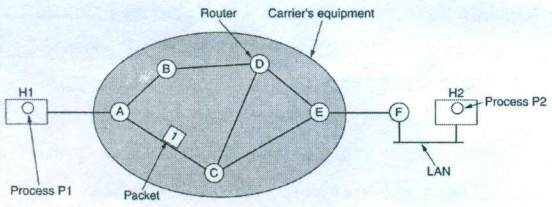
Chapter 5

The Network Layer

Network Layer Design Isues

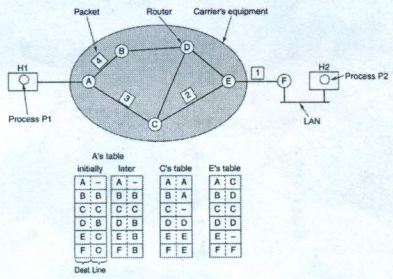
- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Subnets

Store-and-Forward Packet Switching



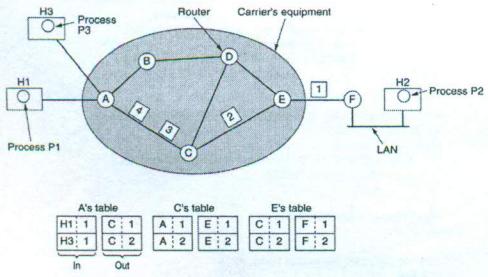
The environment of the network layer protocols.

Implementation of Connectionless Service



Routing within a diagram subnet.

Implementation of Connection-Oriented Service



Routing within a virtual-circuit subnet.

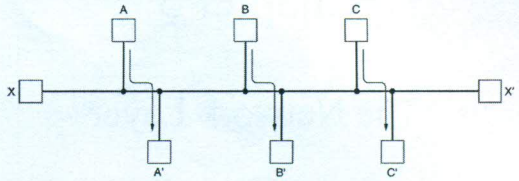
Comparison of Virtual-Circuit and Datagram Subnets

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Routing Algorithms

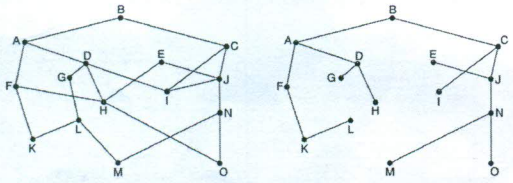
- The Optimality Principle
- Shortest Path Routing
- Flooding
- Distance Vector Routing
- Link State Routing
- Hierarchical Routing
- Broadcast Routing
- Multicast Routing
- Routing for Mobile Hosts
- Routing in Ad Hoc Networks

Routing Algorithms (2)



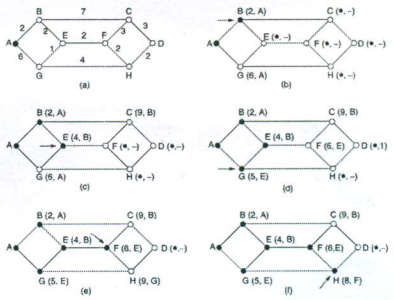
Conflict between fairness and optimality.

The Optimality Principle



(a) A subnet. (b) A sink tree for router B.

Shortest Path Routing



The first 5 steps used in computing the shortest path from A to D. The arrows indicate the working node.

Flooding

```
#define MAX_NODES 1024                  /* maximum number of nodes */
#define INFINITY 1000000000             /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];/* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                        /* the path being worked on */
      int predecessor;                  /* previous node */
      int length;                       /* length from source to this node */
      enum {permanent, tentative} label; /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                /* k is the initial working node */
```

Dijkstra's algorithm to compute the shortest path through a graph.

Flooding (2)

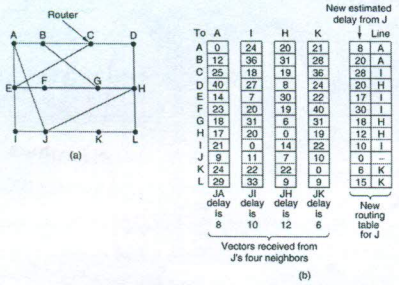
```
do {                                    /* Is there a better path from k? */
    for (i = 0; i < n; i++)             /* this graph has n nodes */
        if (dist[k][i] != 0 && state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
        }

    /* Find the tentatively labeled node with the smallest label. */
    k = 0; min = INFINITY;
    for (i = 0; i < n; i++)
        if (state[i].label == tentative && state[i].length < min) {
            min = state[i].length;
            k = i;
        }
    state[k].label = permanent;
} while (k != s);

/* Copy the path into the output array. */
i = 0;  k = s;
do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

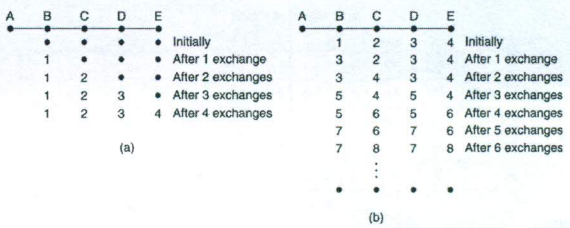
Dijkstra's algorithm to compute the shortest path through a graph.

Distance Vector Routing



(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

Distance Vector Routing (2)



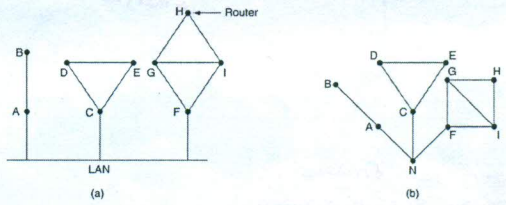
The count-to-infinity problem.

Link State Routing

Each router must do the following:

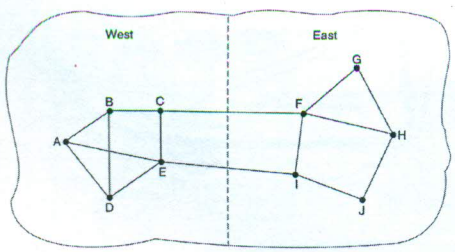
1. Discover its neighbors, learn their network address.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.

Learning about the Neighbors



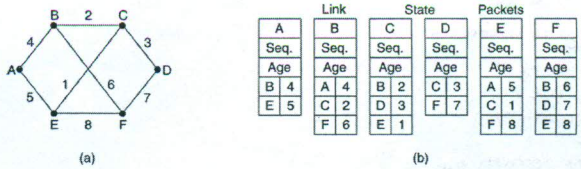
(a) Nine routers and a LAN. (b) A graph model of (a).

Measuring Line Cost



A subnet in which the East and West parts are connected by two lines.

Building Link State Packets



(a) A subnet. (b) The link state packets for this subnet.

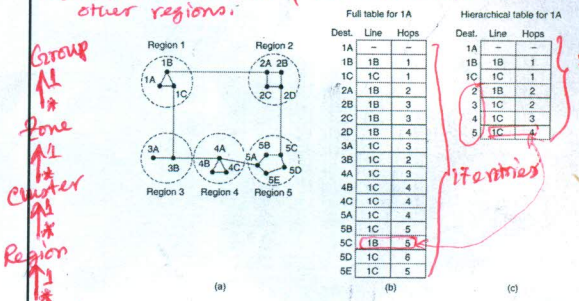
Distributing the Link State Packets

Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

The packet buffer for router B in the previous slide (Fig. 5-13).

Hierarchical Routing (Telephone network)

⇒ knows routers only within own region. No idea about routers of other regions.



Group ↑ Zone ↑ Cluster ↑ Region ↑ Router

7 entries

17 entries

Hierarchical routing.

Adv ⇒ Savings in table entry

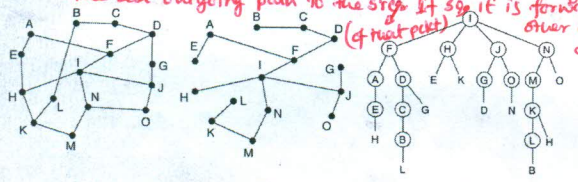
Penalty ⇒ Increased path length

Ex: for Group 5: (from 1A) through 1B → 5+6+6+7+7 = 31; through 1C → 4+5+5+6+6 = 26 ✓ (smaller)

However, from 1A to 5C: through 1B → 5; through 1C → 6

Broadcast Routing

Broadcast routing
- Distinct packet to each router (BW waste, demands a list of dest'n)
- Flooding (BW waste, too many packets)
- Multidest'n routing (each pckt carries a list of dest'n or a bitmap indicating desired dest'n; After getting such a pckt, a router checks whether at least one best route of any of the dest'n goes through it. If so, it creates a new copy listing the dest'ns to be reached for each best line)
- Spanning tree (needs a sink tree, which might not be available; for example, for DVR it is not available)
- Reverse path forwarding: After arrival of a packet, it is checked whether the pckt arrives through the best outgoing path to the src. If so, it is forwarded to all other lines, (if first pckt) otherwise, discarded)



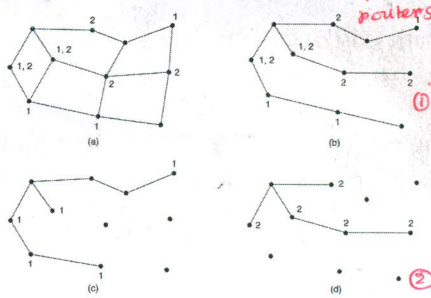
Adv. of Reverse Path forwarding:
1. No spanning tree is needed
2. No list of dest'n or bit map is needed
3. No special mechanism for stopping packet flow

Reverse path forwarding. (a) A subnet. (b) a Sink tree. (c) The tree built by reverse path forwarding.

Multicast Routing

(gno ⇒ group)

⇒ Each router computes a spanning tree covering all other routers, and subsequently prunes it.



Pruning:
① Method 1:
- for LSR
- Each router knows complete topology including which router in which group
- Process starts from end of path to root, and prunes routers that are not in desired gr.

② Method 2:
- For DVR
- Basic algo: Reverse path forwarding
- Process: when a router, with no host in desired gr and no other conn'n to other routers, receives a multicast msg, it replies with a PRUNE msg. If another router with no group member among its hosts receives such a PRUNE msg, it also replies with a PRUNE msg.

(a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

Disadv. of Method 2: Scalability

# of trees needed to be stored = m × n (m = avg # of members in a gr, n = # of gr.s)

Method 3: To alleviate the scalability problem of Method 2
- have only one spanning tree for a gr, instead of having a spanning tree for all members of the group.
- Root of the only sp tree is near the middle of the gr. To send a multicast msg, a host sends a msg to core (core) which then does multicasting along the sp tree (may not optimal for all srcs).

Routing for Mobile Hosts

Mobile host (away from home, still wants to be connected)
- Migrating hosts: Basically stationary hosts who move from one fixed site to another
- Roaming hosts: compute on the run and want to maintain conn'ns as they move around

* Home agent: 1 in an area (LAN or wireless cell); keeps track the hosts whose home is in the area, but who are currently visiting another area
* Foreign agent: 1 or more in one area; which keep track all hosts visiting the area
* After entering to a new area, a host needs to register
* Registration procedure:

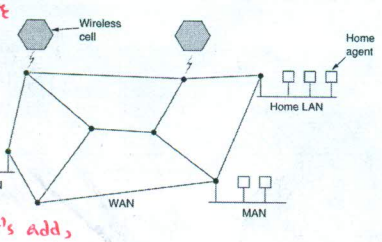
search for foreign agent (by for'n agent's periodic msg or broadcasting from the host)

⇓

Mobile host registers with the foreign agent giving its (Home addr, DLL addr, and sec'n security info)

⇓

foreign agent contacts the home agent with (for'n agent's add, security info)



A WAN to which LANs, MANs, and wireless cells are attached.

⇓

home agent examines the security info, which contains a timestamp indicating freshness of the info

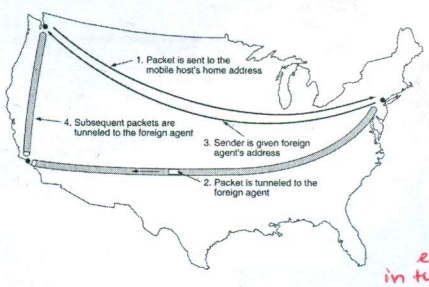
⇓

foreign agent gets an acknowledgement from home agent, it makes an entry in its table, and informs the mobile host that it is now registered.

⇓

Subsequently, packets are routed directly to host via the foreign agent bypassing the home loca'n entirely.

Routing for Mobile Hosts (2)



Packet routing for mobile users.

To send a pckt to a mobile host, sent the pckt to the home LAN of the host

⇓

The home agent then looks for the mobile host's new loc'n & finds out address of the foreign agent handing the mobile host

⇓

encapsulate the packet in the payload field of an outer pckt, and send it to foreign agent (tunneling)

⇓

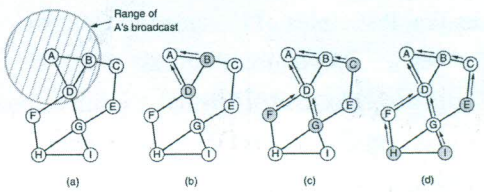
The home agent thereafter informs the sender that subsequent msg exchange can be directly enabled through

Routing in Ad Hoc Networks

Possibilities when the routers are mobile:

1. Military vehicles on battlefield.
   - No infrastructure.
2. A fleet of ships at sea.
   - All moving all the time
3. Emergency works at earthquake .
   - The infrastructure destroyed.
4. A gathering of people with notebook computers.
   - In an area lacking 802.11.

Route Discovery



a) (a) Range of A's broadcast.
b) (b) After B and D have received A's broadcast.
c) (c) After C, F, and G have received A's broadcast.
d) (d) After E, H, and I have received A's broadcast.

Shaded nodes are new recipients. Arrows show possible reverse routes.

Route Discovery (2)

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count

Format of a ROUTE REQUEST packet.

Route Discovery (3)

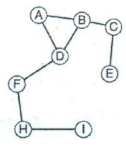
Source address	Destination address	Destination sequence #	Hop count	Lifetime

Format of a ROUTE REPLY packet.

Route Maintenance

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

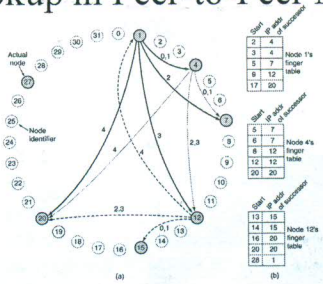
(a)



(b)

(a) D's routing table before G goes down.
(b) The graph after G has gone down.

Node Lookup in Peer-to-Peer Networks



(a) A set of 32 node identifiers arranged in a circle. The shaded ones correspond to actual machines. The arcs show the fingers from nodes 1, 4, and 12. The labels on the arcs are the table indices.
(b) Examples of the finger tables.

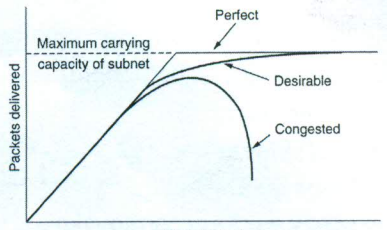
* Congest^n control: Ensures that the subnet is able to carry the offered traffic
* flow control: (point-to-point traffic betⁿ sender & recvr), Ensures that a faster sender cannot continually xmit data faster than the recvr is able to receive (through some direct feedback from recvr).

Cause of congest^n:
① memory: Insufficient or infinity
② Processor: Slow
③ Line: Low BW

Congestion Control Algorithms

- General Principles of Congestion Control
- Congestion Prevention Policies
- Congestion Control in Virtual-Circuit Subnets
- Congestion Control in Datagram Subnets
- Load Shedding
- Jitter Control

Congestion



When too much traffic is offered, congestion sets in and performance degrades sharply.

Guess: what happ would happen if routers have infinite memory?
— Congest^n will get worse, not better. Because, by the time pckts get to the front of the Q, they have already timed out (repeatedly) and duplicates have been sent. All these pckts will be dutifully forwarded to the next router, increasing load all way to the dest^n.
— No drop at all due to having available Q size

Open loop: Solves the problem by good design
- when to accept new traffic
- when to discard pckts and which ones
- making scheduling decisions at various points in the network

Closed loop: Based on the concept of a feedback loop

General Principles of Congestion Control

Closed loop:

1. Monitor the system.
   - detect when and where congestion occurs.
2. Pass information to where action can be taken.
3. Adjust system operation to correct the problem.

metrics:
① % of pckts discarded due to lack of buffer space
② Avg Q length
③ # of pckts timeout & retransmitted
④ Avg pckt delay
⑤ Stdev of pckt delay

increase implies congest^n increasing

① Pckt sent to src announcing the problem
② fill reserved bit or field in all outgoing pckts if from a router under congest^n to warn all neighbors
③ Hosts or routers periodically send probe pckts out to explicitly ask about congest^n

Congestion Prevention Policies

immediate ack of each pckt: extra traffic
Ack piggyback in reverse traffic: extra timeout & retx

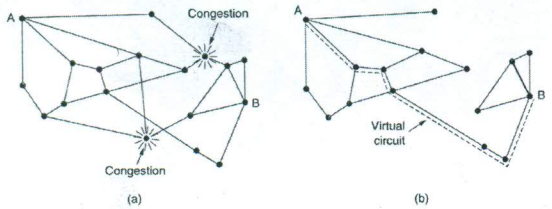
Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

tight flow control reduces data rate and reduce congest^n
extra pckt retx if too short, longer response time if too long
Q per input line / Q per output line / Q per priority based / round robin

Policies that affect congestion.

Admission control: No more VC, once congest^n is signaled
Carefully route all new virtual circuits around problem areas. (as shown in fig)
Reserv^n (negotiate an agreement between the host and subnet when a virtual ckt is set up)

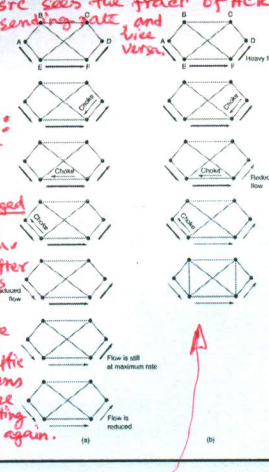
Congestion Control in Virtual-Circuit Subnets



(a) A congested subnet. (b) A redrawn subnet, eliminates congestion and a virtual circuit from A to B.

Congestion Control in Datagram Subnet

- Actions are made based on line utiliza, u (whereas u > threshold, then it's a warning state)
  - $u_{new} = a u_{old} + (1-a) f$
    - a: constant; determines how fast the router forgets recent history
    - f: instantaneous line utilization
- 2 methods in warning state
  - Indirect method (warning bit): sets a special bit in header and relay. After reaching dest^n, the set bit is copied in ACK and sent back to src. When the src sees the frame of ACK with this bit set increasing, it drops sending rate and vice versa.
  - Hop-by-Hop Choke Packets
    - Direct method (choke pckt): router sends a choke pckt back to src host, giving it the dest^n found in the choke pckt. The original pckt is tagged so that it generates no more choke pckt farther in the path. The src reduces tx rate by X% after getting a choke pckt and ignores for an interval ignoring any more choke pckt during that interval. The same process of reducing traffic happens in each hop it passes through. Increment: less
      - reduce^n: 50% → 25% → ....
    - Hop-by-Hop choke pckt: choke pckt takes effect at every hop it passes through.
      - Adv: Quick relief at the point of congest^n
      - Disadv: Using up more buffers upstream.



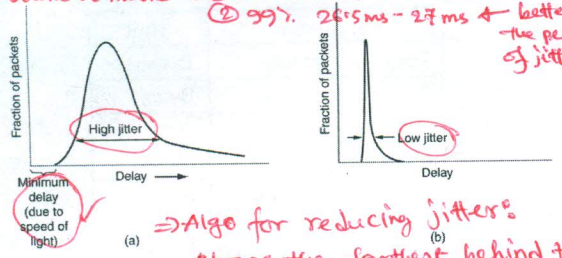
(a) A choke packet that affects only the source.
(b) A choke packet that affects each hop it passes through.

* load shedding: throw away pckts that cannot be handled. which one to throw away?
  - file: Old is better than new (wine) } 2 policies
  - Voice: New " " " old (milk) }
* Random Early Detection (RED): Discards pckts before exhausting the whole buffer space.
  - If avg q length (a running avg) becomes higher than a threshold, then it starts dropping pckt
  - For dropping, it picks pckts at random as it cannot know which source is causing the most trouble

Jitter Control

Std dev of pckt arrival times or q delay

Ex: sound or movie → ① 50%. 20ms, 50%. 30ms
② 99%. 25.5ms - 27ms ← better from the perspective of jitter



(a) High jitter. (b) Low jitter.

⇒ Algo for reducing jitters:
choose the farthest behind the schedule pckt competing for output, and get it next

Quality of Service

- Requirements
- Techniques for Achieving Good Quality of Service
- Integrated Services
- Differentiated Services
- Label Switching and MPLS

Not covered

⇒ Techniques for achieving good QoS:
Soln 1: Overprovisioning ⇒ Expensive

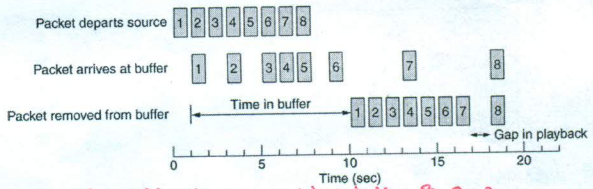
Requirements

4 parameters for characterizing flows

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

How stringent the quality-of-service requirements are.

Soln 2: Buffering

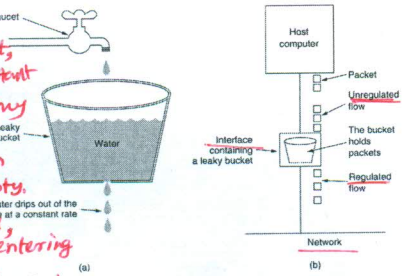


- No effect on reliability & BW
- Delay ↑, jitter smoother
- Not applicable for real-time services

Smoothing the output stream by buffering packets.

Soln 3: Traffic shaping: Smooths out traffic on server side, rather than on client side; Regulates the avg. rate (burstyness) of traffic. (Sliding window → amount of data in Xsit)
⇒ User and subnet agree on a certain traffic pattern at the time of conn setup (service-level agreement)
⇒ Monitoring a traffic flow (traffic policing)
- Easier in VC; However, in datagram subnet, it is implemented in TL

Soln 4: The Leaky Bucket Algorithm



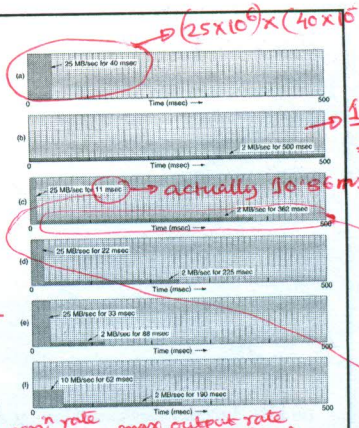
(a) A leaky bucket with water. (b) a leaky bucket with packets.

⇒ No matter at which water enters the bucket, the output flow is at constant rate ρ when there is any water in the bucket. Output flow is 0 when the bucket is ~~full~~ empty. Once the bucket is full, any additional water entering it spills over the sides and is lost.

⇒ Uneven flow from user process
↓
Even flow to network
+
Smoothing out burst
+
Reduce the chance of congestion

⇒ 2 variants
- 1 pckt per clock tick to network (all same-size pckt)
- byte variant for variable size pckt
  - → counter is incremented by n in each clock tick
  - → if counter > size of the first pckt in q, then xmit it and counter ← counter − size of transmitted bytes

The Leaky Bucket Algorithm



(a) Input to a leaky bucket. (b) Output from a leaky bucket. Output from a token bucket with capacities of (c) 250 KB, (d) 500 KB, (e) 750 KB. (f) Output from a 500KB token bucket feeding a 10-MB/sec leaky bucket.

$(25\times10^6)\times(40\times10^{-3}) = 1\times10^6$ Total

$\frac{1\times10^6 - 0}{2\times10^6} = 500$ ms

actually 10.86 ms

capacity $C + \rho S = M S$ (token gen rate, burst length → max output rate, burst length (time))

Added capacity due to token generation during C transmission

Burst length, $S = \frac{C}{M-\rho} = \frac{250\times10^3}{(25-2)\times10^6} \approx 11$ ms

Dura^n after burst $= \frac{\text{Total} - MS}{\rho} = \frac{1\times10^6 - 25\times10^6 * 11\times10^{-3}}{2\times10^6} \approx 363$ ms

Capacity added due to new tokens generated (for capacity C)

25 KB	25 KB	.....	25 KB	25 KB
①	②		⑩	⑪

due to adding tokens in the previous bursty slots (Actually should be ~ 2KB×10 = 20KB)

Varia^n : Token bucket ⇒ Allows a level of burstiness.
⇒ The leaky bucket holds tokens, generated by a clock at the rate of one token every ΔT sec.
→ Each pckt tx consumes a token
→ At the beginning, all tokens in the bucket can be captured, and thus a burst equivalent to bucket capacity can be allowed.
→ In case of bucket being filled up, tokens (NOT pckts) are discarded.
⇒ pckt variant: counter + = 1 ← token corresponding to 1 pckt
⇒ byte variant: counter + = k ← token corresponding to k bytes
(If enough token corresponding to the ~~first~~ # of bytes of the 1st pckt in the bucket is available, then the pckt will be transmitted)

Trade off : Smoother traffic → Token bucket then Leaky bucket

Token bucket < Rate leaky bucket < Max^n rate of network

Smoother traffic

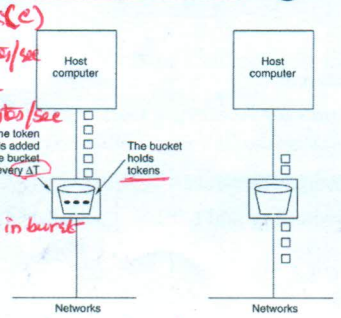
The Token Bucket Algorithm

* Token bucket capacity (C)
Token arrival rate → ρ bytes/sec
Burst length → S sec
Max output rate → M bytes/sec
Now, max^n # of bytes in an output burst →

$C + \rho S = M S$

(# of bytes in burst = # of bytes in burst)

So, $S = \frac{C}{M - \rho}$



for $C = 250$ KB, $M = 25$ MB/s, $\rho = 2$ MB/s,

$S = \frac{250 \times 10^3}{(25-2) \times 10^6} \approx 11$ ms

(a) Before. (b) After.

Durat^n after a burst = $\frac{\text{total} - MS}{\rho} = \frac{1 - 25 \times 0.011}{2} \times 363$ ms

(for total = 1MB)

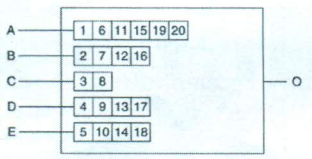
Admission Control

→ rate is less than max^m network rate, however, greater than token generat^n rate.

Parameter	Unit
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

An example of flow specification.

Packet Scheduling

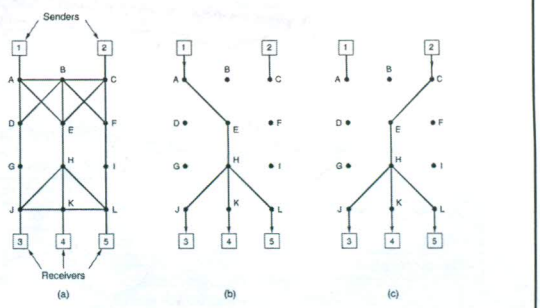


Packet	Finishing time
C	8
B	16
D	17
E	18
A	20

(b)

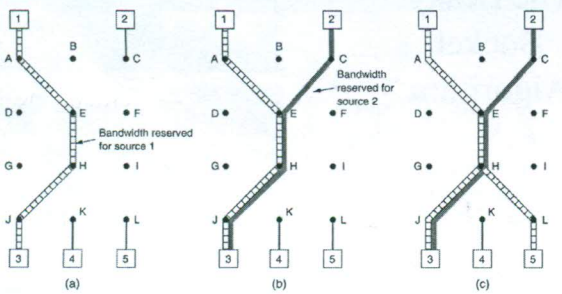
(a) A router with five packets queued for line O.
(b) Finishing times for the five packets.

RSVP-The ReSerVation Protocol



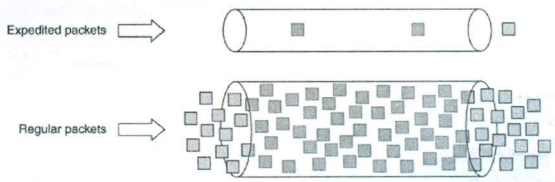
(a) A network, (b) The multicast spanning tree for host 1.
(c) The multicast spanning tree for host 2.

RSVP-The ReSerVation Protocol (2)



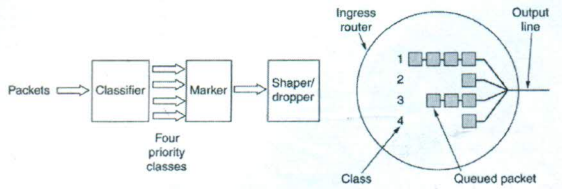
(a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel, to host 2. (c) Host 5 requests a channel to host 1.

Expedited Forwarding



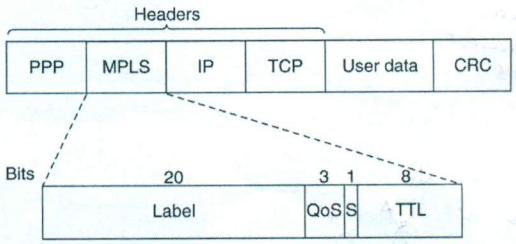
Expedited packets experience a traffic-free network.

Assured Forwarding



A possible implementation of the data flow for assured forwarding.

Label Switching and MPLS

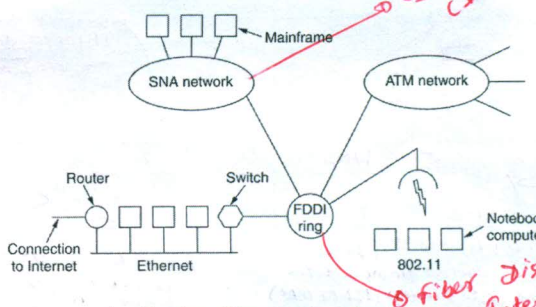


Transmitting a TCP segment using IP, MPLS, and PPP.

Internetworking

- How Networks Differ
- How Networks Can Be Connected
- Concatenated Virtual Circuits
- Connectionless Internetworking
- Tunneling
- Internetwork Routing
- Fragmentation

Connecting Networks



A collection of interconnected networks.

Systems Network Architectures (IBM's proprietary networking architecture)

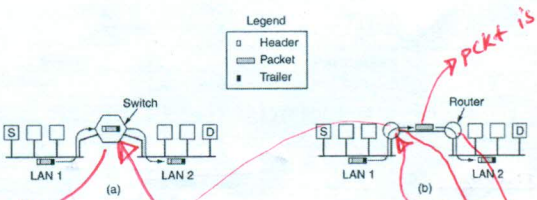
fiber Distributed Data Interface (acts as a backbone)

How Networks Differ

Item	Some Possibilities
Service offered	Connection oriented versus connectionless
Protocols	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Addressing	Flat (802) versus hierarchical (IP)
Multicasting	Present or absent (also broadcasting)
Packet size	Every network has its own maximum
Quality of service	Present or absent; many different kinds
Error handling	Reliable, ordered, and unordered delivery
Flow control	Sliding window, rate control, other, or none
Congestion control	Leaky bucket, token bucket, RED, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, by packet, by byte, or not at all

Some of the many ways networks can differ.

How Networks Can Be Connected



(a) Two Ethernets connected by a switch.
(b) Two Ethernets connected by routers.

pckt is transmitted

Entire frame is xpotted

extracts pckt from the incoming frame

puts incoming pckt to a frame's body

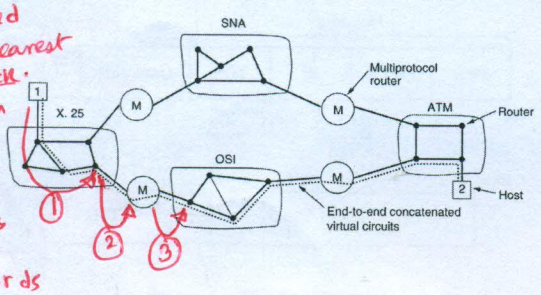
Can handle multiple protocols (multiprotocol router)

Aspect	Virtual Ckts	Connectionless
Sequencing	Guaranteed ✓	No guarantee
Delayed duplicate pckts	Avoided ✓	Can arrive
Table space for each conn^n	Needed	Not needed ✓
Alternate route in case of congest^n	Not available	Available ✓
Robustness in case of route failure	No	Yes ✓
Applicability	Only when VCs are available	Always ✓

Concatenated Virtual Circuits

Process

① VC is established to the router nearest the dest^n network.
② Another VC from that router to an external gateway (multi-protocol router) is established.
③ Gateway records existance of this VC in a table and then constructs another VC to a router in the next subnet.



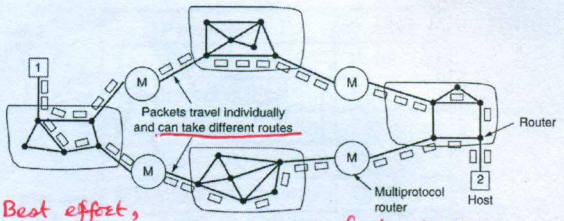
Internetworking using concatenated virtual circuits.

↓ continues

⇒ Network never reorders pckts in a flow as all data pckts do follow the same seq of gateways

All reliable or All not reliable } Better (mixture is not good)

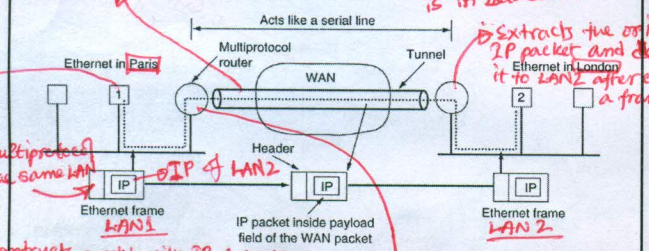
Connectionless Internetworking



- Best effort, just injects pckt and hopes for the best
- Pckts of same flow may not traverse same seq of gateways (So, ordered delivery is not guaranteed)
- Multiple routes. A connectionless internet can be used, So, higher bandwidth than concatenated VCs

Tunneling

A big tunnel extending from one multi-protocol router to another



Tunneling a packet from Paris to London.

Src & dest^n are of same type of network, however, different type of network is in between them

Extracts the original IP packet and delivers it to LAN2 after creating a frame of it.

addr of multiprotocol router of the same LAN

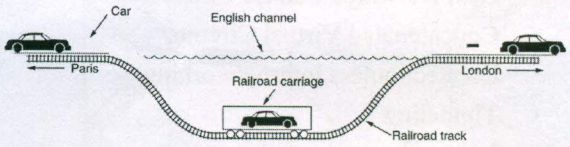
IP of LAN2

① Constructs a pckt with IP of LAN2
② Constructs an Ethernet frame with the pckt having dest^n address (DLL or MAC) of multiprotocol router of the same LAN.
③ Put the frame to ethernet

Removes IP pckt from the frame, Insert the pckt into WAN network layer pckt. The new pckt is addressed to the multiprotocol router of the next LAN.

Real-world

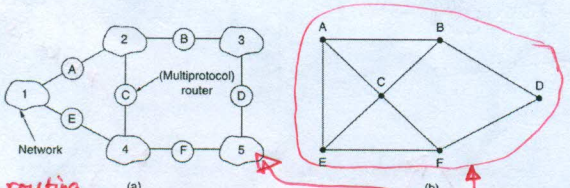
Tunneling (2)



Tunneling a car from France to England.

This slide will come earlier

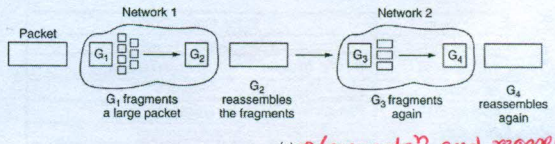
Internetwork Routing



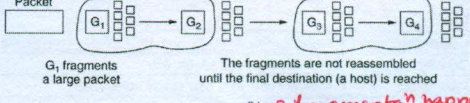
(a) An internetwork. (b) A graph of the internetwork.

2 levels of routing
- Within each network: interior gateway protocol
- Between the networks: exterior " "

Fragmentation



(a) ⇒ fragmentn and reassembling happen in all networks in an independent manner



(b) ⇒ fragmentn happens only in the first network. Reassembling happens only in the last network. [all fragmented pckts are treated independently as an IP pckt]

(a) Transparent fragmentation. (b) Nontransparent fragmentation.

Transparent to all networks

Transparent fragmentn

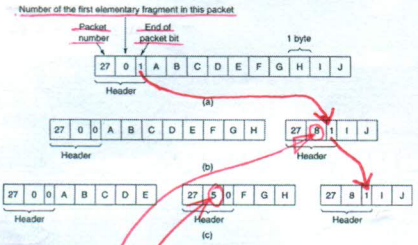
Disadvantages

① Each exit gateway must know whether it has received all the fragmented pckts or not. So, it needs to have a "counter" or "end of pckt" field.
② Each pckt follows same exit gateway having no provision of following disjoint paths. So, performance decreases.
③ Overhead of doing fragmentn & reassembling in each hop.

Nontransparent fragmentn

- Every host must be capable of reassembling
- Each pckt needs to have separate headers. So, the overhead remains in the journey.

Fragmentation (2)



Fragmentation when the elementary data size is 1 byte.
(a) Original packet, containing 10 data bytes.
(b) Fragments after passing through a network with maximum packet size of 8 payload bytes plus header.
(c) Fragments after passing through a size 5 gateway.

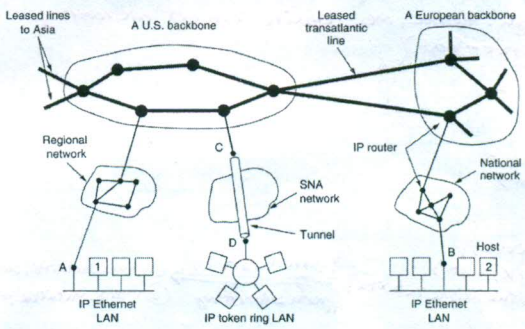
The Network Layer in the Internet

- The IP Protocol
- IP Addresses
- Internet Control Protocols
- OSPF – The Interior Gateway Routing Protocol
- BGP – The Exterior Gateway Routing Protocol
- Internet Multicasting
- Mobile IP
- IPv6

Design Principles for Internet

1. Make sure it works.
2. Keep it simple.
3. Make clear choices.
4. Exploit modularity.
5. Expect heterogeneity.
6. Avoid static options and parameters.
7. Look for a good design; it need not be perfect.
8. Be strict when sending and tolerant when receiving.
9. Think about scalability.
10. Consider performance and cost.

Collection of Subnetworks

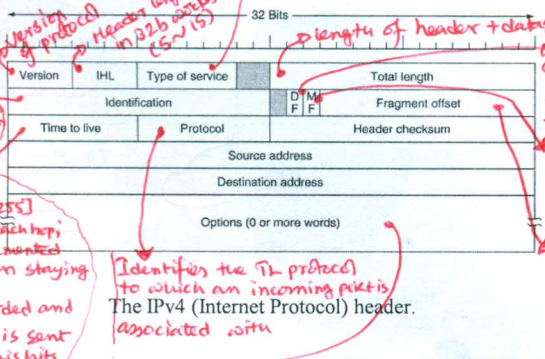


The Internet is an interconnected collection of many networks.

Distinguishes among various types of services
precedence indicates priority; 0 → Normal, 7 → Network control pckt
D T R → Reliability, Throughput, Delay
Indicates what is care most

The IP Protocol

Header ⇒ 20 B fixed part + Variable length optional part (max header size is 15 words = 60 B)



The IPv4 (Internet Protocol) header.

At dest.ⁿ, determines to which datagram the newly-arrived fragment belongs to
All fragments from same datagram have same identifier
A counter that limits lifetime of a pckt. [max val = 255] [Decremented in each hop; Also can be decremented in a router when staying long] [pckt gets discarded and a warning pckt is sent to src, when this hits zero]
Identifies the TL protocol to which an incoming pckt is associated with
Length of header + datagram (max 65,535 B)
Don't fragment (Orders the router not to fragment as the dest.ⁿ is not capable of reassembling)
More fragments (All fragments except the last one has this bit set 1)
The pos.ⁿ at which the current fragment belongs to in the datagram

The IP Protocol (2)

Option	Description
Security	Specifies how secret the datagram is → Military purpose
Strict source routing	Gives the complete path to be followed → helps when routing table is corrupted and when timing is measured
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Some of the IP options.

Also for debugging routing algo
to the opt field
for bug tracking by system managers
If any country is must or must not be traversed (for political or economical reasons)

Variable length: 1 B | 1 B | variable
identify the option
length of option
data
Pad (padding to make options as a multiple of 4 bytes)

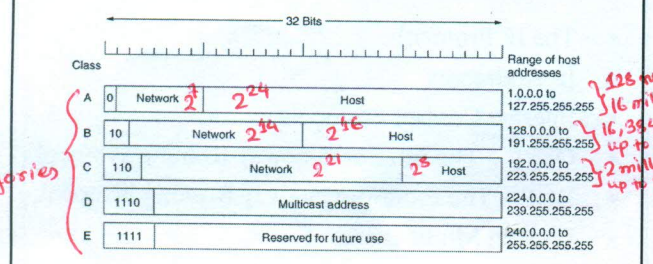
⇒ Big endian order of tx (from left to right, with higher order bit of "version" first)
(SPARC ⇒ Big Endian; Pentium ⇒ Little endian)

Network | Host

All 0's indicate this network — All 1's indicate all hosts

IP Addresses

No two nyes in the Internet can have the same IP addr.



128 networks, 16 million hosts each
16,384 networks, up to 64K hosts
2 million networks, up to 256 hosts each

5 categories

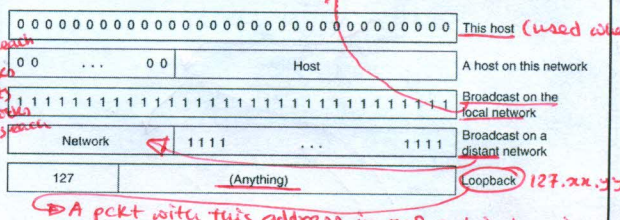
IP address formats. (classful addressing)

Network numbers are managed by a nonprofit org? called ICANN (Internet Corporation for Assigned Names and Numbers)

IP Addresses (2)

0 → This network

-1 (all 1's) ⇒ All hosts on the local network

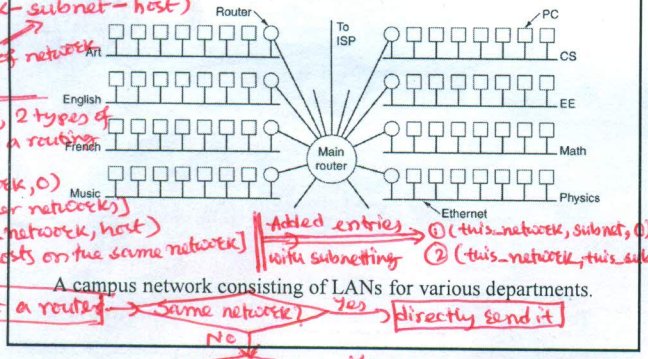


⇒ A pckt with this address in src addr put in the wire, rather it is processed locally by treating it as an incoming pckt. In this way, a sender can send a pckt without knowing address of the local network. [can be used in testing, for example, whether TCP/IP stack is configured properly or not in a m/c]

Special IP addresses.

⇒ In case of increasing # of depts, using the structure of (network-host) might not be enough. For example, let a university starts with a network with class B. If the # of networks (say a network for each new dept.) grows, and all connects to the same network using a repeater, then the max # of repeaters per ethernet (i.e., 4) may reach very quickly. So, 2-level hierarchy will not be enough. Hence, we require 3-level hierarchy. (Network-subnet-host)

Subnets



A part of network

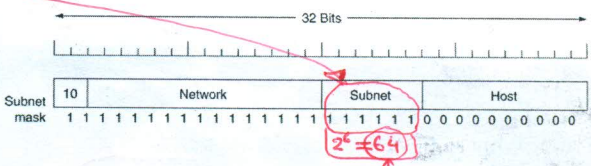
Normally, 2 types of entries in a routing table —
① (network, 0) [for other networks]
② (this_network, host) [for hosts on the same network]

Added entries with subnetting → ① (this_network, subnet, 0) ② (this_network, this_subnet, host)

A campus network consisting of LANs for various departments.

pckt at a router → Same network? Yes → directly send it
No → Dest? known? Yes → forward to next router
No → forward to a default router having more extensible table

Subnets (2)



A class B network subnetted into 64 subnets. [for a university having 35 departments]

⇒ 2 ways of subnetting and dispatching a pckt after being arrived at a main router

① Lookup a 65,535-entry table and decide (for class B)
Disadv.:
- large table
- lot of manual maintenance when new host is added or removed

② Create a subnet # by taking some bits from host addr.

⇒ class A : 16 million hosts [too big]
⇒ class C : 256 hosts [too small]
⇒ class B : 65,536 hosts [just right]

Three Bears problem [fixed-size block] [So, needs the classless one]

→ Classful:

Copy of IP addr → Right shift → Class # (4 bits) → Mask off → Network # → Right align to 32 bits → Table lookup

16 possibilities: 8 for class A, 4 for class B, 2 for class C, 1 each for class D, class E

CIDR – Classless InterDomain Routing

(CIDR)

→ Allocates IP addresses in variable-size blocks, without regard to the classes

→ such as /21, /22, ...

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

2^{32-21}

2^{32-22}

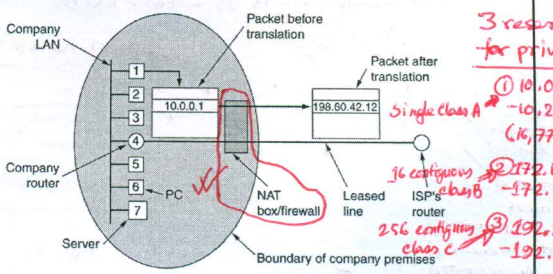
A set of IP address assignments.

→ Classless:

pckt arrive → Extract IP dest? addr (net+host) full addr → Scan routing table entry-by-entry after masking → Take the entry which has the longest match

⇒ Routing table:
classful: Network # | Line
classless: Mask (32 bits) | Network # | Line

NAT – Network Address Translation



Placement and operation of a NAT box.

3 reserved ranges for private IPs

① 10.0.0.0 – 10.255.255.255/8 — single class A (16,777,216 hosts)
② 172.16.0.0 – 172.31.255.255/12 — 16 contiguous class B
③ 192.168.0.0 – 192.168.255.255/16 — 256 contiguous class C

for example, for an address 194.24.28.1 (11000010 00011000 00011100 00000000) will match to R (even though having a match with P)

P: 11000010 00011000 00010000 00000000
Q: 11000010 00011000 00001000 00000000
R: 11000010 00011000 00011000 00000000

(/20) (/21) (/21)

longest match

ARP ⇒ Broadcasts a pckt asking who is the owner of the IP address under searching. When owner of the IP address gets it, it replies with its own ethernet addr.

3/25/2017

[A solution without keeping all mappings of (IP, Ethernet) addrs]

Adv. of ARP → Simplicity (requires only storing IP addr & subnet mask)

Optimizations in ARP:

① caching results obtained by an ARP request.

② Sending own mapping with ARP request such that others can cache it.

③ Send own mapping during booting. It enables storing the mapping in others' caches. If the mapping (IP) gets matched with another, then the new one will not be.

** To allow changes, ARP caches timeout after few minutes.

Internet Control Message Protocol

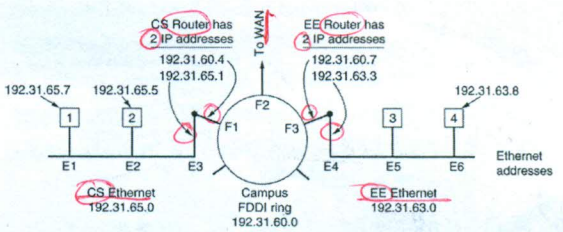
Unexpected events are reported using ICMP

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo request	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Used when the subnet of a router cannot locate the dest? or a pckt with DF bit cannot be delivered because a "small-pckt" network stands in the way

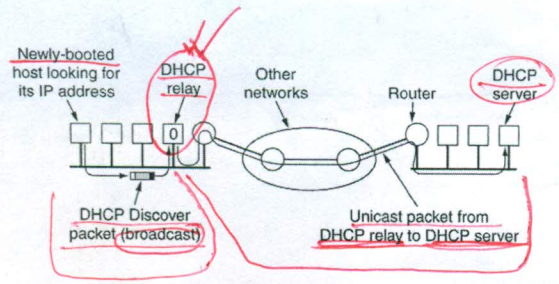
The principal ICMP message types.

ARP– The Address Resolution Protocol



Three interconnected /24 networks: two Ethernets and an FDDI ring.

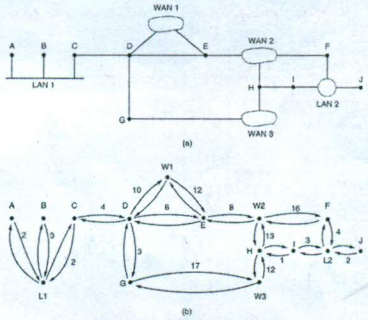
Dynamic Host Configuration Protocol



Operation of DHCP.

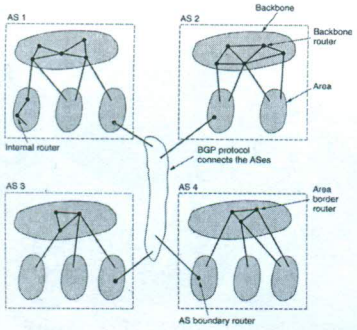
DHCP Relay agent intercepts broadcast msg from a newly-booted m/c and unicasts that to DHCP server, which might be in a distant LAN.

OSPF – The Interior Gateway Routing Protocol



(a) An autonomous system. (b) A graph representation of (a).

OSPF (2)



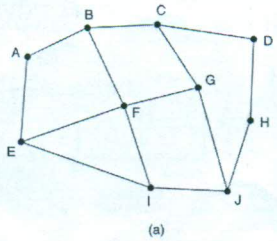
The relation between ASes, backbones, and areas in OSPF.

OSPF (3)

Message type	Description
Hello	Used to discover who the neighbors are
Link state update	Provides the sender's costs to its neighbors
Link state ack	Acknowledges link state update
Database description	Announces which updates the sender has
Link state request	Requests information from the partner

The five types of OSPF messeges.

BGP – The Exterior Gateway Routing Protocol



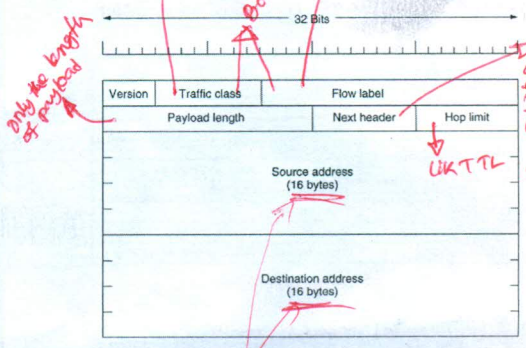
Information F receives from its neighbors about D

From B: "I use BCD"
From G: "I use GCD"
From I: "I use IFGCD"
From E: "I use EFGCD"

(b)

(a) A set of BGP routers. (b) Information sent to F.

The Main IPv6 Header



The IPv6 fixed header (required).

Extension Headers

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

IPv6 extension headers.

Extension Headers (2)

Next header	0	194	4
Jumbo payload length			

The hop-by-hop extension header for large datagrams (jumbograms).

Extension Headers (3)

Next header	Header extension length	Routing type	Segments left
Type-specific data			

The extension header for routing.