- Traffic Shaping, Policing, and Link Efficiency -

Traffic Shaping vs. Policing

There are two methods for managing traffic that exceeds a specified rate:

- **Traffic shaping**
- **Traffic policing**

These methods are often necessary on the edge separating a customer's network from a provider's network. Providers often force the customer to adhere to a specific policy of *service* (or *committed rate).*

This policy is referred to as the **Service Level Agreement (SLA)** between the customer and provider. Shaping and policing mechanisms differ in how each handles violations of the SLA.

Shaping is usually implemented on the *customer* side, and will **buffer** traffic that exceeds the provider's committed rate. Thus, shaping can slow the traffic rate and siphon out traffic in compliance with the provider's SLA.

Buffering traffic will often create delay and jitter, which can negatively impact sensitive traffic types. Shaping also requires sufficient memory to queue buffered traffic. Shaping provides no mechanism to re-mark traffic that exceeds the committed rate.

Policing is usually implemented on the *provider* side, and will either **drop** or **re-mark** traffic that exceeds the provider's committed rate. TCP traffic that is dropped will be forced to resend, which may result in TCP global synchronization or starvation issues.

Policing can be implemented for both *inbound* and *outbound* traffic on an interface. Shaping can *only* occur on *outbound* traffic on an interface.

(Reference: http://www.cisco.com/en/US/tech/tk543/tk545/technologies_tech_note09186a00800a3a25.shtml)

* * *

Transfer Rate – The Token Bucket System

Cisco describes the regulation of a transfer rate as a **token bucket** system, which is comprised of three parts:

- **Committed Information Rate (CIR)** – specifies the traffic rate dictated by the SLA, measured in *bits per second (bps).*
- **Burst Rate (B_c)** – specifies the amount of traffic to be sent within a given time interval, measured in *bits*.
- **Time Interval (T_c)** – identifies the time interval for each burst, measured in *seconds* or sometimes *milliseconds*.

The CIR is calculated using the formula:

$$\textbf{CIR (bps)} = \textbf{B}_c \textbf{ (bits)} / \textbf{T}_c \textbf{ (seconds)}$$

With a token bucket system, the *bucket* is filled with *tokens,* and each token represents one byte. Thus, to transmit a 50-byte packet, the bucket must contain a *minimum* of 50 tokens. Tokens are consumed as traffic is transferred, and the bucket is refilled with tokens at the speed of the CIR. If the bucket is full, then excess tokens will *spill out* and are wasted. The capacity of the bucket is defined by the burst rate.

If the data (in bytes) to be transmitted is *less* than the number of tokens currently in the bucket, then the traffic is **conforming** to the policy and is (usually) forwarded.

If the data (in bytes) to be transmitted is *more* than the number of tokens currently in the bucket, then the traffic is **exceeding** the policy. This excess traffic can be **shaped** (buffered) or **policed (**dropped or re-marked), depending on the configured policy.

The above describes a **One Token Bucket** system. Some SLA policies allow for bursts that are higher than the normal burst rate (B_c), during periods of non-congestion. This is referred to as the **excess burst (B_e) rate**.

The excess burst rate is an *optional* configuration option, and is defined as a **Two Token Bucket** system. Excess tokens are not wasted, but are instead placed in an excess bucket. The capacity of this bucket is defined by the excess burst rate. Traffic that exceeds the normal token bucket can borrow tokens from the excess bucket – but will usually be dropped first during congestion. Traffic that exceeds the excess bucket is **violating** the policy**.**

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

* * *

Configuring Traffic Shaping

Shaping is usually implemented on the *customer* side, and will **buffer** traffic that exceeds the provider's committed rate. Shaping will thus slow the traffic rate and siphon out traffic in compliance with the provider's SLA.

Cisco IOS devices support multiple methods of traffic shaping:

- **Generic Traffic Shaping (GTS) –** implements shaping on a *per-interface* basis using the *traffic-shape* command.
- **Class-Based Shaping –** implements shaping on a *per-class* basis using the *shape* command within a MQC policy-map.
- **Distributed Traffic Shaping (DTS) –** offloads traffic shaping from the router processor to *Versatile Interface Processors (VIPs)*. DTS is only available on high-end Cisco platforms.
- **Frame Relay Traffic Shaping (FRTS) –** implements Frame-Relay specific shaping mechanisms, such as BECN or FECN. FRTS is only available on a Frame-Relay interface or subinterface, and is covered extensively in the Frame-Relay guide.

To configure basic Generic Traffic Shaping (GTS):

```
Router(config)# interface serial0/0
Router(config-if)# traffic-shape rate 256000 64000 64000
```

The *traffic-shape rate* command is followed by three values, representing:

- The committed information rate (CIR)
- The normal burst rate (B_c)
- The excess burst rate (B_e)

If the normal burst (B_c) and excess burst (B_e) values are equal, then excess burst is not allowed. Specific traffic can be shaped using an access-list:

```
Router(config)# access-list 101 permit tcp any any eq 80

Router(config)# interface serial0/0
Router(config-if)# traffic-shape group 101 256000 64000 64000
```

Class-Based Shaping is accomplished within a policy-map:

```
Router(config)# policy-map MYPOLICY
Router(config-pmap)# class MYMAP
Router(config-pmap-c)# shape 256000 64000 64000
```

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

* * *

Configuring Traffic Policing

Policing is usually implemented on the *provider* side, and will either **drop** or **re-mark** traffic that exceeds the provider's committed rate.

Traffic policing can be implemented directly on an interface using the *rate-limit* command (part of the **Committed Access Rate (CAR)** IOS feature). However, MQC policy-maps are the preferred method for policing traffic:

```
Router(config)# class-map MYMAP
Router(config-cmap)# match protocol http

Router(config)# policy-map MYPOLICY
Router(config-pmap)# class MYMAP
Router(config-pmap-c)# police 256000 64000 128000 conform-action transmit
exceed-action set-dscp-transmit af13 violate-action drop

Router(config)# interface serial0/0
Router(config-if)# service-policy input MYPOLICY
```

The *police* command is followed by three values, representing in order:

- The committed information rate (CIR)
- The normal burst rate (B_c)
- The excess burst rate (B_e) *(optional)*

The above three values are then followed by each *condition,* and its corresponding *action.* The three conditions are as follows:

- *conform-action* – indicates traffic that is sent at *less* than the CIR.
- *exceed-action* – indicates traffic that exceeds the *normal* burst rate.
- *violate-action – (optional)* indicates traffic that exceeds the *excess* burst rate.

The available policing *actions* for each condition include the following:

- *transmit*
- *drop*
- *set-prec-transmit* (sets the IP precedence value)
- *set-dscp-transmit* (sets the DSCP value)
- *set-qos-transmit* (sets the QoS group value)

To view traffic policing statistics:

```
Router# show policy-map interface
```

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

* * *

Compression Overview

Compression increases the bandwidth available on a link, by reducing the size of data. Compression algorithms shrink data by exploiting blank spaces and repeating patterns within headers and payloads. Compression efficiency depends on the algorithm used.

All forms of compression introduce *latency* on the devices performing the compression or decompression. However, end-to-end latency is usually reduced overall, as smaller packets result in shorter serialization delays. Environments that are *not* experiencing congestion should not implement compression, as it will add overhead with little or no benefit.

Compression can be controlled by either *hardware* or *software*. Hardware-controlled compression is more efficient, as software-based compression requires CPU interrupts, thus introduces more latency than hardware-based compression.

There are three general categories of compression:

- **Payload compression**
- **Link compression**
- **Header compression**

Payload compression will compress the payload and all headers, *except* for the Layer-2 header. It is enabled on a per-link basis, and is supported by most serial technologies. Cisco devices support three algorithms for Layer-2 payload compression:

- Stacker
- Predictor
- Microsoft Point-to-Point Compression (MPPC)

Link compression will compress the payload and all headers, *including* the Layer-2 header. This is only supported on point-to-point serial technologies like PPP or HDLC. Multi-access technologies like Frame-Relay or ATM require an uncompressed Layer-2 header to make forwarding decisions.

To enable compression on an interface:

```
Router(config)#  interface serial0/0
Router(config-if)#  encapsulation ppp
Router(config-if)#  compress predictor
```

(Reference: http://www.cisco.com/en/US/tech/tk713/tk802/technologies_tech_note09186a00801b3b86.shtml)

* * *

Header Compression

Header compression *will not* compress the payload, and thus is less hardware intensive than payload compression. Instead, the Layer-3 and Layer-4 headers of a packet will be compressed. This is most efficient for traffic that has a small payload, such as VoIP. The ratio of header to payload size is very large with such traffic, thus introducing considerable overhead.

The two most common types of header compression are as follows:

- **TCP Header Compression**
- **RTP Header Compression (cRTP)**

To enable outgoing TCP Header compression on an interface:

Router(config)# *interface serial0/0*
Router(config-if)# *ip tcp header-compression*

To enable outgoing TCP Header compression *only* if incoming TCP traffic on that interface *is also compressed*:

Router(config)# *interface serial0/0*
Router(config-if)# *ip tcp header-compression passive*

RTP header compression (cRTP) compresses the *RTP*, *UDP*, and *IP* headers of a packet. cRTP can reduce the collective header size from 40 bytes to 2 bytes.

Both TCP and RTP header compression can be configured directly on an interface, or within a MQC policy-map for a specific class:

Router(config)# *class-map VOICE*
Router(config-cmap)# *match protocol rtp audio*
Router(config)# *class-map HTTP*
Router(config-cmap)# *match protocol http*

Router(config)# *policy-map MYPOLICY*
Router(config-pmap)# *class VOICE*
Router(config-pmap-c)# *compression header ip rtp*
Router(config-pmap)# *class HTTP*
Router(config-pmap-c)# *compression header ip tcp*

Router(config)# *interface serial0/0*
Router(config-if)# *service-policy output MYPOLICY*

(Reference: http://www.cisco.com/en/US/docs/ios/12_2t/12_2t13/feature/guide/fthdrcmp.html)

* * *

Link Fragmentation and Interleaving (LFI)

Link fragmentation and **interleaving (LFI)** prevents large packets from causing excessive queuing and serialization delays for smaller packets, such as VoIP packets. If a large packet is queued first, the smaller packet(s) must wait until the large packet is serialized in its entirety.

Fragmentation and interleaving operate in conjunction with each other, and provide no benefit if implemented separately. Large packets are *fragmented*, or split into smaller packets, and assigned sequence numbers for reassembly on the receiving end. This allows smaller packets, such as VoIP traffic, to be *interleaved* within the fragmented larger packets. This reduces latency, as the sensitive traffic is no longer forced to wait until the larger packets are transmitted in their entirety.

LFI is supported on Multi-link PPP (MLP), Frame-Relay, and ATM links. To configure LFI for Multi-link PPP:

Router(config)# *interface serial0/0*
Router(config-if)# *encapsulation ppp*
Router(config-if)# *ppp multilink*
Router(config-if)# *multilink-group 1*

Router(config)# *interface serial0/1*
Router(config-if)# *encapsulation ppp*
Router(config-if)# *ppp multilink*
Router(config-if)# *multilink-group 1*

Router(config)# *interface Multilink1*
Router(config-if)# *ppp multilink*
Router(config-if)# *multilink-group 1*
Router(config-if)# *ppp multilink interleave*
Router(config-if)# *ppp multilink fragment-delay 20*

The *ppp multilink interleave* command enables LFI.

The *ppp multilink fragment-delay* command is used to specify the maximum desired delay for traffic, measured in milliseconds. LFI will attempt to fragment packets accordingly to meet this delay requirement.

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

* * *

QoS for VPNs (qos pre-classify)

Tunnels encapsulate the original IP headers with new headers. By default, Cisco IOS devices will copy the ToS field from the original header into the tunnel header. Thus, if QoS is applied based on ToS values, such as IP Precedence or DSCP, no additional QoS configuration is required for tunneled links.

If QoS is applied based on *other* parameters within the IP header, such as port or address, then the Cisco **QoS for VPNs** feature is required. This feature is supported for the following tunnel types:

- Generic Routing Encapsulation (GRE)
- IP in IP (IPIP)
- Layer 2 Tunneling Protocol (L2TP)
- Layer 2 Forwarding (L2F)
- Point to Point Tunneling Protocol (PPTP)
- Internet Protocol Security (IPSec)

The QoS for VPNs feature utilizes the *qos pre-classify* command. This command forces the IOS to create a copy of the original IP packet, including the original headers. QoS policies are then applied based on the headers of the *copied original* packet, instead of the *encapsulated (tunneled)* packet.

The following describes how to apply the MQC *service-policy* command in conjunction with *qos pre-classify:*

- Applying the *service-policy* to a **tunnel** interface *without* the *qos pre-classify* command will classify based on the *pre-tunnel* header.
- Applying the *service-policy* to a **physical** interface *with* the *qos pre-classify* command will classify based on the *pre-tunnel* header.
- Applying the *service-policy* to a **physical** interface *without* the *qos pre-classify* command will classify based on the post-tunnel header.

Applying a *service-policy* to a physical interface will affect all tunnels off of that interface.

(Reference: http://www.cisco.com/en/US/docs/ios/12_2t/12_2t2/feature/guide/ftqosvpn.html; http://www.cisco.com/en/US/tech/tk543/tk545/technologies_tech_note09186a008017405e.shtml)

* * *

Configuring QoS for VPNs (qos pre-classify)

For GRE tunnels, the *qos pre-classify* command is configured on the tunnel interface:

```
Router(config)# interface tunnel1
Router(config-if)# qos pre-classify
```

For IPSec tunnels, the *qos pre-classify* command is configured on the *crypto map:*

```
Router(config)# crypto map MYTUNNEL 10 ipsec-isakmp
Router(config-crypto-map)# set peer 172.16.5.1
Router(config-crypto-map)# set transform-set MYTRANSFORM
Router(config-crypto-map)# match ip address 101
Router(config-crypto-map)# qos pre-classify
```

If a GRE tunnel is being used with IPSec, the *qos pre-classify* command must be configured on both the tunnel interface and the *crypto map.*

Control Plane Policing (CoPP)

Control Plane Policing (CoPP) is a mechanism to protect the control plane of Cisco IOS switches and routers from Denial of Service (DoS) attacks. A concerted stream of management traffic sent to a switch/router processor can keep CPU utilization at 100%. This may result in slow response times, missed routing updates, and queue saturation.

CoPP is managed using MQC. Instead of applying the *service-policy* to an interface, it is applied to the control-plane, which is treated as its own entity.

```
Router(config)# class-map MYMAP
Router(config-cmap)# match access-group 101

Router(config)# policy-map MYPOLICY
Router(config-pmap)# class MYMAP
Router(config-pmap-c)# police 80000 conform transmit exceed drop

Router(config)# control-plane
Router(config-cp)# service-policy input MYPOLICY
```

(Reference: http://www.cisco.com/en/US/docs/ios/12_3t/12_3t4/feature/guide/gtrtlimt.html)

* * *