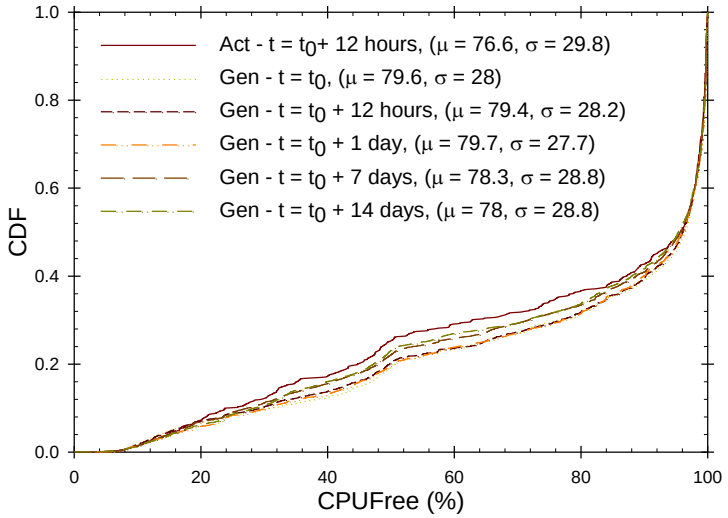




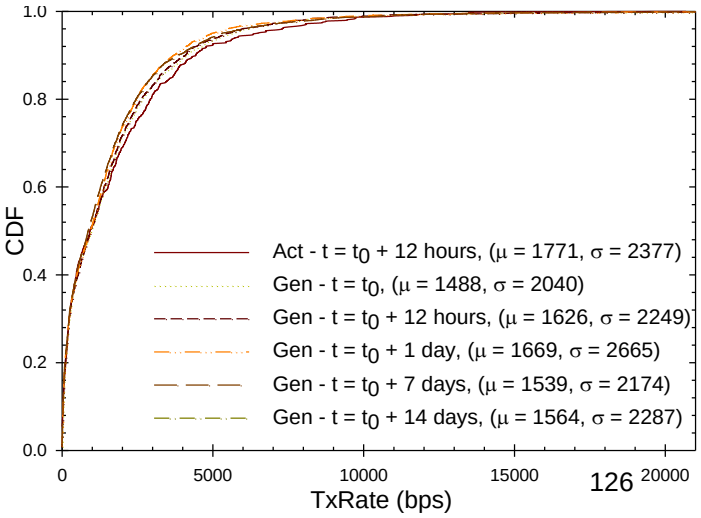
- NumCores establish correlation between static & dynamic
- Generate synthetic traces with n nodes, a_s static & a_d dynamic attributes over a given time t
- Also generate multi-attribute queries
- Beta version available – www.cnrl.colostate.edu/Projects/CP2P/

Resource Generation – Validation

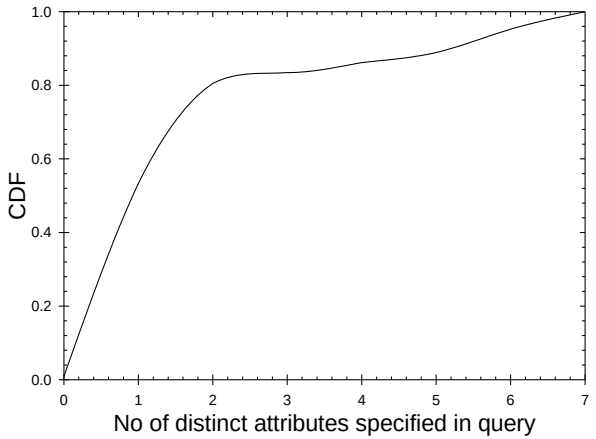


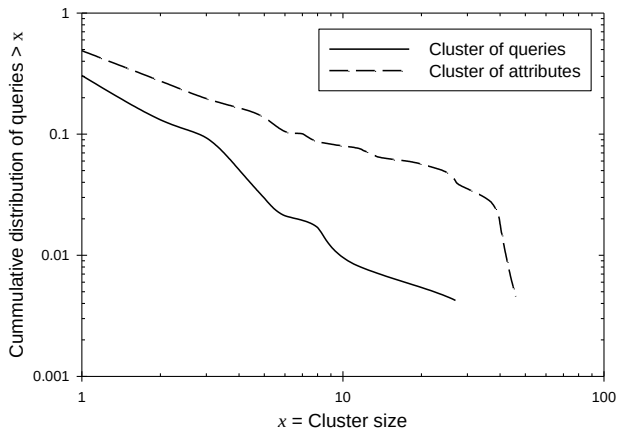


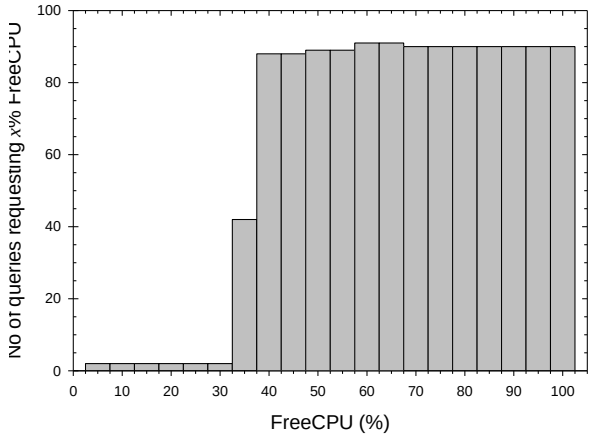
- Using 300 nodes over a week → generated 5,000 nodes over 2 weeks
- Satisfy Kolmogorov-Smirnov (KS) test with a significance level of 0.05
- Statistically accurate data



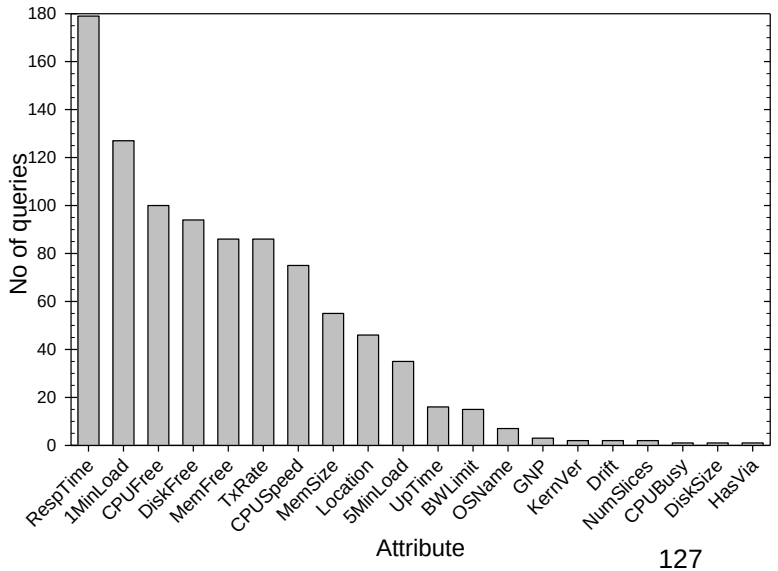
Query Characteristics







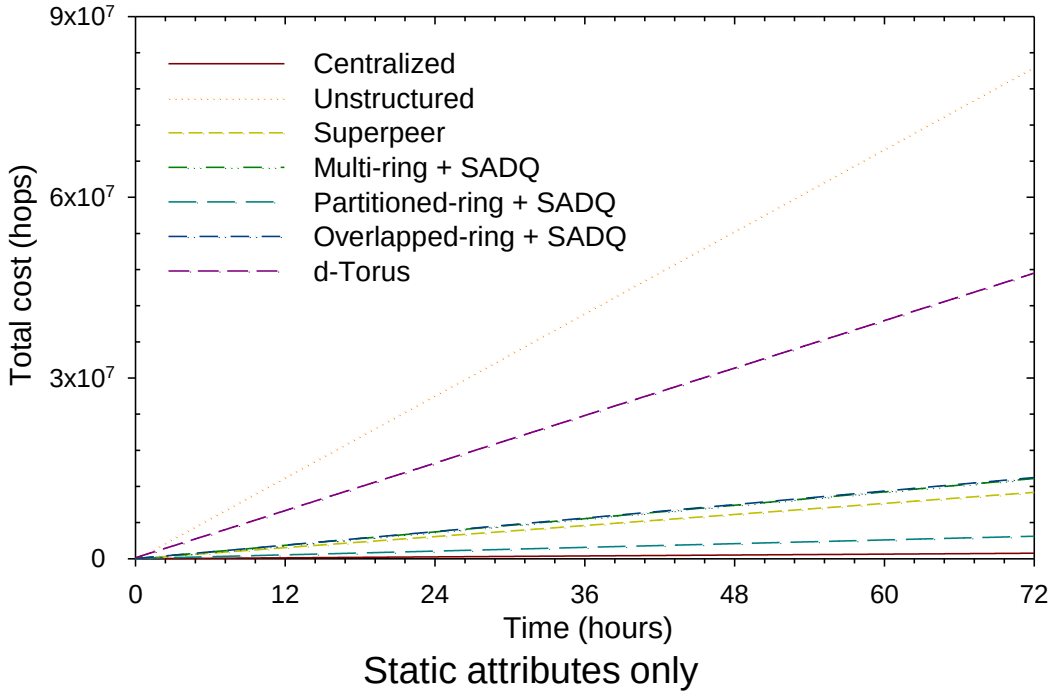
- 441 queries, 9 moths
- Few attributes in a query
  - 80% queries specify 1 or 2 attributes
- Skewed but not Zipf's
- Less specific attribute ranges
  - 89% queries request CPUFree of 40-100%
  - 70% queries request DiskFree of 5-1000GB
- Dynamic attributes are popular
- Large number of resources
  - 73 resources per query



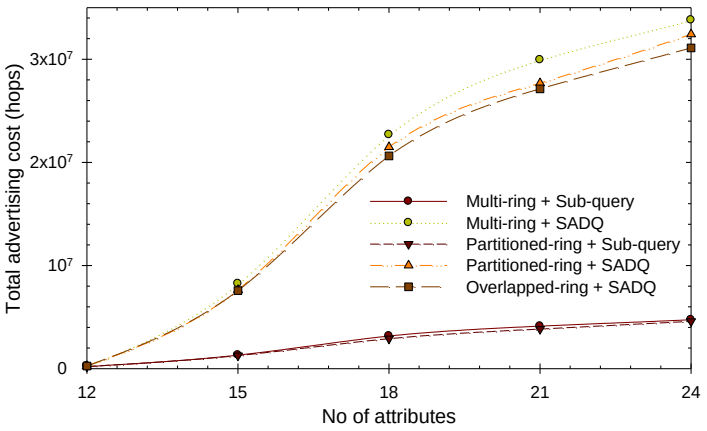
Comparison of Existing Solutions

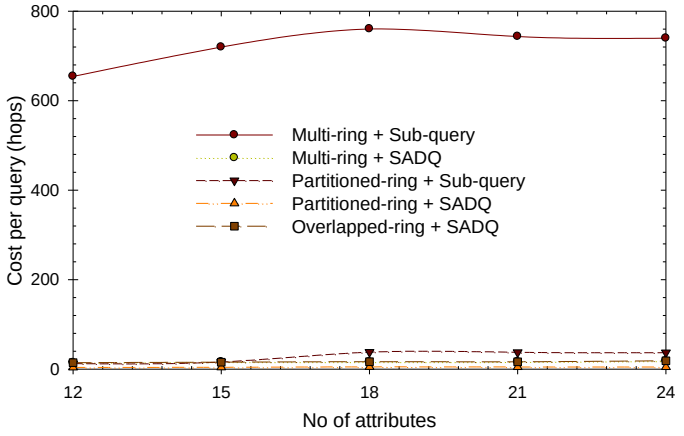
- Most prior assumptions are not valid
  - Resources
    - Many attributes, mixture of distributions, skewed, correlated, change rapidly
  - Queries
    - Few attributes, request many resources, large range of attribute values, skewed
- Need to validate existing designs under real-workloads
  - Mostly extensions of single-attribute solutions
- Contributions
  - Simple cost model for advertising & querying
  - Simulated 7 designs using PlanetLab resources & query traces
  - Unable to deliver desired performance, load balancing, etc.

Simulation of Different Solutions



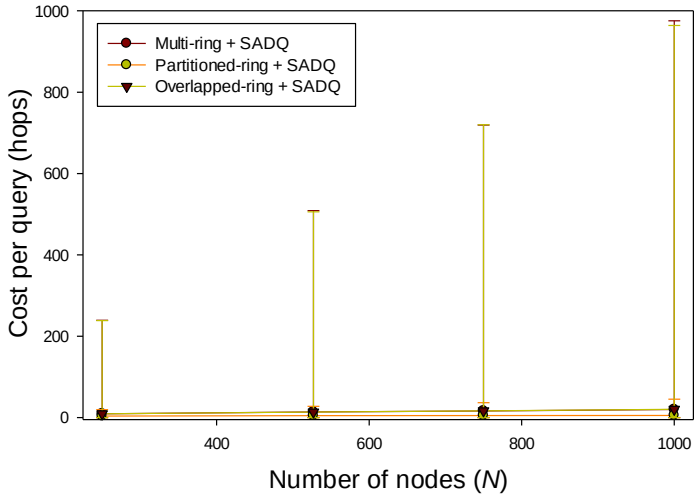
Static attributes only



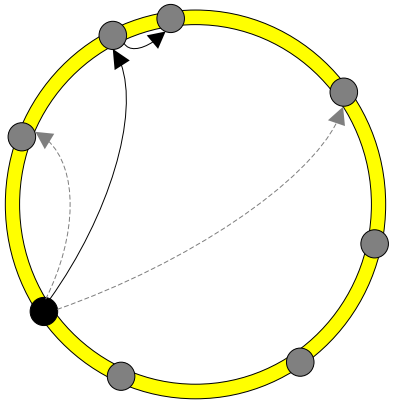


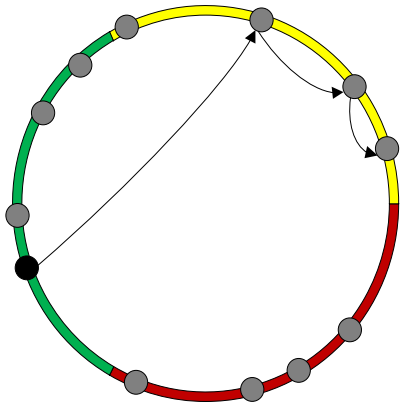
- Rings – higher advertising cost
- Partitioned ring – Disproportionate load distribution
- Superpeers – Balance cost & load but can't locate all resources

Simulation of Different Solutions (cont.)



N	Multi-ring + SADQ			Partitioned-ring + SADQ			Overlapped-ring + SADQ		
	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max
250	0	9.2	239.1	0	3.7	19.4	0	9.1	238.4
527	0	13.7	509.0	0	4.6	27.6	0	13.5	506.0
750	0	16.2	719.1	0	4.9	36.6	0	16.5	719.9
1000	0	19.8	975.5	0	5.3	45.3	0	20.4	963.8



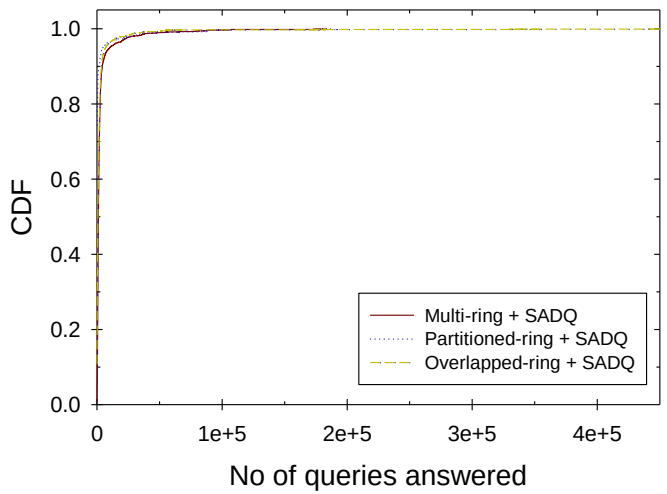


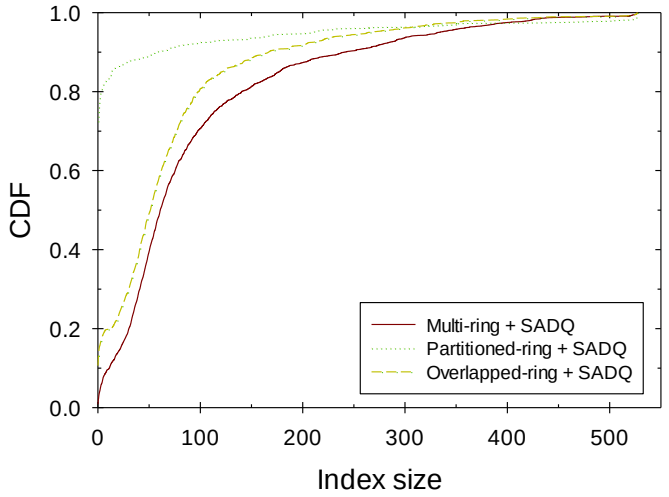
- Large range of attribute values → cost of ring-based designs $O(N)$

Load Distribution

Architecture	Total Cost per Query		Query Load				Index Size	
			Min		Max		Min	Max
	SWORD	Uniform	SWORD	Uniform	SWORD	Uniform		
Centralized	2.03	2.03	950,000	950,000	950,000	950,000	527	527
Unstructured	69.5	94.8	4,859	1,272	268,497	37,824	1	1
Superpeer	6.5	9.5	81,021	22,390	289,626	87,209	17	36
Multi-ring + SADQ	48.3	69.0	0	0	178,492	22,943	0	527
Multi-ring + Sub-queries	398.8	120.8	0	0	624,837	57,518	0	230
Partitioned-ring + SADQ	36.6	37.0	0	0	185,972	15,840	0	527
Partitioned-ring + Sub-queries	40.7	16.4	0	0	432,859	46,946	0	527
Overlapped-ring + SADQ	46.0	67.2	0	0	391,738	57,524	0	527

N = 527, *Attributes* = 24





- Unbalanced index size & query load

Challenges & Opportunities

Challenges	Opportunities
Diversity in • Resources • Application requirements • Complex inter-resource relationships	New solutions • Support large number of resources & attributes • Consider real-world resource & query characteristics • How to specify application requirements & constraints • Efficiently track & match inter-resource relationships

No solution satisfy select, match, & bind	Supporting select, match, & bind within a single solution
	• Track inter-node
	  • Bandwidth, latency, jitter, packet loss, etc.
	  • Social relationships
	• Distributed resource binding

Challenges & Opportunities (cont.)

Challenges	Opportunities
High cost • Query cost • Advertising dynamic attributes	Enhance performance • Better support for dynamic attributes • Reduce query cost • New DHT mechanisms • Efficient updates – static/dynamic thresholds to reduce number of updates
Load balancing	Dynamic/adaptive solutions • Based on queries & updates • Based on resources being indexed • Supporting many attributes & values • Some attributes have few values
Overcoming resource failures & unavailability	Resource compensation • Substituting one resource with another • CASA – can process faster to accommodate high transmission delay due to lack of bandwidth

Challenges & Opportunities (cont.)

Challenges	Opportunities
Increasing user participation	Incentives, security, & trust • Essential in collaborative P2P • Virtual currency schemes to support community clouds
Capturing large & high resolution datasets	Tools to • Capture datasets • Generate statically accurate synthetic datasets

Comments & Questions

Anura P. Jayasumana

Electrical & Computer Engineering,

Colorado State University,

Fort Collins, CO 80525&

Anura.Jayasumana@Colostate.edu

www.engr.colostate.edu/~anura

Bibliography

1. E. Adar and B.A. Huberman, *Free Riding on Gnutella*, 2000.
2. J. Albrecht, D. Oppenheimer, D. Patterson, and A. Vahdat, *Design and implementation tradeoffs for wide-area resource discovery*, ACM Trans. Internet Technol, 8(4), Sep. 2008.
3. H. M. N. D. Bandara and A. P. Jayasumana, *Exploiting Communities for Enhancing Lookup Performance in Structured P2P Systems*, IEEE Int. Conf. on Communications (ICC2011), June 2011.
4. H. M. N. Dilum Bandara and Anura P. Jayasumana, *On Characteristics and Modeling of P2P Resources with Correlated Static and Dynamic Attributes*, IEEE GLOBECOM '11, Dec. 2011.
5. H. M. N. D. Bandara and A. P. Jayasumana, *Characteristics of Multi-Attribute Resources/Queries and Implications on P2P Resource Discovery*, 9th ACS/IEEE Int. Conf. On Computer Systems And Applications (AICCSA 2011), Dec. 2011.
6. H. M. N. D. Bandara and A. P. Jayasumana, *Community-Based Caching for Enhanced Lookup Performance in P2P Systems, 2011, under review.*
7. H. M. N. D. Bandara and A. P. Jayasumana, *Evaluation of P2P Resource Discovery Architectures Using Real-Life Multi-Attribute Resource and Query Characteristics*, IEEE Consumer Communications and Networking Conf. (CCNC '12), Jan. 2012.
8. T. Banka, P. Lee, A. P. Jayasumana and J. F. Kurose, *An Architecture and a Programming Interface for Application-Aware Data Dissemination Using Overlay Networks*, COMSWARE 2007, Jan. 2007.
9. A. R. Bharambe, M. Agrawal, and S. Seshan, *Mercury: Supporting scalable multi-attribute range queries*, ACM SIGCOMM '04, Aug.-Sep. 2004.
10. R. Bland, D. Caulfield, E. Clarke, A. Hanley, and E. Kelleher, "P2P routing," Available: http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/p9.html

Bibliography

11. D. Brookshier, *Overview of JXTA*, Aug. 2002, Available: www.developer.com/java/other/article.php/10936_1450221_1
12. M. Cai, M. Frank, J. Chen, and P. Szekely, *MAAN: A multi-attribute addressable network for grid information services*, Journal of Grid Computing, Jan 2004.
13. Z. Chen, K. Xue, and P. Hong, *A study on reducing chunk scheduling delay for mesh-based P2P live streaming*, 7th Int. Conf. on Grid and Cooperative Computing, 2008, pp. 356-361.
14. Cisco Systems, Inc., *Cisco Visual Networking Index: Forecast and Methodology, 2008–2013*, June 2009.
15. Cisco Systems, Inc., *Approaching the Zettabyte Era*, June 2008.
16. E. Cohen and S. Shenker, *Replication strategies in unstructured peer-to-peer networks*, ACM SIGCOMM ‘02, Aug. 2002.
17. P. Costa, J. Napper, G. Pierre, and M. Steen, *Autonomous resource selection for decentralized utility computing*, 29th Int’l. Conf. Distributed Computing Systems, June 2009.
18. G. DeCandia et al., *Dynamo: Amazon's highly available key-value store*, ACM SIGOPS Operating Systems (SOSP '07), vol. 41, no 6, Oct. 2007, pp. 205-220.
19. P. Ganesan, B. Yang, and H. Garcia-Molina, *One torus to rule them all: Multi-dimensional queries in P2P systems*. 7th Int’l Workshop on the Web and Databases (WebDB ‘04), June 2004.
20. P. Ganesan, K. Gummadi, and H. Garcia-Molina, *Canon in G major: designing DHTs with hierarchical structure*, 24th Int. Conf. on *Distributed Computing Systems, 2004, pp.* 263-272.
21. P. B. Godfrey and I. Stoica, *Heterogeneity and load balance in distributed hash tables*, IEEE INFOCOM, Mar. 2005.

Bibliography

22. S. Guha, N. Daswani, and R. Jain, *An Experimental Study of the Skype Peer-to-Peer VoIP System*, 5th Int. Workshop on Peer-to-Peer Systems (IPTPS '06), Feb. 2006.
23. Y. Guo, C. Liang, and Y. Liu, *Adaptive queue-based chunk scheduling for P2P live streaming*, IFIP Networking, May 2008.
24. X. Hei, Y. Liu, and K. W. Ross, *IPTV over P2P streaming networks: the mesh-pull approach*, IEEE Communications Magazine, vol. 46, no. 2, Feb. 2008, pp. 86-92.
25. T. Koponen et al., *A data-oriented (and beyond) network architecture*, SIGCOMM '07, 2007.
26. S. Kwan and J. K. Muppala, *Bag-of-tasks applications scheduling on volunteer desktop grids with adaptive information dissemination*, IEEE LCN '10, Oct. 2010, pp. 560-567.
27. C. Leng, W. W. Terpstra, B. Kemme, W. Stannat, and A. P. Buchmann, *Maintaining replicas in unstructured P2P systems*, ACM Int. Conf. on Emerging Networking Experiments and Technologies (CoNEXT), Dec. 2008.
28. B. Leong, B. Liskov, E. D. Demaine, *EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management*, 12th Int. Conf. on Networks, 2004.
29. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, *Search and replication in unstructured peer-to-peer networks*, 16th Int. Conf. on Supercomputing (ICS '02), June 2002, pp. 84-95.
30. Microsoft, *Introduction to Windows Peer-to-Peer Networking*, Sep. 2006, Available: http://technet.microsoft.com/en-us/library/bb457079(d=printer).aspx
31. Microsoft, *Peer Name Resolution Protocol*, Sep. 2006, Available: http://technet.microsoft.com/en-us/library/bb726971(printer).aspx
32. R Morselli, B. Bhattacharjee, A. Srinivasan, and M. A. Marsh, *Efficient lookup on unstructured topologies*, 24th ACM Symposium on Principles of Distributed Computing (PODC '05), July 2005.

Bibliography

33. P. Lee, T. Banka, A. P. Jayasumana, and V. Chandrasekar, *Content Based Packet Marking for Application-Aware Processing in Overlay Networks*, IEEE LCN '06, Nov. 2006.
34. P. Lee, A. P. Jayasumana, S. Lim, and V. Chandrasekar, *A Peer-to-Peer Collaboration Framework for Multisensor Data Fusion*, Int. Joint Conf. on Computer, Information, and Systems Sciences, and Engineering (CISSE '07), Dec. 2007.
35. A. Legout, N. Liogkas, E. Kohler, and L. Zhang, *Clustering and Sharing Incentives in BitTorrent Systems*, SIGMETRICS '07, June 2007.
36. J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," In Proc. of IEEE, vol. 96, no. 1, Jan. 2008, pp. 11-24.
37. P. Maymounkov and D. Mazières, *Kademlia: A peer-to-peer information system based on the XOR metric*, 1st Int. Workshop on Peer-to-peer Systems (IPTPS '02), Feb. 2002, pp. 53-65.
38. V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy and A. E. Mohr, *Chainsaw: eliminating trees from overlay multicast*, 4th Int. Workshop on Peer-to-Peer Systems (IPTPS), Feb. 2005, pp. 127-140.
39. V. Ramasubramanian and E. G. Sirer, *Beehive: O(1) lookup performance for power-law query distributions in peer-to-peer overlays*, 1st Symposium on Networked Systems Design and Implementation (NSDI), 2004, pp. 99-112.
40. R. Ranjan, A. Harwood, and R. Buyya, *Peer-to-Peer based resource discovery in global grids: a tutorial*, IEEE Commun. Surveys, vol. 10, no. 2, 2008.
41. A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, *Load balancing in structured P2P systems,* 2nd Int. Workshop on P2P Systems (IPTPS), Feb. 2003.
42. W. Rao, L. Chen, A. W. Fu, and Y. Bu, *Optimal proactive caching in peer-to-peer network: analysis and application*, 6th ACM Conf. on Information and Knowledge Management (CIKM '07), Nov. 2007, pp. 663-672.

Bibliography

43. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A scalable content-addressable network*, ACM Special Interest Group on Data Communication (SIGCOMM '01), Aug. 2001.
44. J. Seedorf, S. Kiesel, and M. Stiemerling, *Traffic Localization for P2P-Applications: The ALTO Approach*, IEEE P2P '09, 2009.
45. H. Shen, C. Xu, and G. Chen, *Cycloid: A constant-degree and lookup-efficient P2P overlay network*, Performance Evaluation, vol. 63, no. 3, Mar. 2006, pp. 195-216.
46. H. Shen, A. Apon, and C. Xu, *LORM: supporting low-overhead P2P-based range-query and multi-attribute resource management in grids*, 13th Int. Conf. on Parallel and Distributed Systems, (ICPADS '07), Dec. 2007.
47. M. Sozio, T. Neumann, and G. Weikum, *Near-optimal dynamic replication in unstructured peer-to-peer networks*, 27th ACM symposium on Principles of Database Systems (PODS '08), June 2008, pp. 281-290.
48. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, *Chord: a scalable peer-to-peer lookup service for internet applications*, ACM SIGCOMM '01, 2001, pp. 149-160.
49. I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, *Internet Indirection Infrastructure*, ACM SIGCOMM, Aug. 2002.
50. D. Stutzbach, R. Rejaie, and S. Sen, *Characterizing unstructured overlay topologies in modern P2P file-sharing systems*, IEEE/ACM Transactions on Networking, vol. 16, no. 2, April 2008.
51. Y. Tang, L. Sun, K. Zhang, S. Yang, and Y. Zhong, *Longer, better: On extending user online duration to improve quality of streaming service in P2P networks*, IEEE Int. Conf. on Multimedia and Expo, July 2007.
52. H. Xie, A. Krishnamurthy, A. Silberschatz, and Y. Richard Yang, *P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers*, P4P WG Whitepaper, May 2007.

Bibliography

53. J. Yao, J. Zhou, and L. Bhuyan, *Computing real time jobs in P2P networks*, IEEE LCN, Nov. 2006, pp. 107-114.
54. T. Zahn and J. Schiller, *MADPastry: a DHT substrate for practicably sized MANETs*, 5th Workshop Applications and Services in Wireless Networks, June/July 2005.
55. B. Zhang, A. Iosup, J. Pouwelse, D. Epema, and H. Sips, *On Assessing Measurement Accuracy in BitTorrent Peer-to-Peer File-Sharing Networks*, Delft University of Technology, 2009.
56. M. Zhang, Y. Xiong, Q. Zhang, and S. Yang, *On the optimal scheduling for media streaming in data-driven overlay networks*, GLOBECOM '06, Nov.-Dec. 2006.
57. X. Zhang, J. Liu, B. Li, and T. P. Yum, *CoolStreaming/DONet: a data-driven overlay network for efficient live media streaming*, INFOCOM 2005, Mar. 2005.
58. M. Zhong, K. Shen, and J. Seiferas, *Replication degree customization for high availability*, European Conf. on Computer Systems (EuroSys '08), Apr. 2008, pp. 55-68.

59. J.F. Buford, H. Yu and E.K. Lua, P2P Networking and Applications, Morgan Kaufmann, 2009.