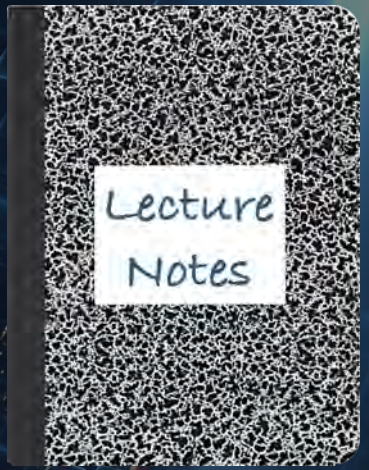
CS 417 – DISTRIBUTED SYSTEMS



Week 11: Data In Motion

Content Delivery Networks (CDNs) & Edge Computing

Paul Krzyzanowski

How do you update ~1B phones?

The **Flash Crowd** Problem

Serving content from one location presents problems

- Scalability
- Reliability
- Performance

At scale: a popular event triggers millions of simultaneous requests

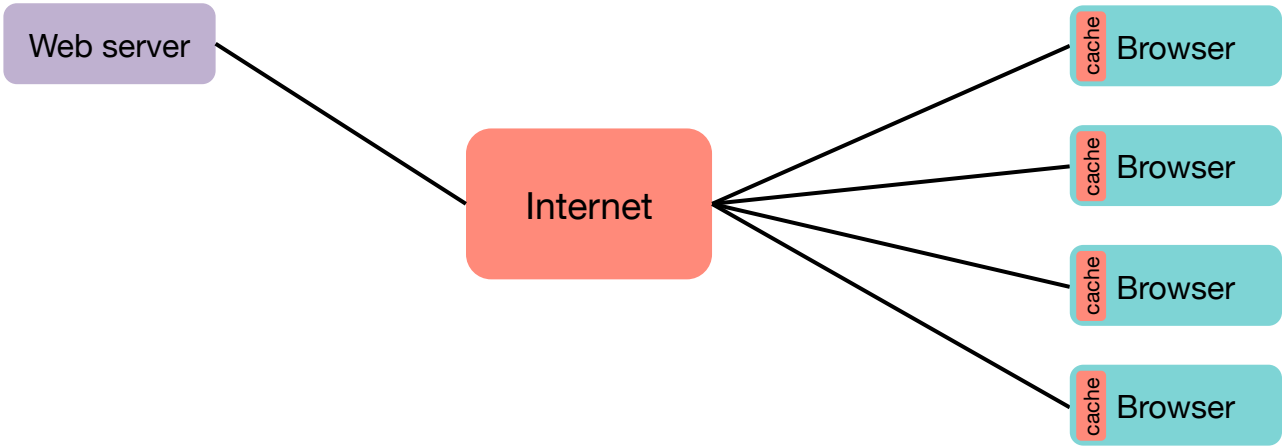
- Server gets overwhelmed, connections time out, site goes offline
- The bandwidth alone would saturate any single building's network link

Solution: distribute content across many servers close to users

Pre-CDN Approaches

How can we make content access more efficient?

Serving & Consuming Content: Browser Caches

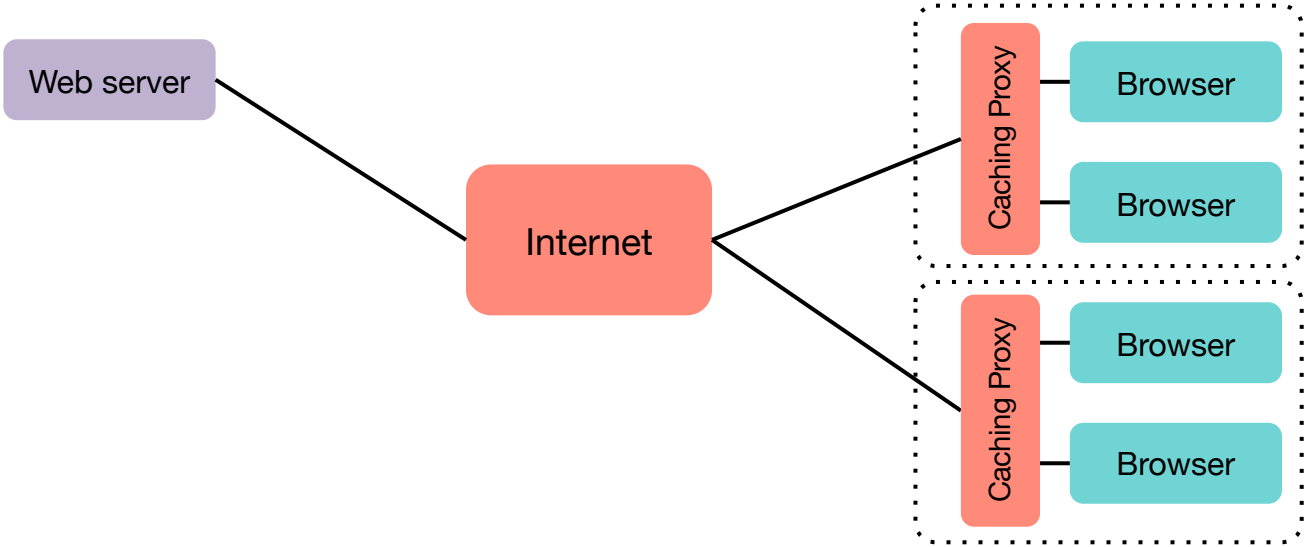


Every request goes to the server.

Repeated requests from one client may be optimized by **browser-based caching**

… but that cached data is local to the browser

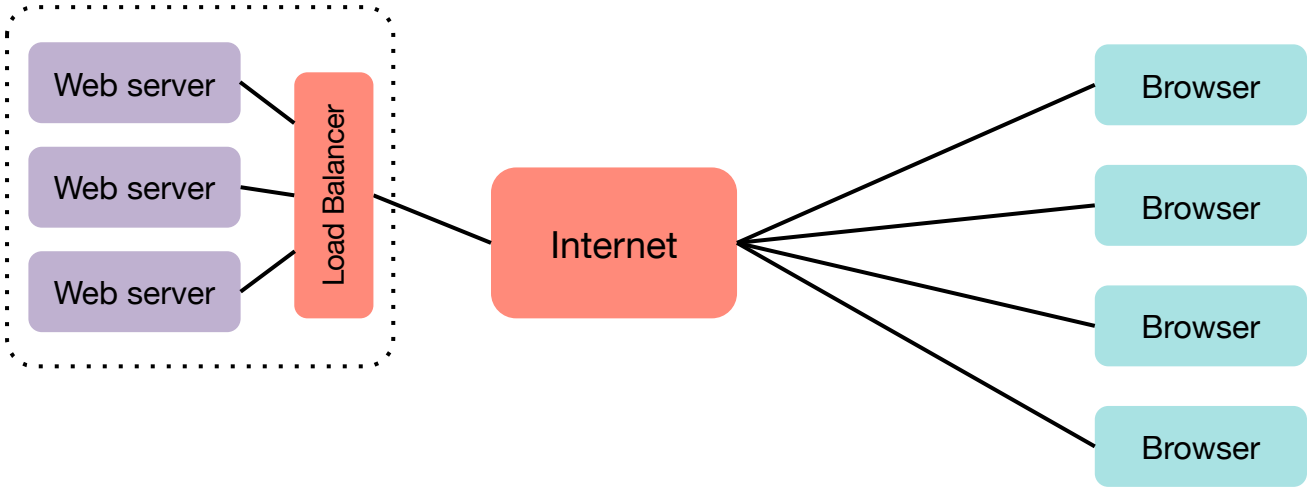
Caching Proxies



Caching proxy in an organization or ISP.

Take advantage of what others before you have recently accessed.

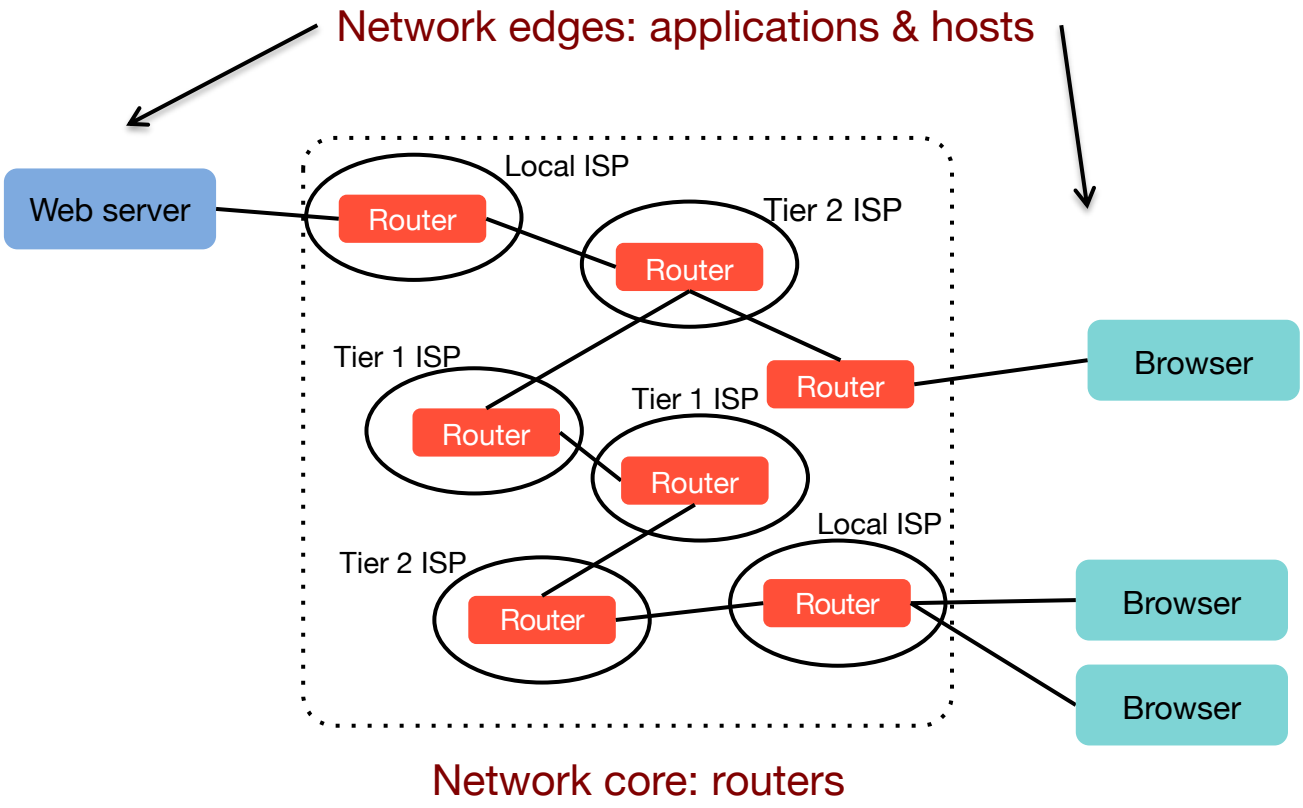
Load Balancing



Increase capacity at the server – multiple servers may serve content

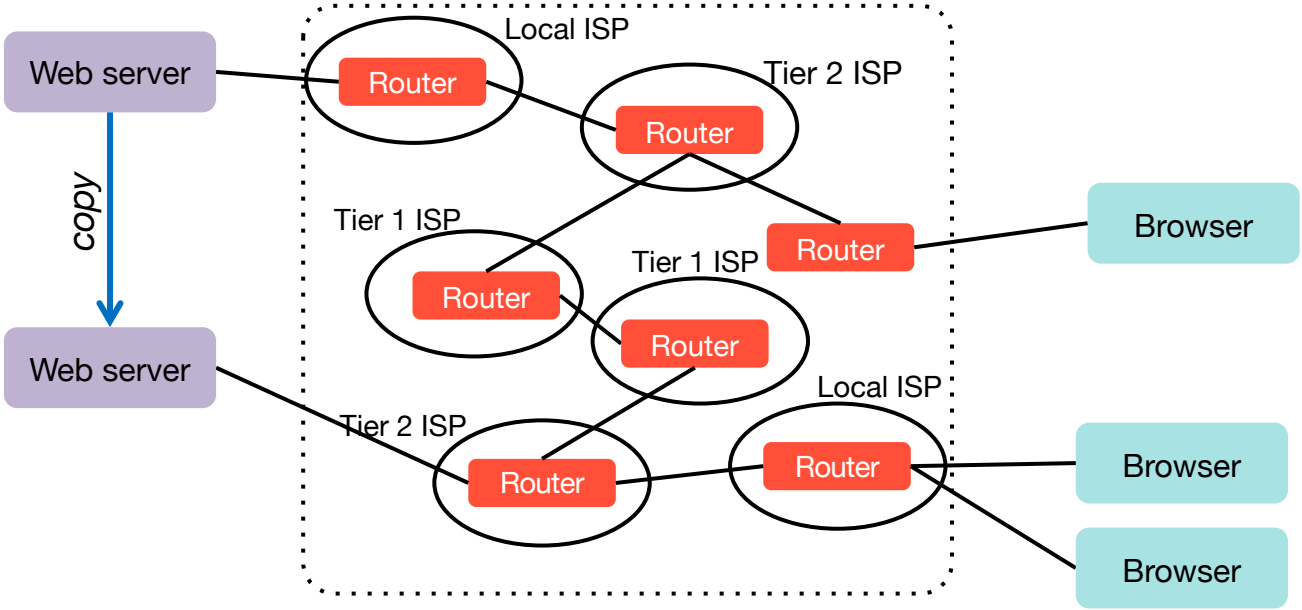
Internet connectivity can be a bottleneck … + latency from client to server.

Internet End-to-End Packet Delivery



Mirroring (Replication)

- Replicate multiple servers across ISP links
- Use multiple ISPs: location-based load balancing, ISP & server fault tolerance



Content Delivery Networks

How 72% of all global internet traffic is served

What is a Content Delivery Network?

Distributes **cached copies of content** across hundreds or thousands of servers worldwide

User requests a file
→ CDN serves it from a nearby **edge server**, not the origin

What gets offloaded (the bulk of the bytes):

- Images, video, CSS, JavaScript libraries, downloadable files
- All **static** content that doesn't change per-user

What stays on the origin:

- Application logic, database queries, user-specific responses
- Small fraction of total traffic

Content Delivery as a Service

Some of the largest Internet companies run their own CDNs
(Google, Microsoft, Amazon, Meta, eBay, Apple, …)

- Redundant, globally-distributed data centers connected to many ISPs
- Example: Google search
  - Mirroring across data centers
  - Load balancing within a data center

- **For most companies, it doesn't make sense**
  - Huge capital expense
  - Huge operating costs
  - Capacity is not always needed, so most networks & servers will be underutilized

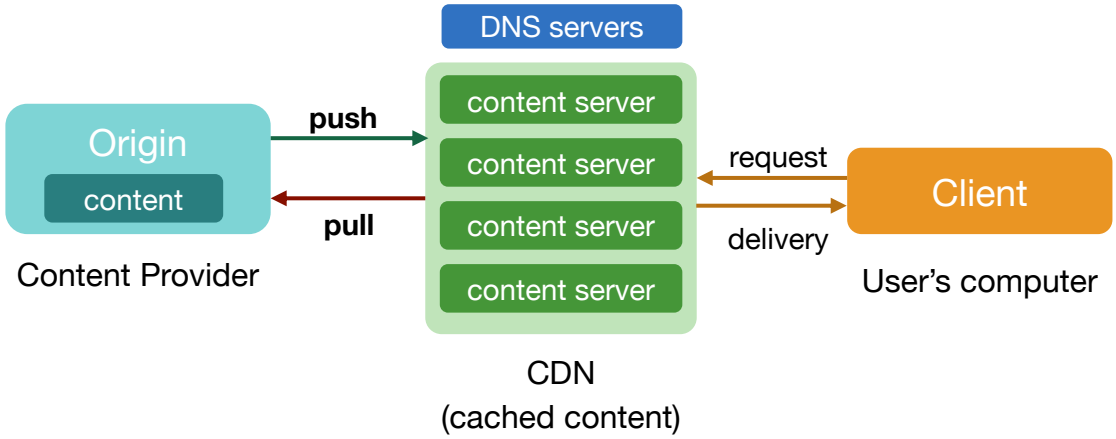
CDNs are a service

- *Let someone else figure out, pay for, and run scalable content delivery*

CDN Providers

- **Akamai (1998)**: founded by MIT researchers studying the flash crowd problem
  - Built the first large-scale commercial CDN; dominant for most of the 2000s
  - DNS-based routing; servers deployed inside ISPs worldwide
- **Cloudflare**: CDN + security focus; uses anycast routing globally
- **Amazon CloudFront**: integrated with AWS; default for AWS workloads
- **Fastly**: developer-focused; low-latency edge logic and fine control
- **Google**, **Microsoft**: also operate CDN products for customers
- *And others… (Microsoft Azure CDN, Google Cloud CDN, Gcore, Bunny.net, …*

CDN Structure: Pushing & Pulling



Push CDNs

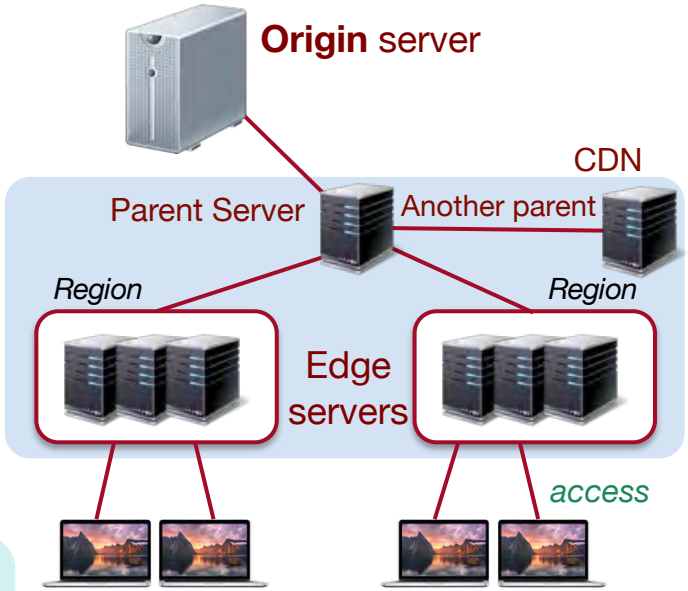
- Origin must store content manually onto delivery nodes

Pull CDNs

- Delivery nodes request content from the origin

CDN Architecture: Tiers

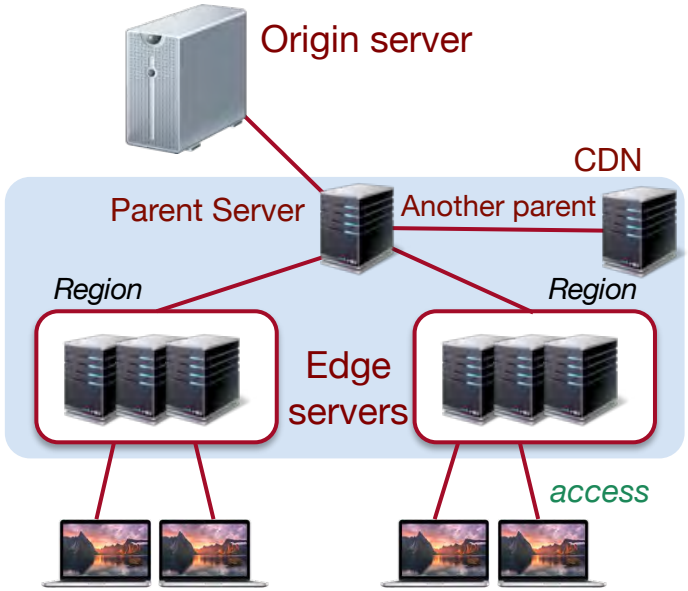
- **Edge servers**
  Closest to users, often inside ISPs or at IXPs
- **Parent servers**
  Sit above edge tier, cache what edge servers may not carry
- **Origin**
  The content provider's actual infrastructure



IXP: Internet Exchange Point
Locations where multiple networks interconnect
E.g., DE-CIX in Frankfurt or AMS-IX in Amsterdam

CDN Cache Resolution Sequence

1. Edge checks its local cache
2. Edge queries other edge servers in the region
3. Edge asks its parent server
4. Parent checks peer parents in other regions
5. Last resort: parent fetches from origin



DNS-Based Request Routing

CDN Content Serving Goal

Try to serve content to clients from caching servers that are:

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

DNS Redirection: The Mechanism

- Content provider sets up a **CNAME** record pointing to the CDN domain
  - TTL on CDN DNS responses is deliberately short (often 30 seconds)
    - Allows rapid traffic shifting in response to failure or overload
- User's browser resolves the domain
  - Hits CDN's **dynamic DNS servers**
  - CDN's DNS server returns the IP of a nearby, healthy edge server
    - Selection factors: user location, server load, network conditions
- A user in London and a user in LA resolve the same hostname differently
- CDN continuously monitors servers
  - Failed/overloaded nodes are removed from DNS responses

Content lookup: DNS Example

The company's original content is hosted on its server = origin server

Edge servers in the CDN cache content and take place of the actual servers that host the site's content

Example

`www.staples.com` is an alias (a CNAME in DNS) that maps to `www.staples.com.edgekey.net` and `edgekey.net` is an Akamai domain:

```
www.staples.com.              20247 IN CNAME   www.staples.com.edgekey.net.
www.staples.com.edgekey.net.  21600 IN CNAME   e6155.a.akamaiedge.net.
e6155.a.akamaiedge.net.       20    IN A       23.201.180.183
```

Users think they're connecting to `www.staples.com`, but are really connecting to an Akamai caching server.

Akamai's Mapping System

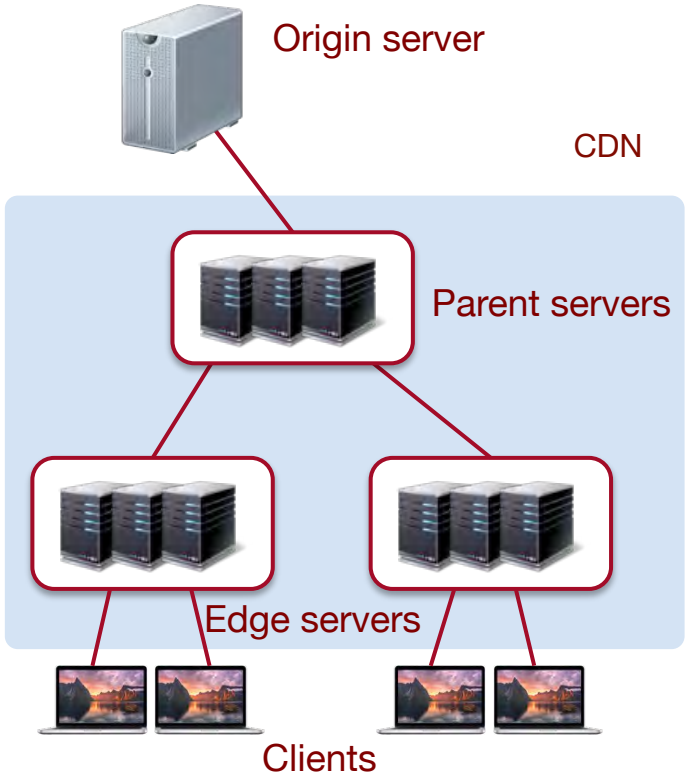
- Edge server selection goes beyond simple geographic proximity
- Factors considered:
  - User location: derived from IP address of the DNS resolver
  - Network topology: BGP tables and traceroute measurements
  - Server load: edge servers report current load to monitoring infrastructure
  - Server health: failed health checks exclude a server from DNS responses
  - Network performance: measured latency and packet loss between regions
- Goal: return IP for an edge server that is close, available, and cached

Benefits of a CDN

1. Caching
2. Routing
3. Security

1. Caching

- Goal: Increase hit rate on edge servers
  - Reduce hits on origin servers
  - Serve static content
  - Dynamic content won't be cached
- Two-level caching
  - If edge servers don’t have the data, check with parent servers
  - A parent server will check other parents before sending a request to the origin



1. Caching: Caching Controls

The organization can tell a CDN how to cache specific content:

- Use HTTP headers
- Ignore HTTP headers and use a custom time-to-live
- Never cache

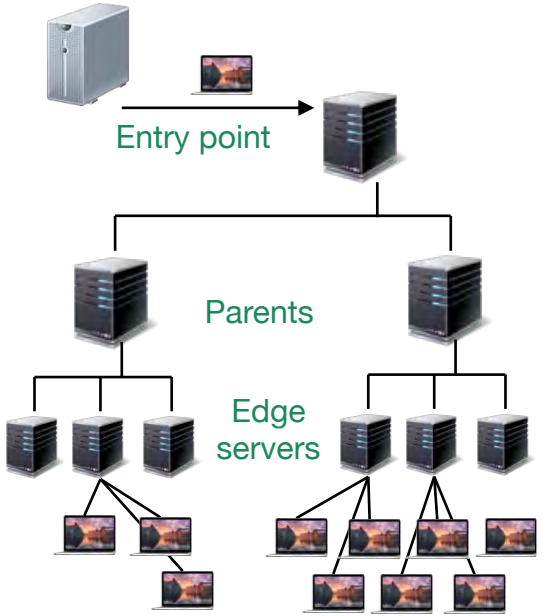
An HTTP `Cache-control` header can specify:

- `max-age`: specify how long a file can stay in the cache (seconds)
- `no-store`: don't cache – content that changes and shouldn't be cached
- `no-cache`: revalidate each request (e.g., check with origin via `if-modified-since`)
- `public`: content can be cached publicly for all requests
- `private`: only the user's browser is allowed to cache the content

- `CDN-Cache-Control` header for CDN-specific behavior

1. Caching: Static vs. Dynamic vs. Streaming Content

- **Static content**
  - Cached depending on the original site's requirements (never to forever)
- **Dynamic content**
  - Akamai uses *Edge Side Includes* (ESI)
    - Break page into fragments with independent caching rules
    - Assembled at the edge on demand
    - *Example: news site*
      - *masthead (1 week), headlines (1 hour), personalized sidebar (never)*
- **Streaming media**
  - Live stream pushed to CDN entry-point server
  - Distributed to edge servers and then to end users



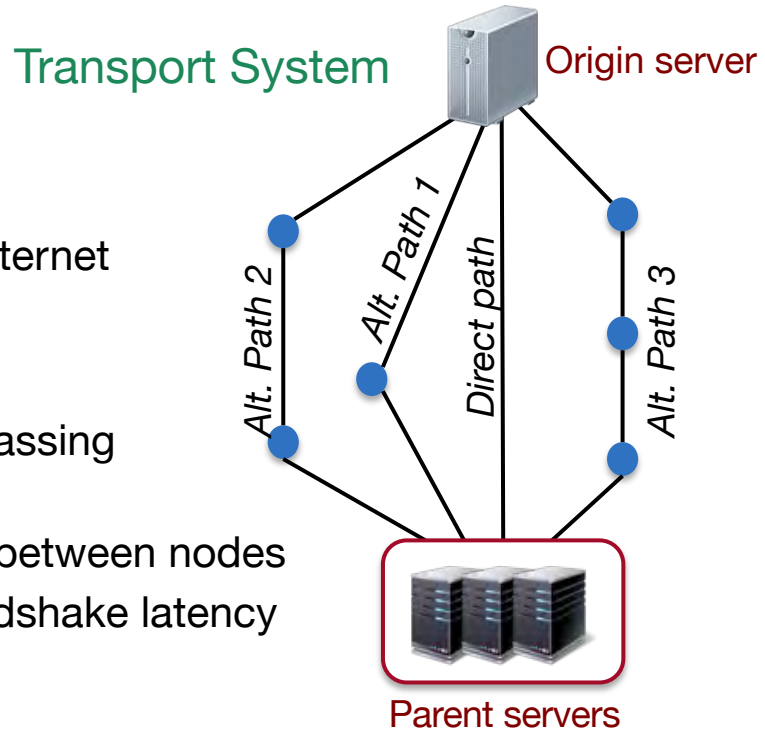
2. Routing: Collecting Network Performance Data

- **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

2. Routing: Overlay Network & Transport System

The Internet is a collection of many autonomous networks

- Routing is based on business decisions
  - Peering agreements, not performance



CDN Overlay Network

- Application-level network built on top of the Internet
- All nodes know about each other
- Periodically measure latency & packet loss
- Select the best path through CDN nodes, bypassing suboptimal public routes
- **Persistent TCP connections** are maintained between nodes
  - Avoids reconnection overhead and TLS handshake latency

2. Anycast Routing: The Alternative to Dynamic DNS

DNS-based routing limitation

- Routing decision made at DNS resolution time (before connection)
- Conditions may change after DNS TTL caches the response

Anycast (used by **Cloudflare**)

- Many servers worldwide share the **same IP address**
- BGP routing sends packets to the **nearest server** advertising that address
- No need for dynamic DNS — browser connects to CDN IP directly
- Failover is automatic: BGP reroutes in seconds if server goes down
- Caveat: long-lived connections (WebSockets) may break on BGP update

Most large CDNs **combine both**: anycast to nearest PoP, DNS for finer-grained selection

3. Security

- **DDoS absorption**
  - DDoS = flood a server with so much traffic it can't respond to anyone
  - Your origin: 10 Gbps link. Attacker: 500 Gbps. CDN: hundreds of thousands of servers absorb it
- **Origin IP hidden**
  - Users only see CDN addresses; attackers can't reach your servers directly
- **Web Application Firewall (WAF)**
  - Inspects HTTP requests at the edge and blocks malicious ones (SQL injection, bots) before they reach the origin
- **TLS termination at edge**
  - TLS = the encryption behind HTTPS
  - CDN handles the TLS handshake at a nearby edge server
  - Faster for users (shorter round trip) and offloads crypto work from origin

Video Streaming at Scale

The Video Streaming Problem

- Netflix, Amazon Prime: hundreds of millions of subscribers worldwide
- A viewer watching a 2-hour film makes thousands of HTTP requests
  - Each request: a short video segment (2-10 seconds of video)
  - Each segment: several MB at typical bitrates
- Multiply by hundreds of millions of simultaneous viewers
  - This is a continuous, sustained load 24/7
- Latency matters
  - A segment that arrives late causes buffering & a delay to the user
  - CDNs must deliver segments fast enough to keep the player's buffer full

Peak Events: The Flash Crowd at Scale

- **2022 FIFA World Cup final**: ~1.5 billion viewers across broadcast + streaming
- **2024 Paris Olympics**: 23.5 billion minutes streamed on Peacock
  - Up to 60 simultaneous live event streams; 300 live events in one day
- **2024 Jake Paul vs. Mike Tyson fight on Netflix**: ~65M concurrent viewers
  - This is the record for peak concurrent viewers on a single platform
- **2026 World Cup** (US/Canada/Mexico): projected 1.6B+ for final

Live events are fundamentally harder than video on demand:

- Segments arrive just ahead of demand; pre-caching is impossible

HTTP Live Streaming (HLS) and MPEG-DASH

Modern streaming uses segment-based delivery over plain HTTP

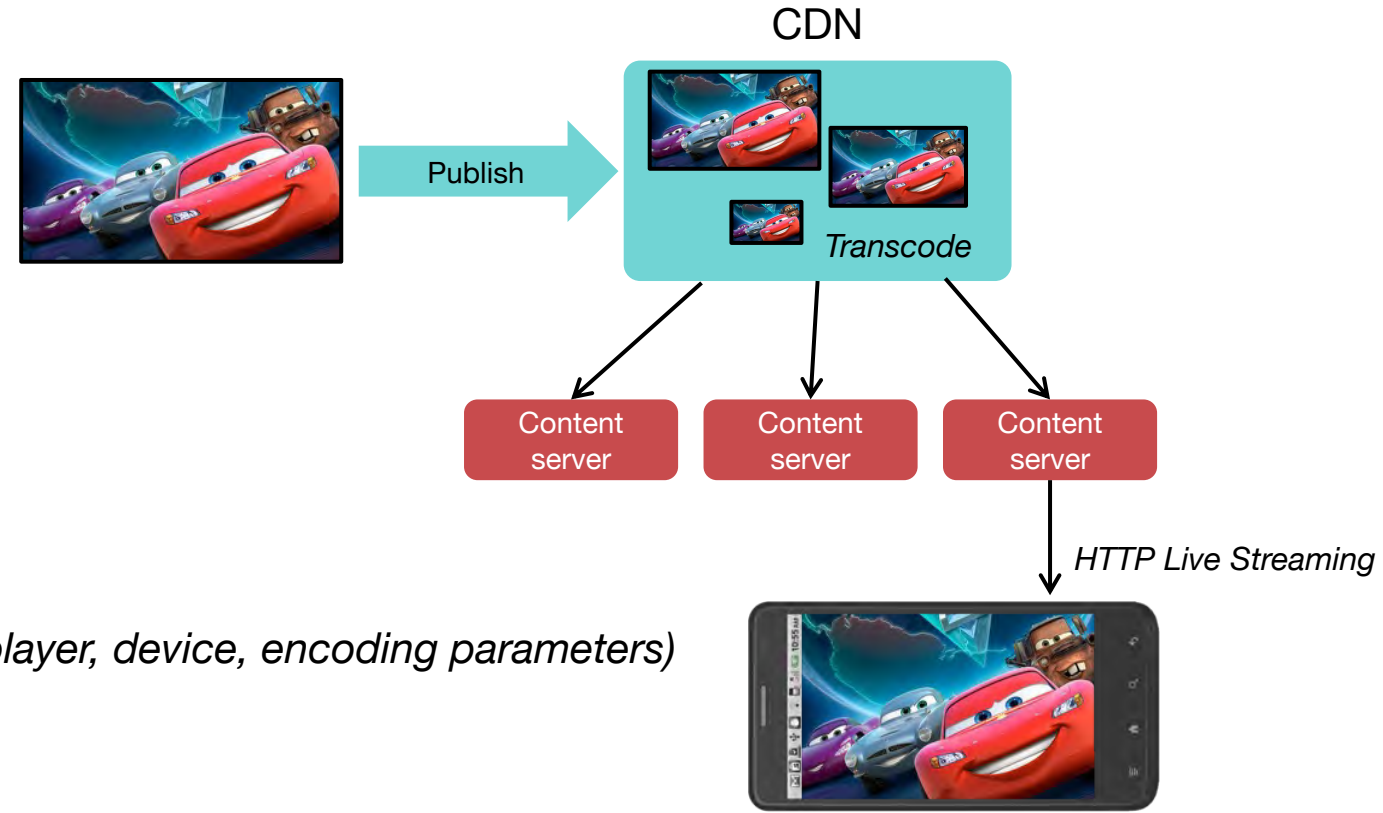
- **HTTP Live Streaming (HLS)**: developed by Apple; required for iOS/Safari
- **MPEG-DASH**: open international standard; codec-agnostic
  - Used by Netflix, YouTube, Amazon Prime, Disney+ on non-Apple devices

- From a CDN perspective, both are architecturally identical:
  - Video broken into short segments (2-10 seconds each)
  - Stored as regular HTTP files
    - Any CDN edge can cache and deliver them
  - A **manifest file** lists available segments and bitrates
  - Player requests segments sequentially
    - CDN serves content from cache

Adaptive Bitrate (ABR) Streaming

- Video encoded at multiple bitrates and resolutions (ABR transcoding)
  - Example: 4K/25 Mbps, 1080p/8 Mbps, 720p/4Mbps, 480p/1.5 Mbps
- Player monitors download speed and buffer level continuously
  - Player requests each next segment at the bitrate it can currently sustain
  - **Network degrades**: player steps down to lower quality automatically
  - **Network improves**: player steps back up
  - **Result**: video degrades gracefully on a slow connection, not stalling
- From CDN perspective: each bitrate variant is a separate set of files
  - Popular variants (1080p, 4K) get cached; rare variants may not

Adaptive Bitrate (ABR) Streaming



Live Video Delivery

- Live stream arrives at a CDN entry point (ingest server)
  - Entry point distributes to parent servers in multiple regions
  - Parent servers distribute to edge servers
  - Edge servers serve video segments to viewers
- This is analogous to IP multicast: source → tree → leaves
  - Uses HTTP segment delivery over CDNs, not IP multicast
- Live segments cannot be pre-cached; they arrive just ahead of demand

Edge Computing

Edge Computing

CDNs distribute **content**. Edge computing distributes **computation**

- Motivation: edge node ~10ms from user vs data center ~100ms
- Run logic on the CDN node already nearby

Cloudflare Workers — popular example

- JavaScript inside **V8 isolates** — lightweight sandboxes, start in microseconds
- Much faster than containers or VMs
- Use cases: auth checks, routing, A/B testing, image transforms

Architectural limit: state

- Round-trip to central DB erases latency benefit
- Edge-local storage (KV store, Durable Objects) helps for simple state
- Complex transactional logic still belongs at the origin

The End