理解透彻--802.1d，802.1w，802.1s

一、STP：

在谈本主题之前，先简单的对 STP(802.1d)做个回顾.STP 是用于打破 2 层环路的协议，但这个协议有个最明显的缺点，就是当层 2 网络重新收敛的时候，至少要等待 50 秒的时间(转发延迟+老化时间).50 秒的时间对于一个大型的层 2 网络来说，是一个漫长的过程(何况这只是个理论时间，实际情况还会更长).虽然 Cisco 对 STP 的这些缺点开发出了些弥补性的特性，比如 Port Fast，Uplink Fast 和 Backbone Fast，用于加快层 2 网络的收敛时间——套用王朔的话"看上去很美".虽然这些"新"特性能够改善 STP 的一些不足，但是，这些特性是 CISCO 私有的，而非行业标准；此外，这些特性要求我们做额外的配置，如果缺乏对这些技术的理解，还有可能导致环路问题.

802.1d 标准中对端口状态的定义有：

1. *禁用(disabled).*
2. *堵塞(blocking).*
3. *监听(listening).*
4. *学习(learning).*
5. *转发(forwarding).*

二、RSTP：

RSTP 是 IEEE 802.1w 标准定义的，目的就是为了改进 STP 的一些不足，并且在某些情况下，RSTP 要比之前所提到的那些 Port Fast，Uplink Fast 和 Backbone Fast 技术更为方便.Cisco 的 Catalyst 交换机中不支持纯 RSTP，而支持的是 RPVST 或叫 RPVST+.但是在比较古老的交换机型号中比如 CATALYST 2900XL/3500XL 里，不支持 RSTP 与 RPVST+(或叫 RPVST+)，还有些型号比如 CATALYST 2948G-L3/4908G-L3，CATALYST 5000/5500 和 CATALYST 8500 不支持 RSTP.

802.1w 标准中对端口状态的定义有：

1. *丢弃(discarding).*
2. *学习(learning).*
3. *转发(forwarding).*

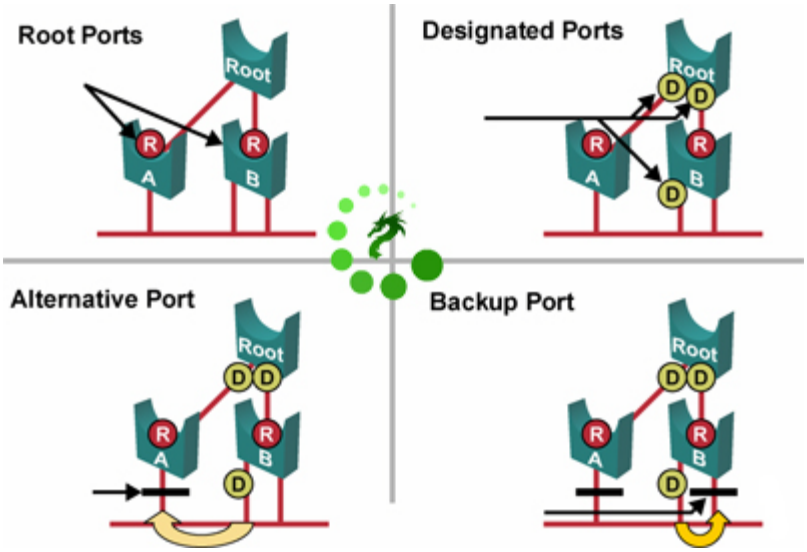
丢弃状态，实际上就类似 802.1d 中禁用、阻塞、监听状态的集合.

在 802.1w 中，根端口(root port，RP)和指定端口(designated port，DP)仍然得以保留；而堵塞端口被改进为备份端口(backup port,BP)和替代端口(alternate port,AP).不过,生成树算法(STA)仍然是依据 BPDU 决定端口的角色.和 802.1d 中对 RP 的定义一样，到达根桥(root bridge)最近的端口即为RP.同样的,每个桥接网段上,通过比较BPDU,决定出谁是DP.一个桥接网段上只能有一个DP(同时出现两个的话就形成了层 2 环路).

在 802.1d 中，非 RP 和 DP 的端口，它的状态就为丢弃状态，这种状态虽然不转发数据，但是仍然需要接收 BPDU 来保持处于丢弃状态.AP 和 BP 同样也是这样.

AP 提供了到达根桥的替代路径，因此，一旦 RP 挂掉后，AP 可以取代 RP 的位置.

BP 则提供了一条 DP 的备份冗余，如果当前网桥有两条路径到达桥接网段，那么一个端口为 DP，一个端口则为 BP，BP 有一个比 DP 高的 Port ID，所以它不转发任何数据，当 DP 断掉，BP 就变为 DP.

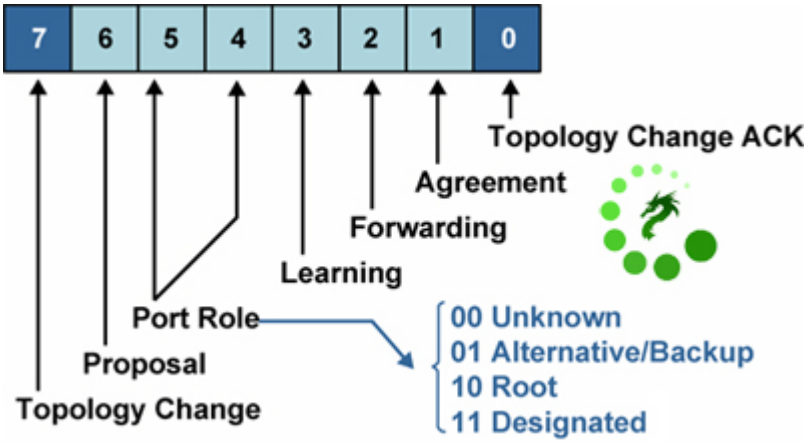


在 RSTP 里，BPDU 的格式稍稍变化了一些，在 802.1d 里，BPDU 只有两个标签选项：

1. 拓扑改变(TC).

2. 拓扑改变确认(TCA).

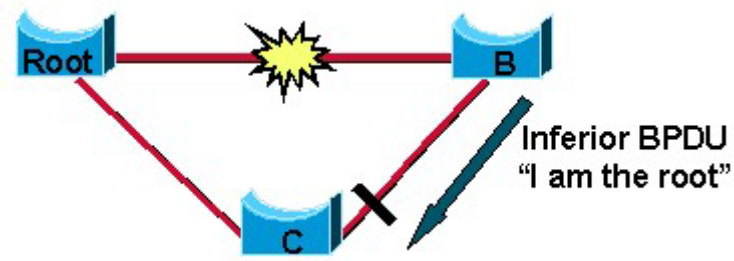
而 RSTP 中的 BPDU 采用的是版本 2 的 BPDU，换句话说 802.1d 网桥不兼容这种新的 BPDU.这种新的 BPDU，在原先的 BPDU 基础上增加了 6 个标签选项：



BPDU 的处理方式，和 802.1d 也有些不同，取代原先的 BPDU 中继方式(非根桥的 RP 收到来自根桥的 BPDU 后,会重新生成一份 BPDU 朝下游交换机发送出去),802.1w 里的每个网桥,在 BPDU 的 hello time(默认 2 秒)时间里都将生成 BPDU 发送出去(即使没有从根桥那里接收到任何 BPDU).如果在连续 3 个 hello time 里没有收到任何 BPDU，那么 BPDU 信息将超时不被予以信任.因此，在 802.1w 里，BPDU

更像是一种保活(keepalive)机制.即如果连续三个 HelloTime 未收到 BPDU，那么网桥将认识它丢失了到达相邻网桥 RP 或 DP 的连接.这种快速老化的方式使得链路故障可以很快的被检测出来.

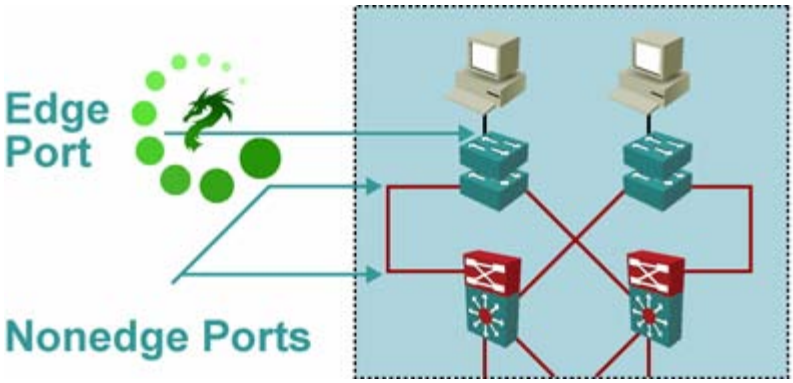
在 RSTP 里，类似 Backbone Fast 的下级 BPDU(inferior BPDU)也被集成进去.当交换机收到来自 RP 或 DP 的下级 BPDU 时，它立刻替换掉之前的 BPDU 并进行存储：



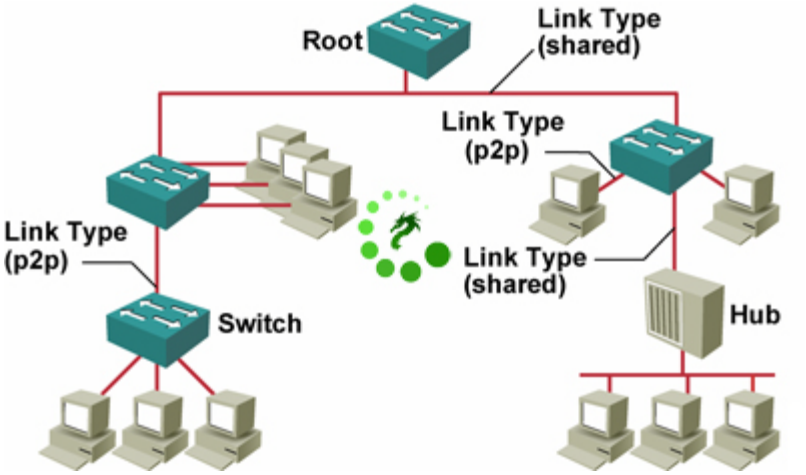
如上图，由于 C 知道根桥仍然是可用的，它就立刻向 B 发送关于根桥的 BPDU 信息.结果是 B 停止发送它自己的 BPDU，接收来自 C 的 BPDU 信息并将连接到 C 的端口做为新的 RP.

传统的 802.1d 标准里，STA 是被动的等待层 2 网络的收敛(由于转发延迟的定义).若对 STP 默认的计时器进行修改，又可能会导致 STP 的稳定性问题；而 RSTP 可以主动的将端口立即转变为转发状态,而无需通过调整计时器的方式去缩短收敛时间.为了能够达到这种目的,就出现了两个新的变量：边缘端口(edge port)和链路类型(link type).

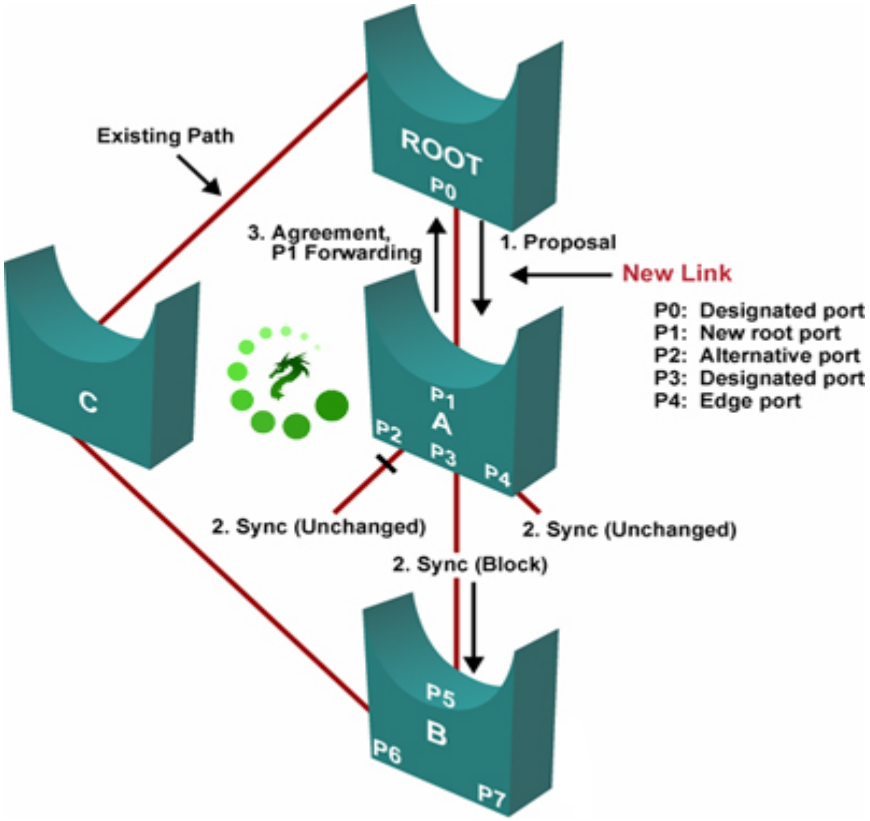
边缘端口(EP)的概念，和 CISCO 中 Port Fast 特性非常相似.由于连接端工作站的端口，是不可能导致层 2 环路的，因此这类端口就没有必要经过监听和学习状态，从而可以直接转变为转发状态.但是和 Port Fast 不同的是，一旦 EP 收到了 BPDU，它将立即转变为普通的 RSTP 端口.



RSTP 快速转变为转发状态的这一特性，可以在 EP 和点到点链路上实现的.由于全双工操作的端口被认为是点到点型的链路;半双工端口被认为是共享型链路.因此 RSTP 会将全双工操作的端口当成是点到点链路，从而达到快速收敛.



当一个新交换机加入拓扑中时，对于 802.1d，当 STA 决定出 DP 后，仍然要等待 30 秒的转发延迟才能进入转发状态；而在 802.1w 里：



假设根桥和交换机 A 之间创建了一条新的链路，链路两端的端口刚开始均处于丢弃状态，直到收到对方的 BPDU. 当 DP 处于丢弃或者学习状态，它将在自己将要发送出去的 BPDU 里设置提议位(proposal bit)，如上图的 p0 和步骤 1. 由于交换机 A 收到了上级(superior)信息，它将意识到自己的 P1 应该立即成为 RP. 此时交换机 A 将采取同步(sync)动作，将该上级 BPDU 信息洪泛到其他的所有端口上并保证这些端口处于同步状态(in-sync).

当端口满足下列标准之一时，即处于同步状态：

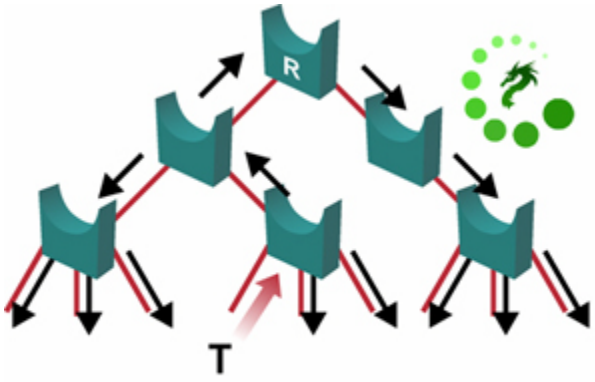
1. 端口为 EP.

2. 端口为堵塞状态(即丢弃，或者为稳定拓扑).

假设交换机 A 的 P2 为 AP，P3 为 DP，P4 为 EP. P2 和 P4 满足上述标准之一，因此为了处于同步状态，交换机 A 将堵塞 P3，指定它为丢弃状态，其他端口处于同步状态(步骤 2). 交换机 A 将解除 P1 的堵塞并做为新的 RP，并向根桥反馈确认信息(步骤 3)，这个信息其实是之前步骤 1 所发的提议 BPDU 信息的拷贝，只不过是把提议位设置成了认可位(agreement bit). 当 P2 收到这个认可信息后，它立即进入转发状态. 由于 P3 之前被堵塞了，当步骤 3 完成后，P3 也执行之前 P0 所经过的步骤 1，向下游交换机发出提议 BPDU 信息，尝试快速进入转发状态. 依次类推.

由于提议机制非常迅速，因此 RSTP 不需依赖任何计时器. 如果一个指定为丢弃状态的端口，在发出提议 BPDU 信息后没有收到认可信息，该端口会回退到 802.1d 标准，从监听到学习，再慢慢进入转发状态. 这种情况多发生在不理解 RSTP BPDU 的交换机端口上.

RSTP 里另外一个快速进入转发状态的机制，和 CISCO 对 STP 的扩展技术 Uplink Fast 很相似. 当网桥丢失了 RP 后，它会把自己的 AP 直接设置为转发状态(新的 RP). 因此对于 RSTP 来说，Uplink Fast 的特性无需手动配置. 还有一点和 802.1d 不同的是，当交换机检测到拓扑变化后，产生 TC 信息，直接洪泛给整个网络，而无需像 802.1d 那样先报告给根桥：



The originator of the TC directly floods this information through the network.

三、MST：

MST 是由 IEEE 802.1s 标准制定，来自 CISCO 私有的 MISTP 协议(Multiple Instances Spanning Tree Protocol). 和 RSTP 一样，MST 在某些 CATALYST 交换机上也不支持，比如：CATALYST 2900/3500XL，CATALYST 2948G-L3/4908G-L3，CATALYST 5000/5500 以及 CATALYST 8500.

在谈 MST 之前先说说关于 trunk 的原始版本 IEEE 802.1q，该标准制定了 CST(Common Spanning Tree). CST 假定整个层 2 网络只有一个 STP 的实例，也就是说不管整个层 2 网络划分了多少个 VLAN，都只有一个 STP 的实例. CST 的一些优劣：

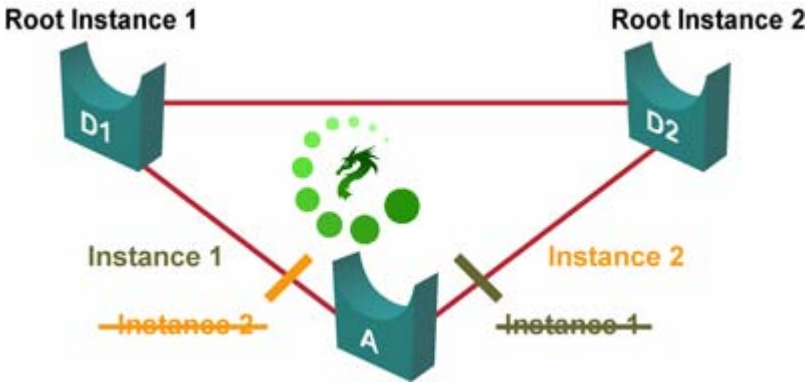
缺点：无法实现 STP 的负载均衡.

优点：节约 CPU 资源，整个层 2 网络只需要维护一个 STP 的实例.

而后续的 802.1q 增强了对 VLAN 的支持，出现了 PVST+(每 1 个 VLAN 有 1 个 STP 的实例).

802.1s 结合了 PVST+和 802.1q 的优点，将多个 VLAN 映射到较少的 STP 实例.之前的 PVST+的优点是可以实现 STP 的负载均衡，但对 CPU 资源是个负担.而 MST 减少了不必要的 STP 的实例.

如下图，假设 D1 和 D2 分别为 VLAN 1 到 500 和 VLAN 501 到 1000 的根桥，如果用 PVST+，就将有 1000 个 STP 的实例，但是实际上整个层 2 网络只有 2 个逻辑拓扑，所以优化办法是将 STP 的实例减少到 2 个，同时保留 STP 负载均衡的优点：



从技术角度来看，MST 的确是最佳解决方案，但是对端用户而言却并不是必需的，因为 MST 通常要求比 802.1d 和 802.1w 更为复杂的配置，并且还可能遇到与 802.1d 的协同操作问题.

之前我们提到，多个 VLAN 可以映射到一个 STP 的实例上.但是，决定哪个 VLAN 和哪个 STP 实例相关联，以及 BPDU 的标签方式以便交换机可以鉴别出 VLAN 与 STP 实例信息，这是个问题.这个时候就出现了一个类似 BGP 里 AS 的概念：区域(Region).MST 的区域是指处于同一管理范围的交换机组.为了能够成为 MST 区域里的一部分，交换机必须享有相同的配置属性：

1.以 26 个字母命名的配置名(32 字节).

2.配置修正号(2 字节).

3.对应 4096 个 VLAN 的元素表.

在做 VLAN 到 STP 实例映射的时候，要先定义 MST 的区域，但这个信息不会在 BPDU 里传播，因为对于交换机来说，它只需要知道自己和邻居交换机是否处于同一个 MST 区域.因此，只有一份 VLAN 到 STP 实例的映射摘要信息，配置修正号与配置名随着 BPDU 被传播出去.当交换机端口收到该 BPDU 后，它将解读该摘要信息，和自身的摘要信息做个比较，如果结果不同，那么该端口将成为 MST 区域的边界：

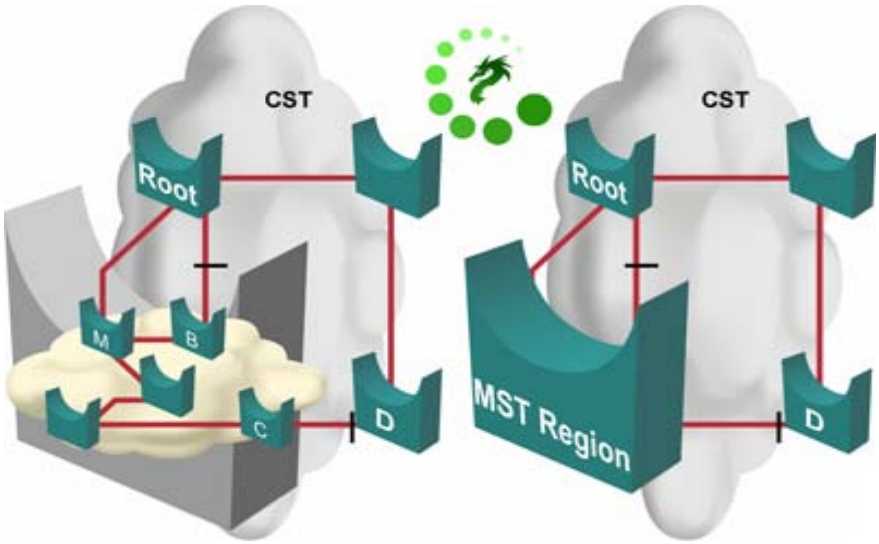
根据 802.1s 的定义，MST 网桥必须能够处理至少两种实例：

1.一个 IST(Internal Spanning Tree).

2.一个或多个 MSTI(Multiple Spanning Tree Instance).

当然到目前为止，802.1s 只是个"准标准"，这些术语可能随着最终版的 802.1s 而有不同的叫法.CISCO 支持 1 个 IST 和 15 个 MSTI.

由于 MST 源自 IEEE 802.1s，因此，要必须让 802.1s 和 802.1q(CST)协同操作.IST 实例向 CST 发送或从 CST 那里接收 BPDU.IST 实例其实是 RSTP 实例的简化，它扩展了 MST 区域里的 CST.IST 可以看做 CST 外部的整个 MST 区域的代表：



如上图，这两种图例职能相同.在典型的 802.1d 环境里，你可能会看到堵塞 M 和 B 之间的通信；同样的，你可能期望堵塞图中 MST 区域里的某个端口(而不是堵塞 D 的端口).但是，由于 IST 是做为整个 MST 区域的代表，因此，你看到的就是对 B 和 D 的堵塞.

MSTI 也是 RSTP 的简化版实例，它只存在于 MST 区域的内部.MSTI 默认自动运行 RSTP，而无需额外的配置.和 IST 不同的是，MSTI 永远不会和 MST 区域外部通信.另外，只有 IST 会向 MST 区域外发送 BPDU，而 MSTI 不会.在 MST 区域内，网桥相互交换 MST BPDU，这些 MST BPDU 对 IST 来说可以看成是 RSTP BPDU.

配置 MST 示例：

Switch(config)# spanning-tree mst configuration

/---进入 MST 配置模式---/

Switch(config-mst)# instance 1 vlan 10-20

/---将 VLAN 10 到 20 映射到实例 1 里，VLAN 范围为 1-4094，实例范围为 0-4094---/

Switch(config-mst)# name region1

/---命名 MST 区域，32 字节长的字符，大小写敏感---/

Switch(config-mst)# revision 1

/---配置修正号，范围是 0 到 65535---/

Switch(config-mst)# show pending

/---显示等待用户确认的配置信息---/

```
Pending MST configuration
Name                [region1]
Revision            1
Instance            Vlans Mapped
------------------------------------
0                   1-9，21-4094
1                   10-20
------------------------------------
```

Switch(config-mst)# exit

/---应用配置并退出 MST 配置模式---/

Switch(config)# spanning-tree mode mst

/---启用 MST，同时让 RSTP 生效---/

指定 MST 根桥与配置 MST 网桥的优先级：

Switch(config)# spanning-tree mst {instance-id} root {primary|secondary} [diameter net-diameter [hello-time ses]]

对于 MST，半径范围只能为 0；默认配置信息 2 秒发送 1 次，可选修改范围为 1-10 秒.

Switch(config)# spanning-tree mst {instance-id} priority {priority}

端口优先级的值范围是 0-61440，以 4096 递增，值越低，优先级越高，默认为 32768.

配置 MST 端口优先级与路径开销：

Switch(config)# spanning-tree mst {instance-id} port-priority {priority}

端口优先级的值范围是 0-240，以 16 递增，值越低，优先级越高.

Switch(config)# spanning-tree mst {instance-id} cost {cost}

路径开销的值范围是 1 到 200000000，取决于接口带宽.

配置 MST 的相关计时器：

Switch(config)# spanning-tree mst hello-time {sec}

默认配置信息 2 秒发送 1 次，可选修改范围为 1-10.

Switch(config)# spanning-tree mst forward-time {sec}

默认转发延迟为 15 秒，可选修改范围为 4-30.

Switch(config)# spanning-tree mst max-age {sec}

指定 MST 实例的最大生存周期，默认为 20 秒，可选修改范围为 6-40.

指定 BPDU 的最大跳数：

Switch(config)# spanning-tree mst max-hops {hop-count}

默认为 20 跳，可选修改范围为 1-255.

Switch(config-if)# spanning-tree link-type point-to-point

定义链路类型为点到点：

一些验证命令：

Switch#show spanning-tree mst configuration

验证 MST 区域信息.

Switch#show spanning-tree mst [instance-id]

验证 MST 实例信息.

Switch#show spanning-tree mst interface [interface-id]

验证特定接口的 MST 实例信息.