- QoS Classification and Marking -

Classifying and Marking Traffic

Conceptually, DiffServ QoS involves three steps:

- Traffic must be *identified* and then *classified* into groups.
- Traffic must be *marked* on trust boundaries.
- *Policies* must be created to describe the *per-hop behavior* for classified traffic.

DiffServ QoS relies on the **classification** of traffic, to provide differentiated levels of service on a per-hop basis. Traffic can be classified based on a wide variety of criteria called **traffic descriptors**, which include:

- Type of application
- Source or destination IP address
- Incoming interface
- Class of Service (CoS) value in an Ethernet header
- Type of Service (ToS) value in an IP header (*IP Precedence* or *DSCP*)
- MPLS EXP value in a MPLS header

Access-lists can be used to *identify* traffic for classification, based on address or port. However, a more robust solution is Cisco's **Network-Based Application Recognition (NBAR)**, which will dynamically recognize standard or custom applications, and can classify based on payload.

Once classification has occurred, traffic should be **marked**, to indicate the required *level* of QoS service for that traffic. **Marking** can occur within either the *Layer-2* header or the *Layer-3* header.

The point on the network where traffic is classified and marked is known as the **trust boundary.** QoS marks originating from *outside* this boundary should be considered *untrusted*, and removed or changed. As a general rule, traffic should be marked as *close to the source* as possible. In VoIP environments, this is often accomplished on the VoIP phone itself. Traffic classification should not occur in the network core.

Configuring DiffServ QoS on IOS devices requires three steps:

- Classify traffic using a **class-map.**
- Define a QoS policy using a **policy-map.**
- Apply the policy to an interface, using the **service-policy** command.

* * *

Layer-2 Marking

Layer-2 marking can be accomplished for a variety of frame types:

- Ethernet – using the 802.1p Class of Service (CoS) field.
- Frame Relay – using the Discard Eligible (DE) bit.
- ATM - using the Cell Loss Priority (CLP) bit.
- MPLS - using the EXP field.

Marking Ethernet frames is accomplished using the 3-bit **802.1p Class of Service (CoS)** field. The CoS field is part of the 4-byte *802.1Q* field in an Ethernet header, and thus is only available when 802.1Q VLAN frame tagging is employed. The CoS field provides 8 priority values:

Type	***Decimal***	***Binary***	***General Application***
Routine	0	000	*Best effort* forwarding
Priority	1	001	Medium priority forwarding
Immediate	2	010	High priority forwarding
Flash	3	011	VoIP call signaling forwarding
Flash-Override	4	100	Video conferencing forwarding
Critical	5	101	VoIP forwarding
Internet	6	110	Inter-network control (*Reserved*)
Network Control	7	111	Network control (*Reserved*)

Frame Relay and ATM frames provide a less robust marking mechanism, compared to the Ethernet CoS field. Both Frame Relay and ATM frames reserve a 1-bit field, to prioritize which traffic should be dropped during periods of congestion.

Frame Relay identifies this bit as the **Discard Eligible (DE) field**, while ATM refers to this bit as the **Cell Loss Priority (CLP) field**. A value of *0* indicates a *lower likelihood* to get dropped, while a value of *1* indicates a *higher likelihood* to get dropped.

MPLS employs a **3-bit EXP (Experimental) field** within the 4-byte MPLS header. The EXP field provides similar QoS functionality to the Ethernet CoS field.

* * *

Layer-3 Marking

Layer-3 marking is accomplished using the 8-bit **Type of Service (ToS)** field, part of the IP header. A mark in this field will remain *unchanged* as it travels from hop-to-hop, unless a Layer-3 device is explicitly configured to overwrite this field.

There are two marking methods that use the ToS field:

- **IP Precedence** - uses the first three bits of the ToS field.
- **Differentiated Service Code Point (DSCP)** – uses the first six bits of the ToS field. When using DSCP, the ToS field is often referred to as the **Differentiated Services (DS)** field.

These values determine the **per-hop behavior (PHB)** received by each classification of traffic.

IP Precedence

IP Precedence utilizes the **first three bits** (for a total of **eight values**) of the ToS field to identify the *priority* of a packet. Packets with a higher IP Precedence value should be provided with a better level of service. IP Precedence values are comparable to Ethernet CoS values:

Type	***Decimal***	***Binary***	***General Application***
Routine	0	000	*Best effort* forwarding
Priority	1	001	Medium priority forwarding
Immediate	2	010	High priority forwarding
Flash	3	011	VoIP call signaling forwarding
Flash-Override	4	100	Video conferencing forwarding
Critical	5	101	VoIP forwarding
Internet	6	110	Inter-network control (*Reserved*)
Network Control	7	111	Network control (*Reserved*)

By default, all traffic has an IP Precedence of 000 (*Routine*), and is forwarded on a best-effort basis.

Normal network traffic should not (and in most cases, *cannot*) be set to 110 (*Inter-Network Control*) or 111 (*Network Control*), as it could interfere with critical network operations, such as STP calculations or routing updates.

* * *

Differentiated Service Code Point (DSCP)

DSCP utilizes the **first six bits** of the ToS header to identify the *priority* of a packet. The first three bits identify the **Class Selector** of the packet, and is backwards compatible with IP Precedence. The following three bits identify the **Drop Precedence** of the packet.

Class Name	*Binary*	*Class Selector*	*Drop Precedence*
Default	000 000	0	
AF11	001 010	1	Low
AF12	001 100		Medium
AF13	001 110		High
AF21	010 010	2	Low
AF22	010 100		Medium
AF23	010 110		High
AF31	011 010	3	Low
AF32	011 100		Medium
AF33	011 110		High
AF41	100 010	4	Low
AF42	100 100		Medium
AF43	100 110		High
EF	101 110	5	

DSCP identifies six Class Selectors for traffic (numbered 0 - 5). **Class 0** is default, and indicates *best-effort* forwarding. Packets with a higher Class value should be provided with a better level of service. **Class 5** is the highest DSCP value, and should be reserved for the most sensitive traffic.

Within each Class Selector, traffic is also assigned a **Drop Precedence**. Packets with a *higher* Drop Precedence are **more likely** to be dropped during congestion than packets with a *lower* Drop Precedence. Remember that this is applied only *within* the same Class Selector.

The Class Name provides a simple way of identifying the DSCP value. **AF** is short for **Assured Forwarding,** and is the type of service applied to Classes 1 – 4. If a packet is marked AF23, then the Class Selector is **2** (the *2* in 23) and its Drop Precedence is High (the *3* in 23).

Packets marked as Class 0 (Default) or Class 5 (**Expedited Forwarding** or **EF**) do not have a Drop Precedence.

* * *

Modular QoS CLI (MQC)

The **Modular QoS CLI (MQC)** is an improved command-line implementation of QoS that replaced legacy CLI commands on IOS devices. MQC is considered *modular* because it separates classification from policy configurations.

There are three steps to configuring QoS using MQC:

- Classify traffic using a **class-map.**
- Define a QoS policy using a **policy-map.**
- Apply the policy to an interface, using the **service-policy** command.

Classifying and Marking Traffic using MQC

Traffic is classified using one or more of the **traffic descriptors** listed earlier in this guide. This is accomplished using the **class-map** command:

Router(config)# *access-list 101 permit tcp any 10.1.5.0 0.0.0.255 eq www*

Router(config)# *class-map match-any LOWCLASS*
Router(config-cmap)# *match access-group 101*

The *access-list* matches all *http* traffic destined for *10.1.5.0*/24.

The *class-map* command creates a new classification named *LOWCLASS.* The *match-any* parameter dictates that traffic can match *any* of the traffic descriptors within the class-map. Alternatively, specifying *match-all* dictates that traffic must match *all* of the descriptors within the class-map.

Within the class-map, *match* statements are used to identify specific traffic descriptors. The above example (*match access-group*) references an access-list. To match other traffic descriptors:

Router(config)# *class-map match-any HICLASS*
Router(config-cmap)# *match input-interface fastethernet0/0*
Router(config-cmap)# *match ip precedence 4*
Router(config-cmap)# *match ip dscp af21*
Router(config-cmap)# *match any*

The above is *not* a comprehensive list of descriptors that can be matched. Reference the link below for a more complete list.

(Reference: http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfmcli2.html)

* * *

Network-Based Application Recognition (NBAR)

Cisco's **Network-Based Application Recognition (NBAR)** provides an alternative to using static access-lists to identify protocol traffic for classification. NBAR introduces three key features:

- Dynamic protocol discovery
- Statistics collection
- Automatic traffic classification

NBAR provides classification abilities beyond that of access-lists, including:

- Ability to classify services that use dynamic port numbers. This is accomplished using the stateful inspection of traffic flows.
- Ability to classify services based on sub-protocol information. For example, NBAR can classify HTTP traffic based on payload, such as the host, URL, or MIME type.

NBAR employs a **Protocol Discovery** process to determine the application traffic types traversing the network. The Protocol Discovery process will then maintain statistics on these traffic types.

NBAR recognizes applications using NBAR **Packet Description Language Modules (PDLMs)**, which are stored in flash on IOS devices. Updated PDLMs are provided by Cisco so that IOS devices can recognize newer application types.

NBAR has specific requirements and limitations:

- NBAR requires that Cisco Express Forwarding (CEF) be enabled.
- NBAR does not support Fast EtherChannel interfaces.
- NBAR supports only 24 concurrent host, URL, or MIME types.
- NBAR can only analyze the first 400 bytes of a packet. Note: This restriction is only for IOS versions previous to 12.3(7), which removed this restriction.
- NBAR cannot read sub-protocol information in secure (encrypted) traffic types, such as HTTPS.
- NBAR does not support fragmented packets.

(Reference Reference: CCNP ONT Official Exam Certification Guide. Amir Ranjbar. Pages 110-112: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/ps6612/ps6653/prod_qas09186a00800a3ded.pdf)

* * *

Configuring NBAR

To enable NBAR Protocol Discovery on an interface:

Router(config)# *ip cef*

Router(config)# *interface fa0/0*
Router(config-if)# *ip nbar protocol-discovery*

To view statistics for NBAR-discovered protocol traffic:

Router# *show ip nbar protocol-discovery*

```
FastEthernet0/0
                        Input                    Output
                        -----                    ------
Protocol                Packet Count             Packet Count
                        Byte Count               Byte Count
                        30sec Bit Rate           30sec Bit Rate
                        30sec Max Bit Rate       30sec Max Bit Rate
----------------------- ------------------------ ------------------
http                    15648                    15648
                        154861743                154861743
                        123654                   123654
                        654123                   654123
ftp                     4907                     4907
                        954604255                954604255
                        406588                   406588
                        1085994                  1085994
```

NBAR classification occurs within a MQC class-map, using the *match protocol* command:

Router(config)# *class-map match-any LOWCLASS*
Router(config-cmap)# *match protocol http*
Router(config-cmap)# *match protocol ftp*

Matching traffic based on sub-protocol information supports wildcards:

Router(config)# *class-map match-any HICLASS*
Router(config-cmap)# *match protocol http host *routeralley.com**
Router(config-cmap)# *match protocol http mime "*pdf"*

Custom protocol types can be manually added to the NBAR database:

Router(config)# *ip nbar port-map MYPROTOCOL tcp 1982*

Updated PDLMs can be downloaded into flash and then referenced for NBAR:

Router(config)# *ip nbar pdlm flash://unrealtournament.pdlm*

(Reference: http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t8/dtnbarad.pdf)

* * *

Creating and Applying a QoS Policy using MQC

After traffic has been appropriately classified, **policy-maps** are used to dictate how that traffic should be treated (the per-hop behavior).

```
Router(config)# policy-map THEPOLICY
Router(config-pmap)# class LOWCLASS
Router(config-pmap-c)# set ip precedence 1

Router(config-pmap)# class HICLASS
Router(config-pmap-c)# set ip dscp af41
```

The *policy-map* command creates a policy named *THEPOLICY*. The *class* commands associate the *LOWCLASS* and *HICLASS* class-maps created earlier to this policy-map.

Within the policy-map class sub-configuration mode, *set* statements are used to specify the desired actions for the classified traffic. In the above example, specific *ip precedence* or *ip dscp* values have been marked on their respective traffic classes.

A wide variety of policy actions are available:

```
Router(config)# policy-map LOWPOLICY
Router(config-pmap)# class LOWCLASS
Router(config-pmap-c)# bandwidth 64
Router(config-pmap-c)# queue-limit 40
Router(config-pmap-c)# random-detect
```

The above is *by no means* a comprehensive list of policy actions. Reference the link below for a more complete list. Policy actions such as queuing and congestion avoidance will be covered in great detail in other guides.

Once the appropriate class-map(s) and policy are created, the policy must be applied directionally to an interface. An interface can have up to **two** QoS policies, one each for inbound and outbound traffic.

```
Router(config)# int fa0/0
Router(config-if)# service-policy input THEPOLICY
```

Any traffic matching the criteria of class-maps *LOWCLASS* and *HICLASS*, coming inbound on interface *fa0/0*, will have the actions specified in the policy-map *THEPOLICY* applied.

(Reference: http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfmcli2.html)

* * *

Troubleshooting MQC QoS

To view all configured class-maps:

Router# *show class-map*

```
Class Map LOWCLASS
 Match access-group 101

Class Map HICLASS
 Match protocol http host *routeralley.com*
 Match protocol http mime "*pdf"
```

To view all configured policy-maps:

Router# *show policy-map*

```
Policy Map THEPOLICY
    Class LOWCLASS
      set ip precedence 1
    Class HIGHCLASS
      set ip dscp af41
```

To view the statistics of a policy-map on a specific interface:

Router# *show policy-map interface fastethernet0/1*

```
FastEthernet0/0

   Service-policy input: THEPOLICY

     Class-map: LOWCLASS (match-all)
       15648 packets, 154861743 bytes
       1 minute offered rate 512000 bps, drop rate 0 bps
       Match: access-group 101
       QoS Set
         ip precedence 1
           Packets marked 15648
```

* * *