1 GSW… Bridging and Switching

Sandwiched between the physical and media access layers of local area networking (such as Ethernet) and the routeing of the Internet layer of the IP protocol, lies the thorny subject of bridges. Bridges are considered part of the data-link layer in the OSI model, so they speak the same language as the LAN: they can read and interpret the Ethernet header, for example.

As you might imagine, bridges tend to work fine provided that the two LANs they are trying to connect together speak the same language (for example Ethernet). On the other hand, try and connect two LANs using different data-link layer protocols together (perhaps an Ethernet with a Token Ring network) with a bridge, and life suddenly gets very complicated. The bridge has to speak two different data-link layer languages and translate between them. The simplest thing to do in this situation is not to use a bridge: if you want to connect two different types of LAN together, use a router.

However, provided everyone is using an Ethernet variant, LAN bridges can be useful, they don't cause too many problems, and they are the basis of the *switch*, a very popular type of intelligent hub.

1.1 Why Bridge?

There are several reasons why adding bridges to a network might be a good idea:

- The network would be too big without it (either geographically or in terms of the number of stations);
- The capacity of the network can be increased;
- The reliability of the entire network can be increased: now any physical layer problems (for example breaks in a bus cable) doesn't stop the entire network working, but only those stations on the segment on one side of the bridge;
- Bridges can be programmed not to forward frames from certain stations to certain other stations, so financial traffic doesn't go past the engineer workstations, enhancing security.

For example, consider four stations on an unbridged Ethernet bus:

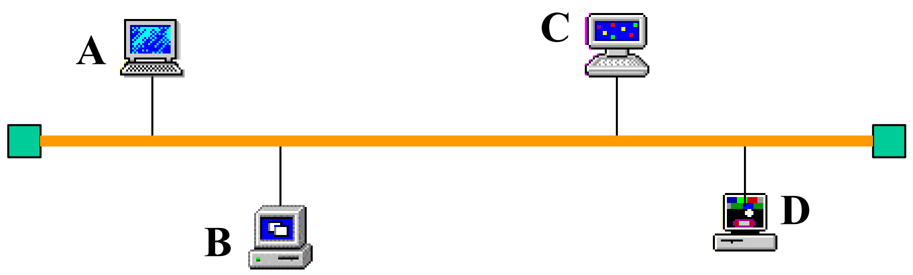


Figure 1-1 An Unbridged Ethernet

Suppose that station **A** wishes to talk to station **B**, and station **C** wishes to talk to station **D** at the same time. On the same Ethernet bus, these frames would collide, and both **A** and **C** would have to back off and re-transmit sometime later. If **A** and **C** had a lot of data to send to **B** and **D** respectively, the best that could happen is that each station ends up with about half the available capacity of the Ethernet between them.

Now suppose the network was divided into two, and a bridge placed between them:

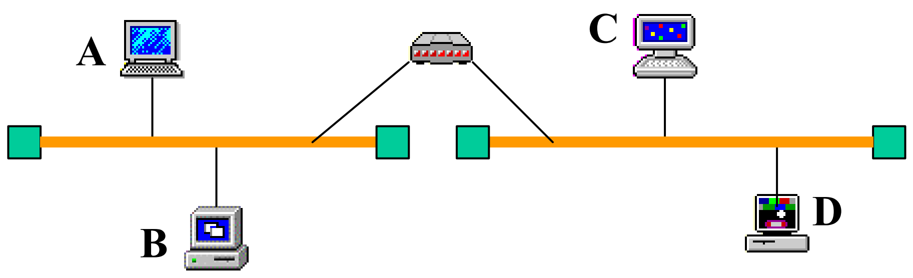


Figure 1-2 An Ethernet with a Local Bridge

From the physical layer's point of view, there are now two networks. Both station **A** and station **C** can talk at the same time, doubling the effective capacity of the network. The bridge only has to forward frames that are intended for stations on the other side of the bridge, and provided there are comparatively few of these, there can be a big increase in the capacity of the overall network.

1.1.1 Switches and Hubs

Probably the most common 'bridges' around now are not called bridges, they're called *switches*. A switch can be thought of as a bridge with multiple ports, and with only one station attached to each port[1].

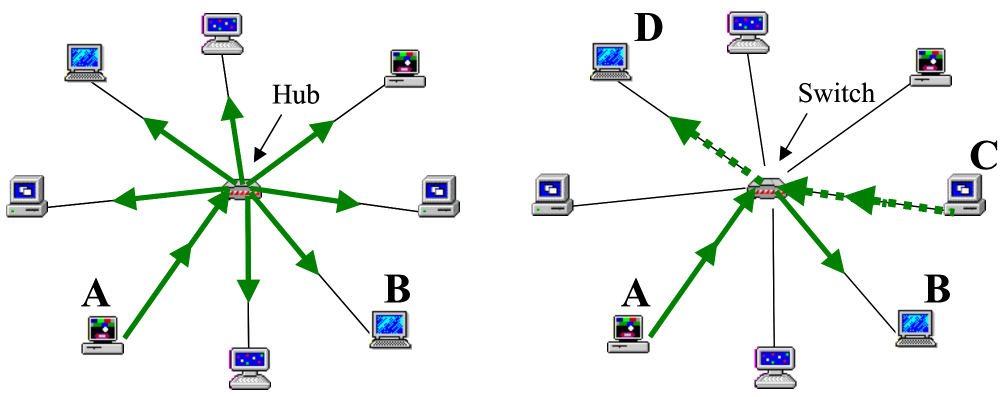


Figure 1-3 The Difference between Hubs and Switches

The main difference between a switch and a hub is illustrated above. A hub will take the incoming frame from any station (in this case station **A**) and send it out to all other stations. This means that the destination (in this case **B**) receives the frame, but so do all the other stations on the LAN. It's not very secure (anyone can eavesdrop on anyone else's conversations) and it doesn't give a very high capacity (only one station can transmit at any one time).

Replace the hub with a switch, and a frame sent from **A** to **B** will be sent by the switch only to station **B**. This means that another frame, from **C** to **D**, can be sent through the switch at the same time. This gives much better security (only the intended recipient receives the frame)

[1] Some devices that call themselves switches can have more than one station attached to each port, but I'll try and keep things simple for now.

and much better capacity (several stations can transmit at the same time, provided no two of them are trying to send a frame to the same destination[2]). The only real disadvantage to using a switch is cost, and these days with the price of VLSI chips dropping continuously, this cost difference isn't that significant anyway.

The switch is acting exactly like a *multiport bridge*. Like most[3] bridges, it needs to know the EUI-48 identifier (formerly known as the MAC address) of all the stations on all of its ports so it knows where to send the incoming packet (remember bridges work at the data-link layer, and in data-link layer language, the addresses are the EUI-48 identifiers).

Although not strictly necessary, most switches have some buffer memory in them, so they can store frames that arrive when the output port to their destination is currently being used. This allows switches (like bridges, but unlike hubs) to have different speeds of networks attached to each port. Most switches can freely combine 10BASE-T and 100 BASE-T networks. (Obviously, if you're receiving a frame from a 10 BASE-T network and forwarding it to a 100 BASE-T network, you have to store the packet and send it on later, you can't just start transmitting the frame as soon as it arrives. That is obvious isn't it? It should be.)

1.1.2 Local and Remote Bridges

Bridges have at least two MAC layers and two PHY layers, one for each port. However, there is no reason why the MACs and PHYs in the bridge have to be in the same physical location. If they do happen to be in the same box (the more usual case), the bridge is known as a *local bridge*. If they are separated by some distance, the bridge is called a *remote bridge*. The two ends of a remote bridge can be on different continents.

It doesn't really matter how the frames get from one port of the bridge to the other (in the case of remote bridges they are often sent over the Internet as the data fields in IP frames: a process known as *tunnelling*) as long as they reach the other end intact and can be transmitted onto the LAN unchanged at the other end.

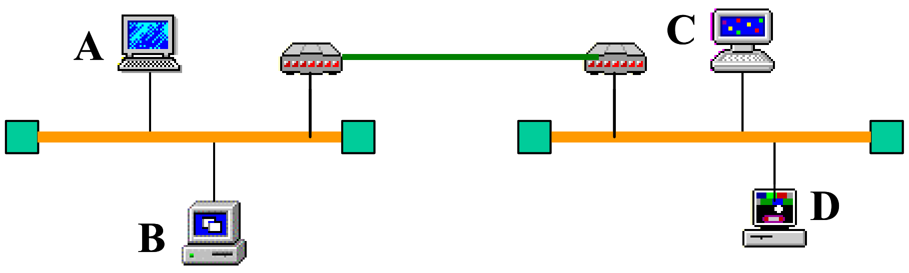


Figure 1-4 An Ethernet with a Remote Bridge

It might help to clarify this idea to consider what the content of the various frames look like at various points on the way from **A** to **D** in the case of a remote bridge. In the figure below, note that the packet leaving the IP layer at the transmitting station is identical to the packet arriving

[2] In fact, even if two stations do try and transmit to the same destination at the same time, this wouldn't cause a collision with most switches: they would just store one of the frames in buffer memory inside the switch and send it out later. Switches are quite clever these days.

[3] But not quite all… see the discussion about source-routeing bridges later.

at the IP layer in the receiving station. The same is true for the Ethernet MAC frame leaving the transmitting station, and the Ethernet MAC frame arriving at the receiving station.

In-between the two halves of the remote bridge, however, the frame looks different. Typically, it will consist of the entire Ethernet frame, put between a header and footer used to carry the frame over the link between both halves of the remote bridge. This is another example of *encapsulation*: putting one frame inside another one so it can be carried across a different network.

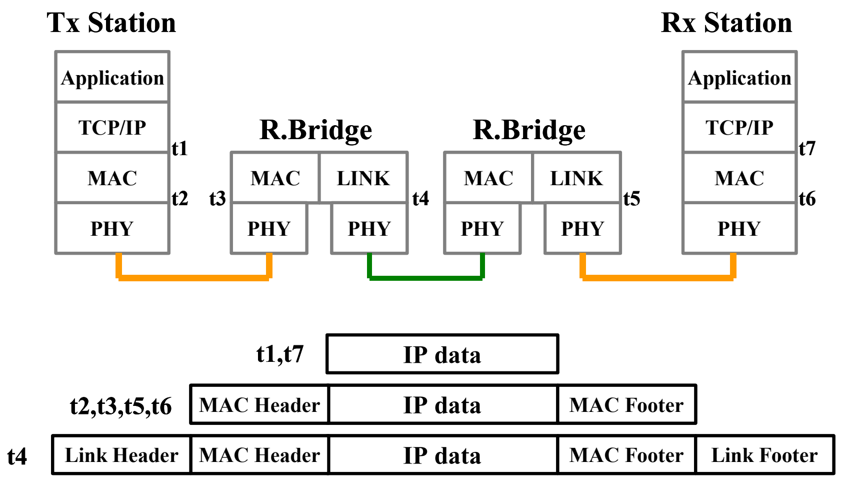


Figure 1-5 Frame Structures with Remote Bridges

1.2 Bridge Tables

Bridges have to receive all frames sent on the LAN, and look at the destination addresses to work out whether they have to forward the frames or not, and if so, out of which port. How do they know what to do?

Clearly, they need to know which stations are on which side of the bridge. There are two ways in which they can find out this information. The first is if someone tells them (usually the network manager, by typing all the addresses of all the computers on the network into the bridge). This gives a lot of security (no-one with another MAC address will be allowed to receive any frames from the bridge), but it's rather time-consuming, and prevents simple mobility (imagine someone with a wireless LAN connection on their laptop – they're not going to want to wait for someone to tell the bridges every time they move from one room to another).

The second (and more usual) method is to get the bridges to work out where the nodes are by themselves, and this they can readily do by looking at the source addresses from the frames that arrive. If a frame arrives on a particular port, the bridge knows that node with that source address must be on that side of the bridge, and from then on, all frames destined for that address should be forwarded out of that port.

All of which is fine, provided there are no loops in the network. If there are, then we've got a serious problem…

1.3 The Problem of Loops

Consider the network below, with three Ethernet segments connected by three bridges. Just after all the bridges are switched on, none of the bridges knows about any of the attached stations, so when bridge 1 receives a frame from station **B** addressed to station **A**, it forwards it, just in case station **A** is on the other side of the bridge (which in this case it is).

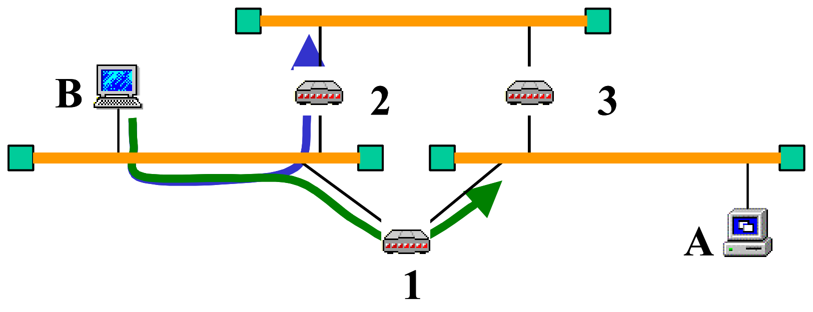


Figure 1-6 Three Bridges in a Loop

However, the frame will also arrive at bridge 2, which also doesn't know where station **A** is, so it forwards it too. Both bridges now think they know where station **B** is, so they make a note that station **B** is on the port from which they received the frame.

The frames carry on, and both of them now arrive at bridge 3. This gives bridge 3 a problem. It receives two copies of the same frame, both arriving from station **B**, but both coming from different directions. How does it decide which side station **B** is on? There's no easy way to do it. Also, what should it do with the frames?

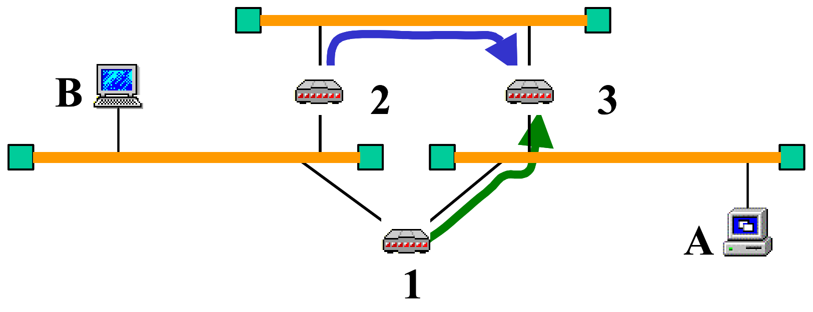


Figure 1-7 Three Bridges in a Loop Getting Confused

Well, since bridge 3 has no idea where station **A** is, all it can do is forward both frames. The net result is two new frames, both appearing to come from station **B**, arriving at bridges 1 and 2 respectively, both from the opposite direction from the last frame from station **B** that arrived at each of these bridges. Both bridges thought they knew where station **B** was… but now? It's very confusing for a poor bridge, this sort of thing.

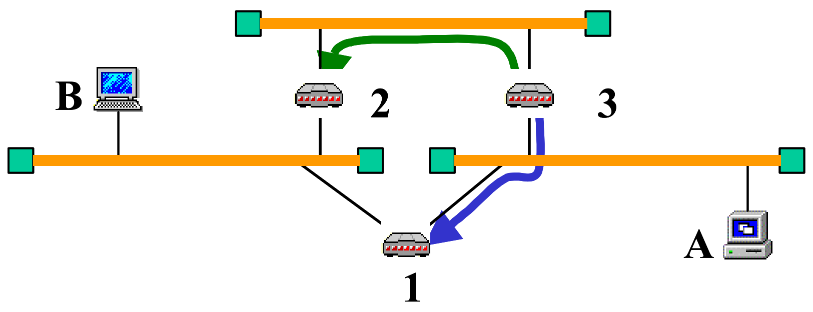


Figure 1-8 Three Bridges in a Loop Getting More Confused

It doesn't stop there. Since none of the bridges know where station **A** is, they continue to forward the frames, and the two frames circle around the loop for ever, consuming all available network bandwidth, and preventing anyone else from sending anything.

There are two obvious solutions to his problem, neither of which work very well. The first is to manually program the bridges to tell them where all the nodes are, and not allow them to forward any frame to anyone else. The second is to add some memory in the bridges, so they remember the last few frames they transmitted, and if they see any of these frames again within a short time, they know not to forward them, effectively breaking the loops.

The reason the first of these solutions doesn't work (apart from the pain of having to type into each bridge a list of all the nodes on the network with their location) is the problem of broadcast frames. Some frames on a LAN are sent to a broadcast MAC address[4] (all '1's) so that all stations can receive them. All bridges have to forward these frames no matter what is programmed into their bridging tables, so at least these frames will circulate forever.

The reason the second solution doesn't work is that sometimes a station might want to send two identical frames in quick succession (for example, if the reliable LLC type 2 was being used, and the first frame arrived corrupted, an identical frame would be resent; you really don't want the bridge to throw away the retransmitted frame).

The solution that does work is simple in theory, but a bit more complex in practice: don't have two loops in a bridged network. This is fine, except that network managers rather like having bridging loops; it adds redundancy to the network. With a loop, if one bridge breaks down, or needs to be replaced, the entire network doesn't have to be stopped, the frames can just get to their destinations by a different route.

1.3.1 The Spanning Tree

To solve this problem, the spanning tree algorithm was invented. This algorithm determines a set of bridges that put all nodes on a spanning tree[5].

---

[4] For example ARP frames.

[5] A spanning tree is a graph (in the graph theory sense: a set of nodes with links connecting them) that connects all of given set of nodes with no loops. In this case the nodes (the circles) are the sub-networks and the links connecting them are bridges. For example:

[continued on next page…]

This 'switches off' redundant bridges[6], until only one route remains connecting any two points in the network, and these troublesome loops never occur. Before explaining how it works, note that each bridge has a unique *identifier* number. In addition, each port on a bridge is given a *port number*, and a *cost*, which is related to the desirability of using that port (in general, the faster the network connected to the port, the lower the cost).

For example, consider the following network:

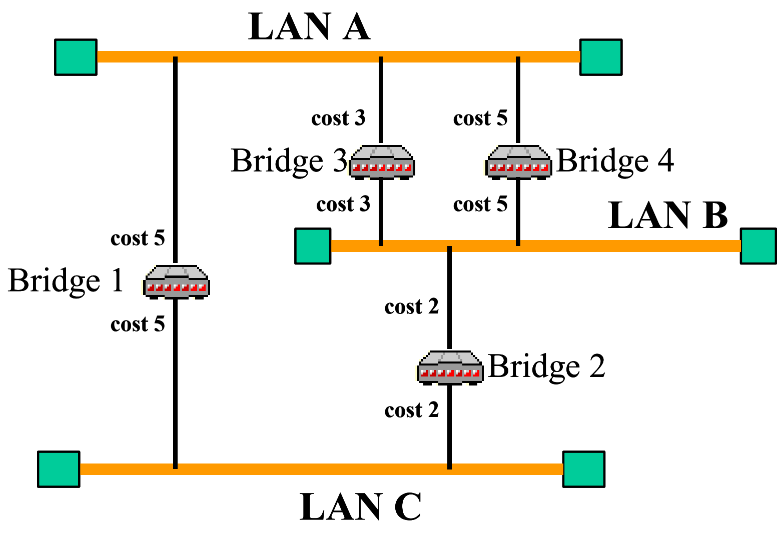
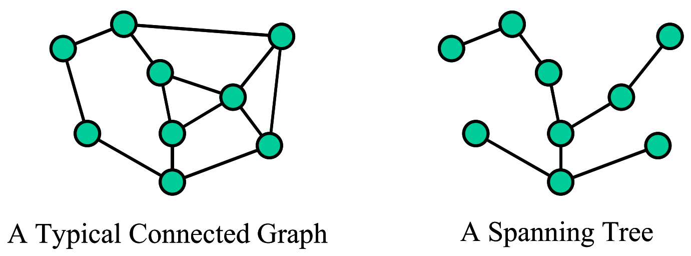


Figure 1-9 A Bridged Network with Loops

The basic operation of the algorithm is that bridges send messages to each other as follows:

- At the start of the algorithm, all bridges send messages on all of their ports telling all other bridges on the same LANs their bridge number, with the cost field set to zero[7].
- All bridges keep a note of the lowest bridge number in any message they receive from any bridges so far.
- Any bridge that receives a message from a bridge with the lowest bridge number they've heard about so far, starts to send messages containing this lowest bridge number (rather than their own bridge number) out from all other ports, with a cost set to the cost to



[6] Well, not literally switches them off, it just stops them forwarding frames.

[7] There is a special group MAC address (01-80-c2-00-00-00) to make this easier. Only bridges receive packets sent to this MAC address.

transmit a frame from this bridge to the bridge with the lower bridge number. (This it can calculate by adding the cost to transmit a frame out from the port to the value of the cost in the incoming message.)

- After a while, all bridges in the network will be sending messages with the same bridge number (the bridge with the lowest number, otherwise known as the *root bridge*), with the cost of transmitting a frame from the bridge to the root bridge.

- If a bridge receives a message quoting the current root bridge number with a lower cost than the cost from the bridge to the root bridge, the bridge stops forwarding frames from that port (since it knows there is another, lower cost route from the network attached to that port to the root bridge via a different bridge).

For example, in the diagram above, bridge 1 has the lowest bridge number, so this will become the root bridge. Once the other bridges have realised this, they'll start sending messages out of their other ports (in this case, the ports connected to LAN B) with the total cost of sending a frame to the root bridge: that's a cost of 2 for Bridge 2, a cost of 3 for bridge 3 and a cost of 5 for bridge 4. These three bridges will receive each other's frames, and bridge 3 and 4 will realise that there is a lower cost route to get from LAN B to the root bridge than they can provide, and they'll promptly stop forwarding frames to/from LAN B. There are now only two active bridges (bridge 1 and bridge 2) and no more loops in the network.

1.3.2 Non-optimal Routeing in the Spanning Tree

While this method removes loops very successfully, it does result in the network being divided into two halves, one on each side of the root bridge; and all frames from one half destined for the other half must pass through the root bridge. This can keep the root bridge very busy (and hence slow), and it's often not the best thing to do. For example, consider the network shown below, where bridge 1 is the root bridge (since it has the lowest bridge number):

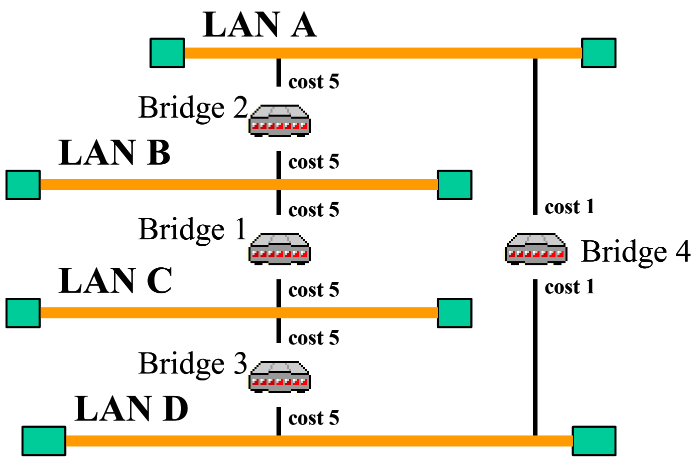


Figure 1-10 Non-Optimum Routeing in Spanning Trees

To send a frame from LAN A to LAN D, it would have to go through bridges 2, 1 and 3. It can't go straight through bridge 4 (much cheaper, with a total cost of 1 rather than 3*5 = 15), since bridge 4 isn't on the spanning tree, and therefore is not allowed to forward frames.

Over the years several techniques have been proposed to allow bridge 4 to be used for these frames in this sort of network, but they add significant complexity to the algorithm, and since large bridged networks are rare (it's usually better to use routers to divide up a larger network[8]) it's not something that's usually a problem in real life.

1.3.3 The Spanning Tree Protocol

The Spanning Tree Protocol (STP) and its replacement the Rapid Spanning Tree Protocol (RSTP) are the protocols that implements this idea on bridged IEEE 802 LANs, with a few modifications to provide greater fault tolerance, speed up convergence after some fault conditions, and prevent frames circulating endlessly round bridging loops while the spanning tree is still being worked out. For full details, refer to the IEEE 802.1D-2004 standard[9].

Spanning tree packets are sent out by the root bridge every 30 seconds or so, and if a few minutes (the exact value is configurable) go past without a bridge receiving a spanning tree frame from the root bridge, a bridge will assume that the root bridge has died, and start the spanning tree process again.

1.4 Transparent and Source-Routeing Bridges

According to the definitions in most of the textbooks, bridges connect LANs at the data link layer, and since the data-link layer does not do any routeing, bridges must be transparent[10]: in other words, from the point of view of the transmitting and receiving stations on either end of the link, they can't tell that there is a bridge between them; they think that they are on the same LAN. Putting this another way: there can be no routeing information in packets that go across a bridge. Put yet another way: two networks connected by a bridge look like two different networks to the physical layer, but only one to the data link layer. We often draw diagrams something like:

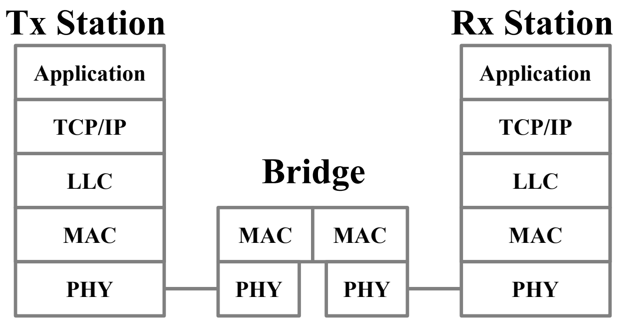


Figure 1-11 Protocol Layer Diagram with Bridge at the MAC Layer

[8] Apart from getting round this problem, there are several other advantages to using routers for larger networks, not least of which is that it cuts down the number of broadcast frames that each station has to spend time dealing with. Broadcast frames go straight through bridges, but not through routers.

[9] Available free from the Get IEEE 802 website, http://standards.ieee.org/getieee802/802.1.html.

[10] The real definition of a bridge is "anything which connects LANs, and which is standardised by a committee set up to standardise bridges". This can be different. (See Radia Perlman's excellent book "Bridging and Routing" for all you could ever want to know about bridges.)

At the risk of over-emphasising the point: a bridge does not understand the language of the LLC; it has no idea what any of the information inside the MAC data field means. Note also that the bridge has two MACs and two PHYs, one on each network that it links. It receives all frames on one network intended for stations on the other network, and re-transmits them on the other network without modification. A frame forwarded by a bridge is identical to the frame as received by that bridge. The bridge is said to be *transparent*.

Or at least that was the original idea…

1.4.1 Source Routeing Bridges

There is another way of doing things. Source routeing isn't really bridging, it's routeing, but it operates at the data-link (MAC) layer rather than the network layer, and it has been standardised by a data-link layer working group, so it is classified as bridging.

A source-routeing frame forwarded by a bridge does look identical to the frame as received by the bridge, so in that sense the bridges are still transparent, they don't modify the frames (unlike routers). Source-routeing bridging is not, however, transparent to the MAC layer protocol in the stations. You can't just add a source-routeing bridge to an Ethernet network and expect things to work with the same protocols running in the stations. You have to modify the data-link layer protocols in the end stations.

Source routeing frames know how to get to their destination. They know about the bridges in the way. There is routeing information in the frames. The frames do not look like frames being sent to other stations on the same LAN. The stations have to know that the destination station is on another LAN. For these reasons, SR-bridging (source-route bridging) violates the textbook definition of bridging.

Source routeing is achieved by adding an additional MAC field into the frame to contain the routeing information. It goes just after the source address, and just before the LLC SAP field. The presence of this field is indicated by setting the 'group/individual' field in the source address[11]. The source routeing field then looks like this:

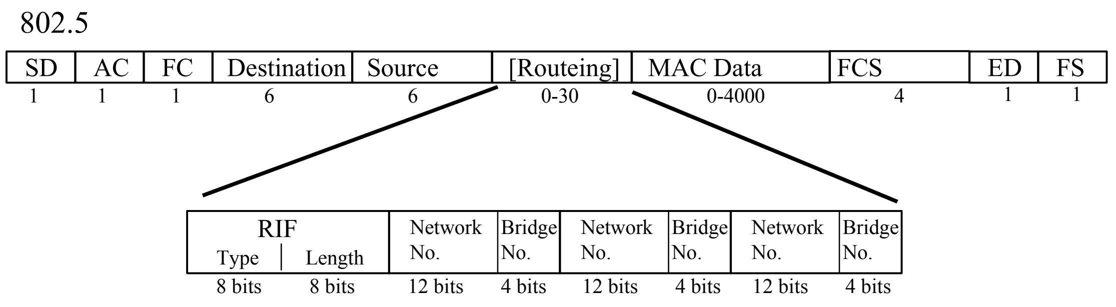


Figure 1-12 Source-Routeing Frame Structure

With source-routeing, each bridge is given a number, and each sub-network is given a number. Each frame is transmitted with a list of which networks and which bridges to go through to get to the destination.

[11] After all, it makes no sense to transmit a frame from a group address, so this bit might as well be used for something useful.

You might wonder how the transmitting station knows how to get to the destination. The answer is it sends short *explorer frames* to find out. If a station wants to transmit a frame to another station, and doesn't know where the destination is, it sends out an explorer frame with a very short source routeing field, consisting of the RIF field only (so that the stations don't need to know the number of the network they are on). On arriving at the bridges on the local network, the network number and bridge number is added to the source-routeing field in the frame, and the now longer frames are forwarded onto the new networks (source-routeing bridges are not transparent to explorer frames).

These explorer frames then spread out throughout the network, being forwarded by every bridge they come across, provided they haven't been through that bridge before (each bridge checks the source-routeing field to see if it has already forwarded the explorer frame; if it has, the frame is not forwarded again; otherwise you'd get the explorer frames going round in loops).

Eventually, the destination station will receive a copy of the explorer frame for each possible route through the network. In networks with a lot of bridges, it might receive a large number of copies of the explorer frame. Usually, it will reply to each one (these replies are not explorer frames, they travel straight back through the same path they came by).

The original station then receives several replies to its initial explorer frame. Typically, it chooses the first one to arrive, notes the route it came by, and stores this route in memory. From then on, all frames sent to this destination will travel by this route.

This method has a few real advantages over transparent bridging. For example, it allows bridges to be put in parallel, or bridging loops to exist with no problems, and no need for the spanning tree algorithm. It removes the need for bridges to keep bridging tables up-to-date with records of which stations are on which side of the bridge. It removes the inefficiencies of having a spanning tree: frames can go by the most direct route from any station to any other station (for example, in Figure 1-10, frames could go direct from LAN A to LAN D using bridge 4). It also allows *load balancing*: put two bridges in parallel between the same two networks, and they can share the load of forwarding frames between them: the less-used bridge will usually forward the explorer frames faster than the busier bridge, and hence be chosen for any new conversation. Finally, the bridges don't have to do a 48-bit look-up of the destination address for every frame that arrives to see if the address appears in their bridging table; all the bridge has to do is look at the source-routeing field to see if it's network and bridge number is there.

No-one ever does source routeing on Ethernet, although it is being considered for wireless LANs, so it's worth knowing a bit about it.

1.5 Key Points

- There are four advantages to bridging: capacity, size, reliability and security. Be able to explain how all of these are increased using bridges.

- Know the difference between a hub and a switch, and the advantages and disadvantages of each.

- Know how transparent bridges work, and how they build up their bridging tables.

- Be able to describe how the spanning tree algorithm works, and why it is needed.

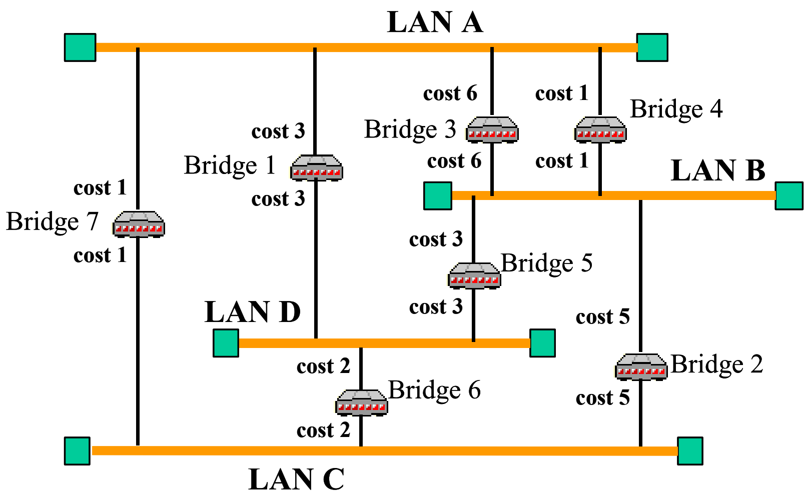
- Be able to work out, given a network diagram with bridge and cost information, what bridges would be on the spanning tree.
- Source-routeing is another form of bridging, which has several advantages over transparent bridging, simplifying the design of the bridges, but increasing the complexity of the stations.

1.6 Tutorial Questions

1) True or false:

a) Two stations talking to each other via a transparent bridge cannot tell whether there is a bridge in-between them or not.
b) When sending a packet to another station on the other side of a bridge, you set the MAC address of the packet to the MAC address of the bridge.
c) Using a hub, it is very easy to 'snoop' on other people's packets.
d) Using a switch, it is impossible to 'snoop' on other people's packets.
e) It is possible to bridge between a 10baseT Ethernet and a gigabit Ethernet network.
f) If a bridge was set up between a 10BaseT Ethernet and a 100BaseT Ethernet, the frame leaving the bridge would be identical to the frame entering the bridge.
g) If a bridge was set up between a 10BaseT Ethernet and a gigabit Ethernet, the frame leaving the bridge would be identical to the frame entering the bridge.
h) You can't have two active bridges forwarding frames in parallel between the same two LANs.

2) The following network contains four LANs and seven bridges. The cost functions associated with each bridge port are also shown. Explain how the spanning-tree algorithm works, and determine which bridges lie on the spanning tree in this case.



Give two examples in this network of routes that packets would take between LANs that are not the lowest possible cost route.

Suppose the cost associated with each bridge port represents the time taken to transmit or receive a packet from that port. Would the network work more efficiently if the bridges were re-numbered?

3) Is it possible to have two transparent bridges between the same two networks if the bridging tables are set up by hand?

**4) How do bridges look-up in their tables to see whether a certain MAC address resides on the other side or not? How could you store and organise the tables to minimise the time taken to do this look-up?