Bridges exchange control messages, \( M = (S, R, H) \), containing the sender’s address, \( S \), the sender’s best found root address, \( R \), and the number of hops from the receiver to that root, \( H \). A bridge always has a notion of the best message it has so far, \( M_B \). The goal is to find a unique root bridge and paint ports on bridges black if they actively forward both control and data packets, gray if they forward data packets but not control packets, or white if they forward neither type. Note that black ports always face the path away from the root; gray ports always face the path to the root; and white ports are unused ports to break the loops.

Let every bridge perform the following algorithm:

**Step 1:** Let \( M_B = (A_{self}, A_{self}, 0, x) \), and color every port black. \( A_{self} \) is the address of this bridge, and \( x \) doesn’t matter. Let \( n_p = 0 \) for each port \( p \).

**Step 2:** Given \( M_B = (S, R, H, p) \), send message \( (A_{self}, R, H + 1) \) on every black port.

**Step 3:** Wait to receive a new message on any port.

- If timeout with no new packets on any port then go to Step 4.
- If message \( M_{new} = (S', R', H') \) arrives on port \( p' \); \( n_{p'} = 0 \); go to Step 5.

**Step 4:** If \( A_{self} = R \) in \( M_B \), go to Step 2. (this bridge is a root and generates new packets).

- If \( n_{p'} = n_{p'} + 1 < n_{max} \), go to Step 3. (allows for timing variations).
- else go to Step 1. (too many timeouts, must be a problem with current tree).

**Step 5:** Let \( M' = (M_{new}, p') = (S', R', H', p') \). Let \( M_B = (S, R, H, p) \).

- If \( R' < R \) (this port found a better root); or if \( R' = R \) and \( H' < H \) (this port has a shorter path to the root); or if \( R' = R \) and \( H' = H \) and \( S' < S \) (found a lower address neighbor on path to root); then \( M_B = M' \); color \( p' \) gray; color other ports black; go to Step 2.

- If \( R' = R \) and \( H' = H \) and \( p' \neq p \) and \( S' > S \); then color \( p' \) white (some other port has lower address neighbor on path to root);

- or if \( R' = R \) and \( H' = H + 1 \) and \( S' < A_{self} \); then color \( p' \) white (sending bridge and this bridge are both the same distance to the root. Sending bridge has lower address);

- If \( p' \) is colored white go to Step 3; (such messages are not forwarded)

Go to Step 2. (no new info, simply pass message on)

What ports does a bridge forward data packets between? Only on black or gray ports.

To see that the algorithm converges note that \( M_B \) decreases monotonically in lexicographic order, and for a given \( M_B \), ports can only be colored from black to white. The number of bridges originating packets (Step 4) also monotonically decreases.

Assume that when all ports are black, then all LAN’s are connected. Let us see if all LAN’s are connected when the algorithm is done. We first show that any packet that is forwarded by the root bridge will eventually reach every bridge. The lowest numbered bridge will be the root. All of

---

1This is based on the IEEE 802.1D standard. In this standard, black, gray, and white ports are denoted, designated, root, and blocked ports.
its ports will be colored black. Any bridge one hop from the root will continue to receive message packets from the root. These one hop bridges will always have at least one black port connecting to bridges two hops away. This continues until all bridges are reached. Every LAN is connected since on every LAN, at least one bridge must be black. Thus each bridge will forward packets to the correct output for connectivity. By step five, they also listen to a LAN that is on the way from the root.

Since the connections are symmetric—a bridge will listen on ports that it sends and vice versa—every bridge has a forwarding path to the root and the network is connected.

Finally, we show that the network has no loops. Since the network is connected, every bridge has agreed on the same root. Suppose there exists a circuit using only black and gray ports. For any two bridges in the circuit that share a LAN, the number of hops to the root must be different. Otherwise, the last case in Step 5 would have colored the ports connecting to the LAN white on one of the bridges. It follows that there must be some bridge \( B \) on the circuit that is \( H \) from the root and the two adjacent bridges are less than \( H \) from the root. In fact these adjacent bridges must be both \( H-1 \) from the root or \( B \) would be closer than \( H \) to the root. But these nodes have the same root and the same number of hops from the root, so one of \( B \)'s ports toward these adjacent bridges must be colored white by the second to last case in Step 5. Q.E.D.

We will demonstrate the algorithm with an example.

Let bridges B1, B2, B3, B4 be connected to LAN’s La, Lb, Lc, and Ld as shown. The number next to each bridge is the port number. Bridge Bx’s address is \( x \). A port is colored black unless otherwise noted. The below table shows the algorithm progress. At the end, port 2 in bridge 3 would not send any packets on any ports. It would still be looking for spanning tree algorithm messages. Try to see what would happen if B1 or B2 suddenly failed.

<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_B )</td>
<td>( \text{Prt1} )</td>
<td>( \text{Prt2} )</td>
<td>( M_B )</td>
<td>( \text{Prt1} )</td>
</tr>
<tr>
<td>( \text{Snd} )</td>
<td>1,1,0,0</td>
<td>1,1,1</td>
<td>1,1,1</td>
<td>2,2,0,0</td>
</tr>
<tr>
<td>( \text{Rcv} )</td>
<td>3,3,1</td>
<td>2,2,1</td>
<td>1,1,1</td>
<td>1,1,1</td>
</tr>
<tr>
<td>( \text{Snd} )</td>
<td>1,1,0,0</td>
<td>1,1,1</td>
<td>1,1,1</td>
<td>1,1,1,1</td>
</tr>
<tr>
<td>( \text{Rcv} )</td>
<td>X</td>
<td>X</td>
<td>gray</td>
<td>2,1,2</td>
</tr>
<tr>
<td>( \text{Snd} )</td>
<td>1,1,0,0</td>
<td>1,1,1</td>
<td>1,1,1</td>
<td>1,1,1,1</td>
</tr>
<tr>
<td>( \text{Rcv} )</td>
<td>X</td>
<td>X</td>
<td>gray</td>
<td>2,1,2</td>
</tr>
</tbody>
</table>