

ELEC / COMP 177 – Fall 2015

Computer Networking

→ HTTP Revisited

Some slides from Kurose and Ross, *Computer Networking*, 5th Edition

Upcoming Schedule

- **Project 2 – Python HTTP Server v2**
 - Starts next week!
- Checkpoint 1 - **Due Oct 4th**
- Checkpoint 1 - **Due Oct 11th**
- Final Project - **Due Oct 18th**

HTTP Operation - Revisited

State-of-the-Web

Slashdot (15) slashdot.org

Step Motors for the Agricultural Industry [Learn More](#)

SAP Turn Data Into Decisions A Big Data Resource Center sourceforge

Slashdot Channels Jobs Newsletter Submit Login Join

stories [Follow Slashdot stories on Twitter](#)

submissions

popular [« Newer](#) [Older »](#)

blog

all stories

ask slashdot

book reviews

games

idle

yro

cloud

hardware

linux

management

mobile

science

security

Follow us:

London Tube Cleaners Don't Want Fingerprint Clock-In

Posted by **Unknown Lamer** on Tuesday September 17, 2013 @12:04AM from the wait'll-they-roll-out-dna-based-timecards dept.

Bismillah writes

"Biometrics is hot stuff, not just for Apple but cleaning companies like the UK division of Denmark's IIS which tidies the London Underground railway network. However, the cleaners [aren't happy about having to clock in and out with biometric fingerprint sensors](#), and are [taking industrial action](#) to stop the practice."

[Read More](#) security uk biometrics

Today Yesterday

FEMA Grounds Private Drones That Were Helping To Map Boulder Floods

Posted by **Unknown Lamer** on Monday September 16, 2013 @10:13PM from the we-don't-know-what-we're-doing-but-we'll-arrest-you dept.

First time accepted submitter MrMagoAZ writes

"An interesting article about a questionable reaction by FEMA in response to the flooding in Colorado. It seems a small firm was working free of charge with County

Build Your Dream 1U Rackmount

LOGIC SUPPLY

Slashdot Poll

I use spinning-drive storage media ...

- For absolutely everything (or just about)
- More than solid-state, but not exclusively
- About the same as I use solid-state storage media

State-of-the-Web

- Loading `slashdot.org`
 - 99 requests for files
 - 15 HTML
 - 3 stylesheets
 - 36 images
 - 35 scripts
 - 2 XHR
 - 8 “other” (empty – ads?)
 - 760 KB

How can we do this quickly / efficiently?

HTTP/1.0 Operation

- 1 file transferred per socket connection
 - **Client** opens socket
 - **Client** sends request
 - **Server** sends reply
 - **Server** closes socket

Opportunity for improvement here...

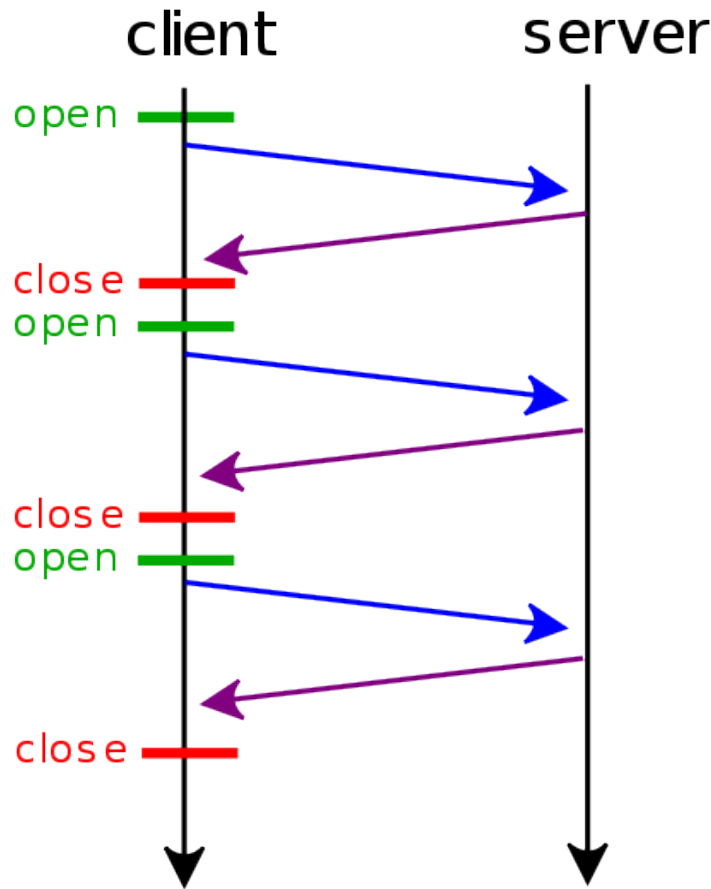
HTTP/1.1 Operation

(with Persistent Connections, aka Keep-Alive)

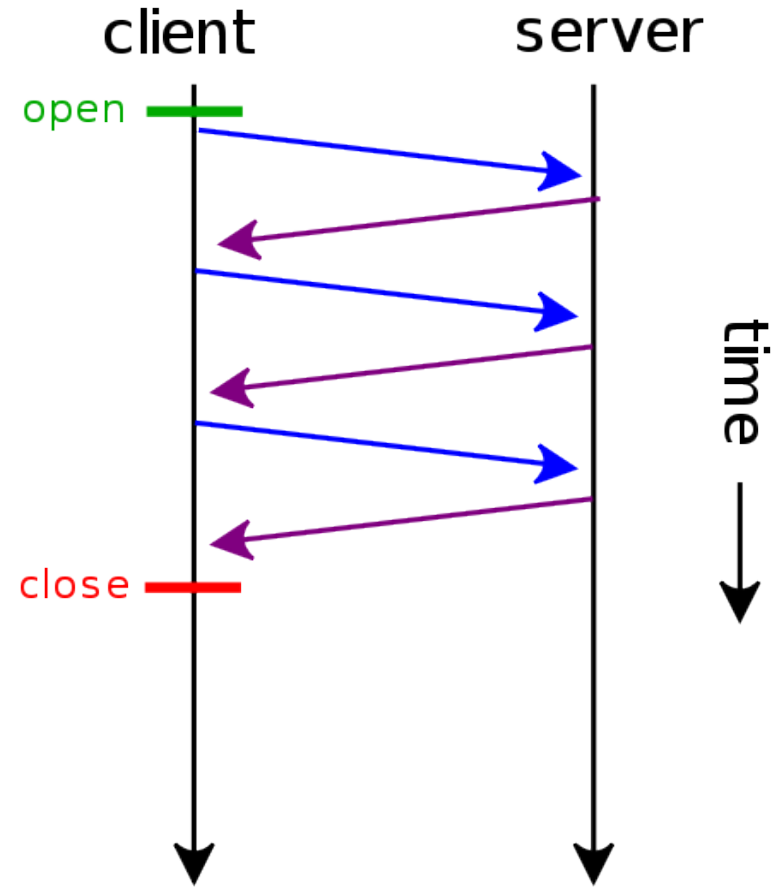
- Multiple files transferred per socket connection
 - **Client** opens socket
 - **Client** sends request 1
 - **Server** sends reply 1
 - **Server** keeps socket open for “a while”
 - **Client** sends request 2
 - **Server** sends reply 2
 - **Server** keeps socket open for “a while”

Persistent Connections

multiple connections



persistent connection



Persistent Connections

- **What are the advantages of persistent connections?**
 - **Client:** Reduced latency for requests 2- n (no need to open a new connection)
 - **Server:** Reduced CPU/memory usage (fewer connections to manage)

Persistent Connections

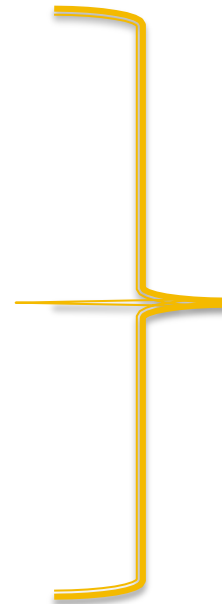
- The `content-length` header (provided by the server response) is the length of the file in bytes
- **Why is this header required when using persistent connections?**
 - The client needs to know when the **file is finished**
 - *Previously, the server closing the socket would signal the end-of-file condition*

Persistent Connections

- What if I don't know the length of the file at the beginning? (e.g. dynamic content)
- **HTTP Chunked Encoding**
 - New header (`Transfer-encoding: chunked`)
 - Send a "chunk" of data with a known length
 - Can send subsequent chunks with known length
 - Final chunk at end with length of zero bytes
- Client always knows
 - How much data to expect next
 - When the end-of-file is reached

HTTP/1.1 Operation (with Pipelined Connections)

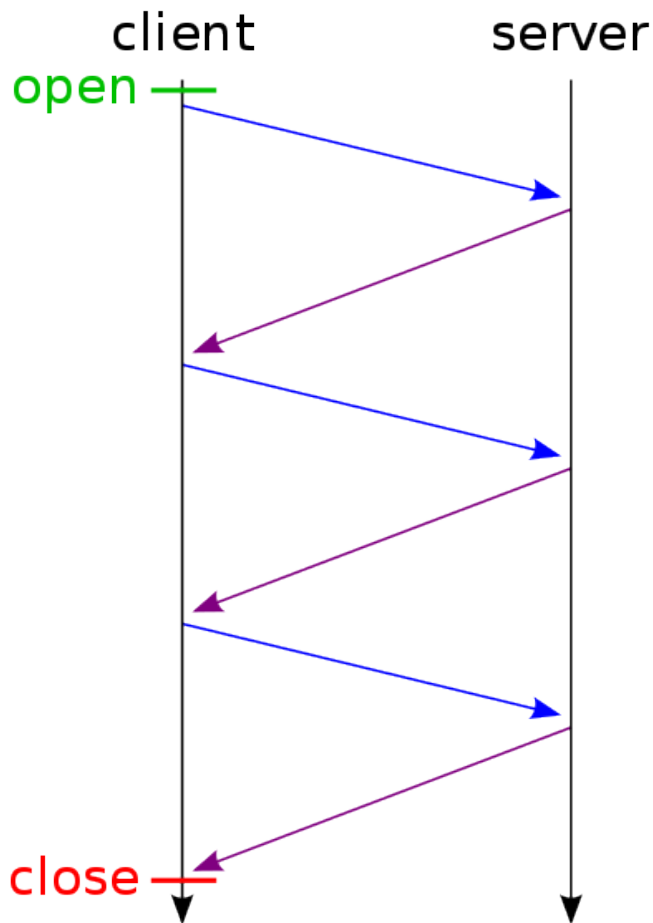
- Multiple files transferred per socket connection
 - **Client** opens socket
 - **Client** sends request 1
 - **Client** sends request 2
 - **Client** sends request n
 - **Server** sends reply 1
 - **Server** sends reply 2
 - **Server** sends reply n
 - **Server** keeps socket open for “a while” (i.e. keep-alive)



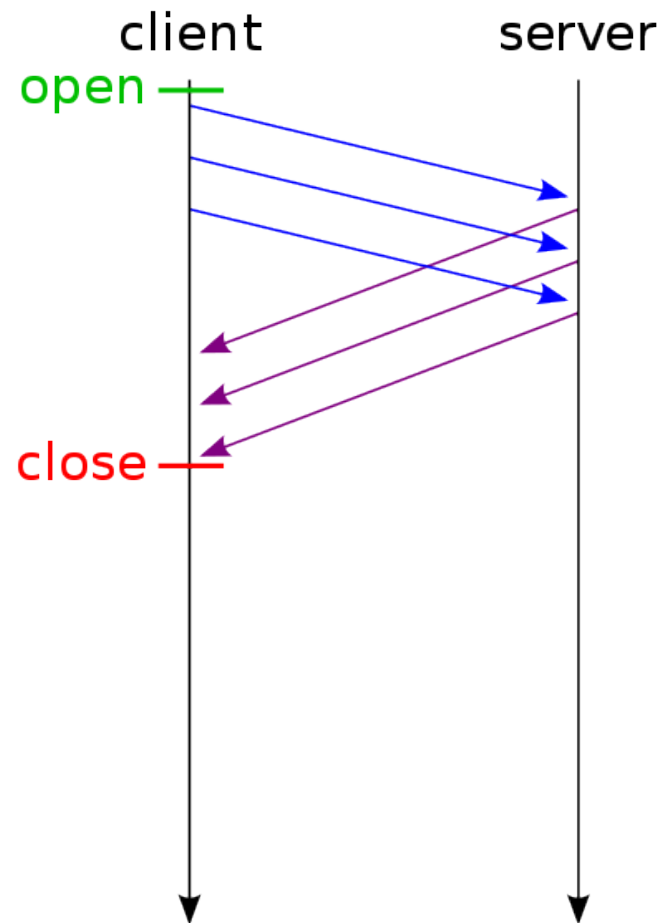
Client and server communication can overlap. The server does not have to wait for the client to finish sending requests to reply to the first request...

Pipelined Connections

no pipelining



pipelining



time ↓

Pipelined Connections

- **What are the advantages of pipelined connections?**
 - **Client:** Reduced latency for requests 2- n (server can immediately send subsequent files)
- **Note:** You can have both persistent and pipelined connections together