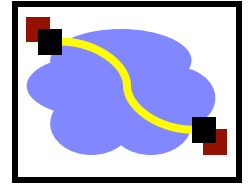


# 416 Distributed Systems

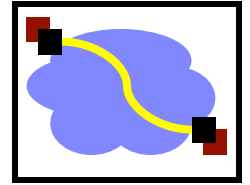
Feb 15, 2017 – CDNs

# Outline



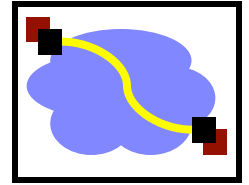
- DNS Design (317)
- **Content Distribution Networks**

# Typical Workload (Web Pages)

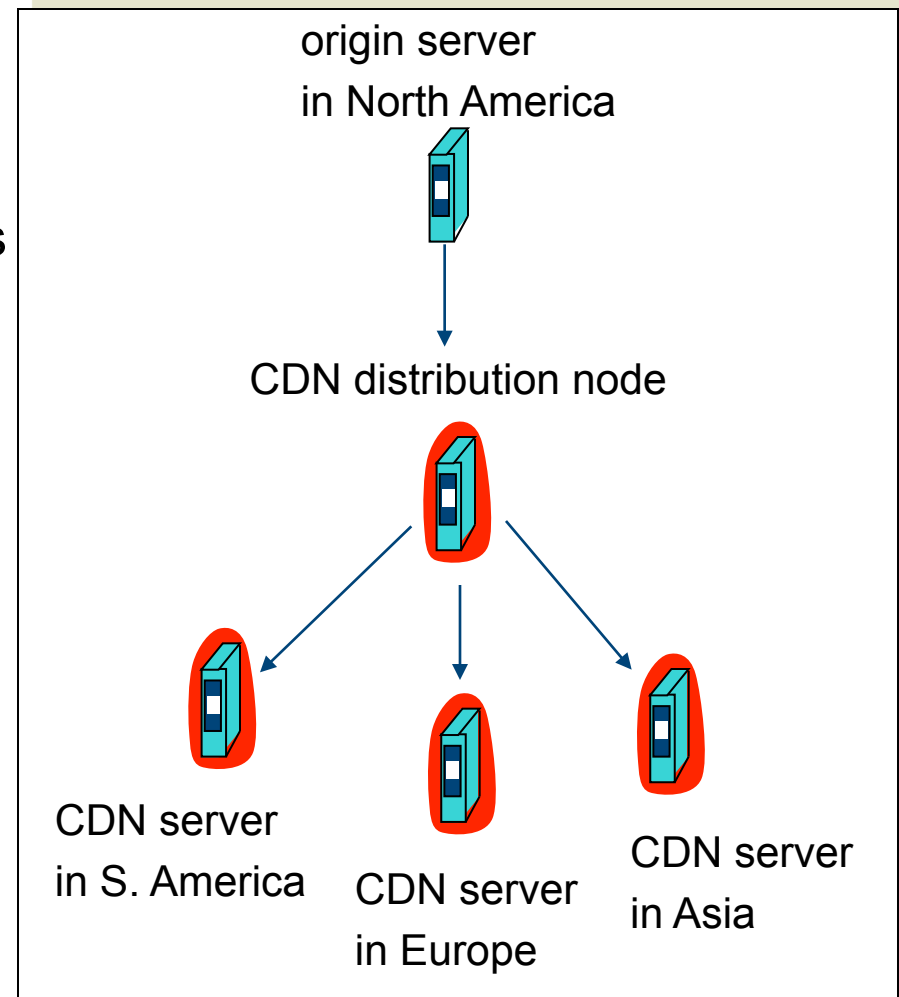


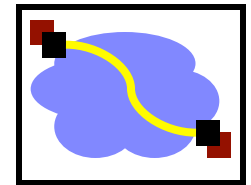
- Multiple (typically small) objects per page
- File sizes are heavy-tailed
- Embedded references
- This plays havoc with performance. Why?
- Solutions?
  - Lots of small objects & TCP
    - 3-way handshake
    - Lots of slow starts
    - Extra connection state

# Content Distribution Networks (CDNs)



- The content providers are the CDN customers.
- Content replication
- CDN company installs hundreds of CDN servers throughout Internet
  - Close to users
- CDN replicates its customers' content in CDN servers. When provider updates content, CDN updates servers





## ▼ Sections

[Main](#)

[Apple](#)

[AskSlashdot](#)

[Books](#)

[Developers](#)

[Games](#)

[Hardware](#)

[IT](#)

[Idle](#)

[Index](#)

[Interviews](#)

## Political Sites Scale Up For Election Traffic

Posted by [timothy](#) on Tuesday November 04, @12:15PM

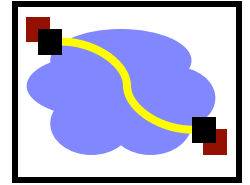
from the [hearken-are-those-trumpets](#) dept.

[miller60](#) writes

"News sites and political blogs are expecting extraordinary traffic tonight as Americans track results of the Presidential election, and are [scaling their infrastructure](#) to meet the challenge. Yahoo anticipates its Election Night traffic may be [three times the volume](#) seen in 2004, when it had 80 million page views on Election Day and 142 million more visits the following day. Hosting companies say customers have been ordering extra servers and load balancing services, while content delivery networks are also expecting a busy night. Will traffic approach record levels? Akamai's [Net Usage Index](#), which tracks traffic to its customer news sites, is one metric to watch."

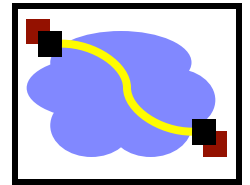


# Content Distribution Networks & Server Selection



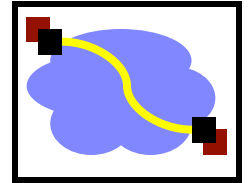
- Replicate content on many servers
- Challenges
  - How to replicate content
  - Where to replicate content
  - How to find replicated content
  - How to choose among known replicas
  - How to direct clients towards replica

# Server Selection



- Which server?
  - Lowest load → to balance load on servers
  - Best performance → to improve client performance
    - Based on Geography? RTT? Throughput? Load?
  - Any alive node → to provide fault tolerance
- How to direct clients to a particular server?
  - As part of routing → anycast, cluster load balancing
    - Not covered ☹️
  - As part of application → HTTP redirect
  - As part of naming → DNS

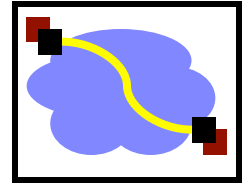
# Application Based



- HTTP supports simple way to indicate that Web page has moved (30X responses)
- Server receives Get request from client
  - Decides which server is best suited for particular client and object
  - Returns HTTP redirect to that server
- **Can make informed application specific decision**
- May introduce additional overhead →  
multiple connection setup, name lookups, etc.

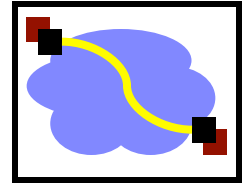


# Naming Based



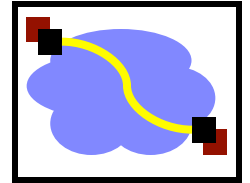
- Client does name lookup for service
- Name server chooses appropriate server address
  - DNS A-record returned is “best” one for the client
- What information can name server base decision on?
  - Server load/location → must be collected
  - Information in the name lookup request
    - Name service client → typically the local name server for client (not the client itself)

# How Akamai Works



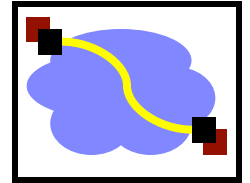
- Clients fetch html document from primary server
  - E.g. fetch index.html from cnn.com
- URLs for replicated content are replaced in html
  - E.g. `` replaced with ``
- Client is forced to **DNS resolve** `aXYZ.g.akamaitech.net` hostname

# How Akamai Works



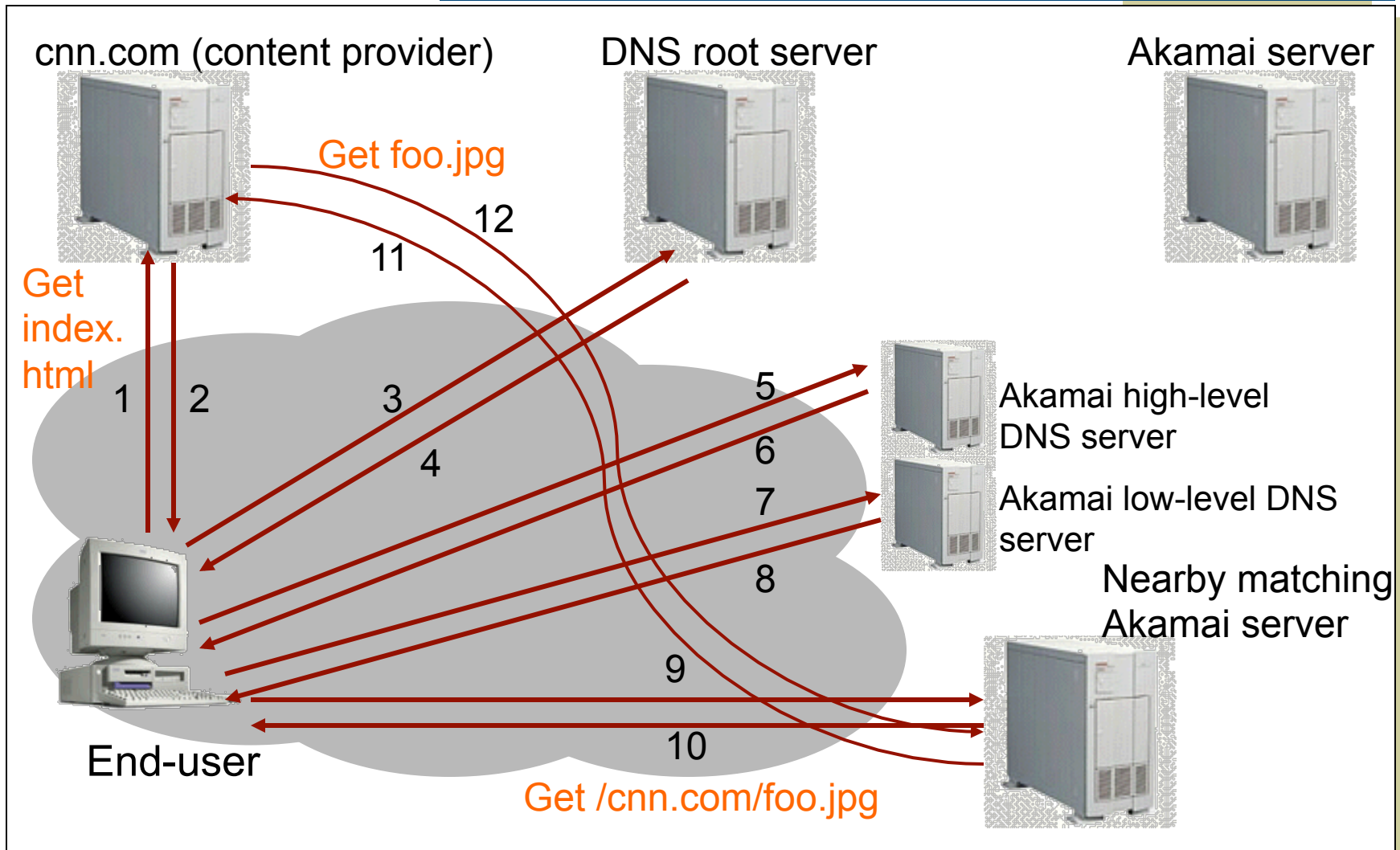
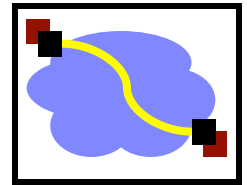
- Akamai only replicates static content (\*)
  - Modified name contains original file name
  - Akamai server is asked for content
    - First checks local cache
    - If not in cache, requests file from primary server and caches file
- \* (At least, the version we're talking about today. Akamai actually lets sites write code that can run on Akamai's servers, but that's a pretty different beast)

# How Akamai Works

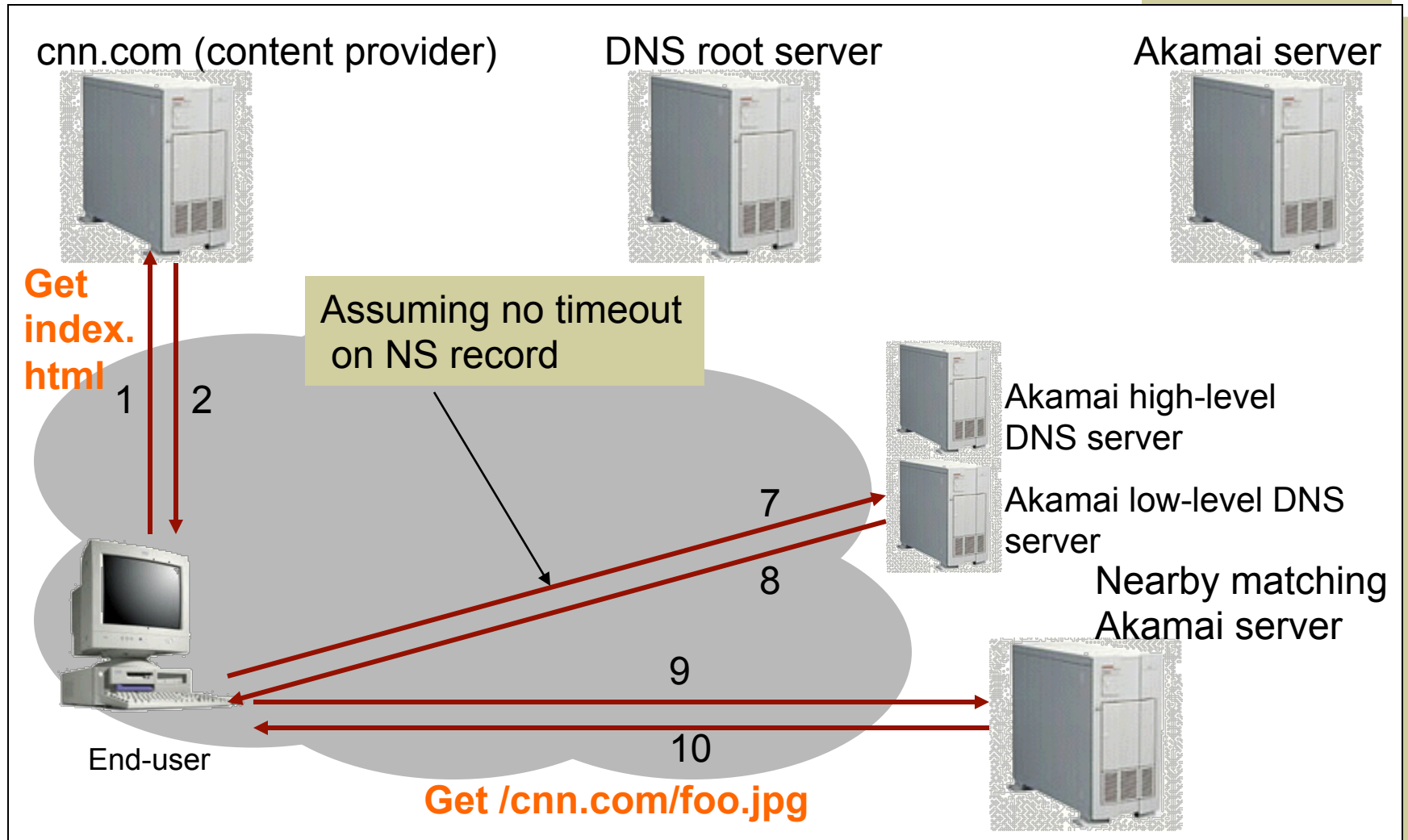
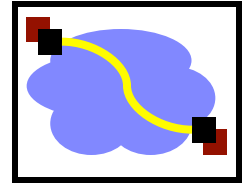


- Root server gives NS record for akamai.net
- Akamai.net name server returns NS record for g.akamaitech.net
  - Name server chosen to be in region of client's name server
  - TTL is large
- G.akamaitech.net nameserver chooses server in region
  - Should try to chose server that has file in cache - How to choose?
  - Uses aXYZ name and hash
  - TTL is small → why?

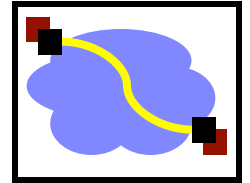
# How Akamai Works



# Akamai – Subsequent Requests

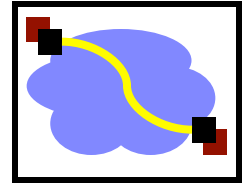


# Simple Hashing



- Given document XYZ, we need to choose a server to use
- Suppose we use modulo
- Number servers from  $1 \dots n$ 
  - Place document XYZ on server  $(XYZ \bmod n)$
  - What happens when a servers fails?  $n \rightarrow n-1$ 
    - Same if different people have different measures of  $n$
  - Why might this be bad?

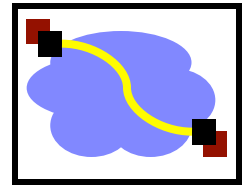
# Consistent Hash



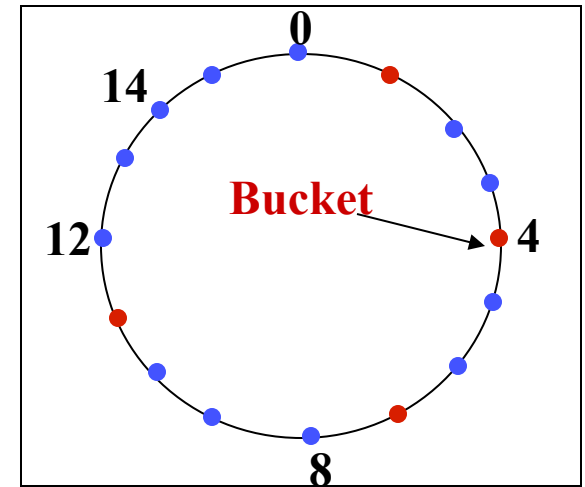
- “view” = subset of all hash buckets that are visible (a bucket is e.g., a server)
- Desired features
  - Smoothness – little impact on hash bucket contents when buckets are added/removed
  - Spread – small set of hash buckets that may hold an object regardless of views
  - Load balance – across all views # of objects assigned to hash bucket is small



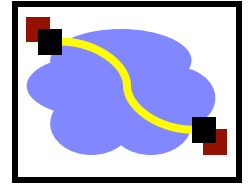
# Consistent Hash – Example



- 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 unit interval
  - Hash of object = closest bucket
- Smoothness → addition of bucket does not cause movement between existing buckets
- Spread → small set of buckets that lie near object
- Load balance → no bucket is responsible for large number of objects



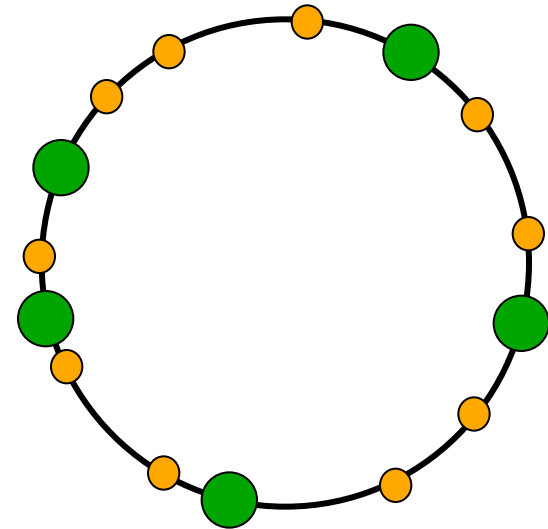
# Consistent Hashing



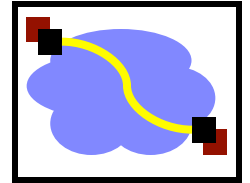
- **Main idea:**

- map both **keys** and **nodes** to the same (metric) identifier space
- find a “rule” how to assign keys to nodes

**Ring is one option.**

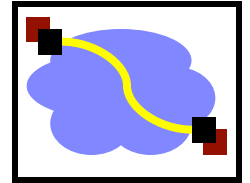


# Consistent Hashing



- The consistent hash function assigns each node and key an  $m$ -bit identifier using SHA-1 as a base hash function
- **Node identifier:** SHA-1 hash of IP address
- **Key identifier:** SHA-1 hash of key

# Identifiers



- $m$  bit identifier space for both keys and nodes
- **Key identifier:** SHA-1(key)

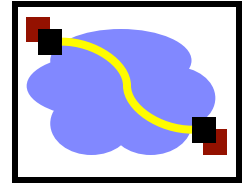
Key="LetItBe"  $\xrightarrow{\text{SHA-1}}$  ID=60

- **Node identifier:** SHA-1(IP address)

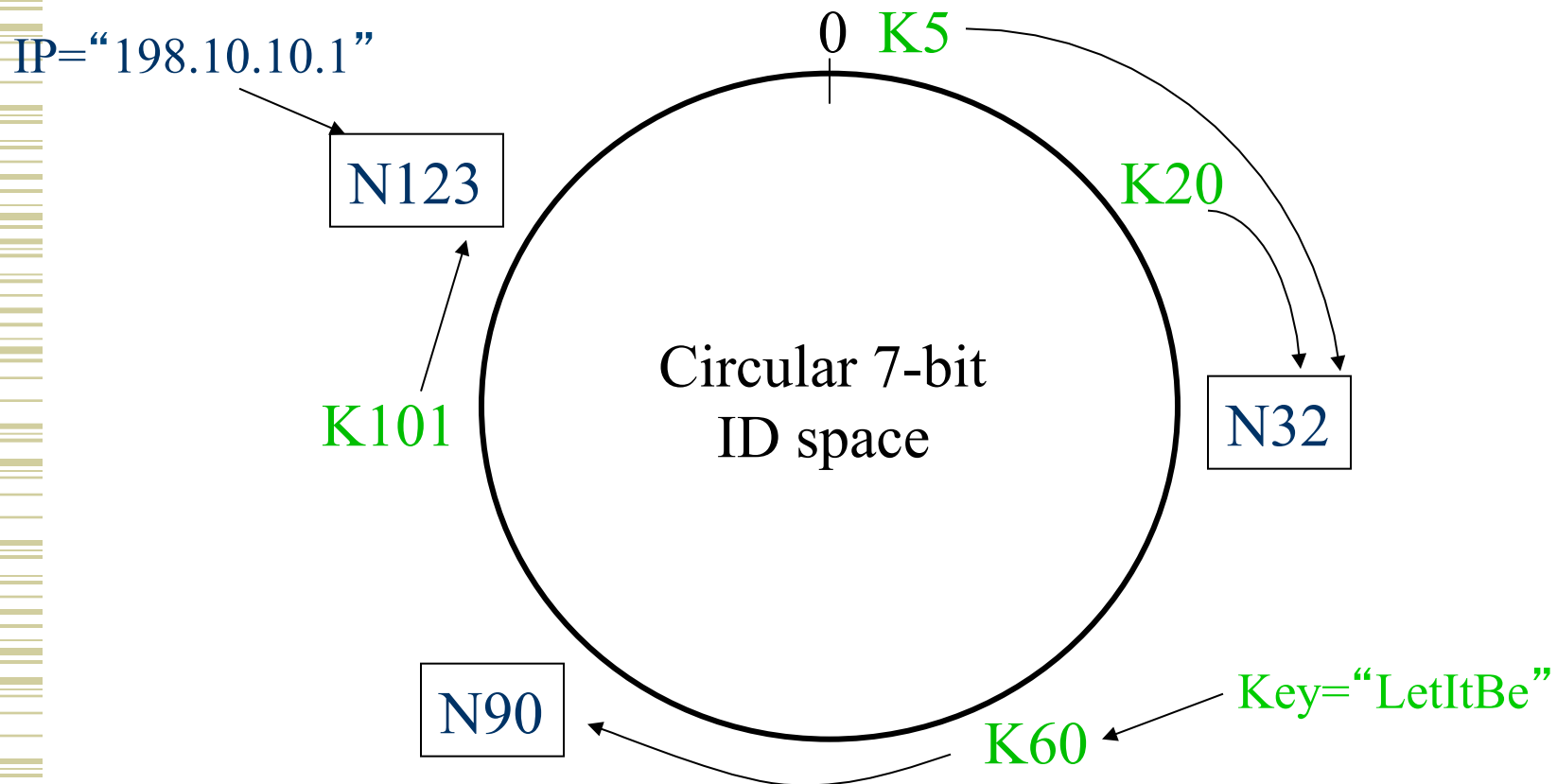
IP="198.10.10.1"  $\xrightarrow{\text{SHA-1}}$  ID=123

- How to map key IDs to node IDs?

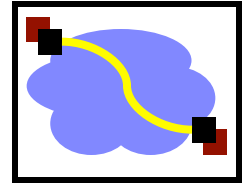
# Consistent Hashing Example



**Rule:** A key is stored at its **successor**: node with next higher or equal ID

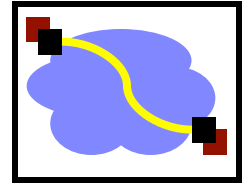


# Consistent Hashing Properties



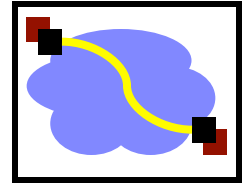
- **Load balance:** all nodes receive roughly the same number of keys
- For  $N$  nodes and  $K$  keys, with high probability
  - each node holds at most  $(1+\varepsilon)K/N$  keys
  - (provided that  $K$  is large enough compared to  $N$ )

# Load Balance



- Redirector knows all CDN server Ids
- Can track approximate load (or delay)
- To balance load:
  - $\text{Load}_i = \text{Hash}(\text{URL}, \text{ip of server}_i)$  for all  $i$
  - Sort  $\text{Load}_i$  from high to low
  - find first server with low enough load
- Benefits?
- How should “load” be measured?

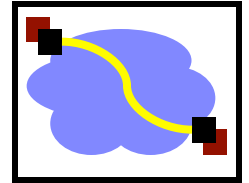
# Consistent Hashing not just for CDN



- Finding a nearby server for an object in a CDN uses centralized knowledge.
- Consistent hashing can also be used in a distributed setting
- P2P systems like BitTorrent, need a way of finding files.
- Consistent Hashing to the rescue
  - Need a way to route in a decentralized way between nodes; but easy to come up with a distance metric!

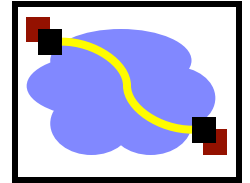


# Issues with HTTP caching



- Caching is nice but...
- Over 50% of all HTTP objects are uncacheable – why?
- Challenges:
  - Dynamic data → stock prices, scores, web cams
  - CGI scripts → results based on passed parameters
  - SSL → encrypted data is not cacheable
    - Most web clients don't handle mixed pages well → many generic objects transferred with SSL
  - Cookies → results may be based on passed data
  - Hit metering → owner wants to measure # of hits for revenue, etc.

# Summary



- DNS (last time)
  - Globally distributed, weak consistency
  - Manual delegation
  - Recursive/iterative lookups
  - Designated set of root servers (sensitive)
- Content Delivery Networks move data closer to user, maintain consistency, balance load
  - Consistent Caching maps keys AND buckets into the same space
  - Consistent caching can be fully distributed, useful in P2P systems using structured overlays