Chapter 5 - The Network Layer

Objective

Move packets from source machine to destination machine.

Network topology considerations

For a fully connected local area network (LAN), this is done by datalink layer.
Thus the ***Network Layer*** is important only in a ***Store and Forward*** network.

Functions typically performed by network layer

Routing
Congestion management
Billing (possibly)

Service Classes vs Internal Organization

Service class: A description of the service that the layer provides
Internal organization: How it provides it

Independent in theory
For reasonable implementations they are in practice related.

Service Classes provided to Transport layer

Connection oriented

Reliable (In order delivery with no loss or duplication)

Unreliable (In order delivery with possible loss)

Connection less

Unreliable

Summary of differences:

Issue	***Reliable C-O***	***Unreliable C-O***	***Unreliable C-L***
Setup	Required	Required	Not possible
Addressing	Circuit number	Circuit number	Full dest address
Sequencing	Guaranteed	Loss possible	Reordering possible
Error control	No errors allowed	Loss possible	Loss, reorder, dup
Flow control	In network layer	In network layer	Not provided
Option negotiation	Yes	Yes	No

Internal Organization

Connection oriented or (*virtual circuit*): All packets follow the same route (the telephone system model).

Examples

Reliable - X.25 networks, SNA
Unreliable - ATM, ISDN

Connection less or (*datagram*): Each packet is a separately routable entity (the mail system model).

Examples
Unreliable - IP

Reliable service implies *in order* delivery... thus reliable-connection less service is *Really* hard to provide and thus not generally available.

Summary of implementation differences:

Issue	***Connection oriented***	***Connection Less***
Circuit setup	Required	Not possible
Addressing	Circuit number	Full source/dest
State info	Circuit state info kept	No state information
Routing	Done at setup	Per packet routing
Node failure effects	Terminate all circuits	Possible lost packets
Congestion control	Reserve resources	Packet dropping
Complexity	In network layer	Moved to transport layer
Class suitability	Best for conn oriented	Best for conn less

So which is "best"

Arguments in favor of unreliable datagram organization (IP):

Subnet *can't be trusted* to be reliable why try.
Since routing is packet by packet *load balancing is much easier*.
Some apps don't require perfectly reliable service so why deliver it.

Arguments in favor of reliable virtual circuit organization (SNA / X.25):

If the underlying datalink / physical layer is unreliable, then retransmission at each hop minimizes total expected end-to-end tries per successful transmit.

Arguments in favor of unreliable virtual circuit organization (BISDN / ATM):

If your app requires in order delivery it is much more efficient to keep packets in order in the network layer than to resequence in the transport layer.
Header information can be smaller since full addresses aren't needed.
Its much easier to provide guaranteed service rates.
Providing reliability *hurts* multimedia traffic

Routing algorithms:

Objectives:

Correctness	Packet gets to intended destination.
Simplicity	Router doesn't require excessive network or computational overhead
Robustness	Minimize degradation of network when components fail.
Stability	Avoid unstable adaptive procedures.
Optimality	Maximize throughput or minimize delay
Fairness	But don't starve anybody

Classes:

Non-adaptive:	Don't respond to changes in topology or traffic.
Adaptive:	Base routing on current estimate of traffic and topology.
	Centralized
	Distributed
	Isolated
Hybrid:	Some combination of above approaches

Non-adaptive routing:

Static or table driven routing:

Network administrator creates a *unique* table for *each* node of the form:

Dest	Via
A	C
B	D
C	C
D	D
E	D

Each node in the network appears in the Dest column of each table.
Only directly connected nodes appear in the Via column.

Tables of this form are used in virtually all routing.. The method of construction is what differs.

This static construction method is used in SNA.

Evaluation:

Good for small networks with low traffic variability.

Flooding:

Send each incoming packet out on:

All links or

All links except the one it came in on.

Requires a damping mechanism

e.g. discard packet when hop count reaches *n*

Damping can be avoided if

Packets are not returned on incoming links *and Physical Topology* of the net is a tree.

Packets are sequence numbered. Each station remembers the highest sequence number seen from each source. Duplicates are not forwarded.

Evaluation:

Easy to implement

Always picks the optimal route.

Can cause extreme congestion.

Suitable for use with small network control packets

IP uses flooding for routing info interchange

SNA uses flooding for notification of operative/inoperative routes.

Variations:

Random walk;

Partial flooding

Random walk

Choose outgoing link randomly

Evaluation:

Easy to implement but not robust or efficient

Isolated Adaptive Algorithms:

Hot potato:

Enqueue packet on shortest link station output queue.

Variations:

Static routing unless queue length exceeds "N"

etc.

Evaluation:

Easy and efficient to implement but not very robust.

Might be used with *backward learning* until routes have been learned.

Packet may never reach destination.

Backward learning (can also be called distributed):

Each packet carries source id and hop count.

Hop count is incremented each transmission.

Suppose routing table for node M looks like:

Dest	Via	Hops
J	C	7
K	D	5

If a packet arrives *from J via E with 4 hops* table is modified as:

Dest	Via	Hops
J	E	4
K	D	5

because the fact that the packet the packet reached M from J in 4 hops also means it is possible to go back to J in four hops. Since this method only learns for the better, it is necessary *to purge tables* from time to time.

Duplexed tables can solve the stale data problem

On the hour
- purge table B
- begin routing via table A
- update tables A and B based on incoming traffic.

On the half hour
- purge table A
- begin routing via table B
- update tables A and B based on incoming traffic.

The active table has:
- At least 1/2 hour of learned routes
- But no more than 1 hour of *bad* routes

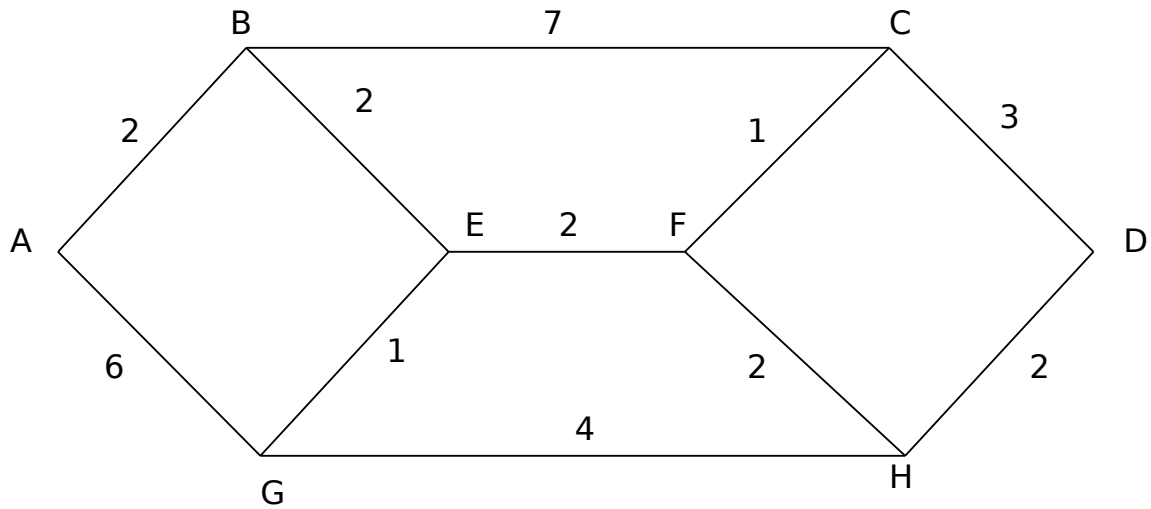
Evaluation:

Most suitable for hops as a cost metric.
- Forward time delays != reverse delays
- However, it is possible to accumulate reverse path delays on forward passing packets.

Requires good balance of traffic to converge rapidly to optimal routes.

Distributed adaptive algorithms:

Distributed Shortest Path First (SPF) Routing: An example of *Link State Routing*



Step 1: Construct cost table for all nodes. Each station sends delays to each of its connected neighbors. Messages are sent via flooding in IP. Each message fills in one row of the table.

	A	B	C	D	E	F	G	H
A	0	2					6	
B	2	0	7		2			
C		7	0	3		1		
D			3	0				2
E		2			0	2	1	
F			1		2	0		2
G	6				1		0	4
H				2		2	4	0

Step 2: Initialize the routing table:

Initial routing table for source A:

Dest	Via	Cost	Status
A	A	0	Perm
B	?	∞	N/A
C	?	∞	N/A
D	?	∞	N/A
E	?	∞	N/A
F	?	∞	N/A
G	?	∞	N/A
H	?	∞	N/A

Step 3: Make N passes over the routing table adding a single node for each pass. In each pass, costs to node reachable from the *most recently added permanent node* are updated.

After Pass 1

Dest	Via	Cost	Status
B	B	2	Perm*
C	?	∞	N/A
D	?	∞	N/A
E	?	∞	N/A
F	?	∞	N/A
G	G	6	Temp
H	?	∞	N/A

After Pass 2

Dest	Via	Cost	Status
B	B	2	Perm
C	BC	9	Temp
D	?	∞	N/A
E	BE	4	Perm*
F	?	∞	N/A
G	G	6	Temp
H	?	∞	N/A

After Pass 3

Dest	Via	Cost	Status
B	B	2	Perm
C	BC	9	Temp
D	?	∞	N/A
E	BE	4	Perm
F	BEF	6	Temp
G	BEG	5	Perm*
H	?	∞	N/A

After Pass 4

Dest	Via	Cost	Status
B	B	2	Perm
C	BC	9	Temp
D	?	∞	N/A
E	BE	4	Perm
F	BEF	6	Perm*
G	BEG	5	Perm
H	BEGH	9	Temp

After Pass 5

Dest	Via	Cost	Status
B	B	2	Perm
C	BEFC	7	Perm*
D	?	∞	N/A
E	BE	4	Perm
F	BEF	6	Perm
G	BEG	5	Perm
H	BEFH	8	Temp

After Pass 6

Dest	Via	Cost	Status
B	B	2	Perm
C	BEFC	7	Perm
D	BEFCD	10	Temp
E	BE	4	Perm
F	BEF	6	Perm
G	BEG	5	Perm
H	BEFH	8	Perm*

After Pass 7

Dest	Via	Cost	Status
B	B	2	Perm
C	BEFC	7	Perm
D	BEFCD	10	Perm*
E	BE	4	Perm
F	BEF	6	Perm
G	BEG	5	Perm
H	BEFH	8	Perm

Analysis of Distributed SPF routing:

N = Number of stations.
M = Average number of links

Complexity of computation performed by each node $O(N^2)$.
Number of messages interchanged $O(N^2)$
Length of each message $O(M)$
Amount of data interchanged $O(N^2M)$

Evaluation:

Has proven to work well in very large networks with highly variable traffic.

Bellman-Ford Routing - an example of *Distance Vector Routing*

Each node maintains

an exact delay to each of its *connected neighbors*

an estimated delay to *each node in the network* (init 0).

Each routing interval each node sends its estimated delay for entire net to each adjacent node.

To	***A***	***I***	***H***	***K***	***Delay***	***Via***
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	12	10	I
J	9	11	7	10	0	-
K	24	22	22	0	6	K
L	29	33	9	9	15	K

Here we wish to compute the routing table for node J which is directly connected only to A, I, H, and K with the specified outgoing cost metric. The columns labeled A, I, H, and K are the routing updates received from each directly connected neighbors

J - A = 8 J-I = 10 J-H = 12 J-K =6

When all tables have been received:

```
For each destination node
{
        For each estimated delay received from a connected node
        {
                Add my delay to the connected node
                If less than the best delay so far
                {
                        Enter this total delay
                        Via the sender of the table
                }
        }
}
```

Analysis of Bellman Ford

N = Number of stations.
M = Average number of links

Complexity of computation performed by each node O(NM)·
Number of messages interchanged O(NM)
Length of each message O(N)

Amount of data interchanged ($O(N^2M)$)
This is actually much better than SPF since each routing packet only goes 1 hop.

Evaluation

Can be shown to converge to optimal routes with static loads.
Had stability problems when originally used in IP.
Suffers from "counting to infinity" when loss of a router or link occurs.

The Counting to Infinity Problem

Suppose the A-B link becomes active after B-C, C-D, and D-E

```
     (Everyone's view of distance to A)
                                        Number of
                                        exchanges
A-------B-------C-------D-------E
       16      16       16     16        0
        1-A    16       16     16        1
        1-A     2-B     16     16        2
        1-A     2-B      3-C   16        3
        1-A     2-B      3-C    4-D      4
```

Suppose Link B-A Fails (No split horizon)

At exchange 5, B doesn't hear from A and thus uses the update from C
The count to infinity is demonstrated below
Each column indicates the column head nodes route to *A*
Values in the column are in the form (*cost, via)*

```
A-------B-------C-------D-------E
       3-C     2-B      3-C    4-D      5
       3-C     4-B/D    3-C    4-D      6
       5-C     4-B/D    5-C/E  4-D      7
       5-C     6-B/D    5-C/E  6-D      8
       7-C     6-B/D    7-C/E  6-D      9
       7-C     8-B/D    7-C/E  8-D     10
```

Split horizon with poisoned reverse has been proposed as the solution. If node C's route to A uses B as a next hop, then node C will advertise to node B a cost of 16 to node A.
(Node B no longer receives an optimistic update from C... because C's route passes through B)

```
A-------B-------C-------D-------E
       16     2-B      3-C     4-D      5
       16      16      3-C     4-D      6
       16      16       16     4-D      7
       16      16       16     16       8
```

When three nodes are involved in a deceptive cycle it may still be necessary to count to infinity.

```
            F
          /   \                        Number of
         /     \                       exchanges
A-------B-------C-------D-------E
```

B	F	C	D	E	
16	16	16	16	16	0
1-A	16	16	16	16	1
1-A	2-B	2-B	16	16	2
1-A	2-B	2-B	3-C	16	3
1-A	2-B	2-B	3-C	4-D	4

Suppose Link B-A Fails without split horizon / poisoned reverse

B	F	C	D	E	
3-C/F	2-B	2-B	3-C	4-D	5
3-C/F	3-C	3-F	3-C	4-D	6
4-C/F	4-B	4-B	5-C	4-D	7

(etc... like 5 station case)

With split horizon and poisoned reverse:

Don't advertise a route to the station that you learned it from!

With split horizon..and poisoned reverse a really pathological count to infinity ensues (Thanks to X. Gong for pointing this one out!) -

B	C	F	D	E
16	3-F	3-C	3-C	4-D
4-C	16	16	4-C	4-D
16	16	5-B	16	5-D
16	6-F	16	16	16
7-C	16	16	7-C	16
16	16	8-B	16	8-D
16	9-F	16	16	16
10-C	16	16	10-C	16
16	16	11-B	16	11-D
16	12-F	16	16	16
13-C	16	16	13-C	16
16	16	14-B	16	14-D
16	15-F	16	16	16
16	16	16	16	16

Performance issues

The performance of a routing algorithm *depends strongly upon the details of the implementation.*

Problems with the original use of Bellman Ford *(in the pre-Internet days of the ARPANet)* **-**

Instantaneous queue lengths were used as a delay measure.
Messages were exchanged every 128 msec.
Long, high priority packets adversely affected flow.
Route computations based on local information made it difficult to ensure consistency.
Too slow in adapting to major (distant) topology/congestion changes.
Too quick to adapt to minor (nearby) ones.
Long term loops were often observed.

Defects of Instantaneous Queue length based delay measure.

- Heterogeneous speeds or propagation delays not handled properly.
- Queue length not an accurate measure of total delay.
- *Instantaneous* queue length not a good predictor of *average* delay.

Changes made in the *(new ==1980)* SPF algorithm.

Each node measures average delay experienced by outgoing packets in the past 10 seconds.

Delay = (sent time - arrival time) + propagation delay + transmission time

Significant change in delay causes a routing update to be generated.

Change threshold is initially 64 ms.

Reduced by 12.8 ms each 10 sec.

Becomes 0 after a minute.

Updates are routed via flooding.

Incoming line is included.

Return is used as an ACK.

Flooding is controlled via discarding of packets seen twice.

Lines restarting are placed in a "wait" state for 1 minute.

Routing updates but no data sent on waiting lines.

Node also can't come up for a minute because its lines cant come up.

Ensures new nodes have correct network database before joining net.

Hierarchical routing

Designed to minimize table space for very large networks:
Nodes are clustered into regions.
Each routing table contains entries for all elements of routers own region but a single address for every other region.
Optimal number of levels in ln(N).
Each table has e ln(N) entries.

Routing table for node 1A from example in the book

Linear		Hierarchical	
Dest	Via	Dest	Via
1A	---	1A	
1B	1B	1B	1B
1C	1C	1C	1C
2A	1B	2	1B
2B	1B	3	1C
3A	1B		
3B	1C		
:			

Prefix based routing in the core coupled with subnet based routing in a stub AS is an example of hierarchical routing in the Internet.

Today's internet routing

Autonomous Systems (the nets comprising the Internet)

Examples

Clemson

An MCI backbone

An AS **can** contain multiple IP networks

Routing within an Autonomous System

routed - implements a Bellman Ford (Vector distance type algorithm)

gated - implements an SPF (Link State type algorithm)

Routing between Autonomous Systems

egp - implements a Bellman Ford (Vector distance type algorithm)

bgp - implements a Bellman Ford (Vector distance type algorithm) but transmits complete reachability paths to defeat counting to infinity.

Autonomous Systems are classed as

Transit (backbone)

End-user (actually) -

Stub or

Mult-homed

These make major use of *default* routes

Each AS advertises prefixes that it owns or knows about to its BGP peers.

Each advertisement consists of on or more prefixes followed by the complete AS path through which the route passes.

Broadcast routing -

Objective - Send a packet to every host in a network.

Approaches -

Send directly to each other node (must know address of each node).

Flooding (easiest and most robust... but may cause bad congestion)

Multi-destination routing (must know address of each node)

Place distribution list in the header.
Each imp makes a copy of the packet for each line which is the best route to at least one of the destinations.
Distribution list in the header is reconstructed to include only those destinations reached on that line.
If the routing tables actually duplicate the SPF tree then this method is optimal

Spanning tree routing (must know spanning tree)

Use actual spanning tree from source to all destination.

Reverse path forwarding (don't need to know anything!)

If a broadcast packet arrives on the best link back to the original source, forward it on all other links.

Otherwise discard it.

The number of unnecessary transmissions = 2 x the number of links not in the optimal spanning tree from the source.

Multicast routing

Some similarities to broadcasting but it is more difficult

Requires a mechanism for
joining and departing from multicast groups
group = set of systems interested in receiving a particular multicast

Also requires a mechanism for routing for eliminating routers not interested in the multicast from the spanning tree..

Each router must maintain for each link and multicast the number of interested downstream listeners.

When the number == 0, the router stops forwarding the multicast over that link.

To join or leave a packet can be sent to the source.

At present multicasting is not widely used in the internet (though it would be a natural for RealAudio/Video broadcasts.)

Congestion Management

Congestion

Excessive queue lengths at routing nodes

Result in:

Long end to end delays
Lost packets, or, in the extreme, deadlock

The *only* cure for *persistent* congestion is more bandwidth!

Preallocation of resources / Admission control (as in telephone system).

Can be used in connection oriented implementations to ensure that data is allowed to enter the network no faster that than it can be delivered.

This is the fundamental principle underlying ATM QoS guarantees

At call setup time resource requirements propagate through all switches on the selected path.

If the requested resource level is not available the call is refused or re-routed

Packet Discarding

Can be used in unreliable connection oriented or datagram networks (ATM and IP)
When buffers run low discard packets.

Disadvantages:

Applicable only to unreliable implementations.
May not work well since packet source will just time out and retransmit.

Optimizations.

Try to discard packets that haven't traveled far to minimize loss of work. (But retransmitted packets will get back faster!)

Constrained allocation of buffers to output links was once a "hot" research topic when memory was expensive.

k packet buffers
s links
Unconstrained: max buffers per link = *k*.
Dedicated: max buffers per link = *k / s*.
Constrained (Irland): max buffers per link = *k / sqrt(s)*
Constrained (Kamoun): min buffers / link = *s*

Now memory is cheap, buffers are plentiful, so the question is what should queue length be limited to for "optimal" performance

Cisco IP routers historically used constrained queue lengths (~40 packets) / link.

Usage of packet discarding):
Internet, ATM

Probabalistic drop algorithms (RED Random Early Detection

When queue length is short *P[drop arriving packet] = 0*
As queue becomes longer *P[drop arriving packet]* increases
At some point *P[drop arriving packet] = 1*
These algorithms are designed specifically to interact with TCP

Fair queuing

Designed to ensure that heavy traffic sources don't unfairly impact light traffic sources.
Maintain a queue for each connections
Service queues in round robin order
Deficit round robin allows flows sending small packets to send more than one packet per turn.
Requires a stateful router

Fluid based schedulers
Model (weighted) 1 bit-at-a-time round robin
The next packet scheduled in the real system is the one that would finish first in the fluid system.
Ensure max-min fair allocation of capacity

Maxmin fair load assignment

For a single channel with multiple flows whose demands are specified in increasing order, maxmin fair allocation is a straightforward *O(n)* algorithm. Capacity is allocated to a single flow at each iteration of the allocation loop.

If the flow's demand is less than or equal to its fair share of the remaining capacity, it receives its demand. Otherwise it receives its fair share. At each iteration, the fair share is the remaining capacity divided by the remaining number of flows and after allocation the remaining capacity is reduced by the amount allocated.

```
/* If demands are ordered maxmin can be O(N) */

   left = CAPACITY;
   for (i = 0; i < COUNT; i++)
   {
      share = left / (COUNT - i);
      if (demands[i] < share)
         allocs[i] = demands[i];
      else
         allocs[i] = share;

      left = left - allocs[i];
      sum += allocs[i];

                  }
```

Traffic shaping

Consider a single server queuing system

λ = arrival rate
μ = service rate
$\rho = \lambda / \mu$ = utilization

If the arrival and service distributions are exponential then

$\tau = \lambda / (\mu - \lambda)$ is the average queing delay
$\lambda \rightarrow \mu$ then expected queuing delay $\rightarrow \infty$

but if both distributions are deterministic (constant) then as
$\lambda \rightarrow \mu$ then expected queuing delay = 0

If packets are fixed size then μ is constant. The objective of traffic shaping is to make λ as close to constant as possible.

In an interval of time *T* congestion builds if the arrival rate *over that interval* is greater than the service rate. Congestion is reduced if the arrival rate over that interval is less than the service rate.

The variance of the number of arrivals per unit time constrains the degree to which the number of arrivals in a given unit time may exceed the mean.

Example

Suppose service rate is 100 packets / second and the mean arrival rate is 80 packets per second
If the queue is initially empty and 200 packets arrive in a given second then the queue length will be at least 100 at the end of that second.
However, if the variance of the arrival process is 9, the standard deviation will be 3 and the probability of ever seeing more than 100 arrivals in a second will be virtually nil.

Correlated arrivals

If the number of arrivals per unit time is *i.i.d.* (independent and identically distributed), then the number of arrivals at time *t+1* is statistically independent of the number at time *t.* Because of the stochastic nature of the the arrival process, it is therefore unlikely the observed number of arrivals will exceed the mean throughout a long interval *[t, t + n].* However, if the arrivals per unit time are positively correlated, this is not the case.

In the exponential distribution, the standard deviation is equal to the mean. Empirical studies of "real world" arrival distributions show that standard deviations are much worse than exponential *and* the traffic demonstrates "self similar behavior"

For a discrete process of length n $\{X_1, X_2, \ldots X_n\}$ with known mean and variance, an estimate of the autocorrelation may be obtained as

$$\hat{R}(k) = \frac{1}{(n-k)\sigma^2} \sum_{t=1}^{n-k} [X_t - \mu][X_{t+k} - \mu]$$

Self-similar behavior is technically defined as the autocorrelation of the number of packet arrivals per unit time being non-summable.

Self-similar behavior can be "roughly" described as
the variance of the number of arrivals *per unit time*
decreases more slowly than for exponentially distributed arrivals
as the value *unit time* is increased.

Self-similar behavior implies that if the number of arrivals in the last interval was >> than the mean it is likely to be so in this interval as well.

Self-similar behavior implies that if the number of arrivals in the last interval was << that the mean it is likely to be so in this interval as well.

==> Self-similarity + High variance in the number of arrivals per unit time leads to alternating build up of large queues and then draining them out.

Summary

If the variance in the arrival process is sufficiently small that the the arrival rate almost never exceeds the service rate, then positively correlated arrival (even self-similar) processes are not harmful.

However, if the variance is sufficiently large that the arrival rate substantially exceeds the service rate on a regular basis then positively correlated arrival (even self-similar) processes substantially exacerbate the problem!

Traffic shaping mechanisms

Leaky bucket

arriving packets at network admission points accumulate in the "bucket"
packets drain out of the buckets at a fixed rate (< maximum rate supported by the link)
packets arriving when the bucket is full are either
discarded or
cause the generating process to block
under heavy loads *guarantees* deterministic interarrival times

Token bucket

tokens accumulate in the bucket at a fixed rate.
tokens that arrive when the bucket is full are discarded.
before a transmission can occur a token must be claimed and destroyed
packets arriving when the bucket is full are either
discarded or
cause the generating process to block
token bucket *does* allow short (bucket sized) bursts to be injected at a rate higher than the steady state admission rate.

Observation
All links act like leaky buckets!

Fact: The aggregation of a relatively small number of *on/off* traffic sources with high variance holding times in the *on* or *off* states has characteristics consistent with self-similarity.

Flow control:

Constrain the maximum number of packets that can be outstanding on any connection.

Connection might mean

- Single app to single app
- Single imp to single imp
- Single host to single host

Traditional view:

- Works better the more sessions that are grouped together.
- Recent research in self-similarity calls this view into question

Disadvantages

- Result from bursty nature of net traffic.
- Controls strict enough to prevent congestion unduly limit net service delivery.
- Controls that permit high throughput and good service don't prevent congestion when many users demand large service bursts simultaneously.
- Solution: SNA/TCP use *dynamic window sizes.* (large when congestion is absent, smaller as congestion grows).

Usage:

- Internet (TCP window congestion window)
- SNA Virtual route pacing. (Service class IMP to IMP based.)

Choke packets:

Each output line monitored as follows:

$$u_{new} = a\, u_{old} + (1 - a) u_{instantaneous}$$

If an output line becomes congested

For each packet transmitted on congested link

Send packet back to originating host telling host to reduce traffic by X percent.

Host receiving a choke packet will set a timer.

After timer expires normal traffic can be resumed.

Usage: SNA, ATM ABR

Deadlocks

Are a problem only with *reliable* network layers. Unreliable networks can always drop packets and *livelock* instead!

Direct store and forward..

All of A's buffers are filled with packets for host at B.
All of B's buffers are filled with packets for host at A.
Neither A or B can successfully receive and thus transmit a packet.
Sample code studied in class potentially had this problem.

Indirect store and forward..

All of A's buffers are filled with packets for host at B.
All of B's buffers are filled with packets for host at C.
All of C's buffers are filled with packets for host at A.

Reassembly deadlock (See RFC 626)

An example occured at UCLA, Dec 21 1973 on the old ArpaNet (Pre-IP)
Dest IMP was responsible for reassembly.
All buffers can became full of partially reassembled datagrams.

Internetworking:

Connecting of networks differing in:

Underlying technology: LAN, Satellite, Store and Forward WAN, HSN
Administrative control: Computer Science, DCIT, Engineering.
Network architecture: TCP/IP, SNA, ISO/OSI

Layers at which internetworking can occur:

Layer 1: Repeater.
Layer 2: Bridge
Layer 3: Gateway (router)
Layer 4+: Protocol converters (aka Application gateways)

Bridges

Advantages over repeaters:

Reduced traffic.
Some security.

Problems inherent in bridging 802.X and 802.Y

Different frame formats.
Different max packet sizes.
Different bit transmission rates.
Absence/presence of priority.
AC bits.

In practice 802.X is bridged only to 802.X -- and even that causes some interesting problems with respect to priority and AC bits when X = 4 or 5

Internetworking at the Link layer

Transparent (spanning tree) bridge.

Objective: Plug and play (no mods to software necessary).

Operates in promiscuous modes on all links

Source LAN = Dest LAN ==> Discard
Source LAN != Dest LAN ==> Forward
Dest unknown ==> Flood.

Backward learning used to know where to forward.

Spanning tree algorithm needed because of possible cycles

Tree constructed every few seconds via flooding device ID.

Spanning tree alg:

1 - Elect a *root bridge*
2 - For each *bridge* calculate *distance to root bridge*
3 - For each *LAN* select the *designated bridge* = closest to the root bridge
4 - For each *bridge* select the *root port* = port closest to root bridge
5 - Select remaining ports to be included in spanning tree = those for which "self" is a designated bridge

Every bridge has a root port

1 - The root port is always upstream toward the root bridge
2 - The root port is always in the spanning tree

A bridge may or may not have designated ports

1 - Designated ports are always downstream away from the root
2 - The actual spanning tree consists of LAN's with each LAN attached to a designated port on "root end" and a root port on the other end.

Root bridge election

Configuration messages have the following format:

(Root ID, Transmitting Bridge ID, Cost (Distance to root), Port ID)

Given two messages C1 and C2, C1 is better than C2 if:

Root ID in C1 is numerically smaller than root ID in C2
If Root ID's are equal break ties on cost = distance to root
If costs are also equal break ties on Transmitting bridge ID
If Transmitting bridge ID's are equal break ties with Port ID.
(Happens only if transmitter has 2 ports on same LAN)

Each bridge starts by sending a message with itself as root

If it receives a better message *on one of its attached LAN's* it
updates Root ID and Cost its own message using the new information
ceases transmitting configuration messages *that LAN.*

When algorithm stabilizes only one bridge per LAN transmits.

That bridge is called the designated bridge for that LAN

Calculating cost to Root

There will be a "wining" configuration message received on each port
That message will be transmitted by the designated bridge for that LAN.
The lowest (numerically) root id received is the root bridge.
The cost from *this* bridge to the root is computed by adding 1 to that cost.
The designated bridge for a LAN is the one with the lowest cost to the root.

Calculating Root port

Root port is the port with the lowest cost path to the root (subject to usual tie breakers.)

Selecting designated bridge

If my message is the best (hence) ultimately only message transmitted on the LAN then I am the designated bridge for that LAN

Selecting designated ports

If i am the designated bridge for a LAN, then my lowest numbered port on that LAN is the designated port. If I have only one port per LAN then my port on any LAN for which I am designated bridge is a designated port.

Forwarding rules

Forward all broadcast and destination unknown packets on from ports in spanning tree to other ports on spanning tree.
Discard all data packets received on ports not in tree.
Continue to run spanning tree alg on *all* ports
Packets traveling upstream are Tx'd on root ports (or by hosts) and received on designated ports (or by hosts).
Packets traveling downstream are Tx'd on designated ports and received on root ports.

Disadvantages

The primary problem is that since the tree is constructed based upon totally arbitrary bridge ids, strongly sub-optimal routes may be constructed.

I think that I shall never see
A graph more lovely than a tree.

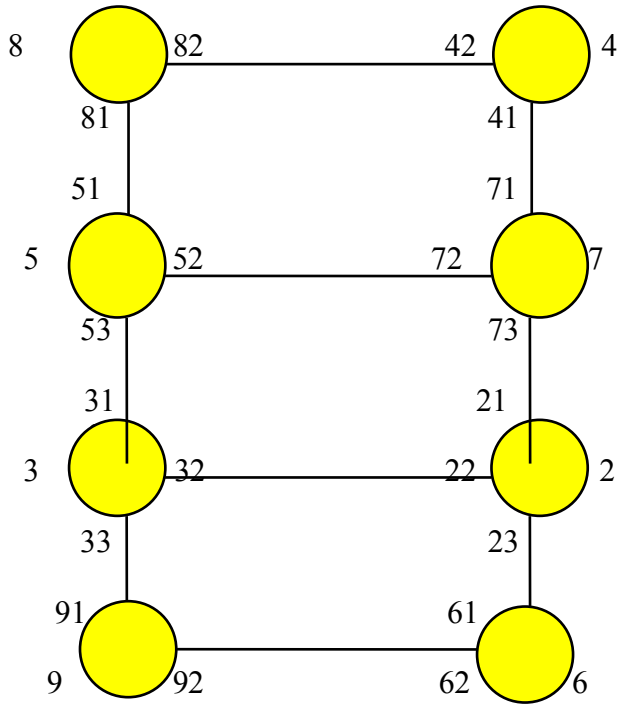
A tree whose crucial property
is loop-free connectivity.

A tree that must be sure to span
So packets can reach every LAN.

First, the root must be selected.
By, ID, it is elected.

Least cost paths from root are traced.
In the tree, these paths are placed.

A mesh is made by folks like me,
Then bridges find a spanning tree.
- Radia Perlman



Source routing bridge

Each LAN has a 12 bit sequence number
Each Bridge has a 4 bit sequence number
Sender node specifies exact route each packet is to take.
Route looks like B2-L4-B3-L5-B2-L6
If route is unknown a *discovery* packet is flooded through the net.
As discovery packets are forwarded, the route is recorded.
When a discovery packet reaches the destination, a reply is sent on the reverse route.
The first reply received "wins".

Summary of bridge differences:

Issue	**Transparent**	**Source routing**
Orientation	Connection less	Connection oriented (SNA)
Transparency	Fully transparent	End user transparent
Configuration	Automatic	Net manager at install
Routing	Spanning tree	Optimal
Route identification	Backward learning	Flooding
Failures	Handled in net	Host software
Complexity	Bridges	Bridges + Hosts

Systems Network Architecture

Connectivity Elements

The *transmission group* (TG) is the basic unit of connectivity.

TG is one or more links connecting two IMPS.

0 - 255 TG's between any node pair.

Routing affinity is TG based.

TG activation and deactivation:

First active link makes TG active.

Last link to fail makes TG inactive.

Assigning links to TGs.

Many TG's with few links ==> better workload partitioning.

Few TG's with many links ==> greater the reliability (and the resequence problem.)

Routing Elements

Explicit route (ER) is the basic end to end routing unit.

An ER must be invertible to be usable.

An ER is uniquely identified by a (Dest IMP#, ER #)

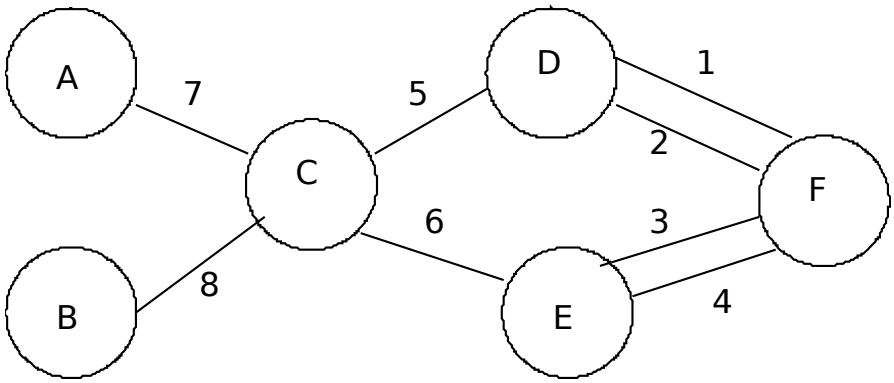
The ER must be defined by net admin to each imp through which it passes.

ER numbers range from 0, to 15

Example routing table for node F:

Dest IMP	**ER #**	**TG_ID**
A	4	2
	5	3
B	3	2
	4	1
C	1	1
	2	2
	3	4

Note that it is the combination of (DEST, ER#) that actually identifies the Route.
Duplicate ER#'s are supported at each node.



Also note that it would *not be possible* at nodes A and B to define the following ER's

A => C => D => F
B => C => E => F

and call them both (F, 3).

However it *would be possible* to define the following ER's

A => C => D => F
B => C => D => F

and call them both (F, 3).

If two ER's having the same dest IMP and number ever pass through the same IMP, they must merge from that point forth.

ER States

Inoperative: Some TG's in the ER thought not to be operative.
Operative: All TG's in the ER thought to be operative.
Active: All TG's verified to be active and ER verified to be invertible.

Each time a TG becomes operative:

IMPS at each end exchange a list of ER's known to them.

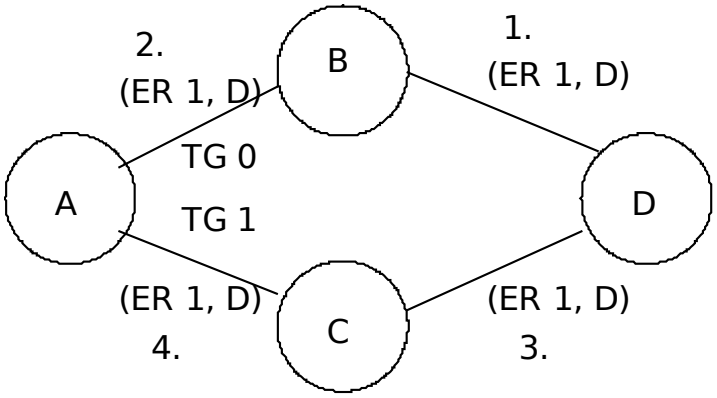
List is propagated through the net with selective flooding.

If I get a record for (J, 5) but I don't have (J, 5) defined in my *static* tables then I discard the entry.

Each time a TG becomes inoperative

A list of ER's passing through the down link is propagated via selective flooding.

Example of an ER's becoming operative.



1. TG connecting B and D becomes active. D sends NC_ER_OP (D, 1) to B.
2. TG connecting B and A becomes active. B sends NC_ER_OP (D, 1) to A. A now adds (D, 1) to ER state table specifying TG 0 as a candidate link.
3. TG connecting C and D becomes active. D sends NC_ER_OP (D, 1) to C.
4. TG connecting C and A becomes active. C sends NC_ER_OP (D, 1) to A. A now adds an entry for TG 1 to the state table.

ER State table:

Maintains the state of all ER's that are defined to exist at an IMP by the system administrator and for which at least one possible TG is active.

Dest	ER#	State	Reverse ER Mask	ER Length	TG-1	TG-2	TG-3
A	4	OP			2		
	5	OP			3	2	
B	3	ACTIVE		6	5	2	1
	4	OP		2	2		
C	1	IN_OP		1	1		
	2	IN_OP		2	2		
	3	IN_OP		4	4		

Session Activation

The primary session partner makes a request of the form (Dest IMP, Class of Service).
Primary host determines a list of suitable (VR ID's, Dest IMPs) and passes it to VR mgr.
If VR is already active, it's ID is returned and used.
If not the VR manager passes the ER manager the first (VR ID, Dest IMP).
The ER manager uses the ERN_MAP_LIST to map the VR to its underlying ER.

What happens next depends on state of the ER.

Active: Returns ER_ACTIVATED to VR Manager.
Inoperative: Returns ER_NOT_ACTIVATED to VR Manager.
Operative: Returns the result of an attempt to activate the VR.

ER Activation.

Attempt to send ACTIVATE_ER to the requested destination:

At each hop bits in the reverse ER mask for which there is no ER to the source are zeroed out.

If ACTIVATE_ER makes it to the other end and a reverse ER exists, the route is activated.

Alternate routes:

ER marked contend in static tables.

Alternate TG's exist in state table.

Multiple ACTIVATE_ER's sent.

First one back wins.

Congestion control in SNA

Mechanisms:

- Flow control
  - Session level
  - VR level

- Choke packets
  - VR level

Flow control is called *Pacing* in SNA

Bits in FID4 TH identify a PIU as a pacing request or pacing response.
PIU's are sent in groups called *pacing groups*
First packet of each group has pacing request bit set.
Complete group sent but no pacing response => wait.

State variables used:

Min pacing group size:	Def: # of hops.
Max pacing group size:	Def: 3 x # of hops.
Current pacing group size:	Init: Min PGS
Pacing count:	Init: Min PGS

Responding to moderate congestion in an intermediate node

Set CWI in each PIU flowing in forward direction.
End node sets CWRI in its next pacing response.
Receiver of CWRI must decrease current pacing group size by 1.

Responding to server congestion

Set RWI in all PIU's flowing on the reverse of congested ER(s).
Receiver of RWI must set current pacing group size to min PGS.

End nodes can just withhold pacing responses.

Network Layer In Internet Protocol IP

IP is the network layer protocol of the "TCP/IP" protocol family.
IP is an unreliable datagram based protocol designed to move datagrams in an internet.
Relationship among Internet Protocols:

```
+------+ +-----+ +-----+     +-----+
|Telnet| | FTP | | TFTP| ... | ... |
+------+ +-----+ +-----+     +-----+
      |   |        |           |
     +-----+     +-----+     +-----+
     | TCP |     | UDP | ... | ... |
     +-----+     +-----+     +-----+
        |           |           |
   +---------------------------+----+
   |    Internet Protocol & ICMP   |
   +---------------------------+----+
                  |
   +---------------------------+
   |   Local Network Protocol  |
   +---------------------------+
```

IP datagram header:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version: 4 bits

Indicates the format of the internet header. This document describes version 4.

IHL: 4 bits

Internet Header Length is the length of the internet header in 32 bit words..

The minimum value for a correct header is 5, maximum 15.

Type of Service: 8 bits

Provides an indication of the abstract parameters of the quality of service desired. To be used to guide the selection of the actual service parameters.

Bits	0-2: Precedence (000b – 100b derived from US military signal procedures)	
Bit	3: 0 = Normal Delay,	1 = Low Delay.
Bits	4: 0 = Normal Throughput,	1 = High Throughput.
Bits	5: 0 = Normal Reliability,	1 = High Reliability.
Bit	6: 0 = Normal cost	1 = Minimize cost
Bit	7: Reserved for Future Use.	

Precedence

111 - Network Control	110 - Internetwork Control	101 - CRITIC/ECP
100 - Flash Override	011 - Flash	010 - Immediate
001 - Priority	000 - Routine	

Differentiated Services. ..

TOS bits have been recently redfined as *diffserv* bits (no one used them anyway) This field is defined in RFC 2474 and obsoletes the TOS field.

00	***01***	***02***	***03***	***04***	***05***	***06***	***07***
Codepoint						unused	

Codepoint. 6 bits.

Codepoint	***Description***	***References***
000000	CS0	RFC 2474
001000	CS1	RFC 2474
010000	CS2	RFC 2474
011000	CS3	RFC 2474
100000	CS4	RFC 2474
101000	CS5	RFC 2474
110000	CS6	RFC 2474
111000	CS7	RFC 2474
001010	Assured Forwarding 11	RFC 2597
001100	Assured Forwarding 12	RFC 2597
001110	Assured Forwarding 13	RFC 2597
010010	Assured Forwarding 21	RFC 2597
010100	Assured Forwarding 22	RFC 2597
010110	Assured Forwarding 23	RFC 2597
011010	Assured Forwarding 31	RFC 2597
011100	Assured Forwarding 32	RFC 2597
011110	Assured Forwarding 33	RFC 2597
100010	Assured Forwarding 41	RFC 2597
100100	Assured Forwarding 42	RFC 2597
100110	Assured Forwarding 43	RFC 2597
101110	Expedited Forwarding PHB	RFC 2598, RFC 3246

Total Length: 16 bits

Total Length is the length of the datagram, measured in octets. It was once recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams *(This is no longer true)*

Identification: 16 bits

An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Flags: 3 bits

Fragmentation Control Flags.

Bit 0: reserved, must be zero

Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment.

Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.

```
  0   1   2
+---+---+---+
|   | D | M |
| 0 | F | F |
+---+---+---+
```

Fragment Offset: 13 bits

Indicates where in the datagram this fragment belongs.

Measured in units of 8 octets (64 bits).

Time to Live: 8 bits

Indicates the maximum time the datagram is allowed to remain in the internet system.
If zero, then the datagram must be destroyed.
This field is modified in internet header processing.
The time was originally measured in units of seconds
Every module that processes a datagram must decrease the TTL by at least one.
In present operations the TTL has become a pure hop counter. An initial value of 30 is common.

Protocol: 8 bits

Indicates the next (transport) level protocol used in the data portion of the internet datagram. Values for various protocols are specified in "Assigned Numbers" [9].

Header Checksum: 16 bits

A checksum on the header only.

Since some header fields change (e.g., *necessarily time to live*), this must be recomputed and verified at each point that the internet header is processed.

The checksum algorithm is:

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

It is provisional and may be replaced by a CRC procedure, depending on further experience.

Source Address: 32 bits

The source address.

Destination Address: 32 bits

The destination address.

Classful Address Formats (now deprecated)

```
High Order Bits   Format                            Class
---------------   --------------------------------  -----
      0            7 bits of net, 24 bits of host    a
      10          14 bits of net, 16 bits of host    b
      110         21 bits of net,  8 bits of host    c
      1110        Multicast addresses                d
```

Classless addressing: prefix(network part).hostpart/prefixlen

Allows for much more efficient division of the old Class A space.

Operational issues

Routing -

Routers in stub and mulitihomed AS's rely on *default routes* to reach all networks outside the AS and some networks inside the AS.

Routers in transit AS's rely on OSPF to construct *intra AS* routes and BGP to construct *inter AS* routes.

All *inter AS* routing is classless. Large routing tables (> 250,000 entries!) have the form

prefix/prefixlen | next hop IP address

Destination addresses are full 32 bit addresses, so the table matching algorithm is *longest matching prefix wins*.

Routing tables could be *much smaller* if IP addresses could be *reallocated* using geographic and ISP affinity.

Congestion control -

TCP flow control is the basis for congestion management in the internet.

The first packet exchange in a TCP connection is a *stop and wait* exchange.

The size of the window of unacked packets then grows until a *drop* occurs.

The occurence of a *drop* causes the window size to be reduced by half.

Window size gradually grows back in the absence of further drops

Much research has been conducted on the effect of router dropping algorithms on:

the impact on the performance of specific TCP sessions
fairness among competing TCP sessions
global internet performance

IP - V6

Design objectives

- Support billions of hosts
- Reduce size of routing tables (in *transit* AS's)
- Allow faster processing within routers
- Improve security (authentication and privacy)
- Better support for different classes of service
- Better support for multicasting
- Allow a host to roam without having to change IP address (mobile IP)
- Support continued evolution
- Support indefinite coexistence with IPV4.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| DS Byte        |    Flow Label                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Payload Length             |  Next header  | Hop Limit     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-                                                         -+-+
|                                                               |
+-+-                   Source  Address                       -+-+
|                                                               |
+-+-                                                         -+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-                                                         -+-+
|                                                               |
+-+-                   Dest    Address                       -+-+
|                                                               |
+-+-                                                         -+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version: 4 bits

The value 6

DS (Diffserv) byte 8 bits

This field is used by the source and routers to identify the packets belonging to the same traffic class and thus distinguish between packets with different priorities. Values have the same meaning as the new diffserv byte in IPV4

Flow label: 20 bits

An experimental mechanism for establishing *pseudo connections*

A non zero value => flow label field is significant

A flow is uniquely identified by a (source IP, dest IP, flow label)

Support for flows should make it possible to

Reserve resources

Provide service guarantees

Payload length: 16 bits

Amount of data *following* this header.. IPV4 length included header length

Next header: 8 bits

An overloaded field that specifies

One of 6 (currently defined) extension headers

or -- if this is the last header

one of the standard Transport protocol IDs (TCP=6, UDP=17, etc)

Hop limit 8 bits: Same as TTL in IPV4

"Features" of IP V4 omitted in V6

Fragmentation eliminated

All routers required to support 576 byte datagrams

Larger datagrams treated just like DF was set

Objective -- eliminate inefficiencies of fragmentation and reassembly

Header checksum eliminated

Had to be recomputed by each router

Source and Destination address: 16 bytes (128 bits!)

Address allocation

Prefix	**Usage**	**Number**
0000 0000	Reserved (including IP V4)	2^{120}
0000 0001	Unassigned	2^{120}
0000 001	OSI NSAP addresses	2^{121}
0000 010	Novell Netware IPX addresses	2^{121}
0000 011	Unassigned	2^{121}
0000 1	Unassigned	2^{123}
0001	Unassigned	2^{124}
001	Unassigned	2^{125}
010	Provider based addresses	2^{125}
011	Unassigned	2^{125}
100	Geographic based addresses	2^{125}
101	Unassigned	2^{125}
110	Unassigned	2^{125}
1110	Unassigned	2^{124}
1111 0	Unassigned	2^{123}
1111 10	Unassigned	2^{122}
1111 110	Unassigned	2^{121}
1111 1110 0	Unassigned	2^{119}
1111 1110 10	Link local use addresses	2^{118}
1111 1110 11	Site local use addresses	2^{118}
1111 1111	Multicast addresses.	2^{120}

Address notation: Colon separated hexadecimal

(Similar to 802.2 convention... but on 16 bit blocks)

8000:0000:0000:0000:0123:4567:89ab:cdef

Network Layer in ATM (Asynchronous Transfer Mode)

Service characteristics

Unreliable
Connection oriented (mostly)
Fixed size transfer units (cells)
48 bytes payload
5 bytes header

ATM is actually more of a rectangular solid than a stack of layers encompassing

Physical layer
Data link layer
Network layer specification
The slices of the solid are sometimes called *planes (e.g the signaling plane, the management plane)*

An ATM connection is called a *virtual channel connection (VCC)*

Point to point channels are normal but
Multicast channels are permitted.
Channels are *simplex* but can be paired to provide duplex service
(with direction dependent! service guarantees.)
Packets may be discarded .. but may *not* be reordered.

A two level connection hierarchy exists

A virtual *path* may contain multiple virtual channels
Core routing was designed to be done on a VP basis
VP setup requires some serious processing at each intermediate node
VC setup requires only endpoint participation -- assuming resources can be suballocated from the VP

Cell formats

UNI (User to network interface)

Bits	ID	Description
4	GFC	General flow control
8	VPI	Virtual path identifier
16	VCI	Virtual channel identification
3	PTI	Payload type
1	CLP	Cell loss priority
8	HEC	Header Error Control

NNI (Network to network interface)

Bits	ID	Description
12	VPI	Virtual path identifier
16	VCI	Virtual channel identification
3	PTI	Payload type
1	CLP	Cell loss priority
8	HEC	Header Error Control

Field significance

GFC - Should be ignored
VPI - Note that hosts have only 256 "bundles" available
VCI - Some (1-31) are "reserved" for control functions
PTI - Set by transmitter but may be modified in transit
CLP - 0 => high priority - try not to drop
HEC - $X^8 + X^2 + X + 1$ - Is used for single bit ECC and multibit ED

PTI data (AAU == ATM User to ATM User)

PTI	Meaning
000	User cell - no congestion - AAU = 0
001	User cell - no congestion - AAU = 1
010	User cell - congestion - AAU = 0
011	User cell - congestion - AAU = 1
100	Maintenance info between switches
101	End - to - end maintenance inf
110	Resource management info (Available bit rate (ABR) cong control)
111	Reserved.

Connection management

Both permanent and switched (dialed) VCC's are supported

Permanent

Like a leased line. Must be manually configured in the ATM switch by a *sysadmin.*

Switched

Like a phone call

Setup procedure defined in Q.2931

Multiple mechanisms exist for connection setup.. this is the ``standard'' two-step procedure

Send a request to create and new connection on VPI 0 VCI 5

If this request is accepted a new VC id (VPI=n, VCI=m) is returned by the switch

The Virtual Circuit Assignment Problem (also known as *label switching*):

A virtual circuit is to be set up connecting node A with node H.
It passes through nodes D, F, and G in that order.

Problem: How do we give it a unique circuit identifier in a distributed way:

Suppose node A decides to call it circuit 13.
Suppose node D already is using a circuit 13.

Solution *(label switching):* Let the circuit be known by a different identifier at each hop.

Basic algorithm:

1. Host wishing to create a new outbound VC chooses the smallest available circuit number and sends setup packet to its IMP.
2. That number is entered into IMP's incoming table with "H" for source identifier.
3. The routing manager within the IMP picks the next hop for the packet.
4. The first unused outbound circuit number to that IMP is selected as the outgoing number and is placed in the setup packet.
5. The next hop's identity and the selected circuit number are placed in the outgoing column of the same row of the IMP's routing table.
6. The packet is forwarded and the process repeats until the destination is reached.

How does one handle duplex circuits?

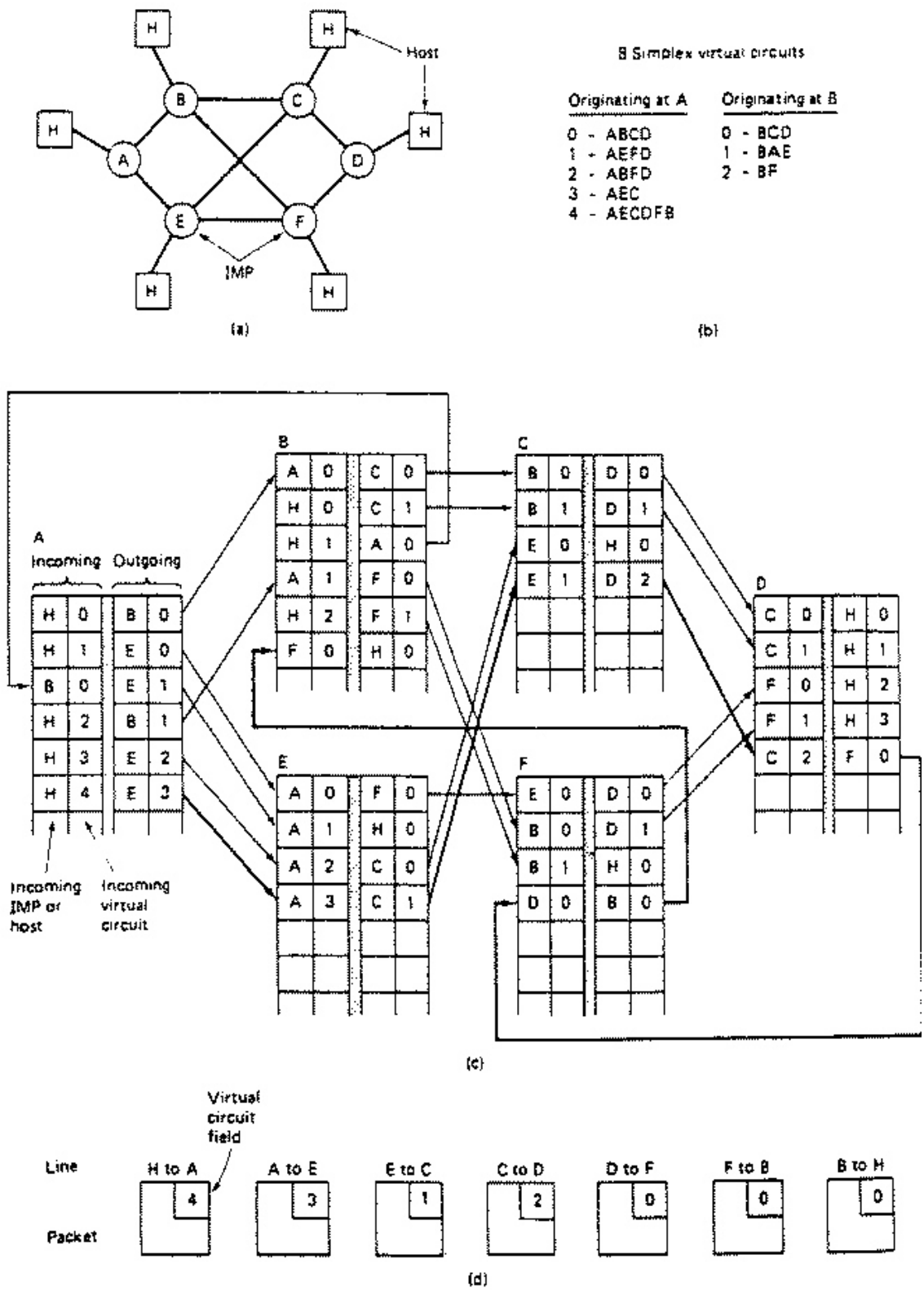


Fig. 5-6. (a) Example subnet. (b) Eight virtual circuits through the subnet. (c) IMP tables for the virtual circuits in (b). (d) The virtual circuit changes as a packet progresses.

Virtual Channel Establishment

Message	**Host meaning**	**Direction**
Setup	Please establish VC to specified dest	H -> N
Call Proceeding	Your setup packet looks OK.	N -> H
Connect	Your call was accepted at other end	N-> H
Connect Ack	Good, I'm ready to transmit	H -> N
Release	I'd like to hang up now	H -> N
Release Complete	Ack for release	N -> H

Actual flow is point to point across the *entire* path.

Source	Msg	Network	Msg	Dest
--->	Setup			
	Call Proc	<---		
		--->	Setup	
			Call Proc	<---
			Connect	<---
		--->	Connect Ack	
	Connect	<---		
--->	Connect Ack			
--->	Release			
	Release Ack	<---		
		--->	Release	
			Release Ack	<---

Addressing in ATM

ATM addresses are 20 bytes long. Three formats have been defined:
The first byte specifies the form of the address:

Form 1 (0x39): 20 bytes based on OSI addresses
Bytes 2-3: Country code
Byte 4: Format of rest of address
Byte 5 - : Authority, domain, area, address+
Form 2 (0x47):
Bytes 2-3: International Org code
Remainder: See example below:
Form 3(0x45) :
CCITT E.164 (International telephone number)

Form 2 addressing example:

The ATM network identifier (first 9 bytes)

An initial byte of 0x47 implies that the next two bytes represent the ICD (International Code Designator) which in turn identifies the international organization that owns the remaining address space.

The next 6 bytes effectively identify an *ATM Network*, that is, a collection of interconnected switches and end-systems operated by an administrative entity. In theory these identifiers are handed out by the international organization identified by the ICD.

In practice most ATM LAN's are private and it is common to use:
47.02.03.04.05.06.07.08.09 as the first 9 bytes.

The ATM switch identifier (next 4 bytes)

IBM proposed a hierarchy in which

the first two bytes represent the routing domain number *RDN*
the next represents the ATM cluster number *ACN*
and the last the individual switch or hub number *HN*.

This strategy predated PNNI but remains relevant for connecting PNNI and non-PNNI hosts.

The host component of the address (final 7 bytes)

The next 6 bytes of the ATM address are the end system identifier *ESI* and correspond to the *MAC* address of an ethernet component. These bytes are "hardwired" into the NIC at the factory.

The final byte is called the selector

It allows a single host or switch to host up to 255 distinctly addressable services.

In summary

The first 13 bytes of any host's ATM address are inherited from the switch to which it is attached using a protocol called ILMI (Interim local management interface).

The next 6 bytes are derived from a globably unique address built into the ATM NIC at the time it is manufactured (just as with Ethernet NICs).

The last byte can be viewed as a logical "port" number.

Service categories in ATM

Originally based on three parameters:

Real time vs non real time requirements
Fixed vs variable bit rate requirements
Connection oriented versus connection less

Original service classes (4 of the possible 8 combinations were specified)

A - Real time, fixed rate, connection oriented
B - Real time, variable rate, connection oriented
C - Non real time, variable rate, connection oriented
D - Non real time, variable rate, connectionless

Traffic categories in ATM (Have eclipsed the original service classes in importance)

CBR - Constant bit rate
A guaranteed fixed bit rate service
Useful for carrying (multiplexed) present day telephone traffic

VBR - Variable bit rate
RT-VBR: Real time compressed video and audio
NRT-VBR: Non real time variable rate traffic.

ABR - Available bit rate.
Minimum rate guaranteed
Best effort at higher rates up to some Max
Rate feed back is provided here

UBR - Unspecified bit rate.
No limits on admission rate
No limits on *discard* rate

Relationship between the *Traffic Classes* and the *Service characteristics*

Service Characteristic	**CBR**	**RT VBR**	**NRT VBR**	**ABR**	**UBR**
Bandwidth guarantee	Yes	Yes	Yes	Opt	No
Real time guarantee	Yes	Yes	No	No	No
Suitability for bursty traffic	No	No	Yes	Yes	Yes
Congestion feed back	No	No	No	Yes	No

Quality of Service (QoS) and the Q.2931 signaling protocol.

Requirements are specified and agreed upon during call setup. These messages are exchanged by switches using the SSCOP link protocol that we studied earlier

```
(991082451.620699) TO NETWORK (0.0.5):
  _pdsc = 9 "Q.2931 user-network call/connection control message"
  _cr_len = 3
  call_ref = 1 (0x1)
  msg_type = 0x05 "SETUP"
  _ext = 1
  _flag = 0 "instruction field not significant"
  _action_ind = 0 "clear call"
  msg_len = 104 (0x68)
    _ie_id = 0x58 "ATM adaption layer parameters"
      _ext = 1
      _cs = 0 "ITU-T standardized"
      _flag = 0 "instruction field not significant"
      _action_ind = 0 "clear call"
      _ie_len = 11 (0xb)
        aal_type = 5
          _id = 0x8c "Forward maximum CPCS-SDU size"
            fw_max_sdu = 1516 (0x5ec)
          _id = 0x81 "Backward maximum CPCS-SDU size"
            bw_max_sdu = 1516 (0x5ec)
          _id = 0x83 "AAL mode (UNI 3.0 only)"
            aal_mode = 1 (0x1)
          _id = 0x84 "SSCS type"
            sscs_type = 0 (0x0)
    _ie_id = 0x59 "ATM traffic descriptor"
      _ext = 1
      _cs = 0 "ITU-T standardized"
      _flag = 0 "instruction field not significant"
      _action_ind = 0 "clear call"
      _ie_len = 9 (0x9)
        _id = 0x84 "Forward peak cell rate (CLP=0+1)"
          fw_pcr_01 = 353207 (0x563b7)
        _id = 0x85 "Backward peak cell rate (CLP=0+1)"
          bw_pcr_01 = 353207 (0x563b7)
        _id = 0xbe "Best effort indicator"
    _ie_id = 0x5e "Broadband bearer capability"
      _ext = 1
      _cs = 0 "ITU-T standardized"
      _flag = 0 "instruction field not significant"
      _action_ind = 0 "clear call"
      _ie_len = 3 (0x3)
        bearer_class = 16 "BCOB-X"
        _ext = 0
          _ext = 1
          trans_cap = 0x00 "Non-real time VBR (reception only)"
        _ext = 1
        susc_clip = 0 "not susceptible to clipping"
        upcc = 0 "point-to-point"
```

```
_ie_id = 0x5f "Broadband low-layer information"
  _ext = 1
  _cs = 0 "ITU-T standardized"
  _flag = 0 "instruction field not significant"
  _action_ind = 0 "clear call"
  _ie_len = 9 (0x9)
    _lid = 3
      uil3_proto = 0x0b "ISO/IEC TR 9577"
        _ext = 0
          _ext = 0
          ipi_high = 0x40
            _ext = 1
            _ipi_low = 0
              _ext = 1
              _snap_id = 0
              oui = 41022 (0xa03e)
              pid = 1 (0x1)
_ie_id = 0x70 "Called party number"
  _ext = 1
  _cs = 0 "ITU-T standardized"
  _flag = 0 "instruction field not significant"
  _action_ind = 0 "clear call"
  _ie_len = 21 (0x15)
    _ext = 1
    _plan = 2 "ATM endsystem address"
      _type = 0 "unknown"
      cdpn_esa = 47 2 3 4 5 6 7 8 9 0 0 3 1 0 1 2 3 4 5 2
_ie_id = 0x6c "Calling party number"
  _ext = 1
  _cs = 0 "ITU-T standardized"
  _flag = 0 "instruction field not significant"
  _action_ind = 0 "clear call"
  _ie_len = 21 (0x15)
    cgpn_plan = 2 "ATM endsystem address"
    cgpn_type = 0 "unknown"
    _ext = 1
    cgpn = 47 2 3 4 5 6 7 8 9 0 0 3 1 0 0 77 88 a1 15 0

_ie_id = 0x5c "Quality of service parameter"
  _ext = 1
  qos_cs = 3 "Standard defined for the network"
  _flag = 0 "instruction field not significant"
  _action_ind = 0 "clear call"
  _ie_len = 2 (0x2)
    qos_fw = 0
    qos_bw = 0
```

Other parameters:

Peak cell rate	PCR	Max rate in cells/second
Sustained cell rate	SCR	Long term average cell rate
Minimum cell rate	MCR	Minimum acceptable cell rate
Cell delay variation tolerance	CDVT	Maximum acceptable jitter
Cell loss ratio	CLR	Limit on percentage of lost cells
Cell transfer delay	CDT	Min and max delivery time
Cell delay variation	CDV	The variance of the actual CDT
Cell error rate	CER	Fraction to be delivered w/o error

Congestion management in ATM

For CBR and VBR call admission control is the only viable solution

For UBR cell discarding is used.

For ABR a congestion feedback mechanism is used

After every *k* cells, a resource management (RM) cell must be transmitted.
RM cell contains the rate the sender would *like* to transmit at the moment.
This value is called the *ER* explicit rate
Congested switches may reduce the *ER* before forwarding the RM cell
The RM cell is reflected back to the sender at the destination and may have the *ER* lowered on the reverse path as well.
On receiving the RM cell the sender is expected to set its *ACR* actual cell rate to the valued specified as the *ER*.

Other mechanisms

Switches can also initiate RM cells (choke packets) specifying an *ER*.
Switches can also set the congestion bit in data cells.
Unfortunately these cells may get dropped!

Thus ABR senders are expected to monitor for lost RM cells and reduce ACR accordingly

Routing in ATM

PVC's (static routing) are commonly used in trunk routes today

Other forms of static routing are used in ATM LAN's

PNNI (Private Network to Network Interface) is a dynamic routing protocol through which switches automatically deduce the topology of private ATM networks and perform routing.

Only the 1st 13 bytes of an ATM address are relevant to call routing