# Faster flushing with fewer addresses

Vipin Jain
Mick Seaman

This note describes improvements to the topology change detection, notification, and filtering database flushing mechanisms of IEEE Std. 802.1D. These enhance the timeliness of filtering information changes while reducing their number and network scope. While retaining backward compatibility and the simplicity of the existing reconfiguration mechanisms, the proposed improvements provide quicker restoration of service between communicating stations and reduced flooding of traffic while station location information is relearnt. These proposals thus support both Availability and Scalability goals.

In general these improvements may be thought of as orthogonal to "High Availability Spanning Tree", but they can make particularly effective use of the analysis and characteristics of that prior proposal.

## Introduction

In a bridged local area network, MAC Bridges note and communicate active topology[1] changes resulting from Spanning Tree Protocol operation, flushing the learnt end station location information held in all their filtering databases. Thus stations whose addresses appear to move as a result of a reconfiguration do not remain unreachable[2].

Specifically, a topology change is detected whenever a bridge port transitions to the Forwarding state[3], or from the Forwarding or Learning states to the Blocking state[4].

The Root Bridge is notified, and subsequently informs all the bridges in the network that their databases should be flushed. The mechanisms supporting this communication are built upon the regular spanning tree configuration message propagation, ensuring that all bridges will always find out about a topology change. If for some reason the configuration message propagation itself fails a further topology change is guaranteed. This in turn will be communicated to all bridges: unless of course a further topology change occurs.

Thus the protocol requirement is not that any given change be communicated reliably, but that after the last of any set of changes the knowledge that some change has occurred reaches all bridges. This bounds the protocol burden by merging simultaneous changes[5].

Topology change communication takes place in two stages. First the root is notified by periodic transmission of Topology Change Notification BPDUs toward the current Root, until acknowledged by a Topology Change Acknowledgment flag set in a Configuration BPDU. Next the Root sets a Topology Change flag in the Configuration BPDUs that it originates for an extended period. The setting of this flag is propagated by the Designated Bridges for each LAN. This procedure is made robust against a complete partitioning of the network, possibly resulting in a Root in one partition holding a topology change notification originally detected in the other, by requiring that any bridge considers becoming the root as a topology change.

Formally the Topology Change flag does not mandate instant flushing of databases, but causes each bridge to age its filtering database entries rapidly[6]. The fast ageing time is the same as that required to transition a newly Designated Port from the Blocking state to the Learning state. If the topology change was originally detected through a local loss of connectivity, i.e. a transition from Forwarding to the Blocking, the flushing of filtering database entries will be completing just as the complementary gain in connectivity begins to take effect, i.e. a newly designated Bridge port starts learning[7].

[1] The spanning tree algorithm selects a loop free, fully connected, active topology from the available physical topology which may be redundantly connected.

[2] In today's networks it is unclear to what extent service is restored by (a) flushing obsolete learnt information or (b) relearning from new traffic. For example: in IP networks the communication path through switches between an end station and its designated router may be restored by ARP exchanges.

[3] Elsewhere described as a gain in connectivity.

[4] Elsewhere described as a loss of connectivity. This is from the point of view of the bridge detecting the change. Some other bridge will experience a corresponding gain in connectivity.

[5] While information is lost, the bounding of both local bridge state and transmission resources is a highly desirable result.

[6] To realize all the benefits of the "High Availability Spanning Tree" proposal, with instant failover, we need to flush or move addresses more rapidly than this.

[7] A simplification since it does not account for the delays in notifying the Root of the topology change and in the Root communicating to other bridges.

## Proposed Changes

This note proposes ways to accelerate flushing of learnt addresses and to flush fewer addresses. It begins by studying the topology changes that can occur in a bridged local area network, showing that all reconfigurations can be described in terms of a few simple types. Only one of these, detaching a branch of the spanning tree and reattaching it elsewhere is fundamentally interesting. The address relearning requirement reduces to updating address information[8] that has been learnt along the path between old and new points of attachment of the branch to the network[9].

While a number of ways might be designed to accurately scope address relearning to only that path and the precise set of addresses involved, this note focuses on simple compatible enhancements to the existing spanning tree protocol. Although these involve more addresses and more of the network than strictly necessary, they maintain the original bounds on protocol and processing requirements, and are compatible with existing bridges in a network.

Flushing address entries is equivalent to adding them to all the bridge ports. For a bridge capable of enhanced filtering services,[10] addresses only need to be added to ports that may lead to the new point of attachment of a moved branch. This is how the minimum filtering database changes following topology change detection, notification and communication[11] are described. A bridge that cannot make such fine grained changes can add addresses to more ports or truly flush those addresses Relearning will eventually repair the excess flooding.

A new and detailed set of rules for making database changes following port state changes, receipt of TCN BPDUs, or receipt of Configuration BPDUs with the Topology Change flag set are proposed. The existing standard only flushes the filtering database on the last of these events.

The new rules:

(a) add the addresses learnt on a port that becomes Blocking or Disabled to the Root Port[12] on that bridge

(b) add the addresses learnt for all other ports, including the Root Port, to the Designated Port that receives a TCN BPDU, and

(c) add the addresses learnt for all Designated Ports to the Root Port on receipt of a Configuration BPDU with the Topology Change flag set or a TCN on the Root Port[13].

In addition, local topology change detection, or receipt of a TCN BPDU on a Designated Port or a Root Port[14] causes the receiving bridge to set a local copy of the topology change flag for each other port.

To accelerate relearning of station location, true flushing or bulk addition of addresses to ports rather than accelerated ageing is specified[15]. This increases the effectiveness of the relearning that takes place, since relearnt addresses are aged rapidly.

To prevent addresses from being repeatedly flushed by a repeating sequence of Configuration BPDUs generated with the topology change flag set, this flag is no longer set for the sum of Max Age and Forward Delay[16] but for a small number of Configuration BPDU transmissions per port[17]. While not offering quite as strong a guarantee as at present, i.e. notification of a topology change or a further topology change, this is consistent with the reliability strategies used in other protocols[18].

The topology change flag is set but not reset by the reception of Configuration BPDUs[19] on the root port, being reset only on expiry of the transmission count. The effect of these changes is to start communicating the topology change down branches of the tree passed by the notification on its way to the root. Perhaps even more important than the acceleration of the communication of topology changes is that this change removes the need for a bridge that becomes the root to start communicating a topology change even if none of its ports has changed state.

Figure 1 summarizes how topology changes are communicated in the existing standard. Figure 2 shows how this proposal changes that.

---

[8] For stations in the branch that is detached and reattached.

[9] An interesting special case is where the old and new attachment points are on the same bridge, and there is no need to notify or flush filtering database entries in other bridges at all.

[10] Where an address can be present on more than one port at a time, and if so present flooding to that address is limited to those ports.

[11] In this note the term “detection” is used specifically to mean the process local to a bridge which notes a change of port state, the term “notification” refers to signaling towards the current root, and the term “communication” to signaling from the root to other bridges.

[12] Could also add to Designated Ports, best choice is scenario dependent.

[13] Copes with a shared media case.

[14] Shared media case again.

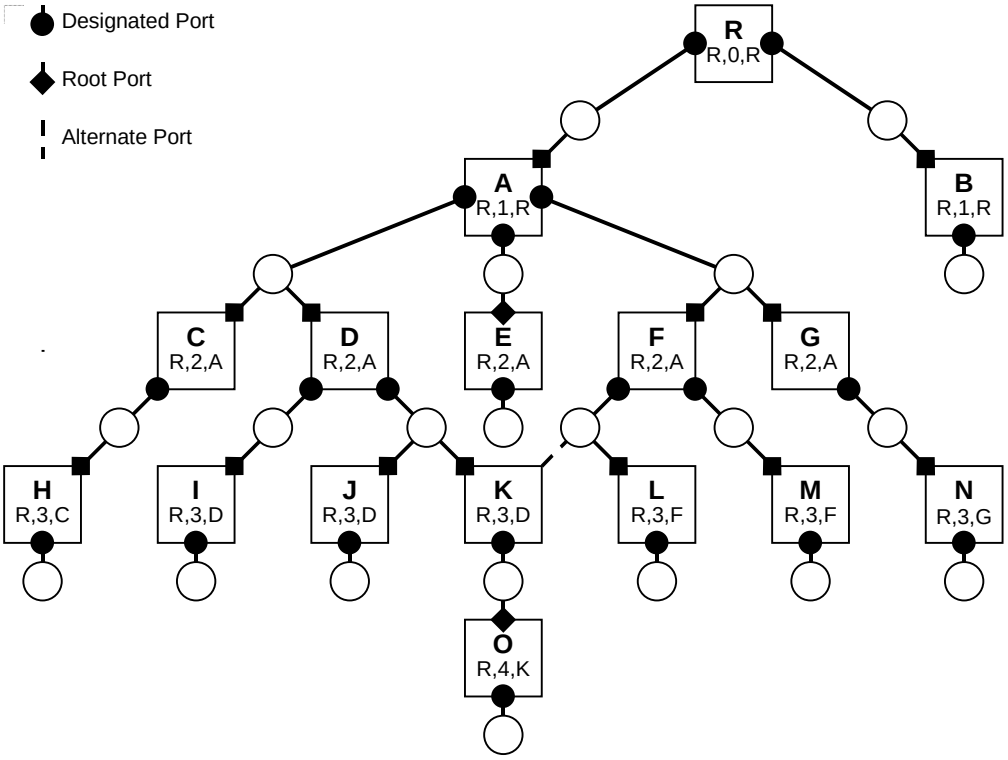
[15] Bridges that have long relearning delays after receiving frames carrying as yet unlearnt source addresses should attempt to balance the true flushing rate to match their relearning capabilities.

[16] 35 seconds by default.

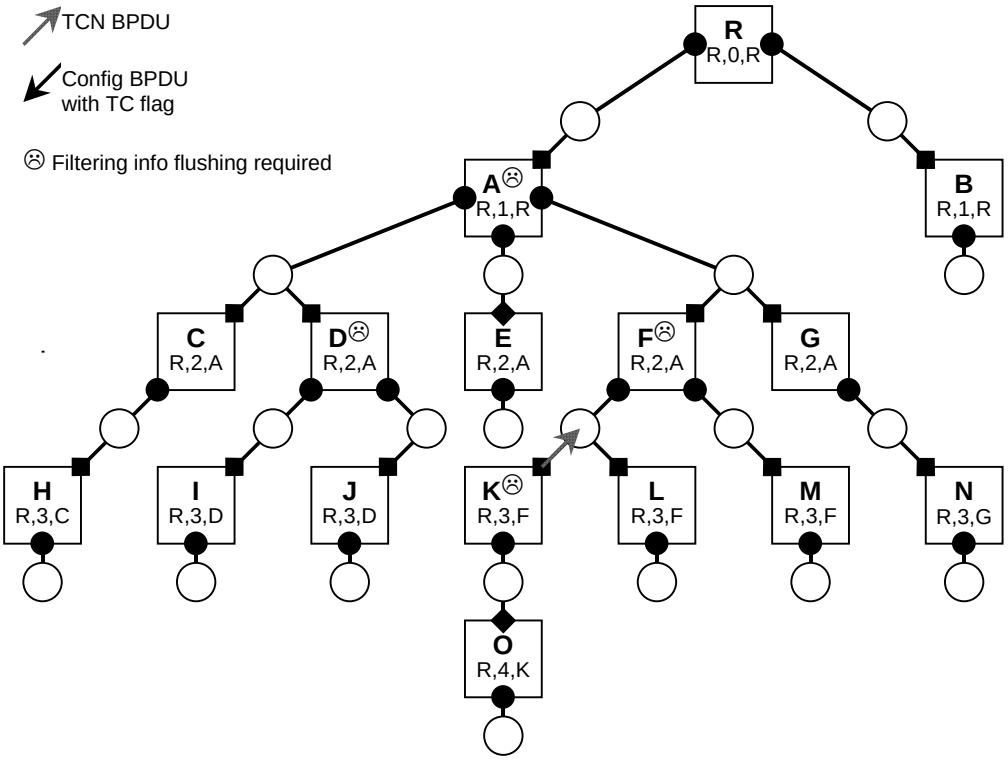
[17] Probably 3.

[18] Since the main cause of loss in LANs is receiver overrun, it makes sound sense to take design decisions like these that reduce the amount of extraneous traffic and hence diminish the probability of receiver overrun. Just as for other protocols extraordinary losses will be recovered in time.

[19] Of course it is set only if the TC flag is set in these BPDUs.

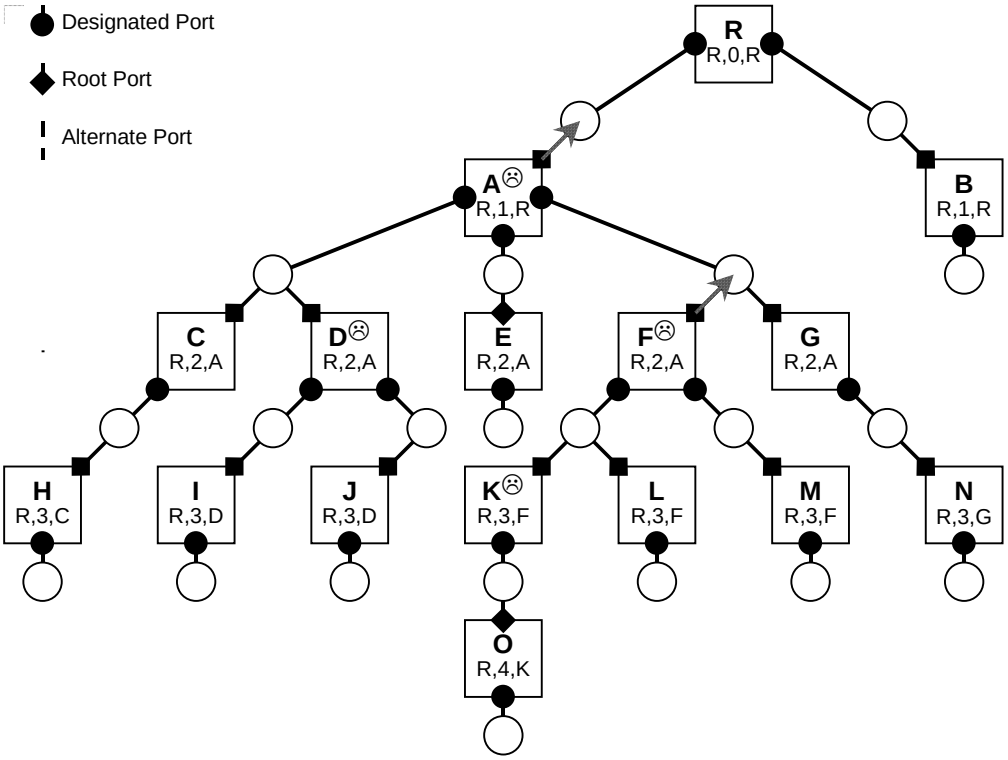



An example network before reconfiguration. The root, root path cost, and root port designated bridge is shown for each bridge.

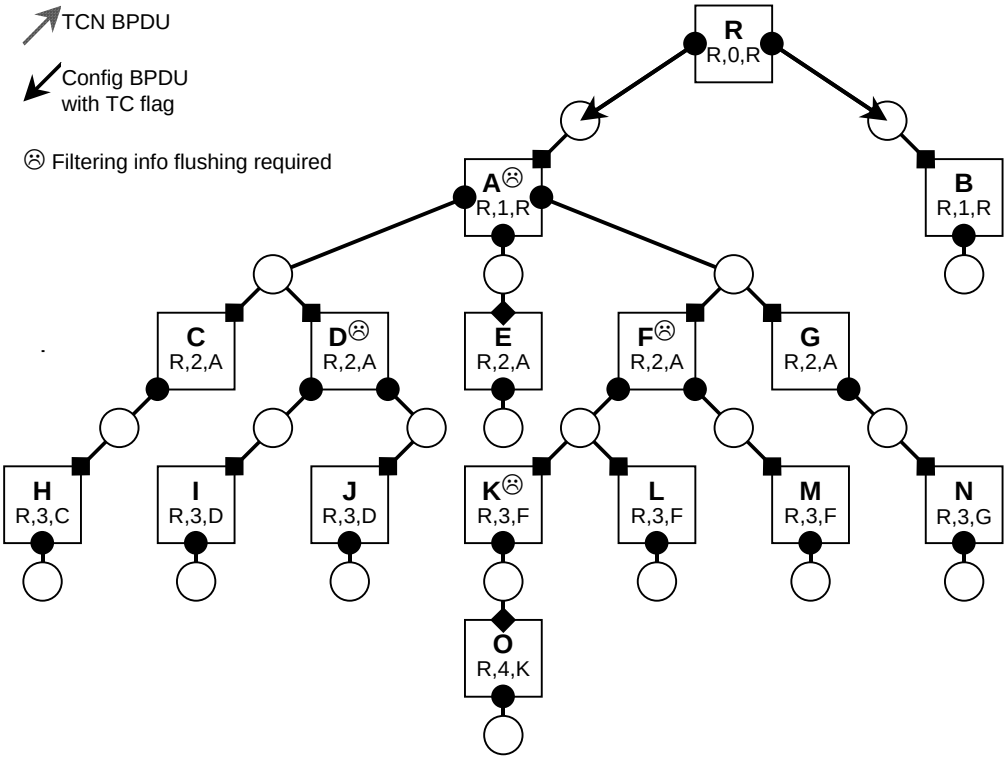



K's Root Port fails, so an alternate port is selected as the new root port, and a TCN is sent. For High Availability Spanning Tree this is done immediately, for the standard spanning tree this is done when the new root port becomes forwarding.

**Figure 1 – A Topology Change Example with the standard algorithm (I)**

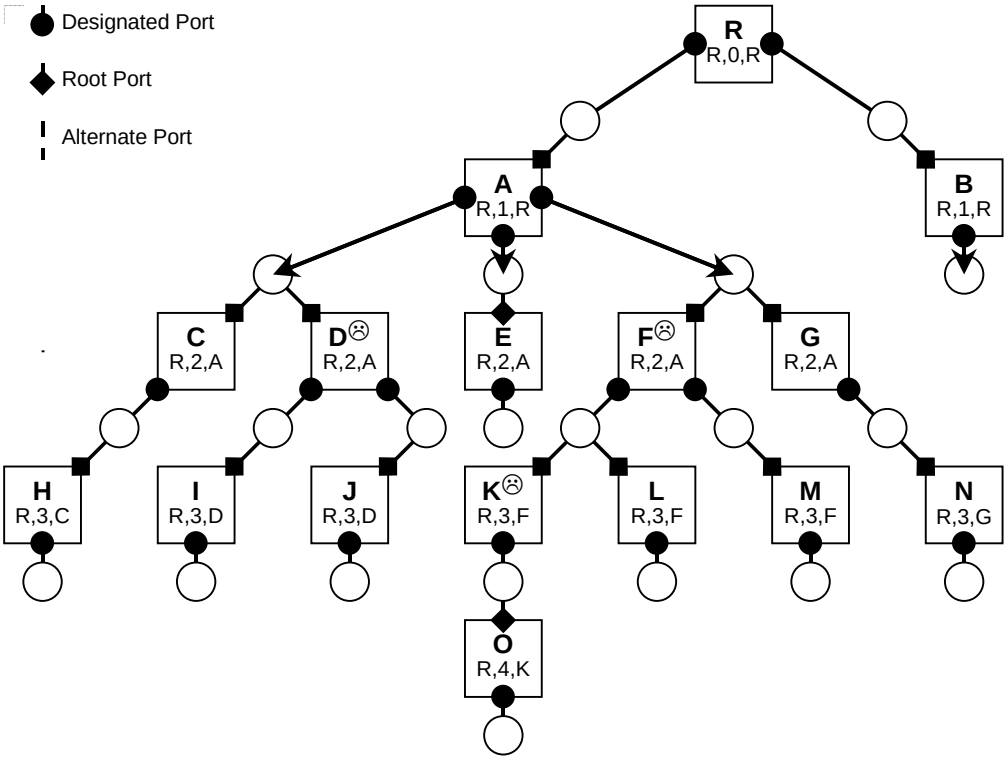



F and then A forward the TCN to R, they will retry until they get a Config BPDU with a TCack.

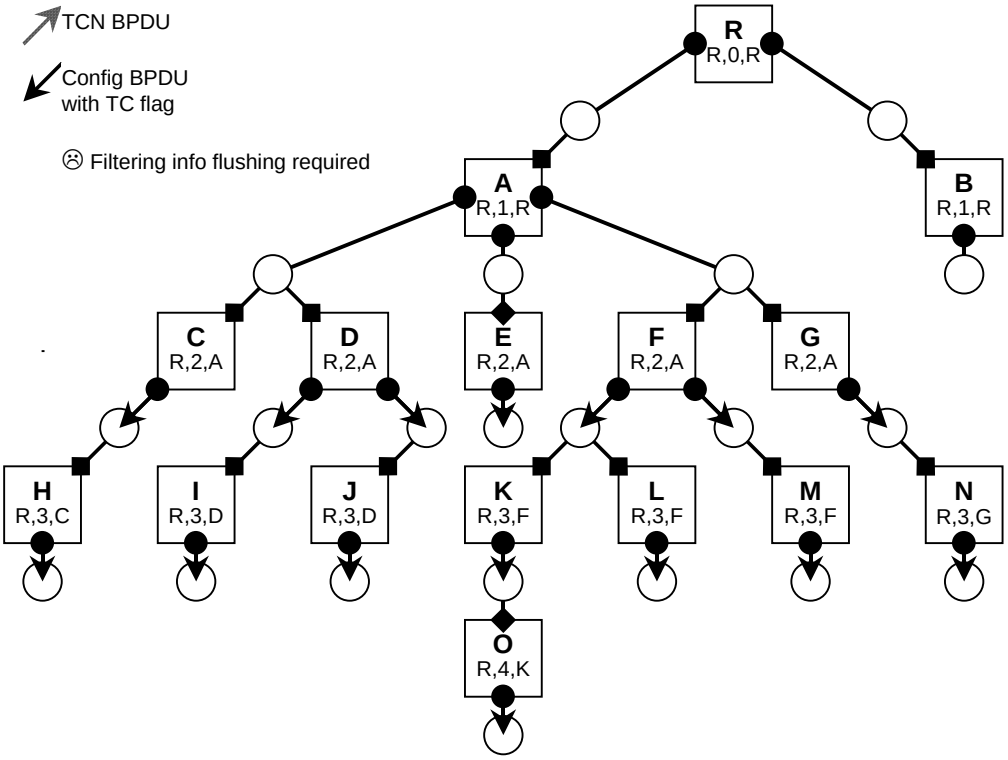



R starts flushing its entire database, and sends Config BPDUs on its Designated Ports.

**Figure 1 – A Topology Change Example with the standard algorithm (ii)**

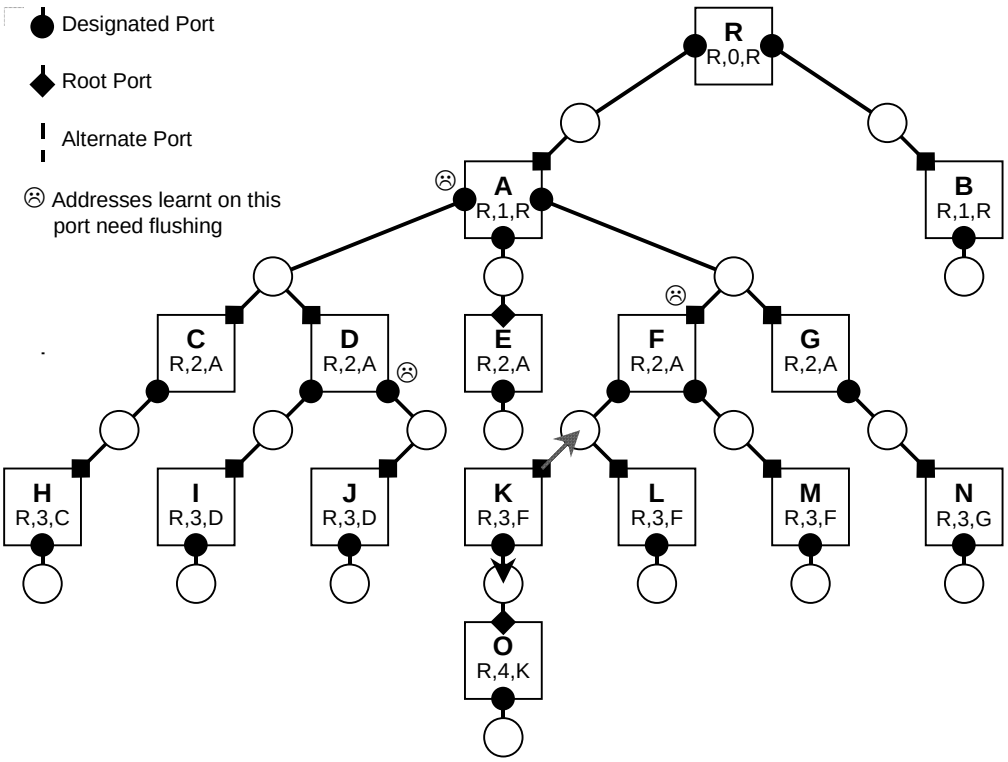



A and B start flushing their databases and forward the Config BPDUs with TC thru their own designated ports.

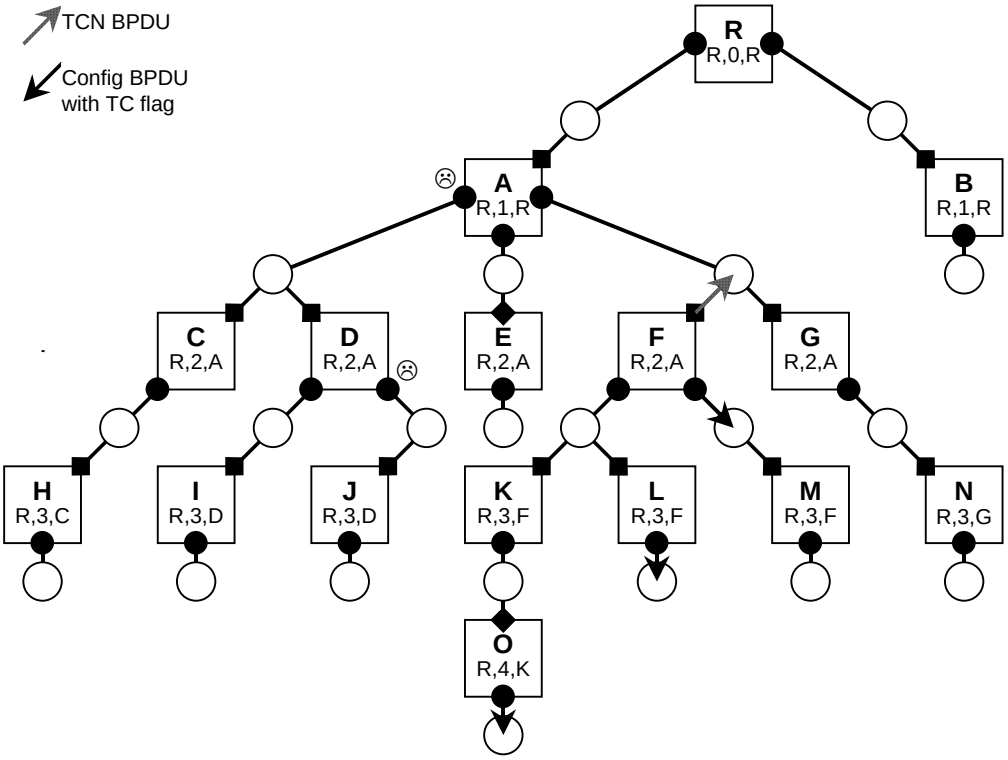



C, D, E, F and G do likewise, and as they too receive the TC, so do H thru N, and finally O. This diagram shows all those stages together.

**Figure 1 – A Topology Change Example with the standard algorithm (iii)**

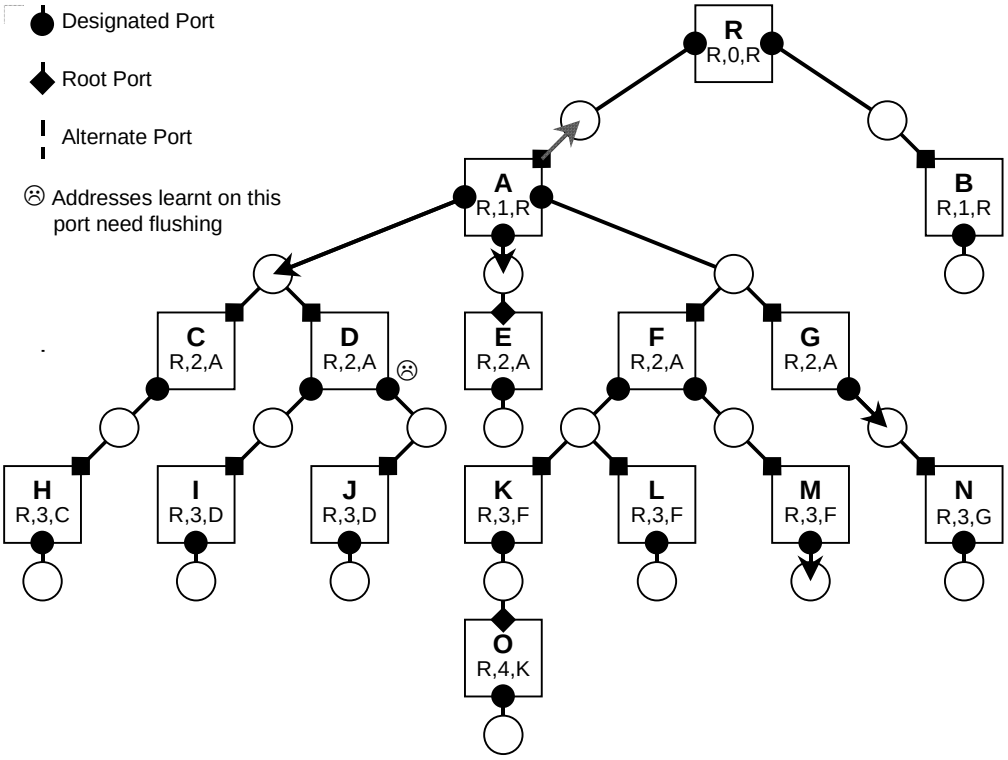



Same initial topology and change as in figure 1. K selects a new root port, moves addresses to that port, and sends a TCN. K also sends a Config BPDU with the TC flag set on its designated port(s).

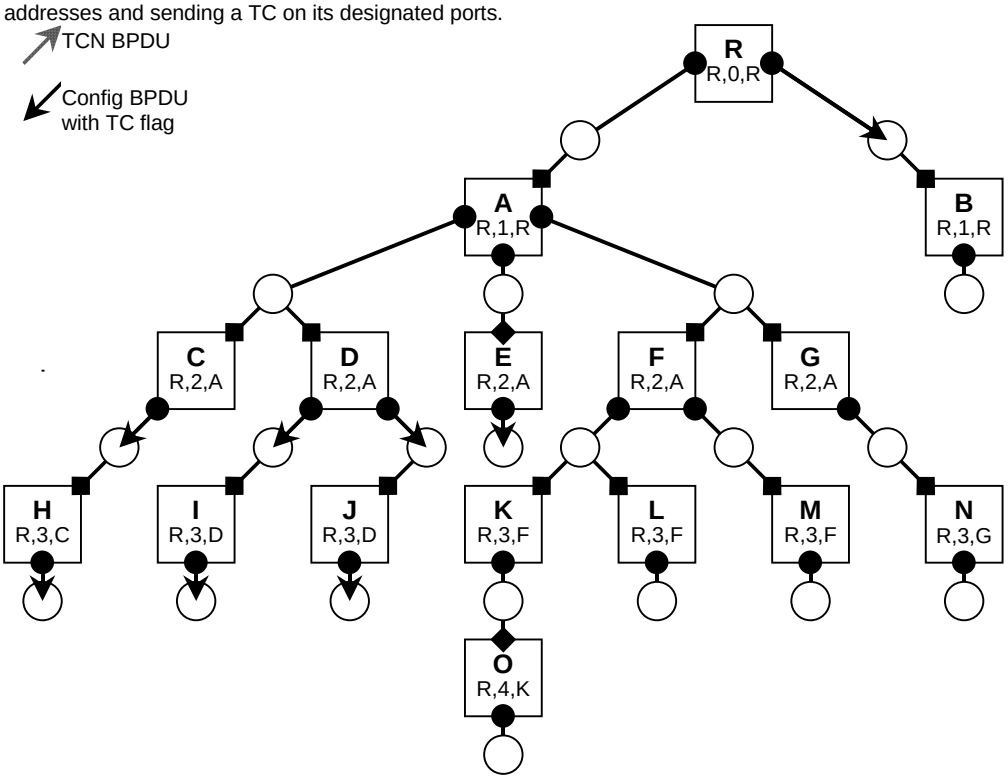



L receives the TCN and sends a Config BPDU with the TC flag set on its designated port(s) and flushes addresses learnt on those ports. F receives the TCN and forwards it toward R flushing entries learnt on its root port. F also sends a Config BPDU with TC on its other designated port(s) flushing entries learnt on those ports.

**Figure 2 – A Topology Change Example with the proposed enhancement (i)**




G receives the TCN, sending a TC on its Designated Port and flushing addresses learnt on that port. M receives the TC, flushes addresses learnt on its designated port(s) and forwards the TC. A receives the TCN and forwards it toward R while flushing addresses and sending a TC on its designated ports.




R receives the TCN and sends a TC on its designated port, flushing addresses on that port. C and D receive the TC from A, flush addresses on their designated ports and forward the TC through those ports. Eventually the TCs reach all bridges and LANs.

**Figure 2 – A Topology Change Example with the proposed enhancement (ii)**

## Summary of Benefits

The scope, amount, and duration of flooding of frames sent to unknown destination addresses following topology changes is reduced.

While the backbone or core switch in the network is still exposed to traffic flooded from all the branches in the network, only the path leading from the core to a newly attached portion of the network is exposed to traffic flooded from all those branches. Further, addresses relearnt immediately after the change are effective for longer, reducing the duration of excess traffic.

Flushing of learnt, but now incorrect, information happens more quickly, minimizing disruption of service. This is particularly true if the reconfiguration is confined to a branch of the tree. Flushing is initiated before knowledge of a topology change reaches the root.

The number of topology changes detected may be reduced. In particular, elimination of the requirement for a bridge to detect a topology change just because it has become a root can be used to eliminate network wide flushing of addresses resulting from power cycling of redundant or edge switches.

## Analyzing Topology Changes

The spanning tree algorithm and protocol selects an active topology from the available physical resources by making some bridge ports blocking and others forwarding. Any change in this active topology from initial to final stable states can be thought of as a combination of one or more of the following operations:

(a) a root move, where the connectivity is unchanged but the bridge identified as the root changes

(b) a rotation, where the port roles (Root Port, Designated Port) on an individual bridge change although there is no true change in topological connectivity

(c) a partition, where the result of the change is two separate networks

(d) a merge, where two previously separate networks become one

(e) a branch move, where part of the network partitions (is detached) from the rest and then merges (is reattached) with the rest of the network again through a different physical link.

When a partition, merge, or branch move occurs, a root move and a possible accompanying rotation may happen from the viewpoint of only some of the bridges in the network. A topology change will only result in a partition if there are no physical links connecting the two partitions. In one of those there will be an accompanying root move. Similarly a merge will always result in a root move in one of the prior partitions.

A branch move could be represented as a partition followed by a merge and a possible rotation. However our focus in this note is on the necessary changes in learnt filtering database information so this decomposition is not helpful.

Examining these operations we see that:

(a) a root move does not by itself require relearning, and will only play a part in relearning if the root has a special role in the relearning triggering mechanism - as is the case with the current standard. The proposed enhancements remove the need for that special role.

(b) similarly a rotation does not require relearning, and would only do so if the relearning triggering algorithm was based on port role rather than connectivity – this is not the case for either the current standard or the proposed enhancements.

(c) following a permanent partition some addresses that were previously reachable are so no longer, however no amount of relearning will help! So, for the purpose of our analysis, this type of topology change is not interesting.

(d) a merge of previously separate networks results in more addresses becoming reachable. Since the addresses of one part will not have been previously known to the other there will be new learning to be done, but not relearning required.

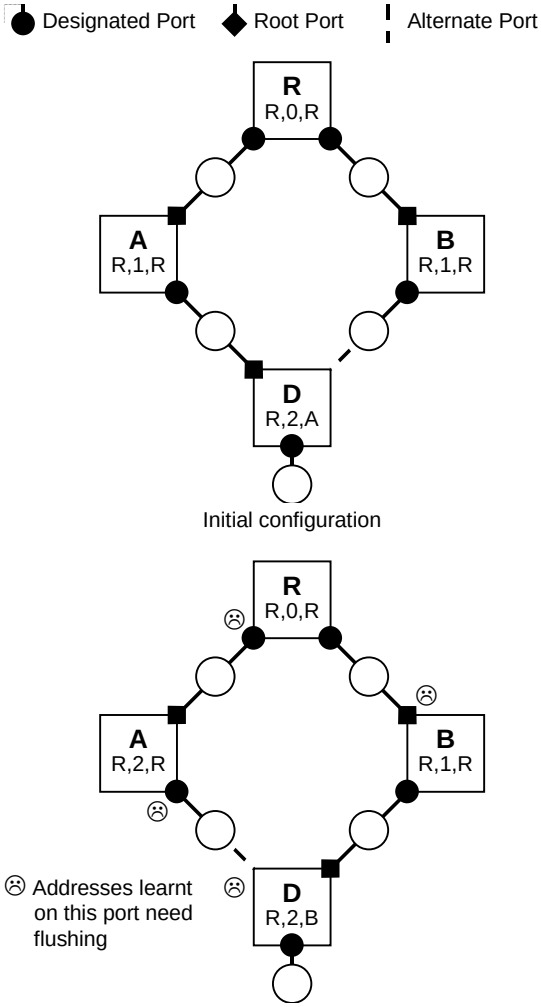
(e) only in the last case of a temporary partition and reattachment is there relearning required. All such changes can be described as branch moves simply by looking at them in terms of the move of the part of the network that does not contain the initial (and final) root bridge.

However individual bridges, only able to directly observe connectivity changes on their own ports, may not be able to detect the difference between cases (e) and (d), while (a) and (b) may involve temporary changes in connectivity also, so may be falsely identified as “interesting” topology changes. The important thing is that no long-term denial of service results from this lack of clarity, though there may be a little excess flooding.

## Detecting Changes in Connectivity

A topology change is 'detected' by a port if it has transitioned to the Forwarding state[20], or was previously Forwarding or Learning and has become Blocking[21].

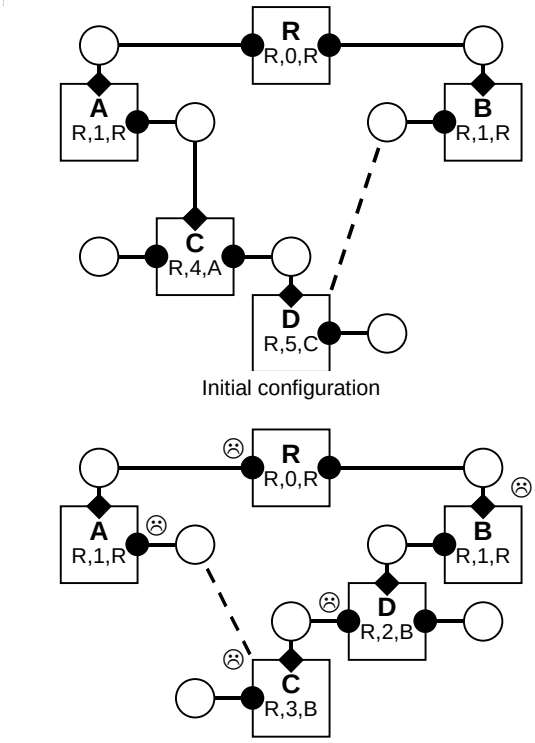
In many cases in a structured network, the same bridge that transitions a port to Forwarding will also have transitioned one to Blocking, as in Figure 3.




Parameter changes cause D to select D-B as Root Port, transitioning it to Forwarding , and Blocking D-A.

**Figure 3 – Connectivity Change**

That is not always the case, as can be seen in the following example (Figure 4).




Parameter changes cause D to select D-B as Root Port, transitioning it to Forwarding , and C to select C-D, Blocking C-A.

**Figure 4 – Connectivity changes at two bridges**

Here C has "lost connectivity" whereas D has gained[22]. Note that the bridge(s) between D, where the connectivity gain has taken place, and the root R have to flush addresses previously learnt on their root port(s). Bridges between C, where the connectivity loss took place, up to and including D have to flush entries learnt on their designated ports[23]. The latter also applies to bridges down each branch of the tree passed between D and R, and other branches of the tree attached to R.

The proposed enhancements do not rely on notifying losses in connectivity, but rather the bridge that detects a connectivity gain (D in the example) sends Configuration BPDUs with the TC flag set through its designated ports as well as sending a TCN toward the root. Those Configuration BPDUs are used to flush addresses learnt on designated ports.

[20] In this case there is also a requirement that at least one port be Designated. This rider is really nit-picking and is rarely effective, in any revision to 802.1D we should remove it since it necessitates a scan or record of the states of all ports in a procedure that could otherwise focus on a single port. In a bridge with many ports this is a completely unnecessary complication.

[21] A topology change is not detected if a port is put into the Disabled state, from the point of view of the mechanisms described in the standard that is O.K. since for that change to be "interesting" some other port should (eventually) transition to Forwarding.

[22] While there may also be temporary transitions on the C-D ports these are not guaranteed and it would be dangerous to rely on them.

[23] This makes the scenario symmetric for the moved branch and the rest of the network (except of course the latter is at the root of the tree). This symmetry is intuitively required.

The original spanning tree algorithm's topology change detection relied entirely on detecting and notifying connectivity gains, on the grounds that any loss in connectivity in a network that had not partitioned would necessarily be accompanied by a gain. Connectivity losses were also included when it was pointed out that healing of a LAN by an auto-partioning or manageable repeater could cause a bridge port to block without there being a corresponding forwarding transition[24]. This was an astute observation at the time[25], but is no longer of significant relevance to the way that bridged networks are constructed and it is proposed that only connectivity gains, i.e. transitions to forwarding give rise to topology change notifications. As a local optimization to speed the rate at which useless forwarding database entries are discarded, bridge ports that transition to either the Blocking or the Disabled states from Learning or Forwarding should have their associated addresses flushed.

## Receiving Change Indications

When both topology change notifications and consequent configuration messages with the TC flag set signal a connectivity gain it becomes clear both types of message have essentially the same meaning. They both indicate an addition of end stations to part of the network whence they came. There is no way of knowing where those end stations were originally attached.

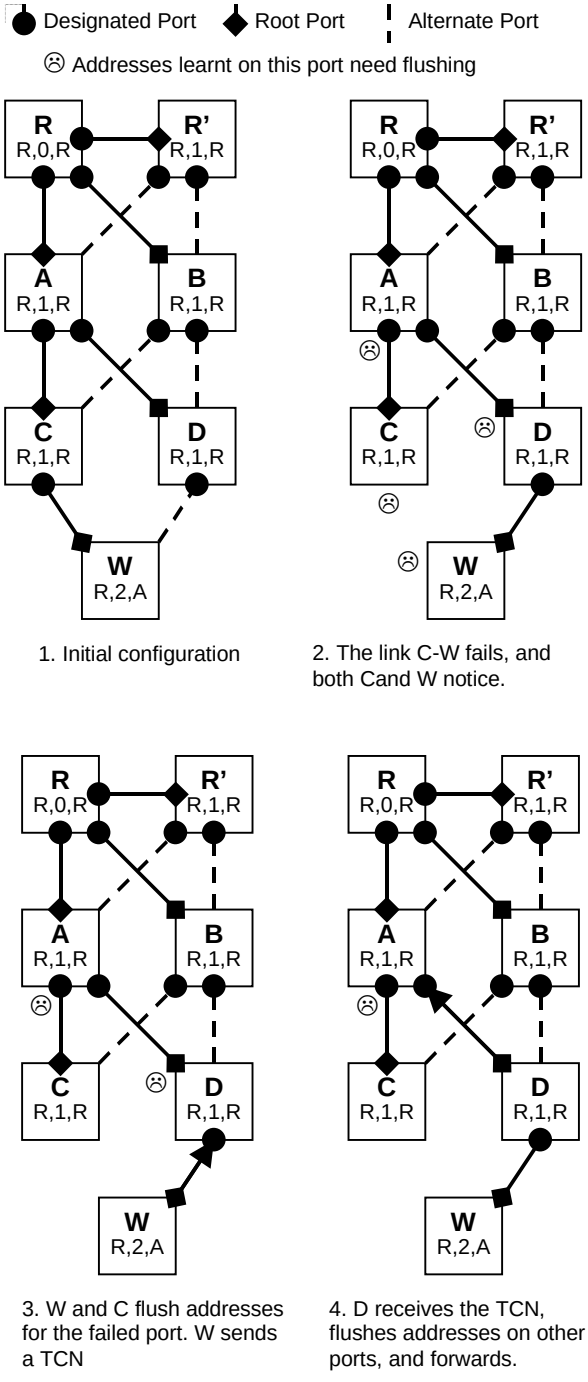
The correct response is therefore to flush all the addresses on ports other than the reception port[26].

As described more fully above, this:

(a) allows addresses to be flushed sooner, since flushing can occur when a TCN is received, rather than waiting for the notification to get to the root and back.

(b) flushes fewer addresses, since not all addresses are flushed on all bridges.

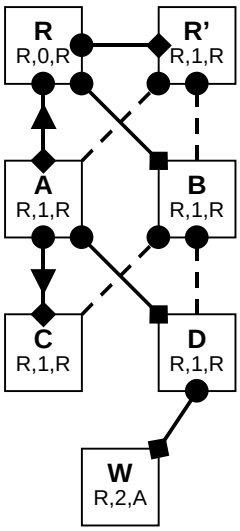
Figure 5 illustrates the application of these new rules in a real scenario.



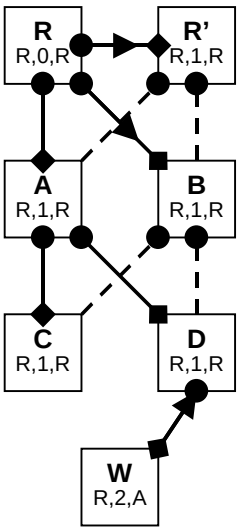

[24] Since the other bridge port attached to the second half of the now healed partition would have been forwarding already.

[25] Ann Ambler, Spider Systems, circa 1987 (?).

[26] Or, if the bridge can, to add the addresses for those other ports to the reception port.




5. A receives and forwards TCN, flushes, and sends a Config BPDU with TC to C.




6. R receives the TCN and sends Config BPDUs with TC on its other ports.

**Figure 5 – Change Example**

In this particular example, all the mislearnt addresses are flushed after the TCN generated by W proceeds two hops, to D and then to A. Moreover, if there are no prior topology changes in progress, and neither of the TCN BPDUs are lost, that will happen as fast as D and A can handle the TCN, without any protocol imposed delays.

## An Alternative Description

The suggested improvement could be described by beginning with the idea that a bridge that transitions a port to forwarding simply transmit a special multicast packet on non-blocked ports. The purpose of this packet would be to let all the other bridges know of the topology change. Such a packet would naturally follow the paths taken by the combination of the TCN BPDUs and the Config BPDUs with TC flag set.

From this alternative beginning, the present proposal could be understood as adding reliability, rate limiting, and backwards compatibility without having to define new messages or send additional packets. Marking the message toward the root (the TCN) as being distinct from the message away from the root (the Config BPDU with TC flag set), also diminishes the likelihood of the change indication looping, while allowing flushing to proceed even if ports to be included in the final active topology are currently blocked[27].

Though it might be thought that a special multicast packet would travel faster than a TCN BPDU, the need for each bridge to take action on it means that it would not actually do so in most implementations. Moreover it is hard to meet both reliability and overall rate limiting goals without mandating hop by hop processing. So, it appears that the present proposal for modifying the actions taken on TCNs and Config BPDUs with the TC flag set is the better choice.

## Backwards Compatibility

If the Root Bridge sets the TC flag on Config BPDU transmissions on every port, including on a port that has received a TCN BPDU[28], backwards compatibility is assured without additional arrangements[29]. Bridges implementing the improvement will flush addresses learnt on all ports, other than the reception port, when a TCN BPDU is received, and will flush addresses on all ports other than the root port when a Config BPDU with TC flag set is received. Existing standard bridges not implementing the proposal would simply delay flushing addresses until the Config BPDU was received, and then flush all addresses.

This approach results in more flushing than is proposed above, where the Root Bridge sets the TC flag only on ports other than that receiving the TCN. For this more selective solution to work in a network with a mix of old and new bridges, some indication of the presence of those old bridges is required. This can be accomplished by defining a 'not-break bit' in the TCN. This is set by a new bridge detecting a topology change. While new bridges can forward the TCN with the existing set of the 'not-break bit', old bridges will naturally generate or forward TCNs without this bit. In this way a new Root Bridge can determine whether the TC flag needs to be set for a while on the port receiving the TCN.

A small exception to this rosy picture of backwards compatibility lies in the problem of complete network partitions where topology change notifications may be lost into the other partition. For a network of all new bridges this is no longer a problem, but for old bridges it may be. Given:

(a) the low likelihood of complete partitions in network designed to be highly available

(b) the fact that bridge ageing timers are very rarely extended to accommodate bridged WAN links these days,

(c) the continued absence of 'speak only when spoken to, or not at all' end stations from our networks.

I suggest we punt on this problem, and allow bridges implementing the proposed improvements to not assume a topology change simply because they become root bridge – on power up or at any other time.

[27] A plain multicast message would be discarded at a blocking port, though the topology change indication process would commence again once that port transitioned to forwarding.

[28] In the alternative described further below, if a TCN BPDU is received on more than one port within the timescale for which TC flags are set, then the TC flag will be set on all ports anyway.

[29] With one small exception, noted below.

## Conclusion

A backwards compatible improvement to the topology change detection, notification, and filtering database flushing mechanisms of IEEE Std. 802.1D has been described. This improvement retains the simplicity of the existing mechanisms but enhances the timeliness of filtering information changes while also reducing their number and network scope.

It is suggested that the proposed improvement form part of the proposed new work on rapid reconfiguration and service restoration in bridged local area networks.