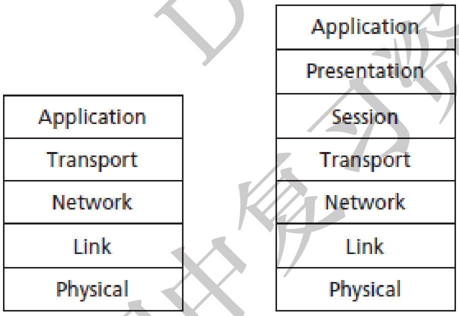
1. Introduction to computer network

1.1 概念

略

1.2 重点

- Protocol Layering
    - application-layer **message**
    - transport-layer **segment**
    - network-layer **datagram**
    - link-layer **frame**
    - **header fields** and a **payload field**
    - Pros and cons
        - Pro: provides a common abstraction for various network technologies
        - Con: Duplicate functionality
        - Con: Performance
        - Con: Header gets big
        - Layer violation
    - Which one does what?



a. Five-layer Internet protocol stack　　**b. Seven-layer ISO OSI reference model**

    - Application Layer: where network applications and their application-layer protocols reside.
    - Transport Layer:
        - TCP provides a connection-oriented service to its applications, **guaranteed** delivery (reliable), flow control, segmentation, and congestion control.
        - The UDP protocol provides a connectionless service to its applications. This is a no-frills service that provides no reliability, no flow control, and no congestion control.
    - Network layer:
        - Best effort delivery from one host to another(global)
    - Link layer:
        - Best effort delivery from one host to another(local)
- Packet switching: pros and cons

Circuit Switching Vs Packet Switching

Circuit Switching	Packet Switching
Physical path between source and destination	No physical path
All packets use same path	Packets travel independently
Reserve the entire bandwidth in advance	Does not reserve
Bandwidth Wastage	No Bandwidth wastage
No store and forward transmission	Supports store and forward transmission

3

- Delay: analysis and calculation
  - $d_{\text{end-end}} = \sum (d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}})$
  - Down to the bottom, data are just electronic signal! It travels through some media.
  - Understand the whole process:
    - packet arrive at a switch (input buffer)
    - switch processes it d_{proc}
    - switch dispatch it to a port, entering an output queue d_{queue}
    - packet is transmitted to the media (cable/fibre/air) d_{trans}(us-ms): L/R
    - the signal travels through media d_{prop}(ms): d/s

1.3 习题

1. Suppose users share a 2 Mbps link. Also suppose each user transmits continuously at 1 Mbps when transmitting, but each user transmits only 20 percent of the time.

a. When circuit switching is used, how many users can be supported?

b. For the remainder of this problem, suppose packet switching is used. Why will there be essentially no queuing delay before the link if two or fewer users transmit at the same time? Why will there be a queuing delay if three users transmit at the same time?

c. Find the probability that a given user is transmitting.

d. Suppose now there are three users. Find the probability that at any given time, all three users are transmitting simultaneously. Find the fraction of time during which the queue grows.

Answer: a) 2 users can be supported because each user requires half of the link bandwidth.

b) Since each user requires 1Mbps when transmitting, if two or fewer users transmit simultaneously, a maximum of 2Mbps will be required. Since the available bandwidth of the shared link is 2Mbps, there will be no queuing delay before the link. Whereas, if three users transmit simultaneously, the bandwidth required will be 3Mbps which is more than the available bandwidth of the shared link. In this case, there will be queuing delay before the link.

c) Probability that a given user is transmitting = 0.2

d) Probability that all three users are transmitting simultaneously = $\binom{3}{3} p^3 (1-p)^{3-3}$

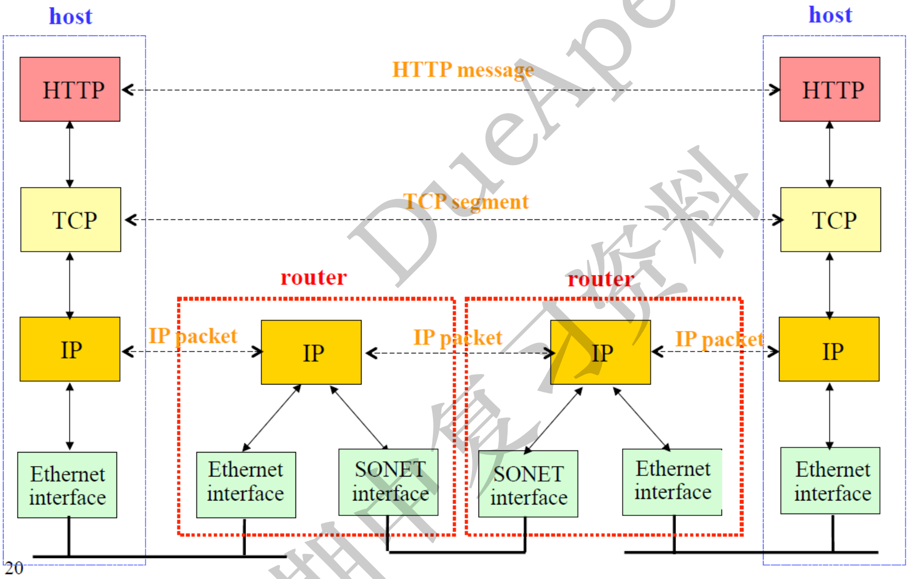
= $(0.2)^3$ = 0.008. Since the queue grows when all the users are transmitting, the fraction of time during which the queue grows (which is equal to the probability that all three users are transmitting simultaneously) is 0.008.

2. List five tasks that a layer can perform. Is it possible that one (or more) of these tasks could be performed by two (or more) layers?

Answer: Five generic tasks are error control, flow control, segmentation and reassembly, multiplexing, and connection setup. Yes, these tasks can be duplicated at different layers. For example, error control is often provided at more than one layer.

6. Which layers in the Internet protocol stack does a router process? Which layers does a link-layer switch process? Which layers does a host process?

Answer: Routers process network, link and physical layers (layers 1 through 3). (This is a little bit of a white lie, as modern routers sometimes act as firewalls or caching components, and process Transport layer as well.) Link layer switches process link and physical layers (layers 1 through2). Hosts process all five layers.



7. Which of the following services does the Internet network layer provide for the Internet transport layer? (possible multiple answers)

a. in-order delivery of data segments between processes

b. best effort delivery of data segments between communicating hosts

c. multiplexing and demultiplexing of transport layer segments

d. congestion control

2. Application Layer

2.1 概念

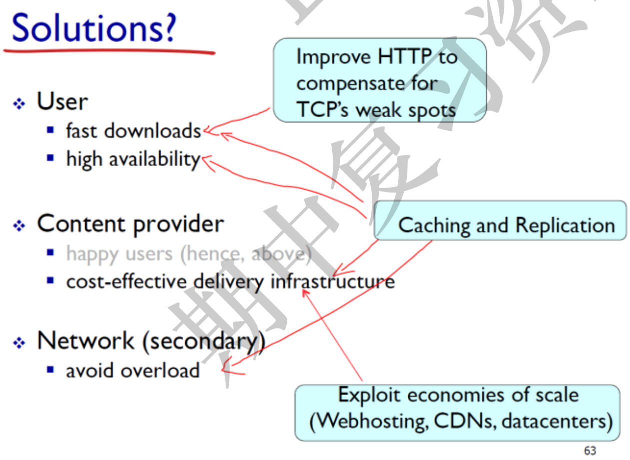
- Application architecture: client/server architecture, P2P architecture
    - **Server:**
        - Exports well-defined request/response interface
        - long-lived process that waits for requests

        - Upon receiving request, carries it out
    - **Client:**
        - Short-lived, user side, initiate communication
    - **P2P:**
        - no always-on server
        - arbitrary end systems (peers) directly communicate
        - Symmetric responsibility
- Purpose of computer network: allow Inter-Process Communication (IPC)
- Socket: interface between the application process and the transport-layer protocol. Defined by IP address + port number
- 4 dimension for choosing transport-layer protocol (important):
    - Reliability, throughput, timing and security
    - loss-tolerant applications
    - bandwidth-sensitive applications
    - elastic applications
- **HTTP**(important)
    - Web page: often consist of a base HTML file and several referenced objects
    - URL: hostname and path
    - HTTP is over TCP, port 80
    - Stateless
    - Non-Persistent and Persistent Connections
    - RTT
    - Request: request line, header lines, entity body
        - request line: method, URL, HTTP version
        - header lines: tell server more info about browser and what I want
        - Entity body: after one blank line. Can be empty for GET
    - Response: status line, header lines, entity body
        - status : 200 OK, 301 moved, 404 Not found
        - Last-Modified, conditional GET
        - Set-cookie: used for user-server interaction
    - Caching, institutional cache, CDN
    - All Text: simple(readable) but not efficient
    - HTTP 1.1: more request method(delete etc.), persistent connection with pipelining
- HTTPS
    - Provide security by authentication and bidirectional encryption
- ~~FTP~~
    - ~~TCP, port 20/21~~
    - ~~Control connection, data connection; out-of-band, in-band~~
    - ~~NAT~~
- ~~SMTP~~
    - ~~Used for mail~~
    - ~~Similarity and difference to HTTP:~~
        - ~~SMTP pushes file to server~~
        - ~~HTTP pulls data from server~~
        - ~~SMTP require 7-bit ASCII encoding~~
        - ~~HTTP send images/document independently, SMTP combine them together~~
- **DNS** domain name system (important):

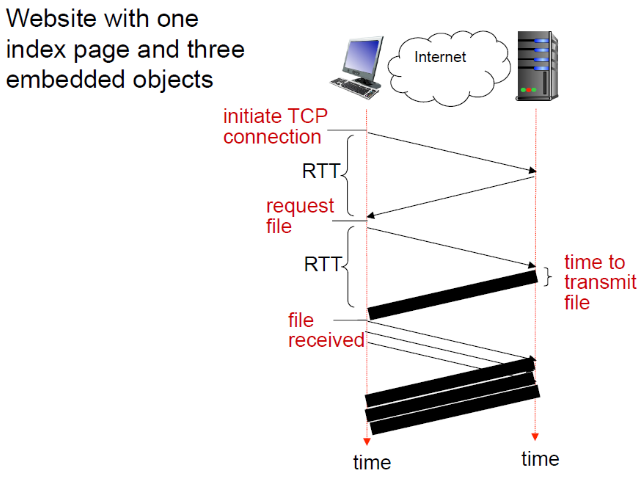
- Distributed, hierarchical
    - Application layer
    - Resolve domain name to IP address
    - Usually UDP
- DASH: Dynamic, adaptive streaming over HTTP
    - Server divide video into chunks and encode it using different rates
    - Server provide manifest file for different chunk/rate
    - Client request suitable chance/rate based on periodic measurement of C/S bandwidth
- ~~**P2P**(important)~~
    - ~~Bit torrent: Tracker and torrent~~
        - ~~rarest first: what and why?~~
        - ~~tit-for-tat: what and why?~~
        - ~~optimistically unchoke: what and why?~~
        - ~~Free-riding: Can someone who never send, receive whole file?~~
    - ~~DHT (Distributed hash table), p2p-transport p13~~
        - ~~Hash table: key-value pair~~
        - ~~Peer has ID, store key-value in peer that has closest ID to key.~~
        - ~~Peer churn: store 2 successor~~
        - ~~Short cuts~~

2.2 重点

- `Few method to improve HTTP performance:`



    - `Persistence connection + pipelining: introduced in HTTP1.1`



  - Web caches (proxy): satisfy client request without involving origin server
    - acts as both client and server
    - typically cache is installed by ISP (university, company, residential ISP)
    - reduces response time and traffic
    - PPT 74-77: cheaper and faster
  - Conditional GET
    - utilising Last-modified and If-modified-since header fields
    - reduces traffic
  - Replication and CDN
    - Replicate popular Web site across many machines
    - CDN: Caching and replication as a service
      - Basically, stores content in a server close to user
      - distributed, application-level infrastructure
      - pull and push: predict access rate
      - different with cache: application level, dynamic, smarter(push)

- HTTP: How many socket for n client?
  - N+1
- DNS: Goals
  - No naming conflicts
  - Scalable
  - Distributed, autonomous administration
  - Highly available
  - Lookups should be fast
- DNS: Three intertwined hierarchies
  - Hierarchical namespace
  - Hierarchically administered
  - (Distributed) hierarchy of servers
    - Root servers
    - Top-level domain(TLD) servers
    - Authoritative DNS servers
  - Example: .edu, berkeley.edu, eecs.berkeley.edu
- DNS resolution:
  - Recursive or iterative query: example

    - Cache and TTL
    - Types of DNS record:
        - Type A: (relay1.bar.foo.com, 145.37.93.126, A)
        - Type NS: (foo.com, dns.foo.com, NS)
        - Type CNAME: (foo.com, relay1.bar.foo.com, CNAME)
        - Type MX: (foo.com, mail.bar.foo.com, MX)
        - PTR
- Insert DNS records
    - Register NS and A record to TLD
        - (networkutopia.com, dns1.networkutopia.com, NS)
        - (dns1.networkutopia.com, 212.212.212.1, A)
    - Insert A(www.networkutopia.com, xxx.xxx.xxx, A) and MX records to authoritative DNS server of network Utopia

Visiting webpage of Network Utopia

1. Alice in Sydney wants to visit web page of Network Utopia
2. Alice's browser sends DNS query to local DNS server
3. Local DNS server contacts TLD server for com (assuming TLD server address is cached in local server to avoid going to root)
4. TLD server sends reply to local server with two RRs
    1. **(networkutopia.com, dns1.networkutopia.com, NS)**
    2. **(dns1.networkutopia.com, 212.212.212.1, A)**
5. Local DNS server sends DNS query to **212.212.212.1 asking for Type A record for** www.networkutopia.com
6. **The response provides RR that contains say 212.212.71.4**
7. **Alice's browser now initiates HTTP request to 212.212.71.4 to get the web pages for Network Utopia**

2.3 习题

Tutorial I 2, 3, II 3,4,5

~~1. How does SMTP mark the end of a message body?~~ How about HTTP? Can HTTP use the same method as SMTP to mark the end of a message body?

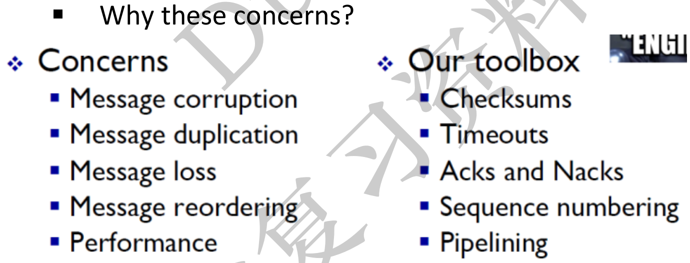
Answer: SMTP uses a line containing only a period to mark the end of a message body. HTTP uses "Content-Length header field" to indicate the length of a message body. No, HTTP cannot use the method used by SMTP, because HTTP message could be binary data, whereas in SMTP, the message body must be in 7-bit ASCII format.

3. Transport Layer

3.1 概念

- Application layer is our boss, network layer is ours to command!
- Because…
    - Network layer only find path, doesn't guarantee delivery/order/path
- Therefore…
    - Transport layer need to be able to do it! Or not…
- Transport layer provide logical connection between **processes;** network layer provide communication between hosts
    - Segmentation and reassembling

    - Multiplexing and de-multiplexing: multiple sockets(processes/port)
    - Source port / **destination port**
    - Source IP / **destination IP**
    - Non-persistent HTTP will have **different** socket for each request(connection)
    - Welcoming socket/connection socket
- Connectionless demux/connection oriented demux
    - Connectionless: UDP. Defined by 2-tuple (dest IP + dest port)
    - Connection-oriented: Defined by 4-tuple (src IP and src port is also used)
- UDP
    - Segment header: 64 bits. Source port(return address), dest port, length(in byte, including header), checksum, payload
    - Checksum: 1's complements
        - Detect single (or odd number) bit error. Cannot detect 2 (or even) bit error!
        - Calculation: add together, wraparound carry bit, then take complement.
        - Calculated over header and data
        - Error correction unnecessary (periodic messages)
    - Application: DNS, DHCP, SNMP, Gaming, video/audio
- TCP (important)
    - Handshake/acknowledgement
        - Why these concerns?



    - Feature: transport_part2 p21
        - Reliable, duplex, connection-oriented, flow controlled
    - segment header: 160+ bits
        - source port, dest port, seq(counting byte), ack#(counting byte), flags, checksum, receive window(expected byte#)
    - MSS(maximum segment size):
        - 1460 = MTU(maximum transmission unit, limit of IP datagram, 1500)-IP header(20)-TCP header(>=20)
    - Sequence number: ISN + k
        - ISN: initial sequence number
        - K: kth byte of stream
    - ACK sequence number
        - = next expected byte
        - =seq# + length
        - Cumulative ACK
    - Buffers out-of–order packet
    - Single timer
    - Fast retransmit: use duplicate(3) acks to trigger early retransmission
    - Rwnd: limit amount of in-flight data to rwnd value
    - Connection management

- Syn-flooding and syn cookie

3.2 重点

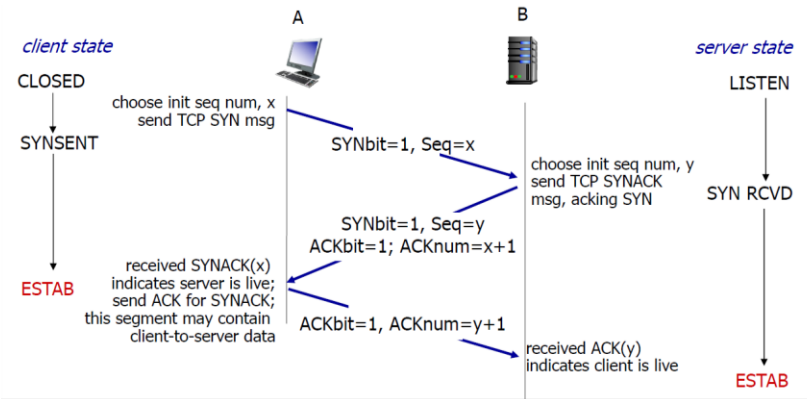
- Why 4 tuple for a TCP socket?

Everybody can connect to google.com:80 (even same computer can open different window), but how does Google's server know what to send to everyone? Therefore src IP and src port are needed.

UDP: also need src IP and src port, but user application receives everything, need to handle it. TCP socket doesn't receive these noise.

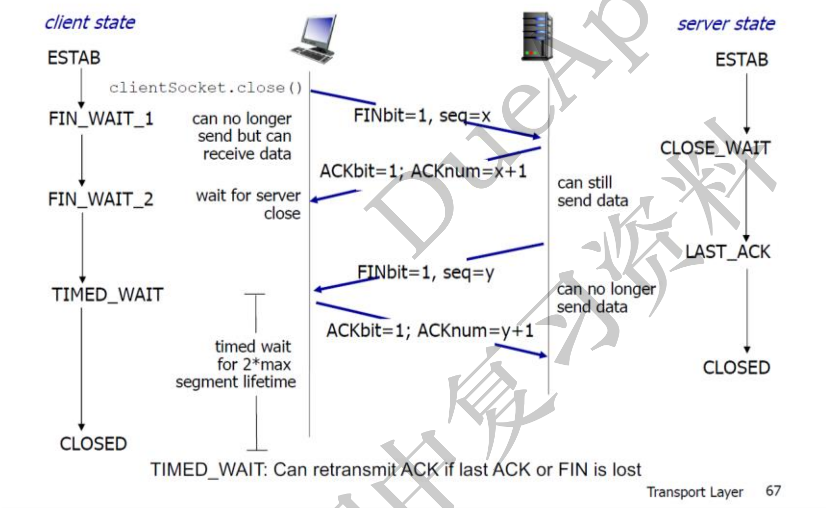
- Develop a reliable transport protocol(rdt) step by step: p2p_transport p54+
    - 1.0: assume network layer is reliable, transport layer doesn't have to do anything!
    - 2.0: deals with error bit. (partial) solution:
        - Error detection (with checksum)
        - Feedback: ack/nak
    - 2.1: deals with corruption of ack/nak, and possible duplication by:
        - Sequence number: sender add a number to pkt, receiver keeps track of which to expect
        - Stop and wait
        - Q: why only (0,1) sufficient for seq#?
    - 2.2 Nak free!
        - Ack last packet received
    - 3.0: deals with loss:
        - Timeout is introduced
    - Pipelined(sliding window) 3.0:
        - Deals with utilization: how?
        - Go-Back-N:
            - timer for oldest in-flight pkt
            - ack(n) means all packets up to n are received (cumulative ACK)
            - on timeout(n), retransmit packet n and all higher seq# in window
            - for receiver: out-of order pkt: re-ack
        - Selective repeat
            - Timer for each pkt
            - for receiver: Buffer out-of-order pkt, Ack individually
            - window size: <= (seq# range)/2
        - Study the state machine
- TCP timeout estimation
    - $$\text{EstimatedRTT} = (1-\alpha)*\text{EstimatedRTT} + \alpha*\text{SampleRTT}$$
      $$\text{DevRTT} = (1-\beta)*\text{DevRTT} + \beta*|\text{SampleRTT}-\text{EstimatedRTT}|$$
      (typically, $\beta = 0.25$)
    - $$\text{TimeoutInterval} = \text{EstimatedRTT} + 4*\text{DevRTT}$$
    - Retransmission is excluded(p44)

- TCP connection management
  3-way handshake



Syn loss: timout(3s)

4-way termination



3.3 练习

R8, R9, R14, R15, P14,P25