



Information Society
Technologies



Title:		Document Version:	
D1		1.0	
QoS Monitoring and Measurement Benchmarking			
Project Number:	Project Acronym:	Project Title:	
IST-2000-26418	NGNI	Next Generation Networks Initiative	
Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type* - Security**:	
31/08/2002	08/10/2002	R – PP	

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other
 ** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Responsible:	Organization:	Contributing WP:
Elisa Boschi	(Fraunhofer FOKUS)	WP2

Authors (organizations):
 Tanja Zseby (Fraunhofer FOKUS), Florian Schreiner (Fraunhofer FOKUS)

Abstract:
 This document proposes an overview of the existing measurement techniques pointing out for each of them the most significant implementations and research projects. To make the overview more complete we have introduced information on traffic generators, time synchronization, sampling and standardization efforts as well as a section related to research project aiming to combine measurement with simulation techniques.

Keywords:
Measurement, Monitoring

TABLE OF CONTENTS

INTRODUCTION	4
1 NETWORK MEASUREMENT	6
1.1 PASSIVE VS. ACTIVE MEASUREMENT	6
1.2 ACTIVE MEASUREMENT	7
1.2.3 SURVEYOR (Advanced Network and Services/ Common Solutions Group R&E Network Measurements).....	7
1.2.4 Skitter Project (CAIDA)	9
1.2.5 Cisco Internet Performance Meter / Service Assurance Agent	10
1.2.6 RIPE TTM.....	11
1.2.7 Others.....	14
1.3 PASSIVE MEASUREMENT	16
1.3.1 CoralReef Passive Monitor (CAIDA).....	16
1.3.2 NeTraMet.....	17
1.3.3 Others.....	18
1.4 HYBRID MEASUREMENT (BOTH ACTIVE AND PASSIVE).....	18
1.4.1 Agilent Firehunter	18
1.4.2 NIMI (NSF/DARPA)	19
1.4.3 Others.....	20
1.5 OTHER TECHNIQUES	21
1.5.1 Mantra-Monitor & Analysis of Traffic in Multicast Routers	21
2 MEASUREMENT, MODELLING, AND SIMULATION	22
2.1 NMS ACTIVITIES.....	22
2.1.1 NETWORK MANAGEMENT AND CONTROL USING ON-LINE COLLABORATIVE SIMULATION	22
2.2 OTHERS	24
2.2.1 INTERMON.....	24
3 RELATED TOPICS.....	27
3.1 TIME SYNCHRONIZATION.....	27
3.1.1 NTP	27
3.1.2 GPS	29
3.1.3 DCF77.....	30
3.2 TRAFFIC GENERATORS.....	31
3.2.1 TG.....	31
3.2.2 NetSpec	31
3.2.3 Packet Shell.....	31
3.2.4 Rude/Crude	32
3.2.5 MGEN.....	32
3.2.6 UDPgen.....	32
3.3 STANDARDIZATION	32
3.3.1 IETF IPPM working group.....	32
3.4 SAMPLING.....	35

3.4.1 State of Art.....	36
3.4.2 Deployment of Sampling Techniques for packet selection.....	36
3.4.3 Sampling Methods	37
3.4.4 Sampling Parameters.....	39
CONCLUSIONS.....	42
REFERENCES	43
APPENDIX A	47

INTRODUCTION

The infrastructure of the Internet can be considered the cybernetic equivalent of an ecosystem. The last mile connections from the Internet to homes and businesses are supplied by thousands of capillaries, small and medium sized Internet Service Providers (ISPs), which are in turn interconnected by 'arteries' maintained by transit (backbone) providers. The global infrastructure of the Internet consists of a complex array of competing telecommunications carriers and providers, a very difficult infrastructure to analyze diagnostically except within the borders of an individual provider's network. Nonetheless, insights into the overall health and scalability of the system are critical to the Internet's successful evolution.

Attempts to adequately track and monitor the Internet were greatly diminished in early 1995 when the U.S. National Science Foundation (NSF) relinquished its stewardship role over the Internet. The transition to the actual competitive industry for Internet services left no framework for the cross-ISP communications needed for engineering or debugging of network performance problems and security incidents. Nor did competitive providers, all operating at fairly low profit margins and struggling to meet the burgeoning demands of new customers and additional capacity, place a high priority on gathering or analyzing data on their networks. This attitude is strengthened by the general lack of quality measurement or analysis tools to support these endeavors, and the absence of baseline data against which an analyst can track changes in the system's behavior.

As a result, today's Internet industry lacks any ability to evaluate trends, identify performance problems beyond the boundary of a single ISP, or prepare systemically for the growing expectations of its users. Historic or current data about traffic on the Internet infrastructure, maps depicting the structure and topology of this amorphous global entity, or projections about how it is evolving simply do not exist.

That is not to say that no measurement of the Internet occurs. There are numerous independent activities in the area of end-to-end measurement of the Internet. Typically spawned by end users with an interest in verifying performance of their Internet service, these measurements involve an end host sending active probe traffic out into the network and recording the delay until that packet returns to its source. Unfortunately such traffic measurements involve a large number of parameters that are difficult if not impossible to model independently, and the resulting complexity renders elusive any comparability or useful normalization of the gathered data. There are research groups trying to deploy technology and infrastructure to support more standardized measurement and evaluation of performance and reliability of selected Internet paths, and what specific segments of a given path limit that performance and reliability, but such efforts are slow and have thus far been unable to meet the needs of any of the user, research, or ISP communities.

Network measurements fall into two broad categories: Passive and active. Passive measurements depend entirely on the presence of appropriate traffic on the network under study, and have the significant advantage that they can be made without affecting the traffic carried by the network during the period of measurement. However, it can be

much more difficult or impossible to extract some of the desired information from the available data.

Active measurements, on the other hand, directly probe network properties by generating the traffic needed to make the measurement. This allows much more direct methods of analysis, but also presents the problem that the measurement traffic can have a negative impact on the performance received by other kinds of traffic.

This tension between measuring the performance of a network and actually using it to carry real traffic necessitates care in the design of any program of active measurements. In the remainder of this document we will highlight activities and outstanding problems in the areas of passive and active measurements and efforts to combine measurement and monitoring to other areas like modeling or simulation. We will give an overview on the existing tools and techniques, tools and techniques somehow related to measurement, and recent research projects.

1 NETWORK MEASUREMENT

In this section we will give an overview on the existing measurement techniques and existing tool, grouping them on the basis of the used technique.

1.1 PASSIVE VS. ACTIVE MEASUREMENT

Active measurements inject test traffic into the network in order to measure network characteristics. In contrast to this passive measurements rely on the traffic that already exist in the network only. In order to distinguish pure passive measurement methods from methods where the packets on the network are modified (e.g. a timestamp is added to the packet) we call the later *semi-active* measurement methods.

The development of active measurement methods to survey large parts of the Internet with a high precision is an active field of research [1], [2], [3], [4]. Active measurements give a prediction on how traffic would be treated in the observed part of the network. They are controllable experiments that can be performed at any time and with any kind of traffic that is of interest for the specific measurement objective.

Nevertheless active measurements are based on the sending of additional test traffic through the network under test. This test traffic always generates additional load on network links and routers and can significantly influence the measurement results. It may lead to additional costs if accounting is usage-based and it might bother intermediate providers. Especially if test traffic is not recognizable as such, providers may suspect an attack.

In contrast to this, passive measurements are based on the already existing traffic in the network. They provide a statement about the treatment of the current traffic at the moment in the observed network section. Since no test traffic is generated, passive measurements can only be applied in cases where the kind of traffic we are interested in is already present in the network. This is the case for most applications where a statements about the actual situation in the network is required (like SLA validation, traffic engineering).

Passive measurements mainly have been used for simple tasks like packet counting [9, 10, 28] and associated metrics, like volume, so far. The area of packet filtering and classification is an active research field that permanently comes up with new fast algorithms and methods to improve the filtering performance.

Passively measuring round-trip metrics is possible at a single measurement point by using packet pair matching techniques that rely on existing packet pairs like TCP-SYN/SYN-ACK, DNS request/response, etc. Nevertheless this method does only work for protocols based on packet pairs. It requires classification based on higher layer information and only measures round-trip metrics.

Difficulties arise if two measurement points are involved in the measurement like for pure passive one-way delay (OWD) measurements. An approach to realize passive OWD

measurements is to generate a timestamp and a unique packet ID for each packet at the two measurement points and send this information to a control instance that calculates the delay. The packet ID is needed to associate the timestamps from the different measurement points to the correct packet. The first problem that occurs here is the synchronization of clocks at the measurement points. Solutions include the usage of GPS, radio signals (like DCF77) and NTP based approaches. Different possibilities also exist for the generation of a unique packet ID.

Packet-IDs can be generated by using “compression” functions (hash functions, CRCs, etc.) over the invariant header fields and parts of the payload [5, 6]. Important is that (i) the probability of collisions (the generation of the same packet ID for non-identical packets) is low, (ii) the packet ID generation consists of operations that allow a fast and inexpensive realization and (iii) the size of the ID is as small as possible in order to reduce the amount of data captured. Methods for generating packet IDs are discussed below.

Further Issues:

- privacy issues when capturing traffic from customers (occur only in passive measurements)
- difficulties in packet event correlation when packets are lost or duplicated
- amount of data captured

1.2 ACTIVE MEASUREMENT

Active measurements inject test packets into the network and observe their behavior. Some active measurement tools require cooperation from both endpoints of the measurement. We summarize the existing active tools below

1.2.3 SURVEYOR (Advanced Network and Services/ Common Solutions Group R&E Network Measurements)

The Surveyor project [11] consistently measures end-to-end unidirectional delay, loss, and routing among a diverse set of measurement probes throughout the Internet. The goal of the project is to create technology and infrastructure to allow users and service providers (at all levels) to have an accurate common understanding of the performance and reliability of paths through the Internet. Surveyor measures the One-way Delay [7] and One-way Loss [8] metrics being developed by the IPPM (Internet Protocol Performance Metrics) working group of the IETF (Internet Engineering Task Force). Surveyor is currently deployed at 41 sites in the higher education and research community, including universities, US national labs, and international sites.

The Surveyor project's goal is to measure the performance of wide-area network connectivity among the participants, using well-defined metrics over long periods. If a

participant is measuring a sufficiently large number of paths, the performance data may be somewhat representative of the participant's general Internet connectivity.

Currently, the Surveyor infrastructure measures one-way delay, one-way loss, and route information along Internet paths. One-way delay and loss are measured according to the IETF IPPM One-way delay metric [7] and One-way packet loss metric [8]. Detailed methodology can be found in those documents. Route information is gathered by using a modified version of traceroute [12].

Delay and loss are measured using the same stream of active test traffic. A Poisson process on the sending machine schedules test packets.

Each test packet is of minimal size: 12 bytes of user data, essentially a sequence number and a timestamp. The test packets are sent using UDP, so the actual packet size, excluding any MAC header, is 40 bytes. The minimal test packet was chosen because the initial goal is to have a good understanding of how the underlying paths perform, to get a "baseline". As new analyses are developed, we will consider sending out packet trains or packets of variable size.

Because the Surveyor measurement machines are equipped with GPS hardware for time synchronization, timestamps carried by test packets have global meaning. The receiving machine determines the delay of the test packet directly: it simply subtracts the time in the received packet from the current time. Because the packets are timestamped, and the receiver knows the schedule they will be sent on, the receiver knows when a sent packet does not arrive. This is how packets are known to be lost. In the current implementation, if a packet does not arrive within 10 seconds, it is deemed to be lost. Lost packets are treated as sent packets with infinite delay.

Route information is gathered for the same paths as delay and loss. The information is gathered using a modified version of traceroute [12]. The modifications are threefold. First, we are more persistent in the face of failure. If an ICMP TTL exceeded message is not forthcoming, we try ten rather than the default three times. Second, we are less persistent in the face of success. Rather than send out three probes for each TTL, failure or success, we stop sending probes once we get a success. Third, we do not keep any of the timing information. Round-trip times generated by traceroute are notoriously inaccurate, since the response message is generated by an exceptional condition at each hop along the path. Rather than rely on this data, we do not keep it at all.

Route information is gathered based on a schedule generated by a truncated Poisson process.

There are three major components of the Surveyor infrastructure: measurement machines, the database, and the analysis server. The measurement machines are the dedicated machines which we deploy at participating sites. The measurement machines report back to a central database that catalogs all the performance data. Finally, analyses are performed by, and made available through an analysis server. The analysis server is accessed via HTTP.

Each Surveyor measurement machine is a Dell desktop PC with 200, 333, and 400 MHz Pentium processors deployed in the field. In addition to the basic PC, each machine has an appropriate network interface card, and a GPS card. They currently support Ethernet (both 10bT and 100bT), FDDI, and OC3 ATM network interfaces. They use GPS receiver cards manufactured by TrueTime in the field. These cards keep the current time on-board (so a read on the system bus is required to read the time). They have both ISA

and PCI flavor cards. The GPS cards use an antenna and GPS daughter board manufactured by Trimble. An RG-59 or RG-58 coaxial cable, depending on wire run length, is used to connect the antenna to the card. The machines run the BSDI version 3.1 operating system.

The Surveyor measurement machines collect performance data and buffer them to local disk. Once every few minutes the measurement machines are polled for new performance data; if there is data, it is uploaded to the central database.

The daily summary plots, traceroute data, and the "raw" daily summary statistics are made available using a http server. Primary access is to the plots and traceroute data; the raw data is intended for other analysis programs. There are three daily plots available for each path: one showing summary statistics on delay, one showing loss throughout the day, and one showing a histogram of delay values. Currently there are four general views of the plots available: calendar, per-site, per-path, and animated.

1.2.4 Skitter Project (CAIDA)

CAIDA's skitter tool [13] actively collects topology and performance data from approximately 22 sources (as of April 2001) around the world to hundreds of thousands of destinations in Ipv4 address space. The data collected by skitter is useful for more than just topological visualization, however, since then it also contains a lot of information about the performance of specific paths through the Internet.

Skitter works using a process somewhat analogous to medical x-ray tomography, a technique where a three-dimensional image is achieved by rotating an x-ray emitter around the subject and measuring the intensity of transmitted rays from each angle, and then reconstructing the resulting two-dimensional images into a three-dimensional object. Geologists rely on similar techniques to build models of seismic activity using cross-section images (slices) of the earth. Data gathered from tomographic scans play an important role in developing models to analyse and predict select phenomena.

CAIDA is currently using skitter to gather infrastructure-wide (global) connectivity information (what's connected to what?), and round trip time (RTT) and path data (how does a packet get from A to B and how long does it take?) for more than 30,000 destination hosts from six source monitors spread throughout the United States, with additional monitors deployed in the U.S., Europe, and Asia in mid-1999.

Skitter measures the Internet path to a destination by sending several ICMP echo request packets towards the destination host in a similar fashion to the common 'ping' utility. However, skitter sends these packets with very low 'time to live' (TTL) values. Every router in the Internet automatically decrements the TTL value of each forwarded packet as part of an overall scheme to prevent persistent traffic looping. If the TTL value reaches zero, the router will discard the packet and send an error notification back to the sender. skitter sends a series of probe packets from the measurement host with progressively larger TTL values, and as each error message is received the measurement host is able to determine the path taken through the network from the source to the destination. This is essentially the same procedure as used by the traceroute utility.

Probing paths from multiple sources to a large set of destinations throughout the current Internet address space allows both topological and geographical representations of a

significant fraction of Internet connectivity, the latter admittedly constrained by the abysmal lack of geographic mapping data for Internet address space. Supporting tools also analyse the frequency and pattern of routing changes (when and how often are alternative paths used between the same two endpoints?)

Like any active measurement program, it is essential that skitter measurements impose only a minimal load on the infrastructure as it takes its measurements. Skitter probe packets are very small, 52 bytes in length, and typically only probe destination hosts at approximately hourly intervals.

1.2.5 Cisco Internet Performance Meter / Service Assurance Agent

The Cisco Internet Performance Monitor (IPM) [14] is an application for monitoring the performance of multi-protocol networks. It is announced to fulfill many tasks including troubleshooting network performance problems between devices, raising alerts if threshold are exceeded, monitoring latency, availability, jitter, packet loss, and errors between two network points, and providing web-based access to long-term network information. To fulfill all these chores, the IPM solution consists of three parts: the IPM server, the IPM client applications, and the Service Assurance (SA) Agents.

The IPM server provides central services and functions as a measurement database. It manages the exchange of data between the measurement devices and its central database. Either the specialized client software or a Web browser can be used to gather the required performance data from the IPM server. Through the Web interface it is possible to access the measurement definitions, view Web-based reports of the performance data, and to access IPM data. Neither the server nor the client are a source for performance measurements. They are only used to configure an SA Agent, which actually performs the measurements and is running on a Cisco router. This is at the same time a prerequisite for using the Internet Performance Monitor: an SA Agent enabled Cisco router.

Therefore the Service Assurance Agent is not a part of the IPM application distribution, but it is a feature embedded into the Cisco IOS software of routers or Catalyst Route Switch Module. This SA Agent is the source for all measurements and needs to be also the target for certain measurements. SA Agent is capable of performing tests at the network (IP), the transportation (TCP, UDP) and the application layer. The support protocols at the application layer include: HTTP, DNS, DHCP, and some others. Usually availability and overall service delay, like connecting to a web server and downloading a specified page, are measured for the application protocol measurements. For the measurement of the network layer, e.g. the Internet Protocol, SA Agent uses tools very similar, or even identical to ping and traceroute. The simple measurements at the transportation level (TCP, UDP) include connectivity and "service" latency, e.g. how long does it take to establish a TCP connection to a certain server. Additionally the SA Agent can make enhanced UDP measurements, which include the round-trip latency, the per-direction loss, the per-direction jitter, the network availability and network errors. While the simple tests can use any IP-enabled device, that is offering the required services, e.g. is available to handle TCP requests, the enhanced UDP measurements can only be taken between Cisco's SA Agents.

The SA Agents usually can make several measurements per hour, but if request performs tests up to every 10 seconds. The Internet Performance Monitor polls the probe routers once every hour to collect the summarized statistics since the last poll. It is also possible to use a Real Time Statistics window, which displays the measurement results immediately after the tests, but this data is not archived in the IPM database, only the summarized data from the hourly polls is stored.

1.2.6 RIPE TTM

The RIPE Test Traffic Measurement (TTM) is a service offered to members wishing to host a RIPE NCC Test-box. Currently 60 Test-Boxes are participating in the Ripe Network, mainly in Europe.[44] Test boxes actively measure Internet delays and losses by sending time-stamped packets to each other. The goal of the project is to do independent measurements of connectivity parameters, such as delays and routing-vectors, in the Internet.

TTM's goal is to provide standardised metrics for one-way delay and one-way packet loss between measurement devices in a format, easily understood by users. Since more and more of Internet routing is becoming asymmetrical, i.e. packets use different paths between two nodes depending on the direction of traffic, One-way measurements are vital. The TTM project complies with standards that are maintained and developed by the IETF's IP Performance Metrics Working Group(see section 3.3 for details) (RFC's 2330, 2678 through 2681). [45]

Measurement Topology and metrics:

Currently, the RIPE TTM infrastructure measures one-way delay, one-way loss, and route information along Internet paths. One-way delay and loss are measured according to the IETF IPPM One-way delay metric and One-way packet loss metric. Route information is gathered by using traceroute [Jacobson89]. A further IPDV analysis is done on the Data-analysis Server running Cern's ROOT which can be accessed individually via Http. Gathering further bandwidth data with pathrate is well studied, but yet not integrated, since there is no IPPM standard for bandwidth measurements yet [47]. Since Ripe and Surveyor claim to measure IPPM metrics, it is worth to compare the measured data of both. A one month analysis study compared both approaches and basically found out, that both measure the same.[48, 49]

Delay and loss are measured using the same stream of active test traffic. A Poisson process on the sending machine schedules test packets. Forthcoming integrations will offer different scheduling approaches, which will be made user-configurable [50]. These should include different packet-sizes, sampling frequencies and even simulation of real-time applications.

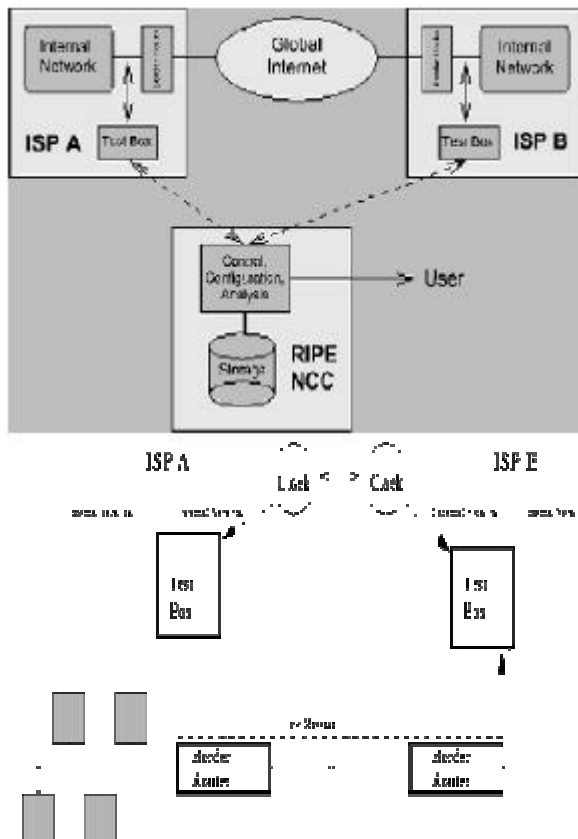


Figure 1: RIPE 158, Design document (June 1997) [46]

Packet-size and Sampling:

The probe packets are 128 bytes long, and contain a UDP frame with destination port 6000, and a UDP payload of 100 bytes. This is the TTM systems Type-P definition for the Type-P-One-way-Delay metric framework. The sampling process is Poisson scheduled, with a mean rate of 3 Packet/minute. [51]

Timekeeping and Accuracy:

Each TTM Box is equipped with a GPS receiver. This receiver outputs a PPS-Signal (pulse per second) on the DCD control line of the PC's serial port, generating interrupts, which are used by NTP to phase-lock the BSD-kernel system clock. A user-space daemon schedules test probes. When a probe is to be sent, the packet is built in user-space, the kernel system clock is read, the time is inserted into the packet, and it is passed into kernel-space to be queued for transmission. After the packet is received by the destination, it is time-stamped again with the kernel clock, which is synchronised by the same GPS mechanism. A One-way-Delay measurement is produced by subtracting the time the packet was sent from its arrival time. [51]. Lost packets are removed from the calculation.

Route information is gathered for the same paths as delay and loss. The information is gathered using traceroute [Jacobson89] every 10 minutes . Round-trip times generated by traceroute are notoriously inaccurate, rather than rely on this data, Ripe does not store the Data at all.

Once a day, data collected at the TTM machines is retrieved to a central point . One-way-Loss can also be determined, by comparing the records from the source for packets sent, and from the destination for packets received. [51]

The measured errors show some expected systematic offset, and a variable error of less than 50 μ s at a 95% confidence level. Even better results can be achieved using dag cards.[52,53]

Infrastructure:

There are three major components of the Ripe infrastructure: measurement test-boxes, the MYSQL database, and the ROOT analysis server.

The measurement machines are the dedicated machines which Ripe deploys at participating sites. The measurement machines report back to a central MYSQL-database that catalogs all the performance data.

Finally, analyses are performed by, and made available through an analysis server, running ROOT from CERN. The analysis server is accessed via Http, Perl Scripts offer configuration and scaling of the Plots.



Figure 2: RIPE Plots on Demand Service

The Test-Box:

Ripe offers two ways of hosting a Test-Box. Members have the choice to order a pre-configured Test-Box from Ripe or to build their own Test-Box.

Currently Ripe supplies their Customers with “GroupD Test Boxes” [54], that include a Dell PowerEdge 350 (850 Mhz Celeron, Alteon Copper Gigabit Ethernet Intel Pro 100+, 100S, Dual-Port Pro) with standard accessories and a Trimble Acutime2000 GPS antenna.

Ripe recommends a minimal System Configuration for “home-built” Boxes of:

Intel Pentium-MMX (200 MHz or more), Pentium-II (200 MHz) or more, Celeron (300 MHz or more), 32 Mb minimum RAM, 3COM 3c905/3c905b/3c905c Ethernet Cards,

Trimble Palissade [55] or Acutime 2000 [56] Gps Receivers, DB25 cat5 UTP cable. The machines run FreeBSD 2.2.8. The Software includes the NTP-Package from Mills [57] to synchronise the internal Clock with PPS, the timestamping, tracerouting, measurement component, and an auto-alarming Daemon based on CFEngine, which is programmed to send alarm messages if either clock-synchronisation errors occur or measurement data exceeds a certain threshold (if the recent results are outside the expected range) [58].

The Database:

The Ripe measurement machines collect performance data and buffer them to local disk. Once a day the measurement machines are polled for new performance data; if there is data, it is uploaded to the central MYSQL database. Due to the fact that currently each Box collects Data from measurements from all other Boxes, Ripe is expecting a N^2 Problem [59], since currently 60²=3600 paths are measured and setup for more boxes is planned. This leads not only to a storage problem, since several Tbytes have to be collected each year, it also introduces more and more artificial traffic and data, that has to be computed on every single box. Data the most of which will never be looked at by a human. This is why Ripe plans to give members the choice which paths should be measured, so that they can configure the boxes of interest [60].

The Analysis-Server:

Ripe gives its customers two separate options of analysing measured data, one for immediate check [61] of recently measured data on the boxes and one for long term data analysis.

For immediate checks, the test boxes are equipped with a web interface [62], that gives graphical and interactive immediate feedback on current data. The measured data can be analysed with the RRD Tool [63]. It offers scaleable loss and delay plots [64], plots about the System Status [65] (Cpu-Load, Memory availability and Processes created) and Plots about the GPS-Status [66](Reception, availability and S/N ratio of each satellite).

For extensive analysis Ripe provides their customers with a centralised Server. The server runs Cern's ROOT [67] for data analysis, which offers configurable plots on demand. These Plots include daily, weekly, monthly Delay and -Loss plots [68], trend plots [69] which analyse certain percentiles, IPDV-plots (Delay Variation also known as Jitter), traceroutes [70] and additional statistics such as Mean, RMS, 2.5 and 97.5 percentiles of delays, min. and max. hops, packet loss statistics, Connectivity statistics and over-all statistics.

Primary access is to the plots and traceroute data, the raw ROOT data is intended for other analysis programs, but can be made available to members on demand via FTP, who can do additional analysis with their own installation of ROOT.

1.2.7 Others

1.2.7.1 TRIUMF Network Monitoring

his Canadian national research facility [16] uses Perl script to trace paths toward specific nodes of interest to TRIUMF users. Packet loss and delay measurements are summarized

and graphed daily from pings occurring at 10 minutes intervals. Traceroute data for hopcount statistics and graphs is gathered four times daily. Network topology visualization maps are generated from the traceroute data.

1.2.7.2 pathchar

pathchar [17] estimates performance characteristics of each node along a path from a source to destination. Leverages the ICMP protocol's Time Exceeded response to packets whose TTL has expired. Sending a series of UDP packets of various sizes to each hop, pathchar uses knowledge about earlier hops and the round trip time distribution to this hop to assess incremental bandwidth, latency, loss, and queue characteristics across this link.

1.2.7.3 pchar

pchar [18] is similar to pathchar, pchar attempts to characterize the bandwidth, latency, and loss of links along an end-to-end path through the Internet. pchar works on both IPv4 and IPv6 networks. Written in C++, recent additions to pchar include: an SNMP query feature, and better IPv6 detection at configure-time.

1.2.7.4 netperf

netperf [19] is a benchmark that can be used to measure the performance of many different types of networking. It provides tests for both bulk data transfer and request/response performance using either UDP or TCP.

1.2.7.5 fping/ping

fping [20] is a ping variant suitable for use in scripts. fping will issue ICMP echo requests to a list of hosts in round-robin fashion. fping output is meant to be parsed by scripts. Like the standard ping tool it measures hop-to-hop latency and packet loss.

1.2.7.6 echoping

echoping [21] is a utility for measuring TCP/UDP latency by sending to an arbitrary (default 'echo') port. It includes support for testing HTTP query latency.

1.2.7.7 PingER

The PingER (Ping End-to-end Reporting) project [22] deploys software to monitoring sites throughout the high-energy physics research (HEP) community. It too aims to make consistent, long-running measurements. Surveyor measures unidirectional delay and loss instead of round-trip delay and loss. Surveyor also uses a different measurement schedule: PingER sends a series of ICMP echo request messages grouped together on a fixed schedule; Surveyor constantly sends out small UDP packets on a Poisson schedule.

1.2.7.8 traceroute

traceroute [12] directs a packet to each router along a path without actually knowing the path, by setting the IP TTL field from 1 to n until the ultimate destination is reached. Upon receiving a packet with an expired (0) TTL, the hop generates an ICMP Time Exceeded response back to the source, thus identifying the hop and its round trip delay. Each UDP packet is sent to a probably-unused port, so when the destination receives the packet it responds with ICMP Port Unreachable.

1.2.7.9 Brix 1000 Verifier

Brix 1000 Verifier [23] creates a regional ISP service demarc point. Calculates network and application statistics by measuring characteristic application transaction times. Measures round-trip latency, jitter, and packet loss. Runs active test modules (e.g., VPN, VoIP, web, streaming media, email, news, differentiated services, DNS, SNMP, router status) as specified by BrixWorx software. It is a built-in hardware packet-timestamp engine. Optional GPS module provides worldwide, accurate synchronization of timestamps to sub-millisecond precision. Both active probes using simulated transactions and passive measurement of actual transactions is supported. This customer-located equipment separates packet-forwarding and network-testing data paths to ensure wire rate traffic flow under all conditions. Test suites are automatically provisioned by BrixWorx software. New tests can be dynamically added without resetting or data loss.

1.2.7.10 Chariot

Chariot [24] evaluates the performance of networked applications, performs stress tests of network devices and predicts networked application performance prior to deployment. You can use Chariot's performance data to optimise your network and measure the performance impact of proposed network changes.

1.3 PASSIVE MEASUREMENT

Passive measurements observe actual traffic without perturbing the network. Passive monitors must process the full load on the link, which can be problematic on high speed links. While passive measurement does not require cooperation or coordination from end hosts, the quality of passively gathered data critically depends on monitor placement, which does require cooperation from network operators. We propose an overview of the state of the art here below.

1.3.1 CoralReef Passive Monitor (CAIDA)

CoralReef [25] is a comprehensive software suite developed by CAIDA to collect and analyze data from passive Internet traffic monitors, in real time or from trace files.

Realtime monitoring support includes system network interfaces (via libpcap), FreeBSD drivers for Apptel POINT (OC12 and OC3 ATM) and FORE ATM (OC3 ATM) cards, and support for Linux drivers for WAND DAG (OC3 and OC12, POS and ATM) cards. The package also includes programming APIs for C and perl, and applications for capture, analysis, and web report generation. Plans include development of an OC48mon monitor (development is continuing jointly with the University of Waikato and others under CAIDA's NGI project), and eventually an OC192mon monitor as well. The CoralReef suite also includes software for analysis of traces collected by these type of monitors.

CoralReef is a suite of flexible, high performance Internet traffic data collection and analysis tools. The CoralReef suite is a totally passive monitoring system which does not require any additional network infrastructure and does not increase network traffic or interfere with other network devices. Monitoring of optical networks is done with an optical splitter which diverts a small fraction of the light from the optical fiber to the monitor device.

CoralReef provides a set of drivers, libraries, utilities and analysis software for passive network measurement. The package includes many ready made solutions, but is still evolving. We expect to refine and greatly enhance CoralReef's functionality and ease of use in the near future. The CoralReef software is known to work under FreeBSD (2.7, 2.8, 3.0, 3.1, 3.2, 3.4), Linux (2.0.36, 2.2) and Solaris (2.5, 2.6), and is expected to work under most other Unix-like systems. The Apptel POINT and FORE ATM card drivers work only under FreeBSD; the DAG card is supported only under Linux. This release includes utilities that are under development.

1.3.2 NeTraMet

NeTraMet [9, 10] is an open-source implementation of RTFM, the IETF standard, generalized architecture for measuring traffic flows. NeTraMet is an accounting meter which runs on a PC under DOS or a Unix system. It builds up packet and byte counts for traffic flows, which are defined by their end-point addresses. Addresses can be Ethernet addresses, protocol addresses (IP, DECnet, EtherTalk, IPX or CLNS) or 'transport' addresses (IP port numbers, etc), or any combination of these. The traffic flows to be observed are specified by a set of rules, which are downloaded to NeTraMet by a 'manager' program. Traffic flow data is collected via SNMP from NeTraMet by a 'collector' program. NeMaC, a combined manager and collector program, is supplied with NeTraMet. It downloads rules to meters, and collects data from them. Although a meter may only have one manager, its data can be collected by several collectors, which do not have to be synchronised. NeMac can manage and collect data from an arbitrary number of meters.

The format of NeMaC's collected flow data files is very general; the contents of data lines in the file is completely specified by the user. ASN.1 opaque objects are used to retrieve flow data so as to minimise the overheads in using SNMP for this purpose.

NeTraMet provides a valuable tool for analysing network traffic flows, and should prove to be of interest to anyone interested in network monitoring, capacity planning, performance measurement, etc.

1.3.3 Others

1.3.3.1 WAND

The WAND (Waikato Applied Network Dynamics) [26] project has made some unidirectional latency and loss measurements [27]. Like Surveyor, WAND uses GPS to synchronize clocks. WAND can capture packets passively. In particular, it was designed to capture ATM cells passively, recording a timestamp and signature of each cell. These signatures can be correlated off-line to find one-way delays accurate to 10 nanoseconds. WAND has also developed an Ethernet interface that uses the same technique, and is working on an IP packet-over-SONET interface. Surveyor uses active measurements, and thus does not have to concern itself with the problems of capturing user data.

1.3.3.2 RIPE-NCC (Reseaux IP Europeens Network Coordination Center) RIS-Routing Information Service Project.

RIPE [15, 34] is a collaborative organization open to groups and individuals operating wide area IP networks in Europe and beyond. The objective of the RIPE Routing Information Service (RIS) Project is to collect default free inter-domain BGP routing information. RIS uses multiple route collectors to integrate multiple views, and archives routing updates to support longer term trend analysis. A prototype web interface enables data archive queries by domain prefix, AS number, or timeframe.

1.4 HYBRID MEASUREMENT (BOTH ACTIVE AND PASSIVE)

1.4.1 Agilent Firehunter

Firehunter [29] is a customizable, scalable, multi-platform Service Management solution for the Internet Service Provider and for the network departments of medium to large sized enterprises. Firehunter can automatically measure, monitor, and verify the quality of the involved Internet services. The product is targeted for three different sized customers, there is a entry level product, which only provides core functionality, a mid-sized product, which provides support for all basic Internet services, and can be easily applied to networks of up to 40 servers. Finally there is a professional version, which emphasizes large networks, with more than 40 servers and companies that want to use advanced e-business services like Virtual Private Networks (VPN) or conduct E-Commerce over the Internet.

The Firehunter architecture consists of three building blocks: the graphical user interface (GUI), the diagnostic measurement server (DMS) and the independent measurement agents. The GUI application allows access to summaries and detailed information regarding the measured quality of service of the network. At the same time the GUI

allows for the customization of measurement tasks and for the adapting of reporting functionality. The DMS performs analysis and correlation of all the data collected by the measurement agents. This data is stored and used to calculate thresholds and baselines for the regular usage. Reports or alarms are triggered, if certain parts show an abnormal behavior compared to these thresholds and baselines. To execute the measurement tests the agents are used. The agents can either perform passive tests, like monitoring certain files or system states, or perform active tests, like creating a stimulus for server measurements or emulating a client.

Firehunter supports active and passive measurements. Most of the available tests are for application protocol testing, for example it passively monitors the number of connections for a HTTP port and actively tests the availability and response times. In this case a Firehunter agent simulates a web browser and measures the time it takes for downloading a web page from a server. The Firehunter agents can simulate other applications like mail and news. For server measurement Firehunter relies on 'vmstat' a tool that monitors for example the CPU idle time, the number of process to be run, and the free memory. Therefore the agent needs to be installed on the server under test for this kind of passive measurement.

Permanent monitoring and measuring are the key concepts of the Firehunter architecture, through which the measurements baselines are established. These are the foundation for user-defined thresholds. Through the definition of four different thresholds the severity of a fault is estimated. If a certain threshold is exceeded, Firehunter triggers the corresponding event or alarm. The triggered event is user configurable, which allows for a many different reactions to a problem. A good example is that an operator at the GUI interface of Firehunter further investigates the problem, to find the primary cause of the failure and to hopefully solve it. Except for the entry level version, Firehunter supports the monitoring and the generation of reports for Service Level Agreements. This is a special form of contract, which identifies certain performance values or a certain Quality of Service. This for example enables a Internet Service Provider not only to offer best effort, but to guarantee certain performance parameters. These can include availability, response times, and other performance related values.

1.4.2 NIMI (NSF/DARPA)

NIMI [30] is designed for large scale measurements over the Internet. It was developed to allow researchers to do their measurement around the world and not only in their lab. The vision of more than 1000 measurement probes all within differently administered domains has influenced NIMI intensively. It supports encryption for all data transfers and allows access authentication. On the other hand, NIMI does not come with a build in measurement method. It is only designed as an infrastructure for many different measurement methods.

NIMI can be conceptually divided into two components: the individual NIMI probes and the external control components. While the probe only fulfils the very limited but demanding task of performing and recording the measurement, it does not analyze nor display any results. The external components fulfil the task of calculating the final

measurement results, combining differently gathered data, and to control the measurement procedure. Additionally they govern the access control for the corresponding NIMI probes.

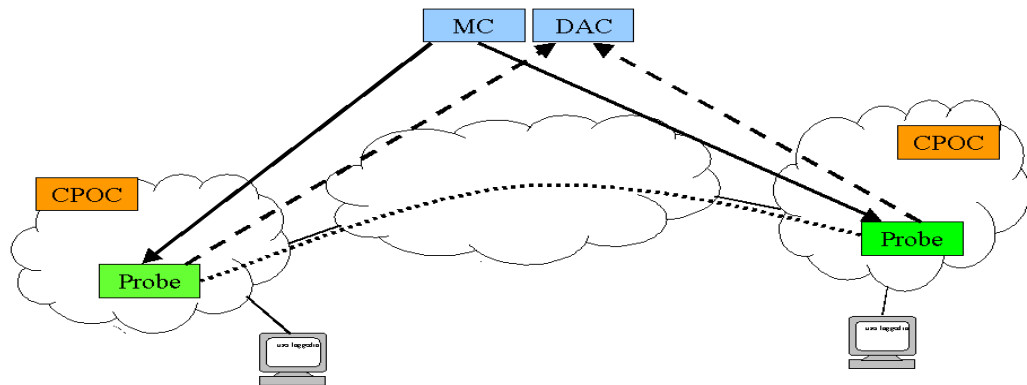


Figure 3: NIMI Application Components

The application itself consists of four main building blocks which communicate over TCP/IP with each other. These include the “actual” measurement probe daemon (NIMI), a client for requesting measurements (mc), a program for receiving and analyzing of measurement results (dac), and a daemon for management, control, and configuration of “NIMI probes” (cpoc).

For the measurement of Internet metrics, NIMI relies so far on “traceroute”, “treno”, and a slightly modified version of “ping”. Traceroute is used to identify the route the test packets take. Treno is a tool to measure bulk transfer throughput. It operates by sending a stream of UDP packets to which the destination returns either ICMP TTL expired or ICMP port unreachable messages. Ping is used to perform delay and connectivity measurements

1.4.3 Others

1.4.3.1 Internet2 (Abilene)

Abilene [31] is an advanced backbone network that connects regional network aggregation points (gigapops) to support the work of Internet2 universities as they

develop advanced Internet applications. Network monitoring activities include active multicast ping (mping) measurements as well as SNMP based collection of data.

1.4.3.2 NWS-Network Weather Service (NPACI)

The Network Weather Service [32] is a distributed system that provides NPACI users a way to select high performance computing resources on which to run their applications. NWS periodically monitors and dynamically forecasts performance to various NPACI network and computational resources over a given time interval (e.g., seconds, minutes, hours). NWS uses a distributed set of performance sensors from which it gathers readings of instantaneous conditions. Numerical models and statistical methods are then used to generate 24 candidate forecasts of conditions; the forecast with lowest statistical error is then presented to the user. NWS aims to forecast TCP/IP end-to-end throughput and latency from a user application perspective.

1.5 OTHER TECHNIQUES

1.5.1 Mantra-Monitor & Analysis of Traffic in Multicast Routers

The Mantra [33] java-based tool is used to monitor various aspects of multicast behaviour at the router level. The results from Mantra are aimed to depict the snapshots of various constituents of the multicast infrastructure from the point of view of the monitored routers. Visualization snapshots and accompanying tables are updated every 15-30 minutes and may be consulted at [mantra].

The Design of Mantra is based on four major modules each of which is responsible for performing a major monitoring task. With the help of the Data Collector, Mantra collects monitoring data from multicast routers and prepares the collected data for further processing. Some of the main functions of this module include acquiring data from the router, assuring the accuracy of data transfer and pre-processing the raw data. Collected data is converted to Mantra's local data format. A set of tables provides a standard framework for storing the monitoring information. A second component, the router-table processor, maps each of the pre-processed data tables to the corresponding local table(s).

This process also involves estimating the missing data sets, estimating duration of various entries in the table, and removing noise from the data set.

The Data Logger stores the processed data. This data can be later used for detailed off-line analysis as well as for long-term trend analysis. The fourth component, the Data Analyzer, is a collection of modules that allow Mantra to analyze monitoring data and generate online results. Every time a new set of data tables is processed and logged, analysis modules are invoked. This provides a realtime set of results for display. Some of the analysis that Mantra provides include session analysis for monitoring multicast group membership patterns, analysis of traffic local to the collection point, and route analysis for monitoring connectivity of multicast networks.

2 MEASUREMENT, MODELLING, AND SIMULATION

In this section we try to show how measurement could be useful or combined with other research fields. We discuss about some research projects combining monitoring and measurement with modeling and simulation.

2.1 NMS ACTIVITIES

In the last years DARPA has sponsored a number of projects in the areas of network monitoring, modeling and simulation [35]. In the following sections we describe those that we consider the most interesting related to measurement and to combining measurement with other areas (i.e. modeling and simulation). More information on the NMS Projects can be found at <http://www.darpa.mil/ipto/research/nms/>.

2.1.1 NETWORK MANAGEMENT AND CONTROL USING ON-LINE COLLABORATIVE SIMULATION

This project [36], carried on at the Rensselaer Polytechnic Institute, aims to develop a system of collaborative, on-line simulators to support a suite of distributed network management and control functions. The on-line collaborative simulators they propose can predict the network performance under different sets of traffic parameters therefore enabling an automatic network management to select an optimal parameters in response to the changing medium range temporal traffic patterns.

The basic idea of the architecture is illustrated in figure 1. The collaborative on-line simulation architecture operates in the management plane and interfaces with the control plane of the network. In particular, it does not interfere with the packet-by-packet data-

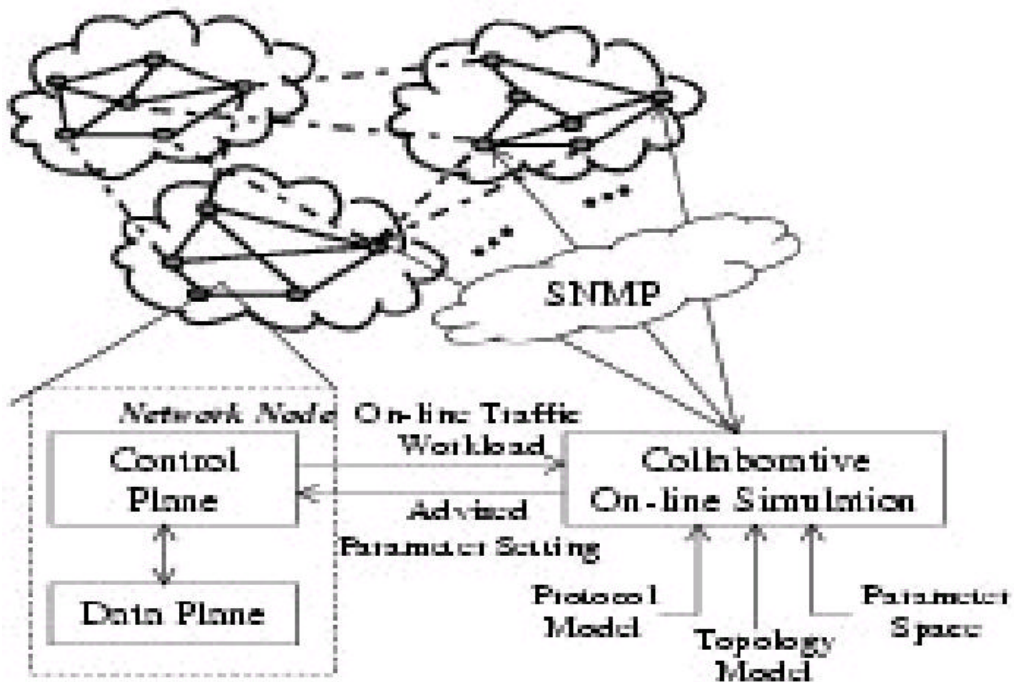


Figure 1: Collaborative on-line simulation scheme

Figure 4: Collaborative on-line simulation scheme

plane operation of the network. The architecture is mainly composed of autonomous on-line simulators, which continuously monitor and model the network conditions and topology. Based upon the on-line model of traffic and topology, the simulators can execute simulations to evaluate the performance of the network for a given set of protocol parameters. The assumption is that network control protocols (e.g.: traffic management, routing protocols) are sensitive to traffic loads and a subset of their parameters. The goal then is to have the on-line simulation system search for better parameter settings applicable to the current traffic and topology mix. The on-line simulation scheme uses a best effort parameter search strategy whose emphasis is not on “full” optimization, but on continuously and increasingly moving the system toward a “better” operating point.

Collaborative simulators (whose structure is shown in figure 2) monitor the network conditions, collect the relevant information, communicate with other simulators and execute collaborative on-line simulation. Based on the simulation results (that can be seen as predictions), the simulators keep tuning the network parameters to the better operation point to fit the current network conditions.

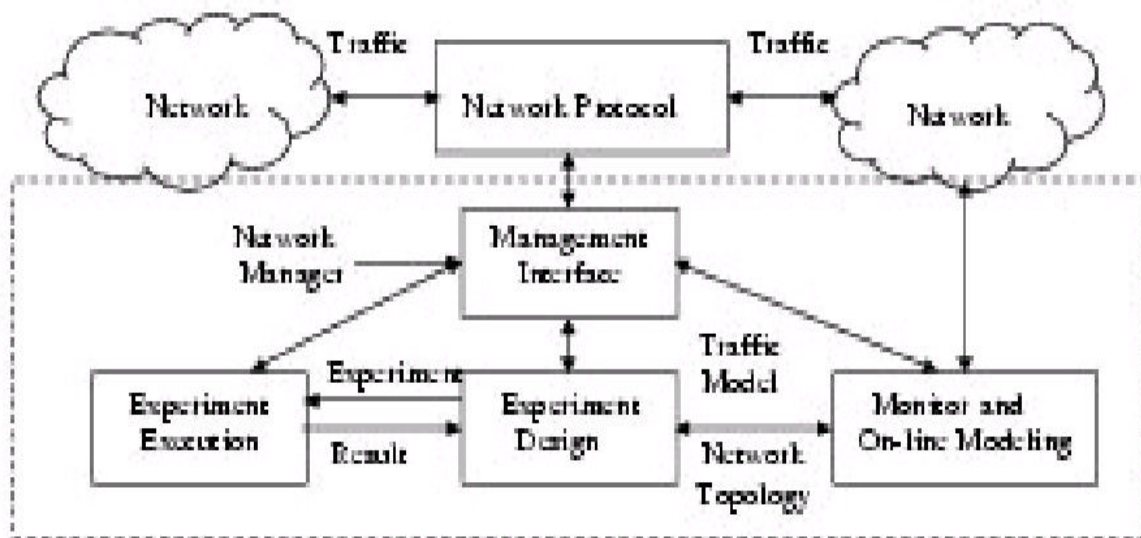


Figure 2: Structure of on-line simulator

Figure 5: Structure of on-line simulator

2.2 OTHERS

2.2.1 INTERMON

The main goal of the INTERMON [37] architecture is the integrated measurement, modeling and visual data mining for analysis of inter-domain QoS and traffic issues based on integrated DataBase (IM-DB). The INTERMON DataBase is obtained for a specific usage of the system (ISP operator or end-users).

The users (customers) may request the execution of monitoring, modeling, visual data mining and configuration tasks through a common Graphical User Interface. Figure 3 presents the main components of the architecture.

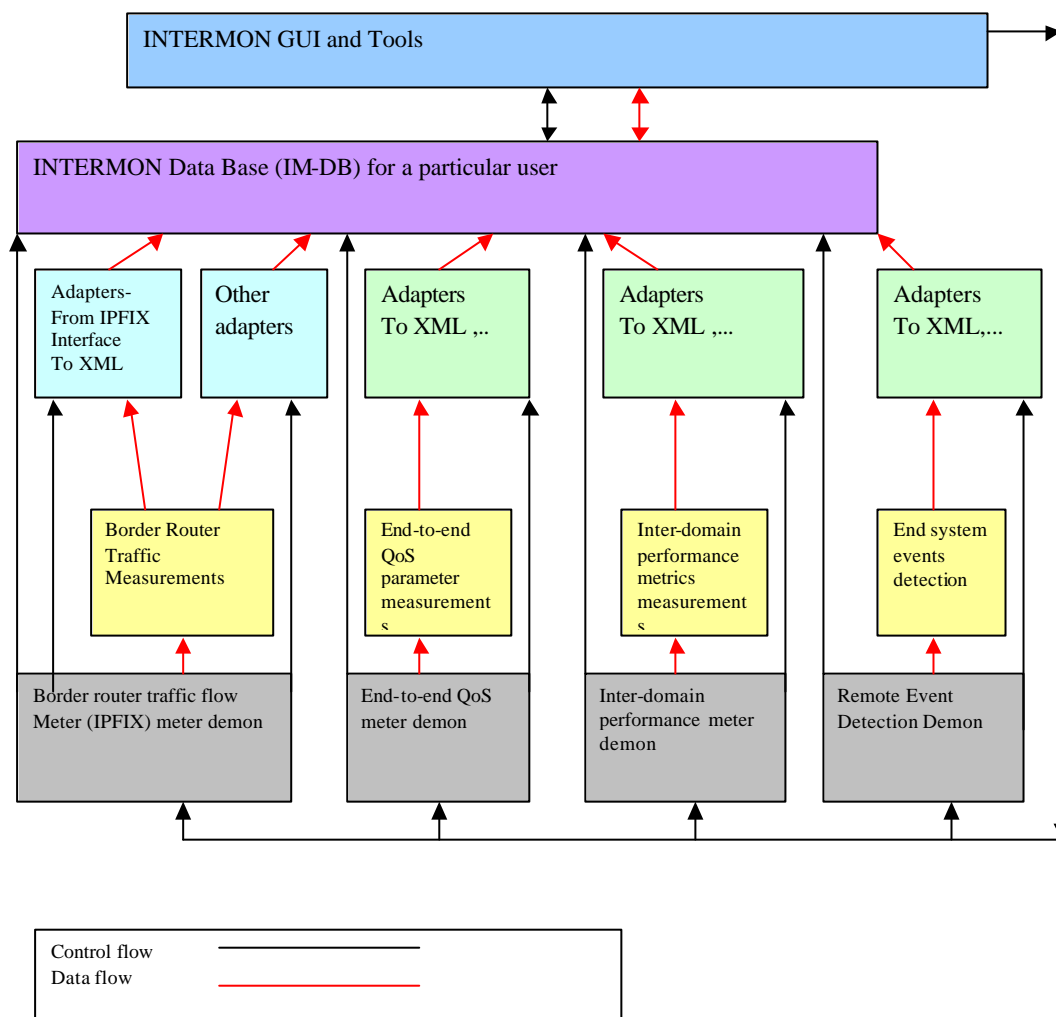


Figure 6: INTERMON tools, IM-DB and relationships to distributed measurement sources

The INTERMON data base (IM-DB) is a relational data base which relate topological, measurement and modelling information from different INTERMON tools:

- structure discovery
- monitoring tools
- modelling toolkit.

The IM-DB modelling information, obtained by the modelling toolkit based on the measurements, are used for building INTERMON simulation environment.

Where the structure discovery and modelling toolkits are designed to operate directly with the IM-DB, the monitoring toolkits could use so called adapters in order to transform their measured data into the INTERMON data base.

The IM-DB used for interworking of different tools for inter-domain monitoring, modelling and visualisation includes filtered measurement information from different measurement sources (OPENESS supported).

The source measurement data could be provided in the form of:

- measurement server files (for instance border router traces) or as
- data bases for collection of measured data (ISP-DB) of the INTERMON meters at different collection points : at border router (for border router traffic collection), at end-points (for end-to-end QoS parameter values or events), at points selected for inter-domain performance metric collection
- measurement data base from different other systems (i.e. AQUILA data base []).

Adapters in the INTERMON architecture could be used to transform the measured data into IM-DB presentation. Adapters could be used in order to support “OPEN” INTERMON architecture, i.e. support of various kind of measurement data provided also by other tools and data base. For instance, an adapter could be written to transform AQUILA data base measurement data into INTERMON data base data models.

Adapters as well as meter tools are configured by the INTERMON GUI. Remote Meters (demons or data collection routines) are configured dependent on the measurement scenarios. In addition IM-DB is containing modelling and topology presentation obtained from topology discovery tool , as well as some administrative data (users, permissions, access rights , etc) The IM-DB is created for each user of the INTERMON system (Telecom, end to end service user). The IM-DB obtains its data from the different kinds of sources using adapters.

3 RELATED TOPICS

In this section we propose an overview of topics somehow related to measurement. We describe time synchronization, give an overview on traffic generators, write about sampling techniques and standardization efforts carried out by the IETF working groups.

3.1 TIME SYNCHRONIZATION

The coordination of measurements and some measurement metrics needed to be based on synchronized measurement systems. While the coordination of measurements can also be done by 3-way handshake over a communication channel, at least with one way delay measurements a local synchronization of both sides is needed. There are three most popular ways to synchronize two or more systems:

- Network Time Protocol (NTP)
- Global Positioning System (GPS)
- Radio Clock Systems (e.g. DCF77)

NTP is a time distribution protocol and implements with its software distribution means to discipline the local kernel clock of a computer. These mechanisms to discipline a local clock can be used with GPS or DCF77 signal inputs, even if there is no time distribution needed. On the other hand the synchronization over the network via NTP is much cheaper and easier to install compared with GPS and DCF77 systems with its external receivers and antennas. But even with local GPS synchronization a backup NTP time source can increase the overall time stability.

The normal way to synchronize a measurement system or a computer is to deliver a 1 pps (pulse per second) signal to it, where the first edge represents the exact start time of a second derived from UTC (universal coordinated time). If there is an internal clock in the computer or measurement system, it will use this 1 pps signal to synchronize its internal clock to UTC. If there is no internal reference clock (mostly in pure measurement systems, e.g. mostly hardware based) an additional clock for incrementing the internal counters of the measurement system is needed. Preferably a 10 MHz clock is used here, which is delivered by an external reference clock (e.g. driven by GPS or DCF77 signal or based on atomic clocks).

3.1.1 NTP

NTP as a time distribution protocol is based on a multi-tiered system where each layer is called a „stratum“. Servers at each level peer with each other and provide time services to lower levels. Servers at the top or in stratum 1 are directly connected to atomic clocks or radio based time receivers. By compensating for their distance from the authoritative time

sources (GPS satellite system, radio clock sender) these receivers provide highly accurate time services to stratum 1 servers.

Because clocks can fail, stratum 1 servers peer with other stratum 1 servers. NTP assumes that there is a correct time value and that by using multiple sources, unreasonable values can and should be discarded. This is not simple or weighted averaging. If their clock diverges from the times provided by the peers by more than the reasonable amount, stratum 1 servers will stop using their own clock.

Below stratum 1, NTP servers are supposed to obtain time from servers above them as well as at their own level (stratum). The configuration instructions say that each top level server within a specific domain should be a client to at least two servers at the level directly above it and peer with all the other servers in their own domain at their same level as well as at least one other outside peer on the same level. The client servers receive time from but never provide time to the servers at the next higher stratum. Peers receive time from and provide it to other peers. There should be at least three coordinated top level servers in each domain so each network should communicate with at least 6 outside servers at the next higher (numerically lower) stratum and at least 3 outside peers in the same stratum.

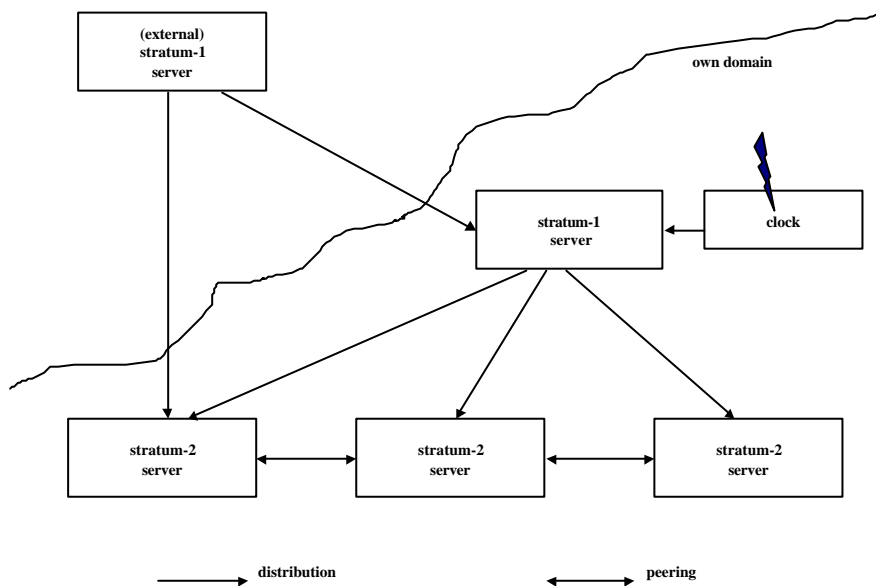


Figure 7: Distributing Time via NTP

When acting as a client and obtaining time from another NTP server, the NTP server measures the total time it takes to get a response. It assumes that the network delay is one half the total and calculates the offset (amount to adjust the local clock) based on this. Because of variable network conditions the actual delay may be different on the two parts of the trip. By talking to several servers, it should get a fairly tight clustering of valid times and discard the times that are clearly off because the returning part was much longer or shorter than the outgoing part or for other reasons.

When the NTP server is first run on a computer, it is very active in talking to the servers from which it obtains its time so it can determine the network delay and a reasonable starting offset. It also starts to calculate the local computer's drift (the amount by which the clock is fast or slow). After the drift is calculated the normal behaviour is to save it in a file (ntp.drift) so that following computer or server restarts it doesn't need to repeat all the work it does the first time it runs.

The first time an NTP server runs, after calculating the offset, it typically makes a single large time adjustment. Since this could result in moving the clock back by a significant amount which might cause problems for some applications, the large adjustment can be prevented with a command line argument. Not using the large adjustment may result in the NTP server taking a very long time to adjust the computer's time to a correct value. After the large adjustment is made or if it's not allowed, the NTP server makes a series of very small clock adjustments.

The NTP server calculates the local computer's drift and uses this to continuously adjust the clock to compensate for the drift. It also adjusts the drift calculation as necessary. As long as the NTP server is running (up to a point), the computer gets an increasingly accurate clock. Since it only slowly adjusts the time based on multiple outside sources, over time each computer running an NTP server should have a clock that gets closer to actual Coordinated Universal Time (UTC) which is NTP's goal. If an NTP server has several sources, it's less likely to be pushed away from the correct time by problems on a single server that it synchronizes with. Once NTP has been running for a while and as long as it remains running, a computer should be able to maintain accurate time for hours or longer if it's cut off from its time sources.

The accuracy of NTP synchronization depends on the end-to-end transmission performance over the network, including the NTP client and server. A accuracy of 1 ms is possible, if the NTP client accesses directly a stratum-1 server.

3.1.2 GPS

The Global Positioning System (GPS) is a satellite-based positioning and navigation system that is funded and operated by the United States Department of Defence. Although originally developed for use by the U.S. military, GPS now supports thousands of civilian users worldwide and is employed in a wide range of applications.

GPS is composed of a space segment, a control segment, and a user segment. The space segment consists of a constellation of 24 satellites along with several spares. Each satellite orbits the Earth once every 12 hours on one of six orbital planes. The GPS constellation is positioned in such a way that any given time, 5 to 8 satellites are visible from any point on Earth. The control segment is a system of monitor stations, ground antennas, and a master control station. The monitor stations measure signals from all visible satellites. The accumulated data is then processed by the master control station to calculate satellite orbits and to update satellite navigation messages, including clock corrections. The revised information is transmitted back to the satellites via the ground antennas, and finally data is transferred over radio signals to GPS receivers. The receivers comprise the user segment, which is the portion of the GPS system that converts signals into timing and positioning information for users. To perform positioning and timing calculations, GPS employs a triangulation technique. With this method, GPS receivers

measure and compare the travel time of radio signals sent from four visible satellites with known positions. Three of the measurements are used to calculate the receiver's position in 3-dimensions, and the fourth is used to determine time. These signals, known as pseudo-random code (PRC), are unique to each satellite. This uniqueness allows all the signals to broadcast over the same frequency.

One caveat to this method of computation is that GPS users often reduce their costs by employing receivers with less accurate clocks. Although this has the potential of introducing error into positioning and timing calculations, GPS avoids this problem by taking an extra satellite range measurement during the triangulation phase. This allows the system to correct any timing offset and thus maintain GPS's overall high level of accuracy. Nevertheless, the triangulation technique requires that each of the satellites transmit its PRC in a highly synchronous manner. A timing error of just 1/1000 of a second would produce a measurement error of nearly 200 miles. Therefore, precise timing is a critical element for the proper implementation of GPS.

GPS keeps its own system time that is derived from a composite clock consisting of all operational satellite clocks and the UTC timing standard. Each GPS satellite contains four atomic clocks (2 caesium and 2 rubidium), offering a very high level of precision. The satellites transmit clock information as part of the signals that are sent to the monitor stations. The master control station then gathers the data to calculate timing errors and to make appropriate clock corrections. When the revised timing signal is uploaded to the satellites, GPS system time can be broadcasted to the receivers during the satellite navigation message.

The main drawback of GPS timing is the antenna position with direct intervisibility to four or more satellites, if no position of the GPS receiver is known and entered in the system.

The accuracy of the GPS receiver itself is down to 100 ns. An overall system, e.g. a PC running FreeBSD and NTP can reach accuracy of 50 us.

3.1.3 DCF77

The German radio system DCF77 can be used all over Central Europe, it is based on time distribution over long wave radio signals at 77,5 kHz. There are two types of time information hidden in the distributed signal:

First of all there is the well-known timing telegram based on 59 second pulses over the minute. The telegram starts after the missing 60th second pulse. Every beginning of a second is marked by lowering the amplitude (AM) of the carrier frequency by 75% for 0.1 s or 0.2 s. The length of this time marks represent a binary encoded one or zero. The information broadcast includes time, date, parities and status bits. Because of the filtering and signal processing of this radio signal to suppress interferences there is a timing offset about 10 ms in the radio receiver. While this can be subtracted for the time calculation there are also fluctuation in the signal which lead to bad trigger conditions for the start of a second. An overall accuracy is limited to 1 ms under good conditions.

The second time distribution with is based on phase noise in the 77,5 kHz carrier signal. This noise is a pseudo-random bit sequence of 512 bit that is transmitted between the 1 second time marks. A radio clock using a wideband receive can correlate this sequence

with local sequence of pseudo-random bits. This will allow to gain time marks with a dispersion of single microseconds. The accuracy is typical 20 us to UTC.

The advantage of DCF is the inexpensive time receiver for a less accurate timing and the easier installation of an antenna, which need no direct intervisibility to the sky.

The main disadvantages of DCF77 have to do with the radio transmission. One problem with radio signals is that atmospheric may interfere with the signal. By principle one has to know for the more precise time synchronization the distance to the sender near Frankfurt. Furthermore since it is limited in its coverage, the DCF77 signal can only be used for measurements within Central Europe.

3.2 TRAFFIC GENERATORS

We propose an overview of the existing, Open Source tools for traffic generation.

3.2.1 TG

TG [38] is one of the public domain tools from SRI. Can generate constant, uniform, exponential on/off UDP or TCP traffic etc. Its time scope is real time. It requires as input a textual description of the traffic to be generated and gives as output text.

3.2.2 NetSpec

NetSpec [39] is a tool designed to provide sophisticated support for experiments testing the function and performance of networks. NetSpec uses a scripting language that allows the user to define multiple traffic flows from/to multiple computers. NetSpec can emulate a couple of different traffic types, has inetd support, allows to build measurement daemons.

3.2.3 Packet Shell

This software represents the "Packet Shell" [40], an extensible Tcl/Tk based software toolset for protocol development and testing. The Packet Shell creates Tcl commands that allow you to create, modify, send, and receive packets on networks. The traffic generated may be IP, IPv6, ICMP, ICMPv6, TCP, Ethernet, TLI. The operations available for each protocol are supplied by a dynamic linked library called a protocol library. These libraries are silently linked in from a special directory when the Packet Shell begins execution.

3.2.4 Rude/Crude

RUDE [41] stands for Real-time UDP Data Emitter and CRUDE for Collector for RUDE. RUDE is a small and flexible program that generates traffic to the network, which can be received and logged on the other side of the network with the CRUDE.

Currently these programs can generate and measure only UDP traffic. The operation and configuration might look similar to the other available traffic generator tool called MGEN but these programs do not share any code. Actually these tools were designed and coded because of the accuracy limitations in the MGEN program. MGEN operates with system timers and e.g. in the Linux kernel on PC-platforms the timer resolution is only 10ms. That is pretty poor, so we decided to do our own traffic generator which hasn't got this limitation.

3.2.5 MGEN

MGEN [42] provides programs for sourcing/sinking real-time multicast/unicast UDP/IP traffic flows with optional support for operation with ISI's "rsvpd". It now also includes support for scripted generation of packet flows with the IP TOS field set. The MGEN tools transmit and receive (and log) time-stamped, sequence numbered packets. Post-test analyses of the log files can be performed to assess network or network component ability to support the given traffic load in terms of packet loss, delay, delay jitter, etc. MGEN has been used to evaluate the capability of networks and devices to properly provide IP Multicast and RSVP support.

3.2.6 UDPgen

UDPgen [43] is a tool for generating UDP traffic. It aims on maximizing the packet throughput especially for Gigabit Ethernet. To maximize the throughput the traffic generator runs completely in the Linux kernel. This allows sending at much higher rates than with a user space program. The toolset also includes a tool that counts UDP packets at the receiver and calculates the packet interarrival times. This tool also runs in kernel space to minimize CPU time needed and therefore be able to count all packets.

3.3 STANDARDIZATION

3.3.1 IETF IPPM working group

The Framework for IP Performance Metrics [84] defines the general framework, for which IP Performance Metrics have been developed and new ones are currently being developed by the IPMM Working Group. It identifies criteria for the development of new metrics and defines a terminology for the discussion and the description of new metrics

Each metric is supposed to be a carefully specify quantity related to the performance and reliability of the Internet. There might not exist a practical way of measuring this metric, as long as the metric is defined clearly and without ambiguity in meaning this is fine. The units used in the definition and measurement must follow the international metric system. The unit for information is bit and any used prefixes follow the metric meaning, for example one kilobit means decimal 1000 bits. All times are expressed in UTC.

For each metric one or more measurement methodologies may be specified, that are capable of testing the desired metric. Some possible technologies to test a metric include the direct measurement, the projection from lower-level measurements, estimation from more aggregated metrics, and estimation for a certain time from measurements at related times. These are only ideas for measurement methodologies a lot more might be available. While defining a metric a measurement approach might also be discussed, this approach will not be formally part of the specification. An important aspect for the measurement methodology is to lead to repeatable results under identical conditions. But in the Internet it is almost impossible to arrange identical conditions, because one can not repeat the "random" traffic of other Internet users. Therefore this very hard statement is eased up a little bit, so that small variations in conditions, are leading to only small changes in the resulting measurements.

The framework distinguishes three different kinds of metrics:

- Singletons, are metrics that are atomic in their nature, even so if a 'bulk throughput capacity' involves measuring a number of packets, it still could be defined as a singleton metric.
- Samples, are metrics that are derived from a given singleton metric by taking a number of distinct instances together
- Statistics, are metrics that are derived from a sample metric by calculating statistics of the singleton metric value, which belongs to the sample.

Many approaches are discussed for taking samples. A common way for sampling is the periodic one, where samples are taken at certain, fixed intervals. Even so this method is very simple to implement, it also has disadvantages: it might not observe periodic behavior of the metric measured, it might be easily identified as test traffic and handled differently, and the periodic behavior can drive the network into a state of synchronization. A better approach for sampling is to use "random additive sampling". Two possible ways of generating this kind of distribution is to either use a Poisson Sampling or Geometric Sampling, which is closely related to Poisson Sampling. For a detailed description of either, please have a look at [86].

Other important issues, discussed in this framework are timing and coping with errors and uncertainties. For the implementation of any of these metrics and methods, it is necessary to quantify the introduced errors. This can not be done through the metric itself, because it might for example include inaccurate timestamps due to a high load of a measurement machine. Derived or composite metrics might propagate errors and therefore need special analysis of measurement uncertainties. The timing and its errors

are in detail discussed in the framework for IPPM. Because this aspect will easily overload this deliverable, only a very rough example for the possible timing errors that are discussed in the IPPM framework is given. For the measurement of one-way-delay highly synchronized, but not necessarily accurate clocks are needed. Accuracy stands for how close a clock is to the true time, while synchronized means that the clocks have the identical time, but not necessarily the true time. The next part is, that the skew of the clocks, or even a lot more important the skew difference between the clocks is small. The skew is the frequency difference between the clock and the true time. For the One-way Delay a time stamp is used at least two places. If the clocks are synchronized well at the start of measurement, but the skew difference is high, then only the first packets show realistic delay values, because after a while the clocks are not synchronized any more. Please refer to the literature for a detailed discussion of this research aspect [59, and others].

The framework expects, that not all metrics can be measured in one piece, but need to be composed of different measurements. A spatial and a temporal composition are defined. Usually a path contains several subpaths. If each of these subpaths is measured independently and the results are then combined it is called spatial composition. If a metric is extrapolated from measurements that are taken at a related time, the framework calls this temporal composition. Both types of composition introduce new errors and need a qualified and quantified analysis of the uncertainties.

The IPPM measurement framework, specified in RFC 2330, only discusses how to define metrics and measurement methodologies. Problems that might arise during measurements are discussed also, as well as suited ways to arrange the results of the measurements. In the following the already by the IPPM working group defined metrics are summarized:

The IPPM Metrics for Measuring Connectivity (RFC 2678)[87]: the singleton metric for measuring connectivity, which is named Type-P-Instantaneous-Unidirectional-Connectivity has as its input a source address Src, a destination address Dst, and a measurement Time T. The result is a boolean value. Using this singleton a Bi-directional-Connectivity metric is defined, where the Unidirectional is used for the forward and backward path and the measurement result is only true if both Unidirectional measurements are true. By adding a duration dT to these singleton metrics new metric for measuring connectivity over a time interval [T, T+dT] are defined. Finally a "generally useful" [87] connectivity metric is defined, which can use different packet type. This is the Type-P1-P2-Interval-Temporal-Connectivity metric, for which also a measurement methodologies is defined. During the test interval N packets are randomly, but uniformly distributed sent to the Dst. Each packet that is received by the Dst triggers in return a reply packet to Src. If at the Src a reply packet is received the test is successful can terminate. The recommend values for the number of packets N is 20, which should be send over an interval of 50 seconds, the waiting time for a reply packet should be 10 seconds. Therefore the test should maximal last 60 seconds.

The One-Way Delay Metric for IPPM (RFC 2679)[88]: defines first a singleton metric. This Type-P-One-way-delay metric has a source address Src, a destination address Dst, and a measurement time T as input parameters, and as a result a delay time. The measured delay is either a real number of seconds or infinite, if the packet is not

received by the Dst. The measurement methodology for this metric is only very generic, because it mainly depends on the used Type-P packet (TCP, UDP, size, ...). But still for every kind it is necessary to synchronize Src and Dst, use only randomized padding data to stop unrealistic compression of data, and to place a timestamp into the packet, which needs to be taken just before the packet leaves the Src. At the Dst another timestamp needs to be taken at the arrival of this packet. The delay is then calculated from subtracting the starting time from the receiving time according to the timestamps. If the packet is not receiving within a reasonable period of time the "measured" delay is set to infinite. The threshold for this reasonable time is a parameter of the methodology.

With this singleton metric a poisson sampling metric is defined. The Type-P-One-way-Delay-Poisson-Stream consists of a Src, a Dst, a starting time T_0 , an end time T_f , and a rate λ . The times and the rate are used to compute a pseudo-random Poisson distribution of measurement times in the desired time interval. The metric has as its result an sequence of pairs. Each pair contains a time T and a delay time dT . Where the time T represents the input parameter T of the singleton metric and the delay is identical to the singleton.

From the sample metric many statistics are defined including the Percentile, a Median, a Minimum, and an Inverse Percentile. These statistics are used to ease up analyzing of network performance. For a detailed description please refer to [88].

The One-Way Packet Loss Metric for IPPM (RFC 2680)[89]: this metric is very similar to the One-Way Packet Delay metric defined in [88]. The main difference is the resulting value this metric has. It is a boolean value, where a '0' indicates a successful packet transmission and a '1' stands for a lost packet. But, one could transfer the delay metric into this loss metric, each delay that is finite is equal to a successful transfer and any infinite delay stands for a lost packet. For the Packet Loss only one statistic is defined: the Packet-Loss-Average, which is the average of all loss results over a certain time number of packets. To sustain a useful average, several hundred packets should be transmitted over a one minute, which is usually not wanted.

The Round-trip Delay Metric for IPPM (RFC 2681)[90]: this document describes the round-trip delay in a very similar way as [88] and [89] describe the one-way delay and packet loss. First a singleton is defined for measuring the Round-trip Delay at a certain time. From this a sample metric is defined, which checks the Round-trip time using a Poisson distributed stream over a certain interval. In the next step statistics are defined, that are build on top of the specified metrics. There are the identical to the One-Way Delay statistics, but of course use the Round-Trip Delay metrics as their underlying metrics.

3.4 SAMPLING

Increasing data rates and growing measurement demands increase the requirements for data collection resources. For measurement scenarios in backbone networks it is often

required to measure whole traffic aggregates instead of single flows. Furthermore some measurement methods require the capturing of packet headers or even parts of the payload. All this can lead to an overwhelming amount of measurement data, resulting in high demands regarding resources for storage, transport and post processing.

In some cases specialized hardware helps to fulfill these demands but on the other hand increases the costs for providing the measurement. Since measurements are mainly a supporting functionality for the service provisioning, measurement costs usually should be limited to a small fraction of the costs of the network service provisioning itself. Therefore a reduction of the measurement result data is crucial to prevent the depletion of the available (i.e. the affordable) resources. Such a reduction can be achieved by a reasonable deployment of sampling techniques. Sampling helps to prevent an exhaustion of resources and to limit the measurement costs. Examples for applications that benefit from sampling are given in [74].

3.4.1 State of Art

Sampling has been applied to measurements for different purposes. In [71] different sampling techniques are introduced and a first classification of methods is described. In [76] it is shown how the packet count on a link can be estimated by neglecting small traffic sources in a controlled way. In [72] sampling techniques are used for traffic characterization. Systematic, stratified and simple random sampling techniques are used to estimate the distribution of packet sizes and inter-arrival times from a packet trace from an entrance interface to the NSFNET national backbone. In [75] sampling is used to find out the path of packet flows in a network. A pattern matching technique is used to select the packets of interest. [73] uses a similar approach based on ATM cell patterns for QoS measurements. Count-based sampling has already been implemented in products like Cisco's NetFlow [78], InMon's sMon [79] and the meter NeTraMet [80].

3.4.2 Deployment of Sampling Techniques for packet selection

The deployment of sampling techniques aims at the provisioning of information about a specific characteristic of the parent population at a lower cost than a full census would demand. In order to plan a suitable sampling strategy it is therefore crucial to determine the needed type of information and the desired degree of accuracy in advance.

First of all it is important to know the type of metric that should be estimated. The metric of interest can range from simple packet counts [76] up to the estimation of whole distributions of flow characteristics [72].

Secondly, the required accuracy of the information and with this, the confidence that is aimed at, should be known in advance. For instance for usage-based accounting the required confidence for the estimation of packet counters can depend on the monetary value that corresponds to the transfer of one packet. That means that a higher confidence could be required for expensive packet flows (e.g. premium IP service) than for cheaper

flows (e.g. best effort). The accuracy requirements for validating a previously agreed quality can also vary extremely with the customer demands. These requirements are usually determined by the service level agreement (SLA).

Sampling is considered as part of the metering process as defined in the IPFIX requirement document [50]. It can be applied at different functions of the metering process (e.g. during packet header capturing, before or after classification, etc.). In the following we consider a measured IP packets with its observation point and timestamp as basis elements of the parent population. And all packets in the flow of interest as the parent population. Please note that with the IPFIX flow definition the flow of interest can also include all packets on the link.

The sampling method and the parameters in use must be clearly communicated to all applications that use the measurement data. Only with this a correct interpretation of the measurement results can be ensured.

3.4.3 Sampling Methods

Sampling Methods can be characterized by the sampling algorithm, the trigger type used for starting a sampling interval and the length of the sampling interval. These parameters are described here in detail.

3.4.3.1 Sampling Algorithm

The sampling algorithm describes the basic process for selection of samples. In accordance to [71] and [72] we define the following basic sampling processes:

Systematic Sampling

Systematic sampling describes the process of selecting the starting points and the duration of the selection intervals according to a deterministic function. This can be for instance the periodic selection of every n th element of a trace but also the selection of all packets that arrive at pre-defined points in time. Even if the selection process does not follow a periodic function (e.g. if the time between the sampling intervals varies over time) we consider this as systematic sampling as long as the selection is deterministic. The use of systematic sampling always involves the risk of biasing the results. If the systematics in the sampling process resembles systematics in the observed stochastic process (occurrence of the characteristic of interest in the network), there is a high probability that the estimation will be biased. In this context it also has to be considered that there might be systematics (e.g. periodic repetition of an event) in the observed process which one might not be aware of in advance.

Random Sampling

Random sampling selects the starting points of the sampling intervals in accordance to a random process. The selection of elements are independent experiments. With this,

unbiased estimations can be achieved. In contrast to systematic sampling, random sampling requires the generation of random numbers. One can differentiate two methods of random sampling:

- **n-out-of-N sampling:** In n-out-of-N sampling n elements are selected out of the parent population that consists of N elements. One example would be to generate random numbers and select all packets which have a packet position equal to one of the random numbers. For this kind of sampling the sample size is fixed.
- **Probabilistic sampling (see also [74]):** In probabilistic sampling the decision whether an element is selected or not is made in accordance to a pre-defined selection probability. An example would be to flip a coin for each packet and select all packets for which the coin showed the head. For this kind of sampling the sample size can vary for different trials. The selection probability is not necessarily the same for each packet and can depend on other parameters (e.g. the packet content) [74].

Stratified Sampling

The basic idea behind stratified sampling is to increase the estimation accuracy by using a-priori information. The a-priori information is used to perform an intelligent grouping of the elements of the parent population. With this a higher estimation accuracy can be achieved with the same sample size.

Stratified sampling divides the sampling process into multiple steps. First, the elements of the parent population are grouped into subsets in accordance to a given characteristic. This grouping can be done in multiple steps. Then samples are taken from each subset.

The stronger the correlation between the characteristic used to divide the parent population and the characteristic of interest (for which an estimate is sought after), the easier is the consecutive sampling process and the higher is the stratification gain. For instance if the dividing characteristic were equal to the investigated characteristic, each element of the sub-group would be a perfect representative of that characteristic. In this case it would be sufficient to take one arbitrary element out of each subgroup to get the actual distribution of the characteristic in the parent population. Therefore stratified sampling can reduce the costs for the sampling process (i.e. the number of samples needed to achieve a given level of confidence).

3.4.3.2 Sampling Frequency and Interval-Length

According to [71] and [72] we differentiate sampling techniques by the event that triggers the sampling process. The trigger determines what kind of event starts and stops the sampling intervals. With this the sampling frequency and the length of the sampling interval (measured in packets or time) is determined. It is also possible to combine start

and stop triggers of different types (e.g. start a 10 s measurement interval every n-th packet). Nevertheless, due to the unknown relation between number of packets and duration of an interval this can lead to unexpected overlapping of sampling intervals. We distinguish the following techniques:

Count-based Trigger

With this method the packet count triggers the start and stop of a sampling interval. One example is the systematic sampling of every n-th packet of a specific type. For count-based sampling it is necessary to integrate a packet counter into the meter. Since non-intrusive measurements are based on the traffic in the network only, the time that it takes until the n packets of a specific type are seen by the probe is unknown. This means the duration of the sampling process is undetermined (and can be infinite) if the sampling goal requires a minimum sampling size (number of packets).

Time-based Trigger

In time-based sampling the arrival time of a packet at the meter determines whether this packet is captured or not. One example is to capture packets every 30 seconds. If the stop trigger is also a point in time the sampling interval length is given as the time duration between this two points. Since it is unknown how many packets arrive in a specific time interval the number of packets captured with this technique is unknown (and can be zero). This has to be taken into account if a minimum sampling size is required.

Packet-content-based Trigger

With this method the content (or parts of the content) of the packet itself (header, payload or both) triggers the sampling process. This can be achieved by direct comparison of parts of the packet with a reference pattern [73] or by matching the result of a function performed on packet content [75].

3.4.4 Sampling Parameters

The decision whether to select a packet or not is based on a function which takes packet properties and sampling parameters as input. The sampling parameters usually remain the same for the sampling process and are pre-defined by the administrator. A special case are sampling parameters that depend on packet properties (e.g. selection probability dependent on packet content). In such cases only the function which describes the dependency is fixed in advance. Packet properties are examined per packet and are only available after the packet has arrived at the meter.

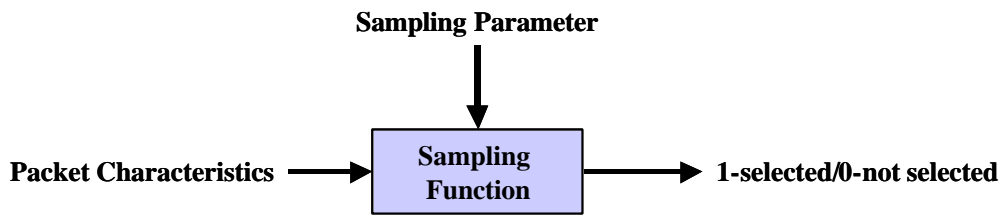


Figure 8: Sampling Function

Selection decision = $f(\text{sampling parameters, packet properties})$

Packet properties are: packet position, arrival time, packet content (header fields, parts of payload) and observation point.

Which packet properties are used as input for the sampling function is determined by the used sampling algorithm. For count based sampling the packet position is used as input. For time-based sampling the arrival time and for content-based sampling (parts of) the packet content (e.g. header fields). It is also possible, that the algorithm differs with regard to the observation point on which the packet was observed. The sampling parameters differ for the different sampling techniques.

3.4.4.1 Parameters for systematic sampling

For systematic sampling the deterministic function which is used for the packet selection needs to be given. For periodic sampling the start of the first selection interval, the length of the selection interval (given in number of packets or as time duration) and the spacing between selection intervals needs to be specified.

←interval length = 7 → ←spacing = 5 →

Packet position: 1 2 3 4 5 6 7 8 9 10 11 12 13..

In sample: 1,2,3,4,5,6,7, 13,...

Selecting every x -th packet would be a special case with selection interval=1 and spacing= $x-1$.

3.4.4.2 Parameters for random sampling

For random n -out-of- N sampling only the sample size n needs to be specified. This can be done either as an absolute number or as fraction of the parent population n/N .

For probabilistic sampling the selection probability p needs to be specified. If the selection probability depends on other parameters (e.g. packet content), the function that expresses this dependency has to be specified.

3.4.4.3 Parameters for stratified sampling

For stratified sampling one has to specify classification rules for grouping the elements into subgroups and the sampling scheme that is used within the subgroups. For the sampling scheme within the subgroups the parameters have to be specified as described above.

CONCLUSIONS

We have summarized and compared research-oriented measurement infrastructures that attempt to measure global Internet behavior and offer public web-accessible reports. In fact, no single organization is truly measuring global Internet behavior, because the global Internet is simply not instrumented to allow such measurement.

A measurement infrastructure can help sites identify abnormal or threatening network activity, facilitate traffic engineering and capacity planning, track long term trends, and enable collection of special-purpose data for experiments. Two basic types of measurements, passive and active, incur different costs and benefits to sites using them. Both types of measurement still require research into improved aggregation and data correlation techniques, as well as methods for coherent data sharing among ISPs and users. We gave an overview on these techniques and described the main projects and available measurement tools making use of active or passive measurement (or of a combination of both).

Identifying areas in which current measurement infrastructures can complement one another, or evolve to use more standard and comparable methodologies, would advance efforts to measure global Internet behavior.

In the second chapter we have outlined how measurement techniques could be combined with other fields (in this case we considered modeling and simulation) to offer new research possibilities. Of great importance are also “complementary” areas, which in this case means tools, research projects or ideas that may be of use to measurement. This is why we dedicated the third chapter describing the state of the art in fields somehow related and/or helpful to it. We are aware the list of topics, tools, and projects we described may be incomplete. We think anyway that it covers the main points in the actual measurement research and hope it may give a good overview of the field and be a good starting point for future research work.

REFERENCES

- [1] V. Paxson, A. Adams, M. Mathis: "Experiences with NIMI", The First Passive and Active Measurement Workshop (PM 2000), Hamilton, New Zealand, April 3-4, 2000.
- [2] S. Kalidindi, M. Zekauskas: "Surveyor: An Infrastructure for Internet Performance Measurements", Proceedings of INET'99, San Jose, CA, USA, June 22-25, 1999
- [3] V. Paxson, J. Mahdavi, A. Adams, M. Mathis: "An Architecture for Large-Scale Internet Measurement", IEEE Communications Vol 36 No 8, p. 48-54, August 1998
- [4] Henk Uijterwaal, Olaf Kolkman: "Internet Delay Measurements using Test Traffic - Design Note", RIPE NCC, Document RIPE-158, May 1997
- [5] Nick Duffield, Matthias Grossglauser: "Trajectory Sampling for Direct Traffic Observation", Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, August 28 - September 1, 2000.
- [6] Ian D. GRAHAM, Stephen F. DONNELLY, Stele MARTIN, Jed MARTENS, John G. CLEARLY: "Nonintrusive and Accurate Measurement of Unidirectional Delay and Delay Variation on the Internet", INET'98, Geneva, Switzerland, 21-24 July, 1998
- [7] G. Almes, S. Kalidindi, and M. Zekauskas, "A One-way Delay Metric for IPPM," work in progress, Dec. 1998.
- [8] G. Almes, S. Kalidindi, and M. Zekauskas, "A One-way Loss Metric for IPPM," work in progress, Dec. 1998.
- [9] <http://www2.auckland.ac.nz/net/Accounting/ntm.Release.note.html>
- [10] <http://www.caida.org/outreach/papers/2001/NTMintro/>
- [11] <http://www.advanced.org/surveyor/>
- [12] V. Jacobson, traceroute, <ftp://ftp.ee.lbl.gov/traceroute.tar.Z>, 1989.
- [13] <http://www.caida.org/tools/measurement/skitter>
- [14] Cisco Internet Performance Monitor: <http://www.cisco.com/warp/public/cc/pd/wr2k/nemo/>
- [15] RIPE Network Coordination Centre, Test Traffic Project Homepage <http://www.ripe.net/test-traffic/index.html>
- [16] <http://sitka.triumf.ca>
- [17] <http://www.caida.org/tools/utilities/others/pathchar>
- [18] <http://www.employees.org/~bmah/Software/pchar/>
- [19] <http://www.netperf.org/netperf/NetperfPage.html>
- [20] <http://www.stanford.edu/~schemers/docs/fping/>
- [21] <ftp://ftp.internatif.org/pub/unix/echoping>
- [22] L.Cottrell, "Pinger Tools," <http://www.slac.stanford.edu/xorg/icfa/ntf/tool.html>, May 1998.

- [23] BRIX 1000 Verifier: <http://www.brixnet.com/products/brix1000.html>
- [24] <http://www.netiq.com/products/chr/default.asp>
- [25] <http://www.caida.org/tools/measurement/coralreef>
- [27] I. D. Graham, S. F. Donnelly, S. Martin, J. Martens, and J. G. Cleary, "Nonintrusive and Accurate Measurement of Unidirectional Delay and Delay Variation on the Internet," Proc. INET '98, Jul. 1998.
- [26] I. D. Graham, et al., Waikato Applied Network Dynamics (WAND) Project homepage, <http://atm.cs.waikato.ac.nz/wand/>, 1998.
- [29] <http://firehunter>
- [28] http://www.cisco.com/warp/public/cc/cisco/mkt/ios/netflow/tech/napps_wp.htm
- [30] <http://www.ncne.nlanr.net/nimi>
- [31] <http://hydra.uits.iu.edu/abilene/traffic>
- [32] <http://nws.npaci.edu/NWS/>
- [33] <http://www.caida.org/tools/measurement/mantra> [34] <http://www.ripe.net/ripence/mem-services/ttm>
- [35] <http://www.darpa.mil/ipto/psum2000/G203-0.html>
- [36] <http://networks.ecse.rpi.edu/~olsim>
- [37] <http://www.ist-intermon.org>
- [38] <http://www.caip.rutgers.edu/~arni/linux/tg1.html>
- [39] <http://www.itc.ku.edu/netspec/>
- [40] <http://playground.sun.com/psh/>
- [41] <http://www.atm.tut.fi/rude>
- [42] <http://manimac.itd.nrl.navy.mil/MGEN/>
- [43] <http://www.fokus.fhg.de/usr/sebastian.zander/private/udpgen>
- [44] http://www.ripe.net/ripence/mem-services/ttm/Talks/0201_RIPE41_HU/sld008.html
- [45] <http://www.ripe.net/ttm/General/ttmfaq.html#10>
- [46] RIPE 158, Design document (June 1997), <http://www.ripe.net/ttm/Documents/RIPE/RIPE158/note.html>
- [47] http://www.ripe.net/ripence/mem-services/ttm/Talks/0201_RIPE41_MS/index.html
- [48] PAM Meeting Hamilton, NZ, April 3-4, 2000, <http://www.slac.stanford.edu/grp/scs/trip/pam-mar00.html>
- [49] Comparing Two Implementations of the Delay and Loss Metrics, Kalidindi, Uijterwaal, Wilhelm, Zekaukas, IPPM March 18, 1999
- [50] <http://www.ripe.net/ripence/mem-services/ttm/Documents/RIPE/RIPE158/node2.html>
- [51] Passive Calibration of an Active Measurement System, Donnelly, Graham, Wilhelm, PAM2001 Amsterdam, April, 23-24th, 2001
- [52] http://www.ripe.net/ripence/mem-services/ttm/Talks/0105_RIPE39_RW/sld021.html
- [53] http://www.ripe.net/ripence/mem-services/ttm/Talks/0105_RIPE39_RW/sld004.html

- [54] <http://www.ripe.net/test-traffic/General/dell.html>
- [55] <http://www.trimble.com/>
- [56] <http://www.trimble.com/products/catalog/timing/acutime2000.htm>
- [57] <http://www.eecis.udel.edu/~mills/ntp.htm>
- [58] <http://www.ripe.net/ttm/General/alarm.html>
- [59] http://www.ripe.net/test-traffic/Talks/0201_RIPE41_HU/sld020.html
- [60] http://www.ripe.net/test-traffic/Talks/0201_RIPE41_HU/sld021.html
- [61] http://www.ripe.net/test-traffic/Talks/0110_RIPE40_HU/sld019.html
- [62] <http://tt01.ripe.net:10259/>
- [63] <http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/index.html>
- [64] http://tt01.ripe.net:10259/cgi-bin/inbound_delays.cgi
- [65] <http://tt01.ripe.net:10259/cgi-bin/box1.cgi>
- [66] <http://tt01.ripe.net:10259/cgi-bin/gps64s-std.cgi>
- [67] <http://root.cern.ch/>
- [68] http://www.ripe.net/ttm/General/tt_examples.html
- [69] <http://www.ripe.net/ttm/General/trends.example.gif>
- [70] http://www.ripe.net/ttm/General/tt_routing.html
- [71] Paul D. Amer, Lillian N. Cassel: Management of Sampled Real-Time Network Measurements, 14th Conference on Local Computer Networks, October 1989, Minneapolis, pages 62-68, IEEE, 1989
- [72] K.C. Claffy, George C Polyzos, Hans-Werner Braun: Application of Sampling Methodologies to Network Traffic Characterization, Proceedings of ACM SIGCOMM'93, San Francisco, CA, USA, September 13 - 17, 1993
- [73] I. Cozzani, S. Giordano: Traffic Sampling Methods for end-to-end QoS Evaluation in Large Heterogeneous Networks. Computer Networks and ISDN Systems, 30 (16-18), September 1998.
- [74] Nick Duffield, Albert Greenberg, Matthias Grossglauser, Jennifer Rexford: A Framework for Passive Packet Measurement, Internet Draft draft-duffield-framework-papame-01, work in progress, February 2002
- [75] Nick Duffield, Matthias Grossglauser: Trajectory Sampling for Direct Traffic Observation, Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, August 28 - September 1, 2000.
- [76] Jonathan Jedwab, Peter Phaal, Bob Pinna: Traffic Estimation for the Largest Sources on a Network, Using Packet Sampling with Limited Storage, HP technical report, Management, Mathematics and Security Department, HP Laboratories, Bristol, March 1992, <http://www.hpl.hp.com/techreports/92/HPL-92-35.html>
- [77] J. Quittek, T. Zseby, B. Claise, S. Zander, G. Carle, K.C. Norseth: Requirements for IP Flow Information Export, Internet Draft <draft-ietf-ipfix-reqs-05.txt>, work in progress, August 2002
- [78] NetFlow Services and Applications, White Paper, Cisco Systems, 1999
- [79] <http://www.inmon.com/faq.htm>

- [80] N. Brownlee: Traffic Flow Measurement: Experiences with NeTraMet, Request for Comments 2123, March 1997
- [81] K.C. Norseth, Paul Calato: IPFIX Data Model, Internet Draft draft-ietf-ipfix-data-00.txt, work in progress, February 2002
- [82] K.C. Norseth, Ganesh Sadasivan Architecture Model for IP Flow Information Export Internet Draft draft-ietf-ipfix-architecture-02.txt , work in progress, June 2002
- [83] Tanja Zseby, Reinaldo Penno, Nevil Brownlee: IPFIX Applicability, Internet Draft draft-zseby-ipfix-applicability-00.txt, work in progress, June 2002
- [84] V. Paxson, G. Almes, J. Mahdavi, M. Mathis: "Framework for IP Performance Metrics", RFC 2330, June 1998
- [85] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss: "An Architecture for Differentiated Services" Internet Engineering Task Force, RFC 2475, December 1998
- [86] J. Mahdavi, V. Paxson: "IPPM Metrics for Measuring Connectivity", RFC 2678, September 1999
- [88] G. Almes, S. Kalindi, M. Zekauskas: „A One-way Delay Metric for IPPM“, RFC 2679, September 1999
- [89] G. Almes, S. Kalidindi, M. Zekauskas: "A One-way Packet Loss Metric for IPPM", RFC 2680, September 1999
- [90] N. Brownlee, C. Mills, G. Ruth: "Traffic Flow Measurement: Architecture", Informational RFC 2722, October 1999

APPENDIX A MEASUREMENT TOOLS COMPARISON

Tool	Active / Passive	Analysis Type	Monitors	Countries
Coral Reef	Passive	Workload	3	1
IEPM	Active	Performance	38	14
I2 (Abilene)	Both (SNMP)	Workload, performance	12	14
Mantra	MBGP Routing	Multicast performance	17 routers	world-wide
MAWI (WIDE)	Passive (SNMP)	Workload, performance	4	1
NIMI	Active	Performance	35	6
NPACINWS	Both	Performance	40	1
PPNCG	Active	Performance	2	3
RIPE-RIS	Passive (SNMP) Routing	Topology, Routing	4 collectors 45 peers	world-wide
Skitter	Active	Topology, Performance	22	world-wide
Surveyor	Active	Topology Performance	51	9
U-Oregon Route Views	Passive (SNMP) Routing	Topology, Routing	40 peers	world-wide
TRIUMF	Active	Topology Performance	1	N/A