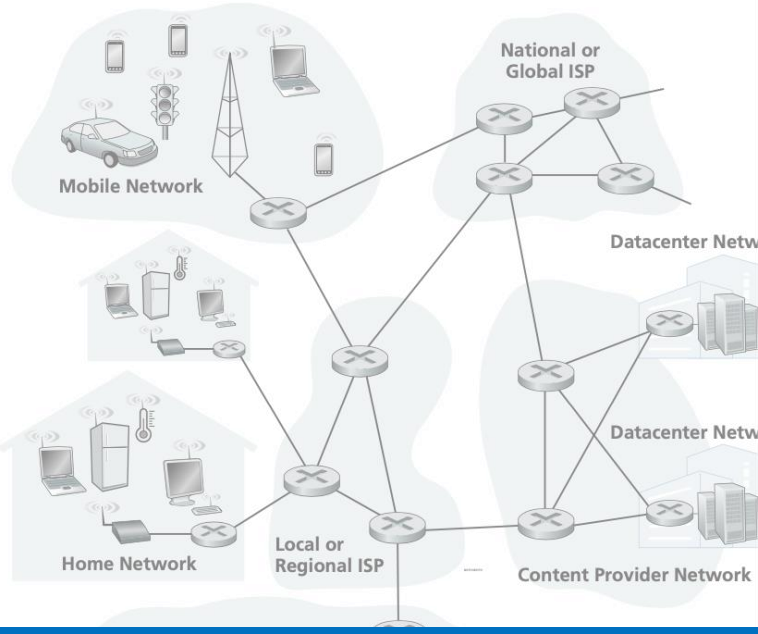
ICS 第 0.0.0.13 次小班课

Class 16

Fall 2023, Dec 13th, 2023



Schedule for Today

- **Review of Network Programming**

  回课：网络编程

- **A Network Tour: Starting From the Thin Waist**

  从网络栈的“细腰”出发谈网络

- **Exercises**

  巩固练习

A Network Tour: Preliminaries

网络漫游

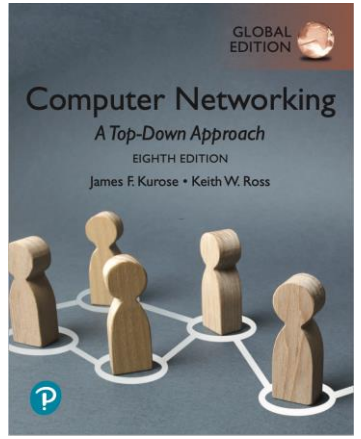
Why this tour?

There is always *more* to every topic in ICS. However, **Chap 11** of CSAPP is probably the only chapter that leaves you *aware of that*.

- Chap 11 focuses more on the Programming aspect of the network, i.e. *how* over *why*.
- Hopefully this tour will supplement the missing big picture.

Why start from the Thin Waist?

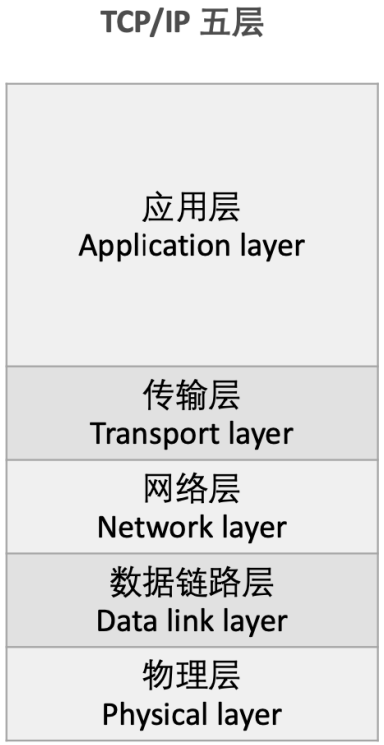
- Common practice is *top-down* or *bottom-up*.
- However, starting from the thin waist means starting from a more familiar zone…



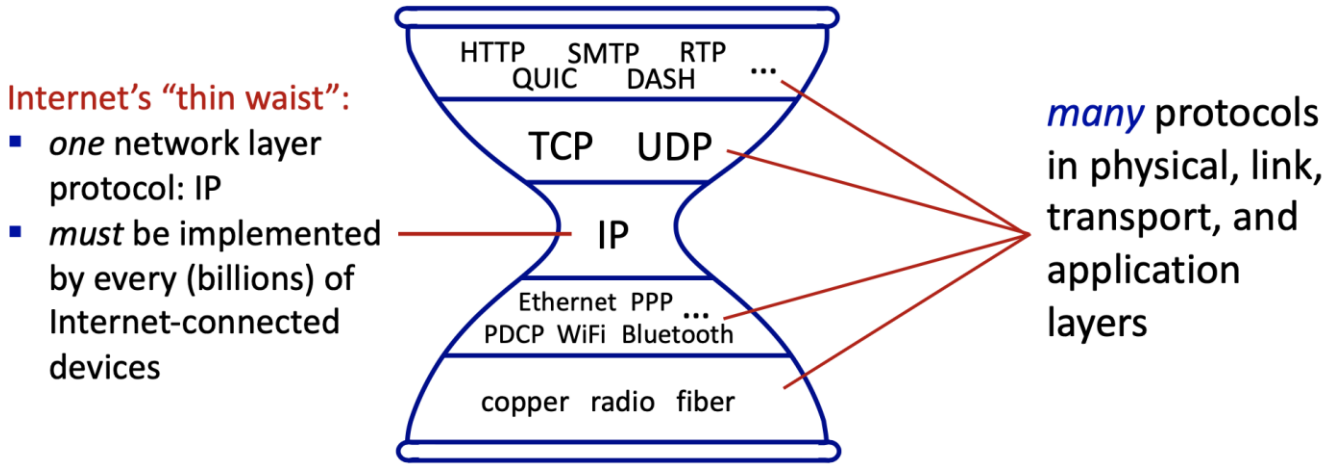
A Network Tour: Preliminaries
网络漫游

Before we start…

You should know of the concept of **protocol stack**, and the thin waist of it.



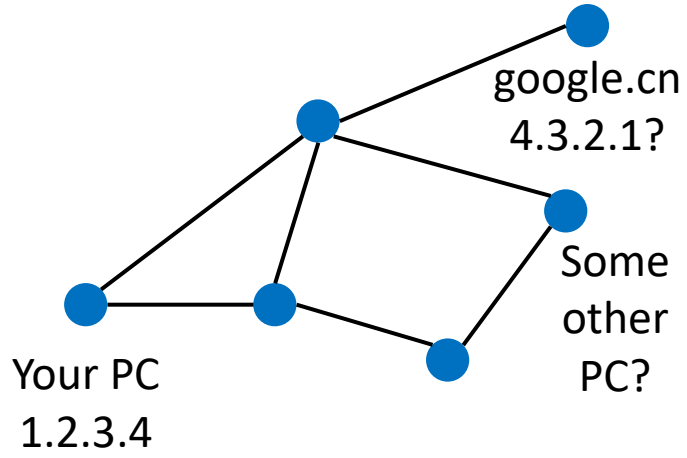
- Network functionality is too complex to implement all at once.
- The industry takes the classical **layered** approach.



A Network Tour: Starting from IP

网络漫游：从IP开始

- It is unlikely that you have never heard of **IP/IP address** (or jokes about it).



- But what do you *actually* know about IP, beside that you seem to be able to *find stuff* on the Internet with an **IP address**?
- And do you conceptualize the network as the graph above, where each node is labeled with an IP address?

A Network Tour: Network Layer

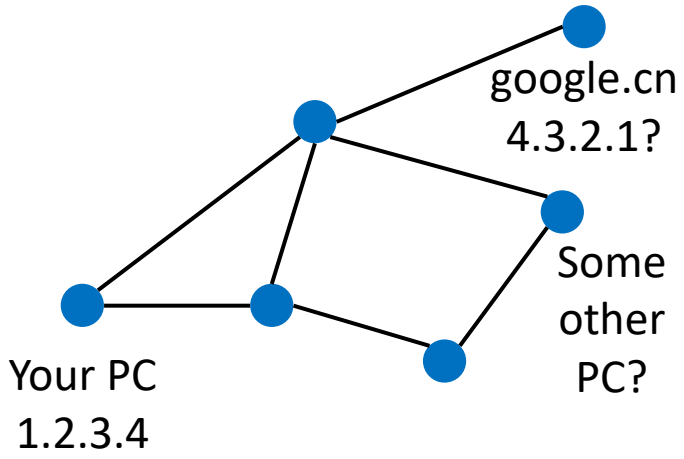
网络漫游：网络层

- The first point is actually accurate: **IP address** is for *locating things in a network*.

  IP is the thin waist, because we generally need only one way of addressing.

- **IP (Internet Protocol)** is a **network layer** protocol. It specifies how network interfaces in a network are *addressed*, and how said addresses can be *reached* via **routing**.

- However, the graph view on the right needs a few tweaks to be right…

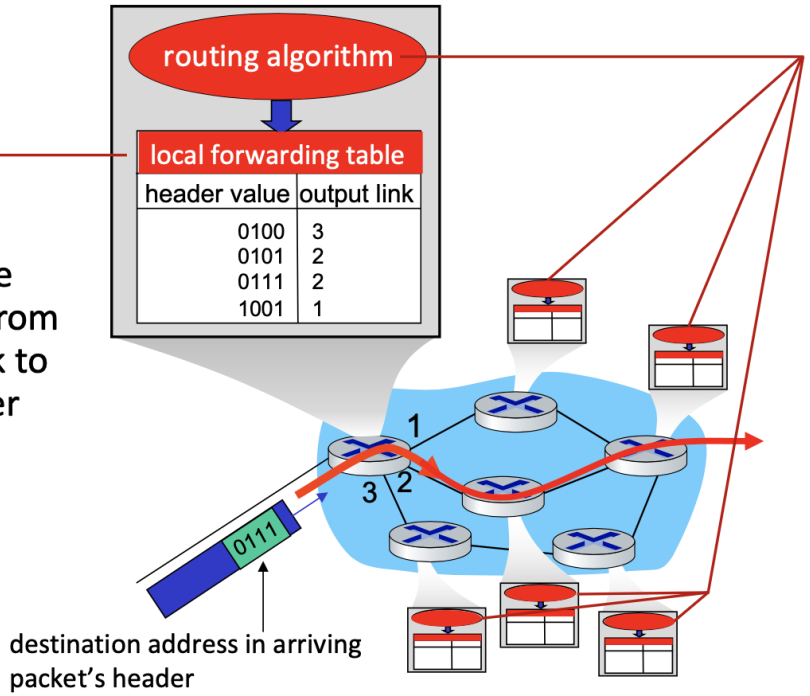


A Network Tour: Network Layer
网络漫游：网络层

A. Routers & Hosts

Think of your wireless router at home, and your phones/laptops as hosts!

Internet is not just *computers on a graph*. Most of the nodes are in fact **routers** running routing algorithm for reachability, while hosts operate at endpoints.



Forwarding:

- aka "switching"
- *local* action: move arriving packets from router's input link to appropriate router output link

Routing:

- *global* action: determine source-destination paths taken by packets
- routing algorithms

1. Routers run routing algorithms to obtain **routing tables**. Think of it like "IP x to port y" entries.

2. And they forward incoming packets according to the table, ***with a buffer***.

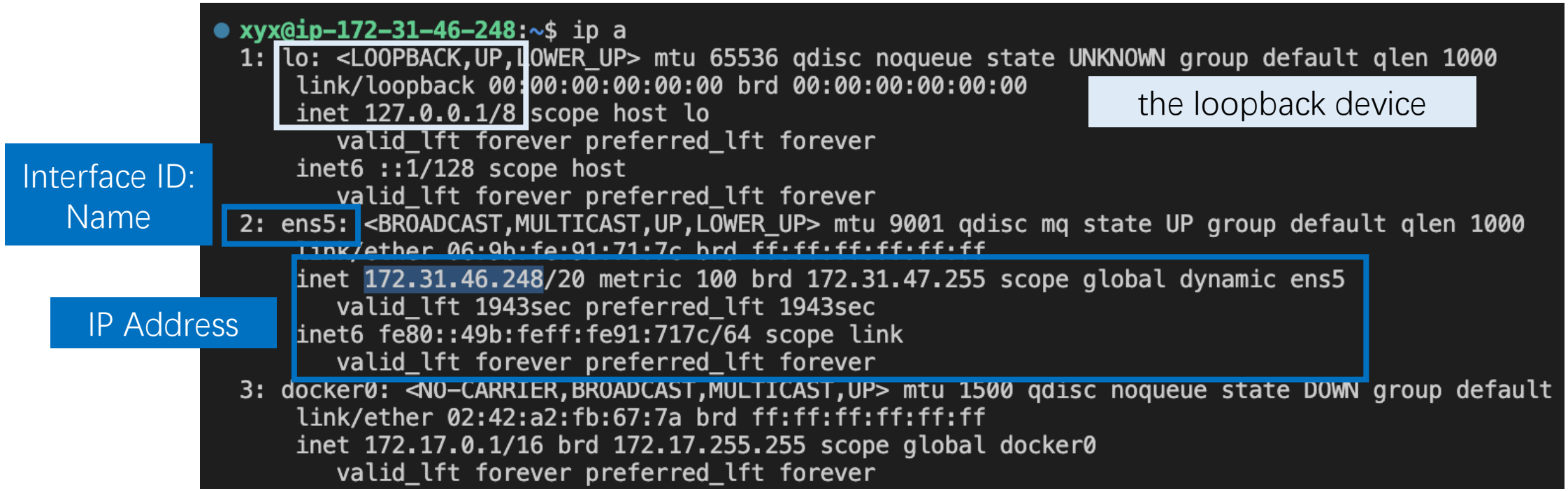
A Network Tour: Network Layer

网络漫游：网络层

B. IP Addressing (1)

IP specifies how **network interfaces** in a network are *addressed*…

Try ip a in a Linux terminal (a is short for address):



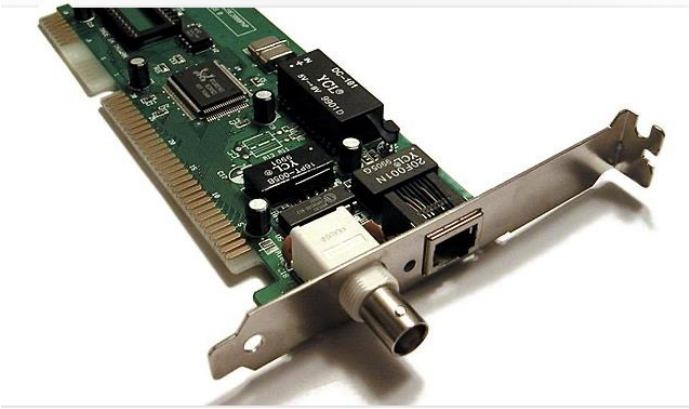
A Network Tour: Network Layer

网络漫游：网络层

B. IP Addressing (1)

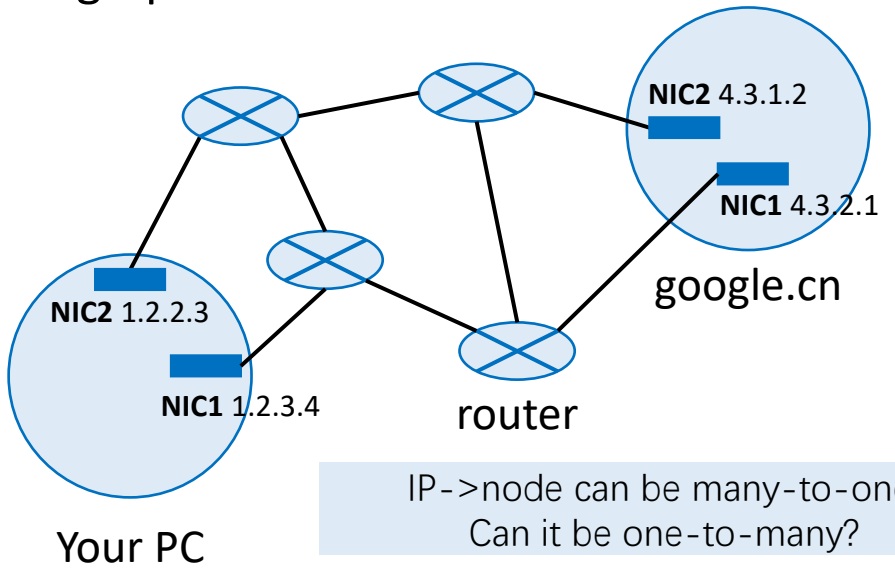
To be precise, IP addresses **NIC**s, not computers/nodes/hosts…

It is perfectly fine for a host/router to have multiple NICs, and thus have multiple IP addresses.



A **network interface controller** is a computer hardware component that connects a computer to a computer network.

So the graph is more like:



IP->node can be many-to-one.
Can it be one-to-many?

A Network Tour: Network Layer
网络漫游：网络层

B. IP Addressing (3): Subnets, Classful & CIDR

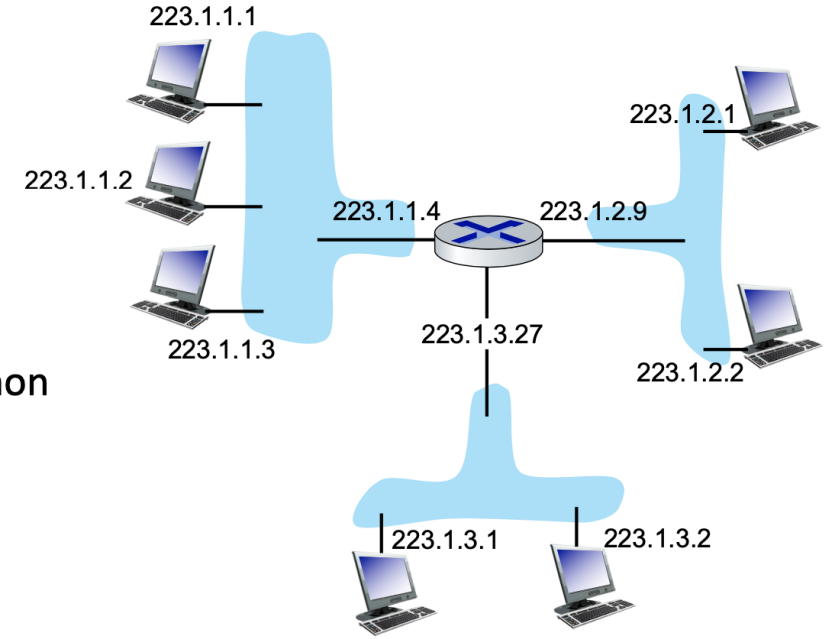
Take this IP: 172.31.46.248/20. Why is it **dotted**, and what is the **slash(/)**?

```
    inet 172.31.46.248/20 metric 100 brd 172.31.47.255 scope global dynamic ens5
```

Subnets

The slash(/) means a **subnet mask**. 172.31.46.248/20 means the leftmost 20 bits describe a subnet. The remaining 12 bits specify a host.

- *What's a subnet ?*
  - device interfaces that can physically reach each other without passing through an intervening router
- IP addresses have structure:
  - subnet part: devices in same subnet have common high order bits
  - host part: remaining low order bits



network consisting of 3 subnets

A Network Tour: Network Layer
网络漫游：网络层

Why Subnets?

- To give the network some structure – for routers!

  (The routing table need not contain all IP addresses, only subnets. Routing tables can be really large…)

- Enables **private networks and public networks**.
  - Certain subnets are reserved for private assignment. In fact, the IP you see with `hostname –i` is most likely private.
  - IP addresses can be *reused* across private networks. This helps with the IPv4 depletion problem (more on that later).

A Network Tour: Network Layer

网络漫游：网络层

Classful IP Addressing

Before CIDR: Classful Addressing

Type	Subnet mask
Type A	255.0.0.0
Type B	255.255.0.0
Type C	255.255.255.0

Type	Private Address	Simplified Repr
Type A	10.0.0.0 - 10.255.255.255	10.0.0.0/8
Type B	172.16.0.0 - 172.31.255.555	172.16.0.0/12
Type C	192.168.0.0 - 192.168.255.255	192.168.0.0/16

- These private addresses endures. By “private”, we mean that the address cannot be used in the public Internet domain. Subnets can use these addresses within their own domain. So when IPv4 was announced depleted in February 2011, the actual number of devices that connects to Internet was significantly more than 2^32.
- With IPv6, even each sand can be addressed.

A Network Tour: Network Layer

网络漫游：网络层

Other Special IP Addresses

- Localhost [or loopback]: 127.0.0.1
- Broadcast [to all devices in the subnet]: 255.255.255.255

- Full story: Given the subnet a.b.c.d/x
  - When all the remaining 32-x bits are 0, it indicates the network address of the subnet.
  - When all the remaining 32-x bits are 1, it is the broadcast address of the subnet.
  - This is why you can't use either of them as actual host addresses: they are reserved for this use.

A Network Tour: Network Layer

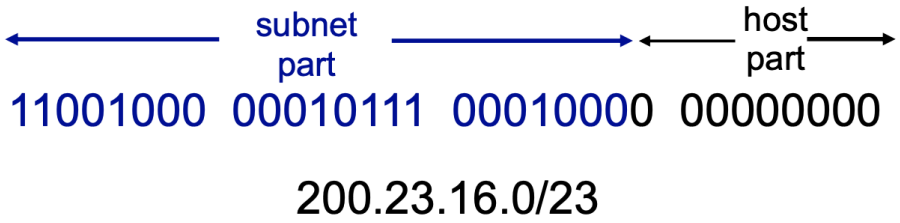
网络漫游：网络层

CIDR Addressing

IP addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced “cider”)

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address



IPv4, IPv6 and IP address depletion

IPv4 addresses (32-bit) have been depleted in February 2011. IPv6 (128-bit), despite being over 20-year-old and offering more addresses, is still not quite here because of its tricky differences.

The only way Internet is still growing, and what enables YOU to launch random Linux VMs that also have IP addresses, is to **reuse private IP addresses**.

Being private, your VM’s IP address is only reachable within your LAN (you cannot make it a public server), and your classmate’s VM IP can be exactly the same as yours.

Wonder how your VM connects to the Internet anyway?

A Network Tour: Network Layer

网络漫游：网络层

B. IP Addressing (4): IP Configuration*

General idea: IP addresses are always **assigned**.

- **Public** Internet is maintained by **ISPs** (**Internet Service Provider**, for example China Mobile). They set up routers and their IPs, and sell you routers with assigned IP addresses.
- **Private** addresses, then, are managed by enterprises/yourself however they want. When Wi-Fi is enabled on your PC, it uses DHCP (Dynamic Host Configuration Protocol) to contact your wireless router and then a DHCP server, which *leases* a private IP to you.

lease timeout!

```
inet 172.31.46.248/20 metric 100 brd 172.31.47.255 scope global dynamic ens5
   valid_lft 1943sec preferred_lft 1943sec
inet6 fe80::49b:feff:fe91:717c/64 scope link
   valid_lft forever preferred_lft forever
```

- To bridge between private and public, there is **NAT (Network Address Translation)**.

  Basically, all out-going private traffic takes the one public IP address that you bought from the ISP, and in-going traffic is redirected to their local IP destinations. *Read for more on Wiki*!

A Network Tour: Network Layer

网络漫游：网络层

A Final Case Study

✧ **Weicheng Zhao**：“每个主机映射到一个ip地址。”如何理解主机？我的电脑有一个ip地址，192开头那个，连接上校园网查询我的ip，是“111.205.230.27 -> 北京市 联通,北京大学”，我的移动网络ip是“223.104.39.154 -> 北京市 移动数据上网公共出口”，挂上vpn我的ip可以在全球各地.....那么每个主机的ip地址是属于电脑这台设备的还是某个网络的？连接到的无线网络的ip地址（校园网、移动网）又是什么？人们常说的要保护好自己的ip地址，不要轻易泄露，又是出于何种考虑？

- The 192.xx.xx.xx IP is **private**, assigned by the network you are connect to.
- The 111.205.240.27 is probably owned by PKU, bought and assigned from ISP. In this case, you are connected to a wireless **AP** (Access Point) on campus.
- The 223.104.39.154 is assigned from another ISP, that you paid for your 4G/5G service. It is probably the IP of the **base station** your cell phone is talking to.
- The VPN is another whole mess (and fun ☺). In a nutshell, you talk to **proxy servers** via IPSEC (secure IP protocol, that sets up tunnels from you to the proxy). The tunnel is encrypted, so it only looks as if the proxy is accessing the network for you.
- The public IPs are maintained by ISPs, and they know the **physical locations**.

A Network Tour: Down to Link Layer

网络漫游：链路层

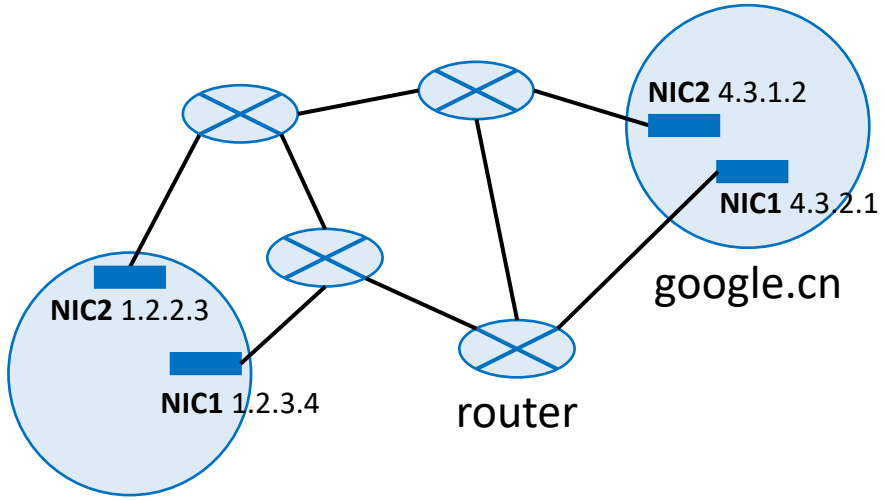
By far we only talked about the Network Layer, i.e. *how to locate stuff*.

> Maybe you are already overwhelmed. Please collect yourself, because I'm not going into too much details for the following layers…

Looking down, there remain two problems: (a) how to format the data sent in the network, and (b) how the data is actually sent. The **link layer** tackles (a), and the **physical layer** tackles (b).

Still not Accurate Enough ->

What do we do with the data when we put messages onto NIC?
How does NIC actually send out data?
Is there only **routers** in between?



A Network Tour: Down to Link Layer

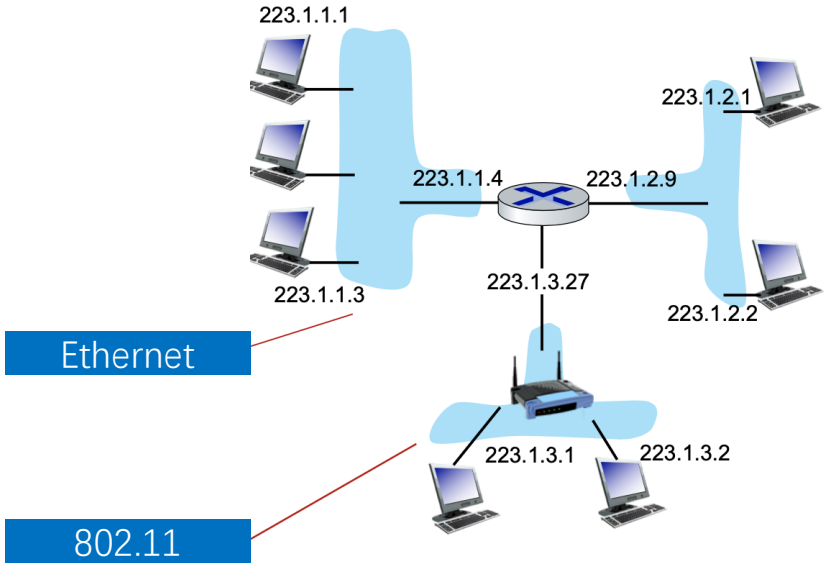
网络漫游：链路层

Two Exemplar Link Layer Protocols

- **Ethernet.** This is for wired networks. It specifies an **MTU** (Maximum Transmission Unit, i.e. the maximum packet size), which has some *funny* outdated relation with the latency and bandwidth of old wires.

- **802.11, or Wi-Fi**. This is for wireless networks. Wireless is a more challenging setting, and the protocol specifies **error detection and correction** (because of noise), data **compression** (because of limited bandwidth), and **encryption** (because it's broadcast).

Hopefully this shows the **necessity** of link layer protocols. Sending raw data is not viable in lower layers, especially when the actual medium is noisy/restrictive.



A Network Tour: Down to Link Layer

网络漫游：链路层

MAC Address

Each Ethernet adapter has a globally unique 48-bit address that is stored in a nonvolatile memory on the adapter. A host can send a chunk of bits called a *frame* to any other host on the segment. Each frame includes some fixed number of *header* bits that identify the source and destination of the frame and the frame length, followed by a *payload* of data bits. Every host adapter sees the frame, but only the destination host actually reads it.

CSAPP P643

In a Wi-Fi network, MAC addresses specify source and destination devices.

```
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 06:9b:fe:91:71:7c brd ff:ff:ff:ff:ff:ff
```

- Besides an IP address, each NIC also has a 48-bit MAC address in its ROM, also known as the *physical address*.
- These MAC addresses are globally unique, and are not changed.*

Why both MAC address and IP address?

I guess everyone faces this problem when they learn about both addresses. Here is a short answer**: The MAC address (unique) is for identifying *who you are*, while the IP address (structured and assigned) tells about *how to find you***.

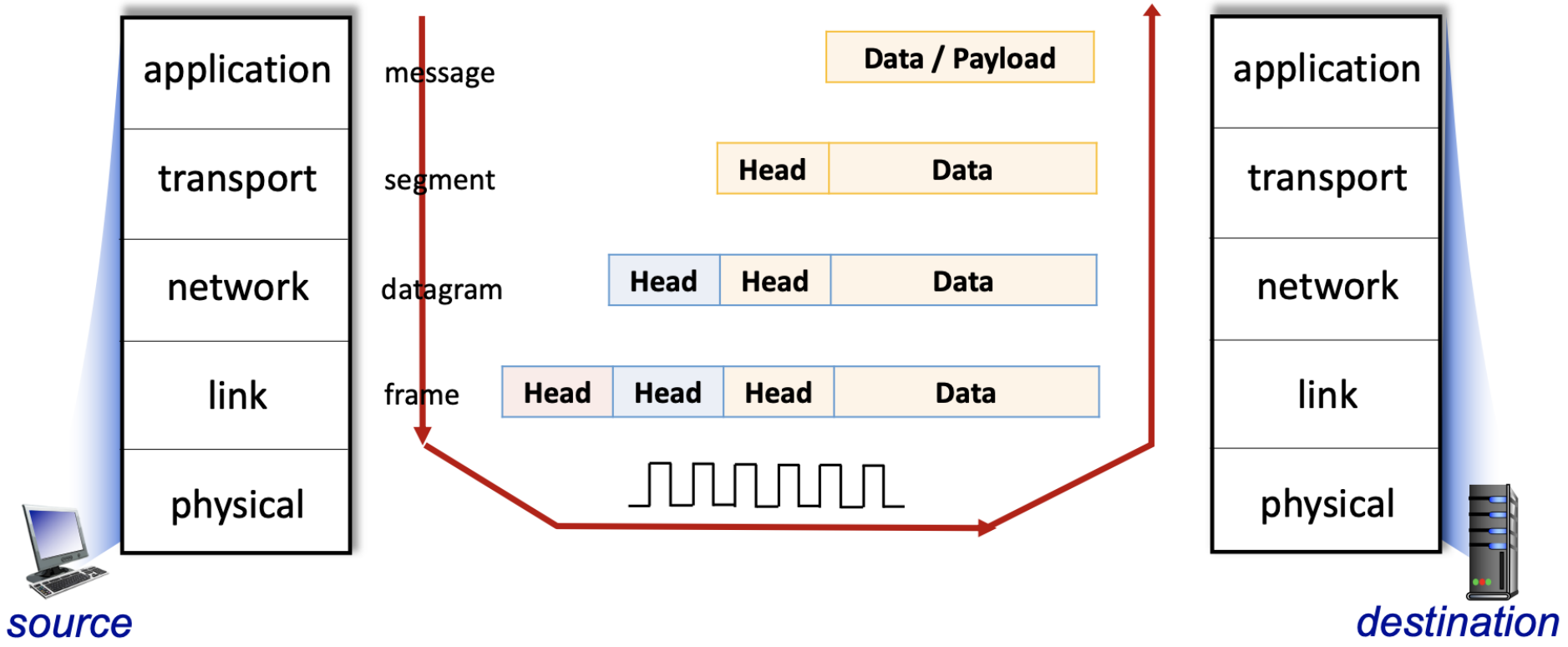
*: There is still a *but*. You can search for it yourself.

A Network Tour: Down to Link Layer

网络漫游：链路层

Encapsulation

We should note that all protocols discussed by far will add **headers** to their payload.

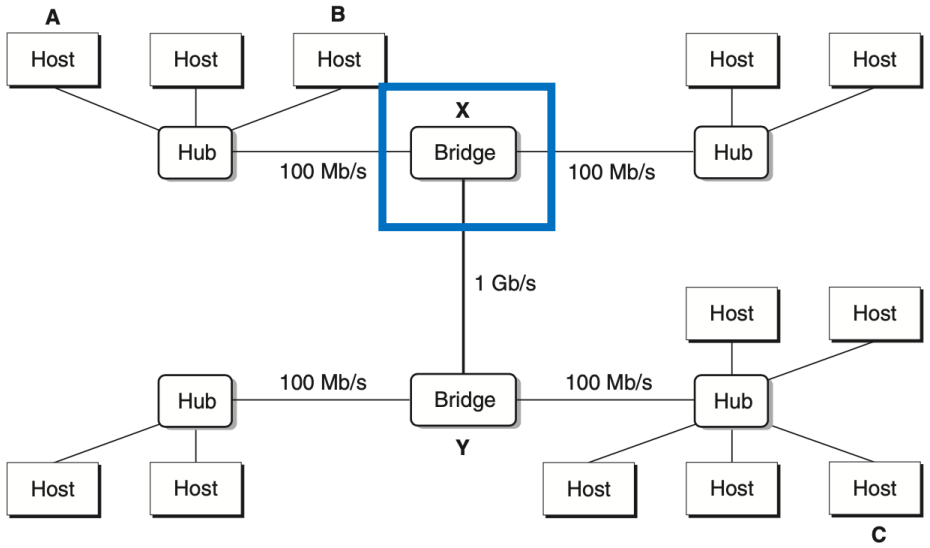


A Network Tour: Down to Link Layer

网络漫游：链路层

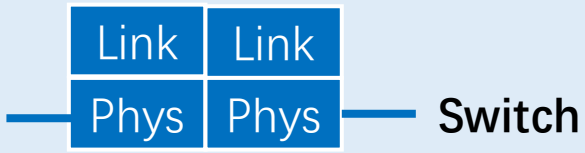
Encapsulation & Switch / Router

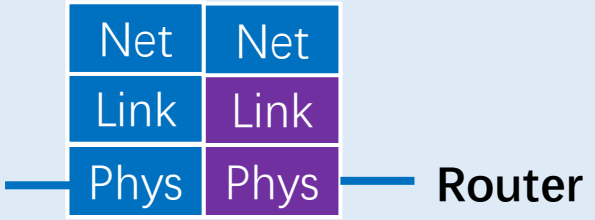
Encapsulation helps separate the layers, and bridge between different networks.



A bridge is *not* a router.

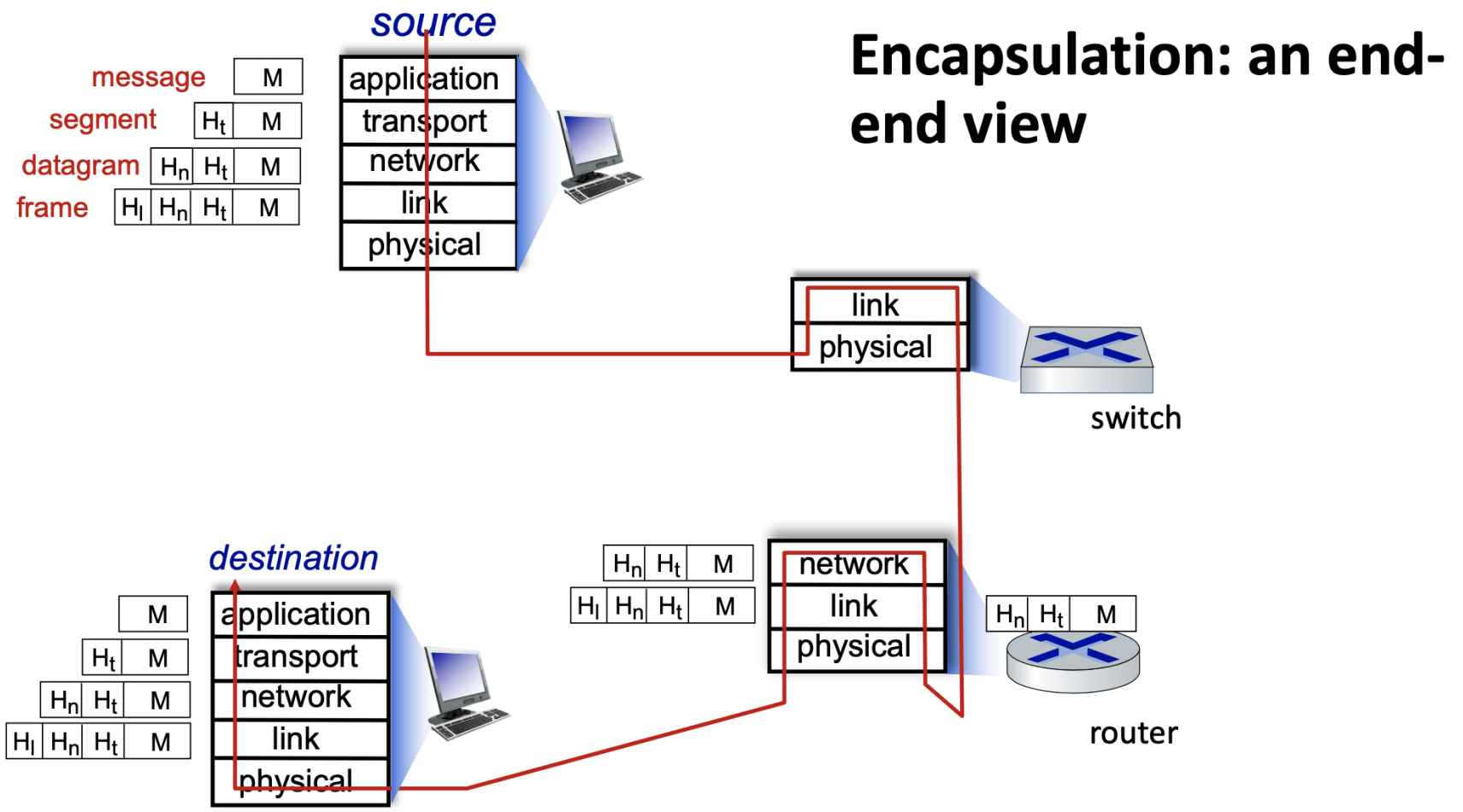
- Bridges/switches operate at the link layer.
- They also build a **forwarding table**, not by routing algorithms, but with a special memory/broadcasting scheme, using **MAC addresses**.
- They do not scale very well, and are used only for connecting a LAN.





A Network Tour: Down to Link Layer

网络漫游：链路层



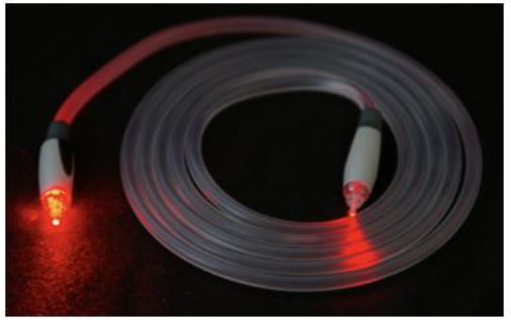
A Network Tour: Physical Layer

网络漫游：物理层

As the bottom layer, the **physical layer** is about how data is actually sent.

- We are working with *hardware* and the *physical* world now. We care about **fiber**, **copper wire**, **radio** and their physical properties to design hardware encoders and decoders.

wire



Fiber optics

satellites

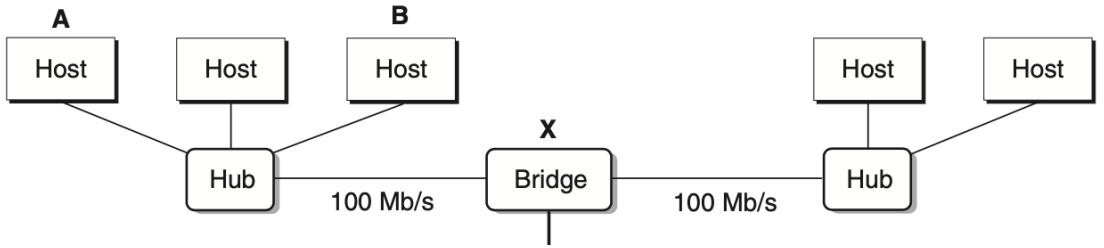
- Still, this is **more than materials**. For example, we use the Fourier transform to robustly encode information with sine waves. We design algorithms to avoid conflicts, or de-noise antenna array signals.

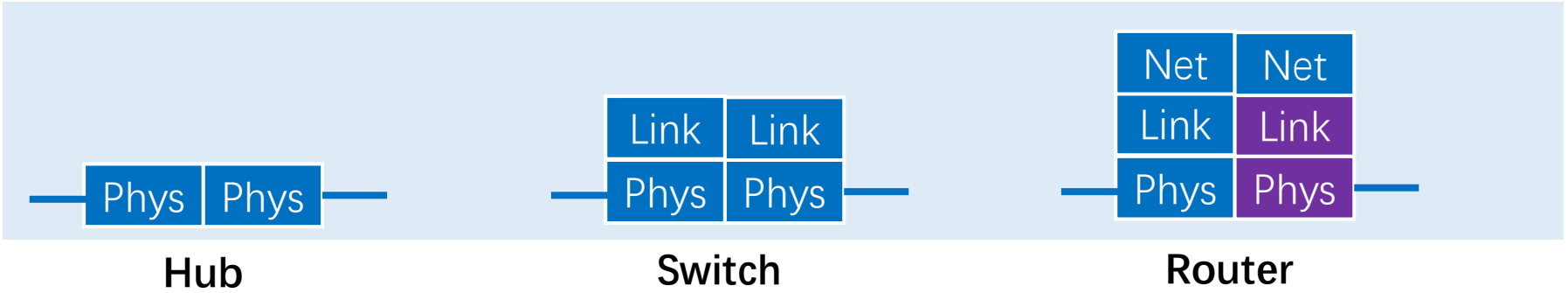
  Mathematics is also CS. How fun.

A Network Tour: Physical Layer
网络漫游：物理层

Hub

A hub operates in the physical layer, and it is nothing magical. Just think about a bunch of wires connected together.





A Network Tour: A Recap

网络漫游：中场回顾

We have now toured the lower half of the protocol stack. Note that we already have a working network that provides **best-effort** delivery.

Much more accurate than the graph in the beginning…

Billions of connected computing *devices*:

- *hosts = end systems*
- running *network apps* at Internet's "edge"

Packet switches: forward packets (chunks of data)

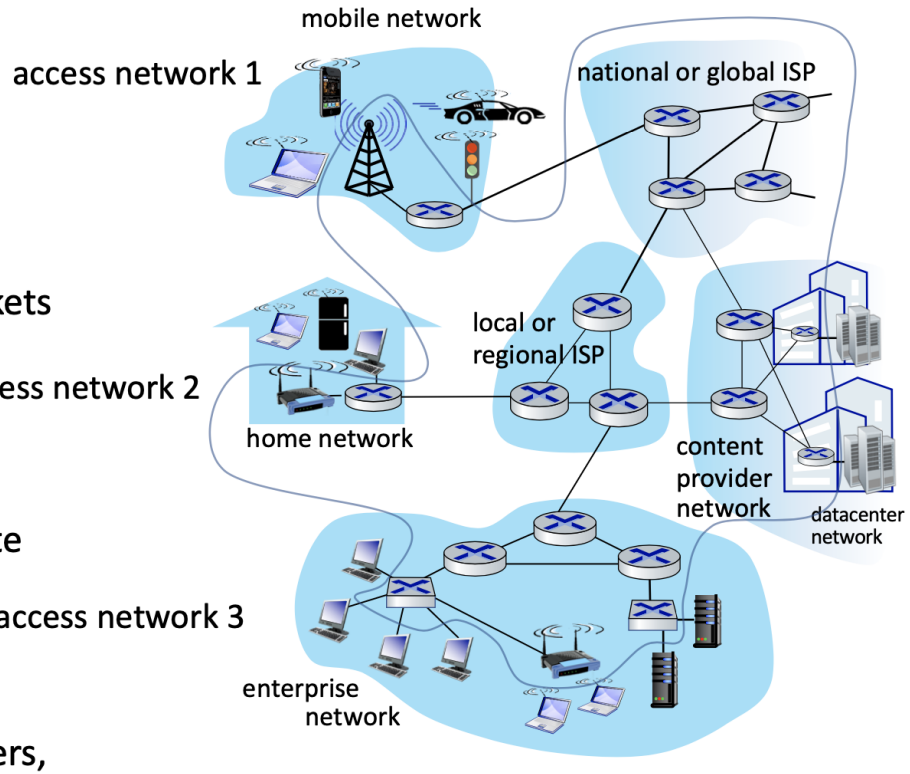
- *routers, switches*

Communication links

- fiber, copper, radio, satellite
- transmission rate: *bandwidth*

Networks

- collection of devices, routers, links: managed by an organization



A Network Tour: Transport Layer

网络漫游：传输层

Why the transport layer, and what does it include?

- The (raw) IP interface is not very programmer-friendly…
- And it is only **best-effort** delivery, **not reliable**.

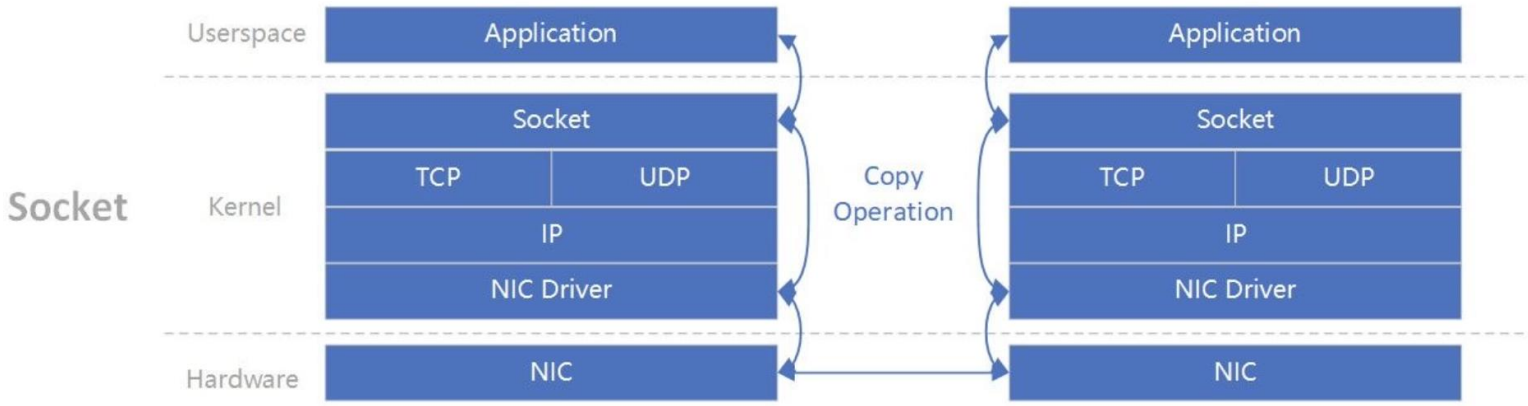
Why is IP not reliable?
You may think of the dynamic topology of the network, or the lossy links (like the wireless settings).
These can lead to loss of packets. However, another (and probably dominant) factor is **buffering in switches and routers**.
Suppose a switch/router is forwarding multiple flows to one out-link. The speed of the out-link will not keep up, and the switch/router must buffer the packets, but not endlessly. When the buffer is full, newer packets are simply discarded.

- We need a **programmer friendly interface** with multiplexing (multiple applications can use the network simultaneously).
- And **reliable** delivery!

A Network Tour: Transport Layer
网络漫游：传输层

Socket: The Interface

- Network **sockets** are merely an interface invention. It combines an IP address with a **port**, with the port number providing the multiplexing functionality.
- Processes use sockets as **endpoints** of network data transmission.
- In Linux, the socket is actually *a general abstraction* above multiple transport layer protocols, and appears to be a file descriptor.



Both TCP and UDP have ports and sockets. The SOCK_STREAM type in socket() means TCP, but there is also one for UDP.

A Network Tour: Transport Layer
网络漫游：传输层

TCP: The Reliability

- The **TCP (Transmission Control Protocol)** is the prominent reliable data transport protocol nowadays.

  Bit of a monstrosity itself; details are omitted.

- TCP is **connection-oriented**. All TCP sessions start with a connection setup.

  Reasons why this matters are also a bit involved. Simply put, a connection is needed for reliability and flow control. Moreover, the underlying network are actually optimized for these flows.

There is also the **UDP (User Datagram Protocol)**, that gives no reliability guarantee, and is connectionless. It is basically the raw IP with sockets on top.

Why both UDP and TCP?

UDP includes the *usability* from "interface," but not the possible *overhead* from "reliability."
Certain apps may use UDP for the lesser overhead, like voice chatting where latency matters.

A Network Tour: Transport Layer
网络漫游：传输层

Programming Over the Transport Layer

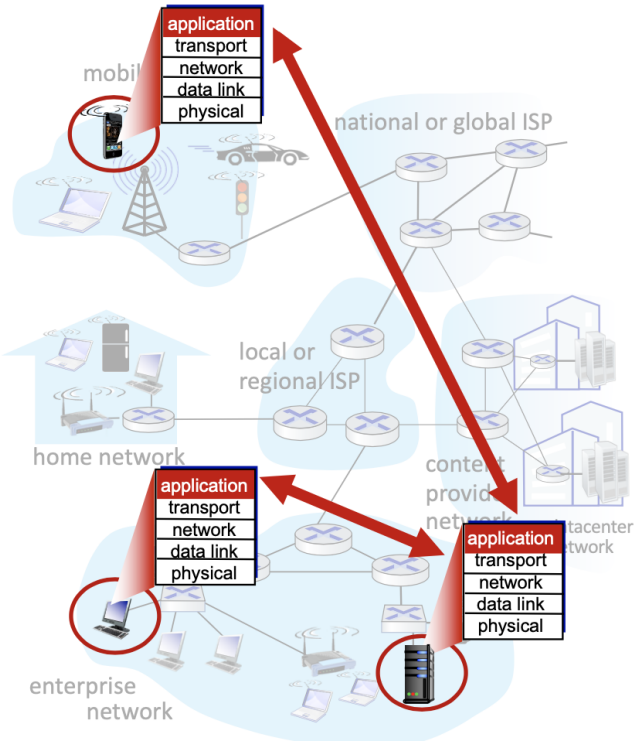
As is mentioned, the Linux network programming interface is a POSIX interface that sits on top of any transport layer protocols. Everything below is kernel code.

write programs that:

- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation



A Network Tour: Application Layer
网络漫游：应用层

Why the Application Layer?

What TCP provides is merely a *reliable two-way byte stream*. We still need to:

- Decide how to make sense of the data sent and received.
- TCP is especially tricky for it does not guarantee any boundary within the stream. One message sent may be split into two when received, so structuring the data is very much needed.

> This is why HTTP specifies a clear header, and \r\n to separate lines.

- **Exemplar Application Protocols**: HTTP, SFTP, IMAP… These all use TCP, but there are also protocols over UDP, like RTP (Realtime-Transport Protocol).

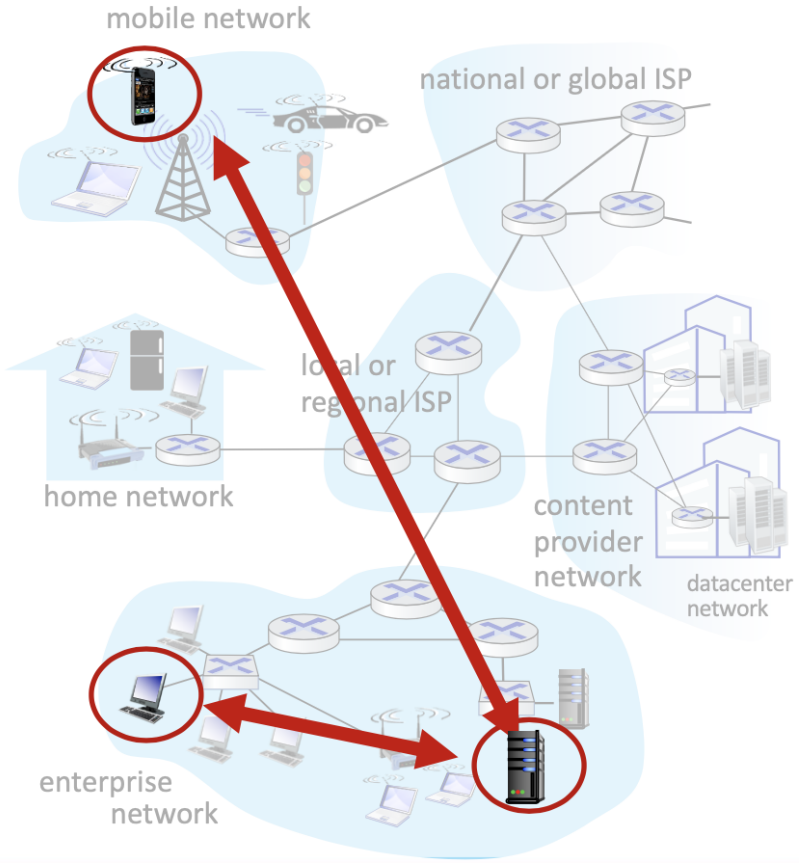
A Network Tour: Application Layer
网络漫游：应用层

Client-Server Paradigm

server:
- always-on host
- permanent IP address
- often in data centers, for scaling

clients:
- contact, communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do *not* communicate directly with each other

- examples: HTTP, IMAP, FTP



A Network Tour: Application Layer
网络漫游：应用层

HTTP

HTTP (Hyper Text Transfer Protocol) is typically used to transfer web contents.

- web page consists of *objects,* each of which can be stored on different Web servers
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of *base HTML-file* which includes *several referenced objects, each* addressable by a *URL,* e.g.,

This is just mnemonics. Need DNS to translate to IP address.

`dromniscience.xyz/someDept/pic.gif`

host name　　path name

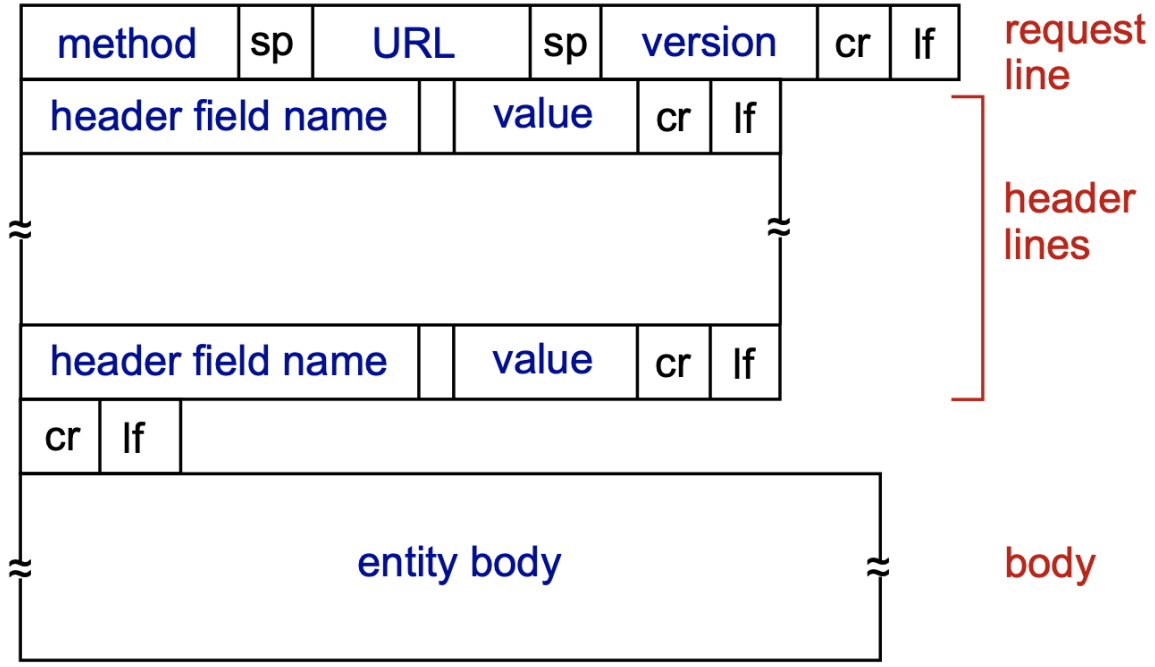
`47.94.237.126/index.html`

host addr　　path name

A Network Tour: Application Layer

网络漫游：应用层

HTTP request message: general format



A Network Tour: Application Layer
网络漫游：应用层

Other HTTP request messages

POST method:

- web page often includes form input
- user input sent from client to server in entity body of HTTP POST request message

GET method (for sending data to server):

- include user data in URL field of HTTP GET request message (following a '?'):

HEAD method:

- requests headers (only) that would be returned *if* specified URL were requested with an HTTP GET method.

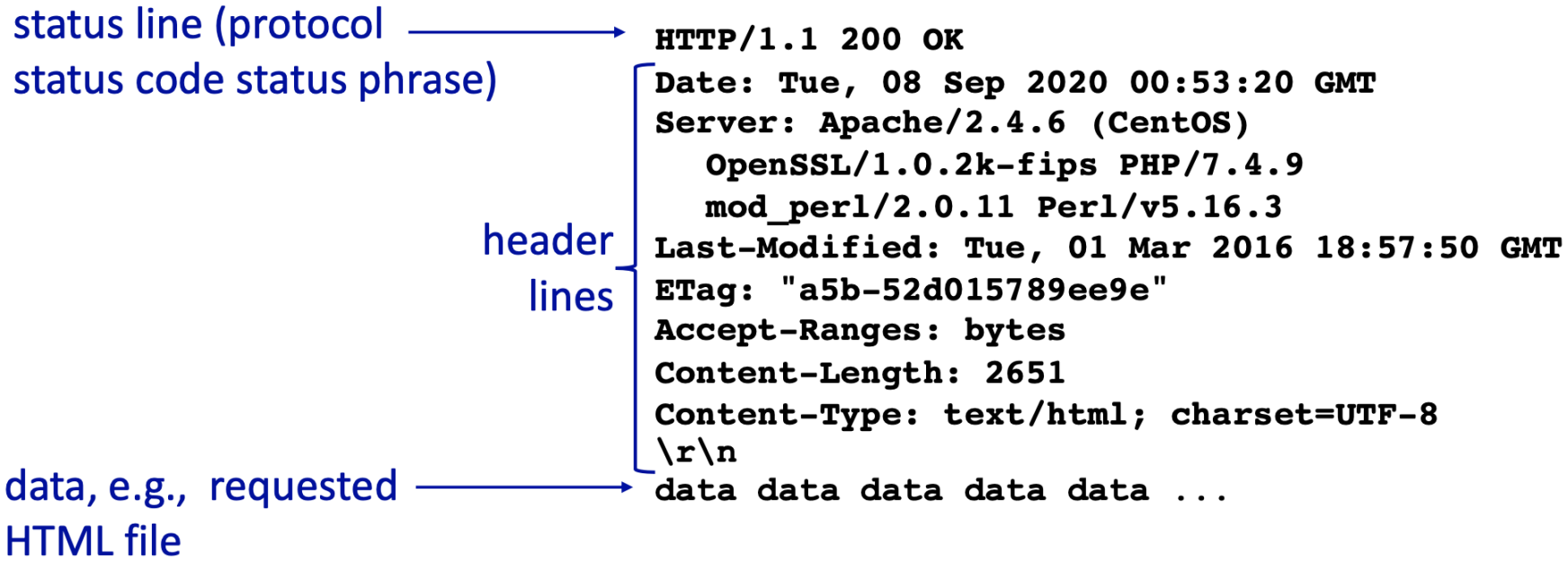
PUT method:

- uploads new file (object) to server
- completely replaces file that exists at specified URL with content in entity body of POST HTTP request message

A Network Tour: Application Layer

网络漫游：应用层

HTTP response message



What you need to do in the proxylab is to relay these HTTP requests and responses, plus some caching.

A Network Tour: Application Layer

网络漫游：应用层

HTTP response status codes

- status code appears in 1st line in server-to-client response message.
- some sample codes:

200 OK
- request succeeded, requested object later in this message

301 Moved Permanently
- requested object moved, new location specified later in this message (in Location: field)

400 Bad Request
- request msg not understood by server

404 Not Found
- requested document not found on this server

505 HTTP Version Not Supported

500-504 Elective: coupon collector

A Network Tour: Application Layer

网络漫游：应用层

DNS

DNS (Domain Name System) translates domain names into IP addresses.

- Domain names are separated by dots(.), and are traversed backwards.

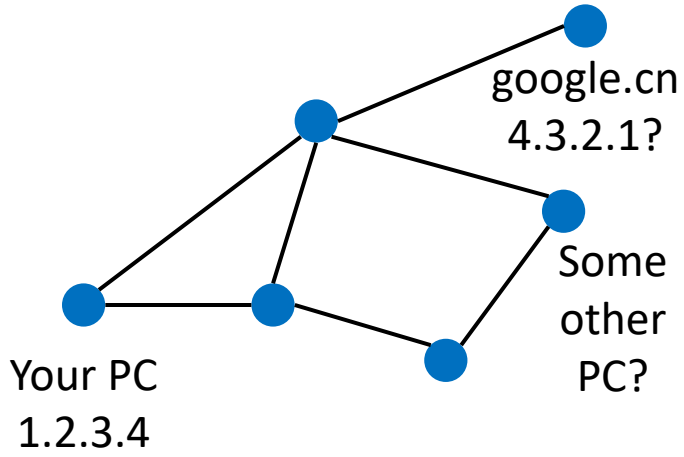
  E.g. pku.edu.cn is firstly a part of .cn; then .edu.; then .pku.

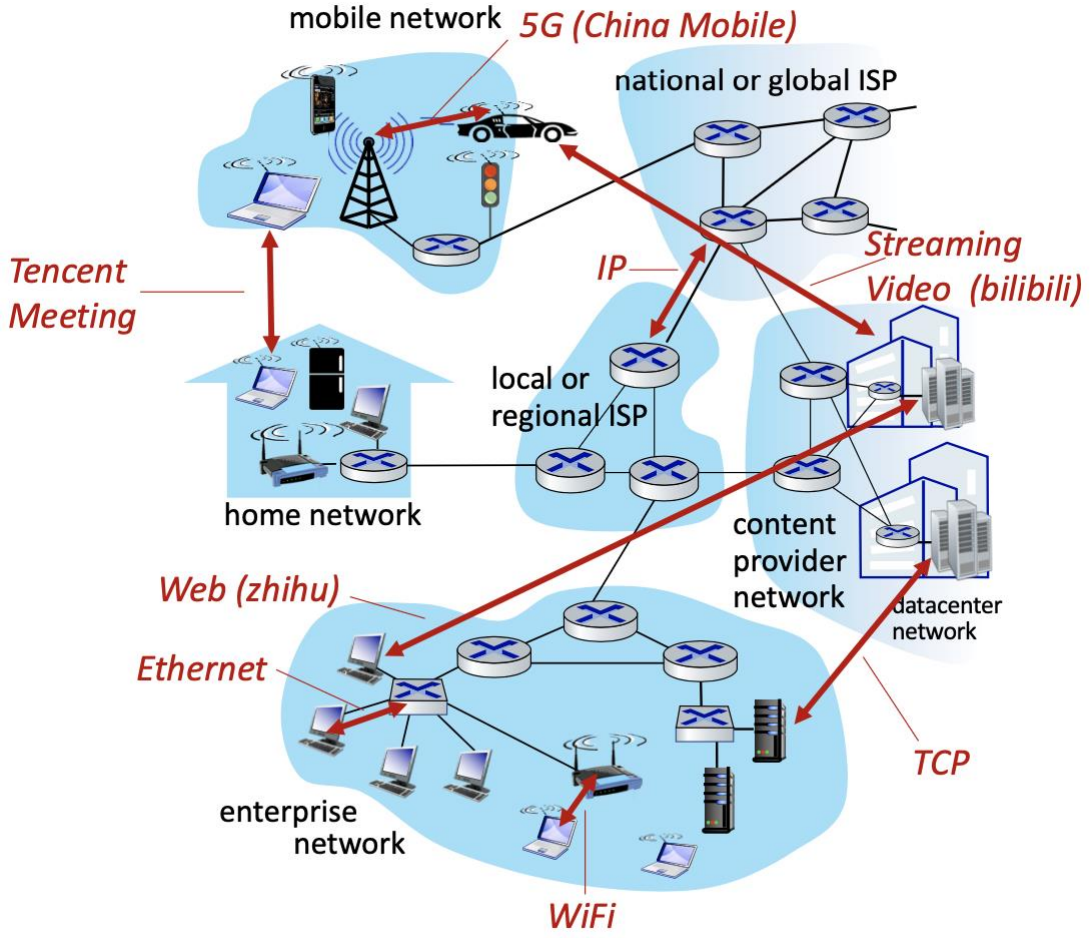
- Each level corresponds to a set of DNS servers that provide the resolution service. DNS servers in PKU are responsible for xxx.pku.edu.cn.
- DNS actually provides interesting opportunities. In fact, the name-to-IP mapping may be **one-to-many**, especially with popular contents (like the Bilibili video database). Basically, the content is copied to multiple places (and IPs), and the DNS server gets to pick one IP that is the closest to the client / least loaded. This trick enables **CDN (Content Distribution Network).**

A Network Tour: A Complete View

网络漫游：完整视角

Refresh Your Conceptual View!





Still a lot more to learn in the Computer Network course··· *Alas*.

Next Class…

- **Chapter 12: Concurrent Programming**
  - ..
- **Prepare for the Exam!**
  - ..