Networking Crash Course

CC BY
Matt Terwilliger

Before We Start

Client/Server Model

Client requests information from **server** over pre-established **protocols**.

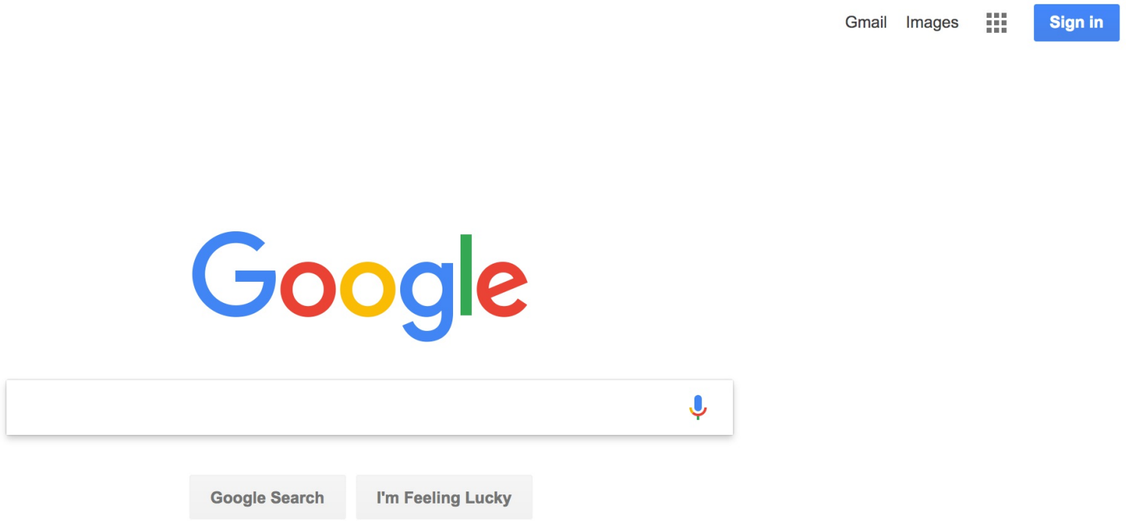
TCP/IP Model

Application Layer

Transport Layer

Internet Layer

Link Layer



Client?

Server?

Protocol?

Digging deeper

Let's use a tool we've seen before:

```
curl http://www.google.com/
```

Digging deeper

Let's use a tool we've seen before:

```
curl http://www.google.com/
```

```
Connected to www.google.com (172.217.0.36) port 80 (#0)
```

- `www.google.com` - what we typed in
- `172.217.0.36` - ???

DNS

```
dig www.google.com
```

DNS

```
dig www.google.com
```

```
dig +trace www.google.com
```

- Pattern: "I don't know the answer, but I know who does"
- Recursive/distributed approach
  - Limits data each server is required to store
  - No single source of truth for the entire Internet (redundancy)
  - Easier to manage
- Caching

Digging deeper

```
Connected to www.google.com (172.217.0.36) port 80 (#0)
```

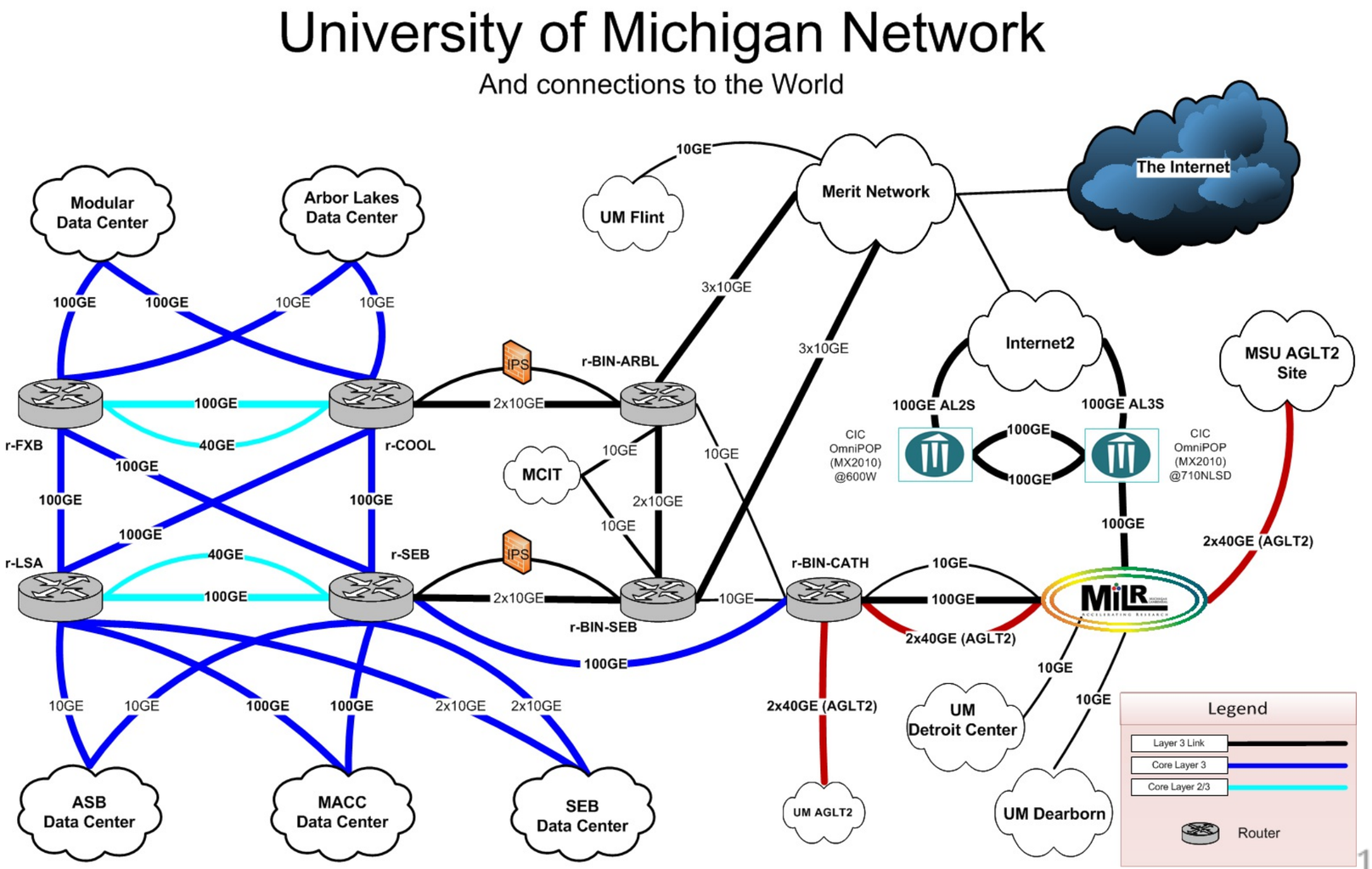
- `www.google.com` - what we typed in
- DNS translates `www.google.com` to `172.217.0.36`
- How does my computer (client) get to `172.217.0.36` (server)?

Routing

```
traceroute www.google.com
```

Routing

```
traceroute www.google.com
```



Routing (continued)

System maintains routing table

```
route -n get www.google.com
```

Routing (continued)

System maintains routing table

```
route -n get www.google.com
```

What is en0?

```
system_profiler
```

Digging deeper

```
Connected to www.google.com (172.217.0.36) port 80 (#0)
```

- `www.google.com` - what we typed in
- DNS translates `www.google.com` to `172.217.0.36`
- Computer uses routing table to find `172.217.0.36`
- Missing a layer (Transport)

Ports

An IP uniquely identifies an interface.

- Why do we need ports?

Ports

An IP uniquely identifies an interface.

- Why do we need ports?

TCP vs UDP

Two popular Transport Layer protocols (but not the only ones!)

TCP: More guarantees (ordering, best-effort delivery attempt)

UDP: Less overhead (Fast)

Ports

An IP uniquely identifies an interface.

- Why do we need ports?

TCP vs UDP

Two popular Transport Layer protocols (but not the only ones!)

TCP: More guarantees (ordering, best-effort delivery attempt)

UDP: Less overhead (Fast)

Different use cases

- Farther down the model we go the dumber the protocols are
- Not everything has the same requirements
  - Gaming service might not care if some data is lost
  - Websites obviously would

Ports

An IP uniquely identifies an interface.

- Why do we need ports?

TCP vs UDP

Two popular Transport Layer protocols (but not the only ones!)

TCP: More guarantees (ordering, best-effort delivery attempt)

UDP: Less overhead (Fast)

Different use cases

- Farther down the model we go the dumber the protocols are
- Not everything has the same requirements
  - Gaming service might not care if some data is lost
  - Websites obviously would

What about what we just saw?

HTTP: TCP port 80

DNS: UDP port 53

Ports (continued)

Q: What makes a server a server?

Ports (continued)

Q: What makes a server a server?

Client **connects** to a **listening** server.

Ports (continued)

Q: What makes a server a server?

Client **connects** to a **listening** server.

Simple example

Server:

```
nc -l 9999
```

Client:

```
nc 127.0.0.1 9999
```

What about HTTP?

Back to our cURL example

Request:

```
> GET / HTTP/1.1
> Host: www.google.com
> User-Agent: curl/7.49.1
> Accept: */*
```

What about HTTP? (continued)

Back to our cURL example

Response:

```
< HTTP/1.1 200 OK
< Date: Wed, 07 Dec 2016 06:54:09 GMT
< Expires: -1
< Cache-Control: private, max-age=0
< Content-Type: text/html; charset=ISO-8859-1
< P3P: CP="This is not a P3P policy! See https://www.google.com/support/accounts
/answer/151657?hl=en for more info."
< Server: gws
< X-XSS-Protection: 1; mode=block
< X-Frame-Options: SAMEORIGIN
< Set-Cookie: NID=91=srTc7LxMuO_1keewbJvEnV6-ck0Q_GZRtdQmfhGaWQmVCS4L6e2aCuNxky8
i2hDPZwdqbZ2PkA9QFsU3GIOAArpsqPp8mBzr3UqOEc8BiD5V_GTYpVXrqnNw9Ew6XZKsNQYaIy6Tbpr
b-Q; expires=Thu, 08-Jun-2017 06:54:09 GMT; path=/; domain=.google.com; HttpOnly
< Accept-Ranges: none
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
```

Try it

- Open a web browser
- Let's find the minimum set of information in a valid HTTP request

Try it

- Open a web browser
- Let's find the minimum set of information in a valid HTTP request

```
nc -l 9999
```

- **Note:** Not port `80`. Why?
- `<C+d>` to signal end of input

Try it

- Open a web browser
- Let's find the minimum set of information in a valid HTTP request

```
nc -l 9999
```

- **Note:** Not port `80`. Why?
- `<C+d>` to signal end of input

Request:

```
> GET / HTTP/1.1
```

Response:

```
< HTTP/1.1 200 OK
```

Wrap Up

So much more

- SSL/TLS
- DHCP
- ARP
- IPv6
- NAT
- Firewalls
- etc.

Friday