Link state bridging

Mick Seaman

While link state routing protocols are less likely to create temporary loops than their distance vector predecessors, loops are still possible if, for example, two network links fail about the same time. This is a serious problem for bridging, particularly for multicast. This note proposes a complementary neighbour to neighbour Tree Agreement Protocol (TAP[1]), based on RSTP/MSTP's Proposal/Agreement[2], that ensures loop-free active topology(ies).

This proposal does not require bridging topology calculation by any particular protocol (link state or otherwise): IS-IS is used as a convenient place-holder, but TAP is independent of IS-IS packets and the frames that convey them[3]. IS-IS simply selects the *Root Identifier* and the *Root Port*[4] and provides the path cost to the root for one or more trees—a simple link state based single spanning tree replacement, multiple spanning trees if desired, and/or a tree per bridge for shortest path bridging[5]—in each bridge.

Each bridge port transmits TAP messages, with a *priority vector*[6] for each tree, to make **promises** to the other bridges connected to the same LAN: a *Designated Port* can **contract** that none of its bridge's connected[7] parents will agree to new *Root Port* with a *worse* priority, while a *Root Port* can **agree** that its bridge's connected children will not issue a contract with a *better* priority[8]. If a *Designated* and *Root Port's* **Contract** and **Agreement** match, loop-free connectivity can be created without delay, and loops avoided otherwise. Temporary *cuts* in connectivity allow promises to be made before contracts propagate all the way from the *Root*, or agreements all the way from *Edge Ports*. After IS-IS results are uniformly available, each port transmits at most one TAP message (for its neighbour to process, but not to respond to or propagate) to provide full connectivity[9].

TAP can construct loop free shortest path trees for unicast and source specific multicast using only destination address based egress checking on point-to-point links. Ingress checks are required for shared media, normal multicast (VID or source address based checks), and for frames forwarded on the SST or an MSTI (VID based only)[10].

LSPs are normally flooded throughout a network. Sequence numbers assigned by the source are checked at each node before forwarding, so looping LSPs do not consume excessive bandwidth. LSP distribution is thus independent of any forwarding that depends on the LSPs themselves, but slower than transmission over an existing tree. This note suggests initial transmission also over the shortest path tree rooted at the source, to provide rapid recovery from single link failure in an otherwise stable network[11]. Examples are provided, including automatic ring protection.

TAP messages are best carried in spanning tree BPDUs, so the boundary of the shortest path capable bridge region can be determined dynamically and plug and play interoperability with existing bridges provided. This note suggests a BPDU format.

---

[1] TAP is just a temporary working name/acronym for the purposes of this note, suggestions for a better name and acronym are welcome.

[2] I suggested MSTP Proposal/Agreement like mechanisms when P802.1aq started, but the crucial part of the solution—discarding a promise is new.

[3] Most important: the distribution of LSPs is not held up while calculations are being performed.

[4] All italicised terms are defined by IEEE Stds 802.1D and 802.1Q.

[5] If shortest path bridging is supported, each bridge's Root Port has to be selected so that the shortest path trees compose a congruent set, i.e. the path followed to any bridge A (say) by the shortest path tree rooted at any other bridge B is the same (but in reverse of course) as the path followed to B by the shortest path tree rooted at A. This ensures that shortest path transmit and receive between the two bridges follow the same path, and requires consideration of the whole path whenever a tie break between equal cost paths is required to select the Root Port.

[6] Loosely, the cost from the tree root, but including the Root Identifier (in case the root changes), and other tie breaker fields.

[7] Rootward connectivity can be cut, to allow local agreements.

[8] Parents and children can establish new connectivity that violates existing promises but are aware that they cannot take advantage of those promises, and hence first have to transition (an) other port(s) to Discarding, so they are no longer connected.

[9] As long as the LANs are point-to-point.

[10] Source address learning can be used with VID based checks.

[11] I am not the first to suggest tree distribution of LSPs, and increasing the number of LSPs is probably a bad idea in large routed networks: this note shows it can be very effective at protecting rings and other bridged networks.

1. Potential loops

Figure 1 shows the stable active topology of a tree, rooted at bridge 101, in a network with zero, one, then two link failures. Figure 2 is the resulting topology if bridges 606, 707, 808, and 909, are still unaware of the failures, bridges 202 and 303 area aware of the first, and 404 and 505 are aware of both.[1].

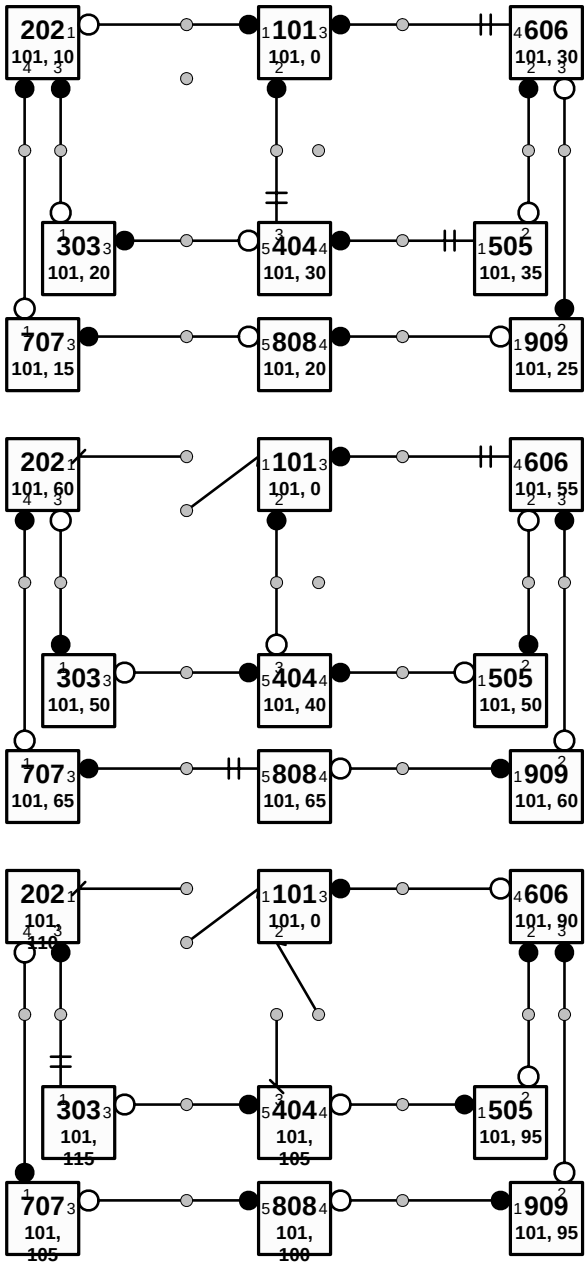

Figure 1—Tree topology with link failures

A loop (202-303-404-505-606-909-808-707-202) has formed. Moreover the loop would not be defeated if each port checked its neighbours *Port Role*, either by exchanging protocol or by performing a link state calculation for its neighbour (with its own knowledge of the links, refer to Figure 1). TAP prevents the loop by sustaining *cuts* in the active topology whenever a bridge's parent (connected by the bridge's Root Port) is further from the tree Root than the same bridge's child (connected by a Designated Port).

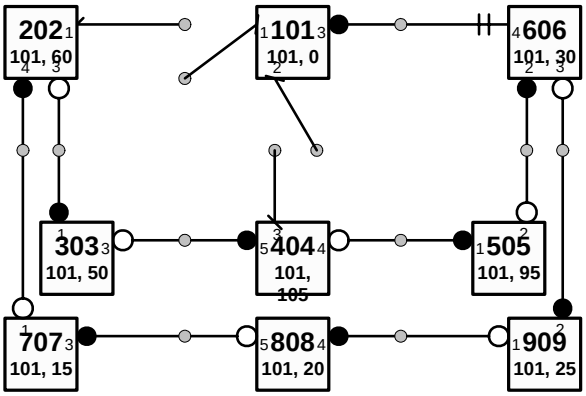

Figure 2—Forwarding loop

2. Promises, contracts, and agreements

Each bridge port transmits TAP messages, for each tree, to the other bridges attached to the same individual LAN. Each message includes a tree identifier, the *Port Role* (*Root*, *Designated*, *Alternate*, or *Backup*), the *Port State* (*Discarding*, *Learning*, or *Forwarding*[2]), a *Proposal* flag, a **Promise Flag**, a **Promise Number**, a **Discarded Promise Number,** and the transmitting port's *designated priority vector*[3] (i.e. the *Root Identifier* and *Root Path Cost* provided by IS-IS, and the port's own *Bridge Identifier* and *Port Identifier*[4]). If the **Promise Flag** is not zero, the priority vector constitutes a **promise** that is either an **Contract**[5] (if the transmitting port is *Designated*) or an **Agreement** (otherwise). A **promise** is held by a receiving port unless it has transmitted a TAP message with a matching or more recent **Discarded Promise Number,** and is **outstanding** at the transmitting port unless such a message has been received. A **Contract** is **applicable** if it is *better* than any **Contract outstanding** on any other port that is not *Discarding*. An **Agreement** is **applicable** if it is held by a port with no **outstanding Agreements.**

[1]I assume that the set of all possible topology protocols contains at least one protocol that is capable, following N changes in the physical network topology, of delivering to each active node (i.e. to each bridge in the present case) topology information that reflects an arbitrary (and possibly different) subset of those N changes, subject only to the requirement that the net effect of all changes will be delivered within a known bounded interval after the last change.

[2]*Disabled* is left out because a *Disabled* port neither transmits or receives. *Disabled* ports are treated as *Discarding* in the rest of this note.

[3]Just the first four components, as for RST/MST BPDUs.

[4]When a PDU conveys multiple TAP messages, the Bridge Priority and Port Priority are per tree: the Port Number and Bridge Address are not replicated.

[5]MSTP uses the term Agreement for promises that go down as well as up the tree, using a different term (Contract) for the former facilitates explanation.

A port that is attached to a point-to-point LAN can transition to *Forwarding* without delay if it is: (a) a *Root Port* that holds an **applicable Contract**, and any other port that is not *Discarding* holds a *worse* **applicable Agreement**; or (b) a *Designated Port* that holds an **applicable Agreement** and the bridge (1) holds a *better* **applicable Contract** on its *Root Port*, or (2) has a *root priority vector* (the *Root Identifier* and *Root Path Cost* from IS-IS, and the bridge's own *Bridge Identifier*, and a *Port Identifier* of 00-00) that is better than the **Agreement**, and (i) a *Discarding Root Port*; or (ii) is the *Root.*

A bridge can transmit a **Contract** on a *Designated Port* if it is: (a) the *Root*; or (b) the *Root Port* holds a **Contract** that is *better* than the bridge's own *root priority vector* or (c) if its *Root Port* or the *Designated Port* is *Discarding*. An **Agreement** can be transmitted on a *Root Port* if each of the bridge's other ports that is not *Discarding* is an *Edge Port* or holds an **Agreement** that is *worse* than the bridge's own *root priority vector.* An **Agreement** can also be transmitted on a *Root*, *Alternate*, or *Backup Port* if it is *Discarding*.

A bridge can receive and hold a received **promise** whether it agrees with the **promise** or not. For example, *Designated Port* can hold a **Contract**, or a *Root Port* an **Agreement**. Usually such a **promise** becomes useful when IS-IS next provides results. However when IS-IS completes a calculation, the bridge discards any **Agreement** held by a *Root Port* or **Contract** held by a *Designated Port*, and sends a TAP message with the **Discarded Promise Number**. A **promise** is also discarded, and a TAP message sent, whenever a subsequent **promise** is received.

3. Hasty promises

A bridge that lacks a **promise** on one port can still **promise** on another, if the first is made *Discarding*. Such temporary *cuts* speed reconfiguration, allowing promises to be made before they can propagate up (or down the tree). After IS-IS results are uniformly available, full connectivity for a new active topology is delayed by no more than the time taken by a bridge to receive one TAP message from its neighbour.

RSTP makes *cuts* at *Designated Ports*, but is easily changed to *cut* at a *Root Port* (to reduce the number of *cuts* required in a given bridge, at the cost of delaying connectivity through some ports, but without sacrificing interoperability). Cutting can also be delayed to give **Agreements** a chance to propagate up the tree, when that is likely to happen more quickly than a particular bridge implementation can make and then repair *cuts*. What should TAP specify?

A given data frame can be transmitted both up and down a single spanning tree or multiple spanning tree instance, and while each bridge only has to apply egress checks to ensure loop free transmission of unicast this would require cutting at both *Root Ports* (for rootbound[1] frames) and *Designated Ports* (for edgebound frames). However a shortest path destination rooted tree only requires *Root Port* egress checks, so *Root Port* cuts are preferred by TAP for both shortest path and other trees. An exception is made for non-Edge Ports attached to shared media LANs: these are cut because they cannot support rapid Contract Agreement, and would otherwise slow active topology provision over point to point links.

When IS-IS chooses a new *Root Port* for a shortest path tree it is always cut, pending receipt of an **applicable Contract**. However if the port was previously an Alternate Port in a stable active topology, that **Contract** should already be available. Otherwise the change of *Root Port* indicates that another port on the new path to the tree *Root* is *Discarding* and requires an **Agreement** before the new active topology is fully connected. By blocking its changed *Root Port*, the bridge can send that **Agreement** without waiting for **Agreements** from its own *Designated Ports*.

When a single spanning tree or multiple spanning tree instance chooses a new *Root Port*, that port is only cut if the *Root Port* for the shortest path tree for the SST or MSTI's *Root* has also been cut (or is already *Discarding*): thus avoiding unnecessary cutting, for example, when a *Root* at one end of a chain of bridges has been replaced by a new *Root* at the other end.

4. Promise numbers

Each bridge port maintains a per-tree **Promise Number** (PN) and **Discarded Promise Number** (DPN)**,** that are drawn from a 2 bit wrapping sequence number space (0,1,2,3,0,..) and transmitted in TAP messages for the tree. These numbers allow the port to identify **outstanding promises** it has made, to notify its neighbour of promises it has discarded, and to make fresh promises. No more than one received promise is held at a time, and message reception causes any existing promise to be discarded. If a received message is not a promise or its PN equals the port's DPN, the port's DPN is set to the received PN, and a

[1] A frame is rootbound in a particular bridge if it is travelling towards the root, and edge bound if it is travelling away from the edge.

message is transmitted for the tree; otherwise the received promise is held, and the DPN is set to received PN minus one (wrapped). A port's PN is not incremented for every TAP message sent, but only when: (a) the next PN is **available**, and the type of promise (None, **Contract** or **Agreement**) changes or a *worse* **Contract** is promised; (b) the next PN is now available, and would have been used for the last message if available then; (c) the current PN matches a received DPN. The next PN is available if a DPN has been received for any of the prior three PNs. Provided that the last received DPN is just prior to the current PN, the only **outstanding promises** are those transmitted since the current PN was last selected.

This **Promise Number** protocol assumes that the communications channel between neighbours is FIFO, a more complex protocol could be used if message reordering was possible.

5. Alternate and Backup Ports

There is no difference between the (lack of) connectivity provided by *Designated Ports* that are *Discarding* and of other *Discarding* ports. IS-IS is only used to identify *Root Ports*: all other ports are initially assumed *Designated*. TAP selects other *Role*s so *Alternate* and *Backup Ports* can send **Agreements**[1].

If a *Designated Port* receives a TAP message with a *better* priority vector and a *Designated Port Role* it changes its own *Port Role* to *Alternate* (if the message was from a port on another bridge) or *Backup* (otherwise), makes the port *Discarding* and transmits an **Agreement**. If a subsequent message from the same port has a priority vector that is worse than the receiving ports *designated priority vector*, the latter becomes *Designated* again (and transmits).

A *Designated Port* that receives an **Agreement** from an *Alternate* or *Backup Port* would normally become *Forwarding*, but that transition may serve little purpose[2] and can be suppressed so that multicast frames with source specific destination addresses can be forwarded on source rooted trees without requiring any ingress checking on point-to-point links.

6. Shared media

TAP, like RSTP/MSTP, does not attempt rapid *Forwarding* transitions on shared media. However if TAP is being used on a specific individual LAN it can assume that all bridges attached to the LAN are TAP capable, as any that are not are discovered to ensure interoperability. *Forwarding* transitions can then proceed on the usual control frame loss assumptions, with regular transmission and TAP's use and development of RSTP's *dispute* procedure (as follows) reducing the necessary *Forward Delay* to 2 seconds. If a *Learning* or *Forwarding Designated Port* (attached to shared media or a point-to-point LAN) receives a message that also claims to be from a *Learning* or *Forwarding Designated Port*, the port becomes *Discarding* again, the *Forward Delay Timer* is restarted, and a TAP message transmitted. Moreover a *Root Port* will revert to *Discarding* if it receives a **Contract** that is not **applicable** on receipt and either it or the **Contract's** transmitter was *Learning*, but not *Forwarding*.

7. Lost TAP messages

Except on shared media (see above) TAP messages are sent when required, and do not perform any per-tree keep alive function. The design goal is to use events to trigger message transmission to provide full connectivity rapidly and without relying on timers. A change in the physical topology will often trigger sufficient events that two messages have to be lost before recovery times are compromised. However messages will be lost, and a high physical change rate could cause TAP to bump against its transmission rate limiters, so retransmission is needed to ensure that the protocol makes progress. Any *Designated* or *Root Port* that is not *Forwarding* will set the *Proposal* flag and transmit once per second: the recipient of the *Proposal* will send an **Agreement** or a **Contract** as soon as one is available. *Forwarding Designated Ports* attached to point-to-point links transmit a message per tree every 30 seconds.

8. Scaling

TAP, as described in this note, uses per tree messages. Substituting LSP sequence numbers may be difficult, except in the case of a single change, which will not pose the risk of loops (on link loss) anyway. However rapid message transmission by any bridge only occurs when failure of a network link causes the tree to reconfigure locally, i.e. when the bridge or its neighbour is on the subtree whose *Root Port* is attached to the failed link. In a richly connected network a small proportion[3] of trees passing through any given node will be affected by any given failure,

[1]The delay is having TAP select Roles, rather than using IS-IS to calculate them, is rarely significant. The TAP exchange usually takes place before the result is needed to handle a network failure.

[2]The Designated Port needs to be Forwarding if an entity associated with the Alternate Port is only reachable from the attached LAN, e.g. if the Alternate Port's bridge also routes some traffic to and from the LANs or some management functionality is not reachable through the bridge's relay function.

while a network that has links whose failure affects most trees is unlikely to be large.

A PDU that carries TAP messages does not have to encode all the information required (see 2 above) separately: a single copy of the transmitting *Bridge Identifier* and *Port Identifier* is sufficient. Shortest path bridges also prefer to communicate within their own SPT *Region*, so a single copy of the *External Path Cost* component of the SST's *Root Path Cost* is sufficient. Only the Proposal flag, Promise flag, Port Role (2 bits), Port State (2 bits), Promise Number (2 bits), Discarded Promise Number (2 bits), the *Root Identifier's Bridge Address* (6 octets), and *Internal Root Path Cost* (4 octets) need be encoded per shortest path tree—a total of 12 octets. A single Ethernet frame can carry the TAP messages for over a hundred trees[1], and every message to be sent does not have to be encoded in the same frame. The number of TAP messages to be transmitted by each port, to recover after a single IS-IS update, is less than one per affected tree on average. Networks of a thousand nodes should be possible with (say) fewer than six frames per second per port to carry TAP messages.

9. Accelerated LSP distribution

LSPs are normally flooded by hop by hop. In a typical bridge implementation this means scheduling a control plane process that inspects and records receive LSP sequence numbers, and makes the decision to forward the LSP (or not) to other ports. The performance of link state network reconfiguration after failure is thus a poor second to protection switching, particularly when the repair has to be effected at some distance from the fault. However link state reconfiguration requires no pre-configuration and fewer spare resources. Good support for basic multicast is a distinguishing feature of bridge networks, and by transmitting the initial LSP to a multicast address that is bridged (in addition[2] to using an address that just reaches neighbours attached to the same individual LAN) fresh LSPs can be disseminated widely. If the SST is used by both nodes at the ends of a failed link to send LSPs, at least one should reach each node where a tree needs to be repaired. The paths provided by the SPT for each node (if supported) may be shorter, is at least as reachable, and should improve the delivery probability.

10. Root failover

This note began with an example where TAP, or something like it, is required to prevent loops. It is also important to avoid significant delay when there is no risk of a loop. The simplest cases occur in structured networks where a failed *Root Port* is replaced by an existing *Alternate*. Consider the network in Figure 3, what happens when 303's link to 101 fails?

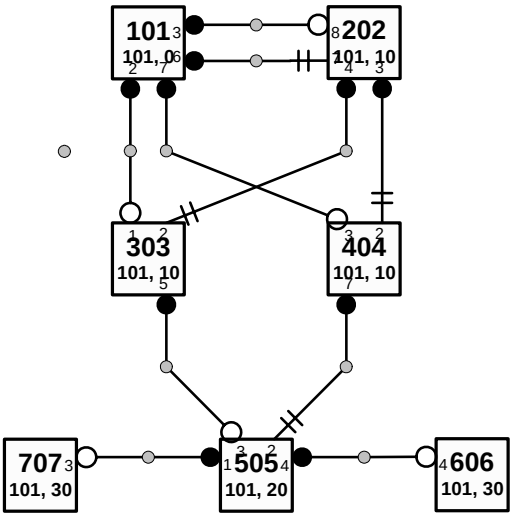

Figure 3—Structured network (fragment)

303 already has a **Contract** from 202, which will become acceptable as 303's distance from the Root increases, while both that **Contract** and 303's new *root priority vector* remain *better* than the **Agreement** it had previously received from 505, so 303 can send a new **Agreement** to 202 without having to make any cuts. 202 may want to make use of that fresh **Agreement** if its own distance from 101 increases later. See Figure 4.

Figure 4 assumes that 303 performs its link state calculation, selects its new Root Port, and sends the Agreement before 505 notices the original failover. However 505's failover to 404 is quite independent: again the new Root Port can be selected and an Agreement sent without blocking any ports (whether 606 and 707 have processed the failure yet or not). See Figure 5.

Figure 5 also shows transmission of fresh Agreements by 505, and a fresh Contract by 303.

It has to be said that the designer of this network lives on the edge: all the bridge's priorities play a role in the

[3]At a rough estimate 2 in p where p is the number of ports at each node, five hundred trees for a very large network of two thousand 8-port bridges, one hundred for network of a thousand twenty port bridges. More trees are affected at nodes with less connectivity if links directly attached to those nodes fail: half the trees at an edge node with dual links to the core, but connectivity for almost all those trees is recovered by selecting an Alternate Port as the new Root Port, without any protocol being required.

[1]116 if the TAP messages are encoded in an MSTP BPDU after the fixed header.

[2]It is probably best to send two copies of the initial LSP, one by direct multicast and one hop-by-hop (as normal) as the latter has less chance of being dropped as it progresses through the network (though failures tend to reduce network traffic).

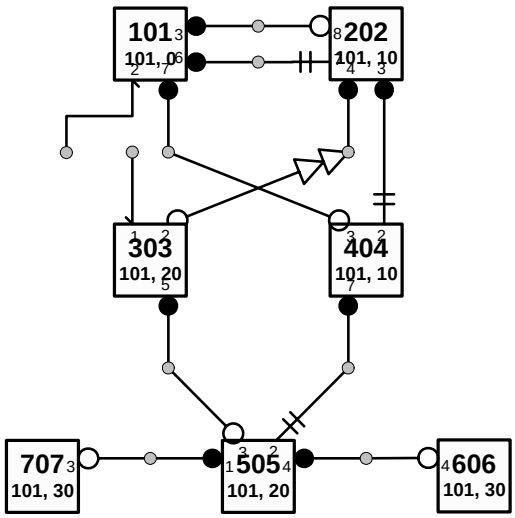

Figure 4—Failover in progress

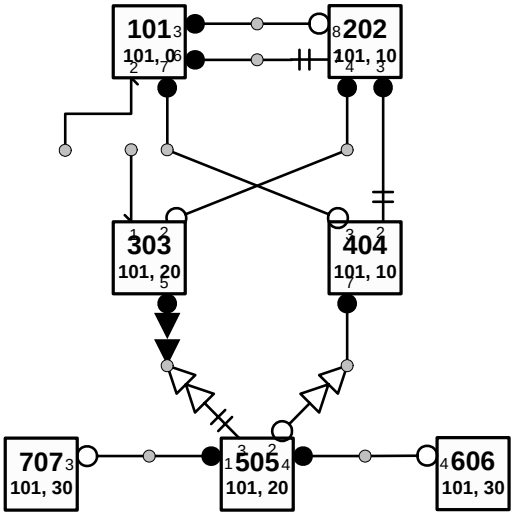

Figure 5—Failover complete

initial configuration and/or reconfiguration. A lower port path cost for the 101-202 links would give the same results without all that administration. The conclusion from this example, remains the same: TAP has no adverse impact in this simple case.

11. Ring protection

How does TAP and a link state protocol with accelerated LSP distribution) perform in networks with long chains[1,2] of bridges?. How close can fully mesh capability come to the performance of a point solution that only works in a single ring? Figure 6 shows a tree (shortest path, single spanning, or multiple spanning tree instance) rooted at 111 and blocked (to avoid loops). If the 222-333 link is lost, and 111 and 333 accelerate the transmission of their new LSP by sending it over their shortest path trees (or the SST if SPTs are not required), how will TAP repair the network?

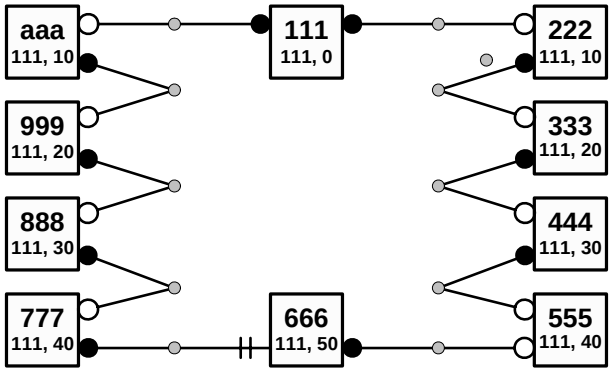

Figure 6—Single chain mesh

Assume (for simplicity) that all the bridges calculate the new topology at the same time (the time taken for one or other of LSPs carrying new of the failure to travel to each bridge being negligible unless the network links are extremely long). Figure 7 shows what follows: 666 selects a new *Root Port*, and sends an **Agreement** to 777 (if 666 was prescient it would know that this is unnecessary, but it proceeds strictly according to the algorithms already described in this note) which it can do since the *Root Port* is still *Discarding* (666 lacks a suitable **Agreement** from 555); 555 and 444 block their *Root Ports* in order to send **Agreements** and **Contracts** (with their new *root priority vectors*); 333 sends an **Agreement** but does not have to block its new *Root Port* as it has no other *Forwarding* ports.

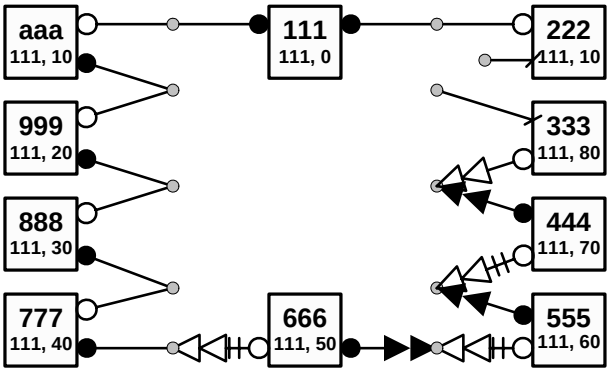

Figure 7—Repair (first step)

Once these messages have been received all the ports can be made *Forwarding* (Figure 8). Just one TAP message has been sent by each of the bridge's whose role in the topology has changed, and the repair is

[1]A bridge is in a chain if its only possible connection to a significant *Root* is through two of its ports, the other ports always providing edgebound connectivity

[2]An inordinate amount of attention has been paid to rapid restoration of service in 'single chain mesh networks' (networks whose physical topology is not loop free—and is hence a mesh—but whose nodes all belong to a single chain, a.k.a rings) despite the fact more complex topologies are usually deployed (rings of rings etc.) which necessitate a variety of complex props to what was going to be (but rarely is) a simple protection protocol.

complete! The delay to prevent loops would be no greater for a ring with two hundred nodes.

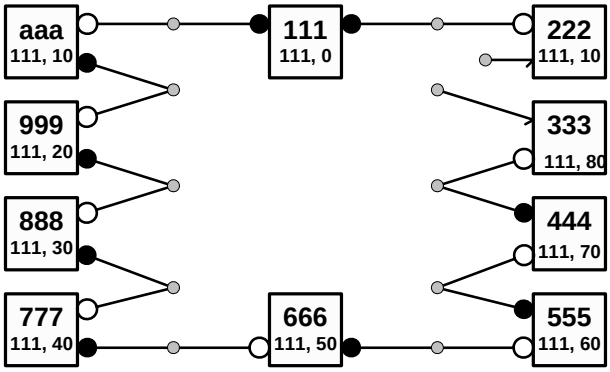

Figure 8—Repair complete (second step)

If the link failure had been between aaa and 999, the repair would have taken no longer (see Figure 9).

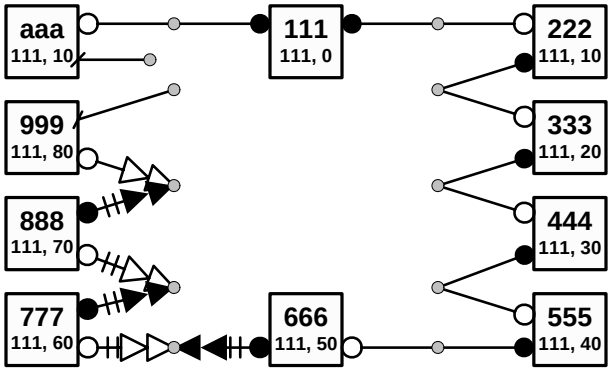

Figure 9—Another repair (first step)

The new **Contract** sent by 666 also asks 777 to discard 666's **outstanding Agreement**, but 666 doesn't have to wait for a response: 777 has already performed the discard, and includes that information in its own **Agreement**. After the messages in Figure 9 have been received all the ports can be made *Forwarding*.

Of course we cannot assume that all nodes complete link state updates at the same time. In Figure 10 bridges 999 and 777 complete first. When 888 and 666 complete they have already received new promises (and notification that their old **outstanding promises** have been discarded), so they can transition ports to *Forwarding* as they transmit their new promises (Figure 11) to complete the repair (Figure 12).

12. Using BPDUS

Most of the above information is already conveyed in *MST BPDUs*[1], in a way that is RSTP compatible. TAP messages should be carried in spanning tree BPDUs, as that ensures interoperability with existing bridges.

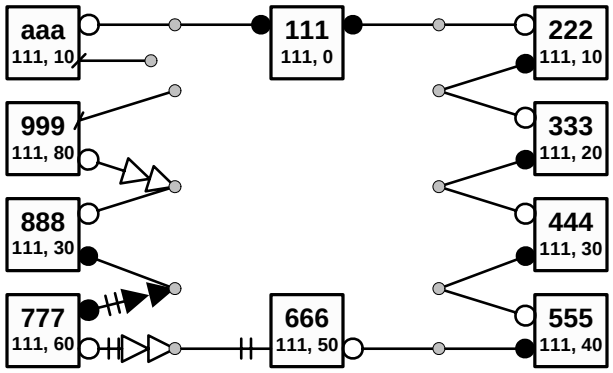

Figure 10—Some updates complete

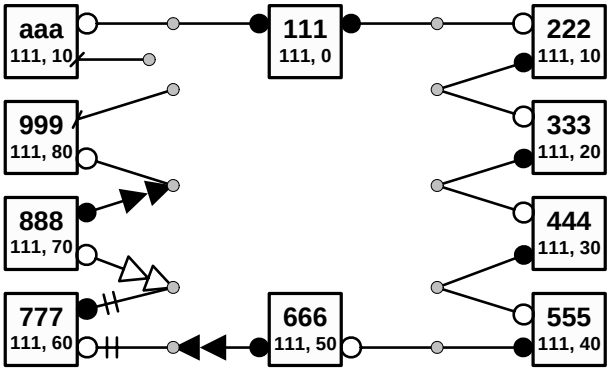

Figure 11—All updates complete

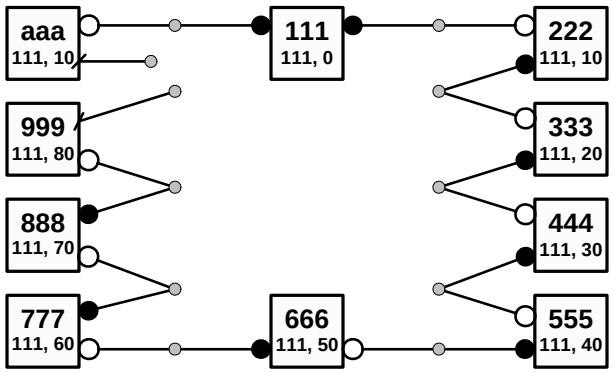

Figure 12—All messages received

The BPDU *Protocol Version Identifier* should be incremented so that TAP capable bridges can spot those that are not. The SST and MSTI **Promise Flags** are encoded in 802.1Q-2005's *Agreement Flag* fields (a **promise** is simply an *agreement* renamed so those edgebound and rootbound *agreements* can be easily distinguished). The SST and MSTI **Promise Numbers** and **Discarded Promise Numbers** are new (4 bits per tree), they are probably best encoded in a new TLV after existing fields. TAP message for SPT's should also be encoded in a new TLV (12 octets per message).

MSTIs can be configured either by IS-IS (whether multi-topology support is being used or not is invisible

[1]Note that an RSTP bridge cannot acquire new topology information that makes it closer to the root than its parent, as the latter processes the topology information as it is communicated, so RSTP lacks the bi-directional agreement (called Contract Agreement in this note for clarity) provided by MSTP.

to TAP) or by existing MSTP. The existing mechanisms (developed for P802.1ah) to partition the responsibility for VIDs should be used to determine which protocol is used for each MSTI.

13. Using IS-IS

In addition to information already mentioned in this note, it is desirable that IS-IS LSPs be capable of carrying the External Root Path Cost for the SST and selected MSTIs from any *Root Port* or *Alternate Port* at the edge of the link state Region (dynamically determined by BPDUs for interoperability with existing bridges). This may not be important in the very long term but would facilitate deployment of link state islands within existing bridged networks.

14. Apology

Bridging and routing each have their own terminology, differing even when much the same problem is being discussed. Different propositions are ‘obvious’ to each community of designers, while others are regarded as novel even though well known when consider using the other terminology and techniques. This note was originally intended to be accessible to those approaching the loop free link state shortest path bridging problem from the IS-IS and routing direction. However the lack of precision involved in mixing terminology (even when routing terminology seemed easier for a specific purpose) led me to abandon the attempt for the time being.