



*LET'S  
BUILD  
TOMORROW  
TODAY*

# *QoS Configuration Migrations From Classic IOS to IOS XE*

David Roten, Technical Marketing Engineer

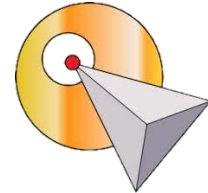
BRKARC-2031

# BRKARC-2031 Agenda

- Introduction
- What is new with QoS in IOS XE
- IOS XE vs IOS Classic behavior differences
- Platform specific caveats for IOS XE
- Specific use case migrations from IOS classic to IOS XE
- Conclusion

# Introduction

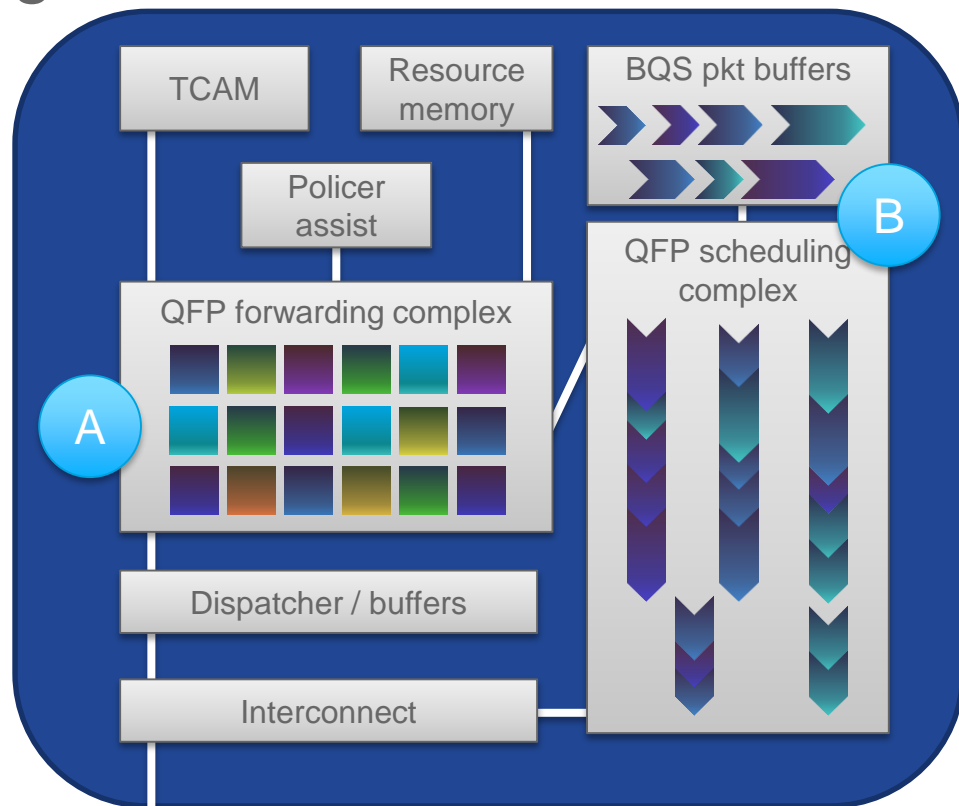
- Multicore forwarding
  - It's like HDTV, prettier picture but all of a sudden you can see all the duct tape on set
- More scheduling parameters for detailed control
  - Same bat-configuration, different bat-behavior, what gives?
- More rules for constructing hierarchies
  - What do you mean Legos aren't the same thing as Duplos?
- Etherchannel QoS combination
  - IDIC – Infinite Diversity in Infinite Combination is a Vulcan thing
- Multiple platforms behave the same way now
  - One Ring to Rule Them All



# QoS data path forwarding

ASR1000 QFP hardware engine handles packet forwarding. In ISR4000 routers this is done via software using the same microcode.

- A. Available core is allocated for packet processing (ingress QoS, police, WRED drops, mark, match)
- B. Based on default and user configurations, packets are scheduled for transmission based on the egress physical interface. All scheduling and queuing happens in this second phase. More hardware regimented.



# *What's new with QoS in IOS XE*

# What's new with QoS in IOS XE

- 2 versus 3 parameter scheduler
- Queue-limit options
- 2 levels of priority
- Aggregate GEC QoS
- Tunnel QoS configurations
- Service-fragments
- Service groups
- Shaping parameters ignored
- LLQ vs. PAK\_PRI

# *2 versus 3 parameter scheduling*



# IOS XE MQC based QoS – 3 parameter scheduler

- **IOS XE** provides an advanced 3 parameter scheduler

Minimum               - bandwidth

Excess                - bandwidth remaining

Maximum             - shape

- 3 parameter schedulers share excess bandwidth equally in default configuration
- `bandwidth` and `bandwidth remaining` may not be configured in the same policy-map
  - **IOS XE** schedulers are capable of supporting this configuration and there is a roadmap feature to allow configuration of both parameters simultaneously

# What are the three parameters?

```
policy-map child
  class voice
    priority level 1
    police cir 2000000 (bit/sec)
  class critical_services
    bandwidth 5000
  class internal_services
    shape average percent 100
  class class-default
!
policy-map parent
  class class-default
    shape average 25000000
    service-policy child
```

**Minimum** is defined by the `bandwidth` command or priority classes with policers. Classes with these directives are guaranteed to receive at least and maybe more bandwidth.

**Excess** is defined by the `bandwidth remaining` command. Excess is the amount of bandwidth available once all the minimums guarantees are satisfied. By default, classes have a remaining ratio of 1 even if bandwidth remaining is not configured.

**Maximum** is implemented by shapers. Traffic rates beyond the shaper rates will be held in queues.



# IOS XE MQC based QoS – 3 parameter scheduler

- The bandwidth remaining (BR) adjust sharing of excess bandwidth
- Two options are available:
  - bandwidth remaining ratio X, where X ranges from 1 to 1000, with variable base
  - bandwidth remaining percent Y, where Y ranges from 1 to 100, with fixed base of 100
- bandwidth remaining percent (BR%) based allocations remain the same as classes are added to a configuration
- bandwidth remaining ratio (BRR) based allocations adjust as more queuing classes are added to a configuration with or without BRR configured
  - base changes as new classes are added with their own ratios defined or with a default of 1
- By default, all classes have a bandwidth remaining ratio or percent value of 1



# 3 versus 2 parameter scheduler behavior

policy-map test

class D1

bandwidth percent 12

class D2

bandwidth percent 6

class class-default

bandwidth percent 2

- **2 parameter schedulers share excess bandwidth proportionally**

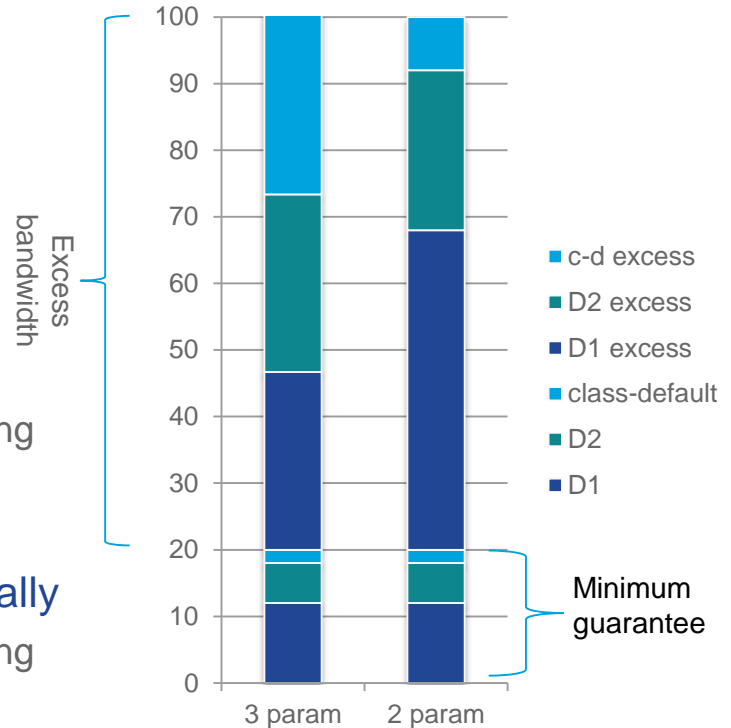
if the above policy-map is overdriven, the 80% of the remaining bandwidth is proportionally split amongst all three queues

$$12\% + (12 / (12+6+2) * 80\%) = 60\%$$

- **3 parameter schedulers share excess bandwidth equally**

if the above policy-map is overdriven, the 80% of the remaining bandwidth is equally split amongst all three queues

$$12\% + (80\% / 3) = 12\% + 26.67\% = 38.67\%$$



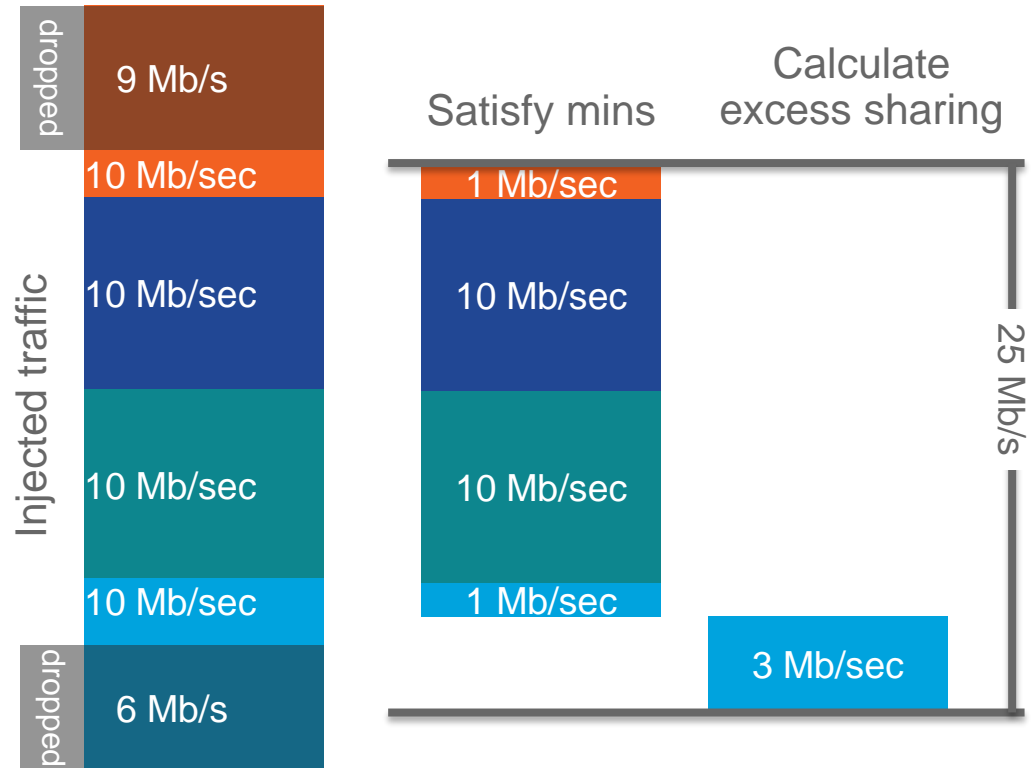
# 3 versus 2 parameter scheduler behavior



- 3 parameter schedulers share excess bandwidth **equally** after satisfying minimum requirements
- 2 parameter schedulers share excess bandwidth **proportionally** based on the minimum rate configured  
Gives even more of an advantage to traffic that was given a heavy weight in the first place
- To get 2 parameter behavior from IOS XE scheduler, use BRR configuration  
This grants no minimums but the excess sharing will provide the same type of behavior and be proportional from the satisfied minimum to max rate which is essentially the same as what the 2 parameter schedule provides with minimum + proportional excess.

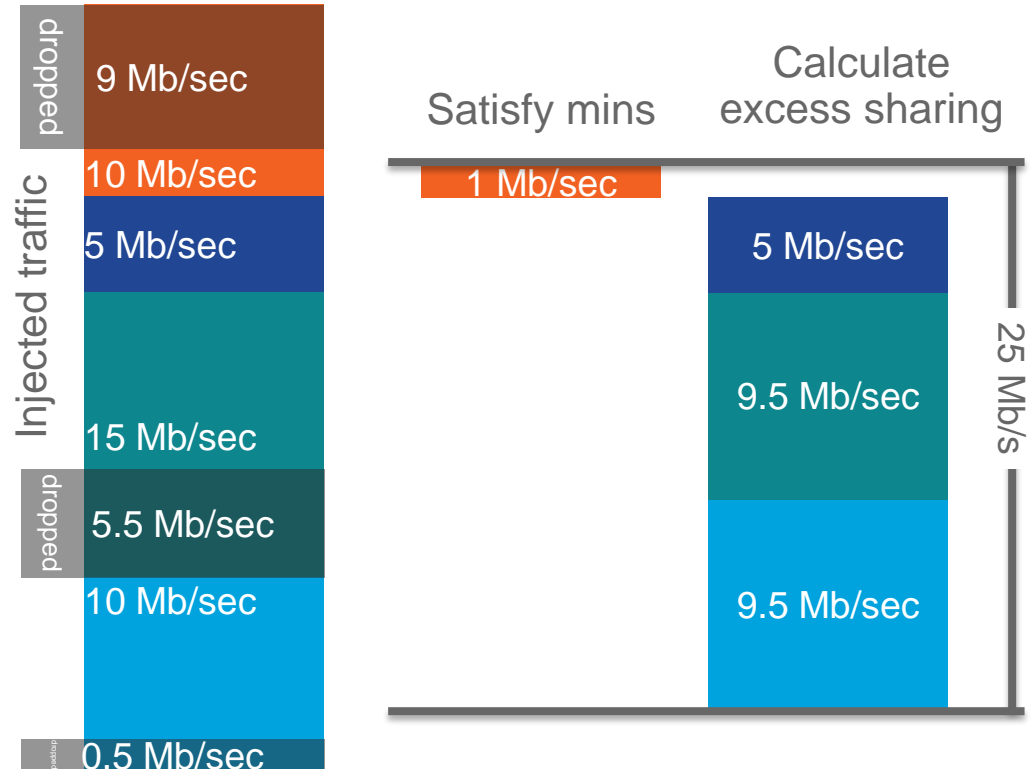
# QoS 3 parameter scheduler – minimum

```
policy-map child
  class voice
    priority level 1
    police cir 1000000 (bit/sec)
  class critical_services
    bandwidth 10000 (kbit/sec)
  class internal_services
    bandwidth 10000 (kbit/sec)
  class class-default
    bandwidth 1000 (kbit/sec)
!
policy-map parent
  class class-default
    shape average 25000000
    service-policy child
```



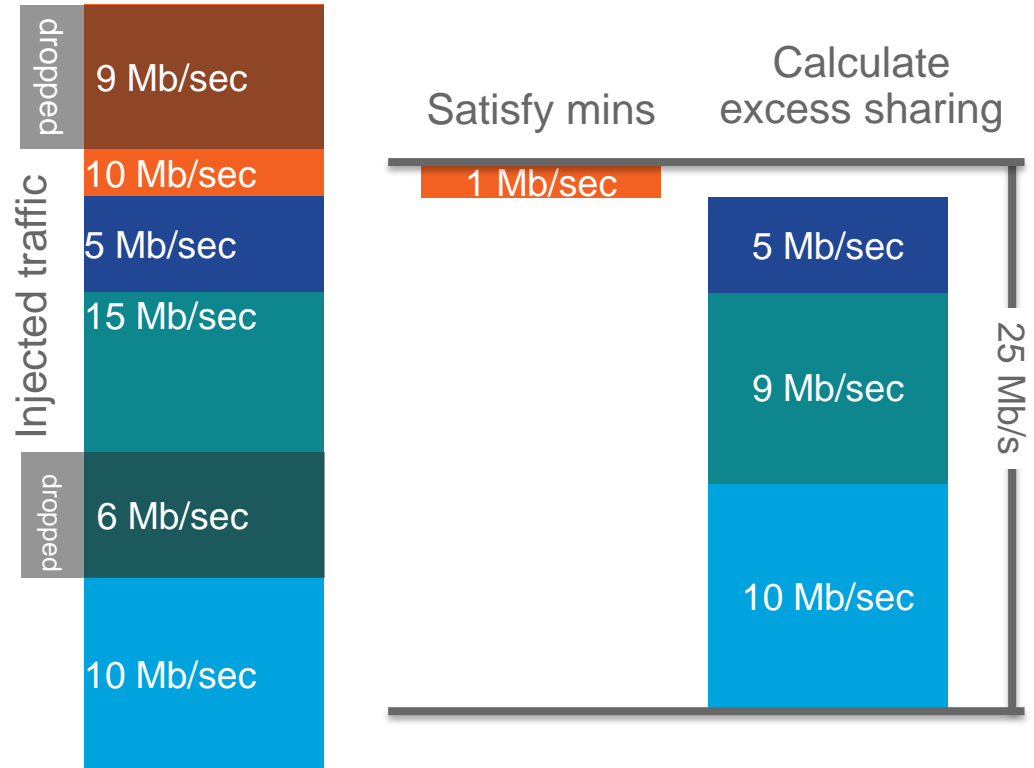
# QoS 3 parameter scheduler – excess ratio

```
policy-map child
  class voice
    priority level 1
    police cir 1000000 (bit/sec)
  class critical_services
    bandwidth remaining ratio 4
  class internal_services
    bandwidth remaining ratio 1
  class class-default
    bandwidth remaining ratio 1
!
policy-map parent
  class class-default
    shape average 25000000
    service-policy child
```



# QoS 3 parameter scheduler – excess %

```
policy-map child
  class voice
    priority level 1
    police cir 1000000 (bit/sec)
  class critical_services
    bandwidth remain percent 40
  class internal_services
    bandwidth remain percent 10
  class class-default
!
!
policy-map parent
  class class-default
    shape average 25000000
    service-policy child
```





# BR% vs BRR behavior

```
policy-map test
class D1
bandwidth remaining percent 12
class D2
bandwidth remaining percent 6
class D3
bandwidth remaining percent 2
```

```
policy-map test
class D1
bandwidth remaining ratio 12
class D2
bandwidth remaining ratio 6
class D3
bandwidth remaining ratio 2
```

	D1	D2	D3	c-d
BR percent	12%	6%	2%	80%
BR ratio	57%	29%	9.5%	4.5%
	12 / 21	6 / 21	2 / 21	1 / 21

- Each class is driven with a packet stream which exceeds the parent shaper rate
- BR% offers class-default 80% since it is based on a total pool of 100
- BRR offers class-default only 4.5% since it is based on a pool of 21 (12+6+2+1)

# *Queue limit options*

# Queue-limit options

- **Classic IOS** offered only units of packets for defining queue limits
- **IOS XE** offers packets and also offers time and bytes (on ASR1000)
- Time and byte based are essentially the same thing since time is simply converted into bytes based on the speed of the interface (or parent shaper)
  - 150 ms latency on GigEthernet interface with MTU of 1500

$$0.150 \text{ sec} \times \frac{1E9 \text{ bits}}{\text{sec}} \times \frac{\text{byte}}{8 \text{ bits}} = 18,750,000 \text{ bytes}$$

# Queue-limit options

- Defining policy-maps in time units allows flexibility to have a single policy-map work for multiple interfaces instead of needing multiple variations of a single policy-map
- Using time / bytes based configuration gives a consistent latency profile
  - The latency associated with 4000 packets is much different if those packets are 64 or 1500 bytes
  - Assuming a GigE interface the range is 2 msec to 48 msec
- All queue-limit units in a policy and its related hierarchy must be the same units.
  - Precludes modification on the fly, must remove, modify and reapply

# Queue-limit options

```
policy-map queue-limit
class class-default
  queue-limit 150 ms
```

```
Rtr#show policy-map int GigabitEthernet0/3/0
GigabitEthernet0/3/0
```

Service-policy output: queue-limit

```
Class-map: class-default (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0000 bps, drop rate 0000 bps
Match: any
```

```
queue limit 150 ms/ 18750000 bytes
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts output/bytes output) 0/0
```

```
Rtr#show policy-map int Serial1/1/0
Serial1/1/0
```

Service-policy output: queue-limit

```
Class-map: class-default (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0000 bps, drop rate 0000 bps
Match: any
```

```
queue limit 150 ms/ 828937 bytes
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts output/bytes output) 0/0
```

# IOS XE – queuing memory management

- Packet Buffer DRAM treated as one large buffer pool
  - QFP has specific pool of memory for packet buffering, ISR4400 platforms use shared FP memory, ISR4300 platforms use common memory
- Two main building blocks for packet buffer DRAM
  - Block: each queue gets 1 Kbyte blocks of memory
  - Particle: packets are divided into 16 byte particles and linked together
- Advantages to such an implementation:
  - Less complex than buffer carving schemes
  - Fragmentation is minimal & predictable due to small sized blocks & particles
- Thresholds exist to protect internal control traffic and priority traffic

# Default queue-limit

- **Classic IOS** used 64 packets as the default queue-limit for all queues
- **IOS XE** uses 512 packets for priority queues and 50ms (with one exception of 25ms on ESP-40) of MTU sized packets for all other queues, with a strict minimum of 64 packets
- **IOS Classic** used 64 bytes for everything
- **IOS XE** platforms ignore the following buffering parameters for interface QoS
  - interface hold-queue
  - global IOS buffers configuration
  - IOS displays the affects of the commands, but no effect on traffic forwarding or QoS behavior

# IOS XE – queue limit management

- Without MQC QoS configuration, interface default queues have 50 ms of buffering in a packets based configuration (except on ASR1000 with ESP-40 which uses 25 ms)

$$queue\_limit_{packets} = \frac{interface\_speed_{bits/sec} \times 0.050_{sec}}{interface\_mtu_{bytes/packet} \times 8 \frac{bits}{byte}}$$

- MQC created queues have 50 ms of buffering by default in a packets based configuration (on all IOS XE platforms). The **speed** used however depends...

$$queue\_limit_{packets} = \frac{speed_{bits/sec} \times 0.050_{sec}}{interface\_mtu_{bytes/packet} \times 8 \frac{bits}{byte}}$$



# IOS XE – queue limit management

- Default queue limits are based on 50ms (except for ASR1000 ESP-40 at 25ms)
- If `bandwidth` parameter is configured, then that rate is used as the speed value
- If `bandwidth percent` parameter is configured, then that rate based on the percentage from the parent is used as the speed value
- If a `shape` parameter is configured, then that rate is used as the speed value
- If only `bandwidth remaining` is configured, then the parent's speed value is used

$$queue\_limit_{packets} = \frac{speed_{bits/sec} \times 0.050_{sec}}{interface\_mtu_{bytes/packet} \times 8 \frac{bits}{byte}}$$

# IOS XE – queue limit management

```
policy-map child
  class p7
    priority
    police cir percent 2
  class p6
    bandwidth 10000
    shape average 200000000
  class p5
    shape average 200000000
  class class-default
!
policy-map parent
  class class-default
    shape average 800000000
    service-policy child
```

Priority classes always have a default queue depth of 512 packets

Bandwidth classes use the configured bandwidth value to come up with  $\frac{10E6 \times 0.050}{1500 \times 8} = 41 \rightarrow 64 \text{ packets}$

Shape only classes use the configured shape value to come up with  $\frac{200E6 \times 0.050}{1500 \times 8} = 832 \text{ packets}$

$$queue\_limit_{packets} = \frac{speed_{bits/sec} \times 0.050_{sec}}{interface\_mtu_{bytes/packet} \times 8 \frac{bits}{byte}}$$

# IOS XE – queue limit management

```
policy-map child
  class p7
    priority
    police cir percent 2
  class p6
    bandwidth remaining ratio 20
    shape aver 30000000
  class p5
    bandwidth remaining ratio 10
  class class-default
!
policy-map parent
  class class-default
    shape average 800000000
  service-policy child
```

Priority classes always have a default queue depth of 512 packets

Classes with shape use the configured shape value as follows:  $\frac{30E6 \times 0.050}{1500 \times 8} = 125 \text{ packets}$

Bandwidth remaining only classes use the parent shape value as follows:  $\frac{800E6 \times 0.050}{1500 \times 8} = 333 \text{ packets}$

$$queue\_limit_{packets} = \frac{speed_{bits/sec} \times 0.050_{sec}}{interface\_mtu_{bytes/packet} \times 8 \frac{bits}{byte}}$$

# *2 levels of priority*

## 2 levels of priority

- **IOS Classic** offers a single level of priority
- **IOS XE** offers 2 levels of low latency queuing
- **IOS Classic** priority configuration migrates forward without modification
- It is recommended to utilize priority level 1 for voice and level 2 for video
- Priority level 1 queues are always drained before priority level 2 queues.

# Priority differences

- **IOS XE**: Naked / strict priority consumes ALL available bandwidth from the parent
  - No other classes in that policy can use `bandwidth` command to reserve a minimum
- **IOS Classic**: Naked / strict priority consumes 99% of all available bandwidth from the parent, 1% reserved for all other classes in aggregate
- **IOS Classic** priority commands with policer, use the parent's minimum rate for calculation. (**IOS XE** QoS uses the parent's maximum rate.)
- **IOS Classic** limits priority throughput to the minimum guarantee of the parent during physical interface congestion.

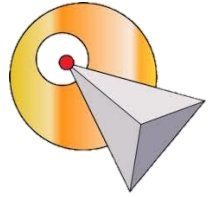
# Priority behavior differences

	IOS Classic	IOS XE
<b>Strict Priority</b>	Maximum throughput is minimum guaranteed for the particular child	Maximum throughput is up to the parent maximum
	By default, minimum guaranteed is physical bandwidth / number of classes. Minimum can be configured in the parent.	Minimum cannot be configured in parent.
	If parent oversubscribed, each child will get up to its minimum of priority traffic	If parent oversubscribed, each child will get proportional bandwidth according to the parent excess configuration
<b>Priority percent</b>	Maximum congestion throughput is calculated based on the parent minimum or maximum value, whichever is less.	Maximum congestion throughput is always calculated on the parent maximum
	n/a	If physical interface is oversubscribed, each child will get proportional bandwidth <a href="#">according to the parent excess configuration</a>
<b>Police percent</b>	Determined by maximum rate	Always determined by maximum rate
	When used in conjunction with priority, the percentage of the minimum guaranteed is deducted from bandwidth available to other classes.	Minimum guaranteed for priority propagation uses rate determined from the maximum.

# *Aggregate Etherchannel QoS*

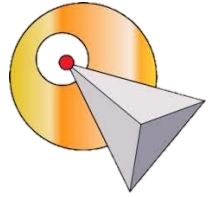


# IOS XE Etherchannel QoS support



- With VLAN based load balancing
  1. Egress MQC queuing configuration on Port-channel sub-interfaces
  2. Egress MQC queuing configuration on Port-channel member link
  3. Policy Aggregation – Egress MQC queuing on sub-interface
  4. Ingress policing and marking on Port-channel sub-interface
  5. Egress policing and marking on Port-channel member link
  6. Policy Aggregation – Egress MQC queuing on main-interface (XE2.6 and higher)
- Active/standby with LACP (1+1)
  7. Egress MQC queuing configuration on Port-channel member link (XE2.4 and higher)
  9. Egress MQC queuing configuration on PPPoE sessions, model D.2 (XE3.7 and higher)
  10. Egress MQC queuing configuration on PPPoE sessions, model F (XE3.8 and higher)
- Etherchannel with LACP and load balancing (active/active)
  8. Egress MQC queuing configuration supported on Port-channel member link (XE2.5 and higher)
  11. **Aggregate Etherchannel QoS support on Port-channel main-interface (XE3.12 and higher)**

# IOS XE Aggregate GEC QoS



- Without  $\mu$ code gymnastics, all queuing QoS hierarchies must be rooted to a physical interface
- Runs in direct opposition to the concept for a logical Etherchannel interface since multiple interfaces need QoS in aggregate
- Aggregate GEC QoS creates a new hierarchy that is rooted on a logical recycle interface
- After packets run through the aggregate schedule, they are recycled for an abbreviated run through the PPEs for Etherchannel processing. Queued again through internally defined member-link hierarchies.

# IOS XE Aggregate GEC QoS

- Policy-maps can be attached to Port-channel main-interface for input and output directions
- No restrictions for contents of the policy-map other than what exist already for GigabitEthernet interfaces
- For example, all of the following are supported:
  - 3 levels of hierarchy (queuing + nonqueuing)
  - 2 levels of priority
  - WRED
  - fair-queue, plus other features

# IOS XE Aggregate GEC QoS

- Aggregate GEC member-links are configured internally with high and low priority queues.
- Traffic classified as high priority in the aggregate GEC policy, are enqueued through the high priority queues on the member-link hierarchies.
- Priority levels 1 and 2 in the user-defined policy-map on the Port-channel main-interface will use the same priority queue on the physical interface

# IOS XE Aggregate GEC QoS

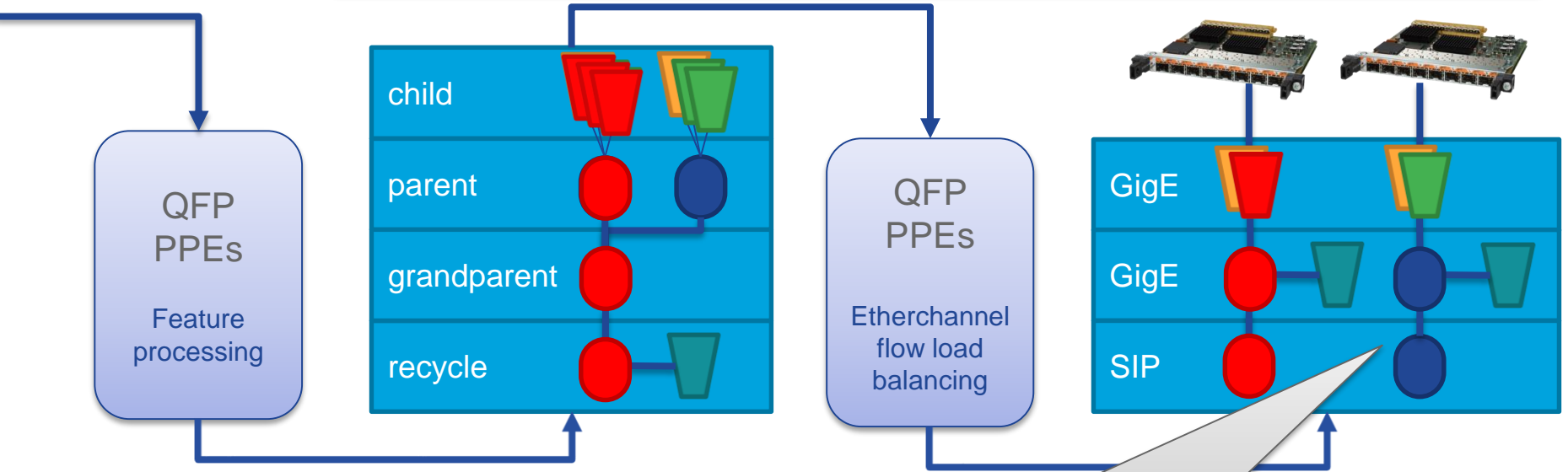
- If Port-channel does not have enough bandwidth to service all configured minimums, policy-map will go into suspended mode
  - For example, there are more than 1Gb/sec of minimums and a 2 member Port-channel interfaces has one of the member links go down
  - Once enough member links are available to provide the guaranteed minimums, service-policy will leave suspended mode and go into effect.
- Does not support TenGigabitEthernet interfaces (10 / 100 / 1000 only)

# IOS XE Aggregate GEC QoS

- If one of the member links is overdriven, it will exert back pressure on the aggregate hierarchy
  - Flow based load balancing could be lopsided and one of the member links is utilized at 100% and its egress queues fill up
  - When this happens, backpressure is exerted on the aggregate hierarchy and all traffic will be queued even if it is destined for an underutilized member link

# Aggregate GEC QoS backpressure

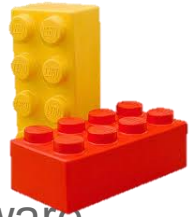
Feedback from GigE to grandparent should happen fast enough that the grandparent can slow down before GigE starts tail dropping. Current software does not meet this requirement for 10GE.



# *Tunnel QoS*



# Basic IOS XE tunnel QoS rules



- Tunnel hierarchies have to be rooted to a physical interface in the hardware
  - This is the source of most of the restrictions with tunnel QoS
- Traffic can only be queued once. So if there isn't magic glue to link together a queueing policy-map on the tunnel to the egress physical interface then the configuration is not supported.
  - The only supported case is physical interface with class-default only shaper
- **Classic IOS** supports a full queueing policy-map on the tunnel, and then another full queueing policy-map with user based queues on the physical interface.
  - Queues the packet twice
  - Classifies the packet twice

# Egress GRE tunnel QoS Details

## Interface and Tunnel Policy Combination Use Cases

	physical interface QoS policy with queuing actions	physical interface QoS policy without queuing actions
tunnel policy-map with queuing actions	Class-default only policy-map on physical interface supported. Packets will be managed by tunnel policy then rate limited by the interface policy (class-default only).	Tunnel packets bypass interface policy-map
tunnel policy-map without queuing actions	Tunnel packets go through tunnel policy-map fully and then through interface policy-map (class-default only)	Tunnel packets go through tunnel policy-map fully and then through interface policy-map (interface default queue). Interface policy-map has no effect. Traffic is only classified once in IOS XE.

- Maximum of two level policy-map hierarchies allowed on GRE tunnels.
- Encryption is executed prior to egress queuing.
  - In XE2.5 software and later, traffic is classified and prioritized by the tunnel QoS policy-map before being enqueued for cryptography. Packets marked as priority levels 1 and 2 will move through a high priority queue for the crypto hardware.

# Per SA QoS for DMVPN

- Single tunnel interface configuration with multiple QoS policies available
- Hub QoS hierarchies have the these
  - 2 level queuing hierarchy
  - Parent can have only class-default with shaper / bandwidth remaining
- Additional support for physical interface class-default only policy-map
- If egress interface changes, QoS on DMVPN tunnel stays intact after switching physical egress interfaces
- Supports up to 4000 DMVPN tunnels with QoS
- RP2 / 1001-X / 1002-X for this level of scale, other hardware varies
- Key feature for IWAN DMVPN headend along with adaptive QoS

# Per SA QoS for DMVPN config

```
policy-map child
  class voice
    police cir 100000
    priority
  class critical_data
    bandwidth remaining ratio 10
  class class-default
    bandwidth remaining ratio 1
!
policy-map parent-15meg
  class class-default
    shape average 15000000
    bandwidth remaining ratio 15
    service-policy child
!
policy-map parent-10meg
  class class-default
    shape average 10000000
    bandwidth remaining ratio 10
    service-policy child
!
policy-map gr-parent
  class class-default
    shape average 10000000
```

```
interface Tunnel 1
  ip nhrp map group 10meg service-policy output parent-10meg
  ip nhrp map group 15meg service-policy output parent-15meg
  ip nhrp map group 20meg service-policy output parent-20meg
!
interface GigabitEthernet0/0/0
  service-policy output gr-parent
```

Grandparent shaper is allowed on egress interface for DMVPN tunnels.

Client identifies a specific NHRP group and matching QoS policy is applied. Client is manually configured with the matching NHRP group.

# DSCP Tunnel Marking example

```
policy-map change_marking
  class class-default
    set precedence tunnel 6
!
interface Tunnel 1
  ip address 1.0.0.1 255.255.255.252
  tunnel source 20.0.0.1
  tunnel destination 30.0.0.2
  service-policy output change_marking
!
interface GigabitEthernet0/0/0
  ip address 30.0.0.1 255.255.255.0
```

Packet at ingress



Packet at egress

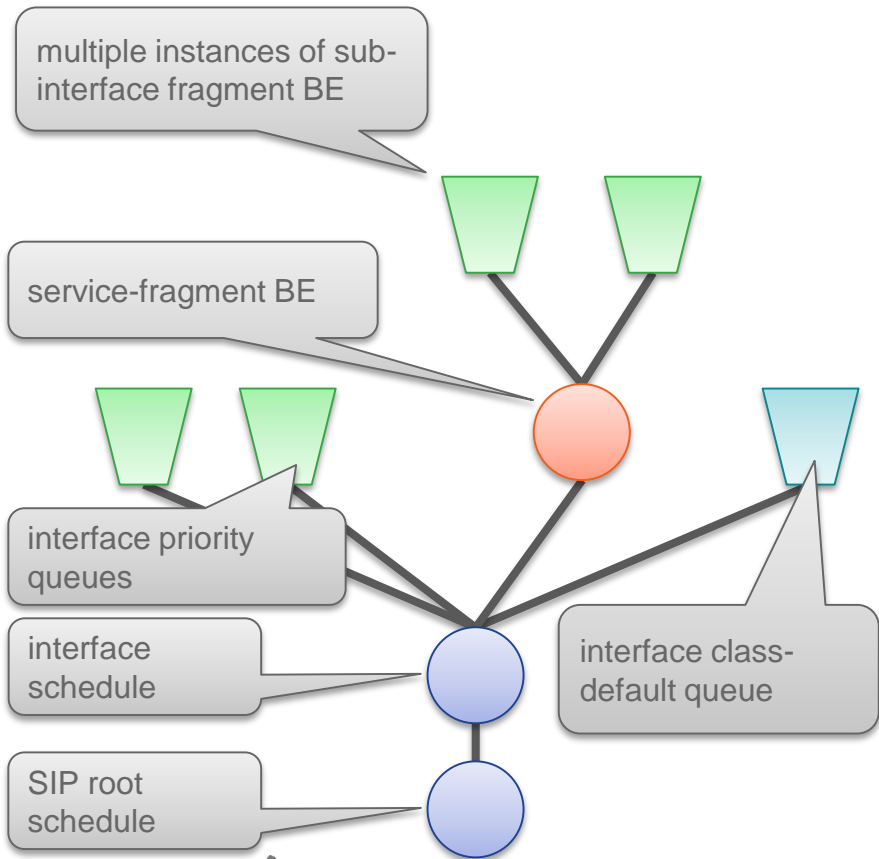


# *Service fragments*

# ASR 1000 QoS service-fragments



- Typically, hierarchies are very rigid with strict parent – child relationships
- Service-fragments allow QoS hierarchies which allow queues to be parented by an entity outside of the strict hierarchy
- Model 4
  - Allows queue conservation in scale configurations
- Model 3
  - Allows aggregation shaping of some traffic while having per session / sub-interface limits on other classes of traffic



Cisco *live!*

# ASR 1000 QoS service-fragments 4

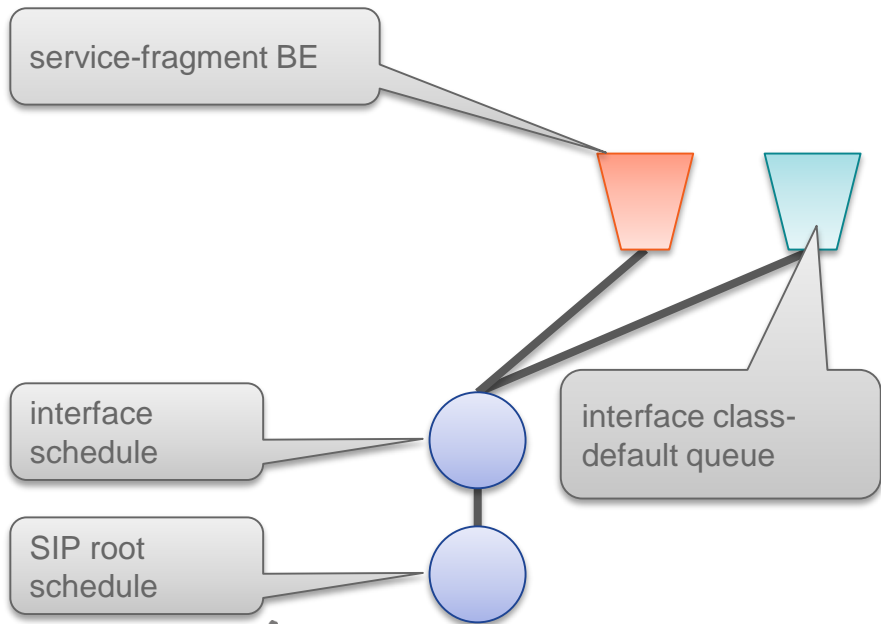
```

policy-map sub-int-mod4
  class EF
    police 1000000
  class AF4
    police 2000000
  class class-default fragment BE
    shape average 75000000
!
policy-map int-mod4
  class data service-fragment BE
    shape average 128000000
  class EF
    priority level 1
  class AF4
    priority level 2
  class AF1
    shape average 50000000
    random-detect
!
interface GigabitEthernet0/0/0
  service-policy output int-mod4
!
interface GigabitEthernet0/0/0.2
  service-policy output sub-int-mod4

```

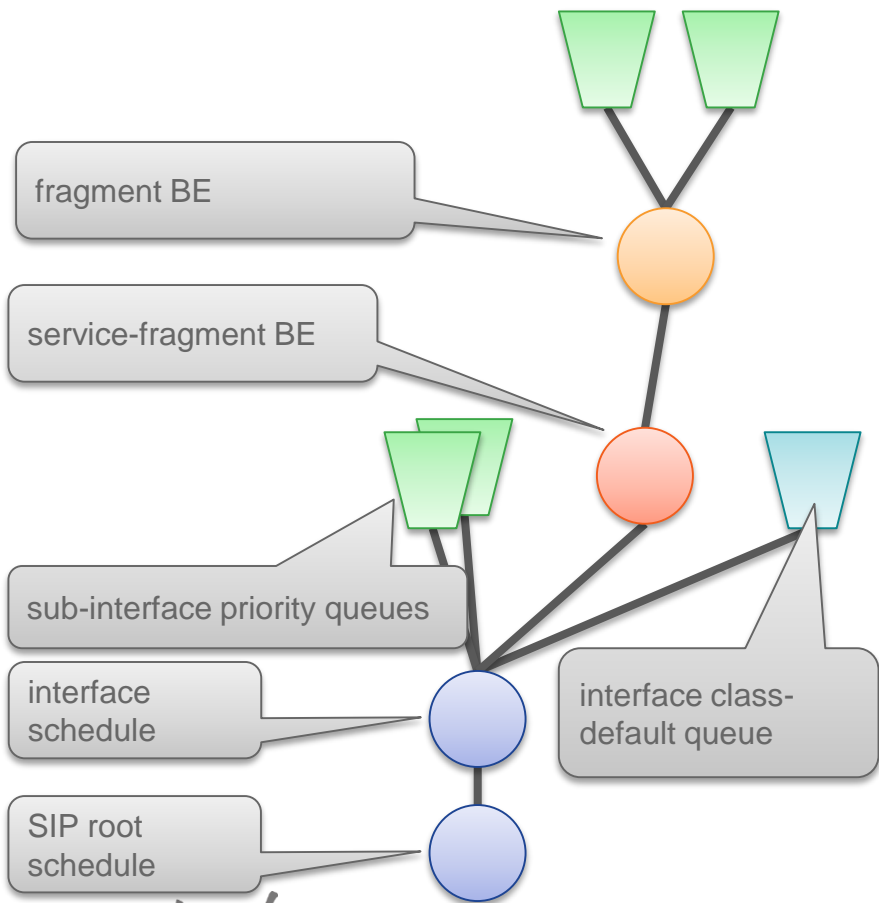


# ASR 1000 QoS service-fragments 3

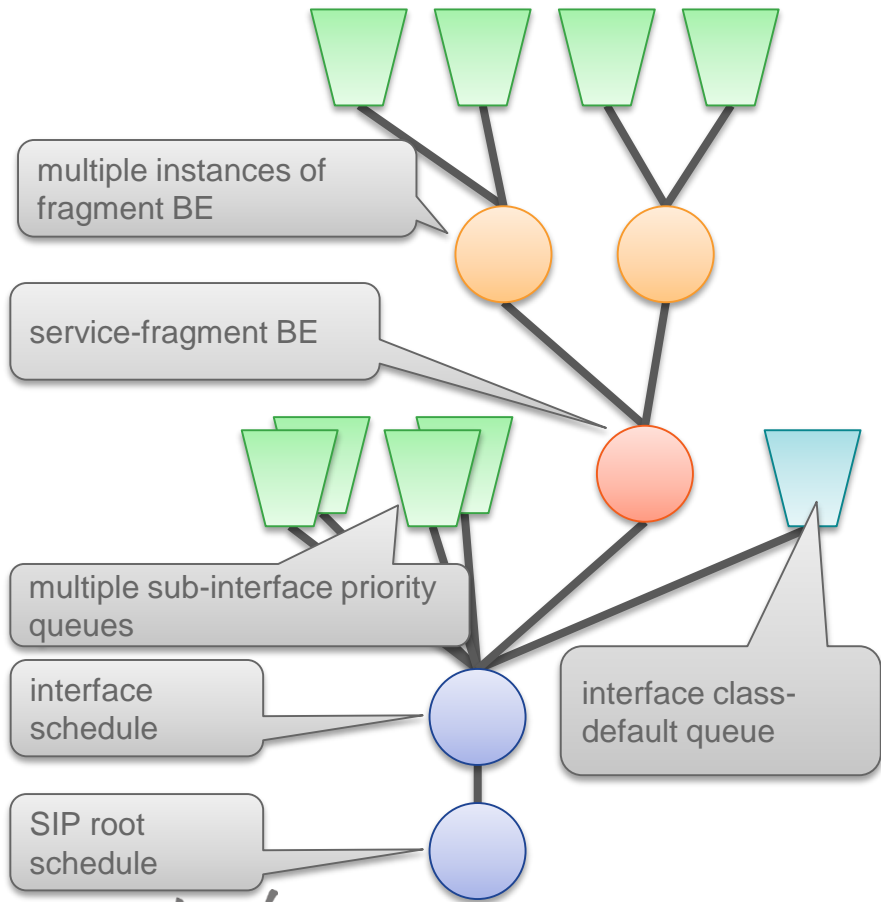


```
policy-map int-mod3
  class data service-fragment BE
    shape average 128000000
  class class-default
    shape average 100000000
!
policy-map sub-int-mod3
  class EF
    police 1000000
    priority level 1
  class AF4
    police 2000000
    priority level 2
  class class-default fragment BE
    shape average 115000000
    service-policy sub-int-child-mod3
!
policy-map sub-int-child-mod3
  class AF1
    shape average 100000000
  class class-default
    shape average 25000000
```

# ASR 1000 QoS service-fragments 3



```
policy-map int-mod3
  class data service-fragment BE
    shape average 128000000
  class class-default
    shape average 100000000
!
policy-map sub-int-mod3
  class EF
    police 1000000
    priority level 1
  class AF4
    police 2000000
    priority level 2
  class class-default fragment BE
    shape average 115000000
    service-policy sub-int-child-mod3
!
policy-map sub-int-child-mod3
  class AF1
    shape average 100000000
  class class-default
    shape average 25000000
```



Cisco *live!*

# ASR 1000 QoS service-fragments 3

```

policy-map int-mod3
  class data service-fragment BE
    shape average 128000000
  class class-default
    shape average 100000000
!
policy-map sub-interface-mod3
  class EF
    police 1000000
    priority level 1
  class AF4
    police 2000000
    priority level 2
  class class-default fragment BE
    shape average 115000000
    service-policy sub-int-child-mod3
!
policy-map sub-interface-child-mod3
  class AF1
    shape average 100000000
  class class-default
    shape average 25000000
  
```

# *Service groups*

# What are service-groups?



- Service-groups allow linking multiple L3 sub-interfaces and L2 service instances together for the purpose of aggregated QoS
- Before service-groups
  - QoS policies could be applied to individual L3 sub-interfaces, individual L2 service instances, or to ethernet main interfaces
  - In order to group multiple L3 or L2 entities together for QoS, a “mega-policy” on the main interface which classified multiple vlans in the topmost layer was required.
  - If various groups of vlans on the same physical interface required QoS, the configuration quickly became unmanageable.

# Support for service-groups

- Added in XE3.15, released March 2015
- Supported on ASR1000, CSR1000V, and ISR4000 series platforms
- Same functionality across all above platforms
- Same scalability for all platforms
  - No dependence on ASR1000 RP or ESP version
  - No dependence on ASR1000 fixed chassis version
  - Same scalability for ISR 4300 and ISR4400 platforms

# New configuration commands

```
policy-map alpha
  class-default
    shape average 10000000
!
interface GigabitEthernet0/0/0
  service instance 11 ethernet
  encapsulation dot1q 11
  group 10
  service instance 12 ethernet
  encapsulation dot1q 12
  group 10
!
interface GigabitEthernet0/0/0.13
  encapsulation dot1q 13
  group 10
!
interface GigabitEthernet0/0/0.14
  encapsulation dot1q 14
  group 10
!
service-group 10
  service-policy alpha
```

Use the `group` keyword to put service instances and sub-interfaces into a service-group.

Use the `service-group` command as the application point for QoS policies.

# Configuration

- Ingress and egress policy-maps are supported on service-groups
- Up to three levels in policy-maps (ingress and egress)
  - Same hierarchy restrictions as policy-maps applied to main-interfaces
- Support for all Ethernet interface speeds
  - 10 / 100 / 1000 / 10000
- No support for non-ethernet interface types
- Statistics are collected on a per service-group level and will not be available per service-group member unless explicitly classified as part of the service policy



# Restrictions

- All members of a given service-group must be on the same physical interface
- Service-groups are **not** supported on port-channel interfaces
  - Applies to both legacy or aggregate QoS port-channels
- A sub-interface or service instance can belong to only one service-group at time
- Sub-interfaces and service instances in a service-group can not have a policy-map applied other than on the service-group
- Tunnels with QoS egressing through service-group not supported

# Disallowed configuration examples

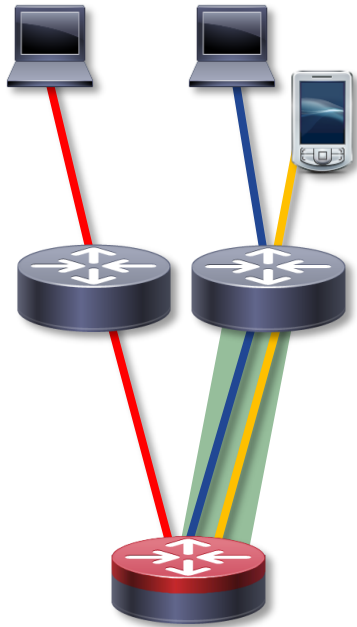
```
policy-map alpha
  class-default
    shape average 10000000
!
policy-map beta
  class-default
    shape average 10000000
!
interface GigabitEthernet0/0/0
  service instance 10 ethernet
  encapsulation dot1q 10
  service-policy alpha
  group 10
!
service-group 10
  service-policy beta
```

Policy-map is **not** allowed on sub-interface or service instance when included in a service-group.

```
policy-map alpha
  class-default
    shape average 10000000
!
policy-map beta
  class-default
    shape average 10000000
!
interface GigabitEthernet0/0/0.10
  encapsulation dot1q 10
  group 10
!
interface GigabitEthernet0/0/1.20
  encapsulation dot1q 20
  group 10
!
service-group 10
  service-policy beta
```

Service-group members can **not** span across multiple physical interfaces.

# Service groups use case



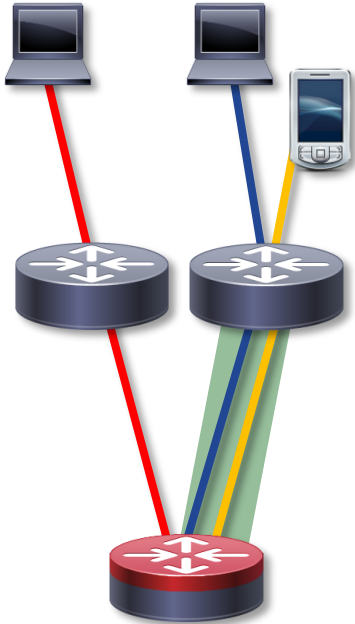
- **Left side branch**

- Traditional deployment with a single VLAN servicing the entire branch

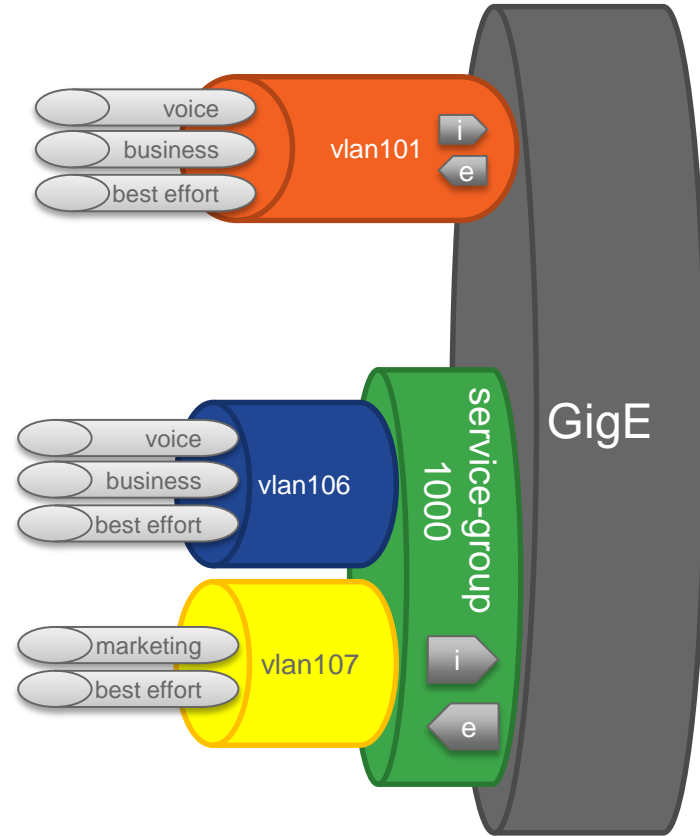
- **Right side branch**

- serviced by a single downlink across the WAN but uses VLANs (blue and yellow) to differentiate business traffic and customer BYOD traffic.
- From the headend ASR1000, it is necessary to rate limit traffic to the CPE as a whole but also make guarantees to the business class traffic over the BYOD traffic via vlan classification

# Service groups use case



```
policy-map service-group-1000
  class-default
    shape average 10000000
    service-policy right-branch
!
policy-map right-branch
  class-map vlan106
    bandwidth remaining ratio 20
    service-policy workers
  class-map vlan107
    bandwidth remaining ratio 1
    service-policy guests
!
policy-map workers
  class-map voice
    priority
    police cir 1000000
  class-map business
    bandwidth 8000
  class-map class-default
!
policy-map guests
  class-map marketing
    bandwidth remaining ratio 10
  class-map class-default
    bandwidth remaining ratio 1
```



Cisco *live!*

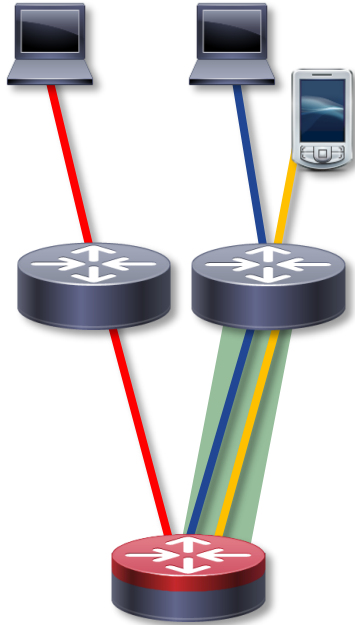
vlanX classes are basic match vlan X class-maps.

BRKARC-2031

© 2015 Cisco and/or its affiliates. All rights reserved. Cisco Public

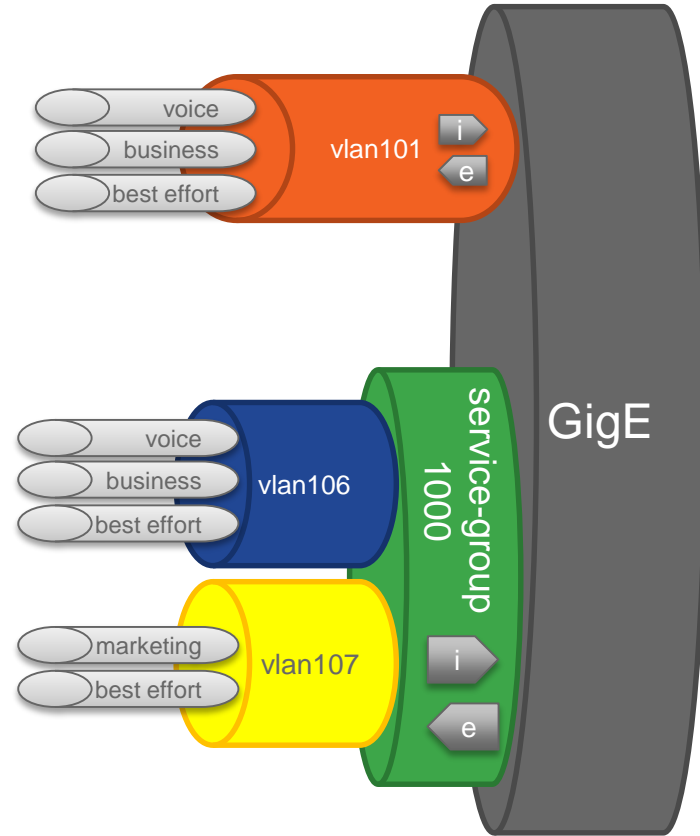
60

# Service groups use case



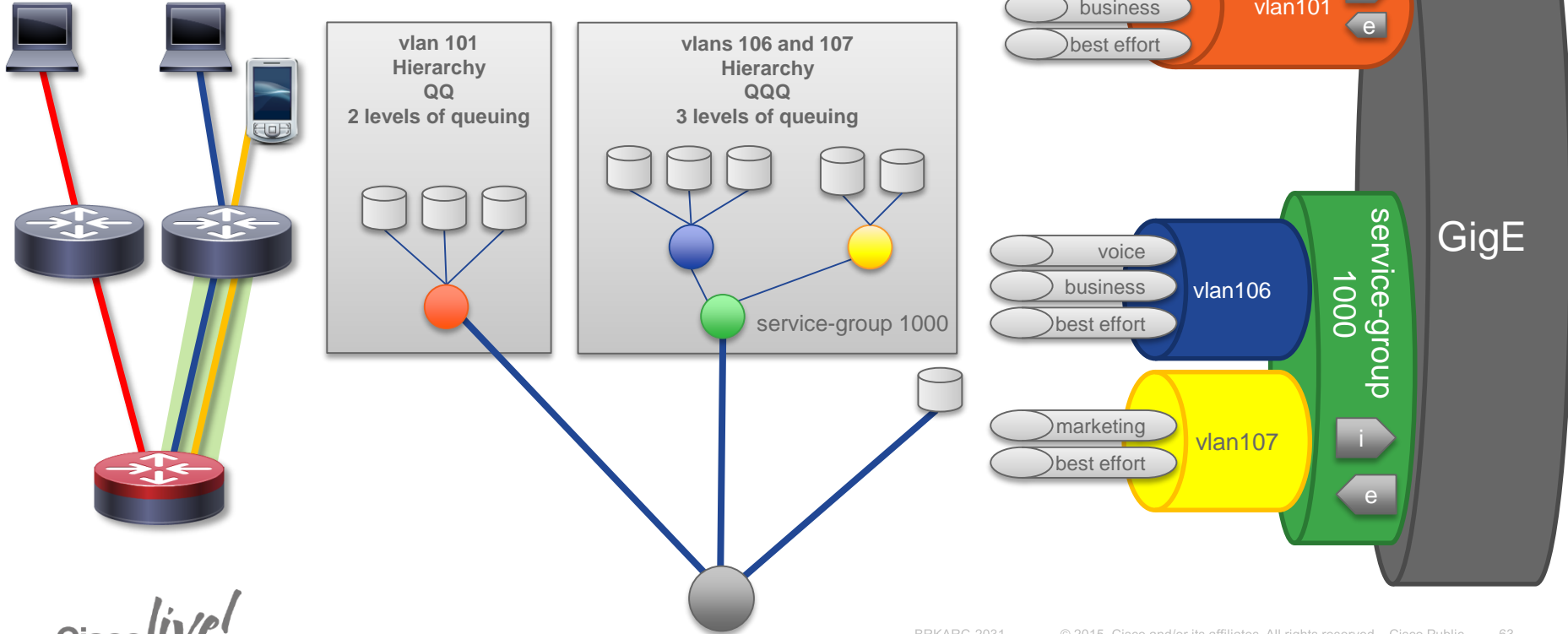
vlanX classes used in the following slides are basic match vlan X class-maps.

```
class-map match-any vlan101
  match vlan 101
!
class-map match-any vlan106
  match vlan 106
!
class-map match-any vlan107
  match vlan 107
!
class-map match-any vlan106-107
  match vlan 106
  match vlan 107
```





# Service groups use case



# Same setup without service groups

- An identical configuration without service-groups is not possible
- In order to build a QoS mega-policy on the main interface, it would require:
  - 3 levels of queuing
  - grandparent level that classifies based on vlan
  - previous two items items are mutually exclusive
- The following slide presents that **invalid** configuration.
- The permitted configuration would only allow 2 levels of queuing and then a third level that could mark or police traffic only



# Same setup without service groups

```

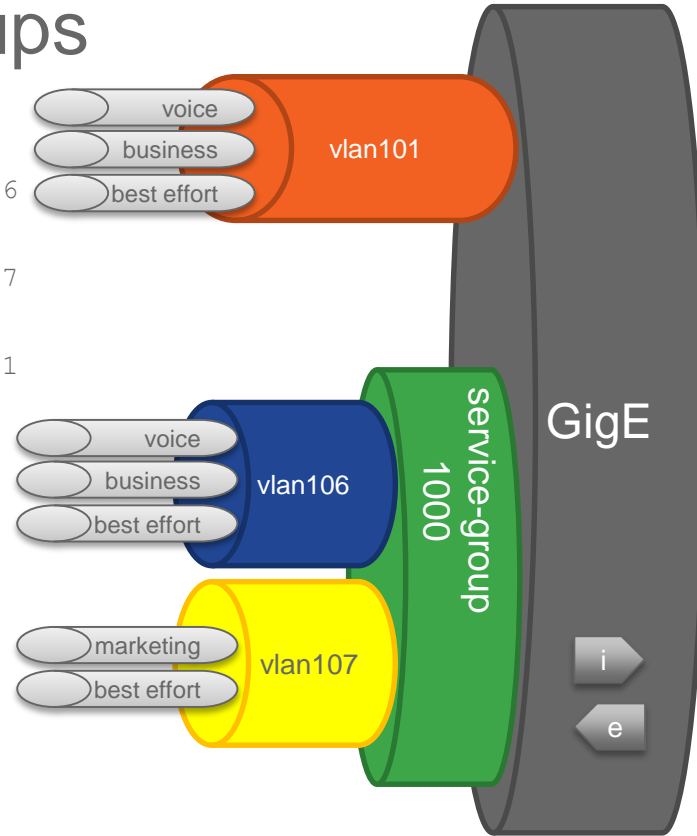
level 1
policy-map GigE
  class vlans-106-107
    shape average 10000000
    service-policy right-branch
  class vlan-101
    shape average 5000000
    service-policy workers
!
level 2
policy-map right-branch
  class-map vlan106
    bandwidth remaining ratio 20
    service-policy workers
  class-map vlan107
    bandwidth remaining ratio 1
    service-policy guests
!
level 3
policy-map workers
  class-map voice
    priority
    police cir 1000000
  class-map business
    bandwidth 8000
  class-map class-default
!
policy-map guests
  class-map marketing
    bandwidth remaining ratio 10
  class-map class-default
    bandwidth remaining ratio 1
  
```

```

interface Gig0/0/0.106
  encapsulation dot1q 106
!
interface Gig0/0/0.107
  encapsulation dot1q 107
!
interface Gig0/0/0.101
  encapsulation dot1q 101
!
interface Gig0/0/0
  service-policy GigE
  
```

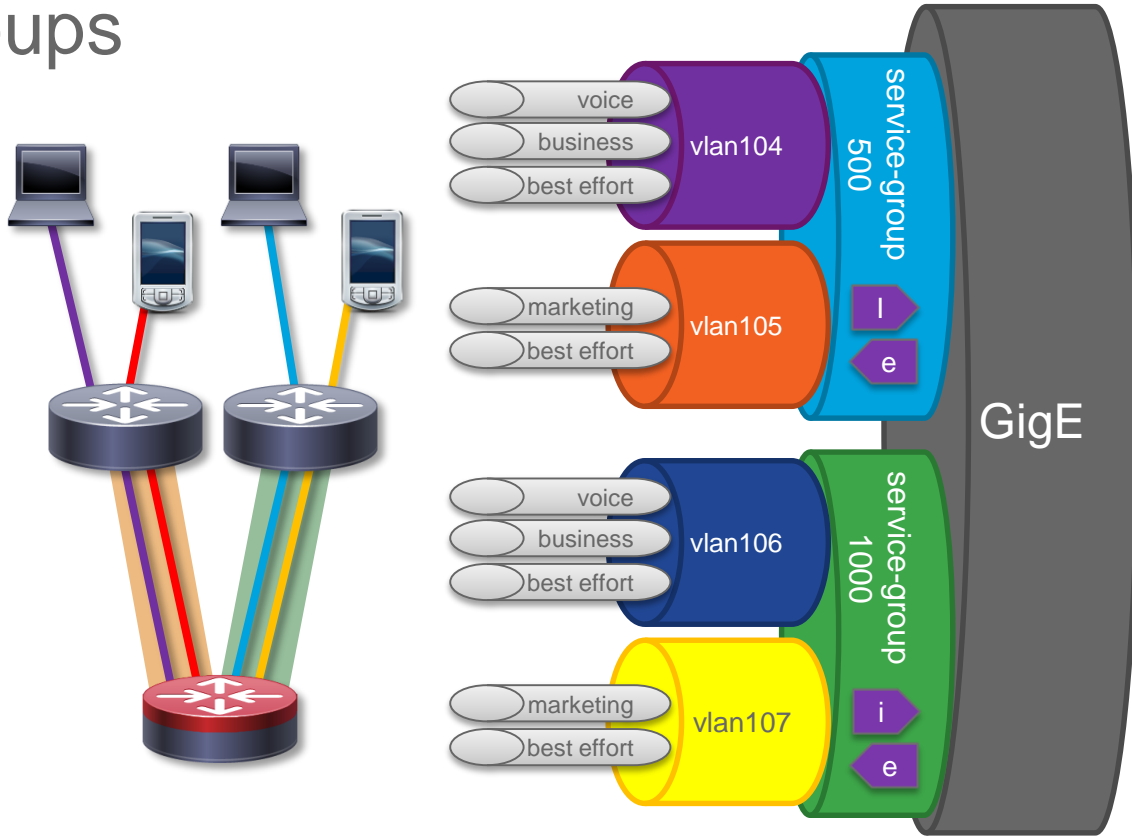
Note that this is a invalid configuration because the grandparent level is not class-default only.

vlanX classes are basic match vlan X class-maps.



# Multiple service groups

- Multiple service groups per physical interface are allowed
- A given service group is unique per chassis and can not be reused across multiple physical interfaces
- Service group numbers are only locally significant



# Scalability

- Up to 1000 service-groups per system
- Up to 1000 members of a single service group
- Up to 8000 sub-interfaces and service instances can be members of a service group per system

# New commands available

- `show running-config service-group`
- `show service-group {service-group-identifier | all}`
- `show service-group interface type number`
- `show service-group stats`
- `show service-group state`
- `show service-group traffic-stats`
- `show policy-map target service-group {service-group-identifier}`
- `show ethernet service instance detail`
- `clear service-group traffic-stats`

# New internal commands available

- `debug service-group {all | error | feature | group | interface | ipc | member | qos | stats}`
- `show platform software interface {rp | fp} active [brief]`
- `show platform hardware qfp active interface platform {service-group if name}`
- `show platform hardware qfp active feature qos interface-string {service-group if name}`

Acquire the `service-group if name` **with the** `show platform software interface` command.

# *Shaper parameters ignored*

# Shaper parameters ignored

```
IOSXE(config-pmap-c)#shape average ?  
<8000-10000000000> Target Bit Rate (bits/sec).  
percent % of interface bandwidth for Committed information rate
```

```
IOSXE(config-pmap-c)#shape average 200000000 ?  
<32-8000000000> bits per interval, sustained. Recommend not to configure, algo finds the best v  
account Overhead Accounting  
<cr>
```

```
IOSXE(config-pmap-c)#shape average 200000000 100000 ?  
<0-1544000000> bits per interval, excess.  
account Overhead Accounting  
<cr>
```

- IOS XE schedulers ignore the  $b_c$  and  $b_e$  parameters given with the shape command. Any special tuning done for classic IOS shapers will be lost when brought to IOS XE.
- Policer  $b_c$  and  $b_e$  parameters are used and behavior is the same as classic IOS.

# *Priority versus PAK\_PRI*



# What is PAK\_PRI

- Internally generated packets considered so important that they are always “no drop”
- Associated with protocols where reliable delivery is considered highly desirable
- Not all packets for a given protocol are considered PAK\_PRI

# How is PAK\_PRI handled

- PAK\_PRI packets will show up in QoS class classification stats but will be queued in the interface default queues (non-ATM)
  - Classification and queuing stats won't match
  - ATM interfaces have a default queue on a per VC basis and those are used
- PAK\_PRI packets are **never** subject to dropping
  - Only if physical packet memory is exhausted will PAK\_PRI be dropped
  - 5% of packet memory is reserved for PAK\_PRI packet only
- PAK\_PRI packets **are not treated with LLQ** unless classified into that class, and they will actually move through the low latency queue

# PAK\_PRI protocols

## Layers 1 and 2

- ATM Address Resolution Protocol Negative Acknowledgement (ARP NAK) o ATM ARP requests
- ATM host ping operations, administration and management cell(OA&M)
- ATM Interim Local Management Interface (ILMI)
- ATM OA&M
- ATM ARP reply
- Cisco Discovery Protocol
- Dynamic Trunking Protocol (DTP)
- Ethernet loopback packet
- Frame Relay End2End Keepalive
- Frame Relay inverse ARP
- Frame Relay Link Access Procedure (LAPF)
- Frame Relay Local Management Interface (LMI)
- Hot standby Connection-to-Connection Control packets (HCCP)
- High-Level Data Link Control (HDLC) keepalives
- Link Aggregation Control Protocol (LACP) (802.3ad)
- Port Aggregation Protocol (PAgP)
- PPP keepalives
- Link Control Protocol (LCP) Messages
- PPP LZS-DCP
- Serial Line Address Resolution Protocol (SLARP)
- Some Multilink Point-to-Point Protocol (MLPP) control packets (LCP)

## IPv4 Layer 3

- Protocol Independent Multicast (PIM) hellos
- Interior Gateway Routing Protocol (IGRP) hellos
- OSPF hellos
- EIGRP hellos
- Intermediate System-to-Intermediate System (IS-IS) hellos, complete sequence number PDU (CSNP), PSNP, and label switched paths (LSPs)
- ISIS hellos
- Triggered Routing Information Protocol (RIP) Ack o TDP and LDP hellos
- Resource Reservation Protocol (RSVP)
- Some L2TP control packets o Some L2F control packets o GRE IP Keepalive
- IGRP CLNS
- Bidirectional Forwarding Protocol (BFD)

## IPv6 Layer 3

- Miscellaneous protocols

This list is not considered to be complete nor exhaustive and is subject to change in various versions of software.

# *Classic to IOS XE QoS behavior differences*

# Changes in behavior – classic to IOS XE



Reference

	Classic IOS	IOS XE
Priority (naked) priority	Can consume up to 99% of parent bandwidth <b>up to min guarantee of parent.</b>	Can consume up to 100% of parent bandwidth. Class-default and user classes can be completely starved. <b>No parent min guarantee limit.</b>
Priority (naked) priority	Cannot be used in the same policy-map with bandwidth.	
priority oversubscription	Allowed via tunnel / sub-interface config, not in a given policy-map	
Police (with or without priority) police cir percent X	Rate is based on parent's maximum rate.	
Conditional priority priority percent X	Rate is based on lesser of <b>max</b> or <b>min</b> for parent. Policer is activated if parent node or physical interface is congested.	Rate is based on <b>max</b> rate of parent. Policer is activated if parent node or physical interface is congested.

# Changes in behavior – classic to IOS XE

	Classic IOS	IOS XE
Unallocated bandwidth (min configuration)	All granted to class default	Equally spread amongst all non-priority classes
bandwidth	Can be used at any level	Can only be used at the leaf level of a hierarchy.
bandwidth remaining percent X (excess configuration)	Rate is based on lesser of parent's <b>maximum</b> or <b>minimum</b> rate.	Rate is based on parent's maximum rate minus any priority class minimums satisfied.
Unallocated bandwidth remaining percent X (excess configuration)	Allocated equally amongst classes with no excess configuration.  If all classes have excess config, granted proportionally amongst all excess configured classes.	

# Changes in behavior – classic to IOS XE

	Classic IOS	IOS XE
Multiple policy-maps in egress path (MPOL)	Supported  Queuing policy-map with classification on Tunnel and then again on egress physical interface	Generally no.  Certain restricted configurations are supported with hierarchical policy-map on sub-interfaces, tunnels, etc. with class-default only shaper on main-interface.
shape burst parameters	Defaults to 1 second of bursting. <code>bc</code> and <code>be</code> parameters are adjustable.	Defaults to 1 ms of burst time. <code>bc</code> and <code>be</code> parameters are accepted on the CLI but have no affect on operation.
<code>fair-queue</code> scalability	Up to 4096 fair queues per class, configurable.	16 fair queues per class. Scalability not configurable.

# *Platform specific caveats*

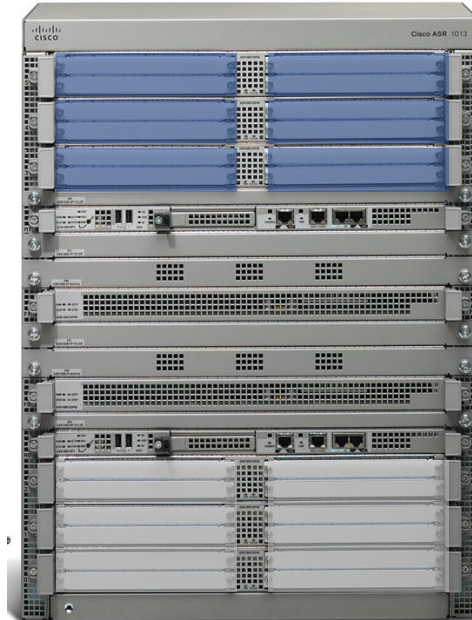


# QFP queue distribution for ESP100 and ESP200

- QFP 3
- QFP 2
- QFP 1
- QFP 0



ASR1006 with  
ESP100 and SIP40



ASR1013 with  
ESP100 and SIP40



ASR1013 with  
ESP200 and SIP40

# ASR1000 ESP specific QoS specifications

Card / Chassis	Packet memory	Max Queues	TCAM
ASR 1001	64 MB	16,000	5 Mb
ASR 1001-X	512 MB	16,000	10 Mb
ASR 1002-X	512 MB	116,000	40 Mb
ESP-5	64 MB	64,000	5 Mb
ESP-10	128 MB	128,000	10 Mb
ESP-20 and ESP-40	256 MB	128,000	40 Mb
ESP-100	1G (512 MB x 2)	232,000 *	80 Mb (40 Mb x 2)
ESP-200	2G (512 MB x 4)	464,000 *	80 Mb (40 Mb x 2)

\* Queues must be distributed amongst physical locations in the chassis.

# ISR4000 QoS specifications

Card / Chassis	Packet memory	Max Queues	TCAM
ISR4321	Variable	8,000	Software lookup
ISR4331	Variable	16,000 *	Software lookup
ISR4351	Variable	16,000 *	Software lookup
ISR4431	Variable	16,000 *	Software lookup
ISR4451	Variable	16,000 *	Software lookup

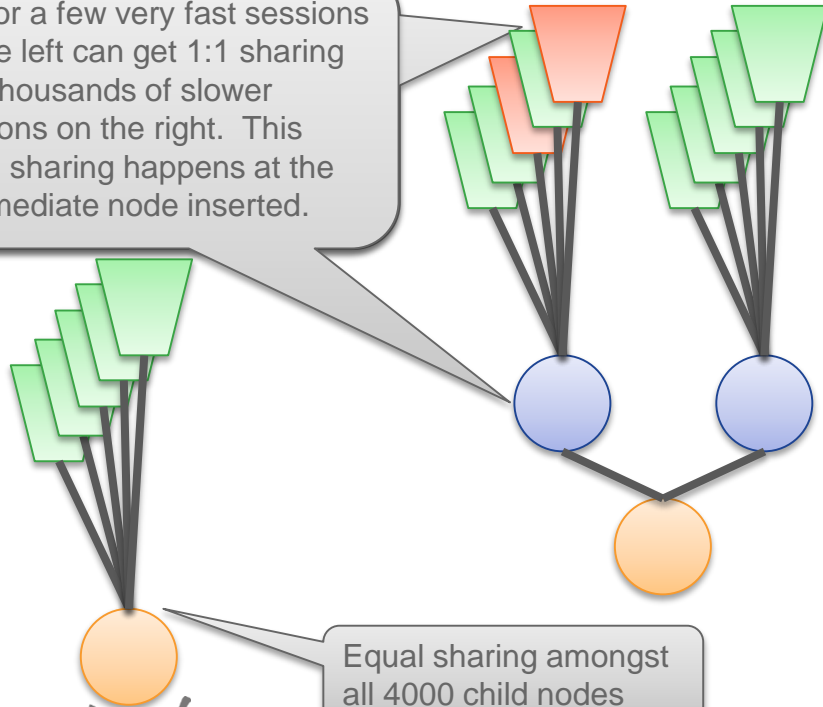
\* Current tested maximum, scale could reach 32,000 with additional validation .

# 4000 node aggregation

- On platforms using the 2<sup>nd</sup> generation and newer ASICs (1001-X, 1002-X, ESP100, ESP200), there is a limit of 4000 children per QoS node
- This difference does not impact 99% of enterprise configurations
- Broadband configurations most likely affected.
- Any location which has more than 4000 children will be split into two nodes that equally divide the children in a random fashion.
- No affect on rate limiting, but there is a impact on bandwidth sharing during congestion

# 4000 node aggregation

One or a few very fast sessions on the left can get 1:1 sharing with thousands of slower sessions on the right. This equal sharing happens at the intermediate node inserted.



Equal sharing amongst all 4000 child nodes

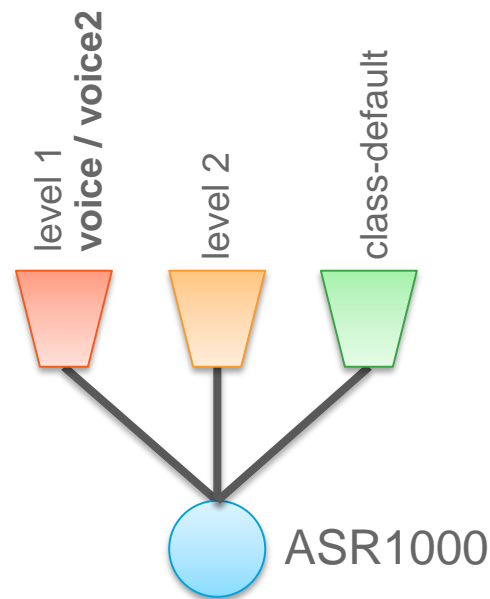
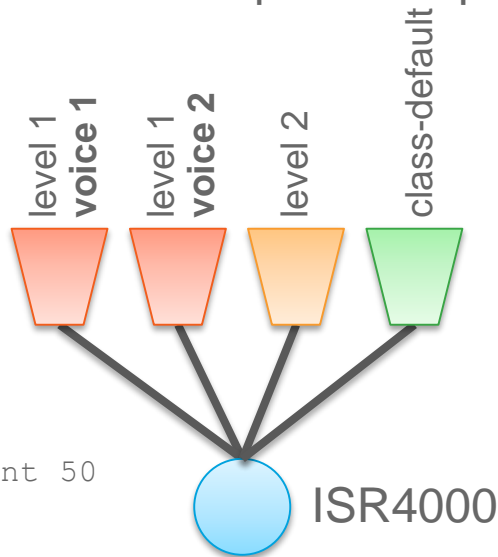
Cisco *live!*

- If the bottommost node in each hierarchy is congested sharing (excess) must be used to allocate bandwidth.
- The scenario on the right will give equal sharing amongst all.
- The left scenario will allow the two sessions in red to get up to 50% of the bandwidth causing multiple sessions on the right node to suffer.
- **Workaround:** distribute sessions across multiple ethernet sub-interfaces with bandwidth remaining ratio configuration

# Independent and consolidated priority queues

- ASR1000 has consolidated priority queues for all classes at level 1 and level 2
- ISR4000 and CSR1000V have independent queues

```
policy-map child
  class voice
    priority level 1
    police cir 1000000
  class voice2
    priority level 1
    police cir 1000000
  class video
    priority level 2
    police cir 5000000
  class other
    bandwidth remaining percent 50
  class class-default
```



# *Use case migrations*

# Use case migrations

- Microbursts – queue limit tuning
- Configurations using bandwidth



# New unexplained drops

What's with all these drops on a better / faster box?

- Previously installed 7200s with QoS worked fine. When we replaced the boxes with ASR1000s we started to see tons of drops on the egress WAN interfaces. Average utilization via CLI and SNMP never shows interface utilization high enough to cause the parent shaper to drop traffic.
- With all this fancy QoS and packet buffer memory why do we start of have these problems now?



# New unexplained drops

What's with all these drops on a better / faster box?

```
policy-map EGRESS
  class VoIP
    priority percent 15
  class PRIORITY_DATA
    bandwidth remaining percent 30
  class MISSION_CRITICAL_DATA
    bandwidth remaining percent 40
  class SCAVENGER_DATA
    bandwidth remaining percent 1
    random-detect
  class class-default
    bandwidth remaining percent 3
!
policy-map SHAPE_730Mbps
  class class-default
    shape average 730000000
    service-policy EGRESS
```

```
Class-map: MISSION_CRITICAL_DATA (match-any)
  24718716075 packets, 21181619094739 bytes
  30 sec offered rate 237687000 bps, drop rate 262000 bps
Match: ip dscp cs3
Queueing
  queue limit 3041 packets
  (queue depth/total drops/no-buffer drops) 0/324178/0
  (pkts output/bytes output) 24718391897/21181154463832
  bandwidth remaining 12%
```

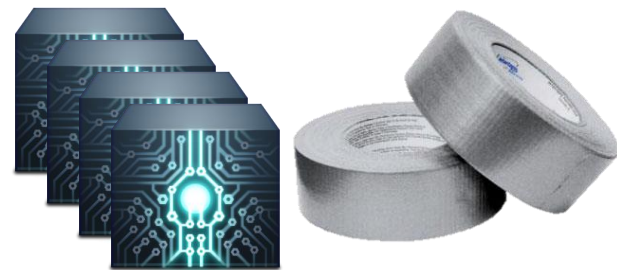
**MISSION\_CRITICAL\_DATA** should always have access to at least  $730E6 * 85% * 40% = 248 \text{ Mbit/sec}$ .

This class is running close to that on a 30 second average which almost promises that there are going to be microbursts higher than that.

Any 10GE interfaces that feed into this 730Mb/sec uplink would amplify that affect over GigE interfaces.

# Microbursts

What's with all these drops on a better / faster box?



- Newer platforms can forward traffic truly at line rate
  - No delays with ingress buffering or delays
  - No delays with packet processing with multiple cores servicing traffic
- Older platforms were single threaded and processed packets more or less serially compared to in parallel on the newer boxes
- When bursty traffic arrived on older platforms, those bursts were smoothed in the forwarding phase by the ingress buffering and serial processing before hitting egress QoS queuing code.
- Newer platforms do not have those “smoothers” and need more egress buffering to compensate. Sometimes the default buffers of for 50ms just are not enough.

# Microbursts

What's with all these drops on a better / faster box?



Fix is 3 fold

1. Realize that the function of QoS is **not** to preserve all packets. It is to prioritize traffic, rate limit, and provide minimum guarantees under congestion
2. Increase queue limits to deal with bursty conditions  
Consider the use of time or byte based queue-limits to have true max latency protection and maximum capacity to absorb bursts
3. Use policer markdown with WRED to tamp down TCP bursting if applicable  
still drops traffic but uses better management

# Microbursts – increase queue limits

What's with all these drops on a better / faster box?



```
policy-map EGRESS
  class VoIP
    priority percent 15
  class PRIORITY_DATA
    bandwidth remaining percent 30
    queue-limit 100 ms -or- queue-limit 6083
  class MISSION_CRITICAL_DATA
    bandwidth remaining percent 40
    queue-limit 150 ms -or- queue-limit 9125
  class SCAVENGER_DATA
    bandwidth remaining percent 1
    random-detect
    queue-limit 200 ms -or- queue-limit 12166
  class class-default
    bandwidth remaining percent 3
    queue-limit 200 ms -or- queue-limit 12166
!
policy-map SHAPE_730Mbps
  class class-default
    shape average 730000000
    service-policy EGRESS
```

```
policy-map EGRESS
  class VoIP
    priority percent 15
  class PRIORITY_DATA
    bandwidth remaining percent 30
    queue-limit 6083
  class MISSION_CRITICAL_DATA
    bandwidth remaining percent 40
    queue-limit 9125
  class SCAVENGER_DATA
    bandwidth remaining percent 1
    queue-limit 12166
    random-detect
    police cir 7300000 pir 14600000
    conform-action transmit
    exceed-action transmit-set-dscp af12
    violate-action transmit-set-dscp af13
  class class-default
    bandwidth remaining percent 3
    queue-limit 12166
```

If these deep queues are constantly full, you have a bandwidth problem and not a bursting problem.

# Bandwidth configurations

- Due to the 2 vs 3 parameter scheduler changes, minor tweaks are necessary to keep the same behavior with excess sharing

```
policy-map child
  class voice
    priority percent 15
  class BUSINESS_APP_1
    bandwidth percent 30
  class BUSINESS_APP_2
    bandwidth percent 40
  class SCAVENGER
    bandwidth percent 1
    random-detect
  class class-default
    bandwidth percent 3
!
policy-map PARENT
  class class-default
    shape average 50000000
    service-policy child
```

```
policy-map child
  class voice
    priority percent 15
  class BUSINESS_APP_1
    bandwidth remaining percent 30
  class BUSINESS_APP_2
    bandwidth remaining percent 40
  class SCAVENGER
    bandwidth remaining percent 1
    random-detect
  class class-default
    bandwidth remaining percent 3
!
policy-map PARENT
  class class-default
    shape average 50000000
    service-policy child
```

# *Conclusion*

# Conclusion

- Multicore forwarding is great, but....
  - It exposes your network to bursty traffic. Be armed with knowledge to explain and the tools to handle it.
- More is better, with parameters that is...
  - Just be sure to understand how the excess bandwidth is shared evening vs. proportionally between configurations
- All these rules...
  - With hardware forwarding comes a bit more rigidity. Know which use cases are covered, where the gaps are, and who to yell at to cover use cases that are missing



# Conclusion

- Have QoS and Etherchannel make nice
  - It's finally possible to do aggregate queueing across flow based load balanced interfaces.
- Be aware of new features like service-groups and service-fragments for innovative designs
- WAN aggregate and branch routers singing "We are family!"
  - By using ASR1000 for WAN aggregation along with ISR4000 routers in the branch you can expect the same behavior at both places in the network with identical configurations.



# Continue Your Education

- Demos in the Cisco campus
- Walk-in Self-Paced Labs
- Table Topics
- Meet the Engineer 1:1 meetings
- Related sessions

*Thank you*



**CISCO**

*TOMORROW starts here.*