An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN

Radia Perlman

Digital Equipment Corporation
1925 Andover St., Tewksbury MA 01876

Abstract

A protocol and algorithm are given in which bridges in an extended Local Area Network of arbitrary topology compute, in a distributed fashion, an acyclic spanning subset of the network.

The algorithm converges in time proportional to the diameter of the extended LAN, and requires a very small amount of memory per bridge, and communications bandwidth per LAN, independent of the total number of bridges or the total number of links in the network.

Algorhyme

I think that I shall never see
A graph more lovely than a tree.

A tree whose crucial property
Is loop-free connectivity.

A tree which must be sure to span
So packets can reach every LAN.

First the Root must be selected.
By ID it is elected.

Least cost paths from Root are traced.
In the tree these paths are placed.

A mesh is made by folks like me
Then bridges find a spanning tree.

Introduction

Local area networks are limited in geography, traffic, and number of stations. A single local area network will often not meet the needs of an organization for these reasons. Conventional LAN interconnection mechanisms, for instance XNS [1], Sytek[2], IBM1[3], IBM2[4], DNA[5] require cooperation from the stations with compatible protocols layered above the protocol necessary to connect to a single LAN.

An approach that is transparent to stations, and thus allows a station to participate in an extended LAN with no modification, is presented in [6],[7], and [8]. In this approach, a bridge connected to two or more links will listen "promiscuously" to all packets transmitted on each of its links, and forward packets received on one of the links onto the others. A bridge also learns of the location of stations relative to itself, so that it will not forward traffic for a station onto a link unnecessarily.

This approach assumes that the topology is a tree (loop-free). However, requiring a topology to be loop-free means there are no backup paths in the case of bridge or LAN failures. Also, because the technology allows network growth so easily, it might be difficult to prevent someone from adding a bridge and creating a loop. A loop in the topology might cause severe performance degradation in the entire extended network due to congestion caused by infinitely circulating packets. It is undesirable to have a network that can be brought down so easily, merely by plugging a cable into the wrong place.

Thus we have designed an algorithm that allows the extended network to consist of an arbitrary topology. The algorithm is run by the bridges, and computes a subset of the topology that connects all LANs yet is loop-free (a spanning tree). The algorithm is self-configuring. The only a priori information necessary in a bridge is its own unique ID (MAC address), which we are assuming can be attained in some manner, for instance with a hardware ROM containing the value. [9]

Terminology

1. station -- a node in the extended LAN that does not forward packets. It is connected to the extended LAN solely for the purpose of communicating with other stations. It does not participate in the spanning tree algorithm.

2. bridge -- a node connected to two or more LANs, for the purpose of forwarding packets between the LANs.

3. link -- a connection from a bridge to a single LAN.

4. extended LAN -- the collection of LANs connected by bridges.

5. Root--the bridge chosen to be the root of the tree to be formed. Note that there is exactly one Root in the network, chosen dynamically.

6. Designated Bridge--the bridge on a LAN closest to the Root, with a tie-breaker to enforce uniqueness. Thus there is exactly one Designated Bridge per LAN in the network, chosen dynamically.

Goals of the Spanning Tree Algorithm

- Allow interconnection of LANs compatibly with LAN standards, so that stations need not know whether they are attached to a single LAN or to a network of LANs.

- Allow redundant bridges and LANs in an extended LAN, for instance so that connectivity can be preserved after bridge or LAN failures.

- Be self-configuring, so that each bridge need only know its own ID and a well-known generic group address.

- Memory requirements of a bridge should not grow with the number of LANs or bridges, so that a network cannot be misconfigured.

- The communications bandwidth consumed by the algorithm on any particular LAN should be a constant (and small) amount, regardless of the total number of bridges or LANs in the net.

- The algorithm should stabilize as quickly as possible to a deterministic spanning tree (no loops, complete connectivity) in any size network. Assuming no lost messages, it should stabilize in some small multiple of the round trip delay across the network.

- The algorithm should allow bridges to keep caches of station membership (a forwarding data base), to cut down on traffic.

- No permanent loops should ever form. If no messages are lost, no transient loops should form (in any legal sized network). The algorithm should minimize the probability of transient loops, given the possibility of lost messages and the possibility of a network larger than designed for. Similarly, transient behavior should not cycle a link (turn it off, then back on again) in a legal sized network if no messages are lost.

- The algorithm should be self-stabilizing, once any malfunctioning equipment is repaired or disconnected.

- The algorithm should break loops even if some of the bridges in the loop do not implement the algorithm. (Of course it cannot break a loop in which none of the bridges implement the algorithm.)

- The algorithm should require only a connectionless service (LLC type 1) for any LANs on which it is run.

- The algorithm should allow tuning of the configuration by adjustment of parameters. This tuning of the configuration may help performance, but the algorithm should still work acceptably with default values.

Link States

Each bridge has two or more links. For each LAN to which a bridge is attached, the bridge computes a state for that link. In all states, the algorithm continues to run. The only difference between these states is whether data packets are forwarded to and from the LAN.

- FORWARDING -- Forward Data Packets
- BACKUP -- Do not forward data packets
- PRE_FORWARDING -- Do not forward data packets yet; however, unless an event reverts the link to BACKUP, the state will shortly become FORWARDING.
- PRE_BACKUP -- Continue forwarding data packets; however, unless an event reverts the link to FORWARDING, the state will shortly become BACKUP.

HELLO Messages

The Designated Bridge on each LAN periodically transmits a HELLO Message, with period HELLO_TIME, broadcast to the well-known group address "all bridges". The contents of the HELLO Message are:

1. Transmitting bridge ID
2. ID of bridge assumed to be Root
3. Length of best known path to Root
4. Age of the HELLO (time since information from the Root has propagated on this path)
5. Link identifier (of local significance to the transmitting bridge)
6. MAX_AGE (a parameter, passed in HELLOs so that the network will agree on timer values)

If the Root or a Designated Bridge along the path to the Root fails, some time threshold will be exceeded in which no HELLO messages were received on that path. This is detected by keeping track of the age of the HELLO message. When the age exceeds MAX_AGE, the information is discarded, triggering bridges to recompute a new Root or new path.

The AGE field is included in HELLO messages so that a bridge other than the Root can initiate sending a HELLO message, without making the Root's last HELLO seem more recent than it should be. When a HELLO is initiated by the Root, the age will be 0. As the information about the HELLO is held in memory, the age is increased. If a Designated Bridge (other than the Root) initiates transmission of a HELLO, the age field in the transmitted HELLO is filled in to be the age of the stored HELLO. When a HELLO is received, the age is set to the received value initially, and then increased as the information is held.

The Algorithm

As the algorithm is initially presented, primary/secondary status is ignored. In the section "Extensions" the modifications necessary to provide such designation is described.

Electing the Root and Designated Bridges

The Root is the bridge in the network with the smallest ID. When a bridge receives a HELLO Message, it compares the ID of the Root in the HELLO Message with the currently best known Root. If the Root in the HELLO message has a lower ID than the currently known Root, then information about the current Root is discarded and overwritten by information about the newly discovered Root.

The Designated Bridge on a LAN is the bridge with the shortest path to the Root, on that LAN. In case of ties, the bridge with the smallest ID is the Designated Bridge on that LAN. The Root is distance 0 from itself, and claims 0 as the length of its path to the Root in its transmitted HELLO Messages. Other bridges compute their distance from the Root as 1 greater than the minimum distance received from a HELLO Message from the Designated Bridge on any attached LAN.

A bridge B initially attempts to become Root (and therefore Designated Bridge on each of its LANs). When a Root with a lower ID is discovered in a received HELLO message, B then computes its distance from that Root, and attempts to become Designated Bridge on each of its links. If a HELLO Message is received on a link from a bridge that is closer to the Root than B, or equally close with a transmitting bridge's ID lower than B's (as a tie breaker to ensure a unique Designated Bridge per LAN), B defers to the other bridge and stops sending HELLO messages on that link.

The Root is Designated Bridge on each of its LANs, since it is the only bridge which is 0 from the Root. Any other bridge B is some distance K from the Root. B derived its distance K by receiving a HELLO from the Designated Bridge on one of its links containing HOPS of K-1. Thus bridges other than the Root can not be Designated Bridge on all of their LANs -- one of the LANs has to contain a bridge closer to the Root.

Transmission of HELLO Messages

The Root periodically broadcasts a HELLO on each of its links, with period HELLO_TIME. Other bridges only broadcast a HELLO on a link if they are Designated Bridge on that LAN. Since there is exactly one Designated Bridge on a LAN, exactly one bridge on each LAN will periodically broadcast a HELLO message.

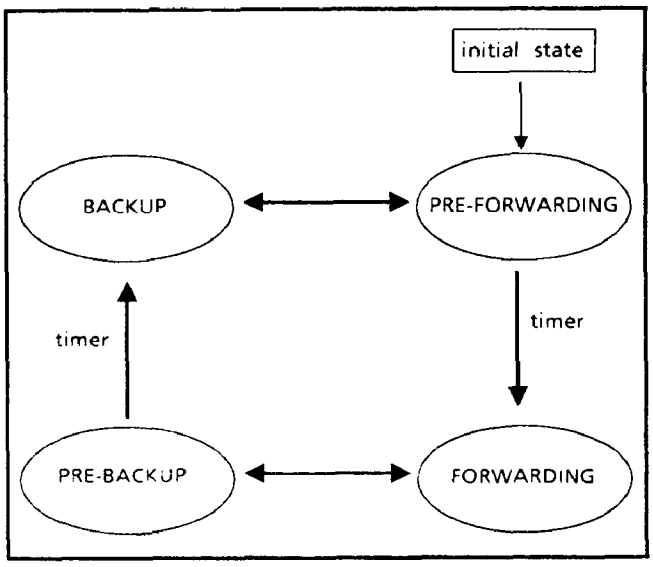
The HELLO packet has an age in it, indicating the age of the information contained therein. Usually a Designated Bridge will transmit a HELLO only upon receipt of a HELLO on the link towards the Root, with a younger age. However, a Designated Bridge will also transmit a HELLO on a particular link upon receipt of a HELLO from a different bridge on that LAN that should not be Designated Bridge.

If the age field exceeds some maximum, the information is discarded. In this way failure of the Root or Designated Bridge will be detected, and the other bridges will be triggered to compute a new Root or a new path to the Root.

Link States and Databases

Each bridge, A, keeps a per-link database for each its links, of the state of that link, the information from the latest HELLO Message from the Designated Bridge on that LAN, whether A should send a HELLO on that link, and a timer, activated in states PRE_FORWARDING and PRE_BACKUP, whose expiration will move the link from PRE_FORWARDING to FORWARDING state, or from PRE_BACKUP to BACKUP state.

State Transitions



Upon startup, bridge B initializes each link to be in the PRE_FORWARDING state, with the link database for each link initialized to claim B as the Root, B as the Designated Bridge on that link, B's distance from the Root to be 0, and the timer set to PRE_FORWARDING_DELAY.

The events that cause the algorithm to run are receipt of a HELLO Message, and a timer pulse. If a HELLO Message is received that supercedes what is stored in the link database, the information in the link database is overwritten. The timer pulse triggers the Root to transmit a new HELLO message (if HELLO_TIME has transpired since the last HELLO message), triggers each bridge to update the age of the information in the link database for each link (causing the information to be discarded if MAX_AGE is exceeded), and triggers links to transit from PRE_FORWARDING or PRE_BACKUP states (if the delay time has transpired).

When an event causes information in the link database to change (either receipt of a different and superior HELLO message, or expiration of the stored HELLO information), bridge B recomputes its own distance from the Root, and examines each link to determine if B should take over as Designated Bridge. If all information about the current Root has expired, then B will attempt to become the new Root, and therefore Designated Bridge on each of its LANs.

If B is not the Root, it selects a single link which is closest to the Root. If more than one link is equally closest to the Root, B chooses one arbitrarily. B attempts to make the state of that link FORWARDING as follows:

- If the link is in FORWARDING or PRE_BACKUP, the state of the link is changed immediately to FORWARDING.
- If the link is in PRE_FORWARDING, nothing is done.
- If the link is in BACKUP, the state is changed to PRE_FORWARDING, and the timer is set to PRE_FORWARDING_DELAY.

Likewise, for each link for which B is Designated Bridge, B attempts to make the state of that link FORWARDING.

For each other link (links other than the one closest to the Root or for which B is Designated Bridge), B attempts to make the state BACKUP as follows:

- If the link is in BACKUP or PRE_FORWARDING, the state of the link is changed immediately to BACKUP.
- If the link is in PRE_BACKUP, nothing is done.
- If the link is in FORWARDING, the state is changed to PRE_BACKUP, and the timer is set to PRE_BACKUP_DELAY.

Handling Data Traffic

Bridges maintain a forwarding data base of stations. An entry in the forwarding data base consists of a station address, together with the link which is in the direction from which traffic from that station was seen.

Assuming for the moment that all links are in FORWARDING state, the use of the forwarding data base is as follows. The forwarding data base is constructed based on the source field of packets. Traffic is forwarded based on the destination field. If a packet is received on link L, with destination D and source S, then an entry is made in the forwarding data base that S resides in the direction of link L. If no entry for D exists, then the packet is forwarded back onto all links except for L. If an entry for D exists, with link L, then the packet is not forwarded. If an entry for D exists with some other link Q, then the packet is forwarded onto Q.

But links can be in other states than FORWARDING. The rules are as follows:

- Data traffic is ignored when received from links in the BACKUP state.
- Data traffic is examined for source address, for the purpose of updating the forwarding data base, from links in the other states (FORWARDING, PRE_FORWARDING, and PRE_BACKUP).
- Data traffic is forwarded to and from links in the FORWARDING and PRE_BACKUP states.

Interaction with Simple Bridges

Simple bridges are ones that do not participate in this algorithm, but pass HELLO messages through transparently. Thus LANs connected with simple bridges will look to this algorithm like a single LAN.

As long as there is more than one bridge participating in the algorithm in every loop, the algorithm will not even detect the simple bridges. However, if there is only one bridge in a loop that participates in the algorithm, the effect is that it will look to that bridge as though it has multiple attachments to the same LAN. This topology will be detected by the algorithm because a bridge will receive its own HELLO message back on a different link.

The HELLO message should include a link number, of local significance to the Designated Bridge. If the Designated Bridge B receives a HELLO from itself on link L1, that claims to have been transmitted on L2, then B should place the link with the larger link number into PRE_FORWARDING state, with timer PRE_FORWARDING_DELAY. Each time such a HELLO is received, the timer should be reset to PRE_FORWARDING_DELAY. If multiple links are involved in the loop, then only one of them will remain in the FORWARDING state. All others will remain in the PRE_FORWARDING state, and the loop will not exist for data traffic.

Timer Values

To discuss the appropriate timer settings below, let us define the following:

- MaxDelay == the maximum one-way delay across the extended LAN
- MaxLostMsgs == the maximum number of consecutive messages that (within an acceptable probability) can get lost
- MaxPropTime == MaxDelay + (HELLO_TIME * MaxLostMsgs)

MAX_AGE -- MAX_AGE is a parameter that should be greater than or equal to MaxPropTime. If the age of a received HELLO message exceeds MAX_AGE, the information is discarded, and the bridge which discarded the information will attempt to take over as Designated Bridge, based on a different path towards the Root, if a HELLO Message that has not timed out is stored for some other link. If all information about the current Root has timed out and been discarded, the bridge will attempt to take over as the new Root, until and unless it hears about a better Root in a subsequent received HELLO message.

The reason MAX_AGE must be at least MaxPropTime is so that a Root that is currently operational will not be mistakenly timed out. The Root can transmit one HELLO that can make it through the extended LAN virtually immediately. (Technically, the Root's HELLO does not propagate through the net, but is received only by the bridges attached to the same LANs as the Root. However, those neighbor bridges transmit HELLOs as a result of receiving a HELLO, so the effect is similar.) The next few HELLOs can get lost (up to MaxLostMsgs). Then the next HELLO might encounter maximum delays, and take the MaxDelay time to reach the periphery of the net.

If MAX_AGE were shorter, then the bridges on the periphery would have timed out the Root before the second HELLO were received, in worst case behavior.

PRE_BACKUP_DELAY -- To prevent transient partitions, a bridge keeps a link in state PRE_BACKUP for a time PRE_BACKUP_DELAY before switching the link from state FORWARDING to BACKUP.

In state PRE_BACKUP, if newer information arrives suggesting the link should be in FORWARDING state, the link is switched to state FORWARDING without delay.

PRE_BACKUP_DELAY should be greater than or equal to MAX_AGE + MaxPropTime.

The reason it needs to be that large is so that sufficient time will have elapsed so that all bridges have received information about the current topology before any bridges stop forwarding.

The worst case is when the Root fails. As described above, it can take a MaxPropTime from the time the final HELLO was transmitted by the Root to the time it is received on the periphery of the net. Bridges near the old Root (in the worst case) will have timed out the Root MaxPropTime before bridges on the periphery. MAX_AGE after receipt of the last HELLO, a bridge will have timed out the Root. If the bridge destined to be the new Root is maximally far from the old Root, it will issue its first HELLO as the Root MAX_AGE + MaxPropTime after transmission of the final HELLO by the old Root. If the bridge destined to be the new Root is maximally far from the old Root, news of the new Root can take MaxPropTime to reach the rest of the net. Thus following the failure of the old Root, it can take MAX_AGE + 2 * MaxPropTime for news of the new Root to reach all portions of the net.

Thus to be completely safe, bridges should wait 2 * MaxPropTime after timing out the Root to ensure they will be making decisions based on the new topology. Thus PRE_BACKUP_DELAY should be greater than or equal to 2 * MaxPropTime.

For simplicity of parameter setting, PRE_BACKUP_DELAY = 2 * MAX_AGE.

PRE_FORWARD_DELAY -- To prevent transient loops, a bridge keeps a link in state PRE_FORWARDING for a time PRE_FORWARD_DELAY before switching the link from state BACKUP to state FORWARDING.

While in state PRE_FORWARDING, if newer information arrives suggesting the link should be in BACKUP state, the link is switched to state BACKUP without delay.

Since loops are far more serious than temporary partitions, it is highly desirable that after a topology change all bridges that should not be forwarding in the new topology will have stopped forwarding before any bridges start forwarding.

News of the new topology will have spread to all bridges within a window of MaxPropTime. Thus MaxPropTime after PRE_BACKUP_DELAY, all links that will be turning off in the new topology will have turned off. At this point it is safe to start turning links on. Thus PRE_FORWARDING_DELAY should be greater than or equal to PRE_BACKUP_DELAY + MaxPropTime.

For simplicity of parameter setting, PRE_FORWARDING_DELAY = 3 * MAX_AGE.

Setting Parameters

A network management facility, such as discussed in [8], allows parameters to be set at a node.

To allow tuning of a particular configuration, parameters HELLO_TIME and MAX_AGE are network management settable (the other timers are derived from MAX_AGE).

HELLO_TIME can differ from bridge to bridge, provided that the Root's HELLO_TIME not be so large as to violate the inequality that MAX_AGE be greater than or equal to MaxPropTime. Thus HELLO_TIME can be set independently at each bridge, with an appropriate default so that performance would be adequate without the necessity of setting this parameter.

However, the network must agree on MAX_AGE. To ensure consistency, MAX_AGE (from which the other parameters are derived) is passed in the HELLO messages and the value passed by the Root is used.

Analysis

Exactly one Root in the net

Suppose there are two Roots, A and B, with A having the numerically smaller ID. If there is physical connectivity between A and B, there will be some LAN with a path to both A and B. On that LAN, as soon as a bridge issues a Hello with information about A, the other bridges on that LAN will overwrite their information about B. This will propagate eventually to B, which will stop attempting to be Root.

Exactly one Designated Bridge per LAN

Each bridge has a unique ID. Of the set of bridges on a LAN that have the smallest hop count from the Root, only one of them can have the smallest ID. Thus there is a unique Designated Bridge per LAN.

A bridge will attempt to become Designated Bridge on a LAN unless a different bridge which has a "better" HELLO message (closer to the Root, or equally close, with lower ID) transmits.

Thus if any bridges are attached to a LAN, one of them will become Designated Bridge.

All LANs are included in the formed graph

If there is physical connectivity between a LAN and the bridge that is Root, then news of the Root will have spread to that LAN, and the Designated Bridge will chosen because of its path to the Root. A bridge which is Designated Bridge on a LAN will be in the formed graph, and will include that LAN in the formed graph.

Since every LAN has a Designated Bridge, every LAN (to which there is physical connectivity from the Root) is included in the formed graph.

Amount of Memory

The amount of memory required for finding an acyclic spanning subset of the topology is just the amount needed for the per link database.

To keep the IDs of the Root and the Designated Bridge requires theoretically the log of the number of nodes, but in practice the fixed value of 48 bits for each is sufficient.

To keep the distance from the Root requires the log of the network diameter, but in practice a fixed value of 16 bits or even 8 bits would suffice.

The other pieces of information are of fixed length.

Bridge B must keep this information for each link attached to B, not each link in the network.

Communications Bandwidth

Maintenance of the algorithm when topology has not recently changed requires one HELLO per HELLO_TIME to be transmitted on each LAN by that LAN's Designated Bridge. In the absence of topology changes, only the Designated Bridge on a LAN will transmit HELLOs, and the only event to cause that bridge to transmit a HELLO is receipt of a HELLO message on its upstream LAN (or for the Root, the periodic HELLO-TIME trigger).

Thus regardless of the total size of the network, only one control message will be sent each HELLO_TIME on each LAN, unless the topology changes.

Following a topology change, a few extra HELLO messages might be transmitted, for instance by a bridge that has just come up, or by all bridges on a LAN following information having timed out. Additionally, the Designated Bridge issues a HELLO in response to a "worse" HELLO, to more quickly synchronize bridges that have just come up. However, there is no need to do this more frequently than HELLO_TIME, so a hold down is instituted to prevent a bridge from transmitting more than one HELLO per HELLO_TIME. Thus the maximum amount of control traffic on a LAN even during topological changes is limited by the total number of bridges on that LAN, not by the total number of bridges in the network.

Note that the overhead of the algorithm is independent of the characteristics of the stations' data traffic (e.g. factors such as frequent connection establishment).

The formed graph is a Tree

The properties of a tree are:

1. It has a unique root.
2. Each node other than the root has a unique predecessor closer to the root.

Analysis of the algorithm is simplified by assuming that the formed graph is bipartite, i.e. that there are two types of nodes:

1. Bridges, and
2. LANs.

The ancestor node of a Bridge is the LAN in the direction of the Root. The ancestor node of a LAN is the Designated Bridge. Thus all nodes have a unique ancestor. Likewise, there is a unique Root, which is the bridge with the lowest ID.

Deterministic Behavior

Given a particular physical topology, behavior of the algorithm is completely predictable. The routes chosen do not depend on the order in which messages were received, or on the order in which bridges were brought up, or on the history of previous topologies.

There are many advantages to deterministic behavior:

1. reproducibility -- It is easier to maintain a network if conditions are reproducible. If routes depended on chance occurrences such as the order in which bridges were brought up, or the order in which messages were received, then it is difficult to diagnose problems, since behavior is not reproducible.

2. configuratiblity -- It is easier to configure a network if behavior of a particular topology can be calculated.

3. predictability -- Users will be more satisfied with a particular level of performance, if the same conditions always produce the same level of performance. However, if sometimes performance is noticeably better, due to chance occurrences, users will not be satisfied with the usual level of performance. Similarly if sometimes performance is noticeably worse, due to chance occurrences under what appears to the user as identical conditions, the network will not be viewed as satisfactory.

Extensions

It might be desirable to influence the topology that is computed by the spanning tree algorithm. One facility that might be desirable is the ability to designate some bridges as "primary" bridges, and others as "secondary" bridges. The algorithm should compute a topology in which no secondary bridges appear, unless no topology exists consisting solely of primary bridges. Also, it might be desirable to minimize through-traffic on some LANs, for instance those that might be lower speed or more congested with local traffic. Also, it might be desirable to configure the network so that topological changes will have minimal impact.

All of these facilities are trivial extensions to the algorithm. These facilities can be provided as follows. The bridge ID should actually consist of a priority field, settable via network management, concatenated as the most significant byte onto the unique 48 bit ID assigned by the hardware.

If the topology is mostly a set of tree-like structures off a backbone, then the priority should be set to be the number of levels away from the backbone LAN that the bridge is. In this way, a bridge will be chosen as Root that is closest to the actual backbone, which is preferable, since it minimizes through-traffic on local segments, and minimizes actual topology changes that would occur due to changing Roots. (If the topology doesn't change, then the endnode caches are still valid, and the network is not partitioned temporarily while nodes switch over to other routes. If the backup Root is very near the old Root, then the topology will not change significantly when the old Root dies, and the net will experience no disruptions due to Root switchover.)

If the only switch is "primary" or "secondary", then a "1" will be the high order bit of the ID if "secondary", and a "0" if the bridge is designated "primary".

HOPS should be a double length field, with a high order part and a low order part. If a bridge is designated "secondary", instead of adding 1 to HOPS to get its distance from the Root (incrementing the low order part of HOPS), it will increment the high order part of HOPS.

To minimize traffic on some LANs, links can be assigned a cost. Then, instead of adding 1 to the distance to the Root given by the Designated Bridge on that LAN to obtain this bridge's distance from the Root, it would add the cost of the link. This will tend to place links with higher costs towards the leaves of the tree, which will minimize through traffic.

One Way Connectivity

Connectivity between bridges can be one-way if one transmitter is broken, or the other's receiver is broken, or some channel hardware works in only one direction. One-way connectivity might be a common hardware failure mode, and it is desirable that the algorithm prevent loops in the face of one-way links.

This algorithm can prevent one-way links from causing loops. A Designated Bridge will detect a problem if another bridge on the same LAN persists in sending HELLO messages, because that would indicate that the other bridge is not receiving the Designated Bridge's HELLOs. Note that if the other node disagrees about the identity of the Root, there is no loop. If the other node agrees about the Root, there is a loop.

If there is no loop, then no harm is done by continuing to forward packets to and from the LAN. Thus if the other node disagrees about the identity of the Root, no further action is taken (other than perhaps logging the condition).

If the other node does agree about the Root, the Designated Bridge must stop forwarding packets to and from the LAN.

Summary

The algorithm presented here maintains a spanning acyclic subset of a general mesh topology. It requires a very small, bounded amount of memory per bridge, independent of the total number of LANs or the total number of bridges. It requires a very small, bounded amount of communications bandwidth on each LAN, independent of the total number of LANs or the total number of bridges. It tolerates lost messages and efficiently utilizes the broadcast nature of multiaccess LANs. It requires no effort on the part of stations. The computed topology converges in at most twice the round trip delay across the extended network. The computed topology is deterministic. Bridges implementing this algorithm can coexist with simpler bridges that do not implement this algorithm, and loops will still be broken, provided that no loop exists composed solely of bridges that do not implement the algorithm.

References

[1] Boggs, Shoch, Taft, and Metcalfe, "PUP: An Internetwork Architecture," IEEE Transactions on Communications, April 1980.

[2] C. Sunshine, D. Kaufman, G. Ennis, and K. Biba, "Interconnection of Broadband Local Area Networks", Eighth Data Communications Symposium, Massachusetts, 1983.

[3] Norman Strole, "A Local Communications Network Based on Interconnected Token-Access Rings: A Tutorial", IBM J. Res. Develop, Vol 27, No 5, Sept., 1983.

[4] Kian-Bon Sy, Daniel A. Pitt, Robert A. Donnan, "An Architecture for Interconnecting LAN Segments", IBM Corporation, Technical submission to IEEE 802 LAN standards committee, July 13, 1984.

[5] Radia Perlman, "Incorporation of Multiaccess Links Into a Routing Protocol", Eighth Data Communications Symposium, Massachusetts, 1983.

[6] Bill Hawe, Alan Kirby, Anthony Lauck, "An Architecture for Transparently Interconnecting IEEE 802 Local Area Networks", Paper presented at the IEEE 802 meeting in San Diego, CA on October 1984

[7] Bill Hawe, Alan Kirby, Bob Stewart, "Transparent Interconnection of Local Networks with Bridges", Journal of Telecommunication Networks, June 1984.

[8] George Varghese and Bill Hawe, "Extended Local Area Network Management Principles," Paper presented at the IEEE 802 meeting in San Diego, CA on October 1984

[9] Yogen Dalal and Robert Printis, "48-bit Absolute Internet and Ethernet Host Numbers", Seventh Data Communications Symposium, 1981.