Internetworking

Multiple networks are a fact of life:

- Growth. Individual departments in a university buy LANs for their own machines and eventually want to interconnect with other campus LANs.
- Fault isolation, geography, and security. Even when feasible to use one network, an organization can obtain exclusive control over a single local network.

Internetworking deals with the issues of interconnecting multiple networks.

Physical networks can be connected at several levels:

1. *Repeaters* operate at the physical layer, copying signals from one LAN to another. They operate at the bit level, and have no notion of what the bits (or even frames!) mean. Repeaters operating at the physical layer operate at level 1.
2. *Bridges* operate at the data link layer, copying frames from one LAN to another. They perform *store-and-forward packet switching*, but use only level-2 (e.g. frame fields) information. Bridges operate at level 2.
3. *Routers* or *gateways* operate at the network layer. Because they operate at the network layer, they are fully aware of different network technologies, and can overcome such problems as interconnecting different network technologies. Also known as level 3 routers.
4. *Protocol converters* operate at higher levels (e.g., “level 4”). For instance, protocol converters can translate between OSI mail and SMTP (Internet) mail formats.

Bridges

The idea behind *bridges* is to *transparently* interconnect LANs. By transparently, we mean that a host treats all other hosts as if they were attached to the local LAN, when in fact, they may be connected to remote LANs.

Bridges operate in *promiscuous mode*, reading every frame on a LAN, regardless of who it is addressed to. The decision as to where to forward a frame (if at all) is made by looking only at the frame's header. That is, a bridge does not look inside the data portion of the frame (hence, level 2).

Bridges run into immediate difficulties. In particular, frame format and service provided differ for each LAN type:

Maximum frame size: 802.3: 1518 bytes; 802.4: 8191 bytes; 802.5: unlimited. It is simply not possible for an ethernet to carry large frames originating from token ring LANs.

Different operating speeds: Token rings operate at 4 (or 16) Mbps, while 802.3 operates at 10Mbps. Bridges will have to buffer frames. When congested, bridges must discard frames.

Differing semantics: What should the 802.5 "copy" bit mean in bridged environments? The sending host may declare a destination down if the copy bit is not set, but is it correct for the bridge to set it? No, because the bridge may be up, although the actual destination host may not be.

In addition, host software that assumes hosts are on the directly connected LAN may fail. For example, if a destination is actually behind several bridges, the round trip delay to the destination may be higher than the sender assumes, and the sender may time out and retransmit too soon.

Priorities: How should one interpret the priorities of frames on one type of LAN when forwarding across another? CSMA/CD LANs do not support priorities, while priorities mean different things for token ring and token bus networks.

Conclusion: Bridging arbitrary networks is infeasible in the general case. In particular, the information needed to handle the above issues properly are not available at level 2. Indeed, that is why there is a network layer in the first place.

Note, however, that some vendors sell such products anyway, ignoring the difficulties.

There are two types of bridges on the market today: *spanning tree bridges* and *source routing bridges.*

Spanning Tree Bridges

Spanning tree bridges were designed with *transparency* as a primary goal. A customer should be able to buy a bridge, insert it between two networks, and have everything work correctly with no hardware, software, or configuration changes on either hosts or existing bridges. How do they work?

1. Each bridge maintains a table that maps destination addresses to the outgoing interface. (Analogous to routing tables in routers.)
2. Bridge operates in promiscuous mode, reading every frame on each of its connected LANs, and the routing decision is made as follows:
   (a) Extract the source and destination address from the frame, and find the corresponding table entries for each address.
   (b) If the two table entries point to the same interface, discard the frame. Why? If both pointers point to the same interface, both the sender and recipient are on the same local network (as far as we can tell), and the frame doesn't need to be forwarded.
   (c) Otherwise, if the two pointers are different, send the frame out on the LAN given by the routing entry for the destination address.
   (d) If the destination is not in the table, flood the frame on all interfaces (except the one on which it arrived). We don't know where the destination is, so let's be conservative and send it everywhere. That way we can be sure that the packet traverses the LAN on which the destination resides.
3. Bridges use backward learning to build tables. They determine which LAN to use to reach destination X by recording the interface on which frames having source address X arrive on.
4. The table is a cache; unreferenced entries are periodically flushed, allowing machines to be moved from one LAN to another.

Use of Spanning Tree Bridges

The above approach works only for tree-structured topologies. Why? Frames will loop forever (there is no time-to-live field in LAN frames) if there are multiple distinct paths between any two bridges. To handle arbitrary topologies, bridges use a special protocol to build a spanning tree:

1. Bridges that are not part of the spanning tree are unused. That is, they are specifically excluded from the tree and do not forward packets. They are available for backup, however, should one of the other bridges or LANs fail.
2. Bridges periodically rebuild tables. They regularly exchange topology information, allowing them to detect the failure of a bridge or LAN. When a bridge or link that is part of the spanning fails, a new spanning tree is constructed.

Advantages: Easy to use. Just install the bridges. No software changes are needed hosts.

Disadvantages:

1. Does not support multipath routing. By definition, only the bridges that belong to the spanning tree are used.
2. The path between any two hosts may not be the optimal path. An optimal path may traverse a bridge that is not part of the spanning tree and cannot be used.
3. Broadcast and multicast frames must be flooded in all cases.

Spanning tree bridges have become extremely popular in CSMA/CD networks.

Vendors also offer bridges that connect LANs separated by fiber or phone links.

Source Routing Bridges

Source routing bridges take a completely opposite approach from spanning tree bridges:

1. They are not transparent. Hosts treat frames sent locally differently from those sent through bridges. Conceptually, the sending host specifies a road map saying which bridges the frame must go through to reach its destination.
2. Each LAN is assigned a 16-bit LAN number, and each bridge on a LAN is assigned a 4-bit bridge number. The numbers must be unique and are set by the network administrator.
3. Each frame carries a *source route* listing the path the frame is to take. The path consists of a sequence of [LAN number, bridge number] pairs.
4. Sending hosts (rather than bridges) responsible chooses the source route. Host selects paths by broadcasting (flooding) special *discovery frames.* A discovery frame includes space for each bridge to add its number to the recorded path.
5. Eventually, a discovery frame reaches the destination host, which returns it to the sender. Each returned message contains a viable path, and the sending host chooses the shortest one.
6. How many frames are generated? Unfortunately, the discovery process leads to *frame explosion.* The destination may receive an exponential number of copies of the original frame.

Use of Source Routing Bridges

Advantages: uses the optimal route. Also can make use of multiple paths to same destination. Because paths aren't required to always lie along the spanning tree, better use of resources.

Disadvantages:

1. Not transparent to hosts; hosts must participate in source routing. This is a significant disadvantage.
2. Installing new bridges non-trivial. Specifically, a system administrator must assign LAN numbers and bridge numbers. Improper configuration leads to disaster.
3. Each host must detect bridge failure on its own (e.g., using timeouts). With spanning tree bridges, the bridges hold that responsibility, and once they have reconfigured, all hosts start using the new path at the same time.

Not surprisingly, IBM supports source routing bridges, while DEC supports spanning tree bridges. IBM markets token ring networks, while DEC has always been big on Ethernets.

Gateways/Routers

Can use terms interchangeably or think of routers as within a subnet (same network) versus gateways (between subnets). Text calls gateways multi-protocol routers.

Gateways are packet switches that operate at the network layer (level 3).

Operating at the network level gives gateways increased flexibility compared to bridges in terms of:

1. Translating addresses between dissimilar networks.
2. Fragmenting large packets for transmission across networks that carry only small maximum packet lengths.
3. Selecting an appropriate path through the subnet.
4. Enforcing policies (e.g., don't forward any local packets off of this network).

Because gateways do more work than bridges, they generally run slower than bridges.

Gateway Ownership

One issue that arises with gateways is who owns them. Typically, bridges connect LANs of one organization, and the issue does not arise there. The ownership question is important because someone has to be responsible for the gateway's operation and dual ownership frequently leads to finger pointing when something goes wrong.

One solution is to use *half gateways*. If two countries are involved, for instance, each country owns its half of the gateway, with a wire separating the two. A special protocol operates over the wire, and each half of the gateway is responsible for implementing the protocol.

For example, the CCITT X.75 standard is used to connect half gateways in connection-oriented networks.

Connection vs. Connectionless

Internetworking in a connection-oriented environment operates essentially as in the single network case:

1. The sending host opens a virtual circuit, but a circuit goes through gateway hops.
2. Any two neighboring gateways at the internetworking level must be connected to a common network.
3. Regular router-based virtual circuits connect neighboring gateways on the same physical network).
4. The end-to-end virtual circuit is a concatenation of individual virtual circuits through each of the networks along the path.

Connectionless internets operate just as connectionless networks. A host sends a packet to a neighboring gateway, which forwards it the next gateway, and so forth. Just as with connectionless networks, gateways make only a *best-effort* attempt at delivering the packet.

When a gateway receives a packet, it selects the interface to send the packet out on and encapsulates the packet using the local data link layer format. As a packet moves from gateway to gateway, it is repeatedly encapsulated and unencapsulated as it travels across each network.

Special case is is *tunneling* between same type of networks across a foreign network(s).

Fragmentation

How does one transmit packets across networks whose *maximum transmission unit* (MTU) is smaller than the packet being transmitted:

1. Connection-oriented internets avoid this problem. How? By selecting a maximum packet size at connection set up time. Once the connection is established, the path never changes, so the sender can select a packet size and never again worry that it will be too large.
2. In connectionless internets, the appropriate packet size depends on the path used. Thus, it can change at any time.

In the general case, setting a minimum MTU for all networks is impractical. Why? A minimum MTU would have to be small, yet sending larger packets should be encouraged for efficiency reasons. Solutions:

1. Have gateway drop packets that are too large to send across a network and return an error message to the sender. The sending host could then retransmit the data in a smaller packet.
2. Have gateway *fragment* large packets into several *fragments*, each small enough to traverse the network.

Transparent Fragmentation

With *transparent fragmentation* end hosts (sender and receiver) are unaware that fragmentation has taken place. A gateway fragments a packet, and the next-hop gateway on the same network reassembles the fragments back into the original packet. Drawbacks?

1. All fragments must travel through the same gateway. Why? So they can be reassembled by the next-hop gateway.
2. Gateways must be careful to avoid reassembly lockup. (The deadlock problem discussed earlier, where a gateway has used up all of its buffer space to hold fragments and can no longer accept new ones).
3. Reassembling fragments uses precious gateway resources that could otherwise be used forwarding packets).

Another approach is to have gateways fragment packets, while hosts perform reassembly (if needed). However, now every host must be prepared to do reassembly.

Problems associated with fragmenting:

1. Fragmenting increases waste: the sum of the bits of the individual fragments exceeds the number of bits in the original message.
2. Loss of a single fragment requires an end-to-end retransmission. That is, the loss of a single fragment has the same effect as losing the entire packet.
3. More work to forward three small packets than one large one. The cost of forwarding packets includes a fixed per-packet cost, that includes doing the route lookup, fielding interrupts, etc.

Firewalls

Require all network traffic to/from organization to go through a single point (firewall).

Switching Architectures

Gateways (and bridges) are usually special-purpose or dedicated machines because they must switch many packets.

Packet switches quickly become CPU-bound. The need to process packets quickly leads to interesting operating system design issues: minimizing data copying, minimizing interrupt processing and context-switch time, etc.

Leads to interesting architecture design issues: dual-ported memory, multiprocessors, etc.