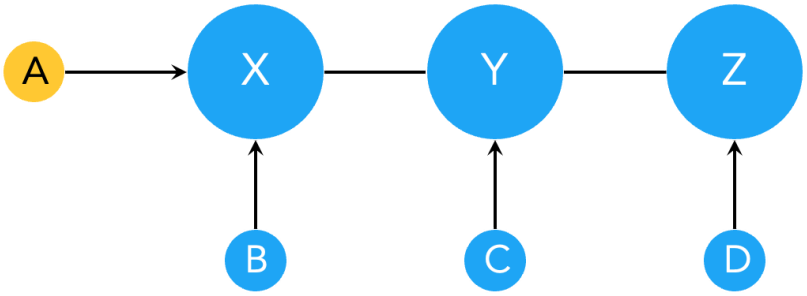
Lecture 12: BGP wrapup, ports

| Advertised by… | Export to… |
|---|---|
| Customer | Everyone |
| Peer | Customers only |
| Provider | Customers only |

**Warmup**: Given the following AS relationships, which ASes will A know about?

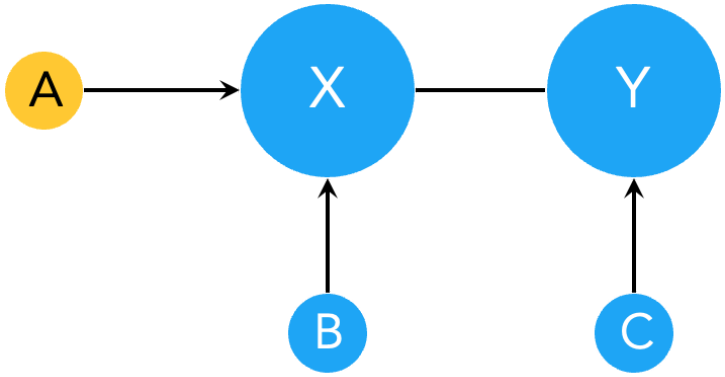



Customer ("A is customer of X")

Peer

**Warmup II:  warmup strikes back**

What happens if C suddenly starts advertising A's prefix?



## Prefix hijacking

**Who "owns" an IP prefix?**

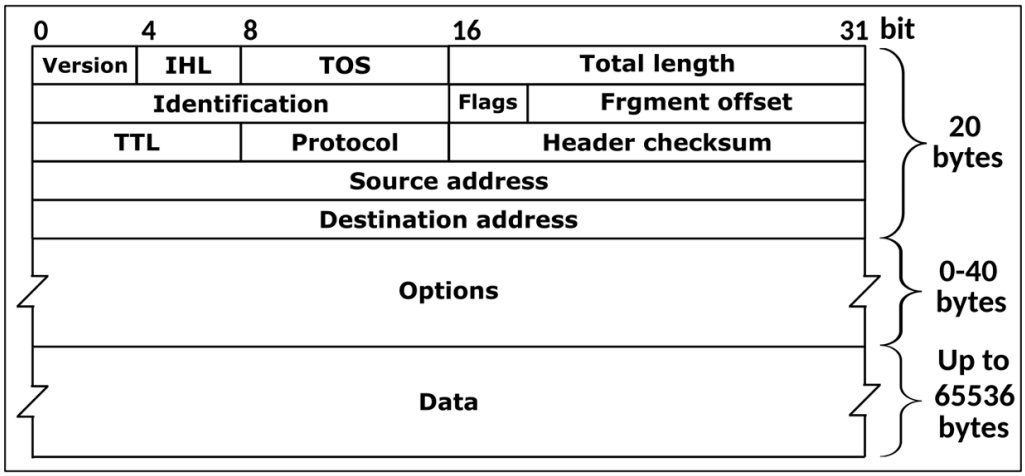
Allocated by internet authorities, hierarchically:

- Top-level:  Internet Assigned Numbers Authority (IANA)
- Regional Internet Registries
- Internet Service providers

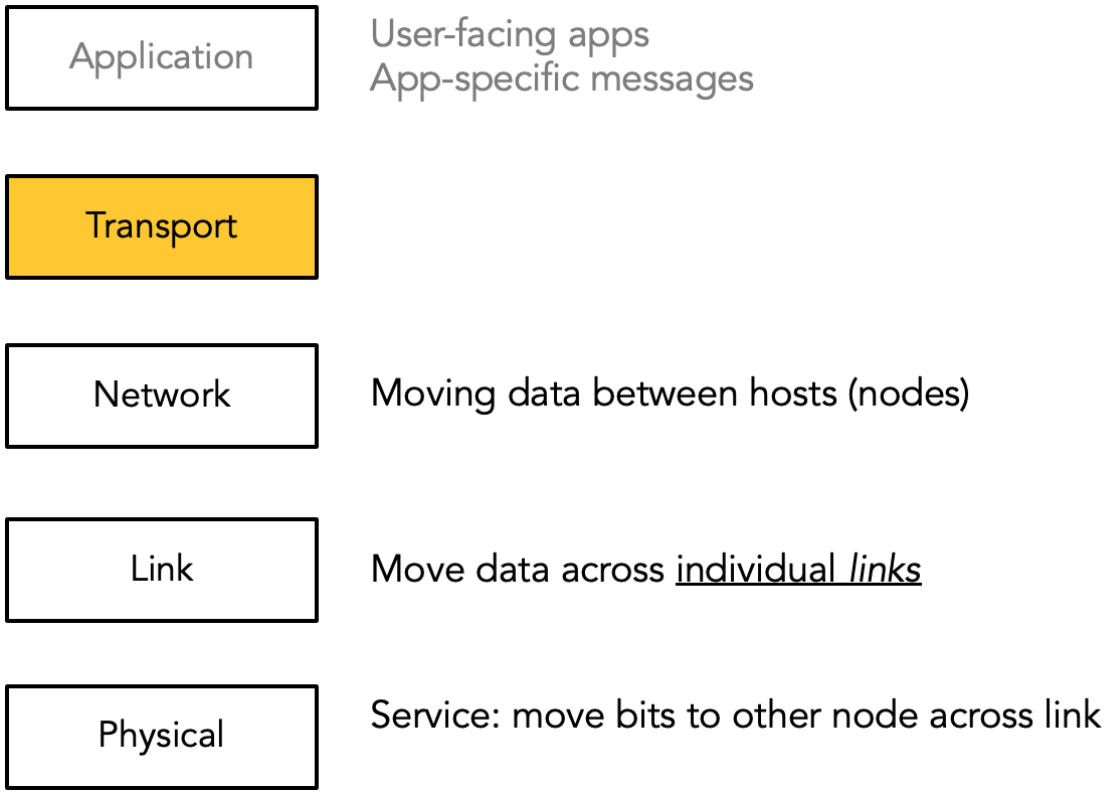
# Next unit:  Transport layer

The story so far:  network layer

The IP header:



Where do we go from here?




Two major types of transport protocols:

**TCP**: "reliable, connection-oriented byte stream"

**UDP**: "unreliable datagram service"

Key concept for today: port numbers

**Sketch:  how to support multiple applications?**

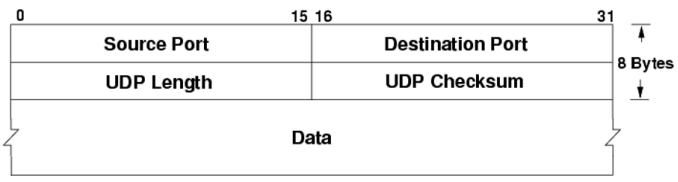
## Transport layer:  the goal

Use port numbers to multiplex connections using the same IP

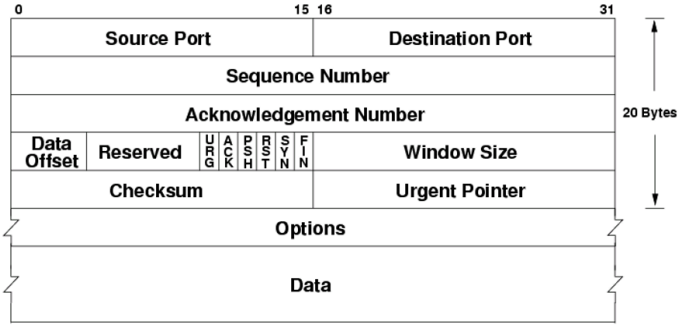
## What's a port number?

- 16-bit unsigned integer, 0-65535
- port < 1024:  "Well known port numbers"
- port > 20000 (usually):  "ephemeral ports", for general app use

UDP header




TCP header



This might seem familiar...

Client

```
func main() {
      conn, err := net.Dial(“tcp”, “127.0.0.1:5000”)
      // . . .
      conn.Write(...)

}
```

Server

```
func main() {
      listenConn, err := net.Listen(“tcp”, “127.0.0.1:5000”)
      for {
         clientConn, err := listenConn.Accept()

         go handleClient(clientConn)
      }
}
```

How connections work:  a pre-TCP sketch

| Local | | Remote | | |
|---|---|---|---|---|
| **IP** | **Port** | **IP** | **Port** | **State** |
| | | | | |
| | | | | |

| Local | | Remote | | |
|---|---|---|---|---|
| **IP** | **Port** | **IP** | **Port** | **State** |
| | | | | |
| | | | | |

Demo:  netstat

Port scanning