Introduction

Circuit Switching & Packet switching

Pros and cons of circuit switching

PRO: Uninterrupted connection, reserved and dedicated for you.
PRO: Potentially faster depending on how much reserved / generally super performance
CON: Reserved channel even for idle connections, can't be used by anyone. I.e. Waste of resources.

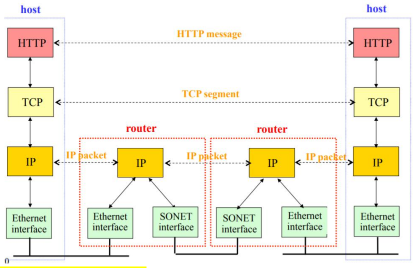
Pros and Cons of packet switching

PRO: Requires less infrastructure. With circuit switching, you have dedicated linking so will require more infrastructure.
PRO: when a transmission line fails, other transmission lines can be selected, which improves the reliability of transmission.
CON: Packet loss, Packets sent to the receiver by packet switching may be out of order. It need to extra sorting operation.

Layering

PRO: Introducing an intermediate layer provides a common abstraction for various network technologies
PRO: if no layering, each new application has to be re-implemented for every network technology
CON:Headers start to get really big.   E.g., typically TCP + IP + Ethernet headers add up to 54 bytes
CON:Information hiding may hurt performance   E.g. packet loss due to corruption vs. Congestion
CON: Layer N may duplicate lower level functionality   E.g., error recovery to retransmit lost data

Internet Layered Architecture

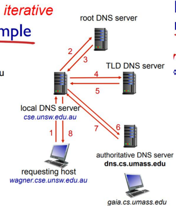


2.3 DNS(Hierarchy)

DNS name resolution example — iterative

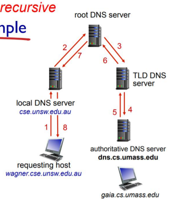
- host at wagner.cse.unsw.edu.au wants IP address for gaia.cs.umass.edu

iterated query:
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example — recursive

recursive query:
- puts burden of name resolution on contacted name server



Inserting records into DNS

- example: new startup "Network Utopia"
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into .com TLD server:
    `(networkutopia.com, dns1.networkutopia.com, NS)`
    `(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com

DNS resource records (RRs)

Each DNS reply carries one or more RRs

DNS: distributed db storing resource records (RR)

RR format: `(name, value, type, ttl)`

type=A
- **name** is hostname
- **value** is IP address

type=NS
- **name** is domain (e.g., foo.com)
- **value** is hostname of authoritative name server for this domain

type=CNAME
- **name** is alias name for some "canonical" (the real) name
- `www.ibm.com` is really `servereast.backup2.ibm.com`
- **value** is canonical name

type=MX
- **value** is name of mailserver associated with **name**

why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

2.4 P2P

Pros and Cons of P2P

Pros	Cons
Self-scalability: new peers bring new service capability, as well as new service demands. **Speed**: Parallelism, less disagreements **Reliability**: Redundancy, fault tolerance **Geographic Distribution**	Fundamental problems of decentralised control: **Synchronisation**: If one thing changes, everyone else has to update with the new changes too, otherwise it won't work with new and old copies of the file everywhere.

$$D_{P2P} \geqslant \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\} \quad D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

F- File size
u-Upload
d-download

Application layer

application	application layer protocol	underlying transport pr
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

UDP: SNMP, FPS, RIP, DNS, DHCP, video chat

2.1 HTTP

HTTP is stateless.

HTTP is all text.

- *Sending "12345678" as a string is 8 bytes*

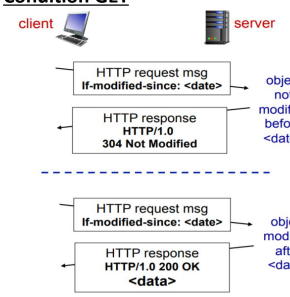
Web use cookie to keep "state"
弃所有失序分组
Improve HTTP performance: cache and replication.

- Usually index.html is downloaded first. Upon inspecting index.html, the clients download all the objects referenced in index.html
- Q. For an index file containing 2 objects, what is the response time for (a) non-persistent HTTP, (b) Persistent HTTP without pipelining, and (c) Persistent HTTP with pipelining?
- A.
  - 2xRTT + some additional file transmission delay to get the index file (1xRTT for opening TCP connection and 1xRTT for downloading index)
  - For non-persistent, each object costs 2xRTT
  - For persistent without pipelining, each object costs 1xRTT
  - For persistent with pipelining, all object downloaded in 1xRTT
  - (a) 2+2x2 = 6xRTT + some file tx delay
  - (b) 2+2 = 4xRTT + some file tx delay
  - (c) 2+1 = 3xRTT+ some file tx delay

why Web caching?
reduce response time for client request
reduce traffic on an institution's access link
Internet dense with caches: enables " poor " content providers to effectively deliver content

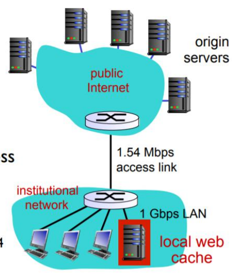
total delay=LAN delay+access delay+Internet delay

Condition GET



Caching example: install local cache

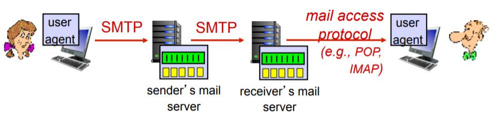
Calculating access link utilization, delay with cache:
- suppose cache hit rate is 0.4
  - 40% requests satisfied at cache, 60% requests satisfied at origin
- access link utilization:
  - 60% of requests use access link
- data rate to browsers over access link = 0.6*1.50 Mbps = .9 Mbps
  - utilization = 0.9/1.54 = .58
- total delay
  - = 0.6 * (delay from origin servers) +0.4 * (delay when satisfied at cache)
  - = 0.6 (2.01) + 0.4 (~msecs)
  - = ~ 1.2 secs
  - less than with 154 Mbps link (and cheaper too!)



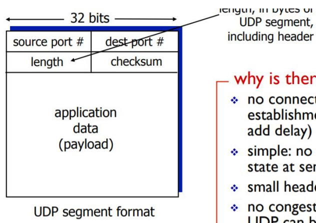
2.2 SMTP

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

SMTP Compare with HTTP

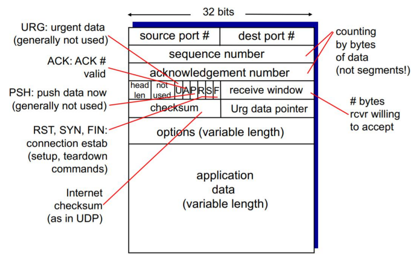


Transport layer (TCP and UDP)



why is there a UDP?
- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired

TCP segment structure



Checksum 计算: binary add, 回卷(wraparound), 取反

Reliable UDP protocol

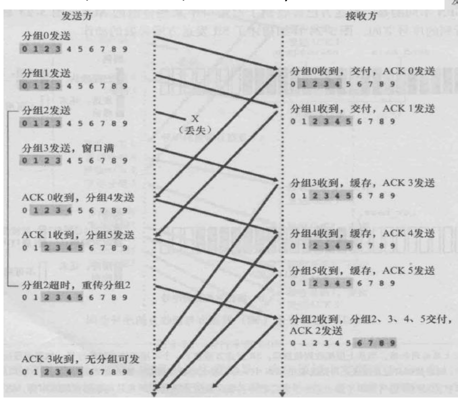
窗口大小: GBN: 如果 sequence number 为 n, 则发送窗口大小最大值为 $2^n - 1$,接收窗口为 1

SR: 如果 sequence number 为 n, 则发送窗口和接受窗口的最大值都为 2^{n-1}

1.GBN 采用累积确认(cumulative acknowledgement), 并丢

(使用序号,累积确认, checksum, 超时重传)

2. SR
非累积确认



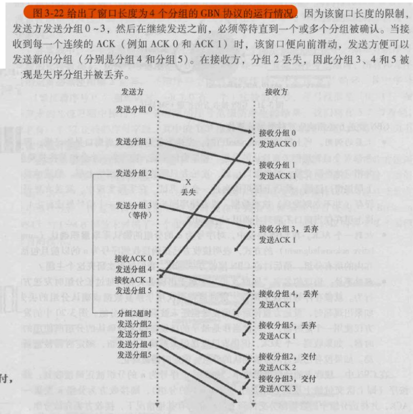


图 3-22 给出了窗口长度为 4 个分组的 GBN 协议的运行情况。因为该窗口长度的限制，发送方发送分组 0 ~ 3，然后在继续发送之前，必须等待直到一个或多个分组被确认。当接收到每一个连续的 ACK（例如 ACK 0 和 ACK 1）时，该窗口便向前滑动，发送方便可以发送新的分组（分别是分组 4 和分组 5）。在接收方，分组 2 丢失，因此分组 3、4 和 5 被发现是失序分组并被丢弃。

TCP

机制(累积确认, 超时重传机制)
重传超时间隔:
TimeoutInterval = EstimatedRTT + 4*DevRTT
EstimatedRTT=(1-α)Estimated+α SampleRTT

TCP Flow Control

Sender: LastByteSend - LastByteACK <= rwnd
Rece: LastByteRecv - LastByteRead <= RecvBuffer
rwnd=RecvBuffer-(LastByteRecv-LastByteRead)

TCP Connection management

Three-way shake hand
1. SYN = 1, random choose a client_isn
2. SYNACK: SYN=1, ACK sequence: client_isn + 1, random choose a server_isn
3. SYN = 0, ACK sequence: server_isn + 1

TCP connection close
1. client send a TCP packet, FIN=1
2. server send a ACK packet, Then Send a FIN=1
3. client send server FIN=1 ACK

TCP congestion control

At sender, LastByteSend-LastByteACK <= min{cwnd, rwnd}

TCP is self-clocking

Duration: Slow-start, congestion avoidance, fast recovery

TCP Tahoe: 1.slow-start, cwnd=cwnd*2

2. packet loss event ocurr,(three duplication ACK or timeout) ssthresh=cwnd/2, cwnd=1

3. congestion avoidance: when cwnd>ssthresh, cwnd=cwnd+1

TCP Reno: 1. slow-start, cwnd=cwnd*2

2.congestion avoidance: when cwnd>ssthresh, cwnd=cwnd+1

3. packet loss event: when three duplication ACK, cwnd=cwnd/2, ssthresh=cwnd/2(enter fast re-transmission, after that cwnd=cwnd+1), when timeout, ssthresh=cwnd/2, cwnd=1

UDP checksum only can detect one bit error

Q7. Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely sure that no bit errors have occurred? Explain. Would things be different with TCP?

A7. No, the receiver cannot be absolutely certain that no bit errors have occurred. This is because of the manner in which the checksum for the packet is calculated. If the corresponding bits (that would be added together) of two 16-bit words in the packet were 0 and 1 then even if these get flipped to 1 and 0 respectively, the sum still remains the same. Hence, the 1s complement the receiver calculates will also be the same. This means the checksum will verify even if there was transmission error. Since TCP uses the same checksum mechanism, the above would hold true with TCP as well.

Network Layer(Data Plane, Control Plane)

Routing and Forwarding

routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

Forwarding table use (Tenary content Address memory)TCAM to forward IP datagram

Why forwarding table copy to input line card?

Answer: Because forwarding direction can be determine in the input line card, eliminates the need to call a central routing processor, avoiding centralized processing

Distance Vector(Bellman-ford)

Cost to

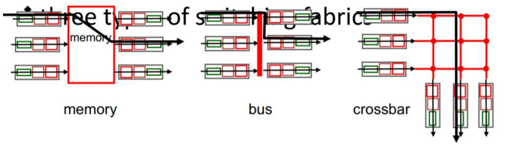
		u	v	x	y	z
From	v	1	0	3	4	5
	x	2	3	0	1	2
	y	3	4	1	0	3
	z	4	5	2	3	0

Switching fabrics

❖ transfer packet from input buffer to appropriate output buffer

❖ switching rate: rate at which packets can be transfer from inputs to outputs

- often measured as multiple of input/output line rate
- N inputs: switching rate N times line rate desirable

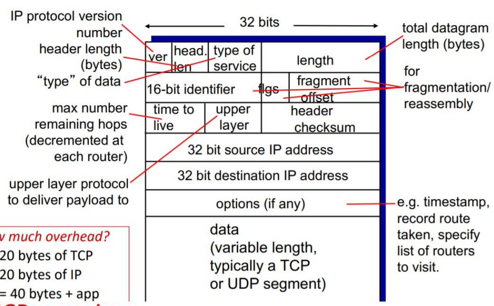


HOL(head of link blocking): queued datagram at front of queue prevent others in queue moving forward.

How much Buffering?

$$\frac{\mathrm{RTT}*C}{\sqrt{N}}(C: link-capacity, N: \quad N-flows)$$

RECAP: IP datagram format



DHCP overview:

- host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet

Dijkstra's Algorithm

Step	Set N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					

Oscillations

算法可能存在震荡现象

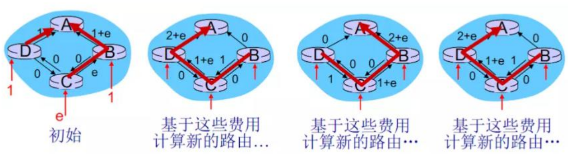


image.png

当链路状态更新的太快并且不断变化的时候，假设我们发出一个分组，结果还没到目的地，路由表就更新了，然后这个数据报就一直在路由间切换，最后由于ttl到0，直接丢弃。这就是震荡现象。

Data Link Layer + WLAN + multimedian

Link layer service: framing, link access, reliable transport between adjacent node flow control, error detection, error correct

MAC and ARP

MAC Address vs. IP Address

- MAC addresses (used in link-layer)
  - Hard-coded in read-only memory when adapter is built
  - Like a social security number
  - Flat name space of 48 bits (e.g., 00-0E-9B-6E-49-76)
  - Portable, and can stay the same as the host moves
  - Used to get packet between interfaces on same network
- IP addresses
  - Configured, or learned dynamically
  - Like a postal mailing address
  - Hierarchical name space of 32 bits (e.g., 12.178.66.9)
  - Not portable, and depends on where the host is attached
  - Used to get a packet to destination IP subnet

Switch

Switches vs. routers

both are store-and-forward:

- routers: network-layer devices (examine network-layer headers)
- switches: link-layer devices (examine link-layer headers)

both have forwarding tables:

- routers: compute tables using routing algorithms, IP addresses
- switches: learn forwarding table using flooding, learning, MAC addresses

Ethernet

preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rat

Data link layer switch(self learning)

地址	接口	时间
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
...	...	...

Summary

- Video has strict timing requirements for playout
- Videos are often segmented into chunks or blocks:
  - Each block is about 2-10 seconds long (many frames in each block)
  - Each block has to start playing at strict timing intervals for smooth video
- Streaming video over networks would have no problem if network had constant delay
- Network delay is variable: challenge for video streaming
  - Video frames or blocks may be delayed for playing (*video freezing effect*)
- Stored video streaming solution:
  - Playout delay at client to absorb delay variations
  - Some blocks/frames are initially buffered before the playout of the video starts

---

SDN

Control decision for the entire network is made at centrally.

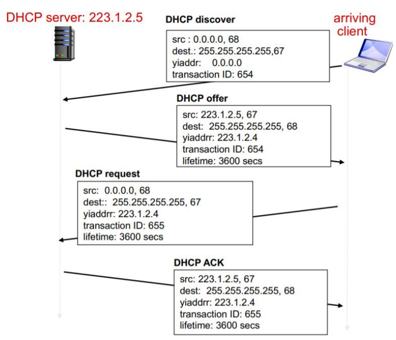
- central controller transfer software controlled control logic to each router dynamically

Openflow in SDN: is a standard specified how SDN router communicate with a central control server

It can used by central controller tell a router how to treat certain type of packet, behavior of router in network can be rapidly modified at central controller. Useful for load balancing in response to rapid change in network data path.

CAM(content Addressable memory)

With CAM, OS can examine the contents of the entire memory in one single cycle. But is expensive than RAM, so It used in routers, not PC. TCAM (Ternary CAM) is used by most routers to store router table.(due to it can handle masks)



NAT

10.0.0.0/8 172.16.0.0/12 192.168.0.0/16

Advantage:

1.range of addresses not needed from ISP:

2.can change addresses of devices in local net without notifying outside world

3. can change ISP without change IP address

disadvantage: violate layering principle, end-to-to argument.

Pratical Issue:

Some applications embed IP address or port number in their message payload. (DNS, FTP)

NAT traversal solution

1. Statically configure NAT to forward incoming connection requests at given port to server 2.Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol.
3.relaying, NATed client establishes connection to relay – external client connects to relay – relay bridges packets between to connections.