Lecture note_v3

2019年12月24日 10:05

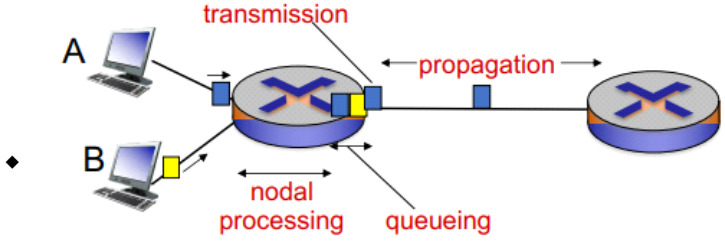
- 1/5 Lecture 1(Network edge, Network, core, access net, end system==end system)
  - Why does standard is important for protocol?
    - Protocols define the format, order of message send and received among network entities, and actions taken on message transmission.
  - Infrastructure:
    - **End hosts:** The things that connected to internet, i.e. PC, server, wireless laptop, smartphone
    - **Communication links**: i.e. Fiber, copper wire, radio, satellite.
    - **packet switch:** i.e. Router and switches
  - **Network edge**: refers to device at end-system, such as laptop, iPad, cell phone
  - **Network core:** Interconnected Routers, network of network, the backbone of network structure
  - **Access net**: the connection between edge and core, e.g.Ethernet, WIFI network, celler tower.
  - End Systems: the host that run application/program, e.g. web, email
  - The TCP/IP Five-layer network model:
    -

#	Layer Name	Protocol	Protocol Data Unit	Addressing
5	Application	HTTP, SMTP, etc..	Messages	n/a
4	Transport	TCP/UDP	Segment	Port #'s
3	Network	IP	Datagram	IP address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

- 1/8 Lecture2 (Packet switching Vs Circuit switching)
  - **Packet switching**
    - Congestion can happen, packet might get dropped, retransmission can happen
    - Host
    - Router
    - Store and forward
    - Queuing
    - What are the advantage of packet switching?
      - Good for bursty data
      - Resource share
      - Simpler, and cheaper to setup
    - What are the dis-advantage of packet switching?
      - Possible to loss packet if congestion is heavy
  - **Circuit switching (old one)**
    - An very old communication system for telephone network, an physical electrical connection need to be established between two host.
    - End-to-End resourced reserved
    - Dedicated resource, so no sharing resources
    - Performance guaranteed
    - Common in traditional telephone network
    - What are the advantage of circuit switching?
      - Guaranteed performance(throughput, delay)
    - What are the dis-advantage of circuit switching?
      - Complex and expensive to setup
  - Packet switching Vs Circuit switch
    - CS has explicity set up phase but PS doesn't.
    - CS reserved resources, and each

    - Why do we choose Packet switch over Circuit switch?
        - Allow resources sharing
        - Simpler to implement, and lower cost
- What to design a good protocol? Any performance metric
    - **Throughput(吞吐量)**--How much data you can send in certain time? like speed/velocity(bit/sec) = Traffic load(bits)/Time(s). Like watching video in Youtube, Netflix.
    - **Latency/Delay**-how long does it take a bit to send from one end to the other?
    - Reliability--no data loss
    - Security--no hacking
    - Privacy--GDPR
    - Censorship--not under government's monitor
    - Eco-friendly
    - Inexpensive
    - Easy to upgrade or reconfigure

- 1/10, Lecture 3 (Delay, throughout, reliability)
    - **Performance**
        - **Throughput** ("Fluid model") --The amount of data you can pump to link at certain time, in unit bps(bit/sec).
            - Concept: **Bottleneck transmission rate**, measures the number of useful rate(bits/sec) delivered at the receiver, and is different from but related to the individual link data rates
            - Instantaneous: rate at given point in time
            - Average: rate over longer period of time
            - **Bottleneck link**: The throughput of a transfer is limited by the link with the slowest throughput along the path - bottleneck link. (you cannot pump data faster than the rate of the slowest link)
        - **Delay** (four source: nodal processing节点处理, queueing delay, propagation delay)
            - **Queueing delay(**排队延迟): Size of packets(bits)/throughtput(bit/sec), time waiting at output link for transmission, depends on congestion level of router
                - (c) (2 points) What is the queuing delay of this link if there is queue of 10 packets, created due to cross traffic? Assume 1000 bit packets as before.
                - $d_{queue} = \frac{N}{a} = \frac{10 \times 1000}{1 \times 10^7 \text{ bit/sec}} = \frac{1 \times 10^4 \text{ bits}}{1 \times 10^7 \text{ bit/sec}} = 1 \times 10^{-3} \text{ sec}$
            - **Processing delay(**处理延迟): Time taking for checking bit error, determine at output link
            - **Transmission delay(**传输延迟**)**: L(bits)/R(bps), Time takes to send the packet, depends on packet length/size, and link bandwidth
                - (b) (2 points) What is the transmission delay of the link, i.e., the time between when the first and last bits of a packet are sent? Assume a packet is 1000 bits.
                - $d_{trans} = \frac{L}{R} = \frac{1 \times 10^3 \text{ bits}}{1 \times 10^7 \text{ bit/sec}} = 1 \times 10^{-4} \text{ sec}$
            - **Propagation delay(**传播延迟): d(m)/s(m/s).That is, the bits have to propagate/transmit at the speed of waves/light in the transmission medium to reach the other end. This delay depends on the length of the wire, and is usually only significant for long distance links. Ex: So a radio wave takes 1 microsec for a distance of 300 metres. Speed of light in copper is around 2*10^8 metres. So it takes 10 nanosec to travel a 2 meter long wire.

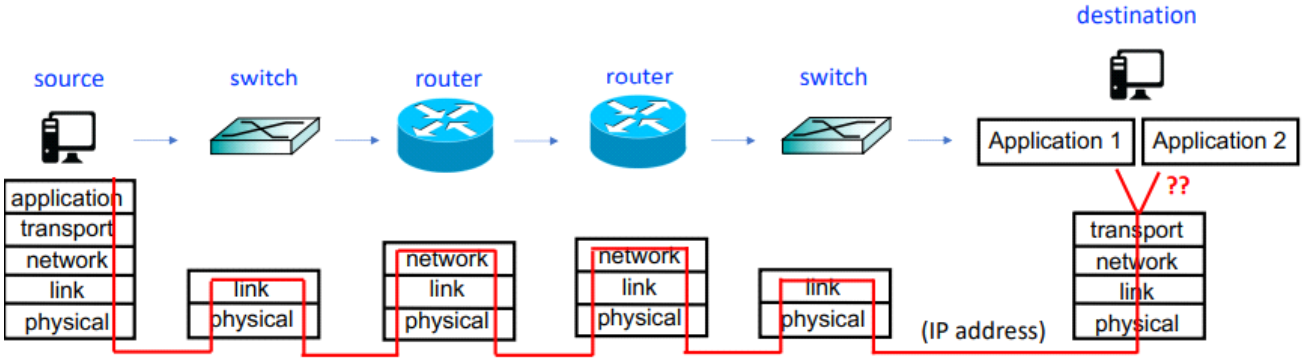


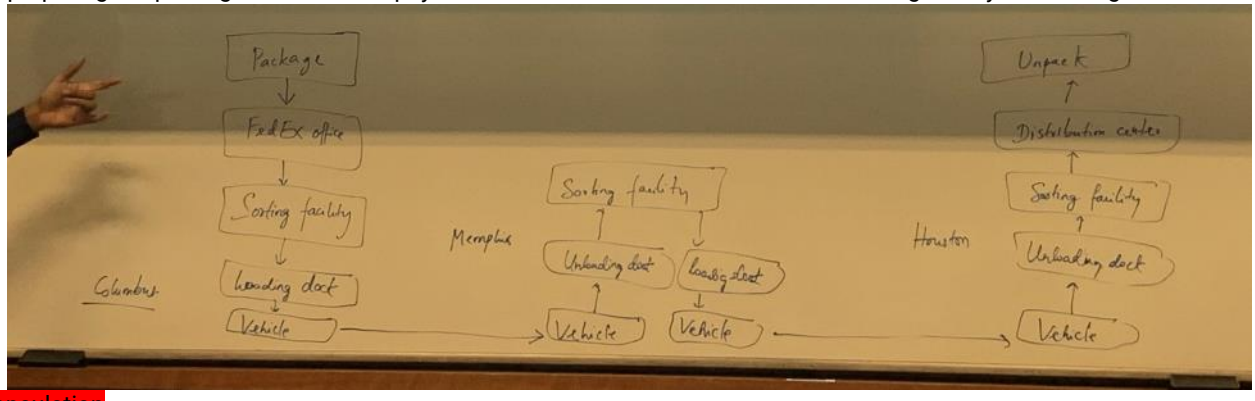
$$d_{hop} = d_{queue} + d_{proc} + d_{trans} + d_{prop}$$

            - **Hop:** "Hop" is just a regular english word (meaning "to jump"), and not an acronym. In networking, hop refers to how many "jumps" a packet has to make before reaching its destination. In our case, each time a packet encounters a router it is considered as a hop. The hop delay I was referring to in class, is basically the delay the packet experiences between one hop to the next hop. https://en.wikipedia.org/wiki/Hop_(networking)
        - **Reliability**:
            - Buffer size has limited capacity

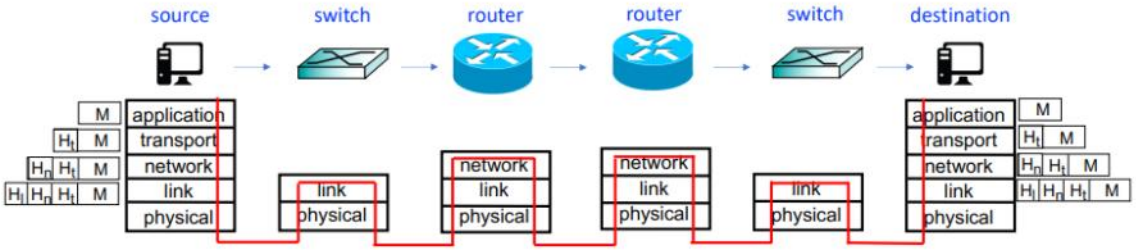
        - Packet may be dropped if buffer is full
        - The lost packet may be retransmitted
        - Packet might take different path and might arrive out-of-order

- 1/13 Lectures 4:
    - The internet protocol(IP)
        - **Routing**: process of determining the end-to-end path from source to destination
            - Analogy: (CA)(Menlo park)(Hacker way)(1) <==> 127.0.10.100
        - The responsibility of IP protocal: Router follow a routing algorithm for forwarding packet, and each packet includes "address" of its destination
    - IP Performance: provide no guarantees
    - End-to-end principle
        - Connecting several existing network by using routers
        - Function of routes: forward data from one network to the next
    - Layering in the Internet
        - Original goal for the internet: design for general usage.
        - **Application**: programmed in the host, support network application
        - **Transport**: provide realizable data transfer/communication
        - **Routing/Network layer:** provide source-to-destination end systems communication, e.g. connecting the existing network, receive/send data between network, deciding routing path.
        - **Link layer:** provide the reliable transfer/communication between neighboring network nodes, e.g. router and end-system
        - **Physical layer:** provide bitwise signal communication, e.g. converting the bits to voltage or EM wave for signal transmission
    - Why do we want this layering structure design?
        - Easy development for complex application
        - Easy maintenance, and system

- 1/15 Lecture 5: (Encapsulation, responsibility and application example for each layer)
- //Chapter 2 Network application
    - **Addressing process on Hosts**: There may have multiple process/application running on Host.
        - Solution: includet both IP address and a process Identifier(aka Port number) in packet
        - 

    - Analogy of OSI in Delivering system: transferring a large amount of data from LA to boston
        - preparing the package --> Fillout the paybill --> Send to FedEx office --> Send to sorting facility --> Loading dock -->
        - 

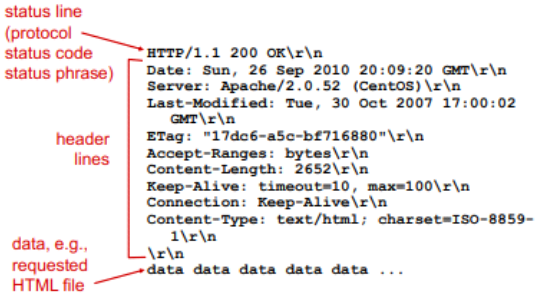
    - Encapsulation



- It's a symmetrical delivering procedure
- M is the payload of segment, stands for message
  - **payload**: the actually data we want to send (not include any header, or meta data)
- H_t, H_n, H_i are the header for each layer
- header tell the information about the packet, like ID, meta information
- link --> physical: converting bits data to certain signal or wave

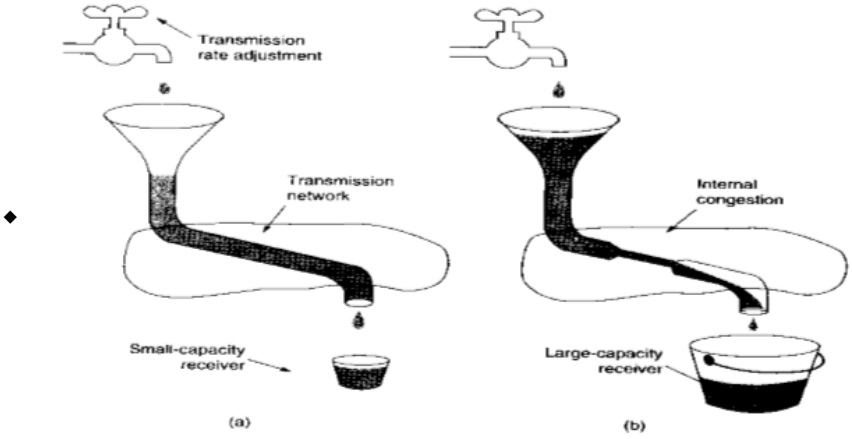
- **Network application**
  - Primary responsibility:
    - implement the software on different end system
    - Communicate over the network
  - Three key questions to ask (When creating a new application):
    - What functions and communication operations must the application perform? E.g E-mail, Web app, Ride-sharing?
    - What type of transport layer services does it requires? E.g. TCP/UDP?
    - How can we provide those transport layer services to the application? e.g. Sockey API
  - Application layer protocol: HTTP
    - **Example of HTTP request/response message**

HTTP request message



HTTP response message

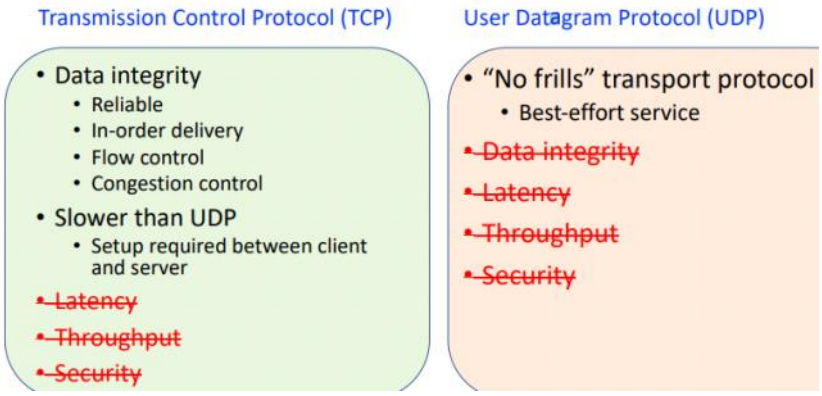
- 1/17, Lecture6: (network application layer, TCP vs UDP); 1/22, Lectuer7: (Socket API)
  - Network Application Architecture
    - **Client-Server**
      - Server: Always on, has permanent IP address
      - Clients: communicate with server whenever needed
      - Concept: Applications are typically structured as cliend-server applications. When you browse the web, your browser is the "client" and the web server is the "server". The server is always on, waiting for clients, Clients approach the server when needed. Server are maintained in data center or server farms.
    - **Peer-to-peer**
      - Not always-on server, each end-system directly communicate with each other
      - Scalability: New peers/user bring new service capacity
      - Advantage:
        - Scalability: less load on server. New peers can bring new service to network, as well as demand
        - faster download: as the number of client increase,
        - Anonymity
        - More robust
      - Some terminology: "peer churn", churn rate, BitTorrent

        - Two reason for how file sharing system work: a spawn of user has desire on single file, or a user have the entire file and want to help others
- Transport layer service
    - Data integrity
        - no loss of data
    - Latency
        - low delay
            - Elastic traffic and Inelastic traffic
    - Throughput
        - large throughput (amount of data allow to transfer)
    - Security
        - Good Encryption
- Transport layer protocal
    - Two types for two transport services:
        - **TCP(Transmission control protocol)**: TCP provides connection-oriented reliability for in-order delivery of a byte stream, important for application like E-mail, Terminal Access, Web and file transfer, which need reliable data transfer.
            - Connection-oriented: a connection oriented protocol, use Three-Handshake connection
            - Reliability: reliable delivery of data, but slower in speed
            - Flow control: sender/receiver speed matching
            - 

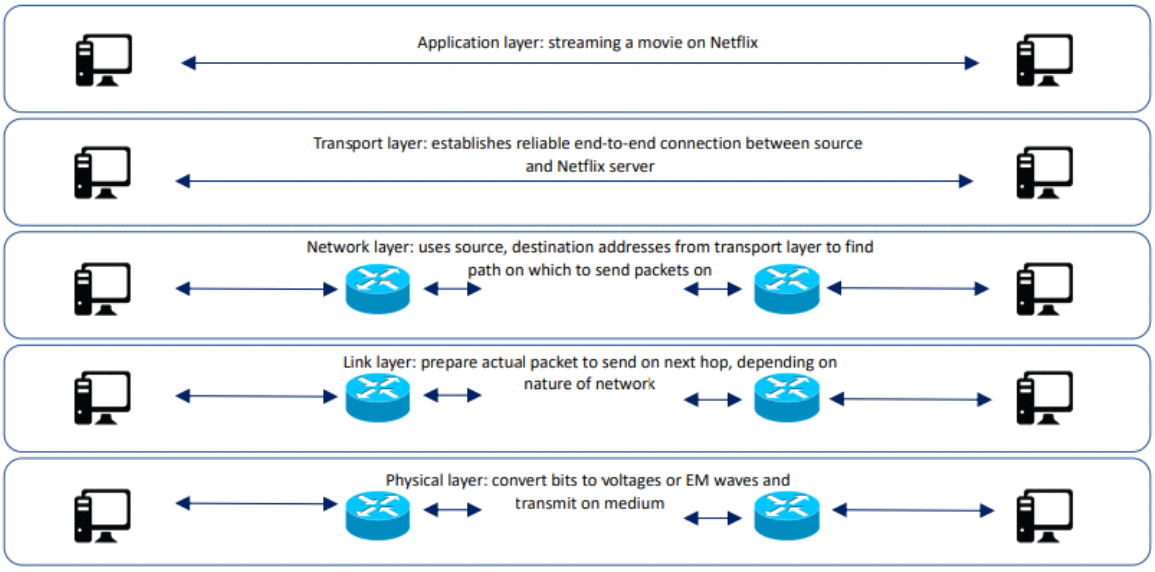
            - Congestion-control mechanism: breaks long messages into shorter segments, so that a source throttles its transmission rate when the network is congested.
        - **UDP (User datagram protocol)**: UDP just delivers message without any reliability promises, no flow control, no congestion control USP is preffered by some real time application, such as Internet telephony, Real-time video conferencing, audio/video streaming(视频数据流), which can tolerate some loss .
            - Connectionless protocol (无连接协议): No connection need to build for data transferring
            - No reliability, no security. However, small overhead(header size), low latency(No congestion control), so good for real-time communication software
    - Real world example:
        -

Application	Application layer protocol	Underlying transport protocol
E-mail	Simple Mail Transfer Protocol (SMTP)	TCP
Web	Hypertext Transfer Protocol (HTTP)	TCP
File transfer	File Transfer Protocol (FTP)	TCP
Streaming multimedia	HTTP (e.g., YouTube), Real-time Transport Protocol (RTP)	TCP UDP
Internet telephony	Session Initiation Protocol (SIP), RTP, proprietary (e.g., Skype)	TCP or UDP

    - Comparing TCP Vs UDP

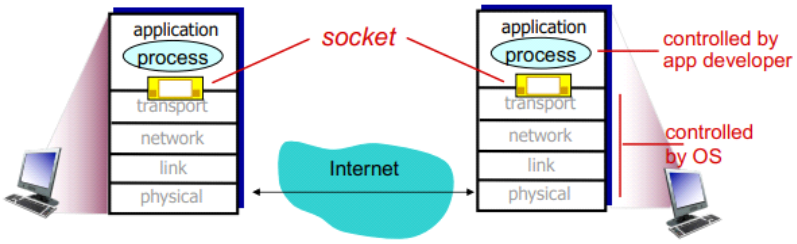


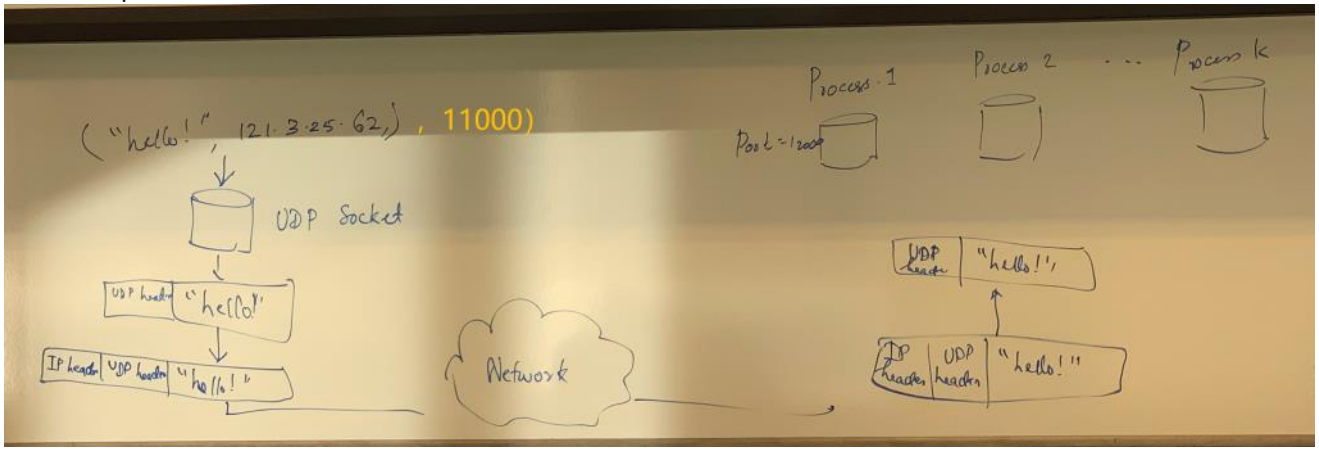
- HW01
  - List five layers of Internet
    - What are the five layers in the Internet Protocol stack?
      - Application layer
      - Transport layer
      - Routing/Network layer
      - Linking layer
      - Physical layer
    - What are the principal responsibilities of each of these layers?
      - Application layer: Implement the software in the end-system
      - Transport layer: provide reliable data transfer
      - Network/Routing layer: decide the best path to send the packets
      - Linking layer: provide reliability in sending packet to next "hop" (from one router/endhost to the next, along the path found by network/routing layer)
      - Physical layer: converting bit data to voltage or EM wave



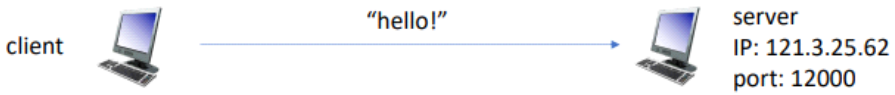
    - Give an example of a protocol, application, or link technology at each layer?
      - Application layer: HTTP(Run on TCP), SMTP, FTP, Telnet, DNS(run on UDP)
        - HTTP is for web application, SMTP is for email transfer, FTP is for file transfer, telnet is outdated cliend-server protocol.
      - Transport layer: TCP, UDP
      - Network/Routing layer: IP
      - Linking layer: Wifi, ethernet
      - Physical layer: radio tower, fiber optic, modem, Electromagnetic wave

- Socket API
  - Concept of socket: Application developers use socket interface to send and receive "message". (e.g., client sends server a message requesting a web page). Think of sockets as post boxes, and application layer delivers the message (e.g. GET or POST) from mail boxes. The transport layer handles how the message is delivered, much like how the postal system handles how mail is delivered.
  - Socket programming
    - Purpose: learn how to build client/server application that communicate using sockets

- Socket: door between application process and end-end-transport protocol
- Analogy: Socket == Door
- 

- **Addressing processes on Hosts**
  - Motivation: If several sockets open on a machine, how do we know which socket a message is destined to? (Note: IP address gives the information of which machine should be deliver to, but doesn't help beyond that.) ==> Answer: Port numbers.
  - **IP address:** for identifying which machine(End host) the packet should be send to. (32 bit for IPV4, and 128 bit for IPV6)
  - **Port number**: 16-bit identifier that uniquely identifies the several open sockets on a machine. (because there can be multiple process running on an computer).
    - Also specifies the socket address
    - Common port number: HTTP--80, mail--25
      - why they have certain port number? ==> Used pretty often
    - Size of port number: 16 bit
    - In server side, there can be multiple client want to reach it.
    - Binding: bind the port number to UDP socket. so the message that has port number 12000 will be delivered to specific socket in the server.
- **Socket programming with UDP**
  - UDP: provide unreliable transfer of groups of bytes("datagrams") between client and server
    - No handshaking before sending data
    - Sender attaches IP address and port # to each packet
    - Rcvr extracts sender IP addresses at port# from received packet
  - Visualization of procedure:
    - 

  - UDP Client: Create socket ==> attach (ip address and port number) to each encoded packet ==> send to server ==> and close it
    - Overall process

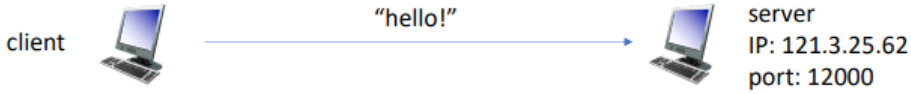


Python UDP Client

- include Python's socket library → `from socket import *`
- create UDP socket → `clientSocket = socket(AF_INET, SOCK_DGRAM)`
- attach server name, port to message; send into socket → `clientSocket.sendto("hello!".encode(), ("121.3.25.62", 12000))`
- close socket → `clientSocket.close()`

- Detail explanation:
  - create UDP socket → `clientSocket = socket(AF_INET, SOCK_DGRAM)`
    - get_sock comment,
    - AF_INET, create the socket for over communication. ip refer ,
    - SOCK_DGRAM: specifies we are using UPD protocol
  - attach server name, port to message; send into socket → `clientSocket.sendto("hello!".encode(), ("121.3.25.62", 12000))`
    - first is the message that you want to send, must convert to binary code
    - Second is the server ip and port number

- UDP Server: create socket ==> bind port number to socket ==> recv message and decode it
  - Overall operation



Python UDP Server

- `from socket import *`
- create UDP socket → `serverSocket = socket(AF_INET, SOCK_DGRAM)`
- bind socket to local port number 12000 → `serverSocket.bind(("", 12000))`
- read from UDP socket into message, getting client's address (IP and port) → `message, clientAddress = serverSocket.recvfrom(2048)` `message = message.decode()`
  - Client reads a line of characters (data) from its keyboard and sends the data to the server.
  - The server receives the data and converts characters to uppercase.
  - The server sends the modified data to the client
  - The client receives the modified data and displays the line on its screen.
  - Note: You might need a while loop to keep this process going
  - Detail explanation:
    - bind socket to local port number 12000 → `serverSocket.bind(("", 12000))`
      - binding the port number to a socket on the server
      - ip address also can bind to address, so only the source ip message will be received
    - read from UDP socket into message, getting client's address (IP and port) → `message, clientAddress = serverSocket.recvfrom(2048)` `message = message.decode()`
      - recvfrom: received message from..
      - 2048: a safe number, the size of buffer, if the message greater than 2048 byte will be truncated. If the

server is not running, the package will be lost.

- ClientAddress give us the address of client, so Server can send another message back to client as well

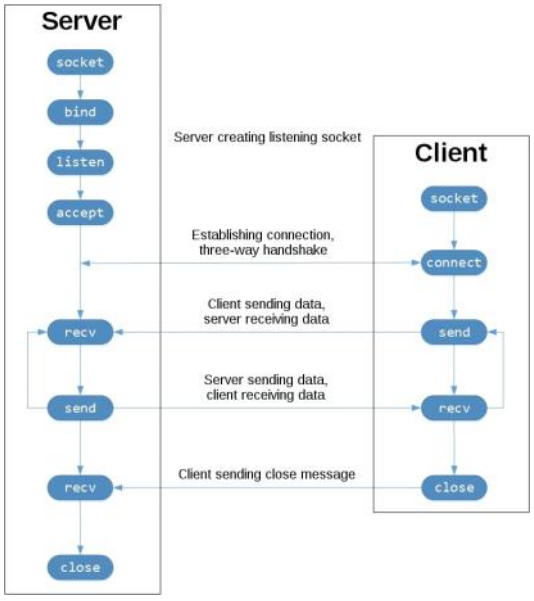
- Summary:
  - create a socket: socket(AF_INET, SOCK_DGRAM)
  - bind ip and port#: sock.bind(("121.3.25.62", 12000))
  - send message: sock.send("hello".encode())
  - recv message: sock.recvfrom(2048)
- hw2: Implementation
  - UDP_ping_client.py

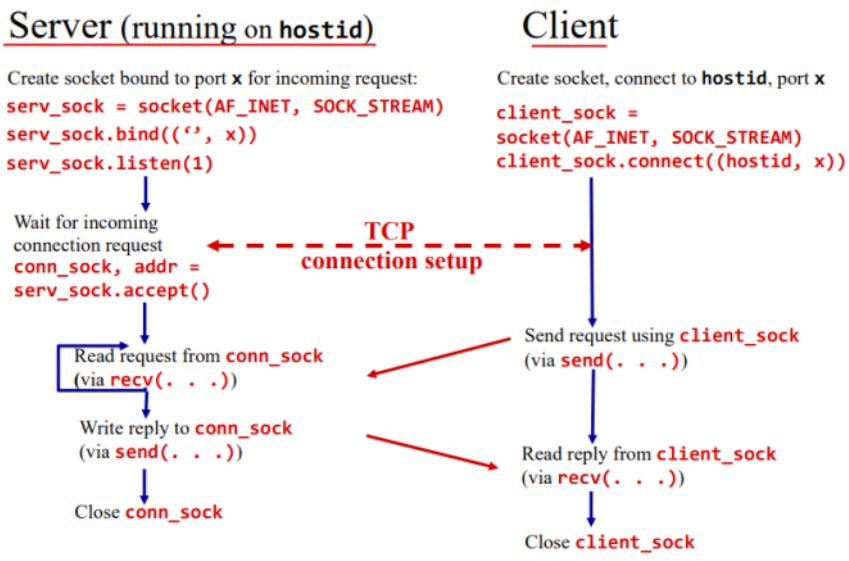
```python
serv_name = "127.0.0.1"
serv_port = 12000

# Create a UDP socket
client_socket = socket(AF_INET, SOCK_DGRAM)
# Set the timeout for 1 second
client_socket.settimeout(1)
# Send 10 pings to the server
for i in range(10):
    try:
        msg = "Ping " + str(i) + " " + time.ctime()
        # print(msg)
        # Set the timmer:
        start = time.time()
        #   Send the ping message to server
        client_socket.sendto(msg.encode(), (serv_name, serv_port))
        # Wait up to one sec for a reply; If reply is received, print the response message
        data, addr = client_socket.recvfrom(1024)
        # Stop the timer:
        stop = time.time()
        print("Packet received from "+ addr[0]+ ": " + data.decode(encoding='ascii'))
        # Look up the Python documentation for the timeout value on datagram socket
        print("Round-trip-time(RTT): "+str(stop-start)+"\n")
    except:
        # Otherwise, assume the packets lost and print "Request timed out"
        print("Request timed out.\n")
        continue
        # Proceed to the next one
client_socket.close()
```

  - UDP_ping_server.py

```python
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Assign IP address and port number to socket
serverSocket.bind(("", 12000))

while True:
    # Generate random number in the range of 0 to 10
    rand = random.randint(0, 10)
    # Receive the client packet along with the address it is coming from
    message, address = serverSocket.recvfrom(1024)
    print("THe message received is :" + message)
    # Capitalize the message from the client
    message = message.upper()
    # If rand is less is than 4, we consider the packet lost and do not respond
    if rand < 4:
        continue
    # Otherwise, the server responds
    serverSocket.sendto(message.encode(), address)
```

- **Socket Programming with TCP**
  - TCP Socket Flow



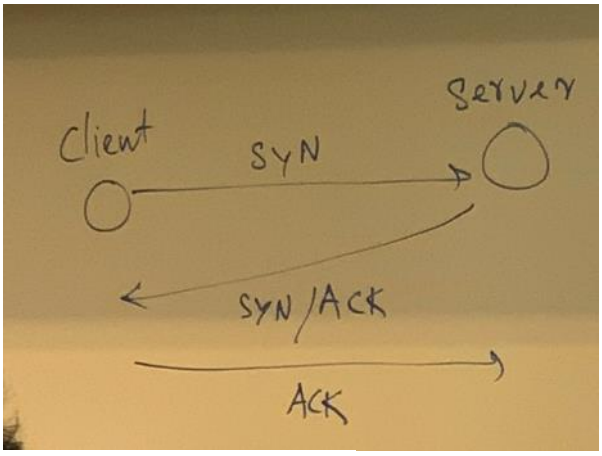


- TCP Client: Create socket(SOCK_STREAM) ==> make connection to server("121.3.25.62", 12000) ==> Send encoded message to server ==> Close it
  - Overall Operation

    Python TCP Client

    ```
    from socket import *
    clientSocket = socket(AF_INET, SOCK_STREAM)

    clientSocket.connect(("121.3.25.62", 12000))

    clientSocket.send("hello!".encode())
    ```

  - A detail looking
    - `clientSocket = socket(AF_INET, SOCK_STREAM)`
      - Create TCP socket in client side
    - `clientSocket.connect(("121.3.25.62", 12000))`
      - TCP is an reliable protocols, so it will need to build an TCP connection between client and server before any communication, by Three hand-shake.

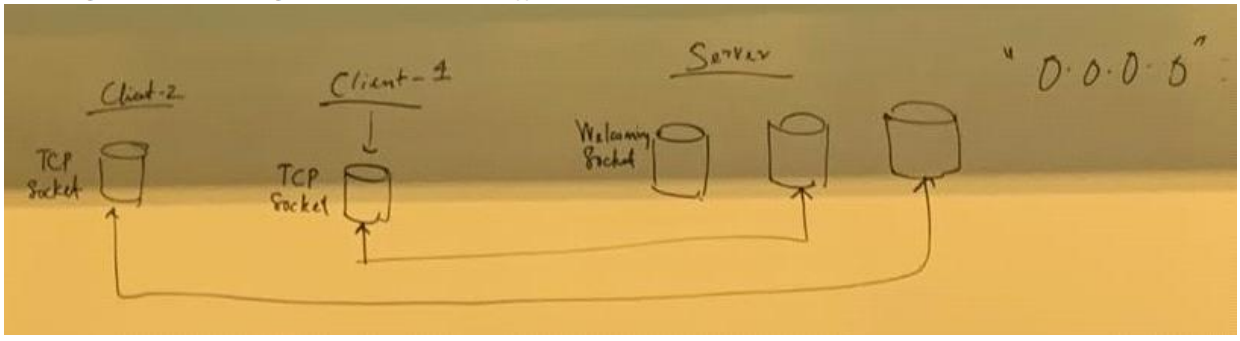


- `clientSocket.send("hello!".encode())`
  - the message that client want to send
  - writing data to TCP send buffer, and it's the responsibility of TCP to send message to TCP receive buffer.
  - ConnectionSocket.recv(1024) will receive the packet. 1024 is the maximum number of byte allow to receive, any thing beyond that will be remain in TCP receive buffer.
- `bytestream = bytestream.decode()`
  - The data is transferred in byte, so you need to convert to human language

- TCP Server: create a socket(SOCK_STREAM) ==> Bind the port number to socket(.bind(("", 12000)))==> Acknowledge the connection from client ==> recv packets from client and decode it
  - Overall Operation

Python TCP Server

```
from socket import *
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(("", 12000))
serverSocket.listen(1)
connectionSocket, addr = serverSocket.accept()

bytestream = connectionSocket.recv(1024)
bytestream = bytestream.decode()
```

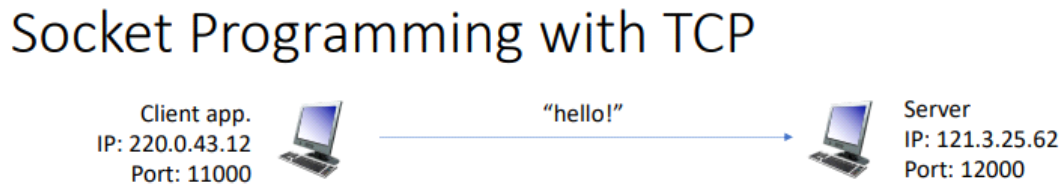


- `serverSocket = socket(AF_INET, SOCK_STREAM)`
  - Create TCP socket in server side as "welcoming socket
- `serverSocket.bind(("", 12000))`
  - binding the ip addr and port number to socket
  - By setting the first argument to empty, it allows server to receive package from multiple client. Otherwise it will only received packet from certain user
- `serverSocket.listen(1)`
  - Queueing request.
  - (1) means the maximum number allow to queue is 1 packet
- `connectionSocket, addr = serverSocket.accept()`
  - welcoming socket will return the client socket and it's ip address
  - A blocking call, if no request, it will just waiting.
  - `clientSocket.connect(("121.3.25.62", 12000))`
    - If on client attempt to connect the server, it will send an connect request. connect() is a blocking call as

well, it will wait server respond, before sending message

- `bytestream = connectionSocket.recv(1024)`
- `bytestream = bytestream.decode()`

- Byte stream
  - After building the reliable connection, TCP will pass the sequence of byte to the socket, and truncate to transport layer packages.
- problem of TCP protocol
  - You need to create a socket for each time you want to send an package
    - there can be
  - To create an "Welcoming TCp socket" at server side. So, every-time a client want to send an package to server, it send to an certain port only (let's call it welcome port). Moreover, it will create an connection socket,



Python TCP Client

```
from socket import *
clientSocket = socket(AF_INET, SOCK_STREAM)

clientSocket.connect(("121.3.25.62", 12000))

clientSocket.send("hello!".encode())

clientSocket.close()
```

Python TCP Server

```
from socket import *
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(("", 12000))
serverSocket.listen(1)
connectionSocket, addr = serverSocket.accept()

bytestream = connectionSocket.recv(1024)
bytestream = bytestream.decode()

connectionSocket.close()
serverSocket.close()
```

  - What happen if have multiple client:
    - that use listen(num_client), to specify the maximum number of client you want to take time
    - and use an while loop, and established multiple connectionSocket to the client
  - What if we have a very large data, which beyond the size of receive buffer, want to send to server, would the packet get lost?
    - No, there something called "Flow control", sending buffer would send any data that is beyond the capacity of receive buffer

- 1/24, lecture8-9: DNS name resolution

32 bit IP address (e.g., 127.11.3.20)

- IPv6 addresses 128 bits long
- Public IP addresses are reachable by anyone over the Internet

  - Analogy:
    - Problem of telephone number : hard to remember
    - Solution: use your friends name
- Q: Why do we need two name? (DNS and IP)
  - Because it's hard to remember IP address. However the router can't easily deal with English-variable names when doing packet forwarding (DNS is hard to implement lookup table, all english words, and lack of security), so IP address is better for lookup table (Because your router need to figure our where to send your data, and the way your router can figure out is to ask lookup table).
  - An **IP (Internet Protocol) address** is an address that identifies a host or router on the Internet. In version 4 of the Internet Protocol(also known as IPv4 address), a 32-bit numbers (roughly ~4B address). Version 6(aka IPv6) modifies these addresses to be 128-bit numbers.
  - **DNS(Domain name system)**: IP addresses are hard to remember. Imagine having to type https://172.34.56.123 every time you wanted to access a web site. Ideally, we would have a more memorable way to remember the web site. That's where the domain name comes in. An example of a domain name is www.cs.nyu.edu. Domain names are hierarchical: www.cs.nyu.edu belongs to the edu top-level domain (TLD), the nyu subdomain within the edu TLD, the cs subdomain within the nyu subdomain, and the www subdomain within the cs subdomain.
- URL vs DNS
  - DNS is the subset, or part of URL

- How does the DNS were translated to IP address
  - **Naïve solution:** have a file that mapped every domain name to its IP address. (So, in early day before 1984, there was a single file called HOSTS.TXT that maintained this mapping and which people swapped with each other over the internet. To add a new host, the host's owner had to call an operator at Stanford Research Institute! If you don't believe me, open up /etc/hosts on your UNIX/Mac machines. This is a remnant from the HOSTS.TXT era
  - **Another solution**: have a local centralized server for lookup., (e.g. DNS server, DNS name resolution)
    - single point of failure
    - difference from previous is, previous everyone have a "Address book", but right now, only one copy of "Address book" remain in centralized server
    - But, the problem is a lot of "overhead" need to done in centralized server, because everyone is sending "lookup request" to centralized server. So, Traffic volume, Latency are the dis-adv of this solution.
  - **Final Solution: DNS**
    - So, how does DNS work? DNS has a hierarchy of servers that mirrors the hierarchy found in domain names themselves (Figure 1). At the top of the hierarchy are the root servers. One level below are the TLD(Top-level domain) servers, one for each TLD. Below this are the authoritative servers, typically one for each organization such as NYU, Google, or Facebook. In addition, there are also local servers owned and operated by service providers that are provided as a convenience to the end users.
      - They are distributed databased, implemented in hierarchy of many name server (i.e. Local server, Root server, TLD--top-level domain servers, and authoritative servers). If local name server cannot resolve the name, it will contact the root server, and root server contact down the DNS hierarchy and returns mapping to local name server.

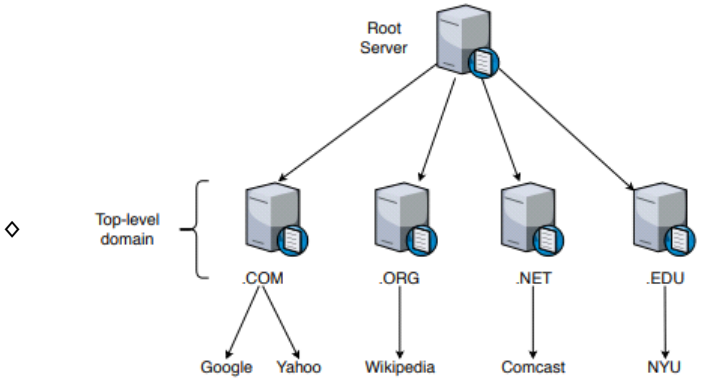
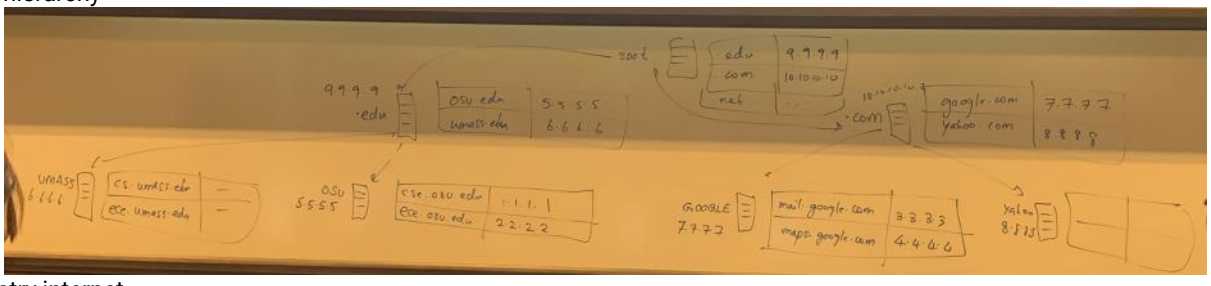
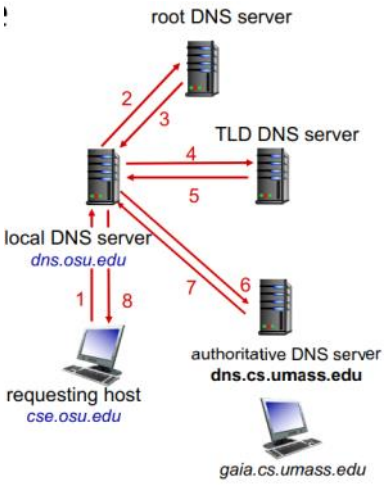


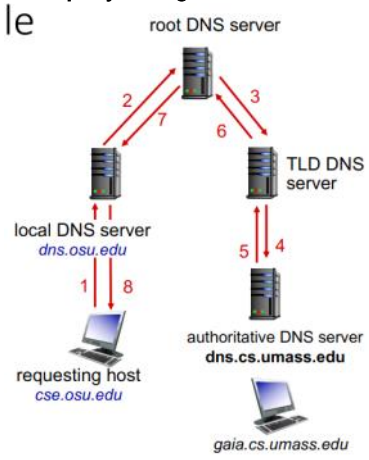
Figure 1: Hierarchy of DNS servers

    - Example: "www.wikipedia.org"
      - org: to-level domain name
      - "wikipedia": subdomain name
    - Note:
      - DNS was running at UDP not TCP
- DNS hierarchy
  - 
- Registry internet
  - New university registering process: apply for domain name(newuniversity.edu--> after approved --> add register to Top-level domain server (.edu)
  - Root server: has the registry of top-level server(e.g. .EDU, .COM.)
  - Authoritative server (e.g. mail.google.com, cse.osu.edu)
  - local-name server: e.g. Spectrum, AT&T
  - Authoritative server == Local-name server??
    - No, local-name server is like local network provide: AT&T, Spectrum..., not part of DNS Hierachy
  - is .edu, .com are called Domain name as well?
    - Yes
- **Domain Name Resolution**: Example
  - **Iteratve query: Directly contact model**



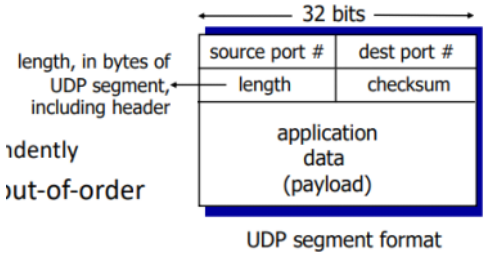
- First requesting host contact/consult the cache, or local-name server. If it doesn't have the IP address of the domain name you ask for, then it will contact the next server(e.g. Root DNS server). If Root DNS doesn't have it, **local-name server** will directly contact the next server(e.g. Top-level domain server), and so on. (Root DNS server ==> TLD DNS server ==> authoritative DNS server)

- **Recursive query: Delegation model**

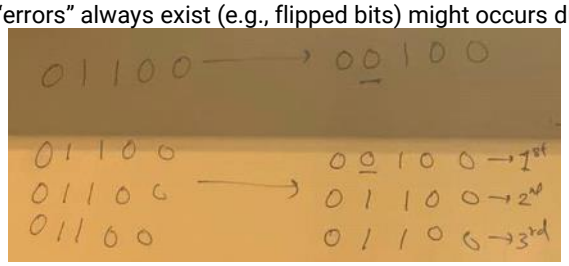
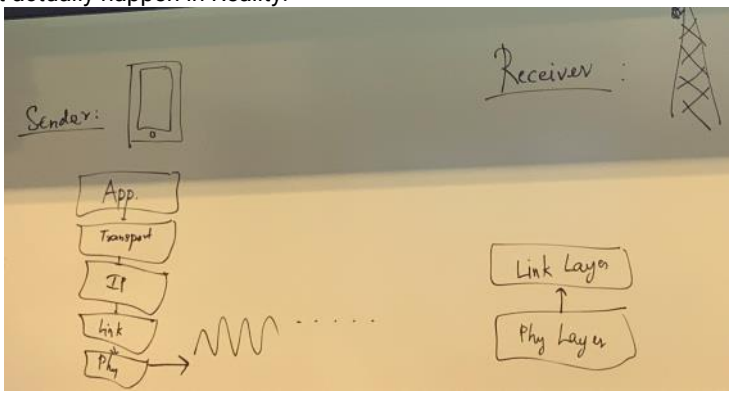
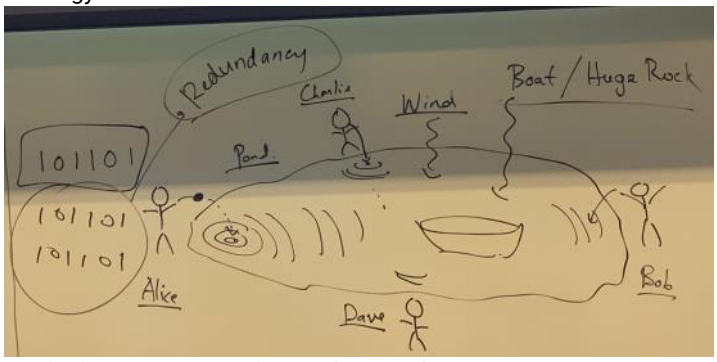


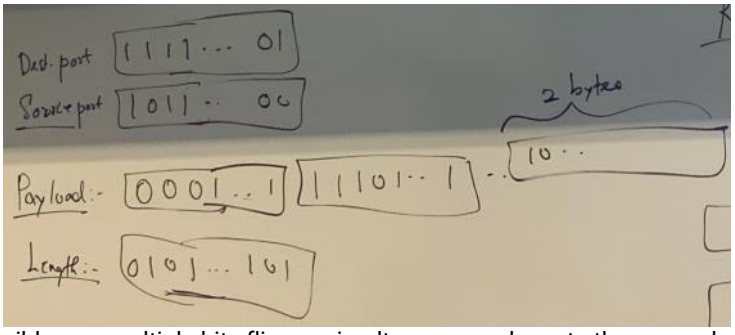
- First requesting query/consult the cache, or local-name server. If it doesn't have the IP address of the domain name you ask for, then the it will contact the next server(Root DNS server). If Root DNS doesn't have it, **Root DNS server** will be delegated to ask next server(e.g. Top-level domain server), and so on..

- Regardless of whether it was iterative(directly contact) or recursive(delegates the work to root DNS)

- 1/29: Lecture10, Transport layer with UDP protocol(Ponds analogy Vs real world example), error detection mechanism(Checksum)
  - Terminology: three terms means the same thing
    - Application layer: "message"
    - Transport layer: "segment"
    - IP/Networking layer: "datagram"
  - **UDP: User datagram protocol**
    - 

      - 8 byte in total: source port(2byte, 16 bit) + dest port(2byte, 16 bit) + length(2byte, 16 bit) + checksum(2byte, 16 bit) = 2byte * 4 = 8 byte
    - Datagrame == segment
    - payload: the actual message you want to send
    - Why do we need source/dest port# ?
      - you want the UDP receiver to know which socket to follow
    - Why do we need length?

- The length is useful, because different packets might have differ size
- Why do we need checksum?
  - Detecting “errors” (e.g., flipped bits) in transmitted segment
- What is error detection and how does it happen
  - Problem:
    - This due to the nature of physical media(transmission interference, noise, lost of signal, signal attenuation…), the “errors” always exist (e.g., flipped bits) might occurs during the transmission.
    - 
- Adv:
  - lower overhead compared to TCP
- What actually happen in Reality:
  - 
  - wireless == unguild transmission
  - wire == guilded transmission
    - More robus, but has more constrain . (e.g. you cannot send message without the wire between two device)
- Pond analogy
  - 
  - Goal: Alice want to send some bit to Bob
  - Three disturbance (noise, interference, loss of packets)
    - communication between Charlies and Dave == **Interference**
    - wind == **Noise**, happen caused by the nature of physical nature, e.g. thermal energy dissipation
    - Boat/Huge rock == modern building (Concrete) will absorb the signal, and it might cause the **loss of packets**
  - Additional issues: signal attenuation 信号衰弱
- Error detection mechanism: **Checksum segment**
  - In reality it's hard to avoid those disturbance, so we need a way, error detection mechanism, to detect the error (e.g. fliping bits)
    - Use checksum -- it's the additional segment contents
      - checksum = one's complement of sum
        - sum = Dest port + source port + Payload + Length + 1(In bit)



- But it's still possible, e.g. multiple bits flippes simultaneous and create the same checksum as the correct checksum
  - So we have second layer of error detection system, which is more stricker
  - How does the checksum was calculated:

example: add two 16-bit integers

```
                1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
                1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
              ---------------------------------
wraparound  (1) 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
              ---------------------------------
sum             1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
checksum        0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

- What is wraparound: because you have 17 bit after the addition, but only 16 bits is allowed, so we add that carry bit to the rest of 16 bits.

- Example:

Original Data

10011001	11100010	00100100	10000100
1	2	3	4

k=4, m=8

Sender

```
1   10011001
2   11100010
    ---------
   (1)01111011
        1
    ---------
    01111100
3   00100100
    ---------
    10100000
4   10000100
    ---------
   (1)00100100
        1
    ---------
Sum: 00100101
CheckSum: 11011010
```

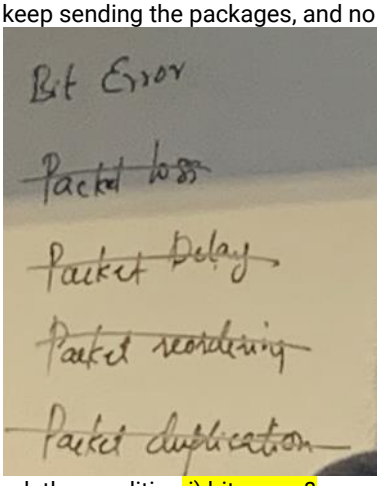
Reciever

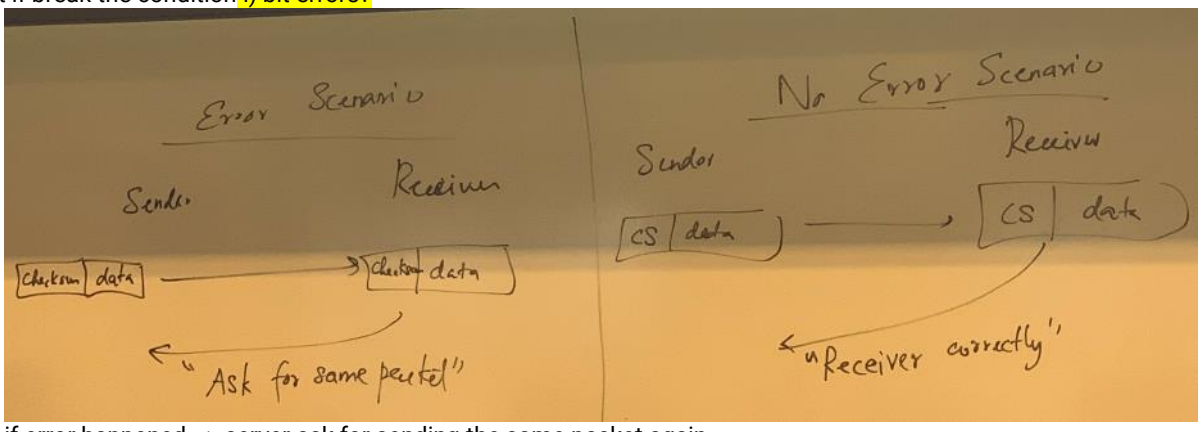
```
1   10011001
2   11100010
    ---------
   (1)01111011
        1
    ---------
    01111100
3   00100100
    ---------
    10100000
4   10000100
    ---------
   (1)00100100
        1
    ---------
    00100101
    11011010
    ---------
Sum: 11111111
Complement: 00000000
```

Conclusion: Accept Data

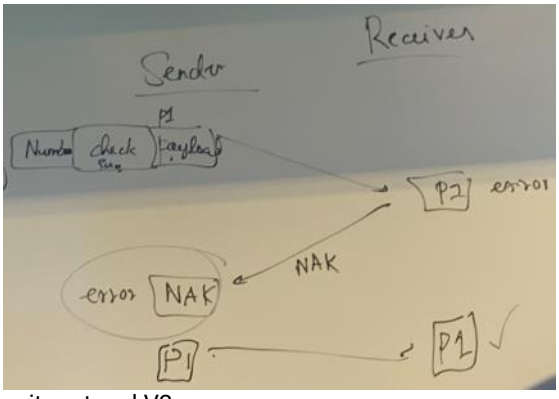
- Another solution, is Alice sending multiple message. It's like voting system, choose the signal that has the most votes.
- Are there <u>two mechanism for error detection</u> in internet communication?
  - --> Checking the sequence/packet number
  - --> using checksum to match the correct computation

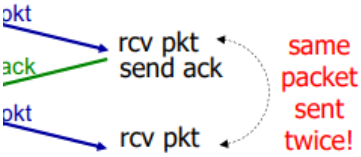
- 1/31
  - Reliable Data Transfer
    - Why is UDP unreliable?
      - bit errors
      - packet loss
      - packet delay

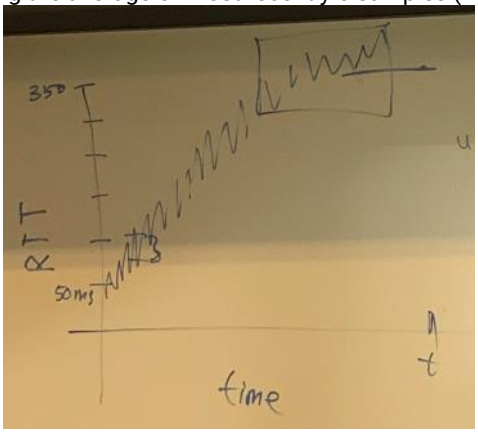
    - packet reordering:
        - differing packet might choose different route, so the time it take to travel might differ. So the original packing order might be break.
    - Packet duplication: getting multiple packet at different time
- Stop-and-wait protocol: Version 1
    - if everything is perfect, none of above 5 bad thing would happen, what does a reliable protocol would looks like?
        - Just keep sending the packages, and no any checking or connection would need
        - 

    - What if break the condition i) bit errors?
        - 

        - if error happened --> server ask for sending the same packet again
        - otherwise --> keep sending the next one
        - Note: don't keep the sender hanging
    - **Packet Reliability: Ack && Nak protocal**
        - Ack: acknowledge
        - Nack: non-acknowledge
        - Receiver:
            - If no error, send Ack
            - if errors, send Nak
        - Sender:
            - if Ack comes back, send next packet
            - If Nak comes back, send the same packet
    - **It's two way communication, so what if the Ack send by receiver was get corrupted?**
        - There are two possible scenarios, 1) The Ack get corrupted -- Sender send same packet again, the problem of duplication would occur. 2)The Nak get corrupted-->sender send same packet agains, so works fine, the sender still need to send the same packet again!

- 

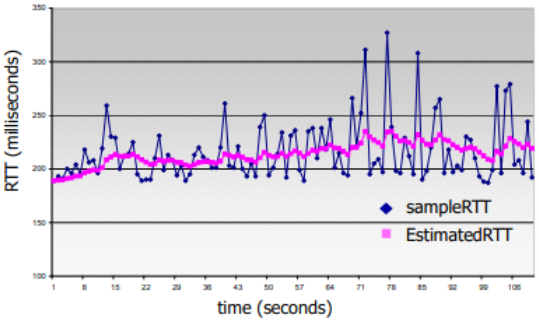
- Stop-and-wait protocol V2
- Problem of duplicated packet (receiver get confused: "why are you sending me the same packets again?")
  - Solution is including a packet number: adding an ID (packet number) for each packet, so receiver would know if two are the same if two packet has identical packet number, and receiver can discard the duplication!
  - 

- Stop-and-wait protocol V3
  - What if break the condition ii) no packet loss?
    - Such as the ack that receiver send get loss? What would happen?
      - Sender would keep waiting
      - So we might want to setup timeout mechanism
    - So, how long we should wait? --> retransmission timer
      - Too short: premature timeout, unnecessary retransmissions
      - Too long: slow reaction to packet loss
      - **Retransmissions timer**: We can do an estimation, such the avg of traveling time of previous packets, if longer than 95% CI, we can just assume the packet get lost!
      - **RTT**(round-trip time): the time for packet travel from sender to receiver, and from receiver back to sender.
    - naïve solution:
      - Taking the average of most recently 5 samples (measured RTT time)
        - 

SampleRTT: measured time from segment transmission until ACK receipt

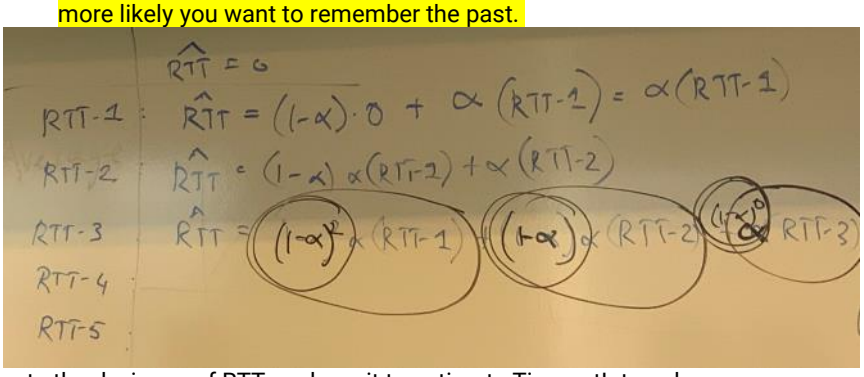
SampleRTT will vary, want a "smoother" estimate

Solution: average over several recent measurements



    - Better solu:
      - Exponential weighted moving average (EWMA): Taking the last 5(x) samples, the most recent packets,

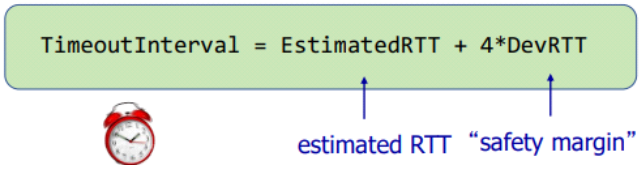
and to take the average on them. So, we discard the old data, and only look at the recent packet.

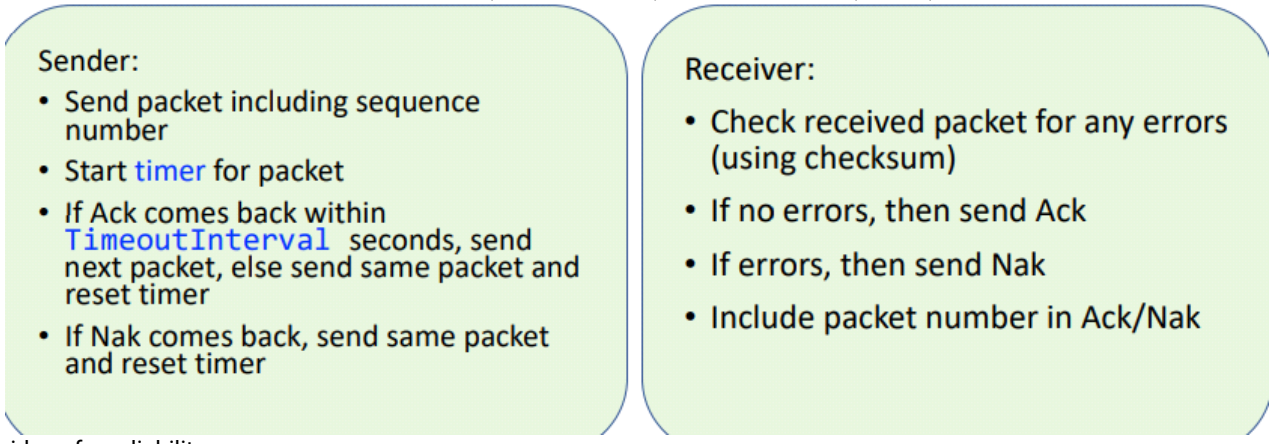
- $\texttt{EstimatedRTT} = (1-\alpha)\texttt{EstimatedRTT} + \alpha(\texttt{SampleRTT})$
- Why it's called EWMA?
  - Because The fact that the weights decrease exponentially
  - alpha takes effect on current RTT, and (1 - alpha) takes effect on the history of sample RTT.
  - The larger the alpha, the more likely you want to forget the past. The smaller the alpha, the more likely you want to remember the past.
- 

- Compute the deviance of RTT, and use it to estimate TimeoutInterval
  - **TimeoutInterval** estimated the time sender should wait before retransmission happens

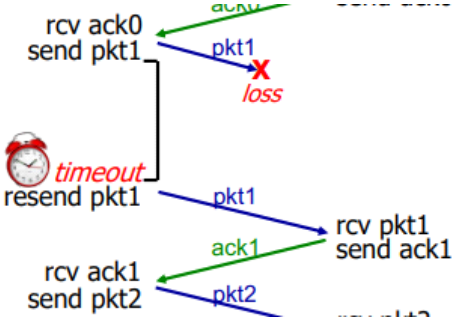
    Estimate SampleRTT deviation from EstimatedRTT:

    $\texttt{DevRTT} = (1-\beta)\texttt{DevRTT} + \beta|\texttt{SampleRTT} - \texttt{EstimatedRTT}|$ (typically, $\beta = 0.25$)

  - 

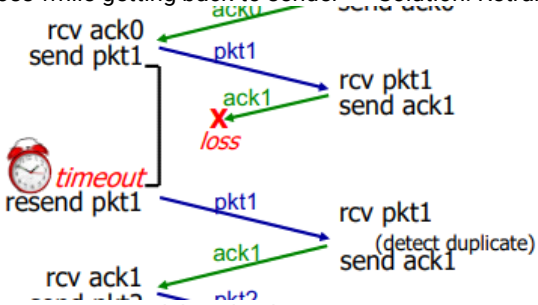
- Stop-and-wait protocol: Final version
  - Keyword: packet number, Ack/Nck, error detection(with checksum), Timeout-Interval(EWMA)
  - 

  - main ideas for reliability
    - Ack/Nak
    - Packet number
    - Retransmission timer
  - Three scenarios
    - Packet loss while sending to receiver --> solution: retransmission timeout

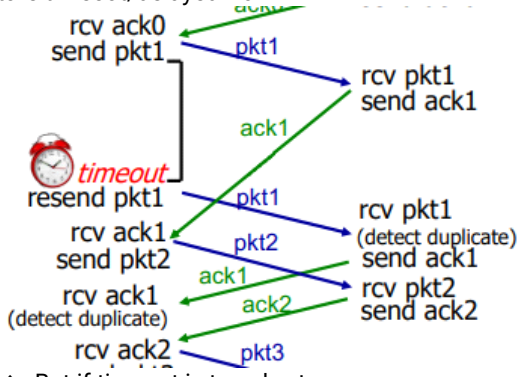


- If the pkt1 that sender sends get lost, receiver won't receive it, and then for a period of time, sender will sent it again

□ Ack loss while getting back to sender --> Solution: Retransmission timeout

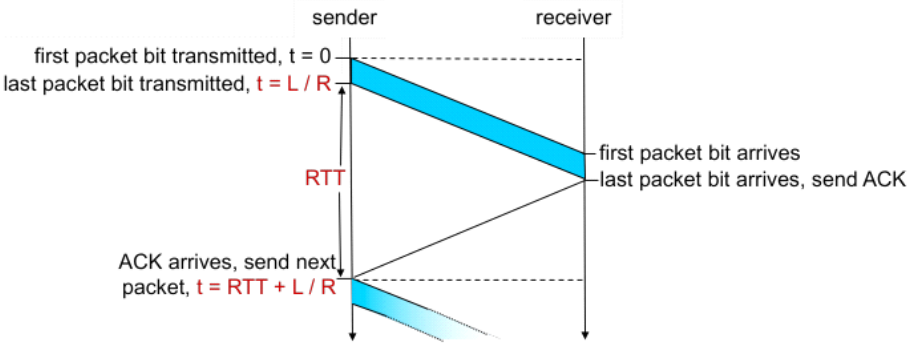


□ Premature timeout/delayed Ack



◇ But if timeout is too short,

- How to measure the performance
  - **Throughput:** total length of packet(bit) / total amount of time had taken(sec)
  - **Utilization**: transmission delay(in sec) / (Round trip time + transmission delay)(in sec)



L = 8000 bit packet

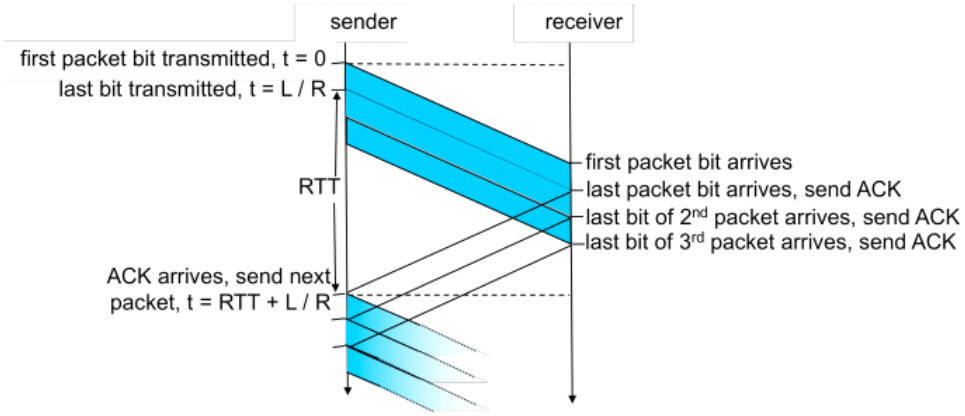
R = 1 Gbps link

RTT = 30 ms

$$\text{Utilization} = \frac{\frac{L}{R}}{\frac{L}{R} + RTT} = 0.027\ \%$$

$$\text{Throughput} = \frac{L}{\frac{L}{R} + RTT} = 266\ kbps$$

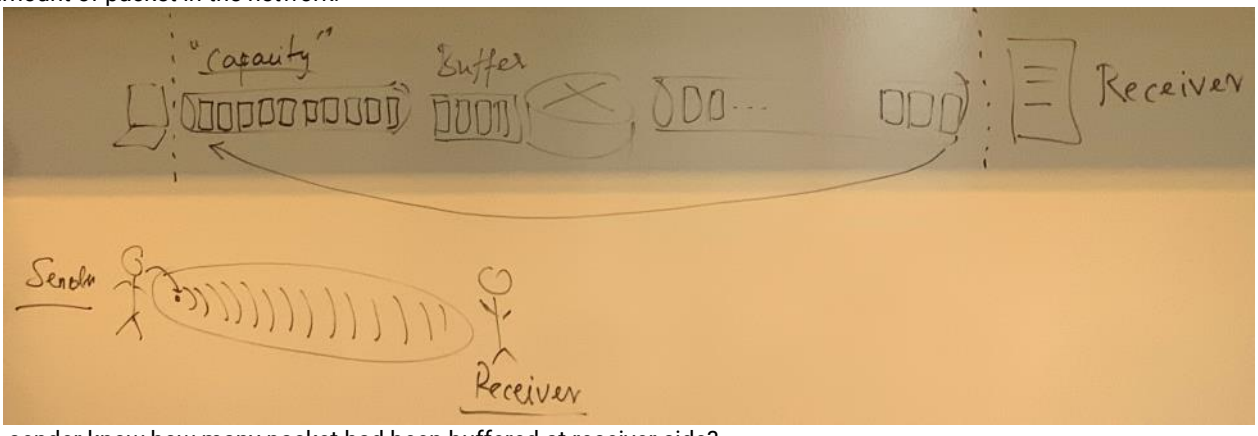
- Anyway to improve the performance?
  - **Pipelining:** Sending the other packet before the Ack/Nak being received



L = 8000 bit packet
R = 1 Gbps link
RTT = 30 ms

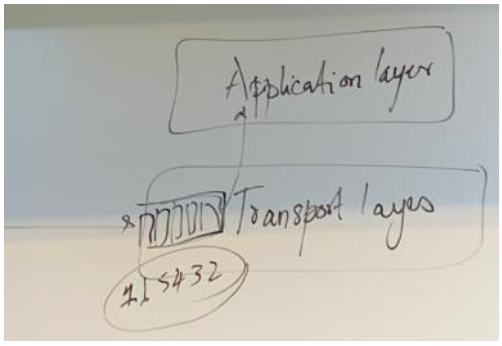
$$\text{Utilization} = \frac{\frac{3L}{R}}{\frac{L}{R} + RTT} = 0.081\ \%$$

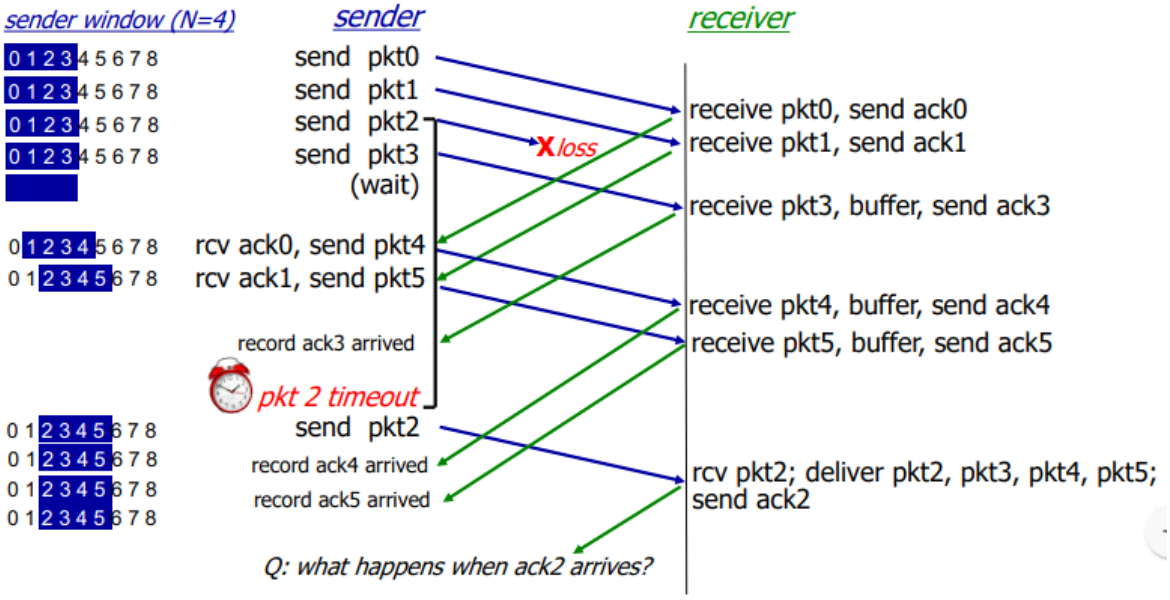
$$\text{Throughput} = \frac{L}{\frac{L}{R} + RTT} = 800\ kbps$$

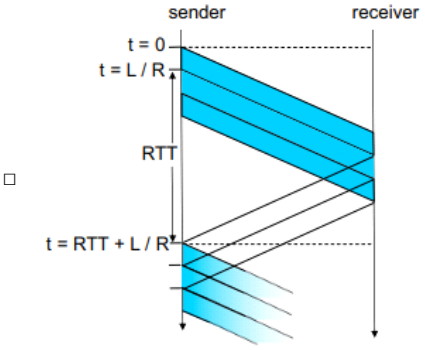
- So, is good to keep sending more packet?
  - No, if your first packet get loss, there is higher chance your next packet would get loss as well!
- So, what would be a good number of packet we should send?
  - A pond has a limited amount of "ripple" it can hold. Similarly, transmission link also has limited for the number of packet you can send, and it's same for the buffer link. So, if you send more packet the link can hold, the excessive packet would get drop.
- So, how does sender know the capacity of network?
  - It doesn't know, but can be estimated
- Even sender know the capacity of network, how does it can ensure it doesn't send too many packet than the network can support?
  - Idea: Assume the sender know the capacity of network. We will use a counter to figure out how many packet we can send. Send one packet, increment the counter. Receive a ACK, decrement the counter. So, the counter tells use the amount of packet in the network!
  - 

- <u>How does sender know how many packet had been buffered at receiver side?</u>
  - In worse scenario, the number of packet need to be buffered = counter - number of packet get "un-ack"
  - In best scenario, zero number of packet would need to be buffered

- Window:
  - **window size:** <u>the number of packet sender allow to send within a RTT round.</u>
  - you slide the window until the first packet received, so all the packet had arrived, and can be process/reassemble to original orders!
  - Assume knowing the capacity, and number of packet in receiver buffered ==> chosse the min of these two

- 

- 2/7: Lecture 14(Window protocal, Congestion control)
  - Review four way to ensure the reliability for TCP
    - ACK/Nak
    - Sequence Number
    - Retransmission timer
    - Sliding window
  - Window protocal cont…
    - Sender: if there are unsued seq 3 available in window, send packet
      - 

        - slide window will move to pkg6. All the 3,4, and 5 had received, so when pkg arrived, the slide window is ready to move.
  - Performance of sliding window:
    - 

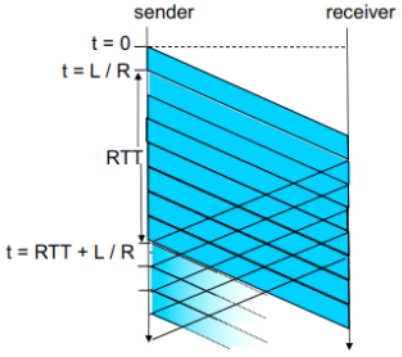
    - Utilization = Amount of time being occupied / total time
      - $$\text{Utilization} = \frac{\frac{WL}{R}}{\frac{L}{R} + RTT} \cong \frac{WL}{R * RTT}$$
      - W = window side
      - L is the length of packet, bit
      - R is the transmission rate. Aka, link capacity in unit bit/sec

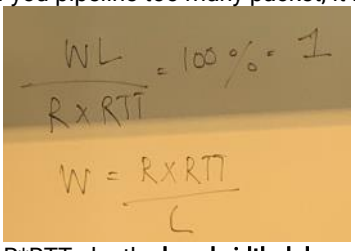
- L/R = transmission delay, in sec
- RTT = Round trip-delay time, in sec. latency from a host A to host B and back to host A. This is simply twice the one-way latency of going from host A to host B, if the forward and return paths are symmetric
- R*RTT = data link's capacity(in bits/sec) * RTT(in sec), maximum amount of data user allowed to send. Aka **bandwidth delay product**
  - It's equivalent to the maximum amount of data you can send in sec. Practically R*RTT is way larger than L/R, so we can approximate it with R*RTT
  - Ex:

    Moderate speed satellite network: 512 kbit/s, 900 ms round-trip time (RTT)

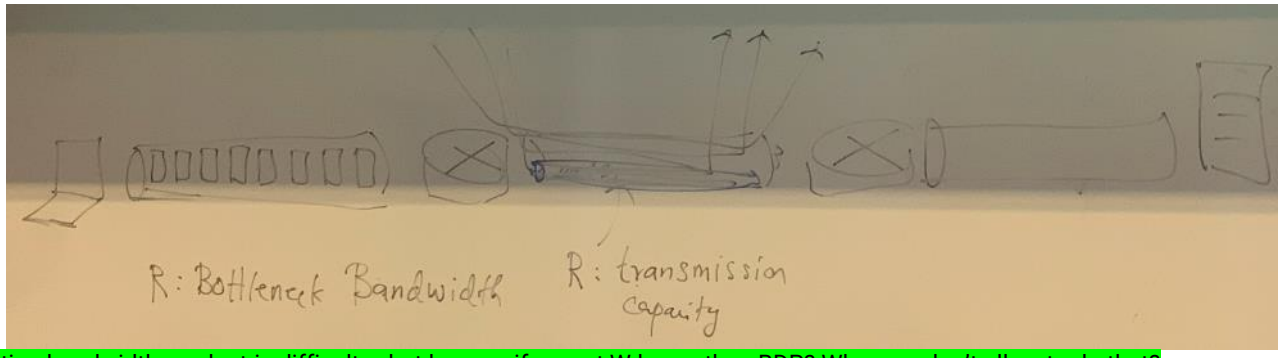
    - $B \times D = 512 \times 10^3 \text{ b/s} \cdot 900 \times 10^{-3} \text{ s}$
      $= 460,800 \text{ b} = 460.8 \text{ kb} = 57.6 \text{ kB}$

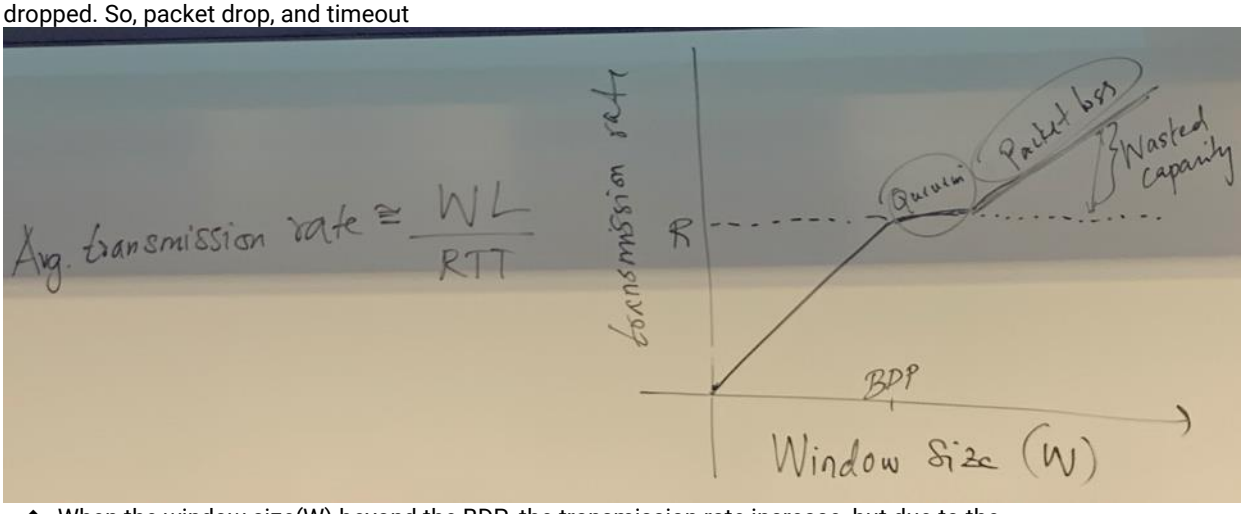
- Throughput吞吐量: == how fast the sender is sending, so the windows size is proportional to the throughput
  - $\text{Throughput} = \frac{WL}{\frac{L}{R} + RTT} \cong \frac{WL}{RTT}$
- Can we just keep increasing W(window size) for greater utilization and throuput? If not, what is the right window size?
  - Utilization cannot be more than 100%
  - $\text{Utilization} = \frac{\frac{WL}{R}}{\frac{L}{R} + RTT} \cong \frac{WL}{R * RTT}$

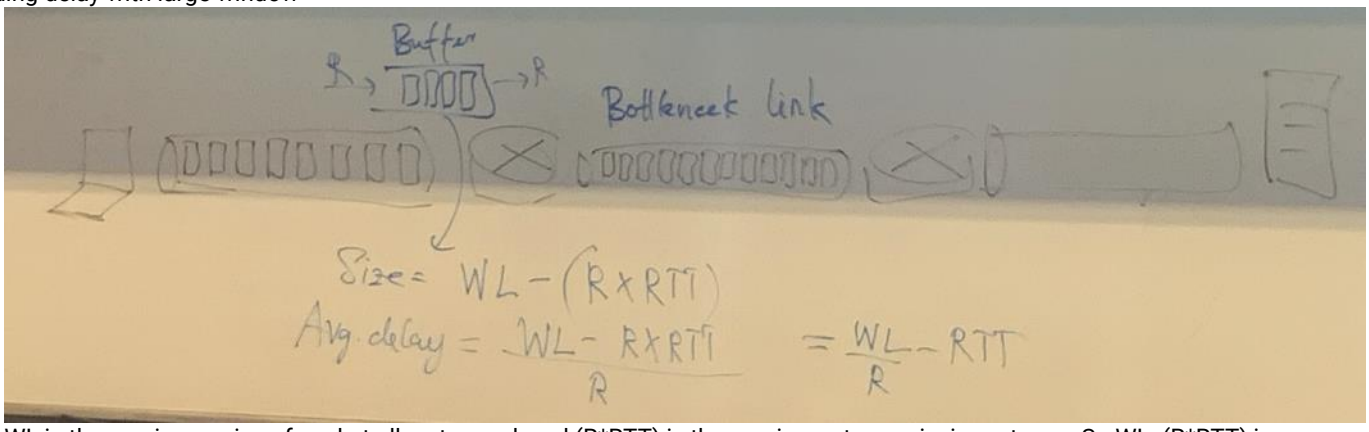


  - Therefore, $W \leq \frac{R*RTT}{L}$ → bandwidth-delay product
  - No, if you pipeline too many packet, it might cause heavy congestion. Therefore the maximum utilization is:
    - 

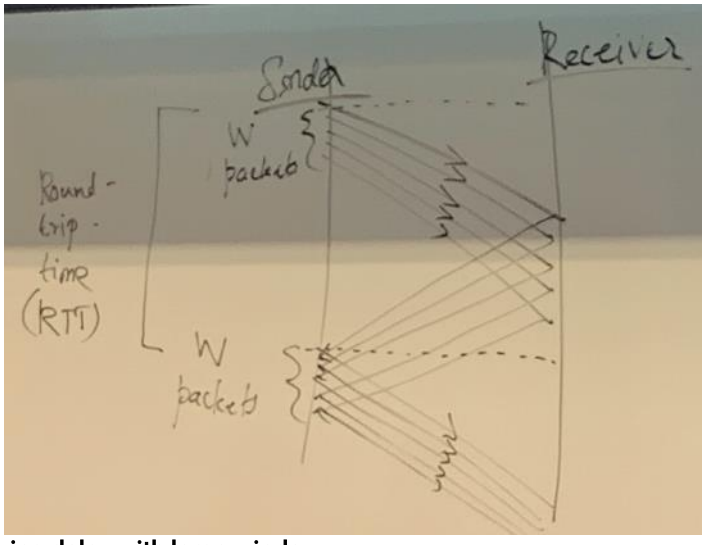
    - R*RTT aka the **bandwidth delay product**
    - Why do we let the utilization to 1
- Q: Why is estimating the BDP is hard?
  - Because you only know the bandwidth at your Access network(Router), and you had no way of knowing the bandwidth/transmission rate at others router.
    - Bandwidth delay product(带宽时延乘积) depends on R and RTT. R is the **bottleneck transmission rate**(The smallest bandwidth/capacity of link at the transmission path, which changes over time. RTT depend on whether the sender or a receiver is on a WiFi, Ethernet, or 4G link. Each link technology has different propagation delay. Also, RTT can change because of the changes in network path between sender and the receiver (e.g. A router was taken down for maintenance). Thus, neither R nor RTT is known up front, and setting the sliding window size to BDP is impractical.
    - The R here refers to the Bottleneck bandwidth(The smallest bandwidth/capacity of link at the transmission path). Ex, in Youtube server, there can be million of user in using these application, so if the Bandwidth at server size may be the bottleneck bandwidth.

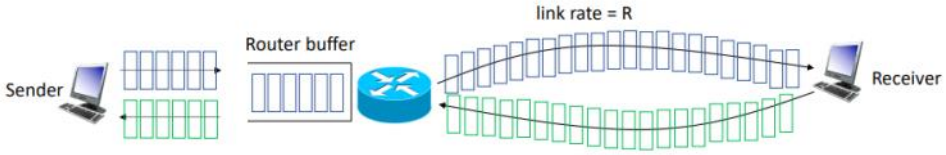
    - 

  - Q: Estimating bandwidth-product is difficult, what happen if we set W larger than BDP? When we don't allow to do that?
    - It may cause buffer overflow, and packets would get dropped, and then retransmission would happen.
      - If the window size beyond BDP, it will worse the delay, and cause a negative feedback loop. Queue has finite numbr of buffer, if you send excessive packet, and it will cause buffer overflow, and the packet might get dropped. So, packet drop, and timeout
      - 

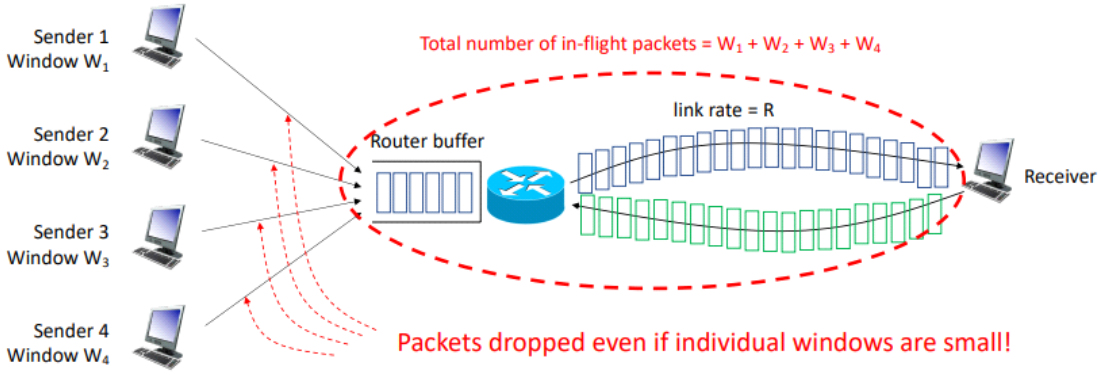
        - When the window size(W) beyond the BDP, the transmission rate increase, but due to the retransmission(associated with timeout interval), more and more packet get loss and transmission rate get wasted!
- Queuing delay with large window
  - 

  - WL is the maximum size of packet allow to send, and (R*RTT) is the maximum transmission rate ==> So WL- (R*RTT) is the maximum queuing size
    - ==> Average Queuing delay = (WL - RTT)/R <==> think of t = s/v, queuing delay is like the average waiting time
      - **queuing size** == number of packets in the buffer
      - **WL** == maximum number of packets allow to send
      - **R*RTT** == maximum number of packet live in Bottleneck link
    - ==> So larger the window size, worse the queuing delay
  - Analogy of physic:
    - queuing size, WL, R*RTT == distance
    - transmission rate (R) == velocity
    - RTT, queuing delay == time
- Retransmission of packets:
  - Ex: If queue size grows, and this time, if you upload a very large file to DropBox, most likely packets will be dropped and/or packet timer will trigger, which leading to retransmission of packets

- 2/10-2/12
  - Review:
    - 

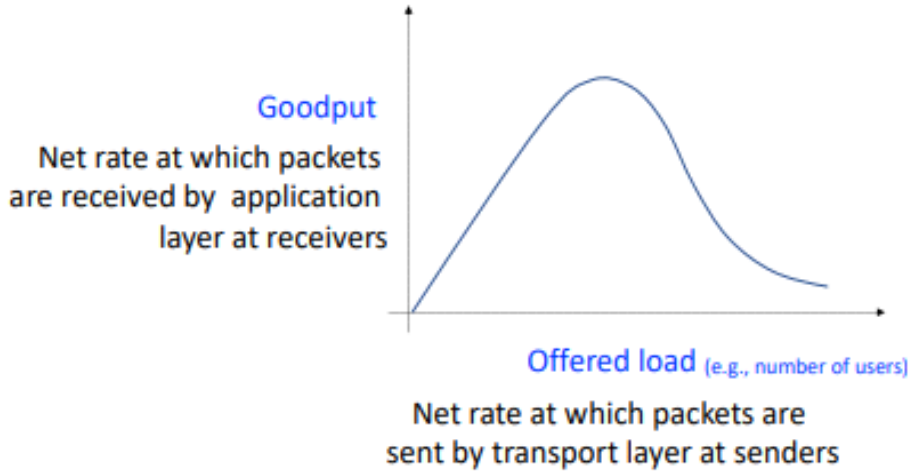
  - **Queuing delay with large window**
    - If WL > bandwidth-delay product, packets would start queuing
    - 

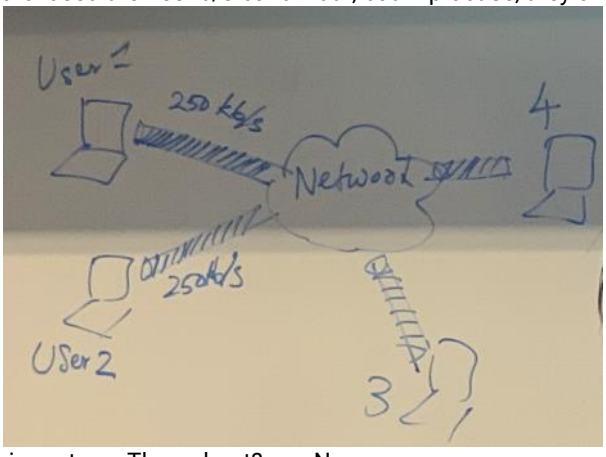
    - Note: RTT here refers to round-trip-time without any queueing
    - Queue size = $WL - (R * RTT)$ bits
    - Queuing delay = $\frac{WL}{R} - RTT$ seconds
      - (Little's law)

      Making window size too large worsens delay!
    - So the throughtput is determiend by the window size, but the windows size cannot beyond the limit of bandwidth, so if you want a fast network speed, not only you should increase the bandwidth but also the server side, because here, we are talking able the bottleneck bandwidth
    - Why is the throughput determine by L(length of packet)?
      - Throughtput is determined by the WL/RTT, Rtt is fixed, as well as the packet size(L)
    - Is tranmission rate equivalent to Network speed? ==>
      - The speed of network is determined by the bottleneck bandwidth
      - Transmission rate is determined by the transmission media, e.g. optic fiber, or the speed of electromagnetic signal wave
  - If queue ize grows, ackets will get dropped and/or packet timers will trigger, which leads to the retransmission of packets:
    - 

- Congestion collapse
  - **Congestion collapse** is a term for a situation in which the **offered load** (i.e. the demand for the network's services) is

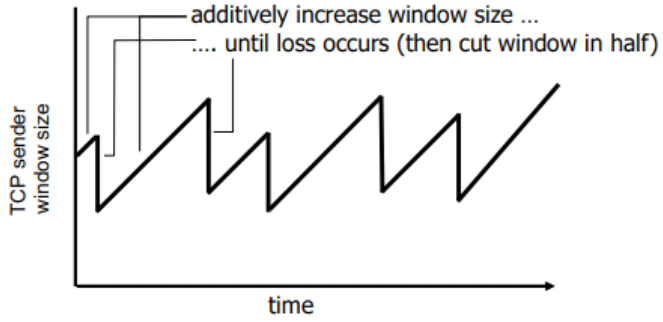
increasing, but the **Goodput** (overall utility of the network) to its users is decreasing. Possibly it's caused by a single user picked a very large W or there are too many users sharing a link, and the packets get dropped, and send again, and so on. As a result, they just keep sending multiple copies of packet, and the transmission caused the queueing delay to grow further, and lead to even more retrainsmission, which soon leads to even more queues.

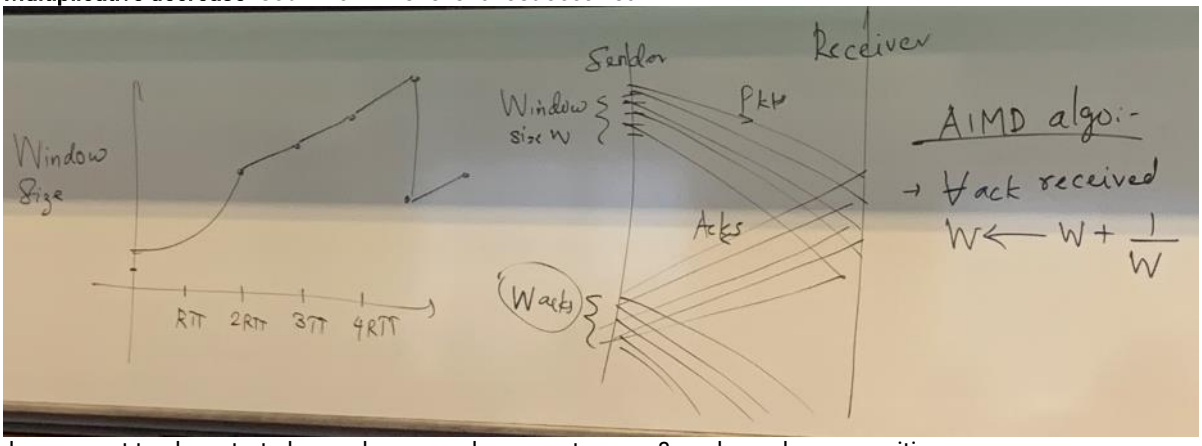


    - Offered load is depends on the number of packet were send from sender
    - goodput is the rate of useful packet being received by receiver
    - At peak, as the number of user(workload) increase, Due to the limit of bandwidth? queuing size, the applicable transmission capacity decrease, so the pack drop and retransmission happened.
  - Ex: User purchased the 250kb/s bandwidth, but in practise, they only gain the 1/10 of bandwith that they paid
    - 

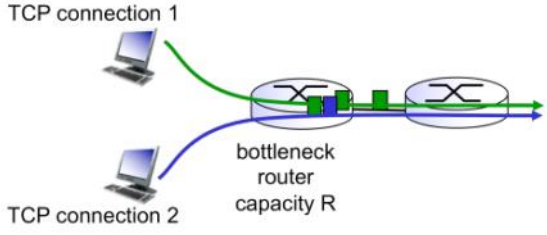
  - Transmission rate == Throughput? ==> No
    - 每个link上的transmission rate都会不同, 但是Throughput值得就是一个具体的值, that is the bottlenect transmission rate
- [Congestion control:](#) How to Find "Right" window size?
  - Two phase of congestion control: Slow start, and Congestion avoidance
    - **Slow start:** similar to accelerating phase, to get the right speed on the highway.
      - Initialize the window size to 1, and double window size for each ACK received. Whenever a packet loss detected, it cuts the windows size down to half. After this, it enters a mode called congestion avoidance, where much more gentle changes to the window size.
      - Initial rate(==1) is slow but ramps up exponentially
    - **Congestion Avoidance**: similar to maintaining this speed in highway
      - We use **AIMD (additive-increase multiplicative-decrease)** algorithm to adjusted window size dynamically. It allows a sliding window sender to adaptively find a window size equal to the BDP of the network. -- 上升的话每次加一，下降的话直接减半。

- AIMD saw tooth behavior: probing for bandwidth

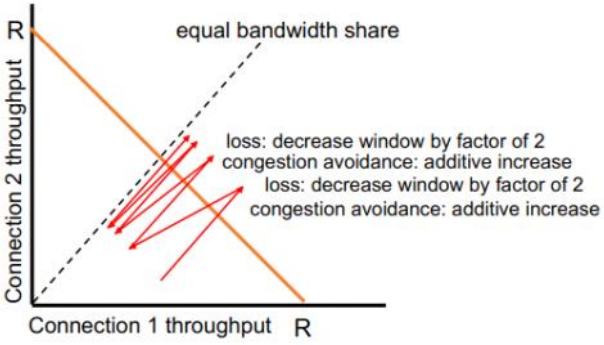


  - Concept: In Congestion avoidance, the sender increases its window by 1 every RTT, as opposed to 1 every ACK in Slow Start. This means the window size increases additively by a constant amount. It decreases its window in the same way as Slow Start, by halving the window. In other words, the decrease is multiplicative.
  - Q: Why do we call it AIMD
    - **Additive increase:** increase windom size by 1 for every RTT
    - **Multiplicative decrease**: Cut in half whenever a lost occurred
    - 

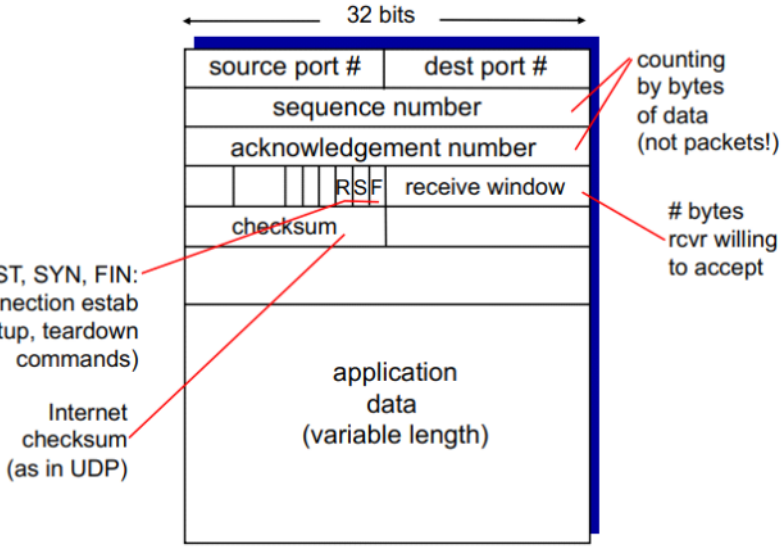
    - Q: Do we always reset to slow start phase whenever a loss event occurs? --> depends on severities
      - There are many variation of TCP (depends of your Internet Provide, e.g. ATT), and they might have different strategies in defining the severity of loss/congestion
      - e.g. timeout error occurred, tons of packet get loss --> more serious loss --> reset to slow start phase
      - e.g. several duplicated packet error occurred, few packet get lost --> mild loss ==> Don't care
      - when TCP received multiple ACK, it knows it's gonna be a serious congestion, so it's more likely will be reset to slow start , if just one packet probably not.

- TCP fairness
  - flows: an TCP connection
    - Ex: if there are two TCP connection, fairly speaking, the capacity rate of two flow should be split evenly
    - 

  - Example of two competing connection: Conn 1 might connect at first before Conn 2, so the Conn 1 might has a larger window size than Conn 2, but TCP would adjusted them evenly and gradually.
    - The dashed line is equally splied situation



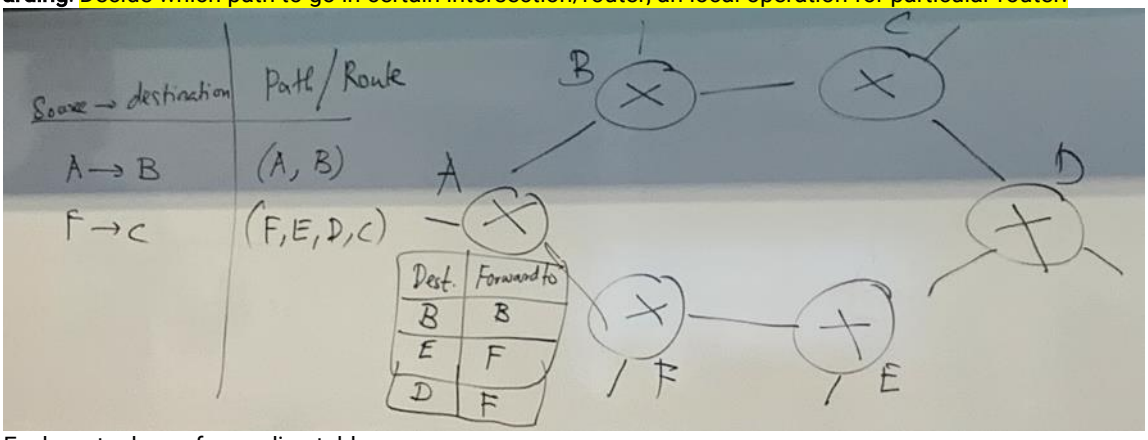
- TCP segment structure

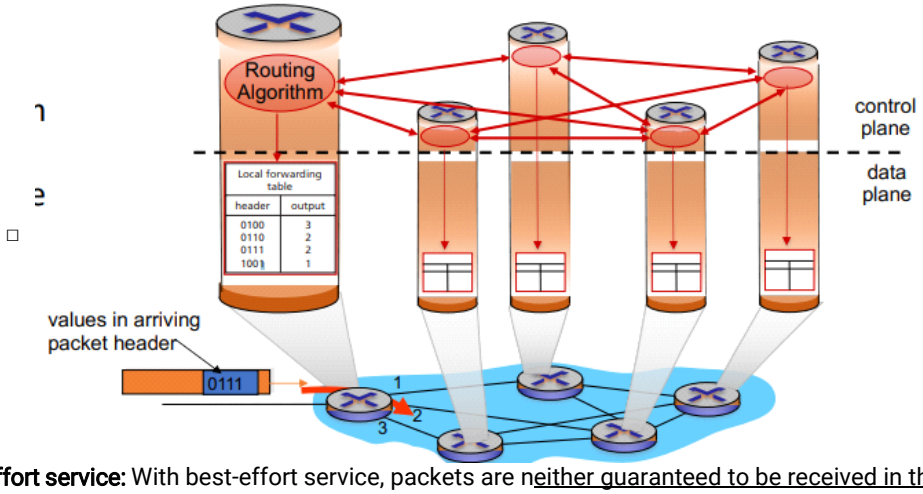


  - source/dest port: the port number of sender/receiver
  - Flags:
    - RST: reset
    - FIN: finished connection?
    - sequence number were decided by the number of byte of data, so they might have difference sequence number at the beginning.
    - SYN: both have a sequence number, and they are initialize randomly, we want to know the staring number of you packet, and you want to know the starting sequence number of my packet as well, so SYN could tell us
  - Receive window: # of byte rcvr is willing to accept
- **Summary:**
  - Five methods to provide reliable TCP connection
    - Ack/Nak protocal communication
    - **Sequence Number** == packet number
    - Retransmission
    - Window sliding--> to have both utilization and realiability
    - Congestion control
  - When to use UDP? and When to use TCP?
    - if the reliability is not seriously important, and a fast transmission is desired--> UDP, and skype,
    - If you want a strong reliability --> TCP, but might suffer from congestion, e.g. Browsing website
      - Netflix, Facebook, website, almost everything is using TCP,
      - example of buffering/loading when you watching youtube
- Resource:
  - nyu, Lecture 6: Congestion collapse.
  - Bandwidth-delay product, https://en.wikipedia.org/wiki/Bandwidth-delay_product

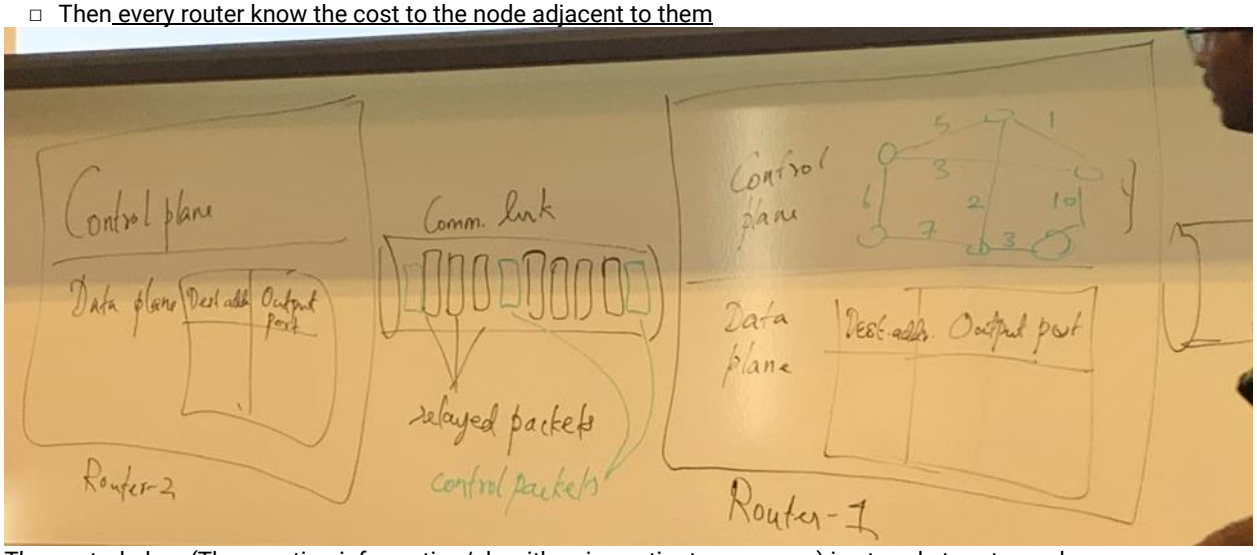
02/17 Lecture16, Routing layer

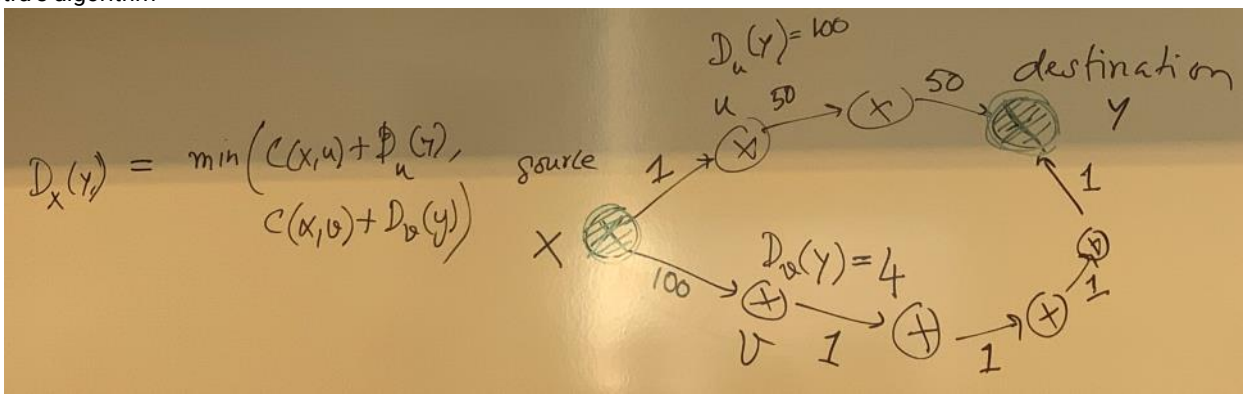
- Terminology:
  - Sending side: encapsulate transport layer segments into IP datagrams
  - **Routers**: examine header of IP datagram and to determine the route a packet should forward
  - Receiving side: deliver transport layer segment from network layer to transport layer
  - **"transport layer segments"** is defined as the packets in Transport layer
  - **"datagram"** is defined as the packet in Network layer

- **Routing**: Decide the overall route taken by packet from source to destination
- **Forwarding**: Decide which path to go in certain intersection/router, an local operation for particular router.
  - 

    - Each router has a forwarding table:
    - Network operator: like ATT, the networking decide what the path the packet to go
      - In the forwarding table at router A, if we know the path of packet (A->B), so the next router to forward is B
      - In the forwarding table at router A, if we know the path of packet (A-->E), so the next router to forward is F
      - In the forwarding table at router A, if we know the path of packet (A-->D), so the next router to forward is F
  - What is the difference between routing and forwarding?
    - "Forwarding" is about moving a packet from a router's input link to the appropriate output link. "Routing" is about determining the end-to-routes between sources and destinations.
  - Control plane, Data plane
    - **Control plane:** determine how datagrame is routed from Source host to Dest host (E.g. Network-wide logic, routing algorithm)
    - **Data plane**: determine how datagrame is routed from input port to outport port (e.g. Local router function, Forwarding functions)
    - 

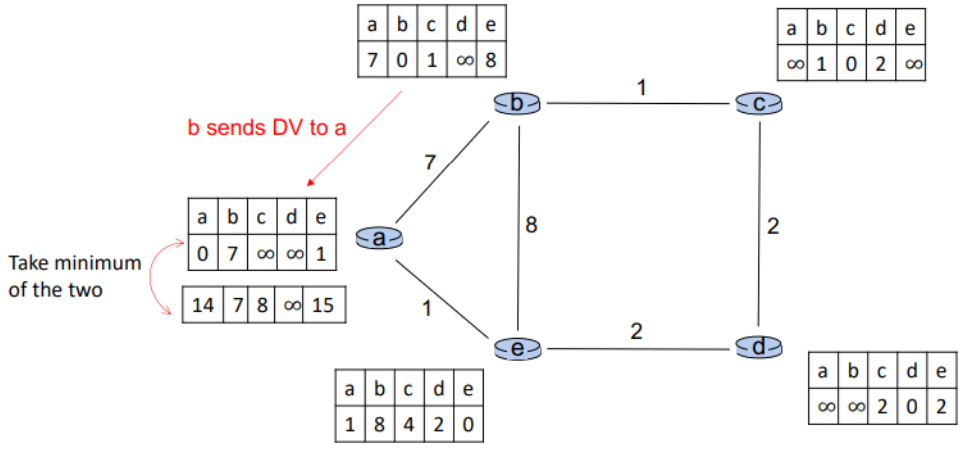
  - **Best-effort service:** With best-effort service, packets are neither guaranteed to be received in the order in which they were sent, nor is their eventual delivery even guaranteed. There is no guarantee on the end-to-end delay nor is there a minimal bandwidth guarantee.
- Performance of routing
  - Cost ==> Cheap
  - Speed ==> fast
  - congestion severity ==> mild
- Graph abstraction of the network:
  - Cost of path ==> Larger cost == small bandwidth == severe congestion
  - Question: How to find the least-cost path between two routers?
    - e.g. Dijkstra's algorithm
- Routing algorithm classification
  - **static:** routes don't change or changes slowly over time
  - **Dynamic**: routes change quickly (Don't cover in this course)
- **Link state protocol**
  - Both LS and DV are decentralized algorithm (distributed to each router in the path, instead of making decision by one)
  - How does router know the cost of other router?
    - Link state advertisement
      - Need to have the information of entire map to run the Dijksta's algorithm
      - determine the cost function (e.g. propagation delay, link capacity, and etc)

- Then every router know the cost to the node adjacent to them
- 

  - The control plane(The rouoting information/algorithm, i.e. entire topogy map) is stored at router and communicated through communication link.
- Dijkstra's algorithm
- 

  - What is difference from Bellman-Ford equation?
  - $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$ for each node $y \in N$

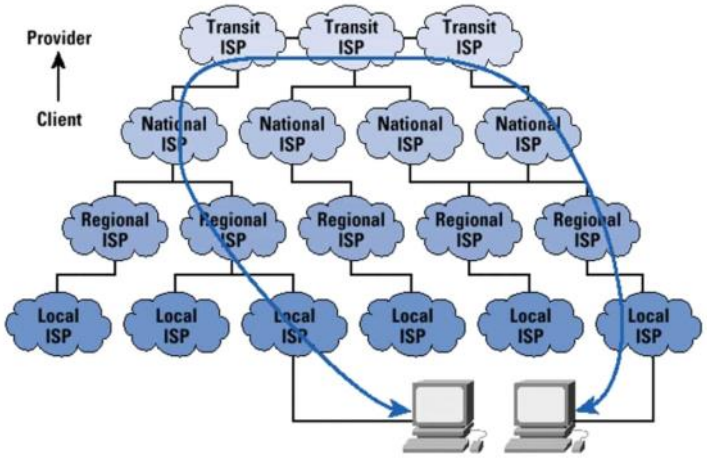
    Bellman-Ford equation

- **Distance vector protocal:** Initialization stage
  - For each router in the path:
    - maintain its distance vector
    - know cost to each neighbor
    - So each router know the cost from itself to it's neighbor
  - 

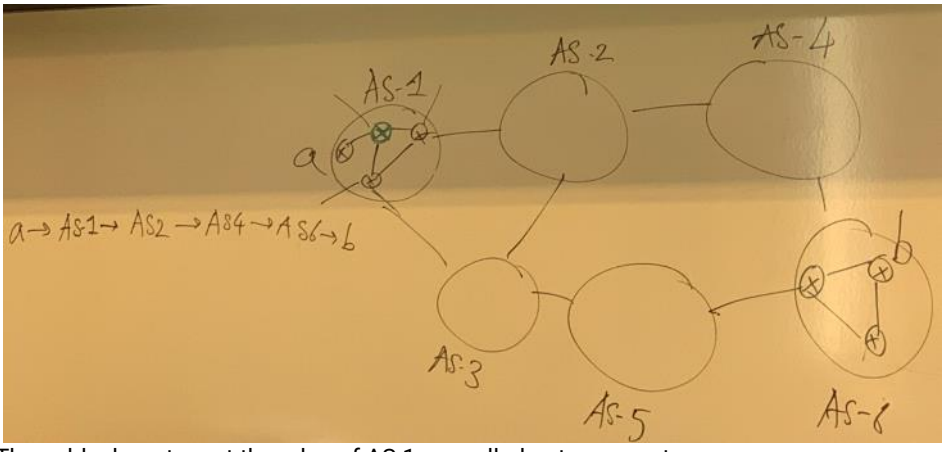
    - Here d sends its DV to a

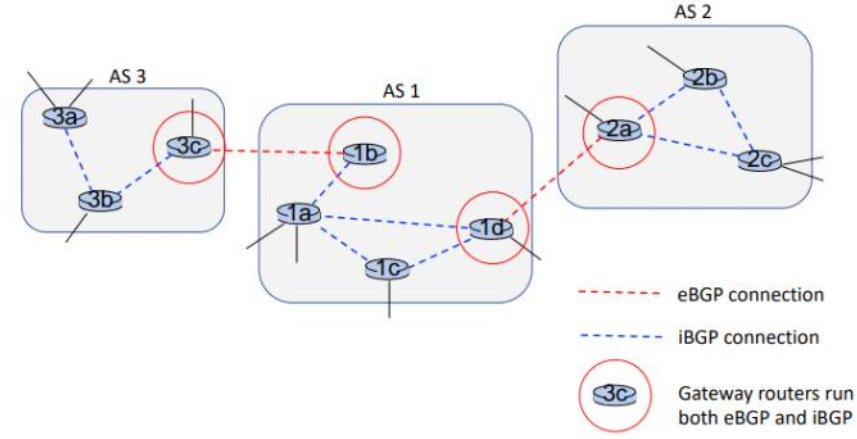
        - there are two table, one is the old DV that a is maintaining, it has all the cost from a to its neighbors
        - Second table is the DV from b + the cost(a, b), e.g. path from a->b->a == 7 + 7 = 14 , Cost(a->b->b) == 7, Cost(a->b->c) == 7 + 1 = 8
- Comparison of LS and DV algorithms
    - **Link state:**
        - At each node
            - Broadcast link cost information to all nodes in the network, such that all nodes has a identical ad complete view of entire map and link costs.
            - Run Dijkstra's algorithm to compute least cost path from node to all other nodes
            - Update forwarding table based on resulting least cost paths.
    - **Distance vector**
        - It's a decentralized algorithm. (Contrast to Link state, a centralized algorithm)
        - No like LS, for which need to know the information fo entire map. For DV, we run Bellman-ford algorithm, so each node maintains an estimate of its distance to all other nodes: called a distance vector. From time to time nodes share their distance vectors with their neighbors
        - When a node receives a new distance vector from its neighbor, it updates its own distance vector
        - The updated distance vector is then shared again with neighbors
    - The actual implementation were determined at Control plane (by network provider), but both has practical implementation in reality.
    - What is different between two is it's initization step? One broadcast link cost to all nodes, so each router has information about every router in the network? Or Is there a central administrator to manage this
        - **For LS(link State)**, it use broadcast strategy, so every router has the infromation about entire network,whenever a new router added to the network, the topological graph at each router will be update. However, it can be substantially large to store the entire map of internet, so we have something called autonomous system(manage by local ISP), so every router only need to remember the Networking map of local region.
        - **For DV(distance Vectors)**, it use the local network strategy. So each router only need to maintain the distance cost from its neightbor, so whenever a new router added connect to it, the cost information will be updated.
        - For LS, it use dijskra's algorithm. The Dijskra algorithm required the information of entire map before running it. However, for DV, we don't have the information about entire network, so we use Belllman-Ford algorithm.
        - What is store d is difference
            - ***LS:***
                - with n nodes, E links, O(nE) msgs sent
                - preferable for small networks
            - ***DV:***
                - exchange between neighbors only
                - preferable for large networks
        - runtime is idfferent
            - ***LS:***
                - $O(n^2)$
                - may have oscillations
            - ***DV:***
                - convergence time varies
                - may be routing loops
        - Can cause different malfunctions
            - *robustness:* what happens if router malfunctions?
            - ***LS:***
                - node can advertise incorrect *link* cost
                - each node computes only its *own* table
            - ***DV:***
                - DV node can advertise incorrect *path* cost
                - each node's table used by others
                    - error propagate thru network
- How to make the routing scalable?
- How would you plan a jouney from columbus to Boston?
    - First, find a sequence of major city you need to travel through

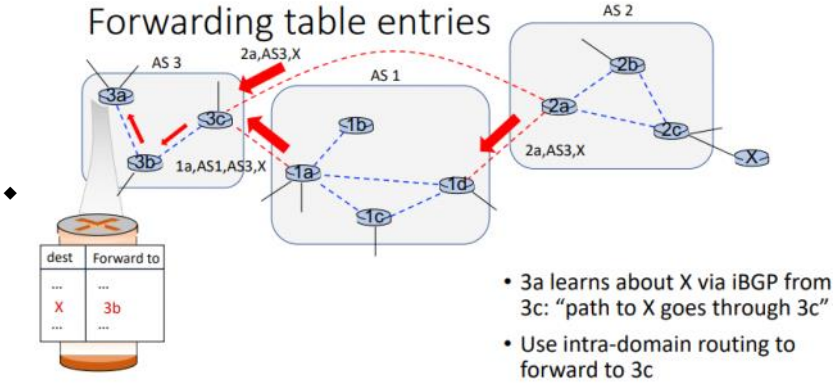
- (Midterm1 Stop here….)
- 2/19, Lecture 18: Networking layer: Control plan: Inter-domain routing

- HW4 Q4: Interdomain Vs Intradomain routing
  1. What is intradomain routing?
     - routing among hosts and routers within the same AS.
  2. What is interdomain routing?
     - Routing across AS(autonomous system)
  3. What is the goal of an intradomain routing algorithm such as link-state or distance-vector routing?
     - To find a path between two entities within an autonomous system by optimizing on metrics such as distance etc
  4. What is the goal of an interdomain routing algorithm like BGP?
     - The goal is to ensures reachability. Discovering information regarding routing paths between domains and propagating that information to the internal routers within the domains.
  5. Why are these goals different?
     - The goal of **interdomain routing** is to ensure the rechability, i.e. to ensure that theres is some path to get from one host on the internet to another. This is contrast to the **intradomaing routing**. With intradomain routing, the goal was to find a good path that optimize some performance or cost, e.g. lowest latency, highest bottleneck link capacity, etc. This is possible only in intradomaing routing, because all routers are under a single administrative control. Whereas, in interdomaing routing, the network communication have more complex policies connected between different AS, which usually based on the business logic and trust between their network administrator.
- **Autonomous system(AS)** -- 自治系统
  - Network of routers controlled by single administrative, such as a univeristy, an enterprise company(i.e. Yahoo, Facebook), an Internet Service Provide(i.e. ATT, Spectrum), or a content provider(i.e. Googl, Netflix, or Facebook)
- **Intra-domain routing (Within a AS)**
  - Means routing among hosts and routers within the same AS. All routers in same AS must run same intra-domain routing protocal, but router in different AS can run different intra-domain routing protocol, e.g. LS or DV.
- **Inter-domain routing (Between differnet AS)**
  - Means routing across different AS. The goal is to ensures reachability. The inter-domain communication is usually managed by BGP protocal, without the interdomain routing, as an OSU student, you won't be reach to the ourside world.
- How many gateway router exist, and how many AS could exist?
  - There can be as many gateway router as you want, and it's possible all the router in an AS are gateway router
  - There also can be as many AS as you want, but it's more expensive to make a direct connection to upper level ISP than the regional ISP.
    - So the graph at the local ISP might be large, and the high-level graph can be sparse. So, the inter-AS routing was control by any machine or person, it's just determined by the Business model for different company. The administrative of AS can control it.
    - 

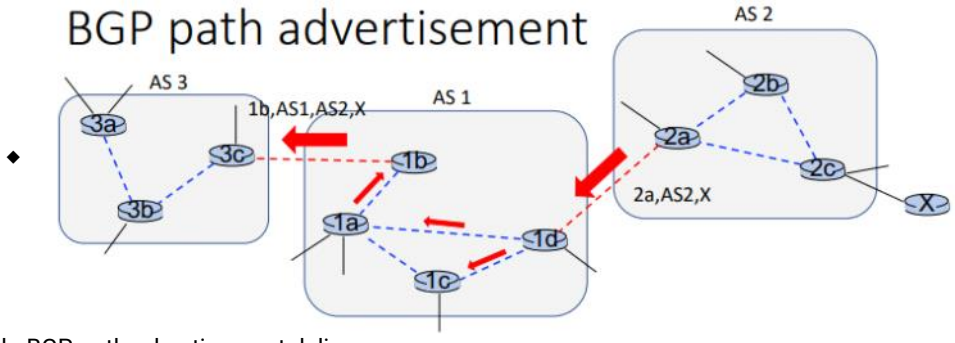
  - Q: What is the difference between Intra-AS routing protocol(e.g. LS, DV) and inter-AS routin protocol (e.g. BGP)?
    - Inter-AS communication was managed by BGP advertisement, and intra-AS communication was managed by algorithm, either LS or DV.
- **Border routers** (aka Gateway router)
  - A border router is simply an IP router that is responsible for the task for forwarding packets between AS. F Border routers sit at the boundary of a domain and have a direct connection to another border router sitting at the boundary of an adjacent domain.
  - The Border gateway routers perfrom inter-domain as well as intra-domain routing

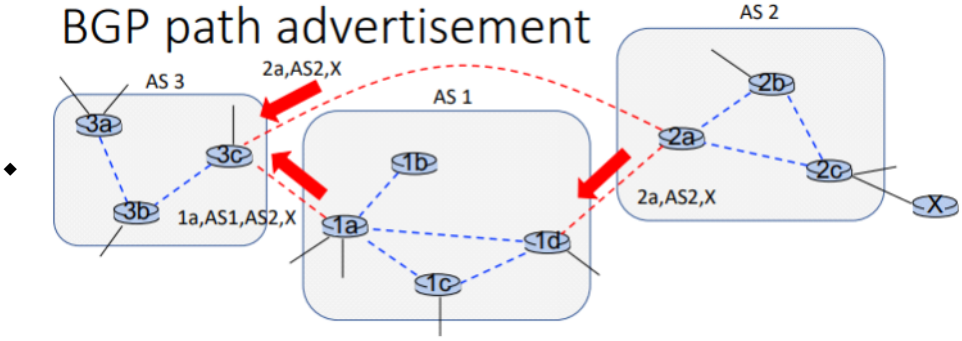
- 

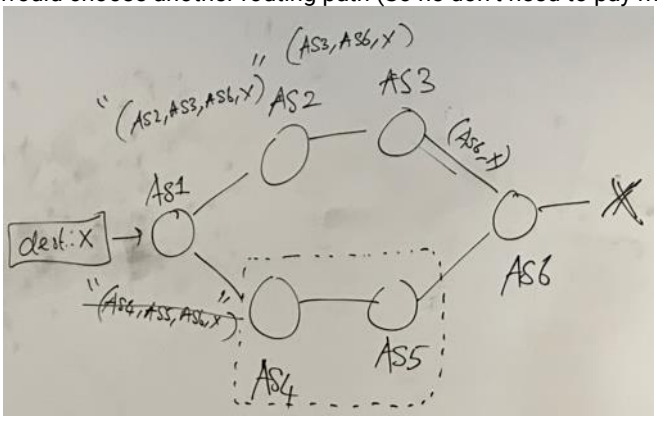
    - E.g. Those black routers at the edge of AS-1 are called gateway routers
- **BGP(**Border gateway protocol)
  The predominant protocol for interdomain routing is the Border Gateway Protocol (BGP). This protocol is responsible for: 1)Discovering information regarding routing paths between domains ==> eBGP, and 2)Propagating this information to the internal routers within a domain. ==>iBGP In more detail, the first step is responsible for figuring out a sequence of domains that gets a packet from a source domain to a destination domain. The second step is responsible for telling an internal router in the source domain which border router it needs to send packets to so that the packets can eventually find their way to the destination.
    - How the information being exchanged between two AS via eBGP?
        - Based on eBGP protocal, 1b gateway/border router at the boundary of AS1 will tell(or promise) the gateway router 3c two things: 1)You can reach any IP address within AS1 by adding prefix (1b, AS1), 2)You can reach any IP address within AS2 by adding the sequence of domain predix (1b, AS1, AS2).
        - 

            - **Path Advertisement** is like "Making promise". For example of following graph, when 2a sending the eBGP advertisement, 2a promise it has the infromation(routing info) to X. So, AS1 will use its own algorithm to compute a inter-AS routing path and extend the reachability information from X to next AS(i.e. AS-3)
            - 

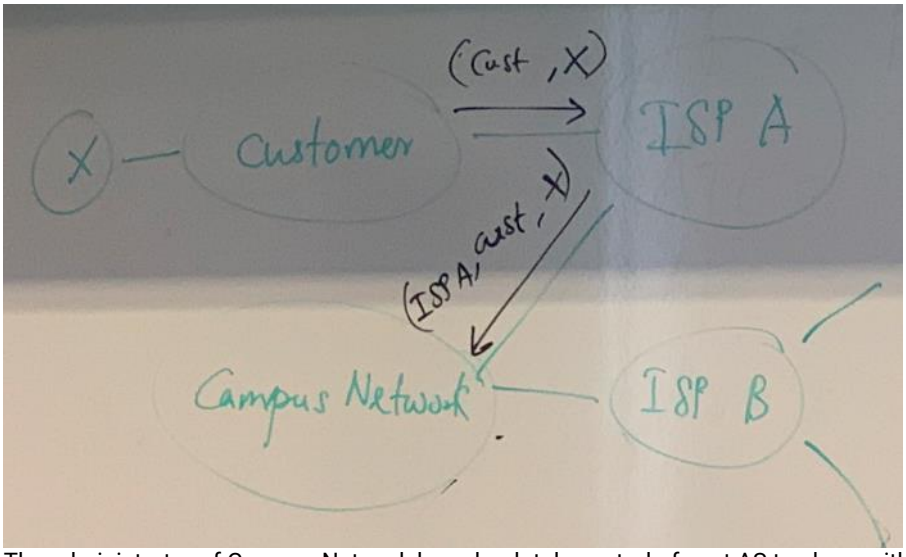
    - What is the difference between eBGP and iBGP
        - The **eBGP** (for external BGP) protocol is responsible for obtain subnet reachability information from neighboring/external AS'es. Also known as "path vector" Protocol.
        - The **iBGP**(internal BGP) is responsible for propagateing reachability information to the routers within/inside a AS
            - In short, eBGP say, "Let's gather some useful data outside my house"
            - iBGP says, "Wooo, delivers this fancy news to everybody in my family.."

- BGP path advertisement
  - Common BGP path advertisement process:
    - Let's see an example to how gateway router can learn tranmission path via BGP protocol, or how interdomain protocol works: (Say X want to send some package to 3c, and we need to find a way to make connection between them)
      - After router 1d receives path prefix (2a, AS3, X) from router 2a (via eBGP).
      - Based on AS1 policy, router 1d may accepts the path and propagates this reachability information to all AS1 routers (via iBGP) .
      - Based on AS1 policy, router 1b can advertises the path prefix (1b, AS1, AS3, X) to router 3c(via eBGP). Such that, the router 3c will know how to make connection to X.
    - 

  - Multiple path BGP path advertisement delivery process
    - Gateway routers may learn multiple paths to destination as well, for example:
      - Router 3c can learns about path prefix (AS1, AS2, X) from router 1a
      - Router 3c can also learn another path predix (AS2, X) from 2a.
      - Based on their business policy, router 3c can choose either paths and advertises this reachability information within its AS via iBGP protocal.
    - 

  - Is possible there are two AS routing sequence that has same cost? if so, who will decide which routing path will be used?
    - Yes, it depends on Business model, or relationship between two AS, or company.
    - For example, there can be two way from AS1 to X. Even AS1-AS4-AS5…X has a cheaper cost, but not necessary the packet will be send in this way. Because it's possible AS1 and AS5 are competitor, and they hate each other, so AS1 would choose another routing path (So he don't need to pay money to AS5 for inter-AS communication).
    - 

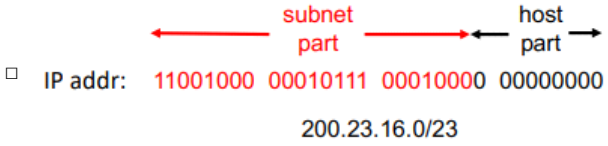
  - BGP route selection
    - If router learns about more than one route to destination AS, select route based on following three:
      1) AS administrator policy decision



- The administrator of Campus Network has absolutely control of next AS to share with. E.g. If Campus Network has a bad business relationship with ISP B, it can just block this information with ISP B.

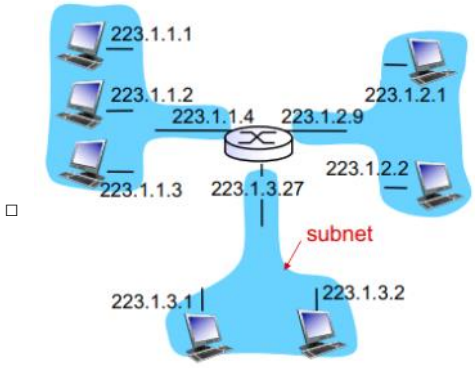
2) Path visiting fewest number of AS'es
3) Choose the path with smallest cost (intra-domain routing cost)

- ## 2/28: Slide18: Inter-domain routing
  - IP addressing
    - IP address: 32-bit identifier for host, router interface
    - Interface: connection between host/router and physical link
  - **Subnets**
    - IP address (identical 32 bit)
      - Subnet part: Higher order bit -- first 23 bit
      - Host part: Lower order bits -- last 9bit

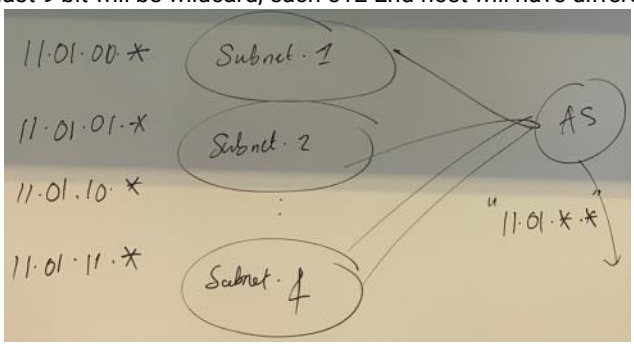


200.23.16.0/23

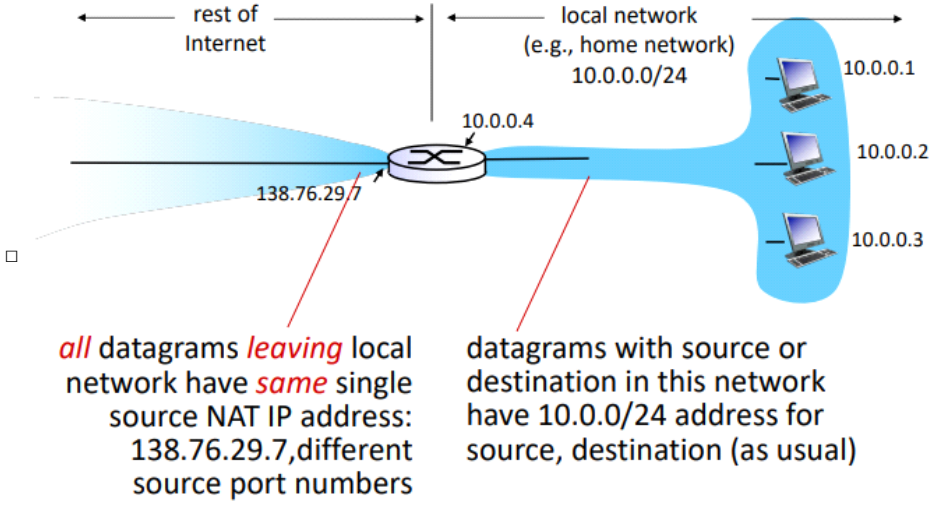
    - What is subnet:
      - Devices that interfaces with same subnet can physically reach each other without intervening router.
      - the IP terms, a network interconnecting three host interfaces and one router interface form a **subnet**. IP addressing assigns an address to this subnet: 223.1.1.0/24, where the /24 ("slash-24") notation, sometimes known as a **subnet mask**, indicates that the leftmost 24 bits of the 32-bit quantity define the subnet address.
      - Device interfaces with same subnet part in their IP address.
      - Can physically reach each other without intervening router
      - Share higher order bit IP address
      - The 223.1.1.0/24 subnet thus consists of the three host interfaces (223.1.1.1, 223.1.1.2, and 223.1.1.3) and one router interface (223.1.1.4). Any additional hosts attached to the 223 .1.1.0/24 subnet would be required to have an address of the form 223.1.1.xxx. There are two additional subnets shown in Figure 4.18: the 223.1.2.0/24 network and the 223.1.3.0/24 subnet.



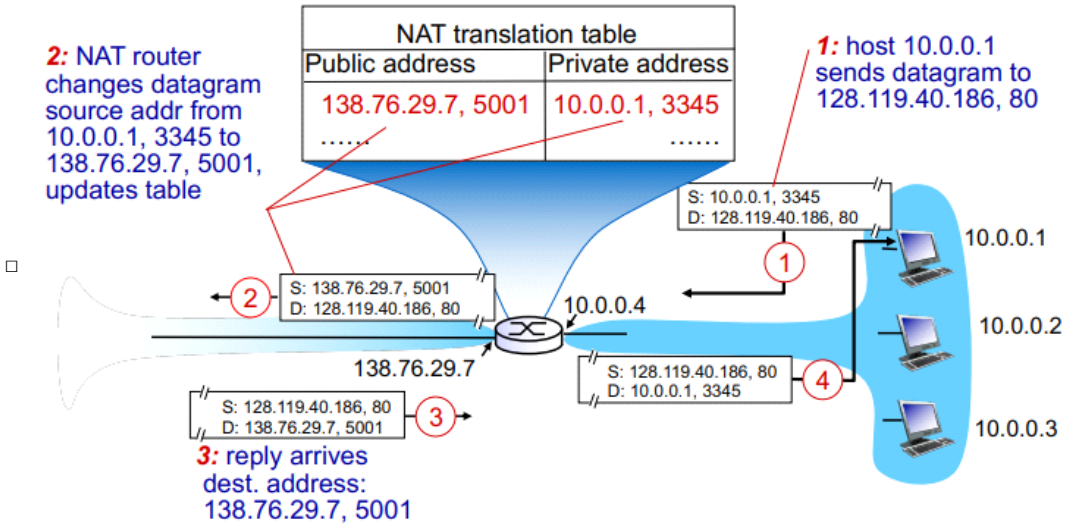
network consisting of 3 subnets

- Hierarchical addressing: **Route aggregation (or Address aggregation)**
  - totally about 2^9 = 512 End-host can be contained within a subnet
  - The last 9 bit will be wildcard, each 512 End-host will have different lower order bit within a subnet.
  - 

- IP addressing
  - Handling IP address shortage
    - Use IPV6--128 bit
    - Use private IP addresses -- requires network address translation
  - Private IP addressing
    - Local address is not visible from outside world, they can change addresses of devices in local network without notifying outside world, but NAT is needed to implement this.
- **Network Address Translation(NAT)**
  - <u>There is a NAT translation table will help us to translate the private IP address to public IP address within a subnet network.</u>
    - E.g. we want to send a packet from Source: 10.0.0.1 to a Dest: 128.119.40.196, the router (managed by private network administrator) will have an NAT to convert our local addrsses to a public address, and it also can be used to hide user physical location.



  - NAT implementation
    - Outgoing datagrames(发出去的): replace (source IP addr, port #) of every outgoing datagram to (NAT IP addr, new port #)
    - Remember in NAT translation table: Translation pair between Source (IP, Port#) and NAT(IP, Port#)
    - Incoming datagrams(发进来的): replace (NAT IP addr, new port #) of every incoming datagram to (source IP addr, port #)
    - Example: Look close on how S and D addr changes from 1 to 2, and from 3 to 4:

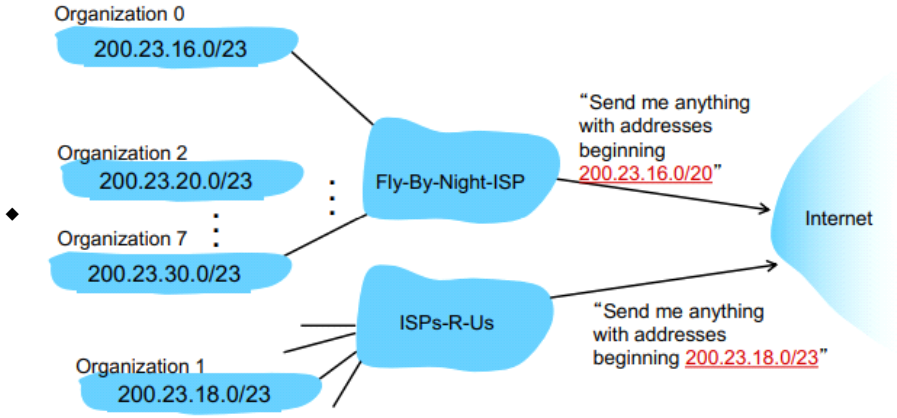


- Advantage:
  - Provides better Security
  - Reduce the number of public IP address

- 3/4
  - Router Architecture
    - input port
  - Destination-base forwarding

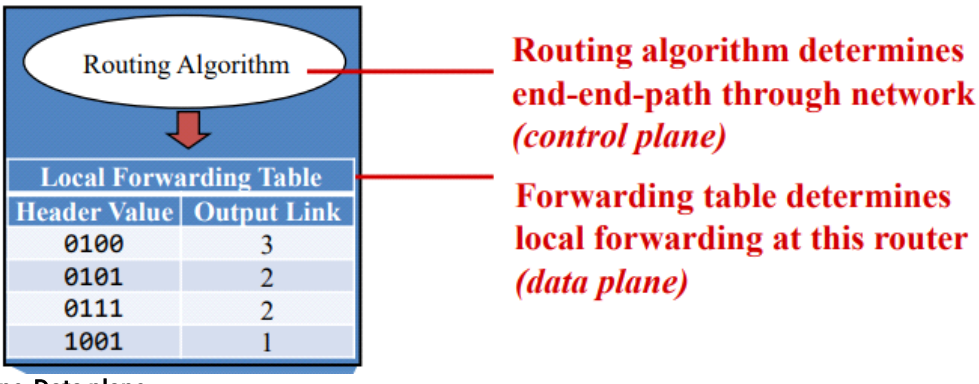
Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

    - All above three are different public IP addr, but they have different range for their local addr. Some of them have 11 bit range(last 11 bit are differ for their local network addr), but some of them only have last 9 bit are differ
    - Note: link Interface 1 is the subset/subnet of Link interface 2?
  - Longest Prefix matching: [reference, http://www.cs.toronto.edu/~ahchinaei/teaching/2016jan/csc358/Tut08-taSlides.pdf]
    - When there are multiple Dest addr you can match in forwarding table, you pick the Dest addr that has the longest prefix to forward. In short, match as many as possible!
    - For example: (using the above Dest addr table)
      - DA: 11001000 00010111 00010110 10100001 Interface 0
      - DA: 11001000 00010111 00011000 10101010 Interface 1
      - All implement in hardward, use the hash table to implement/ directly
    - Another example:
      - ISPs-R-Us has a more specific route to Organization 1, the addr of ****18.0** purchase by them, so the packet was send to ISPs-R-US

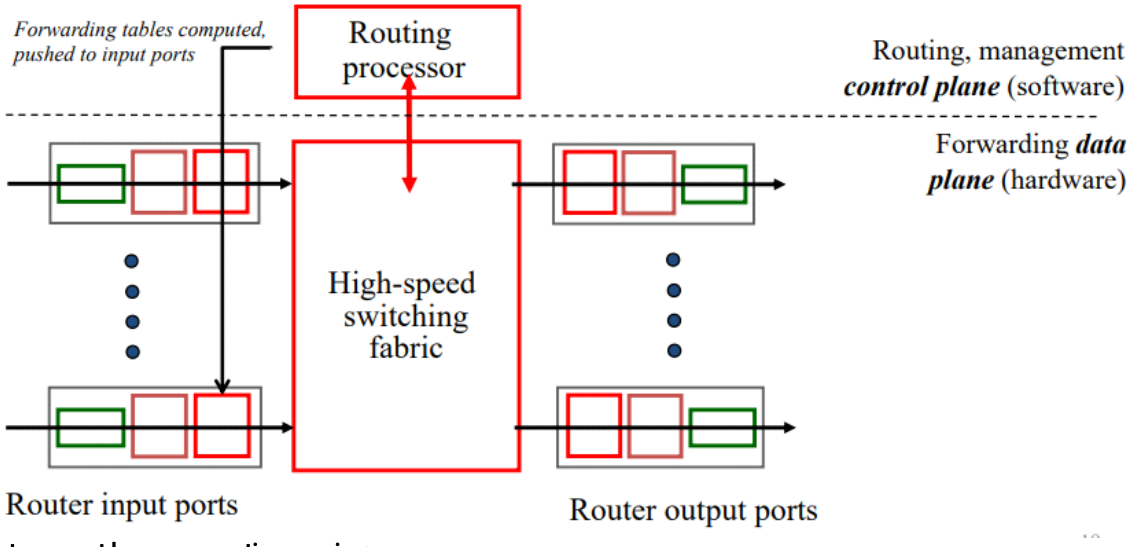


- 3/6 Slide19: Networking layer: Data Plane: Switching fabric
  - **Network Sevice model:**
    - Motivation:
      - 1)Can the transport layer rely on the network layer to deliver the packet to the destination?
      - 2)When multiple packets are send, will they be delivered in the order in which they were sent?
      - 3)Will the amount of time between the sending of two sequential packet transmissions be the same as the amount of time between their reception? -- 发送两个连续的包传输之间的时间是否与接收它们之间的时间相同?
      - 4) Will the network provide any feedback about congestion in the network?
    - What is Network service model: It defiens the characteristics of end-to-end delivery of packets between sending and receiving hosts:
      - **Guaranteed delivery:**This service guarantees delivery of the packet, such that a packet sent by a source host will eventually arrive at the destination host.
      - **Guaranteed delivery with bounded delay:** This service not only guarantees delivery of the packet, but guarantess the delivery time is within a specified host-to-host delay bound (for example, within 100 msec).
      - **In-order packet delivery:** This service guarantees that the orders packets arrive at the destination in the order that they were sent.
      - **Guaranteed minimal bandwidth**: This network-layer service emulates the behavior of a transmission link of a specified bit rate (for example, 1 Mbps) between sending and receiving hosts. As long as the sending host transmits bits (as part of packets) at a rate below the specified bit rate, then all packets are gaurantee will be delivered to the destination host. 只要发送主机以低于指定比特率的速率传输比特(作为包的一部分)，那么所有的包最终都会被发送到目标主机.
      - **Security**: The network layer could encrypt all datagrams at the source and decrypt them at the destination, thereby providing confidentiality to all transport-layer segments.
    - However, None of this services are guarantee in today Internet service. The Internet's network layer provides a single service, known as **"best-effort service"**.今天的互联网只保证一件是：我尽量 - -!
      - The Internet's basic best-effort service model combined with adequate bandwidth provisioning have arguably proven to be more than "good enough" to enable an amazing range of applications, including streaming video services such as Netflix and voice-and-video-over-IP, real-time conferencing applications such as Skype and Facetime.

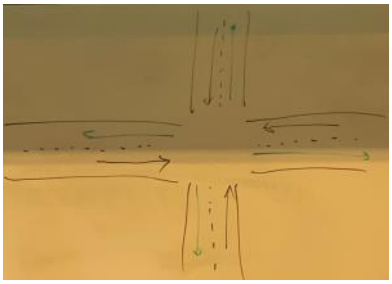
  - **Networking Layer Vs Transport layer comparison:**
    - **Segment Vs datagram**
      - **"transport layer segments"** is defined as the packets in Transport layer
      - **"datagram"** is defined as the packet in Network layer
    - **Routing Vs forwarding**
      - **Routing**: Decide the overall route taken by packet from source to destination, or end-end-path through network(Control plane)
      - **Forwarding**: Decide which path to go in certain intersection/router, determine the local forwarding at this router.



- **Control plane, Data plane**
  - **Control plane:** determine how datagrame is routed from Source host to Dest host (E.g. Network-wide logic, routing algorithm)
  - **Data plane**: determine how datagrame is routed from input port to outport port (e.g. Local router function, Forwarding functions)

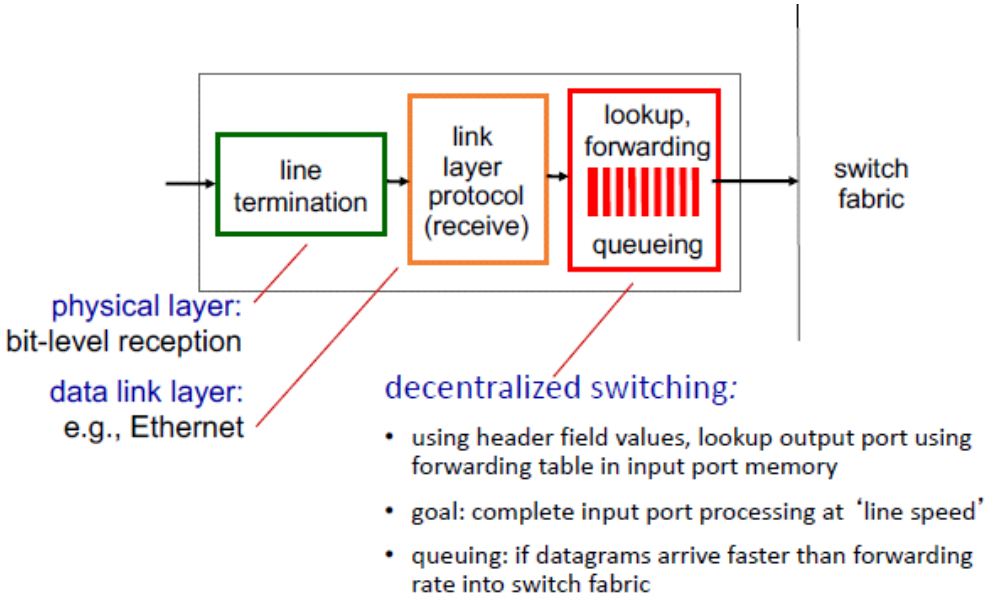


  - **Network vs. transport layer connection service**:
    - **Network/Routing layer**: provide services between two hosts
    - **Transport layer:** provide services between two processes on end-system
  - Forwarding data plane(includes input port, output port, and switching fabric) are almost always implemented in hardware, which support as extremly high speed data transfer.
- **Switching fabric(交换结构)**
  - **Goal**: transfer packet from input port/buffer to appropriate output port/buffer
  - **Terminology**
    - **Routing:** It's a algorithm implemented in the **control plane**. The algorithm that calculate the path/route taken by packets as they flow from a sender to receiver.
    - **Forwarding:** It's a function implemented in the **data plane**. When a packet arrices at a router's input link, the router must move the packet from input port to the appropriate output link.
      - What is difference between routing and forwarding?
        - **Forwarding** is a more router-local action, and **routing** refers to a network-wide process that determines the end-to-end paths from source to destination. Analogy for this, the routing is like planing a trip from Pennsylvania to Florida, and forwarding is like the process of getting through a single interchange(交叉路口).



    - **Input port**: Three key functions performs at input port:

1) the physical layer function of terminating an incoming physical link at a router; this is shown in the leftmost box of an input port in Figure.它执行物理层功能，终止进入路由器的物理链路;
2) An input port also performs link-layer functions needed to interoperate with the link layer at the other side of the incoming link; this is represented by the middle boxes in the input and output ports. 输入端口还执行与进入链路另一侧的链路层互操作所需的链路层功能; 这由输入和输出端口中的中间框表示
3) Perhaps most crucially, a lookup function is also performed at the input port (也许最重要的是，在输入端口也执行查找功能); this will occur in the rightmost box of the input port. It is here that the forwarding table is consulted to determine the router output port to which an arriving packet will be forwarded via the switching fabric.在这里，参考转发台来确定路由器的输出端口，到达的包将通过交换结构转发到路由器的输出端口。

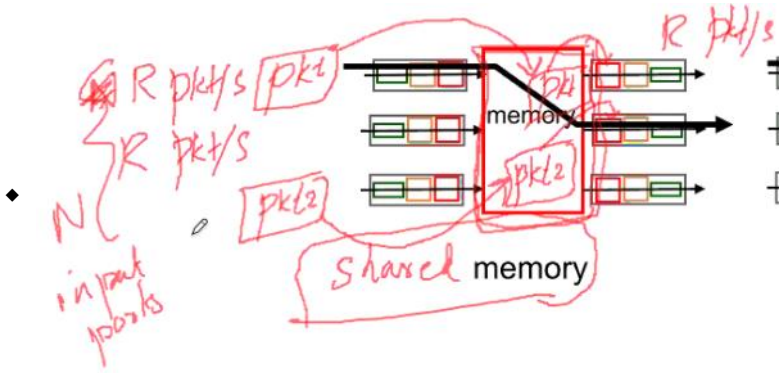


- **Output port:** 接受从Switching fabric 传来的packet，然后再传送到Pyhsical layer. An output port stores packets received from the switching fabric and transmits these packets to the outgoing link by performing the necessary link-layer and physical-layer functions. (Note: an output port will typically be paired with the input port on the same line card)
- **Switching fabric**: The switching fabric connects the router's input ports to its output ports. This switching fabric is completely contained within the router-a network inside of a network router!
- **Switching rate**: rate at which packets can be transferred from inputs to outputs buffer/port
- line rate: == transmission rate, link rate means capacity of incoming link
- **Tick:** The time that it takes to receive the smallest size packet on an input port or send the smallest size packet on an output port. (Two are typically equal, because input port and output port are connecting to the same cables)

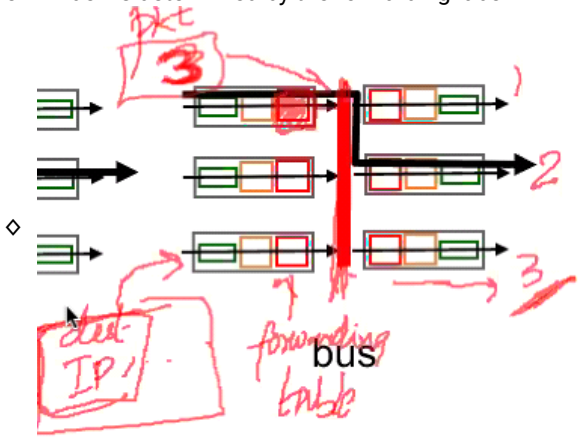
○ Q: Where does the Switching fabric is implemented? ==> A router's inut port, output ports, and switching fabric are almost always implemented in hardware!

○ Analogy of highway traffic policy implementation
- Traffic signal
- 4-way stop
- roundabout -- (交通）环岛

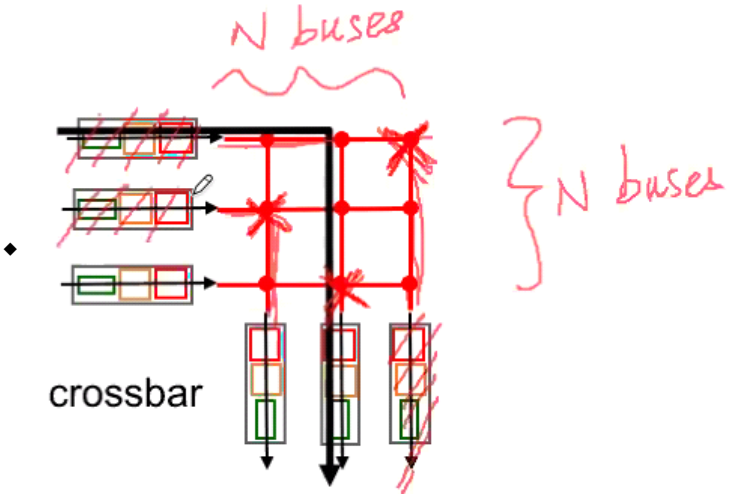
○ Three types of switching fabric
- **Switching via memory: Share Memory architecture**
  - When every packet reached the memory pool, they shared memory. Only one comment queue sitting at switch.
  - Bad: This pool needs to support up to N enqueues per tick and up to N dequeues per tick so that the memory is sufficiently powerful to handle the worst case of a packet on each input and output port at every time tick. (See the following pictures, if the line rate is R pkt/s, you would need a NR pkt/s read/write speed for memory)
  - Good: The ability to dynamically allocate more or less of the same memory to different output ports on demand.



- **Switching via a Bus (aka Output-queued swtich)**
  - Idea: Instead of having one shared memory pool for all inputs and all outputs, we have N separate memories/queue buffer for each output port. Then each memory only needs to be powerful enough to support N enqueue and 1 dequeue per tick,
    - If multiple packets arrive to the router at the same time, each at a different input port, all but one must wait since only one packet can cross the bus at a time. Because every packet must cross the single bus, the switching speed of the router is limited to the bus speed; in our roundabout analogy, this is as if the roundabout could only contain one car at a time. Nonetheless, switching via a bus is often sufficient for routers that operate in small local area and enterprise networks.
  - **Good**: a considerable reduction in the number of operations per tick, comparing to Sharing pool architecture. (N read per tick can still be challenging some time)
  - **Bad**: The downside is the memory can no longer be shared across output ports. Every output port is designed to have an exclusive memory dedicated to a particular output port alone, and that cannot be allocated for other output port, even you have one is in idle and the other is filling up(装满) with packets.
    - the IP index is determined by the forwarding label



- **Crossbar(aka Input Queueing, The most popular one in real world application)**
  - This architecture requires N memory memories as well, but they only need to powerful enough to support 1 read/enqueue and 1 write/dequeue per tick.
  - The property for cross bar(input-queue router): 1)Each input is connected to at most one output. 2)Each output is connected to at most one input.
  - Thus, this problem can be formulated as a bipartite matching problem. However, solving this matching problem, with maximal match or maximum weight matching, is hard at the speeds of a router.

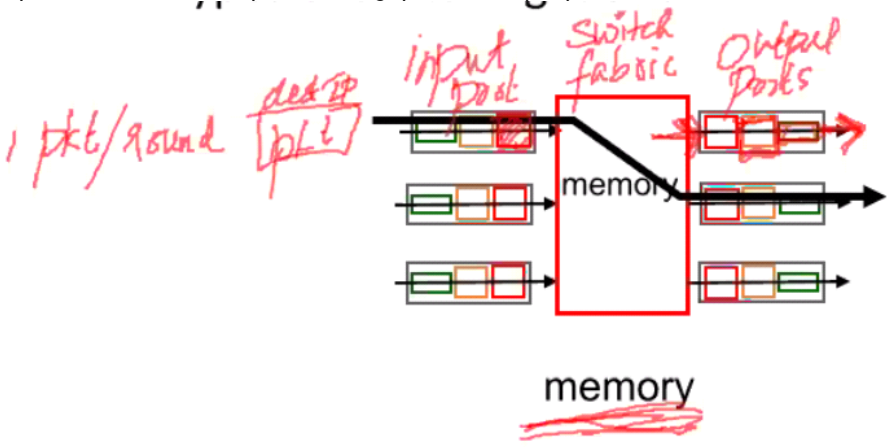


- you are only allow perfect matching, e.g.
  - imput port 1 -- output port 3
  - imput port 2 -- output port 1
  - imput port 3 -- output port 2

- Note: each round can only connect/transport 1 imput port to 1 output port
- Note: The number of input and output ports is typically the same, because there is only one physical port.
- Who actually decide the which switch policy to use? -->software? the router manufacture?
  - ==> Router manufacture, cost, currently technology trend
- Could you compare the difference between these three switching mechanism? Which has the best performance?
  - ==> crossbar

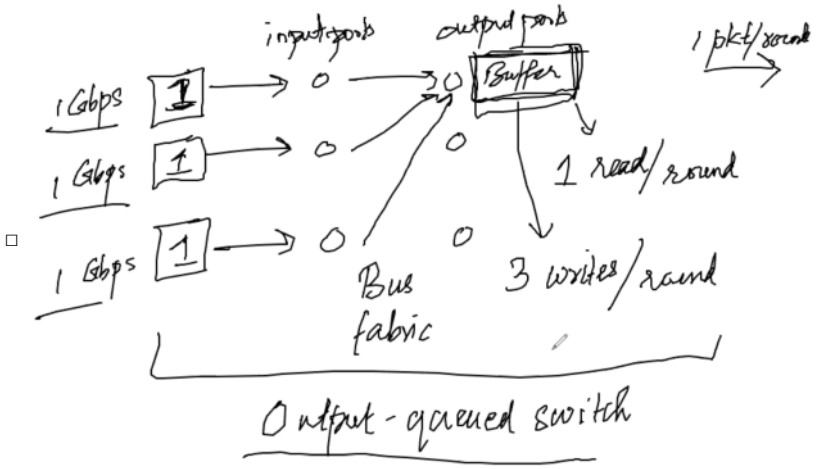
3/23 Slide19: Data plane (cont...)

- Review
  - Three type of switching pattern
    - Shared memoery
      - Output-queued swtich: memoery being queued at output port(there are input and output port)
      - Input-queured switches: the packet being queued at input port



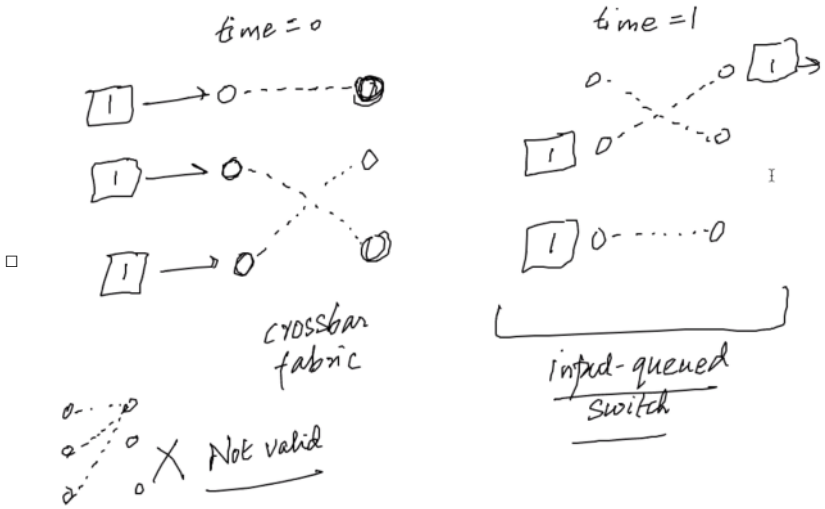
- yellow: link layer packet, encapsuluate the packet, e.g. link layer header
- Red: networking layer, has forwarding table
- the link layer at output port, conver data to ananlog signal for physical layer
- the queueing can happen either input or output port

    - Bus



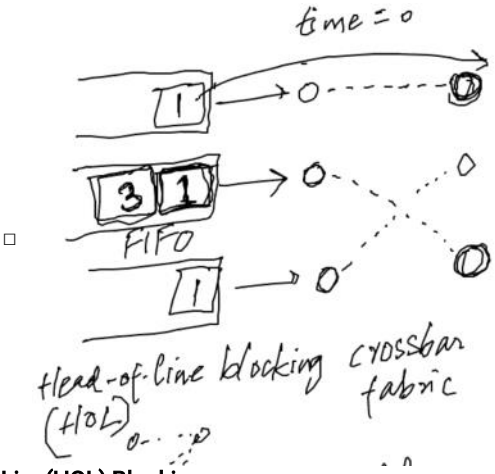
- all the packet have to go to input port 1, when three packet arrived. When we are looking at the queue at output port 1, the buffer need to be able to support 1 read and N write in worse case.
- What is the advantage of bus over others
  - Compare to Sharing memory architecture: When there are n packet arrive at input port, the Share memory need to be fast enough to support N reads and N write at a time. So shareing memory architecture need to have a high speed switch, which can be very hard to do.

- Crossbard

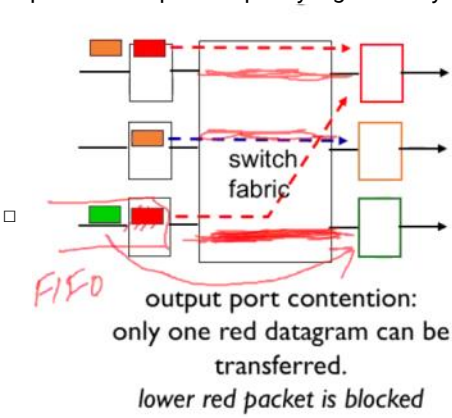


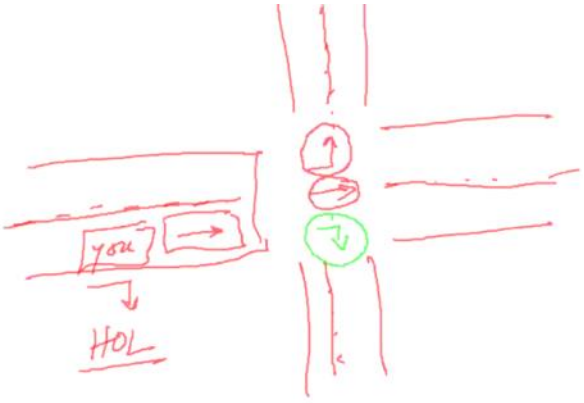
- The most popular implementation in real world
- Bus is output queued switch, and crossbar is input queued switch(It only allow to match one switch at a time slot, you cannot match two input port to same output port at a time!)
- You can have different configuration at different time, but only one packet can be send to one output port at a time, so it's limited by the configuration
- only need to fast enough to support 1 read & 1 write in any case. e.g. In worst case, 3 packet arrive that need to deliver to output port 1,

- Queuing problem for cross-bar switch
  - Q: If another packed had arrived at input port 2, destined for output 3, why wouldn't it switch the two packets, so its not just sitting there doing nothing?
    - because we are using FIFO technique, so it must wait until all packet in front of it get clear. it's known as **Head-of line blocking.**

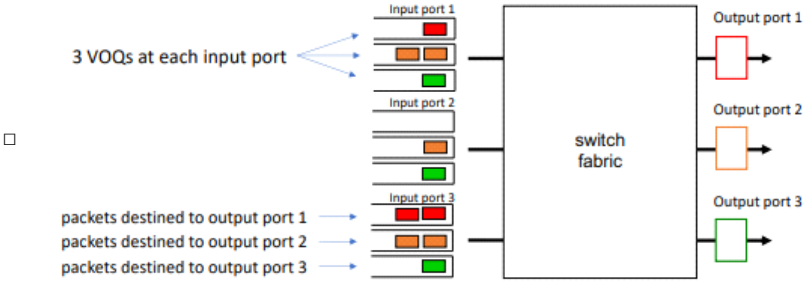


- **Head-of-the-Line(HOL) Blocking**:
  - Idea: Every input port has memory that is organized as FIFO with one enqueue and one dequeue per tick. At evrey tick, an output
  - What is HOL blocking? --> queued datagram at front of queue prevents others in queue moving forward
  - Example:
    - The green packet doesn't do anything useful, so it has to wait until the red datagram get transferred.
    - This problem drop the capacity significantly on the line rate

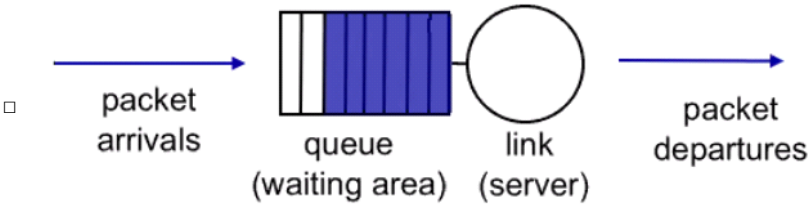




    - Q: Suppose input port 2 is connected to output 3 and input port 3 is connected to 2. Is that still HOL blocking if there is no actual connection from input port 3 to output port 3? ==> No, it's not
- So, how possible this problem was being solve in real world?
  - **Virtual Output Queue(VOQ)**: Maintain N separate queues for each input port, based on destination port of packet
    - Maintain three input virtual queue(FIFO policy) at a time, each queue was destine to certain outport port. With this VOQ, the capacity can be bring the capacity back to 100%

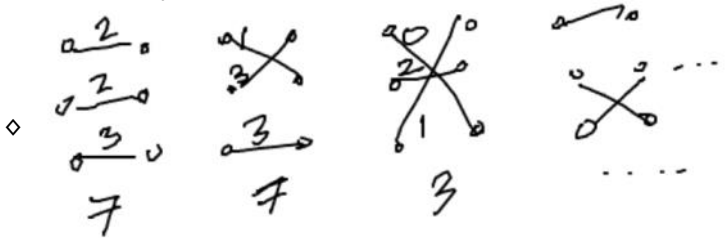


  - We call it VOQ is because the packet are destine to ourport, the input port is actually input port, and we are trying to minimize the number of packet queued at input!
- **Scheduling policy: Packet Scheduling**
  - Goal: Understanding What is scheduling policy? --> Deciding what is the most appropriate way to match the connection between input port and output port at next time slot! Usualy it depends on the number of packet are waiting at VOQ. If there are more packet, you would like to give it a high priority in switch. The job of packet scheduling is to decides which packet to transmit next from among the queued packets. There are three common scheduling policy for output queue:
    1) **FIFO**: transmit packets in the same order as they arrived -- No guarantee whatsoever.

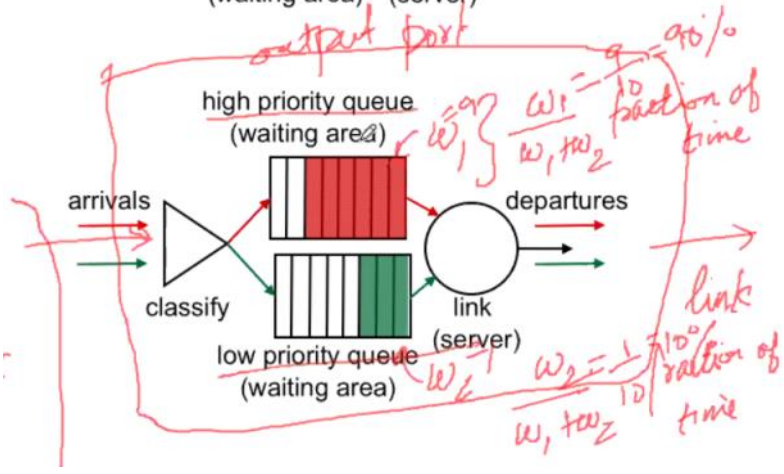


2) **Priority scheduling**: Sending the packet based on the their priority. When a flow of higher priority packet exists, send it before flow of lower priority. May end p starving some flows, not really fair, but useful when you want to prioritize certain type of traffic.
3) **Max weight scheduling**
   - Can possibly achieve 100% throughput
   - Idea: Draw a bipartite graph, find a match such that total weight of match in the graph is maximum. For example of the following graph, the best match would has a max weight: 2 + 2 + 3 = 7

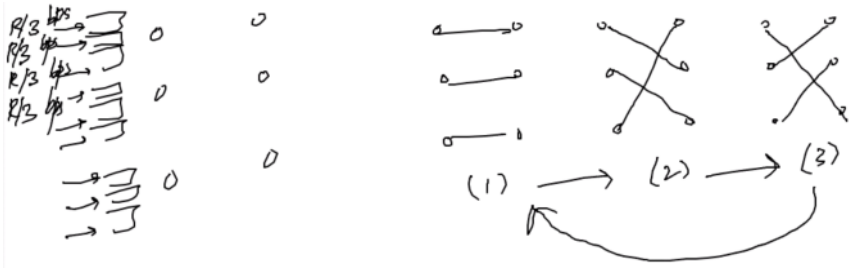
   - The dumpest way is to looking all the possible match, and find the max weight (If tie, just pick arbitrary)



   - **Time slot**: Time it take to send one packet from input port to output port
   - There are many other scheduling policy: Maiximum algorithm, througput optimality, there are hundreds of work had done, so ask google!
   - Some variation:
     - Weighted fair scheduling: Lets specify the weights percentage.
     - Weighted fast scheduling: taking the pakcet who has a higher weight, the pencentage is change dynamically! (Based on the amount of packet in their waiting area_



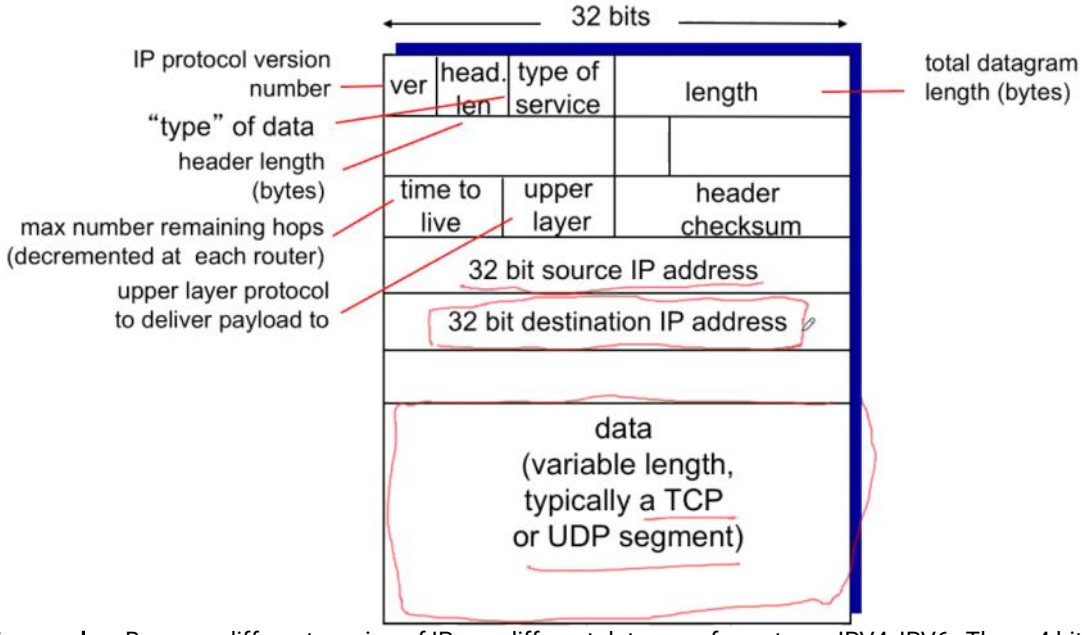
4) **Round Robin Scheduling**
   - Idea: you cicle those matching repeatively. Has no priority, sending packet in Round Robin fashion, taking one packet from high priority queue and taking one from low priority, and so on and so forth.
   - So it's a more fair, uniformly draining the packet for each VOQ(virtual output queue)



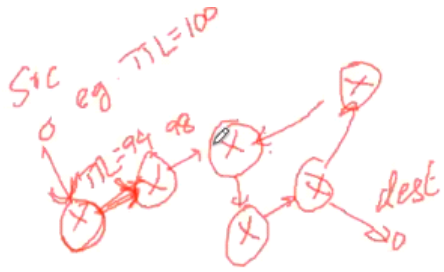
- Q: Does it necessary to have the same number of input port and output port?
  - Pyhsically, input and output port are siting in same line card, so the total number of input port is equal to the total number of output port
- [Reference]: http://www2.ic.uff.br/~michael/kr1999/6-multimedia/6_06-scheduling_and_policing.htm

- Active Queue Management(AQM): If the packet overflow in queue, the scheduler will take action proactively. ----Not on exam
  - It won't wait buffer get full, instead, it will drop the packet before buffer get full, aka proactively dropping technique
  - or another technique is called ECN (explicity congestion notification): So when the buffer filled about 90%, we will mark the packet that is about getting full
    - Instead of reacting after the fact congestion had happen, the ECN technique can proactively start to taking the action, marking the packet, the notify end-host the buffer is about to host, to slow the transmission rate or window size.
  - --> Why do we want to do that? 1)To control the size of buffer. Dropping packet is bad, so it good to have a better control of how to transmitting packet from input port to output port. 2)to avoid the synchronization problem.

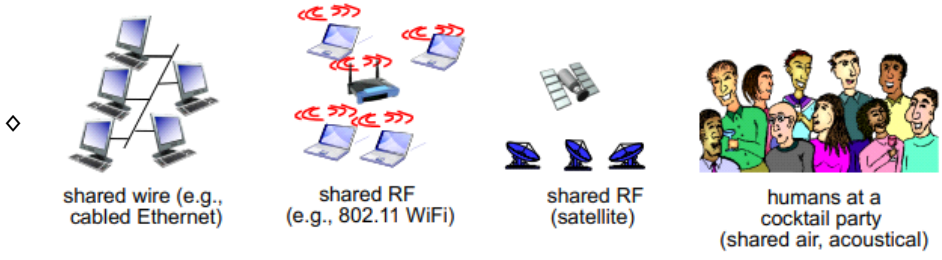
- **IP datagram format (example of IPv4 datagram)**
  - **Data(payload):** contains the Transport-layer sement, and other type of data, e.g. ICMP message…. So, when the data comes down from transport layer, you're going to add a network layer header to the packet, which is everything on the top of data.



  - **Version number**: Because different version of IP use different datagram format, e.g. IPV4, IPV6…These 4 bit number can determine how to interpret the remainder of IP datagram.
  - **Header Length:** The length of IP header. Because an IPv4 datagram can contain a variable number of options, these 4 bit are needed to determine where in the IP datagram the payload actually begins,.
  - **Type of service**: The type of service (TOS) bits were included in the IPv4 header to allow different types of IP datagrams to be distinguished from each other. e.g. telephony application is a real-time service, which is different from non-real-time level of service, such as FTP. Therefore we can use it on the priority scheduling.
  - **Datagrame Length**: The length of the IP datagram (header plus data), measured in bytes.
  - (The blank aread in the middle)Identifier, flags,fragmentation offset: These three fields have. to do with so-called IP fragmentation, a topic we will consider shortly. Interestingly, the new version of IP, 1Pv6, does not allow for fragmentation.
  - **Time to live:** The time-to-live (TTL) field is included to ensure that datagrams do not circulate forever (due to, for example, a long-lived routing loop) in the network. e.g. let it be 100 TTL. the value will decrement by one for each router had touched, and when the TTL get to zero, we will stop forwarding the packet. But if there is bug in your algorithm, you don't want the packet to keep cycling forever.

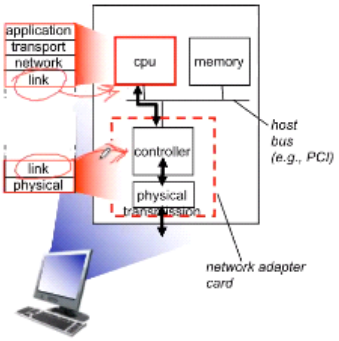
- 

    - **Upper layer:** help to decide what is the upper layer protocol that this data is come from, such as TCP or UDP in transport layer.
    - **Header checksum**: detect the bit erroors in a received IP datagram. The header checksum is computed by treating each 2 bytes in the header as a number and summing these numbers using 1s complement arithmetic.(Discuss previously)
    - **Source and Dest IP:** Those are very important because, as we say, the router use them to consult the longest prefix matching and decide what's the appropriate outport port to forward the packet to. The source Ip address is also important, so when the client/host received packet and return the acknowledgement back, so it know where the message should send to.

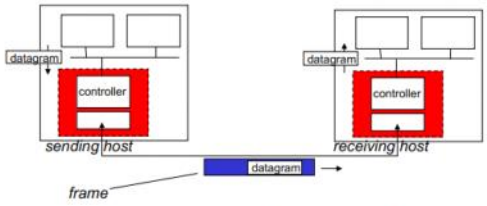
- (45:10 - end)3/30 Slide 20: Link Layer
    - Definition
        - **Node**: Let's call hosts and routers as nodes. Generally, it means any device that runs a link-layer protocol, such as hosts, routers, switches, and WiFi access points.
        - **link** is the channels that connect adjacent nodes along the communication path, e.g. Wired link---Ethernet, fiber, optical cable, wireless links -- cellular, WIFI, satellite. A link is bidirectional, an output port will typically be paired with input port on the same line card(接口卡)!
            - Wired links: Ethernet, fiber, coaxial cable
            - Wireless link: cellar tower, Wifi, satellite
            - LAN
        - **Framing**: A frame consist of a data field, in which the network-layer datagrame is inserted, and a number of headers fields. Almost all link-layer protocols encapsulate each network-layer datagrame within a link-layer frame before transmission over the link.
            - Analogy to other network layer:
                - Transport layer: the packet is called segment
                - Networking layer: the packet is called datagram
                - Link layer: the link-layer packet is called frame
        - **Link access:** If link shared between multiple users(e.g. WiFi), a MAC(medium acess control) protocal can specifies the rules by which a frame is transmitted onto the link.
            - Two types of "links access"
                - **Point-to-point**:
                    - point-to-point links that have a single sender at one end of the link and a single receiver at the other end of the link. e.g. the link between Ethernet switch and host. There are many link-layer protocols have been designed for point-to-point link, e.g. point-to-point protocol(PPP), and high-level data link control (HDLC).
                - **Broadcast**
                    - Broadcast link have multiple sending and receiving nodes all connected to the same, single, shared broadcast channel. In practice, hundres or event thousands of nodes can directly communicate over a broadcast channel. More common in **wireless link,** e.g. 802.11 wireless LAN, WiFi, cellphone tower, and etc, but also possible in wire cable (e.g. Ethernet) So, when a node send a packet, everyone in the range could "heard"/have access to it.
                    - The protocol for shared broadcast link is called mutiple access protocols.
                    - However, when mutiple nodes share a single broadcast link--the so-called multiple access problem. Here, the MAC protocol serves to coordinate the frame transmissions of the many nodes.
        - 

    - Where is the link layer implemented
        - Part of software part of hardware, but most of link-layer functionality is implemented in hardware, the link layer is implemented in a network adapter(网络适配器), also known as a network interface card(NIC, 网卡).
            - Software: funcitons like Identifying the correct MAC address should be received, are implemented in OS(Operation

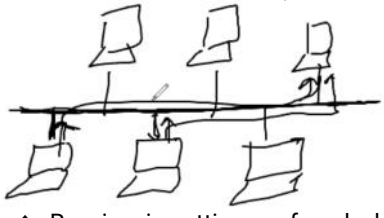
System) as software
    - Hardware: functions like, error bit checking, or decoding algorithm are happened in hardware. The pyhsically layer is also implemented at NIC(Network interface card), e.g. convert the signal to voltage or electromagnetic wave and send out to the wire.



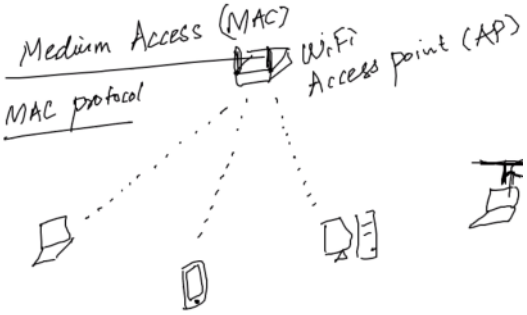
- Adaptors communication



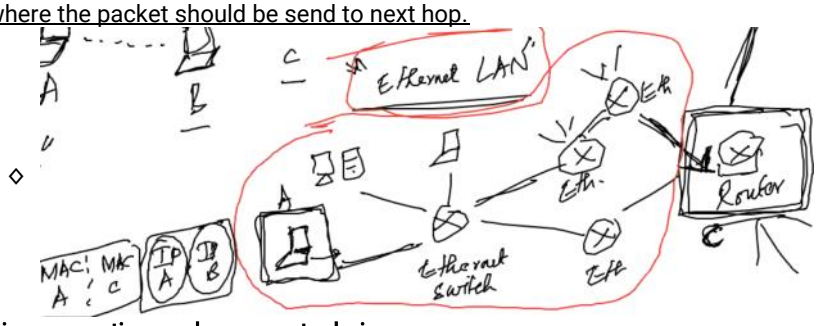
- Sending side:
  - Encapsulate datagram in frame
  - Add error checking bits, reliable data transfer etc.
- Receiving side:
  - Look for errors, reliable data transfer etc.
  - Extract datagram, and pass to upper layer

- What is the responsibility/main functions of link-layer?
  - The link-layer has the responsibility of transferring packet from one nodes/hops to its adjacent node. Links can be between end host and first hop router, or between two IP hops in the core. Different link layer technologies are united by IP. Link layers run on many different physical media. Examples: Ethernet over copper or fiber, wireless, long distance fiber links.
- **Link Layer services:**
  - There are four major Link Layer services: 1)Framing, 2)Link access, 3)error correction, and 4)reliable delivery between adjacent nodes)
    1) **Framing:** encapulates IP datagrame into a frame. When packet goes down from networking layer, a link layer packet(frame) will be added to the IP datagrame.
       - Why do we need frame (link-layer header)?
         - Application --> When packet reached Transport layer, transport layer will encapsulate a TCP header to Data --> Routing layer will add an IP header --> Link layer will add the link layer header to Data, aka Frame
       - What exactly was included in link-layer header?
         - Includes MAC address(Medium access control) -- Src and Dest MAC addr. It's same like routing header includes both Src and Dest Ip addr
       - Why cannot directly convert to voltage, meganetic wave?
         - because if there are multiple user sharing the networking, that might cause the collision!
       - Problem of collision in the past (Connected by ethernet)
         - 
           - Reveiver is getting confused when multiple sharing happens, it cannot distinguish the packet come from which sender.
           - So this should become the job of link layer. It should only allow one people talking at a time, so let's discuss some solution of how to solve this collision problem.
    2) **Link access(连接访问):** If link shared between multiple users(e.g. WiFi link), a MAC(medium acess control) protocal can specifies the rules by which a frame is transmitted onto the link.
       - **MAC(Medium access):** IP address is for router, and MAC is for Identify the end-point devices in link level
       - So, in this case, if multiple user access to a shared communication medium simultaneiously, the MAC protocal could take care this collision problem
       - The shared communication media includes WiFI networks, where EM waes belonging to a certain frequency

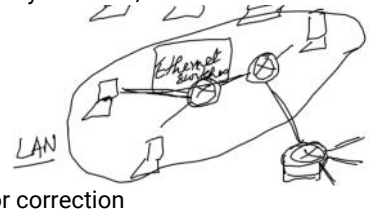
range (2.4 to 2.48 GHz) are reserved for WiFi communication. Regardless of how many users are in a particular room, the same 80 MHz range (2.4–2.48 GHz) must be shared across all these users in some manner.



- How does **frame** was being used to send the packets?
  - The squared one at the right is Router(has all the IP forwarding table). It's the one that has IP forwarding table, and others in the left are Ehternet switches, which is the most popular implementation right now.
  - In link-layer, ethernet switch don't understand the IP address, have no idea about the forwarding table, or even anything about other layer, such as TCP header. So, they only can use the frame(MAC addres) to deliver the packet to the nearest router, and the router will consult it's forwarding table and decided where the packet should be send to next hop.



3) **Error detection, correction, and recovery techniques**
   - Some popular techniques, e.g. parity bits, checksum, or CRC(Cyclic Redundancy Check) to detect bit error in received frame.
   - There are errors caused by signal attenuation, noise. Receiver need to identifies and corrects bit errors without resorting to retransmission.
   - Link layer switch, or Ethernet switch

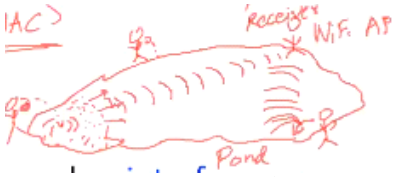


   - Error correction
     - it's different from error detection, e.g. checkum
     - error correction is more complicate task than detection. Error usualy can be caused by signal attenuation, and noise. After the error was detected, the receiver
     - The error correction data has it's limit on how many bit can correct(if has some flip bit error)
     - The more error correction you includes the more robust of you link layer deliver, if more than
       - there is trader off between the number of bit of error correction you includes and the robustness of correction(decoding scheme)
     - Why not inlcudes the error correction data in transport layer?
       i) Because all those lin layer service are implement in hardware
          - network interface card, that connect to your mother chip
       ii) You cannot affor all those heavy cost computation in software.

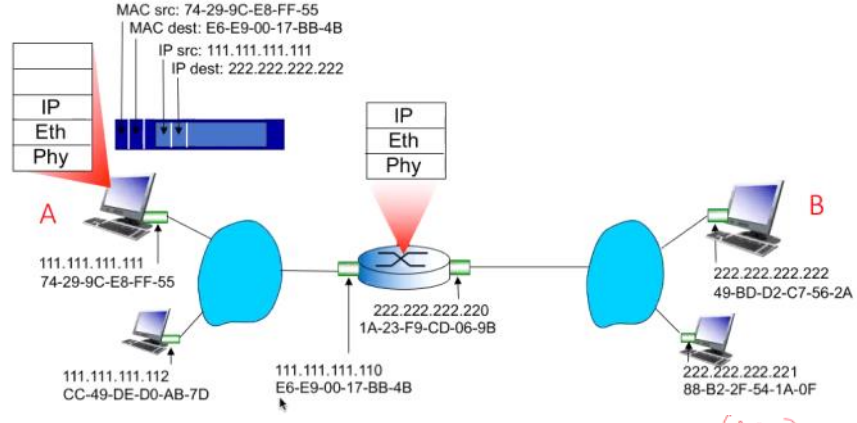
4) **Reliable delivery between adjacent nodes**
   - Why do we need both link-level and end-to-end realibility? If you have the reliability in link-layer, why do you still need the reliability in Transport layer?
     - The link-layer protocol only provide the reliability in a local level, e.g. you can make sure the packet can be transmitted between two router, but you don't have a end-to-end reliability, which is the job of Transport layer to make sure the packet can be transmitted between end-host.
     - e.g What if there is one router along the path have corruption, or break? In this time, if we still have the reliabilty from transport layer (Not routing layer?), we can make a detour to the destination, and avoidng the constantaneous packege dropping!

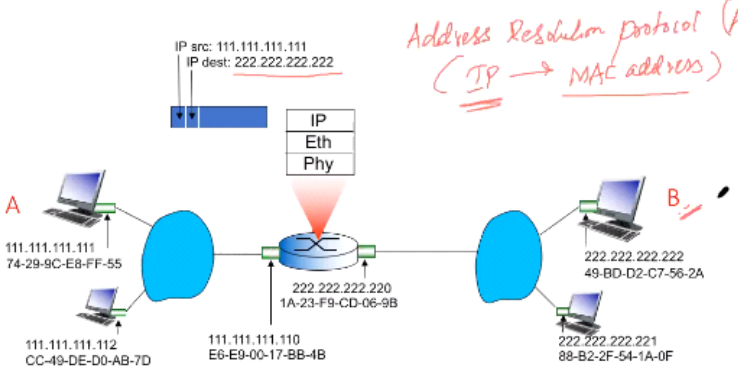
- ◇ 

- **MAC (Multiple access protocols, or Medium Access control protocol)**
  - Motivation:
    - In a home networking, we allow multiple users sending packets to a share WiFi router. When two or more devices/nodes are transmitting simultaneously, the Collision could happen. Therefore, we want to design a mechanism to avoid the collision when multiple devices broadcasting the signal.
      - We are all familiar with the notion of broadcasting--television has been using it since its invention. We are all familiar with the notion of broadcasting-television has been using it since its invention. But traditional television is a one-way broadcast (that is, one fixed node transmitting to many receiving nodes), while nodes on a computer network broadcast channel can both send and receive.
      - Perhaps a more apt human analogy for a broadcast channel is a cocktail party, where many people gather in a large room (the air providing the broadcast medium) to talk and listen. A second good analogy is something many readers will be familiar with-a classroom-where teacher(s) and student(s) similarly share the same, single, broadcast medium.
      - However, there is one central problem. Because we allow all nodes(or end devices) are capable of trasmitting frames at the same time, then all of the nodes can receive multiple grames at the same time as well, and that's when collision happens. When there is a collision, the signals of the colliding frames become inextricably tangled together, and the broadcase channel is wasted, like the interference of ripple in a pond.

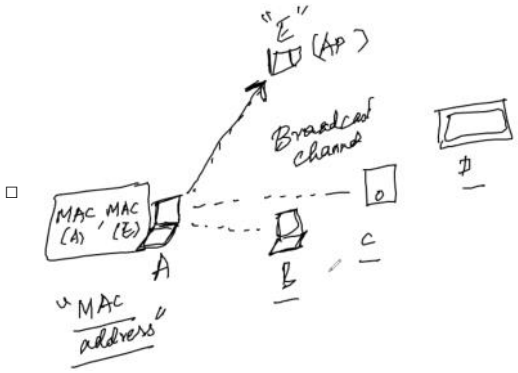
  - Understand the problem so far:
    - Single shared broadcast channel. e.g. Multiple usre sending the packet to a shared WiFi router.
      - When the receiver listening the signal from two device in a same time, there may have some overlapping and may confuse the receiver. Consider the example of a pond:
        - 

    - So we need a distributed algorithm(no master node/administrator devices) that determines how the channel is shared by nodes, or when nodes can/cannot transmit packets/frame.
    - Communication about channel sharing must use channel itself!
      - No out-of-band channel for coordination
        - e.g. your lap must use the home-Wifi network for coordination
      - In-band Vs Out-of-band communication
        - **"in-band" communication**: All devices are talking/communicating within a same netowork provider.
        - **"out-of-band" communication**: The devices are communicate across different network ISP, e.g. one in China, other end is in US
          - ◇ It means that the communication of any control messages between the nodes, that may be required in the MAC protocol, must happen over the network itself and not through a separate communication link(network ISP). This is called "in-band" communication. If the nodes are in separate location (service provided by differnet ISP), dedicated communication links available to exchange the MAC protocol's control messages it is called "out-of-band" communication.
          - ◇ E.g., in the Aloha protocol when a collision happens, the receiver communicates to the senders that a collision has happened through the same network. This is in-band communication.
          - ◇ Instead, if the receiver had separate low-capacity wired links to the senders just for sending such control information, that would be called out-of-band communication.
  - Why do we need both IP addr and MAC addr?
    - IP addrs is for networking layer, and MAC addr is for Link layer, and link layer shouldn't know anything about networking layer or IP address.

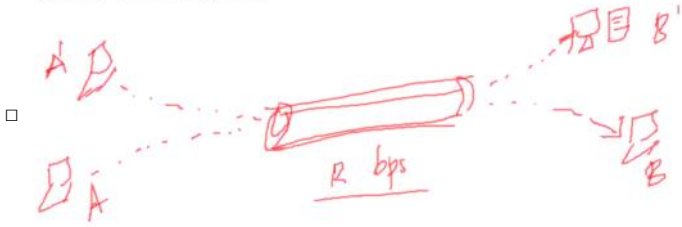
        - The networking layer is used to determing the S/D between two host/router), not for the Link-layer. And, you do want to have a decentralized mechanism(no master node/administrator devices), such that the link-layer can identify the device without depending on other layer protocol.
        - Consider the encapsulation mechanism, in link-layer, ethernet switchs have no idea about the forwarding table, or even anything about other layer, such as TCP, or HTML protocol. So, they only thing can use to deliver the packet to the nearest router is the frame(MAC addres).
            - Note1: remember the functions of link layer is to transport the IP datagram between two IP hops, it can be either between end-host and first hop router, or between two IP hops in the core
            - Note2: Also, Link layer has no idea about the IP addr, consider it as the packet you want to deliver, so don't open it!!
- Property:
    - MAC adddresss are unique and fixed across all adapters (like SSN).
    - MAC addresses are 48 bits long.
        - One interesting property of MAC addresses is that no two adapters have the same address. How does a company manufacturing adapters in Taiwan make sure that it is using different addresses from a company manufacturing adapters in Belgium? The answer is that the IEEE manages the MAC address space. In particular, when a company wants to manufacture adapters, it purchases a chunk of the address space consisting of 224 addresses for a nominal fee. An adapter's MAC address is analogous to a person's social security number, and an IP address is analogous to a person's postal address. As a person moves, his postol adddr will changes but his security number remain fixed permanently. Just as a person may find it useful to have both a postal address and a social security number, it is useful for a host and router interfaces to have both a network-layer address and a MAC address.
- Implementation:
    - Everytime we want to pass througth this "Giant blue cloud",network switch, the link-layer will resolve the MAC address and IP address, and decide the the next hop to go.
    - E.g. If node A want to send a message to node B, node A will be the starting point, and router R will be the next hop, and node B will be the final stop. **In link-layer**, node A will encapsulate the Link-layer frame with src/dest MAC addrress. When the packet reached the router R (via broadcast, the IP datagrame will be broadcast in the channel), it will decapsulate the frame. Now, we are in routing-layer. Router will consult its forwarding table with the src/dest IP addr and determine where is the next hop the packet should be send to. In the sender interface, router R will also encapsulate a new src/desc MAC address(but src/dest IP addr don't changes), and so on and so forth, until the packet reached destination.
        - 

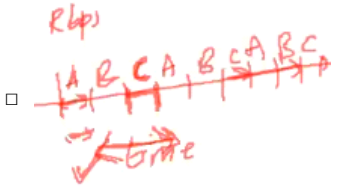
        - 

- How does router know which device is the one that A wanna send to? Or How MAC is being used in this example? （比如说，在这个broadcast environment，我怎么知道这个信号是不是发给我的？其他的device也可以收到吗?)
    - Each devices have a unique MAC addr, and when a node send out its link-layer frame, each adjacent nodes can compare its MAC addr and the Dest Addr that nodes send out. If they match, the packet will be received, otherwise, the packet will be dropped.

- 

  - Note: In MAC addr routing, only the end deviced has MAC address but switch don't, and the switch don't have the concept of "forwarding table". So how does switch know where the frame should be send to? ==> With the help of ARP(Address resolution protocol), which don't cover in this class.

- **4/3 Slide 20: Data Link-layer cont...**
  - Goals/Ideal properties/characteristics we want to achived
    1. One node can transmit at a throughput/rate of R bps.
    2. M nodes can transmit at average rate of R/M
    3. The protocol is decentralized; that is, there is no master node that represents a single point of failure for the network. (E.g. No special node to coordinate the transmission; No synchronization of clocks, slot)
    4. The protocol is simple to implement
       - Note: Those goals are pretty similar to the TCP
  - 

  - There are dozons of MAC protocol, but we can classify them to one of Three categories:
    a. **Channel partitioning protocol (e.g. TDMA, FDMA)**
       - Devide channel into smaller "pieces""
       - Allocate piece to node for exclusive use, kinda like circuit switching
         - So, no matter if a node has a packet to send or not, the channel will preserved this time slot for it.
         - 

    b. **Random access protocol (e.g ALOHA, CDMA/DC)**
       - Channel not divided, it allow collision.
         - When a node has something wanna to send, it doesn't care whether has something to send, instead, it will just go ahead and transmit the packet.
       - "recover" from collision
         - It allows the collision to happen, but it has way to detect it and recover from collision.
         - When there is a collision, each node involved in the collision repeatedly retransmits its frame (that is, packet) until its frame gets through without a collision. But it doesn't necessarily retransmit the frame right away. Instead it waits a random delay before retransmitting the frame.
         - Because the random delays are independently chosen, it is possible that one of the nodes will pick a delay that is sufficiently less than the delays of the other colliding nodes and will therefore be able to sneak its frame into the channel without a collision.
    c. **"taking turns" protocol (e.g. polling, token-passing)**
       - Nodes take turns, but nodes with more to send can take longer turns

4/6 Slide 20 cont..: Three categories of MAC address protocol

- **Channel partitioning:**
  - **TDMA(Time division multiple access)**:
    - **Description:**
      - TDM divides time into time frames and further divides each **time frame** into N time slots. (The TDM time frame should not be confused with the link-layer unit of data exchanged between sending and receiving

adapters, which is also called a frame. In order to reduce confusion, in this subsection we'll refer to the link-layer unit of data exchanged as a packet.) Each time slot is then assigned to one of the N nodes. Whenever a node has a packet to send, it transmits the packet's bits during its assigned time slot in the revolving TDM frame. Typically, slot sizes are chosen so that a single packet can be transmitted during a fixed time slot. Figure 6.9 shows a simple four-node TDM example. Returning to our cocktail party analogy, a TDM-regulated cocktail party would allow one partygoer to speak for a fixed period of time, then allow another partygoer to speak for the same amount of time, and so on. Once everyone had had a chance to talk, the pattern would repeat.

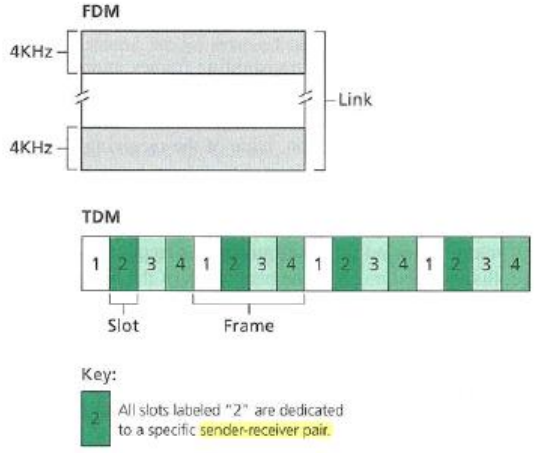
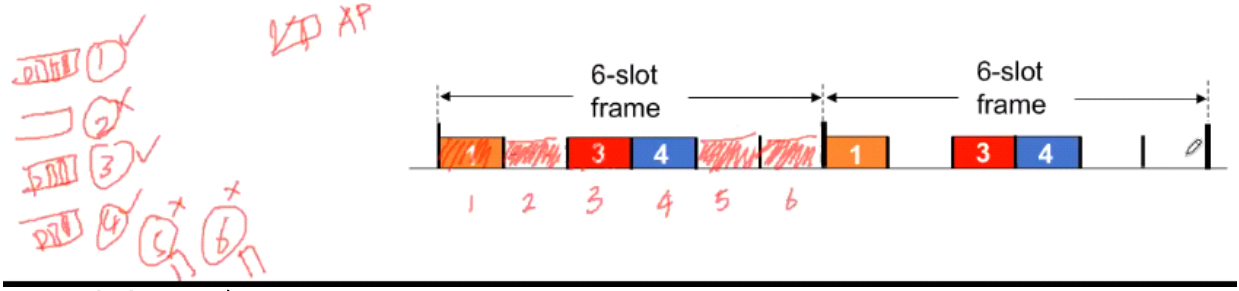


Figure 6.9 ♦ A four-node TDM and FDM example

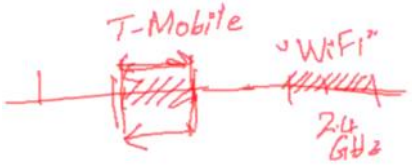
- **Good**:
  1) Meet MAC property
  2) Eliminates collisions and is perfectly fair: Each node gets a dedicated transmission rate of R/N bps during each frame time.
- **Bad**:
  - First, a node is limited to an average rate of R/N bps even when it is the only node with packets to send
  - A second drawback is that a node must always wait for its turn in the transmission sequence-again, even when it is the only node with a packet to send. Imagine the partygoer who is the only one with anything to say (and imagine that this is the even rarer circumstance where everyone wants to hear what that one person has to say). Clearly, TDM would be a poor choice for a multiple access protocol for this particular party.
- Example 02:
  - E.g. In a particular case of 6-station LAN, 1,3,4 have packet so send, but slot 2,5, have no packet to send, so they are idle. So, another ananlogy is like, there are 6 device connect to the Home-WiFI network, and only user 1,3,4 are downloading some movie, so they need to use the channel, but the user 2, 5, 6 just don't have packet to send, so the channel just being idled.
    - It's possible that the utilization rate can be 100%, for example of left side panel:
    - If the traffic is not busy, many channel will remain idel, so the utilization can be bad.
    - TDMA requires the synchronization of clock, so before the transmission, every nodes/sender need to make aggrement on when is the time-slot begin and when ends, to ensure that no collision or big lagging would happen!



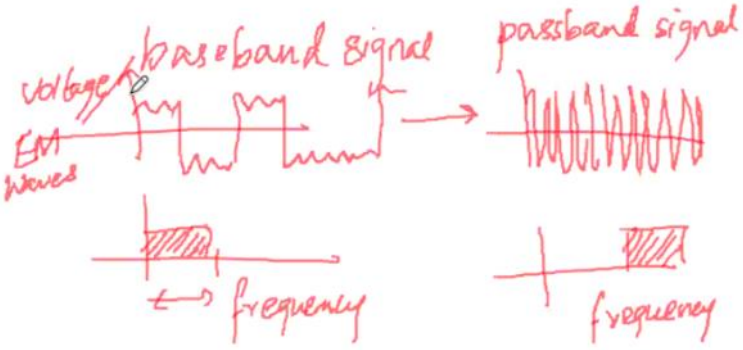
- **FDMA(Frequency division multiple access)**:
  - Desctiption:
    - FDM divides the R bps channel into different frequencies (each with a bandwidth of R/N) and assigns each frequency to one of the N nodes. FDM thus creates N smaller channels of R/N bps out of the single, larger R bps channel. FDM shares both the advantages and drawbacks of TDM. It avoids collisions and divides the bandwidth fairly among the N nodes. However, FDM also shares a principal disadvantage with TDM-a node is limited to a bandwidth of R/N, even when it is the only node with packets to send.
  - Motivation:
    - TDMA ends up wasting slots. TDMA can be modified to skip slots for users that do not have data waiting to be transmitted, but this ends up complicating TDMA considerably. Another simple mechanism to arbitrate

access to a shared medium is Frequency Division Multiple Access (FDMA), where each user is given a slice of the frequency range (or bandwidth). Such a slice is also called a channel.

- **Idea(not on exam, just background information):**
  - each users/nodes own a specific bandwidth, there is no scheduling across time, so the user who owns this frequency band can send data anytime he want! E.g. The bandwidth that used for LTE 4G is about 800MHZ, but the Wifi router for your home device may be like 2.4GHZ. And, different ISP controls different bandwidth of frequency.
  - Why do we want to change the bandwidth? Is it for collision control? Why not just sending the signal in base band?
    - For collision control. e.g. ATT, T-mobile, or Verizon user have different frequency, If all of these different sellers or providers are using the same frequency band. When the signals from these different providers are all going to interfere with each other, so you always want to shift the base band into disjointed band freuqnecy, to make sure no overlapping.
    - For modulate between different transmission medias. E.g. Cooper act as a good conductor for specific range of frequency, so any range freuqncy smaller than that, the signal can be seriously attenuated. The size of the antennas that you need to transmit a signal is proportional is going to be proportional to the wavelengths of the signal. So, it you have a very low frequency, then the wavelength of signal can be pretty high, and that means you need a huge antenna, which is not desirable.
  - divide the channel for different frequency bandwidth, e.g. Wift--2.4 GHz, celluer -- 800 MHz…
    - The Wifi in you home using a small range of frequency, and it share though different user, so it can has more power than satellite.

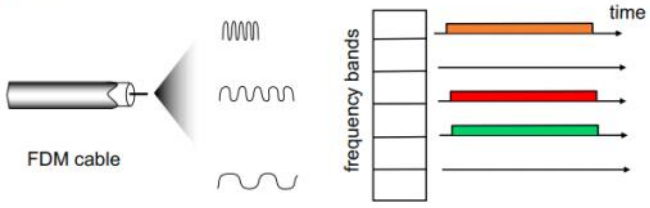


      - The frequency defines the range of signal that a user can send, the actual signal is like that step function, or sinusoidal wave form(at left top plot). And, this is done via Fourier transformation.



- Good:
  - It avoids collision and divides the bandwidth fairly among the N nodes.
- Bad:
  - a node is limited to a bandwidth of R/N, even when it's the noly node with packets to send.
  - if some channel has no packet to send, such frequency bands idle and get wasted

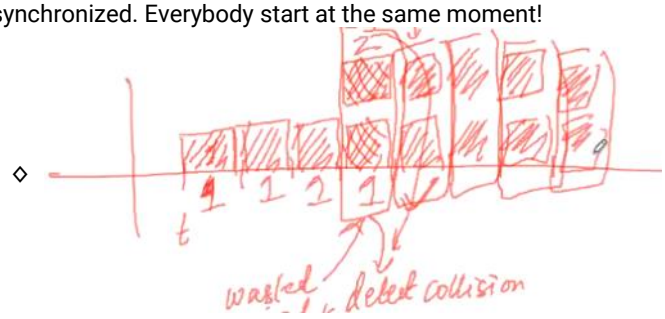
Example: 6-station LAN. 1, 3, 4 have packets to send, frequency bands 2, 5, 6 idle



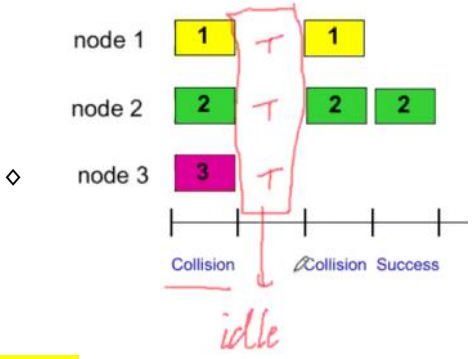
- **Random access protocol**
  - How does the collision is being detected?
  - How does it being recovered?
  - **ALOHA net (阿罗哈) – Slotted ALOHA**
    - Description:
      - unslotted ALOHA prorotocl is a fully decentralized protocol. and it requires all nodes synchronize tranmission at the beginning of a slot
      - In both slotted and pure ALOHA, a node's decision to transmit is made independently of the activity of the other nodes attached to the broadcast channel. In particular, a node neither pays attention to whether another

node happens to be transmitting when it begins to transmit, nor stops transmitting if another node begins to interfere with its transmission.

- Example:
  In our cocktail party analogy, ALOHA protocols are quite like a boorish partygoer who continues to chatter away regardless of whether other people are talking. As humans, we have human protocols that allow us not only to behave with more civility, but also to decrease the amount of time spent "colliding" with each other in conversation and, consequently, to increase the amount of data we exchange in our conversations.
- Assumption:
  - All frames consist of exactly L bits.
  - Time is divided into slots of size L/R seconds (that is, a slot equals the time to transmit one frame).
  - Nodes start to transmit frames only at the beginnings of slots.
  - The nodes are synchronized so that each node knows when the slots begin.
  - If two or more frames collide in a slot, then all the nodes detect the collision event before the slot ends.
- **Operation:**
  - If node has a fresh frame, trainsmit entire frame in next slot.
  - If no collision, a frame is trainsmitted succssfully, and the node can prepare a new frame for transmission!
  - If there is a collision, retransmit the frame with probability p in each subsequent slot until success.
    - It's like each user will "flip a coin" independently, and use it to decide whether to send the packet at next time slot
    - In this case, we have a collisio at slot 4, so the entire frame is wasted, but the probability of having a sequence of collision will be very small.
    - Frame == channel? ==> Frame is the link-layer unit of data exchanged between sender and receiver, which defines the length of data we can send. Channel is the pipeline?
    - Could we do a first come first server? I mean when there is a colision happen. ==> The time is synchronized. Everybody start at the same moment!



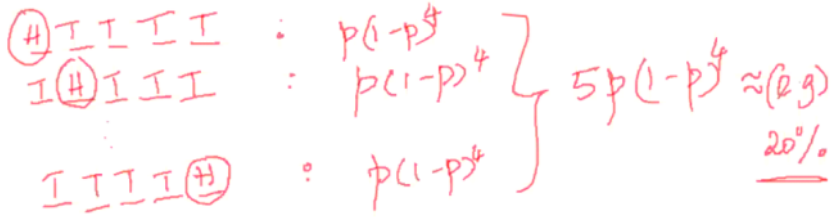
- **Good/Pros:**
  - Single active nodes(A node is said to be active if it has frames to send) can continuously transmit at full rate of channel.
  - It's Decenralized: because each node detects collisions and independently decides when to retransmit. (Slotted ALOHA does however, require the slots to be synchronized in the nodes;)
  - Simple to implement
- **Bad/Cons/Drawback:**
  - Collision waste the time-slots: Slotted ALOHA works well when there is only one active node, but the efficiency is bad when there are mulitple active nodes. Two possible efficiency concerns here:
    1. **Collision wasted slot:** The fist and third slot is being wasted because of collision happen! (If time parameter is too small, you might has a lot of idles slot).



    2. **Idel slot:** The second slot is empty because of the probabilistic transmission policy.
  - Instead of stop transmitting the packet while collision detected, it still choose to send the packet, but nodes may be able to detection collision in less than time to transmit packet
  - Requires Clock synchronization
- **ALOHA efficientcy**
  - Probability that a given nodes transmits is p; the probability that the remaining nodes don't transmit is (1-p)^N-1 ==> the probability that a given nodes has a successful slot is p * (1-p)^N-1. ==> Because there are N

nodes, the probability that any one of the N nodes has a successful slot is = N * p * (1-p)^N-1,

- Can be use to define average utilization of link: In this case, only 20% of change the transmission will be success



  - At best: we can have 37% transmission utilization. Thus the maximum efficiency transmission rate of a channel is not R bps but 37% R bps!

- **CSMA(Carrier sense multiple access)**
  - Two important rules for polite conversation:
    1) **Listen before speaking**. If someone else is speaking, wait until they are finished. In the networking world, this is called carrier sensing-a node listens to the channel before transmitting. If a frame from another node is currently being transmitted into the channel, a node then waits until it detects no transmissions for a short amount of time and then begins transmission.
    2) If someone else begins talking at the same time, stop talking. In the networking world, this is called **collision detection**- a transmitting node listens to the channel while it is transmitting. If it detects that another node is transmitting an interfering frame, it stops transmitting and waits a random amount of time before repeating the sense-and-transmit-when-idle cycle.
  - Q: do collisions occur in the first place?
    - After all, a node will refrain from transmitting whenever it senses that another node is transmitting. The answer to the question can best be illustrated using space-time diagrams:
    - At time t0, node B senses the channel is idle, as no other nodes are currently transmitting. Node B thus begins transmitting at t0, with its bits propagating in both directions along the broadcast medium. The downward propagation of B's bits in Figure 6.12 with increasing time indicates that a nonzero amount of time is needed for B's bits actually to propagate (albeit at near the speed of light) along the broadcast medium. At time t1 (t1 > t0), node D has a frame to send. Although node B is currently transmitting at time t1, the bits being transmitted by B have not reach to D, and thus D senses the channel idle at t1. In accordance with the CSMA protocol, D thus begins transmitting its frame. A short time later, B's transmission begins to interfere with D's transmission at D. From Figure 6.12, it is evident that the **end-to-end channel propagation delay** of a broadcast channel -- the time it takes for a signal to propagate from one of the nodes to another -- will play a crucial role in determining its performance. The longer this propagation delay, the larger the chance that a carrier-sensing node is not yet able to sense a transmission that has already begun at another node in the network.

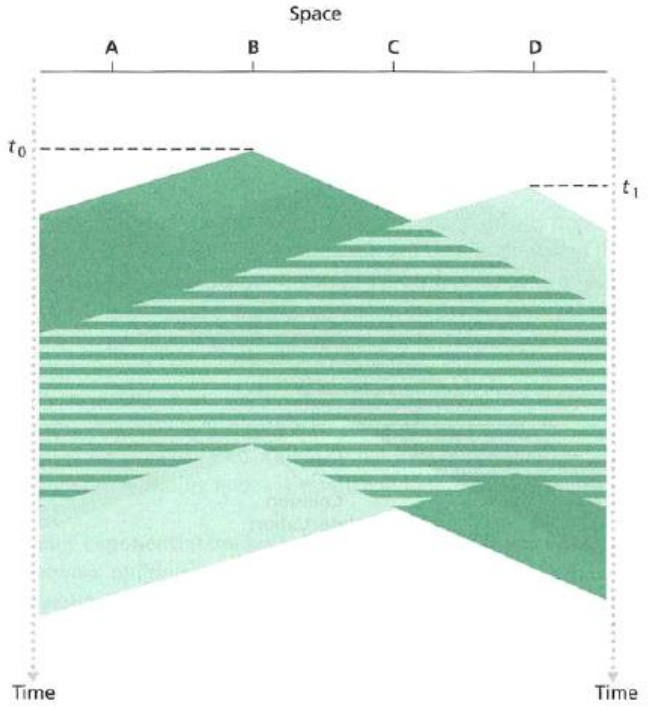


Figure 6.12 ♦ Space-time diagram of two CSMA nodes with colliding transmissions

- **CSMA/CD (Carrier Sense Multiple Access with Collision Dection) – Used in Ethernet**
  - Motivation:

- CSMA do not perform collision detection; both B and D continue to transmit their frames in their entirety even though a collision has occurred. When a node performs collision detection, it ceases transmission as soon as it detects a collision. Clearly, adding collision detection to a multiple access protocol will help protocol performance by not transmitting a useless, damaged (by interference with a frame from another node) frame in its entirety.

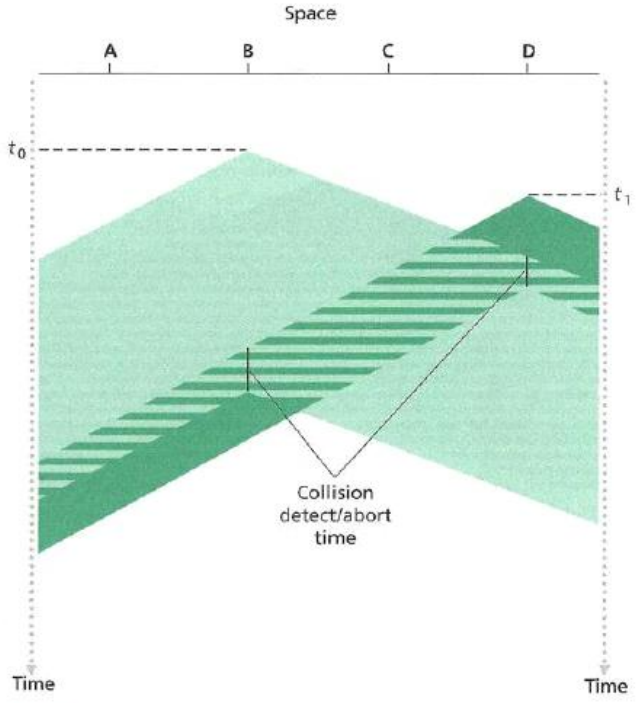
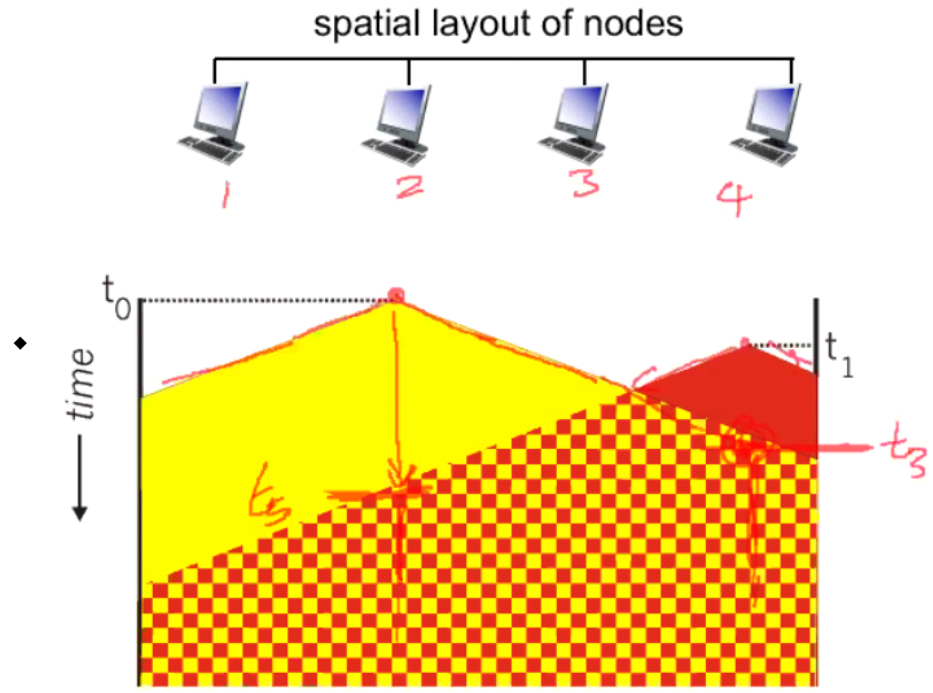


Figure 6.13 ♦ CSMA with collision detection

- **CSMA/CD Algorithm/Operation:**
  1) After receivied a datagram from network layer, prepare link-layer frame and put into the buffer
  2) If channel is busy, then wait until it sense no signal being transmitted and then trasmits the packet/frame.
  3) While transmitting, keep monitoring the signal energy in the broadcast channel.
  4) if no collision detected, the frame transmitted successfully; otherwise abort transmission and send/broadcast jam signal.
  5) After aborting, enters binary exponential backoff (wait a random amount of time) and return to step 2:
     - After m-th collision, NIC chooses K at random from {0, 1, …, 2m-1} and waits K*512 bit times and returns to step 2 (e.g., 5.12 microseconds for a 100 Mbps Ethernet).
- **Binary exponential backoff algoirthm(二进制指数后退算法).**
  - if two nodes transmitted frames at the same time and then both waited the same fixed amount of time, they 'd continue colliding forever. What we'd like is an interval that is short when the number of colliding nodes is small, and longer when the number of colliding nodes is large. This problem is solved elegantly with Binary exponential backoff algoirthm.
  - Specifically, when transmitting a frame that has already experienced n collisions, a node chooses the value of Kat random from {0,1 ,2, ... 2"-1 }. Thus, the more collisions experienced by a frame, the larger the interval from which K is chosen. For Ethernet, the actual amount of time a node waits is K · 512 bit times (i.e., K times the amount of time needed to send 512 bits into the Ethernet) and the maximum value that n can take is capped at I 0.
  - For example: If the node chooses K = 0, then it immediately begins sensing the channel. If the node chooses K = I , it waits 512 bit times (e.g., 5.12 microseconds for a 100 Mbps Ethernet) before beginning the sense-and-transmit-when-idle cycle. After a second collision, K is chosen with equal probability from {0,1,2,3}. After three collisions, K is chosen with equal probability from {0,1,2,3,4,5,6,7}. After 10 or more collisions, K is chosen with equal probability from {0,1 ,2, ... , 1023}. Thus, the size of the sets from which K is chosen grows exponentially with the number of collisions; for this reason this algorithm is referred to as binary exponential backoff.

- Could the collision still happen?
  - Yes, the propagation delay still exist, it's possible when one send the packet, but other's not listen and is in the middlle of packet transmission.. Because of propagation delay, the packet take time to transmit. As a result the entire packet transmission time might get wasted!
  - So CSMA can minimize the collision but not eliminate it. The longer this propagation delay, the larger the chance that a carrier-sensing node is not yet able to sense a transmission that has already begun at another node in the network. Therefore, shorter the propagation delay, better the utilization for CSMA:
    - In this example, at t1, nodes 4 sense no packet being transmitted, so he start to send the packets. However,

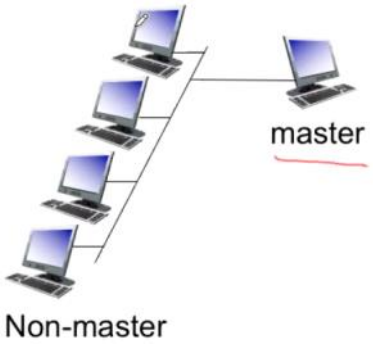
in fact, nodes 2 had transmitted a nodes at t0, but it takes some time to have nodes 4 get notified until t3, so a collision happened here, and whole frame from both nodes get dropped/wasted!



- Pros:
  - Collision can be detected within short time, so if the collision detected (By using the magnitude of overlapping signal), we will choose the stop transmit the packet!
  - When only one node has a frame to send, the node can transmit at the full channel rate (e.g. , for Ethernet typical rates are lO Mbps, 100 Mbps, or 1 Gbps).
- **What is carreier sensing?**
  - It listen before transmition. If it sense the channel is being idle, start transmit the entire frame; Otherwise we defer transmission until sense no signal energy and then transmit. (A nodes cannot read the information at channel, but we can detect whether the channel is being used or not.)

- ALOHA Vs CSMA with Detection
  - After a collision detected; ALOHA throws the randomness in choosing whether to send a frame or not, and CSMA/CD use a randomized interval in deciding the waiting time to try again! (Something associate with Binary exponenetial backoff)
  - the probability p is fixed in ALOHA, but the randomized waiting time in CSMA/CD is chosen dynamically, like previous window size.
- **Taking-Turn protocols**
  - Motivation:

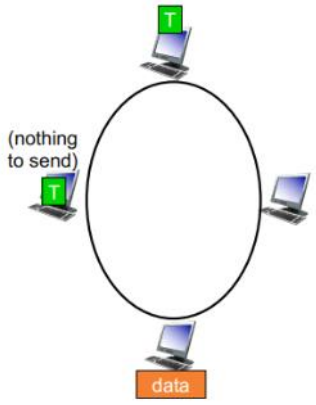
    Recall that two desirable properties of a multiple access protocol are (I) when only one node is active, the active node has a throughput of R bps, and (2) when M nodes are active, then each active node has a throughput of nearly R/M bps. The ALOHA and CSMA protocols have this first property but not the second.
  - Two important Taking-Turn protocols:
    - **Polling protocol**
      - Description:
        - The polling protocol requires one of the nodes (can be anyone) to be designated as a master node. The master node polls each of the nodes in a round-robin fashion. In particular, the master node first sends a message to node 1, saying that it (node 1) can transmit up to some maximum number of frames. After node 1 transmits some frames, the master node tells node 2 it (node 2) can transmit up to the maximum number of frames. (The master node can determine when a node has finished sending its frames by observing the lack of a signal on the channel.) The procedure continues in this manner, with the master node polling each of the nodes in a cyclic manner. One real world application is Bluetooth.
        - If a nodes has a data to send, master node will poll the frame from nodes and start the transmission.
        - If a nodes had no data to send, master node will skip it and move to next nodes.

◇ 

- Advantage:
  1. Eliminates the collisions and empty slots that plague random access protocols.
  2. Achieve a much higher efficiency than random access protocols
  3. Meet the second property of MAC protocol
- Drawback:
  - The first drawback is that the protocol introduces a **polling delay/overhead** -- the amount of time/resources required to notify a node that it can transmit. If, for example, only one node is active, then the node will transmit at a rate Jess than R bps, as the master node must poll each of the inactive-nodes in turn each time the active node has sent its maximum number of frames.
  - The second drawback is potentially more serious. If the master node fails, the entire channel becomes inoperative.

▪ **Token-passing protocol**

- Description:
  - In this protocol there is no master node. A small, special-purpose frame known as a token is exchanged among the nodes in some fixed order.

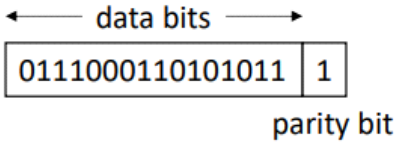
◇ 

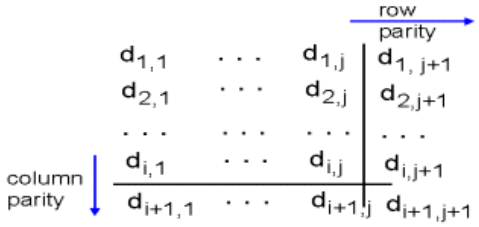
- Example:
  - For example, node l might always send the token to node 2, node 2 might always send the token to node 3, and node N might always send the token to node l. When a node receives a token, it holds onto the token only if it has some frames to transmit; otherwise, it immediately forwards the token to the next node. If a node does have frames to transmit when it receives the token, it sends up to a maximum number of frames and then forwards the token to the next node.
- Advantage:
  - **Decentralized** and highly **efficient**.
  - Barely Meet the first and second property of MAC protocol
- Drawback:
  - The failure of one node can crash the entire channel.
  - If a node accidentally neglects to release the token, then some recovery procedure must be invoked to get the token back in circulation.
  - Token delay/overhead -- The time/resources required to pass over the token.

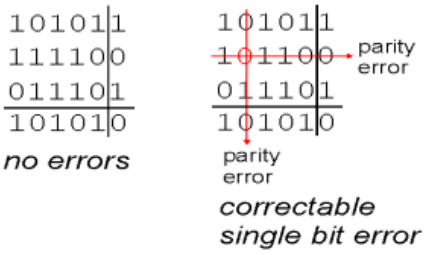
○ Note: Collision detection in wireless is difficult, because the power of signal is weak, so we do collision avoidance in wireless(e.g. WIFi), it's more like proactive scheme. So, after you transmit a packet, you waiting a random amount of time before transmitting next packet.

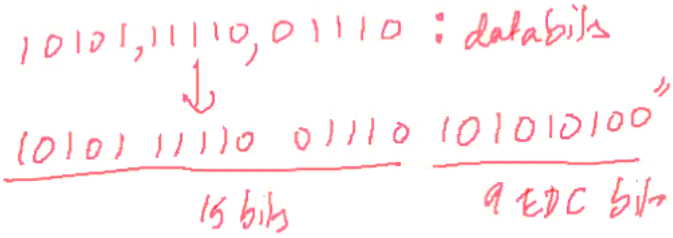
- **Error detecion and correction**
  - Description:
    - Error can be caused by signal attenuation, noise, or interference
    - Error detection: detect whether bits have dlipped within packet
    - Error correction: automatically correctly flip the bits without requiring retransmission, e.g. LDPC)low-density parity check code
    - Note: the research at error detection/correction had been study over 70-80, so the currection application is really

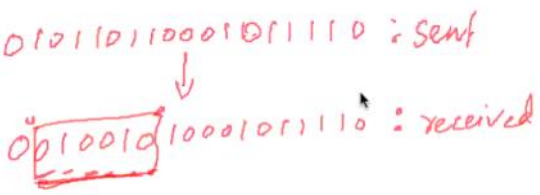
the state-of-art, very mathmatically determination.

- EDC (Error detection and correction)
  - Idea: append "redundant" bits to frame, for detection purpose. Generally, those techniques are not 100% reliable, but greater the redundancy greater the reliability.
- Single bit parity:
  - x = the number of 1 in data bits
    - If x%2 != 0 ==> parity bit = 1;
    - Otherwise, parity bit = 0;
  - Example:
    - 

- **Two-dimensional parity-check code:**
  - Introduction: A multidimensional parity-check code (MDPC) is a simple type of error correcting code that operates by arranging the message into a multidimensional grid, and calculating a parity digit for each row and column. The two-dimensional parity-check code, usually called the optimal rectangular code, is the most popular form of MDPC.
  - Example:
    - 

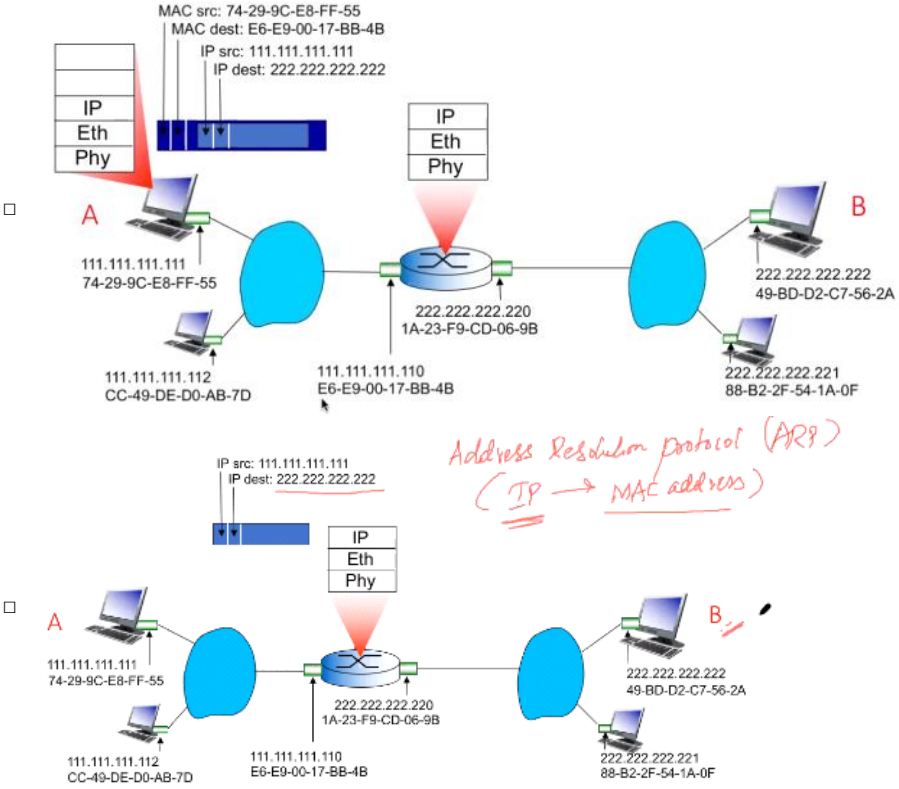
    - 

    - 

  - Drawback:
    - Can detect and correct single-bit error, but, for double-bit or multiple-bit error, only able to detect the error.
- **CRC (Cyclic Redundancy Check)**
  - D: data bits, as a binary number
  - r: is chosen arbitrary, defines the number of bit in G. The larger the r the more reliable the transmission
  - G: generator code, generated by random generator, size(G) = r +1
  - Goal: choose r CRC bits, such that
    - <D, R> is exactly divisible by G (modulo 2)
    - Receiver knows G, divides <D, R> by G, if non-zero remainder, then error detected!
    - Can detect all burst error less than r+1 bits
      - burst error means the a long sequence of error bit.
        - 

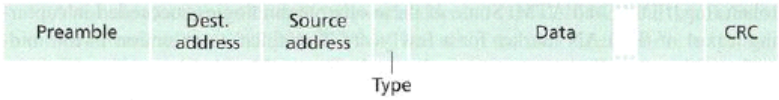
        - Example of long division:
            - Note: When you doing this long division, since the "modulo 2" mechanism, instead of doing the regumar mathmetic subtraction, we do the "modulo 2 version subtraction", which is XOR operation. Example of fifth line, 1010 - 1001 =0011, so we are actually doing the XOR operation.
- MAC Addresses
    - Description:

      One interesting property of MAC addresses is that no two adapters have the same address. How does a company manufacturing adapters in Taiwan make sure that it is using different addresses from a company manufacturing adapters in Belgium? The answer is that the IEEE manages the MAC address space. In particular, when a company wants to manufacture adapters, it purchases a chunk of the address space consisting of 2^24 addresses for a nominal fee. An adapter's MAC address is analogous to a person's social security number, and an IP address is analogous to a person's postal address. As a person moves, his postol adddr will changes but his security number remain fixed permanently. Just as a person may find it useful to have both a postal address and a social security number, it is useful for a host and router interfaces to have both a network-layer address and a MAC address.
    - Implementation:
        - Everytime we want to pass througth this "Giant blue cloud",network switch, the link-layer will resolve the MAC address and IP address, and decide the the next hop to go.
        - E.g. If node A want to send a message to node B, node A will be the starting point, and router R will be the next hop, and node B will be the final stop. In link-layer, node A will encapsulate the Link-layer frame with src/dest MAC addrress, when the packet reached the router R, it will decapsulate the frame, and we are in routing-layer. Router will use the src/dest IP addr to determine where is the next hop the packet should be send to. In the sender interface, router R will also encapsulate a new src/desc MAC address(but src/dest IP addr don't changes), and so on and so forth, until the packet reached destination.

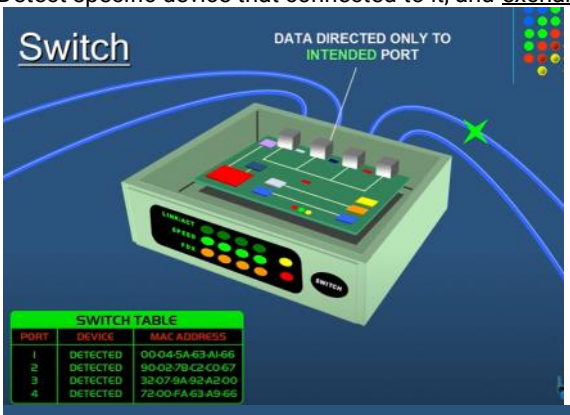


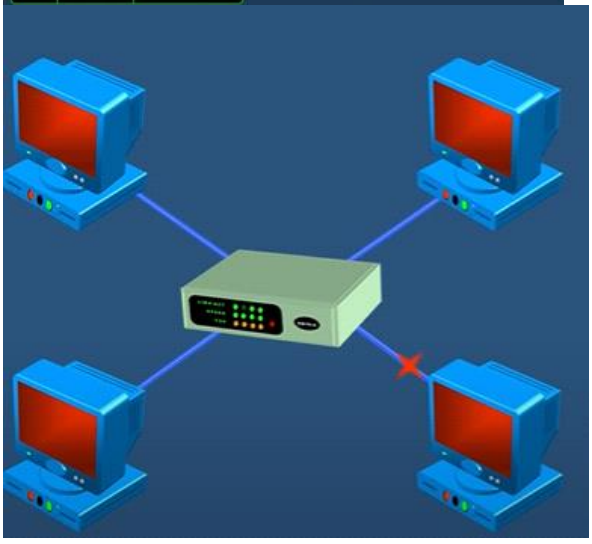
        - What is differnet bettween the IP addr routing and MAC addr routing?
            - In MAC addr routing, only the end deviced has MAC address but switch don't, and the switch don't have the concept of "forwarding table". So how does switch know where the frame should be send to? With the help of ARP(Address resolution protocol)
- Ethernet
    - Motivation: implement based on switch based, so multiple user sharing is not today's network problem.
        - Ethernet has pretty much taken over the wired LAN market. Today, Ethernet is by far the most prevalent wired LAN technology, and it is likely to remain so for the foreseeable future.
          There are many reasons for Ethernet's success. **First**, Ethernet was the first widely deployed high-speed LAN. Because it was deployed early, network administrators became intimately familiar with Ethernet-its wonders and its quirks-and were reluctant to switch over to other LAN technologies when they came on the scene. **Second**, token ring, FDDI, and ATM were more complex and expensive than Ethernet, which further discouraged network administrators from switching over. **Third**, the most compelling reason to switch to another LAN technology (such as FDDI or ATM) was usually the higher data rate of the new technology; however, Ethernet always fought back, producing versions that operated at equal data rates or higher. Switched Ethernet was also introduced in the early

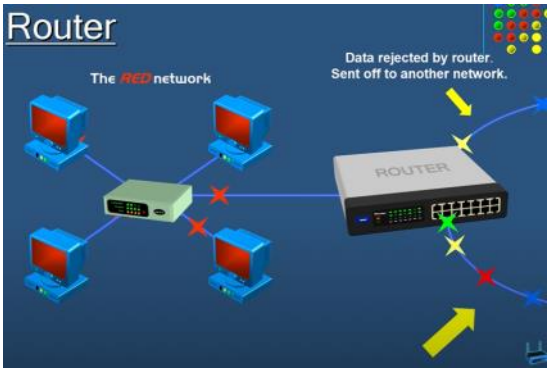
1990s, which further increased its effective data rates. **Finally**, because Ethernet has been so popular, Ethernet hardware (in particular, adapters and switches) has become a commodity and is remarkably cheap.

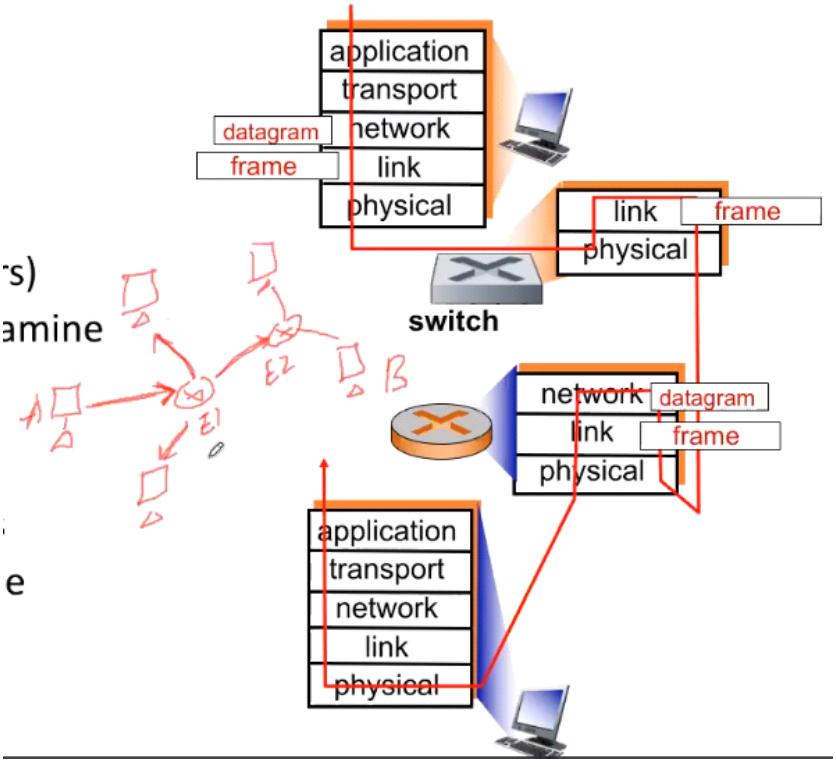
- Ethernet frame structure
  - 

    - Preamble:
      - Total 8 byte for preamble, the first 7 bytes with pattern 10101010 followed by one byte with pattern 10101011 (The last one byte is for syncrhonization)
      - Used to synchronize receiver, sender clock rates
        - The Ethernet frame begins with an 8-byte preamble field. Each of the first 7 bytes of the preamble has a value of 10101010; the last byte is 10101011. The first 7 bytes of the preamble serve to "wake up" the receiving adapters and to synchronize their clocks to that of the sender's clock.
      - Why do we need the synchronization?
        - Because the adapter at receiver side and sender side has different frequencies, so the frame might not be transmitted at exactly the target rate, so we need to synchonrize their clock.

- Ethernet switch (similar to the mechanism of router)
  - Link-layer device
    - Store, forward Ethernet frames
    - Examines incoming frame's MAC address, and forward frame to appropriate outgoing link
    - Can buffer packets
  - Forwarding table
    - Each switch has a switch table for forwarding packets
    - Each entry of table ha
  - Motivation: switch is actually what is being used in today modern network.
- **Switch Vs Routers (Trade of between simplicity and performance)**
  - Switch:
    - Used to exchange data only within a Local area network, not outside the network(Outide the network need to be able to read IP address).
    - Can Detect specific device that connected to it, and exchange Data directed only to intended port.
      - 

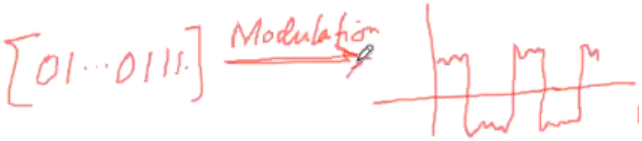
      - 
  - Router:
    - Router are used to connect network, and allow to exchange data for different local area network.

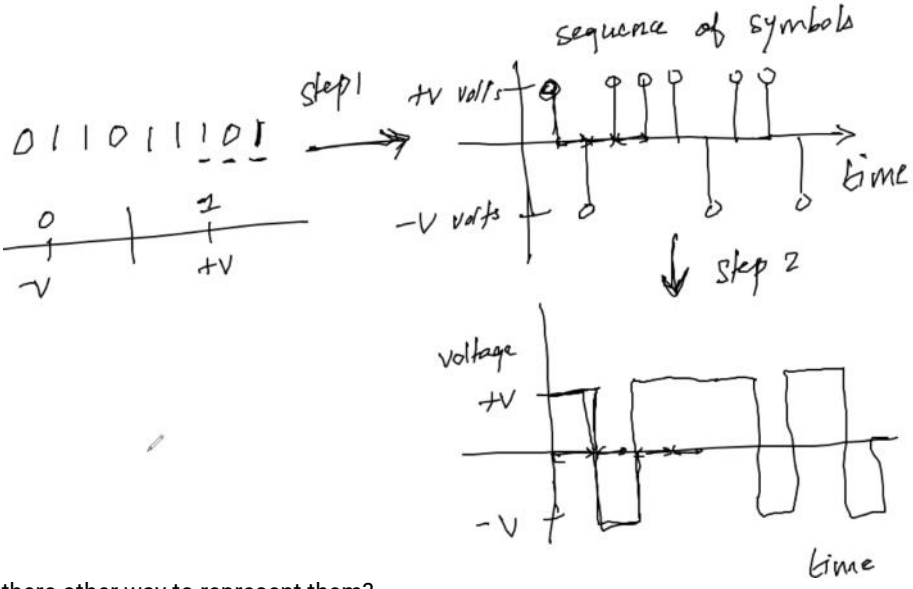
- 

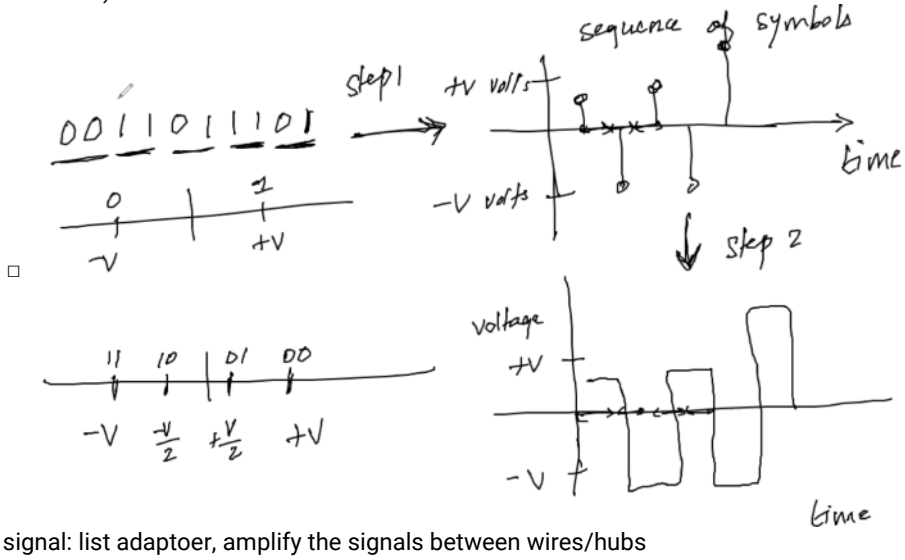
  - **Similarity**:
    - Both are store-and-forward. However, the routers forwards the datagram's destination IP address, and the switch forwards the frame's destination MAC address.
    - Both have forwarding tables
  - **Difference**
    - Routes: have complicated routing algorithm to compute the forwarding table.
    - Switches: no forwarding table, it's more like a plug-and-play devices (No need to configure). They learn the forwarding table automatically via flooding(which means broadcasting the MAC address to all devices), but what they learn are not necessary the optimal path. E.g. when a packet arrives switch, if the forwarding table is empty, or the packet Dest MAC address doesn't match any entry on the switch forwarding table. Then the Ethernet switch is gonna send the link-packet to all neightboor nodes' interface (except the one which it arrives).
- 

4/22 Slide 22: Pyhscial layer:

- What is the functionality of pyhsical layer:
  - It transform these bits from the link layer into analog signal(e.g. Voltage, EM wave) at sender side, when it arrived receiver you transform back to bit again. But the actual implementation at physical layer can varies depending on the transmission medium.

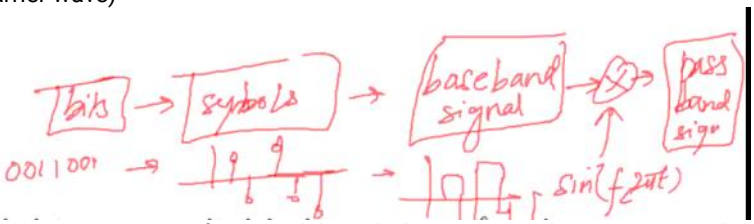
- Modulation: Understanding Modulation! | ICT_#7
  - What is **modulation**: It's the process of converting the sequence of bit that the link layer wants to send to the continuous analog signal that physical layer wants.
    - 

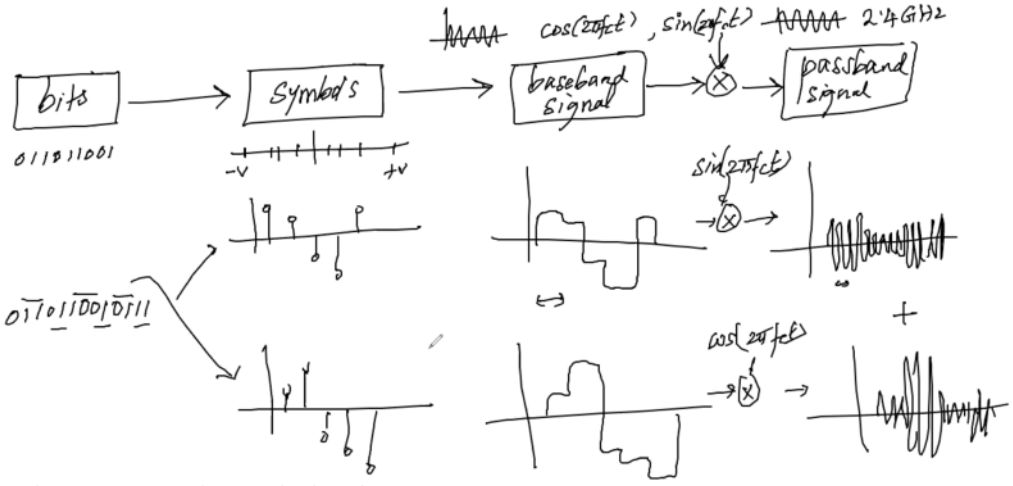
  - What is an neat way to transform the digital bit to analogy signal?
    - 

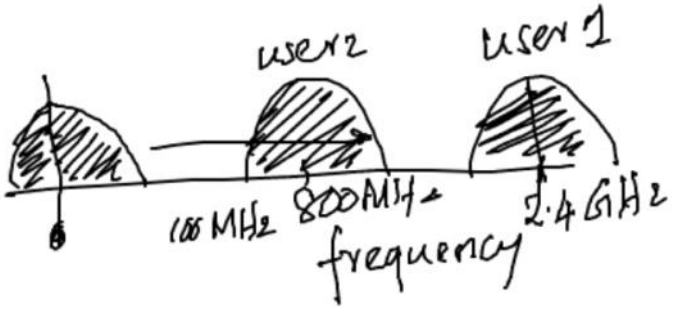
  - Q: is there other way to represent them?
    - ==> have other variance between -v to +v so that we can represents multiple bits, known as PAM(Pulse Amplitude Modulation)
    - 

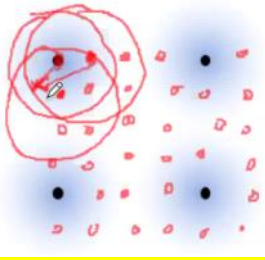
  - carrier signal: list adaptoer, amplify the signals between wires/hubs
- **Process of modulation:**, https://www.youtube.com/watch?v=Iyzpt3bKTTI&list=PLuUdFsbOK_8pWzW7KJjbiJ8Ow0cdHlApV&index=8
  - Take a sequence of bit convert to **impulse symbol**, and convert sequence of symbol to **baseband signal**(which is more continuous signal), and then convert baseband signal to **passband signal** (which mean shiting to a higher frequence

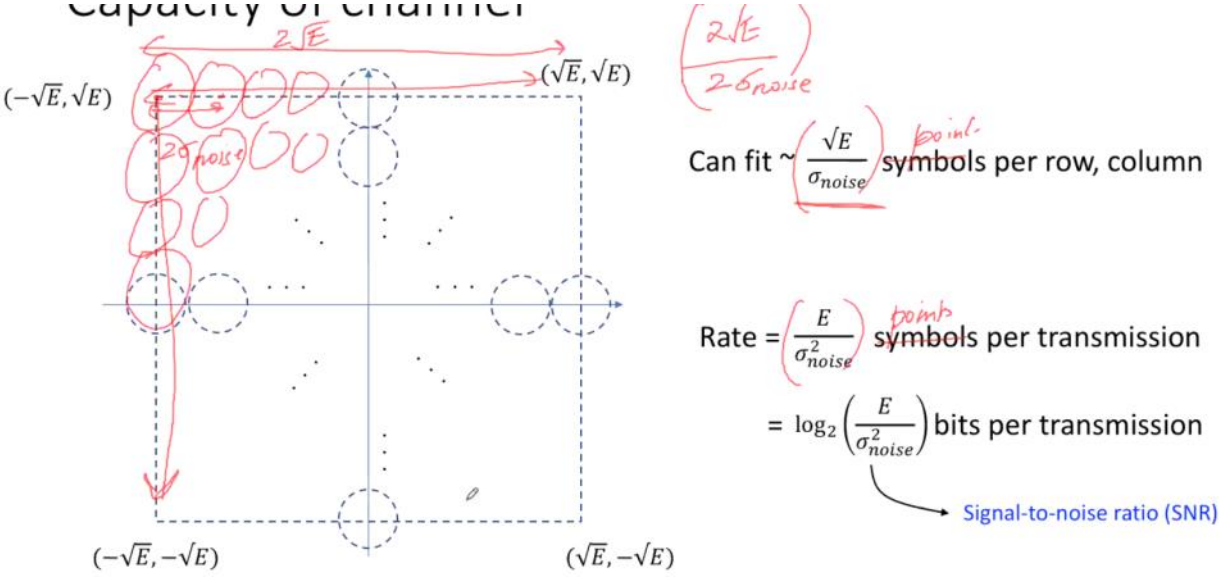
band via carrier wave)

- 

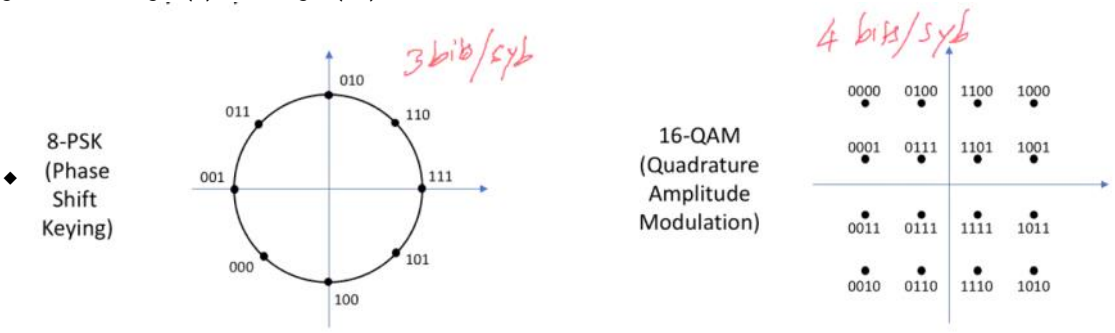
- You can send two sequence of sinusoidal symbol, one sine and one cosine, and they can be transmitted simultaneously.
  - 

- What is the purpose to change the band?
  - bacause the baseband signal has wider range wave, and different media might use different bandwith (e.g antennat using 800MHz, and WiFi use 2.4 GHz), so we could shiting the baseband to a narrower range, and to reduce the interference.
  - Because the interference is very noisy if we have multiple users
    - 

- How does the carrier wave conver the baseband signal to passband signal?
- Why do we converts those bit to two dimension? Instead of higher dimension, like 3, 4...
  - Because noise. What user receive is not exactly the symbol of what has sends
  - And, when we have multiple user, if the noise region is wide, we might have a lot of noise will be overlapped, and we cannot tells two symble apart anymore!
    - 
  - Why couldn't we spread/space out the distance between two transmission point so they don't overlap?
    - Because of power constrain. The amount of energy/power reqruied to transmit the signal is proportion of the square of amplitude.
  - We cannot compact those point too close to each other <u>because of te level of robustness wo against noise will decrease</u>, and we also know that we cannot spread out those point too far away from origin <u>because of power limitation</u>. So this lead to the question **what is really the best way to organize those points**? ==> (2*the length of rectangular / 2*radius of noise region)



- Q: why the number of bit you can send is log2…?
  - E.g. 8-PSK is log_2(8) == 3, log_2(16) == 4



- Difference between wired and wireless:
  - Most of this discussion carries over to the wireless physical layer as well. The difference between the wired and wireless physical layer have to do with attenuation, multipath, and interference in a wireless medium. **Attenuation** refers to how the voltage level of the transmitted bits degrades as we get further and further from the transmitter (this is just a consequence of the EM wave being an electric field). **Multipath** refers to how the EM waves from the transmitter to the receiver can take different paths. The waves arriving at the receiver from different paths can constructively or destructively interfere at the receiver depending on small differences in the receiver's position. One simplified view of both attentuation and multipath is that it reduces the SNR at the receiver in a wireless link. **Interference** refers to how EM waves from multiple transmitters can arrive at the receiver at the same time, destroying the reception of the signal for all transmitters in the process.
  - These 3 problems typically don't show up on a wired link, which is why wired links have higher SNRs (and hence higher capacity) than wireless links.
- **Capacity of channel**
  - Capacity: the maximum rate at which bits can reliably be transmitted across the channel
  - Capcity of Gaussian noise: the maixmum number of bit you can send per symbol/transmission
    - $\frac{1}{2}\log_2(1 + SNR)$
  - How many symbol we can send per second?
  - Why is bandwidth important to us?
    1. Different channel can accept/limit to certain range of frequency/bandwidth (e.g. 300 mHz for antenna, 2.4Ghz for Wifi)
- **Nyquist theorem:**
  - if you want to produce a bandwidth B hertz, you can send a symbol of gap 1/2B per second, you can construct a continuous sinusoid signal.
  - It's allowed to increase the gap but not decrease, because the frequence will be greater than B hertz.
  - Because FCC define the range of frequence/bandwidth you can send, so you want to use it as much as you can, and Nyquist theorem could tell you what is the best Symbol transmission rate (number of symbol per second) you should use.
- Capacity of channel

- capacity of link: throughput
- throughtput is the property for a user
- and the capacity of link is for the channel

Summary:
1) The reason to have Nyquist theorem is because FCC defines the range of frequency/bandwidth you can send, so you want to use it as much as you can (because of that cost money), and by using Nyquist theorem, it could tell us what is the best Symbol transmission rate (number of symbol per second) you should use, right?
2) I kinda confused with the logic of several words in the slide: the capacity of the channel, number of bit per symbol, number of symbol per second. I wonder how do they being measured? and what are the relationship between them?

The capacity of the channel is measured by the maximum number of bit you send over a link per second, which is defined by this formula:

$$2B \times \frac{1}{2}\log_2(1+SNR) = B\log_2(1+SNR) \text{ bits/second}$$
$$= B\log_2\left(1+\frac{E}{\sigma^2}\right) \text{ bits/second}$$

E: power per symbol
σ^2: noise power

where, B is the bandwith in Hertz, E is the signal power in Watt(which varies as the square of the voltage), sigma^2 is the noise power. The overall unit is the capacity of the channel in bits/sec.

and the number of bit per symbol is defined by this formula:

$$= \log_2\left(\frac{E}{\sigma^2_{noise}}\right) \text{bits per transmission}$$

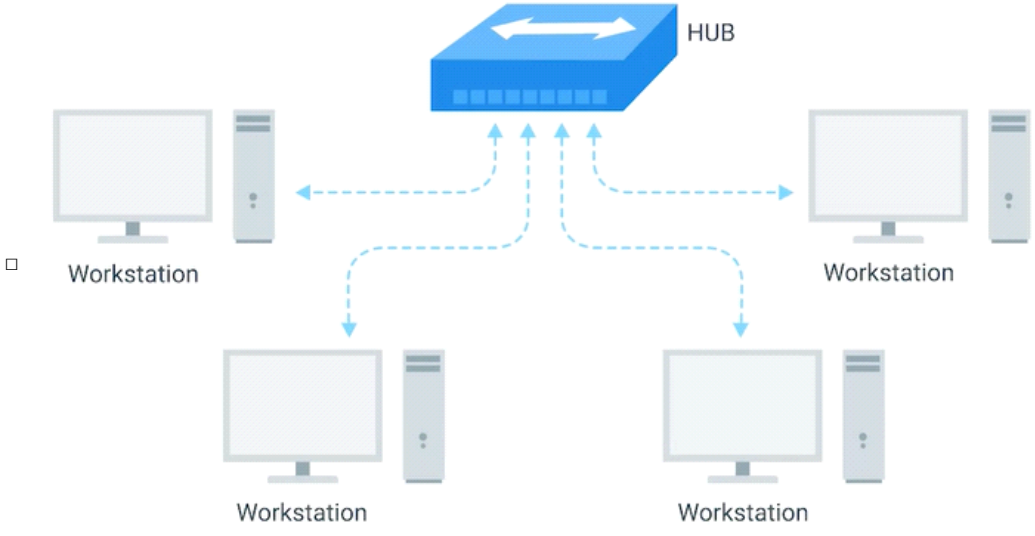
Signal-to-noise ratio (SNR)

and the number of bit per symbol is defined by this formula (Which supported by Nyquist theorem? Or Shannon theory?):

$$\frac{1}{2}\log_2(1+SNR) \text{ bits per transmission}$$

From <https://outlook.office.com/mail/inbox/id/AAQkADY1YzdkZjI2LWU1MjQtNDRiZC05YTJhLTVlODA4YTgwYmViZgAQADOZfpy3ghVAiVvDf8yyuO0%3D>

Extra note:
- Definition:
  - Hub: a physical layer device that allows for connections from many computers at once. It can possibly cause a large collision domain(A network segment where only one device can communicate at a time), and seriously slow down the network communicaiton, and that"s the reson hubs are fairly rare.
    - Switch: aka switching hub. It's similar to a hub, but it allows multiple device to communicate.

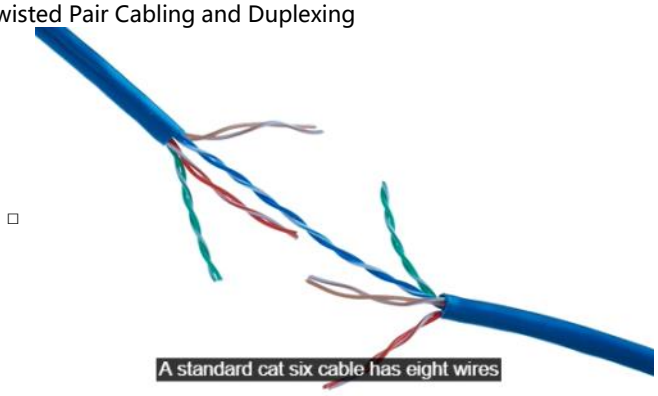


- Translating bits into voltages
  - Why does each value (or bit pattern) correspond to a range of voltages as opposed to a specific voltage?
    - This is to provide a certain level of robustness to **noise**, which can modify the voltage while it is being transmitted

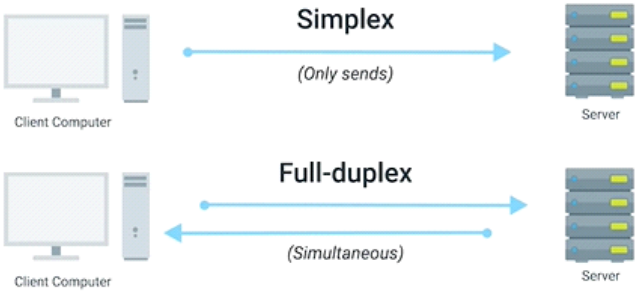
from the sender to the receiver end of a wire.
    - Noise is an unavoidable reality of any analog system, and is the key difference between the analog and digital worlds. Noise occurs for many reasons, e.g., manufacturing defects in the cables and the transmitter/receiver, imperfect contact between the transmitter and the cable, etc. Noise is what you are hearing when you hear static on your radio.

5/1 Week1:

- The Basics of Networking Devices:
    - Cables:
        - Connects difference devices to each other, allowing data to be transmitted over them.
        - Crosstalk: when a electrical pulse on one wire is accidentally detected on aother wire.
        - Copper cable: e.g. Cat5(categoies 5 cable), Cat5e, and Cat6
        - Fiber cable:contain individual optical fibers, which are tiny tubes made out of glass about the width of a human hair.
    - Routers
        - Hubs and switches: the primary devices used to connect computers on a single network, usually referred to as a LAN, or local area network, a physical layer devices.
        - Routers: A device that knows how to forward data between independent networks, a networking layer devices
- Physical layer:
    - Twisted Pair Cabling and Duplexing



    - Duplex communication: The concept that information can flow in both directions across the cable, e.g. phone call.
    - Simplex communication: The process is unidirectional, e.g monitor



- Lecture 18: Peer-to-Peer applications

- 6.4.4 Virtual Local Area Networks (VLANs)