CAN201 Final

Lecture 1 Intro

The whole picture of the network

- 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 closer look at internet structure

Network edge

- hosts: clients and servers
  - takes applications message
  - breaks into smaller chunks, known as packets, of length L bits
  - Transmits packet into access network at transmission rate R
    - link transmission rate, aka link capacity aka link bandwidth 带宽

$$packet\,transmission\,delay = time\,needed\,to\,transmit\,L-bit\,packet\,into\,link = L/R$$

- servers often in data centers

Physical media

- physical media: Guided media (copper fibber coaxial cable glass)
  - Twisted pair : 双绞线: 常见的有限传输介质 由两根绝缘铜线组成 Category 5规格可以支持最高传输速率为100 Mbps
  - Coaxial cable: 同轴电缆: 由两个同心的铜导体构成 可以双向传输 如HFC网络
  - Fiber optic cables: 光纤（每个脉冲代表一个bit），可以实现高速点对点传输，具有较低的误码率 和 良好的抗电磁干扰能力
- physical media: Unguided media (radio: electromagnetic waves)

- Signal carried in electromagnetic spectrum
- No physical wave
- Bidirectional
- Propagation environment effects: Reflection / Obstruction by objects/ interference 干扰

Different methods to access the internet

- Early methods
  - Dial-up Internet access(PSTN)
    - Modems are required 调制解调器
      decode audio signals into data for computers or routers → encode digital signals back into audio for transmission to another modem
  - Digital subscriber line(DSL)
    - Telephone line based, to central office DSL Access Multiplexer(DSLAM)
    - DSL 电话线上的数据通往internet
    - DSL 电话线上的语音通往电话网
    - Bandwidth : Upstream transmission rate ~< 3Mbps(typically < 1Mbps), Downstream transmission rate ~< 50 Mbps(typically < 10Mbps)
- Modern methods
  - FTTH: fiber to the home. FTTH using the passive optical networks(PONs) distribution architecture 无源光分布架构的光纤到户
  - Wireless (base Stations → provide the signal)

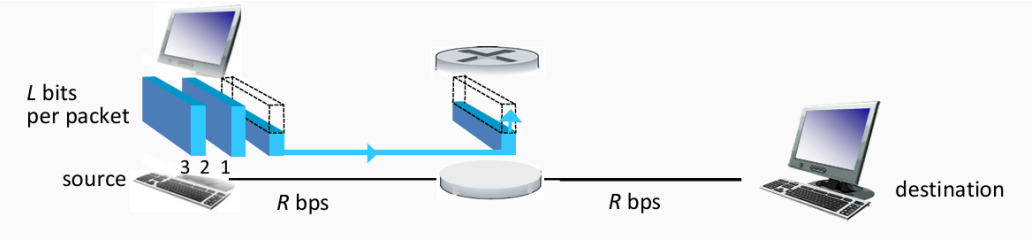
Access networks

- enterprise networks
  - Companies Universities
  - 混合了有线和无线链路技术 链接各种交换机和路由器
  - 以太网(Ethernet): 有线接入 速率100Mbps 1Gbps 10Gbps
  - WiFi: 无线接入点速率：11，54，450Mbps
- data center networks

Network core

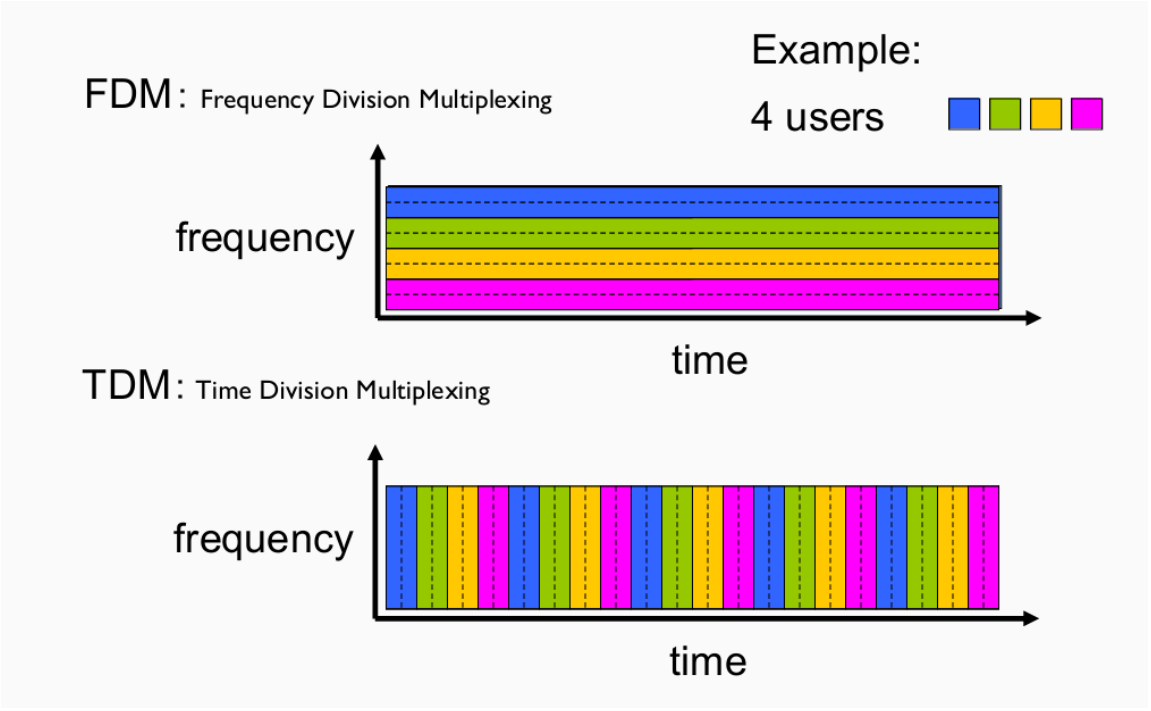
- Mesh of interconnected routers 互联路由器的网状结构
- packet- switching:主机将应用层消息分割成packet
- 网络将packets从一个路由器转发到下一个路由器，沿着从源到目的地的路径跨越链路
- Two key network-core functions
  - Forwarding:

        - aka switching
        - local action: move arriving packets from router's input link to appropriate router output link(根据本地转发表来路由操作)
        - 干活 路由器在执行 查路由表 把数据包从正确的接口发出去（按照导航开车）
    - Routing
        - global action: determine source-destination paths taken by packets
        - routing algorithms
        - 路由器在思考 到目的地有哪几条路 那条路最好 然后生成路由表（导航规划路线）
- Packet Switching:
    - Store-and-forward
        - Takes L/R s to transmit (push out) L-bit packet into link at R bps
        - End-end delay = 2L/R 假设 无传播延迟(propagation delay)
        - One-hop 单跳 有几个节点就跳几次



    - queueing delay, loss
        - 一段时间内的到达链路的arrival rate (in bits) 超过 链路的传输速度
            - packets will queue, wait to be transmitted on link
            - packets can be dropped(lost) if memory (buffer) fills up
- Alternative: Circuit switching(电路交换)
    - dedicated resources: no sharing, 保证性能
    - Circuit segment is idle if not used by call
    - 传统电话网络中常用

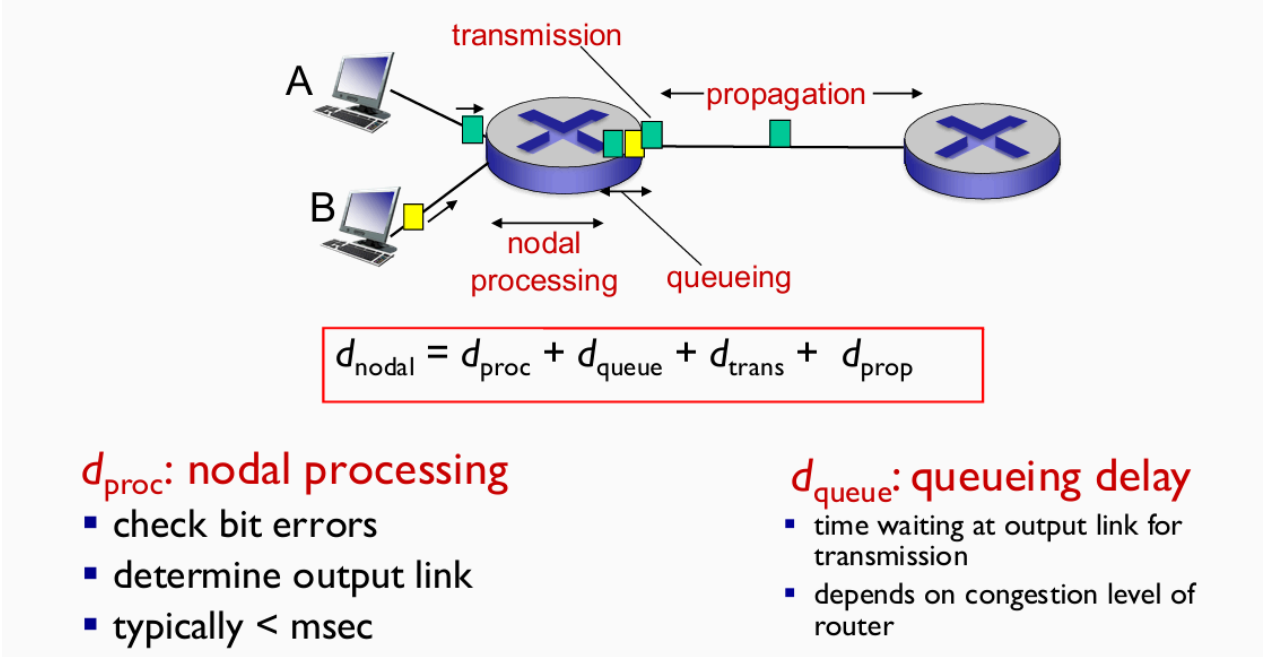
- FDM（频分复用） vs TDM(时分复用)

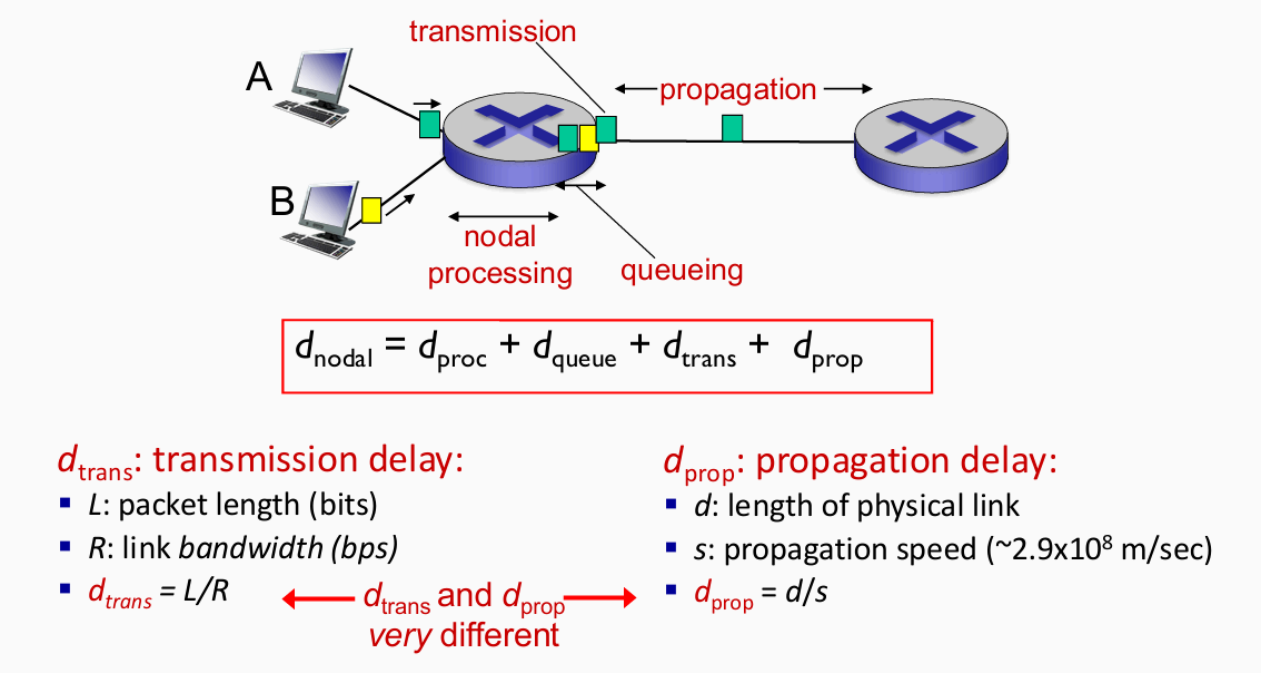


- Packet switching VS circuit switching
  packet switching allows more users to use network
  - PS advantages: resource sharing / simpler, no call setup
  - PS drawbacks: excessive congestion 过度拥塞:delay and loss, protocols needed for rdt(reliable data transfer),congestion control
  - circuit-like behavior PS? --Bandwidth guarantees

Evaluate the performance

- Package Loss / Delay / Bandwidth
- Four sources of packet delay





- 传输延迟：取决于包长度 传播延迟：取决于链路长度
- Queueing delay:
  - R: link bandwidth(bps)
  - L: packet length(bits)
  - a: average packet arrival rate
    - La/R ~0: avg. queueing delay small
    - La/R -> 1: avg. queueing delay large
    - La/R > 1: more work arriving than can be serviced, average delay infinite
- Packet loss
  -Queue (buffer) preceding link in buffer has finite capacity.Packet arriving to full queue dropped(lost)
  - 可能重传 也可能完全不重传
- Throughput
  - rate (bits/time unit) at which bits transferred between sender/receiver
  - bottleneck link瓶颈传输: link on end-end path that constrains end-end throughput

Service models

- Internet protocol stack
  - application: supporting network applications
    - FTP,SMTP,HTTP, DNS
  - transport: process-process data transfer
    - TCP, UDP
  - network: routing of datagrams from source to destination
    - IP, routing protocols
  - link: data transfer between neighboring network elements

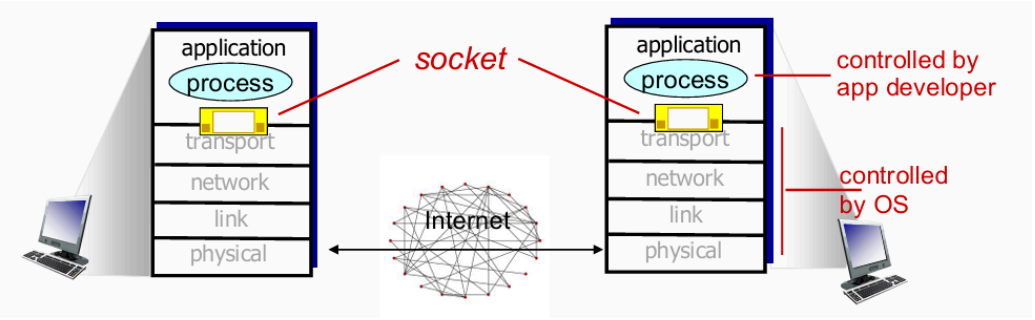
        - Ethernet, Wifi, PPP
    - Physical: bits "On the wire"
- ISO / OSI reference model
    - ISO / Open System Interconnection
    - 多出presentation（加密 压缩...）/ session层(同步 检查点 数据交换的恢复) internet stack 通过应用层来实现这些缺失的功能
- message(application)→segment(transport)→datagram(network)→frame(link)

Lecture 2 Application Layer: principles / web / HTTP

Principles of network Application

Processes communicating

- Process: program running within a host 在主机内运行的程序
- 在同一主机内，两个进程使用进程间通信（inter-process communication） defined by OS
- processes in different hosts communicate by exchanging messages
- Sockets
    - Process sends/ receives messages to/ from its socket
    - analogous the door
        - Sending process to "push" message out of the door
        - Sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process
        - 2 sockets involved: one on each side



        -Addressing processes
    - process must have identifier
    - Host device has unique 32-bit IPv4 and/or 128-bit IPv6

Paradigms for network Apps

Client-Server Paradigm

- Server
    - Always-on host

    - Permanent IP address
    - Often in data centers for scaling 拓展.
- Clients
    - Contact, communicate with server
    - May be intermittently connect to the internet
    - May have dynamic IP address
    - Do not communicate directly with each other

P2P diagram

- No always-on server is needed
- End system directly exchange data
- client process / server process on the same host 客户端进程/服务器进程位于同一主机上
- Self scalability -new peers bring new service capacity, as well as new service demanda
- Maybe dynamic IP addresses-complex management

App-Layer protocol defines

- Paradigm: CS or/and P2P
- Types of messages exchanged:request /response
- Message syntax: what fields in messages & how fields are delineated
- Message semantics
    - meaning of information in fields
- Message timing: when and how

Internet transport protocols services

TCP / UDP

- Securing TCP: Secure Sockets Layer - SSL
    - Provides encrypted TCP connection
    - Data integrity
    - End-point authentication
    - At Application Layer

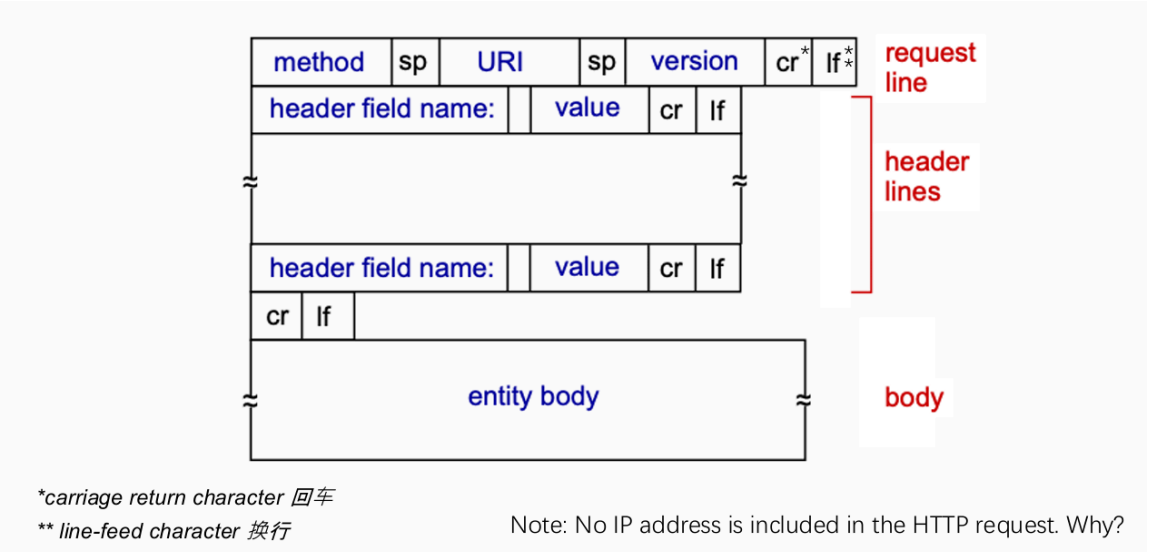
Web Application - HTTP

HyperText Transfer Protocol

Features of HTTP(1.1)

- C/S model

    - Client: browser that requests, receives(using HTTP protocol) and show Web objects(Render)
    - Server: Web server sends(using HTTP protocol)objects in response to requests
- Uses TCP
    - Client initiates TCP connection (creates socket) to server, port 80(443 for https)
    - Server accepts TCP connection from client
    - HTTP messages(application-layer protocol messages)exchanged between browser(HTTP client) and Web server(HTTP server)
    - TCP connection closed
- HTTP is Stateless
    - server maintains no information about past client requests



- HTTP 1.1 Additional methods
    - PUT / DELETE / PATCH / TRACE / OPTIONS / CONNECT
    - origin:GET / POST / HEAD
- request headers(key-value pairs, metadata about the request and the client)
    - Host name
    - Authentication
    - Content types
    - User-agent information
    - Caching / Cookie
    - Types of connection
        - Non-persistent HTTP 非持久化HTTP
            - At most one object sent over connection(connection then closed)
            - Downloading multiple objects required multiple connections
            - RTT:time for a small packet to travel from client to server and back round trip time
            - HTTP response time: OneRTT to initiate TCP connection + One RTT for HTTP request and first few bytes of HTTP response to

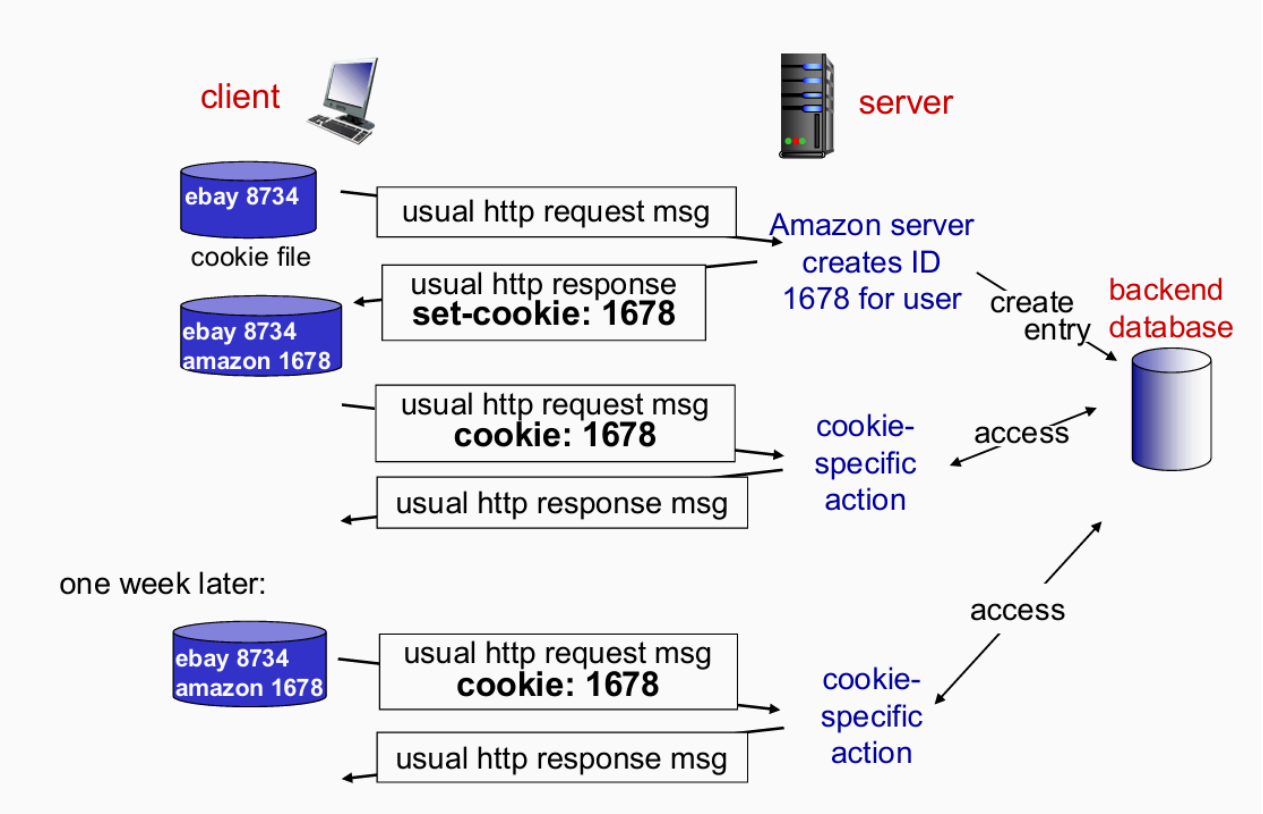
return + File transmission time

- Persistent HTTP
  - Multiple objects can be sent over single TCP connection between C/S
  - Client sends requests as soon as it encounters a referenced object

- HTTP response status code
  - status code appears in 1st line in Server-to-Client response message
  - 200 OK | 301 Move Permanently | 400 Bad Request | 404 Not Found |505 HTTP Version Not Supported

HTTP is "Stateless" ——So User-Server state: cookies

- cookie header line of HTTP response message
- cookie header line in next HTTP request message
- cookie file kept on user's host, managed by user's browser
- back-end database at Web site



- used for Authorization / Recommendations / User session state

Web caches

- cache acts as both client and server(server for original requesting client / client to origin server)
- Typically cache is installed by ISP

- Pros:
    - Reduce response time for client request
    - Reduce traffic on an institution's access link
    - Internet dense with caches: enables "poor" content providers to effectively deliver content(P2P like)

Conditional GET

- don't send object if cache has up-to-date cached version
    - no objects transmission delay
    - lower link utilization
- cache: specify date of cached copy in HTTP request
  `If-modified-since:<date>`
- server: response contains no object if cached copy is up-to-date:
  `HTTP/1.0 304 Not Modified`

HTTP/2

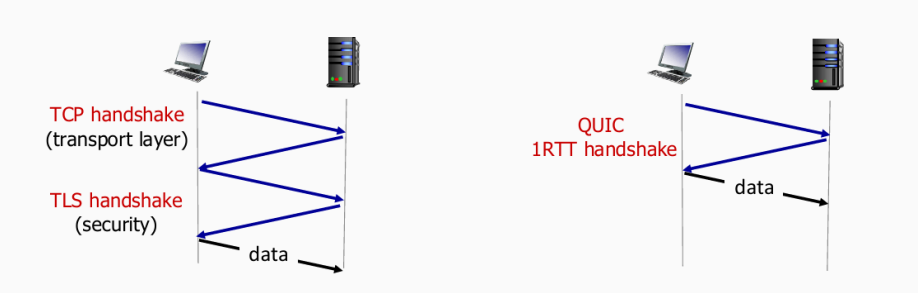
key goal: decreased delay in multi-object HTTP request
对象被分成帧，帧传输交错进行

- HTTP1.1: introduced multiple, pipelined GETs over single TCP connection
    - FCFS responds in-order to GET requests
    - 可能出现队头阻塞HOL（head-of-line blocking）
      小对象可能不得不等待传输，因为位于大对象后
    - loss recovery (retransmitting lost TCP segments)stalls object transmission
- HTTP/2: increased flexibility at server in sending objects to client:
    - methods / status codes / most header fields unchanged from HTTP/1.1
    - transmission order of requested objects based on client-specified object priority(not necessarily FCFS)
    - push unrequested objects to client 将未请求的对象推送到客户端(服务器推送)
    - divide objects into frames, schedule frames to mitigate HOL clocking(帧调度)

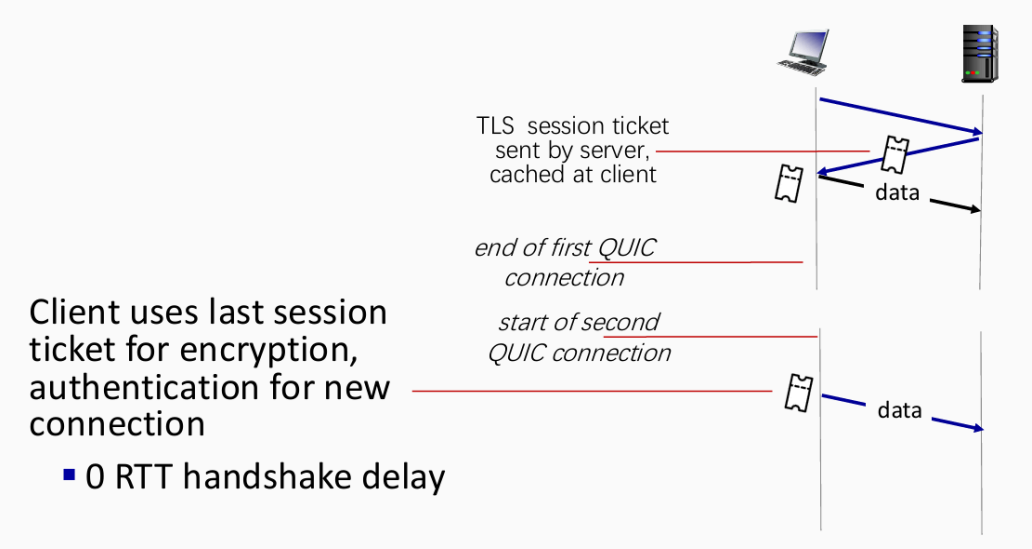
HTTP/3

- HTTP/2 over single TCP connection means:
    - recovery from packets loss still stalls all object transmission
      Packet丢失的恢复仍然会停止所有对象的传输
    - No security over raw TCP connection
- HTTP/3: adds security, per object error-and congestion-control over UDP
    - QUIC: Quick UDP Internet Connection

- application-layer protocol, on top of UDP
    - increased performance of HTTP
- error and congestion control(UDP but TCP like)
- connection establishment: reliability, congestion control, authentication, encryption, state established in one RTT
- TCP + TLS vs QUIC
    - TCP(reliability, congestion control state) + TLS(authentication, crypto state)
      2 serial handshakes(4 handshakes)
    - QUIC(reliability, congestion control, authentication, crypto state)
      1 handshake



- 0- RTT Connection establishment



Lecture 3 Application Layer: DNS / P2P / Socket

RESTful API

- Representational State Transfer API
- A set of recommended styles/rules for API design
    - Use of standard HTTP methods(GET, POST, PUT, DELETE, etc.) to operate on resources
    - Resources are identified via URIs where each URI represents a resource
      通过URI标识资源

  - Statelessness:服务器不在请求之间存储任何客户端会话信息
  - Use of standard HTTP status code to represent the outcome of requests
  - GET / users / 123 | POST / users | PUT / users / 123 | DELETE / users /123

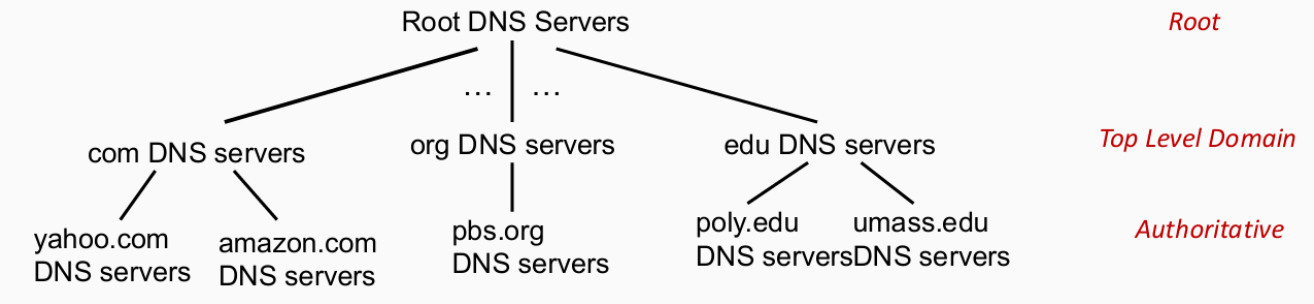
Domain Name System(DNS)

- Application-layer protocol:
  - C/S architecture
  - UDP(port 53)
  - hosts, name servers communicate to resolve names(name / address translation)
    主机，名称服务器通信以解析名称（名称/地址转换）
- Distributed database implemented in hierarchy of many name servers

DNS Services

- Hostname to IP address translation(A)
- Host aliasing(cname)
- Mail server aliasing(mx)
- Load distribution
  - Replicated复制 Web servers: many IP addresses correspond to one name

DNS Structure



- Client wants IP for www.amazon.com
  - Client queries root server to find com DNS server
  - Client queries .com DNS server to get amazon.com DNS server
  - Client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: root name servers

- Contacted by local name server that can not resolve name
- Root name server:
  - Contacts authoritative name server if name mapping not known
  - Get mapping

- Returns mapping to local name server

DNS: TLD, authoritative servers

- Top-level domain(TLD) servers
  - Responsible foe com, org, edu, aero, jobs, museums, and all Top-level country domains, eg.: cn, uk, fr, ca, jp
- Authoritative DNS servers
  - Organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
  - Can be maintained by organization or service provider

DNS: local DNS name server

- Each ISP (residential ISP, company, university) has one
- Also called "default name server"
- When host make DNS query, query is sent to its local DNS server
  当主机进行DNS查询时，查询被发送到其本地DNS服务器
  - Has local cache of recent name-to address translation pairs(but may be out of date)
    有最近名称到地址转换对的本地缓存(可能过时)
  - Acts as proxy, forwards query into hierarchy
    充当代理 将查询转发到层次结构中

DNS name resolution example

- Host at XJTLU wants IP address for www.feimax.com
- Iterated query 迭代查询
  - contacted server replies with name of server to contact
  - "I don't Know this name, but ask this server"
- Recursive query 递归查询
  - Puts burden of name resolution on contacted name server
    将名称解析的负担放在被联系的名称服务器上
  - Heavy load at upper levels of hierarchy
    层次结构上层负载重

DNS: caching, updating records

- Once(any) name server learns mapping, it caches mapping
  - Cache entries timeout(disappear) after some time(TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited

- Cached entries may be out-of-date
    - if name host changes IP address, may not be known Internet-wide until all TTLs expire

DNS records

DNS: distributed database storing resource records(RR)
`RR format:(name,value,type,ttl)`

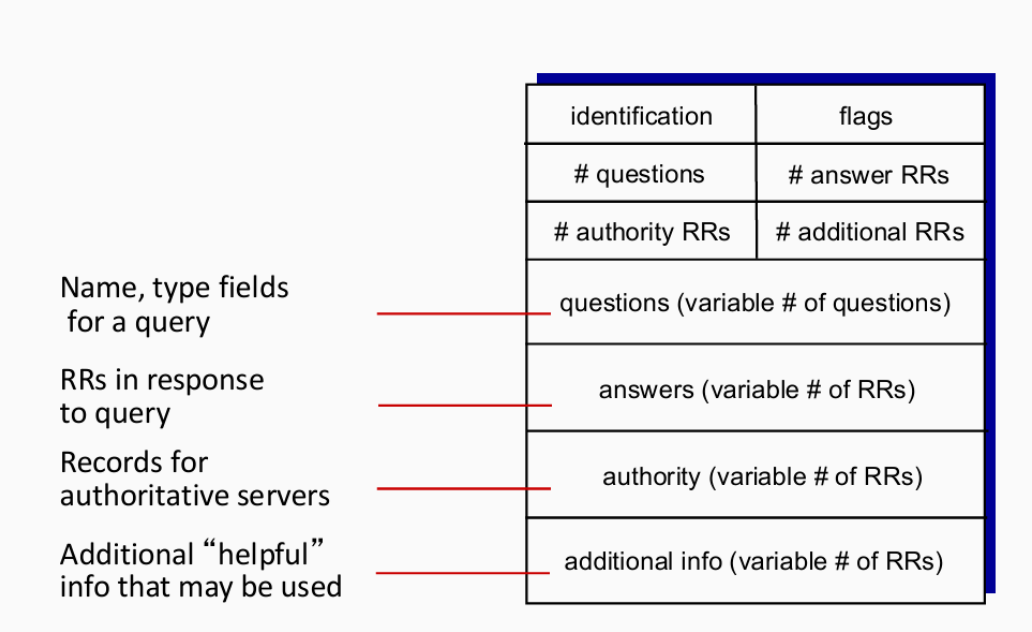
- type = A
    - name is hostname
    - value is IP address
- type = NS
    - name is domain(e.g. doo.com)
    - value is hostname of authoritative name server for this domain
- type = CNAME
    - name is alias name for some "canonical"(the real) name
    - www.taobao.com is really www.taobao.com.danuoyi.tbcache.com
    - value is canonical name
- type = MX
    - value is name of mail server associated with name

DNS protocol, messages

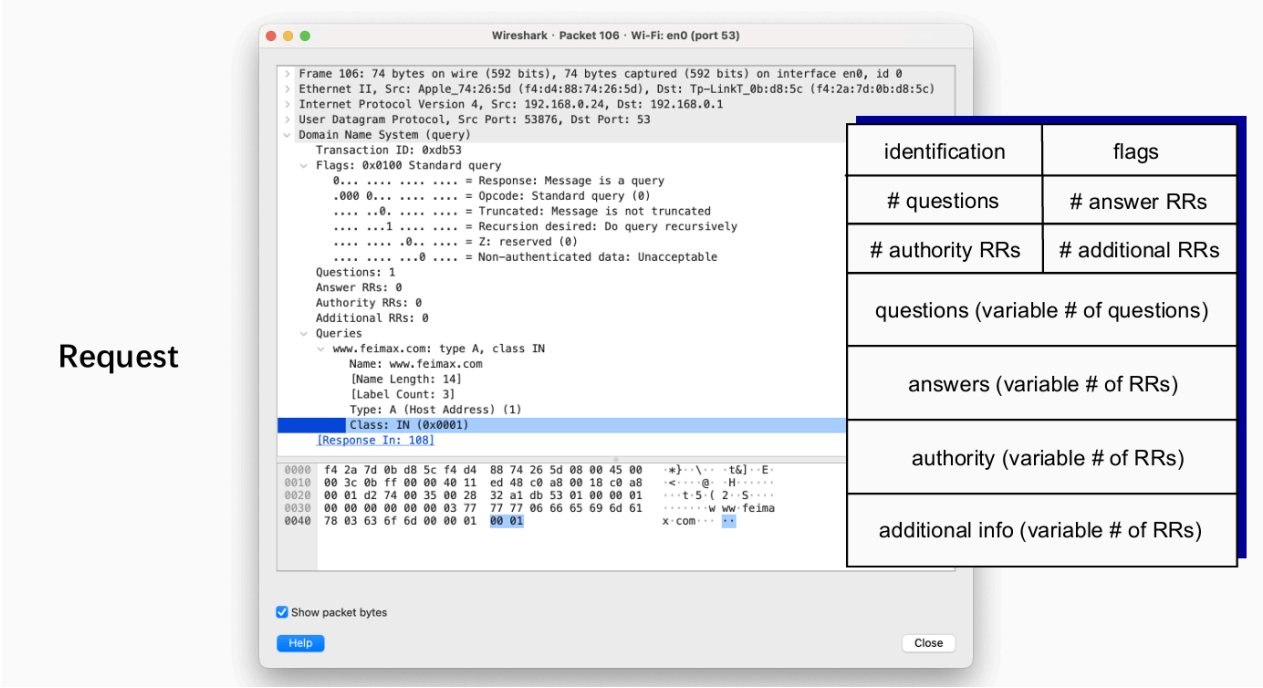
Query and reply messages, both with same message format

- Message header:
    - identification: 16bit # for query, reply to query uses same #
    - flags:
        - query or reply
        - recursion desired
        - recursion available

- reply is authoritative



- Request message



- Transaction ID 用来匹配请求和响应 / 响应里会用同一个ID
- Flags : 0x0100 standard query
  - RD =1 表示 客户端希望DNS服务器进行递归查询
- 计数字段(4个)
  - Questions: 1
  - Answer RRs: 0
  - Authority RRs: 0
  - Additional RRs: 0
- Queries(问题部分)
  - Name 要查询的域名
  - Type 查询类型 A记录 要IPv4地址

        - Class: IN 互联网地址空间
- Response message
    - 多一行 answer 与DNS records一样

P2P Applications

- No always-on server
- Arbitrary end systems directly communicate
- Peers change IP addresses 对等节点改变IP地址

File distribution: C/S vs P2P

- C/S
    - server transmission: must sequentially send (upload) N files copies
        - time to send 1 copy: F/u_s
        - time to send N copy: NF/u_s
    - client: each client must download file copy
        - d_{min} = min client download rate
        - max client download time: F/d_{min}
    - time to distribute F to N clients using client-server approach
      $D_{c-s} > \max\{NF/u_s, F/d_{min}\}$
      increases linearly in N
- P2P
    - server transmission: must sequentially send(upload) at least one file copies:
        - time to send 1 copy: F/u_s
    - client: each client must download file copy
        - min client download time: F/d_{min}
    - client: as total must download NF bits
        - max upload rate(limiting max download rate) is $u_s + \sum u_i$
    - time to distribute F to N clients using P2P approach
      $D_{P2P} > max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$
      increases linearly in N, but so does this, as each peer brings service capacity

BitTorrent

- The throughput increases with the number of downloaders via the efficient use of network bandwidth

---

- Tracker: a central server keeping a list of all peers participating in torrent
  跟踪器：一个中央服务器，跟踪参与种子的所有对等节点
- Torrent: group of peers exchanging chunks of a file
- To share a file or group of files, the initiator first creates a .torrent file, a small file that contains
  - Metadata about the files to be shared
    - SHA-1 hashes of all pieces
    - Piece size
    - Length of the file
    - A tracker reference
  - Information about the tracker, the computer that coordinates the file distribution
- Peer joining torrent 对等节点加入种子
  - 没有pieces，但会随时间从其他对等节点积累
  - 向追踪器注册以获取对等节点列表，连接到一部分对等节点（邻居）
- While downloading, peer uploads pieces to other peers
- Peers may come and go
- Once peer has entire file, it may(selfishly) leave or remain in torrent
- Leecher $\rightarrow$ Seeder
  - As soon as a leecher has a complete piece, it can potentially share it with other downloaders
  - Eventually each leecher becomes a seeder by obtaining all the pieces, and assembles the file. Verifies the"checksum" of the file

---

Pieces selection policy

- the order in which pieces are selected by different peers is critical for good performance
- inefficient policy, then peers may end up in a situation where each has all identical set of easily available pieces, and none of the missing ones
- If the original seed is prematurely taken down, then the file cannot be completely download
- Micro view
  - Rarest First
    - General rule
    - Determine the pieces that are most rare among your peers, and download those first
    - This ensures that the most commonly available pieces are left till the end to download

- Random First Piece
  - Special case, at the beginning
  - Important to get a complete piece ASAP
  - Select a random piece of the file and download it
- Endgame Mode
  - Special case
  - Near the end, missing pieces are requested from every peer containing them
  - this ensures that a download is not prevented from completion due to a single peer with a slow transfer rate
  - Some bandwidth is wasted, but in practice, this is not too much

Internal Mechanism

- Choking Algorithm 阻塞算法
  - choking是暂时拒绝上传，用于避免搭便车者（只下载不上传的人）
  - Tit-for-tat strategy
- Optimistic Unchoking 乐观解阻塞
  - 一个对等节点向当前以最高速率向它发送块chunks的四个对等节点发送块
    - 其他对等节点被阻塞（不从它们那里接受块）
    - 每10s重新评估前4名
  - 每30s，随机选择另一对等节点，并开始向其发送块
    - “乐观解除阻塞”这个对等节点
    - 新选择的对等节点可能加入top 4
      - 发现当前未使用的，比现在正在使用的连接更好的连接
      - 为新对等节点提供最低限度的服务

Upload- Only mode

- Once download is complete, a peer can only upload
- Upload to those with the best upload rate. This ensures that pieces get replicated faster, and new seeders are created fast

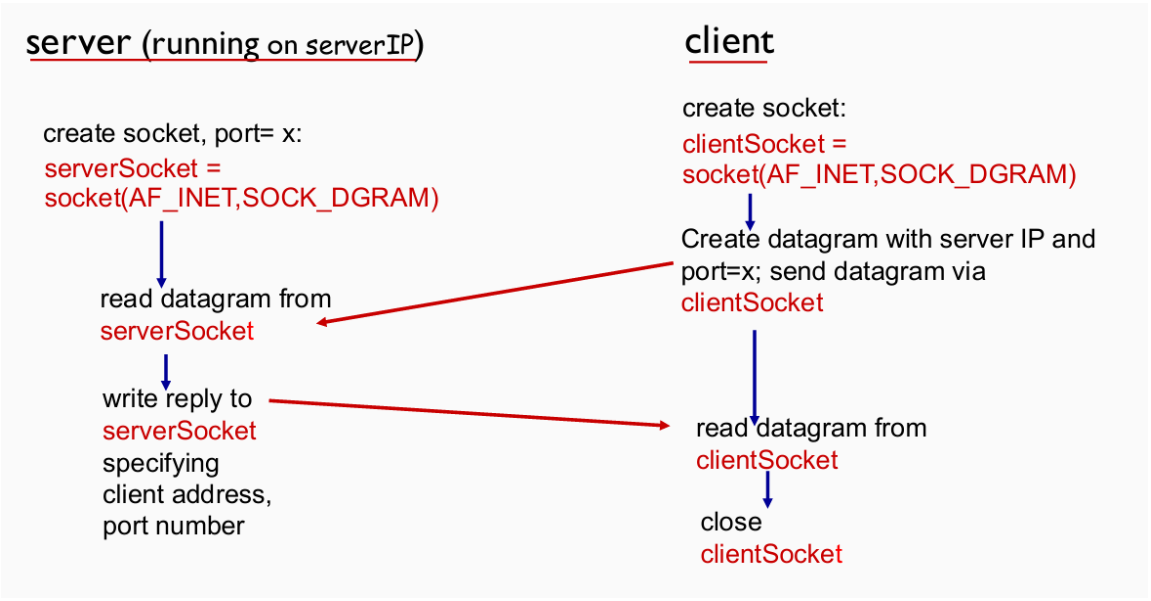
Socket programming

Socket: door between application process and end-end-transport layer

UDP socket programming

- no connection between C/S
  - No handshaking before sending data

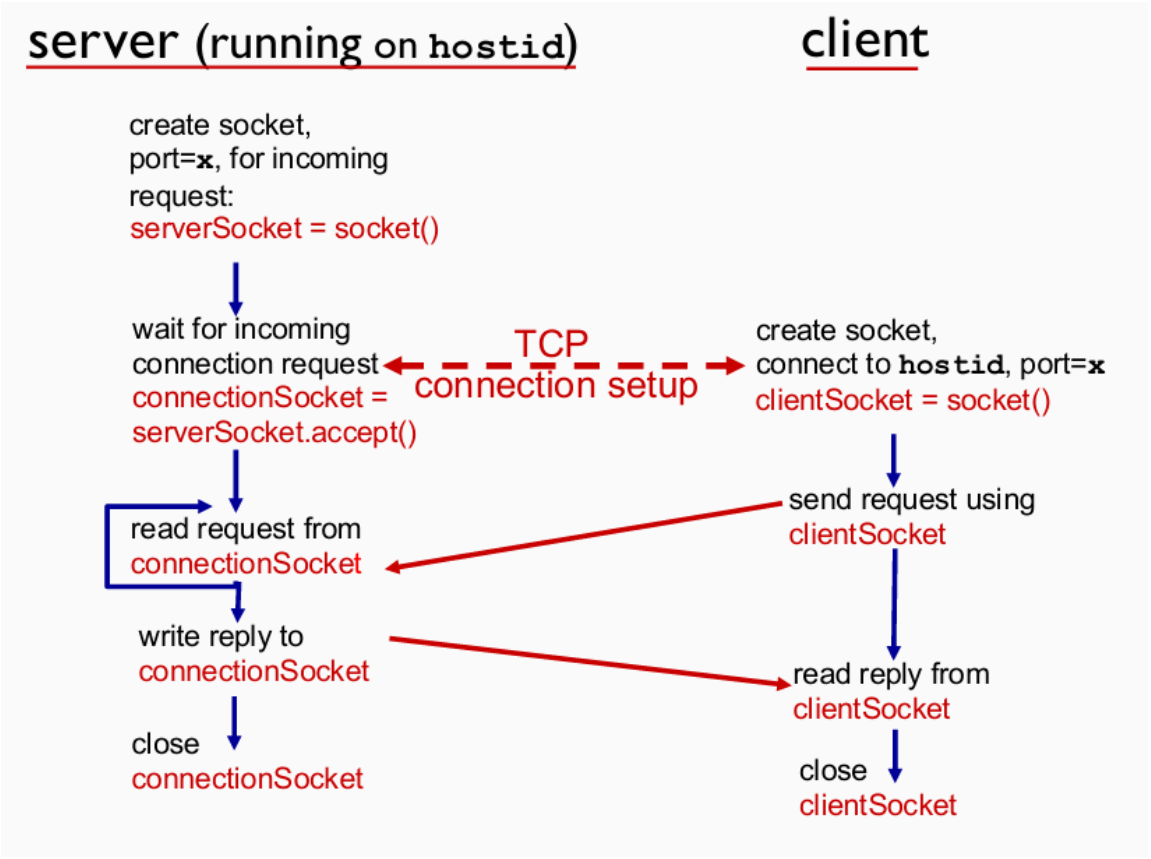
- Sender explicitly attaches IP destination address and port # to each socket
- Receiver extracts sender IP address and port # from received packet



- Application viewpoint
  - UDP provides unreliable transfer of groups of bytes(datagrams)

TCP socket programming

- Client must contact server
- Application viewpoint
  - TCP provides reliable, in-order byte-stream transfer("pipe")between client and server



Lecture 4 Transport Layer: multiplexing / UDP / rdt

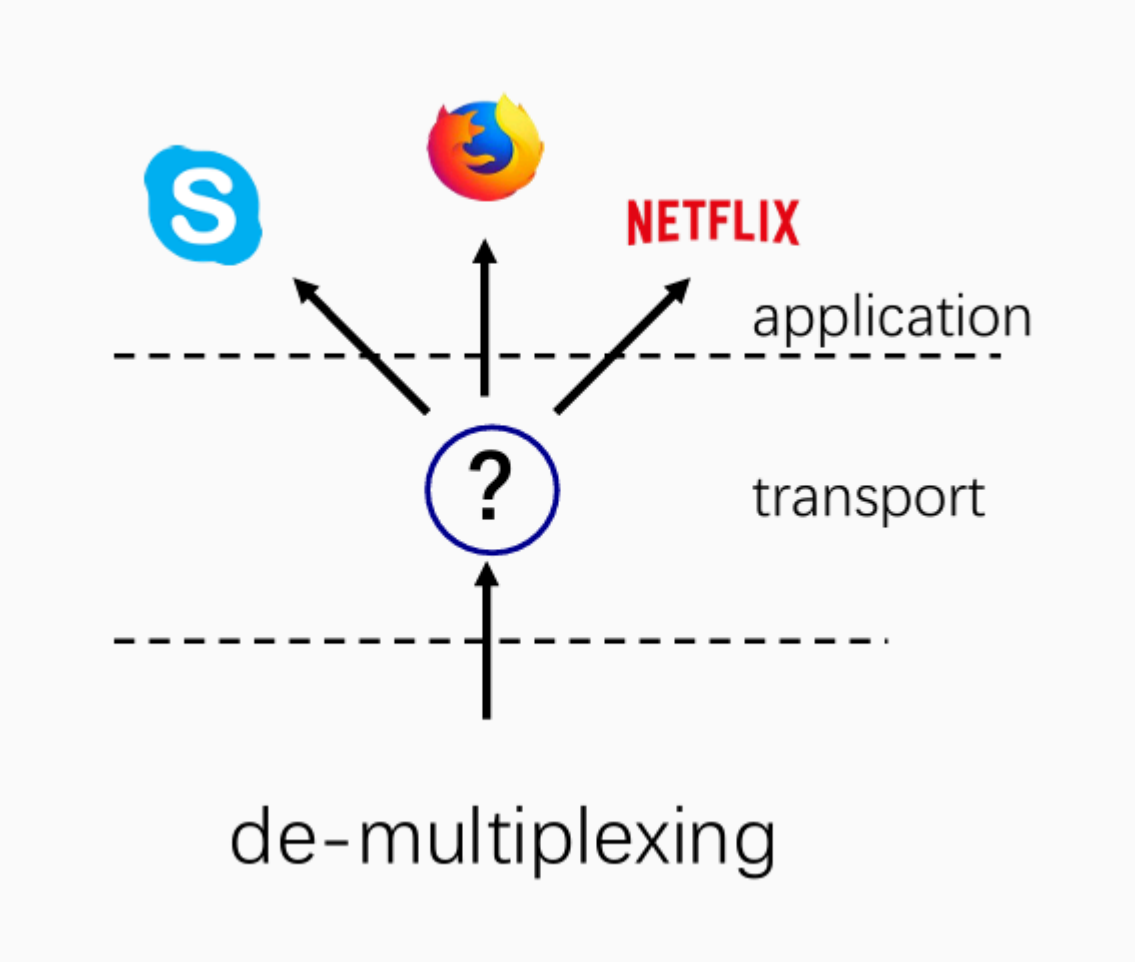
Transport Layer services overview

- Transport Layer Actions
  - Sender:
    - is passed an application layer message
    - determines segment header fields values
    - creates segment
    - passes segment to IP
  - Receiver
    - receives segment from IP
    - checks header values
    - extracts application layer message
    - demultiplexes message up to application via socket
- Two principal internet transport protocols
  - Unreliable, unordered delivery: UDP
    - No-frills extension of "best-effort" network layer
  - Reliable, in-order delivery: TCP
    - Congestion control
    - Flow control
    - Connection setup
  - Services not available
    - Delay guarantees
    - Bandwidth guarantees

Multiplexing and demultiplexing

- Multiplexing as sender
  - handle data from multiple sockets, add transport header(later used for demultiplexing)
    添加运输层头部
- Demultiplexing as receiver
  - use header info to deliver received segments to correct socket
- 复用(multiplexing): 多条数据流共用一条链路发送，在发送端，将来自多个源的数据，通过标识信息组合到同一信道中传输（邮局把恒多人的信装上同一辆车）application layer → transport layer(+ source port number) → IP layer
- 解复用(demultiplexing): 收到数据后，把它交给正确的应用/进程，在接收端，根据报文中的标识字段，将数据交付给正确的上层实体（到站后按地址分给不同的进程）

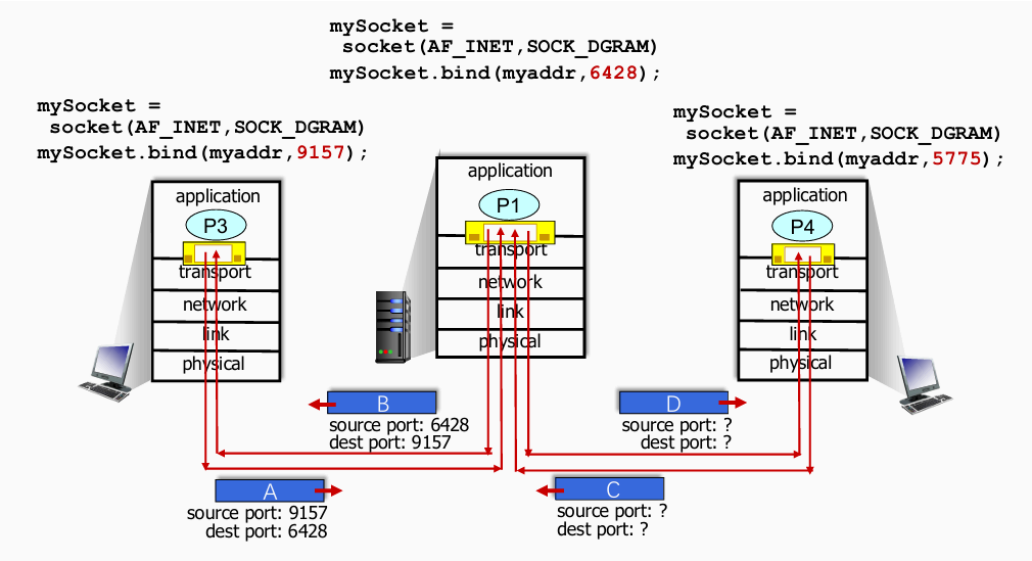
- why 复用？ 把数据交给正确进程，网络层(IP)只认识主机，不认识进程，所以传输层使用端口号（其中TCP还看4-tuple）来区分不同应用



Demultiplexing work

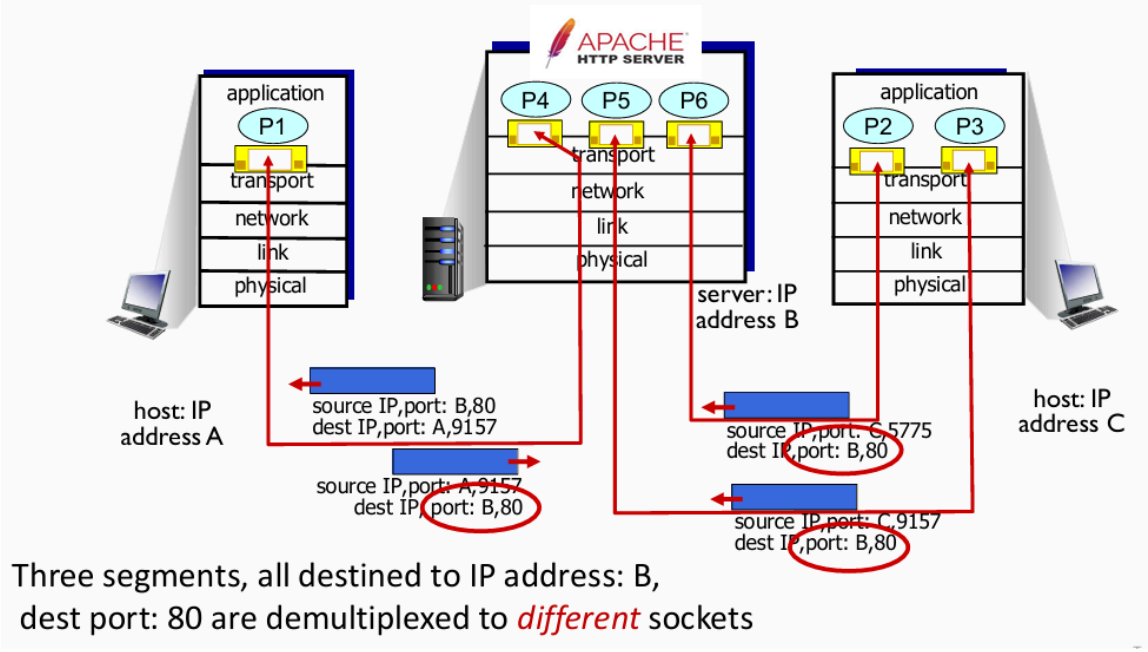
- Host receives IP datagrams
  - Each datagram has source IP address, destination IP address
  - Each datagram carries one transport layer segment
  - Each segment has source, destination port number
- Host uses IP address & port numbers to direct segment to suitable socket
- Connectionless demultiplexing(UDP)
  - When creating datagram to send into UDP socket, must specify:
    - Destination IP address

- Destination port number



  - 虽然如此 但实际上只检查目的端口号 不同源IP地址/源端口号的IP数据包将被导向相同socket

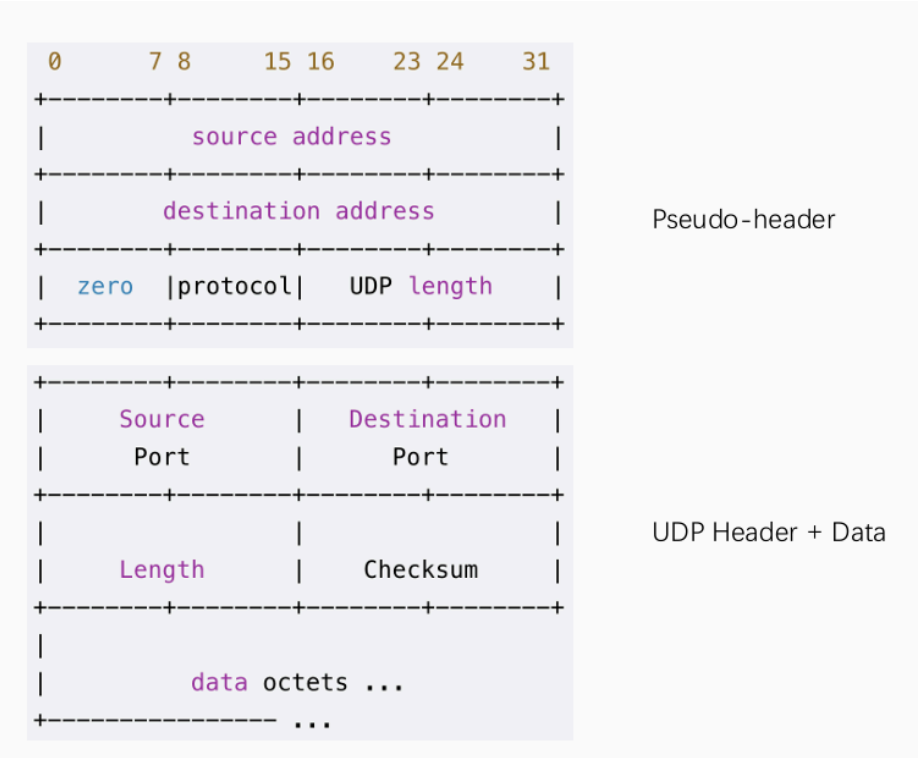
- Connection-oriented demux(TCP)
  - TCP socket identified by 4-tuple
    - source IP address
    - source port number
    - dest IP address
    - dest port number
  - Demux: receiver uses all four values to direct segment to appropriate socket
    接收方使用四个值将段（segment）导向适当的套接字
  - Server host may support many simultaneous TCP sockets
    - each socket identified by its own 4-tuple
  - Web servers have different sockets for each connecting client



Connectionless transport: UDP (User Datagram Protocol)

- Feature:
  - Simple and straightforward
  - Best effort
  - Lost
- Connectionless
  - No handshaking
  - Each UDP segment handled independently of others(out of order to APP)
- UDP use:
  - Streaming multimedia apps
  - DNS
  - HTTP / 3
- Reliable transfer over UDP
  - Add reliability at application layer
  - Application-specific error recovery
- UDP: Transport Layer Actions
  - sender actions:
    - is passed an application-layer message 接收应用层消息
    - determines UDP segment header fields values 确定UDP段头字段值
    - creates UDP segment
    - passes segment to IP
  - receiver actions
    - receives segment from IP
    - checks UDP checksum header value
    - extracts application layer message
    - demultiplexes message up to application via socket
- UDP: segment header
  - source port#
  - dest port#
  - length
  - checksum
    - Goal: detect "errors" in transmitted segment
    - Sender
      - Treat segment contents, including header fields, as sequence of 16-bit integers
      - Checksum: addition(one's complement sum) of pseudo header, UDP header and UDP data

        - Sender puts checksum value into UDP checksum field
    - Receiver
        - Compute checksum of received segment
        - Check if computed checksum equals field value
            - NO -error detected
            - YES - no error detected, but maybe errors nonetheless
    - Pseudo Header
        - source IP
        - dest IP
        - protocol(UDP 18)
        - UDP length



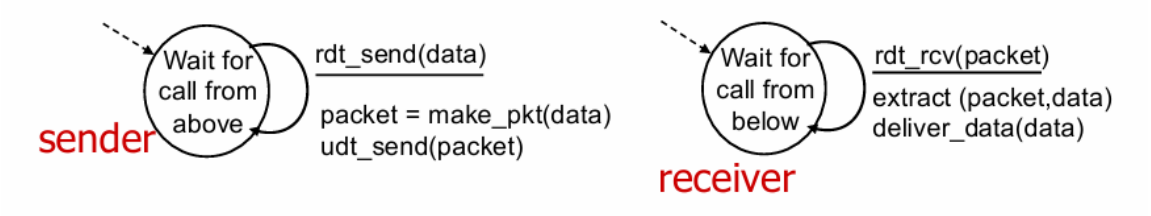
Principles of reliable data transfer

Reliable data transfer protocol(`rdt`): interfaces

- `rdt_send()` :called from above (e.g. by app.). Passed data to deliver to receiver upper layer
  由上层调用 传递数据给接收器上层
- `deliver_data()` :called by `rdt` to deliver data to upper layer
  由 `rdt` 调用 传递数据给上层
- `udt_send()` : called by `rdt` to transfer packet over unreliable channel to receiver
  由 `rdt` 调用，通过不可靠信道将数据包传递给接收方
- `rdt_rcv()` : called when packet arrives on receiver side of channel

`rdt1.0` : reliable transfer over a reliable channel

- 完全理想
- Underlying channel perfectly reliable
  - No bit errors
  - No loss of packets
- Separate FSMs for sender, receiver:
  - Sender sends data into underlying channel
  - Receiver reads data from underlying channel



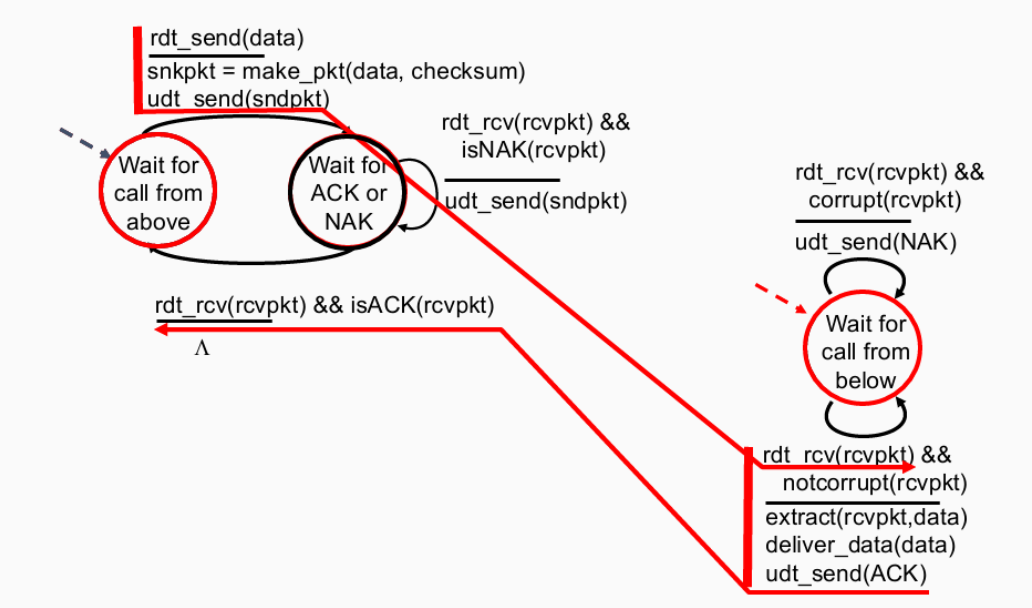
`rdt2.0` : channel with bit errors

- Underlying channel may flip bits in packet
  - checksum to detect bit errors
- Recover from errors:
  - ACKs: receiver tells sender that `pkt` received OK
  - Negative acknowledgements(NAKs):
    - receiver tells sender that `pkt` had errors
  - New mechanisms in `rdt2.0` (beyond `rdt1.0`)
    - Error detection
    - Receiver feedback: control msgs(ACK, NAK) rcvr → sender
- FSMs
  - `rdt2.0` : operation with no errors
    - Sender: Wait for call from above
      `rdt_send(data)`
      `sndpkt = make_pkt(data,checksum)`
      `udt_send(sndpkt)`
      上层交付数据，封装成带checksum的分组，发送到不可靠信道
      → Wait for ACK/NAK
    - Receiver: Wait for call from below
      `rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)`
      `extract(rcvpkt, data)`
      `deliver_data(data)`
      `udt_send(ACK)`
      分组没损坏，提取数据，交付给上层，回复ACK

- Sender: receive ACK

  `rdt_rcv(rcvpkt) && isACK(rcvpkt)`

  确定发送成功，回到wait for call from above, 准备发送下一个数据



- `rdt2.0`: Error scenario
  - 数据分组在途中损坏
  - Receiver

    `rdt_rcv(rcvpkt) && corrupt(rcvpkt)`

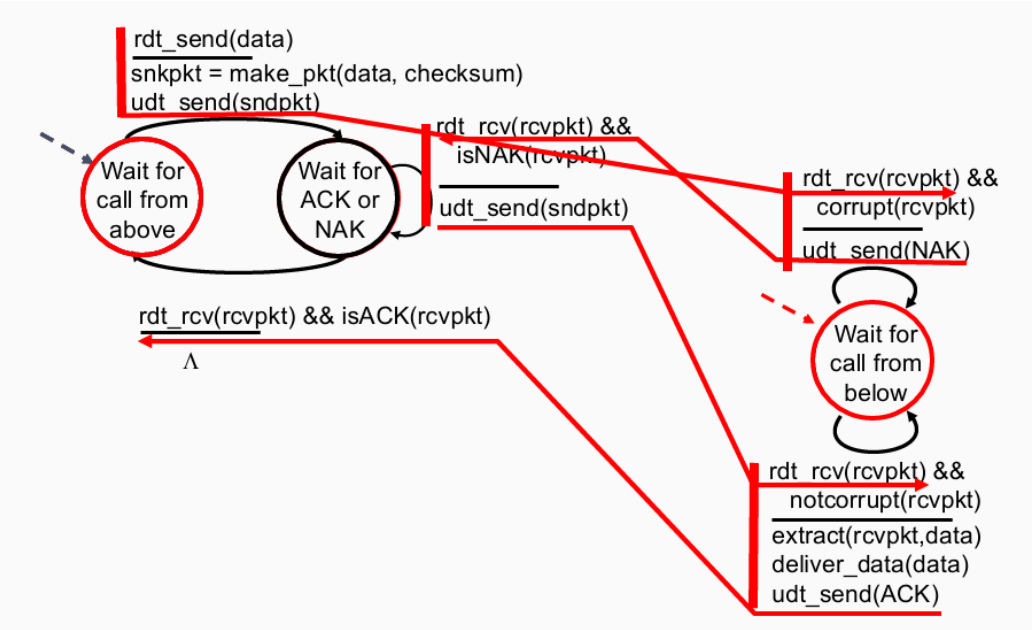
    `udt_send(ACK)`

    校验和不通过，不交付数据，发送NAK
  - Sender

    `rdt_rcv(rcvpkt) && isNAK(rcvpkt)`

    `udt_send(sndpkt)`

    知道上一次数据出错 重传原来分组

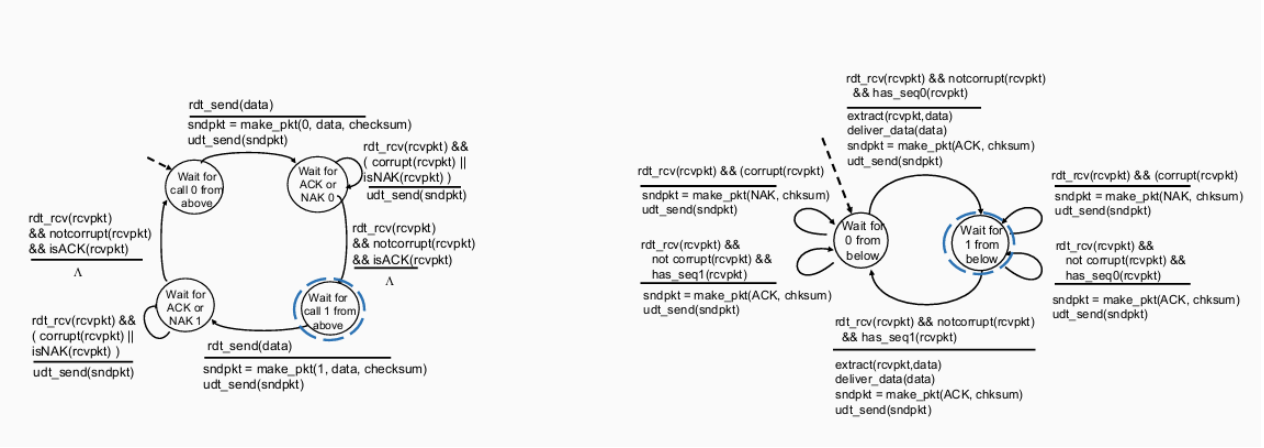


- `rdt2.0` has a fatal flaw
  - if ACK/NAK corrupted
    - Sender don't know what happen at receiver

    - can't just retransmit → possible duplicate
- Handling duplicates
    - Sender retransmits current pkt
    - Sender adds sequence number to each pkt
    - Receiver discards(doesn't deliver up) duplicate packet
      接收方丢弃重复数据包

`rdt2.1`: sender, handles garbled ACK/NAKs

- 给每个数据包加1-bit 序号（0/1）
- FSMs

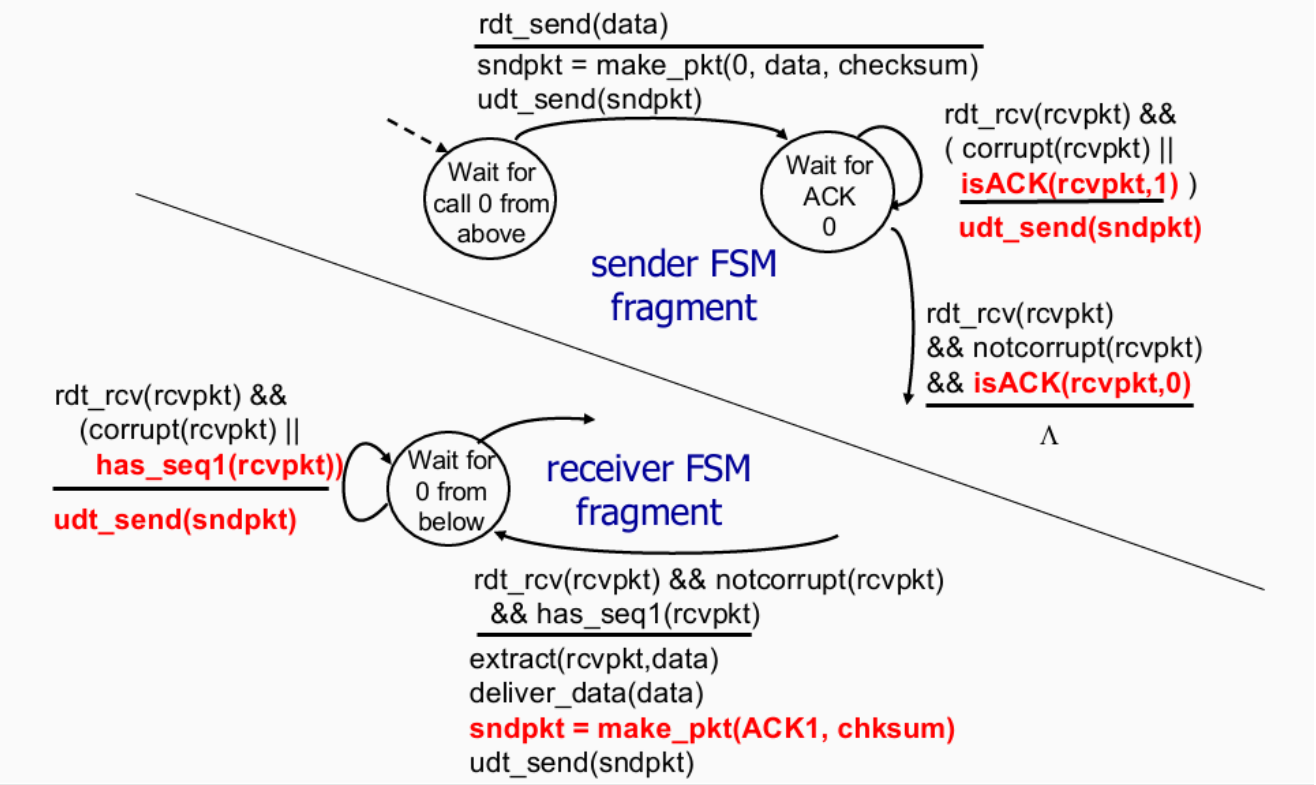


    - Sender: Wait for call 0 from above
      `rdt_send(data)`
      `sndpkt = make_pkt(0, data, checksum)`
      `udt_send(sndpkt)`
      发送序号为0的数据包 → Wait for ACK/NAK 0
    - Sender: Wait for ACK / NAK0
        - ACK0
          `rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt)`
          → Wait for call 1 from above
        - wrong / NAK
          `rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || isNAK(rcvpkt))`
          → retransmit seq = 0的数据包
    - Sender: Wait for call 1 from above
    - Sender: Wait for ACK or NAK 1, mirror as ACK/NAK0

    - Receiver: receive correct seq = 0
      `rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq0(rcvpkt)`
      提取数据，交付上层，发送ACK0，→ Wait for 1 from below

    - Receiver: wrong / seq =1(重复包)
      `rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || has_seq1(rcvpkt))`
      不交付数据，发送ACK1（上一次正确包的ACK）
    - Receiver: Wait for 1 from below
      逻辑对称
- Discussion
    - Sender:
        - seq# added to pkt
        - Two seq. #'s (0,1) will suffice 应为 2.1属于停等协议 发送方发送一个数据包后必须等待接收方的ACK / NAK，在任意时刻信道上最多只有一个未确认的数据包
        - Must check if received ACK/NAK corrupted
        - Twice as many states
    - Receiver
        - must check if received packet is duplicate
            - state indicates whether 0/1 is expected pkt seq#
        - Note: receiver can't know if its last ACK/NAK received OK at sender

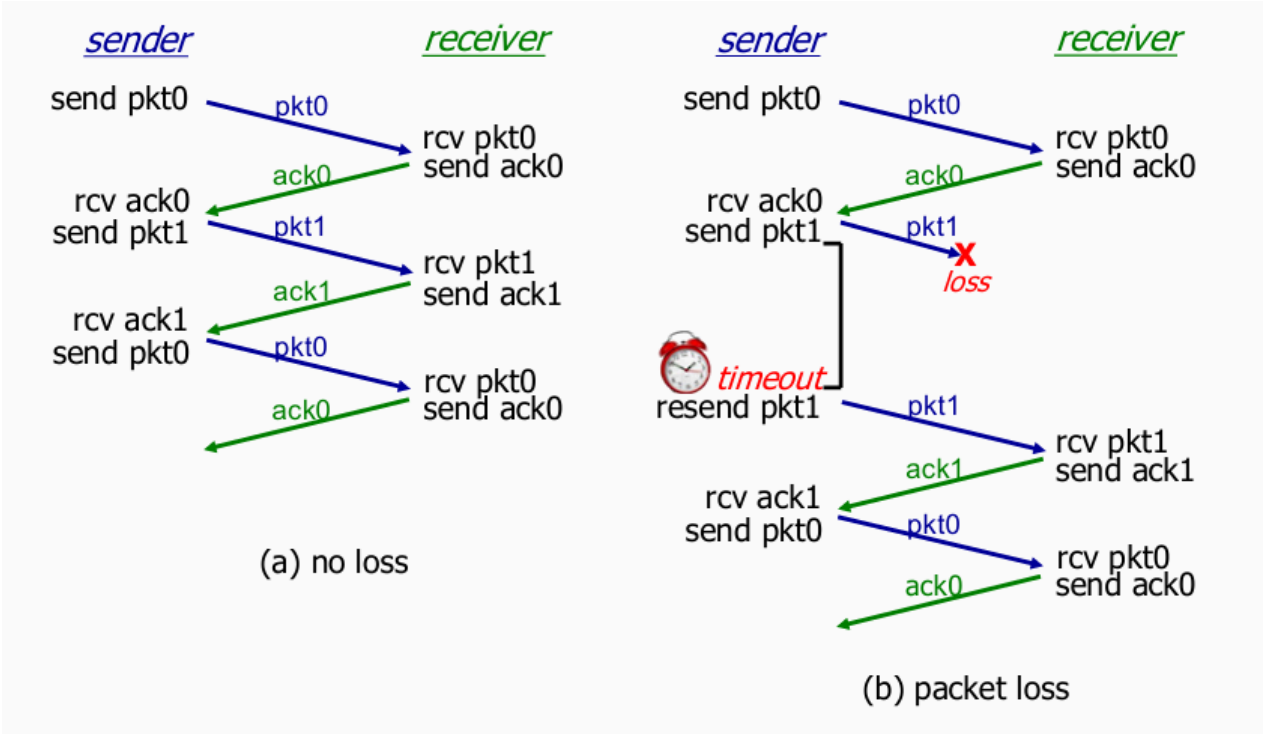
`rdt2.2` : a NAK-free protocol

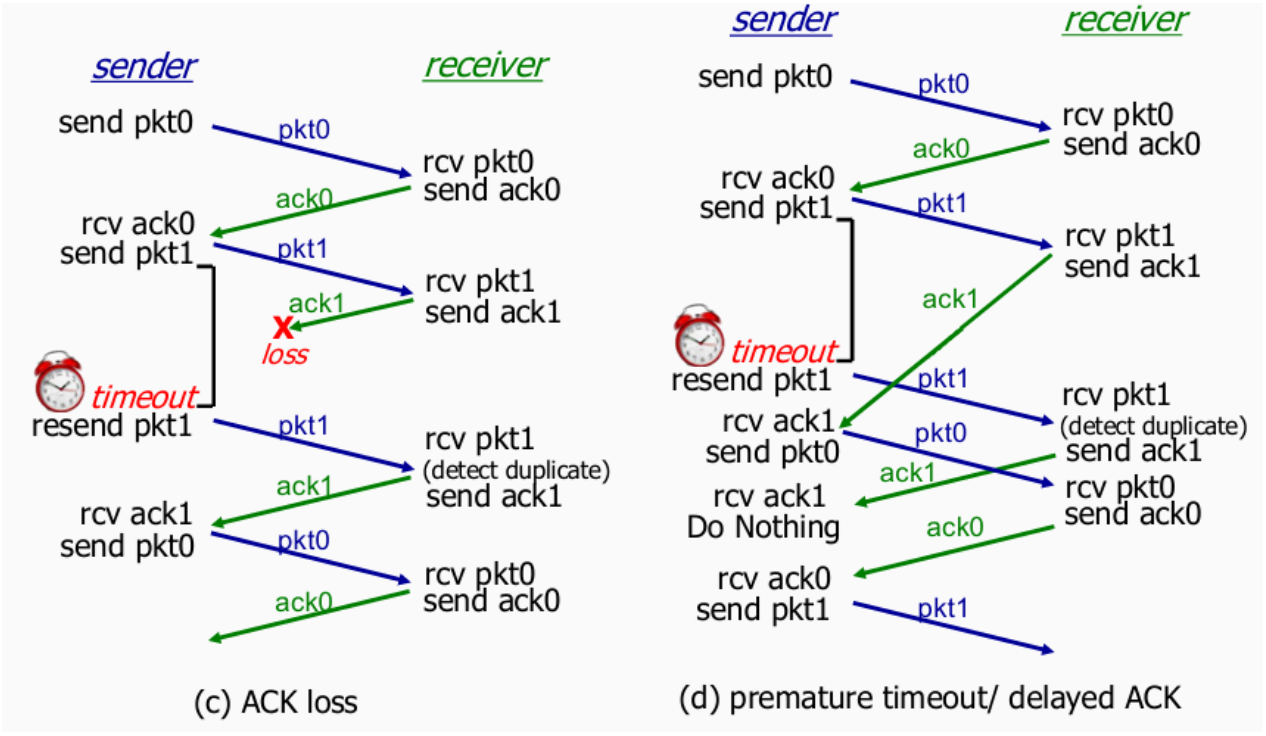


`rdt3.0` : Channels with errors and loss

- 核心思想： 开启一个计时器 来记录合理时间收ACK，但是仍然是stop-and-wait operation
- Underlying channel can also lose packets(data,ACKs)

    - checksum, seq#, ACKs, retransmissions...
- Sender waits "resonable" amount of time for ACK
    - Retransmits if no ACK received in this time
    - If pkt(or ACK) just delayed(not lost):
        - retransmission will be duplicate, but seq. #'s already handles this
        - Receiver must specify seq# of pkt being ACKed
    - Requires countdown timer
- `rdt3.0` in action





- Performance of `rdt3.0`
    - performance stinks

- 1 Gbps link,15ms prop.delay, 8000 bit packet

$$D_{trans} = L/R = \frac{8000bits}{10^9 bits/sec} = 8microsecs$$

- U_{sender}: utilization - fraction of time sender busy sending

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

Lecture 5 Transport Layer: TCP

Pipelined protocols

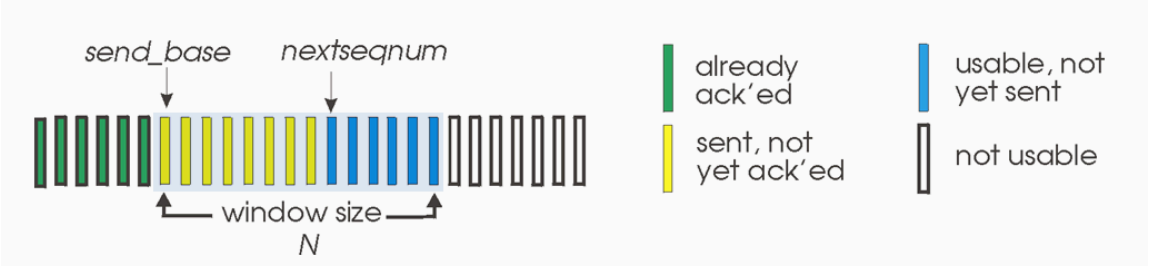
to increase utilization

- Pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged packets
  允许多个在途且尚未确认的pkts
  - Range of sequence of numbers must be increased
  - Buffering at sender and / or receiver

Two forms: Go-Back-N and Selective repeat

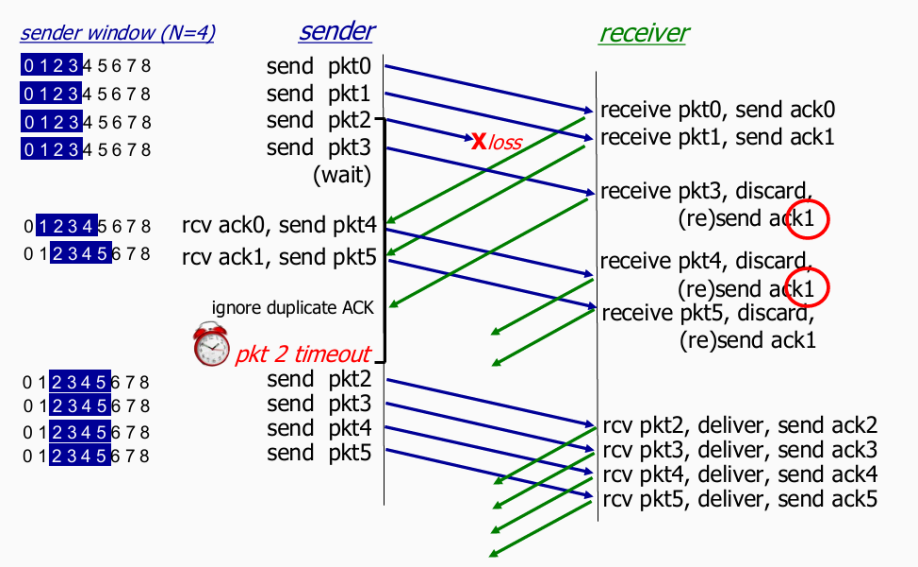
Go-Back-N(GBN):

- Sender can have up to N unacked packets in pipeline
- Receiver only sends cumulative ACK
  - Doesn't ACK packet if there is a gap
- Sender has timer for oldest unACKed packet
  - when timer expires, retransmit all unacked packets
- GBN in action
  - Sender:"window" of up to N, consecutive transmitted but unACKed pkts



    - k-bit seq# in pkt header
    - cumulative ACK: ACK(n): ACKs all packets up to, including seq# n
      - On receiving ACK(n): move window forward to begin n+1
    - timer for oldest in-flight packet
    - timeout(n): retransmit packet n and all higher seq# packets in window

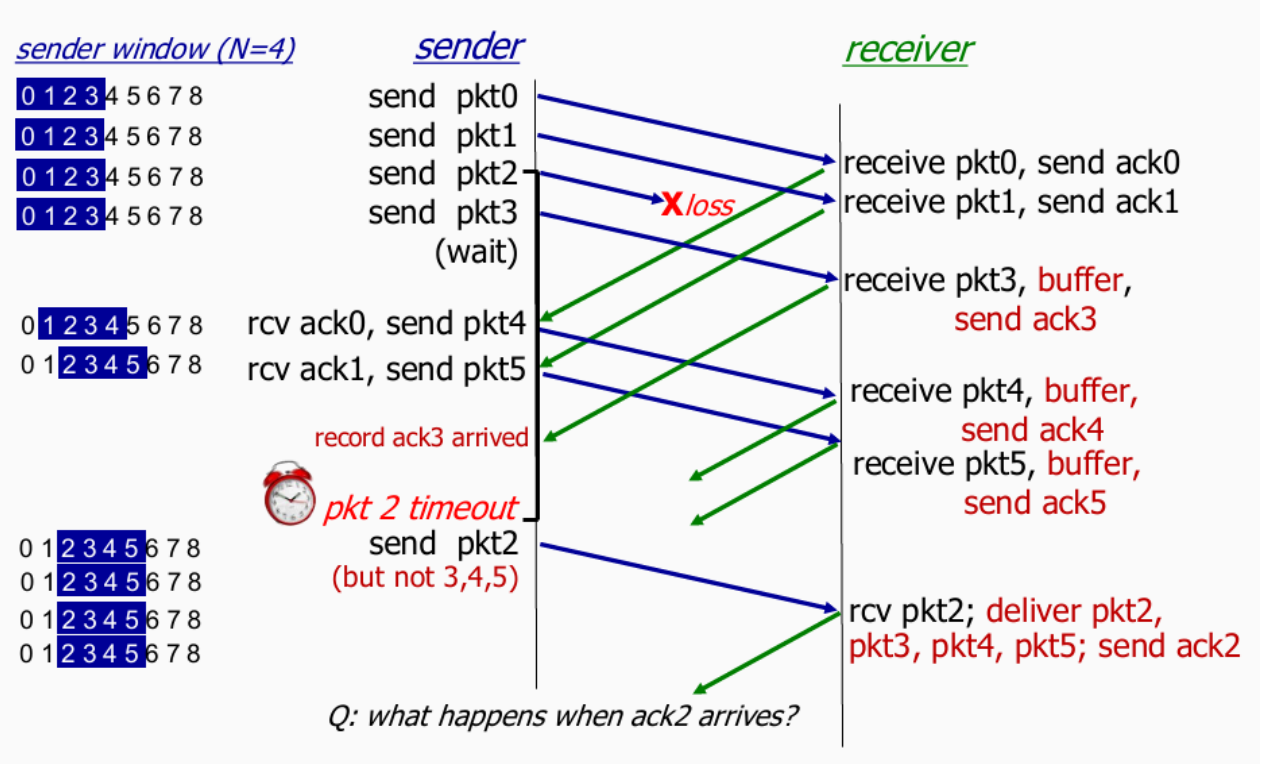
- Receiver: ACK-only
  - always send ACK for correctly-received packet so far, with highest in-order seq#
    总是为到现在为止正确几首的最高按序序列号数据包发送ACK（只接受顺序的）
    只发最高顺序
    - may generate duplicate ACKs
    - need only remember `rcv_base`



Selective Repeat(SR)

- Sender can have up to N unacked packets in pipeline
- Receiver sends individual ack for each packet
  - buffers packets, as needed, for in-order delivery to upper layer
- Sender maintains timer for each unacked packet
  - when timer expires, retransmit only that unacked packet
  - maintains "windows" over N consecutive seq # s
    - limits pipelined, "in flight"packets to be within this window

- SR in action



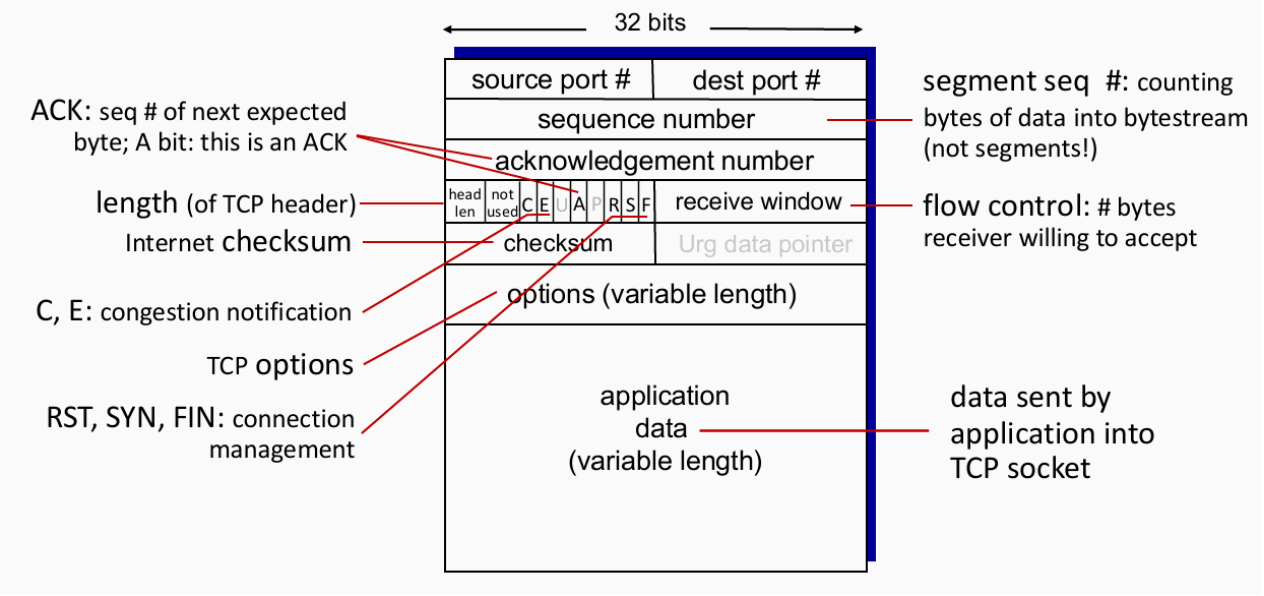
- A Dilemma
  - 窗口大小过大 + 有限的序列号空间 → 接收方将旧包当成新包

$$W \leq \frac{N}{2}$$

  - When $W > \frac{N}{2}$:
    - |seq #| = 4 {0,1,2,3}
    - W = 3
    - Step 1 第一次发送
      - sender 连续发送：pkt0, pkt1, pkt2
      - receiver 正确收到并交付：0，1，2
    - Step 2 ACK 出问题
      - ack0, ack1 sender 收到
      - ack2 loss
      - sender以为pkt2 没成功， receiver其实已经交付了pkt2
    - Step 3 序列号回绕
      - sender继续发送pkt3, pkt0, pkt1
      - 但是pkt0，pkt1 是新一轮的数据，但序列号和旧的一样
    - Step 3 歧义发生
      - receiver发现一个pkt0
      - 无法判断这是旧的pkt0重传还是新一轮的pkt0
      - 就窗口与新窗口发生重叠

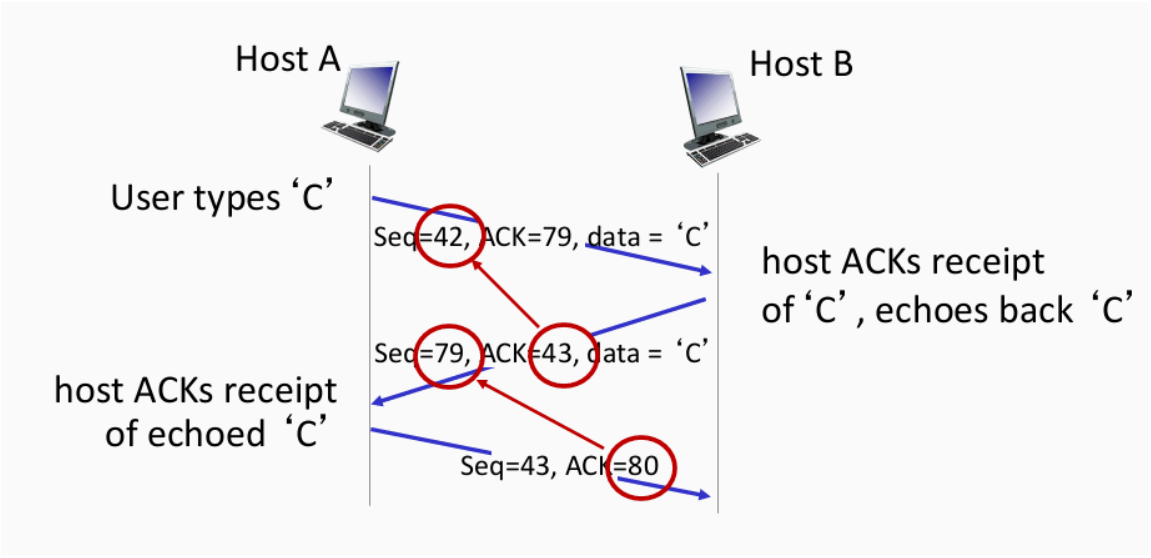
TCP: connection-oriented transport

- Overview
  - Point to point: one sender, one receiver
  - Reliable, in-order byte stream
  - Pipelined: TCP congestion and flow control set window size
  - Full duplex data: bidirectional data flow in same connection
  - Connection-oriented: handshaking(exchange of control msgs) inits sender, receiver state before data exchange
  - Flow controlled: sender will not ovewhelm receiver
- TCP segment structure



U: UAG, urgent data 紧急数据(generally not used)
A: seq # of next expected byte
P: PSH push data now(generally not used)
C,E: congestion notification
R,S,F: RST|SYN|FIN connection management
receive window: flow control(#bytes receiver willing to accept)

- TCP sequence numbers, ACKs
  - Sequence numbers: byte stream "number" of first byte in segment's data
    段数据中第一个字节流的编号
    TCP 传的是字节流 每个字节都有一个编号， sequence number指本段数据中第一个字节的编号
    seq = 1000 数据长度 = 500bytes
    那么这个端携带的是: 字节1000-1499
  - Acknowledgements:
    - seq # of $\underline{nextbyte}$ expected from other side
    - commulative ACK
    - 如果receiver发ACK = 1500 标识0-1499都收到了 正在等1500



- Host A 给这个数据包编号为42，告诉B 我收到78以前的数据，正在等待ACK(79)
- Host B 确认并回显，B收到了A的Seq =42，且长度为1，所以B的ACK = 43
- Host A在Step 1说想要79 发Seq = 79
- Host A 收到了B发来的C A需要告诉B我收到了，为纯ACK包 不带数据

TCP round trip time(RTT), timeout

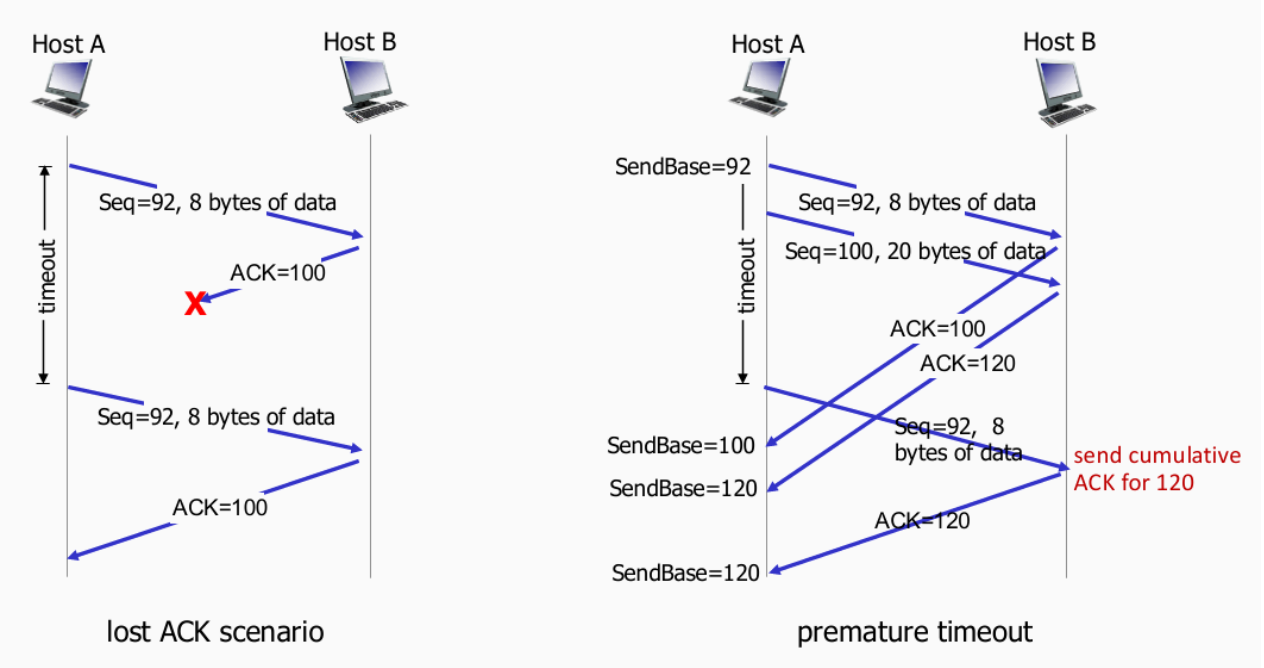
- How to set TCP timeout value
  - Longer than RTT
  - Too short: premature timeout, unnecessary retransmissions
  - Too long: slow reaction to segment loss
- The estimation of RTT
  - SampleRTT: measured time from segment transmission until ACK receipt
    从段传输到收到ACK所测量的时间
    - ignore retransmissions
  - $$EstimatedRTT_n = (1-\alpha) * EstimatedRTT_{n-1} + \alpha * SampleRTT$$
    - 指数加权移动平均 EWMA
    - Influence of past sample decreases exponentially fast 为了消除过去的影响
    - Typically $\alpha = 0.125$
    - (α 过大 EstimatedRTT剧烈波动 受当前网络瞬间变化影响太大；α 很小，EstimatedRTT反应会很迟钝 当网络环境真的变差时 更新的太慢)
  - Timeout interval: EstimatedRTT + "safety margin"
  - Estimate SampleRTT deviation from EstimatedRTT

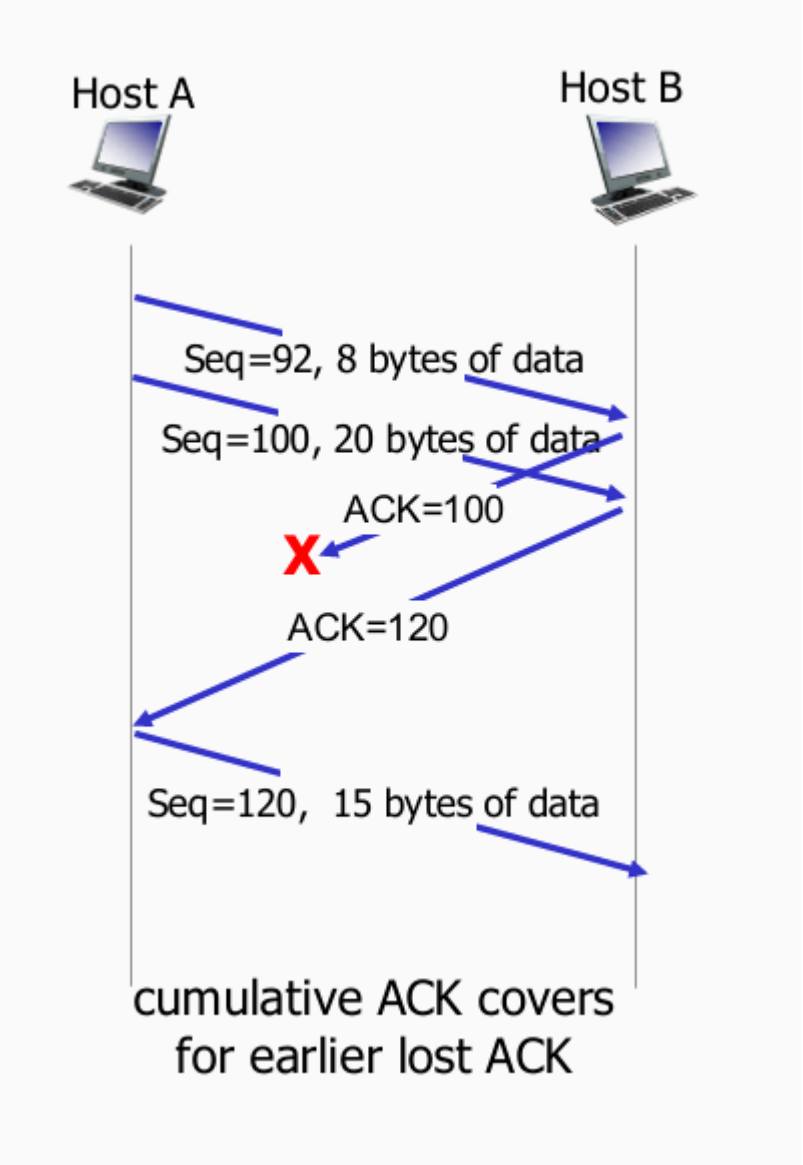
$$DevRTT = (1-\beta) * DevRTT + \beta * |SampleRTT - EstimatedRTT|$$

$$TimeoutInterval = EstimatedRTT + 4 * DevRTT$$

TCP reliable data transfer

- TCP creates rdt service on top of IP's unreliable service
  - pipelined segments
  - cumulative acks
  - single retransmission timer
    - retransmissions triggered by
      - timeout enents
      - duplicate acks
- TCP sender events:
  - data received from app:
    - create segment with seq#
    - Seq# is byte-stream number of first data byte in segment
    - Start timer if not already running
      - Think of timer as for oldest unacked segment
      - Expiration interbal: Timeout
  - Timeout:
    - Retransmit segment that caused timeout
    - Restart timer
  - Ack received
    - If ack acknowledges previously unacked segments
      - Update what is known to be ACKed
      - Start timer if there are still unacked segments
- retransmission scenarios

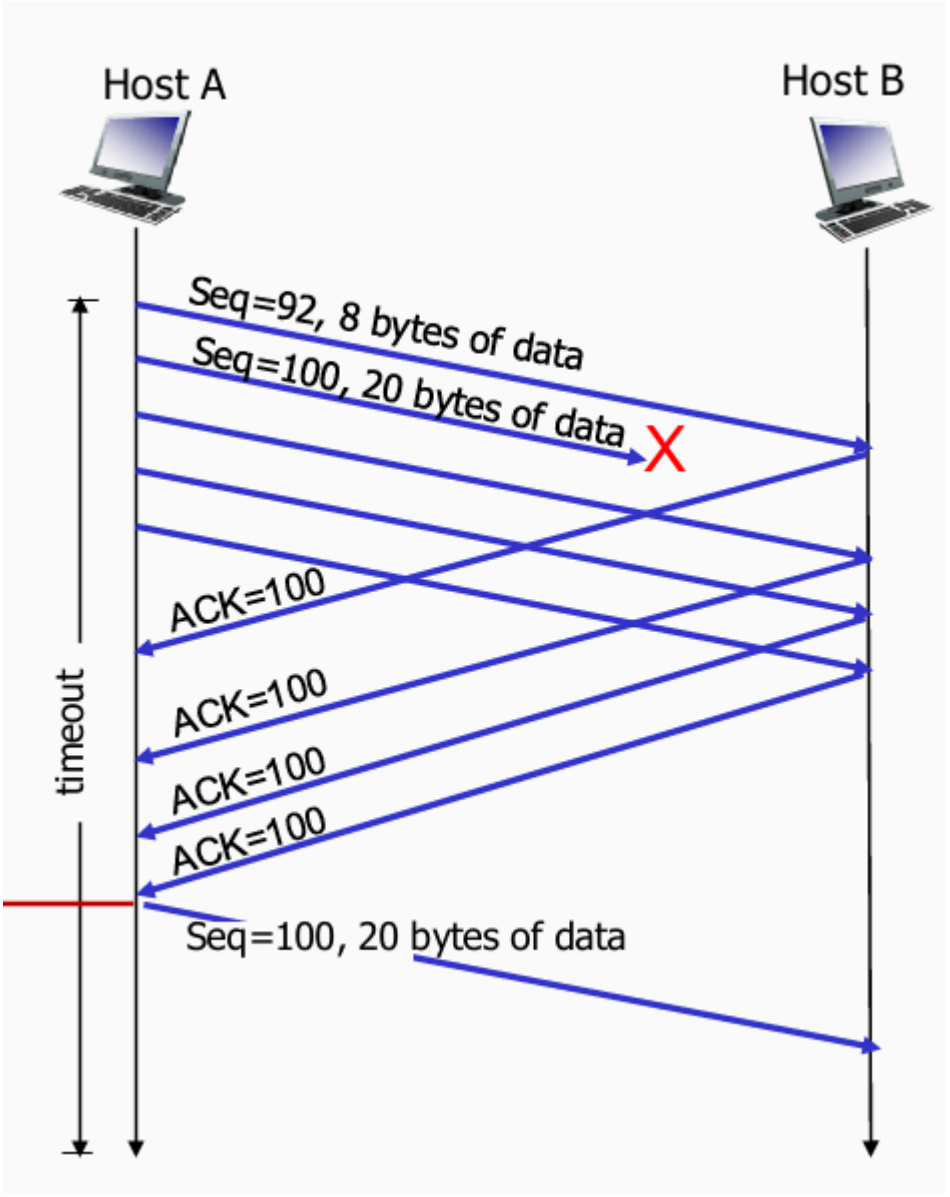




本质上对于两边都是黑盒 B请求120之后的说明前面的肯定收齐了 所以不需要重传

- TCP fast retransmit
    - if sender receives 3 additional ACKs for same data(triple duplicate ACKs), resend unACKed segment with smallest seq#

- likely that unACKed segment lost, so don't wait for timeout



TCP flow control (`rwnd`)

- TCP receiver "advertises" free buffer space in `rwnd` field in TCP header
  - `RcvBuffer` size set via socket options(typically default is 4096 bytes)
  - many operating systems auto-adjust `RcvBuffer`
- Sender limits amount of unACKed("in-flight") data to received `rwnd`
- guarantees receive buffer will not overflow

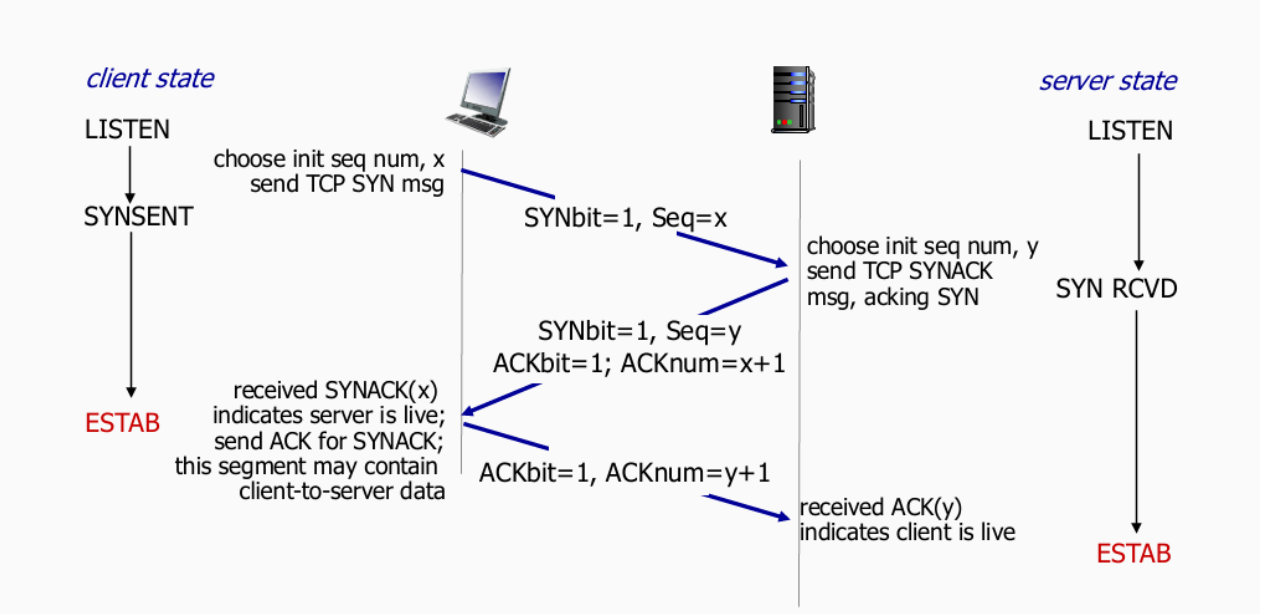
Connection management

Before exchanging data, sender / receiver "handshake":
Agree to establish connection(each knowing the other willing to establish connection) Agree on connection parameters

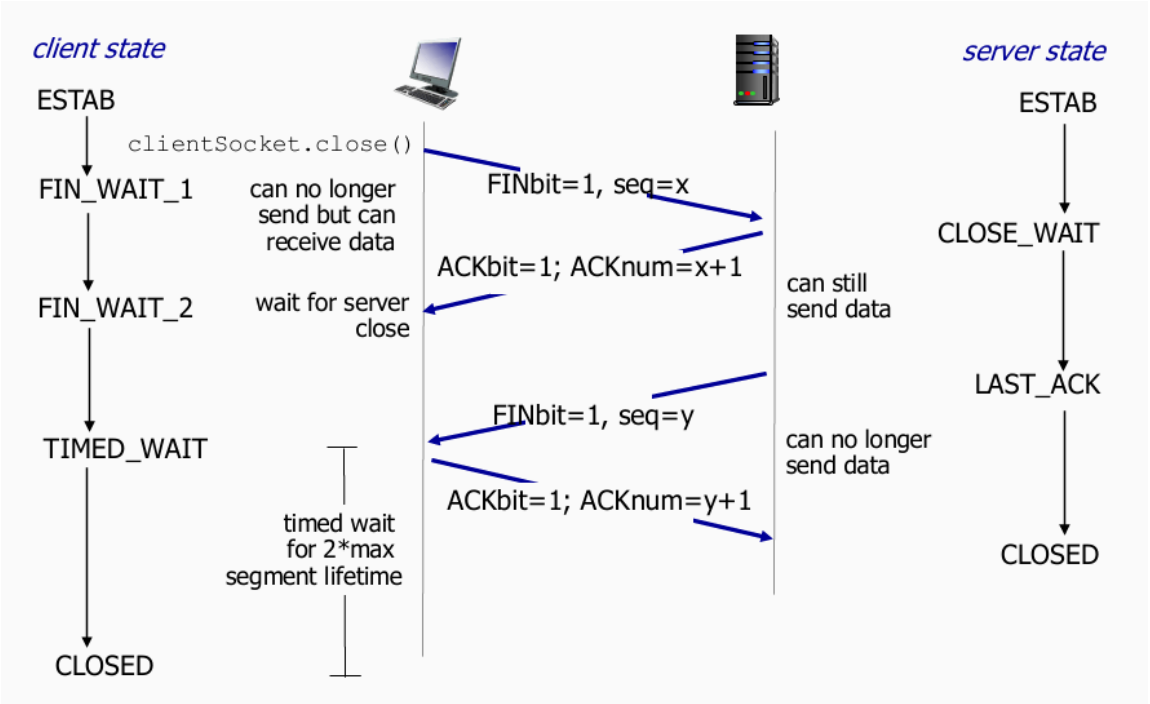
- 2-way handshake not always work network:
  - Variable delays

- Retransmitted messages(e.g. `req_conn(x)` due to message loss)
- Message reordering
- Can't see other side
- Half open connection / dup data accepted

- TCP 3-way handshake



- TCP: Closing a connection
  - Client, server each close their side of connection
    - send TCP segment with FIN bit =1
  - Respond to received FIN with ACK
    - On receiving FIN, ACK can be combined with own FIN
  - Simultaneous FIN exchanges can be handled



Lecture 6: Transport Layer: TCP congestion control

Congestion

- too many source sending too much data for network to handle(中间的网络堵住了)
- Manifestions:
  - long delays(queueing in router buffers)
  - packet loss(buffer overflow at routers)
- Different from flow control(one sender too fast for one receiver)
- congestion control's neck: network / routers | Flow control's neck: Receiver
- TCP sender遵循：

$$发送窗口 = min(rwnd, cwnd)$$

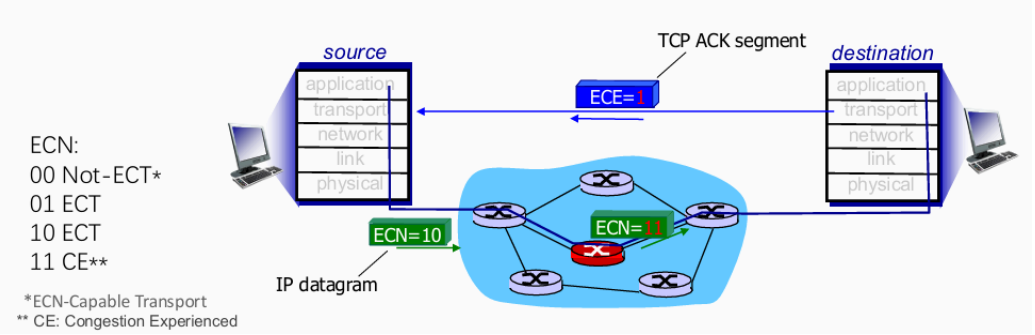
- Causes and costs
  - Load increases
    - many hosts send data at the same time, increasing the number of packet routers must process
  - Queues build up
    - If packets arrive faster than they can be forwarded, they start to queue in router buffers
  - Buffers overflow
    - when buffers are full, new packets are dropped
  - Retransmissions add load
    - Dropped packets trigger retransmissions, adding more traffic

$\rightarrow$ Higher delay, lower throughput, wasted banwidth

Approaches towards congestion control

- End-end congestion control
  - no explicit feedback from network
  - congestion inferred from obvious loss, delay
  - approach taken by TCP
- Network-assisted congestion control
  - routers provide direct feedback to sending / receiving hosts with flows passing through congested router
  - may indicate congestion level or explicitly set sending rate
  - TCP ECN
    - Two bits in IP header(ToS fields) marked by $\underline{network\ router}$ to indicate congestion
    - Congestion indication carried to receiving host

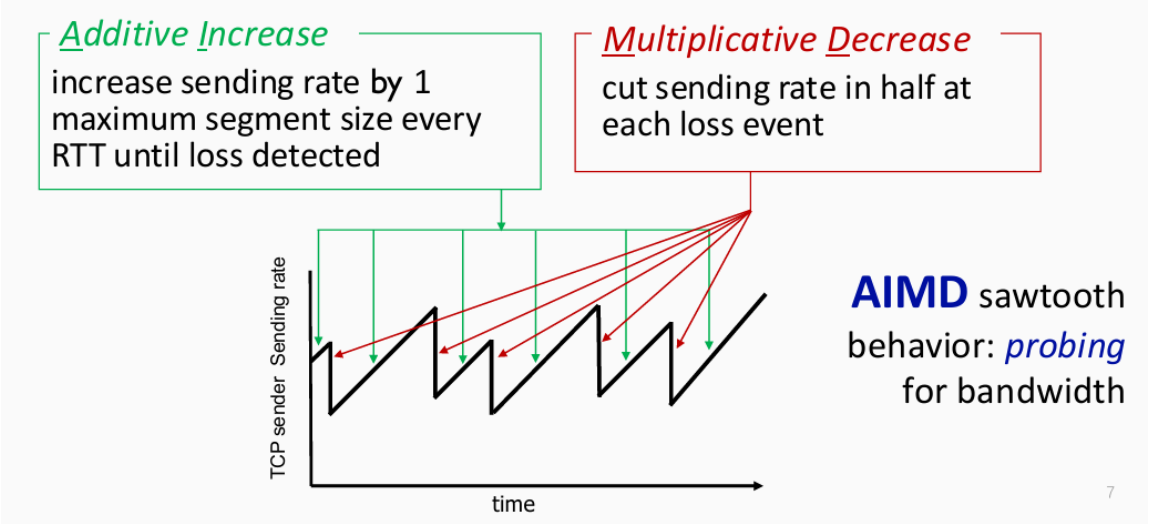
- Receiver(seeing congestion indication in IP datagram) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion



- ECN = 10 / 01 标识支持ECN机制（ECT，ECN- Capable Transport）
  ECN = 11 标识CE（Congestion Experienced，经历过堵塞）

TCP congestion control

- AIMD
  - A distributed, asynchronous algorithm
  - approach: senders can increase sending rate until packet loss(congestion) occurs, then decrease sending rate on loss event



    锯齿行为: 带宽探测
  - Multiplicative decrease detail: sending rate is
    - Cut in half on loss detected by triple duplicate ACK(TCP Reno)
    - Cut in 1 MSS (maximum segment size) when loss detected by timeout(TCP Tahoe)
  - Details
    - `cwnd` : congestion window
    - TCP sending behavior:
      - roughly: send `cwnd` bytes, wait RTT for ACKs, then send more bytes

$$TCP\ rate \approx \frac{cwnd}{RTT} bytes/sec$$

        - TCP sender limits transmission:

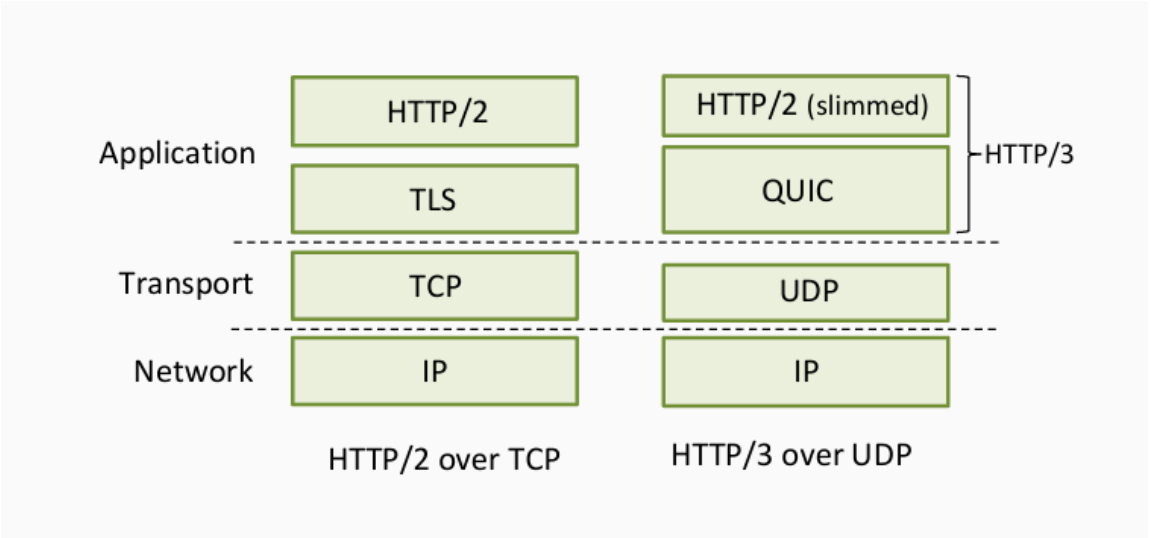
$$LastByteSent - LastByteAcked \leq cwnd$$

        - `cwnd` is dynamically adjusted in response to observed network congestion
- Slow start
    - when connection begins, increase rate exponentially until first loss event
        - initially `cwnd` = 1 MSS
        - double `cwnd` every RTT
        - done by incrementing `cwnd` for every ACK received
        - initial rate is slow, but ramps up exponentially fast
- TCP: from slow start to congestion avoidance
    - switch to linear from the exponential increase when `cwnd` gets to 1/2 of its value before timeout
    - Implementation:
        - variable `ssthresh`
        - on loss event, `ssthresh` is set to 1/2 of `cwnd` just before loss event
- TCP throughput
    - 对AIMD机制的数学总结，意味着TCP长期平均传输速度 / RTT时间
    - avg. TCP throughput as function of window size, RTT
        - ignore slow start, assume always data to send(TCP 发送窗口大小一直在0.5W-1W中线形波动 计算平均值)
    - W: window size(measured in bytes) where loss occurs
        - avg. window size(# in-flight bytes) is 3/4 W
        - avg. throughput is 3/4 per RTT

QUIC: Quick UDP Internet Connections

- application-layer protocol, on top of UDP
    - increase performance of HTTP

    - deployed on many Google servers, apps



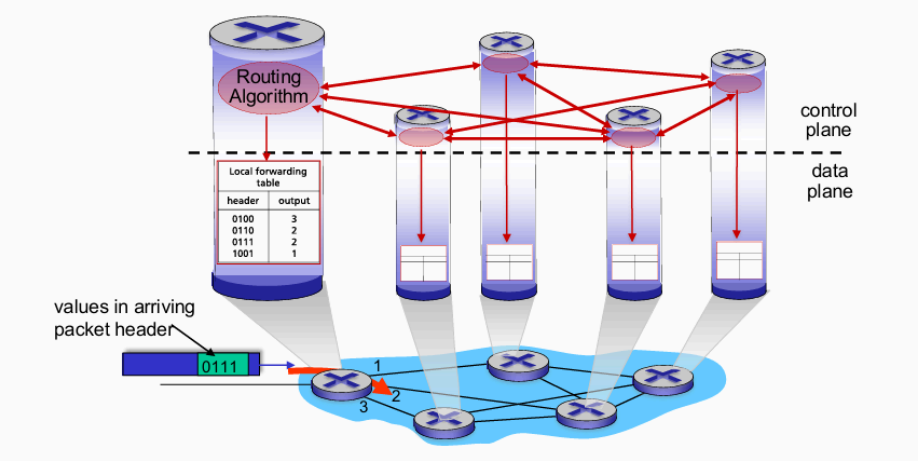
    - error and congestion control: similar with TCP ones
    - connection establishment: reliability, congestion control, authentication, encryption, state established in one(or even 0) RTT
    - Multiple application-level "streams" multiplexed over single QUIC connection 多个应用层流在单个QUIC连接上多路复用
        - separate reliable data transfer, security
        - common congestion control
- QUIC streams: parallellism, no HOL blocking

Lecture 6 Network Layer: Router / Internet protocol

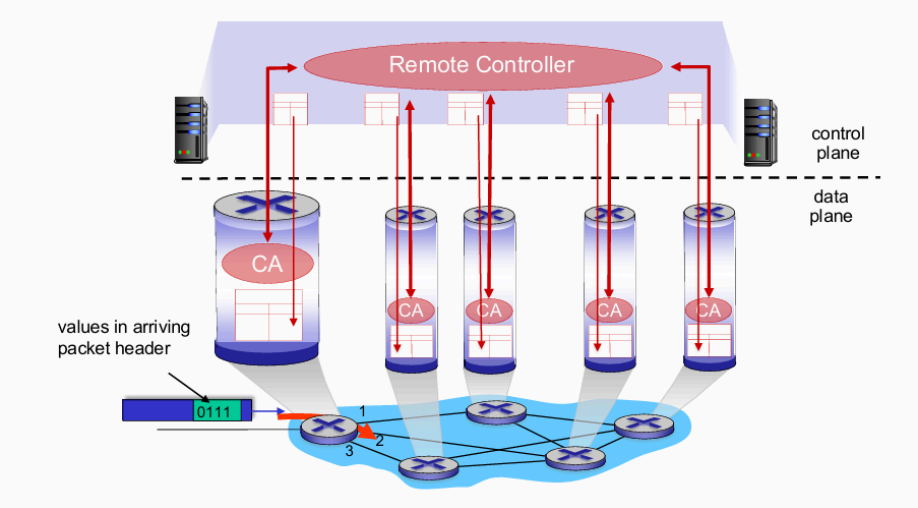
Overview of network layer

- Network layer services and protocols
    - Transport segment from sending to receiving host
        - sender: encapsulates segments into datagram, passes to link layer
        - receiver: delivers segment to transport layer protocol
    - Network layer protocols in every Internet device: hosts, routers
    - Routers
        - examines header fields in all IP datagrams passing through it
        - moves datagrams from input ports to output ports to transfer datagrams end-end path
- Two key network-layer functions
    - Forwarding: move packets from router's input to appropriate router output
    - Routing: determine route taken by packets from source to destination
        - routing algorithms
- Network layer: data plane, control plane
    - Data plane

    - local, per-router function
    - Determines how datagram arriving on router input port is forwarded to router output port
    - Forwarding function
  - Control plane
    - Network-wide logic
    - Determines how datagram is routed among routers along end-end path from source host to destination host
    - Two control-plane approaches
      - traditional routing algorithms: implemented in routers



      - Software Defined Networking(SDN): Implemented in (remote) servers

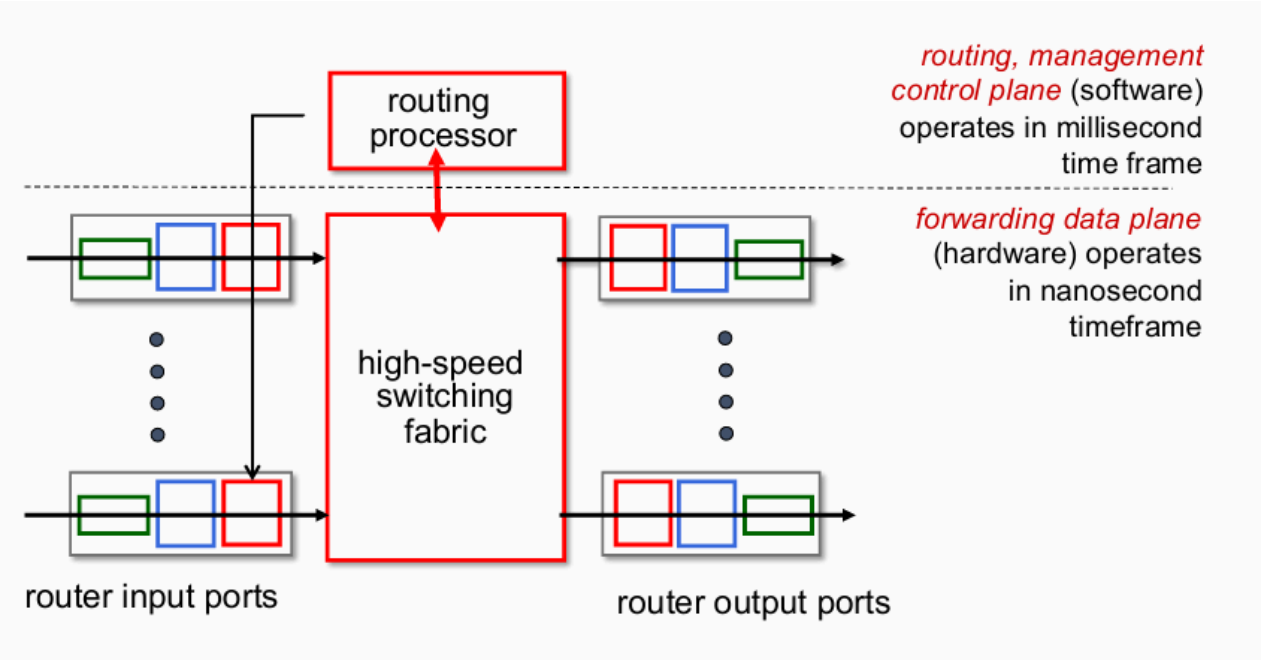


- Network-layer service model
  - Internet(best effort), No guarantees on
    - successful datagram delivery to destination
    - timing or order of delivery
    - bandwidth available to end-end flow

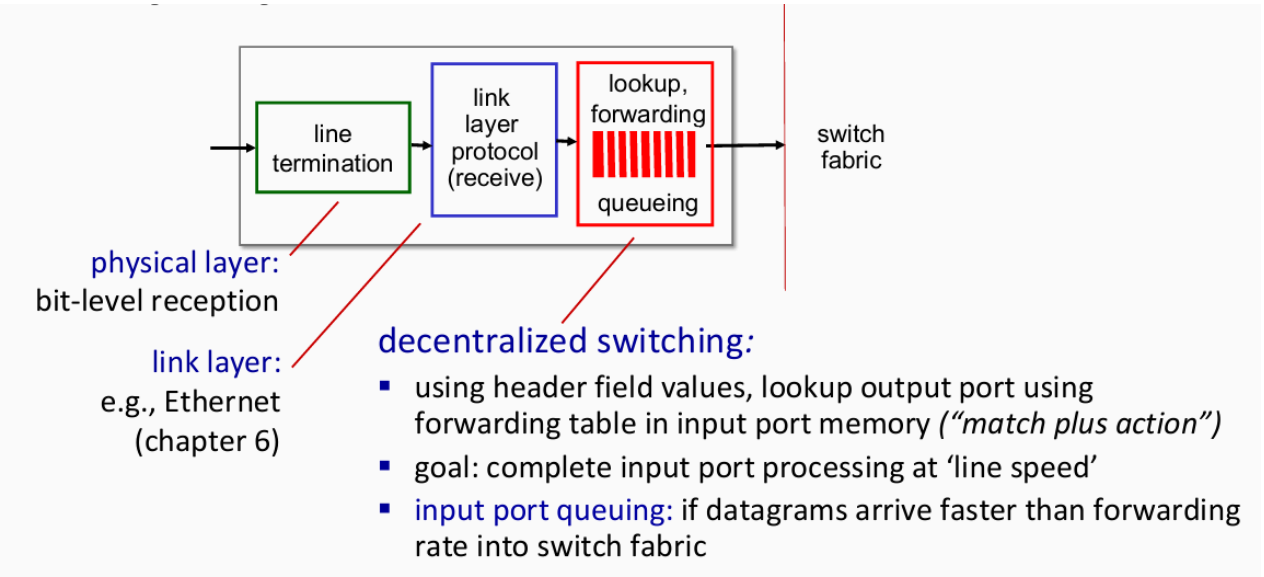
Router

router architecture overview

- High-level view of generic router architecture



- Input port functions



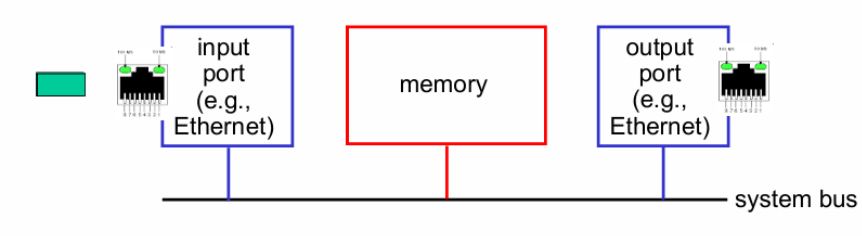
  - 分散式交换，通过输入端口内存中的forwarding table查找输出端口 (match + action)
  - destination-based forwarding: forward based only on destination IP address(traditional)

forwarding table

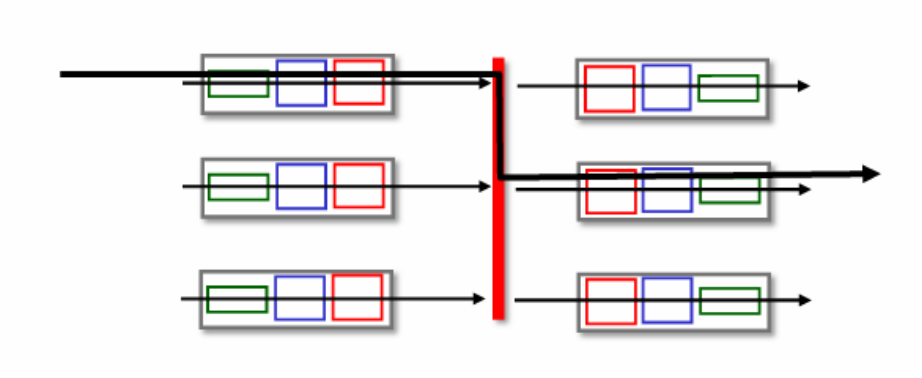
	Destination Address Range	Link Interface
200.23.16.0 through 200.23.23.255	11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
200.23.24.0 through 200.23.24.255	11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
200.23.25.0 through 200.23.31.255	11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
	otherwise	3

- generalized forwarding: forward based on any set of header field values
  通用转发：基于任何一组头部字段值转发
- Longest prefix matching
  - when looking for forwarding table entry for given destination address, use longest address prefix that matches destination address
  - Longest prefix matching: often performed using Ternary Content Addressable Memories(TCAMs) 三元内容可寻址储存器
    - content addressable: present address to TCAM: retrieve address in one clock cycle, regardless of table size

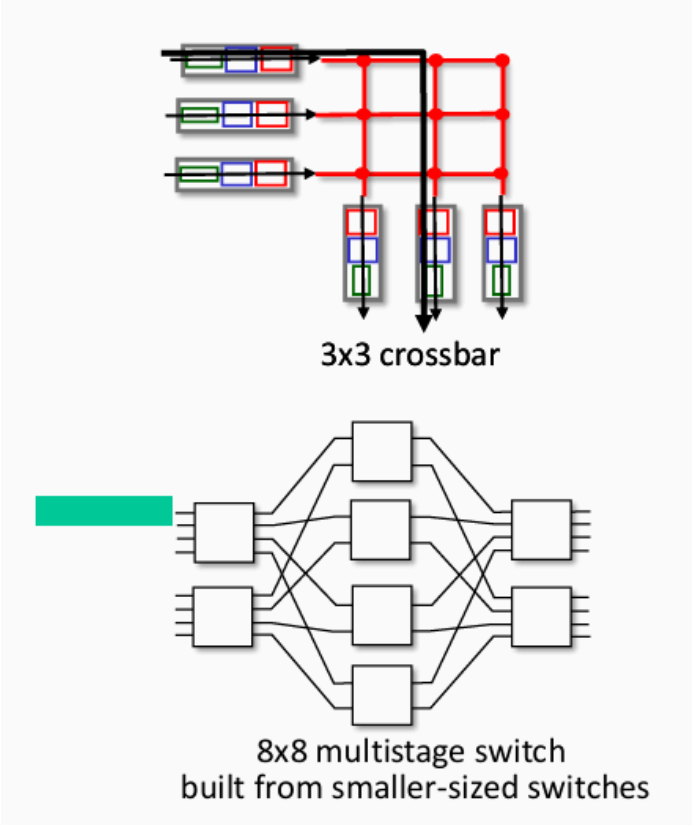
- Switching fabrics 交换结构
  - transfer packet from input link to appropriate output link
  - switching rate: rate at which packets can be transfer from inputs to outputs
    - often measured as multiple of input / output line rate
    - N inputs: switching rate N times line rate desirable
  - Three major types of switching fabircs
    - Switching via memory
      - first generation routers
      - traditional computers with switching under direct control of CPU
      - packet copied to system's memory
      - speed limited by memory bandwidth(2 bus crossings per datagram)



- Switching via a bus
  - datagram from input port memory to output port memory via a shard bus
  - bus contention: switching speed limited by bus bandwidth
  - 32 Gbps, Cisco 5600: sufficient speed for access routers

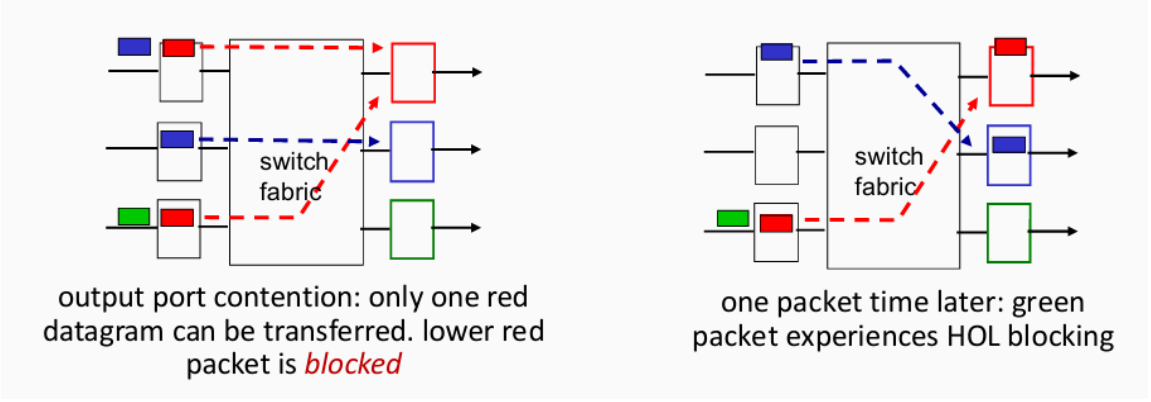


- Switching via interconnection network
  - Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
    - multistage switch: n × n switch from multiple stages of smaller switches
      多级交换机：由多级较小交换机构成的交换机
    - exploiting parallellism 利用并行性
      - fragment datagram into fixed length cells on entry
      - switch cells through the fabric, reassemble datagram at exit
      - 进入时将datagram分成固定长度的cells 然后在出口处组装数据包



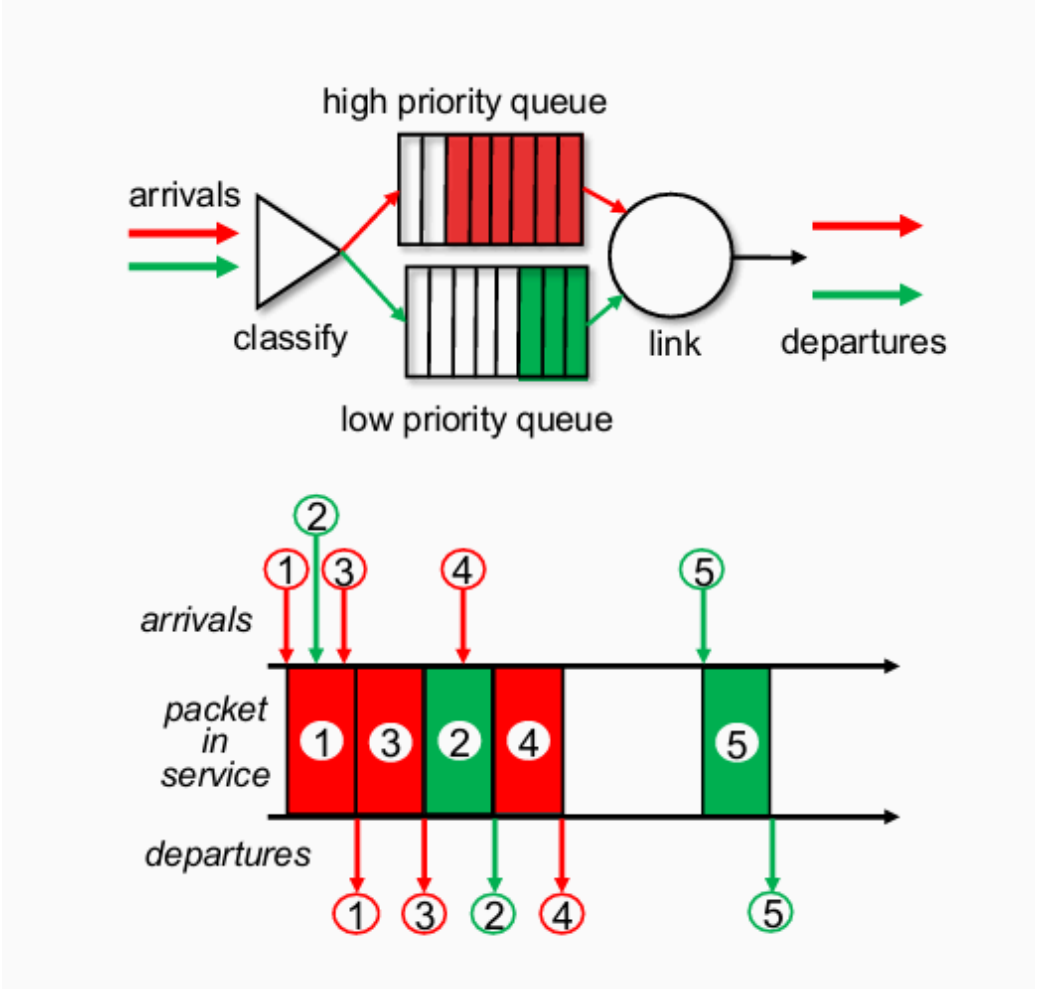
- Scaling, using multiple switching "planes" in parallel
  - speed up, scale up via parallellism

- Input port queuing
  - If switch fabric slower than input ports combined → queueing may occur at input queues
    - queueing delay and loss due to input buffer overflow
  - Head of Line(HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



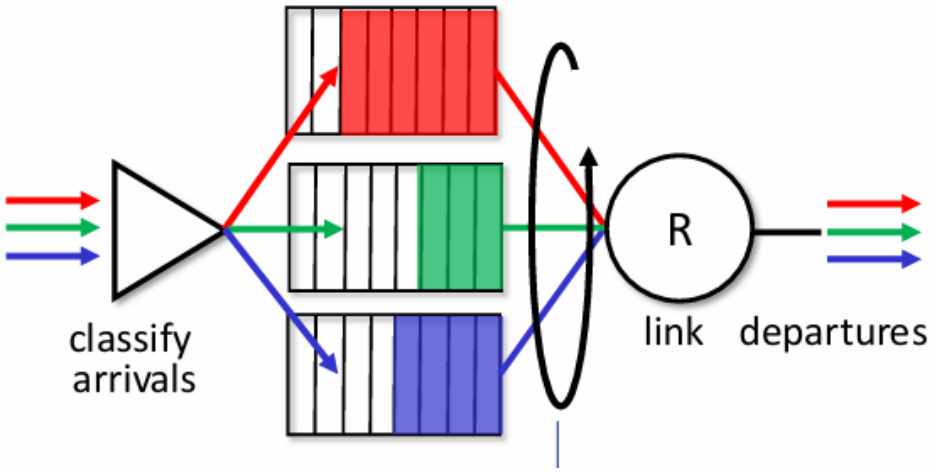
- Output port queuing
  - Buffering required when datagrams arrive from fabric faster than link transmission rate
    - Drop policy: which datagrams to drop if no free buffers
    - Scheduling discipline chooses among queued datagrams for transmission

        - Priority scheduling who gets best performance, network neutrality
    - Datagrams can be lost due to congestion, lack of buffers
- Buffer Management
    - drop: which packet to add, drop when buffers are full
        - tail drop: drop arriving packet
        - priority: drop / remove on priority basis
    - marking: which packets to mark to signal congestion(ECN, RED)
- Packet Scheduling policies
  Packet scheduling: deciding which packet to send next on link
    - FCFS
        - packets transmitted in order of arrival to output port
        - also called FIFO
    - Priority scheduling
        - arriving traffic classified, queued by class(any header fields can be used for classification)
        - send packet from the highest priority queue that has buffered packets
        - FCFS within priority class 同一优先级的类采取FCFS



    - Round Robin(RR) scheduling

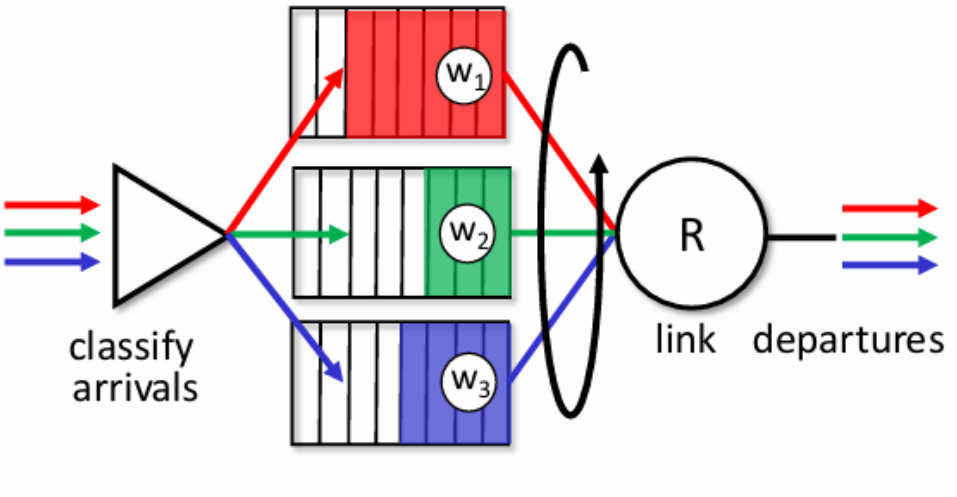
- arriving traffic classified, queued by class(any header fields can be used for classification)
- Server cyclically, repeatedly scans class queues, sending one complete packet from each class(if available) in turn



- Weighted fair queueing (WFQ)
  - generalized Round Robin
  - each class, i, has weight w_i, and gets weighted amount of service in each cycle:

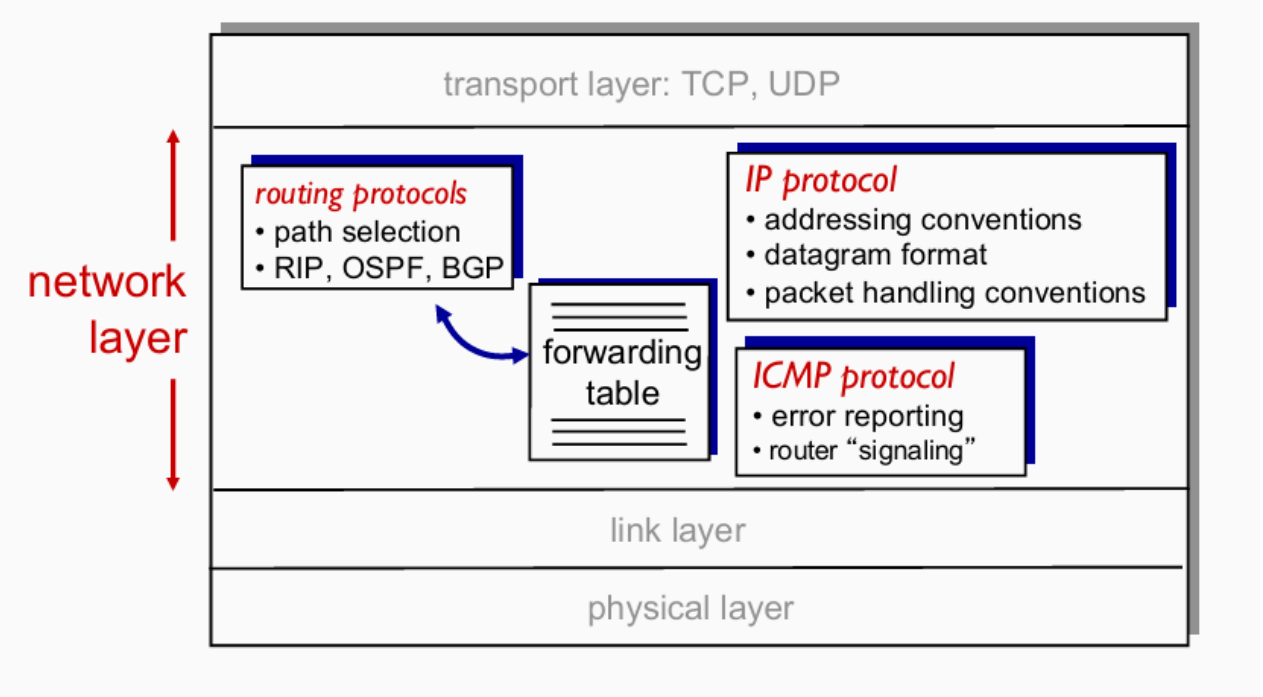
$$\frac{W_i}{\Sigma_j W_j}$$

  - minimum bandwidth guarantee(per-traffic-class)

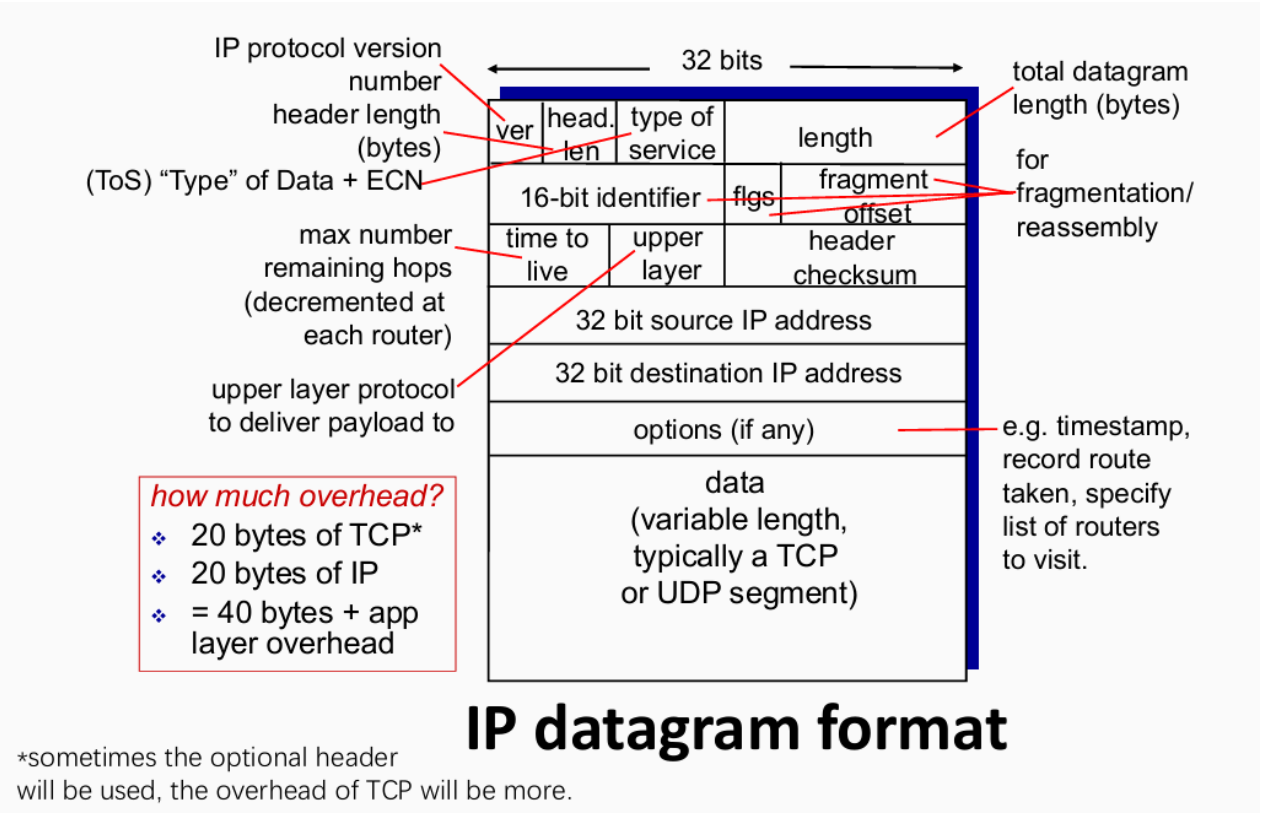


Internet Protocol

- Host, router network layer functions:

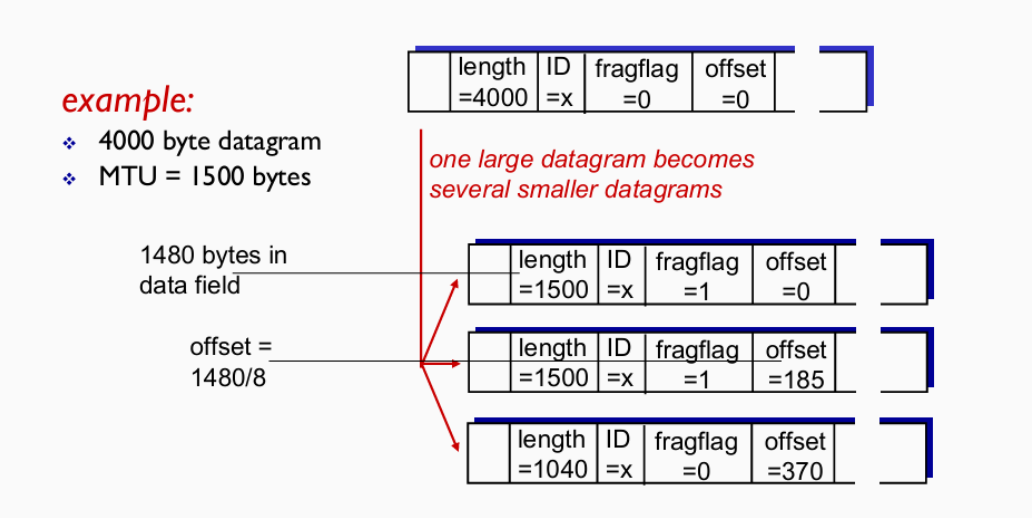


- IP datagram format



  - Header checksum 只校验头部 不校验数据部分（数据部分由TCP / UDP）自主校验

- IP fragmentation, reassembly
  - Network links have MTU(max.transfer size, largest possible link layer frame)
    - Different link types, different MTUs
  - Large IP datagram divided("fragmented") within net
    - One datagram becomes several datagrams
    - Reassembled only at final destination

- IP header bits used to identify, order related fragments
- 

  - TCP header 20bytes / IP header 20 bytes in normal
  - Offset 字段的单位是8bytes 所以用1480 / 8
  - flag = 1 标识后面还有段

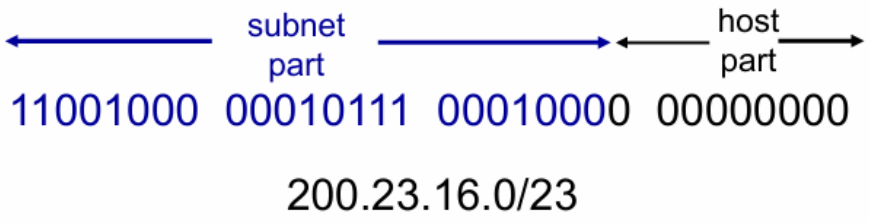
Lecture 7 The network Layer: Routing Algorithm

The Internet Protocol

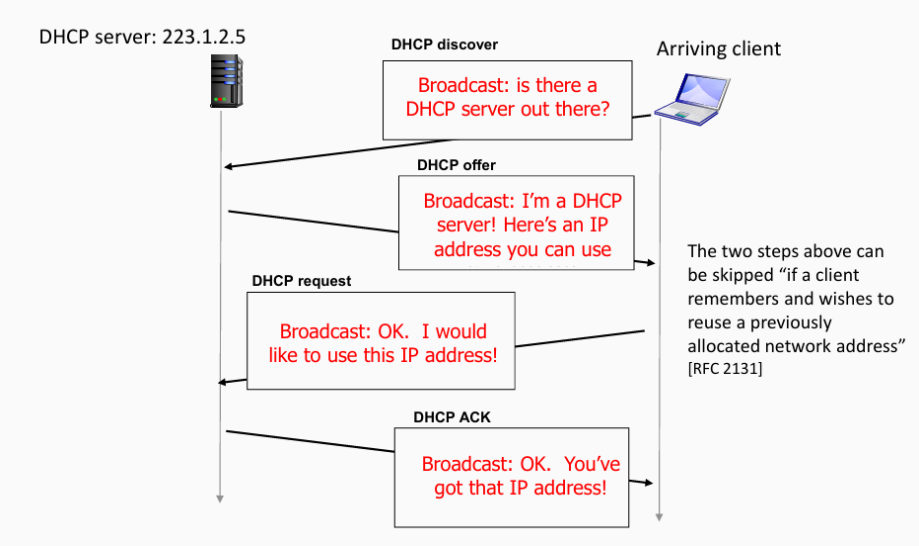
IP Addressing

- Interface: connection between host / router and physical link
  - routers typically have multiple interfaces
  - host typically has one or two interfaces(wired ethernet / wireless 802.11(WiFi))
  - wired Ethernet interfaces connected by Ethernet switches
  - wireless WiFi interfaces connected by WiFi base stations
- IP address: 32 bit identifier associated with each host or router interface
- Subnets
  - device interfaces that can physically reach each other without passing through an intervening router(无需经过中间路由器)
  - IP addresses have 2 parts
    - subnet part: devices in same subnet have common high order bits
    - host part: remaining low order bits
  - Recipe for defining subnets
    - detach each interface from its host or router, creating "islands" of isolated networks
    - each isolated network is called a subnet
    - 同一个子网内，设备可以通过交换机直接对话，不同子网，必须通过路由器才能通信

- Network classes
  - Classful addressing 分级寻址
    - The network portion of an IP address were constrained to be 8, 16, or 24 bits in length
    - Subnets with 8, 16, 24-bit subnet addresses were known as class A, B and C networks
    - A class B(/16) subnet supporting $2^{16} - 2 = 65,534$ hosts, which is too large
  - CIDR (Classless InterDomain Routing)
    - subnet portion of address can have arbitrary length
    - address format: a.b.c.e/x, where x is # bits in subnet portion of address



  - A host get IP address by 2 methods
    - hard-code by sysadmin in config file
    - DHCP(Dynamic Host Configuration Protocol)
      - Allow host to dynamically obtain its IP address from network server when it joins network
        - can renew its lease on address in use
        - allows reuse of addresses(only hold address while connected / on)
        - support for mobile users who join / leave network
      - DHCP Client-Server Scenario



        - 如果客户端记得并希望重用先前分配的网络地址，最开始的两部可以跳过
    - More than IP address
        - can return more than just allocated IP address on subnet:
            - address of first-hop router for client
            - name and IP address of DNS server
            - network mask(indicating network versus host portion of address)
    - Example
        - Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server
        - DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
        - Ethernet frame broadcast on LAN, received at router running DHCP server
        - Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP
- Network get subnet part of IP address by getting allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20

ISP can then allocate out its address space in, say, 8 blocks:

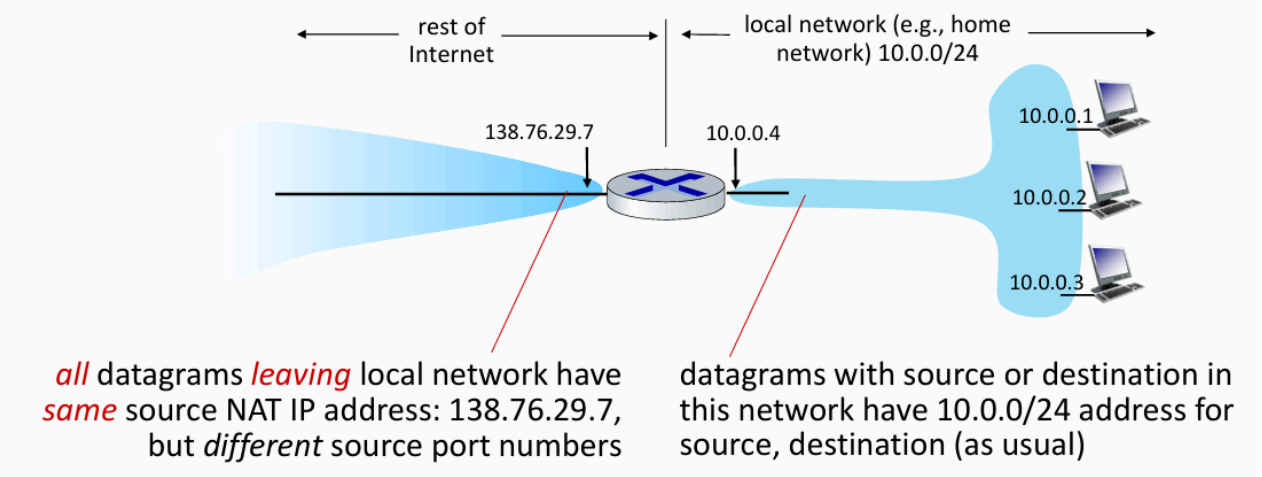
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

- Hierarchical addressing: route aggregation 路由聚合
    - Hierarchical addressing allows efficient advertisement of routing information
    - The ability to use a single prefix to advertise multiple networks is often referred to as address aggregation or route aggregation
    - more specific routes(通过 longest prefix match 实现)
- The method of an ISP get block of addresses
    - ICANN: Internet Cooperation for Assigned Names and Numbers
        - allocates IP addresses through 5 regional registers(RRs) (who may then allocate to local registers)

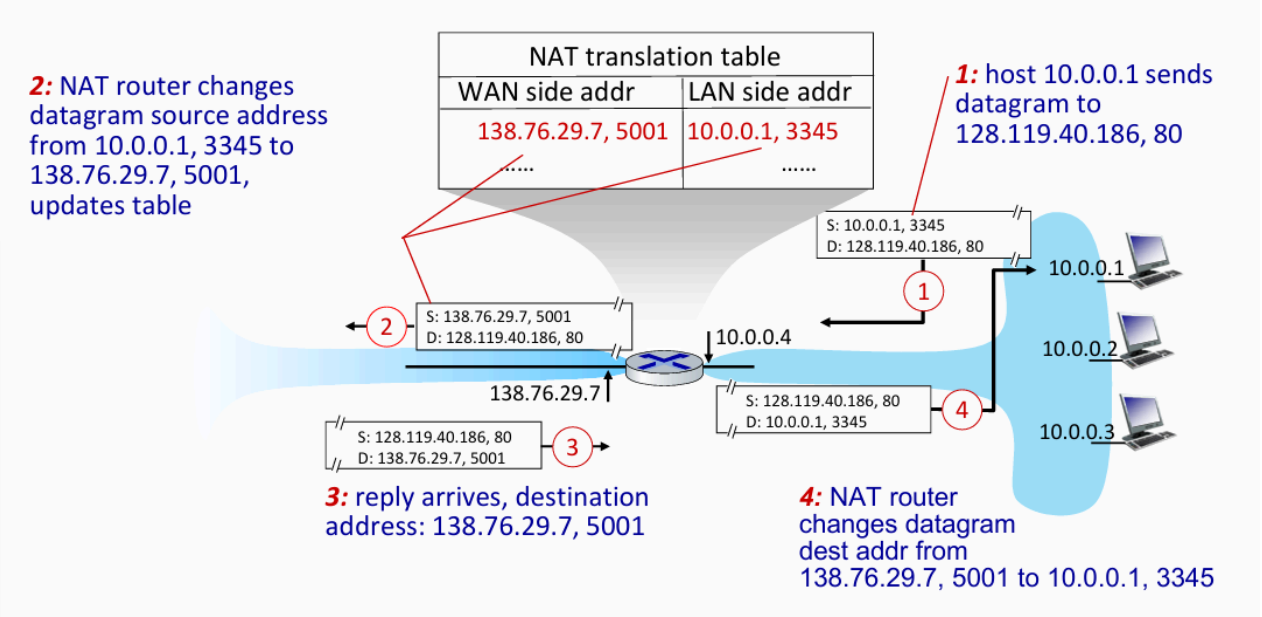
- manages DNS root zone, including delegation of individual TLD(.com, .edu)

NAT

- NAT(Network Address Translation): all devices in the local network share just one IPv4 address as far as the outside is concerned



- All devices in local network have 32-bit addresses in a "private" IP address space(10/8, 172.16/12, 192.168/16 prefixes that can only be used in local network)
- Advantages
  - only one IP address needed from provider ISP for all devices
  - can change addresses of host in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - security: devices inside local network not directly addressable, visible by outside world
- Implementation: NAT router must(transparently)

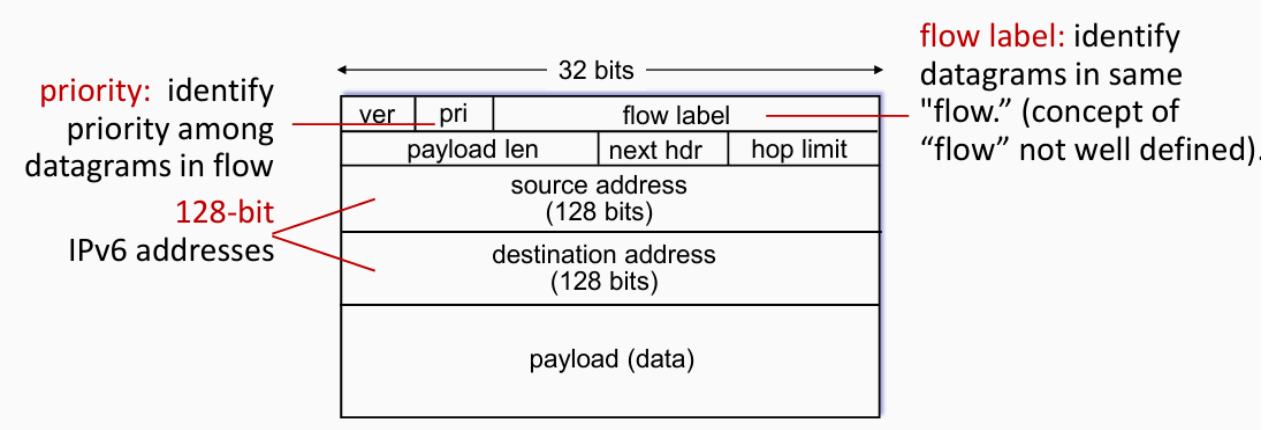


  - outgoing datagrams: replace(source IP address, port#) of every outgoing datagram to (NAT IP address, new port#)

        - remote clients / servers will respond using (NAT IP address, port #) to (NAT IP address, new port#)translation pair
    - remember(in NAT translation table) every(source IP address, port #) to (NAT IP address, new port#)translation pair
    - incoming datagrams: replace(NAT IP address, new port #)in destination fields of evert incoming datagram with corresponding(source IP address, port#)stored in NAT table
- Controversial:
    - routers should only process up to layer 3
    - address "shortage" should be solved by IPv6
    - violates end-to-end argument(port # manipulation by network-layer device)
    - NAT traversal: what if client wants to connect to server behind NAT

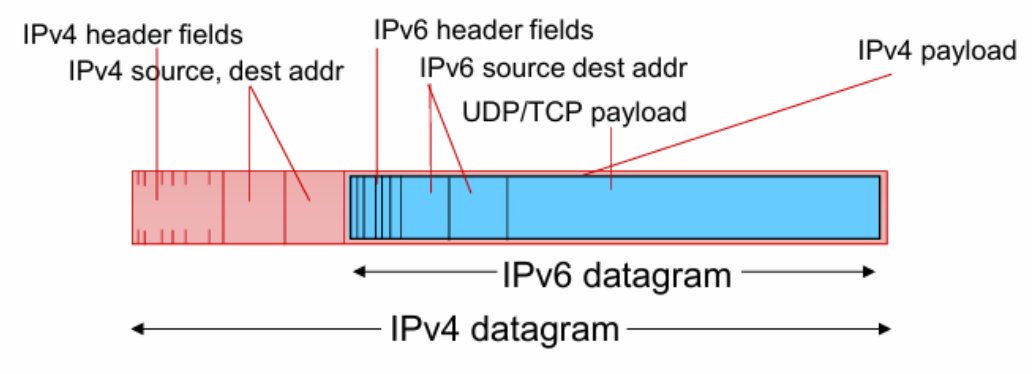
IPv6

- initial motivation: 32-bit IPv4 address space would be completely allocated
- Additional motivation
    - Header format helps speed processing / forwarding
    - Header changes to facilitate QoS
- IPv6 datagram format



    - Fixed-length 40-byte header
    - No fragmentation allowed(at intermediate routers, see "packet too big"new ICMP message type)
    - No checksum
    - No fragmentation / reassembly / flag
    - No options(available as upper-layer, next-header protocol at router)
- Transition from IPv4 to IPv6
    - Tunneling: IPv6 datagram carried as payload in IPv4 datagram among IPv6 routers

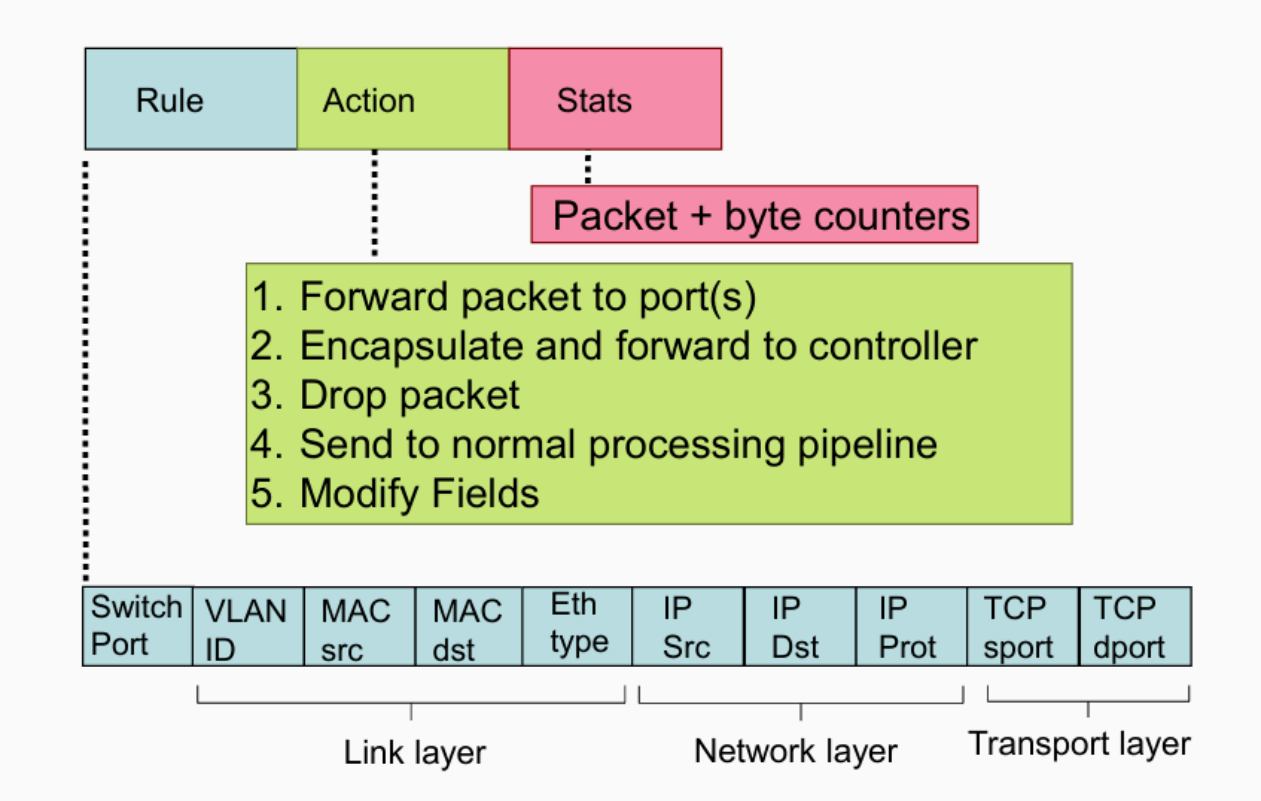
- tunneling used extensively in other contexts(4G / 5G)



Generalized Forwarding and SDN

Flow Table

- flow: defined by header field values(in link, network, transport-layer fields)
- generalized forwarding: simple packet-handling rules
  - match: patterns values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: # bytes and # packets
- OpenFlow: Flow Table Entries



- match + action: abstraction unifies different kinds of devices
  - router
    - match: Longest destination IP prefix

        - action: forward out a link
    - Switch
        - match: destination MAC address
        - action: forward or flood
    - Firewall
        - match: IP addresses and TCP/UDP port number
        - action: permit or deny
    - NAT
        - match: IP address and port
        - action: rewrite address and port

Routing Protocols

Goal: determine "good" paths (equivalently routes), from sending host to receiving host through network of routers
Path: sequence of routers packets will traverse in going from given initial source host to given final destination host

- Routing Algorithm Classification
    - global: all routers have complete topology, link cost info
        - link state algorithms
    - decentralized: iterative process of computation, exchange of info with neighbors
        - routers initially only know link costs to attached neighbors
        - distance vector algorithms
    - dynamic: routes change more quickly
        - periodic updates or in response to link cost changes
    - static: routes change slowly over time
- A link-state Routing Algorithm - Dijkstra
    - Net topology and link costs known to all nodes
        - Accomplished via "link state broadcast"
        - All nodes have same info
    - Computes least cost paths from one node("source") to all other nodes
        - Gives forwarding table for that node

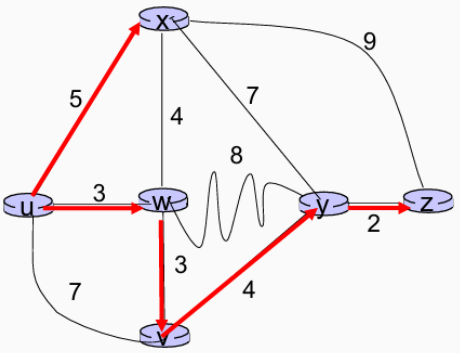
- Iterative: after k iterations, know least cost path to k dest. 's

computing routing table in u:

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



- Distance Vector Algorithm
  - Bellman-Ford equation(dynamic programming)
    let

$$d_x(y) = cost\ of\ least-cost\ path\ from\ x\ to\ y$$

    then

$$d_x(y) = min_v\{c(x,v) + d_v(y)\}$$

    $c(x,v)$ is cost to neighbor v
    $d_v(y)$ is cost from neighbor v to destination y
  - Key-idea:
    - From time to time, each node sends its own distance vector estimate to neighbors
    - when x receives new DV estimate from neighbor, it updates its own DV using B-F equation
  - Iterative, asynchronous
    - each local iteration caused by
      - local link cost change
      - DV update message from neighbor
  - Distributed
    - Each node notifies neighbors only when its DV changes
      - Its neighbors then further notify their neighbors if necessary

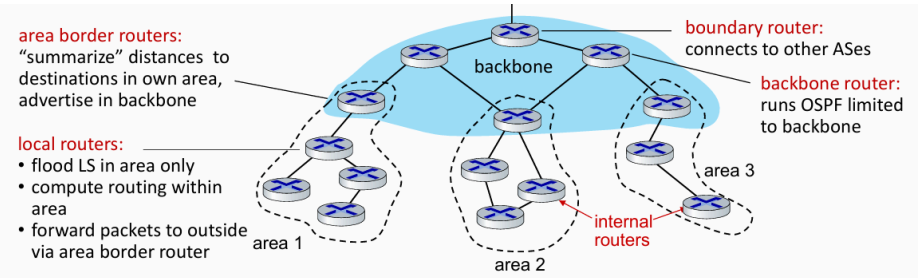
Lecture 8 The Network Layer: Control Plane

- Internet Approach to Scalable Routing
  - Aggregate routers into regions known as "autonomous systems"(AS)
  - Intra-AS(aka "intra-domain")
    - routing among routers within some AS("network")

        - all routers in AS must run same intra-domain protocol
        - routers in different AS can run different intra-domain routing protocols
        - gateway router: at "edge" of its own AS, has link to router in other AS'es
    - Inter-AS
        - gateways perform inter-domain routing
    - Forwarding table configured by both intra- and inter- AS routing algorithm
        - Intra-AS routing determines entries for destinations within AS
          AS 内部路由确定AS内部目的地的条目
        - Inter-AS & Intra-AS determine entries for external destinations
          AS间和AS内部路由去确定外部目的地的条目

Intra-ISP routing: OSPF

- Open Shortest Path First Routing
    - Classic link-state
        - each router floods OSPF link-state advertisements (directly over IP rather than using TCP/UDP)
        - multiple link costs metrics possible: bandwidth, delay
        - each router has full topology, use Dijkstra's algorithm to compute the forwarding table
    - security: all OSPF messages authenticated(to prevent malicious intrusion)
    - Hierarchical OSPF
        - Two-level hierarchy: local area, backbone
            - link-state advertisements flooded only in area, or backbone
            - each node has detailed area topology; only knows direction to reach other destinations



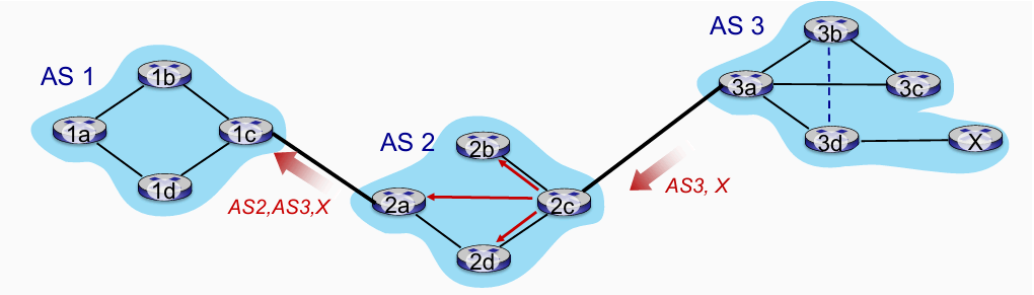
Routing among ISPs: BGP

- BGP (Border Gateway Protocol): the de facto inter-domain routing protocol
- allows subnet to advertise its existence, and the destinations it can reach, to rest of Internet "I'm here, here is who I can reach and how"
- BGP provides each AS router a means to
    - obtain destination network reachability info from neighboring ASes(eBGP)

从相邻AS获取目标网络可达性信息(eBGP)
- determine routes to other networks based on reachability information and policy
更具可达性信息和策略确定到其他网络的路由
- propagate reachability information to all AS-internal routers(iBGP)
- advertise(to neighboring networks) destination reachability info

- BGP basics
  - BGP session: two BGP routers("peers, speakers") exchange BGP messages over semi-permanent TCP connection:
    - advertising paths to different destination network prefixes(e.g, to a destination /16 network)
    - BGP is a "path vector" protocol
  - When AS3 gateway 3a advertises path AS3,X to AS2 gateway 2c:
    - AS3 promises to AS2 it will forward datagrams towards X
- BGP protocol message
  - exchanged between peers over TCP connection
  - BGP messages:
    - OPEN: opens TCP connection to remotes BGP peer and authenticates sending BGP peer
    - UPDATE: advertises new path(or withdraws old)
    - KEEPALIVE: keeps connection alive in absence of UPDATEs; also ACKs OPEN request
    - NOTIFICATION: reports errors in previous; also used to close connection
- Path attributes and BGP routes
  - BGP advertised path: prefix + attributes = "route"
    - Path prefix: destination being advertised
    - two important attributes
      - AS-PATH: list of ASes through which prefix advertisement has passed
      - NEXT-HOP: indicates specific internal-AS router to next-hop AS
  - policy-based routing:
    - router receiving route advertisement to destination X uses policy to accept/reject a path(never route through AS W, or country Y)
    - router uses policy to decide whether to advertise a path to neighboring AS Z (does router want to route traffic forwarded from Z destined to X)

- example



  - AS2 router 2c receives path advertisement AS3,X(via eBGP) from AS3 router 3a
  - Based on AS2 policy, AS2 router 2c accepts path AS3,X propagates(via iBGP) to all AS2 routers
  - Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

- BGP: Achieving Policy via Advertisement
  - ISP only wants to route traffic to/from its customer networks(does not want to carry transmit traffic between other ISPs-a typical real world policy)
- Hot Potato routing: choose local gateway that has least intra-domain cost, don't worry about inter-domain cost

The difference of Intra-, Inter-AS Routing

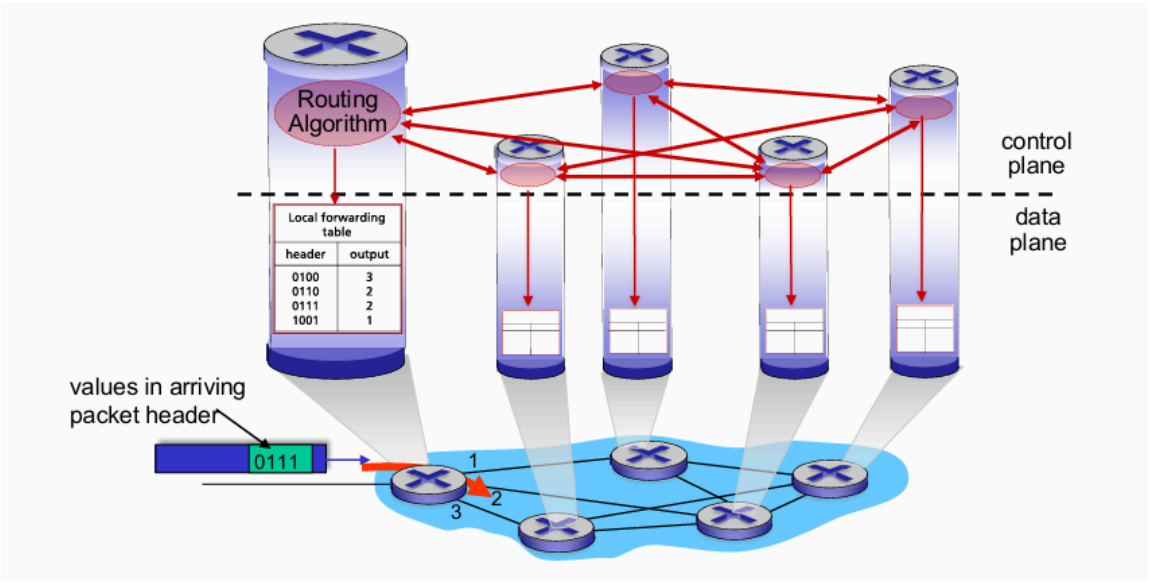
- policy:
  - inter-AS: admin wats control over how its traffic is routed, who routes through its network
  - intra-AS: single admin, so policy less of an issue
- scale: reducing forwarding table size, routing update traffic
  - hierarchical routing: limiting the scope of full topological information
  - BGP routing to CIDRized destination networks(summarized routes)
- performance:
  - intra-AS: can focus on performance
  - inter-AS: policy dominates over performance

SDN control plane

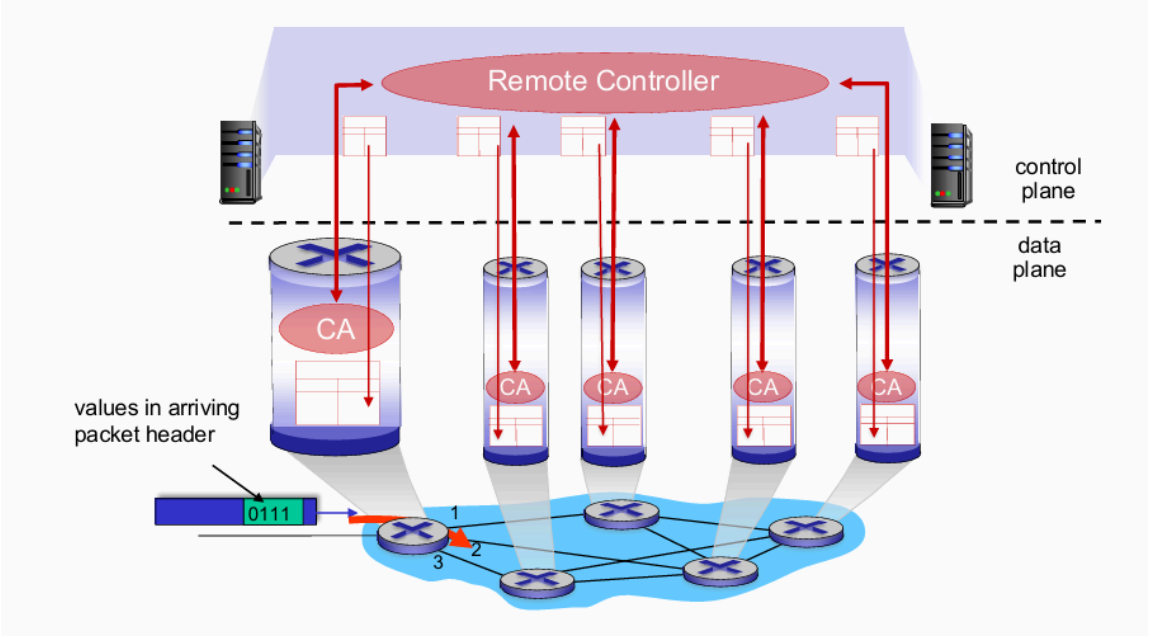
上文中都讲述传统的单路由器视角：data plane负责当一个包从端口A进来时，查看转发表，发现目的地应该去端口B，于是通过路由器内部的交换结构（switching fabric）把包扔到端口B，核心动作是转发forwarding；Control plane则负责思考和决策，它运行OSPF / BGP等协议，和邻居交换信息，计算出“去往目的地X的最短路径是走端口B”，然后把这个结果写进Data plane的转发表中。
SDN时代: data plane 继续留在路由器中，现在路由器的内部不在运行复杂的路由协议OSPF，只保留data plane然后根据Flow Table干活；Control Plane被抽离出来，被搬到云端

或一台专门的服务器上，叫做SDN控制器（controller），它统一计算全网的路径，然后通过OpenFlow等协议，远程告诉下面所有的路由器该怎么转发

- Per-router Control Plane
  - Individual routing algorithm components in each and every router interact with each other in control plane to compute forwarding tables(traditional)

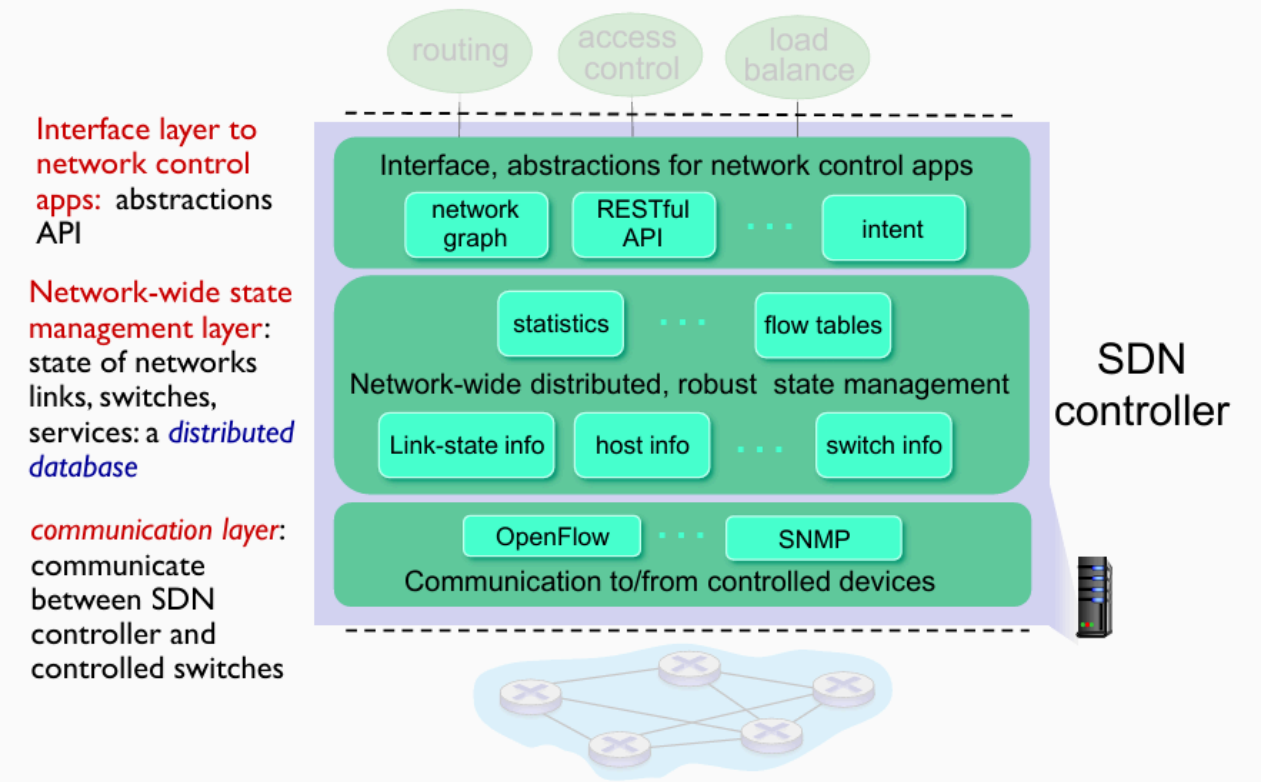


- Logically Centralized Control Plane
  - A distinct(typically remote) controller interacts with local control agents(CAs) in routers to compute forwarding tables
    独立的控制器与路由器中的本地控制代理交互以计算转发表



  - easier network management: avoid router misconfigurations, greater flexibility of traffic flows
  - Table-based forwarding(recall OpenFlow API) allows "programming" routers
    - Centralized "programming" easier: compute tables centrally and distribute

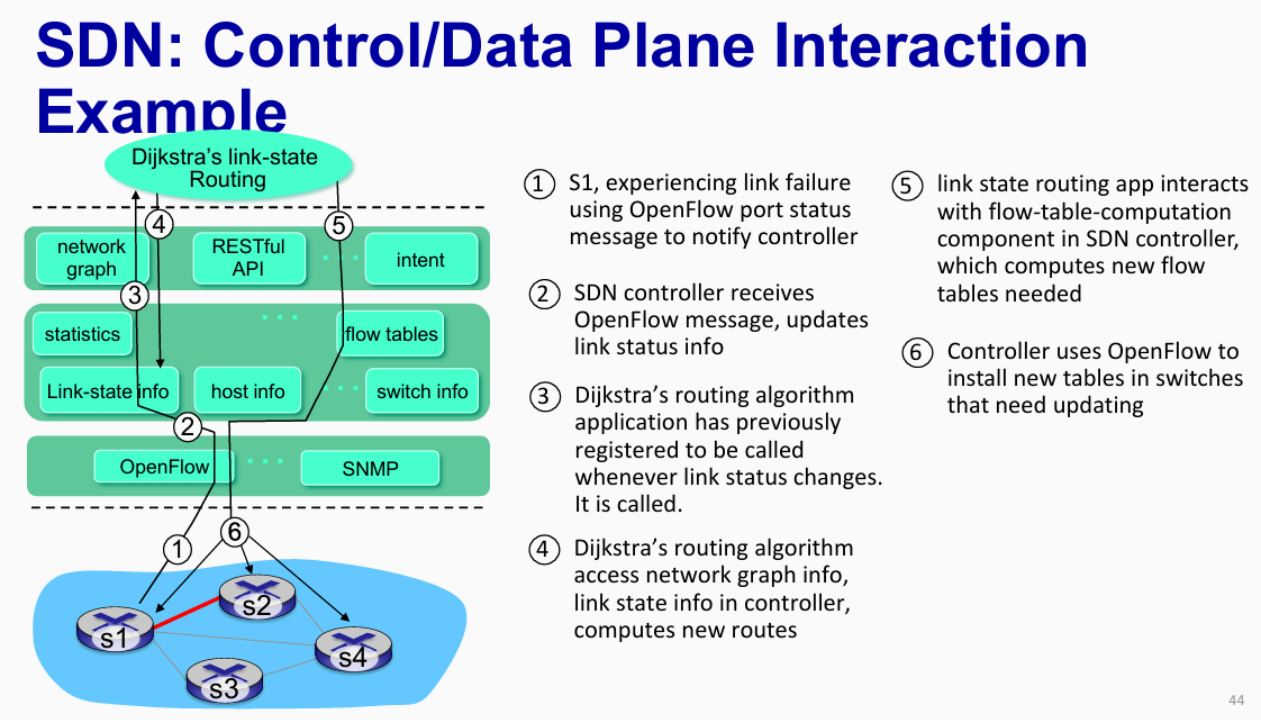
- Distribute "programming more difficult: compute tables as result of distributed algorithm(protocol) implemented in each and every router



- SDN Perspective: Data Plane Switches
  - Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
  - Switch flow table computed, installed by controller
  - API for table-based switch control(e.g. OpenFlow)
    - defines what is controllable and what is not
  - Protocol for communicating with controller(e.g. OpenFlow)
- SDN Perspective: SDN controller
  - Maintains network state information
  - Interacts with network control applications "above" via northbound API
  - Interacts with network switches "below" via southbound API
  - Implemented as distributed system for performance, scalability, fault-tolerance robustness
- SDN Perspective: Control Applications
  - Network-control apps
    - "brains" of control: implement control functions using lower-level services, API provided by SDN controller
    - Open: can be provided by 3^{rd} party: distinct from routing vendor, or SDN controller

OpenFlow Protocol

- Operates between controller, switch(并不是路由器不参与 而是路由器被剥夺了灵魂 降级成了一台交换机 因为)
- TCP used to exchange messages
    - optional encryption
- 3 classes of OpenFlow messages:
    - controller-to-switch
    - asynchronous(switch to controller)
    - symmetric(misc)
- Controller-to-Switch Messages
    - Features: Controller queries switch features, switch replies
    - Configure: Controller queries/ sets switch configuration parameters
    - Modify - state: add, delete, modify flow entries in the OpenFlow table
    - Packet-out: controller can send this packet out of specific switch port
- Switch-to-Controller Messages
    - Packet-In: transfer packet (and its control) to controller
    - Flow-removed: flow table entry deleted at switch
    - Port status: inform controller of a change on a port
- Example: Control / Data plane Interaction



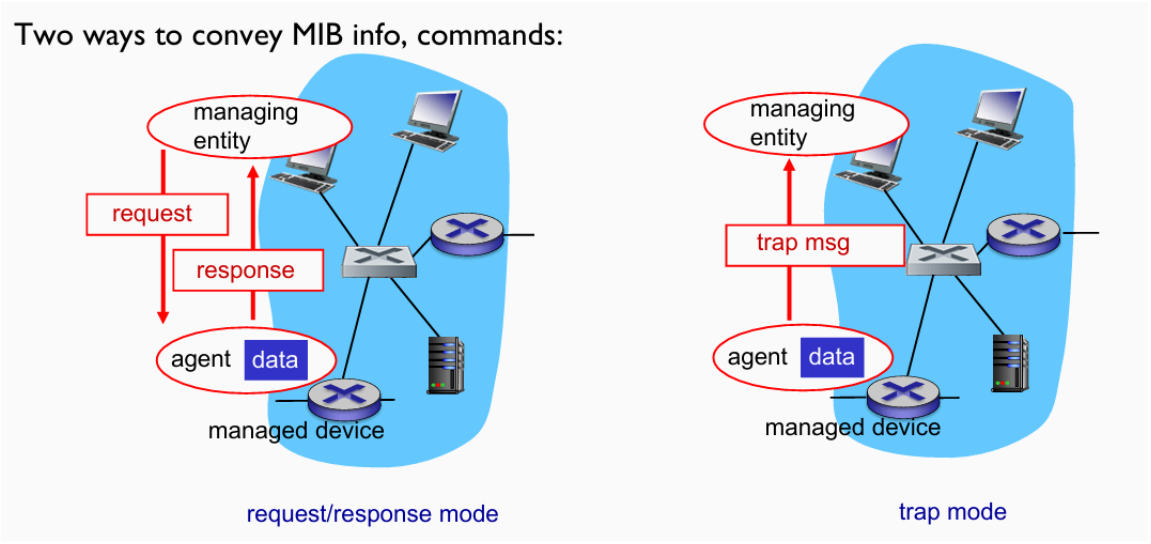
Internet Control Message Protocol (ICMP)

- used by hosts and routers to communicate network-level information
  主机和路由器之间通信
    - error reporting: unreachable host, network, port, protocol
    - echo request/reply(used by ping)

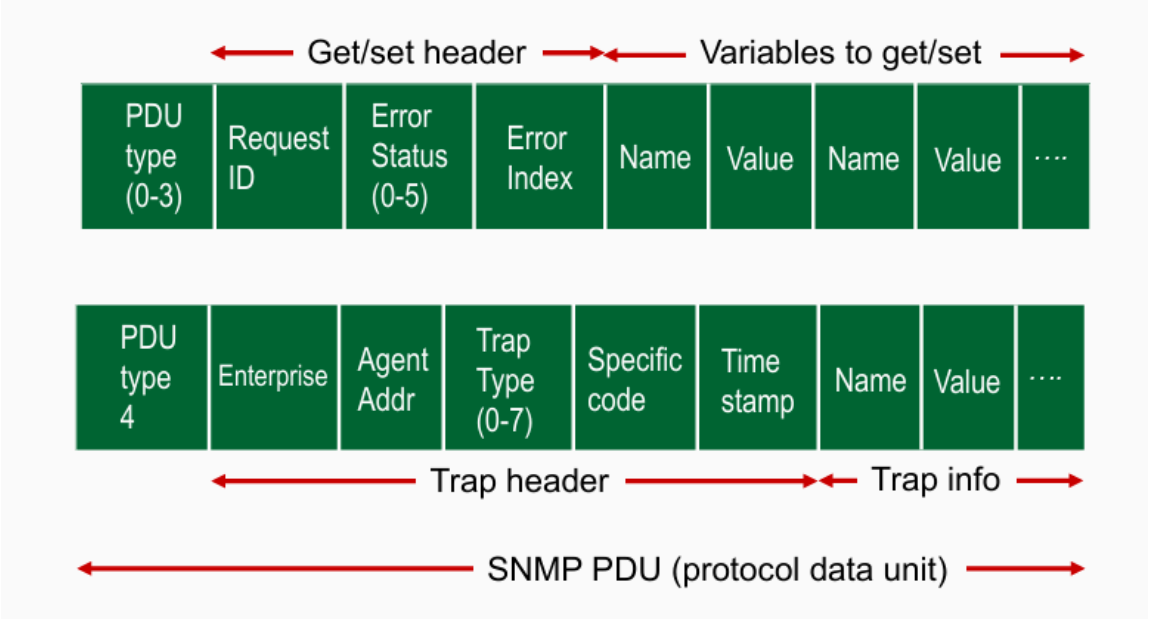
- network-layer "above" IP:
  - ICMP messages carried in IP datagrams, protocol number:1
- ICMP message: type, code plus header and first 8 bytes of IP datagram causing error
- Trace route and ICMP
  用来探测从源主机到目的地之间经过了哪些路由器：源主机向目的地发送一系列UDP数据包，故意设置TTL，当第n组包到达第n个路由器时，TTL刚好归零，路由器动作：丢弃数据包，并向源主机发送一个ICMP消息（type11，code0），意思是传输中超时，获取信息：这个ICMP回包的“源IP地址”就是这个路由器的IP，Traceroute会记录下这个地址并且计算往返时间。当UDP包最终到达目标主机时，由于目的地是一个随机的高端口（通常是接收方没有开启的端口），目标主机不会应为TTL超时，目标主机返回一个ICMP“Port Unreachable”，源主机收到这个特定报错，就知道已经摸到终点的门牌号，于是停止探测
  - source sends sets of UDP segments to destination
    - 1^{st} set has TTL = 1, 2^{nd} set has TTL = 2, etc.
  - datagram in n^{th} set arrives to n^{th} router:
    - router discards datagram and sends source ICMP message(type 11, code 0)
    - IP address of router where TTL expired is source IP address of datagram containing this ICMP message
  - when ICMP message arrives at source: record RTTs
  - Stopping criteria
    - UDP segment eventually arrives at destination host
    - destination returns ICMP "port unreachable" message(type3, code 3)

Network management, configuration

- Autonomous systems (aka. network): 1000s of interacting hardware/software components
- Managed devices contain managed objects whose data is gathered into a Management Information Base(MIB)
- SNMP(Simple Network Management Protocol)
  -用来监控/配置数据平面和控制平面



- Message Types / functions
  - GetRequest | GetNextRequest | GetBulkRequest: manager-to-agent:"get me data"
  - InformRequest: manager-to-manager: here's MIB value
  - SetRequest: manager-to-agent:set MIB value
  - Response: Agent-to-manager: value, response to Request
  - Trap: Agent- to-manager: inform manager of exceptional event
- SNMP: Message Formats



Lecture 9 The Link Layer I

Introduction

- terminology:
  - hosts, routers: nodes
  - communication channels that directly connect physically adjacent nodes: links
    直接连接物理相邻节点的通信信道：链路

- layer-2 packet: frame, encapsulates datagram
- link layer has responsibility of transferring datagram from one node to physically adjacent node over a link
  负责通过一条链路将数据包从一个节点传输到物理相邻节点

Context

- datagram transferred by different link protocols over different links
  - e.g. WiFi on first link, Ethernet on next link
- each ink protocol provides different services
  - e.g., may or may not provide reliable data transfer over link

Services

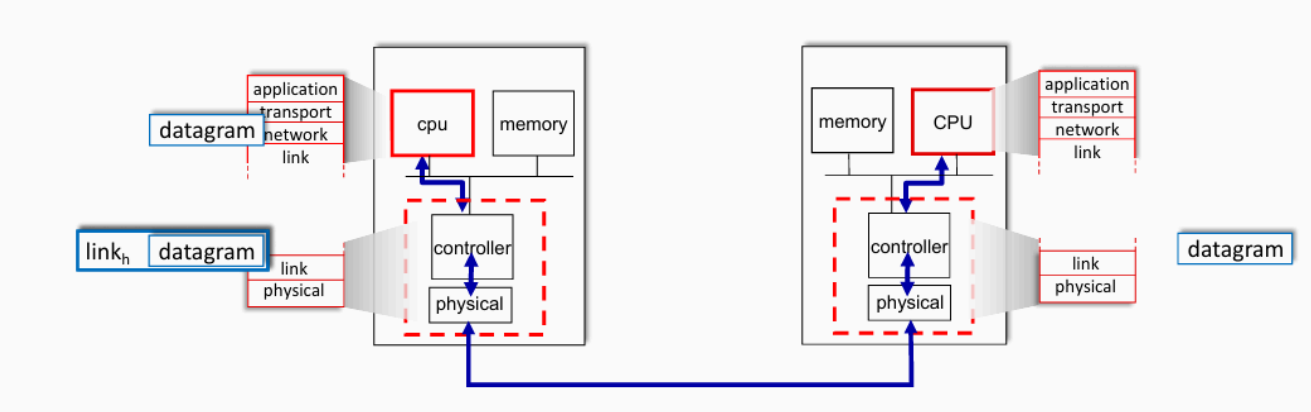
- framing, link access
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses in frame headers identify source, destination(different from IP address)
- Reliable delivery between adjacent nodes(相邻节点之间的可靠交付)
  - seldom used on low bit-error links
  - wireless links: high error rates
- Flow control
  - pacing between adjacent sending and receiving nodes
    相邻发送和接收节点间的步调控制
- error detection
  - errors caused by signal attenuation, noise
  - receiver detects errors, signals retransmission, or drops frame
- error correction
  - receiver identifies and corrects bit error(s) without retransmission
- half-duplex and full-duplex
  - with half duplex, nodes at both ends of link can transmit OR receive, but not at the same time
  - with full duplex, end and receive simultaneously

Host Link-Layer Implementation

- 链路层在电脑那个位置运行
- In each and every host
- Link layer implemented in "Adapter"(网络适配器 / 网卡) or on a chip

    - Ethernet card, WiFI 802.11 card, or chipset
    - For link and physical layers
- Attached into host's system(motherboard) buses
    - USB / PCI / Thunderbolt
- Hardware / software / firmware

Adaptors Communicating



- sending side:
    - encapsulates datagram in frame
    - adds error checking bits, reliable data transfer, flow control, etc.
- receiving side:
    - looks for errors, reliable data transfer, flow control, etc.
    - extracts datagram, passes to upper layer at receiving side

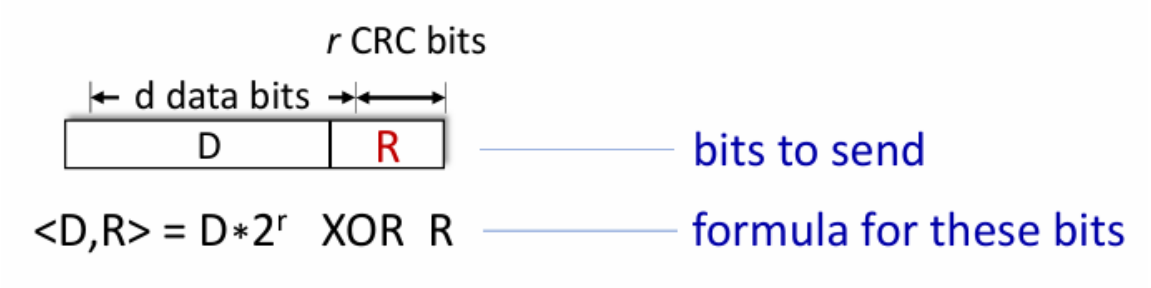
Error detection and correction Techniques

Error Detection

- EDC: error detection and correction bits(e.g., redundancy 冗余)
- D: data protected by error checking, may include header fields
- Error detection not 100% reliable
    - protocol may miss some errors, but rarely
    - larger EDC field yields better detection and correction
- Three techniques for detecting errors
    - Parity Checks
        - basic ideas(simple, detect odd or even numbers of errors)
    - Checksum
        - used in transport layer(but link layer can use variants)
    - Cyclic Redundancy Checks(CRC)
        - used in link layer in an adapter (most powerful)

- Parity Checking
    - single bit parity
        - detect single bit errors
        - Even parity: set parity so there is an even number of 1's 偶校验
        - Odd parity: set parity bit so there is an. odd number of 1's 奇校验
    - At receiver:
        - compute parity of d+1 received bits, if not even, the error detected
        - can detect odd number of bit flips(if two bits flip, it may fail)
    - 2-D parity
        - detect 2-bit errors(adds row and column parity bits)
        - detect and correct single bit errors without retransmission
        - 能纠正一个错误，i.e. 如果第二行错了 第三列也错了 那它们的焦点肯定是坏掉的那个bit位，接受方可以直接反转那个比特位
- Internet checksum
    - detect errors(i.e. flipped bits) in transmitted segment
    - sender:
        - treat contents of UDP segment(including UDP header fields and IP addresses) as sequence of 16-bit integers
        - checksum: addition(one's complement sum) of segment content
        - checksum value put into UDP checksum field
    - receiver:
        - compute checksum of received segment
        - check if computed checksum equals checksum field value
            - not equal - error detected
            - equal - no error detected. But maybe errors nonetheless
- Cyclic redundancy check(CRC)
    - more powerful error-detection coding
    - widely used in practice
    - all CRC calculations are done in modulo-2 arithmetic(without carries in addition or borrows in subtraction)
        - modulo-2 Arithmetic 加法无进位，减法无错位（不进位加法）
            - addition
                - $0+0=0$
                - $0+1=1$
                - $1+0=1$
                - $1+1=0$
            - subtraction
                - $0-0=0$

        - $0-1=1$
        - $1-0=1$
        - $1-1=0$
    - Bitwise exclusive -or(XOR)
    - $A+B=A-B=A\oplus B$
- D: data bits(given, think of these as a binary number)
- G: bit pattern(generator), of r+1 bits(given, specified in CRC standard)



- sender:
    - given D and G(r+1) bits, append r bits R such that $D\cdot 2^r\ XOR\ R$ is divisible by G(mod 2)
    - $$D\cdot 2^r\ XOR\ R=nG$$
        - 原始数据移位后，把余数填进去，得到的结果一定能被生成多项式整除
        - 2^r 表示左移r位
        - R 是算出的余数（FCS校验码）
        - nG表示能被G整除的数
- receiver:
    - knows G, divides $<D,R>$ by G. If remainder not equal to zero, error detected!
    - can detect single-bit-error, double-bit error, all burst errors less than $r+1$ bits
    - widely used in practice(Ethernet, 802.11 WiFi)
- 对于数据D：
    - 发送方把数据D后面补几个0，然后用一个标准的除数（生成多项式G）进行模2除法
    - 得到余数：这个余数R就是CRC校验码（Frame check Sequence）
    - 拼接：发送方把数据D和余数R一起发送出去
    - 接收方收到一长串东西后，用同样的除数G再做一次模2除法
    - 判断：如果余数=0，说明传输无误，如果有余数，说明出错了

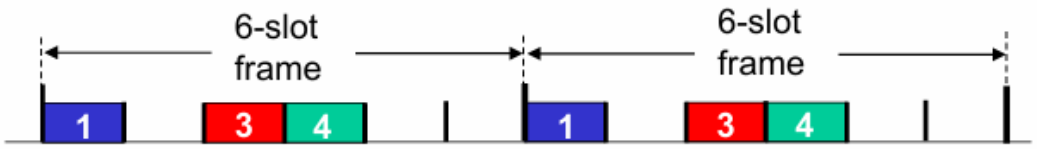
Multiple Access Links and Protocols

- Two types of "links"
    - point-to-point
        - point-to-point link between Ethernet switch, host
        - PPP for dial-up access
    - broadcast(shared wire or medium)
        - old-school Ethernet
        - upstream HFC in cable-based access network
        - 802.11 wireless LAN, 4G/5G, satellite

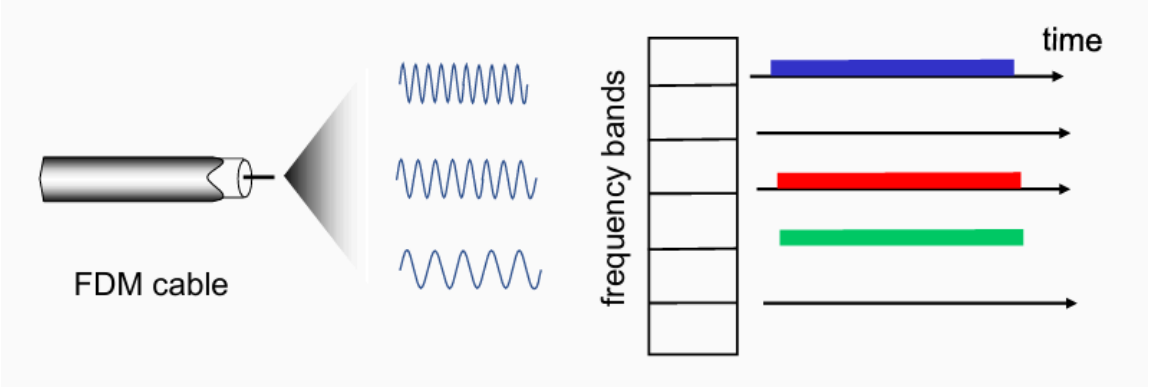
Multiple Access Protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference(干扰)
    - collision if node receives two or more signals at the same time
      节点同时收到两个或更多信号，则发生冲突
- distributed algorithm that determines how nodes share channel, i.e. determine when node transmit
- communication about channel sharing must use channel itself
    - no out-of band channel for coordination
      没有用于协调的带外信道
- Ideal Multiple Access Protocol
    - given: multiple access channel(MAC) of rate R bps
    - Requirements
        1. when one node wants to transmit, it can send at rate R
        2. when M nodes want to transmit, each can send at average rate R/M
        3. fully decentralized:
            - no special node to coordinate transmissions
            - no synchronization of clocks, slots
        4. Simple
- MAC Protocols Taxonomy
    - channel partitioning
        - divide channel into smaller "pieces"(time slots, frequency, code)
          将信道划分为更小的片段（时隙，频率， 码）
        - allocate piece to node for exclusive use
    - random access
        - channel not divided, allow collisions
        - "recover" from collisions
    - Taking turns
        - nodes take turns, but nodes with more to send can take longer turns

- Channel Partitioning MAC protocols: TDMA(time division multiple access)
  - access to channel in rounds
  - each station gets fixed length slot(length = packet transmission time) in each round
  - unused slots go idle
  - example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



- Channel Partitioning MAC protocols: FDMA(frequency division multiple access)
  - channel spectrum divided into frequency bands
  - each station assigned fixed frequency band
  - unused transmission time in frequency bands go idle
  - example: 6-station LAN, 1,,3,4 have packet to send, frequency bands 2,5,6 idle



- Random Access Protocols
  - when node has packet to send
    - transmit at full channel data rate R
    - no a priori coordination among nodes 没有先验协调
  - two or more transmitting nodes: "collision"
  - random access protocol specifies:
    - how to detect collisions
    - how to recover from collisions(e.g. via delayed retransmissions)
  - examples of random access MAC protocols
    - ALOHA, slotted ALOHA

        - CSMA, CSMA/CD, CSMA/CA
    - CSMA(Carrier Sense Multiple Access)载波侦听多路访问
        - Simple CSMA: listen before transmit
            - if channel sensed idle: transmit entire frame
            - if channel sensed busy: delay transmission 推迟传输
            - human analogy: don't interrupt others
        - CSMA/CD: CSMA with collision detection
            - collisions detected within short time
            - colliding transmissions aborted, reducing channel wastage
            - collision detection easy in wired, difficult with wireless
            - human analogy: the polite conversationalist
            - reduces the amount of time wasted in colisions(transmission aborted on collision detection)
        - collisions can still occur with carrier sensing
            - propagation delay means two nodes may not hear each other's just started transmission
              传播延迟意味着两个节点可能听不到对方刚开始的传输冲突
            - collisions: entire packet transmission time wasted
                - distance & propagation delay play role in determining collision probability

---

    - channel partitioning MAC protocols
        - Pro: share channel efficiently and fairly at high load
        - Con:inefficient at low load: delay in channel access, 1/N bandwidth allocated even if 1 active node
    - random access MAC protocols
        - Pro: efficient at low load: Single node can fully utilize channel
        - Con: high load: collision overhead

- Taking turns MAC protocols
    - look for best of both worlds
    - Polling(轮询):
        - centralized controller "invites" other nodes to transmit in turn
        - typically used with "dumb" devices
        - concerns:

        - polling overhead
        - latency
        - single point of failure(master)
    - Bluetooth uses polling
- token passing:
    - control token message explicitly passed from one node to next, sequentially
        - transmit while holding taken
    - concerns:
        - token overhead
        - latency
        - single point of failure(token)

LANs

addressing, ARP

- 32-bit IP address:
    - network layer address for interface
    - used for layer 3(network layer) forwarding
    - e.g. 128.119.40.136
- MAC(or LAN or physical or Ethernet) address:
    - function: used "logically" to get frame from one interface to another physically-connected interface(same subnet, in IP-addressing sense)
    - 48-bit MAC address(most for LANs) burned 烧录 in NIC ROM 网卡ROM, also sometimes software settable
    - e.g.: 1A-2F-BB-76-09-AD(hexadecimal notation)

- each interface on LAN
    - has unique 48-bit MAC address
    - has a locally unique 32-bit IP address
- MAC address allocation administered by IEEE
- manufacturer bus portion of MAC address space(to assure uniqueness)
- analogy:
    - MAC address: like Social Security Number
    - IP address: like postal address
- MAC flat address: portability

- can move interface from one LAN to another
- recall IP address not portable: depends on IP subnet to which node is attached

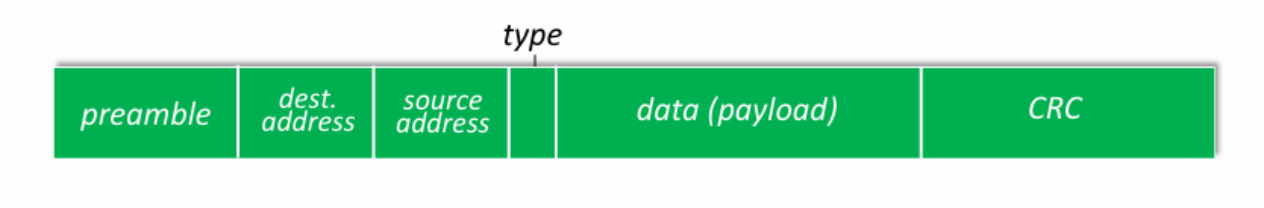
---

- ARP: Address Resolution Protocol
  - ARP table: each IP node(host, router) on LAN has table
    - IP/ MAC address mappings for some LAN nodes
      <IP address; MAC address; TTL>
    - TTL: time after which address mapping will be forgotten(typically 20 mins)
  - ARP in action
    - example: A wants to send datagram to B
      - B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address
      - Steps:
        - A broadcasts ARP query, containing B's IP address
          - destination MAC address = FF-FF-FF-FF-FF-FF(空着)
          - all nodes on LAN receive ARP query
        - B replies to A with ARP response, giving its MAC address（并且将A也记在自己的ARP表中）其他的主机看到，丢弃不予理睬
          - Q: 那其他主机为什么不顺便记录一下A的IP和MAC地址到自己的ARP表中呢
          - A: 避免缓存污染，ARP是在内存中，它的空间是有限的，设计遵循按需加载；如果表中已经有这么一条记录，那么他会更新这条记录，比如MAC变了，更新成新的；安全性考虑，会给ARP欺骗留出巨大后门，发送假广播
        - A receives B's reply, adds B entry into its local ARP table
- routing to Another subnet: Addressing
  - walkthrough: sending a datagram from A to B via R
    - focus on addressing - at IP(datagram) and MAC layer(frame) levels
    - assume that
      - A knows B's IP address(DNS)
      - A knows IP's address of first hop router R(network configuration:DHCP)

- A knows R's MAC address(ARP)

Ethernet

- "dominant" wired LAN technology
  - first widely used LAN technology
  - simpler, cheap
  - kept up with speed race: 10Mbps - 400Gbps
  - single chip, multiple speeds
- Physical Topology
  - bus: popular through mid 90s
    - all nodes in same collision domain(can collide with each other)
  - switched: prevails today
    - activate link-layer 2 switch in center
    - each spoke runs a separate Ethernet protocol(nodes do not collide with each other)
- Ethernet Frame Structure
  sending interface encapsulates IP datagram(or other network layer protocol packet) in Ethernet frame



- preamble:
  - used to synchronize receiver, sender clock rates
  - 7 bytes of 10101010 followed by one byte of 10101011
- addresses: 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address(e.g. ARP packet), it passes data in frame to network layer protocol
    如果适配器接收到目的地址匹配或广播地址（例如ARP数据包）的帧，他将帧中的数据传递给网络层协议
  - otherwise, adapter discards frame
- type: indicates higher layer protocol
  - mostly IP but others possible, e.g., Novell IPX, AppleTalk
- CRC: cyclic redundancy check at receiver
  - error detected: frame is dropped
- Features:
  - connectionless: no handshaking between sending and receiving NICs
  - unreliable: receiving NIC does send ACKs or NAKs to sending NIC

    - data in dropped frames recovered only if initial sender uses higher layer rdt(e.g. TCP), otherwise dropped data lost
  - Ethernet's MAC protocol: un-slotted CSMA/CD with binary backoff

Conclusion

- MAC 在链路层IP地址没用，MAC扁平化，无论何时何地，MAC地址都不会变，这里有一个问题就是既然MAC唯一标识，那为什么不能直接删掉IP?
  - MAC地址只能告诉我是谁，但只有IP地址能告诉大家“我在哪儿”，MAC地址是乱序的，是厂家生产网卡时，烧录进去的，而IP是有层级的，最长前缀匹配规则，可以分层分配。如果只用MAC地址通信，全球MAC地址没有地域规律，全世界的每一个路由器都要维护一张包含几十亿条记录的表格，记录每一台手机，每一台电脑具体在哪条网线上；如果用IP，路由器用CIDR进行聚合，路由器只需要记住：前缀以及方向即可，路由表可以就此减小。另外的，可以屏蔽底层差异，不论底层是光纤，网线还是信鸽，IP协议都把它们统一封装起来。IP地址给所有设备提供了一个统一的语言
- 交通规则：MAC协议（信道划分/ 随机接入CSMA，CSMA/CD / 轮询），规定 谁什么时候可以说话
- 具体的实现：以太网-有限局域网技术，把上面的都融合起来，早期使用总线型，巨大冲突域，必须使用CSMA/CD，现代改为switch型，中间由交换机连接
- 有了ID(MAC)才能定点传输，有了ARP才能根据IP找到MAC，有了CSMA才能防止冲突，有了Ethernet和switch构成现代网络

Lecture 10 The Link Layer II

LANs

Switches

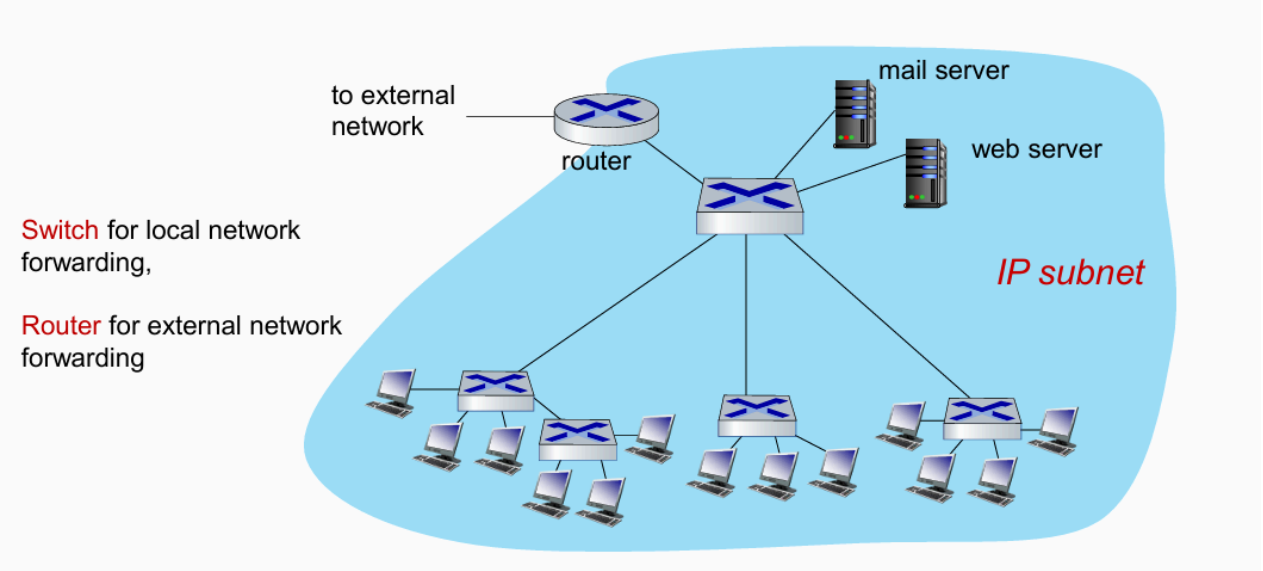
- switch is a link-layer device: takes an active role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment 检查传入帧的MAC地址，有选择的将帧转发到一个或多个出站链路，当帧需要在网段上转发时，使用CSMA/CD访问网段
- transparent: hosts unaware of presence of switches
- plug-and-play, self learning
  - switches doesn't need to be configured
- multiple simultaneous transmissions 多路同时传输
  - hosts have dedicates, direct connection to switch
  - switches buffer packets
  - Ethernet protocol used on each incoming link, so:

        - no collisions, full duplex
        - each link is its own collision domain
    - switching: A-to-A' and B-to-B' can transmit simultaneously, without collisions
        - but A-to-A' and C to A; can not happen simultaneously
- Switch forwarding table
    - each switch has a switch table, each entry has MAC address of host, interface to reach host, time stamp
    - looks like a routing table
- Switch: Self learning
    - switch learns which host can be reached through which interfaces
        - when frame received, switch learns location of sender: incoming LAN segment
        - records sender / location pair in switch table
        - example:
            - frame destination, A' location unknown: flood
            - destination A location known: selectively send on just one link
    - ARP & switch self learning
        - ARP 介于网络层和链路层之间 知道目标的IP地址，想要得到目标的MAC地址
        - Self-learning 看到了帧的源MAC地址，记住这个MAC地址在那个端口
        - 合作过程：
            - 主机A知道A‘的IP，但不知道MAC地址
            - 主机A发送一个ARP请求， 谁是192.168.88.130，请把你的MAC告诉我（广播帧）
            - 交换机收到这个ARP包，自学习，看了一眼源MAC（A的MAC），原来A在端口1，记录到switch table
            - A’收到请求，回复我说192.168.88.130，我的MAC是 ...
            - 交换机处理响应：
                - 自学习：交换机看了一眼源MAC（A‘的MAC），记录到switch table
                - 查交换表，得知A在端口1，于是把包发给端口1
- Switch: frame filtering/ forwarding
    - when frame received at switch
        - record incoming link, MAC address of sending host
        - index switch table using MAC destination
        - if entry found for destination
          then{
          if destination on segment from which frame arrived

then drop frame
else forward frame on interface indicates by entry
}
else flood / *forward on all interface except arriving interface* /

- 先更新MAC地址表，然后查找目的地，决定转发，过滤还是泛洪， 如果找不到则泛洪，找到了，但在别处，转发，在隔壁则过滤/丢弃， 防止资源的浪费

- Small institutional network



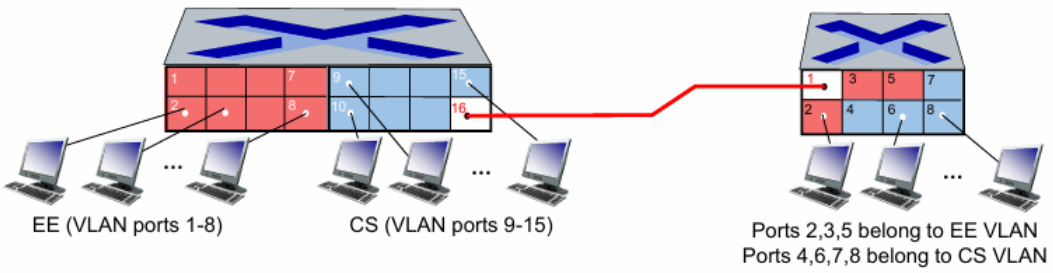
Switches vs routers

- both are store-and-forward
  - routers: network-layer devices(examine network-layer header)
  - switches: link-layer devices(examine link-layer header)
- both have forwarding tables
  - routers: compute tables using routing algorithms, IP addresses
  - switches: learn forwarding table using flooding, learn, MAC addresses

VLANs

- Motivation
  - If the LAN sizes scale, users change point of attachment
    - single broadcast domain:
      - scaling: all layer-2 broad cast traffic(ARP,DHCP.unknown MAC) must cross entire LAN
      - efficiency, security, privacy issues
    - administrative issues:
      - CS users moves office to EE -physically attached to EE switch, but wants to remain logically attached to CS switch

- Port-based VLANs
    - switches supporting VLAN capabilities can be configured to define multiple virtual LANs over single physical LAN infrastructure
      支持虚拟局域网功能的交换机可以通过配置，在单一物理局域网基础设施上定义多个虚拟局域网
    - port-based VLAN: switch ports grouped(by switch management software) so that single physical switch operates as multiple virtual switches
    - traffic isolation: frames to/from ports 1-8 can only reach ports 1-8
        - can also define VLAN based on MAC addresses of endpoints, rather than switch port
    - dynamic membership: ports can be dynamically assigned among VLANs
        - 动态成员资格：端口可以动态分配给不同的虚拟局域网
    - forwarding between VLANs: done via routing (just as with separate switches)
        - in practice vendors sell combined switches plus routers
    - trunk port: carries frames between VLANs defined over multiple physical switches



data center networking

- challenges:
    - multiple applications, each serving massive numbers of clients
    - Managing / balancing load, avoiding processing, networking, data bottlenecks
- Load balancer: application-layer routing
    - receives external client requests
    - Directs workload within data center
    - Returns results to external client(hiding data center internals from client)
    - Apache, Nginx or DIY using Python / nodejs
- Rich interconnection among switches, racks:
    - Increased throughput between racks(multiple routing paths possible)
    - Increased reliability via redundancy 通过冗余提高可靠性

Lecture 10: Network Security I

- Definition
    - Confidentiality: only sender, intended receiver should understand message contents
        - sender encrypts message
        - receiver encrypts message
    - Integrity: sender, receiver want to ensure message not altered(in transit, or afterwards) without detection
    - Authentication: sender, receiver want to confirm identity of each other
    - Access and Availability: services must be accessible and available to users

Principles of cryptography

breaking an encryption scheme

- cipher-text only attack: Trudy has cipher-text she can analyze 唯明文攻击
    - 2 approaches:
        - brute force: search through all keys
        - statistical analysis
- Known-plaintext attack:
    - Trudy has known plaintext corresponding to cipher-text(pairs)
        - e.g., in mono-alphabetic cipher, Trudy determines pairings for a, l, i, c, e, b, o
- chosen-plaintext attack
    - Trudy can choose/feed the plaintext message and obtain its corresponding cipher-text form

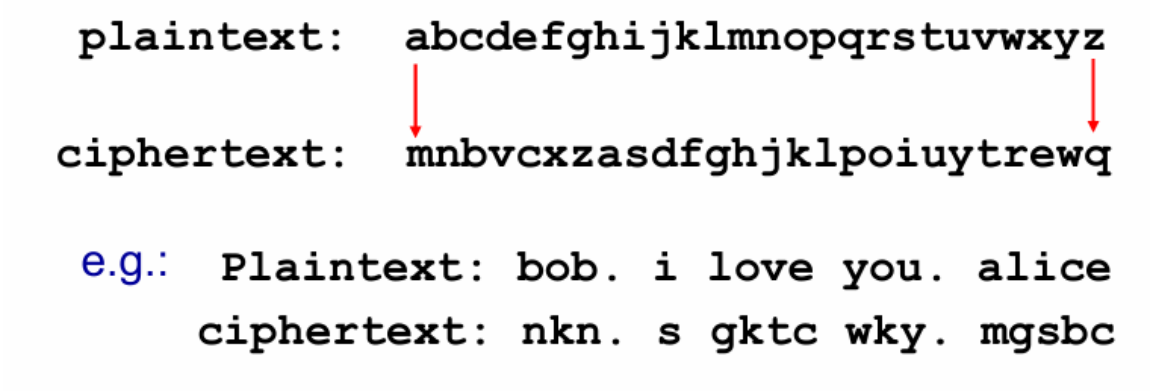
Symmetric key cryptography

- Two ends share same(symmetric) key, fast but requires secure key sharing(key distribution problem)
- e.g. Caesar Cipher
    - For English text, the Caesar cipher would work by taking each letter in the plaintext message and substituting the letter that is k letters later(allowing wraparound) in the alphabet

encryption schema

- substitution cipher: substituting one thin for another

- mono-alphabetic cipher: substitute one letter for another

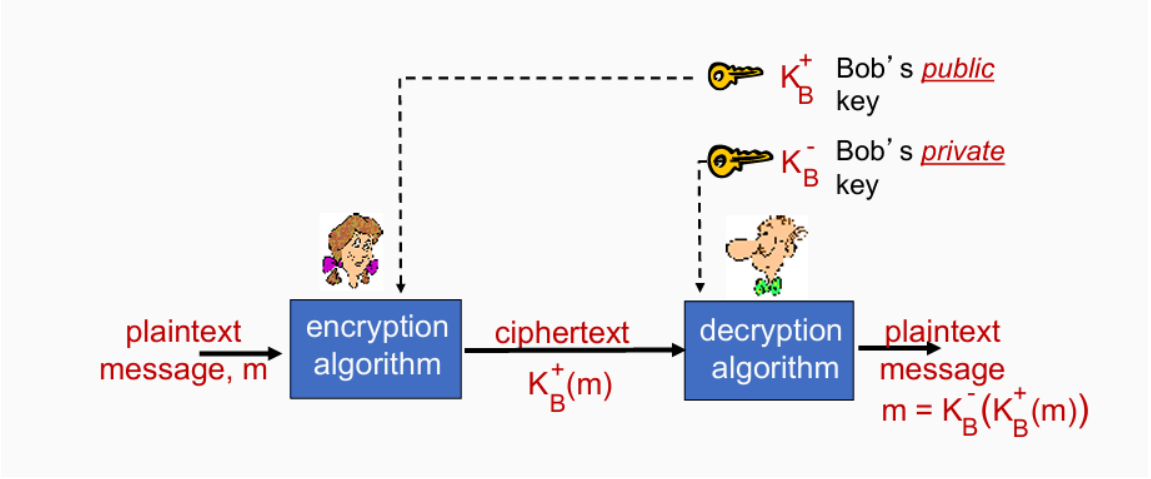


- A more sophisticated encryption approach(多密码本)
  - n substitution(poly-alphabetic) ciphers, $M_1, M_2, \ldots M_n$
  - cycling pattern:
    - e.g., $n = 4 : M_1, M_3, M_4, M_3, M_2$
  - for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
    - dog： d from M_1, o from M_3, g from M_4
- Symmetric ket crypto: DES
  - Data Encryption Standard
    - US encryption standard
    - 56-bit symmetric key, 64-bit plaintext input
    - block cipher with cipher block chaining
    - weak by modern standards
    - making DES more secure:
      - 3DES: encrypt 3 times with 3 different keys
- AES: Advanced Encryption Standard
  - symmetric-key NIST standard, replaced DES
  - processes data in 128 bit blocks
  - 128, 192, or 256 bit keys
  - a machine could brute force decryption(try each key) taking 1 sec on DES, takes 149 trillion years for AES

Public Key Cryptography

- symmetric key crypto
  - requires sender, receiver know shared secret key
  - but how to agree on key in first place?
- Public key crypto
  - totally different approach
  - sender, receiver do not share secret key

- public encryption key known to all
- private decryption ket known only to receiver



- given public key k_B^+. it should be impossible to compute private k_B^-

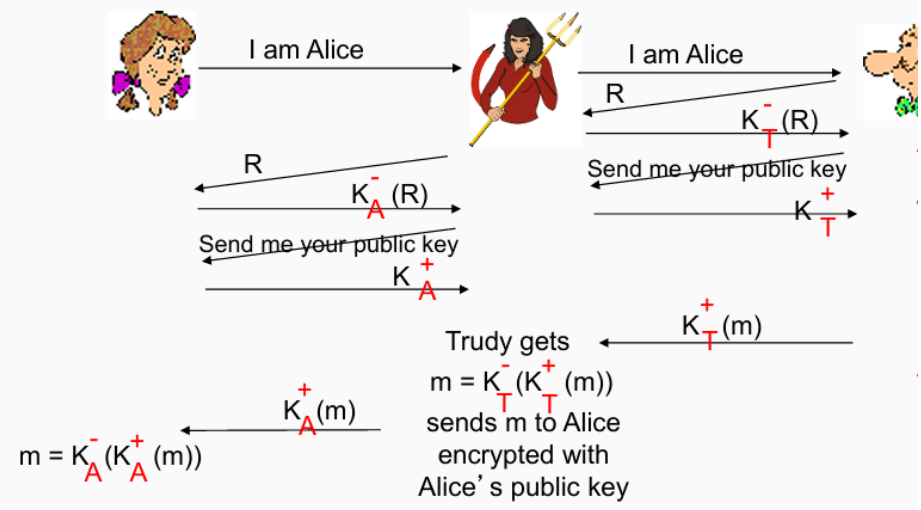
- RSA: Rivest, Shamir, Adleman algorithm
  - Getting ready
    - message: just a bit pattern
    - bit pattern can be uniquely represented by an integer number
    - thus, encrypting a message is equivalent to encrypting a number
      加密消息等价于加密一个数字
  - Creating public / private key pair
    1. choose two large prime numbers p, q.
    2. compute $n = pq$, $z = (p-1)(q-1)$
    3. choose e(with $e < n$) that has no common factors with z(e, z are "relatively prime")
    4. choose d such that $ed - 1$ is exactly divisible by z ($ed \ mod \ z = 1$)
    5. public key is (n, e), private key is (n, d)
  - encryption, decryption
    0. given (n, e) and (n, d) as computed above
    1. to encrypt message $m(< n)$, compute $c = m^e \ mod \ n$
    2. to decrypt received bit pattern, c, compute $m = c^d \ mod \ n$
  - an important property
    - use public key first, followed by private key is the same as using private key first, followed by public key
  - Why is RSA secure
    - suppose you know public key (n, e). how hard is to determine d?
    - essentially need to find factors of n without knowing the two factors p and q
  - RSA in practice: session key
    - DES is at least 100 times faster than RSA

- solution: Hybrid encryption: use public key crypto to establish a secure connection, then establish a second key - a symmetric session key- for encrypting data
- Bob and Alice use RSA to exchange a symmetric key K_s
- Once both have K_s, they use symmetric key cryptography

Lecture 11 Network Security II

Authentication

- Goal: Bob wants Alice to "prove" her identity to him
  - $\underline{Protocol\ ap1.0}$: Alice says "I'm Alice"
    but in a network Bob cannot see Alice, so Trudy simply declares herself to be Alice
  - $\underline{Protocol\ ap2.0}$:Alice says "I'm Alice" in an IP packet containing her source IP address
    Trudy can create a packet "spoofing" Alice's address
  - $\underline{Protocol\ ap3.0}$: Alice says "I'm Alice" and sends her secret password to prove it
    playback attack: Trudy records Alices's packet and later plays it back to Bob
  - $\underline{Protocol\ ap3.1}$: Alice says "I'm Alice" and sends her encrypted secret password to prove it
    record and playback still works
  - $\underline{Protocol\ ap4.0}$: Alice says "I'm Alice", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key
    - requires shared symmetric key. can we using public key techniques?
  - $\underline{Protocol\ ap5.0}$: use nonce, public key cryptography
    - Alice says "I'm Alice", Bob sends R, Bob computes $K_A^+(K_A^-(R)) = R$ and knows only Alice could have the private key, that encrypted R such that $K_A^+(K_A^-(R)) = R$
    - security hole:
      - man (or woman) in the middle attack: Trudy poses as Alice(to Bob) and as Bob(to Alice)

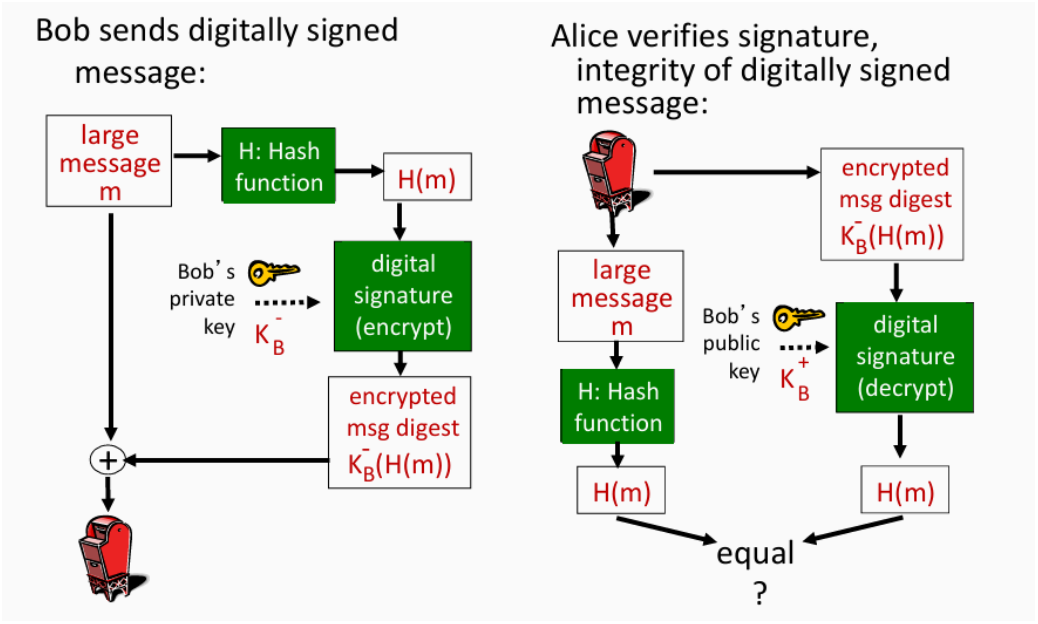


    - Even with encryption, Trudy intercepted and sent her own encrypted data is still works
- Authentication Goal 1:
    - Data origin authentication: confirm who created / sent the message
    - solution: Message Authentication Code(MAC)
- Authentication Goal 2:
    - Message integrity: ensure the message was not modified in transit
    - Solution: Digital Signatures

Message Integrity

- Digital signatures
    - cryptographic technique analogous(likened) to hand-written signatures
        - sender digitally signs the document, establishing he is the document owner / creator.
        - verifiable, unforgeable: recipient(Alice) can prove to someone that Bob, and so no one else, must have signed document
- Message digests 消息摘要
    - Problem of Public Key Encryption:
        - Computationally expensive to public-key -encrypt long messages 使用公钥加密长信息计算成本高
    - Message Goal: fixed-length, easy-to-compute digital "fingerprint"
        - apply hash function H to m, get fixed size message digest, H(m).
    - Hash function
        - ensures integrity -any bit change creates a new digest
        - properties:
            - many-to-1

        - produces fixed-size message digest(fingerprint)
        - Easy to compute: H(m)
        - given message digest x, computationally infeasible to find m such that x=H(m)
    - Digital signature = signed message digest



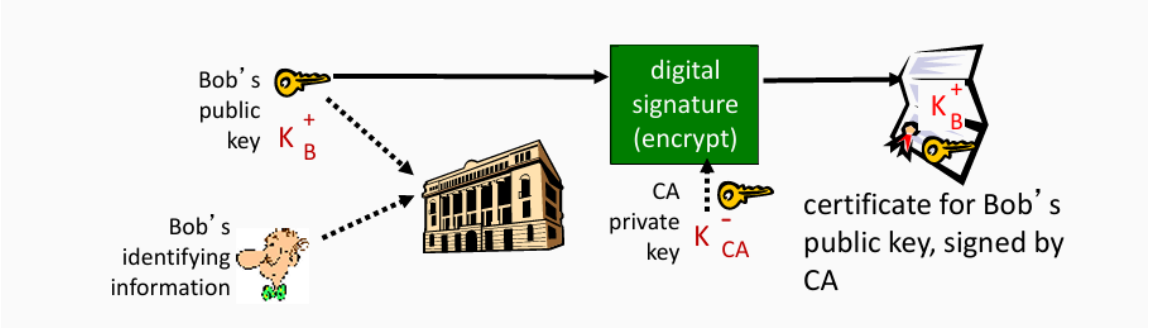
    - Hash + Digital Signature
      Instead of signing the full message(expensive), sign only its hash

- Internet checksum: poor crypto hash function
  Internet checksum has some properties of hash function
  - Produces a fixed length digest(16-bit sum) of message
  - is many-to-one
  - 核心算法是反码求和，对抗篡改能力是很弱的
- Hash function algorithms
    - MD5 hash function widely used
        - computes 128-bit message digest in 4-step process
        - arbitrary 128-bit string x, appears difficult to construct message m whose MD5 hash is equal to x
    - SHA-1 is also used
        - 160-bit message digest
    - SHA-256

Public-key certification

- Certification authority(CA): binds public key to particular entity, E
- E (person, router) register its public key with CA
    - E provides "proof of identity" to CA
    - CA creates certificate binding E to its public key

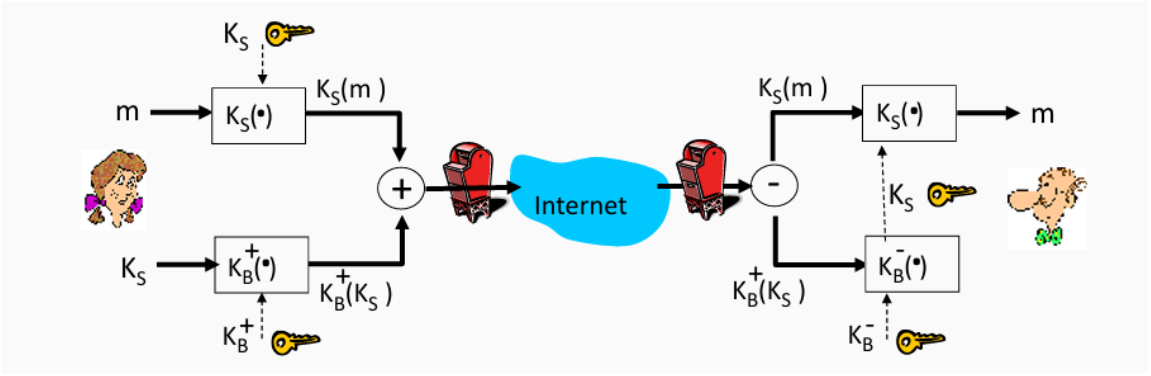
- certificate containing E's public key digitally signed by CA-CA says "this is E's public key"



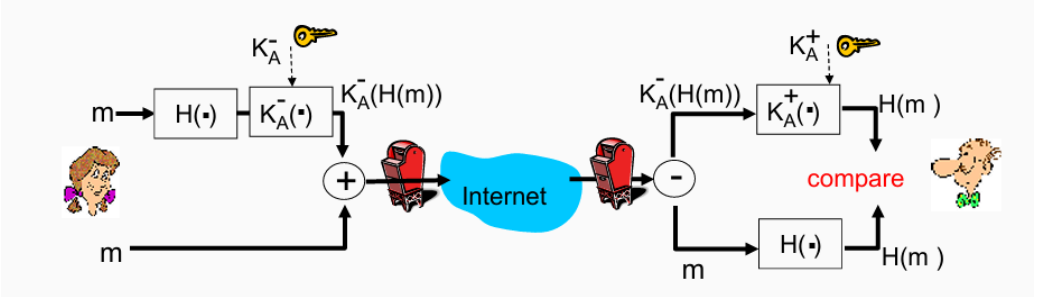
- When Alice wants Bob's public key:
  - gets Bob's certificate(Bob or elsewhere)
  - apply CA's public ket to Bob's certificate, get Bob's public key
  - 这避免了ap5.0中的安全问题

Securing e-mail

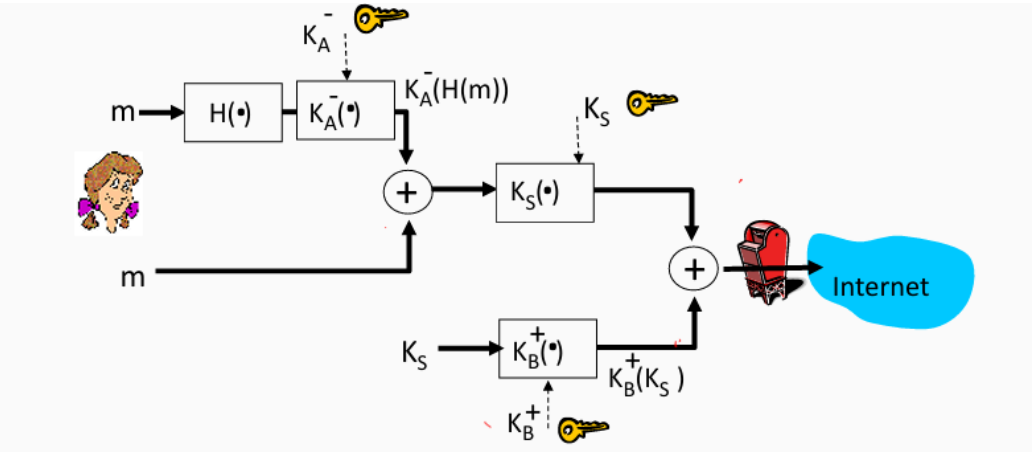
- Sender
  - generates random symmetric private key, K_s
  - encrypts message with K_s
  - also encrypts K_s with Bob's public key
  - sends both $K_s(m)$ and $K_B^+(K_s)$ to Bob



  - If he wants to provide sender authentication message integrity
    - Alice digitally signs message
    - sends both message(in the clear) and digital signature



  - If he wants to provide secrecy, sender authentication, message integrity
    - Alice uses three keys: her private key, Bob's public key, newly created symmetric key

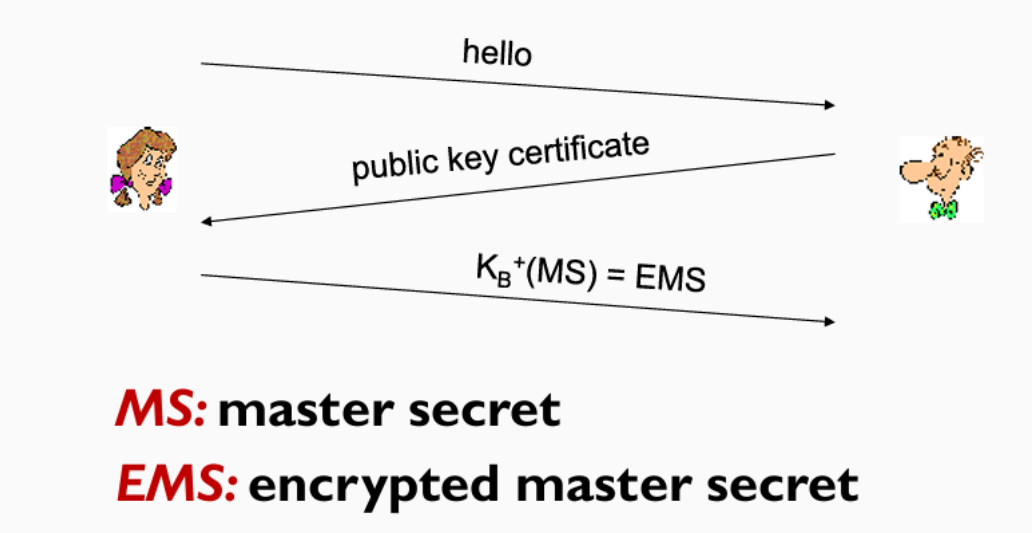


  - Alice先把消息m放入哈希函数中，得到一个指纹，然后用自己的私钥对这个指纹加密，Alice又生成了一个临时的会话密钥（对称密钥），他要用这个K_s把消息和数字签名的指纹打包，但是Bob现在没有K_s，Alice用Bob的公钥把K_s加密起来，
  - Bob收到后，用自己的私钥先解密，得到K_s，然后拆K_s对称密钥下的包，得到明文消息m，和Alice的数字签名，然后验证身份，先拿出Alice的公钥去解密那个数字签名，如果解开了就得到原始指纹$H(m)$，Bob比较自己刚解密出来的明文m跑出来的哈希函数$H(m\backslash)$和$H(m)$一样吗，如果指纹一样，说明没被修改过
- Receiver
  - uses his private key to decrypt and recover K_s
  - uses K_s to decrypt $K_s(m)$ to recover m

Securing TCP connections: SSL

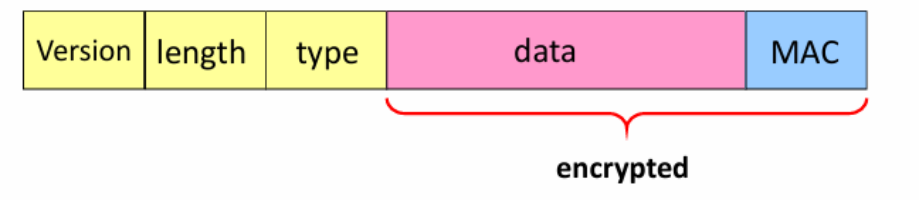
- SSL: Secure Sockets Layer
  - widely developed security protocol
    - support by almost all browsers, web servers
    - https
    - billions $/year over SSL
  - variation -TLS: transport layer security
  - provides
    - confidentiality
    - integrity
    - authentication
  - original goals
    - Web e-commerce transactions
    - encryption(especially credit-card numbers)
    - Web-server authentication
    - optional client authentication

        - minimum hassle in doing business with new merchant
    - available to all TCP applications
        - secure socket interface
- Toy SSL: a simple secure channel
    - Four phases
        - handshake: Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret



        - key derivation: Alice and Bob shared secret to derive set of keys
            - considered bad use same key(the master secret key) for more than one cryptographic operation
                - use different keys for message authentication code(MAC) and encryption
                - $MAC = H(m + s)$ m:message s: MAC key
            - four keys generated from the MS:
                - $K_c =$ encryption key for data sent from client to server
                - $M_c =$ MAC key for data sent from client to server
                - $K_s =$ encryption key for data sent from server to client
                - $M_c =$ MAC key for data sent from server to client
            - keys derived from key derivation function(KDF)
                - takes master secret(MS) and possibly some additional random data and creates the key
        - data transfer: data to be transferred is broken up into series of records
            - break stream in series of records
                - each record carries a MAC
                - receiver can act on each record as it arrives
            - 为什么不直接加密TCP数据流？如果整个数据流，MAC只能放在最后，在连接结束前，都不知道数据的完整性
            - issue: attacker can capture and replay record or re-oder records

        - solution: put sequence number into MAC:
            - MAC = $H(M_x, sequence||data)$
            - note: no sequence number field(in the record)
        - issue: attacker could replay all records
        - solution: use nonce
    - connection closure: special messages to securely close connection
        - problem: truncation attack(by MITM attacker)截断攻击
            - attacker forges TCP connection close segment(FIN)
            - one or both sides thinks there is less data than actually is
        - solution: different record types, with one type for closure
            - type 0 for data; type 1 for closure
        - MAC = $H(M_x, sequence||type||data)$



- Toy SSL isn't complete
    - How long are the fields
    - which encryption protocols
    - client and server want negotiation
        - allow client and server to support different encryption algorithms
        - allow client and server to choose together specific algorithm before data transfer

- Real SSL
    - SSL cipher suite
        - public-key algorithm
        - symmetric encryption algorithm
        - MAC algorithm
          SSL supports several cipher suites
        - negotiation: Client, server agree on cipher suite
            - client offers choice
            - server picks one
    - Handshake
        - Purpose
            - server authentication
            - negotiation: agree on crypto algorithms
            - establish keys

        - client authentication(optical)

    1. client sends list of algorithms it supports, along with client nonce
    2. server chooses algorithms from list; sends back: choice + certificate + server nonce
    3. client verifies certificate, extracts server's public key, generates `pre_master_secret`, encrypts with server's public key, sends to server
    4. client and server independently compute encryption and MAC keys from `pre_master_secret` and nonces
    5. client sends a MAC of all the handshake messages
    6. server sends a MAC of all the handshake messages

    - 最后两步保护握手免受篡改。在步骤1中，客户端提供的一系列算法可能有的强有的弱，中间人可以删除列表中较强的算法，最后两步防止了这种情况应为都是加密的

Lecture 12 Network Security III

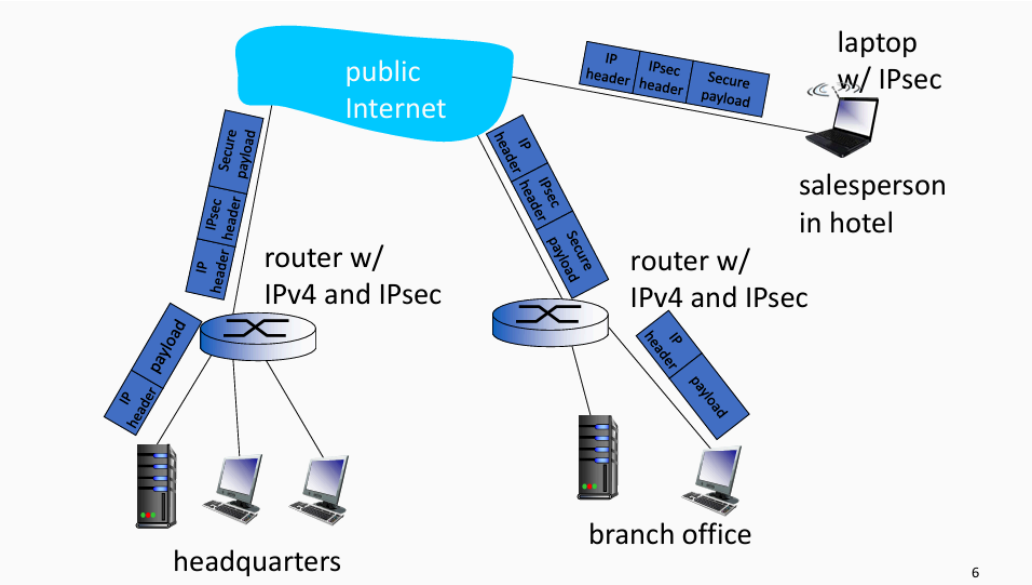
Network layer security: IPSec

- TLS secures individual connections; IP layer security protects all packets
- Between 2 network entities
    - sending entity encrypts datagram payload, payload could be TCP / UDP segment, ICMP message, OSPF message
    - all data sent from one entity to other would be hidden from any third party(that presumably is sniffing the network):
        - web pages, emails, P2P file transfers, TCP SYN packets...
    - blanket coverage across apps

Virtual Private Networks(VPNs)

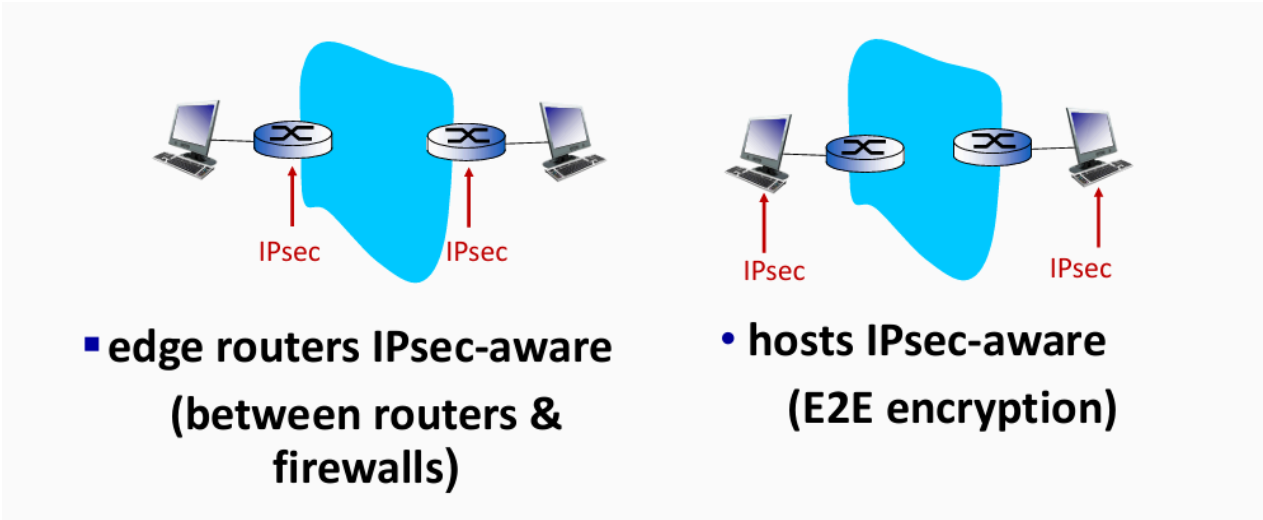
- Motivation:
    - Institutions often want private networks for security
        - An institution could actually deploy a stand-alone physical network that is completely separate from the public internet
        - costly: separate routers, links, DNS infrastructure
    - VPN: institution's inter-office is sent over public Internet instead
        - encrypted before entering public internet

- Logically separate from other traffic



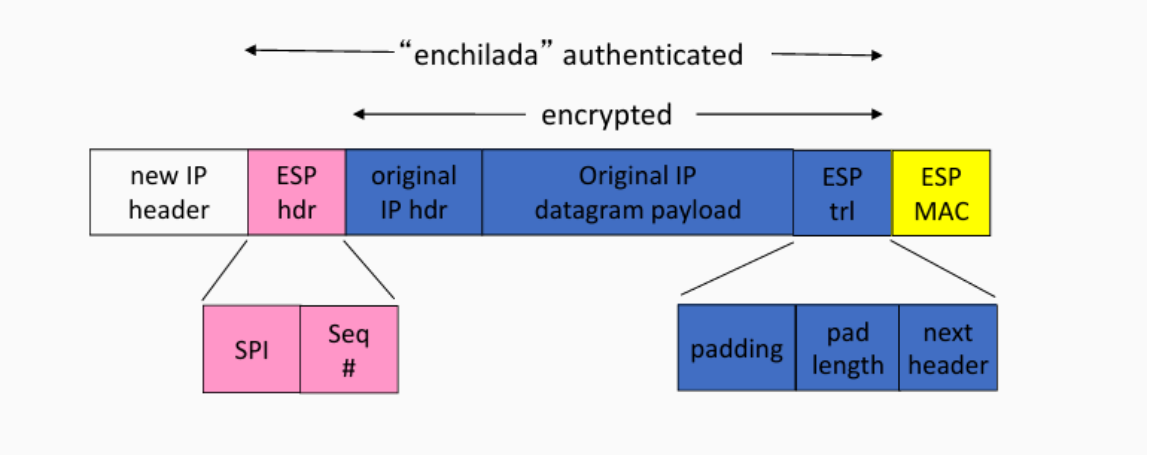
IPsec

- data integrity - detect modification
- origin authentication - confirm sender identity
- replay attack prevention - stop packet re-injection
- confidentiality - encrypt payload
- 2 protocols providing different service models
  - Authentication Header(AH)
    - provides source authentication and data integrity but no confidentiality
      提供源认证和数据完整性，但不提供机密性
  - Encapsulation Security Payload(ESP)
    - provides source authentication, data integrity, and confidentiality
    - more widely used than AH
- tunneling mode & host mode



- Security Associations(SAs)
  - 逻辑，单向连接

    - before sending data, SA established from sending to receiving entity
        - SAs are simplex: logical connection for only one direction
    - ending, receiving entitles maintain state information about SA
    - how many SAs in VPN w/ one headquarters office, one branch office, and n traveling salesperson? --2n+2
- Security Association Database(SAD)
    - 记录了当前所有活跃连接的具体参数（密钥是什么，算法是AES还是DES，序列号是多少），路由器收到一个加密包，看里面的SPI编号，去查SAD表，找到对应的解密钥匙
    - Each endpoint holds SA state in security association database(SAD), where it can locate them during processing
    - when sending IPsec datagram, R_1 accesses SAD to determine how to process datagram
    - when IPsec datagram arrives to R_2, R_2 examines Security parameter index(SPI) in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly
- IPsec datagram
    - focus for now on tunnel mode with ESP



    - How to convert original datagram to IPsec datagram
        - appends to back of original datagram(that includes original fields) an ESP trailer field
        - encrypts result using algorithm & key specified by SA
        - appends to front of this encrypted quantity the "ESP header"
        - creates authentication MAC over the whole entity, using algorithm and key specified in SA
        - appends MAC to back of the entity, forming payload
        - creates band new IP header, with all the classic IPv4 header fields, which it appends before payload
- Security Policy Database(SPD) 策略
    - policy: For a given datagram, sending entity needs to know if it should use IPsec or normal IP

- needs also to know which SA to use
    - may use: source and destination IP address; protocol number
- Info in SPD indicates "what" to do with arriving datagram
- info in SAD indicates "how" to do it

IKE：Internet key Exchange

- previous examples: manual establishment of IPsec SAs in IPsec endpoints
- manual keying is impractical for VPN with 100s of endpoint
- instead use IPsec IKE(Internet Key Exchange) protocol, specified in RFC 5996

Conclusion

VPN是目的，IPsec是实现手段。互联网是不安全的，之前的TLS只能保护某个应用/网页，而IPsec可以保护整个IP层，只要经过IP层统统加密，不愿铺设昂贵的物理专线，选择VPN这种形式。IPsec包含两个主要的协议AH和ESP，AH只防止篡改不加密，而ESP可以同时做到是VPN的主力。

聚焦ESP，他的核心模式是Tunnel Mode，在路由器对路由器的场景下，你的电脑发出的包是明文，到了公司的边界路由器，路由器会加密封装成一个新的IP包发给对端服务器，对于数据包的处理是先加上尾巴，在后面加填充物（padding），为了凑整加密块大小，然后加密，把原始包加上尾巴都加密，然后加头，在前面加上SPI索引号和序列号。对ESP头 + 加密内容 + 尾巴计算MAC值，贴在最后面，然后套上一个新的IP头，源/目的地址是路由器的地址，而不是电脑的地址
IPsec的大脑是SA/SAD/SPD，SA是逻辑单向连接，SAD内部配置具体怎么加密，这是由IKE协商决定的，然后SAD来负责存储。当路由器收到一个加密包，看里面的SPI编号，查SAD表找到对应的解密钥匙，SPD则在路由器发包之前先查SPD，决定要不要加密，要不要建立SA。

Operational security: firewalls and IDS

Firewalls

- isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others
- Purpose
    - prevent denial of service attacks
        - SYN flooding: attacker establishes many bogus TCP connections, no resources left for "real" connections
    - prevent illegal modification / access of internal data
        - e.g. attacker replaces CIA's homepage with something else
    - allow only authorized access to inside network
        - set of authenticated users / hosts

- 3 types of firewalls
  - stateless packet filters(network layer) 只看IP头和端口号
    - Internal network connected to Internet via router firewall
    - router filters packet-by-packet, decision to forward / drop packet based on:
      - source IP address, destination IP address
      - TCP/UDP source and destination port numbers
      - ICMP message type
      - TCP SYN and ACK bits
  - stateful packet filters(transport layer)会跟踪TCP状态
    - track status of every TCP connection
      - track connection setup(SYN), teardown(FIN): determine whether incoming, outgoing packets "make sense"
      - timeout inactive connections at firewall: no longer admit packets
  - application gateways(application layer)
    - filter packets on application data as well as on IP/TCP/UDP fields
    - example: allow select internal users to telnet outside
      - require all telnet users to telnet through gateway
      - for authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
      - router filter blocks all telnet connections not originating from gateway

IDS

- packet filtering:
  - operates on TCP/IP headers only
  - no correlation check among sessions
- IDS: intrusion detection system
  - deep packet inspection: look at packet contents(将数据包中的字符串与已知病毒攻击字符串进行比对)
  - examine correlation among multiple packets
    - port scanning
    - network mapping
    - DoS attack