SECOND QUARTER 2005, VOLUME 7, NO. 2

IEEE COMMUNICATIONS SURVEYS
The Electronic Magazine of Original Peer-Reviewed Survey Articles

www.comsoc.org/pubs/surveys

APPLICATION-ORIENTED NETWORK METROLOGY: METRICS AND ACTIVE MEASUREMENT TOOLS

FABIEN MICHAUT AND FRANCIS LEPAGE, CRAN CNRS UMR7039

ABSTRACT

The science of communication network metrology consists of measuring the performance of networks. This article is a tutorial on application-oriented measurement tools and techniques for IP networks. First the principles of active measurements are introduced and two important active measurement initiatives are presented. Since metrology often requires precision in timing, a basic overview related to time in computers and networks will be presented, followed by a definition of principal network metrics. Metrics that will be discussed are one-way delay, delay variation, round-trip time, packet loss, packet reordering, route, and bandwidth. For each parameter, a definition is given and related measurement tools and techniques are presented.

Communication network metrology is the science of measuring network performance. It is a new research area for packet-switched networks. The first significant work was performed by Paxson [1] in the mid 1990s.[1] He used a measurement infrastructure allowing him to capture the traces of 20,000 end-to-end TCP connections between 35 hosts in nine different countries. The study of the traces was completed with an analysis of end-to-end routes (determined with the traceroute utility). This was the first study enabling the observation of end-to-end Internet traffic dynamics: routing stability, paths asymmetry, TCP dynamic, packets out-of-order delivery, etc.

Since then, network metrology has become the subject of many research projects and laboratories (Surveyor [2], National Internet Measurement Infrastructure (NIMI) [3–5], Réseaux IP Européens (RIPE) [6], Netsizer [7], Cooperative Association for Internet Data Analysis (CAIDA) [8], SPRINT/IPMON [9], METROlogie POur L'Internet et ses Services (METROPOLIS) [10]). Some of them have no clear objectives (storing data in measurement databases, measuring Internet growth, etc.). Others focus on traffic characterization, traffic modeling, the study of traffic matrices, and network cartography. Traffic characterization consists of determining the traffic volume in a given network. It also deals with variability of traffic volume and nature with time. The traffic nature is the repartition of the traffic according to protocol (TCP, UDP, etc.) and application (Web, mail, real-time multimedia). Studies are also being carried out regarding packet size, the number of simultaneous flows, flow size, and composition. The purpose of traffic modeling is to determine models of packet arrival, flow arrival, and packet loss processes. Other studies focus on paths that are used by packets in the network and analyze routing dynamics and path symmetry. Finally, certain projects try to map out the Internet by checking all IP addresses in the network. All these studies have strong repercussions [11]:

- For Internet Service Providers (ISPs): traffic characterization and modeling create easy dimensioning of networks.
- For network equipment designers: packet size repartition is crucial when designing routers.

Moreover, now that the Internet is used for applications with Quality of Service (QoS) constraints (real-time applications, etc.), ISPs and their customers negotiate QoS levels through a Service Level Agreement (SLA). In this context, metrology is useful in verifying that SLAs are met.

Network measurement can be performed either actively or passively. Surveyor, NIMI, RIPE, and Netsizer are examples of active measurement projects, while CAIDA and SPRINT/IPMON work on passive measurements. METROPOLIS deals with both measurement classes.

[1] Previous works on telecommunication lines or circuit-switched networks focus on similar performance parameters (delay, jitter, etc.). However, conceptual and technical differences between these networks and packet-switched networks induce a new metrology approach.

1553-877X/05/$20.00 © 2005 IEEE

Passive techniques are carried out by observing network traffic flows. They consist of capturing packet headers and analyzing them. The best example of a capture tool is tcpdump, which is based on the libpcap library [12]. These techniques do not add to network traffic. Passive measurement can be done on two levels:

- At a microscopic level measurements are performed on each packet traveling across the measurement point. An example of collected information is packet size.
- At a macroscopic level measurements are performed on flows. In this case, aggregation rules are necessary to match packets into flows. Examples of collected data are the number of flows per unit of time, flow bitrate, etc. The "Real-time Traffic Flow Measurement" (RTFM) working group of the Internet Engineering Task Force (IETF) has worked on macroscopic measurements. It has helped create a general passive measurement architecture and describe the manner in which aggregation rules are defined. Today this working group is closed.

Passive measurement techniques are particularly suitable for (but not limited to) traffic engineering because they show flow dynamics and distribution. The main problem with passive measurement is the data volume. The volume of captured data can become very large on high-capacity links. Moreover, it is hard to obtain end-to-end measurements passively: the presence of traffic traveling between two measurement points cannot be ensured and matching measurements that are performed on different measurement points is difficult. Consequently, passive measurements are usually used to determine metrics characterizing one particular network element ("at-a-point" metrics).

On the other hand, active measurements are performed by sending probe packets to the network. The measurement flow travels from source to destination. Upon reaching its destination, it is possible to calculate metrics by analyzing them. Active measurements can determine the end-to-end QoS experienced by a measurement flow for a particular path and then measure the QoS as it is seen by applications. Additionally, active measurements offer the flexibility to send probe packets streams with particular properties (bitrate, packet size, etc.). The main drawback of active measurements is that additional network traffic is introduced. This "intrusive" (or "invasive") characteristic can potentially modify the properties that are trying to be measured. First, it can result in measurement errors or bias; second, it can lead to network overload. The measurement traffic thus has to be limited to avoid network disturbance and measurement errors.

As previously mentioned, the major idea behind active measurements is to perform QoS monitoring and to verify SLA. However, since active measurements determine the end-to-end QoS as seen by applications, this type of measurement is also useful for distributed applications (particularly when complete control of network resources is not possible). In this case, an understanding of the QoS of the network can allow applications to adapt their execution to resource fluctuations. For example, applications can adapt their throughput to the loss of the network or the size of play-out buffers to the network delay [13]. More complex application adaptation schemes have been proposed [14–19]. In this context, we will examine new developments of application-oriented metrology techniques and tools. Note that metrology for traffic engineering is beyond the scope of this article.

The main objective of this article is thus to give a tutorial on application-oriented measurement tools and techniques. The first section presents active measurement generalities. It presents active measurement principles and limitations, and discusses the IETF working group dedicated to active measurement and the NIMI project. Since active measurements often need time and clock constraints to be achieved, the second section deals with timing considerations. Each of the next seven sections is devoted to the following network parameters: one-way delay, delay variation (also referred to as jitter), round-trip time, packet loss, packet reordering, route, and bandwidth measurement. Definitions, measurement techniques, and tools[2] are presented for each parameter. Then we discuss the problem of "intrusiveness" that occurs with active measurements. Finally, we present our conclusions.

Active Measurements

Principles

Active measurements consist of sending probe packets into the network from a source host (probe sender) to a destination host (probe receiver). By choosing particular properties at departure (packet size, inter-departure time, bitrate, etc.), it is possible to calculate metrics by analyzing the probe stream characteristics (arrival time, inter-arrival time, etc.) at the destination. Thus, one can determine end-to-end metrics (from the source to the destination). Note that round-trip measurements can also be made by making the probe receiver send back responses to the source. In this case, measurements are performed at the source.

Active measurement tools can be classified into the following categories:

- Cooperative tools, which consist of separate sender and receiver programs that are respectively installed on the source and destination hosts.
- Non-cooperative tools, which consist of only one program that includes the sending and receiving tasks. It avoids installing dedicated software on the target end-system.

Notes related to security:

- Certain measurement tools are based on forcing routers and hosts to generate ICMP packets. For security purposes, routers and hosts can be configured to limit (or avoid) the generation of ICMP messages. Additionally, certain network sites use a firewall to filter incoming ICMP traffic. In these cases, measurements are disrupted. This limitation concerns all the metrology techniques using ICMP messages.

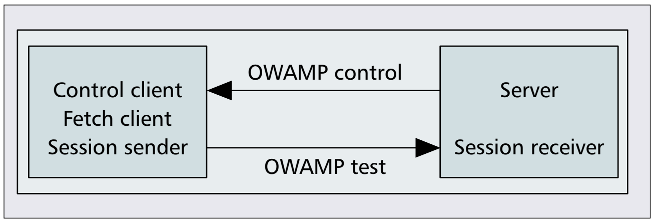
Some tools require having a listening TCP socket on the target host (such as Web servers). They use TCP behavior to force the target host to send packets back (for example, by sending SYN packets to the target). Since sending large streams of SYN packets can be interpreted as a denial of service attack, many network sites often use a firewall to filter or limit incoming SYN traffic. This results in disrupting measurements.

For precise results, certain active measurement techniques require strong time-related constraints to be achieved. Examples of these constraints are:

- Timing accuracy in sending probes (for example, when sending probe streams that have to match inter-departure constraints.)
- Accurate time stamping of probes upon arrival.
- Accurate time synchronization of the source and the destination to allow clock comparison between the two hosts.

Then, active measurement techniques will often induce strong constraints on clocks (clock accuracy, clock stability, etc.), and measurement errors will result from clock characteristics. Timing considerations are presented later.

[2] *Most of the tools presented here are listed and downloadable through the Caida laboratory Web site [8].*



■ **Figure 1.** *One-Way Active Measurement Protocol: different logical roles can be played by the same host.*

A parameter measurement is always performed at a specific protocol layer. If the parameter needs to be used at another layer, the result must be modified in order to take into account the protocols or devices operating between measurement and using layers. This "encapsulation effect" is discussed in the following section. Next, we will discuss the group in the IETF that works on active measurements and the NIMI project, which was the first large-scale project dealing with active measurements.

Encapsulation Effect

This article deals with metrology at the application layer, but some of the tools presented work at lower protocol layers. The user must be aware of the significant differences that can exist between the measured value and the real value of the parameter at the application layer. Likewise, a measurement result could appear incorrect when a parameter value is known, but at another layer. For example, the capacity of a channel is often known at the data link layer and the capacity measurement tools at the IP layer could give results that are difficult to analyze. Layer overhead effect is presented in the section dedicated to bandwidth.

Prasad *et al.* explain in [20] the effect of level 2 encapsulation and its consequences according to packet length, and in [21] they explain the effect of store-and-forward devices in capacity estimation. There are also deterministic and probabilistic parts in the value difference between layer 2 and layer 3.

In conclusion, it is recommended to measure a parameter at the user level. If this is not possible, it is necessary to carefully modify the measured value to obtain an accurate estimation of the useful value.

IP Performance Metric (IPPM) Working Group

The IPPM working group of the IETF deals with active measurements. It defines standard metrics and statistics, which allow us to compare active measurements produced by different tools, for example. The IPPM has written RFCs on metrics measurement and statistics calculation related to different parameters (packet loss, delay, etc.). In the remainder of this article, we give metric definitions according to IETF standards when available.

The IPPM recommends Poisson and periodic sending processes of probe packets [22, 23]. The latter appears better adapted to measuring continuous multimedia streams. Note that IPPM is working on defining a Management Information Base (MIB) to retrieve the results of standard metrics. The IPPM is also actively working on the specification of a measurement protocol, called the One-Way Active Measurement Protocol (OWAMP) [24], to enable communication among test equipment. The intent is to create a simple protocol that will allow equipment from different manufacturers to interoperate, that is to say, initiate test streams and the exchange of packets to collect metrics. In fact, OWAMP consists of two inter-related protocols:

- The *OWAMP-Control* protocol is used to initiate, start, and stop test sessions and collect their results.
- The *OWAMP-Test* protocol can exchange probe packets between two end-points.

OWAMP defines the following logical roles (different roles can be played by the same host, Fig. 1):

- The *Session-Sender* is the probe sender (source) of a test session.
- The *Session-Receiver* is the probe receiver (destination) of a test session.
- The *Server* manages test sessions. It is able to configure session end-points and give test session results.
- The *Control-Client* is the end-system that requests test sessions.
- The *Fetch-Client* is the end-system that requests test session results.

OWAMP sends test streams with negotiated characteristics such as packet size, number of packets, inter-departure time, etc. For each probe, the server will return to the Fetch-Client the send and receive timestamps and an indication if the probe was lost. The Fetch-Client is then able to calculate all one-way metrics defined by the IPPM (loss, delay, etc.).

OWAMP represents a significant advancement in active measurement because it will "standardize" active measurement tools. We will later discuss existing measurement tools that are often developed using proprietary techniques and which are not interoperable.

Owping/owampd [25] and QoSMet [26] implement OWAMP. These tools are cooperative tools, but it should be noted that clocks of hosts that implement OWAMP session senders and receivers must be synchronized.

National Internet Measurement Infrastructure

NIMI was a project started by Vern Paxson to instrument the global Internet. The main objectives of NIMI were:

- To diagnose performance problems.
- To study how the network behaved and evolved.
- To measure performance delivered by ISPs.
- To facilitate public access to Internet measurements.

A key characteristic of NIMI is that the measurement infrastructure it has defined is scalable. This characteristic is fundamental in the deployment of thousands of measurement points and increasing the number of available measurable paths. In this context, the NIMI architecture consists of a measurement probe (NIMI probe), a measurement client (MC), a configuration point of contact (CPOC), and a data analysis client (DAC) [27]. The MC injects measurement requests into the NIMI infrastructure. It is the only component (which can run from a desktop machine) operated by the end-user. A CPOC manages the probes that are in its control zone (it gives policies to each probe, etc.). A NIMI probe is composed of a daemon *nimid* and plugable measurement modules. Modules are external to nimid and are third-party software, such as treno [28] or traceroute [29]. nimid is unaware of the modules and only executes this software. "Wrappers" are developed to obtain the results generated by the modules and put them into a standardized form. The NIMI probe ships results to a DAC, which acts as a repository and post-processor of the data.

Bulk-Transfer-Capacity (TCP throughput), packet loss, and hop count are end-to-end measurements collected by NIMI probes.[3] NIMI is also used to determine a matrix of distance in the Internet. The NIMI infrastructure is used with other metrol-

[3] *NIMI website allows to access to collected measurements [3].*

ogy projects such as MINC (Multicast-based Inference of Network-internal Characteristics [30, 31]) and METROPOLIS.[4]

Timing Considerations

As seen in the previous section, active measurement techniques often result in constraints to the clocks in the measuring end-systems (clock accuracy, clock stability, etc.).[5] This section presents a basic overview of time and computer clocks.[6] We first define the parameters related to the uncertainty of clocks. As measurement tools are essentially developed under Unix, we then discuss how a Unix clock works. Clock synchronization is needed in order for several measurement techniques to be performed, and so we will explain the basis of clock synchronization. Finally, numerous techniques assume that probe packets are accurately sent at predefined moments and/or time stamped upon arrival. We will explain how this can lead to measurement error.

Definitions

Following are parameters related to clock uncertainty [33]:

- Synchronization error (offset): measures the extent to which two clocks agree on what time it is. It is noted T_{SYNCH}.
- Accuracy: measures the extent to which a given clock agrees with Coordinated Universal Time (UTC).
- Resolution: measures the precision of a given clock (tick). For example, the clock on an old operating system (OS) might have a resolution of about 10ms (Linux, older versions than 2.2.0, etc.).
- Skew: measures the change of accuracy, or of synchronization, with time. For example, a clock might gain 1.3ms per hour, and thus be 27.1ms behind UTC at one time and only 25.8ms an hour later. We will also speak of the skew of one clock in relation to another.
- Drift [34]: measures the change of skew with time. Drift changes with temperature, and is often negligible. Note that drift is sometimes used to talk about skew. In some papers, skew is used to talk about synchronization.

Unix Clock [32]

A computer contains two clocks: a hardware clock and a system clock. The hardware clock allows the computer to keep track of time when the computer is turned off. It is used when the computer is booted up to update the system clock. This latter is implemented as a counter of timer interrupts that are generated by the computer quartz oscillator. The period of interrupts is usually 10ms. It means that when an interrupt occurs, a periodic task increments the system clock with a value corresponding to the period of the timer interrupts (here, 10ms). This value characterizes the granularity of the clock and is called the *tick*. Such a granularity is too crude. To improve it, the OS thus uses the clock cycle register (which counts central processing unit [CPU] clock cycles) to interpolate between the interrupts. This technique requires knowing the number of CPU cycles per time unit. It is measured at boot time by counting the number of cycles between several periods of interrupts (usually around 50ms on Linux and 1.3s on RT-Linux).

This kind of clock is prone to error:

- Changes in frequency of the quartz oscillator (due to changes in temperature and the age of the oscillator) introduce skew.
- The measurement of the number of CPU clock cycles per time unit is based on using the oscillator during a short time interval. The interpolation thus inherits the skew of the standard oscillator.
- This latter measurement suffers from integer arithmetic effects.

Clock Synchronization

Clock synchronization is a complex problem. It can be achieved by using Global Positioning System (GPS) cards, radio clock receivers, or with Network Time Protocol (NTP) servers. Using GPS cards is the most precise yet most expensive solution, requiring the installation of exterior antennas (on the roof). Radio clock receivers, while cheaper than GPS cards, are not as accurate for many reasons.[7] In fact, using NTP servers is the most common solution for clock synchronization; it is expected to be accurate to about 10ms on a WAN [35]. NTP synchronization, while not perfect, can nevertheless offer significant accuracy given the scale on which measurements are made. For example, when measuring delays to the order of hundreds of milliseconds, a synchronization error of 10ms may be sufficient.

NTP daemon is used in every scheme. It regulates clock frequency and controls clock skew [34, 36] according to information given by the time source (GPS, NTP servers, etc.). Instead of directly changing the value of the clock, NTP controls its frequency by adjusting the clock tick. This technique prevents discontinuities in the clock. Note that discontinuities might happen with implementations of NTP for Windows. In fact, these implementations seem to be Simple Network Time Protocol (SNTP) implementations [37].

In [32] the authors have shown that using NTP has positive effects on clock stability, but only on large time scales. Indeed, when synchronized to NTP servers, the NTP daemon is subject to estimate false clock offset due to changes in network delay resulting in unnecessary clock adjustments.[8]

Errors and Uncertainties Related to Wire-Time vs. Host-Time

Measurement techniques often imply sending packets at specific times and timestamping upon their arrival. These timings are referred to as "wire times." In practice, we can only directly measure the time from when the source grabs a timestamp just prior to sending the test packet and when the destination grabs a timestamp just after having received the test packet. These timings are refereed to as "host times." Measurements then include errors related to these uncertainties at the sender and receiver sides, respectively.

[4] *The NIMI infrastructure has been extended in Switzerland and in France within the METROPOLIS project.*

[5] *However, the reader should be aware that not all of the measurement tools require a highly accurate clock. Time-constraints depend on the chosen measurement technique and the order of magnitude of the measured parameter.*

[6] *The paper [32] gives a good overview of time considerations.*

[7] *This receiver accuracy depends on the propagation conditions of the radio-frequency signal.*

[8] *In this context, Pazstor and Veitch have proposed in [38] an alternative software clock based on the clock cycle register. (Here, the accuracy of the estimation of the CPU cycle period is improved.) It offers a better resolution (around 1ns for a 1GHz CPU) and better stability. This clock is suited for measurements that do not require end-to-end synchronization.*

The error at the sender side depends on the latency to timestamp the packet, move the packet from user to kernel space, and transmit it on the network interface card. Note that the OS scheduler can possibly assign the computing resources to other concurrent processes between the time-stamping and the sending operation. In [32] the authors explain that this phenomenon can be reduced significantly by carefully designing the sender software (affecting a high priority to the sending task or programming it on a real-time OS as a real-time task).

The error at the receiving end depends on the latency to receive a packet in the OS, move the packet from kernel to user space, and timestamp the packet upon arrival. To minimize this error, one can use network interface cards dedicated to packet capture with integrated high accurate timestamping [32] (these cards are directly synchronized to a GPS receiver). Another solution is to timestamp packets at the kernel level: packets are timestamped by the network interface driver when entering and before moving to user space. Kernel-level time-stamping can be performed on Unix using the SO_TIME-STAMP datagram socket option. In this case, the recvmsg() call will return a timestamp corresponding to when the datagram was received. Kernel-level timestamping can also be made using the libpcap and Winpcap libraries (on Unix and Windows, respectively) [12] that can capture packets in the interface driver. To our knowledge, quantifying the difference between kernel-level and application-level timestamping is an open issue. It could be done by comparing kernel-level and application-level timestamps for the same packet. However, this kind of measurement strongly depends on hardware specifications (CPU frequency, etc.), the design of measurement software, system specifications, and load (OS version, number of user and kernel tasks that are being executed, etc.).

One-Way Delay

Definitions

The one-way delay is the time it takes a packet to go from source to destination. It includes propagation delays, transmission delays, and queuing delays in intermediate systems (routers, switches).

The IPPM defines [33] the one-way delay for a type-P packet as follows:

Consider two network hosts, SRC and DST. For a real number dT*, the "Type-P-One-way-Delay" from SRC to DST at* T *is* dT *means that SRC sent the first bit of a Type-P packet to DST at wire-time* T *and that DST received the last bit of that packet at wire-time* T + dT.

The IETF defines the following errors and uncertainties and proposes the following methodology for measurements calibration [33].

Errors and Uncertainties Related to Clocks — T_{SRC} refers to the observed time when the packet is sent by the source clock. T_{DST} refers to the observed time when the packet is received by the receiver clock. The uncertainty in a measurement of one-way delay is related to uncertainties in the clocks of the source and destination hosts.

- The synchronization error between the source clock and the destination clock will contribute to error in the delay measurement. If we know T_{SYNCH}, we can correct for clock synchronization by adding T_{SYNCH} to the uncorrected value of $T_{DST} - T_{SRC}$.
- The accuracy of a clock is important only in identifying the time at which a given delay is measured. Accuracy is not crucial to the accuracy of measurement of delay.

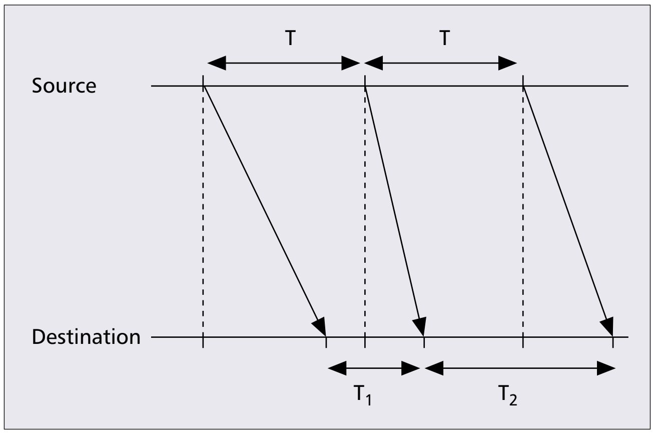


Figure 2. *Delay variation — impact on data periodicity.*

- The resolution of a clock adds to uncertainty about any time measured with it. Resolutions of the source clock and the destination clock are denoted as R_{SRC} and R_{DST}, respectively.
- A part of the skew $T_{SYNCH} = f(t)$ can be approximated as a linear function plus some higher-order terms. An understanding of the linear component can be used to correct the clock. Using this correction, the residual T_{SYNCH} is made smaller. The function $E_{SYNCH}(t)$ is used to denote an upper bound on the uncertainty in synchronization.
- Measurements include additional uncertainties related to wire-time vs. host-time at the sender and receiver ends: H_{SRC} and H_{DST}.

Put together, the clock-related problems introduce an uncertainty of $E_{SYNCH}(t) + R_{SRC} + R_{DST} + H_{SRC} + H_{DST}$.

Calibration — The goal of calibration is to determine measurement error. From the previous sections, error in measurements can be bounded by $E_{SYNCH}(t) + R_{SRC} + R_{DST} + H_{SRC} + H_{DST}$.

Error can be evaluated in certain configurations using an isolated network: clock-related uncertainties are minimized through the use of a GPS time source. The sum of $E_{SYNCH}(t) + R_{SRC} + R_{DST}$ is small. The host-related uncertainties, $H_{SRC}+H_{DST}$, can be bounded by connecting two instruments with an isolated LAN segment, for example. If the test packets are small, network connection will have a minimal delay that may be approximated by zero. The "average value" of repeated measurements is the systematic error, and the variation is the random error. Unfortunately, this calibration technique cannot be used for end-systems that are geographically distant, as in the case of computers connected via the Internet.

Measurement Methodology and Tools

To measure one-way delay, a packet is stamped with the current time and sent to the destination. At the destination, the packet timestamp and the destination clock are read out. The delay is then equal to the difference between the two values. This technique implies that the clocks of the two end-systems are synchronized.

One-way delay measurement is implemented in owping/owampd [25] and QoSMet [26]. These tools can measure one-way delay along the forward and reverse path according to IPPM recommendations.

Notes:

- In practice, the one-way delay is often calculated as half the round-trip time. But as Paxson explains in [1], paths tend to be progressively more asymmetric. The authors in [39] observed the same tendency for links (DSL, modem, satellite, etc.). Thus, this calculation is incorrect.

```
$ ping 64.124.140.199
PING 64.124.140.199 (64.124.140.199) from 193.50.39.64 : 56(84) bytes of data.
64 bytes from 64.124.140.199: icmp_seq=0 ttl=241 time=163.773 msec
64 bytes from 64.124.140.199: icmp_seq=1 ttl=241 time=159.967 msec
64 bytes from 64.124.140.199: icmp_seq=2 ttl=241 time=159.973 msec
64 bytes from 64.124.140.199: icmp_seq=3 ttl=241 time=159.975 msec
64 bytes from 64.124.140.199: icmp_seq=4 ttl=241 time=159.975 msec
64 bytes from 64.124.140.199: icmp_seq=5 ttl=241 time=159.976 msec
--- 64.124.140.199 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max/mdev = 159.967/160.606/163.773/1.471 ms
```

Figure 3. *Ping output.*

- The authors in [1] and [39] give an overview of problems that can arise when measuring delay.

One-way Delay Variation

Delay variation is a key metric for many applications. For example, a high delay variation can disrupt the transfer of continuous media of voice (Fig. 2). In this case, the "rhythm" of audio data delivery is crucial to ensure quality sound restitution.

Definitions

The IPPM defines [40] the delay variation for a type-P packet as follows:

Consider two network hosts, SRC and DST. For a real number ddT *"The type-P-one-way-ipdv" from SRC to DST at* T_1, T_2 *is* ddT *means that SRC sent two packets, the first at wire-time* T_1 *(first bit), and the second at wire-time* T_2 *(first bit) and the packets were received by DST at wire-time* $dT_1 + T_1$ *(last bit of the first packet), and at wire-time* $dT_2 + T_2$ *(last bit of the second packet), and that* $dT_2 - dT_1 = ddT$.

Errors and Uncertainties Related to Clocks — When measuring one-way delay variation, errors and uncertainties are:
- Related to clock resolution. In this case, the uncertainty is two times that of a single delay measurement.
- Related to wire-time vs. host-time.

Additional Metrics — The IPPM also defines the peak-to-peak delay variation. It is defined as the difference between the maximum and minimum values of a sequence of delay variation values. The use of the term "jitter" is deprecated. Nevertheless, this term is often used to define the absolute value of the delay variation. In some cases, jitter is computed by taking the absolute value of delay variation values and applying an exponential filter indicated in [41]. Some authors define the delay variation as the average deviation of the delay.[9]

Measurement Methodology and Tool

As delay variation is computed as the difference between the delay of two consecutive packets, it requires the measurement of one-way delay.

Note: Measurement of delay variation does not need clock synchronization because errors will cancel each other when the delay difference is calculated. Additionally, the effect of skew is rather small over the concerned time scales.

One-way delay variation and peak-to-peak delay variation measurement is implemented in QoSMet [26]. QoSMet can measure these metrics along the forward and reverse path according to IPPM definition.

[9] *In this case, delay is assumed to have a normal distribution [42].*

Round-Trip Time (RTT)

Definitions

The RTT is the delay from the source to the destination and back.

The IPPM defines [43] the RTT for a type-P packet as follows:

Consider the network hosts, SRC and DST. For a real number dT*, the "Type-P-Round-trip-Delay" from SRC to DST at* T *is* dT *means that SRC sent the first bit of a Type-P packet to DST at wire-time* T*, DST received that packet, immediately sent a Type-P packet back to SRC, and that SRC received the last bit of that packet at wire-time* T + dT.

Errors and Uncertainties Related to Clocks — When measuring RTT, errors and uncertainties are:
- Related to clock resolution. Timestampings are performed on the same host, and then error is $2 * R_{SRC}$.
- Related to wire-time vs. host-time, $H_{initial} + H_{final}$.
- Related to the destination producing a response: H_{refl}.

Measurement error can be estimated using a similar approach as for one-way delay (isolated network, small size packets, here we have $2 * R_{SRC} + H_{initial} + H_{final} + H_{refl}$).

Measurement Methodology and Tool

To measure RTT, a packet is stamped with the current time and sent to the destination. When the packet is completely received at the destination, it sends a corresponding response packet back to the source. The RTT is equal to the difference between the receiving time at the source and the time stamp value. The measurement is easier than one-way delay measurement, because there is no issue with source and destination clock synchronization. Clock skew is negligible due to the measurement time-scale. The greater problem is anything that would cause a discontinuity in the clock. This might happen with the SNTP protocol, for example.

The best known RTT measurement tool is ping (Fig. 3). Ping is a non-cooperative tool that measures RTT using ICMP echo-reply probes. When a host receives ICMP echo packets, it sends back reply packets that are the same size as echo packets. The source then sends ICMP echo packets to the target host and measures the time it takes the corresponding ICMP reply packets to come back.

Packet Loss

Definitions

The reliability of the path is expressed by the packet loss rate. This metric is equal to the number of non-received packets divided by the total number of sent packets.

The IPPM defines [44] the one-way packet loss for a type-P packet as follows:

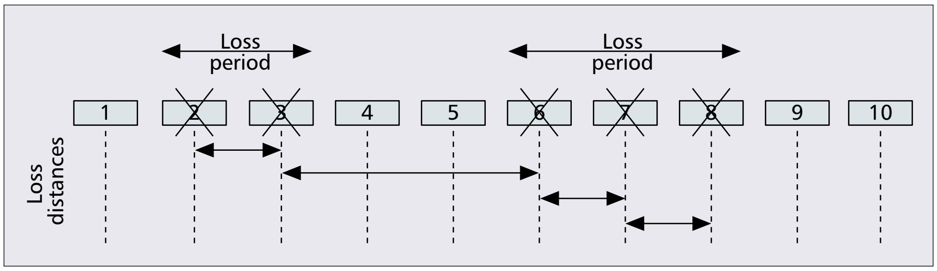


Figure 4. *Loss period and loss distance.*

Consider two network hosts, SRC and DST. The "Type-P-One-way-Packet-Loss" from SRC to DST at T *is 0 means that SRC sent the first bit of a Type-P packet to DST at wire-time* T *and that DST received that packet. The "Type-P-One-way-Packet-Loss" from SRC to DST at* T *is 1 means that SRC sent the first bit of a type-P packet to DST at wire-time* T *and that DST did not receive that packet.*

Additional Metrics — It is generally agreed that a packet loss is an indication of congestion. The next packet may also get lost. As a result, the losses are dependent. In [45] the authors have analyzed Internet packet loss statistics. Their results verify that packet loss exhibits dependence. The loss pattern (or loss distribution) is a key parameter in certain applications (such as voice and video). Two different loss distributions for the same loss rate could potentially produce different performance degradation.

In this context, the IPPM proposes metrics and statistics on loss pattern [46]. Assuming that consecutive packets in a time series sample are given sequence numbers that are consecutive integers, it defines two metrics as loss distance and loss period. The loss distance is the difference in sequence numbers of two consecutively lost packets that may or may not be separated by received packets. A loss period is a sequence of consecutive packets that have been lost, starting when a packet is lost and the preceding packet is received, and ending when a packet is received and the preceding packet is lost (Fig. 4). Using these two metrics, useful statistics can be calculated as the average length of loss periods and the average length of inter-loss periods.

The IPPM also proposes a statistic called one-way loss noticeable rate. A loss of a packet is said to be "noticeable" if the distance between the lost packet and the previously lost packet is no greater than Δ, where Δ is the "loss constraint." This statistic is useful for multimedia codecs that are able to sustain losses by "concealing" the effect of loss by making use of past history information. By choosing delta based on codec sensitivity, one can measure how "noticeable" a loss might be for quality purposes. Figure 5 shows an example (from [46]) where loss constraint is equal to 99. A loss rate of one percent with a spread of 100 between losses may be more desirable for some applications compared to the same loss rate with a spread that violates the loss constraint.

Measurement Methodologies and Tools

According to IETF recommendations, detection of non-received packets is performed using time-out values. If the packet fails to arrive within a reasonable period of time, the packet is considered lost. The definition of reasonable is vague and depends on user needs [44]. Note that it is impossible to use a time-out value when the source and the destination clocks are not synchronized.[10]

[10] *In practice, loss detection can also be based on analysis of packet sequence numbers (assuming that a packet is lost when n next packets are arrived).*

One-way packet loss measurement, according to IPPM recommendations, is implemented in owping/owampd [25] and QoSMet [26]. These tools can measure loss along the forward and reverse path. QoSMet also implements the one-way loss noticeable rate metric.

Ping is the better known tool for measuring losses. It determines round-trip losses of ICMP probe packets (Fig. 3) and is not able to distinguish a difference between a loss that occurs in the forward or reverse path. Note that experiments conducted in [45] have shown that packet loss is highly asymmetric.

Alternative Methodology based on TCP Acknowledgment Scheme — Reference [47] introduces TCP-based measurement methodology that measures packet loss rate. It can estimate one-way loss rate in both directions of an end-to-end connection through observation of TCP behavior. The technique is based on TCP Acknowledgment (ACK) analysis. It operates in two phases. During the data-seeding phase, the source sends a series of TCP packets to the destination. The second phase is called the hole-filling phase. This phase is about discovering which of the packets sent in the previous phase have been lost by analyzing TCP ACK packets. If the last sent packet has been acknowledged, none of the packets have been lost. If this is not the case, the ACK can determine the number of the first lost packet (Fig. 6a). The source retransmits the corresponding packet and records that a packet has been lost. The procedure is repeated until the last packet sent during the first phase has been acknowledged.

Backward losses are determined by detecting ACKs that were lost. The main problem is knowing the number of ACKs that were sent by the destination. The ideal condition is that the target sends a single ACK for every data packet it receives. Unfortunately, this is not the case for most TCP implementations. Receivers do not respond immediately, but instead wait (from 100 to 500ms) for additional packets to limit the number of ACKs.

To force the destination to respond to each packet received, the methodology takes advantage of the TCP "fast-retransmit" algorithm. This algorithm was initially designed to allow a receiver to explicitly request the retransmission of a lost packet. The receiver uses it when he detects a "hole" in the received sequence numbers. Fast-retransmit sends an ACK immediately when an out-of-order packet is received, but for the last in-sequence TCP segment. During the data-seeding phase, the source skips the first sequence number and then ensures that all ensuing packets will be received out-of-sequence. It forces the receiver to acknowledge each received packet (Fig. 6b). Therefore, the source is able to calculate the loss rate on the reverse path knowing the number of received packets at the destination and the number of ACKs it has received . This measurement methodology is implemented in Sting [47], which is a non-cooperative tool.

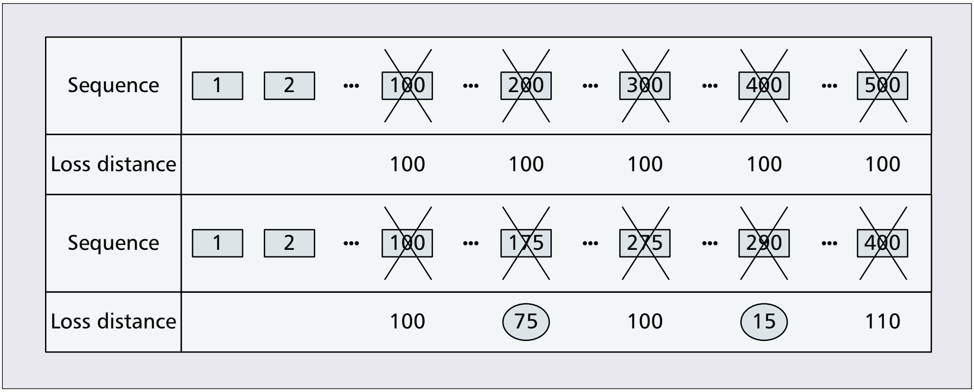


Figure 5. *Noticeable losses with Δ = 99. Encircled numbers indicate noticeable losses.*

PACKET REORDERING

Ordered delivery of packets is essential for real-time media streaming applications [48] and the TCP protocol [49–51]. Packet order must be considered in play-out buffer dimensioning of real-time streaming applications and in [50] it was clearly shown that out-of-order delivery affects TCP. For instance, out-of-order packets can cause TCP senders to unnecessarily retransmit packets and/or the congestion window to increase at a slower rate than normal. It has led some authors to propose modifications to TCP to better tolerate packet reordering [52, 53] (these works mainly focus on enhancing the retransmit scheme of TCP) and others to propose Partially Ordered Connection (POC) protocols that accept limited packet reordering [54–56].

Packet delivery order can be changed for different reasons:

- When packets from the same stream use different paths (multi-path routing [57]).
- In [50] it was shown that out-of-order delivery is essentially caused by parallelism in network devices and logical links (packets can take different paths through a switch, for example). Packet delivery order can then be modified even when using the same route.
- Packet reordering can occur when layer 2 retransmission occurs (particularly across wireless links) [49].
- Reordering occurs when packets of a flow are assigned to multiple buffers that have different service rates.

The reader can find a good overview on packet reordering in [50].

DEFINITIONS

As explained in [58], it is possible to interpret the reordering of packets differently. The following example is taken from [58]. Consider two packet sequences (1, 3, 4, 2, 5) and (1, 4, 3, 2, 5). The different possible interpretations are:

- *Non-respect of the total order*: Packets 2, 3, and 4 are out of order in both cases.
- *Non-respect of the packet number growth (losses are accepted)*: Only packet 2 is out of order in the first sequence, while packets 2 and 3 are out of order in the second.
- *Expected packet not arrived (losses are not accepted)*: Packets 3 and 4 are out of order in both the sequences.
- *Arrival sequence numbering does not correspond to the packet number*: Packets 2, 3, and 4 are out of order in the first sequence, while packets 4 and 2 are out of order in the second sequence.

This ambiguity results in different propositions of reordering definitions and metrics. Reordering is currently being discussed at the IPPM and two drafts have been proposed.

Jayasumana et al. Proposition — In [58] the authors define *out-of-order* packets as follows: *"When a packet other than the expected packet arrives, it is considered as an out of order packet, provided it is not a duplicate of an already received packet."* A packet is thus said to be an *early-packet* when it arrives before its expected place in the sequence; it is considered a *late-packet* when it arrives after its expected place. Based on these definitions and assuming that an arrived packet with a sequence number greater than the expected is stored in a hypothetical buffer to recover from reordering, the three following metrics are calculated:

- The reorder density is the distribution of buffer occupancy frequencies $FB[i]$ normalized with respect to the total number of occurrences $\Sigma FB[i]$, provided that $FB[i]$ is the number of times the buffer occupancy takes the value of i.
- The early-packet density is the distribution of early frequencies $FE[i]$ normalized with respect to the total number of occurrences $\Sigma FE[i]$, provided that $FE[i]$ is the number of packets that arrived i places early.
- The late-packet density is the distribution of late frequencies $FL[i]$ normalized with respect to the total number of occurrences $\Sigma FL[i]$, provided that $FL[i]$ is the number of packets that arrived i places late.

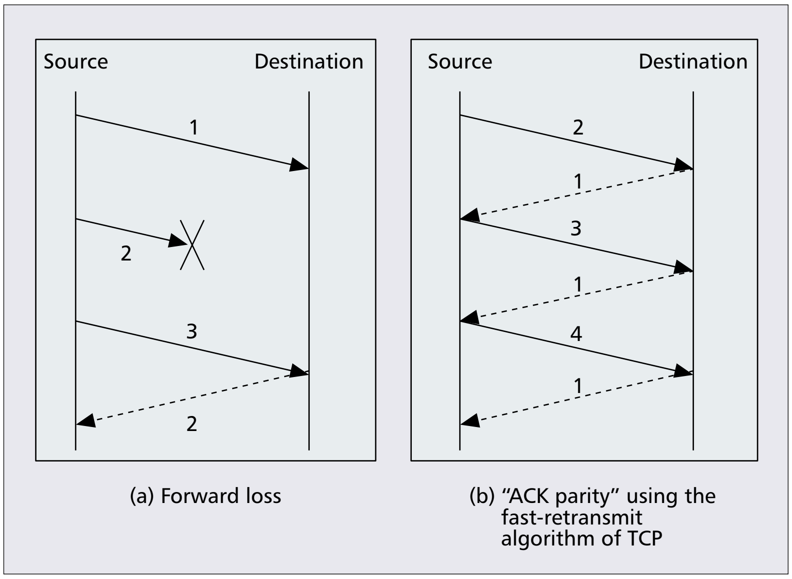
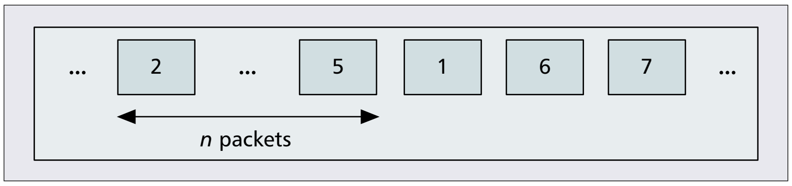


Figure 6. *Loss measurement technique based on TCP acknowledgment scheme.*

```
On successful arrival of a packet with sequence number s˜:
        if s >= NextExp, /* s is in-order */
                then
                NextExp = s + 1;
                Type-P-Reordered = False;
        else            /* when s < NextExp */
                Type-P-Reordered = True
```

■ Figure 7. *Type-P-reordered metric calculation.*



■ **Figure 8.** *n-Reordering metric.*

Note that to cope with packet losses, an *occupancy threshold* is defined. It expresses the tolerance of the application to the maximum allowed hypothetical buffer size. If an out of order packet needs to be stored in the hypothetical buffer already filled to the value of the occupancy threshold, the currently expected packet is considered to be delayed more than the tolerance and thus is assumed lost.

Morton et al. Proposition — The other draft [48] defines *out of order* or *reordered packets* as arriving packets with sequence numbers smaller than their predecessors. This is equivalent to the reordering definition of Paxson [1]. Then only "late" packets are declared reordered. It is the only way to distinguish reordering from packet losses (basing the definition on "premature" packets leads to ambiguity between reordering and losses[11]). This definition is specified in pseudo-code in Fig. 7 with Type-P-Reordered the calculated metric, *s* the packet sequence number applied at the source, in units of messages, and NextExp the Next Expected Sequence number at the Destination, in units of messages.

Additional Metrics — Additional metrics are proposed in the Morton *et al.* proposition [48]. The *Reordered-Ratio* metric defines the ratio of reordered packets of a stream of packets. The *Reordered-Extent* metric gives the extent to which packets are reordered, i.e. the maximum distance (in packets) from a reordered packet to the earliest packet received that has a larger sequence number. The extent then corresponds to "offset" metrics (*Late-Time* and *Byte-Offset*) indicating buffer time or storage in bytes that a receiver must possess to accommodate reordered packets. Finally, the authors have defined a TCP-relevant metric, *n-Reordering*, whose aim is to detect unnecessary TCP retransmissions due to the "fast-retransmit" algorithm, which is triggered by a series of duplicate ACKs. If the sender sees three duplicate ACKs, it assumes that the data (immediately after the byte being acknowledged) has been lost and retransmits that data. Out of order delivery can cause duplicate ACKs and then abnormally fast retransmission [50]. The n-Reordering metric is defined as follows. A received packet i ($n < i \leq l$) with source sequence number $s[i]$, is n-reordered, if and only if $s[j] > s[i]$ for all j with $i - n \leq j < i$ (Fig. 8). Detecting instances of n-reordering with n greater than 3 is useful for detecting unnecessary retransmissions.[12]

Measurement Methodologies and Tools

The reordering metric measurement according to [48] is implemented in owping/owampd [25] and QoSMet [26]. According to Morton *et al.* recommendations, these tools can measure along the forward and reverse path by sending streams of packets and by analyzing packets sequence numbers.

Alternative Methodology based on Selective Acknowledgments (SACKs) — To measure reordering, the authors in [50] calculate the number of SACKs [59] blocks required to cover out of order packets in its test stream, if the session were a TCP connection (a test consists of sending repeated ICMP echo packets to a remote end-system and evaluating the order of the ICMP echo reply packets). This technique requires eliminating losses by renumbering received packets to ensure that all the holes in the SACK scoreboard are due to reordering and not due to packet loss. This metric then has a minimal value for in-order data and the property to match an intuitive assessment of the amount of scrambling in a given data set. The use of ICMP echo request/reply packets does not distinguish any difference between reordering in the forward or the reverse path.

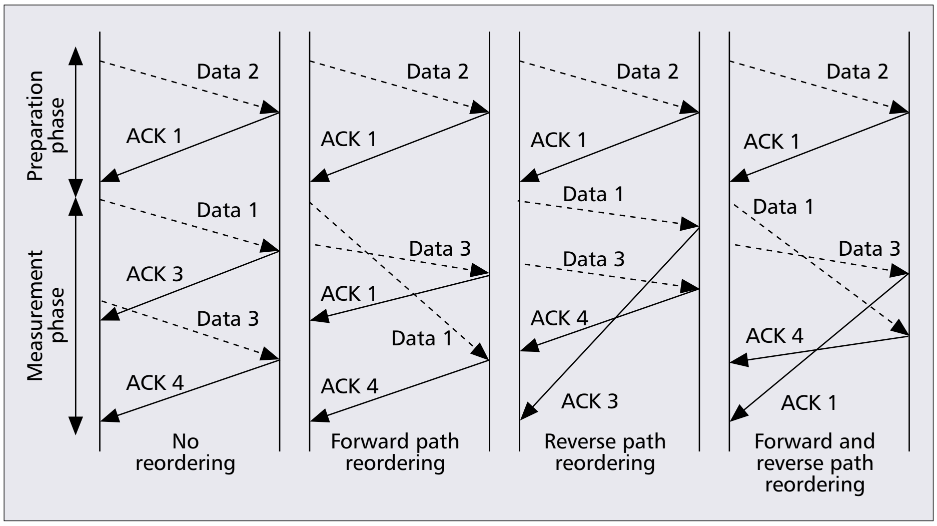
Alternative Methodologies based on Pairs of Packets – In [49] packet reordering is measured as the number of exchanges between pairs of test packets. For measuring this, the authors have proposed a series of techniques that use the behavior of the TCP and IP protocols to estimate one-way reordering:

• The first technique is the "single connection test." It consists of using a single TCP connection to the remote end-system. It takes advantage of the cumulative acknowledgment scheme of TCP: by sending a pair of packets (with label "data 1" and "data 3" in Fig. 9 and analyzing the corresponding received ACKs, the source is able to detect if packets are reordered in the forward and/or in the reverse path. One of the limitations to this approach is that two samples must be generated and delivered from the remote host. This may not occur for two reasons: first, if one of the data packets or one of the ACKs is lost, or second, if the receiver chooses to send a single ACK for the two packets due to the delayed acknowledgment algorithm. (In case of in-order delivery at the receiver, see earlier section.)

• To cope with the previous limitations, a second technique, called the "dual connection test," is introduced (Fig. 10). It consists of analyzing the identification field of the IP packet header (IP packet identifier (IPID)). The IPID field has been designed for fragmentation purposes: the same IPID value is copied into each fragment of a datagram. The receiver uses it to identify fragments belonging to the same datagram. To work, this technique implies that the IPID value is unique for each datagram. The common IPID implementations use an increasing function to generate IPID values. The IPID field can then be used to determine the order in which

[11] *Note that the scope of this draft is to define a metric that is orthogonal to the loss metric. In this context, the different approaches presented in the two drafts are complementary.*

[12] *The authors in [48] clearly note that "the definition of n-reordering cannot predict the exact number of packets unnecessarily retransmitted by a TCP sender (). The definition is less complicated than a TCP implementation where both time and position influence the sender's behavior."*



■ **Figure 9.** *Single connection test. Note that sample packets are 1 byte TCP data packets in this example.*

packets are sent. The dual connection test uses this property to measure reordering across a pair of TCP connections. Two packets are sent to the remote host (one on each connection) with sequence numbers greater than that expected to force the receiver to send ACKs immediately. (To avoid the delayed acknowledgment algorithm, see earlier section.) Knowing the order in which packets are sent and looking at the IPID values of the ACKs, it is possible to detect if packets are reordered in the forward and/or reverse path (Fig. 10). (The association between sample packets and their ACKs is easily performed by using the source and destination port numbers. It is for this reason that two connections are used.) However, this technique has some limitations: first, all OSs do not generate IPID using an increasing function, and second, using two connections can be problematic when connections are aliased through a transparent load balancing device, since each connection can be assigned to a separate host.

•To address the previous limitations (delayed acknowledgment algorithm, non-increasing IPID, load balancers), a third technique, the "SYN test," is introduced. Load balancers always forward packets of a given TCP connection to the same host. Then, the "SYN test" uses a single TCP connection to take advantage of TCP's three way handshake, and consists of sending pairs of packets that are TCP SYNs differing only in their starting sequence number. As shown in Fig. 11, the sender will receive different sequences of packets according to the presence (or absence) of reordering in the forward path and/or reverse path. These techniques are implemented in Sting [47].

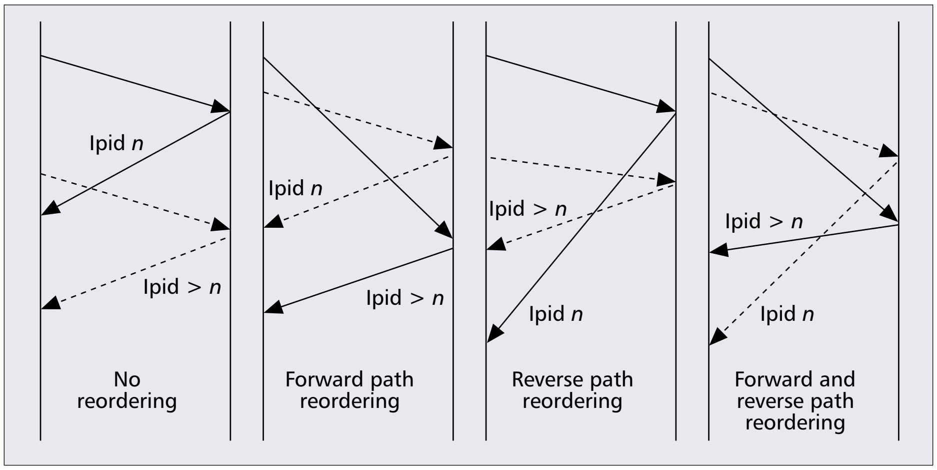
Route

Definition

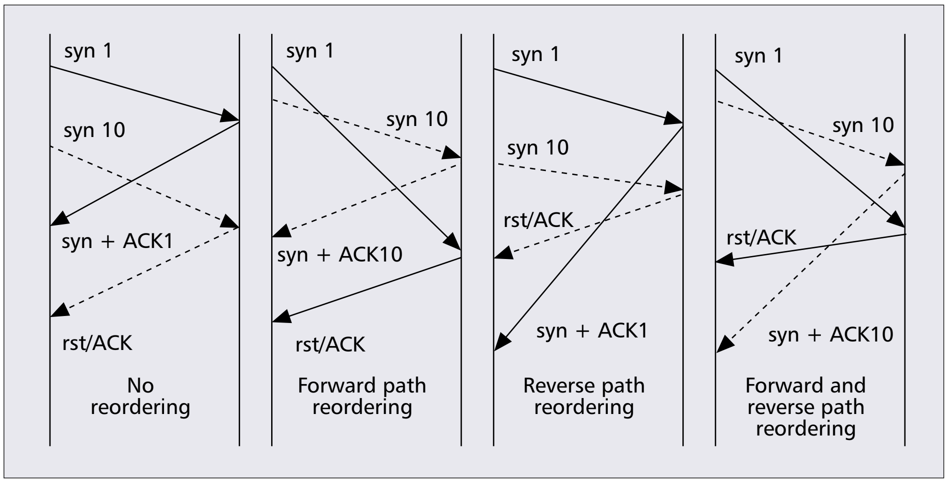
The route is an ordered sequence of nodes that represent a path from a source to a destination crossed by the exchanged traffic. Each node is identified by its IP address. A complete route from a source to a destination may consist of a single IP address or multiple IP addresses.

Measurement Methodology and Tool

The route from source to destination can be determined by taking advantage of the "time to live" (TTL) field of IP packets (the TTL field was meant to prevent packets from circulating around routing loops). Routers have to decrease it by one unit when processing the packet. If the router decreases the TTL field to zero, it discards the packet. In this case, the router sends back an ICMP "time exceeded" message informing the sender that the packet was dropped. To trace the route to the destination, the source first sends UDP packets to the destination with TTL fixed to 1. The first path router then discards these packets and sends back ICMP messages to



■ **Figure 10.** *Dual connection test.*



■ **Figure 11.** *SYN test.*

the source to inform it of the drops. ICMP message headers include the router address, which allows the source to identify the first hop. Next, the source sends packets with TTL fixed to 2, which identify the second hop. It proceeds in this fashion until it receives a reply from the destination (Fig. 12).

This methodology has been proposed and implemented by Van Jacobson in "traceroute" [29]. It is a non-cooperative tool. Note that experiments performed by Paxson [1] have shown many traceroute limitations. The principle ones are:

- The route can change between successive probe packets. There is no guarantee that probes of different hops will take the same route as previous probes.
- Traceroute assumes that intermediate routers send back ICMP messages. As explained before in the article, some routers could not generate ICMP messages. In this case, traceroute will fail to give a complete route.
- The goal of traceroute is to determine the route at the IP layer and not to provide layer 2 hops (ATM switches, for example). Traceroute is not designed to elicit lower-layer hops. The user should thus be aware that successive IP routers can be connected through different link-layer technologies (ATM, Frame Relay, etc.).

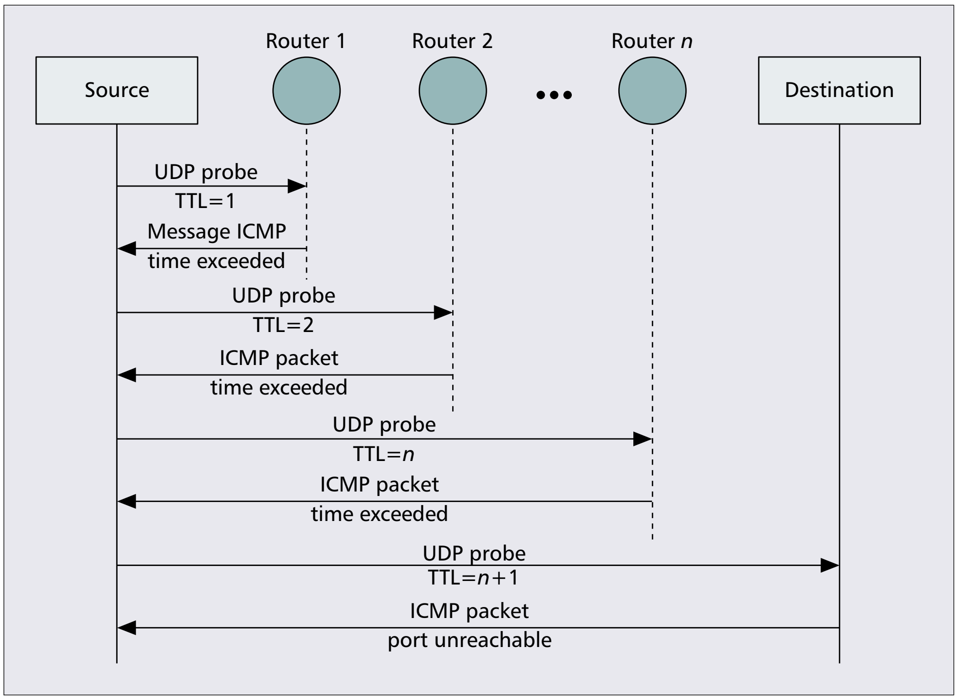
BANDWIDTH

More precisely, four bandwidth measurements can be performed:

- Capacity/raw bandwidth of a link
- End-to-end capacity of a path
- Available bandwidth of a link
- Available bandwidth of a path

To these four parameters, we add the bulk-transfer-capacity (BTC), which is a pertinent bandwidth-related metric in TCP/IP networks. In the following sections we will define these parameters and present the measurement techniques and tools for each parameter.[13]

[13] *An overview of bandwidth measurement techniques and methodologies is given in [20].*



■ **Figure 12.** *Route determination.*

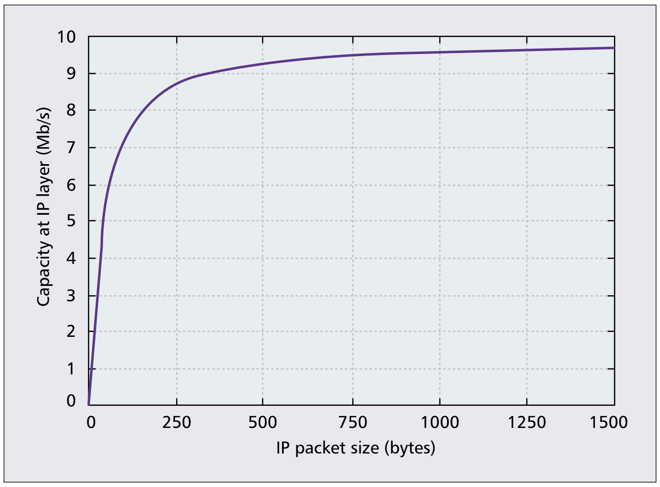


Figure 13. *Link capacity delivered to IP packets as a function of packet size.*

Definitions (based on Definitions from [20] and [60])

P is a network path from source to destination. P is a sequence of H store-and-forward links. We assume that P is fixed and unique (no routing changes or multipath forwarding occur during the measurements).

- The (per-hop) capacity/raw bandwidth of a link defines the maximum rate at which packets can be transmitted by the link. The capacity of the link i is noted C_i.
- C is the end-to-end capacity of the path:

$$C = \min_{i=1\ldots H} C_i \tag{1}$$

- The *narrow link* is the link with the smallest capacity along the path. The narrow link is the link j such as:

$$C_j = \min_{i=1\ldots H} C_i \tag{2}$$

- We assume that the link i is transmitting $C_i u_i$ bits during a time interval T. u_i is the utilization rate of the link i during T, with $0 \le u_i \le 1$. The available bandwidth A_i of the link i is:

$$A_i = C_i(1 - u_i) \tag{3}$$

The available bandwidth of a link depends on the traffic load at that link.

- The available bandwidth A of the path P during the time interval T is the minimum of the available bandwidth of all links that compose P:

$$A = \min_{i=1\ldots H} \{C_i(1-u_i)\} = \min_{i=1\ldots H} A_i \tag{4}$$

- The *tight link* is the link with the minimum available bandwidth along the path. The tight link is the link j such that:

$$A_j = \min_{i=1\ldots H} A_i \tag{5}$$

- The BTC represents the achievable throughput by a TCP connection on the end-to-end path.

Note: The term *bottleneck link* has been used in the past to refer to both the *tight link* as well as the *narrow link*.

Encapsulation Overhead

Bandwidth (especially capacity) is often known at the data link layer. Prasad *et al.* have shown that the corresponding capacity at the IP layer depends on the size of the IP packet relative to the layer 2 overhead [20]. Indeed, the layer 2 header is the same size regardless of the IP packet.

The transmission time T_3 of an IP packet of size L_3 is:

$$T_3 = \frac{L_3 + H_2}{C_2} \tag{6}$$

with H_2 the size of the layer 2 header and C_2 the layer 2 capacity.

The corresponding bandwidth at the IP layer is:

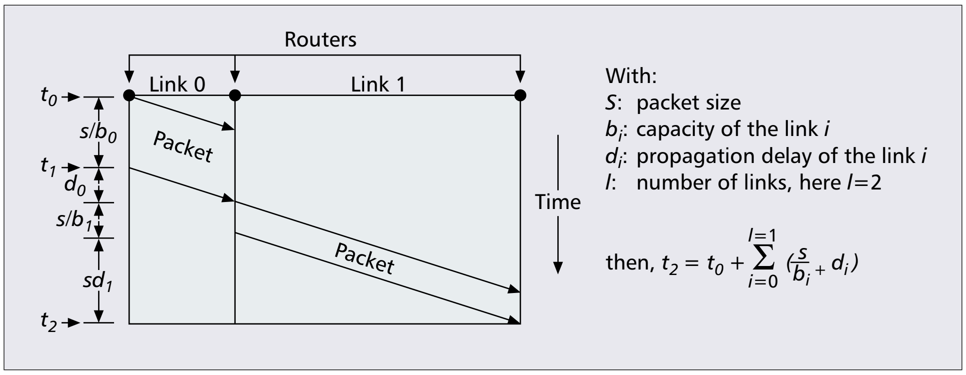
$$C_3 = \frac{L_3}{T_3} = \frac{L_3}{\dfrac{L_3 + H_2}{C_2}} = \frac{C_2}{1 + \dfrac{H_2}{L_3}} \tag{7}$$

Equation 7 shows that bandwidth at the IP layer depends on IP packet size and the layer 2 header size.

Figure 13 shows the IP capacity as a function of packet size for the 10BaseT Ethernet [20]. It has been obtained by using equation 7 with C_2 = 10 Mb/s, H_2 = 8 + 18 + 12 = 38 bytes = 304 bits (preamble, header, inter-frame gap). In this example, the IP capacity is 7.24 Mb/s for 100 byte packets and 9.75 Mb/s for 1500 byte packets.

Per-hop Capacity

In this section we present the main models to measure per-hop capacity and their implementations: the one-packet model and the multi-packet model. The reader can find extensions of the one-packet model in [39] (a case of asymmetric

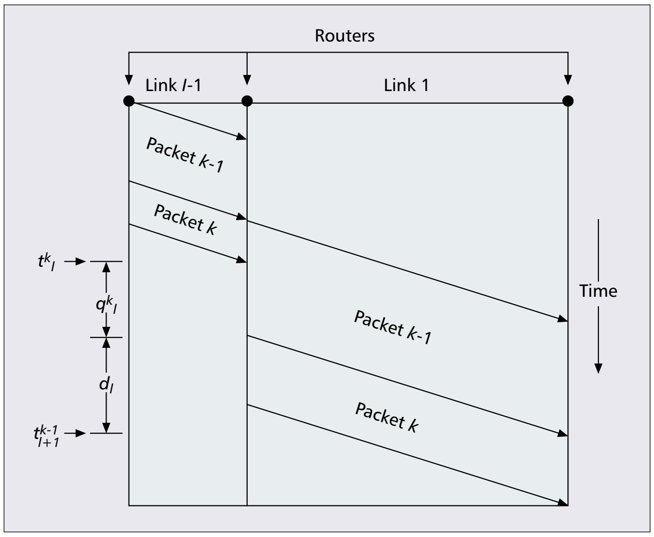


Routers
Link 0
Link 1
t_0
s/b_0
t_1
d_0
s/b_1
sd_1
t_2
Packet
Packet
Time
With:
S: packet size
b_i: capacity of the link i
d_i: propagation delay of the link i
l: number of links, here l=2
then, $t_2 = t_0 + \sum_{i=0}^{l=1} (\frac{s}{b_i} + d_i)$

Figure 14. *Per-hop capacity measurement.*

```
metro1:/# pathchar -m 1500 193.54.10.10
pathchar to 193.54.10.10 (193.54.10.10)
 doing 32 probes at each of 45 sizes (64 to 1500 by 32)
 0 localhost
 |   8.2 Mb/s,   158 us (1.78 ms)
 1 gw1.sciences.stannet.ciril.fr (193.50.39.254)
 |    17 Mb/s,    42 us (2.57 ms),   15% dropped
2 esstin.u-nancy.fr (193.54.10.10)
2 hops, rtt 401 us (2.57 ms), bottleneck 8.2 Mb/s, pipe
6680 bytes
```

■ **Figure 15.** *Pathchar output.*



■ **Figure 16.** *Multi-packet model.*

links) and in [61] ("packet quartets"). We do not present these complex extensions and improvements in this article, given its tutorial nature.

One-Packet Methodology — The one-packet model, also known as the Variable Packet Size (VPS) model, was introduced by Jacobson [62] and Bellovin [63]. By one-packet model it is inferred that the model does not account for intra-flow queuing delay. The model assumes that delay is linear with respect to packet size, and then proposes to express the packet delay as a function of the packet size and the capacity of each crossed link. Figure 14 shows the example of a packet that crosses two links. The delay $t_l - t_0$ to reach node l is the sum of the transmission delays

$$\sum_{i=0}^{l-1} \frac{s}{b_i}$$

and propagation delays

$$\sum_{i=0}^{l-1} d_i$$

of all the links along the path (using the notation of Fig. 14):

$$T_l = t_l - t_0 = \sum_{i=0}^{l-1}\left(\frac{s}{b_i} + d_i\right) \tag{8}$$

The d_i's are constant because they are propagation delays, then:

$$T_l = s\sum_{i=0}^{l-1}\left(\frac{1}{b_i}\right) + K \tag{9}$$

Formula 9 is the equation of a line with a slope k_l:

$$k_l = \sum_{i=0}^{l-1}\left(\frac{1}{b_i}\right) \tag{10}$$

One can then determine the k_l's by measuring the delays from node 0 (the source) to the node l of packets of different sizes s, and using linear regressions. When the k_l's are known, the capacity of each link is calculated using the following equation:

$$b_{l-1} = \frac{1}{k_l - k_{l-1}} \tag{11}$$

To summarize, the method then consists of:
- Determining packet delays of different sizes from a source to each router that is on the path between the source and the destination
- Calculating the k_l's using linear regressions
- Calculating the capacities using Eq. 11

One-Packet Implementations — Pathchar [62], Bing [65], clink [66], and pchar [64] implement the one-packet technique. All these tools are non-cooperative tools. A screenshot of pathchar output is presented in Fig. 15.

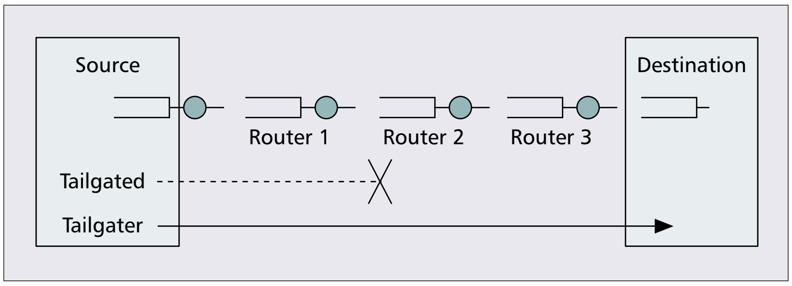
In all these implementations, the source reaches each router by sending probe packets and exploiting the TTL field of IP packets and ICMP "time exceeded" messages as traceroute. This technique allows the sender to measure the RTT to each router. This mechanism is repeated for different packet sizes. Assuming that links and paths are symmetric, Eqs. 9 and 11 are now:

$$RTT_l = s\sum_{i=0}^{l-1}\left(\frac{2}{b_i}\right) + K, \quad b_{l-1} = \frac{2}{k_l - k_{l-1}} \tag{12}$$

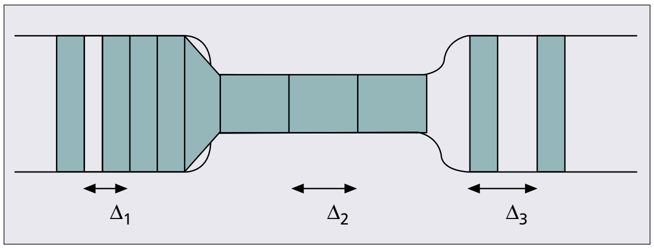
Notes:

The one-packet model uses packet transmission time (or serialization time) to perform calculations. The authors in [21] have shown that store-and-forward layer 2 devices (Ethernet switches, for example) can cause significant underestimation of capacity. Indeed, these types of devices introduce additional serialization delay.[14]

[14] The authors in [21] have also shown that cut-through layer 2 devices (such as ATM switches) introduce a constant delay term, so they do not affect the accuracy of one-packet tools.



■ **Figure 17.** *Tailgating technique.*



■ **Figure 18.** *Packet dispersion (from the fluid analogy in [68]). Here,* $\Delta_2 = \max(\Delta_1, L/C_2) = L/C_2$, $\Delta_3 = \max(\Delta_2, L/C_3) = \Delta_2$.

Lower-layer fragmentation effects can affect one-packet tool precision by adding lower-layer headers [61] (minimum frame size in Ethernet networks, for example).

The model assumes that packets observe no queuing. This assumption is not realistic: in practice, queuing increases delays and results in distorting calculations. Additionally, using RTTs doubles the possibility that queuing will affect measurements.

- To compensate, the authors in [62] and [63] base their calculations on the minimum of several observed delays of a particular packet size. Many packets may then be required to filter out queuing effect and perform the regression with confidence [67].
- Downey *et al.* [66] reduce measurement traffic by avoiding sending packets once they detect the convergence of a bandwidth estimate (using statistical methods).
- In [61], the bandwidth calculation is based on delay variation measurement instead of delay measurement. It limits the number of probes sent to the network.
- To limit the number of packets it sends, the multi-packet model has been introduced in [67]. It is derived from the one-packet model. We present it in the following section.

Multi-Packet Methodology — As explained before, the one-packet model is costly in terms of the traffic that must be sent. To compensate, Lai *et al.* have introduced the multi-packet model (or tailgating model) in [67]. This model is derived from the one-packet model and focuses on intra-flow queuing delays.

Figure 16 shows the example of two consecutive packets $k-1$ and k that cross two links (exponents indicate packet numbers). It also shows that the arrival time of packet k at link l is equal to its transmission time t_k^0 plus the sum of all the previous link latencies d_i, transmission delays (s_k/b_i), and queuing delays due to other packets in the same flow q_k^i. Using Eq. 8, t_k^l is then:

$$t_l^k = t_0^k + \sum_{i=0}^{l-1}\left(\frac{s^k}{b_i} + d_i + q_i^k\right) \tag{13}$$

q_k^i is modeled using the following equation:

$$q_l^k = \max\left(0, t_{l+1}^{k-1} - d_l - t_l^k\right) \tag{14}$$

The multi-packet model equation is obtained by combining Eqs. 13 and 14:

$$t_l^k = t_0^k + \sum_{i=0}^{l-1}\left(\frac{s^k}{b_i} + d_i + \max\left(0, t_{l+1}^{k-1} - d_l - t_l^k\right)\right) \tag{15}$$

Assuming that it is possible to send one packet with no queuing and a second packet that queues behind the first packet at a specific link l_q, the delay can be split into the time to travel to l_q, the time spent at l_q, and the time spent after l_q:

$$\begin{aligned} t_l^k &= \left[t_0^k + \sum_{i=0}^{l_q-1}\left(\frac{s^k}{b_i} + d_i\right)\right] + \left[\frac{s^k}{b_{l_q}} + t_{l_q+1}^{k-1} - t_{l_q}^k\right] + \left[\sum_{i=l_q+1}^{l-1}\left(\frac{s^k}{b_i} + d_i\right)\right] \\ &= t_{l_q}^k + \frac{s^k}{b_{l_q}} + t_{l_q+1}^{k-1} - t_{l_q}^k + \sum_{i=l_q+1}^{l-1}\left(\frac{s^k}{b_i} + d_i\right) \\ &= \frac{s^k}{b_{l_q}} + t_{l_q+1}^{k-1} + \sum_{i=l_q+1}^{l-1}\left(\frac{s^k}{b_i} + d_i\right) \end{aligned} \tag{16}$$

By including the assumption that the first packet experiences no queuing and by simplifying:

$$\begin{aligned} t_l^k &= \frac{s^k}{b_{l_q}} + \sum_{i=0}^{l_q}\left(\frac{s^{k-1}}{b_i} + d_i\right) + t_0^{k-1} + \sum_{i=l_q+1}^{l-1}\left(\frac{s^k}{b_i} + d_i\right) \\ &= \frac{s^{k-1}}{b_{l_q}} + \sum_{i=0}^{l_q-1}\left(\frac{s^{k-1}}{b_i}\right) + \sum_{i=l_q}^{l-1}\left(\frac{s^k}{b_i}\right) + t_0^{k-1} + \sum_{i=0}^{l-1}(d_i) \end{aligned} \tag{17}$$

For more compact notation, we define β and δ such that:

$$\beta^n = \sum_{i=0}^{n} d_i, \quad \delta^n \sum_{i=0}^{n} b_i \tag{18}$$

Using these definitions:

$$t_l^k = \frac{s^{k-1}}{b_{l_q}} + s^{k-1}\delta^{l_q-1} + s^k(\delta^{l-1} - \delta^{l_q-1}) + t_0^{k-1} + \beta^{l-1} \tag{19}$$

From the previous equation, we can derive a system of equations for the capacities of each of the links along a path. There are l equations for l links. The equation for the link l_q is:

$$b_{l_q} = \frac{s^{k-1}}{t_l^k + \dfrac{s^k - s^{k-1}}{\beta^{l_q-1}} - \dfrac{s^k}{\beta^{l-1}} - t_0^{k-1} - \delta^{l-1}} \tag{20}$$

β^{l-1} and δ^{l-1} are calculated as in the one-packet model in a separate phase. As explained before, this phase is very costly in terms of the traffic that has to be sent because it must send packets of many sizes and many packets per size. However, the multi-packet technique does it once for the entire path, while the one-packet technique does this once for every link.

The method then consists of using the recursive structure of equation 20 to estimate the b_i's:

1 Initialize $l_q = 1$.
2 Determine β^{l_q-1} i.e., measure the delay from the source to the node l_q.
3 Calculate the capacity b_{l_q}.
4 Increment l_q and go to step 2.

The main advantage of this method is that the delays to each router need only to be measured for one packet size.

Multi-Packet Implementation — The multi-packet model is implemented in Nettimer [67]. It is non-cooperative.[15] In practice, the largest possible non-fragmented packet with a particular TTL field is sent. It is immediately followed by the smallest possible packet. The smaller packet almost always has a lower transmission delay than the larger packet's transmis-

[15] *nettimer can also work cooperatively as it implements other measurement techniques.*

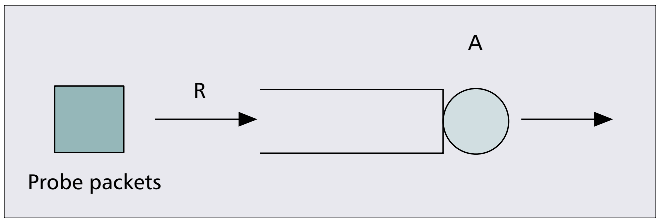


Figure 19. *The probe rate model.*

sion delay on the next link. Then the smaller packet (the tailgater) queues continuously after the larger packet (the tailgated). The tailgated packet is dropped at a particular router due to its TTL. The tailgater can then continue without queuing to the destination (Fig. 17). To cope with ICMP limitation, Nettimer introduces a new idea to measure the RTT to the destination by using the TCP FIN/RST mechanism: the tailgater is a TCP FIN message to force the receiver to send back a TCP RST message.

Note:
- The equations of the model are transformed to use RTTs instead of one-way delays, assuming capacities are symmetric.

End-to-End Capacity

End-to-end capacity can be determined using per-hop capacity tools and by taking the minimum value of the capacities of the links that compose the path. However, the packet-pair dispersion method can directly measure it. In this section we present the packet-pair dispersion methodology, its implementations, and its limitations.

Packet-Pair Dispersion Methodology — This method is meant to show that two back-to-back packets (a packet-pair) will be spread out in time when they arrive at the narrow link (the "packet dispersion" phenomenon shown by Jacobson in [68], Fig. 18). When two packets of size L arrive with a time distance Δ_{in} at a link i, assuming that no cross traffic will be inserted between the packets, the output time distance is then:

$$\Delta_{out} = \max\left(\Delta_{in}, \frac{L}{C_i}\right) \tag{21}$$

At the receiver, packet dispersion is:

$$\Delta_R = \max_{i=1\ldots H}\left(\frac{L}{C_i}\right) \tag{22}$$

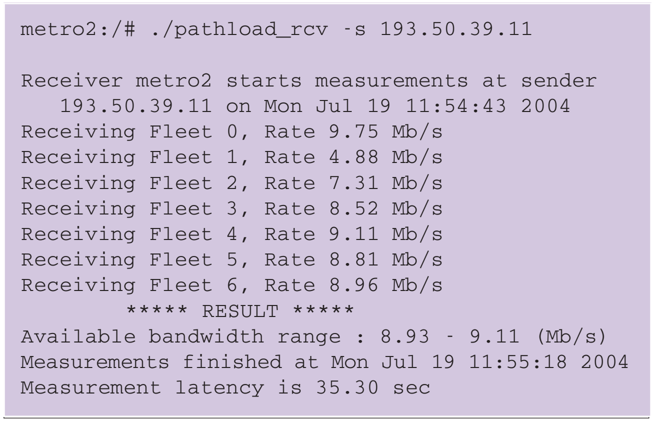
Δ_R is measured by the receiver by time stamping the probe packets upon reception. Then, the end-to-end capacity can be calculated by the receiver as:

$$C = \frac{L}{\Delta_R} \tag{23}$$

Packet-Pair Dispersion Implementations — bprobe [69], sprobe [70], pathrate [71, 72], and nettimer [73] implement the packet-pair dispersion technique:
- To work non-cooperatively, bprobe assumes that packet dispersion remains constant when packets come back to the sender. In this case, measurement is done with ICMP Echo-Reply packets and the end-to-end capacity is measured by the sender. It assumes paths and links to be symmetric.
- sprobe is able to measure the end-to-end capacity bidirectionally (upstream and downstream) by exploiting TCP protocol to generate packet-pairs (SYN and RST messages, similar to the sting method). Then it is non-cooperative.
- nettimer works cooperatively; the measurement is performed passively by analyzing traffic traces (off-line analysis).

Packet-Pair Dispersion Limitations — Packet loss, reordering, and the use of multichannel links [74] affect performances of the packet-pair technique. Additionally, performances depend on the size of probing packets, the presence of cross traffic, and the minimal dispersion that the end-terminal is able to measure. The main studies on this have been performed in [72] and are summarized in the following three paragraphs.

Effect of Cross Traffic — Cross traffic affects measurement quality. First, the packet-pair dispersion technique assumes that no cross traffic is transmitted between probing packets. However, it can happen. In this case, packet dispersion increases and leads to underestimating the end-to-end capacity. Second, the first packet of the pair can be delayed more than the second at a link that follows the narrow link due to the presence of cross traffic in the queue. In this case, the dispersion decreases and the end-to-end capacity is overestimated. The authors in [72] have daemonstrated that these two effects cause the distribution of packet-pair dispersion to be multimodal: some modes correspond to underestimates of the capacity (Sub-Capacity Dispersion Range (SCDR)), a mode corresponds to the capacity (Capacity Mode (CM)), and others are related to overestimates (Post-Narrow Capacity Mode (PNCM)). Then to take into account the effect of cross traffic, the measurement tools need to use filtering techniques:
- pathrate proceeds by sending packet-pairs and packet trains. The packet-pair probing phase will obtain a "complete" distribution that includes all kinds of modes (SCDR, CM, and PNCM) while packet train probing determines SCDR modes. Based on a heuristic rule, pathrate selects which mode is the capacity mode (the minimum mode of the first distribution that is greater than the modes of the second).
- bprobe uses union/intersection statistical filtering to select the capacity mode from different sets of measurements.
- To eliminate measurement samples related to SCDR and PNCM, nettimer makes the following observations. First, packets that arrive with a higher bandwidth than they were sent with correspond to the PNCM case. Those



```
metro2:/# ./pathload_rcv -s 193.50.39.11

Receiver metro2 starts measurements at sender
   193.50.39.11 on Mon Jul 19 11:54:43 2004
Receiving Fleet 0, Rate 9.75 Mb/s
Receiving Fleet 1, Rate 4.88 Mb/s
Receiving Fleet 2, Rate 7.31 Mb/s
Receiving Fleet 3, Rate 8.52 Mb/s
Receiving Fleet 4, Rate 9.11 Mb/s
Receiving Fleet 5, Rate 8.81 Mb/s
Receiving Fleet 6, Rate 8.96 Mb/s
        ***** RESULT *****
Available bandwidth range : 8.93 - 9.11 (Mb/s)
Measurements finished at Mon Jul 19 11:55:18 2004
Measurement latency is 35.30 sec
```

Figure 20. *pathload output [71].*

samples are easily discarded. Second, samples influenced by cross traffic (related to SCDR and PNCM cases) do not tend to correlate with each other (assuming that cross traffic has random packet sizes and arrives randomly at the links along the path). nettimer then uses a kernel density estimation method to determine which samples are more significant.

Probing Packet Size — The size of probing packets can affect measuring. The consequence of a large packet size is to increase the time interval in which a cross traffic packet can interfere with the probing packets. At the same time, a packet size that is too small is not ideal either. As L decreases, the dispersion decreases and measurements are more sensitive to the case in which the first packet has a longer delay than the second. The empirical conclusion of [72] from internet experiments is that a packet size of 800 bytes seems to be ideal. Note, however, that in bprobe the authors argue that the optimal size for probe packets is the maximum transmission unit (MTU) and that nettimer cannot choose the probe packet size because of passive measurements.

Minimal Measurable Dispersion — The range of measurable value is limited by the minimal dispersion that the end-terminal is able to measure. This latter criterion depends on time-related errors and uncertainties when receiving a packet. The authors in [72] have shown that the minimal measurable dispersion with SUN and PC under FreeBSD and Solaris is 30μs to 40μs. It can measure a maximal bandwidth of 160 Mb/s. Those values have been obtained for 800-byte probe packets, which correspond to an optimal packet size according to the authors. This constraint is mentioned in nettimer and pipechar, though not in bprobe and sprobe.

Available Bandwidth (Links/Paths)

The techniques to measure the available bandwidth can be classified into three categories: Packet Train Dispersion (PTD), Probe Gap Model (PGM), and Probe Rate Model (PRM). In addition, we present Multi Router Traffic Grapher (MRTG), which is usually used as a reference for testing available bandwidth measurement tools.

Packet Train Dispersion Methodology — The PTD is the simplest technique proposed to measure available bandwidth. It was introduced in [69]. It consists of sending $N > 2$ back-to-back packets of size L (a packet train of length N) to the receiver. The rate at which the N packets are sent must be larger than the available bandwidth of the tight link. When arriving at the receiver, one can measure the packet train dispersion $\Delta(N)$, i.e., the amount of time between the receipt of the last bit of the first packet and the last bit of the last packet. For a train of length N, the receiver measures $\Delta(N)$ and estimates the available bandwidth A as follows:

$$A = \frac{(N-1)L}{\Delta(N)} \tag{24}$$

Packet Train Dispersion Implementations — cprobe implements the PTD model [69].[16] cprobe is non-cooperative: it sends a short stream of ICMP echo packets to the target host and assumes that it will respond by sending back ICMP Echo-Reply packets. Then the sender measures the dispersion of the ICMP Reply packets. Note that in this case, characteristics of the reverse path and cross traffic may affect the results.

[16] *The tool pipechar [75] is often classified as a PTD tool [21, 76]. However, the article describing it does not clearly explain its measurement methodology.*

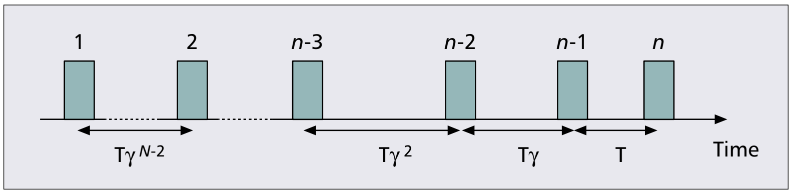


Figure 21. *Chirp probe train.*

Packet Train Dispersion Limitations — Dovrolis *et al.* have shown in [72] that formula 24 gives a metric called Asymptotic Dispersion Range (ADR). The ADR metric is related to the utilization of all links in the path. An absence of cross traffic in the path means that ADR is equal to the end-to-end capacity as with the packet-pair technique. In any other case, ADR is not related to available bandwidth [20] and is difficult to analyze [20, 72].

Probe Rate Model — The PRM model is based on the concept of self-induced congestion. Assuming that FIFO queuing occurs at all routers along the path, that cross traffic follows a fluid model, and that changes of cross traffic rate are slow [76], we can represent the network by a queue with a service rate equal to the available bandwidth A (Fig. 19).

If a source sends probes to a destination through the queue at a rate R less than A, probes will experience similar delays. On the other hand, if R is greater than A, probes will queue in the network and experience increasing delays. The PRM model is thus based on the observation that the delays of successive probing packets increase when the probing rate exceeds the available bandwidth in the path.

The PRM model consists in probing the network at different rates and detecting (at the destination) the point when delays start to increase. At this point, probing rate is equal to the available bandwidth.

Probe Rate Model Implementations — Pathload [60] and PathChirp [77] implement the PRM model. These tools are cooperative (Fig. 20).

Pathload introduces a technique based on Self Loading Periodic Stream (SLoPS): the algorithm consists of sending a stream of packets to the receiver. The receiver measures the delay of each received packet and analyzes its variation. If the delay is constant i.e. the stream rate is expected to be less than the available bandwidth, another stream is sent to the receiver at a greater rate. If the delay increases, i.e. the stream rate is expected to be greater than the available bandwidth, another stream is then sent to the receiver at a rate between the two precedent values. This technique is repeated and the algorithm converges by dichotomy to the available bandwidth value. Due to its iterative nature, this algorithm can have long convergence times [60, 78].

PathChirp proposes sending an exponentially spaced “chirp” probing train (Fig. 21). The main advantage of this approach is to minimize the probing traffic load. Indeed, a single chirp is able to probe the network at different rates. Moreover, using a chirp of n packets allows pathChirp to exploit $n - 1$ packet spacings that would require $2n - 2$ packets using a packet-pair technique.

Probe Rate Model Extensions — The Trains of Packet-Pairs (TOPP) method mixes the PRM model and the packet-pair probing technique [78]. It introduces a new metric, the *propor-*

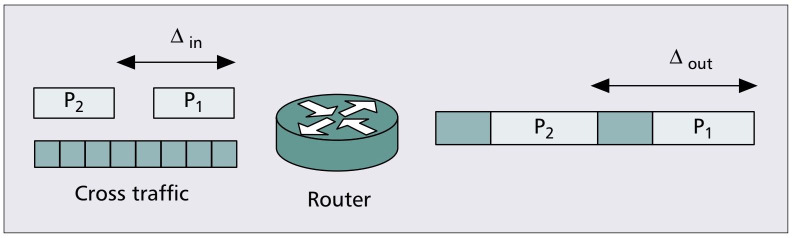


Figure 22. *Probe gap model.*

tional share f, and uses it to evaluate available bandwidth. Proportional share is defined as the share of the link bandwidth that a new connection (with an *offered load*, or rate o) will obtain on that link:

$$f = \frac{o}{C_i u_i + o} C_i \tag{25}$$

This definition makes the strong assumption that routers share their bandwidth using proportional stateless scheduling policy (First Come First Serve (FCFS), for example) and use a random dropping policy of packets at buffer overflow. At the node i, given a sender rate of $o_0 = o$, the connection will obtain the following proportional share bandwidth:

$$o_i = \begin{cases} o_{i-1} & \text{if } o_{i-1} < A_i \\ \dfrac{o_{i-1}}{C_i u_i + o_{i-1}} C_i & \text{if } o_{i-1} \geq A_i \end{cases} \tag{26}$$

The rate at the receiver will be $f = o_n$ and will represent the bandwidth experienced by the probe packets. The measurement method consists of sending a set of n separated pairs of equally sized packets L (a train of packet pairs) starting at some rate o_{min}. The probing rate o is then increased and another train is sent. This goes on until o reaches some rate o_{max}. According to Eq. 26, the measured rate at the destination will be $f = o_n = o$ until o reaches the available bandwidth A. Thereafter, $f = o_n$ will correspond to the proportional share bandwidth at the tight link (link j such as $A_j = A$). It is then possible to detect the rate o which is equal to the available bandwidth A. This technique implies that the destination measures the experienced bandwidth f. It is estimated as:

$$\tilde{f} = \frac{L}{\Delta T} \text{ with } \Delta T \text{ the time distance between packets at the destination} \tag{27}$$

Notes:
- TOPP has only been simulated using ns-2; no implementation is available.
- TOPP is not limited to the estimation of the path available bandwidth. It is also able to measure available bandwidth and capacity of every link on the path. However, these latter measurements are prone to error because TOPP assumes that links are in Smallest Surplus First order (SSF), which means that for every $i < j \leq H$, $A_i < A_j$.

Probe Gap Model — The PGM model consists in capturing the relationship between the dispersion of a packet-pair and the rate of cross traffic at the bottleneck link of a path (Fig. 22). The PGM makes the same assumptions that the PRM makes, and also assumes that the bottleneck link is both the tight link and the narrow link.

The authors in [79] have shown that the packet-pair dispersion increases linearly with the cross traffic rate if the queue of the bottleneck router does not become empty after the first packet of the pair leaves the router and before the second packet arrives at the router (Joint Queuing Region (JQR) condition):

$$\Delta_{out} = \frac{L + C_T \Delta_{in}}{C} \tag{28}$$

where Δ_{in} is the initial gap, Δ_{out} the output gap, L the size of the probe packets, (L/C) the time to process the first packet, and $(C_T\Delta_{in}/C)$ the time to process the cross traffic that arrives between the two probe packets. The rate of the cross traffic at the bottleneck link is thus:

$$C_T = \frac{C\Delta_{out} - L}{\Delta_{in}} \tag{29}$$

Then, assuming that the end-to-end capacity C is known (or measured) and determining the cross traffic rate at the bottleneck, C_T can simply calculate the available bandwidth of the path $A = C - C_T$.

Probe Gap Model Implementations — The PGM method is implemented in Initial Gap Increasing (IGI) [79] and spruce [76]. These tools are cooperative. IGI uses packet trains, i.e. a longer sequence of evenly spaced packets to probe the network. In this case, packet-pairs that compose a train are not independent. If a packet-pair is composed of packets P_k and P_{k+1}, the next packet-pair will be composed of packets P_{k+1} and P_{k+2}. In other words, the Δ_{out}'s are correlated. To deal with this, IGI bases its calculations on Δ_{out} samples that are greater than Δ_{in}.

To ensure that it operates under the JQR condition and does not flood the bottleneck, IGI adjusts the time distance in based on the following observations:
- Δ_{in} should be smaller than (or equal to) the transmission time of a probe packet at the bottleneck to ensure that the queue of the bottleneck router does not become empty after the first packet of the pair leaves the router and before the second packet arrives at the router.
- Too small of a Δ_{in} will flood the bottleneck. In this case, probe packets and cross traffic will not properly interleave, resulting in underestimating the cross traffic rate.

Experiments carried out in [79] have shown that the optimal initial gap is obtained when the average output gap equals the initial gap. IGI starts by sending packet-pairs with a small Δ_{in} and increases it until the average output gap equals the initial gap.

Notes:
- To improve dispersion measurement precision, IGI uses kernel-level timestamping.[17]
- IGI starts its execution by measuring the end-to-end capacity (similar to nettimer).

To cope with packet-pair dependence, spruce sends a Poisson process of packet-pairs. Additionally, spruce adjusts the average inter-pair gap to ensure that the probe rate is a minimum of 240 Kb/s and five percent of the end-to-end capacity. To ensure that it operates under the JQR condition, spruce sets the time distance Δ_{in} to the transmission time of a 1500-byte data packet at the bottleneck.[18]

[17] *Timestamping operations are performed at the kernel level in the root version of IGI, and at the application layer in the normal version.*

[18] *The rate of the cross traffic at the bottleneck link is then*

$$C_T = \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}} C$$

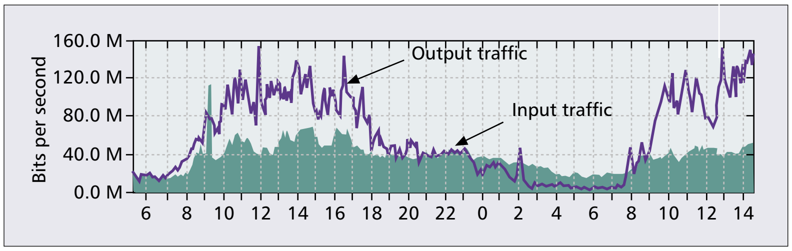


Figure 23. *MRTG output.*

Notes:
- To improve dispersion measurement precision, spruce uses kernel-level timestamping.
- The path capacity C is specified by the user when starting the spruce sender.

According to their authors, spruce and IGI give relatively accurate results even if the PGM assumptions are not met. Moreover, these techniques give results much faster than pathload. (Measurement times of spruce and IGI are similar, approximately 10 seconds.)

Multi Router Traffic Grapher — MRTG [80] is a program that uses SNMP to read the load of routers (traffic counters). Given the capacity of a link, it is then possible to calculate its available bandwidth (capacity minus its load). MRTG generates HTML pages containing images, which provide a representation of the traffic bitrate (Fig. 23). Due to its implementation, MRTG does not offer a measurement granularity smaller than five minutes. To determine the available bandwidth between two end-points, it is necessary to identify all the routers of the path and to have permission to read their MIBs (read-only access). The authors in [81] propose MRTG++, a patched version of MRTG that improves its granularity down to 10s. The authors present Available Bandwidth Estimator (ABEst), which is a tool that obtains information about link capacity and utilization within an autonomous system (AS) from routers. ABEst computes linear predictions of available bandwidth for each link. Prediction can limit the amount of data collected and reduce the signaling traffic and router load. One limitation of ABEst is that it requires routers to modify probing packets with SNMP data.

The measurement of available bandwidth is currently under discussion at the IPPM. A recent draft [82] proposes a technique based on a measurement of the TCP congestion window. This technique is not intrusive. Furthermore, the draft proposes a prediction algorithm to predict values of the available bandwidth. This prediction algorithm is the same as in ABEst.

Bulk-Transfer-Capacity

Definition — The BTC (Bulk-Transfer-Capacity or Bulk-Transport-Capacity) represents the achievable throughput by a TCP connection on the end-to-end path. The IPPM proposes a framework for defining BTC metrics [83] in which the following BTC definition is given: *Bulk-Transport-Capacity is a measure of a network's ability to transfer significant quantities of data with a single congestion-aware transport connection (e.g., TCP[19]). The intuitive definition of BTC is the expected long time average data rate of a single ideal TCP implementation over the path in question:*

$$BTC = \frac{data\_sent}{elapsed\_time} \tag{30}$$

Several TCP distributions are in use today (Tahoe, Reno, etc.) and implement in various way the congestion control algorithms published in [68] and [84]. A BTC measurement methodology should theoretically define which TCP implementation it works for.

Measurement Methodology and Tools — The best known tool that measures BTC is TReno [28] ("Traceroute Reno"), which emulates TCP (Reno) using UDP packets. Like pathchar and traceroute, it exploits the TTL field to force the target host to send back messages that it uses to simulate TCP ACKs. It is thus non-cooperative.

Iperf [85] is another tool to measure BTC. It was initially designed to provide TCP connection statistics for helping users to tune TCP window sizes. It works by establishing a TCP connection and trying to transmit data as quickly as possible. Iperf uses the TCP implementation of the operating system that the host is running, and it is cooperative.

Intrusiveness/Invasiveness

A very common question in measurement is the skew of the real value introduced by the measurement technique itself. This is especially true in the case of active measurements done in running systems. The property of a tool related to the amount of traffic it puts in the network is called intrusiveness (or invasiveness). Intrusiveness depends on the level of traffic generated by the tool. This traffic depends on the tool used and particularly on the measured parameter: tools measuring one-way delay or round-trip time generate few packets, so they are not intrusive except if the bandwidth is small. Measurement of packet loss and packet reordering cannot give objective results without sending a significant number of packets in real traffic conditions, so this is more intrusive. These types of active measurements are done during test periods. During the operating period the corresponding parameters can be obtained by passive measurements. Prasad *et al.* [20], who studied intrusiveness of bandwidth estimation tools, claimed that "*an active measurement tool is intrusive when its average probing traffic during the measurement process is significant compared to the available bandwidth in the path.*" They analyzed the impact of three types of tools: Packet Pair/Train Dispersion (PPTD), BTC, and one-packet tools. PPTD tools create short traffic bursts at high rates; however, these bursts last for only a few milliseconds and are separated by long time intervals. The average probing rate using pathload is typically less than 10 percent of the available bandwidth. It should be noted, however, that due to long range dependence (LRD) [86] there remain perturbations in the normal running after stopping the stress of the network. BTC measurement tools are classified as intrusive because they fulfill all the available bandwidth during the measurements, even if they use TCP, thus controlling congestion. One-packet tools seem less intrusive in long distance paths because there is only one packet per round-trip.

It is a fact that using active measurement tools results in intrusiveness problems in network traffic. Nowadays it is only defined qualitatively. Further research must be done in order to define a metric of intrusiveness and to quantify it. More generally, the key challenge in active measurement is to send a minimal amount of traffic to the network and to obtain the most accurate measurements possible.

[19] *Note that BTC metrics can be defined for other transport protocols than TCP.*

	Measurement tools	IETF metrics	Additional information
Per-hop capacity	pathchar [62], bing [65], clink [66], pchar [64], nettimer [67]		[39, 66]
Available bandwidth	cprobe [69], pipechar [75], pathload [60, 87], TOPP [78], pathChirp [77], spruce [76], IGI/PTR [79]	Internet draft [82]	
End-to-end capacity	bprobe [69], pathrate [71, 72], nettimer [73], sprobe [70]		[1]
Bulk transfer capacity	Iperf [85], TReno [28]	[83]	
One-way delay	owping/owampd [25], QoSMet [26]	RFC 2679 [33]	[1, 35, 39, 88]
One-way delay variation	Iperf [85], QoSMet [26]	RFC 3393 [40]	[42]
Round-trip time	ping	RFC 2681 [43]	[35]
Packet loss	ping, sting [47], owping/owampd [25], Iperf [85], QoSMet [26]	RFC 2680 [44], RFC 3357 [46]	[35]
Packet reordering	sting [47, 49], owping/owampd [25], QoSMet [26]	Internet draft [48], Internet draft [58]	[50]
Route	traceroute [29]		[1]

Table 1. *Summary of QoS parameters, related tools, IETF metrics, and additional information.*

Conclusion

Packet-switched communication network metrology is a relatively new field of research. However, several techniques and tools have already been proposed to perform measurements. The purpose of this article was to review application-oriented network metrology techniques and tools. The network parameters concerned were one-way delay, delay variation, round-trip time, packet loss, packet reordering, route. and bandwidth. A summary of their related measurement tools and IETF metrics is presented in Table 1. We have shown that the IPPM group of the IETF has proposed standard definitions for some parameters and is still working on the standardization of others. Nevertheless, existing measurement tools seldom conform to IPPM recommendations.

Table 2 summarizes measurement tool characteristics, and shows previous tool limitations. Each existing measurement tool is often dedicated to the measurement of a single parameter. Therefore, it is difficult to measure different parameters between two end-systems at the same time. The measurement tools almost always enable measurements to be carried out in only one way between two end-systems. Bidirectional measurements would require tools to be installed and parameterized, the execution to be started, and the results to be logged onto the two sites. In this context, we believe that work such as OWAMP or the metrology service proposed in [89] represent a great advance in proposing frameworks to "standardize" and integrate active measurement tools.

Additional research must be done in the network metrology community. Although this science has been around for a decade, it is now becoming a key issue due to the importance of verifying SLA. Consequently, improvements must be made. For the time being, no reference tool is available to validate or to compare existing tools. Existing tools are not capable of quantifying measurement uncertainties. Only when advancements are made on these tools will we then be able to quantify the errors. In this sense, guidelines must also be defined for measurement tool test and calibration. Finally, intrusiveness, i.e. perturbation induced by active measurement, is still an open issue. To date, no measurement traffic limitation rule has been established.

Acknowledgments

We would like to thank the anonymous referees for their many constructive comments.

References

[1] V. Paxson, "Measurements and Analysis of End-to-end Internet Dynamics," Ph.D. dissertation, Computer Science Division, University of California, Berkeley, and Information and Computing Sciences Division, Lawrence Berkeley National Laboratory, University of California, Apr. 1997.

[2] The Surveyor Website, Available: http://www.advanced.org/surveyor

[3] The National Internet Measurement Infrastructure (NIMI) Website, available at http://www.ncne.nlanr.net/nimi

[4] V. Paxson *et al.*, "An Architecture for Large-scale Internet Measurement," *IEEE Commun. Mag.*, vol. 36, no. 8, 1998, pp. 48–4.

[5] V. Paxson, A. Adams, and M. Mathis, "Experiences with NIMI," *Proc. Passive and Active Measurements (PAM) Wksp.*, 2000.

[6] The Réseaux IP Européens (RIPE) Web site, available at: http://www.ripe.net

[7] The Netsizer Web site, available at: http://www.netsizer.com

[8] The Cooperative Association for Internet Data Analysis (CAIDA) Web site, available at: http://www.caida.org/

[9] C. Fraleigh *et al.* "Design and Deployment of a Passive Monitoring Infrastructure," *Proc. Passive and Active Measurements (PAM) Wksp.*, 2001.

[10] Réseaux National de Recherche en Télécommunications. (2001) METROlogie Pour l'Internet et ses Services (METROPOLIS), available at: http://www.telecom.gouv.fr/rnrt/, http://www-rp.lip6.fr/metrologie

[11] P. Owezarski, "IP Network Monitoring and Measurements: Techniques and Experiences (Tutorial)," *Int'l. Wksps. on Inter-*

Tools	Per-hop capacity	Available bandwidth	End-to-end capacity	Bulk-Transfer-capacity	One-way delay	One-way delay var.	Round-trip time
bing	u						
bprobe			u				
clink	u						
cprobe		u					
IGI		u					
Iperf				b		b (UDP)	
nettimer	u		u/b				
owping					b		
pathchar	u						
pathChirp		u					
pathload		u					
pathrate			u				
pchar	u						
ping							u/rt
pipechar/NCS	u	u					
QoSMet					b	b	
sprobe			b				
spruce		u					
sting							
traceroute							
TReno				u			

■ Table 2. *Measurement tool characteristics (continued on next page).*

active Distributed Multimedia Systems and Protocols for Multimedia Systems (IDMS/PROMS'2002), Coimbra, Portugal, Nov. 2002.
[12] The Tcpdump and Libpcap Web site, available at: http://www.tcpdump.org
[13] X. Wang and H. Schulzrinne, "Comparison of Adaptive Internet Multimedia Applications," *Institute of Elec., Info. and Commun. Engineers (IEICE) Trans. Commun.*, vol. E82-B, June 1999, pp. 806–18.
[14] V. Bharghavan and V. Gupta, "A Framework for Application Adaptation in Mobile Computing Environments," *Proc. IEEE Compsac'97*, Nov. 1997, available at: http://citeseer.nj.nec.com/bharghavan97framework.html
[15] P. Keleher, J. K. Hollingsworth, and D. Perkovic, "Exposing Application Alternatives," *Proc. IEEE Int'l. Conf. Distributed Comp. Sys.*, 1999, pp. 384–92.
[16] S. N. Bhatti and G. Knight, "Enabling QoS Adaptation Decisions for Internet Applications," *Comp. Net.*, vol. 31, 1999, pp. 669–92.
[17] A. Friday *et al.*, "Developing Adaptative Applications: The MOST Experience," *Integrated Computer-Aided Engineering*, vol. 6, no. 2, 1999, pp. 143–57.
[18] G. Blair *et al.*, "A Principled Approach to Supporting Adaptation in Distributed Mobile Environments," *Proc. 5th IEEE Int'l. Symp. Software Engineering for Parallel and Distributed Systems (PDSE-2000)*, June 2000.
[19] F. Michaut and F. Lepage, "A QoS architecture for application execution adaptation," *Proc. 3rd ACIS Int'l. Conf. Software Eng., Artificial Intelligence, Net. and Parallel/Distributed Comp. (SNPD'02)*, June 2002.
[20] R. S. Prasad *et al.*, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," IEEE Network, Nov. 2003.
[21] R. Prasad, C. Dovrolis, and B. Mah, "The Effect of Layer-2 Store-and-Forward Devices on Per-Hop Capacity Estimation," *Proc. IEEE Infocom*, San Francisco, USA, Apr. 2003.
[22] V. Paxson *et al.*, "Framework for IP Performance Metrics, RFC 2330," May 1998.
[23] V. Raisanen, G. Grotefeld, and A. Morton, "Network Performance Measurement for Periodic Streams, RFC 3432," Nov. 2002.
[24] S. Shalunov *et al.*, "A One-Way Active Measurement Protocol (OWAMP), Internet Draft, Work in Progress," 2004.
[25] owping/owampd: An Implementation of the One-Way Active Measurement Protocol, available at: http://e2epi.internet2.edu/owamp/

Tools	Packet loss	Packet re-ordering	Route	Measurement class	Environment	Protocol	Operating sytem
bing				a	nc	ICMP	Most Unix
bprobe				a	nc	ICMP	SGI Irix
clink				a	nc	ICMP	Most Unix
cprobe				a	nc	ICMP	SGI Irix
IGI				a	c	UDP	Most Unix
Iperf	b UDP)			a	c	TCP, UDP	Most OS
nettimer				a/p	c/nc	TCP	Linux
owping	b	b		a	c	UDP	Linux
pathchar				a	nc	UDP	Most Unix
pathChirp				a	c	UDP	Most Unix
pathload				a	c	UDP	Most Unix
pathrate				a	c	UDP	Most Unix
pchar				a	nc	ICMP	Most Unix
ping	u/rt			a	nc	ICMP	Most OS
pipechar/NCS				a	nc	ICMP	Most Unix
QoSMet	b	b		a	c	UDP	Linux
sprobe				a	nc	TCP	FreeBSD Linux
spruce				a	c	UDP	most Unix
sting	b	b		a	nc	TCP	FreeBSD Linux
traceroute			u	a	nc	UDP, ICMP	most OS
TReno				a	nc	UDP, ICMP	most Unix

Measured parameters. u: unidirectional (measurements are made from source to destination); b: bidirectional (measurements are made from source to destination or from destination to source); u/rt: unidrectional and round-trip (measurements are made for "source-destination-source" path only).
Measurement class. a: active; p: passive.
Tool type. c: cooperative; nc: non-cooperative.

■ Table 2. *Measurement tool characteristics.*

[26] F. Michaut, "QoSMet, a Quality of Service Measurement Tool," 2004, available at: http://michaut.valerie.free.fr/qosmet

[27] A. Adams and M. Mathis, "A System for Flexible Network Performance Measurement," Japan, 2000.

[28] M. Mathis and J. Mahdavi, "Diagnosing Internet Congestion with a Transport Layer Performance Tool," *Proc. INET'96*, Montreal, QC, June 1996.

[29] V. Jacobson, Traceroute, available at: hftp://ftp.ee.lbl.gov/traceroute.tar.gze

[30] The Multicast-based Inference of Network-internal Characteristics (MINC) Web site, available at: http://www-net.cs.umass.edu/ minc/

[31] A. Adams *et al.*, "The Use of End-to-end Multicast Measurements for Characterizing Internet Network Behavior," *IEEE Commun. Mag.*, vol. 38, no. 5, May 2000.

[32] D. V. A. Pasztor, "A Precision Infrastructure for Active Probing," *Proc. Passive and Active Measurements (PAM) Wksp.*, 2001.

[33] G. Almes, S. Kalidindi, and M. Zekauskas, "A One-Way Delay Metric for IPPM, RFC 2679," Sept. 1999.

[34] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, Oct. 1991, pp. 1482–93.

[35] W. Jiang and H. Schulzrinne, "QoS Measurement of Internet Real-Time Multimedia Services," Columbia University, New York, Tech. Rep. CUCS015-99, Dec. 1999.

[36] D. Mills, "Network Time Protocol (Version 3), Specification, Implementation and Analysis, Request For Comments RFC 1305," Mar. 1992.

[37] __, "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, Request For Comments RFC 2030," Oct. 1996.

[38] A. Pasztor and D. Veitch, "PC Based Precision Timing Without GPS," *Proc. ACM SIGMETRICS'02*, 2002.

[39] W. Jiang and T. F. Williams, "Detecting and Measuring Asymmetric Links in an IP Network," *Proc. GlobeComm Global Internet 1999*, 1999.

[40] C. Demichelis and P. Chimento, "IP Packet Delay Variation Metric for IPPM, RFC 3393," Nov. 2002.

[41] H. Schulzrinne *et al.*, "RTP: A Transport Protocol for Real-Time Applications, Request For Comments RFC 1889," Jan. 1996.

[42] A. Hafid and G. von Bochmann, "Quality-of-Service Adaptation in Distributed Multimedia Applications," *Multimedia Systems*, vol. 6, no. 5, 1998, pp. 299–315.

[43] G. Almes, S. Kalidindi, and M. Zekauskas, "A Round-trip Delay Metric for IPPM, RFC 2681," Sept. 1999.

[44] __, "A One-Way Packet Loss Metric for IPPM, RFC 2680," Sept. 1999.

[45] M. S. Borella and D. Swider, "Internet Packet Loss: Measurement and Implications for End to End QoS," *Proc. ICPP Wksps. Architectural and OS Support for Multimedia Applications/Flexible Commun. Sys./Wireless Net. and Mobile Comp.*, 1998.

[46] R. Koodli and R. Ravikanth, "One-way Loss Pattern Sample Metrics, RFC 3357," Aug. 2002.

[47] S. Savage, "Sting: A TCP-based Network Measurement Tool," *Proc. USENIX Symp. Internet Tech. and Sys.*, Boulder, USA, Oct. 1999, pp. 71–79.

[48] A. Morton *et al.*, "Packet Reordering Metric for IPPM, Internet Draft, Work in Progress," 2004.

[49] J. Bellardo and S. Savage, "Measuring Packet Reordering," *Proc. ACM SIGCOMM Internet Measurement Wksp.*, Marseille, France, Nov. 2002.

[50] J. C.-R. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is not Pathological Network Behavior," *IEEE Trans. Net.*, vol. 7, no. 6, Dec. 1999, pp. 789–98.

[51] S. Jaiswal *et al.*, "Measurement and classification of Out-of-Sequence Packets in a Tier-1 IP Backbone," *Proc. ACM/SIGCOMM Internet Measurement Wksp.*, Marseille, France, Nov. 2002.

[52] E. Blanton and M. Allman, "On Making TCP More Robust to Packet Reordering," *ACM Comp. Commun. Rev.*, vol. 32, no. 1, 2002.

[53] M. Zhang *et al.*, "Improving TCP Performance under Reordering with DSACK," *Int'l. Comp. Science Institute*, Tech. Rep. TR-02-006, July 2002.

[54] P. Amer *et al.*, "Partially-Ordered, Partially-Reliable Transport Service for Multimedia Applications," *Proc. 1st ARLA/ATIRP Conf.*, MD, USA, Jan. 1997.

[55] P. Berthou *et al.*, "Partial Ordered and Reliable Multimedia Transport Protocol for Satellite Communications," *Proc. 5th European Conf. Satellite Commun. (ESCSC-5)*, Toulouse, France, Nov. 1999.

[56] V. Lecuire, F. Lepage, and K. Kammoun, "Enhancing Quality of MPEG Video through Partially Reliable Transport Service in Interactive Application," *Proc. 4th IFIP/IEEE Int'l. Conf. Management of Multimedia Net. and Services (MMNS'2001)*, Chicago, USA, Oct. 2001.

[57] V. Paxson, "End-to-end Routing Behavior in the Internet," *Proc. ACM SIGCOMM'96*, Stanford, USA, Oct. 1996, pp. 25-39.

[58] A. Jayasumana *et al.*, "Reorder Density Function — A Metric for Packet Reordering Measurement, Internet Draft, Work in Progress," 2003.

[59] M. Mathis *et al.*, "TCP Selective Acknowledgment Options, Request For Comments RFC 2018," Oct. 1996.

[60] M. Jain and C. Dovrolis, "Pathload: A Measurement Tool for Available Bandwidth Estimation," *Proc. Passive and Active Measurements (PAM) Wksp.*, 2002, available at: http://www.pathrate.org

[61] A. Pasztor and D. Veitch, "Active probing using packet quartets," *Proc. ACM/SIGCOMM Internet Measurement Wksp.*, Marseille, France, Nov. 2002, available at: http://www.icir.org/vern/imw-2002/

[62] V. Jacobson, "Pathchar, a Tool to Infer Characteristics of Internet Paths," Apr. 1997, available at: ftp://ftp.ee.lbl.gov/pathchar/ msri-talk.pdf

[63] S. Bellovin, "A Best-Case Network Performance Model," ATT Research, Tech. Rep., Feb. 1992.

[64] The Pchar Web site, available at: http://employees.org/~bmah/Software/pchar

[65] P. Beyssac, (1995) BING, a Bandwidth Measurement Tool based on pING, available at: http://www.cnam.fr/reseau/bing.html

[66] A. B. Downey, "Using Pathchar to Estimate Internet Link Characteristics," *Proc. ACM SIGCOMM Conf. Applications, Tech., Architectures, and Protocols for Comp. Commun.*, 1999, pp. 222–23.

[67] K. Lai and M. Baker, "Measuring Link Bandwidths using a Deterministic Model of Packet Delay," *Proc. ACM SIGCOMM Conf. Applications, Technologies, Architectures, and Protocols for Comp. Commun.*, Stockholm, Sweden, 2000.

[68] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM Symp. Commun. Architectures and Protocols*, Aug. 1988, pp. 314–29.

[69] R. Carter and M. Crovella, "Measuring Bottleneck Link Speed in Packet-switched Networks," Boston University, Tech. Rep. 1996-006, Mar. 1996, available at: http://citeseer.nj.nec.com/carter96measuring.html

[70] S. Saroiu, (2001) SProbe: A Fast Tool for Measuring Bottleneck Bandwidth in Uncooperative Environments, available at: http://sprobe.cs.washington.edu

[71] C. Dovrolis, The Pathrate and Pathload Web site, available at: http://www.pathrate.org

[72] C. Dovrolis, P. Ramanathan, and D. Moore, "What Do Packet Dispersion Techniques Measure?" *Proc. INFOCOM*, Apr. 2001, pp. 905–14.

[73] K. Lai and M. Baker, "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth," *Proc. USENIX Symp. Internet Tech. and Sys.*, San Francisco, USA, Mar. 2001.

[74] V. Paxson, "End-to-end Internet Packet Dynamics," *IEEE/ACM Trans. Net.*, vol. 7, no. 3, 1997, pp. 277–92.

[75] G. Jin *et al.*, "Network Characterization Service (NCS)," *Proc. 10th IEEE Symp. High Performance Distributed Comp.*, Aug. 2001.

[76] J. Strauss, D. Katabi, and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," *Proc. Internet Measurements Conf.*, 2003.

[77] V. Ribeiro *et al.*, "pathchirp: Efficient Available Bandwidth Estimation for Network Paths," *Proc. Passive and Active Measurements (PAM) Wksp.*, 2003.

[78] B. Melander, M. Bjorkman, and P. Gunningberg, "A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks," *Proc. GlobeComm Global Internet Symp.*, 2000.

[79] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE JSAC*, vol. 21, no. 6, Aug. 2003.

[80] T. Oetiker and D. Rand, The Multi Router Traffic Grapher (MRTG) Web site, available at: http://www.people.ee.ethz.ch/~oetiker/webtools/mrtg

[81] T. Anjali *et al.*, "ABEst: An Available Bandwidth Estimator within an Autonomous System," *Proc. IEEE Globecom*, 2002.

[82] T. Anjali *et al.*, "Available Bandwidth Measurement in IP Networks, Internet Draft, Work in Progress."

[83] M. Mathis and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics, Request For Comments RFC 3148," July 2001.

[84] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control, Request For Comments RFC 2581," Apr. 1999.

[85] The Iperf Web site, available at: http://dast.nlanr.net/Projects/Iperf

[86] P. Owezarski and N. Larrieu, "Internet Traffic Characterization — An Analysis of Traffic Oscillations," *Proc. 7th IEEE Int'l. Conf. High Speed Networks and Multimedia Commun. (HSNMC 2004)*, Toulouse, France, June 2004.

[87] M. Jain and C. Dovrolis, "End-to-End Available BW Measurement Methodology, Dynamics, and Relation with TCP Throughput," *Proc. ACM SIGCOMM Conf.*, 2002.

[88] V. Paxson, "On Calibrating Measurements of Packet Transit Times," *Proc. ACM SIGMETRICS'98*, 1998.

[89] F. Michaut and F. Lepage, "A Tool to Monitor the Network Quality of Service," *Proc. IFIP-IEEE Conf. Network Control and Eng. (NET-CON'2002)*, Oct. 2002.

BIOGRAPHIES

FABIEN MICHAUT (fabien.michaut@cran.uhp-nancy.fr) received his engineer degree at the ESSTIN Engineering School, France, in 1999, and his Ph.D. at Henri Poincaré University in Nancy, France, in 2003. He is currently a lecturer at Henri Poincaré University of Nancy. His recent research work has focused on adaptation of distributed applications to communication network Quality of Service (QoS) and QoS metrology.

FRANCIS LEPAGE (francis.lepage@cran.uhp-nancy.fr) is a full professor at Henri Poincaré University in Nancy, France. He received his Ph.D. in 1976 and his Doctor es Sciences degree in 1986 at the University of Nancy. His research interest is in critical real-time controlled systems, especially real-time constrained communication networks. He has supervised 17 Ph.D. theses and is the author or co-author of four books and approximately 100 publications.