

Web Caching



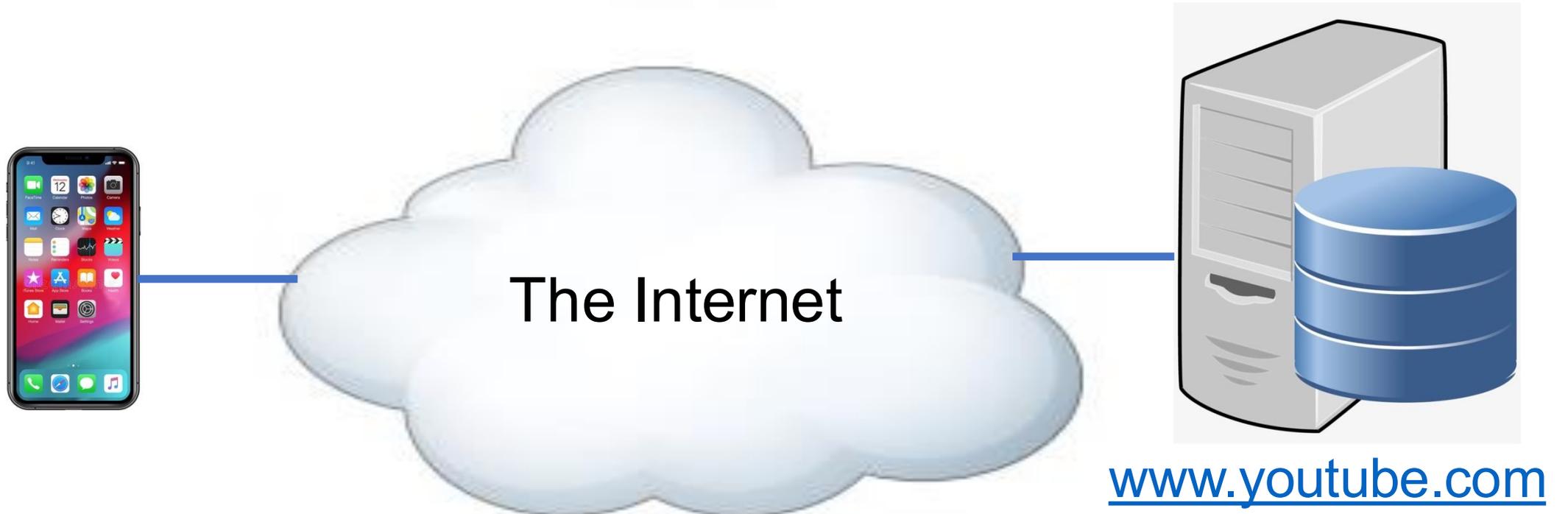
COS 316: Principles of Computer System Design

Lecture 10

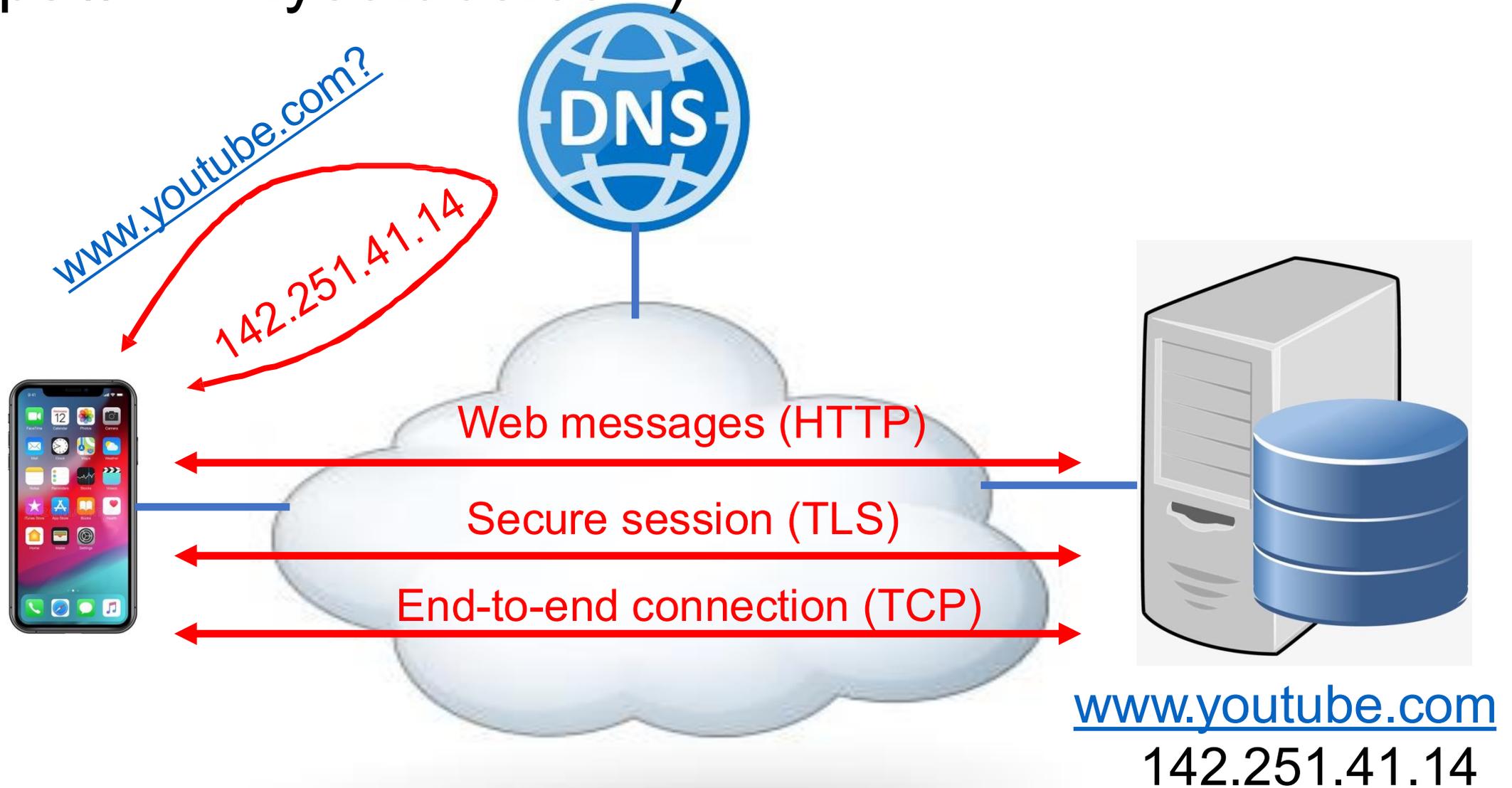
Wyatt Lloyd

Downloading a Web Page

User visits <https://www.youtube.com>



Downloading a Web Page (<https://www.youtube.com>)



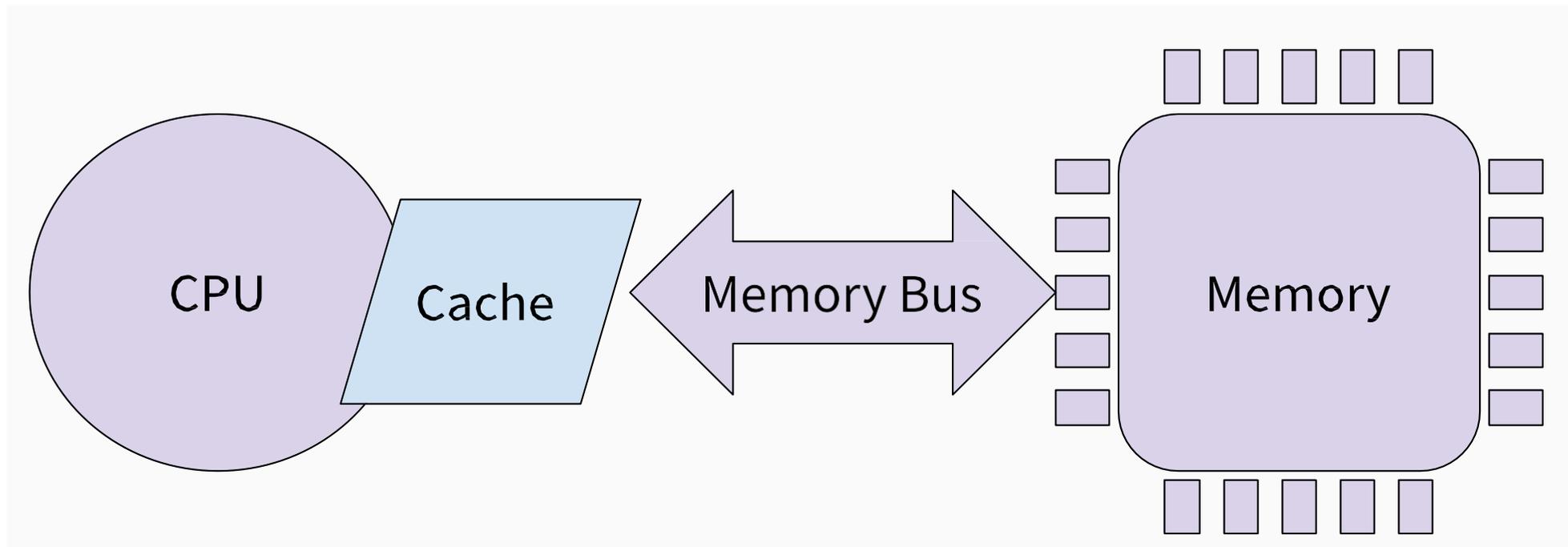
Multiple Problems

- User latency
 - Round-trips to query multiple DNS servers
 - Multiple round-trips with the Web server
 - Delivery of a (possibly large) Web item
- Server overhead
 - Handling many requests from many clients
 - Financial costs to deploy enough servers
- Network bandwidth
 - Traffic on many links in multiple networks
 - Financial costs for the affected networks



A Solution: Caching

- Keep all data in bigger, cheaper, slower storage
- Keep copies of active data in smaller, more expensive, faster storage



What do we cache?

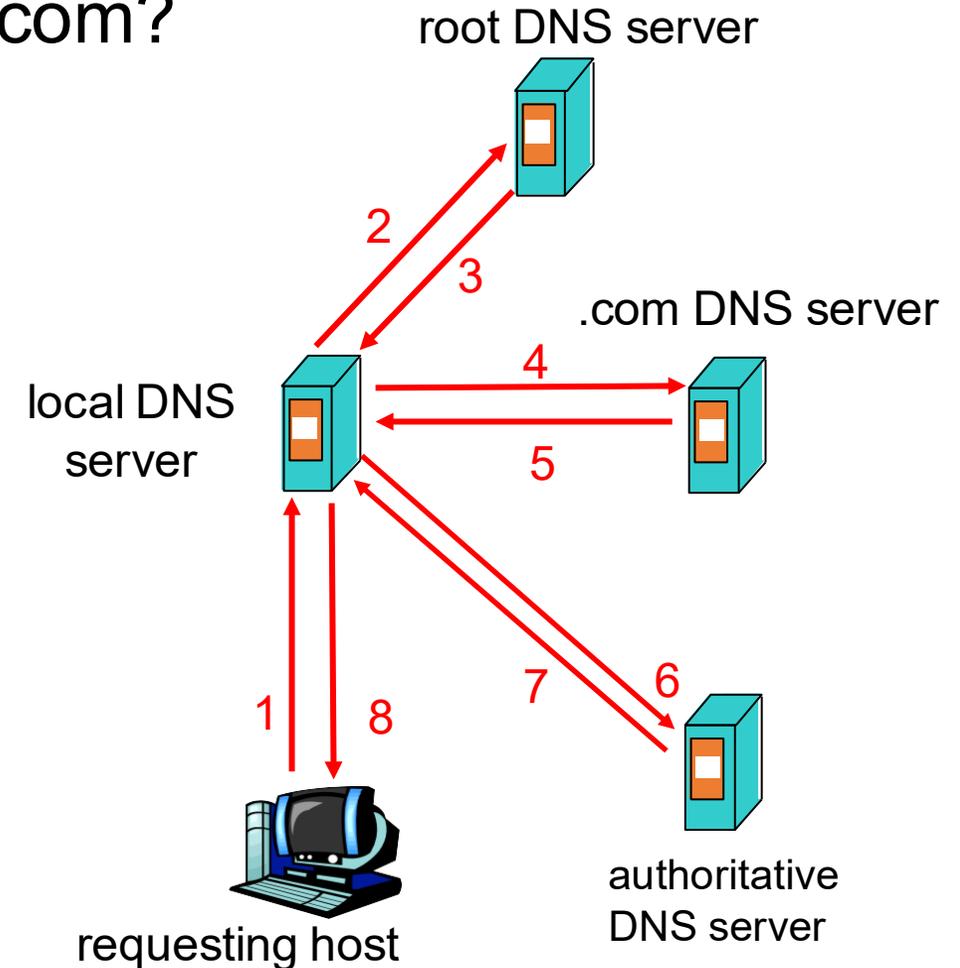
- Data stored verbatim in slower storage
- Previous computations – recomputations are a kind of `slow storage`
- Examples
 - CPU memory hierarchy
 - File system page buffer
 - Domain Name System (DNS)
 - Content Distribution Networks (CDN)
 - Web browser caches
 - Database caches

Caching to the Rescue: Domain Name System

- What to cache?
 - Mapping of popular names to IP addresses
 - E.g., www.youtube.com → 142.251.41.14
 - Mapping of parts of names to DNS server IPs
 - E.g., .com top-level domain → 192.26.92.30

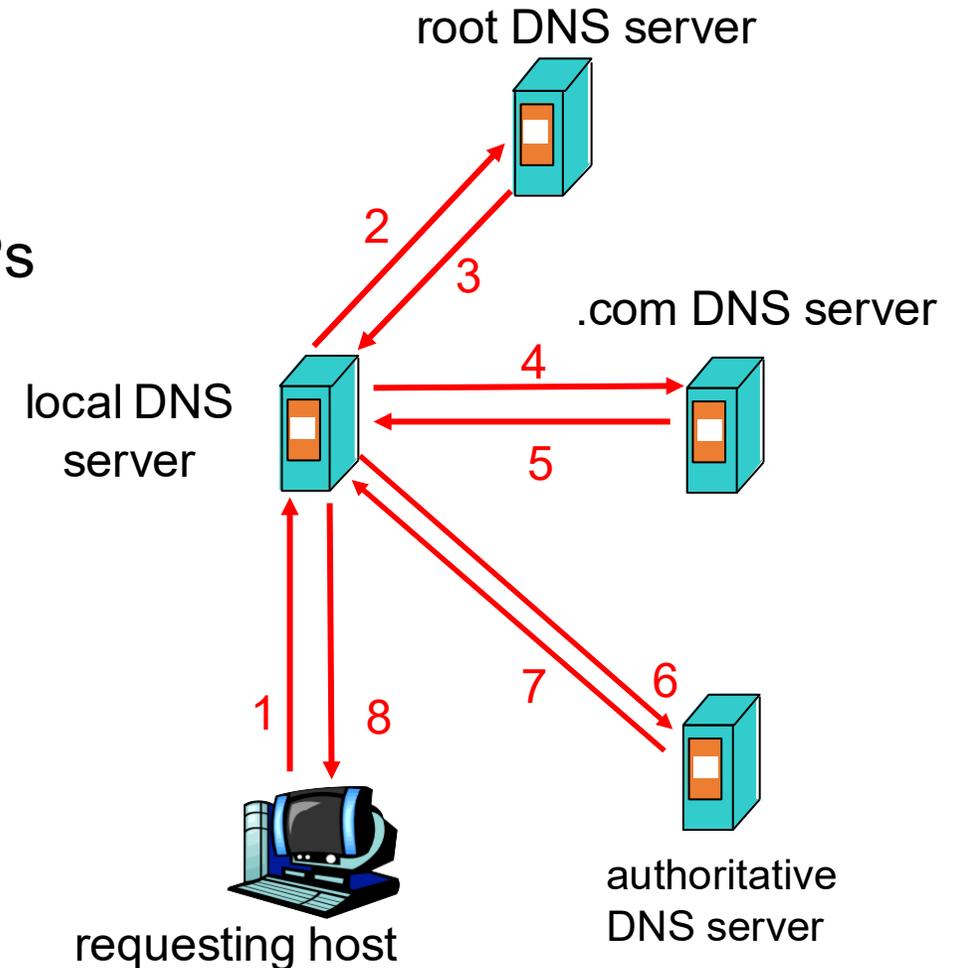
Domain Name System (DNS) 101

- What is the IP address for `www.youtube.com`?



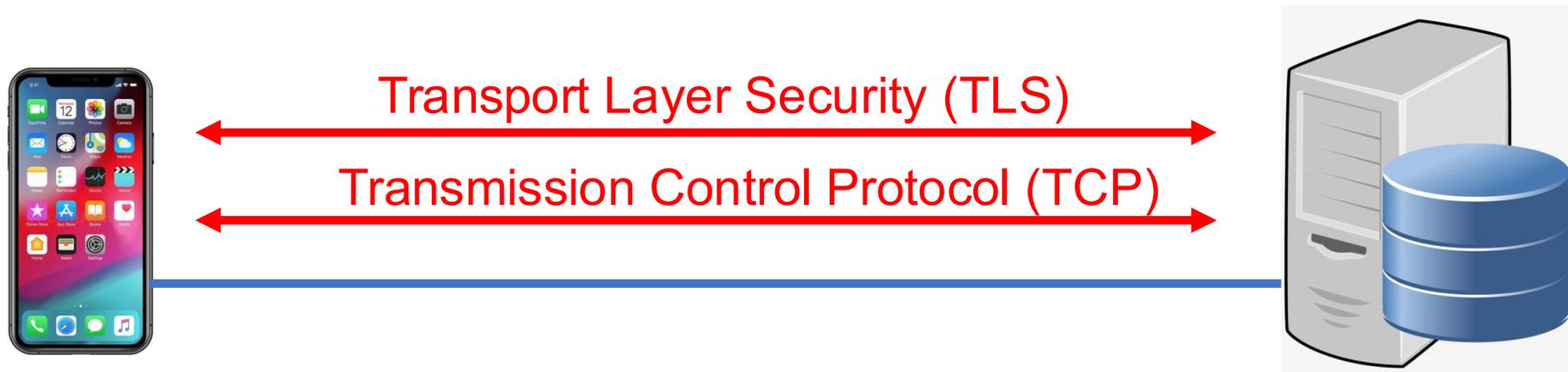
Caching to the Rescue: Domain Name System

- What to cache?
 - Mapping of popular names to IP addresses
 - E.g., www.youtube.com → 142.251.41.14
 - Mapping of parts of names to DNS server IPs
 - E.g., .com top-level domain → 192.26.92.30
- Where to cache?
 - Local DNS server (e.g., for the campus)
 - Client machine (e.g., user's browser)
- How to avoid stale information?
 - Cached entries have a limited “time to live”



Caching to the Rescue: Communication Channel

- End-to-end communication
 - TLS: confidentiality, integrity, and authenticity
 - TCP: ordered, reliable delivery of byte stream
- Establishing the channel is expensive
 - Communication delays, creating data structures, and computing keys
- Exploit **temporal locality** by reusing the channels

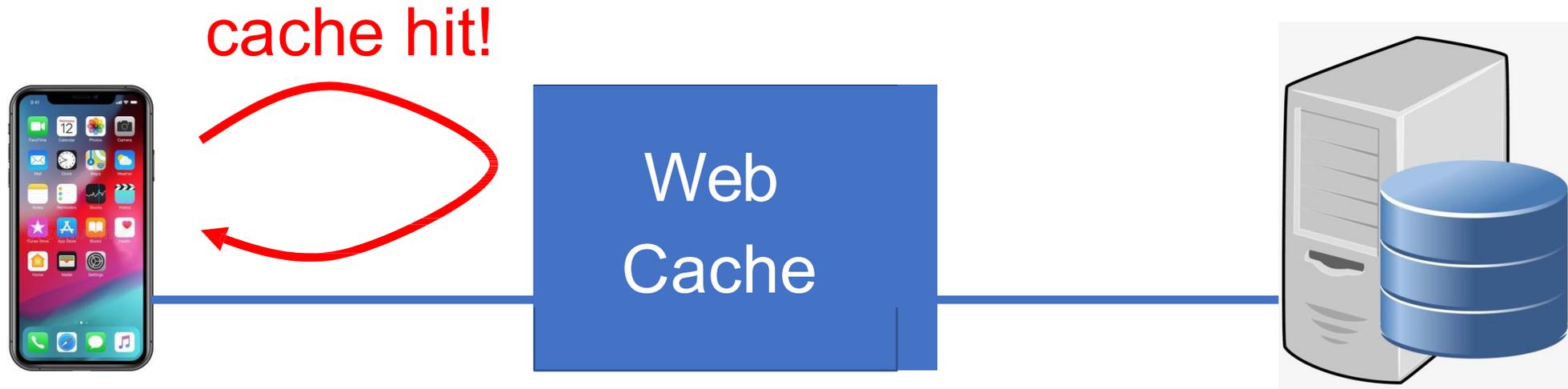


Temporal Locality

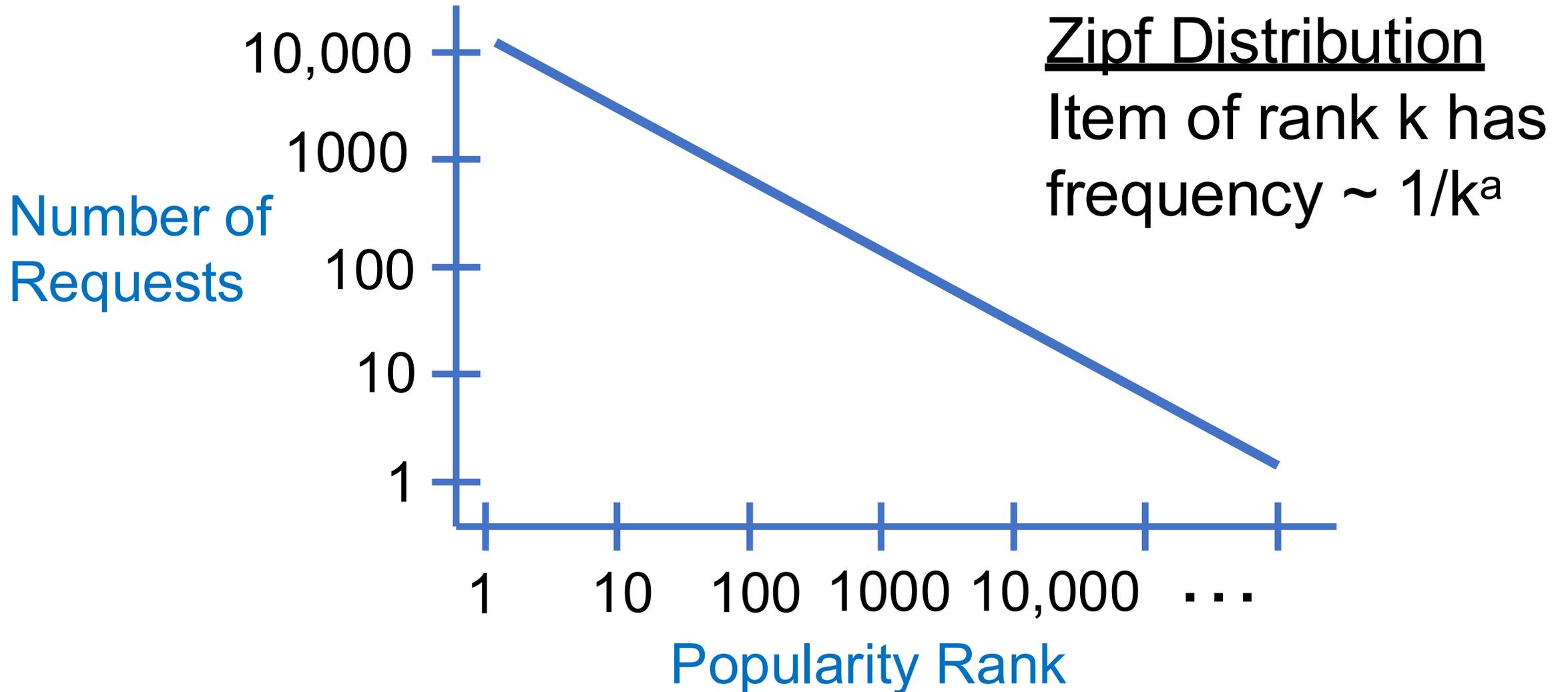
- Temporal locality: nearness in time
- Data accessed now was probably accessed recently
- Useful data tends to continue to be useful

Caching to the Rescue: Web Items

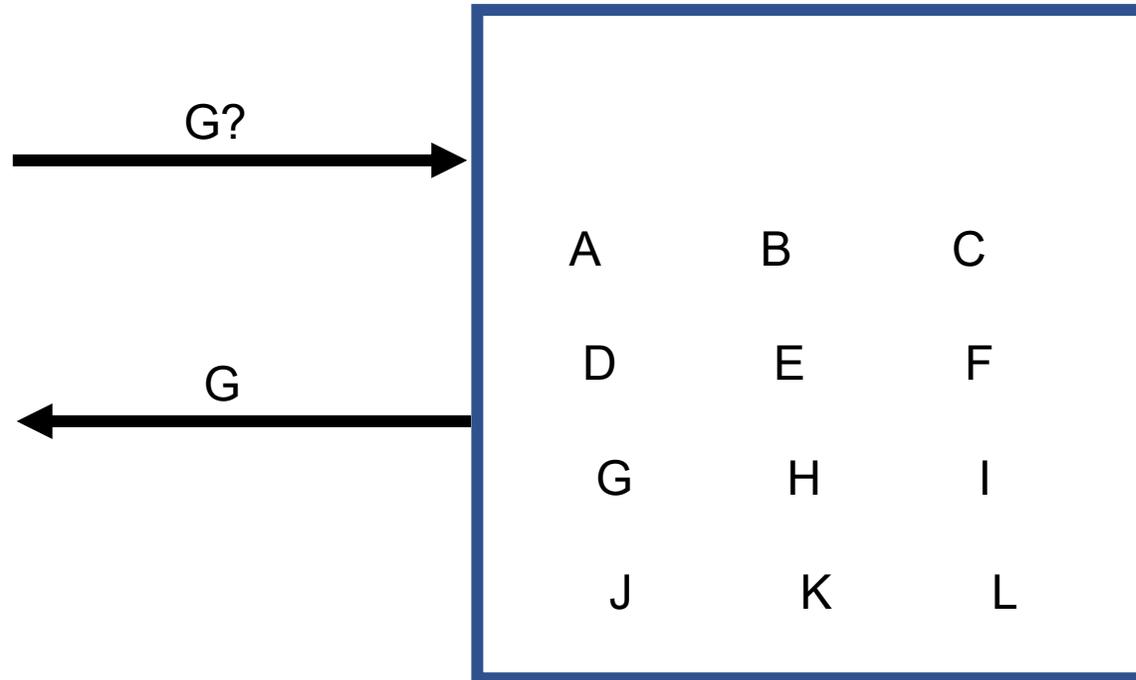
- Cache Web items closer to the client
 - Reduce latency
 - Reduce server overhead
 - Reduce use of network bandwidth



Web Caching Should Work Well!

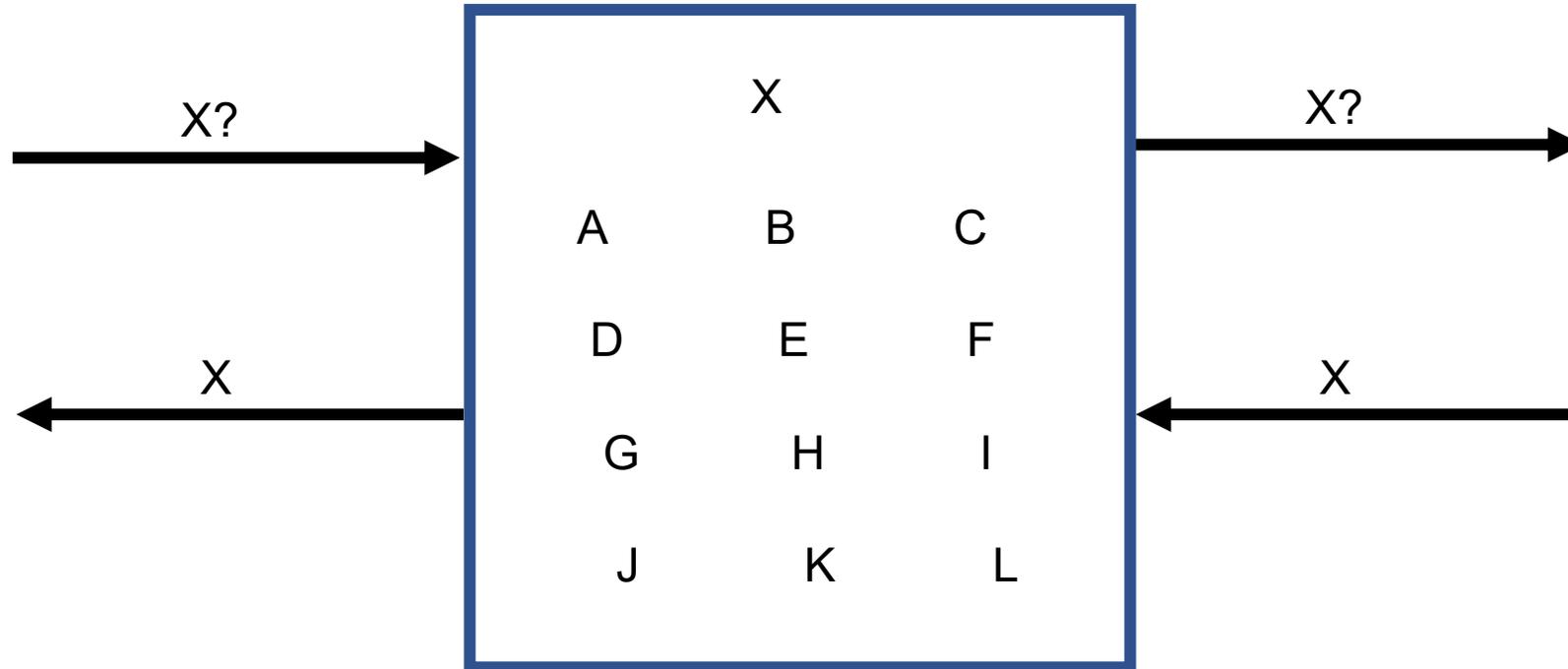


CDN Cache Hit



On cache hit, retrieve the object from the cache!

CDN Cache Miss



If I want to store X, what do I get rid of to make space?

Cache Algorithms 101

- First In First Out (FIFO)
 - Least recently used (LRU)
 - Least frequently used (LFU)
 - Belady (Offline optimal)
-
- (Note: all fully associative today)

First-In-First-Out (FIFO)

- Evict objects added to cache longest ago
- Very simple!
- 3 item cache example:
 - Request stream: a, b, a, c, a, d, a, e, a, f, g
- Can we do better?

Least Recently Used (LRU)

- Evict object used longest ago
 - “Objects used more recently are more likely to be accessed again”
 - Exploits temporal locality
- Implementation: Update access time for every hit
- 3 item cache example:
 - Request stream: a, b, a, c, a, d, a, e, a, f, g
 - Request stream: h, h, h, i, j, k, h

Least Frequently Used (LFU)

- Evict object with fewest hits
 - “Objects used more often are more likely to be accessed again”
 - If tie, use LRU
- Implementation: Update access count for every hit
- 3 item cache example:
 - Request stream: a, b, a, c, a, d, a, e, a, f, g
 - Request stream: h, h, h, i, j, k, h
 - Request stream: l, l, m, n, o, m

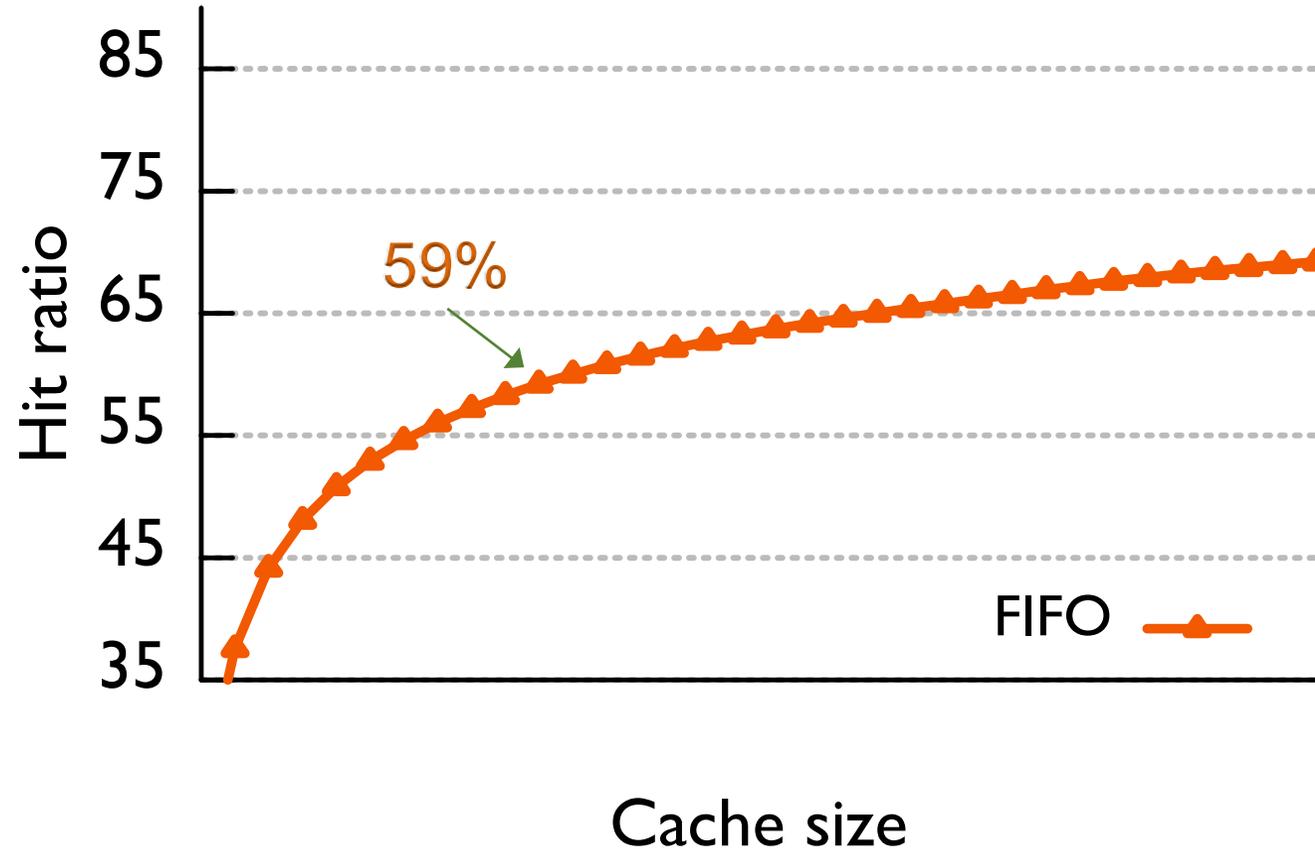
Belady (Offline Optimal Caching)

- What is the best a caching algorithm could do?
- Offline: uses knowledge of the future
 - (Can't use in practice)
- Evict the object with the furthest **next** access time
 - Worst object to keep in the cache
- 3 item cache example:
 - Request stream: h, h, h, i, j, k, h
 - Request stream: l, l, m, n, o, m

Effectiveness of Algorithms for CDN Caching?

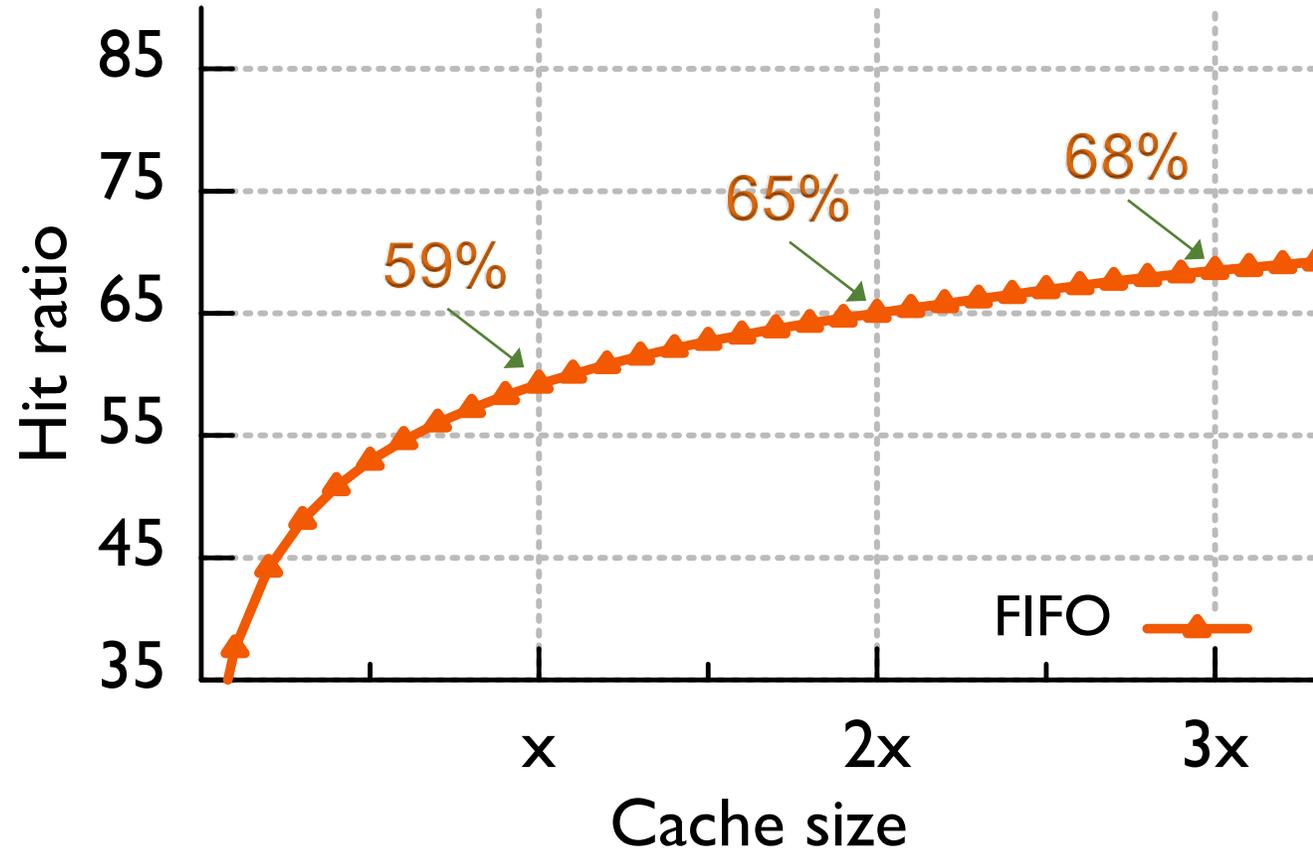
[Figures from From “An Analysis of Facebook Photo Caching” at Symposium on Operating System Principles, 2013.]

Edge Cache with Different Sizes



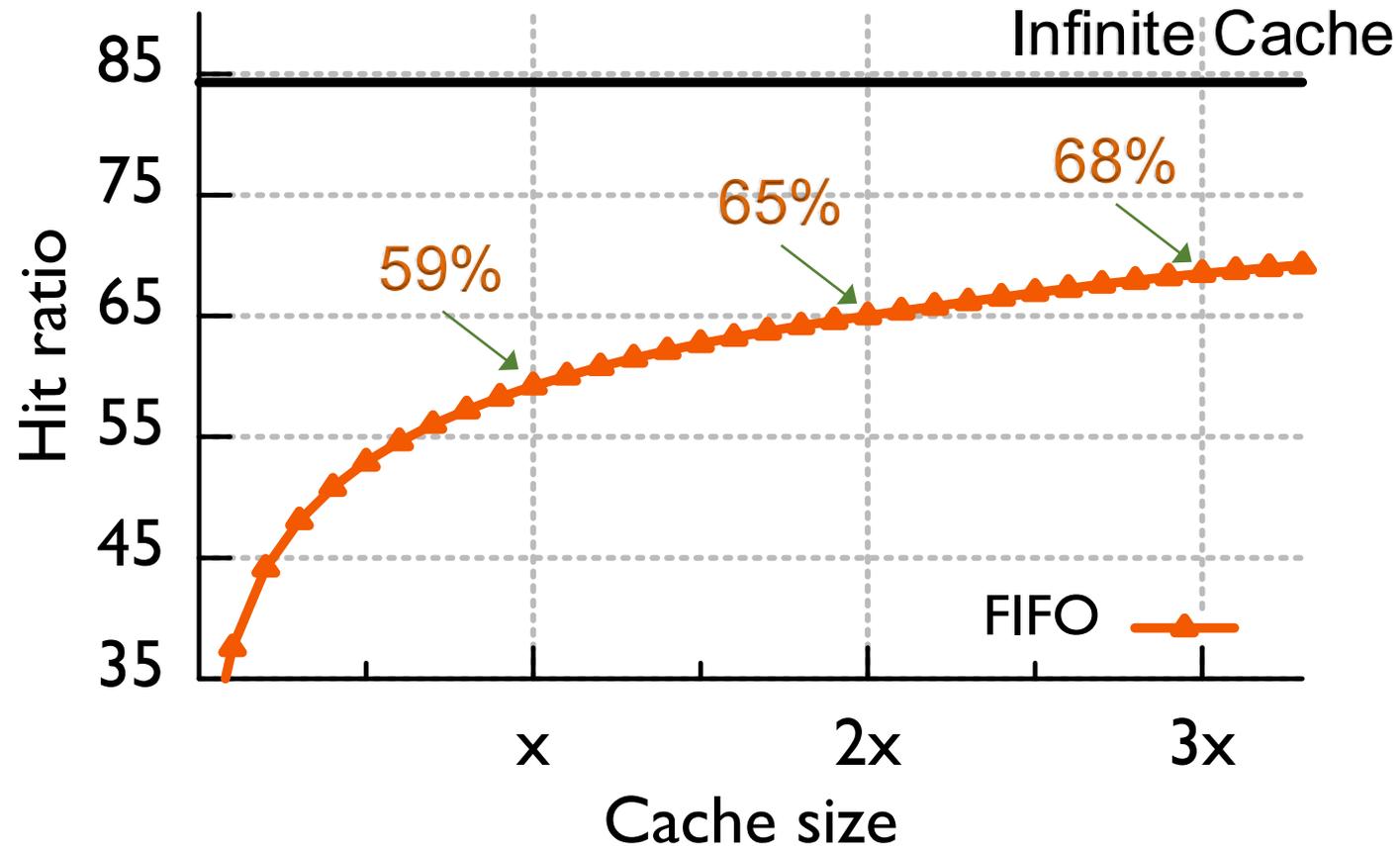
- Facebook's San Jose CDN edge cache circa 2013

Edge Cache with Different Sizes



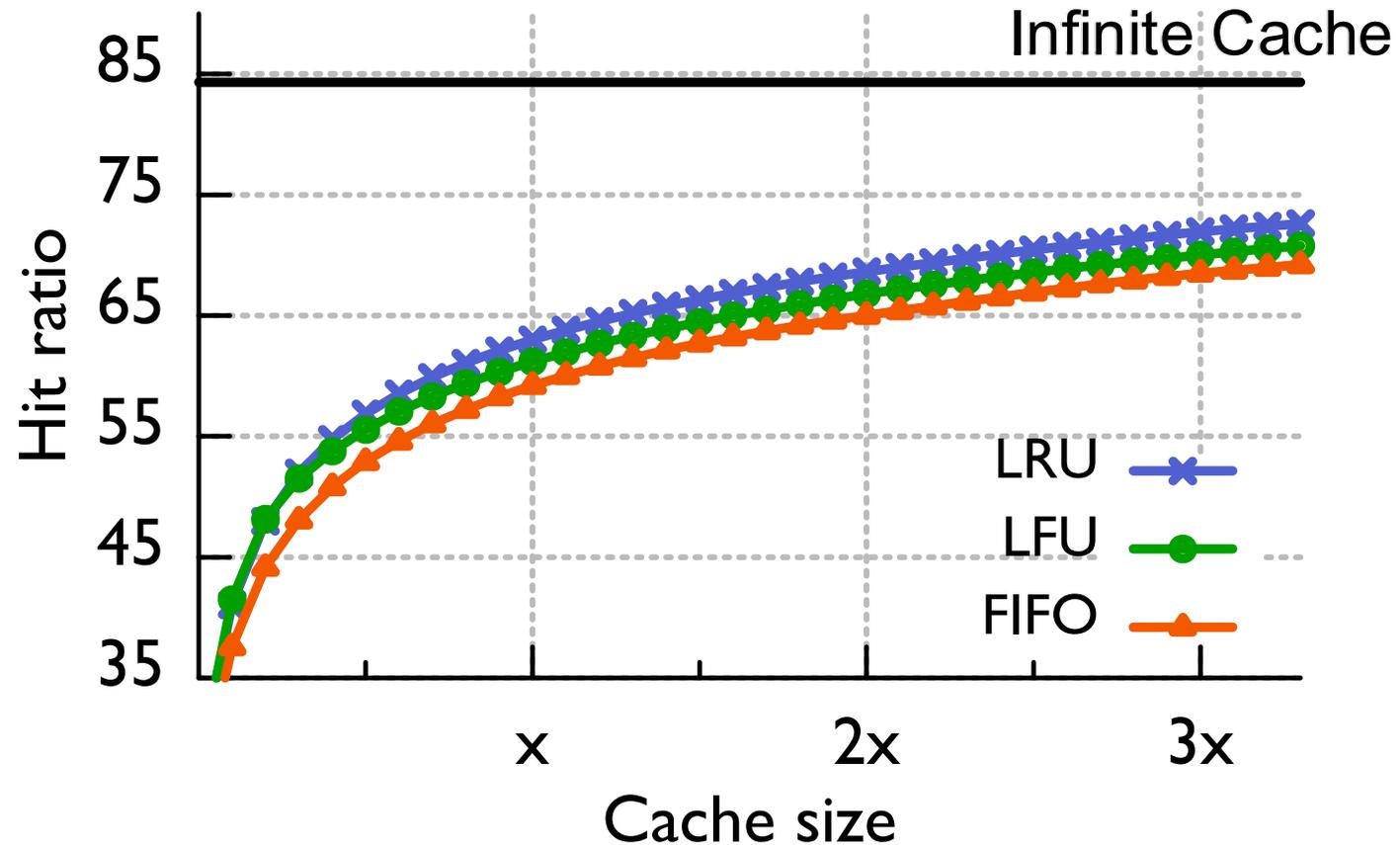
- “x” estimates deployment size (59% hit ratio)

Edge Cache with Different Sizes



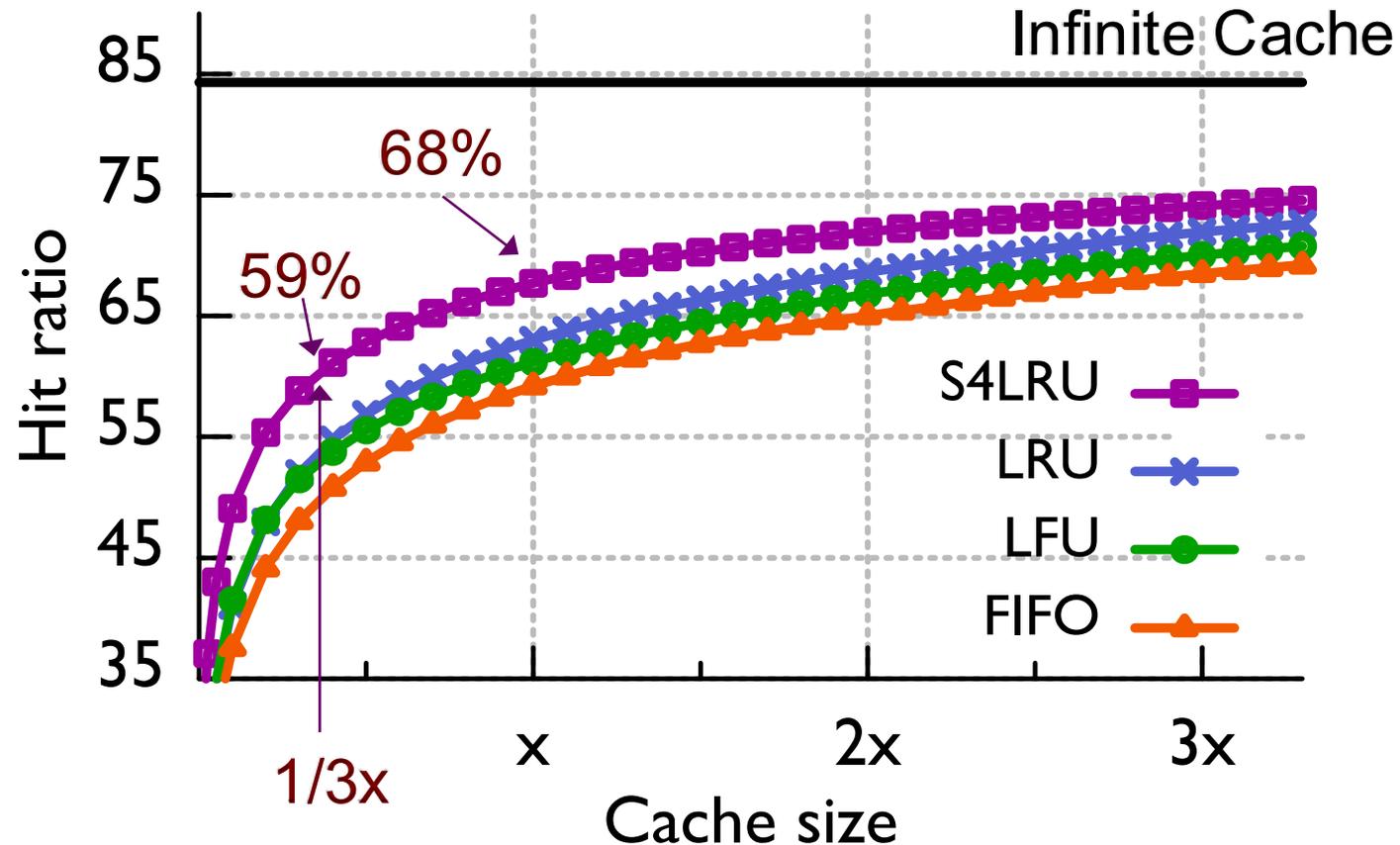
- “Infinite” size ratio needs 45x of capacity

Edge Cache with Different Algos



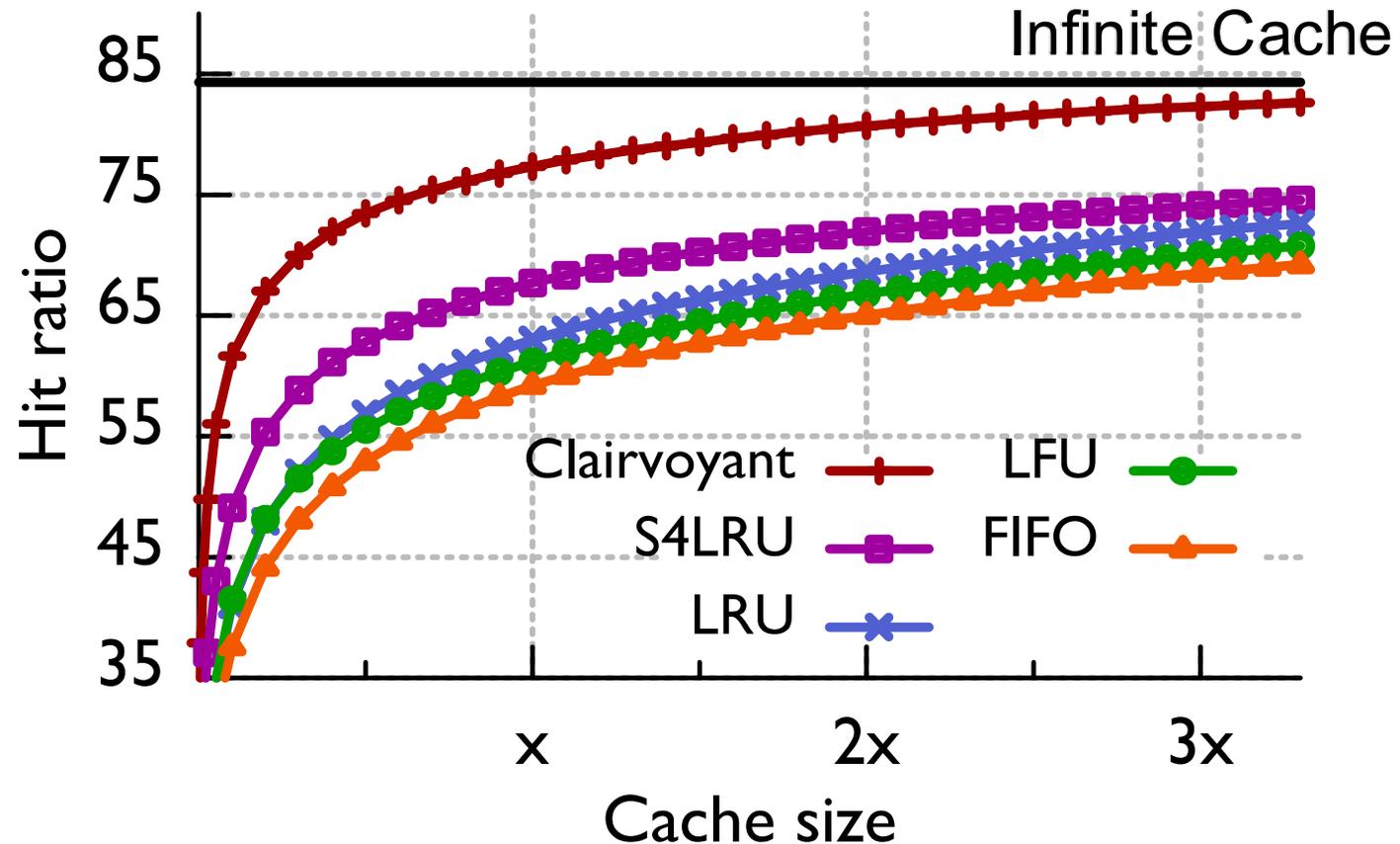
- LRU > LFU > FIFO

Edge Cache with Different Algos



- S4LRU is a more complex algorithm, uses recency and frequency

Edge Cache with Different Algos



- Clairvoyant (Bélády) shows we can do much better!

Cache Consistency

Some Web Content is Not Cacheable

- Dynamic content
 - E.g., stock prices, scores
- Content generated by scripts
 - Results depend on the specific parameters
 - E.g., <https://www.google.com/search?q=php+script+url>
- Personalized content
 - E.g., based on cookie sent by the browser
- Encrypted content
 - Cannot decrypt without the appropriate key

Last Updated 9:19pm EST  

^IXIS NASDAQ INSURANCE	4,445.89	-19.43	-0.4%
^GSPC S&P 500	1,367.59	+1.85	+0.1%
AAPL APPLE INC.	525.76	+3.35	+0.6%
T AT&T INC.	30.36	+0.02	+0.1%
GOLD RANDGOLD RESOURCE	114.86	-0.87	-0.8%
V VISA INC.	116.86	-0.68	-0.6%
YHOO YAHOO! INC.	14.86	-0.03	-0.2%

Powered by EduLifeLine.com



Cache Consistency Challenges



Personal
Cache

Shared
Cache



Web cache needs to know

- Whether to cache an item
- How long to cache an item
- Whether to check an item's freshness
- Whether it is okay to return a stale item
- Whether the item has sensitive data

Cache Consistency Challenges



Personal
Cache

Shared
Cache



Web cache needs to know

- Whether to cache an item
- How long to cache an item
- Whether to check an item's freshness
- Whether it is okay to return a stale item
- Whether the item has sensitive data

Server knows the content

- Whether the item is dynamic
- How often the item changes
- Whether the item has changed
- Whether stale information is useful
- Whether item contains sensitive data

Scalability challenge: the server cannot remember every client that has cached an item

HTTP Response Header for Cache Control

- Whether to cache
 - no store: no cache should store it
- Who should cache
 - private: only a private cache (e.g., browser)
 - public: any cache, including shared ones
- How long to cache
 - max-age=N: for N seconds
 - must-revalidate: check with the server (don't return stale item)

Cache-Control: public, max-age=86400, must-revalidate

Cache Validation: Client Checks Freshness



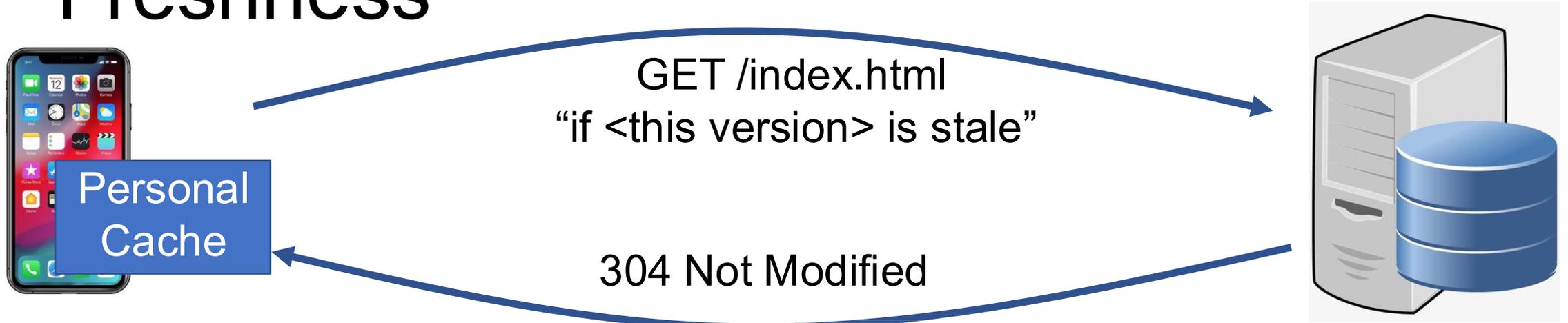
Personal
Cache

GET /index.html
"if <this version> is stale"

304 Not Modified



Cache Validation: Client Checks Freshness



How do they identify the "version"?

- Timestamp
 - When the item was modified by the server
 - E.g., Last-Modified: Tue, 03 Mar 2026 07:28:00 GMT
- Version number
 - Entity tag provided by the server
 - E.g., ETag: "33a64df551425fcc55e4d42a148795d9f25f89d4"

Cache Placement

Client Machine (e.g., Browser)

Advantages

- Very low latency
- Preserves access bandwidth
- Available when disconnected

Disadvantages

- Low hit rate due to “cold” misses
- Many cache consistency checks
- Incomplete logs at the server



Personal
Cache



Client Network (Forward Proxy Cache)

Advantages

- Low latency
- Preserves enterprise bandwidth
- Hits for locally popular content

Disadvantages

- Cost to deploy the cache
- Many consistency checks
- Incomplete logs at the server



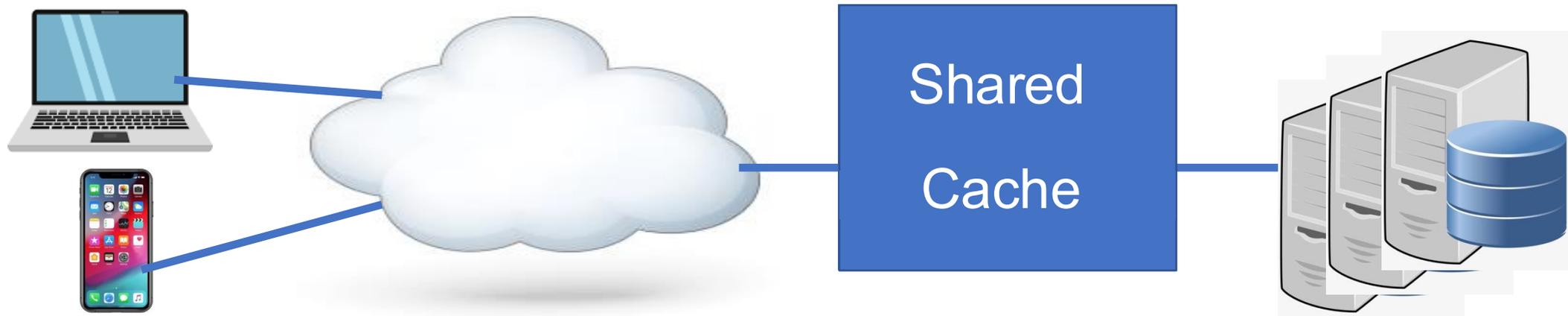
Server Network (Reverse Proxy Cache)

Advantages

- High hit rate across global users
- Greater cooperation with server
- Complete request logs for server
- Preserves server bandwidth

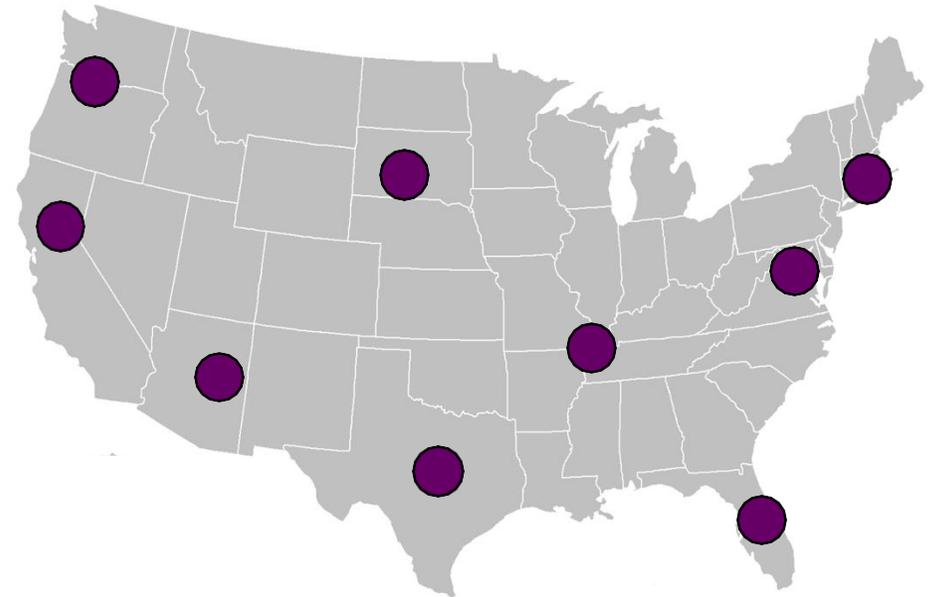
Disadvantages

- Costs to deploy the cache
- Does not reduce latency much
- Consumes wide-area bandwidth



Content Distribution Network (CDN)

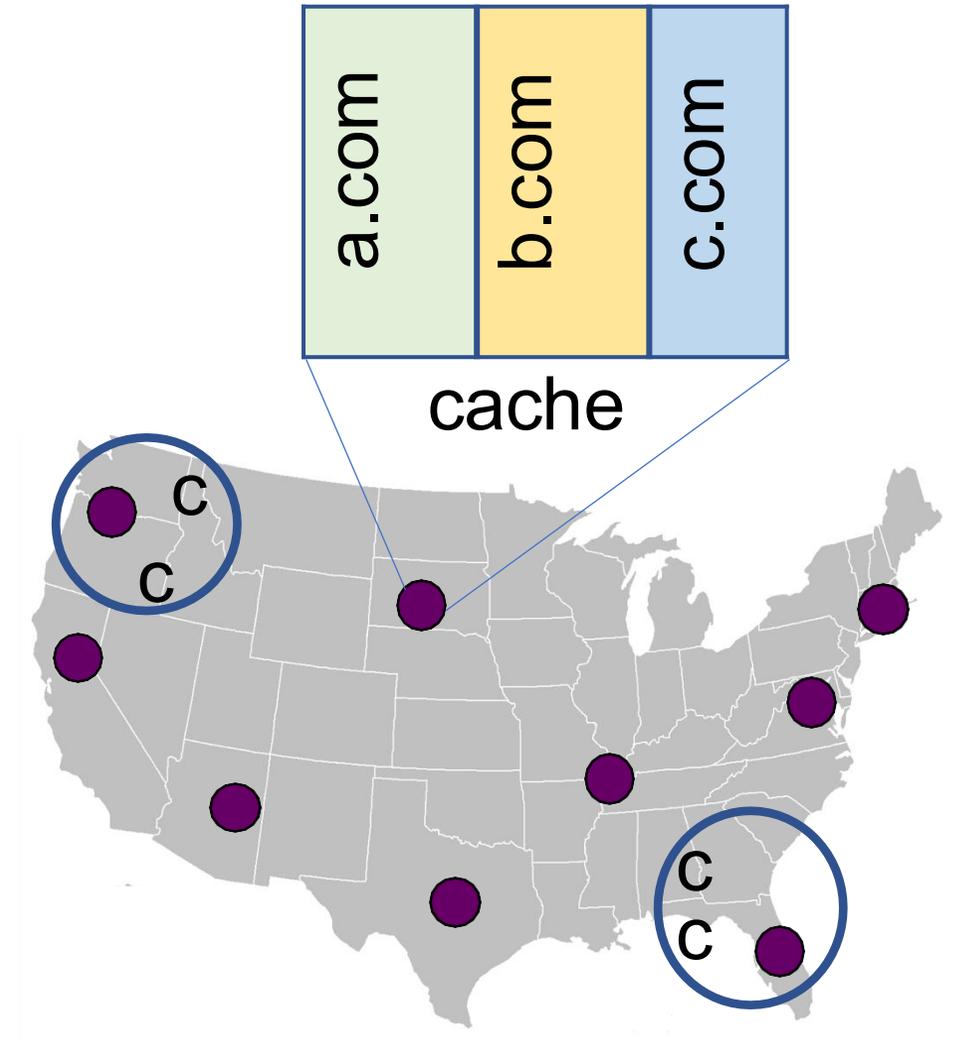
- Outsourced caching infrastructure
 - Caching for clients and servers
 - Dedicated equipment and software
 - Trained staff, best practices, etc.
- Coordination with the server
 - Generating non-cacheable content
 - Providing detailed measurement data
- Smart cache placement
 - Many caches: handle large request load
 - Close to many clients: reduce latency



More than 4200 locations in 135 countries

CDN Challenges

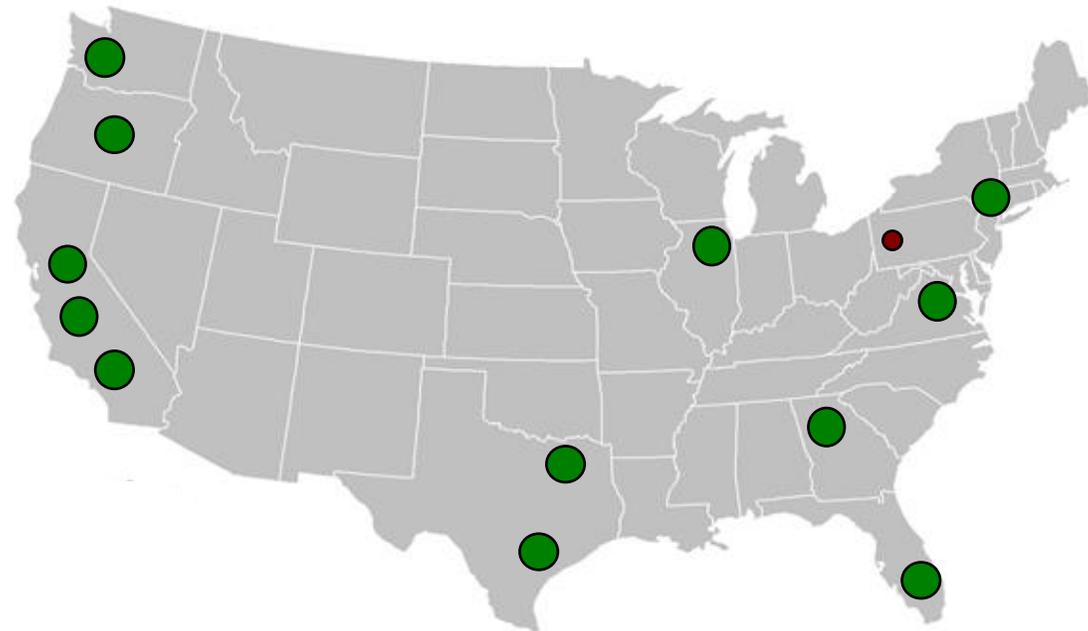
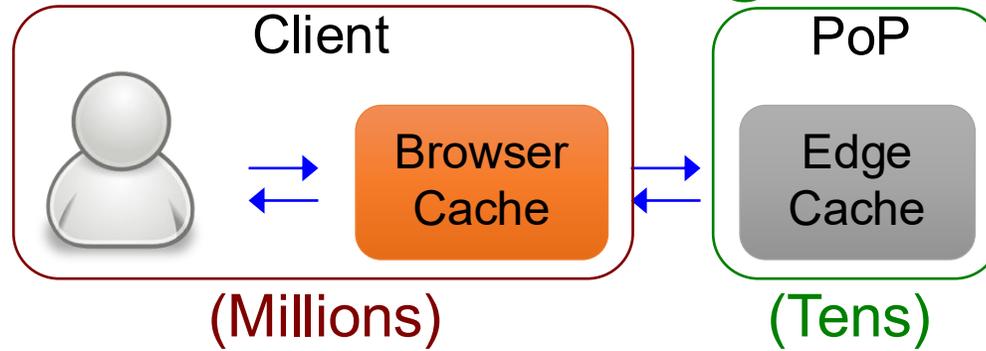
- Where to place edge sites?
 - Close to many clients, with reasonable cost
- Where to replicate a server's content?
 - Many edge sites → duplicated data
 - Few edge sites → larger client latency
- How to direct a client to an edge site?
 - Proximity: for low latency
 - Light load: to reduce congestion
- How to manage each cache?
 - Maximize hit rate?
 - Minimize miss penalty?
 - Fairness across origin servers?



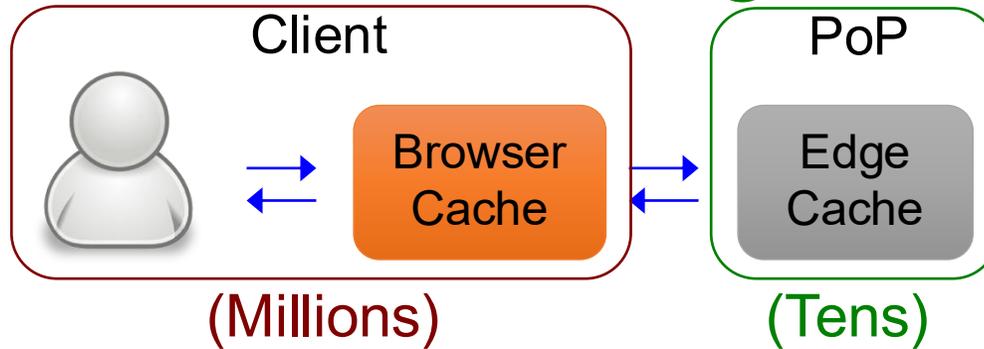
A Hierarchy of CDN Caches

[Figures from Qi Huang's 2013 SOSP Talk]

Geo-distributed Edge Cache (FIFO)



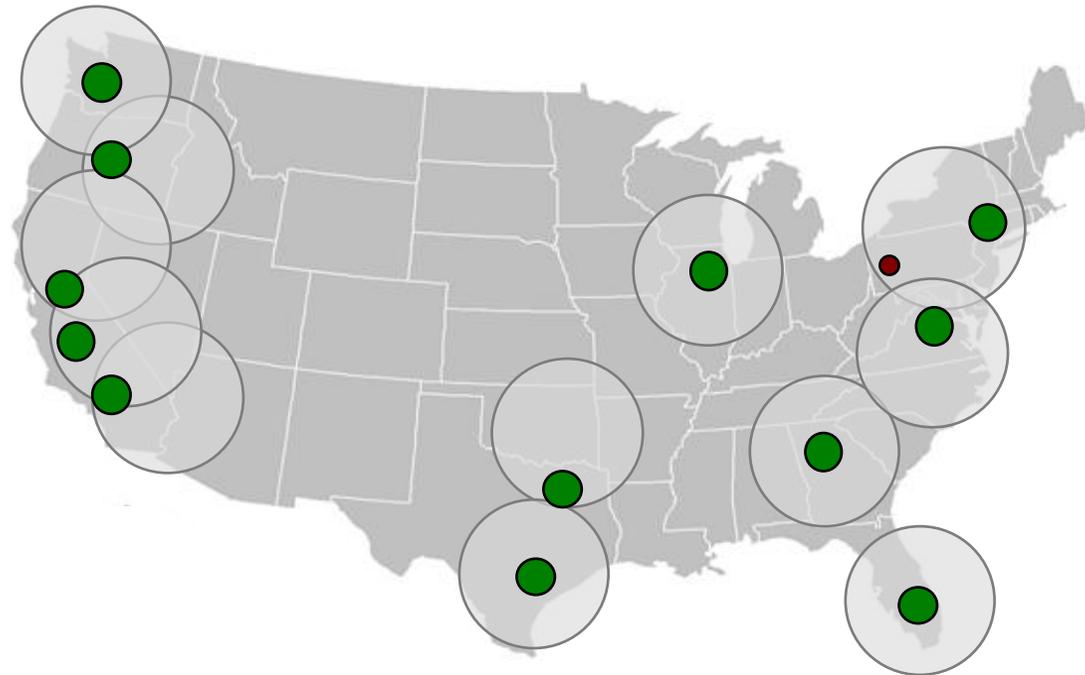
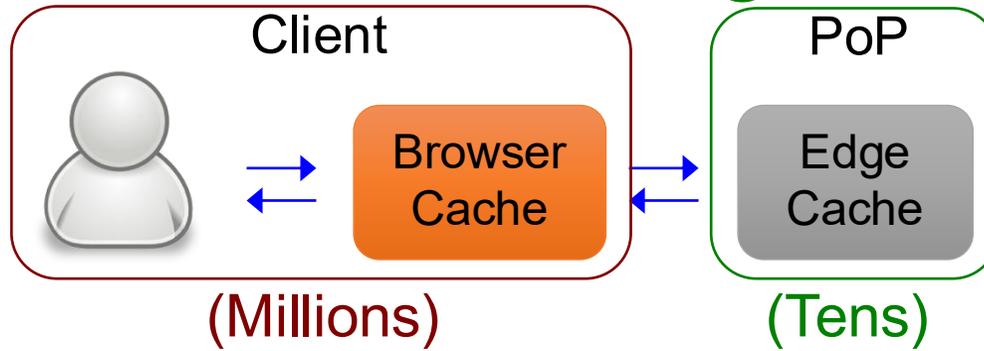
Geo-distributed Edge Cache (FIFO)



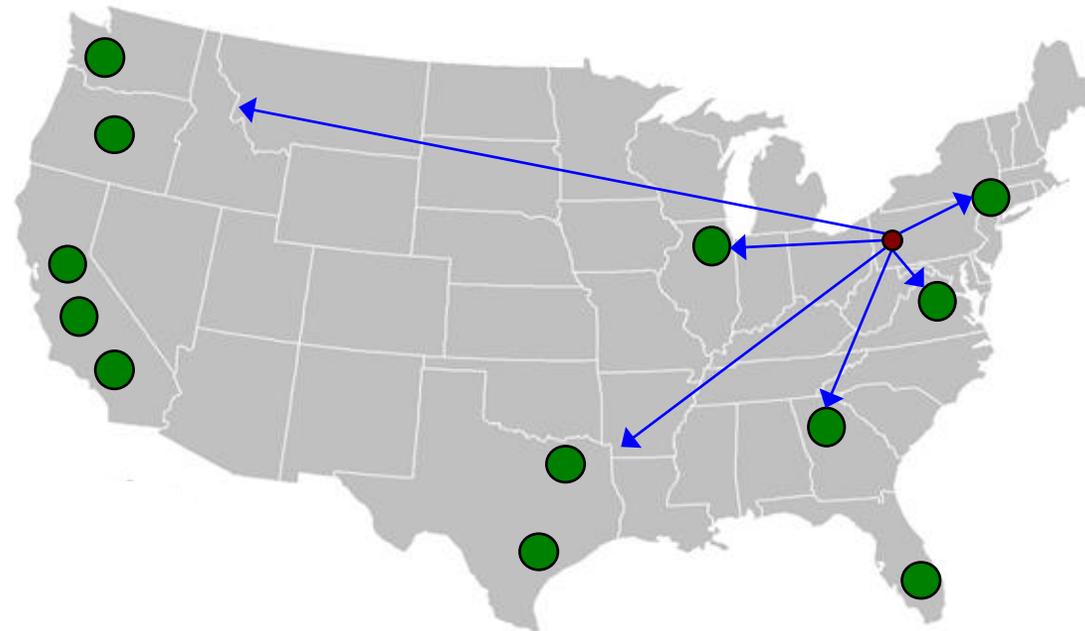
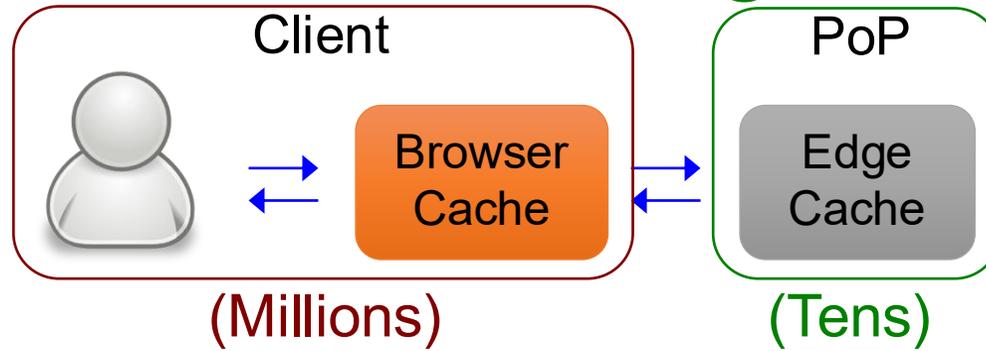
Purpose

1. Reduce cross-country latency
2. Reduce Data Center bandwidth

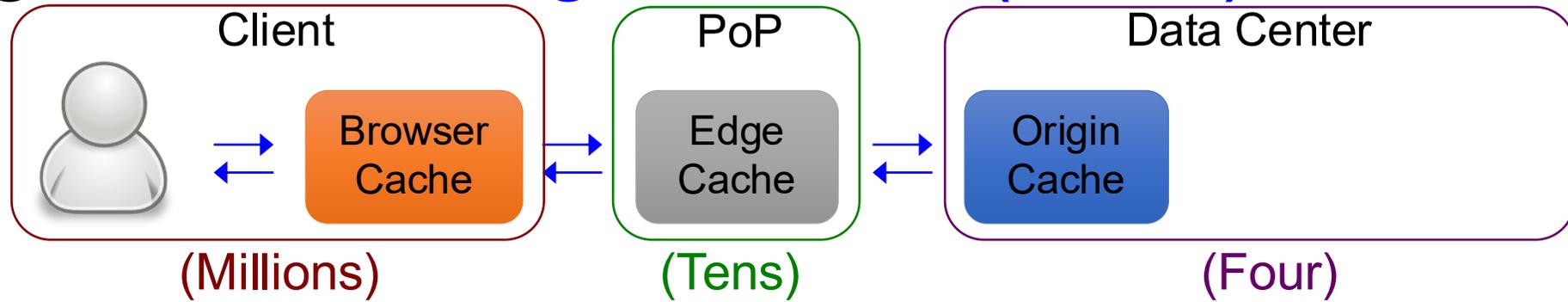
Geo-distributed Edge Cache (FIFO)



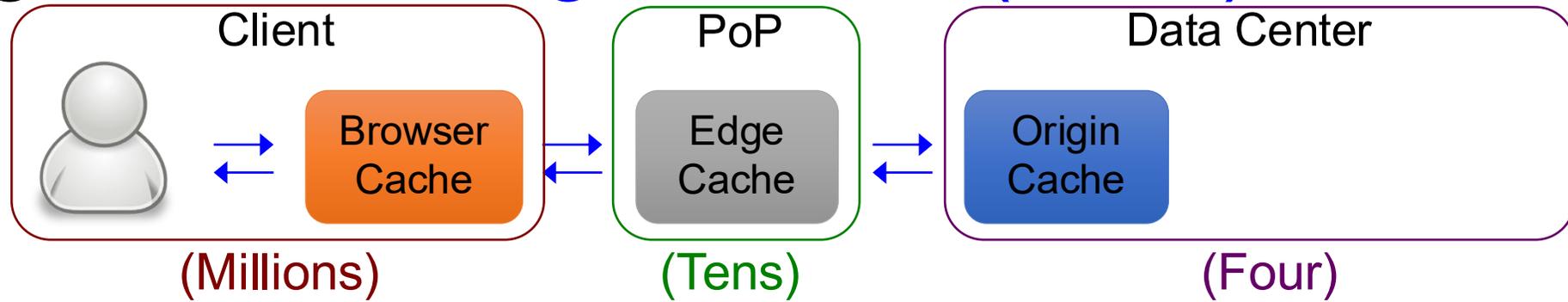
Geo-distributed Edge Cache (FIFO)



Single Global Origin Cache (FIFO)



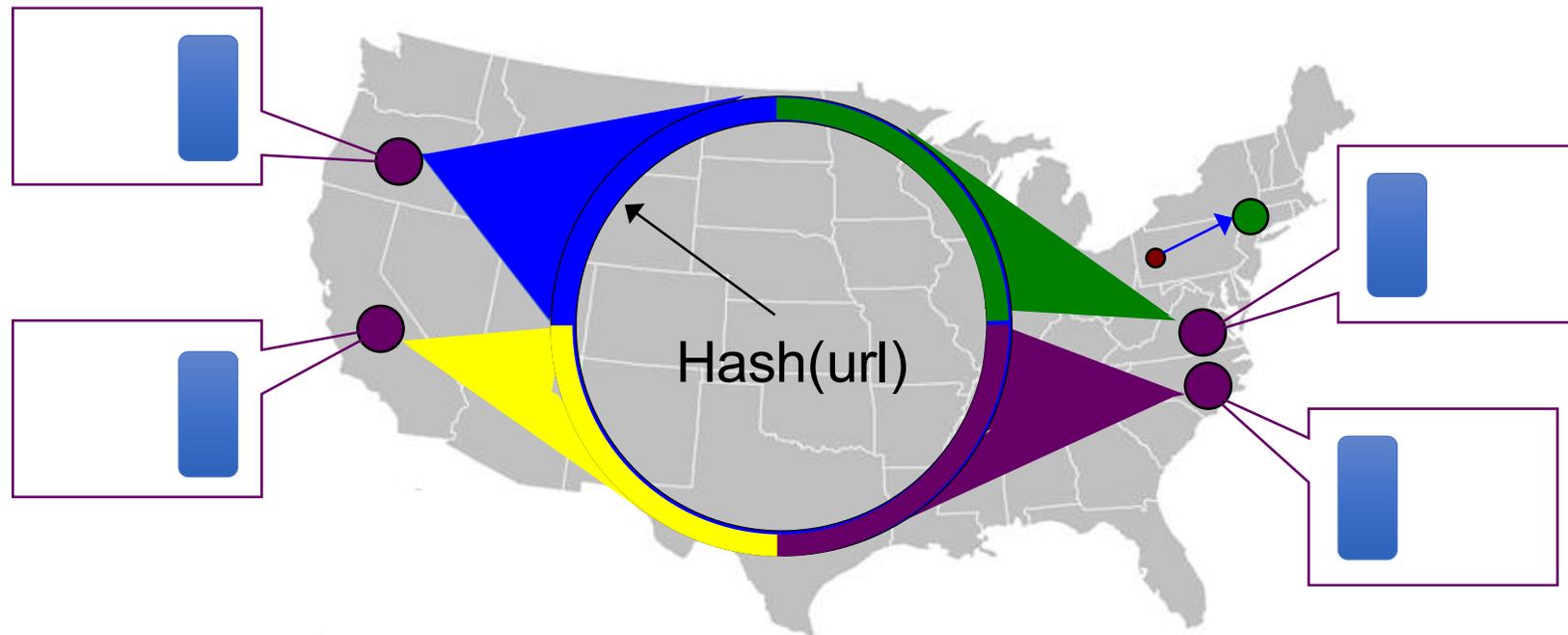
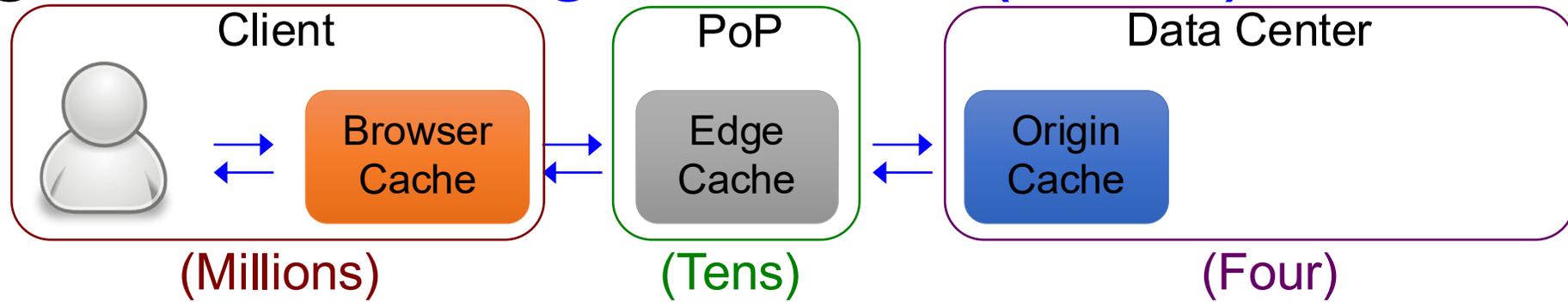
Single Global Origin Cache (FIFO)



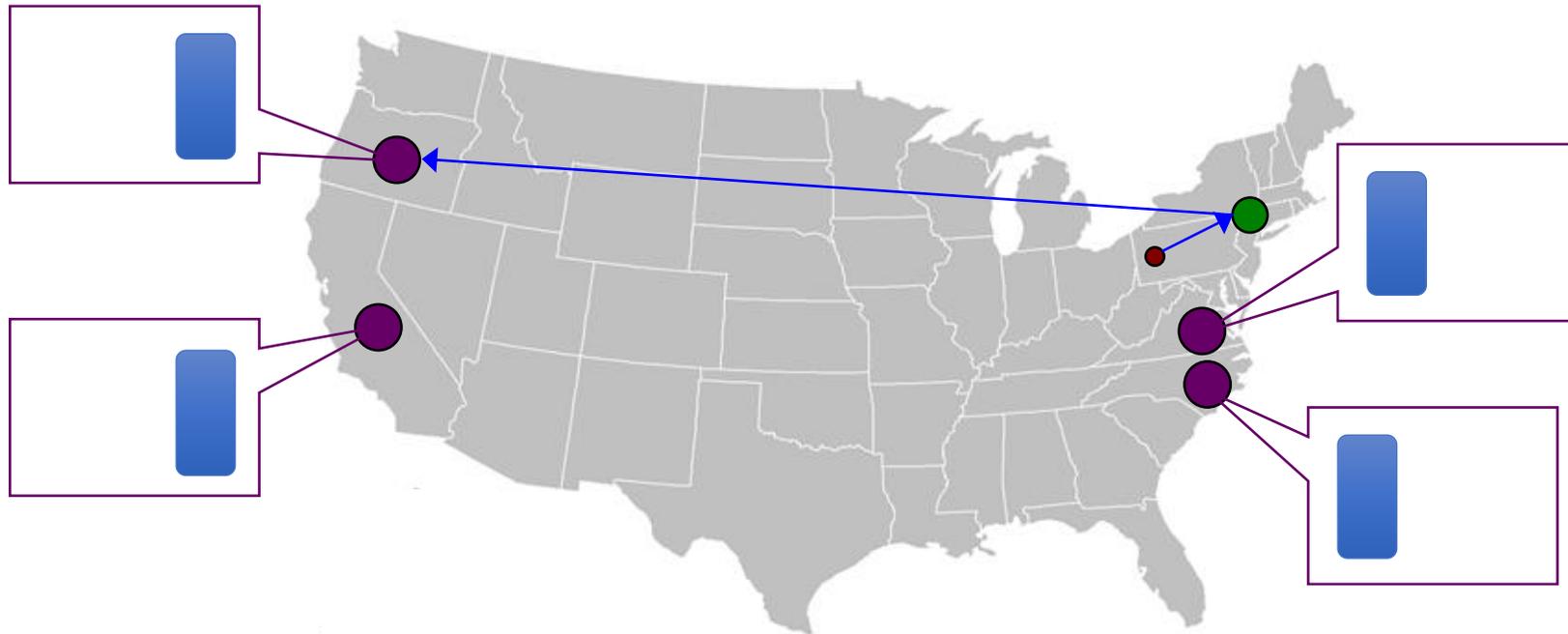
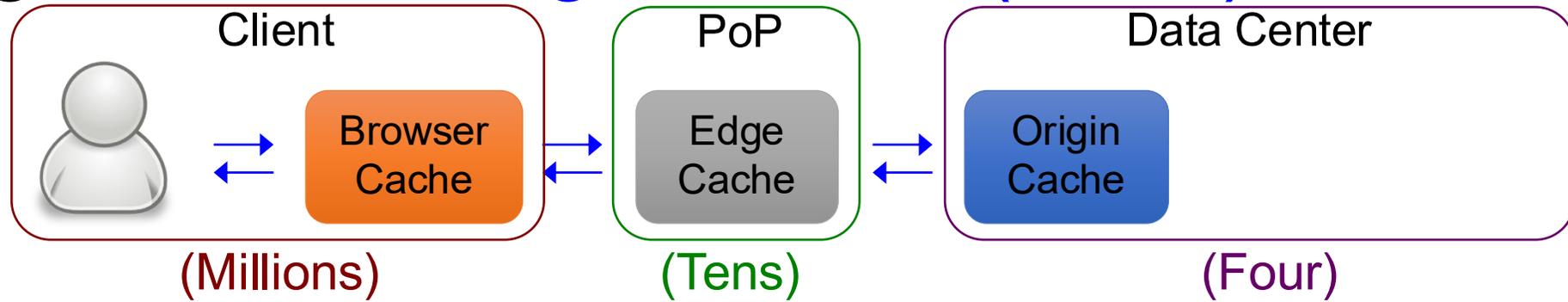
Purpose

1. Minimize I/O-bound operations

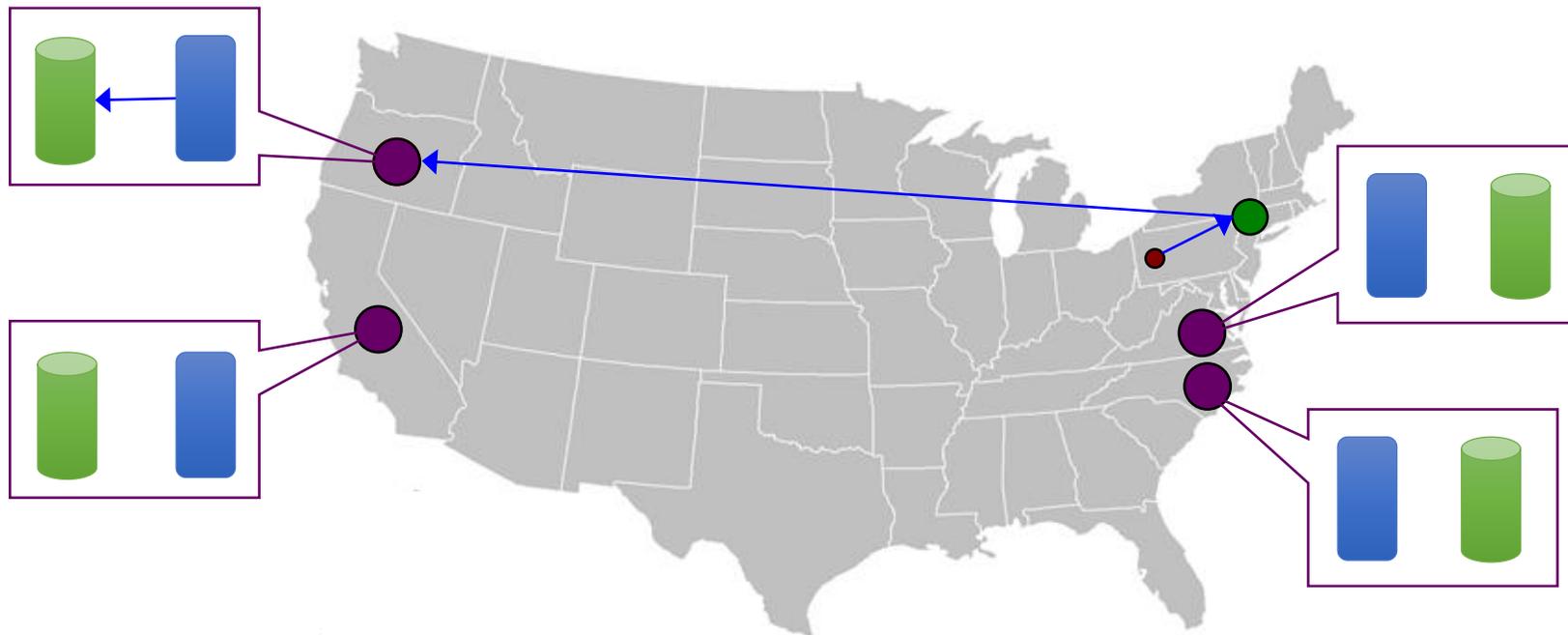
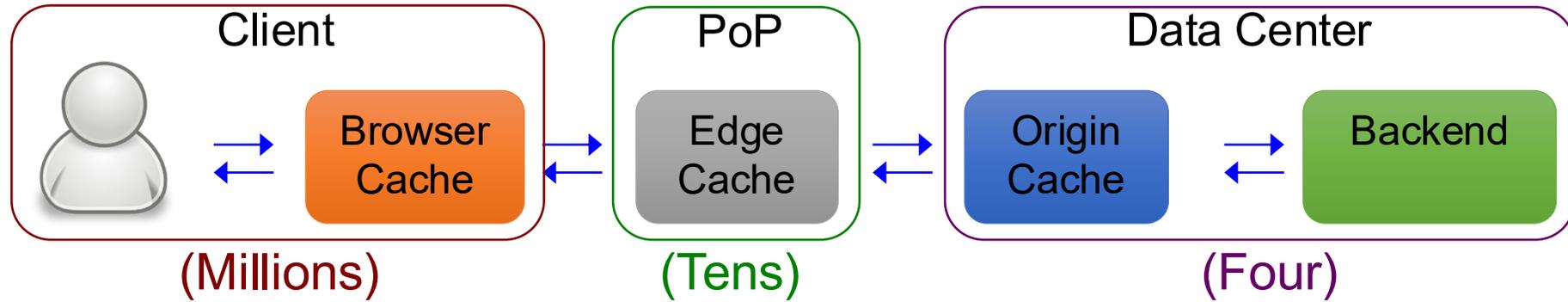
Single Global Origin Cache (FIFO)



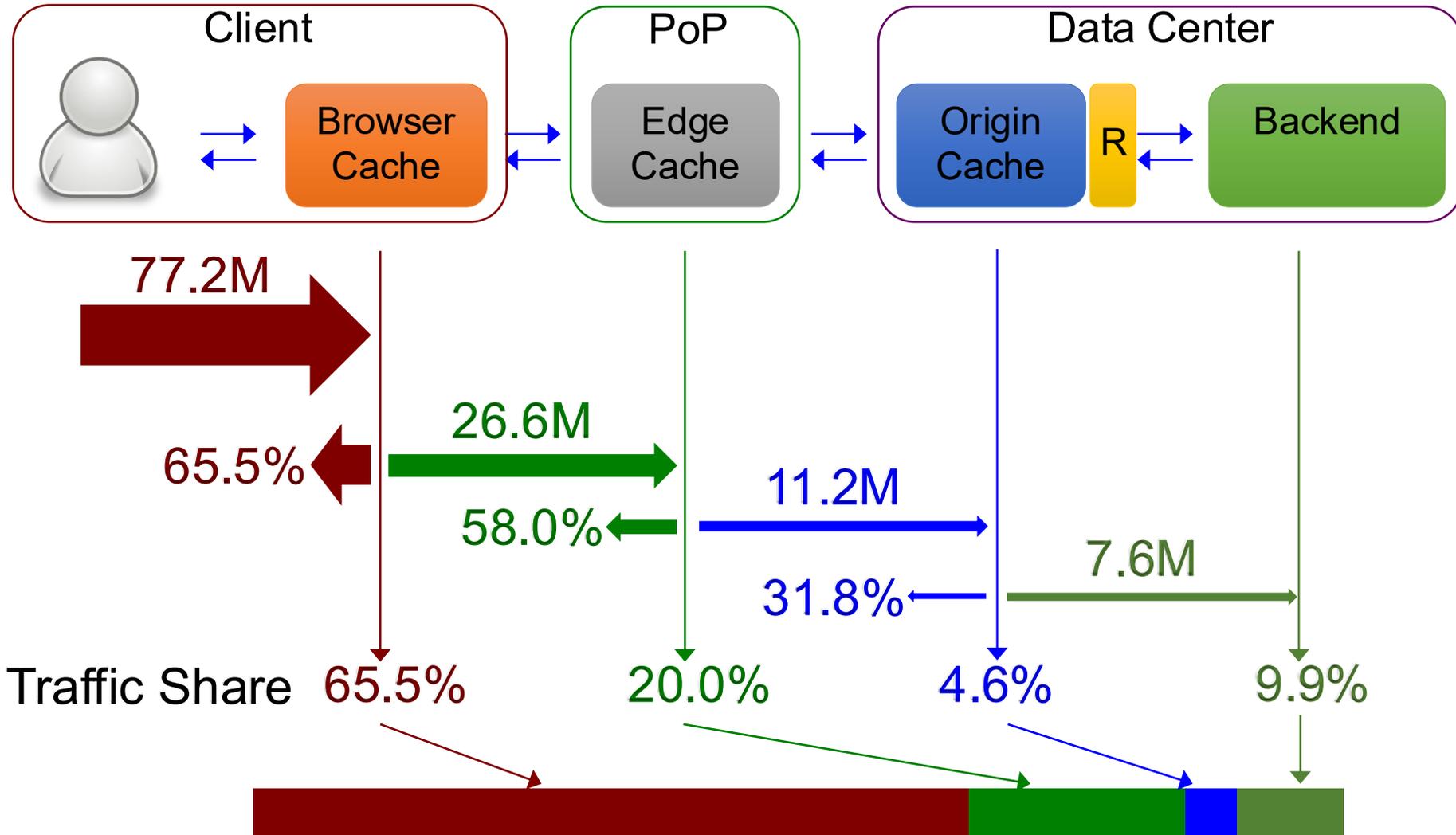
Single Global Origin Cache (FIFO)



Backend



CDN Effectiveness



Conclusions

- Downloading a Web page
 - Name resolution, transport connection, secure session, web messages
- Benefits of caching
 - Reduces user latency, server load, and network bandwidth
- Cache replacement
 - Maximize hit rate by trying to predict the future
- Cache consistency
 - Efficient ways to avoid returning unnecessarily stale responses
- Content distribution networks
 - Caching close to clients, while working on behalf of the servers

