Life of a Web Request

An Overview of Web Applications

Lecture Goals

- Understand the components of a web application.
- Understand what makes web applications different from regular applications.
- Have a ***vocabulary*** to talk about web applications.
- Understand why web programming is *hard*.

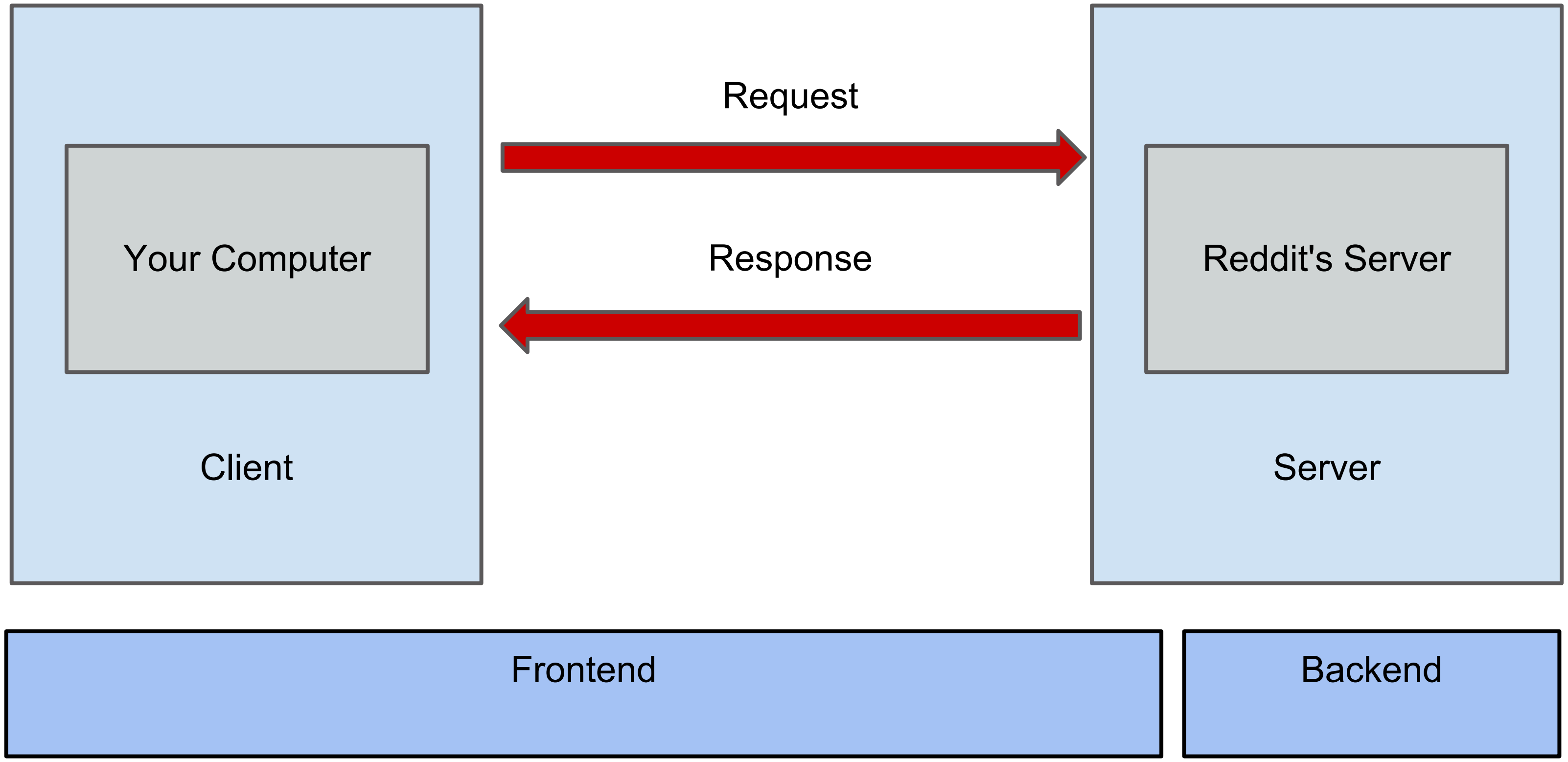
Motivating Example

What happens when you type

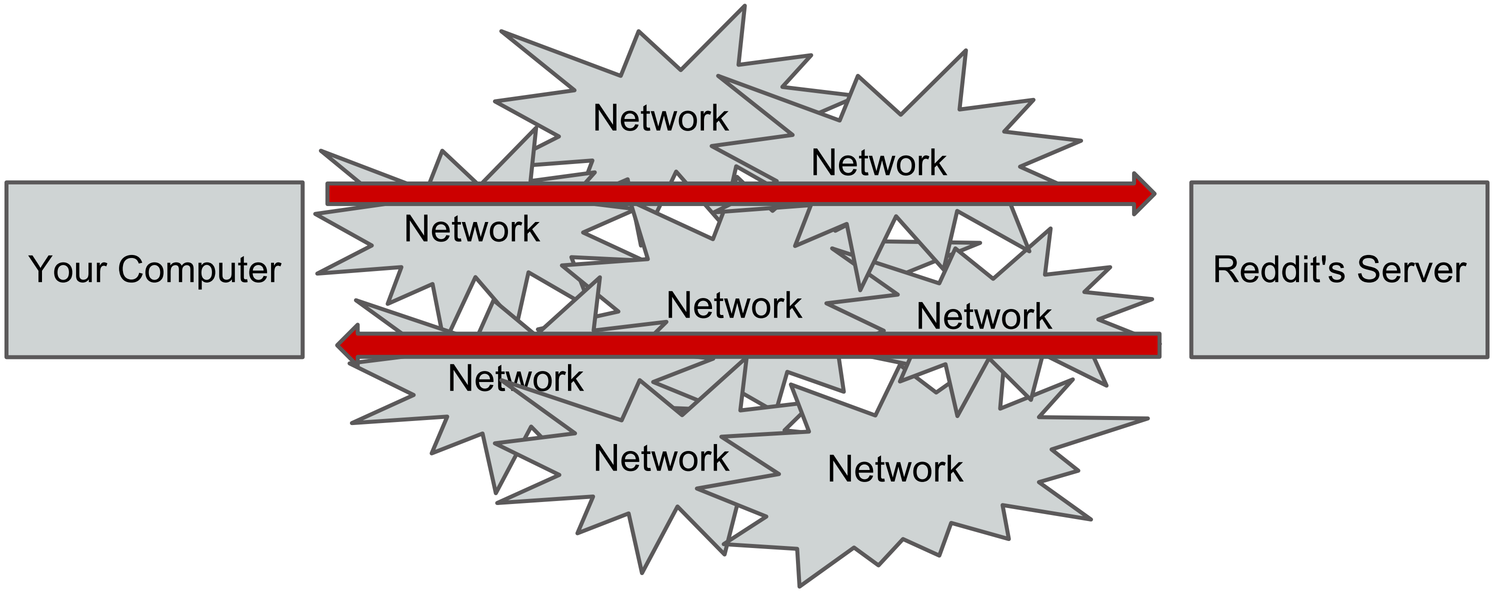
http://www.reddit.com/r/explainlikeimfive/

into your browser?

Motivating Example



Motivating Example



Requests and responses are ***routed*** through many different networks before reaching their destination.

A response and request may take very different paths to their destinations!

The Internet

A network of (heterogeneous) networks of machines communicating across different levels of abstraction.

The Internet (Example)

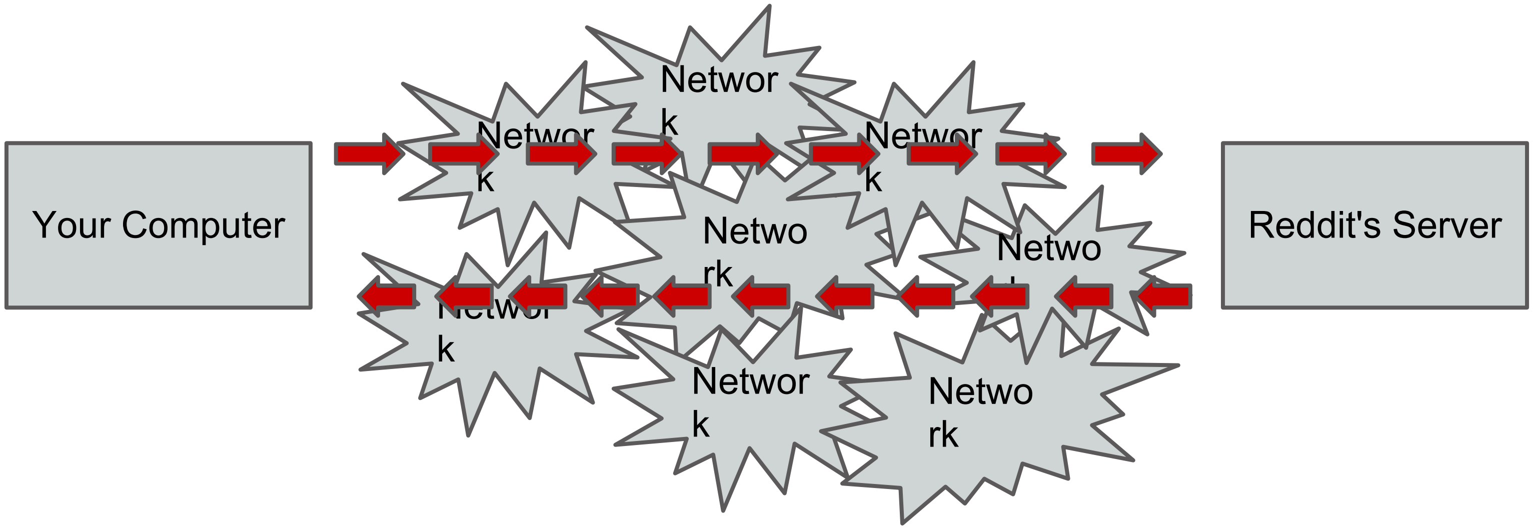
```
traceroute
```

A network of networks

- Networks differ in:
  - ***mediums***
  - ***protocols***

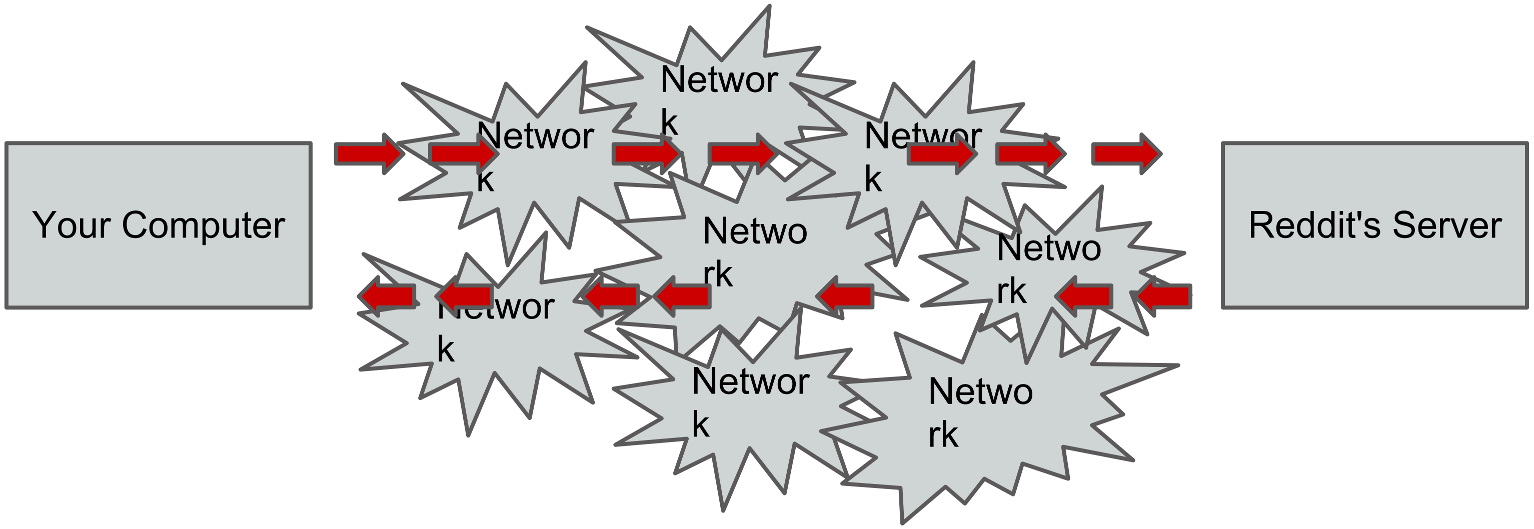
- Lots of different tools are used to send traffic within and across networks.

Packets



Requests and responses are broken into small ***packets*** of data. Each one may take a different route between client and server!

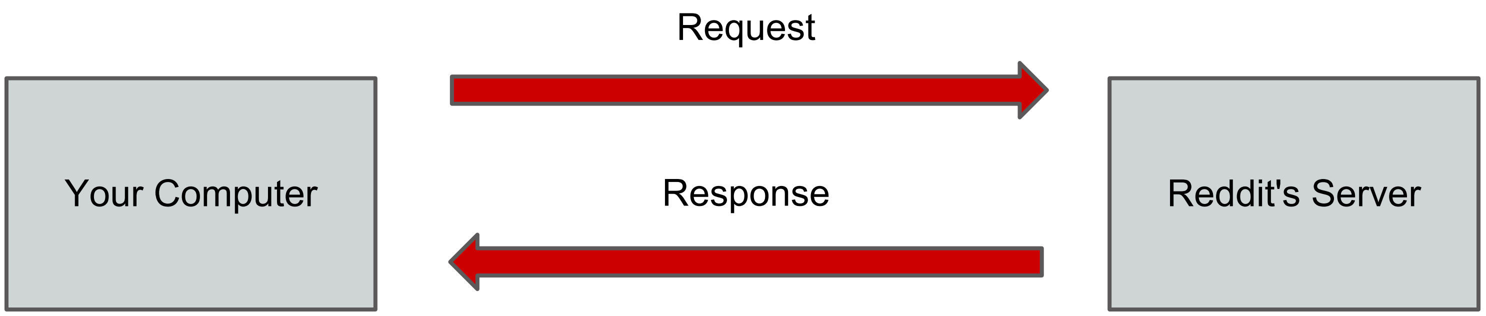
Packet Loss



Packets are lost along the way. Internet protocols can ensure that lost data is resent, as well as confirm its receipt.

Motivating Example

Lots of this interaction we get for free.



Lets us think of a web application as just the client and the server.

Addresses on the Internet

How do the individual packets know where to go?

Packets use ***IP Addresses*** (**I**nternet **P**rotocol) to get to their destination.

But we don't remember IP addresses, we remember ***URLs*** (**U**niversal **R**esource **L**ocator).

Addresses on the Internet

- **URL**: `www.reddit.com/r/explainlikeimfive`
- **IP Address**: `173.194.73.99`
- ***DNS*** (**D**omain **N**ame **S**ervice) is a protocol that allows machines to translate URL's into IP addresses.
- DNS enables websites to link URLs to specific IP addresses by ***DNS Resource Records***.

Addresses (Example)

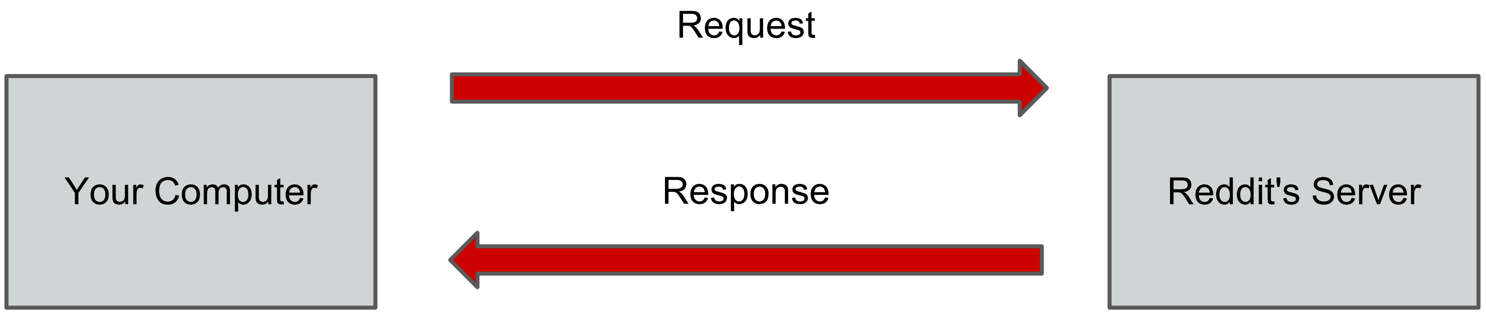
```
nslookup
```

Web Requests

- A ***web request*** is a request for a web page.

- ***HyperText Transfer Protocol*** (HTTP) is the language used to issue and respond to web requests.

HyperText Transfer Protocol



- HTTP is the language used to communicate between clients and servers.

- HTTP contains ***verbs*** for distinguishing between different types of requests:

`GET`, `PUT`, `POST`, `DELETE` (and others...)

HTTP Verbs

- `GET` - request a web page

  e.g. `GET http://www.reddit.com/r/todayilearned`

- `POST` - send some resource data to a website

  e.g. make a new post on reddit

- `PUT` - update a resource on a website

  e.g. make an edit to a post

- `DELETE` - remove a resource from a website

  e.g. delete a post

HTTP Verbs (Example)

```
telnet
```

State and HTTP

HTTP is a ***stateless*** protocol. We use ***cookies***, ***URL variables*** as well as other methods to save state.

State is any *stored* information that may change over time.

`http://www.google.com/search?`**`q=reddit;`**

Web Application Components

LAMP Stack

- **L**inux - an operating system; to run the applications
- **A**pache - a web server; to serve ***web requests***
- **M**ySQL - a database; to access/store data
- **P**HP - a scripting language; to add ***dynamic content***

Web Servers

What they do:

- Routing requests
- Generating responses
- Storing/updating/deleting data
- Sending responses

Generating Responses

Responses come in three major forms:

- ***HTML*** (**H**yper**T**ext **M**arkup **L**anguage)
  - for humans
  - CSS *(**C**ascading **S**tyle **S**heets) are how we make HTML pretty.*
- ***XML*** (e**X**tensible **M**arkup **L**anguage)
  - for computers
- ***JSON*** (**J**ava**S**cript **O**bject **N**otation)
  - for computers

HTML/CSS (Example)

`http://www.reddit.com/r/todayilearned`

XML/JSON

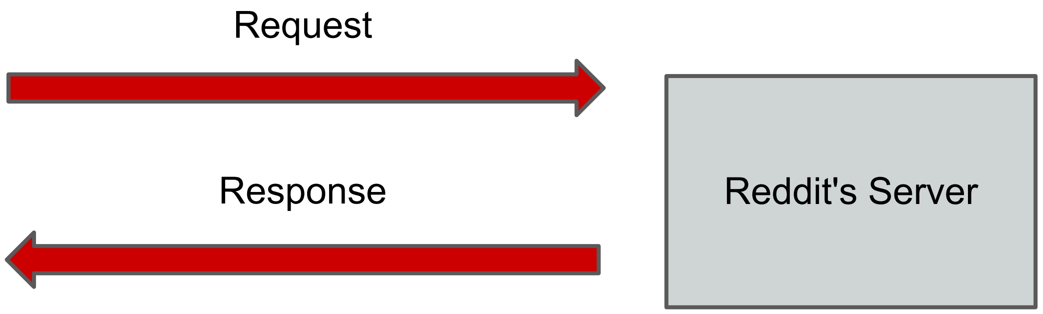
- Servers don't always generate responses for humans, but also for ***browsers***.
- JSON/XML are ways to markup data.
- They make it easy to describe ***structured data***.
- A big part of the ***Web 2.0*** world (more on this later).

JSON (Example)

https://api.stackexchange.com/2.1/questions?order=desc&sort=activity&site=stackoverflow

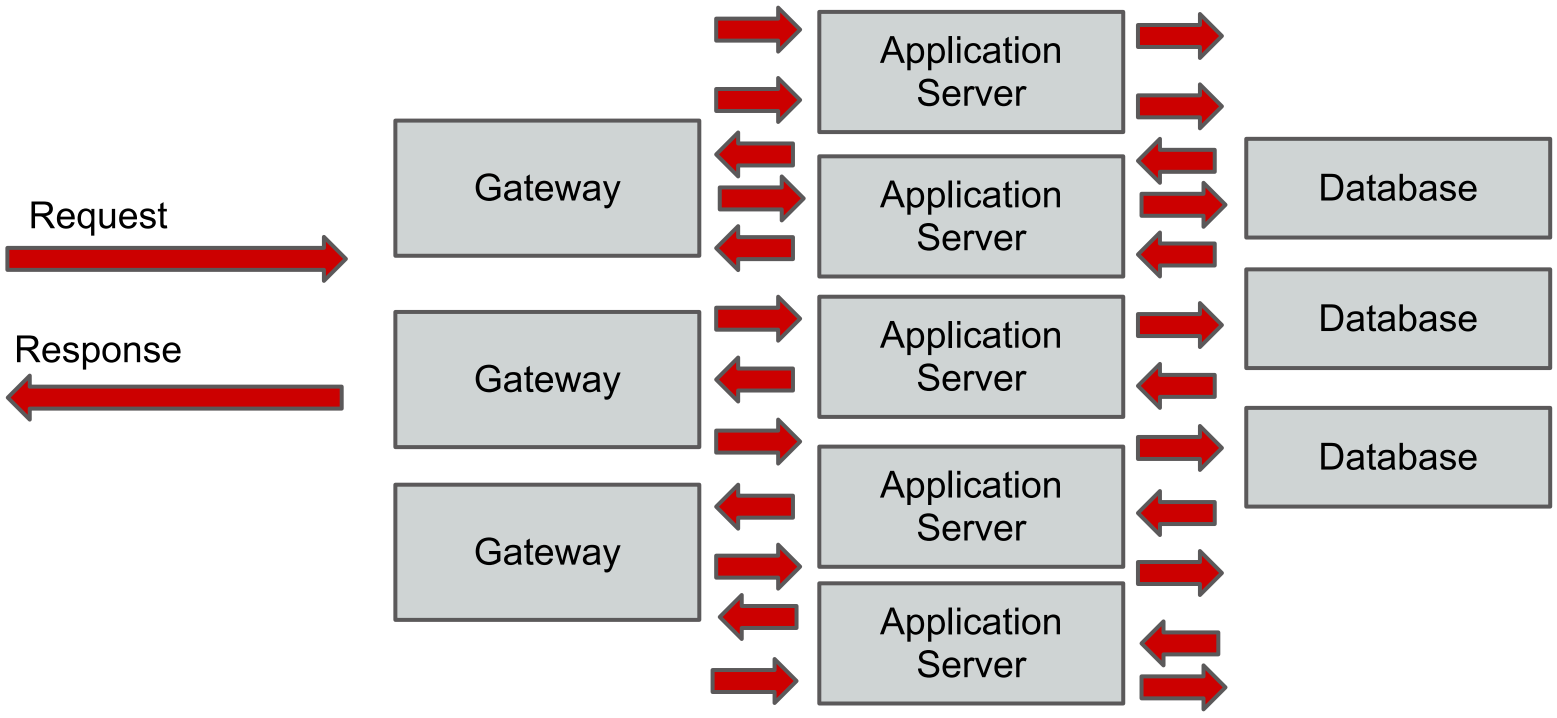
http://jsoneditoronline.org/

Web Servers



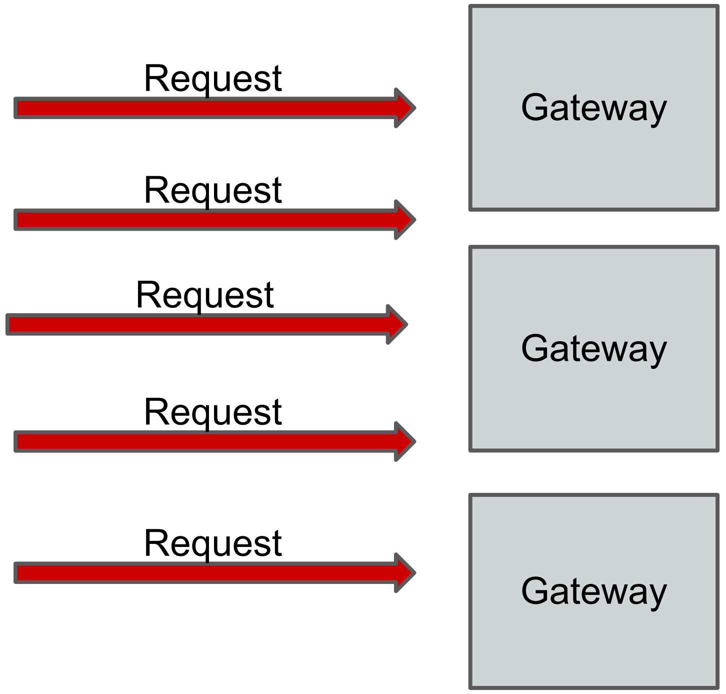
Reddit (and any other ***to scale*** web application) doesn't simply have one server for all requests!

Web Servers



The responsibility of ***serving a web request*** is distributed across many different computers.

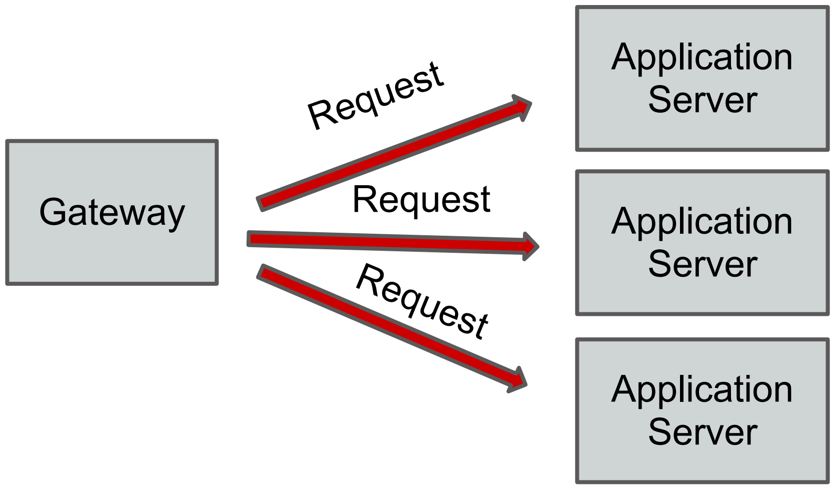
Gateways



Gateways are very simple web servers that forward requests to many different servers.

Focus on minimizing the ***load*** on any one particular application server.

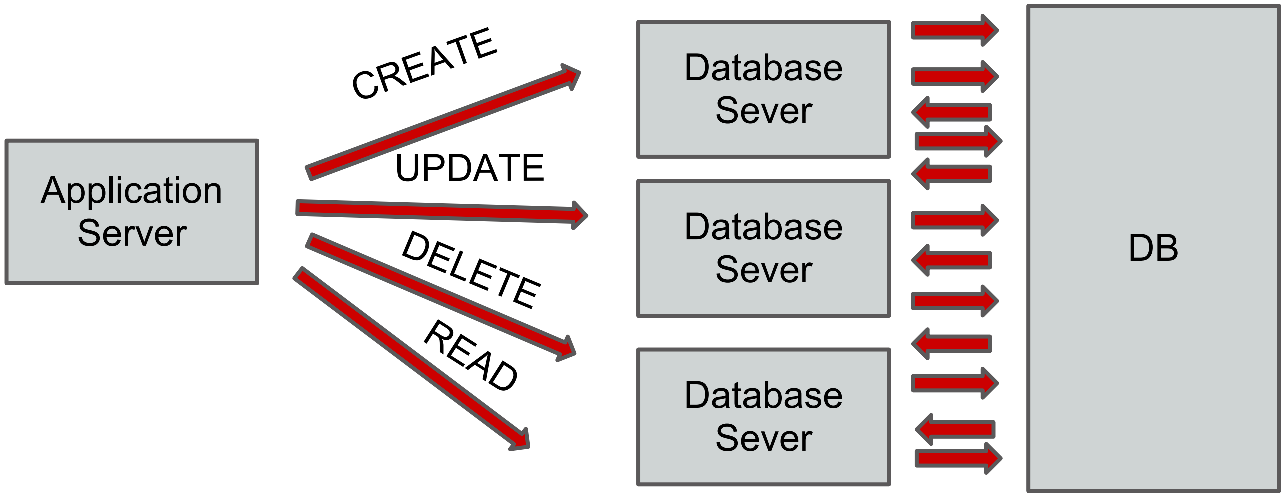
Application Servers



Application servers are where the real work is done. They handle any required data processing, communication with databases, and generate the ***web response*** (often, a web page).

Examples: Apache, Nginx, Tornad, *etc.*

Databases



Application servers keep track of state (e.g. user accounts, forum posts, likes) by ***C**reating*, ***R**eading*, ***U**pdating* and ***D**eleting* (***CRUD***) data stored as ***records*** in a ***database***.

Database servers serve as gateways to a database, and help reduce the latency of response.

Databases

Databases use the techniques of ***ACID*** to provide efficient and safe data storage.

- ***A**tomicity*
- ***C**onsistency*
- ***I**solation*
- ***D**urability*

Transactions are individual updates to a database.

SQL

- One of the standard languages for interacting with a database is ***SQL***.
- SQL can be thought of as *a programming language for data*.
- SQL Databases: MySQL, SQLite, Oracle, *etc*.

SQL (Example)

```
CREATE TABLE users (
    name STRING;
    email STRING;
);

INSERT INTO users
    (name, email)
    VALUES
    ("Samuel Messing",
    "sbm2158@columbia.edu")
;
```

SQL (Example)

```
SELECT * FROM users WHERE
name="Samuel Messing";
```

Alternatives to SQL

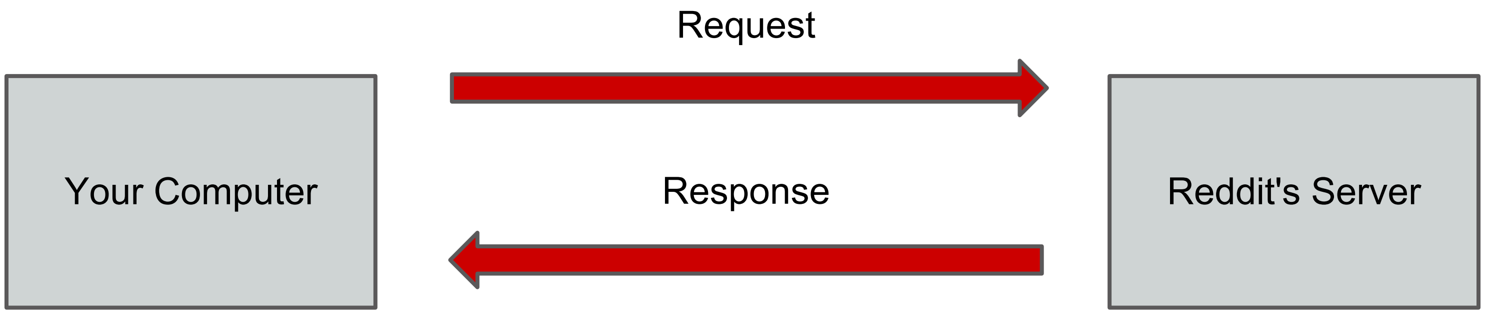
- With lots of data, SQL databases may be too slow (because of ACID guarantees).
- Performance can increase by relaxing constraints, or using different ***data structures***.
- NoSQL Databases: BigTable, Cassandra, MongoDB

Dynamic Content

What do we mean by ***Web 2.0***?

- ***AJAX*** - **A**synchronous **J**avaScript **A**nd **X**ML
- A mechanism to add ***dynamic content*** to a webpage by making your browser do more.

Dynamic Content

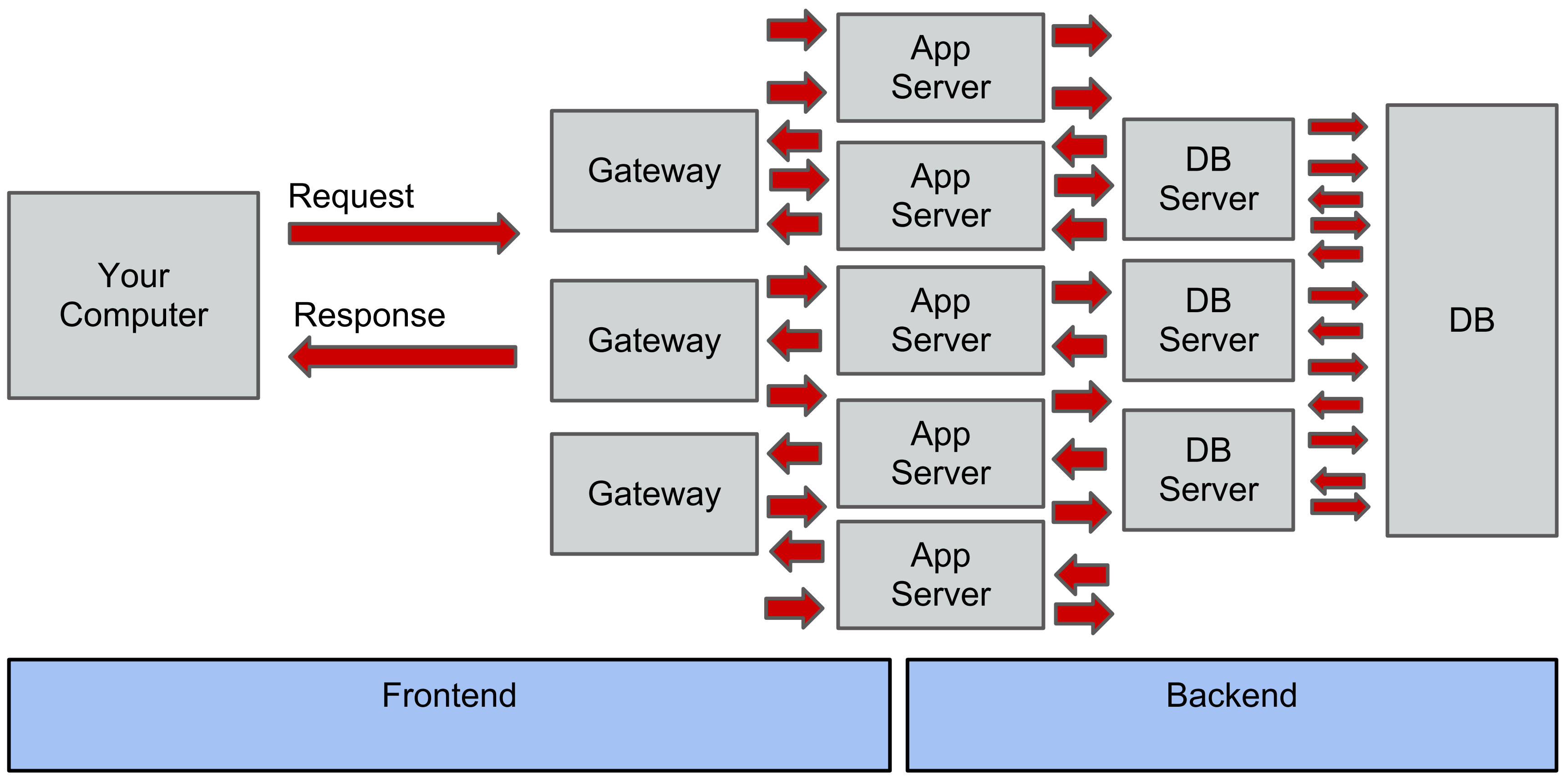


- Network speeds are **slow** (~ seconds)
- Want websites to be **fast**
- Make more use of ***client-side*** code to make websites feel snappy.

Dynamic Content (Example)

`http://www.twitter.com/`

Web Applications



Not Covered In This Lecture

- ***Security*** - This is a HUGE topic!
- ***Network Protocols*** - How traffic actually moves around the web.
- ***Web Standards*** - Rules governing how browsers work, making life easier for web programmers.
- ***Browser Differences*** - Often need to do special work for individual browsers (mostly IE).
- ***Frameworks*** - frameworks package the elements of web applications together, making development easier.

Thank you!

Email me your questions:
`sbm2158@columbia.edu`

Samuel Messing

Appendix

5 layers of abstraction

1. Application (*e.g.* Siri, iTunes, Netflix, ESPN, *etc.*)
2. Transport (*e.g.* TCP, UDP, *etc.*)
3. Internet (*e.g.* IP)
4. Link (*e.g.* Ethernet, ARP, *etc.*)
5. Physical (*e.g.* Radio Waves, Light, *etc.*)

traceroute

```
smrz@shannon ~ $ traceroute www.reddit.com
traceroute: Warning: www.reddit.com has multiple addresses; using 24.143.194.72
traceroute to a659.b.akamai.net (24.143.194.72), 64 hops max, 52 byte packets
 1  192.168.1.1 (192.168.1.1)  2.023 ms  0.798 ms  0.665 ms

 2  cpe-24-193-240-1.nyc.res.rr.com (24.193.240.1)  70.091 ms  35.690 ms  36.584
ms

 3  tenge-0-2-0-6-nycmnyr-rtr01.nyc.rr.com (24.168.135.169)  13.153 ms  12.531 ms
32.064 ms

 4  bun120.nycmnytg-rtr001.nyc.rr.com (184.152.112.63)  24.370 ms  20.170 ms
26.382 ms

 5  bun6-nycmnytg-rtr002.nyc.rr.com (24.29.148.250)  19.841 ms  24.548 ms  20.725
ms

 6  107.14.19.24 (107.14.19.24)  34.406 ms  21.037 ms  15.915 ms
```

nslookup

```
smrz@shannon ~ $ nslookup www.google.com
Server:        209.18.47.61
Address:  209.18.47.61#53

Non-authoritative answer:
Name:     www.google.com
Address: 74.125.131.104
Name:     www.google.com
Address: 74.125.131.105
Name:     www.google.com
Address: 74.125.131.106
Name:     www.google.com
Address: 74.125.131.147
Name:     www.google.com
Address: 74.125.131.99
Name:     www.google.com
Address: 74.125.131.103
```

telnet

```
smrz@shannon ~ $ telnet www.google.com 80
Trying 173.194.73.99...
Connected to www.google.com.
Escape character is '^]'.
GET /
HTTP/1.0 200 OK
Date: Fri, 22 Feb 2013 00:00:03 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: PREF=ID=129ce702bc7f058d:FF=0:TM=1361491203:LM=1361491203:S=AUai-
bic6LBsTy41; expires=Sun, 22-Feb-2015 00:00:03 GMT; path=/; domain=.google.com
Set-Cookie: NID=67=F-6NTXvo1LD4d0Wz2C2LWK0naKKVnq-
toR6sBjEogn66anIttZnQl_PVZuiY_MH02PdwzwbekINCsN858IRU6k-CdBWhm0x7-
qVNDGpNv7ghS5m_ZtcvpHce8DlHYs7Y; expires=Sat, 24-Aug-2013 00:00:03 GMT; path=/;
domain=.google.com; HttpOnly
P3P: CP="This is not a P3P policy! See http://www.google.
com/support/accounts/bin/answer.py?hl=en&answer=151657 for more info." [...]
```

telnet (continued)

```
smrz@shannon ~ $ telnet 173.194.73.99 80
Trying 173.194.73.99...
Connected to vb-in-f99.1e100.net.
Escape character is '^]'.
GET /search?q=reddit;
[...]
<div id="topstuff"></div><div id="search"><div id="ires"><ol><li class="g"><h3
class="r"><a href="/url?q=http://www.reddit.com/&amp;sa=U&amp;
ei=i7UmUajuFoaM0QH_o4G4AQ&amp;ved=0CBgQFjAA&amp;
usg=AFQjCNFPuyjH5ywh1eXiS8K9E9sfVx1mxA"><b>reddit</b>: the front
page of the internet</a></h3><div class="s"><div class="kv" style="
margin-bottom:2px"><cite>www.<b>reddit</b>.com/</cite><span class="flc"> - <a
href="/url?q=http://webcache.googleusercontent.com/search%3Fq%3Dcache:
Sq6ykWCuNUMJ:http://www.reddit.com/%252Breddit%253B%26hl%3Den%26ct%3Dclnk&amp;
sa=U&amp;ei=i7UmUajuFoaM0QH_o4G4AQ&amp;ved=0CBkQIDAA&amp;usg=AFQjCNHtjAOT7-
YsmM74IrpkwKiOrNm3ow">Cached</a>
[...]
```

rails console

```
create_table "listeners" do |t|
  t.string   "name"
  t.string   "email"
  t.datetime "created_at", :null => false
  t.datetime "updated_at", :null => false
end
```

rails console (continued)

```
1.9.3-p374 :001 > me = Listener.new(name: "Samuel Messing",
email: "sbm2187@columbia.edu");


1.9.3-p374 :002 >   me.save
   (0.1ms)  begin transaction
  Listener Exists (0.2ms)  SELECT 1 AS one FROM "listeners"
WHERE LOWER("listeners"."email") = LOWER('sbm2187@columbia.edu')
LIMIT 1
  SQL (2.9ms)  INSERT INTO "listeners" ("created_at", "email",
"name", "updated_at") VALUES (?, ?, ?, ?)  [["created_at", Fri,
22 Feb 2013 17:30:36 UTC +00:00], ["email", "sbm2187@columbia.
edu"], ["name", "Samuel Messing"], ["updated_at", Fri, 22 Feb
2013 17:30:36 UTC +00:00]]
   (0.8ms)  commit transaction
 => true
1.9.3-p374 :003 >
```