

[illegible]

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

00-1

/

---

- 01 Introduction
- 02 Physical Layer
- 03 Data Link Layer
- 04 MAC Sublayer
- 05 Network Layer
- 06 Transport Layer
- 07 Application Layer
- 08 Network Security

# Network Layer

**Main service:** Provide facilities for getting data from a source to a destination ⇒ **routing**.

Again, distinguish between connectionless and connection-oriented services. There are two *implementation* techniques:

- **Virtual circuits** are complete routes that are set up in advance.
- **Datagrams** comprise individual packets of which the route is determined on the fly: they hop from router to router.

**Note:** The services are independent of their implementation:

	Datagram	Virtual circuit
CL	UDP over IP	UDP over IP over ATM
CO	TCP over IP	ATM AAL1 over ATM

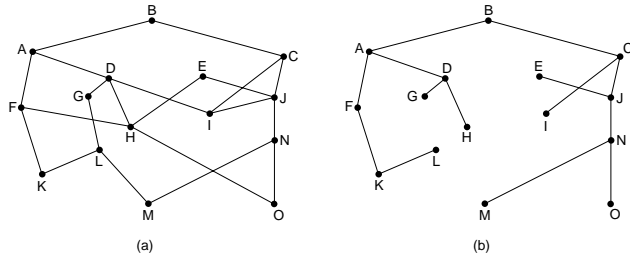
## Comparison

Issue	Datagram	Virtual circuit
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State info	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
QoS	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

**Main issue:** Routers that constitute the network layer of a network, should cooperate to find the **best routes** between all pairs of stations.

**Observation:** All optimal routes from station A to other stations in the network, jointly constitute a **sink tree**:



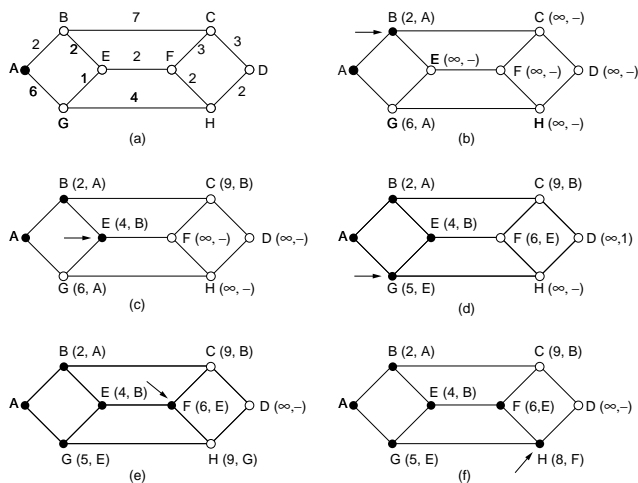
**This means:** Routers have to collaborate to build the sink tree (or something that comes near to that) for each source station.

05 – 3

Network Layer/5.2 Routing

## Shortest Path Routing

**Basic idea:** During each step, select a newly reachable node at the lowest cost, and add the edge to that node, to the tree built so far.



05 – 4

Network Layer/5.2 Routing

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

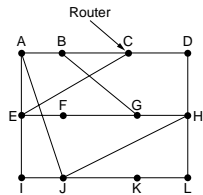
---

## Network Layer/5.2 Routing

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

---

## DVR – Example



(a)

Table (b) shows routing tables for nodes A, I, H, K, and J.

To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K

Below the table, the delay from J to its neighbors is listed:

Neighbor	Delay
JA delay is	8
JI delay is	10
JH delay is	12
JK delay is	6

These four delays are labeled as "Vectors received from J's four neighbors".

The "New routing table for J" is the last column of the main table, showing the next hop and the estimated delay from J.

(b)

**Note:** DVR is pretty expensive: it turns out to be  $O(n^3)$  which is too high for a fast convergence.

## DVR – Count-to-infinity

**Problem:** In DVR, it is possible to never find the route in the presence of node crashes:

A	B	C	D	E
•	•	•	•	•
1	•	•	•	Initially
1	2	•	•	After 1 exchange
1	2	3	•	After 2 exchanges
1	2	3	4	After 3 exchanges
1	2	3	4	After 4 exchanges

(a)

A	B	C	D	E
1	2	3	4	Initially
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	7	6	After 5 exchanges
7	8	7	8	After 6 exchanges
•	•	•	•	•

(b)

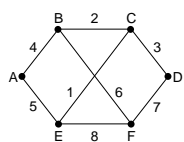
## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

[illegible]

- [illegible]

# LSP: Building Packets

**Again simple:** just put in a sequence number and aging information. The hard part is when to build them. Practice shows that once an hour is often enough.



(a)

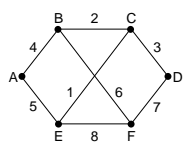
Link		State		Packets	
A		B	C	D	E
Seq.		Seq.	Seq.	Seq.	Seq.
Age		Age	Age	Age	Age
B	4	A	4	B	2
C	2	C	3	A	5
E	5	F	7	C	1
		F	6	E	8

(b)

## LSP: Distribution

**Basic idea:** use a flooding algorithm, and dam the flood through sequence numbers: all routers maintain a list of *(source, seq. number)*-pairs.

To safeguard against old data, down links, etc. an age is added to an LSP. The age is decremented once a second, and every time it is forwarded by a router. When the age hits zero, the LSP is discarded.



(a)

Link		State		Packets	
A		B	C	D	E
Seq.		Seq.	Seq.	Seq.	Seq.
Age		Age	Age	Age	Age
B	4	A	4	B	2
C	2	C	3	A	5
E	5	F	7	C	1
		F	6	E	8

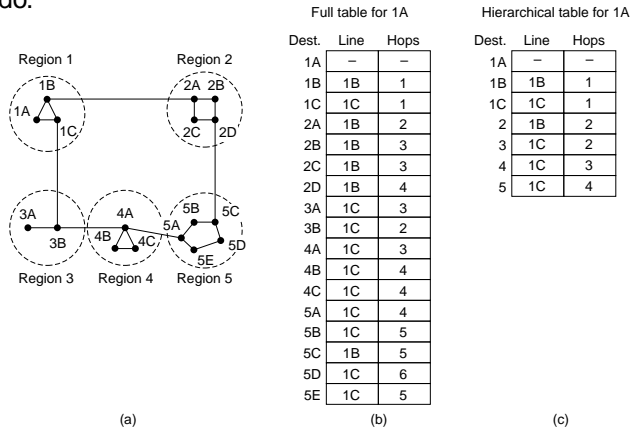
(b)

Source	Seq.	Age	Send flags			ACK flags			Data
			A	C	F	A	C	F	
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	

# Hierarchical Routing

**Problem:** No routing algorithm discussed so far can scale: all of them require each router to know about all others  $\Rightarrow$  too demanding with respect to memory capacity and processing power.

**Solution:** Go for *suboptimal* routes by introducing **regions**, and separate algorithms for **intra-region** and **inter-region** routing. Two or three levels will generally do.



05 – 13

Network Layer/5.2 Routing

## Broadcast Routing

**Problem:** We want to send a message to (almost) every host on the network. In practice, this means we're talking about (interconnected) LANs, or (relatively small) WANs.

- Just send the message to each host individually. Not really good.
- Use flooding. Acceptable provided that we can dam the flood.
- Use multidestination routing, by means of a bitmap or list that is sent along with the packet. A router checks the destinations, and splits the list when forwarding it across different output lines.
- Build a sink tree at the source and use that as your multicast route. The sink tree has to be a spanning tree (as in Dijkstra). The routers need to know the trees.

05 – 14

Network Layer/5.2 Routing

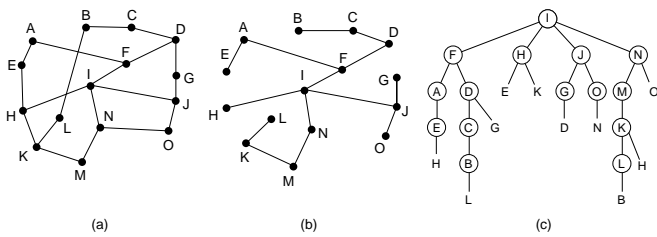


# Broadcasting: Reverse Path Forwarding

**Problem:** Suppose we don't know every spanning tree. How can we construct one at low cost?

**Solution:** Make the assumption that when a router  $R$  forwards a packet for node  $N$ , it chooses an outgoing link that lies on a minimal spanning tree rooted at  $N$ .

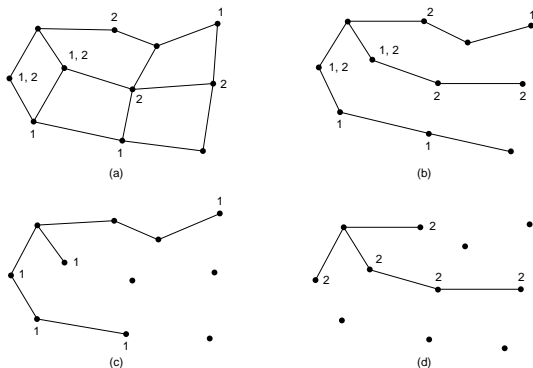
**Question:** When is this a realistic assumption?



## Multicast Routing

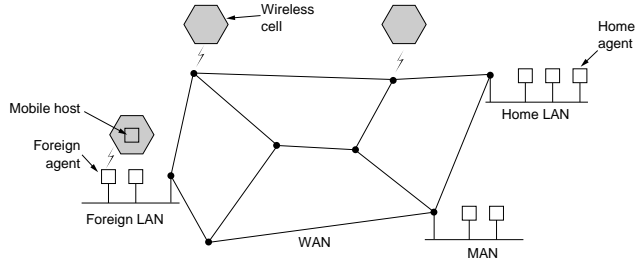
**Problem:** But suppose that we want to send a message to only a subset of all the nodes in a network. In that case, we need to know when a host enters or leaves the **multicast group**.

**Solution:** Construct a spanning tree (at each router) for the entire network. Use the group-id to prune paths to nodes that do not contain members for that group.



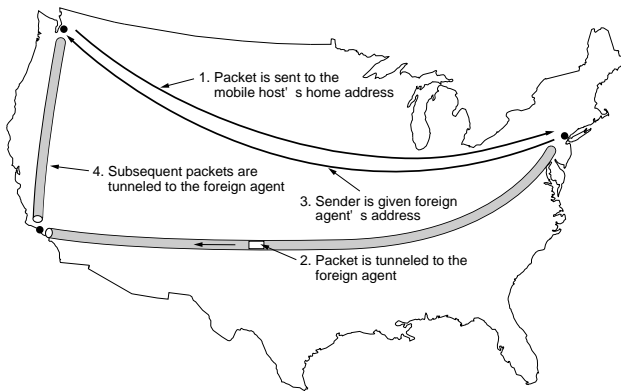
## Routing for Mobile Hosts (1/2)

**Problem:** How can we forward messages to things that are constantly on the move, preferably in a wide-area context?



**Note:** The problem is not moving hosts from LAN to LAN, but finding the mobile computers. It becomes a bit easier when we assume that there is always a **home location**.

## Routing for Mobile Hosts (2/2)



- Tunneling: sending an IP packet in an IP packet. That's the way to keep the routers ignorant of the fact that they're routing something else.
- Adapt routers: when you send the new address back to the source, intermediate routers can adapt their tables.

# Routing in Ad Hoc Networks

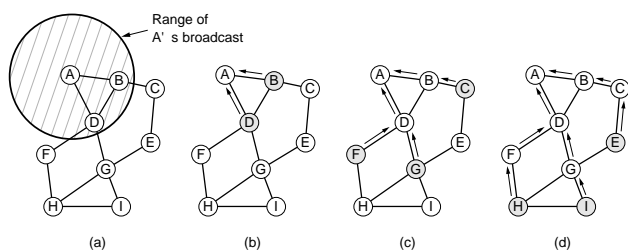
**Essence:** Not only are the hosts mobile; the routers are mobile as well. Often necessary where there is no fixed infrastructure. Example networks:

- A fleet of ships, military vehicles in a battlefield, etc.
- Spontaneous networks of PDAs, notebooks, digital phones and other mobile devices

**Problem:** we are faced with dynamically changing topology; nothing is fixed anymore.

## Ad Hoc On-demand Distance Vector – Route Discovery (1/2)

**Starting point:** Represent the network as a graph in which any two nodes are connected only if they can communicate directly with each other.



**Issue:** when *A* wants to send a message to *I*, it needs to know where to forward the message to (*B* or *D*)  $\Rightarrow$  broadcast a *route request* (which will reach only *B* and *D*) if *A* does not have a route to *I*.

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

[illegible]

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

(b)

[illegible]

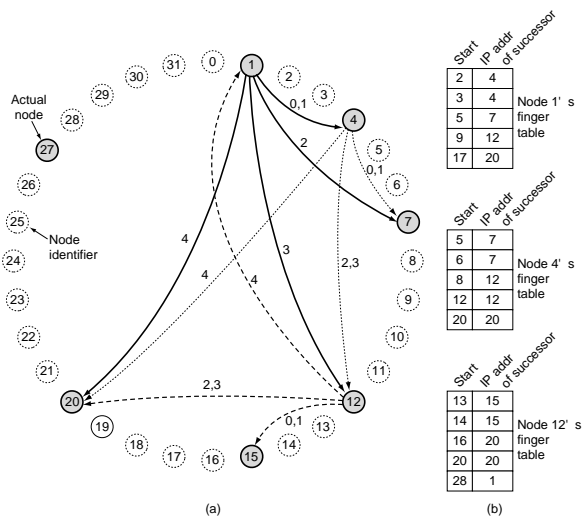
# Peer-to-Peer Networks

**Basics:** Consider a large collection of nodes, each having an IP address. Each node has a globally unique 160-bit **node identifier**. Logically, all *possible*  $2^{160}$  nodes are organized in a ring

**Important:** Nodes are assumed to store files. Each file has a unique 160-bit **key**. The node with the smallest ID larger than *key* is assumed to store the file (identified as  $\text{succ}(\text{key})$ ).

**Simple solution:** If node  $k$  is looking for file *key*, with  $k < \text{key}$ , it sends a lookup request to  $\text{succ}(k)$ . Not very efficient.

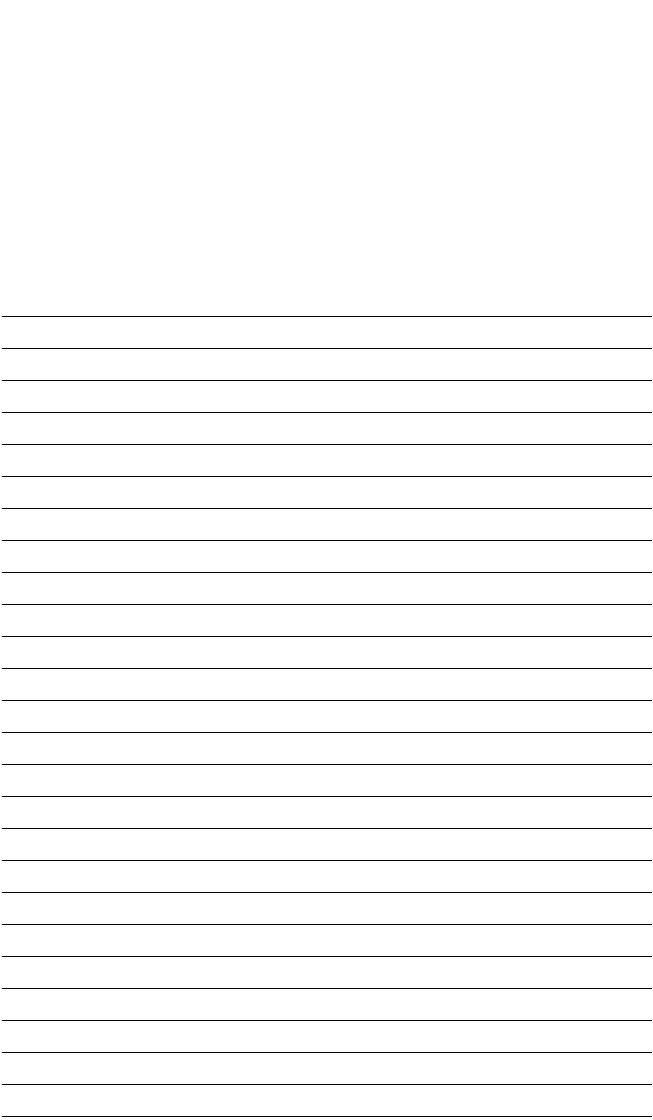
## Routing in P2P Networks (1/2)



Each node  $k$  has a **finger table**  $F_k$  with  $m$  entries:

- $F_k[i].start = k + 2^i \pmod{2^m}$
- $F_k[i].addr = \text{IP address of } \text{succ}(F_k[i].start)$

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

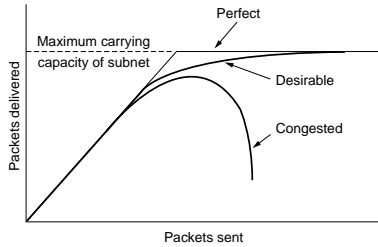


## Network Layer/5.2 Routing

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

# Congestion

**Problem:** when too many packets have to be transmitted through the network, we can get into a serious performance problem:

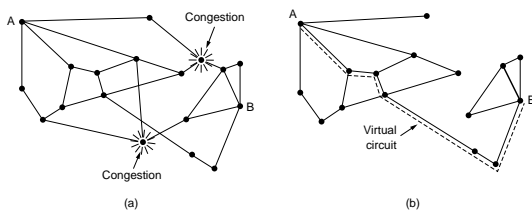


- Congestion can be caused by lack of bandwidth, but also by ill-configured or slow routers.
- Make a distinction between open loop and closed loop solutions: is feedback provided (closed) or not (open)?
- Avoid congestion by avoiding bursts: **shape** your packet traffic, and let the network provider do the **traffic policing** (monitor the flow).

## Congestion Control Virtual Circuits

**Principle:** when you set up a circuit, be sure that congestion can be avoided.

- Admission control: if it's too busy, just refuse to set up a virtual circuit. This is the same as refusing new users at an FTP site.
- Select alternative routes when a part in the network is getting overloaded (i.e., temporarily rebuild your view of the network):



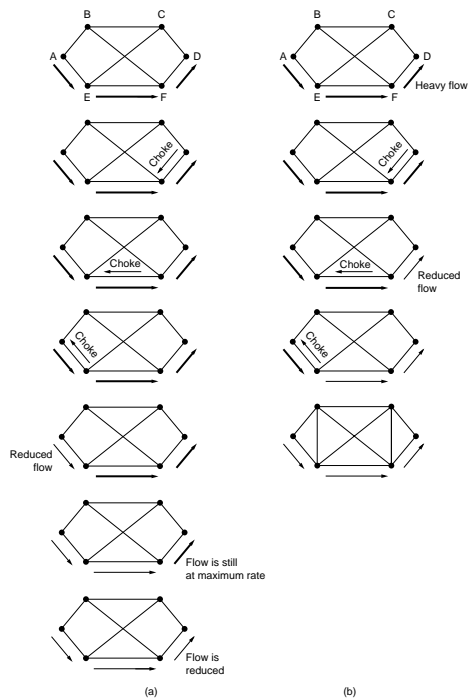
- Negotiate the quality of the circuit in advance, so that the network provider can reserve buffers and the like. Resources are guaranteed to be there.

# Choke Packets

**Basic idea:** Send info back to the source when the network starts to perform bad. A router checks the status of an output line: if it's too occupied, it sends a choke packet to the source. The host is assumed to be cooperative, and that it will slow down (not always a good assumption).

**Problem:** The return path for a choke packet may be so long, that the destination is going to get into trouble anyway – there may simply be too much on the way already. Then use a “push-back” approach:

## Choke Packets in WANs





# Quality of Service

**Definition:** A stream of packets from source to destination is called a **flow**.

**Quality of Service (QoS):** The needs of each flow are determined by *reliability*, *delay*, *jitter*, and *bandwidth*.

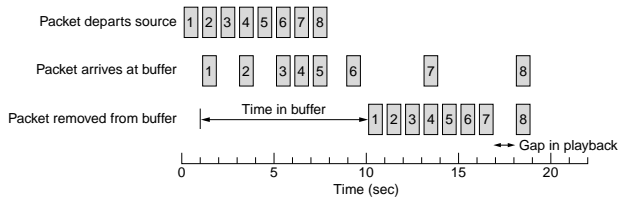
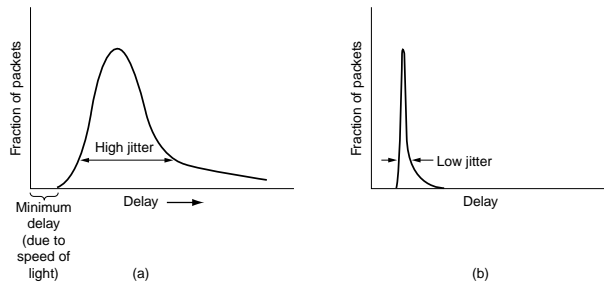
Application	Reliability	Delay	Jitter	Bandw.
E-mail	high	low	low	low
File transfer	high	low	low	medium
Web access	high	medium	low	medium
Remote login	high	medium	medium	low
Audio on demand	low	low	high	medium
Video on demand	low	low	high	high
Telephony	low	high	high	low
Videoconferencing	low	high	high	high

## Techniques for Good QoS

- Overprovisioning
- Buffering to reduce jitter
- Traffic shaping and policing
- Resource reservation
- Admission control
- Packet scheduling

# QoS: Buffering

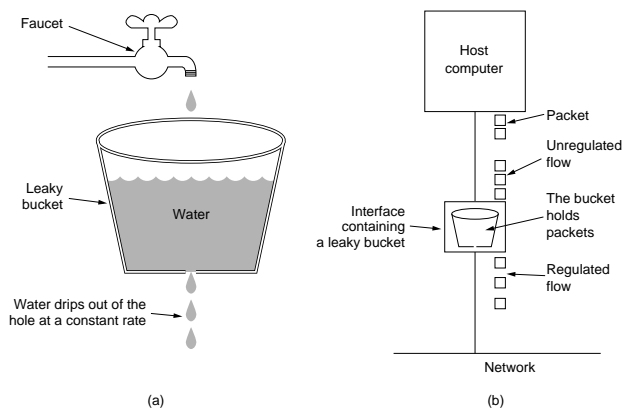
**Basics:** just try to reduce jitter as much as possible by buffering incoming packets at the receiver before passing them to the application:



05 – 33

Network Layer/5.4 Quality of Service

## QoS: Traffic Shaping – Leaky Bucket

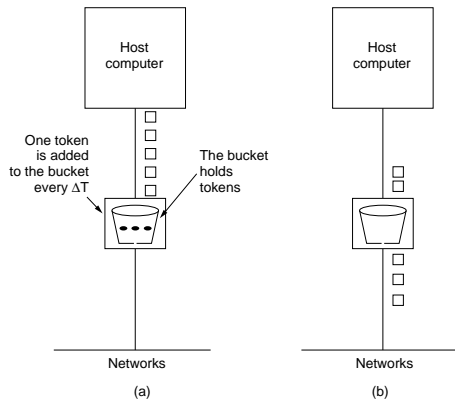


**Note:** we've eliminated bursts completely: packets are passed to the network when available, and all at the same rate. This may be a bit overdone. Also, packets can get lost.

05 – 34

Network Layer/5.4 Quality of Service

# QoS: Traffic Shaping – Token Bucket

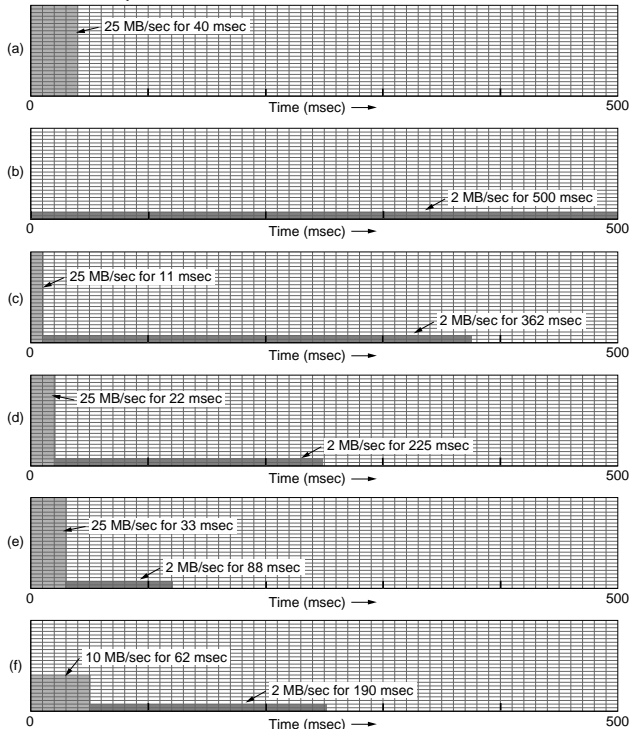


- Tokens are added at a constant rate; as soon as enough tokens have been saved, one or more packets can be sent.
- To avoid still too much burstiness, put a leaky bucket behind a token bucket (with a larger rate – why?).

05 – 35

Network Layer/5.4 Quality of Service

## QoS: Effects of Buckets

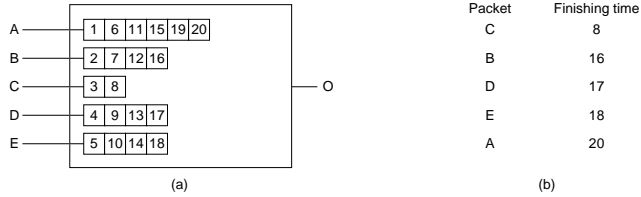


05 – 36

Network Layer/5.4 Quality of Service

# QoS: Packet Scheduling

**Problem:** What happens when you need to support multiple flows and one of them is too resource-consuming. To enforce cooperation, use **weighted fair queuing**:

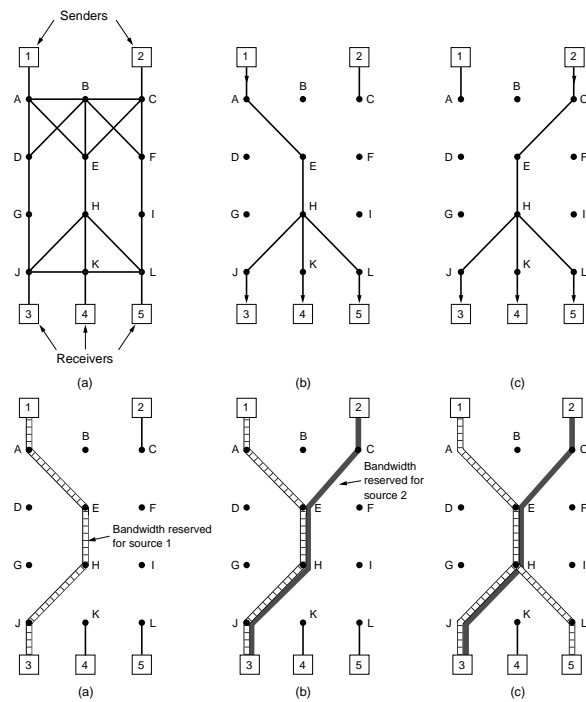


## Integrated Services

**Problem:** In order to efficiently support streams, we cannot set up a single connection per stream, but need to integrate things. This is particularly the case with multicast applications, for which we then need to assume a large and frequently changing group of receivers.

**Basic idea:** We set up multicast trees from sources to destinations, but this time take into account the bandwidth needed by the receivers. Bandwidth can be reserved on **pre-constructed** trees. If there's not enough bandwidth as required by the receiver, a failure is reported back.

# Resource reSerVation Protocol



05 – 39

Network Layer/5.4 Quality of Service

## Differentiated Services

**Problem:** Integrated services require connection setup. Instead, offer a means for local QoS decision-making  $\Rightarrow$  differentiated services (for routers in one administrative domain).

**Basic idea:** For each **class** of applications, reserve resources. In other words: differentiate (in advance) the applications that the network will have to support.

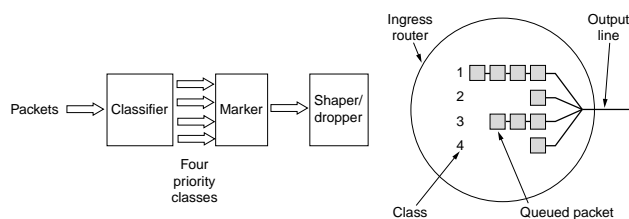
**Example:** let routers differentiate **regular** from **expedited** traffic. Packets belonging to either class will be marked as such.

05 – 40

Network Layer/5.4 Quality of Service

# Differentiated Services: Assured Forwarding

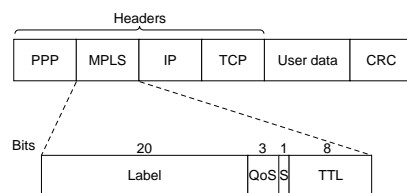
**Basics:** Distinguish four priority classes and three discard probabilities, leading to 12 combinations.



## Label Switching and MPLS

**Essence:** Instead of having standard routers provide QoS on datagram routing, let them try to establish connections.

**Technique:** Add a connection-id to datagrams and let routers take that ID as index into a table with already determined routes, identified by outgoing interfaces.



**Note:** Routes can be established on-demand, or when a router comes up (see book). Also note that the label is used *per router* to match incoming-to-outgoing interfaces. A label may be changed when a packet leaves the router.

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

[illegible][illegible]

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

- \_\_\_\_\_

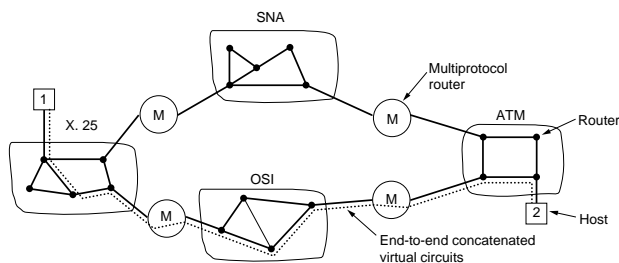
# The Differences

Issue	Differences
Service offered (*)	Connection-oriented vs connectionless
Protocols (*)	IP, IPX, CLNP, AppleTalk, etc.
Addressing	Flat (802) vs hierarchical
Multicasting	Present versus absent
Packet size (*)	Network-specific maximum
Quality of Service	None versus a lot versus who knows
Error handling	Reliable, (un)ordered delivery
Flow control	Sliding window, rate control, etc.
Congestion control	Leaky bucket, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Timeouts, flow specs, etc.
Accounting	Connect time, packets, bytes, none

**Don't worry:** it's impossible to resolve all differences. The solution is to just take a simple approach (like the Internet). We now only consider the starred (\*) issues.

## Concatenated Virtual Circuits

**Basic idea:** Assume the constituent networks support virtual circuits. In that case, the internet can use virtual circuit technology through concatenation.

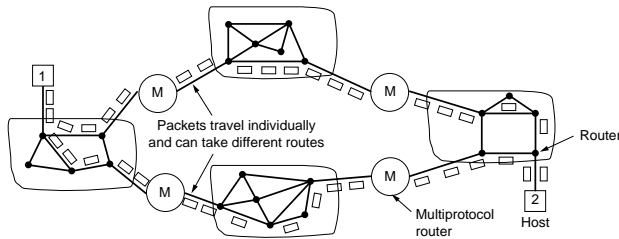


**Note:** If one of the constituent networks does not support virtual circuits, or, for example, provides only unreliable data transport, simple concatenation will be hard.



# Using Datagrams

**Basic idea:** Have the network layer offer only datagram services: unreliable, unordered packet flow. Often, connection-oriented services are not even supported (Internet).

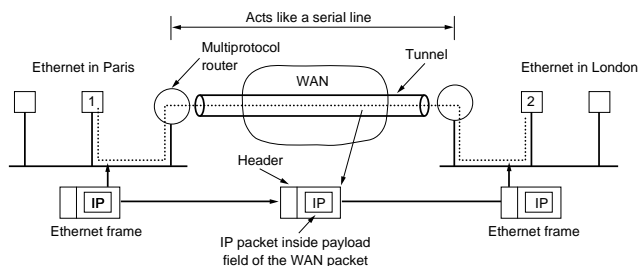


**Main problem:** Addressing – different networks use completely different addresses, so how do I address a host in an IP network when I've only got CLNP?

**Solution:** You don't solve it, but instead consider, for example, IP as a universal network protocol. Lots of universal network protocols are available

## Tunneling

**Basic idea:** We can solve a lot of the internetworking problems when we can assume that the source and destination are on the same type of network. In that case, we need only to **tunnel** packets through intermediate networks.

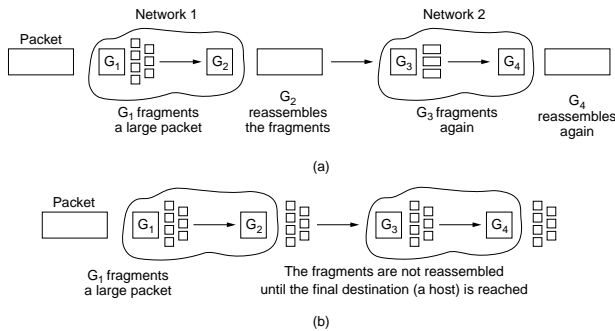


**Question:** We already came across tunneling somewhere else. Where?

# Fragmentation

**Problem:** Different networks may impose different maximum packet sizes. This means that we may have to split a packet into smaller ones when forwarding it through an intermediate network  $\Rightarrow$  **fragmentation**.

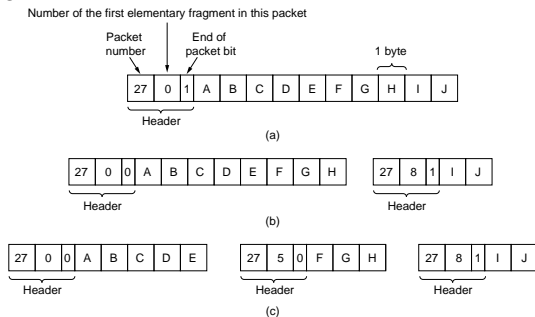
**Note:** We had the same problem when dealing with bridges, but it couldn't be solved there (why not?).



## Fragmentation – Reassembly

**Problem:** When we create fragments, how do we paste them together again:

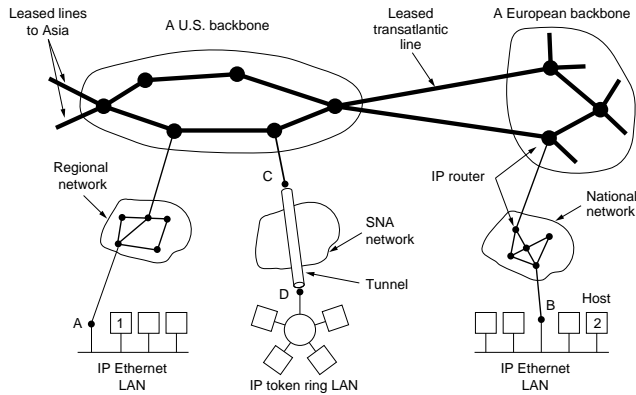
- a fragment may be fragmented again by successive intermediate networks  $\Rightarrow$  when a fragment arrives at the destination, we have to know exactly where it fits into the original packet. **Solution:**



- fragments may pass through unreliable networks, and may be lost  $\Rightarrow$  we need to decide what to do about lost fragments. **Solution:** Discard the entire packet (what do you think will happen then?).

# The Internet Protocol (IP)

**The Internet:** view it as a collection of **autonomous systems** connected together by a bunch of **backbones**:

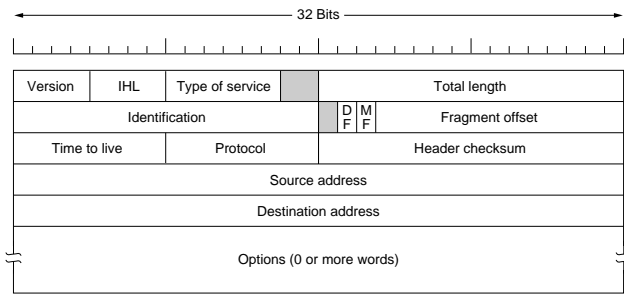


## Internet Model

Application
Transport
<b>Network</b>
Host-to-network

- An application offers a data stream to the transport layer, using either connection-oriented or connectionless services.
- The transport layer breaks up the data stream into datagrams, and passes these to the network layer.
- The datagrams are routed through the internet, occasionally fragmented when needed. Routers always pass the datagram to the underlying data link layer (generally LANs and (dial-up or leased) telephone lines).

# IP Header



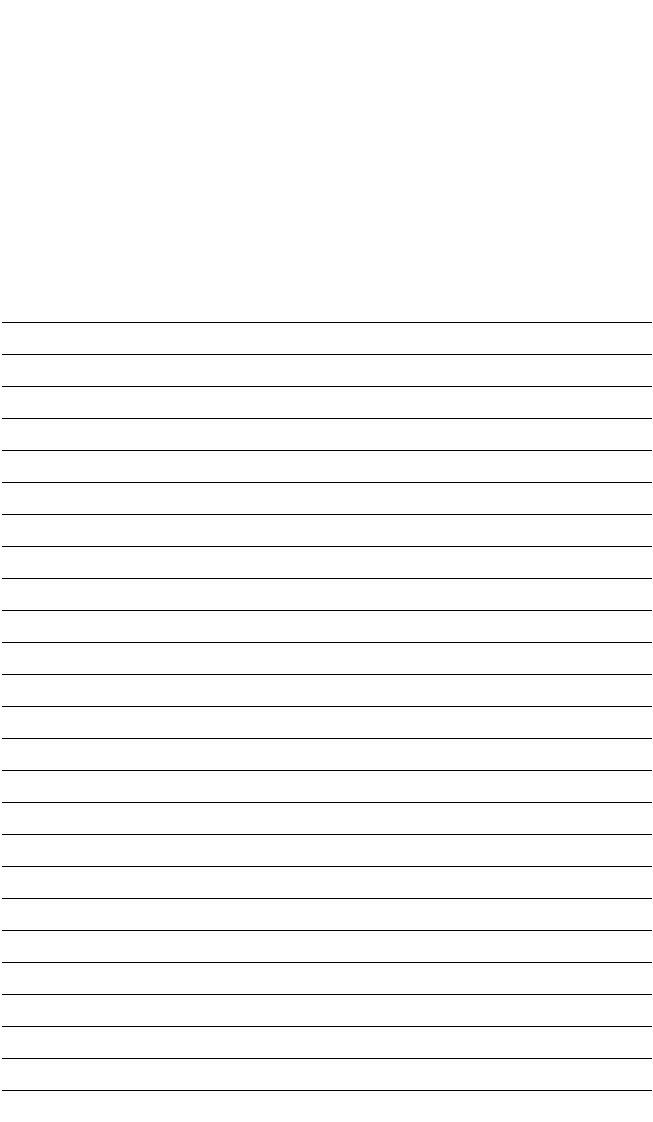
IHL	Length of the header
TOS	3-bit priority plus 3-bit flag (delay, throughput, reliability)
TLen	Length header + payload
ID	Datagram id
DF	Don't fragment this datagram
MF	There are more fragments after this one
FOff	Offset in original datagram where this fragment belongs.
TTL	Maximum number of hops allowed
Prot	Globally defined ids for transport prot.
HCS	Checks the headers (has to be calculated at each hop)

## IP Options

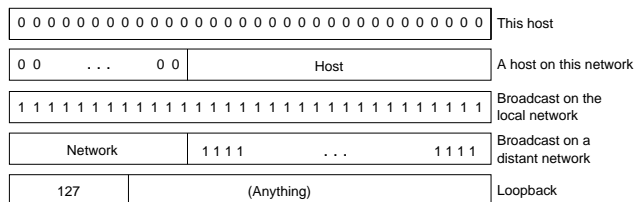
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

- Security is hardly used: specifying that a datagram is “really top secret” is not such a good idea.
- Source routing can be effectively used to force routes: debugging, politics, and mobile hosts.
- Record route and Timestamping are mainly used for debugging.

**Note:** Not all routers actually support these options (as noticed when people really wanted to use loose source routing).

[illegible]

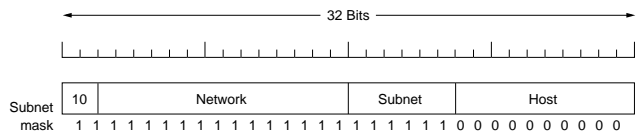
Class	Max. networks	Max. hosts/network
A	126	16,777,214
B	16,382	65,536
C	2,097,150	254



## Subnet Masking

**Problem:** All hosts on the same network must have the same network number. This may cause a single organization to acquire several classes of addresses (e.g. each time it adds a LAN), that would subsequently need to be announced worldwide (**WHY?**).

**Solution:** Use a single network address for the entire organization, and internally divide the host address space into a **subnet address** and a host id:



**Note:** We've introduced a 3-level routing hierachy.

# Classless Interdomain Routing

**Problem:** Although there are “enough” IP addresses, the use of classes is rapidly making them a scarce resource. The main problem is that too many class B addresses are being used. At the same time, we can’t simply assign class C addresses without adequate routing table management (**HUH!?**).

**Solution:** Assign class C addresses in contiguous blocks of 256 addresses. Also, partition the address space into four zones:

194.0.0.0	–	195.255.255.255	Europe
198.0.0.0	–	199.255.255.255	North America
200.0.0.0	–	201.255.255.255	Central/South America
202.0.0.0	–	203.255.255.255	Asia and the Pacific

A series of contiguous blocks is assigned, together with a **mask**...

## CIDR – Example

Site	# Addr.	Range	Notation
S1	2048	194.24.0.0 – .7.255	194.24.0.0/21
S2	1024	194.24.8.0 – .11.255	194.24.8.0/22
–	1024	194.24.12.0 – .15.255	194.24.12.0/22
S2	4096	194.24.16.0 – .31.255	194.24.16.0/20

Update all routers (in Europe) with three entries, each consisting of a (base address, mask)-pair. When an IP datagram arrives at a router, the latter tries to find the *base address* by using the masks:

$(194.24.17.4 \& 255.255.248.0) \neq 194.24.0.0 \quad \checkmark$   
 $(194.24.17.4 \& 255.255.240.0) = 194.24.16.0 \quad OK$   
 $(194.24.17.4 \& 255.255.252.0) \neq 194.24.8.0 \quad \checkmark$

**Note:** the “/21” notation indicates that the mask consists of 21 1-bits, followed by 0-bits only.

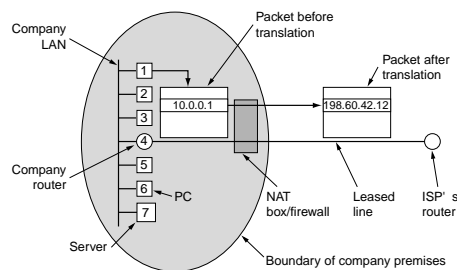
# CIDR – Aggregate Entries

**Note:** In principle, all routers worldwide need to store the masks to do the appropriate routing.

**Observation:** For many routers outside 194.0.0.0, the only thing they see is that there are (at least) 3 network addresses for which packets follow the same route. These entries can be **aggregated** into 194.24.0.0/19 with a single submask of 19/13 1/0 bits.

## Network Address Translation

**Essence:** Reserve a couple of IP address ranges for local networks that operate behind a special router (which can also operate as a firewall), and allow only **outgoing** connections.



**Example:** When a connection is set up from address 10.0.0.1, port X, the router sends it off using source address 198.60.42.12 (it's ISP-supplied address) on port Y, and registers the mapping  $X \leftrightarrow Y$ . When a reply comes in for port Y, it is sent back to 10.0.0.1 on port X.

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

[illegible]

- 
- This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

[illegible]

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.



## Address Resolution Protocol (2/2)

1. *Router*: Ask each host on the LAN whether they have the requested IP address. This is done by encapsulating the query as an ARP message in a datalink frame, and broadcasting it.
2. *Host*: Recognizes it is dealing with an ARP message, checks whether it has the requested address, and if so, sends a reply back with its datalink address. **Question**: how can the host recognize an ARP message?
3. *Router*: Recognizes a reply ARP message, and (generally) caches the IP address with the datalink address. It can then forward IP datagrams to the correct host, encapsulating them in datalink frames.

**Question**: what should the router do when no one replies?

05 – 63

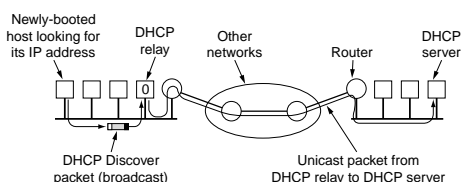
Network Layer/5.6 The Internet Protocol

## Reverse Address Resolution Protocol

**Problem**: So how does a host know its own IP address?

**Solution**: The host uses a limited broadcast (i.e. restricted to its own network) to query a RARP server for its IP address. The RARP server maintains a table of (datalink address, IP address) mappings.

**Observation**: RARP has largely been replaced by **Dynamic Host Configuration Protocol (DHCP)**. It does the same, but a bit more like support for bootstrapping a host.



05 – 64

Network Layer/5.6 The Internet Protocol

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

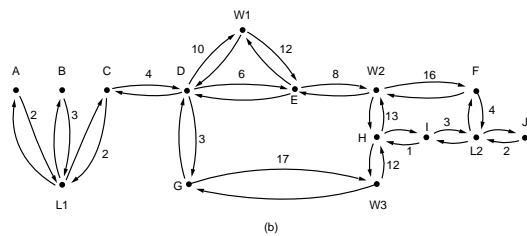
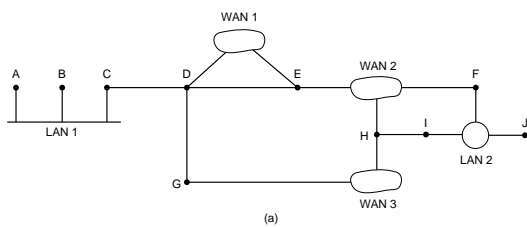
- 
- This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

## This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

\_\_\_\_\_

- Openness: the algorithm should be made publicly available so that anyone could implement it.
- Support for different distance metrics (hops, delays, costs, etc.)
- Dynamic and efficient adaptability to changing topologies.
- Support routing based on type of service (already specified in IP, but no one actually used it). Especially important for real-time (multimedia) traffic.
- Support for load balancing: when a route is heavily used, another one should be selected.
- Support hierarchical routing.
- Offer security.
- Support IP tunneling.

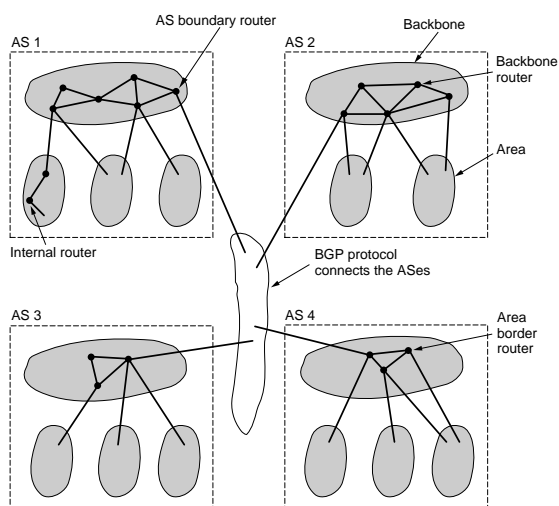
# OSPF – Routing Graphs



**Note:** OSPF build different graphs, depending on the distance metrics it uses (delay, throughput, reliability). Of course, physical links are always taken into account.

**Note:** Each LAN is represented by a **designated router** that acts as a representative of the other routers on that LAN.

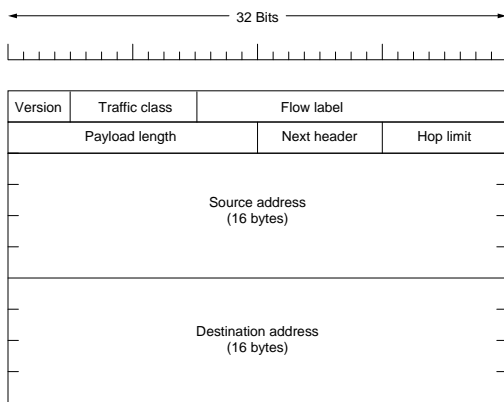
## OSPF – Hierarchical Routing



**Note:** We can use the same algorithm for the areas and the backbone. In fact, the backbone behaves as just another area. This means it should be contiguous.



## IPv6 – Header



**Note:** The flow label is used to set up a pseudoconnection between source and destination. It identifies a flow for which, for example, bandwidth has been reserved.

**Note:** A simpler header is almost impossible – further info is provided by *next headers*.

**Note:** No checksum, and no fragmentation fields.

## IPv6 – Address Space

**Big difference:** IPv6 uses 16-byte addresses. This is really a lot:  $7 \times 10^{23}$  addresses per square meter. It does allow us to be less efficient with address allocation: 72% is unassigned.

Prefix	Usage	Fraction
0000 0000	Reserved (incl. IPv4)	1/256
0000 0001	Unassigned	1/256
0000 001	OSI NSAP addresses	1/128
0000 010	Novell Network IPX addresses	1/128
0000 011	Unassigned	1/128
0000 1	Unassigned	1/32
0001	Unassigned	1/16
001	Unassigned	1/8
010	Provider-based addresses	1/8
011	Unassigned	1/8
100	Geographic-based addresses	1/8
101	Unassigned	1/8
110	Unassigned	1/8
1110	Unassigned	1/16
1111 0	Unassigned	1/32
1111 10	Unassigned	1/64
1111 110	Unassigned	1/128
1111 1110 0	Unassigned	1/512
1111 1110 10	Link local use addresses	1/1024
1111 1110 11	Site local use addresses	1/1024
1111 1111	Multicast	1/256

## IPv6 – Extension Headers

**Basic idea:** Keep the main header as simple as possible, and provide any further information in an (optional) extension header:

Ext. header	Description
Hop-by-hop options	Information for routers
Routing	Full or partial route to follow
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted payload	Info on the encrypted contents
Destination options	Additional info for destination

Most of them are quite obvious – see Tanenbaum for further details.

**Important:** Note that fragmentation is still supported, but that only the source host can do it. Routers never fragment datagrams anymore.

## IPv6 – Security

**Illustrative example:** There was a lot of discussion on where and how to incorporate security in IPv6:

- If you are really concerned about security, would you trust anything else but end-to-end encryption? **Question:** what does this mean!?
- Having security in the network layer offers a generally useful service to many applications. Those that don't want to use it, just ignore it.
- Network-layer protocols have to run in every country. Some countries disallow cryptosystems that the government can't decrypt easily.
- Are the default crypto-algorithms good enough? For example, MD5 has recently been cracked.

The main issue here, as with almost every protocol, is to decide in which layer we should put functionality. There are many people who argue that only end-to-end solutions should be applied. The rest (i.e. general solutions) will never be good enough.

This image shows a single sheet of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page. There are approximately 20 lines visible. The paper has a slight shadow on the right side, suggesting it's resting on a surface. There is no handwriting or other markings on the paper.