# DRSTP: A Simple Technique for Preventing Count-to-Infinity in RSTP Controlled Switched Ethernet Networks

**Syed Muhammad Atif** syed.muhammad.atif@gmail.com
*M.S Computer Networks*
*Department of Computer System Engineering*
*Usman Institute of Technology*
*Karachi, Pakistan.*

**Abstract**

Ethernet is a dominant local area network (LAN) technology from last three decades. Today most LANs are switched Ethernet networks. Spanning tree protocol is a vital protocol for smooth operation of switched Ethernet networks. However the current standard of spanning tree protocol for Ethernet – commonly known as Rapid Spanning Tree Protocol or in short RSTP – is highly susceptible to classical count-to-infinity problem. This problem adversely effects the network convergence time, depending upon how long count-to-infinity situation persists in the network, and thus leads to network congestion and packet loss. In the worst case, even forwarding loops may be induced that further enhances the network congestion. Thus, the dependability of RSTP controlled Ethernet networks are highly questionable due to its vulnerability against the count-to-infinity problem. This paper first discusses the count-to-infinity problem in spanning tree controlled Ethernet networks, in general and in RSTP controlled Ethernet networks, in particular. Then this paper proposes a simple solution to overwhelm this problem efficiently.

**Keywords:** Network Reliability, Count-to-Infinity, Network Convergence, RSTP.

## 1. INTRODUCTION

For last three decades, Ethernet is the most prominent local area network (LAN) technology. It can be seen everywhere from home offices to small offices and from medium size companies to even in large enterprises. Ethernet is usually preferred over its contemporary technologies – such as Fiber Distributed Data Interface (FDDI), Copper Distributed Data Interface (CDDI), Token Ring and Asynchronous Transfer Mode (ATM) – because of its low cost, market availability and scalability to higher bandwidths. Today, there are millions of Ethernet stations world-wide and large numbers of applications are running on them. Due to this ubiquity of Ethernet, and the ever-decreasing cost of the hardware needed to support it, most manufacturers now build the functionality of an Ethernet card directly into PC motherboards, obviating the need for installation of a separate network card.

Ethernet was originally developed at Xerox PARC in 1973. In its most basic form, Ethernet is a shared medium in which stations are not explicitly required to know location of each other. This scheme works well when the numbers of stations are few tens. As number of stations on the medium increases the performance and throughput of Ethernet decreases dramatically. To cope with this problem, Ethernet switches were introduced. Ethernet switch is a multi-port network device that forwards frame to specific ports rather than, as in conventional hub, broadcasting every frame to every port. In this way, the connections between ports deliver the full bandwidth available. Since these Ethernet switches works transparently, thus other network devices are completely unaware of their presence. With the introduction of switch, the performance, throughput and scalability of Ethernet has been significantly improved. Today most Ethernet networks are point-to-point switch networks. This study focuses on the dependability of such Ethernet networks under partial network failure.

Every Ethernet switch maintains a table – usually called forwarding table – in a local cache to forward incoming frames. Every entry in the forwarding table has a MAC address and the associated switch port. For each incoming frame, the switch looks up its destination address in the forwarding table to find the switch port to which the address is associated. If it is found, the frame is forwarded out that switch port. Otherwise the frame is forwarded in a “best effort” fashion by flooding it out all switch port except the one that received it. This is known as unknown unicast flooding. Further switches use dynamic address learning mechanism to populate forwarding table. When a frame **F** with a source address **S** arrives at switch port **P**, the switch create an entry in the forwarding table by assuming that the same port **P** can also be used to forward frame destined to **S**. Support of unknown unicast flooding and dynamic address learning by the switches impose a requirement that the underlying network must be cycle free. The reasons for this are two-fold. First is to avoid broadcast and unknown unicast frame from circulating forever in the network. Because, unlike IP, Ethernets frame do not have a Time-to-Live (TTL) field. Second is to prevent address learning mechanism from malfunctioning. Because a switch may receive frames from a station via multiple switch ports in cyclic network.

Ethernet networks usually have redundant links to increase network availability. However, networks that have redundant links also contain cycles and thus violating the requirement needed for proper functioning of switches. To alleviate this problem, switches in the network distributedly computes an active tree topology – by definition, tree has no cycle – over the underlying network that spans all the switches in the network to maintain full network connectivity. Each switch in the network places some of its ports in active mode while other in standby mode. Set of ports in the active modes form a spanning tree and only those are used for forwarding frames. Whereas ports in standby mode are reserve for use in case of link or switch failure. Protocols used by switches to compute such a tree topology are called spanning tree protocols.

The dependability of Ethernet therefore heavily relies on the spanning tree protocol under use. However, there are some serious concerns about the reliability of the current Ethernet standard spanning tree protocol – commonly known as Rapid Spanning Tree Protocol [1]. Cisco documented some pathological causes for forwarding loops in RSTP [2]. It also provides some proprietary solutions such as Loop Guard [3] and Unidirectional Link Detection (UDLD) protocol [4] to address few specific problems of RSTP. Elmeleegy et al. shows that count-to-infinity problem may be exhibited by RSTP under certain conditions [5]. They also proposed Etherfuse [6], a device to alleviate the problem of count-to-infinity in the existing network The incident of network disruption at the Beth Israel Deaconess Medical Center in Boston – in which the network suffered from disruptions for more than three days due to problems with the spanning tree protocol – also proved that the concerns about the reliability of RSTP are quite genuine. Vulnerability of RSTP against count-to-infinity problem is the main cause of its unreliability. This paper discuses count-to-infinity problem in spanning tree controlled network, in general, and in RSTP controlled network, in particular. The paper also provides a simple yet effective solution to prevent this problem by extending RSTP.

The rest of paper is organized as follows. Section 2 gives a brief overview of RSTP. Section 3 explains the conditions under which a spanning tree controlled network may suffer from count-to-infinity. Section 4 elaborates count-to-infinity problem in RSTP. Section 5 describes my proposed solution to this problem, the Delay Rapid Spanning Tree Protocol (DRSTP). Section 6 discusses related work. Followed by, Section 7 which concludes the paper.

## 2. OVERVIEW OF SPANNING TREE PROTOCOLS

Spanning Tree Protocol (STP) is the earlier standard Ethernet spanning tree protocol. It was first proposed by Perlman in [7]. Current standard Ethernet spanning tree protocol – known as Rapid Spanning tree Protocol (RSTP) [1] – is a modified version of STP. It inherits all the basic concepts of STP but design in such a manner that it is much faster than STP. RSTP [1] is here mainly due to the work of Mick Seaman presented in [8], [9], [10], and [11]. This section gives a brief overview of both these protocols. STP requires a unique identifier (ID) for every switch and every port within a switch. Using a distributed procedure, it elects the switch with the smallest ID as the root. A spanning tree is then constructed, based on the shortest route (path) from each switch to the root (switch and port IDs are used to break ties). The routing information is exchanged in the form of Bridge Protocol Data Units (BPDUs). The port that has received the best information for a route (path) to the root is called the root port. Other ports in the switch send BPDUs with their path cost to the root to other switches in the network. Ports that receive inferior information than the one they are sending are chosen to be designated ports. A port is said to be backup port if it receives superior information transmitted by its own switch. All remaining ports are alternate ports. Every switch brings its root port and its designated ports into a forwarding state thus only these ports are used to forward data frames. All remaining ports – alternate and backup ports – are kept in a blocking state and thus are not used for data forwarding.

In the event of a topology change, STP depends upon timers before switching ports to the forwarding state. This is to provide enough time for the new information to spread across the network. These conservative timers are used to guard against prematurely switching a port to the forwarding state that may lead to a forwarding loop. Due to these timers convergence time of STP may be up to 50 seconds [2]. Whenever a switch gets disconnected from the Root Switch, it waits until the information cached at its root port is aged out, then it starts accepting BPDUs from other switches to discover another path to the root.

In STP, only Root Switch generates BPDU. All non-root switches wait to receive them on their root ports then relay to their designated ports after adjusting the appropriate fields such as Root Path Cost, Sender Bridge Identifier etc. A switch losing a BPDU can be due to a problem anywhere along the path to the Root Switch.

RSTP [1] preserves all the basic concepts of STP but introduce few optimizations to reduce convergence time. Those are,

1. RSTP switches can process inferior BPDUs to detect topology changes.
2. When an RSTP switch is connected to point-to-point links, it uses handshake (sync), rather than timer to transition a Designated Port to forwarding state.
3. If the Root Port of a switch fails, RSTP can quickly retire the Port and make an Alternate Port its new Root Port. This new Root Port can be placed in the forwarding state without any delay.
4. In RSTP, every switch sends its own BPDUs whether it received one on its Root Port or not. RSTP switch expects to receive a BPDU within three Hello times. If the BPDU is not received within this time, the switch presumes it had lost connection with its neighbor. Of course, if a switch detects a loss of a link on its own port, it immediately assumes its neighboring connection is lost.

A topology change can result in the invalidation of a switch’s learned address location information. This requires the flushing of the forwarding table that caches stations’ locations. Both STP and RSTP [1] use some sort of address flushing mechanism. But address flushing mechanism of RSTP [1] is much faster than that of STP.

## 3. COUNT-TO-INFINITY IN SPANNING TREE CONTROLED NETWORKS

Count-to-infinity problem is not new to the world of routing. All known distance vector routing protocols such as RIP [12] and EIRP [13] employ some sort of mechanism to encounter this problem. However, this problem is still new to the world of switching. It was first mention by Mayer at el. [14] in 2004 that RSTP [1], a well known spanning tree protocol, may exhibit count-to-infinity problem under certain conditions. This highly undesirable behavior of RSTP was later studied in detail by Elmeleegy at el. [5]. This section will explain why and when a spanning tree protocol may become vulnerable to count-to-infinity.

In a fully converged spanning tree controlled network all alternate ports are dual rooted i.e. have two distinct path to the Root Switch. One path of an alternate port to the Root Switch passes through its link's designated port while the other path passes through its switch's root port. However an alternate port may loss its one or both paths to the Root Switch if the root port of its upstream switch fails. So in a network in which a switch suffering from the root port failure, an alternate port may have no, one or two path(s) to the Root Switch and thus will be called orphan, single rooted and dual rooted alternate port respectively in this text. Orphan alternate ports must not be used to reunite a network segregated due to the root port failure of a switch. Because such alternate ports have information which is no longer valid. Moreover, dual rooted alternate ports are not used by spanning tree protocols to prevent forwarding loops. This left only single rooted alternate ports that can be used to reunite the temporarily segregated network and they have the potential to do so. Hence the underlying spanning tree protocol must use only single rooted alternate ports to restore connectivity.

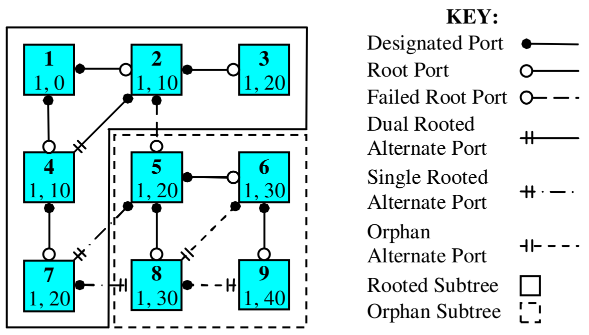


**FIGURE 1:** Different types of alternate ports in a network after failure of the root port of switch 5.

In a fully converged spanning tree controlled network, failure of the root port (or the designated port associated with the root port) of a switch results into segregation of underlying spanning tree into two distinct subtrees namely a *rooted subtree*, a subtree that still have the Root Switch, and an *orphan subtree*, a subtree that no longer have the previous Root Switch. It has to be noted that since all the switches in the *orphan subtree* have lost their path to previous Root Switch through their respective root ports. Therefore dual rooted alternate ports cannot exist in *orphan subtree*. In contrast, all the switches in *rooted subtree* have a path to the Root Switch through their respective root port. Hence orphan alternate ports cannot exist in *rooted subtree*. However, single rooted alternate ports can be found in both subtrees near their common boarder. An alternate port in the *rooted subtree* is single rooted if and only if its associated designated port is in the orphan sub tree. Similarly an alternate port in the *orphan subtree* is single rooted if and only if its associated designated port is in the *rooted subtree*. These facts are depicted in Figure 1 through an exemplary network. Each switch is represented by a small box. The top number in the box is the Switch ID, the lower set of numbers represents the Root Switch ID as perceived by the switch and the cost to this Root Switch. All links have cost of 10. Figure 1 shows the snapshot of network immediately after failure of the root port of switch **5**. Switches **1** to **4** and switch **7** are in *rooted subtree* and switch **5**, **6**, **8** and **9** are in *orphan subtree*. Alternate port of switch **4** is still dual rooted as it is inside the *rooted subtree*. Moreover, Alternate port of switch **7** and that of switch **8** connected to switch **7** are single rooted alternate ports as they are near the common boarder of two subtrees. While alternate of switch **8** connected to switch **6** and that of switch **9** are orphan alternate ports as they are inside the *orphan subtree*.

Switches in a spanning tree controlled network use messages to communicate with each other. These messages experience a transmission delay when passing through the network. Thus, failure of the root port of a switch may put all its downstream switches, that is switches in *orphan subtree*, into an inconsistent state for a period of time. The absolute period of inconsistence for a switch **B** is from the time when one of its upstream switch's root port (or the designated port associated with the upstream switch's root port) fails to the time when this information will be received on the root port and all alternate ports (if any) of the switch **B**. The effective period of inconsistence for a switch **B** is a bit small and it spans from the time when the first time switch **B** receives failure information of its upstream switch's root port on its root (or alternate) port to the time this will be received on all its remaining alternate port(s) (and the root port). Clearly, only inconsistent switches may have orphan alternate port(s) because of lack of information. Further, such switches cannot differentiate an orphan alternate port from the other two types of alternate ports.

Count-to-infinity only occurs in the part of network constituting the *orphan subtree*, if six conditions are satisfied simultaneously. Three of them have to be satisfied by an inconsistent switch **B**:

1. Switch **B** has an orphan alternate port **a** such that its root path cost is smaller than that of the best single rooted alternate port in the network.
2. Switch **B** starts to declare its orphan alternate port **a** as designated port or the root port when it is still in the effective inconsistent port or switch **B** is declaring its orphan alternate port **a** as designated port when it is entering into the absolute inconsistent state.
3. Switch **B** is injecting the stale BPDU through its retiring orphan alternate port **a** that is becoming designated port or through its retiring root port that is becoming the designated port because the orphan alternate **a** is becoming the new root port.

Two conditions must be satisfied by an upstream switch **A** along with above three conditions:

4. Switch **A** accepts the stale BPDU, transmitted by switch **B**, on its designated port **d**, as it is conceived as superior BPDU by switch **A**. This makes port **d** the new root port of switch **A**. It may happen only if the switch cannot differentiate between stale and fresh BPDUs.
5. Switch **A** begins to propagate the stale BPDU further through its now designated ports.

One condition needs to be met by underlying network.

6. There is at least one (unbroken) cycle in the network passing through switch **A**'s new root port **d** and switch **B**'s orphan alternate port a.

The first and the last condition for count-to-infinity are unavoidable in a high available fault tolerant network. However, remaining conditions can be easily avoided from being satisfied, by making slight modifications in underlying spanning tree protocol, to make the underlying network completely secure from the highly treacherous count-to-infinity problem.

When count-to-infinity occurs, the stale information begins to circulate in cycle and thus increments the root path cost of suffering switches with a definite offset, equal to the cycle's path cost, in each complete cycle. Theoretically speaking, count-to-infinity in the network may be temporary or absolute. Temporary count-to-infinity in the network terminates after a definite interval of time. On the other hand absolute count-to-infinity persists forever. Temporary count-to-infinity may occur in a temporarily segregated network, a segregated network that has at least one single rooted alternate port, in which a switch in *orphan subtree* mistakenly turns its orphan alternate port into root or designated port to reunite the segregated network. When this happen count-to-infinity lasts until root path cost of one of the suffering switch exceed to that of the best single rooted alternate port in the network. Absolute count-to-infinity may occur in an absolutely segregated network, a segregated network that has no single rooted alternate port in the network, in which a switch in *orphan subtree* wrongly moves its orphan alternate port in the root port or designated port to reunite the segregated network. As the best single rooted alternate port in the absolutely segregated network has the root path cost of infinity, so count-to-infinity will theoretically last forever.

Both absolute and temporary count-to-infinities are highly undesirable because they adversely effects the convergence time, and thus decreases network availability. They also lead to network congestion and packet loss. Count-to-infinity may induce forwarding loops [15] that results in further increase in the network congestion.

Backup port can be made designated port after failure of its corresponding designated port without any count-to-infinity into the network. The reason is two folded. First, all the root ports on the shared medium start to pretend like single rooted alternate ports that can provide a path to Root Switch through the backup port corresponding to the failed designated port. Second, the root path cost of these pretending single rooted alternate ports is better than that of all orphan alternate ports in the *orphan subtree* i.e. violation of condition 1 of six conditions required for count-to-infinity. Change in port cost of the root port of a switch also forces the port to act like a single rooted alternate port.

Elmeleegy et al. claimed in [5, 15] that injection of stale cached information at alternate port, because of declaring an orphan alternate port as the root port, may cause count-to-infinity. The above discussion further extends this claim by mentioning that injection of stale cached information of the root port of a switch into the network, through a designated port of the switch forming due to retirement of an orphan alternate port, also have potential to induce count-to-infinity into the network (see section 4 for illustrative elaboration).

## 4. COUNT-TO-INFINITY IN RSTP CONTROLLED ETHERNET NETWORKS

RSTP [1] is specifically designed to minimize the convergence time of Ethernet networks. To achieve this goal, RSTP switches uses cached information after an event of failure. But they perform no check to determine whether the received or cached information is fresh (valid) or stale (invalid). This aggressive and optimistic behavior of RSTP switches makes the underlying network highly vulnerable to count-to-infinity problem. This section will explain count-to-infinity in RSTP controlled network.

To illustrate the problem, I will give four specific examples and relate their behaviors to clauses in the IEEE 802.1D (2004) [1] standard. The 7 relevant rules that govern the operation of RSTP that are identified from the IEEE 802.1D (2004) [1] standard are given below.

1. A switch declares itself Root Switch if it perceives itself as the best switch of the network. This will happen if the switch has recently joined a network or it losses its current root port and it has no alternate port. (Clause 17.6).
2. Switches send its own Bridge Protocol Data Unit (BPDU) at regular intervals to guard against loss and to assist in the detection of failed components (LAN, switches and switch ports). (Clause 17.8).
3. A switch immediately transmits its own BPDU on its designated ports if the information it conveys has been changed i.e. when it believes the root has changed or its cost to the root has changed. (Clause 17.8).
4. A switch ages out a received BPDU after three consecutive misses. This is only if the switch cannot physically detect its failure. (Clause 17.21.23).
5. Switch assigns a port role to its each and every port as follows (Clause 17.7):
    a. A port becomes root if it is receiving the best BPDU.
    b. A port becomes alternate if it receives a superior BPDU from another switch and it is not root.
    c. A port becomes designated if receiving BPDU is inferior.
    d. A port becomes backup if it receives a superior BPDU from another port of this switch.
6. An alternate port of a switch can be immediately moved into forwarding state if its current root port has lost its status. (Clauses 17.10).

7. An arrived BPDU can be accepted if and only if it is better (numerically less) or it is from same designated switch and same designated port as that of receiving port's port priority vector. (Clause 17.6).

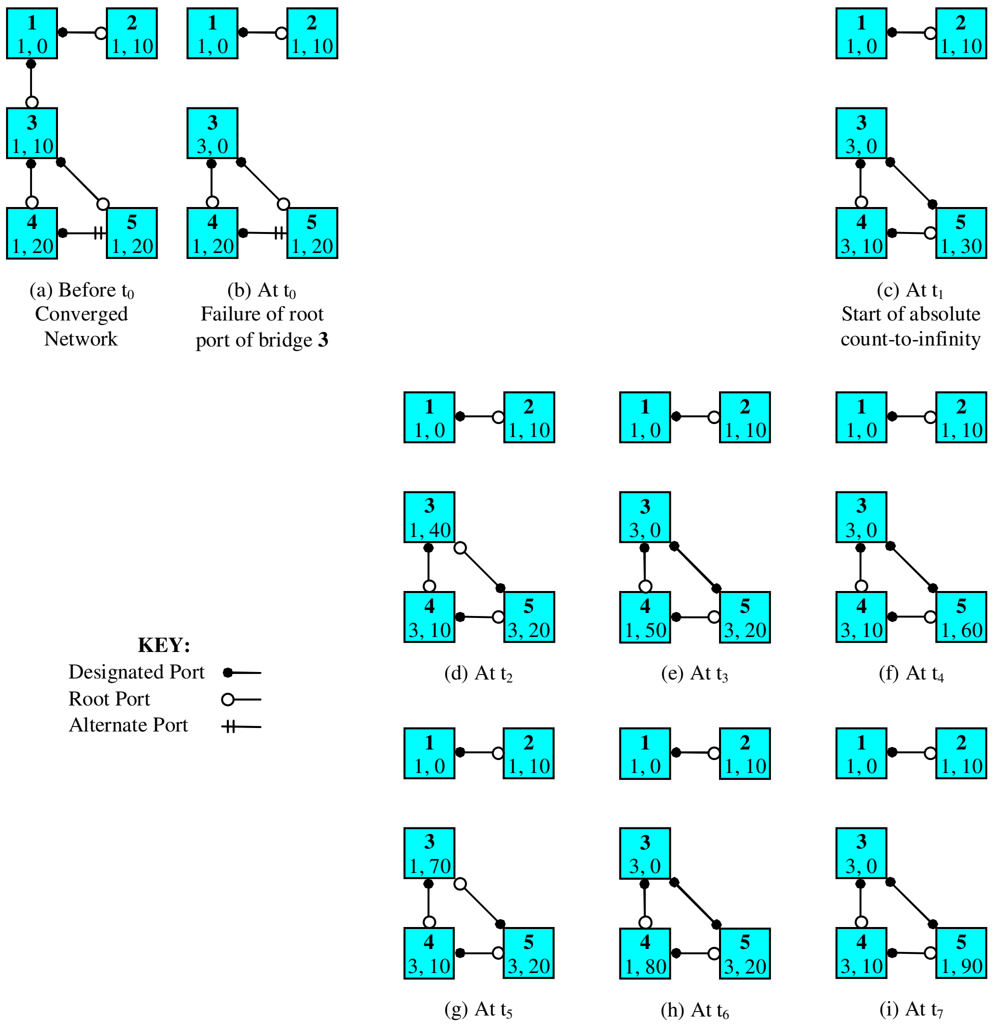


**FIGURE 2:** A network suffering from absolute count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as the new root port.

Now consider the network of switches shown in Figure 2. All links have cost of 10. Figure 2(a) shows the converged network before time $t_0$. At time $t_0$ the root port of switch 3 has failed (see figure 2(b)). This port failure divides the network into rooted and orphan subtrees. So switch **1** and **2** are in *rooted subtree* whereas switch **3**, **4** and **5** are in *orphan subtree*.

At time $t_0$, switch **3** performs the following actions (see figure 2(b));

1. As it realizes its root port has failed, it elects itself as the Root Switch since it has no alternate port (rule 1).
2. Immediately sends an inferior BPDU with itself as the Root Switch on all its designated ports (rule 3).

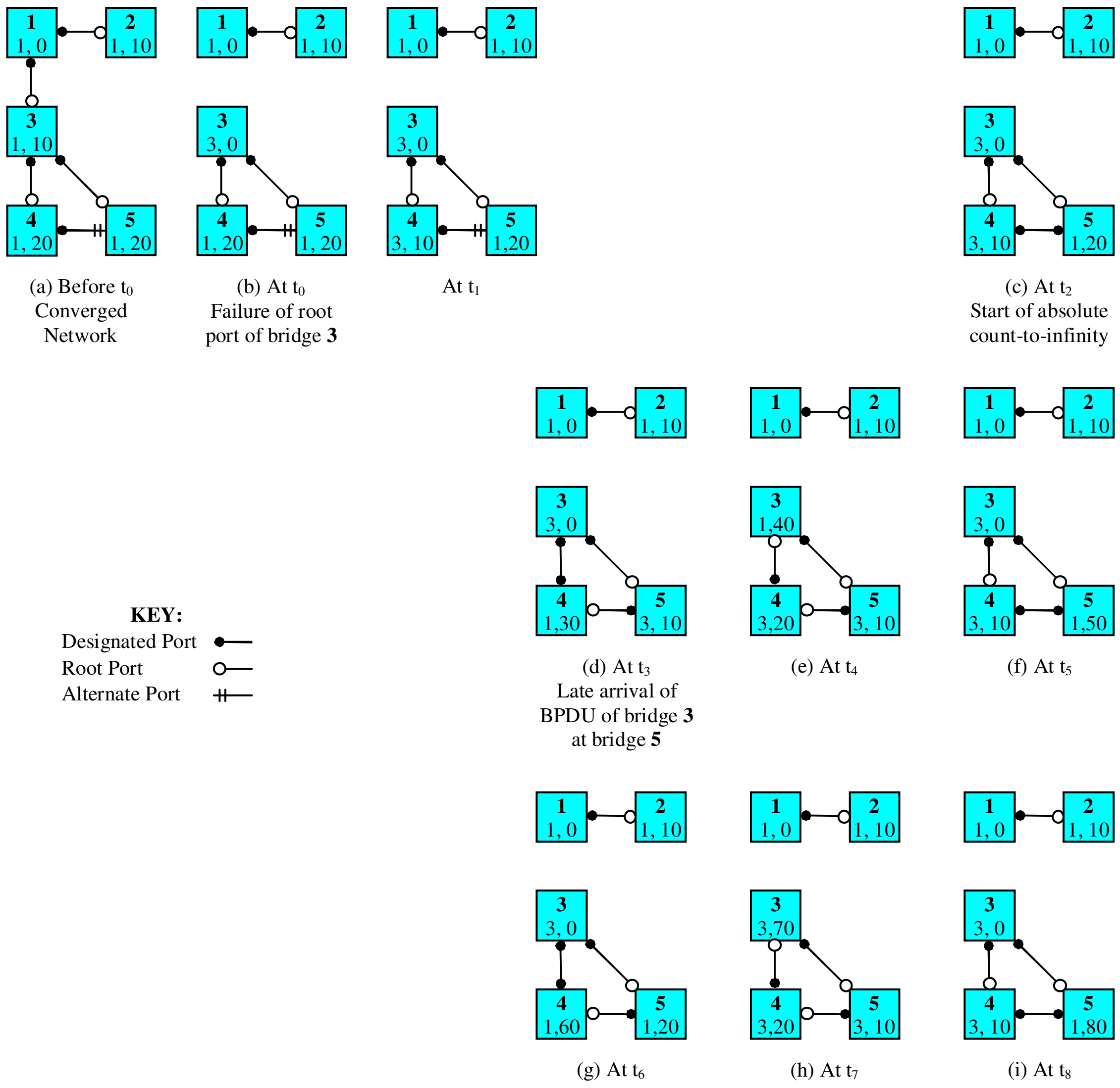


**FIGURE 3:** A network suffering from absolute count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as designated port.

Note that now both switch **4** and switch **5** are in an absolute inconsistent state because they still believe switch **1** as the Root Switch.

At time $t_1$, switch **4** takes the following actions (see figure 2(c));

1. Receives and accepts the inferior BPDU from switch 3 (rule 7).
2. Reelects its port to switch 3 as its root port but this time with switch 3 as the root (rule 5 a).
3. Immediately sends an inferior BPDU with switch 3 as root on its designated port (rule 3).

At time $t_1$, switch **5** executes the following tasks (see figure 2(c));

1. Receives and accepts the inferior BPDU from switch **3** (rule 7).
2. Incorrectly turns its orphan alternate port (a port connected to switch **4**) into root port to the now inaccessible root i.e. switch **1** (rule 6). This is because since at time $t_1$ switch **5** is in the effective inconsistent state.
3. Switch **5** injects the invalid information of its orphan alternate port into the network (rule 3) by sending the BPDU on its now designated port (a port connected to switch **3**) and thus initiate the count-to-infinity.

At time $t_2$, switch **5** performs the following actions (see figure 2(d));

1. Receives and accepts the inferior BPDU from switch **4** results in the end of effective inconsistent state (rule 7).
2. Reelects its port to switch **4** as its root port but this time with switch **3** as the root (rule 5 a).
3. Immediately sends this fresh but inferior BPDU on its designated port (rule 3).

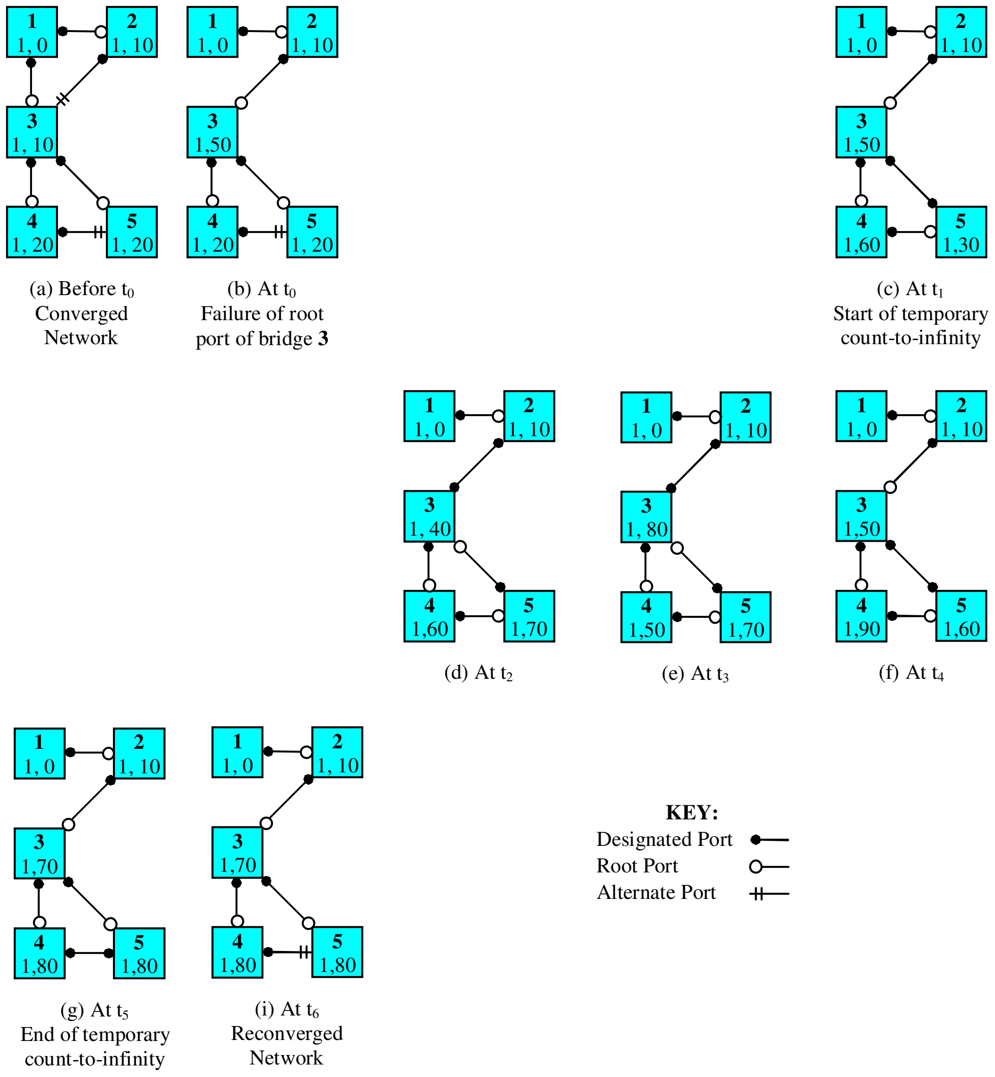

(a) Before t0 Converged Network
(b) At t0 Failure of root port of bridge 3
(c) At t1 Start of temporary count-to-infinity
(d) At t2
(e) At t3
(f) At t4
(g) At t5 End of temporary count-to-infinity
(i) At t6 Reconverged Network
KEY:
Designated Port
Root Port
Alternate Port

**FIGURE 4:** A network suffering from temporary count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as the new root port. Link between switch 2 and switch 3 has cost 40 whereas all other links have cost 10.

At time $t_2$, switch **3** performs the following actions (see figure 2(d));

1. Receives and accepts the stale BPDU from switch **5** (rule 7).
2. Reelects its port to switch **5** as its root port with switch **1** as the root (rule 5 a).
3. Immediately sends the stale BPDU on its designated port (rule 3).

For the rest of time stale BPDU with switch **1** as root, and fresh BPDU with switch **3** as root will chase each other.

Count-to-infinity may also occur in the considered network if switch **5** turns its orphan alternate port into designated port using rule 5 c. It is illustrated in figure 3. This will happen when the switch **5** receives switch **3**'s root port failure information on its alternate port (port connected to switch **4**) before it receives this information on its root port (port connected to switch **3**).

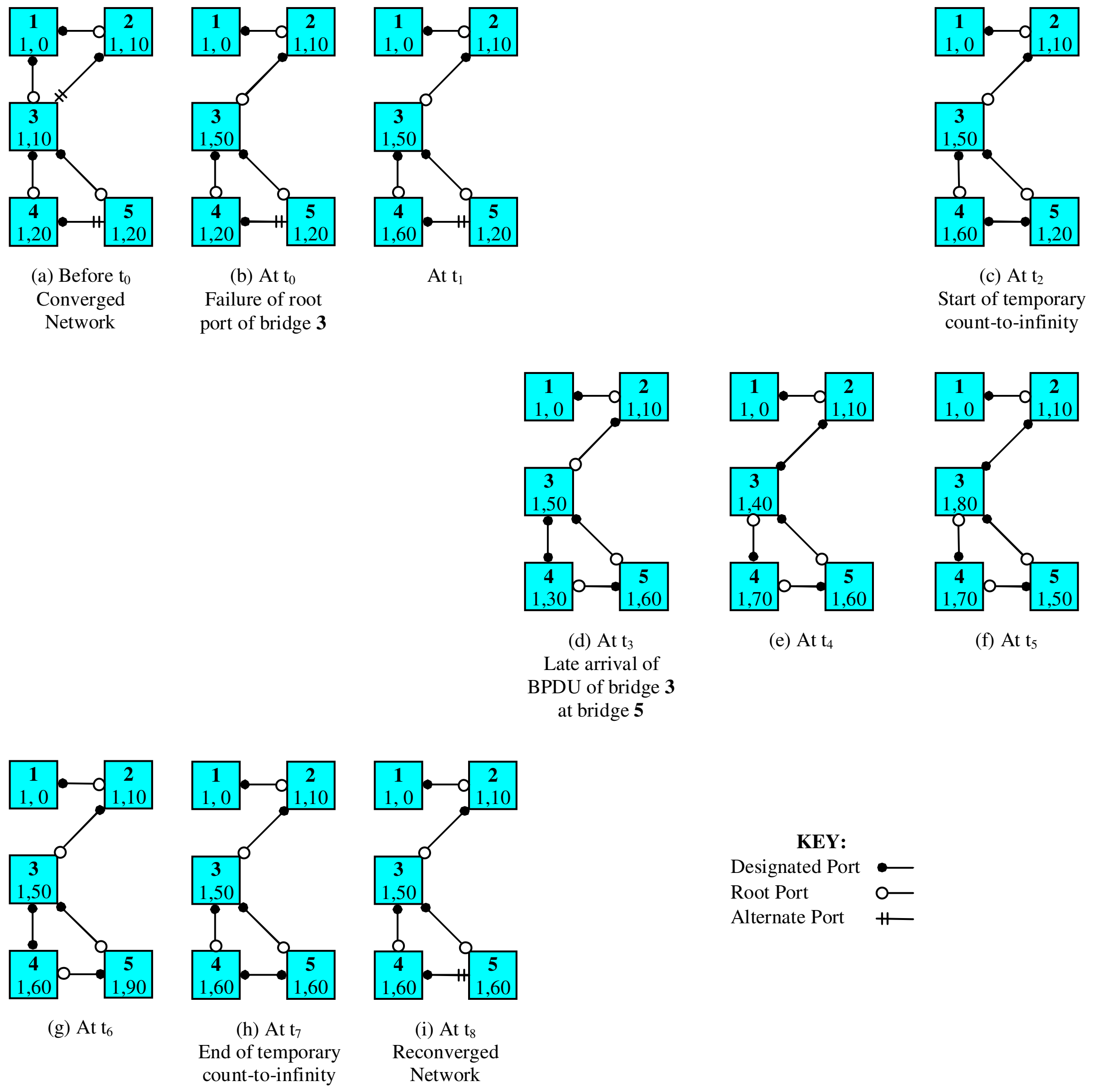

(a) Before $t_0$ Converged Network
(b) At $t_0$ Failure of root port of bridge **3**
At $t_1$
(c) At $t_2$ Start of temporary count-to-infinity
(d) At $t_3$ Late arrival of BPDU of bridge **3** at bridge **5**
(e) At $t_4$
(f) At $t_5$
(g) At $t_6$
(h) At $t_7$ End of temporary count-to-infinity
(i) At $t_8$ Reconverged Network
KEY:
Designated Port
Root Port
Alternate Port

**FIGURE 5:** A network suffering from temporary count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as designated port. Link between switch 2 and switch 3 has cost 40 whereas all other links have cost 10.

Rule 3, rule 5 c, rule 6 and rule 7 of RSTP play a vital role in inducing absolute count-to-infinity into the network. Rule 3 allows a switch to rapidly propagate the information of root port failure to downstream switches through its designated ports. On the other hand, rule 7 forces the downstream switches to accept this failure information. Moreover, when a switch receives this failure information it may turns its alternate port into root or designated port, even when it is in inconsistent state, on the bases of its invalid cached information (rule 5 c and rule 6). As a result count-to-infinity may induce into the network.

RSTP is also susceptible to temporary count-to-infinity. Figure 4 is showing a network that suffers from temporary count-to-infinity because switch 5 is declaring its orphan alternate port as new root port. Where as figure 5 is showing the same network suffering from temporary count-to-infinity as switch 5 is announcing its orphan alternate port as designated port. Rule 3, rule 5 c, rule 6 and rule 7 that play vital role in induction of absolute count-to-infinity are also responsible for temporary count-to-infinity. Same lines of reasoning that are used for explaining absolute count-to-infinity in RSTP can also be used for temporary count-to-infinity.

In summary, RSTP [1] is vulnerable to both absolute and temporary count-to-infinities. The reason is two folded. First, RSTP switches have tendency to use their alternate ports, to rapidly converge the network, even when they are in effective inconsistent state and so may inject stale (invalid) information into the network through their orphan alternate ports or through their retiring root ports. Second, RSTP switches cannot distinguish between stale (invalid) and fresh (valid) information (BPDU) and so stale information may last unnoticeably into the network for long time. This undesirable behavior of RSTP leads to unpredictable convergence time that may as high as tens of seconds [5], [14] and [15]. Count-to-infinity may also induce forwarding loop in RSTP controlled network that lead to network-wide congestion and packet loss as explained in [15].

## 5. DRSTP: THE DELAY RAPID SPANNING TREE PROTOCOL

Delay Rapid Spanning Tree Protocol – DRSTP – is an extension to RSTP. It is designed specifically to ensure that an Ethernet network converge as quickly as possible, after a link, port or switch failure, without inducing count-to-infinity into the network. The best thing about this solution is that it is completely backward compatible to legacy RSTP\STP switches.

DRSTP prevents count-to-infinity problem in mixed environment by simply forcing DRSTP switches to postpone transmission of BPDUs on recently retiring root or alternate port during the estimated period of effective inconsistence. Moreover, DRSTP switches also defer to transmit better BPDUs received from legacy switches for time equal to estimated period of effective inconsistence. This is to ensure that stale better BPDUs transmitted by legacy switches will not spoil the network. The next subsection will drive mathematically the estimated period of effective inconsistence. It is noteworthy that period of effective inconsistence for a bridge usually last for only few hundreds of microseconds in most cases.

### Derivation for Draining Out Stale BPDUs

In RSTP [1] cost of a link, by default, is inversely proportional to the bandwidth of the link and thus represents the time to transmit single bit on the link. Mathematically,

$$t = kc \qquad (1)$$

where, t is transmission time of single bit,
c is the cost of the link and
k is the constant of proportionality and it is, by default, equal to 0.05 picoseconds according to [1].
So the transmission time $T_{transmission}$ of a “n bits BPDU” is

$$T_{transmission} = nt$$
$$T_{transmission} = nkc \qquad (2)$$

Total time $T_{total}$ taken by BPDU can be defined as

$$T_{total} = T_{transmission} + T_{propagation} \qquad (3)$$

where, $T_{propagation}$ is time taken by BPDU to travel through the link.
But, in Ethernet $T_{propagation}$ is negligible, so
$$T_{total} \approx T_{transmission}$$
$$T_{total} = nkc \qquad (4)$$

Consider a network of switches as shown in figure 6. Let **R** be the Root Switch of that network. Consider a switch **F** such that $T_f$ be the total time taken by a BPDU send by Root Switch **R** to reach to switch **F**. Consider another switch **B** of the network. Let $T_r$ and $T_a$ be the time taken a BPDU, send by Root Switch **R**, take to reach the root port *r* and the alternate port *a* of switch **B** respectively. Suppose $c_r$ and $c_a$ be the root path cost of the root port *r* and alternate port *a* of switch **B** respectively.

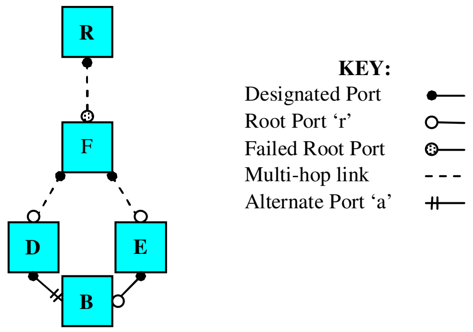


**FIGURE 6:** Network of switches used for deriving period of effective inconsistence for switch B.

So,

$T_r = nkc_r$ (5)

and

$T_a = nkc_a$ (6)

Suppose the root port of switch **F** fails. So the switch **F** sends a BPDU, announcing switch **F** as Root Switch, on all of its designated ports. It put switch **B** in *orphan subtree* and so in inconsistent state. Let $T_1$ and $T_2$ be the total time taken by the BPDU send by switch F to reach to the root port *r* and the alternate port *a* of switch B respectively. Hence, the period of effective inconsistence $\Delta T$ for switch **B** can be defined as the difference of time $T_2$ to time $T_1$ i.e.

$\Delta T = T_2 - T_1$ (7)

But

$T_r = T_f + T_1$ (8)

and

$T_a = T_f + T_2$ (9)

Using (5),(6),(8) and (9)

$\Delta T = nk(c_a - c_r)$ (10)

Since, *a* is the alternate port of switch **B**, so the following relation will hold:

$c_a - c_p \leq c_r \leq c_a$ (11)

where $c_p$ is port path cost of alternate port *a* of switch **B**

Therefore, using (11)

$\Delta T \leq nkc_p$ (12)

So, the estimated value of inconsistent port timer D, see section 5.2 for definition inconsistent port timer D, must be:

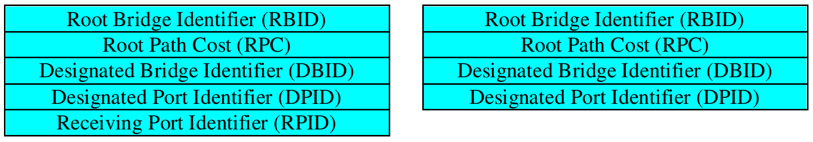
$D = nkc_p + t_{processing} + C \qquad (13)$
where, $t_{processing}$ is the average BPDU processing time
and $C$ is the additive constant to handle variations in $\Delta T$.

Moreover, if stale BPDU is injected by a switch in effective inconsistent state, it can be at most $nkc_p$ unit of time ahead of fresh BPDU. Hence, the estimated value of inconsistent port timer $D$ can also be use as that of count-to-infinity suppression timer $S$, see section 5.2 for definition count-to-infinity suppression timer $S$.

For network having slowest link of 10Mbps, average BPDU processing time $t_{processing}$ of 48.8µs and additive constant $C$ of 50µs, the estimated values for inconsistent port timer $D$ and count-to-infinity suppression timer $S$ are no more than 150µs. These are very small and quite acceptable values.

However, the above derivation is valid only with assumption that all links in the network have default cost and no BPDU loss is occurring. More careful network analysis is need for networks using non-default link cost to make good estimation of values of inconsistent port timer $D$ and count-to-infinity suppression timer $S$. However, it is expected that the two timers' value remains low for most commercial networks even when they are not using default link cost.

**Protocol Definition**
Like RSTP [1], operation of DRSTP can be defined precisely with the help of priority vectors. Figure 7 is showing the structure of an RSTP Priority Vector and RST (Configuration) BPDU respectively.


Root Bridge Identifier (RBID)
Root Path Cost (RPC)
Designated Bridge Identifier (DBID)
Designated Port Identifier (DPID)
Receiving Port Identifier (RPID)
Root Bridge Identifier (RBID)
Root Path Cost (RPC)
Designated Bridge Identifier (DBID)
Designated Port Identifier (DPID)

(a) RSTP Priority Vector (b) RST (Configuration) BPDU

**FIGURE 7:** Structure of RSTP Priority Vector and RST BPDU.

In detail, the DRSTP modifies the RSTP as follows:

1. An DRSTP switch associates two timers with each switch's port namely inconsistent port timer and count-to-infinity suppression timer.
2. In an DRSTP switch, a port is not eligible to transmit BPDUs, when inconsistent port timer is running on that port.
3. In an DRSTP switch, a port cannot participate in the root port election, if count-to-infinity suppression timer is running on it. So, such ports cannot become the root port.
4. An DRSTP switch divides received RST (Configuration) BPDUs into four distinct types namely Better RST (Configuration) BPDU, Repeated RST (Configuration) BPDU, Inconsistent RST (Configuration) BPDU and Worse RST (Configuration) BPDU. It is in contrast to RSTP which divides receiving BPDU into only three major types i.e. superior BPDU, repeated BPDU and inferior BPDU.
5. An DRSTP switch considers a received RST (Configuration) BPDU as Better RST (Configuration) BPDU if it is better (numerically less) than currently stored BPDU (Port Priority Vector).
6. An DRSTP switch handles a received RST (Configuration) BPDU as Repeated RST (Configuration) BPDU if it is same (numerically equal) as currently stored BPDU (Port Priority Vector).

7. An DRSTP switch treats a received RST (Configuration) BPDU as Worse RST (Configuration) BPDU if it is worse (numerically greater) than currently stored BPDU (Port Priority Vector) but it is not received from previous source. A receiving BPDU is said to be received from previous source if its Designated Bridge Identifier (DBID) and Designated Port Identifier (DPID) are equal to that of Port Priority vector of receiving port.
8. An DRSTP switch believes that the received RST (Configuration) BPDU is an Inconsistent RST (Configuration) BPDU if it is worse (numerically greater) than currently stored BPDU (Port Priority Vector) and it is received from previous source.
9. In DRSTP switch, a port starts its count-to-infinity suppression timer when it receives a better RST (Configuration) BPDU from a legacy RSTP (STP).
10. In DRSTP switch, a port starts its inconsistent port timer if it receives an Inconsistent RST (Configuration) BPDU.
11. DRSTP should be assigned a new protocol version. It enables DRSTP switches to differentiate between RST BPDUs transmitted by legacy RSTP switches and RST BPDUs transmitted by DRSTP switches.

**Discussion**

An STP switch discards an Inconsistent BPDU, a worse BPDU from previous source. This is the major cause of slow convergence of STP. Whereas, an RSTP switch handles an Inconsistent BPDU as if it were a better BPDU. But this behavior of RSTP switches makes it vulnerable to count-to-infinity problem. In contrast, an DRSTP switch considers an Inconsistent BPDU as a marker for beginning of effective inconsistent state. So, a port of an DRSTP switch starts its inconsistent port timer when it receives an Inconsistent BPDU. It prevents the port from injecting probably stale BPDUs through retiring alternate or root port and thus making violation of condition 3 of six conditions required for count-to-infinity. A port in an DRSTP switch starts its count-to-infinity suppression timer if it receives a Better BPDU from a legacy switch. A port running count-to-infinity suppression timer is not allowed to participate in root port election. This is because legacy switches have a tendency of injecting stale BPDUs when they are in effective inconsistent

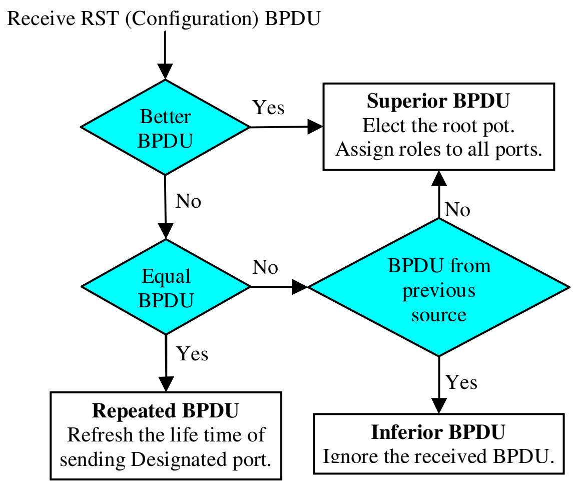

Receive RST (Configuration) BPDU
Better BPDU
Yes
Superior BPDU
Elect the root pot.
Assign roles to all ports.
No
No
Equal BPDU
No
BPDU from previous source
Yes
Yes
Repeated BPDU
Refresh the life time of sending Designated port.
Inferior BPDU
Ignore the received BPDU.

**FIGURE 8:** Processing of received RST (Configuration) BPDU in RSTP.

state. By disallowing a port to participate in root port election, the switch ensures violation of condition 4 of six conditions required for count-to-infinity.

DRSTP is backward compatible to both STP and RSTP. But DRSTP cannot prevent count-to-infinity in the presence of legacy STP switches in the network. This is because STP switches discards Inconsistent BPDUs and so making the period of effective inconsistence considerably high i.e. in the order of tens of seconds. Moreover, DRSTP is a count-to-infinity prevention technique, so count-to-infinity may occur in the network even in the presence of DRSTP switches.

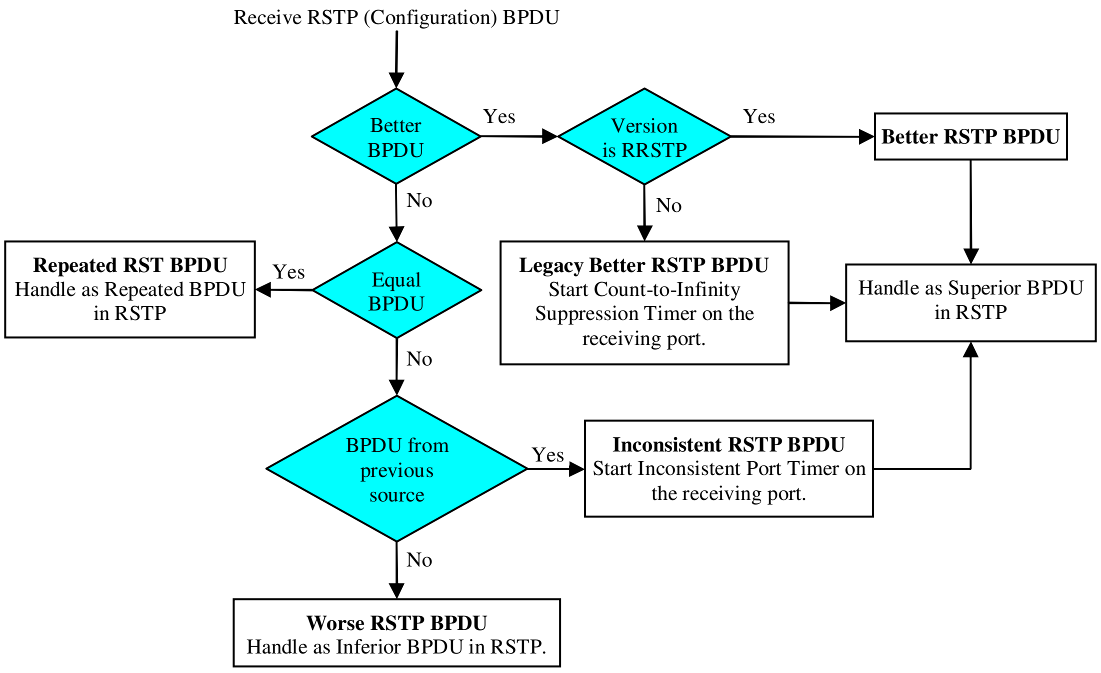

Receive RSTP (Configuration) BPDU
Better BPDU
Yes
Version is RRSTP
Yes
Better RSTP BPDU
No
No
Repeated RST BPDU Handle as Repeated BPDU in RSTP
Yes
Equal BPDU
Legacy Better RSTP BPDU Start Count-to-Infinity Suppression Timer on the receiving port.
Handle as Superior BPDU in RSTP
No
BPDU from previous source
Yes
Inconsistent RSTP BPDU Start Inconsistent Port Timer on the receiving port.
No
Worse RSTP BPDU Handle as Inferior BPDU in RSTP.

**FIGURE 9:** Procession of received RST (Configuration) BPDU in DRSTP.

Figure 8 is showing the processing of received RST (Configuration) BPDU by a legacy RSTP switch. Whereas, figure 9 is showing the processing of received RST (Configuration) BPDU by an DRSTP switch. Handling of port failure in RSTP and DRSTP is shown in figure 10.

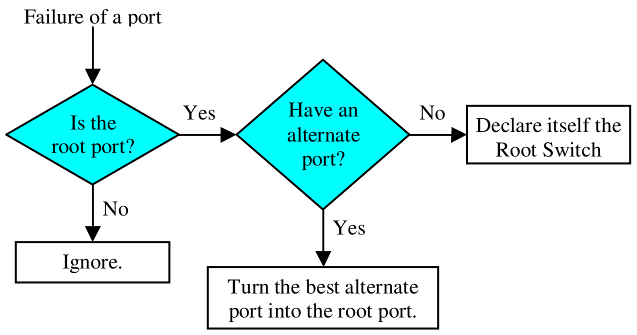

Failure of a port
Is the root port?
Yes
Have an alternate port?
No
Declare itself the Root Switch
No
Yes
Ignore.
Turn the best alternate port into the root port.

**FIGURE 10:** Handling of port failure in RSTP and DRSTP.

## Comparison With Contemporary Protocols

This section will critically discuss DRSTP with other contemporary protocols. The four other protocols that will be used for comparison are STP [7], RSTP [1], RSTP with Epoch [5][15] and Ether Fuse [6]. The five key aspects that will be discussed during comparison are vulnerability against count-to-infinity, convergence time, protocol implementation, extra hardware requirement, and backward compatibility.

Both STP [7] and RSTP [1] are susceptible to temporary and absolute count-to-infinities. In contrast, DRSTP provide protection, to some extend, against both type of count-to-infinities.

"RSTP with Epoch" is a new protocol that specifically designed to address the count-to-infinity problem but unfortunately it is vulnerable against temporary count-to-infinity.

STP exhibits very slow convergence time of up to 50s [2]. In contrast, RSTP may converge with in 1-3s due to its aggressive and optimistic approach. But this low convergence time is showed by RSTP only in absence of count-to-infinity. In contrast, DRSTP is expected to usually exhibit convergence time of 1-3s. Convergence time of RRSTP with Epoch is also comparable to that RSTP.

DRSTP is completely backward compatible to RSTP. It is also compatible to STP but at the expense of exposure to count-to-infinity. "RSTP with Epoch" is also backward compatible to legacy switches. But it does not ensure protection against count-to-infinity in mixed environment having legacy switches.

Ether Fuse [6] is a solution proposed by Elmeleegy et al. to protect network of legacy switches from adverse effects of count-to-infinity. Ether Fuse [6] uses a reactive approach to the problem that is at first it allows count-to-infinity to occur but stops it before it become severe. This approach of Ether fuse toward the problem is in sharp contrast with other protocols as they use a proactive approach. Ether Fuse is a completely standalone solution that has its own memory and hardware requirement. In contrast, DRSTP neither require extra memory nor hardware for deployment. Further, DRSTP can be built very easy and quickly using already available RSTP modules. In fact, DRSTP require subtle changes in only three state machine of RSTP namely Role Selection State Machine, Port Information State Machine and Transmit State Machine.

Hence DRSTP can be considered as an easy to implement backward compatible solution to reduce the occurrences of count-to-infinity in spanning tree controlled Ethernet networks with very little compromise on convergence time due to insertion of a very small delay of few hundred microseconds. As it is decreasing the frequency of cont-to-infinity, so the overall reliability of Ethernet networks will increase considerably.

| | | STP | RSTP | DRSTP | Ether Fuse | RSTP with Epoch |
|---|---|---|---|---|---|---|
| Frequency of Count-to-infinity | Temporary | High | High | Low | -- | High |
| | Absolute | High | High | Low | -- | Zero |
| Convergence time | In case of no count-to-infinity | Up to 50s | 1-3s | 1-3s | -- | Order of round trip time to Root Switch |
| | In case of count-to-infinity | Order of maximum message age | Order of maximum message age. | Order of maximum message age. | -- | Order of maximum message age |
| Approach to handle count-to-infinity | | N/A | N/A | proactive | reactive | proactive |
| Backward compatibility | | N/A | Yes | Yes | Yes | Yes |

**TABLE1:** Comparison of DRSRP with other contemporary protocols.

## 6. RELATED WORK

Reliability and scalability of Ethernet are main concerns for researchers for last two decades. Some researchers believe that reliability of Ethernet can be enhanced by use of link state routing protocols. One of such attempts is Rbridges that is proposed by Perlman [16]. Garcia et al. also proposed use of link state routing to substitute spanning tree [17].

Turn-prohibition is another technique used in Ethernet to improve scalability and reliability. Up/Down proposed by Schroeder et al. [18], Turn Prohibition (TB) proposed by Starobinski et al. [19], Tree-Based Turn-Prohibition (TBTP) proposed by Pellegrini et al. [20] and Hierarchal Up/Down Routing and Bridging Architecture (HURP/HURBA) proposed by Ibáñez et al. [21] are few well-known algorithms based on this technique.

SEATTLE proposed by Kim et al. [22] is a completely new layer 2 network architecture. However, it is not a backward compatible solution. Sharma et al. [23] introduce a multiple spanning tree architecture that improves the throughput and reliability over when using a single spanning tree. SmartBridges [24] uses the techniques of diffusing computation [25] and effective global consistency to achieve loop-freeness.

Instead of using other techniques, "RSTP with Epochs" proposed by Elmeleegy et al. [5] and [15] made an effort to increase reliability of spanning tree itself. It extends RSTP [1] to eliminate count-to-infinity. Unfortunately "RSTP with Epochs" [5] and [15] has no ability to handle count-to-infinity in mixed environment. Moreover, it cannot tackle temporary count-to-infinity problem even in full environment. DRSTP, an extension of RSTP, tries to mitigate count-to-infinity problem in mixed environment having legacy switches. It is completely backward compatible because it proposes changes only in interpretation of received BPDU.

## 7. CONCLUSION & FUTURE PLAN

This paper presents classical count-to-infinity problem in a novel fashion and point out that count-to-infinity can be temporary or absolute in a spanning tree controlled network. The paper then shows that RSTP [1] is susceptible to both temporary and absolute count-to-infinity. Spanning tree protocols like RSTP [1] that are exposed to count-to-infinity problem exhibit poor convergence, depending upon how long count-to-infinity situation persist. This paper also proposes a simple and effective solution – named as Delay Rapid Spanning Tree Protocol – to mitigate to count-to-infinity problem in RSTP. To achieve his goal, DRSTP inserts a small delay of few hundred microseconds before injecting its own cached information on recently retiring alternate or root port. Moreover, DRSTP hesitates to use a port as the root port for a small period of time when it is receiving Better BPDUs from legacy switches. Hence, it is expected that the solution will significantly enhance the dependability of Ethernet network without compromising much on its availability.

My future plan is to design a spanning tree protocol that will provide guaranteed protection against both absolute and temporary count-to-infinities.

## 8. ACKNOWLEDGEMENT

I would like to express gratitude to my parents for their unconditional support. I would also like to acknowledge the efforts of cooperative team of IJCN in making my maiden publication possible.

## 9. REFERENCES

1. LAN/MAN Standards Committee of the IEEE Computer Society. *"IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges - 802.1D"*. 2004.

2. Cisco Systems, Inc. *"Spanning Tree Protocol Problems and Related Design Considerations"*. Available: www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a00800951ac.shtml

3. Cisco Systems, Inc. *"Spanning-Tree Protocol Enhancements using Loop Guard and BPDU Skew Detection Features"*. Available: www.cisco.com/warp/public/473/84.html

4. Cisco Systems, Inc. *"Understanding and Configuring the Unidirectional Link Detection Protocol Feature"*. Available: www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a008009477b.shtml

5. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *“On Count-to-Infinity Induced Forwarding Loops in Ethernet Networks”*. In IEEE Infocom 2006.

6. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *“EtherFuse: An Ethernet Watchdog”*. In ACM SIGCOMM 2007.

7. R. Perlman. “*An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN”*. In the proceedings of 9th ACM Data Communications Symposium. New York, USA, 1985.

8. M Seaman. *“High Availability Spanning Tree”*. Available: www.ieee802.org/1/files/public/docs1998/hasten7.pdf.

9. M. Seaman. *“Speedy Tree Protocol”*. Available: www.ieee802.org/1/files/public/docs1999/speedy_tree_protocol_10.pdf.

10. M. Seaman. *“Truncating Tree Timers”*. Available: www.ieee802.org/1/files/public/docs1999/truncating_tree_timing_10.pdf.

11. V. Jain and M. Seaman. *“Faster flushing with fewer addresses”*. Available: www.ieee802.org/1/files/public/docs1999/faster_flush_10.pdf.

12. G. Malkin. *“RIP version 2”*. RFC 2453. Nov 1998.

13. Cisco Systems, Inc. *“Enhanced Interior Gateway Routing”* Available www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094cb7.shtml.

14. Myers, T. E. Ng, and H. Zhang. *“Rethinking the Service Model: Scaling Ethernet to a Million Nodes”*. In 3rd Workshop on Hot Topics in networks. 2004.

15. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *“Understanding and Mitigating the Effects of Count to Infinity in Ethernet Networks”*. IEEE/ACM Transactions on Networking, February 2009.

16. R. Perlman. *“Rbridges: Transparent routing”*. In IEEE Infocom 2004.

17. R. Garcia, J. Duato and F. Silla. *“LSOM: A link state protocol over MAC addresses for metropolitan backbones using optical Ethernet switches”*. In 2nd IEEE International Symposium on Network Computing and Applications. 2003.

18. M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, C. Thacker. *“Autonet: A High-Speed, Self–Configuring Local Area Network Using Point–to–Point Links”*. IEEE Journal on Selected Areas in Communications, 9(8):1318–1335, 1991.

19. D. Starobinski, G. Karpovsky, F. Zakrevsky. *“Applications of network calculus to general topologies”*, IEEE/ACM Transactions on Networking, 11(3):411–422, 2003.

20. F. D. Pellegrini, D. Starobinski, M. G. Karpovsky and L. B. Levitin. *“Scalable cycle-breaking algorithms for gigabit Ethernet backbones”*. In IEEE Infocom 2004.

21. Guillermo Ibáñez, Alberto García-Martínez, Juan A. Carral, Pedro A. González, Arturo Azcorra, José M. Arco. *“HURP/HURBA: Zero-configuration hierarchical Up/Down routing and bridging architecture for Ethernet backbones and campus networks”*, Computer Networks, 54(1):41-56,2010.

22. C. Kim, M. Caesar, and J. Rexford. "*Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises"*. In ACM SIGCOMM. 2008.

23. S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. Viking: "A multispanning tree Ethernet architecture for metropolitan area and cluster networks". In IEEE Infocom. 2004

24. T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson. *"SmartBridge: A scalable bridge architecture"*. In ACM SIGCOMM. 2000.

25. E. W. Dijkstra, C. S. Scholten. *"Termination detection for diffusing computations"*. Information Processing Letters, 11(1):14, 1980.