CSCI 339
Distributed Systems

Lecture 10
CDNs and Web Proxy Caches
Mar 6, 2023
Administrative Details

• Project 2
  • Milestone due today
  • Everything due next Tue (including writeup)
• Porcupine paper summary due Thur
• Midterm on Glow next week
  • We’ll review on Monday in class
  • Midterm available on Glow from Mon Mar 13 – Mon Mar 20
  • Timed (90min?)
  • Fair game: projects (2), papers (4, SWORD is not required), lectures through next Monday
• Look for common themes
• “Study guide” posted on course webpage
Random Aside
Project 2 Writeup

• Should include:
  • Abstract/introduction that describes the purpose of the project
  • Description of the system's arch, including assumptions made
  • Overview of your “experimental setup”
  • Discussion/graph/table that addresses avg response time for 500 sequential searches
  • Discussion/graph/table that addresses avg response time for 500 sequential buys
  • Discussion/graph/table that addresses concurrent requests (searches and buys) with multiple clients
  • Conclusion/summary that addresses any problems and summarizes results
  • Sample client usage (screenshot is fine, helps make grading easier)
  • Run clients and server on different machines
Last Time

- Wrapped up RPCs
  - Focused on XML-RPC
  - Briefly mentioned SOAP, Java RMI, REST
  - Any questions??
- Started talking about CDNs
Today's Outline

• Discuss web proxy caches, CDNs, and CoDeen
• Maybe learn about email – another important modern network service
  • SMTP - Simple Mail Transfer Protocol
  • POP - Post Office Protocol
Recap: Akamai (CDN)

- Clients fetch html doc from primary server (e.g., CNN)
  - E.g. fetch index.html from cnn.com
- URLs for replicated content are replaced in html
  - E.g. `<img src="http://cnn.com/af/x.gif">` replaced with `<img src="http://a73.g.akamaitech.net/7/23/cnn.com/af/x.gif">`
- Client is forced to resolve aXYZ.g.akamaitech.net hostname
**CDN Example – Akamai**

GET `http://cnn.com`

1 - DNS Lookup
2 - Fetch page w/ "Akamaized" content
3 - DNS Lookup for Akamai URLs
4 - Fetch content

"Akamaized" response object has inline URLs for secondary content at (after resolving CNAMEs) `a73.g.akamaitech.net` and other Akamai-managed DNS names.

Akamai servers store/cache secondary content for "Akamaized" services.
How Akamai Works

• How is content replicated in CDN?
• Akamai only replicates static content (*)
• Modified name contains original file name
• Akamai server is asked for content
  • First checks local cache
  • If not in cache, requests file from primary server and caches file

* (At least, the version we're talking about today. Akamai actually lets sites write code that can run on Akamai's servers, but that's a pretty different beast)
How Akamai Works

cnn.com (content provider)  
DNS root server  
Akamai server

End-user  
Get index.html  
Get /cnn.com/foo.jpg
Akamai – Subsequent Requests

1. cnn.com (content provider)
2. DNS root server
3. Akamai high-level DNS server
4. Akamai low-level DNS server
5. Nearby matching Akamai server
6. End-user

Get index.html

Get /cnn.com/foo.jpg
Impact on DNS Usage

- DNS is increasingly being used for server selection
  - What are reasonable DNS TTLs for this type of use
  - Typically want to adapt to load changes
- How does this affect caching?
Another option: Web Proxy Caches

- User configures browser
- All Web accesses go thru 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
Caching Example (1)

Assumptions
- Average object size = 100,000 bits
- Avg. request rate from institution's browser to origin servers = 15/sec
- Delay from institutional router to any origin server and back to router = 2 sec

Consequences
- Utilization on LAN = 15%
- Utilization on access link = 100%
- Total delay = Internet delay + access delay + LAN delay
  = 2 sec + minutes + milliseconds
Caching Example (2)

Possible solution
• Increase bandwidth of access link to, say, 10 Mbps
• Often a costly upgrade

Consequences
• Utilization on LAN = 15%
• Utilization on access link = 15%
• Total delay = Internet delay + access delay + LAN delay
  = 2 sec + msecs + msecs
Caching Example (3)

Install cache
- Suppose hit rate is .4

Consequence
- 40% requests will be satisfied almost immediately (say 10 msec)
- 60% requests satisfied by origin server
- Utilization of access link reduced to 60%, resulting in negligible delays
- Weighted average of delays
  = .6*2 sec + .4*10msecs < 1.3 secs
Problems

• Many HTTP objects are uncacheable – why?
• Not easily solvable
  • Dynamic data ➔ stock prices, scores, web cams, facebook, etc
  • CGI/PHP scripts ➔ results based on passed parameters
• Problems with obvious fixes
  • SSL ➔ encrypted data is not cacheable
    • Most web clients don't handle mixed pages well ➔ many generic objects transferred with SSL
  • Cookies ➔ results may be based on passed data
Reliability and Security in the CoDeeN Content Distribution Network

Limin Wang, KyoungSoo Park, Ruoming Pang, Vivek Pai, Larry Peterson

Princeton University
Content Distribution Networks

- Replicates Web content broadly
- Redirects clients to "best" copy
  - Load, locality, proximity
- Offloads work from origin servers
- Multiplexes load spikes
  - Reduces overprovisioning
- Ex: Akamai, Mirror Image, Speedera
What Is CoDeeN?

- Academic Content Distribution Network/Web Proxy cache
- Forward/reverse proxies, redirector
- 100+ proxy servers on PlanetLab (in 2004)
- Continuous service, decentralized control
- Deployed for getting real traffic
Who Is The Target Audience?

• Now (2003)
  • Users wanting better performance
  • People seeking "anonymity"

• Next
  • Content providers seeking load sharing

• Later
  • General support for absorbing flash crowds
  • Avoid the "Slashdot Effect"
How Does It Work?

• Server surrogates (proxies) on most North American sites
  • Originally everywhere, but we cut back
• Clients specify proxy to use
  • Cache hits served locally
  • Cache misses forwarded to CoDeeN nodes
    • Maybe forwarded to origin servers
Goals of CoDeeN

• Provide open content distribution
• Improve web performance & reliability
• Platform for testing new innovations
  • Particularly in live environments
• Keep CoDeeN running 24/7
  • Security
  • Reliability
How Does CoDeeN Work?

Each CoDeeN proxy is a forward proxy, reverse proxy, & redirector.

CoDeeN Reliability & Security
USENIX-04
Types of Security Problems

- Spammers
- Bandwidth hogs
- High request rates
- Content thieves
- Worrisome anonymity

Commonality: using CoDeeN to do things they would not do directly
The Root of All Trouble

CoDeeN Reliability & Security
USENIX-04

No End-To-End Authentication

(Malicious) Client
Approaches to Security

- Desired: allow only "safe" accesses
- No research in "partially open" proxies
- Our approach
  - Rate limiting
  - Privilege separation
Rate Limiting

- 3 scales capture burstiness
- Exceptions
  - Login attempts
  - Vulnerability tests
  - Repetition, request spreading
Privilege Separation

Site B
Proxy

Unprivileged Request

Site A
Proxy

Privileged Request

Remote
Client

Site A
Client

Site A
Server
Thoughts?

• This was a meaty paper!
• What were the high-level ideas that you learned?
• What did you find surprising?
• What did you find confusing?
Moving on...
Intro to Email: Terminology

• **Mail User Agent**: end-user email program

• **Mail Transfer Agent**: program responsible for communicating with remote hosts and transmitting/receiving email
  - Think of this as the mail server software

• **Mail Exchanger**: host with specific IP address (registered with DNS) that takes care of email for a domain
Simple Mail Transfer Protocol
SMTP

- Used to send mail messages between mail servers (Message Transfer Agents)
SMTP Protocol

• One of the core Internet protocols
• SMTP sender is the email client
• SMTP receiver is the email server
• Alternating dialogue:
  • Client sends command and server responds with command status message
  • Order of the commands is important
  • Status messages include ASCII encoded numeric status code (like HTTP) and text string
  • Typically runs on port 25
SMTP Commands

• **HELO** - identifies sender (kinda)

• **MAIL FROM:** - starts a mail transaction and identifies the mail originator

• **RCPT TO:** - identifies individual recipient. There may be multiple **RCPT TO:** commands

• **DATA** - sender ready to transmit a series of lines of text, each ends with \r\n. A line containing only a period '.' indicates the end of the data
SMTP Data Format

- ASCII only - must convert binary to an ASCII representation to send via email
  - Attachments handled using MIME extension
- What if we want to send a line containing only a period?
  - We need an escape character sequence…
  - Sender prepends a period to any line starting with a period (in the message)
  - Receiver strips the leading period in any line that starts with a period and has more “stuff”
Typical Exchange

albrecht:~ jeannie$ telnet bull.cs.williams.edu 25
Trying 137.165.8.2...
Connected to bull.cs.williams.edu.
Escape character is '^]'.
220 bull.cs.williams.edu ESMTP Sendmail; Mon, 6 Mar 2023 06:51:24 -0500 (EST)
HELO sysnet.cs.williams.edu
250 bull.cs.williams.edu Hello sysnet.cs.williams.edu, pleased to meet you
MAIL FROM: <jeannie@cs.williams.edu>
250 2.1.0 <jeannie@cs.williams.edu>... Sender ok
RCPT TO: jeannie
250 2.1.5 jeannie... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
This is a test
This is really cool
.
250 2.0.0 m1PBpOGY042309 Message accepted for delivery
QUIT
Leading Period

DATA
354 Enter mail, end with "." on a line by itself
Hi class - this message is a test of SMTP
..
..some more text here
..
.
250 2.0.0 mIPBpOGY042309 Message accepted for delivery
Resulting Message:

Hi class - this message is a test of SMTP
.
..some more text here
.
.
Other SMTP Commands

- **VRFY** - confirm that a name is a valid recipient
- **EXPN** - expand an alias (group email address)
- **TURN** - switch roles (sender <=> receiver)
- **SOML** - Send Or Mail; if recipient is logged in, display message on terminal, otherwise email.
- **SAML** - Send and Mail
- **NOOP** - send back a positive reply code
- **RSET** - abort current transaction
Mail Headers

• Email messages contain many headers
  • Some headers are created by MUA
  • Some automatically added by MTA
• Every MTA adds (at least) a "Received:" header
  • May add many more
  • Spamassassin?
• Some of the headers are read (and parsed) by intermediate MTAs, but the content is always ignored and passed on unmodified
POP – *Post Office Protocol*

- Used to transfer mail from a mail server to a MUA
POP (version 3)

- Similar to SMTP command/reply lockstep protocol
- Used to retrieve mail for a single user
  - Requires authentication
- Commands and replies are ASCII lines
  - Replies start with "+OK" or "-ERR".
  - Replies may contain multiple lines
POP-3 Commands

- **USER** - specify username
- **PASS** - specify password
- **STAT** - get mailbox status
  - Return number of messages in the mailbox
- **LIST** - get a list of messages and sizes
  - One per line, termination line contains '.' only
- **RETR** - retrieve a message
- **DELE** - mark a message for deletion from the mailbox
- **NOOP** - send back positive reply
- **RSET** - reset; all deletion marks are unmarked
- **QUIT** - remove marked messages and close the (TCP) connection
A POP-3 Exchange

albrecht:~ telnet fuji.cs.williams.edu 110
Trying 137.165.8.2...
Connected to fuji.cs.williams.edu.
Escape character is '^]'.
+OK POP3 at fuji.cs.williams.edu server ready

USER jcool
+OK Name is a valid mailbox

PASS nottelling
+OK jeannie has 1 visible message in 1761 octets.

STAT
+OK 1 1761

LIST
+OK 1 visible messages (1761 octets)
1 1761

*Note: We no longer run a POP server for the CS dept. 😞 This is an old exchange.
POP3 Example Continued

- RETR 1
  +OK 1761 octets
  Received: from fuji.cs.williams.edu
  From: "Jeannie Albrecht" <jeannie@cs.williams.edu>
  To: <jeannie@cs.williams.edu>
  Subject: test
  Date: Tue, 11 Mar 2020 10:31:21 -0400
  Content-Type: text/plain; charset="US-ASCII"

  test
  .
IMAP

- IMAP stands for *Internet Message Access Protocol*
- Very widely used today
- Used to transfer messages from server to client
IMAP vs. POP

IMAP

Client decides where to store messages. Copy them locally or leave them on the server.

Client has full control over their mail folders. Can create, delete, purge or move them around - even between different accounts on different servers.

POP

Clients independently download email. Once downloaded it is deleted from the server.

http://wiki.bath.ac.uk/display/bucstech/Imap+overview
IMAP Advantages Over POP

- POP clients connect, download, and disconnect. IMAP clients often stay connected and download on demand.
- POP only allows one connected client at a time; simultaneous access by multiple IMAP clients is allowed.
- POP does not allow message state (i.e., read/unread) to be stored on server; IMAP does.
- POP does not allow for server-side searchers, IMAP does.
IMAP Disadvantages

- More complex protocol
- Clients need to maintain TCP/IP connection to server (this is a big deal!)
  - If server goes down, user cannot access (even previously read) messages
  - Flaky network connections render clients unusable