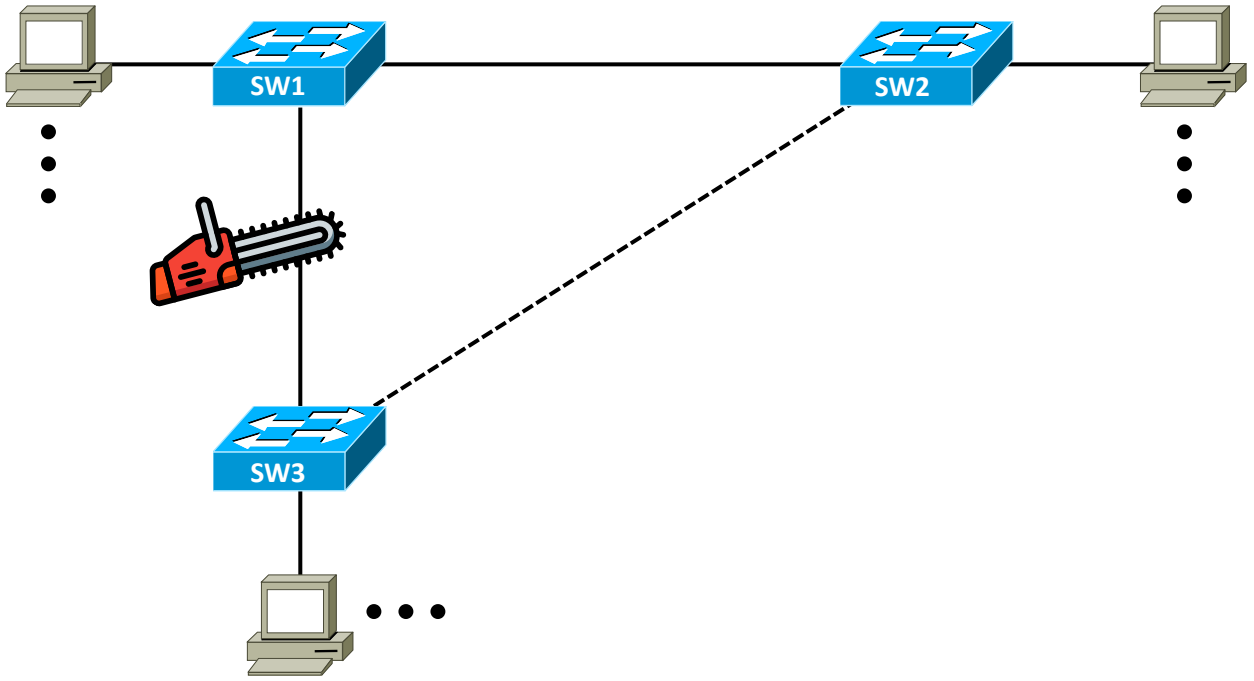
# Administração de Redes 2023/24

## Spanning Tree Protocol (STP) e variantes

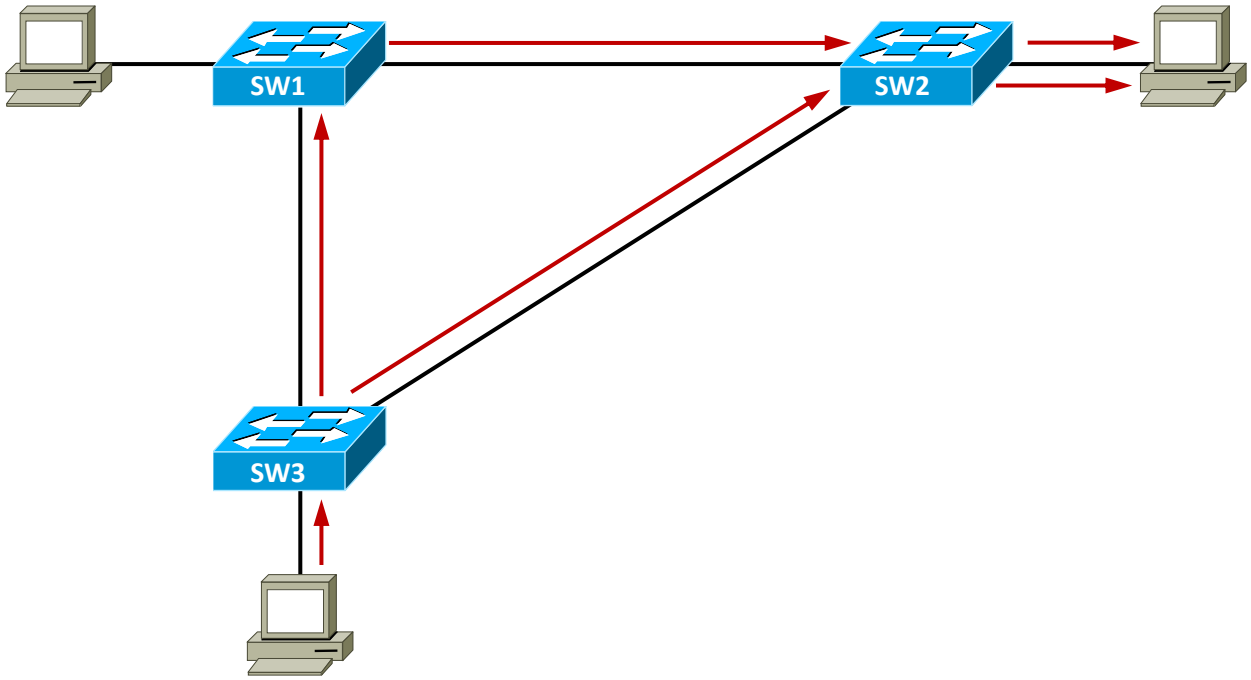
# Necessidade de ligações redundantes

- O que acontece se a ligação SW1↔SW3 for cortada?
- Máquinas no SW3 deixam de comunicar com o resto da rede…
- Ligações redundantes permitiriam tolerância a falhas



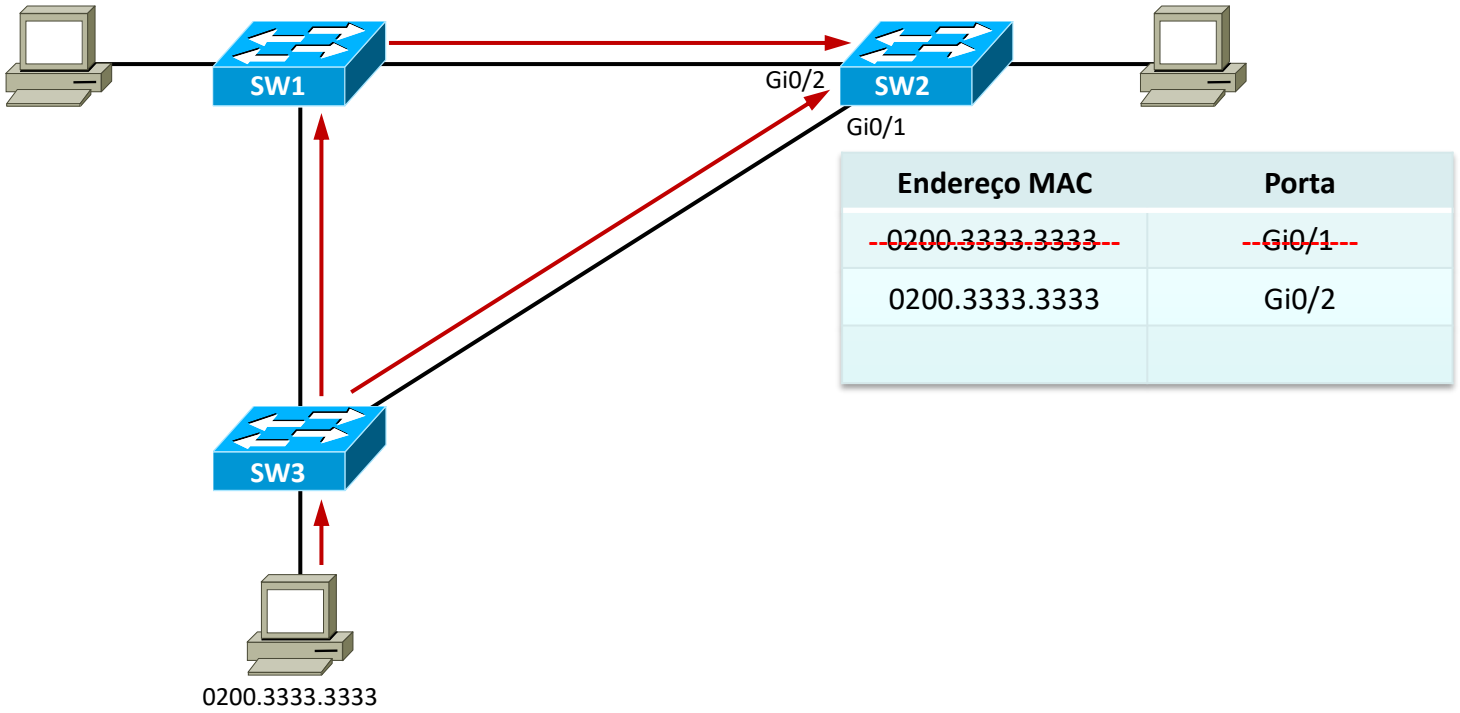
# Problemas com ligações redundantes

- Podem ser entregues várias cópias de um pacote



# Problemas com ligações redundantes

- Instabilidade das tabelas CAM*
  - Associam endereços MAC a portas físicas

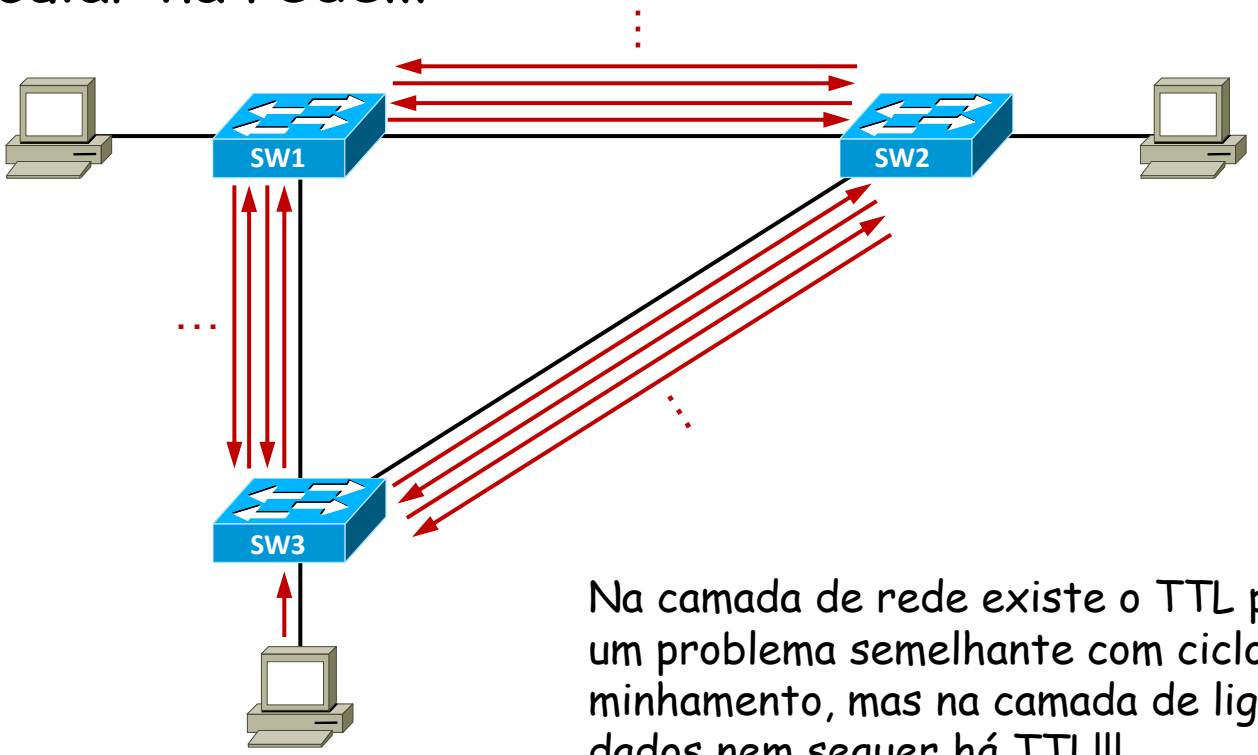



| Endereço MAC | Porta |
|---|---|
| ~~0200.3333.3333~~ | ~~Gi0/1~~ |
| 0200.3333.3333 | Gi0/2 |
| | |

* Content-Addressable Memory

# Problemas com ligações redundantes

- Tempestade de broadcast — tramas ficam para sempre a circular na rede!!!




Na camada de rede existe o TTL para aliviar um problema semelhante com ciclos de encaminhamento, mas na camada de ligação de dados nem sequer há TTL!!!

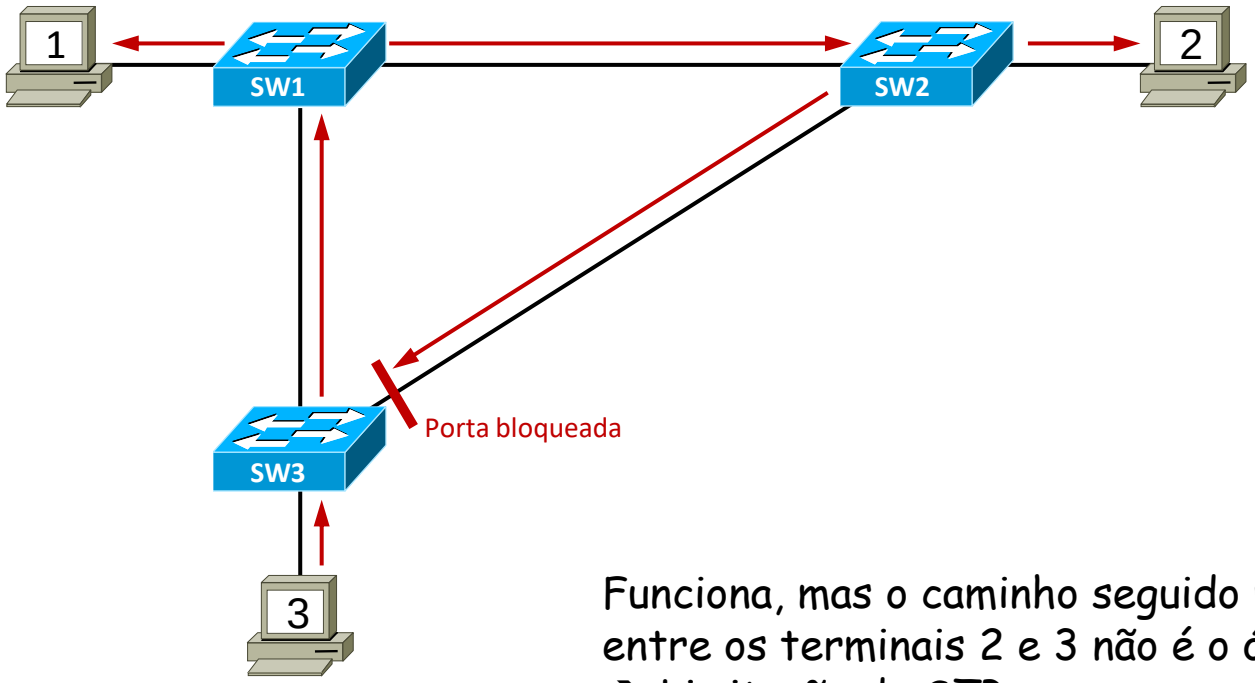
# Problemas com ligações redundantes

- Mesmo sem redundância intencional, podem criar-se ciclos por erro humano

Uma solução possível:

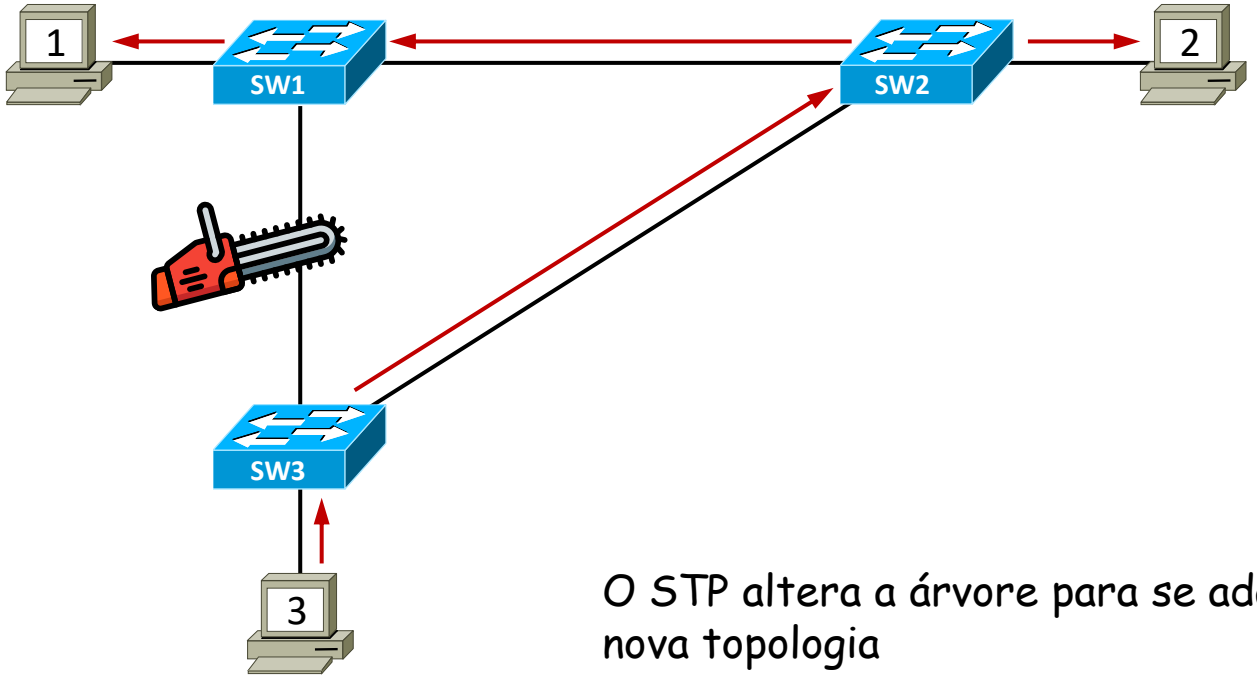
- Bloquear algumas portas para quebrar os ciclos
  - Topologias acíclicas não sofrem destes problemas
- Se falhar alguma ligação, rever o conjunto de portas bloqueadas
- Este mecanismo deve ser automático
  - Exige coordenação entre os comutadores

# Solução: todas as ligações funcionais




Funciona, mas o caminho seguido pelas tramas entre os terminais 2 e 3 não é o óptimo…
➔ Limitação do STP

# Solução: falha na ligação SW1↔SW3




O STP altera a árvore para se adaptar à nova topologia

# Spanning Tree Protocol (STP)

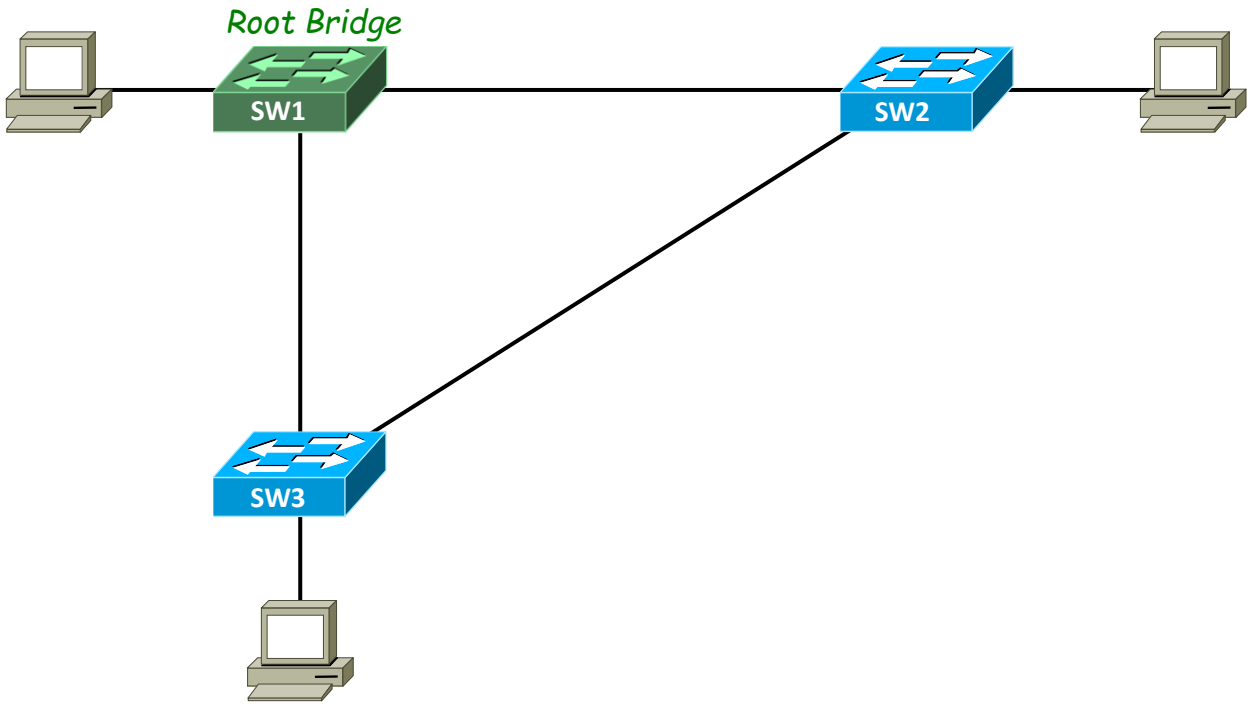
- Norma IEEE 802.1D
- Objectivo: bloquear algumas portas para que a topologia activa seja acíclica — uma *árvore de cobertura*
  - Não necessariamente mínima
  - Abrange todas as ligações (não apenas todos os nós)

- Troca de mensagens entre os comutadores ou *bridges**
  - BPDU (*Bridge Protocol Data Units*)
- Eleição de um que funciona como raiz da árvore
  - *Root bridge*
- Identificar portas necessárias para transmitir tramas
  - em direcção à *root bridge*
  - vindas da *root bridge*
- As portas que não forem necessárias são bloqueadas

*Nestes slides, os termos “comutador” e “*bridge*” usam-se indistintamente (do ponto de vista do STP são semelhantes)

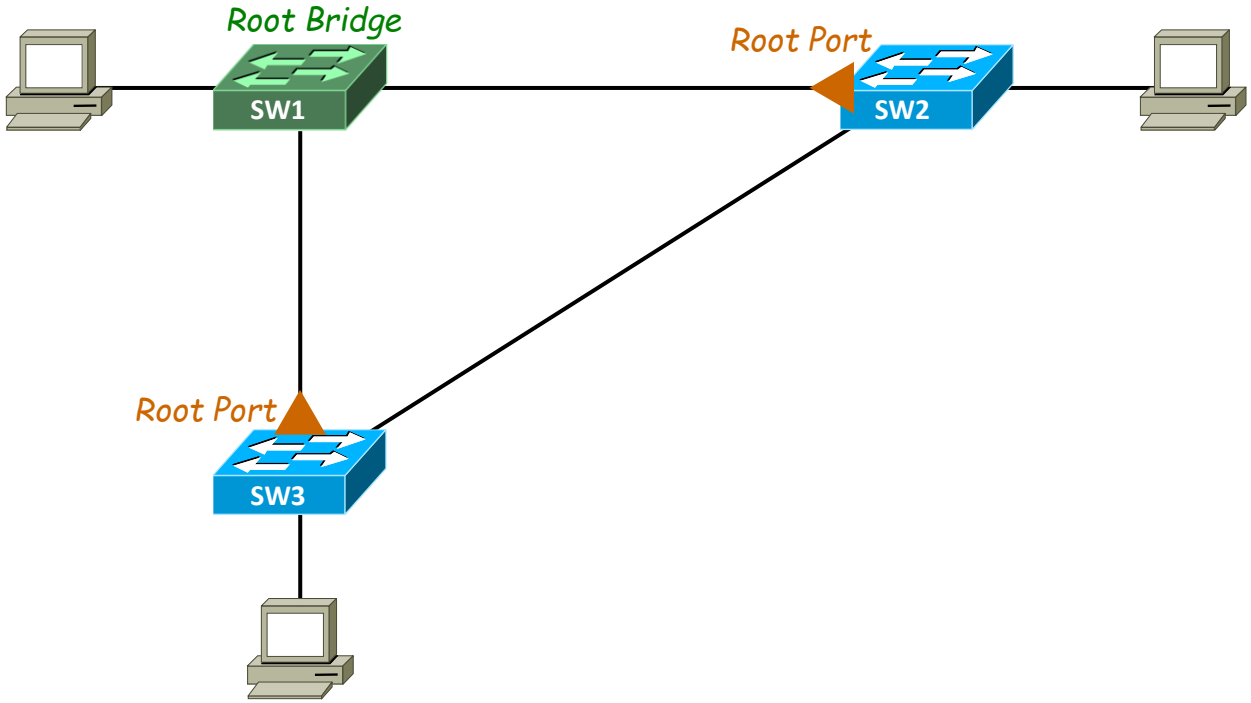
# Spanning Tree Algorithm (STA)

1. Eleger um comutador (ou *bridge*) como raiz: *root bridge*



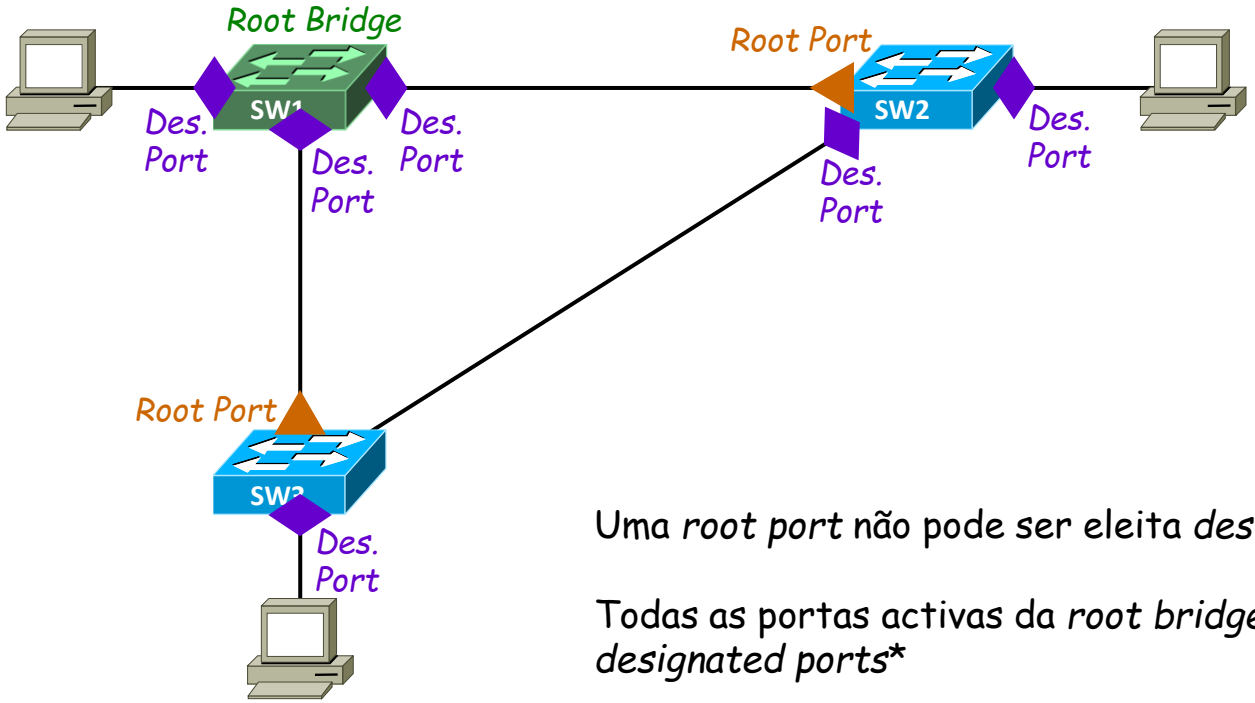
# Spanning Tree Algorithm (STA)

2. Em cada comutador (ou *bridge*), escolher uma porta para transmitir em direcção à *root bridge*: *root port*



# Spanning Tree Algorithm (STA)

3. Em cada ligação, escolher um comutador para transmitir nessa ligação: *designated port*

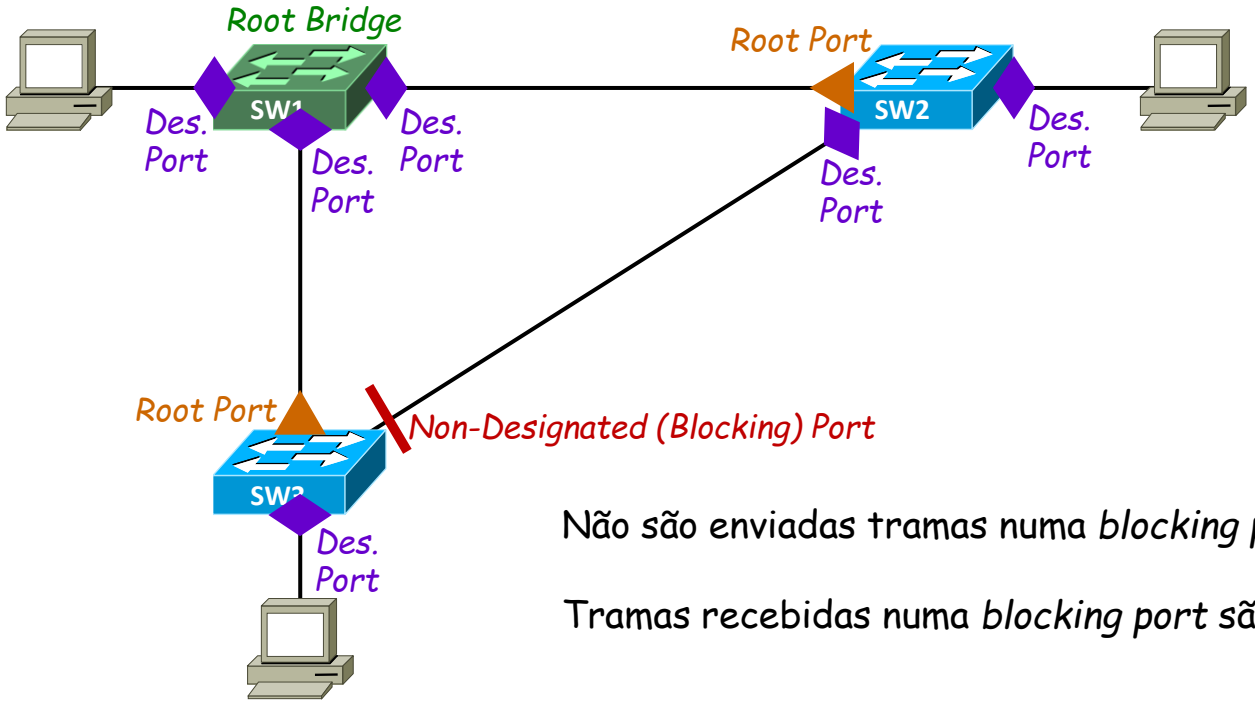



Uma *root port* não pode ser eleita *designated port*

Todas as portas activas da *root bridge* tornam-se *designated ports**

# Spanning Tree Algorithm (STA)

4. As restantes portas activas tornam-se *blocking ports**




Não são enviadas tramas numa *blocking port*

Tramas recebidas numa *blocking port* são descartadas

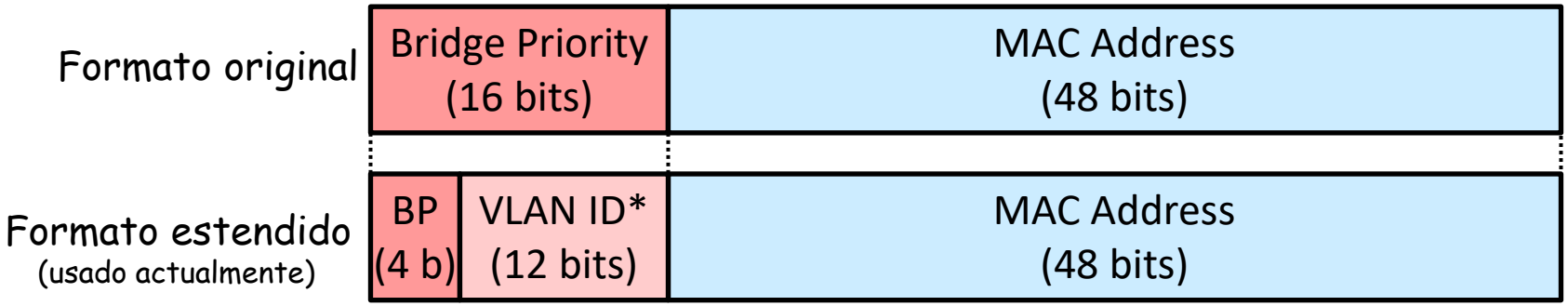
* Também chamadas *non-designated ports*

# Eleição da *root bridge*

- Todos os comutadores participam na eleição
- Cada comutador tem um *Bridge ID (BID)*
- Vai tornar-se *Root* o que tiver o BID mais baixo
- Inicialmente, cada comutador acha que é a *Root*
  - Gera uma ***Configuration BPDU**** a cada 2s, que contém (entre outros)
    - *Bridge ID* do próprio comutador/*bridge*
    - *Root ID* (BID do que considera ser a *Root*)
    - Custo do caminho até à (que ele considera ser a) *Root*
    - Alguns temporizadores (*Hello timer, MaxAge timer, Forward timer*)
- Se descobrir que há um melhor candidato a *Root*, deixa de gerar BPDU, mas reenvia as que recebe na *Root Port* (actualizando-as)
  - *A Root* continua a gerar BPDU a cada 2s

* Também chamada *Hello BPDU* por serem geradas a cada *Hello time*

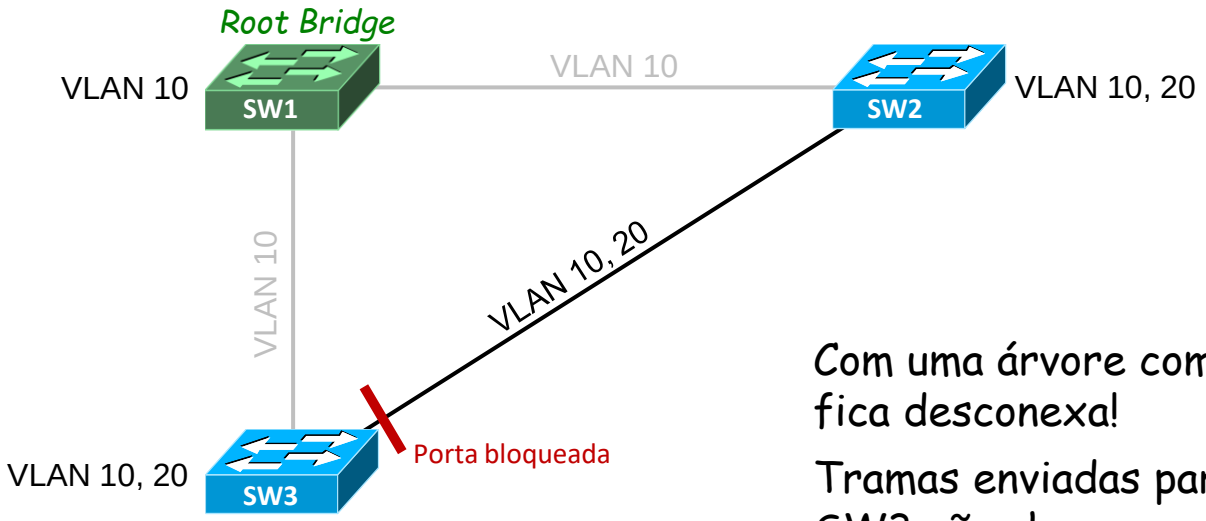
# Eleição da *root bridge*




Formato do *Bridge ID*

- *Bridge Priority* entre 0 e 65535
- No formato estendido inclui um *System ID Extension* com o número da VLAN
  - Possibilita uma instância de STP por VLAN
  - O valor de prioridade tem que ser múltiplo de 4096 ($2^{12}$)
    - *Default* = 32768
  - Exemplo: Prioridade = 32768 + VLAN = 1 ➔ Bridge Priority = 32769
- O comutador com BID mais baixo é eleito *Root Bridge*

# Porque é necessária uma instância de STP por VLAN?



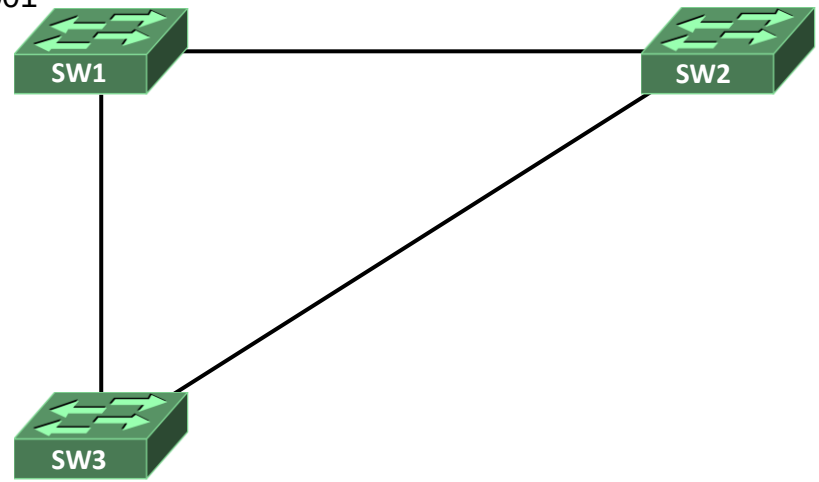

Com uma árvore comum, a VLAN 20 fica desconexa!

Tramas enviadas para a VLAN 20 no SW3 não chegam ao SW2, e vice-versa

# Eleição da *root bridge*

- Inicialmente, cada comutador acha que é *Root Bridge*

My BID: 32769:0200.0001.0001
Root BID: 32769:0200.0001.0001
Root Cost: 0

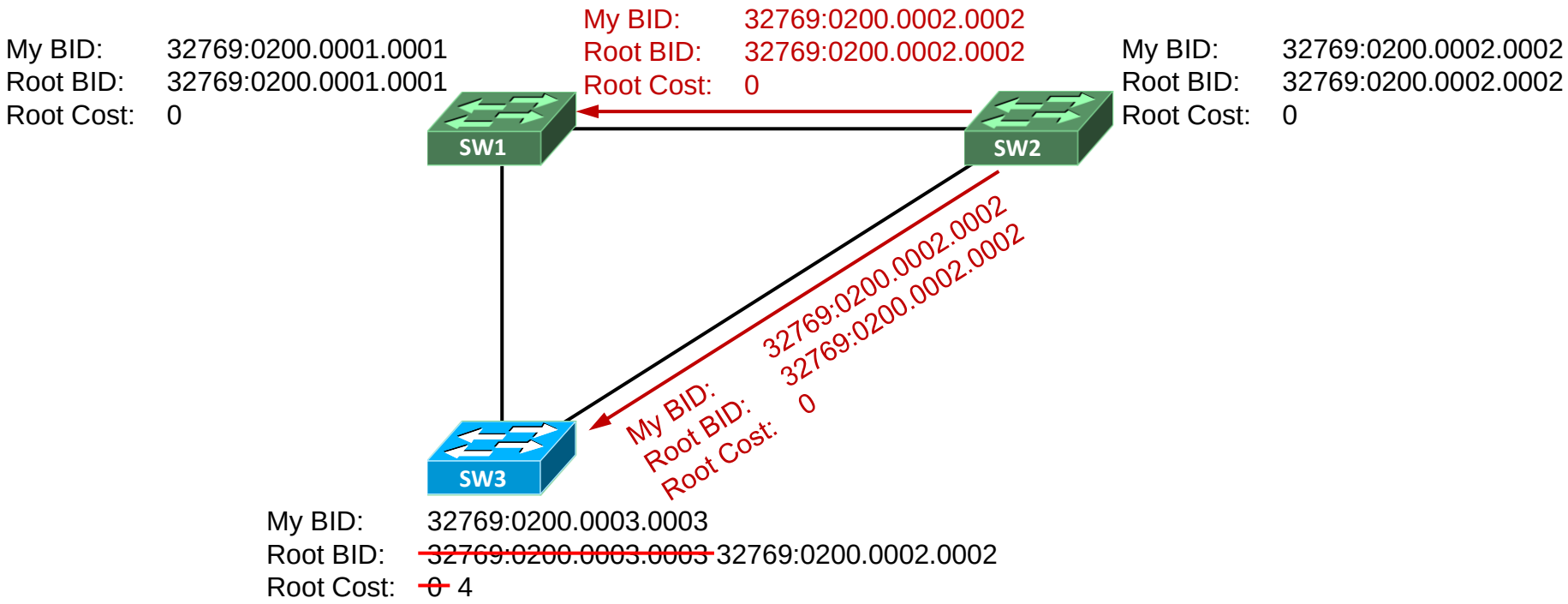



My BID: 32769:0200.0002.0002
Root BID: 32769:0200.0002.0002
Root Cost: 0

My BID: 32769:0200.0003.0003
Root BID: 32769:0200.0003.0003
Root Cost: 0

# Eleição da *root bridge*

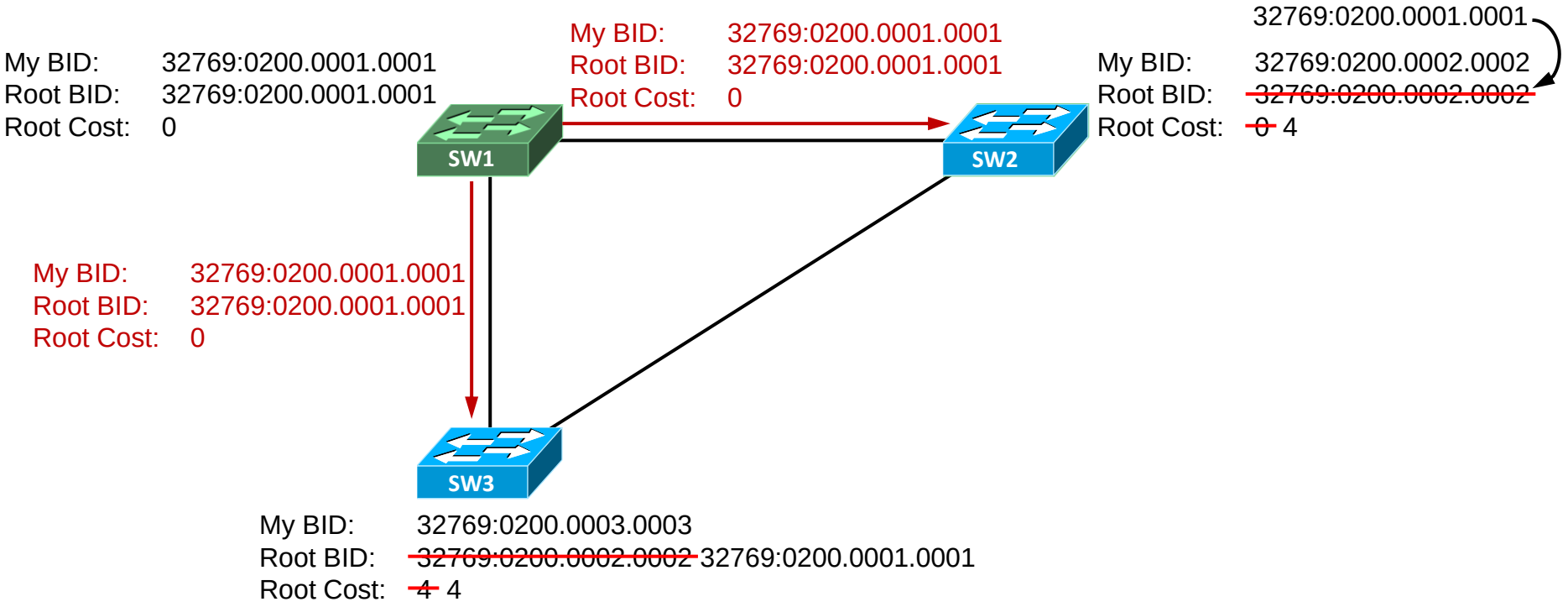
- Agora SW3 acha que é SW2 a *Root*




*O Root Cost* obtém-se somando o custo da interface em que a BPDU foi recebida ao indicado na propria BPDU. *O* custo *default* para uma interface *Gigabit Ethernet é* 4.

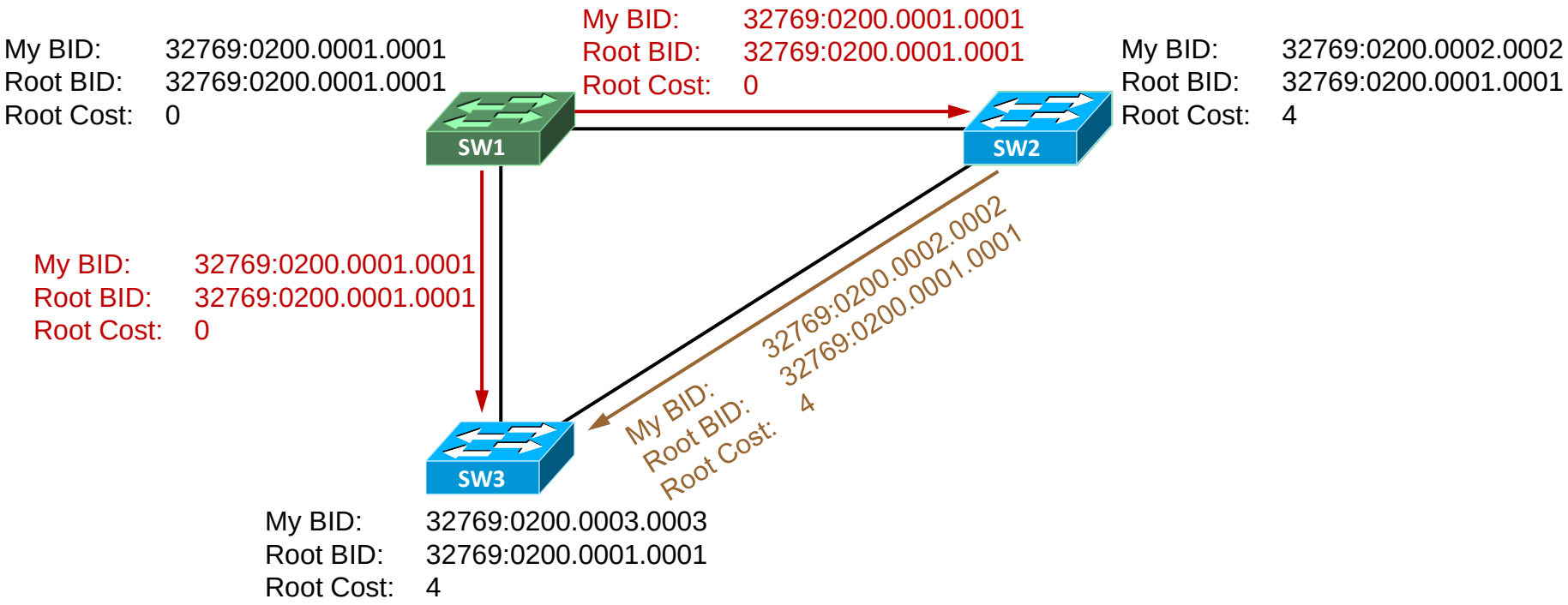
# Eleição da *root bridge*

- Agora todos sabem que é SW1 a *Root*



# Eleição da *root bridge*

- Em regime permanente, a *Root* gera *Configuration BPDU* a cada 2s
- Os outros reenviam nas *Designated Ports* as *Configuration BPDU* que recebem na *Root Port* (actualizadas)




- Se a *Root Bridge* falhar, o processo reinicia-se

# Configurar a *Bridge Priority*

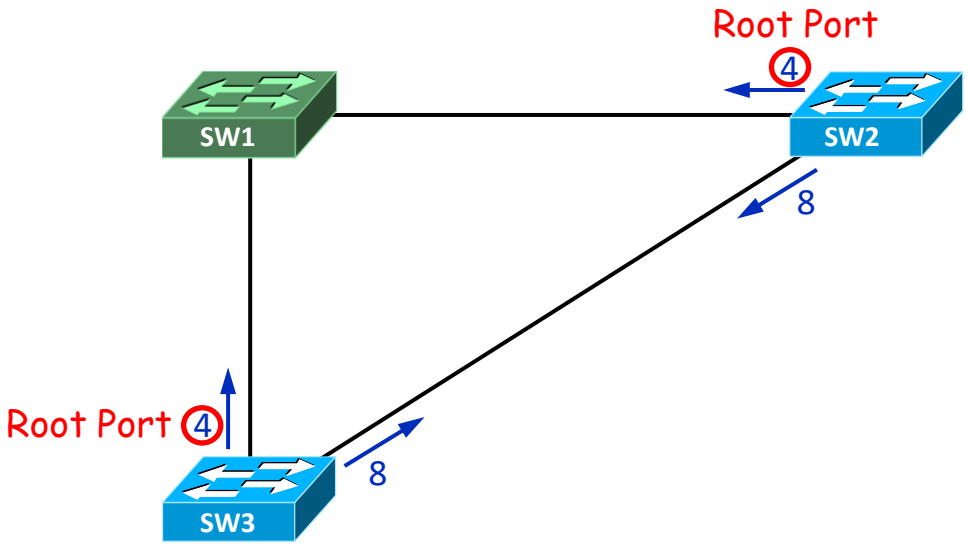
- **Método 1**: configuração directa da prioridade para uma VLAN ou gama de VLAN
  `spanning-tree vlan` *`range`* `priority` *`value`*
  - O valor deve ser múltiplo de 4096

- **Método 2**: indicação de que o comutador que deve tornar-se *Root Bridge*, assumindo a prioridade numericamente mais baixa na rede
  `spanning-tree vlan` *`range`* `root primary`

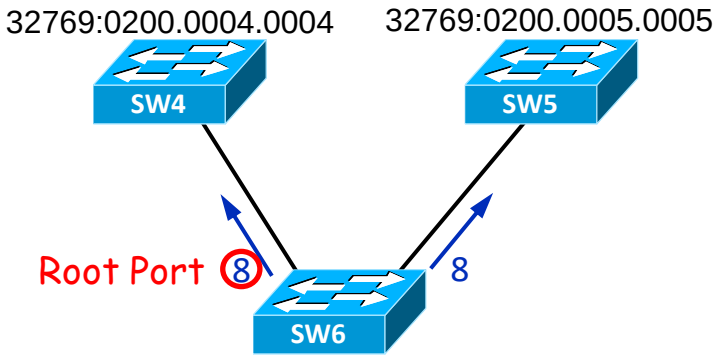
  Também é possível designar um *backup* para o caso do anterior falhar
  `spanning-tree vlan` *`range`* `root secondary`
  - Valor imediatamente abaixo do *default* (assume que todos os outros têm o *default*)

# Selecção da *root port* em cada comutador

- Durante o processo de eleição da *Root Bridge* é também determinado o *Root Path Cost*
- Em cada comutador excepto a *Root Bridge*, a porta com menor custo é escolhida como *Root Port*
  - Responsável por enviar tramas em direção à *Root Bridge*



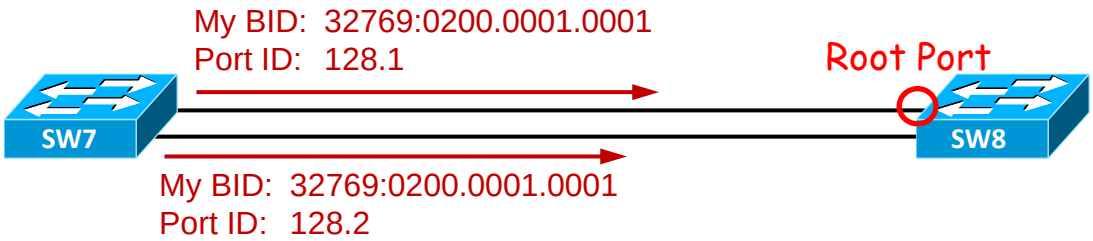

Em caso de empate é escolhida a que liga ao vizinho com BID menor



# Selecção da *root port* em cada comutador

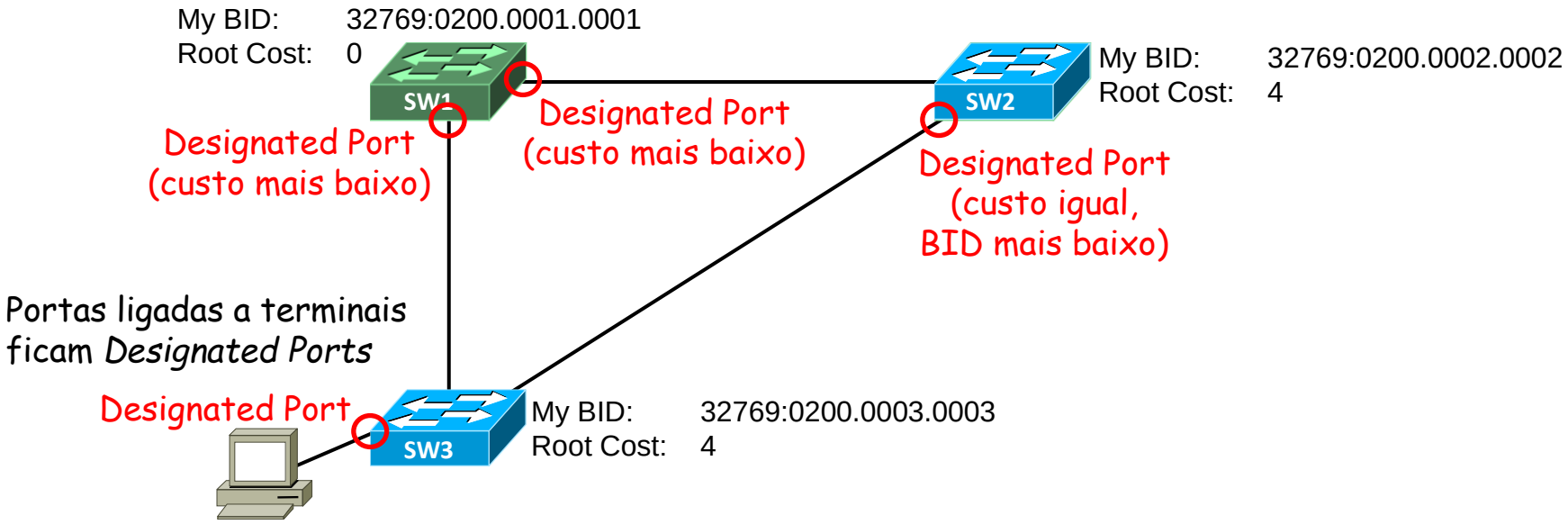
- E se houver mais de uma ligação entre um par de comutadores?
  - *Root Path Cost* e BID do vizinho idênticos
- As BPDU incluem um *Port ID* constituído por
  - *Port Priority* entre 0 e 240 em múltiplos de 16 ($2^4$)
    - *Default* = 128
  - Número de porta


- Fica *Root Port* a que recebeu a BPDU com *Port ID* mais baixo
  - *Port ID* da porta do vizinho, não da sua!



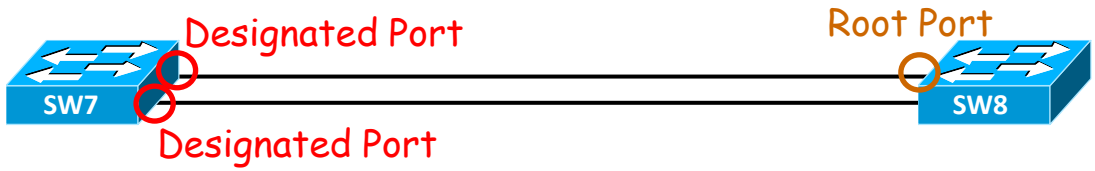
# Eleição da *designated port* em cada segmento

- Entre as restantes portas é preciso eleger uma *Designated Port* por segmento LAN (domínio de colisão)
  - DP é responsável por reenviar tramas para esse segmento
- É escolhida a que anuncia um *Root Path Cost* mais baixo
  - Ou seja, a que pertence ao comutador com esse custo mais baixo
  - Por isso, uma *Root Port* nunca pode ser *Designated Port*
- Em caso de empate, é escolhida a do comutador com BID mais baixo

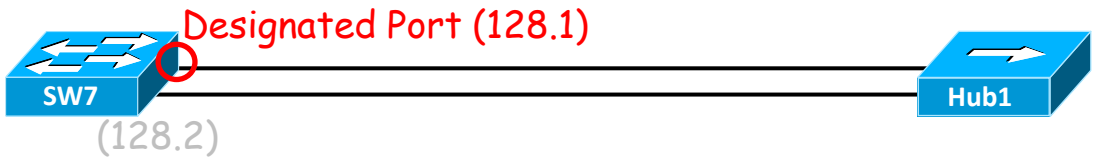


# Eleição da *designated port* em cada segmento

- Se houver várias ligações entre um par de comutadores, as portas respectivas no que está a montante ficam todas *Designated Port*
  - No que está a jusante, no máximo uma pode ficar *Root Port*




- Se houver várias portas ligadas a um segmento (e.g., *Hub*), só fica *Designated Port* a que tiver o *Port ID* mais baixo
  - Improvável encontrar este caso em redes actuais




- Todas as restantes portas ficam *Blocking Ports*

# *Root Path Cost, Port Path Cost, Designated Path Cost*

- Cada ligação tem um *Designated Path Cost* que é o custo anunciado pelo *Designated Port* dessa ligação
  - É o *Root Path Cost* do comutador a que pertence essa porta (*Designated Bridge*)
- Cada porta tem um *Port Path Cost* que é o custo isolado dessa porta
- Cada comutador tem um *Root Path Cost* que é
  - Zero na *Root Bridge*
  - A soma do *Designated Path Cost* com o *Port Path Cost* da *Root Port* nos outros comutadores

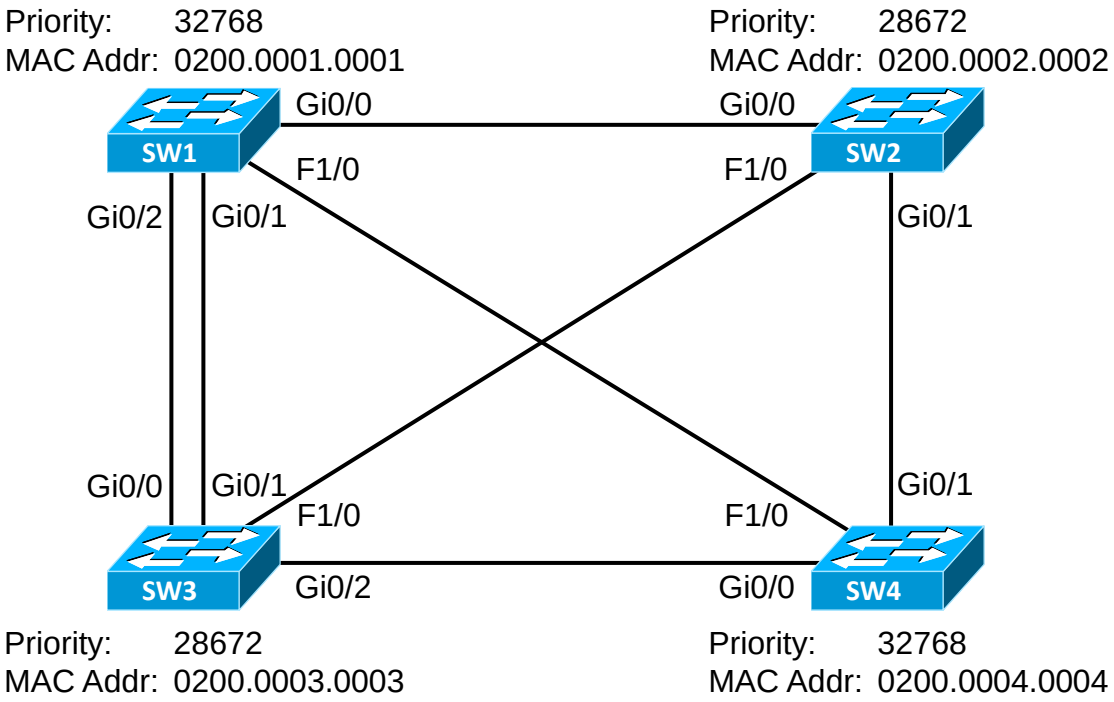
# *Port Path Cost*: valores recomendados

| Bit Rate | Custo IEEE antigo | Custo IEEE recente |
|---|---|---|
| 10 Mb/s | 100 | 2 000 000 |
| 100 Mb/s | 19 | 200 000 |
| 1 Gb/s | 4 | 20 000 |
| 10 Gb/s | 2 | 2 000 |
| 100 Gb/s | — | 200 |
| 1 Tb/s | — | 20 |

- O custo é atribuído em função do *bit rate* efectivo, não o nominal
  - Se uma interface *Gigabit Ethernet* estiver ligada a 100 Mb/s, tem custo 19
- Custos IEEE antigos são o *default* no Cisco IOS
- Podem alterar-se globalmente para os recentes com o commando `spanning-tree pathcost method long`
- Pode alterar-se o custo de uma interface específica com `spanning-tree cost` *cost*

# Exercício

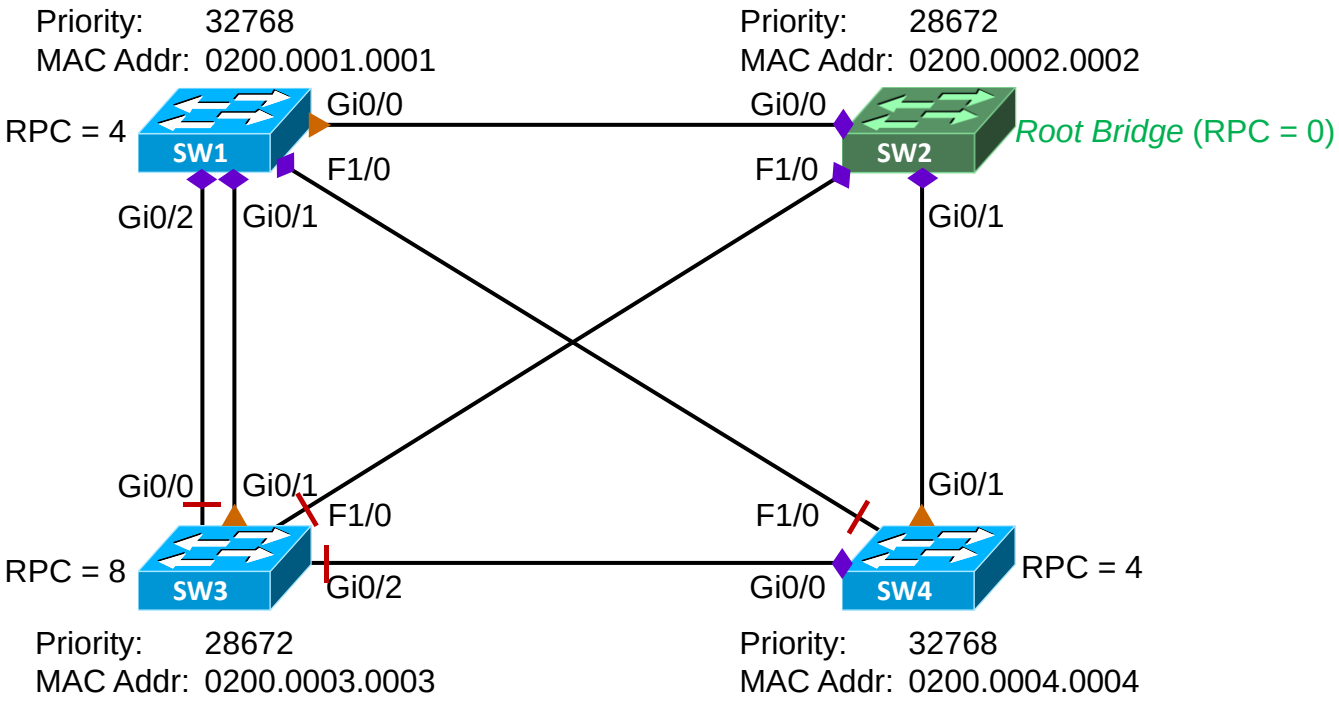
- Que comutador será *Root* Bridge?
- Que portas irão ficar RP, DP e BP?
- Qual será o *Root Path Cost* de cada comutador?




NOTA: Este é um exercício que deve conseguir resolver sozinho, sem consultar a solução.

# Exercício – Solução

▲ *Root Port*
◆ *Designated Port*
— *Blocking Port*



# Funcionamento com a rede estável

- *A Root Bridge* gera uma *Configuration BPDU* a cada 2 segundos (*default*) e envia-a nas suas *Designated Ports*
- Quando um comutador recebe essa BPDU na sua *Root Port*, reenvia-a nas suas *Designated Ports*
  - Após modificar alguns campos: *Bridge ID*, *Root Path Cost*, …
- Consequentemente, a *Configuration BPDU* vai ser enviada uma vez em cada ligação activa na topologia
- Este processo repete-se até haver alguma alteração à topologia
- A recepção periódica das BPDU indica que o caminho até à *Root Bridge* continua funcional
- Se o comutador deixar de as receber ou começar a recebê-las com parâmetros diferentes é porque algo mudou
  - O comutador reage iniciando um processo para alterar a topologia da *spanning tree*

# Estados das interfaces (portas)

- *Blocking*
  - A porta não é RP nem DP, por isso não envia tramas e ignora as recebidas
  - Processa BPDU recebidas
- *Listening*
  - A porta tornou-se RP ou DP
  - Fica neste estado 15s para garantir que a topologia estabilizou
  - Não recebe nem envia tramas
  - Processa BPDU recebidas, (re)envia BPDU se for DP
- *Learning*
  - A porta é RP ou DP e transitou do estado Listening
  - Fica neste estado durante 15s
  - Descarta tramas recebidas, mas usa os endereços MAC de origem dessas tramas para preencher a tabela CAM
  - Não envia tramas
  - Processa BPDU recebidas, (re)envia BPDU se for DP

# Estados das interfaces (portas)

- *Forwarding*
  - É o estado final de uma RP ou DP
  - Recebe e envia tramas
  - Processa BPDU recebidas, (re)envia BPDU se for DP
- *Disabled*
  - Porta administrativamente desactivada ou com erro

| Acção | *Blocking* | *Listening* | *Learning* | *Forwarding* | *Disabled* |
|---|---|---|---|---|---|
| Recebe e processa BPDU | ✓ | ✓ | ✓ | ✓ | ✗ |
| (Re)envia BPDU | ✗ | Se for DP | Se for DP | Se for DP | ✗ |
| Reenvia noutras portas tramas que recebeu nessa | ✗ | ✗ | ✗ | ✓ | ✗ |
| Reenvia nessa porta tramas que recebeu noutras | ✗ | ✗ | ✗ | ✓ | ✗ |
| Aprende endereços MAC nessa porta | ✗ | ✗ | ✓ | ✓ | ✗ |

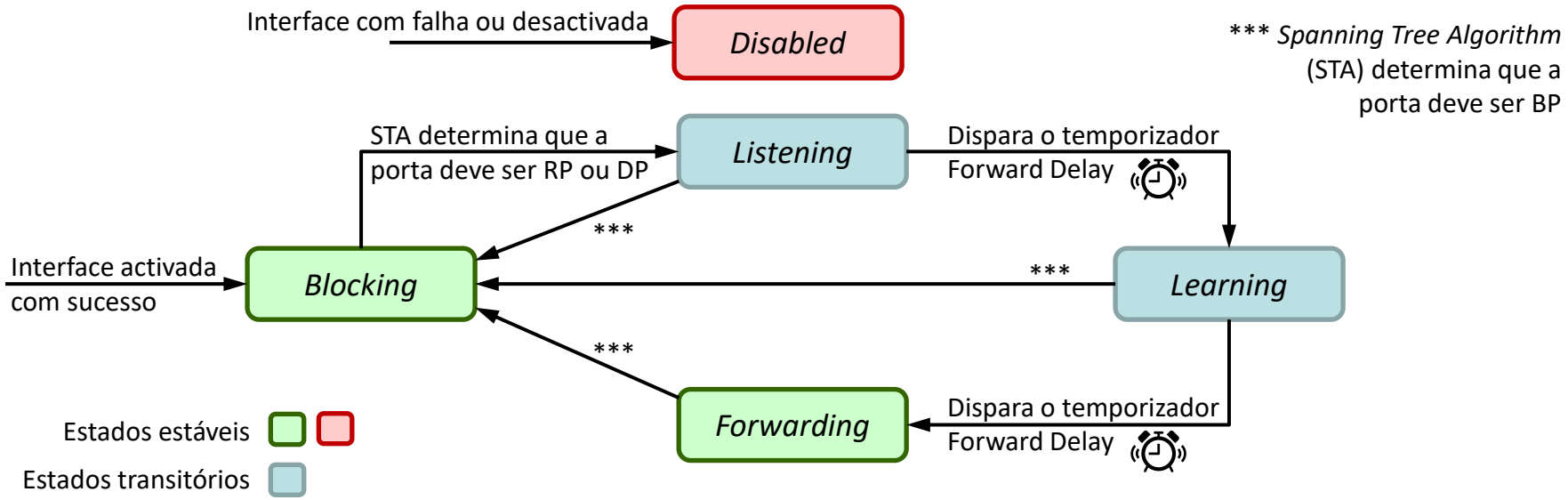
# Temporizadores

- O STP usa vários temporizadores
- O valor dos temporizadores é determinado pela *Root Bridge* e aceite pelos outros

| Temporizador | Valor *default* | Descrição |
|---|---|---|
| Hello | 2 s | Periodicidade com que a *Root Bridge* envia *Config. BPDUs* |
| MaxAge | 20 s | Quanto tempo o comutador deve esperar após receber a última BPDU até iniciar alterações à topologia |
| Forward Delay | 15 s | Tempo que uma porta deve permanecer em cada estado transitório *(Listening* e *Learning)* quando passa do estado *Blocking* para o *Forwarding* |

# Mudanças de estado

- O STP usa os conceitos de função (*role*) e estado de uma porta
  - Funções: *Root Port*, *Designated Port*, *Blocking Port*
  - Estados: *Blocking*, *Listening*, *Learning*, *Forwarding*, *Disabled*
- *Blocking Ports* passam imediatamente para o estado *Blocking*
- *Root Ports* e *Designated Ports* tendem para o estado *Forwarding*, mas passam pelos estados temporários *Listening* e *Learning*



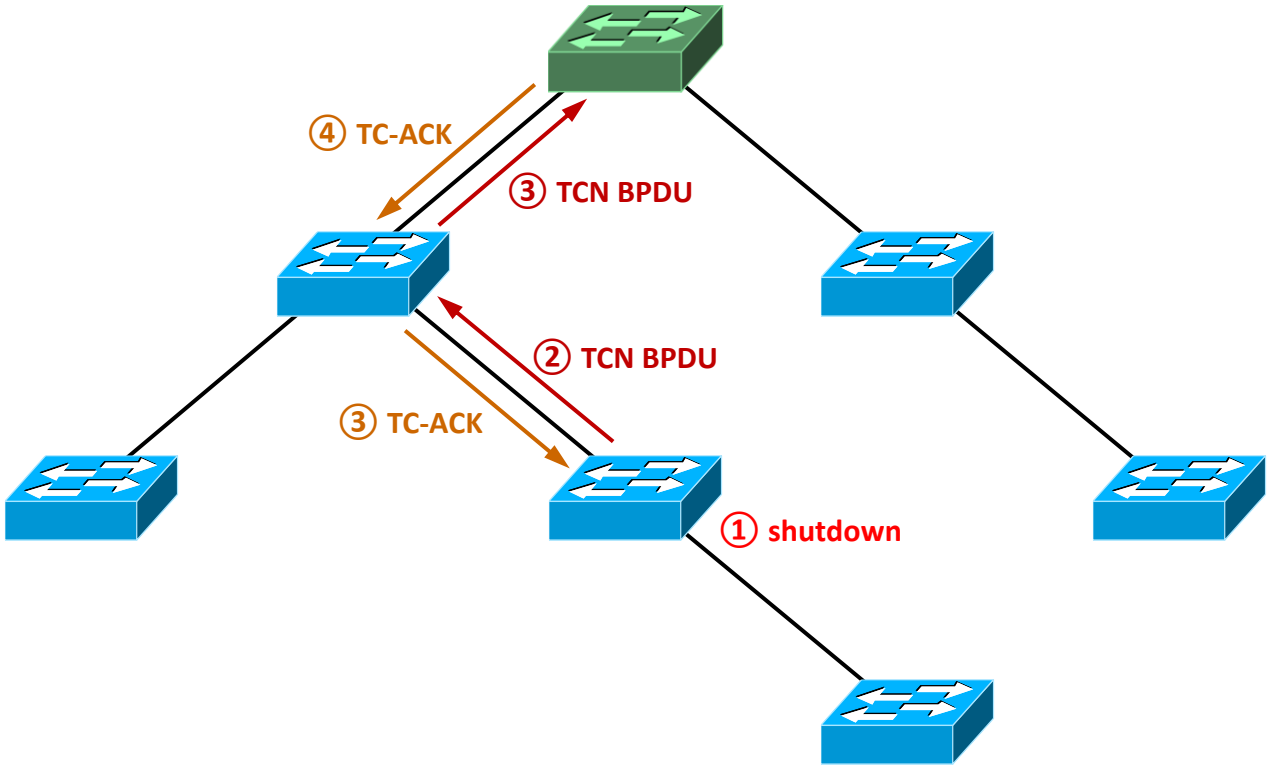
# Mudanças de estado — tempos

- Quando uma porta é ligada ou activada demora 30s até ficar operacional (estado *Forwarding*)
  - Tempo de passagem pelos estados *Listening* (15s) e *Learning* (15s)
  - Pode criar problemas a terminais configurados por DHCP (timeout antes de a interface ficar operacional)
- Quando uma RP falha pode demorar ainda mais até outra porta ficar operacional com essa função
  - Se a falha ou perda de *link* for detectada (falha directa), também demora 30s
  - Caso contrário (falha indirecta), demora 50s até que
    - a última BPDU recebida na RP actual expire (MaxAge = 20s)
    - a nova RP atravesse os estados *Listening* (15s) e *Learning* (15s)
- Estes tempos tão longos são uma limitação importante do STP

# Topology Change Notification

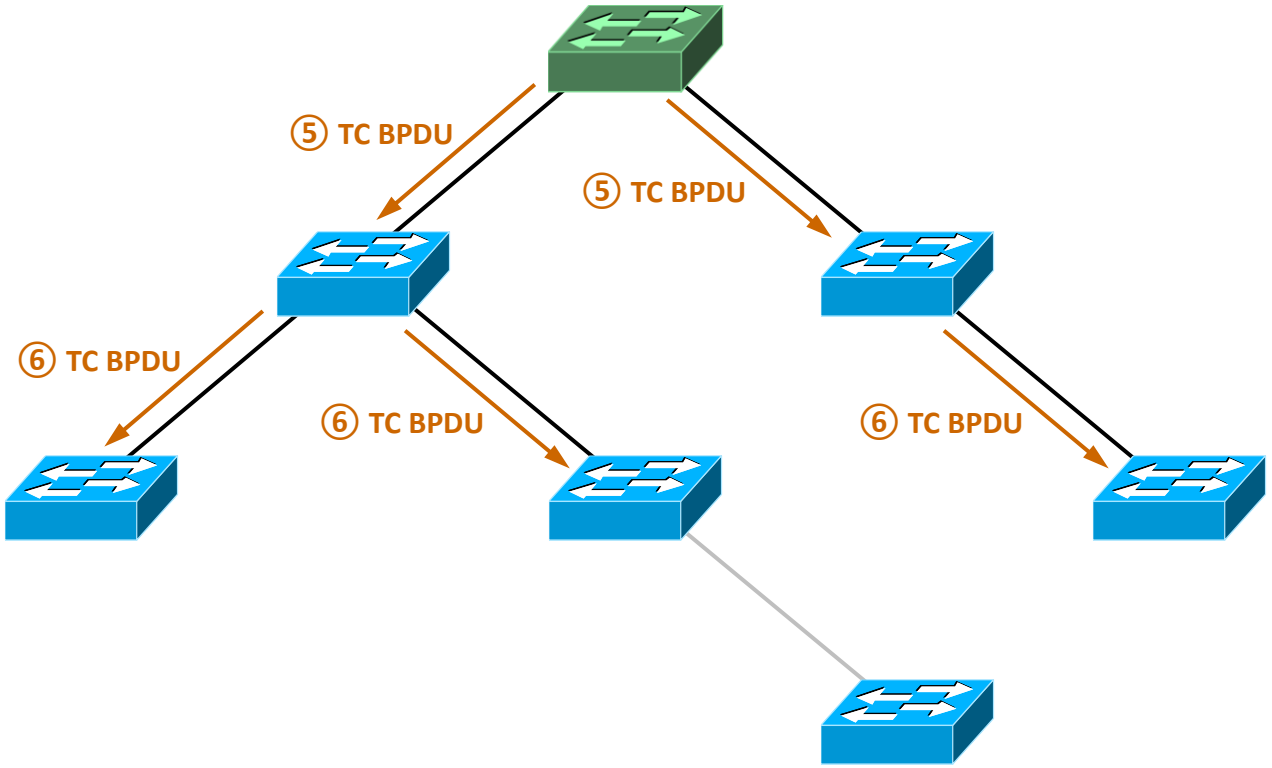
- Quando há alterações à topologia, é natural que haja alterações às associações <Endereço MAC, Interface> nas tabelas *CAM*
- Nessa situação, o prazo normal para essas entradas (300s) é excessivo
- Para reduzir a disrupção nas comunicações neste caso, o STP tem um mecanismo de notificação de alterações à topologia
- Quando um comutador detecta uma alteração, envia na sua RP uma *Topology Change Notification* BPDU
- O comutador a montante
  - Confirma a recepção com um TC-ACK (*Configuration BPDU* com flag *Topology Change Acknowledgement*)
  - Reenvia-a na sua RP
- Quando a *Root Bridge* recebe a TCN BPDU, passa a enviar as *Configuration BPDU* com a flag *Topology Change* activa durante 35s
  - MaxAge + Forward Delay
- Quando um comutador recebe essa BPDU, reduz o tempo de vida das entradas na tabela *CAM* a 15s (Forward Delay)

# Topology Change Notification




Uma **mudança de topologia** ocorre quando um comutador passa uma porta para o estado *Forwarding* ou passa uma porta do estado *Forwarding* ou *Learning* para o estado *Blocking* ou *Disabled*.

# Topology Change Notification




Os comutadores expiram as entradas na tabela CAM passados 15s de inactividade

# Obter informação

```
SW4#show spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    32769
              Address     0200.0001.0001
              Cost        8
              Port        2 (GigabitEthernet0/1)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32769  (priority 32768 sys-id-ext 1)
              Address     0200.0004.0004
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi0/0               Altn BLK 4         160.1    Shr
Gi0/1               Root FWD 4         128.2    Shr
Gi0/2               Desg FWD 4         128.3    Shr
Gi0/3               Desg LIS 4         128.4    Shr
```

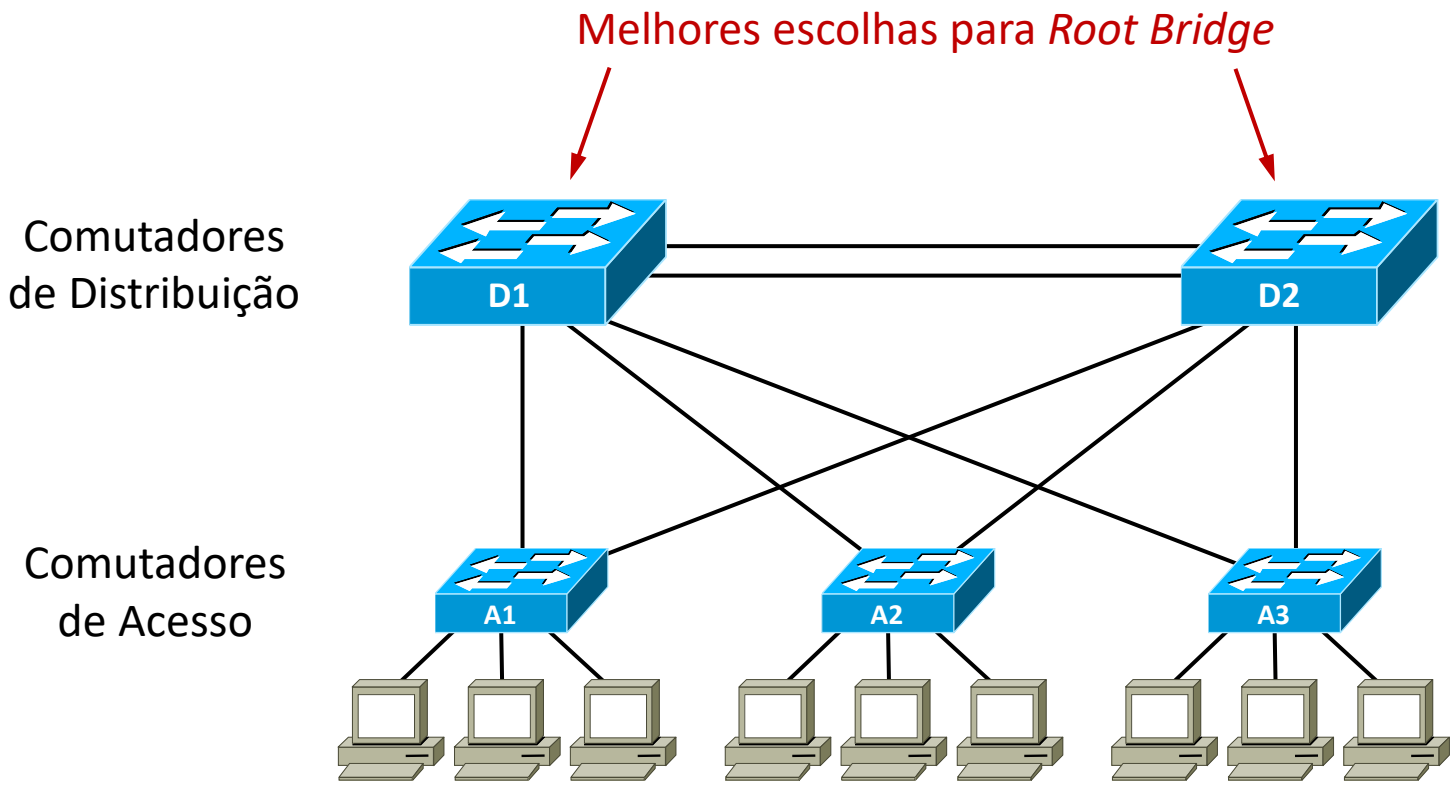
Variante do protocolo a correr (802.1D neste caso)

Info. sobre *Root Bridge*, *Root Path Cost*, *Root Port*

Info. sobre o próprio comutador

Informação sobre as interfaces

# Hierarquia de comutadores



# PortFast

- Com STP, uma interface demora 30s após a detecção de *link* até ficar *Forwarding*
  - Tempo de passar pelos estados *Listening* e *Learning*
  - Pode causar problemas a terminais configurados por DHCP (*timeout* antes de obter endereço)
- Configurando uma interface como *PortFast*, ela transita imediatamente de *Blocking* para *Forwarding*
  - Extensão proprietária da Cisco
- Não são geradas TCN BPDU quando se ligam ou desligam portas *PortFast*
- Usa-se em portas onde vão ligar-se terminais
  - Ligar a porta a outro comutador pode criar um ciclo ➔ CUIDADO!!!
  - Se receber uma BPDU numa porta *PortFast*, o *PortFast* é desactivado para ela
- Configurar uma interface como *PortFast*:
  `spanning-tree portfast` (na configuração da interface)
- Activar *PortFast* por *default* em todas as interfaces configuradas em modo acesso:
  `spanning-tree portfast default`

# BPDU Guard

- Numa porta onde se liga um terminal não são recebidas BPDU
- Activando *BPDU Guard* (extensão proprietária da Cisco) nessa interface, é gerado um erro caso seja recebida uma BPDU
  - Só pode acontecer ligando-a a outro comutador
- Se for recebida uma BPDU, é gerado um erro e a interface é desactivada
  - Fica em estado `err-disabled`
  - Ver portas nesse estado: `show interfaces status err-disabled`
- Para voltar a activá-la, o administrador deve fazer `shutdown` / `no shutdown`
- É boa prática configurar interfaces onde ligam terminais com *PortFast* e *BPDU Guard*
- Activar *BPDU Guard* por *default* para todas as portas *PortFast*: `spanning-tree portfast bpduguard default`

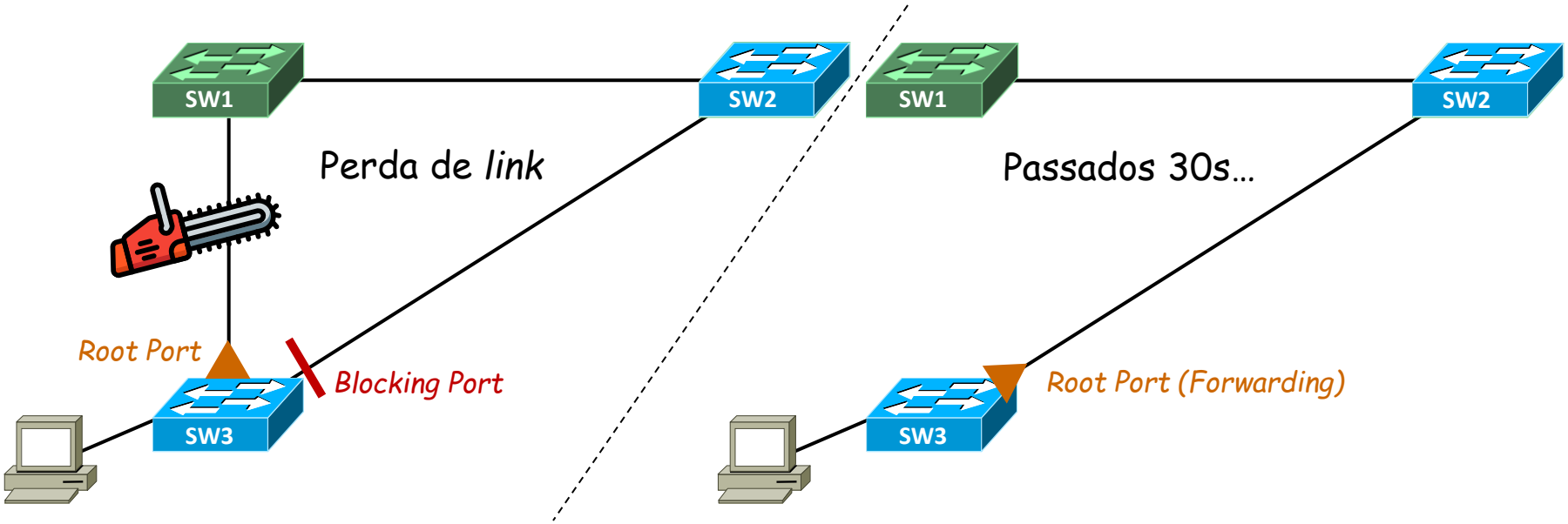
# UplinkFast

- Permite recuperar rapidamente de perda de ligação na *Root Port*
- Extensão proprietária da Cisco
- Só deve ser configurada em comutadores da camada de acesso (com terminais directamente ligados) e com *uplinks* redundantes
  - Só funciona se o comutador tiver *Blocking Ports*

- Quando o comutador detecta perda de ligação na *Root Port*
  - Escolhe uma porta alternativa para *Root Port*
  - Passa-a imediatamente para o estado *Forwarding*
    - Sem passar pelos estados *Listening* e *Learning*
  - Envia tramas *dummy* (sem dados) tendo como endereço de origem o MAC de cada terminal ligado nesse comutador
    - Endereço de destino *multicast* para chegarem a toda a rede
    - Objectivo: actualização rápida das tabelas CAM nos outros comutadores

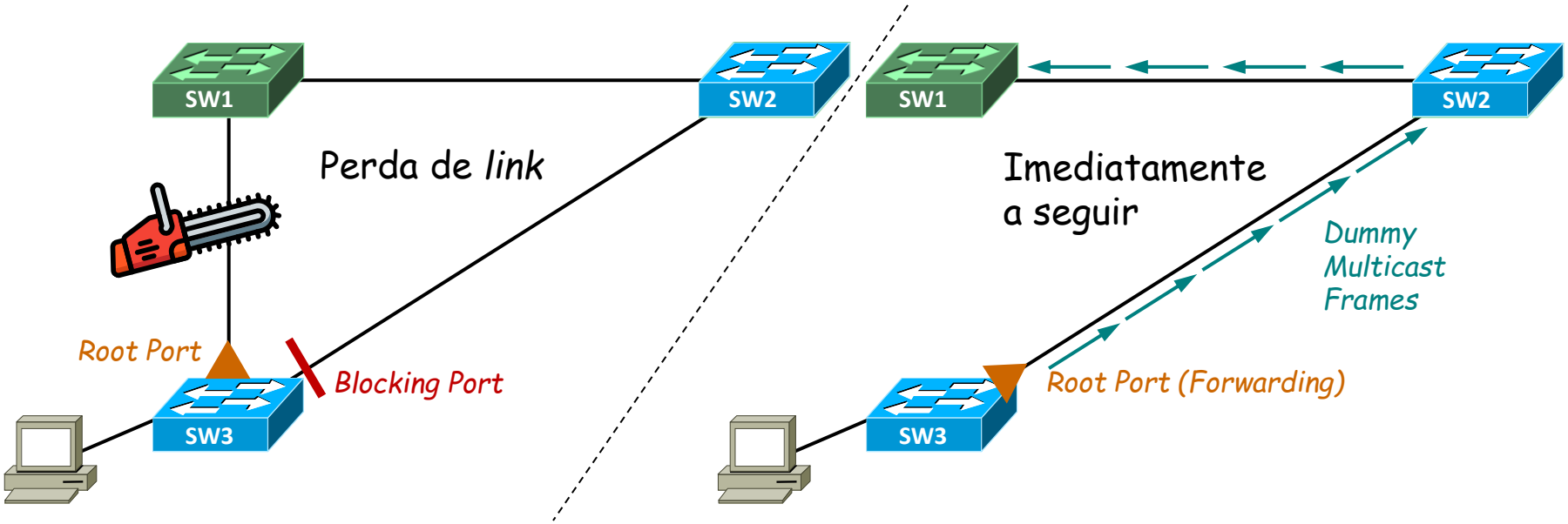
- Activar *UplinkFast:*
  `spanning-tree uplinkfast`

# Sem UplinkFast




1. SW3 detecta perda de ligação no *Root Port*
2. Escolhe uma nova porta para *Root Port*. Essa porta passa
   1. 15s no estado *Listening*
   2. 15s no estado *Learning*
3. Finalmente converge (passados 30s!)
   - Mas os outros comutadores ainda têm entradas desactualizadas na tabela CAM

# Com UplinkFast




1. SW3 detecta perda de ligação no *Root Port*
2. Escolhe uma nova porta para *Root Port* e passa-a para o estado *Forwarding*
   - A reconvergência é quase imediata
3. Envia tramas multicast *dummy* para actualizar as tabelas CAM dos outros comutadores
   - NOTA: nem todas as implementações enviam estas tramas

# Notas sobre o *UplinkFast*

- Ao activar o *UplinkFast* são automaticamente feitas algumas alterações com implicações na árvore STP
  - A *Bridge Priority* é alterada para 49152 para evitar tornar-se *Root*
    - A menos que esteja explicitamente configurado outro valor
  - Os custos das interfaces são aumentados em 3000 para evitar que o comutador vizinho escolha a porta ligada a essa como *Root Port*
    - O *UplinkFast* é para usar em comutadores da camada de acesso

- Se o *uplink* anterior voltar a ter ligação, não volta a tornar-se RP imediatamente
  - Espera 35s (2×*ForwardDelay* + 5s) para dar tempo à porta do outro comutador para passar a *Forwarding*
  - Idem se se adicionar um novo *uplink* melhor

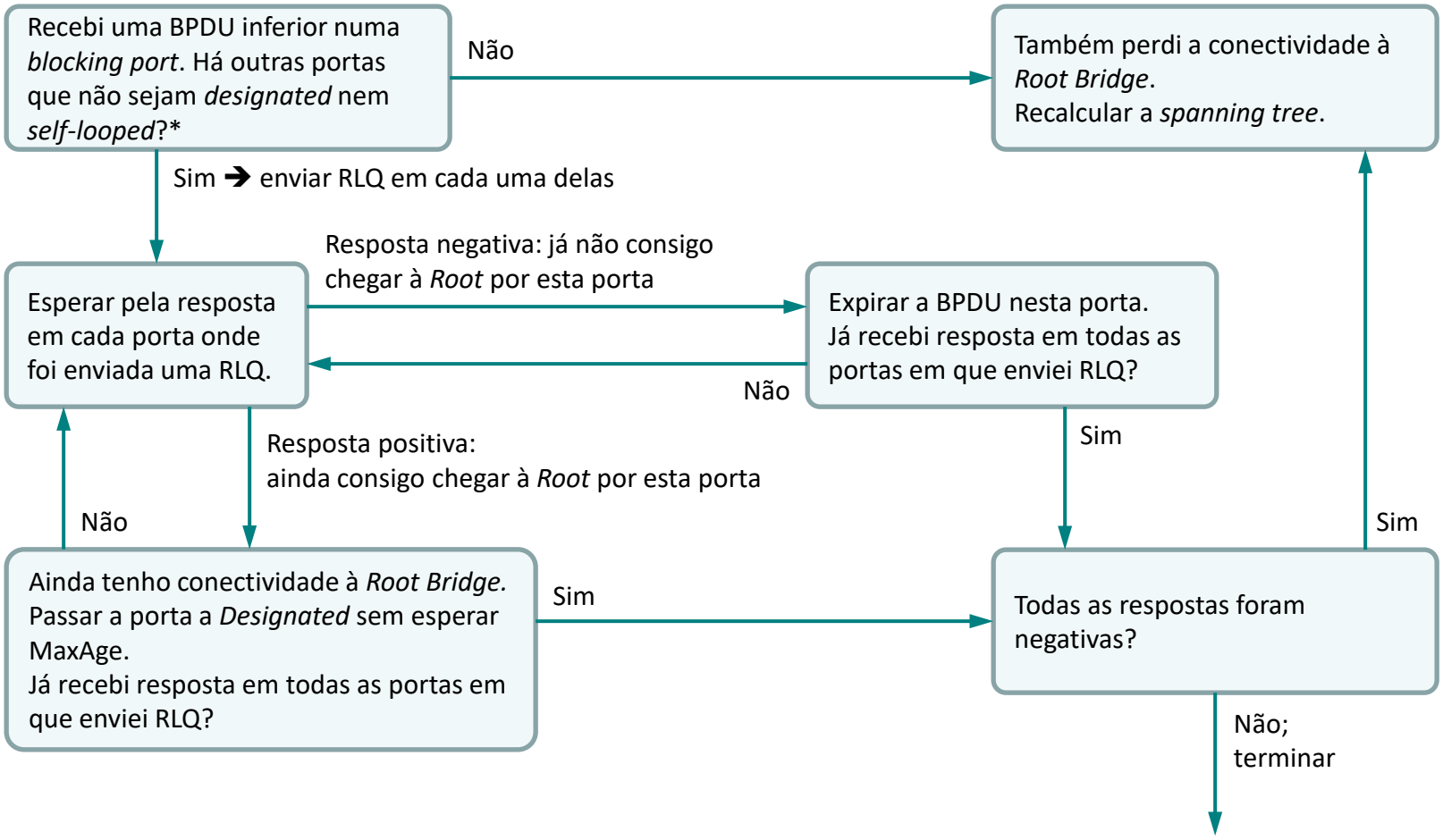
# *BackboneFast*

- Permite poupar 20s quando o comutador detecta indirectamente uma perda de ligação
- Extensão proprietária da Cisco
- Pode ser configurada em todos os comutadores

- Comutador recebe uma BPDU inferior (pior do que a sua) numa *Blocking Port*, vinda da *Designated Bridge*
  - Envia *Root Link Query BPDUs* para verificar se ainda tem conectividade à *Root Bridge*
  - Se ainda tiver, pode expirar imediatamente a BPDU superior que tinha recebido antes, sem ter que esperar MaxAge (20s)
    - Passa essa porta a *Designated Port* e começa a enviar BPDU por ela

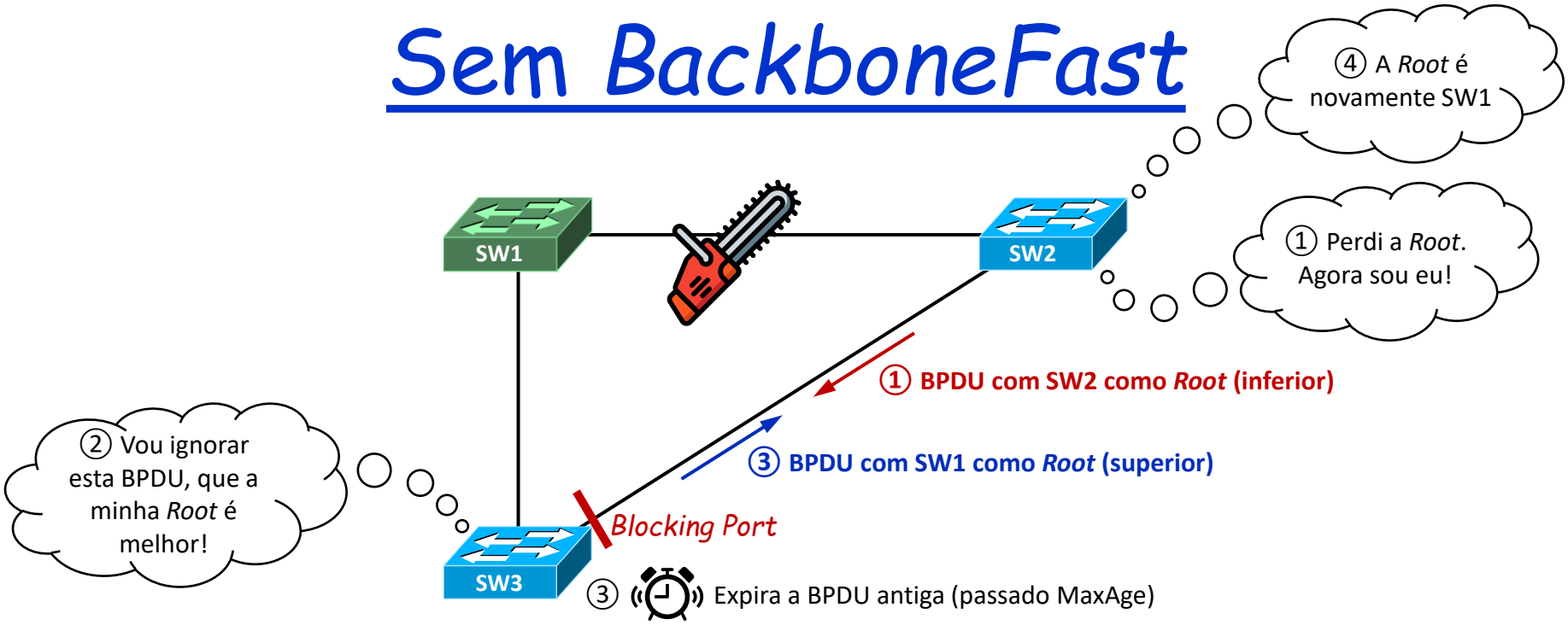
- Activar *BackboneFast*:
  `spanning-tree backbonefast`

# BackboneFast



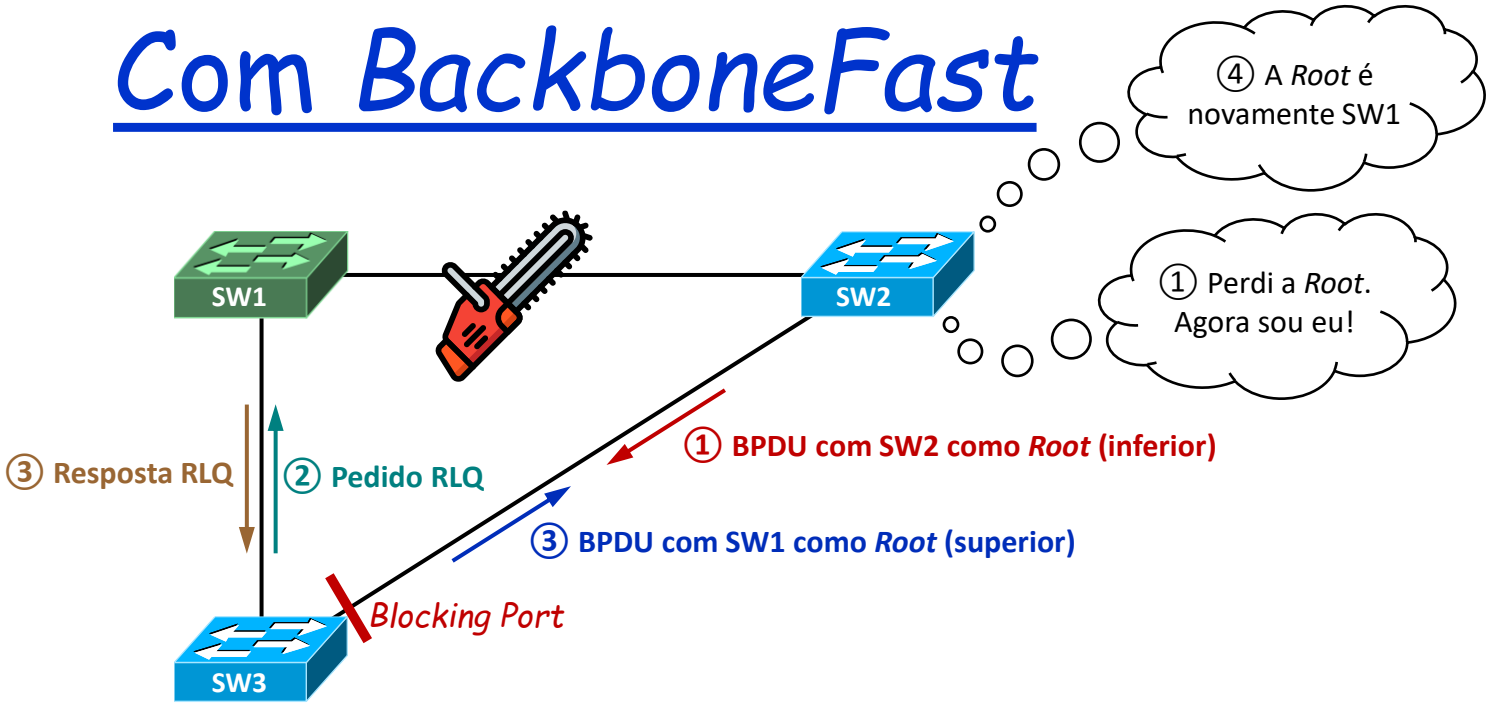

* Ou seja, há outras portas que possam conduzir à *Root Bridge* (incluindo, naturalmente, a *Root Port*)?

# Sem BackboneFast




1. SW2 detecta perda de ligação no *Root Port*. Como não tem alternativa, assume-se *Root* e envia BPDU em conformidade a SW3
2. SW3 recebe a BPDU inferior e ignora-a
3. Quando a BPDU antiga expira (MaxAge), passa a porta de *Blocking* para *Designated* e começa a enviar BPDU
   - Mas ainda terá que passar pelos estados *Listening* e *Learning*
4. SW2 ouve a BPDU superior do SW3, passa a porta de *Designated* para *Root Port* e deixa de enviar BPDU

# Com BackboneFast




1. SW2 detecta perda de ligação no *Root Port*. Como não tem alternativa, assume-se *Root* e envia BPDU em conformidade a SW3
2. SW3 recebe a BPDU inferior e envia pedido RLQ a SW1 para ver se chega à *Root*
3. SW3 recebe resposta RLQ, passa a porta de *Blocking* para *Designated* e começa a enviar BPDU
   - Ainda terá que passar pelos estados *Listening* e *Learning*, mas poupou quase 20s
4. SW2 ouve a BPDU superior do SW3, passa a porta de *Designated* para *Root Port* e deixa de enviar BPDU

# Rapid Spanning Tree Protocol (RSTP)

- *O Rapid Spanning Tree Protocol* RSTP foi originalmente definido na emenda 802.1w
- Objectivo principal: resolver o problema dos tempos de convergência excessivamente longos do STP
  - Reage após a perda de apenas 3 BPDU (em vez de 10)
  - Evita depender de temporizadores
  - Usa mecanismos semelhantes a *PortFast, UplinkFast* e *BackboneFast*
- Mantém retrocompatibilidade com o STP
  - Podem misturar-se na mesma rede
  - Quando um comutador RSTP descobre que o outro lado só fala STP, passa a falar também STP nessa porta
- Alterações ao formato e regras de envio de BPDU, estados de porta e funções de porta

Slides adaptados de https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/24062-146.html

# Estados e funções de porta

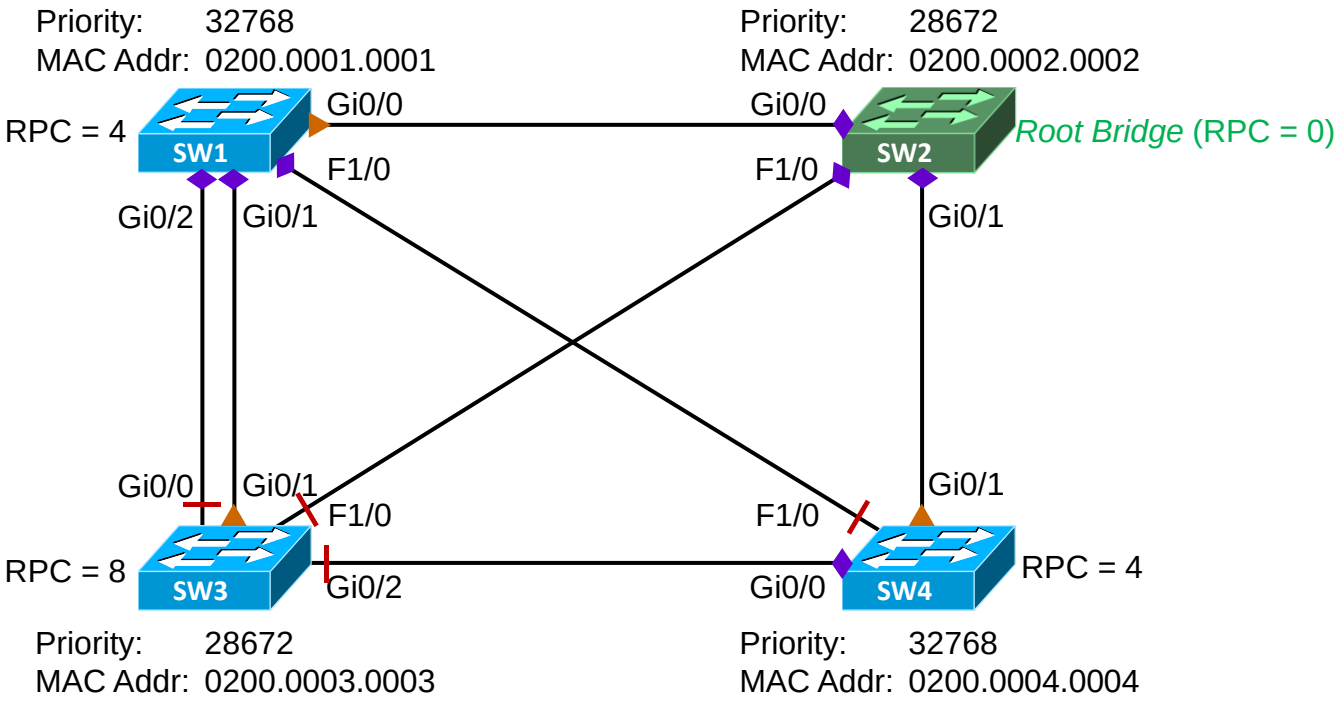
- O STP definia 5 estados de porta, nos quais misturava dois aspectos diferentes
  - Função desempenhada pela porta na topologia
  - Se a porta reenvia ou bloqueia as tramas
- A especificação do RSTP define explicitamente funções de porta e estados de porta, separando esses dois aspectos

# Funções de porta

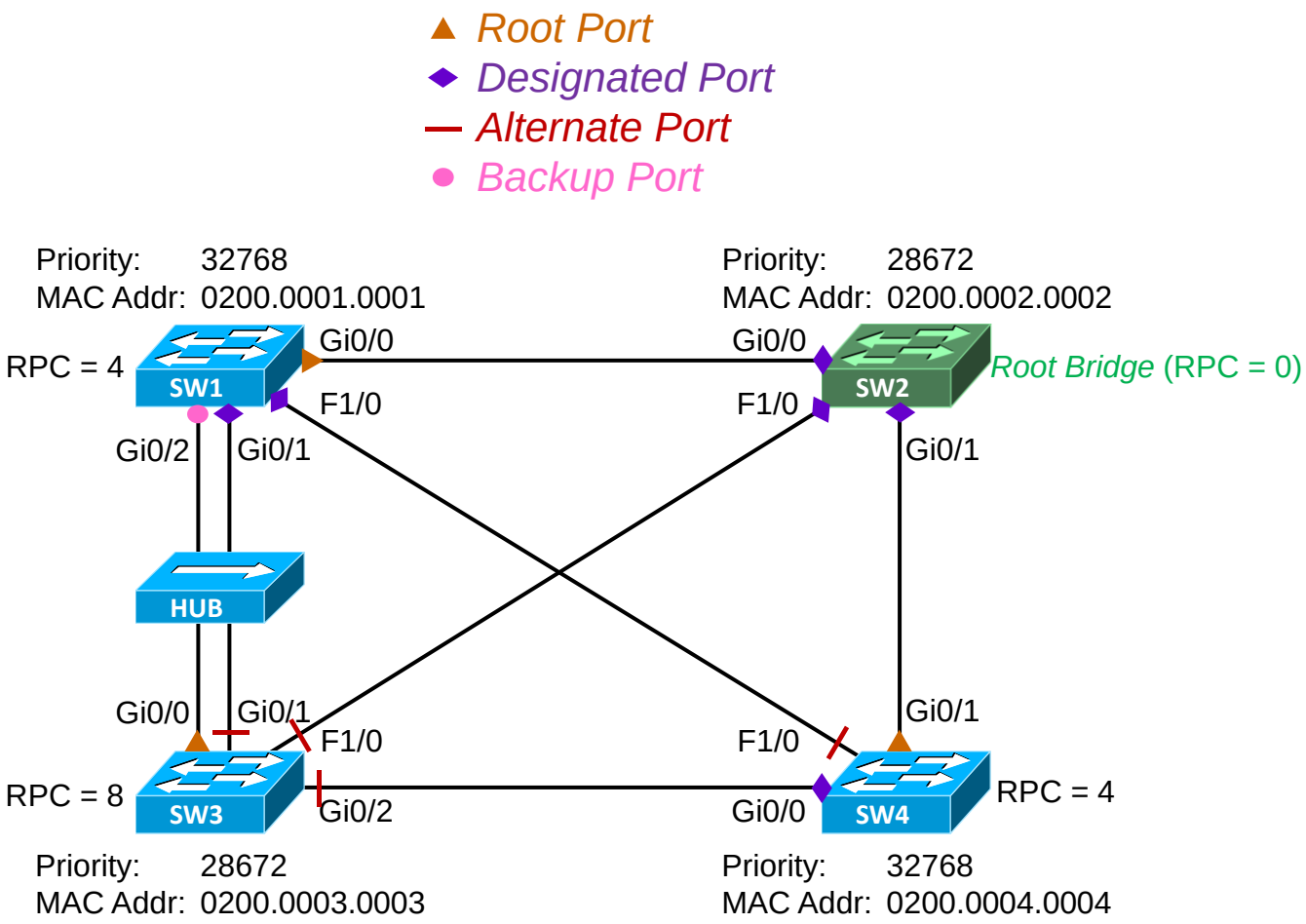
- O RSTP define 5 funções que uma porta pode desempenhar:
  - *Root Port:* melhor caminho para a *Root Bridge*
    - Recebe BPDU superiores de uma *bridge* diferente
  - *Alternate Port:* caminho alternativo para a RB
    - Fica no estado *Discarding*, mas pode substituir o RP e passar ao estado *Forwarding* imediatamente (tal como com *UplinkFast*)
    - Recebe BPDU superiores de uma *bridge* diferente
  - *Designated Port:* reenvia tramas para o segmento de rede
    - Pertence à *bridge* que envia a melhor BPDU para esse segmento
  - *Backup Port:* porta alternativa à DP, se esta falhar
    - Aparece apenas se ligarmos mais de um porta de um comutador a um mesmo segmento de rede (concentrador) — não se encontra em redes actuais
    - Fica no estado *Discarding*
    - Recebe BPDU superiores da própria *bridge*
  - *Disabled Port:* porta desactivada adiministrativamente

# Exemplo anterior com RSTP

▲ *Root Port*
◆ *Designated Port*
— *Alternate Port*



# Exemplo com *Backup Port*



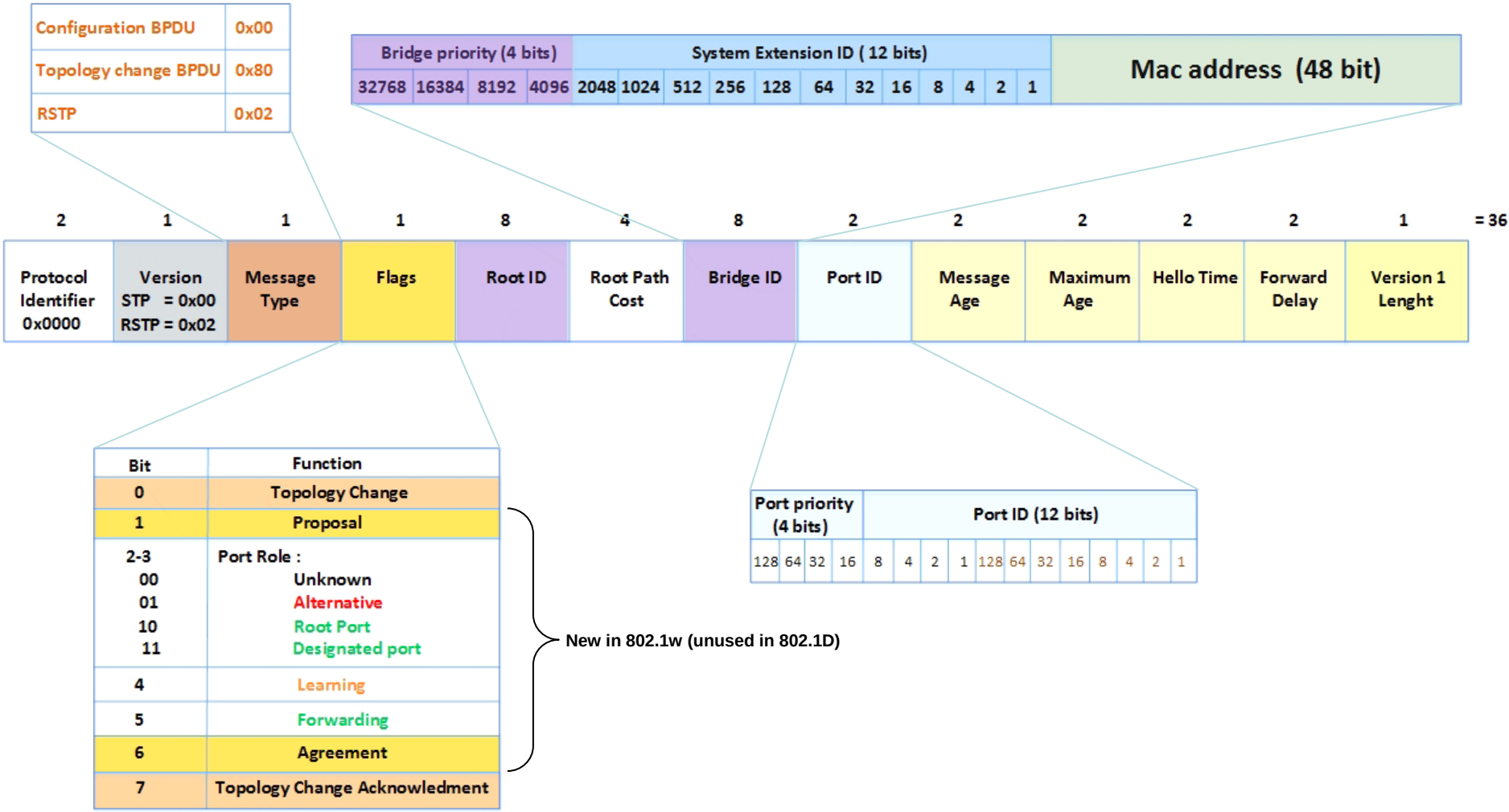

Notar também a troca das portas Gi0/0 e Gi0/1 no SW3. Como agora recebem a mesma BPDU, aplica-se uma regra adicional de desempate: fica RP a porta com PortID (local) mais baixo.

# Estados de porta

- O RSTP define os seguintes estados de porta:
  - *Discarding*
  - *Learning*
  - *Forwarding*

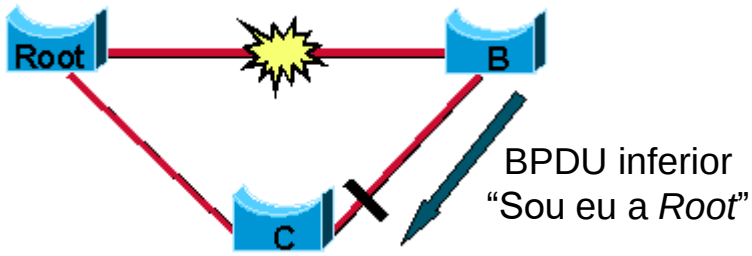
- Diferenças em relação ao STP:
  - O estado *Listening* não existe (foi considerado desnecessário)
  - Os estados *Disabled* e *Blocking* foram fundidos num só, o *Discarding*

# Formato das BPDU



# Diferenças no tratamento das BPDU

- Cada *bridge* gera BPDU a cada *HelloTime* (*default* = 2s)
  - No 802.1D apenas eram repassadas as recebidas na *Root Port*
- Envelhecimento mais rápido
  - Não receber 3 BPDU consecutivas de um vizinho (*root* ou *designated bridge*) significa que se perdeu a conectividade a esse vizinho
    - Demora apenas 6s (3 x *HelloTime*)
    - No 802.1D o problema poderia estar mais a montante, dado que as BPDU apenas eram repassadas
  - A BPDU recebida expira imediatamente, sem ter que esperar *MaxAge*
- BPDU inferiores são aceites
  - Em resposta, sabendo que mantém conectividade à *root*, a *bridge* envia imediatamente a sua BPDU (superior)
  - Mecanismo análogo ao *BackboneFast*



# Transição rápida para o estado *Forwarding*

- A transição muito mais rápida para o estado *Forwarding* é a característica mais importante do RSTP (e que lhe dá o nome)
- Uma *bridge* consegue verificar activamente se é seguro transitar uma porta para *Forwarding* sem depender de temporizadores
  - Com 802.1D tinha que esperar passivamente pela reconvergência
- Dois conceitos importantes para a reconvergência rápida são
  - Portas *Edge*
  - Tipo de ligação

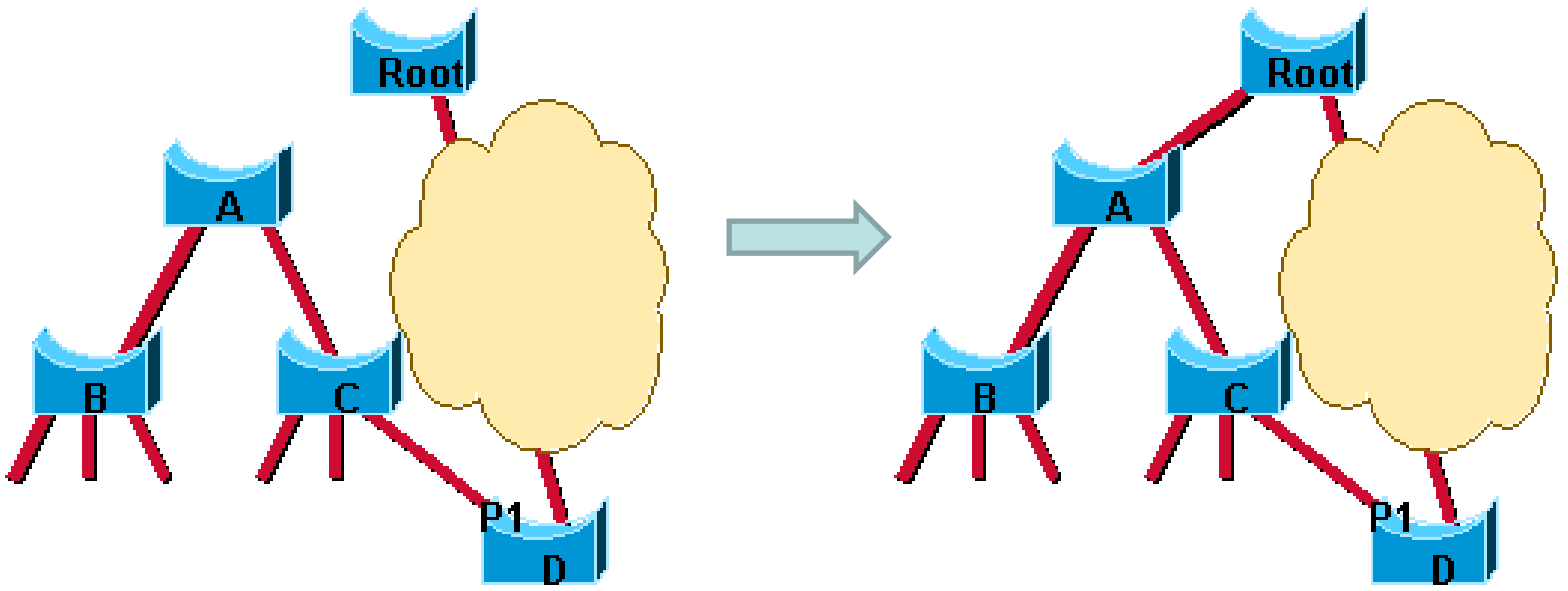
# *Edge Ports*

- *Edge Ports* são portas directamente ligadas a terminais (ou *routers*)
  - Conceito semelhante ao *PortFast*
    - No Cisco IOS, a configuração é semelhante
- Por não estarem ligadas a comutadores ou *bridges*, *Edge Ports* não podem criar ciclos
  - É seguro transitá-las imediatamente para o estado *Forwarding*
- Se for recebida uma BPDU numa *Edge Port*, ela passa a ser tratada como uma porta "normal"
  - Estado operacional diferente do configurado
- Alterações ao estado da ligação de uma *Edge Port* não geram alterações topológicas

# Tipo de ligação

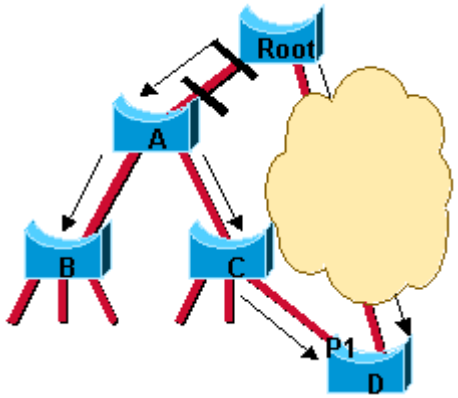
- Portas que não sejam *Edge* podem corresponder a ligações ponto-a-ponto ou partilhadas
  - Ponto-a-ponto — ligação *full duplex*, possivelmente a outra *bridge*
    - No máximo, uma
  - Partilhada — ligação *half duplex* a um segmento partilhado, e.g., um concentrador (*hub*)
    - Pode ter várias *bridges* ligadas
- Tipo de ligação automaticamente inferido do modo *duplex*
  - Mas é possível forçar por configuração
- Transição rápida só é possível em *Edge Ports* e em ligações ponto-a-ponto
  - Nas redes actuais já não se usam ligações partilhadas ☺

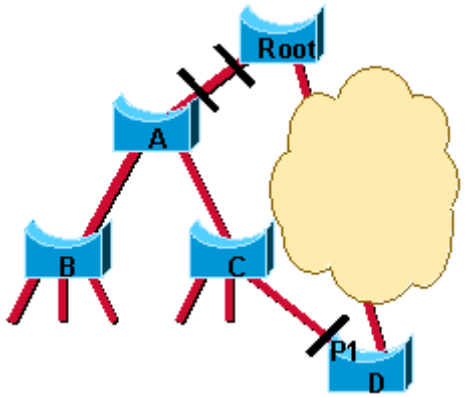
# Convergência: STP (802.1D) vs. RSTP (802.1w)




Cenário de teste: acrescentar uma ligação entre a *bridge A* e a *Root Bridge*

# STP (802.1D)

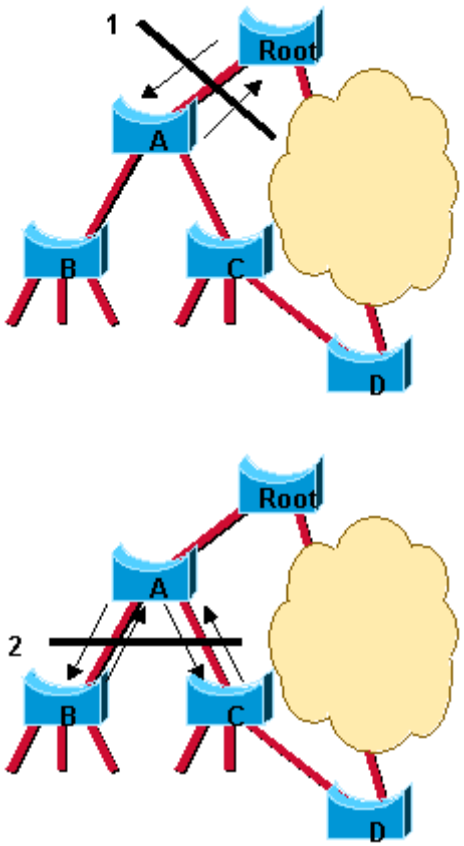



- Ambas as portas da nova ligação são bloqueadas para prevenir ciclos
  - Estado *Listening*
- As BPDU começam a fluir da *Root Bridge* através de *A*
- Reconhecendo essas BPDU como superiores, B e C aceitam-nas e propagam-nas
- Rapidamente, essas BPDU chegam a D
- Uma vez que são inferiores às recebidas pelo outro lado, D bloqueia a porta P1
- Eventualmente, as portas da nova ligação passam ao estado *Forwarding*, mas durante 30s…
- … parte da rede ficou inacessível ☹

O problema é a falta de um mecanismo de *feedback* rápido para indicar que a rede convergiu e as portas entre *A* e a *Root* podem ser desbloqueadas

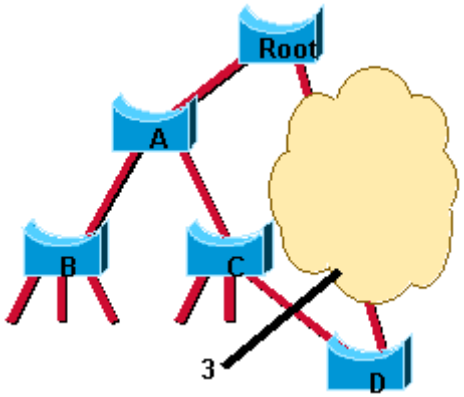
# RSTP (802.1w)




- Ambas as portas da nova ligação ficam *Designated Blocking** e começa uma negociação
  - Mecanismo Proposta-Acordo
- Ao receber a BPDU da *Root*, *A* bloqueia as suas *Designated Ports* que não são Edge
  - Ligações a B e a C
  - Esta operação designa-se *sync*
- Imediatamente a seguir, *A* autoriza a *Root* a passar a porta para o estado *Forwarding*
- O processo repete-se nas *bridges* B e C
  - Todas as outras portas de B são *Edge*, pelo que a sua ligação a *A* pode ficar activa sem risco
  - A BPDU de D é inferior à de C, pelo que a porta de C que o liga a D será *Designated*

*No `show spanning-tree` o estado *Discarding* aparece como *Blocking*

# RSTP (802.1w)



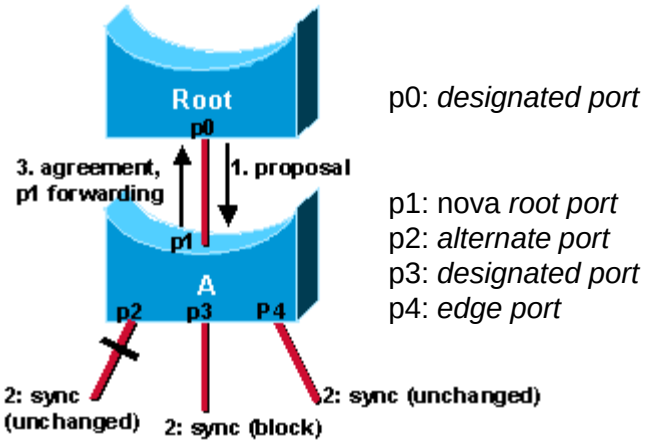

- Em D, o custo para chegar à *Root* através de *C* é superior ao do caminho que já tinha
  - D mantém a *Root Port* que tinha
  - D bloqueia a porta que a liga a *C*
- Disrupção muito menor da rede…
- … e sem depender de temporizadores!

**IMPORTANTE:**

- A negociação só é possível em ligações ponto-a-ponto
- Se o administrador se esquecer de configurar as *Edge Ports* como tal, a sua conectividade é afectada ao adicionar a ligação entre *A* e a *Root*
  - Não são recebidas BPDU de resposta (*Agreement*)

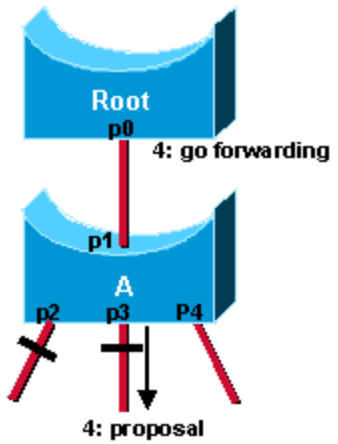
# Mecanismo Proposta-Acordo

- Quando uma ligação é adicionada, ambas as portas ficam *Designated Discarding* (*Blocking*)
- As BPDU enviadas nas portas em estado *Discarding* ou *Learning* (e apenas nestes) têm a *flag Proposal* activa
- Ao receber uma BPDU *Proposal* superior à sua, a *bridge A* inicia um *Sync* nas outras portas
- Uma porta está sincronizada com a nova informação se
  - Está bloqueante (estado *Discarding* numa topologia estável), ou
  - É uma *Edge Port*
- Neste caso, apenas a p3 não cumpria um dos critérios e teve que ser bloqueada



# Mecanismo Proposta-Acordo

- Com as outras portas em *Sync*, A pode activar a porta p1 e responder com um *Agreement*
  - Cópia da BPDU recebida mas com a *flag Agreement* em vez da *Proposal*
- Após receber o *Agreement*, a *Root* passa a porta p0 ao estado *Forwarding*, sem risco
- Na porta p3, ainda bloqueada, A envia uma *Proposal* ao seu vizinho abaixo
  - O processo propaga-se pela topologia, repondo rapidamente a conectividade
  - Não depende de temporizadores!
- Se não fosse recebido o *Agreement*, a porta p0 transitaria lentamente para o estado *Forwarding*, como no 802.1D



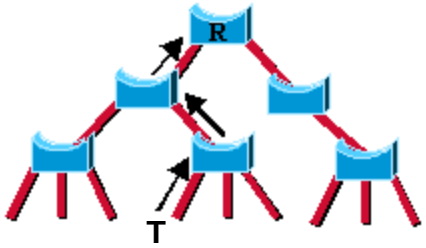

NOTA: A Cisco implementa um mecanismo proprietário que torna o processo ainda mais eficiente, bloqueando temporariamente apenas as portas em direcção à que acabará por ser bloqueada

# "UplinkFast"

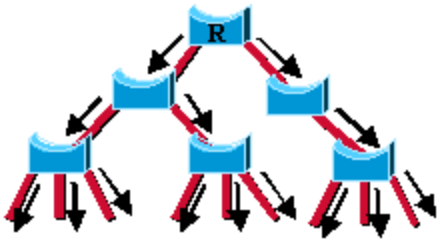
- O RSTP inclui um mecanismo análogo ao *UplinkFast*
  - Activo automaticamente, não precisa de configuração
- Se a *bridge* perder ligação na *Root Port*, põe a melhor *Alternate Port* em modo *Forwarding*
  - Passa a ser a nova *Root Port*
- Isto dá origem a uma *Topology Change*
- O mecanismo TC do 802.1w limpa as entradas na *CAM* da *bridge* a montante
  - Não há necessidade dos *dummy multicast packets*

# Novo mecanismo *Topology Change*

- Alteração de topologia no 802.1D:




*A Topology Change Notification* propaga-se até à *Root Bridge* (de forma fiável)

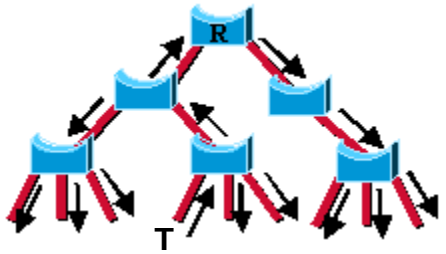



Durante MaxAge + ForwardDelay, a Root envia TC BPDU para acelerar renovação das tabelas CAM

- No 802.1w, só a passagem de uma porta não-*Edge* ao estado *Forwarding* é considerada alteração de topologia
  - Perdas de conectividade não são (ao contrário do 802.1D)

# Novo mecanismo *Topology Change*

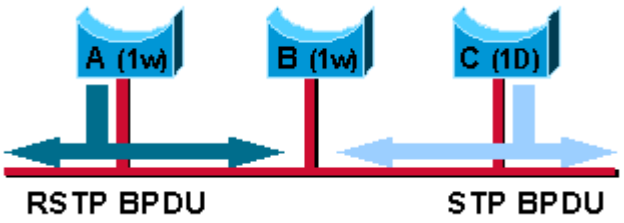
- Quando uma *bridge* 802.1w detecta uma alteração à topologia
  - Inicia o temporizador *TC While* nas suas *Designated Ports* que não são *Edge* e na *Root Port*, se necessário
    - Duração 2x*HelloTime*
  - Durante esse tempo, envia BPDU com o bit TC, inclusive na *Root Port*
  - Limpa os endereços *MAC* associados a essas portas
- Ao receber uma BPDU TC numa porta, uma *bridge* efectua esse mesmo procedimento nas outras portas
- É a partir da *bridge* onde a alteração ocorreu que as BPDU TC são enviadas
  - Não da *Root Bridge*



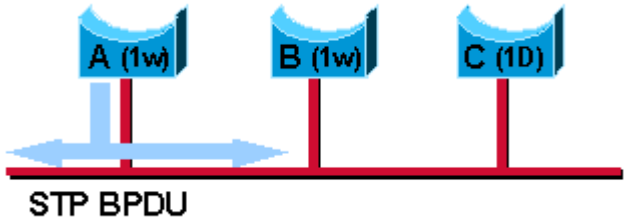

- A limpeza das *CAM* pode temporariamente dar origem a mais *broadcasts*, mas favorece a restauração rápida da conectividade

# Retrocompatibilidade entre RSTP e STP

- Uma *bridge* RSTP é capaz de interoperar com *bridges* STP
  - Mas a convergência rápida deixa de ser possível
- Cada porta mantém a indicação da versão do protocolo a usar
- Um temporizador de migração (3s) limita a frequência de alteração da versão a usar na porta
  - Iniciado quando a porta fica activa ou quando há alteração na versão
- A versão a usar é a da BPDU recebida após expirar o temporizador




C não entende a BPDU de A e envia a sua própria (inferior, neste caso)




A recebe a BPDU de C e muda a porta para o modo 802.1D. Agora C entende a BPDU e aceita-a como superior.

# Outras variantes do STP

- Para além do STP tradicional e do RSTP, existem outras variantes:
  - Per-VLAN Spanning Tree (PVST+)
    - Propriedade da Cisco
  - Rapid Per-VLAN Spanning Tree (RPVST+)
    - Propriedade da Cisco
  - Multiple Spanning Trees (MST)
    - Definido pelo IEEE: 802.1s, depois integrado no 802.1Q-2005

# Per-VLAN Spanning Tree (PVST+)

- Essencialmente, STP com uma árvore por VLAN
- Principais diferenças entre STP e PVST+
  - O STP cria uma árvore comum, enquanto o PVST+ cria uma árvore por VLAN
  - No STP é enviado apenas um conjunto de BPDU, enquanto no PVST+ são enviadas BPDU independentes por VLAN
  - No STP as BPDU são enviadas para um endereço MAC multicast definido pelo IEEE (`0180.c200.0000`), enquanto no PVST+ são enviadas para um endereço MAC multicast definido pela Cisco (`0100.0ccc.cccd`) em portas *trunk*
  - Em ligações *trunk*, o STP envia as BPDU na VLAN nativa, enquanto o PVST+ envia as BPDU com o encapsulamento da VLAN respectiva
  - O PVST+ adiciona um campo (TLV) às BPDU indicando o VLAN ID, o que não é necessário no STP que ignora as VLAN
  - No STP o *System ID Extension* é sempre 0, enquanto no PVST+ contém o VLAN ID
- Activar PVST+: `spanning-tree mode pvst`

# Rapid Per-VLAN Spanning Tree (RPVST+)

- Essencialmente, RSTP com uma árvore por VLAN
- Principais diferenças entre RSTP e RPVST+
  - O RSTP cria uma árvore comum, enquanto o RPVST+ cria uma árvore por VLAN
  - No RSTP é enviado apenas um conjunto de BPDU, enquanto no RPVST+ são enviadas BPDU independentes por VLAN
  - No RSTP as BPDU são enviadas para um endereço MAC multicast definido pelo IEEE (`0180.c200.0000`), enquanto no RPVST+ são enviadas para um endereço MAC multicast definido pela Cisco (`0100.0ccc.cccd`) em portas *trunk*
  - Em ligações *trunk*, o RSTP envia as BPDU na VLAN nativa, enquanto o RPVST+ envia as BPDU com o encapsulamento da VLAN respectiva
  - O RPVST+ adiciona um campo (TLV) às BPDU indicando o VLAN ID, o que não é necessário no RSTP que ignora as VLAN
  - No RSTP o *System ID Extension* é sempre 0, enquanto no RPVST+ contém o VLAN ID
- Activar RPVST+: `spanning-tree mode rapid-pvst`