Cisco IOS IP SLAs offers many different operations, but a majority is using one of them: UDP Jitter. One single operation can capture at the same time the delay, jitter and loss in both directions, and report a comprehensive amount of information.

All of IP SLAs so-called operations are sending synthetic traffic though the network to measure its ability to deliver packets. Those packets are additional data, crafted for the only purpose to actively measuring various metrics such as delay, loss, and delay variation in a consistent, systematic and reproducible manner.

The purpose of this document is to help you understand how the operation is working, what kind of information is reported, how it is calculated, as well as to avoid some common pitfalls.

Contents

- 1 Example of a UDP Jitter Operation Output
- 2 Interpreting IP SLA Jitter Operation Output
  - 2.1 Accuracy
  - 2.2 Granularity
- 3 Delay
- 4 Packet Loss
- 5 Jitter
  - 5.1 Positive and Negative Jitter
  - 5.2 The Good News and the Bad News
- 6 Interpretation Starts with Knowing What to Expect
  - 6.1 UDP Jitter Configuration
  - 6.2 Spectrum of Test
  - 6.3 The effect of the Number of Packets
  - 6.4 The effect of the Frequency
  - 6.5 The effect of the Interval
  - 6.6 What if I Have Different Quality of Service (QoS)?
- 7 Selecting Measurement Points
  - 7.1 Close to the end user
  - 7.2 Hub and Spoke
  - 7.3 End to End
  - 7.4 Selecting the Most Important Measurement Points
- 8 Performance
- 9 UDP Jitter Operation vs. RTP VoIP Operation

Example of a UDP Jitter Operation Output

```
c38d14-1#sh ip sla sta 1 details
IPSLAs Latest Operation Statistics
IPSLA operation id: 1
Type of operation: udp-jitter
        Latest RTT: 1 milliseconds
Latest operation start time: 15:16:59.005 UTC Thu Jun 25 2009
Latest operation return code: OK
RTT Values:
        Number Of RTT: 1000             RTT Min/Avg/Max: 1/1/2 milliseconds
Latency one-way time:
        Number of Latency one-way Samples: 0
        Source to Destination Latency one way Min/Avg/Max: 0/0/0 milliseconds
        Destination to Source Latency one way Min/Avg/Max: 0/0/0 milliseconds
```

```
        Source to Destination Latency one way Sum/Sum2: 0/0
        Destination to Source Latency one way Sum/Sum2: 0/0
Jitter Time:
        Number of SD Jitter Samples: 999
        Number of DS Jitter Samples: 999
        Source to Destination Jitter Min/Avg/Max: 0/1/1 milliseconds
        Destination to Source Jitter Min/Avg/Max: 0/1/1 milliseconds
        Source to destination positive jitter Min/Avg/Max: 1/1/1 milliseconds
        Source to destination positive jitter Number/Sum/Sum2: 29/29/29
        Source to destination negative jitter Min/Avg/Max: 1/1/1 milliseconds
        Source to destination negative jitter Number/Sum/Sum2: 29/29/29
        Destination to Source positive jitter Min/Avg/Max: 1/1/1 milliseconds
        Destination to Source positive jitter Number/Sum/Sum2: 22/22/22
        Destination to Source negative jitter Min/Avg/Max: 1/1/1 milliseconds
        Destination to Source negative jitter Number/Sum/Sum2: 22/22/22
        Interarrival jitterout: 0       Interarrival jitterin: 0
        Over thresholds occurred: FALSE
Packet Loss Values:
        Loss Source to Destination: 0           Loss Destination to Source: 0
        Out Of Sequence: 0      Tail Drop: 0    Packet Late Arrival: 0
Packet Skipped: 0
Voice Score Values:
        Calculated Planning Impairment Factor (ICPIF): 0
        Mean Opinion Score (MOS): 0
Number of successes: 120
Number of failures: 0
Operation time to live: Forever
Operational state of entry: Active
Last time this entry was reset: Never
```

Interpreting IP SLA Jitter Operation Output

IP SLA measures jitter and delay per packet, and display or report an aggregate of those values. As of today, there is no way to look at the raw, per packet information. Instead, a global view is provided that includes minimums, maximums and averages of different metrics.

First of all, let us summarize the metrics reported in a typical **show** command:

- Delay (per direction, and round trip)
- Packet Loss (per direction)
- Jitter (per direction, positive and negative)

It is important to note that under certain conditions, IP SLAs will not be able to report some values. These conditions will be discussed later in this document, but here are the two metrics that are not always available:

- One-way delay
- In which direction packets are lost

Accuracy

IP SLAs uses a software time stamping mechanism on both the sender and the responder, reporting time-based values generally accurate at +- 1 ms. However, this is not always guarantee, and a lot of factor will influence this value including but not limited to the platform, the IOS image, the features enabled, the type and volume of traffic, and so on.

External influence such as the platform itself, the interface type, other features running at the same time, the forwarding load, the system temperature, and so on can lead to better or worst results. It is advised that you

perform you own qualification tests in your particular environment if the accuracy is very important for you. Any system that does not have access to hardware timestamping will suffer from inaccuracies like this.

In practice, values accurate within a few milliseconds are fine because typically, the applications do not have strict latency/delay variation requirements. For instance, VoIP delay is to remain at a maximum of 120 ms one way, and 30 ms jitter. An accuracy a few milliseconds in enough in this case.

Granularity

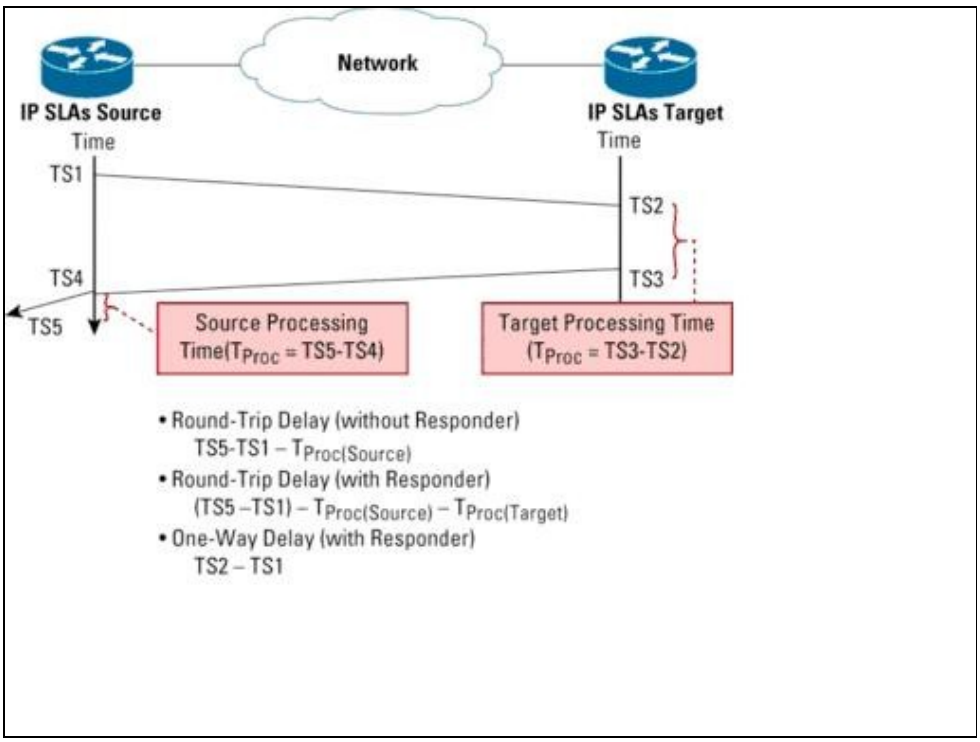
IP SLAs reports all values in milliseconds. It is possible to report values in microseconds instead via the optional command **precision microsecond**.

It is important to understand that this function will not add more accuracy, but will display the results with more granularity.

Delay

The delay values are reported by IP SLAs after the operation has finished execution, and will report at least the round trip time delay in milliseconds or microseconds (depending the configuration).

As you can see in the below diagram, multiple timestamps are collected at both the source and the destination to have a clear picture of the transit time, and processing time spent on the responder. Those timestamps are then used to compute round-trip time delay, and if the clocks were synchronized, one-way delay.



If this seems quite easy and accurate for round trip delay, however, for one-way delay values the situation is rather different: the time on the source and destination router has to be exactly the same, at least as much as possible. The smallest drift in clock synchronization will measure a direction as slower than reality, while the other direction will be measured as faster than reality. The round trip time will remain unchanged.

Fluctuating one-way delays with constant round-trip time is a very good indication that the problem lies in the clock distribution. For best results, it is recommended that every measurement point (sender and

responder) be connected to a local reference clock, such as GPS sourced clock. NTP can be used through a local network, or back-to-back link, with excellent accuracy.

It is recommended that you do not use the network you are monitoring as a transport for NTP time synchronization. Indeed, the measurements accuracy are impacted by the state of the network, and this is precisely what you are measuring. As noted in RFC2330: First, NTP's accuracy depends in part on the properties (particularly delay) of the Internet paths used by the NTP peers, and these might be exactly the properties that we wish to measure, so it would be unsound to use NTP to calibrate such measurements.

Packet Loss

Five types of packet loss or assimilated events can be measured with IP SLA:

- Packet loss in the source to destination (packetLossSD)
- Packet loss in the destination source (packetLossDS)
- Tail Drop: we know it has been dropped, but we do not know in which direction. This is when the last packet(s) of the test streams were dropped, because in this case, we do not receive the sequence numbers. In older releases, this is called *Packet MIA* for *missing in action*. In the MIB, the notation PacketMIA is still in use.
- Packet Late Arrival: the packet did arrive, but so late that the underlying application probably considered it as dropped, or at least not useful. Think about a VoIP application. If one packet arrives much later than expected, it is too late because the conversation keeps going. This packet is assimilated to a drop.
- Packet Misordering: the packet arrived but not in the right order. This may or may not be considered as a packet drop. (packetOutOfOrder)

The cool thing is the power that lies behind those numbers. Different values can be calculated the way you want it. For instance the total number of packets dropped is:

packetDropped = RTTMonPacketLossSD + RTTMonPacketLossDS + RTTMonPacketMIA

The total percentage of packets that have dropped during the instance is:

drop_rate_%age = 100 * packetDropped / (RTTMonNumOfRTT + packetDropped)

Many other values can be calculated, and that is entirely up to you to decide what parameters are important.

Jitter

IP SLA will give you a tremendous amount of details with the jitter results, but it is up to you to use them appropriately. There have been many questions from the past years, and also fake assumptions on this topic. The very purpose of this document is to draw some general guidelines for a reliable, reproducible and meaningful jitter interpretation.

Any given path in a network is having a minimum delay induced by different components on which we have little or no influence: transmission delay, switching time, queuing time, serialization delay and so on. Even under the best possible conditions there is always a minimum delay.

Positive and Negative Jitter

If the conditions change (for instance, there is a sudden burst on the network and queues start to build on an interface) then this delay will increase. The transit time between two consecutive packets will not be the

same anymore: the second packet will have to go through a longer queue, spending more time, and generating positive jitter.

Once this burst is over, the queue will progressively reduce, reversing the situation. Out of two consecutive packets, the second one will spend less time in the queues, and will therefore generate negative jitter.

The Good News and the Bad News

That being said, it is clear that positive jitter is an indication that the situation is actually getting worse, less favorable for the underlying application. Positive jitter is *bad news*, and indication that the delay is increasing and the network has some level of congestion. On the other side, negative jitter is a sign that the network is actually getting healthier, les busy, the queues are reduced, so in that respect, this is *good news*.

Now that we have this information in perspective, it is clear that the *bad* jitter is actually the positive while the *good* jitter is actually negative. But for some applications, any jitter big enough can cause harm.

With the following set of values, you will be on your way for a meaningful interpretation of your data:

- Percentage of packets that had positive jitter: it estimates how many packets are actually introducing jitter. If a large ratio of the population introduces jitter it might not be a problem as long as the introduced jitter per packet remains low (see next metric).
- Average jitter per packet that had positive jitter: it gives you an idea on how much jitter is introduced once a positive jitter is experienced. A big jitter increase per packet means a large latency dynamic in your network which is not good.
- Percentage of packets that had negative jitter: it is an estimation of how many packets it takes to the network to compensate the jitter. If this is much higher than the percentage of packets that had positive jitter, it may be a sign that your network is having a hard time to absorb the traffic bursts. Generally speaking they should be within the same range.
- Average jitter per packet that had negative jitter.

Interpretation Starts with Knowing What to Expect

Loss, jitter, delays, precision, accuracy, all that only makes sense in the light of the supporting application. What are the services and applications you are transporting? What are their requirements? What are they sensible to? What are the most important performance limiting factors? Where are the important measurement points?

There is no common answer, and every case is different. A good SLA deployment will start by knowing exactly what to expect on the network, what kind of quality and availability?

UDP Jitter Configuration

Various parameters will impact what the IP SLAs UDP Jitter operation is doing, and how accurately this will fit the reality. The operation itself can be configured and tuned using several parameters. These parameters will be discussed later in this document. First, the notion of *Spectrum of Test* will be introduced.

Spectrum of Test

The operation's spectrum of test (SoT) is defined as a percentage of the time the operation is actually sending traffic, and testing the network. The default UDP Jitter operation sends packets for 200 ms every minute, and that is a SoT of 0,3%. In practice, anything interesting happening in your network has 99,7% chance to remain unnoticed.

Getting a better network view for fine-grain statistics and pro-active troubleshooting will require a larger spectrum of test.

The effect of the Number of Packets

The parameter *num-of-packets* changes the total number of packets sent during an iteration of the configured operation. The more packets are sent, the larger the spectrum of coverage. More packets also mean more traffic and more bandwidth.

The effect of the Frequency

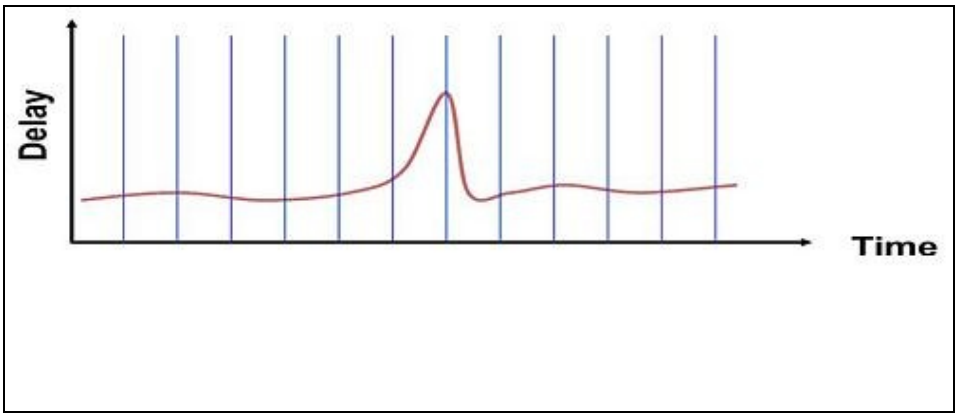
The *frequency* parameter is expressed in second, and is the amount if time between two instances of a running operation. The default value is sixty seconds, and therefore, the operation will be executed every minute. Setting a smaller frequency value increases the frequency and the spectrum of test.

If it is true that increasing the number of packet or the frequency both lead to a larger spectrum of coverage, the results are much more diluted if you increase the number of packets. It is perfect if the results are to compare performance over a long period of time (like every hour or more). For real-time troubleshooting, SLA validation, and so on a fine-grained performance measurement is more desirable and it is recommended to increase the frequency.

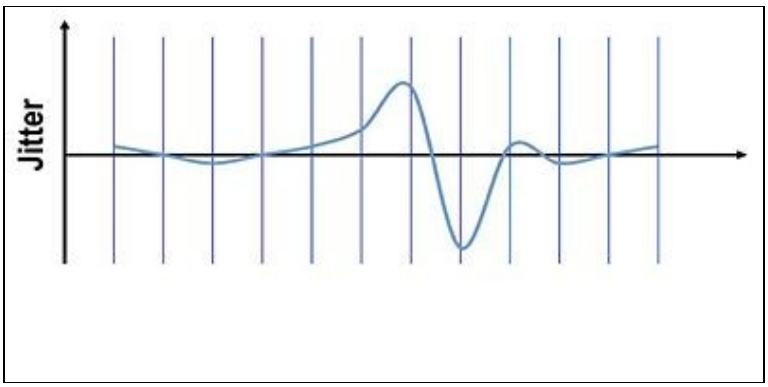
The effect of the Interval

One can send a test packet every second, or every ten seconds, or even fifty packets per second. A test packet is a test packet, but the interval between two packets plays an important role.

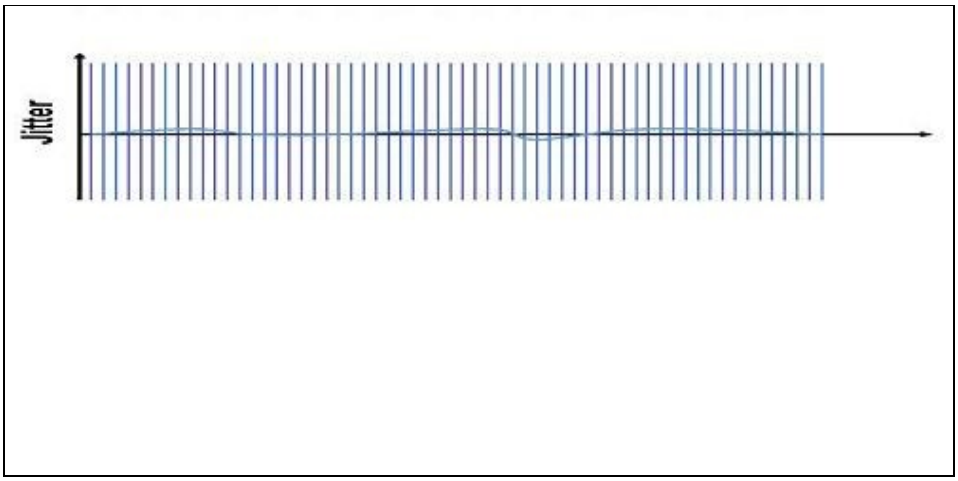
This behavior can be viewed graphically. The following illustration shows a network that has fluctuating delay over of period of time:



One could send a slow pace of test packets, and will capture somewhat high jitter (or delay variation) values:



If at the same time, someone else performs the same test, at a faster pace, the measured delay variation will be much smaller:



Conceptually, I could have no jitter if I manage to send packets fast enough. So what is the point of measuring jitter then?

The key is in the application. What is the jitter-sensitive application running on the network? Is it like a voice stream? If so, what is the interval between packets? The jitter experienced by an application is a function of the effective network delay variation and the interval between packets.

It unsound to measure with the wrong interval, as it is unsound to compare delay variation results performed with different measurement intervals.

As an example, for most VoIP codecs, the interval is 20 ms.

What if I Have Different Quality of Service (QoS)?

When there is more than one class of service, it is important to test them all separately, and with the same test. Indeed, to compare and quantify the difference of treatment between multiple classes, it is desirable to compare comparable things.

What kind of conclusion can you draw on two classes measured with different operations, and of course, having different parameters? Our recommendation in this case would be to set a test that covers the need of all the classes, and run the same one in all the classes.

One parameter that can be changed without affecting the comparability is the frequency. For instance, the default class could be measured once a hour, while the voice class is measured once a minute, and the

business class once per five minutes. The frequency changes, but the operation remains basically the same.

Most of the time, UDP Jitter Operation is the operation of choice for all of your classes.

Selecting Measurement Points

Depending the context, the measurement target, the platforms and so on, the measurement points might be placed differently. There is nothing like one-size-fits all for probe placement, but here are a few tricks.

Close to the end user

Generally speaking, what is measured should be as close as possible to the experience of the user. Therefore, probe placement should be very close the data provider and consumer. For a branch office, the router that is in the branch office itself is a good candidate to be a measurement point. A switch that connects IP Phones is another good example, very aligned with the reality.

For SLA, include only what you can influence

Hub and Spoke

This test method is very common, and gathers performance statistics between single points such as a POPs or data centers, to multiple remote locations such as branch offices. In such a case, it is strongly advised to have the sender running in the central location, and let the smaller routers on the edge do the responder job.

There are many reasons to have one sender and many responders:

- Senders have all the intelligence in a single point, including the operations configuration, the statistics, and that is only piece of equipment to manage.
- Easy to manage the number of operations executed at the same time, because the scheduling happens only at one point. With the muti-operation scheduler feature, this is becoming even easier. On the other hand, having many senders to one responder make the number of packet per second on the responder difficult if not impossible to predict.
- Should you want a new feature, or a bug fix in IP SLAs, all the intelligence is in the sender. We keep the responder as simple as possible, and there is no calculation being made, so there is barely any upgrade needed. It is easier to upgrade only one sender, than a lot of routers. If this is a dedicated router, it's not even in the switching path and you can upgrade/play at will without disturbing the production traffic. Definitely a big plus!

End to End

If the type of traffic you are monitoring is like peer-to-peer such as IP phone to IP phone, you will want to test close to the endpoints, like in an access router or switch. But that also means a lot of points, probably way to much to measure them all. One of the solutions is to combine a hub and spoke test topology (with a sender close to one call manager, or close to the internet exit point) with end-to-end tests.

The hub and spoke topology will cover all the endpoints, so you will get notified if something goes wrong in a particular location. On the top of that, some end-to-end operation can be executed between the most critical points.

Selecting the Most Important Measurement Points

If you cannot measure everything, and generally you cannot, you will have a make a decision and voluntarily cover only a subset of your network. One method is to use a traffic matrix.

A traffic matrix is a global view of the entry and exit points of your network, as well as the quantity and type of traffic in your network. With a traffic matrix, you can immediately identify the biggest consumers. For example, if you have a test coverage objective of twenty percent, tests for the top 20% talkers can be set up and this can represent in practice a very large portion of your total traffic.

Other method exists, and here again, there is no general guideline that can fit any requirement.

Performance

IP SLAs is an applications, it runs in IOS, and as such will need some CPU resources. While evey case is different, there are generic guidelines that can help you maintain your CPU usage as low as possible:

1) schedule the operations to minimize the amount of concurrent operations running at the same time. There is a feature called the *Multi-operation Scheduler* that helps you do that. By scheduling smoothly operations though an optimal period of time, not only you will avoid spikes, but you will maintain your CPU at the lowest possible value for the task.

2) Configured operations, even when they are not running, will incrase the CPU usage of the running operation. Try not to leave unnecessary operations configured, and remove them using the **no ip sla <op_id>** command.

3) Short intervals forces the router to schedule the IP SLAs process more often, raising the global CPU usage. Try to not go below a 20 ms interval, unless you really have to.

UDP Jitter Operation vs. RTP VoIP Operation

This paper covered only UDP Jitter, an operation that sends a stream of UDP test packets specially crafted for performance measurement. There is another operation called "RTP VoIP". There's fundamental differences between the two:

- RTP VoIP sends a real RTP stream, while UDP Jitter sends a simulated stream. Both looks the same from the outside, but RTP VoIP packets contains a real RTP stream.
- RTP VoIP uses and requires at least one hardware codec to run. It needs DSP resources and voice interfaces to run on.
- RTP VoIP will compute the MOS score based on the delay and number of DSP frames lost. UDP Jitter provides IP-based metrics, while RTP VoIP provides DSP-based metrics.
- RTP VoIP is an operation that is very specific for VoIP, and UDP Jitter is more generic.
- RTP VoIP is not designed to be deployed at large scale, while UDP Jitter is.

It is recommended to use UDP Jitter as much as possible, and it will be enough for the large majority of the cases. When more detailed information is needed, or for troubleshooting purposes, an RTP VoIP operation can be triggered on demand at specific time and location. RTP VoIP is not meant to replace UDP Jitter.