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

Back To Index

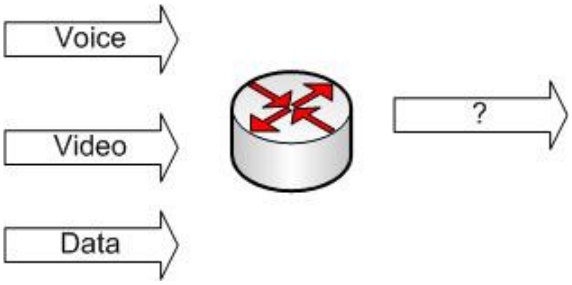
Introduction To Quality Of Service

Overview

Quality of Service (QoS) is a way to give delay-sensitive traffic the attention it deserves while (hopefully) still getting "regular" traffic to its destination in a timely manner.

As network admins, you and I use QoS to get predictable transmission results in our networks. We like predictability, particularly when it comes to voice and video traffic.

When you break QoS down to its very core, we're answering one question: "*Which data should be transmitted first*?" That used to be an easy question to answer, since we didn't have a lot of time-sensitive data traveling across our network. Then again, a floppy disk used to store all the information we needed, and that's not true anymore either!



On today's networks,we have different data types - voice and video, just to name two - and we have traffic that's much more time-sensitive than it used to be. When our network gets congested, QoS helps to ensure that the voice and video streams receive the proper level of attention, which in turn ensures that voice and video presentations are of high quality.

You've probably watched a video presentation that was more than a little choppy, or listened to streaming video that would play the stream for 30 seconds, then stop for a second or two, then pick back up. That's *jitter,* and our major weapon to fight jitter is QoS.

Fighting jitter isn't the only reason we configure QoS. Here are four other major reasons to use QoS:

- Provide adequate bandwidth, and guarantee it in some situations
- Provide consistent and predictable packet delivery
- Prevent delay, especially with voice and video transmission
- Prevent packet loss (especially via *tail drop*, which we'll address in another section)

I'll speak of "voice and video" quite a bit in this course, and while both types of data benefit from QoS, the application of QoS may be a bit different when you're dealing with video packets as opposed to voice packets.

Video packets are larger than voice packets, and video tends to be more bursty than voice, particularly with video conferencing. From personal experience, I'll tell you this - video conferencing is *very* bursty and *very* hard on a network that doesn't usually carry such transmissions! The proper application of QoS will give you an acceptable level of quality in both your voice and video transmissions while keeping jitter to a minimum.

Network congestion isn't the only potential issue with voice and video traffic. As you'll see later in this course, *header overhead* is a real problem with voice packets. We'll discuss that in detail during the VoIP section, but keep in mind that congestion *and* overhead are problems with voice packets.

There Are Three Overall Steps To A QoS Deployment

The basic process we'll use to apply QoS to our network is *"Identify, Classify, Define and Apply"*

- *Identify* the traffic that needs QoS, and determine the QoS requirements of that traffic.
- Use the requirements to *classify* the traffic.
- *Define and apply* a policy for each class of traffic.

Defining and applying a policy has three main steps of its own, or rather three values that should be defined and applied:

A maximum bandwidth limit should be set. You don't want any traffic class taking up all of your bandwidth!

A minimum bandwidth guarantee should be set as well. One of the major reasons we use QoS is to guarantee a certain level of service.

Assign each class a priority level in proportion to the level of service the traffic class should receive when compared to the other traffic classes. If one traffic class is basically twice as important as another, assign that class a priority twice as high as the other.

Not All Delays Are Created Equal

It's a fair bet that sometime during your studies, you're going to wonder why we really need so many different ways to perform QoS. The main reason we need different ways to combat transmission delay is that we've got different kinds of delay in the first place.

Variable-Length Delays:

Queuing Delay is the amount of time a packet spends in the exit queue before being transmitted.

Processing Delay is the time it takes the network device to move a packet from the incoming queue into the appropriate outgoing queue.

Fixed-Length Delays:

Serialization Delay is the time it takes to place the frame onto the physical medium.

Propagation delay is the amount of time it takes for the bits to cross the physical media from the tranmission point to the point of reception.

The propagation delay is affected by the speed and utilization level of the CPU as well as the features you've got running on both the input and output interfaces handling the packets. That's why you've got to be careful and only run the features you need - every feature we configure on a Cisco

router or switch has a cost to the CPU!

The sum of those four delays is the overall *end-to-end delay*.

QoS Models

As with most things in Ciscoland, we've got more than one way to do things. When it comes to QoS planning, we have three different models to choose from. Let's take a look at the pros and cons of each.

Best-Effort

If you don't have a QoS model in place, you actually do. (No additional charge for the Zen lesson.) The default level of QoS is *best-effort.*

Best-effort QoS is just that - best-effort. No priority is given to any traffic. If your network is carrying voice or video traffic, best-effort is definitely *not* the way to go. Not to say that all best-effort QoS applications are bad; if you're on the Internet, you're using a best-effort network. That in itself means that best-effort QoS is highly scalable.

There's nothing wrong with using best-effort QoS. You're probably using quite a bit of it right now for network applications. Just remember that there are no classes, no default classes, no preferential treatment for high-quality traffic - no nothin'!

Integrated Services (IntServ)

The Integrated Services model, more popularly known as *IntServ,* performs admission control via the *Resource Reservation Protocol* (RSVP) to send an advance signal and reserve network resources in advance of the data actually traveling across the network. Once the end-to-end bandwidth reservation is in place, the data is transmitted.

Here's a little more info about RSVP:

- Uses Internet Protocol (IP) ID 46
- Uses TCP/UDP ports 3445
- Is not a routing protocol; rather, it's a *signaling* protocol
- Initial RFC is 2205; if you're going to use RSVP over IP tunnels, I recommend reading RFC 2746 as well
- If RSVP cannot reserve the required bandwidth, the application will not work.

That sounds great, and it's certainly better than best-effort QoS! However, there are some drawbacks, the biggest being that it's a waste of bandwidth to have the entire end-to-end path reserved in advance.

Additionally, IntServ isn't as scalable a solution as we'd like. Everything we do on a router or switch has a cost of some kind, and in this case it's RSVP overhead. One or two paths won't cause much overhead, but as the number of reserved paths increases as a network becomes larger, the RSVP overhead will take its toll.

IntServ actually requires *six* functions to run on all network devices along the path from source to destination:

- end-to-end signaling
- admission control (actually responds to the end-to-end service request)
- classification
- policing
- queuing
- scheduling

Differentiated Services (DiffServ)

Differentiated Services is the latest of the three models, and many would agree that it's also the greatest. DiffServ doesn't use RSVP, but instead uses *Per-Hop Behavior* (PHB) to allow each router across the network to examine the packet and decide what service level it should receive.

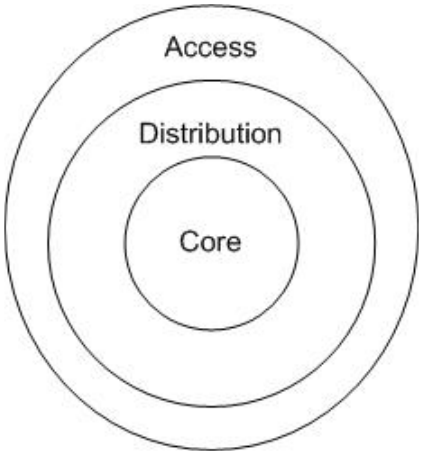
With DiffServ, one router along the path from source to destination could consider a packet to be of the highest priority, while another router could consider it "just another packet".

There is no advance signaling with DiffServ - no "hey, here comes a really important packet!" advance notice. Each hop along the way from source to destination makes its own decision as to how important a packet is or isn't. This lack of advance signaling is why DiffServ is considered more scalable than IntServ, since no bandwidth is reserved in advance of the actual transmission.

A term you hear often with DiffServ is "marking and classification". *Marking* a packet is the process of assigning the packet a value reflecting the level of QoS it should receive, while *classification* is placing that packet into a queue in accordance with that level of QoS.

Those of you who have taken my Switching courses know what I'm about to say - when you perform classification with switches, *don't perform classification on the core switches!*

What follows is the Cisco Three-Layer switching model; both Cisco and I recommend that you perform traffic marking and classification as close to the edge of the network as possible.



When it comes to marking, there are different values we can use to decide what value to mark the frame or packet with. In my experience, here are the four that are used most often:

- IP Precedence (IP Prec)
- Differentiated Services Code Point (DSCP)
- CoS value
- Interface that received the data (ingress interface)

Which one you choose depends on your particular network's needs, and of course, the OSI layer at which the marking is taking place. We'll look at several options as we work through the rest of the course.

Okay, So Which Model Should I Use?

Every network is different, and so are every network's needs. Here's a basic list of questions to ask before deciding on a model.

What applications are running on the network? Best-effort may be just fine for your network; if you're running voice and video, it's not a good choice.

To what extent do you want to tie down your network's transmissions? In some networks, you may not even want to configure QoS, but in most of today's networks you need to exercise at least some control over data transmission. That's what QoS is all about!

How much is all of this gonna cost? Cost isn't just measured in money, but in time. A proper QoS deployment takes time to configure and even more time to plan. That proper deployment can also make your life a lot easier - as long as it's cost-efficient!

"Hot Spots And Gotchas"

As always, reading this section is no substitute for reading the entire section!

Why do we go to all the trouble of configuring QoS? QoS allows us to have some predictability in our network while allowing for special attention to be given to high-priority traffic.... unless you're using no QoS, in which case you're actually using best-effort QoS!

Nothing annoys our clients more than jittery voice and video traffic, and it's up to us - and our QoS deployment - to hold jitter to an absolute minimum.

Another reason we use QoS is to avoid tail drop. When the queue is full, packets are dropped as they arrive at a queue since there's no place to put them. Tail drop does not allow us to specify packet types to drop first, and in today's networks we prefer a scheme that does allow us to give preferential treatment to some packet types in both delivery and the order in which they're dropped.

Congested queues are one major reason we configure QoS; header overhead is also an issue, again mostly with video and voice traffic. We have some header compression techniques that can help with that overhead, and we'll cover those in another section (TCP Header Compression and RTP Header Compression).

Best-Effort QoS:

It's easy to dismiss "best-effort" anything, but best-effort QoS must be pretty good - the Internet uses it! Why? High scalability.

Best-effort is the implicit, default QoS model; if you haven't configured any specific QoS model, you're using best-effort. Best-effort QoS means that all traffic is treated just the same - that includes video and voice!

You're probably using some best-effort QoS in your network right now. Many vital network applications - email chief among them - have no specific QoS policy configured for them. I'm sure we'll all agree that email is quite important - just consider how upset everyone gets when it's down!

Voice and video traffic begs for a guarantee of priority handling, which best-effort cannot guarantee.

While both video and voice can be bursty, video is generally considered to be "burstier" than voice. I can personally vouch for that, believe me. :)

Why does voice and video traffic need special handling? By using QoS

to give preferential treatment to such traffic, we avoid jitter. We need to have voice and video traffic delivered in a predictable manner, without delays between packets arriving at their destination.

Integrated Services (IntServ)

IntServ is not particularly scalable due to its use of the Resource Reservation Protocol (RSVP). RSVP uses IP protocol ID 46 and TCP/UDP port number 3455.

RSVP requires applications to signal their intent to transmit in advance of the data being transmitted and indicate the level of QoS the application requires. As a result, a path across the network is reserved *in advance* of that transmission. This bandwidth reservation does allow you to set a high level of service for voice, video, and other delay-sensitive traffic, but the sheer overhead of RSVP can be a problem.

RSVP is a great choice if you have bandwidth to be reserved in this fashion - and not allow any other application to use that bandwidth, even if the application reserving the bandwidth isn't using it at the moment - but in today's networks we usually do not have that luxury.

IntServ actually requires *six* functions to run on all network devices along the path from source to destination:

- end-to-end signaling
- admission control (actually responds to the end-to-end service request)
- classification
- policing
- queuing
- scheduling

Differentiated Services (DiffServ)

Highly scalable, much more so than IntServ

Generally uses DSCP or IP Precedence to determine the appropriate QoS level, but source and destination IP addresses can be used as well.

Offers more levels of QoS than the other models

Can be more complex to configure and get the exact QoS results you want than other models

No pre-transmission signaling such as that used by RSVP. Rather, the traffic is divided into classes, and each individual class will have its own level of QoS. It's basically a three-step process:

Identify your traffic type and flows. What level of QoS do your traffic flows require?

Place the traffic types into classes

Configure appropriate QoS levels for each class

There's no "one-size-fits-all" way of determining the most efficient QoS model for your network. A basic three-step model I like to use:

What applications are currently on the network? Do they require IntServ or DiffServ, or are they running just fine on best-effort QoS?

To what level do we need to apply out QoS model? Do we have multiple apps that require special handling?

Finally, how much will this QoS model cost us in time, effort, and our client's money?

Other Topics

Our delay types:

Processing - time it takes for a packet to move from the input interface to the queue leading to the output interface

Queueing - time a packet stays in the output queue

Serialization - time it takes to put a frame onto the physical medium

Propagation delay is the amount of time it takes for the bits to cross the physical media from the transmission point to the point of reception.

End-to-End - Overall time taken from very beginning of transmission to actually being received. ETE delay is the sum of the first four delays listed.

Not all delays are created equal! The serialization and propagation delays are fixed in length; the processing and queueing delays are variable in length.