VISOLVE
OPEN SOURCE SOLUTIONS

QOS

Bandwidth Management

Prepared By
Visolve Squid Team

Abstract

Today in the modern communication world, the traffic that exists in the Internet is becoming more and more abnormal. This was mainly due to increase in number of users day by day which results in bandwidth congestion, poor response time for end user's etc., The most efficient solution to this problem is to manage and allocate the existing bandwidth almost equally using suitable queuing disciplines and filters that exist as the Quality of Service(QoS) support in linux . It is a full featured technology which may reduce the cost and improve network performance.

Introduction

The main purpose of this white paper is to discuss various techniques and concepts used in Quality of Service (QoS) support in linux. In the QoS concept, the discussion is widened particularly towards three basic building blocks namely Queuing Disciplines, Classes and filters. First a detailed description on various queuing disciplines, classes and filters will be discussed followed by the working of u32 classifier along with Class based queuing discipline.

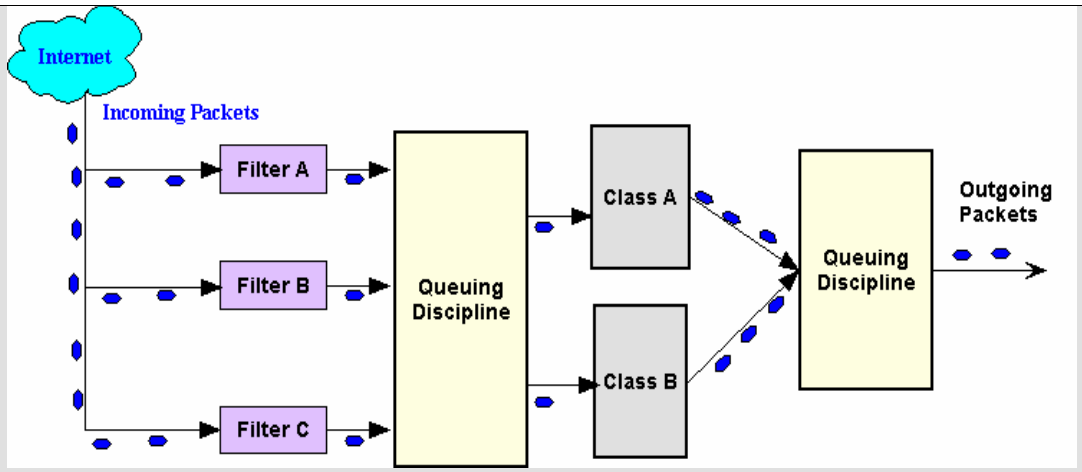
What is Quality of Service (QoS)?

QoS is defined as the proficiency of a network element to furnish some degree of commitment for congenial network data delivery. In other words, QoS means, satisfying customer application requirements, providing a network that is transparent to its users. QoS does not generate bandwidth. Instead it only administers the bandwidth according to the application demands and network management settings.

QoS support in Linux :

The QoS support in linux consists of the following three basic building blocks, namely :
Queuing Discipline
Class Based queuing
Filters/Policers/Classifiers

The packets from the internet flow directly into the filter and from there they reach the respective queuing discipline. The queuing discipline in turn moves the packets to the classes. The main function of the classes is to hold the packets under it, which makes the queuing disciplines to request the filters in order to identify which class actually the packets belongs. These classes also does not hold the packets permanently within them, but instead they uses another queuing discipline for the purpose of holding the packets.



Queuing disciplines

Every network device has a queuing discipline associated with it. The main function of the queuing discipline is to control how packets enqueued on that particular device. The various queuing disciplines that are supported in linux includes

- Class Based Queuing (CBQ)
- Token Bucket Flow (TBF)
- First In First Out (FIFO)
- Stochastic Fair Queuing (SFQ)
- Asynchronous Transfer Mode (ATM)
- Random Early Detection (RED)
- Generalized RED (GRED)
- Diff-Serv Marker (DS_MARK)
- Clark-Shenker-Zhang (CSZ)
- Priority
- Traffic Equalizer (TEQL) etc.

The following lines gives a brief overview about each and every queuing disciplines.

Class Based Queuing

Class-Based Queuing (CBQ) is a most popular mechanism used with operating system design mainly to prevent complete resource denial to any particular class of service. CBQ is a variation of priority queuing, where several output queues can be defined. A preference for each of the queues can be set indicating the service preference and the amount of queued traffic, measured in bytes.

Token Bucket Filter

A token bucket filter is used to control the amount and volume of traffic being sent onto the network and the rate at which the traffic is being sent. The service of the incoming packets is governed by the arrival of tokens inside the buffer (bucket). Each

	arriving token lets one incoming data packet out the queue and is then deleted from the bucket. i.e., The packets are stored in a 'bucket' and are send at a certain frequency. When there are too much packets, they have to wait until they can be send. <u>First In First Out</u> This is one of the simplest queuing disciplines available. In FIFO queuing, all packets are stored in a single queue and are transmitted in the order that they are received. It is both a queuing and a scheduling mechanism. It is the most popularly used queuing discipline among the other. So by default, all interfaces have FIFO as their queuing discipline. <u>Stochastic Fair Queuing</u> As the name implies, it consists of dynamically allocated number of first in first out queues and during each and every conversation it allows only one queue. This makes each packet to travel to its destination without any postponement. The main advantage of this queuing discipline is that it allows fair sharing the link between several applications. <u>Asynchronous Transfer Mode (ATM)</u> This ATM queuing discipline has the ability to send the traffic directly over a specified permanent virtual circuit (PVC) without using any private IP addresses. This queuing discipline uses the standard outgoing interface, so there is no need for any special routing decision to be made. In this ATM queuing discipline, some ATM related information is attached and the packet is sent to a FIFO or any other attached queuing discipline. The dequeue function of the ATM queuing discipline then encapsulates and sends the packets over the specified PVC. <u>Random Early Detection (RED)</u> When the TCP/IP starts transmitting the packets from one end to the other, neither end knows what is the available bandwidth. In the initial stage, the transmission starts slowly and increases its speed gradually. At one stage the link will start filling up which makes the RED to drop some packets and also it passes information to the TCP/IP indicating the congestion in the link. i.e., it allows for queue congestion avoidance by observing queue sizes. This is the main advantage of this queuing discipline. This simulates real congestion, so TCP reacts to the packet loss by reducing transmission speed thus reducing congestion. <u>Generalized RED (GRED)</u> The GRED is more generalized form of the RED. GRED supports multiple drop priorities and buffer sharing. In times of congestion, when it has to drop packets, it will drop them with the drop probability that is associated with the queue. The GRED then queues up the packet in the CBQ, which in turn sends the packet to the ds_mark queuing discipline for the packet to be transmitted.

	Diff-Serv Marker (DS_MARK) The diffserv queuing discipline offers a framework within which the ISP's can offer each customer a range of network services which are differentiated on the basis of performance. Clark-Shenker-Zhang (CSZ) CSZ presents a more precise but less flexible and less efficient approach. The main idea is to create WFQ flows for each guaranteed service and to allocate the rest of bandwidth to dummy flow-0. Flow-0 comprises the predictive services and the best effort traffic. It is handled by a priority scheduler with the highest priority band allocated for predictive services, and the rest to the best effort packets. Priority In this, the router maintains multiple queues. High priority packets are put in the high priority queue and low priority ones in the low priority queue. Packets from high priority queue are first sent on the link. Only when the high priority queue is empty, packets from low priority queue are sent. Traffic Equalizer (TEQL) It's used to combine several internet connections to one virtual connection i.e., It allows to bundle interfaces. It is recommended although not by any means required to use interfaces of equal speed. This is due to packet reordering occurring if using a relatively slow and a fast interface together.

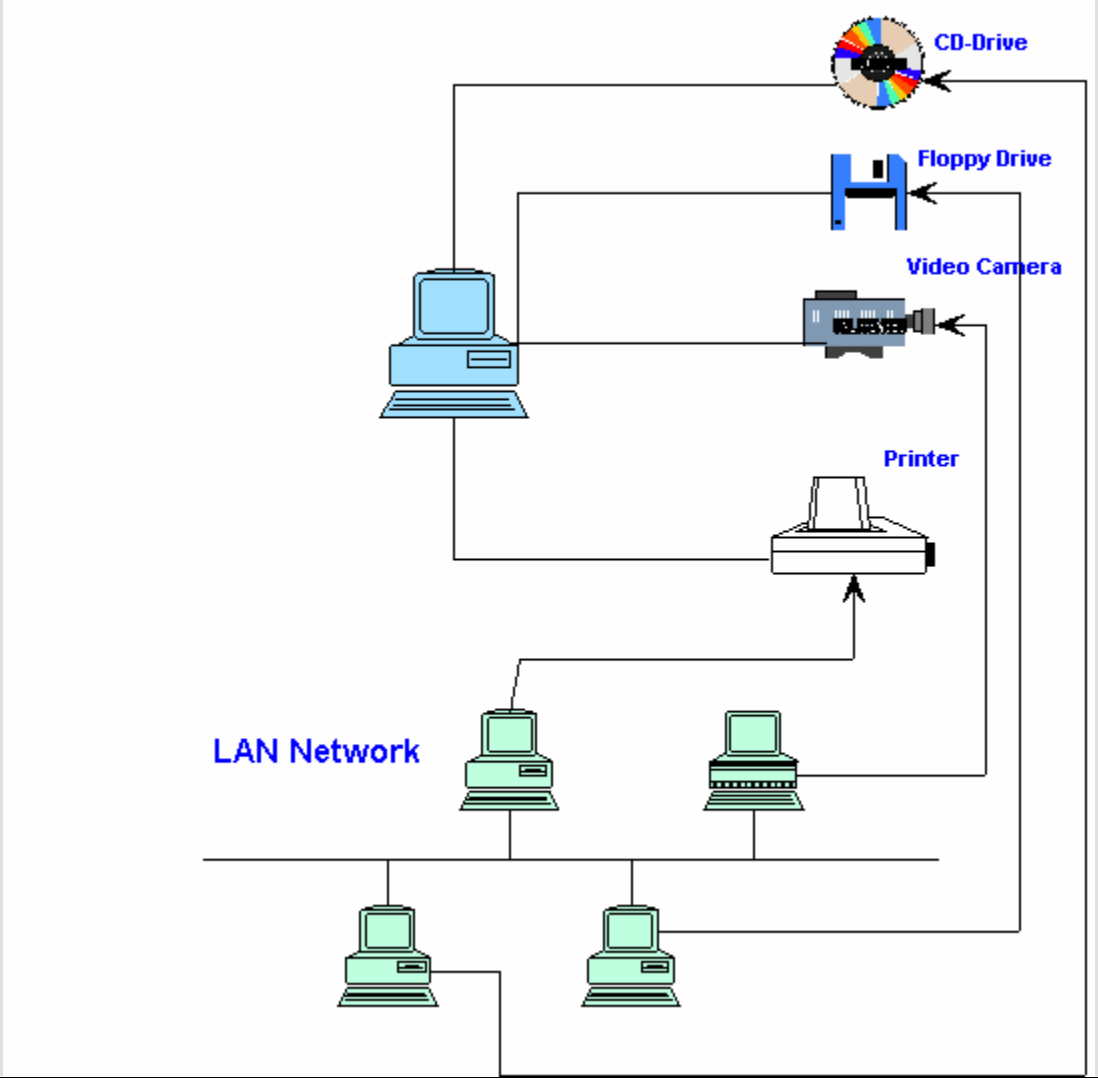
Classes	The occurrence of classes and their functionalities are the basic properties of the queuing disciplines. Normally the queuing disciplines have a one to one relationship with the classes. Each class owns a separate queue. The main function of the classes is to hold the packets under it, which makes the queuing disciplines to request the filters in order to identify which class actually the packets belongs. The class can be identified by two ways. One is by means of a class identifier (u32 data type) which is indicated by the user himself. The other is by means of an Internal Identifier (used within the kernel itself) which is used more frequently by the various functions present in the class. Each and every queuing discipline may have a number of classes. These classes does not hold the packets within itself, but instead it uses another queuing discipline for the purpose of holding the packets. The queuing discipline will again have a number of classes and the process will proceed like that, which makes the quality of service in linux matchless with any other. The various queuing disciplines that support classes are namely Class Based Queuing, Clark-Shenker-Zhang, Diff-serv Marker and First In First Out (default one).

Classifiers/Filters	The main function of the filters is to organize the packets based on their properties. The classification of packets may be done by listening their respective IP header, IP address, port numbers etc., There are 3 major components in the system : The classifier, the queues, and the scheduler. <u>Filters</u> The packet first flows into the system through the classifier. It is the classifier's duty to decide what to do with the packet. A very comprehensive set of user-configurable policies that make up the policy database control how the classifier identifies each packet and what it does with each packet. The policy database will be discussed in great details later in this document. When the classifier sees the packet, it can do one of three things : **1.Discard the packet :** This allows the classifier to provide a very robust and granular packet filtering mechanism. **2.Forward the packet at real time :** This means that the packet bypasses the entire bandwidth management system and is immediately forwarded by the device. The end-result is effectively the same as if bandwidth management was not enabled at all. **3.Prioritize the packet :** This allows the mechanism to provide actual bandwidth management services.

Types of Filters	RSVP (v4/v6) U32 Route Fw/ipchains (Fire wall based classifier) Tcindex Police Estimator

RSVP	The RSVP protocol is used by a host to request specific qualities of service from the network for particular application data streams or flows. RSVP is also used by routers to deliver quality-of-service (QoS) requests to all nodes along the path (s) of the flows and to establish and maintain state to provide the requested service. RSVP requests will generally result in resources being reserved in each node along the data path. RSVP carries the request through the network, visiting each node the network uses to carry the stream. At each node, RSVP attempts to make a resource reservation for the stream. To make a resource reservation at a node, the RSVP daemon communicates with two local decision modules, admission control and policy control. Admission control determines whether the node has sufficient available resources to supply the requested QoS. Policy control determines whether the user has administrative permission to make the reservation. If either check fails, the RSVP program returns an error notification to the application process that originated the

request. If both checks succeed, the RSVP daemon sets parameters in a packet classifier and packet scheduler to obtain the desired QoS. The packet classifier determines the QoS class for each packet and the scheduler orders packet transmission to achieve the promised QoS for each stream.



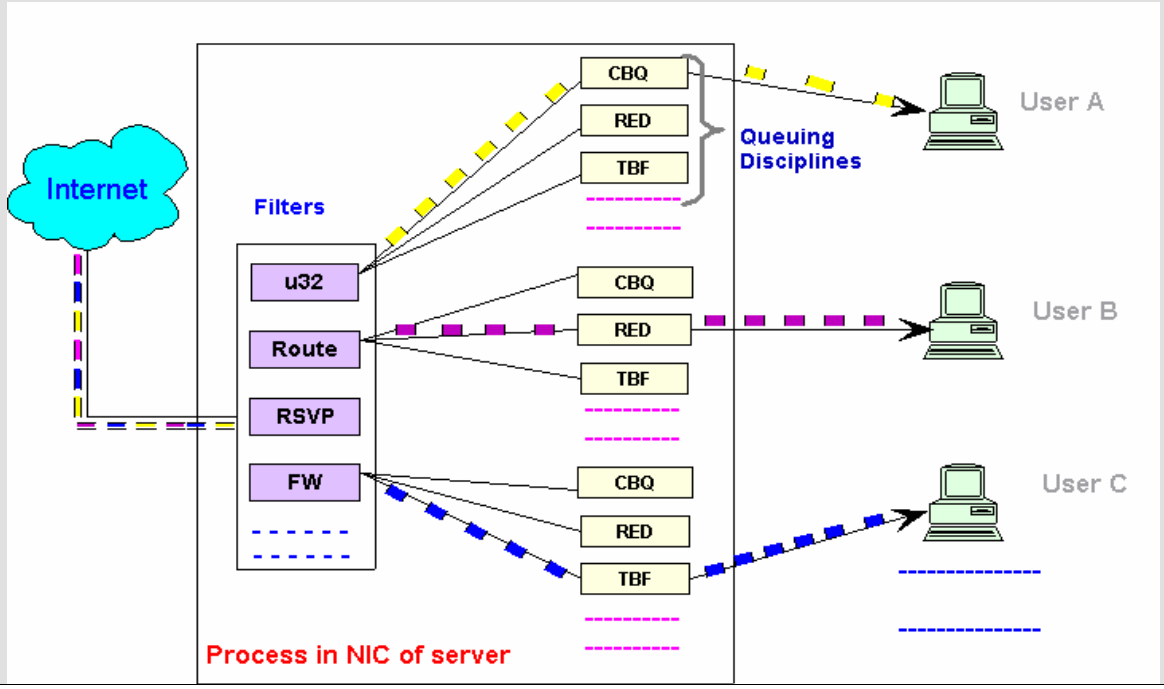
Different Classifiers

U32 classifier : The U32 filter is the most advanced filter available in the current implementation. Classification can be done based on the destination IP address, destination TCP/UDP port, source IP address, source TCP/UDP port, TOS byte and protocol.

ROUTE classifier : This classifier filters are based on the results of the routing tables. When a packet that is traversing through the classes reaches one that is marked with the "route" filter, it splits the packets up based on information in the routing table.

Fw (Fire wall based classifier) : The "fw" classifier relies on the firewall tagging the packets to be shaped.

	Police & Estimator : They may be used as a parameter to the filters. In Squid Box, packets which are redirected by a smart switch or router to the Squid box still need to be redirected to the port where Squid is listening on. Redirecting these packets cannot be done by Squid. Redirecting packets must be done by the Linux kernel, using the IP-chains program. The kernel then receives a packet on port 80, looks at the firewall configuration, and adjusts the packet appropriately i.e. by changing the destination port to 3128, or whatever port Squid is running on. If you need IP Filter redirection, then use the -enable-ipf-transparent configure option in Squid to support certain HTTP clients (HTTP/1.0 clients, NOT sending the Host header). However, normal browsing using the popular browsers will work even without it.

QoS as Bandwidth Management	Bandwidth is a measurement of the running data from one computer to another. All transmitted signals, whether analog or digital, have a certain bandwidth. The same is true of receiving systems. An example it would take more bandwidth to upload a streamline video than to upload a HTML file. Bandwidth is directly proportional to the amount of data transmitted or received per unit time. Bandwidth management is a dedicated effort to optimize the traffic, rather than increase. In real scenario, the Class based queuing discipline works fine with u32 classifier which might be a good complement for ISP's and other enterprises. QoS is a full featured technology that can control or reduce costs and improve network performance. A variety of traffic shaping, prioritization, filtering, congestion management features such as Committed Access Rate (CAR), Custom, Priority and Weighted Fair Queuing (WFQ), Resource Reservation Protocol (RSVP), Weighted Random Early Detection (WRED), and Policy Based Routing are used . These capabilities will also decrease latency and improve application availability. 

U32 filter with CBQ	As we already discussed, there are 3 major components in the system namely the classifier, the queues, and the scheduler. The packet first flows into the system through the classifier. It's the classifier's duty to decide what to do with the packet. A very comprehensive set of user-configurable policies that make up the policy database control how the classifier identifies each packet and what it does with each packet. When the classifier sees the packet, it can do one of three things: 1) Discard the packet 2) Forward the packet at real time 3) Prioritize the packet. Here the filter followed is U32. The U32 filter is the most advanced filter available in the current implementation. It has a robust behavior in spite many filter rules because it is entirely based on hashing tables. It could be explained in simplest form as, U32 filter is a list of records, each consisting of two fields: a selector and an action. The selectors, namely u32,u16,u8 are compared with the currently processed IP packet until the first match and the associated action is performed. The U32 selector contains definition of the pattern,that will be matched to the currently processed packet. Precisely, it defines which bits are to be matched in the packet header and nothing more, but this simple method is very powerful. The simplest type of action would be directing the packet into defined CBQ class. The CBQ algorithm is aware of a predefined bandwidth configured per policy. Recall that each policy has its own queue. As policies are configured, they can be given a maximum allotted bandwidth number, in Kbps . If the scheduler is operating through the CBQ algorithm, as each queue is visited for packet forwarding, the maximum bandwidth of the associated policy is examined. If forwarding ?this? packet from the queue will violate the bandwidth configured within the policy, then the scheduler skips this packet and chooses another packet from another queue of the same priority. This way, the classifier can govern the scheduler not to allow certain applications to go over a pre-defined bandwidth allotment. Finally, it's the job of the scheduler to take packets from the CBQ queue and forward them.

CBQ Features	The CBQ (Class Based Queuing) is a highly flexible queue that allows for a variety of class based queuing policies. The CBQ allows up to eight separate queues, or Classes, that can be configured in various ways to achieve a variety of desired traffic prioritization results. Users may define any number of "policies" and can then assign a policy to each Class or queue within the CBQ class. These policies will determine priority, bandwidth allocation, bounded or isolated Class states, packet burst accommodation, delay parameters and

	queue size, and packet drop precedence that will affect the traffic behavior within the Object. Packet scheduling will subsequently occur based on these policy settings and the additional configuration options that specify output link regulation. Multiple CBQ classes may be used within a configuration to achieve a variety of bandwidth management goals, such as managing more than eight classes of traffic. The heart of the CBQ is the packet scheduler. Depending on the policy assigned to the Class, packets will be scheduled for output first by priority and second by allocation. Classes using the same priority will be scheduled using a Weighted Packet Round Robin algorithm (WRR) based on the allocation values assigned in the policy definition. Individual Classes will be limited to the percentage of assigned bandwidth unless the Efficient Mode is indicated or the Class is allowed to 'borrow' unused bandwidth from the output link.

Conclusion	As a whole Bandwidth management is a powerful value added service. By using this value added service, we hope that the user traffic can be controlled and network resources could be used efficiently.

About ViSolve.com

ViSolve is an international corporation that provides technical services, for Internet based systems, for clients around the globe. ViSolve is in the business of providing software solutions since 1995. We have experience of executing several major projects and we are now completely focused on leading Internet technologies, Testing QA and support. We are committed to the Open source movement and in the same lines we provide free support for products like Linux, Apache and Squid to the user community.

Document Version : 1.0 | Created On : 28-01-02 | Updated On : 29-05-06