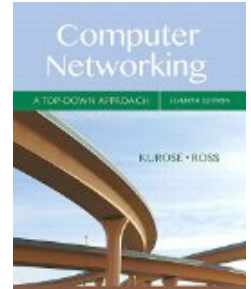


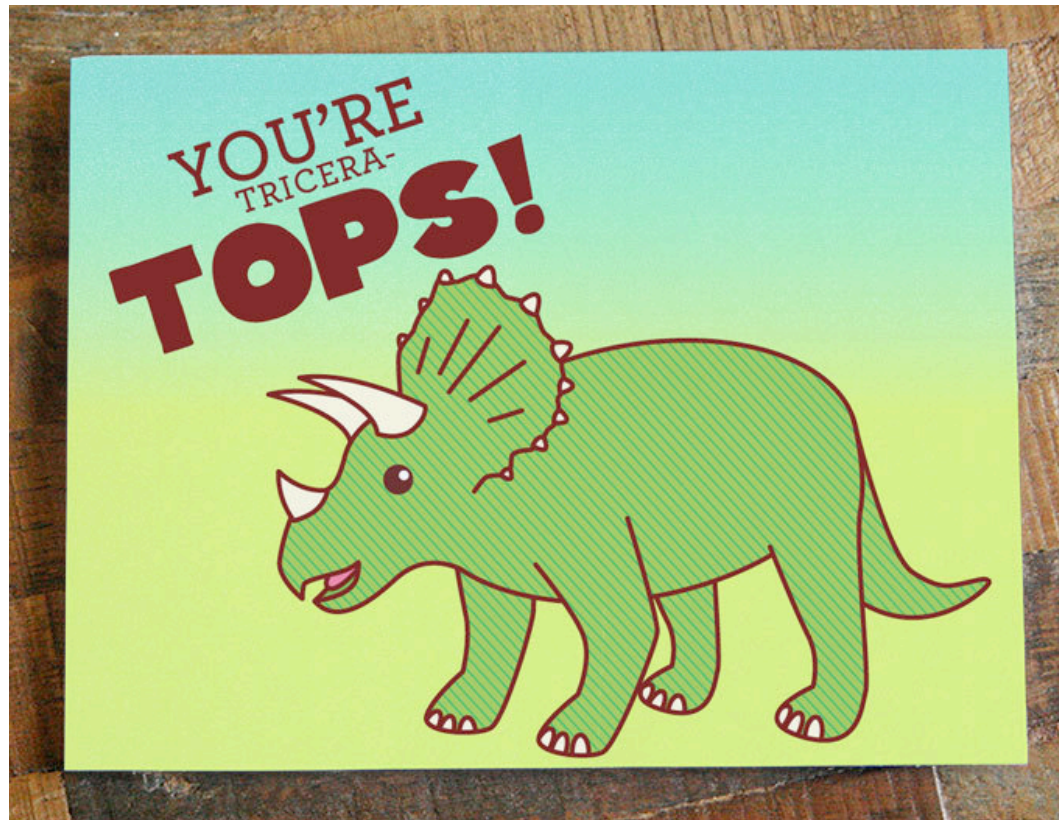
# COMP 375: Lecture 05



- **News & Notes:**
  - Quiz #1 in class TODAY
  - Project #1 due Monday @ 10PM
- **Reading (Fri, Feb. 9)**
  - Section 2.3 (Email)

# Quiz #1

- Closed book. Closed notes.
- Happy Send a Card to a Friend Day!



Section 2.2

# **THE WEB & HTTP**

How many **objects** need to be sent by the server to fully load this page?



Sat Garcia @ University of San ... Outlook Web App


home.sandiego.edu/~sat/ Search

University of San Diego ABOUT ME TEACHING RESEARCH

## Saturnino (Sat) Garcia

Assistant Professor of Computer Science  
Department of Mathematics and Computer Science  
University of San Diego  
Office: Serra Hall, Room 159E  
Phone: (619) 260-4017  
e-mail: sat @ sandiego.edu



### About Me

Hi, my name is Saturnino ("Sat") Garcia and I'm an assistant professor of computer science in the [Math and Computer Science department](#) at the [University of San Diego](#). My research is in the area of program analysis, compilers, and parallel computing.

I received my Ph.D. in Computer Science in 2012 from [UC San Diego](#) where I was advised by [Michael Taylor](#). Before that, I graduated from [Drexel University](#) in 2005 with a B.S. in Computer Engineering. While there, I worked with Drs. Moshe Kam and Kapil Dandekar. Going even further back, I was born and raised in the small town of [Willard, Ohio](#). People often inquire into the origin of my name. I shared my name with my father, a 1st generation Mexican-American whose purchase of a Tandy computer set me on my path to computer science.

### Teaching

University of San Diego:

Current (Fall 2014):

A.	1
B.	2
C.	3
D.	4
<b>E.</b>	> 4

# You can simulate an HTTP client using the telnet program.

1. Telnet to your favorite Web server:

**telnet home.sandiego.edu 80**

*Opens TCP connection to port 80 (default HTTP server port) at USD's homepage server. Anything typed is sent to server on port 80 at home.sandiego.edu*

2. Type in a GET HTTP request:

**GET /~sat/ HTTP/1.1**

**Host: home.sandiego.edu**

By typing this in (hit enter/return twice), you send this minimal (but complete) GET request to the HTTP server.

3. Look at response message sent by HTTP server!

# An HTTP request contains a required request line plus optional headers.

**Request line** (GET, POST, etc.)

carriage return (CR) character  
line-feed (LF) character

**Header lines**

```
GET /index.html HTTP/1.1\r\n
Host: home.sandiego.edu.edu\r\n
User-Agent: Firefox/47.0\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

Request MUST end with **blank line**.

Which of these is a **valid HTTP request** message for the following URL?

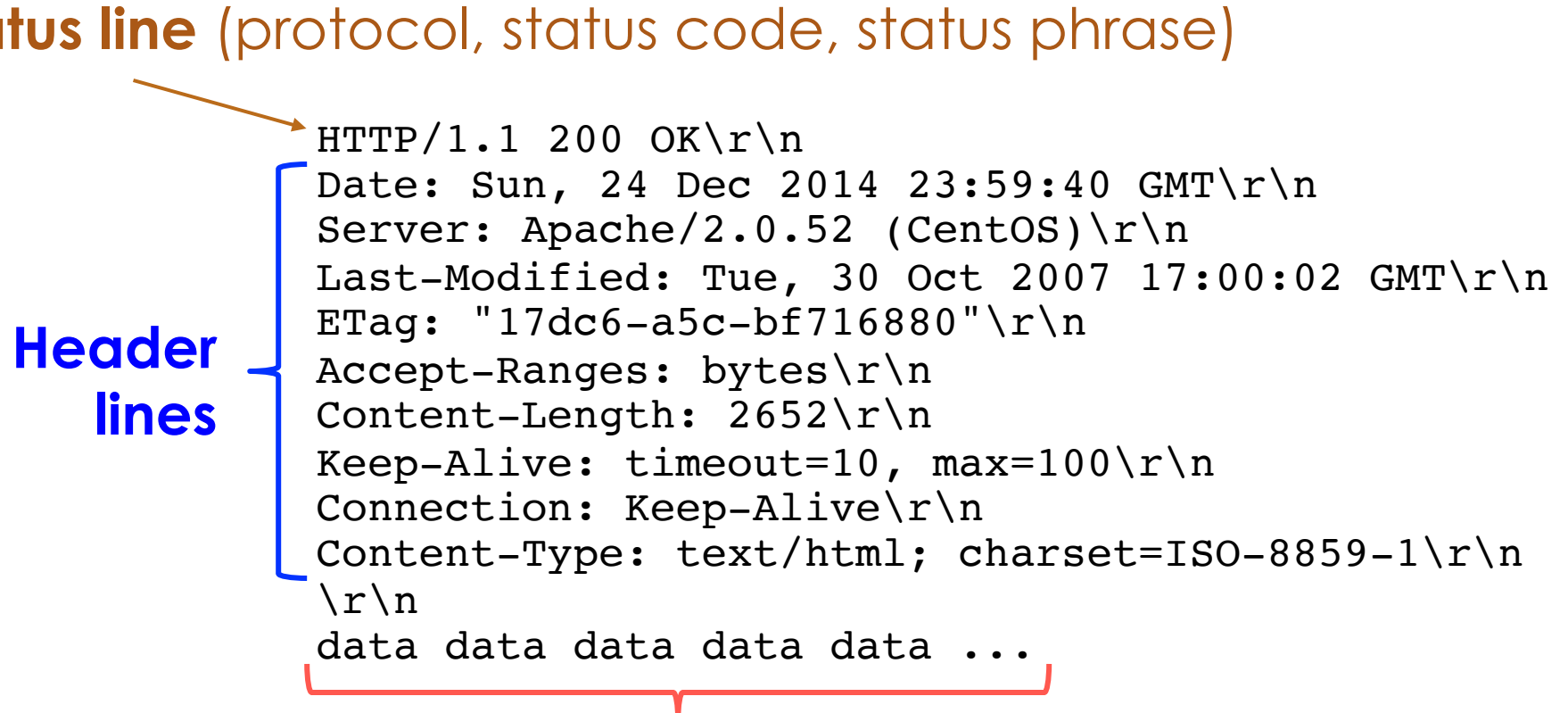
`http://home.sandiego.edu/~sat/meow.jpg`

<b>A.</b>	GET meow.jpg HTTP/1.0 Host: home.sandiego.edu/~sat/
<b>B.</b>	GET /~sat/meow.jpg HTTP/1.0 Host: home.sandiego.edu
<b>C.</b>	GET home.sandiego.edu/~sat/meow.jpg HTTP/1.0 Host: home.sandiego.edu
<b>D.</b>	More than one of the above
<b>E.</b>	None of these are correct.

# HTTP responses follow a similar format to requests.

**Status line** (protocol, status code, status phrase)

**Header  
lines**



The diagram illustrates the structure of an HTTP response. It shows a sequence of lines: a status line, followed by several header lines, and finally a data section. An orange arrow points from the 'Status line' label to the first line of the response. A blue bracket groups the subsequent lines as 'Header lines'. A red bracket groups the final line as 'The data'.

```
HTTP/1.1 200 OK\r\n
Date: Sun, 24 Dec 2014 23:59:40 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

The **data** (e.g. the requested HTML file)

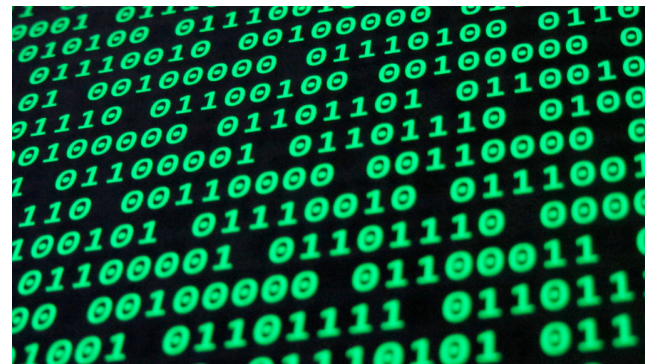


HTTP trades efficiency for simplicity  
by being text-based.

***Text-based Protocol (HTTP):***

```
HTTP/1.1 200 OK\r\n  
Date: Sun, 24 Dec 2014 23:59:40 GMT\r\n
```

***Binary Protocol (e.g. DNS):***



# Sometimes HTTP requests aren't OK.

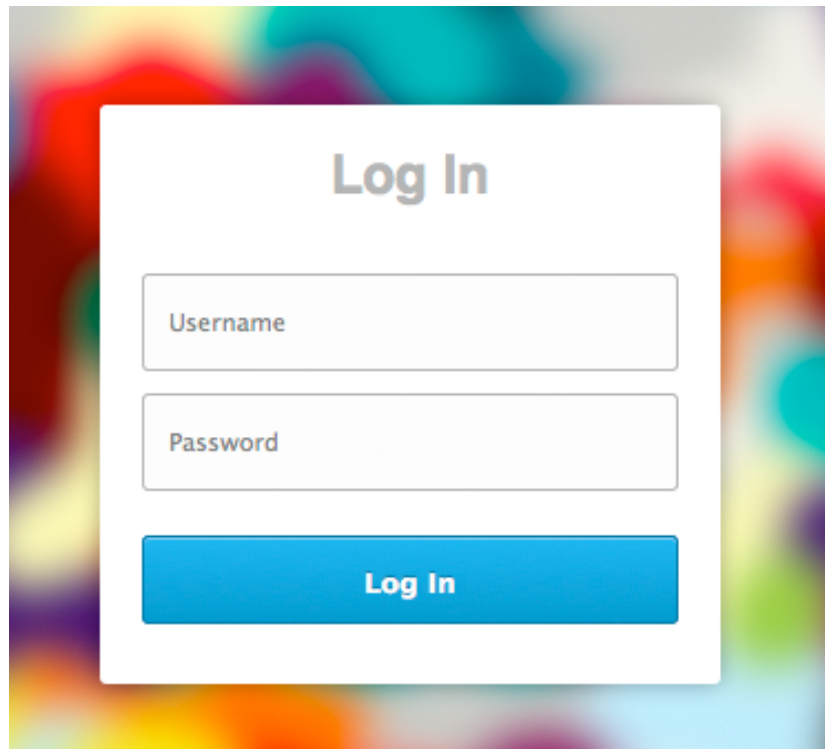
Use telnet to request the following pages and note the status code of response.

1. `http://home.sandiego.edu/~sat/nada.html`
2. `http://home.sandiego.edu/~sat/solutions.html`
3. `http://home.sandiego.edu/~sat` (no trailing '/')

Finally, try a “gibberish” request to `home.sandiego.edu` and note the response status code.

We've seen how to GET data from a server.

*How do we go about **sending** data?*

A login form titled "Log In" is centered on a white background. It features two input fields: "Username" and "Password", both with light gray borders. Below these fields is a blue button with the text "Log In" in white. The entire form is set against a blurred, colorful background.

HTTP provides two basic mechanisms for sending “form” data to a server.

### **GET method:**

- Input is uploaded in URL field of request line:

**`www.somesite.com/animalsearch?monkeys&banana`**

### **POST method:**

- Web page often includes form input
- Input is uploaded to server in request entity body

You should be able to differentiate between use cases for GET and POST.

In what scenarios would you use GET? POST?

	GET	POST
A.	Piazza post	Search terms, Take-out order
B.	Search terms, Take-out order	Piazza post
C.	Search terms	Piazza post, Take-out Order
D.	Piazza post, Search terms, Take-out Order	
E.		Piazza post, Search terms, Take-out Order