Chris Bryant, CCIE #12933 http://www.thebryantadvantage.com

Back To Index

Intro To Marking, Classification, And NBAR

Overview

Marking And Classification - Or Is It Classification And Marking?

Call it either, but remember that these are two separate processes, and classification is done before marking!

- Classification identifies a certain type of traffic
- Marking is assigning a value to that class of traffic

This being Cisco, naturally we have some options when it comes to marking traffic. Some marking techniques occur at Layer 2 of the OSI model, some at Layer 3. We'll start with L2 marking techniques.

L2 Marking Options - Code Of Service

You spent a lot of time studying trunking protocols during your CCNA studies, and you're about to spend just a bit more. Don't worry, we're not going to go over all the differences between ISL and dot1q again. What we do need to concentrate on is a value contained in the dot1q header - the *CoS value.*

Officially called the *Priority Field* and the *802.1p* Field, the three-bit CoS field has eight possible values.

000 - Routine (Best-Effort) 001 - Priority 010 - Immediate

011 - Flash 100 - Flash Override 101 - Critical (Voice Bearer)

110 - Internet (Reserved) 111 - Network (Reserved)

The highest non-reserved value, then, is 101 - Critical, sometimes referred to as *Voice Bearer,* which is a "name-is-the-recipe" name if ever there was one.

Note that these values are only going to be present when frames are trunked, regardless of whether ISL or dot1q is the trunking protocol.

What if they're sent across a Frame Relay link? You know about the congestion indicators FECN and BECN, but don't forget the 1-bit *Discard Eligible* bit! When congestion occurs to the point where frames must be dropped, frames with a DE of 1 are more likely to be dropped than frames with a DE of 0.

ATM has a *Cell Loss Priority* (CLP) field that serves the same purpose. (If you're not familiar with ATM, cells are used rather than packets.)

L3 Marking Options - IP Precedence And DSCP

Of these two, IP Prec came first, but DSCP is the more popular of the two options. You'll see why in just a minute, but let's take a quick look at where both the IP Prec and DSCP values come from in the first place!

In an IP header, there's an eight-bit field called the Type Of Service (ToS) byte. Of those eight bits, the first three bits from left to right - often called the three *most significant bits* - comprise the IP Prec value. As with CoS, this gives us eight possible values.

000 - Routine 001 - Priority 010 - Immediate

011 - Flash 100 - Flash Override 101 - Critical

110 - Internetwork Control (Reserved)

111 - Network Control (Reserved)

With those last two values reserved, that only leaves us six IP Prec values that can actually be assigned to "regular" traffic. Using IP Prec isn't "wrong" - it's just that using DSCP may be a little more right!

Why? Because DSCP gives us many more options for marking traffic. The terminology is a little different from our IP Prec discussion - hey, this is

networking, we have to have some different names for the same thing, right? With DSCP, the ToS byte is referred to as the *Differentiated Services* field (DiffServ), and the first *six* bits make up the DSCP.

DiffServ field: 000000 00

DSCP ECN

The *Explicit Congestion Notification* bits at the end of the DiffServ field do exactly what they sound like they do.

DSCP And Per-Hop Behavior (PHB)

Both the most significant bits and the least significant bits can link a given DSCP value to a certain PHB.

Expedited Forwarding results in the least jitter of the four PHBs we'll look at here, which gives you a hint as to what kind of traffic can best use this level of QoS - voice and video! If the most significant bits of the DSCP are set to 101, you're using EF. (That's equivalent to an IP Prec setting of 5, the highest assignable level.)

As with PQ's High-priority queue, you don't want to mark too much of your traffic as EF! Some non-Cisco documentation I've seen recommends that you keep the EF-marked traffic at less than 30% of your network's overall traffic, but that seems a little high to me. Again, it depends on your network's capacity and needs.

Assured Forwarding guarantees delivery as long as a predefined transmission rate is adhered to. Naturally, if that rate is exceeded, that traffic is likely to be dropped when congestion is encountered. If the three most significant bits are set to any of the following, AF is in use.

001 010 011 100

Default PHB is in use when the three most significant bits of the DSCP are all set to zero. As the name suggests, this PHB is assigned to any unmarked traffic and is basically "best-effort" service.

Class Selector PHB is set when the *least* significant three bits of the DSCP are all set to zero. DSCP is backward-compatible with IP Prec, and it's the class selector PHB that makes this possible.

Let's take a look at how the AF and CS bits work together. Remember, the three most significant bits of the DSCP value are AF bits and the least significant are CS bits. The final two bits are the ECN bits and aren't used to consider drop probabilities.

The eight DSCP bits: AF AF AF CS CS CS ECN ECN

The AF bits are used to predefine four classes. The CS bits are then used to determine the drop probability with a value of 1 - 3:

- 3 indicates the highest level of drop probability
- 2 is the intermediate level of drop probability (or "medium"
- 1 is the lowest level

You'll see these numbers expressed as "AFxx", with the first "x" represented by the AF bits; that simply indicates the AF class. The second "x" represents the CS value, which indicates - say it with me - drop probability! For example, "AF 32" indicates AF Class 3 with a medium drop probability.

Here's a list of the classes and their drop probability:

Class 1: AF11 is low, AF12 is medium, AF13 is high

Class 2: AF21 is low, AF22 is medium, AF23 is high

Class 3: AF31 is low, AF32 is medium, AF33 is high

Class 4: AF41 is low, AF42 is medium, AF43 is high

NBAR

The *Network Based Application Recognition* feature is a great place to start your QoS deployment - after all, before you worry about marking or classifying traffic, you've got to identify the traffic on your network! NBAR works from OSI layers 4 - 7 to identify and classify protocols, making it possible for you to create an accurate QoS configuration.

You may also be familiar with the use of NBAR to block the "Code Red" virus. It's highly doubtful you'll see any questions on your exam regarding that, but if you'd like to learn more about that, just put "nbar code red" in your favorite search engine and you'll quickly find some Cisco documentation on that subject.

NBAR has two basic purposes:

- Identifying traffic on a per-protocol basis
- Application monitoring

NBAR performs application monitoring via its *Protocol Discovery* feature. Protocol Discovery is enabled on a per-interface basis, and this helpful feature can tell you what applications are running on that interface

and the amount of bandwidth they're using.

NBAR is capable of categorizing traffic in three ways:

- protocol
- port number
- payload content (default limit: 400 bytes)

NBAR also examines network traffic and classifies it by TCP and UDP port numbers, whether those port numbers be statically or dynamically assigned. When dynamic port numbers are involved, NBAR is said to be performing a *stateful inspection*.

NBAR can identify IP protocols that are neither TCP-based not UDP-based as well, and NBAR's capabilities go beyond identifying port numbers. NBAR can identify and classify WWW traffic according to the URL as well.

Best of all, NBAR's capabilities are continually extended through the development of ***Packet Description Language Modules*** (PDLM). Not only do these PDLMs allow your NBAR deployment to identify more and more different types of traffic, but a router reload is not necessary, and you don't need a new IOS image.

We'll look at the command to load PDLMs in just a moment, but for now keep in mind that you're generally going to store PDLMs in Flash.

You will need a Cisco CCO number to access PDLMs; the download page can be quickly accessed through your favorite search engine.

NBAR can go above and beyond "just" examining port numbers. Through *subport classification*, NBAR can examine the payload and classify packets on other values, such as the Multipurpose Internet Mail Extension (MIME) type or, as previously mentioned, the URL. This detailed packet examination is called *deep packet inspection.* (NBAR cannot support more than 24 MIME or URL matches at the same time.)

NBAR Configuration

Cisco Express Forwarding (CEF) must be enabled on the interfaces that will run NBAR.

Enabling NBAR protocol discovery (an optional command, but required for application monitoring):

```
R2(config-if)#ip nbar protocol-discovery
```

Verify with *show ip nbar protocol-discovery.* To illustrate the results, I ran BGP on this interface for a few minutes before running this command.

This command's output is verbose to say the least, so I will not show the entire output here.

```
R2#show ip nbar protocol-discovery

 Serial0/0
                             Input                    Output
                             -----                    ------
 Protocol                 Packet Count             Packet Count
                          Byte Count               Byte Count
                          5min Bit Rate (bps)      5min Bit Rate (bps)
                          5min Max Bit Rate (bps)  5min Max Bit Rate (bps)

   bgp                      19                       38
                            1197                     2033
                            0                        0
                            0                        0
```

Loading a PDLM (assuming the PDLM is in Flash) :

```
R2(config)#ip nbar pdlm ?
  WORD  Full path of the PDLM file

R2(config)#ip nbar pdlm flash://ccnp.pdlm
```

To verify the port numbers used by NBAR, run *show ip nbar port-map*. To verify a single port number, put the protocol name at the end of that command. I've truncated the output of the first command.

```
R2#show ip nbar port-map
port-map bgp                      udp 179
port-map bgp                      tcp 179
port-map citrix                   udp 1604
port-map citrix                   tcp 1494
port-map cuseeme                  udp 7648 7649 24032
port-map cuseeme                  tcp 7648 7649

R2#show ip nbar port-map dns
port-map dns                      udp 53
port-map dns                      tcp 53
```

NBAR Limitations And Restrictions

If you're interested in using NBAR in your network, be *sure* to visit Cisco's website for the latest documentation. In the meantime, here's a partial list of NBAR's limitations.

NBAR does *not* support or analyze:

- Non-IP traffic
- MPLS packets

- Fragments
- Packets created by the local router (the one actually running NBAR)
- Packets destined for the local router

Additionally, you cannot run NBAR on these interface types:

- Interfaces running encryption or tunneling
- Dialer interfaces
- Fast Etherchannel (NOTE - that's *Fast Etherchannel*, not *Fast Ethernet*)

The dialer interface limitation was removed as of IOS 12.2(4)T.

And one more thing...

- The only switching mode that supports NBAR is CEF.

Configuring Traffic Classes And Policies With NBAR

Creating these policies is a simple and now-familiar three-part process:

- Create the traffic *class* with the *class-map* command
- Create the traffic *policy* with the *policy-map* command
- Apply the policy to the appropriate interfaces with the *service-policy* command

I placed "class" and "policy" in italics to remind you of the proper order - creating the policy first won't work!

The *class-map* command is used to name the class, and more importantly, to define the match criteria. I'll use IOS Help to display two options that seem a little confusing at first...

```
R2(config)#class-map ?
  WORD       class-map name
  match-all  Logical-AND all matching statements under this classmap
  match-any  Logical-OR all matching statements under this classmap
```

... just remember that the name is the recipe! If you use the *match-any* option, only one of the criteria we're about to list has to match; if you use the *match-all* command, well, all of the criteria have to match! In this example, we'll go with *match-any* and call the class-map EXAMPLE.

```
R2(config)#class-map match-any EXAMPLE
R2(config-cmap)#

R2(config-cmap)#match ?
  access-group         Access group
  any                  Any packets
```

```
  class-map            Class map
 cos                   IEEE  802.1Q/ISL  class  of  service/user  priority
values
  destination-address  Destination address
  discard-class        Discard behavior identifier
  dscp                 Match DSCP in IP(v4) and IPv6 packets
  fr-de                Match on Frame-relay DE bit
  fr-dlci              Match on fr-dlci
  input-interface      Select an input interface to match
  ip                   IP specific values
  mpls                 Multi Protocol Label Switching specific values
  not                  Negate this match result
  packet               Layer 3 Packet length
  precedence           Match Precedence in IP(v4) and IPv6 packets
  protocol             Protocol
  qos-group            Qos-group
  source-address       Source address
```

We're going to select *protocol* here, and I'm not going to show you every choice that IOS Help gives us... but rest assured there are a lot of them! We'll use the last two protocols listed in the IOS Help output, *winmx* and *xwindows.*

```
  winmx             WinMx file-sharing application
  xwindows          X-Windows remote access

R2(config-cmap)#match protocol winmx
R2(config-cmap)#match protocol xwindows
```

Note that we can write more than one match protocol statement with no problem. Since we applied the *match-any* option to the class-map, traffic *matching* either match statement will -- you guessed it -- match!

Now we'll write the traffic policy itself, and call it IPPREC. The *class* command names the class that will be called when this policy-map runs, and we'll use IOS Help to display the *set* command's options.

```
R2(config-pmap-c)#set ?
  atm-clp         Set ATM CLP bit to 1
  cos             Set IEEE 802.1Q/ISL class of service/user priority
  discard-class   Discard behavior identifier
  dscp            Set DSCP in IP(v4) and IPv6 packets
  fr-de           Set FR DE bit to 1
  ip              Set IP specific values
  mpls            Set MPLS specific values
  precedence      Set precedence in IP(v4) and IPv6 packets
  qos-group       Set QoS Group
```

If we choose *precedence* and use IOS Help again, we're shown a now-familiar list of values.

```
R2(config-pmap-c)#set precedence ?
  <0-7>            Precedence value
  cos              Set packet precedence from L2 COS
  critical         Set packets with critical precedence (5)
  flash            Set packets with flash precedence (3)
  flash-override   Set packets with flash override precedence (4)
  immediate        Set packets with immediate precedence (2)
```

```
  internet        Set packets with internetwork control precedence (6)
  network         Set packets with network control precedence (7)
  priority        Set packets with priority precedence (1)
  qos-group       Set packet precedence from QoS Group.
  routine         Set packets with routine precedence (0)
```

You can enter either the number or the desired precedence's full name.

```
R2(config-pmap-c)#set precedence network
```

All traffic matching either of the class-map's *match* statements will have its IP Precedence set to 7 (network).

Finally, we need to apply this policy! We'll use the *service-policy* command to do so, and note that this policy can be applied to inbound or outbound traffic. The direction of the packets affected by the policy must be defined when the policy is applied.

```
R2(config)#int s0/0
R2(config-if)#service-policy ?
  input   Assign policy-map to the input of an interface
  output  Assign policy-map to the output of an interface

R2(config-if)#service-policy input ?
  WORD  policy-map name

R2(config-if)#service-policy input IPPREC ?
  <cr>
```

Verify the class-map configuration with *show class-map*. Note the *class-default* statement.

```
R2#show class-map
 Class Map match-any class-default (id 0)
   Match any

 Class Map match-any EXAMPLE (id 1)
   Match protocol winmx
   Match protocol xwindows
```

Verify the policy-map with *show policy-map* and/or *show policy-map interface.*

```
R2#show policy-map
  Policy Map IPPREC
    Class EXAMPLE
      set precedence 7

R2#show policy-map interface serial0/0

 Serial0/0

  Service-policy input: IPPREC

    Class-map: EXAMPLE (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
```

```
    Match: protocol winmx
      0 packets, 0 bytes
      5 minute rate 0 bps
    Match: protocol xwindows
      0 packets, 0 bytes
      5 minute rate 0 bps
    QoS Set
      precedence 7
        Packets marked 0

  Class-map: class-default (match-any)
    13 packets, 1005 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: any
```

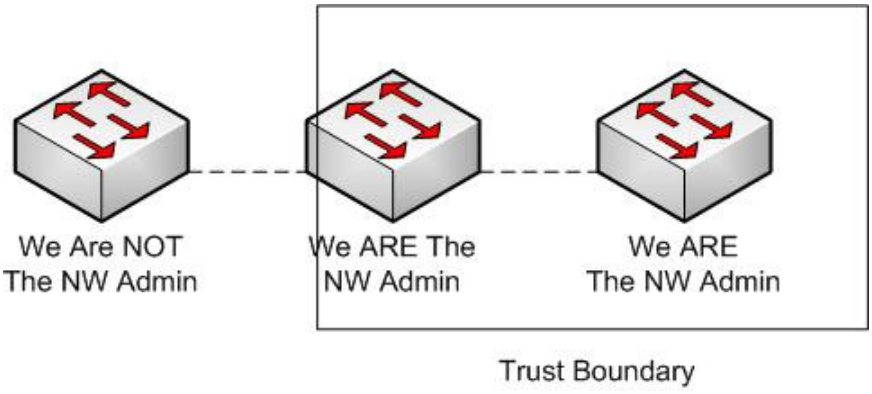
We'll see more of NBAR in later sections, using SDM.

Should Incoming Traffic Markings Be Trusted?

As network admins, we're suspicious types by nature, and that suspicion should extend to incoming CoS, IP Prec, and DSCP values as well!

A trust boundary is the point at which your network no longer trusts such an incoming value. For example, let's look at a three-switch network - two switches that are under our direct control, and one that is not.

If we're not directly responsible for a switch, we probably don't want to unconditionally trust any values coming in from that switch. That means the trust boundary would not extend to that switch.



And remember -- *keep classification away from the core switches!*