/EXPRESS.JS

"You think, you can"

INTERNET COMMUNICATION

AFP

BitTorrent

TCP

HTTPS

POP

9P

IMAP

BGP

HTTP

UDP

SCP

SSL

ICMP

FTP

SSH

IP

PPP

NFS

SMTP

HTTP

client "the internet" server
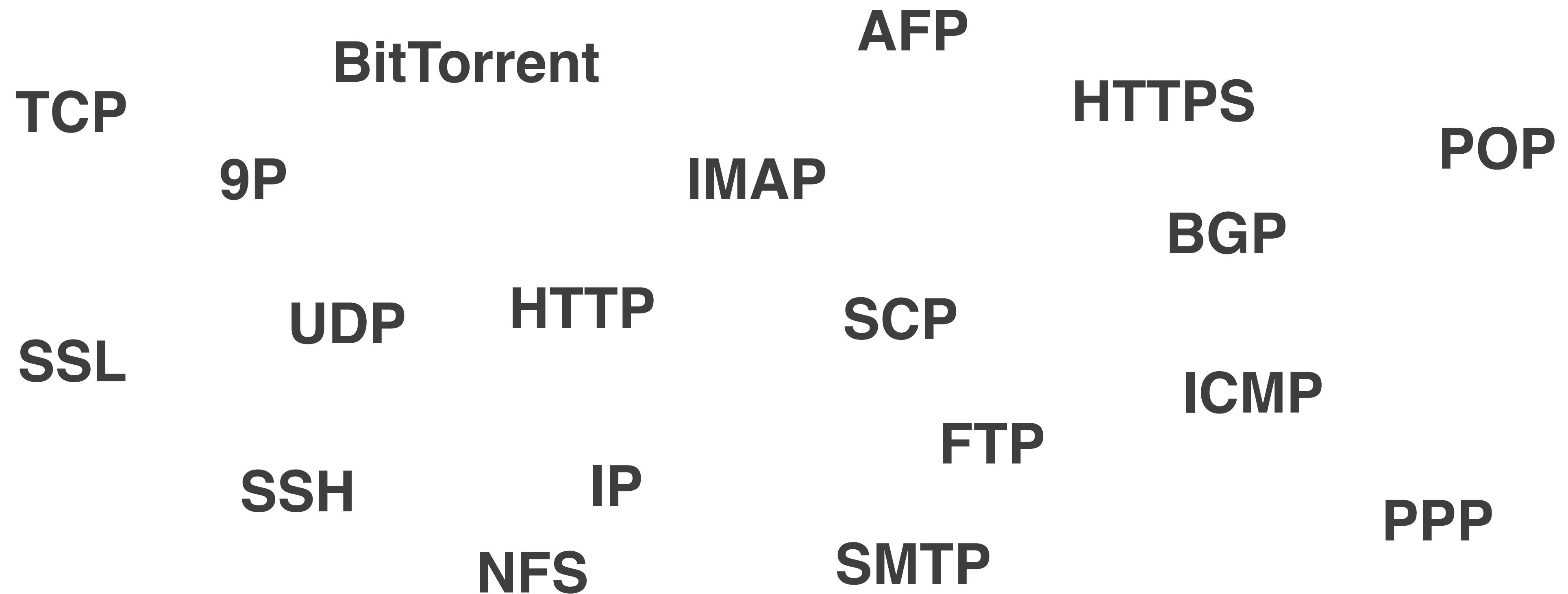
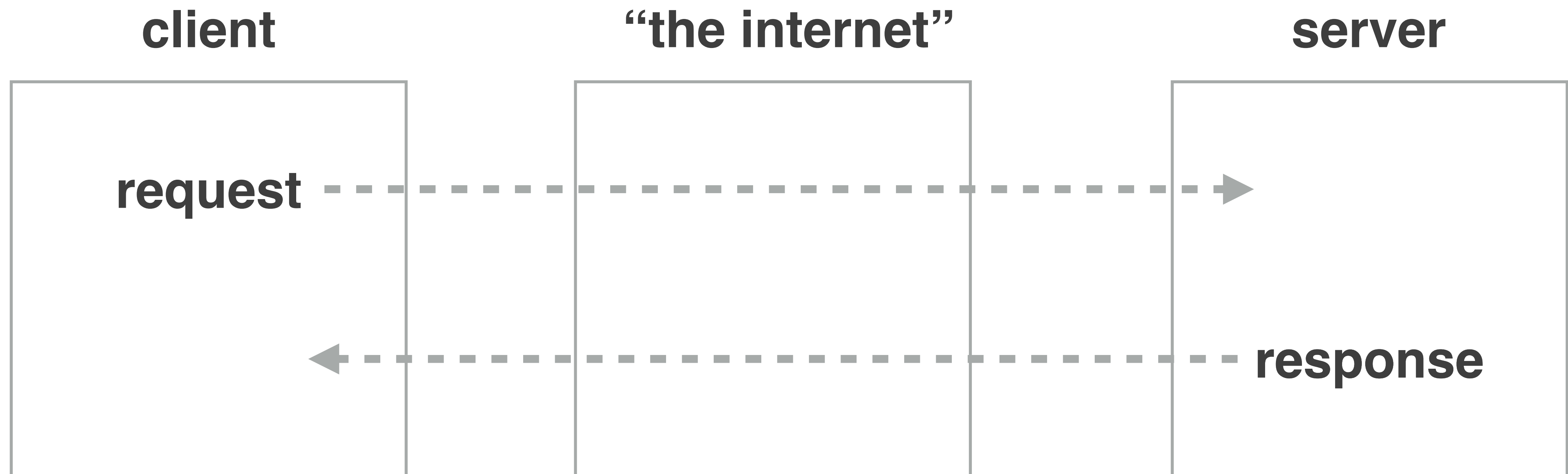request ⇢ ⇢

⇠ response

HTTP

Every request gets exactly one response

PROTOCOL

- ◎ **Rules**

- ◎ **Specification, NOT implementation**

- ◎ **Often used for communication**

node's `http` library is a not implementation of HTTP

HTTP REQUEST

just a message with a certain format

verb route

headers

body

```
POST /docs/index.html 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
```

(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

```
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>
```

headers

(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"

EXPRESS

A node library for request handling

<CODE>

</CODE>

EXPRESS

◉ **treats requests as objects, created by event**

◉ **matches on verb AND route**

◉ **allows chaining of many handlers**

◉ **enables modular layering with "routers"**

POP QUIZ

CLIENT

Something that makes (HTTP) requests

SERVER

Something that responds to (HTTP) requests

REQUEST

A formatted message sent over the network by a client. Contains VERB, URI (route), headers, and body.

RESPONSE

A server's reply to a request (formatted message). Contains headers, payload, and status.

REQUEST-RESPONSE CYCLE

*The client **always** initiates by sending a request, and the server completes it by sending **exactly one** response*

EXPRESS MIDDLEWARE

A function that handles responding to requests.
Of the form: `function(req, res, next){...}`

REQUEST QUERY STRING

A way to pass data from client to server.

verb route

```
POST /docs/index?x=123&foo=that HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-L
Accept-E
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT 5.1)

bookId=12345&author=Nimit
```

headers

In express…
request.query = {x:123, foo: 'that'}

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

body

REQUEST BODY

A way to pass data from client to server.

route

headers

```
POST /docs/index?x=123&foo=that HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
```

In express…

request.body = {bookId:12345, author: 'Nimit'}

bookId=12345&author=Nimit

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

body

REQUEST PARAMS

A variable portion of the URI.

ROUTER

A "layer" of route handlers (middlewares).

GOTCHAS

◎ **`app.get`** v **`app.get`**

◎ **routes are not file paths**

◎ **order matters**

◎ **`req.params`** v **`req.query`** v **`req.body`**

◎ **`app.use`** v **`app.all`**