Processing delays: 4 types of delay. Processing delay is from nodal processing (when packet goes to the node), where they check bit errors and determine output link (usually < msec). Queuing delay is from time waiting at output link for transmission, depending on congestion level of router (changes accordingly to the network).(packet length bits, average packet arrival rate) *La / R* (link bandwidth bps) if ~ 0 (queuing delay small), if 🡪 1 (queuing delay gradually grows), if > 1 (more “work” arriving than can be serviced, average delay is ∞). Transmission delay is from (packet length (bits)) *L* / *R* (link bandwidth (bps)). Propagation delay is the bandwidth (bits/sec) of the length, where (length of physical link) *d / s* (propogation speed ~2x108 m/sec). dnodal = dproc + dqueue + dtrans + dprop. Internet protocol stack (ISO/OSI): Application (exchange information between hosts, *messages*), Presentation (allow applications to interpret data), Session (synchronization, checkpointing, recovery of data exchange), Transport (exchange packets between hosts, error recovery, congestion and flow control, *segments*), Network (route packages from source host to destination host, *datagram*), Link (deliver packets from one end of a transmission channel to the other, error recovery, flow control, coordinate transmission sharing, *frame*), Physical (define electrical/optical signals that represent bit sequences, *bits*)

A screenshot of a cell phone

Description automatically generatedA picture containing screenshot, text

Description automatically generatedTransport layer (TCP, UDP): Function of transport layer is to have a form of reliability of transport of data and logical communication between processes. Multiplexing handles data from multiple sockets, and adds a transport layer, which is used for demultiplexing to deliver and receive the correct socket (TCP socket identified by 4-tuple: source/dest IP address, source/dest port number). UDP (User Datagram Protocol): connectionless, used to stream multimedia apps, simple. It has a checksum, which detects errors in transmitted segments. To recover from errors, acknowledgements (ACKs) has the receiver explicitly tell sender that the packet received okay, (NAKs is opposite—not okay). Duplicates stop and wait for receiver’s response. Usender = (L/R) / (RTT + L/R). To calculate time that sends the next packet: t = RTT + L/R. TCP has retransmission (packet getting dropped that later getting it reissued), and letting source know when things happen. TCP has condition control and flow control (thinking about how it changes). Two generic forms of pipelined protocols: go-Back-N and selective repeat. For go-Back-N, sender can have up to N un-ACKed packets in pipeline, receiver only sends cumulative ACK (doesn’t ACK if there’s a gap), and sender has timer for oldest un-ACKed packet (when timer expires, retransmit all un-ACKed packages). Selective repeat: sender can have up to N un-ACKed packets in pipeline, receiver sends individual ACK for each packet, and sender maintains timer for each unACKed pkt. Sequence numbers: byte stream “number” of first byte in segment’s data. Acknowledgements: sequence number of the next byte in segment’s data. EstimatedRTT = (1 – α) \* EstimatedRTT + α \* SampleRTT. To estimate SampleRTT deviation from EstimatedRTT: DevRTT = (1–β)\*DevRTT+β\*|SampleRTT-EstimatedRTT|. TimeoutInterval = EstimatedRTT + 4 \* DevRTT. TCP creates reliable data transfer service: ignore duplicate ACKs, ignore flow control, congestion control. Flow control: receiver controls sender, so sender won’t overflow receiver’s buffer by transmitting too much, too fast. Before exchanging data, sender/receiver “handshake”: agree to establish connection & parameters. TCP congestion control: additive increase (increase cwnd by 1 MSS every RTT until loss detected) LastByteSent – LastByeAcked ≤ cwnd, multiple decrease (cut cwnd in half after loss) rate ~ cwnd/RTT bytes/sec. TCP slow start: exponentially increate rate until first loss event. Due to timeout: cwnd 🡪 1 and window grows exponentially. Due to duplicate ACKs, cwnd 🡪 cwnd/2 and goes into congestion avoidance (ssthresh = ½ of cwnd). Fast recovery: detect lost segments via duplicate ACKs.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedTwo key network-layer functions: forwarding (move packets from router’s input to appropriate router input) data plane, and routing (determine route taken by packets from source to destination) control plane. Two control-plane approaches: traditional routing algorithms (implemented in routers), and software-defined networking (SND) (implemented in (remote) servers). [physical layer] Green: line termination 🡪 [data link layer] blue: link layer protocol (receive) 🡪 [decentralized switching] lookup, forwarding, queuing 🡪 switch fabric. Decentralized switching: uses header field values, lookup port using forwarding table in input port memory. Destination-based forwarding: forwarding based only on destination IP address (traditional). Generalized forwarding: forward based on any set of header field values. Longest prefix matching: when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address. Three types of switching fabrics: memory (direct control of CPU, limited bandwidth), bus (limited by bandwidth), and crossbar (overcome bus bandwidth). Fabric slower than input ports combined 🡪 queuing may occur at input queues (queuing delay and loss due to buffer overflow). Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward. Scheduling mechanisms: FIFO, priority scheduling, Round Robin (RR) scheduling (scan queues, and send when complete), Weighted Fair Queuing (WFQ) (generalized RR). IP fragmentation, reassembly: network links have MTU (max transfer size), so large IP datagram is fragmented, IP header bits used to identify and reassemble, respecting order related fragments (the last packet fragmentation flag is set to 0). The difference between IPV4 and IPV6 is the bits (32 and 64 bits). IP address classes: A: 1.0.0.0 – 127.255.255.255 (subnet mask: 255.0.0.0 🡪 IP address /8), class B: 128.0.0.0 – 191.255.255.255 (subnet mask: 255.255.0.0 🡪 IP address /16), class C: 192.0.0.0 – 223.255.255.255 (subnet mask: 255.255.255.0 🡪 IP address /24). CIDR (Classless InterDomain Routing): address format: a.b.c.d/x. Subnet: device interfaces with same subset part of IP address, can physically reach each other without intervening router. The network portion is extended by splitting up the host number [network prefix | host number] 🡪 [network prefix | subnet number | host number].IP addresses are classed, difference in network IDs and host IDs. Effective way is subnetting, which uses netmaking. You can use /x (x is the network; the remaining bits is the host). For new networks, you can only have 2n-2 = IP address. DHCP (Dynamic Host Configuration Protocol): dynamically get address from server, allows reuse of address and support for mobile users. Negotiation with server and client uses broadcast. It happens and not specific IP address because clients that do not have an IP address needs to receive message through the MAC address, which is broadcasted out. Once the client receives it and responds to the DHCP request, it’s broadcasted again so that the other DHCP servers know that this IP address is taken (acknowledgement). Very economical (DHCP, subnetting, NAT). NAT (network address translation): all datagrams leaving local network have same single source NAT IP address, different source port numbers. Datagrams with source/destination in other network have their usual address for source, destination. Nobody has static IPs for people who use networks, so we have a very small number of IP address given out to users who join the network. This is for private addresses, where you already have an IP address and the router does IP switching, that switches and maps the port to the port address (IP spoofing). When the receiver receives the address, it switches once again in response to sending it to the host. Tunneling: IPv6 datagram carried as a payload in IPv4 datagram among IPv4 routers. This creates a new header and places it as a package and removes the package once it reaches another IPv6 router.

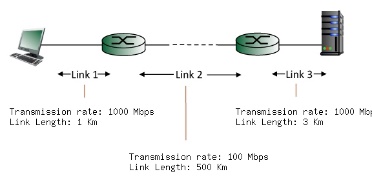
A close up of a device

Description automatically generatedA picture containing text, map

Description automatically generatedA close up of a map

Description automatically generatedData-link layer has the responsibility of transferring datagram from one node to physically adjacent node over a link. Services: framing, link access, flow control, error detection, error correction, half-duplex and full-duplex. MAC (LAN, physical, Ethernet) address is used “locally” to get frame from one interface to another physically connected interface. ARP (address resolution protocol): for each host, they have their own IP and MAC addresses (uses transmission within the LAN). ARP tries to build a table for each IP node on to have their own IP and MAC address, and TTL (time to live). ARP checks to see if it's in the same subnet: Host A to Host B: It finds its own IP network and sees destination ID and makes another mask. If it's the same, then it's in the same subnet. Then it looks at ARP. Otherwise, it sends out broadcast asking for the specific IP Address. ARP in same LAN (addressing): Host A sends to Host B (MAC address not in A’s ARP table), A broadcasts ARP query packet, containing B’s IP address (all nodes receive the ARP query), B receives ARP packet and replies to A with it’s (B’s) MAC address, caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out). To route to another LAN, ARP sends the MAC address of the router, which uses the IP address to follow up. ARP goes into 2 sections: sender then to router, and router to receiver. Random access protocols: is for nodes that have a packet to send, and it specifies how to detect collisions, and how to recover from collisions. CSMA (Carrier Sense Multiple Access): listens before transmitting; if a channel sensed idle, transmit entire frame, else defer transmission. Link layer uses FCFS. It is responsible for listening to see if the channel is busy. Collisions can still occur (due to propagation delay): propagation delay means two nodes may not hear each other’s transmission, and collision makes the entire packet transmission time wasted. CSMA/CD (collision detection) (Ethernet): carrier sensing, deferral as in CSMA, where collisions detected within short time, and colliding transmissions aborted, reducing channel wastage (once it detects collision, it cuts off transmission for both nodes, back off time is randomly generated). 802.11 LAN is wireless with hidden nodes. Signal may not get to the other end, so we use collision avoidance. Wireless hosts communicate with base stations (access point AP). Basic Service Set (BSS) in infrastructure contains: wireless hosts, access point (AP): base stations, and ad hoc mode: hosts only. 802.11 passive scanning: (1) beacon frames sent from APs, (2) association request frames sent: H1 to selected AP, (3) association response frame sent from selected AP to H1. 802.11 active scanning: (1) probe request frame broadcast from H1, (2) probe response frames sent from Aps, (3) association request from sent H1 to selected AP, (4) association frame sent from selected AP to H1. IEEE 802.11 MAC protocol: CSMA/CA: 802.11 sender: (1) if sense channel is idle for DIFS (Distributed Interframe Space), then transmit entire frame (no CD), (2) if sense channel is busy then start random back off time timer counts down while channel idle transmit when timer expires; if no ACK, increase random back off interval (repeat (2)). 802.11 receiver: if frame received is okay, return ACK after SIFS (Short Interframe Spacing) (ACK needed due to hidden terminal problem). Avoid collisions using small reservation packets. Host A transmits data, AP sends ACK that it has not been collided, and B receives. A is always sender; B is always receiver. Mobility approaches: let routing handle it (not scalable to millions of mobiles), let end systems handle it (indirect routing, direct routing). We should always keep our own IP address from network to network. All IPs are supposed to be routed together and belong together, so it's weird. Registration: Foreign agent knows about mobile, home agent knows location of mobile. home (ISP: laptop) and visited network (Foreign Agent, mobile): Ask foreign agent new network. Indirect routing: mobile uses two addresses (permanent address (used by correspondent; transparent) and care-of-address (used by home agent to forward datagrams to mobile)), triangle routing: correspondent-home-network-mobile (inefficient when in same network), ongoing connections may be maintained. Sender sends to home agent, home agent redirected. Sender sends back, directly to the sender. Direct routing: overcome triangle routing problem, non-transparent to correspondent: correspondent must get care-of-address from home agent. Messages sent directly to the foreign agent. Before sending, home agent responds saying this is my new IP address, so foreign agent can send it there.

Routing Algorithms: Dijkstra’s: one source to every node, Forwarding table: shortest-path tree (see destination and nodes), Distance vector algorithm: find each route and shared with immediate neighbors. Comparison of LS and DV algorithms; LS broadcast link info to all, and node can advertise incorrect link cost and each node computes only its own table. DV exchanges between neighbors only (convergence time varies), and DV node can advertise incorrect path cost, each node’s table used by others, and error propagate through network.

Consider the figure below, with three links, each with the specified transmission rate and link length

Find the end-to-end delay (including the transmission delays and propagation delays on each of the three links, but ignoring queueing delays and processing delays) from when the left host begins transmitting the first bit of a packet to the time when the last bit of that packet is received at the server at the right. The speed of light propagation delay on each link is 3x10\*\*8 m/sec. Note that the transmission rates are in Mbps and the link distances are in Km. Assume a packet length of 8000 bits. Give your answer in milliseconds.

Link 1 transmission delay = L/R = 8000 bits / 1000 Mbps = 0.008000 msec. Link 1 propagation delay = d/s = 1 Km / 3\*10\*\*8 m/sec = 0.003333 msec.

Link 2 transmission delay = L/R = 8000 bits / 100 Mbps = 0.080000 msec. Link 2 propagation delay = d/s = 500 Km / 3\*10\*\*8 m/sec = 1.666667 msec.

Link 3 transmission delay = L/R = 8000 bits / 1000 Mbps = 0.008000 msec. Link 3 propagation delay = d/s = 3 Km / 3\*10\*\*8 m/sec = 0.010000 msec.

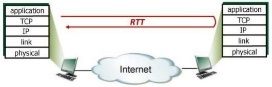
Thus, the total end-to-end delay is the sum of these six delays: 1.776000 msecs.

A screenshot of a cell phone

Description automatically generatedA close up of a device

Description automatically generatedConsider the figure below in which TCP a sender and receiver communicate over a connection in which the sender-to-receiver segments may be lost. The TCP sender sends initial window of four segments at t=1,2,3,4, respectively. Suppose the initial value of the sender-to-receiver sequence number is 123 and the first four segments each contain 575 bytes. The delay between the sender and the receiver is 7 time units, and so the first segment arrives at the receiver at t=8. As shown in the figure, two of the four segment(s) are lost between the sender and the receiver.   
Given the sequence numbers associated with each of the four segments sent by the sender. List the sequence of acknowledgements transmitted by the TCP receiver in response to the receipt of the segments received. Give the value in the acknowledgement field of each receiver-to-sender acknowledgement and give an explanation as to why that acknowledgement number value is being used.

TCP sequence numbers is based on the data (bit order). the first packet is 0, the second packet adds the size (ex: 0+512). When acknowledged, the sent back data is the last one, packet 2 being sent back is expecting (512 + 512). Sequence number: byte stream “number” of first byte in segment’s data. Acknowledgments number: sequence number of the byte expected from the other side (cumulative ACK). TCP allows piggybacking, both sequence and acknowledgement number. A receiver handles out-of-order segments by looking at the implementation, TCP spec doesn’t say.

Suppose that TCP's current estimated values for the round trip time (estimatedRTT) and deviation in the RTT (DevRTT) are 360 msec and 39 msec, respectively. Suppose that the next three measured values of the RTT are 260, 340, and 260 respectively. Compute TCP's new value of estimatedRTT, DevRTT, and the TCP timeout value after each of these three measured RTT values is obtained. Use the values of α = 0.125 and β = 0.25.

After the 1st estimatedRTT: estimatedRTT = 0.875\*360 + 0.125\*260 = 347.5 ms DevRTT = 0.75\*39 + 0.25\*(abs(260 - 347.5)) = 51.125 ms TimeoutInterval = 347.5 + 4\*51.125 = 552 ms

After the 2nd estimateRTT: estimatedRTT = 0.875\*347.5 + 0.125\*340 = 346.5625 ms DevRTT = 0.75\*51.125 + 0.25\*(abs(340 - 346.5625)) = 39.984375 ms TimeoutInterval = 346.5625 + 4\*39.984375 = 506.5 ms

After the 3rd estimateRTT: estimatedRTT = 0.875\*346.5625 + 0.125\*340 = 335.7421875 ms DevRTT = 0.75\*39.984375 + 0.25\*(abs(260 - 335.7421875)) = 39.984375 ms TimeoutInterval = 335.7421875 + 4\*48.92382812 = 531.4375 ms

A close up of a piece of paper

Description automatically generatedThe result of sending that flight of packets is that either (i) all packets are ACKed at the end of the time unit, (ii) there is a timeout for the first packet, or (iii) there is a triple duplicate ACK for the first packet. Give the times at which TCP is in slow start, congestion avoidance and fast recovery at the start of a time slot, when the flight of packets is sent. •Give the times at which the first packet in the sent flight of packets is lost, and indicate whether that packet loss is detected via timeout, by tripleduplicate ACKs. \*\*Timeout 🡪 ssthresh = 1, triplc ACK 🡪 sshtresh = 1/2ssthresh

when TCP starts, it has a slow start. (values: cwnd, ssthresh). Slow start is greater than congestion avoidance (exponential growth). Whenever there is a timeout, cwnd = 1. when n windows have been acknowledged, n = n + 1. Triple duplicate acknowledgement: out of order, it acknowledges the previous packet received in order. OR when packet has dropped. When this happens, fast recovery state occurs. This is supposed to be a very short term state. The result of sending that flight of packets is that either (i) all packets are at the end of the time unit, (ii) there is a timeout for the first packet, and there is a triple duplicate ACK for the first packet: slow start 🡪 1, ssthresh is 4, becomes slow start 🡪 There is another time out, the cwnd remains as 1,ssthresh becomes 2 🡪 slow start until ssthresh is reached, and then congestion avoidance occurs.

(Ch 3: P31) [4pts] Suppose that the five measured SampleRTT values (see Section 3.5.3 in textbook ) are 106 ms, 120ms, 140 ms, 90 ms, and 115 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, using a value of α=0.125 and assuming that the value of EstimatedRTT was 100 ms just before the first of these five samples were obtained. Compute also the DevRTT after each sample is obtained, assuming a value of β=0.25 and assuming the value of DevRTT was 5 ms just before the first of these five samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained.

After obtaining first SampleRTT 106ms: DevRTT = 0.75\*5 + 0.25 \* | 106 - 100 | = 5.25ms EstimatedRTT = 0.875 \* 100 + 0.125 \* 106 = 100.75 ms TimeoutInterval = 100.75+4\*5.25 = 121.75 ms

After obtaining 120ms: DevRTT = 0.75\*5.25 + 0.25 \* | 120 – 100.75 | = 8.75 ms EstimatedRTT = 0.875 \* 100.75 + 0.125 \* 120 = 103.16 ms TimeoutInterval = 103.16+4\*8.75 = 138.16 ms

After obtaining 140ms: DevRTT = 0.75\*8.75 + 0.25 \* | 140 – 103.16 | = 15.77 ms EstimatedRTT = 0.875 \* 103.16 + 0.125 \* 140 = 107.76 ms TimeoutInterval = 107.76+4\*15.77 = 170.84 ms

After obtaining 90ms: DevRTT = 0.75\*15.77 + 0.25 \* | 90 – 107.76 | = 16.27 ms EstimatedRTT = 0.875 \* 107.76 + 0.125 \* 90 = 105.54 ms TimeoutInterval = 105.54+4\*16.27 =170.62 ms

After obtaining 115ms: DevRTT = 0.75\*16.27 + 0.25 \* | 115 – 105.54 | = 14.57 ms EstimatedRTT = 0.875 \* 105.54 + 0.125 \* 115 = 106.72 ms TimeoutInterval = 106.72+4\*14.57 =165 ms

A screenshot of a social media post

Description automatically generated(Ch 4: P5) [4pts] Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows: a. Provide a forwarding table that has five entries, uses longest prefix matching, and forwards packets to the correct link interfaces. b. Describe how your forwarding table determines the appropriate link interface for datagrams with destination addresses: 11001000 10010001 01010001 01010101 ||| 11100001 01000000 11000011 00111100 ||| 11100001 10000000 00010001 01110111a) Prefix Match, Link Interface ||| 11100000 00, 0 ||| 11100000 01000000, 1 ||| 1110000, 2 ||| 11100001 1, 3 ||| otherwise, 3

b) Prefix match for first address is 5th entry: link interface 3 Prefix match for second address is 3nd entry: link interface 2 Prefix match for third address is 4th entry: link interface 3

(Ch 4: P8) [Extra credit 4pts] Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17/24. Also suppose that Subnet 1 is required to support at least 60 interfaces, Subnet 2 is to support at least 90 interfaces, and Subnet 3 is to support at least 12 interfaces. Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

We can divide 11011111.00000001.00010001.00000000/24 (223.1.17/24)into two networks with netmask 26 which are 1011111.00000001.00010001.00000000/26 (223.1.17.0/26) and 1011111.00000001.00010001.01000000/26 (223.1.17.64/26) Therefore 11011111.00000001.00010001.00000000 (223.1.17.0/26) with netmask 26 can contain 2^6 - 2 = 62 machines. This will form subnet 1 The network 1011111.00000001.00010001.01000000/26 (223.1.17.64/26) can have 2^6 - 2 = 62 machines which is too small for subnet 2 and too large for subnet 3. We can divide this network further for subet 3. For subnet 3 we only need 12 hosts, which means we need the last 4 bits for the host ID. Therefore the network can be 1011111.00000001.00010001.01000000/28 (223.1.17.64/28) 11011111.00000001.00010001.10000000/25 (223.1.17.128/25) will contain 2^7 - 2 = 126 machines. So this will be subnet 2. A further division of this subnet will give subnets with 2^6 - 2 = 62 machines which is not useable as subnet 2.

Subnetting table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Subnet | 1 | 2 | **4** | 8 | 16 | 32 | 64 | 128 | 256 |
| Host | 256 | 128 | **64** | 32 | 16 | 8 | 4 | 2 | 1 |
| Subnet Mask | /24 | /25 | **/26** | /27 | /28 | /29 | /30 | /31 | /32 |

Given network ID: 192.168.4.0**/24**, required to get 3 subnets: ^^^^^^^^^^^^^^^^ /26 is the new subnet mask for these 4 subnets, 64 host IDs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network ID | Subnet Mask | Host ID Range | # of usable host | Broadcast ID |
| (original) 192.168.4.**0** | /26 | 192.168.4.**1** – 192.168.4.**62** | 64-2 (network & broadcast ID) = 62 | 192.168.4.**63** |
| 192.168.4.**64** | /26 | 192.168.4.**65** – 192.168.4.**126** | 62 | 192.168.4.**127** |
| 192.168.4.**128** | /26 | 192.168.4.**129** – 192.168.4.**190** | 62 | 192.168.4.**191** |
| 192.168.4.**192** | /26 | 192.168.4.**193** – 192.168.4.**254** | 62 | 192.168.4.**255** |

A drawing of a face

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Forwarding table is then made with the shortest-path tree, seeing the destination and nodes to get there (destination, link)

Let dx(y): cost of least-cost path from x to y, then dx(y) = min {c(x,v) + dv(y)}

Expression = min taken over all neighbors v of x, c(x,y) = cost to neighbor v, dv(y) = cost from v to y

A screenshot of a cell phone

Description automatically generated