Link state agreement

Mick Seaman

Two earlier notes[1] proposed an agreement protocol that ensures loop-free active topology(ies) without requiring that bridges cease to forward after every topology change. The protocol has been documented in P802.1aq, and in particular in clause 13.17 of P802.1aq/D2.5. This note proves that the forwarding rules and protocol guarantee loop-free link state operation for (a) simple shortest path destination-based unicast (b) hop-by-hop multipath unicast (c) spanning tree (unicast with possible flooding, and multicast) (d) multicast forwarding with source-specific multicast addresses. The analysis can be extended to additional forwarding rules (for E-TREE, for example).

The agreement protocol was originally modelled on RSTP/MSTP's Proposal/Agreement mechanism[2] and used per tree explicit priority vector communication. The simple extension to use an Agreement Digest had forwarding constraints that were stronger than necessary[3], inviting different and possibly incompatible readings of D2.5. This note improves the description, selecting optimizations that use a common approach to details for SPBM and SPBV, and for explicit priority vectors (for the CIST) and the agreement digest (for all other trees) — minimizing the record keeping required for a full compatible set of forwarding rule. Thanks to Nigel Bragg for his comments on D2.5[4], which made me realize the need for the improvements: these provide additional detail for the ballot comment resolution changes.

This note was primarily written to explore alternatives. The arguments developed are thus more complex than needed just to prove the principal results A simpler contribution, specific to the latter would be welcome.

This note is not yet complete, various opportunities for optimization and extension are being considered, as well as improvements in presentation. However its present state should be good enough to facilitate discussion during the March 2010 meeting.

1. Overview

A goal of this note is to increase the rigour used in the loop-free arguments, and to provide tools for their extension and refinement. Mathematical notation is used (2) in an effort to be concise and exact, though the argument is spelled out in plain English[5].

Agreement protocol exchanges occur between link state neighbours[6], who use the agreements sent and received to ensure that a network's active topology at any instant is loop-free—even if delays in information propagation and processing mean that network nodes have arbitrarily different ideas of the physical topology as it changes.

The agreements and agreement protocols considered in this note are restricted in two ways. First, the information extracted from a participating node's link state database should always be that for the currently calculated topology, and not extend beyond that for the node, or its immediate neighbours on each port. Any additional record of past topologies is to be at worst proportional to the number of immediate neighbours. This restriction limits the complexity of the ways in which we examine the topology—the agreement algorithms themselves cannot be allowed to become a significant contributor to the delays that necessitate their use. Second, the agreements have to be

[1] Link state bridging, 24th March 2008. Link state bridging part 2, July 12th 2008.

[2] Those familiar with IP fast reroute will also find some parallels or similarities here, though the objectives are somewhat different and the development of the agreement protocol has proceeded independently. I have not had the time to revisit IP FRR and translate terminology or add specific references.

[3] Because distance vector and link state operation pose different challenges. In the former distance information flows down the tree, so a child never 'jumps above' its parent and agreements only have to be checked when new connectivity is made. In contrast one of link states' key advantages would be negated by requiring agreement information to propogate along a tree, even if that were possible with a digest. Connectivity has to be broken whenever the agreements in place are insufficient to justify that connectivity were it new. With this breaking the obvious extension from distance vector to link state is over constraining.

[4] And for comments on an earlier draft of this note.

[5] I have read too many published papers in which a perfectly simple argument is dressed up in mathematics, apparently to add 'weight' or merit and to establish that the author belongs to the same club as the reviewers (a club excluding most implementers). Translating into English is often tedious (virtus dormitiva).

[6] The link state neighbours of immediate interest are bridges attached to the same LAN, but the technique could obviously be applied to routing at any level.

compact—easily carried by a simple protocol within a single frame even in the largest networks we aim to support—again to limit the complexity and delay associated with agreements. This second restriction points to the use of a summary (digest) for all but a few destinations or trees, with an agreement convention that uses the link state database identified by a given digest value[1].

The agreements considered, whether explicit or digest based, use comparisons between and the values of the per tree (per destination or source) *priority vectors* for a bridge and each of its neighbours.

NOTE—A *priority vector* is equivalent to the routing concept of distance, generalized to allow for (a) the construction of a tree whose root (to or from which distance is measured) can change; (b) multiple regions, within which any distance is less significant than distance between regions. A priority vector with components {Root Identifier, Root Path Cost} is equivalent to the distance from the best possible Root with priority vector {0, 0}.

A number of different forwarding rules are defined (3.1– 3.6) with the choice of rule for any particular frame depending on the frame's VID and whether that frame is multicast or unicast. VID based rule selection allows SPBV, SPBM, and other forwarding methods (e.g. PBB-TE, E-TREE) to coexist. To minimize the total state, each rule is described as specifying a tree, though that may confuse readers who focus on just one rule: for shortest path unicast forwarding the tree is rooted at the destination; for shortest path multicast the tree is source rooted; and in general spanning tree forwarding the root can be at any node, and frames travel both up and down the tree.

The chosen forwarding rules (and supporting agreement conventions) focus on maintaining connectivity when links or LANs fail, rather than (for example) the earliest possible use of using links that are added to the network. If a node's distance for a path[2] to the root (a bridge's priority vector) decreases (gets better) that node has to either (a) delay sending agreements with the new distance or (b) stop forwarding frames on that path, until (c) the agreement messages from the node's neighbours signal their awareness of the new distance. On the other hand if a node's distance gets worse, the node can begin or continue forwarding from (and/or to, as specified by the particular rule) any given neighbour—up to the point where that neighbour is forwarding to the node in the belief that the node is closer to the root but that is no longer guaranteed. Figure 1 shows a fragment of a network using spanning tree forwarding (bridges A, B, C, and D, connected by point-to-point LANs) to illustrate this 'failure friendly' approach.

In Figure 1 the relative distance of bridges from the root is indicated by their height on the page. The initial network topology, for which each bridge has transmitted and received agreements, is on the left (a). The path from D to the root lies through C and then B. If the link C-A were to fail (b), C can start forwarding to B, provided C is still above D's minimal agreed distance for forwarding through C. An additional failure in the network might increase B's distance (c) but provided B remains above (closer to the root than) C's minimal forwarding distance as shown in (b), B can continue to forward frames. Note that this distance can be greater than C's previous best distance to the root through A (in (a)).

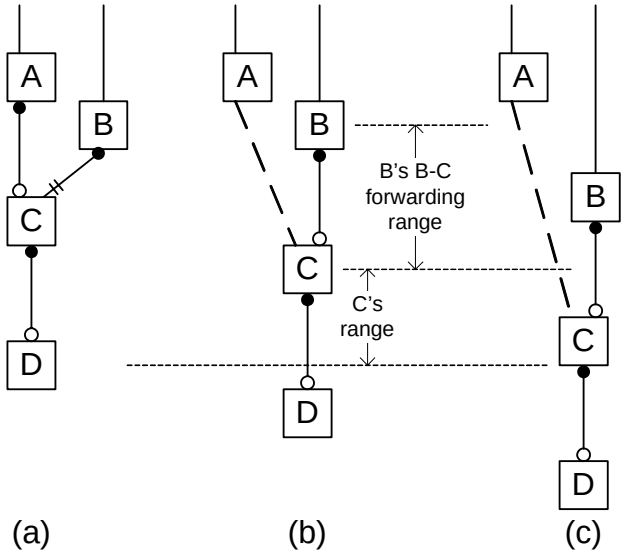

Figure 1—Connectivity restoration and continued forwarding after multiple failures

So forwarding can be maintained, even in the face of (some) multiple failures which cause the network nodes to have temporarily different views of the topology. Eventually their views will converge, their agreement digests will become intelligible to one another once more, and fresh agreements will be made (and superseded agreements discarded). However it has to be said that some network failures will result in a loss of connectivity that cannot be repaired until fresh agreements have been exchanged. The behavior of the agreement protocol in these cases is summarized below, after discussion of the preferred form of each of the forwarding rules.

[1]In principle a very small amount of additional information, just a few bits, could also be carried per destination/tree/network node. I have not yet seriously studied the potential benefit/opportunity, as the digest alone yields satisfactory results.

[2]Each path considered comprises the port/next hop node to/from which forwarding is in question and the best path from that first hop node on.

Link state agreement

In simple shortest path forwarding to known unicast destinations (for point-to-point LANs, as used by SPBM, 3.1) each frame is forwarded (or not) through the one port that provides the shortest path to the destination, no matter what port the frame is received on. Each bridge ensures that it has an agreement from every neighbour, that that neighbour is either closer to the destination (and so will not forward a frame for that destination to the bridge) or is further away and will not forward the frame unless the minimum forwarding distance criteria is met. To put this another way, each agreement comprises two elements:

a) A statement as to the relative distance of the two bridges—is the neighbour 'above' (closer to the root) or 'below' (further away).

b) The minimum distance, "I guarantee to remain below", at which the lower bridge will forward frames to the upper.

Item (a) is equivalent to the neighbour's *Port Role* (as defined in 802.1Q[1]) with each bridge's Root Port providing connectivity towards the root, while Designated Ports provide connectivity toward the leaves of the tree. The role names 'Up Port' (towards the root) and 'Down Port' can be also used to avoid accidentally invoking the additional characteristics associated with spanning tree port role names (bi-directional forwarding, applying port state on egress and ingress for every port). The 'Up Port' role can be considered to cover both Root and Alternate Ports.

Since an agreement protocol participant may not receive all frames sent by his neighbour, and not all values of the agreement digest may be intelligible to a bridge that has chanced to process a succession of changes in a different order or in different combinations, it is the greatest value of the minimum distances agreed that is of interest. A bridge that transmits an agreement digest (for a given topology) through an 'Up Port' is making a promise not to forward a frame through that port unless its distance is greater than or equal to the sum of its neighbour's distance (in that topology) plus the link cost[2]. If the neighbour uses that agreement in a forwarding rule he is promising not to forward the frame unless his distance to the root is less than that sum. Consistent application of the forwarding rule in successive bridges ensures that any forwarded frame gets successively closer to the destination.

This simple rule does not require additional logic to ensure that a frame is not forwarded through the same port on which it was received, and in fact the frame can be forwarded through any 'Up Port' with an agreed 'Down Port' neighbour (subject to the forwarding bridge abiding by the terms of any agreement transmitted). Loop-free hop-by-hop multi-path forwarding (3.3) is completely covered.

Spanning tree forwarding uses both ingress and egress controls. Frames are relayed from a bridge's Root Port[3] (provided that port's outstanding agreements allow it to forward at its current distance from the root) to Designated Ports (that have received agreements that commit their neighbours to forward only if they are at a greater distance), while frames received on an agreed Designated Port can be transmitted on the Root Port and other agreed Designated Ports. Thus an upward going frame (proceeding towards the Root, through bridges with successively better priority vectors[4]) can be turned into a downward going frame. However this change of direction can happen at most once: since each bridge has just one Root Port a downward going frame (heading away from the Root) cannot be turned into an upward going frame. Further, since each LAN has a single agreed Designated Port (and Designated Bridge), the turned around frame will not revisit any LAN. The proof that the spanning tree forwarding rules guarantee loop-free operation thus comprises two parts: (a) frames proceeding successively from the Root Port of one bridge to the next Designated Port of the next pass through bridges that are successively closer to the root (successively further away for a frame proceeding in the other direction) so those ports canot be arranged in a loop; and (b) there is at most one forwarding Designated Port (and thus one Designated Bridge with a single Root Port) for each LAN in the network.

The forwarding rules specify which ports can relay frames given the agreements held (received, and not explicitly discarded by the protocol) and outstanding (transmitted, without receipt of a subsequent discard). Equally they determine what agreements can be sent, and which discarded, given the forwarding ports. Since the agreements for different trees are associated with the same digest, and cannot be separated one from another except by filtering through the current

[1] In exactly the same way as originally defined for IEEE Std 802.1D.

[2] EISS-SPB includes mechanisms to ensure that link costs are symmetric.

[3] The port currently providing the shortest path to the root.

[4] As is conventional in computer science and graph theory, and despite the confusion this causes (unicast 'upstream' being 'downward', and 'downstream' being 'upward') trees are considered to have their roots at the top. with trunk and branches proceeding downward. Equivalently better priority vectors (numerically lesser) are depicted higher on a page, with worse priority vectors (numerically greater) below.

state of the link state database, there is little scope for strategically delaying agreement transmission once a new topology (and its corresponding digest) has been calculated—the forwarding pattern is reduced as required and fresh agreements sent as soon as possible.

This note maps the mathematical description of the agreements and forwarding rules of most interest to SPB (4.3) to the real state variables and conditions (4.3) used in the P802.1aq state machine description.

An important aspect of the agreement algorithm and protocol is that a number of messages can be 'in flight' between the participants, even if that only means sitting in a transmit or receive queue. Agreements are not necessarily delivered faster than new link state topologies are computed. One bridge can never be sure of the state of another except as circumscribed by agreements received (and not yet discarded) and by those sent and still outstanding. The fact that a port attached to a point-to-point link is an agreed Designated Port does not directly mean that the neigbouring bridge's port is a Root or Alternate Port, but that it is not also an agreed Designated Port and is thus only forwarding if it is a Root Port (for simple spanning tree, for example). The protocol's use of sequence numbers, and the precise conditions under which new agreements are sent, old agreements discarded, and protocol state variables updated are detailed in 4.5. While absolute protection against misordering of agreement protocol messages is difficult, moderate misordering (2-3 messages) is handled (as discussed in 5).

Since the complete absence of agreement means that loop-free forwarding is impossible, an agreement protocol participant never simply discards all past agreements. However this restriction does not require an excessive number of message exchanges when fresh agreements are required for forwarding. Consider the case where two participants A and B, say (attached to the same LAN, and agreed on a stable topology) process a new link state database change. Assume A completes the link state calculation first, reduces its forwarding to the intersection of that permitted by past agreements and the new topology, and sends the new agreement digest to B, implicitly discarding any prior agreements that conflict with that digest. At this point B has not yet finished its calculation, but when it does it will recognize and use the fresh agreements from A, put all the forwarding for the new topology in place, discard all previous received agreements and transmit to A. On receipt of this message A can also put all the forwarding for the new topology in place, and discard all agreements it has from prior topologies. To achieve full forwarding on the new topology required A and B to each transmit (and receive) just one message. The same is true if they complete their link state calculations at much the same time and the messages 'cross'.

2. Basic terminology and notation

The following terminology and notation is used to facilitate discussion:

{A, B,... H}	—	the bridges in a network.
{W, X,Y, Z}	—	free variables identifying any neighbouring bridges W, X, Y, Z ∈ {A, B,... H}.
n_Y	—	the immediate neighbours of any given bridge Y.
p_{Y-Z}	—	the port of bridge Y that connects to bridge Z ∈ n_Y.
cost	—	a priority vector (distance) increment.
distance	—	a priority vector (distance from the root of a tree).
up, upward	—	connectivity, potential connectivity, or forwarding a frame, towards the root of a tree.
down, downward	—	connectivity, potential connectivity, or forwarding a frame away from the root of a tree.
downstream	—	'downstream' is used only in the context of known unicast forwarding, and denotes connectivity, potentially connectivity, or forwarding towards the destination. The term is potentially but inevitably confusing as the communication may be modeled as forwarding on a tree rooted at the destination, in which case 'downstream' and 'up' (or 'upward') refer to the same direction.
upstream	—	'upstream' is used only in the context of known unicast forwarding. When the communication is modeled as forwarding along on a tree rooted at the destination, 'upstream' and 'down' (or 'downward') refers the same direction.
c_{Y-Z}	—	the cost (priority vector increment) associated with p_{Y-Z}. Note that mechanisms are used to ensure that costs are symmetric: $c_{Y-Z} = c_{Z-Y}$.
t_i	—	a network topology (and all the information that can be extracted from that topology), t_i ∈ {any possible active topology}.
g_i	—	the agreement di**g**est for t_i . We assume that, to an acceptable level of probability in any given network, ∀ $t_i \neq t_j$ ($g_i \neq g_j$), so each agreement digest identifies the complete network topology when received (or subsequently processed) by a bridge whose link database reflects that topology. Thus any given digest can convey all the agreements possible in any agreement convention based on the bridges in a given active topology, their identifiers, port identifiers and costs, distances on shortest or constrained paths etc.
a_i	—	an agreement that can be expressed by reference to t_i.
$a_{Y\rightarrow Z}$	—	the set of agreements transmitted (advertised) by Y to Z, Z ∈ n_Y, and considered outstanding by Y.
$a_{Y\leftarrow Z}$	—	the set of agreements received from Z, and held (i.e. not discarded) by Y, that matched an agreement calculated by Y (either on receipt or subsequently). Note $a_{Y\leftarrow Z} \subseteq a_{Y\rightarrow Z}$.
Y_i	—	the shortest path distance for bridge Y in topology *i*, similarly X_i, Z_i for bridges X, Z.
$Y_{i.Y-Z}$	—	the distance for bridge Y (in topology *i*) for the path that is constrained to visit Z as the first hop and is the shortest path thereafter: $Y_{i.Y-Z} = c_{Y-Z} + Z_i$. Similarly $X_{i.X-Y}$.
$Z_{i.Y-Z}$	—	the distance for bridge Z (in topology *i*) for the path that is constrained to visit Y as the first hop and is the shortest path thereafter: $Z_{i.Y-Z} = c_{Y-Z} + Y_i$. Similarly $X_{i.X-Y}$.
Y_Y	—	the shortest path distance for bridge Y in the latest topology computed by bridge Y.
$Z_i < Y_i$	—	'Z_i less than Y_i', meaning that Z is closer to the root than Y in t_i, also (since trees are conventionally shown with their roots uppermost, with better priority values higher on the page) 'Z_i is above Y_i', 'Y_i is below Z_i'. Entirely equivalent to $Y_i > Z_i$.
Y_{Y-Z}	—	the distance for bridge Y (for the latest topology computed by Y) for the path that is constrained to visit Z as the first hop and is the shortest path thereafter.
$\underline{X}_{Y\rightarrow X}$	=	the least distance advertised by Y (transmitted by and considered to be currently outstanding by Y) for paths from X constrained to Y as the first hop with Y < X, infinite

if there is no such outstanding agreement. Similarly for $\underline{Y}_{Z\rightarrow Y}$ (distance for Y, constrained to Z as first hop and advertised by Z).

$\underline{Y}_{Y\leftarrow Z}$ = the least distance for Y, for paths constrained to Z as the first hop, in agreements received from Z with (Z < Y), infinite if there is no such received agreement.
Note $(a_{Y\leftarrow Z} \subseteq a_{Z\rightarrow Y}) \Rightarrow (\underline{Y}_{Z\rightarrow Y} \leq \underline{Y}_{Y\leftarrow Z})$

$\overline{Y}_{Y\rightarrow Z}$ = the greatest outstanding distance advertised by Y for paths from Y constrained to Z as the first hop with Z < Y, infinite if any of the outstanding agreements have (Y < Z), and zero if there is no outstanding agreement. Similarly $\overline{Y}_{Y\rightarrow X}$.

$\overline{Z}_{Y\leftarrow Z}$ = the greatest distance for Z (for paths constrained to Y as the first hop) in agreements received from Z with (Y < Z), infinite if any of the agreements received from Z has (Z < Y), zero if no agreement has been received. Similarly $\overline{X}_{Y\leftarrow X}$.
Note $(a_{Y\leftarrow Z} \subseteq a_{Z\rightarrow Y}) \Rightarrow (\overline{Y}_{Z\rightarrow Y} \geq \overline{Y}_{Y\leftarrow Z})$

A single port on bridge Y (say) can provided connectivity to more than one neighbouring bridge. The following notation covers shared media, without complicating the presentation of the simpler point-to-point only case to the extent that it would be desirable to present the latter separately (thus bulking out this note and increasing the chance of mistakes):

$\{\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}\}$ — sets of bridges $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \subset$ {A, B,... H} attached to the same LAN.

$p_{Y-\boldsymbol{Z}}$ — a port of bridge Y that connects to bridge $Z \in \boldsymbol{Z}$, $\boldsymbol{Z} \in n_Y$. $Z \in \boldsymbol{Z} \Rightarrow p_{Y-Z} \equiv p_{Y-\boldsymbol{Z}}$

$a_{Y\rightarrow \boldsymbol{Z}}$ — the set of Agreements transmitted by bridge Y to each bridge $Z \in \boldsymbol{Z}$, $\boldsymbol{Z} \in n_Y$, $(Y \notin \boldsymbol{Z})$ and considered outstanding by Y.

$a_{Y\leftarrow \boldsymbol{Z}}$ — the set of Agreements received from all bridges in $\boldsymbol{Z}$, and held by Y.

Conventional logic and set notation is used with the above.For those for whom this is a distant memory, and to save my self embarrassment lest I have stretched this in unapproved ways, here is a quick refresher:

$\in$ — is a member of.

$\wedge$ — logical and.

$\vee$ — logical or.

$\neg$ — logical not (higher precedence than $\wedge$ and $\vee$).

$\forall$ — for all, as in $\forall$ x(p(x)) — for all x, proposition or condition p(x) is true.

: — such that, as in $\forall$ x:(q(x))(p(x)) — for all x such that q(x) is true, p(x) is true.

$\exists$ — there exists, as in $\exists$x:(p(x)) — there exists an x such that p(x) is true.

$\equiv$ — is equivalent to, for example $(x < y) \equiv (y > x)$.

$\Rightarrow$ — implies, for example $(x < 2) \wedge (y > 3) \Rightarrow (y > x)$.

There are, of course, many ways of saying the same thing, and I have often picked what seems (to me) to be the clearest exposition in a local context without attempting overall consistency or minimal use of variants. In particular note that $\forall$ x:(q(x))(p(x)) $\equiv$ $\forall$ x($\neg$q(x) $\vee$ (p(x)) — saying ‘for all x such that q(x), p(x) is true’ is equivalent to saying ‘for all x, either q(x) is false or p(x) is true’.

3. Loop-free forwarding conditions

This section (3) considers a number of sets of forwarding rules (3.1– 3.6).

3.1 Destination-based unicast forwarding

In simple destination-based unicast forwarding, each frame destined for a destination D (say) is forwarded through the one port that provides the shortest path to D, no matter what port the frame is received on. A bridge's forwarding logic usually discards a received frame rather than forwarded it back through the receiving port, but this forwarding rule does not rely on that 'split-horizon' check' (see also 3.2). The only forwarding plane control available to each bridge is to remove the entry for D from its forwarding database, thus causing the frame to be dropped. For the tree rooted at D:

$$\begin{aligned} \text{spUnicast}_{Y-Z} \quad \equiv \quad & (Y_{Y-Z} = Y_Y) && \wedge \quad \ldots . \text{(1a.1)} \\ & (\exists a_i \in a_{Y\leftarrow Z} : ((Z_i < Y_i) \wedge (Y_{i.Y-Z} \leq Y_Y))) && \ldots . \text{(1a.2)} \\ \wedge \ & (\forall X \in n_Y && \ldots \ldots \text{(1b)} \\ & (\forall a_i \in a_{Y\rightarrow X} : (Y_i < X_i)\ (Y_Y < X_{i.X-Y}))\ ) && \ldots . \text{(1b.1)} \end{aligned}$$

Equation 1 considers forwarding of any given data frame through port p_{Y-Z} of bridge Y. spUnicast_{Y-Z} is true iff the frame can be forwarded, and depends on the agreements received and sent on p_{Y-Z} and sent onY's other ports. It can be summarized as allowing Y to forward frames to Z only if: (a.1) Y's shortest path to the destination lies through Z; and (a.2) an agreement has been received from the 'up' port neighbour Z, agreeing that the port provides upward connectivity, and Y's distance is now no better than it was for connectivity through Z in that agreement (thus guaranteeing that Z, if still forwarding, remains above Y); and (b, b.1) none of the agreements that Y may have sent to any neighbour (of those indicating that Y provides a possible 'upward' path) claim that Y will remain closer to the root than it is currently (the claimed distance is less than the best that the neighbour could have and satisfy its own (a.2) condition to forward frames to Y).

The benefit of using the agreement protocol and the rule specified by eqn. 1 for known unicast destination forwarding can be summarized as follows. Each bridge can forward frames while ensuring that there are no loops, even after a topology change, provided that each bridge is not (a) closer to the destination than required for the path through its downstream neighbour (using that downstream neighbour's advertised distance) to be the shortest path (b) further from the destination than any of its upstream neighbours that can take advantage of its own advertised distance to forward frames to it. Equation 1 assumes (in a.1) that there is a unique shortest path (through a single port p_{Y-Z}) to any destination, so each frame is forwarded through only one port, and that is the same port.

More formally, for bridges interconnected by point-to-point links there are two ways in which loops can arise. First, bridges might be connected in a circle with each bridge's 'up' port (satisfying constraints (a.1) and (a.2) in eqn. 1 as well as (b.1)), connected to another bridge's port that merely satisfies (b.1). Second, two connected bridges might each transmit a given received frame back to the other, either on the receiving port or on another port connecting the two bridges. This second case can thus be considered a sub-case of the first—the smallest possible loop.

Consider bridges D, C, B, A, each believing itself to be forwarding a frame (successively) on a path to a given destination. Instantiating eqn. 1 for the relevant two ports on each of the bridges B and C we have:

$$\begin{aligned} \text{spUnicast}_{B-A} \quad \equiv \quad & (B_{B-A} = B_B) && \wedge \quad \text{(2a) from (1a.1)} \\ & (\exists a_i \in a_{B\leftarrow A} : ((A_i < B_i) \wedge (B_{i.B-A} \leq B_B)) && \wedge \quad \text{(2b) from (1a.2)} \\ & (\forall a_i \in a_{B\rightarrow A} : (B_i < A_i)\ (B_B < A_{i.A-B})) && \wedge \quad \text{(2c) from (1b.1)} \\ & (\forall a_i \in a_{B\rightarrow C} : (B_i < C_i)\ (B_B < C_{i.C-B})) && \quad \text{(2d) from (1b.1)} \end{aligned}$$

Link state agreement

$$\begin{aligned}
\text{spUnicast}_{C-B} \quad \equiv \quad & (C_{C-B} = C_C) && \wedge \quad \ldots . \text{(2e) from (1a.1)} \\
& (\exists a_i \in a_{C\leftarrow B} : ((B_i < C_i) \wedge (C_{i.C-B} \leq C_C)) && \wedge \quad \ldots . \text{(2f) from (1a.2)} \\
& (\forall a_i \in a_{C\rightarrow B} : (C_i < B_i)\ (C_C < B_{i.C-B})) && \wedge \quad \ldots . \text{(2g) from (1b.1)} \\
& (\forall a_i \in a_{C\rightarrow D} : (C_i < D_i)\ (C_C < D_{i.D-C})) && \ldots \ldots . \text{(2h) from (1b.1)}
\end{aligned}$$

Since $a_{C\leftarrow B} \subseteq a_{B\rightarrow C}$:

$$\begin{aligned}
\text{spUnicast}_{C-B} \ \wedge \ \text{spUnicast}_{B-A} & \\
\Rightarrow \quad & (\forall a_i \in a_{B\rightarrow C} : (B_i < C_i)\ (B_B < C_{i.C-B}) && \ldots . \text{from (B–A:2d), (1b.1)} \\
& \wedge \\
& (\exists a_i \in a_{C\leftarrow B} : ((B_i < C_i) \wedge (C_{i.C-B} \leq C_C) && \ldots . \text{ from (C–B:2f), (1a.2)} \\
\Rightarrow \quad & B_B \leq C_C
\end{aligned}$$

So B and C will only forward a frame if $B_B \leq C_C$; and C, B, and A will only forward the frame if $A_A \leq B_B \leq C_C$. This rules out any possibility of a loop through A and C, or A and any subsequent bridge on the path being one and the same, or A and C being one and the same.

Clearly eqn. 1 has a closely related alternative (eqn. 3): each bridge is (a) not closer to the destination than it has advertised to its downstream neighbour as the minimum distance from which it would forward to that neighbour (b) not further from the destination than those distances as advertised to it by each of its upstream neighbours.

$$\begin{aligned}
\text{spUnicast}_{Y-Z} \quad \equiv \quad & (Y_{Y-Z} = Y_Y) && \wedge \quad \ldots . \text{(3a.1)} \\
& (\forall Y_i \in a_{Y\rightarrow Z} ((Z_i < Y_i) \wedge (Y_{i.Y-Z} \leq Y_Y))) && \ldots . \text{(3a.2)} \\
& \wedge (\forall X \in n_Y && \ldots . \text{(3b)} \\
& (\exists a_i \in a_{Y\leftarrow X} : ((X_i < Y_i) \vee (Y_Y < X_{i.X-Y}))\ ) && \ldots . \text{(3b.1)}
\end{aligned}$$

Although the eqn. 1 rule may appear to require less agreement, I believe there is no practical performance difference: messages to discard prior agreements are needed when the change in the distance would be significant enough to require fresh agreements in eqn. 3.

Equations 1 and 3 leave open the possibility of each bridge keeping a separate record of every agreement sent or received, so that each can be individually discarded. Practical agreement protocols are not so selective: it is only necessary to remember the most constraining agreement sent and the most permissive one received. With this change eqn. 1 can be rewritten as eqn. 4, (below):

$$\begin{aligned}
\text{spUnicast}_{Y-Z} \quad \equiv \quad & (Y_{Y-Z} = Y_Y) && \wedge \quad \ldots . \text{(4a.1)} \\
& (\underline{Y}_{Y\leftarrow Z} \leq Y_Y) && \wedge \quad \ldots . \text{(4a.2)} \\
& (\forall X \in n_Y && \ldots . \text{(4b)} \\
& (Y_Y < \underline{X}_{Y\rightarrow X})) && \ldots . \text{(4b.1)}
\end{aligned}$$

and eqn. 3 can be rewritten as eqn. 5:

$$\begin{aligned}
\text{spUnicast}_{Y-Z} \quad \equiv \quad & (Y_{Y-Z} = Y_Y) && \wedge \quad \ldots . \text{(5a.1)} \\
& (\overline{Y}_{Y\rightarrow Z} \leq Y_Y) && \wedge \quad \ldots . \text{(5a.2)} \\
& (\forall X \in n_Y && \ldots . \text{(5b)} \\
& (Y_Y < \overline{X}_{Y\leftarrow X})) && \ldots . \text{(5b.1)}
\end{aligned}$$

Note that (4b.1) does not say that Y has to be closer to the destination than any of its neighbours, merely that it has to be closer than any of its neighbours to which Y has advertised itself as a possible next hop to the destination.

Similarly (5b.1) requires Y to be closer to the destination than any neighbour that has told Y that it considers Y a possible next hop to the destination can be when that neighbour's distance allows it to forward frames to Y.

It is easy to see that equations 1 and 4 guarantee loop-free forwarding. Consider bridges C, B, A lying (successively) on a path to a destination and recall that $(a_{C\leftarrow B} \subseteq a_{B\rightarrow C}) \Rightarrow (\underline{C}_{B\rightarrow C} \leq \underline{C}_{C\leftarrow B})$. C, and subsequently B, will only forward a frame if:

$$
\begin{aligned}
&spUnicast_{C-B} \wedge spUnicast_{B-A} \\
&\quad \Rightarrow (B_B < \underline{C}_{B\rightarrow C}) && \ldots\ldots\ldots \text{(B–A:4b.1)} \\
&\quad \wedge \\
&\quad (\underline{C}_{C\leftarrow B} \leq C_C) && \ldots\ldots\ldots \text{(C–B:4a.2)} \\
&\quad \Rightarrow B_B \leq C_C
\end{aligned}
$$

And C, B, and A will only forward the frame if $A_A \leq B_B \leq C_C$ ruling out any possibility of a loop through A and C, or A and any subsequent bridge on the path being one and the same.

Similarly using eqn. 5 and $(a_{C\leftarrow B} \subseteq a_{B\rightarrow C}) \Rightarrow (\overline{C}_{B\leftarrow C} \leq \overline{C}_{C\rightarrow B})$:

$$
\begin{aligned}
&spUnicast_{C-B} \wedge spUnicast_{B-A} \\
&\quad \Rightarrow (B_B < \overline{C}_{B\leftarrow C}) && \ldots\ldots\ldots \text{(B–A:5b.1)} \\
&\quad \wedge \\
&\quad (\overline{C}_{C\rightarrow B} \leq C_C) && \ldots\ldots\ldots \text{(C–B:5a.2)} \\
&\quad \Rightarrow B_B \leq C_C
\end{aligned}
$$

3.2 Destination-based unicast forwarding with split-horizon

If a bridge's forwarding logic never forwards a frame back through the reception port, eqn. 4 can be simplified to give eqn. 6 below.

$$
\begin{aligned}
&spUnicast_{Y-Z} \equiv (Y_{Y-Z} = Y_Y) && \wedge \quad \ldots\ldots \text{(6a.1)} \\
&\quad (\underline{Y}_{Y\leftarrow Z} \leq Y_Y) && \wedge \quad \ldots\ldots \text{(6a.2)} \\
&(\forall X \in n_Y && \ldots\ldots\ldots \text{(6b)} \\
&\quad (Y_Y < \underline{X}_{Y\rightarrow X})) && \ldots\ldots \text{(6b.1)}
\end{aligned}
$$

3.3 Multipath destination-based unicast forwarding

Equation 4 does not explicitly restrict forwarding to a single 'up' port, except in as much as the distance metric is specified as providing a total ordering (to ensure that tie-breaking is always supported) so only one port can meet the criterion (4a.1). Such a restriction would be unnecessary if the prevention of loops were the only concern. Although any given frame should only be transmitted through a single 'up' port to avoid duplicate delivery of frames, the choice of best port can be made on a frame by frame basis, as several ports could satisfy (4a.2). The agreement protocol and rules of the previous section thus allow frames for different flows to be distributed, per hop, on multiple paths to a single destination. The multiple paths can be "equal cost" or "near equal cost" within the constraints of eqn. 4.

3.4 Spanning tree forwarding

Equation 7 (below) specifies an stForwarding$_{\boldsymbol{X}-Y-\boldsymbol{Z}}$ forwarding rule that can be consider as defining both a truth value for any pair of bridge Y's ports and a set of port tuples for bridge Y. For example:

stForwarding$_{\boldsymbol{X}-Y-\boldsymbol{Z}}$ = True for any given $\boldsymbol{X}$, Y, $\boldsymbol{Z}$ iff frames can be forwarded from p_{Y-X} to p_{Y-Z}.

$[p_{Y-\boldsymbol{X}}, p_{Y-\boldsymbol{Z}}]$ $\in$ stForwarding$_{X-Y-Z}$ iff frames can be forwarded from p_{Y-X} to p_{Y-Z}.

stForwarding$_{\boldsymbol{X}-Y-\boldsymbol{Z}}$	$\equiv$	$(\boldsymbol{X} \neq \boldsymbol{Z}) \wedge$ (		. . . . . .(7a)
(	$(\exists Z \in \boldsymbol{Z}$: (	$(Y_{Y-Z} = Y_Y)$	$\wedge$	. . . . (7b.1)
		$(\forall\ a_j \in a_{Y\rightarrow \boldsymbol{Z}}$: $(Z_j < Y_j)$ $(Y_{j.Y-Z} \leq Y_Y))$	$\wedge$	. . . . (7b.2)
	$(\forall\ X \in \boldsymbol{X}$ (	$(\exists a_j \in a_{Y\leftarrow X}$: $((Y_j < X_j) \wedge (Y_Y < X_{j.X\text{-}Y})))$	$\wedge$	. . . .(7b.3i)
		$(\forall\ a_i \in a_{Y\rightarrow X}\ Y_i < X_i)$))		. . . (7b.3ii)
) $\vee$				
(	$(\exists X \in \boldsymbol{X}$: (	$(Y_{Y-X} = Y_Y)$	$\wedge$	. . . . (7c.1)
		$(\forall\ a_j \in a_{Y\rightarrow \boldsymbol{X}}$: $(X_j < Y_j)(Y_{j.Y-X} \leq Y_Y))$	$\wedge$	. . . . (7c.2)
	$(\forall\ Z \in Z$ (	$(\exists a_j \in a_{Y\leftarrow Z}$: $((Y_j < Z_j) \wedge (Y_Y < Z_{j.Z\text{-}Y})))$	$\wedge$	. . . .(7c.3i)
		$(\forall\ a_i \in a_{Y\rightarrow Z}\ Y_i < Z_i)$))		. . . (7c.3ii)
) $\vee$				
(	$(\forall\ X \in \boldsymbol{X}$ (	$(\exists a_j \in a_{Y\leftarrow X}$: $((Y_j < X_j) \wedge (Y_Y < X_{j.X\text{-}Y})))$	$\wedge$	. . . .(7d.1i)
		$(\forall\ a_i \in a_{Y\rightarrow \boldsymbol{X}}\ Y_i < X_i)$))	$\wedge$	. . . (7d.1ii)
	$(\forall\ Z \in \boldsymbol{Z}$ (	$(\exists a_j \in a_{Y\leftarrow Z}$: $((Y_j < Z_j) \wedge (Y_Y < Z_{j.Z\text{-}Y})))$	$\wedge$	. . . .(7d.2i)
		$(\forall\ a_i \in a_{Y\rightarrow Z}\ Y_i < Z_i)$))		. . . (7d.2ii)
))				

Equation 7 has four principal components: (a) requires p_{Y-X} and port p_{Y-Z} to be attached to different LANs, a condition that applies to alternatives (b), (c), and (d) following; (b) permits forwarding from a Designated Port to the bridge's Root Port; (c) forwarding from the Root Port to a Designated Port; and (d) forwarding between two Designated Ports. Criteria (b.1) and (c.1) require that there be a single Root Port, at least for any given frame (see also 3.3 and footnote[1]), while (b.2) and (c.2) require that any agreement sent by that Root Port that could be used by a Designated Port (in c.3i, b.3i) not overstate the distance that Y requires to become forwarding through the Root Port. In mathematical English (b.2) and (c.2) can be rendered: "for all agreements sent such that Y is not claiming to be a Designated Port, Y's quoted distance for use of that port as the shortest path is less than Y's current distance". Criteria (b.3i), (c.3i), (d.1i), (d.2i) require that Y hold suitable agreements for each Designated Port that is forwarding: "for all (other) bridges attached to the LAN there exists an agreement such that that bridge recognizes that I have a better right to be the Designated Port for the LAN and that bridge is quoting a distance at which it can forward frames to the LAN that is greater than my current distance". Criteria (b.3ii), (c.3ii), (d.2ii) require that Y has no outstanding agreement that another bridge could interpret as granting that bridge a better right to be the Designated Port for the LAN.

While stForwarding$_{\boldsymbol{X}-Y-\boldsymbol{Z}}$ specifies forwarding for all pairs of Y's ports (p_{Y-X} to p_{Y-Z}) it is clearly symmetric—stForwarding$_{\boldsymbol{X}-Y-\boldsymbol{Z}}$ $\Rightarrow$ stForwarding$_{\boldsymbol{Z}-Y-\boldsymbol{X}}$—so the rule can be rewritten (eqn. 8) to express the forwarding condition (both ingress and egress permitted) for any port p_{Y-Z}, provided that it is understood that a frame is never forwarded back through the receiving port. Port p_{Y-Z} is clearly either a Root Port (8a) meeting the criteria of (7b.1 and 7b.2) or a Designated Port (8b) meeting the criteria common to (7b.3), (7c.3), (7d.1), and (7d.2).

[1] Spanning Vines, 5th March 2002, discusses the use of the Alternate Ports of a single tree to provide multi-tree connectivity on a frame-by-frame basis.

Link state agreement

$$
\begin{aligned}
&\text{stForwarding}_{Y-\boldsymbol{Z}} \quad \equiv \\
&(\exists Z \in \boldsymbol{Z} : (\quad (Y_{Y-Z} = Y_Y) && \ldots . \text{ (8a.1)} \\
&\qquad \wedge \quad (\forall\, a_j \in a_{Y\rightarrow \boldsymbol{Z}} : (Z_j < Y_j)(Y_{j.Y-Z} \leq Y_Y)) \quad \vee && \ldots . \text{ (8a.2)} \\
&(\forall\, Z \in Z \quad (\quad (\exists a_j \in a_{Y\leftarrow Z} : ((Y_j < Z_j) \wedge (Y_Y < Z_{j.Z\text{-}Y}))) && \ldots . \text{ (8b.1)} \\
&\qquad \wedge \quad (\forall\, a_i \in a_{Y\rightarrow Z} : (Z_i < Y_i)(Y_{j.Y-Z} \leq Y_Y))) && \ldots . \text{ (8b.2)}
\end{aligned}
$$

Since practical agreement protocols do not select agreements that are to be discarded from arbitrary positions in the protocol history, but only need to remember the most constraining agreement sent and the most permissive agreement received, eqn. 8 can be simplified to yield eqn. 9 just as eqn. 4 was derived from eqn. 1.

$$
\begin{aligned}
\text{stForwarding}_{Y-\boldsymbol{Z}} \quad \equiv \quad & (\exists Z \in \boldsymbol{Z} : ((Y_{Y-Z} = Y_Y) \wedge (\overline{Y}_{Y\rightarrow Z} \leq Y_Y))) \quad \vee && \ldots . \text{ (9a)} \\
& (\forall\, Z \in \boldsymbol{Z}\, ((Y_Y < \overline{Z}_{Y\leftarrow Z}) \wedge (\overline{Y}_{Y\rightarrow Z} \leq Y_Y))) && \ldots . \text{ (9b)}
\end{aligned}
$$

While $\text{stForwarding}_{\boldsymbol{X}-Y-\boldsymbol{Z}}$ and $\text{stForwarding}_{Y-\boldsymbol{Z}}$ naturally cover both shared media and point-to-point connectivity, a slight simplification is possible in the latter case (eqn. 10):

$$
\begin{aligned}
\text{stForwarding}_{Y-Z} \quad \equiv \quad & ((Y_{Y-Z} = Y_Y) \wedge (\overline{Y}_{Y\rightarrow Z} \leq Y_Y)) \quad \vee && \ldots . \text{ (10a)} \\
& ((Y_Y < \overline{Z}_{Y\leftarrow Z}) \wedge (\overline{Y}_{Y\rightarrow Z} \leq Y_Y))) && \ldots . \text{ (10b)}
\end{aligned}
$$

If bridge B (say) is forwarding frames to and from a LAN through a Designated Port, while bridge C is forwarding to and from the same LAN using its Designated Port, and using $(a_{C\leftarrow B} \subseteq a_{B\rightarrow C}) \Rightarrow (\overline{C}_{B\leftarrow C} \leq \overline{C}_{C\rightarrow B})$ we have:

$$
\begin{aligned}
& ((B_B < \overline{C}_{B\leftarrow C}) \wedge (\overline{B}_{B\rightarrow C} \leq B_B))) \quad \wedge && \ldots . \text{ (B–C:10b)} \\
& ((C_{C-B} = C_C) \wedge (\overline{C}_{C\rightarrow B} \leq C_C)) && \ldots . \text{ (C–B:10a)} \\
\Rightarrow \quad & B_B < \overline{C}_{B\leftarrow C} \leq \overline{C}_{C\rightarrow B} \leq C_C \\
\Rightarrow \quad & B_B \leq C_C
\end{aligned}
$$

so, just as with eqn. 5, there can be no loop of bridges connected Root Port to Designated Port to Root Port etc. To prove that loops do not occur we also need to show that there can be at most one forwarding Designated Port for any given LAN. If B and C were both forwarding Designated Ports we would have:

$$
\begin{aligned}
& ((B_B < \overline{C}_{B\leftarrow C}) \wedge (\overline{B}_{B\rightarrow C} \leq B_B))) \quad \wedge && \ldots . \text{ (B–C:10b)} \\
& ((C_C < \overline{B}_{C\leftarrow B}) \wedge (\overline{C}_{C\rightarrow B} \leq C_C))) && \ldots . \text{ (C–B:10b)} \\
\Rightarrow \quad & B_B < \overline{C}_{B\leftarrow C} \leq \overline{C}_{C\rightarrow B} \leq C_C \quad \Rightarrow \quad B_B < C_C \\
& \text{and} \\
& C_C < \overline{B}_{C\leftarrow B} \leq \overline{B}_{B\rightarrow C} \leq B_B \quad \Rightarrow \quad C_C < B_B
\end{aligned}
$$

a contradiction, so there can be at most one forwarding Designated Port per LAN.

The question naturally arises as to whether there are complementary alternate versions of these rules using $\underline{Y}$ rather than $\overline{Y}$, in the same way that eqn. 4 is an alternative to eqn. 5, with positive agreements being received on Root Ports (‘up’ ports) while Designated Ports (‘down’ ports) are only constrained by the agreements they send rather than those they received. However, unlike the known unicast destination case, frames are transmitted through ‘down’ ports as well as through ‘up’ ports and it is essential that there be at most one forwarding Designated Port per LAN. If we wish to minimize the recorded state by using the same limit ($\overline{Y}$ or $\underline{Y}$) for both spanning tree and known unicast forwarding I believe we have to use rules for $\overline{Y}$ (and their derivatives).

3.5 Spanning tree forwarding with explicit priority vector signaling

Among its other uses, the CIST (strictly speaking the IST) is intended to provide remedial connectivity within MST and SPT Regions. While link state procedures can be used to compute the IST more rapidly than is often possible with distance vector, the CIST's dependencies on anything that might possibly go wrong (continuous churn in some part of the network, for example) or be misconfigured should be kept to a minimum. Since interoperability with SST and MST bridges at the boundary of an SPT Region requires the use of RST/MST BPDUs, the agreements for the CIST are carried as explicit priority vectors exactly as they are with RSTP/MSTP. Carrying CIST agreements in this way has two consequences. First, the loop-free guarantee does not depend on the bridges connected to a LAN agreeing on the port path cost of that LAN. Second, since the port path cost is not communicated in the BPDU, the forwarding rules cannot depend on it either: a bridge cannot calculate the minimum distance at which its neighbour may forward frames towards it but has to depend simply on the agreement information sent by that neighbour. In short $\overline{Z}_Z$ (as explicitly communicated in the agreement) is used in place of $\overline{Z}_{Y\leftarrow Z}$ (as calculated from an agreement digest) in the forwarding rules of eqn. 9 and eqn. 10.

This chance has no effect on the permitted connectivity when an agreement is sent on a Root Port, but does mean that temporary discarding will more often occur (following some LAN failure) on Designated Ports that connect to Alternate Ports. This unwanted effect can be avoided by changing the priority vector sent through Alternate Ports so it actually reflects $\overline{Z}_{Y\leftarrow Z}$, i.e. Z's own stated view of the distance at which it can become forwarding on the port with first hop Y) rather than Z's current distance. However the possible confusion arising from making such a change in the standard specification makes it probably not worthwhile. A network that contains (some or all) bridges that do send $\overline{Z}_{Y\leftarrow Z}$ will work perfectly well since $\overline{Z}_{Y\leftarrow Z}$ can never be less than (better than) $\overline{Z}_Z$.

3.6 Multicast forwarding with source specific multicast addresses

Shortest path multicast forwarding requires some identification of the source, and in SPBM this is provided by using a source-specific multicast destination address. In networks of point-to-point LANs it is possible to use egress filtering for such a destination address to provide loop-free multicast connectivity, without the need to check the source address of the frame on ingress. Considering distances for the tree rooted at the source:

$$\begin{aligned}
ssMulticast_{Y-Z} \quad &\equiv \quad (Z_Y = Z_{Y-Z}) && \wedge \quad \ldots\ldots(11a)\\
&\quad\ (\exists a_j \in a_{Y\leftarrow Z} : ((Z_j = Z_{j.Y-Z}) \wedge (Y_Y < Z_{j.Y-Z}))) && \wedge \quad \ldots\ldots(11b)\\
(\forall X \in n_Y : (Y_Y \neq Y_{Y-X})\ &(\forall a_j \in a_{Y\rightarrow X}\ (Y_j \neq Y_{j.Y-X}))) && \ldots\ldots(11c)
\end{aligned}$$

Equation 11 says that Y will forward the source specific multicast through any port p_{Y-Z} to bridge Z, provided that: (a) Y believes that the shortest path from Z to the source lies through Y itself; and (b) Y holds at least one agreement to that effect from Z; and (c) Y itself has such agreements (for the tree in question) outstanding at only one of Y's neighbours, which provides Y's own next hop towards the source. In summary, Y ensures that it will not forward more than one copy of a sourced multicast by removing all forwarding database entries permitting egress for that multicast if Y has told more than one parent that that parent is Y's shortest path to the source.

While it is easy to detect (and record) when condition (11c.1) first inhibits forwarding, restarting forwarding is not so easy unless the strategy is to wait until all ports have synchronized (see protocol) with their neighbours. Furthermore condition (11b) makes determining the neighbour's Root Port an essential part of the loop prevention process, while it might be more convenient not to require that the link state computation share that information, or treat it as an optimization. Equation 12 uses source address ingress filtering to address both these issues, allowing the multicast to be forwarded on some LANs only to be discarded by the neighboring bridge:

$$\begin{aligned}
ssMulticast_{X-Y-Z} \quad &\equiv \quad (Y_{Y-X} = Y_Y) && \wedge \quad \ldots\ (12a.1)\\
&\quad\ (\forall a_i \in a_{Y\rightarrow X} : (X_i < Y_i)(Y_{i.Y-X} \leq Y_Y)) && \wedge \quad \ldots\ (12a.2)\\
&\quad\ (\exists a_j \in a_{Y\leftarrow Z} : ((Y_j < Z_j) \wedge (Y_Y < Z_{j.Z-Y})) && \wedge \quad \ldots\ldots(12b)\\
&\quad\ (Z_Y = Z_{Y-Z}) && \ldots\ldots(12c)
\end{aligned}$$

Equation 12 specifies forwarding from p_{Y-X} through one or more ports p_{Y-Z} : (a.1) p_{Y-X} provides the shortest path to the source (and is the only one of Y's ports that does so); (a.2) Y is no closer to X than its minimum forwarding distance as specified in all agreements outstanding on p_{Y-X} with X<Y; (b) Y holds such an agreement from Z, with Z's minimum forwarding distance greater than Y's current distance; (c) an optimization not required to prevent loops or frame duplication, do not forward to Z unless (in the current topology calculated by Y) the frame is being forwarded to Z's Root Port.

Equation 13 is eqn. 12 simplified (as with previous equations) to record only the most constraining agreement sent and the most permissive received.

$$\begin{aligned} ssMulticast_{X-Y-Z} \quad \equiv \quad & (Y_{Y-X} = Y_Y) \ \wedge \ (\overline{Y}_{Y\rightarrow X} \leq \ Y_Y)) && \wedge \quad \ldots\ldots (13a) \\ & (Y_Y < \ \overline{Z}_{Y\leftarrow Z}) && \wedge \quad \ldots\ldots (13b) \\ & (Z_Y = Z_{Y-Z}) && \quad \ldots\ldots (13c) \end{aligned}$$

Consider bridges B and C connected to the same LAN. Neither can be forwarding frames received from that LAN back to the same LAN (X and Z the same) as (13a) requires $(\overline{Y}_{Y\rightarrow X} \leq \ Y_Y)$ while (13b) requires $(Y_Y < \ \overline{Z}_{Y\leftarrow Z})$. Consider B forwarding frames to the LAN, and C receiving the frames and forwarding to a further LAN. Instantiating (13b) for B's port B–C and (13a) for C's port C–B $\Rightarrow \ B_B < \ C_C$, so no forwarding loop is possible.

4. Protocol

While it is not my intent to depart from the agreement protocol previously described and documented in P802.1aq/D2.5, this note does provide additional detail and the protocol description in this note may differ (particularly in its use of local variable names). I have felt free to question and possibly improve on the latter as a part of a diligent search for the best description and have devoted little if any time to mechanically aligning terminology. On the other hand this note's attempt at an economical description may not be the easiest to integrate with existing specifications and implementations (or might over constrain the latter). Questions as to if or how P802.1aq should be changed should be considered separately.

4.1 General terminology

participant — a participant in the agreement protocol.

actor — the participant under discussion

partner — a participant other than the actor, but exchanging messages with the actor and potentially with other participants attached to the same LAN.

4.2 Messages and rules

The forwarding rules specify what ports (or pairs of ports) can forward (transmit and/or receive) data frames given a set of agreements received (and not discarded) and a set of agreements transmitted (and not yet explicitly discarded by their recipient). The rules can equally well be read as specifying what fresh agreements can be transmitted, and what received agreements discarded, given the forwarding ports. The protocol places the control of whether an agreement is to be discarded in the hands of the participant that has received that agreement. The mere calculation of a new active topology does not force a crisis so far as a participants is concerned, the current forwarding pattern can proceed uninterrupted. Once the forwarding pattern has been reduced to one that accords with that required if fresh agreements were to be sent, or held agreements discarded, then those agreements can be sent and discarded as desired.

4.3 Forwarding rules

The algorithms and protocol discussed in this section (4) are designed to cover loop-free forwarding requirements for the use of SPBV with point-to-point links and shared media, and the use of both SPBV and SPBM with point-to-point links. Spanning tree forwarding is supported both by digests and explicit information to allow the use of the latter to support the IST, maximizing the chance of providing remedial connectivity should some complex convergence problem occur. SPBM's source specific multicast forwarding is supported without relying on a source ingress check to avoid any dependence on unicast FDB entries, thus not constraining the use of SPBM unicast loop mitigation as an alternative to loop-free guarantees. Specifically this section covers:

a) Simple shortest path destination-based unicast forwarding and (hop-by-hop) multipath unicast (3.1 and 3.3) over point-to-point links, using the spUnicast_{Y-Z} rule of eqn. 5 and agreement digests.

b) Spanning tree forwarding using both point-to-point and shared media (3.4, 3.5) using the $\text{stForwarding}_{Y-Z}$ rule of eqn. 9 with both agreement digests and explicit priority vectors.

c) Multicast with a source-specific multicast address (3.6) using the ssMulticast_{Y-Z} rule of eqn. 13 and agreement digests.

Note, at present the details are only shown for point-to-point links to avoid the clutter that comes with keeping track of multiple partners.

The relevant rules are repeated below:

$$\text{spUnicast}_{Y-Z} \equiv (Y_{Y-Z} = Y_Y) \wedge \quad \ldots. (5a.1)$$

$$(\overline{Y}_{Y\rightarrow Z} \leq Y_Y) \wedge \quad \ldots. (5a.2)$$

$$(\forall X \in n_Y \quad \ldots\ldots (5b)$$

$$(Y_Y < \overline{X}_{Y\leftarrow X})) \quad \ldots. (5b.1)$$

$$\text{stForwarding}_{Y-Z} \equiv ((Y_{Y-Z} = Y_Y) \wedge (\overline{Y}_{Y\rightarrow Z} \leq Y_Y)) \vee \quad \ldots\ldots (10a)$$

$$((Y_Y < \overline{Z}_{Y\leftarrow Z}) \wedge (\overline{Y}_{Y\rightarrow Z} \leq Y_Y))) \quad \ldots\ldots (10b)$$

Link state agreement

$$ssMulticast_{X-Y-Z} \equiv (Y_{Y-X} = Y_Y) \wedge (\overline{Y}_{Y\rightarrow X} \leq Y_Y)) \wedge \quad .....(13a)$$

$$(Y_Y < \overline{Z}_{Y\leftarrow Z}) \wedge \quad .....(13b)$$

$$(Z_Y = Z_{Y-Z}) \quad .....(13c)$$

4.4 Participant state

The link state procedures for any given protocol participant, Y, provides that participant with the following per tree information when each link state calculation completes:

- designatedPriority — Y_Y, see definition.
- calculatedDigest — g_Y, see definition.

and the following per tree for each port (identified as, or as one of, p_{Y-Z} or p_{Y-X} in the forwarding rules:

- selectedRole — RootPort iff (
- neighbourPriority — Z_Y or X_Y see definition, calculated (for port from the neighbour Priority provided.

The agreement protocol also calculates the following per tree per port from the information provided.

- rootPriority — Y_{Y-Z}, see definition, calculated (for port from the neighbour Priority provided.[1]

The following per tree per port information records the agreements received (and held) and those transmitted (and still outstanding).

- agreedPriority — $\overline{X}_{Y\leftarrow X}$, see definition, in (5b.1), and $\overline{Z}_{Y\leftarrow Z}$ in (9b), (10b), (13b), also see 3.5.
- outstandingPriority — $\overline{Y}_{Y\rightarrow Z}$, see definition, in (5a.2), (9a), (10a).

The transmission and reception of agreements is recorded per port. Those inherent in the transmission and reception of digests are simply recorded by noting the digests involved:

- txDigest — the digest currently being transmitted.
- rxDigest — the last digest received.

The following boolean variables[2] can be used to summarize the agreement state per tree per port:

- agreedU = (designatedPriority < agreedPriority) // i.e. iff (5b.1), (10b), (13b)
- agreed = (agreedU && (outstandingPriority < designatedPriority)) // i.e. iff (10b),
- agreedM = (agreedU && (neighbourPriority == (designatedPriority + portCost)) // i.e. iff (13b and 13c).
- syncedU = (agreedU && (the FDB does not permit egress for frames with the unicast address))
  || (agreed && ((neighbourPriority + portCost) == designatedPriority))
  // the port is a Root Port or a multi-path forwarding Alternate Port
- synced = agreed || (the Port State is discarding for frames subject to spanning tree forwarding).
- syncedM = agreedM || (the FDB has no egress for frames with the source-specific multicast address).
- allSyncedU = syncedU is True for all ports.
- allSynced = syncedU is True for all ports (other than the Root Port if this is the Root Port[3]).
- allSyncedM = syncedM is True for all ports.
- agree = allSyncedU && allSynced && allSyncedM

Once agree is True for all trees txDigest can be updated to calculatedDigest.

[1] P802.1aq/D2.5 currently defines rootPriority only for the Root Port, thus making it a per tree variable.
[2] The variables agreed, synced, allSynced, and agree were originally specified for RSTP, then used in MSTP, and are specified for SPB in 802.1aq /D2.5.
[3] I have been around this aspect of defining allSynced several times and need to check back with 802.1Q and its revision history.

4.5 Agreement messages

For the time being I have left out the explicit priority vector agreements, and just described the digest handling.

The protocol has to deal with the fact that there may be a number of messages ‘in flight’ or at least languishing in transmission or reception buffers awaiting processing. The topology (and the accompanying digest) can change and change back again, so the reception of a matching digest in an agreement message is not sufficient to indicate that a protocol partner is not holding additional agreements.

Each agreement message, however it is carried in protocol frames, comprises the following parameters:

— digest — the agreement digest.

— an — the agreement number, drawn from a small circular sequence number space, for subsequent use by the receiver of the message in the dan field.

— dan — the discarded agreement number.

The ‘discarded agreement number’ identifies the last agreement digest processed by the prior to discarding agreements that contradict or are not a subset of those expressed by the digest. Specifically the transmitted and received digest, an, and dan, together with the last calculatedDigest identify the following conditions.

- txDigest != calculatedDigest

  The forwarding carried out by the bridge (node) has not yet been reduced to a pattern that is a subset of of that in the topology identified by the calculatedDigest and permitted by outstanding and held agreements.

- tx.dan == rx.an

  The agreements currently held by the actor are those required by (or a subset of those required by) txDigest and the received message defined by rx.an. In other words txDigest was calculated (and the agreements held filtered by the corresponding topology) after the received message with rx.an was received

- tx.dan == rx.an - 1

  The link state calculation identified by txDigest was completed, and conflicting received agreements discarded, prior to receipt of the rxDigest identified by rx.an. In other words the actor may be holding received agreements that conflict with txDigest, though these will be a subset of the union of those permitted by txDigest and those permitted rxDisgest.

- tx.dan == rx.an - 2

  The actor may be still holding received agreements that conflict with txDigest since it has not recognized (or completed processing for) any recent digest. If the actor receives additional digests (with inceremented rx.an) that do not match its calculatedDigest it will increment tx.dan so it continues to lag tx.dan at rx.an - 2, thus rotating the sequence number so that its partener can send further agreement messages.

5. Misordering

<<Extend sequence numbering beyond the bare minimum, discussion and definition of ‘modest misordering’, impossibility of full misordering protection in any circular sequence number space protocol (why not mentioned elsewhere, e.g. for TCP?).>>

6. To be done

A list of outstanding issues and remaining/known work:

1) Special case ‘next hop is the destination’ and possibly ‘last hop was directly from the source’.

A. Mathematical symbols

To write this note I had to spend some time finding out how to enter various symbols in Framemaker. This is by no means as easy as it should be, and web searches turned up material that was partially correct at best[1]. As a memo to myself, possibly useful to others, this Annex lists some of the symbols and how to enter them (in Framemaker 8/ Windows XP). Unless otherwise specified the Symbol font is used. In the following description 'Shift+q" means (for example) hold down the Shift key and type 'q' (the character '+' is not actually typed). This notation follows that used in Framemaker documentation. I have included Unicode (UTF-16) code points and character descriptions in case the characters appear differently in some future version of Framemaker, or are rendered differently under some circumstances (display, version of Acrobat, printer driver, printer, etc.).

…	—	Control+q Shift+i. Unicode 2026, character 'ellipsis'.
←	—	Control+q Shift+b. Unicode 2190, 'arrowleft', 'LEFTWARDS ARROW'.
→	—	Control+q (. Unicode 2192, 'arrowright', 'RIGHTWARDS ARROW'.
↔	—	Control+q Shift+g. Unicode 2194, 'arrowboth', 'LEFT RIGHT ARROW'.
∀	—	Turn smart quotes off (Format > Document > Text Options), type a double quote □, use Symbol font. Smart quotes can be turned back on. Unicode symbol 2200, 'universal', 'FOR ALL'.
⇒	—	Control+q Shift+w. Symbol font. Unicode 21D2, 'arrowdblright', IMPLIES.
∃	—	$. Symbol font. Unicode 2203, 'existential', 'THERE EXISTS'.
∅	—	Control+q full-stop (period). Symbol font. Unicode 2205, 'empty set'.
∈	—	Esc ^ Shift+i. Symbol font. Unicode 2208, 'element', 'ELEMENT OF'.
∉	—	Esc % Shift+i. Symbol font. Unicode 2209, 'notelement', 'NOT AN ELEMENT OF'.
∞	—	Control+q 4. Symbol font. Unicode 221E, 'infinity'.
∧	—	Esc grave-accent Shift+u. Symbol font. Unicode 2227, 'logicaland'.
∨	—	Esc single-quote Shift+u. Symbol font. Unicode 2228, 'logicalor'.
∩	—	Esc comma Shift+c. Symbol font. Unicode 2229, 'intersection'.
∪	—	Esc grave-accent Shift+e. The grave-accent character is indistinguishable (in the Times Roman font used in most of this note) from an opening single quote, but on the keyboard is on the same key as ~. Symbol font. Symbol font. Unicode 222A, 'union'.
∴	—	\. Symbol font. Unicode 2234, 'therefore', 'THEREFORE'.
≈	—	Control+q Shift+h. Symbol. Unicode 2248, 'approxequal', 'ALMOST EQUAL TO'.
≠	—	Control+q 6. Unicode 2260, 'notequal', 'NOT EQUAL TO'.
≡	—	Control+q <. Unicode 2261, 'equivalence', 'IDENTICAL TO'.
≤	—	Control+q #. Unicode 2263, 'lessequal', 'LESS THAN OR EQUAL TO'.
≥	—	Control+q 8. Unicode 2264, 'greaterequal', 'GREATER THAN OR EQUAL TO'.
⊂	—	Esc grave-accent Shift+i. Unicode 2282, 'propersubset', 'SUBSET OF'.
⊃	—	Esc single-quote Shift+e. Unicode 2283, 'propersuperset', 'SUPERSET OF'.
⊆	—	Esc single-quote Shift+i. The single-quote 'character is on the same key as ". Symbol font. Unicode 2286, 'reflexsubset'.
¬	—	Control+q /. Symbol font. Unicode 2310, 'logicalnot'.

[1]The most useful (though error ridden) reference I found was http://www.adobe.com/devnet/framemaker/pdfs/Character_Sets.pdf now archived in my Notes folder. This describes Framemaker 7.0 character sets. The Framemaker 8 distribution includes file:///C:/Program%20Files/Adobe/FrameMaker8/Documents/Character_Sets.pdf which has many additional errors. Other useful references are http://en.wikipedia.org/wiki/Unicode_Mathematical_Operators and http://en.wikipedia.org/wiki/Table_of_mathematical_symbols.

B. Alternative rules and formulations

There is a fair amount of fine detail, with accompanying choices and considerations, in the forwarding rules. It has not always been obvious to me which choices should be made. In particular the initially 'obvious' rules proved to be more constraining than necessary. This annex provides a place to record some of these choices and decisions, to lessen the chance of revisiting them in the future, as well as recording other changes as an audit trail in case I have made a mistake.

C. Distances, vectors etc.

While working on this note I noticed that there was a fair amount of scope for proliferation of various types of distances and limits on distances: $\overline{Y}_{Y\rightarrow Z}$ — the greatest value outstanding out of the distances that Y has communicated for itself to Z — being one example. This annex helps me to keep track of these, principally so that the examples used for the purpose of definition in 2 remain relevant, but also to spot possibly unnecessary variants.

Y_Y	— defined, used in eqn. and many (all) others
Y_{Y-Z}	— defined, used in eqn. a.1
X_i	— defined by Y_i, used throughout
Y_i	— defined, used throughout
Z_i	— defined by Y_i, used throughout
$X_{i.X-Y}$	— defined by $Y_{i.Y-Z}$, used in eqn. b.1, eqn. 1b.1, eqn. 3b.1
$Y_{i.Y-Z}$	— defined, used in eqn. a.2
$Z_{j.Y-Z}$	— defined, used in eqn. 11b.1
$\underline{Y}_{Y\leftarrow Z}$	— defined, used in eqn. 4a.2
$\underline{X}_{Y\rightarrow X}$	— defined, used in eqn. 4a.2
$\overline{Y}_{Y\rightarrow Z}$	— defined, used in eqn. 5
$\overline{Y}_{Y\rightarrow X}$	— defined by $\overline{Y}_{Y\rightarrow Z}$, not currently used
$\overline{X}_{Y\leftarrow X}$	— defined by $\overline{Z}_{Y\leftarrow Z}$, used in eqn. 5b.1
$\overline{Z}_{Y\leftarrow Z}$	— defined, used in eqn. 9b, eqn. 10b, eqn. 12c