LAN Bridge Spanning Tree Animation
Final Year Dissertation
Computer Science BSc

Simon Arlott
Supervisor: Peter King

June 5, 2006

DECLARATION

I, Simon Arlott, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, programs) are properly acknowledged at any point of their use. A list of references employed is included.

Signed:

Date:

Acknowledgements

I would like to thank my supervisor Peter King for advice and support. I would also like to thank Pierre Weil, Alex Shapiro, and Paul Tenny for their support and proof reading abilities.

Abstract

The IEEE 802 committee has developed a series of algorithms for enabling bridges to be placed between LANs without the stations being aware of it. If the network formed has loops, the bridges run an election protocol to ensure that only sufficient bridges to form a spanning tree remain active. This project is to develop a system for specifying a configuration of LANs, simulating the operation of the spanning tree algorithm, and displaying the simulation in real-time using an interactive graphical environment.

Contents

1 Introduction **4**

2 Background **7**
2.1 Spanning Tree Algorithm 7
2.2 IEEE 802.1D Standard: The Spanning Tree Algorithm and Protocol 13

3 Requirements **15**
3.1 Network Configuration 15
3.2 Simulation 16
3.3 Display 17

4 Functional Specification **19**
4.1 Network Configuration 19
4.2 Simulation 20
4.3 Display 21

5 Design **23**
5.1 User interface 23
5.2 Internal data structures 27
5.3 Simulation 29

6 Testing **30**
6.1 Network Configuration 30
6.2 Simulation 30
6.3 Display 31

7 Evaluation **32**
7.1 Network Configuration 32
7.2 Simulation 38
7.3 Display 45
7.4 Scripting 47

8 Discussion **49**
8.1 Achievements 49
8.2 Limitations 50
8.3 Extensions and future work 51

9 Appendix - User Guide **54**

10 Appendix - Javadoc Summary **66**

Chapter 1

Introduction

Aims and Objectives

Networks use bridges to forward packets transparently from one part of the network to another, allowing a large network to be split up into sections improving performance and efficiency. For redundancy there will usually be multiple physical paths that can be used and when these create a loop the potential for broadcast packets to repeat indefinitely occurs. The spanning tree algorithm was developed to identify a suitable loop-free network and enable forwarding on only the bridge ports required to implement it.

The aim of this project is to create a program to set up a network configuration and run a simulation of the spanning tree algorithm on it to demonstrate what the algorithm does, and the status of the bridges and network when the algorithm is running.

Network configuration

A graphical interface will be developed allowing the configuration to be created, edited, and displayed. Since the algorithm will be demonstrated with the same configuration multiple times a feature to save and reload will be provided.

Simulation

The speed of the simulation will be variable and can be stopped and resumed at any time. It will be interactive, allowing the current status of any part of the network to be obtained and modified to see how the algorithm reacts to changes. Automated operation must also be provided to allow the simulation to run through certain situations on its own.

Display

The current spanning tree that has been produced from running the algorithm should be shown on the display. When displaying the simulation, packets between one part of the network and another should be shown to add animation to the system and to indicate where traffic is flowing. The program should be easy to use, without being too simple and unable to show enough detailed information at the same time.

Problems solved

To provide the network system that the simulation will run on a set of Java classes was developed to organise the creation, maintainance and display of the network. For the simulation an implementation of the spanning tree protocol was required and the example pseudo-code from the IEEE standard[2] has been converted to Java to provide this. Handling several packets moving around the network at once without becoming unresponsive required efficient processes and adequate granularity of updates.

The first attempt at producing an interface for editing the network was a system allowing free placement of LANs with the mouse pointer snapping to appropriate positions on and in line with LANs to keep it connected properly and lined up. However this method proved to be quite complicated to implement and was not particularly easy to use, so the grid-based system was used instead. The circle in figure 1.1 is the current position snapped to based on the position

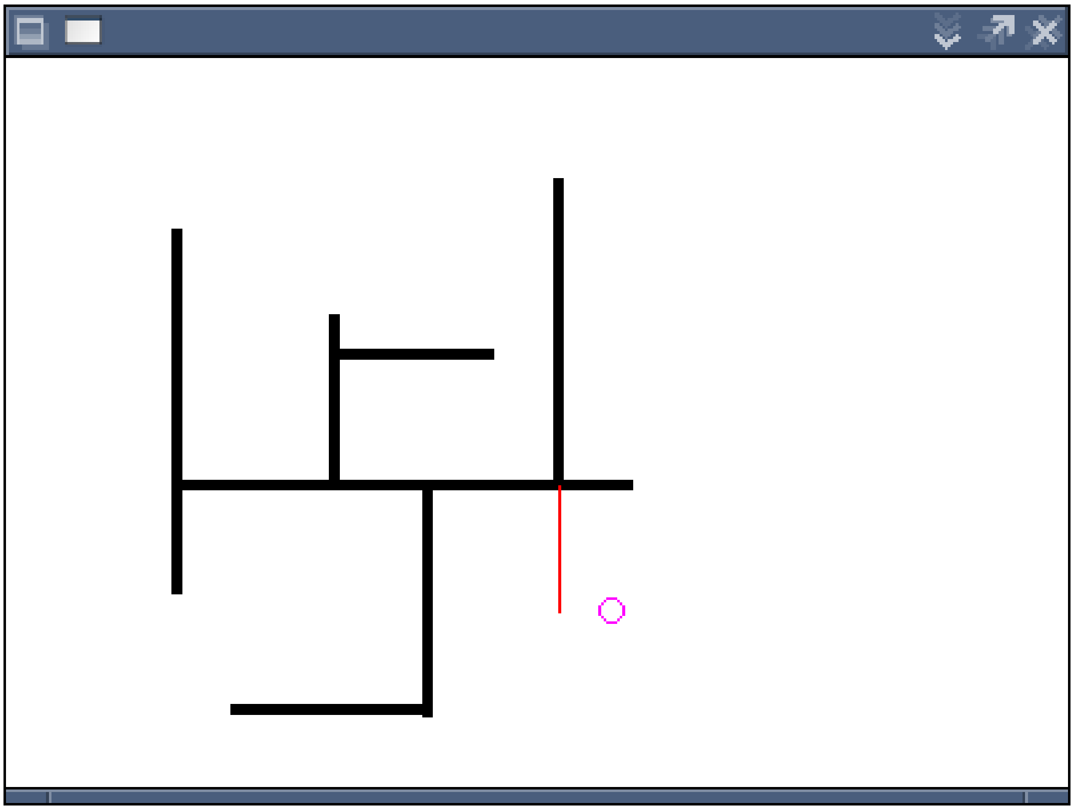

Figure 1.1: Freeform LAN placement with snap feature

of the mouse pointer and the thin line representing the parallel LAN.

Methods

A prototype was developed in Java to investigate possible network display and editing methods (see above) then the application was developed, also in Java, using the Eclipse SDK[1] which has very useful debugging features.

Results

Almost all of the aims and objectives have been met except a couple which were too complicated to implement (Section 8.1) and the result is a sophisticated application that allows the creation of a network and animates a simulation of the spanning tree protocol on it.

[1]http://www.eclipse.org/platform/

Chapter 2

Background

The IEEE 802 committee has developed a series of algorithms for enabling bridges to be placed between LANs without the stations being aware of it. If the network formed has loops, the bridges run an election protocol to ensure that only sufficient bridges to form a spanning tree remain active. This project is to develop a system for specifying a configuration of LANs, simulating the operation of the spanning tree algorithm, and displaying the simulation in real-time using an interactive graphical environment.

Interconnections, Second Edition — a guide to Internet networking describing Bridges, Routers, Switches and Internetworking Protocols (chapter 3) is used as the basis for the background information. [1]

2.1 Spanning Tree Algorithm

The spanning tree algorithm is designed to dynamically discover a subset of the topology that is loop-free but still has enough connectivity so that there is a path between every LAN. Bridges transmit messages between each other that allow them to calculate a spanning tree (Figure 2.1).

Configuration Messages

Root Bridge election A single bridge is elected on all the LANs to act as the *Root Bridge*. All the bridges must agree on which bridge is the root bridge which will be at the centre of the spanning tree with all the other bridges connected out from it.

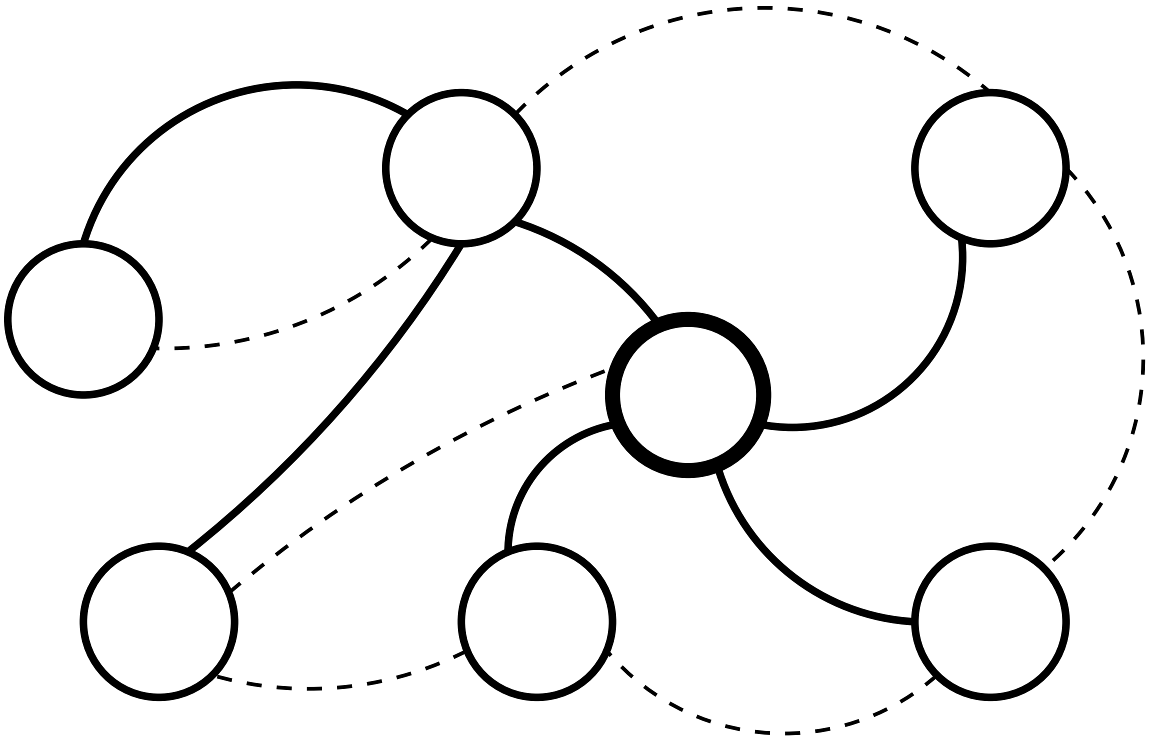

Figure 2.1: Example loop-free spanning tree (the root bridge is in bold)

Shortest path to root All bridges calculate the shortest path from themselves to the *Root Bridge*, this is based on the number of hops through the network to reach the root bridge computed from the cumulative path costs.

Designated Bridge election Each LAN elects a *Designated Bridge* from among the bridges residing on that LAN, which is the one closest to the *Root Bridge*. The *Designated Bridge* will forward packets from that LAN to the *Root Bridge*.

Root port Each bridge will choose a port known as the *root port* that is the best path to the *Root Bridge* based on the path costs involved.

Select spanning tree ports Each bridge will select the ports to be included in the spanning tree, these will be the root port and any ports for which that bridge is a *Designated Bridge* for the LAN.

Relevance to simulation

The simulation will need to model these messages and the algorithm to create an accurate simulation.

Relevance to network configuration

Configuration of the network will determine which bridge will become the *Root Bridge* based on an ID number. Different isolated networks will have different root bridges.

Spanning Tree Algorithm Refinements

Failures

The algorithm based on handling one-time configuration messages only handles the initial start-up of the network, and new bridges or links, but it does not adapt to failures of bridges or links. Configuration messages for each port are stored with a *message age* field which is incremented after each unit of time. If the message age reaches a certain threshold it will be discarded. The *Root Bridge* transmits messages with an age of 0, and other bridges will retransmit it with the same age. When new bridges appear the *Designated Bridge* (the bridge closest to the root bridge) may need to retransmit the message which it will do using the current stored value of the message age.

Avoiding Temporary Loops

Since temporary partitions in a bridge topology are worse than loops, the algorithm minimises the probability of loops by requiring that bridges wait for a certain period of time before changing a bridge port to the forwarding state. The time required should be sufficient for the change in topology to spread to the rest of the network.

Station Cache Timeout Values

Bridges learn the location of stations, and stations might move so it is important that a bridge forget the location of a station after a certain timeout period - this time can be several minutes. When a topology change occurs the bridges notify the *Root Bridge* which will set a topology change flag in its confirmation messages for a period of time to make all the bridges use a shorter station cache timeout of several seconds so that the new positions of stations are learned again and the network stabilises quickly.

Port ID

Each port has an associated ID to allow the spanning tree to be deterministic in that given the same topology the bridge will always select the same root port, and to allow a bridge to have two ports on the same LAN.

Performance Issues

Performance of the bridges is important for the spanning tree algorithm because lack of receipt of messages causes bridges to add connectivity, and extra connectivity (loops) is potentially disastrous.

One-way Connectivity

If the hardware of bridges fails so that connectivity between two bridges on a LAN becomes one-way, then loops can be created. The standard does not define what to do in this situation.

Networkwide Parameters

There are configurable parameters that all the bridges must agree on in order for the algorithm to work properly. To ensure that all the bridges use the same parameters, the *Root Bridge* includes the values in its configuration messages.

Settable Parameters

The following parameters can be set:

Bridge priority An ID used to influence the choice of the *Root Bridge* and the *Designated Bridge*.

Port priority A value used to influence the choice of ports when a bridge has two ports connected in a loop.

Hello time The time that elapses between generation of configuration messages by a bridge that assumes it is root.

Max age The message age value at which a stored configuration message is considered too old and discarded.

Forward delay The amount of time before a port is allowed to become part of the spanning tree.

Long cache timer The cache timeout used under normal conditions.

Path cost The cost to be added to the root path cost field in a configuration message received on this port, which is required to determine the cost of the path to the root through this port. (Set individually per port.)

Relevance to simulation

The simulation should allow a bridge to be disabled, as if it had failed, to show how the algorithm adapts. The stored message age needs to be incremented at a suitable speed. The time used for caching station locations under a stable topology needs to be long enough to allow inspection of the current cache of addresses at each bridge before the entries are expired, yet also short enough to allow the bridge to detect when a station has been moved. Since moving stations can only be done while the simulation is temporarily paused, the cache timeout value may need to be artificially lowered when a station is moved so that it expires soon enough for the simulation to be useful - the 15 minute timeout suggested may be suitable in a real network where it could take 15 minutes to move a station from one LAN to another, but not in a simulation where stations can be moved instantly.

Performance issues of bridges are not relevant to the simulation since the whole simulation will need to be carried out in a single thread to allow the speed of the simulation to be changed and paused reliably. Packets may become queued up when the network is congested, this can be handled by a FIFO buffer with a suitable limit to ensure that a flood of data packets do not cause the same effect as a bridge not having enough processing power to respond to configuration messages quickly.

Relevance to network configuration

The network configuration will require a method of setting all the relevant parameters for each bridge on a per-bridge basis.

Relevance to display

The status of the bridges, the network and the spanning tree topology need to be displayed in a graphical form so the effect of the algorithm can be seen. Information about the current status of a bridge including the station cache should be made available by selecting the bridge.

Octets	
2	protocol identifier
2	version
1	message type
1	flags: TCA[1] TC[2]
8	root ID
4	cost of path to root
8	bridge ID
2	port ID
2	message age
2	max age
2	hello time
2	forward delay

Table 2.1: Configuration Message Format

Octets	
2	protocol identifier
2	version
1	message type

Table 2.2: Topology Change Notification Format

Bridge Message Formats

The spanning tree algorithm uses only two types of message: configuration messages (Table 2.1) and topology change notification messages (Table 2.2).

Relevance to simulation

The simulation will need to implement both message formats and support sending and receiving of them for the simulated bridges.

2.2 IEEE 802.1D Standard: The Spanning Tree Algorithm and Protocol

The IEEE 802.1D Standard section 4 describes the algorithm and protocol of the Spanning Tree algorithm for a Bridged Local Area Network topology. The standard has requirements of bridges and the algorithm, and a detailed explanation of the protocol and how it works including

[1]Topology Change Acknowledgement
[2]Topology Change

sample pseudo-code in C. [2]

Relevance to simulation

Pseudo-code example in C and detailed explanation of the protocol and algorithm sets out what the simulation will have to do to model the spanning tree algorithm.

Relevance to network configuration

The standard suggests values for settable parameters, which can be used as defaults in the network configuration part of the program.

Chapter 3

Requirements

3.1 Network Configuration

Loading and saving

Save network configurations to a file so they can be loaded again later. It would be very inconvenient to users if they had to recreate networks over and over again each time the program is run. Networks can also be saved and given to other users to load.

Graphical edit interface

Create and modify a network of LANs, hosts and bridges using the mouse. A graphical interface is far easier to use than configuring a 2D network using a text file or inputting commands.

Ability to reconfigure network while the simulation is paused

This allows the possibility of adding more bridges and hosts to see the effect on the existing network. Networks are rarely static, bridges can fail, LANs can be disconnected from parts of the network and hosts can be moved around so users will need to be able to model these activities too.

Bridge identification

Each bridge and port requires an ID, so these must be configurable. The sequentially assigned IDs will not be very useful if they cannot ever be changed.

Bridge parameters

Individual settings of the algorithm parameters for each bridge. Required to show the effect of changing the parameters on the way in which the bridges discover other bridges on the same LAN.

3.2 Simulation

Variable speed

The overall speed of the simulation can be changed to see the changing state more clearly. When running at maximum speed the effects of changes can be seen more quickly, but packets on the network cannot be seen easily if they moving around at high speed and so a slower speed is required to see the effect at the level of packet activity.

Pause function

The simulation can be paused at any time so that the state of bridges in the network can be viewed without it changing. The ever changing network when the topology is being reconfigured could be viewed by slowing the network down but pausing it completely is better, instead of having the simulation just running all the time the program is open.

Automation using scripting

Define the initial state and changes in the state of bridges and hosts using a list of events to follow. The automated script should be saved with the network configuration.

Stepping function

Runs the simulation until the next major event, such as a bridge making a change to its record of the root ID. The automated scripting system requires manual configuration so a function to immediately start the simulation going until a predefined type of event is useful for quick tests of the network.

Trigger a packet from a host to another host

Manually send a packet from one host to another or all hosts.

Random packet sending state for hosts

Automatically sends out packets from hosts to other hosts on the network at random. It would be very useful to simulate activity on the network without having to repeatedly send packets from different hosts.

Accessibility

It must be possible to control the simulation and switch between network objects using only the keyboard.

3.3 Display

Show/modify current status of a bridge

Information about the current state of bridges should be shown when they are selected and allow modification of the operation of the bridges as required for the simulation.

View log of a bridge's status

Information about the past states of the bridge as it changes in response to the spanning tree configuration packets.

Visual indication of packets

Animate packets moving around the network to show activity. The simulation would be boring and it would be hard to determine what activity, if any, was occuring if there was no indication of the packets moving from one part of the network to another.

Spanning tree

Show the current spanning tree configuration to indicate which parts of the network are being connected to the rest of the network by the bridges.

Control hosts

Hosts need to be selectable and made to create data packets with a random or specified destination.

Control simulation

The simulation will require a way to control the speed, pause, and stepping functions and also the scriptable automation function.

Chapter 4

Functional Specification

4.1 Network Configuration

Loading and saving

The network layout and current configuration values are kept by the program while it is running (Section 5.2), this information can be reset to a blank network after prompting the user to save it. Saving the network involves writing the details of the network to a file. The network can be loaded from or reverted to a saved file. Loading or reverting the network will reset the simulation.

Graphical edit interface

The LANs, hosts and bridges in the network are shown in the same way as used for the simulation. A grid to which the mouse pointer snaps is used for positioning components and simplifying the layout. New network components can be created by selecting the type of component and then clicking where to start and finish placing it.

Existing components can be moved with differing freedom depending on the type; bridges, and LANs with attached hosts can be moved as a single group while hosts can be moved along the attached LAN or relocated to a different LAN. Components can also be deleted, removing them from the simulation.

Components with configurable settings such as hosts and bridges can be selected which will update a list of properties showing the current settings and allowing them to be changed.

Ability to reconfigure network while the simulation is paused

While the simulation is paused it is possible to modify the network without affecting the simulation unneccessarily — moved components will still have any packets in transit at the new location, components extended in length will not remove packets currently moving on them. Deleting components will result in packets on those components being lost. New LANs can be connected to bridge ports which will be detected by the simulation when it is resumed.

Bridge identification

Each bridge and bridge port's ID can be set while the simulation is paused or running using the same method as other network component settings, but any changes will only have an effect the next time the spanning tree algorithm is run.

Bridge parameters

The bridge parameters for the spanning tree algorithm can be changed or set to suggested defaults at any time.

4.2 Simulation

Variable speed

The speed of the simulation can be changed using a standard slider interface.

Pause function

A pause button will pause or resume the running simulation. While paused the current state of the bridges can be inspected and modified without additional events occurring. The network editing mode will automatically pause the simulation.

Stepping function

To allow the simulation to be run incrementally the paused simulation can be run until a significant event, such as a new bridge configuration packet being received that changes the state of a bridge.

Automation using scripting

The scripting system is able to define a list of states the simulation should run through, optionally being made to pause at each state and allowing network component settings to be changed when each state is reached and the simulation is started or resumed.

Random packet sending state for hosts

An option in the configuration of each host will enable or disable the random sending of packets to other hosts known to be reachable at some point on the network. This can be changed while the network is being edited and also when the simulation is running.

Trigger a packet from a host to another host

Individual packets to other hosts can be triggered from the options available while the simulation is running when a host is selected.

4.3 Display

Show/modify current status of a bridge

When a bridge is selected the current status is shown and configurable settings can be changed.

View log of a bridge's status

The most recent state changes are logged and shown when the bridge is selected.

Visual indication of packets

Packets are shown as coloured lines moving along the LAN and host connections, at intersections they will be doubled to reflect the broadcast nature of the network.

Spanning tree

The current spanning tree set up by the algorithm is shown by changing the colour of the connections between LANs and bridge ports to grey instead of black when they are not used in the tree.

Control hosts

Selecting a host will show a button that makes it send a packet to a specified host and an option to enable or disable the random packet sending feature.

Control simulation

The simulation can be paused or resumed and also run incrementally using the stepping function through toolbar buttons.

Chapter 5

Design

The program will be implemented in Java and use the Swing API for the graphical user interface. This section describes the results of the design phase with example mock-ups of the intended layout.

5.1 User interface

The user is presented with a menu for accessing the file-related commands and help, and a toolbar for the simulation functions. Three modes are available for working with the network, the simulation mode (Figure 5.3), the layout edit mode (Figure 5.1), and the scripting mode (Figure 5.2). Toolbar buttons for changing to the edit and scripting modes will pause the simulation (if running) and change to that mode.

In both modes the mouse pointer will snap to positions splitting up the available space into 10 by 10 pixel squares. In the edit mode a grid will be displayed underneath the network showing where the mouse pointer will snap to, helping with alignment of components.

For the layout edit and simulation modes a panel interface for modifying selected network components will be shown at the side of the window (optionally on the left or right). In the scripting mode this will show the list of states.

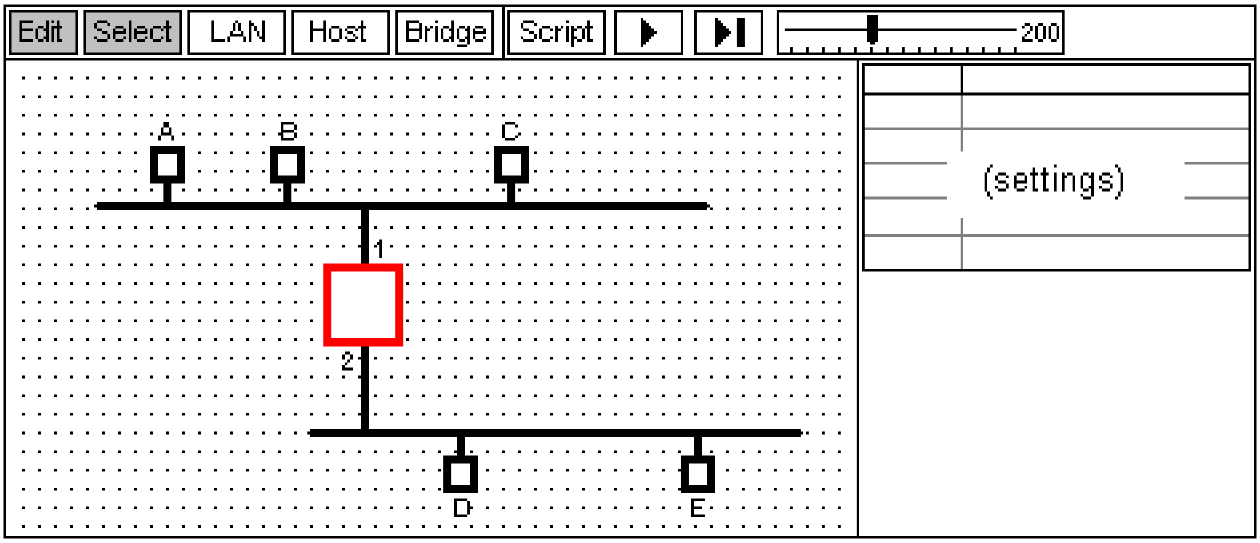

Figure 5.1: Layout edit mode

Layout edit mode

Additional toolbar buttons will be displayed which will facilitate switching between four options: selecting components, adding LANs, adding hosts, adding bridges. The configuration panel will change to be oriented towards changing network settings instead of showing the current simulated network state.

To make the interface simpler to use for modifying the network layout, the creating and moving of components will be available using only the mouse buttons without the need for using the toolbar to select an action as well as a component type.

Clicking with the primary mouse button will initiate placement of a new component and clicking with the secondary mouse button while hovering over an existing component will initiate moving components. While placing a component, the primary mouse button will finish and the secondary mouse button will cancel the operation. For movement, the primary mouse button will move the component to its new position and the secondary mouse button will cancel the operation. Pressing the *delete* key on the keyboard while a component is selected for moving will delete that component (and any connected hosts in the case of LANs).

When selecting components the primary mouse button will change the currently selected

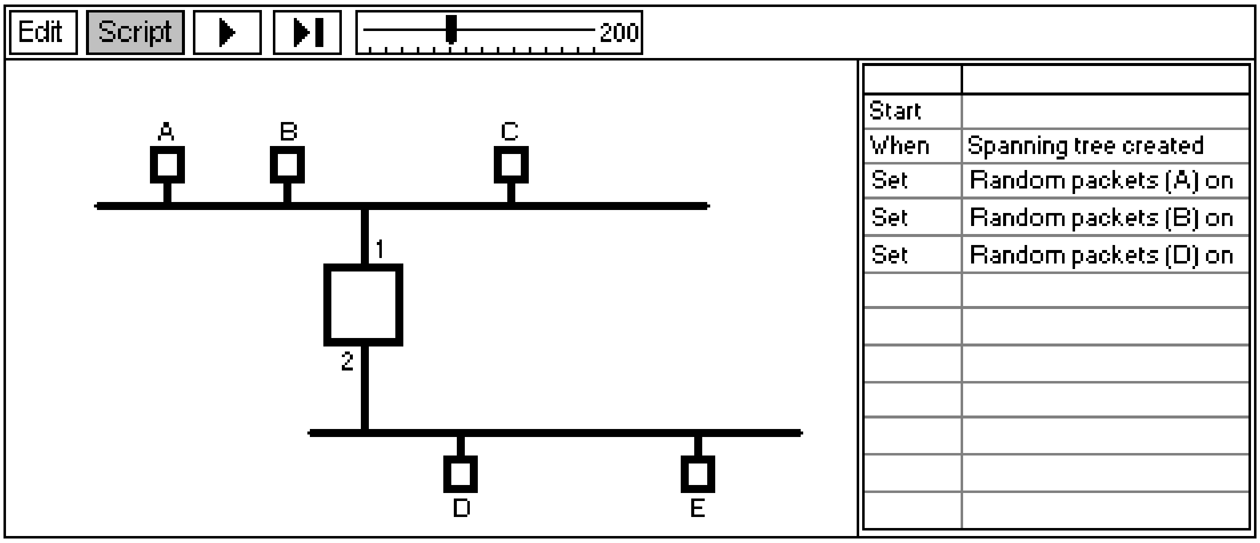

Figure 5.2: Scripting mode

component to the one under the mouse pointer and the configuration panel will change to show the settings for that component and allow them to be changed. For convenience moving and deleting will also work in this mode.

Scripting mode

In this mode the states, component setting changes and the option to pause when the state is reached will be shown as a list in the configuration panel. The network will still be visible so that the IDs of the network components to be modified are available.

Simulation mode

Toolbar buttons for pausing, resuming and stepping through one event or state change of the script will be used to control the simulation state. A slider will control the overall speed of the simulation.

When a component is selected using the primary mouse button the configuration panel will change to show the status, commands and simulation settings for that component. For hosts this will show buttons to send out a packet to a specified host and the option to enable random packet sending. For bridges the current spanning tree parameters (as set by the root bridge or

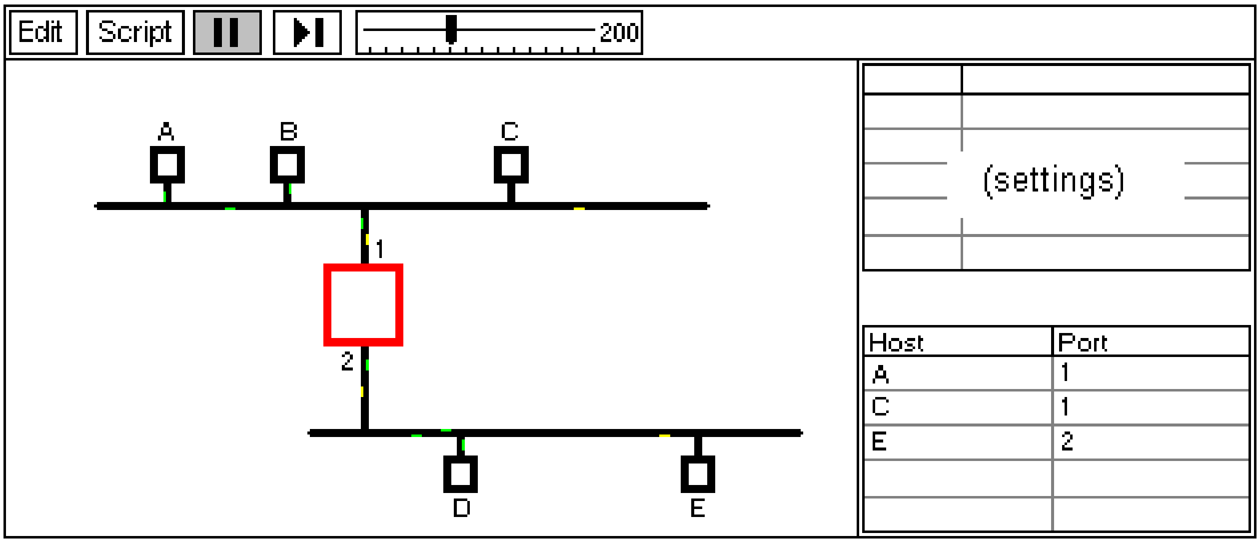

Figure 5.3: Simulation mode

from the default settings) will be shown, along with the current host address table and a log of recent bridge activity. An option to show the host address table as a separate on-top window or as part of the displayed network should be made available so that the host tables of multiple bridges can be shown at once.

Display

The display of the network will use thick lines for LANs and connections (to allow for an area for animated packets to be shown on), and boxes for the hosts and bridges with text labels to identify them. Packets will be shown as three different colours, one for host to host communication, one for bridge configuration message packets and another for topology change notification packets (Figure 5.3).

Help

Context-sensitive online help will be provided through the use of tool-tips, a link to the user guide will be available from the help menu.

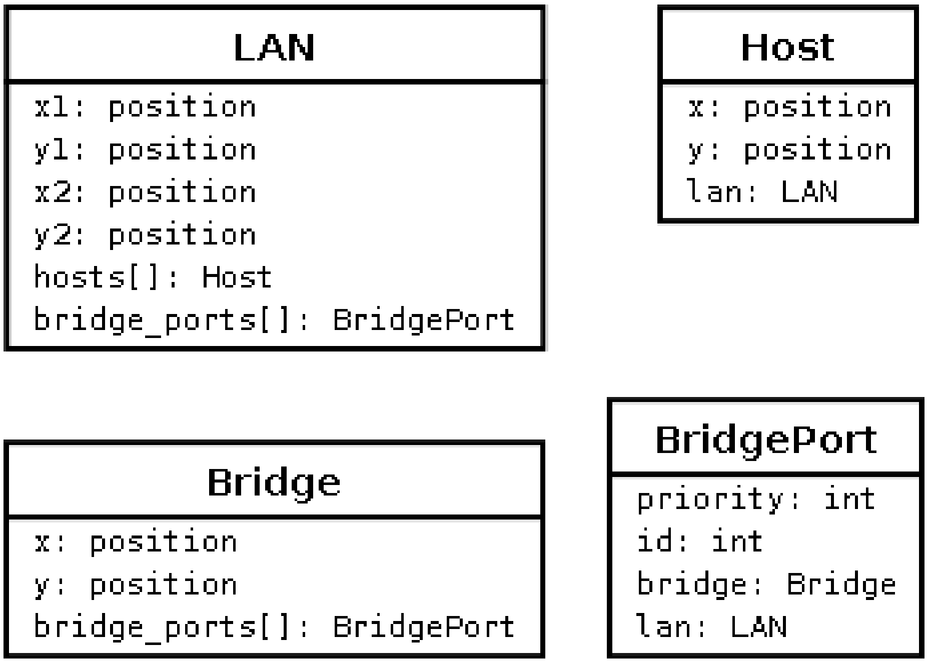

Figure 5.4: Objects used to represent network components

5.2 Internal data structures

The network components will be stored internally as objects linked to each other (Figure 5.5) as well as with graphical co-ordinates (Figure 5.4) to allow the simulation to be run on it without complex detection of connected components and the display process to be as simple as possible for better performance. This linking process will be carried out at the point of adding it to the network layout using the edit mode, or when loading the data from a file.

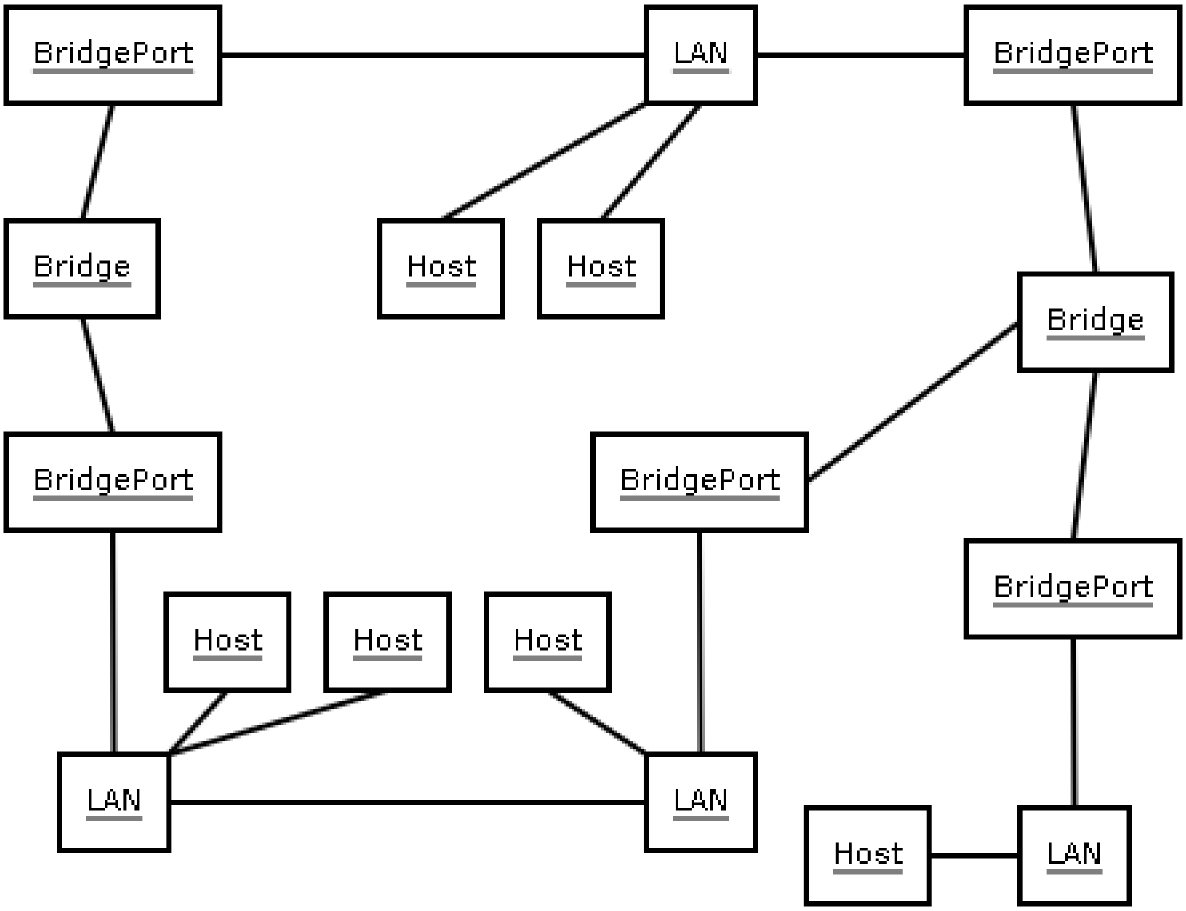

Figure 5.5: Internal data structure of network components

5.3 Simulation

The simulation process will keep track of all the packets on the network and run in a loop at the speed set by the user. With each iteration of the loop each packet will be moved along the LAN duplicating it at every connection (a host, another LAN or a bridge port) until it has nowhere else to go and is discarded. The LANs which receive the duplicate packet send it along in the same way. Hosts and bridges will receive their copy of the packet and process it, with the bridges forwarding it to another LAN or generating new packets if necessary. Bridge timers will also be checked each time the loop is run and react as required by the spanning tree protocol.

Information about the current state of the algorithm will be kept in each bridge object, instead of globally, although information about the global state will be available for the scripting mode. When running from a script the simulation will check it after each loop iteration to find out if the simulation should stop or settings should be changed. Information about certain key event types that occur during each iteration will be stored so that it can be used when checking the scripting state elements.

Chapter 6

Testing

6.1 Network Configuration

The network configuration testing involved ensuring that loops involving only LANs (since LANs can be connected to other LANs for a more flexible editing environment) cannot be created at any time, and checking that packets are handled correctly if the network is modified while the simulation is running (but paused). The ability of the program to avoid ambiguity in the network by not allowing network components to overlap was tested to make sure the displayed behaviour never differs from the expected one.

The ease of use of the program has also been evaluated by developing a list of tasks the user would want to perform when creating and modifying a network, and examining how this would be done without having any prior knowledge of the software.

6.2 Simulation

Implementation of the spanning tree algorithm

The simulation was tested by creating a set of diverse test networks and running them and checking the resulting bridge configurations to check that it responds correctly in all possible situations that the algorithm was designed to handle such as bridge ports connected to the same LAN, varying spanning tree algorithm settings on each bridge, multiple bridge failures or hosts moving LANs to ensure the simulation will run as expected.

User interface

The control of the simulation was be tested to ensure it behaves as expected, with no undesired behaviour resulting from pausing and running the simulation, or from using the stepping function in conjunction with the pause function.

6.3 Display

Testing of the display was an evaluation of how well the displayed network components and packet animation serve the aims and objectives of the program.

Chapter 7

Evaluation

7.1 Network Configuration

The program does not allow loops of LANs to exist by checking that there is no loop whenever a LAN is added - if there is a loop the new LAN will not be added to the network (Figures 7.1 and 7.2). The component being added changes colour from to green to red to indicate that it cannot be placed.

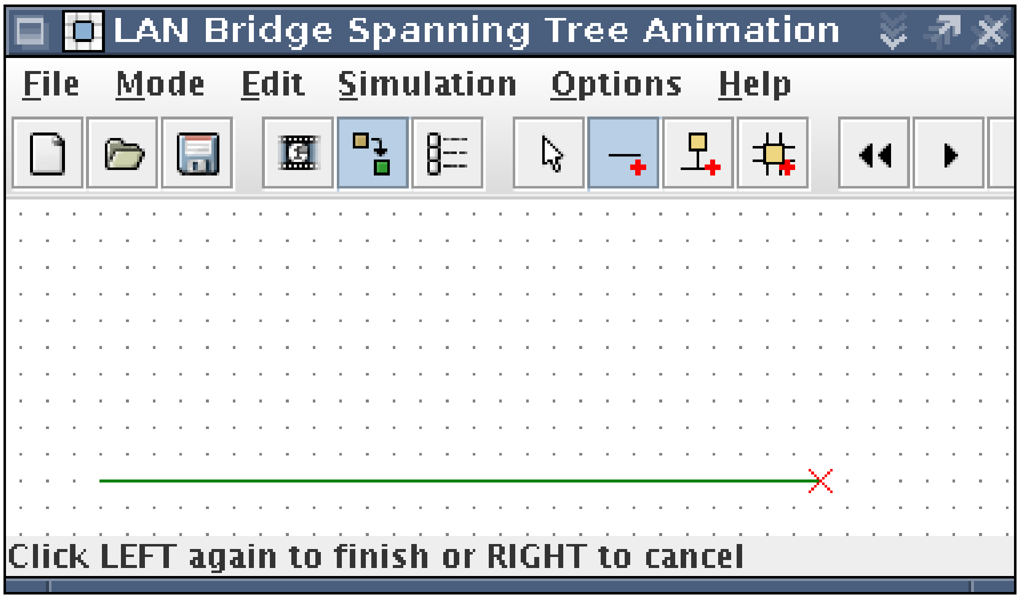

Figure 7.1: Adding a LAN to the network

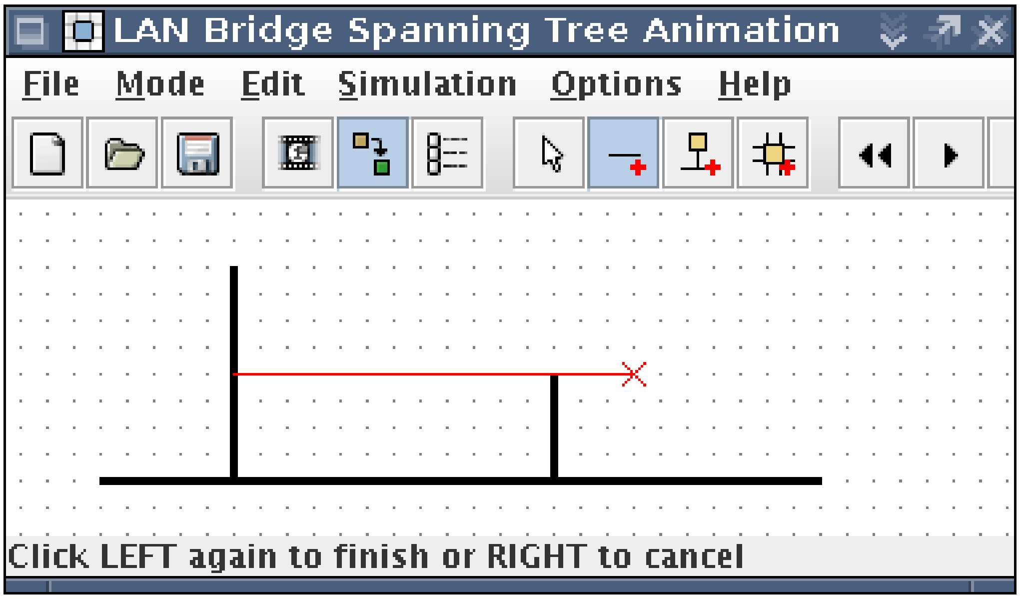

Figure 7.2: Attempt to make a loop

Network components are not allowed to overlap (except LANs which will join if overlapped), so that the displayed network is always a clear representation of the virtual one (Figures 7.3, 7.4 and 7.5). The overlap detection does not prevent bridge components being placed right next to each other if such a configuration is desired.

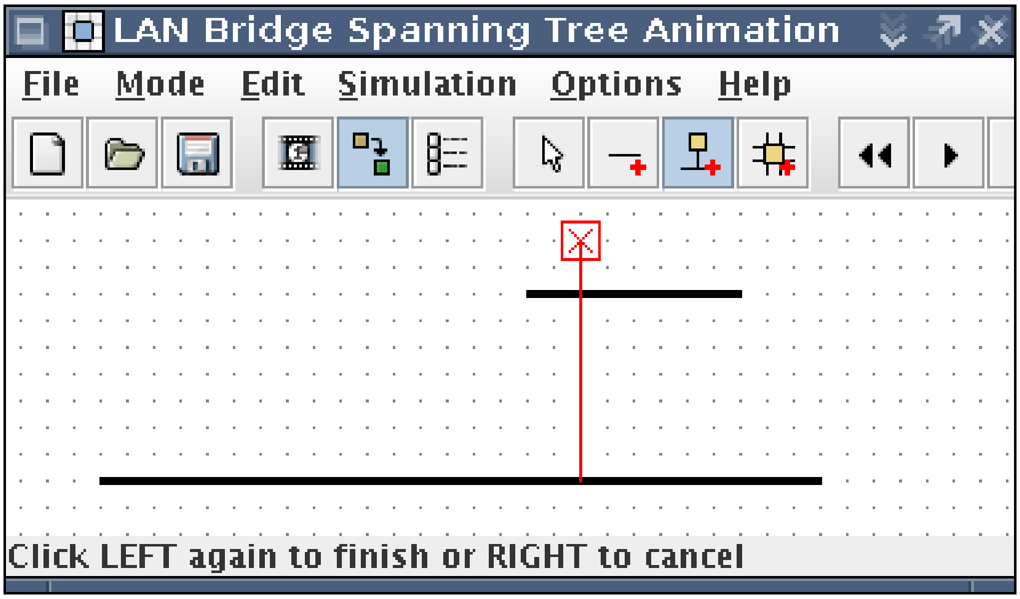

Figure 7.3: Attempt to add a host that goes through another LAN

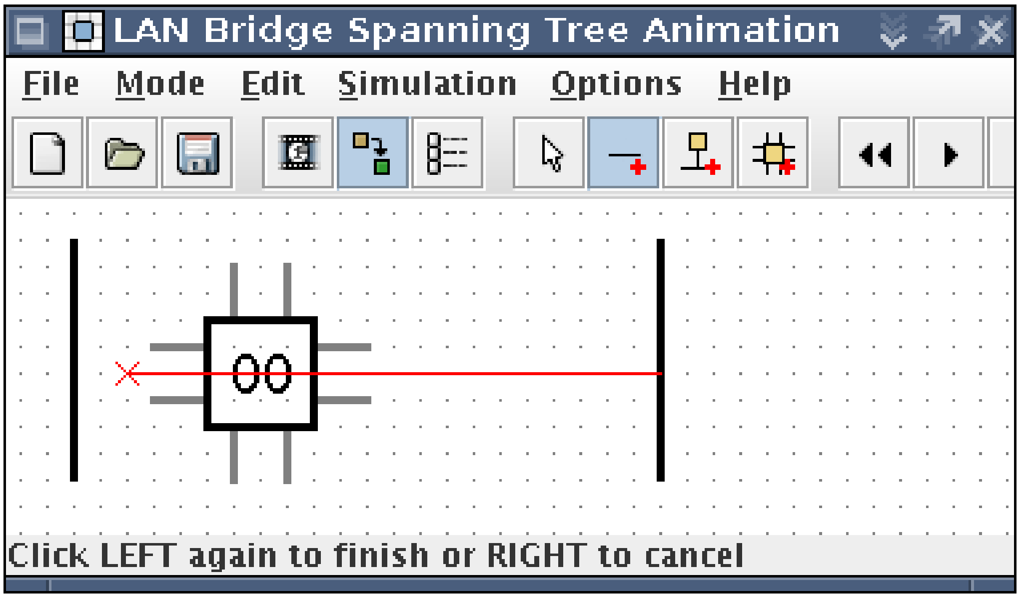

Figure 7.4: Attempt to add a LAN that goes through a bridge

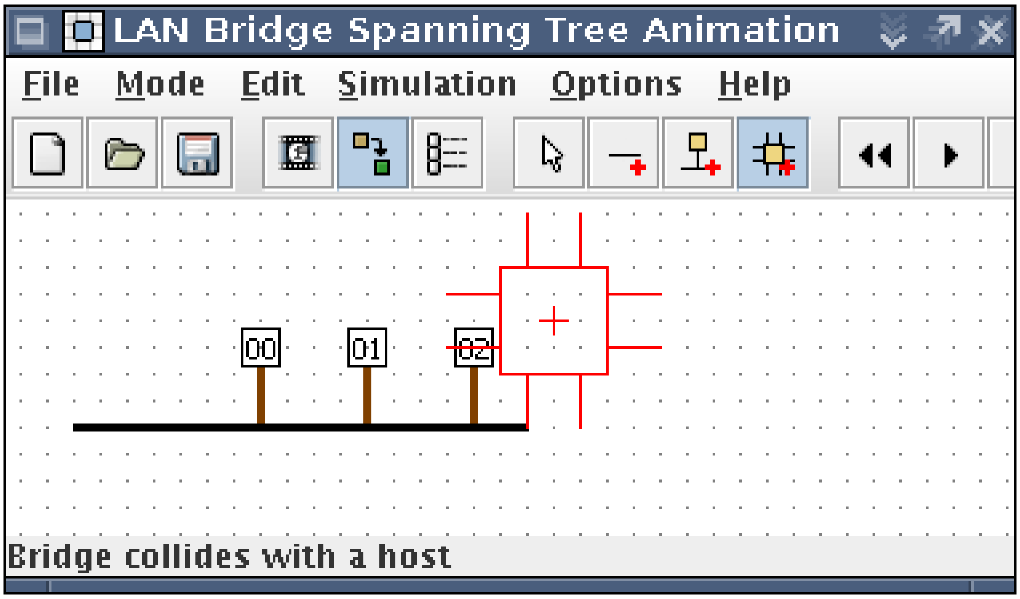

Figure 7.5: Attempt to add a bridge that would obscure part of the network

Creating a new network

To evaluate the usability of the system it must be possible to easily create a network without any prior knowledge of the program. The process of creating some LANs, adding hosts and bridges to them and then editing their parameters is considered here.

Starting up the program for the first time, the status text at the bottom states how to start creating the first LAN (Figure 7.6) and then after placing those adding a host by selecting the add host menu item or using the toolbar is explained in the same way (Figure 7.7). Moving components to make room for a bridge is a simple task with the components being moved changing colour to indicate if the new position is valid or not (Figure 7.8). Once the bridge has been added (Figure 7.9) after selecting the add bridge mode the parameters for the bridge (Figure 7.10) and hosts (Figure 7.11) can be set using the selection mode which is activated in the same way as the component add modes. The network is now configured and ready for the simulation to be run.

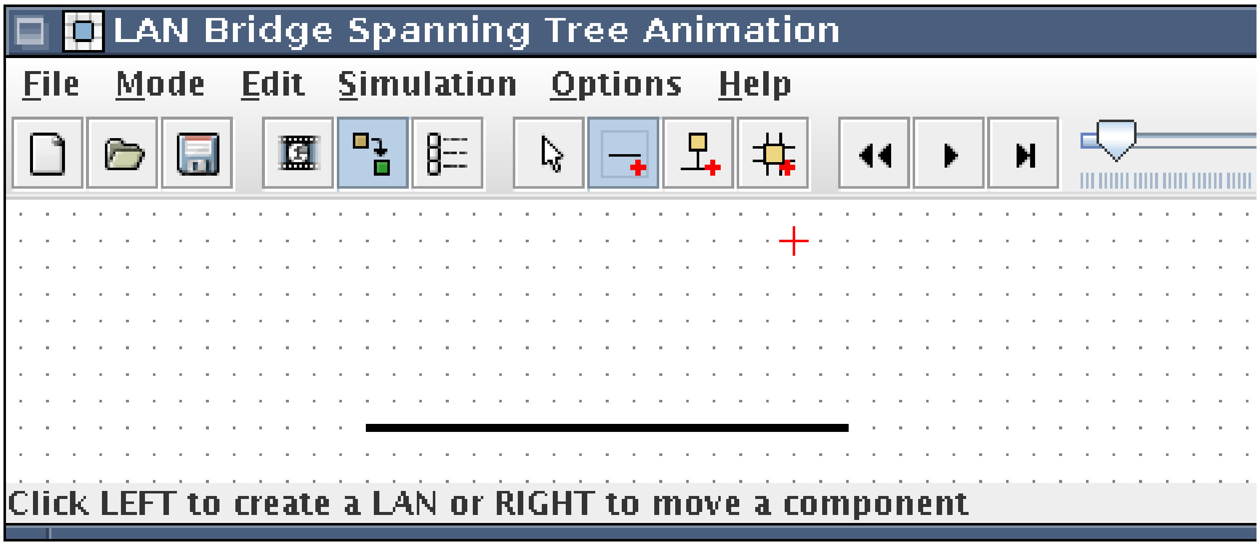

Figure 7.6: Step 1: Creating the first LAN

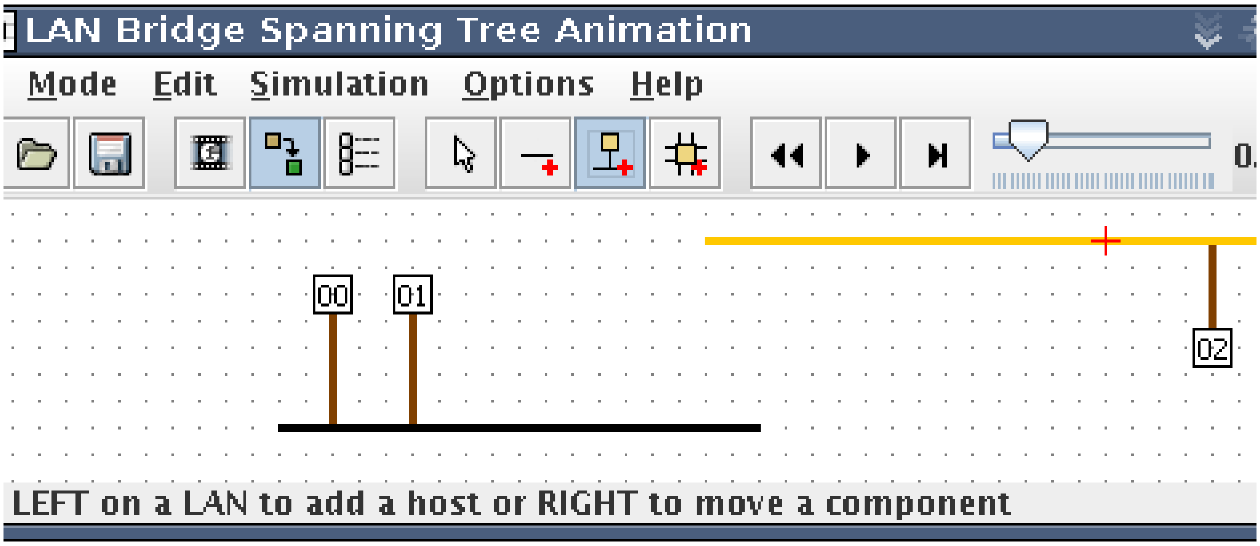

Figure 7.7: Step 2: Adding hosts to the LANs

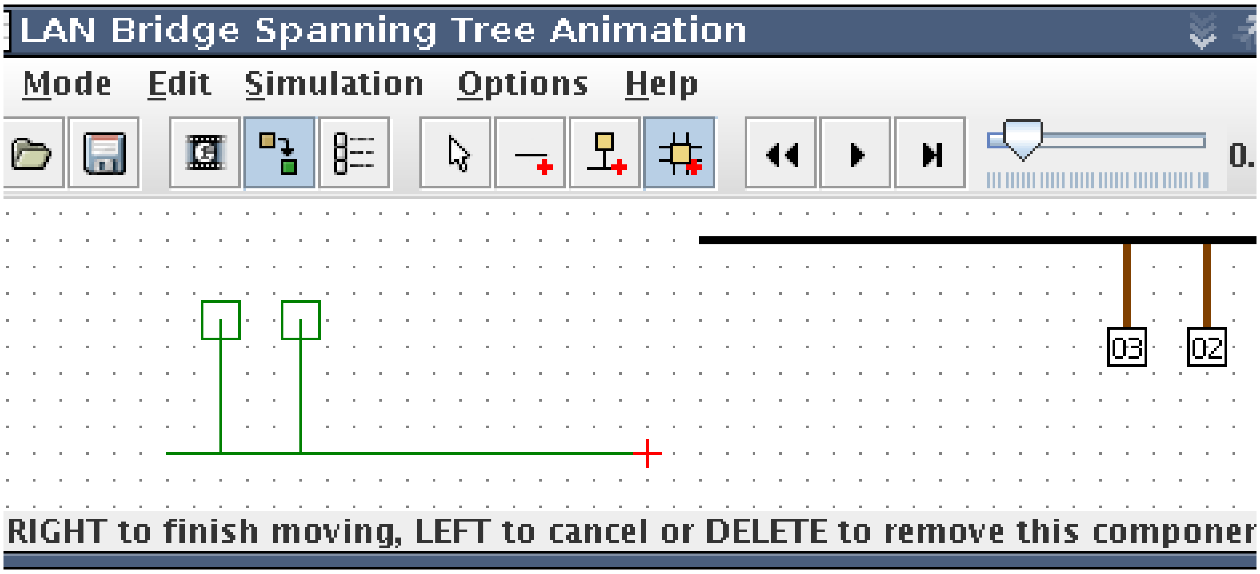

Figure 7.8: Step 3: Moving components if more space is required

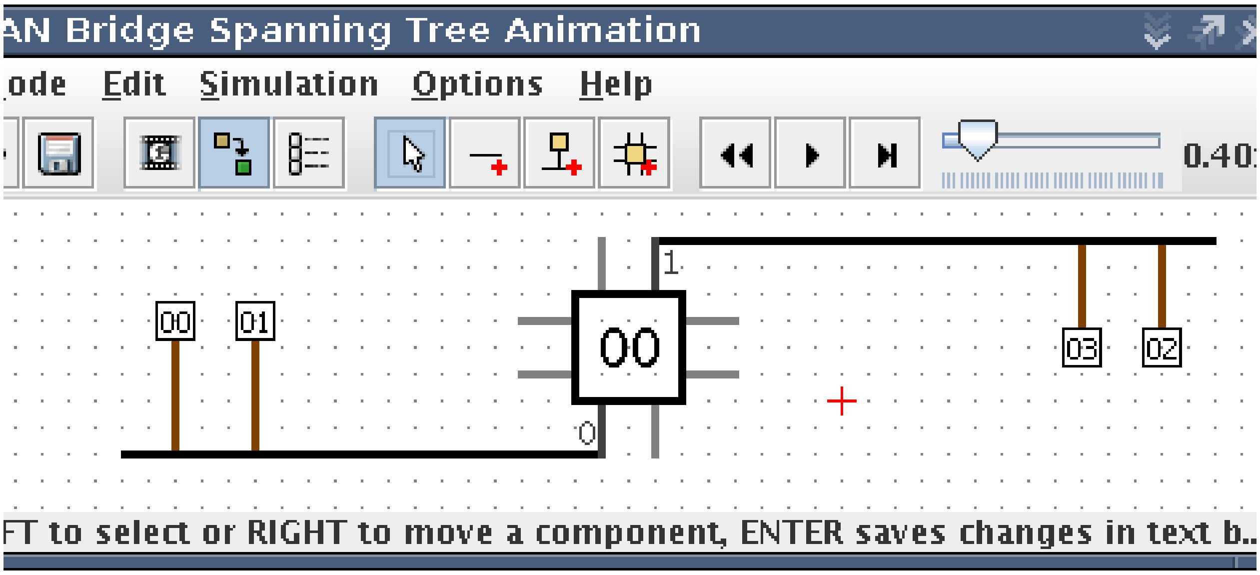

Figure 7.9: Step 4: Bridge created

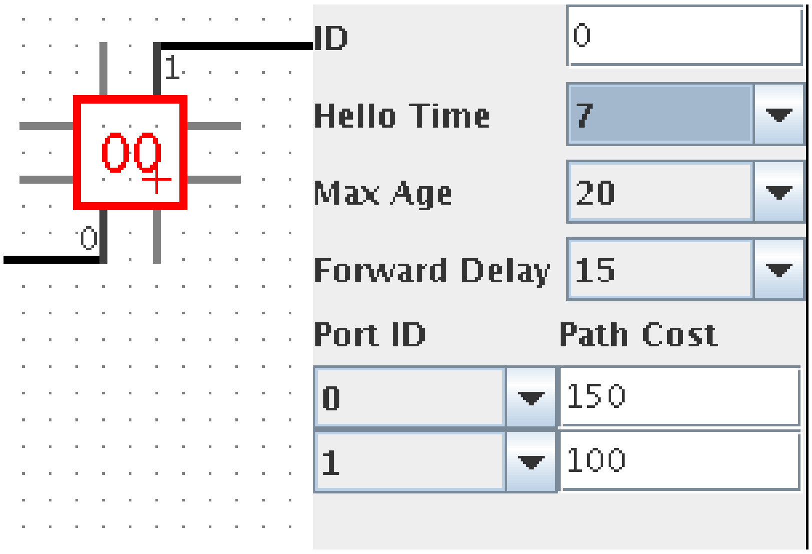

Figure 7.10: Step 5: Modifying the bridge parameters

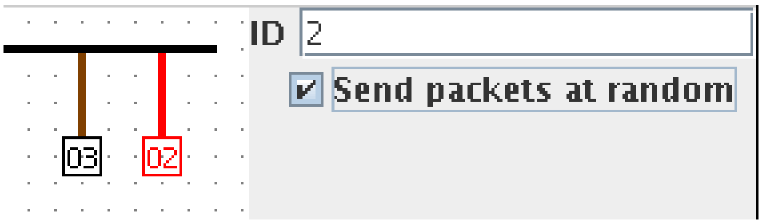

Figure 7.11: Step 6: Modifying a host's parameters

7.2 Simulation

Test Networks

Testing the network with one redundant link (Figure 7.12) and multiple redundant links (Figure 7.13) produce correct spanning trees.

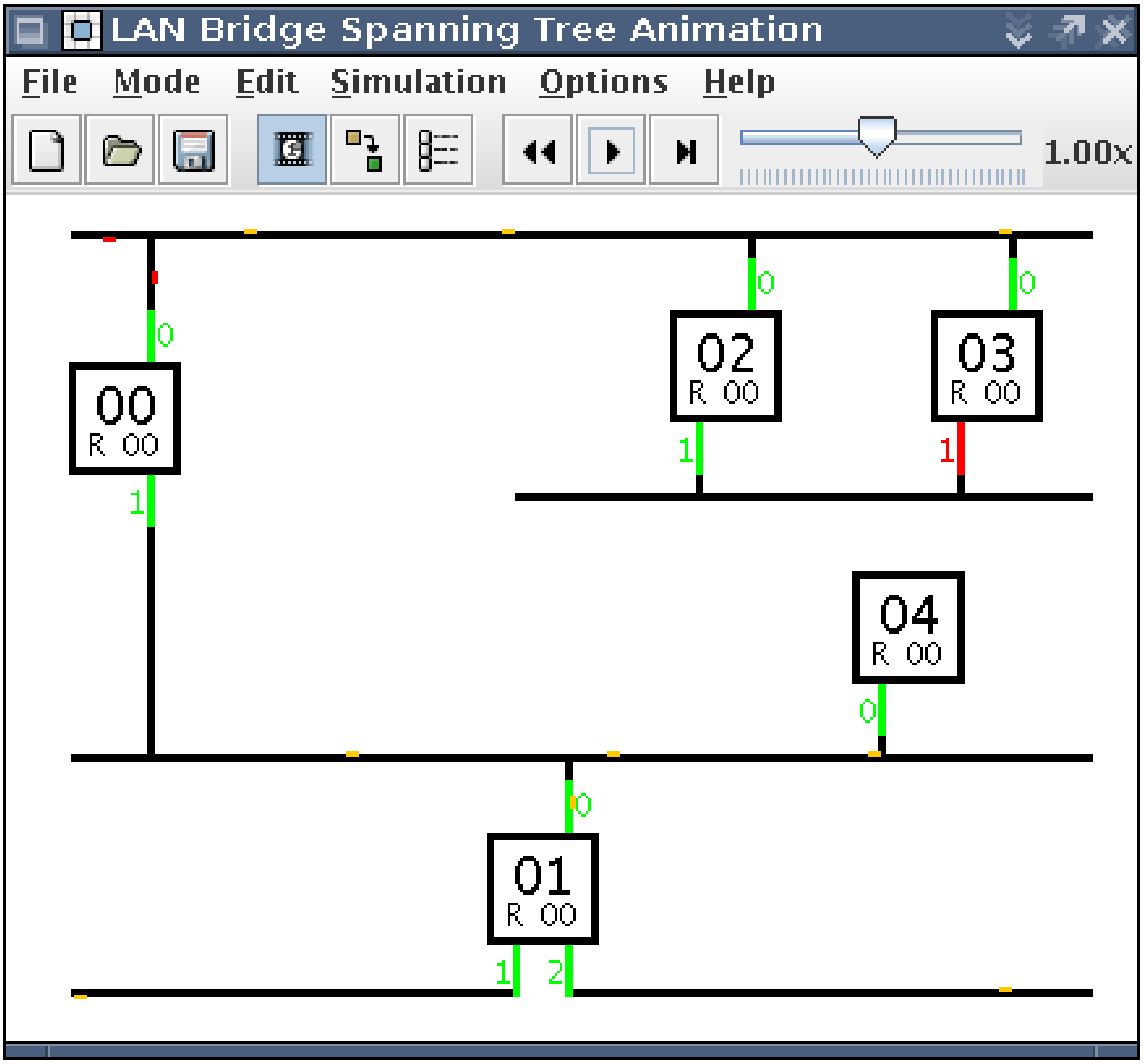

Figure 7.12: Network with a redundant link

Disabling a bridge is detected once the last message from the root has expired and the bridges renegotiate the root (Figure 7.14) correctly.

Two isolated networks can perform the spanning tree algorithm correctly amongst themselves (Figure 7.15).

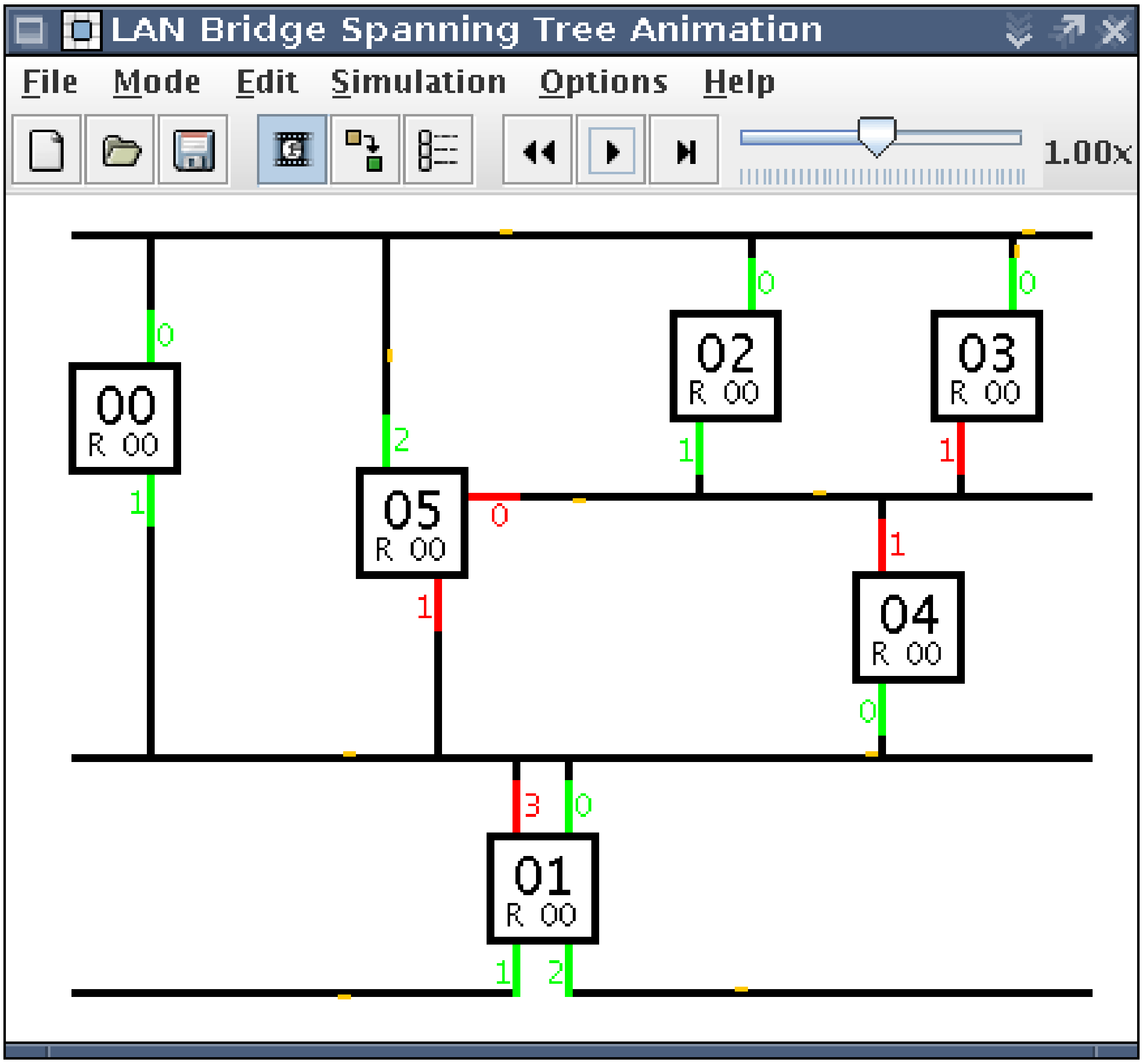

Figure 7.13: Network with multiple redundant links

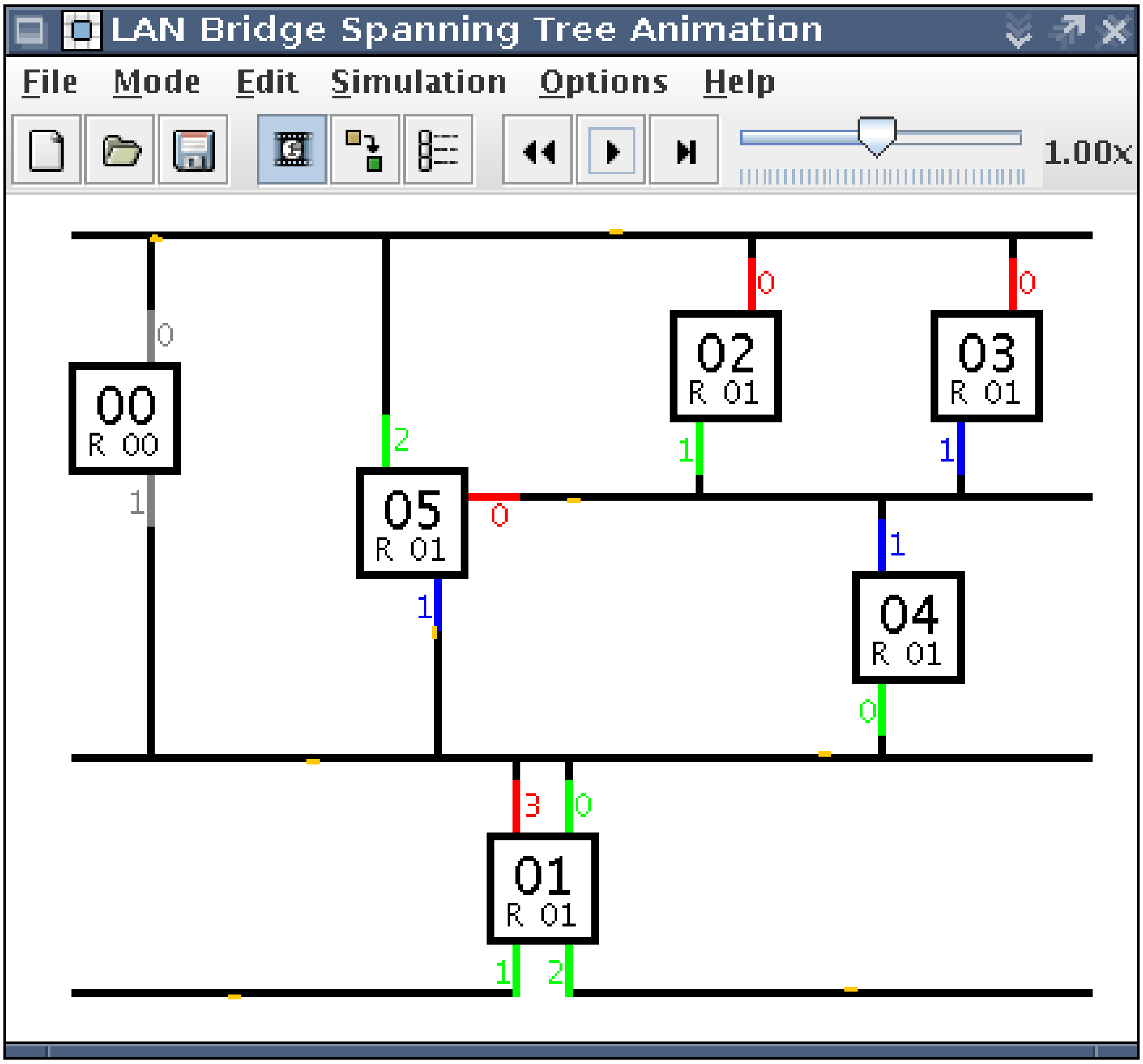

Figure 7.14: Network with multiple redundant links handling the root bridge being disabled

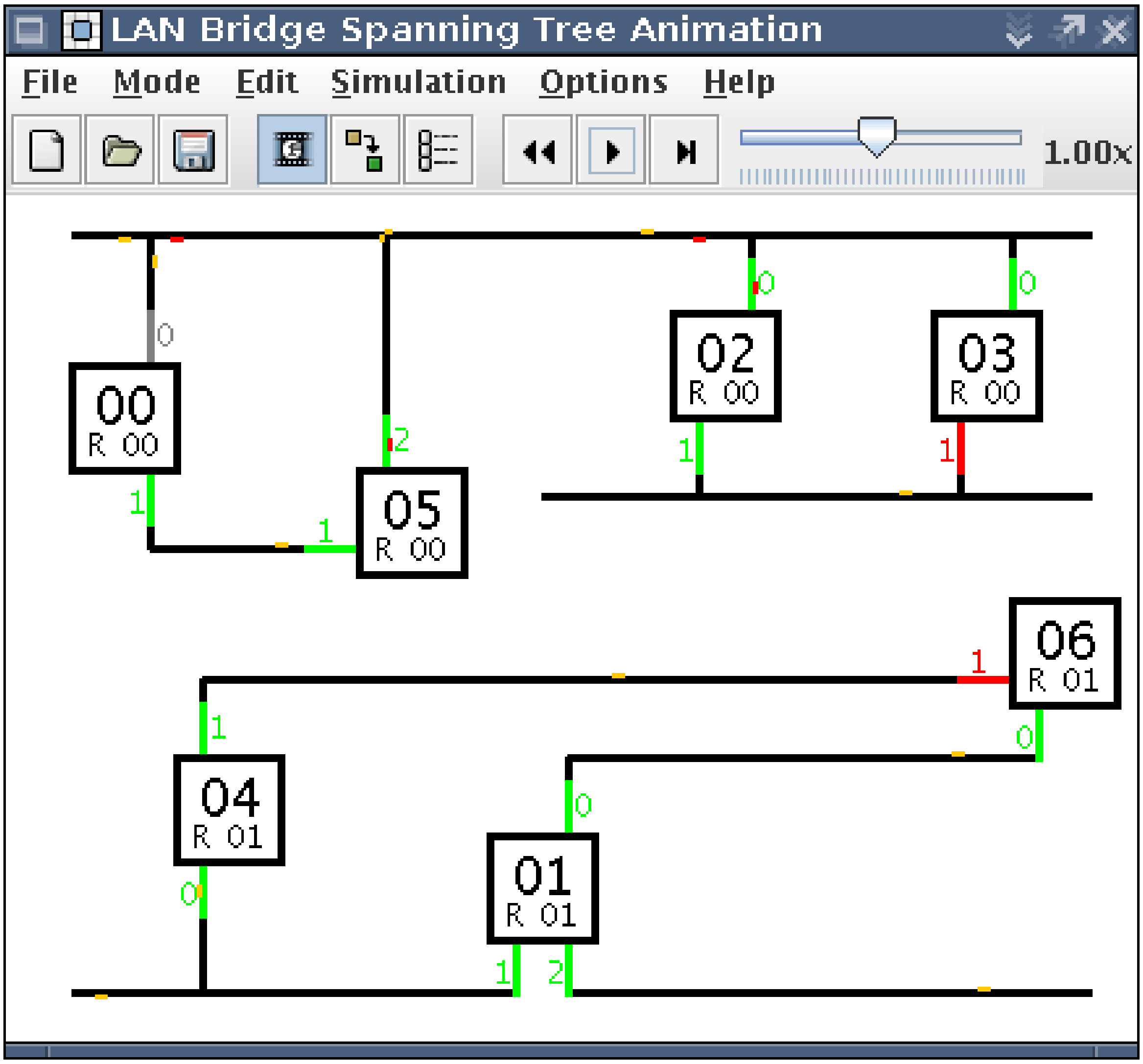

Figure 7.15: Two isolated networks with separate spanning trees

When a host moves to another part of the network the host cache is updated correctly (Figures 7.16 and 7.17). While the topology is considered changed but not acknowledged, the host cache expires twice as fast to handle the appearance of hosts moving because the spanning tree has changed.

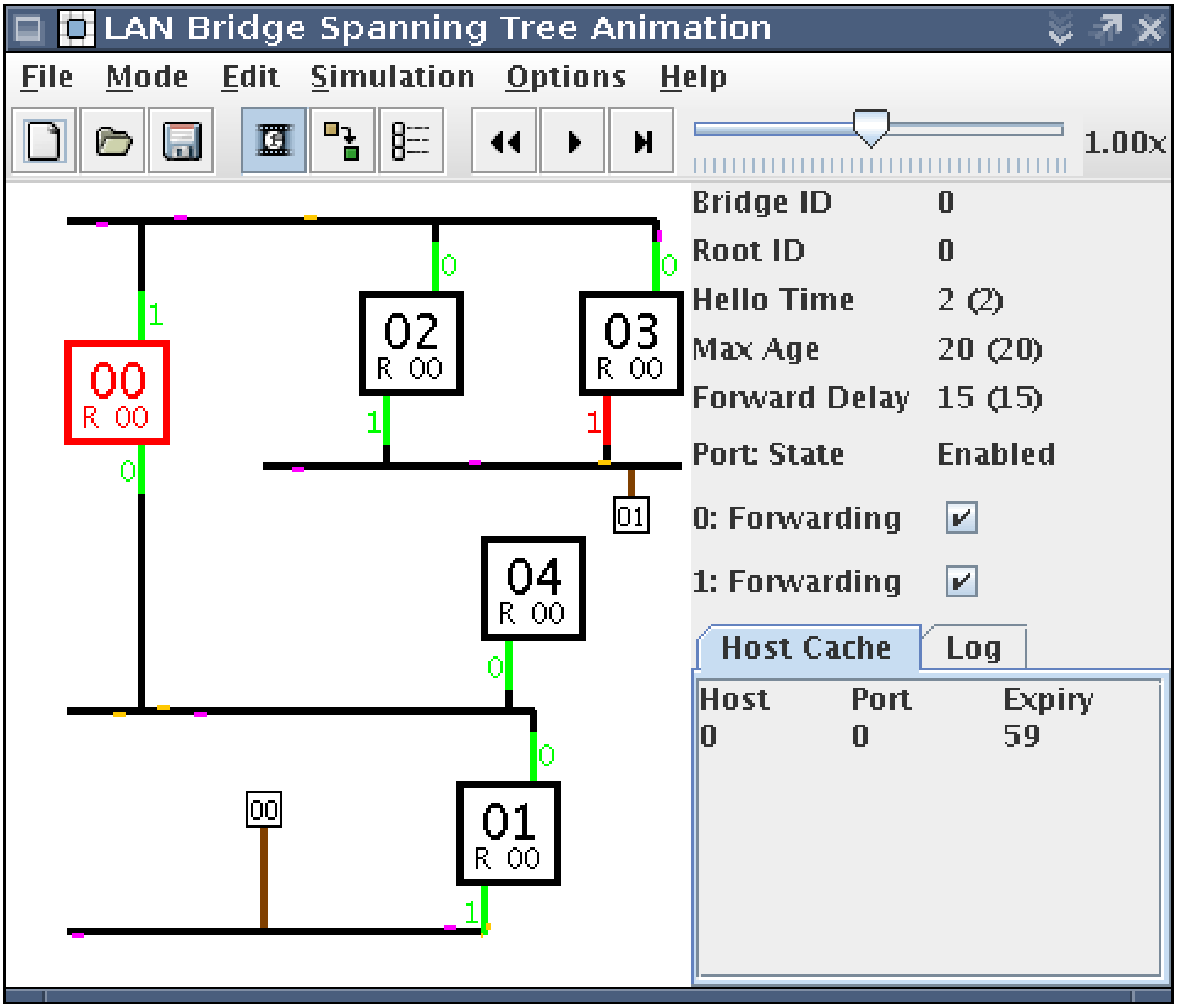

Figure 7.16: Host cache at bridge 0 containing host 0 seen from port 0

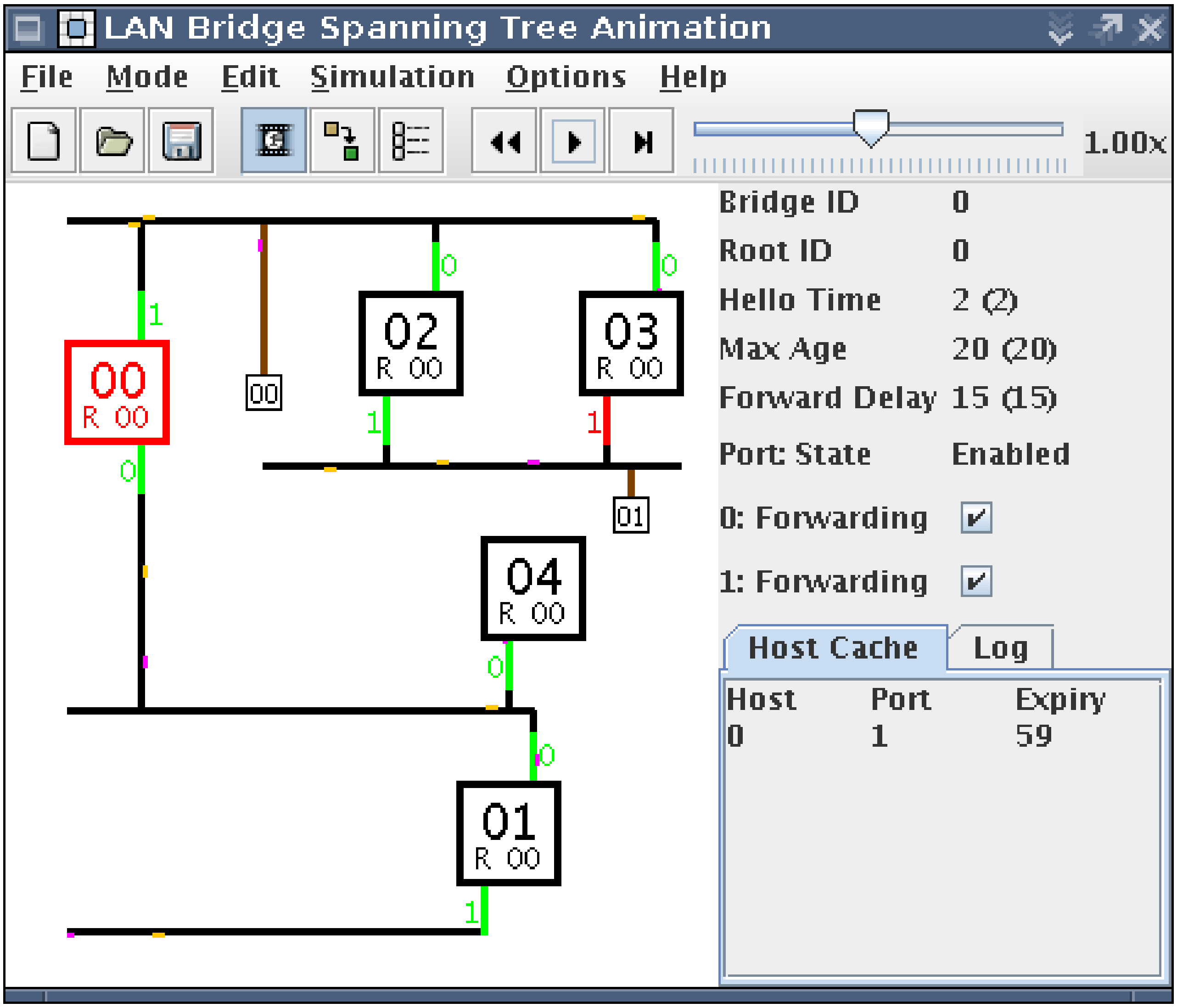

Figure 7.17: Host cache at bridge 0 containing host 0 seen from port 1

The protocol handler on the bridges updates the settings in use to be the same as the root bridge while retaining its own configuration should it become the root bridge as specified in the protocol standard.

User interface

The simulation interface correctly handles starting in and pausing in both normal and stepping mode, and switching between them while the simulation is running or paused. When the mode is changed away from the simulation view the simulation is automatically paused but will remember its state on returning to the simulation view.

7.3 Display

The display shows all types of component — LAN, bridge and host clearly in an uncluttered way without showing too much information at once (Figure 7.18) while still providing useful status information (such as the current root ID at each bridge). Packets are coloured to indicate their type (Figure 7.19) so that those related to the spanning tree protocol can be seen amongst those sent between hosts. Bridge ports change colour (Figure 7.19) to indicate their current state which makes it easy to see the spanning tree in effect.

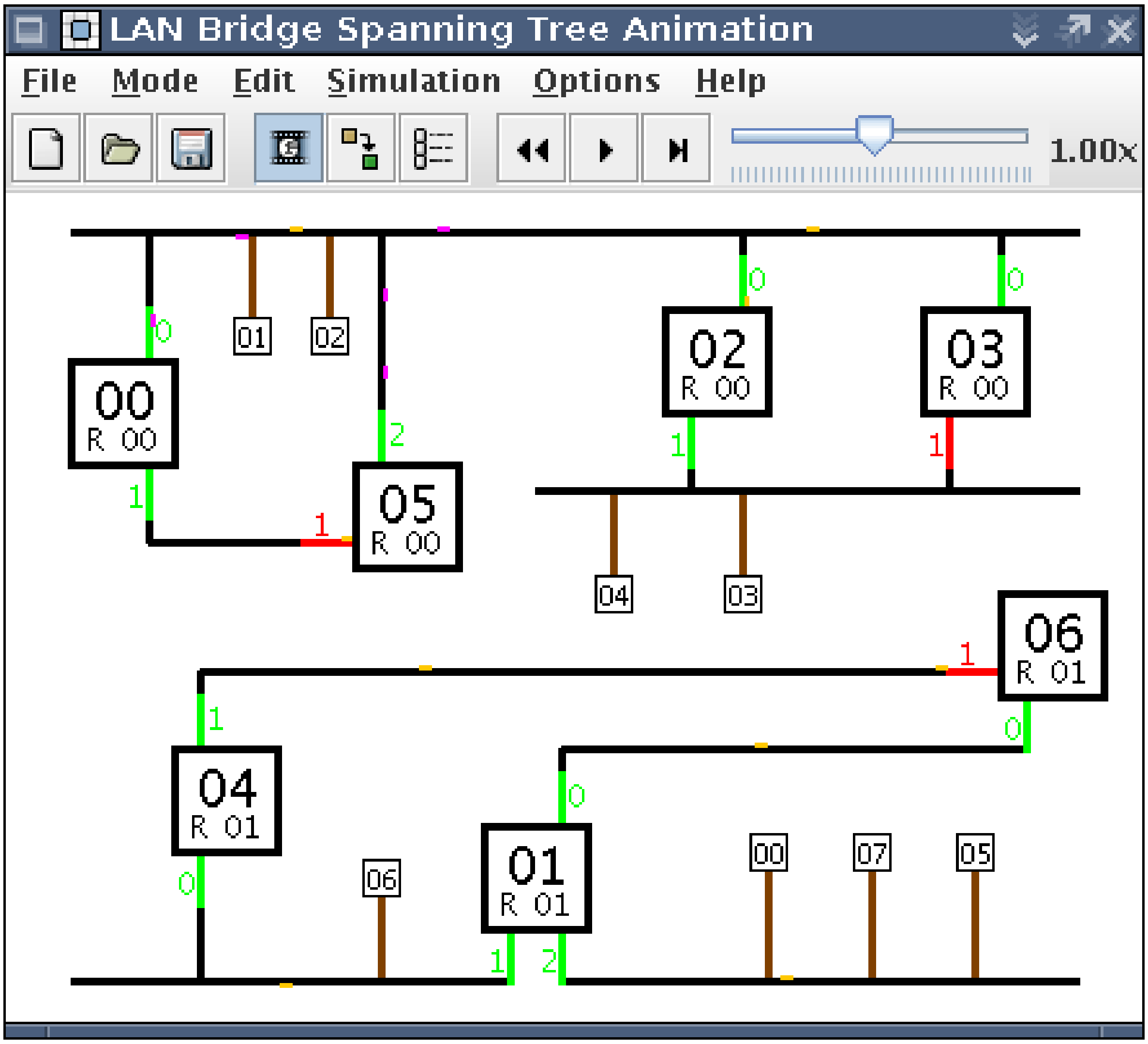

Figure 7.18: Display of simulation showing all components and packets on the network

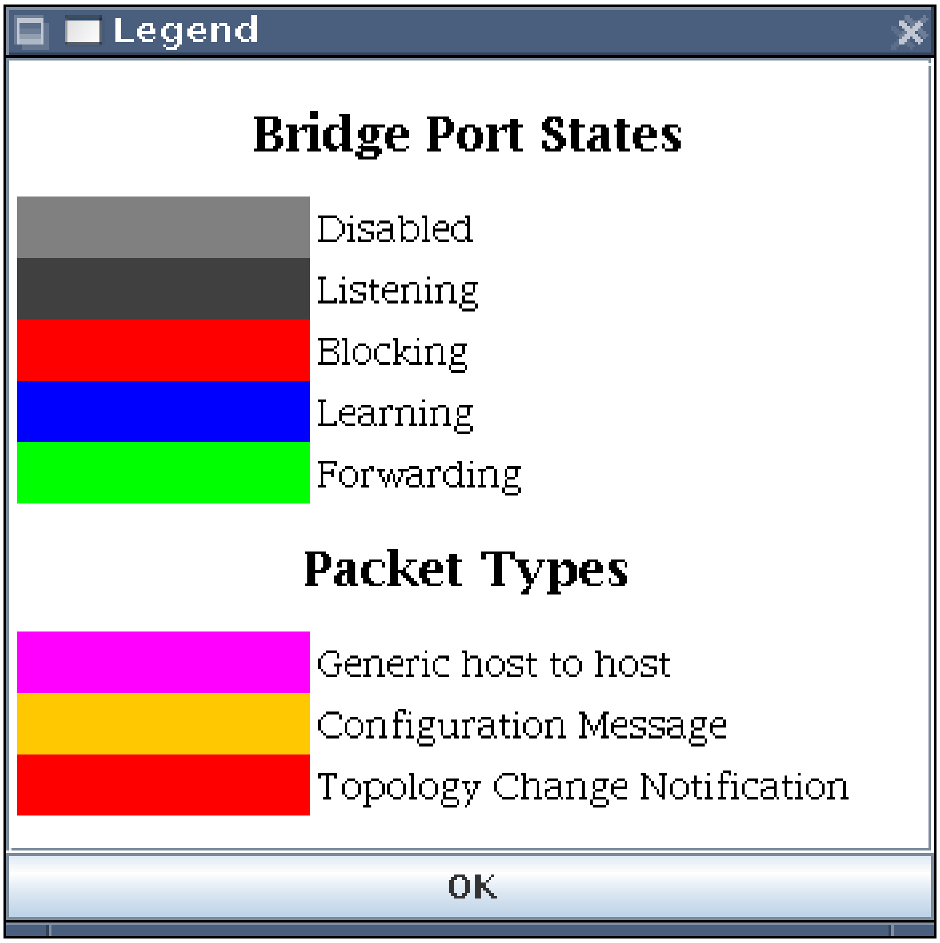

Figure 7.19: Legend of colours for the bridge ports and packet types

7.4 Scripting

The scripting system has a useful interface for creating and modifying scripts of practically unlimited length (Figure 7.20). Further screenshots of each script element type are in the user guide appendix.

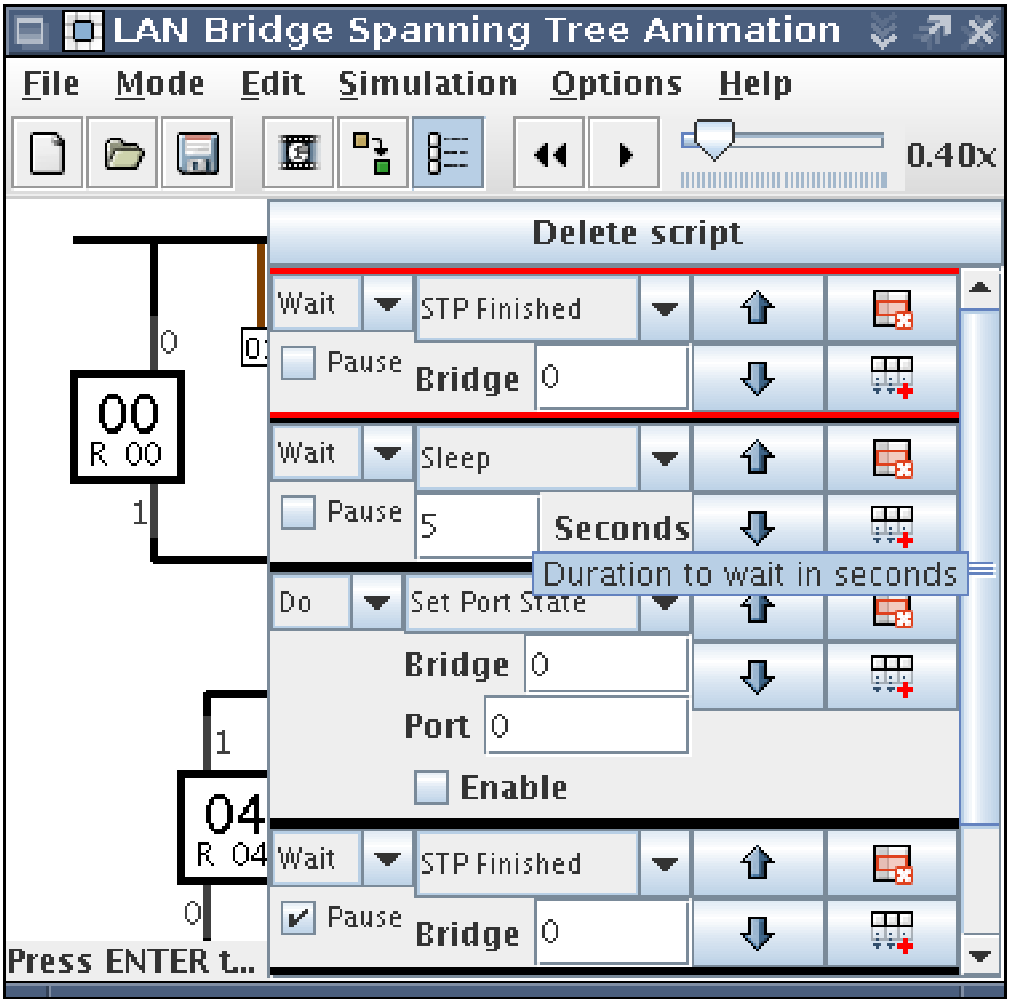

Figure 7.20: Example of a script for automating changes to the network settings

There are two types of script element which can be used, *states* that wait for events to occur and after optionally pausing continue with the rest of the script, and *actions* that perform an action on a network component or change a setting.

States

Spanning Tree Protocol Finished This checks that the spanning tree protocol is finished (all the bridges have the same root ID and all ports are in the forwarding or blocking states) for the part of the network connected to the specified bridge. Since a bridge ID has to be given, the state of different isolated networks can be independantly checked.

Port State This state is matched when the specified port on a specified bridge changes to the specified state (Disabled, Listening, Learning, Forwarding or Blocking).

Host Received Waits for the specified destination host to receive a packet from the specified source host. Such an event would indicate that the bridges are configured appropriately between the two hosts to allow them to communicate.

Root ID Waits for the root ID on the specified bridge to change to the given value.

Sleep Waits for the specified number of seconds (of network time).

Actions

Reset Bridges Resets all the bridges on the network.

Set Port State Enables or disables a port on the specified bridge.

Send Packet Generates a new packet from the specified source host to the specified destination host or all hosts.

Random Packets Enables or disables the random packet sending mode on the specified host.

Chapter 8

Discussion

8.1 Achievements

All of the requirements stated for the aims and objectives have been met except some which where too complicated to implement. The accessibility requirement mentions being able to select network components using only the keyboard but this would be too difficult to do because the focus could be anywhere when the user wants to change to another component, and the ordering would be arbitrary and potentially very confusing. It is still possible to start/pause the simulation using keyboard shortcuts. Originally the display would be able to show multiple bridge host caches but drawing the host cache onto the network display area itself would be too complicated, and the use of *always on top* windows is not very portable in Java and would cause problems when the main window is moved around.

Networks (including scripts) can be saved to and loaded from files stored in a human readable format (XML) and edited and configured on screen using the mouse at any time even while the simulation is paused. These modelled networks use broadcast LANs and support all of the relevant bridge properties for the demonstration of the spanning tree protocol. To make the process easier to understand packets destined for hosts will only be copied from the LAN to a connected host if that host is the destination — having all host packets broadcast would make it harder to see when the packet has reached its final destination.

The simulation itself supports varying the speed, multiple modes of operation (normal, stepping mode, and scripted mode), and pausing the process both manually and automatically upon switching to a different mode (network/script edit). Hosts can be configured to send out packets at random or send a packet to any or all hosts on request.

The status of bridges while running the simulation is both displayed on the view of the network (showing the root ID at each bridge and the state of the ports) and on a side panel that appears when a bridge is selected showing the host cache, log and other protocol-related settings.

Packets on the network are shown as rectangular blocks of colour moving along one side or the other of LANs, hosts and bridge ports. They are duplicated at connections between LANs and bridge ports since this is a simulation of a broadcast network. Where they start from hosts or bridge ports or reach a destnination host, bridge port or end of a LAN they are animated to enter and leave the network smoothly instead of simply disappearing.

The program is capable of running as a standalone application or under Java Web Start[1] and will automatically detect which environment is being used when starting so that it can run under the Web Start sandbox using the file opening/saving services.

8.2 Limitations

The spanning tree protocol defines a priority for both bridges and ports, but these are not neccessary in the simulation since there will be relatively few bridges and ports (some bridges have more than 256 ports[1]). The ID alone is adequate and is limited to 0–255 for bridges and 0–15 for ports. The protocol allows for up to 2^{64} bridge IDs and 2^{16} port IDs (inclusive of the priority). There is also a hard limit of 1000 on the number of hosts. All of these limits are set to appropriate levels for the display of the network — drawing a very large host address isn't possible in the space available. It is unlikely that a network simulation would reach these limits before the available screen area is exhausted.

[1] http://java.sun.com/products/javawebstart/

The requirements for the scripting mode state that the user should be able to select network components from the display when making up a script — this has not been implemented because tracking and handling the required focus changes would have been too complicated. It is not much of an inconvenience for the user to have to type in the values for bridge, port and host IDs.

The program is unable to detect when the user has the mouse buttons swapped so the instructions to press the left or right mouse button are only relevant to right-handed mouse configurations — when the buttons are swapped the input event received by Java is too. This is a minor problem since most programs always to refer to the mouse buttons as if the user is right-handed.

When running under Java Web Start, the maximum file size is limited to 256KB depending on the Java Web Start implementation used at runtime. It is possible to create a network which requires more than this to store.

8.3 Extensions and future work

The program is designed to keep each major section relatively separate, different classes manage the network components, display, simulation and script handling. The scripting and event tracking can easily be extended to implement new types of event, script state checks and actions. All of the source is well commented and includes *javadoc* comments to create per-class and per-method documentation (see the appendix for a sample of this).

Possible future work on the scripting ability could add the ability to make conditional statments, branching and loops in the script. The user interface for modifying scripts is good but could be improved.

A useful feature which the program lacks is the ability to undo/redo a change to the network configuration, this could make modifying the network simpler — if a mistake is made the change wouldn't have to manually be reverted. A variation on this would be the ability to run the whole simulation in reverse, either fully running the simulation backwards or just returning to a previous state. Both of these changes would currently require making complete copies of the network at every change, making the simulation performance poor.

Bibliography

[1] Radia Perlman. *Interconnections Second Edition*. Addison-Wesley, 2000.

[2] *ANSI/IEEE Std 802.1D, 1993 Edition*. ISO/IEC 10038 : 1993.

Chapter 9

Appendix - User Guide

(placeholder page 1 of 11)

(placeholder page 2 of 11)

(placeholder page 3 of 11)

(placeholder page 4 of 11)

(placeholder page 5 of 11)

(placeholder page 6 of 11)

(placeholder page 7 of 11)

(placeholder page 8 of 11)

(placeholder page 9 of 11)

(placeholder page 10 of 11)

(placeholder page 11 of 11)

Chapter 10

Appendix - Javadoc Summary

(placeholder page 1 of 2)

(placeholder page 2 of 2)