

# CS162 Operating Systems and Systems Programming Lecture 24

## Client-Server

April 23, 2012  
Anthony D. Joseph and Ion Stoica  
<http://inst.eecs.berkeley.edu/~cs162>

## Client-Server

- One or more clients interacting with one or more servers providing a service, e.g.,
  - Web
  - E-mail, chat
  - Printer
  - Airline reservation
  - On-line shopping
  - Store/streaming video, audio, and/or photos
  - ...
- In this lecture
  - End-to-end message communication
  - Remote Procedure Calls
  - World Wide Web

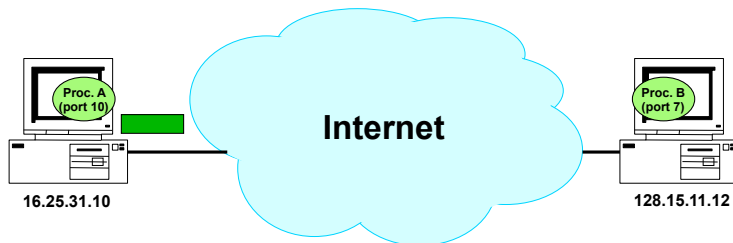
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.2

## Message Passing

- Process A (e.g., client) sends a packet to process B (e.g., server)

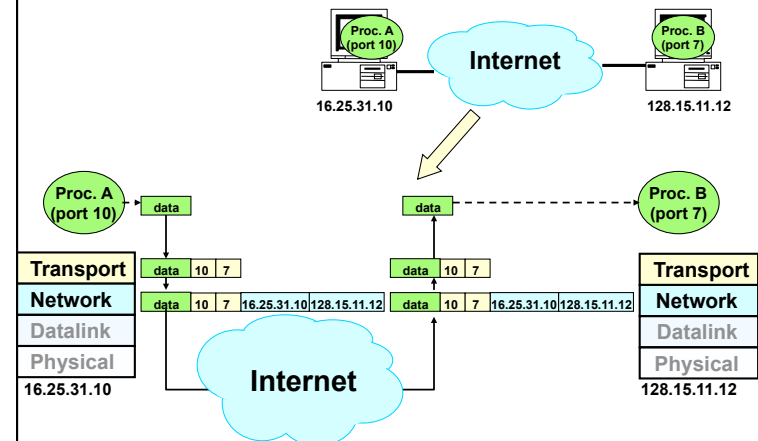


4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.3

## Message Passing Details



4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.4

## From Message Passing to Remote Procedure Call

- Raw messaging is a bit too low-level for programming
- Another option: Remote Procedure Call (RPC)
  - Looks like a local procedure call on client
  - Translated automatically into a procedure call on remote machine (server)
- RPC's can be used to communicate between address spaces on different machines *or the same machine*
  - Services can be run wherever it's most appropriate
  - Access to local and remote services looks the same
- Examples of modern RPC systems:
  - CORBA (Common Object Request Broker Architecture)
  - DCOM (Distributed COM)
  - RMI (Java Remote Method Invocation)
- Implementation:
  - Uses request/response message passing “under the covers”

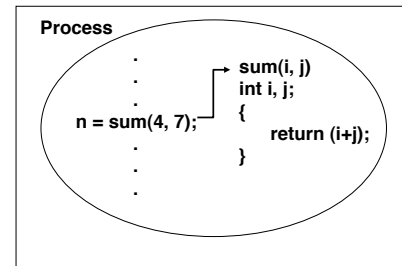
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.5

## Example: Local Procedure Call

Machine



4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.6

## Remote Procedure Call

- **Transparently** invoke a procedure (services) implemented in a different address space either on the same machine or a **different** machine
  - Services can be run wherever it's most appropriate
  - Access to local and remote services looks the same
- Challenges:
  - Argument (parameter) passing, potentially across different architectures
  - Discover where the service is located
  - Handle failures **transparently**

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.7

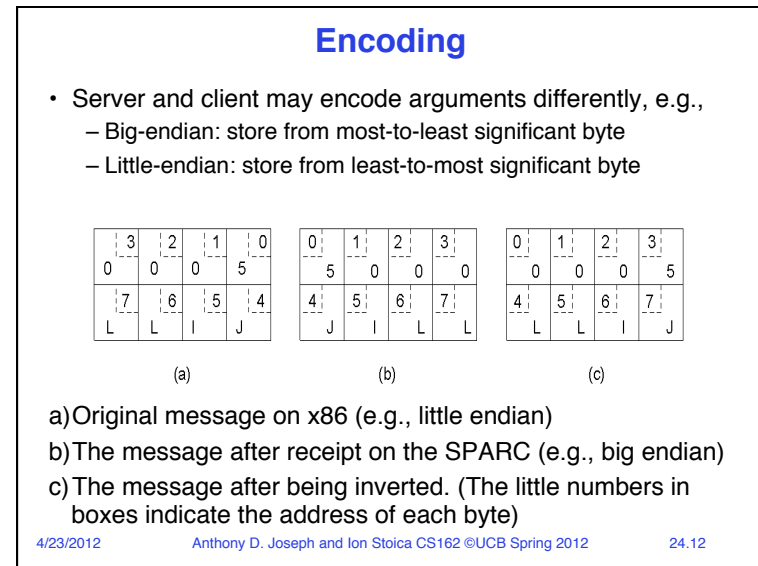
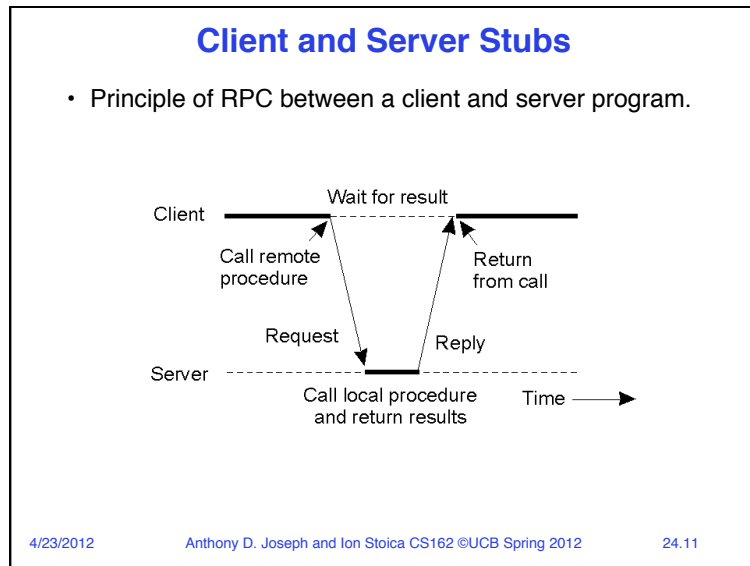
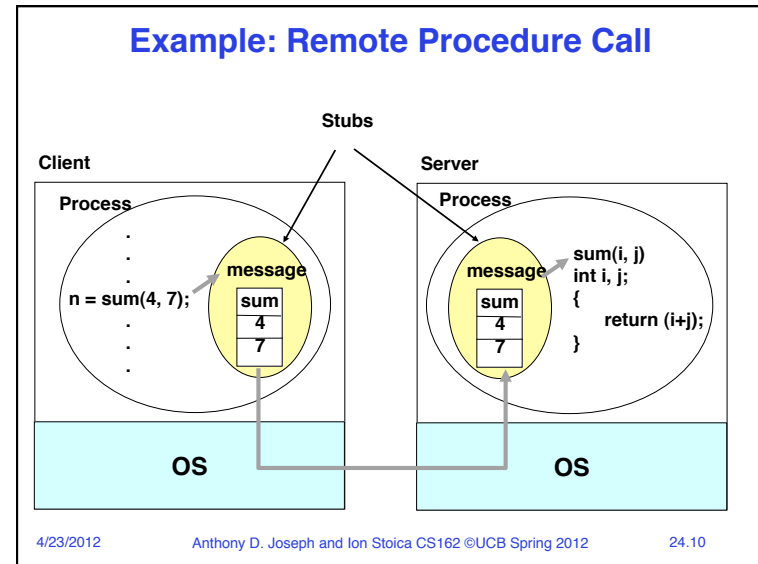
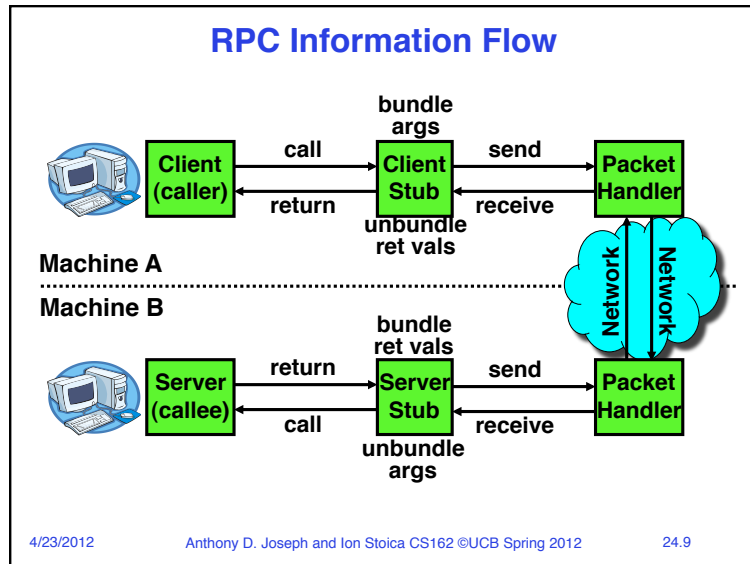
## RPC: Argument Passing

- Client and server use “stubs” to glue pieces together
  - Client-side stub is responsible for “marshalling” arguments and “unmarshalling” the return values
  - Server-side stub is responsible for “unmarshalling” arguments and “marshalling” the return values
- **Marshalling** involves (depending on system) converting values to a canonical form, serializing objects, copying arguments passed by reference, etc.
  - Needs to account for cross-language and cross-platform issues
- Technique: compiler generated stubs
  - Input: interface definition language (IDL)
    - » Contains, among other things, types of arguments/return
  - Output: stub code in the appropriate source language

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.8



## Parameter Specification and Stub Generation

- a) A procedure  
b) The corresponding message.

```
foobar( char x; float y; int z[5] )
{
  ....
}
```

(a)

foobar's local variables	
x	
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

(b)

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.13

## Service Discovery: RPC Binding

- How does client know which machine to send RPC?
  - Need to translate name of remote service into network endpoint (e.g., host:port)
  - **Binding**: the process of converting a user-visible name into a network endpoint
    - » Static: fixed at compile time
    - » Dynamic: performed at runtime
- Dynamic Binding
  - Most RPC systems use dynamic binding via name service
  - Why dynamic binding?
    - » Access control: check who is permitted to access service
    - » Fail-over: If server fails, use a different one

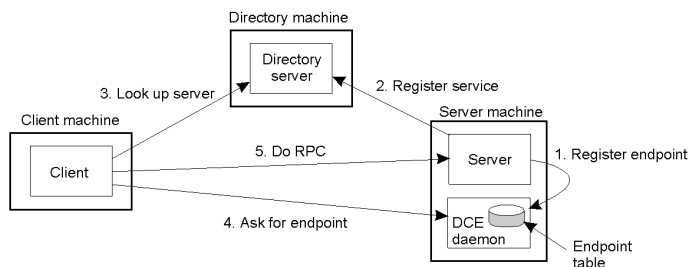
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.14

## Example of RPC Binding

- Distributed Computing Environment (DCE) framework
- DCE daemon:
  - Allow local services to record their services locally
  - Resolve service name to local end-point (i.e., port)
- Directory machine: resolve service name to DCE daemon (host:port) on machine running the service



4

## RPC Semantics in the Presence of Failures

- The client is unable to locate the server
- The request message from the client to server is lost
- The reply message from the server is lost
- The server crashes after receiving a request
- The client crashes after sending a request

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.16

## Client is Unable to Locate Server

- Causes: server down, different version of server binary, ...
- Fixes
  - Return (-1) to indicate failure (in Unix use *errno* to indicate failure type)
    - » What if (-1) is a legal return value?
  - Use exceptions
    - » Transparency is lost

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.17

## Lost Request Message

- Easiest to deal with
- Just retransmit the message!
- If multiple message are lost then
  - “client is unable to locate server” error

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.18

## Lost Reply Message

- Far more difficult to deal with: client doesn't know what happened at server
  - Did server execute the procedure or not?
- Possible fixes
  - Retransmit the request
    - » Only works if operation is **idempotent**: it's fine to execute it twice
  - What if operation not idempotent?
    - » Assign unique sequence numbers to every request

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.19

## Server Crashes

- Three cases
  - Crash after execution
  - Crash before execution
  - Crash during the execution
- Three possible semantics
  - At least once semantics
    - » Client keeps trying until it gets a reply
  - At most once semantics
    - » Client gives up on failure
  - Exactly once semantics
    - » Can this be correctly implemented?

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.20

## Client Crashes

- Let's the server computation **orphan**
- Orphans can
  - Waste CPU cycles
  - Lock files
  - Client reboots and it gets the old reply immediately

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.21

## Client Crashes: Possible Solutions

- **Extermination:**
  - Client keeps a log, reads it when reboots, and kills the orphan
  - Disadvantage: high overhead to maintain the log
- **Reincarnation:**
  - Divide times in epochs
  - Client broadcasts epoch when reboots
  - Upon hearing a new epoch servers kills the orphans
  - Disadvantage: doesn't solve problem when network partitioned
- **Expiration:**
  - Each RPC is given a lease  $T$  to finish computation
  - If it does not, it needs to ask for another lease
  - If client reboots after  $T$  sec all orphans are gone
  - Problem: what is a good value of  $T$ ?

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.22

## RPC Semantics: Discussion

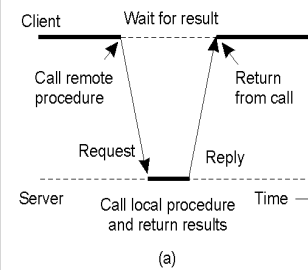
- The original goal: provide the same semantics as a local call
- Impossible to achieve in a distributed system
  - Dealing with remote failures fundamentally affects transparency
- Ideal interface: balance the easy of use with making visible the errors to users

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.23

## Asynchronous RPC (1)



- (a)
- The interconnection between client and server in a traditional RPC
  - The interaction using asynchronous RPC

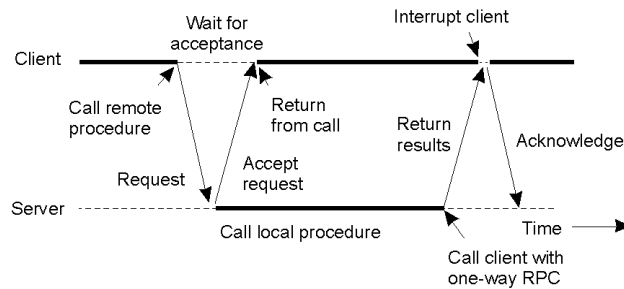
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.24

## Asynchronous RPC (2)

- A client and server interacting through two asynchronous RPCs



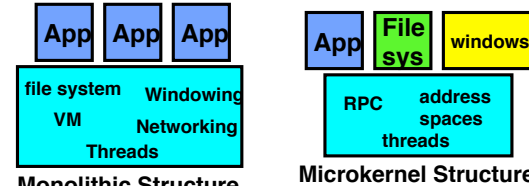
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.25

## Microkernel operating systems

- Example: split kernel into application-level servers.
  - File system looks remote, even though on same machine



- Why split the OS into separate domains?
  - Fault isolation: bugs are more isolated (build a firewall)
  - Enforces modularity: allows incremental upgrades of pieces of software (client or server)
  - Location transparent: service can be local or remote
    - » For example in the X windowing system: Each X client can be on a separate machine from X server; Neither has to run on the machine with the frame buffer.

4/23/2012

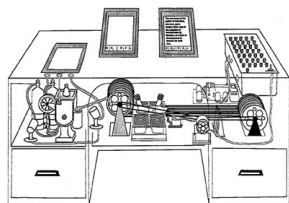
Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.26

## The Web – History (I)



Vannevar Bush (1890-1974)



4/23/2012

Memex Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.27

- 1945: Vannevar Bush, Memex:
- *“a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility”*

(See <http://www.iath.virginia.edu/elab/hf10051.html>)

## The Web – History (II)



Ted Nelson

- 1967, Ted Nelson, Xanadu:
  - A world-wide publishing network that would allow information to be stored not as separate files but as connected literature
  - Owners of documents would be automatically paid via electronic means for the virtual copying of their documents
- Coined the term “Hypertext”

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.28

## The Web – History (III)



Tim Berners-Lee

- World Wide Web (WWW): a distributed database of “pages” linked through [Hypertext Transport Protocol \(HTTP\)](#)
  - First HTTP implementation - 1990
    - » Tim Berners-Lee at CERN
  - HTTP/0.9 – 1991
    - » Simple GET command for the Web
  - HTTP/1.0 – 1992
    - » Client/Server information, simple caching
  - HTTP/1.1 - 1996

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.29

## Administrivia

- Wednesday 4/25 guest lecture – Harry Li from Facebook
- No sections this week
  - TAs will have extended office hours (see post on Piazza)
- Project 4 design due Today at 11:59pm
  - Design review sign up sheets are all online <http://piazza.com/class/spring2012/cs162/732>
- Final exam – Fri 5/11 11:30-2:30pm in 145 Dwinelle
  - Comprehensive
  - Closed book, two double-sided handwritten notes sheets
  - Let us know any conflicts ASAP
  - Review session next week (TBA on Piazza)

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.30

## 5min Break

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.31

## The Web

- Core components:
  - Servers: store files and execute remote commands
  - Browsers: retrieve and display “pages”
  - Uniform Resource Locators (URLs): way to refer to pages
- A protocol to transfer information between clients and servers
  - Hyper Text Transport Protocol (HTTP)

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.32



## Uniform Record Locator (URL)

protocol://host-name:port/directory-path/resource

- Extend the idea of hierarchical namespaces to include anything in a file system
  - <ftp://www.cs.berkeley.edu/~adj/publications.html>
- Extend to program executions as well...
  - [http://us.f413.mail.yahoo.com/ym/ShowLetter?box=%40B%40Bulk&MsgId=2604\\_1744106\\_29699\\_1123\\_1261\\_0\\_28917\\_3552\\_1289957100&Search=&Nhead=f&YY=31454&order=down&sort=date&pos=0&view=a&head=b](http://us.f413.mail.yahoo.com/ym/ShowLetter?box=%40B%40Bulk&MsgId=2604_1744106_29699_1123_1261_0_28917_3552_1289957100&Search=&Nhead=f&YY=31454&order=down&sort=date&pos=0&view=a&head=b)
  - Server side processing can be incorporated in the name

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.33

## Hyper Text Transfer Protocol (HTTP)

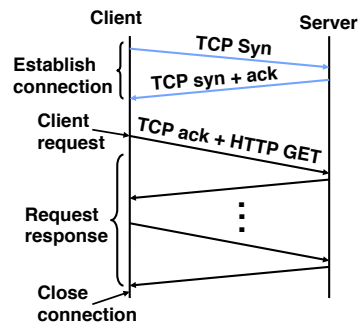
- Client-server architecture
- Synchronous request/reply protocol
  - Runs over TCP, Port 80
- Stateless

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.34

## Big Picture



4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.35

## Hyper Text Transfer Protocol 1.0 Commands

- GET\* – transfer resource from given URL
- HEAD\* – GET resource metadata (headers) only
- POST<sup>x</sup> – provide input for a process identified by the given URL (usually used to post CGI parameters)
- PUT<sup>x+</sup> – store/modify resource under given URL
- DELETE<sup>x+</sup> – remove resource

\*mandatory    <sup>x</sup>may have side effects    <sup>+</sup>idempotent

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.36

## Response Codes

- 1x informational
- 2x success
  - 200 OK is most common
- 3x redirection
- 4x client error in request
  - 403 Forbidden and 404 Not Found are most common
- 5x server error; can't satisfy the request
  - 500 Internal Server Error is most common

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.37

## Client Request

- Steps to get the resource:

<http://www.eecs.berkeley.edu/index.html>

1. Use DNS to obtain the IP address of [www.eecs.berkeley.edu](http://www.eecs.berkeley.edu)

2. Send to an HTTP request:

```
GET /index.html HTTP/1.0
```

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.38

## Server Response

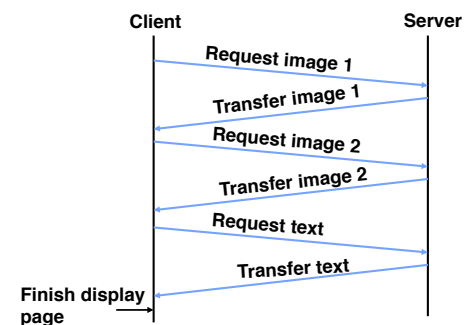
```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 1234
Last-Modified: Mon, 19 Nov
2001 15:31:20 GMT
<HTML>
<HEAD>
<TITLE>EECS Home Page</TITLE>
</HEAD>
...
</BODY>
</HTML>
```

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.39

## HTTP/1.0 Example



4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.40

## HTTP/1.0 Performance

- Create a new TCP connection for each resource
  - Large number of embedded objects in a web page
  - Many short lived connections
- TCP transfer
  - Too slow for small object
  - It takes time to ramp-up (i.e., exit slow-start phase)
- Connections may be set up in parallel (5 is default in most browsers)

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.41

## HTTP/1.0 Caching Support

- A modifier to the GET request:
  - `If-modified-since` – return a “not modified” response if resource was not modified since specified time
- A response header:
  - `Expires` – specify to the client for how long it is safe to cache the resource
- A request directive:
  - `No-cache` – ignore all caches and get resource directly from server
- These features can be best taken advantage of with HTTP proxies and caches
  - Locality of reference “increases” if many clients share a proxy/cache

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.42

## HTTP/1.1 (1996)

- Performance:
  - Persistent connections
  - Pipelined requests/responses
  - Byte-serving
  - ...
- Efficient caching support
  - Network Cache assumed more explicitly in the design
  - Gives more control to the server on how it wants data cached
- Support for virtual hosting
  - Allows to run multiple web servers on the same machine

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.43

## Persistent Connections

- Allow multiple transfers over one connection
- Avoid multiple TCP connection setups
- Avoid multiple TCP slow starts (i.e., TCP ramp ups)

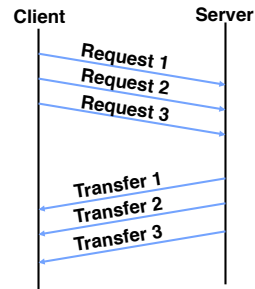
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.44

## Pipelined Requests/Responses

- Buffer requests and responses to reduce the number of packets
- Multiple requests can be contained in one TCP segment
- Note: order of responses has to be maintained



4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.45

## Caching and Replication

- Problem: You are a web content provider
  - How do you handle millions of web clients?
  - How do you ensure that all clients experience good performance?
  - How do you maintain availability in the presence of server and network failures?
- Solutions:
  - Add more servers at different locations → If you are CNN this might work!
  - Client-side and/or server-side Caching
  - Content Distribution Networks (Replication)

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.46

## WWW Caching

- Use client-side caching to reduce number of interactions between clients and servers and/or reduce the size of the interactions:
  - Time-to-Live (TTL) fields – HTTP “Expires” header from server
  - Client polling – HTTP “If-Modified-Since” request headers from clients
  - Server refresh – HTML “META Refresh tag” causes periodic client poll
- What is the polling frequency for clients and servers?
  - Could be adaptive based upon a page’s age and its rate of change
- Server load is still significant!

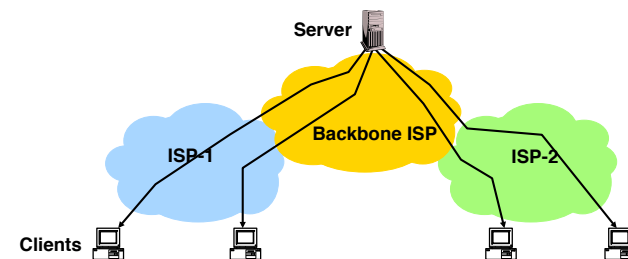
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.47

## “Baseline”

- Many clients transfer same information
  - Generate unnecessary server and network load
  - Clients experience unnecessary latency



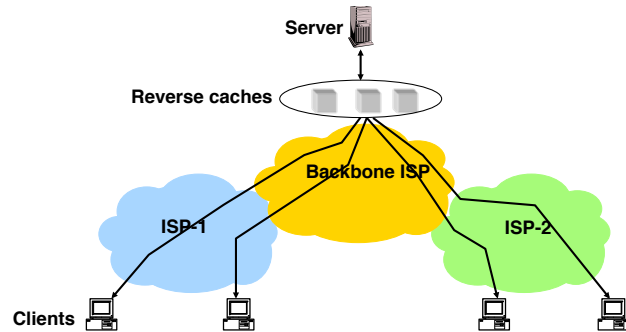
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.48

## Reverse Caches

- Cache documents close to server → decrease server load
- Typically done by content providers
- Offloads busy server machines



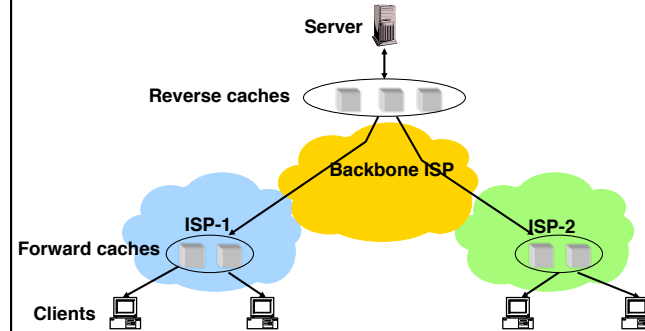
4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.49

## Forward Proxies

- Cache documents close to clients → reduce network traffic and decrease latency
- Typically done by ISPs or corporate LANs



4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.50

## Content Distribution Networks (CDNs)

- Integrate forward and reverse caching functionalities into one overlay network (usually) administrated by one entity
  - Example: Akamai
- Documents are cached both
  - As a result of clients' requests (pull)
  - Pushed in the expectation of a high access rate
- Beside caching do processing, e.g.,
  - Handle dynamic web pages
  - Transcoding

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.51

## Example: Akamai

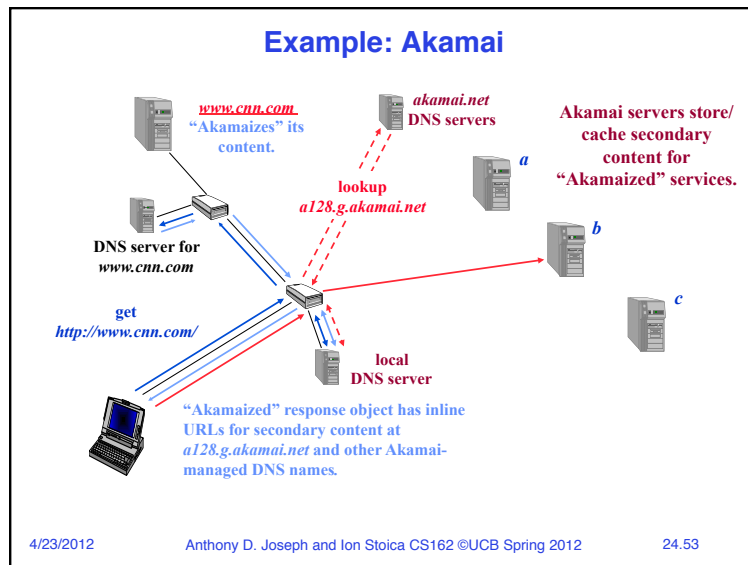
- Akamai creates new domain names for each client content provider.
  - e.g., [a128.g.akamai.net](http://a128.g.akamai.net)
- The CDN's DNS servers are authoritative for the new domains
- The client content provider modifies its content so that embedded URLs reference the new domains.
  - "Akamaize" content, e.g.: <http://www.cnn.com/image-of-the-day.gif> becomes <http://a128.g.akamai.net/image-of-the-day.gif>.

4/23/2012

Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012

24.52

52



- ### Caching Challenges
- Caching static traffic easy, but only ~40% of traffic
    - Zipf-like distribution of community references to web pages
  - Dynamic and multimedia is harder
    - Multimedia is a big win: Mbytes/Gbytes versus Kilobytes
  - Same cache consistency problems as before
    - What is replacement policy?
  - Caching is changing the Internet architecture
    - Places functionality at higher levels of comm. protocols
    - Authentication and computation in caches
- 4/23/2012      Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012      24.54

- ### Summary
- Remote Procedure Call (RPC): Call procedure (service) on remote machine
    - Provides same interface as local procedure call
    - Automatic packing and unpacking of arguments without user programming (in stub)
  - Hypertext Transport Protocol: request-response
    - Use DNS to locate server
    - HTTP 1.1 vs. 1.0: added support for persistent connections and pipeline to improve performance
    - Caching: key to increase scalability
- 4/23/2012      Anthony D. Joseph and Ion Stoica CS162 ©UCB Spring 2012      24.55