

Low-Cost Precise QoS Measurement Tool

Sven Ubik, Vladimír Smotlacha (Cesnet)

Sampo Saaristo (Tampere University of Technology)

Juha Laine (Soon Communications)

26.6.2001

1 Abstract

When implementing networks with QoS guarantees, precise measurement of network QoS characteristics is needed. Primary characteristics of this class are packet loss, throughput, delay, delay variation as well as distribution of delay and delay variation.

Commercial network analyzers are often very expensive, do not include provisions for precise time synchronization needed for one-way delay measurement and are closed in a sense that it is not possible to integrate them with other applications for measurement result processing.

In this article we describe an architecture and implementation of a simple low-cost measurement tool that can be used for precise measurements of all of the above listed characteristics in small laboratories.

2 Introduction

When implementing networks with QoS guarantees, precise measurement of network QoS characteristics is needed. We need to observe behaviour of various traffic conditioning and traffic engineering mechanisms on different patterns of data. This behaviour consists of two parts which can influence each other – general phenomena occurring in processing of packetized data and implementation-specific phenomena occurring as a result of a specific implementation of scheduling mechanisms.

An example of general phenomena is behaviour of a traffic microflow when certain specific handling is applied to the whole traffic aggregate, to which the microflow belongs [tiziana-microflow]. This is a common situation in now increasingly popular diffserv approach where specific traffic handling is applied to whole aggregates, rather than to individual microflows, but we still need to

know the influence on microflow behaviour.

An example of an implementation-specific phenomena is traffic shaping precision, that is how much we can rely on the shape of the traffic being changed as specified and what shaping granularity is available in particular implementation.

Measuring of QoS characteristics is also needed for SLS [sls] (Service Level Specification) auditing, that is the verification that both the user and network behave in compliance with the specified traffic contract between the user and network.

An important practical issue related to specific traffic handling for QoS implementation purposes is the observation of additional load imposed on routers by application of such mechanisms. We found that certain mechanisms significantly increase the load of routers [tiziana-microflow][ubik-traffic-shaping].

For the purposes mentioned above we need a tool that can generate a set of data streams of a specified shape and another tool that can measure all primary QoS characteristics with sufficient precision and time granularity. These characteristics include packet loss, throughput, delay, delay variation as well as distribution of delay and delay variation. The required precision and time granularity depends on the purpose of measurement. For network QoS measurement for evaluation of influence on the quality of audio/video services, precision at the order of 1 to 10 ms seems to be acceptable [victor]. However, when we want to verify behaviour of a certain scheduling mechanism, we need to observe dispatch times of individual packets in which case the precision at the order of 1 to 10 microseconds is needed. This is the time required to send one minimum-length packet on Fast Ethernet or Gigabit Ethernet networks (which is about the same, Gigabit Ethernet runs at 10 times higher speed, but its minimum-length frame is 512 bytes long). When auditing an SLS, the time granularity in which we need to measure the QoS characteristics is given by the granularity of the SLS itself.

3 Related Work

Commercial network analyzers can provide very high precision, but are often very expensive, out of budget of many small laboratories. In many cases they do not include provisions for precise time synchronization needed for one-way delay measurement. They either provide sending and receiving ports on the same device, thereby avoiding the need for time synchronization, but prohibiting to make measurements between two separate physical locations or they require time synchronization to be provided externally. It is also sometimes difficult to integrate these analyzers with other custom-tailored applications for further processing of measured data.

A lot of free network performance measurement tools are also available. Some of them are specialized in certain aspects of TCP intricacies [rfc2398]. However, most of them can only measure throughput or delay. Usually only average throughput over certain period of time can be measured (e.g., netperf [netperf]). Detailed insight into instantaneous throughput measured with selected time granularity is not possible. Delay is usually measured as round-trip (e.g., ping) or external time synchronization is required. If the application also includes a traffic generator, often only a “full-speed-ahead” stream can be generated. It is not possible to generate a stream with specified rate and shape. The most commonly used means for time synchronization in the Internet is the NTP protocol [ntp]. Precision of time synchronization which can be achieved using NTP is generally about 5 ms [ntp-precision], but depends on external conditions, such as network operational characteristics (which cannot be fully compensated for by the NTP algorithm) and the load of the machine executing the NTP daemon (which interferes with measurements, which use NTP, but can in turn significantly increase the machine load).

Very precise atomic clock provide extreme accuracy, but they are also very expensive. Currently the overall best solution in terms of precision and cost seems to be using GPS receivers. However, there are certain issues which must be considered in practical use of GPS receivers for time synchronization for QoS measurement purposes.

RIPE NCC started in 1997 the Test Traffic project [ripe-box], which uses computers located in different networks, each with its own GPS receiver. The goal of this project is different from ours, it is designed for long-term measurements of one-way delay and data loss (as a metrics of network connectivity) between several tens of ISPs.

In the rest of this article, we describe architecture and implementation of a simple low-cost tool for precise measurement of all primary network QoS parameters. The tool uses GPS receivers and is based on the RUDE/CRUDE packet generator and collector. We also describe our practical experience with using this tool.

4 Architecture

We set the following requirements on the architecture of the measurement tool:

- Simultaneous generation of multiple traffic streams with specified rate and shape
- Measurement of packet loss, throughput, delay, delay variation and distribution of delay and delay variation

- Characteristics calculated with specified time granularity
- Precision at the order of 10 microseconds
- Open to development of applications for further processing
- Low cost

The architecture is depicted in Fig. 1. The principal part is the packet generator called RUDE and a companion packet collector called CRUDE [rude], running on two Linux machines. Both programs were developed at the Tampere University of Technology as part of the Faster 2000 project. RUDE generates a set of traffic streams according to the specification in its configuration file. CRUDE receives generated packets and stores a short snapshot about each packet in a binary file. This file is then converted to a text file, which is then supplied to *gdplot*, a utility that computes required QoS characteristics and depicts them in a graphic notation. Both PCs, with RUDE and CRUDE, have precisely synchronized time for one-way delay measurement. This is achieved by sending precise seconds pulses from a GPS receiver to a serial port of each of the PCs. If both PCs are at the same location, one GPS receiver with a distribution unit can be used. Otherwise each PC must have its own GPS receiver. We describe individual parts in more detail below.

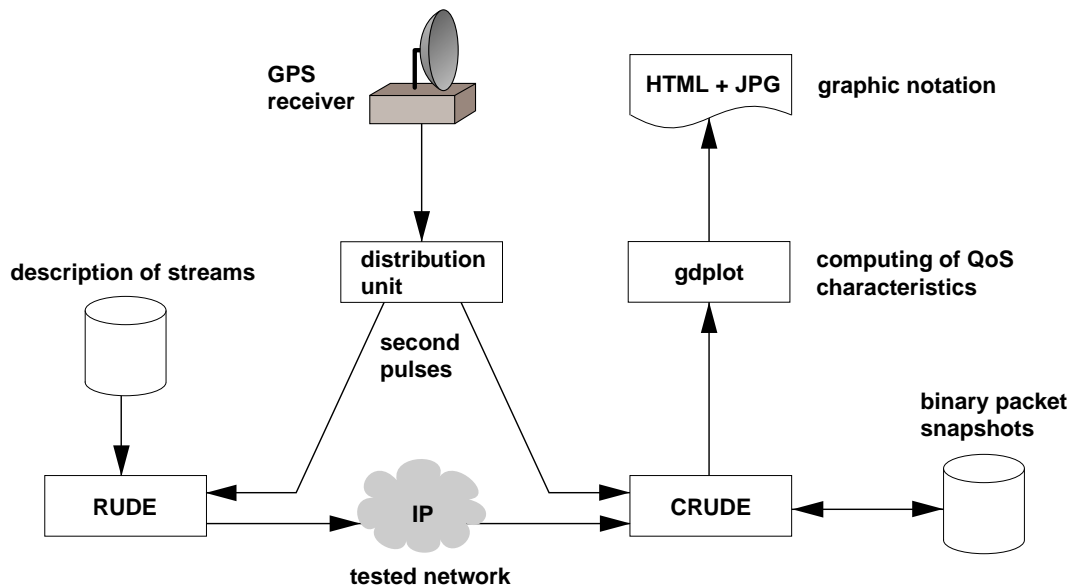


Figure 1: System architecture

4.1 RUDE/CRUDE

RUDE/CRUDE [rude] utilities are similar in operation to mgen/drec [mgen] utilities, but are much more precise and can be used at higher speeds. RUDE stands for Real-Time UDP Data Emitter. Currently two types of traffic streams can be generated. The CONSTANT stream consists of a steady flow of UDP packets of a specified size sent out at a specified rate. Both the size of packets and the rate can be programmed to be changed at any time during the flow generation. The TRACE stream consists of UDP packets with packet sizes and inter-packet gaps specified separately for each packet in a trace file. RUDE can generate multiple CONSTANT or TRACE streams simultaneously. A command for setting the TOS byte in the IP header is also available.

We developed a simple utility which can gather information about packet sizes and their timestamps from the real network via interfacing to tcpdump [tcpdump] packet sniffer. It then creates a trace file for RUDE so that a “production-like” traffic stream can be generated in a laboratory.

CRUDE is a receiver and logging utility for packets generated by RUDE. It generates a snapshot about each received packet. The snapshot includes a stream identifier specified in RUDE configuration, sequence number of the packet within the stream, transmitting timestamp, receiving timestamp and the packet size in bytes. The snapshots can be either displayed on-the-fly in a text form on the standard output or logged in a binary form to the file to be decoded into the text form later. The latter option reduces I/O operations during measurement allowing for packet processing at higher speeds. We describe some performance characteristics later in section 4. The timestamps indicating when each packet has been sent and received are stored with 1 microsecond resolution.

In contrast to mgen, RUDE can dispatch packets with steady precision of 1 microsecond (provided that the PC’s clock is precisely adjusted). It has also lower and stable overhead resulting in low additional delay (introduced by RUDE/CRUDE into measurement results), which can be easily compensated.

4.2 GPS time synchronization

Synchronization of computer clocks in internetwork environment has been for more than two decades based on the Network Time Protocol (NTP) [ntp]. NTP controls the computer clock time and frequency using an adaptive feedback loop - originally PLL, in the latest version hybrid combination of PLL/FLL (Phase/Frequency Lock Loop) [pllfill]. Although the clock can be reliably synchronized in this way to a few tens of milliseconds in the global Internet and slightly better than a millisecond in LAN, it is not enough for precise delay and jitter measurement.

The NTP program package implements not only the NTP protocol and clock control, but also drivers for various externally connected clock sources including precise PPS (pulse per second) [stone].

Currently, one of the cheapest and easily available high-precision external clock sources is a GPS receiver. GPS is primarily designed for obtaining a global position by receiving signal from a set of satellites. However, it also provides a precise global time because the position calculation is based on the synchronization of an internal clock with the atomic clock in a satellite.

Accuracy of PPS depend on the receiver model. We used a low-cost Garmin GPS 35-LVS which should provide accuracy of 1 microsecond. More expensive receivers with the accuracy of tens of nanoseconds are also available (e.g., Motorola UT Plus Oncore provides accuracy of 50 ns).

Another aspect to be considered is the resolution of a system clock in a PC. The system clock design has not been changed since the early AT/XT. The base is a quartz oscillator 14.31818 MHz, whose output is divided by 12 to obtain 1.193182 MHz clock. This frequency enters a 16-bit programmable counter (originally the Intel 8254 chip). This counter is readable, but even when we use the current state of the counter, the resolution is in any case limited to 0.838 microsecond (1/1.193182 MHz).

To improve the kernel clock function, nanokernel [nanokernel] has been designed and implemented. Currently it is available as a patch for standard Linux and FreeBSD operating systems. The goal is to offer a function for reading time with the resolutions up to a nanosecond. The principle is to use the Pentium TSC (Time Stamp Counter) register, which counts CPU ticks, that is CPU frequency. Therefore, the resolution is dependent on the CPU clock and it is really at the order of 1 nanosecond with current Pentium processors.

Time-dependent measurements across the network require time synchronization between all involved computers or other equipment. The easiest way used mainly in global-scale tests is an independent GPS receiver for each computer — this solution is deployed in the Test Traffic Project (RIPE) [ripe-box]. For LANs, a more economical solution is to use a common GPS receiver as a reference clock source. This clock source can be distributed over the LAN using special Ethernet adapters [schmid]. We chose a simpler way — to directly distribute the PPS signal from the source to computers via standard category 5 cabling.

We used Garmin GPS 35-LVS receiver, which provides NMEA output in RS-232 (standard serial line level) and PPS in TTL level. As PPS signal should be connected to the DCD pin of the serial interface (originally Data Carrier Detect signal from modem), it requires a voltage level converter from TTL to RS-232. The NTP package version 4.0.99k, including the nanokernel patch has been installed on several Pentium PCs with CPU frequencies from 133 to 860 MHz

and kernel version 2.2.17 or 2.2.18. We constructed our own distribution unit with TTL to RS232 converter, which allows to distribute the signal from a GPS receiver to several (currently 4) destinations.

4.2.1 Clock precision discussion

Based on our analysis of possible causes that influence the computer clock precision, we divide them into the following factors:

- Base accuracy of PPS from GPS receiver - 1 microsecond
- Delay in voltage level convertor - order of 100 ns
- Delay in cables - every 300 m means 1 microsecond at light speed
- Delay in the computer caused by interrupt handling
- Short time fluctuations of quartz frequency (frequency offset of quartz crystal and its long time instability is compensated by PLL/FLL) - not worse than 100 ns
- Stability of PLL and FLL loopbacks - estimated under 5 microseconds [nanokernel]

For the purposes of one-way delay measurement, we need the clocks in PCs on both sides to be synchronized, rather than being accurately set to the global time. When both PCs use the same source of PPS, we do not need to care about factor 1. Factors 2 and 3 are small, constant and are eliminated when both PCs get PPS signal over the same path. Factor 5 is also small. Therefore, the most important issue is the variation of the remaining factors 4 and 6. Factor 4 can be measured by software echo of PPS signal (PPS is echoed to another serial line output by the interrupt routine after its processing). We measured offset between PPS and its echo by a two-channel oscilloscope with the following results:

Pentium 150 MHz	11 - 20 microseconds
PentiumIII 860 MHz	7 - 12 microseconds

Table 1: Offset between PPS and its echo

We found that when CPU was lightly loaded, the delay was nearly constant and close to the lower end of the indicated range. When CPU was under heavy load (kernel compilation) the delay reached the higher end of the range. Values in the middle of the range were rather rare. In other words, the delay distribution

tended to have two maximums. There are two important points. First, when PPS comes when the interrupt is disabled, it is significantly delayed and is therefore usually excluded from the following processing as suspiciously out-of-range value. Second, although the fluctuation of a pulse processing time seems to be relatively high, its influence on the computer clock precision is lower. PPS do not control the clock directly, only average offset from the current time calculated from the last 16 pulses controls PLL. As the measurement was made by visual reading from the screen, we cannot present the data in a more exact way. But we can conclude that after compensations discussed above and after deduction of the constant part of factor 4, the computer clocks can be synchronized with the precision of 10 microseconds in the worst case.

In one-way delay measurement, an additional delay is introduced by processing packets in TCP/IP stacks and in the measurement application. This delay is constant and can be deducted from the measurement results.

4.3 Computing and displaying of results

Computing of QoS characteristics and displaying of results in a graphic notation is done by a common utility called `gdplot`, which we developed as a part of the project. The name is derived from using the GD graphics library [`gdlib`]. Although both functions are included in one application, there is an internal division into two parts with interface defined by data structures.

The first part computes QoS characteristics. Its functionality is illustrated on a diagram in Fig. 2. It first reads packet snapshots received from CRUDE one at a time by the `getNextPacket()` function. It sorts snapshots into traffic streams and determines the measurement period to which the packet belongs. A measurement period is a fraction of time, inversely proportional to the time granularity, over which throughput and other QoS characteristics are computed. For example, if we perform certain test for 10 seconds, we may set the measurement period to 100 ms to obtain 100 steps in which throughput or other characteristics will be computed and plotted as they change over the test duration. The measurement period can be specified arbitrarily with 1 microsecond resolution. Some other applications can also compute changes of QoS characteristics over the time, but support only decimal multiples of allowed measurement periods, for example, 10 ms, 100 ms, etc. Support for an arbitrary measurement period is important, for example, for traffic shaping tests, because the shaping period is often set to non-decimal multiple of a time unit (e.g., 12 ms) as a result of shaping implementation constraints in a router. `gdplot` can read more files produced by CRUDE at one time, depicting characteristics for several selected streams in one graph. Several other parameters can be specified on a command line including start and end time of calculation of characteristics as well as the measurement period.

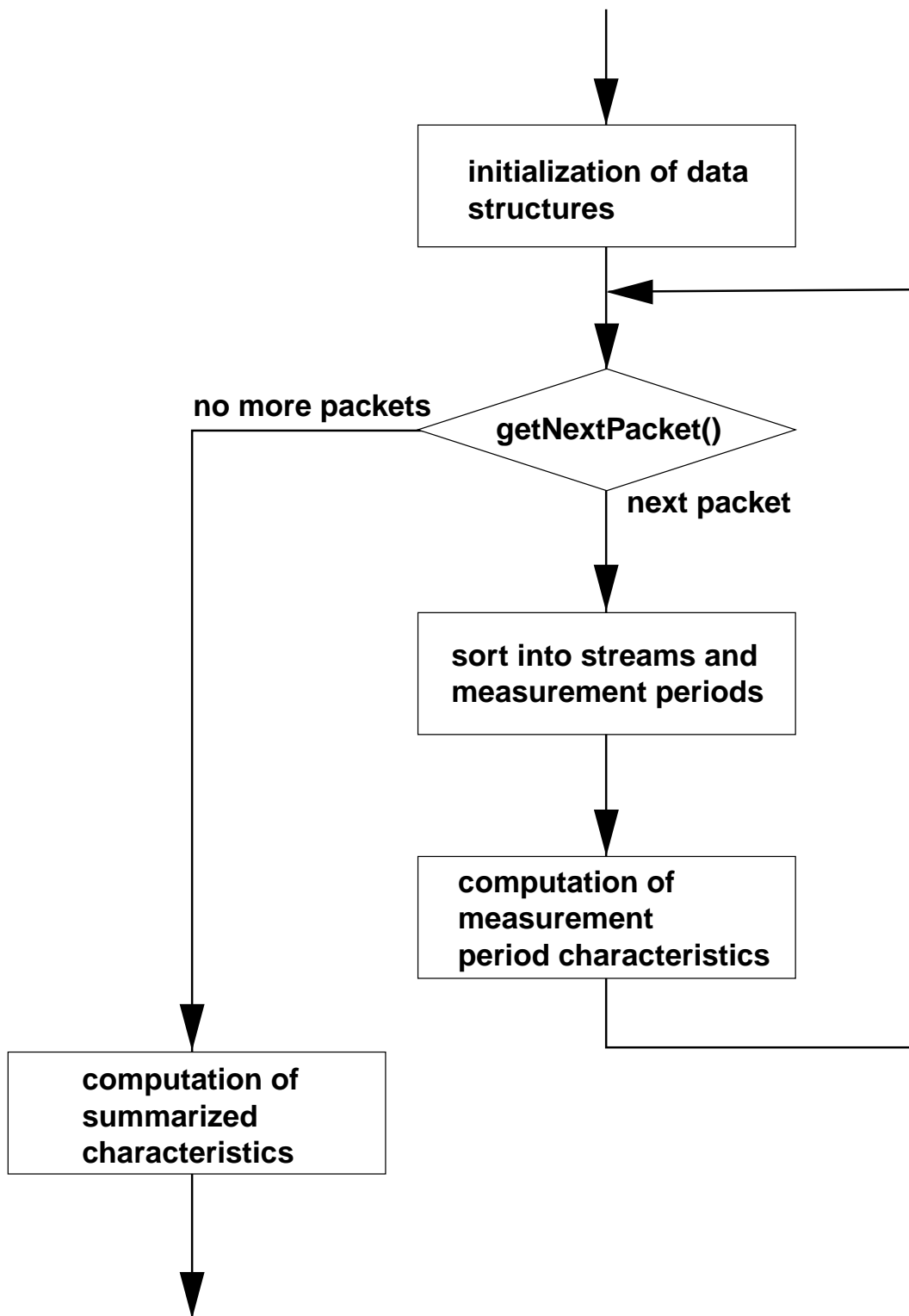


Figure 2: qosplot control flow

Other applications for further processing of measurement results can be integrated at two places. They can either read the snapshots produced by CRUDE or they can use the pre-processed data in structures produced by gdplot.

5 Practical experience

To check the precision and stability of time synchronization between two machines, we set up the configuration according to Fig. 1 and connected the two PCs directly over a Gigabit Ethernet connection. With minimum-length frames (512 bytes + 8 bytes preamble), we measured one-way delay of 22-26 microseconds over the period of 1 hour. The constant part of this delay consists of two parts: the time needed to send the frame at the connection speed (about 4 microseconds) and overhead of processing in RUDE/CRUDE, TCP/IP stacks and network adapters (by running both RUDE and CRUDE on the same machine, we measured overhead 7 microseconds, but it still does not include the whole TCP/IP stack and network adapters). The delay variation is caused by factors 4 and 6 discussed in section 3.2. We can conclude that in our implementation of the proposed method on lightly loaded machines we can measure one-way delay with the precision of 2 microseconds.

We used our system for measurements performed as a part of the QoS in IP project [qosinip] done in CESNET. Several example graphs obtained from gdplot as a result of the traffic shaping precision test done on the Fast Ethernet interface of the Cisco 7500 router are shown in Fig. 3 to Fig. 5. The input stream, which was being shaped, had a constant throughput of 5 Mb/s, generated as a sequence of bursts with the steady rate of 20 Mb/s for 100 ms followed by gaps with no packets sent for 300 ms. This stream was generated twice with packets 64 bytes and 1500 bytes long. Traffic shaping was configured with CIR=5 Mb/s (Committed Information Rate), Bc=500 kb (Committed Burst size) and Be=0 (Exceed Burst size). The test lasted for 10 seconds and the measurement period was set to equal the shaping interval Bc/CIR=100 ms.

Throughput of the output stream that passed through a router is shown in Fig. 3. Green color corresponds to the stream of 1500-bytes packets and red color corresponds to the stream of 64-bytes packets. The stream of 1500-bytes packets experienced some transitional effect during the first three seconds. After that the stream was shaped perfectly. When we looked at packet timestamps, we found that packets were dispatched by the router with the precision of 1 packet to keep the volume of data sent during the shaping period to Bc bits. However, the stream of 64-bytes packets was not shaped correctly. The bursts were passed to the output. We found that the router processor was overloaded at this packet rate, packet size and shaping period configured.

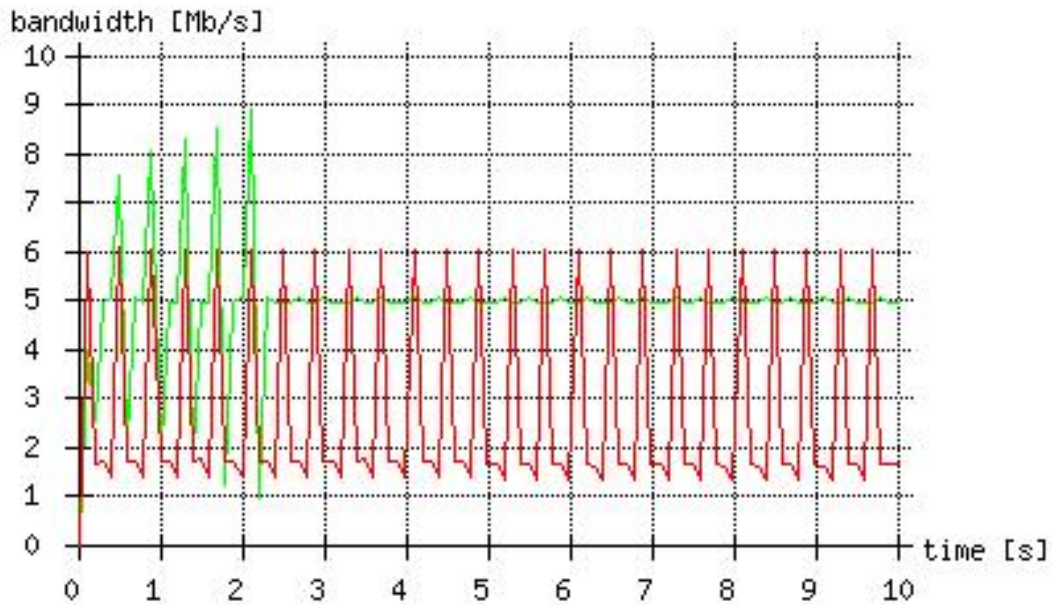


Figure 3: Throughput after traffic shaping

Average delay per measurement period of packets that passed through a router is shown in Fig. 4. Each measurement period, corresponding to one shaping interval, contains two sections - one with very low delay and one with significantly higher delay. The reason is that the router passes some of the packets from each input burst to the output immediately (with packet processing delay) until B_c bits are sent out. The remaining packets of the input burst must be delayed and sent during following shaping intervals to keep the maximum of B_c bits sent out in a shaping interval.

Delay distribution of packets that passed through a router is shown in Fig. 5. The first column corresponds to packets sent during the same shaping period in which they arrived. The other three significant columns correspond to packets that were delayed and sent out during one of the next three shaping periods.

6 Conclusion

We constructed a complete environment for precise measurement of all primary network QoS characteristics, that is packet loss, throughput, delay, delay variation as well as distribution of delay and delay variation. The cost of the whole environment was about 300 Euro (basically the price of the GPS receiver, the price of other components was negligible except for two PCs, which are however not dedicated for measurements).

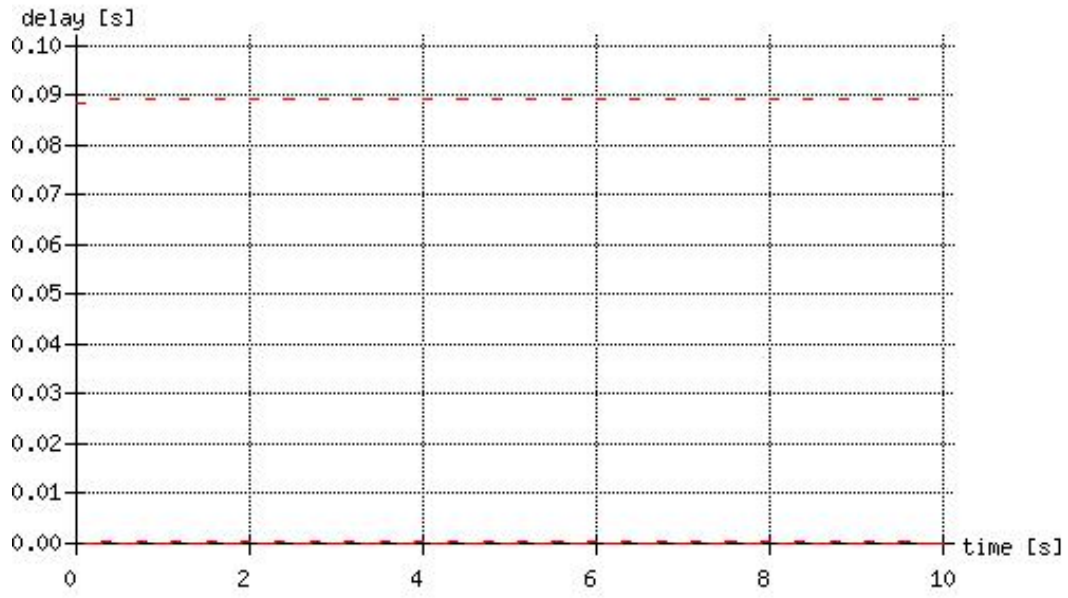


Figure 4: Delay after traffic shaping

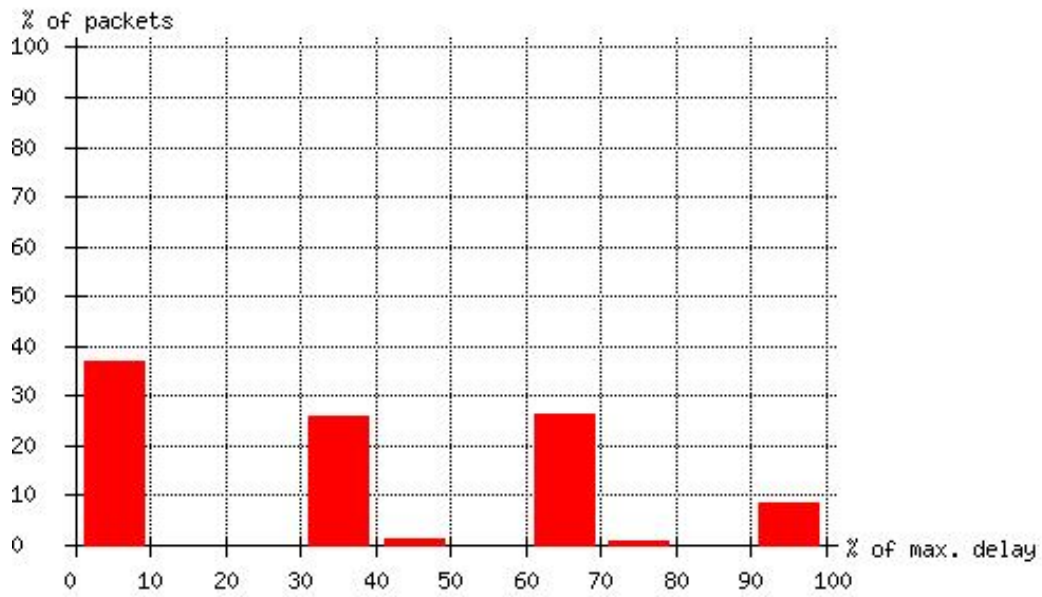


Figure 5: Delay distribution after traffic shaping

RUDE/CRUDE is a simple tool that can be used to generate and measure network traffic. It is distributed under GPL license and is thus available to everyone. The tool has been used in QoS measurements done at Tampere University of Technology [tampere-measurements].

One-way delay, which is traditionally the parameter most difficult to measure due to the need of time synchronization, can be measured with the precision about 2 microseconds, which is sufficient for a broad class of QoS measurements. With current PC hardware configuration available, testing streams of the specified shape and the rate of up to 500 Mb/s can be generated. Comparable commercial measurement systems are much more expensive.

We successfully used the described environment for the measurement of the precision of traffic shaping in a router [ubik-traffic-shaping]. Arbitrarily adjustable measurement granularity and high precision allowed us to verify the correctness of the traffic shaping algorithm.

We plan to support adjustable correction of certain sources of imprecision discussed in section 3.2 by command line arguments and improve graphical rendering of resulting graphs.

References

- [radius] C. Ligney: *RADIUS Accounting*, Request for Comments 2139, Internet Engineering Task Force, June 2000.
- [tiziana-microflow] Tiziana Ferrari: *Support of Delay and Jitter Requirements*¹, research project.
- [sls] Danny Goderis et al.: *Service Level Specification Semantics and Parameters*², Internet Engineering Task Force, November 2000.
- [ubik-traffic-shaping] Sven Ubik: *Traffic Shaping Precision test*³.
- [victor] Victor Reijs: *Perceived quantitative quality of applications*⁴.
- [rfc2398] S. Parker, C. Schmechel: *Some Testing Tools for TCP Implementors*⁵, Request for Comments 2398, Internet Engineering Task Force, August 1998.
- [netperf] *Netperf: A Network Performance Benchmark*⁶.

¹<http://www.cnaf.infn.it/ferrari/tfngn>

²<http://www.ietf.org/internet-drafts/draft-tequila-sls-00.txt>

³<http://www.cesnet.cz/english/projects/qosip/testShaping7500Ge>

⁴http://www.heanet.ie/Heanet/projects/nat_infrastruct/perceived

⁵<http://www.ietf.org/rfc/rfc2398.txt>

⁶<http://www.netperf.org>

- [ntp] D.L. Mills: *Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI*⁷, Request for Comments 2030, Internet Engineering Task Force, October 1996.
- [ntp-precision] Ulrich Windl, David Dalton, Marc Martinec (eds.): *The NTP FAQ and HOWTO*⁸, section 5.1.10.
- [rude] Juha Laine, Sampo Saaristo, Rui Prior: *RUDE and CRUDE: Real-time UDP Data Emitter and Collector*⁹.
- [mgen] Brian Adamson, Sean Gallavan: *The Multi-Generator (MGEN) Toolset*¹⁰.
- [tcpdump] *Tcpdump: traffic dump utility*¹¹.
- [gdlib] Thomas Boutell, et al.: *gd: A Graphics Library*¹².
- [qosinip] Sven Ubik, Vladimir Smotlacha: *QoS in IP*¹³, research project, CESNET, 2001.
- [pllfill] D.L. Mills: *Adaptive hybrid clock discipline algorithm for the Network Time Protocol*, IEEE/ACM Transactions on Networking, 6, 5 (October 1998), 505-514.
- [stone] J. Mogul, D. Mills, J. Brittonson, J. Stone and U. Windl. *Pulse-per-second API for Unix-like operating systems*¹⁴, Request for Comments 2783, Internet Engineering Task Force, March 2000.
- [nanokernel] D.L. Mills and P.-H. Kamp. *The nanokernel*, Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting, Reston VA, November 2000.
- [ripe-box] *Test Traffic Project*¹⁵.
- [schmid] Ulrich Schmid, Martin Horauer and Nikalaus Kero. *How to Distribute GPS-Time over COTS-based LANs*, Proc. Precision Time and Time Interval (PTTI), November 2000.

⁷<http://www.ietf.org/rfc/rfc2030.txt>

⁸<http://www.eecis.udel.edu/ntp/ntpfaq/NTP-a-faq.htm>

⁹www.atm.tut.fi/rude/

¹⁰<http://manimac.itd.nrl.navy.mil/MGEN>

¹¹<http://www.tcpdump.org>

¹²<http://www.boutell.com/gd>

¹³<http://www.cesnet.cz/english/project/qosip>

¹⁴<http://www.ietf.org/rfc/rfc2783.txt>

¹⁵<http://www.ripe.net/test-traffic>

[tampere-measurements] J. Laine, J. Harju, J. Karjalainen, L. Lemponen and S. Saaristo. *Real-Time Traffic Measurements in a Differentiated Services Network*, accepted to ICC 2001, Helsinki, Finland, June 11-15, 2001.