university of groningen

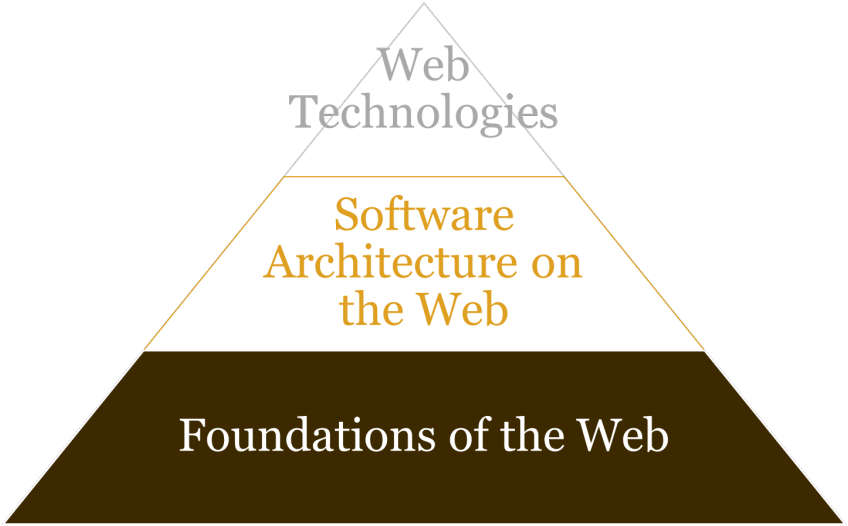
Web Engineering (WBCS008-05)

Set 2: Foundations of the Web

Vasilios Andrikopoulos
v.andrikopoulos@rug.nl

Outline

- Internet in a nutshell
- URIs
- HTTP
  - Basics
  - Caching
  - Authentication
- Content delivery



university of groningen

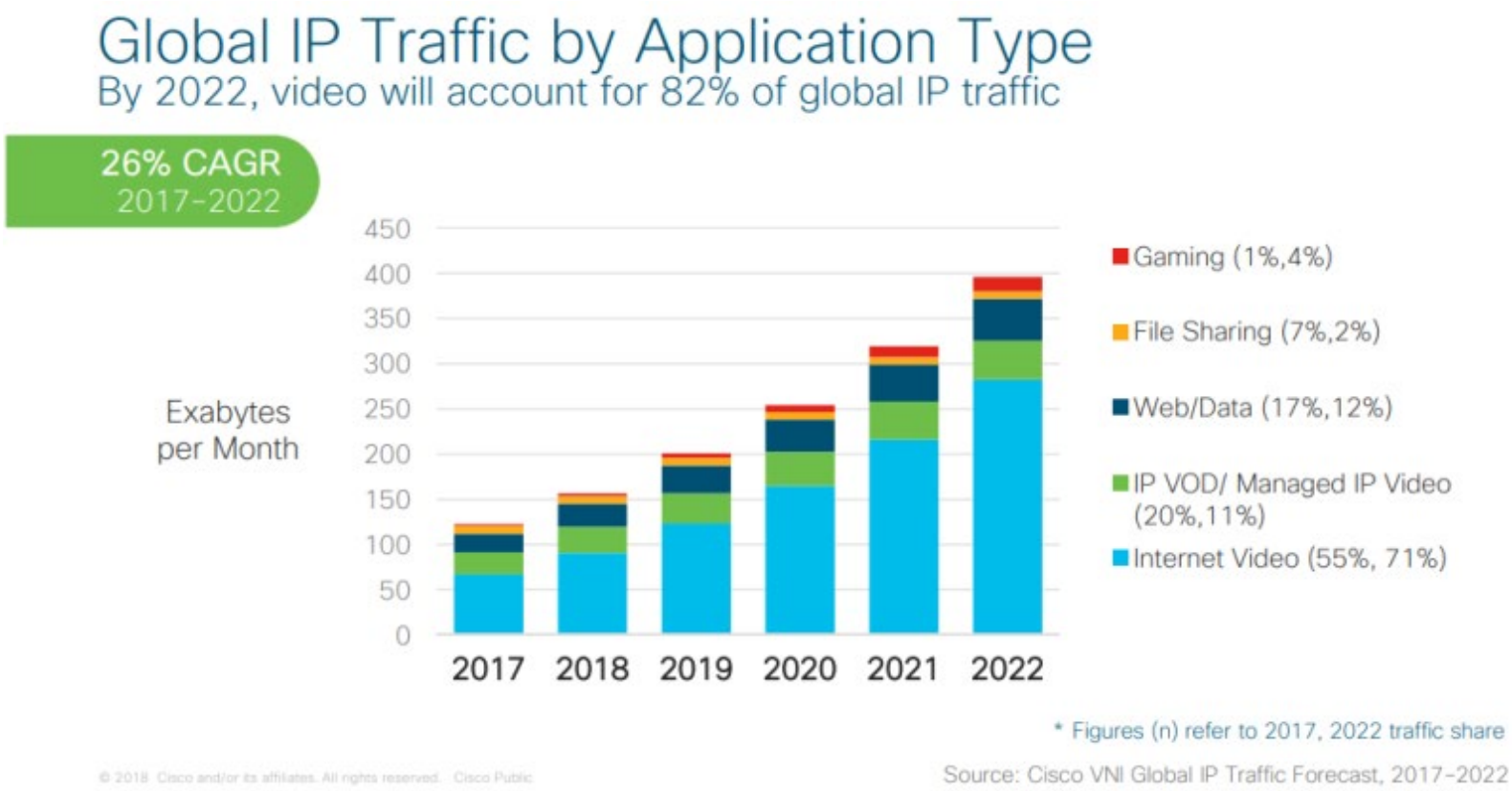
What is the Web? [from Set 1]

- World Wide Web: an information system for the retrieval of resources that are
  - **Identified by Uniform Resource Locators (URLs)**
  - **Connected to each to other by (hyper)links**
  - **Accessible over the Internet**
  - Resources: any type of (hyper)media but mainly hypertext

The Internet

university of groningen

The Web ≠ The Internet



The Internet

- Not an actual network, but a collection of heterogeneous networks
- Organic evolution of predecessors
  - ARPANET for US DoD purposes in the late 60s/70s
  - CS/NSF/ANSNET in the early 80s
- Internet from the late 80s on

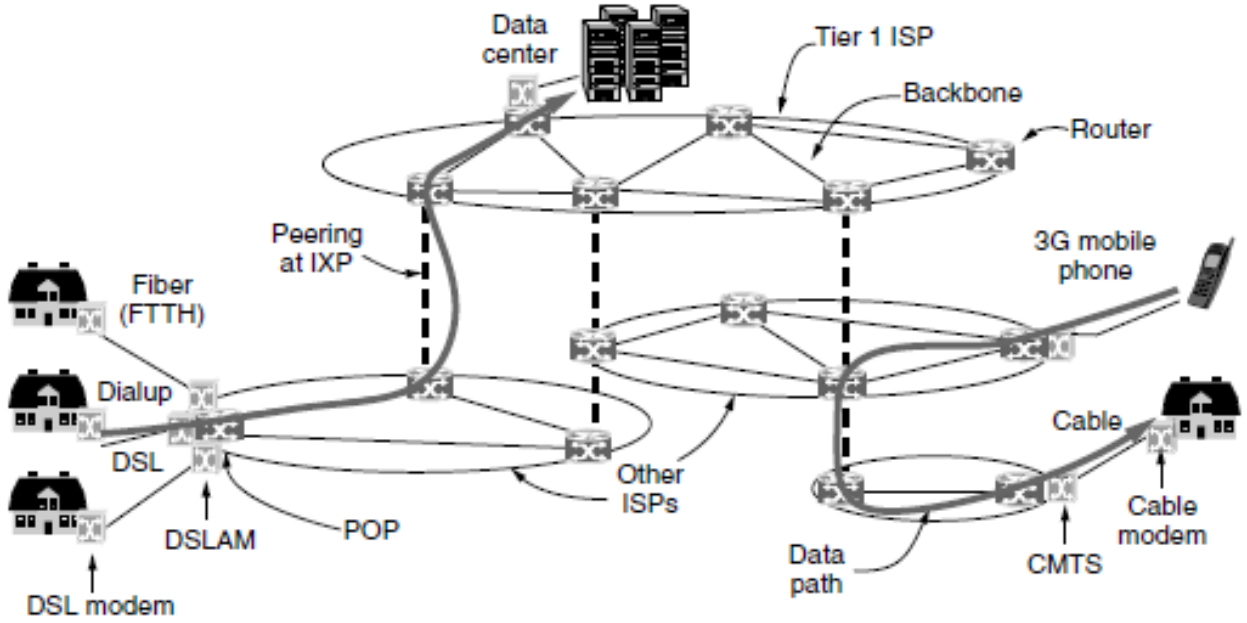
- Check out this article/map collection about the history of the Internet for more

university of groningen

The TCP/IP Model

Layer	Role	Protocols
Application	Contains all high-level protocols	HTTP, SMTP, DNS, …
Transport	Allows sources and targets to converse	TCP, UDP
Internet	Allows hosts to inject packets in any network to travel across them	IP, ICMP
Link	Interface between hosts and transmission links	DSL, SONET, 802.11, Ethernet

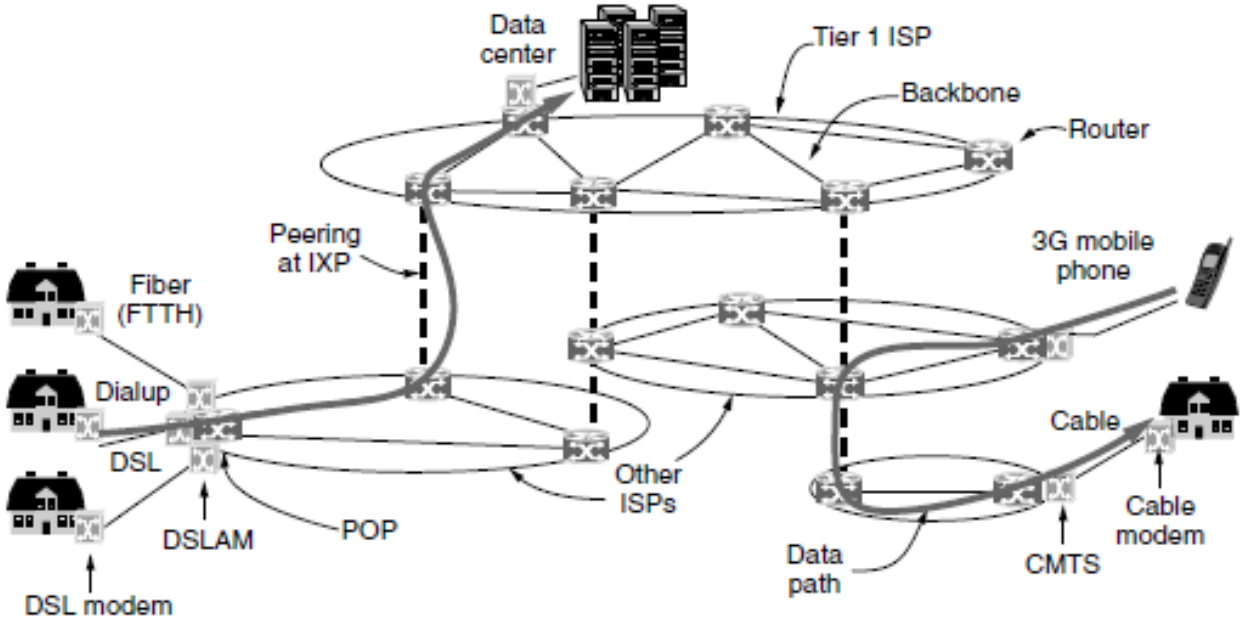
Internet architecture in a nutshell



- Internet Service Providers (**ISP**s) offering connectivity
- User-perceived bandwidth limited by the "last mile" of network (analog or digital) but ISP networks fully digital
- Points of Presence (**POP**s): location where customer packets enter the ISP network

university of groningen

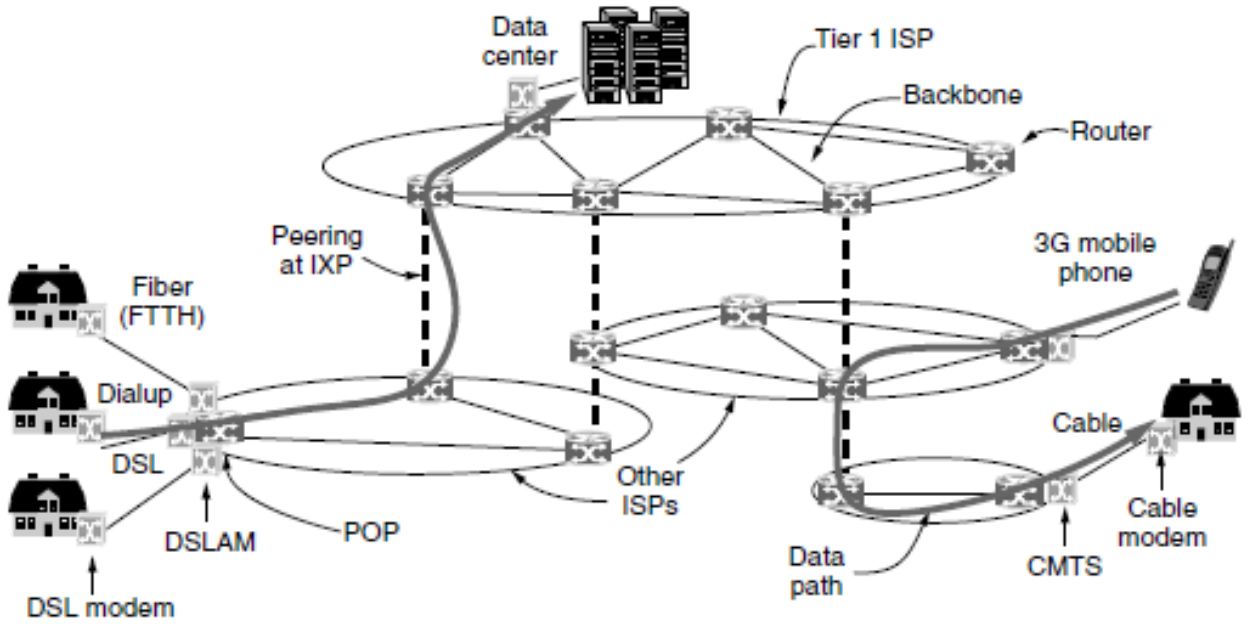
Internet architecture in a nutshell



› Internet eXchange Points (IXPs) **peer**ing ISP networks
  - See [IXP map]

university of groningen

Internet architecture in a nutshell



- ISPs may pay for **transit** through another network
- Packet path depends on the peering choices of the ISPs
- Tier 1 ISPs forming the Internet backbone

URIs

There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors.

Attributed to Phil Karlton (in its original form) and Leon Barmbick (for the off-by-one addition)

See thread

Resource Identification

- Fundamental questions for any hypertext/media system
  - How to identify individual entities (documents, people, data sets, etc.)?
  - Where are they located/how to access them?

- Early systems (e.g. HyperCard) identified documents only in the scope of a single system

- The Web uses Uniform Resource Identifiers (URIs) as global identifiers

university of groningen

Uniform Resource Identifiers

- **URI**: A reference identifying an abstract or physical resource
  - Form: **`<scheme>:[//<authority>]<path>[?<query>]`**
  - Can be a URL(ocator), URN(ame), or both

- In general case very simple
  - e.g. `postcode:9747AG`

- In many schemes the path is hierarchical
  - e.g. `<a href="../pics/">` where / has specific semantics

- Query components for additional information
  - e.g. …/`projects?page=3`

university of groningen

URI Schemes

- **URL**: The subset of URIs that identifies resources by their primary access mechanism (e.g. network location)
  - Form =

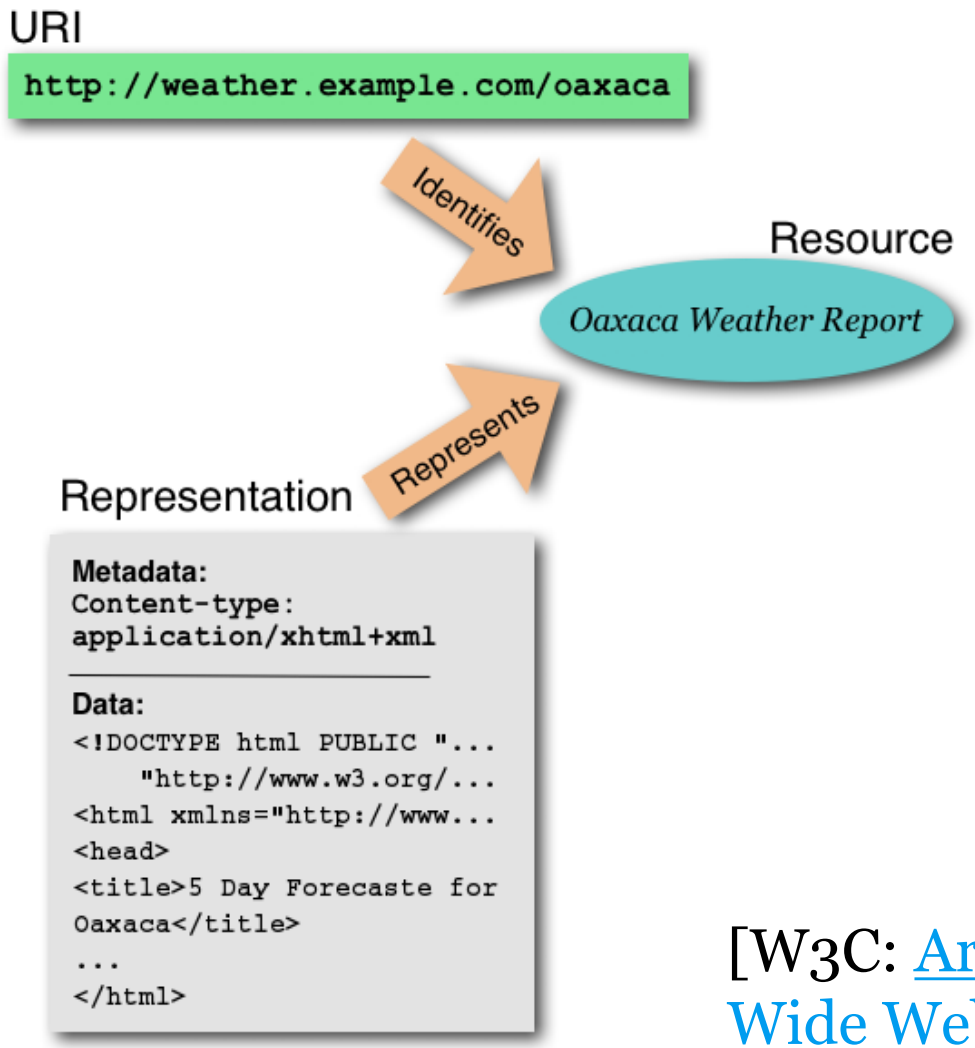
```
transport://user:password@host:port/path[?search][#fragmentid]
```

  - Is a physical address of a resource

- **URN**: The subset of URIs that uniquely identifies a resource independent of its primary storage location
  - Identifier remains stable even if the resource is moved to a different physical location
  - Is a logical address of a resource

university of groningen

URIs, Resources & Representations



[W3C: Architecture of the World Wide Web, Volume One]

HTTP
Part 1: Basic Concepts

HTTP in a nutshell

- **HyperText Transfer Protocol (HTTP)** as a simple request/response protocol
  - Based/relying on TCP (default port: 80)
  - Exchanged messages consist of header and MIME-like body

- HTTP is **reliable** and **connection oriented**
  - **No state** is maintained between request/response pairs, i.e. no references to past request/response interactions are possible
  - **Connections** can however persist

- Three major versions: 1.0, 1.1, 2

university of groningen

Roles In HTTP

- **Client**, or User Agent: Program that submits a request
- **Server**: Program that processes a request and returns a response (if applicable)
- That is, implementing the client-server architectural pattern for Distributed Systems see for example

Roles In HTTP (cont.)

- **Origin Server**: Server where a given resource resides (or is to be created)

- **Proxy**: A program that can act on behalf of an origin server
  - Client sends request explicitly to proxy, not to origin server
  - Proxy forwards request to origin server if it cannot be processed locally
  - Proxy may rewrite all or part of the message before passing on (“refresh”)
  - Many different types of proxies (see Web proxies)

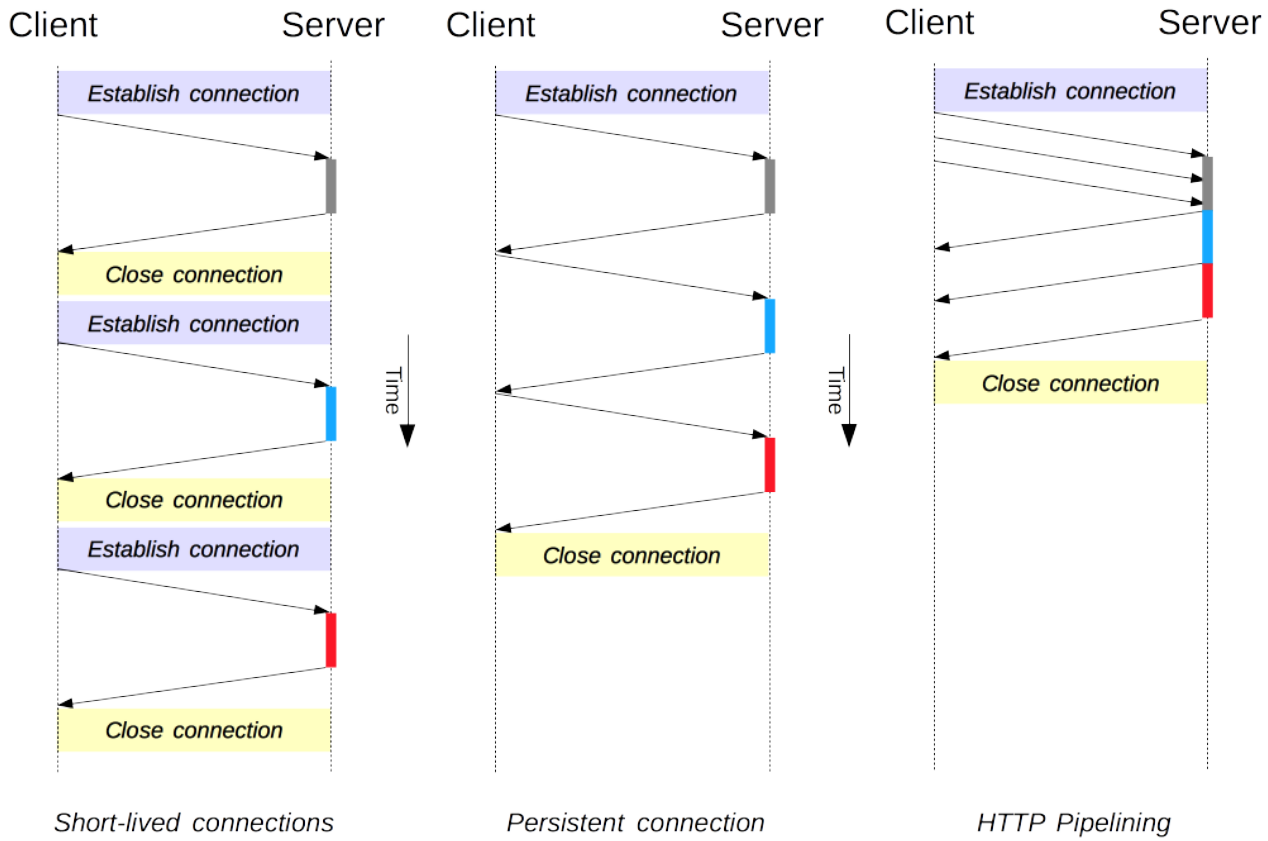
Roles In HTTP (cont.)

- **Gateway**: A server that acts as intermediary for some other server
  - Unlike proxy, client interacts with gateway as if it were the origin server for the request
  - If necessary, gateway translates request

- **Tunnel**: Program that acts as a "blind" relay between two connections
  - Tunnel only routes messages, i.e. it does not peek at them
  - Used when communication really needs to pass through an intermediary e.g. a firewall

university of groningen

Connection Management

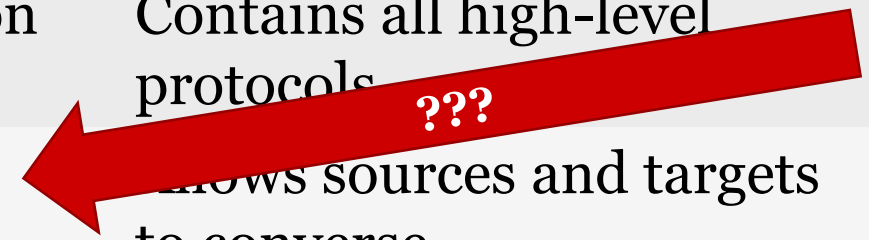


- HTTP/1.0: one transaction per connection
- HTTP/1.1: multiple transactions over a persistent connection with pipelined processing of requests
- HTTP/2: multiplexed requests over the same connection (plus a radically different transport model)

university of groningen

The TCP/IP Model x HTTP2

Layer	Role	Protocols
Application	Contains all high-level protocols	HTTP, SMTP, DNS, …
Transport	Allows sources and targets to converse	TCP, UDP
Internet	Allows hosts to inject packets in any network to travel across them	IP, ICMP
Link	Interface between hosts and transmission links	DSL, SONET, 802.11, Ethernet



Content negotiation

- Resources available in multiple representations, i.e. variants of the same resource

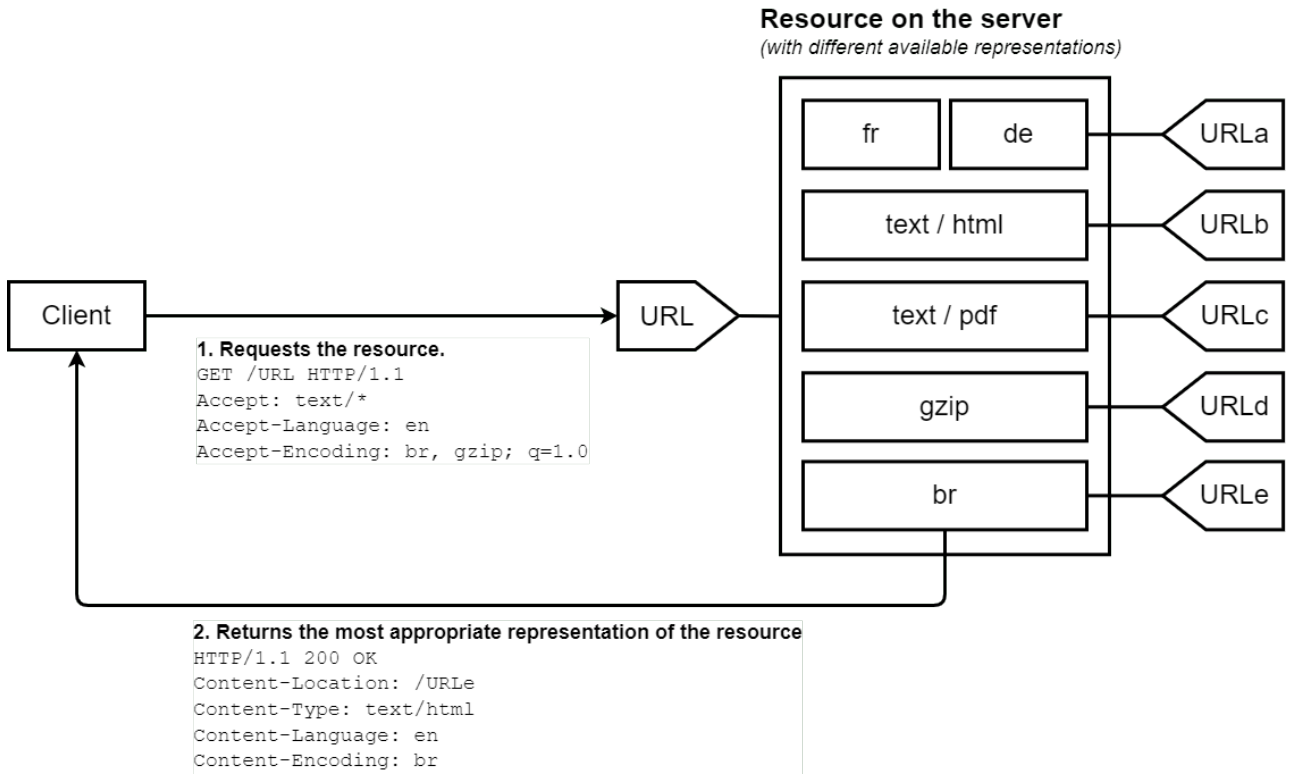
- Specific representation to be served to the client is determined by content negotiation mechanism

university of groningen

Types of content negotiation

› Server-driven

- Client includes request headers, and server makes “best effort” based on these headers
- Advantage: Avoids 2nd client request to select best variant
- Disadvantages: "Best" selection is affected by intended use of client (view, print,…) which cannot be described via headers

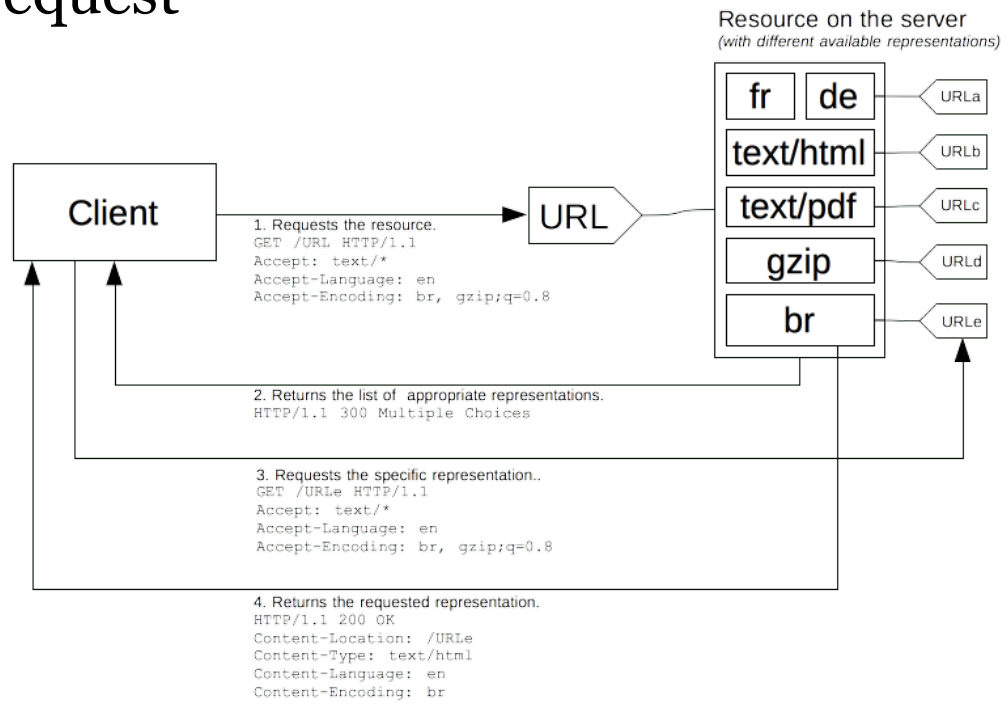


university of groningen

Types of content negotiation (cont.)

› Client/agent-driven

- Server responds with list of available variants, and client chooses best
- Advantage: Intended use can influence decision; blends well with scripting
- Disadvantage: At least 2 roundtrips for request



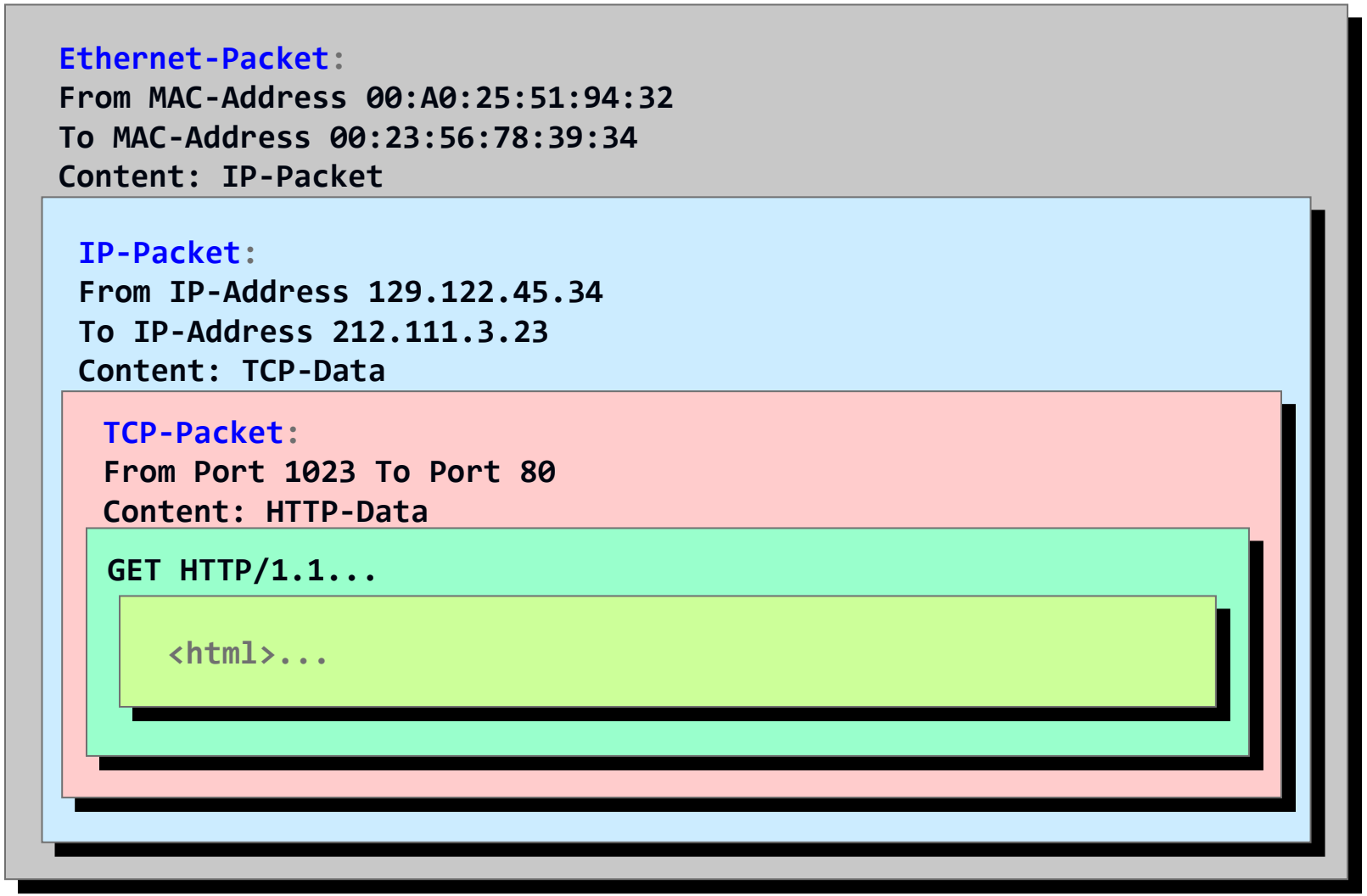
› Transparent

- Mixture of the above: No standard support in HTTP

HTTP
Part 2: Messages & Methods

university of groningen

Protocol nesting



university of groningen

HTTP message structure (in BNF)

```
HTTP-message =      request-line | status-line
                    message-header*     // 0..N
                    CRLF   //Carriage Return Line Feed
                    message-body?// 0..1
```

```
message-header = general-header | entity-header |
                      request-header | response-header
```

```
(general | entity | request | response)-header =
                 field-name ":" field-content CRLF
```

university of groningen

General & Entity Headers

Describe message itself, **not** the contained entity

```
general-header =
        Cache-Control | Connection | Date | Pragma |
        Trailer | Transfer-Encoding | Upgrade | Via
```

Information about the message body (i.e. the **entity**)

```
entity-header =
        Allow | Content-Encoding |
        Content-Language | Content-Lenght |
        Content-Location | Content-MD5 | Content-Range |
        Content-Type | ETag | Expires | Last-Modified
```

Sample General Headers

- `Cache-Control`: Directive for caching (see Caching)
- `Connection`:
  - Set to `Close` if client or server doesn't support persistent connection
  - Server will drop connection after response if `Close` is set
- `Transfer-Encoding`: Encoding applied to the message (not to the entity (=content encoding))
  - `Chunked`: Message body sent as sequence of chunks
  - `gzip`: Coded in gzip-format
- `Via`: If message is sent via gateways or proxies each intermediate node adds a corresponding `Via`-field
  - Including protocol used to receive the message
  - Allows to detect cyclic routing and protocol capabilities of intermediaries

Some Entity Headers

- `Content-Encoding`: How entity has been encoded (compression mechanism, e.g. zip, jpeg)
- `Content-MD5`: Fingerprint of complete message body
  - Allows detection of integrity
  - Allows to detect at server whether resource has been changed
- `Expires`: Specifies date-time-stamp of when response is outdated (unfortunately optional – see caching)
- `Last-modified`: Date-time-stamp when entity has been changed last time

university of groningen

HTTP Request Message

```
request =
 request-line
 (general-header | request-header | entity-header)*
 CRLF
 message-body?

request-line =
 method SP requestURI SP HTTP-Version CRLF
       //SP=space

HTTP-Version = "HTTP" "/" DIGIT ("." DIGIT)?

requestURI =
 "*" | absoluteURI | abs_path | authority
```

university of groningen

HTTP Request Message (cont.)

```
method =
 "OPTIONS" | "GET" | "HEAD"| "POST" | "PUT" |
 "DELETE" | "TRACE" | "CONNECT" | extension-method

request-header =
 Accept | Accept-Charset | Accept-Encoding |
 Accept-Language | Authorization | Expect | From |
 Host | If-Match | If-Modified-Since |
 If-None-Match | If-Range | If-Unmodified-Since |
 Max-Forwards | Proxy-Authorization | Range |
 Refer | TE | User
```

university of groningen

Sample Request Headers

- **`Host`**: Mandatory header that contains the URI or IP address and port of the server hosting the requested resource
- **`If-Match`**: Perform requested action only if all of the listed entity tags are the same for the stored resource
  - Often used for `PUT`s if an entity should only be overwritten if it has not been modified in the meantime
- **`If-Modified-Since`**: Used for conditional **`GET`**
- **`User-Agent`**: Type and version of the browser

Request URIs

- `*`
  - Request does not apply to particular resource but to server itself
  - E.g. OPTIONS * HTTP/1.1
- `absoluteURI`
  - Required if request is made to a proxy
  - The `Host` header is ignored in this case
- `abs_path`
  - Used to request a resource *directly* from the origin server
  - Client establishes direct TCP/IP connection to port 80 of origin server
  - The value of the `Host` header must be set accordingly
  - `abs_path` specifies resource on that server
- `authority`
  - Only used by `CONNECT` method for tunneling
  - Requested resource is always determined by examining both the request-URI and the `Host` header

Request methods (supportive)

- **`CONNECT`**: Used for tunneling
- **`HEAD`**: No message body but just header is requested
  - Used for validation of cached response messages
- **`OPTIONS`**: Determine properties of a server or resource
- **`TRACE`**: Only status code is responded; real interest is in sequence of `Via` headers
  - Each server along the path adds a `Via` entry!

university of groningen

Request methods (main)

- **`DELETE`**: Discard an entity
  - Origin server replies that it has the intent (nothing more!) to delete sometimes
- **`GET`**: Retrieve the entity addressed via request-URI
  - To be served by reading/processing a file or making a request
  - Partial **`GET`** via `Range` request header
- **`POST`**: Send data to server; substitute existing entity
- **`PUT`**: Create new entity or substitute an already existing one
  - **`PATCH`**: partial replace

university of groningen

Interlude

Note: This should read ‘safe’ not idempotent

Will Pearse @rombulow · 29 Apr 2018
You know how HTTP GET requests are meant to be idempotent? Well, do I have the story for you ... a while back I added WiFi control to our garage doors with little Wemos D1s.
90 2.5K 4.0K

Will Pearse @rombulow · 29 Apr 2018
The Wemos expose a simple web page with a link that says "Toggle". The endpoint for the link activates a relay, which is hooked up to the push-button on the garage door, which makes the door raise/lower/halt.
2 43 302

Will Pearse @rombulow · 29 Apr 2018
(This /toggle endpoint responds to GET requests. I threw the code together in minutes and was too lazy to spend another couple minutes figuring out POST. #regret)
3 34 319

Will Pearse @rombulow · 29 Apr 2018
Safari eventually figured out I used this /toggle page regularly, so added it to my favourites. Which are iCloud-synced between all my devices.
2 48 398

Will Pearse @rombulow · 29 Apr 2018
So every time I opened a new tab on my laptop, desktop, iPhone or iPad the garage door opened or closed. Late at night, early in the morning, or randomly throughout the day...
21 273 1.4K

Will Pearse
@rombulow
Follow

This, kids, is why GET requests should be idempotent.

Request properties

- **Safe** requests are non state-altering
  - User is not to be held accountable for the result of the interaction
    - E.g. without implicitly having to pay, or registering for advertisement, or…
  - Users do not commit themselves to anything by querying a resource or following a link
  - `GET`, `HEAD`, `OPTIONS` (and `TRACE`) are considered to be safe

Request properties (cont.)

- **Idempotent** requests are side-effects free
  - The side-effects of N > 0 identical requests is the same as for a single request
  - `GET`, `HEAD`, `OPTIONS`, `PUT`, `DELETE`, are idempotent (when implemented correctly)

- All safe methods are also idempotent, but not vice versa
  - `POST` (and `PATCH`) are neither

PUT vs POST

- Different semantics of `requestURI`
  - URI in a `POST` identifies the resource, i.e. the service that will handle the enclosed entity
  - URI in a `PUT` identifies the entity enclosed within the request

GET vs POST

- Use `GET` for:
  - Safe interactions
  - In practice, URIs are limited by some (older) servers to 256 or 4000 bytes, i.e. for longer URIs use POST with parameters in message body
- Use `POST` for:
  - State-altering interactions with observable results
  - The user be held accountable for the results of the interaction

university of groningen

HTTP Response Message

```
response =
 status-line
 (general-header | response-header | entity-header)*
 CRLF
 message-body*

status-line =
 HTTP-version SP status-code SP reason-phrase CRLF

response-header =
 Accept-Ranges | Age | Location |
 Proxy-Authenticate | Retry-After | Server |
 Vary | Warning | WWW-Authenticate
```

university of groningen

Sample Response Headers

- **`Age`**: Estimated age of entity
  - Caches or server with caches must fill this header field (see Caching)
- **`Server`**: Type and version of the Web server
- **`Accept-Ranges`**: Whether the server accepts accessing the resource in parts only (e.g. byte or line intervals)
- **`Retry-After`**: Estimated period after which the service is again up and running
- **`WWW-Authenticate`**: Name of authentication scheme and parameters to be used when status code **`401 - unauthorized`** has been responded (see Authentication)

Status Codes Categories

- 1xx: Informational
  Request received, continuing process
- 2xx: Success
  The action was successfully received, understood, and accepted
- 3xx: Redirection
  Further action must be taken in order to complete the request
- 4xx: Client Error
  The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error
  The server failed to fulfill an apparently valid request

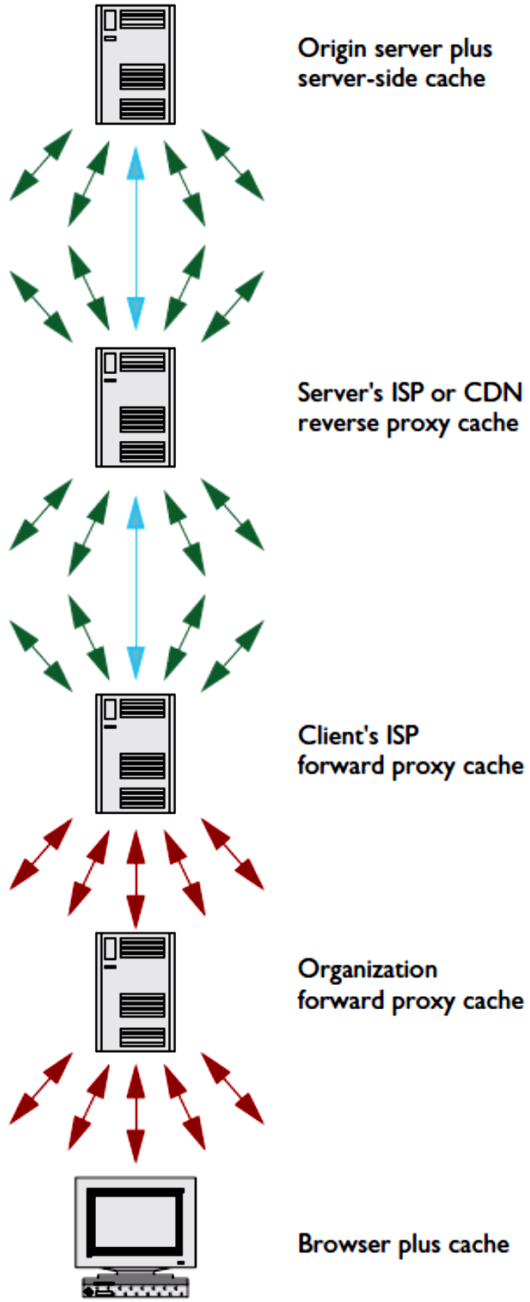
See also HTTP Cats

HTTP
Part 3: Caching

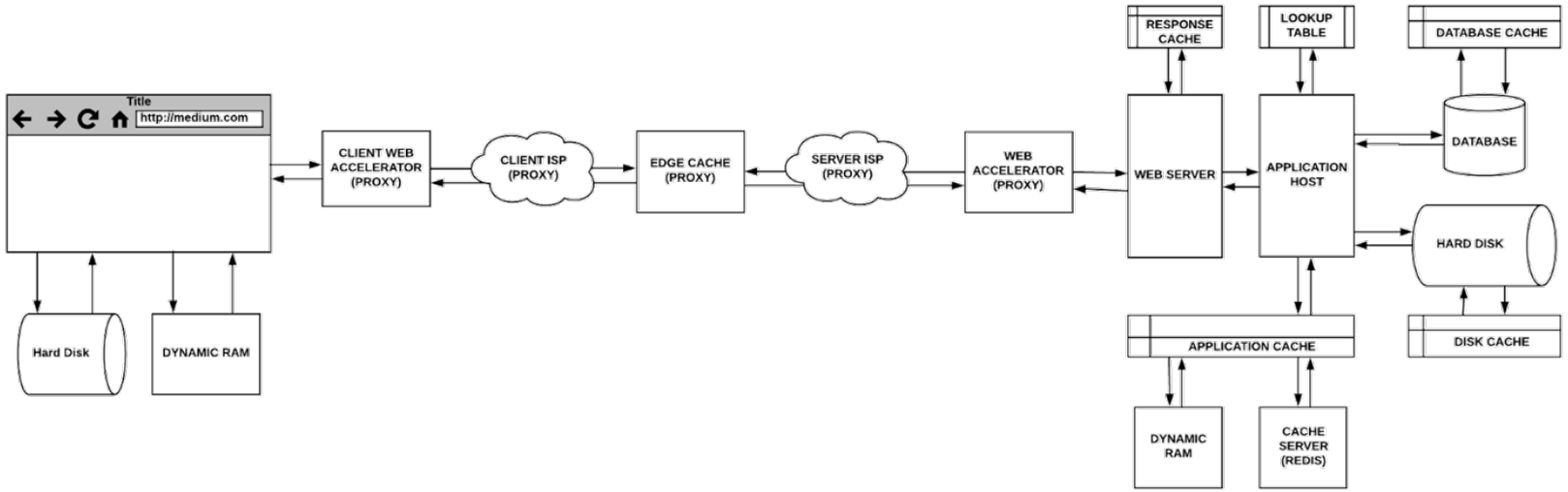
Caching in a nutshell

- Cache is storage for temporary persisting response messages
  - Purpose is to improve response time for requesting same response message

- Different topologies
  - Client-side (browser)
  - Server-side (origin)
  - Separate cache(s) on proxies

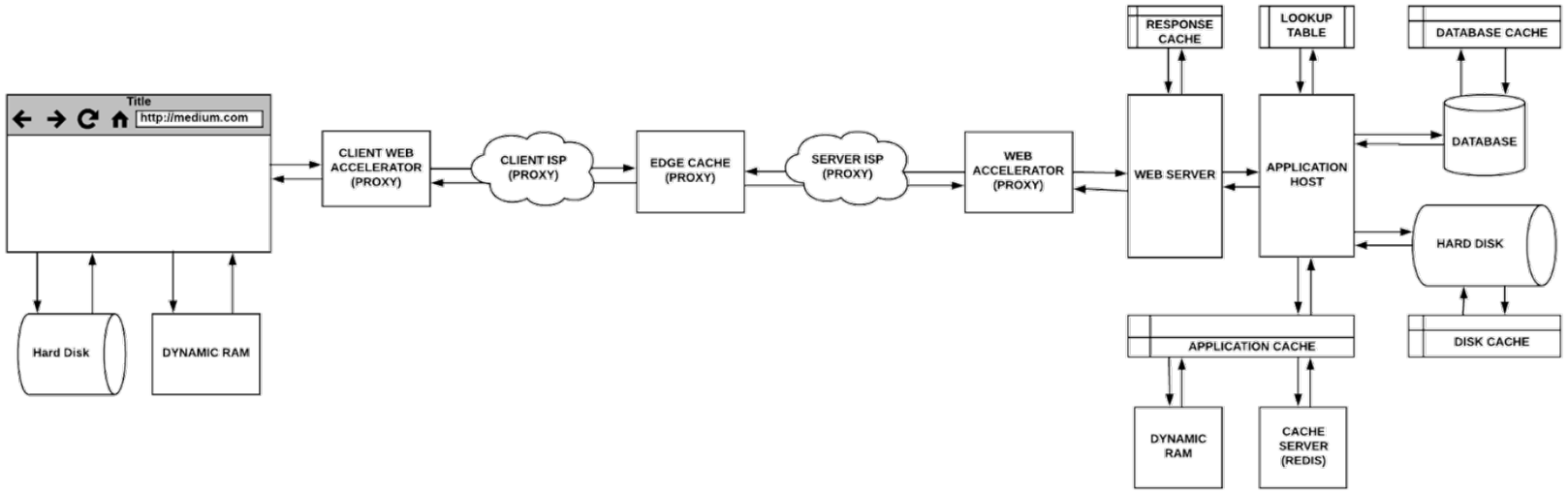


Web proxies & caching pipelines



- Forward proxy ≡ proxy: external facing stand-ins for the origin server(s) (N:1 relation to origin server)
- Reverse proxy: internal facing proxy, single entry point usually with load balancing between origin servers (1:N to origin servers)

Web proxies & caching pipelines (continued)



- Web accelerator: proxy with prefetching + compression, speeding up encryption, image manipulation, etc.
- Edge caching between network boundaries through CDNs (see CDN part)

Benefits of caching

- Reduces network bandwidth usage = cost reduction for both content providers and consumers
- Decreases perceived delays on the user side = increases user-perceived content value
- Removes load from origin server = reduces infrastructure/support costs on provider side, allows for faster serving of non-cached resources on client side

Challenges of caching

- If resource on origin server is changed, the copy stored in the cache is **outdated → cache invalidation**

- Thus, cache must implement mechanisms **to keep cached copies** ("secondary copies") **in sync** with the resource themselves ("primary copy") **→ cache consistency**

There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors.

Attributed to Phil Karlton (in its original form) and Leon Barmbick (for the off-by-one addition)

See thread

HTTP cache consistency

› HTTP mechanism to ensure cache consistency is based on
  - **timestamps for expiry** and
  - **validity checking**

› HTTP cache consistency based on the assumption that all parties depend on same time basis
  - Especially caches and origin servers are recommended to align their clocks based on worldwide precise time standard
  - Usage of Network Time Protocol (NTP) recommended

Semantic Transparency

- HTTP fundamental design principle:
  Semantic Transparency
  - Usage of cache (or any proxy server) must have no impact on client or origin server
  - Each request produces the same response as if the request would have been served directly by the origin server itself
  - Deviations only tolerable on explicit request from client or origin server, or a warning must be produced

university of groningen

Expiry Model for Cache Correctness

- Each response has a life-time
  - Before life-time is exceeded, response is called **fresh**
  - After life-time is exceeded, response is **outdated** or **stale**

- Life-time may be zero or infinite
  - Zero life-time responses are never fresh
  - Infinite life-time responses are never outdated

HTTP principle for caching

- Cache stores all fresh responses
- Cache monitors life-time of responses: "freshness control“
- Outdated responses must not be used for serving requests with some exceptions:
  - Client explicitly accepts outdated responses, or
  - It is impossible to contact origin server for getting valid copy of outdated response (`110 - response is stale`)

HTTP principle for caching (cont.)

- Each cache must always set response header field `Age`
  - Cache estimates age of cached response, i.e. how long response has not been refreshed
  - General header field `Date` can be used for this purpose – set by origin server when response message is created
  - Indicates to client that the response has been cached somewhere on its way to client

Server Determined Expiry

- Server sets life-time of response by setting:
  - Entity header field **`Expires`**
  - General header field **`Cache-Control`**
    - **Directive `max-age`**: life-time (of the response) in seconds
  - If both specified then **`max-age`** overwrites **`Expires`**

- Server can request that no copy of response is cached
  - E.g. if response is known to dynamically change at each request
  - Server simply specifies a past date as expiry date
  - **`Cache-Control`** header field is set to **`must-revalidate`**
    - Cache must never use cached response for serving requests

university of groningen

Cache Determined Expiry

- Each cache must always determine validity (freshness of cached response)
  - Response still fresh iff **`freshness-period > age`**
    - Freshness period may be based on **`max-age`** and **`Date`** (**`Date + max-age > current date`**)
  - Freshness period may also be based on difference of **`Expires`** and **`Date`** headers (**`Expires - Date > Age`**)

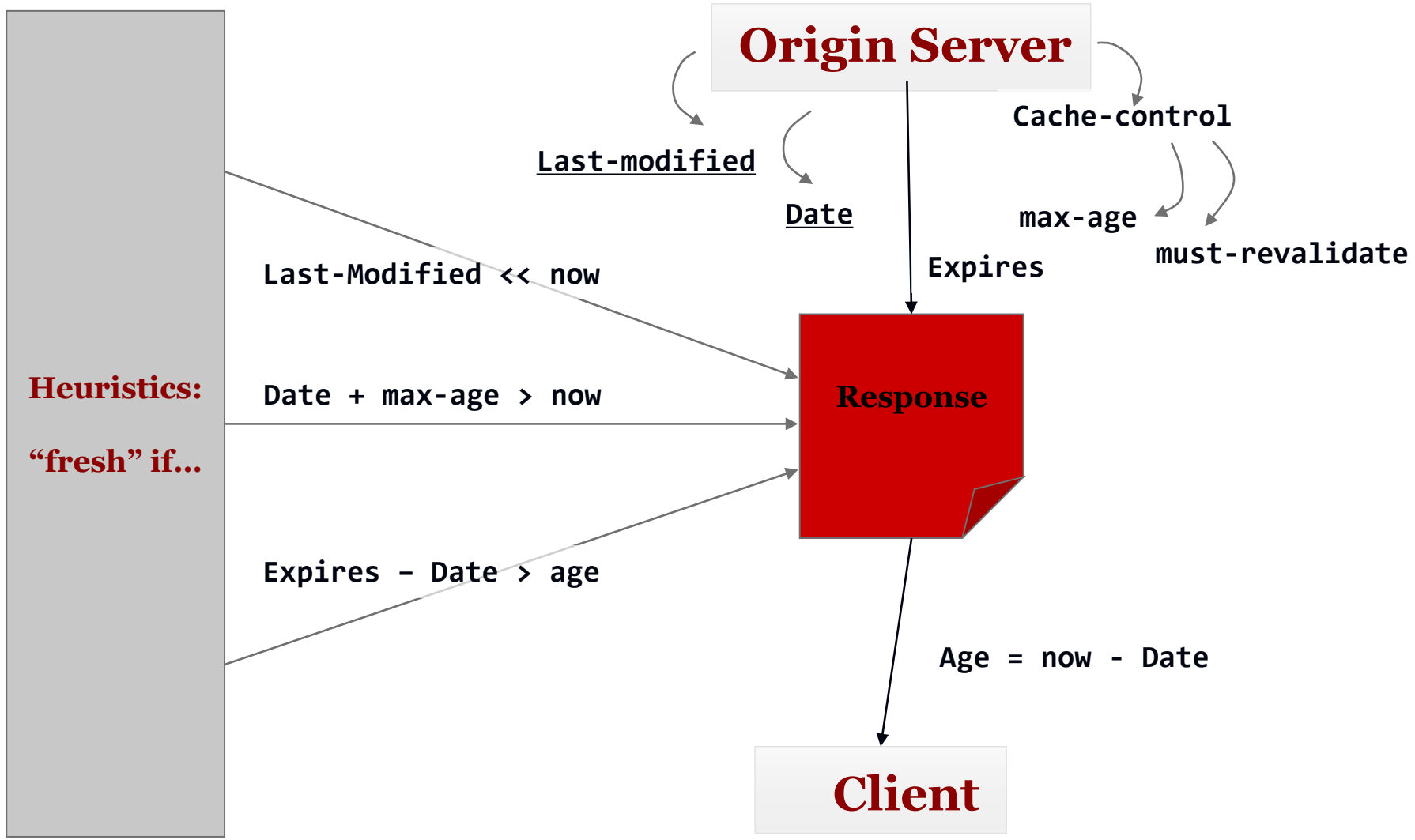
- But both **`Expires`** and **`Cache-control`** are actually **optional!** (as all headers)

Cache Determined Expiry

- In absence of explicitly defined expiry of responses, caches are allowed to determine validity *heuristically*
  - E.g. based on `Last-modified` header (mandatory) – if no change in recent past then probably no change in immediate future either
  - In general, semantic transparency **no longer guaranteed**!

→ More sophisticated HTTP Validation model through `ETags`

Cache Expiry In a Nutshell



HTTP
Part 4: Authentication

university of groningen

HTTP Authentication Framework

- Defined by RFC 7235 as a challenge & response scheme
- Authentication realm: a set of resources requiring authentication to be accessed
- Using the `WWW-Authenticate` response header for the challenge and the `Authorization` request header field for credentials submission:

```
WWW-Authenticate: <type> realm=<realm>
Authorization: <type> <credentials>
```

university of groningen

Proxy authentication

› Same scheme but with `Proxy-Authenticate` for the challenge and `Proxy-Authorization` for credentials (and status code `407` instead of `401` for unauthorized access)

```
Proxy-Authenticate: <type> realm=<realm>
Proxy-Authorization: <type> <credentials>
```

- Provides access to the contacted proxy server only, not origin server
- Credentials usually propagated across proxy servers in the same organization to give single entry scheme impression

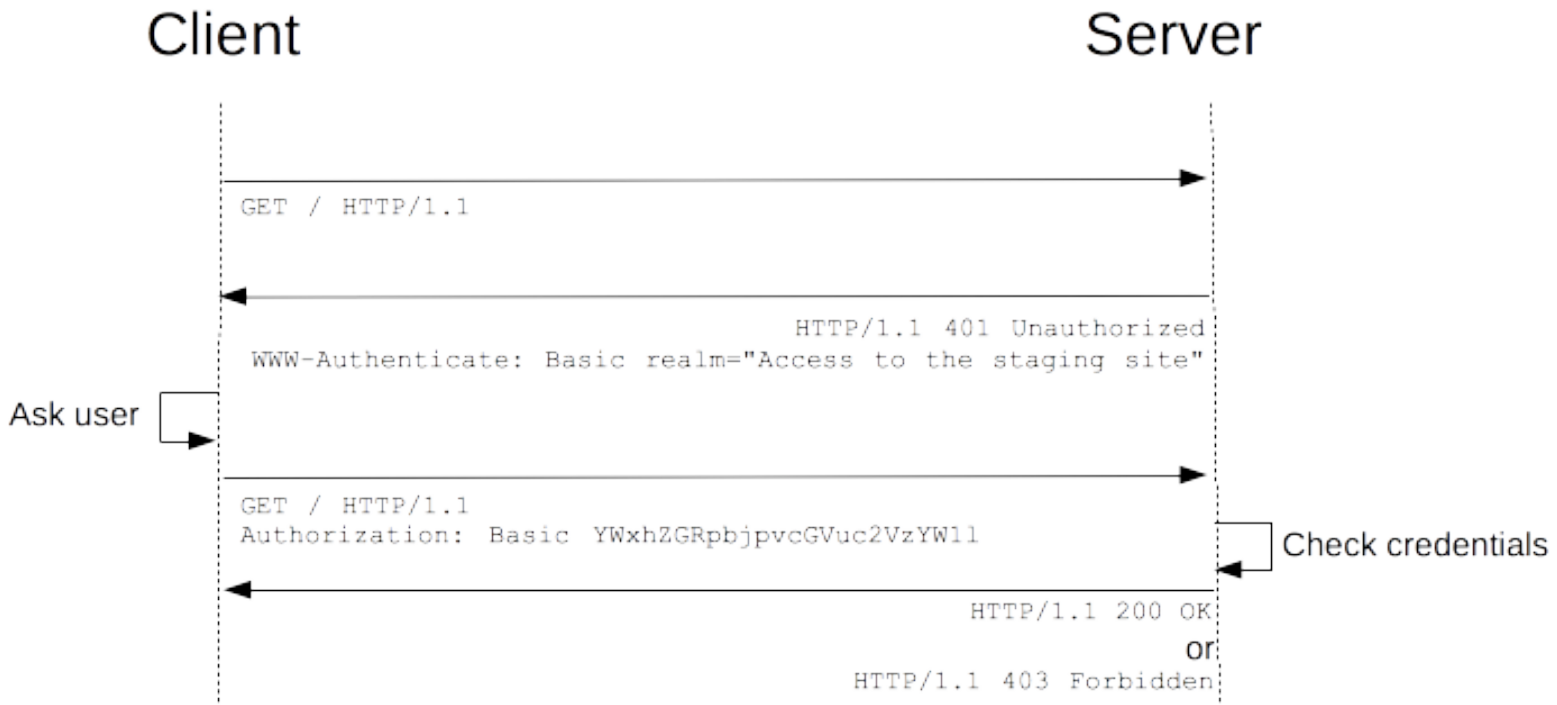
university of groningen

Authentication schemes

- Basic
  - (see following slides)
- Bearer
  - Based on OAuth 2.0
- Digest
  - Hashed username, password not in clear text
- For more in this list

university of groningen

Basic scheme



Basic scheme

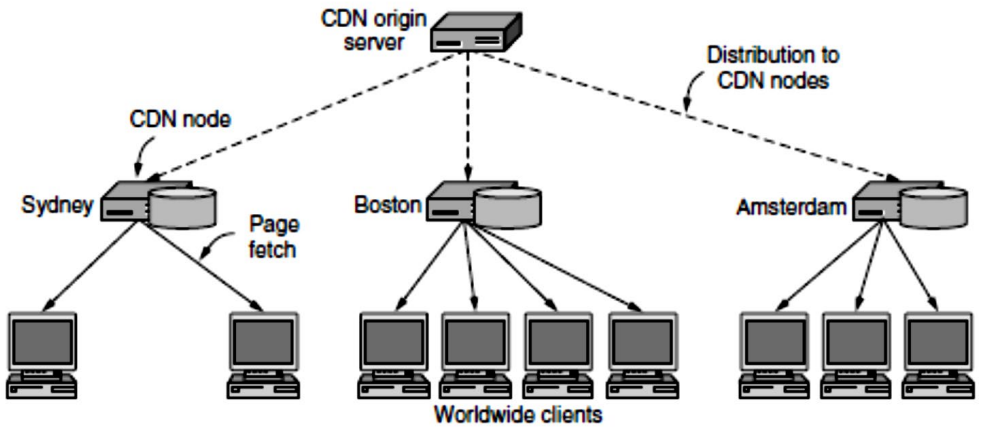
- Transmits credentials as user ID/password pairs in base64 encoding i.e. not encrypted
  - Effectively not secure (!) – credentials in plain text
  - HTTPS (HTTP over TLS) assumed to be used for security purposes

- HTTP clients (used to) allow credentials embedding in the URL (!)
  - `https://username:password@www.example.com/`
  - Deprecated and restricted by modern browsers

Content Delivery Networks

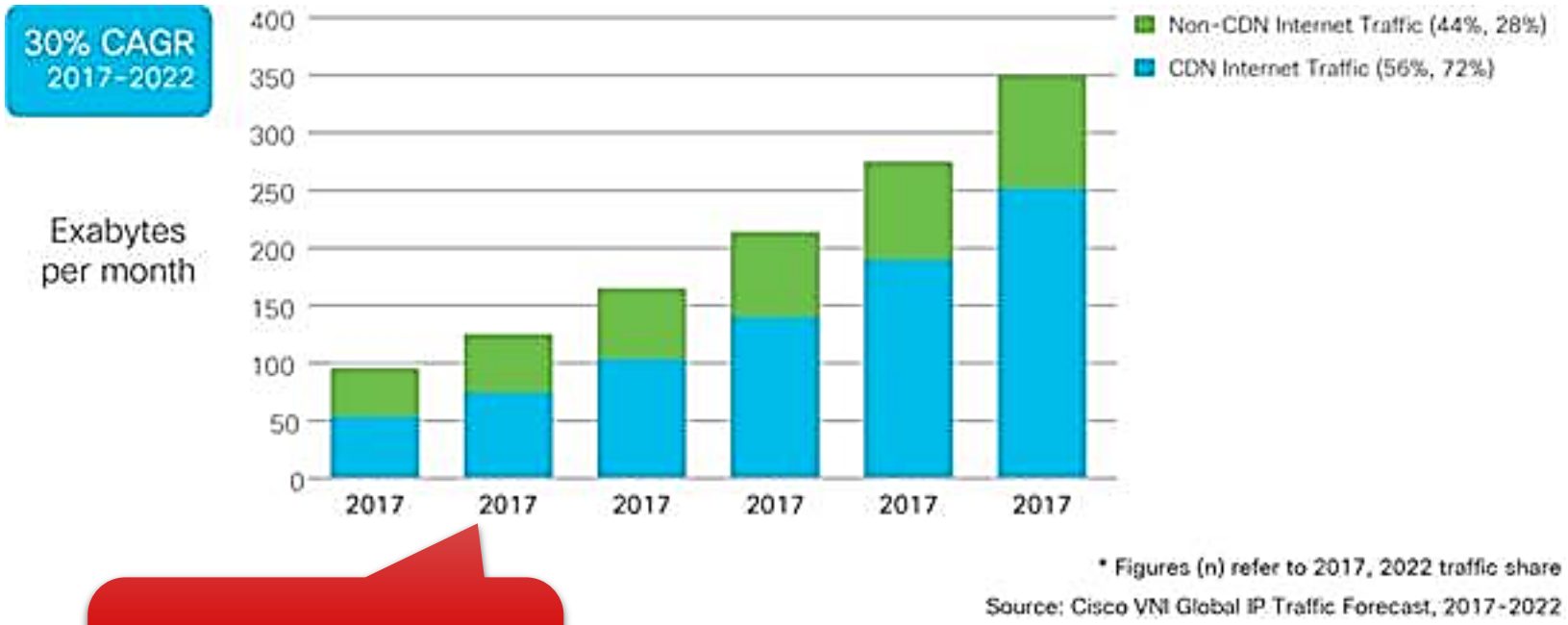
Content Delivery Networks

- "Inverse"/edge proxy
  - Provider puts a copy of a resource on CDN node
  - CDN copies it on multiple local nodes
  - Client's requests for it are directed to nearest node to use
- Most popular approach based on DNS redirection
  - CDN resolving DNS requests to closest nodes



university of groningen

CDN Traffic



This should read 2018, etc.

university of groningen

CDN use in practice

```
(function() {
      var preloadImage = document.createElement('img’);
      preloadImage.src = cdnURL + 'img/logo.png'; })();
```

Example from https://css-tricks.com/adding-a-cdn-to-your-website/

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
```

Example from https://getbootstrap.com/docs/4.3/getting-started/introduction/

CDN Benefits

- Reduced latency (less network hops)
- Scaling to demand
- Increased reliability
- Abstraction from data transfer
- Security against DDOS attacks

Source material

- Andrew Tanenbaum and David Wetherall, Computer Networks, 5th edition

- Mozilla Developer Network

Self-evaluation questions

- What do the acronyms ISP, POP, and IXP stand for and what is their purpose?
- How are the terms URI, URL, and URN defined, what is their purpose, and what is the relation between them?
- What is the difference between a proxy, a gateway, and a tunnel server?
- Under which conditions are HTTP requests safe/idempotent? Which HTTP methods are considered safe/idempotent?
- What are the differences between PUT and POST in terms of request URI semantics, and pragmatics (i.e. how they are to be used)?

Self-evaluation questions (cont.)

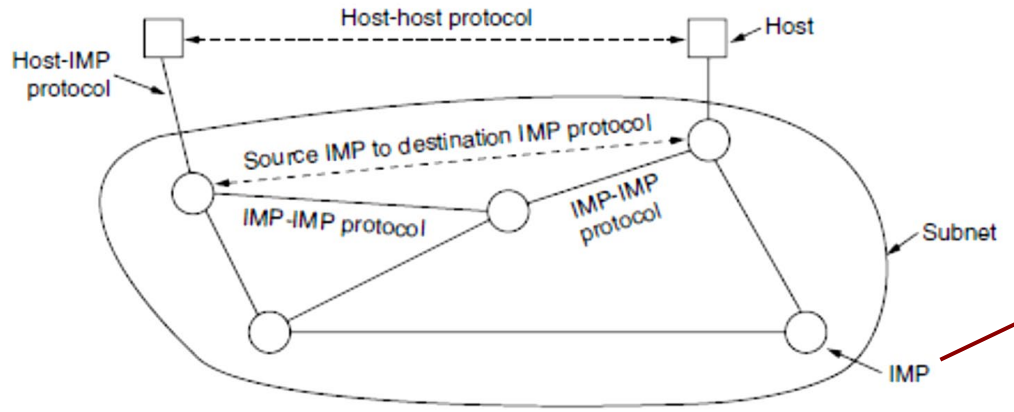
- Which proxy types are common on the Web, and what is their function?
- What are the benefits of caching?
- How is the principle of Semantic Transparency for HTTP defined? Under which conditions is it violated by the use of caches?
- How can a server stop a proxy or client from caching a response?
- How can a cache determine if it is fresh in the absence of an Expires header? If Cache-control is missing too?
- How does the Basic authentication scheme for HTTP work? Under which conditions should it be used?
- How do CDNs work and what benefits do they offer?

Next lecture

REST

Appendix:
a short history of the Internet

ARPANET



Interface Message Processor

- Distributed switching system for US DoD purposes in the 50s designed by RAND Corporation but rejected by AT&T
- ARPANET (1967) as a way of connecting computers in subnets
- Subnet tender implemented by a consulting firm (BBN) *without any* host software

university of groningen

ARPANET

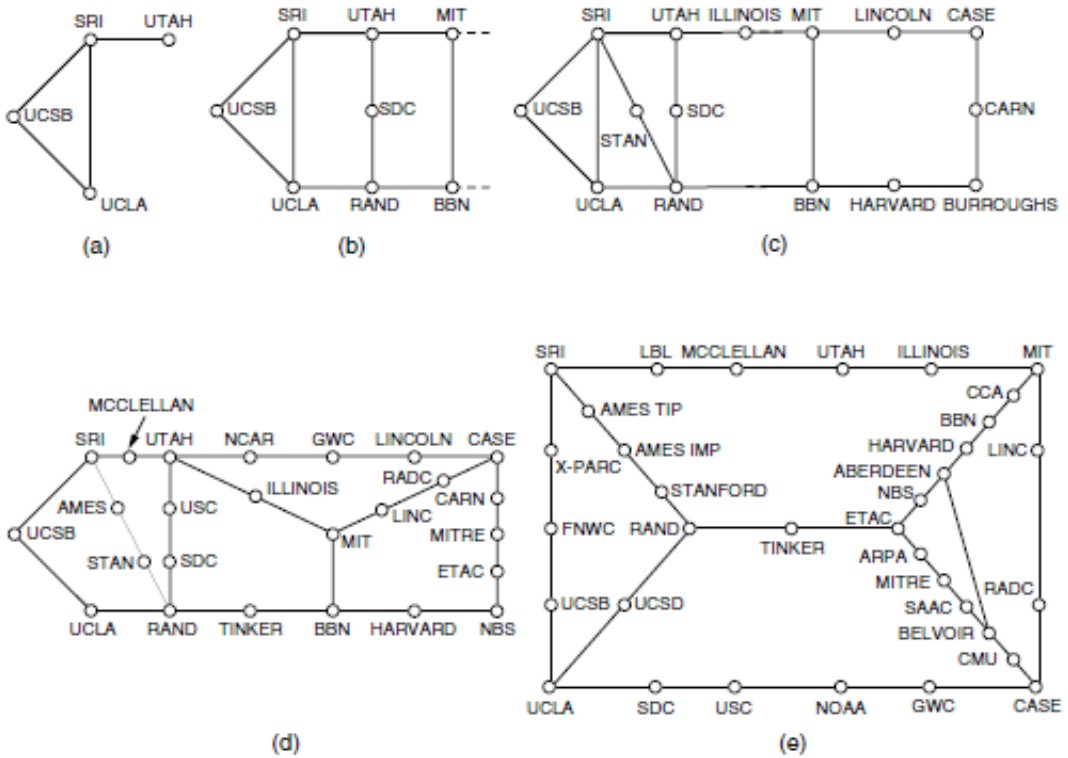


Figure 1-27. Growth of the ARPANET. (a) December 1969. (b) July 1970. (c) March 1971. (d) April 1972. (e) September 1972.

› TCP/IP model (1974) invented and adopted as the means to deal with internetwork communication

ARPANET successors: CSNET/NSFNET

- US National Science Foundation (NSF) funds CSNET for US Computer Science departments in 1981
  - Backbone network of supercomputing centers
  - IMPs replaced by fuzzballs talking natively TCP/IP
  - About 20 regional networks connected to the CSNET backbone resulting into NSFNET
  - Carnegie Mellon University acting as the bridge with ARPANET

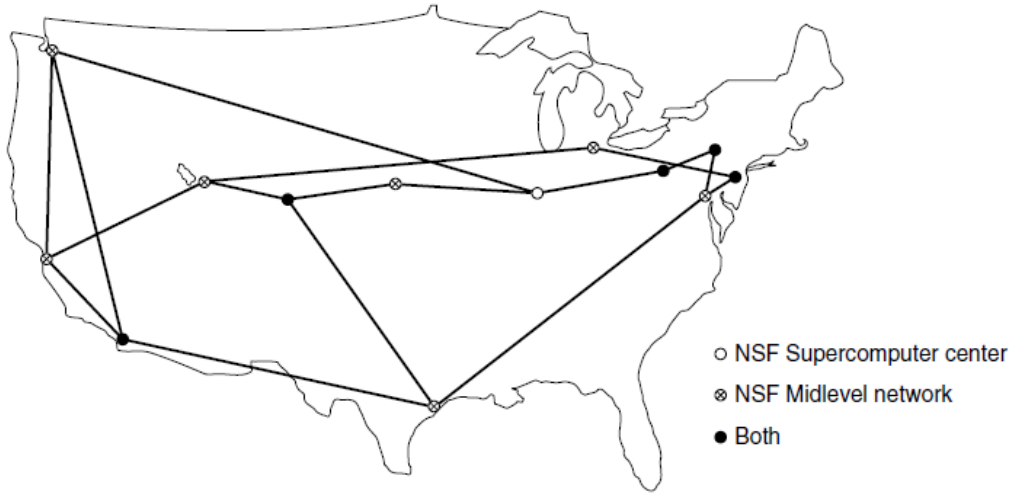


Figure 1-28. The NSFNET backbone in 1988.

ARPANET successors: ANSNET

- Advanced Networks and Services (ANS) non-profit set up for commercialization purposes
  - Took over and upgraded NSFNET into ANSNET
  - NSF already awarded contracts to four different network operates to establish Network Access Points (NAPs) across the country
  - Backbone service providers required to connect to all four NAPs