Chapter 2
Methods and Techniques for Measurements in the Internet

An in-depth understanding of the Internet traffic mix is of paramount importance for network management tasks, such as optimizing the underlying infrastructure for emerging applications. However, the Internet traffic mix changes over time and is very complex when it comes to measurements and classification techniques. Its traffic profile also changes depending on the measurement points [1]. Although there is no de facto way to perform measurements on the Internet, there are good IETF documents that highlight some important elements in this context [2, 3]. However, as the Internet evolves at a fast pace, it is hard to have a general measurement framework that covers all aspects of the future Internet [4]. One clear example is the recent rise of virtualization technologies in computer networking. Virtualization techniques are bringing a new set of challenges from the point of view of the measurement process (cf. Sect. 2.5).

Figure 2.1 presents an overview of the measurement process where one can identify three major steps, namely, capturing, processing, and exporting/recording, at packet level. Figure 2.2 depicts a detailed view of a proposed measurement reference architecture focusing on non-intrusive methods, developed by Tanja Zseby in her PhD thesis [5]. Each abstract and concrete component of this reference model plays an important role in the measurement process and analysis, as follows:

1. *Measurement process* encompasses all the functions needed to perform the network measurements.
2. *Observation point* is where the real traffic capture occurs at packet level.
3. *Measurement controller* configures the tasks of the measurement process.
4. *Packet and flow records* are standardized structures to report packet or flow information.
5. *Collecting process* is the component responsible for receiving packet or flow records from the measuring process.
6. *Single point (SP)/multipoint (MP)/inter-domain (ID) analyses* are the components responsible for processing either packets or flow records to extract useful network management information.

S. Fernandes, *Performance Evaluation for Network Services, Systems and Protocols*, DOI 10.1007/978-3-319-54521-9_2

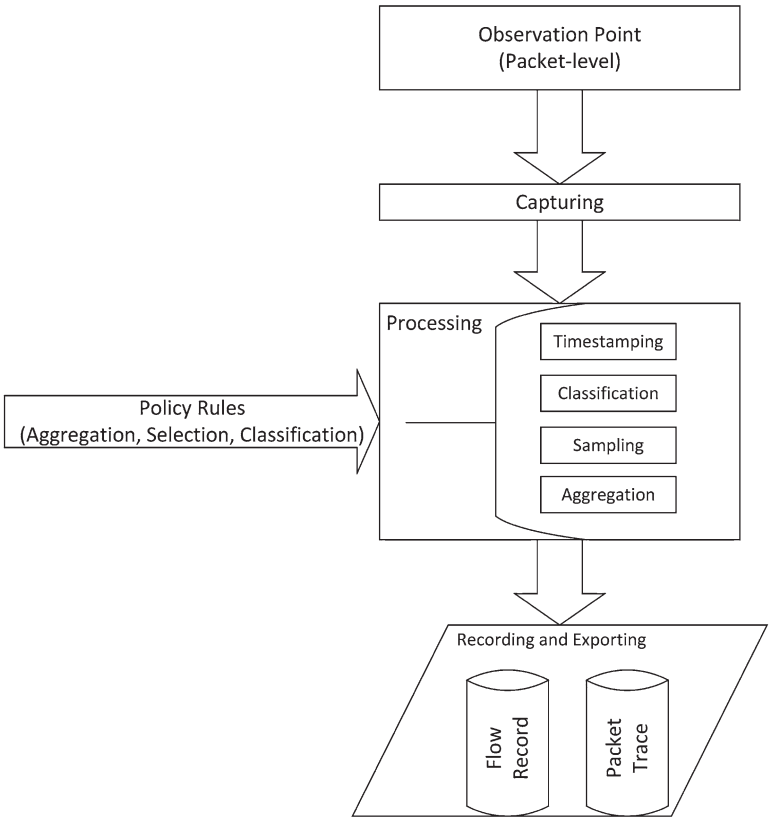

Fig. 2.1 An overview of the measurement process [5]

The measurement process per se deserves a detailed explanation since it is one of the most important elements in the measurement architecture. This process can encompass several other functions, beyond the capture process, such as time stamping, classification, sampling, and the like. It is worth emphasizing that errors at this point might propagate to other components of the measurement architecture, thus probably making results unreliable. As depicted in Fig. 2.1, the four major steps in the measurement process are capturing, processing, record generation, and exporting. Each step can be briefly explained, as follows:

I. *Capturing*: the actual mechanism of acquiring packets from the network interface for further processing. Packets at this level can be trimmed down to only the header or a portion of the actual payload.
II. *Processing*:

II.1. Time stamping: adding time-related information to the packets as they arrive. Time stamps range from nanoseconds to milliseconds, depending on the clock reference source.

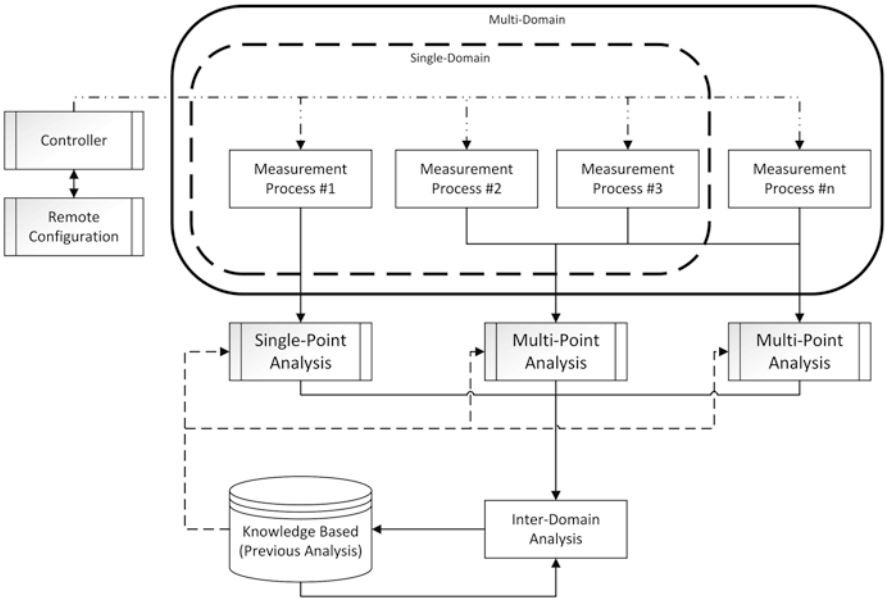

Fig. 2.2 The measurement reference architecture, by Tanja Zseby [5]

II.2. Classification: group packets according to a set of rules
II.3. Selection: yields a subset of the incoming stream of packets, by applying filtering or sampling techniques
II.4. Aggregation: a data summarization technique

III. *Record generation*: standardized way to export information after the aggregation process
IV. *Exporting*: the use of communication protocols to transfer packets or flow records to the collecting process

In this chapter, I dissect the building blocks of the Internet measurement realm.

2.1 Passive vs. Active vs. Hybrid Measurements

First, let's clear up a bit on the fundamental concepts around measurements on the Internet. The distinction between "performance metric" and "measurement methodology" must be clear in every network performance analyst's head. In RFC2330 [6], Paxson et al. stated that "there are several quantities related to the performance and reliability of the Internet that we'd like to know the value of. When such a quantity is carefully specified, we term the quantity a metric." In [7], Morton complements such definition by stating it as a "quantity, produced in an assessment of performance and/or reliability of the network." The definition of performance

metrics specific to IETF-based protocols is given by Clark and Claise [8] as "a quantitative measure of performance, specific to an IETF-specified protocol or specific to an application transported over an IETF-specified protocol." As far as we are concerned to methods of measurements or measurement methodology, one good general definition is given by Morton [7] as "The procedure or set of operations having the object of determining a Measured Value or Measurement Result." I stick to this definition and we will use the term [7] *measurement methods* or *measurement methodologies* interchangeably. You might find the term *measurements* sufficiently general to encompass both performance metrics and method concepts. It has been used by the Internet research community to describe all the work involving performance assessment. However, in a number of cases, it is better to use accurate terms.

When one makes a comparison between active and passive measurement methods, one might have a first impression that there is no middle ground here. To put it simply, if a certain technique injects packets into the network in order to measure or calculate something (i.e., the performance metric of interest), it is definitely an active measurement method one. More formally, as presented in [7], active measurements depend on a dedicated measurement packet stream and observations of the stream. On the other hand, in the case of a technique that only sniffs the network and do not add any extra load on it, it is for sure a passive one. Again, more formally, passive measurements depend solely on observation of one or more existing packet streams. However, there is a middle ground indeed. The concept of hybrid measurement methods has arrived recently. Hybrid methods use a combination of both active and passive methods. They are used to assess either active or passive metrics as well as new metrics that could be within only in the scope of hybrid metrics [7, 9, 10].

It is worth making clear some preconceived notions about the processing power requirements for active, passive, or hybrid measurements. The idea that passive techniques require less processing power in the network nodes or that the active techniques are highly intense on processing usage does not hold. Depending on the measurement scenario, one can have a very light and precise active technique (e.g., think of ping) or a very processing intense passive one (e.g., think of NetFlow running on a 400Gbps backbone core network switch). In fact, one might be aware that there are some risks associated with passive measurements. In [10] the authors pointed out that a passive method might create a load on the device that could potentially change the measurement environment itself.

It has been said many times (no reference though, since it is obvious) that any active technique should not overload the network, let say injecting no more than 5% of the link or path capacity. Of course one should have in mind that the network links are deployed to transport real data traffic, not signaling/control ones. The main problem with this rule of thumb is that on the Internet most of the time it is difficult to infer the network bottleneck unless we are dealing with point-to-point communications in a single link. But, this must not encourage anyone to

develop active techniques and do not care about the actual sending rate of the tool he/she is writing. Let's just be reasonable in that matter. In fact, the observer effect is valid here. Although it is difficult to pinpoint a formal definition, the observer effect in science and engineering means that act of observing (i.e., measuring, in our context) might have an influence on the object being observed/measured. Of course, this might remind the reader of some physics concepts, but they are not related, really. One can learn this lesson from an interesting quote by Fred Mosteller and John Tukey, two famous statisticians: "The only way to find out what will happen when a complex system is disturbed is to disturb the system, not merely to observe it passively."

To give the reader detailed definitions on the measurement methods, we rely on [7, 10], since there have been a number of discussions around it within the IETF's Working Group on IP Performance Metrics (IPPM). In [7], Morton defines active measurement methods as the methods that:

(i) "The packet stream of interest is generated as the basis of measurements."
(ii) "The packets in the stream of interest have fields or field values dedicated to measurements."
(iii) "Source and destination of the measurements points are known a priori."
(iv) "The characteristics of the packet stream is known at least by the source."

A complementary definition is given by Zheng et al. [10] as "The process of measuring performance or reliability parameters by the examination of traffic (IP Packets) injected into the network, expressly for the purpose of measurement by intended Measurement Point(s)." Passive measurement methods are the ones that are:

(i) "Based solely on observations of undisturbed and unmodified packet stream of interest"
(ii) "Dependent on the existence of one or more packet streams to supply the stream of interest"
(iii) "Dependent on the presence of the packet stream of interest at one or more designated observation points"

Now, with all those definitions in hand, we are able to classify and identify performance assessment on the Internet as *active*, *passive*, and *hybrid* and also have a clear view of the metrics and methods for each class. For an in-depth understanding of large-scale performance assessment on the Internet, we refer the reader to a survey paper entitled "A Survey on Internet Performance Measurement Platforms and Related Standardization Efforts" [11] and to the references therein. Figure 2.3 presents an example of passive measurements, whereas Fig. 2.4 represents a typical active measurement scenario.

As the main ideas behind the concepts of passive, active, and hybrid measurements are clear, let's delve into the nuances of collecting real data traffic. The interested reader may want to have a look in [12–14] to understand standardizing efforts for network measurement platforms.

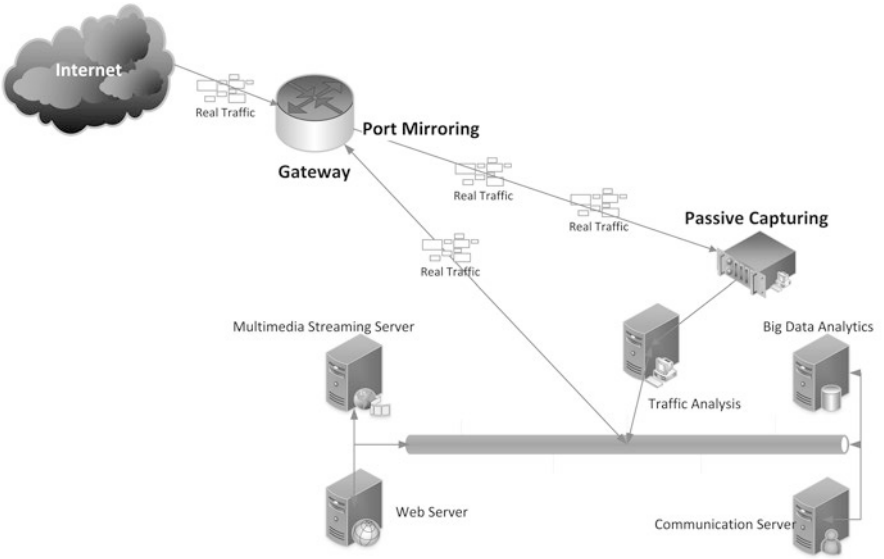

Fig. 2.3 Passive measurement process

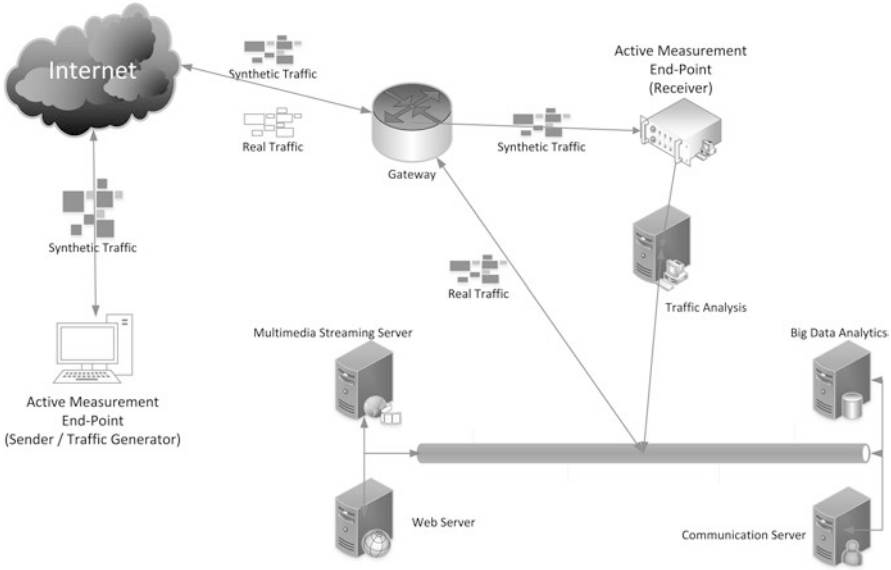

Fig. 2.4 Active measurement process

2.2 Traffic Measurements: Packets, Flow Records, and Aggregated Data

One important decision to be made when preparing a performance assessment plan regards the granularity level of information you will need. In general, the more detailed information on the collected performance metric, the larger its corresponding requirement for data storage. Suppose you need to derive an analytical model from measurements in order to predict the utilization of a certain Gbps WAN link, for the next 6 months. Let's assume that the traffic profile is well behaved (i.e., a stationary ergodic process), in which a simple time series technique (e.g., moving average) would be sufficient to model it. Given the target time scale in hand (i.e., months), it does not make sense to collect traffic at time scales smaller than minutes. It would be a waste of storage resources and it will not add more precision to the model. For example, you decide to collect traffic at the WAN port of the router. You have some options to it, as follows:

(i) Mirroring the traffic at the switch and collecting all the packets coming to/from the router. This first option will give you detailed information, probably at the microsecond level, but the amount of storage needed for a single day will be huge, in the order of 100s of gigabytes or even terabytes (it depends on the packet capture strategy. Capturing only the packet headers is an economical way to do it).
(ii) Activating the collection of flow records (e.g., IPFIX or NetFlow). This option will be more efficient in terms of storage requirements while still giving some room for further analysis, such as the understanding of distribution of flow sizes and duration.
(iii) Activating the MIB/SNMP counts on the interface. The last option will hide most information but will be very efficient, since SNMP clients generally poll the SNMP server every 5 min only. In such a case, one full day of SNMP traffic (e.g., ingress bytes) will yield only 288 samples (12 samples/hour x24 h).

Please notice that granularity not only refers to time scales but also means the level of information that the collected traffic trace carries inherently. It is straightforward to see that full payload packet-level trace would give the network analysts the most accurate information because the trace has virtually every bit of information that the sender-receiver pair exchanged. Please note that in the case of encrypted traffic, the payload itself will not be of any help, unless you either have the encryption key to decrypt its content or can use specific technique [15, 16] to overcome this issue. On the other hand, the more aggregated the collected trace is, the less detailed information is. For example, if you collect MIB counts via SNMP only, there is no way to identify the set of applications that generated that traffic. All important information for traffic classification and identification is lost, such as source and destination ports and IP addresses, protocol numbers, packet header flags, and the like (cf. Fig. 2.5). Last, but not least, an important source of information

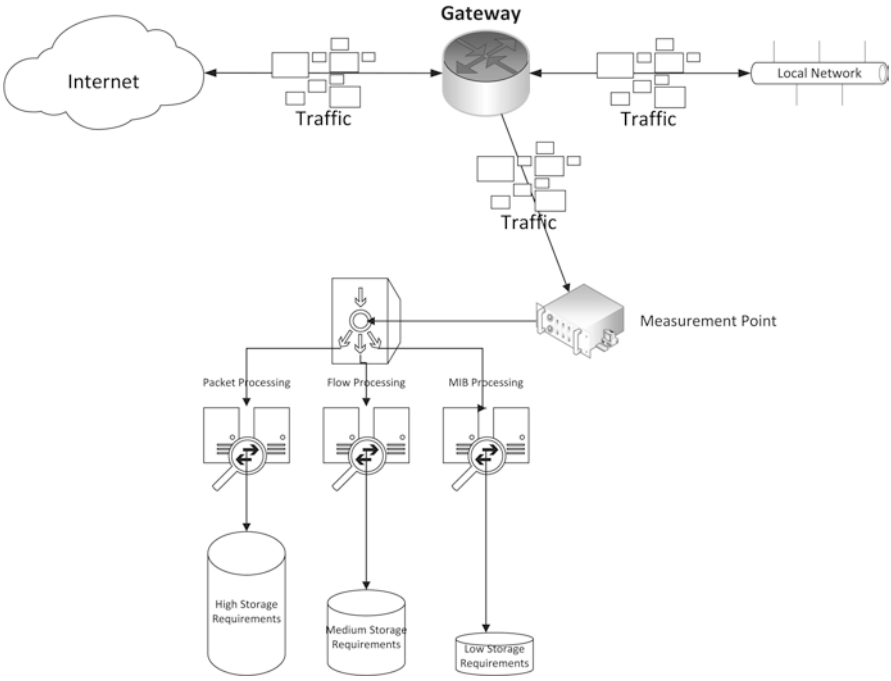

Fig. 2.5 Storage requirements for different granularity levels of the measurement process

is the system's logs. There are a number of ways to activate logs for several services (e.g., HTTP, DHCP, DNS, etc.) in most operating systems.

It is clear that the network analyst must make important decisions when selecting the appropriate level of information she will need. A top-down approach would be better if she knows what will be the purpose of her analysis exactly. We propose a general rule of thumb as follows:

I. Decide on the type of analysis you need.
II. Choose the appropriate time scale and information granularity.
III. Choose the appropriate techniques and tools.
IV. Calculate the amount of storage necessary.
V. If there are not enough resources available, return to Step I or II.

To give an example, let's assume you need to identify what is causing congestion at the company's WAN link at peak hours (e.g., 9 am, noon, and 1 pm–4 pm). You don't know exactly what is the underlying cause, since a number of possibilities are in place, such as applications, services, denial of service (DoS) attacks, router/routing malfunctioning, ISP firewall services, and the like. You should assume that coming to the head of the IT department with a request to upgrade link capacity urgently would not solve the problem. It may alleviate the symptoms temporarily, but it would not solve it definitely. Well, one might argue that this is a never-ending and

recurrent type of problem. Anyway, you have to solve it and stick to the IT budget. You might get some clues from performance logs in the server systems to identify if the source of the problem is either inside or outside the network and, if it is inside, in which subnetwork. From there, you can manually dig into other system's logs (including per-port router and switches utilization) to narrow down your search. It will take time and it might be painful. In this particular case, an in-depth traffic analysis is likely to be necessary (*step I: type of analysis*). In other words, you might need to inspect traffic at packet's payload level. Therefore, one more effective and efficient approach would be starting collecting (if this process is not already present) and analyzing SNMP counts (e.g., in/out traffic via any SNMP tool) in the Internet gateways, core routing/switching devices, major systems that provide services, and the like. Please be aware that dealing with virtual networks environments is somewhat more tricky and might require different strategies. We will delve into monitoring virtual environments in Sect. 2.5. Initially, there is no need to start collecting full packets at every network interface, since this might pose a heavy burden on the storage facilities. Choosing the appropriate time scale and information granularity (*step II*) is of paramount importance at this point, even if you have to revisit it later. It is better to start with analyzing SNMP counts (aggregate level) and then have a look at IPFIX records (flow level) to narrow down to a particular subnetwork, device, host, or service (*step III*). If there is no clear answer to troubleshoot, it is time to design a plan to collect packets (*step IV*). It is hard to identify protocol misbehavior with flow-level information; thus, deploying or activating DPI systems is a way to go. If lack of storage resources is an issue (*step V*), you must have a step back and find suitable techniques to infer application traffic from flow records without resorting on port-based information.

In another example, let's assume you have analyzed through SNMP (e.g., using the RRD tool) [17] the WAN link utilization and noticed that there was a substantial incoming traffic (e.g., 50% of link capacity) from 5 pm to 8 am, when there is no one working at any office during these hours. If you only need to identify the IP address of the receiving nodes, you will just need to activate IPFIX/NetFlow at the gateway router, collect traces for the given time frame, evaluate which source and destination IP addresses were generating the most traffic, and then check your DHCP log (also assuming you have dynamic distribution of IP addresses) to pinpoint the exact end hosts within your network. However, if you need to report a particular usage of an application (e.g., that might be prohibited according to the given corporate policies), then the IPFIX/NetFlow information might not be enough. As we will see in Chapter 4, one way to perform traffic classifications and identification is to rely on port information of the communication sessions available at IPFix/NetFlow records. Since a number of applications (e.g., peer-to-peer applications) may use a variety of source and destination ports, such analysis is not accurate and the analysts must rely on computational intelligence strategies. Karagiannis et al. [18] presented an in-depth traffic analysis showing that P2P applications use strategies to hide themselves into port numbers assigned to well-known applications, in order to evade from firewalls' blocking rules. Therefore, port-based traffic analysis has become highly inaccurate and a wave of computational intelligence

techniques came into play for flow-based traffic classification. We refer the reader to [19, 20] as an excellent overview of the use of machine learning techniques and tools for traffic classification.

The above examples are more common for IT professionals. From the point of view of scientific research, the suggested approach is likely to be the same, although the given researcher might probably have a more clear objective (at least the type of analysis) in mind, as the result of the raised hypotheses and research questions. Let's say that you want to evaluate the effect of a self-similar traffic on the virtual switch buffer size requirements. I am not suggesting that this is a valid research topic. I just want to emphasize that the given keyword self-similar requires you to evaluate traffic at several time scales. If you just collect an aggregate throughput at every second, you will probably not be able to calculate the Hurst parameter [21, 22] with accuracy. Thus your simulation, emulation, or experimental work must deal with the highest precision time-stamping mechanisms as possible, in order to collect packets at nanosecond (or even lower) scale. This type of measurements not only adds huge and strict storage requirements but also push you to buy expensive specialized hardware (e.g., GPS-enabled network interface cards [23] with high-precision time-stamping mechanisms) for the measurements.

2.3 Sampling Techniques for Network Management

This subsection has partially some contents from the research paper [24], which I am one of the coauthors.

Commonly used monitoring tools at flow and packet levels suffer from lack of scalability relative to link capacity. Monitoring high-speed links, in the order of hundreds of GB or TB, yields an immense volume of data for storage and further processing. It is worth emphasizing that as link capacity and the number of flows grows, maintaining individual counters for each individual flow traversing a certain router becomes computationally and/or economically impractical [25]. The technical challenges for deploying accurate network monitoring are exceedingly large in the processing, storage, and transport dimensions, as follows:

I. The capturing processes in routers compete with the routing tasks.
II. Temporarily storing a large number of flow samples requires more memory.
III. Exporting sampled data to network management tools floods networks with control traffic.

Sampling techniques are proven statistical strategies for dealing with high-volume data, in which network traffic and topologies are some clear examples. Statistical sampling is also an important strategy for social network analysis (e.g., Facebook, Twitter, and the like) [26] due to their ever-increasing data generation. One of the main advantages of sampling is the potential for lowering the processing cost, as well as the storage and transport requirements in the given monitoring infrastructure. Therefore, in order to make the analysis of large datasets (or high-throughput

traffic in real time) faster while keeping accuracy in check, one must use only a small, although relevant, subset of such traffic data that holds its inherent properties. Such subset will be then analyzed with guarantees of preservation of the original statistical properties. Please note that sampling is usually an irreversible process, thus causing information losses [25]. Figure 2.6 shows a general scenario for traffic monitoring with flow-based classification and identification.

As far as we are concerned to network traffic measurements, sampling techniques can be applied during the actual measurements (e.g., at packet level), to the traffic trace files (e.g., at either packet or flow level), or even after data summarization (e.g., at flow or higher aggregation level). As an example, Fig. 2.7 depicts the potential benefits of using sampling techniques for the case of flow-level traffic

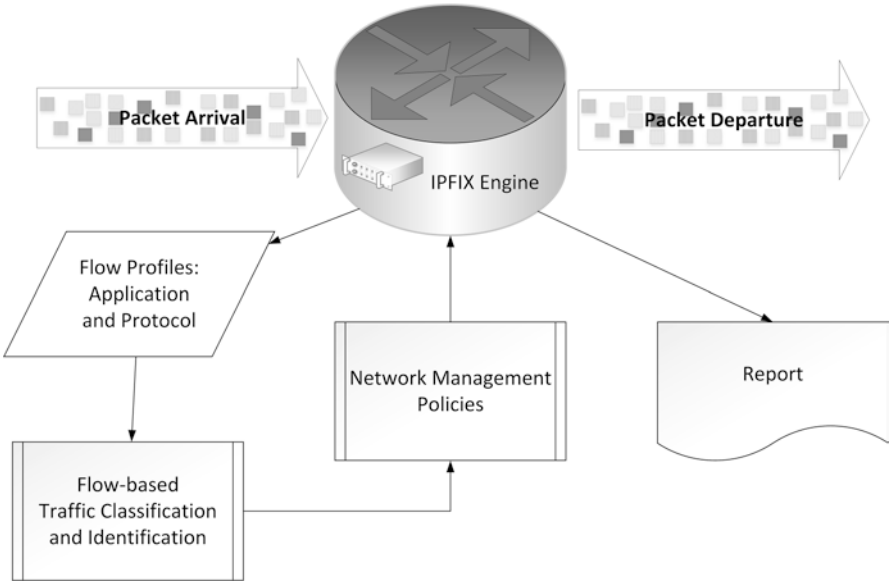

Fig. 2.6 Flow-based traffic classification and identification

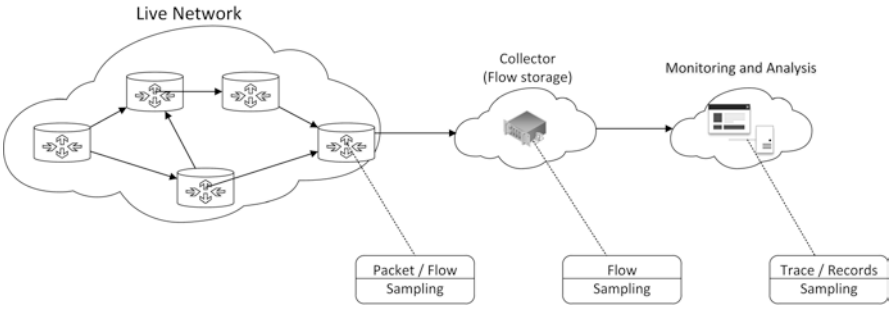

Fig. 2.7 Sampling network traffic

data. It shows where sampling can be deployed as a flow-level filtering capable of selecting a representative set of flows only. The sampled set will give the analyst similar accurate view of the traffic mix as in the original trace. Consider that a scenario where a network management software (e.g., Cisco's NetFlow) within a router builds flow records from the passing packets and an external collector retrieves and stores such flow records. In the figure, I highlight the possible vantage points – namely, packet/flow sampling, flow sampling, and trace/record sampling – for deploying such techniques at different components of the network management scenario. Real-time sampling may be performed directly by routers when they are capturing packets to create flow records. Post-processing (offline) sampling can only be performed after the flow records were created and dumped into the router's memory and transmitted to the flow collector. I also consider that if offline sampling is being used, data reduction techniques may be applied to reduce storage requirements at the collector, to reduce communication overhead from the collector to the network management application, and to reduce further memory and processing requirements. In the case of sampling in real time, it will certainly reduce the processing and memory requirements for the routing equipment and the communication overhead among the network management software components. We also argued that the processing and memory requirements will certainly be reduced at the router since they usually preserve all flows in memory until they are flushed to the collector and the memory is cleared. Our main argument is that if an effective sampling technique is deployed, the router may avoid storing all flows. The implications are that the allocated memory for some flows can be released, thus the processing work for all the remaining flows will be performed faster.

Although the advantages of applying sampling techniques for network monitoring are clear and stimulating [26], it is worth emphasizing that they pose additional challenges for network security analysis. Detection of network abnormalities, attacks of several types, failures, and the like, will become harder and more complex to perform since the sampling process may remove relevant information from the traffic. As network security goes way beyond traditional usage-based accounting and billing, traffic profiling, SLA validation, etc., the accuracy of security analysis techniques will likely suffer from less information available [27]. In [28] the authors show convincing arguments that network behavior analysis or intrusion detection techniques suffer from less accuracy when dealing with sampled data. They argued that sampled data impacts the effectiveness of the algorithms for security analysis due to the inherent distortion of traffic features they cause.

In the scientific literature, a number of sampling techniques have been proposed in several different knowledge areas. As far as we are concerned to computer networks [24], some of those techniques have been applied to the packet capture process, whereas others to flow records selection. The most common approaches for sampling for network monitoring are the systematic, random, adaptive, and stratified ones. We give you an overview of some of the sampling strategies common in the network monitoring realm.

It is intuitive to see that the simplest sampling technique is the systematic sampling, where one sample is selected out of N packets or flow records. A general

deterministic sampling function can be used in the systematic approach, for instance, a time-based periodic selection, where samples are collected at a constant time interval. A small variation is the random sampling where the samples are selected according to a probability distribution function. For the uniform case (also known as simple random sampling), n samples are selected out of N packets or flow records. The main drawback of these techniques is that it might lead to low accuracy when dealing with traffic that exhibits heavy-tailed probability distribution properties, which are very common on the Internet [29]. Dealing with mice and elephant flows as well as traffic seasonality (i.e., temporal cycles) poses additional challenges [24].

Adaptive sampling is a more elaborate technique. In most cases, it adapts the sampling rate depending on the amount of incoming data. In the case of computer networks, it makes sense to lower the sampling rate when traffic volume is high (and vice versa), so you can control the sampled traffic volume [30]. A particularly interesting approach is the use of size-dependent sampling, or Smart sampling, or Threshold sampling. This is the case when an object of size x is selected according to a sampling probability function [33]. Given a specific threshold z, a flow record or a packet size with x bytes has a probability $p_z(x)$ of being sampled, where

$$p_z(x) = \min\left\{1, \frac{x}{z}\right\}$$

It is straightforward to see that flows carrying bytes greater than z are always selected. Samples of size less than a threshold z are selected with probability x/z. With the application in network traffic, Duffield et al. [31, 32] showed that they could control the resulting sample set size. Furthermore, they presented an approach capable of inferring the probability distribution for the number and length of flows in the original Internet traffic [34]. In [24], Fernandes et al. proposed a stratified sampling and flow classification methodology that achieved higher reduction levels (on the order of 0.1%) with excellent accuracy for the estimates of the sum, mean, and variance for flow duration and sizes.

2.4 Internet Topology: Measurements, Modeling, and Analysis

An in-depth understanding of the Internet topology is of paramount importance for a number of performance evaluation studies. For example, when dealing with large-scale (e.g., worldwide) analysis of peer-to-peer applications, it is important that the underlying infrastructure reflects the actual network connectivity of routers and links on the Internet. Otherwise, performance assessment might be biased or inaccurate due to poor representation of how real networks are connected at different levels (e.g., at the link, network, PoP, or AS layers). Figure 2.8 shows an Internet topology where one can observe different connectivity levels, namely, at router, point of presence (PoP), and AS levels. Evaluation of network protocols, as well as

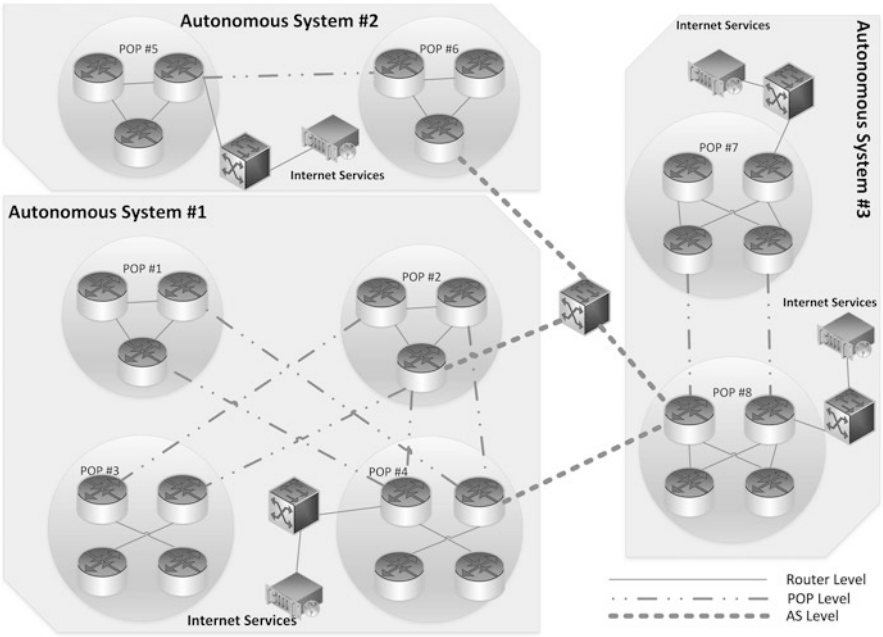

Fig. 2.8 Internet topology resolution: router, PoP, and AS levels

the resilience of the underlying infrastructure, has been a recurrent research topic where an accurate representation and understanding of the multiscale structure of the Internet play an important role. It is clear that realistic models are necessary although an adequate model will always depend on the research objective. In other words, some models have detailed information to form the topology, such as link capacities, buffer sizes, etc., whereas high-abstract models might have only connectivity information at AS levels. High abstraction levels might not be sufficient for performance analysis of inter-domain routing protocols [35] but can be enough for economical analysis.

As pointed out by Roughan and Willinger [36], there are some real motivations for discovering and modeling Internet topologies, from engineering reasons to network management improvements to the deep scientific understanding of a real large-scale complex network.

You might be wondering why is it so difficult to have the big picture as well as a detailed view of the Internet topology. To begin with, you will find no consensus in the literature on the definition of Internet topology. You have to recall that the Internet is composed of networks of different sizes. Each network might have his own administrative domain, and as the commercial and business competition grew since the early 1990s, there are no clear incentives for the ISPs to disclose their topologies. In fact, business competition encourages hiding the internal network structure to avoid performance-related comparisons or exposing security vulnerabilities. For decades, researchers have been trying to understand if the Internet could

be modeled by a single theoretical model that would reveal some invariant network properties. In the past, some claimed to have discovered it [37, 38], but researchers have been challenging them with irrefutable evidence [35, 36, 39–41].

Theoretical characterization of the Internet topology claimed to be validated by experimental work has been the core of some available topology generator tools. In addition, measurements and discovery techniques for Internet topology have been an active research topic for over two decades [35, 42–47]. The challenges in this topic, from the research and engineering viewpoint, are due to the fact that the Internet connectivity is intrinsically dynamic at several levels. Changes in the access and core network occur frequently since ISPs make modifications to their networks by adding, removing, or reallocating routers to provide better services to their users. Also, due to economical reasons, ISPs might split or merge, thus forming new networks at autonomous system (AS) levels [48]. One of the most important questions for your performance evaluation is: At what Internet topology resolution you will need for my performance analysis? Different motivations will require different views on the connection of the Internet structural components. Analyses range from economical to human behavior to protocols and everything in between.

2.4.1 Internet Topology Resolution

The classic definition of an autonomous system (AS) is given by RFC 1771 [49], updated by RFC 4271 [50], which states:

> The classic definition of an Autonomous System is a set of routers under a single technical administration, using an interior gateway protocol and common metrics to route packets within the AS, and using an exterior gateway protocol to route packets to other ASs. (RFC 4271, title)

RFC 1930 shorts this definition by stating:

> An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy. (RFC 1930, title)

Recent numbers show that the Internet is composed of approximately 50 K ASes [47] with 500 K routers. They are managed by different companies in a number of geographic areas. The mapping between ASes to organizations has been a recent topic of research [51], and CAIDA has recently made available some datasets [52]. Such mapping can be useful for an in-depth understanding of the topological, administrative, and economic relationships on the Internet. In fact, as the ASes have their own network infrastructures and the organizations have their particular business goals, ASes can have a variety of purposes. For example, ASes can be a commercial Internet Service Provider (ISP), an education network (e.g., Brazilian National Research and Educational Network – RNP), a Content Distribution Network (e.g., Akamai), and the like.

In a broad categorization, Internet topology can be modeled at two levels, namely, autonomous system (AS) and network (router) levels. However, there are good efforts

to characterize the Internet topology at other levels, such as at physical links, router interfaces, subnetworks, and Point of Presence (PoP). Therefore, as far as we are concerned to representing Internet topologies as network graphs, the vertices (or nodes) and edges can take different roles. For modeling Internet topology at AS level, vertices are the ASes (identified by the 32-bit AS number), whereas edges are the high-level interconnections (i.e., it can be constituted of multiple physical links) between them, which can be somehow measured by passive or active techniques [53]. For network level, the vertices are routers and edges are the network layer connectivity with abstraction to the underlying link layers or logically aggregated tunnels (e.g., MPLS).

It is worth emphasizing that – surprisingly to some – neither there is a current ground truth for the whole Internet topology, nor there are tools that can discover the Internet's underlying topology with 100% accuracy [47] nor analytical modeling that can precisely represent the structural components of the Internet. Sometimes the problem relies on the measurement approach (e.g., traceroute-like tools), on the datasets coming from the data plane or control plane (e.g., with missing or mapping issues between AS and the address space), and even on the abstraction itself (e.g., an AS as a node is clearly an oversimplification). The mismatch between the requirements/motivations, what you need to measure, and what information is available to extract from data generally leads to erroneous conclusions. For example, using BGP routing tables to infer AS-level or router-level Internet topology is not precise since some interconnections might not be used by a certain AS for a number of reasons.

From the point of view of research, the most important aspect might not be inferring 100% of the Internet topology, neither understanding its inherent structural characteristic, but its engineering design rationale. The main argument here is that fitting a model to data is not the best approach when the measurement process or even the data itself is error-prone, incomplete, or anecdotal. We refer the reader to [35–37] for an in-depth view on the challenges for revealing the Internet topology precisely. When comparing popular Internet topology models with actual network topologies, one will find a number of discrepancies, such as the nonexistence of highly connected routers. In fact, prominent researchers (e.g., Willinger and Roughan [36]) have clear arguments that the widespread concept of a Power-Law distributions for router connectivity on the Internet is not supported by any empirical and statistical evidence as inference results from a number of measurements techniques cannot be trusted.

Researchers have been developing a number of tools and techniques to ease the topology discovery process. Traceroute was the initial tool of choice, but due to its inherent limitations, such as interface disambiguation, IP-alias resolution, and the like, other tools have arisen, such as mrinfo [53], MERLIN [55], Scamper [54], Rocket Fuel [78], etc. And of course, it should be clear that traceroute's main objective is diagnostic, not topology discovery. One of the most difficult aspects of mapping the Internet topology is deploying and managing a large-scale measurement infrastructure. For instance, in order to deploy active measurement probes, one must carefully select their vantage points. In addition, a number of publicly available datasets have encouraged researchers to develop analytical network models to represent its inherent properties. However, there are too many problems with topology measurements and their derived models (and concluding remarks from them) that it is safe to state "We

did your best, but our best was not good enough" or "Nice try." So, no luck so far. Just be careful when using datasets, measurement tools, and "validated" models. Most active or passive tools fail to identify routers and links in the deep Internet core, which is a loose definition, but we can define it as routers and links that are n routers away from the edge, in the m_{th} AS networks, where n, m is yet to be determined.

2.4.2 Internet Topology Discovery: Tools, Techniques, and Datasets

I hope at this point you are not expecting good news in this matter. I mean I hope you are not expecting that this section will give you plenty of options to find good tools for synthetic generation of topologies to support your performance evaluation work. Just recall that both data plane and control plane measurement tools have several inefficiencies at several topology resolution levels. And those measurement results formed the basis of most topology generator tools' core engine. However, as long as you are not looking for a graph-based view of the Internet topology, there is light at the end of the tunnel. The follow subsections give a glimpse of which tools, techniques, and datasets are likely to be useful to use in a good performance evaluation plan. Particularly, Scamper [54], MERLIN [55], Combined Optimization and Layered Design (COLD) [57], IGen [56], Topology Zoo [79], Internet Atlas [80], and Macroscopic Internet Topology Data Kit (ITDK) [81] must be your safe bet. Please note that this is not an exhaustive list of options, but a more recent view on the good representative ones.

2.4.2.1 Topology Discovery Tools

Scamper[1] is an open-source IPv4/IPv6 active measurement tool from CAIDA. By probing router interfaces, Scamper's main goals are network performance analyses and topology discovery. One of the motivations behind the development of Scamper was to build a prober for large-scale measurements and analyses with support to many different techniques. Scamper architectural components include several features, such as optimized and multipath traceroute for topology discovery, ping, precise alias resolution, one-way loss inference, and transport protocol behavior analyses. This set of features makes Scamper one of the most scalable and precise probe tools currently available.

It supports a wide range of techniques to conduct elaborated measurements. As described in the paper [54], Scamper has been used in a variety of contexts and scenarios, such as to router-level connectivity discover between a source and destination (it is the tool used for the CAIDA's macroscopic Internet topology discovery project) and to the evaluation of the impact of IPv6 tunnels in IPv4/IPv6 paths.

[1] http://www.caida.org/tools/measurement/scamper/.

MERLIN [55] is a router-level topology discovery tool derived from mrinfo[2], designed for large-scale measurements. MERLIN's architecture has two processes responsible for sending probes and processing the respective response from routers. Performance analysis presented in [55] shows that MERLIN overcomes the limitations of mrinfo and it has good accuracy to conduct topology discovery measurements.

2.4.2.2 Topology Generator Tools

IGen[3] and COLD[4] fill an important gap left by the existing synthetic router- and Pop-level topology generator tools. They both yield more realistic synthetic topologies that can be used in network performance evaluation experiments. In [56] the authors describe the network design heuristics used to build IGen and show that IGen addressed the most common issues in popular synthetic topology generator tools, such as BRITE and GT-ITM. In essence, IGen's methodology (cf. Figure 2.9, IGen's Methodology) is comprised of six main steps, namely:

(i) Router placement
(ii) PoP identification
(iii) Topology building from PoPs
(iv) Building backbones
(v) Selection of link capacities and network paths
(vi) Internal BGP overlay

Details can be found in the original paper [56].

Combined Optimization and Layered Design (COLD)[5] [57] deals with PoP-level synthetic network generation. COLD takes into account concrete network parameters instead of abstract ones from graph theory. As far as we are concerned to the generation of different (statistically speaking) topologies while keeping certain invariant properties, COLD has the following approach. Given the input parameters, it actually generates a deterministic topology. Stochastic variability comes from the randomization of contexts, such as PoP location and traffic matrix. In addition, COLD authors argue that it is possible to generate router- and AS-level network from the available code.

2.4.2.3 Topology Datasets

It seems that PoP-level characterization is a good way to develop further either router-level or AS-level topologies. The research community still has a long journey down that road. If synthetic topology generators are not enough and you want to test

[2] http://svnet.u-strasbg.fr/mrinfo/mrinfo.man.html.

[3] http://inl.info.ucl.ac.be/softwares/igen.

[4] https://github.com/rhysbowden/COLD.

[5] https://github.com/rhysbowden/COLD.

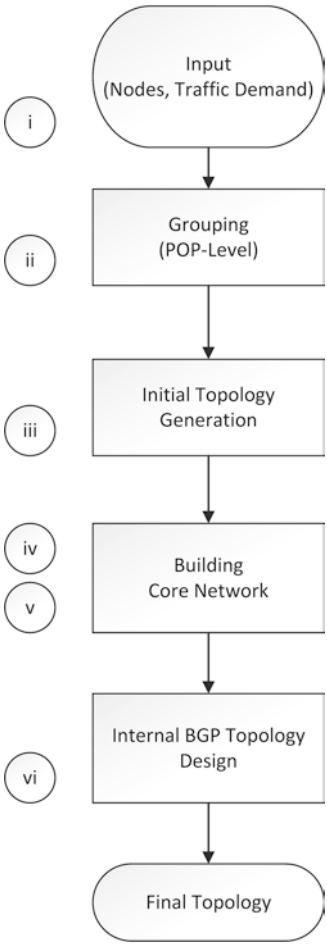

Fig. 2.9 IGen's methodology

your ideas on real network topologies, there are some good datasets out there. Just be aware that the statistical analysis might be done with other methods than varying the underlying network structure.

The Macroscopic Internet Topology Data Kit (ITDK)[6] is a major initiative from CAIDA. It is essentially a repository of active measurements from its Ark infrastructure. ITDK provides router-level topologies as results of two different strategies for alias resolution. Topologies are available in the form of files, namely, nodes, link, node-to-AS, hostnames, and node-geolocation files. Future extensions of ITDK plan to include AS-level views of the Internet.

As an alternative, the Topology Zoo project[7], developed by the University of Adelaide, Australia, provides a variety of real-world network topologies. As of early

[6] http://www.caida.org/data/internet-topology-data-kit/index.xml.

[7] http://topology-zoo.org/index.html.

2016, it has over 250 networks in the dataset in from commercial and educational networks. It also comes with several levels of details, such as PoP and physical levels. They collected data directly from network providers and after some processing (for more details we refer the reader to their Web site), they make available the transcribed network in a text-based Graph Markup Language format.

2.5 Challenges for Traffic Measurements and Analyses in Virtual Environments

Networking environments that strongly rely on virtualization technologies, such as cloud computing and network virtualization, is likely to bring a whole new set of measurement methods and metrics. This is on top of the existing ones so far discussed in this chapter. A number of features that are only within the scope of such environments must be added to the performance evaluation toolbox, for example, elasticity or on-demand scalability, which means the ability of the resource provider to add or remove resources to meet a certain service requirement, which might be an integral part of SLA agreements, and therefore must be measured and reported precisely. There is a clear need for a general framework for performance evaluation in virtualized environments and the first steps in this direction have already been made [58]. With the view of computing resources as services, performance evaluation cannot be now restricted to infrastructure monitoring only. Application performance has become an important aspect of the performance evaluation in such new environments. The main argument here is that monitoring virtual environment cannot be seen as a simple case of enabling promiscuous mode only in the virtual machines distributed in the virtualized environment. There are several accuracy issues when performing measurements in virtual environments and the research community has begun to pay attention to this problem [59]. In fact, network monitoring and troubleshooting within environments with several virtualization layers has drawn the attention from both industry and academic communities [60–64].

In this section, we give an overview of the advances, challenges, and practical guidance when dealing with measurements in virtualized environments. We focus on three main scenarios, namely, cloud computing, network virtualization, and SDN.

2.5.1 Cloud Computing Environments

A number of factors contribute to cloud computing performance variation and degradation, such as inter- and intra-VM communications, CPU's load and scheduling policies, type of hypervisor, and the like. There is strong evidence that performance instability in cloud computing environments is mainly due to preset hypervisor's and VM's configurations. Surprisingly, in most cases, cloud provider's SLA agreements do not mention any terms on throughput or latency metrics. In [61], Xu et al.

highlight that the performance of VMs in an IaaS cloud is more degraded and variable as compared to similar VM running on the physical server in a local site. They point out that different cloud environments have their own causes of performance issues. For instance, in the case of single-server virtualization, possible causes of performance degradation are related to CPU, cache hierarchy, shared memory, and the like. In the case of single large datacenter, shared network and storage resources and local VM migration and deployments might impose performance overheads. Finally, they highlight that in the case of distributed data centers, WAN VM or storage migration plays an important role on performance variability.

Recently, the work of Shea et al. [60] investigated how the virtualization layer affects network performance in cloud computing environments, focusing on performance degradation and variation of throughput (sending and receiving performance), latency, and losses. They revealed several important causes of such degradation that any performance analyst or network engineer must be aware of or take into account when dealing with performance issues. For example, they showed that it is very common that TCP traffic suffers from throughput degradation even when the host VM has a very low CPU load on it. The situation gets worse when the VM utilization increases to the point that TCP throughput drops as much as 87%. And it seems that for UDP traffic also suffers from performance degradation, but to a smaller degree. Other performance metrics, such as RTT, can have a fivefold increase when a certain VM is at high utilization load. The main problem here seems to be how the *netback* and *netfront* processes [82] deal with the incoming packets in the given network interface. An interesting observation is that even if the VM instances and all configuration parameters are kept the same, there will still be huge variations on performance. In a nutshell, the main causes for most instabilities in network performance lie on the hypervisor's CPU scheduler and network architecture. Their analysis was based on Xen, but there are some evidence that similar behavior happens in other hypervisors [62, 63]. We refer the reader to Xen's Network Throughput and Performance Guide[8] for some tips on how to properly configure the Xen hypervisor to avoid network performance bottlenecks (cf. Fig. 2.10). The new elements in the virtualized environment, such as the netback and netfront drivers, the virtual interface (VIF), and the shared pages, bring several performance bottlenecks into the system.

In a similar analysis as Shea, although a bit more focused on mobile cloud computing, Maneesh Chauhan [63] investigated how CPU and network load, along with other VM configurations and factors, affect the performance of KVM-QEMU-based cloudlets. The concluding remarks of his work are well aligned with a number of other findings in the cloud performance research community. For example, common network performance metrics (e.g., throughput and RTT) are highly affected by the amount of processing resources allocated to a certain VM. In other words, the fewer CPU resources are allocated to the VM, the lower the performance metrics values. An interesting finding is related to the provisioning of asymmetric VMs on the host machine. For instance, an almost idle VM may have an increase in delays due to

[8] https://wiki.xen.org/wiki/Network_Throughput_and_Performance_Guide.

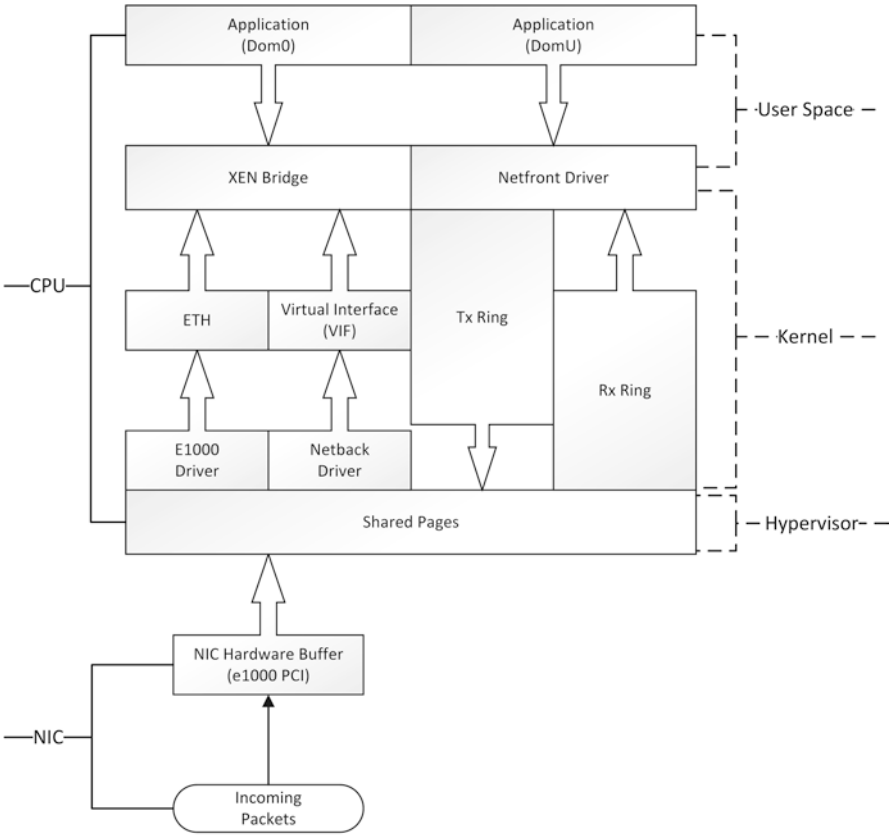

Fig. 2.10 Packet handling in a virtual network system

resource sharing with high workload VM. In fact, it is becoming clear that VM performance varies across different IaaS cloud providers, even if one set up the same VM configuration, as Jayasinghe et al. show [64]. Also, the underlying network infrastructure may cause performance overheads for applications running in cloud environments. In general, overlay networks introduce layers of protocols that have severe effects on upper layers (e.g., transport and application layers). Crisan et al. [65] have recently investigated the influence of overlay network configuration on performance metrics of partition/aggregate and three-tier workload application services. They conclude that, although it is manageable, an overlay network does have an impact on the performance of higher layer applications. Last, but not the least, Persico et al. [66] show how virtualization techniques affect measurements in cloud environments. They show that virtualization and dynamic allocation strategies in cloud infrastructures directly influence performance metrics, such as throughput. For instance, they show that maximum network throughput for a certain VM is a function of its size, geographical location, and other parameters.

2.5.2 *Virtualization at Network Level*

Now that we are aware of how virtualization technologies have an impact on several performance metrics of the protocol stack, it is important to understand how to conduct sound benchmarking tests. It should be clear now that a number of factors might impact the accuracy of the results from any network performance testing when virtual elements are involved.

The first step on discussing methodologies for network performance evaluation in virtual environments has been recently taken by the IETF's Benchmarking Methodology Working Group (BMWG) and IP Performance Metrics (IPPM) Working Groups. Two *informational drafts* have been proposed and they are currently under discussion within the BMWG community [67] as well as an update for the IPPM's Framework (RFC 2330) [6] regarding timers and time stamps definitions and considerations in virtualized environments. This is a new avenue for both BMWG and IPPM communities since BMWG "has traditionally conducted laboratory characterization of dedicated physical implementations of internetworking functions" [68], whereas the Framework for IP Performance Metrics (RFC 2330) did not tackle virtualization technologies originally. Morton [68] highlights that different hardware and software vendors that support virtualized network functions (VNF) provide configuration flexibility, but pose critical challenges to overcome (regarding testing device under test (DUT) to obtain meaningful results). It is clear that the packet journey through the additional layers added by virtualization might result in large variance delays or even imprecise measurements [69].

In [70], Huang et al. discuss important methodological aspects for benchmarking performance of virtual networks based on virtual switches. In this case, the given virtual switch is the device under test (DUT). They address a number of issues such as some initial test considerations, key performance indicators (KPI), test setup, and several benchmarking tests. The authors start discussing that tester's calibration must be explicitly detailed. This is because any tester running in VMs faces performance issues as the DUT VM. They suggest the use of the same type of hypervisor as the DUT. Other particular performance factors for testers include allocated CPU and memory resources to the VM, frame sizes, and the like. Suggested KPIs are throughput, CPU consumption, memory consumption, frame loss rate, and latency. Examples of performance factors that affect the KPIs are frame sizes, the number of VMs allocated in the DUT, and hypervisor type.

Specifically addressing VNF, Morton presents the methodological aspects for benchmarking VNF instances in commodity platforms [70]. He delves into essential considerations for platforms and testing in the VNF scope, such as new hardware components and clear documentation of configuration parameters and settings. For configuration parameters and settings, one must include the number of server blades (if any), CPUs, caches, hypervisor, VM, as well as VNF-related information such as the actual functions implemented, reserved resources, the number of VNFs in the VM host, number of network interfaces, and the like. As far as we are concerned to testing, Morton discusses that new classes of benchmarks and metrics are needed, such as:

(i) Time to deploy VNFs
(ii) Time to migrate VNFs
(iii) Time to create a VN in the underlying infrastructure

One tricky part of developing performance tests in virtual environments is related to clock accuracy and synchronization. This is an important issue when dealing with latency measurements (i.e., RTT or one way). Of course, increased delays (either actual or perceived) affect a number of applications running in virtual environments and must be measured accordingly and precisely. Figure 2.11 shows several measurement points where timing and synchronization should be carefully observed. Understanding one-way delays is sometimes more important than RTT measurements. This is the case when latency asymmetry is in place, which is very common on the Internet. The One-Way Active Measurement Protocol (OWAMP) is the standard protocol for one-way delay (OWD) measurements. The protocol explicitly

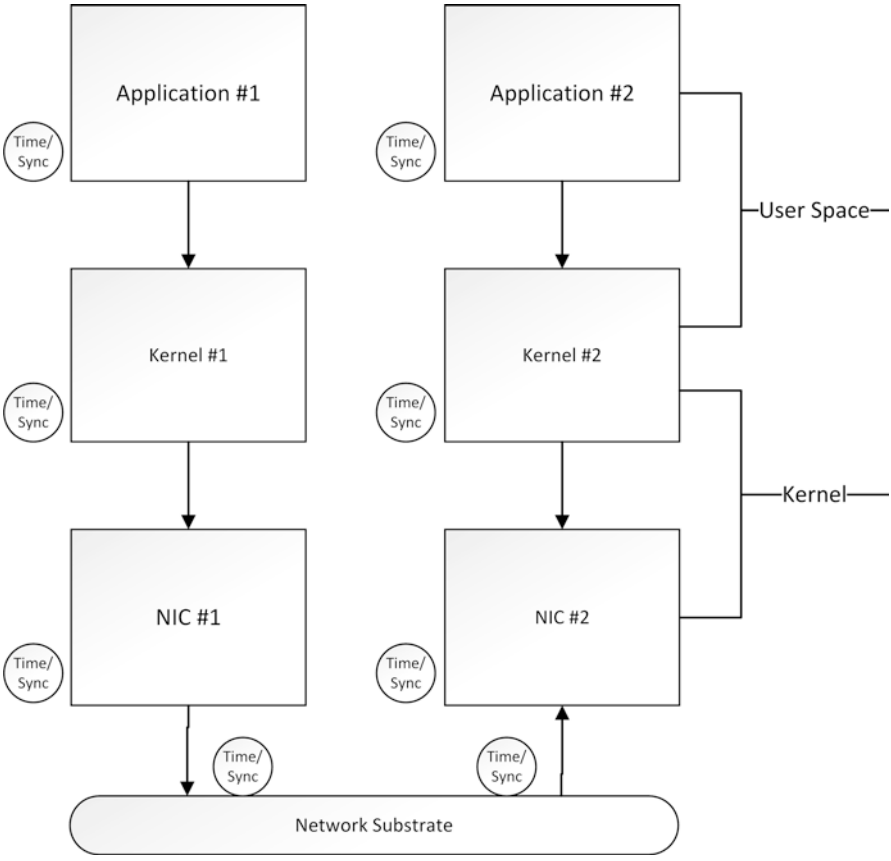

Fig. 2.11 Measurement points where timing and synchronization should be carefully observed

relies on sending and receiving time stamps for every measurement packet during a test session. The owampd tool[9] implements the OWAMP protocol in a flexible and efficient way. One of the requirements of OWAMP is the need of synchronized clocks so that the results can be considered reliable. OWAMP developers highlight two important aspects of a test setup when using owampd, namely, i) the use of at least four Network Time Protocol (NTP) clocks and ii) to not use virtualization packages. The developers warn that virtualization packages "will exhibit clock instability that will make most OWAMP measurements chaotic." In [71] the authors investigate how delay components in virtual environments (e.g., processing and queuing delays) affect the general end-to-end RTT measurements. They provided a methodology to distinguish between sending and receiving delays in RTT measurements. The bottom line is that old concepts, methods, and techniques for measuring clock-related KPIs must be reevaluated and well-understood.

2.6 Bandwidth Estimation Methods

As virtualization technologies pose many challenges to obtaining precise measurements from a number of layers, one might expect that active measurements techniques and tools would suffer from similar accuracy issues. This is the case of both network capacity and available bandwidth estimation techniques and tools. Examples of capacity estimation tools are pathrate [72] and CapProbe [73], whereas examples of available bandwidth (ABW) estimation tools are pathChirp [74, 75] minProbe. The underlying problem here lies in the time measurement accuracy, which might not have a serious impact on low-speed bandwidth, but it does have on high-speed connections [76]. Time measurement accuracy is affected by several factors, such as VM scheduling, interrupt moderation in NICs, OS's rate limiters (e.g., token bucket shapers), and the like. Recall that most estimation tools send a series of time-stamped probe packets to a certain network path and calculate the appropriate metric by performing some analyses over the time stamps of the received probed packets. Wang et al. [75] also point out that bursty nature of the Internet traffic makes the current approaches to performing not so well under such conditions.

Apparently, *minProbe* tackles well the issues of accuracy in high-speed network paths. However, as it is a target to high-speed networks only, so far there are no testing results on its performance in virtualized environments. PathComp [77] deals with some virtualization issues for capacity estimation, whereas PacketTick [76] is able to overcome the limitations imposed by virtualization technologies on ABW estimation. PacketTick is particularly good in providing resilience to cross traffic and it performs well in estimating the ABW in public clouds.

[9] http://software.internet2.edu/owamp/.

References

1. Richter, Philipp, et al. 2015. Distilling the Internet's Application Mix from Packet-Sampled Traffic. *International Conference on Passive and Active Network Measurement*. Springer International Publishing.
2. Brownlee, N., C. Mills, and G. Ruth. 1999. RFC 2722-Traffic Flow Measurement. Architecture 10.
3. Mills, C., D. Hirsh, and G. R. Ruth. 1991. *RFC 1272: Internet Accounting: Background* . RFC, Network Working Group.
4. Claffy, K.C. 2000. Measuring the internet. *IEEE Internet Computing* 4 (1): 73–75.
5. Zseby, Tanja. 2005. *Statistical sampling for non intrusive measurements in IP networks*. PhD diss., Berlin Institute of Technology.
6. Paxson, Vern, et al. 1998. *RFC 2330: Framework for IP Performance Metrics*.
7. Morton, A. 2016. Active and Passive Metrics and Methods (with Hybrid Types In-Between). No. RFC 7799.
8. Clark, A., and B. Claise. 2011. Framework for Performance Metric Development. RFC 6390, October.
9. Recommendation, I. T. U. T. 2016. Internet protocol data communication service-IP packet transfer and availability performance parameters.
10. Zheng, L., Elkins, N., Lingli, D., Ackermann, M., and G. Mirsky. 2015. Framework for IP Passive Performance Measurements. Work in Progress, draft-zheng-ippm-framework-passive-03, June.
11. Bajpai, Vaibhav, and Jürgen Schönwälder. 2015. A survey on internet performance measurement platforms and related standardization efforts. *IEEE Communications Surveys & Tutorials* 17 (3): 1313–1341.
12. Bagnulo, Marcelo, et al. 2014. Building a standard measurement platform. *IEEE Communications Magazine* 52 (5): 165–173.
13. ———. 2013. Standardizing large-scale measurement platforms. *ACM SIGCOMM Computer Communication Review* 43 (2): 58–63.
14. Biersack, Ernst, Christian Callegari, and Maja Matijasevic. 2013. *Data Traffic Monitoring and Analysis*. Berlin/Heidelberg: Springer.
15. Alshammari, Riyad, and A. Nur Zincir-Heywood. 2015. How Robust Can a Machine Learning Approach Be for Classifying Encrypted VoIP? *Journal of Network and Systems Management* 23 (4): 830–869.
16. Shahbar, Khalid, and A. Nur Zincir-Heywood. 2014. Benchmarking two techniques for Tor classification: Flow level and circuit level classification. *Computational Intelligence in Cyber Security (CICS), 2014 IEEE Symposium on. IEEE*.
17. RRDtool. Accessed Sept 2016. http://oss.oetiker.ch/rrdtool/.
18. Karagiannis, Thomas, et al. 2004. Is p2p dying or just hiding?[p2p traffic measurement]. *Global Telecommunications Conference*, 2004. GLOBECOM'04. IEEE. Vol. 3. IEEE.
19. Nguyen, T.T. Thuy, and Grenville Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials* 10 (4): 56–76.
20. Sommer, Robin, and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. *2010 IEEE Symposium on Security and Privacy. IEEE*.
21. Park, Kihong, and Walter Willinger, eds. 2000. *Self-similar network traffic and performance evaluation*. New York: Wiley.
22. Leland, Will E., et al. 1994. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on networking* 2 (1): 1–15.
23. Endace. The Genius of DAG. http://www.endace.com/endace-dag-high-speed-packet-capture-cards.html. Accessed Sept 2016.
24. Fernandes, Stenio, et al. 2008. A stratified traffic sampling methodology for seeing the big picture. *Computer Networks* 52 (14): 2677–2689.

25. Zseby, Tanja, Thomas Hirsch, and Benoit Claise. 2008. Packet sampling for flow accounting: Challenges and limitations. In *International Conference on Passive and Active Network Measurement*. Berlin/Heidelberg: Springer.
26. Das, Gautam, et al. 2008. Efficient sampling of information in social networks. *Proceedings of the 2008 ACM Workshop on Search in Social Media*. ACM.
27. Bartos, Karel, and Martin Rehak. 2012. Towards efficient flow sampling technique for anomaly detection. *International Workshop on Traffic Monitoring and Analysis*. Berlin/Heidelberg: Springer.
28. ———. 2015. IFS: Intelligent flow sampling for network security–an adaptive approach. *International Journal of Network Management* 25 (5): 263–282.
29. Willinger, Walter, Vern Paxson, and Murad S. Taqqu. 1998. Self-similarity and heavy tails: Structural modeling of network traffic. *A Practical Guide to Heavy Tails: Statistical Techniques and Applications* 23: 27–53.
30. Hernandez, Edwin A., Matthew C. Chidester, and Alan D. George. 2001. Adaptive sampling for network management. *Journal of Network and Systems Management* 9 (4): 409–434.
31. Duffield, Nick, Carsten Lund, and Mikkel Thorup. 2005. Learn more, sample less: Control of volume and variance in network measurement. *IEEE Transactions on Information Theory* 51 (5): 1756–1775.
32. ———. 2002. Properties and prediction of flow statistics from sampled packet streams. *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment.* ACM.
33. Duffield, Nick, and Carsten Lund. 2003. Predicting resource usage and estimation accuracy in an IP flow measurement collection infrastructure. *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*. ACM.
34. Kompella, Ramana Rao, and Cristian Estan. 2005. The power of slicing in internet flow measurement. *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*. USENIX Association.
35. Alderson, David, et al. 2005. Understanding internet topology: Principles, models, and validation. *IEEE/ACM Transactions on Networking* 13 (6): 1205–1218.
36. Willinger, Walter, and Matthew Roughan. 2013. Internet topology research redux. *ACM SIGCOMM eBook: Recent Advances in Networking*.
37. Siganos, Georgos, et al. 2003. Power laws and the AS-level internet topology. *IEEE/ACM Transactions on Networking (TON)* 11 (4): 514–524.
38. Barabási, Albert-László, and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286 (5439): 509–512.
39. Roughan, Matthew, et al. 2011. 10 lessons from 10 years of measuring and modeling the internet's autonomous systems. *IEEE Journal on Selected Areas in Communications* 29 (9): 1810–1821.
40. Doyle, John C., et al. 2005. The “robust yet fragile” nature of the Internet. *Proceedings of the National Academy of Sciences of the United States of America* 102 (41): 14497–14502.
41. Willinger, Walter, David Alderson, and John C. Doyle. 2009. Mathematics and the internet: A source of enormous confusion and great potential. *Notices of the AMS* 56 (5): 586–599.
42. Doar, Matthew B. 1996. A better model for generating test networks. *Global Telecommunications Conference, 1996. GLOBECOM'96.'Communications: The Key to Global Prosperity*. IEEE.
43. Calvert, Kenneth L., Matthew B. Doar, and Ellen W. Zegura. 1997. Modeling internet topology. *IEEE Communications magazine* 35 (6): 160–163.
44. Donnet, Benoit, and Timur Friedman. 2007. Internet topology discovery: A survey. *IEEE Communications Surveys & Tutorials* 9 (4): 56–69.
45. Trajkovic, Ljiljana. 2010. Analysis of Internet topologies. *IEEE Circuits and Systems Magazine* 10 (3): 48–54.
46. Shavitt, Yuval, and Udi Weinsberg. 2011. Quantifying the importance of vantage point distribution in internet topology mapping (extended version). *IEEE Journal on Selected Areas in Communications* 29 (9): 1837–1847.

47. Motamedi, Reza, Reza Rejaie, and Walter Willinger. 2015. A Survey of Techniques for Internet Topology Discovery. *IEEE Communications Surveys & Tutorials* 17 (2): 1044–1065.
48. Wang, Xiaoming, and Dmitri Loguinov. 2006. Wealth-Based Evolution Model for the Internet AS-Level Topology. *INFOCOM*.
49. Rekhter, Y., and T. Li. 1995. RFC 1771. *A Border Gateway Protocol* 4: 1–54.
50. Hares, S., and Y. Rekhter. T. Li. 2006. A Border Gateway Protocol 4 (BGP-4). RFC 4271.
51. Rasti, Amir H., et al. 2010. Eyeball ASes: From geography to connectivity. *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. ACM.
52. CAIDA Data. AS to Organizations. https://www.caida.org/data/as-organizations.
53. Pansiot, Jean-Jacques, et al. 2010. Extracting intra-domain topology from mrinfo probing. In *International Conference on Passive and Active Network Measurement*. Berlin/Heidelberg: Springer.
54. Luckie, Matthew. 2010. Scamper: A scalable and extensible packet prober for active measurement of the internet. *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. ACM.
55. Mérindol, Pascal, et al. 2011. MERLIN: MEasure the router level of the INternet. *Next Generation Internet (NGI), 2011 7th EURO-NGI Conference on*. *IEEE*.
56. Quoitin, Bruno, et al. 2009. IGen: Generation of router-level Internet topologies through network design heuristics. *Teletraffic Congress, 2009. ITC 21 2009. 21st International. IEEE*.
57. Bowden, Rhys, Matthew Roughan, and Nigel Bean. 2014. COLD: PoP-level Network Topology Synthesis. *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*. ACM.
58. Coutinho, Emanuel Ferreira, et al. 2015. Elasticity in cloud computing: A survey. *Annals of Telecommunications-Annales Des Télécommunications* 70 (7–8): 289–309.
59. Whiteaker, Jon, Fabian Schneider, and Renata Teixeira. 2011. Explaining packet delays under virtualization. *ACM SIGCOMM Computer Communication Review* 41 (1): 38–44.
60. Shea, Ryan, et al. 2014. A deep investigation into network performance in virtual machine based cloud environments. *IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE*.
61. Xu, Fei, et al. 2014. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE* 102 (1): 11–31.
62. Callegati, F.; Cerroni, W.; Contoli, C.; Santandrea, G., Performance of Network Virtualization in cloud computing infrastructures: The OpenStack case. *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, vol., no., pp.132–137, 8–10 Oct 2014.
63. Chauhan, Maneesh. 2014. *Measurement and analysis of networking performance in virtualised environments*. MSc thesis.
64. Jayasinghe, Deepal, et al. 2014. Variations in performance and scalability: An experimental study in IaaS clouds using multi-tier workloads. *IEEE Transactions on Services Computing* 7 (2): 293–306.
65. Crisan, Daniel, et al. 2014. Datacenter applications in virtualized networks: A cross-layer performance study. *IEEE Journal on Selected Areas in Communications* 32 (1): 77–87.
66. Persico, Valerio, et al. 2015. Measuring Network Throughput in the Cloud: The case of Amazon EC2. *Computer Networks* 93: 408–422.
67. BMWG – Informational Draft.
68. Morton. Al, 2016. *Considerations for benchmarking virtual network functions and their infrastructure*. Work in Progress, August.
69. Fabini, Joachim, and Tanja Zseby. 2015. M2M communication delay challenges: Application and measurement perspectives. *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings. IEEE*.
70. Huang et al. 2015. Benchmarking methodology for virtualization network performance. Working in Progress, October
71. Whiteaker, Jon, Fabian Schneider, and Renata Teixeira. 2011. Explaining packet delays under virtualization. *ACM SIGCOMM Computer Communication Review* 41 (1): 38–44.

72. Dovrolis, Constantinos, Parameswaran Ramanathan, and David Moore. 2004. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions On Networking* 12 (6): 963–977.
73. Kapoor, Rohit, et al. 2004. CapProbe: A simple and accurate capacity estimation technique. *ACM SIGCOMM Computer Communication Review* 34 (4): 67–78.
74. Ribeiro, Vinay Joseph, et al. 2003. Pathchirp: Efficient available bandwidth estimation for network paths. Passive and active measurement workshop.
75. Wang, Han, et al. 2014. Timing is everything: Accurate, minimum overhead, available bandwidth estimation in high-speed wired networks. *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM.
76. Zhang, Ertong. 2015. *Bandwidth Estimation for Virtual Networks*. Computer Science and Engineering: Theses, Dissertations, and Student Research. PhD thesis, University of Nebraska-Lincoln, Paper 95.
77. Zhang, Ertong, and Lisong Xu. 2014. Network Path Capacity Comparison without Accurate Packet Time Information. *2014 IEEE 22nd International Conference on Network Protocols*. IEEE.
78. Spring, Neil, et al. 2004. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking* 12 (1): 2–16.
79. Knight, Simon, et al. 2011. The internet topology zoo. *IEEE Journal on Selected Areas in Communications* 29 (9): 1765–1775.
80. Durairajan, Ramakrishnan, et al. 2013. Internet atlas: A geographic database of the internet. *Proceedings of the 5th ACM Workshop on HotPlanet*. ACM.
81. The CAIDA UCSD Internet Topology Data Kit - <03/2016>, http://www.caida.org/data/internet-topology-data-kit.
82. Xen's Network Throughput and Performance Guide, https://wiki.xen.org/wiki/Network_Throughput_and_Performance_Guide. Accessed Sept 2016.

http://www.springer.com/978-3-319-54519-6

Performance Evaluation for Network Services, Systems and Protocols

Fernandes, S.

2017, XI, 175 p. 75 illus., 3 illus. in color., Hardcover
ISBN: 978-3-319-54519-6