Load balancing
Overview of implementations

- In the following we will discuss different implementations of load balancing
  - with the Domain Name System
  - with hardware
  - with software
    - with a modified IP stack
    - with a user-space application

Load balancing

With DNS: Round-robin DNS

- The Domain Name System (DNS) resolves domain names, like heig-vd.ch into IP addresses
  - The owner of a domain name can configure DNS with more than one IP address for a domain name
  - This can be used for simple load balancing
- Round-robin DNS is a load balancing mechanism entirely implemented by DNS servers and DNS clients
  - The service provider configures DNS with multiple A records for a domain name
  - When a DNS client tries to resolve the domain name the server responds with multiple A records and **changes their order at each response**
  - The DNS client picks the first address and uses it to connect to the service

```
$ dig google.com

; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status:
NOERROR, id: 41863
;; flags: qr rd ra; QUERY: 1, ANSWER: 11,
AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.          IN  A

;; ANSWER SECTION:
google.com.       87  IN  A   173.194.116.37
google.com.       87  IN  A   173.194.116.32
google.com.       87  IN  A   173.194.116.33
google.com.       87  IN  A   173.194.116.38
google.com.       87  IN  A   173.194.116.35
google.com.       87  IN  A   173.194.116.40
google.com.       87  IN  A   173.194.116.46
google.com.       87  IN  A   173.194.116.36
[...]
```

Load balancing
With DNS: Round-robin DNS

- Advantages
  - Easy to set up, does not require any additional infrastructure
- Disadvantages
  - The mechanism distributes requests blindly without knowing the load of the servers or if they are working or not
  - Adding or removing servers is a slow process: Changes to a DNS zone propagate slowly.
    - Especially critical when a machine crashes, it needs to be removed quickly to avoid users trying to connect to it.

Load balancing

With hardware: Load balancing appliances

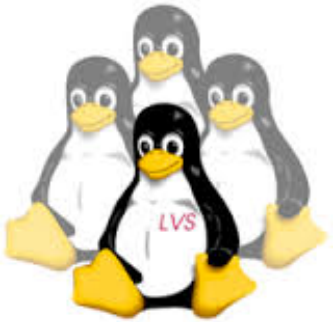
- The most straightforward way to balance requests between multiple servers is to use hardware appliances.
  - An *appliance* is a hardware box with preinstalled software. For the customer it is a black box that he simply plugs in.
  - In the industry hardware load balancers are called several different names: Web switch, Application switch, Content switch, Content router
- Often load balancing is just one function among many others: web caching, GSLB, HTTPS acceleration, DOS protection, ...
- A load balancing appliance acts like a network address translation (NAT)-capable router, just backward.

- Advantages:
  - Able to handle very high loads
    - 15 M concurrent sessions
    - 50 Gbit/s throughput
- Disadvantages:
  - Very expensive, in the range of $100k

Load balancing

With software: IP Virtual Servers

- IP Virtual Servers are similar in purpose and use to web switches, but they are implemented as a software package that can be run on traditional hardware.
  - Composed of
    - a user space application
    - a customized IP stack
- A virtual server exposes a single virtual IP address for a service and the distributes the traffic across a set of machines with private IP addresses
- Example: Linux Virtual Server



Load balancing

With software: Application layer load balancers

- Application layer load balancers operate entirely in user space
  - Bundled as stand-alone programs or web server modules
- Examples:
  - **HAProxy** TCP and HTTP load balancer
  - **Pound** reverse proxy and load balancer
  - Apache web server, module **mod_proxy_balancer**
  - **nginx** reverse proxy server
  - **Varnish** caching HTTP reverse proxy
  - Amazon Web Services (AWS) **Elastic Load Balancer** (ELB)

Load balancing

How to decide which server gets the request?

- Typically the goal of a load balancer is to spread the load evenly between servers.
  - It is surprisingly difficult for a load balancer to measure the load of a server:
    - A server processes requests very quickly (from a few milliseconds to a few hundred milliseconds)
    - The system load can change rapidly
    - Sending information about the system load to the load balancer takes time — so much time that the information is no longer accurate. This is called the *staleness* problem.
  - Therefore one tries to use other information to distribute load

Load balancing
Policies

- Load balancers may implement different algorithms or **policies** to determine to which server a request should be sent
  - **Round robin** — One request per server in a uniform rotation. Variation: The same amount of traffic per server in a uniform rotation.
  - **Weighted round robin** — Like round robin, but the administrator can assign weights to the servers to direct more load to bigger servers.
  - **IP hash** — Choose the server by hashing the user's IP address.
  - **Least connections** — Choose the server to which the load balancer currently has the least number of connections established.
  - **Predictive** — Usually based on *round robin* or *least connections* with some added ad-hoc expressions to compensate for information staleness.
  - **Available resources** — Choose the server with the most available resources (e.g., lowest system load). Suffers from staleness problem.
  - **Random** — Choose a server randomly. Often combined with *available resources* to form a probabilistic algorithm.
  - **Weighted random** — Like *random*, but the administrator can assign weights to the servers.

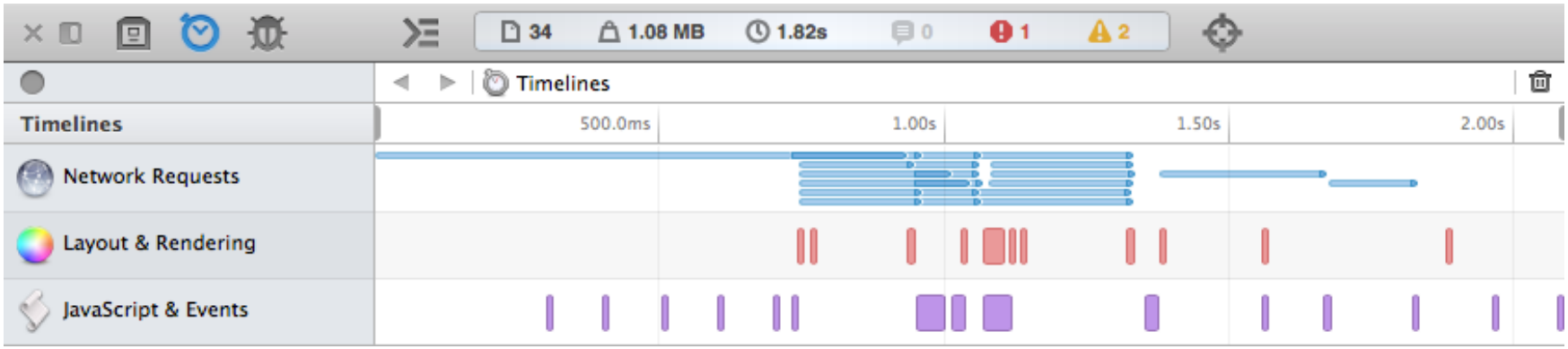
Load balancing
Session stickiness

- Some web applications are not stateless because they store the state of a user session. When the session is stored locally by the server, all user requests must go to the same server or the user will see weird behavior.
  - A load balancer with **session stickiness** routes the initial user request according to normal policy, but then sends all subsequent requests to the same server.
  - The concept of session stickiness goes against the goal of balancing load. If possible, one should avoid it by making applications stateless.
    - Session state can be stored on the client side in cookies.
    - Or on a shared database.
- How can a load balancer with session stickiness know which sessions do currently exist?
  - By listening on the conversation between client and server. The server needs to store a session identifier with the client that the client then returns with every request. This can be done by
    - a cookie (JSESSIONID, PHPSESSIONID),
    - rewriting the URLs to append the id in a parameter.
  - The session automatically times out if the user does not send any more requests.
    - The load balancer needs to be configured with the same timeout as the server to stay synchronized.

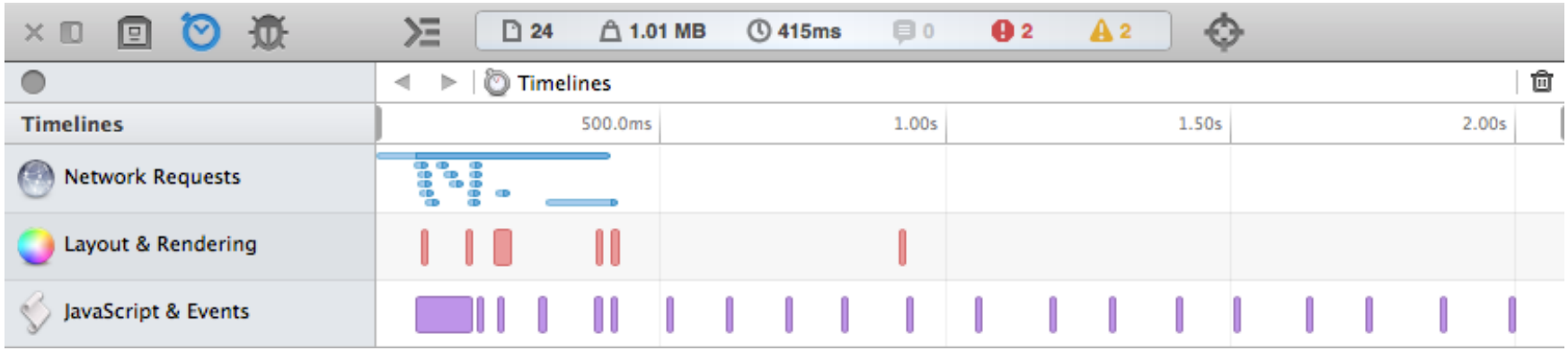
Caching

Introduction — Client-side web caching

- Time to load the page http://www.heig-vd.ch
  - empty cache: 1.8 s

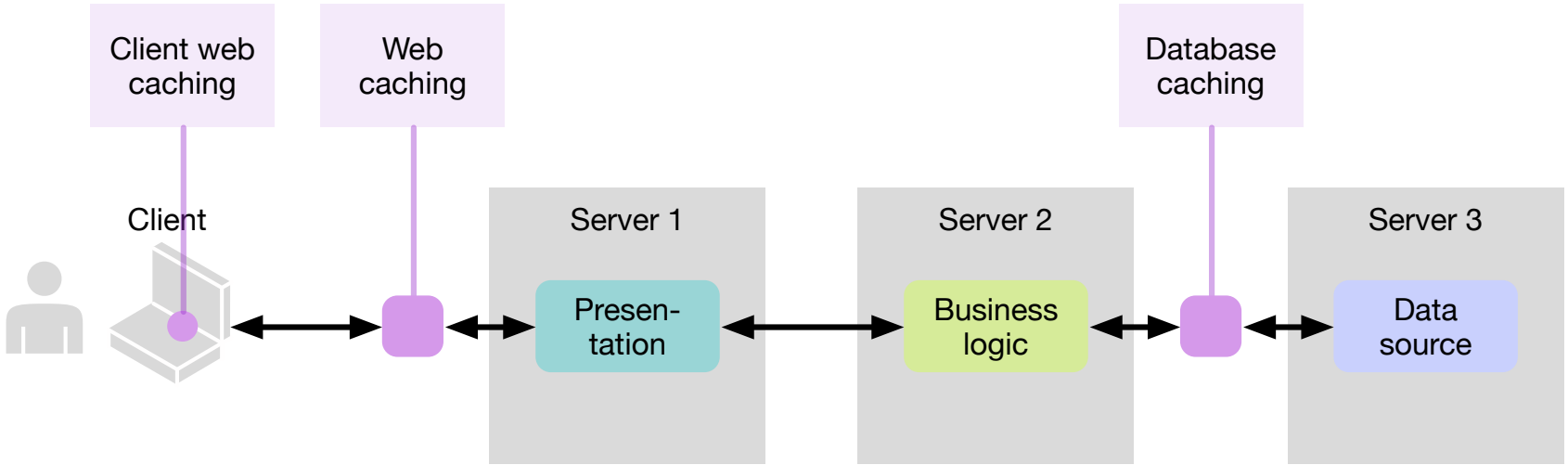


  - primed cache: < 0.5 s



Caching
Overview

- A cache contains a copy for often-requested items. Fetching items from the cache instead of fetching them from the original system is faster and often saves CPU cycles and/or network bandwidth.
  - A client-side cache is already included in every web browser.
- For building a high-performance web sites a cache can be inserted between layers at several points:
  - Between web client and presentation layer: HTTP or **web caching**
  - Between business logic and data source: **Data caching**



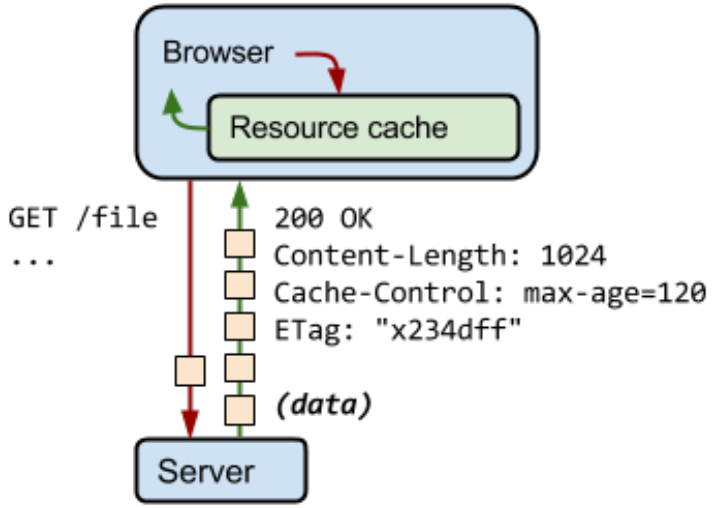
Web caching
What should be cached?

- The web content (HTML pages, images, style sheets, ...) that is sent from server to client over HTTP can be divided into two classes:
  - **Static content** is always the same, independent of the HTTP request, or changes very slowly (new release of the web site). A web server typically retrieves static content from storage and sends it without further processing.
  - **Dynamic content** is *generated* by an application running in the web server. It can be different for each request.
- Typically static content can be cached, while dynamic content cannot.
- Typical candidates for caching are images, style sheets and scripts.
- Caching becomes very important if the volume is big and/or the content needs to be distributed to a lot of people:
  - Videos
  - Software and software updates
  - Anti-virus definition updates
  - ...

Web caching

HTTP caching — Cache-Control header

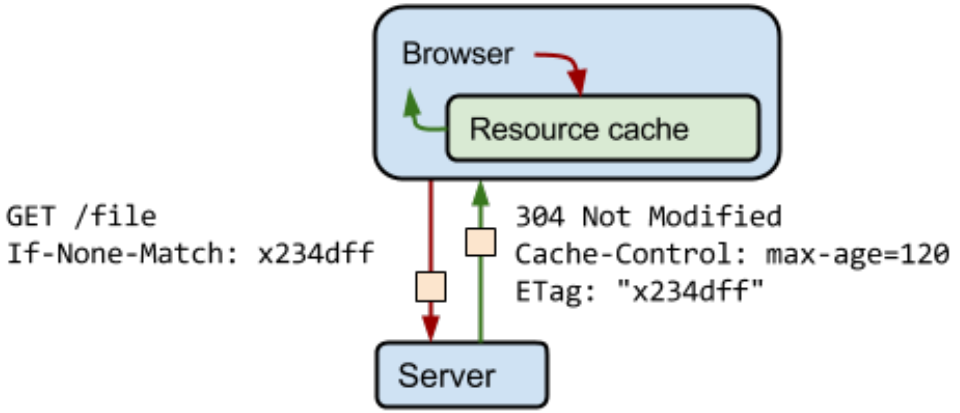
- When an HTTP server returns a resource a client requested, it can specify if and how the resource can be cached and reused by the client and/or intermediate caches.
- This is done with the `Cache-Control` header which can contain the following directives:
  - `max-age` — Specifies the maximum time in seconds that the fetched response is allowed to be reused.
  - `no-cache` — The response can be reused, but the client must first check with the server if the response has changed (validation). If the resource is validated no download is necessary.
  - `no-store` — The response must not be stored. For every future requested the response must be downloaded again.
- `public` — The response can be used by multiple users. Intermediate caches may cache it.
- `private` — The response is intended for a single user. Intermediate caches must not cache it.



Web caching

HTTP caching — ETag header

- There is a separate mechanism to avoid downloading a resource again if it has not changed: the ETag.
  - When returning a response the server computes a token, a fingerprint of the resource, and sends it in the ETag header.
    - The fingerprint is often a hash of the resource.
  - When the client puts the resource into its cache it also stores the token with it.
  - When the client has another request for the resource, and the resource is still in the cache, but it is no longer "fresh",
    - the client sends a request to the server and includes the token
    - the server checks the token against the current resource. If it still matches (the resource has not changed), it returns a "304 Not Modified" response instead of returning the resource.
  - The client can mark the resource in the cache as "fresh" for the duration indicated by the server.



Web caching

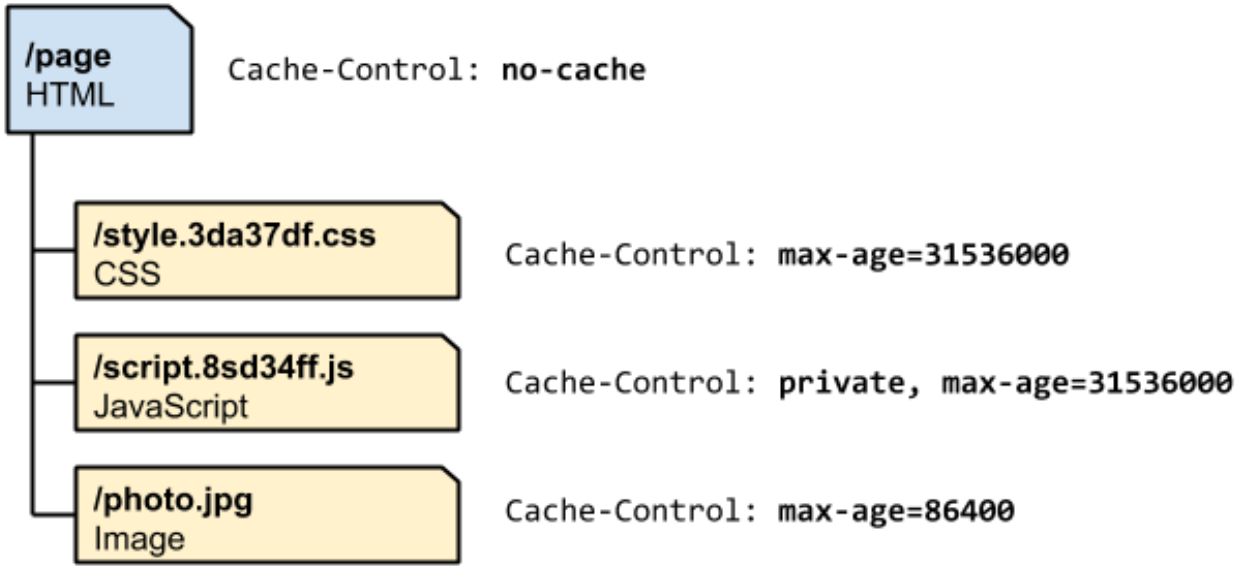
HTTP caching — Client behavior

- When the browser processes a request it proceeds as follows:
  - If there is a "fresh" response for the request in the cache it uses this response.
    - A response is "fresh" if less time elapsed than indicated by max-age.
  - If the response in the cache is no longer "fresh" the browser performs a validation with the server.
    - It sends a request to the server and includes the ETag token.
    - If the server responds with "304 Not Modified" the client uses the response in the cache and refreshes it in the cache.
  - In all other cases the client downloads the resource from the server again.

Web caching

HTTP caching — Cache invalidation

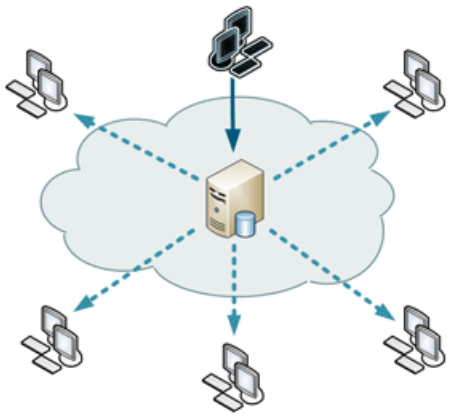
- There are many components of a web page that change infrequently, for example design elements such as images, stylesheets and scripts. It is good practice to give them a max-age far into the future (e.g., 365 days).
- What if the site owner decides to update the design? How can she invalidate the cached design elements in the users' browsers?
  - Solution: Include a fingerprint in the resource name, for example `style.3da37df.css`. When the resource changes, its name changes as well.



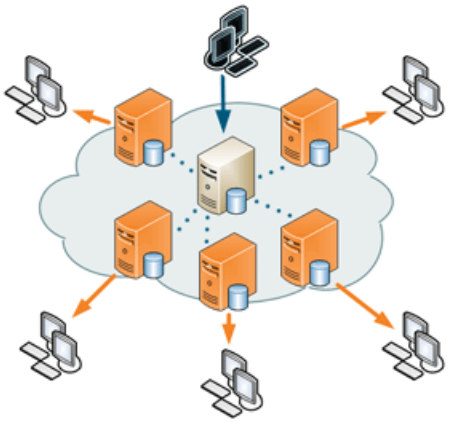
Web caching

Content distribution and caching as a service: Content Delivery Networks (CDN)

- A **Content Delivery Network** (CDN) is a collection of web caches that are geographically dispersed to deliver content to users more efficiently. It offers content distribution and caching as a service to anybody who operates a web site.
  - A CDN aims to have a cache server as close as possible to the user.
  - A good CDN has agreements with many local network providers (e.g., Swisscom, Cablecom) to operate a cache server in their network. Win-win for network provider and CDN.
- Terminology:
  - **Origin server**: The web server operated by the customer.
  - **Edge server**: A web cache operated by the CDN



One central web server



Content Delivery Network

Web caching

Content distribution and caching as a service: Content Delivery Networks (CDN)

- Advantages
  - For the user: Shorter network distance to the cached content: lower round-trip times, higher bandwidth → better performance
  - For the local network provider: Less load on the network, less peering traffic
  - For the customer: Less load on the server, happier customers
- When a user requests a page the CDN's **mapping system** selects an appropriate edge server.
  - For example choose the server with the fewest network hops or the server with the quickest response time.
  - Each CDN has its own mapping system and its mechanisms are not always published.
  - Ideally the mapping system is integrated into DNS and is transparent to the user and the operator of the web site.
  - Sometimes the inner workings of the mapping system can be guessed by examining the responses to a DNS request (see following page).

Web caching

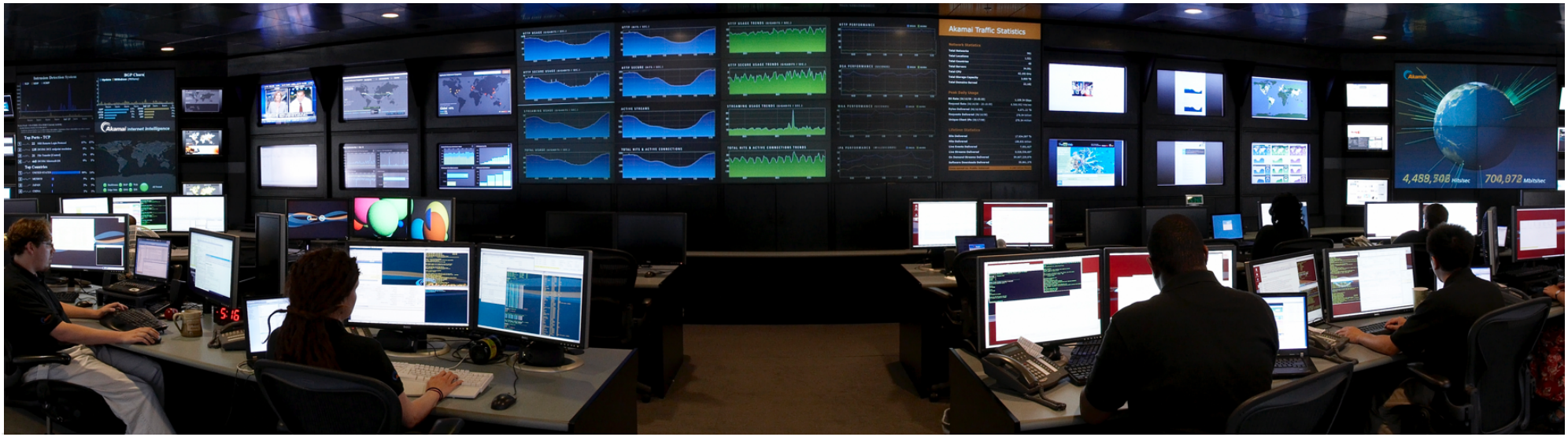
Content distribution and caching as a service: Content Delivery Networks (CDN) — Mapping system

```
$ dig images.apple.com

; <<>> DiG 9.8.3-P1 <<>> images.apple.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41338
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;images.apple.com.                      IN   A

;; ANSWER SECTION:
images.apple.com.                 318   IN   CNAME images.apple.com.edgesuite.net.
images.apple.com.edgesuite.net. 18318   IN   CNAME images.apple.com.edgesuite.net.globalredir.akadns.net.
images.apple.com.edgesuite.net.globalredir.akadns.net. 319 IN CNAME a199.dscgi3.akamai.net.
a199.dscgi3.akamai.net.            20   IN   A     195.176.255.165
a199.dscgi3.akamai.net.            20   IN   A     195.176.255.160

;; Query time: 32 msec
;; SERVER: 10.192.22.5#53(10.192.22.5)
;; WHEN: Tue Oct 21 15:28:19 2014
;; MSG SIZE  rcvd: 207
```

Web caching

Caching as a service — Akamai

- First of the big Content Delivery Networks, founded in 1998
- Today biggest CDN worldwide
  - 150'000 servers
  - in 1'800 locations
  - in 1'200 networks
  - in 92 countries
- Delivers between 15-30% of all web traffic



Web caching

Content Delivery Network

- Akamai real-time monitor of global web traffic on October 21, 2014 12:03 CEST
  - The day before Apple released iOS 8.1
- See http://www.akamai.com/html/technology/dataviz1.html