BRIDGE PROTOCOL SPECIFICATION

by

Nina T. Plotkin

Memorandum No. UCB/ERL M92/98

27 August 1992

BRIDGE PROTOCOL SPECIFICATION

by

Nina T. Plotkin

Memorandum No. UCB/ERL M92/98

27 August 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

BRIDGE PROTOCOL SPECIFICATION

by

Nina T. Plotkin

Memorandum No. UCB/ERL M92/98

27 August 1992

ELECTRONICS RESEARCH LABORATORY

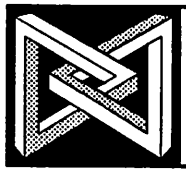
College of Engineering
University of California, Berkeley
94720

Bridge Protocol Specification

Nina T. Plotkin

Abstract

The BayBridge performs three tasks: Learning, Forwarding and Spanning Tree configuration. The implementation of these tasks was designed to be compatible with the IEEE 802.1d Bridge Standard. We describe how the BayBridge Learning and Forwarding techniques are derived from 802.1 concepts, and give some implementation specific details. The Spanning Tree Algorithm is a distributed algorithm and is described in detail. In addition the choice of parameter values for time critical parameters of the Spanning Tree Algorithm are presented.



The Bay Bridge

Project Report: 6 Revision: 1.0
Department of Electrical Engineering
and Computer Sciences
University of California at Berkeley

Contents

1 Introduction

The BayBridge [5] is a dualport remote bridge designed to interconnect FDDI and SMDS networks. The BayBridge has one FDDI port which receives and transmits packets from an FDDI network. The second port on the BayBridge receives and transmits packets from an SMDS network. When the BayBridge receives a packet from the FDDI network, it decides whether or not to forward that packet onto the SMDS network. Similarly, when it receives a packet from the SMDS network, it decides whether or not to forward that packet on the FDDI network.

The BayBridge works in conjunction with local bridges, and other BayBridges, to form a "bridged network". Typically, local bridges connect similar networks (for example, FDDI to FDDI or ethernet or token ring), and remote bridges connect dissimilar networks. The goal of a local bridge is to extend the capabilities and resources of a single LAN. The goal of remote bridges is to further expand these networks by connecting a number of "locally bridged networks" together via a backbone. In this implementation, the SMDS network serves as the backbone network. Moreover, the BayBridge is considered a remote bridge since FDDI and SMDS networks differ in a number of ways. For example, FDDI is a broadcast network and SMDS, from our single CPE point of view, is a point-to-point network. FDDI uses flat addressing, SMDS uses hierarchical addressing. FDDI is a private network, SMDS is a public network. FDDI runs at 100 Mbps and SMDS (with our system) runs at either DS1, DS3 or STS-3 rates.

In order to work together, the bridges themselves must communicate. They communicate by transmitting a specific type of message to one another. This message type is called a Bridge Protocol Data Unit (BPDU). The bridges use these messages to form a spanning tree topology. (In the spanning tree topology we consider bridges as nodes in a tree, and LANs as the links in a tree.) The purpose of this hierarchical structure, superimposed on the network, is to break loops in the network and avoid packet duplication. If, after this topology has been agreed upon and formed, a bridge node becomes disfunctional, or there is a link failure, then the bridges must change the current topology and agree on a new spanning tree. A new tree is formed to find new routes through the bridged network, so that packets can bypass the failed section of the network. The spanning tree algorithm is a distributed algorithm.

In order to perform these functions, a bridge carries out three main tasks: a forwarding process, a learning process, and a spanning tree process. **The forwarding process** makes decisions about whether and how to forward packets. In order to make these decisions, the BayBridge needs routing information. In

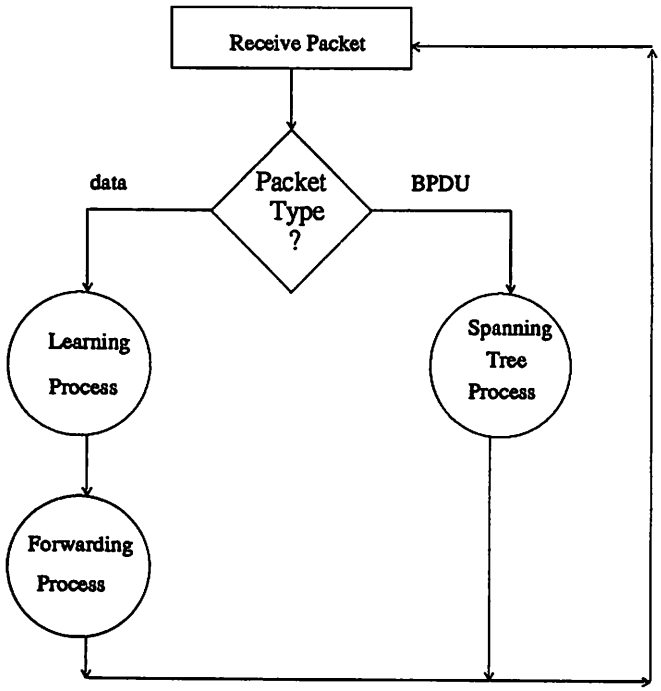


Figure 1: Overview of Bridge Functions

bridges these tables are called filtering tables, since bridges are said to filter packets. (We refrain from using the expression *routing table* as this is typically used by higher layer protocols for more complex store-and-forward routing schemes.) These tables are used to keep directional information as to how to forward packets. More precisely, the table indicates through which port a packet should be forwarded in order to reach its final destination. **The learning process** constructs the filtering tables. **The spanning tree process** takes care of the bridge-to-bridge BPDU communication, forms the spanning tree, detects topology changes and dynamically adjusts the spanning tree according to these topology changes. A simple flowchart showing these basic tasks is given in Figure 1.

In the BayBridge implementation, the learning and forwarding processes will be implemented in hardware, and the spanning tree process will be implemented in software. The learning and forwarding decisions are implemented in hardware in order for the BayBridge to be fast. It will be fast since learning and forwarding decisions are made on a packet-by-packet basis; however spanning tree decisions are made much less frequently. The hardware-software interface required for the bridging protocol is specified in [8]. Our remote bridge is designed according to

the philosophy outlined in the IEEE 802.1d Standard [1]. A few changes were needed (two address filtering tables which contain different information from each other, and support of two methods for computing message ages), since the BayBridge is a remote bridge and not a local bridge. The BayBridges are designed to be compatible with IEEE 802.1d local bridges, and to interoperate with them.

Other protocol conversion functions such as address translation, encapsulation/decapsulation, and segmentation/reassembly are also carried out in the BayBridge. However, these are not considered part of the "bridge protocol", and are not the subject of this document. Section 2 presents the basic bridge learning and forwarding schemes as defined in the IEEE 802.1d Standard. Section 3 describes the BayBridge's implementation of the learning and forwarding functions. Section 4 presents the Spanning Tree Algorithm, and discusses its characteristic performance parameters.

2 General Bridge Learning and Forwarding

In this section, we describe succinctly the basic technique that bridges employ for learning and forwarding, as defined in the IEEE 802.1d Standard. In the next section, we discuss some implementation details of the BayBridge, and then explain how learning and forwarding are implemented on the BayBridge. For example, the implementation is dependent on the fact that the BayBridge is a two-port remote bridge.

2.1 General Bridge Learning

Typically, local multiport bridges have filtering tables that contain the following information:

DESTINATION ADDRESS, NEXT OUTGOING PORT #, AGE TIMER

When a bridge receives a packet, it reads the source address and notes the incoming port on which the packet arrives. This information is then placed in the table. The port number indicates on which side of the bridge a host lies. Thus any packets in the future headed for that host are forwarded on the appropriate port. The Age Timer is used to measure the age of an entry in the table. When table entries exceed a certain age, they are deleted. Only current, active information is maintained in the filtering table. If an entry is already in

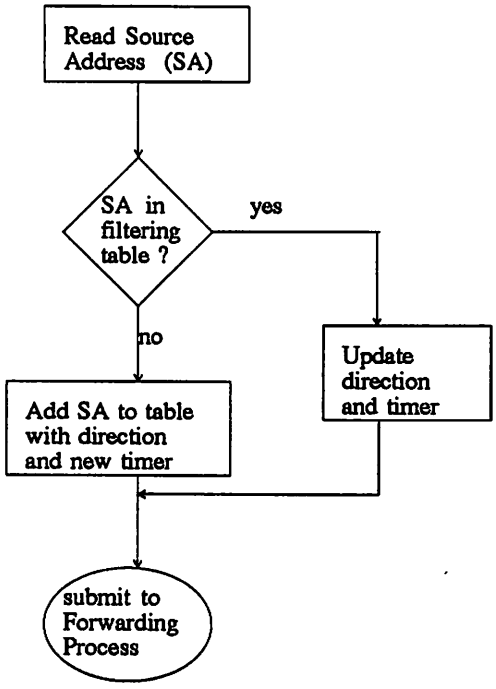


Figure 2: Basic Learning Process

the table, then the learning process updates the port value and the timer. The port value is updated in case the host has changed its location. Hosts can retain their address even if they move from one part of the network to another. If the source address is not in the filtering table, then it is entered into the table. See Figure 2.

Learning is carried out "on-the-fly" and is thus very quick. (This is quite different from query-response learning mechanisms.) By the time the entire packet has been copied to the bridge, the learning process can already be finished. Once it reads the source address, the rest of the learning process can go on in parallel, as the remainder of the packet is copied to the bridge.

2.2 General Bridge Forwarding

When a packet arrives, the bridge reads the destination address and looks it up in the filtering table. If the outgoing port number is different from the incoming port, then the packet is forwarded on the outgoing port. If the port numbers

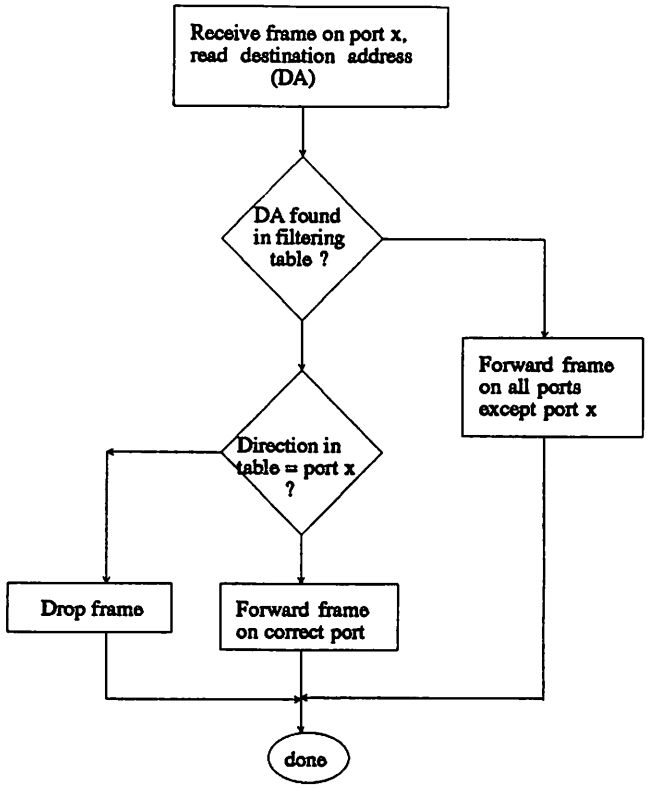


Figure 3: Basic Forwarding Process

are the same, then the packet is discarded; it is already traveling on the correct side of the bridge, and does not need to be duplicated. If there is no entry for the packet in the table, then the bridge sends a copy out on all of its ports except the incoming port. In other words, it floods all of its ports other than the one the packet arrived on. This process is described in Figure 3.

3 BayBridge Learning and Forwarding

3.1 Traffic Types

Figure 4 depicts the different types of packets that flow through the bridge. The Spanning Tree Process on one bridge node transmits and receives BPDUs from other Spanning Tree Processes on different bridges. BPDUs can be received on

either of the BayBridge ports. In this implementation, the BayBridge connects directly to a workstation, which is itself a host. This host can also be the final destination of a data packet. Clearly, the FDDI port receives and transmits FDDI packets, and the SMDS port receives and transmits SMDS packets.

3.2 BayBridge Filtering Tables

The BayBridge will maintain two filtering database tables: a local table and a remote table. In this section, we discuss the rationale behind the information in the tables, and not their specific implementation. The local table is used to store forwarding information about hosts that lie on the FDDI side of the BayBridge. The remote table is used to store forwarding information concerning hosts that lie on the SMDS side of the BayBridge. In other words, the local table stores information about hosts generating traffic type FS, and the remote table stores information about hosts generating traffic type SF (Figure 4).

The IEEE802.1d standard requires the following information to be maintained in the filtering database:

DESTINATION ADDRESS, NEXT OUTGOING PORT #, AGE TIMER.

FDDI hosts have 48 bit addresses derived from their MAC addresses. In the *destination address* field, the local filtering table will store the MAC addresses of the hosts on the FDDI side of the BayBridge. The *next outgoing port #* field will not be needed for the local table. All entries in the local table belong to the FDDI side of the BayBridge (according to our definition of local table). Moreover, there is only one port that connects to the FDDI side (i.e. no choice between multiple ports connected to the locally bridged FDDI networks). Therefore, all entries in the local table would have the same *next outgoing port #*. Hence it is not needed, we can simply associate every entry inthe local table with the single FDDI port. (Another reason this field is not needed is given in Section 3.4 on Forwarding.) The timer field is used to measure the age of the entry. It is set to zero when an entry is first placed in the table. The timer must be updated every second (i.e. has a granularity of 1.0 seconds). After an entry becomes older than the allowed Ageing Time parameter, it is removed from the table. (The Ageing Time parameter is specified in section x.)

It is important to note that the FDDI ring is essentially a broadcast network in the sense that every packet put on the ring will be seen by every station on that ring. In contrast, the SMDS network (when using single CPE access) can be viewed as a point-to-point network. Thus for one BayBridge to transmit a packet to another BayBridge requires knowledge of the specific SMDS address

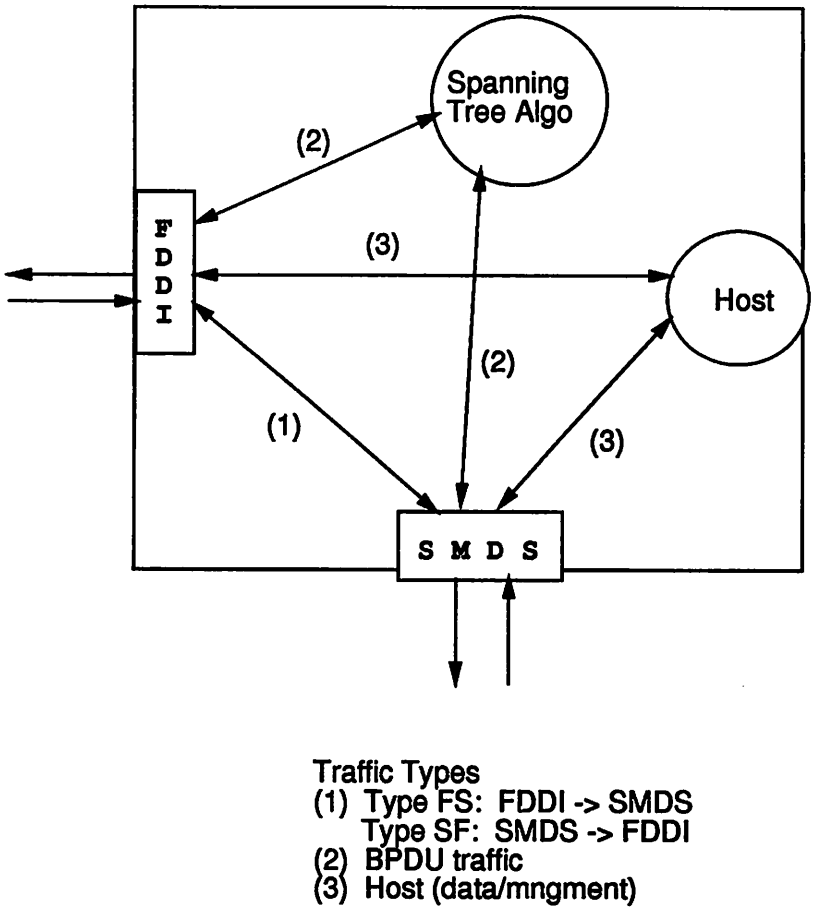


Figure 4: Flow of different traffic types through bridge

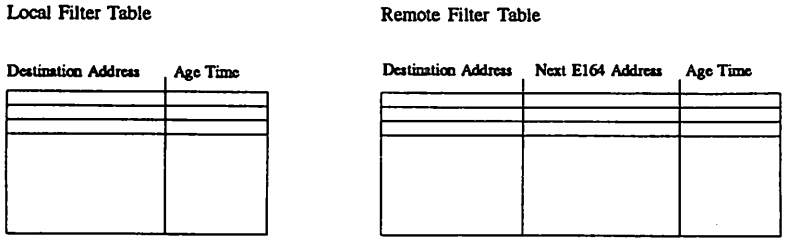


Figure 5: Contents of Filtering Tables

of that BayBridge. The SMDS addressing scheme uses E.164 addressing. In order to send a packet to all other BayBridges, a multicast address must be used.

All of the traffic we transport is generated by and destined for FDDI hosts. Some FDDI packets will have to travel through the SMDS network. Each FDDI packet travelling through the SMDS network will traverse two BayBridges. The link between these two BayBridges is a point-to-point link, and the necessary addressing information must be available in the remote table.

The remote table will store MAC addresses of the hosts on the SMDS side of the BayBridge in the *destination address* field. The *next outgoing port #* will contain the SMDS E164 address of the next remote bridge to which the packet should be forwarded. The timer field will be used in the same fashion that it is used in the local table (described above). Therefore, the two filtering database tables will look as in Figure 5.

FDDI group addresses are not store in either table. Therefore packets with group addresses are always multicasted through SMDS (i.e. "flooded"). This is done in order to allow hosts separated by large distances, or hosts in different bridge clusters, to participate in the same group address.

3.3 BayBridge Learning

When a packet arrives from the FDDI side of the BayBridge (traffic type FS), the learning process reads the MAC address. If the entry is already in the local table then the timer is set to indicate a "fresh" entry. If the entry is not found, then it is added to the local table. When a packet arrives from the SMDS side (traffic type SF), it will be an FDDI packet encapsulated in an SMDS packet (since all packets originate from FDDI hosts). The information learned from traffic type SF is put into the remote filtering table. The FDDI source address must be read and entered as the destination address in the filtering table. The

SMDS source address provides the E164 address required in the remote table. The learning for traffic type SF, requires reading both the FDDI source address and the SMDS source address from the same packet.

3.4 BayBridge Forwarding

When a packet arrives from the FDDI side, first a table lookup is performed on the destination address in the local table. If the destination is found in the local table, then the packet is dropped, because the destination lies on the FDDI side of the BayBridge. If it is not found, then a table lookup in the remote table is performed. If the entry is found, then the packet is forwarded to a specific BayBridge on another side of the SMDS network. If the entry is not found in the remote table, then the packet is multicast to all BayBridges. (This corresponds to the notion of "flooding all ports except the incoming port" in the original bridging algorithm.)

When a packet arrives from the SMDS side, a table lookup is performed in the remote table. If the destination address appears in the remote table, then the packet is disgarded since the destination lies on the SMDS side (same side packet received on) of the Baybridge. This will occur when packets are multicast between BayBridges. If the destination is not found in the remote table, then the packet is automatically forwarded on the FDDI port (without doing a local table lookup). The reason for this is as follows. If this packet is already arriving on the SMDS port, then either the transmitting BayBridge knows that the destination is on the FDDI side of the receiving BayBridge, or the transmitting BayBridge did not find it in any of its filtering tables and has multicast the packet to all BayBridges. In the first case, the packet with destination known to be on the FDDI side, clearly must be forwarded. In the second case, if the destination was not found in the remote table, then the packet must be forwarded for flooding purposes. So if the packet is not found in the remote table, then it will be forwarded on the FDDI side regardless of whether it was singlecast or multicast, and regardless of information in the local table [1]. Flow charts summarizing the BayBridge learning and forwarding techniques are given in Figure 6.

[1] This is the second reason, see Section 3.2, why the *next outgoing port #* field is not needed in the local table.

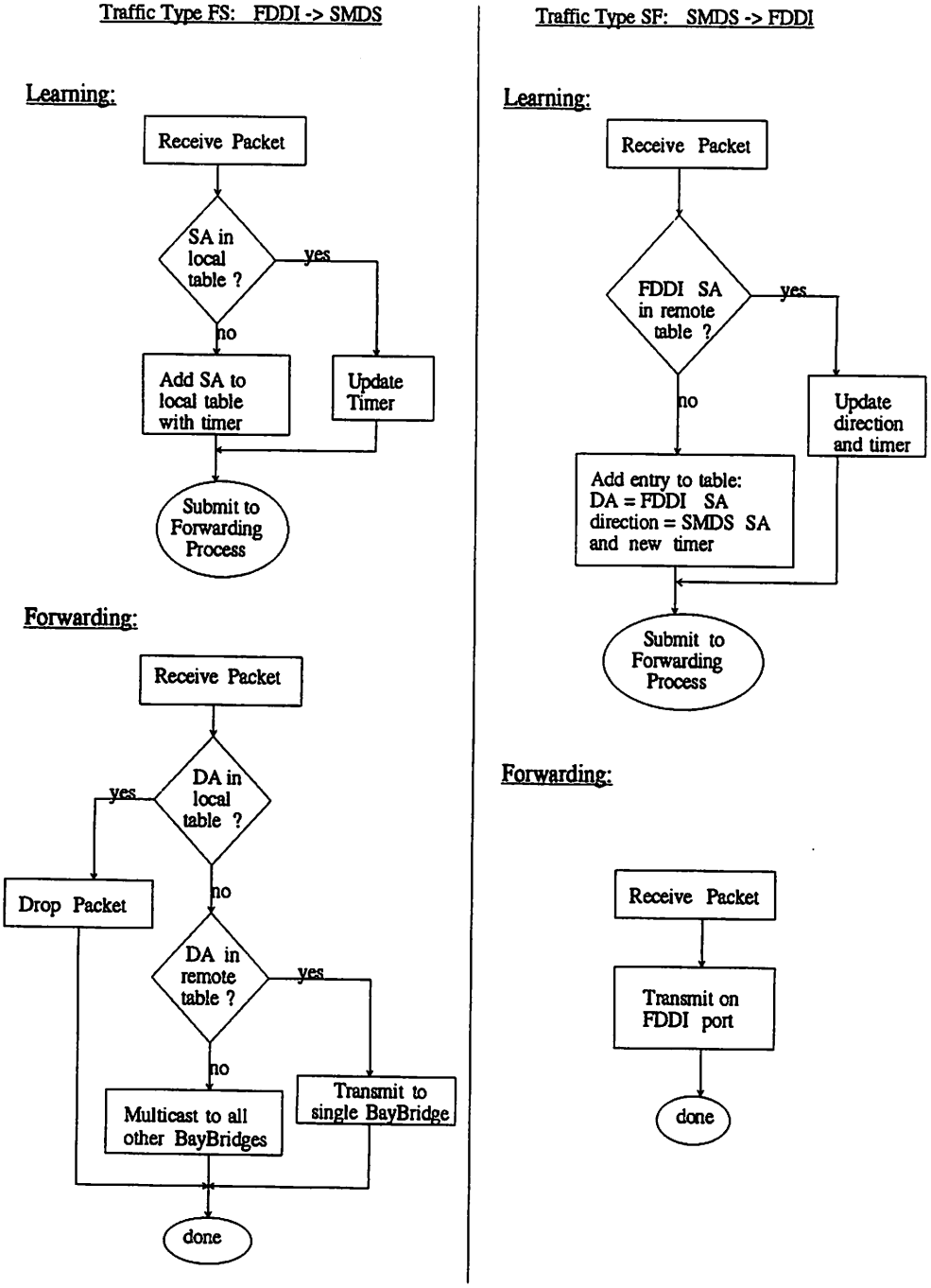


Figure 6: BayBridge Learning and Forwarding

4 Spanning Tree Protocol

4.1 Overview of Basic Operation

The primary function of the Spanning Tree Algorithm is to organize the bridges in a hierarchical fashion such that a there is only one path, of bridges and LANs, between any two LANs in the entire bridged network. In other words, any loops in the network topology must be broken[2]. The forwarding and learning processes described earlier assume that there are no loops in the topology of the bridges and LANs. It is necessary to remove loops in the topology to prevent packet looping and packet duplication. Figure 7 gives an example of how packet looping and duplication can occur. Suppose station X transmits a packet to an unknown destination. Both B1 and B4 will each send a copy to L2, creating a duplicate packet. B2 will forward both of these onto L3, and then B3 will again forward both of these back to L1. This process will repeat duplicating and looping until either, the network fails due to overload, or the destination address is learned by all bridges AND either B1 or B4 is temporarily disabled. (Actually duplication and looping will occur both clockwise and counterclockwise in this example, as the two types of arrows indicate.)

The hierarchical structure chosen to accomplish this task of eliminating loops is a spanning tree. The Spanning Tree Algorithm is used to transform an arbitrary mesh topology into a single, acyclic spanning tree. A bridged network can be viewed as a spanning tree if we consider the bridges to be nodes in an undirected graph, and the LANs to be hyper-edges[3].

The spanning tree algorithm is a distributed algorithm. The bridges communicate by sending a special type of message to each other called a Bridge Protocol Data Unit (BPDU). These messages contain enough information in them for all the bridges to agree on a specific topology, to detect failures when links or bridges malfunction, and to reconfigure the topology accordingly. The terms *configuration process* (or *reconfiguration process*) and *steady state operation* are used throughout the document. *Configuration process* refers to the state of the system while the bridges are in the process of forming a spanning tree. After the configuration process has finished, or converged, then the system is said to be in a *steady state*. During *steady state* operation, no changes in the topology occur.

The first task to be done when computing the spanning tree, is to choose a

[2]By "breaking" loops, we mean temporarily disabling network components such as bridge ports.

[3]Hyper-edges are defined as edges which may connect more than two nodes, but always have a unique parent node.

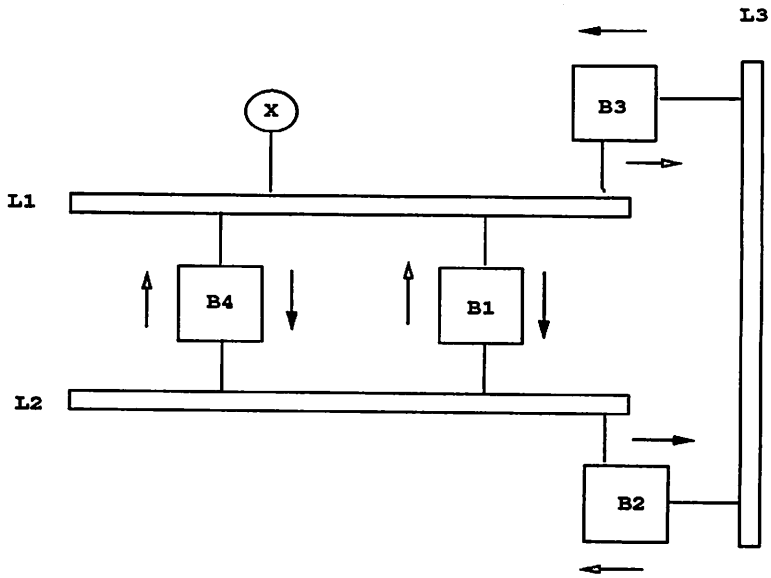


Figure 7: Packet Looping and Duplication

root bridge. Each bridge is assigned a priori a unique bridge ID. The bridge with the lowest ID number has the highest priority. The highest priority bridge will become the root bridge. Once a root bridge has been chosen, each of the remaining bridges must decide which one of its ports lies "in the direction" of the root. This port is called the root port. The root port is the port which offers the lowest cost path to the root. Each port, within a bridge, is also assigned a unique ID number. The bridge IDs are unique within the scope of the entire bridged network, but the port IDs are unique only within the scope of a single bridge. If the path costs through two ports are equal, then the bridge selects the port with the lowest ID (highest priority) to be root port.

A single LAN may have many bridges attached to it, only one of which can become a "designated bridge". Each LAN is assigned one "designated bridge" during the configuration process. The designated bridge is the one which offers the least cost path to the root. Designated bridges will choose one of their ports (different from the root port) to be a "designated port". The designated port is a port which attaches the "designated bridge" to its LAN. Clearly the other bridges attached to this same LAN, which are not "designated bridges", must be informed which bridge is the designated bridge.

Ports that are chosen to be the root port or a designated port are put into a "forwarding" state. The other ports on a bridge, do not participate in the

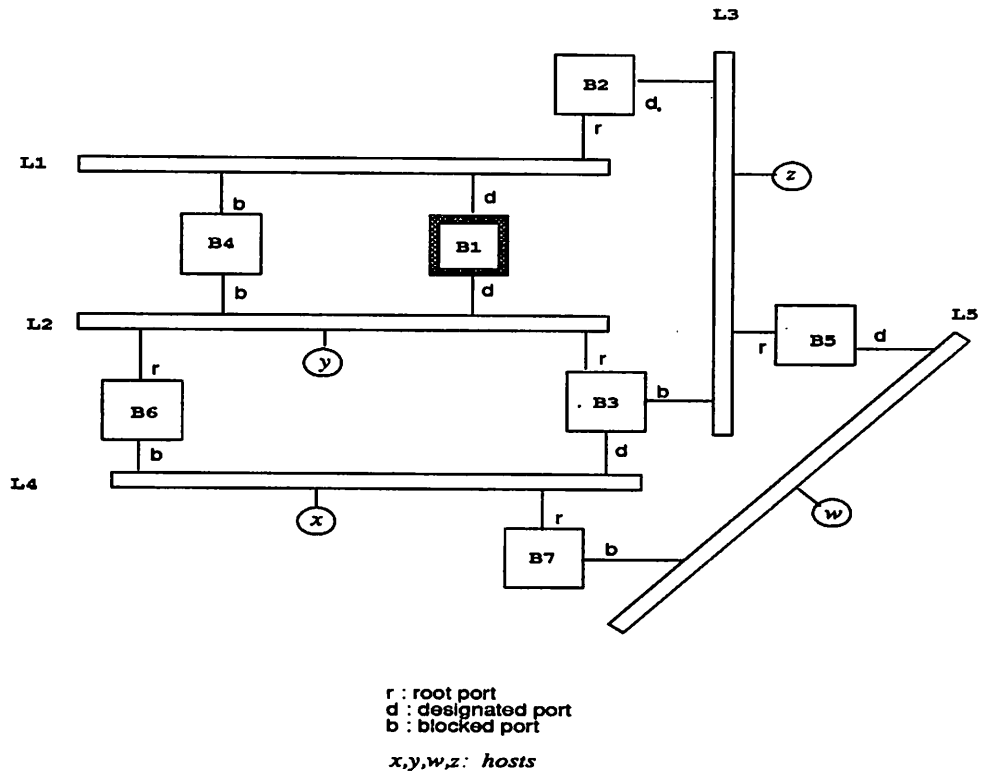


Figure 8: Sample Network after Spanning Tree Algorithm converges

forwarding of data packets, are put in a "blocking" state. There are a total of four possible states a port can be in. The "listening" and "learning" states are used when a port is being transferred from the "blocking" state to the "forwarding" state. This is explained further in Sections 4.2 and 4.5.

Example

Figure 8 shows the state of a sample network after the Spanning Tree Algorithm has finished. Bridge 1 is the root because it has the lowest ID. Each of its ports are designated ports. B4 is not needed since B1 also connects L1 to L2 and has a lower ID. So both ports on B4 are blocked. The root ports, designated ports and blocked ports of each bridge are marked on the Figure. If we count path cost in number of hops, then both B2 and B3 offer the same path cost to the root for L3. However B2 is chosen as the designated bridge since it has a lower bridge ID. Similary, both B5 and B7 offer the same path cost for L5 to the root. Again B5 is chosen as the designated bridge because it has a lower ID. (B3 overrides B6 as the designated bridge for L4 for the same reason.)

BPDUs

The BPDU messages contain, for example, information about which bridge is believed to be the root, how far away the root bridge is, bridge id of the transmitting bridge, and information on how old the BPDU is believed to be (message age). These BPDUs are transmitted around the spanning tree as follows. The root bridge periodically transmits a BPDU every "Hello Time" seconds (typically 1-4 seconds). Other bridges only broadcast a BPDU on a link if they are Designated Bridge on that LAN. When a bridge receives a BPDU on its root port, it updates some of the fields (eg: message age), and transmits a copy of the BPDU on each of its designated ports. So the BPDU starts at the root and is propagated down the tree until all bridges have received it. Since there is exactly one Designated Bridge on a LAN, there will be exactly one BPDU message transmitted on a single link at a given time during steady state operation.

Traffic Flow

The spanning tree affects the flow of traffic in the following sense. The example in Figure 9 is derived from the Sample Network shown in Figure 8. BPDUs are initiated by the Root and flow only downwards along the tree. Data packets flow both up and down the tree, but always along the tree links. This example shows that the traffic flowing through the root bridge is likely to be heavier than the traffic flowing through other bridges. This example also shows that the Spanning Tree Algorithm may not choose optimal paths. For example, a packet going from host w to host x would have to flow through 3 LANs and 4 bridges to reach its destination. It would be shorter, again assuming a hop count measurement, to simply cross B7 (see Figure 8). The reason the Spanning Tree Algorithm doesn't always choose optimal paths, is because the priorities used for selecting roots and designated bridges are the bridge IDs. These are assigned arbitrarily by management, without regard to traffic flow, congestion, etc.

Topology Changes

Topology changes are caused by bridge or link failures. A bridge can either *be notified* of a topology change, or *detect* a topology change. A bridge is notified of a topology change when the appropriate flags are set in BPDUs. A bridge detects a topology change as follows. Normally, bridges receive BPDUs on their root ports and blocked ports. BPDUs received on blocked ports, are processed but not responded to. BPDUs received on root ports usually result in another BPDU being transmitted on designated ports. BPDUs flow through the spanning tree on a periodic basis, and every bridge knows the period. Thus if a bridge does not receive a BPDU on a port it expects to, within a fixed time period, then it assumes there has been a topology change. This is implemented as a timer expiry.

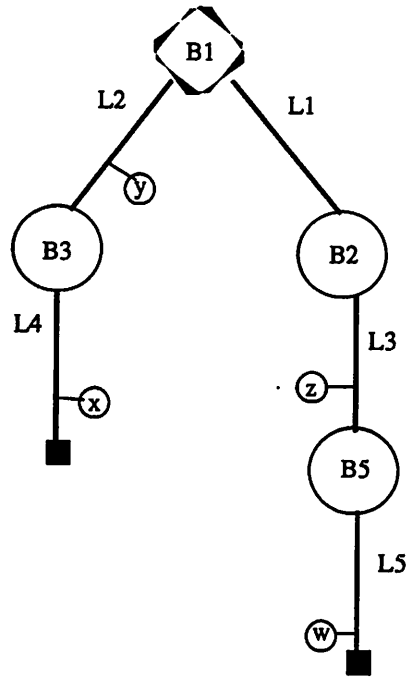


Figure 9: Directions of BPDU and Data traffic flow

After a bridge detects a topology change, it must notify the root. The root then in turn notifies all the other bridges. Once all the bridges have been notified, the reconfiguration process begins. Most of the topology changes that occur during the reconfiguration process are likely to occur near the location where the failure took place. Bridges located far from the failure may not experience much of a topology change at all. The reconfiguration process involves six tasks.

1. Select Root Bridge
2. Select Root Port on each Bridge
3. Select Designated Bridge for each LAN
4. Select Designated Port on each Designated Bridge
5. Change Port States
6. Clean up Filtering Tables

Since new routes are computed for the spanning tree, around the failed component, it is also necessary to compute new routes for packets. Therefore, the ageing timer in the filtering tables is set to a small value, so entries are aged out more quickly.

Consider again the sample network in Figure 8. If, for example, B2 were to fail, then B3 would take over as Designated Bridge for LAN L3. Hence the blocked port on B3 would now become a designated port. As another example, suppose the root bridge B1 were to fail. B2 would become the next root bridge. Figure 10 shows the state of all the ports after this topology change. Now B6 needs to select the root port based on port ID because the path cost in hops is the same through both of its ports.

Algorithm Implementation

The Spanning Tree Algorithm requires the following features in its implementation. Every bridge, or node, runs an identical version of the algorithm. Each bridge maintains a set of variables or parameters for its information. This information describes the bridge's current understanding of the topology of the system. It does not contain a complete description of the topology, but rather just information on the root, the bridges' relation to the root (distance or cost), and topology information near the bridge (i.e. its relations to its neighboring bridges and LANs). In order to implement the Spanning Tree Algorithm, a bridge needs two types of parameters and timers. **Bridge parameters and timers** are global parameters, whereas **Port parameters and timers** can be considered local parameters. There is one set of bridge parameters per node. There is one set of port parameters per port on a node. The bridges transmit

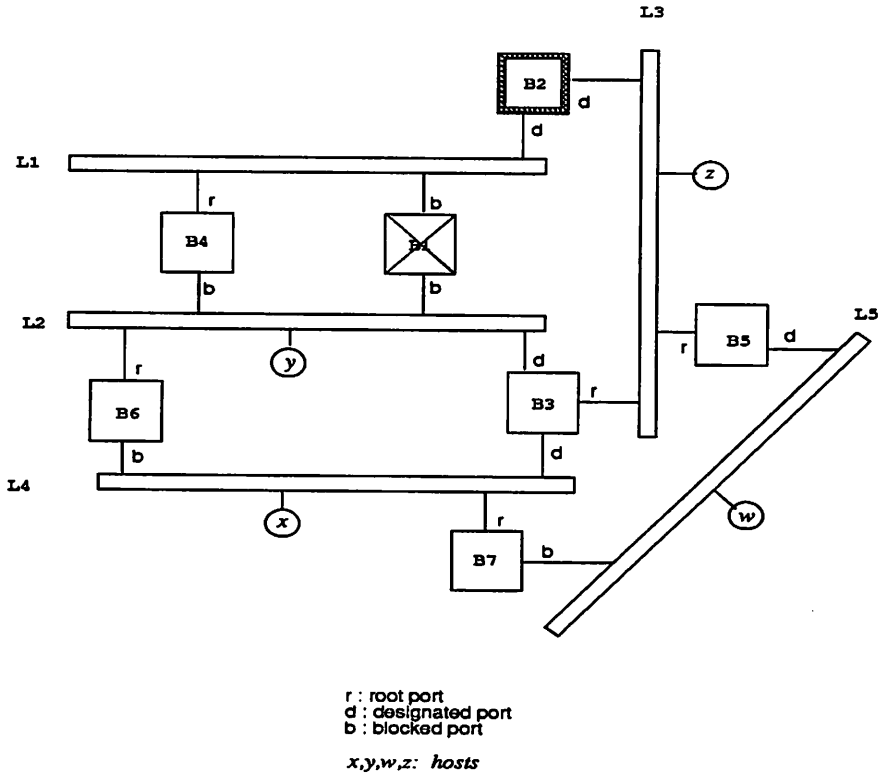


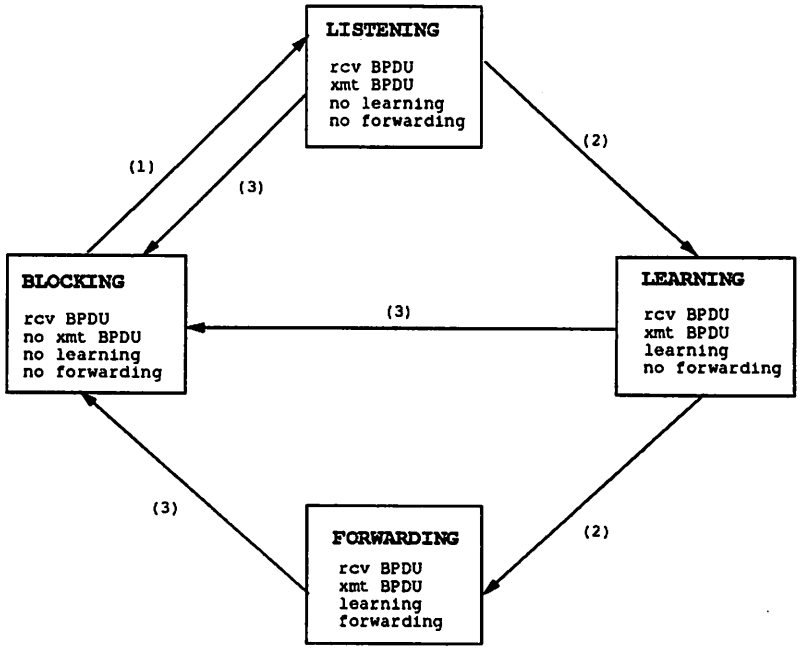
Figure 10: Sample Network after B1 failure

BPDUs to each other in order to share some of this information. Each time a BPDU is received on a port, some of that port's parameters and timers, and some of the bridge's parameters and timers, may be updated. Section 4.2 describes in detail the different Port States. Section 4.3 defines and categorizes all of the information maintained at a bridge node. Section 4.4 defines the contents of a BPDU message, and explains how new BPDUs are generated from bridge information.

4.2 Port States

In the spanning tree protocol a port can be in one of four states: blocked, listening, learning, or forwarding. A port stays in both the "listening" state and the "learning" state for a fixed about of time. The amount of time is determined by the **Forward Delay** parameter, and is the same for both states. When a port enters the listening state, a forward delay timer is initialized. When the timer expires, the port moves into the learning state and the timer is initialized again. After the next timer expiry, the port moves into the forwarding state. Figure 11 shows these four states, the events that cause a port to move from one state to another, and the processing activies of each state. The following list specifies in more detail which processes are enabled and disabled for a port in a given state.

1. **Blocked:**
   (a) BPDUs shall be received and processed as usual. Note that BPDU's are not resubmitted for transmission.
   (b) The Learning Process is blocked. Thus this process does NOT add station location information into the Filtering Database.
   (c) The Forwarding Process is blocked. Received frames are discarded and not forwarded on any port.
2. **Listening:**
   (a) BPDUs shall be received and processed as usual. BPDU's are submitted for transmission.
   (b) The Learning Process is blocked. Thus this process does NOT add station location information into the Filtering Database.
   (c) The Forwarding Process is blocked. Received frames are discarded and not forwarded on any port.
3. **Learning:**



(1) Algorithm selects as Designated Port or Root Port
(2) Expiry of Forward Delay Timer
(3) Algorithm selects as not Designated Port or Root Port

Figure 11: Port States

(a) BPDUs shall be received and processed as usual. BPDU's are submitted for transmission.

(b) The Learning Process is active. Thus this process DOES add station location information into the Filtering Database.

(c) The Forwarding Process is blocked. Received frames are discarded and not forwarded on any port.

4. **Forwarding:**

(a) BPDUs shall be received and processed as usual. BPDU's are submitted for transmission.

(b) The Learning Process is active. Thus this process DOES add station location information into the Filtering Database.

(c) The Forwarding Process is active. Received frames are processed and forwarded according to the information in the Filtering Database.

We now explain the need for the two intermediate states, the "listening" and "learning" states. Since there are propagation delays in passing protocol information throughout the bridged network, the bridges learn of topology changes at different times. They also learn the new topology at different times, and change their port states asynchronously as well. Because there is no synchronization between bridges during reconfiguration, there cannot be a sharp transition from one active topology to another. Sharp transitions could lead to a number of undesirable situations. Temporary tree partitions could form, and temporary loops could form. In order to avoid temporary tree partitions, we must wait long enough to be sure that all bridges have learned the new topology before changing port states. In order to avoid loops, all forwarding ports - which are to be turned off in the new topology - must be turned off before any blocked ports - which are to be turned on in the new topology - are actually turned on [7]. In the worst case, two nodes can learn the new topology within a window of MaxNetworkDelay time. As soon as one bridge learns the new topology, it turns off any previously forwarding ports which are to be blocked in the new topology. However it must wait MaxNetworkDelay time for the other nodes to learn the new topology and turn off their forwarding ports if necessary. Then the bridge can move ports into a forwarding state. This waiting period is implemented in two stages: a "listening" stage and a "learning" stage. A port spends 1/2 * MaxNetworkDelay time in each state. The transition from "blocking" to "forwarding" is separated into two stages for the following reason. During the first stage, old information in the filtering tables is timed out. During the second stage, new information is learned in order to minimise the initial flooding that would occur under a new topology. During the reconfiguration we also want to wait for the frame lifetime of any frames forwarded using the old topology

14 Bridge Parameters	*Set by Management*	*Set dynamically by Protocol*	*Used only when bridge is Root*	*Conveyed in BPDU*
current Root ID		*		*
current Root Path Cost		*		*
current Root Port		*		
current Max Age		*		*
current Hello Time		*		*
current Forward Delay		*		*
Bridge ID	*			*
Bridge Max Age	*		*	(if root)
Bridge Hello Time	*		*	(if root)
Bridge Forward Delay	*		*	(if root)
Topology Change detected FLAG		*		*
Topology Change FLAG		*		*
Topology Change TIME	*		*	
Hold TIME	*			

Table 1: Bridge Parameters

to expire, before forwarding new frames. The precise calculation of the **Forward Delay** parameter (time spent in each of the intermediate states), which is a function of the maximum network delay and maximum frame lifetime, is carried out in Section 4.6.2.

4.3 Information maintained at each Bridge node

4.3.1 Information Classification

Tables 1, 2, 3, and 4 list the Bridge Parameters, Bridge Timers, Port Parameters, and Port Timers. These tables give classification information about the parameters. They show commonality in the types of parameters and also outline some of their differences. For example, some parameters are used to maintain *current* topology information. Others are used only when the bridge is acting as the root bridge. Some of these parameters are set by management and are static. Other parameters change dynamically during the operation of the protocol. Another difference in the types of parameters, is that some of them need to be conveyed in the BPDUs, and others do not.

The third column of Table 1 shows which parameters are used only when the bridge is the root. The *current* parameters are, of course, always used, whether the bridge is root or not. However, if bridge B1 is the root, then its current

3 Bridge Timers	*Used only when bridge IS root*	*Used only when bridge is NOT root*	*Timeout value set equal to*
Hello Timer	*		current Hello TIME (bridge parameter)
Topology Change Timer	*		Topology Change TIME (bridge parameter)
Topology Change Notification Timer		*	current Hello TIME (bridge parameter)

Table 2: Bridge Timers

9 Port Parameters	*Set by Management*	*Set dynamically by Protocol*	*Conveyed in BPDU*
Port ID	*		*
Port State		*	
Path Cost		*	(only if port is designated port)
Designated Root		*	*
Designated Cost		*	
Designated Bridge		*	
Designated Port		*	
Topology Change Acknowledge FLAG		*	*
Configuration Pending FLAG		*	

Table 3: Port Parameters

MaxAge parameter = B1's Bridge MaxAge parameter. Moreover, all the bridges in the network will have their current MaxAge parameter set equal to B1's Bridge MaxAge parameter. The root bridge transmits its Bridge Max Age, Bridge Hello Time and Bridge Forward Delay parameters in the BPDU, and all the other bridges set their *current* parameters according to the corresponding value in the BPDUs.

In Tables 2 and 4 the timeout values for the bridge and port timers are derived from Bridge parameters. In Table 3 the first three port parameters give information about the port itself. The next four parameters give information about the LAN to which the port is attached. The last two parameters are boolean flags. Since the BayBridge is a dualport bridge, two sets of port parameters and timers are kept at each BayBridge.

3 Port Timers	*Timeout value of timer*
Message Age Timer	current Max Age parameter
Forward Delay Timer	current Forward Delay parameter
Hold Timer	Hold Time parameter

Table 4: Port Timers

4.3.2 Definitions of Parameters and Timers

In this section we define the bridge parameters, bridge timers, port parameters and port timers listed in the tables above. Additionally, the usage or method of computation is provided, when appropriate,

Bridge Parameters

1. **current Root ID:** The Bridge ID of the current root bridge.
2. **current Root Path Cost:** The cost of the path from this bridge to the root, through the current root port. If the bridge is the root, then this parameter is zero. Otherwise, Root Path Cost := Root port's Designated Cost parameter + Root port's Path Cost parameter.
3. **current Root Port:** The port ID of the port which offers the lowest cost path to the Root. The cost to the root through a given port is equal to the port's *Designated Cost* parameter plus the port's *Path Cost* parameter.
4. **current Max Age:** The current Max Age parameter := current Root Bridge's Bridge Max Age parameter. If the protocol information exceeds this age, then it is regarded as invalid. The protocol information refers to those bridge and port parameters which are dynamically set by the protocol (i.e. current information, see Tables). The age of this protocol information is computed using the Message Age Timer at each port.
5. **current Hello Time:** The current Hello Timer := current Root Bridge's Bridge Hello Time. If the bridge is root, the Hello Time specifies the time interval between periodic transmissions of Configuration BPDUs.
6. **current Forward Delay:** The current Forward Delay := current Root Bridge's Bridge Forward Delay parameter. It specifies the timeout value to be used by the bridge's forward delay timer. The forward delay timer is used for 3 things: the time a port spends in the listening state, while moving from the blocking state to the learning state; the time a port spends in the learning state while moving from the listening state to the

forwarding state; the value used for the ageing timer of the dynamic entries in the filtering table during a topology update.

7. **Bridge ID:** The unique bridge identifier of the bridge. The identifier is 8 octets in length. The least significant 6 octets are dervied from the MAC address; the 2 most significant octets used to set priorities. (See BPDU Format Section 4.4.)

8. **Bridge Max Age:** The value of the Max Age parameter, to be used by all bridges, when this bridge is the root.

9. **Bridge Hello Timer:** The value of the Hello Timer parameter, to be used by all bridges, when this bridge is the root.

10. **Bridge Forward Delay:** The value of the Forward Delay parameter, to be used by all bridges, when this bridge is root.

11. **Topology Change Detected FLAG:** A boolean parameter which is set equal to true when this bridge has detected a topology change. The root bridge must then be informed of this change.

12. **Topology Change FLAG:** This boolean flag is set to true when the bridge is notified of a topology change. This flag is set equal to the Topology Change flag in the received Configuration BPDU.

13. **Topology Change TIME:** If the bridge is the root, then this specifies the time period during which Configuration BPDUs are transmitted with the Topology Change FLAG set. Topology Change TIME := Bridge MaxAge + Bridge Forward Delay.

14. **Hold Time:** The minimum time period between transmissions of two consecutive BPDUs through a given port. No more than two BPDUs can be transmitted in a Hold Time period.

Bridge Timers

1. **Hello Timer:** If the bridge is root, then this timer determines the frequency of transmission of BPDUs.

2. **Topology Change Notification Timer:** This timer determines the frequency of transmission of TN BPDUs to the Designated Bridge on the LAN to which this bridge's root port is attached. TN BPDU's are repeatedly sent to the Designated Bridge until it responds by sending a BPDU with the Topology Change Acknowledge FLAG set. This is used only when the bridge is not root.

3. **Topology Change Timer:** This timer determines the time period during which BPDUs are transmitted, with the Topology Change FLAG set. This is used only when the bridge is root.

Port Parameters

1. **Port ID:** The port ID of the associated bridge port.
2. **Port State:** The current state of the port (i.e. disabled, blocked, listening, learning, forwarding).
3. **Path Cost:** The path cost is the contribution, of the immediate link, to the total cost to the root, IF this port were the root port.

$$Path\_cost = \frac{1000}{Attached\_LAN\_speed\_Mbps}$$

4. **Designated Root:** Equal to Root ID in BPDU received from the designated bridge from the LAN to which this port is attached.
5. **Designated Cost:** Cost of path to root through designated port of LAN to which this port is attached.
6. **Designated Bridge:** Bridge ID of the designated bridge for the LAN to which this port is attached.
7. **Designated Port:** Port ID of the designated port for the LAN to which this port is attached.
8. **Topology Change Acknowledge FLAG:** If the flag is set, then the Topology Change Acknowledge FLAG in transmitted BPDUs must be set. This flag will be set if the port receives a Topology Notification BPDU.
9. **Configuration Pending FLAG:** A boolean parameter set to record that a Configuration BPDU should be transmitted on expiry of the Hold Timer for the associated Port. This parameter is used, in conjunction with the Hold Timer for the Port, to ensure that Configuration BPDUs are not transmitted too frequently.

If a port is NOT the designated port for a given LAN (it is either a root port or a blocked port), then the Designated Root, Cost, Bridge and Port parameters correspond to information on another bridge; specifically the bridge which is the designated bridge for the LAN to which the port is attached.

Port Timers

1. **Message Age Timer:** This timer is used to measure the age of a BPDU received on the associated port. If the timer exceeds the value of MaxAge, then the BPDU information is discarded. This timer is also used to detect topology changes.

2. **Forward Delay Timer:** This timer determines the time a port spends in the Listening and Learning states.

3. **Hold Timer:** This timer ensure that the Configuration BPDUs are not transmitted too frequently through a bridge port.

4.4 BPDU Message Format

The initial two octets of all types of BPDUs specify the Protocol Identifier. The IEEE 802.1d Standard [1] has reserved one value for the Protocol Identifier which is used to specify the Local Spanning Tree Protocol. When this value is used, the following BPDU message format (see Figure 12) must be adhered to. These parameters are essential to the operation of the protocol. The Standard places no restrictions on message formats for other Protocol Identifier values.

There are two types of BPDUs: the Configuration BPDU and the Topology Change Notification BPDU. The parameters of each BPDU are determined by the BPDU type. The Topology Change Notification BPDU (TCN BPDU) consists of only 4 octets. These are the same octets as the first 4 octets of the Configuration BPDU. These four octets differ only in the 4th octet, which specifies the BPDU type. The format of the Configuration BPDU is given in Figure 12.

1. **Protocol Identifier:** This is a two octet field which identifies the specific protocol for which the BPDUs are to be used. The value of this field is determined by the Standards committees. The value 0000 0000 0000 0000 identifies the Spanning Tree Algorithm and Protocol.

2. **Protocol Version Identifier:** This is a one octet field which identifies the particular version of a protocol for hwich the BPDUs are to be used. This version identifier should be specified as an unsigned binary number. The IEEE 802.1d Standard requires the value 0000 0000 for this field.

3. **BPDU Type:** This is a one octet field. The value 0000 0000 denotes a Configuration BPDU. The value 0000 0001 denotes a Topology Notification BPDU.

4. *Flags:*

Protocol ID	Protocol Version ID	BPDU type	Flags	Root ID	Root Path Cost	Bridge ID	Port ID	Message Age	Max Age	Hello Time	Forward Delay
(2)	(1)	(1)	(1)	(8)	(4)	(8)	(2)	(2)	(2)	(2)	(2)

(N) : N = number of octets

Figure 12: Configuration BPDU format and parameters

(a) **Topology Change Flag:** This flag is encoded in bit 1 of Octet 5 of the BPDU. The flag is set when the bit takes the value 1.

(b) **Topology Change Acknowledge Flag:** This flag is encoded in bit 8 of Octet 5 of the BPDU. The flag is set when the bit takes the value 1.

(c) Bit positions in the Flag field which are not used by the protocol are reset, and will retain the value 0.

5. **Root Identifier:** This is an eight octet field. The bridge transmitting the Configuration BPDU fills in this field with the unique bridge ID of the bridge it believes to be the current root.

6. **Root Path Cost:** This is a four octet field. This field specifies the cost of the path from the transmitting bridge to the root bridge. The value is encoded as an unsigned binary number. The IEEE Standard requires this field to take values in the range 1 - 65535, with the value of 1 being an absolute minimum.

7. **Bridge Identifier:** This is an 8 octet field. The value is encoded as an unsigned binary number. The two most significant octets can be set by management, and thus allows the relative priority of bridges to be controlled. The least significant six octets are equivalent to the IEEE 48-bit format MAC address. These are Universally administered addresses,

and are used to ensure global uniqueness. If two bridge identifiers are compared, the lesser number shall denote the bridge of higher priority.

8. **Port Identifier:** This is a two octet field. The value is encoded as an unsigned binary number. The least significant octet is the port number as derived from the hardware. The most significant octet can be set by management and thus allows the relative priority of the ports to be controlled. If two port identifers are compare, the lesser number shall denote the port of higher priority.

9. **Message Age:** This is a two octet field. The value is encoded as an unsigned binary number. This field gives the age of the BPDU from when it was originally generate by the root.

10. **Max Age:** This is a two octet field. The value is encoded as an unsigned binary number. The field specifies the Maximum Age that a BPDU is allowed to reach. If it becomes older than MaxAge, then the BPDU information is disgarded. The value is determined by the current root. MaxAge = current Root's Bridge MaxAge parameter. The parameter is distributed throughout the network to ensure that each bridge uses a consistent value for its current MaxAge parameter. (All message age timer timeout according to same value.)

11. **Hello Time:** This is a two octet field. The value is encoded as an unsigned binary number. The field specifies the time interval between the generation of Configuration BPDUs by the root.

12. **Forward Delay:** This is a two octet field. The value is encoded as an unsigned binary number. The field tells a bridge which value to use for current Forward Delay parameter and its Forward Delay Timer. The parameter is distributed throughout the bridges to ensure that each bridge uses a consistent value for the Forward Delay Timer, when transferring the state of a port to the forwarding state.

When a bridge generates a BPDU, it will encode the fields, as follows, according to the information maintained at that bridge.

- Protocol ID := 0000 0000 0000 0000.
- Protocol Version ID := 0000 0000.
- BPDU Type := 0000 (Configuration BPDU) or 0001 (TCN BPDU).
- Flags:
  - Topology Change Flag := TC flag in most recently received BPDU.

  - Topology Change Acknowledge Flag := 1 if BPDU generated in response to a TN BPDU.

- Root ID := Bridge parameter: Root ID.
- Root Path Cost := Bridge parameter: Root Path Cost.
- Bridge ID := Bridge parameter bridge id.
- Port ID := ID of port through which about to transmit.
- Message Age := value of Message Age Timer on root port + estimated transmission delay.
- Max Age := Bridge parameter: current Max Age.
- Hello Time := Bridge parameter: current Hello Time.
- Forward Delay := Bridge parameter: current Forward Delay

4.5 Algorithm

The Spanning Tree Algorithm can be viewed as comprising two primary tasks, **detection** and **configuration**. The detection process must detect topology changes. These changes can be due either to failures or to recoveries (eg: bridges which become operational again after a failure). Network configuration takes place during initialization and reconfiguration occurs after a topology change. However these two processes (configuration and reconfiguration) are essentially the same and we do not need to distinguish between them. First we discuss the decisions to be made during the configuration process. Then we discuss the technique used in the detection process.

4.5.1 Configuration Process

Select Root
Initially, when there is no root, each bridge itself tries to become root. Each bridge transmits BPDUs on each of its ports, and sets its current Root ID parameter equal to its own Bridge ID. When it receives another BPDU on one of its ports, it compares the Root ID in the BPDU message with its current Root ID parameter. If the Root ID in the BPDU is less than the current Root ID parameter, the the Root ID parameter is set equal to this lower Root ID. This means the bridge gives up the role as the root, and allows the higher priority bridge to become root. This process of updating the current Root ID parameter, continues until no more changes occur at any of the bridges. At this point the

lowest Root ID has reached every bridge, and all bridges should be in agreement concerning which bridge is the root bridge.

Determine Path Cost
Distance to the Root is computed as follows. The Root is distance 0 from itself, and claims 0 as the length of its path to the Root in its transmitted BPDU Messages. When a bridge receives a BPDU message, it looks at the cost field in the BPDU message which gives the cost from the root to the previous bridge. Now the bridge must add the incremental cost of getting from it to the previous bridge. A simple technique would be simply to add 1 each time to the cost; this would represent the hop count. The Spanning Tree Protocol adds a link cost which is defined to be 1 over the speed of the link. Hence higher speed links have lower cost. Having computed the cost of the path from the root to this current bridge, the bridge now forwards the packet on its designated ports. The bridge sets the BPDU Root Path Cost field := Root Port's Designated Cost parameter. The next bridges to receive these BPDU messages will repeat the process.

Select Root Port
Each port has a Designated Cost parameter which specifies the cost to the root through that port. The root port is selected as the port with the lowest cost estimate (Designated Cost) to the root. The Designated Cost parameter at a port is updated when a BPDU is received on that port. It is updated by adding the Path Cost parameter of that port to the Root Path Cost value in the received BPDU. Since ports on a single bridge, may be connected to different speed links, each port may have different incremental path costs. Hence each port must keep track of its own path cost. If an update cause the Designated Cost of one port to become lower than the root port's, then the new port will take over as root port.

Once the Root bridge and Root port have been selected then all of the other *current* Bridge parameters can immediately be set, from BPDUs arriving on the root port.

```
IF bridge = root
  THEN  current Max Age = Bridge Max Age
        current Hello Time = Bridge Hello Time
        current Forward Delay = Bridge Forward Delay
  ELSE  current Max Age = BPDU Max Age
        current Hello Time = BPDU Hello Time
        current Forward Delay = BPDU Forward Delay
```

Select Designated Bridge and Designated Port
If two bridges are attached to the same LAN through ports which are not their

root ports, then only one of them can be chosen as designated bridge. After receving a BPDU message on each of their root ports, they will transmit updated BPDU messages through those ports that are attached to the same LAN. The bridges will pick up each others' BPDUs through nonroot ports. If one bridge sees that the other bridge offers a lower cost path to the root for that LAN than itself, then it will allow the other bridge to become designated bridge, and put its port in a blocking state. Recall that even if the port is in blocking state, it continues to receive BPDUs. It can continually check that the other bridge still offers a lower cost path. If the other bridge fails, then this bridge will try to take over as designated bridge for than LAN. The port which connects the Designated Bridge to its LAN becomes the Designated Port. If the bridge has multiple port connections to the LAN, then the port with lowest port ID is selected.

Once all the *current* Bridge parameters have been determined, and all the *Designated* Port parameters have been determined for each port, then the bridge can move its ports into their proper state.

4.5.2 Detection Process

Compute Message Age
The Message Age field of a BPDU is used to indicate the approximate time a specific BPDU has been alive in the network since it was originally generated by the root bridge. The IEEE 802.1d standard does not require a specific method to be used to compute this age. Since this time effects the performance of the Algorithm (time to detect and time to reconfigure), it is important to be careful not to underestimate or overestimate this time. The age should be overestimated in order to avoid premature timing out of correct current information.

When the root initiates a BPDU, it sets the Message Age field equal to 0. Each bridge forwards the BPDU with updated values. Each bridge should increment the Message Age field by the approximate time it takes a BPDU to travel one hop (1 hop = 1 LAN + 1 BayBridge). We assume that the propagation time of a BPDU on a single LAN is negligible compared to the total time it spends on a single bridge. After n bridges, the Message Age should be approximately equal to the sum of the bridge transit delays across each bridge. The bridge transit delay is composed of the algorithm processing time and the transmission time (includes queueing delays). We can use the Message Age Timer to measure the algorithm processing time. We use an average estimate to measure the transmission time.

When a bridge receives a BPDU on its root port, it sets the Message Age Timer of that port equal to the value of the Message Age field in the received BPDU.

The timer then begins to count up. After the BPDU has been processed, a new BPDU has been generated, and is ready to transmit, then the new BPDU Message Age field is set equal to the current timer value plus the estimated average transmission delay. (Precise calculation of the message age increment, added at each bridge, is given in Section 4.6.2.)

Topology Changes

A bridge detects a failure when one of its port's Message Age Timers expires. This means that it has not received a BPDU within the maximum allowable amount of time. Upon detecting a topology change, the bridge sends a Topology Change Notification (TCN) BPDU through its root port towards the root. Since the BPDU is transmitted through the root port, it is sent on a LAN towards the root. This will be picked up by the Designated Bridge for that LAN. In other words, a bridge node notifies its parent node in the spanning tree. The parent node must acknowledge its child node. The parent accomplishes this by sending a regular BPDU, with the Topology Change Acknowledge Flag set, to the child node (through its Designated Port). The child node sends the TCN BPDU periodically towards its parent node until it receives the acknowledgement. The bridge timer Topology Change Notification (TCN) Timer determines the period between repeated transmissions of the TCN BPDU. Each parent node must repeat this process sending a TCN BPDU to its parent, until the TCN BPDU reaches the root of the tree. Notice that the TCN BPDU is no more than a flag to indicate to other bridges that some change has occurred. It contains no information as to WHERE or WHAT the change was.

Once the root has been notified, it is then responsible for notifying all the other bridges that a change occurred. This is accomplished by setting the Topology Change Flag in regular BPDU transmissions. This flag will be set in BPDUs for a period of time, determined by the bridge timer Topology Change Timer. This time period must be long enough to ensure that all bridges have been notified of the change. Since some BPDUs may get lost, it is necessary to transmit BPDUs with this flag set repeatedly. If the root itself is down, then every bridge will have detected the topology change itself after MaxNetworkDelay.

A topology change can also arise from a recovery, not just a failure. A recovery refers to bridges which were down previously, and become functional and ready to participate again in the extended network. When a bridge starts to function, it will receive other BPDUs flowing through the network. If it chooses to respond to these BPDUs, it may transmit BPDUs which contain information that supersedes another ports' parameters (for example, perhaps the new bridge has higher priority to be Designated Bridge). If a bridge receives a BPDU on one of its ports, with information that supersedes its current state information, then the reconfiguration process begins.

4.6 Selection of Parameters Values for Algorithm Performance

4.6.1 Definitions of Spanning Tree Performance Parameters

We are primarily interested in calculating analytically values for the MaxAge and ForwardDelay parameters. In order to do this, it is necessary to calculate a number of other parameters as well. The parameters used in our calculations are listed in Table 5. This section describes the method and rationale for calculating the recommended values for all of these parameters. Definitions are provided (in the table) for those which may not be self explanatory, or were not defined in Section 4.3.

4.6.2 Calculation of Spanning Tree Performance Parameters

In this section we explain how numerical values were chosen for those parameters in Table 5. Some of the values had to be assumed. IEEE recommends the following provisional values:

- $dia = 7$
- $bt\_d = 1$ seconds
- $mma\_d \leq 0.5$ seconds
- $lost\_msgs = 3$

The list below presents the following information for each of the parameters: a formula, a reason for the formula (if it is not obvious), and the resulting numerical value. The numerical values are computed using the above assumed values and the others calculated below.

1. **bpdu_d**
   - $bpdu\_d = bt\_d$.
   - The transmission delay should be less than the bridge transit delay, $bpdu\_d \leq bt\_d$. The recommended value selects the worst case, at equality.
   - Recommended Value: $bpdu\_d = 1.0$ sec
2. **life**

Symbol	Name	Definition
dia	maximum bridge diameter	Maximum number of bridges between any 2 end stations in the bridged network.
bt_d	maximum bridge transit delay	The maximum time a data packet can spends on a bridge, from beginning of reception, until end of transmission.
bpdu_d	maximum BPDU transmission delay	The maximum time a BPDU waits, starting from when it is ready to transmit, until it begins transmission of the pending BPDU.
mma_d	maximum media access delay	(network dependent)
life	maximum frame lifetime	The longest time it should take a packet to reach its destination.
msg_tu	Message Age Timer time unit	Granularity, or resolution of Message Age timer
max_msg_age_inc	maximum Message Age increment	The maximum increment to be added to previous msg_age, when setting new BPDU msg_age field.
max_msg_age_est	maximum message age estimate	Message age is typically overestimated. This sets max on overestimate.
hello_t	Hello Time	
hold_t	Hold Time	
lost_msgs	number of consecutively lost messages	Assume probability of losing more than this number of consecutive messages very low.
msg_prop	maximum message propagation	Message propagation across longest path in network.
max_age	Max Age	
forward_d	Forward Delay	

Table 5: Symbol Definitions

- $life = (dia \times bt\_d) + mma\_d$
- The mma_d delay is included for the initial transmission.
- Recommended Value: $life \leq 7.5$ sec

3. **max_msg_age_inc**

   - The increment cannot be longer than the transit time across the bridge, $max\_msg\_age\_inc \leq bt\_d$. The recommended value selects the worst case, when $max\_msg\_age\_inc = bt\_d$.
   - Recommended Value: $max\_msg\_age\_inc = 1.0$ sec

4. **max_msg_age_est**

   - $max\_msg\_age\_est = max\_msg\_age\_inc \times (dia - 1)$
   - Recommended Value: $max\_msg\_age\_est = 6$

5. **hello_t**

   - $hello\_t = 2 \times bpdu\_d$
   - There is no purpose to transmit BPDUs more frequently than $bpdu\_d$, so $hello\_t \geq bpdu\_d$. However, the multiple 2 is somewhat arbitrary and only a suggested provisional value.
   - Recommended Value: $hello\_t = 2$ sec

6. **hold_t**

   - $hold\_t = bpdu\_d$.
   - If the hold time were greated than $bpdu\_d$, then the message propagation time would depend upon the hold time instead of the BPDU transmission time.
   - Recommended Value: $hold\_t = 1$ sec

7. **msg_prop**

   - $msg\_prop = ((lost\_msgs + 1)hello\_t) + bpdu\_d \times (dia - 1)$
   - This formula represents the worst case time it would take a BPDU to travel across the longest part of the network, allowing for some of the BPDUs to get lost. A bridge must wait at least this amount of time before timing out another bridge node, so that it does not mistakenly timeout a functioning bridge.
   - Recommended Value: $msg\_prop = 14$ sec

8. **max_age**

   - $max\_age = msg\_prop + max\_msg\_age\_est$

parameter	*recommended value*	*maximum*	*range*
maximum bridge diameter	7 bridges		
maximum bridge transit delay	1.0 sec	4.0 sec	
maximum BPDU transmission delay	1.0 sec	4.0 sec	
maximum Message Age increment	1.0 sec	4.0 sec	
Bridge Hello Time	2.0 sec		1.0 - 10.0 sec 1 sec granularity
Bridge Max Age	20.0 sec		6.0 - 40.0 sec 1 sec granularity
Bridge Forward Delay	15.0 sec		4.0 - 30.0 sec 1 sec granularity
minimum Hold Time	1.0 sec		1.0 sec FIXED value

Table 6: Summary of Calculated Parameter Values

- The *max_age* is the *msg_prop* plus the overestimate that might have been made on a BPDU packet.
- Recommended Value: $max\_age = 20.0$ sec

9. **forward_d**

   - $2 \times forward\_d = max\_msg\_age\_est + msg\_prop + life$
   - We compute twice the *forward_d* since the listening and learning port stages are each half the total time needed to transfer a port from a blocking state to a forwarding state. During reconfiguration, the Forward Delay must wait until all bridges have received the new topology information, and until all frames forwarded under the old topology have expired. (See Section 4.2.)
   - Recommended Value: $forward\_d = 15$ sec

These final calculations are summarized in Table 6. The maximum and range of values as suggested in the IEEE 802.1d Standard are also shown.

References

[1] Mac bridges, March 1989.

[2] V. Catania A. Puliafito and L. Vita. Availability and Performability in LAN Interconnection. *Infocom Proceedings*, June 1990.

[3] V. Catania A. Puliafito, S. Casale and L. Vita. A Multiple spanning tree protocol in Bridged LANs. *IFIP*, 1990.

[4] Floyd Backes. Transparent Bridges for Interconnection of IEEE 802 Lans. *IEEE Network*, January 1988.

[5] My T. Le and Nick McKeown. Baybridge project Report 0. *Project Proposal*, September 1990.

[6] Nick McKeown. Baybridge project Report 1. *FDDI-SMDS Bridge Architecture*, December 1990.

[7] Radia Perlman. An Algorithm for Distributed Computation fo a Spanning Tree in an Extended LAN. *Proceedings 9th Data Communications Symposium*, Montreal 1985.

[8] Nina Taft Plotkin. Bridge Protocol Interface. *Bridge Project Report 5*, May 1991.

[9] David Sincoskie and Charles Cotton. Extended Bridge Algorithms for Large Networks. *IEEE Network*, January 1988.