VYATTA, INC. | Application Note

Configuring QoS for VoIP Networks

Open Source Networking

http://www.vyatta.com

Executive Summary

This application note describes ways to configure Quality-of-Service (QoS) parameters on Vyatta for VoIP services in an IP Telephony network. Currently, Vyatta does not support QoS configuration in its CLI and Web GUI. This application note will describe how users can configure Vyatta outside the scope of the Vyatta management tools to provision QoS. In the near future, Vyatta will support integrated QoS configuration in its management tools.

Why QoS?

QoS is one of the most obfuscated technologies used in networking. Different users, vendors, and standards committees have distinct interpretations of what QoS is and how it should be implemented. Recently, Vyatta conducted a poll on its website and asked users to opine on what they would like QoS to be used for in their networks:

What is the most important application for QoS in your network?

- *Voice over IP – 56%*
- *Video – 9%*
- *Real-time data – 7%*
- *Limit apps on WAN – 14%*
- *Limit apps on LAN – 8%*
- *Other – 6%*

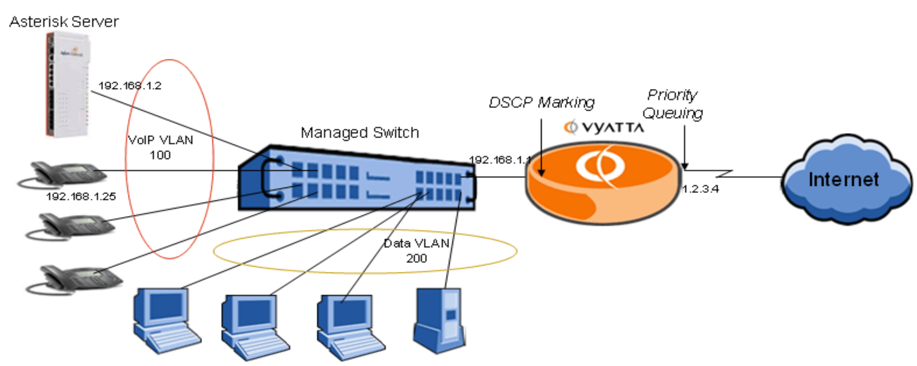
An overwhelming majority of respondents (56%) noted VoIP as the primary driver for network QoS. This corroborates anecdotal conversations Vyatta has had with its user-base.

Using the Linux Kernel for VoIP Prioritization

Vyatta is based on Linux which offers many ways to instrument QoS on the system. The Linux kernel's inherent queuing and packet manipulation capabilities provide a simple way to achieve preferential treatment for VoIP traffic. By identifying VOIP packets at an ingress interface and intelligently queuing them at the egress/outbound interface (typically for traffic exiting the LAN to WAN), we can detect and enforce prioritization of VoIP traffic.

Figure 1 shows a small-medium enterprise network with an Asterisk SIP server/media gateway and Vyatta edge router. In most SMB applications, a simple packet marking and priority queuing mechanism will ensure that VoIP quality is not compromised by other high-bandwidth data applications.

Figure 1: An SMB VoIP Deployment Scenario



VOIP PACKET MARKING

The default queuing discipline (qdisc) on any Linux interface is PFIFO_FAST. While the name PFIFO_FAST suggests a simple First-In-First-Out queuing discipline, it provides more than that. PFIFO_FAST allows for 3 queues each of which represents a different priority level. A packet is placed into a queue depending upon the value of the TOS byte (or DSCP field) in the IP header of a packet.

Let us set up packet marking such that VoIP packets get preferential treatment over other network traffic at an output interface. Moreover, this allows VoIP packets to carry the correct DSCP for upstream routers and switches.

Today, Vyatta does not provide an integrated CLI to set DSCP bits. Users are advised to configure IPTables rules from the bash shell, outside the scope of the Vyatta CLI (login as "root" to access the bash shell).

The following IPTables rules allow DSCP packet marking for VoIP packets:

1) Mark SIP signaling packets

```
iptables -t mangle -A POSTROUTING -p udp --dport 5060 -j
DSCP --set-dscp 46
```

2) Mark RTP media packets

```
iptables -t mangle -A POSTROUTING -m helper --helper sip -j
DSCP  --set-dscp 46
```

[DSCP 46 indicates Expedited Forwarding. Some implementations prefer the VoIP signaling packets to be set to DSCP 26 (Assured Forwarding Class 3 Gold) or 34 (Assured Forwarding Class 4 Gold)].

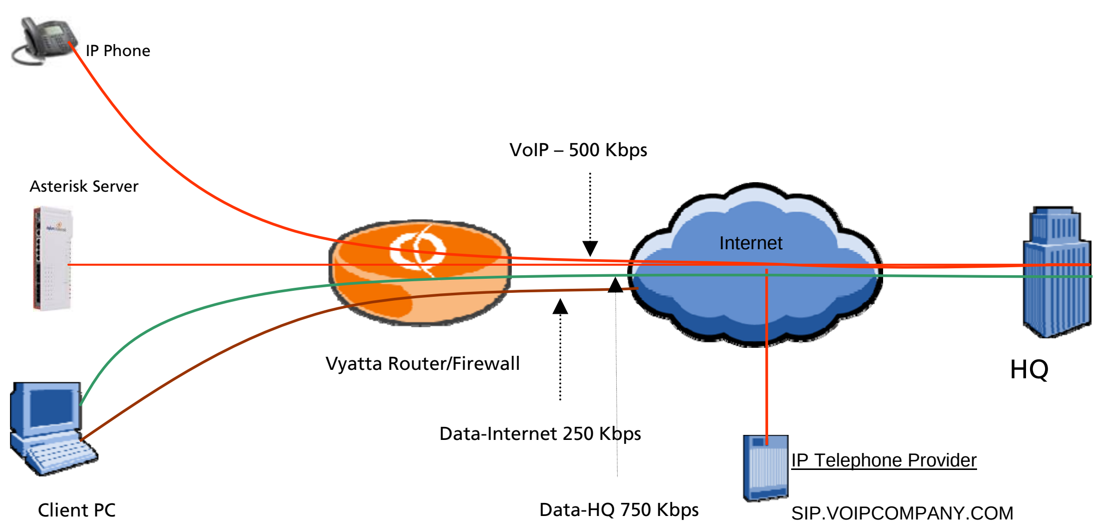
Note that an Asterisk server or the VoIP media gateway system in your network can also be configured to mark the DSCP field on the VoIP signaling packets. The Asterisk server can also instruct the IP phones to mark the VoIP media packets.

TRAFFIC SHAPING AND BANDWIDTH MANAGEMENT

Now that we have marking and simple priority queuing working, let us see how we can manage bandwidth requirements and queuing for multiple applications on a Vyatta router. The following scenario shows how users can use a class-based queuing discipline such as Hierarchical Token Bucket (HTB) to manage QoS for different applications. HTB ensures that the amount of bandwidth provided to each class is at least the minimum of the amount it requests and the amount assigned to it. When a class requests less than the amount assigned, the remaining (excess) bandwidth is distributed to other classes which request service.

Take the example of a network manager who wants to have a simple policy of sharing an egress ethernet WAN link eth1, capable of providing 1.5 Mbps of upstream bandwidth, between data and VoIP in a branch network. The manager wants to allocate 1/3rd of the upstream bandwidth to VoIP and the remaining bandwidth to data. The manager also wants to allocate 3/4th of the data bandwidth for traffic bound for the corporate headquarters. Note that any unused bandwidth can be used by any class which needs it (in proportion of its allocated share).

Figure 2: SMB Traffic classification scenario



The above classification leads to three traffic classes as shown in Figure 2 in the branch network – *VoIP* (500 Kbps), *Data-HQ* (750 Kbps) and *Data-Internet* (250 Kbps). The following steps using the Linux traffic control tool *"tc"* will help us achieve the network manager's QoS goals for the different kinds of traffic.

1) Create an HTB qdisc for ethernet interface eth1 and give it an identifier referred to as handle 1:

```
tc qdisc add dev eth1 root handle 1: htb default 30
```

The **`default 30`** in the above configuration means that any traffic that is not otherwise classified will be assigned to class 1:30.

2) Configure the classes for the different traffic streams for *VoIP, Data-HQ and Data-Internet*. Creating a root class (1:1 as shown below) under the htb qdisc allows the child classes to share unutilized bandwidth in the pipe.

```
tc class add dev eth1 parent 1: classid 1:1 htb rate 1.5mbit
ceil 1.5mbit

tc class add dev eth1 parent 1:1 classid 1:10 htb rate
500kbit ceil 1.5mbit

tc class add dev eth1 parent 1:1 classid 1:20 htb rate
750kbit ceil 1.5mbit

tc class add dev eth1 parent 1:1 classid 1:30 htb rate
250kbit ceil 1.5mbit
```

3) Create filters using IP DSCP and destination IP address to match traffic corresponding to the *VoIP* and *Data-HQ* traffic respectively. Any packet not classified by the two rules will be put in class 1:30 or *Data-Internet*.

```
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32
match ip dscp 46 0xff flowid 1:10

tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32
match ip dst 5.6.7.8 flowid 1:20
```

4) One can optionally attach queuing disciplines with configurable queue lengths to the above traffic classes to control latency as shown below.

```
tc qdisc add dev eth1 parent 1:10 handle 20: pfifo limit 5
```

The above examples show how the Linux traffic control and IPTables instruments can be used to deliver prioritized VoIP treatment in an SMB or enterprise branch environment. For understanding HTB configuration parameters, users are encouraged to visit the HTB user guide at http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm#intro

For users looking for more advanced QoS techniques, Class Based Queuing (CBQ) provides a method to queue as well as shape packets. However, as often is the case with advanced and complex technologies, slight misconfigurations may cause CBQ to work inefficiently and inaccurately at times. This application note will not go into the details of CBQ configuration. Interested users are encouraged to visit the Linux Advanced Routing and Traffic Control (LARTC) website at http://lartc.org/howto/lartc.qdisc.classful.html for more information.

Conclusion

QoS provides a strong example where the vast Linux feature set can be harnessed to provide functionality required by customers on Vyatta's increasingly integrated solution. Today, users can make use of the native Linux kernel QoS mechanisms, through the standard IPtables and tc tools, to configure the appropriate network behavior to protect VoIP and other traffic from congestion. Over the next few releases, Vyatta will integrate the traffic control instruments described in this paper into its configuration and management tools. This will provide users with a way to easily administer and provision VoIP in small-medium business or enterprise branch environments.

References

1) Detailed Linux QoS treatise at LARTC: http://www.lartc.org/

2) HTB manual and user guide: http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm#intro

3) CBQ examples in Asterisk environments: http://www.VoIP-info.org/wiki/view/Asterisk+QoS

4) Diffserv Architecture RFC 2474 & RFC 2475 (http://www.ietf.org)

FEEDBACK

Have comments on this paper? Send them to feedback@vyatta.com