IQM:FAQ.

Last edit 2011-01-11

Contents:

What is IQM?

IP Quality Monitor (IQM) — is a hardware+software appliance, which provides measurement and management of IP quality parameters thrughout network. Network-supported Classes of Services and Zone structure are taken into account during measurements. IQM system supports distributed monitoring of IP network quality parameters. Actually, it is the most used mode of IQM deployment.

What can I use IQM for?

IQM could be used for:

- SLA quality parameters measurement and management
- Active measurement of IP networks quality parameters
- Network services troubleshooting and bottleneck detection.

What is measured?

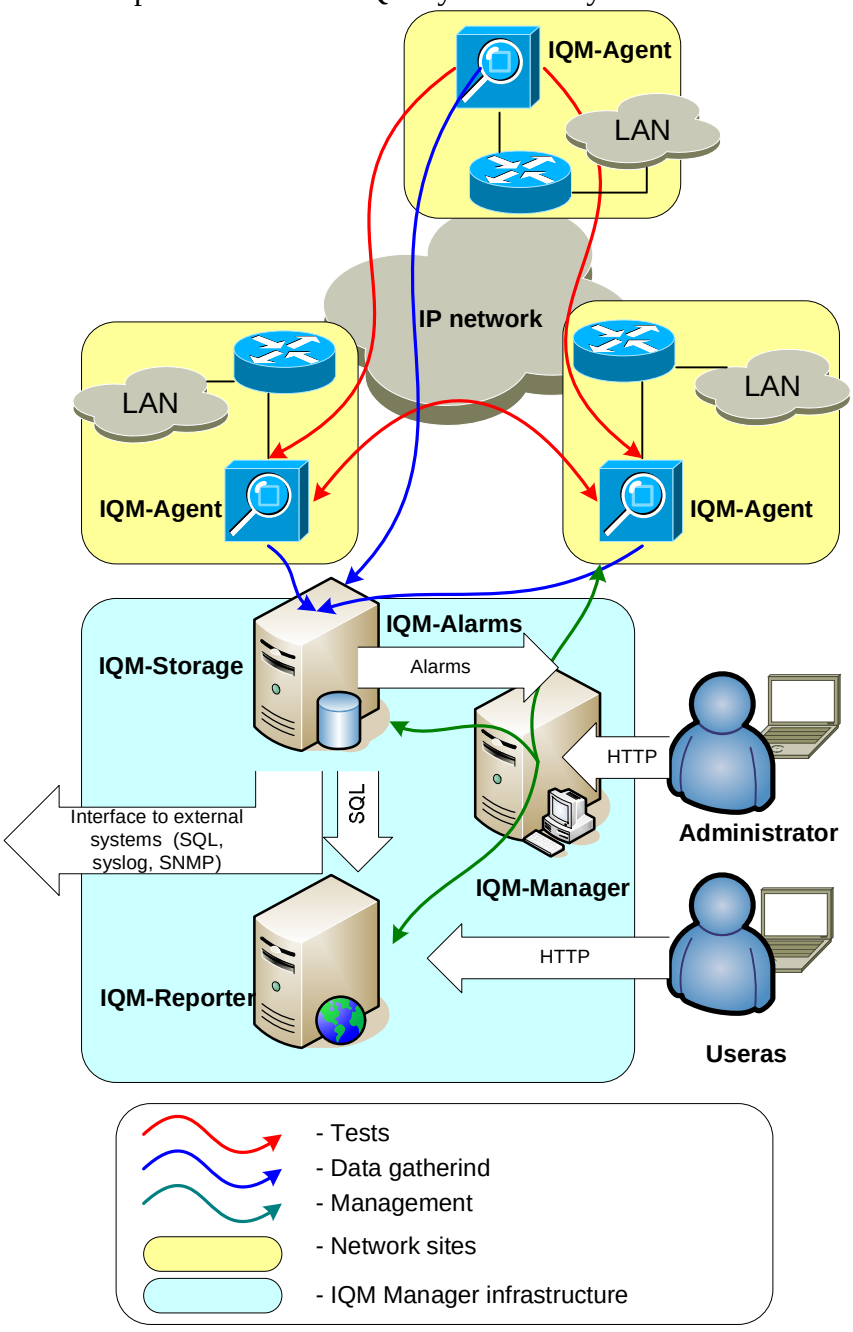
IQM measures:

- Packet loss.

– Delay (RTT).
– Delay variation (Jitter).
– Available bandwidth per class of service and point-to-point connection.
– Service availability via controllable quality parameters

IQM: the functional composition

Functional composition of the IP Qiality Monitor system could be seen here.



There are two main elements: IQMA (Agents) and IQMM (Managers).

Agents: There are special network appliances used for quality measuremets, which are located in the main network nodes. They are small computers running IQMA software, and are called *probes* or *agents*. Agents automatically (on demand or on schedule) run tests between them (agents) and measure delivery parameters losses, delay (RTT), variation (jitter) and bandwidth. Gathered data is than pre-processed, stored in text files and delivered to higher level –

Manager. IQM Manager is the brain of IQM system, providing agent management, data processing and analysis, user interface. IQMM provides:

- IQMA agents management:
  - Policy configuration
  - Threshold management
  - Test by schedule configuration
  - Test by demand configuration
- IQMM (lower level managers) management for distributed systems
- Automated processing of acquired data::
  - Export of data to SQL DB
  - Data reduction and pre-processing for storage
  - Data storage
  - Data analysis and consistency checks
- Data violation signalization
- Stats graphing and user interface

IQM Principles of operations: Simple introduction

IQM Agent runs under Linux or another Unix variant. Bundled versions use our own Linux distribution optimized for low delay. Major part of software is written is C++, so it is fast and compact.

SSH (TCP 22) and NTP (TCP/UDP-123) should be configured and running on the agent and configured on the agent's and LAN firewall (if any). Precise time synchronization is vital for network measurements, so selecting the best available NTP servers is major task.

Inter-agent tests are precisely managed data packets series interchanged between them. Based on the measured timings, major network parameters, such as Packet Loss, PDV (Jitter), RTT (Delay) and Bandwidth are calculated. Some type of measurement require special types of payload and/or packet management. All tests now use UDP. UDP ports in use could be dynamically allocated or statically configured by administrator (for firewalls, etc). Data acquisition stores results to text files. Using pre-configured criteria (lines number, time period etc) CDR files are rotated and delivered to IQMM using FTP or SSH (SCP).

IQMM runs data checks and simplifications, loads data into SQL DB, prepares reports and runs user interface. Alarms are generated when violation occurs.

IQMA control protocol (proprietary) uses TCP port 1189 and features handshake, authorization and so on. Agent password is never transmitted in clear.

Agent IQMA hardware requirements

Hardware for IQMA could be almost any x86- or ARM- compatible platform. IQMA has been ported to NSG-router and SheevaPlug.

General Probe (Agent) hardware requirements

Up to 100 megabits per second:
- Atom 230 and faster,
- 1GB RAM and more,
- USB-Flash (512MB – minimum, 1GB - recommended) , computer MUST support USB flash boot.
- FastEthernet

Up to 200megabits per second:
- Atom 330 and faster,
- 2GB RAM and more,
- USB-Flash (512MB – minimum, 1GB - recommended) , computer MUST support USB flash boot.
- Gigabit Ethernet

Up to 1 Gigabit per second:
- Intel Core2Duo 2GHz and faster
- 2MB cache and more,
- 3GB RAM and more,
- USB-Flash (Fast 1GB minimum, 16-32GB recommended due to speed factor) computer MUST support USB flash boot
1-2 Gigabit Ethernet.

All hardware used should be 100% Linux compatible with 2.6 kernels and higher.

Manager IQMM hardware requirements

IQMM hardware requirements are a functions of test load, system topology, domains number (if distributed), detalisation, user requirements and so on. So it is rather hard to guess required system parameters based on number of probes alone. So, following guesstimates should be considered only as starting point:

Up to 10 agents:

CPU Intel Core2Duo 1.8 GHz and more >=4GB RAM, 300 GB 10Krpm SATA HDD, Fast Ethernet

Up to 30 agents:
Intel Core2Duo 3GHz (e.g. E8400) and more; >=6GB RAM, 2x300 GB 10Krpm SATA HDD, Gigabit Ethernet

Up to 50 agents:
Intel Dual Core i5/Xeon 3GHz 4MB cache (e.g.-i5-650,i5-750)and more ; >=8GB RAM, 3x300 GB 15Krpm SAS or SATA RAID, 2xGigabit Ethernet

Up to 100 agents:
Intel Quad Xeon/i7 (e.g.-Q9550,i7-950) 2.8 GHz/8MB cache and more; >= 16GB RAM, RAID 5,6,10 2TB and more, 2xGigabit Ethernet

Up to 200 agents:
2xIntel Quad/Hexa Xeon (e.g.-X5470,W3570,W5580) 3GHz, 8MB cache and more; >=32GB RAM, RAID 5,6,10 4TB and more, 2xGigabit Ethernet

Keep in mind, RAM is crucial for report generation. If you want fast report generation, increase RAM.

All hardware used should be 100% Linux compatible with 2.6.30 kernels and higher.

What are preconfigured IQM options?

Usually, you can buy IQMA either pre-packaged (Small Form-Factor Micro-ITX PCs featuring Atom Dual-Core board, 2GB RAM, GigaBit Ethernet) or USB Flash Image for use with your own hardware (e.g. obsolete office Pentiums-III or Celerons).

However, there are some other options for low-speed specialized tasks:

Software IQMA module for NSG series 800 router. Provides IQMA functions for popular NSG-800 and NSG-900 routers widely used for POS and ATM terminals. Should be installed by Net-Probe only.

IQMA for ARM. We now support Sheeva Plug and are working on another implementations.

What kinds of tests are available in IQMA?

Currently the following tests are available:

- Wire-speed stress-tests for switched networks.
- Bandwidth test for routed networks.
- Auto speed determination (for fixed loss percentage).
- Basic quality tests: jitter, loss, delay

- Audio/Video streams simulations
- Emulation of different applications/services using CoS
- UDP-echo tests for nodes without IQMA installed.

How is test start time determined?

There are several mechanisms to start test timely.

- **Periodical**: test starts every time period specified elapses since test creation or agent reboot.
- **Template-based**: uses Cron-like strategy.
- **On demand**: Run once or number of times specified by hand. For manual usage.

Periodical tests are most suitable for regular measurements. Since they use limited bandwidth, they do not affect running applications.

Template-based tests are used for regular measurements using stress-tests or other bandwidth-intensive tests affecting payload. You can specify night hours and weekend days in template to avoid harm to payload. Also, template-based tests could make simultaneous start of test group, which is used in diffserv/QoS tests.

On-demand tests are used by operators for irregular and spontaneous troubleshooting, etc. There is no storable history for these tests.

Do tests affect each other?

IQM implements several techniques to overcome mutual influence of running tests and provide concurrent stress tests when they are needed.

Common practice is to lessen or completely eliminate the mutual influence of tests. Mutual influence could be caused by:

1. Sharing hardware resources of probe.
2. Sharing limited network resources.

So, mutual interference affects performance during lack of at least one of the shared resources. When limiting factor is network bandwidth and/or network elements load, measurement errors are affected either by payload influence (in this case we would see the real network behavior) or by stress load, generated by agents. Stress load generation also loads hardware resources both of the agent and network elements. To lessen the burden during periodical stress tests, pseudo-random time-shift interval of 1 to 59 seconds is used. When on-demand tests should be run, systems has ability to temporary suspend periodical and template-based tests in the specified time interval and wait for completion of the tests already running.

For non-stress tests, bandwidth allocation is specified to avoid bandwidth hog.

If concurrent tests are needed, templates should be used to configure them.

What bandwidth does test consume?

Bandwidth limit is specified for all non-stress tests. Default limit is 64 kbits/s, and should not be lessen without considerations, since it could adversely affect measurements quality.

Stress-test are only limited by hardware and network resources. You can see `Agent IQMA hardware requirements` for more information.

How to calculate the proper number of tests for measurement?

Given number of agents, how much tests should we create in order to reliably measure the performance of network in all needed directions?

If you need to run tests in full-mesh mode, then total number of tests should be equal to number of links between nodes:

<Number of tests>=N(N-1)/2
Where N is the number of nodes.

Many networks have explicit or implicit traffic concentration points, which are much less in number than total number of nodes. So it is worth to avoid running tests between nodes, which have small or no traffic between them. For example, in star topology, only the central node should have “active” agent, where terminal nodes could be considered “inactive” and have agents configured to run in “coupled” (responder-only) mode.

In this case, number of tests for full-mesh network should be corrected by subtracting number of “holes” (missing direct inter-nodal links) defined by “inactive” network elements:

<Number of tests>=N(N-1)/2-I(I-1)/2=(N-I)(N+I+1)/2

Where N is the total number of nodes, I – number of “inactive” nodes.
If we use A=N-I as a number of “active” nodes (traffic concentration points), we have:

<Number of test>=A(2N-A-1)/2
In extreme case, when network has explicit star topology with single center, we have:

<Number of tests> = N-1

Result should be multiplied by number of managed classes of service C. i.e. universal formula for number of tests for the network with N total nodes, A active (concentration) nodes and C classes of service should look like

<Number of tests> = A*C*(2*N-A-1)/2

What is maximum depth of history?

The size of stored history is limited only by the available disk space volume. Data acquired in one test, after two aggregations, consumes approximately 150 bytes. Given test interval t minutes and T days total run time, using formula from previous answer:

<Data volume> = 150 * (T*24*60/t) * (A*C*(2*N-A-1)/2) = 108000*T*A*C*(2*N-A-1)/t

Given free disk space V bytes, we have:

<Maximum storable history period (days)> = T = V*t/(108000*A*C*(2*N-A-1))

Given V=100GB disk space, N=100 nodes, A=1 active traffic concentration point, C=3 classes of service, t=1 minute tests interval, we have 4.5 years of history. If we have A=10 then history size drops to 6 months, if we use full-mesh then 33 days.

How is Round-Trip Time (RTT) measured?

One IQM agent is used during U7-type (UDP-echo) test, responder should run UDP-echo service. IQM agent encapsulates test request timestamp in the UDP datagram date field. Time is gathered from the local clock of the IQM agent. Request is forwarded to the measured host running UDP-echo service. According to RFC 862 (Echo Protocol), echo service should reply data without changes. So, the IQM agent gets information on the request origination time from datagram replied. RTT is calculated as a time difference between request origination timestamp and time of reply received. Since both times are gathered from the IQM agent local clock, there is no additional time error.

Two IQM agents are used during U0-type tests. Agents send test request series to their respective counterpart. One-way trip times from source to destination (SD) and destination to source (DS) are measured. Test UDP datagram contains encapsulated timestamp gathered from originator's local clock. Receiving side gets it's own local time. One-way trip time is calculated as a time difference between request origination timestamp (using originator's local clock) and local time of receiver at the test request delivery. RTT is calculated as the sum of the two one-way trip times. Local clocks of the agents should be synchronized.

How is delay variation (jitter) measured?

Delay variation (Jitter) is calculated during tests according to RFC 3550 method:

$$J_i = J_{i-1} + (|D_{i-1,i}| - J_{i-1})/16$$

Where

- J_i – measured delay variation at i-th iteration.
- $D_{i-1,i}$ – difference of receiving times of two consecutive test requests.

$$D_{i-1,i} = (R_{i-1} - S_{i-1}) - (R_i - S_i)$$

R – test request generation time, S – test request receive time.

e-mail: info@net-probe.ru, web: http://www.net-probe.net

During U7-type (UDP-echo) test the variation of round-trip time is measured, using the local time of originating IQM agent.

During U0-type test the variations of one-way trip delays are measured in both directions. Originating agent's local time is used for test timestamp, receiving agent's local time is used for packet delivery time.

How does agents clock synchronization affects measurements?

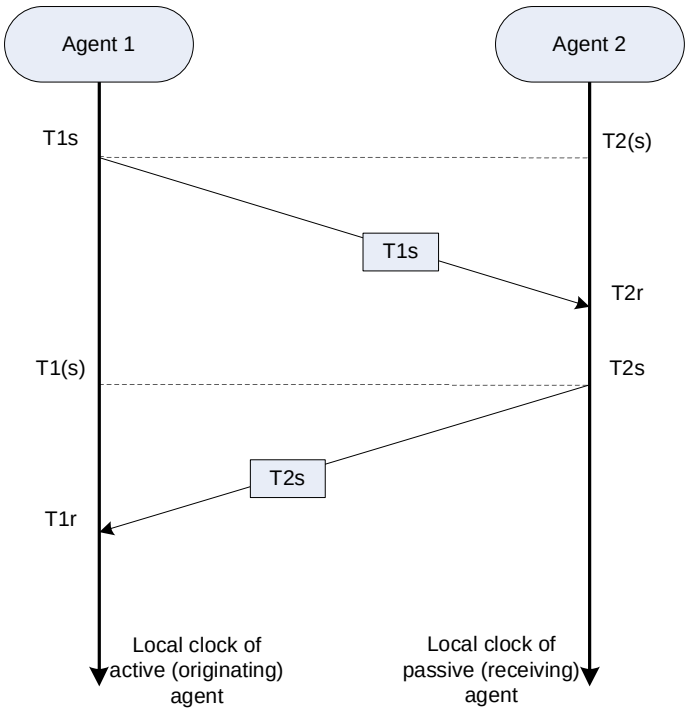
During U7-type (UDP-echo) tests only originating agent's local clock is used, and only RTT parameters are measured. Clock state of the bound passive agent is irrelevant for the measurement precision.

Now, let's see, how is round-trip time calculated for tests involving one-way requests between two IQM agents. (RTT measurement details could be seen in the "How is Round-Trip Time (RTT) measured?" article).

Let us designate local time difference on the agents as dT.

- dT = T2 – T1 (T1, T2 – local times at the corresponding active and passive agents)
- T1s – test request generation time by the active (originating) agent according to its local clock
- T2(s) - test request generation time by the active agent according to passive (receiving) agent's local clock
- T2r – test request delivery time at the passive agent according to its local clock
- T2s – back-flow test request origination time at the passive agent according to its local clock
- T2(s) – back-flow test request origination time at the passive agent according to the active agent local clock
- SDD – Source to Destination delay, **real** test request delay between active and passive agents
- DSD – Destination to Source delay, **real** back-flow test request delay between passive and active agents
- sdd, dsd – **calculated** values of one-way delays

Following diagram shows the process of one-way delays measurements, which are used for round-trip delay (round-trip time) calculations:



Request delivery times T2r and T1r (local clock-based) could be expressed as:

$T2r = T2(s) + SDD = T1s + dT + SDD$ (1)

$T1r = T1(s) + DSD = T2s – dT + DSD$ (2)

As illustrated in the «How is Round-Trip Time (RTT) measured?» article, SDD is calculated as :

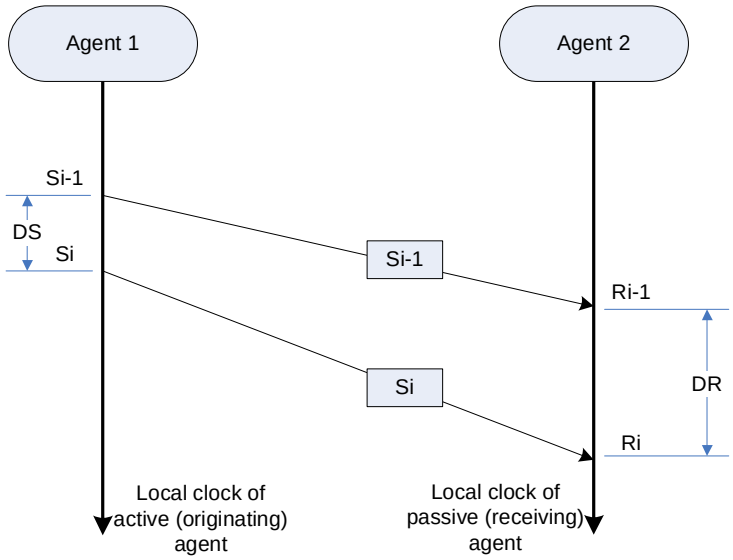
$sdd = T2r – T1s = (using\ (1)) = T1s + dT + SDD - T1s = SDD + dT$ (3)

$dsd = T1r – T2s = (using\ (2)) = T2s – dT + DSD – T2s = DSD – dT$ (4)

$RTT = sdd + dsd = (using\ (3,4)) = SDD + DSD$

As we can see, agents' local clocks offset does not affect the precision of RTT calculation. However, additional condition should be met: dsd>0 and sdd>0, thus for correct calculations, local clocks offset should be no more than one-way delay.

Let's see, how is delay variation (as described in the article "How is delay variation (jitter) measured?") calculated:



Di-1,i parameter is difference of delays of two consecutive requests:

Di-1,i = (Ri-1 – Si-1) - (Ri - Si) = (Si – Si-1) – (Ri – Ri-1) = DS – DR

Since only relative time differences are used in the calculations, local clock's offset does not affect the precision.

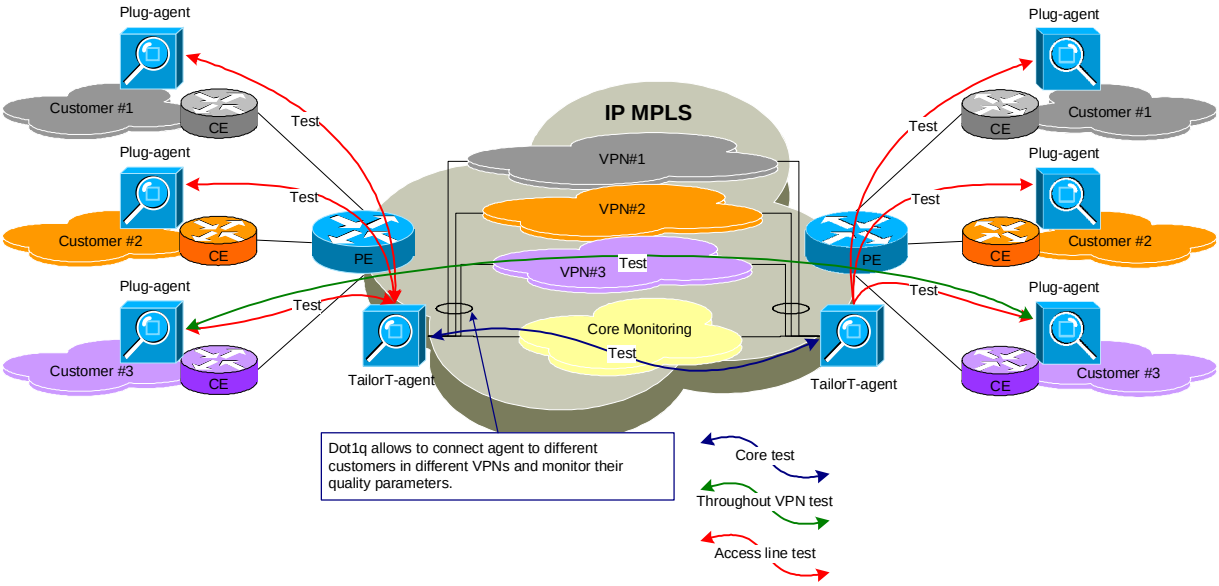
Condition of positive values of calculated parameters dsd > 0 and sdd > 0 should be met, thus local clock offset of agents should be not more than one-way delay for correct calculations.

Is it possible to use IQM in the networks with address space collision?

IQM could be used for quality measurements in the networks with address space collision (intersection, coincidence). This is often the case for IP VPN providers adding SLA quality monitoring. There are two possibilities to handle the task:

1. Deployment of NAT (Network address translation) for external representation of VPN address space. MPLS network could use VRF-aware NAT technology. So, IQM agent deployed in the customer's network core could be represented with unique global (external) IP address in the management network.
2. Administration of customer's address spaces, giving unique addresses to IQM agents and providing IP interconnection between agents and management system.

   One or several IQM agents are deployed on the provider's edge. They should be connected with customer' VPNs through physical or VLAN interfaces. Details could be seen here:



e-mail: info@net-probe.ru, web: http://www.net-probe.net