

Network performance measurement tools
a comprehensive comparison

Rody Schoonderwoerd

master thesis - November, 2002

Abstract

In this thesis we will make a comparison between commonly used network performance measurement tools. The comparison focuses at the observed network characteristics, the used measurement methodologies and the resulting observations. There are a few common active probing techniques (measurement methodologies) on which most tools are based. These are variable packet size, packet dispersion and path flooding measurements. The first of which is used mostly for estimation of per-link bandwidth capacity and per-link available bandwidth. The second is mostly used for estimating the bandwidth capacity and available bandwidth over the path. The last technique is used to measure the bulk-transfer capacity, which resembles the available bandwidth. From our experiments we conclude that the results of the tools vary a lot when the tools are applied to a certain network connection. This may be caused by them not being compatible with current hardware technologies or ignoring the existing layer 2 network devices. We have also observed the degree of intrusiveness of the tools, which also shows great variety. When considering using any of these tools as sensors in a grid performance measurement infrastructure, careful thought has to be given to the degree of intrusiveness, running times and hardware compatibility.

Contents

1	Introduction	4
1.1	Related work	4
2	Network Characteristics	6
2.1	Network topology	6
2.2	Why characteristics?	7
2.2.1	Hierarchy of topology and characteristics	8
2.3	Bandwidth	9
2.3.1	Throughput	10
2.3.2	End-to-end available bandwidth	10
2.3.3	Bulk-transfer capacity	11
2.3.4	End-to-end bandwidth capacity	11
2.3.5	Per-link capacity	11
2.4	Routing	12
2.5	Delay	12
2.6	Loss	13
2.7	Summary	13
3	Measurement methodologies	16
3.1	Active or Passive	16
3.2	Delay	17
3.3	Packet dispersion technique	17
3.4	Self-Loading Periodic Streams (SLOPS)	20
3.5	Variable Packet Size (VPS)	20
3.5.1	The basics of the Variable Packet Size technique	20
3.5.2	The VPS technique and layer 2 devices	21
3.5.3	Even-Odd	22
3.6	Tailgating	23
3.7	TCP simulation	24
3.8	Routing and topology	24
3.9	Summary	25

4	General properties	28
4.1	Cooperativeness of environment	28
4.2	Intrusiveness	29
4.3	Scalability	30
4.4	Network requirements	30
	4.4.1 Security	30
	4.4.2 Operating systems and portability	31
4.5	Cross-traffic	31
5	The experiment	33
5.1	Environment	33
5.2	Experiment design	34
5.3	Results	35
	5.3.1 Running times	35
	5.3.2 Generated traffic estimate	36
	5.3.3 Bandwidth type & Measured bandwidth	36
	5.3.4 Observed intrusiveness	38
5.4	Bugs, crashes, failures	39
6	Conclusions	41
6.1	Conclusions	41
6.2	Future work	42

Chapter 1

Introduction

Over the past years, many software tools have been developed to gain information on network performance. These Network Performance Measurement Tools (NPMTs) are based on different principles. They incorporate various techniques and focus on various network characteristics. This document aims at making a comparison of NPMTs at two levels: the observed results and the measurement methodologies producing these results.

Comparing many different programs requires consistent terminology. However, while reading tool documentation, it appeared that consistency was often missing. For example, what is the exact difference between measuring bandwidth 'per-link' and 'per-hop'? Whatever the causes of the inconsistency may be, it forces us to be very clear about the terminology. This is done in Chapter 2: some commonly used terminology is compared and a choice is made on which definitions we use. Chapter 3 explains some commonly used techniques based on the terminology defined in the previous Chapter. These techniques are linked to the tools using them. Chapter 4 compares the general properties of the NPMTs. These properties are security issues, network requirements and the like. In Chapter 5 we describe an experiment to compare the NPMTs using data generated by the tools while running on a wide-area network. Finally, Chapter 6 contains conclusions and future work.

1.1 Related work

Over the past ten years, a lot of software tools have been developed to measure properties of the Internet. These properties are often related to transfer speed of data from one host to another. When a new tool was built, it was compared to other tools that perform comparable tasks. Some examples of this can be found in [33, 19, 25, 2], in which tools are compared to various other NPMTs and techniques.

For use in grid environments, [13] compares several monitoring architectures. The comparison is based on many properties resembling those we use: scalability, intrusiveness, security and more. [13] does not provide any quantitative results. The monitoring architectures can operate with many different sensors to obtain information about the network. Our document is a comparison of smaller tools which could probably be used as

sensors.

Each NPMT was built using some algorithm. This algorithm is used to measure a particular quantity related to network performance. Much research has been done into network performance issues. Many individual research projects will be discussed in the following chapters.

Chapter 2

Network Characteristics

The main purpose of this chapter is to explain the term 'characteristics' and related terminology. To do so, we need a consistent set of basic terminology. Section 2.1 defines our view of network topology. Using the terms defined here, we can explain network characteristics in Section 2.2 and the relation to network metrics. Sections 2.3 to 2.6 deal with particular characteristics and Section 2.7 provides a summary of term definitions. This chapter is not meant to be introducing a standard, but just to explain the way we use the terminology.

2.1 Network topology

When a packet is sent over a network, it may need to pass several intermediate store-and-forward devices. These devices are considered either layer 2 or layer 3 devices of the OSI model explained in [47]. These devices use a routing algorithm to determine where the packet needs to go and have a queuing algorithm. The device is called a 'node'. Each node is viewed as having an interface receiving packets (incoming interface), an interface sending packets (outgoing interface) and their protocol stacks. The ordered list of nodes 1 to n forms the 'path' along which the packets travel from source node to destination node (this requires physical wire connecting them). Now we can define 'link' as being the combination of the outgoing interface of node i (chosen from $1..n-1$), the incoming interface $i+1$ and the wire between the two, see Figure 2.1. Links are not shown in Figure 2.2, because a link between two layer 3 devices is in fact often a path consisting of layer 2 devices. So a link may be regarded as a path when considering different protocols on various OSI layers.

Some people speak of 'hops' when talking about links. We prefer to use the number of hops between two nodes as a distance measure, using the term 'hops' for *this* purpose only. The NPMTs measure characteristics either 'end-to-end' across the entire path, or 'per-link' (sometimes called 'per-hop'). The differences between these methods will become clear when we discuss measurement methodologies in Chapter 3.

But what is topology exactly? There is, of course, a hardware point of view. The topology of a network is the way it is composed of nodes and links and their locations. [22] states that applications may have different views of locations: physical and functional.

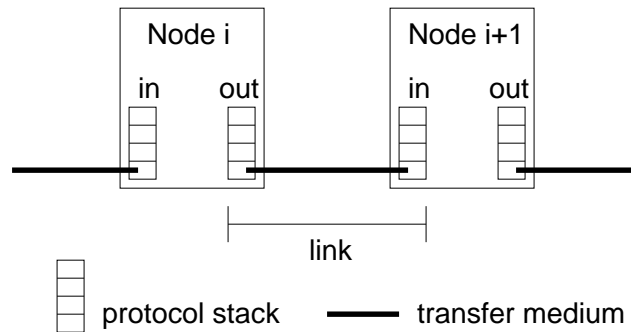


Figure 2.1: One link of a network path

Physical means that the distance *in meters* between nodes is defined as real-life physical location of the network nodes. Opposed to that is the functional view, which considers a node to be closest *in time* if packets can be transferred faster to it than to any other node. The transfer speed is of course limited by the capabilities of the physical hardware and packets from other sources: the cross-traffic. The functional view is an end-to-end point of view, ignoring most of the physical network devices and using performance characteristics to measure distance. Section 2.2 explains why and how we use the term characteristic.

2.2 Why characteristics?

In the recent history of network performance measurement, there have been some problems concerning terminology. The IETF use 'metrics', 'measurement methodologies' and 'measurements'. The RFC on IP Performance Metrics [5] provides a framework for creating metrics. For example in [3], which defines the metric of 'bulk transfer capacity'. But what is a metric? In [5] we find "quantities related to the performance and reliability of the Internet". This, of course is a very vague definition, so here's another point of view.

The Network Measurements Working Group (NMWG) of the Global Grid Forum use a different set of terms in [22]: 'characteristic', 'measurement methodology' and 'observation' as opposed to respectively 'metric', 'measurement methodology', 'measurement'. Characteristics are defined as "intrinsic properties of a portion of the network that are related to the performance and reliability of the Internet" [22]. A measurement methodology is "a technique for recording or estimating a characteristic" [22]. An observation is "an instance of output from a measurement" [22]. The main reason for this new set of terms was that the IETF terms were not well fitted to the needs of the NMWG and were somehow misinterpreted by others. We decided to adopt the NMWG terminology, although it is yet another set of terms.

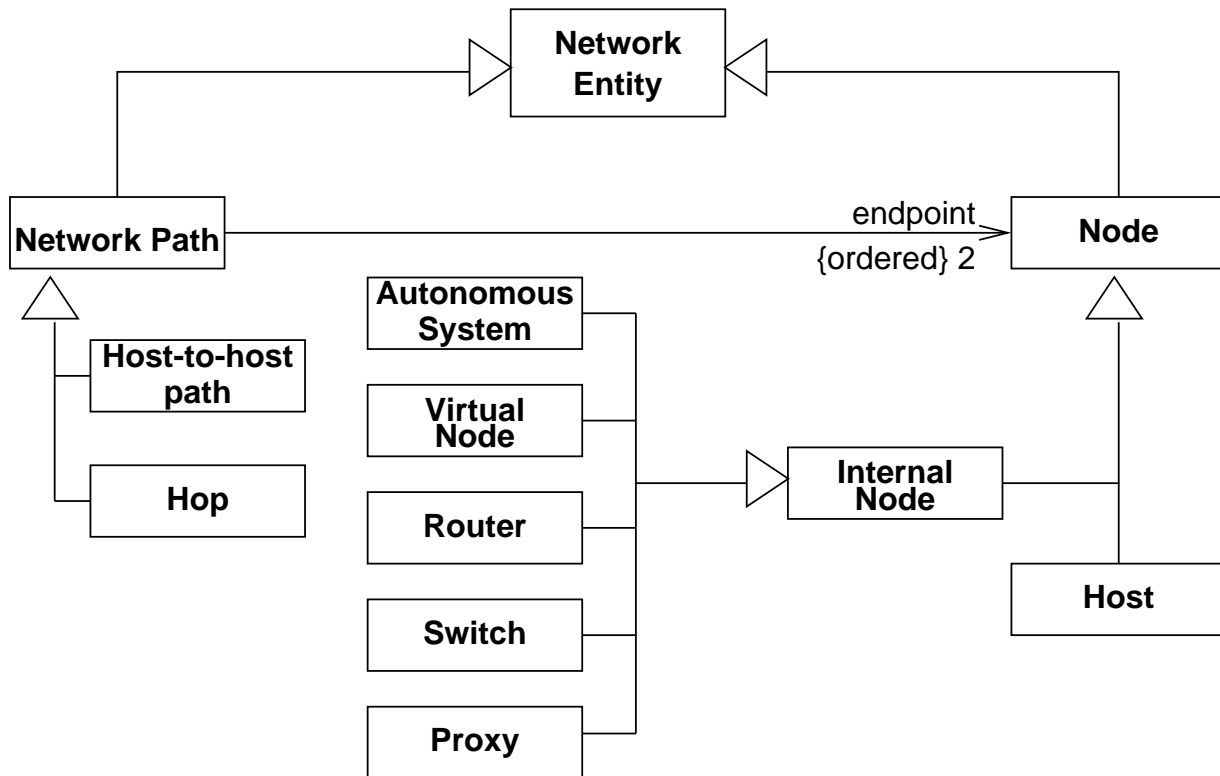


Figure 2.2: Network topology entities [22]

2.2.1 Hierarchy of topology and characteristics

The NMWG derived a hierarchy of network characteristics, which we will use here. A clear distinction is made between network topology and network characteristics. Figure 2.3 shows how the terms defined in the previous sections relate to each other.

From the figure it may appear that every characteristic can be applied to every network entity. This is of course not true, because certain characteristics describe properties of paths between two nodes and do not apply to a single node. For each characteristic there may be an infinite number of measurement methodologies to calculate or estimate it. In this thesis we will discuss only those characteristics and methodologies which are used by the NPMTs we compare.

Figure 2.4 shows the hierarchy of characteristics. Characteristics can be divided into two categories, or rather ways to observe them: those describing properties of individual links (per-link, also known as per-hop) and those only working with entire (layer 3 device) paths (also known as end-to-end). In our comparison of NPMTs, we will focus mainly on bandwidth characteristics measured by the tools. The remainder of this chapter will deal with various network characteristics. Not all of these will be used in our comparison of network performance measurement tools.

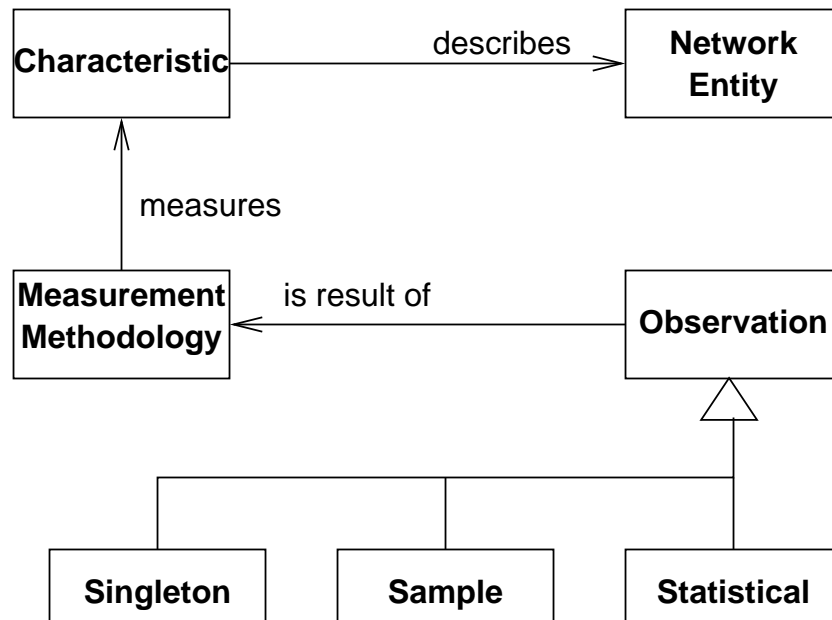


Figure 2.3: Relationships between the defined terms [22]

2.3 Bandwidth

Like in [15], sometimes only a general definition of bandwidth is given: “Bandwidth is a measure of the quantity of data that can be carried by a network in a fixed period of time” [15]. The NMWG divide bandwidth into four sub-characteristics shown in Figure 2.4: achievable bandwidth, bandwidth capacity, available bandwidth and bandwidth utilization. Each of those is said to be possible end-to-end and per-link. This provides us with a total of eight different types of bandwidth. However, after studying NPMT documentation and [26], we created the following list of commonly used terms related to bandwidth:

- Throughput
- Bulk-transfer capacity
- End-to-end available bandwidth
- End-to-end bandwidth capacity
- Per-link capacity

How do these relate to the NMWG variants? First of all, some are clearly the same, only throughput and bulk-transfer capacity (BTC) are problem cases. We decided to relate these to the NMWG in the following way. Throughput is the same as achievable bandwidth and the BTC is a variety of throughput. These terms will be explained later. There are (at least) two more commonly used network characteristics, which are derived from the above:

1. bottleneck bandwidth

This is just another name for end-to-end bandwidth capacity [26], but is also defined as “The maximum throughput that is ideally obtained across the slowest link of a network path” [33].

2. bandwidth utilization

This is defined as “The aggregate traffic currently on that link or path” [22]. We will not discuss utilization any further, because none of the tools claims to measure it. However, an estimate can simply be found: the bandwidth capacity minus the available bandwidth.

2.3.1 Throughput

According to [43], throughput should be defined as “The amount of data transferred from one place to another or processed in a specified amount of time”. Another definition is: “The number of transferred bytes over a network path during a fixed amount of time,” from [33]. The latter is much more specific and therefore fits better to our needs, so we will use: “Throughput is the amount of bytes transferred over a network path during a fixed amount of time”. In the hierarchy of characteristics, throughput is the same as achievable bandwidth. A specific type of throughput is the bulk-transfer capacity (Section 2.3.3).

2.3.2 End-to-end available bandwidth

The end-to-end available bandwidth is, roughly said, what is left of a path’s capacity, given its cross-traffic caused by other sources. Below are four possible definitions, which each provide interesting parts to use:

- “(...) the end-to-end availablebandw is defined as the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic in P.” [24] (P is a path.)
- “The maximum attainable throughput of a newly started flow over a network path” [33]
- “The maximum throughput that the path can provide to an application, given the path’s current cross traffic load” [22]
- “The available bandwidth of a network element is its physical bandwidth minus the bandwidth of the network element currently used by other traffic.”[2]

Combining these four, we created the following: “The end-to-end available bandwidth is the maximum throughput a path can provide at a certain time, given the current amount of cross-traffic”. We have left out one important part. The first definition says that the available bandwidth equals the bandwidth that can be used without reducing the current

cross-traffic. This is an interesting statement, because this means you cannot measure the available bandwidth by just sending as much as you can. Taking up as much bandwidth as you can will squeeze the cross-traffic streams and reduce them, resulting in an estimate of throughput or bulk-transfer capacity instead of available bandwidth. When measuring available bandwidth, very little traffic should be generated by the NPMT. This issue is treated in Section 4.2 about intrusiveness.

2.3.3 Bulk-transfer capacity

[26] defines this metric (BTC) using the RFC 3148 [3]: “The BTC of a path in a certain time period is the throughput of a bulk TCP transfer, when the transfer is only limited by the network resources and not by limitations at the end-systems.” Effectively measuring what happens, in terms of bytes per second, when you start a TCP connection and then send as much data as possible. We will use the following definition: “The bulk-transfer capacity is the throughput of a persistent TCP transfer”, using the definition of throughput which was presented in Section 2.3.1.

When comparing the BTC with end-to-end available bandwidth, we notice some striking differences. One fundamental difference is that available bandwidth is the amount of usable bandwidth without affecting the current cross-traffic. The BTC can only be measured by sending as much as possible, thus grabbing as much bandwidth as possible and limiting the cross-traffic. Experiments show in [24] that the BTC results may very well be 20% to 30% higher than the available bandwidth. A second difference is that the BTC is simulating a persistent TCP transfer, thus taking considerable time. This is unlike the available bandwidth measurement methodologies, which take up as little time as possible to reduce intrusiveness.

2.3.4 End-to-end bandwidth capacity

The end-to-end bandwidth capacity is the maximum amount of bits per second the devices on the path can forward from source to destination. Most authors use roughly the same definition for this metric, like “The capacity is defined as the maximum rate that the path can provide to a flow, when there is no other traffic in P.” [25] (P is a path.) and “Capacity is the theoretical maximum link-layer throughput that the link or path has available, when there is no competing traffic” [34]. Based on the throughput, we use the definition “The maximum throughput a path can provide when there is no cross-traffic”.

2.3.5 Per-link capacity

The per-link capacity is the theoretical maximum capacity which a certain link can provide when there is no cross-traffic. This metric is defined by [26] as “the maximum throughput that a single L3 hop can provide to a flow when there is no other traffic in that hop” (a ‘L3 hop’ is a layer 3 link). This characteristic is related to the end-to-end capacity, which is

the minimum of all per-link capacities on a path. The usual technique to measure per-link capacity is the Variable Packet Size technique, described in Chapter 3.

2.4 Routing

Routing information about packets being sent in a network is closely related to topology issues. In Section 2.1 we introduced the concept of functional topology. The functional view is probably more useful for routing than the physical view. We compare this functional view with the “Effective Network Views (ENV)” found in [15]. They speak of effective topology for an application as being “the network organization that is consistent with observable network behavior”. And they define the ENV as providing “a representation of the logical network as perceived by the application” [15]. The ENV is of great practical use, because it allows for optimization at the user application layer. The ENV seems to largely ignore the physical location of network devices and groups those together that “have significant interaction and similarities in response to events on the network” [15]. The forming of these groups will be dealt with in Section 3.8 and represents a way to quantify the functional view of a network.

2.5 Delay

Most NPMTs use or report some variant of the delay characteristic. The two major subtypes of delay are shown in Figure 2.4: one-way delay (OWD) and round-trip time (RTT). These two are closely related. The OWD is the time it takes for a packet to travel from source to destination. The RTT is found by adding the time to travel back. So, ideally, the RTT is about double the OWD time. If not, this may be an indication that the observed path is not symmetrical. This means that, due to routing algorithms and cross-traffic, packets being sent to a certain node use a different route than the packets returning from that node. When measuring the RTT between two hosts, if the paths are not symmetrical, results may be misinterpreted. But even on physically symmetrical paths, there may be effects like cross-traffic and queuing. These effects may slow down a path in one direction or both directions. RTT is reported by traceroute and ping programs.

There is also a document describing an OWD metric, RFC 2679 [6]. A rephrased version of this definition is: the one-way delay from a source to a destination is the elapsed time between sending the first bit of a packet from the source and receiving the last bit of that packet at the destination.

In the RFC 2681 [7], a metric can be found describing the RTT. We provide a rephrased version of this definition. The round-trip delay from a source to a destination is the elapsed time between sending the first bit of a packet from the source, receiving that packet at the destination, immediately sending a packet back to the source and receiving the last bit of that packet at the source.

Both the sending time and receiving time are wire-time as defined in RFC2330. Using an Internet host H and an Internet link L:

“For a given packet P, the ‘wire arrival time’ of P at H on L is the first time T at which any bit of P has appeared at H’s observational position on L.” [5]

“For a given packet P, the ‘wire exit time’ of P at H on L is the first time T at which all the bits of P have appeared at H’s observational position on L.” [5]

Another term associated with delay is ‘latency’, which is the same as the OWD. The delay characteristics are often used by NPMs to estimate bandwidth. Strongly related to the delay characteristics is jitter: “The variation in delay of packets as they flow from one host to another”[22]. Jitter is a statistical observation calculated from the OWD. Therefore, in the hierarchy Figure 2.4, jitter is shown as derived from OWD.

2.6 Loss

Loss characteristics can be described similar to the delay characteristics. As said in [22], a distinction can be made between one-way loss (OWL), round-trip loss (RTL) and various statistical loss properties. In Section 2.5 was shown that round-trip observations may not always be useful. This raises some questions on whether it is useful to calculate RTL. Did the packets get lost before getting to the other end of the path or on the way back? If data is sent in a stream and generally in only one direction, OWL is far more useful than RTL. We find a definition of OWL in RFC2680 [8], which we rephrase to the following. The one-way packet loss from a source to a destination has either the value zero or the value one. The value zero indicates that a packet sent from the source at a certain time has been received by the destination. The value one indicates that the packet was not received by the destination.

This metric is defined to produce a singleton as in Figure 2.3 and can be used to create a time series from which statistical properties may be derived. Loss is caused by routers dropping packets and influenced by cross-traffic. Packets may be queued or dropped and intermediate devices may give lower priority to certain protocols.

2.7 Summary

This summary provides a list of the used definitions.

Characteristics: “intrinsic properties of a portion of the network that are related to the performance and reliability of the Internet.” [22]

Measurement methodology: “a technique for recording or estimating a characteristic.” [22]

Observation: “an instance of output from a measurement.” [22]

Bandwidth utilization: “the aggregate traffic currently on that link or path.” [22]

Throughput: the amount of bytes transferred over a network path during a fixed amount of time.

End-to-end available bandwidth: the maximum throughput a path can provide at a certain time, given the current amount of cross-traffic.

Bulk-transfer capacity: the throughput of a persistent TCP transfer.

End-to-end bandwidth capacity: the maximum throughput a path can provide when there is no cross-traffic.

Per-link capacity: “the maximum throughput that a single L3 hop can provide to a flow when there is no other traffic in that hop.”[26]

Round-trip delay: the elapsed time between sending the first bit of a packet from the source, receiving that packet at the destination, immediately sending a packet back to the source and receiving the last bit of that packet at the source.

One-way delay: the elapsed time between sending the first bit of a packet from the source and receiving the last bit of that packet at the destination.

One-way loss: has either the value zero or the value one. The value zero indicates that a packet sent from the source at a certain time has been received by the destination. The value one indicates that the packet was not received by the destination.

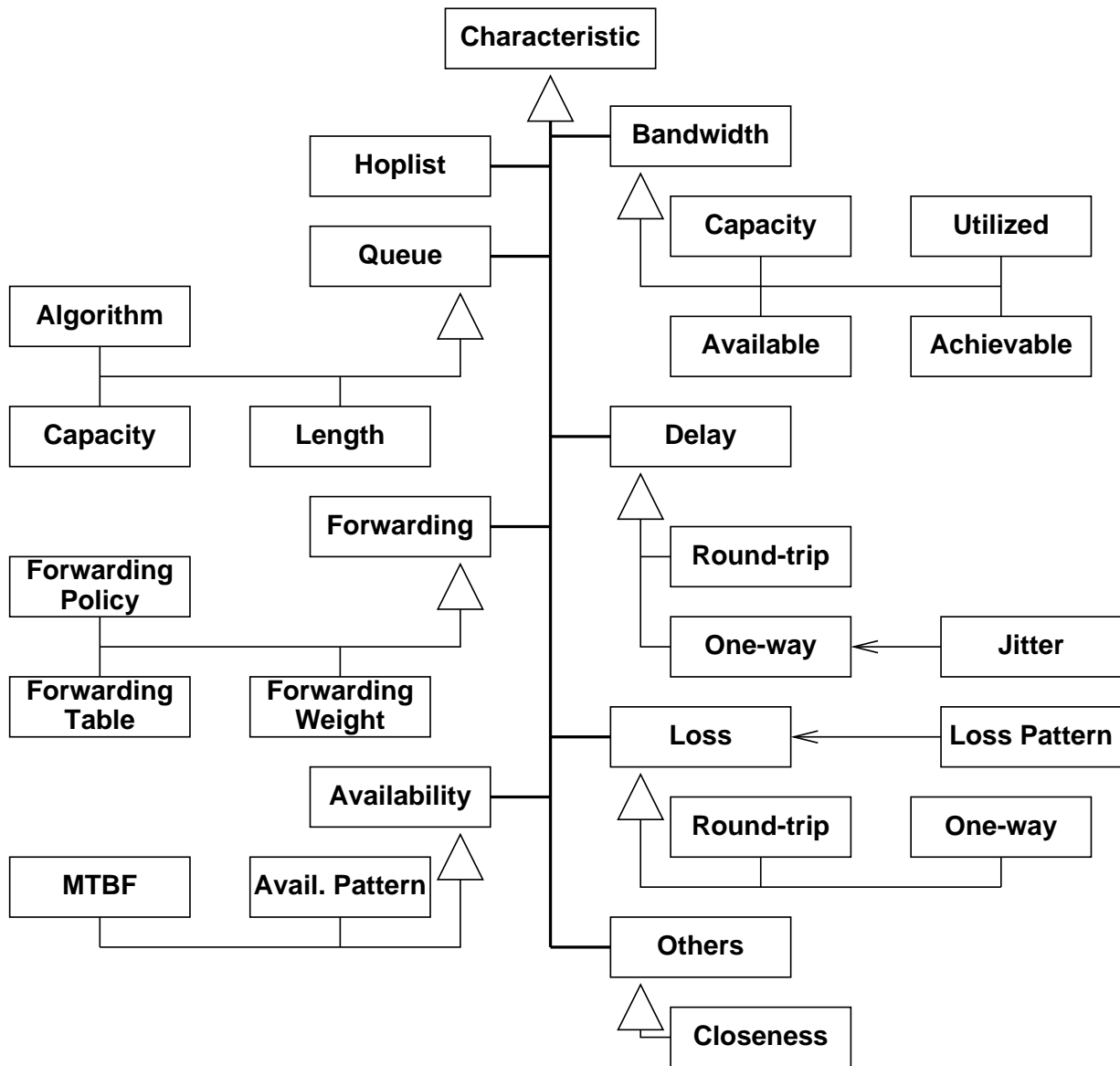


Figure 2.4: Hierarchy of network characteristics [22]

Chapter 3

Measurement methodologies

As the word 'methodology' suggests, this section deals with the way Network Performance Measurement Tools calculate (or rather estimate) network characteristics. Section 2.2 defines a measurement methodology as "a technique for recording or estimating a characteristic" [22]. So 'technique' in this context is a combination of algorithms, protocols and statistics.

Section 3.1 deals with the difference between active and passive monitoring. In the remainder of this chapter, commonly used methodologies are explained in detail and examples of tools using them are provided. The information was mostly taken from the documentation of the tools and research publications by their developers. Table 3.1 lists the main sources of information of each NPMT and the release we used for testing it. Two of the tools were not tested: pathchar and SProbe. The reason for not testing pathchar is that we tested the re-implementation, pchar. SProbe was not tested because of installation problems.

3.1 Active or Passive

The various techniques used by network performance measurement tools can be separated in two categories. One is the 'active measurement', the other is 'passive measurement'. Active measurement means that the tool actively sends probing packets into the network. When measuring passively, the tool monitors the passing traffic without interfering.

Active measurements may be either one-way or two-way. The one-way technique is used by programs like ping [36]. Here, the NPMT sends out packets and waits for automatic replies from standard software on target nodes using ICMP, UDP or TCP protocols. Two-way means that there is a client-server system. The client sends probe packets to the server, which performs some calculations and replies, or just echoes packets back to the client.

Passive measurements are often less reliable than active. In [20] is said that it may be impossible to extract any useful data at all. Many efforts have been made to use passive measurements, like [30]. However, each of the NPMTs in this document uses active probing

Table 3.1: NPMT resources and used releases

Tool name	URL	Used release
bing	http://www.cnam.fr/reseau/bing.html	1.0.4
bprobe	http://cs-people.bu.edu/carter/tools/Tools.html	1.0
cprobe	http://cs-people.bu.edu/carter/tools/Tools.html	1.0
clink	http://rocky.wellesley.edu/downey/clink/	1.0
Iperf	http://dast.nlanr.net/Projects/Iperf/	1.6.1
netest	http://www-didc.lbl.gov/pipechar/	NCS 1.3 beta
Netperf	http://www.netperf.org/netperf/NetperfPage.html	2.2alpha
Nettimer	http://mosquitonet.stanford.edu/~laik/projects/nettimer/	2.3.8
pathchar	ftp://ftp.ee.lbl.gov/pathchar/	not tested
Pathload	http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/	1.0.2
Pathrate	http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/	2.1.2
pchar	http://www.employees.org/~bmah/Software/pchar/	1.4
pipechar	http://www-didc.lbl.gov/pipechar/	NCS 1.3 beta
SProbe	http://sprobe.cs.washington.edu/	not tested
traceroute	part of OS	1.4a12
TReno	http://www.psc.edu/networking/treno_info.html	0.97
ttcp	part of OS	1.12

and only Nettle can also operate in passive mode.

3.2 Delay

The OWD (one-way delay) is the time it takes for a packet to travel from source to destination, using a timestamp and synchronised clocks. The RTT (round-trip time) is measured by sending a small packet to a server and tracking the elapsed time until a response arrives. RTT is reported by traceroute (UDP) and ping (ICMP) programs, which are not part of our NPMT comparison. Nevertheless, these two programs were the first to use their respective techniques and NPMT documentation often refers to them.

3.3 Packet dispersion technique

The packet dispersion technique has been existing for some time now. The concept has been used to design cprobe and bprobe [19]. bprobe sends a series of packets across a network path with as little space as possible between them, thus creating a 'packet train' (packet trains of length two are called 'packet pairs'), see Figure 3.1. The type of the packets is ICMP ECHO or UDP. There may be problems with ICMP ECHO packets as

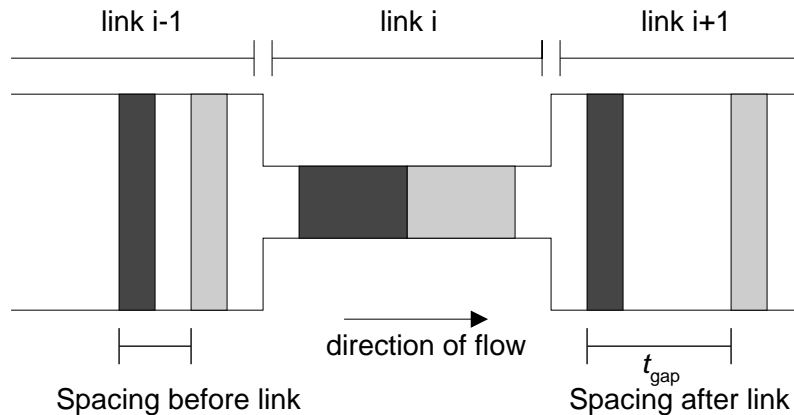


Figure 3.1: The packet dispersion technique

some routers may not process them as quickly as others.

The methodology works like this: the packet train is sent to a target host, which echoes them back. When passing through a link with smaller capacity, packets will be queued. After that link, the packets have some gap between them. The gap or dispersion between two packets is defined in [27] as the time between the arrival of the last bit of the first packet and the arrival of the last bit of the second packet. After the link with the smallest capacity, the gap will (in theory) not get any bigger. The average gap between the packets of a packet train is measured and the estimated capacity \hat{C} (in bytes per second) of the link is calculated as follows [19]:

$$\hat{C} = \frac{P}{t_{gap}}, \quad (3.1)$$

where P is the packet size in bytes and t_{gap} is the average gap in seconds. The capacity is, of course, an estimate. In [10] we find the 'Packet Pair Property' on which Equation 3.1 is based. The Packet Pair Property shows that the size of a probing packet divided by the minimum bandwidth capacity of a path equals the time between the arrival of the two probe packets.

“Let $C_{min(l)} \leq C_i$, ($\forall i, 0 \leq i \leq l$), then if we send two packets of the same size ($P_{first} = P_{second}$) with a small time difference ($t_{first,0} - t_{second,0} \leq \frac{P_{second}}{C_{min(n-1)}}$) and there is no cross-traffic, they will arrive with a difference in time equal to the size of the second packet divided by the smallest bandwidth on the path ($t_{first,n} - t_{second,n} \leq \frac{P_{second}}{C_{min(n-1)}}$)“ [10]

The property uses $C_{min(l)}$ as the minimum of all bandwidth capacities C_i on links $0, 1, \dots, n$. The capacity of the path is used by cprobe to estimate the utilization [19]:

$$\hat{U} = \frac{\hat{A}}{\hat{C}}$$

where \hat{U} is the estimated utilization of the bottleneck link. The estimated available bandwidth \hat{A} is estimated by dividing the number of bytes sent by the time between the arrival of the first and the last packet of a packet train. The authors have tested cprobe by generating traffic on the path and comparing this known amount of utilization to the available bandwidth reported by cprobe: 74% of the estimates are within 5% of the known value, and all estimates are within 25% of the known value.

Since the release of cprobe and bprobe, various similar research has been done. One result is that the estimates by these tools are not what they seem to be, according to [27]. [27] shows that when using only two packets (packet-pair) and creating a histogram of the bandwidth capacity estimates, three types of peaks can be found. The peaks at lower bandwidth (maybe a range of smaller peaks) are the “sub-capacity dispersion range”, caused by interfering cross-traffic packets. The middle peak, hopefully being also the statistical global mode, is probably the correct value for a bandwidth capacity estimate. The peaks at higher bandwidth are the “post-narrow capacity mode”, caused in the links after the narrow links, when the first probing packets have a delay of more than a second. When the number of packets in the train is increased, these peaks converge into one. This is, however, not the best estimate for the bandwidth capacity. It is a different property, called the “asymptotic dispersion rate” (ADR). The ADR is lower than the bandwidth capacity due to the influence of cross-traffic, but it does not equal the available bandwidth. [27] shows test results with different packet sizes and train lengths to support the ADR theory. In [27] is argued that only packet pairs, not packet trains should be used. This new view of packet-trains was used to develop the Pathrate tool.

The same way bprobe may be compared to Pathrate, cprobe may be compared to Pathload. The first version of Pathload was released recently, described in [25]. Very much in contrast to most other tools, Pathload does not calculate its available bandwidth directly. Its algorithm is an adaptive process based on increasing one-way delays of packet pairs when the probing rate of the sender increases. The algorithm is called ‘rate-adjustment algorithm’ [25]. It is explained in Section 3.4.

The packet pair technique may become inaccurate if the amount of cross-traffic increases. [11] shows us four different situations of what might happen when using the packet pair technique, see Figure 3.2. In the ideal case (A), packets are sent with very little time between them. At the bottleneck link the time between packets increases and this is recorded at the other end of the path (assuming client-server system). In case (B) the packets are sent with too much time between them: there is probably no queuing at the bottleneck link. If so, the bottleneck link capacity cannot be determined. The last two problem cases are caused by interfering cross-traffic. Observations will be wrong in case (C) a cross-traffic packet is queued between the two probing packets and causes a time separation bigger than the bottleneck link would have caused. Finally, (D) a packet (or more than one) may be queued before the probing packets while they have already passed the bottleneck link. The resulting observation will be the capacity of the link at which this extra queuing takes place (overestimating the capacity).

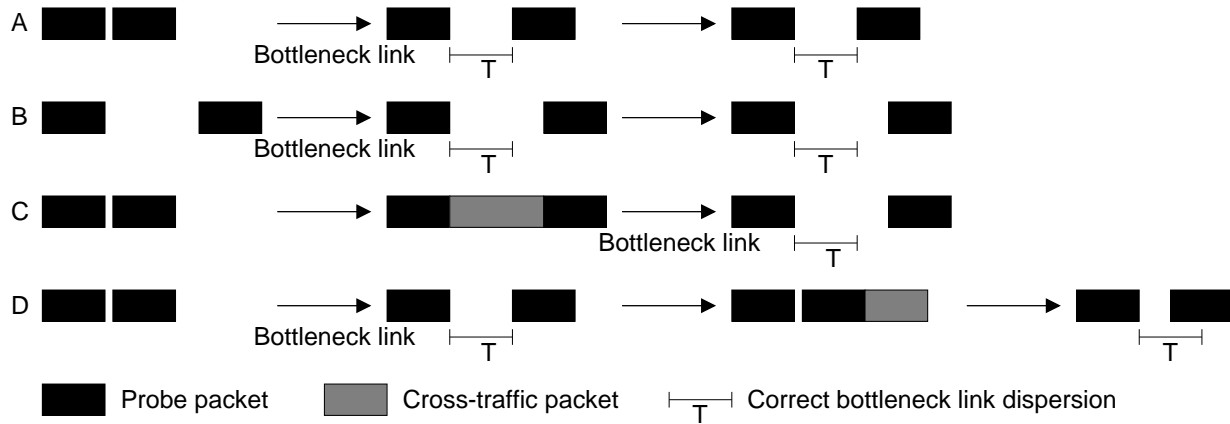


Figure 3.2: The packet pair technique and cross-traffic. Based on [11] (modified).

3.4 Self-Loading Periodic Streams (SLOPS)

The tool Pathload is based on the self-loading periodic streams technique [25] and measures available bandwidth. Several series of streams of packets are sent over the path from source to destination. The rate at which packets are being sent is supposed to be larger than the available bandwidth of the link whose available bandwidth is lowest. Packets will be queued on that link, which processes them at a constant rate. Before being sent, the packets get a timestamp. Upon receiving two successive packets, their difference in timestamps is compared to the difference in their arrival times. This is done by subtracting the One-Way Delays (OWD, arrival time minus timestamp) of each pair of successive packets, expecting to see an increasing trend. Of course, the amount of packets that need to be sent for getting an increasing trend is not known in advance. Pathload uses an adaptive algorithm (rate-adjustment algorithm) to converge the stream iteratively to the available bandwidth. Because of the adaptive algorithm, there is no direct calculation of a bandwidth estimate from the arrival times of packets. This technique has the great advantage of not requiring synchronised clocks. A disadvantage is a possibly higher intrusiveness, see Section 4.2.

3.5 Variable Packet Size (VPS)

3.5.1 The basics of the Variable Packet Size technique

The Variable Packet Size (VPS) technique [26] was first used in the pathchar tool [29] to estimate per-link capacity. The basic idea is that the tool will send a packet in which the TTL (time-to-live) field in the IP header is set to a specific value. Each layer 3 device will decrease this TTL. When the TTL is zero, the device will send an ICMP TTL-exceeded packet back to the source. Upon receiving this ICMP packet, the tool can estimate the RTT (round-trip time). The estimation of RTT is done multiple times for various packet

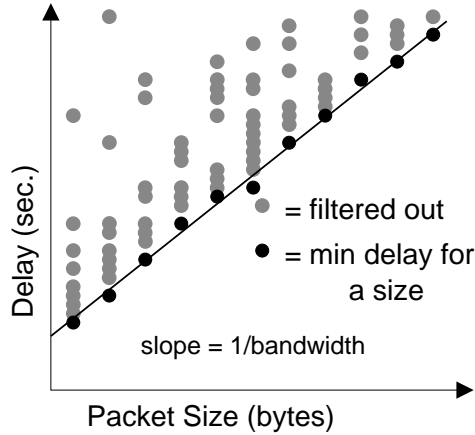


Figure 3.3: Linear regression on minimum packet delays for one link. [10]

sizes. Hopefully, some packet-reply combinations will return to the source without being queued anywhere. These will represent the minimum RTT for each of the various packet sizes. When the minimum RTTs are plotted against their respective packet sizes, the link bandwidth capacity can be calculated from the slope (RTT divided by packet size) of this graph. The capacity of the first link is $\hat{C}_1 = 1/\beta_1$, where β_1 is the slope of the RTT graph [26]. This slope is the observed delay divided by the packet size and equals the inverse of the bandwidth capacity. For each of the following links, the slope equals the sum of all capacity estimates of all the previous links. This means that links farther along the path correspond to larger values of the slope which is cumulative, because the RTT increases with each hop. The bandwidth capacity at a certain link k can now be calculated by subtracting the slope of the previous link $k-1$ from the slope of link k . This results in the following equation:

$$\hat{C}_i = \frac{1}{\beta_i - \beta_{i-1}}, \quad (3.2)$$

in which \hat{C}_i is the capacity estimate for link $i = 1, 2, \dots, n$ and n is the number of links. β_i is the RTT slope of link i , for the first link $\beta_0 = 0$. So, the bandwidth capacity is the inverse of the difference in slopes between two adjacent links, see Figure 3.3. This approach is used by pathchar, ping, clink, pchar and Nettimer (which also uses a tailgating technique) [26].

3.5.2 The VPS technique and layer 2 devices

The VPS technique works fine as long as there are no layer 2 devices (like switches) present on the path being measured. If, there are layer 2 devices, these *will* cause delays by storing and forwarding packets. Layer 2 devices neither decrease the TTL, nor send ICMP

replies. Therefore, layer 2 devices are 'invisible' to all IP and higher protocols and will cause underestimates of the per-link capacity [26, 10]. There is a way to make a rough estimation of the bandwidth capacity based on an underestimate. In [26], the following equation is derived:

$$\hat{C}_1^{L3} = \frac{1}{\sum_{j=1}^{M1} \frac{1}{C_{1,j}^{L2}}} \quad (3.3)$$

\hat{C}_1^{L3} is the capacity estimate for the first link as it is calculated by the VPS based tool. It is equal to 1 divided by the observed round-trip-time slope on this link. This RTT slope depends on the number of layer 2 store-and-forward devices, which cause serialization delays. $M1$ is the number of layer 2 devices on the first link. $C_{1,j}^{L2}$ is the capacity of layer 2 device j on the first link. [26] also gives an equation for the correct value of the capacity of the first link:

$$C_1^{L3} = \min_{j=1 \dots M1} C_{1,j}^{L2}$$

This correct capacity is, in other words, the minimum capacity of all layer 2 devices on the first link. We now have a relationship between the VPS observed capacity, the actual capacity and the number of layer 2 devices. So, when measuring with a VPS based tool, we can compare its output with the true link capacity (if known) and estimate the number of layer 2 devices, which cause the underestimate reported by VPS based tools.

3.5.3 Even-Odd

This technique is used for the clink and pathchar tools and is described in [29]. It is used in addition to the VPS technique and purely a mathematical 'trick' to improve reliability. The probing technique is not altered. The basic idea is quite simple: for each of the probing sizes, divide the set of samples into two: the even and the odd numbered. Next, when applying Equation 3.2, calculate the capacity using

1. only the respective even samples
2. only the respective odd samples
3. the even samples from link i , the odd samples from link $i+1$
4. the odd samples from link i , the even samples from link $i+1$

Now use the minimum and maximum values of these four to form an interval. [29] states that when the interval is narrow (a few percent), the estimate is at least "consistent", if not accurate. Section 5.3 shows the results of our experiment. Inaccurate results may be caused by the effects discussed in Section 3.5.2 on layer 2 network devices.

3.6 Tailgating

The packet tailgating technique, as used in Nettimer for measuring bandwidth capacity, is explained in [10]. There are two phases, which together provide certain values needed to calculate the bandwidth capacity at a link where queuing takes place. [10]. The first phase is like VPS probing, but then for the entire path instead of per link. It yields path bandwidth and delay from source to destination. The second phase is the actual tailgating phase. In this phase, a very large packet (1500 bytes) is sent, followed directly by the smallest possible packet (40 bytes). To get the best results, the following conditions should be met [10]:

- the large packet should not be queued itself due to cross-traffic,
- the large packet should have a TTL field set to l ,
- the small packet should be queued directly after the large packet on link l ,
- the small packet should not be queued after having passed link l .

The first condition will probably not always be met, so observing multiple streams and filtering the results is needed to get a reliable estimate. The tailgating is repeated multiple times, with TTL set from 1 to n , where n is the length of the path (repeat until the error is lower than a certain value).

This is what happens: when arriving at a certain link, the tailgater packet will be queued behind the large packet. Now the tailgater packet gets a delay that is larger than the delay found in the first phase of the measurement, because the large packet has now been queued before it. This delay is recorded and the tailgating is repeated multiple times for each link in random order.

To calculate the bandwidth capacity of all links (at which queuing occurs), $C_{link,q}$, we can use Equation 3.4 in [10]. Calculation is done for each link, just as tailgating has been done for each link. Equation 3.4 is based on the 'Packet Pair Property' in Section 3.3, but it has been adapted to cope with the difference in size between the large packet and the small packet.

$$C_{link,q}(k) = \frac{P_{large}}{\left(t_{small,n} + \frac{P_{small} - P_{large}}{C_{link,q}^{k-1}} - \frac{P_{small}}{C_{link}^{n-1}} - t_{large,0} - d^{n-1} \right)}. \quad (3.4)$$

This equation uses the following symbols: P_{large} and P_{small} are the sizes of the large packet and the tailgater. $t_{small,n}$ is the arrival time of the tailgater packet and $t_{large,0}$ is the time the large packet was sent. $C_{link,q}^{k-1}$ and C_{link}^{n-1} are the capacities of all earlier links. d^{n-1} is the delay of all earlier links minus the queuing times and processing times, so wire-times only. The capacities and delays of earlier links are defined as:

$$\frac{1}{C_{link}^m} = \sum_{i=0}^m \left(\frac{1}{C_{link}(i)} \right) \quad \text{and} \quad d^m = \sum_{i=0}^m d(i).$$

One of the advantages of tailgating is that it doesn't use ICMP (time-exceeded), as some nodes on the path may give low priority to sending those packets. However, Nettimer does require running software on both ends of the path and the source should be able to send the tailgater packet quickly. One more practical problem arises when one link is more than $\frac{1500}{40} = 37.5$ times as fast as the previous on the path, because the large packet will then always be processed before the tailgater arrives [10].

3.7 TCP simulation

TCP simulation is used by the TReno tool. This methodology is used to measure the bulk-transfer capacity. TReno can operate in two modes [46]: UDP and ICMP. The former sends out packets with low TTL (time-to-live) and waits for ICMP time-exceeded replies to return. The latter sends ICMP ECHO requests and waits for replies. Like any other NPMT using ICMP ECHO, some routers may not respond quickly to these requests. If so, the results may be unreliable. TReno simulates TCP and uses a slow-start algorithm as described in [4]. More information on the BTC (bulk-transfer capacity) can be found in [40], which is an expired draft document. In this document we find that TReno uses a rate-halving algorithm, which limits the TCP congestion window to reduce burstiness [37]. TReno also detects the maximum segment size for the path (MTU discovery). We could find no more information on statistical properties of TReno.

3.8 Routing and topology

Since 1988, the traceroute tool by Van Jacobson is used to discover which network links data packets pass when sent to a particular host. This way, a small portion of the network topology becomes visible. Note that all packets may not necessarily follow the same path. The tools that estimate per-link capacity (like pchar) discover all the layer 3 network devices (links) on the path too, while doing their estimating. Most of them use the same technique: just send IP packets with limited TTL field and wait for ICMP-time-exceeded packets to return. Routing information can be extracted from these packets. traceroute is described in detail in [39].

The results of traceroute and bandwidth measurement tools (NPMTs) can be combined into Effective Network Views (ENV), as was explained in Section 2.4. The creation of an ENV consists of two steps (tests) and one optional third [15], but first some choices need to be made. These choices are: which network devices are to be included in the effective topology, and which is going to be the test machine? All results and the ENV are relative to the test machine. The first step is to calculate (or rather estimate) the bandwidth between the test machine and the other relevant network devices and group (or cluster) those with similar observed values together. The second step also involves measuring bandwidth, but now pairs of devices are tested at the same time. Simultaneous testing takes place between the test machine and two devices from the same group. If the two measurements seem to

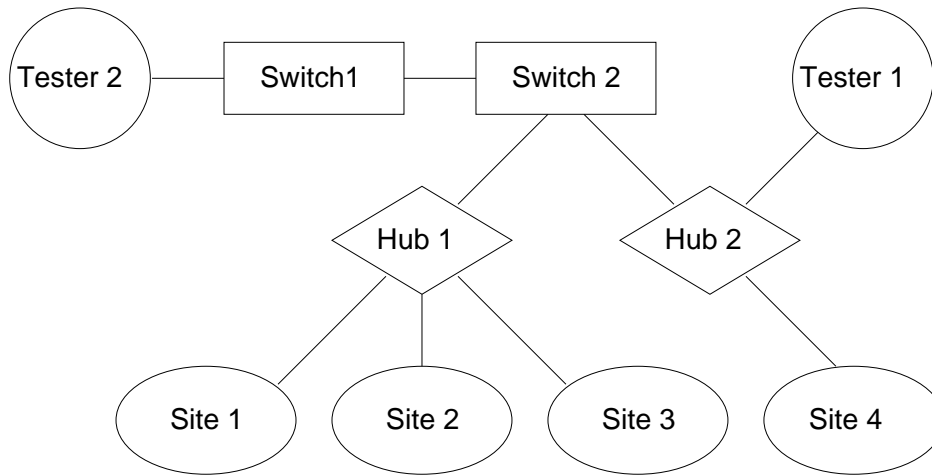


Figure 3.4: Sample physical topology

be significantly interfering with each other, the two devices probably do belong in the same group. Otherwise they should be in different groups. This is repeated for all the devices to form consistent groups. The optional test tries to locate private network resources, which we will not discuss here.

The key observation after applying ENV is that quite different effective topologies may be found, depending on the test machine. Decisions like which process should be run on which machine can be facilitated using the ENV graphical output. This is demonstrated in [15] by an example of an application which consists of a master sending packets to slaves for computation, which return their results to the master. The choice of which machine should be the master has significant influence on execution time of the whole.

Figures 3.4 and 3.6 illustrate two different effective network views, depending on where the tester is placed in the network. All links in Figure 3.4 operate at 10 Mb/s, except for Hub 2 - Site 4: 100 Mb/s. From Figure 3.5 we may conclude that an application at the location of Tester 1 can access Site 4 at much higher bandwidth. From the viewpoint of Tester 2, all communication passes through a slow hub (Figure 3.6).

3.9 Summary

A summary is shown in Table 3.2. A few remarks need to be made here. First of all, the information in Table 3.2 is extracted from manuals and research documents. Some information may not be entirely correct. For example, we have discovered that cprobe measures utilization instead of available bandwidth [19]. When we discuss the results in Section 5.3, we will try to verify that each tool measures the characteristic in Table 3.2.

The tools Pathload, pathchar and Netperf also use the TCP protocol to maintain a control channel between client and server. The control channel is kept silent during measurement with UDP [34, 25, 29].

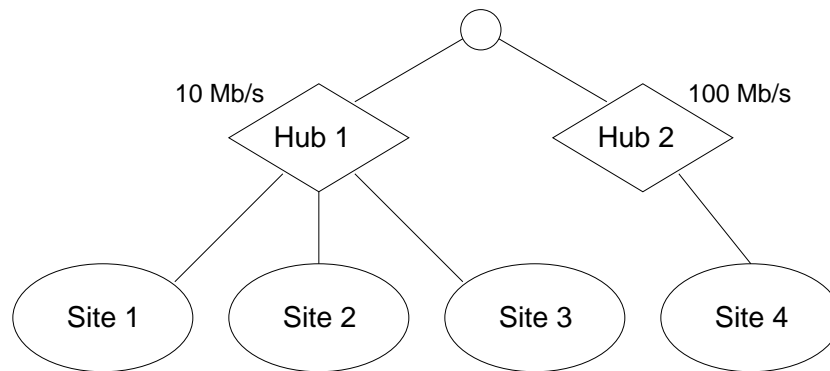


Figure 3.5: Sample effective network view from a fast tester

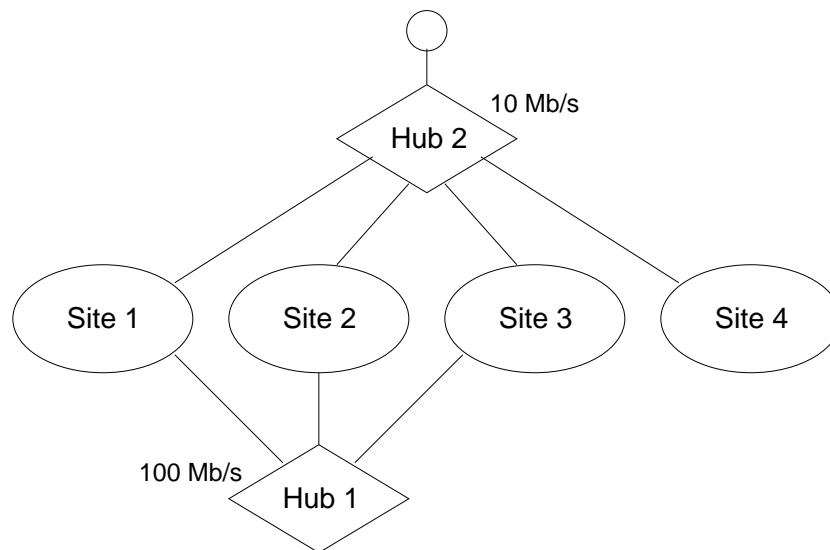


Figure 3.6: Sample effective network view from a slow tester

Table 3.2: Summary of tool properties

Tool name	Active/ Passive	Methodology	Protocol	Characteristic	Path/ Per-link
bing	active	VPS	ICMP	bandwidth capacity, loss, latency	path
bprobe	active	packet pair	ICMP	bandwidth capacity	path
cprobe	active	packet pair	ICMP	bandwidth utilization	path
clink	active	VPS/even-odd	UDP	bandwidth capacity, latency	per-link
Iperf	active	path flooding	TCP, UDP	bandwidth capacity, loss	path
netest	active	packet pair	UDP	bandwidth capacity	path
Netperf	active	path flooding	TCP, UDP	bulk transfer capacity, latency, throughput	path
Nettimer	active, passive	packet pair, VPS/tailgating	TCP	bandwidth capacity	path
pathchar	active	VPS/even-odd	UDP, ICMP	bandwidth capacity, loss, latency	per-link
Pathload	active	SLOPS	UDP	available bandwidth	path
Pathrate	active	packet pair & packet train	UDP	bandwidth capacity	path
pchar	active	VPS	UDP, ICMP	bandwidth capacity, packet loss, latency	per-link
pipechar	active	packet train	UDP	available bandwidth	per-link
sprobe	active	packet pair	TCP	bandwidth capacity	path
traceroute	active	UDP with low TTL	UDP, ICMP	topology, latency	per-link
TReno	active	TCP simula- tion	UDP, ICMP	bulk transfer capacity	path
ttcp	active	path flooding	TCP, UDP	achievable band- width	path

Chapter 4

General properties

In this chapter we will discuss general properties related to NPMTs. Three important issues in measuring performance are cooperativeness of hosts, intrusiveness caused by probing and scalability. In Section 4.1 we compare the NPMTs concerning the degree of cooperativeness they require from the hosts they work on. We derived an equation for calculating the degree of intrusiveness and present it in Section 4.2. This equation is used in the experiment described in Chapter 5. Whether or not NPMTs can be used with measurement infrastructures is discussed in Section 4.3 on scalability. Section 4.4 deals with what the NPMTs require with regard to security issues and operating systems. Finally, in Section 4.5 we discuss problems caused by cross-traffic.

4.1 Cooperativeness of environment

Each performance measurement tool requires some cooperation of the network links and end-points. To make the distinction between various degrees of cooperativeness a little more formal, we created three categories of what a location is required to provide. For each category we give an example of its use. A location of the first category is required nothing but to forward packets (no cooperation). This comes down to routing packets like any link does. Second category: the location should respond properly to some standard protocol demands (standard cooperation). We find this kind of cooperativeness often with NPMTs estimating per-link characteristics. These may need the location to send ICMP time-exceeded packets with high enough priority. Belonging to the third category means that the location should run specific non-standard software (strong cooperation). This usually comes with some kind of client-server architecture. A copy of the tool should run on both locations.

Table 4.1 shows the tools with their requirements of the intermediate links and the end host concerning cooperativeness. Of course, each tool requires strong cooperativeness from the host it runs on.

Table 4.1: Cooperation requirements

Tool name	Intermediate link cooperation			End host cooperation		
	none	standard	strong	none	standard	strong
bing	X				X	
bprobe	X				X	
cprobe	X				X	
clink		X			X	
Iperf	X					X
netest	X					X
Netperf	X					X
Nettimer		X			X	
pathchar		X			X	
Pathload	X					X
Pathrate	X					X
pchar		X			X	
pipechar		X			X	
SProbe	X				X	
traceroute		X			X	
TReno		X			X	
ttcp	X					X

4.2 Intrusiveness

Perhaps the most difficult criterion is a tool’s intrusiveness. Whether or not a tool is intrusive, can be determined subjectively by checking whether the tool causes any of the following two situations. First, does the by the NPMT generated traffic affect the present cross-traffic? And second, does the generated traffic make the observation itself unreliable?

A good objective definition of intrusiveness is very hard to find. We have considered taking a subjective approach, defining a scale like none/low/medium/high. Where ‘none’ applies to passive monitoring and ‘high’ prohibits any useful measuring at all. However, it is desirable to define intrusiveness in a more objective way. Therefore we define intrusiveness as the percentage of capacity that is consumed for the measurements, averaged over the time interval between two measurements. In short, the intrusiveness I can be calculated from Equation 4.1.

$$I = \frac{B}{C} \cdot 100\% \quad (4.1)$$

In this equation, B is the amount of traffic (bits) generated in one second and C is the nominal (true) capacity of the observed path. The nominal capacity is the theoretic-

cal maximum bandwidth the hardware can supply (over the entire path). One difficulty when calculating intrusiveness is which value to use for bandwidth used: some tools run a fixed amount of time (specified in advance), some run until a certain confidence level is reached, others require the user to stop the measurement manually. We have tried to get a reasonably good measurement result and then see how much time and bandwidth it took.

Intrusiveness is a known problem in other areas as well. Considering Grid applications, [12] suggests to limit the amount of messages sent. This can be done by implementing smart and scalable algorithms and protocols.

4.3 Scalability

In this document, we only compare stand-alone performance measurement tools, leaving out the tools which are integrated in larger packages (Internet measurement infrastructures). Some of these are NPACI's Network Weather Service (NWS) [44], CAIDA's Coral-Reef [21], NLANR's Network Analysis Infrastructure (NAI) [41] and more, see [16]. The monitoring and probing parts (sensors) of measurement infrastructures ideally have the feature of being part of a scalable system. This means that protocols and algorithms are supposed to work for small and large-sized systems. There are many possible problems related to scalability. For example: do the used protocols get slow when the problem size increases; will there be much intrusiveness when the system gets bigger?

From the documentation of most NPMTs it appears that scalability was not much of a design issue. The tools can be used as a sensor in a measurement infrastructure, but often not more than one instance of the same tool can run at any given time. Which NPMT to choose as a sensor requires some careful consideration. As we will show in Chapter 5, not all NPMTs produce accurate results when tested on the network configuration we used. But if a reliable and accurate tool is found, scalability and intrusiveness should not be ignored.

4.4 Network requirements

4.4.1 Security

None of the tested tools has documented any effort to prevent abuse. However, security may become rather important when applying NPMTs in computational grids. An example of such is the Globus toolkit. A security architecture for Globus is described in [31] and focuses on "authentication of users, resources and processes (...)" and access control. A security architecture is needed in grid environments, because grids are very dynamic and sites use different protocols and mechanisms (see [32] and [31] for more information).

When using a tool that measures characteristics related to achievable bandwidth (throughput, BTC), it usually sends as much data as it can. To a system operator this may appear to be a denial-of-service attack.

Most of the tools require to be run and/or installed as root. The main reason being that they require a raw socket for IP. If running as root or installing as root is required, then this is shown in Table 4.2 in the column 'Security requirements' as the entry 'root'. There are two tools that don't require root access: pathchar and Pathload. These both work as a client-server system, don't need to be installed with root privileges and can be run by standard users.

4.4.2 Operating systems and portability

All of the tools were tested in a Linux environment. Most of them will also run on SunOS, FreeBSD, some on MSWindows, some on other (specialized/research) operating systems. Table 4.2 lists the operating systems on which the tool should be able to run as stated in the product documentation. Although some of the tools do not have Linux in their OS list, all of them do run on our Linux system. The summary table also shows for each tool whether pre-compiled binaries are available or not.

4.5 Cross-traffic

Most NPMTs use active probing. They work with inter-arrival times of packets in a train, round-trip times, one-way delays or other timing observations. Especially when using packet trains, cross-traffic packets may interfere and cause unreliable results. This is explained in Section 3.3.

There may be several reasons for wanting to use a NPMT. Two examples are: "What is the capacity of this path?" and "What is the available bandwidth when the network load is high?" To create reliable results, the first question can be answered by measuring capacity when load is low (at night, for example). Bandwidth capacity is best measured with little cross-traffic because the packet dispersion technique works better if no cross-traffic packets interfere. The second question is a lot more difficult. Not only is available bandwidth a highly dynamic characteristic, but it needs to be measured when load is high. This means that available bandwidth can only be measured in a reliable way when there is cross-traffic.

We conclude that observations should take place at the proper time of day: either when there is little or much cross-traffic, depending on whether a static or dynamic characteristic is measured.

Table 4.2: Network requirements

Tool name	Security requirements	OS list	Pre-compiled?
bing	root	FreeBSD, FreeBSD, Linux, NetBSD, SunOS, AIX, HP-UX, Solaris, BSDI, Ultrix (DECstation), OSF/1	no
bprobe	root	IRIX	no
cprobe	root	IRIX	no
clink	root	linux only	no
Iperf	none	UNIX, Windows	yes
netest	root	FreeBSD, Linux, Solaris, AIX, IRIX	no
Netperf	root	Linux, FreeBSD, Solaris	yes
Nettimer	root	Linux	no
pathchar	root	freeBSD, netBSD, openBSD, Linux, Solaris	yes
Pathload	none	Linux, Solaris	yes
Pathrate	none	Linux, Solaris	yes
pchar	root	Linux, Solaris, IRIX, FreeBSD, NetBSD, OpenBSD	no
pipechar	root	FreeBSD, Linux, Solaris, AIX, IRIX	no
SProbe	root	Linux, FreeBSD	no
traceroute	root	Linux, FreeBSD, Solaris, MSWindows	part of OS
TReno	root	Unix, Linux	no
ttcp	root	MSWindows, Linux, FreeBSD, Solaris, SunOS, AIX, HP-UX, IRIX	part of OS

Chapter 5

The experiment

This chapter describes the experiment we held to test the NPMTs discussed in the previous chapters. Section 5.1 provides details about the testing environment and Section 5.2 explains how we performed the tests. In Section 5.3 we present the results and discuss them. Finally, Section 5.4 shows things that went wrong during the experiment and provides an explanation when possible.

5.1 Environment

The experiments have been run using the DAS-2 (Distributed ASCI Supercomputer). ASCI is the Advanced School for Computing and Imaging [1]. Our comparison of network performance measurement tools has taken place between the file server nodes of the DAS-2 clusters at two universities in the Netherlands: Vrije Universiteit, Amsterdam and Technische Universiteit, Delft. Each DAS-2 cluster runs the Linux operating system. From the website [23] we have taken the following hardware information:

“DAS-2 is a wide-area distributed computer of 200 Dual Pentium-III nodes. The machine is built out of clusters of workstations, which are interconnected by SurfNet, the Dutch university Internet backbone for wide-area communication, whereas Myrinet, a popular multi-Gigabit LAN, is used for local communication” [23].

The cluster at the VU has a gigabit connection to SurfNet, and all the other clusters have a 100Mbit connection. This means that, in theory, any bandwidth capacity measurement should result in 100Mbit. The SurfNet core has 10Gbit connections.

Using traceroute, we discovered the route used by packets to travel from the VU cluster to the TUDelft cluster and back. The results are shown in Table 5.1. This table clearly shows that the paths are not symmetrical. Therefore, observations based on round-trip methodologies may lead to false conclusions regarding these two paths.

Table 5.1: Results from traceroute.

Domain	VU to TUDelft	TUDelft to VU (read from bottom)
VU 	fs0.das2.cs.vu.nl colossus.cs.vu.nl 130.27.14.1 hka16-1-d01.backbone.vu.nl	fs0.das2.cs.vu.nl colossus.cs.vu.nl - wsk71-2-d01.backbone.vu.nl
SURFnet 	Gi15-2-27.AR5.Amsterdam1.surf.net PO6-0.CR1.Amsterdam2.surf.net PO1-0.AR5.Delft1.surf.net tudelft-router.Customer.surf.net	vu-router.Customer.surf.net PO0-0.AR5.Amsterdam1.surf.net PO10-0.CR1.Amsterdam1.surf.net Gi15-0.AR5.Delft1.surf.net
TUDelft 	dunet1.router.tudelft.nl dunet3.its.tudelft.nl fs3.das2.its.tudelft.nl	dunet1.router.tudelft.nl dunet3.its.tudelft.nl fs3.das2.its.tudelft.nl

'-' Indicates this host is not present on the path, which is one hop shorter.

5.2 Experiment design

All the tools have been about run multiple times to get consistent results. Tests have been run in the evenings, when there is less cross-traffic, and during office hours. Most of the tools require some sort of user input, for example how long the test should run. Whenever possible, default options were favored, except when those did not provide a realistic bandwidth estimate. Some tools require a lot of manual tuning, so one must be careful when drawing conclusions from the test results.

To monitor the amount of traffic generated by each tool, we used the ifconfig program. This program is part of the operating system and provides the amount of sent and received bytes. To determine the amount of traffic generated by each tool, we subtract the amount of sent bytes before running the tool from the amount of used bytes reported after running the tool. Then we also subtract the average network traffic. We use the result from this calculation to determine the degree of intrusiveness I in Equation 5.1.

$$I = \frac{B_{run}}{t_{run}} \cdot \frac{1}{C} \cdot 100\% \quad (5.1)$$

In this equation, B_{run} is the amount of bits used during one run of the NPMT. t_{run} is the amount of time consumed by one run. C is the theoretical capacity of the path (100 Mb/s in our case). This gives us a percentage of capacity used.

The results include running times, generated traffic, observed bandwidth, and intrusiveness. The results of our measurements are all averages. If, for some reason, a tool shows very inconsistent results on different runs, we will provide an interval instead of a single average value.

Table 5.2: Measurement results (end-to-end observations)

Tool name	Running times	Generated traffic estimate	Bandwidth type	Observed bandwidth	Intrusiveness
bing	37 s	6.9 MB	bandwidth capacity	8 Mb/s	1.5 %
bprobe	0.3 s	0.19 MB	bandwidth capacity	22 Mb/s	5 %
clink	58 min	130 MB	bandwidth capacity	19 Mb/s	< 0.5 %
netest	71 s	53 MB	bandwidth capacity	91 Mb/s	6,0 %
Nettimer	1.2 s	0.29 MB	bandwidth capacity	98 Mb/s	1.9 %
Pathrate	73 s	1.8 MB	bandwidth capacity	99 Mb/s	< 0.5 %
pchar	100-120 min	70-80 MB	bandwidth capacity	20 Mb/s	< 0.5 %
Iperf	30 s	356 MB	bulk transfer capacity	94 Mb/s	95 %
Netperf	100 s	1.2 GB	bulk transfer capacity	89 Mb/s	94 %
TReno	13 s	5.9 MB	bulk transfer capacity, available bandwidth	4,1 Mb/s	3.8 %
Pathload	8.4 s	9.1 MB	available bandwidth	98 Mb/s	8,7 %
pipechar	34 s	0.85 MB	available bandwidth	13 Mb/s	< 0.5 %
ttcp	1.5 s	17 MB	throughput	88 Mb/s	89 %
cprobe	0.1 s	0.15 MB	utilization	10 Mb/s	12 %

5.3 Results

Table 5.2 shows a summary of the observed NPMT output. We have included the bandwidth type to be able to compare the measured bandwidth results. The intrusiveness is not reported by the NPMTs, but calculated using Equation 5.1.

Some remarks need to be made concerning the results table. To find the path characteristic for clink, pchar and pipechar, we need to take the minimum value found on the observed path. This particular value is shown in Table 5.2. Also, the value used is a 'realistic' minimum, the values below zero are discarded. The detailed results of the observations by these three tools can be found in Table 5.3 and Table 5.4.

5.3.1 Running times

There is quite some difference in the amount of time that is consumed by one run of the tools. Two tools produce their results very fast: bprobe and cprobe need only a few tenths of seconds. Two others take a long time to report their estimate: clink and pchar. The difference can be explained by the techniques these tools use. cprobe and bprobe use only a few packet pairs, whereas clink and pchar send many packets of different sizes to each of the intermediate links.

Of course, an interesting issue here is whether these long running tools produce any better results than the fast ones? Well, `clink`, `pchar` and `bprobe` estimate the bandwidth capacity. On average, the results are respectively 19 Mb/s, 20 Mb/s and 22 Mb/s for the entire path. These values do not correspond to the true capacity of 100 Mb/s. If we apply the theory about layer 2 devices presented in Section 3.5.2, the results can be explained. This is done in Section 5.3.3. We notice that values are pretty much the same. When only interested in end-to-end results, the fastest running tool may seem the obvious choice.

The tool documentation may give an indication of how long a tool should run. For example, `netest` is said to be finished after about one minute [42] on a 'healthy' network connection, which is confirmed by Table 5.2. Some of the NPMTs allow the user to control how long the tool runs: `bing`, `Iperf`, `netest`, `TReno` and `ttcp`.

5.3.2 Generated traffic estimate

As with the differences in time, there are also many differences in the amount of bytes sent per run by the tools. The most bytes are sent by `Netperf`. It sends more than 1 Gigabyte. The three tools that measure by sending as much data as possible are `Iperf`, `Netperf` and `ttcp`. That is caused by the way they measure the BTC. The amount of data these NPMTs will send is limited only by the available bandwidth and the time they run. This strategy of sending as much data as possible has its problems, because it puts an enormous load on the links. Common sense tells us that this certainly affects the ongoing cross-traffic. There seems to be another way to estimate the BTC. This is demonstrated by `TReno`, explained in Section 3.7. The question remains why `TReno` produces such strange results.

5.3.3 Bandwidth type & Measured bandwidth

End-to-end estimates

When measuring bandwidth, the tools should show nearly equal results. However, this is by far not the case, although some differences have easy explanations. First we need to see which tools give an accurate estimate. If the bandwidth type is capacity, then the result should be about 100 Mb/s for the given path. There is the big difference between tools using ICMP and tools using TCP/UDP protocols. We see that the typical result for an ICMP tool about 20 Mb/s and for a TCP/UDP tool, the result is just below 100 Mb/s. The latter show far better results, because the former underestimate bandwidth due to 'invisible' layer 2 devices. This was explained in Section 3.5.2.

The results for available bandwidth are a bit harder to explain. If we take a closer look at the output of `pipechar`, we see a big difference in results, depending on the direction in which we measure. When measuring from VU to TUDelft, the reported utilization is mostly 70%. In the opposite direction, the utilization at the same time is typically below 20%. Although the path is not symmetrical, this much difference is unexpected. `cprobe` reports about 10 Mb/s, but from [19] it is a bit unclear whether `cprobe` reports the utilization (congestion) or available bandwidth. Assuming utilization, the result of

available bandwidth is actually 90 Mb/s. And finally, we have Pathload, which reports over 97 Mb/s.

We compare these numbers to the results produced by tools that use the bulk-transfer capacity, because these show how much bandwidth can actually be used. Here we see that `ttcp` reports 11 MB/s (in both directions) and `Netperf` reports 89 Mb/s (in both directions). So we may safely say that the BTC is just below 90 Mb/s. As stated in Section 2.3.3, there are some differences between BTC and available bandwidth. When comparing the results of `cprobe` with the BTC tools, we see that these are very much the same. The `Nettimer` tool produces four different results of bandwidth: UDP and TCP, both with single and multiple streams. The bandwidth capacity results for UDP are both 91 Mb/s on all runs. TCP single is about 89 Mb/s, the TCP multiple output is: “not recommended”. Only Pathload and `Nettimer` claim a much higher amount of available bandwidth. [2] claims packet pair NPMTs often underestimate or overestimate the capacity on high-speed networks.

per-link estimates

Table 5.3 shows the bandwidth measured by `clink`, `pchar` and `pipechar` for each link on the chosen path from the DAS-2 cluster VU (Amsterdam) to TUDelft (Delft). This path is not symmetrical, which may cause wrong observations.

Probably because of new hardware technology used by SURFnet, not one of the tools produces any result near the true value. The only useful values are with links 0-1 and 8-9, which are the endpoints of the path. Using Equation 3.3, we can make an estimate of how many layer 2 devices (the $M1$ variable) are probably present on the path. First we need to know the capacity of any layer 2 devices present on each link: $C_{1,j}^{L2}$. This is a problem, because we don't know anything about them. Let us assume they all have the same capacity as the link on which they are. We know that the SURFnet core consists of Gigabit links, VU has 100Mbit to local machines and a Gigabit ethernet connection to SURFnet and TUDelft has 100Mbit to SURFnet. So, on link 0-1 we should observe 100 Mb/s and on link 8-9 the same. This reduces our equation to

$$\hat{C}_1^{L3} = \frac{1}{\sum_{j=1}^{M1} \frac{1}{100}},$$

which may be rewritten as $M1 = 100/\hat{C}_1^{L3}$. When substituting the observed bandwidth capacity of about 28 Mb/s, we find that there are probably three or four layer 2 devices are present on link 0-1. This shows that for most links, a useful estimate is hard to make. It is nearly impossible to tell how fast each link really is, based on the outcome of these NPMTs. If, for example, there is a device working with parallel streams of data, then the capacity measurements will probably neither detect this, nor report a result that equals the sum of all streams.

For many links on the path, the capacity measured by `clink` and `pchar` shows up as a negative number. `Clink` actually reports these numbers, whereas `pchar` reports nothing. `pipechar` estimates available bandwidth, but does report an estimate of the capacity of each

Table 5.3: Per-link results from VU to TUDelft

i	link (links are [i, i+1])	bandwidth (Mb/s)		
		clink	pchar	pipechar
0	localhost	28	28.46	28.5
1	colossus.cs.vu.nl	N	N	13
2	no name	V	V	13
3	hka16-1-d01.backbone.vu.nl	N	N	13
4	Gi15-2-27.AR5.Amsterdam1.surf.net	20	19.77	12
5	PO6-0.CR1.Amsterdam1.surf.net	N	V	13
6	PO0-0.AR5.Delft1.surf.net	N	N	13
7	dunet1.router.tudelft.nl	N	N	13
8	dunet3.router.tudelft.nl	19	29.63	13
9	fs3.das2.its.tudelft.nl			

'N' Means that the result is (nearly always) negative. 'V' Means that the results on that link differ so much that a useful estimate could not be made. These values can be anywhere between 50 Mb/s and 3 Gb/s.

endpoint, both being about 100 Mb/s. Knowing part of the architecture of the SURFnet nodes enables us to conclude that none of the per-link estimating NPMTs produces useful results. Comparing these per-link NPMTs has been done before. [26] shows us that when using 100Mbit and 10Mbit links, results may be quite good. So these bad results may be caused by the fact that these tools have been existing for some time and therefore are not adapted to high speed network hardware. This is confirmed by [2, 11], which say that the VPS and packet pair techniques are accurate for paths with capacity below 100Mbit.

5.3.4 Observed intrusiveness

The last column in Table 5.2 contains the NPMTs intrusiveness. There are three tools scoring particularly good here. These are clink, Pathrate and pipechar. There are some tools that seem to be highly intrusive and thus are competing for bandwidth with the ongoing cross-traffic. These are Iperf, Netperf and ttcp. Of course, we need to make a few remarks here.

First about the tools scoring very good. Because the intrusiveness is based on elapsed time and generated traffic, it is easy to make intrusiveness low by either sending for a very long time or generate very little traffic. It is obvious that clink demonstrates the former method. clink needs to run for about an hour to produce output. The total of all traffic sent by clink is also larger than the total traffic sent by Pathrate and pipechar.

In [11] about the Nettimer tool is said that the tool has been tested on connections up to 100 Mb/s. This may explain the disappointing results of the passive monitoring runs on

Table 5.4: Per-link results from TUDelft to VU

i	links from TUDelft to VU (links are [i, i+1])	bandwidth (Mb/s)		
		clink	pchar	pipechar
0	fs3.das2.its.tudelft.nl	27.5	27.7	96.1
1	dunet3.its.tudelft.nl	285	273	99.3
2	dunet1.router.tudelft.nl	55.0	53.6	151
3	Gi15-0.AR5.Delft1.surf.net	35	N	151
4	PO10-0.CR1.Amsterdam1.surf.net	N	N	151
5	PO1-0.AR5.Amsterdam2.surf.net	N	N	153
6	vu-router.Customer.surf.net	908	1108	151
7	wsk71-2-d01.backbone.vu.nl	25.7	25.8	44.4
8	colossus.cs.vu.nl	818	N	13.0
9	fs0.das2.cs.vu.nl			

'N' means that the result is (nearly always) negative. 'V' means that the results on that link differ so much that a useful estimate could not be made. These values can be anywhere between 50 Mb/s and 3 Gb/s.

gigabit links. The same document reports the intrusiveness of Nettimer as always being below 7% (of the measured bandwidth) on their experiments.

Now we discuss Iperf, Netperf and ttcp. These seem to disrupt all cross-traffic by sending as much data as they can. This is not the only way to measure the bulk-transfer capacity. In [38] we find that TReno can even use only some ICMP packets to estimate the BTC. Calculating intrusiveness when the tool is trying to be as intrusive as possible, is not very useful. However, one way to use this intrusiveness is to compare it to the measured BTC. For ttcp and Iperf, the values are almost identical (about one percent difference, rounded off). The Netperf tool seems to have generated slightly more traffic than its reported BTC.

5.4 Bugs, crashes, failures

This is a list of things we encountered during the testing of NPMTs. The problems are probably caused by any of the following three reasons. First, the NPMT may be incompatible with the physical network devices. Second, the software environment on the host where the NPMT was running gave problems. And finally, some failures are caused by bugs or features of the tools themselves.

- SProbe could not be run due to an incompatible firewall, of which the configuration could not be changed to fit the needs of SProbe.

- TReno produces results only in ICMP mode and even then regularly fails. A closer look at TReno output shows that in UDP mode, the tool encounters 100% packet loss. Maybe the reported packet loss is caused by the firewall configuration, which is the reason SProbe could not run.
- After a long time of trial and error, we have found a working set of options for active probing with Nettimer. This configuration gives reasonable results in 75% of the tests and absurdly high (1.2 Gb/s) results the remaining 25% on the path from VU to TUDelft.
- Nettimer works only in one direction (from VU to TUDelft), because in the other direction there is a difference in link capacity from 100Mb/s -> 10Gb/s, which is a factor of 100. The maximum difference factor the Nettimer algorithm can handle is $1500/40=37.5$. This is calculated by dividing the MTU (maximum transfer unit) by the size of the tailgater (40 bytes).
- cprobe fails to give an estimate from TUDelft to VU other than 0.0 Mb/s. When observing the opposite path, a good result is found. We do not have an explanation for this behaviour.
- The documentation of pipechar claims multiple instances of the tool running on the same path will make each other crash. This happens because packets meant for one instance of the tool may be intercepted and wrongly interpreted by another instance of the same tool. This probably holds for any tool producing packets of which cannot be distinguished to which instance they belong.

Chapter 6

Conclusions

6.1 Conclusions

Our comparison of NPMTs has taken place in an environment featuring high speed network hardware. From reading the NPMT documentation, we knew that some of the tools would not be able to produce a useful result on links with a bandwidth capacity of more than 100 Mb/s. We included NPMTs that measure bulk-transfer capacity. The good thing about BTC is that it reflects the amount of bytes that can really be sent. We can take the BTC as a starting point for our comparison. Iperf, Netperf and ttcp all use a 'path flooding' algorithm conforming to the BTC. If we compare the results of these to the results of Pathload (available bandwidth), we see that Pathload probably overestimates its characteristic. This in contrast to cprobe, which reports 10% utilization and that is quite a realistic value compared to the BTC.

A lot of the compared NPMTs were to measure the path or per-link capacity. Results show that Pathrate provides a very good estimate for the path. The per-link tools clink and pchar may also get it right if taken into account that there are three to five layer 2 devices on the links containing the endpoints. The problem with the layer 2 devices probably also holds for bprobe because it uses the same protocol (ICMP) and then this tool also gives a good estimate if five layer 2 devices are present.

The results of our experiment show very inaccurate bandwidth estimation by NPMTs that operate on per-link basis using the VPS methodology. As other research has shown that high-speed networks may be a problem for packet pair tools, the same probably holds for the tools using the VPS methodology.

Finally, some remarks on using the NPMTs in grid environments. Most of the NPMTs do not allow for simultaneous measurements. A measurement infrastructure should use some kind of algorithm to avoid measurements taking place in parallel. About intrusiveness: based on our results, there are roughly three categories. These are: hardly any intrusive behaviour: typically below 0.5%. Some intrusive behaviour: from 1% to 12%. And full intrusive behaviour: about 90% to 95% of the true capacity, which means a little above 100% of available bandwidth because cross-traffic is being disrupted. The best tools to use

as sensors for a measurement infrastructure should be of the first category to reduce network load. The problem is to find a tool that is both accurate in the specific environment, fast and is not very intrusive. We will not give an advice here on which tool to use, because any such advice would depend on the required characteristic and the working environment.

6.2 Future work

Our study shows that when a network performance measurement tool is needed for a monitoring task, it should first be tested in the relevant environment. This may mean that the tool will not work on high-speed networks. A common problem is that most of the tools have been experimental implementations of a newly developed algorithm that still had some flaws. Also, a network performance measurement tool to be developed in the future should perhaps be applicable in grid networks. This requires being as little intrusive as possible and either able to run in multiple instances at once or sequentially if the time to produce an observation is short enough. Future studies may lead to the development of new NPMTs that can work in any environment and that are based on clear and formal definitions of network characteristics.

Bibliography

- [1] “ASCI home page”, <http://www.asci.tudelft.nl/>
- [2] D.A. Agarwal, B.R. Crowley, G. Jin, G. Yang, “Network Characterization Service (NCS)”, in proceedings of the 10th IEEE Symposium on High Performance Distributed Computing HPDC-10 August 2001, <http://www.didc.lbl.gov/papers/NCS.HPDC01.pdf>
- [3] M. Allman, M. Mathis, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", Request for Comments: 3148, 2001, <http://www.ietf.org/rfc/rfc3148.txt>
- [4] M. Allman, V. Paxson, W. Stevens, “TCP Congestion Control”, Request for Comments: 2581, 1999, <http://www.ietf.org/rfc/rfc2581.txt>
- [5] G. Almes, J. Mahdavi, M. Mathis, V. Paxson, “Framework for IP Performance Metrics”, May 1998, <http://www.ietf.org/rfc/rfc2330.txt>
- [6] G. Almes, S. Kalidindi, M. Zekauskas, “A One-way Delay Metric for IPPM”, Request for Comments: 2679, 1999, <http://www.ietf.org/rfc/rfc2679.txt>
- [7] G. Almes, S. Kalidindi, M. Zekauskas, “A Round-trip Delay Metric for IPPM”, Request for Comments: 2681, 1999, <http://www.ietf.org/rfc/rfc2681.txt>
- [8] G. Almes, S. Kalidindi, M. Zekauskas, “A One-way Packet Loss Metric for IPPM”, Request for Comments: 2680, 1999, <http://www.ietf.org/rft/rfc2680.txt>
- [9] M. Baker, K. Lai, “Measuring Bandwidth”, in proceedings of IEEE INFOCOM '99 March 1999 <http://mosquitonet.stanford.edu/publications/nettimer.ps.gz>
- [10] M. Baker, K. Lai, “Measuring Link Bandwidths Using a Deterministic Model of Packet Delay”, to appear in ACM SIGCOMM 2000, <http://mosquitonet.stanford.edu/~laik/projects/nettimer/publications/sigcomm2000/deterministic.pdf>
- [11] M. Baker, K. Lai, “Nettimer: A Tool for Measuring Bottleneck Link Bandwidth”, 3rd USENIX Symposium on Internet Technologies and Systems March 2001, http://mosquitonet.stanford.edu/~laik/projects/nettimer/publications/usits2001/a_tool_for_measuring_bottleneck_link_bandwidth.pdf

- [12] H.E. Bal, T. Kielmann, K. Verstoep, "Fast Measurement of LogP Parameters for Message Passing Platforms", 4th Workshop on Runtime Systems for Parallel Programming (RTSPP) pp. 1176-1183 held in conjunction with IPDPS 2000, May 2000, <http://www.cs.vu.nl/~kielmann/papers/rtspp00.ps.gz>
- [13] Z. Balaton, P. Kacsuk, N. Podhorszki, F. Vajda, "Comparison of Representative Grid Monitoring Tools", 2000
- [14] "Bandwidth pING", 1995, <http://www.cnam.fr/reseau/bing.html>
- [15] F. Berman, G. Shao, Rich Wolski, "Using Effective Network Views to Promote Distributed Application Performance", International Conference on Parallel and Distributed Processing Techniques and Applications June 1999 , <http://www-cse.ucsd.edu/~gshao/papers/pdpta99.pdf>
- [16] "Caida – Internet Measurement Infrastructure", <http://www.caida.org/analysis/performance/measinfra/>
- [17] "Caida – Performance Measurement Tools Taxonomy", <http://www.caida.org/tools/taxonomy/performance.xml>
- [18] B. Carter, "BPROBE and CPROBE – network probe tools home page ", <http://cs-people.bu.edu/carter/tools/Tools.html>
- [19] R.L. Carter, M.E. Crovella, "Measuring bottleneck link speed in packet-switched networks", Slightly modified version appeared in Performance Evaluation, Vol 27&28, 1996, <http://www.cs.bu.edu/faculty/crovella/paper-archive/TR-96-006/96-006-measuring-bottleneck-link.ps>
- [20] K.C. Claffy, S. McCreary, "Internet measurement and analysis: passive and active measurement", 1999, <http://www.caida.org/outreach/papers/1999/Nae4hansen/>
- [21] "CoralReef", <http://www.caida.org/tools/measurement/coralreef>
- [22] L. Cottrell, R. Hughes-Jones, T. Kielmann, B. Lowekamp, M. Swany, B. Tierney, "A Hierarchy of Network Measurements for Grid Applications and Services", (working document), for discussion at the sixth Global Grid Forum workshop, October 2002, <http://www-didc.lbl.gov/NMWG/measurements.pdf>
- [23] "The Distributed ASCI Supercomputer 2 (DAS-2)", <http://www.cs.vu.nl/das2>
- [24] C. Dovrolis, M. Jain, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput", to appear on the ACM SIGCOMM Conference August 2002, <http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/>
- [25] C. Dovrolis, M. Jain, "Pathload: a measurement tool for end-to-end available bandwidth", in proceedings of the 3rd Passive and Active Measurements Workshop March 2002, <http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/>

- [26] C. Dovrolis, B.A. Mah, R.S. Prasad, "The effect of layer-2 store-and-forward devices on per-hop capacity estimation", to appear at ACM Internet Measurement Workshop November 2002, <http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/>
- [27] C. Dovrolis, D. Moore, P. Ramanathan, "What do packet dispersion techniques measure?", proceedings of the 2001 Infocom, <http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/Papers/infocom01.ps>
- [28] A.B. Downey, "clink documentation", 1999, <http://rocky.wellesley.edu/downey/clink/clink.doc>
- [29] A.B. Downey, "Using pathchar to Estimate Internet Link Characteristics", to appear in ACM SIGCOMM '99, <http://rocky.wellesley.edu/downey/clink/downey.ps.gz>
- [30] A. Feldmann, P. Huang, W. Willinger, "A non-intrusive, wavelet-based approach to detecting network performance problems", to appear at ACM Internet Measurement Workshop November 2001, <http://www.tik.ee.ethz.ch/~huang/publication/wind-imw01.pdf>
- [31] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecture for Computational Grids", in proceedings of the 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998, <ftp://ftp.globus.org/pub/globus/papers/security.pdf>
- [32] "The Globus project", www.globus.org
- [33] S.D. Gribble, S. Saroiu, P.K. Gummadi, "SProbe: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments", submitted to Infocom 2002, 2001, <http://sprobe.cs.washington.edu/sprobe.ps>
- [34] R. Jones, "Netperf: A Network Performance Benchmark, Revision 2.0", 1995, <http://www.netperf.org/netperf/training/netperf.ps>
- [35] K. Keahey, V. Welch, "Fine-Grain Authorization for Resource Management in the Grid Environment", in proceedings of Grid2002 Workshop 2002
- [36] G. Kessler, S. Shepard, "A Primer On Internet and TCP/IP Tools and Utilities", Request for Comments: 2151, 1997, <http://www.ietf.org/rfc/rfc2151.txt>
- [37] K. Lahey, J. Mahdavi, M. Mathis, J. Semke, "The Rate Halving Algorithm for TCP Congestion Control", draft June 1999, http://www.psc.edu/networking/rate_halving.html
- [38] J. Mahdavi, M. Mathis, "Diagnosing Internet Congestion with a Transport Layer Performance Tool", in proceedings of INET'96 June 1996

- [39] G. Malkin, "Traceroute Using an IP Option", Request for Comments: 1393, 1993, <http://www.ietf.org/rfc/rfc1393.txt>
- [40] M. Mathis, "TReno Bulk Transfer Capacity", internet draft, expired August 1999, <http://www.advanced.org/IPPM/docs/draft-ietf-ippm-treno-btc-03.txt>
- [41] "Network Analysis Infrastructure (NAI)", <http://moat.nlanr.net/infrastructure.html>
- [42] "Network Characterization Service (NCS) - Network Test (netest)", <http://www.itg.lbl.gov/~jin/network/netest.html>
- [43] "Network reference", <http://nw.uwplatt.edu/network/reference/terms/tfs.html>
- [44] "Network Weather Service", <http://nws.cs.ucsb.edu>
- [45] S. Parker, C. Schmechel, "Some Testing Tools for TCP Implementors", Request for Comments: 2398, 1998, <http://www.ietf.org/rfc/rfc2398.txt>
- [46] "The PSC Treno Server", http://www.psc.edu/networking/treno_info.html
- [47] A.S. Tanenbaum, "Computer Networks", third edition, Prentice Hall 1996, pages 28-32
- [48] K. White, "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", Request for Comments: 2925, 2000, <http://www.ietf.org/rfc/rfc2925.txt>