COT 4600 Operating Systems Fall 2009

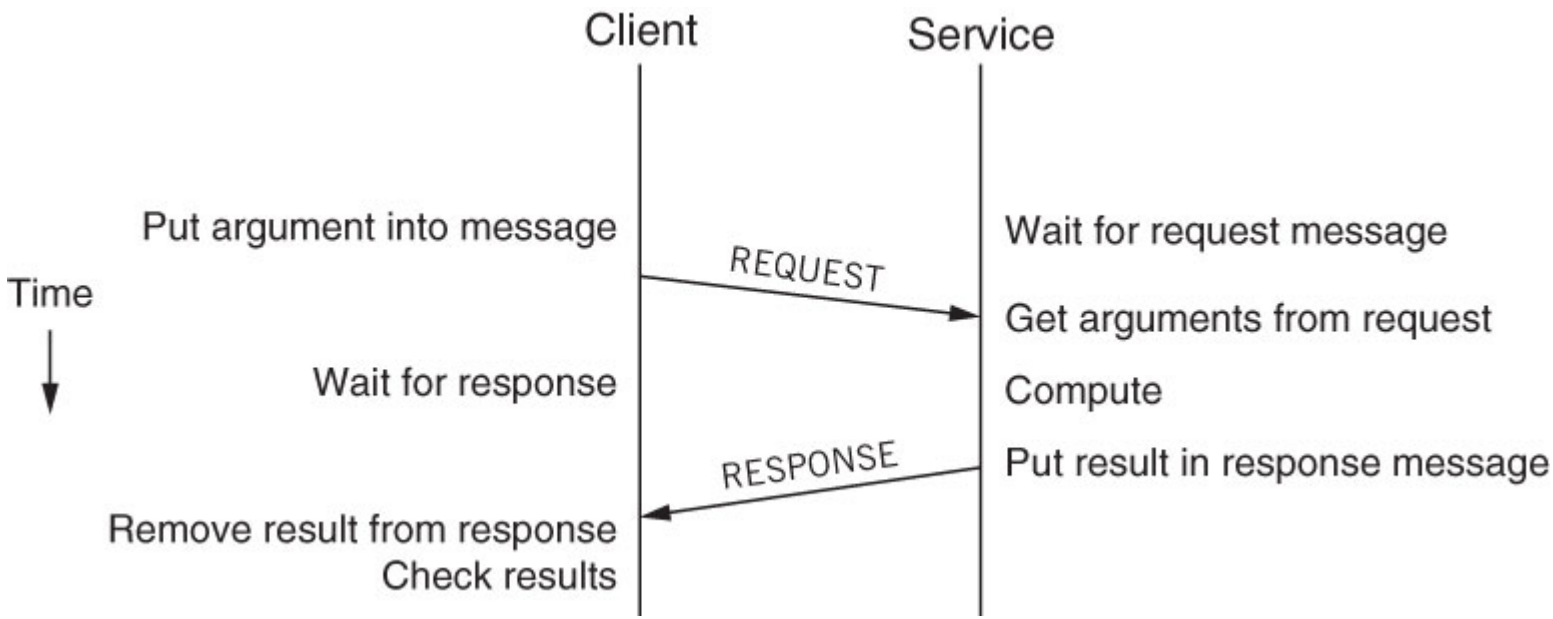
Dan C. Marinescu

Office: HEC 439 B

Office hours: Tu-Th 3:00-4:00 PM

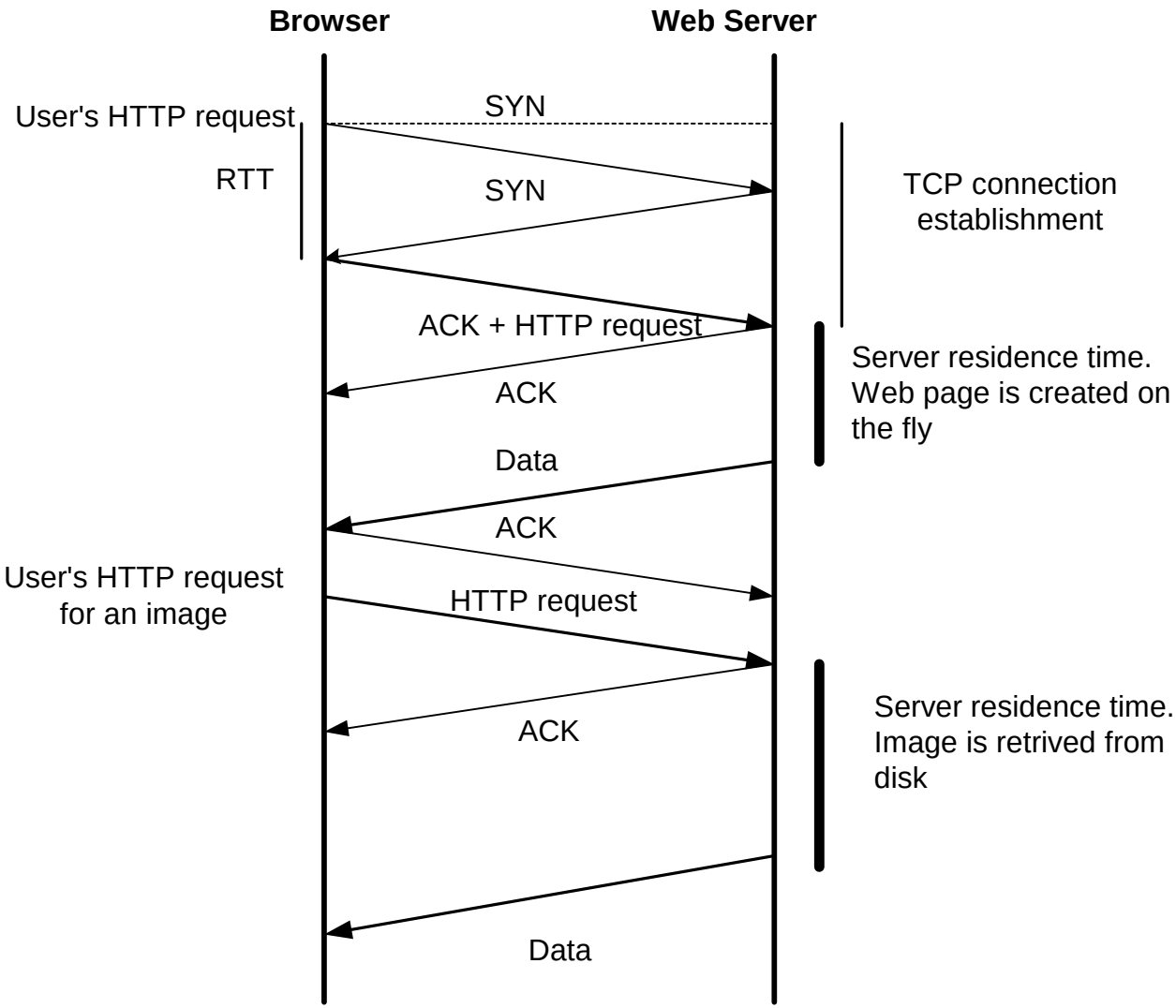
Lecture 11

- Last time:
  - User-friendly names; Lifetime of names
  - Case study: URL
  - Soft modularity
  - Hard modularity
- Today:
  - Discussion of homework 2
  - Client/Service organization
  - Remote Procedure Call (RPC)
  - Domain Name Service (DNS)
- Next Time:
  - Network File System (NFS)


Client
Service
Put argument into message
REQUEST
Wait for request message
Time
Get arguments from request
Wait for response
Compute
RESPONSE
Put result in response message
Remove result from response
Check results

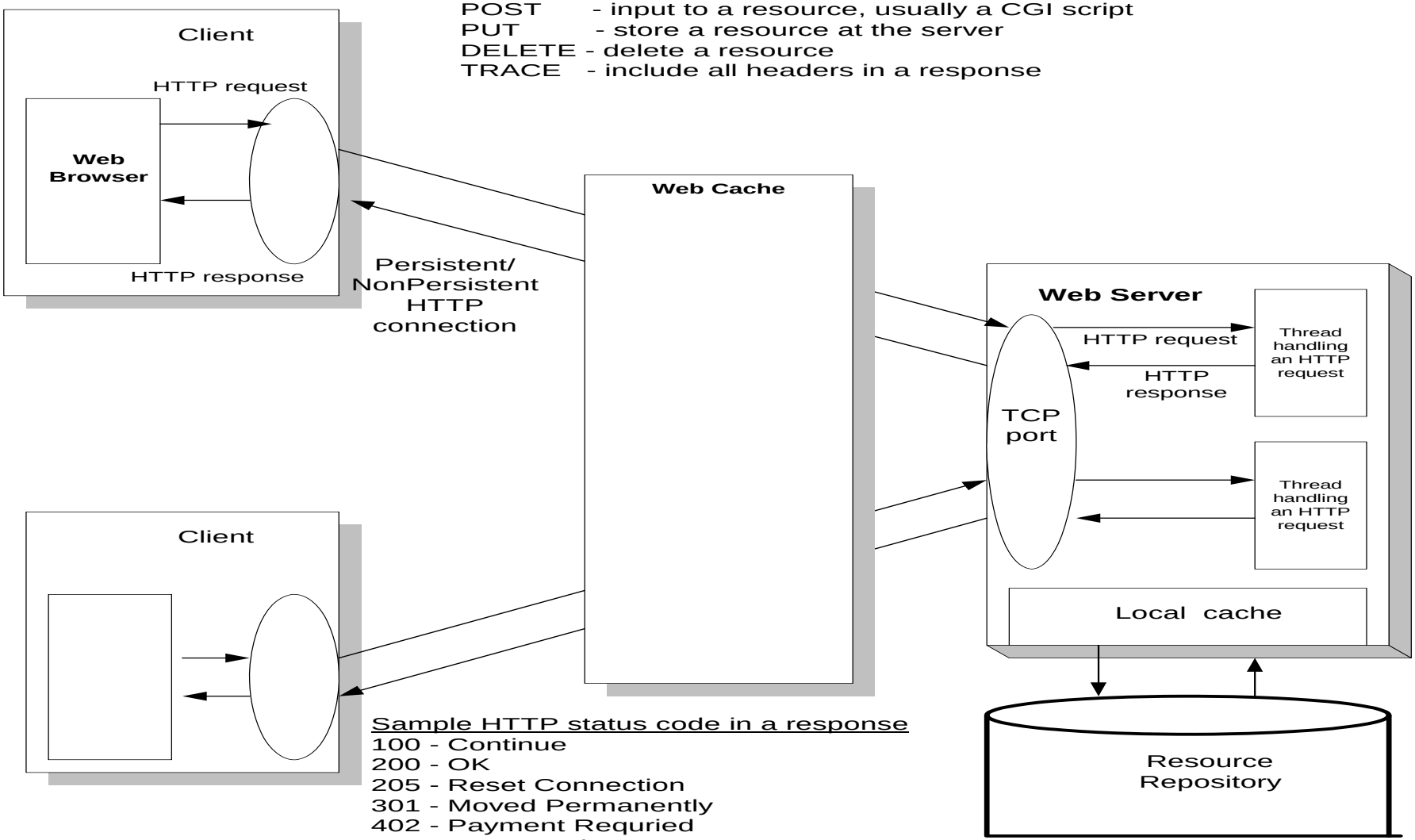
A client-service system → the World Wide Web

- The information in each page is encoded and formatted according to some standard, e.g.
  - images: GIF, JPEG,
  - video: MPEG
  - audio: MP3
- The web is based upon a “pull” paradigm. The server has the resources and the client pulls it from the server.
- The Web server also called an HTTP server listens at a well known port, port 80 for connections from clients.
- The HTTP protocol uses TCP to establish a connection between the client and the server.
- Some pages are created on the “fly” other have to be fetched from the disk.


Browser
Web Server
User's HTTP request
RTT
SYN
SYN
TCP connection establishment
ACK + HTTP request
ACK
Server residence time. Web page is created on the fly
Data
ACK
User's HTTP request for an image
HTTP request
ACK
Server residence time. Image is retrived from disk
Data

An HTTP regust contains one of the following methods:

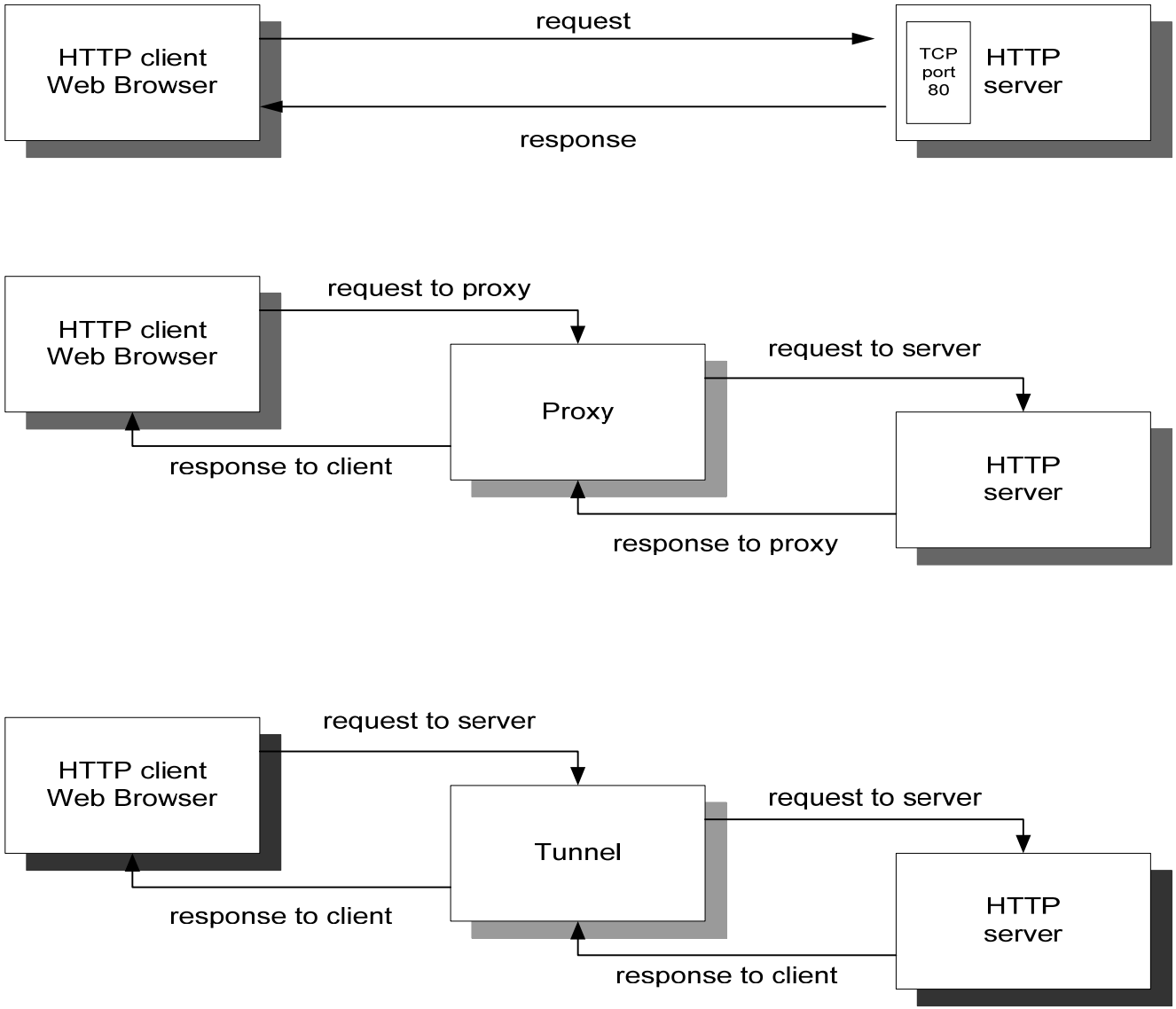
GET - get a resource
HEAD - verify the link and conditions of a resource
POST - input to a resource, usually a CGI script
PUT - store a resource at the server
DELETE - delete a resource
TRACE - include all headers in a response



Sample HTTP status code in a response

100 - Continue
200 - OK
205 - Reset Connection
301 - Moved Permanently
402 - Payment Requried
404 - Not Found
405 - Method Not Allowed
407 - Proxy Authentication Required
415 - Unsupported Media Type
500 - Internal Server Error
504 - Gateway Timeout
505 - HTTP version Not Supported

Client server interactions in HTTP



Client/service organization

- Not only separates functions but also enforces this separation!!
  - No globally-shared state (e.g., through the stack)
  - Errors can only propagate from client to service and vice-versa only if the messages are not properly checked.
  - A client can use time-outs to detect a non-responsive service and take another course of action.
  - The separation of abstraction from implementation is clearer; the client needs only to know the format of the message, the implementation of the service may change.

Heterogeneity

- The client and the service may run on systems with different:
  - internal data representation, e.g., big versus little endian
  - processor architecture, e.g., 32 bit /64 bit addressing
  - operating systems, e.g., version of Linux, Mac OS, etc.
  - libraries
- Multiple clients and services provided/available on systems with different characteristics :
  - the same service may be provided by multiple systems;
  - a service may in turn user services available on other systems;
  - the same client may use multiple services.
- Marshaling/unmarshaling → conversion of the information in a message into a canonical representation and back

Little endian and big endian

Words	0				1			
Bytes	0	1	…	7	0	1	…	7
Bits	$2^0\ 2^1\ 2^2 \ldots$			$\ldots 2^{63}$	$2^0\ 2^1\ 2^2 \ldots$			$\ldots 2^{63}$

Words	n				$n-1$			
Bytes	7	…	1	0	7	…	1	0
Bits	$2^{63} \ldots$			$\ldots 2^2\ 2^1\ 2^0$	$2^{63} \ldots$			$\ldots 2^2\ 2^1\ 2^0$

Timing; response time

- The client and the service are connected via communication channel.
- The response time is a function of the latency and the bandwidth of the communication channel. Distinguish between
  - service time
  - communication time
- Synchronous call → the client blocks waiting for the response. Easier to manage.
- Asynchronous calls → the client does not block.
- Multi-threading and asynchronous calls.
- Message buffering
  - in kernel space (to allow clients to make asynchronous calls)
  - in user space (before sending)

Example: the X-windows (X11)

- X11 → software system and network protocol that provides a GUI for networked computer. Developed as part of Project Athena at MIT in 1984.
- Separates
  - the service program → manipulates the display from
  - the client program → uses the display.
- An application running on one machine can access the display on a different computer.
- Clients operate asynchronously, multiple requests can be sent → the display rate could be much higher than the rate between the client and the server.

Trusted intermediary

- Trusted service acting as an intermediary among multiple clients.
  - Enforces modularity → a fault of one client does not affect other clients.
  - Examples:
    - File systems
    - Mail systems
- Supports thin-clients → a significant part of client functionality is transferred to the intermediary.
    - In a thin client/server system, the only software installed on the thin client is the user interface, certain frequently used applications, and a networked operating system. By simplifying the load on the thin client, it can be a very small, low-powered device giving lower costs to purchase and to operate per seat.
    - The server, or a cluster of servers has the full weight of all the applications, services, and data. By keeping a few servers busy and many thin clients lightly loaded, users can expect easier system management and lower costs, as well as all the advantages of networked computing: central storage/backup and easier security.
    - Because the thin client is relatively passive and low-maintenance, but numerous, the entire system is simpler and easier to install and to operate. As the cost of hardware plunges and the cost of employing a technician, buying energy, and disposing of waste rises, the advantages of thin clients grow. From the user's perspective, the interaction with monitor, keyboard, and cursor changes little from using a thick client.

Editor is a client of →File service which is a client of →Block-storage service
File service is a trusted intermediary.