

https://dspace.org/

A computerised algorithm to optimise large, heterogeneous internets

Brandl, Marcus

2016-08-19
http://hdl.handle.net/10539/20914

Downloaded from a DSpace repository. https://dspace.org/

A COMPUTERISED ALGORITHM TO OPTIMISE LARGE, HETEROGENEOUS INTERNETS.

BY

Marcus Brandl

DECLARATION

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

M Braudl

(signature of candidate)

25 day of February 1994

A dissertation submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 1994.

ABSTRACT

The Spanning Tree Algorithm is considered to be one of the best algorithms for loop resolution in bridged internets, but it does nonetheless have its limitations. Bridges only have a local perspective of the internet which limits their effectiveness in setting up an optimal Spanning Tree. A computerised algorithm has been devised and implemented to view the internet on a global scale. It is demonstrated how a bad Spanning Tree configuration can result in increased delay and poor fault tolerance. The program provides valuable information to aid the network designer and administrator to set up and adjust the network to run at its maximum potential. Problems which can arise when a bridged internet is left to configure itself are also highlighted.

Table of Contents

1. INTRODUCTION 1
   1.1 Computer Communications 1
   1.2 Networks 1
   1.3 Optimised Configuration 2

2. BACKGROUND 7
   2.1 Internetworking Basics 7
      2.1.1 Repeaters 7
      2.1.2 Bridges 7
      2.1.3 Routers 7
      2.1.4 Gateways 8
   2.2 Literature Survey 8
   2.3 The Spanning Tree Algorithm 11
      2.3.1 Operation 11
      2.3.2 Determining the Root Bridge 11
      2.3.3 Configuring the Spanning Tree 12
      2.3.4 The Forwarding Table 14
      2.3.5 Frame Filtering 14
      2.3.6 Frame Forwarding 15
      2.3.7 Re-configuring the Tree 15
      2.3.8 Loosing the Root 16

3. A PROGRAM TO CONFIGURE THE SPANNING TREE 18
   3.1 Predicting the Spanning Tree Configuration . . . 18
   3.2 Determining the Optimal Root 18
   3.3 Re-configuration under Failure 19
   3.4 The Internet Matrix 20
   3.5 Path Determination 21
   3.6 Path Costs 22
   3.7 Ordering the Paths According to Costs 22
   3.8 Position of the Root 22
   3.9 Significance of the Relative Cost 23
   3.10 How the Level Representation Relates to the Cost 24
   3.11 Costs in Internets with Failed Nodes 24
   3.12 The Effect of Local Traffic 25

4. INTERNET TREE ALGORITHM (ITA) IMPLEMENTATION 26
   4.1 Definition of a Node 26
   4.2 Modelling a Multi-bridge LAN 26

5. TESTS AND RESULTS 28
   5.1 The Building Backbone 28
   5.2 Campus Backbone 31
   5.3 A 12 Node Internet 36
   5.4 The WAN Internet 36

6. CONCLUSIONS AND RECOMMENDATIONS 40
   6.1 Improved Design of Internets 40
   6.2 Future Developments 42

Appendix A

Program Modules 44

Appendix B

Using the Internet Tree Algorithm 53
B.1 Defining the Internet as an Internet Matrix . . 53
B.2 Editing the I.T.A. Configuration File 54
B.3 Program Restrictions 56

CHAPTER 1

1. INTRODUCTION

1.1 Computer Communications

The sharing of data between computers attached to different manufacturer's networks has been a requirement ever since people wanted to interchange data between processes. Network protocols were a propriety design and users were usually bound to one manufacturers networking products. This led people to develop various solutions provide a transparent interconnection between different types of networks. A transparent interconnection is a connection in which the end user is unaware that his network protocol has changed. One of the tools adopted by the ISO to aid this interconnection problem is the OSI protocol stack (Halsall, 1988). A first step in designing a protocol converter is to define each network protocol in terms of the OSI protocol stack before an internetwork can be set up. The differences between the protocols are the main functions for conversion.

1.2 Networks

Another requirement which leads to internetworking is as a result of the expansion of networks, i.e.the partitioning of a network into sub-networks to reduce congestion on the network. Too many

local packets on the network, especially those which are not destined for a certain portion of the network, can cause the packet delay to slow down unnecessarily. The introduction of a networking component that will partition a network may slow down packet interchange between the partitions but provide a higher degree of throughput to the work-stations on the individual partitions.

1.3 Optimised Configuration

The problem concerns the sub-optimal configuration of Spanning Trees and how they can be optimised. The thesis will propose ways and means of how to optimise a bridged internet. A major factor which is considered here is the effect of local traffic on LANs.

The paper by Harris and Brandl (1993a) forms the basis of this thesis. It outlines the problems and the importance of choosing the right configuration of a bridged internet. It proves that the a performance penalty is frequently paid if the Spanning Tree Algorithm is allowed to configure the tree unaided.

The greater the degree of mismatch between the networks, the more protocol conversion needs to be performed. The ISO specifications state that network components may be linked at layers 1, 2, 3, and 7 of the OSI 7 Layer Model (Stallings, 1990b). Network linking components have different names depending on which layer performs the linking between the networks.

The research focuses on the data link layer, i.e. layer 2 of the OSI 7 layer model (Schwartz, 1988). A bridge interconnects networks at this layer. A bridge will connect networks with different physical layer characteristics but a common data link layer. It is at this layer that the Spanning Tree Algorithm is implemented.

Loops in the internet may not exist as such. Only redundant links may be present for backup purposes. Routers and gateways operate at layers 3 and 7 respectively. They exchange routing packets with one another and therefore can avoid the development of loops in the internet (McQuillan, 1988). Repeaters, on the other hand, have to be physically installed in such a way that they do not form a loop. The existence of a loop will cause work-stations to crash because of duplicate packets.

When using a bridge to separate local traffic one cannot always keep an overview of their placement in an internet and it was found that they can unwittingly allow loops to exist in the internet. In some cases these loops provide a useful function in that they act as backup paths in case of failure of a bridge or network.

A local bridge has several applications:

- confine local traffic,
- isolate faults, and
- to link networks with different physical media.

In order to automatically resolve the loops created by backup paths, a protocol was developed which was simple enough to be able to function in the data link layer. The Spanning Tree Algorithm was designed to be a simple and effective protocol to form the bridged internet into a tree structure with each bridge blocking certain ports according to the algorithm in order to avoid loops in the internet.

Other loop resolution protocols exist, most notedly the Source Routing Algorithm (Hamner and Samsen, 1988) (Pitt and Winkler, 1987) developed by IBM for Token Ring. This protocol determines the shortest route through the internet for each packet it transmits. The advantage of Source Routing is that it is a dynamic algorithm but the overhead in network traffic and bridge/work-station resources makes it an unpopular choice. Because it is only implemented in a small number of bridges it is not considered in this thesis. It is mostly implemented in Token Ring bridges. Although recently bridges are available which will transparently bridge packets using the Spanning Tree Algorithm or the Source Routing Algorithm.

The Spanning Tree Algorithm is an excellent algorithm to prevent loops in a bridged internet, but certain limitations have been discovered which gave rise to this research. A full understanding of the workings of the algorithm can be used to improve the performance of the internet tree (refer to Section 2.3 which gives an explanation of the Spanning Tree Protocol).

The aim of the Spanning Tree Algorithm is to provide a simple, efficient protocol with little processor overhead. Certain bridge ports are blocked by the Spanning Tree Algorithm to avoid loops, therefore not all the packets can follow the shortest path to their destination but must follow the shape of the tree. This means that some packets will take longer to reach their destination than if they had taken the shortest path.

The two factors that affect the Spanning Tree's efficiency are:

1. the size of the tree, and
2. the placement of the root bridge.

The placement of the root bridge in the Spanning Tree is the most important factor because it will be root bridge's task to maintain the tree. Therefore the number of hops between the root and the end nodes or leaves of the tree determine show quickly a bridge at the leaves detect that there has been a topology change.

The Spanning Tree Algorithm does not take into account the LAN speeds, bridge speeds or data traffic density on the individual LANs in the internet. Only the Spanning Tree packets are slowed down by local LAN traffic, while travelling from source destination. Because there is no control over when the Spanning Tree Algorithm sets up the tree, an optimal tree cannot be guaranteed.

Another point which relates to the design of bridged internetworks. The placement of backup interconnections can

profoundly affect the shape of the tree if a backup bridge port must be activated.

When the Spanning Tree Algorithm activates a blocked bridge port, the newly formed tree should resemble the old tree as closely as possible to keep the average delay across the internet the same. The importance of each bridge is measured by the effect it would have on the average delay if that bridge were to fail.

The user can make adjustments to the configuration by changing the bridge identifier on the bridge in order to optimize the setup of the Spanning Tree. But this will still not tell us how efficiently the internet is operating. This dissertation proposes ways to optimize the configuration of a bridged internet and simulate the tree to show its efficiency.

A model with the following functions is developed and an implementation is demonstrated.

1. Determination of the optimal tree configuration.
2. Simulate increases in local LAN traffic.
3. Demonstrate the importance of backup paths.

CHAPTER 2

2. BACKGROUND

2.1 Internetworking Basics

2.1.1 Repeaters

A relay linking networks at the physical layer (layer 1) are referred to as repeaters. These are the simplest units in the internet work and their primary function is to amplify the electrical or optical signal that is naturally attenuated when travelling through a wire or optical fibre.

2.1.2 Bridges

A relay linking networks at the data link layer (layer 2) are termed bridges. The data link layer is further sub-divided into the MAC and LLC layers of which the latter forms the interface to the network layer. Bridges may link networks at either of these layers and are then referred to as MAC or LLC bridges.

2.1.3 Routers

A relay linking networks at layer 3, the network layer, are termed routers. Routers can make intelligent decisions as to the routes along which specific packets must be forwarded. These intelligent decisions are based on an information sharing protocol between the routers in an internet. Routers generally have a lower throughput than bridges.

2.1.4 Gateways

Gateways link networks at layer 7, the applications layer. Gateways link networks with completely different protocol stacks. Because gateways link two networks at the highest layer in the model, these units are complex, expensive and slow.

2.2 Literature Survey

In much of the literature surveyed, the problem of sub-optimal Spanning Tree configuration is not explicitly covered. Only a few papers indirectly touch on the subject when comparing Source Rooting and the Spanning Tree Algorithm such as that by Zhang (1988). Aspects such as the longer transmission delay and the use of sub-optimal paths in Spanning Tree internets and bridges are discussed.

The first article to hint at the fact that an improvement on the Spanning Tree, as configured by the algorithm, can be achieved is Hancock (1991). The improvement can be made by exercising care during configuration.

Hart (1988) addresses the problem of using the Spanning Tree Algorithm in remote bridges. He investigates how remote links degrade the reliability of the internet. The paper also introduces the concept of a path cost for a link. Unfortunately the paper does not apply its arguments to local bridges. The

concept of the root path cost can be directly applied to local bridges.

The paper by Chiarawongse, Srinivasan and Teorey (1988) proposes several models and sub-models for certain network types and links. It analysis network topologies from the LAN itself up to the physical layer of the network interface cards used. Other factors which the paper looks at is the

- the packet input rate,
- packet size distribution,
- number of nodes on a network,
- local/remote routing of packets, and
- flow control.

The networks and links modelled in this paper, such as the Apollo rings, are now considered old technology. The paper looks into integrating these technologies in a new campus backbone and is therefore trying to model them.

Kershenbaum et. al. propose a method which specifically uses a matrix representation to define their network on the MENTOR program (Kershenbaum, 1991). An algorithm is introduced which is used to optimise mesh topology networks. The paper looks mainly at packet switching networks as its base with routing elements at its nodes. The authors focus mainly on capacity planning as opposed to improving existing networks.

The algorithm is usable in a limited sense in internets using routers as the network nodes, and if the algorithm was to apply

to bridges then the entire algorithm would have to be implemented in all the bridging units in that network. Therefore the online operation of the algorithm is not a practical solution.

The proposed algorithm is of the computational complexity N^2 where N is the number of nodes in the network. It also takes into account the limitations of bandwidth of each link.

The limitations in the MENTOR model are apparent when Ethernet or other CSMA/CD (Carrier Sense Media Access with Collision Detection) physical layer protocol is applied to the model. Another problem is that the MENTOR model treats the links as though there is no other traffic on that link besides the routed traffic. This may not create so much of a problem on a token passing topology network but is of profound importance in CSMA/CD topology networks.

The paper by Harris (1991) suggests the use of an Internet Matrix to model an internet and defines the parameters that completely define each unit of that internet. It covers the cost of links failing and highlights a method of recognising critical links and units of the internet which would cause the greatest increase in internet delay, if allowed to fail.

2.3 The Spanning Tree Algorithm

2.3.1 Operation

The Spanning Tree Algorithm is implemented in MAC layer bridges. The bridges are important devices because the speed of a bridge can affect the transmission delay over the whole internets (Harris and Brandl, 1993b). Due to the position of the MAC layer in the OSI 7-layer model, it cannot perform any routing or congestion control. The IEEE Standard 802.1, Part D "MAC Bridges", defines the Spanning Tree Algorithm which enables the bridge to perform some traffic control functions to try to limit congestion and fault transmission on the internet (Stallings, 1990a). The standard states that only one active path may exist between any two stations in the internetwork.

The Spanning Tree Algorithm configures the internet into a tree which eliminates the possibility of a loop existing (Backes, 1988). Loops in the internet can result in packets being duplicated which can confuse the destination station. Circulating packets in the internet decrease the throughput of the internet by using network resources unnecessarily. At the base of the tree is a root bridge which is responsible for the maintenance of the tree. Each bridge has a bridge identifier and the bridge with the lowest bridge identifier is chosen as the root bridge.

2.3.2 Determining the Root Bridge

Each bridge has a unique identifier which distinguishes it from any other bridge in the internet. The root bridge is chosen to

be the one with the lowest bridge identifier. Incorporated in this identifier is also a priority field which can be preset to improve a bridge's chances of becoming the root bridge. The priority field, in effect, gives the bridge a lower identifier.

This bridge identifier is carried in a Bridge Protocol Data Unit (BPDU). There are essentially two types of BPDUs defined in the Spanning Tree Protocol, namely the Configuration BPDU and the Topology Change Notification BPDU. The formats are illustrated in Fig. 1. The BPDU carries the port identifier because there may be two or more ports on a bridge.

These BPDUs are transmitted on the LAN to which that port of the bridge is connected and is received by all the other bridge ports on that LAN. The BPDUs are replicated by all receiving bridges on all their outgoing ports to the other LANs. This replication is called flooding. When a bridge receives a BPDU whose bridge identifier is lower than its own, it will realize that there is a bridge with a superior claim to be the root bridge and cease transmitting BPDUs. The bridge does still continue to flood BPDUs from other bridges. As these BPDUs propagate through the internet, each bridge will know the lowest bridge identifier and this will be accepted as the root by all the bridges.

2.3.3 Configuring the Spanning Tree

If a bridge port is not in the blocked state then it is defined either as a **root port** or a **designated port**. The root port is defined as that port of a bridge which is closest to the root

bridge. Any bridge will only have one root port. The designated port is defined as that bridge port on a LAN which is found to provide the lowest cost path between that LAN and the root bridge. There can only be one designated port per LAN. Any bridge will have

- one root port, and
- one or more designated ports.

A bridge may also have one or more blocked ports.

Once the root has been established it will still continue to send out configuration BPDUs with the topology change flag set. Bridges receiving these BPDUs directly from the root bridge will automatically know their root port. The remaining bridge ports bid to become the designated port on their respective LANs. A bridge must know its root port before it can participate in the negotiation for the designated port. This is done by using the group address of the bridges on that LAN to ensure that BPDUs are addressed only to those ports attached to the LAN and are not flooded.

When a participating bridge receives a BPDU from a port with a lower root path cost it will cease to transmit BPDUs and sets its port status to blocking. A maximum age time is set by the root bridge. After the maximum age time has expired the port with the lowest root path cost will become the designated port for that LAN. All the unsuccessful bidders will have set their port status to blocking and those port that did not participate in the bidding become root ports and their bridges repeat the cycle.

When all the ports of all the bridges in the internet have been selected as either a root port, a designated port or a blocked port, normal data transfer can commence in the internet.

2.3.4 The Forwarding Table

The function of the forwarding data base, also commonly called the Source Table or the Forwarding Table, is to keep a list of the local stations' Ethernet addresses so that the local traffic can be confined to the source LAN. Each entry in the forwarding table also has an "age" associated with it which is aimed to make the table responsive to topology changes. This "age" is defined by maximum age value in the Configuration BPDU and is set by the root. When a port does not receive packets from a source for a certain amount of time, its entry in the forwarding table is considered "stale" and is removed from the table. The maximum age may be reduced by the root bridge via the BPDUs when it receives a topology change notification BPDU from some bridge.

Initially the forwarding table is empty. The addresses are only entered in the table as the bridges handle the data packets that are transmitted on the LANs. A separate forwarding table is kept for each bridge port.

2.3.5 Frame Filtering

To explain the operation of the forwarding table we will use the general case of a multi-port bridge. We will define the port on which the packet is received as the input port and the adjacent ports as the output ports. When the input port senses a packet

from the LAN, it will compare the destination address of that packet with the entries in its forwarding table. If the destination address matches one of the entries in the table it means that the destination station is on the LAN to which the input port is attached. The packet is then discarded and the operation is called packet filtering.

2.3.6 Frame Forwarding

If no match is found, the output port forwarding tables are screened to find a match for the destination address. If a match is found in one of the ports, the packet is forwarded on that port alone.

If again no match is found on the output port tables, it means that the bridge has not yet received a packet from that station or its entry has aged out. In this case the packet is forwarded by default.

When a packet is marked for forwarding after finding a match in a port forwarding table (this excludes forwarding by default), the source address is added to the forwarding table of the input port to indicate that the source station is in the direction of the input port. If an entry for that address already exists in the table then its age is reset.

2.3.7 Re-configuring the Tree

The root bridge periodically generates Configuration BPDUs on its ports and each bridge receiving this BPDU on its root port will

regenerate it on its designated ports thereby causing a cascading of BPDUs down the Spanning Tree. Bridges on a specific LAN will become aware of the fact if the bridge which owns the designated port breaks down (Backes, 1988). This could also point to the fact that the its root port LAN has a fault on it. The bridges then proceed as explained previously to bid for the new position of the designated port. If the fault is on the root port itself, the port will probably block and then the previous designated port could become a root port and continue to forward packets on the other designated ports which that bridge may have. Once the status of a bridge port has changed, it will send out a Topology Change Notification BPDU on its root port towards the root bridge. The root bridge in return will acknowledge the change by setting the Topology Change Acknowledge flag in the next couple of Configuration BPDU it sends out.

2.3.8 Loosing the Root

All ports on a LAN detect failure when the designated port for that LAN has fails to send a BPDU within the Maximum Age period. They will then negotiate to find a new designated port. The root ports which are one hop away from the root bridge will realize that the Spanning Tree no longer has a root and will start to generate BPDUs in order to determine a new root for the internet. In the mean time the negotiations at the lower levels will cease and those bridges too will begin to bid for the position of the root.

2.4 Optimisation

The Spanning Tree Algorithm sets up a unique tree which is determined by:

- the configuration, and
- the pattern of the identifiers.

Unless the configuration and the pattern of the identifiers are pre-determined in the design of the internet, the configuration of the Spanning Tree will be arbitrary. An arbitrary configuration may perform poorly, i.e. have a high average delay. It may also behave poorly under fault conditions. Chapter 3 discusses a model which enables an optimum configuration to be pre-determined under Spanning Tree.

CHAPTER 3

3. A PROGRAM TO CONFIGURE THE SPANNING TREE

3.1 Predicting the Spanning Tree Configuration

The selection of the root bridge by the Spanning Tree Algorithm depends explicitly on the relative magnitude **and** the location of the bridge in the configuration (see section 2.4). Selection can be influenced by the network supervisor by presetting the priority in the bridge identifier. Unfortunately the supervisor does not know if the selected root bridge is the optimal choice. Optimal means the tree configuration that will result in the lowest average delay for that internet. Optimisation can be achieved by placing the shortest paths, and having the fastest bridges and LANs, nearer the root. The following model will be able to determine the optimum configuration for the Spanning Tree.

3.2 Determining the Optimal Root

A model was developed to find the optimal root bridge. The implementation is a program called the Internet Tree Analyser (ITA). The ITA program takes into account the bridge and LAN speeds, and the topology of the bridged internet (Harris, 1991). The LAN and bridge speeds, in Mbps, are entered in the form of

an adjacency matrix with (m,m) defining the bridge specifications and (m,n) defining the LANs between the bridges. The program determines all the possible paths between the bridges and orders them according to a cost parameter. Several shortest path algorithms were considered before a simplified version of the Bellman - Ford Algorithm (Ephremides and Verdu, 1989) was selected to find all the paths. The cost is defined as the inverse of the speed and gives an indication of the response time. The path cost between any two bridges in the internet is the sum of all the costs of the bridges and LANs that lie in the path between these two bridges. The overall relative cost, which is defined as the sum of all the path costs between the root bridge and all the other bridges in the internet, is used to determine the optimal spanning tree. The bridge which gives the minimum overall path cost, is selected as the root bridge position. The choice of this bridge as the root bridge will ensure that the Spanning Tree is optimally balanced.

3.3 Re-configuration under Failure

The model determines the cost of the Spanning Tree having to re-configure due to a fault. This is done by determining the optimum root and configuration for each failed link separately. This includes the root bridge. The average of the overall relative costs, relative to the Spanning Tree under normal conditions, will be an indication of the re-configurability of the internet under fault conditions.

Since the effect of traffic on the configuration can be significant, a slow LAN with little traffic may offer a greater bandwidth to remote traffic than a fast LAN carrying heavy traffic. The traffic factor is used to determine the overall relative cost.

3.4 The Internet Matrix

The internet matrix represents the configuration of the internet in a form which is easily processed by a computer. The matrix is an N by N matrix where N represents the number of nodes, i.e. bridges. There are two types of entries in the matrix, i.e. the (m,m) and the (m,n) entries. The (n,m) entry and the (m,n) entry are identical because bridge delays are the same whether the packet is transmitted from LAN A to LAN B or vice versa, and therefore the parameters in the matrix may be reflected about the main diagonal.

The point (m,m) on the matrix will define the bridges in the internet by their speeds in terms of frames per second. This is determined by the forwarding rate of the bridge which is internally converted, in the program, to the bit rate in Mega bits per second. The points (m,n) on the matrix represent the LAN or WAN connections between the bridges. Here we define the LAN/WAN speed in bits per second and the local traffic loading factor (if applicable). The loading factor may be defined as that fraction of the bandwidth which is used, on average, by the local

traffic. The bridges in the matrix have an advantage over the links in that they may link to various LANs/WANs, whereas the linking LANs/WANs may only link two bridges. This calls for some modelling to overcome this limitation in the program (see section 4.2).

3.5 Path Determination

To determine the most cost effective path between two bridges all paths must be known.

The paths are calculated using a recursive algorithm. The two nodes at either end are defined as the source and destination bridges. The algorithm finds all the paths to the next nearest bridges. The algorithm is called recursively for each next nearest bridge with that next nearest bridge becoming the source to find a path to the specified destination. The destination bridge remains the same. This algorithm is repeated until it has found all the possible paths. To explain the path determination algorithm, assume that bridge A is the source and bridge Z is the destination. The next nearest bridge to A is found to be F. Now F becomes the source and Z the destination and the algorithm is called again. When all the routes between F and Z have been determined, the algorithm returns to finding all the routes between A and Z. This ensures that all the Paths are determined.

3.6 Path Costs

The **definition of cost** is that it is the **inverse of the speed of the LAN or bridge.** Fast LANs have a high bandwidth and a low cost. This allows us to define the **path cost** as being the sum of the **costs of all the nodes and links in the path** including the source and destination nodes. The path cost is an indication of the time delay a packet will experience in travelling between the source and the destination. A proper time delay can only be determined by taking slowest link and queuing factors into account.

3.7 Ordering the Paths According to Costs

Once all the path costs have been calculated the paths are analyzed to find the shortest path between all the nodes in the internet. The path costs for each source - destination pair are therefore ordered in memory from the lowest to the highest cost path. The path with the lowest cost in each source-destination pair is termed the **Optimal Path** and its cost is termed the **Optimal Path Cost** for that source - destination pair.

3.8 Position of the Root

The **Balance Index** is defined as the sum of all the optimal path costs between the root and all the other nodes. The overall

$$ORC = \sum_{x=1}^{A} OPC_x$$

Matrix 3.1

ORC = Overall Relative Cost
OPC = Optimal Path Cost

relative cost, of the Spanning Tree is the total of all the optimal path costs between the root and all the other nodes as shown in Matrix 3.1. The position of the root bridge is determined by the root which has the lowest optimal path cost.

Each node in turn is taken as the root node and their overall relative costs are compared. The root bridge which produces the lowest relative cost will obviously be the best choice for the job of root.

3.9 Significance of the Relative Cost

The relative cost is effective in finding the root because the relative cost is obtained over the whole internet and every configuration is determined. This gives an indication of the size of the tree, in particular when it comes to transmitting BPDUs through the tree. Therefore that bridge that has the smallest relative cost must also be the one with lowest levels (refer to the level representation of a Spanning Tree in section 3.10). The tree will then only have short branches.

3.10 How the Level Representation Relates to the Cost

The higher the level of the tree, the further the root is from the n'th level leaf and therefore the response time for the communication between two leaves of the spanning tree becomes longer. This is an important issue when the root has to send out Topology Change Notification BPDUs to inform the tree components that some component has failed. The greater the distance between the root bridge and end node bridges, the longer the delay for a Topology Change Notification BPDU in reaching the end nodes. Until the BPDU arrives the end nodes will attempt to send packets via the old Spanning Tree links.

3.11 Costs in Internets with Failed Nodes

The **Cost of Failure** is defined as the ratio of the optimal Overall Relative Cost to the next most optimal Overall Relative Cost, expressed as an average over all the connections, i.e. this is done by calculating the new Overall Relative Cost as each component in the internet fails. This assumes that for each source and destination there is a backup path otherwise the cost of failure is infinitive because some part of the internet will be isolated by the failure of that critical node.

The cost of failure of an internet gives an indication of how effectively the internet configuration can recover from the failure of a LAN or bridge in the Spanning Tree. When a bridge,

bridge port or even a whole LAN fails, the Spanning Tree Algorithm must negotiate a new tree structure which by-passes the failed component. When all the components of the internet are functional, the optimum Spanning Tree is configured by the algorithm and therefore when one of the components fails, a sub-optimal tree results. This means that the Overall Relative Cost of the sub-optimal tree will be higher than that of the optimal tree.

3.12 The Effect of Local Traffic

Expected Traffic is the ratio of the LAN's bandwidth which will be occupied by local traffic. This means that as the local traffic on a LAN increases, the longer the transmission delay will be in traversing that LAN and so the **Path Cost** will increase in terms of our program. The program will calculate the ideal Spanning Tree and attempt to place heavily loaded LANs nearer the leaves of the Spanning Tree. This will avoid the use of heavily loaded LANs as transit LANs for remote traffic in the internet.

CHAPTER 4

4. INTERNET TREE ALGORITHM (ITA) IMPLEMENTATION

4.1 Definition of a Node

A limitation of the program is that a LAN may have only two bridges connected, whereas a bridge may have multiple LANs to which it attaches. Although this is the opposite to what is found in the real world where one is more likely to have 2-port bridges and LANs with multiple bridges on it, it does simplify the process of determining the root bridge.

4.2 Modelling a Multi-bridge LAN

Consider as an example a LAN with three bridges attached to it (see Fig 5.1). Because one cannot have a multi-connected LAN in the ITA program, it must be modelled using a multi-connected bridge instead, as depicted in Fig 5.2. Due to the program structure, each bridge and LAN must have a cost associated with it. This requires some adjustments to the speed of the affected LANs and the introduced bridge to minimise the effect of this bridge.

Consider again the equation for the LAN and bridge costs. To minimise the cost of the bridge, it should have an infinitely

High bandwidth for a negligible cost. Therefore when such a representative bridge is introduced in the Internet Matrix as a '0' in its (m,m) position, and the program sets its speed to 500,000 packets per second.

The introduction of the extra LANs has the result that a packet must traverse two LANs when travelling via the representative bridge. This means that the costs of each LAN must be half of what it would have been for the original LAN. From the equation it can be seen that by doubling the bandwidth, the cost is effectively halved. This doubling of the bandwidth is done automatically by the program when the representative bridge has been defined.

CHAPTER 5

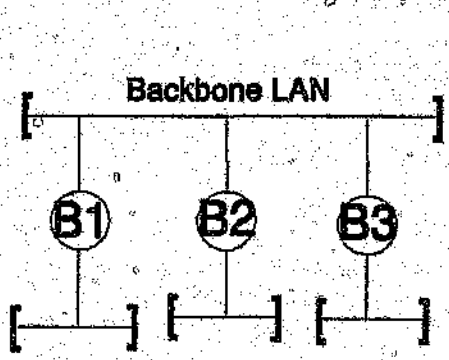
5. TESTS AND RESULTS

Examples of bridged LANs are represented in this chapter to illustrate the use of the ITA program. WAN links are assumed to be dedicated links. Dial-up links can not be part of the Spanning Tree because the Algorithm would have to be recalculated after each connection/disconnection of the dial-up link.

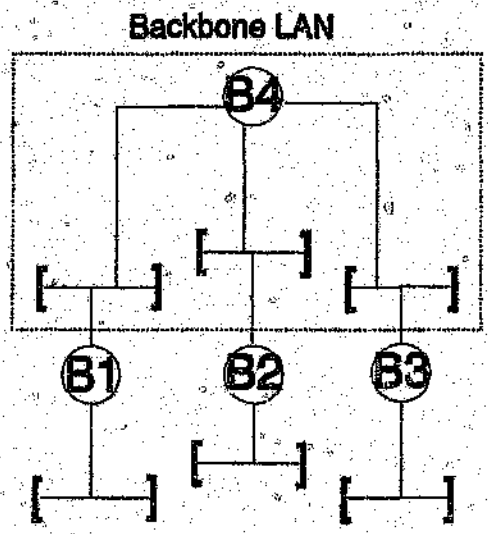
5.1 The Building Backbone

Consider example 1, a simple internet with 3 LANs bridged via a backbone as shown in Figure 5.1(a). This setup would typically be found in the case of a building backbone. Each of the LANs are Ethernet LANs, i.e. 10 Mbps. The bridges have a Forwarding Rate (FORRATE) of 10,000 pps. The local traffic utilization of the LANs is assumed to be an average of about 5%. The bridge identifiers are arbitrarily selected as shown in Figure 5.1.

The backbone LAN has more than two bridges attached to it, therefore it must be modelled as a multi-port bridge as explained in section 4.2. This results in the in an internetwork as shown in Figure 5.1(b).Matrix 5.1 shows that the bridge FORRATEs are 10,000 pps and the backbone LAN speed has been doubled to compensate for the introduction of the dummy bridge. The dummy



(a)



(b)

Figure 5.1 The Building Backbone Internet.

```
0.1  0   0  20(0.05)
     0.1 0  20(0.05)
         0.1 20(0.05)
             0
```

Matrix 5.1

bridge, which is defined by (4,4) in matrix 5.1, is represented by a "0" in the matrix to indicate its role in the internet calculation.

```
0.1    0       0    20(0.05)
0.1    0    20(0.05)
0.1 20(0.05)
 0
```

Matrix 5.2

The values are entered in the internet configuration file as shown in Matrix 5.2 and the program is executed. All the real bridges have an equal claim to be the root, except of course the dummy bridge. The bridge with the lowest bridge identifier (in our case we defined bridge B1 to be the bridge with the lowest bridge identifier) will simply become the root of the Spanning Tree. In this and future examples we will assume that B1 has a lower bridge identifier than B2 and that B2 has a lower identifier than B3, etc. Figure 5.2 shows the Spanning Tree configuration.

Although the back-up analysis is trivial, it shows that bridge B4 is the critical component in this internet. This means that the backbone LAN is the vital component of the internet which can cripple the whole internet because it has no back-up facilities.

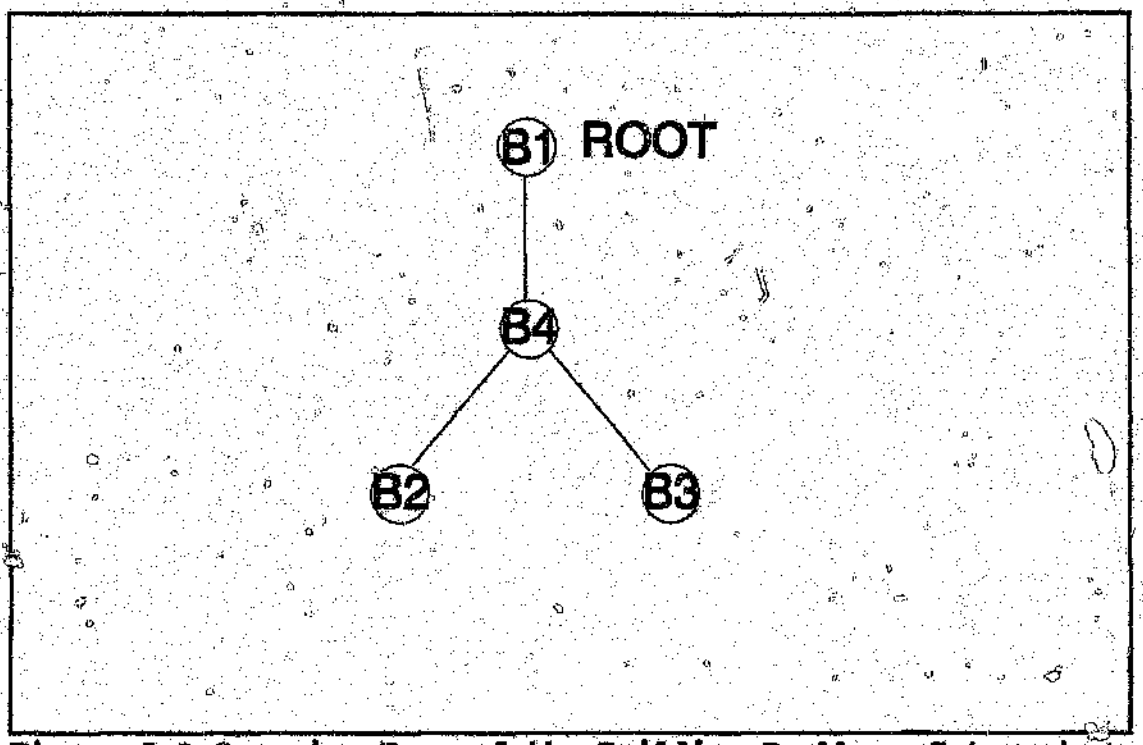


Figure 5.2 Spanning Tree of the Building Backbone Internet.

5.2 Campus Backbone

Consider an 8 LAN internet linked by means of bridges. The LANs are connected by means of 4 3-port bridges and 4 2-port bridges. All the bridges used have a speed of 5000 pps and the LANs are Ethernet (10 Mbps). Different topologies of the same internetwork are discussed to show the improvement in the internet and to illustrate some principles of how to optimise the internet.

Figure 5.3(a) shows the basic configuration with a simple set of backup possibilities built in. It is represented in Matrix 5.3. The overall relative cost of choosing each bridge in turn to be the root is tabulated in Table 5.1.

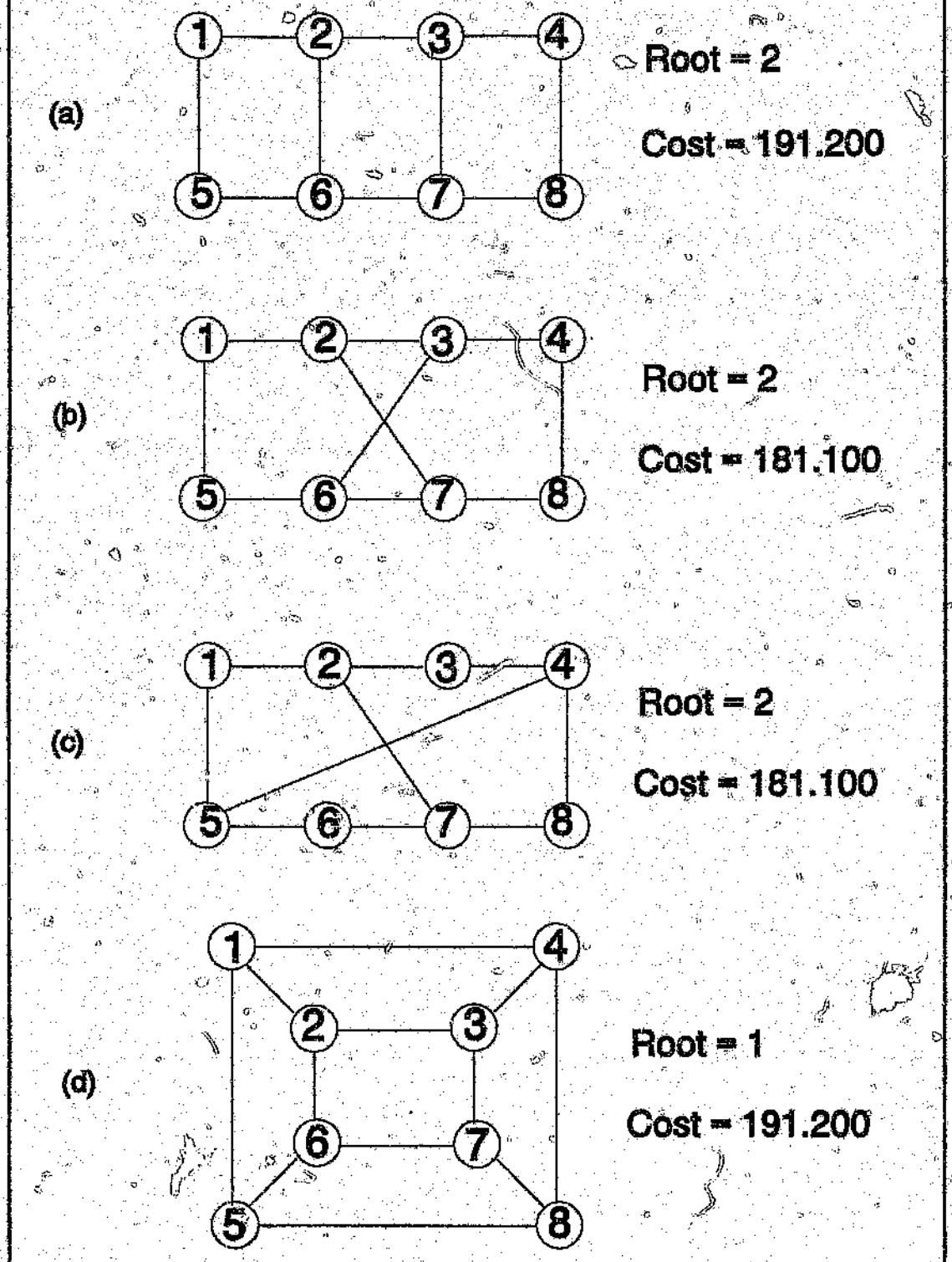


Figure 5.3 Various Topologies of an 8-bridge Internetwork

```
5000 10(0.1)  0  0 10(0.1)  0  0  0
5000 10(0.1)  0  0 10(0.1)  0  0
5000 10(0.1)  0  0 10(0.1)  0
5000    0     0  0 10(0.1)
5000 10(0.1)  0  0
5000 10(0.1)  0
5000 10(0.1)
5000
```

Matrix 5.3

The Balance Index calculations show bridge B2 to be in the best position for the root, as depicted in Figure 5.4(a). In actual fact, any one of bridges B2, B3, B6 or B7 could equally well be the root. Bridge B2 was chosen because it had the lowest bridge identifier. If any of the other bridges B1, B4, B5 or B8 had haphazardly been chosen as the root, the BI would have dropped by 21%, i.e. if bridge B2 is the root the B.I. is 191.2, whereas if bridge B2 is the root the B.I. increases to 231.6 (see Table 5.1). The maximum delay cost in the tree is 2.63. In practice, an increase in the BI means that the average delay through the Spanning Tree increases. Under heavy loading the number of retransmissions due to time-outs will increase.

If the configuration of the internet is slightly altered as in example 3, the efficiency, in terms of the average delay, can be improved. See Figures 5.3(b) and 5.4(b). It was found that example 3 provided a 5% *decrease* in the BI over the configuration in example 2 and the maximum delay cost has *reduced* from 2.63 to 2.18 or by 17.1%. This was achieved by rearranging the configuration of the bridges.

Example 4 is another configuration of the same internet but it

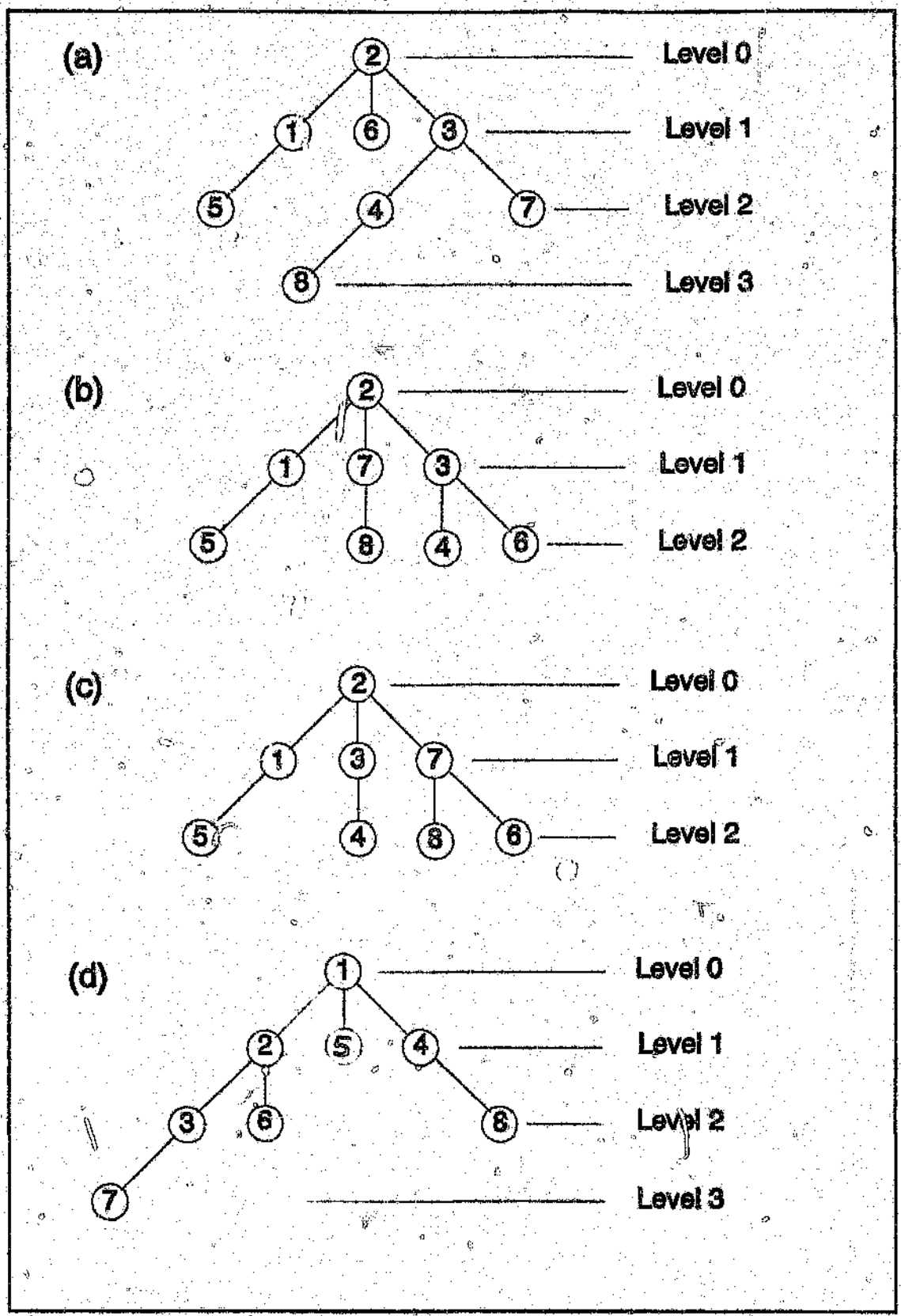


Figure 5.4 Spanning Trees for examples 2 - 5.

yields the same BI and average delay cost. See Figures 5.3(c) and 5.4(c).

The configuration in example 5 is the same internet as in example 2 except that it has two extra backup paths. See Figures 5.3(d) and 5.4(d). The additional links do not improve the BI, but it will remain the same which ever node is selected as the root. It is interesting to note that although the internet would be more reliable, in terms of its Backup Cost, than the previous configurations, it does *sacrifice an increase* of 5% in the BI when compared to the previous configurations.

Table 5.1

	Tree	Balance	Index	
Root Bridge	Example 2	Example 3	Example 4	Example 5
1.	231.6	211.4	201.3	191.2
2.	191.2	181.1	181.1	191.2
3.	191.2	181.1	201.3	191.2
4.	231.6	211.4	181.1	191.2
5.	231.6	211.4	181.1	191.2
6.	191.2	181.1	201.3	191.2
7.	191.2	181.1	181.1	191.2
8.	231.6	211.4	201.3	191.2

5.3 A 12 Node Internet

Example 6, Figure 5.5(a), shows a 12 bridge internet with internet components of different speeds. The bridges have a speed of 100 kbps and the LANs are 10 Mbps, unless explicitly stated otherwise in Figure 5.5(a). The 4-level, optimal Spanning Tree produced by the ITA program is shown in Figure 5.5(b). The slower bridges and LANs are generally be placed near the leaves of the Spanning Tree and act primarily as backup routes. The results of the ITA program are tabulated in Table 5.2.

Table 5.2

Root Bridge	B.I.	Root Bridge	B.I.
1	437.1	7	646.6
2	394.6	8	415.1
3	495.5	9	478.4
4	393.7	10	495.6
5	373.8	11	479.6
6	544.6	12	457.6

5.4 The WAN Internet

In a WAN internet backup links provide resilience when the main link fails. The backup links are usually have a slower speed which provides a path for critical applications. The low-speed link can affect the average delay of the whole internet.

Figure 5.6, shows two sites linked by a 2 Mbps WAN link and a 9600 bps back-up WAN link. Assume that the local traffic on the

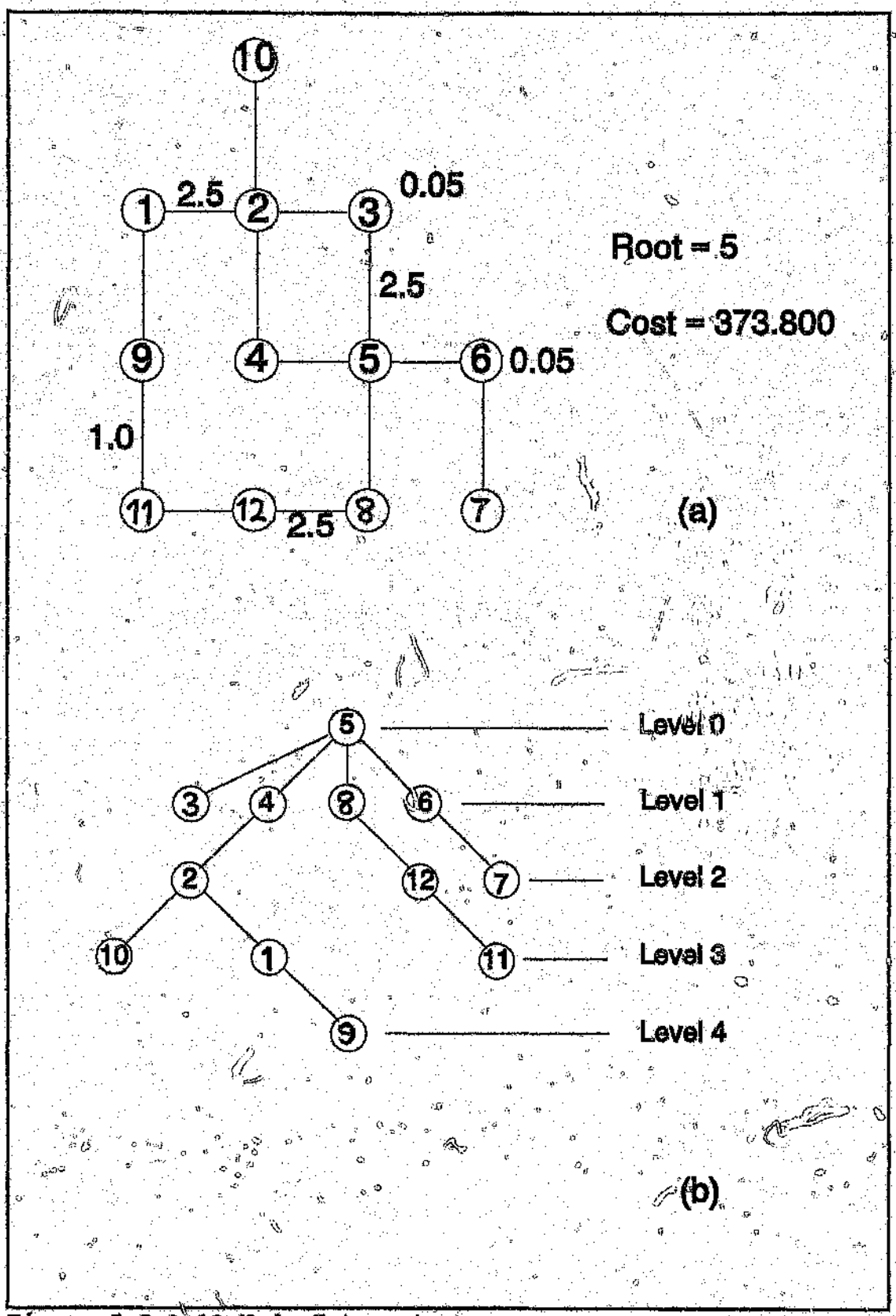


Figure 5.5 A 12 Node Internet

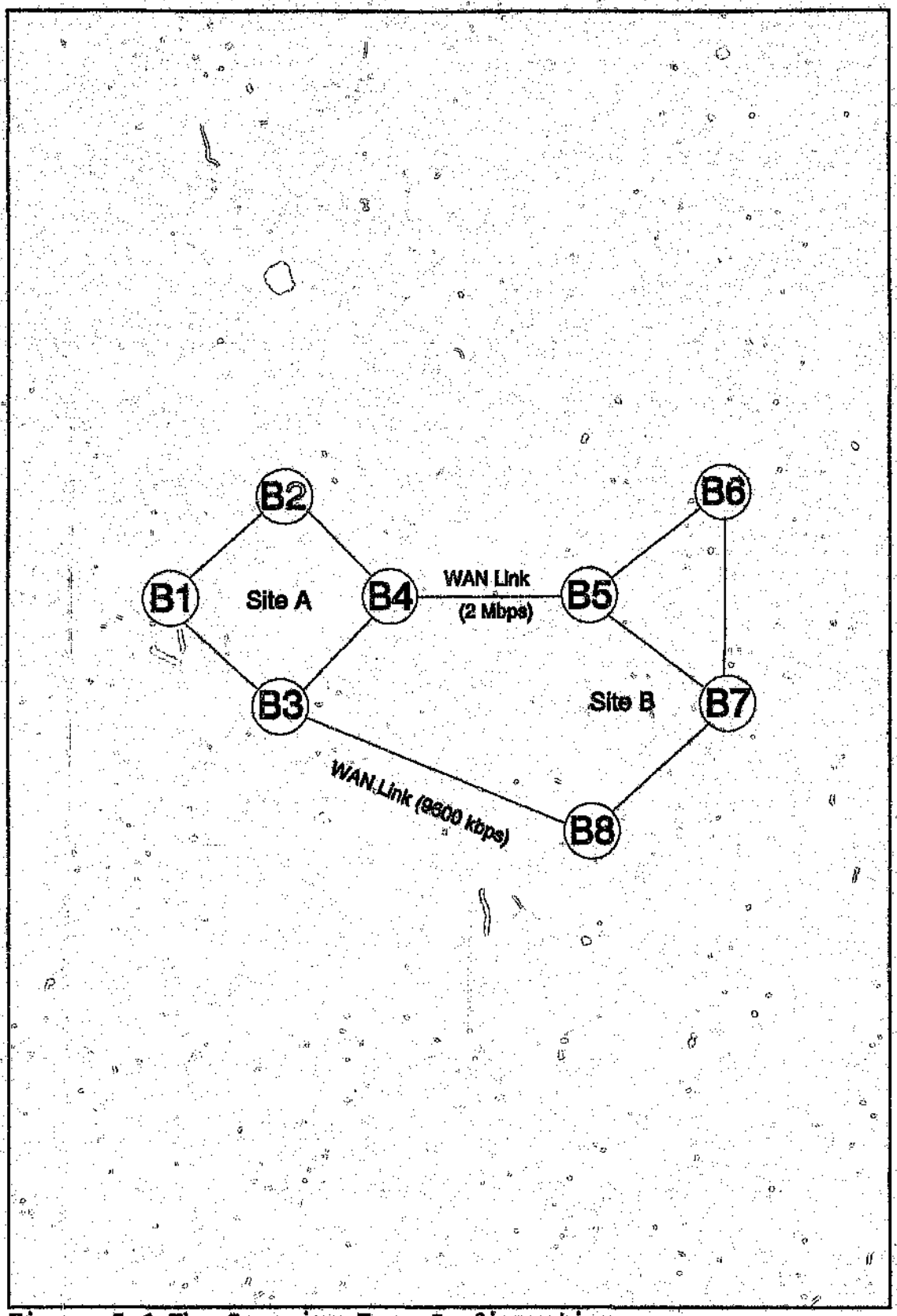


Figure 5.6 The Spanning Tree Configuration

LANs at site B is higher than that at site A. If bridge B8 has the lowest bridge identifier in the internet, it becomes the root bridge.

These factors can cause the Spanning Tree Algorithm to configure the Spanning Tree with the slow link being one of the branches

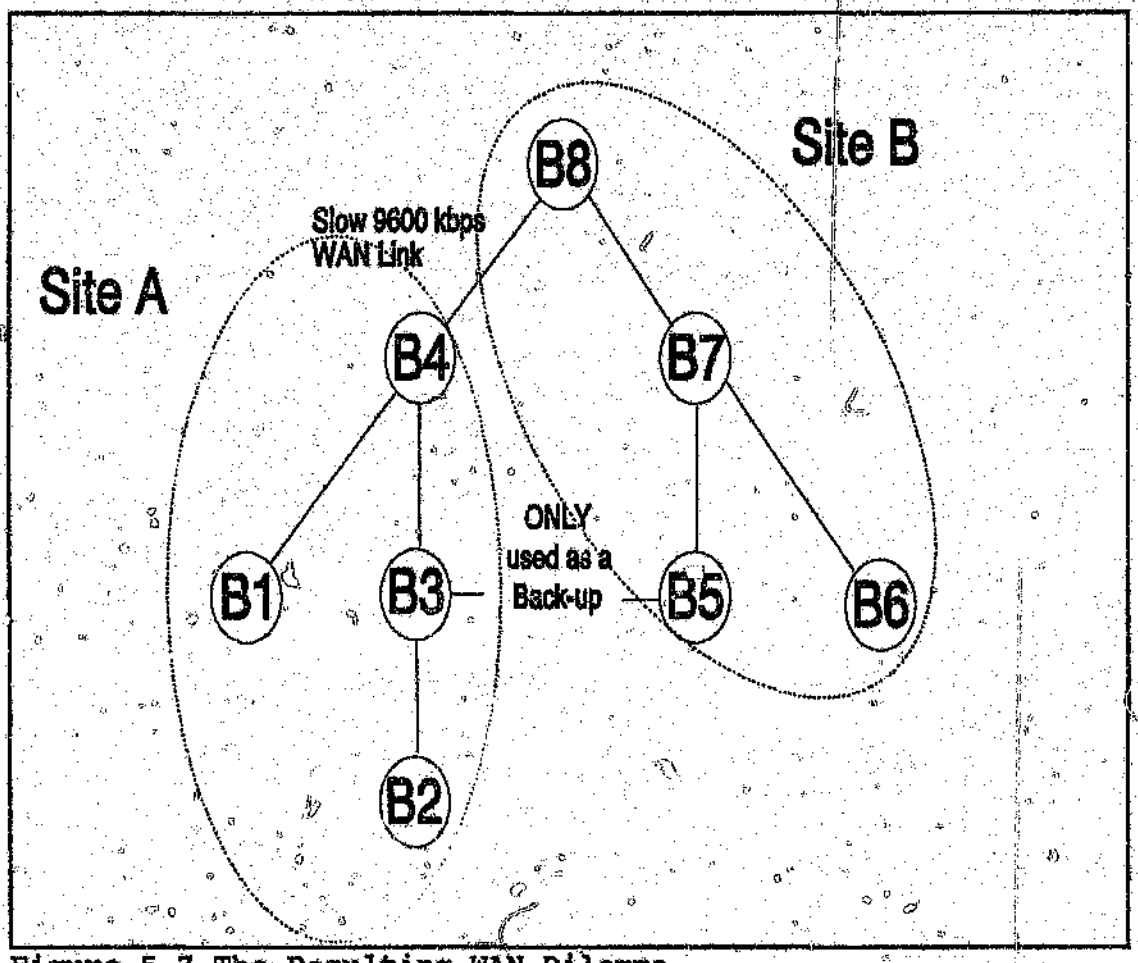


Figure 5.7 The Resulting WAN Dilemma

of the Spanning Tree and the faster, 2Mbps link's bridge ports being blocked. This configuration is shown in figure 5.7.

This example shows that the arbitrary location of Spanning Tree bridges can produce inefficient configurations.

CHAPTER 6

6. CONCLUSIONS AND RECOMMENDATIONS

6.1 Improved Design of Internets

Effective interconnection of networks using bridges is not a trivial task. If the maximum benefit is to be derived from an internet, careful planning of the placement of bridges must be made. Examples in the previous chapter have shown that:

1. The **configuration** of the internet influences the **balance** of the tree.
2. The **location** of the root bridge should be such that the hop count to the end nodes is as small as possible.

These factors would cause the configuration to be sub-optimal. Sub-optimisation results in:

1. An increased delay in the Spanning Tree.
2. Poor fault tolerance.
3. Over design in terms of having too many links.

The operation of the Spanning Tree Algorithm has been explained in this thesis. The effects of the Spanning Tree on the efficiency of the internet has been analysed. The results of the analysis show that:

1. The shape of the tree results in some packets having to follow longer paths because a direct link is blocked by the Spanning Tree

2. The larger the internet, the slower the recovery time is from a topology change.
3. The Spanning Tree protocol chooses an arbitrary root.

The importance of the bridge identifier in the setting up of the Spanning Tree was highlighted. This is the only parameter that can set by the network supervisor so as to influence the configuration of the internet.

The tree level representation was introduced to show the bridges in levels as the distance to the root bridge increases. In Chapter 5 examples 2 and 3 (see figures 5.3 (a) & (b)) the same internet can form a level 2 or a level 3 tree depending on the configuration. The longer the path is from the root bridge to the lowest level, the longer it will take for a topology change notification BPDU to reach this level.

The calculations performed by the program requires a LAN traffic factor to be entered. Reliable LAN traffic measurements can be made and entered as an average. In Chapter 5 example 7 it was shown that if heavy LAN traffic is present during a Spanning Tree re-configuration, the algorithm may set up and maintain a sub-optimal tree.

Internetwork designers, with network reliability in mind, try to include several backup paths in the internet. Backup paths inserted hap-hazardly may result in a sub-optimal tree. In Chapter 5 example 5 a two extra backup paths were introduced to

the configuration in example 2. The resulting tree is no improvement on the tree in example 2, but has a higher average delay than example 3.

The ITA is also performs a backup analysis on the configuration which shows the vital links in the optimal Spanning Tree. This determines which of the bridges should be the most reliable to avoid Spanning Tree re-configurations. Re-configurations cause a major degradation of the internet's performance. Each re-configuration causes inter-LAN traffic to be suspended for 100 milliseconds or more. Role reversal (Hart, 1988) can be reduced by correct design of the configuration.

The examples show that the Spanning Tree set up by the Spanning Tree Algorithm can be pre-determined by analysis. In practice the bridge identifiers can be adjusted to influence the choice of the root bridge. In the examples an improvement of up to 21% was shown in the average transmission delay for the internet.

6.2 Future Developments

An area for future research is to interface the ITA to the Spanning Tree Algorithm running in an internet. The aim of the Spanning Tree Algorithm designers to keep the algorithm simple as possible to reduce processing requirements in the bridge. Thus the algorithm will probably not be incorporated in each bridge.

Currently bridge management is separate from other network management such as faults, performance, configuration, etc. One possibility is to have the ITA algorithm running on a network management system which has more processing power. The management system could then modify the bridge identifiers remotely. The ITA algorithm would run automatically on the management station when bridge links are added or removed. Otherwise, it would be dormant.

Appendix A

Program Modules

BACKUP.NET

float BackupEvaluation()

Calls *BackupEvaluation(int)* for every source - destination (s-d) pair whose link could fail and multiplies the result accumulatively for each pair. The result is returned to the calling routine.

float BackupEvaluation(int failedlink)

For every source - destination (s-d) pair, the routine *CheckPath* is called. *CheckPath* returns a '1' if the given source and destination are unreachable by having the point *failedlink* failing. Returns the value of the variable *BackupCost* from the routine *CheckPath*.

int CheckPath(int ptr, int Failed, float *Backup Cost)

Uses *ptr* (s-d pair) and the failed node *Failed* to determine the next best alternative route and determines the change in cost of using the alternative path.

void swap(int *i1, int *i2)

Swaps the pointers pointing two variables.

void OrderBackup(int thepath, int BackupRoute)

Places the path *thepath* first in the list of paths so that the routine *GetRelativeCost* can be utilised to determine the new path cost.

int BackupGraph()

Provides a graphical output on the screen for the results obtained by the above routines.

LONGPATH.NET

float LongestDelay(int root)

For every s-d pair the routine attempts to find the pair with the highest path cost. It returns the value of the highest path cost.

DISPTREE.NET

class LANonScreen

Once a bridge has been drawn on the screen, its screen co-ordinates are stored in *x* and *y*. Status is set to '1' if the bridge has already been drawn on the screen.

void DisplayTree()

Sets up an array, *PathDone[]*, listing all the paths that have already been drawn on the screen and call the routine *DrawTree* for each path.

void DrawTree(LANonScreen *nets, int b, int *PathDone)

Draws the tree on the screen. This routine is called recursively until all the paths have been drawn on the screen.

int GetMatch(int *PathDone, int ElementInPath)

Determines whether the bridge *ElementInPath* has been drawn on the screen already.

FILEINP.NET

int FileInputs(char *filename)

Reads all the information (bridge speed, LAN speed, LAN load) from the file *filename* and places the relevant values in the global array *thisnet[]*. The routine returns a '0' if successful.

INTERNET.NET

main(argc, *argv[])

This is the main routine of the program calls the other routines to execute all the facets of the program. It also handles minor screen outputs and keyboard inputs.

LANDOWN.NET

void FindNewPaths(int x, int y)

Calls the routine *NewOrder* for every s-d pair, where *(x,y)* represents the link that is down.

void NewOrder(int a, int b, int x, int y)

Determines whether the link that is down *(x,y)* is in the path of the s-d pair *(a,b)*. A new path is determined (if there is an alternative) and reports if it cannot find an alternative path.

LOOPS.NET

int CheckForLoops()

Analysis the internet to determine if there are any loops. This is simply determined by the fact if there is more than one path for any s-d pair then loop does exist.

void LoopReport()

Provides a report on the loops which exist in the internet under analysis.

NETLIB.NET

void MtxToPtr(int ptr, int x, int y)

converts the pointer *ptr* to matrix format *(x,y)*.

int MtxToPtr(int x, int y)

Converts a matrix point *(x,y)* and returns it to the calling routine in pointer format.

void Alarm(char *where, int id)

Prints the message "Allocation error at .... " on the screen when a memory allocation error has occurred.

ORDAGAIN.NET

void OrderAgain()

If more than path exists with the same path cost then the path with the least elements in its path is elected to be the optimum path. This is done to reduce the dependency on too many elements in the path because the more elements you have in a path the greater is the risk of one of them failing.

ORDER.NET

void OrderAll()

Calls the routine *Order* for every s-d pair.

void Order(int source, int dest)

Orders the paths in the global array *thisnet[]* according to cost.

void CopyPaths(int p, TempPathRAM *Temp)

This routine copies the information of an element of *thisnet[]* to a temporary array for manipulation.

PATHE.NET

int GetAllPaths()

For every s-d pair it calls the routine *GetaPath*.

int GetaPath(int source, int destination)

Calls the routine *FindRoute* for every s-d pair in the internet.

void FindRoute(int source, int destination)

Calls the routine *GetBranches* to determine all the links from a specific bridge. The routine *FindRoute* is called recursively for every possible path between the *source* and the *destination*. When a complete path has been determined, it is stored in memory using the routine *StorePath*.

void GetBranches(int network, int connect[])

Determines the bridges which are directly linked to the bridge *network* and stores each of them in the array *connect[]*.

void StorePath()

Adds the path determined by *FindRoute* to the global array *thisnet[]*.

CLASSES.NET and PDECLARE.NET

These two programs define the classes used in the project and the functions used to access them.

class PathRAM

The large array used for storing all the paths determined for the internet.

class TempPath_RAM

A small array designed to hold one set of paths for a s-d pair for manipulation.

class List

A single path array.

class Path

A single path array with manipulatory functions.

class internetmatrix

The global array that stores the bridge and LAN parameters as well as the various paths for each s-d pair.

RELCOST.NET

float GetRelativeCost(int root)

The routine determines the sum of all the paths' costs between

the root bridge *root* and all the other bridges. This gives an indication of the extent of the internetwork.

float OverallRelCost(int root)

Calls the routine *GetRelativeCost* for every s-d pair and returns an average cost taken over all the optimal paths.

float GetRelativeCost(int root, int node1, int node2)

The variables *node1* and *node2* refer to the source and destination respectively. The path between *node1* and *node2* can be broken down into two paths, namely between *root* and *node1* and between *root* and *node2*. The cost of common bridges and LANs in these two paths must be subtracted. The routines *GetFwdElement* and *GetBkdElement* are used to sequentially access the elements in the path either in a forward or backward direction.

int GetFwdElement(Path *pp, int element)

Returns the next element in the path *pp* from *element*

int GetBkdElement(Path *pp, int element)

Returns the previous element in the path *pp* from *element*.

SHUFFLE.NET

void GetAllCosts()

For every s-d pair this routine determines the cost of each path for that pair using the routine *CalcCost_fwd*.

void CalcCost_fwd(int thelink, int thepath)

Determines the total cost for every path and stores the cost in the global array *thisnet[]*.

SPANTREE.NET

int RootNetwork()

Determines the root bridge *RootLAN* by allowing each bridge to be the root in turn and calculating its relative cost using *GetRelativeCost*.

Appendix B

Using the Internet Tree Algorithm

B.1 Defining the Internet as an Internet Matrix

The internet to be modelled must first be translated into a matrix form which the computer program can understand. The matrix will be a simple n by n matrix, i.e. it has an equal number of rows and column. The reason for this is that the rows and column are defined as the bridges of the internet and each bridge is represented once as a column and once as a row.

The intersection of two bridges, defined as position (m,n) in the matrix, represent : LAN or WAN link which forms the transport medium between these two bridges. It also follows that the network connecting Bridge A to Bridge B is the same as the network connecting Bridge B to Bridge A. This means that the matrix is symmetrical about its main diagonal because the entry at point (m,n) would be identical to that at (n,m). This phenomenon is used to significantly reduce the size of data and computing time in the program.

$$\begin{matrix} a & b & c & d & e \\ & f & g & h & i \\ & & j & k & l \\ & & & m & n \\ & & & & o \end{matrix}$$

Matrix B.1

The lower left-hand side of the matrix is therefore discarded as

duplicate information and this provides us with a matrix as shown in Matrix B.1. The matrix points a, f, j, m, and o represent the main diagonal.

The (m,n) entries in the matrix are used to define the LAN/WAN links. This provides the program with the relevant information about the LAN/WAN link, specifically the speed of the network in Mega bits per second (Mbps) and the average data traffic load on the network expressed as a percentage.

Obviously the (m,m) entry in the matrix does not make much sense because there is no internal network inside the bridge connecting a bridge onto itself. Therfore this entry (m,m) will define the bridge itself. It tells the program the maximum number of packets per second (pps) which the bridge can handle.

The simplest way to determine the Internet Matrix of an Internet is to assign each bridge in the network with a number, starting with 1 until each bridge is defined by a number. Then construct a regular n by n matrix and fill in the relevant bridge and network data. Remember that the lower left-hand side of the matrix will contain redundant information.

B.2 Editing the I.T.A. Configuration File

The configuration file of the ITA program is a text based file and may be edited with any one of a vast selection of DOS editors. The ITA configuration file has the extension .CFG and has the format as shown in Matrix B.2. Where N_b refers to the

N_b

$B1_{pps}$ $L12_{Mbps}$ $(L12_{1f12})$ $L13_{Mbps}$ $(L13_{1f13})$ ...

$B2_{pps}$ $L23_{Mbps}$ $(L23_{1f23})$ $L24_{Mbps}$ $(L24_{1f24})$ ... $L2n_M$

.. ..

$B(n\text{-}1)_{pps}$ $L(n\text{-}1)n_{Mbps}$ $(L(n\text{-}1)n_{1f}(n\text{-}1)n)$

Bn_{pps}

Matrix B.2

number of bridges in the internet. If the total number of bridges in a specific internet, including any dummy bridge or bridges which may have to be introduced, add up to five then enter 5 in the first row of the .CFG file. No other data must be inserted in this first line of the file. This indicates to the program how many sets of data it must read from the file.

Bn_{pps} refers to the speed of the n'th bridge in packets per second. All the bridges' data can found as the first entry on each line after the N_b entry. For example, if a bridge is rated as a 15,000 packets per second or frames per second bridge then this entry is merely 15000.

$Ln(n+1)_{Mbps}$ refers to the LAN speed of that specific network in Mega bits per second. An Ethernet network, which runs at 10 Mbps, the entry will be 10. For Token Ring networks the entry will 4 or 16 depending on the speed of the Token Ring network. Arcnet will have a speed of 2.5.

The Local Factor as defined by $Ln(n+1)_{1fn(n+1)}$ is the average local traffic on that network expressed as a factor. this is mostly intended for use when defining bus topolgy networks. In

many cases this factor is not easily determined and an approximation must be made depending on the number of nodes on a network and the type and intensity of the work being performed on the network. Accepted values for this factor range between 5 - 20% for an Ethernet network. Note that Arcnet and Token Ring are ring topology networks and as such are not influenced, to the same extent, by the amount of traffic on the LAN itself, as are bus topology networks like Ethernet. Therefore for ring topology networks this factor can assume the value zero, but must still be entrered into the .CFG file.

B.3 Program Restrictions

Although many different configurations of internets were tested, it cannot be concluded that this program is error free. It was also not the intension of this dissertation to produce a program that could be used on a large scale to improve the efficiency of bridged internets. This program was soley designed to illustrate and suggest solutions to certain problems associated with the Spanning Tree Algorithm. Therefore this program has a limit on the number of bridges per internet it can handle for calculation purposes. The limit has been set to 10. Larger internets may be entered but may cause memory overwrites and even cause a PC to crash.

REFERENECES

Backes F. (1988); Transparent Bridges for Interconnection of IEEE 802 LANs; IEEE Network; Vol. 2; No. 1; Jan 1988; p. 5-9.

Chiarawongse J., Srinivasan M.M. and Teorey T.J. (1988); Performance Analysis of a Large Interconnected Network by Decomposition Techniques; IEEE Network; vol. 2, no. 4, July 1988; pp. 19-27.

Ephremides A. and Verdu S. (1989); Control and Optimization Techniques in Communication Network Problems; IEEE Transactions on Automatic Control; Vol. 34; No. 9; Sept 1989.

Halsall F. (1988); Data Communications, Computer Networks and OSI; 2nd Edition; Addison-Wesley;

Hamner M.C. and Samsen G.R. (1988); Source Routing Bridge Implementation; IEEE Network; Vol. 2; No. 1; Jan 1988; p. 33-36.

Hancock B. (1991); LAN opportunities using multiport bridges; Digital Review; February 4; 1991.

Harris N.G. and Brandl M. (1993a); Configuration Optimization of Spanning Tree; The Transactions of the South African Institute of Electrical Engineers; Vol. 84; No. 1; March 1993; p. 1-8.

Harris N.G. and Brandl M. (1993b); A Study of Factors Affecting

LAN Bridge Performance; The Transactions of the South African Institute of Electrical Engineers; Vol. 84; No. 1; March 1993; p. 37-44.

Harris N.G. (1991); A Model for the Study of Effects of Scale to Aid in the Management of Large Internets; Joint Symposium on Network Management; May 1991.

Hart J. (1988); Extending the 802.1 MAC Bridge Standard to Remote Bridges; IEEE Network; Vol. 2; No. 1; Jan 1988; p. 10-15.

Kershenbaum A., Kermani P. & Grover G.A. (1991); MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing; IEEE Transactions on Communications; Vol. 39; No. 4; April 1991; p. 503-513.

McQuillan J.M. (1989); Routers as building blocks for robust internetworks; Data Communications; September 1989; p. 28-33.

Pitt D.A. & Winkler J. (1987); Table-Free Bridging; IEEE Journal on Selected Areas in Communications; Vol. SAC-5; No. 9; December 1987; p. 1454-1461.

Rickert Jr. J.B. (1990); Evaluating MAC-Layer Bridges - Beyond Filtering and Forwarding; Data Communications International; May 1990; p. 69-72.

Salwen H., Boule R. & Chiappa J.N. (1988); Examination of the

Applicability of Routing and Bridging Techniques; IEEE Network; Vol. 2; No. 1; Jan 1988; p. 77-80.

Schwartz M. (1988); Telecommunication Networks: Protocols, Modeling and Analysis; Addison-Wesley Publishing Company; p. 467-474.

Stallings W. (1990a); Handbook of Computer Communications Standards; Volume 2; Second Edition; Howard W. Sams & Company; p. 207-234.

Stallings W. (1990b); Local Networks; Macmillan Publishing Company; p. 379-407.

Stallings W. (1988); Data Communications, Computer Networks and OSI; Addison-Wesley Publishing Company; 1988;

Zhang L. (1988); Comparison of Two Bridge Routing Approaches; IEEE Network; Vol. 2; No. 1; Jan 1988; p. 44-47.

Author: Brandi Marcus.
Name of thesis: A computerised algorithm to optimise large, heterogeneous Internets.

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2015

LEGALNOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page)for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.