COMP6218: Content Caches

Prof Leslie Carr

Slashdot.org

- The **Slashdot effect**, also known as **slashdotting**, occurs when a popular website links to a smaller site, causing a massive increase in traffic…

Slashdotting

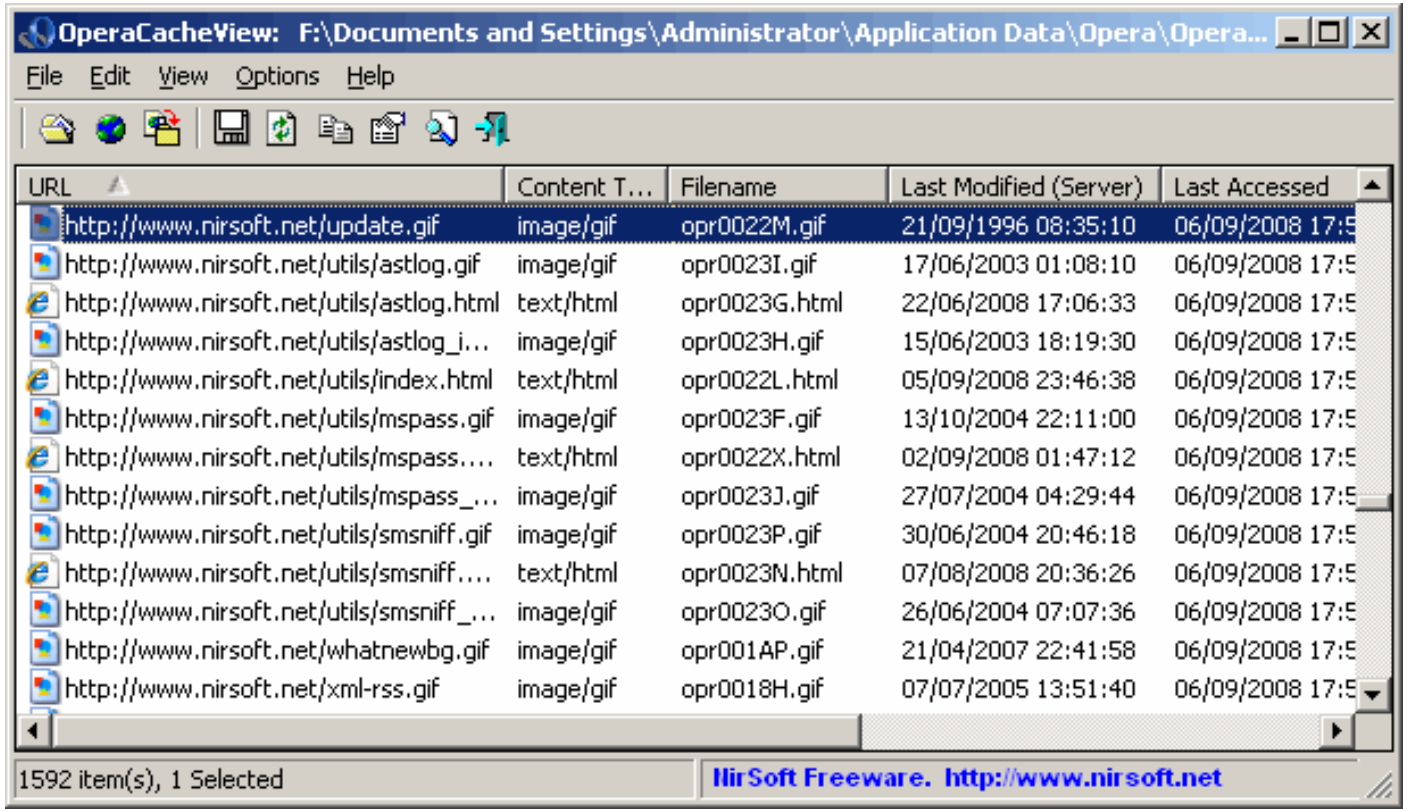
- This overloads the smaller site, causing it to slow down or even temporarily become unavailable.
- The name stems from the huge influx of web traffic that would result from the technology news site Slashdot linking to websites.
  - Somewhat like a DDOS effect
- This lecture is about how the Web solves the problem of too many viewers.

Cache

- The **temporary storage** of **frequently accessed data** stored for **rapid access**
- Original data is stored elsewhere – usually somewhere a long way away that is slower and more inconvenient to get access to
  - Reduces access time/latency for clients
  - Reduces bandwidth usage
  - Reduces load on a server

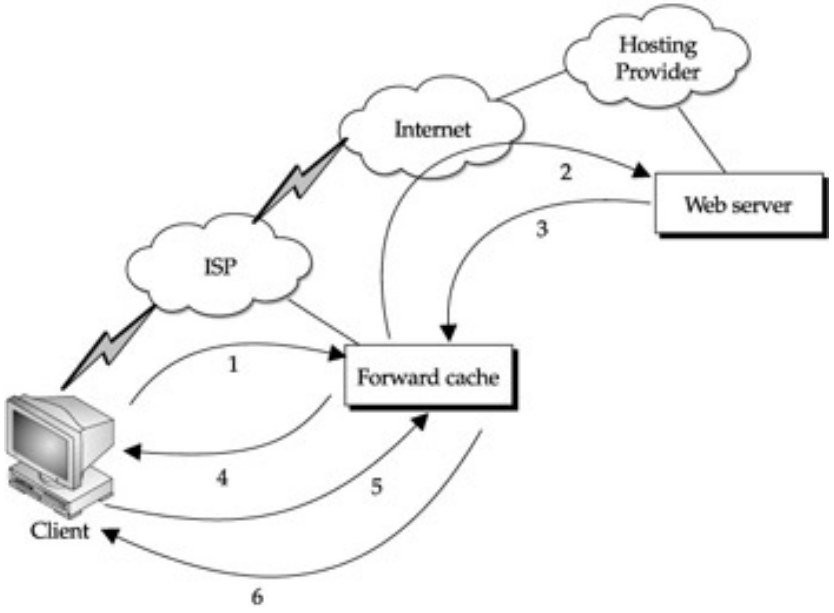
Browser cache

- Cache for a single user / application



Proxy cache

- Cache located close to the clients
  - e.g. University or Internet Service Provider
- Decreases external bandwidth usage
- Decreases network latency



Scale provides the advantage: many users within the ISP may all be asking for the same web pages

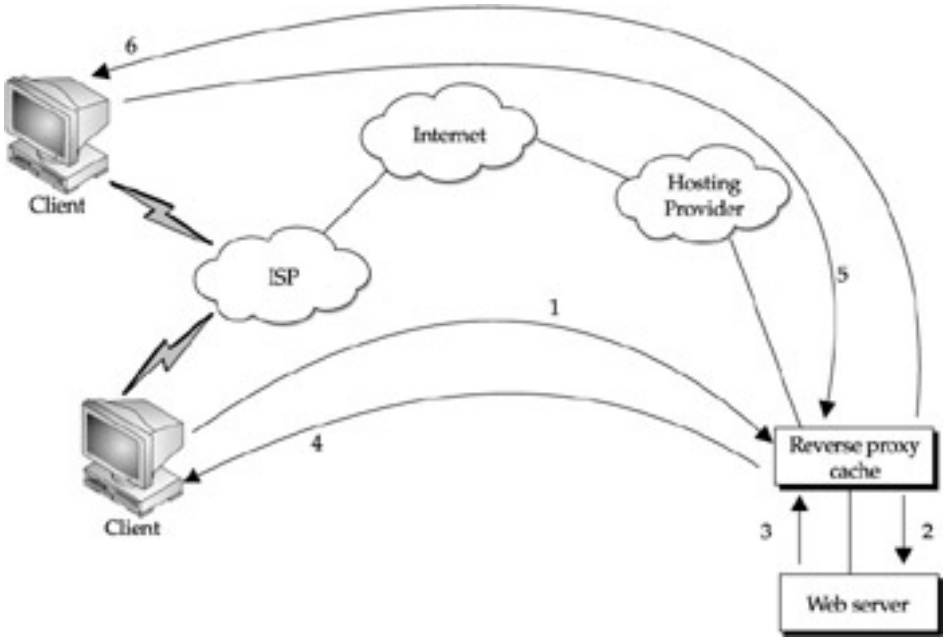
HTTP cache-control headers

- Freshness – how long the cached copy stays “fresh” without revisiting the origin server
- Validation – compare the cached copy to the origin document after it stops being “fresh”
- HTTP headers control browser and proxy caches

HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html

Reverse proxy cache

- Cache proxy located closer to the **origin web server**
- Usually deployed by a Web hosting ISP
- Decreases load on the Web service (database)



- *Several reverse proxy caches implemented together can form a* ***Content Delivery Network***

Content distribution networks

- *Business Requirement:* stream video content to hundreds of thousands of simultaneous users

- *Obvious Web solution:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

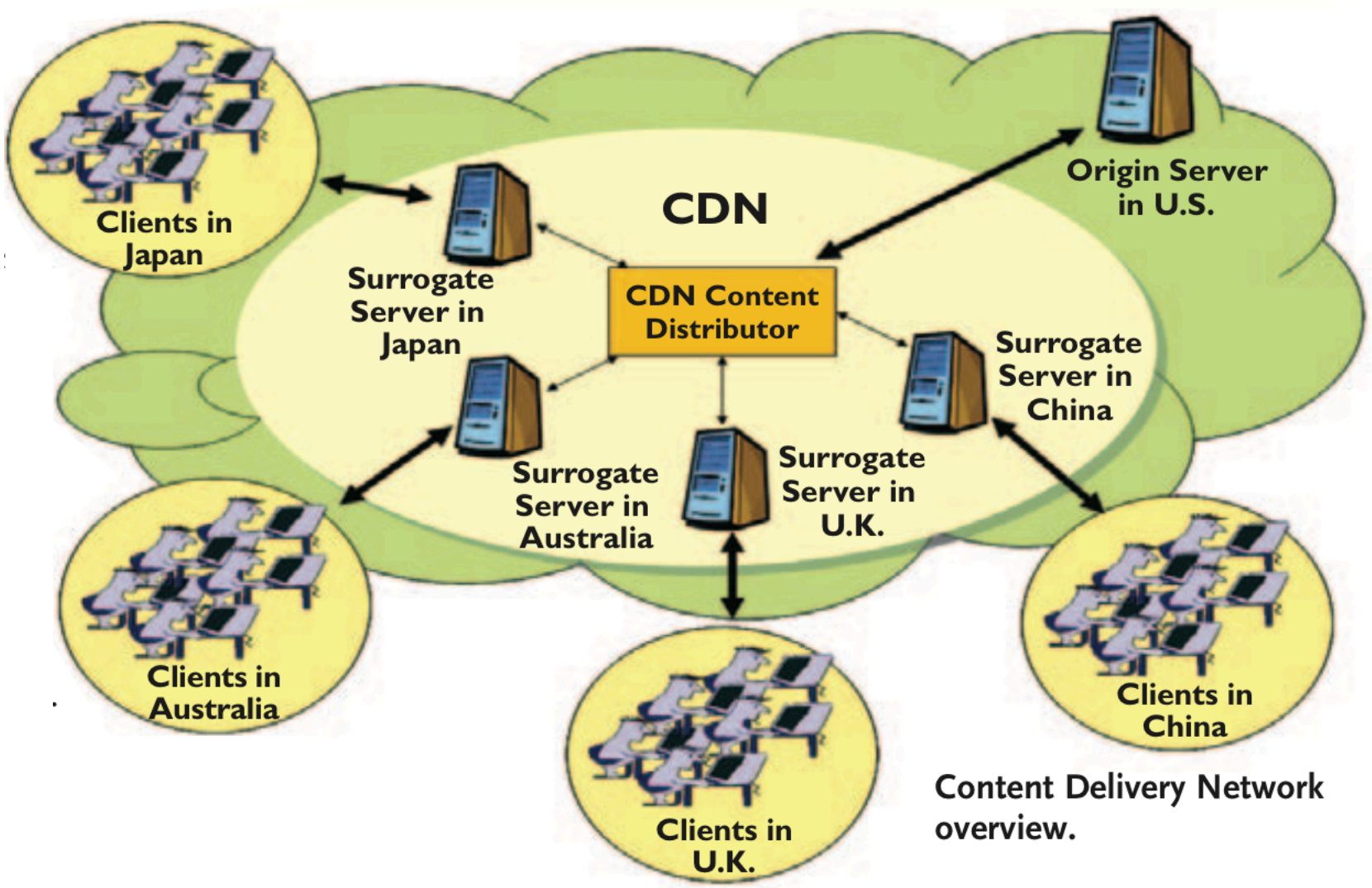
….this solution *doesn't work in practice*

Content distribution networks

- *Business Requirement:* stream video content to hundreds of thousands of simultaneous users

- *Working Web Solution:* store/serve many copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - used by Akamai, 1700 locations

Content Delivery Networks

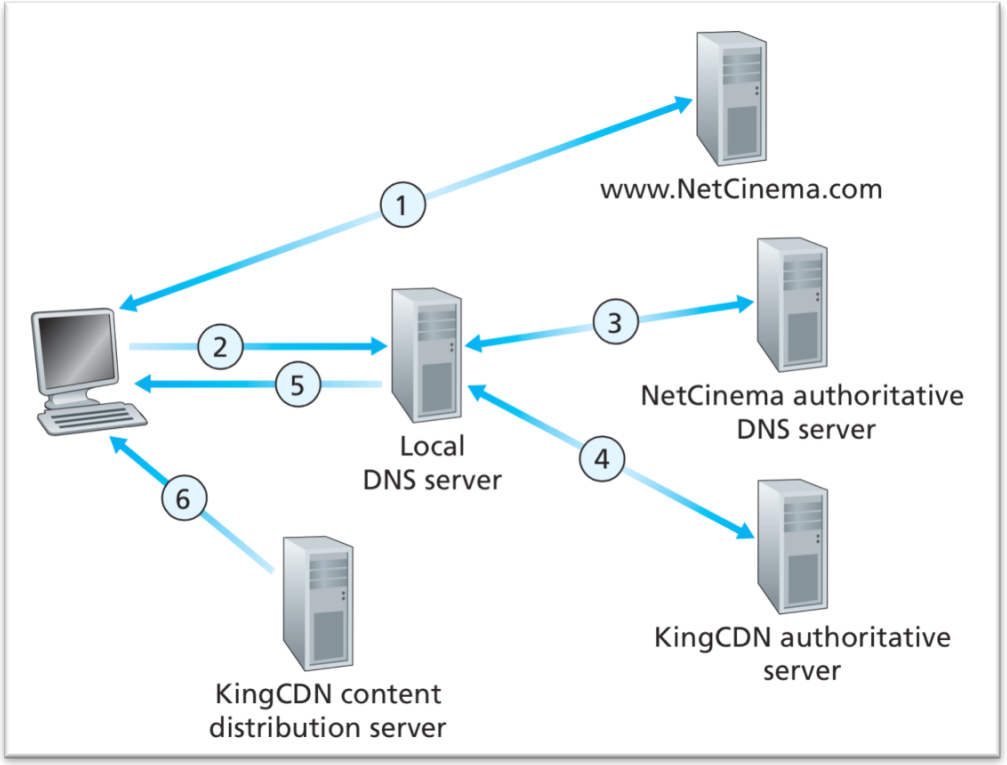


Content Delivery Network overview.

CDN: “simple” content access scenario

Bob (client) requests “Transformers 7” video from the NetCinema service http://netcinema.com/

- Link is to http://video.netcinema.com/6Y7B23V
- Video actually stored in CDN at http://KingCDN.com/NetC6y&B23V



Uses DNS creatively to decide which KingCDN distribution server to use

CDN cluster selection strategy

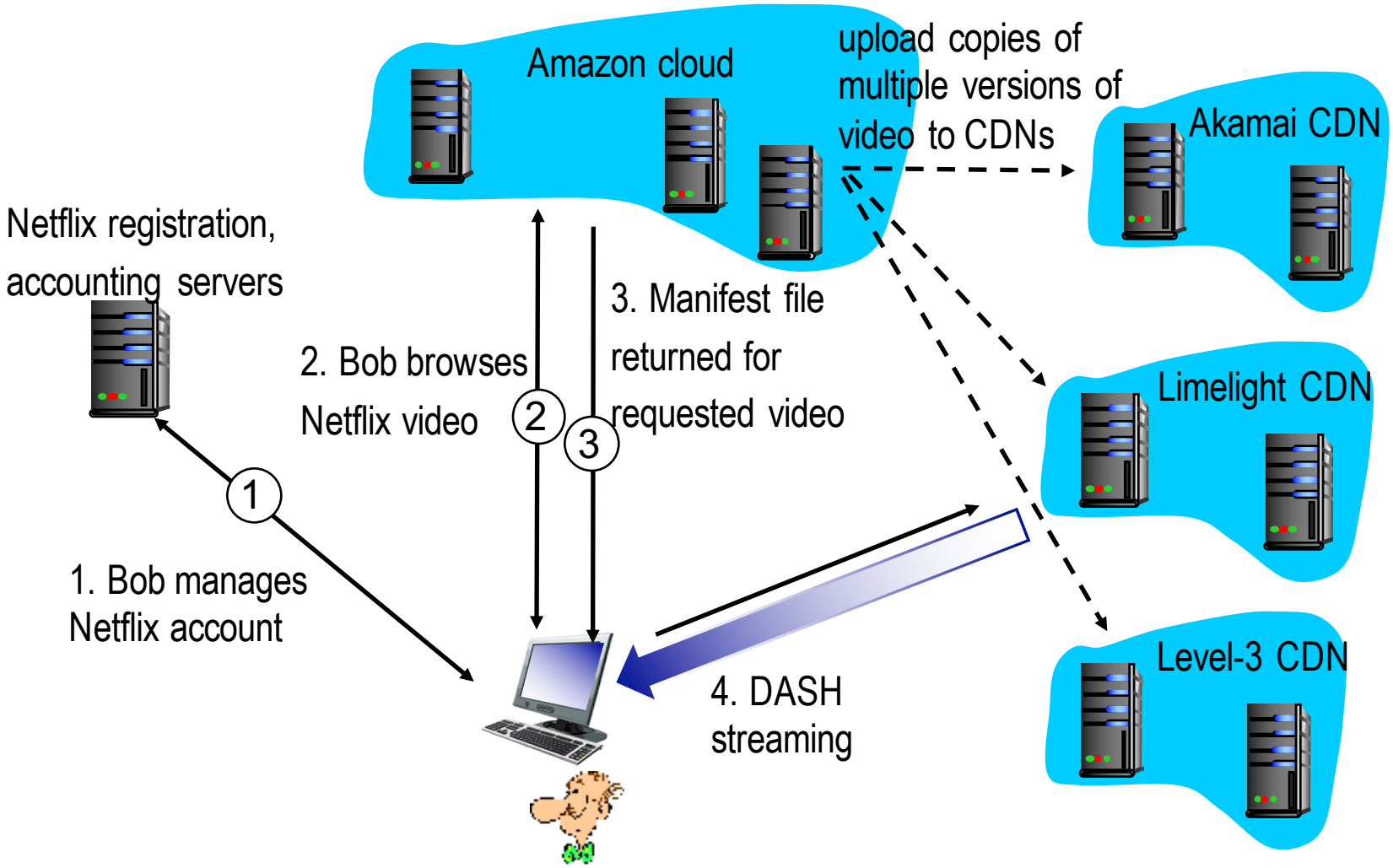
- how does CDN DNS select "good" CDN node to stream to client
  - pick CDN node geographically closest to client
  - pick CDN node with shortest delay (or min # hops) to client (CDN nodes periodically ping access ISPs, reporting results to CDN DNS)

- *alternative:* let *client* decide - give client a list of several CDN servers
  - client pings servers, picks "best"
  - Netflix approach

Case study: Netflix

- 30% downstream US traffic in 2011
- owns very little infrastructure, uses 3rd party services:
  - own registration, payment servers
  - Amazon (3rd party) cloud services:
    - Netflix uploads studio master to Amazon cloud
    - create multiple version of movie (different endodings) in cloud
    - upload versions from cloud to CDNs
    - Cloud hosts Netflix web pages for user browsing
  - *three* 3rd party CDNs host/stream Netflix content: Akamai, Limelight, Level-3

Case study: Netflix



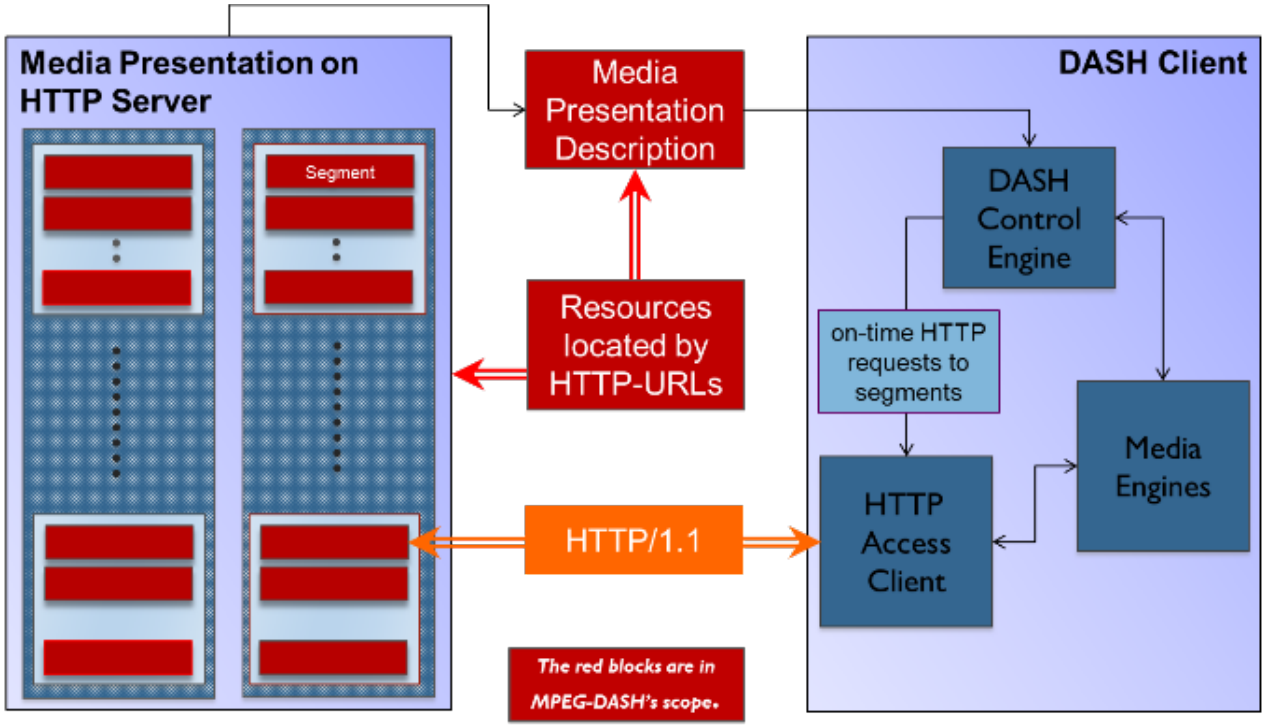
Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

- *"intelligence"* at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur)
  - *what encoding rate* to request (higher quality when more bandwidth available)
  - *where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

MPEG-DASH structure



http://dashif.org/mpeg-dash/