Network Layer

Draw crude map. How to get from one host to another? If each link delivers reliably then is the whole route reliable—a router may fail, limited buffer space (may have to throw packets on the floor).

Data Link Layer—deals with machine-to-machine communication

Network Layer—lowest layer that deals with host-to-host communication, call this *end-to-end* communication.

Four issues:

1. interface between the host and the network (the network layer is typically the boundary between the host and subnet)
2. routing
3. congestion and deadlock
    - When more packets enter an area than can be processed, delays increase and performance decreases. If the situation continues, the subnet may have no alternative but to discard packets.
    - If the delay increases, the sender may (incorrectly) retransmit, making a bad situation even worse.
    - Overall, performance degrades because the network is using (wasting) resources processing packets that eventually get discarded.
4. internetworking (A path may traverse different network technologies(e.g., ethernet, point-to-point links, etc.)
    - packets may travel through many different networks
    - each network may have a different frame format
    - some networks may be connectionless, other connection oriented

Network Layer Interface

The network layer should shield the transport layer from having to know details of the underlying subnet (should not do anything different if sending across Ethernet or across the country on the Internet).

Should the host or the subnet be responsible for the delivery of *all* packets *in order*?

Datagram model: the host is responsible (datagrams are not guaranteed reliably delivered)

Virtual Circuit model: subnet is responsible. Connection-oriented.

Look at Fig. 5-4.

Two Views

- What type of applications want a virtual circuit? file transfer and remote login.
- What about datagrams? When data is small compared to the setup time (transactions).
- Datagrams can be used to implement VC's.

Two views (issue is "does error control belong in network or transport layer?"):

1. DARPA Internet community viewpoint: The subnet is unreliable no matter how it is designed. Thus, host are forced to do error control anyway. Given that they perform error control, why have the network layer duplicate the same function?

   The DARPA TCP/IP Internet is connectionless.

2. Common carrier viewpoint: The connection oriented approach is the right way. Users don't want complex error control protocols in host computers. User's want reliable, trouble-free service. (X.25)

Which group dominated the ISO standardization process?

Which group has real implementation experience?

Routing Overview

The network layer is responsible for routing packets from the source to destination. The *routing algorithm* is the piece of software that decides where a packet goes next (e.g., which output line, or which node on a broadcast channel).

For connectionless networks, the routing decision is made for each datagram. For connection-oriented networks, the decision is made once, at circuit setup time.

Routing Issues

The routing algorithm must deal with the following issues:

- Correctness and simplicity: networks are never taken down; individual parts (e.g., links, routers) may fail, but the whole network should not.
- Stability: if a link or router fails, how much time elapses before the remaining routers recognize the topology change? (Some never do..)
- Fairness and optimality: an inherently intractable problem. Definition of optimality usually doesn't consider fairness. Do we want to maximize channel usage? Minimize average delay?

We'll consider both adaptive—those that take current traffic and topology into consideration—and non-adaptive algorithms.

Routing

Routing is concerned with the question: Which line should router J use when forwarding a packet to router K?

The result of routing decisions cause routers bo build *forwarding tables* indicating which line to use.

There are two types of algorithms:

- *Adaptive* algorithms use such dynamic information as current topology, load, delay, etc. to select routes.
- In *non-adaptive* algorithms, routes never change once initial routes have been selected. Also called *static* routing.

Obviously, adaptive algorithms are more interesting, as non-adaptive algorithms don't even make an attempt to handle failed links.

Adaptive algorithms can be further divided in the following types:

1. *Isolated*: each router makes its routing decisions using only the *local* information it has on hand. Specifically, routers do not even exchange information with their neighbors.
2. *Centralized*: a centralized node makes all routing decisions. Specifically, the centralized node has access to *global* information.
3. *Distributed*: algorithms that use a combination of local and global information.

Non-Adaptive Algorithms

Flooding is a form of isolated (Tanenbaum says it is static) routing. Does not select a specific route. When a router receives a packet, it sends a copy of the packet out on each line (except the one on which it arrived):

- To prevent packets from looping forever, each router decrements a hop count contained in the packet header. Whenever the hopcount decrements to zero, the router discards the packet.
- To reduce looping even further:
  1. Add a sequence number to each packet's header (set by source router injecting pkt into network)
  2. Each source router maintains a private sequence number. When it sends a new packet, it copies the sequence number into the packet, and increments its private sequence number.
  3. For each source router S, a router:
     - keeps track of the highest sequence number seen from S.
     - whenever it receives a packet from S containing a sequence number lower than the one stored in its table, it discards the packet.
     - otherwise, it updates the entry for S and forwards the packet on

Flooding has several important uses:

1. in military applications, the network must remain robust in the face of (extreme) hostility
2. sending routing updates, because updates can't rely on the correctness of a router's routing table.
3. theoretical—chooses all possible paths, so it chooses the shortest one.

In *selective flooding*, a router sends packets out only on those lines in the general direction of the destination. That is, don't send packets out on lines that clearly lead in the wrong direction.

In *random walk* a router selects one or more lines at random.

- equalize load (on network)
- robust

Link Metrics

If we treat the network as a graph with routers as nodes then what is metric used for links between nodes? Possibilities:

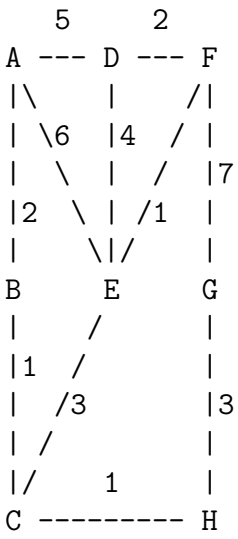
- hops (all links are equal in weight)
- average queue length of outgoing router—only appropriate if delay and capacity are the same for all links. Used in original ARPANET.
- delay (time to send a packet from one router to the next)
- capacity
- ARPANET updated algorithm. Combination to take into account queueing, capacity and delay

  ```
  Router Delay = (LeaveQueue - EnterQueue) + Transmission Time + Latency
                   queueing                      capacity            delay
  ```

  Used known values for capacity and delay then used queueing time to estimate processing time. Good under low load. Variable under high load as heavy queueing delays caused traffic to be routed elsewhere.
- revised ARPANET routing metric: same idea, but compressed metrics so less variation in metrics between low and high loads

Shortest Path Algorithm

What if we "know" the complete topology of the network? Can look at computing the optimal path. What if we have the following network and we want to route a packet from node A to node G. What is the shortest path (do not initially show distance).



Use Dijkstra's algorithm (or variation) treating the network as a connected graph. Basic idea:

- Choose the source, and put nodes connected to source in list to consider.
- From the list to consider choose the nearest node.

Algorithm results:

```
(node, via, total distance)
nodes                    list to consider
-----                    ----------------
(A, -, 0)                B, D, E
(B, A, 2)                C, D, E
(C, B, 3)                D, E, H
(H, C, 4)                D, E, G
(D, A, 5)                E, F, G
(E, C, 6)                G, F
(G, H, 7)                F
(F, E, 7)                -
```

Guaranteed to get the shortest path? How to prove? If an alternate shorter path to a node then we would have already tried the path.

Static Routing (Multi-path Routing)

May not be a single *best* route between two nodes. Can use a routing table. Can compute alternate routes between source and destination by first computing shortest path and then removing these links.

Problem is that these routing tables are initially created and not changed.

Adaptive Routing

Centralized—use a *routing control center (RCC)*. Creates, modifies, and distributes routing tables to other routers. Gathers information from the routers.

Good: adaptive routing, relieve burden on the routers of computing tables.

Problems:

- Does not adapt quickly.
- Quicker the adaption, the more overhead it causes.
- Synchronization of updates (some routers change, but not others, send at each other).
- If RCC crashes the network becomes stale.
- Lines near the RCC are overloaded.

Decentralized Routing

Isolated Routing

Base decisions on local traffic and conditions.

- hot potato—choose output line with the shortest queue
- backward learning—each packet contains source address and number of hops so far. Use this information to learn shortest path to each source. Will learn shortest path to all routers. What is wrong? Only deal with good news, not bad. "Good" may no longer be good due to down router or congestion.

  Must periodically forget and start over (with suboptimal performance after a purge).

Delta Routing

Combine centralized and isolated routing.

RCC (routing control center) computes optimal paths for each router as before. Has a parameter δ to decide if two lines are equivalent. Are equivalent if the delay is within δ of each other.

$\delta \to 0$, centralized
$\delta \to \infty$, isolated, up to the routers.

Modern networks use two dynamic (adaptive) algorithms—distance vector routing and link state routing.

Distance Vector Routing

Distributed routing algorithm (Old Arpanet Routing Algorithm). routers work together.

1. Each router maintains a table (vector) giving the best known distance to a destination and the line to use for sending there. Tables are updated by exchanging information with neighbors.
2. Each router knows the distance (cost) of reaching its neighbors (e.g. send echo requests).
3. routers periodically exchange routing tables with each of their neighbors.
4. Upon receipt of an update, for each destination in its table, a router:
   (a) compares the metric in its local table with the metric in the neighbor's table plus the cost of reaching that neighbor
   (b) if the path via the neighbor has a lower cost, the router updates its local table to forward packets to the neighbor

This algorithm was used in the original ARPANET. Unfortunately, it suffers from the problem: good news travels quickly, bad news travels slowly (count-to-infinity problem).

The fundamental problem with the old Arpanet algorithm is that it continues to use "old" information that is invalid, even after newer information becomes available.

Link State Routing

The "old" Arpanet routing algorithm was replaced in 1979. Problems with old algorithm included:

1. High-priority routing update packets were large, adversely affecting traffic.
2. Network was too slow in adapting to congestion, too fast to react to minor changes.
3. Average queue length was used to estimate delay. This works only if all lines have the same capacity and propagation delay, and doesn't take into account that packets have varying sizes.

In the new algorithm:

1. Each router maintains a database describing the topology and link delays between each router. That is, each router keeps track of the full graph of links and nodes.
2. Each router periodically discovers it neighbors (sends "hello" message on booting) and measures the delays across its links (echo requests—should load be taken into account?), then forwards that information to all other routers (link state packets)
3. Updates are propagated at high priority using flooding. Updates contain sequence numbers, and a router forwards "new" copies of the packet.

   Why use flooding? Because that way routing updates propagate even when routing tables aren't quite correct. ACKs are sent to neighbors.
4. Each router uses an SPF algorithm to calculate shortest paths based on the current values in its database.
5. Because each router makes its calculation using the same information, better routing decisions are made.

Hierarchical Routing

One of the fundamental issues regarding routing is *scaling*. As a network becomes larger, the amount of information that must be propagated increases, and the routing calculation becomes increasingly expensive. Obviously, there are limits to how big a network can be.

Hierarchical routing is an approach that hides information from far-away nodes, reducing the amount of information a given router needs to perform routing:

- Divide the network into regions, with routers only knowing the details of how to route to other routers in its region. In particular, a router does not know about the internal topology of other regions. *Gateway* is a router that knows about other regions.
- A node in each region is designated as an *entry point*, and the entry point knows how to reach the entry points in all the other regions.
- When traffic flows from A to B, it actually follows the path A–AENTRY–BENTRY–B, where AENTRY and BENTRY are the entry points to the respective regions.

Advantage: Scaling. Each router needs less information (table space) to perform routing.

Disadvantage: Sub optimal routes. The average path length increases because there may be a shorter path that bypasses the entry points, but we don't use it.

Hierarchical routing can be extended to multi-levels.

Example: telephone system:

- Area code identifies a region.
- Area code plus the first three digits identify the central office within a specific region.

Broadcasting

Sending a packet to all destinations simultaneously is known as *broadcasting*. There are several ways to implement broadcasting:

1. In broadcast networks, the implementation is trivial: designate a special address as the "all hosts address".
2. Send a unicast packet to each destination. However, this approach makes poor use of resources.
3. Flood packets to all nodes. Flooding generates many packets and consumes too much bandwidth.
4. Use multi-destination routing: Each packet contains a list (or bitmap) of all destinations, and when a router forwards a packet across two or more lines, it splits the packet and divides the destination addresses accordingly.

   This approach is similar to sending unicast packets, except that we don't send individual copies of each messages. However, the copy operations slow down the ability of a router to process many packets.
5. Use a spanning tree. If the network can be reduced to a tree (e.g., only one path between any two pairs of routers), copy a packet to each line of spanning tree except the one on which it arrived.

   Works only if each router uses the same spanning tree.
6. Reverse Path Forwarding (RPF): Use a sink tree (assume sink/source trees are the same). When a packet arrives from router X, if the packet arrived on a line of the sink tree leading to X, the packet is traveling along the shortest path, so it must be the first copy we've seen. Copy the packet to all outgoing lines.

   If the packet arrives on another line, assume that the packet is a copy — it didn't arrive on the shortest path — and discard it.

   RPF is easy to implement and makes efficient use of bandwidth.

Other Routing

Source Routing—Originally used by USENET: ucbvax!decvax!mcvax!marvin

User had to *know* route to remote host. Used large maps of network topology. Can be done automatically from host information stored locally.

Mobile Hosts—must get traffic to *base host*. Need intermediary agents. Look at Fig 5-19. Base host uses encapsulation (tunneling) to send packet to mobile host. Allows mobile host to retain the same network address while mobile.

Multicast Routing—send to a group of machines. Built-in to LAN technology for a site. Use regular IP (with forwarding and tunneling) between sites. Compute a spanning tree of routers. See Fig 5-21.