HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

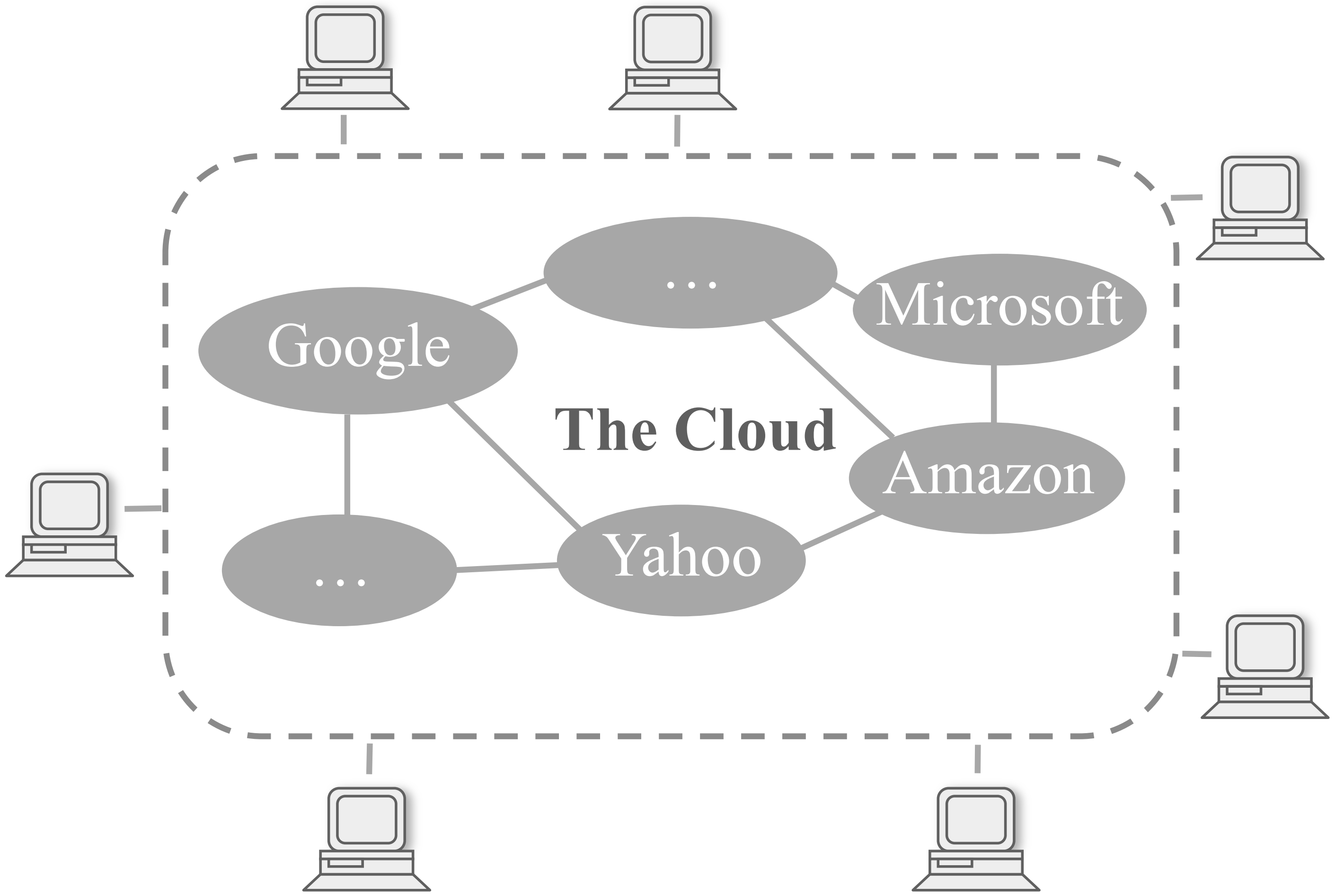
Overlay and P2P Networks

Applications

Prof. Sasu Tarkoma

27.9.2010

Contents


...
Microsoft
Google
The Cloud
Amazon
Yahoo
...

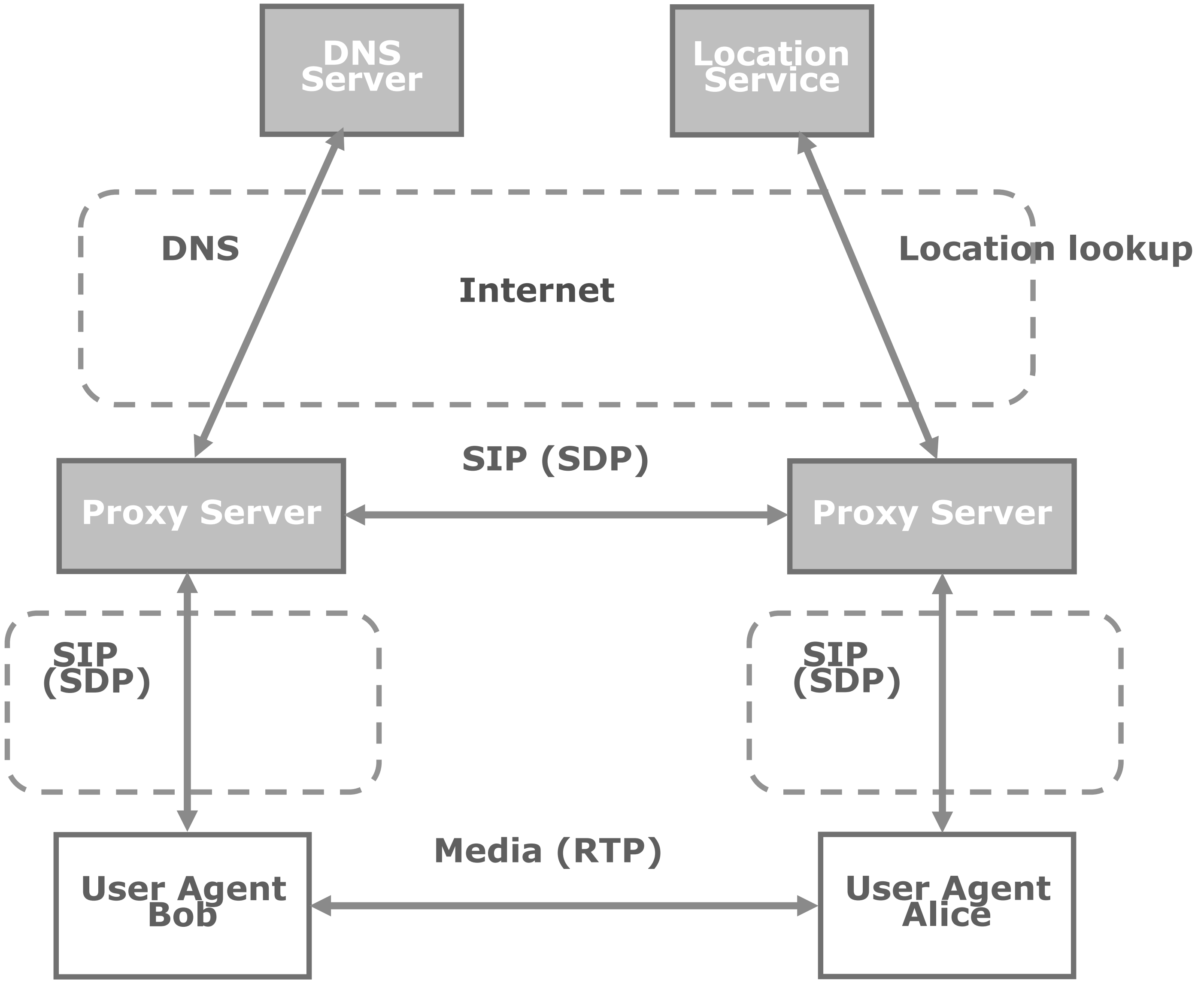
Session Initiation Protocol (SIP)

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

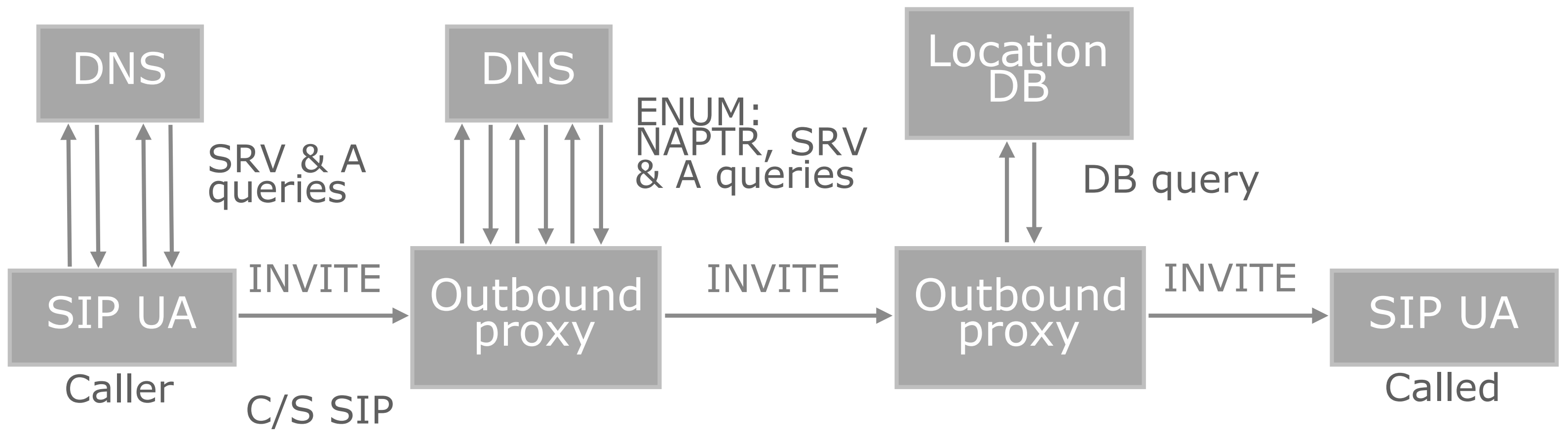
Sessions include Internet multimedia conferences, Internet telephone calls and multimedia distribution

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

SIP Signalling



P2P SIP

SIP is already ready for P2P
- Active standardization in IETF

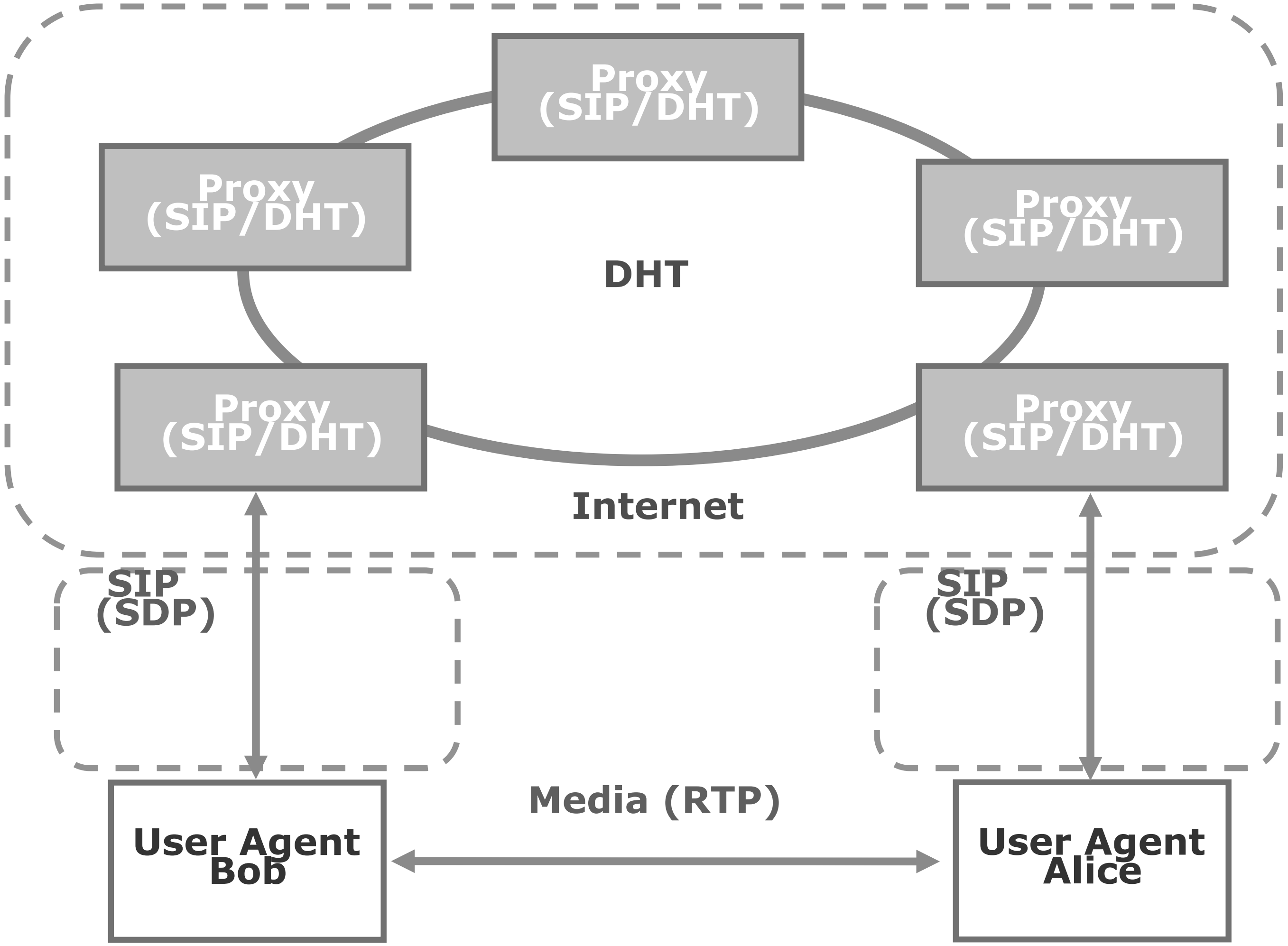
Uses symmetric, direct client-to-client communication

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

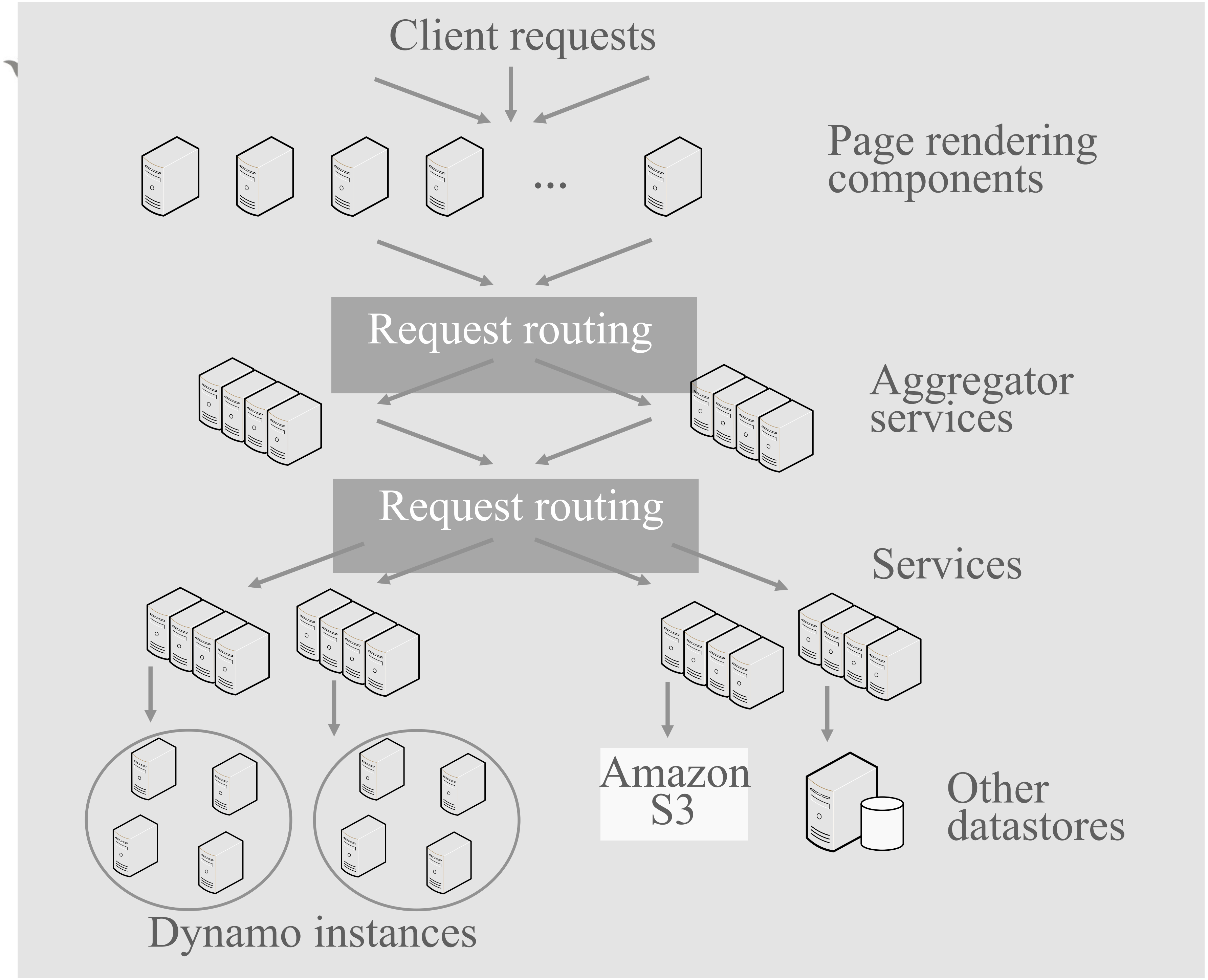
Amazon Dynamo Motivation

Aim is to store various kinds of data and have high availability

Build a distributed storage system:

- Scale
- Simple: key-value
- Highly available
- Guarantee Service Level Agreements (SLA)

Based on the SOSP 2007 presentation and paper: Dynamo: Amazon's Highly Available Key-value Store


Client requests
Page rendering components
...
Request routing
Aggregator services
Request routing
Services
Amazon S3
Other datastores
Dynamo instances

System Assumptions and Requirements

Query Model: simple read and write operations to a data item that is uniquely identified by a key

ACID Properties: Atomicity, Consistency, Isolation, Durability

Efficiency: latency requirements which are in general measured at the 99.9th percentile of the distribution

Other Assumptions: operation environment is assumed to be non-hostile and there are no security related requirements such as authentication and authorization

Service Level Agreements (SLA)

Application can deliver its functionality in bounded time:
Every dependency in the platform needs to deliver its functionality with even tighter bounds

Example: service guaranteeing that it will provide a response within 300ms for 99.9% of its requests for a peak client load of 500 requests per second

Design Consideration

Sacrifice strong consistency for availability

Conflict resolution is executed during ***read*** instead of ***write***

- Use quorums and other techniques

Other principles:

- Incremental scalability
- Symmetry
- Decentralization
- Heterogeneity

Summary of techniques used in *Dynamo* and their advantages

Problem	Technique	Advantage
Partitioning	Consistent Hashing	Incremental Scalability
High Availability for writes	Vector clocks with reconciliation during reads	Version size is decoupled from update rates.
Handling temporary failures	Sloppy Quorum and hinted handoff	Provides high availability and durability guarantee when some of the replicas are not available.
Recovering from permanent failures	Anti-entropy using Merkle trees	Synchronizes divergent replicas in the background.
Membership and failure detection	Gossip-based membership protocol and failure detection.	Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.

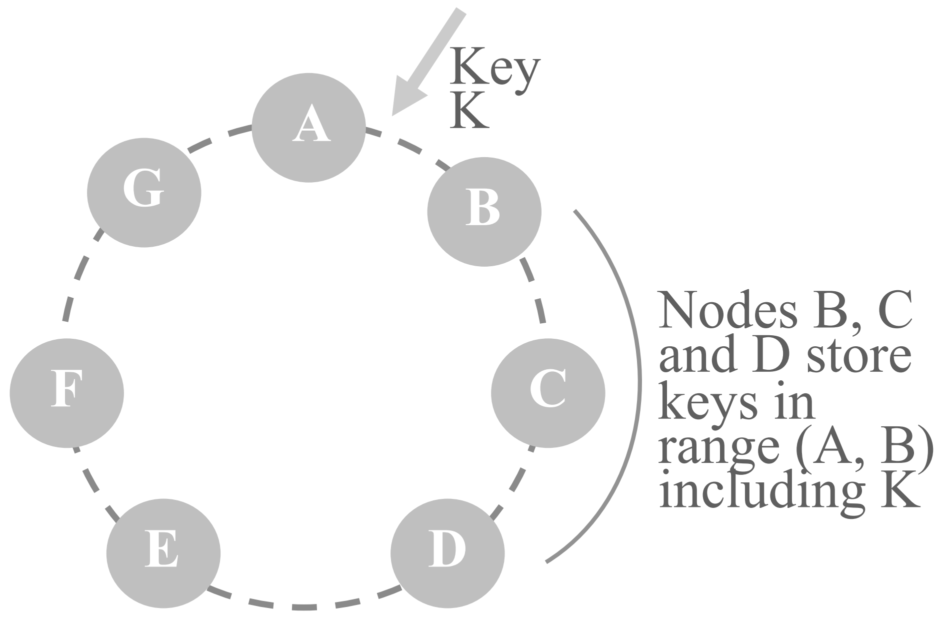
Dynamo Implementation

- Data Stores
  - Nodes in the system are spread around a logical circle
  - Nodes are responsible for the region between it and its predecessor
  - Virtual nodes are evenly dispersed and appear to be regular nodes in the system, but in reality are just handled by the nodes of the system
- Object Data
  - Uses hashing of an object’s key to determine where to store the object
  - Each object is replicated across N nodes (N-1 successor nodes to the coordinator node)

Partition Algorithm

Consistent hashing: the output range of a hash function is treated as a fixed circular space or “ring”.

”Virtual Nodes”: Each node can be responsible for more than one virtual node.

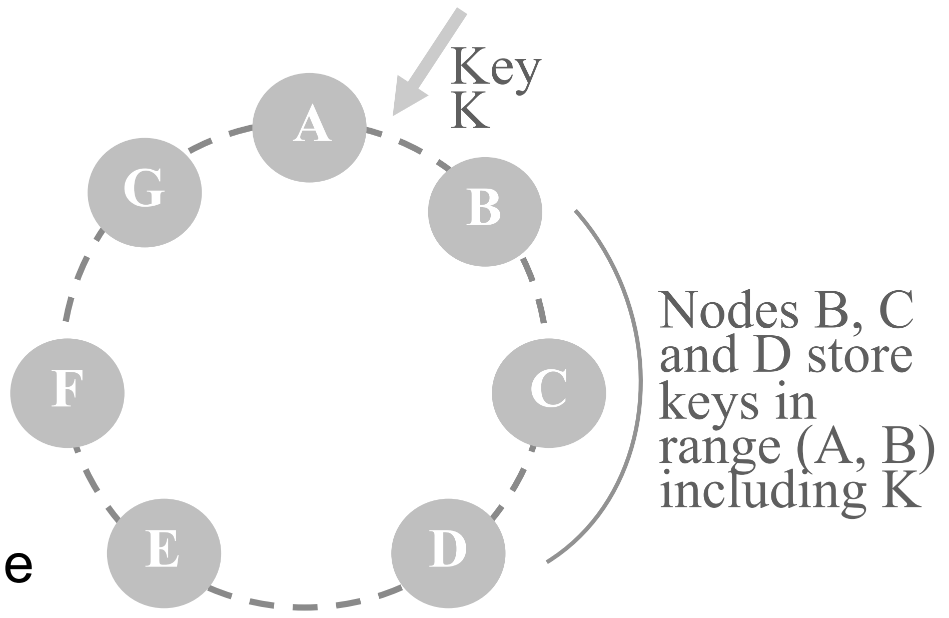


Advantages of using virtual nodes

If a node becomes unavailable the load handled by this node is evenly dispersed across the remaining available nodes

When a node becomes available again, the newly available node accepts a roughly equivalent amount of load from each of the other available nodes

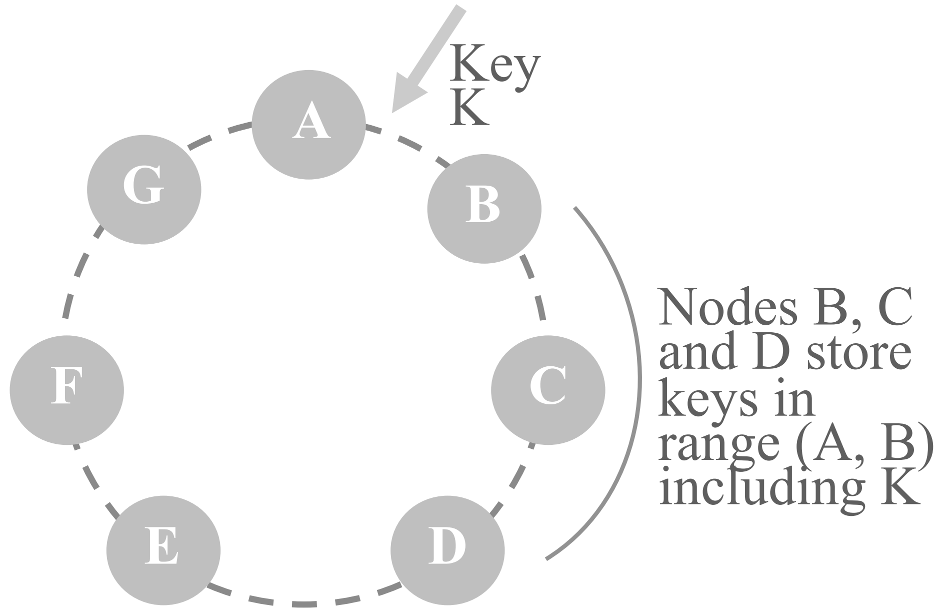
The number of virtual nodes that a node is responsible can decided based on its capacity, accounting for heterogeneity in the physical infrastructure



Replication

Each data item is replicated at N hosts

"*preference list*": The list of nodes that is responsible for storing a particular key



Data Versioning

A put() call may return to its caller before the update has been applied at all the replicas

A get() call may return many versions of the same object

Challenge: an object having distinct version sub-histories, which the system will need to reconcile in the future

Solution: uses **vector clocks** in order to capture causality between different versions of the same object

Vector Clock

A vector clock is a list of (node, counter) pairs

Every version of every object is associated with one vector clock

If the counters on the first object’s clock are less-than-or-equal to all of the nodes in the second clock, then the first is an ancestor of the second and can be forgotten

Sloppy Quorum

The sloppy quorum technique is used to handle temporal faults

R/W is the minimum number of nodes that must participate in a successful read/write operation

Setting **R + W > N** yields a quorum-like system.

In this model, the latency of a get (or put) operation is dictated by the slowest of the R (or W) replicas

R and W are usually configured to be less than N, to provide better latency

Hinted handoff

The hinted handoff is also used to handle temporal faults

Assume N = 3. When A is temporarily down or unreachable during a write, send replica to D

D is hinted that the replica is belong to A and it will deliver to A when A is recovered

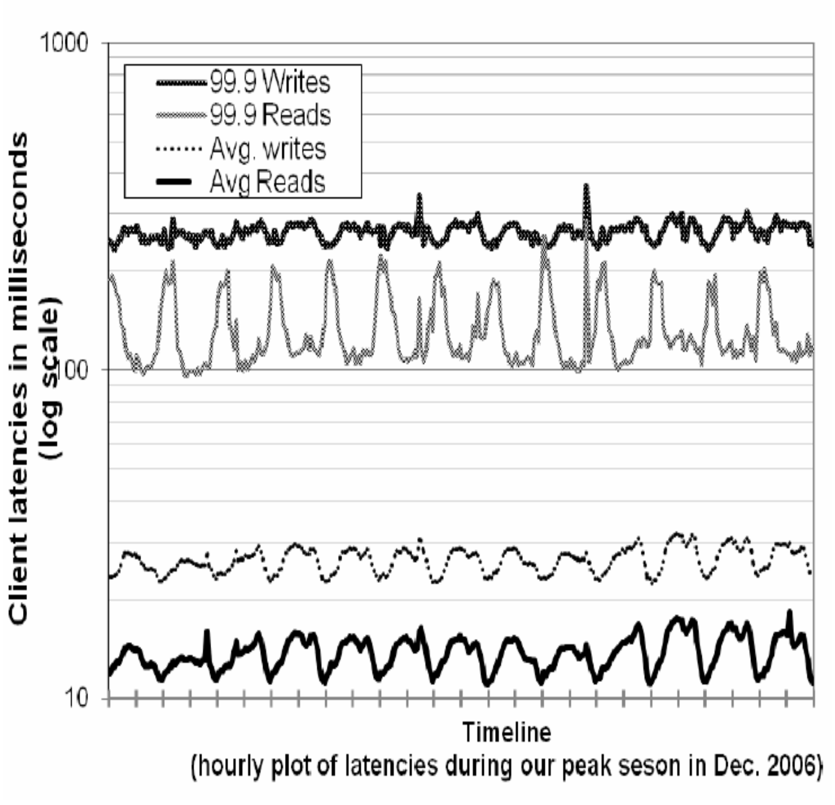
As a result A is always writable

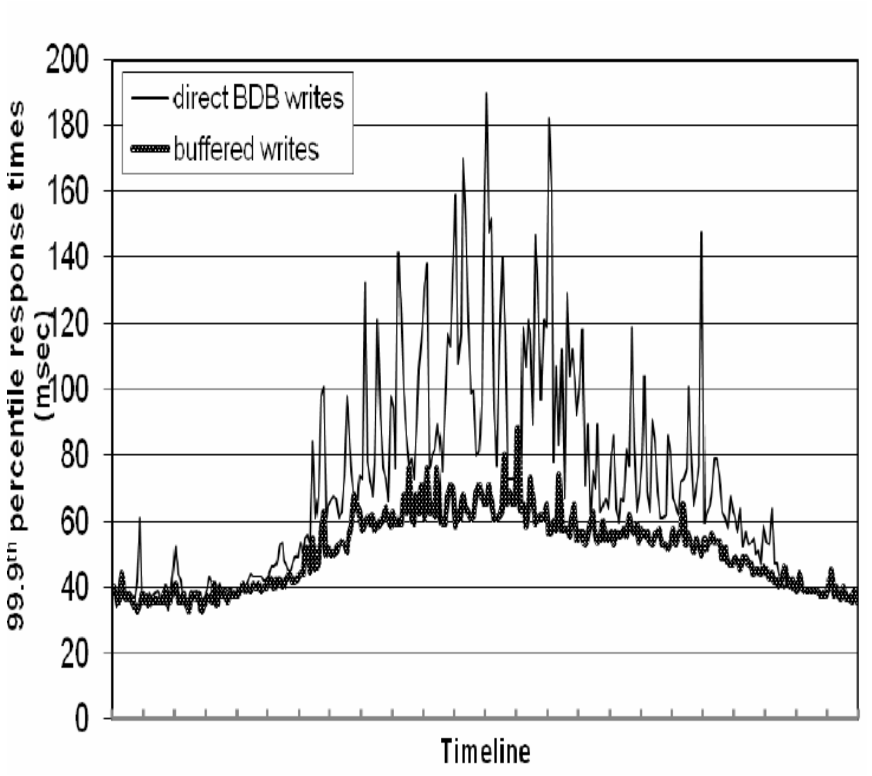
Dynamo Execution

- Writes
  - Requires generation of a new vector clock by coordinator
  - Coordinator writes locally
  - Forwards to N nodes, if W-1 respond then the write was successful
- Reads
  - Forwards to N nodes, if R-1 respond then forwards to user
  - Only unique responses forwarded
  - User handles merging if multiple versions exist

Results

Their response requirement is 300ms for any request (read or write)





Dynamo Summary

"Eventually" consistent data store

Always writable

Decentralized

All nodes have the same responsibilities

Amazon.com's Resolution

- Weakening consistency property in the system
- Increase the availability

Content Delivery Networks (CDN)

Geographically distributed network of Web servers around the globe (by an individual provider, E.g. Akamai).

Improve the performance and scalability of content retrieval.

Allow several content providers to replicate their content in a network of servers.

Motivation

Network cost

- Huge cost involved in setting up clusters of servers around the globe and corresponding increase in network traffic

Economic cost

- Higher cost per service rate making them inaccessible to lower and medium level customers

Social cost

- Monopolization of revenue

CDN Technology

Intelligent wide area traffic management

Direct clients' requests to optimal site based on topological proximity

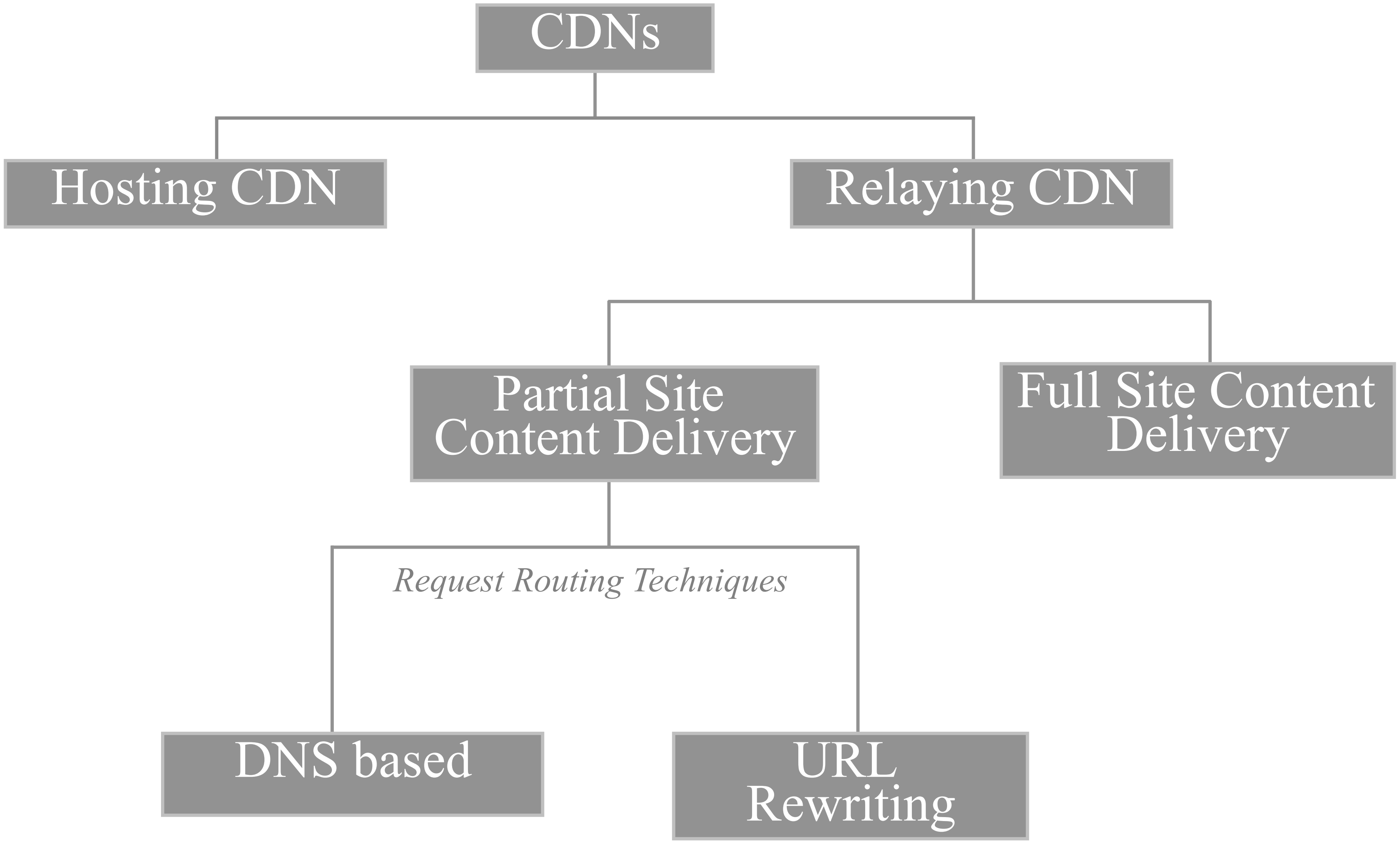
Two types of redirection: **DNS redirection** or **URL rewriting**

Cache

Saves useful contents in cache nodes.

Two cache policies: least frequently used standard and least recently used standard.

CDN Types *(Skeletal)*


CDNs
Hosting CDN
Relaying CDN
Partial Site Content Delivery
Full Site Content Delivery
Request Routing Techniques
DNS based
URL Rewriting

CDN

Replicate content on many servers

Challenges

- How to replicate content
- Where to replicate content
- How to find replicated content
- How to choose among known replicas

- How to direct clients towards replica
  - DNS, HTTP redirect, anycast, etc.

Akamai

Server Selection

Service and content is replicated in many places in network

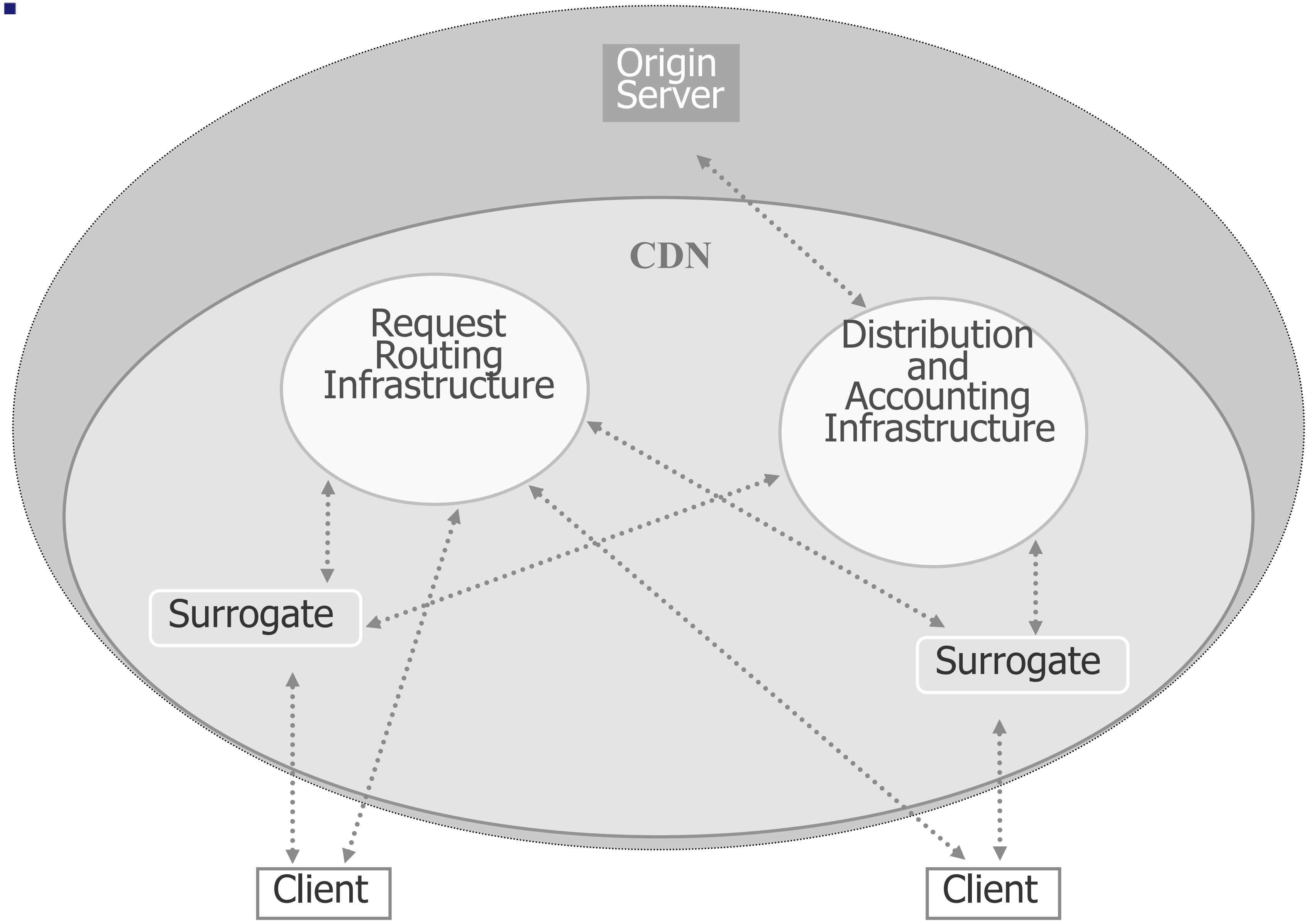
How to direct clients to a particular server?

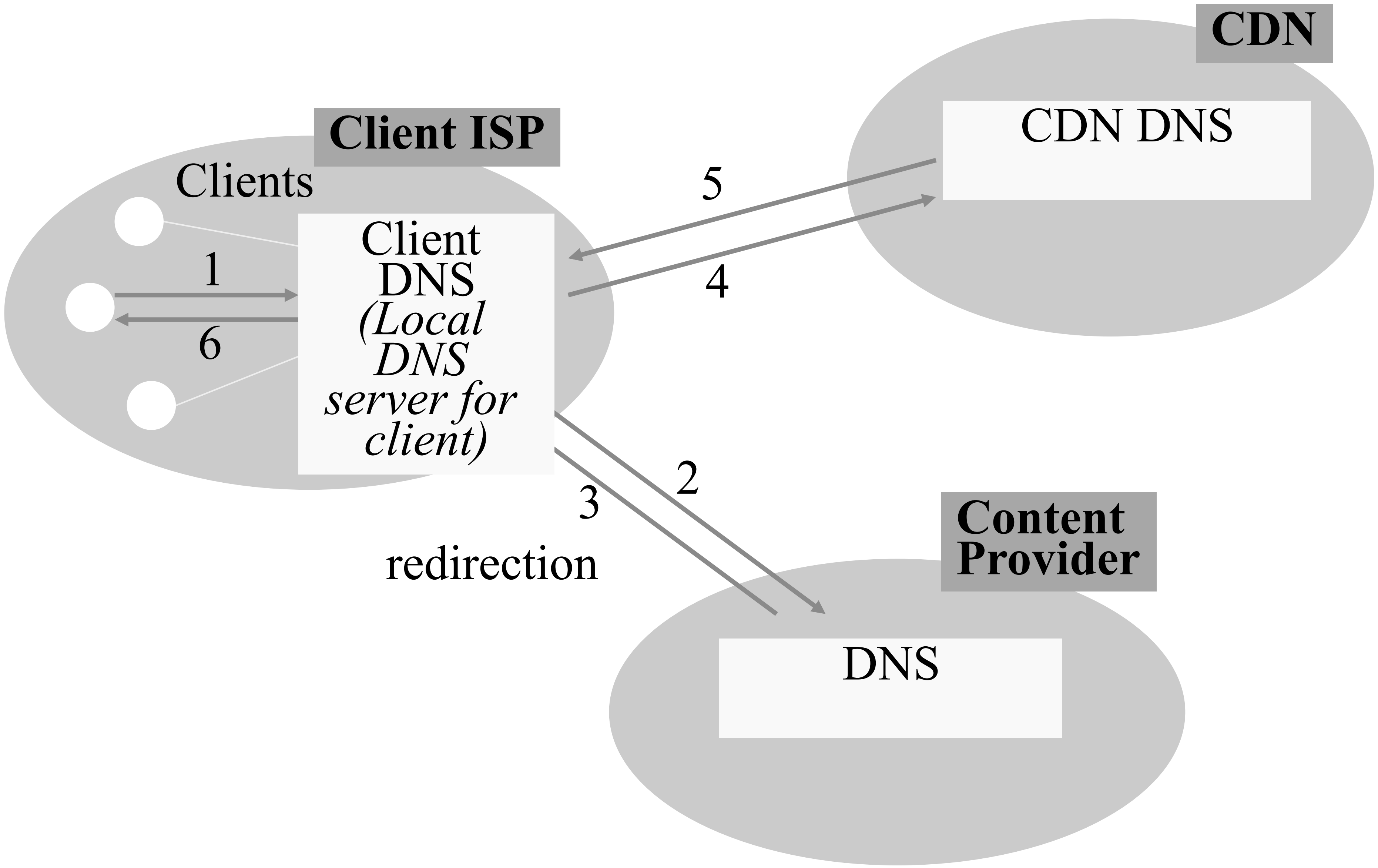
- As part of routing → anycast, cluster load balancing
- As part of application → HTTP redirect
- As part of naming → DNS

Which server to use?

- Best performance → to improve client performance
  - Based on Geography? RTT? Throughput? Load?
- Lowest load → to balance load on servers
- Any active node → to provide availability

CDN Architecture


Origin Server
CDN
Request Routing Infrastructure
Distribution and Accounting Infrastructure
Surrogate
Surrogate
Client
Client


CDN
CDN DNS
Client ISP
Clients
Client DNS *(Local DNS server for client)*
1
6
5
4
2
3
redirection
Content Provider
DNS

CDN	Type	Coverage	Solutions
Akamai	Commercial CDN service including streaming data	Market leader	Edge platform for handling static and dynamic content, DNS-based request-routing
Limelight Networks	Commercial On-demand distribution, live video, music, games, …	Surrogate servers in over 70 locations in the world	Edge-based solutions for content delivery, streaming support, custom CDN for custom delivery solutions, DNS-based request-routing
Coral	Academic Content replication based on popularity (on demand), addresses flash crowds	Experimental, hosted on PlanetLab	Uses a DHT algorithm (Kademlia), support for static content, DNS-based request-routing
CoDeeN	Academic testbed Caching of content and redirection of HTTP requests	Experimental, hosted on PlanetLab, collaborative CDN	Support for static content, HTTP direction Consistent hashing for mapping data to servers
Globule	Academic Replication of content, server monitoring, redirection to available replicas	Apache extension, Open Source collaborative CDN	Support for static content, monitoring services, DNS-based request-routing

Akamai

Clients fetch html document from primary server
URLs for replicated content are replaced in html

Client resolves aXYZ.g.akamaitech.net hostname

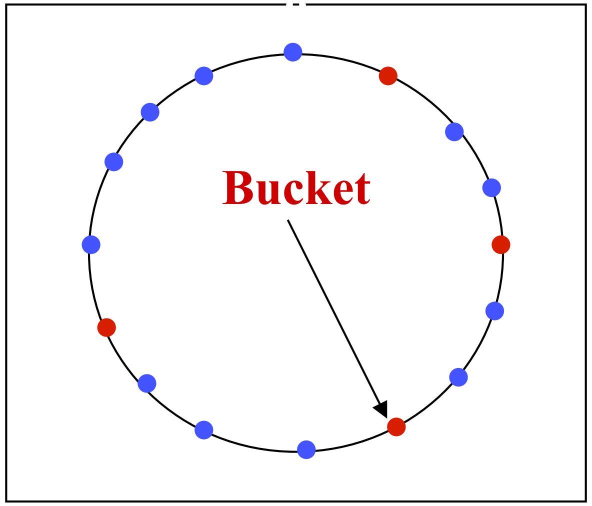
Akamai.net name server returns NS record for g.akamaitech.net
- G.akamaitech.net nameserver choses server in region

- Should try to chose server that has file in cache - How to choose?

- Uses aXYZ name and consistent hash

Consistent Hash Revisited

- Construction
  - Assign each of C hash buckets to random points on mod 2^n circle, where, hash key size = *n.*
  - Map object to random position on circle
  - Hash of object = closest clockwise bucket

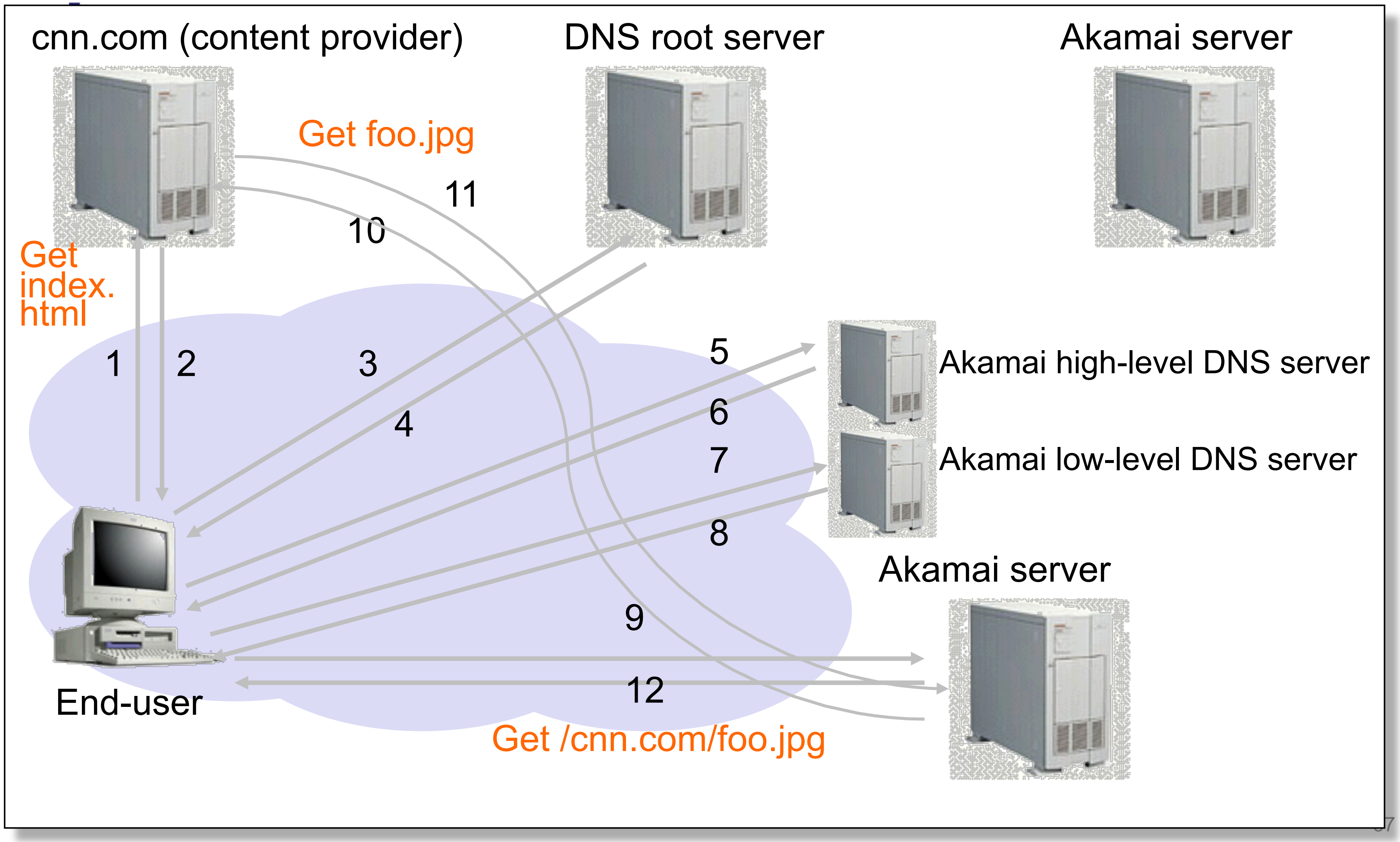


Smoothness → addition of bucket does not cause much movement between existing buckets

Spread & Load → small set of buckets that lie near object

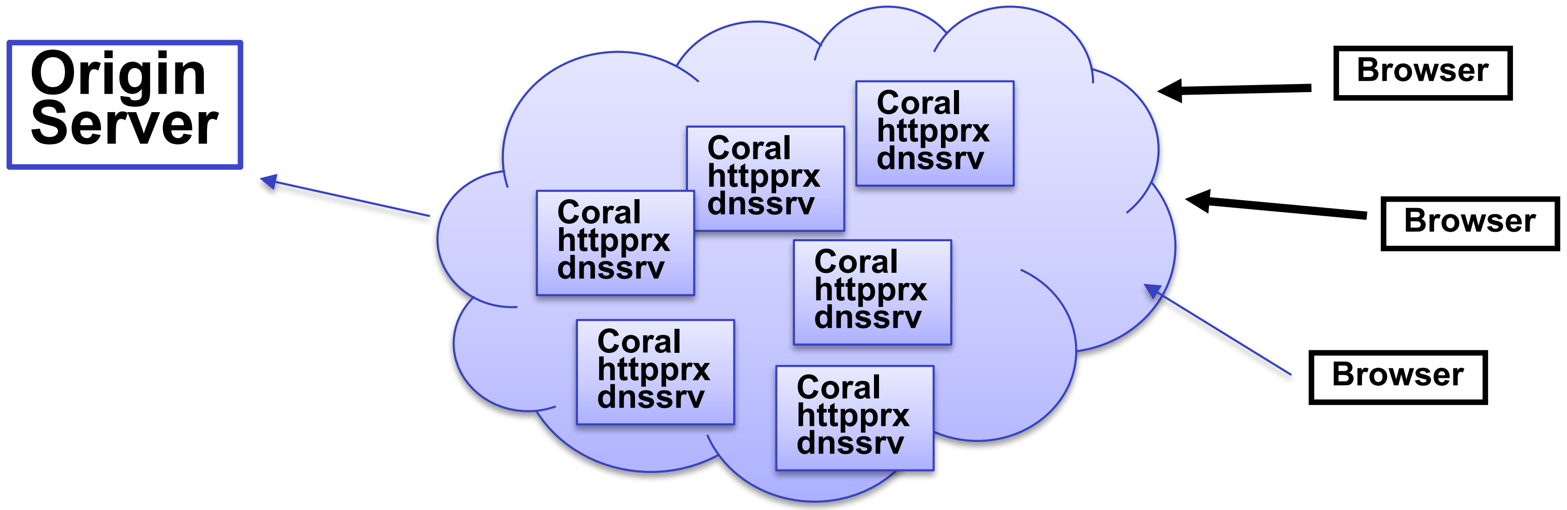
Balance → no bucket is responsible for large number of objects

How Akamai Works



Source: www.cs.cmu.edu/~srini/15-744/S08/lectures/17-DNS.ppt

Coral: An Open CDN



Pool resources to dissipate flash crowds

Implement an open CDN

Allow anybody to contribute

Works with unmodified clients

CDN only fetches once from origin server

Based on NSDI 2004 presentation and paper

Using CoralCDN

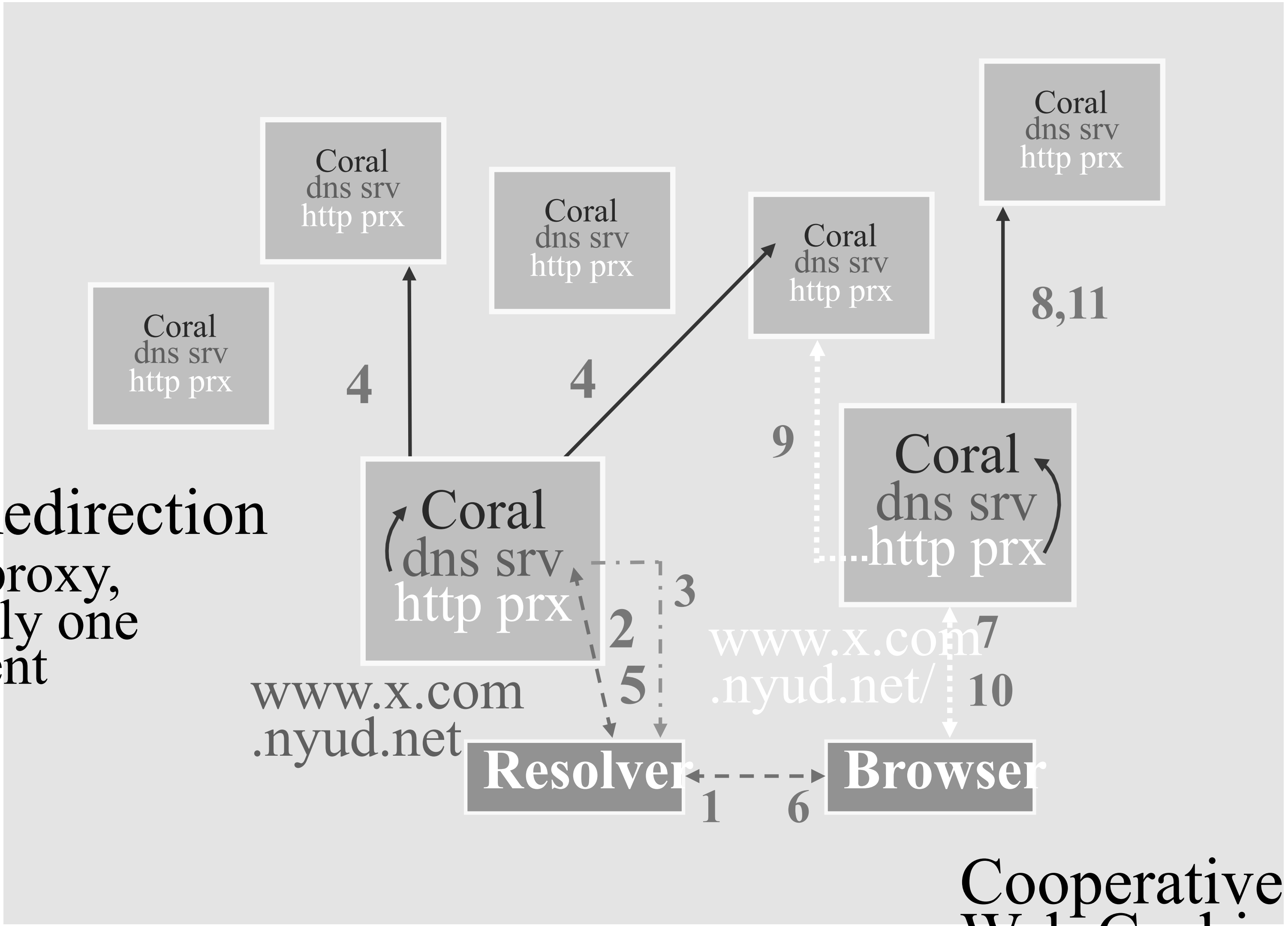
Rewrite URLs into “Coralized” URLs

www.x.com → www.x.com.nyud.net:8090

Coral distributes the load

Who might “Coralize” URLs?

Web server operators Coralize URLs
Coralized URLs posted to portals, mailing lists
Users explicitly Coralize URLs

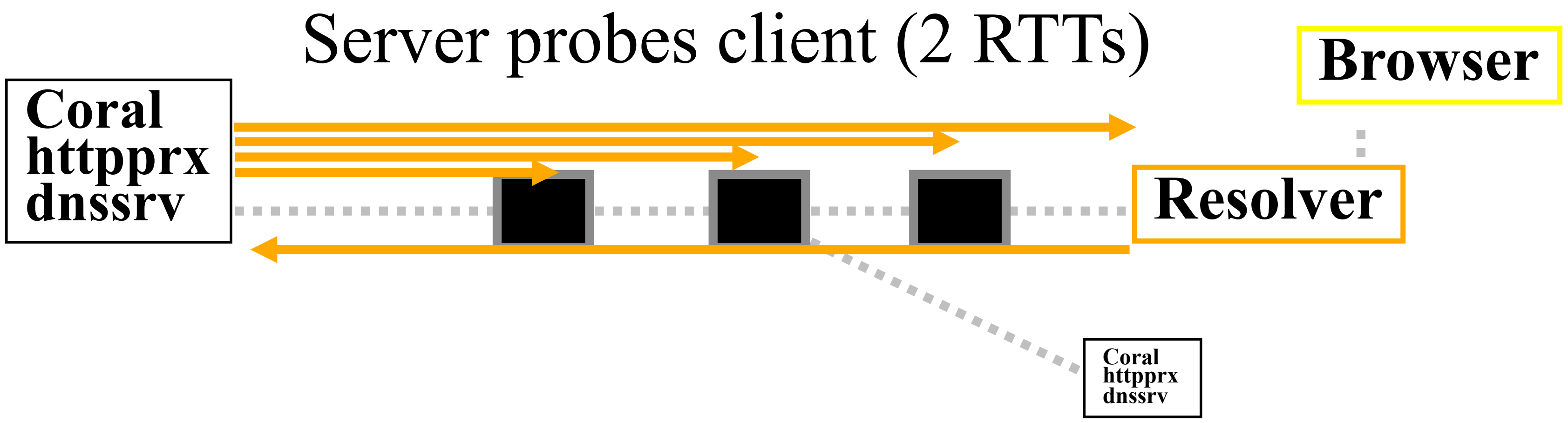


DNS Redirection

Return proxy, preferably one near client

Cooperative Web Caching

DNS measurement mechanism



Return servers within appropriate cluster

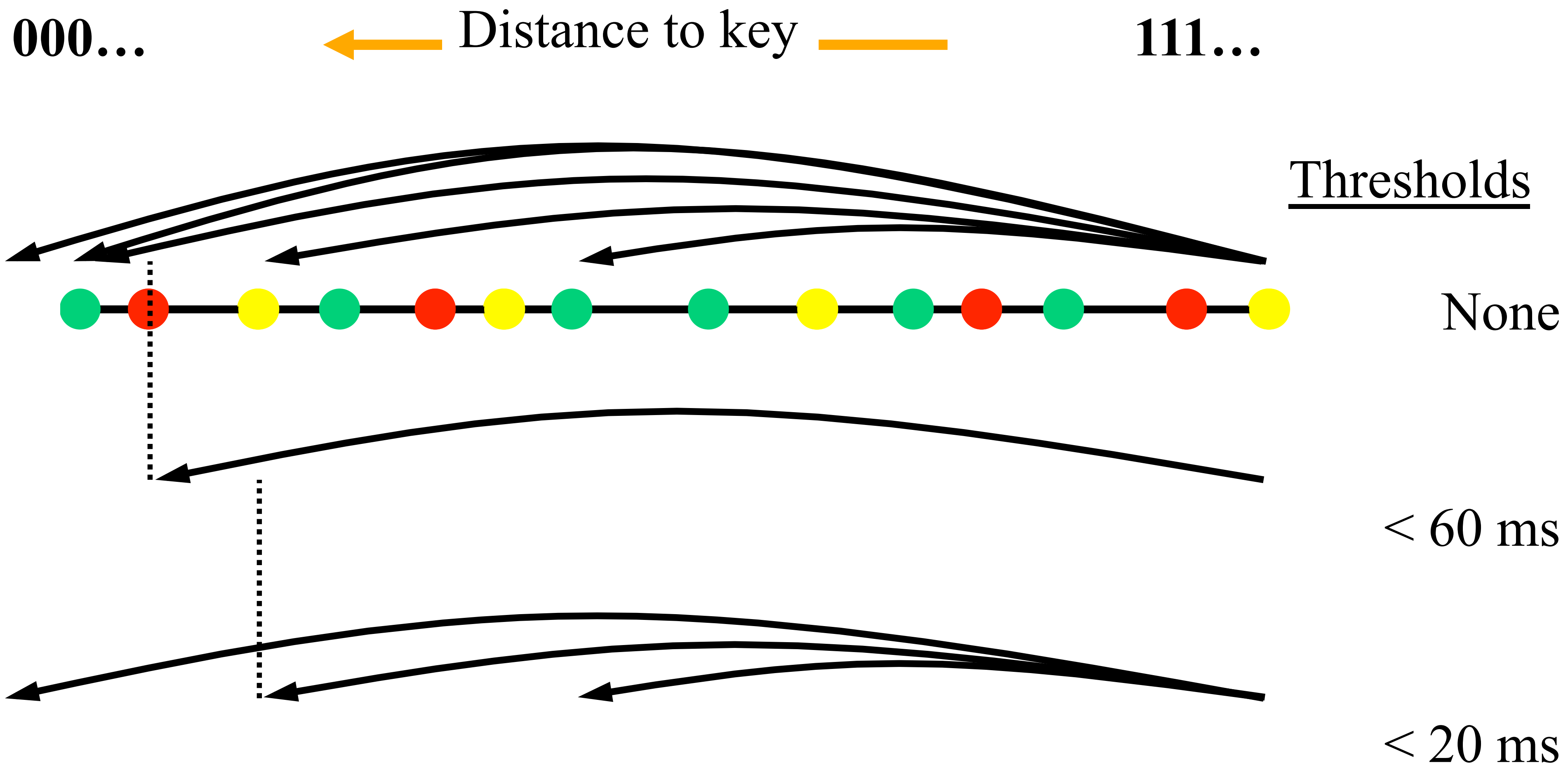
- e.g., for resolver RTT = 19 ms, return from cluster < 20 ms

Use network hints to find nearby servers

- i.e., client and server on same subnet

Otherwise, take random walk within cluster

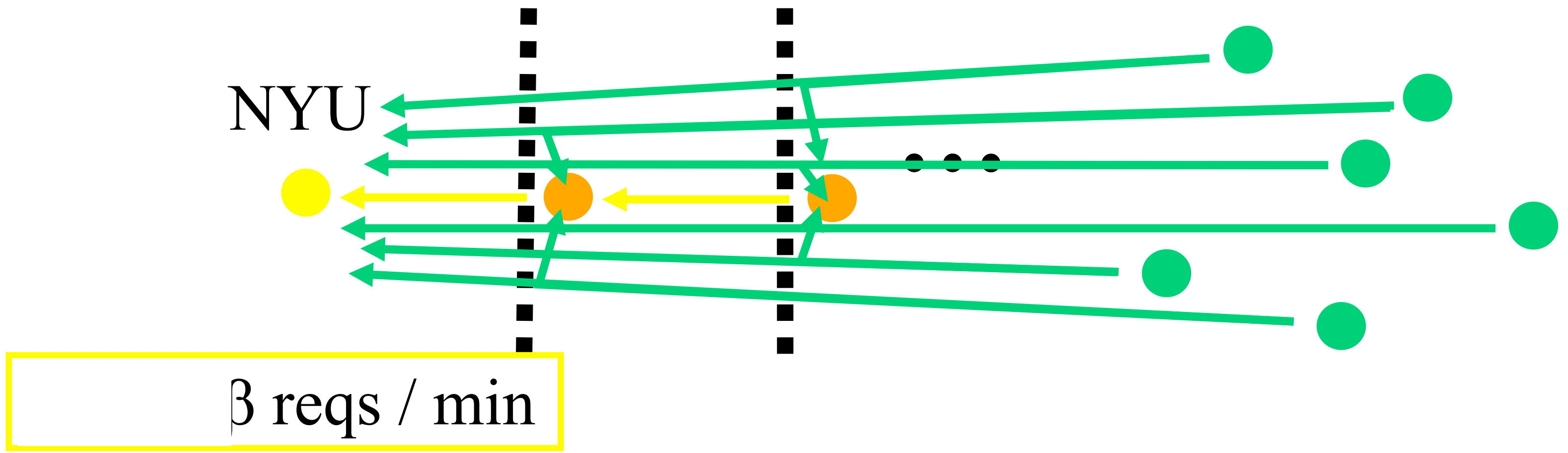
Key-based XOR routing



Minimizes lookup latency
Prefer values stored by nodes within faster clusters

Prevent insertion hotspots

- Store value once in each level cluster
  - Always storing at closest node causes hotspot



Halt put routing at full and loaded node

Full → M vals/key with TTL > ½ insertion TTL

Loaded → β puts traverse node in past minute

Store at furthest, non-full node seen

Challenges for DNS Redirection

Coral lacks…

- Central management
- *A priori* knowledge of network topology
  - Anybody can join system
- Any special tools (e.g., BGP feeds)

Coral has…

- Large number of vantage points to probe topology
- Distributed index in which to store network hints
- Each Coral node maps nearby networks to self

Coral's DNS Redirection

Coral DNS server probes resolver

Once local, stay local

- When serving requests from nearby DNS resolver
  - Respond with nearby Coral proxies
  - Respond with nearby Coral DNS servers
    - → Ensures future requests remain local

Else, help resolver find local Coral DNS server

Applications for DHTs

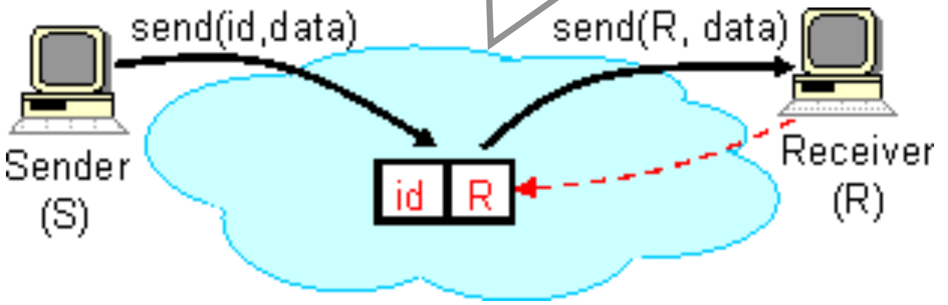
- DHTs are used as a basic building block for an application-level infrastructure
  - Internet Indirection Infrastructure (i3)
    - New forwarding infrastructure based on Chord
  - DOA (Delegation Oriented Architecture)
    - New naming and addressing infrastructure based on overlays
  - OpenDHT: A publicly accessible distributed hash table (DHT) service

Internet Indirection Infrastructure (i3)

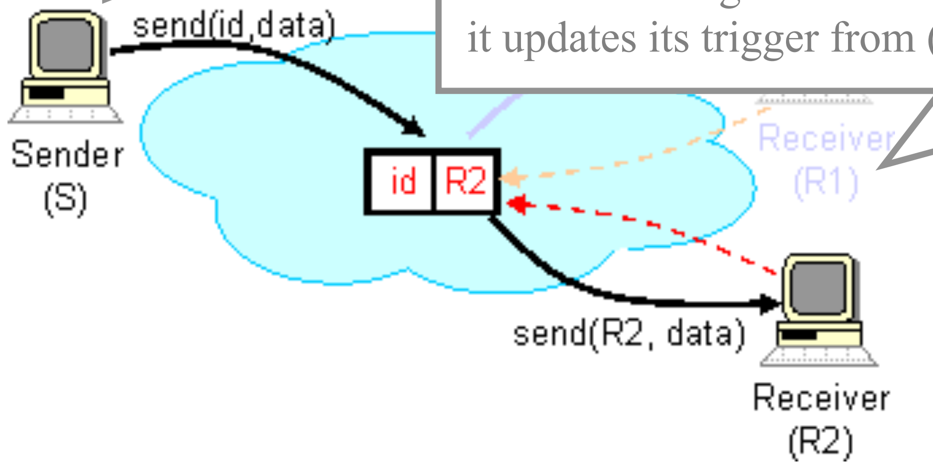
- A DHT - based overlay network
  - Based on Chord
- Aims to provide more flexible communication model than current IP addressing
- Also a forwarding infrastructure
  - i3 packets are sent to identifiers
  - each identifier is routed to the i3 node responsible for that identifier
  - the node maintains triggers that are installed by receivers
  - when a matching trigger is found the packet is forwarded to the receiver

i3 II

- An i3 identifier may be bound to a host, object, or a session
- i3 has been extended with ROAM
  - Robust Overlay Architecture for Mobility
  - Allows end hosts to control the placement of rendezvous-points (indirection points) for efficient routing and handovers
  - Legacy application support
    - user level proxy for encapsulating IP packets to i3 packets



R inserts a trigger (id, R) and receives all packets with identifier id.

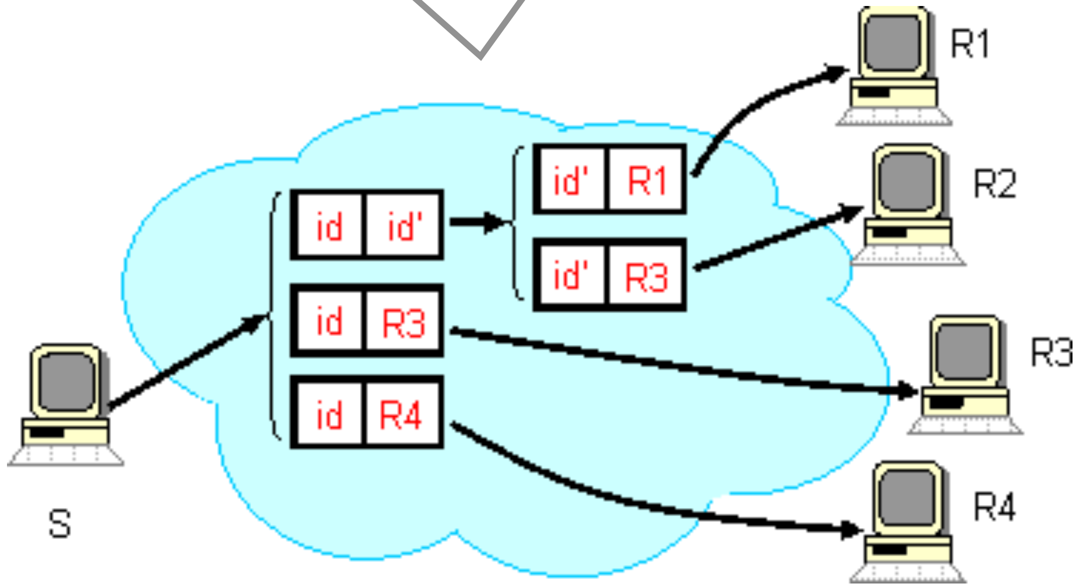


Mobility is transparent for the sender

the host changes its address from R1 to R2, it updates its trigger from (id, R1) to (id, R2).

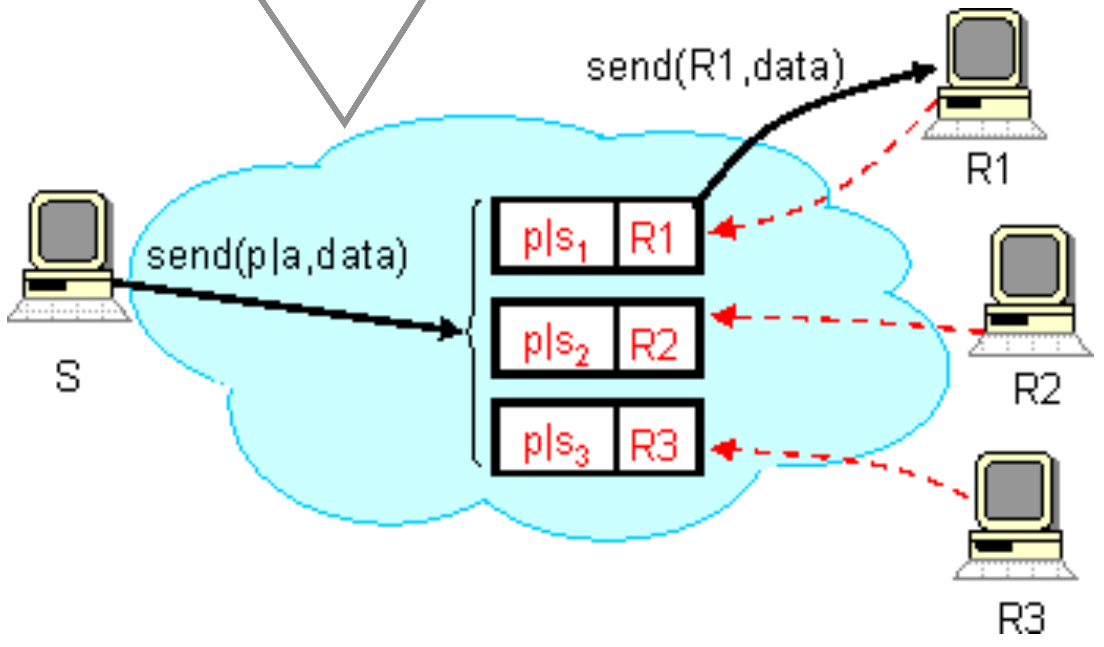
Source: http://i3.cs.berkeley.edu/

A multicast tree using a hierarchy of triggers



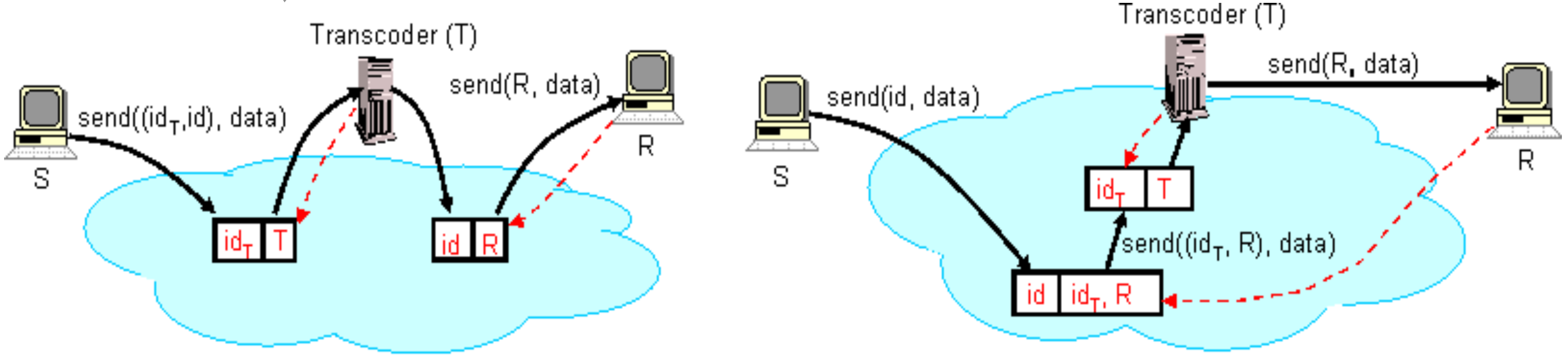
Source: http://i3.cs.berkeley.edu/

Anycast using the longest matching prefix rule.



Source: http://i3.cs.berkeley.edu/

Sender-driven service composition using a stack of identifiers



(a) Sender-driven service composition

(b) Receiver-driven service composition

Receiver-driven service composition using a stack of identifiers

Source: http://i3.cs.berkeley.edu/

OpenDHT

- A publicly accessible distributed hash table (DHT) service.
- OpenDHT runs on a collection of 200 - 300 nodes on PlanetLab.
- Clients do not need to participate as DHT nodes.
- A test bed infrastructure
  - Open infrastructure for puts and gets
  - Organizing clients that handle application upcalls
- OpenDHT: A Public DHT Service and Its Uses. Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. Proceedings of ACM SIGCOMM 2005, August 2005.

Summary

Key applications

- Akamai (consistent hashing)
- Amazon (Dynamo, consistent hashing, ring geometry)
- Kademlia (XOR geometry)
- Coral (XOR geometry)
- BitTorrent, Gnutella, Freenet

Key concepts

- Small worlds, hierarchy, consistent hashing, DHTs

Grading

Course grading will be based on the final exam and the assignments

Up to 20% bonus based on exercises

Possible to get extension to the deadline of the last assignment (please ask if you have questions about the simulator, will be discussed on Thursday)

The exam will be held on 20.10. 16-19 in A111

Main theme	**Prerequisites**	**Approaches learning goals**	**Meets learning goals**	**Deepens learning goals**
Overlay and peer-to-peer networks: definitions and systems	Basics of data communications and distributed systems (Introduction to Data Communications, Distributed Systems)	Knowledge of how to define the concepts of overlay and peer-to-peer networks, and state their central features Ability to describe at least one system in detail	Ability of being able to compare different overlay and p2p networks in a qualitative manner Ability to assess the suitability of different systems to different use cases	Ability to give one's own definition of the central concepts and discuss the key design and deployment issues
Distributed hash tables	Basics of data communications and distributed systems (Introduction to Data Communications, Distributed Systems) Big-O-notation and basics of algorithmic complexity	Knowledge of the concepts of structured and unstructured networks and the ability to classify solutions into these two categories Knowledge of the basics of distributed hash tables Ability to describe at least one distributed hash table algorithm in detail	Ability of being able to compare different distributed hash table algorithms Ability of designing distributed hash table-based applications Knowledge of key performance issues of distributed hash table systems and the ability to analyze these systems	The knowledge of choosing a suitable distributed hash table design for a problem Familiarity with the state of the art
Reliability and performance modelling	Basics of probability theory Basics of reliability in distributed systems	Ability to model and assess the reliability of overlay and peer-to-peer networks by using probability theory Knowledge of the most important factors pertaining to reliability	Ability of analytically analyzing the reliability and performance of overlay and peer-to-peer networks Understanding of the design issues that are pertinent for reliable systems	Familiarity with the state of the art
Content distribution	Introduction to Data Communications	Knowledge of the basic content distribution solutions Ability to describe at least one overlay and p2p network based content distribution solution	Knowledge of different content distribution systes and the ability to compare them in detail Knowledge of several content distribution techniques	Familiarity with the state of the art
Security	Basics of computer security	Knowledge of the basic security issues with overlay and p2p networks Knowledge of the sybil attack concept	Ability to discuss how security problems and limitations can be solved Knowledge of how to prevent sybil attacks	Knowledge of how to prevent sybil attacks Familiarity with the state of the art