Distributed Systems

22. Content Delivery Networks (CDN)

Paul Krzyzanowski
Rutgers University
Spring 2020
iOS 13.4.1
Apple Inc.
982.2 MB

iOS 13 introduces a dramatic new look for iPhone with Dark Mode, new ways to browse and edit photos, and a private new way to sign in to apps and websites with just a tap. iOS 13 is faster and more responsive with optimizations across the system that improve app launch, reduce app download sizes, and make Face ID even faster.

Learn More

Download and Install
Motivation

• Serving content from one location presents problems
  – Scalability
  – Reliability
  – Performance

• “Flash crowd” problem
  – What if everyone comes to your site at once?

• What do we do?
  – Cache content and serve requests from multiple servers at the network edge (close to the user)
    • Reduce demand on site’s infrastructure
    • Provide faster service to users
      – Content comes from nearby servers
Focus on Content

• Computing is still done by the site host’s server(s)

• Offload the static parts – they often make up the bulk of the bytes:
  – Images
  – Video
  – CSS files
  – Static pages
Every request goes to the server.
Repeated requests from one client may be optimized by browser-based caching
… but that cached data is local to the browser
Caching proxy in an organization.
Take advantage of what others before you have recently accessed.
Load Balancing

Internet

Web server
Web server
Web server

Load Balancer

Browser
Browser
Browser
Browser

Increase capacity at the server.
Internet connectivity can be a bottleneck … + latency from client to server.
Internet End-to-End Packet Delivery

Network edges: applications & hosts

Network core: routers
Multihoming

- Get network links from multiple ISPs
- Server has one IP address but multiple links
- Announce address to upstream routers via BGP:

  Provides clients with a **choice of routes** and **fault tolerance** for a server’s ISP going down
Mirroring (Replication)

- Synchronize multiple servers
- Use multiple ISPs: location-based load balancing, ISP & server fault tolerance
Improving scalability, availability, & performance

• **Scalability**
  – Mirror (replicate) servers for load balancing among multiple servers
  – Multiple ISPs if network congestion is a concern

• **Availability**
  – Replicate servers
  – Multiple data centers & ISPs

• **Performance**
  – Cache content and serve requests from multiple servers at the network edge (close to the user)
    • Reduce demand on site’s infrastructure
    • Provide faster service to users
      – Content comes from nearby servers
But these approaches have problems!

- **Local balancing**
  - Data center or ISP can fail

- **Multihoming**
  - IP protocols (BGP) are often not quick to find new routes

- **Mirroring at multiple sites**
  - Synchronization can be difficult

- **Proxy servers**
  - Typically a client-side solution
  - Low cache hit rates

*All require extra capacity and extra capital costs*
Akamai Distributed Caching

- Company evolved from MIT research

- "Invent a better way to deliver Internet content"

- Tackle the "flash crowd" problem

Akamai runs on ~275,000 servers in ~1,500 networks across >130 countries

- Delivers 15-30% of all web traffic
  ... reaching over 30 Terabits per second

- Used by 19 of top 20 U.S. eCommerce retailers; 5 of top 5 travel websites; all 5 branches of the U.S. Department of Defense, all top 25 U.S. Banks and top 10 European banks, ...

http://www.akamai.com/html/about/facts_figures.html
Akamai’s goal

Try to serve clients from servers likely to have the content

– **Nearest**: lowest round-trip time
– **Available**: server that is not too loaded
– **Likely**: server that is likely to have the data
Akamai Overlay Network

The Internet is a collection of many autonomous networks

– Routing is based on business decisions
  • Peering agreements, not performance

– An ISP’s top performance incentives are:
  • Last-mile connectivity to end users
  • Connectivity to servers on the ISP

• Akamai's Overlay network
  – Collection of caching servers at many, many ISPs
  – All know about each other
Overlay Network: (1) DNS Lookup

1. Domain name lookup
   - Translated by **mapping system** to an edge server that can serve the content
   - Use custom **dynamic DNS servers**
     - Take requestor’s address into account to find the *nearest* edge
   - Resolve a host name based on:
     - User location
       (minimize network distance)
     - Server health
     - Server load
     - Network status
     - Load balancing
   - Try to find an edge server at the customer’s ISP
Overlay Network: (2) Transport System

2. Browser sends request to the given edge server
   - Edge server may be able to serve content from its cache
   - May need to contact the origin server via the transport system
Akamai collects network performance data

• **Map network topology**
  – Based on BGP and *traceroute* information
  – Estimate hops and transit time

• **Monitor load**
  – Content servers report their load to a monitoring application
  – Monitoring app publishes load reports to a local (Akamai) DNS server

• **Assign servers**
  – Dynamic DNS server determines which IP addresses to return when resolving names

• **Load shedding:**
  – If servers get too loaded, the DNS server will not respond with those addresses
CDN Structure: Pushing & Pulling

- **Push CDNs**
  - Origin must store content manually onto delivery nodes

- **Pull CDNs**
  - Delivery nodes request content from the origin
Benefits of a CDN

1. Caching
2. Routing
3. Security
4. Analytics
5. Cost
1. Caching

- **Goal**: Increase hit rate on edge servers
  - Reduce hits on origin servers

- **Two-level caching**
  - If edge servers don’t have the data, check with parent servers

- **Static content can be served from caches**
  - Dynamic content still goes back to the origin
1. Caching: types of content

- **Static content**
  - Cached depending on original site's requirements (never to forever)

- **Dynamic content**
  - Caching proxies cannot do this
  - Akamai uses *Edge Side Includes* technology (www.esi.org)
    - Assembles dynamic content on edge servers
    - Similar to server-side includes
    - Page is broken into fragments with independent caching properties
    - Assembled on demand

- **Streaming media**
  - Live stream is sent to an entry-point server in the CDN network
  - Stream is delivered *from the entry-point server to multiple edge servers*
  - Edge servers serve content to end users.
2. Routing

• Route to parent servers or origin via the overlay network

• Routing decision factors:
  – measured latency
  – packet loss
  – available bandwidth

• Results in ranked list of alternate paths from edge to origin

• Each intermediate node acts as a forwarder
  – Keep TCP connections active for efficiency
3. Security

• High capacity
  – Overwhelm DDoS attacks

• Expertise
  – Maintain systems and software

• Extra security software
  – Hardened network stack
  – Detect & defend attacks

• Shield the origin
  – Attacks hit the CDN, not the origin
4. Analytics

• Reports on quality of service, latency, media performance
• Engagement: # views, duration, abandoned plays
• Geography: zip code, continent, region, ISP
• Clients: devices, operating systems
• Most popular content
• Session: bandwidth, referrer URL, session duration
5. Cost

• Infrastructure on demand
  – CDN absorbs majority of content

• Instant worldwide scaling based on demand

• Business advantages
Video Streaming via CDNs
How is live video different?

• Live video cannot be cached
  – Progressive downloads – watch video while downloading
  – vs. direct downloads – download first, watch later

• HTTP Live Streaming (HLS): most popular way to access video
  – Use generic HTTP servers
  – Deliver on-demand video just like any other content

• Adaptive bitrate coding (ABR) – added at CDN
  – Break video stream to chunks (between 2-10 seconds)
  – CDN encodes chunks at various bitrates (quality & resolution)
  – Uses feedback from user’s playback client to pick optimal next chunk
  – Revise constantly
ABR Transcoding

Publish

CDN

Transcode

Content server
Content server
Content server

HTTP Live Streaming

\( f(\text{player, device, encoding parameters}) \)
Server-side Video Ad Insertion

- Pre-roll, post-roll, mid-roll, overlay, etc.
- Clickable ads, skippable ads
- Integrate with ad servers (DoubleClick, LiveRail, Tremor, YuMe, …)
- Supported by Google Dynamic Ad Insertion, Amazon AWS Server-Side Ad Insertion, Limelight Orchestrate™, Verizon Smartplay, …

Example: Limelight Reach Ads
Peer-to-Peer Content Delivery
Gnutella

• Background
  – Created by Justin Frankel and Tom Pepper (authors of Winamp)
  – AOL acquired their company, Nullsoft in 1999
  – In 2000, accidentally released gnutella
  – AOL shut down the project but the code was released

• Big idea: create fully distributed file sharing
  – Unlike Napster, you cannot shut down gnutella
Gnutella: Overview

Gnutella is based on **query flooding**

- **Join**
  - On startup, a node (peer) contacts at least one node
    - Asks who its friends are
    - These become its “connected nodes”

- **Publish**
  - No need to publish

- **Search**
  - Ask connected nodes. If they don’t know, they will ask their connected nodes, and so on…
  - Once/if the reply is found, it is returned to the sender

- **Fetch**
  - The reply identifies the peer; connect to the peer via HTTP & download
Gnutella: Search

Initial query sent to neighbors ("connected nodes" in an overlay network)

Query: where is file X?

Query: where is file X?
Gnutella: Search

If a node does not have the answer, it forwards the query.

Queries have a hop count (time to live) – so we avoid forwarding loops.
Gnutella: Search

If a node has the answer, it replies – replies get forwarded

Query: where is file X?
Reply
Query: where is file X?
Reply
Query: where is file X?
Reply
Query: where is file X?
Reply
Query: where is file X?
Reply
Query: where is file X?
I have X!
Query: where is file X?
Gnutella: Search

Original protocol

- **Anonymous**: you didn’t know if the request you’re getting is from the originator or the forwarder
- Replies went through the same query path

Downloads

- A node connects to the server identified in the reply
- If a connection is not possible due to firewalls, the requesting node can send a *push request* for the remote client to send it the file
Gnutella: Summary

• Pros
  – Fully decentralized design
  – Searching is distributed
  – No control node – cannot be shut down
  – Open protocol

• Cons
  – Flooding is inefficient:
    • Searching may require contacting a lot of systems; limit hop count
  – Well-known nodes can become highly congested
  – If nodes leave the service, the system is crippled
Kazaa

• Background
  – Kazaa & FastTrack protocol created in 2001
  – Team of Estonian programmers – same team that will later create Skype
  – Post-Napster and a year after Gnutella was released
  – FastTrack: used by others (Grokster, iMesh, Morpheus)
    • Proprietary protocol; Several incompatible versions

• Big idea: Some nodes are better than others
  – A subset of client nodes have fast connectivity, lots of storage, and fast processors
  – These will be used as supernodes (similar to gnutella’s ultrapeers)
  – Supernodes:
    • Serve as indexing servers for slower clients
    • Know other supernodes
Kazaa: Supernodes

Supernodes

Ordinary nodes

Ordinary nodes

April 13, 2020 © 2017-2020 Paul Krzyzanowski
Kazaa: publish a file

I have X
Kazaa: search

Supernodes answer for all their peers (ordinary nodes)
Kazaa: Discussion

Selective flooding of queries

• Join
  – A peer contacts a supernode

• Publish
  – Peer sends a list of files to a supernode

• Search
  – Send a query to the supernode
  – Supernodes flood the query to other supernodes

• Fetch
  – Download the file from the peer with the content
Kazaa: Summary

• **Pros**
  – Efficient searching via supernodes
  – Flooding restricted to supernodes

• **Cons**
  – Can still miss files
  – Well-known supernodes provide opportunity to stop service

• **Gnutella also optimized its architecture**
  – Added ultranodes = supernodes
BitTorrent

• Background
  – Introduced in 2002 by Bram Cohen
  – Motivation
    • Popular content exhibits temporal locality: *flash crowds*
      – E.g., slashdot effect, CNN on 9/11, new movies, new OS releases

• Big idea: allow others to download from you while you are downloading
  – Efficient fetching, not searching
  – Single publisher, many downloaders
BitTorrent: Publishing & Fetching

To distribute a file

– Create a `.torrent` file

– Contains
  • Name
  • Size
  • Hash of each piece
  • Address of a tracker server

– Start a seed node: initial copy of the full file

– Start the `tracker` for the file
  • Tracker manages uploading & downloading of the content
To get a file

- Get a .torrent file
- Contact the *tracker* named in the file
  - Get the list of seeders and other nodes with portions of the file
  - Tracker will also announce you to others
- Contact a random node for a list of file piece numbers
- Request a random block of the file
BitTorrent: Downloading a file in chunks

Tracker identifies:
(1) initial system(s) that has 100% of the file (the seed)
(2) which machines have some pieces of the file downloaded

Tracker

Seed node (you can have multiple seeds)

Peer

Leecher: a peer that is downloading a file (and offering uploads)

Peer

Seeder: a peer that has the entire copy of the file

Peer

Swarm: set of peers involved in upload/download for a file

Complete file

Request piece

Request piece

When a peer finished downloading a file, it may become a seed and remain online without downloading any content.
Enable downloads from peers

• **Join**
  – No need to join
    (seed registers with tracker server; peers register when they download)

• **Publish**
  – Create a torrent file; give it to a *tracker server*

• **Search**
  – Outside the BitTorrent protocol
  – Find the tracker for the file you want, contact it to get a list of peers with files

• **Fetch**
  – Download pieces of the file from other peers
  – At the same time, other peers may request pieces from you
BitTorrent Summary

• Pros
  – Scales well; performs well when many participants
  – Gives peers an incentive to share
    • It is sometimes not possible to download without offering to upload

• Cons
  – Search is not a part of the protocol; relies on torrent index servers
  – Files need to be relatively large for it to work well
  – Rare files do not offer distribution
  – A tracker needs to be running to bootstrap the downloads
The end