Known Limitations

This section discusses some of the known limitations of Qosium and how to deal with them.

Table of Contents

1. Performance Considerations

1.1. Qosium Probe's Serving Capabilities

A single Qosium Probe can support theoretically up to about 32000 simultaneous measurements. However, in practice, the number of simultaneous measurements is much lower. Typically the limiting factor is the memory. In a normal PC, you can perform typically hundreds of measurements, but not thousands. You can affect this by packet capturing parameterization at Probe.

1.2. Packet Rates

The higher the packet rate of the measured traffic, the more computational effort measuring it requires. The meaning of packet sizes is negligible. In a typical PC, you can easily perform a single-point measurement of streams of 100000 pkts/s, meaning that a 1 Gbit/s link is filled already with traffic. On more powerful desktop computers and servers, measuring streams of 500000 pkts/s or higher should work fine. Then again, in low-powered embedded devices, the limits can be much lower, e.g., 5000 pkts/s. Hence, the measurement limits are heavily dependent on the device where you run Qosium Probe.

Certain measurement features also affect the reachable performance. A two-point measurement consumes roughly 2.5 times more computational resources than a single-point one. Then, payload-based packet identification is about two-times more laborious than IPv4 or RTP identification. Still, however, the measurable packet rate does not always decrease accordingly. The reason for this is that Qosium Probe is able to exploit multiple processor cores.

1.3. Taking Full Captures and Collecting Packet-Level Results

Full packet captures from a remote measurement point generate a QMCP load about equal to the measured traffic stream. Thus, it is recommended to take full captures only when needed.

Packet-level measurement results also increase the QMCP load between the measurement controller and the primary measurement point. The increase is clearly less when compared to the full capture but still significant. The overhead increases as a function of the measured traffic packet rate. Besides, collecting packet-level statistics is more laborious for the measurement controller than sticking with the average results. Thus, case by case, consider that do you really need the packet-level statistics or would the average results be enough.

2. Network Protocol Support

Qosium can measure the QoS of any application or service using Ethernet, IPv4, or IPv6. Currently, IPv6 support is developed only for the measurement, and the control channel for the measurement happens over IPv4 only at the moment. Bringing full IPv6 support is on the roadmap, but its implementation pace depends on the need. Thus, if you desire full IPv6 support, ask Kaitotek support!

3. Large Send Offload and Large Receive Offload at NIC

3.1. Support in General

Most of the modern NICs are capable of performing some operations normally done by the IP stack of the operating system. Handing calculation from the IP stack to the NIC is called *offloading*. Example methods are Large Send Offload (LRO) and Large Receive Offload (LSO). Qosium can handle most of the LRO and LSO cases without problems. A known limitation still is that the MTU sizes between the measurement

points should match. If not, Qosium's offloading support likely fails.

Some things are worth of noting related to the offloading support of Qosium. First, Qosium's offloading support is available only for the following Packet Identification Methods:

- Pure Payload Based ID (the safest)
- IPv4 ID (works in most cases)

Then, in Linux, with the interface *any*, through which all traffic from all the interfaces is seen, the offloading support of Qosium does not work. The reason is simple: in order for the handling to work, Qosium needs to operate with the real network interface directly, i.e., the one which performs the possible segmentation or reassembly of the packets. With the interface *any*, this is not the case.

Finally, in the tests it has been noticed that some setups can mess up the offloading operations in such way that it affects the measurement. This kind of operation has been noticed to happen at least in Ubuntu 18.04 when the machine is used as a bridge. Also, Windows 10 machines with some USB Ethernet adapters seem also cause similar effects. However, the found cases affect only when the *IPv4 ID* method is selected as the **Packet Identification Method**. Operating with *Pure Payload Based ID* should work fine.

3.2. Turning Offloading Off

In some special cases there might be a need to turn the offloading functions off. This section tells how to do that in different platforms.

3.2.1. Windows

In Windows, you can change the adapter setting through network adapter list (e.g., in Windows 10: *Control panel => Network and Internet => Network Connections*). Open the network adapter properties, and there click *Configure*, which opens up the accurate parameters. There, in leaf *Advanced*, you can change the offloading settings. The changed settings should be permanent.

3.2.2. General Linux

In many Linux based systems, the command line tool Ethtool can be used to manipulate the interface options. First, offloading settings can be checked by a command

```
ethtool -k <interface>
```

Then, you can manipulate the offloading options with an option -K. Remember that if, e.g., segmentation and reassembly offloading is enabled, one can lose some accuracy in the link level delay measurement. Regarding the measurement, harmful offloading features are: Generic Receive Offload (GRO), Generic Segmentation Offload (GSO), and TCP Segmentation Offload (TSO). Disabling these can be done with the following Ethtool command:

```
sudo ethtool -K <interface> gro off gso off tso off
```

3.2.3. Making the Setting Persistent in Ubuntu

To make reassembly setting persistent in Ubuntu, modify the /etc/network/interfaces file and add a post-up line to the network interface to be measured. Replace with the real network interface name in the following example:

```
auto <interface>
iface <interface> inet <static/dhcp>
[...]
post-up /sbin/ethtool -K <interface> gro off tso off
```

3.2.4. Making the Setting Persistent in CentOS/RHEL 7

To make reassembly setting persistent in CentOS and RHEL 7 Modify the `/etc/sysconfig/network-scripts/ifcfg-[interface]` file and add `ETHTOOL_OPTS` as follows:

```
ETHTOOL_OPTS="-K ${DEVICE} gro off tso off
```

If Ethtool is not available, install it with a command `yum install ethtool`.

3.2.5. Making the Setting Persistent in Most of the Linux Systems

Add the Ethtool command to be always run at the start-up of operating system. There are multiple options to do this. Replace with the real network interface name.
Crontab: Type `sudo crontab -e` and add the following line to the end of the file:

```
@reboot  /sbin/ethtool -K <interface> gro off tso off
```

`/etc/rc.local`: open the file with the root rights and add

```
/sbin/ethtool -K <interface> gro off tso off
```

to the end of the file before exit 0.

4. Measuring over Unreliable Network Paths

QMCP stream, as it is currently carried over TCP, even though it is light, is also as vulnerable to network conditions similarly as the measured other streams. Hence, e.g., if the network is congested, QMCP packets might be occasionally lost, and the real-time measurement could become unreliable. Besides occasional delays (no results), the standard TCP connections over unreliable links may break. Thus, if this happens, the two-point measurement can seize. Under very bad network conditions, it might be that the measurement cannot be made. There are at least three ways to cope with this challenge:

The first thing to do is to make sure that the *Robust QMCP Mode* is active (by default, it is). When used, Qosium detects a broken link quickly and is able to recover from it. Qosium will tolerate connection breaks of length defined by parameter **Robust QMCP Timeout**. However, notice that during a measurement, the breaks may occasionally be long, meaning that no results are got during that period of time. Nevertheless, the *Robust QMCP Mode* allows measurements to recover when measuring long periods of time over unreliable links.

If the connection between the Qosium Probes is very bad, the QMCP traffic can also be routed through a reliable connection if possible in the measurement scenario. This is the optimal case, since then the QMCP traffic is not present at all in the measured connection.

If the connection is not good but not very bad either, try to increase the value of the *Packet loss timer* parameter. This might be enough to cope with the unreliable link, but the down side is that the packet loss result calculation is delayed accordingly.

5. On the Supported Network Interfaces

Qosium captures packets directly from the network interface using Pcap. However, some network interfaces might not work properly with Pcap. Thus, if there are problems with capture interfaces, please ask Kaitotek support for assistance.