Flow control and QoS

Flow control: manage the transmit of traffic between two devices; Flow control is concerned with pacing the rate at which frames or packets are transmitted. The ultimate goal of all flow-control mechanisms is to avoid receive buffer overruns, which improves the reliability of the delivery subsystem. By contrast, QoS is concerned with the treatment of frames or packets after they are received by a network device or end node. wrt queue management and queue schedule.

Ethernet level

Currently, no functionality is defined in the Ethernet specifications for the Pause Opcode to interact with the Priority field. So, the Pause Opcode affects all traffic classes simultaneously

flow control

Pause Opcode

send a MAC control package to broadcast address

http://en.wikipedia.org/wiki/Ethernet_flow_control

Tail-drop and the Pause Opcode often are used in concert. For example, when a receive queue fills, a Pause Opcode may be sent to stem the flow of new frames. If additional frames are received after the Pause Opcode is sent and while the receive queue is still full, those frames are dropped

Qos

802.1q 7 Cos.

1. Network control information
2. Voice applications
3. Video applications
4. Controlled load applications
5. Excellent effort applications

6. Best effort applications

7. Background applications

IP level

Flow control

IP employs several flow-control mechanisms. Some are explicit, and others are implicit. All are reactive. The supported mechanisms include the following:

- Tail-drop
- Internet Control Message Protocol (ICMP) Source-Quench

Despite the fact that ICMP Source-Quench packets can be sent before a queue overrun occurs, ICMP Source-Quench is considered a reactive mechanism because some indication of congestion or potential congestion must trigger the transmission of an ICMP Source-Quench message. Thus, additional packets can be transmitted by the source nodes while the ICMP Source-Quench packets are in transit, and tail-drop can occur even after ICMP Source-Quench packets are sent.

- Active Queue Management (AQM)

o RED

o WRED

o DiffServ Compliant WRED

RFC 2309 defines the concept of AQM. implicit and reactive. Rather than merely dropping packets from the tail of a full queue, AQM employs algorithms that attempt to proactively avoid queue overruns by selectively dropping packets prior to queue overrun. The first such algorithm is called Random Early Detection (RED). More advanced versions of RED have since been developed. The most well known are Weighted RED (WRED) and DiffServ Compliant WRED.

Note that in the most generic sense, sending an ICMP Source-Quench message before queue overrun ocurs based on threshold settings could be considered a form of AQM. However, the most widely accepted definition of AQM does not include ICMP Source-Quench.

- Explicit Congestion Notification (ECN)

When congestion is experienced by a packet in transit, the congested router sets the two ECN bits to 11. The destination node then notifies the source node. When the source node receives notification, the rate of transmission is slowed. However, ECN works only if the Transport Layer protocol supports ECN.

IP QoS

- stateful model:Integrated Services Architecture (IntServ)

The IntServ model is characterized by application-based signaling that conveys a request for flow admission to the network. The signaling is typically accomplished via the Resource Reservation Protocol (RSVP).

- stateless model is the Differentiated Services Architecture (DiffServ).

The DiffServ model does not require any signaling from the application prior to data transmission. Instead, the application "marks" each packet via the Differentiated Services Codepoint (DSCP) field to indicate the desired service level.

3 bits Precedence in ToS.

Routine Set routine precedence (0)

Priority Set priority precedence (1)

Immediate Set immediate precedence (2)

Flash Set Flash precedence (3)

Flash-override Set Flash override precedence (4)

Critical Set critical precedence (5)

Internet Set internetwork control precedence (6)

Network Set network control precedence (7)

IP Precedence 6 and 7 are reserved for network information (routing updates, hello packets, and so on). This leaves 6 remaining precedence settings for normal IP traffic flows

TCP level

TCP flow control

Congestion can be detected implicitly via TCP's acknowledgement mechanisms or timeout mechanisms (as applies to dropped packets) or explicitly via ICMP Source-Quench messages or the ECE bit in the TCP header.

When ECN is implemented, 1)TCP nodes convey their support for ECN by setting the two ECN bits in the IP header to 10 or 01. 2)A router may then change these bits to 11 when congestion occurs. Upon receipt, the destination node recognizes that congestion was experienced. The destination node then notifies the source node by setting to 1 the ECE bit in the TCP header of the next transmitted packet.

the primary TCP flow-control algorithms include:

- slow start,
- congestion avoidance,
- fast retransmit,
- fast recovery.

TCP QoS

TCP interacts with the QoS mechanisms implemented by IP. Additionally, TCP provides two explicit QoS mechanisms of its own: the Urgent and Push flags in the TCP header. The Urgent flag indicates whether the Urgent Pointer field is valid. When valid, the Urgent Pointer field indicates the location of the last byte of urgent data in the packet's Data field. The Urgent Pointer field is expressed as an offset from the Sequence Number in the TCP header. No indication is provided for the location of the first byte of urgent data. Likewise, no guidance is provided regarding what constitutes urgent data. An ULP or application decides when to mark data as urgent. The receiving TCP node is not required to take any particular action upon receipt of urgent data, but the general expectation is that some effort

will be made to process the urgent data sooner than otherwise would occur if the data were not marked urgent.

As previously discussed, TCP decides when to transmit data received from a ULP. However, a ULP occasionally needs to be sure that data submitted to the source node's TCP byte stream has actually be sent to the destination. This can be accomplished via the push function. A ULP informs TCP that all data previously submitted needs to be "pushed" to the destination ULP by requesting (via the TCP service provider interface) the push function. This causes TCP in the source node to immediately transmit all data in the byte stream and to set the Push flag to one in the final packet. Upon receiving a packet with the Push flag set to 1, TCP in the destination node immediately forwards all data in the byte stream to the required ULPs (subject to the rules for in-order delivery based on the Sequence Number field). For more information about TCP QoS, readers are encouraged to consult IETF RFCs 793 and 1122.

Traffic Policying/Shaping

The previous sections covered ways you can queue different flows of traffic and then prioritize those flows. That is an important part of QoS. Sometimes, however, it is necessary to actually regulate or limit the amount of traffic an application is allowed to send across various interfaces or networks.

These features come in two different flavors: rate-limiting tools such as CAR, and shaping tools such as GTS or FRTS.

The main difference between these two traffic-regulation tools is that rate-limiting tools drop traffic based upon policing, and shaping tools generally buffer the excess traffic while waiting for the next open interval to transmit the data.

Cisco IOS QoS software includes two types of traffic shaping: GTS and FRTS. Both traffic-shaping methods are similar in implementation, although their command-line interfaces differ somewhat and they use different types of queues to contain and shape traffic that is deferred.

If a packet is deferred, GTS uses a WFQ to hold the delayed traffic. FRTS uses either a CQ or a PQ to hold the delayed traffic, depending on what you configured. As of April 1999, FRTS also supports WFQ to hold delayed traffic.

Policing literally means to drop excess traffic, shaping on the other hand allows the excess traffic to be queued.

Congestion Avoidance

As discussed previously, WFQ, PQ(priority queue), and CQ(custom queue) mechanisms prioritize the traffic that is of highest importance for bandwidth management.

__Congestion avoidance works on a similar problem from a completely different angle. you avoid further congestion by detecting congestion pattern and dropping packets from different flows, which causes applications to slow the amount of traffic being sent. This avoids what is known as global synchronization, which occurs when many IP TCP flows begin transmitting and stop transmitting at the same time. This is caused by the lack of QoS in a service provider's backbone.

Random Early Detection (RED) is a congestion avoidance mechanism

interested issue

TCP/UDP checksum

http://www.tcpipguide.com/free/t_TCPChecksumCalculationandtheTCPPseudoHeader-2.htm

NAT

compatibility issue

http://www.tcpipguide.com/free/t_IPNATCompatibilityIssuesandSpecialHandlingRequirem.htm

types

- Traditional NAT(unidirectional NAT) is designed to handle only outbound transactions; clients on the local network initiate requests and devices on the Internet send back responses.

- Bidirectional NAT, Two-Way NAT and Inbound NAT. However, in some circumstances, we may want to go in the opposite direction. That is, we may want to have a device on the outside network initiate a transaction with one on the

inside. To permit this, we need a more capable type of NAT than the traditional version. This enhancement goes by various names, most commonly Bidirectional NAT, Two-Way NAT and Inbound NAT. All of these convey the concept that this kind of NAT allows both the type of transaction we saw in the previous topic and also transactions initiated from the outside network.

http://www.tcpipguide.com/free/t_IPNATBidirectionalTwoWayInboundOperation-2.htm

1) static mapping.

2) DNS

- Port-Based NAT, “overloaded” NAT, Network Address Port Translation (NAPT) and Port Address Translation (PAT).

http://www.tcpipguide.com/free/t_IPNATPortBasedOverloadedOperationNetworkAddressPor.htm

- Overlapping NAT or “Twice NAT”

http://books.google.com/books?id=Tvj5V_ypR2kC&pg=RA1-PA193&lpg=RA1-PA193&dq=twice+NAT&source=bl&ots=BhtPb-d7qp&sig=D-hehJ1eDyG-SGjn7bF1BYpuu5M&hl=en

TCP is connection oriented, does it mean packets travel in the same route?

http://books.google.com/books?id=HsCjH_V04tUC&pg=PA303&lpg=PA303&dq=tcp+connection+oriented&source=bl&ots=GEq23rT5fS&sig=ymo0rlbG_nOA2UtYpqBWijsNslk&hl=en

Connection Establishment

To establish a connection, TCP uses a 3-way handshake.

note that this is a virtual connection, not a physical connection. the TCP segment is encapsulated in an IP diagram and can be sent out of order,or lost or corrppted. and then resent. Each may use a different path to reach the destination.there is no physical connection.

MSS vs MTU

http://www.tcpipguide.com/free/t_TCPMaximumSegmentSizeMSSandRelationshiptoIPDatagra-2.htm

http://www.tcpipguide.com/free/t_IPDatagramSizetheMaximumTransmissionUnitMTUandFrag.htm

The default MSS for TCP is 536, which results from taking the minimum IP MTU of 576 and subtracting 20 bytes each for the IP and TCP headers.

MSS can be exchanged in SYN set up. each device at the end point of a connection can has its own MSS independently.

MTU = ip payload + IP head.

a minimum MTU of at least 576 bytes This value is specified in RFC 791, and was chosen to allow a “reasonable sized” data block of at least 512 bytes, plus room for the standard IP header(20) and options(40)

MTU Path Discovery

One of the message types defined in ICMPv4 is the Destination Unreachable message, which is returned under various conditions where an IP datagram cannot be delivered. One of these situations is when a datagram is sent that is too large to be forwarded by a router over a physical link but which has its Don’t Fragment (DF) flag set to prevent fragmentation. In this case, the datagram must be discarded and a Destination Unreachable message sent back to the source. A device can exploit this capability by testing the path with datagrams of different sizes, to see how large they must be before they are rejected.

Note that while intermediate routers may further fragment an already-fragmented IP message, intermediate devices do not reassemble fragments. Reassembly is done only by the recipient device.

fragment

Note that while intermediate routers may further fragment an already-fragmented IP message, intermediate devices do not reassemble fragments. Reassembly is done only by the recipient device. Perhaps the most important one is that fragments

can take different routes to get from the source to destination, so any given router may not see all the fragments in a message.

http://www.tcpipguide.com/free/t_IPMessageFragmentationProcess-2.htm

- In IP fragment, only payload is fragmented. however, in PPP, the whole packet is fragmented, that mean the ppp head is also put into the 1st fragment payload.

- offset is specified in units of 8 bytes;

- each message sent between the same source and destination that is being fragmented has a different identifier. The source can decide how it generates unique identifiers. each other fragment is set to the same Identification value to mark them as part of the same original datagram.

Source: http://manoftoday.wikidot.com/network#toc38