The Network Layer

Dr. G. A. Marin

Principal Network Layer Functions

- Gets packets from source to destination (across network)
- Chooses "optimal" paths
- Balances loads in routers and on links
- Knows/updates topology information
- Deals with problems at subnet boundaries.
- Provides services to transport layer.

Services to Transport Layer

- Connectionless (Internet)
  - Send a receive packets independently
  - Each may take a different route
  - No sequencing, no flow-control
- Connection-oriented (Telephone Companies)
  - Set up a connection, give it an identifier and use this for all routing until connection terminates.
  - Agree on parameters like class of service and cost
  - Deliver packets in sequence, reliably
  - Provide flow control to keep sender from overwhelming the receiver.

Connection-oriented subnet principles

- Setup virtual circuits to avoid choosing a new route for each packet (or cell)
- Router maintains a table with one entry per VC.
  - Each packet contains the VC number in header
  - Router sees VC number and knows the inbound port. Places on right outbound port and may change the VC (local significance).
- VC must be taken down when user finishes.
  - VCs are limited resource
  - Associated costs

Connectionless Subnet Principles

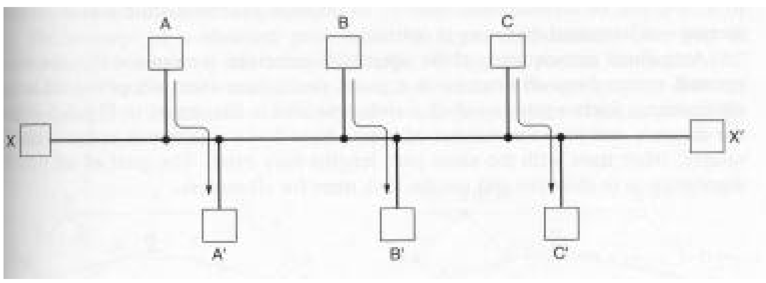
- No routes computed in advance.
- Each packet routed independently.
  - Advantage: adapt to failures quickly
- Router table now says which outbound port to use to reach a particular destination router.
  - Such a table also needed in conniection-oriented case to set up connections.
- Each packet must contain full destination address
  - May be long

Comparison of VC and Datagram Subnet

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

Goals for Routing Algorithms

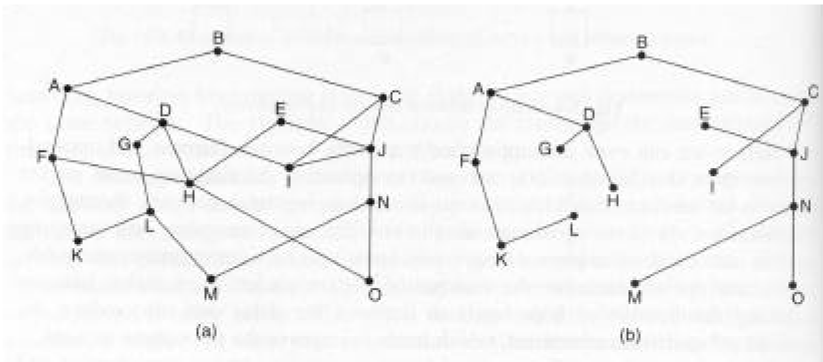
- Simple
- Correct
- Robust (under failures/changes)
- Stable
- Fair
- Optimal



Many choices for "optimal" route

- Minimize delay for user
  - Transmission plus queueing (waiting)
- Minimize total delay in network
- Maximize throughput for user/network
- Minimize cost
- Balance traffic load on network links
- Minimize number of hops in route

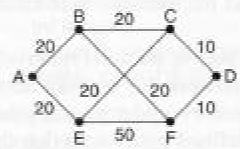
Optimal Routes and Sink Trees



Static Flow-based Routing (Do it yourself.)

- Consider traffic load in addition to topology
- Static means data-flow between each pair of nodes must be relatively stable (busy hour).
- Given capacity and average flow compute mean packet delay on each line of the subnet.
  - start with some initial routing choice
- Calculate a flow-weighted average packet delay for the whole subnet.
- Change the routes and try again...continue until mean packet delay is minimized.
- Must know traffic flow matrix [F_{ij}], capacity matrix [C_{ij}] and network topology.

Full duplex subnet: capacities(kbps), traffic(pps), routing matrix



(a)

Destination

Source	A	B	C	D	E	F
A		9 AB	4 ABC	1 ABFD	7 AE	4 AEF
B	9 BA		8 BC	3 BFD	2 BFE	4 BF
C	4 CBA	8 CB		3 CD	3 CE	2 CEF
D	1 DFBA	3 DFB	3 DC		3 DCE	4 DF
E	7 EA	2 EFB	3 EC	3 ECD		5 EF
F	4 FEA	4 FB	2 FEC	4 FD	5 FE	

(b)

Resulting Analysis of Line Delays and Traffic Weights

i	Line	Load Delay λ_i (pkts/sec)	Capacity C_i (kbps)	Max Load μC_i (pkts/sec)	T_i (msec)	Weight
1	AB	14	20	25	91	0.171
2	BC	12	20	25	77	0.146
3	CD	6	10	12.5	154	0.073
4	AE	11	20	25	71	0.134
5	EF	13	50	62.5	20	0.159
6	FD	8	10	12.5	222	0.098
7	BF	10	20	25	67	0.122
8	EC	8	20	25	59	0.098
Total:		82				

Mean delay per line is $T = 1/(\mu C-\lambda)$, where $1/\mu$ is mean pkt size of 800 bits and λ is mean arrival rate in pps.

Mean network delay is sum of T_i x W_i for i=1 to 8 which is 86 msec here.

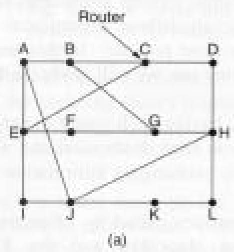
Flooding (automated brute-force)

- Each packet received is transmitted on every outbound line other than one received.
- Large amount of overhead traffic
- Always finds the shortest path (among others)
- Extremely robust (military applications)
- May be done with hop counts or directionality
- Generally impractical

Distance Vector Routing

- Each router maintains table of:
  - Best known distance to each destination router
  - Which link to use to go there
- Router updates table by exchanging information with its neighbors
- Each sends a vector of distances; hence, "distance vector" routing
  - How often?
  - Which metric? (queue length, delay, hops)
  - Must metric be measured?

Distance Vector Table: Router 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
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Delay from A, I, H, K — Vectors received from J's four neighbors

(b)

Counting to Infinity with distance vectors Linear network example.

Entries are what each node has as its distance to A.

Good news travels fast.

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

(a)

But when A is lost...

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)

Note: At each step in above examples, routers are trying to "converge."

Link State Routing

- Discover your neighbors and learn their network addresses.
- Measure delay/cost to each of your neighbors.
- Construct a packet telling what you've learned.
  - Sending info about the "state" of links to your neighbors based on your measurements; hence, "link state routing."
- Send this packet to all other routers.
- Compute the shortest path to every other router.

Through these steps each router learns complete topology plus measured delays. Then uses Dijkstra's algorithm to find shortest path to each other router. (Builds his own routes to each node.)

Dijkstra's algorithm for shortest path

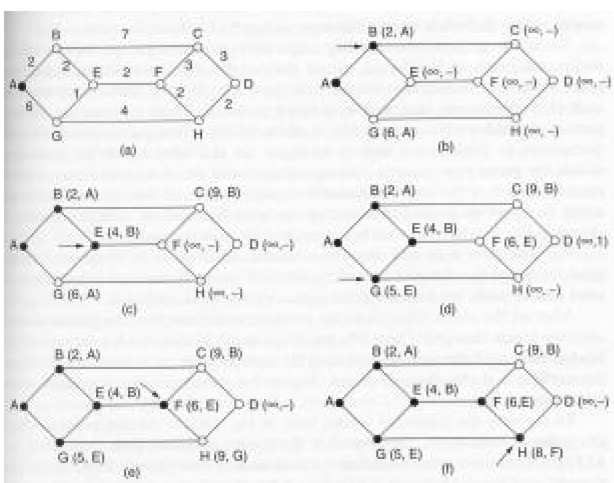
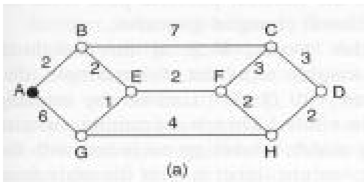


Table-based Dijkstra's

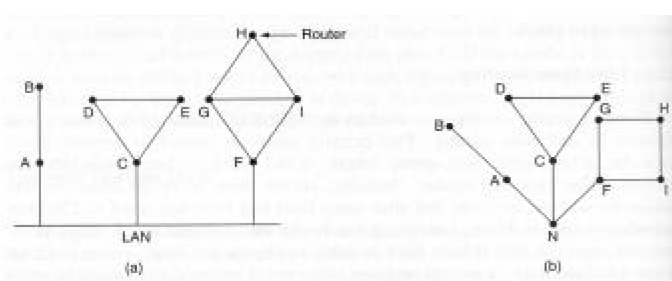


Shortest Path: ABEFHD
Length: 10

i	A	B	C	E	F	G	H	D	S
0	0								ϕ
1		2A				6A			A
2			9B	4B					B
3					6E	5E			E
4							9G		G
5							8F		F
6								10H	H
7								10H	C
									D

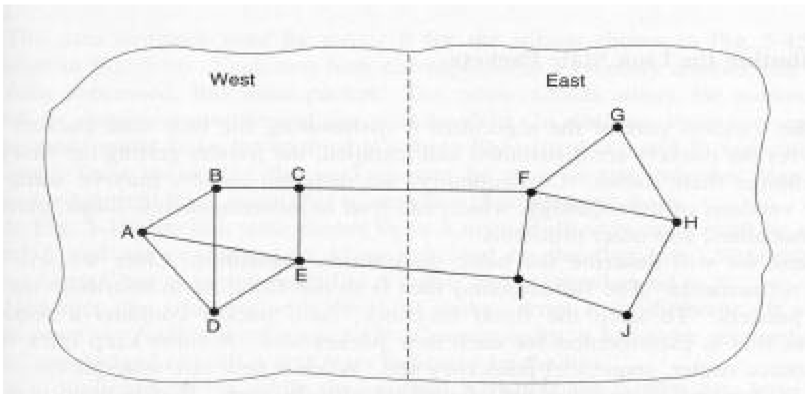
Learning about Neighbors

- When booted send "hello" packet to each neighbor.
- Neighbor responds with globally unique ID.
- LAN is modeled with a node sometimes called a "connection network."



Delay and Oscillation

- Measure delay using "echo" packet.
- Include queueing delay?
  - May cause oscillation.

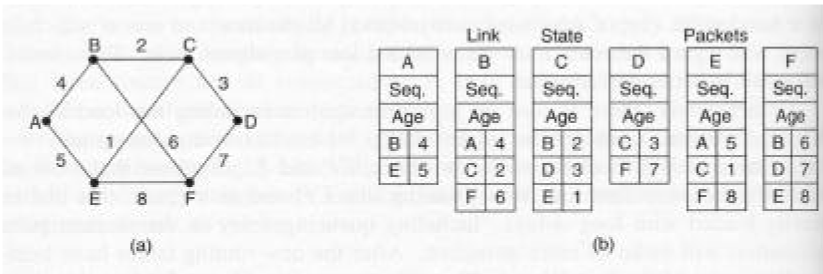


Distribute Link State Packets

- DANGER: Once the network starts updating different routers may be using different versions of topology leading to loops, etc.
- Flooding used to distribute link-state packets.
  - EXCEPT today a "spanning tree" is often built for this.
  - Each packet includes source, seq (incr), age (decr)
  - Seq numbers checked for duplicates which are discarded
  - New seq number forwarded to all neighbors old seq numbers (smaller than latest) are discarded.
  - 32-bit seq number to avoid wrap
  - Hold LS packet briefly to watch for more recent information.
  - All LS packets are ACKed.

Link State Example: Packets Sent

Each entry is delay to neighbor node.



When to update?

Packet Buffer at Router B

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

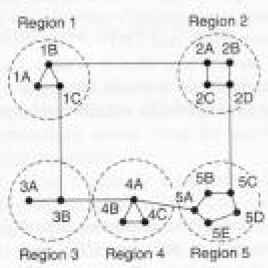
1. First packet to be sent to C,F and Acked to A
2. Second packet to be sent to A,C and Acked to F.
3. Third packet came from E through A and F.

If newer packet arrives before one of these sent, older one will be discarded.

Compute New Routes

- Done after link state packets have arrived.
- Done for each source-destination pair (in both directions) using Dijkistra's algorithm.
- Done by each router.
- Used to update routing tables.
- Examples
  - OSPF (now popular in Internet)
  - IS-IS adopted by ISO...supports multiprotocol

Hierarchical Routing



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

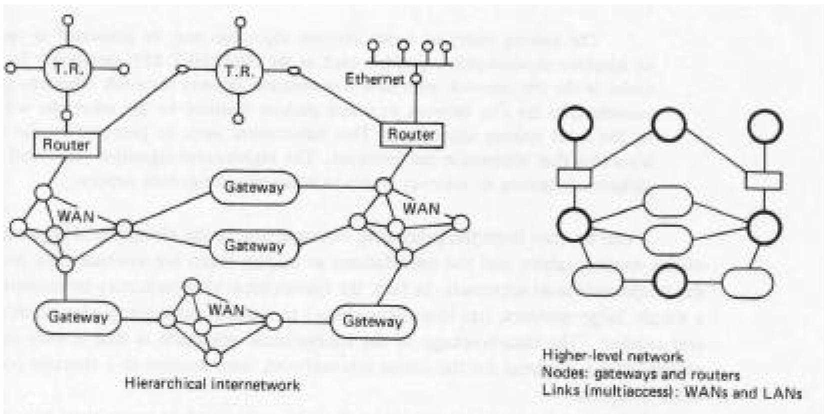
Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Note that best path from 1A to 5C is via Region 2, but this sends via Region 3.

Interconnected Subnetworks

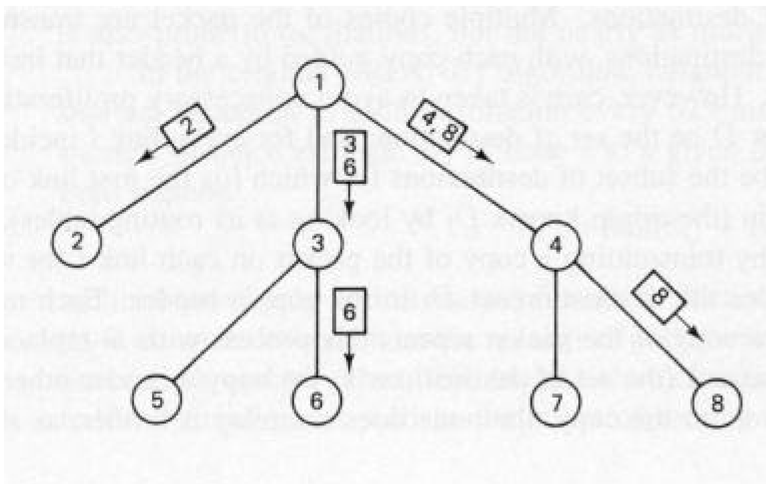


Higher level routing algorithm operates among gateways and routers. Lower level algorithms operate within each subnet.

Exterior Gateway Routing: BGP

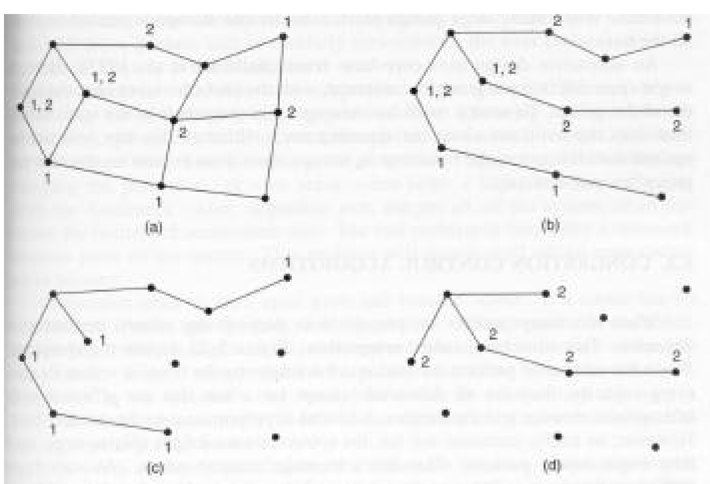
- Interconnects Autonomous Systems
- View of the world is other BGP routers
  - joined by transit networks
  - or by multiconnected networks
  - or by stub networks
- Enforce policies
  - Don't route from Pentagon through Iraq
  - Don't route from IBM through Microsoft
  - Don't route any traffic through my network unless adjacent network pays a fee.
- Uses a modified distance vector routing
  - Sends the entire route to each destination.

Multicast (multi-destination routing)

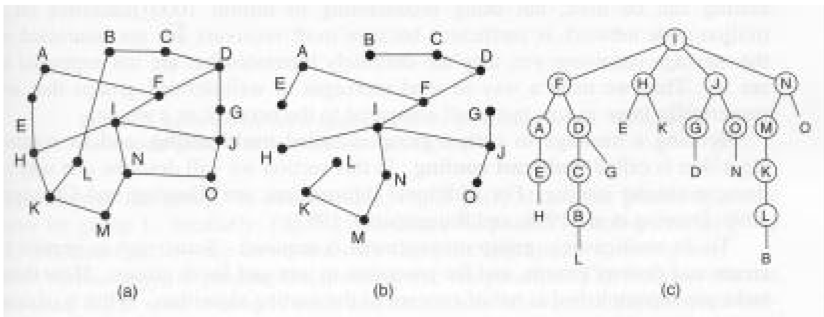


Example: Node 1 sends to nodes 2,3,4,6,8.

Pruning a spanning tree (when routers know about tree)



Reverse Path Forwarding
(when routers don't know spanning trees)

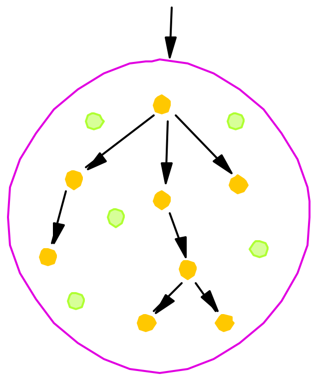


(a) original subnet
(b) spanning tree from router I
(c) reverse path forwarding tree from I
"Did packet arrive on my route TO the source?"

Multicast Routing

- Sending to a group so requires group management
- Ideal is then to forward on a spanning tree to the group
  - May be found as a subset of overall spanning tree
    - works well with link state routing
  - May be constructed independently by group members (distributed)
  - May use reverse forwarding algorithm
    - works with distance vector routing

Multicast Support



- **Group management**
  - join and leave sets
  - build point-to-multipoint trees
  - coordinate in support of multimedia application
- **Point-to-multipoint routing**
  - efficient?