

ISBN978-4-916128-07-2
C3055 ¥2800E

定価：2,800円＋税



9784916128072



1923055028005



Active Network Measurement Theory, Methods, and Tools

Yoshiaki Tanaka and Marat Zhanikeev

The ITU
Association of
Japan, Inc.

Active Network Measurement

Theory, Methods, and Tools

**Yoshiaki Tanaka
and
Marat Zhanikeev**

The ITU Association of Japan, Inc.

Active Network Measurement

Active Network Measurement

Theory, Methods, and Tools

**Yoshiaki Tanaka
and
Marat Zhanikeev**

The ITU Association of Japan, Inc.

Yoshiaki Tanaka is a professor at Global Information and Telecommunication Institute, Waseda University. Marat Zhanikeev is an assistant professor at School of International Liberal Studies, Waseda University.

Active Network Measurement
Theory, Methods, and Tools

Published by The ITU Association of Japan, Inc.
1-8-6 Kaji-cho, Chiyoda-ku, Tokyo, 101-0044 Japan
Phone: +81 3 5207 5711 Fax: +81 3 5207 5731

Copyrighted by Yoshiaki Tanaka and Marat Zhanikeev. All rights reserved.
No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording or information storage and retrieval) without prior permission in writing from the authors and the publisher. Users are not permitted to mount the content of this book as a file on any network server.

Printed in Japan

First edition, 2009

ISBN 4-916128-07-2 C3055 ¥2800E

www.ituaj.jp

Contents

Preface.....	ix
Chapter 1 NGN Standardization and QoS	1
1.1 NGN in a Nutshell	2
1.1.1 Issues to Be Solved by NGN.....	3
1.1.2 Management Issues	4
1.2 Standardization Processes	5
1.2.1 Core Standardization Bodies.....	6
1.2.2 European Organizations	9
1.2.3 IETF	10
1.2.4 Timeline of Standards.....	12
1.3 Transport Layer QoS.....	20
1.3.1 Traditional View of Network Performance.....	20
1.3.2 NGN View at Transport Layer Performance	22
1.4 Application Layer QoS	24
1.4.1 Relation between Application and Transport Layers	24
1.4.2 Application Layer Performance.....	26
1.4.3 Performance Metrics.....	28
Chapter 2 Passive Measurement Technology	31
2.1 SNMP/MIB Technology	32
2.1.1 SNMP/MIB Principles	32
2.1.2 Client-Server Model	33
2.1.3 Network Devices and SNMP	34
2.1.4 Data Types Exchanged by SNMP.....	36
2.1.5 SNMP Communication Patterns	39
2.1.6 SNMP Timeline	43
2.2 NetFlow Technology	44
2.2.1 Traffic Collection Process.....	44
2.2.2 IP Flow Tuples.....	46
2.2.3 Overall Monitoring Architecture.....	48

Chapter 3	Passive Measurement Tools	51
3.1	Common Design Patterns in Monitoring Tools	53
3.1.1	Roles in Monitoring Process	53
3.1.2	Loosely Coupled Monitoring	54
3.1.3	Tightly Coupled Monitoring	56
3.1.4	Distribution of Load in Monitoring Systems	58
3.1.5	Specifics of SNMP	60
3.1.6	Specifics of NetFlow	62
3.2	SNMP-Based Tools	64
3.2.1	Basic Features of MRTG	64
3.2.2	Using PRTG	66
3.2.3	PRTG Output	68
3.2.4	Traffic Components of PRTG	70
3.3	NetFlow-Based Tools	72
3.3.1	NetFlow Compliance	73
3.3.2	top: the UNIX Tool	73
3.3.3	ntop: the Network top Tool	76
3.3.4	User Interface to ntop	77
3.3.5	Statistical Traffic Summaries	79
3.3.6	NetFlow Attributes of ntop	82
3.4	Contemporary Monitoring Realities	84
3.4.1	Example Practical Task	84
3.4.2	SNMP Mapping of the Example Scenario	86
3.4.3	NetFlow Mapping of the Example Scenario	87
3.4.4	Discussion of the Example Scenario	88
3.4.5	Technology Support of Existing Monitoring Targets	90
Chapter 4	Active Measurement Technology	93
4.1	Active Measurement Basics	94
4.1.1	Measurement in Management	94
4.1.2	IP Performance Metrics	97
4.1.3	The Scope of Active Measurements in this Book	99

4.2 Network Performance Metrics	99
4.2.1 Bottleneck Capacity	100
4.2.2 Available Bandwidth.....	100
4.2.3 Round Trip Time.....	101
4.2.4 End-to-End Jitter	102
4.3 Life of a Single Packet in the Path.....	102
4.3.1 Measurement Methodology	103
4.3.2 Shortcomings of Single-Packet Probing	106
4.4 Packet Pair Property	107
4.4.1 Measurement Methodology	107
4.4.2 Packet-Pair versus Single-Packet Techniques.....	109
4.5 Advanced Probe Designs	111
4.5.1 Piggyback Methods	111
4.5.2 Packet Trains	113
4.6 Routing Peculiarities	116
4.6.1 Directionality of Probing Path	116
4.6.2 Loose Coupling	116
4.6.3 Shared Topology	118
 Chapter 5 Active Measurement Methods	 119
5.1 Adaptive Capacity Measurement.....	119
5.1.1 Bottleneck Bandwidth Estimation.....	121
5.1.2 Online Variable Measurement.....	123
5.1.3 Histogram Based Performance Analysis.....	126
5.1.4 Validation Tests.....	132
5.1.5 Discussion of Measurement Methodology	138
5.2 Available Bandwidth Measurement.....	139
5.2.1 Probing Method	140
5.2.2 Method Evaluation	143
5.2.3 Discussion of Measurement Methodology	146

5.3 Lightweight Jitter Estimation.....	146
5.3.1 Estimation Model.....	147
5.3.2 Mining Examples	149
5.3.3 Discussion of Measurement Methodology	151
Chapter 6 Active Measurement Boxes	153
6.1 Standards, Tools, and Projects.....	155
6.1.1 Timeline of Standards and Tools.....	155
6.1.2 Tools and Performance Metrics	157
6.1.3 Major Measurement Projects.....	158
6.2 Test Traffic Measurement Box.....	159
6.2.1 Test Traffic Measurement Project.....	160
6.2.2 TTM Box as a Black Box.....	161
6.2.3 TTM Box Communications.....	161
6.2.4 Role of an Individual Box	163
6.3 QoS Boxes	165
6.3.1 QoSmetrics Box	166
6.3.2 QoSmetrics' Tools	167
Chapter 7 Active Measurement in Context	173
7.1 Management of Measurement Results.....	174
7.1.1 Problem Statement.....	175
7.1.2 Data Mining and Active Probing Results.....	176
7.1.3 Mining Method	178
7.1.4 Mining Examples	183
7.1.5 Discussion of Analysis Methodology	190
7.2 Topological Ramifications of Active Measurement	190
7.2.1 Problem Statement.....	191
7.2.2 Delay Properties of Community Networks	192
7.2.3 Topology Inference Algorithm	194
7.2.4 Discussion of Measurement and Analysis Methodology	201
Bibliography	203
Index	207

Preface

In recent years, Next Generation Networks abbreviated as NGN has become a cliché not only at technical conferences and in research manuscripts but also in communication enterprises where it is associated with a new generation of communication services. Although immediate goals of NGN research and NGN-based businesses are different, – they are all integral parts of the broad area of NGN. In fact, finding the coverage area of NGN while taking into consideration both technical and business aspects of networking has been the main goal of NGN from the start.

Traditionally, network operation is split into two planes – transport and control. These terms are fairly self-explanatory as they are. Transport plane is responsible for basic routing while control plane is traditionally understood as the realm where various activities in regard to network management are conducted. Both planes will be discussed throughout this book whenever the issue of measurement and, specifically, active measurement, enters the picture. Naturally, related networking aspects will be discussed alongside.

Business aspects of NGN are something very unusual from the viewpoint of traditional networking. A network professional never had to look beyond the machinery he/she operates. Business aspects substantially inflate the realm of traditional networking and force network administrators to account not only for ISP's immediate neighbours but also about something called the global Internet, – the ultimate network where each

individual ISP is one among many other players in global scale service provisioning.

All these seemingly disjoint concepts in reality form a very cognitive structure. First, services in the Internet are quickly becoming global thus blurring the focus on individual ISPs. Secondly, QoS, or, in the long version, quality of a global service is also indifferent to individual ISPs and depends on end-to-end path characteristics where individual ISPs are responsible for small parts of the entire path. Thirdly, since services are basically business endeavours in one form or another, QoS requirements of end-to-end performance finally return back to transport layer in form of certain requirements that transport layer has to provide.

It is important to mention that before the transport layer is able to provision the performance required by the higher QoS requirements, the latter have to be properly understood. Unfortunately, in present network that “understanding” still lacks a solid form due to a number of reasons all of which will be unravelled in this book.

Back at the physical layers of the network, NGN separates control plane from transport plane in the new network design. This is another one of those concepts that have never been implemented before. In general, most big concepts used in NGN are novel thus painting all NGN achievements in colours of uniqueness and “first time”-ness.

Transport plane is to be composed of access and core IP networks that will be used to provide global connectivity in future all-IP networks, both wired and wireless. Those parts are not new and already exist in networks today but the usage itself will change majorly in the new framework. In other words, it will not matter where you are and what technical and technological limitations are inflicted upon you via your current access method, you will still be provided adequate global connectivity. As least that is the way NGN promises it to individual users. In fact, constant global connectivity is a requirement for NGN services to be able to operate in the first place.

Control plane, often referred to as service layer, is to be used to connect services and is defined in an abstract way so that services would not depend on underlying transport network technology. This constitutes the main difference as compared to traditional networking technology where services are tightly coupled with transport plane. This is also the main reason why active measurements are increasingly becoming important for network operation in the NGN era.

Coming back to the issue of connectivity within NGN transport layer, taking into consideration that control plane is completely decoupled from

the transport layer, the latter has to be provided uninterrupted service in order to enable NGN services. The lack of connectivity immediately disables any service at the control layer. This strict dependency is the reason why a huge portion of NGN research focuses on provisions of uninterrupted services at transport layer. Besides traditional access methods, new concepts such as location migration, rapid changes in communication quality at transport layer, diversity and changes in capabilities of end devices, and others have to be accounted for.

Besides operational issues, the main task of NGN is to move all currently existing non-IP network technologies to packet switching. The move is complicated by various QoS requirements on the part of various existing technologies. It is difficult for traditional packet switching to support fine QoS granularity. In its original form, the global IP network of the Internet has always been following best-effort scenarios in all its operational aspects. Best effort means that there are no guarantees for on-time delivery of end-to-end traffic. There are many local areas in the Internet which provide some extent of QoS guarantees, but in whole the global network is far from reliable end-to-end QoS guarantees. The issue of heterogeneity also will be addressed in several episodes throughout the book.

Another term that starts with the letter Q is QoE, i.e. Quality of Experience. Based on the above changes in network technologies within NGN, it should be clear that users are getting a much larger share of attention than they used to in traditional networks. In fact, when looking from the viewpoint of a service, the user is the only indicator of the quality of end-to-end path connecting the user and the service. QoE in a very frivolous interpretation is the QoS at the user level. Although this book will venture into the borderline area between QoS and QoE the intended scope of the book does not permit to focus specifically on QoE given that the quality assessment methodology in QoE is fundamentally different from the traditional QoS. QoE in itself is such a large area of expertise that a separate book could be dedicated to QoE alone.

Returning to the physical world again, the Internet in its present state is a mixture of several technologies that provide connectivity at the physical level, the lowest stratum in any protocol stack. When NGN finally converges, all networks will be based on IP protocol. Given that many portions even in core networks are still based on non-IP technology, such as ATM or Token Ring, the shift will require substantial changes in network layout. In fact, this change is already happening today. However, since one of the core concepts of NGN is provisioning all possible services over all-IP networks, even if legacy non-IP technologies are retained with-

in the next generation network they will be forced to interface with NGN through some kind of conversion between all-IP NGN core and a given legacy connection technology.

The shift to all-IP networks entails all flaws of best-effort networks. This includes generally unpredictable multimodal probability distribution of traffic in IP networks. This is a major difference from telephone networks, where arrival rate has a fixed and mathematically well defined value. Traffic in IP networks at very long time intervals follows multifractal distribution, both the mean and variance of which are difficult to define in practice.

This imposes major limitations on the definition of end-to-end transfer delay and packet loss included in Y.1541 recommendation on QoS classes in NGN. These issues are generally collected under the topic of network performance, which is also an active part of NGN standardization process. For the sake of the unity of vocabulary, it should be mentioned that the meaning of “standardization process” is the same as that of “standards process.” Some people use “standards” as an abbreviation of the word “standardization.” In this book both will be used interchangeably.

Now, network performance even as a term is tricky already. On one hand, there are a few traditional definitions of it being an indicator of how well the transport network performs in its operation. On the other hand, applications tend to perceive network performance in more detail than is accessible to the transport layer.

Talking specifically about performance metrics, end-to-end performance is viewed by end-to-end delay and jitter, where jitter in plain words is the variation of the delay. These and several other metrics will be considered in much greater detail in this book since they constitute the basic toolbox used to define network performance by measuring it actively.

Applications, on the other hand, may view network performance a bit differently than it is done at transport layer. Some applications perceive network performance in terms of its ability to handle traffic bursts, which can be described by maximum achievable throughput, average/maximum burst length, and other related characteristics. Generally, QoS definitions at application layer may seem less stringent but are much more diverse in terms of how the performance is defined and measured. This discrepancy in perception of network performance by applications and NGN transport layer is yet to be addressed by NGN standardization process in several years to come.

Since QoE was already mentioned once, it is worth to continue on this track by mentioning that QoE forms yet another upper stratum of methods that evaluate end-to-end network performance, but this time purely

from the viewpoint of individual users. The methods in this area are often purely statistical and thus have little in common with traditional evaluation methods used at the network layer. Additionally, whatever the findings of the former are, it is very difficult to verify them given that users do not really know nor can learn about what is happening on end-to-end paths at the time the service is being provided over the network. This is the sole reason why QoE methods form an independent group of network performance assessment methods that do not relate to traditional network performance very well. It is also very difficult to find even partial compatibility between the two viewpoints at end-to-end network performance.

With all the above shifts both in networking technology proper as well as in methods used to estimate end-to-end performance, end-to-end measurements come out as the only methodology ready to measure and provide knowledge about performance of end-to-end network paths. Before this statement is supported with facts, this book will take a short tour of existing passive performance measurement methods by the end of which the reader hopefully will be ready to accept the fact of the need in active measurements as one the basic building blocks in future NGN services.

It might also be necessary to clear active measurement as the term itself prone to potential misinterpretation. Active measurement refers to a method or a software tool that discovers network performance characteristics in result of sending probes along arbitrary paths. Each individual probe would normally traverse a certain path with a given source and destination IP addresses, thus, resulting in the measurement of network performance characteristics along this very path. Active measurement along a single end-to-end path is a primitive building block of the technology and may be used as a powerful tool in defining the performance of a network through aggregating measurements from many individual paths.

Now, the active probe is normally created as a sequence of dummy packets. The packets are referred to as dummy since the payload they may carry often contains no information at all but is piggybacked to the header in order to create a packet with a specific size. The position of each packet in the sequence as well as the time gap between the packets is also part of active measurement research since these settings pre-define how the probe is to traverse the network. This dependency is exploited by most active measurement methods.

To provide yet more clarity, the adjective “active” added to various terms in this book has no relation to the aging technology called “active networks”. While active networks enclose all its activeness within intermediate routing equipment units, active measurement is happening only at edges and does not require participation on the part of intermediate

routing equipment. There are some minor exceptions from this rule but they are still very far from the concepts offered by the research on active networks.

Yet another misuse of the word “active” is more subtle and is sometimes used to explain high responsiveness and agility on the part of passive measurement methods. Such methods in this book will be called passive monitoring while the word active will normally be attached before the word measurement. Although there is but a subtle difference between the terms, the difference at the physical level is immense. The word active in passive monitoring indicates high responsiveness of a passive monitoring process while when used in the measurement it means literally that active steps are to be undertaken in order to perform a measurement in the first place.

This book will pay special attention to elaborating on the difference between these two terms.

Yoshiaki Tanaka
Marat Zhanikeev

Chapter 1



NGN Standardization and QoS

NGN today remains an ongoing *standardization process* down to the level of standards defining details of its implementation and further operational issues that become important once the network is commenced on the *global scale*. As happens with all projects started from scratch, NGN requires plenty of standardization work conducted prior to implementation.

Some of the areas that require special attention and, more importantly, an outcome in form of a set of rigid standards, – are the technologies and technical solutions where NGN is unique. Standards have to answer to many questions including but not limited to the items on the following list:

- how to separate *transport plane* from the *control plane*;
- what should be included in the *interface abstraction* of the transport layer so that various underlying communication technologies could be abstracted and successfully employed from the control layer of NGN;
- how to classify *QoS requirements*, and, more importantly, how to translate *QoE requirements* into NGN control plane QoS requirements and further into transport layer network performance requirements;

- how to *measure end-to-end performance* and how to represent it at various strata of the NGN technology stack;
- how to manage such a huge structure provided we successfully make it work in the first place.

This chapter offers insight on what is currently happening in the world of NGN standards and what place in it is occupied by standards related to network performance. Since network performance is not an abstract concept in NGN anymore, but rather is a number of practical concepts with practical methods of measurement and assessment for each, naturally, network performance in standardization comes hand in hand with active measurement which is, after all, the main topic of this book.

1.1 NGN in a Nutshell

In its current stage, basics of *network design* and *communication protocols* are embraced by a large number of standardization documents developed for over two dozen years since the Internet first became a global endeavour. NGN today resembles the standardization at the early age of the internet by having many new aspects of networking that did not exist or were not important before. At such an early stage is it forgivable that NGN standards do not yet cover all possible aspects related to the operation of NGN networks yet.

So, before the present state of NGN standardization can be presented in an easily digestible form, it is logical to present directions in which standardization process was developing at the time this book was being written. This in its turn is better explained by elaborating on issues NGN is tackling today including those that do not yet have a clear solution mostly due to the lack of clarity. In other words, the mindmap of NGN standardization is not yet finished, – while large concepts are already written on paper and placed in proper locations based on mutual relations and dependencies, details in smaller font that fill in the gaps between these large concepts are not to be found in many places on this imaginary mindmap.

This section's purpose is to prepare the reader for later presentation of existing standards by presenting the overall mindmap first. This task is best accomplished by adopting a complaint-like narrative of issues that currently exist in NGN standardization process.

1.1.1 Issues to Be Solved by NGN

Jumping straight into the maze of NGN standardization activities, below is the list of some problems the global network encounters today:

- since the time the current network had been developed, a lot has changed on the user end; current needs are extremely *heterogeneous*, include rich *multimedia* components, and require individual treatment in transport layer depending on the nature of content as well as explicit choice on the user end;
- transport layer used to make no distinction about the content; while transport layer will generally retain its current behaviour, there is a need for a superseding logic of *traffic control* based on the nature of the delivery;
- *separation* of transport and control planes in the network is something that has not been done in the history of communications yet; it partially resembles the logic of complex MPLS networks that exist today except NGN will have to be much more dynamic and flexible in its decisions.

The above list is far from having covered the whole range of problems raised by people working in NGN research today. But at least it gives an idea of how different are the tasks that used to exist and still do in the network of today from the tasks that will have to be taken care of by the NGN network in the near future.

As was already mentioned, NGN standardization process has been active for several years now, resulting in a large number of recommendations drafts and even some final documents ready to be implemented in communication industry. Since the core of NGN is multimedia content, various multimedia technologies and protocols have already been solidified as NGN standardization. *IMS* [6] is the international organization responsible for the development of standardization and general international collaboration in this area, IMS being abbreviated from *IP Multimedia Subsystem*.

IMS is a good example of how things are done in the era of NGN. IMS is a *forum* of several dozen international organizations, companies and even countries, all directing their efforts towards establishing common grounds for network communications in the future. Most standards

work in NGN is done in exactly the same way, i.e. as a product of *multi-party collaboration*.

If you accept the fact that the scale of the global Internet is beyond a single organization however large it may be, you should at least suspect that IMS is not the only organization that works on NGN and still retains an extremely practical grip on networking aspects. In fact, 3GPP [1] is arguably its closest rival with realms of both organizations overlapping in many areas. 3GPP is similar to IMS is on way, however, – it is also a *global initiative* and it is not a single group of people but rather a forum of several dozen organizations unified by the urgency of the need in solid standards in the area.

Both IMS and 3GPP will be put in context of each organization's role in the present NGN standardization process.

1.1.2 Management Issues

Based on the contents of the previous subsection it may appear that *multimedia* is the single driver of NGN standardization process. While multimedia-related areas of NGN are clearly booming, the *management of NGN network* at the global scale remains unclear not only within standardization process of NGN but also partially in research. The fact that NGN represents the next generation of networking does not relieve it of the necessity of a well defined management mechanism. Given that NGN is not as loose on communications quality as its predecessor, – traditional network today, QoS and differential treatment based on the nature of content become of utmost importance. This task is clearly much more complicated than it used to be in conventional networking. Two key terms used in this obscure area of NGN are *performance management* and *performance measurement*. These very terms also lay the foundation of a larger area of research in networking targeted by this book, i.e. network measurement. It is necessary to mention that traditional network management does not include the concept of measurement at all, leaving this decision to the actual people in charge of maintenance of local networks. The main purpose of NGN is to create a global network with *QoS guarantees* between any arbitrary end points in it.

Performance measurement left at the hands of local network administrators until now has tended to be *passive* in nature. In fact there are very solid reasons for such passiveness.

First, as will be presented later in this book, *passive performance measurement* normally ensues collection of very detailed *statistics* about network

performance resulting in a huge *bulk of data* and quickly obliterating any chances for online analysis subject of detecting performance problems. Being unable to detect *performance problems in real time* or with delays with the small range of reasonable *near real time*, there is no question about the inability to provide an instant follow-up in form of a management decision based on the real time performance data.

In practice passive performance measurement could take the form of a coarse *packet level* or *flow level traffic dump* in carefully selected network equipment within the margins of a local network. Because the dumping process is normally very performance hungry as well as due to the decoupling of dumping process from the following offline analysis process, the reason why such methods are called “passive” should be clear.

Even if there existed a real time passive measurement system the performance requirements posed by NGN scenarios would still not be met. In order to maintain a global scale NGN service, performance measurement should cease being a part of local maintenance and become a part of *global network management*. Clearly, issues with passive measurement even at relatively small local scale should only get worse with a substantial scale boost in virtually any NGN-based communications scenario.

Having elaborated on the issues that exist in relation to performance measurement in NGN, the rest of this chapter will be dedicated solely to the standardization documents related to network performance in both its management and measurement parts and will ignore the rest of NGN standardization process. It would require a separate book to cover all of NGN standardization documents, but once its measurement and management aspects are singled out it becomes clear as to what particular problems should be cleared before NGN can be deployed at the global scale.

1.2 Standardization Processes

As was already mentioned, NGN standardization in this book is not considered in its full scale but only in the areas related to network performance, its measurement and management. However, even when the search for NGN *standardization documents* is limited to performance measurement and management, the list is still very long.

To make things worse, activities conducted by many *organizations* often overlap making it difficult to find order in the overall standardization process. This section attempts to find some form of order by considering most NGN standardization processes in their interrelated totality.

The order of material in this section is as follows. First, NGN standard-

ization will be considered from the viewpoint of major organizations that participate in this process. Some of them spread their hands across the entire world while some limit their range to Europe while still retaining the *global prospective*.

Separate attention will be given to IETF (Internet Engineering Task Force) as an organization that is not directly related to NGN but has acquired a firm stance in the area of network performance. Since there is not a single organization that merges the two areas together, IETF has to be given a special consideration while a glue has to be provided separately between network performance as defined by IETF and the overall NGN *framework*.

Finally, this section will conclude with a timeline and a list of standardization documents developed within the framework of NGN or somehow related to it.

1.2.1 Core Standardization Bodies

Figure 1.1 displays the current state of standardization processes in the area of NGN and is not limited only to *network performance standardization*. Instead, those organizations that are directly related to network performance as well as its measurement and management aspects are made to stand out from all other organizations.

All *working groups* are collected in boxes representing international organizations or standardization bodies they belong to. Each organization by itself is often large and contains multiple *divisions* and working groups in charge of various areas covered by NGN. Full description of each of these organizations would require a separate book, so only a short description of each is provided below in areas relates to the main focus of the book.

Although it is difficult to grasp the entire structure in Figure 1.1 especially because of the cryptic abbreviations, but the rest of this chapter will strive first to explain some of these abbreviations and then put them in context. If you come back to this Figure 1.1 having a better knowledge of details, it should be easy to realize that each separate organization has a *unique realm* of standardization activities that does not really overlap much. The overlap created by the standardization work in the area of network performance constitutes tiny parts of each organization's activities. The same can be said for all other aspects of NGN standardization.

Below are the details for some of the organizations – boxes in Figure 1.1.

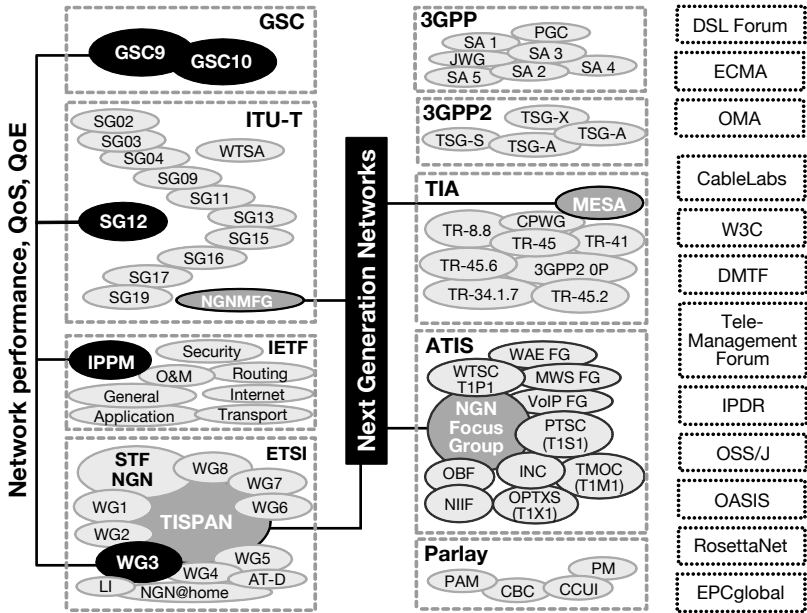


Figure 1.1 Organizations involved in NGN standardization work worldwide.

ITU-T [7] in Figure 1.1 plays the main role and also happens to be the centre of standardization work on NGN network performance through its study group SG12.

ITU stands for *International Telecommunication Union* and ITU-T stands for ITU Telecommunication Standardization Sector which places it in the right place in NGN standardization process. ITU is also one of the oldest international organizations in the area of telecommunications. Its activities date back to 1865 when it was known by its old name, the International Telegraph Union. In second half of 20th century, though, it was one of the main contributors to the development of global telecommunication infrastructure.

Since the early times of ITU-T, the global scale of its activities rooted in the global scale of the membership in this organization. At present time, it has 191 member states and over 700 public and private companies continuously participating in its activities, those related to standardization forming the overpowering majority.

ITU-T has a well established and very rigid standardization process

that any new technology has to undergo before it can be accepted as an international standard. Once it is, however, it normally becomes a *de-facto standard* given the large number of companies that implement the technology in question in their products.

SG12 is the study group (hence the abbreviation SG) responsible for *Performance and Quality of Service*, which is, in fact, its name. Since the number of a study group contains no information in it, study groups in ITU-T are written together with their names.

SG12 is responsible for a number of *normative documents* created in the period from January 2005. The group is still open and very active today. Specific details about the group can be found in [8], which contains description of its activities and is updated regularly. Most standardization documents themselves are restricted for internal use only, however. This policy is understandable given the *preliminary* nature of many of the documents published within the operation of the study group.

The resources at [8] itself may be helpful to discover recent advances in the area of SG12 expertise. The long list at this location contains not only standards, but also intermediate documents coming from regular meetings by the group. In the process the list will reveal where the efforts are being directed by the group.

Also, since this book is a one-time endeavour and will lag behind in several years to come, the resources at [8] could be a good place to gather up-to-date information on the subject. The fundamental elements of the technology and methods involved with network performance measurement and management will remain the same for many years to come thus allowing this book to age gracefully.

It should be noted that ITU-T does not dedicate all its time to NGN. The scale of the organization is much bigger than NGN and many of its study groups are in charge of subjects that are not related to NGN at all. However, given that NGN is the most important global endeavour in the Internet today, ITU-T pays special attention to NGN. This attention is conveyed by the *NGN Focus Group* and *NGNMFG* study group within ITU-T. Those two entities take part in global conversation on various aspects of NGN activities.

GSC is another organization that makes an interesting show case of how standardization process is conducted in the international arena today. GSC stands for *Global Standards Collaboration* and it is not an organization itself but rather a meeting place for many other standardization organizations that often need nothing more but a venue to meet and dis-

cuss areas of common interest. In other words, GSC is a *forum* of global organizations.

More details about GSC can be found at [4]. Of course, ITU is on the list of the member organizations of GSC. Another major organization is ETSI, – this organization will be covered in detail later in this chapter. There are many other organizations in charge of this.

While ITU and other global organizations are not limited in scope to NGN, GSC is all about NGN as indicated in Figure 1.1. If one needs to know what state NGN standardization is currently in, the first place to go is SG12 in ITU-T. The second place is definitely the short list of GSC meetings. Since the latter are held annually, the list of meetings is not too long. The proceedings of each meeting are normally very large in volume, however, which makes it a better source of information than even SG12.

GSC *annual meetings* are hosted by each member organization based on a rotation pattern. Naturally, all normative documents that are created as product of multiparty discussions are published under the title of SGCxx where xx is the sequence number of the meeting. Although many subjects are covered by each meeting, it happened that SGC9 and SGC10 were heavily concerned with the main subject of this very book, i.e. network performance, its characteristics, metrics, methods of its measurement and future stipulations in regard to the management of the newly developed NGN network paradigms. Unlike of ITU-T SG12, GSC documents are released to the public in form of lengthy bundles of reports. Each bundle contains a list of documents with comprehensive description of each which makes the information hunt a relatively easy job.

1.2.2 European Organizations

It is not a coincidence that ETSI hosted the meeting of GSC10 in 2005. ETSI happens to have its own working group on the issues of network performance measurement/management and is playing a vital role in scrupulous analysis of technological ramifications when the network finally shifts to NGN on the global scale.

ETSI [2] tends to be extremely practical in all its standardization activities. The abbreviation stands for *European Telecommunications Standards Institute*. It has recently earned the trust of European Union, thus, obtaining the access to its huge telecommunication market. In spite of the fear of being caught in a partial judgement, ETSI documents are always very infor-

mative, lengthy, and come in form of pleasing to the eye PDF documents.

Although a lion share of actual documents discussed further in this book is created under the auspices of ITU-T, it should not be forgotten that ETSI has played its role in developing contents in these documents through international collaborations within GSC in particular and many other cases of global collaboration. Specifically, the *ETSI TISPAN WG3* (reads ETSI's TISPAN's Working Group Three) [3] is responsible for end-to-end protocols that will be used to accommodate the various NGN multimedia-rich technologies in the future. TISPAN stands for Telecoms & Internet Converged Services & Protocols for Advanced Networks. Again, as was mentioned before, ETSI standardization process is very rigid and one can expect to find plenty of useful information in the web space of the organization.

Given the global nature of such collaborations, most documents that come from such discussions are made available publicly and there is nothing more public than an open *web access*. This kind of openness exists not only for a general viewer, but also for members of member organizations taking part in global collaboration. Public access in this case often also seems the only feasible method of information exchange.

1.2.3 IETF

Until this point in this section all organizations had some kind of interest in NGN or were directly involved in NGN standardization process. IETF is different from those organizations in that it does not really concern itself with NGN as such, but rather provides many useful standards in any possible area of network-related activities. Normally the *granularity* of IETF standards is such a that NGN appears to be infinitely bigger in scale, but when practical considerations are demanded within the NGN framework, there is nothing more handy than an *RFC* (*Request for Comments*) on the topic in question.

IETF [5], abbreviated from *Internet Engineering Task Force*, cannot be left out of the picture displayed in Figure 1.1 without rendering the entire structure incomplete. IETF has already delivered a number of standards that have become *de-facto* in the Internet today.

The main reason for such success is in the process of RFC, or Request for Comments, which is a form of normative document created by whomever might be interested. Literally, anyone can submit their RFC in any area related to the Internet for public and professional review. It does not

mean, however, that RFCs always become *de-facto standards*. There are two other levels that each newly proposed standard should undergo, – the state of Internet Draft Recommendation, and, when this one is approved, the state of Internet Draft.

Internet Drafts are documents that have undergone quite a stern process of *peer* and professional review, normally by researchers and specialists working in exactly the same area as the author for the original RFC. This makes final Internet Drafts a form of collaboration among people in research community.

When the issue of network performance comes into question, IETF has played its final role vested in its *IPPM* charter. IPPM stands for *IP Performance Metrics*, which puts it right in the middle of NGN standards related to network performance.

As most other *global collaboration* activities, IPPM retains a publicly open stature and provides regular updates with information on what has been accomplished by the working group over the years as well as what is planned for the future. IPPM is still an active group and at the time of writing there are meetings planned for the group six months in the future. However plain the target of IPPM might appear, it already possesses a long list of RFC documents developed in this line of work over the years. This book will keep coming back to refer to IPPM on many occasions, since the list of metrics defined by the group constitutes the foundation of network performance and how it can be perceived by measurement.

To give a simple example of how IETF RFCs can be useful, let us take the example of network management. As will be considered later in this book, *SNMP* (*Simple Network Management Protocol*) is the *de-facto* standard tool used by network administrators to take care of the management of their networks. SNMP as the protocol was first created in form of an RFC and only then was implemented by many parties unrelated to each other. Right now there are many working implementations of SNMP, but all of them use the same RFC standard and, thus, are compatible with each other. The issue of compatibility will be considered in detail in this book.

With RFCs it is almost impossible to not find something useful in whatever your quest may be. RFCs can also form natural hierarchies where a later RFC uses an earlier RFC to build on top of it. Such hierarchical clusters can be joined together and provided they successfully complete the standardization process within IETF, can become an Internet Standard. SNMP, in fact, has already become an Internet Standard that incorporates several separate RFCs covering various areas of SNMP-based network management.

IETF's process is globally unique by being able to bring research to

its *practical implementation*. By the way of the established standardization process what begins as early research may later on ripe up to the level of an Internet Standard. In plain works, this is the shortest path a researcher in the area NGN can take to give his/her research a chance to become a practical implementation in a not so distant future.

How distant this future might be varies from RFC to RFC, but the practice shows that the period of 3 to 5 years is enough provided yours is a hot research topic at the time.

1.2.4 Timeline of Standards

In Figure 1.1, SG12 of ITU-T is in the centre of the *global standardization process*. Although standards related to network performance are usually the product of collaboration among several international organizations and companies, normative documents are often released under the name of ITU-T. Partial content may be found in documents published separately by each part-taking organization, so, in some case it might be more helpful to refer to an individual organization rather than to read the final document published by ITU-T. Also, as was mentioned before, ITU-T normative documents are restricted to internal use while in development stage, while related content can be found in publicly accessible form in other organizations.

Figure 1.2 contains the timeline of some normative documents related to network performance. The term itself is not always present in the titles of documents, but the relation can easily be drawn to such terms as QoS and QoE and terms for performance metrics. It is clear from the timeline that recent documents are becoming increasingly concerned with *performance metrics* and methods of their measurement. More than that, a new term, *performance measurement management*, was created to refer to measures undertaken to manage the data obtained from *performance measurements*. The overall picture in Figure 1.2 clearly indicates how increasingly concerned NGN standardization process is with the performance of the global network of the future and specifically in how to measure and manage this performance.

Many documents in Figure 1.2 are still in draft stages given that a single document normally takes at least a year to come to become a complete standard. Many are developed over much longer periods of time.

Apart from the term Draft there is also the term *Reply to Draft* used in the standardization process. Specifically, the issue of performance measurement and management are incomplete standards as of now and many

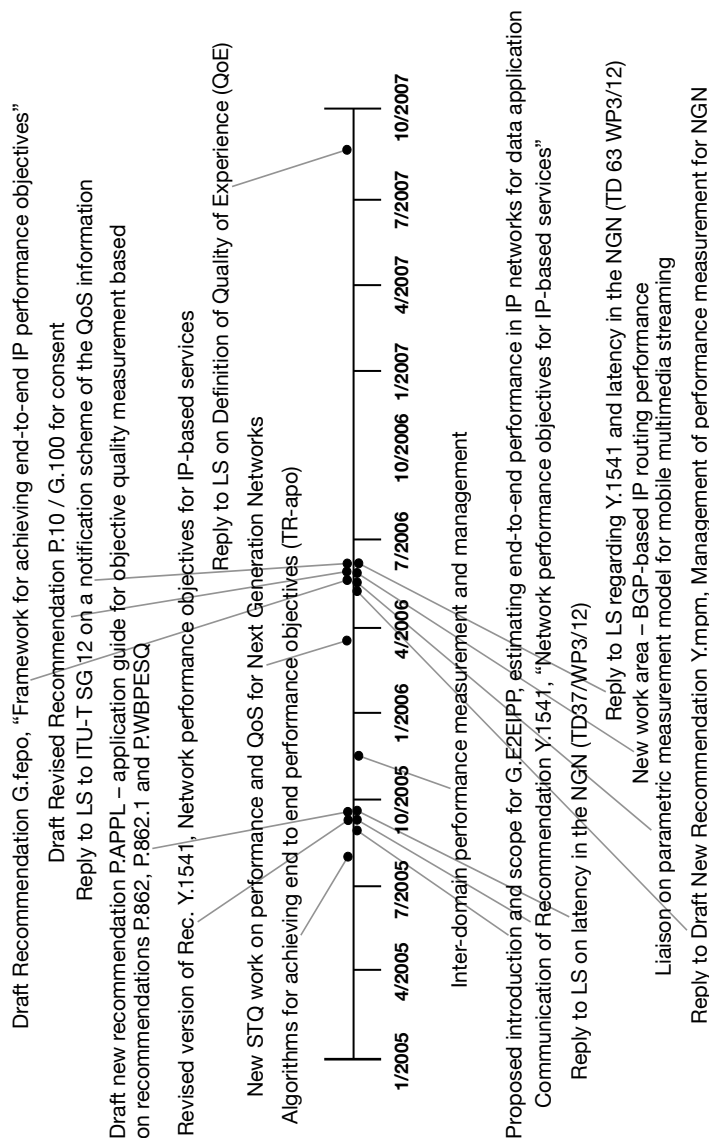


Figure 1.2 Timeline of standardization documents related to network performance.

of documents in Figure 1.2 and later in a table form have Draft or Reply to Draft prefixed to them.

There is another class of documents in Figure 1.2, that cover very specific areas related to a certain standard. All documents in Figure 1.2 are related to network performance, so, the focus of those documents should be something specific in this general areas. For example, the document titled Algorithms for Achieving *End-to-End Performance Objectives* is a very specific very close to an implementation of a certain method or a technology. These documents also play their role in the overall NGN standardization process.

Tables 1.1 through 1.5 contain the chronological list of normative documents produced within several recent years within or by collaboration with SG12 or ITU-T. The list is fairly long and contains more details than the timeline in Figure 1.2. A closer look at it will, however, should leave the same impression, – that NGN standardization process has recently embarked on the path with a few discrete goals:

- to solidify the list of network performance metrics at each stratum within the NGN networking framework;
- to provide methods that would make performance metrics at each stratum of NGN networking compatible in both directions with its immediate neighbours above and below;
- given that the measurement methods exist only for the network performance at the lower level of networking dubbed “transport layer” by NGN, to come up with measurement methods, again, for each separate NGN stratum;
- think out methods to manage *network performance data*, i.e. provide network performance measurement, – NGN is a global endeavour and such methods should be feasible at the global scale.

The above list may have had a few additional entries, but all the above manage to cover the majority of NGN concerns in the area of network performance. It should be noted that the above list does not comprise other aspects of NGN activities. Such a list will span dozens of pages and would make little sense in this book. The above list is a snapshot of NGN activities in the very specific area of network performance, its measurement and, finally, the management of both.

Key documents from the list in Tables 1.1 through 1.5 will be covered

Table 1.1 Table of standardization documents produced in area of network performance in 2003-2007 (part 1).

Date of issue	Standards Organization	Document Title
2005-01-26	2005-01-26	Revision of Recommendation Y.1541, "Network Performance Objectives for IP-based Services"
2005-01-26	ITU-T SG12	Additional information on "Mapping between ITU-T and 3GPP QoS Classes and Traffic Descriptors"
2005-01-26	ITU-T SG12	Liaison on IP Transfer Capabilities and QoS Signaling
2005-01-24	ITU-T SG12	Liaison on PN-3-0062 (TIA/EIA-921) Network Model for Evaluating Multimedia Transmission Performance Over Internet Protocol
2005-01-24	Rapporteurs for Q11/12	Liaison on QoS Interworking
2005-01-14	ETSI TISPAN (WG5)	LS on NGN QoS Framework and Requirements
2005-01-14	ETSI TISPAN (WG5)	LS on Mapping between ITU-T and 3GPP QoS Classes and Traffic Descriptors
2005-01-26	ITU-T SG12	Response to 3GPP Reply on Mapping between ITU-T and 3GPP QoS Classes and Traffic Descriptors
2005-01-26	ITU-T SG12	Liaison on Network Performance Tests for IP-based voice services
2005-03-03	ITU-T SG2	Revised new Recommendation E.QoSParam
2005-03-03	ITU-T SG2	QoS and performance standardisation areas for NGN and QoS interworking
2005-04-05	WG3/FGNGN	LS on Progress on QoS and performance study in FG NGN
2005-09-15	WG3/FGNGN	Algorithms for Achieving End to End Performance Objectives (TR-apo)
2005-10-13	ITU-T SG4	E.QoSParam
2005-10-20	Rapporteurs for Qs13 & 17/12	Communication of Recommendation Y.1541, "Network Performance Objectives for IP-based Services"
2005-10-20	Editor, G.NCTT	Draft new PG.NCTT, Network Contribution to Transaction Time

Table 1.2 Table of standardization documents produced in area of network performance in 2003-2007 (part 2).

Date of issue	Standards Organization	Document Title
2005-10-20	Editor of G.E2EIPP	Proposed Introduction and Scope for G.E2EIPP, Estimating End-to-End Performance in IP Networks for data applications
2005-10-19	Editor P.WBPESQ	Draft new recommendation P.WBPESQ - Wideband extension to P.862 for the assessment of wideband telephone networks and speech codecs
2005-10-19	ITU-T SG12	Terms of Reference for the HN QoS Task force, under JCA HN
2005-10-19	ITU-T SG12	Result of the Joint meeting of ITU-R WP 6Q / ITU-T SG12 on Subjective and Objective Voice and Audio Quality Assessment Methods (Wednesday 19 October 2005) and common action plan
2005-10-20	ITU-T SG12	Response to "LS on Ethernet performance parameters" (TD34/WP3/12)
2005-10-20	Editor of Y.1541	Revised Version of Rec. Y.1541, Network Performance Objectives for IP-based Services
2005-10-20	Editor P.APPL	Draft new recommendation P.APPL - Application Guide for Objective Quality Measurement Based on Recommendations P.862, P.862.1 and P.WBPESQ
2005-10-20	Editor of G.NIMM, Rapporteurs of Q13/12	Final draft of new Recommendation G.NIMM - Network Model for Evaluating Multimedia Transmission Performance Over Internet Protocol for consent
2005-10-20	ITU-T SG12	Reply to LS on Latency in the NGN (TD37/WP3/12)
2005-10-20	ITU-T SG12	Response to "Request for additional guidance regarding Y.1541" (TD32/WP3/12)
2005-10-20	Editor of G.E2EIPP	Summary for G.E2EIPP

Table 1.3 Table of standardization documents produced in area of network performance in 2003-2007 (part 3).

Date of issue	Standards Organization	Document Title
2005-12-13	ITU-T FGNGN	Inter-domain performance measurement and management
2005-12-13	Rapporteur for Q17/16	LS on IP-to-IP gateway testing methodologies
2006-04-20	ETSI TC STQ	New STQ Work on Performance and QoS for Next Generation Networks
2006-05-17	ITU-T SG2	Incoming LS: Draft Recommendation E.802 (formerly E.QoSParam) - Framework and methodologies for the determination and application of QoS parameters
2006-06-09	Rapporteurs for Q7/12	Response to Q17/16 Liaison Statement on IP to IP gateway testing methodologies
2006-06-12	ITU-T SG12, Q17/12	Guidance on the code point space for future expansion of Y.1541 classes
2006-06-12	ITU-T SG12, Q17/12	Reply to Draft new Recommendation Y.mpm, Management of performance measurement for NGN
2006-06-12	SG12, Q17	Reply to LS regarding Y.1541 and Latency in the NGN (TD 63 WP3/12)
2006-06-12	SG12, Q17	New Work Area - BGP-based IP Routing Performance
2006-06-12	ITU-T SG12	Liaison on parametric measurement model for mobile multimedia streaming
2006-06-09	Editor Draft Recommendation G.fepo	Framework for Achieving End-to-End IP Performance Objectives
2006-06-12	Rapporteurs for Qs 2, 7 & 10/12	Response to Q17/16 Liaison Statement on IP to IP gateway testing methodologies
2006-06-12	Rapporteur for Q17/12	Proposed Appendix XI/Y.1541Example and Background on IPDV composition
2006-06-12	ITU-T SG12, Q17/12	Reply to LS to ITU-T SG 12 on a notification scheme of the QoS information
2006-07-19	Chairman, FG IPTV	Incoming LS: QoS and Performance work for IPTV

Table 1.4 Table of standardization documents produced in area of network performance in 2003-2007 (part 4).

Date of issue	Standards Organization	Document Title
2006-11-13	ITU-T SG9 WG5	Incoming LS: Reply to FG IPTV Liaison statement "QoS and Performance work for IPTV"
2006-12-07	ITU-T SG16	Reply LS to SG12 on IP to IP gateway testing methodologies
2007-01-09	JRG-MMQA	Reply to SG12 Liaison to JRG-MMQA: "Study of multimedia streaming interruptions, in preparation for a proposal of a parametric opinion model for multimedia streaming services in a mobile network"
2007-01-23	Rapporteurs for Qs 13 & 14/12	Liaison Statement to VQEG on SG 12 activities on multimedia streaming quality performance modeling
2007-01-23	Editor, G.HLP	Latest draft of G.HLP "Framework for higher layer protocol performance parameters and their measurement"
2007-01-19	SG12, Q17/12	Reply to LS to ITU-T SG 12 on Performance monitoring parameters for IPTV
2007-01-19	Rapporteur for Q8/12	Revised text for Draft Appendix I to ITU-T Recommendation G.109 "The E-model based quality contours for predicting speech transmission quality and user satisfaction from time varying transmission impairments"
2007-01-10	ETSI STQ	Incoming LS: Initial Observation on Draft 3GPP TR 26.944 End-2-End Multimedia Services Performance Metrics (Release 7)
2007-01-23	SG12, Q17	Reply to Ethernet performance related to work in Q9/15
2007-01-23	SG12, Q17	Reply to LS regarding : Speech and audio coding matters (clause 8) (TD 171 GEN)
2007-01-23	Rapporteur for Q17/12	Proposed Appendix on Digital Circuit Emulation Requirements for Y.1541

Table 1.5 Table of standardization documents produced in area of network performance in 2003-2007 (part 5).

Date of issue	Standards Organization	Document Title
2007-01-23	Co-Rapporteurs for Q13/12	Liaison statement on a new Recommendation Y.1562 (G.HLP) - Framework for higher layer protocol performance parameters and their measurement
2007-01-23	Co-Rapporteurs for Q13/12	Liaison statement on a new Recommendation G.1070 (G.OMV) - Opinion Model for Videophone applications
2007-01-23	Co-Rapporteurs for Q13/12	Comments on TR 26.944 "End-2-End Multimedia Services Performance Metrics"
2007-01-23	Co-Rapporteurs for Q13/12	Reply Liaison statement on QoE considerations for multimedia services including IPTV
2007-01-24	SG12, Q9	Reply to LS regarding : Speech and audio coding matters (clause 5) (TD 171 GEN)
2007-01-23	SG12, Q17	Reply to LS on MPLS performance aspects
2007-01-23	SG12, Q17	Reply to LS: Draft new Recommendation Y.mpm, Management of performance measurement for NGN
2007-02-28	ITU-T SG2 (Geneva, 30 January -8 February 2007)	Responses to Liaison Statements from SG12
2007-07-05	ITU-T SG9	Reply to LS on Definition of Quality of Experience (QoE)

in detail further in this book. Better yet, many methods considered in this book will fall under the realm of some of these documents regardless of the fact that the methods themselves were not originally developed for NGN. Those will be good examples that certain technologies and methods do not require NGN as the collective driving force as long as the method or a technology target global end-to-end network performance.

1.3 Transport Layer QoS

As was mentioned previously, NGN separates control plane from transport plane completely. This means that whatever performance metrics are used at the transport layer they do not translate directly into *application layer metrics*.

This is quite different from traditional view on *network performance*. Traditionally, a step-up in the protocol stack is deemed to cause negligently small change in the performance compared to the performance of the lowest, i.e. the physical layer in the *protocol stack*. In plain words, whatever is the performance of your network, it is not affected by the complexity of the protocol stack used in intermediate nodes or edges of a network path.

This section is entirely dedicated to pinpointing this very difference between the *traditional view* at network performance and that advocated by NGN today. The understanding of this difference is crucial for understanding of all the following material that in most parts will assume that implementation is to be done for the future NGN networks.

1.3.1 Traditional View of Network Performance

So, how is network performance regarded traditionally? NGN is not an abstract technology that would build on top of transport layer and thus disregard its contents completely. NGN is the case of a complete remake from the very bottom of the network to the new heights that did not exist before but are planned by NGN networks. In plain words, the currently existing building of networking will be demolished first and then a new building will be built on the same spot but much higher, with much better interior design, better utilities, etc.

Holding on to the metaphor a little longer, the old building is not demolished completely, the ground floor, or the concrete basement, if you will, will remain and will be used by the construction work that has al-

ready started by NGN. This ground floor is the physical, data, and in many places in early NGN, the transport layer of the traditional network stack. Everything above these three layers will be done quite differently within NGN.

However, NGN will still have a transport layer, just as all traditional networks do today. So, what is the difference between the two transport layers?

This difference is best explained by the term *performance degradation*. In fact, this or a similar term is used in very recent standardization documents, such as *Y.mpm* [10] or *G.1010* [9]. These preliminary standardization documents that are currently in their draft stage were written less than a year before this book was being written.

Now, performance degradation is a reasonable term to use in context of network performance since all we care about there is how bad the performance of a particular network is. This small nuance in phrasing will also be discussed later in this book as it has some relevance to measurement methods in context of constantly improving network performance, or, on the opposite side, constantly decreasing network performance degradation.

Network degradation is an interesting way to look at network performance. In this philosophy, networks should all be perfect by default, i.e. if your end-to-end link has 100Mbps bottleneck in it, you still should be able to transmit as little as at least 100Mbps of traffic through it on the other side.

Of course, there is no such thing as constantly perfect networks. Even if you are the only user on an end-to-end path, there are many other reasons why your network will not perform at 100Mbps. This “inability” to provide the *nominal performance* written on the box is something that separates traditional network performance model from that used in NGN networks.

First of all, the longer your end-to-end path the higher is the performance degradation, again, even if you are the only user on the link. Packets have to arrive at each intermediate node, be read and analyzed and only then transmitted further along the end-to-end path until the destination is reached. Let us call this *horizontal performance degradation*. Similarly, each next step up the protocol stack should cause some performance degradation. This is the *vertical* component to performance degradation. Both vertical and horizontal components are very closely related to each other and sometimes even mean the same thing. For example, the more hops you have on an end-to-end link the worse is the end-to-end degrada-

tion both because you have added more hops and because each of those hops has a certain protocol stack which is to be climbed by each packet before it can be routed on to another hop.

In traditional networking this degradation is grossly neglected. Here it would be appropriate to offer an example.

Say, you have an end-to-end path from some place in Japan to some place in U.S.A. Such path would probably use a trans-Atlantic optical fibre link to carry the traffic between the countries. In addition, depending on the locations of the end points of the path, it may span two or three additional hops on each side, some of which may be serviced by ISPs, universities, or private networks.

This would generally result in around 100ms one way end-to-end delay on such a path. Now, what could we change in the protocol stack? Say, we replace a layer 2 bridge by a full-fledged router somewhere at an intermediate node on the path. This will be at least 1 step up the protocol stack, since the router will have to look at the IP and TCP/UDP headers of each packet.

As long as out of 6-10 hop long path only one node is upgraded in such a fashion, this will truly be a negligible change in end-to-end delay. Thus, the approach used in traditional networking is understandable since it makes all calculations easier when processing delays within nodes along the path are neglected. This will be also considered when introducing particular probing methods later in this book.

Some traditional performance methods may choose to account for *performance degradation*. But even in this case the loss at each router/hop on an end-to-end path will be given in form of a constant value. In traditional performance models this makes sense, – whatever the performance loss may be experienced at each router, it is still negligibly small compared to the end-to-end delay even in each separate hop. This consideration drives many traditional performance methods.

1.3.2 NGN View at Transport Layer Performance

NGN takes a more detailed look at network performance from the view point of end-to-end network performance degradation. In some cases the change may result in approach opposite to that in traditional performance models, that is network delays will be neglected relative to the delays caused by NGN-compatible equipment at each node on the end-to-end path.

One example of an *NGN-compatible* equipment is *NGN application server*. NGN application servers will replace traditional routers with added functionality that will allow them to make complex decisions based on user-specified requirements. The complexity and decision making itself is expected to cause delays comparable to those caused by the low-level network infrastructure, thus, adding more sense to the opposite approach in the NGN network performance compared to traditional models.

According to the *TR-NGN-QoS* document developed by *ITU-T FG NGN* as part of *SGC10* held in 2005 [4], end-to-end network performance will contain the components in Figure 1.4. The only acronym *CPN* in the figure stands for *Connection Provider Network*. It is natural that a normal user will require a connection provider to take the traffic to the core network and bring back the replies. Connection providers will be connected to access networks which, in turn, will be interconnected through a core network.

Even if each step in Figure 1.3 would contain only one routing entity, *end-to-end path* would consist of 6 hops from one end to the other. In realistic cases the core network would probably have an ingress point and an egress point, thus, resulting in 2 hops only in the core network, while some access networks and even connection providers also have a complex multi-hop connections from user end to the Internet. Statistics show that an average length of an end-to-end path on the Internet is 6 to 11 hops.

The network design in Figure 1.3 may still accommodate traditional network performance models given that even today that is exactly how most of us connect to end-to-end services over the Internet. In this case you would add up delays on each hop plus fixed values for processing delays at each intermediate node.

In the manner Figure 1.3 is presented, NGN performance models will

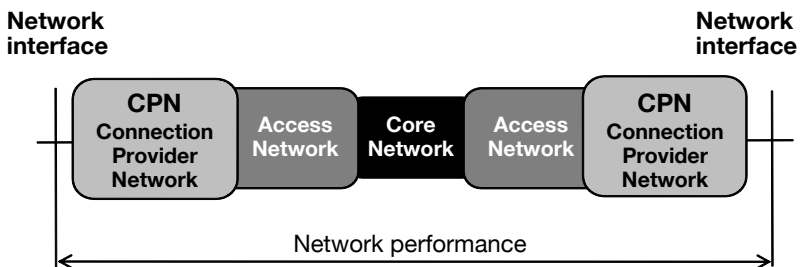


Figure 1.3 NGN components that form end-to-end network performance.

not differ very much from the traditional models as far as network performance models are concerned. The major difference will come with two more performance layers on top of network performance added within the NGN framework. They both will be covered in detail in the next section.

1.4 Application Layer QoS

Transport layer discussed in the last chapter seems not to have been affected by NGN at all. Of course, some subtle difference exists in the way transmission delays and processing delays along end-to-end paths are treated by traditional and NGN performance models and, especially, which of the two delays is charged with responsibility for performance degradation while the other one is completely neglected. Still, the real difference starts from the build-up on top of the traditional network performance, – something that does not exist in traditional performance models but is a very important part of NGN.

In terms of network performance, NGN reveals itself at the application layer and above, leaving the protocol stack below intact. This leaves us with two collateral assumptions. First, even under NGN we can treat all performance metrics as we used to in research on traditional network performance regardless of subtle differences in both performance models for the transport layer. In fact, all measurement methods developed for traditional networks are directly applicable to the transport layer of NGN provided they are used with caution when it comes to QoS guarantees. QoS is the main target of NGN and the application layer of NGN is given control over the transport layer specifically in order to provide a *reliable end-to-end QoS*.

1.4.1 Relation between Application and Transport Layers

New concepts in network performance start happening from the application layer dubbed “control plane” in NGN documentation. Here, at application layer, NGN introduces new players into end-to-end performance arena. Figure 1.4 shows components of an end-to-end path from the viewpoint of NGN. *TE* here is the new party and stands for *Terminal Equipment*. TE is attached to both ends of the path which is a reasonable thing to do.

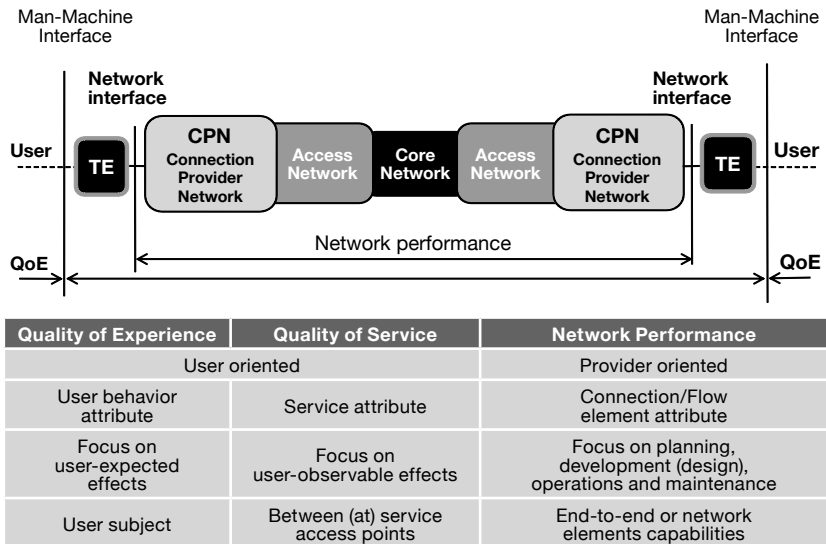


Figure 1.4 Key definitions related to network performance at transport layer.

Also, there are some additional metrics apart from the traditional network performance. Network performance remains solely in the confinement of the transport layer, while QoS and QoE stretch to the full extent of the path and incorporate TEs as well.

The *end-to-end path* itself in Figure 1.4 is novel from the viewpoint of traditional network performance. Traditionally, ends were not considered in end-to-end paths at all. In NGN, however, ends play an important role as the windows through which the user starts accessing an end-to-end service. In a very plain example, if you use your own personal notebook to access a service then this notebook automatically becomes your TE. Some properties of network performance including QoS and QoE are listed in table form in Figure 1.4 and should give a clear distinction among the three basic views at end-to-end network performance.

Since all three of them are partial reflections of network performance under NGN it is important to describe precisely which slots they occupy and how important they are from the viewpoint of measurement. The next subsection will do just that.

1.4.2 Application Layer Performance

This subsection gives detailed explanations about each of the three major network performance indicators, – *network performance*, *quality of service*, and *quality of experience*.

Network performance is a physical performance metric attributed to the end-to-end path without the TE. If to attempt a practical example, this could be the delay from the Ethernet socket in your home to the Ethernet socket in which your content provider plugs the web server you are accessing to get contents in the first place. So, your end-to-end path would have you at one end and the contents server at the other. If you drop your own computer and the contents server at the other end, then network performance is the physical performance of what is left of the end-to-end path. Given that there are still a few completely different technologies such as ATM, MPLS, and others, used at the physical level in networks, this area of network performance is quite heterogeneous and will remain so for several years to come. Hence it is necessary to measure it in order to infer end-to-end network performance characteristics.

QoS may not seem directly related to network performance, but it is an alternative way to represent network performance. As will be shown later, although in NGN standardization process the term QoS is used much more often than network performance, the metrics used to describe QoS are performance metrics, such as delay, jitter, etc. So, it is legitimate to state that QoS is the same as network performance. Only, as per Figure 1.4, QoS represents network performance of the end-to-end path including the very ends, i.e. the Terminal Equipment (TE). Bringing back the above example, by now you would connect your computer to the Ethernet socket at your home and would have your content provider do the same on the other end other path. And now, when the path is completed, QoS of the path can be measured and represented using the same metrics as were used earlier to define network performance.

Although in traditional network performance research, inclusions of ends to the path would not affect end-to-end network performance enough to change anything in performance, this makes perfect sense in NGN. Since control in NGN is put in an independent plane, path ends are responsible for controlling connection at the application layer. QoS constraints require guarantees which can be provided only after negotiations between path ends and possibly some intermediate nodes along the path.

The delays incurred by such negotiations are specific to NGN, which is why they were put in a separate category.

QoS is also natural for NGN. Since services are separate from transport, quality of service should include service clients and servers, which are placed into TE ends of the path.

QoE is a relatively new term developed recently but also independently from NGN standardization process. It is used in some other technical areas to represent the quality of man-machine interface, to draw a plain example. This is also natural for NGN as services in NGN are expected to be based on rich and extremely heterogeneous multimedia content. That said, it is important to remember that recipients of this content are normally humans which can have certain physical abilities as well as requirements when it comes to the quality of content. For example, one can tolerate only so much jitter in video streaming, once it becomes too “jerky”, the user is likely to stop the stream altogether or start a new one elsewhere. These are all descriptive of QoE as being currently defined within NGN. In fact, the standardization process in the area of QoE is very active in NGN today which indicates how important QoE is for practical implementation of services in the future NGN network. All the three above terms will be used interchangeably in this book with the exception of QoE, which is not directly related to network performance or methods used to measure it. In fact, the methods used to measure QoE are very different in nature from those that will be presented in this book. For example, Mean Opinion Score (MOS) is one of the methods actively promoted in NGN standardization process. MOS basically stands for a small statistical apparatus used to process opinions of service users pertaining to the quality of the service they received. Not only the opinions are subjective; but also it is important to keep in mind that even if quantitative characteristics of QoE are obtained from users’ opinions, it is very difficult to connect such results directly to network performance, especially to that at the physical level. The relation exists, but will not be explored in this book. At least a few years will pass before NGN standardization process will start looking for solid relation between the two. This book focuses on trying to create a rigid model of network performance which does not yet exist today.

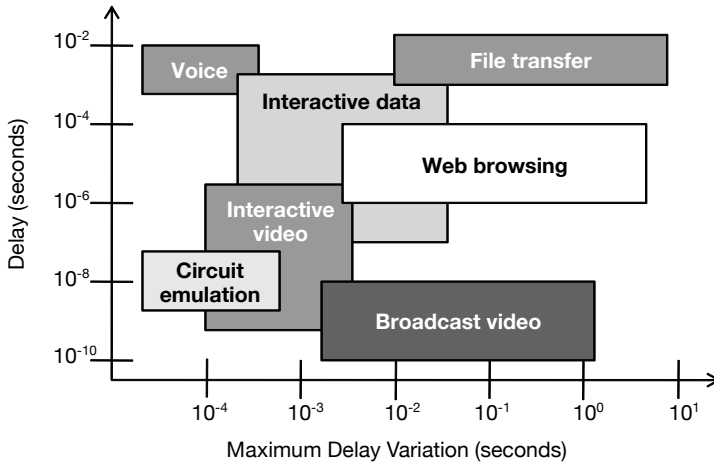


Figure 1.5 Some base parameters reflecting network performance at transport layer.

1.4.3 Performance Metrics

Having used the term *multimedia* in this book many times already, the term has not been clearly defined yet. Figure 1.5 contains a number of *multimedia applications* used in NGN standardization process today. Although people consider this term synonymous with audio and video, this is not always true in NGN. In fact, all information carried over the networks today can be considered multimedia. Web browsing in Figure 1.5 makes a good example.

Another good example is circuit emulation, which is the attempt to move conventional telephone land lines over to the NGN. In fact, given the ongoing process of shifting to all-IP networks, the circuit emulation is one of the important issues in large telephone operators in many countries, thus, becoming one of the most important multimedia in NGN.

One peculiarity about Figure 1.5 is that the quality of end-to-end connection required by these multimedia services is expressed in two basic transport-layer performance metrics – maximum delay variation and loss ratio. End-to-end delay, end-to-end capacity and other important application-level performance requirements are not even mentioned. This simplicity, however, makes it easier to define *QoS classes*, which will be expressed in these metrics and referred to several times throughout this book.

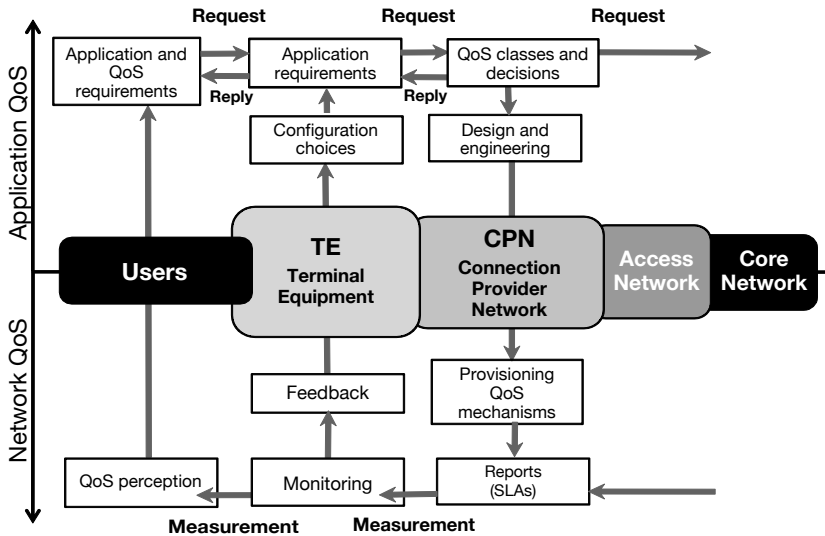


Figure 1.6 Interface between transport and application layers from the viewpoint of network performance.

Finally, given the inclusion of TEs in *end-to-end* QoS definitions, how much the performance is affected by the presence of TEs? This is a very good question, and it takes us to the heart of NGN and its proposed design for overall network. Figure 1.6 contains details on the interface planned among the various components of NGN end-to-end paths. This diagram was also developed as part of *GSC10* in 2005. The diagram in Figure 1.6 shows us how important are the insides of TE nodes. User together with its immediate TE forms a node that assesses QoS and QoE simultaneously through the feedback provided by the system.

How, the process goes like this. First, a user has to make some kind of requirement. In reality, this might be a software product that would only open a video stream if certain end-to-end delay requirement could be satisfied by the network. These requirements are sent to TE which forms application requirements from them. The two sets of requirements are fairly the same and they are definitely expressed in the same set of lower performance metrics. Past TE and delving deeper into the end-to-end path, application requirements translate into QoS classes. Now, QoS classes are different from application requirements in that the former are well defined by NGN standardization documents while the latter are left open for each application to decide for itself. In any case, the entire path

between TEs (not including TEs themselves) operates based on QoS class specified by the TE at connection setup.

TE also plays another important role – the role of the monitor for the QoS requirements made by the user in the first place. Thus, monitoring activities are included as part of TE's job. Finally, these monitoring results are offered to user for viewing which may result in a QoE estimate issued by the user.

The main subject of this book covers the contents of the *monitoring box* in a TE as it is the one in charge of *continuous monitoring* of end-to-end performance of the end-to-end path. The monitoring itself is a broad area and may cover both passive and active measurement of network performance, which will be discussed later in this book.

Of course, measurement methods discussed in this book are not limited to be implemented as a module of TE. Association with TE's monitoring block was drawn so that it would be easy to imagine the role played by active measurement in the future NGN network. Reality scenarios will require the use of active probing even in NGN network core, depending on the needs of each particular performance measurement task. Many such scenarios will be covered by this book.

Chapter 2



Passive Measurement Technology

The essence of *active measurement* is better understood by comparing it to its main rival in network performance management, – *passive measurement*. Although passive may sound like an opposite of active approach to measuring network performance, there are several points where these two opposite technologies overlap.

Major difference between active and passive measurement is in *performance metrics* the two technologies are capable to infer through measurement. Due to the natural limitations most end-to-end performance metrics are accessible only to active measurements while both approaches may be used to measure the rest. Metrics traditionally targeted by passive measurement will be discussed in this chapter along with respective measurement methodologies used.

A huge chasm between active and passive measurements is in the level of access to the physical network infrastructure. Since the knowledge of the physical topology of any closed network is in the hands of administrators of that network, any measurement that spans more than a single closed network is bound to resort to active measurement. A good example of a closed network is a local *Internet Service Provider (ISP)* that provides the access to the Internet to a residential area. That usage pattern gives passive measurement another name it often goes by, – network monitoring.

Naturally, both passive and active methods can be used simultaneously in the same network without a conflict. More than that, if handled

properly, multiple passive and multiple active measurements can be conducted in parallel without degrading each other's reliability. Such cases will also be discussed in this chapter.

2.1 SNMP/MIB Technology

Simple Network Management Protocol (SNMP) is the oldest and the most well established passive measurement technology in the Internet today. The first RFC dedicated to SNMP was released to public in 1990. The protocol has been actively developed ever since then which brought it to the level of SNMPv3, also known as *IETF Standard 62*. As was already mentioned, all standards within IETF undergo a long journey before they reach the level of *Internet Standard*. Not every RFC becomes Standard which can be gathered from the fact that the sequential numbering of RFC is about to go into the upper half of the first dozen thousand while there are only a few dozen standards. Quite often several RFC are released over several years by the process of refinement before they merge into a single Standard. This, in fact, was the case of *SNMP*, which has over a dozen RFCs dedicated to the protocol. The overall timeline of SNMP evolution will be presented at the end of this section while the section itself focuses on the traditional technology.

2.1.1 SNMP/MIB Principles

SNMP principle is very simple. Each *network device* performs *continuous* monitoring of its internal condition and advertises it in form of a structured set of variables. SNMP comes into play only when network manager needs to *poll* information about a given device's condition. That explains the word protocol used in SNMP. How each device monitors itself and how its condition is expressed in form of a set of variables is covered by other technologies. This makes SNMP approach a family of technologies used together.

Management Information Base (MIB) is one of technologies closely related to SNMP. In fact, it is difficult to imagine one without the other. Again, judging from the abbreviation, MIB is responsible for creating a structured view of variables that describe the condition of a network device. SNMP is then used to poll these variables. Practical implementation and related issues will be considered further in this chapter.

Before we delve into technological details about SNMP, some termi-

Table 2.1 Some SNMP terminology related to communications.

SNMP Term	Terms	Definition
Network element	Network device, network equipment, network node	A device connected to the network and made capable of receiving and replying to SNMP commands
Manager	Monitor	Manager monitor a party issuing SNMP requests to network elements
Agent	Meter	Agent meter software installed in network element and is responsible for continuous conversion of its condition into a set of variables

nology should be established. Table 2.1 contains some of the terms. Some more terminology will be established further on.

2.1.2 Client-Server Model

All network communications can be fit into a Client-Server model, client being the requesting and server the listening and replying part of this fundamental block of all possible network communications. For example, in case of a *browser* contacting a remote website somewhere in the Web, the Client-Server model is obvious, – browser is the client and *web server* is on the server side. Web server continuously listens to requests from browsers and upon receipt of such processes them and sends resulting content back to the client, i.e. a web browser.

In SNMP, each network element is the server, serving information about its state to the manager. The case of manager posing as the client in client-server model is quite common in networking. The justification for this is quite simple, – if manager wants to learn the state of a network element, it needs to ask. Similarly, each network element has to expect manager requests, and thus, has to listen continuously. Figure 2.1 visualizes client and server roles in communications between SNMP manager and a network device. In reality, practical management needs dictate slightly more complex communication patterns. Further in this section the communication model presented in Figure 2.1 will be extended. For now it is important to remember that SNMP-related communications involve network elements advertising their current state and managers polling states of multiple elements by sending SNMP commands to them over the network.

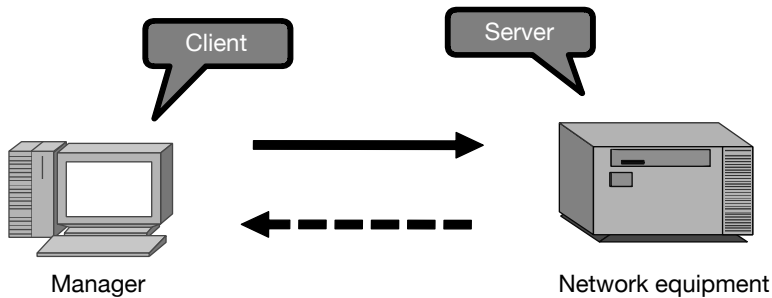


Figure 2.1 Client-Server model applied to communications over SNMP protocol.

2.1.3 Network Devices and SNMP

Having established that SNMP is responsible only for the communications part of the management, it is time to look inside network elements. The landscape of contemporary networks is very rich. Walking the hierarchy of complexity from top down, there are routers, switches, bridges, and hubs. When NGN fully converges it will bring along some brand-new types of network equipment. One of them is an *application server* complexity of which places it above contemporary routers in the above list.

Some secondary branching on each level in the above list should also be taken into account. Each of several prominent manufacturers of routers produces a number of series each containing several models, thus, forming a tree of router modes differing in complexity, functionality, etc. This difference is translated directly into major differences in variables advertised by each network element as its state. To draw a simple example, since hubs do not perform any routing, they do not translate IP addresses nor do they understand the concept of routing given that they operate at lowest level in *ISO protocol stack*. When one needs to monitor performance of a router, the analysis of IP addresses of packets flowing through the router is essential. Therefore, variables advertised by these devices should be essentially different.

The above list of network equipment is far from completion. In reality, any equipment somehow connected to a network, should provide some way for an outsider to monitor its state during its normal operation. This substantially prolongs the list of devices with some extent of built-in SNMP functionality. Every desktop or mobile computer with a Windows

or a Linux installation also has SNMP built into the operating system. IP phones, network printers, web cameras, etc. are only a few examples of a plethora of network-enabled devices that have SNMP pre-installed in them. The fact that they communicate over the network also makes them capable of network-based remote monitoring. Here, SNMP is a logical choice and has been such for two decades now, resulting in both a de-facto and the official IETF standard for a variety of network equipment.

The only solution to the issues of interoperability between software at the client and server sides of SNMP communications is to standardize the tree of variables to a given extent at the same time leaving room for proprietary extensions. That kind of tree was developed by *Internet Assigned Number Authority (IANA)* [6], – another active member of worldwide standardization activities. Figure 2.2 displays the very top part of the standardized MIB.

A standardized MIB is a vague definition. While MIB is the tree-like structure of advertised variables, each leaf in the tree is called an *Object ID (OID)* and constitutes the entire path from the root of standard MIB tree in both a numeric and symbolic forms. To cover both notations, the MIB tree in Figure 2.2 has both symbolic names and number for each node in the tree. The root itself is nameless and numberless, which is why is OID starts with a dot. For example, the internet OID could be written as .1.3.6.1 or .root.iso.org.dod.internet.

Although the tree in Figure 2.2 appears shallow, in reality it can go as

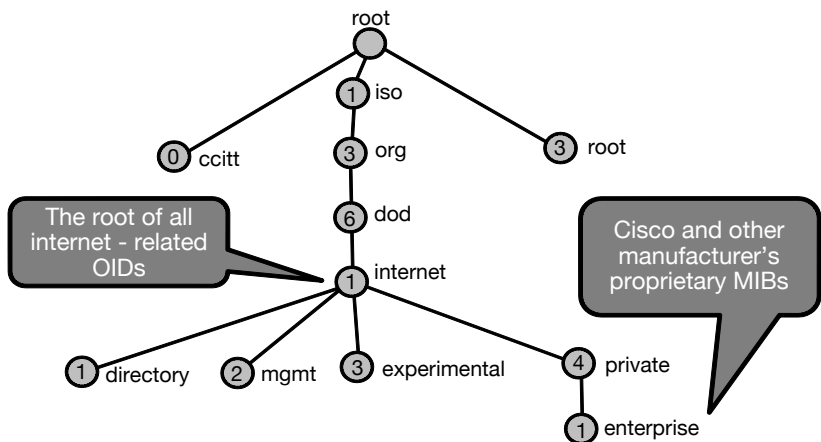


Figure 2.2 Part of the standardized tree of MIB OIDs.

deep as over a dozen hops from the top. As far as current standardization goes, the creator of a brand-new MIB has to decide from the beginning whether the MIB is to be proprietary and limited to use within a small group of devices or become a global custom practice to be shared by all SNMP users worldwide. Especially for this purpose, the standard part of the MIB contains leaves *private* and *enterprise*, which are normally populated by individual manufacturers.

2.1.4 Data Types Exchanged by SNMP

At this point the fact that MIBs contains the structure of information about the state of a network element is well established. Each variable has its own OID that a given MIB file is supposed to provide. The only remaining detail in the overall picture is the *syntax* of *MIB files*. Given that structured textual information is not unique to MIBs the syntax was borrowed directly from *ASN.1 notation*. ASN.1 reads *Abstract Syntax Notation One* and was created well before SNMP sprang to life.

Figure 2.3 contains an excerpt of ASN.1 notation used to write MIB files. Each declaration provides all the necessary information to place the new OID somewhere on a tree created the MIB file in question as an extension to the standard MIB root tree. The location on the tree is defined by the name at the head of the declaration and the number specified on the very last line of the declaration. All capital-letter keywords are part of the rigid notation and some values are also taken from the list of only options available for a given parameter.

Table 2.2 contains the legend for all keywords printed in all-capital letters in Figure 2.3. Values for *SYNTAX*, *ACCESS* and *STATUS* allow only a few conventional options which will be explained below. The very last line in the declaration requires special attention. The beginning of the line uses special *notation* interpreted by parsers as identifier for a location within the MIB tree. The notation includes *symbolic name* of the *parent node* and the second part is the numeric position of the newly created OID among children of the parent node. This allows for precise placing of each OID on the tree defined by the MIB file.

While values for *ACCESS* and *STATUS* are fairly obvious, *SYNTAX* keyword requires special attention because of its physical meaning. Similar to data type definitions supported by a programming language, MIB declarations also allow for only so many different *data types*.

At the lowest level of support by both MIB implementations and SNMP communications there are only three basic data types: *integer*, *octet strings*,

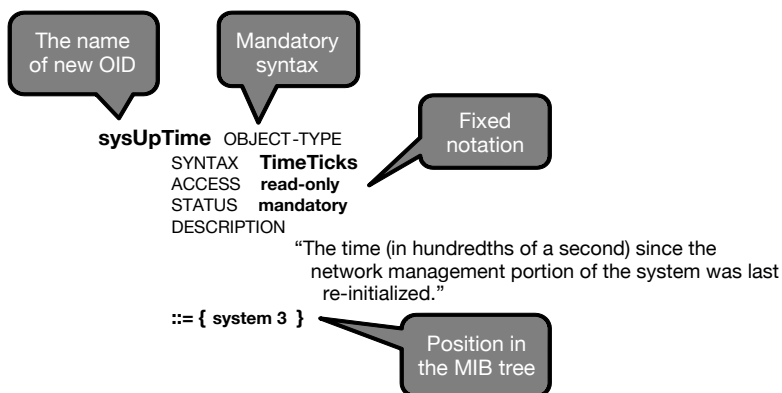


Figure 2.3 Example of ASN.1 notation used to declare OIDs within the MIB file.

Table 2.2 Meanings of each keyword in ASN.1 notation used in MIB files.

ASN.1 keyword	Description
OBJECT-TYPE	When found specifies the beginning of a new OID declaration, the word immediately before the keyword is interpreted as the symbolic name of the new OID
SYNTAX	Defines the type of the content identified by this OID, the list of data types will be provided below
ACCESS	Can only be read-only, read-write, and write-only, which define the three levels of access to the variable identified by the OID
STATUS	Obligations imposed on the implementation of this OID in the actual software, sort of a safeguard for missing support
DESCRIPTION	Arbitrarily long description of the OID, no syntax conventions

and *object* IDs. In practical words this allows to count and perform primitive arithmetic operations, to pass across textual information and, finally, to store and pass OIDs. However, due to much higher diversity required by practical applications of SNMP, there are several other data types created on the basis of the above three fundamental types.

Table 2.3 contains the exhaustive list of application-level data types.

Table 2.3 Application-level data types that can be used with the SYNTAX keyword in MIB declarations.

Application-level data type	Description
IP address	IP address written as octet strings and therefore have to be translated to and from bitwise content when writing to or reading from the variable remotely
counter	A non-negative integer value, the most popular data type in SNMP give instability to store numeric data; when value reaches the maximum value for integer, the counter is reset
gauge	A non-negative integer value that can accommodate values only within a certain range defined by min and max values, i.e. all values lower than the min will be stored as the min; no reset is necessary for this data type
time tick	A hundredth of a second since some event; a kind of a counter started programmatically upon the occurrence of a certain condition
integer	Plain integer value which accommodation both negative and positive values within the 32-bit bounds
unsigned integer	Unsigned integer, i.e. only positive integer values, and, thus, doubled upper bound, lower bound being, naturally, 0
opaque	Alias of “anything”, shuns interpretation of the data type and allows passing of arbitrary information which would not otherwise fit into any of the above data types

While the use for most data types in the table is obvious, some of them could use some elaboration. For example, *gauges* are useful when statistics of network processes are taken into consideration. Most processes in networks are hyper-exponential which defines a large scattering area for values. Imposing reasonable physical limits to such unpredictable fluctuations may help to perform simple monitoring of certain conditions in network elements. Another use of gauges, of course, if a *flag-based alarm* where even a singular occurrence of a given condition should trigger some actions. In this case the count of such occurrences is irrelevant.

The most popular of all data types listed in Table 2.3 are based on primitive *counters*. While counters are just simple integer values they are used to incrementally store various useful statistics required to perform successful monitoring of network elements. As a trivial example, counters are used to count incoming and outgoing traffic per network inter-

face, number of existing traffic flows simultaneously handled by a router, number of routing errors per unit of time, etc. Some practical uses of such counters will be considered further in this chapter.

2.1.5 SNMP Communication Patterns

There are three basic communication patterns within SNMP framework. Logically, since MIBs are used to define the list of variables that can be polled remotely, there should be a means to read and write variables remotely.

Figure 2.4 contains the three communication patterns SNMP protocol is capable of. This part of SNMP technology is the core of SNMP protocol itself which defines the commands that are sent over the network and define certain actions. These commands are referred to as *Programmable Data Unit (PDU)* in SNMP standards.

The level 3 protocol used to transmit PDUs over the network is irrelevant and is normally dependent on the actual infrastructure of a given closed network with an installation of SNMP. Using UDP for communications is a common practice, but SNMP can also be operated over TCP as well.

The PDUs themselves are *textual commands* that resemble the HTTP protocol used by browsers to download web pages. In a similar fashion, each PDU is a call for some data stored at the remote device. In the similar case of a web page, HTTP “PDU”s are used to get separate files from a web server.

Figure 2.4 does not explain individual PDUs, but rather gives an overall view of all possible communication patterns. PDUs can be easily attributed to one of these communication patterns.

Figure 2.4(a) defines the communication pattern required to read a variable from a remote SNMP-enabled device. In this case GET REQUEST PDU is issued and delivered by either UDP or TCP protocol to a fixed port on the remote device. Port numbers can be different depending on a particular SNMP “flavour” and can also be configured for each individual installation. The most common case is when remote devices listen on port 161 for SNMP PDUs.

Source ports used to transmit the GET REQUEST PDU is also important in SNMP because that is where the response to the PDU will be delivered. That part of SNMP is a bit unusual in view of traditional networking but is not impossible. The remote device simply reads the source port from the communication socket that was used to deliver the GET REQUEST

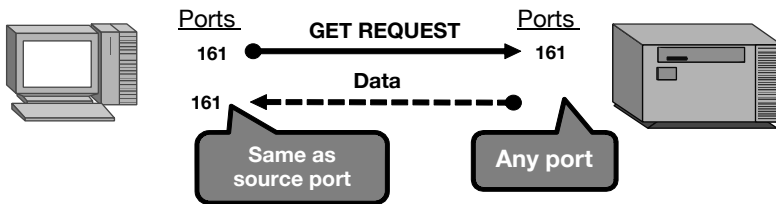
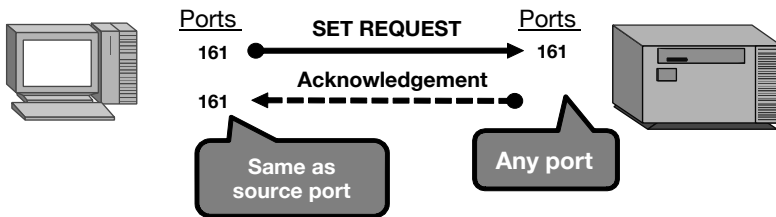
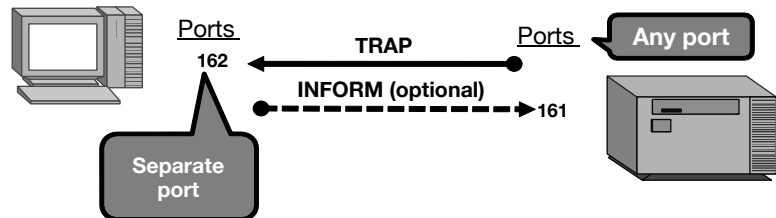
a Reading a variable**b** Writing a variable**c** Firing an alarm

Figure 2.4 Three types of communication within SNMP.

PDU and sends the reply to this very port. So, again, if port 161 is used on the reading machine to send PDUs the same port should be listened to for replies.

The second communication pattern in Figure 2.4(b) describes the procedure used to set variables on remote machines. **SET REQUEST** PDU is used in this case. The communication procedure is the same as the one used to read the variable. Of course, when you set the variable you have to make sure that you declared it as **ACCESS read-write** or **ACCESS write-only** in your MIB. In any other case it is impossible to set the variable. This is

not really a security measure but more of a personal safeguard that will impose certain limitations on the handling of your remote variables. This makes ACCESS declaration more of a design issue.

The third and the more important PDU used in SNMP uses the third communication pattern in Figure 2.4(c). Its clear distinction is in the reversed direction of communication. Also, the name of the PDU, TRAP, speaks for itself. This PDU is used to send alarms from remote devices to the management centre *asynchronously*. The asynchronous delivery is very important for network performance monitoring as it can notify the management centre immediately when an *anomalous condition* occurs.

SNMP TRAP also uses a separate port for TRAPS. Again, configuration may be different from different implementations of SNMP or a particular installation, but the most common case is to use the next port, – 162. The fact that a separate port is used gives a major advantage in monitoring. Management centre can choose to create a separate program to monitor only SNMP TRAPS so that those are detected and acted upon early.

Optionally, recent versions of SNMP allow for an additional INFORM PDU, which is used to acknowledge the receipt of SNMP TRAP to the remote device. This facilitates a more complex logic in communication between management centres and individual devices.

There are a few more PDUs in use to both set and get variables. They are normally created with the intention to simplify reading of remote variables. In fact, in conventional SNMP systems, the ratio of reads versus writes is very high, and in many cases only reads are used with no writes at all. This makes sense for most performance monitoring applications. So, the diversity in PDUs falls mainly on the part of GET PDUs. Table 2.4 contains the list of PDUs.

Finally, to demystify practical implementation of SNMP completely, Figure 2.5 contains the *modular structure* of SNMP components used on both sides of SNMP communications. There are three keywords used in SNMP standards: *Network Management System (NMS)*, *agent* and *managed device*. The role of NMS is quite clear, – it is used to poll performance data from various locations in the network, possibly perform some kind of analysis of this data either online or offline, and catch all traps directed to it from individual managed devices.

The distinction between managed devices and agents is also logical. Having a remote networked piece of equipment does not necessarily mean that it is SNMP-enabled. SNMP agent is what makes a device SNMP-enabled by speaking SNMP protocol to NMS. Agents are also smart components in SNMP communications. They do not only communicate PDUs and data to and from NMS, agents are also in charge of collection of all

Table 2.4 The list of all PDUs currently implemented by SNMP.

PDU	Description
GET REQUEST	Used to retrieve contents of a single MIB variable
GETNEXT RE-QUEST	Used iteratively to retrieve sequences of MIB variables (very helpful for walking portions of MIB structures)
GETBULK RE-QUEST	Even better iterator used to get multiple variables in one response
SET REQUEST	Used to set a remote variable located by its OID
GET RESPONSE	Regardless of the name, used to reply to both GET and SET PDUs, returns contents of MIB variables for GET, and acknowledges status for SET PDUs
TRAP	Used to sent an alert to the management system based on the occurrence of a certain condition on the remote device

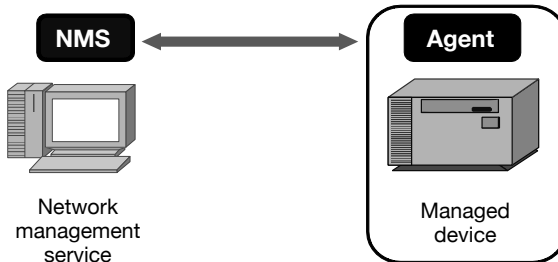


Figure 2.5 The three roles within an SNMP-based monitoring system.

variables defined by one or many MIBs the agent has loaded at startup.

This design allows for a very high level of flexibility. Depending on the type and purpose of a particular device, MIBs will be different but the same agent can be used for all.

A good example is in place here. Consider a desktop machine with Windows OS installed on it. Performance-wise, there are many ways to obtain information about the performance of your own Windows machine, such as how many CPU cycles are used by which program, how much RAM is free, etc. While there is a GUI-based way to obtaining this information in Windows, there is also a programmatic way to do the same

thing. In other words, we can get CPU and RAM usage of our Windows machines in form of variables, or, getting closer to the SNMP terminology, MIB variables.

This small project could be implemented in the following way: a standard SNMP agent would be installed first. A MIB would be written declaring OIDs for CPU and RAM usage separately. Many of such performance indicators may already be available in your standard SNMP agent, but even if not, agents are normally written with some level of extensibility. Thus means that if a particular performance indicator is not covered by your agent, you can write your own program that will be called by the agent in order to obtain a given performance variable.

So, we could write a small program that would check CPU and RAM usage when called by the agent. The final step would be to poll this data remotely. This is a simple step because all we have to do is to issue SNMP GET REQUEST PDUs from some other location in the visible network and wait for GET RESPONSE PDUs from the agent we earlier installed on the Windows machine.

However simple this may sound, even large SNMP-enabled systems are created in a similar way. Some patterns of performance monitoring with the help of SNMP will be considered later in this chapter.

2.1.6 SNMP Timeline

As a technology, SNMP is still a very active area of research which is proved by Figure 2.6. It depicts the timeline of the entire SNMP standardization process started in 1990. The process can be split into several steps each containing a major improvement in the technology and maybe the release of a new version. The most important event in the life of SNMP technology is the creation of IETF *Standard 62*, which is in effect today. Standard 62 is created from a long list of RFCs that cover various aspects of SNMP technology, including security. In fact, security in SNMPv2 was still very underdeveloped, – a serious problem in constantly growing security concerns of the Internet today.

Standard 62 is also known as SNMPv3 which is considered the only mature version of SNMP and is therefore recommended over all other versions. IETF also uses the term “deprecated” to describe the 2 previous SNMP versions and discourages their use for network monitoring. The primary reason for such sternness is the growing number and complexity of security threats which SNMPv3 is presumed capable of coping with.

It happens, however, that the ratio of SNMPv2 currently in use is much

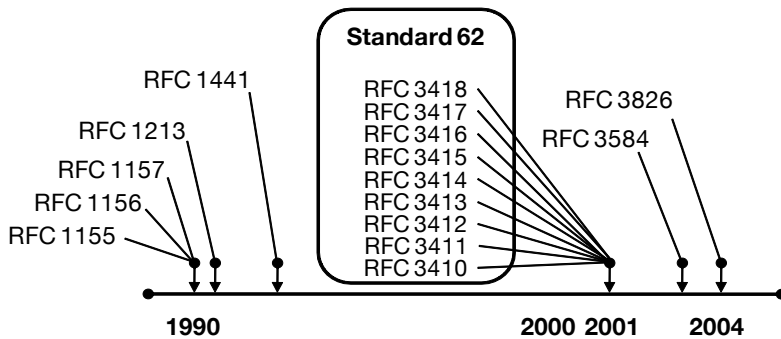


Figure 2.6 Timeline of *SNMP standardization process*, including the development of *Standard 62* which is currently in effect.

higher than that of SNMPv3. The reasons for such lingering are unclear but could be the unwillingness to invest into technological and programming effort related to the shift to a newer version.

2.2 NetFlow Technology

NetFlow [23] is yet another way to look at passive network monitoring. In fact, this approach is completely different from SNMP in that NetFlow is concerned only with *traffic* and is completely indifferent to performance parameters related to the networking equipment. As in the protocol stack transport layer is above the data and the physical protocol layers, NetFlow also a build-up on the top of low-level physical performance parameters.

Apart from the traffic abstraction introduced by NetFlow in form of traffic flows, NetFlow is also very different from SNMP in its approach to getting access to the collected statistical data. There are legitimate reasons for NetFlow's approach, however, and most of them will be covered in this section along with ample examples.

2.2.1 Traffic Collection Process

Figure 2.7 contains the very basic presentation of NetFlow architecture. Like in the case of SNMP, network devices in put in the centre of the monitoring process by the similarly ends there because all NetFlow cares about are traffic packets.

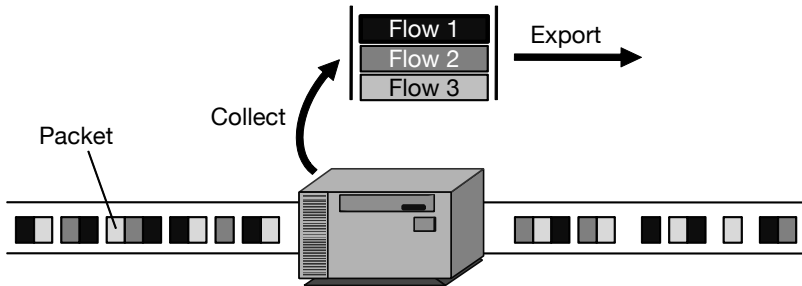


Figure 2.7 Overall process of traffic flow collection in NetFlow framework.

To be more specific, NetFlow is generally used to monitor *layer 3* traffic, i.e. transport packets. The two *transport protocols* encountered at this layer are *TCP* and *UDP*, but there are many other protocols used to carry traffic over IP networks. Above the transport layers are session protocols, such as *HTTP*, *SNMP*, etc., but to NetFlow these protocols are transparent and require additional processing if need to be analyzed.

Now, from the basic networking theory we should know that protocol stack can be found in each packet by looking at its *header*. In fact, there are normally multiple headers in each packet and each higher protocol header is normally wrapped in a header from the lower layer protocol before it can be either dispatched yet lower on the protocol stack or physically transmitted into the network.

This concept of *header wrapping* is exploited by NetFlow. In fact, NetFlow is not the only tool that analyzes *packet headers*, – *tcpdump* is a very old Linux tool that can be used to dump traffic to a file in near real time. NetFlow installs a similar tool on network equipment but performs additional processes to generate traffic flows.

When headers boil down to numbers, 64-byte-header dump is the most common. In this case the first 64 bytes of each packet are dumped to a file in a specific format that can later be re-opened and analyzed offline. There are many “specific” dumping formats most of which use some kind of compression to save disk space used for the dump.

While the 64-byte dump guarantees that your dump will include IP and TCP/UDP headers of each packet, 96-byte dump can offer some insight into sessions, i.e. another level up on the protocol stack. Naturally, it takes more time to dump and later analyze these packets, which makes this dumping less popular than the traditional 64-byte dumping.

NetFlow does not dump packet headers but instead analyzes them in

real time as packets arrive and are processed by the network device. *NetFlow* is interested in packet headers only so that header information can be matches to the table of existing *traffic flows* and possibly result in the update of a line in this table. Regardless of whether a database entry is updated or not based on the current *packet header*, the header is purged while the packet itself is processed by the device in the expected manner. That is, if the network device is a router, the packet has to continue on its route unaware of the fact that its header has been looked at possibly saved in the database in form of a traffic flow.

In the end, NetFlow-enabled devices may contain a large table/database of existing traffic flows, each uniquely identified by the information extracted from packet headers. The term “existing” traffic flows is a stretchable term and can normally be configured depending on how long into the past network device has to allow traffic flows to be retained. This is referred to as “*flow timeout*” and in most cases is 5 minutes, literally meaning that a 5-minute pause can be tolerated on any traffic flow.

The concept of NetFlow export shown in Figure 2.7 will be covered later in this section.

2.2.2 IP Flow Tuples

What was earlier referred to as packet headers in traffic analysis goes by the name of “tuples”. Specifically, the case of 5 tuples is the most common.

Figure 2.8 contains visual presentation of the *5-tuple flow*. There is source IP address, source port, destination IP address, destination port, and, the fifth tuple is the protocol. Some analysis techniques discard the fifth tuple as redundant given that certain source or destination port numbers already contain the information about the protocol, but this tuple is needed if general analysis is to be performed. For example, if you are looking at the flow that has 80 as its destination port, in most cases this would mean that the flow is a TCP connection carrying the contents of a web page on the return path of the connection.

Now, the fact alone that the words source and destination are used to described traffic flow tuples is ample proof that NetFlow is dealing with *unidirectional flows*. So, the return path in Figure 2.8 automatically becomes an independent flow and occupies a separate line in the table at the network device. Some traffic analysis techniques deliberately purge directionality in order to achieve certain targets. The most common case is that of a unidirectional flow, most probably because this is the default operation mode of NetFlow.

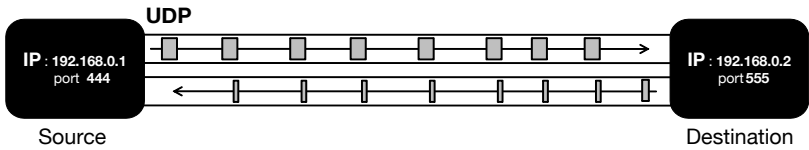


Figure 2.8 Visual definition of 5-tuple identifier of traffic flows.

Basic information provided in NetFlow export can be presented by the shortlist of the following 7 items, including the basic 5 tuples:

1. Source IP address
2. Destination IP address
3. Source port (for example UDP or TCP port)
4. Destination port (for example UDP or TCP port)
5. IP protocol
6. Ingress interface
7. IP Type of Service

While 5 tuples are sufficient for most flow-based traffic analysis, NetFlow is fairly rigorous in providing additional information on top of the 5-tuple base. The list below contains the list of most common information offered for each flow in NetFlow export. The technology of the NetFlow export itself will be considered later in this section.

- version number
- sequence number
- input and output interface indices
- timestamps for the flow start and finish time, in milliseconds, relative to the last boot of the remote device
- number of bytes and packets observed in the flow
- layer 3 information
 - source and destination IP addresses
 - source and destination port numbers

- IP protocol
- Type of Service (ToS) value
- in the case of TCP flows, the union of all TCP flags observed over the life of the flow
- IP address of the immediate next-hop along the route to destination
- source and destination IP masks

2.2.3 Overall Monitoring Architecture

Figure 2.9 displays all activities possible within the NetFlow architecture. There are three basic activities: *export*, *storage*, and *analysis*, where only the export is pure NetFlow and is proprietary while the other two activities are open to public and are implemented in many *traffic analysis tools*. Some of these tools as well as the general usage practice will be considered in following chapters of this book.

The pure NetFlow activity of export is the opposite of what is done in SNMP architectures. As was explained earlier, SNMP is basically a combination of counters that have to be read remotely by the management centre. This architecture had the intrinsic problem of having to read a counter two times and retain at least one its previous value in order to be able to make calculations based on the difference. Counters often *wrap*, which requires additional logic in processing such readings.

With NetFlow the process is fairly trivial. All the management centre has to do is provide a separate machine that will accept traffic flow export from NetFlow-enabled network device. Of course, the device itself has to be configured so that the export is directed as a specific dedicated collector in the network.

The *collector's* job is also trivial. The traffic export it receives is normally sent to another place where it is stored. Optionally, the storage and collection roles can be merged into a single machine if the scenario is not very *performance critical*. Many NetFlow exports carry huge bulk of data and may even stall collector if the performance issues are not properly addressed beforehand.

Now, once the export is collected and properly stored, it has to be analyzed. Taking into consideration performance requirements on the part of collector and storage machines, analyzer is often a separate de-

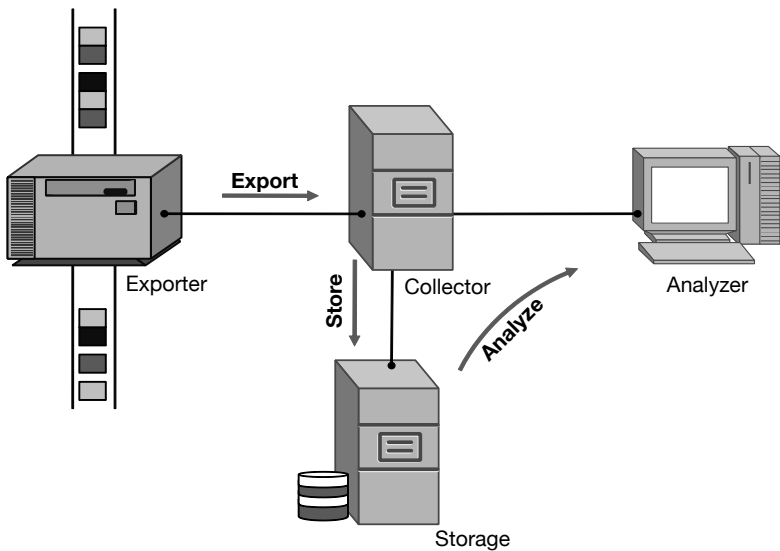


Figure 2.9 Overall NetFlow architecture with all involved roles.

vice, too. If the physical monitoring architecture is close to the one in Figure 2.9 there is even a chance that analysis of NetFlow export can be done in *near real time*.

Recently, the real time component of traffic analysis is becoming increasingly important. The layout in Figure 2.9 is particularly open to various additional installations. For example, one might catch early traffic conditions by performing a quick scan of a few traffic flows in the export. Such processing logic can be placed on the collector prior to the long-term storage of the data. In fact, many traffic analysis methods exploit the flexibility of this architecture in just this fashion.

In fact, the architectural flexibility renders NetFlow more attractive than SNMP in areas where both technologies can be applied to fulfil a monitoring objective. Several examples related to such competitions will be considered later in this book.

Chapter 3



Passive Measurement Tools

Quite literally, relationship between active and passive measurement tools is very akin to the standoff between the quickly becoming popular Firefox and the more traditional Internet Explorer browsers. At the time this book was being written, Firefox occupied only about 13% of the market while IE covered 83%. Literally, IE occupies almost the entire remaining user base. All the other browsers, including Apple's infamous Safari browser are left with but meager crumbs of Internet users worldwide.

At the fear of being insufficiently articulate, the reason the above comparison is used lies in the set of features offered by Firefox versus those found in its yet overpowering rival. As one example of many, in the web-oriented world nowadays, web applications are becoming extremely important for any kind of business as well as for scientific needs. Web 2.0 is another keyword floating in the air for several years now, and brings about some major changes in how things are done in the web. In short, web is quickly becoming home for fully fledged applications just like those that we used to install into our operating systems.

Now, back to the comparison between active and passive tools, – just as Firefox offers rich facilities for debug and analysis of client-side web applications, active measurement tools are there to give you a better insight into the performance of your network. Like Web 2.0 offers higher level of interactivity between users and web applications, active measurement tools are all about efficient online analysis versus offline analysis of a large bulk of data generated by passive tools.

The passive side of this small standoff in the measurement world is considered in this chapter through several tools that exist in the area today. The active side of the story will be uncovered in the final chapters of this book.

Given that there are only two categories of passive monitoring techniques described above, they translate into two main groups of *monitoring tools*, – a tool can be either *SNMP* or *NetFlow*-based. In practice, though, the split is more like *MRTG* versus *NetFlow*, where *MRTG* stands for Multi Router Traffic Grapher and is the name of the tool that for many years used to be the only popular *front-end* of *SNMP*. *NetFlow* is almost always just *NetFlow* and there is a good reason for it, – *NetFlow* in itself is a standalone standard that defines the totality of various aspects of packet flow monitoring and there is really no need to invent a new way of doing things at the front end. Mainly due to the complexity of MIB structures *SNMP* PDUs and the overall way of working things in *SNMP* has to be standard across multiple platforms. This is why *MRTG* quickly became a midway standard for interfacing *SNMP*-enabled network elements.

At last, the level of access one has to the back-end of traffic data is also very important when creating a traffic monitoring system. The larger is your system in terms of the number of monitored entities, the more flexibility is desired and that is where it makes a difference which tool you use.

As a quick example, say, you buy a Cisco router that is *NetFlow*-enabled. You will then have all the freedom within the configuration offered by the firmware version of the *NetFlow* installed at the router. Only so much freedom, though, since native *NetFlow* implementations tend to be a bit restrictive in their setup.

You can also find a small number of *NetFlow implementations* in the public domain. Those are normally products of academic research and tend to be a lot more flexible in configuration. When you have to collect *NetFlow* data from multiple points simultaneously and preprocess data in an unusual way, a public tool is often your only choice.

All the above issues will be considered and given ample examples in this chapter.

3.1 Common Design Patterns in Monitoring Tools

Both SNMP and NetFlow standards go no further than the back end of any *monitoring scenario*. Traditionally, that would be some *network device*, – a router, switch, or a web server, to name just a few. This dictates that the rest of the process should be located at a remote location on the outside of the device. Some of these design patterns are considered in this section.

3.1.1 Roles in Monitoring Process

Figure 3.1 shows that there is in fact yet another location outside of the device that participates in the passive monitoring process. That part is the user interface that is normally implemented as a web service and uses a web browser to access the data produced by the middle device, the collector.

Let us consider the role partitioning in Figure 3.1. The first role, the network device, is self explanatory. It contains factory-shipped and often proprietary implementation of SNMP MIBs and/or NetFlow that prepares the raw data at the back end, hence the term. That part might be configurable at a certain extent but there is usually little the user can do about the raw data collection. Most data manipulations happen in the middle, at the collector.

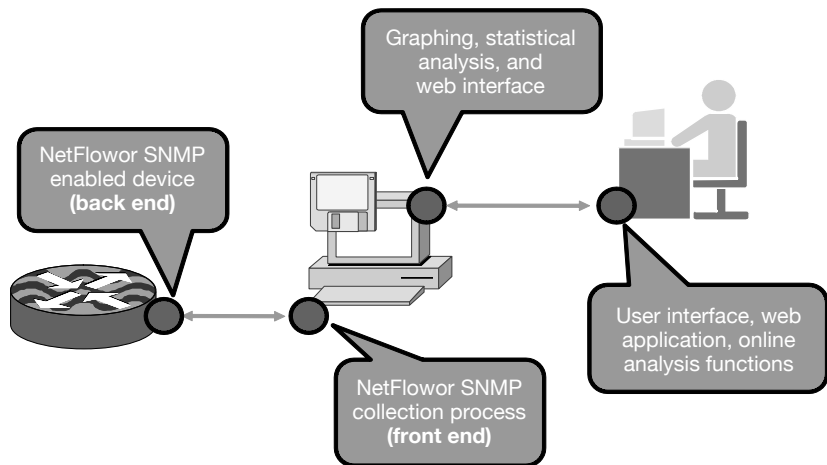


Figure 3.1 Common players in passive monitoring scenarios including traditional back end, indispensable front end, and fairly common web-based user interface.

Now, the collector role in the middle of Figure 3.1 is the most important part of the process since it is in charge of all manipulations that are to produce human-readable versions of raw data. To give a plain example, an estimate of a server's traffic load is impossible to make without two discrete reads off a back end counter. The collector takes care of such operations automatically while without it everything would have to be done manually at the user's side, thus greatly increasing the human work load.

Finally, *user interface*, – the rightmost role in Figure 3.1 is all about user-friendliness. Evolving on the simple example above, a server's traffic load might be a valuable piece of statistics but it is also important to know when that particular statistics happened at the device. When dynamics of a given performance statistic are important, the role of the user interface is just as important as that of the collector process. All existing tools implement some kind of automatic refresh. Although not very technically efficient, it is normally implemented in form of a Javascript-induced automatic page refresh in a browser. Most tools make *autorefresh* interface a configurable parameter, which might be important for some monitoring tasks. Browser throughput, however, imposes certain limitations on how often a page with performance statistics can be refreshed. NetFlow is normally more sensitive in such situations because the raw flow data is normally a lot larger than SNMP statistics.

3.1.2 Loosely Coupled Monitoring

Although the three above players remain unchanged in virtually any passive monitoring scenario, their relative installation locations do. While it is common to use a loosely coupled architecture as per Figure 3.2 in order to split processing load, quite often putting them all together on a single device, i.e. using a *tightly coupled* architecture, is the only option. In particular, examples given later in this chapter all use the latter.

A little bit about notation used in Figure 3.2. “Any” stands for any port and “same” means that the port used for the return traffic is the same as the port of the connection the reply is being sent for. This peculiarity of SNMP operation was already discussed before. After all, ports are not as important as understanding the architecture. For instance, NetFlow uses different ports, but there is no academic point in creating a separate illustration for NetFlow given that the overall architecture is identical.

What happens in loosely coupled architectures is quite simple. The meter (back end) is located where the factory installation placed it in the

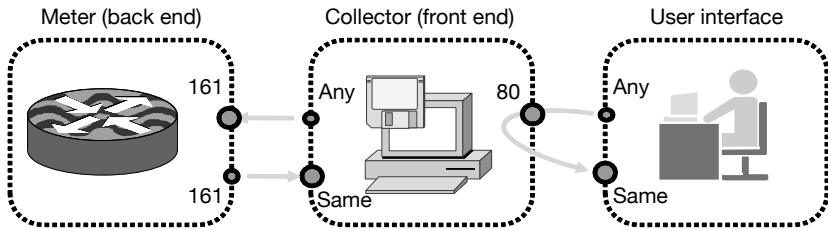


Figure 3.2 Loosely coupled architecture involving the three main players in passive performance monitoring.

first place, i.e. inside the network device itself. Although this was already mentioned before in this book, it is worth mentioning that majority of devices that can have a network connection normally have at least some level of SNMP support built-in. NetFlow, naturally, is built into devices which handle an amount of traffic worth the effort of analysis. SNMP is more common and if you dedicate enough time you can even find out that all recent versions of Windows operating system have a built-in SNMP meter.

Now, collecting this *raw data* is another story. If you bought a Cisco device and integrated it into your network, you do not possess the luxury of installing the collector software into it. There are many reasons for it. For starters, the collector software, as was mentioned before, is not the integral part of either SNMP or NetFlow standard and is thus a separate software product written independently by an individual or a dedicated company. This manner of software development puts a limit to the number of operating systems such software may support. An overpowering majority of software products in the area of passive network monitoring are written for a Unix or a Linux-based machine. Both are similar unless you do something unusual in your software and have to depend on an external library. If this is the case, Unix and Linux take detours from each other's paths and have to be handled separately. As will be mentioned later in this chapter, Windows implementations of SNMP/NetFlow collectors or any related software are extremely rare with only several products existing today.

Ports used for communication between the user interface and the collector are unusual and take root in the web application technology. Since that communication happens over *HTTP protocol*, only one connection is established for this communication originating from the user interface and terminating at the collector. All the data for each request is sent back

piggybacked to *ACK packets* within the same TCP connection. This is the property of any web request and is no different in the architecture from Figure 3.2.

3.1.3 Tightly Coupled Monitoring

Figure 3.3 displays an often case when a single machine is used to host all three parts of the passive monitoring process. This could be a cheap solution for monitoring a web server or a router built in a Linux machine. In non-commercial networking these cases are the commonplace and therefore are worth a quick mention.

In a slightly broader picture, the *user interface* part of the process does not have to be installed on the same machine with all other parts as the user interface is confined within a browser window, thus, fitting into capabilities of virtually any personal computer today.

For the overall system, though, this would make a huge difference. The below explanation will be very clear to those that have already had

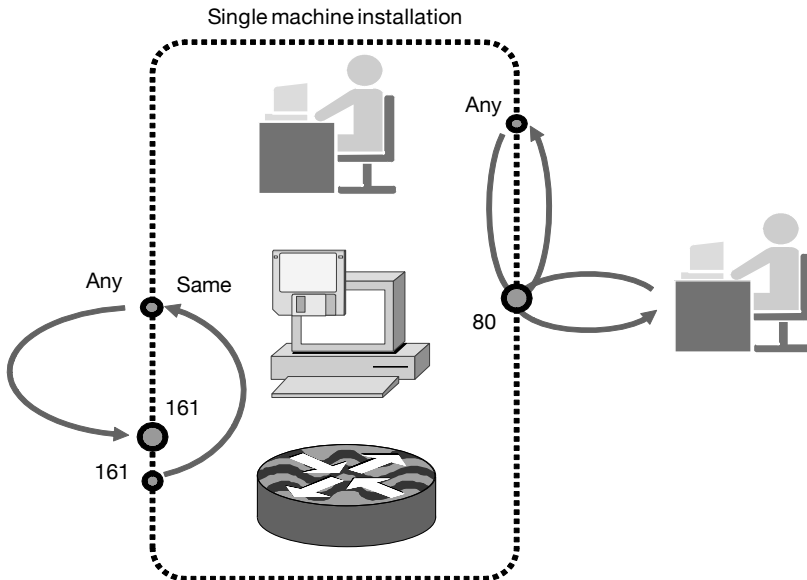


Figure 3.3 Common architecture used in cases when a single machine is to host all parts of the passive monitoring process.

experience putting together such a system, and the rest will have to bear with slightly excessive level of detail.

When you are building a standalone monitoring system on top of a network device that you yourself created in the first place, there are a few major differences from the case when you buy and use some commercial network device:

- you make your own network device, such as a router, a switch, a bridge, or a server, on top of an existing *operating system*; most Unix or Linux operating systems already have these functionalities built in and require only a minor configuration effort to make them work;
- since you provided the physical machine to turn into a network device, you have the freedom to install any additional modules, better yet, you can use the same operating system if you are using a wide-spread system like some flavor of Linux, you can find a monitoring tool written and put into the public domain for you;
- as the third step, i.e. the user interface, installing a web server (Apache, for example) is again a trivial task, and will successfully complete the installation.

Now, whether you integrate user interface in the same machine or not makes a huge difference for the operating system you use to make a network device in the first place. It is common practice to make such an operating system as *lightweight* as possible. This normally ensues that the GUI components of the operating system are not installed, limiting administrators to a *command line*. Since such machines are normally maintained through a *remote terminal*, the absence of the graphical component is never the issue. The command line is sufficient to install a web server, this can even be done remotely, uploading, decompressing and installing software components from source files via a remote terminal.

When you start using a web browser to view the statistics coming from your performance monitoring web application, that is when you need the full graphical environment in your operating system. Hence the tradeoff where you should consider which is easier for you, – to install a much larger version of a unix-like operating system you use for your device, with graphical environment and a set of applications including a web browser, or to use a small footprint of an operating system for the device while relying on other desktop/notebook computers to read per-

formance statistics on the other end of your performance monitoring web application.

The latter is a much more common case in practice. When you download a free software or even write your own web application for the collector role in the monitoring process, you should have a few practical uses of human-readable performance statistics in mind. You might want to put them into a power-point presentation, integrate graphics obtained from the web application into a document for later use, etc. All these things are easier when done on a personal or a company provided computer you use permanently. Interestingly, regardless of the fact that most actively developed Linux flavors like Fedora have nearly the same set of office, presentation and documentation software, most network administrators still use Windows to generate documents from performance statistics.

3.1.4 Distribution of Load in Monitoring Systems

Naturally, if you create from scratch a network device that you plan to strain a lot and expect high performance from, you may want to use a distributed system instead of cramming a single machine with software that might have conflicting needs in CPU cycles.

Although all the above illustrations exhibited a clear cutoff between the collector and the user interface roles in the passive performance monitoring process, the complexity of inner handling of data blurs the borders between the two roles. Additionally, this vague area of shared affiliation contains a few processes that work in parallel. It is necessary to mention, though, that regardless of the multiple processes running at the same time it is very rare for existing performance software to implement them as multiple threads. This direction of development is expected from software vendors in the future and is expected to make the overall process and specifically the user interface a lot more traffic and delay-efficient than it is today.

Figure 3.4 contains the visualization of all processes running in parallel and shared between the collector and the user interface roles in the performance processes. First of all, human users do not necessarily trigger all processes in this part of the monitoring software. There are a few necessities that require regular updates regardless of whether a user is using the web application in question at the time. Some specific examples will be given later in this chapter, but just as a quick example, to provide hourly, daily, weekly, and monthly statistics generated from raw performance data it is necessary to perform regular updates in the background

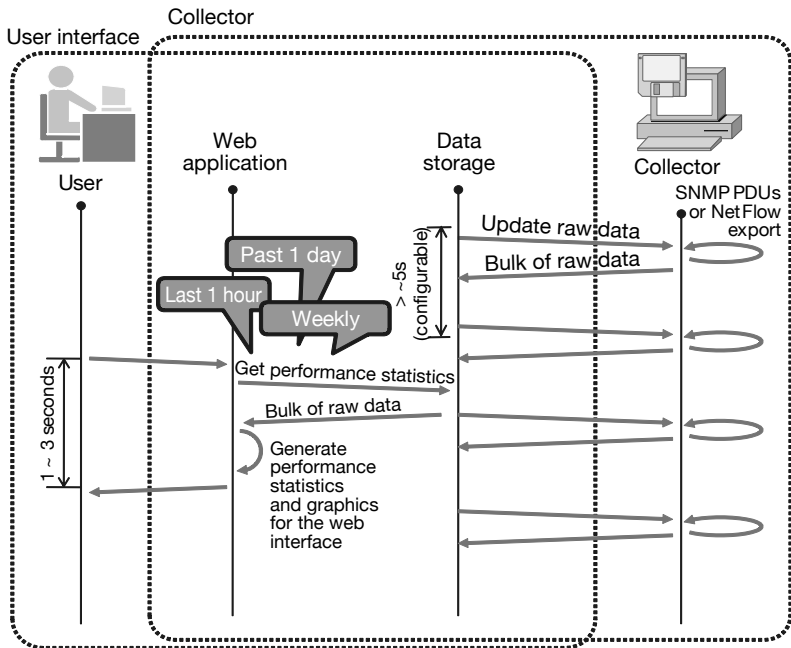


Figure 3.4 Synchronization and procedures between the collector and the user interface roles in passive monitoring processes.

even in the absence of user activity through the web interface at the time.

These regular updates are performed between the Data storage and Collector parallel processes within the software. Data storage process wakes at regular intervals and calls Collector requiring it to collect SNMP/NetFlow statistics from a remote location. *NetFlow* communications are a bit different from *SNMP*, but since the processes do not go beyond the collector, the generalization is still valid.

It may be specific for a particular performance monitoring software, but all the logical threads in Figure 3.4 are rarely implemented as separate threads. This elevates the problem of parallel programming but creates an event-based procedural environment which is prone to excessive delays.

When a user requests a page with a given set of performance statistics, raw performance data has to be compiled into graphical plots by the web application before the page can be served back to the browser. This procedure incurs major delays. This procedure is also very common across various performance monitoring software products mostly due to the fact

that automatic generation of web-viewable plots is the area previously covered and still very actively developed in the Linux world. The library used by most tools is called a *GD library* and has actively supported ports for most platforms thus providing a compatible environment across all of them.

The GD library is a drawing *API* and automatic generation of plots based on numeric data is only one of potential uses for this technology. The final target is the same for all uses, though, – GD library's output is traditionally a GIF image. GIF graphics format is one of the oldest graphics formats in the web and is unlikely that one can find a browser that does not support this format. Therefore, this technology is universal for any environment used by the user to access human-readable performance statistics.

It is necessary to note that GIF is a proprietary format of UNISYS and permission should be properly acquired prior to using this format in any manner. Both in the GD library itself and in all the software tools considered below, it is unclear as to whether this permission has been obtained.

"Flavors" of the GD library also exist that are able to generate PNG and JPEG graphics formats in place of GIFs. PNG is a lossless format and is advised for scientific plots, while JPEG will create smaller files at the cost of some data loss.

The problem with data visualization shared by all performance monitoring tools is that all tools generate plots as raster graphics and never as vector graphics that only recently have obtained support by major browsers. This is also a foreseeable direction of development on the part of actively developed performance monitoring tools and should definitely apply to the examples considered later in this chapter.

3.1.5 Specifics of SNMP

As a summary, Figure 3.5 rounds up all the main features one can expect and preferably demand from an SNMP monitoring tool. There are four main tracks of features.

MIB library is the most important feature of any SNMP-based monitoring tool. While SNMP is the protocol and MIB is a performance metric description language, practical implementation of both varies across tools. Here, it is necessary to mention where MIBs come from. MIBs can come built into a network device, it can be a commercial solution that will include the MIB itself and its implementation on the network device, or,

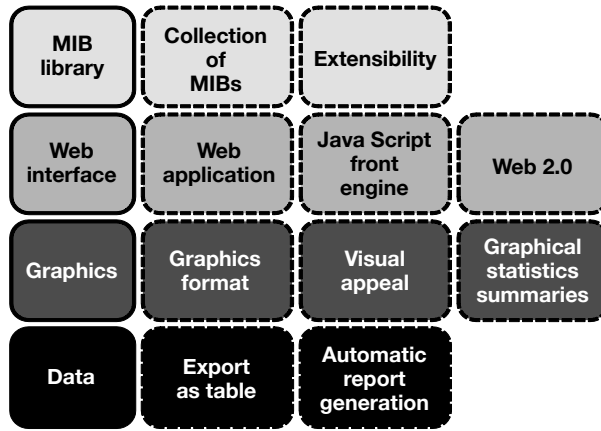


Figure 3.5 Features that should be considered when picking your SNMP performance monitoring tool.

finally, it can be provided as a separate MIB and multiple implementations that you can pick from depending on which operating system you are using. Some tools may provide a number of common MIBs which saves you the time of having to search and install each MIB individually. Extensibility of MIBs, i.e. the ability to add MIBs to your monitoring system is another important sub-feature in this track. One should always remember that MIBs are not only for the remote network device, but should also be made known to the monitoring software.

Otherwise, your monitoring software will not know which metrics to read off the meter at the network device.

Web interface is the second important track. Here, the presence of a web application is an indispensable part. Apart from it, the browser's end of such web application is equally important. When working with the monitoring tool through a browser, all dynamics are written in Javascript, therefore, the quality of the Javascript engine is a major contributor to the quality of experience related to the use of the web application. Another major keyword is *Web 2.0*, which is a step up on the scale of *interactivity* one can have with the remote web application. Sadly, none of monitoring tools studied by the authors employ Web 2.0 design patterns, and this is why users have to wait several seconds on each update of the page with a given performance statistics. With Web 2.0 employed, not only such updates would become seamless, but other elements of interactivity would have a much higher visual appeal.

Graphics track of features covers the properties of plots generated by the monitoring tool automatically. The GD library has already been mentioned above, but the fact that it is shared by most tools does not mean that all tools share the same visualization method in data plots. GD library is simply a drawing API, which means that it is up to each tool as to how to draw *data plots* to make them visually appealing. Naturally, *visual appeal* is another sub-feature in this track. Many tools put additional statistics summaries into data plots alongside with the main data. For example, beside the main line plot of traffic utilization one might want to add another line visualizing dynamics in averages of the same performance metric.

The last track in SNMP monitoring is about how raw or processed data is handled by the tool. For example, when creating reports based on monitoring results, it is often necessary to export data in form of a table to use in documentation, or, optionally, to process further by another software for additional analysis. Some tools offer automatic report generation as an integral service within the web application. This, naturally, makes writing documentation a much easier task.

3.1.6 Specifics of NetFlow

Features in NetFlow-based monitoring tools in Figure 3.6 can also be split into four major feature tracks. However, the nature of traffic monitoring performed by NetFlow-based monitoring versus a broader performance monitoring covered by SNMP creates major differences in these feature tracks.

Arguably, top feature track in NetFlow passive monitoring is the ability to search and filter *traffic flows*. It is difficult to underestimate these features when you discover what your router keeps track of 10^5 flows at any point of time, especially given that 80% of them are 2 – 3 packet flows which are not normally worthy of detailed analysis. Filters are another side of such selective analysis and can be inflicted either when exported by the remote network device or when selected by the monitoring tool for display in a web page. Only the latter is integrated into this track of features, while the former is the part of the next track.

Again, given that there can be 10^6 traffic flows in the current stack of a network device at any point of time, the issue of efficiency is very important for a monitoring tool. Efficiency can be affected in two ways, – both native and publicly developed NetFlow implementations allow for rule-sets to be defined for the device in order to impose limits on which traffic flows to keep track of. Another major boost of efficiency can come from

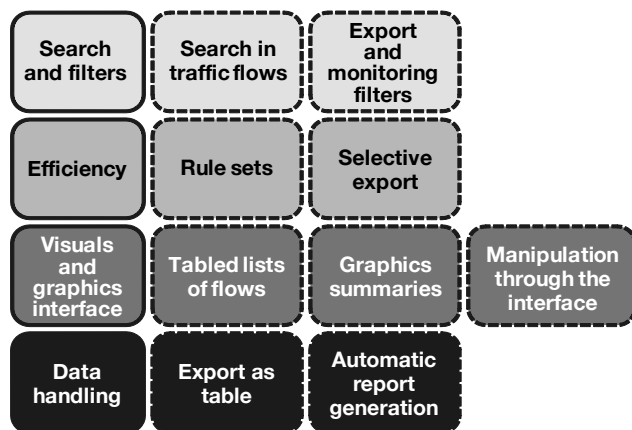


Figure 3.6 Features that should be considered when picking a NetFlow-based monitoring tool.

selective export, where only a limited number of flows for some criteria can be exported while a larger number of flows is physically supported by the device. Naturally, both sub-features can be used in a tandem.

Visualization track of features is otherwise identical to those in SNMP monitoring except for the higher order of flexibility when displaying long tabled lists, – all columns in NetFlow performance tools can be ordered in either increasing or decreasing order. Given that there are many columns in a traditional NetFlow performance tool, it can be a very handy feature at times. This is the integral part of the quality of the web interface offered by the monitoring tool.

Finally, data handling feature track in Figure 3.6 is literally identical to that in Figure 3.5, including identical needs and features offered by popular tools in both SNMP and NetFlow areas of passive performance monitoring.

3.2 SNMP-Based Tools

If you ask an average *network administrator* about SNMP performance monitoring, you are very likely to get the abbreviation *MRTG* in return. MRTG stands for Multi Router Traffic Grapher and is one of the first SNMP *front ends* developed entirely in public domain, i.e. free of charge to download, install and use. This entire section is dedicated to MRTG-like tools.

3.2.1 Basic Features of MRTG

Partially from the name, and definitely from even a short description of the tool you should very easily grasp that the tool is not really an SNMP front-end, but rather a concise version of it created specifically to watch traffic on remote network devices. Traditionally, MRTG only monitors input (I) or output (O) traffic on all interfaces of a device. Therefore, only two particular PDUs for these OIDs are in use by the tool, thus making it quite limited in practical use.

However, it is also true that in most practical *use cases*, all you need to know about your network device is how loaded it is in terms of traffic. All other loads within the device related to this original traffic load, that is your CPU usage, ROM/RAM utilization, etc. are all in direct relation to the *traffic load*. It does, though, depend on your particular network device and there are some cases where not everything in the device relates to traffic.

You can find the tool itself, documentation and whatever the support its developers offer at [13]. Given that the tool is one of the oldest in the market and is definitely amongst the most popular free tools, it is very actively supported to the day.



Figure 3.7 MRTG logo: if you ever encounter this logo in the Internet you should recognize it easily.

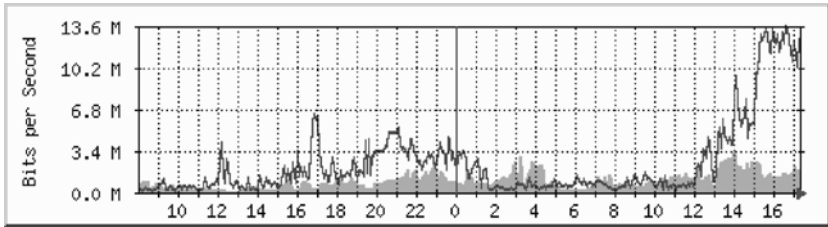


Figure 3.8 Standard look of bits per second traffic statistics generated by MRTG.

MRTG will normally generate a *web page* containing several *graphical plots* containing *traffic statistics* for various time scales, i.e. hourly (as displayed in Figure 3.8), daily, monthly, and so on. Each plot incorporates both the *outgoing traffic* on each *interface*, where incoming traffic is displayed as a green-filled area and outgoing as a dark-blue line, as per Figure 3.8. Vertical axis is, naturally, the *traffic throughput* in bits per second. However trivial it may appear, MRTG does a few things originally:

- for every data snapshot creates not only GIF or PNG data plots, but the entire HTML page, thus making it easier to access through the web interface;
- automatically scales Y axis in such a way that the most detail is shown;
- adds statistical summaries as max, average and current values for both incoming and outgoing traffic.

So, basically, what we have with MRTG is a tool that collects raw SNMP data from remote devices, appends them to a log file containing previous samples, and at the end of each cycle compiles raw data into several graphics plots together with static HTML pages containing these graphics plots. Once this is done, the job of the web application is easy, it only has to grab static HTML pages and serve them back to the browser upon request. This is, in fact, done by a few tools explained later on.

Below is the list of original solutions offered by the initial author of the tool and other developers over the years of this tool's popularity:

1. SNMP protocol implementation is written in Perl, thus, there is no need of linking to any external SNMP libraries;

2. Support for 64-bit counters, thus, removing the problem of having to watch the moment when the counter wraps, – with 32-bit counter–sthis used to be a major problem.
3. Constant size of log files containing raw SNMP data through a special algorithm used to aggregate data.
4. Parts of the tool are written in C in order to boost performance.
5. GIFs are replaced with PNG graphics in MRTG-2.
6. The look of static HTML pages generated by the tool are highly configurable.

However large is the share of graphics and HTML generation done by the MRTG itself, there is still something to be done on the outside of the tool. In fact, there are a few tools that install on top of MRTG in order to help with user interface of the tool. *RRDtool* is the most popular among others.

Now, there is a reason why MRTG was the first to be described in this section. Although the tool claims it supports both Unix and Windows NT platforms, in reality the authors found it difficult to install it on both, especially on Windows. In fact, there is a side distribution of MRTG called *MRTG-XTRA* that is supposed to work on Windows. In practice, again, authors found the installation of MRTG-XTRA just as difficult.

3.2.2 Using PRTG

Instead, this section will be dedicated to a successful installation and further experience from the use of *PRTG* (Paessler Router Traffic Grapher).

PRTG is also a *freeware*, but is a much broader tool than the MRTG itself, although both tools share the original idea about passive network monitoring.

Just like MRTG, PRTG is also a network monitoring and bandwidth usage monitoring tool for Microsoft Windows. Its scope is a bit wider than that of MRTG as PRTG can interface with *NetFlow* as well. Other tools that make *NetFlow interface* their core activity will be considered later in this chapter, while this section concentrates fully on SNMP-based monitoring.

PRTG also has a wider scope of performance metrics it can read off the remote devices, such as *CPU load*, *device temperature*, among others. Raw metrics are stored in a database making it easier to browse processed

performance statistics from the user interface. In fact, the web interface offered by the tool is quite a step-up from what is offered by the tandem of MRTG with RRDtool or similar tandems.

In the terminology of PRTG, users perform performance monitoring by activating *sensors*. Figure 3.9 contains the default list of sensors you get when you first install the tool. Apart from the traditional SNMP traffic sensors there is a number of environmental sensors that are not directly related to traffic.

Installation of PRTG is simple in accordance with common practice among Windows software in general. Besides, this is the case of a stand-

All Sensors			<input type="checkbox"/>
<input type="checkbox"/> Traffic Monitoring Samples			<input type="checkbox"/>
<input type="checkbox"/> Traffic Monitoring via Packet Sniffer			<input type="checkbox"/>
<input type="checkbox"/> Network Sniffing for 10.*.*.* (filtered) (Top Talkers - Top Connections - Top Protocols - Top Mac Addresses)	19 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Outbound Traffic (filtered) (Top Senders)	73 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Inbound Traffic (filtered) (Top Receivers)	55 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Traffic Monitoring via Netflow			<input type="checkbox"/>
<input type="checkbox"/> Network Traffic (filtered) (Top Protocols - Top inbound/outbound Interfaces - Top Talkers - Top Connections)	0 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Outbound Traffic (filtered) (Top Senders)	0 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Inbound Traffic (filtered) (Top Receivers)	0 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Firewall Traffic via SNMP			<input type="checkbox"/>
<input type="checkbox"/> Firewall 1 WAN	1.753 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Firewall 1 LAN	1.504 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Firewall 2 WAN	0 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Firewall 2 LAN	134 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Traffic Monitoring via SNMP			<input type="checkbox"/>
<input type="checkbox"/> Network Card on Marvin	5 kbit/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Environment Monitoring			<input type="checkbox"/>
<input type="checkbox"/> Temperature Rack Bottom	0 Degrees	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Temperature Rack Top	28 Degrees	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Humidity Data Center	0 %	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> LAN Monitoring Samples			<input type="checkbox"/>
<input type="checkbox"/> Server Monitoring: [REDACTED]			<input type="checkbox"/>
<input type="checkbox"/> Used Physical Memory on [REDACTED]	88.768 kbyte	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Number of Processes on [REDACTED]	29 #	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Server Monitoring: [REDACTED]			<input type="checkbox"/>
<input type="checkbox"/> Physical Memory Used on [REDACTED]	348.352 kbyte	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Virtual Memory Used on [REDACTED]	360.448 kbyte	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Number of Processes on [REDACTED]	57 #	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> CPU Privileged Time on [REDACTED]	0 %	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> CPU Processor Time on [REDACTED]	0 %	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> CPU User Time on [REDACTED]	0 %	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Disk Time on [REDACTED]	0 %	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Disk Transfers per Sec on [REDACTED]	1 Transfers/Sec	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Server Monitoring: [REDACTED]			<input type="checkbox"/>
<input type="checkbox"/> logical disk: _Total > disk disk bytes per sec on [REDACTED]	98.598 bytes/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> logical disk: _Total > disk percent disk time on [REDACTED]	1 %	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> memory > memory available kbytes on [REDACTED]	1.458.220 kbyte	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> memory > memory pages per sec on [REDACTED]	0 #/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> physical disk: _Total > disk disk bytes per sec on [REDACTED]	82.128 bytes/second	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> processor: _Total > cpu percent processor time on [REDACTED]	3 %	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> system > system context switches per sec on [REDACTED]	1.134 #/second	<input type="checkbox"/>	<input type="checkbox"/>

Figure 3.9 List of sensors offered by PRTG at the default installation.

alone installation, where SNMP meter, collector, and web application are all installed into a single machine. With Windows installation of PRTG this complexity is transparent and is taken care of by the installer automatically. In the end of the installation process installer loads your default web browser and opens the start page of the web application used for performance monitoring.

PRTG also claims that the tool uses *database logging* internally. If this claim is true, it means that search and creation of complex combinations of performance statistics is a lot easier to generate than when the raw SNMP data is stored in log files. Files are traditionally more difficult to search in both directions of the timeline than lines in a relational database.

3.2.3 PRTG Output

Figure 3.10 contains a simple example when the CPU load sensor was used to collect *performance statistics*. First, it is important that the page was created by a fully fledged web application, the solid proof of which is the URL at the top of the page indicating both that the web application is local to the machine (loopback IP) and that URL contains variables processed by a scripting language at the web application.

Each sensor in PRTG has its own page which dynamically creates contents based on the selected sensor. The name of the sensor is put at the very top of the page. The next line right below the name of the sensor is the path indicating where in the monitoring tree the current sensor is located. CPU load sensor given in the example is in Home/Devices/Local probe/Probe device, which is fairly deep in the tree. It also indicates that other branches in the tree may contain remote probes with many sensors depending on the MIB structure of each given remote device. This particular example is limited in scope to the local sensors in order to simplify explanation of the tool. Besides, if this explanation is followed literally, the immediacy of graphics and textual output as well as ease of use should be a good reward for first time users and provide valuable first time experience without having to build complex systems incorporating multiple devices.

Each plot in PRTG has an extra line in plots keeping trace of *downtime*, i.e. times when PRTG collector failed to read SNMP PDUs off remote (local in this particular example) device. Clearly, downtime is a very rare phenomenon in such systems and almost never occurs in standalone systems.

For each sensor PRTG also offers a table view of performance statistics

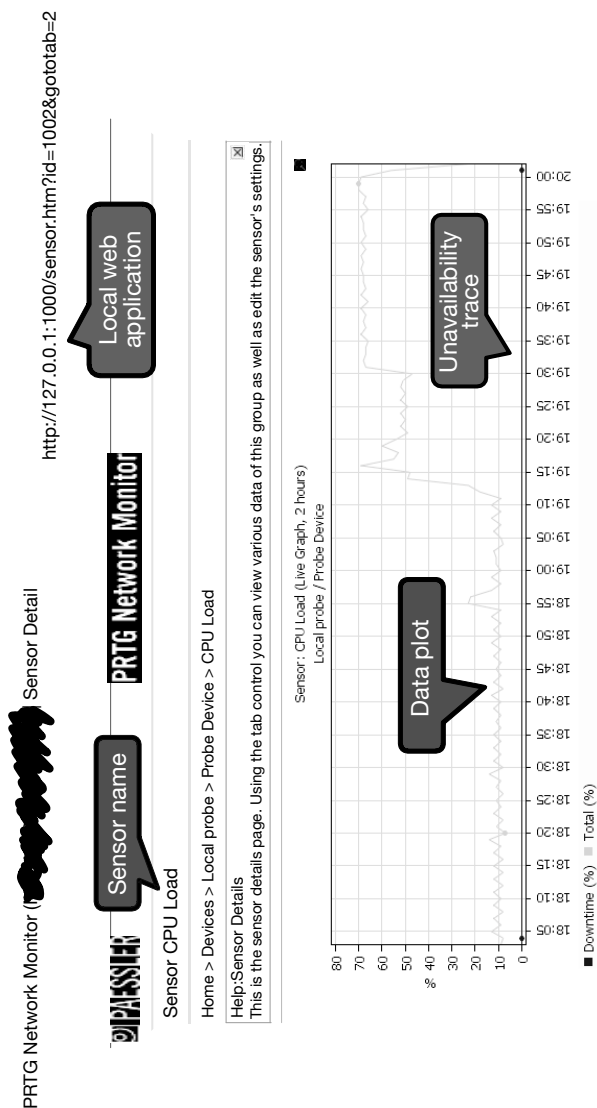


Figure 3.10 Output of CPU load sensor in graphical form, including the HTML header pointing out the fact that the web application is local to the machine.

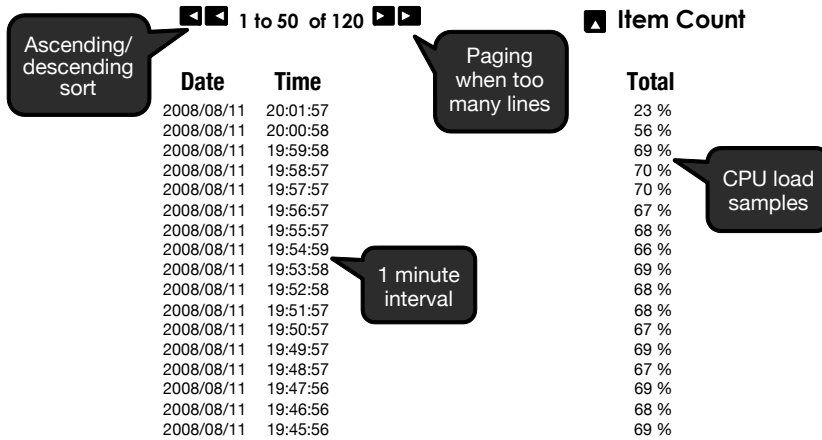


Figure 3.11 Output of CPU load sensor in a rich table form, including paging and sorting capabilities.

below the graphical plot. The tables are sophisticated allowing for ascending and descending sorting by clicking on the column header. Besides, when there are too many lines to display on a single page, PRTG automatically pages the table and places navigation bar at the top of each page. This kind of logic requires sophisticated programming on the part of web application.

Each line in the table view contains a separate column with detailed timestamp with second precision, thus allowing to trace actual samples at given points of time.

At close examination, there was no way found that would export the table as plain text to be used in customized documents. However, a copy / paste of the table on the page would result in a table-like textual data when imported into a spreadsheet or a word document.

3.2.4 Traffic Components of PRTG

A more sophisticated sensor is displayed in Figure 3.12 and Figure 3.13. That sensor is responsible for both incoming and outgoing traffic, thus, being very similar to traditional MRTG. However, PRTG's version of these statistics is richer. For example, PRTG data contains both traffic volume and throughput for both incoming and outgoing traffic, which are not always the same.

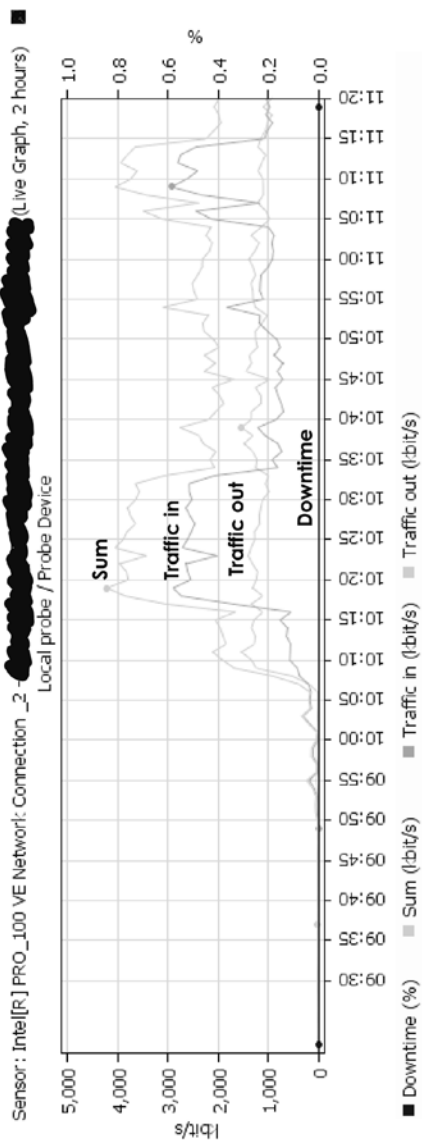


Figure 3.12 Graphical plot from network connection sensor containing incoming traffic, outgoing traffic, and statistical summary.

◀◀ to 50 of 120 ▶▶

▲ Item CountSum

Date	Time	Sum (Volume)	Sum (Speed)	Traffic in (Volume)	Traffic in (Speed)	Traffic out (Volume)	Traffic out (Speed)
2008/08/12	11:19:52	14,433 KByte	1,987 kbit/s	7,599 KByte	1,046 kbit/s	6,834 KByte	941 kbit/s
2008/08/12	11:18:52	15,379 KByte	2,080 kbit/s	7,216 KByte	976 kbit/s	8,163 KByte	1,104 kbit/s
2008/08/12	11:17:52	14,612 KByte	1,988 kbit/s	7,721 KByte	1,051 kbit/s	6,891 KByte	938 kbit/s
2008/08/12	11:16:52	14,231 KByte	1,944 kbit/s	6,766 KByte	924 kbit/s	7,465 KByte	1,020 kbit/s
2008/08/12	11:15:52	15,345 KByte	2,091 kbit/s	7,531 KByte	1,026 kbit/s	7,814 KByte	1,065 kbit/s
2008/08/12	11:14:52	16,400 KByte	2,239 kbit/s	7,923 KByte	1,082 kbit/s	8,477 KByte	1,157 kbit/s
2008/08/12	11:13:52	26,671 KByte	3,641 kbit/s	17,839 KByte	2,436 kbit/s	8,832 KByte	1,206 kbit/s
2008/08/12	11:12:52	27,875 KByte	3,806 kbit/s	20,129 KByte	2,748 kbit/s	7,747 KByte	1,058 kbit/s
2008/08/12	11:11:52	28,803 KByte	3,934 kbit/s	20,479 KByte	2,797 kbit/s	8,324 KByte	1,137 kbit/s
2008/08/12	11:10:52	26,442 KByte	3,610 kbit/s	17,628 KByte	2,407 kbit/s	8,814 KByte	1,203 kbit/s

Figure 3.13 Table data from network connection sensor containing incoming traffic, outgoing traffic, and statistical summary.

In general, the ease of use of PRTG is such that it should be no problem even for first time users to be able to master the tool in under an hour. It can be helpful when the tool is used for testing rather than for the use in running networking systems. In research, in particular, such use is overwhelmingly more popular.

Finally, in spite of the above example PRTG is not limited to a *stand-alone installation*. There are two patterns in which PRTG can be easily employed. First, you can install it on the machine in charge of network administration and make it collect SNMP statistics from *remote network devices*. The second usage is to install it on a self-installed network device and use either a web browser or another installation of PRTG to read SNMP PDUs off the first installation. Both patterns will have to be installed in Windows machines and will work out of the box.

3.3 NetFlow-Based Tools

NetFlow performance monitoring tools are drastically different from SNMP tools. In fact, many of these tools are not specifically based on NetFlow protocol, although all tools use very similar definitions of traffic flows to those used by NetFlow.

To be even more exact, NetFlow protocol is not an official standard as it never passed the stage of an RFC document. Contemporary version of NetFlow, on the other hand, has become a full fledged standard and

changed its name to IPFIX. Whenever you encounter the abbreviation IPFIX, be sure that you are dealing with the technology that contains NetFlow in its core. This section gives a few advices on that.

3.3.1 NetFlow Compliance

Actually, a tool does not have to comply with the NetFlow protocol. All your tool has to do is export information about *traffic flows*. It is the details that you get along with each exported flow information that makes the main difference among *NetFlow-like* tools.

Still, there is a valid reason why all these tools are incorporated under the term “NetFlow”. First of all, NetFlow was originally developed by Cisco and therefore comes pre-installed with most *Cisco* devices that deal with traffic.

Secondly, NetFlow is about the network device itself, not the location where traffic flows are processed and analyzed. Of course, Cisco also provides proprietary collectors that are capable of receiving NetFlow export. But there are many more third-party tools that looked up the description of NetFlow in the RFC and created their own interface capable of receiving the exported data. This is the reason why the majority of flow monitoring tools are written as *NetFlow clients*. A simple web search will prove this.

Thirdly, there is yet another class of NetFlow monitoring tools one of which will be described in detail in this section, – a tool creates its own flow aggregation engine but is capable of exporting flow data in accordance with NetFlow protocol. Clearly, this case is similar to that previously explained for SNMP, – some network devices may be put together by hand instead of buying a factory-made device. In this case, a tool that can mimic NetFlow is indispensable when traffic monitoring is to be performed on the device.

3.3.2 top: the UNIX Tool

Without further ado, this section will introduce the tool called *ntop* [14]. Its logo is in Figure 3.14 and is quite simple, as anything developed as a necessity in someone’s research.

It is easy to understand what *ntop* does if you have prior experience maintaining a Linux or a Unix-like operating system. Most of those have a very useful command line utility called *top*. When run from command



Figure 3.14 The logo of ntop.

line is grabs the entire monitor and turns it into a realtime monitoring utility for all processes running at the time in your operating system.

Example of Linux top in action can be found in Figure 3.15. It is important to understand later on how ntop functions. First of all, top will always give you lots of information about overall system state split into three groups of parameters, – tasks, CPU, and memory. As you can see from the figure swap is the fourth group but it is rarely used for conventional monitoring and only makes sense when the system is substantially loaded, which the example system is not.

Based solely on the overall statistics you can see how much CPU all your processes put together use at the time, how much RAM is in use and how many tasks there are to share all the resources your system has to offer. To artificially create a little bit of load in the example system, the top snapshot was taken while compiling this very book using the L^AT_EX system for typesetting and command line PHP for scripting.

If you use top on a Linux system, however, you should normally get a lot more processes in your list. As one can deduct from the example output in Figure 3.15, the OS used in the example is Cygwin shell running on top of Windows. Since this is an artificial overlay shell, there are not many processes when the shell is idle regardless of the fact that there are many processes running in Windows continuously. You can see those through Window task manager but they are not perceived by Cygwin shell and are not registered by top.

By default, top runs continuously and does not allow you to use command line while it is running. It also regularly updates its statistics and then the screen will blink for an instant while the output is being refreshed by the utility. At every refresh there is a good chance that some of your processes may disappear, new processes may be added to the list, and, more often, processes will change places in the list based on their personal consumption of resources. From the example in Figure 3.15 you can see that the top process in the list is the latex process which comes from the

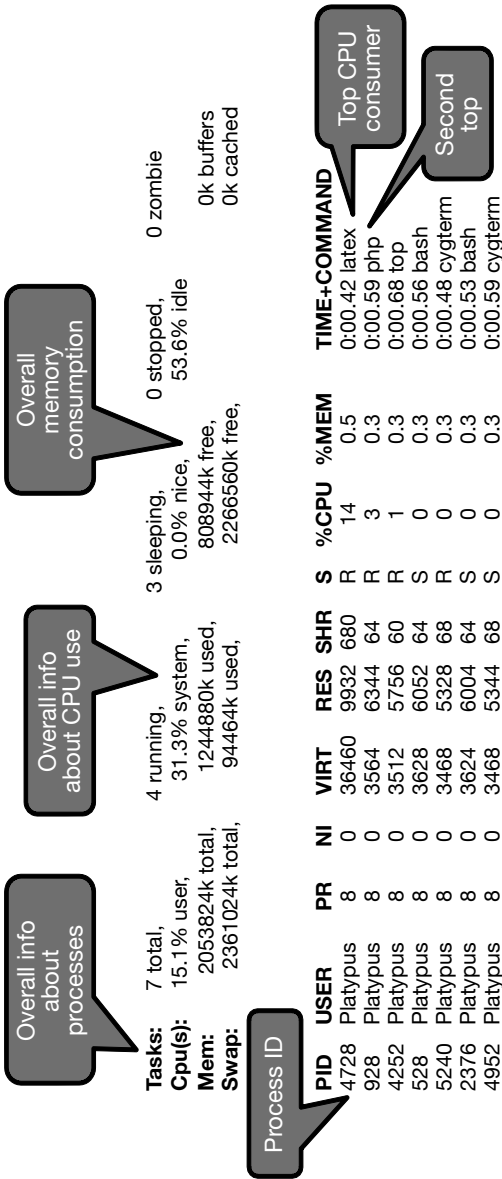


Figure 3.15 Example of the output from real top utility in Linux.

TeX distribution of L^AT_EX system for Windows and is in charge of compiling your .tex source files. This is quite a heavy process which is why it consumes more CPU than any other process in Cygwin shell at the time.

3.3.3 ntop: the Network top Tool

Now, back to ntop. The reason this tool was picked as an example is in part the same reason PRTG was picked to represent SNMP monitoring tools, i.e. this tool has a Windows port and allows for standalone operation.

There is an additional reason for picking ntop, – while there are many tools in SNMP monitoring world, ntop is about all you can get in Net-Flow world in public domain. Besides, the tool has celebrated its 10th anniversary this year and over the years has collected an impressive list of functional additions if to mention numerous fixes coming from both the original author of the tool and its active users.

Like most previously introduced tools, ntop is also the product of academic research into effectiveness of traffic flow monitoring and analysis. Scientific details are out of scope of this book but can be found in two papers published on ntop in [25] and [26]. Those papers are popular enough for you to have a good chance with Google Scholar to give you access to the full text for free.

Without going into too much scientific detail, ntop's internal architecture can be simplified to the design in Figure 3.16. Packets continuously flow through or terminate at your network device which is why you need to use a system level utility to capture those packets for later analysis. This part does not belong to ntop. Instead, authors of the tool used the very famous *libpcap* library that is so popular among *packet analysis* folk that it has a very good Windows port and is used by many tools created specifically or ported to Windows. In fact libpcap is the part of the architecture which is strained the most during operation.

The analysis block of ntop is also unique and offers a number of innovations targeting substantial increases in performance while creating flows from raw packets, performing lookups in temporary flow table when updating, etc. In fact, ntop does claim to have a few interesting solutions within this block. Scientific papers in [25] and [26] should offer enough scientific insight into this issue.

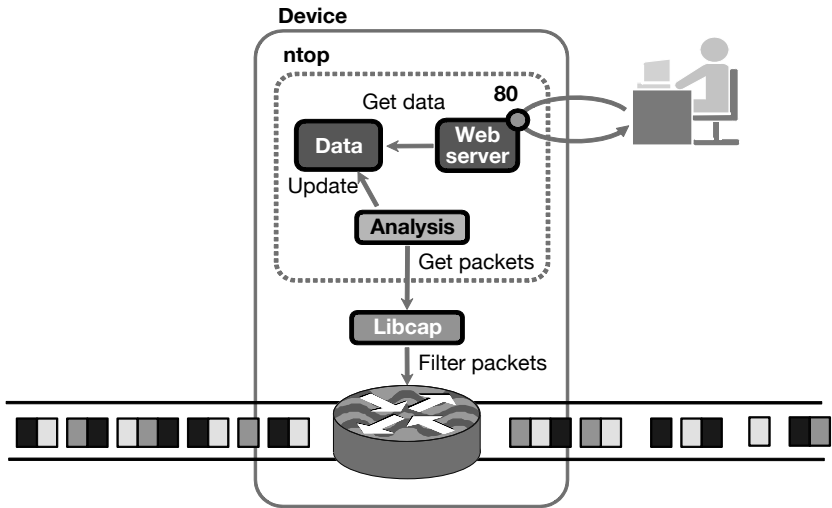


Figure 3.16 Trivial view of ntop architecture. The entire tool is built on top of libpcap library that was originally created for Linux but has a very stable Windows port.

3.3.4 User Interface to ntop

Web application part of the tool is incorporated into ntop. At long as the tool is running you can access current traffic statistics through *web interface*. As was mentioned earlier, this part is supposed to support the name of the tool by being similar in output to the traditional top utility on Linux.

Figure 3.17 displays standard web layout offered by ntop. First of all, the line at the very top will tell you that you are contacting a local web server on the port 3000. In previous figures the number 80 was used as the web server's listening port, but that was only a general example, since standard web listening port is 80. Locally, however, you can pick any port you want as long as you specify this port in URL (as was done in Figure 3.17) and you are certain that your web server is listening on this port. When you install ntop it points your browser to its default page so finding the right URL should not be a problem. You can always find it among the links created by ntop in windows start/programs menu.

The menu below the logo of ntop at the top of the page although looks fairly barren contains all functionality offered by the tool. If you browse

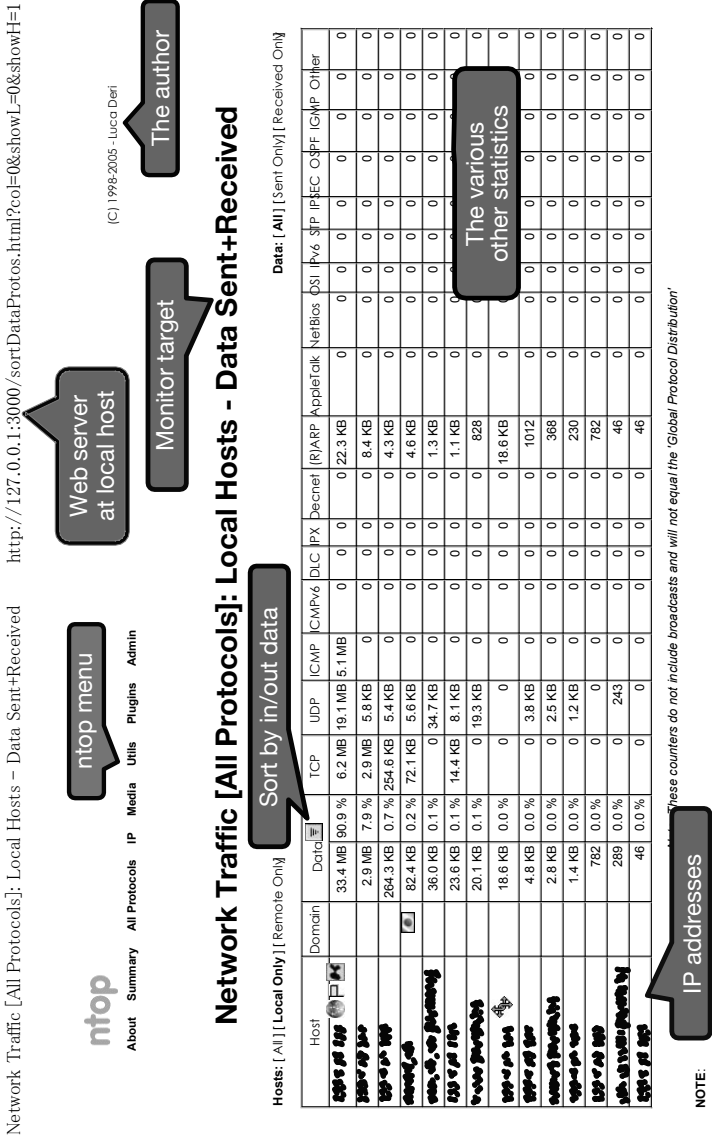


Figure 3.17 Standard display layout of ntop web interface.

around that menu you should realize that it contains a lot of useful functionality even beyond NetFlow.

Finally, the core of each web layout is the centre of the page containing traffic flows. The title of the page should normally tell you what kind of data you are viewing. Given that ntop offers various combinations of raw traffic *flow statistics*, the title is important. Below the title is a standard table containing all flows satisfying the *filtering* conditions you picked for the page. All column names, as it should be, are active and can be used to sort the table in both directions.

Figure 3.18 is the close-up of the table from Figure 3.17 and should allow better view of the data. Like traditional top, ntop always displays top consumers of a given *resource*. When the page first loads the resource is a default choice of ntop itself but can be changed using active links posing as column names. As a coincidence, numeric order in another column may concur with the current selection. This, however, is rare because it would otherwise mean both columns contain perfectly correlated statistics thus making one of the columns redundant.

3.3.5 Statistical Traffic Summaries

Raw statistics are rarely useful to network administrators because long lists of statistics rarely make sense in unprocessed form. To provide some insight into the data you would have to download the table, perform some statistical calculations and only then present the data. To some extent, ntop will do that for you in its summaries. Here, the number of raw data is limited at the benefit of being able to monitor some statistical properties for a few important parameters. The example in Figure 3.19 contains summary of traffic throughput divided into bits per second and packets per second halves. The peak and average values of each of these traffic flow parameters are extremely important for monitoring. As can be found in the example, although the current and peak pps (packet per second) throughput values are high, the average is very low suggesting high volatility.

After the two above examples a good question can be duly raised about relation of ntop to traditional *NetFlow*. This question is especially legitimate when you realize that Figure 3.18 and Figure 3.19 both contains only one IP address per line while traffic flow by definition can be uniquely identified only by the 5-tuple description containing source and destination IP addresses and port number and often the name of the protocol. It is true that ntop is not a traditional NetFlow tool if the above criteria

Host	Domain	Data	TCP	UDP	ICMP	ICMPv6	DLC	IPX	Decnet	(R)ARP	AppleTalk	NetBios	OSI	IPv6	STP	IPSEC	OSPf	IGMP	Other
132.242.242.242		33.4 MB	90.9 %	6.2 MB	19.1 MB	5.1 MB	0	0	0	0	22.3 KB	0	0	0	0	0	0	0	0
132.242.242.242		2.9 MB	7.9 %	2.9 MB	5.8 KB	0	0	0	0	8.4 KB	0	0	0	0	0	0	0	0	0
132.242.242.242		264.3 KB	0.7 %	254.6 KB	5.4 KB	0	0	0	0	4.3 KB	0	0	0	0	0	0	0	0	0
132.242.242.242		92.4 KB	0.2 %	72.1 KB	5.6 KB	0	0	0	0	4.6 KB	0	0	0	0	0	0	0	0	0
132.242.242.242		36.0 KB	0.1 %	34.7 KB	0	0	0	0	0	1.3 KB	0	0	0	0	0	0	0	0	0
132.242.242.242		23.6 KB	0.1 %	14.4 KB	8.1 KB	0	0	0	0	1.1 KB	0	0	0	0	0	0	0	0	0
132.242.242.242		20.1 KB	0.1 %	19.3 KB	0	0	0	0	0	828	0	0	0	0	0	0	0	0	0
132.242.242.242		18.6 KB	0.0 %	0	0	0	0	0	0	18.6 KB	0	0	0	0	0	0	0	0	0
132.242.242.242		4.8 KB	0.0 %	3.8 KB	0	0	0	0	0	1012	0	0	0	0	0	0	0	0	0
132.242.242.242		2.8 KB	0.0 %	2.5 KB	0	0	0	0	0	368	0	0	0	0	0	0	0	0	0
132.242.242.242		1.4 KB	0.0 %	1.2 KB	0	0	0	0	0	230	0	0	0	0	0	0	0	0	0
132.242.242.242		782	0.0 %	0	0	0	0	0	0	782	0	0	0	0	0	0	0	0	0
132.242.242.242		289	0.0 %	243	0	0	0	0	0	46	0	0	0	0	0	0	0	0	0
132.242.242.242		46	0.0 %	0	0	0	0	0	0	46	0	0	0	0	0	0	0	0	0

Figure 3.18 The list of ntop traffic flows listed in decreasing order of sent + received traffic.







Host	Domain	Data			Packets		
		Current	Avg 	Peak 	Current	Avg	Peak
132.5.57.156 		162.9 Kbps	30.5 Kbps	162.9 Kbps	242.5 Pkts/sec	38.9 Pkts/sec	242.5 Pkts/sec
132.5.57.154		73.7 Kbps	3.1 Kbps	73.7 Kbps	140.9 Pkts/sec	5.7 Pkts/sec	140.9 Pkts/sec
132.5.57.150		398.5 bps	273.0 bps	5.7 Kbps	0.6 Pkts/sec	0.2 Pkts/sec	2.1 Pkts/sec
100.0.1.50 		0.0 bps	71.2 bps	787.1 bps	0.0 Pkts/sec	0.1 Pkts/sec	0.7 Pkts/sec
100.0.1.50 		175.6 bps	31.9 bps	175.6 bps	0.2 Pkts/sec	0.0 Pkts/sec	0.2 Pkts/sec
132.5.57.174		0.0 bps	20.4 bps	426.4 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.4 Pkts/sec
132.5.57.174		21.8 bps	17.5 bps	164.0 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.2 Pkts/sec
132.5.57.151 		26.9 bps	16.3 bps	266.2 bps	0.1 Pkts/sec	0.1 Pkts/sec	0.8 Pkts/sec
132.5.57.152		0.0 bps	4.1 bps	55.2 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.1 Pkts/sec
132.5.57.152		1.2 Kbps	2.9 bps	1.2 Kbps	1.1 Pkts/sec	0.0 Pkts/sec	1.1 Pkts/sec
132.5.57.152		0.0 bps	2.9 bps	73.7 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.1 Pkts/sec
132.5.57.152		0.0 bps	1.5 bps	38.5 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.1 Pkts/sec
132.5.57.157		0.0 bps	0.7 bps	6.1 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.0 Pkts/sec
132.5.57.153		0.0 bps	0.0 bps	6.0 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.0 Pkts/sec

Figure 3.19 Summary statistics of throughput offered by ntop in a ready-made processed condition.

are applied to it. Instead, ntop pursues targets that are most common in practical traffic monitoring where *traditional traffic flow* is not always the minimal unit of data.

3.3.6 NetFlow Attributes of ntop

It does not mean that ntop should be discarded from the list of NetFlow monitoring tools. In the menu at the top of the page you can always find the traditional list of current flows as per Figure 3.20. Terminology used by ntop may be slightly different from traditional traffic flow analysis, but all the components are in place. For example, what ntop addresses as client and server are source and destination in traditional traffic flow analysis. This is, however, only a semantic discrepancy and given that the meanings of the words are very much alike you should never face difficulties trying to understand what the tool means. Again, in the world of practical performance monitoring the terminology used by ntop may even make more sense depending on the situation.

To purge all remaining doubts about ntop's compatibility with NetFlow protocol, ntop has the ability both to collect and to pose as NetFlow meter, depending on the setup. Similarly to the case of SNMP performance monitoring, you may want to install ntop on a Linux machine that you want to make into a small-scale router in your company. In this case you may now have full GUI access to this "router" and will have to access its traffic statistics remotely. The solution will contain two installations of ntop, one at the "router" machine and the other at the machine you will be collecting and processing statistics at. The former installation of ntop will be setup to run as a *NetFlow meter* while the latter will collect and can even be installed on a Windows machine.

NetFlow is not the only interface ntop offers, – *sFlow* is another tool slash semi-standard used in traffic monitoring that it is important enough for the authors of ntop to develop an interface for it. If you delve further into ntop menu you will find many such extras that will be very helpful to solve practical problems in traffic monitoring.

Active TCP/UDP sessions

Client	Server	Data Sent	Data Rcvd	Active Since	Last Seen	Duration	Inactive	Latency	Note
157.140.1.100:80	157.140.1.100:80	645	90	08/12/08 16:28:55	08/12/08 16:29:18	23 sec	1:41		
157.140.1.100:80	157.140.1.100:80	435	0	08/12/08 16:28:18	08/12/08 16:28:18	0 sec	2:41		
157.140.1.100:80	157.140.1.100:80	50	66	08/12/08 16:28:27	08/12/08 16:28:27	0 sec	2:32		
157.140.1.100:80	157.140.1.100:80	50	66	08/12/08 16:24:11	08/12/08 16:24:11	0 sec	6:48		
157.140.1.100:80	157.140.1.100:80	291	47	08/12/08 16:23:51	08/12/08 16:23:59	8 sec	7:00		
157.140.1.100:80	157.140.1.100:80	63	147	08/12/08 16:28:20	08/12/08 16:28:20	0 sec	2:39		
157.140.1.100:80	157.140.1.100:80	299	0	08/12/08 16:30:48	08/12/08 16:30:48	0 sec	11 sec		
157.140.1.100:80	157.140.1.100:80	66	66	08/12/08 16:25:20	08/12/08 16:25:20	0 sec	5:39		
157.140.1.100:80	157.140.1.100:80	378	882	08/12/08 16:06:39	08/12/08 16:25:54	19:15	5:05		
157.140.1.100:80	157.140.1.100:80	66	66	08/12/08 16:27:10	08/12/08 16:27:10	0 sec	3:49		
157.140.1.100:80	157.140.1.100:80	435	0	08/12/08 16:27:58	08/12/08 16:27:58	0 sec	3:01		
157.140.1.100:80	157.140.1.100:80	50	66	08/12/08 16:23:49	08/12/08 16:23:49	0 sec	7:10		
157.140.1.100:80	157.140.1.100:80	829	1.2 KB	08/12/08 16:11:06	08/12/08 16:26:53	15:47	4:06		
157.140.1.100:80	157.140.1.100:80	468	0	08/12/08 16:30:34	08/12/08 16:30:34	0 sec	25 sec		
157.140.1.100:80	157.140.1.100:80	435	0	08/12/08 16:30:27	08/12/08 16:30:27	0 sec	32 sec		
157.140.1.100:80	157.140.1.100:80	2.8 KB	74.5 KB	08/12/08 16:30:23	08/12/08 16:30:59	36 sec	0 sec	355.1 ms	
157.140.1.100:80	157.140.1.100:80	1.4 KB	1.0 KB	08/12/08 15:41:23	08/12/08 16:26:10	44:47	4:49		
157.140.1.100:80	157.140.1.100:80	168	48	08/12/08 16:26:16	08/12/08 16:26:33	17 sec	4:26	127.8 ms	

Figure 3.20 Traditional NetFlow-like view of traffic in ntop.

3.4 Contemporary Monitoring Realities

Although this chapter was rigidly divided into SNMP versus NetFlow monitoring tools, in reality of passive network monitoring tools have to be more flexible than that. In fact, at the top of user's desires is a single tool that can cover both SNMP and NetFlow capabilities. The reason for this is obvious, – while professionals and researchers in the area realize that SNMP and NetFlow are very different technologies, users in the meantime deal with practical requirements which cannot be fully satisfied by the either of these technologies. Worse yet, as will be proven by the end of this section, even both these technologies put together cannot solve many problems that persist in network performance monitoring today. Because of these traditionally unsolvable problems active measurement first emerged as a new technology which in turn inspired the writing of this very book.

The proof to the statement that a single traditionally SNMP or traditionally NetFlow tool is not enough to fulfill of passive monitoring goals can be found in tools themselves. In particular, ntop is mostly a NetFlow tool but at the same time it incorporates SNMP functionality in parts that augment the core NetFlow functionality. Similar statement can be made about SNMP tools that in many cases require additional NetFlow functionality to attain a monitoring target.

This section picks a realistic network management target in an attempt to analyze ability by both SNMP and NetFlow tools to attain it.

3.4.1 Example Practical Task

This section will revolve around a hypothetical monitoring scenario in Figure 3.21. This scenario is very plausible in practical network monitoring.

Actually, part of the scenario borders on *network management* as the scenario incorporates actions to be undertaken at the network device when a certain condition is met. Overall scenario goes like this.

At the beginning of the loop the user starts listening on traffic utilization at the network device. A threshold of 80% is used which should give us enough time to react and alleviate the forming congestion. When utilization goes over the threshold the user is to get the list of top 50 flows from the device and make a decision as to which of those flows to constrain in terms of their maximum data rate. This will take some two-way communication between the user and the device as per Figure 3.21.

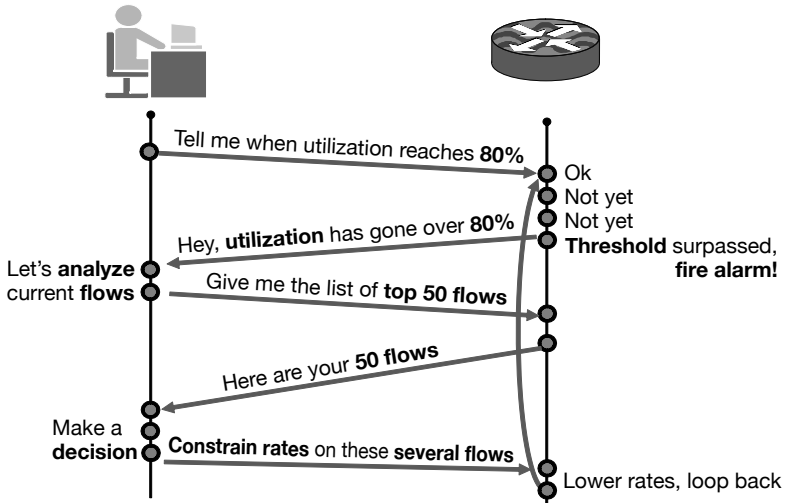


Figure 3.21 Sequence diagram of a full cycle of a practical traffic monitoring activity.

Finally, once the selected flows are constrained in their *data rate*, decrease in overall utilization should ensue thus solving the problem of *congestion*. Therefore, at this point the algorithm can go back to the beginning of the loop.

It is possible that a one-time action may not be sufficient to fully alleviate the congestion. In this case, the device will fire alarm almost immediately after re-entering the loop. This, however, is a positive feature of the algorithm as on each successive alarm the list of top flows (in order of decreasing data rate) will be different which means that the algorithm is fully recursive and will certainly converge to a state where utilization will be less than 80%. In extreme cases, the device may host millions of flows each allowed to send at a very slow rate.

On the physical level, constrained data rates may cause packet loss in flows. In UDP flows this loss cannot be recovered, but in TCP flows this will trigger congestion control algorithm which will retransmit the lost packets. In present networking reality, the possibility of data loss while traffic is routed through the network is always considered a possibility by end applications. This is why ends of a network path are normally much smarter than the routing core and any data loss can be solved if the application is properly designed in the first place. This is to state

that the traffic control solution in Figure 3.21 does not bring any news to traffic as far as end applications are concerned.

3.4.2 SNMP Mapping of the Example Scenario

Now, let us try to map this scenario on either *SNMP* or *NetFlow* at a time. First, Figure 3.22 contains the implementation of the above scenario using only *SNMP* technology. To have access to headers of packets processed by the network device, the famous *SNMP RMON MIB* can be used. *RMON* remembers headers of packets that satisfy a ruleset specified prior to running *SNMP meter*, which allows some room for efficiency through selective dump.

Still, raw headers can be fairly heavy and take long time to download. In fact, if you read *RMON specifications* in [47] you will find clauses where raw packet header dump is not recommended without stringent rulesets limiting the number of flows packets of which should be looked at. This part of *RMON* can be a great performance burden.

As per Figure 3.22, we did succeed at achieving the target but the time it took to download and analyze packet headers delayed the process past the point where a full congestion occurred at network device. This is

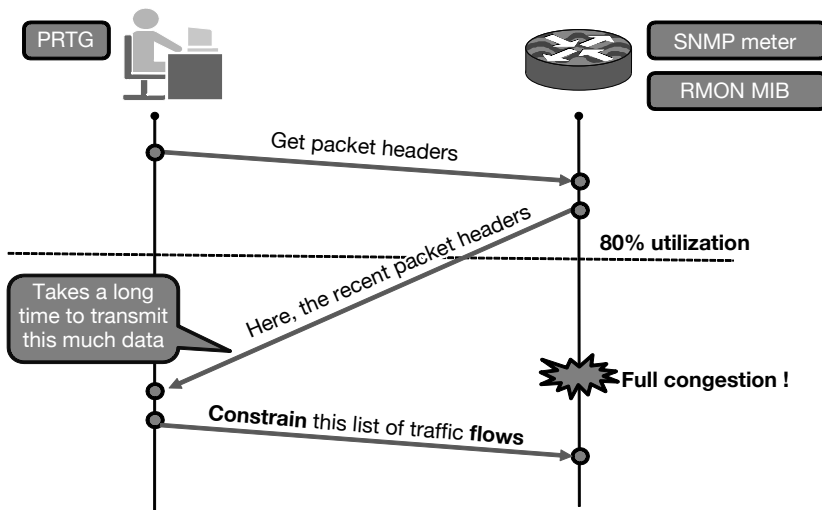


Figure 3.22 Attempt to implement the scenario solely by SNMP means.

a very unpleasant phenomenon when it happens in networks. It is one thing when so called “*heavy hitters*”, – flows with very high data rates and large bulk of data to transmit are constrained at the router, and is completely another when all flows flowing through the network device stall simultaneously thus affecting the entire traffic. This, in fact, is the reason why the threshold of 80% of nominal bandwidth is used as the threshold, – we should have enough time to alleviate congestion before it fully unfolds.

However, since we still were able to constrain heavy hitters, let us consider the algorithm in Figure 3.22 a partial success. Having constrained some flows, it is very probable that when the network device recovers from congestion, its traffic will be substantially lighter than before the congestion.

3.4.3 NetFlow Mapping of the Example Scenario

The NetFlow-only implementation of the scenario ends in even more trivial situation in Figure 3.23. Intrinsically, NetFlow is an export technology where each *network device* takes its own time analyzing the traffic and sends processed summaries about flows at regular intervals or on occurrence of a certain phenomena depending on which setting you use for your NetFlow. This way you do save lots of traffic since each flow may consist of many packets while the flow summary exported by the NetFlow is transmitted only once per export session. Compared with RMON solution above, this is a very efficient way to learn about traffic.

Still, however efficient its traffic monitoring capabilities are, NetFlow is unable to attain the goal of the monitoring scenario in question. Even when it learns that utilization has reached the 80% threshold there is nothing *ntop* on the user’s side can do about it. NetFlow by default is not an *active monitoring* solution, it simply analyzes traffic and exports what it learned to a remote machine for analysis.

Therefore, in the end, just like it happened with SNMP implementation above, the full congestion is bound to occur. Only this time, there is no way for post-congestion control either, – when traffic flowing through the device recovers it is very probable that the next congestion will occur almost immediately after the recovery, thus creating an unbreakable vicious circle.

There is no need for another figure to make it clear that only a hybrid of SNMP and NetFlow implementation will achieve the goal of the target scenario. Since NetFlow is all about efficiency of traffic monitoring and

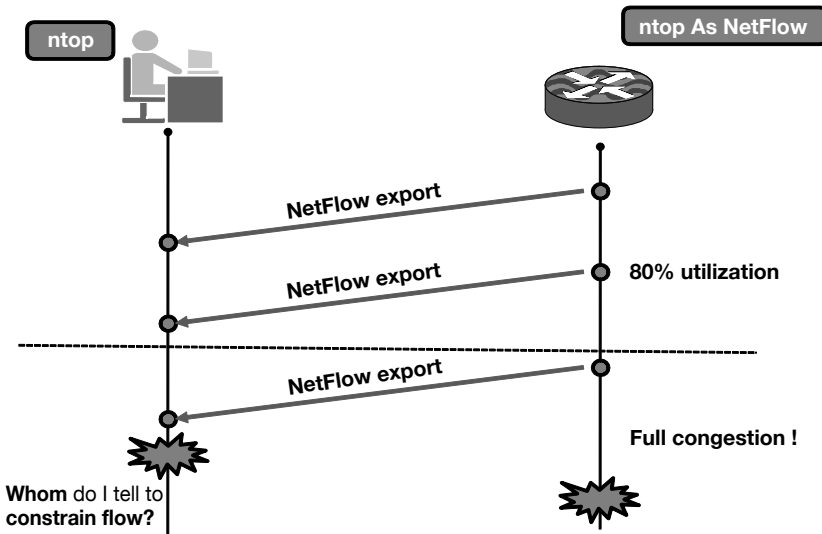


Figure 3.23 Attempt to implement the scenario resorting only to NetFlow.

real-time analysis, the valid implementation should use it as the source of knowledge about traffic based on which a decision is to be made as to which flows to constrain. Prior to that, it would be nice if SNMP could trigger an alarm once overall traffic utilization goes over 80% so that we know exactly when to make the traffic control decision. Finally, when the decision is made, again, SNMP is used to alter dynamic rulesets, restart modules, etc., in order to complete the last part of the target scenario.

3.4.4 Discussion of the Example Scenario

The above is an ample description of a monitoring procedure that already exists and is fairly easy to put together using publicly available tools and components.

However difficult it may have appeared, the above scenario is a simple case given the practical tasks faced by administrators today.

First of all, only a single network device was used in the scenario where administrators have to manage dozens, hundreds, or more routers at the same time. So, *multi-point data collection and analysis* is an important issue. In fact, there are two sides to multi-point data collection.

On one hand, network administrators have to collect performance data from multiple points in the network in order to learn of the overall state of the network. It is very unlikely that an administrator is hired to take care of only one network device. The larger the number of devices administered simultaneously, the larger is the bulk of data and the higher is the complexity of the system itself that collects data from multiple remote meters.

On the other hand, administrators need to collect multi-point performance data to be able to catch various distributed *traffic phenomena*. Network attacks have recently become distributed which means that it is impossible to detect them from a single location. Besides, as the Internet is reaching deeper into everyday life of society, many traditional *social phenomena* become contagious and jumped on to the Internet.

One very vivid example can be given by *Flash Crowd*, a traffic phenomenon that happens when something popular happens in one place in the Internet. Just like crowds gather for major sports events or music concerts, crowds of web users can congest a web server if precautions are not properly made prior to the event. This is, in fact, an independent area of research in traffic analysis.

When raw performance data is collected from multiple points simultaneously, it quickly becomes impossible to process in real time. Precisely because of this most large-scale monitoring endeavors have to do with *offline analysis*. Literally, this means that even if a congestion is about to occur in your network in 10 minutes, you are to learn about it only tomorrow when the bulk of your data is to be analyzed using superior computing power at a special location.

Of course, most dreadful network phenomena are made catchable in real time using SNMP *asynchronous alarms*. Those are not included into the process of raw performance data collection but instead are treated as special cases. This allows for quick solutions to important problems. However, since the answer to why a certain condition occurred in the first place is usually hidden in detailed performance statistics which are not available at the time of quick solutions, making the solution blind and usually blunt.

The above problems have been the main reason why active measurement emerged and is quickly becoming a rival to traditional passive monitoring methods in network management. Many solutions offered by active measurement will be considered further in this book. Some of them will be duly compared with their passive alternatives.

3.4.5 Technology Support of Existing Monitoring Targets

To conclude this chapter, Figure 3.24 lists major performance monitoring objectives coming from network management realities nowadays. There are three major groups, – *performance state* and the ability to perceive it, *performance dynamics* and the ability to follow them, and *online analysis* and the ability to detect occurrence of performance phenomena. Those objectives which can be attained only by applying *active measurement* approach to monitoring are marked with a large black dot. As it turns out there are more of dotted objectives than those that can be traditionally resolved by passive monitoring. It should be noted that most recently emerged performance *monitoring objectives* cannot be supported by traditional passive monitoring any longer. Nowadays, administrators demand highly *dynamic monitoring* tools even though it may come with some loss of precision. This is how active measurement tools are gradually becoming accepted by network management community.

Very special attention should be paid to *billing*. Billing is a part of network monitoring that is in charge of keeping track of how much traffic exactly was transmitted on a given flow or between groups of IP addresses. It is obvious that billing cannot be done using active measurement since the latter has no access to packets and therefore cannot count the traffic. Billing methods are always solidly passive. They are also simple, however, since all you have to do is count how many packets travel on a flow or a set of flows.

Billing is growing old, however. Although it is still used in many

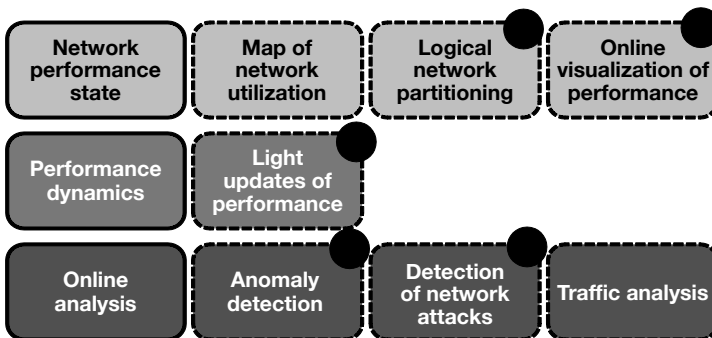


Figure 3.24 Diagram of groups of monitoring objectives. Blocks marked with a black dot can be supported only by applying active measurement methods.

places in the global network, it is losing its ground to two network developments both of which are rooted in the ever growing capacity of the global network.

First of all, if your capacity grows too much, it will be hard for you to catch up with the speed to count all the packets you need. Practical studies and tests performed by the authors suggest that blunt counting of packets works up to 1Gbps even with up-to-date monitoring equipment and good processing power.

Past 1Gbps there is still NetFlow-based billing but even NetFlow has a physical limit at about 10Gbps as indicated by many studies. Given that nowadays the core of some networks has expanded beyond 40-50Gbps, even NetFlow will not catch up with all the packets in such network.

Of course, there are several solutions to *broadband billing*, such as *sampled counting*, for example, but the overall trend in the network is to apply *flat rates* based on time.

There is a rule of thumb in network engineering today, – if you face congestion in your network, increase your capacity tenfold and that should solve all your problems. In many advanced networking projects this is exactly what has been happening over several recent years. Whether this is a good way of dealing with congestion in global networks is for you to witness a few years from now.

Chapter 4



Active Measurement Technology

Until this point the book was running close to a detailed overview of the current state of network performance monitoring technology. Since passive is the old while active is the new management approach, such a long introduction is a prerequisite for grasping the presently occurring technological shift.

From this point on, when a given active measurement technology is being explained it should be easy to compare its targets and achievements to counterparts from the world of passive monitoring. The two worlds almost never clash but very many management objectives can be achieved in both worlds while some others can only be accessible through an active approach.

This chapter is a gentle introduction into the world of active measurement as an integral part of network performance management. The introduction will start at a very low level but almost from the very start it will try to draw a solid line between the passive and active approaches. This line lies in the major difference between the two sets of network parameters, – one is natural as long as you use passive monitoring approaches considered earlier in this book while the other is also a natural output from active inference of network performance.

Naturally, the difference in the two sets of parameters ensues that the methods that produce them should also be different. This chapter will cover such differences as well having once established the difference in the performance parameters themselves.

4.1 Active Measurement Basics

Traditionally, monitoring of network performance has been performed in a passive fashion, which involved *polling* various metrics directly from *network equipment*. One of the most popular tools that performed this task was first defined in *RFC1157* and contained detailed descriptions on the Simple Network Management Protocol (SNMP). Because of its simplicity, SNMP was quickly adopted for use with various kinds of network equipment. It is still widely used to monitor network performance nowadays.

SNMP operation is based mostly on low-level *counters* defined in specific *MIBs* (Management Information Bases), each storing a specific *metric* on each *interface*. Due to this intrinsic quality, a reading party has to read each counter two times and then calculate the value of a metric, taking into account the difference in the counter readings and time interval between readings. As counters are based on solidly defined MIBs, inference of higher level metrics is a complex task.

In plain words, performance information in traditional network management is collected from a large number of network equipment, such as routers, switches, or even hubs.

Naturally, due to relatively large number of monitored network entities in an average network, *online processing* of such bulk data becomes impossible. Even in *offline mode*, it is often impossible to infer network performance based on data from multiple network entities. Instead, traditional network performance monitoring is limited to detection of faults, i.e. conditions with extreme anomalies in operating parameters of network equipment. This area of monitoring is referred to as *Fault Management*, and defines methods of *detection* and quick *recovery* from network faults. In this process, the role of traffic metrics is insignificant.

4.1.1 Measurement in Management

This section will explain why active measurement was founded and now is gaining popularity in the world of network management.

First, traditional network management is represented as part of the illustration in Figure 4.1, where data from several routers are collected and processed by a single location in the network. This location is traditionally referred to as *Network Operations Centre*, abbreviated as *NOC*.

So, before one can learn of the state of the entire network one has to collect information from all routers in question. This literally translates into a

series of requests sent by NOC and directed at each router within the area in question. Network routers respond with whatever information NOC requested earlier. The process is repeated normally for each variable and when one single router has been completed NOC moves on to the next.

In recent years, however, paths in the Internet significantly increased in capacity and became *service-oriented*. As services normally operate on *end-to-end* basis and have various Quality of Service (QoS) demands, they require network administrators to provide *continuous monitoring* of network performance.

To clarify the above statement, let us imagine an end system providing some kind of service to users. The service is provided end-to-end on a path only a part of which is located within the service provider's physical network. This means that although the service provider may be able to measure the performance of its part of the end-to-end path, there is no traditional way to measure performance of the rest of the path.

In order to make this possible other network providers along the path would have to issue permission to the provider of the end-system so that passive measurement could be conducted.

There are two big problems with the above scenario when trying to implement it in the traditional passive way:

- end-to-end paths vary in length, which directly affects how fast the passive performance measurement can be completed before the performance of the end-to-end path can be estimated;
- just being in possession of passive *statistics* about each network device along the end-to-end path does not give you the ability to calculate *end-to-end performance*.

Physical impossibilities attributed to traditional network performance measurement are only part of the problem. With large capacity of links in networks nowadays, continuous monitoring of network performance cannot be done over SNMP due to the raw nature of metrics accessible through the protocol and large overhead created in data exchange.

For example, the knowledge how much network cards of each network device along the path are utilized (utilization is an SNMP variable that can be read off the remote device using SNMP), does not mean that you can calculate the *utilization* of the entire path unless all your network equipment is placed on the same network path and does not have any other communication partners elsewhere. This kind of scenario is possible in lab environment, but in the real life it is very rare.

So, instead of concentrating on a particular network device and its op-

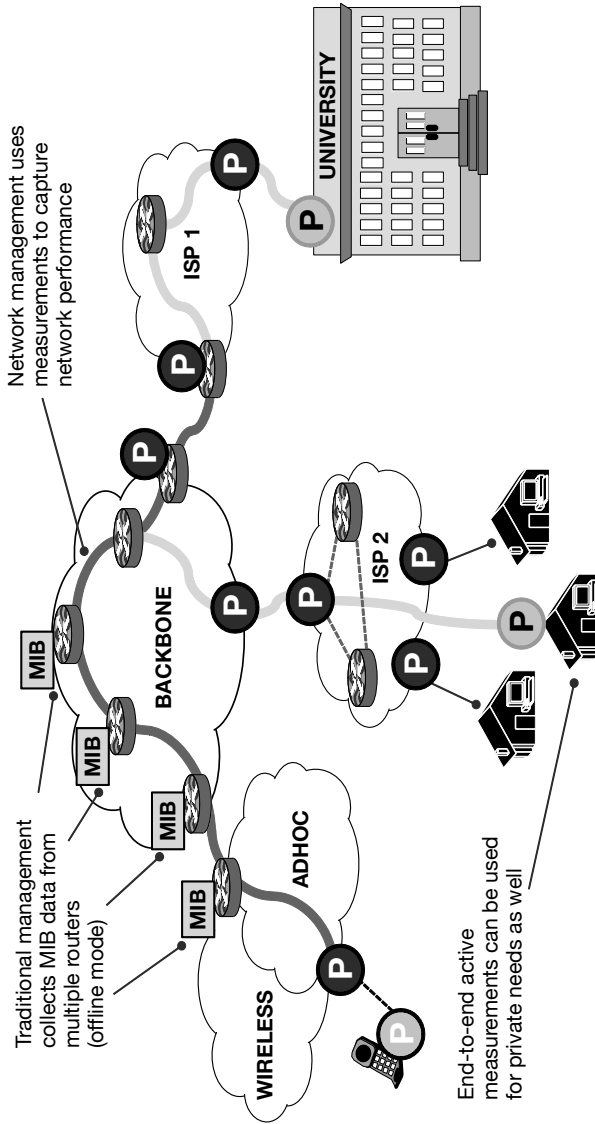


Figure 4.1 Current state of technology: traditional network monitoring and performance management using active measurement.

eration, monitoring is now performed on a whole network path. Figure 4.2 contains two separate paths that span several ISPs to connect edges of an end-to-end path. Since a network service uses this very path to send traffic between edges, measuring performance of this very link is a logical thing to do. *Probes*, therefore, are normally situated at edges of an arbitrary link and result of a measurement on the path is in some way characteristic of its performance.

This trend demanded a fundamentally different approach to network performance monitoring, which was fulfilled in active measurement. Figure 4.2 graphically displays the increasing demand in active measurements that has to do both with growing network capacity and diversification of QoS applications. The former process simply deprives network administrators of the ability to monitor network performance using SNMP. The latter is driven by a more complex process which involves a number of emerging technologies that require end-to-end performance measurements for their operation. Active measurement technology was created to fulfil these requirements.

In brief, active measurements are methods that are able to *infer* the performance of a network by probing it with specially designed packet sequences designed so that they reveal a particular performance characteristic.

4.1.2 IP Performance Metrics

The inability of traditional SNMP-based approach to provide end-to-end performance information demanded by emerging applications and services, caused Internet Engineering Task Force (*IETF*) to create a separate working group called *IP Performance Metrics (IPPM)*. The work of this group resulted in *RFC2330* on IP Performance Metrics released in 1997.

As IPPM work mostly focuses on end-to-end performance metrics, it went beyond the realm of tasks that can be performed by equipment inside the network itself. Instead, this kind of network performance inference would be referred to as active measurement or active probing and would be performed by hosts at the end of an arbitrary network path. Also, calculation of performance metrics would not be based on counters as was in the case of SNMP, but would be inferred indirectly from the result of interference of *active probes* with *cross traffic*. Active probes consist of packets size and interval which are set in such a way that the result of interference with cross traffic should indicate network performance. The list of performance metrics defined in IPPM is displayed in Figure 4.3.

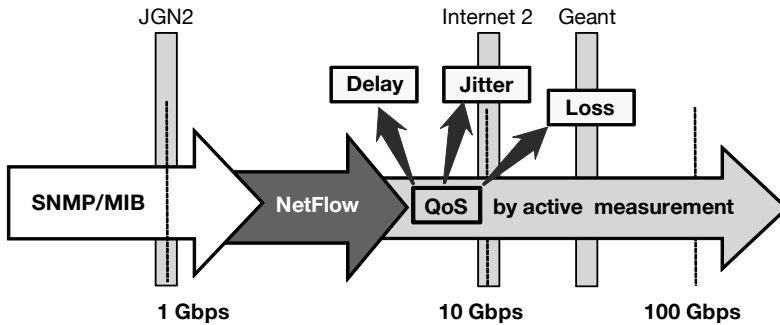


Figure 4.2 Demand of active measurement from the viewpoints of QoS and increasing network capacity.

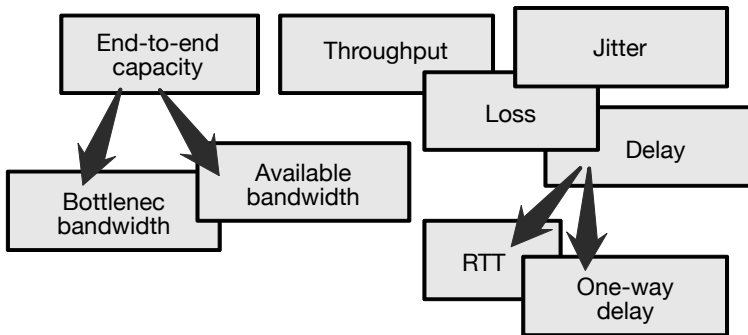


Figure 4.3 Performance metrics defined by IPPM.

In order to create a good active measurement method, one has to have comprehensive understanding of the main players in network traffic. This demands an extensive study of network traffic at various levels of aggregation. As end-to-end measurements are demanded by end applications, the traffic is analyzed from the viewpoint of traffic sources, where each source is a separate application. To perform this study, traffic was collected in LAN and WAN environments at flow level. The study proves that major applications exhibited in traffic affect distribution of main traffic metrics at flow level. Additional study of traffic at packet burst level is needed to understand the scale of interference of probing packets with main traffic components at any given point in time. The fact that traffic in

the Internet has a bursty nature has been proved in many research works on traffic analysis. This work facilitates correct choice of parameters used for active probing, such as packet size, time interval between packets, length of probe, etc.

4.1.3 The Scope of Active Measurements in this Book

All active probing methods developed in this study use *packet-pair property* as a basis of all original methods created in the current study. The packet-pair property itself leaves room for high statistical errors, and, therefore, this study mostly uses major modifications of the basic packet-pair. Some of developed methods use longer sequences containing a few packet-pairs for higher error resilience. Some other methods require special *probe designs* that involve packets of different sizes. Each specific probe design that is used by the present study explores specific performance metrics that are required by each measurement method. Another major concern of the present study is the ability of adapt to performance changes in the process of probing, which are referred to as adaptive methods in this study. Main portion of analysis in this study is beyond traditional tools offered by statistics. Instead, a number of processing tools were created to perform a number of specific tasks.

Finally, as there cannot be an active measurement research without a practical implementation in form of a measurement tool, special attention in this study is paid to implementation of several research ideas developed through a number of years of research. Each of these several projects targets a specific area within the field and is based on an original *measurement method*. All projects use an active measurement platform in both simulation and real network tests. Theoretical assumptions in each project are also based on end-to-end traffic properties established through the extensive traffic analysis that also can be found in this study.

4.2 Network Performance Metrics

This section considers a few targets attainable by active measurement, as defined by RFC2330 [41]. The list of IP performance metrics in RFC2330 is much longer than the list of metrics considered in this section, but some of the metrics either refer to physical link properties, such as, for example, *connectivity*, or are not really usable in a practical way, as, for example, is

the *one-way delay*, which is hard to measure due to the requirement to provide very precise synchronization between measuring hosts. Although such synchronization can be provided for a limited measurement topology, it is not feasible for global scale measurement topologies.

4.2.1 Bottleneck Capacity

Bottleneck capacity is also referred to as *bulk transfer capacity*. It is defined as the capacity of the narrowest link along the path. This definition is very easily explained from the end-to-end viewpoint. As all applications normally operate on end-to-end basis, it is important for an application to tune transmission rate to the narrowest link, i.e. the capacity of the entire path, and not to the capacity of the link it is physically attached to.

To find which link is considered the narrowest in the network is not such a straightforward problem. If there is very little traffic all the way on the end-to-end path it is likely that the hop with the smallest transmission speed becomes the bottleneck. But there are also cases when a hop may have very large capacity but be constantly loaded with heavy *background traffic*, in which case this hop may appear to be the bottleneck from the viewpoint of end-to-end measurements.

Although the definition of bottleneck capacity is fairly simple, a non-trivial statistical processing of active measurement samples is required to infer the value of the bottleneck capacity. Some of statistical solutions in this are considered later in this book.

4.2.2 Available Bandwidth

In plain words, if a link with the largest capacity on the path appears to be the bottleneck based on active measurement results, this indicates that your available bandwidth is low.

For an end-to-end application, to provide reliable transmission it is not only important to infer the bottleneck capacity, but also to learn about available bandwidth at the time the service is being provided to the user over a multi-hop end-to-end path.

Available bandwidth is defined as the unoccupied bandwidth in a link at the time of probing. Similarly, available bandwidth of a path is the bandwidth of the most congested link on that path, as the path can perform only as well as the link with the worst performance can perform.

A major research challenge in available bandwidth measurement is the

verification of results. As performance is changing along with changes in cross traffic, the reliability of available bandwidth measurement results is difficult to establish. However, one method of verification that is commonly used is to get SNMP readings of utilization in parallel with conducting active measurements. This way peaks in utilization should coexist with drops in available bandwidth. However, as it is impossible to synchronize passive utilization monitoring and active probing, this method of verification is intrinsically unreliable.

It is also interesting to note that statistical targets in bottleneck bandwidth versus available bandwidth are directly opposite. While in case of bottleneck bandwidth measurement you use statistics in order to get rid of noise in your data, the same very noise is precious when you are measuring the same network using almost the same packet probes but need to learn the available bandwidth.

4.2.3 Round Trip Time

Round Trip Time (*RTT*) is without doubt the most popular performance metric today. It is very easy to implement as no synchronization is required between the source and destination in each path. Instead, the destination can simply be configured to send ACK packets of some form back to the source as soon as it receives the probing packet. When the source receives the ACK packet, the RTT is calculated as the time of ACK arrival minus the departure time of the measurement packet, thus becoming the RTT value.

RTT measurements can be used to roughly estimate the *physical distance*. In fact, that is how it is used within the famous *Skitter* project conducted by *CAIDA*. This project is used to create a map of the Internet where distance of links is defined directly from RTT measurements. In fact, *Skitter* is a continuous endeavour and is active to the day continuously probing and collecting rough estimates of distance from thousands of nodes across the global Internet.

There are a few new artefacts in networks nowadays that invalidate RTT measurement's usability as distance estimates. For one, *asymmetric routes* are becoming more and more common in the global network as ISPs separate their uplinks from downlinks often running each through a difference partner ISP. In this case the return path for the ACK packet can be substantially different from the forward way, thus, making it impossible to estimate distance by simply taking a half of RTT.

It is also a well known fact that end-to-end delay can be different for

various underlying networking technologies. But since it is impossible to detect the networking technology used at layer 1 and 2 along the path, this nuance in RTT measurements is normally ignored.

4.2.4 End-to-End Jitter

End-to-end jitter brings us back to application world again. There are many applications that demand steady transmission rate. Constant Bit Rate (CBR) transmissions that are used in some video and audio streaming are the two major Internet-based applications of this kind.

The definition of the end-to-end jitter is the degree of changes in traffic in realtime. End-to-end jitter measurements are normally based on RTT measurements by simply calculating the standard deviation from the mean RTT to represent jitter. However, one measurement project conducted within the framework of this book deals with this research problem and proves that a special probe structure can be used to indicate jitter directly from measurements. This saves the need to store a pool of data samples and perform complicated statistical calculations to infer jitter indirectly.

4.3 Life of a Single Packet in the Path

This chapter is looking into basic approaches that exist in active measurement today. It is important to understand the basics of active measurement technologies in order to be able to easily split existing methods into well-known categories. In fact, there are only two major categories of active measurement approaches. Those are *single-packet* and *packet-pair* techniques. Originally, the single-packet technique was the first in place [31]. However, the intrinsic shortcoming of the single-packet technique stimulated creation of a fundamentally new approach, which was called a packet-pair based method due to the fact that it was physically based on probing by pairs of packets.

As there are two fundamentally different measurement approaches in active probing, this chapter focuses on these two fundamental approaches. However, there are also a few methods that extended the fundamental techniques by either constructing additional *probe structures* or introducing special statistical processing methods in order to provide higher reliability in estimates. This study does not focus on details of such statistical methods but discusses special probe structures in details. The reason for such a restriction is that statistical methods can deal with errors only at

the level that is provided in raw data which, in its turn, depends on the error-resilience of a particular probe structure. Therefore, it is important to design an *error-resilient* probe structure prior to conjuring statistical framework for processing its results.

Besides, special probe structures are also useful because by designing a particular structure of a probe it is possible to target a specific performance metric of a network and attenuate all others.

Single packet probing techniques were originally developed for a specific purpose of measuring capacity of each link along an arbitrary path. However, this apparently rigorous purpose of a probing tool was not due to its targeted applications but due to the fact that when probes by single packets, the bottleneck of any arbitrary path can be discovered by singular packets only if the measurement were performed in a recursive manner, adding hops one by one in each loop. Originally, the method was proposed in [31].

4.3.1 Measurement Methodology

The key element of a single-packet probing technique is the recursive nature of probing, i.e. a path is probed in stages where each stage is completed by adding the next hop from a path until the end of a probing path is reached. The topological view of the procedure is depicted in 4.4. However, recursive process alone is not enough to infer the capacity of a link, as the obtained data lacks a reference point. The best solution is to use a number of manually set packet sizes to probe with at each stage. This manner of size-differentiated probing allows to perform comparative sta-

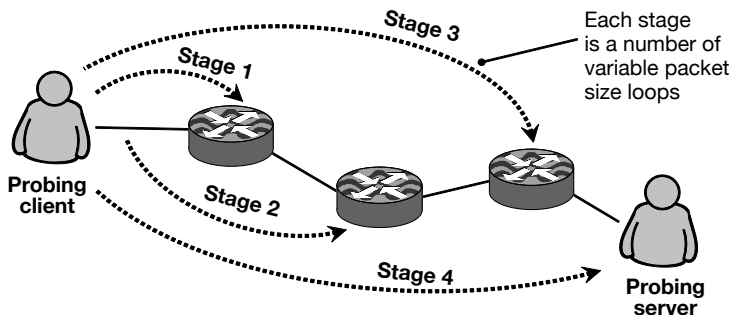


Figure 4.4 Design and sequence of a basic single-packet probing technique.

tistical analysis on RTT against the packet size and finally infer capacity of each measurement target.

Common network equipment does not naturally support a hop-by-hop probing. That is, it is not possible for a common user to send a query to a router and expect a reply. However, it is possible to use Time To Live (TTL) field of ICMP protocol, which allows for setting the maximum count of hops allowed for a packet to traverse. Each router on the path of the packet subtracts the contents of the TTL field by 1. Finally, when the value of the field reaches 0, the router that encountered such packet should drop it and send an ICMP “Time Exceeded” packet back to its destination. This feature was devised at the very creation of routing protocols and therefore is still widely implemented in routers nowadays. In short, the procedure of probing is as displayed in Figure 4.5.

Singular packets originate at Probing Client. The process of probing is split into as many stages as there are hops on the way to the client. The Client does not have to know how many hops there are as the counting can stop once the packet reaches its destination, i.e. Probing Server. One-packet probe techniques assume that at least one packet of in each group of the same size should not encounter any queuing delay.

Providing that each fixed size group has enough packets to encounter relatively “quiet” period in path’s traffic, each group should have the lowest reading as per Figure 4.6. Although the samples in the figure are just an example of single-packet measurement results, general trends are preserved. Those general trends are depicted by a linear function of minimal RTTs for each packet size.

Let us assume a packet of size S traverses a list of capacity C . Then, it will take C/S time to transmit the packet through the link. As was men-

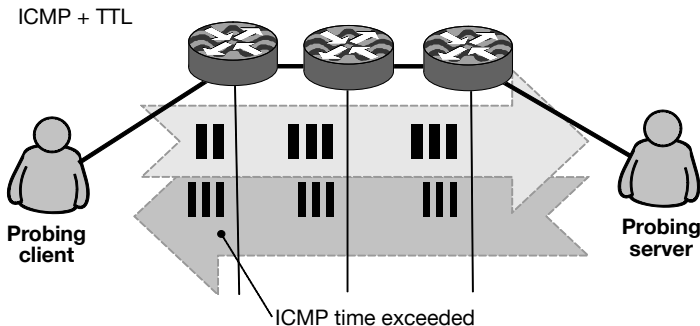


Figure 4.5 Probing sequence and probe structure of single-packet probing techniques.

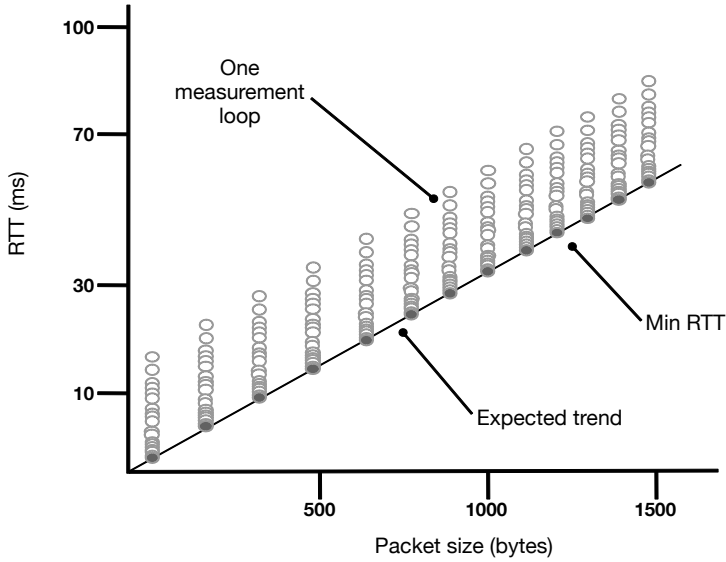


Figure 4.6 View of data as received from probing by variable size single packets.

tioned before, some queuing delay may occur in the buffers of routers or switches, but this delay can be ignored as only the minimum time will be used in the end. The minimum RTT for each specific packet size will consist of two terms: a delay that is independent of packet size and a term that is proportional to the packet size due to serialization delays at each link along the packet's path. Therefore, the minimum RTT $T_i(S)$ for a given packet size S at the hop i can be calculated as:

$$T_i(S) = \alpha + \sum_{k=1}^i \frac{S}{C_k} = \alpha + \beta_i S. \quad (4.1)$$

In Eq.4.1, C_k is the capacity of the k th hop, α is the aggregation of all delays up to the hop i independent from the packet size S , and β_i is the slope of minimum RTT values at the hop i for a probing size S . The slope is given by:

$$\beta_i = \sum_{k=1}^i \frac{1}{C_k}. \quad (4.2)$$

The physical meaning of the slope is depicted in Figure 4.6 in the form of the straight line that connects minimum RTTs obtained from various sizes. A simple rule for calculating capacity of each hop based on the slope of the current and the previous hops can be given as:

$$C_i = \frac{1}{\beta_i - \beta_{i-1}}. \quad (4.3)$$

4.3.2 Shortcomings of Single-Packet Probing

Single-packet probing has one major disadvantage of significant estimation errors. First, RTT readings at layer-3 devices is not reliable as layer-2 devices, such as switches, bridges, etc., introduce additional propagation delays that is one of the main components in the calculation of capacity based on singular packets. Layer-2 devices do not respond to TTL Time to Live setting in packets and are invisible to the measurement. Modifying single-packet probing to avoid such errors remains a research problem to the present day.

Besides the high error rates in calculations, single-packet techniques also suffer from technological point of view. Recent statistics in networking equipment proved that about 13% of all layer-3 devices drop *ICMP* packets silently without sending *ICMP Time Exceeded* packet back to the source. This is necessary as such a reply technique can be exploited as a security hole that can easily allow for creation of large floods of *ICMP Time Exceeded* packets in the network. Therefore, *ICMP*-based probing techniques are not reliable in view of future development of networking technology. The use of common protocols like *TCP* and *UDP* is not possible as these packets are also silently dropped midway without any reply packets. In some cases this field may not even exist if a packet or layer-3 routing equipment can be completely oblivious to it.

Finally, single-packet probing has intrinsic problems with large traffic overhead created by probing. Each added hop requires considerable number of measurements with each particular packet size. As each new hop requires probing with many packet sizes in order to create a reliable slope, the overhead can easily reach the level of a few megabytes for each end-to-end measurement. This and the fact that the process is also time-consuming are the major disadvantages of all single-packet probing techniques.

4.4 Packet Pair Property

Packet pair probing is normally used to measure end-to-end capacity. Its name originated from the underlying probing technique that in fact uses two packets for a single measurement. This is different from single-packet technique in that the use of two packets allows for calculating capacity based on relative metrics other than the absolute RTT as was in the case of single-packet probing. That is, packet pair solved at least one major problem of the single packet technique.

4.4.1 Measurement Methodology

As depicted in Figure 4.7, the measurement using packet pair is done in *end-to-end* fashion, which means that it does not matter how many hops packets have to traverse to the other end of the measurement path. In fact, it is clear that the capacity of each path is limited by the capacity of the narrowest link in the path. This link is traditionally referred to as the *bottleneck*. Therefore, original packet pair techniques address the problem of bottleneck capacity measurement. As there is normally only one bottleneck on each path, an end-to-end measurement procedure should be sufficient to get the results. More details on packet pair methods and their comparison with single packet approach can be found in [22].

All packet-pair techniques require the condition in which the packets are queued at the bottleneck link in a *back-to-back* fashion. In this fash-

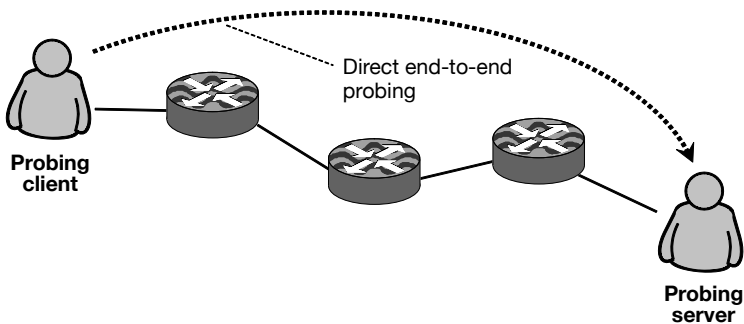


Figure 4.7 Measurement procedure of packet-pair based measurements.

ion a relative zero point is created in the probe at departure and which is why inter-arrival time is indicative of the bottleneck capacity. The back-to-back condition is not so difficult to achieve at most networks, especially at public access networks. In such networks at least one link in a path is drastically different in capacity from all the other links, which automatically creates the back-to-back condition at the entrance of the bottleneck. However, most methods try to achieve this condition by sending packets back to back or with minimal achievable space between them.

Providing two measurement packets of the same size S queued back-to-back at the entrance of the bottleneck, the capacity of the bottleneck C can be easily calculated from the interval T that occurs between the packets in the bottleneck and is preserved throughout the rest of the path:

$$C = \frac{S}{T}. \quad (4.4)$$

As was mentioned before, the capacity of the bottleneck is normally quite different from the average capacity of the path, which is a good guarantee that packets will not queue back to back again at any hop until the end of the path. The process is depicted in Figure 4.8.

Even if there is a small time interval t_1 between the packets at the entry into path, this small interval is obliterated at the bottleneck, where the packets have to queue back-to-back due to the fact that the bottleneck link has less capacity than the previous path. Therefore, the time t_2 should be zero. If this is the case, the bottleneck itself will introduce the interval t_3 between the packets, which will be preserved throughout the rest of the path.

Due to the fact that all metrics in case of packet pair are relative, as is the inter-arrival time itself, etc., it is difficult to make calculations with one-way measurements only. Therefore, packet pair measurements are normally conducted in a round-trip fashion, when the Probing Server's sole purpose is to send an ACK packet immediately after receiving a probing packet.

This kind of immediacy constraint makes it difficult to use TCP protocol, as the method of sending ACK packets depends on any particular TCP protocol implementation. This is the reason why UDP protocol is more commonly used for packet pair based methods.

Another special feature of any packet pair probing method is the extensive use of statistics. Figure 4.9 is a basic distribution of samples where each sample is a measurement obtained from a single packet pair. In case of single packet probing, for each group of measurements with a fixed

packet size only the minimal RTT would be finally used to get a slope. This is a good way of obtaining a *reference point*. As the packet pair metrics are relative there is no reference point. Each measurement sample in packet pair case is of equal importance and cannot be filtered out based on any prior assumptions about network conditions or the bottleneck capacity.

As the example histogram in Figure 4.9 has it, the distribution of measurement samples is normally *multimodal*, i.e. has more than one major maximum. The lack of ability to assume any prior conditions makes all the maximums in each distribution equally probable. Most methods, however, simply take the largest maximum and declare it the bottleneck capacity. Some of the research conducted within the framework of this book proves that this is not always correct. Heavy congestion in the path, for example, renders the assumption about the highest maximum completely invalid. In case of heavy congestion, packet pair may queue back-to-back at a link other than the bottleneck and the measurement results will be representative of a completely different hop within a path.

There are, of course, other conditions which do not agree with simple choice of the highest maximum in distribution. Normally each newly introduced method offers a new statistical approach used in processing of measurement samples. The detailed study of a long list of such methods is beyond the scope of this book.

4.4.2 Packet-Pair versus Single-Packet Techniques

As was already mentioned, single packet techniques are fundamentally different from packet pair in that the packet pair operates with relative metrics which are obtained by observing the differences in the behaviour of the first and the second packet in each pair. On the other hand, single packet techniques use solid RTT readings and can perform fairly complicated calculations.

In case of packet pair, the error resilience in processing of samples is not easily achievable because of multimodal distribution of measurement samples. Therefore, single-packet techniques affect error resilience by changing a method of statistical processing, while packet-pair techniques conjure various probe designs that would introduce additional error resilience.

Another major difference between single packets and packet pairs is the traffic overhead introduced by probing. While any single packet probing technique requires multiple groups of measurements using various packet sizes for each hop, in case of packet pair each measurement results is the fully finished calculation sample. Therefore, packet pair offers more

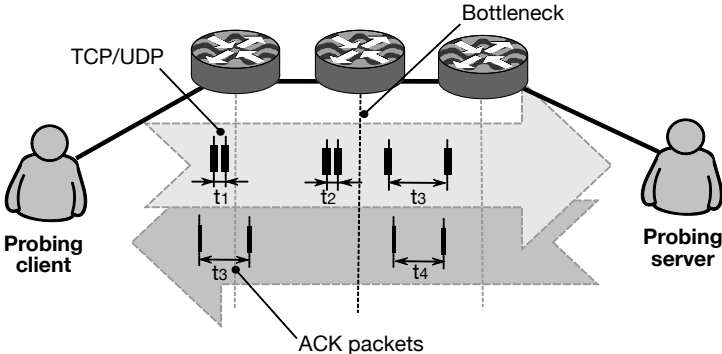


Figure 4.8 Probing procedure and probe structure of packet-pair base techniques.

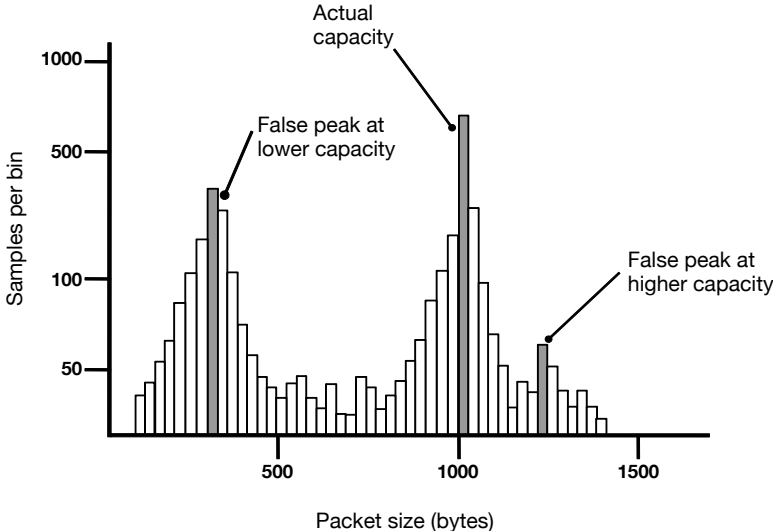


Figure 4.9 Example and difficulties in data analysis with packet-pair probing.

room to adaptive probing where packet pair configuration or processing algorithm would be changed online as soon as a certain condition is detected in the network. A few research projects described within this book contain original methods that enhance original packet pair in order to provide higher error resilience or to be able to measure network performance metrics other than the bottleneck capacity.

4.5 Advanced Probe Designs

Various probing methodologies create packet-pair derivatives not only for the purpose of providing higher error resilience in measurement results, but also to be able to measure a specific performance metrics. For example, some packet pair based techniques target available bandwidth, which is defined as the unoccupied portion of capacity in a path at any given point of time. As packet pair is much easier to handle and allows access to measurement results in the real-time, it is much more popular among measurement methods than single-packet techniques.

This section introduces a few distinct enhancements to the original packet pair measurements. As the present study also contains results from a few major modifications of the packet pair, it is important to discuss other research works for comparative reasons.

4.5.1 Piggyback Methods

Original piggyback method was introduced in [34] and later improved by the same author in [35] and [36], finally resulting in a tool that implemented the method proposed in the research.

Piggyback method operates very similarly to a single-packet technique as it also allows for targeting each hop in a network. However, as depicted in Figure 4.10, this method does not have to stop at every single hop to perform the tedious recursive process. In case of a single-packet technique, the recursive process is required because the calculation of the capacity at each new hop is based on the capacity of the previous hop. The packet-pair piggyback technique relaxes this constraint and instead allows for targeting any particular hop in the target.

The technological assumption of hop-by-hop measurement is based on the exploration of the same feature of layer-3 devices that respond to ICMP packets with expired *TTL* field with *ICMP Time Exceeded* message

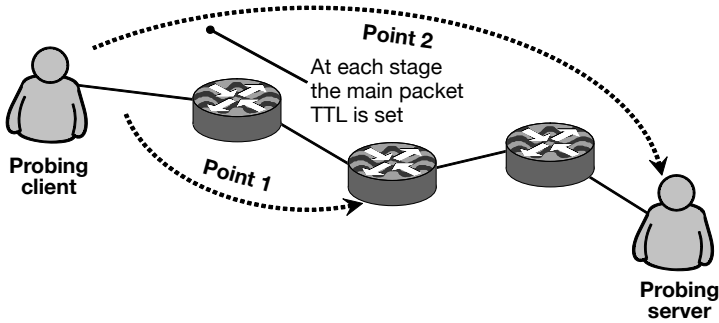


Figure 4.10 Probing capabilities offered by piggyback probing.

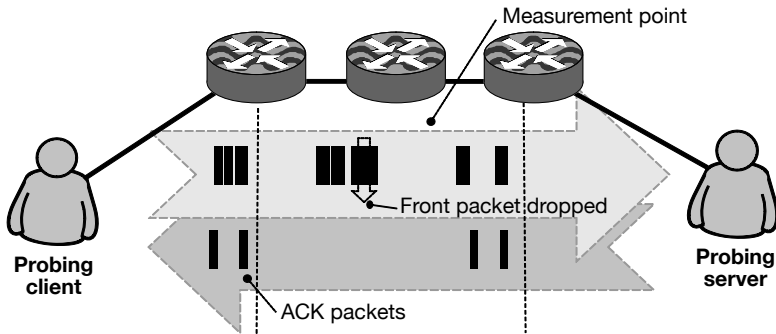


Figure 4.11 Probing procedure and probe structure used in packet-pair piggyback probing.

sent back to the originating host. However, the similarity to a single-packet technique ends there.

As depicted in Figure 4.11, each probe contains a packet pair that is transmitted right after a large ICMP packet. This constitutes the piggyback property of the measurement. Because the head ICMP packet is considerably larger than the following packets from the packet pair, the packet pair will end up queuing back-to-back after the ICMP packet at all hops in the network. Physically, this can be also explained by the differences in speed with which a packet of a certain size traverses the network. The speed has linear dependency on the packet size, i.e. larger packets travel slower than smaller ones.

As per the example in Figure 4.11, the probe targets the second hop in the path. The TTL field of the head ICMP packet is set to 2. The second

router registers zero value of TTL and drops the packet. In this case it is really not important whether the router responds with an ICMP Time Exceeded back to the originating host, as the main measurement is performed by the packet pair.

After the head packet is dropped, the packet pair performs the required measurement of the bottleneck capacity of the rest of the path. This part follows the guidelines of the original packet-pair property.

The abilities of the piggyback method are much more enhanced as compared with traditional packet pair. Not just is it able to measure capacity back-to-back, – it also allows for detecting heavy congestion conditions in the network by pinpointing the location of the hop that is the source of the heaviest interference of probes with cross traffic.

4.5.2 Packet Trains

Packet trains were first introduced in [32] in form of a tool called *Pathload*. This tool nowadays is the most reliable among all available bandwidth measurement tools and is commonly used as a reference point of research. Prior to creation of *Pathload*, the author of the tool also studied in details the properties of *packet dispersion* techniques [28]. Packet dispersion techniques here stand for all techniques that are based on the original packet-pair property.

However, packet trains are quite commonly used in other research works as well, and, therefore, require a detailed explanation of underlying probing technique. Let us consider a *probe train* in Figure 4.13. The probe consists of a number of packets in which the interval between the packets is gradually decreasing. The decreasing time is a major requirement of packet trains that target available bandwidth. Normally, the decreasing rate would be exponential to cover a wide range of values. This exponentially decreasing probe can also be viewed as a number of packet pairs with different interval between pairs of packets. In fact, this is exactly the way the probe trains are viewed because all statistics come from the metrics obtained from individual packet pairs within the probe.

It is necessary to introduce another metric, which can be either the capacity measured by the packet pair or simple raw value of the time interval at the time of departure and arrival. The discrepancy between departure and arrival is also crucial as the whole measurement is based on the comparison between the two.

So, a metric that is visualized for each pair in the train is the mere difference between inter-arrival time at the arrival and the original inter-

arrival time at the departure. Now, the cases depicted in Figure 4.12 could be as follows. Case 1 would consist of the samples 1, 2, 3 and 4, which are stable and stand for a static difference between inter-arrival readings. In fact, this difference is supposed to be close to zero as packets in this case do not interfere with cross traffic nor are affected by bottlenecks. This case physically means that the rate at which the packets are transmitted is lower than the available bandwidth of the link. The packets do not queue back-to-back in this case and instead flow freely all the way to the destination thus retaining the original interval between packets.

Now, starting from a certain position in the probe, the rate of the pairs becomes higher than the available bandwidth. In this case, the packets in pairs cannot retain the interval they obtained at the departure and start queuing back-to-back. This creates the perfect conditions for the traditional packet-pair property in that the interval between packets in pairs represent the capacity of the link. It is also true that the inter-arrival space at the arrival in this case would be higher, i.e. the difference would also be higher, which is the case for samples 5, 6, 7, and 8.

Now, coming back to the basic assumption of available bandwidth probing, it states that the point where the difference in inter-arrival times at arrival and departure becomes positive and keeps increasing with further pairs in the train, can be considered the available bandwidth. Quite literally, in the above example in Figure 4.12, the pair 4 is found to be the *breakpoint*, which means that the available bandwidth at the time of measurement is $C = S/t_4$.

The breakpoint can occur at various points in the probe depending on the range that the probe covers, i.e. the range of rates between S/t_1 and S/t_n where n is the number of packet pairs in the train. The position of the breakpoint also depends on the available bandwidth at the time of the measurement, which is the ultimate goal of the measurement itself.

To focus on available bandwidth, each method that uses trains to measure available bandwidth normally provides a fairly wide range of rates in the train. This way the research can focus on finding the breakpoint in the probe. The failure to provide broad enough range may result in something similar to cases displayed in Figure 4.13. Invalid Case 1 stands for the case when the first packet pair in the train was transmitted at the rate higher than the available bandwidth, which created the patterns without the reference point. Invalid Case 2, on the other hand did not reach the breakpoint, i.e. packet pairs in the trains did not have a transmission rate higher than the available bandwidth. Both patterns in Figure 4.13 may still have minor maximum points, but those are not the breakpoint as they are caused by interference with cross traffic of that particular pair that

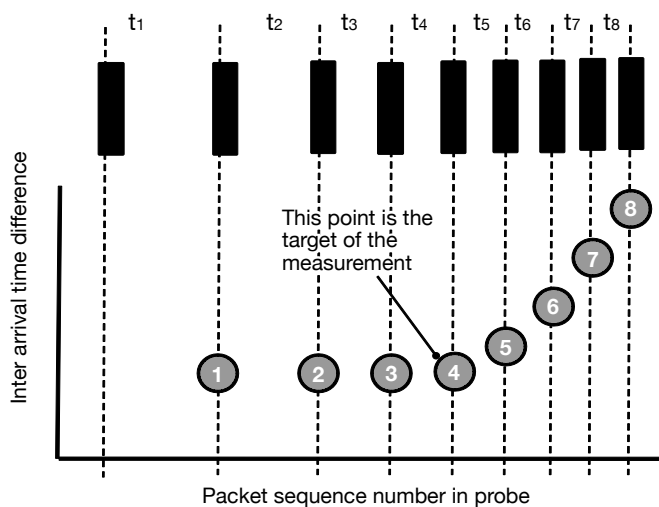


Figure 4.12 Example of interarrival time distribution exploited by packet trains.

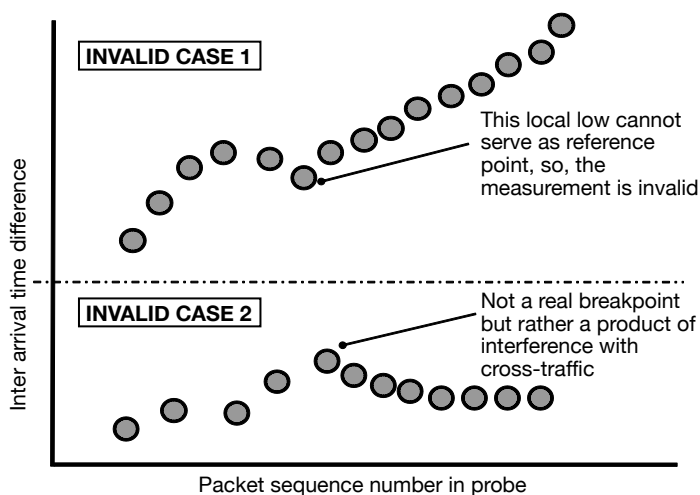


Figure 4.13 Cases that invalidate available bandwidth measurements due to the lack of a reference point.

caused the spike. The breakpoint is easily distinguished as the increasing difference never decrease once past the breakpoint.

4.6 Routing Peculiarities

All the network performance parameters considered in this chapter may create the impression that active measurement is very distant from the physical connectivity. In fact, the inability to properly connect actively inferred performance metrics to physical network phenomena is probably the most actively argued point in active measurement research. This section attempts to theorize the fundamental problem created by active probing via a set of lemmas that are generally true for all end-to-end measurements.

4.6.1 Directionality of Probing Path

Probing paths are directional, as described in Figure 7.1. The *measurement time series* obtained from a measurement on AB is not the same as the time series from CD . Although the traffic in both directions may go through identical intermediate routers, the interference between packets flowing in opposite directions is minimal. The majority of traffic interference originates from background traffic packets lining up before the probing packet in each router along the path. This fact is well established in active probing literature.

For simplicity, the measurements in this book are performed in one direction. Naturally, the background traffic probes also interfere with background traffic flow only if the direction of the traffic flow matches that of the probing stream.

4.6.2 Loose Coupling

The loose coupling problem, depicted in Figure 7.2, can also be categorized as the lack of synchronization problem. It is stated as follows.

Even if measurement paths AB and CD are identical, there is no guarantee that the measurement time series will also be identical. In fact, even if A and C send their probes at the same time, one of the two packets from two different origins will have to wait for the other thus creating subtle differences in the resulting time series.

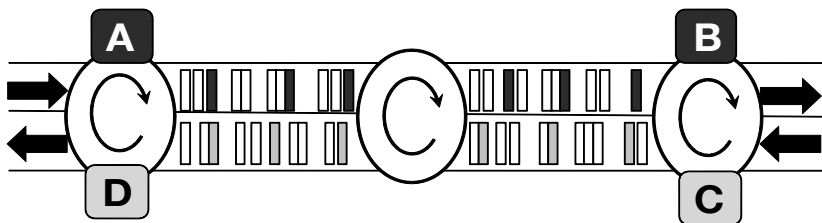


Figure 4.14 The problem of directionality in active probing.

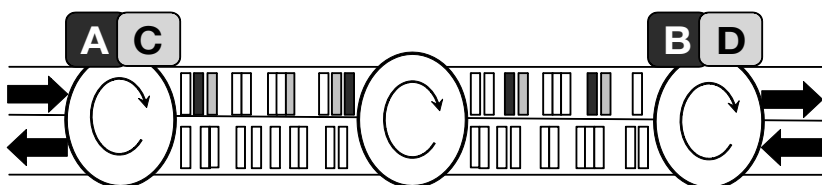


Figure 4.15 Problem of loosely coupled probes.

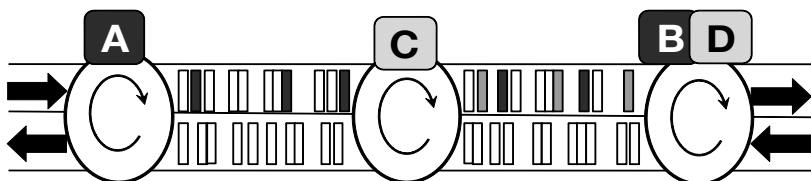


Figure 4.16 Shared topology in probing.

In reality, however, perfect synchronization of *A* and *C* is difficult, which explains why probing packets experience different interference patterns as they travel separately through and end-to-end path.

To compensate for loose coupling a method has to allow a certain degree of freedom in terms of missing features and local differences in the two *time series*. Here, it is important to stress that conventional *data mining*, including time warping techniques, react poorly to a loss of features in a pattern. The proposed data mining method later in this book is more tolerant to such losses.

4.6.3 Shared Topology

The case of shared topology depicted in Figure 4.16 is slightly different from all the cases above. All the previous properties have been rather negative by stating certain assumption impossibilities. The property of the shared topology in Figure 4.16 has two sides.

On one hand, the property should be similarly pessimistic by stating that AB in Figure 4.16 is not the same as CD , which is obvious. More than that, this case is even worse than the previous case where two paths were shared completely. However, the interesting part in shared topology does not lie in traffic interference. Clearly, as far as interference with cross-traffic is concerned, AB is not the same as CD . However, if traffic interference is ignored altogether, shared topology opens interesting prospective for active network measurement.

In active measurement methods that exploit shared topology packet order is normally used to detect whether the path is fully or partially shared.

Based on Figure 4.16, if A can be synchronized with C at a sufficient time granularity, there is a way to detect whether the two paths shared the same topology. Unfortunately, this book will not be able to accommodate the various examples of such methods, but it should be clear that in the contemporary world of rapidly evolving global-scale networks, such technology will inevitably become popular in view of booming global network services.

Chapter 5



Active Measurement Methods

This chapter is dedicated to a number of projects conducted within the framework of this book as implementations of original probing methods proposed by the authors. All the projects can be split into two major groups. One is projects that target the *effectiveness* of actively measuring a certain performance characteristic. This is still an ongoing research in active measurement community as none of the methods proposed to the day are *standardized*, and, therefore, all proposed methods have equal chances to be selected as the most optimal *methodology* for *inferring* a particular metric from the IPPM list.

As the result of a number of separate research works performed by the authors, this chapter will look into methods used to effectively measure bottleneck capacity, available bandwidth, and jitter.

5.1 Adaptive Capacity Measurement

From all the areas of network measurement, the oldest and the most well-studied part of measurement technology targets *bottleneck bandwidth*, which is defined as the link with the smallest bandwidth along an arbitrary path. Considering the fact that the Internet is extremely heterogeneous and may span several different networking technologies on a single path, the task of defining the bottleneck bandwidth of a path may be challenging.

Another characteristic of an arbitrary network path that receives considerable attention today is *available bandwidth* and *throughput*. Available bandwidth indicates what portion of the total bandwidth is not utilized at any particular point of time. Since available bandwidth constantly changes, measurement tools are expected to probe *continuously*.

As measurements are dealing with multi-hop network paths, performance metrics for each hop on the path may also become the target of the measurement. However, due to the intrinsically uncooperative nature of today's networks, *hop-by-hop measurements* are very unreliable not to mention the large amount of activity that is required to perform such measurements.

End-to-end measurements, on the other hand, treat the network as a black box and do not require any cooperation other than a certain response of the network path to probing packets. By performing analysis on probing data, certain performance metrics of the network can be estimated with the help of statistical methods. The majority of recent successfully implemented technologies, such as tomography and proactive monitoring, probe end-to-end without prior knowledge of network topology. Review of some of them can be found in [27].

Validity of measurement results has always been a hot research topic [39]. Many currently existing measurement tools indicate broad range of errors in results and can exhibit differences in behaviour when used in different networking environments [27]. *Intrusiveness*, that is, the load imposed on the network by probing, can also vary greatly for different methods.

High error in measurement results can be explained by multiple modes in data which make it impossible to make an assumption about the character of traffic at the time of measurement [29]. This also makes it impossible to filter measurement data before making an estimate, which forces a measurement tool to use raw data as is.

In this section, it is proposed to partially solve the *validity problem* in bottleneck bandwidth measurements by introducing a relation between network condition at the time of probing and probing parameters. It is believed that this will help to suppress unwanted modes in measurement results, if not to eradicate them completely. A standard change detection function [21] is used to learn of anomalies in network state in real time, and then to use this knowledge as a *feedback*, based on which probing parameters are altered to adjust to the change. This causes the probing process to "follow" cross-traffic in the network at the time of probing. As proved by simulation results and from real network tests, this assumption is, in fact, legitimate.

5.1.1 Bottleneck Bandwidth Estimation

Figure 5.1 offers graphical representation frequently used to depict network bottlenecks. Each hop on a path is represented as a pipe of a certain size, with the *bottleneck* being the narrowest of them.

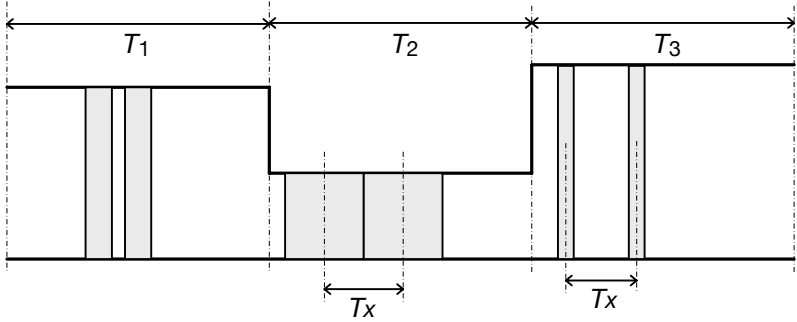


Figure 5.1 Graphical representation of a bottleneck.

Given that a packet of a certain size S traverses the path in Figure 5.1, it is possible to represent one-way delay as a simple summation of delays at each hop :

$$t = T_1 + T_2 + T_3. \quad (5.1)$$

Each of the members on the right side of (5.1) represents the time that will be required for the packet of a certain size to traverse a single hop of the path. Processing time at each node in the network is very small compared to the time each packet spends in transmission and can be disregarded within this simple representation scheme. Since the bandwidth of each hop is different and the packet size is constant, delays for each hop are different, that is, the packet traverses each hop with a different speed.

As it is assumed that the path has a bottleneck, packets will have to slow down inside of the narrowest link. If there are several packets traversing the network next to each other, they will create a queue at the entrance into the bottleneck, as each packet will have to wait for its predecessor to get transmitted. Therefore, if two packets of sufficiently large size traverse the path back-to-back, arrival times at the other end of the path would be different for each packet and can be represented by (5.2) and (5.3).

$$t_{arr}^1 = t_0^1 + T_1 + T_2 + T_3 \quad (5.2)$$

$$t_{arr}^2 = t_0^2 + T_1 + T_x + T_2 + T_3 \quad (5.3)$$

T_x in (5.3) represents the time that the second packet has to spend waiting for the first packet to be transmitted by the narrowest link. t_0^1 and t_0^2 are start times for each packet, and T_1 , T_2 and T_3 are transmission times over each hop in a simple path depicted in Figure 5.1. The presence of T_x is indispensable for *packet-pair property* and can only happen when both packets are transmitted back-to-back, i.e. the difference between t_0^1 and t_0^2 is very small.

Since packets always queue at the bottleneck, it can be states that the delay before the bottleneck is the same for both packets. This enables us to state that the difference ($t_{arr}^2 - t_{arr}^1$) in arrival times of packets at the opposite end of the path is equal to the time the second packet had to wait in the bottleneck for the first packet to be transmitted. Figure 5.2 displays the concept of packet-pair probing. In this model the host at the opposite end of the path responds to each probing packet by immediately transmitting a very small acknowledgment packet, which allows us to assume that the spacing obtained in the bottleneck will be preserved in ACK packets finally received at the source. This round-trip manner of measurement can be conducted without the need to *synchronize* the clocks at both ends of the path.

As the T_x is proportional to the bandwidth of the bottleneck link, one could write the value of the bottleneck bandwidth B as:

$$B = \frac{S}{t_{arr}^2 - t_{arr}^1} = \frac{S}{T_x} \quad (5.4)$$

Therefore, the bottleneck bandwidth in (5.4) can be found using only the packet size S and the *interarrival* space T_x created between the packets in the bottleneck. Other two key properties that are used in the present research are probe interval T_{int} and window of size W that is used for collecting and later statistical processing of samples.

As a pairs of packets is in question, there always exists a probability that each of the two packets will interfere with cross-traffic. If this interference is different for each packet, the packet-pair property in (5.4) may not produce a valid result.

To lower the probability of *interference* with cross-traffic before the bottleneck, the packets should be transmitted with the smallest possible

interval to ensure that no other packets will queue between the measurement packets at the bottleneck. The requirement for interval t_0 between the packets at the source can therefore be written as :

$$t_0 \ll \frac{S}{B} \quad (5.5)$$

According to that equation, for instance, for the bottleneck of 10Mbps and packet size set to 1500b, space between neighbouring packets at the source should be equal to or less than $((1500 + 8) * 8) / 10000000 = 1.2ms$. Slightly complicated nominator is due to the need to convert bytes to bits and compensate for IP header.

$$\frac{B}{t_0} \ll S < MTU \quad (5.6)$$

This could also be considered from the viewpoint of a packet size used in the pair. As per (5.6), it should be much larger than minimum possible size and yet less or equal to MTU not to get fragmented in the process of transmission. For ease of calculation, the value of t_0 in (5.6) could be set to the duration of timeslot in any particular operating system. For example, a bottleneck of 10Mbps cannot be measured with an ordinary Linux Kernel up to 2.4.x version because it provides time slices of 10ms duration.

5.1.2 Online Variable Measurement

Abrupt change detection [21] represented by (5.7) is not new and was applied in a similar way in [38]. It is also quite often used in network monitoring and management systems. No other research that applies the results of anomaly detection to measurement process was found in literature at the time this book was written, however.

$$g_k = (1 - \alpha)g_{k-1} + \alpha (y_k - \mu)^2, \quad g_0 = 0. \quad (5.7)$$

In (5.7), y_k is the current sample, μ is the mean of the pool of samples, and α is the forgetting parameter. The value of g_k is the indicator of an *abrupt change* at the present sample and grows with higher degree of change. This detection function is recursive, that is the next value g_k is based on the previous g_{k-1} . Forgetting parameter α is used to distribute the weight between the previous value and the one that is currently being

calculated. At $\alpha = 1$ the function will have a static outcome and will not feel a change, while with $\alpha = 0$ current outcome will not depend on the previous state. For simplicity and with regard to the fact that network parameters vary within a broad range of values, it is possible to allow neither the bias toward longer memory nor higher dependency on current samples. Therefore, in all tests within the framework of this chapter $\alpha = 0.5$ is used.

$$d = \begin{cases} 0, & g_k < h; \\ 1, & g_k \geq h. \end{cases} \quad (5.8)$$

A simple *threshold rule* is defined in (5.8) to identify changes. When the change is detected, i.e. $g_k \geq h$ and $d = 1$, the tool reacts to it by setting the *window size* and *probing interval* to the lowest margins for each of the values. If the change is not detected, both window size and probing interval values are incremented until they reach the upper bound of each. The value of change detection threshold h is set to 0.001 , which proved to be high enough to rule out all insignificant changes. The nature of network changes is such that changes are well clustered into two major groups: very minor changes similar to *white noise*, and major changes which reflect changes in network condition. Due to size limitations, the response of probing results to various settings of h is not offered in this book.

The present chapter proposes dynamic values for three probing parameters: *packet size*, *probing interval* and *window size*. Physical meaning of packet size and probing interval was explained previously and pertains to the probing action proper. Window size stands for the size of the pool used to collect measurement samples. The window size is important for *statistical processing*, as number of samples used in statistical calculations will directly influence the final estimate.

The setup of window size and probing interval dynamics used in this chapter is as follows. The upper and lower bounds of window size are set to 40 and 10 samples accordingly, and the increment is set to 2 samples per change. For probing interval the values are 5 seconds, 1 second, and 0.5 seconds correspondingly. This setup is by no means the perfect for all the networks. However, the study does not set an optimization goal, and will be satisfied with an improvement against the results from conventional static probing. Besides, a fairly broad range of values applied allows for analysing trends with different setup values. Changes shorter than 0.5 seconds or 10 samples are too short to be of interest. The same logic is applied to upper bounds.

Packet size dynamics are different from the two parameters above, and are related to the current value of the bottleneck estimate:

$$S = \frac{B}{T_{min}} + S_0 \quad (5.9)$$

T_{min} in (5.9) is the minimum measurable interarrival time according to the constraints of any particular operating system, and S_0 is the margin of the packet size value. A standard Linux machine was used for tests where T_{min} was set to 2ms and S_0 to 300 bytes. When packet size dynamics are not used, 1000-byte packets were used.

The whole measurement algorithm is listed in Figure 5.3. Probing is performed in a loop until the data pool is full. For all the histogram and other tests further in this study the pool of 5000 samples is kept at

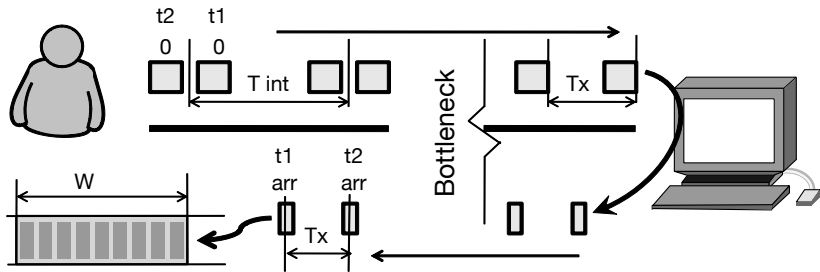


Figure 5.2 Packet-pair end-to-end round trip probing model.

```

1  create a pool
2  for x = 1 to size of pool {
3    send a probe
4    receive UDP ACK packets
5    calculate bottleneck
6    perform change detection
7    if change detected
8      then change probing parameters
9    else do nothing
10   place bandwidth sample in pool
11 }
12 analyze data in pool
13 end

```

Figure 5.3 Procedure of measurement with variable probing parameters.

all times, but it would not always be filled up during the 300 seconds of the measurement period. Clearly, frequent probing may fill the pool at a faster rate, in which case the simulation would exit in the event of the pool overflow. The size of the pool is not the same as the window size, as the latter is smaller and contains only the last 10-40 samples from the pool. Window size has little bearing in histogram analysis as the analysis is performed on all raw samples after the measurement is finished, but is used when online estimation of bottleneck bandwidth is performed, and where both histogram of kernel density analyses are performed using the contents of the window size at any point of time.

Within each loop of the algorithm in Figure 5.3, the sender transmits a packet-pair to the remote host, which generates and sends special UDP ACK packets back to the sender. When the ACK packets are received at the sender, the interval T_x between them is used to calculate the current value of bottleneck B using (5.4). Then, if the change is detected by the recursive change detection function that uses the current sample as well as the contents of the window, probing parameters are set to their lower margins. If the change is not detected, the values are incremented until they reach their upper margin. If the estimate of the bottleneck bandwidth that is created using samples in the window has changed, the packet size is also adjusted in accordance with (5.9). Histogram analysis used later in this section offers the results with only probing interval and window size dynamics, and separately with packet size dynamics included, which is specifically indicated in figures.

5.1.3 Histogram Based Performance Analysis

In view of the problem of multiple modes in measurement data, *histograms* offer the best way to compare data obtained through conventional probing with the proposed variable results. This section offers histogram analysis of data obtained by packet-pairs under light and heavy cross-traffic in the network. Additional tests are performed with a sudden change in the bottleneck value, which, in the real network, would be related to a change in path. As there are some tools that use packet trains [28] [20], which are a case of packet-pair, for the sake of comparison the analysis of behaviour of variable packet trains was also conducted. Bin size for all histograms is constant and set to 100kpbs, which means that 10Mbps-long horizontal axis contains 100 bins.

To be able to properly compare the performance of conventional probing with the proposed variable methods, four subsets are compared:

sparse and frequent static probing, variable without packet size dynamics and variable with packet size dynamics. Probing intervals for sparse and frequent probing are set to lower and upper margins of variable probing, that is 3 seconds and 0.5 seconds correspondingly. The bottleneck used in simulation is set to 1Mbps and is situated in the middle of the measurement path that is 7 hops long.

Light and heavy *utilization rates* in the network are created by outside sources of HTTP, FTP, and multimedia traffic that is routed via a portion of the measurement path. For the simulation analysis a number of realistic models of traffic were used. By using a mixture of realistic traffic types an attempt was made to closely simulate the traffic of an actual network.

Finally, at the end of this section simulation results were verified by a test in the real network within a university campus that proves that the assumptions are valid in practice as well.

Figure 5.4 displays results from probing under light cross-traffic. All histograms support the notion of multimodality by showing a spike very far from the correct estimate.

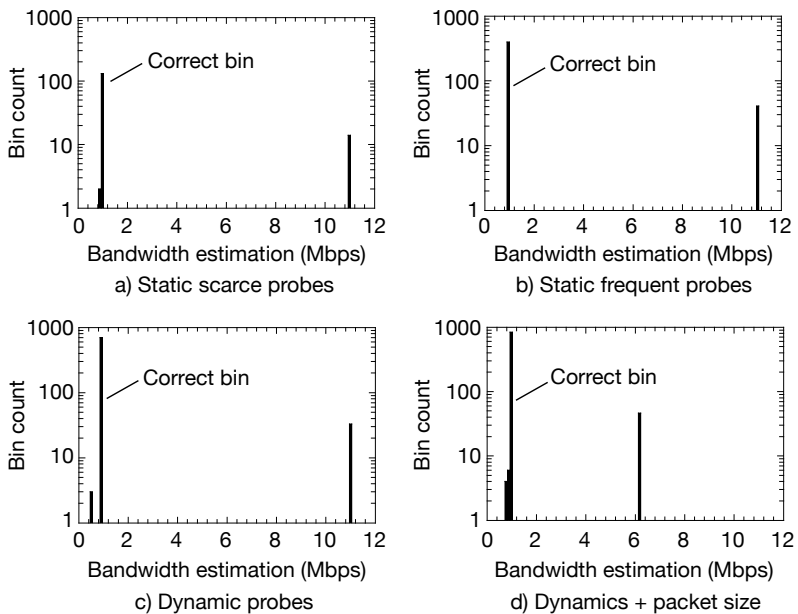


Figure 5.4 Results of static and variable probing with light cross-traffic in the bottleneck.

The malicious bin at *11Mbps* can be explained by cross-traffic *interference*. In fact, cross-traffic packets queued before the packet-pair and, thus, delaying the first packet, are much more common than the packets queued between the probing packets, due to comparatively short interval between probing packets even after the bottleneck. In case of such interference, *11Mbps* bin would closely represent the bandwidth of the rest of the path other than the bottleneck part. In fact, the rest of the path in the simulation study consists of *10Mbps* links.

Since there is not much interference with cross-traffic in this case, all cases display similar performance. Visual comparison of logarithmic vertical axes of Figure 5.4(a), and Figure 5.4(b) makes it clear that more frequent probing does not necessarily improve the *error rate*, as the difference between the main and malicious bins drops from around 10 times in sparse to 5-6 times in frequent probing. Variable probing in Figure 5.4(c) and Figure 5.4(d) indicates yet a wider difference between correct and malicious bins, which proves that dynamic probing has succeeded in suppressing unwanted modes.

Figure 5.5 is the vivid example of the *validity problem* attributed to

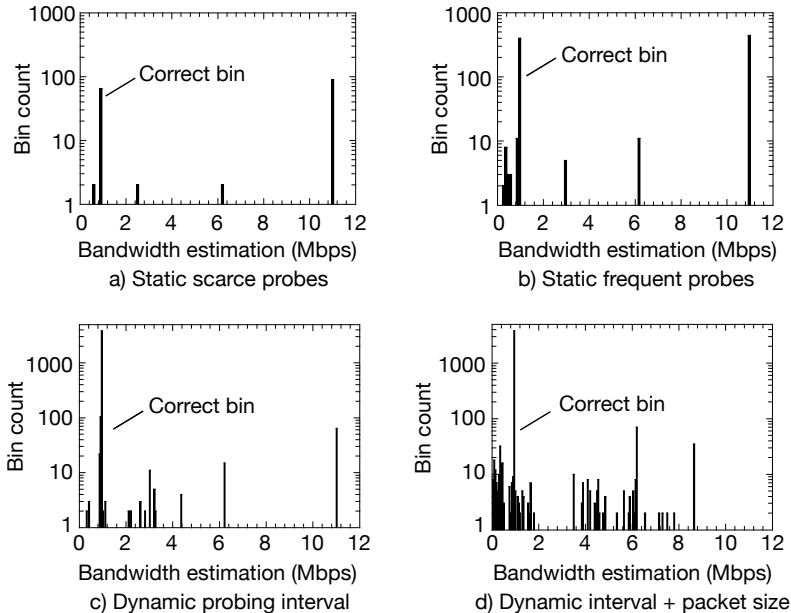


Figure 5.5 Results of static and variable measurements under heavy cross-traffic.

conventional static probing. Under heavy cross-traffic interference conditions, static scarce probes in Figure 5.5(a) result in the wrong outcome of measurement, which in this case would be the malicious bin at *11Mbps*, as the outcome is the largest count among all other bins. Increased frequency of static probing in Figure 5.5(b) does not change that ratio, and the malicious bin is still slightly bigger than the correct one.

It should be noted that the width of all bins is the same, which eliminates the possibility of so-called bin border problem, which is when two close samples are counted in two neighbouring bins. The bin border problem can be visually confirmed by difference in widths of separate bins. None of the histograms in this chapter exhibit the presence of this problem, which means that all values in major bins are mostly grouped tightly around the main value within a bin.

Variable probing in Figure 5.5(c) remains valid even under heavy utilization, as the malicious modes are at most 30-40 times smaller than the main bin. Such a wide gap facilitates correct estimates at all times during the simulation, as there are always several times more correct samples than those from malicious bins. This performance can be explained by the fact that variable probing follows the changes in the traffic and therefore results in less random samples, as would be in the case of static probing, interference of which with the cross-traffic is completely random, and, therefore, extremely multimodal. Performance with *packet size dynamics* included in the set of variable parameters in Figure 5.5(d) performs similarly, offering high validity estimate.

Figure 5.6 and Figure 5.7 offer similar results from simulation with a sudden change in the bottleneck value. Two correct bins are available for selection now, one for the initial bottleneck value set at *1Mbps*, and the other – for the new value of *1.5Mbps*. The outcome is similar to previously displayed simulation with a constant bottleneck. Yet similarly, heavy cross-traffic suppresses the correct bins as in case of static probing, while the opposite is true for variable results.

Many measurement tools use *packet trains* longer than 2 packets, and sometimes longer trains are required, as is the case, for example, with available bandwidth measurements. To display the stability of measurement results with increasing order of packet trains, simulations for 2, 3, 5, and 10 packets in the probe were conducted. Bottleneck bandwidth estimation with trains is made by using the intervals between each pair in train, and in that regard is not any different from a standard packet-pair.

Results for probing bottleneck with static packet trains is displayed in Figure 5.8. Only heavy cross-traffic interference was used for this test. From the figure one can see that static measurement gradually loses focus

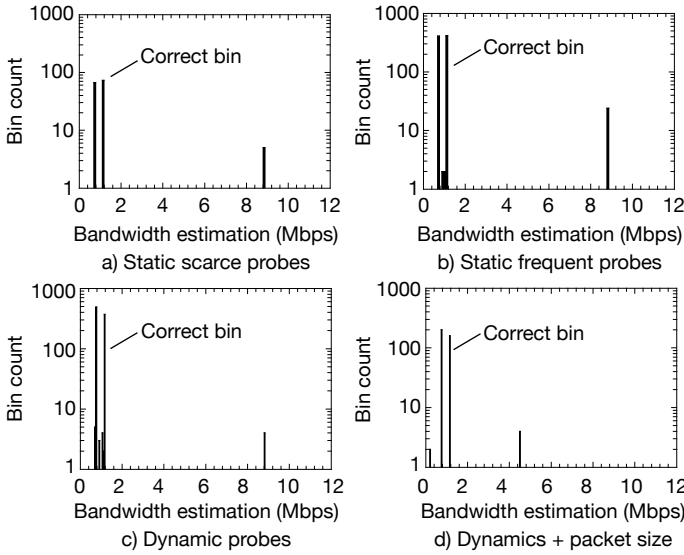


Figure 5.6 Measurements with light cross-traffic in the bottleneck and sudden bottleneck change.

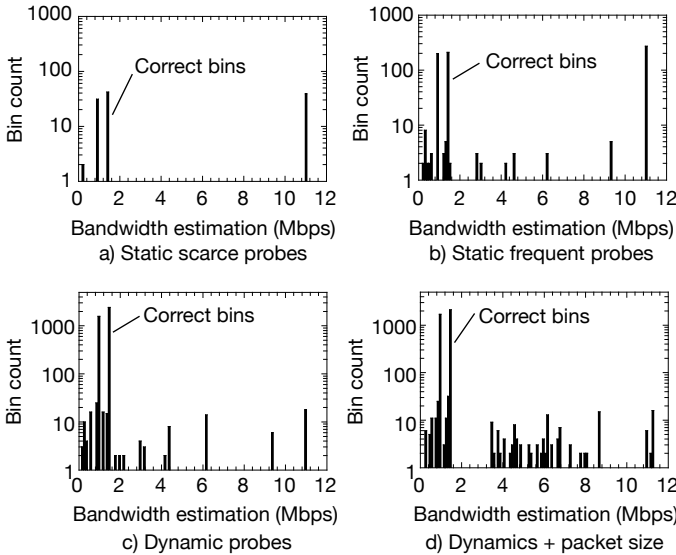


Figure 5.7 Measurements with heavy cross-traffic and sudden change of bottleneck bandwidth.

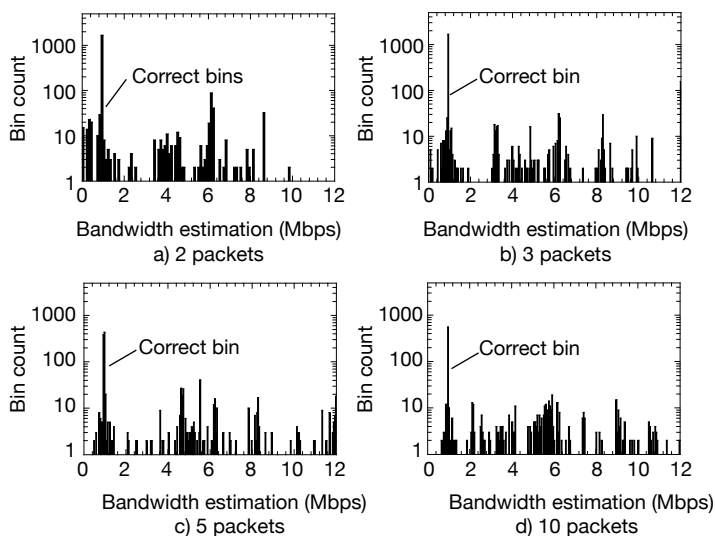


Figure 5.8 Static measurement results with 2, 3, 5 and 10 packets in the probe.

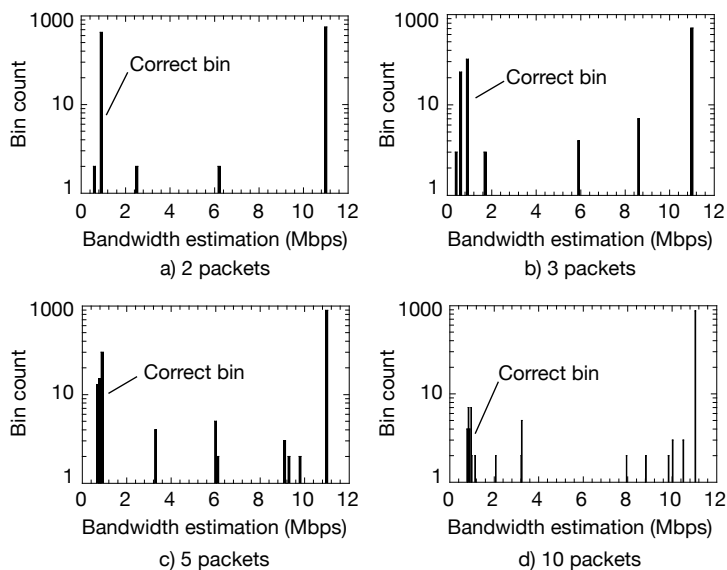


Figure 5.9 Results of variable probing with 2, 3, 5 and 10 packets in the probe.

with the increase in the number of packets in the probe. While 2, 3, and 5 packets in the probe can still produce correct bins of considerable size, the case of 10 packets results in only one malicious bin and almost fully disappeared main bin. As the malicious bin represents the worst case of interference, it remains constant for all lengths of packet trains.

Variable approach to probing in Figure 5.9 displays almost no dependence on the length of the probe, as in all cases the correct bin does not have any rivals among any other modes in the data. However, it is true, that with the increase of the train length, the count of the correct bin becomes smaller, as more samples are becoming incorrect and are distributed on the horizontal axis. This supports the assumption that variable probing should result in increased stability of measurement results.

Although the results supported the fact that the performance of variable method does not change with the increase of train length, it is preferred to use the shortest trains, i.e. pure packet-pairs for measurement as they impose the least possible impact on the network at the time of probing.

5.1.4 Validation Tests

Online estimation of performance metrics is a very important quality of a measurement tool, as it allows uninterrupted operation. Therefore, the tool should aim at producing prompt and correct estimate at any point of time. In this section a separate consideration is given to histograms and kernel density estimation techniques to compare their performance. Since the improvement of the validity of measurement results is the main target of this chapter, the estimation method should be a part of the proposed method.

To compare static and variable methods in respect to producing estimates using histograms, in the simulation an online histogram analysis is performed, which means that a histogram analysis is performed on samples within the window with the arrival of each new sample. To assess the effectiveness of histogram analysis, the certainty rate is tried also, which is calculated as number of samples in the estimate bin to total samples in the window. *Certainty rate* values close to 100% mean that the estimate is the only bin in the pool.

Figure 5.10 exhibits impeccable performance under light cross-traffic. One can see that from the fact that the value of certainty rate never falls below 70% in the left side of Figure 5.10(a) which means that the bin selected from estimate is larger than any other bins in the window. The

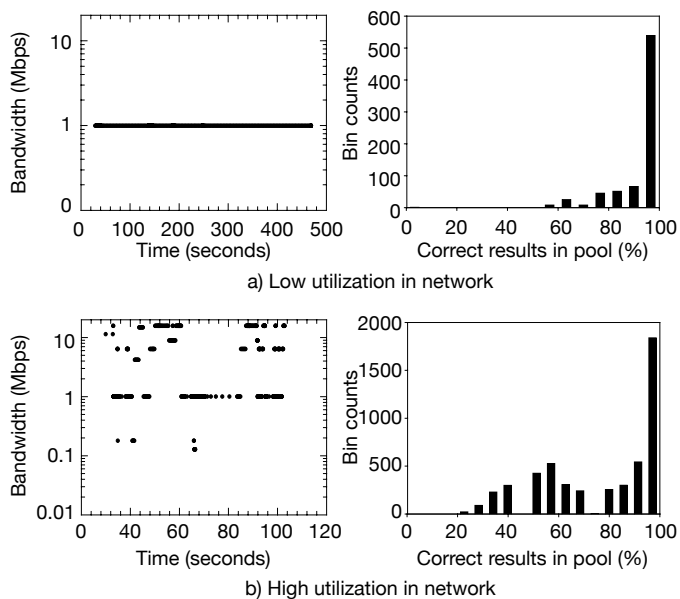


Figure 5.10 Estimation by histogram analysis on data obtained from static measurement.

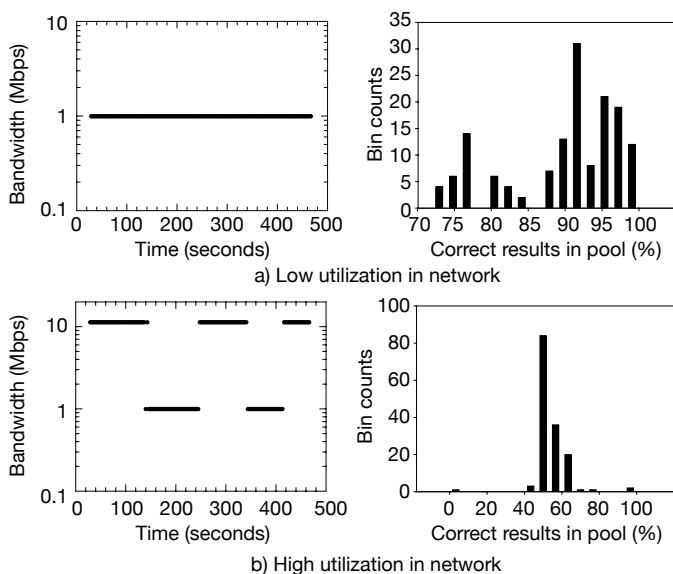


Figure 5.11 Estimation by histogram analysis on data obtained from variable probing.

estimate is at all times correct and shows a straight line at 1Mbps. When the network is highly congested in Figure 5.11(b), histogram analysis performance rapidly deteriorates with the certainty rate of the choice in the vicinity of 50%, which means that the tool has to choose between two bins of almost the same size. This can be confirmed on the right side of Figure 5.10(b), which contains the fluctuations of the estimate between two bins. Another plot that confirms this observation is the fact that the left side of it exhibits mostly results with 50% ratio of correct items, which causes such wide fluctuations. This is clear evidence that histograms do not perform well under high cross-traffic.

On the other hand, histogram-based estimations with variable measurement in Figure 5.11 still hold even under heavy cross-traffic interference. Although with heavy cross-traffic variable probing does not always produce correct estimates, the performance is considerably better compared with static results under heavy cross-traffic. The estimate still fluctuates between correct and incorrect value, but these fluctuations are very short-term, and the majority of estimation points are situated on the straight line at 1Mbps. Histogram plot of correct bins ratio also supports the fact of higher precision attributed to variable probing.

Kernel density is another statistical tool for analyzing data pools. Unlike the histogram analysis, kernel density method does not share the limitations of the bin size, as the estimate in kernel density method is simply a sample with the highest density, i.e. a sample that has the largest number of other samples with closely situated values.

Similarly to histogram analysis, a variable for assessing the performance of *kernel density* is introduced. The kernel density itself, i.e. the density of the sample that is selected as an estimate, is the best indicator of the effectiveness of the estimation. Clearly, the larger the value of kernel density is, the higher is the certainty that the estimate is correct.

Static probing under light cross-traffic in Figure 5.12 performs just as good as its histogram counterpart, but the validity of kernel density method under heavy cross-traffic increased considerably, as there are very few persistent errors in the estimate. However, kernel densities are close to 0 in both cases, which means that the certainty rate of the produced estimate is not high.

Finally, Figure 5.13 displays the results for kernel density processing of data obtained by variable probing. By visual comparison of histogram and kernel density results for variable probing, the increase in validity of kernel density method over histograms with the same set of data can be confirmed as a larger count in the bin with higher value of kernel density. Physically, this means that values that are obtained from variable prob-

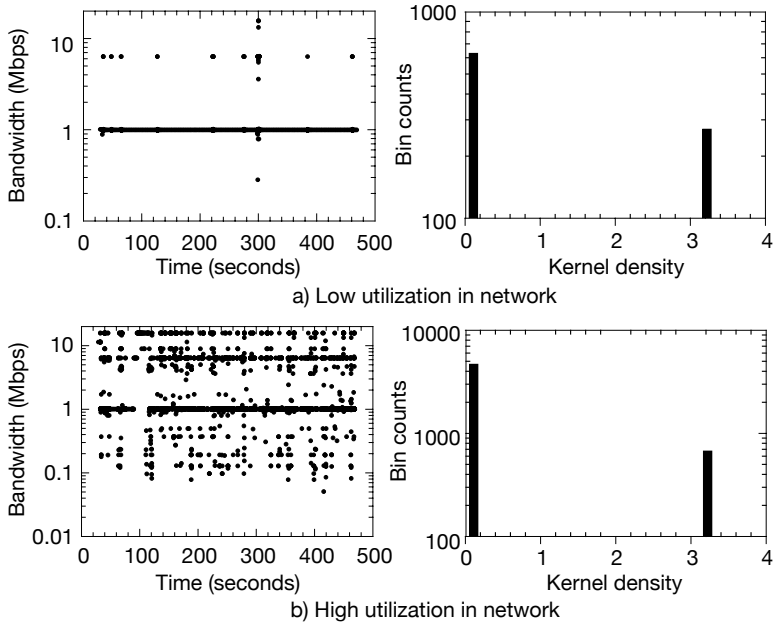


Figure 5.12 Estimation by kernel density function on data obtained from static measurements.

ing tend to cluster in nearby proximity, thus resulting in higher kernel density. Estimates under light cross traffic in Figure 5.13(a) are produced with almost constant high value of kernel density, which means that many samples had relatively similar values. Presence of the same high value of kernel density can be visually confirmed in Figure 5.13(b) at most times. Due to the heavy congestion in the network, the kernel density is more frequently at closer values to 0, but that does not yet mean that the estimate is incorrect, and, in fact, the line at the correct 1Mbps value is thicker than the other lines just below 10Mbps. Most of the time even under heavy cross-traffic, the variable method would issue the correct estimate.

To support the findings of the simulation study, the proposed method in form of a probing tool is implemented and tested in campus network by probing between two separate campuses of the same university.

As the capacity of the university LAN is 100Mbps, with the backbone link of 155Mbps between campuses, an artificial bottleneck on the path was established by connecting the probing host via 10Mbps hub. Both the creation of the probe, its transmission, and calculations upon the receipt

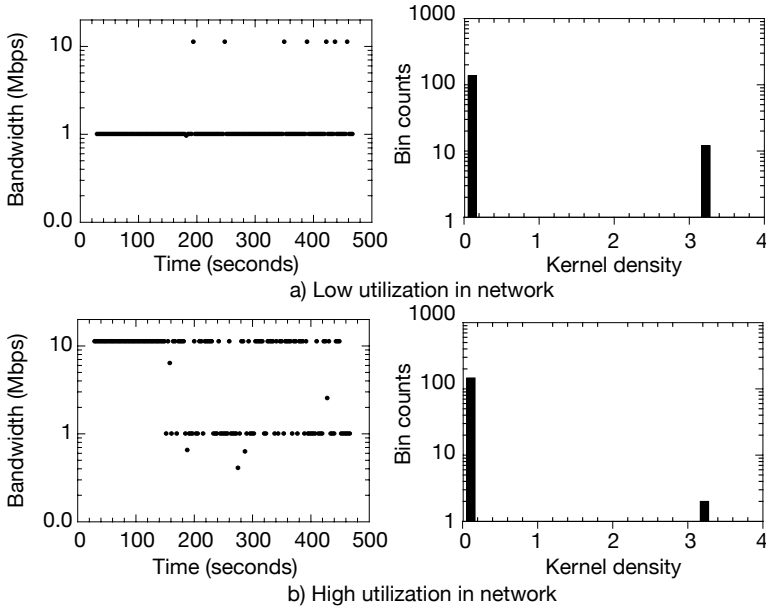


Figure 5.13 Estimation by kernel density function on data obtained by variable probing.

of ACK UDP packets, were performed at the same host, while the probing destination performed a simple task of replying to UDP packets by immediately transmitting UDP ACK packets to the source. This way no synchronization had to be provided between probing source and destination. The measurement source was installed on a fairly high-performance Linux machine with 2.8GHz CPU and 500Mb of RAM, which allowed for performing calculations and the actual probing at the same host. For higher precision control over the transmission of packets a Linux kernel with a real-time patch was used, that allowed for scheduling interrupts in microsecond range.

As each of the three parts of the test in Figure 5.14 was conducted with a different set of settings, each test was performed at a different time. Each test would start at 13.00 Japanese time and collect measurement samples for 90 minutes, after which the histograms would be analyzed in the off-line mode. Sparse and frequent static tests would transmit a probe once every 3 and 1 seconds correspondingly. The same values were set as the upper and lower margins in variable probing. Packet size dynamics were

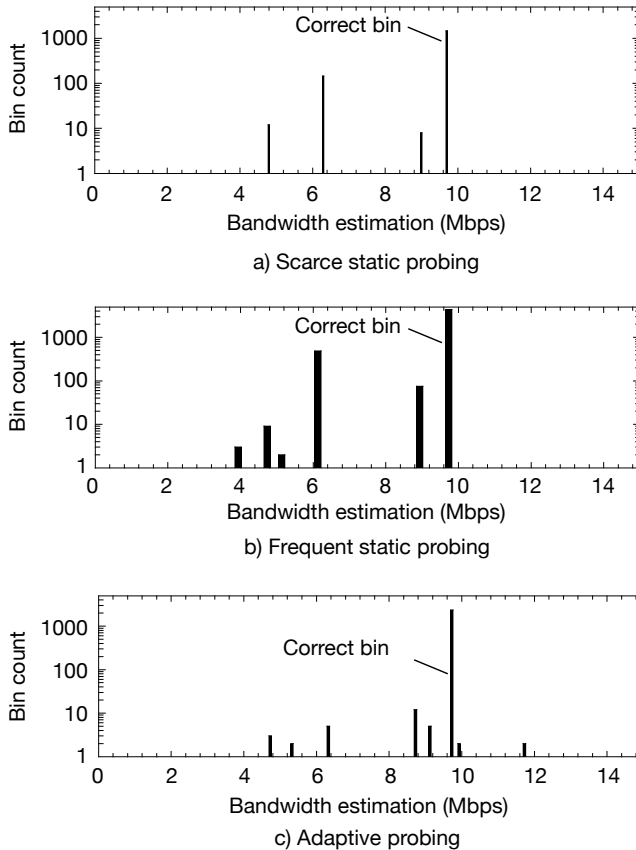


Figure 5.14 Results of test performed in the actual network environment.

not used and its constant value was set to 1500 bytes, which is the MTU of Ethernet.

The results in Figure 5.14 support both the assumption and previously displayed simulation results, as one can see multiple modes in static data, while data obtained by variable probing offers a single bin. Since the backbone of the university network was not nearly as congested as one could generate in a simulation, the case was not witnessed when malicious bin would become taller than the main bin. It has to be mentioned that bin size in these tests was different from the one used in the simulation, and was set to 300kbps, as samples were scattered over a wide range from

cross-traffic interference, and did not produce a clear image at smaller bin sizes. Based on the results in Figure 5.14 one can state that a probing that is based on online feedback about network conditions produces a pool of samples in which the main mode is enhanced while other modes are suppressed, thus resulting in the data that is easier to process by statistical methods. Given the fact that presently complicated methods in processing of network measurement data are eluded because of multiple unknown modes in samples, the proposed method offers a partial solution to this problem by offering “cleaner” data for statistical processing.

5.1.5 Discussion of Measurement Methodology

In this study the issue of validity is argued to be always attributed to measurement tools and derives from the lack of knowledge about distribution of cross-traffic with which probing packets have to interfere as they traverse the network. It is believed that multiple modes in measurement data cannot be filtered out by statistical methods, which is the reason why many currently existing tools use but a very limited statistical apparatus for processing.

Instead of attempting to filter out the unwanted modes in data, it was proposed to create an environment in which the online data about network condition would be used to change probing parameters. This way the anomalies in probing data would find reflection in the way the probing is conducted. It was proved that this filtering approach resulted in enhanced main mode and suppressed unwanted modes. To fulfil this task *abrupt change detection* function was applied, which is well studied and offers very prompt feedback about a change in arbitrary variable. By applying a simple threshold rule to the change detection function it was managed to have an instant feedback of anomaly detection into probing parameters.

Extensive histogram analysis proved the validity of the assumption, as variable probing performed better in all test cases. It was also managed to prove that the connection between network condition and probing parameters successfully attenuates unwanted modes in data. All unwanted modes in the tests were several times smaller for variable cases than for static ones.

The validity of online estimation results was also discussed, which has to deal with smaller number of samples within the window. It was proved that currently widely used histograms perform poorly with measurement data for the reason that “correct” values may be distributed over a number

of neighbouring bins of fixed size. This problem does not exist in another statistical method called kernel density. Kernel density does not care about how scattered the samples are, as it produces the value with the higher density of values around it. This relativity of kernel density is the winning factor, which was proved in online estimation for both static and variable measurement approaches.

This work is ongoing and it is planned to analyze traffic patterns in cross-traffic at the time of probing to correlate them with the measurement outcome. It is believed that such correlation would offer a chance to create a mathematical model of the feedback of network condition back into probing parameters. There is also the plan to support the mathematical proof by extensive tests in real network environment.

5.2 Available Bandwidth Measurement

End-to-end available bandwidth of a network path is the unused portion of its capacity. Knowing the end-to-end available bandwidth is important and beneficial for various network applications such as multimedia streaming, server selection and optimal routing. End-to-end available bandwidth is usually estimated with active probing techniques which send probe traffic from the source host to the destination host of the path. Active probing is free of privileged network access and thus feasible for the end users.

Several active probing techniques have been proposed in recent years. Generally they can be classified into two models according to the measuring object: the *rate-based* model directly measures the transmission rate of the probe stream to match the available bandwidth; the *gap-based* model measures the changes in probe *packet gap* between the sender and the receiver to calculate the cross-traffic rate and then estimates the available bandwidth indirectly. Pathload [33], PTR [30] and pathChirp [43] are examples of rate-based model, while IGI [30] and Spruce [44] are examples of gap-based model. In addition, they can also be distinguished by the pattern of probe streams. Pathload, PTR and IGI use packet trains with even spaces between packets in the train. On the other hand, pathChirp and Spruce use packet trains or sequences of packet pairs with uneven spaces.

Different categories have different advantages and disadvantages. The gap-based model assumes that the narrow link (the link with the least capacity) and the tight link (the link with the least available bandwidth) are identical, which is not necessarily the case. The rate-based model is free

of this problem. The uneven probing pattern, as in *pathChirp*, is quick to discover the turning point within the stream. But its estimate relies on a single packet at the turning point and consequently not very precise or reliable. In contrast, the even probing pattern performs more reliable estimation based on a packet train. But it is slow to converge at the turning point.

5.2.1 Probing Method

In this section a new end-to-end available bandwidth measurement was introduced and called *ABshoot*. It is based on rate model.

Current techniques use either evenly or unevenly spaced *probe streams*, but as to the knowledge, no tool yet has been proposed to use both methods in one tool. Considering different properties of stream patterns, it is believed that by combining the two patterns in one tool, one can potentially improve the performance. *ABshoot* adopts both types of streams. First a single unevenly spaced *probe stream* is used to quickly identify the *turning point* within the stream. Evenly spaced streams are then used to perform a thorough search in the proximity of the turning point so that one can accurately estimate the available bandwidth. *ABshoot* works the two stages for one measurement: zoom and focus.

Generally speaking, rate-based techniques do not need to know the *end-to-end capacity*. However, knowing the end-to-end capacity can help us to efficiently construct the unevenly spaced probe stream, since capacity is the upper bound of available bandwidth. The end-to-end capacity can be estimated with existing probing techniques, as it is done in IGI.

The unevenly spaced stream is built based on the end-to-end capacity.

The per-packet probing rate within the stream starts with a low value and increases linearly until it reaches the capacity. The linear growth of probing rate enables the stream to cover the full range of possible available bandwidth values with uniformly distributed intensity.

Figure 5.15 displays an example of the unevenly spaced probe stream. *PathChirp* also uses unevenly spaced probe streams but it is different from the pattern. It uses a chirp pattern where packets are exponentially spaced. The probing rate increases rapidly within the chirp. However a chirp samples the low rates more frequently than the high rates. For example, if a chirp consists of 15 packets covering the probing rate from 1 Mbps to 12 Mbps and the spread factor is 1.2, the probing rate of first 7 packets are all lower than 3 Mbps, while the last 5 packets correspond to the range from 5 Mbps to 12 Mbps. When there are more packets in a

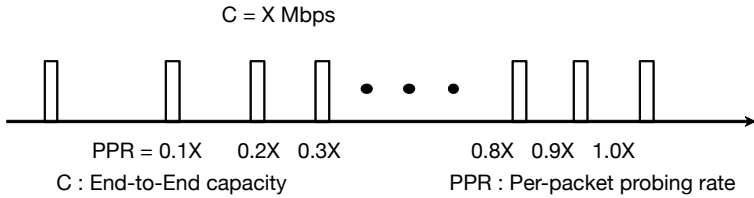


Figure 5.15 The unevenly spaced probe stream.

chirp and the range of probing rates is broader, the low probing rate focus becomes more apparent.

In the unevenly spaced stream of *ABshoot*, the probing rates are uniformly distributed so that the probe can be more efficient. Moreover, the knowledge of capacity helps us build the stream flexibly: when the capacity is high, more packets are needed for one stream; oppositely fewer packets are enough for a path with low capacity.

There are two stages of the probing procedure in *ABshoot*, as it is shown in Figure 5.16. The first is zoom stage and the second is focus stage.

At the zoom stage, one single probe stream with uneven interval is used. It is introduced in the last section. This unevenly spaced stream is aimed to discover the turning point within itself quickly. Here the turning point is defined as the point at which the OWD (*one-way delay*) of probe packet exhibit an increasing trend. To see the OWD difference between consecutive probe packets, one can simply compare the *time gap* between them at the sender with that at the receiver, which avoids the clock synchronization issue between two hosts. The turning point should be the packet from which the OWD of following packets increases gradually until the end of the stream.

Since the probing rate starts to be higher than available bandwidth from the turning point, the actual available bandwidth is possible to lie around the probing rate of the turning point packet. However, estimation based on a single packet is not reliable. The focus stage of *ABshoot* is intended to solve this problem.

At the focus stage, *ABshoot* uses a series of evenly spaced packet trains to refine the turning point of the zoom stage. Here, one packet train consists of a number of packets with the same probing rate. It is more reliable to estimate the available bandwidth with those packet trains than the stream used at the zoom stage.

The first *packet train* at this stage is with the probing rate of the turning point. It is the second stream in Figure 5.16. Then the sender adjusts

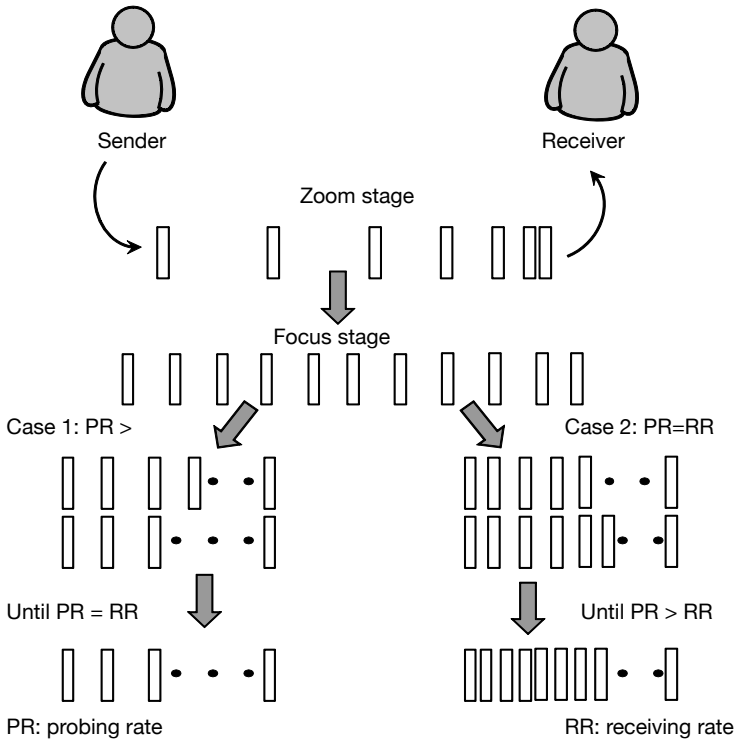


Figure 5.16 The probing procedure of ABshoot.

the probing rate according to the feedback of the receiver. There are two cases for adjusting the probing rate, as it is shown in Figure 5.16. The first one is that the probing rate is higher than the receiving rate. It happens when the probing rate is higher than the available bandwidth so the probe packets are congested within the path. The sender will lower the probing rate of future trains in this case; in contrast, the second case is when the probing rate equals to the receiving rate. It happens when the probing rate is lower than the available bandwidth. In this case the sender will increase the probing rate.

With the confidence that the actual available bandwidth is in close proximity to the turning point found at the first stage, the sender can perform linear search with smaller resolution in order to estimate the available bandwidth with high accuracy. For example, the resolution could be

a slight portion of the end-to-end capacity. For case 1, the sender will keep lowering the probing rate until it becomes equal to the receiving rate. The final estimate result is the probing rate of the last train. On the other hand, in case 2 the sender keeps increasing the probing rate until it becomes higher than the receiving rate. The final estimate is the probing rate of the second last train.

Like other available bandwidth measurement techniques, all parameters in *ABshoot* are tunable, for example, the length of the unevenly spaced streams and evenly spaced trains, the resolution of probing rate between each packet in the uneven stream and between each train. These parameters should be tuned according to the end-to-end capacity.

In order to further improve the efficiency of the probe, *ABshoot* also takes advantage of previous estimates of available bandwidth values. Since the Internet usually behaves with long-term stability [49], the available bandwidth of an Internet path is likely to hold within a limited range for a certain interval. As a result, *ABshoot* simplifies the subsequent measurements: it starts focus stage directly with the available bandwidth value of previous measurement. The time for zoom stage is saved in this case. If too many trains are dispatched but the focus stage does not end, the measurement from zoom stage is aborted as the normal process. This could happen when the actual available bandwidth changes abruptly. The average number of streams will be discussed for one measurement of *ABshoot* later.

5.2.2 Method Evaluation

The evaluation is based a simulation model. In simulations one can control the cross-traffic precisely and scrutinize the actual available bandwidth values closely. It is crucial for the evaluation. The performance of *ABshoot*, *pathChirp* and *IGI* are compared. *PathChirp* and *IGI* are selected to represent the *rate-based* and *gap-based* model respectively. With practical purpose, the tools are evaluated and compared on a path with capacity of 10Mbps. It is because in today's Internet, most paths are with a capacity around 10Mbps, usually constrained by the edge of the network. A tool which could run well in such environment has high practical value. The evaluation configuration is shown in Figure 5.17. The network path under observation is simply with single bottleneck in the middle of the path. Admittedly, this topology is primitive and some special artefacts such as multiple bottlenecks are beyond the scope of the consideration, but the

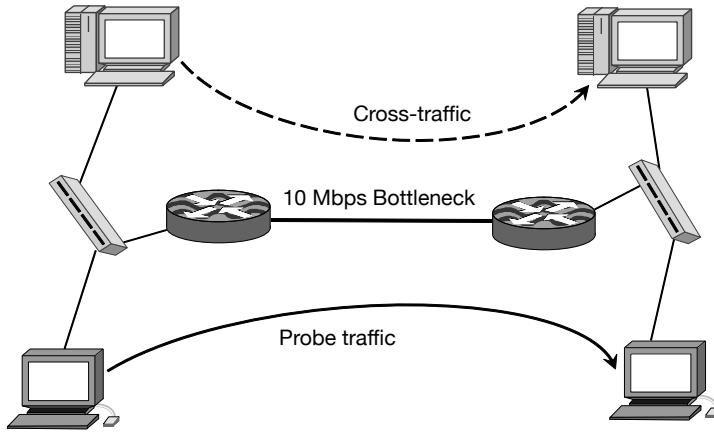


Figure 5.17 Evaluation configuration.

simple topology is favourable for discovering the fundamental characteristics of the tools. The cross-traffic is generated in the form of various popular Internet applications.

All three tools have tuneable parameters. Generally the default values of those parameters are used. The initial gap of *IGI* is exceptional, because the utilization of the path is high for a long interval and the default initial gap is too large. For *IGI*, the probe packet size is 700 bytes, the initial gap is 0.5 ms and the length of a train is 32. For *pathChirp*, the probe packet size is 1000 bytes, the exponential spread factor is 1.2 and the length of the chirp is 15. For *ABshoot* the probe packet size is 1000 bytes, the length of the unevenly spaced stream is 15 and that of the evenly spaced streams is 20.

Since the measurement periods for each tool are different, each tool is run at fixed intervals and then take the average value of all estimates as one result within the interval. In simulation results below, the interval is set to 1 second. It is believed it is easier to observe the characteristics of each tool within a short interval. Those characteristics may be masked if the estimates are averaged over a long time. Within 1 second interval, *IGI* can finish 2 measurements while *pathChirp* and *ABshoot* produce 3 estimates each.

The simulation results are shown in Figure 5.18. The background cross-traffic is light at the beginning of simulation, increases in the middle and reduces again in the end of the test. As one can see, *pathChirp* and *IGI* constantly overestimate the available bandwidth. *PathChirp* is able to roughly show the trend of the change of cross-traffic, but it frequently

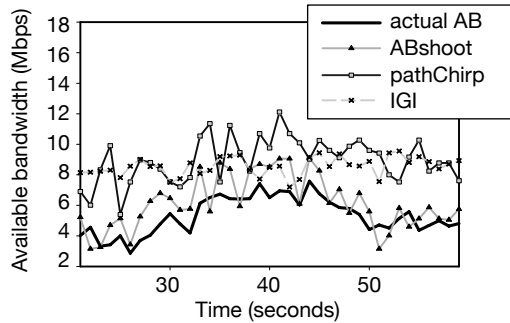


Figure 5.18 Simulation results.

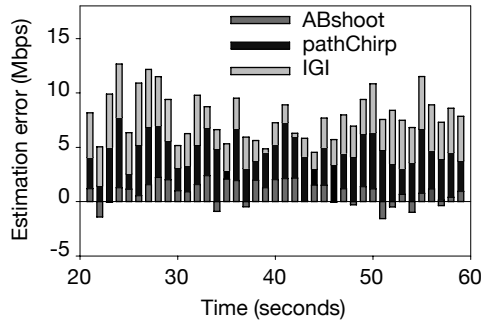


Figure 5.19 Comparison of errors in AB estimation among the three tools.

yields estimates higher than the capacity. *IGI* is generally unresponsive to that change. *ABshoot* gives estimates consistently close to the actual values. The estimates are most of time within a 10% relative error. *ABshoot* is the most reliable among the three.

Figure 5.19 offers a more rigid view of the comparative performance of the proposed tool against *pathChirp* and *IGI*. It is clear from the error plot that *ABshoot* oscillates between minor over- and underestimations, while the two other tools are constantly overestimating the available bandwidth, with *IGI* being the worst of the two. The physical explanation of the oscillation process is that *ABshoot* is constantly converging to the correct value of available bandwidth, but due to its rapidly changing nature, always fails to converge precisely. However, this physical property is intrinsic

in all available bandwidth measurements, and cannot be solved by definition. Tools, however, can be compared based on the error in realtime estimation results, as is proved by Figure 5.19.

5.2.3 Discussion of Measurement Methodology

The amount of overhead traffic is an important property of available bandwidth measurement tool. According to the study of *Spruce*, *pathload* could generate between 2.5 and 10MB of probe traffic per measurement. Less traffic is generated by *IGI* and *Spruce*, with average 130KB and 300KB respectively.

How much probe traffic does *ABshoot* generate for one measurement? This is determined by the number of trains generated at the focus stage, as there is only one stream at the zoom stage. Based on simulation results, the average number of trains is 3. As a result, the probe traffic for one measurement is less than 100KB.

ABshoot is a *rate-based* tool that uses the information of capacity. Unlike *gap-based* tools, errors in capacity estimation will not affect the final result drastically, but rather lead to a long process of measurement. Besides, it is not difficult to detect those errors when many long measurements happen.

In this section, *ABshoot* was introduced, and is a reliable and efficient scheme for end-to-end available bandwidth measurement. It takes advantage of the knowledge of end-to-end capacity to construct the probe streams. Both unevenly spaced and evenly spaced probe streams are used. The former is used at the zoom stage and the latter at focus stage. These unique characteristics give a considerable boost to *ABshoot* performance. *ABshoot* can also adapt to the past record of measurement results to further improve efficiency. *ABshoot* is evaluated and compared with some existing tools in simulations. The results show that *ABshoot* estimates the available bandwidth with high reliability and efficiency.

5.3 Lightweight Jitter Estimation

The variation in bandwidth utilization is an important performance metric of packet networks. It indicates how rapidly and at what degree the traffic is changing and reflects the occupied bandwidth of the network. When other parameters remain unchanged, the variation in bandwidth utiliza-

tion directly affects the latency variation of transmitted packets. In this study, such variations are referred to as *jitter*.

Some applications are interested in learning the variation in bandwidth utilization jitter. For example, video conferencing relies on the metric when dealing with video quality. Basically, all of QoS-related networking technologies are extremely sensitive to *latency* deviations.

In IP-based communication, packets are not guaranteed in a particular order and throughput. As a result, a packet may be transmitted with minimal delay while the following packet may take longer to transmit. Significant delay variation can interrupt the video quality because audio and video signal must be reconstructed from all the received packets in a continuous stream.

Correct estimation of delay variation may require considerable calculation and, therefore, much time and CPU resources, which is not acceptable for real-time applications, as they demand the awareness of the traffic change and should be able to respond immediately. A lightweight method, on the other hand, should define only lightweight calculations.

In this section a *lightweight* model is proposed that allows estimating the *variation* in bandwidth utilization near to real-time. In order to guarantee the speed and simplicity, a special structure of the probe is introduced using a number of different *packet sizes*. Outcome of the method is discrete, with precision depending on the number of packets and granularity of size distribution in the probe. It is believed that results of the present research can be used by applications that need fast outcome and are willing to pay for it with the precision.

The following parts of this section present detailed description of probing method, which was tested in simulation environment later in this section.

5.3.1 Estimation Model

In the traditional *packet-pair dispersion* methods [27] [40], it is known that if the packet size of packet pair is large, the cross traffic (CT) is easily inserted in between the packets of a pair. The larger the packet size, the higher is the likelihood that the probe will suffer from cross-traffic interference. Therefore, one can readily state that probes with larger packet size are more susceptible to *cross traffic*.

In the method, standard probing pairs are used that are transmitted end-to-end. But unlike the traditional packet-pair methods [27], which are



Figure 5.20 Variable sensitivity within the probe.

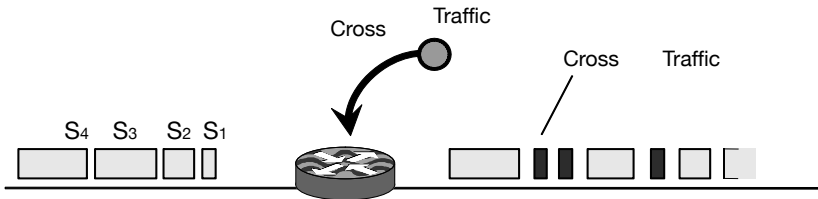


Figure 5.21 Model of interference with cross-traffic.

entitled to use the same packet size, a series of packets are used in gradually increasing size, without identical packet size of any two packets. It is believed that such structure of a probe covers a broad range of sensitivity as shown in Figure 5.20. In the figure there are four packets in increasing size. As it was discussed earlier the sensitivity to CT increases in the order of packet size. At the same time, since packets of various size are subject to cross-traffic interference simultaneously, by comparing the response of various parts of the probe, one can estimate the sensitivity level that responded to the cross traffic with the highest change. Figure 5.21 illustrates the proposed method.

The model operates as follows. For each consecutive pair of packets in the probe, a pool to store arriving samples is prepared. At the arrival of each probe, all of the pools are filled with a new sample, which is defined as a simple inter-arrival interval. The 10 most recent samples are stored at all times.

Also at each new probe arrival, after the new sample is inserted in its corresponding pool, the variance of each pool is calculated. The variance should indicate how drastic the changes are in cross-traffic, as reflected in inter-arrival time.

Final algorithm is a simple selection of final result with the highest value of the variance. For example, if one used five packets in the probe, one would prepare four slots to store arrival samples. With each new

probe, one would get four new variable values, which one would use to find the maximum difference from the previous set of values. The maximum value is declared as the current response maximum. Having found the maximum, it is identified which pair in the probe resulted in the maximum value, and finally release that as the result. As it was mentioned in the beginning, the main purpose of the proposed method is not the exact value for property, but simply the position, which is the sequence number of the packet pair in the probe that offers the highest response.

Because of the discrete nature of the proposed method the granularity of the result depends on the number of packets in the train and the size of the gap in packet sizes among them. Naturally, smaller gaps and more packets in the train would result in higher precision of the proposed discrete estimation.

5.3.2 Mining Examples

To evaluate the proposed model a simulation model was used. Additionally, various traffic patterns were used at the application layer. Because of the nature of the proposal, it is essential to verify its performance in highly controlled simulated environment, and only after such verification should it be possible to test it in the real networks.

The packet train with five packets in it was used, starting with the smallest packet of 100 bytes and fixed gap of 300 bytes. To be able to test the method under a set of various conditions, a large number of traffic patterns was defined, such as HTTP, FTP, e-mail and video conference, to be used simultaneously and provide heavy fluctuations in utilization. Figure 5.22 offers the aggregated results for three levels of cross-traffic volumes in the network.

As one can see in Figure 5.22, response cases, which are simply the sequence numbers of packet-pairs in the probe, are different for different levels of utilization fluctuations. Figure 5.22(a) and (b) display the results for low utilization of the network. Figure 5.22(a) displays the traffic conditions: the number of *utilization variation* of different ratio.

It shows most utilization variation is near 0% which means there are no significant changes in the cross traffic. Figure 5.22(b) shows the result of the proposed model. In that figure case 2 means the inter-arrival time between the third packet and the fourth varies the most among all packet pairs. The result matches the traffic condition in that only case 2 reflects the slight traffic changes. Please note that the number of final results may be less than the number of samples. This is because a result

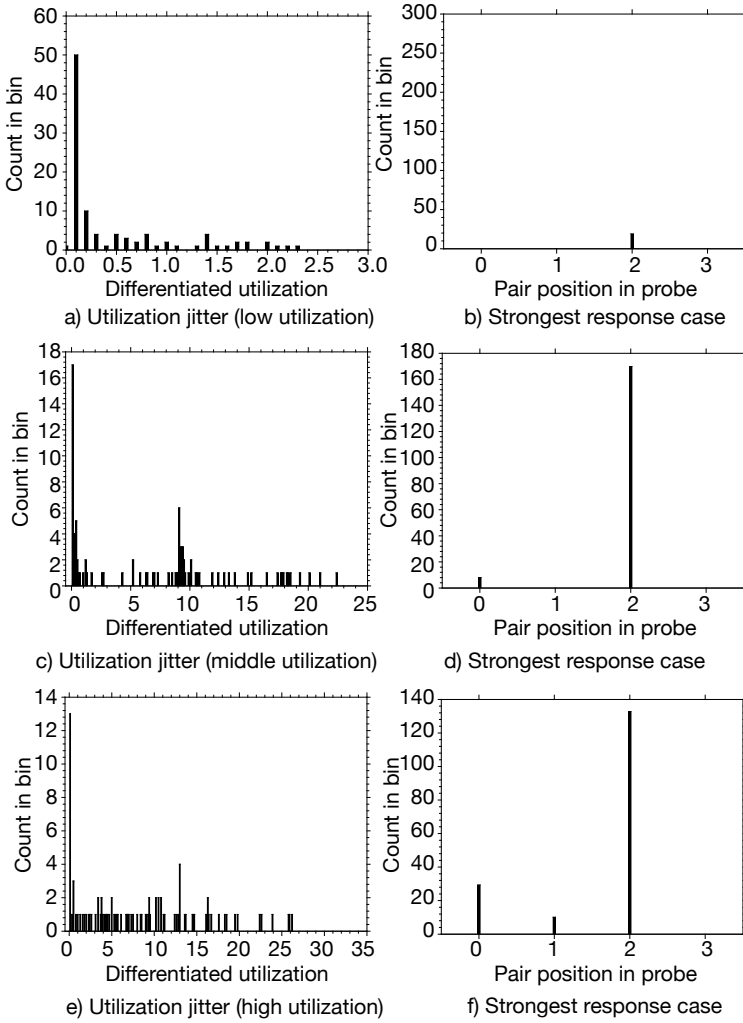


Figure 5.22 Sensitivity to CT depending on packet size.

from certain sample may be too little to regard that it reflects the traffic change. Only the result of a sample greater than a threshold can be one of the final results.

The plot with medium utilization somewhat changes. Figure 5.22(c) exhibits a small spike and a high density area around 10%. This small spike can be found in Figure 5.22(d), which is represented by the case 0. As the majority of bins in Figure 5.22(c) are clustering around near zero values, the case 2 in Figure 5.22(d) is considerably larger than the case 0, which stands for correct performance.

Finally, a more complicated traffic distribution in Figure 5.22(e) allows to visually detect three main areas of concentration, – around 10-12 and much smaller around 5. Verification of this traffic pattern in the proposed cases resulted in Figure 5.22(f), which, truly, shows a fairly adherent behaviour. Thus, the bins between 10 and 15 in Figure 5.22(e) are represented at case 0, similarly to the middle utilization case, while smaller spikes at 5-7 are represented in the discrete model at case 1.

5.3.3 Discussion of Measurement Methodology

In this study, a model to estimate utilization variation of an arbitrary network path was proposed. The model is aimed to discover the extent to which the bandwidth utilization is changing. Results of the proposed method are easy to calculate. Along with the overall simplicity of the model is the swiftness of giving the result and the light-weight probing traffic it inserts into the network. And it also offers flexibility with a set of configurable parameters. By the number of tests performed in a controlled simulated environment, it was proved that the proposed case-based method adequately responds to changes in the traffic conditions in real-time.

Chapter 6



Active Measurement Boxes

First of all, some terminology elaborations are necessary. What this chapter refers to as a *box* is not particularly a device. There will be some cases when the term is used literally but otherwise this chapter addresses finished products based on active measurement methods presented earlier in this both as well as those developed in active measurement research community.

Active measurement is a relatively new field. As was presented earlier, IPPM RFCs became solid in 1999 but serious research in the area did not start until a couple of years later. Today, there are so many authors working in the area and so many papers written on the subject that it is difficult to keep track on the current state of the research field as a whole.

Even in 2003 there were several different methods and some research papers like in [27] were written as summaries of all related research at the time. Research paper in [30] is a similar achievement.

Those summaries/reviews did not however indicate that the research in the area stalled. In fact, several issues still remained at the time, such as *precision* of active measurement results, and *inference* of more application-like performance metrics from probing results. Some issues remain open and even untangled until today and so that this does not turn into a mystery the last chapter of this book will specifically look into active measurements *in context*, where all these open issues will be considered and connected to the beginning of the book, i.e. the NGN standardization process.

Given all this activity it is a wonder why there are so few tools developed to the day. As will be shown in this chapter, there are only a handful of tools that have been propagated through research papers in the first place but later gained popularity after having been released to the public domain as finished tools. Such tools are also called “boxes” in this chapter since the tools are finished.

In fact, to be a “box”, a tool does not have to be a device. Since in active measurement battlefield the smallest bullet is a packet (normally a UDP packet), there is nothing wrong with using conventional operating systems’ UDP stack to send probes on end-to-end network paths. Actually, the above statement is partially untrue as there are issues with using conventional operating systems to send probes. The fact of the matter is that a “normal” operating system offers a limited granularity of time to the user space. For instance, default installation of a Linux operating system could only guarantee 100ms precision. Simple calculation will show that this will impose the upper margin of $1400 * 8 / 0.1 = 112\text{kbps}$ on measurable bandwidth, where 1400 is slightly below the MTU (maximum transfer unit) of Ethernet, 8 is conversion from bytes to bits, and 0.1 is 100ms in seconds, thus resulting in maximum achievable transmission rates.

Two independent areas of measurement research suffer from the above limitation of default OS installations, – active probing and traffic generation. The reason for the former has just been explained, while the latter is affected in the same way by not being able to transmit more than OS’s time granularity will allow.

Of course, this sounds too grim for the present state of technology, but it does not make it any less true. There are, however, ways of working around this problem by either using special operating systems (*realtime* OS) or tweaking your software in such a way that it would either stay permanently in the kernel space or would not return control OS until the entire probe has been transmitted. The latter method is a very dangerous path especially if yours is not the only process running in the operating system at the time the probe is being sent.

With the advent of NGN, active measurement tools are being more in demand than ever. This chapter and the one after will attempt to cover the entire area of this demand both from the viewpoint of currently available tools as well as by trying to fill in the gaps in demand that are not yet filled by supply.

Here, a small explanation about NGN in question is due. The NGN that is presented in this book through its standardization process is not exactly the NGN that is being deployed at the time this book is being written. In some places in global network NGN is said to have already been

deployed and actively exploited. For the most part this is untrue. When a fully fledged NGN emerges, it should have the following components in place:

1. *Application servers* as defined in current NGN standardization process, – those are not to route conventional traffic but to regulate among multiple end-to-end application paths.
2. Mobility support should be fully in place. In short, you are still connected to the network no matter where you are connecting from.
3. *Convergence* with cellular networks should follow the merger between fixed and mobile networks, application servers are to take care of former cellular traffic, traditional telephone lines, TV, etc.; this is still at the stage of work in progress in many NGN working groups.

While active measurements may still not be in a very high demand today, in the NGN as per the above specifications, active measurements will be an integral part of the process. The best evidence of this is the working groups and standards developed by NGN standardization process as this book is being written.

6.1 Standards, Tools, and Projects

Although SNMP is the oldest performance monitoring technology in the global network, NetFlow and *IPPM* appeared relatively at the same time as per Figure 6.1. First tools based on IPPM were developed almost immediately after the standard has been released as RFC [46]. First tools like nettimer and pathchar, however, did not entirely base on the IPPM standard since the one either did not yet exist or has just been released. This was the time when active measurement starting asserting itself as an autonomy within the world of network management.

6.1.1 Timeline of Standards and Tools

In fact, Paxson who wrote the first IPPM RFC on Framework for IP Performance Metrics [46] also wrote his dissertation a few months prior to that. Naturally, his doctoral dissertation was on the same topic as the descendant RFC.

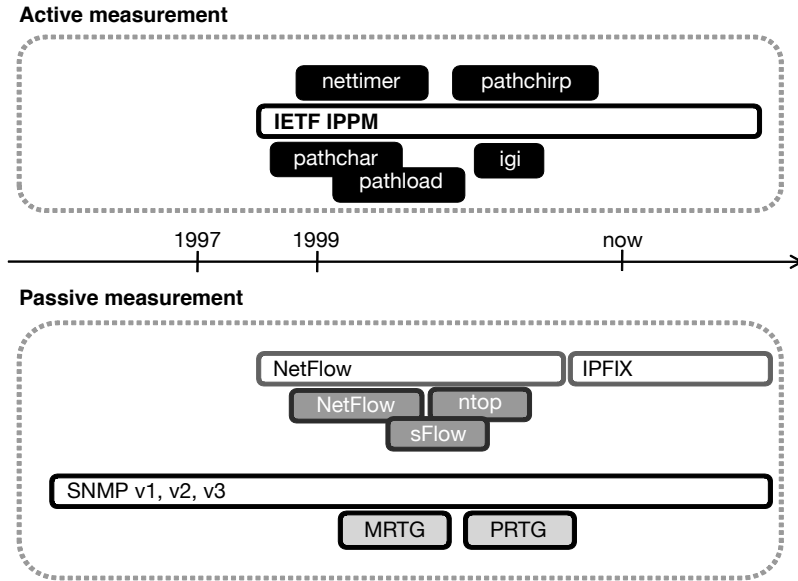


Figure 6.1 Timeline of the two main passive and the only active measurement standard along with accompanying set of tools developed using the relevant standards.

Because during the early years of active measurement research field research was happening at the same time with standardization in the area, there was a minor confusion in terminology. By the year 2002-2003, however, this confusion has evaporated and IPPM terminology took total control.

From Figure 6.1 it may appear as though there are more tools in active measurement than in both SNMP and NetFlow, but this is not the case. SNMP and NetFlow tools if listed in their totality would never fit on one page, while the list in the active measurement block is close to exhaustive.

None of the standards are expected to be discontinued any time soon. The only exception is NetFlow which has matured to the full Internet Standard called *IPFIX*. The core of IPFIX, however, is the good old NetFlow.

6.1.2 Tools and Performance Metrics

Now, let us forget about passive network monitoring and concentrate entirely on active measurement “boxes”. It so happens that each individual research work in the past would target a specific *performance metric* from the IPPM list in [46]. Although that list of much longer, than 3 metrics, practical research seems to be mostly interested in the three from Figure 6.2, – bulk transfer capacity (in IPPM terminology, otherwise known as bottleneck capacity), available bandwidth, and RTT.

If you cross-reference the timeline in Figure 6.1 with the list in Figure 6.2 you should realize that older tools are more willing to target available bandwidth than bulk transfer capacity. In addition, there is another major split that was explained in detail earlier in this book, – single packet techniques versus packet pair.

Because it took several years for the active measurement research to find its proper niche, tools were gradually developing from single packet measurers of bulk transfer capacity to packet pair measurers of available bandwidth. This image should be clear if several tools are considered together over a several years along the timeline in Figure 6.2.

You should note that it does not necessarily mean that bulk transfer capacity can be measured only with one packet probes and such. In active measurements there is great versatility as to what probes can be used to measure what metric. This also was explained very meticulously at the beginning of this book but probably deserves a little comeback in this chapter.

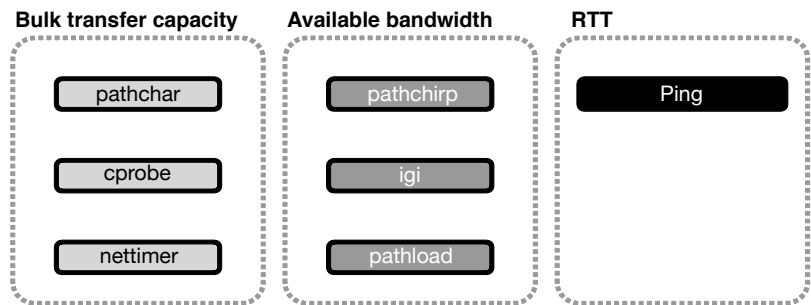


Figure 6.2 Division of most prominent active measurement tools based on targeted IPPM performance metric.

As was already mentioned, active measurement is about two things, – *probe design* and *probing method*. In both, various authors have taken part in semi-playful inventions of interesting probe structures targeting certain qualities of end-to-end or hop-by-hop network performance. Probing method was paid slightly less attention but still remains an active part of the research in the area.

Finally, the ping tool in Figure 6.2 is all by itself in its own category of RTT (Round Trip Time) measurements. To be completely honest, there is very little research in this area and ping tool itself is not really a research material. Quite simply, someone back at the beginning of the Internet needed to know how long it would take a packet to travel back and forth between ends of a network path. Even simpler reason than that was the need to test connectivity. In fact, *connectivity* is a member of the list of IPPM metrics and is measurable, where ping is the tool to do it. End to end delay in this case was easy to obtain given that there is no way for you to verify connectivity unless the packet is *acked* back to you by the opposite end of the path in question, and once this happens it is easy to obtain RTT by subtracting the arrival time from the departure time.

So, end-to-end RTT is more of a side product than a glorious accomplishment of a tedious active measurement research. This is the main reason it is not used much in active measurement research. It is used in many other research areas, most often in statistical explorations of network delays. Although it is within the realm of network performance, it is not an active measurement research per se.

6.1.3 Major Measurement Projects

If you look at the same three metrics from the viewpoint of large-scale projects you get a completely different picture as per Figure 6.3. Here, bulk transfer capacity does not seem very popular with ongoing projects while RTT on the opposite is covered by all three of them. Available bandwidth is covered by two large projects.

Now, in part, the lack of attention towards bulk transfer capacity is sort of justified. First of all, bulk transfer capacity is much easier to measure compared to the overall complexity of available bandwidth measurements. Secondly, in many practical cases measurement projects are conducted by network administrators which means that bulk transfer capacity is a given and there is no need to measure it specifically. In the end, there are many more bulk transfer capacity measurement tools than those targeting any other performance metric, so there is always a quick solution, especially

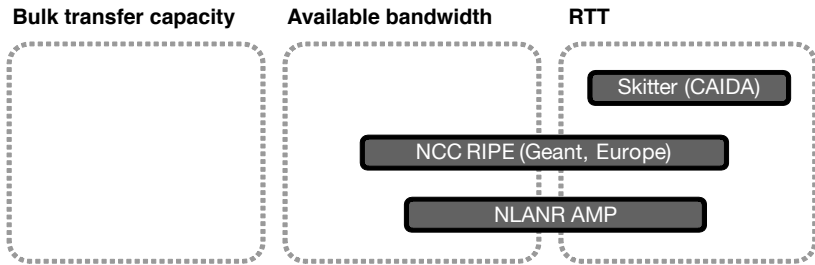


Figure 6.3 Diagram of most prominent active measurement projects in the world today. Horizontal span indicates how much of each metric the tools covered in arbitrarily relative terms.

given that such measurement by definition has to be conducted only once. Networks as they are today experience very rare alterations in topology over very long spans of time. In fact, for the most part this still remains a manual job.

The special attention given to RTT can also be explained. Although it is trivial to measure RTT, projects normally perform this measurement to use it for something else. Like Skitter, for example, is famous for visualizations of global network topologies based entirely on multipoint-to-multipoint RTT measurements. As was already mentioned above, RTT measurements are mostly elements of largely statistical research.

Some of projects from Figure 6.3 will be retouched later in this chapter when actual *measurement boxes* will be considered. It should be noted that there are definitely projects outside of the list in Figure 6.3. The target of such projects in most cases is building either a tool or even a device. Devices can be leased or even sold to third parties to be used in practical measurements. Such cases will be considered in this chapter.

6.2 Test Traffic Measurement Box

Although all the three projects listed in Figure 6.3 are equally global in scale, *TTM Box* is arguably the most commercialized project of all. Besides, authors and current maintainers of the project claim that they have been there from the beginning of IPPM standardization process. Although the connection is not very clear, there is some sort of connection between TTM Box and ntop based on a few presentations from the many available at the website of the project.

6.2.1 Test Traffic Measurement Project

The project is mainly hosted at [17]. The plain look of the website is deceiving because each entry in the menu is full of resources. There are especially many in documentation and presentation. Since TTM Box used to be a scientific research there is no surprise there. From all papers, special attention is to be paid to the paper from PAM2001 conference PDF of which can be easily located in documents at the website. While the paper is a useful source of technical details, the 8-page whitepaper from the same place offers enough information on the user level. Technical details provided further in this section are based on a number of sources about TTM Boxes and user experience but the contents are processed for better presentation instead of using visuals from scientific papers directly.

After all, the user side of TTM Boxes should be different from the viewpoint of its creators. This section is just this, – the user viewpoint of the project.

Overall procedures related to handling TTM Boxes can be found in Figure 6.4.

First, you have to purchase the box. Prices can be found in the white paper as well as on the website. There seem to be two configurations of the box, the “light” and the “professional”. The distinction in functional-

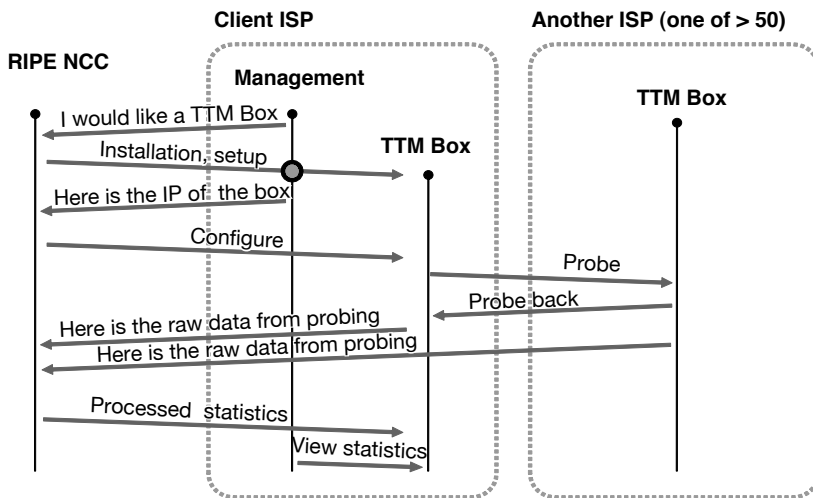


Figure 6.4 Overall process that should be undertaken by an ISP to get its own TTM Box and to start working it.

ity however is limited to differences in software while key hardware components are the same in both “versions”.

The purchase is finalized by the actual installation of the hardware consisting of the TTM Box itself and a GPS receiver connected to the box. While TTM Box can be installed indoors, GPS receiver obviously has to be mounted out in the open.

6.2.2 TTM Box as a Black Box

Before, in the meantime, and after that, the box remains the perfect *black box* to the customer. And there is a very good reason for this. Just because you buy the box it does not mean that you can use it. Consider the ramifications of the opposite case:

- you would know global IP addresses of all other TTM Boxes across the globe;
- you would (have to) know IP address of TTM headquarters which would make you a potential abuser, and even more so if this information is stolen;
- you would not know what to do with the box, – bandwidth probing is fairly complex and requires centralized coordination, i.e. exactly what TTM headquarters are for.

So, you end up with a black box. How, then, does the measurement process itself happen both within this black box and, what is more important, among all other members of TTM community? TTM community is the core of the project. The more members join the project the finer granularity each member would get of the global Internet’s performance. It is unclear as to how many members are in the community at the time this book is being written, but in 2006 there were over 50 members, majority of them in Europe, some in USA and one in New Zealand.

6.2.3 TTM Box Communications

It might seem strange at first, but the black box installed at each site is fairly unintelligent and very far from being independent. As per Figure 6.5, the measurement process requires constant supervision of the main TTM server, a TTM Master Box. To describe a single complete cycle, the process

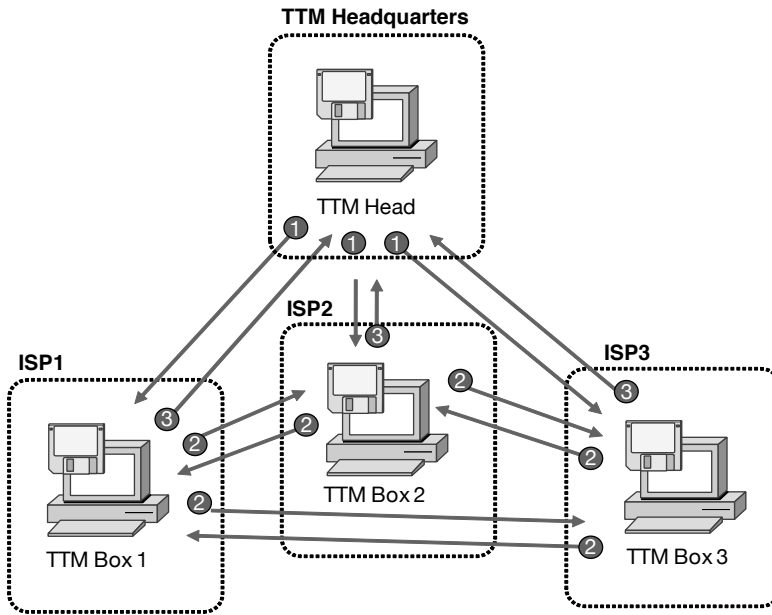


Figure 6.5 Diagram describing one complete cycle of an all-to-all measurement using centralized managed from TTM headquarters.

happens in the following steps (numbers match those in Figure 6.5):

1. TTM Head dispatches measurement tasks out to all TTM Boxes in the community; depending on the logic decided on by the centre, there could be a limited set of members per measurement.
2. Each TTM Box accomplishes a set of measurements as per the dispatched task, various tools are used at this point, again, depending on the centre's decision, which, in turn, relates to customer's wishes or technical needs.
3. Having completed the measurements, each TTM Box sends raw data back to the TTM Head because without the global state it is very difficult to turn raw measurement data from a single point into meaningful statistics about global network performance.

Figure 6.6 is a less crowded diagram of all possible communications TTM Box can participate in. Each TTM Box can communicate with three

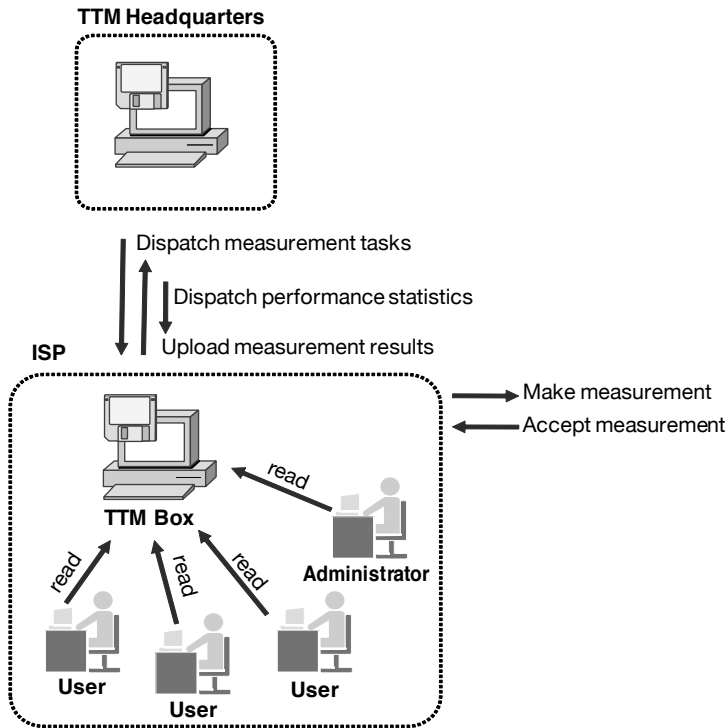


Figure 6.6 Diagram of all possible communications in and out of TTM Box.

parties, – TTM Master Box, any other TTM Box in another ISP, and users that contact the box in order to obtain performance statistics from it. Now, the first two were just covered by the previous visual, but the last communication pattern requires special attention.

6.2.4 Role of an Individual Box

As was already mentioned, TTM Boxes are not very intelligent. So, even though they perform the measurement and in the process have to communicate to other TTM Boxes outside (as per measurement dispatches from TTM Master Box), they are not in the position to process raw measurement data.

As a small detour, it is worth mentioning what kind of raw data each

TTM Box is in charge of. There is a reason each installation includes a GPS receiver. It so happens that the majority of measurement data a TTM Box is to handle is not its own measurement data. Based on active measurement methods considered earlier in this book it should be clear that GPS is used only when one-way measurement is in question. In the one-way measurement, some other TTM Box is to create the probe include timestamp along with it (normally incorporate it along with the packet payload) and ship it out. The destination TTM Box is to accept it, look up its timestamp, check its own time, and make the raw sample in form of the interval between the departure and the arrival time of the probe. Therefore, most of the time each TTM Box is handling only the downlink performance data which is clearly only half the needed performance information.

The other half is provided by TTM Master Box when all measurement data is aggregated from all TTM Boxes in the community. Being in possession of the global state, it is easy to infer performance statistics for each particular TTM Box. When this is done, the TTM Master Box dispatches these ready-made statistics back to each TTM Box to be viewed by the users.

Now is the time to mention that TTM Box is not only the key to the global network performance, it is in fact a service to be provided to the users within your ISP. Those ready-made performance statistics uploaded into your TTM Box can be offered to individual users within your ISP as a web service. Here, there is no technological news, as TTM Boxes use the same web application mechanism *ntop* and *PRTG* use for *SNMP* and *NetFlow* statistics. Those cases have been covered extensively earlier in this book.

It is interesting to note that even administrators of the client ISP, i.e. people that purchased a TTM Box in the first place have no special treatment here, – there is only one web interface to the TTM Box and it is the same regardless of who you are. One could even guess that operators at TTM headquarters frequently contact your TTM Box as users in order to verify whether everything is in proper place.

As a summary, it is necessary to mention that TTM Box is more of a project than a tool. In fact, enclosed in each TTM Box are many measurement tools, most of them conventional and readily available, such as ping, traceroute and even *iperf*. Although there seems to be some ongoing research in to the academic matter related to active measurement methods, at the present moment all measurement tools used by the TTM Box are conventionally available ones.

It does not mean that the box is empty. TTM project had a number of original ideas and even a larger number of successful adaptations of

someone else's great ideas. For example, two different kinds of database are used to store processed performance statistics and raw data. Log files use the same idea that is used in *ntop*, i.e. fixed size. Some people are also looking into various visualization methods to improve visuals served to users through the TTM Box's web interface. As of now, the interface is mostly the same as you would get with PRTG or *ntop* tool.

It is however likely to improve in the near future.

6.3 QoS Boxes

It is one thing to follow IPPM guidelines or even stay within a subset of metrics defined by IPPM and is completely different to wander further into the area of QoS.

First of all, QoS is not yet fully defined in terms of performance metrics. On one hand, there are people that believe that QoS is the same thing as network performance thus making it a precise science with well defined metrics within the IPPM list. But the other view at QoS is entirely subjective making it impossible to describe it by a set of metrics.

For example, questions like "what is the quality of this video stream" are subjective and will require the use of a combination of traditional performance metrics along with some precision tradeoffs to be able to come up with an acceptable answer.

Worse yet, QoE is another buzz word in the networking world today and if you add QoE to the above question things get even more complicated making it impossible to describe performance in terms of rigid metrics. QoE standardization is still ongoing and the direction it is moving in is not very optimistic for those that expect a clear set of metrics. As of now, QoE is essentially assessed through subjective opinion processed through some basic statistics, like MOS (Mean Opinion Score). This path is unlikely to lead to the likes of IPPM specifications.

Therefore, on one hand there is evidence that more details are demanded by a few users about network performance than can be offered by traditional metrics, and on the other a broad proliferation of measurement tools has long been demanded by the general public. This section will look into two particular cases, one is another case of a standard QoS Box and the other is a publicly available general use software agent that claims to provide the entire spectrum of IPPM metrics having completely purged the hardware component.

6.3.1 QoSMetrics Box

The earlier example of the TTM Box may have been misleading towards the notion that today's network performance market is saturated with the likes of TTM Box. This is, however, far from reality. In reality, TTM Box is the only of its kind on the market today.

Arguably the second in scale after the TTM Box project is the QoSMetrics Box. The project's scale is the size of *Renater network* in France. Presently, there are 11 measurement points across the network, each of them obviously furnished with the installation of the QoSMetrics Box provided by the measurement team of the project.

Figure 6.7 contains the overall structure of the project. Given that QoSMetric Boxes are installed within a closed community, there is not much one can learn about the inner workings of the measurement project itself. On the public side, however, users can use the web interface which displays regularly updated statistics uploaded by each probe within the closed community.

At the time the book was written, the project performed 4 kinds of measurement, – end-to-end one-way delay, jitter, packet loss and min/max hop count. The first three metrics will require the use of artificial probes and a specific measurement method, while the hop count is obtained by using the famous traceroute tool that exists virtually on any operating system today including Windows.

Processed statistics are available publicly at [16] in form of a webpage.

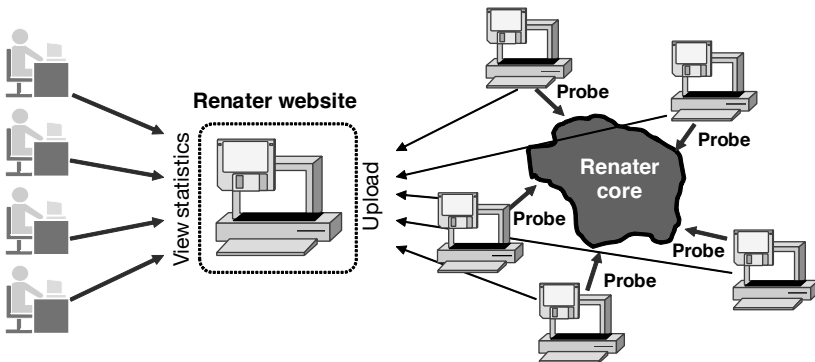


Figure 6.7 The diagram of the service offered by the Renater's QoSMetrics project.

Since there are 11 probes distributed in the network, all measurement results are displayed as 11x11 matrix. The example of a delay matrix can be found in Figure 6.8. Each of the rest of the metrics has a separate table. Some colouring is used to provide some shallow statistical insight into the history of each sample in matrices.

If you keep the website open in your browser, it will automatically update itself every minute refreshing all tables on the page.

At the end, the issue of complexity is worth mentioning. Both TTM Box and QoSMetrics Box projects deal with a small number of probes scattered across the network. In case of TTM Box the network is global and virtually has no limits, while in case of QoSMetrics Box the network is internal to Renater network. TTM Box project has more probes by today but still conducting 50^2 measurements is not a very difficult task and is achievable with a reasonable update interval. Larger user bases should however create additional concerns.

6.3.2 QoSmetrics' Tools

What happens when you remove the need to install specific hardware to be able to measure performance between your own computer and the entire outer world? Especially if this hardware is a GPS receiver, the cheapest versions of which are still about 250USD. Apart from the price, GPS systems are very messy to maintain, – you have to install them outdoors, provide about half the sky of the view to provide round the clock connectivity, etc. As a common user all you have is your small apartment and a desktop computer with a fairly common OS installed on it. Based on statistics, this OS is most probably Windows.

Now, what do you call a person who has the above environment and still need to measure performance between you and the global Internet? They are called “gamers”.

It is unclear whether it came as a surprise to most professionals in the area of active measurement when gamers came forth and demanded software that could measure network performance in accordance with IPPM metrics. They had grasped the terminology and they demanded the service.

Why is network performance so important for *gamers*? From all aspects of it, end-to-end delay and in some cases jitter are the two important variables in network gaming. It is a shame when you lose a game just because the response time of the gaming server to your requests is two times that of your main opponent in the game. Given that distribution of end-to-end

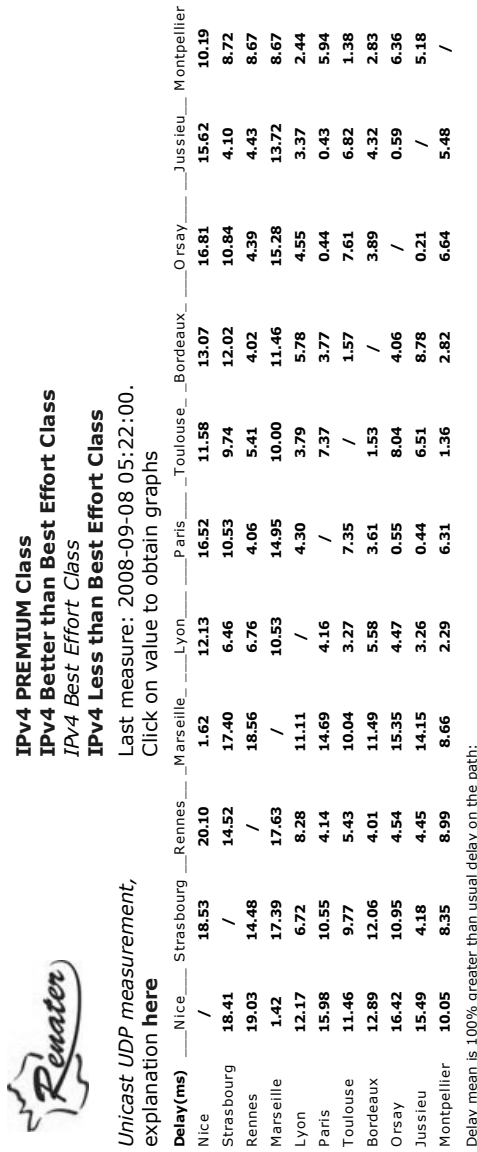


Figure 6.8 Example of a delay map produced regularly by the project.

latencies in the global Internet are very far from being uniform, you end up with this inequality right from the start.

So, while measurement boxes were taking is slow and easy in the research community, gaming community encouraged development of other kinds of boxes, – *software only boxes*.

One company is particularly worth mentioning here. *QoSMetrics* used to be its own network until roughly a year ago (at the time this book was being written) and had nothing to do with the QoSMetrics Box project that was just considered in the previous section. QoSMetrics in this section is the name of the company what was recently taken over by another company very possible to incorporate the end-to-end measurement experience of QoSMetrics in a larger product. NGN was used as one of the keywords in this purchase. Earlier in this book it was repeatedly stated that active measurement is the key technology to NGN when the technology is fully provisioned in networks in a few years to come.

Figure 6.9 offers a simplistic view of tools developed by QoSMetrics and more importantly visualizes their use. There are three major tools that QoSMetrics had to offer. NetWarrior is the flow probe, i.e. the effective solution of flow monitoring. NetAdvisor is a related product that will help you generated effective reports from raw analysis data.

But special attention has to be given to *NetAgent* since only this part is truly end-to-end and comes from the area of active measurement. NetAgent is all about your server provider being nice to you by offering to test network performance between you as the user of its server and the service itself.

This case is special in that it is very NGN-like. While all the above active measurement boxes were limited to ISP or inter-ISP links with global IP accessibility, NetAgent took on the task of measuring the full end-to-end path. This is exactly what NGN is paying special attention too. It does not mean, though, that NetAgent is about NGN. But the use and especially the demand for the tool are indicative of the direction NGN will be paying special attention in the future.

There are several intrinsic problems with such end-to-end measurements and it is unlikely NetAgent was able to solve it due to their nature. All the problems are depicted in Figure 6.10.

First of all, the dispatch of NetAgent tool itself has to be reconsidered taken into account the fact that the user does that over web interface. So, instead of “dispatch NetAgent” in Figure 6.9 it should say “download NetAgent” as in Figure 6.10. Here, the direction is opposite because user has to request the download by clicking on a link on a webpage. The

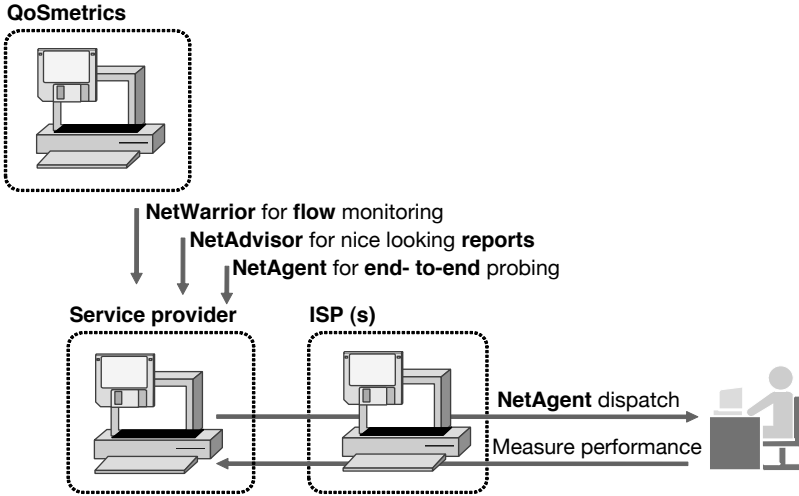


Figure 6.9 Tools offered by QoSmetrics including the special case of NetAgent.

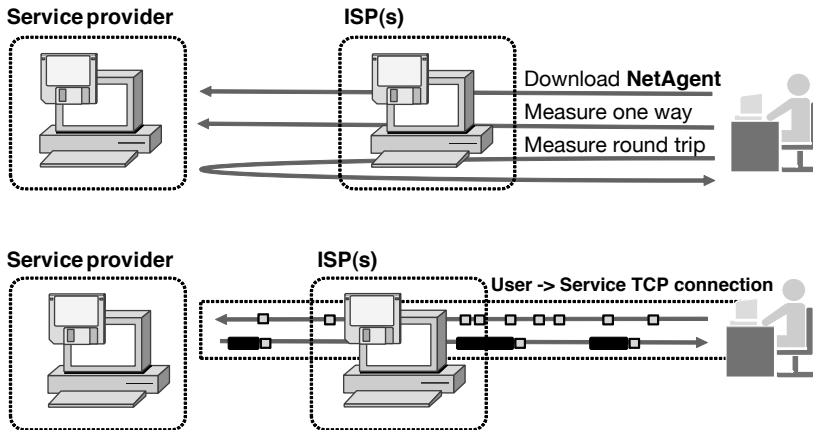


Figure 6.10 Visualization of problems existing in end-to-end probing today.

connection, therefore, originates at the user and terminates at the service provider's end.

This connectivity problem permeates the entire situation with end-to-end probing. In the short version the problem is in the fact that end users do not normally possess globally reachable IP addresses. This means that whatever communications happen between the user and the service provider have to be initiated by the user.

Now, the above is not always true, however. There is a huge area of NAT tunnelling where someone on the either side of an ISP can manage to initiate contact with someone on the inside of it, but this is a fairly complex and very unstable process and is used only by a small subset of network software today. Online gaming usually does not reach this far and remains within the traditional networking toolbox.

This is what happens in a traditional TCP connection, like that initiated by your browser to refresh a portion of the webpage you are currently playing on. Your desktop initiates a TCP connection by contacting the IP address of the web server of your service provider. The reply is then sent back to you by attaching data to ACK packets of your TCP connection. Since you initiated the connection, all requests flow from you towards the service provider and all ACK packets flow in the opposite direction, thus, allowing your server provider to communicate with you.

With modern software packaging one might not be aware of these details given that TCP sockets is the lowest unit of network programming even for the low levels of C/C++ programming. It is necessary, however, to be aware of the fact that in you to service provider communications, the volume of data transmitted back to you is much larger than you transmit towards service provider.

The same is applied to measurement. Because you are limited in connectivity, you can either measure a one-way delay from you to the service provider or a round-trip of the same path.

This reveals a major issue. If the main data is transmitted on the return path how helpful are the measurement results obtained on its forward portion? Asymmetric routes between ISPs are becoming more and more common nowadays, thus, creating an additional issue for round-trip measurements. In summary, end-to-end measurements conducted in such an environment can only be reliable to a certain extent. Many years of research experience dictate that error in such measurements can be very high.

Another huge problem is hardware-free end-to-end measurements is synchronization. GPS in TTM Box explained earlier is not an idle appendix to the system, it provides the precision within 100ns, i.e. more than

enough for measuring modern broadband networks. Besides, when special hardware is used, tweaks in operating systems can allow to increase user-level interrupt precision to 10us, which is good enough for packet-by-packet interrupts in complex probe structures.

In the special hardware-free setting you can only get a certain level of synchronization. For example, the NTP (Network Time Protocol) can offer synchronization within 1ms, which is too coarse for most probing targets. QoSmetrics, however, patented their own network synchronization technique that is supposed to offer better precision.

The interrupt granularity of common operating systems used by gamers still forms a huge precision hurdle. Both Windows and default installation of most Linux operating systems will only offer kernel time slices of 100ms, which is thousand times as much as required by most probing scenarios. It is possible, however, that the subset of network performance that interests gamers among other performance-constrained network services are not as stringent as they are in research community.

Chapter 7



Active Measurement in Context

The context in this chapter stands for a larger picture where active measurement is used in practice for a higher purpose. Based on most explanations earlier in this book, NGN is only one of many *practical implementations* of active measurement. However, given that NGN is still in the early stage of development it is really hard to predict which practical applications within NGN active measurement will be integrated into in the future.

When looking outside of NGN, there are a few examples where active measurement is an integral part of a bigger technology. An example of this was already given in the previous chapter where active measurement was actively used to define network performance of the end-to-end path between the user and the service provider, in a particular gaming server.

There are, however, other uses. Two particular uses are considered in this chapter and are based on practical needs.

First, raw measurement data is very difficult to handle. Tools, methods and projects apply various statistical methods to create a meaningful output. That output is intended for human use and therefore is not very scientific. Bluntly, as long as the visual output is pretty users are fine with it. If machines were asked to view multiple lists of measurement results, they would not know what to do with them.

These lists are referred to as time series in traditional data mining.

The connection with traditional data mining stops there, however, as none of existing pattern recognition methods will be able to extract mean-

ingful information from arbitrary length raw measurement time series. This problem is tackled in this chapter.

The second practical problem tackled in this chapter comes from the nature of active measurement technology. By definition, end-to-end active measurement exists because it concerns the inter-ISP network domains which are not maintained by any single ISP and therefore active measurement is the only technology that can infer network performance in this administrative vacuum.

However, this also efficiently loses the topology of the underlying network. So, even if you have 100 probes scattered all across the global network, you have no way of making topological decisions. In short, there are scenarios where topology needs to be inferred from otherwise topology-less active measurement results. An entire section of this chapter will be dedicated to this issue.

7.1 Management of Measurement Results

As was already mentioned before, ITU-T has recently developed a document on *management of performance measurement for NGN* [19] which considers many issues concerning actions past the measurement itself. The point this chapter is trying to make is that this measurement data should be utilized as integral part of larger technologies. However, what the document in [19] covers is very small compared to all problems that arise when you try to apply *measurement results* to virtually anything.

For example, consider the following questions:

- how does one compare two separate sets of measurement results?
- how does one find out whether two sets of measurement results come from two end-to-end network paths that shared a portion of topology?
- are there more efficient methods of storing raw measurement data other than storing *time series* in a database in the raw form?

So, while NGN standardization is mostly concerned with human aspects of handling processed measurement results, the part where raw measurement data get processed in order to create meaningful output is equally as important.

7.1.1 Problem Statement

The storage, retrieval, and comparison of time series obtained directly by active measurement are not commonly considered in data mining literature. Research in active measurement usually targets a certain network performance metric, such as available bandwidth, bulk transfer capacity, delay, etc., and completely concentrates on achieving good performance by capturing the true values of the target performance metric.

Consequently, active measurement methods normally test a proposed tool on a single path in the Internet to prove the proposal's validity. When multiple paths are used for practical verification, each link is normally measured independently, and the resulting time series are also analyzed independently.

In short, active measurement research is essentially centered around a given *performance metric* and strives to achieve acceptable performance and effectiveness within the needs of a particular end user or application.

Since applications normally operate on a point-to-point basis without perfect precision, the results of such research are acceptable for most uses. Some reviews of the current state of research in the area of active probing can be found in [30] and [44]. These reviews performed a number of tests on several prominent active measurement tools and found that measurement results under identical conditions are extremely inconsistent.

It is conceivable, however, that the quantitative analysis of active measurement results will soon become the key element of network performance analysis. Probing will increasingly become more popular not only within a single ISP but also across multiple ISPs and will be used to study inter-ISP network performance. Studying the performance of a network rather than a single path requires the installation of many probes across the network and using many-to-many complex probing algorithms.

The *time scale* issue is important in any technology that uses active probing. For example, some *TCP benchmarking* can also be called "active measurement," since such methods also send packets into networks in order to infer the characteristics of TCP performance on a network path. TCP benchmarking, a very limited area of active probing, is very different from the active probing discussed in this section. While TCP benchmarking requires time scale in the range of several dozen milliseconds, the time required for TCP window recovery, the active probing used in this section normally stretches over several dozen seconds. In some cases active probing can also be continuous, i.e. probes may be transmitted at regular intervals to perform data processing in real time parallel to probing.

Section 2 of this chapter explains in detail why traditional *data mining* cannot be applied to active measurement results. It also discusses *artifacts* specific to active probing and lists research issues addressed in this chapter. Section 3 proposes a solution in the form of a method of processing active measurement results based on the concept of “excursions” extracted from a measurement time series. Simulation results can be found in Section 4, and Section 5 contains the conclusion.

7.1.2 Data Mining and Active Probing Results

One of the most recent books on time series mining is [37]. Although the book contains very good information on the data mining fundamentals, its methods were not directly used in this section, since our basic approach is quite different from any other methods in the field. These reasons are discussed in this section.

As was already mentioned earlier, active probing also creates a unique form of *time series*, which we referred to as a *measurement time series*. Additional issues are introduced when probing is simultaneously conducted on multiple paths, and there is a requirement to merge a measurement time series from multiple paths into a single result. All these issues are also covered in this section.

Regardless of how advanced contemporary data mining methods are, mining of time series data can simultaneously be both very complex and very primitive. Both the book in [37] and the research work in [42] list primitive sample-by-sample comparison methods along with complex variable-length average window methods.

The reality, however, is that even though many advanced methods are used for mining time series, the results provided by a constant-length average window appear the most stable and fit for a variety of time series sources [37].

The analysis of a time series in *phase space* [42], i.e. disregarding the position of the sample and only using the change between several successive samples also proved to be only applicable to a subset of time series flavours.

Both phase space and the amplitude analysis of measurement time series, however, will fail due to the intrinsic nature of active measurement results. By definition, measurement results originate from the interference of probe packets with traffic flowing through the path at the time of probing. The term “at the time of probing” itself is unclear since no practical method exists that can define exactly at what moment in time a probe

packet interacts with a traffic packet, let alone which packets affected each other's traversal through the network.

Such fundamental uncertainty deprives measurement time series of the possibility to attribute a certain *probability distribution model*. Similarly, no distribution model exists for network traffic, even though many current studies in traffic modelling continue to point to one model or another. This uncertainty is also the main reason why raw samples or averages with any window size will appear completely random.

Measurement results refer to a set of time-value pairs where time is the moment of the measurement, i.e. the time at which the probe was either sent or received depending on which end was used for calculation, and value is the worth of a metric targeted by the measurement.

Several metrics are defined today in IETF IPPM RFC documents. Their overall description can be found in RFC 2330 [41]. However, they all share one attribute, – the presence of either a lower or an upper bound for each metric. For example, RTT or one-way delay of a path has a global minimum which is the shortest time required for a packet to physically travel through the link one way or round-trip.

Since upper bounds can easily be transformed into lower bounds by an inverse operation, each performance metric obviously has a global minimum. We will later use this property to define temporal patterns.

Probing paths are directional, as described in Figure 7.1. The measurement time series obtained from a measurement on AB is not the same as the time series from CD. Although the traffic in both directions may go through identical intermediate routers, the influence of packets flowing in opposite directions is minimal. The majority of traffic interference originates from background traffic packets lining up before the probing packet in each router along the path. This phenomenon is well established in active probing literature.

To preserve relative simplicity, measurements in this section are per-

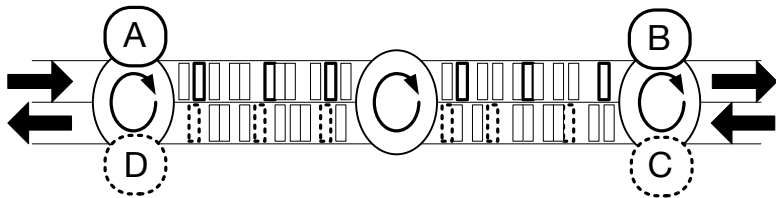


Figure 7.1 The problem of directionality in active probing.

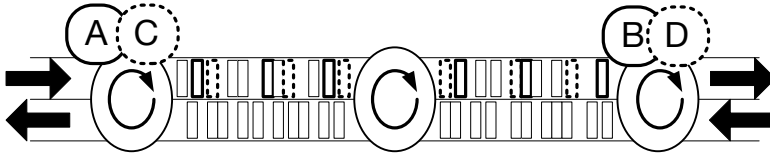


Figure 7.2 Problem of loosely coupled probes.

formed in one direction only. Naturally, the background traffic probes also interfere with traffic flowing in the same direction as the probes.

The *loose coupling* problem, depicted in Figure 7.2, can also be categorized as the lack of synchronization problem. It is stated as follows.

Even if measurement paths AB and CD are identical, there is no guarantee that the measurement time series will also be identical when compared to each other. In fact, even if A and C send their probes at the same time, one of the two packets will have to wait for the other, thus creating subtle differences in the resulting time series.

In reality, however, perfect synchronization of A and C is difficult, which explains why probing packets experience different interference patterns as they travel separately through the path.

To compensate for loose coupling the proposed method has to allow for a certain degree of freedom in terms of missing features and local differences between the two time series. Here, it is important to stress that conventional data mining, including time warping techniques, react poorly to loss of features in a pattern. The proposed data mining method on the other hand is more tolerant to such loss.

7.1.3 Mining Method

Based on the unique properties of the measurement time series explained above, this section describes a method that extracts patterns from a measurement time series. To simplify analysis, a one-way delay is the only metric used to analyze the method's performance, since it reflects one-way interference instead of a round trip.

As previously mentioned, each measurement time series has a global minimum. Therefore, we can assume that a measurement time series consists of a set of *excursions* from the global minimum, as depicted in Figure 7.3. Each excursion has its starting point which is defined as the point

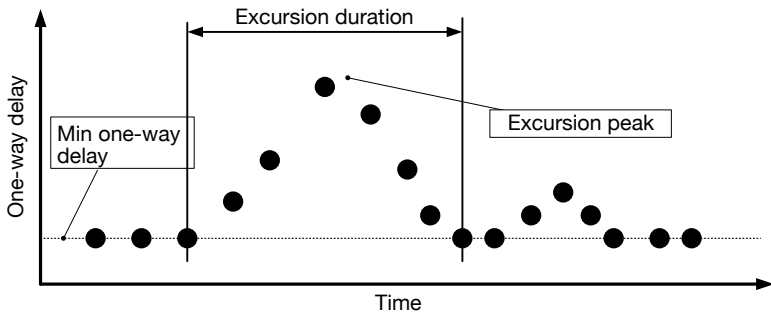


Figure 7.3 Definition of excursions found in active measurement results.

when one-way delay is higher than the minimum, and a similar end point where one-way delay returns to the minimum, and its peak defined as the maximum one-way delay achieved within the excursion.

As was already mentioned previously, both absolute and relative values of metrics cannot be used for direct comparison. In the proposed method absolute values within the excursion are completely ignored, leaving only the starting point of the excursion and its duration, as per Figure 7.4. Excursions are separated by gaps which are defined as periods of time when a one-way delay remains at its minimum. On the physical level, this stands for the lack of interference with background traffic, which is, in fact, quite often the case in real networks. The actual packet traces replayed within the simulation exhibit a bursty nature with relatively long periods of relative traffic “silence.”

Figure 7.4 contains the visual representation of the patterns created by the proposed data mining method. Absolute values in the original time series are completely ignored. A threshold at the 5% level of the minimum value is used to detect when the signal departs from or returns to its global minimum. The mining result is a set of excursions from the global minimum. The time span between excursions, which is defined as a *gap*, is also used to compare different patterns. The process of pattern creation is described in detail below.

This section proposes two patterns based on excursions. The first pattern accounts for both the excursion and the gap and is referred to as the pattern with gap penalty. The other method ignores the gap and accounts only for the duration of the excursion.

The actual pattern is more complicated than just a set of excursions. Even when two separate sets of excursions are extracted from similar ac-

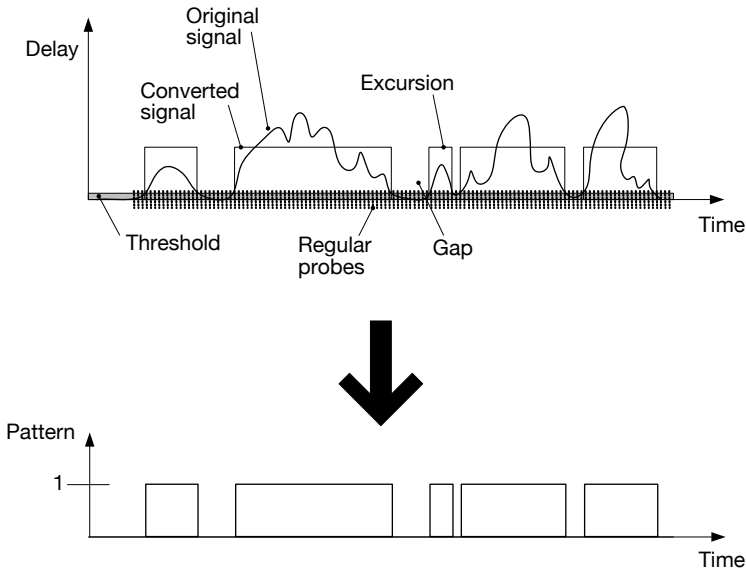


Figure 7.4 Excursions based only on lengths of excursions and idle periods.

tive measurement time series, there is no guarantee that one set of excursions will be identical to the other. Some differences in the set of excursions normally exist and are proportional to the differences in the original active measurement results. Therefore, when two sets of excursions are compared, certain flexibility should be allowed to compensate for missing local features in both sets.

In this section we implemented the above pattern as a comparison algorithm and referred to it as a *pattern*. The pattern is a quantitative representation of a set of excursions and the gaps between them. The details as well as a more solid definition of the pattern are given below.

Considering the practical implementation of the proposed method, all data used in the analysis including the original active measurement results and the extracted excursions are stored as a database table. A separate table is created for each measurement time series. This simple transform is described in Figure 7.5. The original table contains the time series, i.e., the sequence of the time-value pairs. The results of the transform come as a table where each line is an excursion with its starting time and duration. Since the starting time is defined, the time sequence of

the excursions can be restored when calculating the pattern based on the sequence of excursions.

A pattern in this section is a quantitative measure that represents the sequence of excursions and gaps between them from which it was derived. However, when two measurement time series are compared, both the final and all intermediate values of the pattern are considered. In other words, the pattern is a function of a discrete value in time. The method used to compare the two patterns is described in the next section.

We define indexing as a process that converts the contents of the Patterns table in database structure (the table containing the sequence of excursions) into an *index*. The structure of the Patterns table as well as the transform by which it was created from the original time series was previously shown in Figure 7.5.

Index is a more established term in data mining for referring to *temporal patterns* extracted from a time series. It is often used whenever a certain pattern is implemented within a database, where it becomes an index rather than a pattern. Both words are used interchangeably in this study, however. The use of the word pattern is retained since this section targets the analysis of a time series originating from active measurement results and does not pay much attention to practical implementation. Although the implementation of the method uses database tables for both the original time series and the extracted excursions, the database structure is not the main concern of the section. Perhaps database design as well as the efficiency of the method may be improved using the results of research on database indexing techniques. This area, however, is beyond the scope of this study.

The pattern is a sequence of two-dimensional (x, y) values, where x is the time and y is the value of the pattern. Value y is calculated differently by the two methods proposed and tested in this section. One method accounts for the duration of the gap between excursions, and the other ignores it completely.

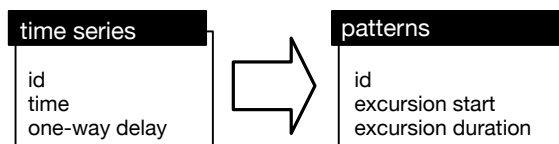


Figure 7.5 Database table used to transform time series into table containing excursions.

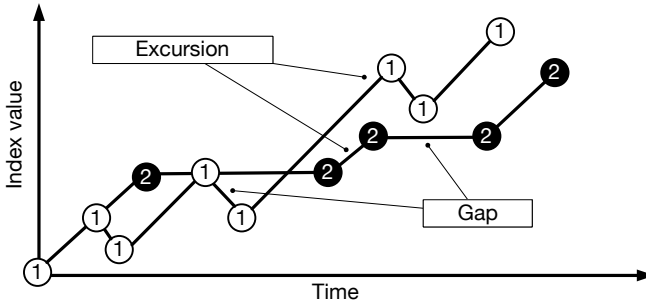


Figure 7.6 Visual representation of how index sequences are created by both methods. Gap penalty is accounted for in 1 and ignored in 2.

Figure 7.6 contains the visual representation of the process used to create patterns by both methods. In the plot, discrete pattern values are connected by lines, and each sequence is marked by the number of the method. The first method created the sequence marked 1, and the mark 2 distinguishes the pattern created by the second proposed method.

Originally, the process in both cases starts at point $(x = 0, y = 0)$. At this time both methods select the first excursion from the Patterns database table. Since ordering by time is used to read the sequence of excursions from the table, this means that the first excursion is also the first in time. From this point on, the procedure used to create a sequence of discrete pattern values is slightly different in each proposed method. The sequences are separately explained below.

The first method accounts for the durations of both an excursion and a gap between two successive excursions. Given the execution is positioned at the starting point of the first excursion and the current position in the plot is $x = 0, y = 0$, the following procedure is performed by this method:

1. The duration of the current excursion is added to both the current x and y .
2. If the current excursion is not the last in the sequence, the gap between the end of this excursion is calculated as the time span between the start of the next excursion and the end of the current one; if the current excursion is the last one in the sequence, the loop exits.
3. Reaching this point in execution means that the next excursion is

found in the database table, and the gap between two successive excursions is calculated; the value of the gap is added to x and subtracted from y , causing negative change in y .

4. The next sequence in the list is selected, and the steps in this list are repeated.

The second method accounts for the duration of the excursion and ignores the gaps between them. Therefore, all steps in the processing algorithm are identical, except where the duration of the gap is included in the pattern. In the second method, the duration of the gap is only added to x and not to y , which means that when gaps are accounted for only time axis is incremented while the index remains unchanged.

In traditional data mining, patterns are created by sequences of average absolute or relative values. Relative values are defined as the absolute value minus the global minimum in the time series. These sequences are compared sample by sample. When the timeline of two sequences is different, each sample in the base sequence is compared to the sample in the query sequence closest to it in time. Error, therefore, is the result of a simple division of a smaller value by a larger value that roughly represents the similarity of both values.

In this section, these methods are referred to as conventional methods. A distinction is made by separating conventional methods into absolute and relative, which reflects whether they use absolute or relative sample values.

7.1.4 Mining Examples

This section contains simulation results in which measurements were performed on a simulated network path. The simulation results in the form of measurement time series were first inserted into a database, where a Pattern table was created for each measurement time series. Index calculations and comparisons were performed by another program that used the database tables as input.

Figure 7.7 contains the overall topology used by all the probing scenarios in this section. The network used in the simulation was a straight path of nine routers, marked by R and a number, a portion of which was used to route background traffic. Each scenario uses a separate set of probing paths. There are two major scenarios; one uses the same path for two separate measurement sets, and the other uses paths of different lengths

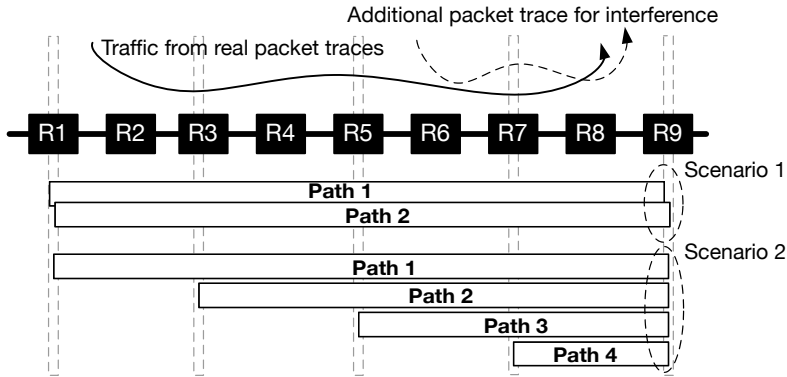


Figure 7.7 Simulation topology and probing scenarios.

but with portions of mutually shared topology. Each scenario will be explained when its corresponding results are presented.

The issue of background traffic is critical for active probing since packet-level interactions are a major source of data. If traffic is artificially generated, packet-level interactions appear very differently from those found in real traffic traces. This normally reflects that artificial generation is based on a well-defined probability distribution model and is heavily dependent on random number generation. With real packet traces, on the other hand, the probability model issue is not important, since packets from the traffic trace are generated and sent into the network exactly when specified in the trace.

In this section, two packet traces from backbone-level links were used. One was downloaded from the WIDE traffic repository [12], and the other is publicly offered by yet another traffic archive [9]. The second packet trace, which was used as a component in the background traffic, is used by the second scenario. Both traces are from 100-Mbps backbone links that constantly experience a medium level of utilization. The method was used to aggregate and split all sources found in a tcpdump-compatible traffic trace into a flow between two nodes, as described in our earlier work in [51].

The links in the simulation are also set to 100 Mbps, which is the maximum rate discovered in packet traces. Since real packet traces also originate from a 100-Mbps backbone link, there is zero chance that traffic replay within the simulation can cause congestion. Even if minor congestion is

caused by active probes added to the replayed traffic, this minor congestion is ignored, and no loss is generated.

Simple one-packet probes are used in simulations, and one-way delay values are calculated at the receiver side. The packet size in all measurement tests was set to 1500 bytes, i.e. the Ethernet MTU.

Figures 7.8 and 7.9 contain the results from index comparisons between measurements as the two index generation methods described in the previous section. Given that the same path is used by both active probe pairs and that excursions are defined by the duration in seconds, it can be assumed that patterns from different probing intervals should be relatively similar. Probing intervals are marked by millisecond values in the plot legends.

In Figure 7.8, gaps are accounted for when calculating the index that results in negative index values. This is natural since real networks rarely congest to a level where one-way delay is seriously affected, and, therefore, there are always more gaps than excursions in extracted patterns, in terms of duration.

Starting from differences at 40 ms and higher, the second pattern with lower probing frequency departs from the base index. Naturally, since frequency is lower, fewer excursions are also detected, and at the same time they become longer since small gaps between them are not registered. For better viewing this artifact is stressed by using step curves in plots. In this way, the duration of excursions is made more visible in the dense areas on the plot.

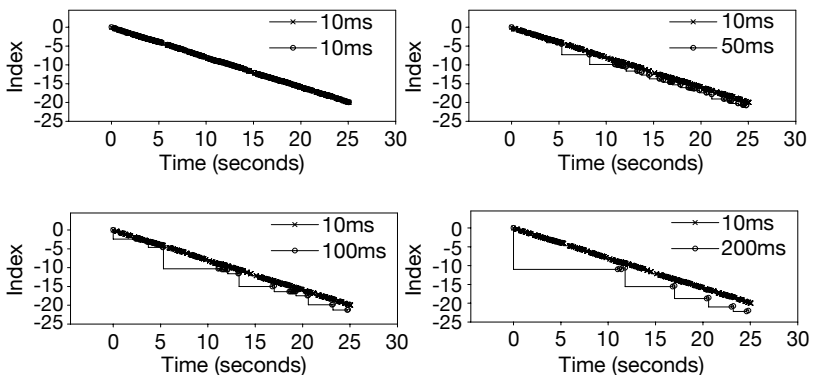


Figure 7.8 Comparing indices created on same path using various probing intervals and considering gaps between excursions.

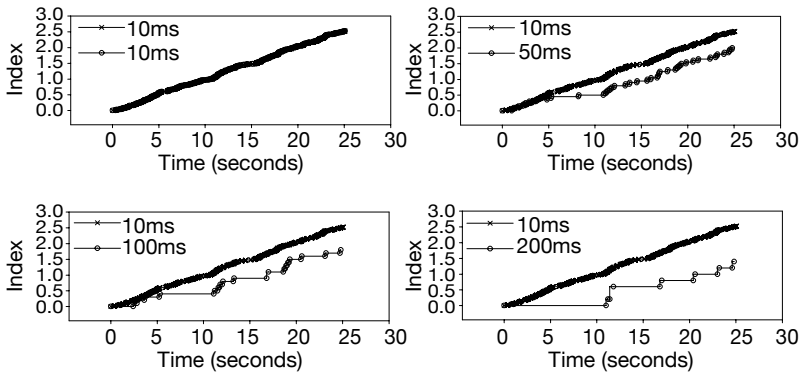


Figure 7.9 Comparing indices created on same path using various probing intervals and ignoring gaps between excursions.

Figure 7.9 contains results obtained using the second method of index generation, i.e. the one that does not account for gaps between excursions, which is clear from the many horizontal lines on the plot. The overall performance is identical, as in the first method, i.e. the match results slowly worsen as the probing interval increases which is caused by overlooking small artifacts in the measured traffic.

The difference of vertical scale in 7.8 and 7.9 is the difference between the aggregate duration of excursions versus the aggregate duration of gaps. As a rule, the former is several times smaller than the latter. In Figure 7.10 the results are normalized to the fraction of 1. Using the relative approach allows effective comparisons of results within the same method or the same environment used to conduct active measurements. Figure 7.10 displays the aggregate error using the two proposed and the two conventional methods. The horizontal axes of all plots represent which index sequences are used, i.e., 1 stands for the 10-ms interval, 2 for the 50-ms interval, and so on. The combination 1 – 1 stands for the matching results between 1 and 1 and 1 – 4 for the matches between indices 1 and 4, respectively. This notation is used throughout this section to discuss comparison results.

The proposed methods in Figure 7.10 present adequate performance in view of the change in probing conditions. With each increasing probing interval the index sequence departs farther from the base sequence, covering a larger area. This area is referred to as error and appears proportional to the growth in the probing interval.

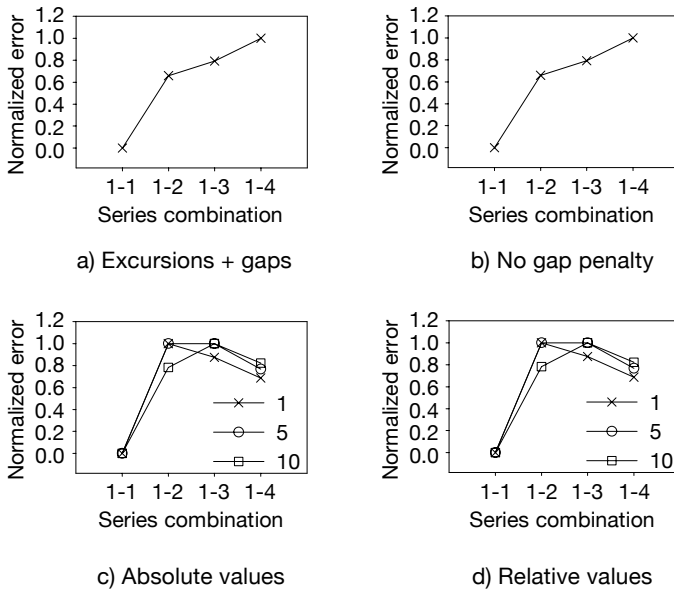


Figure 7.10 Comparison of matching errors produced by proposed indices and traditional sample-by-sample method. Legends in plots stand for smoothing windows of 1, 3, and 10 samples, used by traditional methods, and numbers on horizontal axis stand for sequence number of probing interval.

In conventional methods, a wider probing interval is represented by a larger window size used to calculate the averages. Windows of 1, 5, and 10 samples were used. In all cases, both absolute values and relative values performed poorly. The curve mostly represents the smoothing effect from using a larger window size to calculate the average. Although such a window-based average approximation of time series works for many time series, in cases of active measurement such performance is unacceptable.

From the shape of curves produced by conventional methods, both methods clearly respond only to the change in window size used for the average samples in the measurement time series. In this case, as the curve in Figure 7.10, the window of one produces the sharpest curve, while the windows of samples 5 and 10 result in a smoother curve. The shape of each curve does not correspond to the change in probing conditions.

The second property that has to be tested is the loose coupling of probes. To test this, measurements on the same path were performed

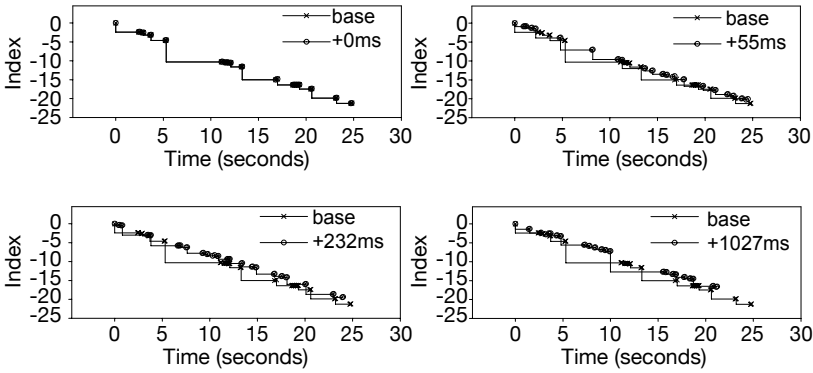


Figure 7.11 Comparison results among indices created from probing results shifted in time against each other. Time shift in ms is stated in plot legends. Gap penalty is applied.

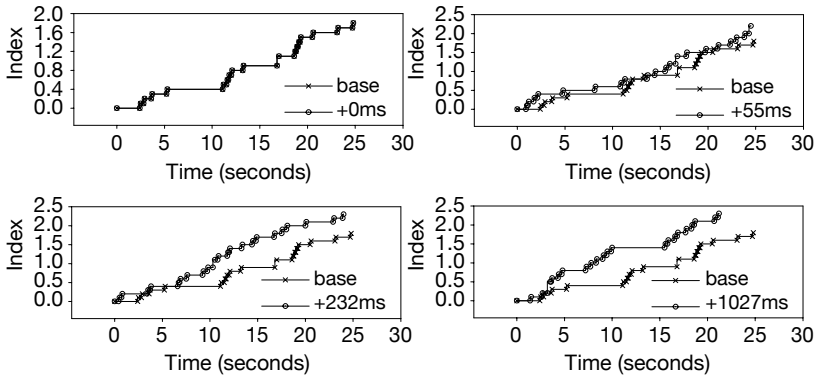


Figure 7.12 Comparison results among indices created from probing results shifted in time against each other. Time shift in ms is stated in plot legends. Gaps between excursions are ignored.

with a constant probing interval of 100 ms, but with additional time shift purposely attributed to one of the active probing sources. In this way, two probes from each source were always separated by the value of this time shift. To ensure that the probes from the two sources are never transmitted at exactly the same time, the four shift values are not round numbers. Each time, the shift is marked in the plot as +xms, which signifies the shift against the base sequence.

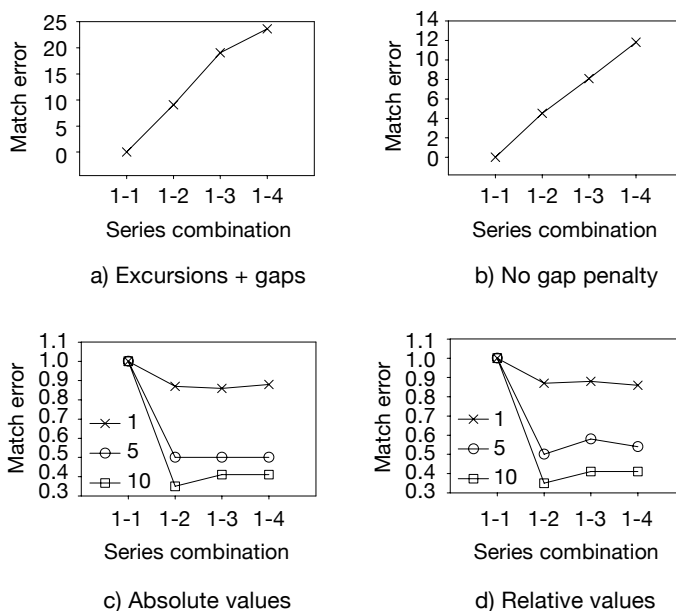


Figure 7.13 Comparison of matching error/ratio values produced by proposed indices and traditional sample-by-sample method. Legends in plots stand for smoothing windows of 1, 3, and 10 samples, used by traditional methods.

The performance in both Figures 7.11 and 7.12 is nearly equally good. However, since vertical scales have one order of magnitude difference between methods 1 and 2, the relative comparison performed in Figure 7.13 places the matching error in both methods in about the same range. Since conventional methods perform sample-by-sample comparisons, they also naturally calculate *match ratio*, which is simple division of the smaller value by the larger one. Error for the proposed methods comes naturally as a distance between index sequences. Since match errors are compared to match ratios, note that one is the opposite of the other.

In Figure 7.13, both proposed methods perform roughly the same, except for the higher error when no gap penalty was used. Considering that there are always more durations in the gaps than in the excursions, the final index in the no-gap case is around 2.5, while the other method reaches 20 by the end of the sequence. Therefore, the error in the no-gap case is comparatively large.

7.1.5 Discussion of Analysis Methodology

This section raised issues related to the practical use of active measurement results. On one hand, today active measurements are traditionally confined to a single path without analyzing the measurement results relatively simultaneously collected from multiple probes in the network.

On the other hand, the mining of measurement results by conventional time series data mining is not valid, proven by the results in this section. Since precise timing of excursion starting time and its duration is not an option in most probing scenarios, the proposed method also had to allow certain flexibility in time. To do that a graphical solution was proposed in two separate forms, one that included a penalty based on gaps between excursions, and one that ignored the gaps.

We considered several measurement problems, such as loose coupling between separate probes, different probing intervals, etc. All of these problems become important when a large scale probing topology is deployed. For each of the discussed problems, a test was performed using actual traffic traces in simulation environments to study the performance of the proposed methods versus conventional sample-by-sample matching.

The method that ignores gap penalty performs equally well or better than the other method when probes are perfectly synchronized and when the probability that two measurement time series will have slightly different sets of excursions is low. In all other cases, i.e., shifts in time (loose coupling) between two probe sequences, different probing intervals, etc., the method that accounts for gaps performs better. The relative difference between results was insubstantial in most cases, which means that both methods can be used almost equally well to create indices from active measurement time series. A particular need for the practical use of indices may prefer one method over the other. The review of the practical uses of generated indices is beyond the scope of this section.

7.2 Topological Ramifications of Active Measurement

Traditional research in on-demand topological solutions is gathered around the two main clusters, – wireless ad-hoc networks and fixed overlay networks. Both are very different in nature, but they both deal with the same problem which is to create a topology out of an arbitrary set of nodes. This section considers the case of an arbitrary set of mixed-technology nodes which are to be joined in a topology based on end-to-end delay

measurements among nodes. The core of the proposal is topology inference based on triangular inequality of end-to-end delay which is finalized in form of an algorithm that allows for efficient detection of a logical topology of a network with no initial topology. The algorithm is scalable and could be a practical solution for many scenarios involving community services created on-demand and intended for a short lifespan.

7.2.1 Problem Statement

The number of global community networks is constantly growing. Skype, PlanetLab, Skitter, etc. are only a few well known names in today's global network. Each of them uses some form of *network embedding*, – the term used to describe translation of end-to-end delay measurements into *Euclidian space*.

Topology inference in large-scale community networks becomes difficult when the number of nodes in the network is large and performing $O(n^2)$ measurements is not practical. Additionally, having all-to-all delay samples does not always solve the topology inference problem which seeks to detect logical topology of a community network rather than Euclidean translation of node positions. In fact, authors of this section are not aware of an algorithm that would be able to infer topology based on raw end-to-end measurements among all individual nodes in a community network.

Network embedding is also becoming an important network service demanded by global community networks. *Vivaldi* [24] and *Meridian* [48] are the two most popular models used for such purpose. Vivaldi is slightly inferior to Meridian in precision which comes from the difference of approaches these two models use for global-scale network embedding.

The problem with traditional network embedding is the incremental nature of the process. At any point of time in the lifespan of a community network, a new node arrives to be integrated into the topology. This triggers the process of *neighbour discovery* which is the source of major differences between Vivaldi and Meridian. Both models use a number of reference nodes selected from already existing topology to triangulate the newly arrived member and finally position it at some coordinates in Euclidean space. These coordinates can be used to find a communication partner which normally would be one of the nodes located nearby. While creating topologies based on Euclidean coordinates is a trivial task, the problem is that Euclidian coordinates do not necessarily represent the logical topology on the network.

This section proposes a simplified algorithm for topological inference in large-scale community networks without attributing Euclidean coordinates to each node. The target of this research is to reveal logical topology with as little measurement overhead as possible. Results are verified on a random physical topology generated by the GT-ITM topology generator [5].

7.2.2 Delay Properties of Community Networks

Most research in network embedding is conducted on sets of end-to-end delay measurements obtained from existing global community networks. Popular delay data sets are *p2psim* [15], *DS²* [50], and *PlanetLab*. Each data set contains delay measurements in meshes containing from several hundred to several thousand nodes. However, no connectivity graphs are provided with these data sets which means there is no way to verify how well results of network embedding match actual geographical locations of nodes and underlying physical structure of the network. This section discusses some of the issues related to delay properties and physical topologies.

Close proximity in terms of physical connectivity is the main source of error in neighbor discovery in both Vivaldi and Meridian models. In case of a triangle containing nodes *A*, *B*, and *C*, and $d(AB)$ signifying the delay between nodes *A* and *B*, the less the physical distance between the nodes, the more frequently $d(AC) = d(AB) + d(BC)$. Literally, node *B* would often be in the middle of the only possible path from *A* to *C*, i.e. *A*, *B* and *C* would be on a straight line.

For example, all four connectivity cases in Figure 7.14 would result in identical delay data sets. As a rule of thumb, the farther the nodes in neighbour discovery are from each other the more reliable is the result of

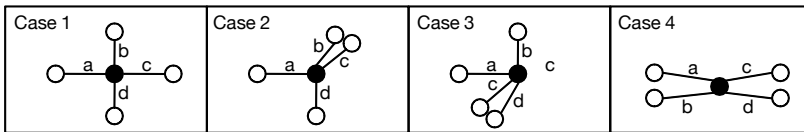


Figure 7.14 Given that legs *a*, *b*, *c*, and *d* are equal in on-way-delay, all the four cases above will produce perfectly identical results.

triangular delay measurements. Additionally, neighbour detection with reference nodes farther apart makes more sense when the target is a logical topology of the network rather than Euclidean coordinates.

Consider the three cases in Figure 7.15. In all three cases one-way delay measurements are conducted from two reference nodes marked *R1* and *R2*. The reason two nodes are used for the measurement is so that a cross-reference of network-wide delay distribution can be obtained. This is also a small-scale version of the proposed algorithm that will be proposed later in the section.

Delay maps for each of the three cases are created using unit distance from *R1* as horizontal and unit distance from *R2* as vertical coordinate in the chart. Each leg in all topologies is exactly one unit of delay long. All topologies contain a fork in connectivity.

The visualizations in Figure 7.15 should be interpreted as follows. When both reference nodes are placed on the same branch of the tree, the fork is not present in the delay map. In this case cross-references in delay measurements have no significance. Cases 2 and 3, however, are correctly reflected in the delay space. More than that, all bridges between branches purposely placed in Case 3 are translated as bridges in delay space as well. This feature is exploited in the proposed algorithm in a more sophisticated form. It should be noted that delay maps are significant only when reference nodes are placed on different branches in the tree. In the proposed algorithm this element is specifically ensured.

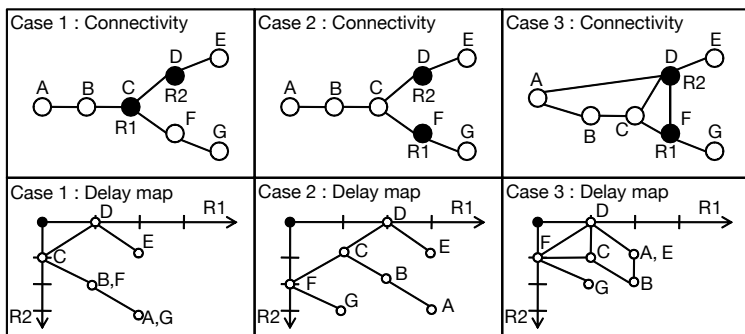


Figure 7.15 Three examples of how physical connectivity can be translated into delay maps by using two reference nodes located at various points within the network.

7.2.3 Topology Inference Algorithm

The fact that delay-based view of a community network is different from its geographical view offers room for *topology inference*. Based on delay jumps and perturbations as described in [45], it can be assumed that delay-based map of a community network should consist of a number of clusters separated by relatively long intra-AS spans. This section makes a number of such assumptions and proposes an algorithm that exploits them for the purpose of topological inference.

In the analysis of end-to-end delays in large-scale community networks the following assumptions are in place:

- distribution of delay is mostly presented in the two extremes, – either very long or very short delays;
- many short delays form clusters which are also located nearby in geographical location;
- long delays lead to major crossroads in the community network, which is not the case with short delays;
- virtual centres of community networks are located equally far from all edges as viewed by end-to-end delay.

Apart from the above assumptions, it is also important to minimize differences in inferred topology depending on which node in a community network starts the process of inference. Contemporary community networks are normally created by an incremental process, which enables continuous expansion of the network as new members arrive. This is true for both PlanetLab and Skype, – the two largest community networks in the world today. Incremental process allows for certain freedom in topological inference, since these networks started from a relatively small number of nodes and kept growing over the years.

It can be foreseen, however, that NGN networks will bring about a new way of creating community networks by giving full capabilities to a large community of mobile users. In this new environment many community networks will have to be created on-the-fly to exist only for a short period of time. Topological inference in such community networks will be the issue of utmost importance.

Additionally, the above futuristic scenario will impose limitations on the performance of a method used to infer topologies of newly created community networks. Literally, there will be no time or processing capacity for $O(2^n)$ end-to-end measurements to define how far members of

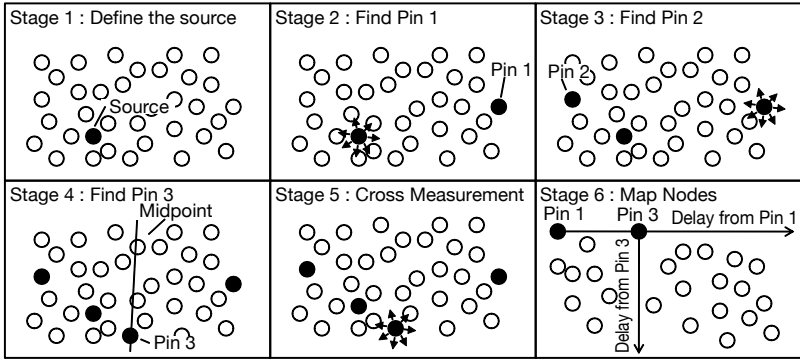


Figure 7.16 Stepwise process of topology inference.

a given community network are located from each other. The number of measurements should be minimal yet offer a solid delay-based view of the network. In the best case, $O(mn)$ complexity is expected, where m is the number of times measurements from a single node to all other nodes is performed, n being the number of nodes and $m < n$. Smaller m values result in faster results of topology inference and are good especially for on-demand communities with short lifespan.

Figure 7.16 displays the overall process required to complete topological inference in an arbitrary size community network. Most steps involve end-to-end measurements from a single node to all the other nodes in the community network. Each step is performed as follows.

Stage 1: “Define the source” is the initial step for any community network. Although in this section the source is selected by random selection, in real like many scenarios are possible. Such as, in content delivery networks this could be the node that has the content and needs to distribute it among community members. Since this section selects the source randomly, it can satisfy any scenario used in practice.

Stage 2: “Find Pin 1” involves end-to-end measurements originating from the source and terminated at all other nodes in the community network. Therefore, with n nodes in the network, $n - 1$ measurements are performed. The purpose of the measurement is to find end-to-end delay between the source and another node. In real networks, round-trip delay measurements can be successfully used instead of one-way delay. Since

this section uses simulated topologies, it is easier to operate with one-way delays which are calculated as the sum of delays of all individual hops on each end-to-end path. End-to-end paths, in turn, are calculated using the shortest path algorithm.

Stage 3: “Find Pin 2” repeats end-to-end measurements, this time using Pin 1 as the source. The purpose of this set of measurements is to find the most distant node from the Pin 1, thus, defining the boundaries of the entire community network. At the end of Stage 2, the set of delay values represents one-dimensional view of delay distribution of the entire community network. To add the second dimension additional steps are required for further details.

Stage 4: “Find Pin 3” is used to create 2-dimensional delay map of the community network. To do this, a mid-point between Pin 1 and Pin 2 is selected as node with the least difference between delays from Pin 1 and Pin 2, i.e. the selected node would be virtually in the middle between Pin 1 and Pin 2. In the result, Pin 3 is obtained and will be used to create 2D delay image of the network.

Stage 5: “Cross Measurement” is the fourth set of measurements originating from Pin 3. This set of delays allows for cross-reference with delays originating from Pin 1 and Pin 2.

Stage 6: “Mapping” is performed by using delays originated from Pin 1 and Pin 3 as coordinates in the delay-based map. Pin 1 is always zero on horizontal axis but may not be zero on the vertical. Pin 3 is the zero on the vertical axis but may be placed at a positive value on the horizontal axis. In the result of this operation, each node in the community network is attributed with 2D coordinates based on end-to-end delay.

From the extensive search for a *topology generator*, *GT-ITM* appears to be the only topology generator that can create multiple-tier topologies. Its closest competitor would be the *BRITE* topology generator but the latter generates only flat topologies which are meaningless for delay-based topological inference performed in this section. All the results below are based on a randomly generated multiple-tier topology with 100 nodes.

Raw topology generated by *GT-ITM* is displayed in Figure 7.17. This topology will be used in this section to perform measurement-based topological analysis.

Before the results of the proposed algorithm are presented it is impor-

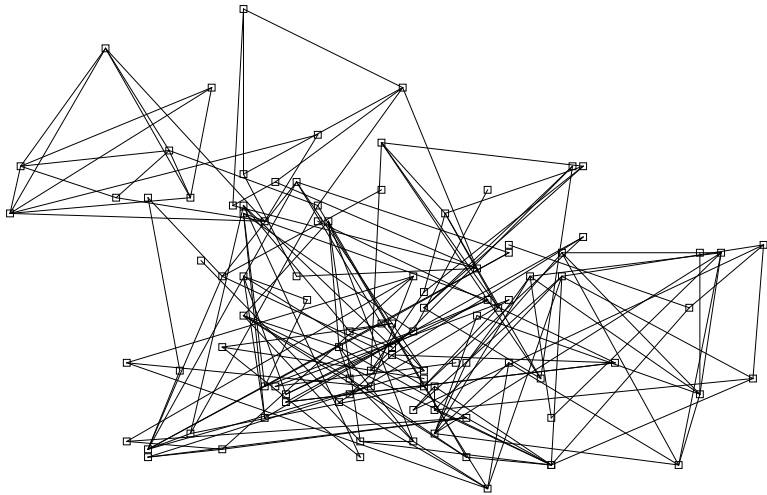


Figure 7.17 Raw topology plotted directly from GT-ITM multi-tier topology generator.

tant to view the underlying topology. Since a randomly generated topology is used to obtain simulation results, actual delays can be analyzed based on the actual graph. Figure 7.18 is used to visualize logical topology of the randomly generated graph as viewed by one-way delay. The centre of the delay-based view was selected arbitrarily by the process of minimizing variance of hop count of end-to-end shortest paths from the central node to all other nodes in the network while also making it closer to the average hop count among all shortest paths in the network. This way the central node is literally the closest node to the centre of all delays in the community network.

An important feature of the delay map in Figure 7.18 is the use of actual delays as Euclidean distances in the visualization. Actual delays are marked by thicker lines and are true only as long as they stay on the same branch.

The visualization in Figure 7.18 is constructed as follows. After the central node is defined, the visual is constructed incrementally by adding neighbours for each node starting from the central node. Neighbours of the central node naturally form the main branches. All central branches split the entire area in sectors of equal angles. Every following branches

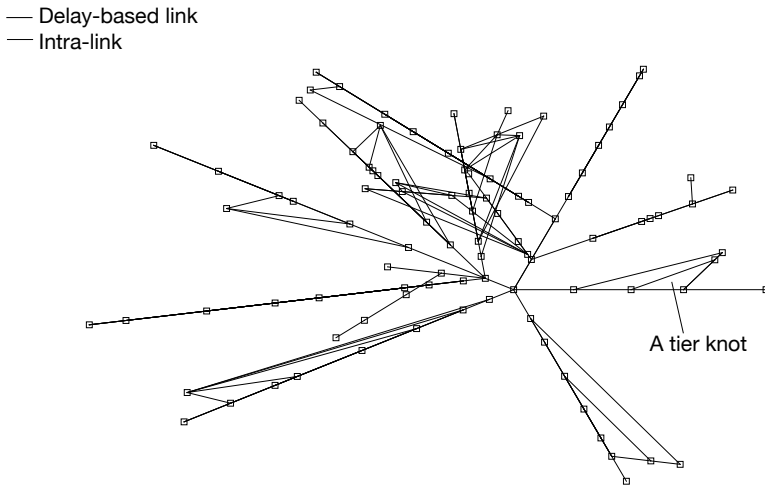


Figure 7.18 Delay-based visualization of network structure from the topology generated by GT-ITM.

equally split smaller angle ranges in such a way that branching would continue in the outward direction from the centre, thus, creating an easily digestible visual image.

Thin lines in Figure 7.18 represent intra-domain connections that normally stand for meshes within a network tier or intra-AS connections.

They are not plotted during the branching process but are added to the visualization at the end. However, they play an important role in understanding topological properties of multiple-tier network graphs. Judging from Figure 7.18, intra-links create only local irregularities, leaving the overall branching structure intact. When links are connecting nodes on the same tier, the structure could be referred to as a tier knot or a tier mesh.

Given that the visualization in Figure 7.18 is based on the actual graph, it will be important later when topology inference results are discussed. One major difference between the delay map in Figure 7.18 and topology inference results is the fact that topology inference results are based only on relative one-way distances and should not be interpreted as Euclidean translations.

The following explanation will base on three different approaches

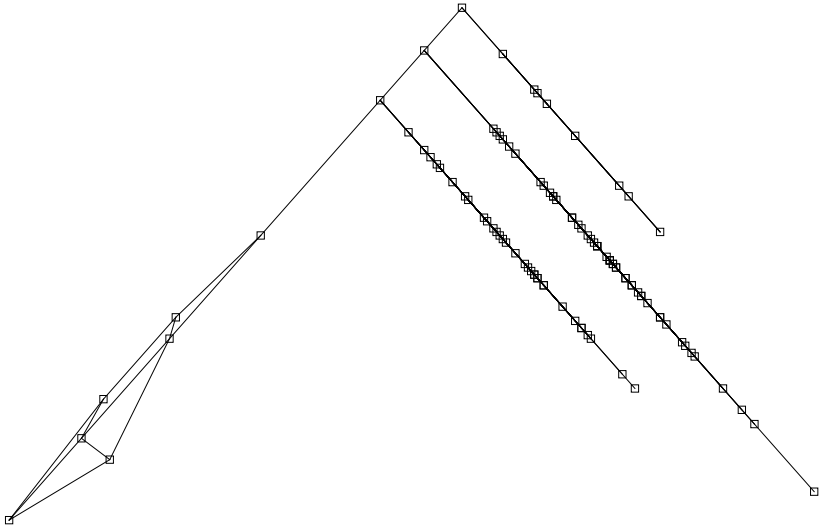


Figure 7.19 Case 1 of measurement-based topology perception.

used in defining coordinates of individual nodes based on end-to-end delay. The results will come from using the proposed algorithm on the same randomly generated multiple-tier network graph.

The Case 1 in Figure 7.19 of the delay map offers a clear view of the network and its topological structure. The very upmost node is the Pin 3, i.e. the perfect centre of the community network as viewed from both Pin 1 and Pin 2. It may not be the perfect centre of the horizontal axis, however, given that the geographical distribution of nodes is irregular. Since the delay map is created from the virtual left and the virtual centre of the network all logical forks in topology are easily revealed as visual forks in the map. This has a physical explanation related to connectivity. When there is a connectivity fork in the underlying physical networks, distances on each side of the fork change independently from each other when translated by delay from the edges of the community network. This independence in change further away from a connectivity fork directly translates into a distance in the delay map. How forks in the networks translate into delay maps was already discussed earlier in this section on a simpler case.

Topological mapping in Case 1 in Figure 7.19 does not detect all topo-

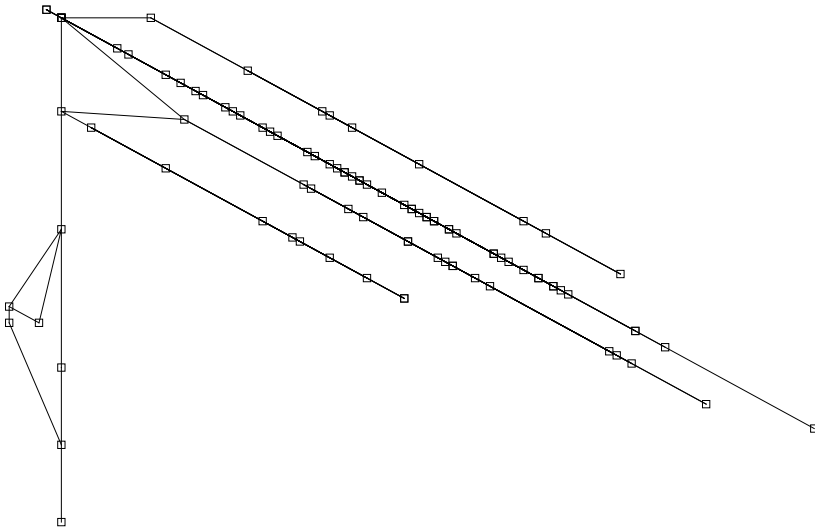


Figure 7.20 Case 2 of measurement-based topology perception.

logical forks. In fact, the more points are used to create delay cross-references, the more forks can be detected in the resulting delay map. Cases 2 and 3 in Figure 7.19 are the examples of such precision boost coming from adding more reference points in the network.

To create delay maps in Case 2 as per Figure 7.20, instead of using a single node in the centre of the network, two nodes at 0.33 and 0.66 of the delay distance between Pin 1 and Pin 2 were used. Accounting for geographical irregularity, the closest nodes to the above positions were selected. In the end, vertical coordinate was obtained as an average of the delay to a node from the two central points. This resulted in more forks revealed through the delay map.

Finally, Case 3 in Figure 7.21 uses 3 central reference points selected at equal intervals between Pin 1 and Pin 2, which reveals yet more topological detail.

It is safe to assume that the more reference points are used to cross-reference the delay map, the more topological details are revealed. There is, however, a physical limit to this process. From many simulations conducted using multiple reference points it was concluded that past 50% of all nodes used as reference points the topological view of the network

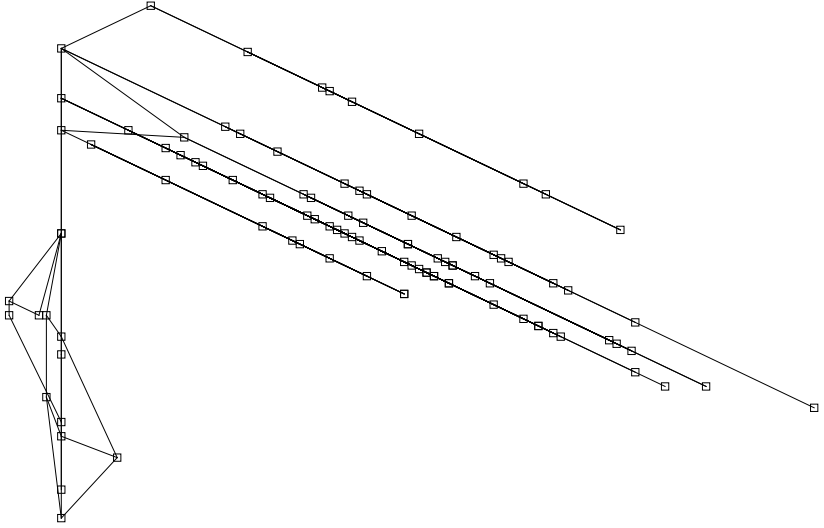


Figure 7.21 Case 3 of measurement-based topology perception.

degrades and starts to resemble the original geographical mesh. Ideally, if all nodes cross-reference each other, no topological information can be inferred from the results, as was already mentioned in the introduction.

It is important to note that Case 3 in Figure 7.21 is the closest view to the one presented earlier in Figure 7.18 given the number of branches from the virtual centre of the community network. The only difference is that delays in Figure 7.21 are relative to reference points selected in the network to perform delay cross-reference. However, the target of this section is not to pinpoint Euclidean coordinates of individual nodes but to detect details of logical topology of the entire community network. Based on Figure 7.21 it is clear that the target has been reached.

7.2.4 Discussion of Measurement and Analysis Methodology

This section proposed to use a simple algorithm based on end-to-end delay measurements to infer logical topology in large-scale community networks. The key assumption of the proposed algorithm is that all networks

have multiple-tiers and when a community network in question is global, end-to-end paths are destined to move up and down in the tier depth.

Another basic assumption is concerned with the precision of the proposed algorithm. The perfect precision is attainable only when there are as many reference points in the network as there are main branches in logical topology. However, even a few reference points are capable to creating visual differences when translated into cross-referenced delay maps.

Main targets of the proposed algorithm are community networks created on-demand with a relatively short lifespan. The opposite case would be a community network started from a single node incrementally adding more nodes as new members join the network. In the latter case, the proposed algorithm should be refined since new members normally form a long-term relationship and there is time to perform a more time-consuming algorithm of topological inference. Vivaldi and Meridian models were created specifically for such incremental processes. With short-term global community networks this approach is not feasible, especially given that the emergence of NGN services will add a large community of mobile users that will be difficult to locate geographically.

The proposed algorithm assumes that higher delay distances in any global network occur from East to West or visa versa. This is true for contemporary networks which form a virtual belt around the globe. Based on this assumption, cross-referencing of delays is performed by using multiple horizontal points rather than trying to split the vertical scale. Although the concepts of horizontal and vertical bears little meaning when translated to network-wide connectivity, in practice this seems to be the legitimate choice of words.

In several cases of delay-based topological inference it was established that the more mid-network reference points are used for cross-referencing end-to-end delays the more topological detail is detected in the resulting mapping. Although practical use cases of the proposed algorithm are not included in this section, it is fair to assume that depending on each particular community network the number of cross-reference points should vary based on the required level of detail and resources allocated to topological inference.

Bibliography

- [1] 3gpp. available at: <http://www.3gpp.org/>.
- [2] ETSI (European Telecommunications Standards Institute. Available at: <http://www.etsi.org/WebSite/homepage.aspx>.
- [3] ETSI TISPAN WG3. Available at: http://portal.etsi.org/tispan/WG3_Tor.asp.
- [4] GSC (Global Standards Collaboration). Available at: <http://www.gsc.etsi.org/>.
- [5] GT-ITM: Georgia Tech internetwork topology generator. <http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html>.
- [6] IANA (Internet Assigned Number Authority). Available at: <http://www.iana.org/>.
- [7] IETF (Internet Engineering Task Force). Available at: <http://www.ietf.org/>.
- [8] IMS Forum. Available at: <http://www.imsforum.org/>.
- [9] Internet Traffic Archive (ITA). Available at: <http://ita.ee.lbl.gov/html/traces.html>.
- [10] ITU-T (International Telecommunication Union, Telecommunication Standardization Sector). Available at: <http://www.itu.int/ITU-T/>.
- [11] ITU-T SG12. Available at: <http://www.itu.int/ITU-T/studygroups/com12/index.asp>.
- [12] MAWI working group traffic archive. Available at: <http://tracer.csl.sony.co.jp/mawi/>.
- [13] MRTG: Multiple Router Traffic Grapher. Available at: <http://oss.oetiker.ch/mrtg>.
- [14] ntop. Available at: <http://www.ntop.org>.
- [15] p2psim. <http://www.pdos.lcs.mit.edu/p2psim/>.
- [16] Renater's QoS Metrics Box. Available at: http://pasillo.renater.fr/metrologie/get_qosmetrics_results.php.
- [17] TTM: Test Traffic Measurements service. Available at: <http://www.ripe.net/ttm>.
- [18] End-user multimedia QoS categories. Technical Report ITU-T Recommendation G.1010, November 2001.

- [19] Management of performance measurement for NGN. Technical Report ITU-T Draft Recommendation Y.mpm, July 2006.
- [20] Bengt Ahlgren, Mats Bjorkman, and Bob Melander. Network probing using packet trains. Swedish Institute of Computer Science, 1999.
- [21] Michele Basseville and Igor V. Nikiforov. *Detection of abrupt changes – theory and application*. Prentice-Hall, 1993.
- [22] J. Bolot. Characterizing end-to-end packet delay and loss in the internet. In *Journal of High Speed Networks*, volume 2(3), pages 305–323, 1993.
- [23] B. Claise. RFC 3954: Cisco Systems NetFlow Services Export Version 9. Technical report, October 2004.
- [24] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM Computer Communications Review*, volume 34.4, pages 15–26, 2004.
- [25] L. Deri and S. Suin. Effective traffic measurement using ntop. *Communications Magazine, IEEE*, 38(5):138–143, May 2000.
- [26] Luca Deri and Stefano Suin. Improving network security using ntop. In *Proc. Third International Workshop on the Recent Advances in Intrusion Detection (RAID)*, 2000.
- [27] Constantinos Dovrolis, R.S. Prasad, M. Murray, and K.C. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. In *IEEE Network*, volume 17, November 2003.
- [28] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure? In *INFOCOM*, pages 905–914, Anchorage, AK, USA, April 2001. IEEE.
- [29] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions in Networking*, 12:963–977, 2004.
- [30] Nangning Hu and Peter Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, 21(6):879–894, August 2003.
- [31] V. Jacobson. Pathchar: a tool to infer characteristics of Internet paths. Available at: <ftp://ftp.ee.lbl.gov/pathchar/>, April 1997.
- [32] M. Jain and C. Dovrolis. Pathload: a measurement tool for end-to-end available bandwidth. In *Passive and Active Measurements (PAM) Workshop*, March 2002.

- [33] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. In *Transactions on Networking*, volume 11 no. 4, pages 537–549, Pittsburgh, Pennsylvania, USA, August 2003. IEEE/ ACM.
- [34] Kevin Lai and Mary Baker. Measuring bandwidth. In *IEEE INFOCOM*, volume 1, pages 235–245, 1999.
- [35] Kevin Lai and Mary Baker. Measuring link bandwidths using a deterministic model of packet delay. In *SIGCOMM*, pages 283–294, 2000.
- [36] Kevin Lai and Mary Baker. Nettimer: a tool for measuring bottleneck link bandwidth. In *Proc. 3rd USENIX Symposium on Internet Technologies and Systems*, pages 123–134, San Francisco, CA, USA, March 2001.
- [37] Mark Last, Abraham Kandel, and Horst Bunke. *Data mining in time series databases*, volume 57. World Scientific, 2004.
- [38] A.J. McGregor and H-W Braun. Automated event detection for active measurement systems. In *Passive and Active Measurements*, pages 23–32, April 2001.
- [39] A. Pasztor and D. Veitch. A precision infrastructure for active probing. In *Passive and Active Measurements (PAM)*, April 2001.
- [40] Attila Pasztor and Darryl Veitch. On the scope of end-to-end probing methods. *IEEE Communications Letters*, 6:509–511, 2002.
- [41] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for ip performance metrics. RFC 2330.
- [42] Richard J. Povinelli. *Time series data mining: identifying temporal patterns for characterization and prediction of time series events*. PhD thesis, Marquette University, Dec. 1999.
- [43] V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Workshop (PAM)*, pages SLAC-PUB-9732, March 2003.
- [44] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *3rd ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 39–44, October 2003.
- [45] L. Tang, H. Zhang, J. Li, and Y. Li. End-to-end delay behavior in the Internet. In *14th IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS)*, pages 373–382, 2006.
- [46] J. Mahdavi V. Paxson, G. Almes and M. Mathis. Framework for IP Performance Metrics. RFC 2330, May 1998.

- [47] S. Waldbusser. RFC 1757: Remote network monitoring management information base. Technical report, February 1995.
- [48] B. Wong, A. Slivkins, and E.G. Sirer. Meridian: a lightweight network location service without virtual coordinates. In *Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications*, pages 85–96, 2005.
- [49] V. Paxson Y. Zhang, N.G. Duffield and S. Shenker. On the constancy of Internet path properties. In *ACM SIGCOMM Internet Measurement Workshop*, pages 197–211, November 2001.
- [50] B. Zhang, T.E. Ng, A. Nandi, R. Riedi, P. Druschel, and G. Wang. Measurement based analysis, modeling, and synthesis of the internet delay space. In *ACM SIGCOMM Conference on Internet Measurement*, pages 85–98, 2006.
- [51] Marat Zhanikeev and Yoshiaki Tanaka. Issues with using real packet traces in simulated environments. In *IEICE Technical Report on Telecommunication Management, No.TM2006-67*, pages 35–40, March 2007.

Index

3GPP, 4
5-tuple flow, 46

A

abrupt change, 123
abrupt change detection, 123, 138
Abstract Syntax Notation One, 36
access, 36
ACK packet, 56
active measurement, 31, 90, 94
active measurement method, 119
active measurement technology, 93
active monitoring, 87
active probe, 97
adaptive, 119
agent, 41
alarm, 38
analysis, 48
annual meetings, 9
anomalous condition, 41
API, 60
application layer metrics, 20
application layer performance, 26
application layer QoS, 24
application server, 34, 155
artifact, 176
ASN.1, 36
asymmetric route, 101
asynchronous, 41
asynchronous alarm, 89
autorefresh, 54
available bandwidth, 100, 120, 139

B

background traffic, 100
back-to-back, 107
billing, 90
black box, 161
bottleneck, 107, 121
bottleneck bandwidth, 119

bottleneck capacity, 100
box, 153
breakpoint, 114
BRITE, 196
broadband billing, 91
browser, 33
bulk of data, 5
bulk transfer capacity, 100

C

CAIDA, 101
certainty rate, 132
Cisco, 73
client-server model, 33
collector, 48
command line, 57
communication pattern, 39
communication protocols, 2
congestion, 85
connection provider network, 23
connectivity, 99, 158
context, 153
continuous, 32, 120
continuous monitoring, 30, 95
control plane, 1
convergence, 155
counter, 38, 94
counter wrapping, 48
CPN, 23
CPU load, 66
cross traffic, 97, 147

D

data analysis, 88
database logging, 68
data collection, 88
data mining, 117, 176
data plot, 62
data rate, 85
datatype, 36

de-facto, 10
 de-facto standard, 8
 de-facto standards, 11
 design pattern, 53
 detection, 94
 device temperature, 66
 directionality, 116
 divisions, 6
 downtime, 68
 DS^2 , 192
 dynamic monitoring, 90

E

effectiveness, 119
 end-to-end, 95, 107
 end-to-end capacity, 140
 end-to-end jitter, 102
 end-to-end measurement, 120
 end-to-end path, 23, 25
 end-to-end performance, 2, 95
 end-to-end performance objectives, 14
 end-to-end QoS, 29
 enterprise, 36
 error rate, 128
 error resilience, 103
 ETSI, 9
 ETSI TISPAN WG3, 10
 Euclidian space, 191
 European Telecommunications
 Standards Institute, 9
 excursion, 178
 export, 48

F

fault management, 94
 feedback, 120
 filtering, 79
 flag, 38
 flash crowd, 89
 flat rate, 91
 flow level, 5
 flow statistics, 79
 flow timeout, 46
 forum, 3, 9

freeware, 66
 front end, 52, 64

G

G.1010, 21
 gamers, 167
 gap, 179
 gap-based, 139, 143, 146
 gauge, 38
 GD library, 60
 global collaboration, 11
 global initiative, 4
 global network management, 5
 global prospective, 6
 global scale, 1
 global standardization, 12
 Global Standards Collaboration, 8
 granularity, 10
 graphical plot, 65
 GSC, 8
 GSC10, 9, 29
 GT-ITM, 196

H

header, 45
 header wrapping, 45
 heavy hitter, 87
 heterogeneous, 3
 histogram, 126
 hop-by-hop measurement, 120
 horizontal performance degradation,
 21
 HTTP protocol, 55

I

IANA, 35
 ICMP, 106
 ICMP Time Exceeded, 106, 111
 IETF, 10, 97
 IETF Standard 62, 32
 implementation, 52
 IMS, 3
 incoming traffic, 65
 index, 181

inference, 97, 119, 153
integer, 36
interactivity, 61
interarrival, 122
interface, 65, 94
interface abstraction, 1
interference, 122, 128
International Telecommunication
 Union, 7
Internet Assigned Number Authority,
 35
Internet drafts, 11
Internet Engineering Task Force, 10
Internet service provider, 31
Internet Standard, 32
intrusiveness, 120
IPFIX, 156
IP flow, 46
IP multimedia subsystem, 3
IP Performance Metrics, 11, 97
IPPM, 11, 97, 155
ISP, 31
ITU-T, 7
ITU-T FG NGN, 23

J

jitter, 146, 147

K

kernel density, 134

L

latency, 147
layer 3, 45
libpcap, 76
lightweight, 57, 147
load, 58
loose coupling, 116, 178
loosely coupled, 54

M

managed device, 41
management information base, 32
management issues, 4

management of measurement results,
 174
management of NGN network, 4
management of performance
 measurement, 174
match ratio, 189
measure, 2
measurement box, 159
measurement method, 99
measurement methodology, 103
measurement projects, 158
measurement results, 174
measurement time series, 116, 176
Meridian, 191
methodology, 119
metric, 94
MIB, 32, 94
MIB file, 36
MIB library, 60
modular structure, 41
monitoring architecture, 48
monitoring box, 30
monitoring objectives, 90
monitoring scenario, 53
monitoring target, 90
monitoring tool, 52
MRTG, 52, 64
MRTG-XTRA, 66
multimedia, 3, 4, 28
multimedia applications, 28
multimodal, 109
multi-party collaboration, 4
multi-point, 88

N

near real time, 5, 49
neighbour discovery, 191
NetAgent, 169
NetFlow, 44, 46, 52, 59, 66, 72, 79, 86
NetFlow client, 73
NetFlow compliance, 73
NetFlow interface, 66
NetFlow-like, 73
NetFlow meter, 82

- network administrator, 64
- network design, 2
- network device, 32, 34, 53, 87
- network embedding, 191
- network equipment, 94
- network management, 84
- network management system, 41
- network operation centre, 94
- network performance, 20, 26
- network performance data, 14
- network performance metrics, 99
- network performance standardization, 6
- NGN, 1, 2, 34, 174
- NGN application server, 23
- NGN-compatible, 23
- NGN focus group, 8
- NGN framework, 6
- NGNMFG, 8
- NGN standardization, 1
- NGN view, 22
- NMS, 41
- NOC, 94
- nominal performance, 21
- normative documents, 8
- notation, 36
- ntop, 73, 76, 77, 87

O

- object, 37
- object ID, 35
- octet string, 36
- offline analysis, 89
- offline mode, 94
- OID, 35
- one-way delay, 100, 141
- online analysis, 90
- online processing, 94
- operating system, 57
- organizations, 5
- outgoing traffic, 65

P

- p2psim, 192
- packet analysis, 76
- packet dispersion, 113
- packet gap, 139
- packet header, 45, 46
- packet level, 5
- packet-pair, 102, 109
- packet-pair dispersion, 147
- packet-pair property, 99, 107, 122
- packet size, 124, 147
- packet size dynamics, 129
- packet train, 113, 129, 141
- parent node, 36
- passive, 4
- passive measurement, 31, 51
- passive performance measurement, 4
- Pathload, 113
- pattern, 180
- PDU, 39
- peer review, 11
- performance, 8
- performance critical, 48
- performance degradation, 21, 22
- performance dynamics, 90
- performance management, 4
- performance measurement, 4
- performance measurement management, 12
- performance measurements, 12
- performance metric, 157, 175
- performance metrics, 12, 28, 31, 98
- performance problems, 5
- performance state, 90
- performance statistics, 68
- phase space, 176
- physical distance, 101
- piggyback, 56, 111
- PlanetLab, 192
- poll, 32
- polling, 94
- practical implementation, 12, 173
- practical task, 84
- precision, 153

- preliminary, 8
- private, 36
- probability distribution model, 177
- probe, 97
- probe design, 99, 111, 158
- probe stream, 140
- probe structure, 102
- probe train, 113
- probing interval, 124
- probing method, 140, 158
- programmable data unit, 39
- protocol stack, 20, 34
- PRTG, 66

Q

- QoE requirements, 1
- QoS, 1, 95, 147
- QoS classes, 28
- QoS guarantees, 4
- QoSmetrics, 169
- QoSMetrics Box, 166
- QoS requirements, 1
- quality of experience, 26
- quality of service, 8, 26

R

- rate-based, 139, 143, 146
- raw data, 55
- read-write, 40
- real time, 5
- real-time analysis, 88
- realtime OS, 154
- recovery, 94
- reference point, 109
- reliable end-to-end QoS, 24
- remote network device, 72
- remote terminal, 57
- Renater network, 166
- Reply to Draft, 12
- Request for Comments, 10
- resource, 79
- RFC, 10
- RFC 1157, 94
- RFC 2330, 97, 99

- RMON MIB, 86
- RMON specifications, 86
- round trip time, 101
- routing, 116
- RRDtool, 66
- RTT, 101

S

- sampled counting, 91
- sensor, 67
- separation, 3
- service-oriented, 95
- sFlow, 82
- SG12, 7, 8, 14
- SGC9, 9
- SGC10, 9, 23
- shared topology, 118
- Simple Network Management Protocol, 11
- single-packet, 102, 109
- Skitter, 101
- SNMP, 11, 32, 52, 59, 64, 86, 94
- SNMP meter, 86
- SNMP standardization process, 44
- social phenomena, 89
- software only box, 169
- standalone installation, 72
- standard, 119
- Standard 62, 43, 44
- standardization bodies, 6
- standardization documents, 5
- standardization process, 1
- standardization processes, 5
- statistical processing, 124
- statistics, 4, 95
- status, 36
- storage, 48
- study group, 7
- symbolic name, 36
- synchronization, 122
- syntax, 36

T

TCP, 45
 TCP benchmarking, 175
 tcpdump, 45
 TE, 24
 temporal pattern, 181
 terminal equipment, 24
 textual command, 39
 threshold rule, 124
 throughput, 120
 tightly coupled, 54, 56
 time gap, 141
 timeline, 12, 43
 time scale, 175
 time series, 117, 174, 176
 TISPAN, 10
 top, 73
 topology generator, 196
 topology inference, 194
 traditional traffic flow, 82
 traditional view, 20
 traffic, 44
 traffic analysis tool, 48
 traffic collection, 44
 traffic component, 70
 traffic control, 3
 traffic dump, 5
 traffic flow, 46, 62, 73
 traffic load, 64
 traffic phenomena, 89
 traffic statistics, 65
 traffic summary, 79
 traffic throughput, 65
 transport layer, 24
 transport layer performance, 22
 transport layer QoS, 20
 transport plane, 1
 transport protocol, 45
 TR-NGN-QoS, 23
 TTL, 111
 TTM Box, 159
 tuple, 46
 turning point, 140

U

UDP, 45
 unidirectional flow, 46
 unique realm, 6
 UNIX tool, 73
 usecase, 64
 user interface, 54, 56
 utilization, 95
 utilization rate, 127
 utilization variation, 149

V

validation, 132
 validity, 120
 validity problem, 120, 128
 variation, 147
 vertical performance degradation, 21
 visual appeal, 62
 Vivaldi, 191

W

Web 2.0, 61
 web access, 10
 web application, 77
 web interface, 61, 77
 web page, 65
 web server, 33
 white noise, 124
 window size, 124
 working groups, 6
 write-only, 40

Y

Y.mpm, 21