



- IT consultancy • Bespoke network training • Network configuration, performance management and auditing

Bridging and Spanning Tree Operation

Need to understand why your bridges and switches operate the way they do, and what you can do to make your network behave as you want? Fed up with prolonged outages when Spanning Tree doesn't recover from failures as you expect? Wasting network resources by not utilising all your inter-switch links? The paper will explain why it all works the way it does and how you can get back in charge.

Transparent Bridging

If you're wondering why, in this day and age, you need to know much about how bridges work—after all, they were superseded years ago—remember that a LAN switch is basically a multi-port bridge. Switches today offer all sorts of extra features, that the lowly DEC bridge of the 80's couldn't have contemplated, but the basics are the same. If you have a switched network, you need to understand bridging issues first.

Traditional transparent bridges, which were designed to allow a Layer 2 Ethernet environment to be extended geographically (remembering that this was back in the days of thick and thin-wire Ethernet LANs, with their 500m/185m maximum end to end distance limitations), had to be able to handle the fact that end stations would not be aware of their presence within the network. A user device wouldn't be expected to know if its destination was on the other side of a bridge, so it was the responsibility of the bridge to get the traffic to the required destination, regardless of where in the bridged environment it was.

In order to do this, a bridge builds up a MAC-address-to-bridge-port mapping for all the stations it sees on the network. It does this by reading the packets passing through it, looking at the **source MAC address**, and remembering which port it saw that packet originate from. Any packets then destined for that MAC address can be sent directly out of the associated port.

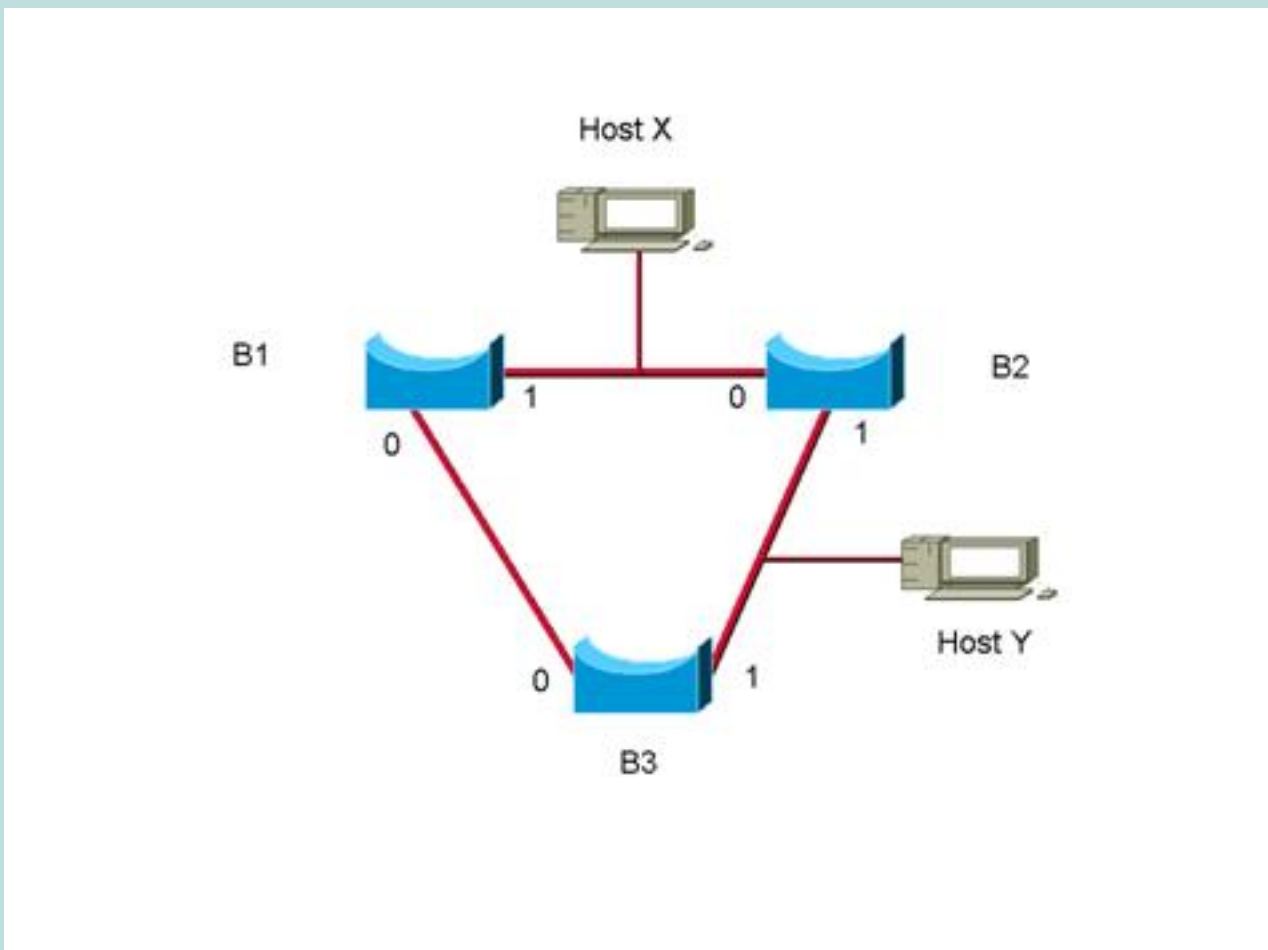
Over time, a bridge would build up a complete picture of where all the devices on a network were (all the

ones that were transmitting anything, anyway), and could forward them out of the correct interfaces. Because the bridge had to be transparent to the network users, if a bridge had not yet learnt where a device was, it would send any packets destined to an unknown address out of every interface (except the one on which the packet had arrived), to make sure it got there—the first bridges only had two ports, so this wasn't too difficult. When it saw the response to that original packet, it could update its tables so that in future it would only need to send the traffic out of the correct port, thus reducing the overall traffic on the LAN. Broadcast traffic, obviously, would have to be forwarded out of all ports (flooded) to ensure it got to all stations.

Bridges reduce the size of an Ethernet collision domain (since any collisions will be kept local to the segment they originate on), and reduce traffic levels. They won't reduce broadcast traffic, though.

Spanning Tree

They also potentially introduce a single point of failure, in that a bridge failure could isolate part of the LAN. So resilience was introduced in bridge design, by building multiple paths between segments. However, a bridged LAN environment cannot handle loops. The problem is illustrated below.



[Fig 1: Spanning Tree Topology](#)

When Host X transmits to Host Y, both bridge B1 and B2 will see the frame. They will update their tables to

show where they believe Host X to be (note: at this time Host Y has not transmitted, so they are unaware of its location):

B1:

Address	Bridge Port
X	1

B2:

Address	Bridge Port
X	0

Both bridges will then forward the frame out of their other interface. In fact it will be received by Host Y at this point, which will (hopefully) reply—in the meantime, the frame from B1 port 0 will reach B3. It will update its table to show how it can reach Host X:

B3:

Address	Bridge Port
X	0

However, it will also receive the forwarded frame from B2—the updated table based on this information would show:

B3:

Address	Bridge Port
X	1

Obviously there is a conflict here. What happens next is worse, however. The frame that B3 received from B1 on its port 0, it will forward out of its port 1 (as it must, since it doesn't know that Host Y has actually now received the packet). The second frame that it received on its port 1, from B2, it must forward out of its port 0. Both bridges B1 and B2 will therefore receive a second copy of the frame they originally forwarded, but

on the wrong port—they will update their tables, with incorrect information, and forward the frame on again. This results in none of the bridges actually knowing where anything is, and duplicate frames flooding the LAN.

The only way, given the constraints on bridges to be transparent to the rest of the network, to prevent this, is to have one of the inter-bridge links be inactive, only coming into use should a failure of a bridge port or link break the path between any two segments.

The algorithm used to dynamically choose the ‘best’ path between bridges taking into account redundant links, is Spanning Tree, developed by Radia Perlman (who not only came up with the algorithm, but wrote a [poem](#) about it too—see her excellent *Interconnections: Bridges, Router, Switches and Internetworking Protocols, Second Edition*, published by Addison-Wesley).

Bridges which run Spanning Tree exchange Bridge Protocol Data Units (BPDUs), containing the information needed to allow them to build a loop-free path between all LAN segments, and reconfigure it when necessary to get round failure points. Within every bridged network, a ‘root bridge’ will be selected: all traffic will be directed towards this bridge by the other bridges on that network. On each LAN, one bridge will be selected as the designated bridge—the one closest to the root bridge according to the algorithm—and it is the only one that can forward data towards the root from that LAN. It will have a ‘root port’, facing the root bridge, and a ‘designated port’ facing the LAN. All other bridges will identify their ‘root port’ as the one which can provide the least cost path towards the root, via the designated bridge, and all other ports on these non-root bridges will be put into a blocking phase, whereby they will continue to receive and process BPDU information from other bridges, but won’t pass any user data

Spanning Tree Algorithm Decision Process

Bridges use a set of parameters to decide which bridge will be root, and which ports on the other bridges to open as root ports, and which to block user traffic on. The root bridge is selected as the bridge which has the **lowest bridge ID**—this is an eight byte number made up of a **bridge priority** and the **48-bit MAC address** of the bridge. The MAC address used will depend on the hardware being used—if we’re talking Cisco here, a switch running Spanning Tree will use an address associated with the backplane or a supervisor module: in an IOS router, it will be a physical address.

The bridge priority value can be between **0 and 65,535**. Again, from a Cisco viewpoint, the default is **32,768**. If you don’t tune your network, then the bridge with the lowest MAC address will become the root. Since this bridge controls the flow of all the traffic in your network, you probably don’t want to leave this selection to chance, in case you end up with the least powerful device, stuck out at the far reaches of your network suddenly becoming the focal point for all your traffic.

The bridges use a **path cost** to determine the path they’ll build between all other bridges and this root. A bridge will receive BPDUs on all of its ports, each detailing the root bridge ID, the path cost to that root and the ID of the neighbouring bridge that sent it the information. From that, it can determine its best (lowest cost) path to the root—if two paths have the same cost, it will choose the one via the neighbour with the lowest ID.

Once a bridge knows which is its root port, it can use that information to see if it has a better path to the root than any of its neighbours, by comparing its root-cost-ID triple with the ones it has received on all its other

interfaces. If this is a better option than any BPDUs it received, it will transmit that value and make itself the designated bridge on that LAN interface: otherwise it will put its interface into a blocking state.

Spanning Tree Configuration

If you want to stay in control of the topology of your layer 2 network, you'll have to decide where you want your root bridge to be, and which paths are to forward and block. You may also be able to tune timers to influence how quickly failures are detected and acted on.

1. Configure the bridge you want to be root to have the numerically lowest bridge priority. Configure a backup root with a slightly higher priority.

Hint: Cisco configuration

To configure a switch to be root regardless of other priorities already set on other switches, use the set spantree root and set spantree root secondary (CatOS) or spanning-tree vlan x root primary/secondary (IOS) commands. Default is 32768. Alternatively you can explicitly set priorities with the set spantree priority or spanning-tree vlan x priority commands.

2. Configure port priorities so that ports you want to be designated have lowest priorities.

Hint: Cisco configuration

To configure port priorities use the set spantree portpri and set spantree portvlanpri (CatOS) or spanning-tree port-priority (IOS) commands. Default is 32/128 (model dependent)

3. Configure path costs so that preferred (higher speed) links have lower costs and are more likely to be used.

Hint: Cisco configuration

By default paths have costs related to their interface speed, eg Ethernet 100, Fast Ethernet 10, GigE 4. To alter these, use the set spantree portcost and set spantree portvlancost (CatOS) or spanning-tree cost (IOS) commands.

4. Configure timers. The Hello time, Forward delay time and Maximum age time can be tuned to reduce reconvergence outages, but be aware that over-enthusiastic shortening of these timers can lead to increased overhead and potential for instability.

The Hello time is the interval between generation of BPDUs by the root bridge. Receipt of a BPDU resets a timer within each bridge—if the Maximum age value is reached before a new BPDU is received, the bridge will assume it has lost contact with the bridge and recalculate its topology. Recommended values are 2 seconds and 20 seconds. The Forward delay timer is the time a bridge will wait for the rest of the network to converge before it starts to send data over a link—the time between moving a port from a listening to a learning state and then learning to forwarding. The recommended value is 15 seconds, which means that at least 30 seconds must pass before a port begins to transmit after a topology change. Reducing this will reduce outage times but risks temporary loops being formed as Spanning Tree converges throughout the network.

Hint: Cisco configuration

Default values are Hello 2 seconds, Max age 20 seconds and Forward delay 15 seconds. To alter these use the following (note that it is recommended that any changes to the Hello setting are made using the diameter parameter in the set spantree root / spanning-tree vlan x root primary commands).

CatOS: set spantree hello, set spantree fwddelay, set spantree maxage

IOS: spanning-tree vlan x hello-time, spanning-tree vlan x forward-time, spanning-tree vlan x max-age

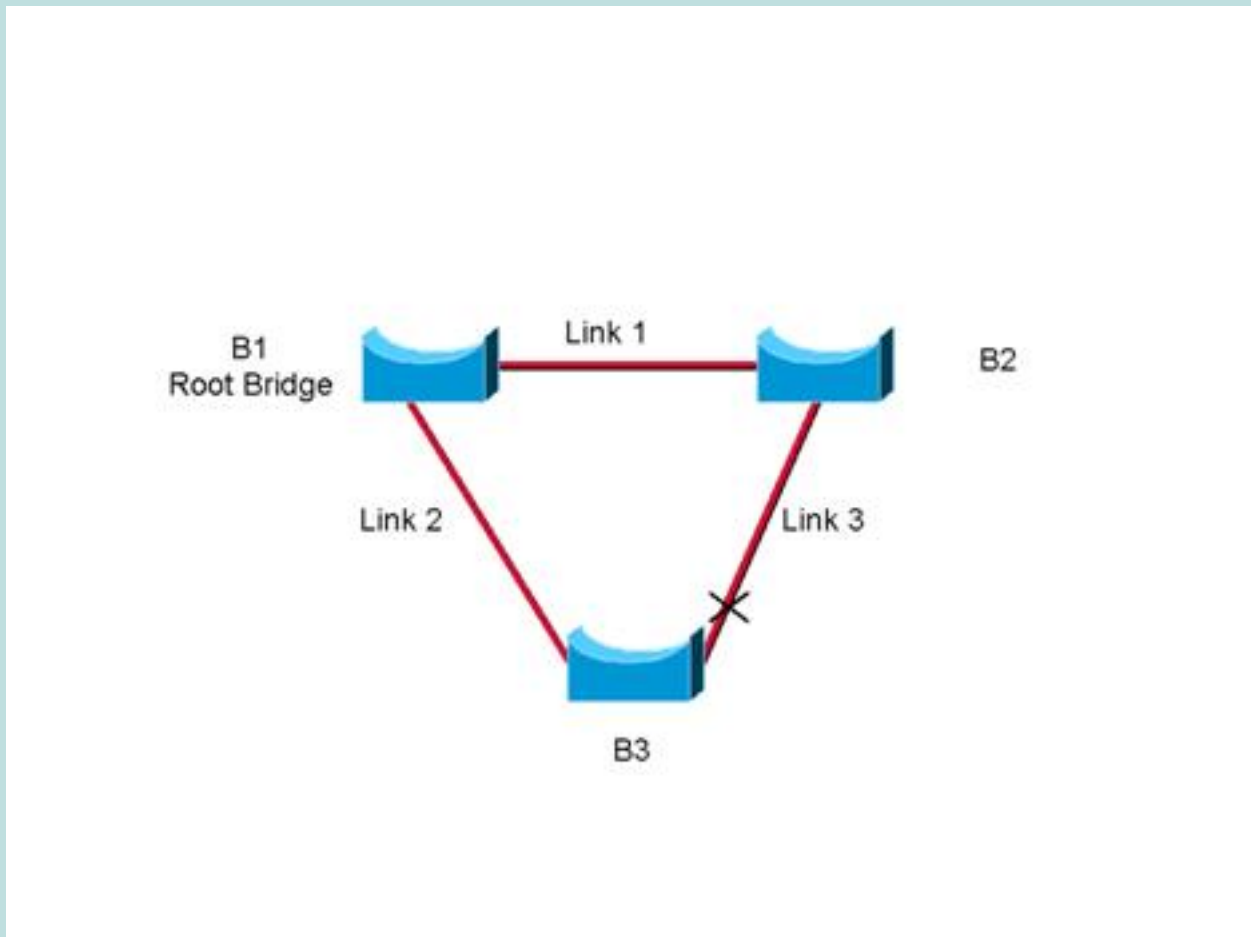
Spanning Tree enhancements

The process for all bridges to learn the state of the network and decide which will be root and which ports will block/forward takes time—typically about 30 seconds, if you don't tweak things, and even if you do, you're still dealing with several seconds of decision-making, during which no user traffic will be forwarded. The days where this is acceptable are past and initially vendors got round this using proprietary mechanisms to short-cut the process.

Cisco-specific mechanisms

On the basis that an access port, which will always and only ever connect to a user workstation, can't form a loop in the network, Cisco introduced the **PortFast** command to allow bridge/switch ports to bypass most of the Spanning Tree learning process and go straight to forwarding. If you've ever had widespread occurrences of NetWare users not being able to log in to the network, this was almost guaranteed to be the fix—without PortFast enabled, on boot up, the PC would issue a Get Nearest Server request before the switch port was actually live, and the Novell Login screen would never appear.

Two other proprietary mechanisms to speed things up were **UplinkFast** and **BackboneFast**.



[Fig 2: UplinkFast/BackboneFast topology](#)

With B1 as root, B3 has its root port connected to Link 2, and put its port connected to Link 3 into blocking mode. Should it then sense that its directly connected Link 2 has failed, it will move its blocked port straight to forwarding, without going through the whole Spanning Tree process. This configuration is designed to be used on access/wiring closet switches, rather than backbone or distribution switches. Note that enabling UplinkFast is a global switch command and it changes the bridge priority value so that it is unlikely that this switch could become root. Network recovery should take about 5 seconds with this configured.

BackboneFast is designed to spot indirect link failures. In [Figure 2](#) above, should Link 1 fail, B3 could not immediately sense the failure, but would see inferior BPDUs (those that claim that the sending bridge is both root and designated bridge) coming from B2, which would have lost all connectivity with the root and therefore have made itself root. B3 would then transition its Link 2 port to a forwarding state without waiting for the full Spanning Tree process. Again this is a global command, and must be enabled for all switches in the network to work properly. Network recovery should take about 30 seconds with this configured.

Standards-based Spanning Tree enhancements

802.1w, Rapid Spanning Tree, was developed and approved by the IEEE to speed up the outages caused during network reconvergence by Spanning Tree decision making processes. The number of states a port can be in is reduced to three (discarding, learning and forwarding) and information aging is speeded up to

enable faster failure detection and reconvergence. Working pretty much like the Cisco proprietary mechanisms, 802.1w also has the concept of a user, or edge, port, which does not require to take part in a Spanning Tree learning process—unlike the Cisco variant, should someone mistakenly connect another switch to one of these ports, it will detect the presence of BPDUs and take itself out of this configuration.

Multiple (per-VLAN) Spanning Tree

The original Spanning Tree specification allowed for only one instance of Spanning Tree to run in a layer 2 network, catering for all VLANs. Proprietary options were added by vendors to allow for multiple instances of Spanning Tree, one per VLAN, so that some limited form of load-balancing could be done—with only one Spanning Tree instance, links and ports were left unused, which was wasteful.

Cisco proprietary mechanisms are Per-VLAN Spanning Tree (PVST+), which allowed one instance of Spanning Tree for each VLAN, and then the Multi-Instance Spanning Tree Protocol (MISTP), which allowed VLANs to be grouped, with a Spanning Tree instance per group, rather than per individual VLAN. There is now a standard, 802.1s—Multiple Spanning Tree Protocol (MSTP), which allows Spanning Tree to be divided into regions, each with Internal Spanning Tree instances, joined via a Common Spanning Tree instance.