HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Overlay and P2P Networks

Applications

Prof. Sasu Tarkoma

10.2.2014

Contents

Summary

- Overlay networks have been proposed
  - Searching, storing, routing, notification,..
  - Lookup (Chord, Tapestry, Pastry), coordination primitives (i3), middlebox support (DOA)
  - Logarithmic scalability, decentralised,…
- Many applications for overlays
  - Lookup, rendezvous, data distribution and dissemination, coordination, service composition, general indirection support
- Deployment open. PlanetLab.

	CAN	Chord	Kademlia	Koorde	Pastry	Tapestry	Viceroy
Foundation	Multi-dimensional space (d-dimensional torus)	Circular space	XOR metric	de Bruijn graph	Plaxton-style mesh	Plaxton-style mesh	Butterfly network
Routing function	Maps (key,value) pairs to coordinate space	Matching key and nodeID	Matching key and nodeID	Matching key and nodeID	Matching key and prefix in nodeID	Suffix matching	Routing using levels of tree, vicinity search
System parameters	Number of peers N, number of dimensions d	Number of peers N	Number of peers N, base of peer identifier B	Number of peers N	Number of peers N, base of peer identifier B	Number of peers N, base of peer identifier B	Number of peers N
Routing performance	*$O(dN^{1/d})$*	*$O(\log N)$*	*$O(\log_B N)$ + small constant*	*Between $O(\log \log N)$ and $O(\log N)$, depending on state*	*$O(\log_B N)$*	*$O(\log_B N)$*	*$O(\log N)$*
Routing state	*2d*	*log N*	*$B\log_B N + B$*	*From constant to log N*	*$2B\log_B N$*	*$\log_B N$*	*Constant*
Joins/leaves	*2d*	*$(\log N)^2$*	*$\log_B N$ + small constant*	*log N*	*$\log_B N$*	*$\log_B N$*	*log N*

DHT: A General Approach

What is an address?

Base b with n digits

How to route efficiently?

Fix at least one digit per hop or take to the numerically closest destination based on routing table

How efficient is this?

$\log_b N$ steps gives O(log N) state and O(log N) hops!

DHT: A General Approach

How to populate routing table?

Iterative nearest neighbour search to fill the routing table.

Get enough information to be able to populate the routing table.

Comparing geometries

Gummadi et al. compared the different geometries, including the tree, hypercube, butterfly, ring, and XOR geometries.

Loguinov et al. complemented this list with de Bruijn graphs.

The conclusions of these comparisons include that the ring, XOR, and de Bruijn geometries are more flexible than the others and permit the choice of neighbours and alternative routes

The ring and XOR geometries were also found to be the most flexible in terms of choosing neighbours and routes

Only de Bruijn graphs allow alternate paths that are independent of each other

Comparison

Can you choose neighbours?

Can you choose routes?

Are there alternative routes?

Are there alternative routes without overlap?

Comparison

	Tree	Hypercube	Ring	Butterfly	XOR	De Bruijn
Neighbour selection	Yes	1	Yes	1	Yes	No
Route selection	1	Yes	Yes	1	Some	Yes
Sequential neighbours	No	No	Yes	No	No	Yes
Independent paths	No	No	No	No	No	Yes

Discussion

Based on previous table the ring looks pretty good

But this is partly due to the sequential neighbours property (predecessor and successor on the ring)

If sequential neighbours is added to other geometries, XOR and de Bruijn are also good

Comparison: Geometries

We observe that the foundations differ across the algorithms, but result in similar scalability properties

The conclusions of several comparisons of the geometries are that the ring, XOR, and de Bruijn geometries are more flexible than the others and permit the choice of neighbours and alternative routes

Note: it is possible to combine these

Example: Pastry that combines the tree and ring geometries

Comparison: Routing

The routing tables of DHTs can vary from size *O*(1) to *O*(*n*). The algorithms need to balance between maintenance cost and lookup cost

From the view point of routing state Chord, Pastry, and Tapestry offer logarithmic routing table sizes, whereas Koorde and Viceroy and support constant or near-constant sizes

Churn and dynamic peers can also be supported with logarithmic cost in some of the systems, such as Koorde, Pastry, Tapestry, and Viceroy

Recent analysis indicates that **large routing tables** actually lead to both low traffic and low lookup hops. These good design points translate into **one-hop routing** for systems of medium size and **two-hop routing** for large systems

Comparison: Churn

Li et al. provide a comparison of different DHTs under churn

They examine the fundamental design choices of systems including Tapestry, Chord, and Kademlia. The insights based on this work include the following:

- **Larger routing tables** are more cost-effective than more frequent periodic stabilization

- **Knowledge** about new nodes during **lookups** may allow to eliminate the need for stabilization

- **Parallel lookups** result in reduced latency due to timeouts, which provide information about the network conditions

Comparison: Network Proximity

Support for network proximity is one key feature of overlay algorithms. The three basic models for proximity awareness in DHTs are:

- **Geographic Layout.** Node identifiers are created in such a way that nodes that are close in the network topology are close in the nodeId space
- **Proximity Routing.** The routing tables do not take network proximity into account; however, the routing algorithm can choose a node from the routing table that is closest in terms of network proximity
- **Proximity Neighbour Selection.** In this model, the routing table construction takes network proximity into account. Routing table entries are chosen in such a way that at least some of them are close in the network topology to the current node

Asymptotic Tradeoffs

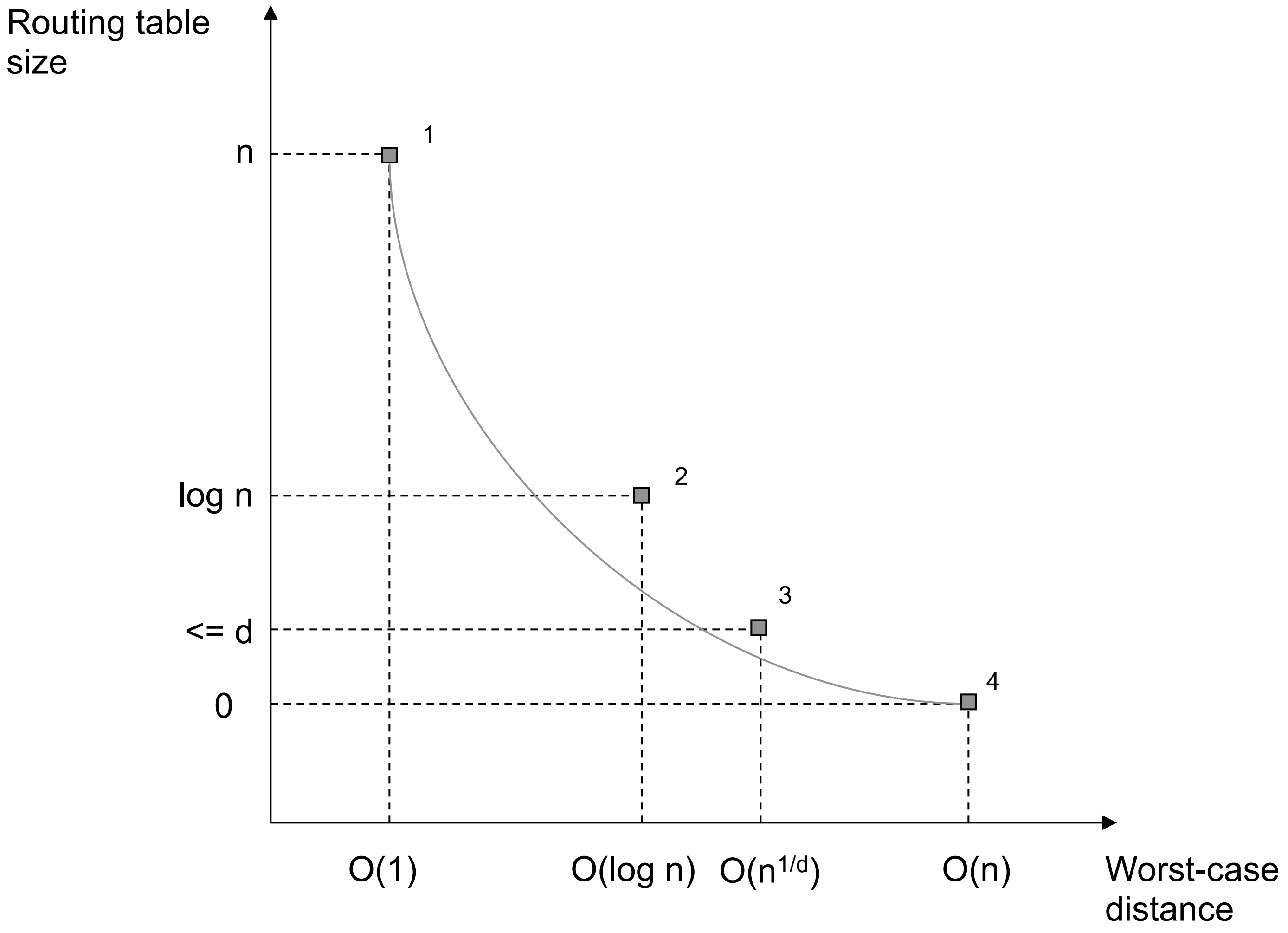
We analyze the asymptotic tradeoff curve between the routing table size and the network diameter

Analysis of the tradeoffs between the two metrics indicate that the routing table size of **Ω(log *n*)** is a **threshold point** that separates two distinct state-efficiency regions

One can observe that this point is in the middle of the symbolic asymptotic curve. If the routing table size is asymptotically smaller or equal, the requirement for congestion-free operation prevents it from achieving the smaller asymptotic diameter

When the routing table size is larger, the requirement for congestion-free operation does not limit the system anymore

Routing table size and network distance



Criticism

There have been two main criticisms of structured systems

The first pertains to peer transience, which is an important factor in maintaining robustness. Transient peers result in churn, which is a current concern with DHTs.

The second criticism of structured systems stems from their foundation in consistent hashing, which makes it more challenging to implement scalable query processing than for unstructured systems. Given that the popular file-sharing applications rely extensively on metadata based queries, simple exact-match key searches are not sufficient for them and additional solutions are needed on top of the basic DHT API

It is also possible to combine structured and unstructured algorithms in so called hybrid models

Applications

BitTorrent Mainline DHT

Decentralized tracker (trackerless torrent)

Based on Kademlia

Uses a custom RPC based on UDP

The **key** is the **info-hash**, the hash of the metadata. It uniquely identifies a torrent.

The **data** is a peer list of the peers in the swarm

Torrents have bootstrap nodes in the overlay

BitTorrent Mainline DHT

Each peer announces itself with the distributed tracker
- Looking up the 8 nodes closest to the info-hash of the torrent and sending an announce message to them

Those 8 nodes will then add the announcing peer to the peer list stored at that info-hash

A peer joins a torrent by looking up the peer list at a specific info-hash

Nodes return the peer list if they have it

Kademlia in Bittorrent Mainline DHT

The implementation extends the single bit model discussed before

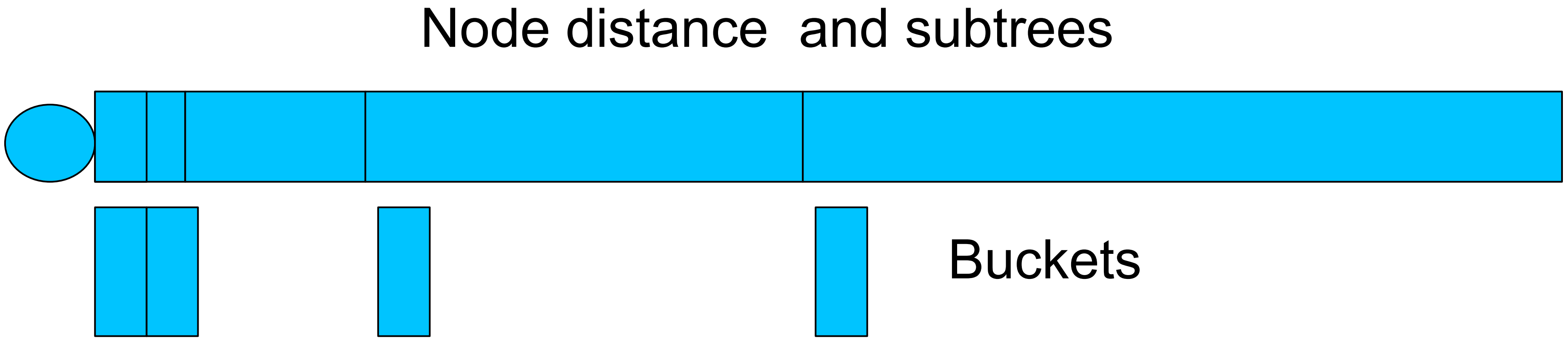
The single bit model can be seen to have a prefix first n-1 bits need to match for the nth list

The extension introduces prefix (group of bits)-based operation with width w for digits, giving $2^w - 1$ k-buckets with the missing one containing the node ID

An m-bit prefix reduces the maximum number of lookups from $\log_2 n$ to $\log_2^w n$

This results in a prefix-based routing table!

Kademlia Routing Table Revisited



Each node knows more about close nodes than distant nodes

Key space of each bucket grows with the power of 2 with the distance

Querying for an ID will on average halve the distance to the target in each step

Query Routing

Goal: Find k nodes closest to ID T

Initial Phase:

- Select α nodes closest to T from the routing table
- Send FIND_NODE(T) to each of the α nodes in parallel

Iteration:

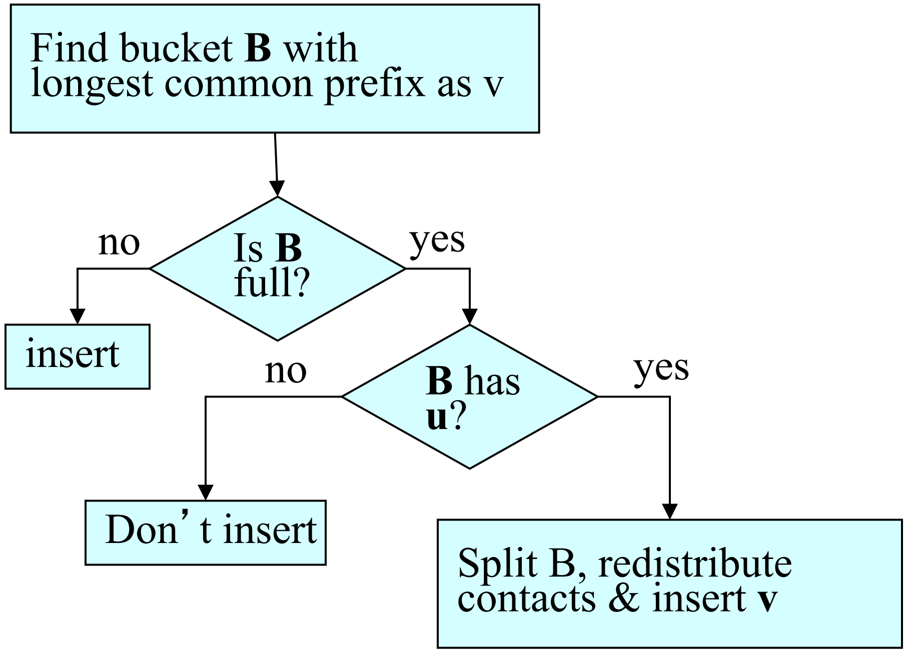
- Select α nodes closest to T from the results of previous RPC
- Send FIND_NODE(T) to each of the α nodes in parallel
- Terminate when a round of FIND_NODE(T) fails to return any closer nodes

Final Phase:

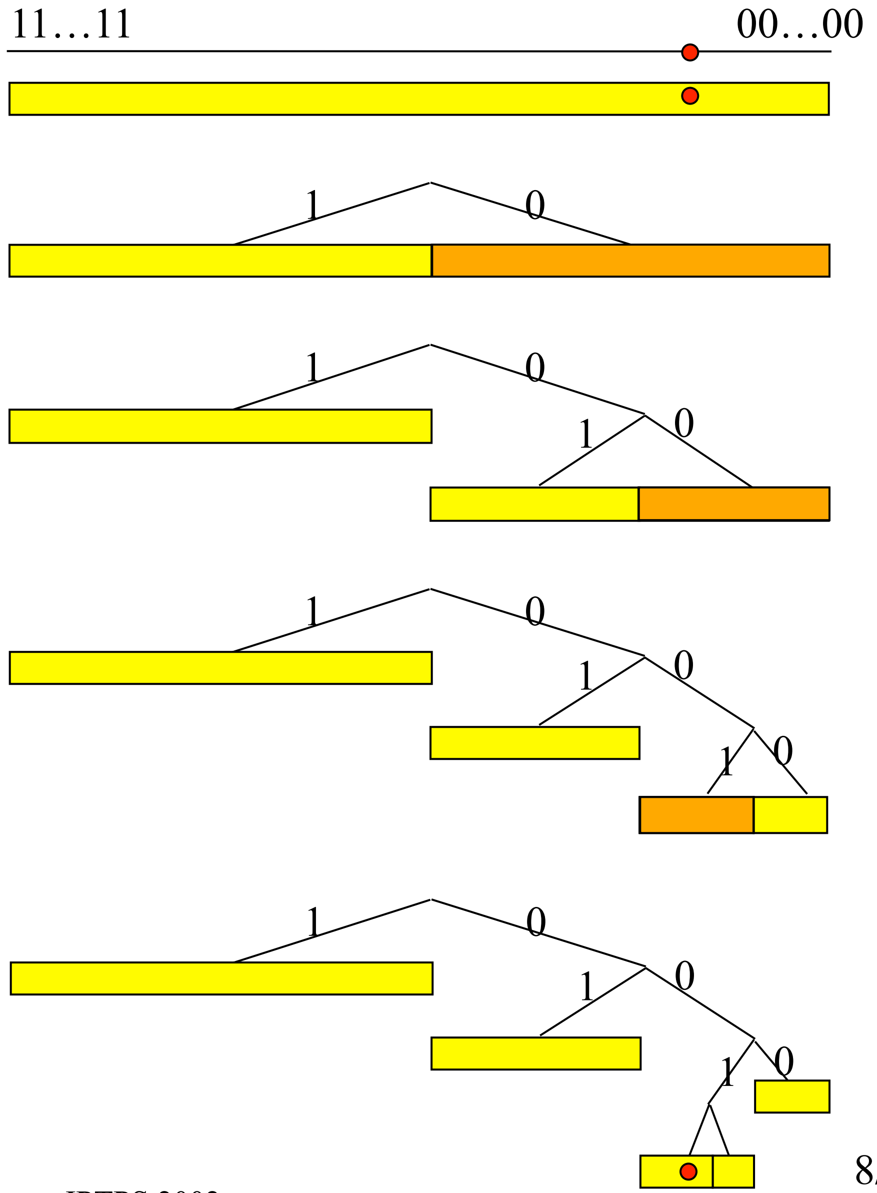
- Send FIND_NODE(T) to all of k closest nodes not already queried
- Return when have results from all the k-closest nodes.

- Joining Node (**u**):
  - ✓ Borrow an alive node's ID (**w**) off-line
  - ✓ Initial routing table has a single k-bucket containing u and w.
  - ✓ u performs FIND_NODE(**u**) to learn about other nodes

- Inserting new entry (**v**)



Find bucket **B** with longest common prefix as v
Is **B** full?
no
yes
insert
B has **u**?
no
yes
Don't insert
Split B, redistribute contacts & insert **v**



A Peer-to-peer Information System Based on the XOR Metric. Presentation at IPTPS 2002.

Comparisons

Kademlia and Chord

- Chord has only one direction on the ring
- Incoming traffic cannot be used to improve routing table
- But Chord has pred/succ (sequential neighbours)

Kademlia and Pastry

- Pastry has more complex table
- Pastry has sequential neighbours

What about Mainline DHT in practice?

Implementation Details

Mainline DHT implements Kademlia with a width of 2, and k = 8 nodes in each bucket

Keys are replicated on the three nodes with nodeID nearest the key with a 30-minute timeout

If a node fails, the keys will be lost

Nodes learn implicitly

- Iterative queries, incoming messages
- Lazy removal
- Ping LRU node when bucket full

Reported Problems with Mainline DHT

An Analysis of BitTorrent's Two Kademlia-Based DHTs

Scott A. Crosby and Dan S. Wallach, 2007

Do the DHTs work correctly? No. Mainline BitTorrent dead-ends its lookups 20% of the time and Azureus nodes reject half of the key store attempts.

What is the DHT lookup performance? Both implementations are extremely slow, with median lookup times around a minute.

Why do lookups take over a minute? Lookups are slow because the client must wait for RPCs to timeout while contacting dead nodes.

Dead nodes are commonly encountered in the area closest to the destination key.

Why are the routing tables full of dead nodes? Kademlia's use of iterative routing limits the ability for a node to opportunistically discover dead nodes in its routing table (refresh, explicit ping)

Design Problems

Iterative search can return dead nodes (no checking)

- Recursive routing would implicitly define liveness

Dead nodes are pruned only with refresh or explicit ping

XOR metric

- cannot enumerate nodes (as in Pastry or Chord)

Nodes can be ordered based on distance to given key

PAST

PAST: Cooperative, Archival File Storage and Distribution

Runs on top of Pastry, pastry routes to closest live nodeId

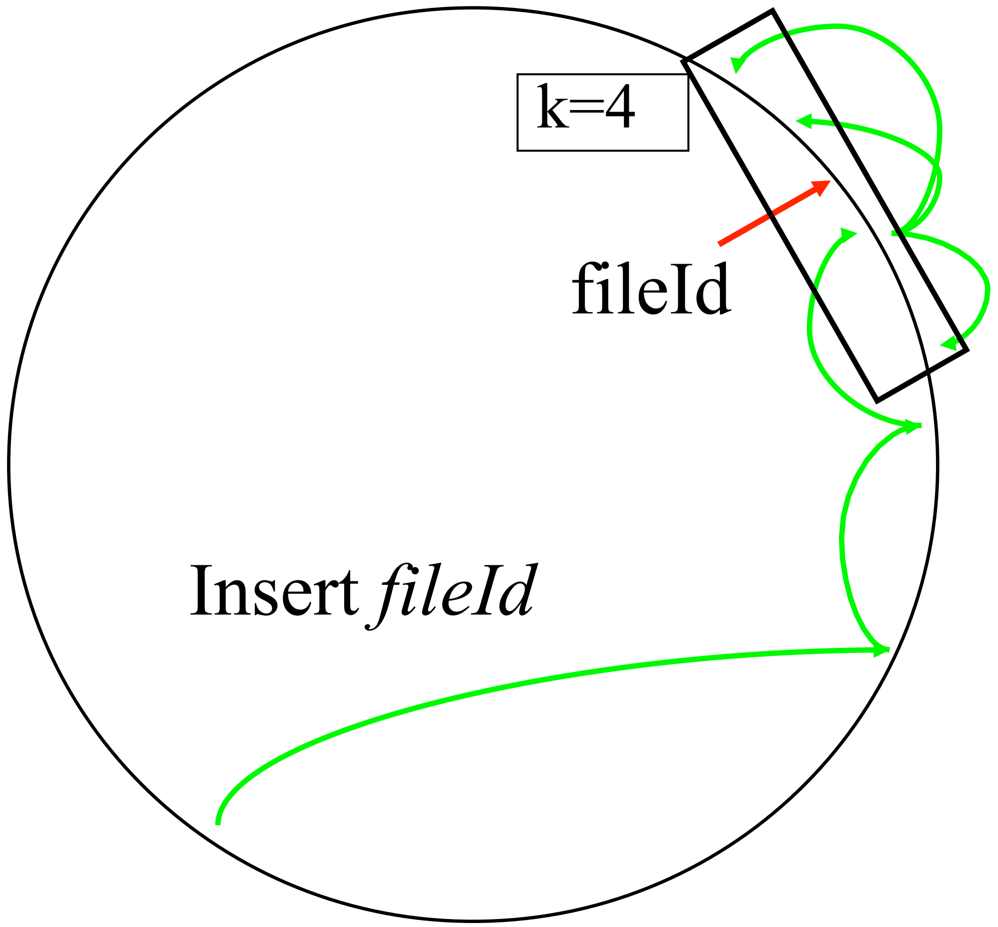
Strong persistence, high availability, scalability

API:

- Insert: store replica of a file at k diverse storage nodes
- Lookup: retrieve file from a nearby live storage node
- Reclaim: free storage associated with a file

Files are immutable!

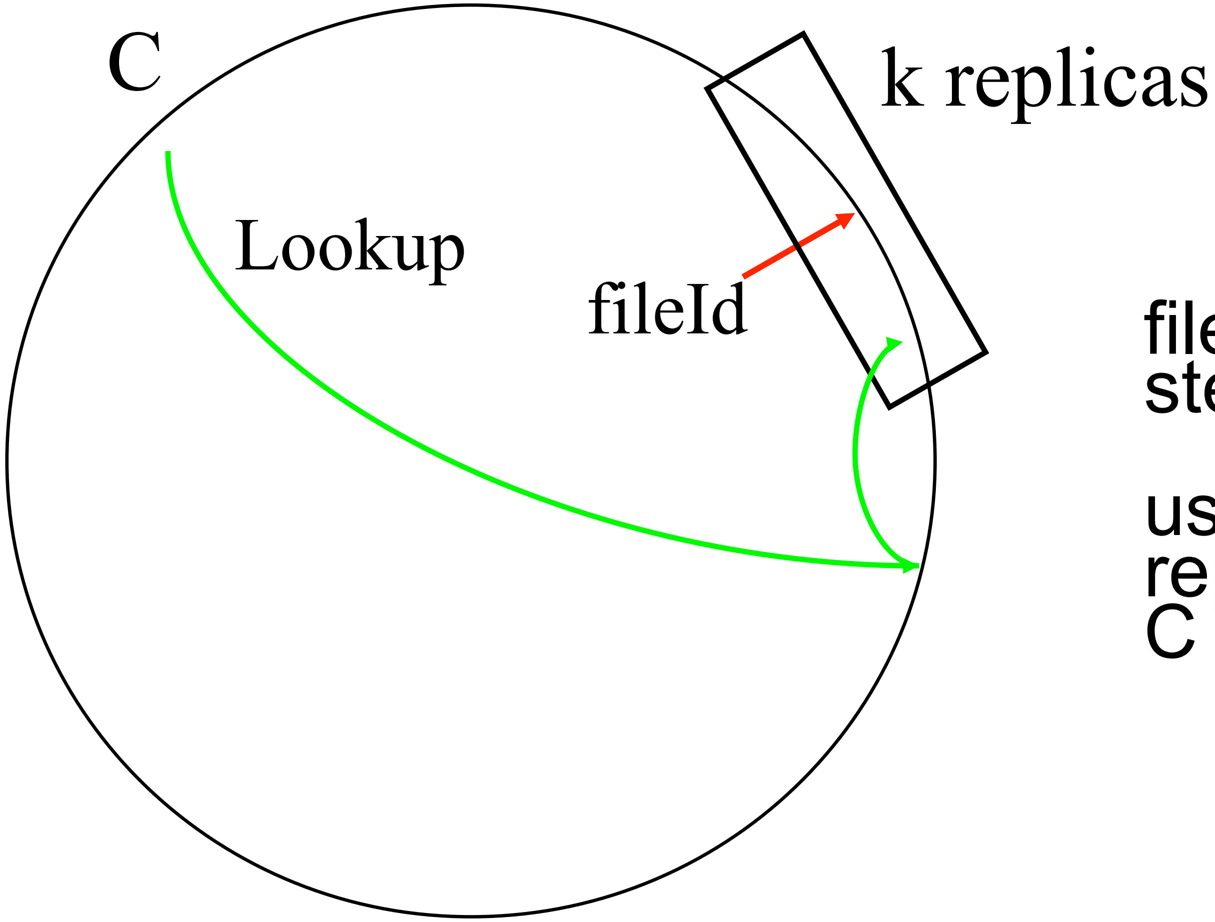
PAST File Storage



Storage Invariant: File "replicas" are stored on k nodes with nodeIds closest to fileId

(k is bounded by the leaf set size)

PAST File Retrieval



file located in $\log_{16} N$ steps (expected)

usually locates replica nearest client C

PAST Features

Caching

- On nodes along the route of lookup and insert messages (as in Freenet)
- Aim to balance load

Security

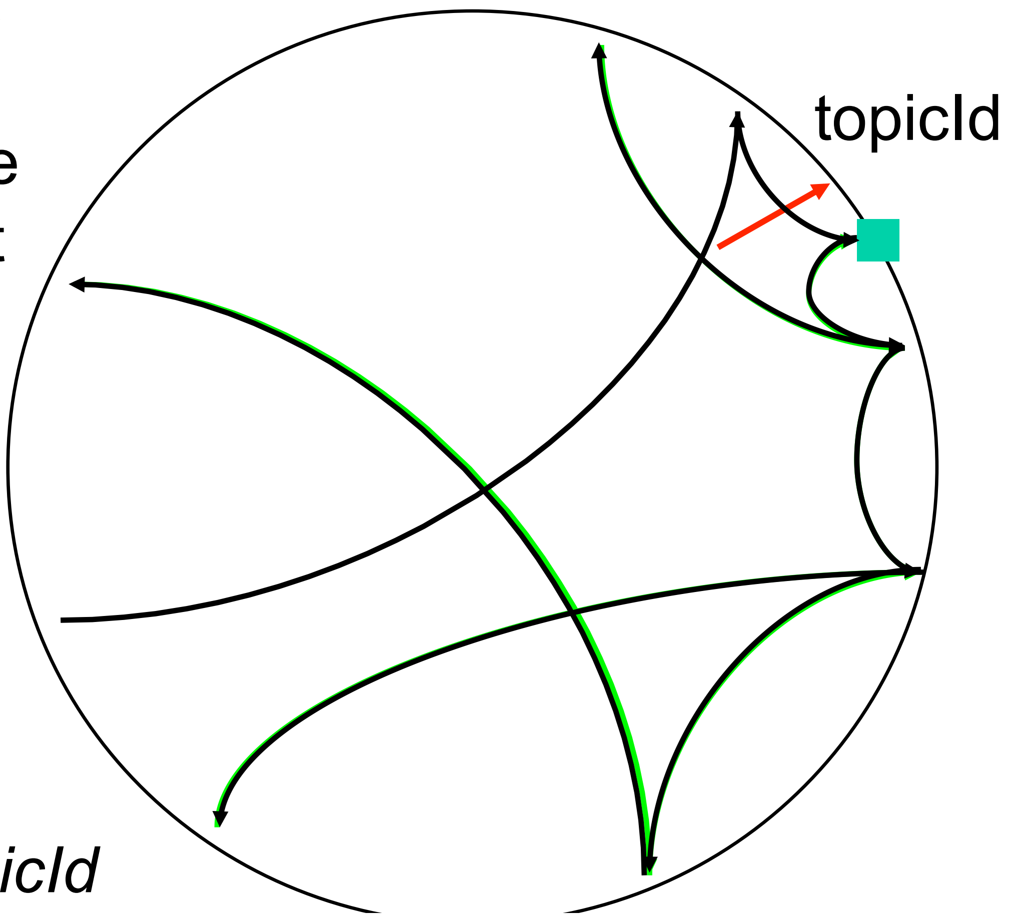
- No read access control, encryption can be used
- File authenticity with certificates
- System integrity: ids non-forgeable, sign sensitive messages
- Randomized routing

SCRIBE

SCRIBE: Large-scale, decentralized multicast

Intrastructure to support topic-based publish/ subscribe applications

Reasonable performance compared to IP multicast



Publish *topicId*

Subscribe *topicId*

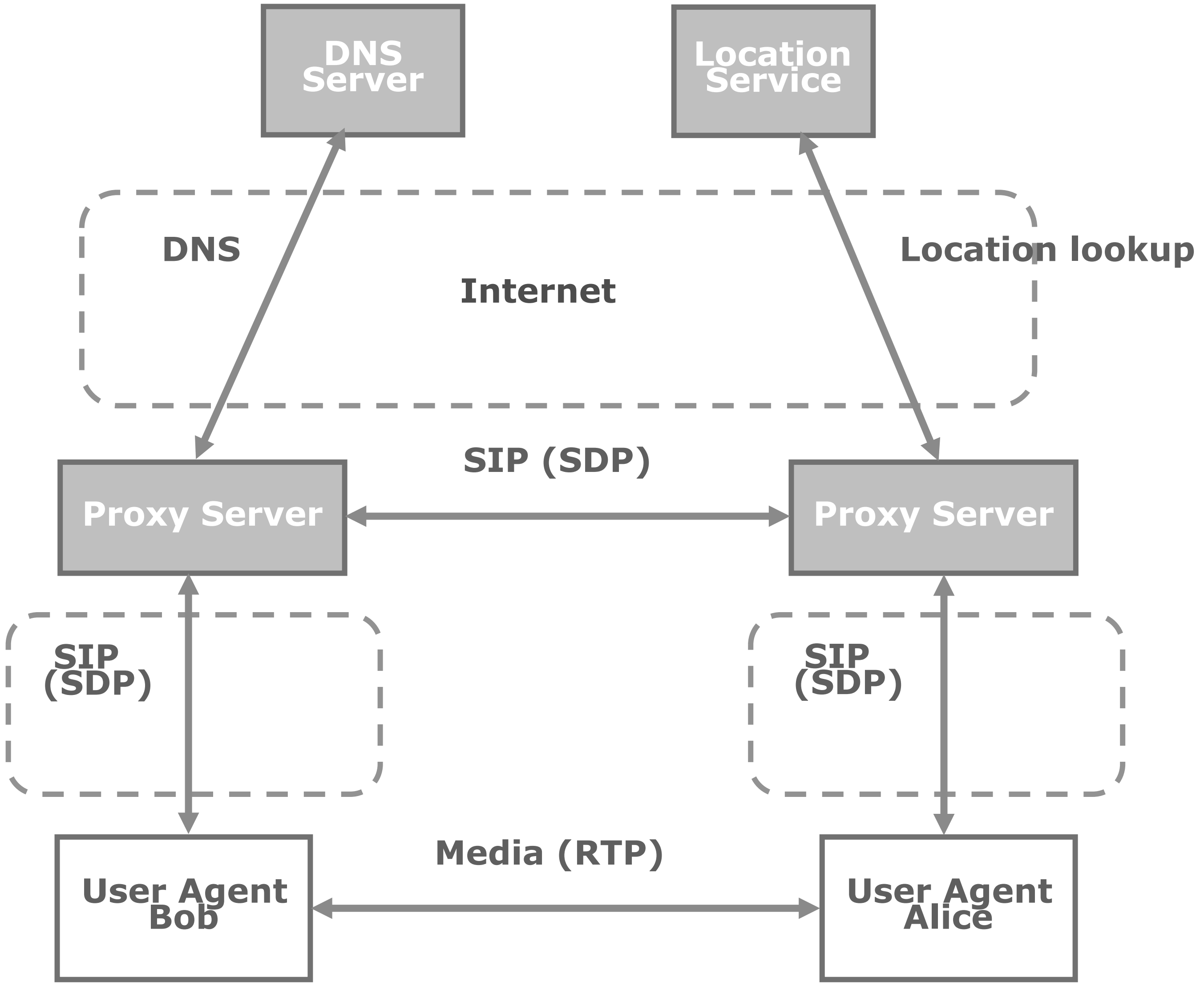
Session Initiation Protocol (SIP)

An Application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants

Sessions include Internet multimedia conferences, Internet telephone calls and multimedia distribution

Members in a session can communicate via multicast or via a mesh of unicast relations, or a combination of these

Text based, model similar to HTTP


DNS Server
Location Service
DNS
Internet
Location lookup
SIP (SDP)
Proxy Server
Proxy Server
SIP (SDP)
SIP (SDP)
Media (RTP)
User Agent Bob
User Agent Alice

P2P SIP

SIP is already ready for P2P Active standardization in IETF

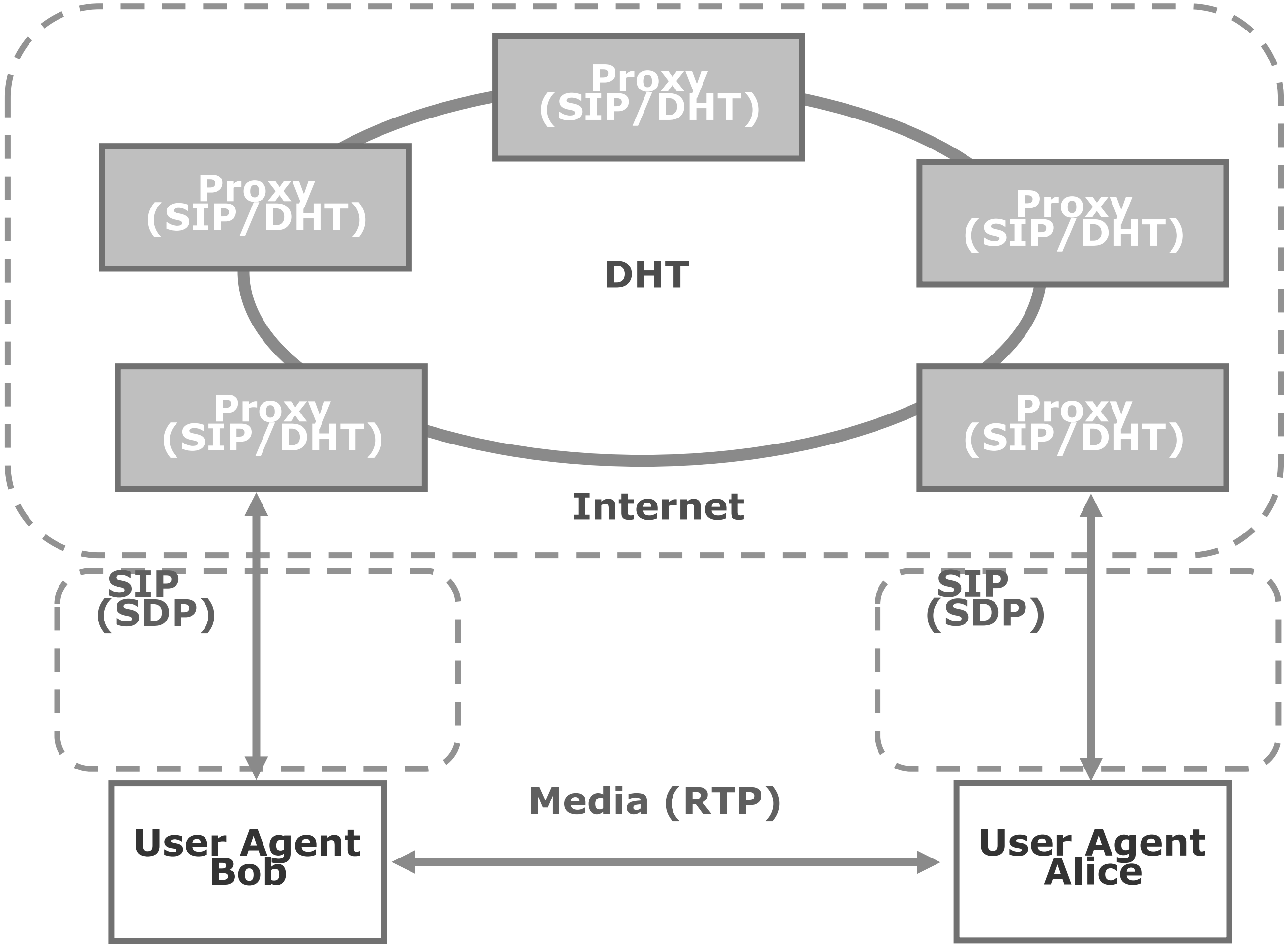
Uses symmetric, direct client-to-client communication

Intelligence resides mostly on the network border in the user agents

The proxies and the registrar only perform lookup and routing

The lookup/routing functions of the proxies/registrar can be replaced by a DHT overlay built in the user agents.

By adding join, leave and lookup capabilities, a SIP user agent can be transformed into a peer capable of operating in a P2P network


Proxy (SIP/DHT)
Proxy (SIP/DHT)
Proxy (SIP/DHT)
DHT
Proxy (SIP/DHT)
Proxy (SIP/DHT)
Internet
SIP (SDP)
SIP (SDP)
Media (RTP)
User Agent Bob
User Agent Alice