

CLIENT-SERVER ARCHITECTURE & HTTP

CLIENTS & SERVERS

- **Client requests a resource**
- **Server responds with resource**
- **These are *roles* — not technical specs or computer types**



CLIENTS & SERVERS





CLIENTS & SERVERS

DEAR ABBY:

My Dad Objects To a Pet Monkey

DEAR ABBY: I am 10 years old and my Daddy said that when I saved enough money I could buy anything I wanted with it.

All my life, I have wanted a monkey. I have saved \$14. I asked Daddy if I could buy a pet monkey and he said no, because I wouldn't know how to take care of it. My Mom is the fussy type. You know, everything has to be just so. Do you know anyone who has a pet monkey, and can give me some advice?

WANTS A MONKEY

DEAR WANTS: I have had two pet monkeys (David and Bathsheba) and, although I love monkeys, your father is right. To quote my son (he was 4 at the time), "Monkeys should live with monkeys, and people should live with people."



ABIGAIL
VAN BUREN



CLIENTS & SERVERS





CLIENTS & SERVERS



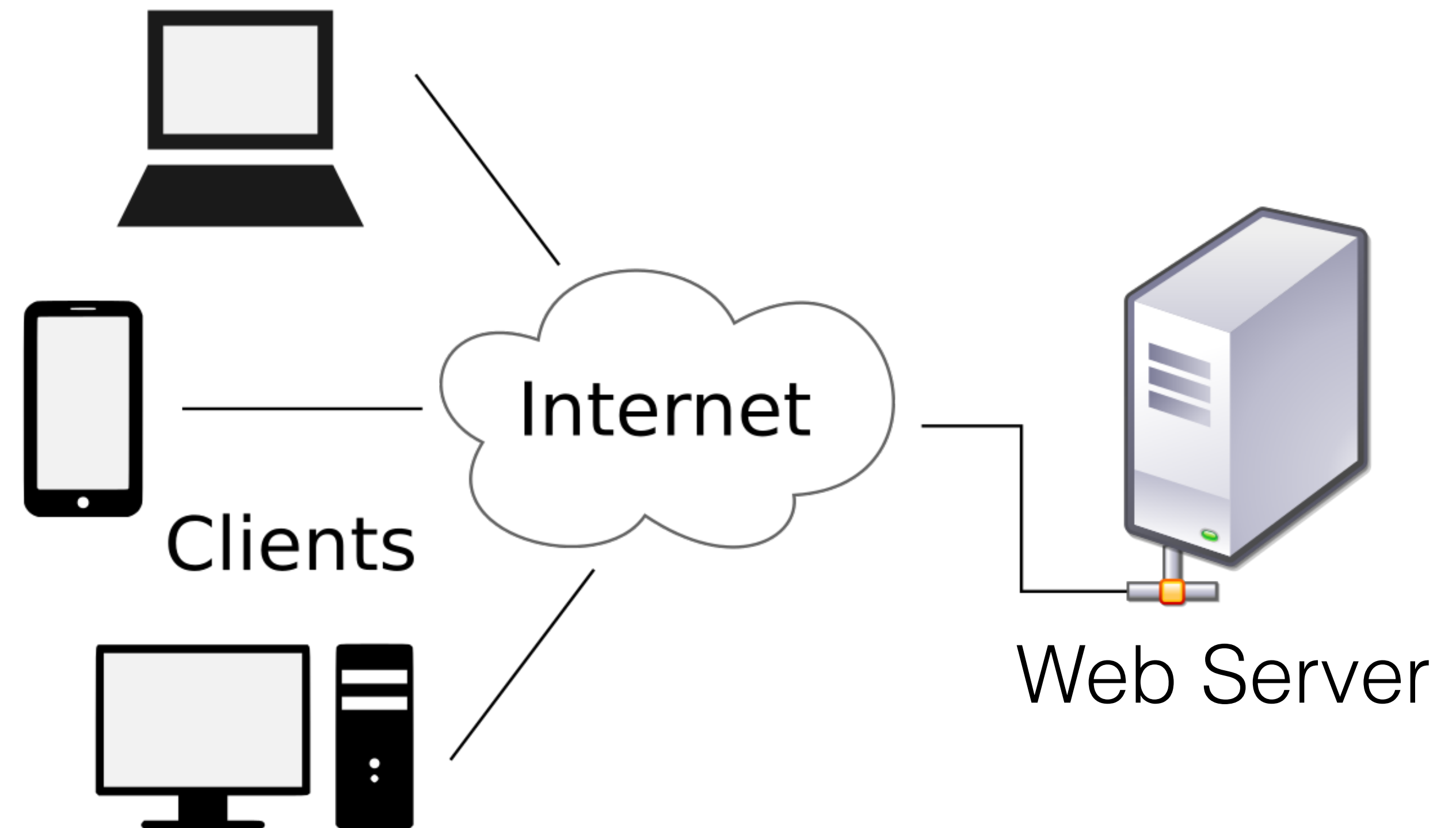


CLIENTS & SERVERS



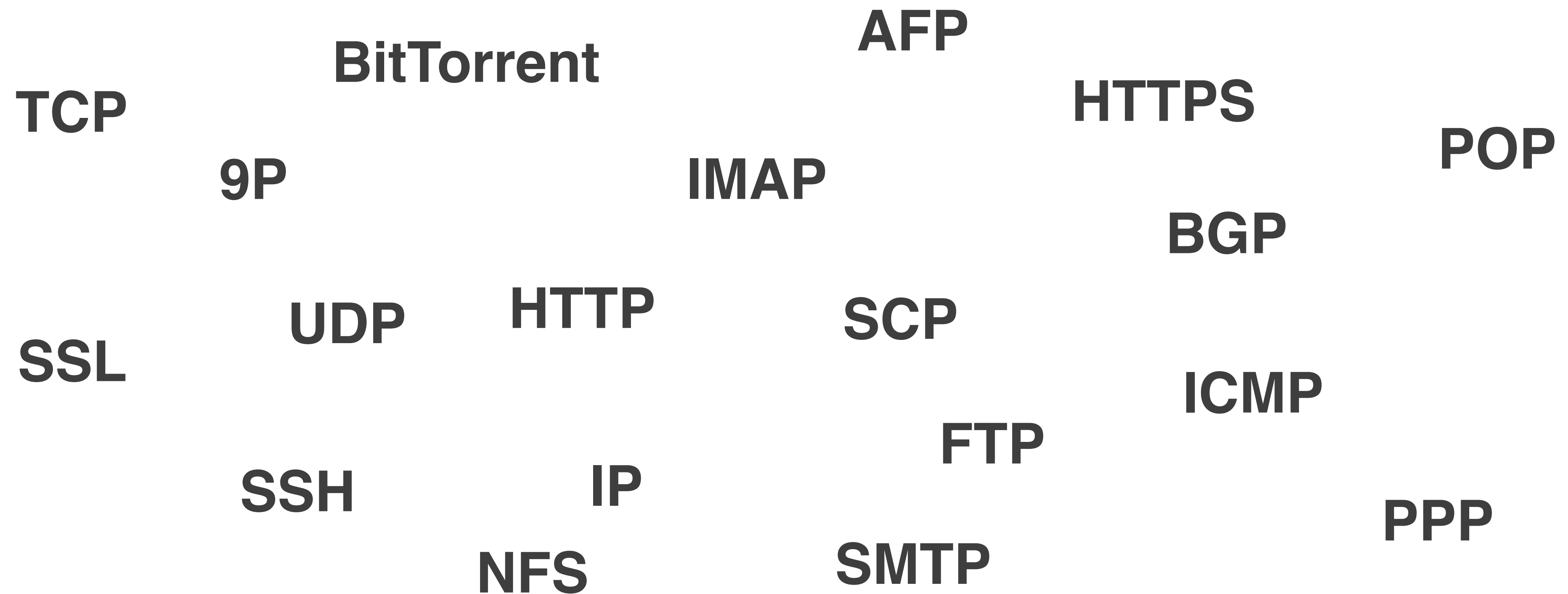
WEB SERVERS

- Processes (running programs) not physical machines
 - Might be running on a laptop,
 - or a Raspberry Pi,
 - or an enterprise-grade workstation...
- Listening on a port for incoming requests
- Send back responses





INTERNET COMMUNICATION PROTOCOLS



PROTOCOL

- **Rules for interaction / communication**
- **Specification, not implementation**

THE KNOCK-KNOCK MESSAGE PROTOCOL

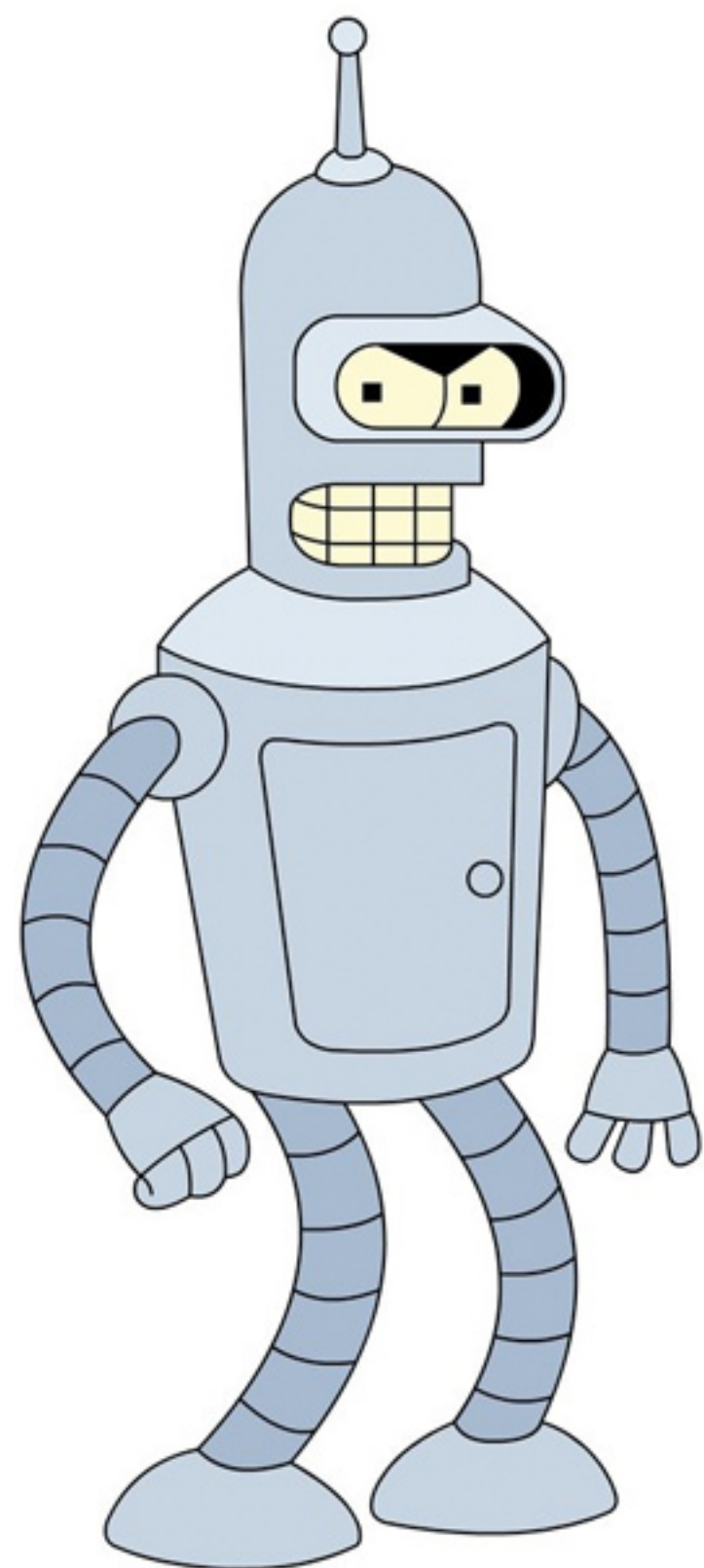
- Joker opens connection with "knock, knock."
- Victim completes handshake with "who's there?"
- Joker transmits identity label: "<IDENTITY>"
- Victim requests clarification: "<IDENTITY> who?"
- Joker delivers payload: "<PUNCHLINE>"
- Joke is now delivered, close connection. Participants may optionally laugh and/or dodge fists.

MESSAGING / APP VS. TRANSMISSION

- KnockKnock is an *application level* protocol
- It specifies the sequence and content of messages
- It does NOT specify how those messages are transmitted



KNOCK KNOCK OVER VOX



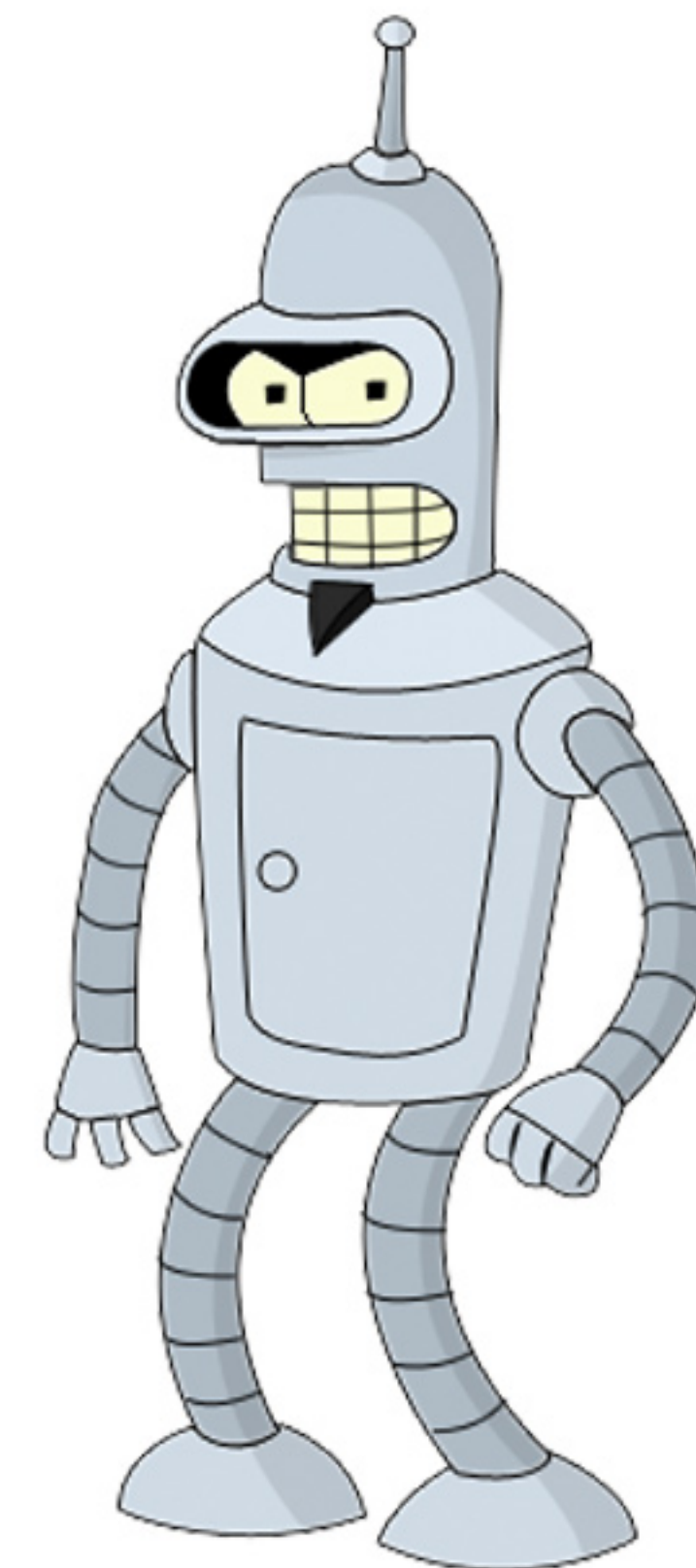
Knock, knock.

Broken state machine.

Knock, knock.

Who's there?

**Broken state machine,
who?**





KNOCK KNOCK OVER TEXT



KNOCK KNOCK OVER BLACKBOARD

Knock, knock...

who's there?

HTTP

HTTP

- An *application-level* communications protocol. You might call it a *messaging* protocol.
- Specifies allowable *metadata* and *content* of messages.
- Does **NOT** specify *how* messages are transmitted!
- STATELESS: does *not* need to remember previous req-res!

HTTP PROTOCOL

- RFC (Request For Comments) [7230 \(link\)](#)
- By the IETF (Internet Engineering Task Force)
- But a *generic* messaging protocol
 - *"HTTP is a generic interface protocol for information systems. It is designed to hide the details of how a service is implemented... independent of the types of resources provided."*

HTTP CLIENTS & SERVERS

◉ Example Clients

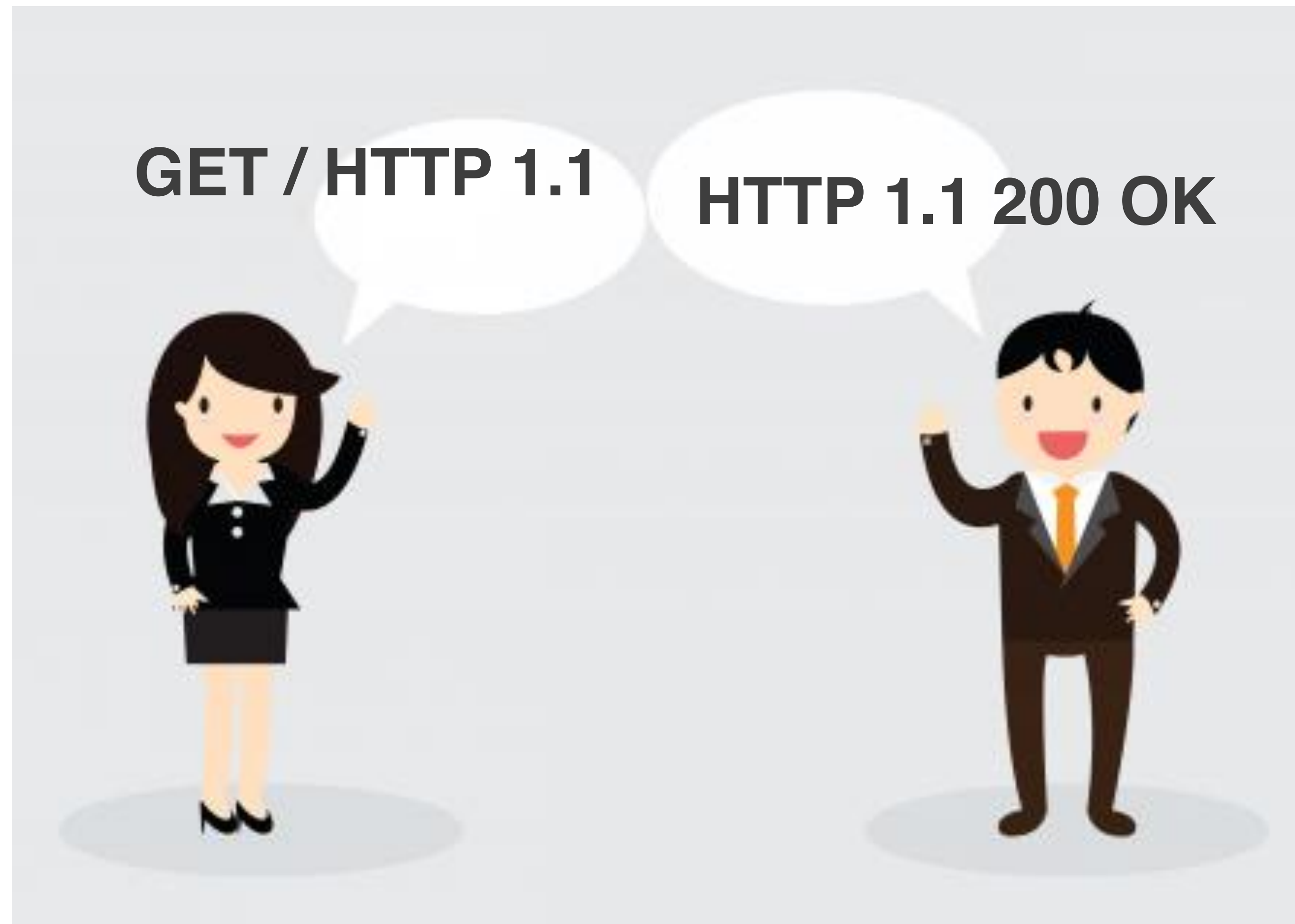
- web browsers
- household appliances
- stereos
- firmware update scripts
- command-line programs
- mobile apps
- communication devices

◉ Example Servers

- web servers
- home automation units
- ◉ networking components
- ◉ office machines
- ◉ autonomous robots
- ◉ news feeds
- ◉ traffic cameras

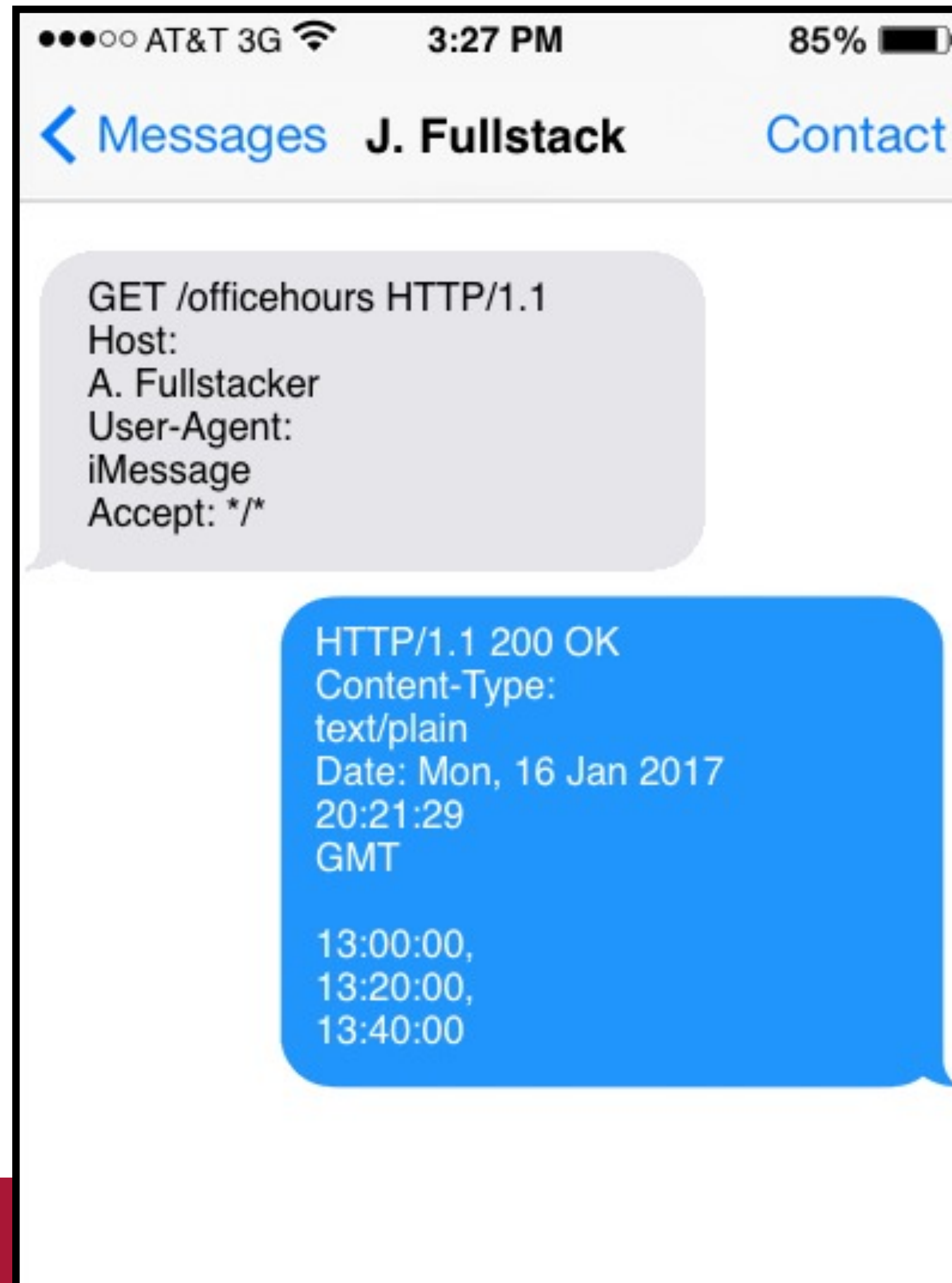
**NOT A TRANSMISSION
PROTOCOL!**

HTTP OVER VOX



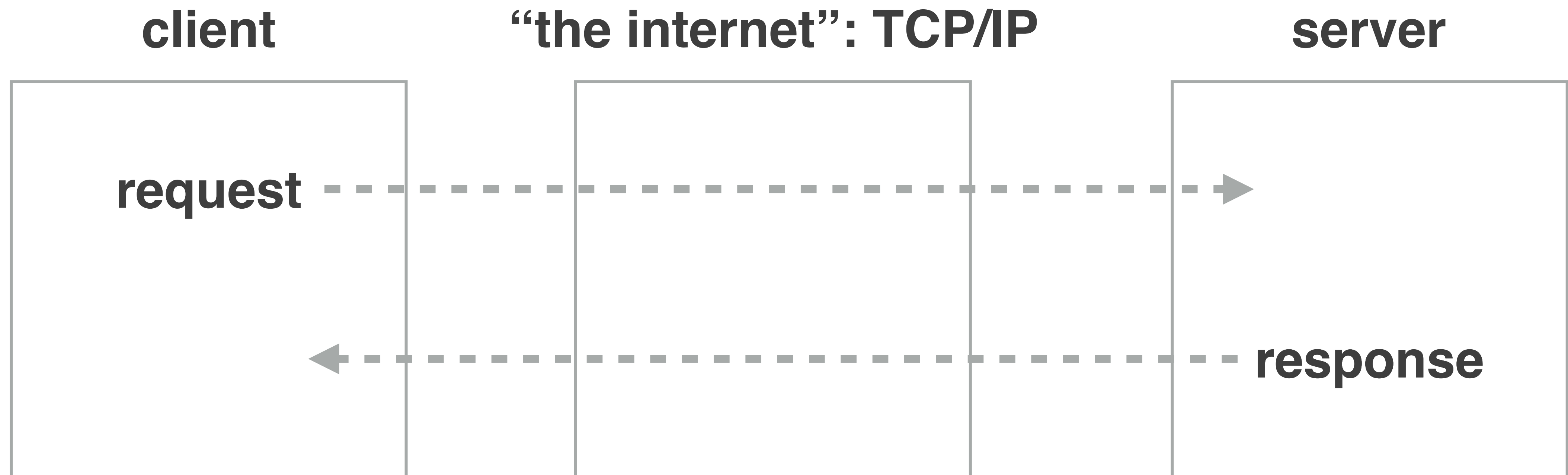


HTTP OVER TEXT





HTTP OVER TCP/IP







HTTP

Every request gets exactly one (total) response
(sometimes a response is broken up into chunks)



HTTP REQUEST

just a message with a certain format

verb URI

headers

```
POST /docs/1/related HTTP/1.1
```

```
Host: www.test101.com
```

```
Accept: image/gif, image/jpeg, */*
```

```
Accept-Language: en-us
```

```
Accept-Encoding: gzip, deflate
```

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

```
bookId=12345&author=Nimit
```

body

(from http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html)



COMMON VERBS

GET “read”

POST “create”

PUT “update”

DELETE “delete”



HTTP RESPONSE

status

headers

```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<html><body><h1>It works!</h1></body></html>
```

(from http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html)

payload/body



COMMON STATUSES

200	“OK”
201	“created”
304	“cached”
400	“bad request”
401	“unauthorized”
404	“not found”
500	“server error”

LAB