Chapter 2
Application Layer

A note on the use of these Powerpoint slides:
We’re making these slides freely available to all (faculty, students, readers). They’re in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we’d like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved
Chapter 2: outline

2.1 principles of network applications
2.2 Web and HTTP
2.3 electronic mail
   • SMTP, POP3, IMAP
2.4 DNS
2.5 P2P applications
2.6 video streaming and content distribution networks
2.7 socket programming with UDP and TCP
Web caches (proxy server)

**goal**: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client
More about Web caching

- cache acts as both client and server
  - server for original requesting client
  - client to origin server

- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?

- reduce response time for client request
- reduce traffic on an institution’s access link
- Internet dense with caches: enables “poor” content providers to effectively deliver content (so too does P2P file sharing)
Caching example:

**assumptions:**
- avg object size: 100K bits
- avg request rate from browsers to origin servers: 15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

**consequences:**
- LAN utilization: 15% problem!
- access link utilization = 99%
- total delay = Internet delay + access delay + LAN delay
  = 2 sec + minutes + usecs
assumptions:

- avg object size: 100K bits
- avg request rate from browsers to origin servers: 15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

consequences:

- LAN utilization: 15%
- access link utilization = 99% = 9.9%
- total delay = Internet delay + access delay + LAN delay
  = 2 sec + minutes + usecs + msecs

Cost: increased access link speed (not cheap!)
Caching example: install local cache

assumptions:
- avg object size: 100K bits
- avg request rate from browsers to origin servers: 15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

consequences:
- LAN utilization: 15%
- access link utilization = ?
- total delay = ?

How to compute link utilization, delay?

Cost: web cache (cheap!)
Caching example: install local cache

Calculating access link utilization, delay with cache:

- suppose cache hit rate is 0.4
  - 40% requests satisfied at cache, 60% requests satisfied at origin

- access link utilization:
  - 60% of requests use access link

- data rate to browsers over access link
  = 0.6*1.50 Mbps = .9 Mbps
  - utilization = 0.9/1.54 = .58

- total delay
  = 0.6 * (delay from origin servers) +0.4 * (delay when satisfied at cache)
  = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs
  - less than with 154 Mbps link (and cheaper too!)
Conditional GET

- **Goal:** don’t send object if cache has up-to-date cached version
  - no object transmission delay
  - lower link utilization
- **cache:** specify date of cached copy in HTTP request
  
  If-modified-since: <date>

- **server:** response contains no object if cached copy is up-to-date:
  
  HTTP/1.0 304 Not Modified

  If-modified-since: <date>

  HTTP/1.0 200 OK

  <data>
Chapter 2: outline

2.1 principles of network applications
2.2 Web and HTTP
2.3 electronic mail
   • SMTP, POP3, IMAP
2.4 DNS
2.5 P2P applications
2.6 video streaming and content distribution networks (CDNs)
2.7 socket programming with UDP and TCP
Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users

- challenge: scale - how to reach ~1B users?
  - single mega-video server won’t work (why?)

- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)

- solution: distributed, application-level infrastructure
Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy within and between images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: instead of sending N values of same color (all purple), send only two values: color value (*purple*) and number of repeated values (N)

frame i

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

frame i+1
Multimedia: video

- **CBR**: (constant bit rate): video encoding rate fixed
- **VBR**: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes

**examples:**
- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in Internet, < 1 Mbps)

**spatial coding example:** instead of sending $N$ values of same color (all purple), send only two values: color value (purple) and number of repeated values ($N$)

**temporal coding example:** instead of sending complete frame at $i+1$, send only differences from frame $i$
Streaming stored video:

simple scenario:

video server
(stored video)

Internet

client
Streaming multimedia: DASH

- **DASH:** Dynamic, Adaptive Streaming over HTTP

  - **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

- **DASH**: Dynamic, Adaptive Streaming over HTTP
- "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)
Content distribution networks

- **challenge**: how to stream content (selected from millions of videos) to hundreds of thousands of **simultaneous** users?

- **option 1**: 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

….quite simply: this solution *doesn’t scale*
Content distribution networks

- **challenge**: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- **option 2**: store/serve multiple 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
  - **bring home**: smaller number (10’s) of larger clusters in POPs near (but not within) access networks
    - used by Limelight
Content Distribution Networks (CDNs)

- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - may choose different copy if network path congested
Content Distribution Networks (CDNs)

“over the top”

Internet host-host communication as a service

**OTT challenges:** coping with a congested Internet

- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

*more .. in chapter 7*
Bob (client) requests video http://netcinema.com/6Y7B23V

- video stored in CDN at http://KingCDN.com/NetC6y&B23V

2. resolve http://netcinema.com/6Y7B23V via Bob’s local DNS
3. netcinema’s DNS returns URL http://KingCDN.com/NetC6y&B23V
4&5. Resolve http://KingCDN.com/NetC6y&B23 via KingCDN’s authoritative DNS, which returns IP address of KingCDN server with video
6. Bob requests video from KINGCDN server, streamed via HTTP
Case study: Netflix

1. Bob manages Netflix account

2. Bob browses Netflix video

3. Manifest file returned for requested video

4. DASH streaming

Amazon cloud

upload copies of multiple versions of video to CDN servers

CDN server

CDN server

CDN server

Netflix registration, accounting servers

Application Layer 2-22