ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Dominique Giger

Passive Measurement of Network Quality

Student Thesis MA-2006-03
6th May 2005 / Winter Term 2005/06

Tutors: Roel Vandewall, Stefan Lampart (Open Systems AG)
ETH-Tutors: Arno Wagner, Placi Flury
Supervisor: Prof. Dr. Bernhard Plattner

Abstract

The importance and necessity of end-to-end fault and performance problem detection in wide area production networks increases with the complexity of the paths, the diversity of network components performance, and dependency on the network. Network monitoring provides insight into the state and performance of the network and whether it is behaving as expected and whether changes in the network have improved or degraded its performance. This is essential for a service provider like Open Systems, because they have to respect Service Level Agreements (SLAs) and want to optimize their systems, improve Quality of Service (QoS) as well as plan for the future. Open Systems is a leading provider of network security world-wide. To measure its network quality, it generates artificial traffic by active ICMP probing. But active monitoring doesn't consider actual customer traffic and places additional load on the network. Thus, the goal of this Thesis is to implement a passive monitoring system which uses existing customer traffic to determine the network performance.

This Thesis starts with a broad survey of known passive monitoring mechanisms. The analysis and classification points out the limits of passive measuring. Hence a short overview of existing active probing systems is presented. The research shows that only one of the proposed methods is well suited for the VPN networks. To develop new approaches of passive monitoring suitable for VPNs, the target VPN environment is analyzed in detail.

The specification and implementation of a passive monitoring mechanism was based on these investigations. The prototype monitors and collects two different network metrics from VPN hosts around the globe: Round Trip Time (RTT) and Packet Loss. To identify network performance problems or anomalous variations in the traffic as well as to evaluate the designed monitoring techniques, many series of plots of the network metrics were done. The encouraging evaluation showed that passive Packet Loss detection and RTT form a sound basis for detecting network performance problems in VPNs.

Abstract

Das Internet erfuhr mit der Globalisierung in den letzen Jahren ein starkes Wachstum. Damit wuchs nicht nur die Vielfalt der Netzwerkkomponenten sondern auch die Komplexität der Netzwerktopologien. Viele Unternehmen sind abhängig von der Erreichbarkeit von IT-Systemen, dem Zustand von Diensten und der Qualität des Netzwerks. Das Netzwerk Monitoring bietet einen Einblick in den Zustand und in die Leistung des Netzes und evaluiert, ob sich die Qualität verbessert oder vermindert hat. Für Service Provider wie Open Systems ist es deshalb wesentlich, permanentes Monitoring zu betreiben und modernste Netzwerk- und Servertechnologien einzusetzen um stets über alle Leitungs- und Systemzustände genauestens informiert zu sein und somit eine hohe Qualität zu ermöglichen. Open Systems ist ein weltweit tätiger Anbieter für Netzwerk-Sicherheit und sichert Geschäftsprozesse und sensitive Daten ihrer Kunden. Das Unternehmen überwacht ihre VPN Gateways mit aktivem ICMP Probing. Diese Proben verursachen aber zusätzlichen Verkehr in den Kundenleitungen, was vermieden werden sollte. Das Ziel dieser Masterarbeit ist es deshalb einen passiven Monitoring Mechanismus zu entwickeln, der Auskunft über die Netzwerkqualität gewährleistet.

Die Arbeit beginnt mit einer umfassenden Studie und Analyse über bereits existierende passive Ansätze. Die Evaluation ergab, dass sich nur ein Ansatz für die VPN Umgebung von Open Systems weiterentwickeln lässt. Er basiert auf dem TCP Protokoll und berechnet so die Round Trip Time (RTT). Die Studie machte zudem auch die Grenzen von passivem Monitoring ersichtlich, weshalb zusätzlich ein kurzer Überblick über aktive Monitoring Techniken erfolgte. Ein weiterer Ansatz für den passiven Monitoring Mechanismus wurde schliesslich bei der detaillierten Analyse von der VPN Umgebung bei Open Systems gefunden. Dank der IPSec Architektur können die Anzahl verworfener Pakete gezählt und als Paket Verlust (Packet Loss) mitgeteilt werden.

Der Prototyp des neuen Mechanismus überwacht und misst den existierenden Kundenverkehr, der einen VPN Gateway durchquert. Anhand dieser Datenpakete berechnet er die RTT und den Packet Loss. Der Prototyp wurde anschliessend unter verschiedensten Gesichtspunkten evaluiert. Plots unterstrichen die gewonnenen Ergebnisse visuell. Das erfreuliche Resultat der Evaluation war, dass der Mechanismus die beiden Metriken zuverlässig messen kann und somit die Netzwerkqualität widerspiegelt.

Contents

List of Figures

List of Tables

List of Abbreviations

ABS	Accounting and Billing System
ABW	Available Bandwidth
ACK	Acknowledgment flag
AES	Advanced Encryption Standard
AH	Authentication Header
AIEPM	Active Internet End-to-End Performance Measurements
AIMD	Additive Increase, Multiplicative Decrease
Argus	Audit Record Generation and Utilization System
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
BTC	Bulk Transfer Capacity
CAIDA	Cooperative Association for Internet Data Analysis
CERNET	China Education and Research Network
CPU	Central Processing Unit
CSG	Communication Systems Group
DC	Data Collector
DNS	Domain Name System
ESP	Encapsulating Security Payload
FAR	Flow Activity Record
FDDI	Fiber-Distributed Data Interface
FTP	File Transfer Protocol
GC	Garbage Collector
GPS	Global Positioning System
GRE	Generic Routing Encapsulation
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IAT	Inter-Arrival Times
IATU	Inter-Arrival Times Unit
IC	Interrupt Coalescing
ICMP	Internet Control Message Protocol
ICV	Integrity Check Value
ID	Identification
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IKE	Internet Key Exchange
IMI	Internet Measurement Infrastructure
IP	Internet Protocol
IPSec	IP Security
ISAKMP	Internet Security Association and Key Management Protocol
ISDN	Integrated Services Digital Network
ISP	Internet Service Provider
KLIPS	Kernel IPSec
LAN	Local Area Network
MAN	Metropolitan Area Network
MIB	Management Information Base

MLM	Mid-Level Manager
MMS	Maximum Segment Size
MP	Measurement Points
MPLS	Multiprotocol Label Switching
MTU	Maximum Transmission Unit
NAI	Network Analysis Infrastructure
NLANR	National Laboratory for Applied Network Research
NMS	Network Management Systems
NTP	Network Time Protocol
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OWD	One-Way Delay
PGM	Probe Gap Model
PPTD	Packet Pair / Train Dispersion
PP	Packet Pair
PPP	Point-to-Point Protocol
PPS	Pulse Per Second
PRE	Passive RTT Estimate
PRM	Probe Rate Model
PTSL	Protocol Trace Specification Language
QoS	Quality of Service
RFC	Request for Comments
RIP	Routing Information Protocol
RMON	Remote network Monitoring
RTFM	Realtime Traffic Flow Measurement
RTT	Round Trip Time
SA	Security Association
SLA	Service Level Agreement
SLoPS	Self-Loading Periodic Streams
SNMP	Simple Network Management Protocol
SPI	Security Parameters Index
SYN	Synchronize Sequence Numbers flag
TCP	Transmission Control Protocol
TD	Train Dispersion
TIK	Computer Engineering and Networks Laboratory
TLV	Type-Length-Value
TOPP	Trains of Packet Pairs
TTL	Time To Live
UDP	User Datagram Protocol
VPN	Virtual Private Network
VPS	Varriable Packet Size
WWW	World Wide Web

Chapter 1

Introduction

Section 1.1 explains the problem statement, Section 1.2 the task description, Section 1.3 the context and Section 1.5 the chosen approach. Further, the environment is elaborated on in Section 1.4.

1.1 Problem Statement

1.1.1 Active Probing

Measurements of the various characteristics of a network provide insight into the state and performance of the network. They are collected by either active or passive monitoring. Understanding the composition and dynamics of the Internet traffic is of great importance for network management to identify and track problems.
Furthermore, business operations require monitoring to ensure appropriate Quality of Service (QoS). A service provider, for example, would like to identify realistic settings of Service Level Agreements (SLAs) and monitor current level of activity to verify they are being met.
Open Systems AG is a leading provider of network security in Switzerland. It runs large international Virtual Private Networks (VPN) for large and midsize companies, which allows different sites to securely communicate with each other. To measure the quality and to ease the management of these large environments, Open Systems monitors the VPN gateways by active ICMP probing.

1.1.2 Drawbacks of Monitoring by Active Probing

Open Systems does active monitoring by sending ICMP echo requests to adjacent gateways and measuring the Round Trip Times (RTT). This creates unwanted extra traffic, which can slow down the VPN connection and fill up the customer's Internet Service Provider (ISP) link. The direct consequence of which is that the customer has to provide some extra bandwidth and therefore has to pay more for the VPN Service.
Another drawback of active probing is that the traffic is artificial, i.e. the volume and other parameters are fully adjustable. Open Systems, in particular, injects only ICMP probes of small traffic volume. But experience shows that the quality characteristic of the Internet often vary depending on factors like packet size, bandwidth used, time of day and protocols like TCP or UDP. A point of principle therefore is if this technique obtains meaningful measurements.

1.2 Task Description

The goal of this Thesis is to build a monitoring system that passively measures the network quality by analyzing existing customer traffic. In this way the stated drawbacks of 1.1.2 are solved. Monitoring real traffic offers additional protocol header information and therefore enlarge the indications to asses the network quality. Performance parameters of additional protocols allows to identify key characteristics such as

- Packet Loss
- Maximum Packet Size

- Layer 1 and Layer 2 Errors
- Number of retransmissions
- Latency (shortest RTT)

But passive approaches are dependent on the existence of customer traffic and limited in their ability isolating the exact fault location. Since the passive approach may require monitoring all packets on the network, there can be privacy or security issues about how to access and protect the data gathered. Therefore passive monitoring should be regarded as complementary to active probing.

1.3 Context

The practical part of this Master Thesis has been conducted in a commercial environment. It is a cooperation of the Computer Engineering and Network Laboratory (TIK) of the ETH and Open Systems AG based in Zurich, Switzerland.

1.3.1 TIK

The Computer Engineering and Networks Laboratory (TIK) was founded in 1989 as a part of the Information Technology and Electrical Engineering Department at ETH Zürich. It focuses on research in embedded computer systems, communications networks and distributed systems. The TIK consists of four groups; each of them specialized in another topic. This Thesis is announced by the Communication Systems Group (CSG) of Prof. Bernhard Plattner which performs research in communication networks, and network security.

1.3.2 Open Systems AG

The Thesis has been conducted in collaboration with Open Systems AG. Open Systems AG is a company specialized in Network Security since 14 years. As part of its Mission Control Services, it provides its customers a VPN environment that allows them a secure communication between different sites. Currently Open Systems manages over 600 security systems in 70 countries.
Since 1999, Open Systems cooporates with the Computer Engineering and Network Laboratory (TIK) of the ETH Zurich and proposes tasks for Diploma Thesis.

1.4 Implementation Environment

The task of this Thesis is to design a passive monitoring mechanism which Open Systems AG plans to install on its VPN gateways. Figure 1.1 on page 9 shows an extract of a typical company topology.

Mission Control The Mission Control Center is located at the Open Systems AG in Zurich. It is responsible for maintaining, monitoring and updating all security systems around the clock.
Every change in a customer's infrastructure is monitored and analyzed by security engineers. They react to system outages or break-in attempts according to specifically defined customizable escalation processes. Depending on time or type of incident the customers then are notified by different ways.

VPN Site VPN sites are customer offices which are located anywhere in the world. Each of these offices consists of about 10 to 200 hosts which are usually all in the same subnet.
Section 3.1 gives a more detailed explanation about the specific characteristics of the VPN environment.

1.5 Approach

The Master Thesis was split into four major subtasks. They have been solved in the following sequence:

1.5.1 Survey of Existing Monitoring Mechanisms

The Thesis started with a comprehensive paper study to analyze, summarize and evaluate known Passive Monitoring Mechanisms. They are documented in Chapter 2.

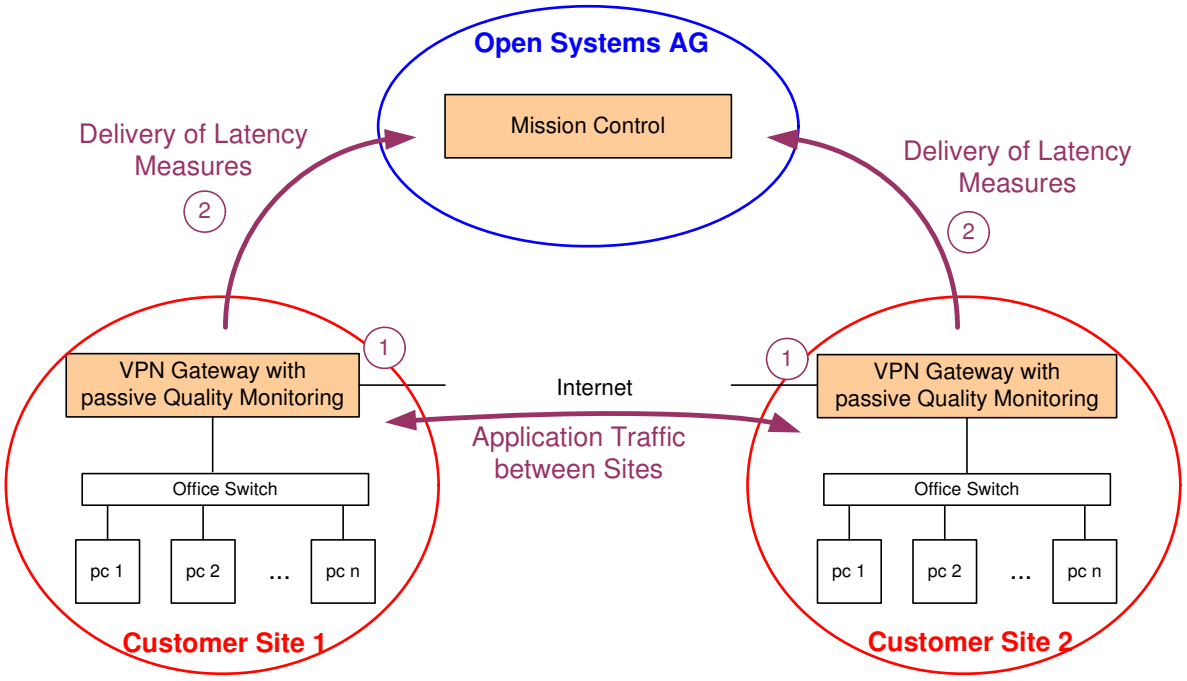

Figure 1.1: Extract of a customer's VPN environment

1.5.2 Specification of a Passive Monitoring Mechanism

The extensive survey highlighted that most existing passive monitoring mechanisms are either too general for the target's environment or they are focused on only one specific traffic type. Therefore, a short investigation of known active probing techniques has been done in Subsection 2.5. Based on the overall analysis a prototype was specified.

1.5.3 Implementation of a Passive Monitoring Mechanism

The specification has been implemented in a prototype consisting of two algorithms. The prototype runs on a Linux based VPN gateway computer that connects an internal LAN with an external network. The prototype monitors traffic passing through the VPN gateway. While the first algorithm reports a minimum and an average RTT the second one counts lost packets. The designs and implementations are described in Chapter 3.

1.5.4 Evaluation

The two algorithms have been installed on the VPN gateways. They collected traffic information and calculated performance metrics. These data files were then evaluated in order to validate that the designed monitoring system is useable.

1.6 Related Work

There are a lot of ongoing projects doing network performance researches. The three most popular ones are [35], [21] and [5]. They do research in both active and passive monitoring and their current status can be visited on their websites.
Approaches concentrating on passive monitoring have been proposed and documented too.
[15] uses Bulk Transfer Capacity metric to measure the available bandwidth. [41], [30], [14] and [32] measures the Round Trip Time (RTT) by observing TCP protocol information. A more accurate metric is calculated by [29] and [4]. They measure the One-Way Delay (OWD) which considers the direction to

measure the time gap.
There are many passive monitoring mechanisms which oberserve the network performance on the basis of flows. NeTraMet [19] offers advanced programming languages for analyzing network flows and building statistical event records. Such as [1] it implements the Realtime Traffic Flow Measurement (RTFM) architecture which is the proposed remote network monitoring system of IETF. Trace, documented in [12] and [13], extends its monitoring infrastructure to a decentralized management platform by using SNMP. Finally, NetFlow [2], first implemented in Cisco routers, is the most widely used measurement solution today. All these mechanisms are summarized in Chapter 2.

Chapter 2

Known Passive Monitoring Mechanisms

There exist several proposals to monitor the network quality in a passive way. These monitoring mechanisms use different metrics to measure the traffic in the network. The metrics are calculated by observing data packets of appropriate protocols of the Internet Protocol Suite.
[6] describes the *Internet Protocol Suite* as a set of communications protocols that implements the protocol stack on which the Internet runs. The protocols can be fitted to a set of layers. Each layer is responsible for a set of problems such as the transmission of data. Table 2.1 depicts the classification of a collection of the most popular and important protocols.

Application	HTTP, FTP, DNS, BGP[1], RIP[1], Telnet
Transport	TCP, UDP , OSPF[2]
Network	IP, IPSec, ICMP[3], IGMP, OSPF[2], BGP[1], RIP[1], ARP
Link	Ethernet, MPLS, PPP, Token ring, FDDI, ATM, ISDN

Table 2.1: Protocol Classification

The *Link layer* is the method to pass packets from the Internet layer of one device to the Internet layer of another and therefore not really part of the Internet protocol suite. Its protocol header fields don't deliver useful information for network quality measurements.

IP performs the basic task of getting packets of data from source to destination and its header information is always monitored for *Network layer* analysis. It can carry data for a number of different higher level protocols. ICMP is used to transmit diagnostic information about IP transmission and IGMP to manage multicast data. They are layered on top of IP but perform network layer functions. All routing protocols, such as BGP, OSPF, and RIP, might seem to belong to higher layer protocols but are actually part of the network layer. These protocols are used for debugging and managing issues of the network layer. Their occurrence is sporadic and therefore not practical for passive monitoring mechanisms, which rely on constant traffic.
IPSec provides security at the network layer and is a standard to make IP communications secure by encrypting and/or authenticating all IP packets.

The *Transport layer* protocols provide the functional and procedural means of data transfers. Its most popular protocols are UDP and TCP.
UDP is a connectionless datagram protocol. It acts as best effort or unreliable protocol. TCP is a reliable, connection-oriented transport mechanism which makes sure data arrives complete, undamaged, and in order. TCP tries to continuously measure how loaded the network is and adapts its sending rate in order to avoid congestion. Furthermore, TCP attempts to deliver all data correctly in the specified sequence. TCP's behavior leads to more comprehensive and specific header information than UDP's. That's why

[1] Run over TCP and UDP respectively, but may also be considered as part of the network layer
[2] Run over IP, but may also be considered as part of the network layer
[3] Run over IP, but may still be considered as part of the network layer

generally only TCP traffic is used for passive network measurements.

The evaluation of the proposed monitoring mechanisms is done according to the characterization of the environment introduced in Section 1.4. The most important criteria are:

Single data capturing point: Each monitoring device possesses only self-collected information. They shouldn't generate additional traffic to exchange their captured data.

Unknown topology: The VPN gateways don't know their neighbors. They can't rely on steady behavior of the network. Routings of data packets can change all the time.

Unknown traffic type: The traffic of each customer network looks different and traffic behavior can change according to day time or season. Each customer uses other kinds of application. Therefore there are no predictions about the number of connections, number of connection setups, duration of connections, data packet sizes and type of application traffic.

Protocols: Monitoring can be done at different interfaces. The set of analyzable protocols differs from interface to interface. The analysis of the environment, described in Section 3.1.1, shows that the following three protocols are prevalent: TCP, IP, and IPSec's ESP. It's known by experience that a large amount of the traffic uses TCP as transport layer protocol. IP is the standard inter-network protocol and therefore available for all traffic. Because of the fact that a VPN environment is targeted, the existence of an ESP header for tunneling is certain.

Sections 2.1 - 2.3 gives a survey about existing passive monitoring mechanisms. They are structured according to the protocol used for data collection. While Section 2.1 discusses monitoring mechanisms related to the transport layer, Sections 2.2 and 2.3 consider protocols of the network layer. The survey of these passive approaches is summarized and classified in Section 2.4. This Section further highlights their limits. Hence active probing techniques conclude in a final subsection this survey.
The interest in known monitoring mechanims capturing IPSec prototcol information is very high since IPSec is a very important factor in the target environment. But no documents were found. Thereupon a design for detecting packet loss was elaborated. Subsection 3.3.2 proposes this self-designed algorithm.

2.1 TCP specific monitoring mechanisms

The monitor mechanisms discussed in Subsections 2.1.1 - 2.1.5 rely on the characteristics of TCP to measure the network quality. This Subsection starts with a short explanation of the TCP protocol to ease the comprehension of the succeeding survey.

TCP is a connection-oriented and reliable transport layer communication protocol. Applications use TCP for stream delivery through the network. TCP divides the byte stream into appropriately sized segments and passes the resulting packets to the Internet Protocol, for delivery through an internet to the TCP module of the entity at the other end.
The typical way a pair of end hosts initiate a connection is a so-called three-way handshake: The client-side opens a connection by sending an initial SYN segment to the server. The server-side then responds with a SYN/ACK. Finally, the client-side responds to the server with an ACK, completing the three-way handshake and connection establishment phase.
Figure 2.1 shows the TCP protocol header.

Window Size TCP uses this field to provide congestion awareness. The window size is limited by the data link layer of the network the computer is attached to and is calculated in terms of *Maximum Segment Size (MMS)*.

Maximum Segment Size *MMS* is defined as the maximum number of data bytes any particular layer-2 technology allows per packet.

Maximum Transmission Unit The *Maximum Transmission Unit (MTU)* is obtained by attaching the TCP/IP headers to the layer-2 packet.

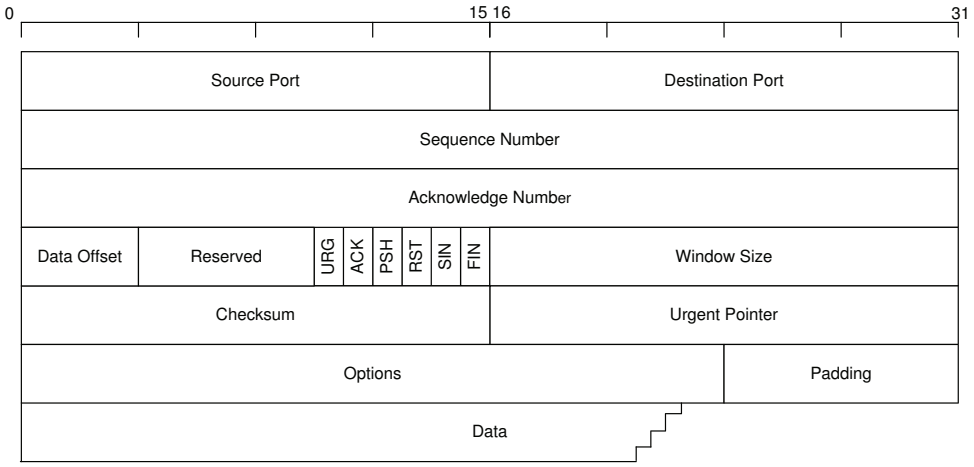

Figure 2.1: TCP protocol header

Sliding Window Protocol TCP typically performs sliding window congestion control by sending an amount of packets round by round. The packets of one round are sent all together and the sender has to wait for the receiver's acknowledgements. The amount of packets sent by round is determined in the header field Window Size. The value of this header field varies during the connection lifetime. To adjust this value, TCP uses an *Additive Increase, Multiplicative Decrease (AIMD)* algorithm.

Additive Increase, Multiplicative Decrease The AIMD algorithm increases the congestion window according to its current mode. In slow start mode, the congestion window is increased by one MMS per received ack, and therefore is doubled once every *Round Trip Time (RTT)*. In congestion avoidance mode the increase of the congestion window is slowed down to one MMS per RTT.

Round Trip Time RTT is defined as the time interval that starts when the TCP sender sends a packet and ends when the sender receives the ACK of that packet.

2.1.1 Passive RTT Estimate Algorithm

The Passive RTT Estimate (PRE) [41] algorithm makes use of the behavior of long-standing TCP connections. A consequence of TCP's sliding window congestion control is that there is a gap between two successive rounds. PRE calculates the RTT by detecting these gaps. There is no restriction on the network topology to use PRE, i.e. the monitor device that implements the algorithm can be situated anywhere in the network.

Algorithm Before the execution of PRE, several variables are initialized according to Table 2.2.

AI	arriving time interval between two packets	Record the arriving time of the first 3 successive packets and calculate two arriving interval T_0 and T_1. AI_0 = min (T_0, T_1)
RTT_E^0	estimated RRT	RTT_E^0 = 32 * A_0
T_{th}^0	threshold value for AI	T_{th}^0 = 2 * (A_0 + 4 * V_0) = 4 * AI_0 (simplified)
V_0	Standard Deviation of AI	V_0 = A_0 / 4

Table 2.2: Initialization

The algorithm for estimating the RTT is an iterative process. Each round i roughly consists of four steps:

1. The device monitors and timestamps the next packet of the targeted TCP flow. It then calculates the new arrival interval AI_i.

2. It compares AI_i with the threshold value T_{th}^i to decide if AI_i indicates a gap.
3. In case that AI_i is an arriving time interval between two packets in the same round (PAI), it updates the mean m and the standard deviation V_i of PAI.
4. If AI_i is a gap, the end of a sending round is found and a new value for the RTT can be calculated.

Advantages

- Simple
- Monitor device can be deployed anywhere in the network

Drawbacks

- Only for long-standing TCP connections

Evaluation According to [41], practical experiments proof the effectiveness and robustness of PRE. Its RTT estimations agreed with the RTT calculations measured by IPPM [1].
The gap detection of PRE seems to conform to the flight behavior described in proposal 2.1.2. This behavior relies on long-term TCP connections. According to the defined target environment's criteria such connections are not guaranteed and therefore PRE's algorithm isn't applicable.

2.1.2 RTT Distribution of TCP Flows

The document [30] presents three different methods of estimating the RTT of TCP flows. It defines a *flow* as a transfer of packets between two ports of a source-destination pair using a particular protocol, and thus a flow is identified by a the 5-tuple consisting of *source IP address, destination IP address, source PORT, destination PORT and the protocol used*. A TCP connection is bidirectional and consists of two flows, one in each direction.
Different monitors, or links, observe the traffic and collect the traces. When a packet has traversed the link, the monitor records its header, gives a timestamp and transfers the data into the PC memory.

For calculating the RTT estimations, flows are divided in five major categories:

TCP-Download Flows The link observes a SYN-ACK packet followed by data packets.

TCP-Feedback Flows The link observes a syn packet followed by a sequence of ack packets.

TCP-Unknown Flows The link never observed a leading SYN or SYN-ACK packet and therefore the TCP flow doesn't fit into the above types.

UDP Flows These flows are often associated with peer-to-peer searches, DNS lookups and real-time streaming.

Other Flows The flow is using another protocol than UDP or TCP.

TCP's behavior can be considered in two different ways.

Fluid View The Fluid View describes TCP as a window-based protocol with two modes of operation that uses the AIMD algorithm. This observation allows calculating the rate in congestion avoidance mode. The rate is defined as the average number of bits transferred over some time interval.

Packet View The Packet View considers the steady state of TCP and tries to recognize a flight behavior. A flight is defined as a sequence of consecutive packets with nearly identical inter-arrival times (IAT) followed by a larger IAT. A TCP flow showing flight behavior would consist of a flight of packets followed by a gap, then another flight of packets and so on. The value of an IAT is given by means of a threshold. The paper differs between IAT, which is the size of the flight, and an IAT unit (IATU). A flight of 1 IATU means that the observed IAT was different from the preceding and following IATs. An IATU value of 2 states that two successive IATs were identical. Other names for the flight behavior are bursts or rounds.

The following methods pursuit different approaches dependent on the view of TCP's characteristics.

[1] The IETF's IPPM workgroup developed this framework for IP Performance Metrics. See RFC 2330 for further information.

SYN based method This method tries to find the time between the syn-ack and the data packet of download flows or, in case of a feedback flow, the time between the sending of syn and ack.

Flight method The method defines the time between the leading edge of a flight and the leading edge of the next flight as the RTT. The flight-finding algorithm consists of three steps:

1. start with IAT = 0 and flightsize = 1
2. compare previous IAT with current IAT
3. if both IATs are within threshold the increment flightsize by one, else if flightsize >1, start a new flight of size 0 else start a new flight of size 1

Rate change method This method proposes the RTT during the congestion avoidance phase in an interval [t0, t1] as:

$$\text{RTT} = \sqrt{\frac{MTU}{\frac{d^2x}{dt^2}}}$$

Assumptions: During this interval [t0, t1], there are no packet drops and the RTT doesn't change.

To ensure that the flow has entered the congestion avoidance phase, the method arbitrarily sets a threshold of 15 packets.

Advantages

- Multiple methods to ensure that the estimated RTT is correct
- Only single point of data capturing required

Drawbacks

- Only for TCP connections

Evaluation Tests done with the implementations of these three methods have shown that no significant differences occurred in the results. The SYN based method provides a good estimate of RTT that can be exploited in real time analysis of Internet traffic. Further, the developer found out that flights are not common and usually not large. In their test environment, they occur with greater frequency in two cases. Either if the flow's RTT is large with medium bandwidth or the RTT is medium with large bandwidth. A third observation is that if the RTT is large then it is possible that the acks are not received during the flow's lifetime. In this case, the flow never makes use of TCP's feedback control.
In consideration of the target environment there's neither a certainty about the existence of flights nor about the duration of TCP connections and if they enter the congestion avoidance mode. For this reason, only the SYN-base method is followed up.

2.1.3 Wren

Wren [14] is a hybrid bandwidth monitoring tool, which passively traces existing application traffic. Additionally it actively injects traffic when necessary to maintain a continuous flow of bandwidth measurements. The survey of this system includes only the passive component.
The Wren system extends the functionality of the Web100 kernel [17] with a buffer to collect characteristics from the incoming and outgoing packets and extend each entry with a timestamp.
The timestamp values of the packets are set according to the CPU clock cycle counter. The recording of the packet traces takes place at both ends of the path. To start the tracing, one machine triggers the other one by setting a flag in the headers of the outgoing packets. The other machine traces only the packets with the header flag set. This ensures the recording of the same range of packets.

Wren uses four different bandwidth techniques to analyze the collected data: Bulk Transfer Capacity (BTC), TCP window, Packet Pair and Self-Loading Periodic Streams (SLoPS).

BTC BTC uses the congestion awareness of TCP to measure the data rate a transport connection can obtain. For this Wren divides the observed total number of bytes sent by the time elapsed.

TCP window The TCP window technique measures the bandwidth by means of TCP's varying congestion window size. Multiplying this variable with the MSS determines the sending rate. Dividing this rate by the RTT calculates the achievable throughput.

Packet Pair The original idea of the packet dispersion technique is to send a train of packets with a rate faster than the bottleneck bandwidth. Finally, the rate that packets are traveling at will be equal to the bottleneck bandwidth. It is inversely related to the spacing between the packets, so the smaller the spacing the faster the rate. Wren uses the idea and analyzes the dispersion of the tightly spaced packet pairs when they arrive at the receiving host. But the success of this analysis depends on the initial rate which has to be greater than the bottleneck, but Wren has no control over the initial sending rate of the streams.

SLoPS SLoPS is used in the measurement tool pathload. Pathload is based on the principle that the one-way delay of a periodic stream increases, if its rate is larger than the available bandwidth. The algorithm of pathload requires cooperation of the sender and receiver hosts. One host actively sends out several UDP packets at a fixed rate and adjusts it in the next iteration after evaluating if it was larger or smaller than the available bandwidth. As already mentioned before, Wren is only using passive techniques and thus it has no control over the initial sending rate of the streams. So there's no guarantee that the algorithm converges to a bandwidth range. But a closer look at the behavior of TCP derives that its protocol essentially does the same as pathload implements.

Advantages

- Redundancy: 4 different techniques to measure the available bandwidth and therefore 4 values to assert the right assumption
- Packet trace happens at kernel-level, therefore the application performance doesn't suffer and there's no need for application modifications
- Packets can be accurately timestamped to nanosecond precision of the kernel

Drawbacks

- Two-sided approach, monitoring takes place at both end hosts of path
- Only for TCP connections

Evaluation The experiments run in different test environments showed some important differences between the four bandwidth measurement techniques. SLoPS, BTC and TCP window are only able to effectively measure the available bandwidth when the TCP protocol starts the AIMD phase. While the packet pair technique could still be applied to more bursty traffic that never opened the AIMD phase of TCP window, it is only applicable to bursts larger than a minimum amount of about 90 KB.
The paper mentioned an uncertainty about the reliability of the packet dispersion technique because of the fact that interrupt coalescing can cause the recorded timestamps to be increased in high bandwidth-delay environments and therefore falsify the measured results. The tests showed, that packet pair is useless when interrupt coalescing occurs. Paper [16] explains, that IC allows a reception of a group of network frames to be notified to the operating system kernel via a single hardware interrupt, thus reducing the interrupt processing overhead, particularly at high packet rates. But this advantage is egalized by the drawback that IC introduces delays that can result in significant TCP throughput degradation.
There are active approaches with these algorithms too since all of them ask for certain traffic patterns to estimate the RTTs. Since the target environment can't rely on certain traffic types, as stated in its criteria list, these algorithms will not be pursued further in this document.

2.1.4 Shared Passive Network Performance Discovery

Shared Passive Network Performance Discovery (SPAND) [15] is designed to measure network performance between well-connected domains communicating with distant hosts through a network with unknown properties. To achieve this, SPAND passively collects application-specific information like response time or Web page download time from a collection of hosts. The measurements are shared within a domain by placing it in a centralized repository.
SPAND is comprised of three different components. *Client Applications* observe the state of the network and summarize its measurements in Performance Reports for the distant hosts. The *Packet Capture*

Host component passively observes all transfers to and from a group of clients and determines an application's performance. From its observation it then generates a report and sends this message to the Performance Server. The *Performance Server*, in turn, stores these reports in a database.

At the time of this writing [15] there are two realizations of the SPAND architecture. The first one measures the performance of generic bulk transfers that use TCP as their data transport by obtaining the following three metrics.

Available bandwidth Total length of a transfer divided by the total time of the transfer, including the initial SYN exchange but not including the FIN exchange

RTT Time when a TCP packet is sent out to when the acknowledgement for that packet is received

Time to completion Time needed to transfer B bytes using a single TCP connection, including connection set-up time

The second one is to measure performance for applications that use HTTP for data transfer. It uses the Web object download time as the primary metric.

Advantages

- Measuring the network quality at application-level shows the performance that the user experiences. Measuring the network quality at application-level shows the performance that the user recognizes.

Drawbacks

- The Packet Capture Host component is not located at the end clients and makes use of several heuristics for measuring metrics like the RTT. This can lead to inaccurate measurements.
- SPAND is only applicable for TCP connections.
- Additional data traffic is generated through sending Performance Reports to a centralized database.

Evaluation SPAND is the only approach that does Performance Reports at application level. This measurement analysis implicates other bottlenecks that don't concern the network quality such as the CPU utilization. On the other hand, application-based performance is what the user perceives and is interested in.
SPAND's packet capture hosts are not located at the end points of the path, and therefore, must use heuristics to infer end-to-end metrics. The capturing points of the target environment are located at the endpoints, so there would be no need for heuristics. SPAND proposes three interesting metrics but SPAND's implementation for their calculation can't be addapted for two reasons. First, SPAND is desingned to measure the performance of specific applications. Since the customer uses differend kinds of application and they aren't predefined, the evaluated metrics shouldn't rely on the application type. Second, the evaluation needs information from other hosts.

2.1.5 Internet Measurement Infrastructure

The Internet Measurement Infrastructure (IMI) [32] architecture uses the RMON[18][2]-type passive monitoring and, for path destination, active probes like traceroute. This survey only summarizes the passive measurement technique of monitoring packets and flows. The architecture consists of several IMI-daemons (IMId) that are distributed over the network. They collect the required information for detecting congestion.
The monitoring is based on *flows*, or flow-identifier, defined as a train of packets with the same 4-tuple *source IP address, source PORT address, destination IP address, destination PORT number*. At the time of this writing [18], IMI only managed to detect congestion of TCP traffic by identifying duplicate sequence numbers in retransmitted packets. The retransmitted packet details are stored in the retransmission log table. IMI tries to find the bottleneck along the path with a distributed analysis.

[2] See Subsection 2.1.1

Algorithm If a daemon IMId-a detects a TCP retransmission in flow F, it executes the following steps:

1. Find the source A and the destination B of F
2. Locate other IMId, if any, between A and B
3. If there are other IMIds along the path, fetch their retransmission logs.
4. If the same retransmission is detected by the other IMId-m then ther is no bottleneck between IMId-a and IMId-m.
5. If the retransmissions detected at IMId-a are not detected at IMId-m then the bottleneck is likely to be between IMId-a and IMId-m.

Advantages

- Simple
- Sharing network data aggregates the available information to encounter problems (e.g. other affected users)
- Traffic generated on demand, therefore fewer messages needed compared to other distributed approaches

Drawbacks

- Only applicable for TCP-connections
- Distributed analysis requires sending of additional SNMP messages through the network

Evaluation Statistics are only discarded in case of TCP retransmissions. Case studies done on a Metropolitan Area Network (MAN) showed that IMI detects congestion with accuracy near 100%.
Further, this distributed Framework can be used in diagnosing and localizing bottlenecks. IMI isn't applicable for the target environment, since it is a distributed approach.

2.2 IP specific monitoring mechanisms

The mechanisms in this section focus on data information generated by the Internet Protocol (IP). It is data-oriented and used by the end hosts for communicating data across a packet-switched (inter-) network. Data is sent in blocks referred to as packets or datagrams. The IP protocol is unreliable, i.e. it makes almost no guarantees about the packet. The packet may arrive damaged, it may be out of order or duplicated, or it may be dropped entirely.
IPv4 is the most common network protocol found in today's public Internet.

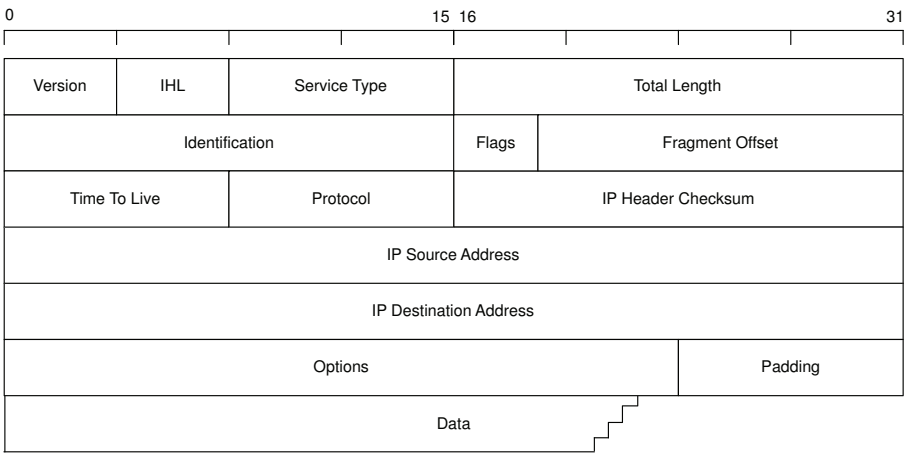

Figure 2.2: IPv4 protocol header

IPv6 is the proposed successor to IPv4.

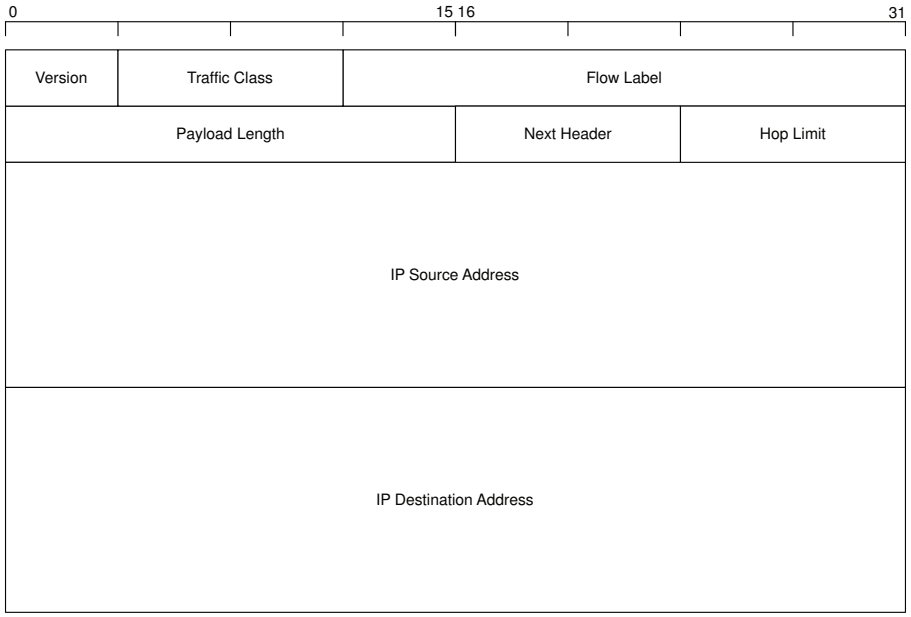

Figure 2.3: IPv6 protocol header

This section's approaches are distributed, i.e. the data is captured at both end points. The host timestamps the collected data, typically denoted as flow. This enables the mechanisms to measure the One-Way Delay (OWD) but requires time synchronization.

One-Way Delay *One-Way Delay (OWD)* is defined as the transfer delay of a packet between two points, in one direction. This metric is more significant as the RTT, taking into account the asymmetries in the internet.

Time synchronization can be achieved in different ways:

Inline Technique This technique is proposed as an active measurement method that does not add probe traffic to the network. It adds information into an active connection in the way that the sender not only observes the packet but also actively adjusts them.

Pulse Per Second *Global Positioning System (GPS)* sources send out *Pulse Per Second (PPS)* signals. The clock synchronization is achieved by this signal.

Network Time Protocol The *Network Time Protocol (NTP)* is a protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks.

2.2.1 Measurement mechanism for IPv6 Inter-networks

The service-oriented measurement mechanisms presented in [4] are using native features of the IPv6 protocol to provide accurate, transparent and reliable traffic flow measurements for any type of traffic.
IPv6 defined the optional functionality as a set of so-called extension headers, which are inserted between the main IPv6 header and the transport layer header. The monitoring mechanism uses the destination options extension header that carries Type-Length-Value (TLV) - encoded options which are only processed by the packet's destination. They implemented two different destination options TLVs: one to measure One-Way Delay (OWD) and one for One-Way Loss.
The algorithm is an inline technique, which decouples the two processes of feeding the packets with measurement data and gathering and analyzing this information. For each defined TLV there exists a pair of source/destination modules that attach themselves to the IPv6 output and input queues. The modules are implemented as dynamically Loadable Kernel Modules running on Linux systems and provide insertion, manipulation, extraction and removal of IPv6 measurement options TLVs.

The source module selects the packets to be instrumented before they are forwarded to the network interface. It then creates the destination options extension header as well as the appropriate measurement TLV and fills the corresponding fields with local measurement data. The destination module observes packets that arrive at the network interface. If a packet contains a destination options extension header carrying the appropriate TLV then the module stores the measurement data. Depending on what is to be measured, it sometimes adds further local measurement indicators to the TLV fields.

Copies of the main IPv6 hearder, the destination options TLV and the transport layer header are carried to the user-space for further data analysis.

Advantages

- Useable for any type of traffic

Drawbacks

- The overhead of 20 respectively 8 Bytes per instrumented packet can influence the network performance in several ways
  - Fragmentation must be avoided
  - Additional packets due to larger data size

Evaluation Several tests are run over a variety of different-capacity network configurations fed with representative types of traffic. This technique produces an overhead of 20 (time-related measurement) respectively 8 Bytes (packet loss measurement) per instrumented packet. So this means for TCP, that more packets are needed, and so more time, to complete the data transfer. For UDP, the overhead was mainly the additional processing delay of adding the extension header to each measured packet. Another important fact is that packet fragmentation due to the additional destination options extension header has to be strictly avoided. Another implication of [4] is that the IPv6-in-IPv4 tunneling might influence to the performance. Although this is not a distributed approach it is inapplicable for the target environment since no header field manipulation or extension is allowed.

2.2.2 One-Way Delay passive measurement system

The One-Way Delay passive measurement system described in [29] focuses on the measurement of the OWD between two network points. It is composed of several *Measurement Points (MP)* and a *Data Collector (DC)*.
A MP collects packets, filters a subset out of them and builds a report containing the packet's timestamp and an identifier, referred to as packet ID. These reports are then reported to the DC where the OWD is evaluated. The fundamental issue is to consistently sample at the two MPs the same set of packets and label them with the same packet ID, so that the DC is able to match the packets.
The OWD calculation depends on the timestamps, generated by the two MPs. For this, clocks of the two MPs have to be synchronized, as already mentioned in the introduction. This is achieved by the Pulse Per Second signal from a GPS source.
To obtain the packet ID, the system applies a hash function over an immutable part of the collected packet's IP header. It is possible that two different packets hash to the same packet ID. The collision detection scheme, which is the most challenging part of the DC, is responsible for this problem. Further tasks of the DC are the filtering of reports, which were sent by only one MP or were delayed.

Advantages

- Accurate measurement of OWD is more significant as the RTT metric.

Drawbacks

- Precise synchronization of MP is needed (here achieved by GPS).
- Additional traffic is generated by sending reports to the DC.
- Design of DC is challenging because it has to detect collision and filter out delayed or single MP's reports.

Evaluation At the time of the report [29], IMI was tested according to tradeoffs for cost vs. precision but was not tested in real network.
Measuring OWD is more accurate than RTT calculations, since it factors the direction into its evaluation. OWD, instead of RTT, is therefore preferred but IMI is not applicable for the target environment. On the one hand, IMI's design is very challenging requires clock synchronization and it's uncertain if IMI performs well in real networks. On the other hand, IMI uses a distributed approach for exchanging information.

2.3 Statistical monitoring mechanisms

The following abbreviations and terms are often used by the monitoring systems described in the subsections 2.3.1 - 2.3.10.

Network Management Systems *Network Management Systems (NMS)* monitor real-time traffic flows to recognize the (sub-) network's behavior. If they detect changes in its deduced characteristics they propose reactions to improve the situation.

Internet Engineering Task Force The *Internet Engineering Task Force (IETF)* is an organization which produces technical standards for the Internet, published as RFCs (Request for Comment). Within the IETF there are many active Working Groups, each pursuing standards activities in their area of interest.

Realtime Traffic Flow Measurement *Realtime Traffic Flow Measurement(RTFM)* is an RFC defined by one of IETF's working group. It proposes RMON, the standard, generalized asynchrounous and distributed architecture for measuring traffic flows.

Remote network Monitoring Remote network Monitoring (RMON) is a *network management standard* proposed by the IETF.

Simple Network Management Protocol The *Simple Network Management Protocol (SNMP)* forms part of the internet protocol suite, defined by the IETF. The SNMP protocol operates at the application layer and can support NMS to monitor and administrate a set of managed objects.

Management Information Base A *Management Information Base (MIB)* is a type of database used to describe a set of managed objects. SNMP makes use of MIBs. An SNMP management console application can then manipulate the objects by using a specified set of commands and queries. A MIB should contain information on these commands and on the target objects to finally tune the network transport to the current needs.

2.3.1 Argus

The network Audit Record Generation and Utilization System (Argus) [1] is an open project that is focused on developing network activity audit strategies. The software consists of a fixed-model RTFM designed to monitor and report the status and performance of all network transactions seen in a data traffic stream.
Argus defines *multiple bi-directional flow* models. The models depend on the layer and/or protocol of the tracked packets. For example, layer 4 TCP and UDP flows are defined as 5-tuple, source IP address, source Port, destination IP address, destination port, and protocol. On the other hand, a layer 3 IPv4 flow is defined as 3-tuple, source IP address, destination IP address, and protocol. For each type of flow there are a set of common identifiers and metrics that Argus tracks. Both the flow-models as well as the recorded metrics are listed in [1].
For all flows the system record, among other things, the start and end of each flow, packet count, and byte count as well as application byte count. With these metrics, it reports the following information for performance analysis.
For IP network performance of a transaction it informs about:

- Connectivity and reachability
- Load
- Packet loss

- Round trip delay
- IP packet jitter and jitter variance

Additional metrics are available for TCP-based flows:

- Window Performance
- Host Flow Control Indications
- Explicit Congestion Notification Indications

Argus can generate flow status records for the contents of captured flows. They are called Flow Activity Record (FAR) and are generated in three different cases. Either when the transaction is started (Start FAR), stopped (Stop FAR) or if a life of a given flow extends the defined Flow Status Timer (Status FAR). The Status FAR is then used to report that the flow is still active.
Argus can be deployed to monitor individual end-systems or an entire enterprise's network activity.

Evaluation Argus combines their evaluation of the network quality by real-time calculation of several different metrics as well as statistical analysis of the network behavior. The project doesn't give an exact explanation how they calculate their metrics and is therefore not considered further.

2.3.2 NeTraMet

NeTraMet [19] is an open-source implementation of RTFM .The architecture defines three network entities:

- Meters gather data about packets and consolidate this to provide flow data
- Meter Readers retrieve flow data from meters using SNMP
- Managers coordinates the activities of meters and meter readers

RTFM identifies a *flow* by a *bidirectional* 5-tuple *(protocol, source and destination addresses, and source and destination port numbers)* having byte and packet counters for each direction. RTFM uses rulesets to describe the flows which a meter is required to measure. Rulesets are usually written in the Simple Ruleset Language.
The implementation's distribution package includes NeTraMet, an RTFM Meter, and NeTraC, a combined RTFM Meter Readers and Manager. The manager reads a configuration file to start collecting the flow data. This file specifies the rulesets to be run, the meters on which to run them and the time intervals when the flow data is to be read. The flow data is then stored in flow data files and can be used to produce daily reports.

Advantages

- Stable and reliable open source implementation

Drawbacks

- Different meters produce redundancy. This requires additional space in memory.
- SNMP packets are sent to the manager/reader for data collection which generates additional traffic

Evaluation According to [19] the RTFM architecture has proved very stable and reliable in use. Further, several sites have performed tests for accuracy of NeTraMet compared with other measurement tools and have reported very good agreement. Since NeTraMet is distributed it doesn't meet the criteria for the target environment.

2.3.3 NetFlow

NetFlow [2], implemented in Cisco routers, is the most widely used measurement solution today. Routers running NetFlow maintain a "flow cache" containing flow records that describe the traffic forwarded by the router. These flow records then are sent using unreliable UDP to a computer that collects, analyzes and archives them.
For each router interfaces, *flows* are identified by important fields of the header: *source IP address, source PORT address, destination IP address, destination PORT address, protocol, and type of service*. For each flow, the router inserts a new flow record in its cache, which keeps additional data such as the

number of packets and bytes and the timestamps for the first and the last packets. As soon as a flow has ended, it is exported to a centralized computer for further analyzation. NetFlow uses four rules to decide whether a flow has finished:

1. End is indicated by TCP flags (FIN or RST)
2. Timeout: Timestamp of the last packet with matching flow ID is a configurable amount of time away.
3. Timeout: Record was created before a configurable amount of time.
4. Flow cache is full

Cisco introduced sampled NetFlow for computers whose processor and memory cannot keep up with the packet rate. NetFlow samples for a configurable parameter N only one of N packets.

Advantages

- System can be optimized for certain type of flows
- Provides information such as
  - Who is using the network, what applications
  - When is the network utilized
  - Where is the traffic going on the network
  - Aggregate traces of packets into a flow record to reduce the amount of traffic

Drawbacks

- Other tests run with NetFlow listed in [3] several problems
  - If the number of produced records is larger than expected, the router exports these entries. The traffic they generate can cause the network to drop packets.
  - NetFlow cannot estimate the number of flows
  - The sampling rate involves tradeoffs between the required amount of CPU and memory usage and the accuracy of the analysis.
  - Analysis groups traffic into intervals. NetFlow estimates how much of a flow that doesn't fit in an interval belongs to each interval. These heuristics introduces additional inaccuracy.
  - Unusual traffic mixes causes NetFlow to crash, e.g. if small flows predominate, NetFlow aggregation doesn't help to keep the needed amount of memory small

Evaluation As listed in the drawbacks, using NetFlow brought up several problems. For some of them improvements were proposed. NetFlow is the traffic measurement solution most widely used by Internet Service Providers (ISPs) to determine the composition of the traffic mix. But NetFlow delivers too little information for measuring the network quality of the target environment. Since NeTraMet is distributed it doesn't meet the criteria for the target environment.

2.3.4 Ntop

Ntop is an open-source web-based traffic measurement and monitoring application. It was initially designed for tackling performance problems of the campus network backbone [9] and then evolved into a more flexible and powerful intrusion detection system [10].
The ntop architecture is divided into three independent components.
The *packet sniffer* collects network packets, filters them and passes them to the packet analyzer. It supports different network interface types and can handle multiple of them simultaneously.
The *packet analyzer* processes one packet at time. The packet headers are analysed according to the network interface being used. Host information is stored in a large dynamic hash table whose entries contain several counters that keep track of the data sent/received by the host, sorted according to the supported network protocols. The hash entry received, or created if not present yet, corresponds to packet source and destination.
To improve the overall performance and decrease the amount of data, ntop performs *caching* in two steps. First level caching is based on GNU gdbm and contains semi-persistent information such as IP

address resolution or remote host operation system. Second level caching stores permanently network events like TCP sessions, performance data and other relevant information is into a database.
Traffic reports are done by the *Report Engine* in either text or HTML.

Advantages

- Efficient packet processing by using hash tables
- Low packet loss because captured packets are buffered twice, inside the kernel and ntop

Drawbacks

- In case of packet loss (buffers full), there occurs problems with the fragment entries
- Ntop doesn't know that peer intends to close the TCP connection (three-way handshake) [9]
- Dynamic hash [8]
  - Hash extension could fail on some systems
  - Hash resize could packet loss was experienced
  - Performance degradation in case of frequent memory management calls
  - Fixed-sized hashes have more efficient hash indexing

nProbe NProbe [8] is open source software for handling Gigabit traffic. It extends ntop by adding NetFlow support. The application therefore captures packets flowing on an Ethernet segment, computes NetFlow flows and exports them to the specified collectors. The collectors can be commercial applications such as Cisco NetFlow Collector or open source tools such as ntop.

Evaluation A lot of performance tests were done with Ntop and its successor NProbe. They are all summarized in the advantages or drawbacks. No analysis about the accuracy of the measurements is reported.
All reports describe the architecture in detail, but they don't document the algorithm for calculating the metrics. Therefore, these two mechanisms are discarded.

2.3.5 Remote network Monitoring

As the networks and services have become larger and more complex, their management gained importance. All the traditional standards of NMS use a centralized client/server paradigm, but the NMS servers cannot collect and process information of a large and complicated network efficiently.
Remote network Monitoring (RMON) [18] is a network management standard to reduce the burden of the NMS server. A NMS using web-based RMON agents systems has a hierarchical distributed architecture.
Each *RMON agent system* or RMON probe catches all packets for the subnetwork attached to it, analyses their headers and stores the necessary information. The centralized NMS server accesses the RMOM agents systems to obtain statistics on the subnetwork.

The RMON probes consist of three components:

Configuration control module It receives from the server control functions and configures them as well as the parameters associated to them.

Packet capture module It captures the packets by using the libcap packet capture driver and stores them in a file.

Packet analyzer module It reads the packets stored in the file, analyzes the packet headers and collects the necessary information.

The information to collect is defined in two different *MIBs*. RMON-1 MIB defines information on the statistics of the subnetwork based on a MAC address. RMON-2 MIB extends the RMON-1 MIB by defining that information of the subnetwork based on the network layer and application layer addresses. To reduce the network traffic between the NMS server and a RMON agent system, RMON-2 uses a TimeFilter as an index to the required statistic table. The agent only sends the entries which are changed after the time value given by the filter.

At the time of writing [18], the system implemented 8 RMON MIB groups, which defines statistics such as total number of packets, the number of errored packets or the packet size distribution.

Advantages

- The analysis process happens at the RMON agents and reduces required processing power compared to a central analysis at the NMS server
- TimeFilter reduces the additional traffic, only the updated data is sent to the server

Drawbacks

- Additional traffic is generated by sending the analyzed data to the NMS server

Evaluation RMON outlines some interesting metrics to evaluate. According to this paper it further reduces the processing and traffic burden of a NMS server. But RMON isn't applicable in consideration with the target environment because it is a distributed mechanism that sends analysis data to a centralized point.

2.3.6 Trace

The Trace Management Platform presented in [12] and [13] is an extension of the SNMP Infrastructure based on the IETF Script MIB to support integrated, distributed and flexible management of high-layer protocols, services and networked applications.

The architecture consists of several components structured in a three-tier model.

> The *management stations* or managers selects the protocol traces of its interest by specifying them using the *Protocol Trace Specification Language (PTSL)* and define management tasks. Additionally, they determine the *mid-level manager (MLMs)* and delegate these tasks through SNMP to them.
>
> The *MLSs* interact, in turn, with monitoring and action agents to execute these tasks and report major events back to the managers.
>
> The *monitoring agents* count the occurrence of the specified traces between a pair of peers. Every time a trace is observed, the data is stored on a mySQL database. The MLMs are then able to evaluate the captured traces and react to certain traffic patterns. When a MLM detects a problem it can run a corresponding script written in Java, Perl or Tcl.
> The *action agents* are responsible for the execution of reactive management procedures, which were created to handle the perceived problems. Therefore the MLMs communicate with the action agents via the Script MIB.

The Trace platform is used in [11] to monitor the response time of transaction-based Internet applications. The implementation extends the monitoring agent so that it stores response-related information.
It specifies with PTSL the protocol transactions whose response times are to be measured. To configure performance measures and retrieve results, the manager interact with the agent via SNMP trough a subset of Application Performance Measurement MIB proposed in [31].
Examples of statistics offered by the MIBs are the maximum, minimum and mean response times or the number of successfully completed traces.
Further, the MIB supports different types of aggregation. For example, reports can be generated for each trace observed or only for each combination of trace and client.

Advantages

- PTSL allows to select the granularity at which protocols are monitored
- Trace is not limited to monitoring but enables management of high-layer network protocols
- Network-management: can react to poor network performance

Drawbacks

- Additional traffic generated by the sent SNMP messages
- Distributed approach is more complex and requires more control

- Computational costs for computing performance-related statistics done in [11]

Evaluation The documents found about Trace neglect some results about the accuracy of the captured metrics. It informs only about their performance experiences. They are listed in the advantages and drawbacks, respectively.
The focus of the papers lies more on PSTL and less on an algorithm for metric calculations and therefore no further investigation in analysis of Trace is done.

2.3.7 OCXmon

The National Laboratory for Applied Network Research (NLANR) is developing a network analysis infrastructure (NAI) to support network research and engineering through the collection and publication of raw data, visualization and analysis of network measurements. The NAI [34] includes a passive monitoring project, an active monitoring project and the collection of network management and control data, i.e. Border Gateway Protocol data and SNMP data. In this survey, only the passive monitoring mechanism using OCXmon is described.
OCXmon monitors have two measurement cards installed so that they can capture, in an asynchronous transfer mode (ATM), the traffic in both directions of a full-duplex connection.
When an IP packet is sent over an ATM connection, it is broken into 48-bytes pieces called cells. The first cell includes the IP and TCP or UDP header and, if any, very little user data, if there are no optional headers in use. The last cell is marked to indicate the packet is complete. Traces of just the first cell from each packet are captured from each monitor several times per day and stored according to the OCXmon trace file format.
After the measurement period some analysis programs are run on these files and both the results of the programs as well as the captured files are transferred to the central NAI machines which publish the data on the Web.
The analysis of the files includes bidirectional transaction analysis and flow analysis. For further analysis the incorporate evaluation tools of other project such as CAIDA[3]'s CoralReef software or the Cichlid tool developed at NLANR.

Advantages

- Visualization of a wide range of network data
- Timestamps are instantly added by the measurement card

Drawbacks

- Only statistical analysis of network behavior done
- Publishes information on the Web, therefore the identity of the users has to be protected
- Additional traffic through sending data files to a central NAI machine

Evaluation The paper focuses on data capturing and doesn't give any information about testing or experience.
As mentioned in the drawbacks only statistical analysis of network behavior is done. This is too coarse-grained for precise network quality reports. That's why this mechanism isn't followed up.

2.3.8 High Performance IP Monitoring Agent for CERNET

The monitoring agent presented in [40] is characterized by a high-performance data collector and a methodology of real-time data analysis and correlation. Customized agents are monitoring particular links of the China Education and Research Network (CERNET) to provide input for network intrusion detection systems, network management systems and accounting and billing systems.
The monitoring agent collects, and aggregates packets to produce either flow records or packet records. The record's timestamp is taken from a globally synchronized clock provided by the network time protocol[4]. The real-time data analysis is performed by user-level multithreading applications. The packet meter, or filter, can filter packets in real time according to dynamically configured rules, using a routing lookup algorithm. Further, it produces statistics such as

[3] Cooperative Association for Internet Data Analysis. CAIDA provides tools and analyses promoting the engineering and maintenance of a robust, scalable global Internet infrastructure, http://www.caida.org/home/
[4] See 2.2

- Packet size distribution
- Packet source/destination port distribution
- Packet protocol distribution
- Packets per second
- Average packet size

The output of the packet meter is sent to the packet classifier. This module classifies the packets into flows by using the Recursive Flow Classification algorithm. Then additional flow-based analysis can be done:

- Top N statistic concerning traffic volume, address uniformity, port settings, number of packets or flows
- Traffic matrices
- Link utilization
- Traffic breakdowns by protocol, network services, etc.
- Routing loops and errors

The records are stored in a data repository to do further analysis. For this, they correlate, associate and refine the measurement data.

Advantages

- No data exchanges between several points needed
- Filter mechanism reduces the needed data storage space

Drawbacks

- Real-time analysis of only one point or link
- Need of clock synchronization for further off-line analysis and correlation of measured data of several links

Evaluation Performance tests showed that the passive monitoring agent is capable of supporting Gigabit Ethernet data rate. According to [40], the monitoring results are useful in many diverse systems such as IDS, NMS and ABS. The paper only lists the metrics for measuring network quality but it totally neglects some algorithms for their calculations. Therefore, it's not further examined.

The focus of the following proposals lies only on the analysis of monitored data.

2.3.9 Measurements Analysis of backbone paths

The focus of paper [23] lies on the analysis of active and passive measurement data collected over several months on three hierarchically different network backbone paths. This survey only summarizes the analysis methods applied to the data collected by different passive measurement tools.

The paper [23] defines following measured characterisics:

Availability is calculated by measuring the uptime or downtime of a network device or service. Additionally, the syslog data is analyzed to ensure that none of the observed network events are caused by unexpected router hardware or software failures.

Utilization is a SNMP metric that compares the amount of inbound and outbound traffic versus the bandwidth provisioned on a link in a network path.

Discards is a SNMP metric that indicates the number of packet discarded for a particular network interface. Errors are a SNMP metric that indicates the number of interface errors (e.g. Frame Check Sequence errors). Large values of errors and discards are an indication of excessive network congestion.

The NetFlow data is used to determine the kinds of traffic and their distribution.

Advantages

- Different types of collected data allow different evaluations

Drawbacks

- Not real-time, only useful to show tendencies or traffic behavior of the observed traffic

Evaluation The paper outlined the measured data captured with the monitoring agent but there isn't any information about the accuracy of the data analysis.
The document proposes good metrics for measuring network quality but their analysis doesn't take place in real-time. Therefore, it's not further examined.

2.3.10 Analysis of End-to-End Internet Faults

Microsoft Research [36] describes an analysis framework to characterize end-to-end Internet failures. Therefore, they define that a web transaction consists of a client resolving a web server name to the corresponding IP address, establishing a TCP connection to the server, and downloading the object of interest using the HTTP protocol.
Figure 2.4 divides transaction failures into three categories, depending when the failure occurs.

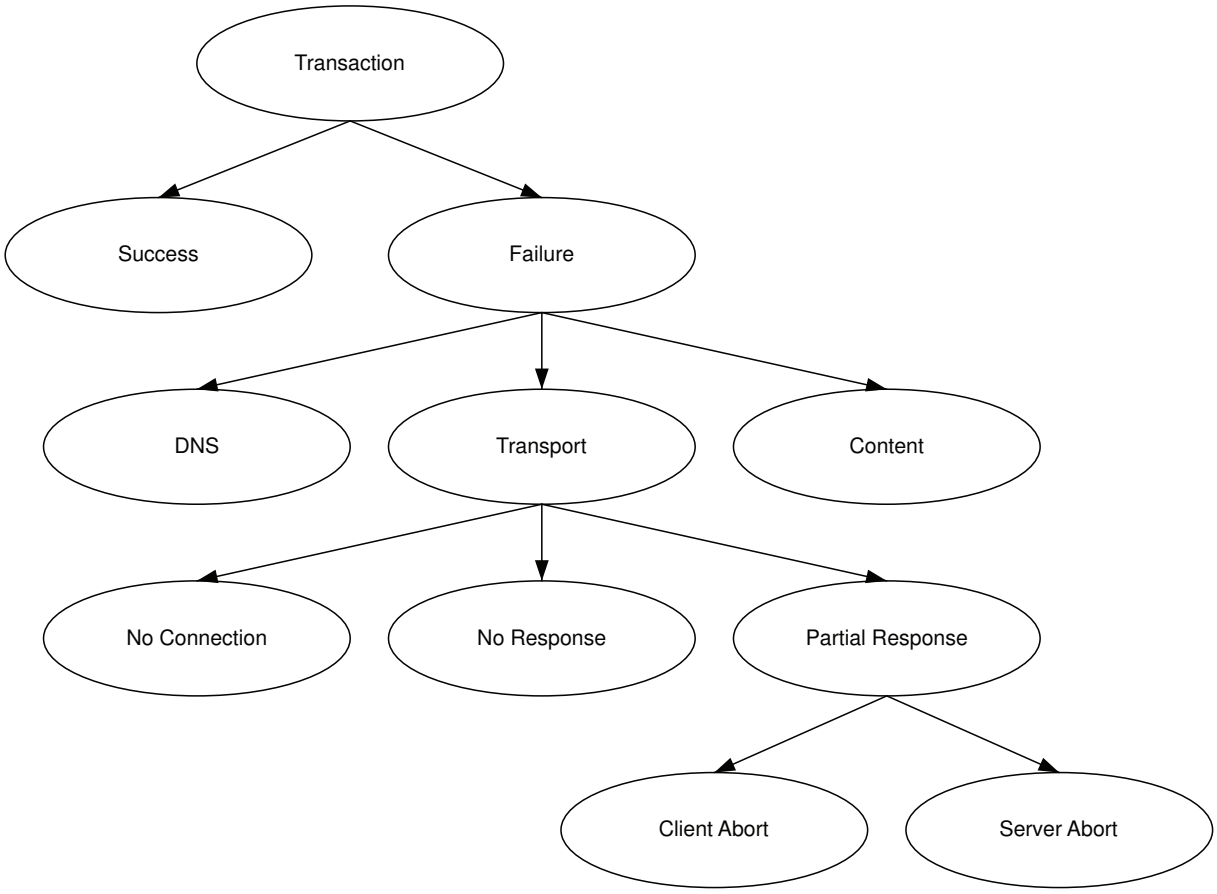

Figure 2.4: Transaction Failure Categories

DNS website name cannot be resolved

Transport failure at the TCP connection level

Content TCP connection is successful but the server doesn't supply the desired content and returns an error indication instead.

This survey focuses on the *Transport failures* which can in turn be categorized in three parts.

No Connection The client is unable to connect the server, therefore no SYN handshake.

No Response The client is able to establish a connection and sends its request but it doesn't receive any response.

Partial Response The client receives only part of the server's response before the connection is prematurely terminated.

Further analysis can then be done by correlation of the local failure observations. Hence, failure episodes have to be identified. Failure episodes correspond to periods with abnormally high failure rate and are categorized from the viewpoint of end users:

Client-side A client is experiencing an abnormally high aggregate failure rate in its communication across many servers. This suggests a root cause at or close to the client.

Server-side A server is experiencing an abnormally high aggregate failure rate in its communication across many clients. This suggests a root cause at or close to the server.

Client-Server-specific A specific client-server pair is experiencing an abnormally high failure rate, but neither client nor server is experiencing an abnormally high failure rate in aggregate.

Other

At first, DNS related information is recorded by the two tools dif and wget. The data measured includes DNS lookup time, the download time, and the failure code, if any.
The analysis of TCP-level information, recorded by tcpdump or windump, consists of two parts: First, the cause of connection failures is categorized by no connection, no response and partial response. Second, packet retransmissions are counted. This is done by counting the total number of packets sent by the server and the number of retransmitted packets. The packet counts are aggregated across multiple connections to compute the loss rate. Attention should be paid to differ between packet retransmissions and packet reordering. The distinction is done by comparing the length of time by which a packet is out of order with the connection's RTT.

Advantages

- Simple
- Classification of traffic failures by the view of end-user
- Part of the analysis is applicable by all TCP connections, i.e. not restricted to web transactions

Drawbacks

- Analysis requires data from both end points
- Only for TCP connections

Evaluation The mechanism gathered failure data over a period month. The evaluations indicate a promising approach to diagnose Internet faults. But the feasibility of this system hinges on several client-based issues. Since no predictions about the client behavior can be done and the fact that this system needs shared information for the classification, it doesn't fulfill the criteria of the target environment.

2.4 Classification and Comparison

Summary The following two tables give an overview of the discussed monitoring mechanisms and is based on the Sections 2.1 - 2.3. Table 2.3 summarizes the protocol specific measurement mechanisms.

Mechanism	Protocol	Distributed	Capturing Point	Proposed Algorithm(s)	Calculated Metric(s)
PRE (2004)	TCP			gap detection	*Real-Time:* RTT
Wren (2004)	TCP		end host, kernel level	(1) BTC (2) TCP window (3) Packet Pair (4) SLoPS	*Real-Time:* RTT
Estimate RTT (2002)	TCP		end point, kernel level (DAG cards)	(1) SYN-based (2) Flight (3) Rate Change	*Real-Time:* RTT
SPAND (2000)	TCP	x centralized per-domain repository	not at the end points, application level		*Real-Time:* available bandwidth RTT time to completion
IMI (1999)	TCP	x local data storage	not at the end points, kernel level	Duplicate sequence Numbers	*Real-Time:* Retransmissions
mechanism for IPv6 (2004)	IP	(x) in-line technique	end points (2), kernel level		*Real-Time:* one-way delay one-way loss goodput/throughput
OWD measurement (2004)	IP	x centralized data collector	end points (2), kernel level	(1) OWD (2) probability of analysis failures	*Real-Time:* OWD

Table 2.3: Summary of protocol specific approaches

Table 2.4 on the next page does the same for the flow-based mechanisms.
The two tables point out that protocol specific systems are well suited for calculating real time metrics such as the number of retransmissions during a connection. Further, all the time-related metrics of the protocol headers need to be timestamped. Table 2.3 points out that calculating the OWD needs a distributed approach, while RTT estimation only requires monitoring at a single point.
Flow-based methods are used for statistical analysis. This analysis can either be real-time or the measured data can be evaluated later. Further, it can be based on all packets or only on a selection. Unfortunately, the implementation details of the systems listed in table 2.4 aren't very accurate. They only specify different distribution metrics, and sometimes partly the flow selections, but they don't describe their analysis algorithms to calculate their results.

Conclusion The investigation of existing passive monitoring mechanisms offers a wide range of possibilities for analyzing the network performance. Table 2.5 concludes this survey by summarizing the most important advantages and drawbacks of the methods discussed in Subsections 2.1.1 - 2.3.10, as interpreted from the papers studied.
The descriptions of some approaches don't contain the required details and have many open questions about implementation details as well as performance issues behind. Nevertheless, the detailed evaluation according to the target environment[5] as well as the results stated by the paper's authors are presented after each individual mechanism. They bring up that some ideas of monitoring mechanisms related to the transport layer are reusable for the design of the target environment's monitoring mechanism. Further, they identify the limits of the passive approach.

[5] See Introduction 1.4

Mechanism	Distributed	Capturing Point	Analysis Type
Argus (2005)	x	end users, kernel level	*Real-Time:* Round Trip Delay Packet Loss Statistical: Connectivity
NeTraMet (2001) (RTFM)	(x)	single or multiple end users, kernel level	*Statistical Real-Time analysis:* *Statistical event records* about flow duration, application type, etc.
NetFlow (2004)	x (centralized DB Server)	Cisco meters at routers, kernel level	*Statistical:* e.g number of bytes, flag fields, time of the flow
Ntop (1999)	x (SQL DB, application client communicates via UDP)	end users, kernel level	*Statistical:* e.g. protocol distributions, network service usage, network utilization, IP/non-IP distribution, open sockets, etc.
RMON (2001)	x centralized NMS, collect already analyzed data	one agent per subnetwork, kernel level	*Statistical Real-Time analysis:* e.g. packet size distribution, total number of packets, errored packets
Trace (2002 - 2004)	x	one or more managers, kernel level	*Real Time:* response time *Statistical Real-Time analysis and management:* react to specified traffic patterns e.g. number of traces in an interval, minimum response time, mean response time, etc.
OCXmon (2000) (NLANR)	x centralized data storage	multiple monitors, kernel level	*Statistical analysis* of some time intervals
Cernet (2003)	x no message-exchange	multiple monitors, kernel level	*Statistical analysis:* e.g. packet size distribution, packets per second, average packet size

Table 2.4: Summary of the statistical approaches

Outlook The conclusion raises the question whether or when passive monitoring is better than active probing. Therefore, Subsection 2.5.1 of the next Section 2.5 compares passive monitoring versus active probing in more detail. Subsection 2.5.2 follows with an overview about existing active monitoring techniques.

2.5 Active Monitoring Mechanisms

With regard to the Conclusion and Outlook of Subsection 2.4, more investigation in the analysis of active probing techniques is necessary.

2.5.1 Passive Monitoring vs. Active Probing

As already mentioned in the Introduction 1.1, network measurement techniques can be classified into two categories: passive measurement and active probing. While the passive approach monitors the existing traffic that passes a certain point, active probing injects test traffic and measures the service obtained from the network. Both of them have their values as well as drawbacks; hence they should be regarded as complementary.

Mechanism	Distributed Components	Generates Network Traffic		Protocol Specific
		through data collection	through configuration	
PRE (2004)	-	-	-	+
Wren (2004)	-	n.a.	+	+
Estimate RTT (2002)	-	-	-	+
SPAND (2000)	+	+	-	+
IMI (1999)	+	+	-	+
mechanism for IPv6 (2004)	+	-	-	+
OWD measurement (2004)	+	+	n.a.	+
Argus (2005)	+	+	+	-
NeTraMet (2001) (RTFM)	+ -	+ -	- -	- -
NetFlow (2004)	+	+	-	-
Ntop (1999)	+	+	-	-
RMON (2001)	+	+	+	-
Trace (2002 -2004)	+	-	+	-
OCXmon (2000) (NLANR)	+	-	-	-
Cernet (2003)	(+)	-	-	-

Table 2.5: Drawbacks and advantages of the summarized monitoring mechanisms

Passive Monitoring

Advantages

- Doesn't increase the traffic on the network
- Measures real traffic

Drawbacks

- Limited to paths that have recently carried user traffic
- Limited ability to isolate exact fault location
- Unknown conditions of observed traffic
- Unknown volume of gathered data
- Privacy and security issues

Active Probing

Advantages

- Explore entire network
- Can be generated under well-defined and controlled conditions

Drawbacks

- Measures artificial traffic (evaluated metrics may be incorrect)
- Creates extra traffic, intrusive

2.5.2 Active Monitoring Mechanisms

Several groups have designed active monitoring mechanisms to measure the network performance. They focus on techniques for estimating the bandwidth. In the context of data networks, the term bandwidth quantifies the data rate that a link or a path can transfer. But this term is too imprecise. This Section therefore starts with an introduction of three bandwidth metrics before it continues with an overview about the existing concepts. The definition of these metrics is taken from [28].

Capacity The *capacity* is the maximum possible bandwidth that a link or a path can deliver.

Available Bandwidth The *available bandwidth (ABW)* is the maximum unused or spare capacity at a link or a path during a certain period of time.

Bulk Transfer Capacity The *Bulk Transfer Capacity (BTC)*[1] is the maximum throughput obtainable by a single TCP connection.

Different applications place different requirements on the network. Real time applications may require little bandwidth but are very sensitive to delay, while, at the other extreme, bulk transfers are very insensitive to delay but require as much bandwidth as possible. Taking this into account, the existing mechanisms can be grouped into two main categories: bulk transfer or packet dispersion with probing packets.

Bulk Transfer Capacity This very simply method consists of simply sending data from A to B at the highest possible rate. This method has the advantage of correctly measuring the network throughput but it is very intrusive, since it actually saturates at least one of the links on the path between A and B. In most cases, BTC is an upper bound on the available bandwidth and a lower bound on the path's capacity.

Probing Packets Techniques using probing packets for evaluating network performance can be further subdivided in four major techniques: *Variable Packet Size (VPS)*, *Packet Pair*[1] */ Train Dispersion (PPTD)*, *Self-Loading Periodic Streams (SLoPS)*[1] and *Trains of Packet Pairs (TOPP)*. The former two techniques estimate the capacity, the latter two measure the available bandwidth.

VPS VPS probing needs only one monitor site to measure the capacity of each hop along a path. To measure the RTT from the source to a particular router it uses the *Time-To-Live (TTL)* field of the IP header. If it expires at the particular hop, the router discards the packet and returns an ICMP "Time-exceeded" error message back to the source.

PPTD PPTD probing techniques require double-ended measurements. *Packet Pair (PP)* probing measures the end-to-end capacity by sending multiple packet pairs from the source to the destination. Each packet pair consists of two packets of the same size sent back-to-back. By measuring the changes in the packet spacing at the receiver, the sender can estimate the bandwidth properties of the network path. To filter out erroneous bandwidth measurements the tools propose union and intersection statistical filtering as well as sending many packet pairs with variable size. *Packet Train Dispersion (TD)* probing extends PP probing by using multiple back-to-back packets.

SLoPS SLoPS attempts to bring the stream rate close to the available bandwidth with an iterative algorithm similar to binary search. To measure the end-to-end available bandwidth the source sends a number (about 100) of equal-sized packets to the sink at a certain rate. The receiver notifies it about the OWD trend of each stream. Thereupon the sender adjust the rate of the next packet train.

TOPP TOPP uses a similar approach as SLoPS to estimate the available bandwidth. The main difference between the two methods lies in the statistical processing of the measurements. While SLoPS uses a binary search for the rate, TOPP increases it linearly. Further, TOPP can also estimate the capacity of the tightest link on the path.

More recently, new techniques modify the self-loading methodology of SLoPS or TOPP. These techniques can be distinguished in two main double-ended approaches:

[1] See also Section 2.1.3: Approach to use these techniques in a passive way.

Probe Gap Model The *Probe Gap Model (PGM)* evaluates the time gap between the arrivals of two successive probes at the receiver. PGM use a train of probe packet to generate a single measurement.

Probe Rate Model The *Probe Rate Model (PRM)* is based on the concept of self-induced congestion. If the source sends packet streams at a lower rate than the available bandwidth, the arrival rate at the receiver will match the sending rate of the source. In contrast, if the probe traffic is sent at a rate higher than the available bandwidth, the arrival rate at the receiver will be less than the sending rate. Thus, the techniques are searching the turning point of the rate.

Table 2.6 lists existing active probing tools, each of them implementing one of the explained techniques.

Tool	Metric	Technique	End-to-End	Monitoring interval
Pathchar	Capacity	VPS	No (per hop)	Single-ended
Clink	Capacity	VPS No (per hop)	Single-ended	
Pchar	Capacity	VPS	No (per hop)	Single-ended
Bprobe	Capacity	PP	Yes	Single-ended
Nettimer	Capacity	VPS, PP	Yes	Both
Pathrate	Capacity	PPTD	Yes	Double-ended
Sprobe	Capacity	PPTD	Yes	Single-ended
Cprobe	ABW	TD	Yes	Single-ended
Pathload	ABW	SLoPS	Yes	Double-ended
IGI	ABW	PGM[2]	Yes	Double-ended
Spruce	ABW	PGM[2]	Yes	Double-ended
Pathchirp	ABW	PRM[3]	Yes	Double-ended
Treno	BTC	Emulated TCP throughput	Yes	Single-ended
Cap	BTC	Standardized TCP throughput	Yes	Double-ended
Iperf	Achievable TCP throughput	Parallel TCP connections	Yes	Double-ended
Netperf	Achievable TCP throughput	Parallel TCP connections	Yes	Double-ended

Table 2.6: Active Proping Tools

Additionally, there are several projects ongoing that develop infrastructures of Active Internet End-to-End Performance Measurements (AIEPM). For further information, see [7], which compares the 5 major public AIEPMs.

2.6 Conclusion

The goal of this Thesis is to design and implement a passive monitoring mechanism to evaluate the current network quality. All highlighted ideas of the survey are TCP specific. Since the target environment's traffic type is unpredictable, these ideas indicate a first metric to evaluate the network performance but they don't cover the whole range of the customer's traffic. To expand the range of possible metrics, more detailed studies in the targeted VPN environment were pursued and described in Section 3.1. In conjunction, proposals of active probing algorithms can be developed if the passive monitoring mechanism achieves its limits. This happens when the detailed analysis of a detected network problem is best done using active probing. Subsections 3.5.1 and 3.5.2 design alerting mechanisms that use active probing for further analysis purposes.

[2]PGM is based on the concept of SLoPS.
[3]PRM is based on the concept of SLoPS.

Chapter 3

Design and Implementation

A large number of different passive monitoring mechanisms has been observed in the Sections 2.1 - 2.3. Chapter 2 helped to gain a basic knowledge about network monitoring but no existing technique is directly applicable to the target environment. The RTT metric of the TCP-specific approaches[1] is used to measure the TCP traffic performance in the customer network. But since the type of the network traffic is unknown the design of the Thesis's monitoring mechanism can't depend only on this one metric. However the existence of an IPSec protocol is definitely guaranteed. The design of Open System's environment was analyzed and is documented in Section 3.1. It assures the existence of the ESP protocol header. With this information an algorithm to count Packet Loss was developed.
This Chapter describes the existing target environment in more detail in Section 3.1. Before the designs of the two algorithms are introduced in Section 3.3 the demanded requirements are specified in Section 3.2. Thereupon Section 3.4 describes their implementation. The passive monitoring mechanism was then extended by the design of an alerting system which concludes this Chapter.

3.1 Network Environment and Metrics

This Section starts with an overview about the general concepts of IPSec. With the gained basics Subsections 3.1.1 and 3.1.2 describe the target environment in more detail. This helps to deduce adequate metrics which are summarized in Subsection 3.1.4.
A Virtual Private Network (VPN) is a network which can safely be used within a company or by several different companies or organizations as if it were private, even though some of its communication uses insecure connections. All traffic on VPN connections provides the necessary confidentiality, sender authentication and message integrity by using cryptographic tunneling protocols.
Internet Protocol SECurity, IPSec, provides network-level security for IP. It is not the only technique available for building VPNs, but it is the only method defined by RFCs and supported by many vendors.
IPSec uses two core protocols to provide traffic security by adding a header containing security information to a datagram:

IP Authentication Header (AH) provides *data integrity, data origin authentication and anti-replay service (optional)*

Encapsulation Security Payload (ESP) may provide *confidentiality (by encryption), and limited traffic flow confidentiality, connectionless integrity, data origin authentication and an anti-replay service*.

These protocols may be applied alone or in combination with each other. Both, AH and ESP, support two modes of use: transport and tunnel mode. These two modes describe how AH or ESP processes the IP datagram, e.g. what specific parts of the IP datagram are protected, or how the headers are arranged. They are defined as two types of *Security Associations (SA)*. SAs are the fundamental concept of IPSec to control the granularity at which a security service is offered. The Internet Society defines a SA as follows:

"A Security Association is a simplex "connection" that affords security services to the traffic carried by it. [...] To secure typical, bi-directional communication between two hosts, or between two security gate-

[1] See Section 2.1

ways, two Security Associations (one in each direction) are required. A security association is uniquely identified by a triple consisting of a Security Parameter Index (SPI), an IP Destination Address, and a security protocol (AH or ESP) identifier." (RFC 2401 ([26]))
A transport mode Security Association is a security association between two hosts and provides primarily protection for upper-layer protocols. The security protocol header appears immediately after the IP header (and any options) and before any higher layer protocols like TCP or UDP. A tunnel mode Security Association is applied to an IP tunnel and provides protection to the entire IP packet. To achieve this, there is an outer IP header that specifies the IP tunnel processing destination and an inner IP header with the ultimate destination address for the packet. The entire original, or inner, IP packet is treated as the payload and travels through a tunnel.
Tunnel mode is a common choice for VPN implementations, which are based on tunneling of IP datagrams through an unsecured network such as the Internet. Additionally, VPN requires privacy. This means that datagrams are protected against intermediate devices changing them and against examining their contents as well. The presence of the AH header allows to verify the integrity of a message, but it doesn't encrypt it. Thus, AH provides authentication but not privacy. That's what ESP is for.
The ESP format has several fields that are the same as those used in AH but it packages its fields in a very different way. Instead of having just a header, ESP divides its fields into three components:

ESP Header This contains two fields, the SPI and Sequence Number, and comes before the encrypted data. Its placement depends on whether ESP is used in transport mode or tunnel mode, as explained in the topic on IPSec modes.

ESP Trailer This section is placed after the encrypted data. It contains padding that is used to align the encrypted data, through a Padding and Pad Length field. Further, it also the Next Header field for ESP.

ESP Authentication Data This field contains an Integrity Check Value (ICV), computed in a manner similar to how the AH protocol works, for when ESP's optional authentication feature is used.

Figure 3.1 shows how an ESP packet is processed in tunnel mode.

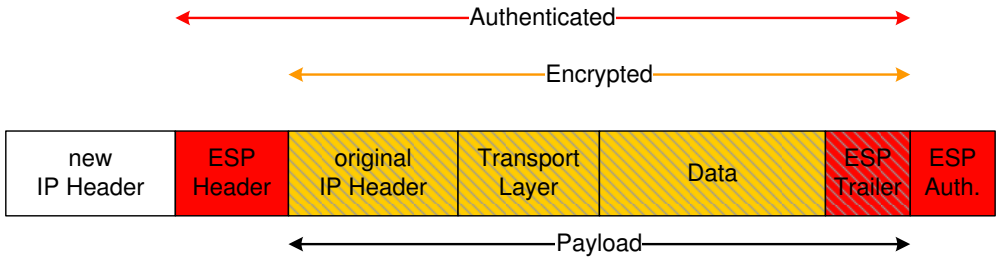

Figure 3.1: ESP in tunnel mode

3.1.1 Open Systems VPN Environment

Open Systems uses the IPSec implementation Linux FreeS/WAN. Three protocols are used:

AH (Authentication Header) provides a packet-level authentication service

ESP (Encapsulating Security Payload) provides encryption plus authentication

IKE (Internet Key Exchange) negotiates connection parameters, including keys, for the other two

The implementation has three main parts:

KLIPS (Kernel IPSec) implements AH, ESP, and packet handling within the kernel

Pluto (an IKE daemon) implements IKE, negotiating connections with other systems

various scripts provide an administrator's interface to the machinery

FreeS/WAN provides both IPSec modes. In Tunnel mode, the gateways provide tunnels. For this KLIPS hooks into the routing code in the LINUX kernel and reroutes the traffic according to the IPSEC extended static routing tables. To bypass this double routing, the VPN gateways of Open Systems use another approach. It connects the gateway in transport mode which is defined as a host-to-host connection. This mode doesn't make use of the IPSEC routing tables and therefore enables dynamic routing in the LINUX kernel just in one step. This advantage of the transport mode is canceled by the drawback that the original IP header isn't encrypted, as shown in Figure 3.2.

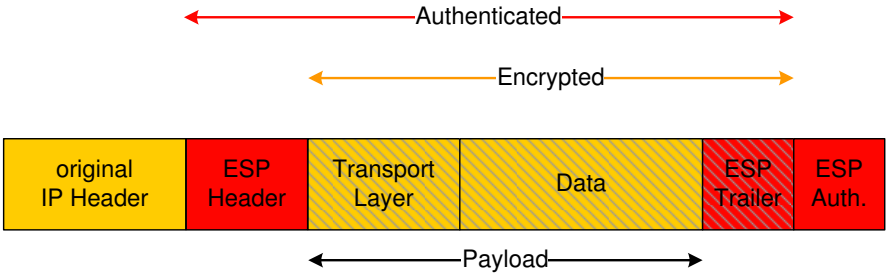

Figure 3.2: ESP in transport mode

This problem is solved by the *Generic Routing Encapsulation (GRE)* protocol. GRE is one of several existing proposals for the encapsulation of one protocol over another one, or in this case, for transporting IP over IP. This is done in two steps. First, a GRE header is added to the payload, i.e. the original IP header and its data. The resulting GRE packet is then encapsulated in an outer protocol, or delivery protocol. This combination of GRE with IPSec's transport mode enables a transport mode with low additional data overhead. Figure 3.3 depicts the new tunnel mode.

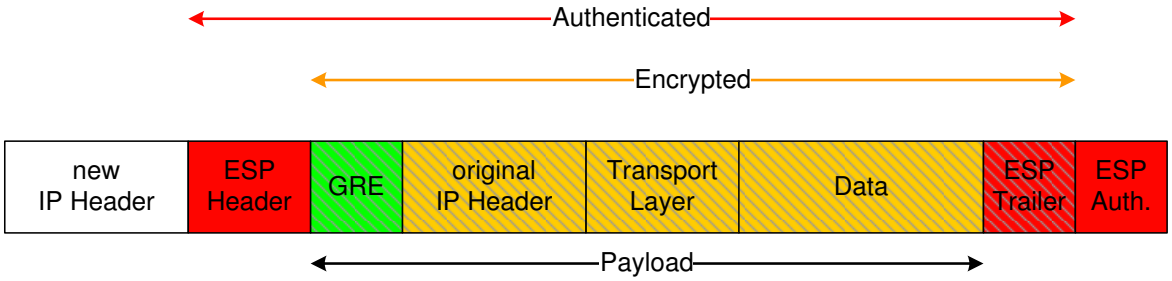

Figure 3.3: GRE and transport mode

3.1.2 Data Packet Capturing and Data Formats

The data packets can be captured at different interfaces showing them at various processing stages.

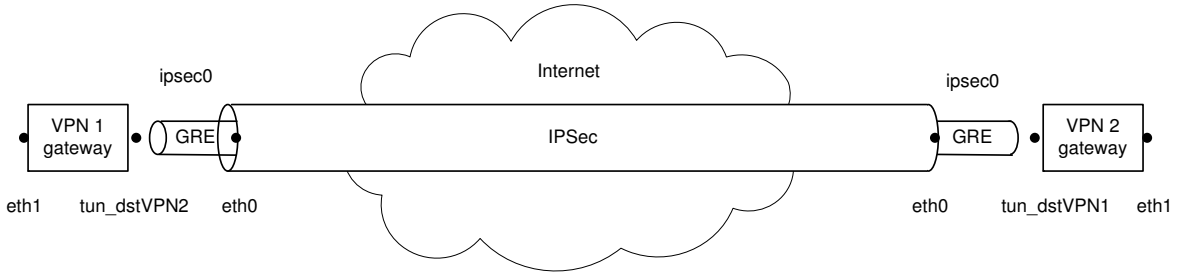

Figure 3.4: VPN Environment Setup

Eth1 represents the interface between the VPN gateway and the customer sites, i.e. the incoming and outgoing packets at this point are the original ones generated from the end-user machines and therefore not processed yet. Packets observed at the interface tun_dstVPN have the similar format as the one at *eth1*.
Then the GRE protocol header is attached to the original one and the resulting GRE packet is encapsulated in a new IP header. This packet format, show in Figure 3.5, is visible at the interface *ipsec0*.

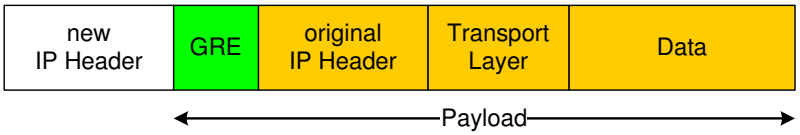

Figure 3.5: GRE packet at ipsec 0

The final processing stage, performed by the IPSec implementation, splits the new IP header and the GRE packet and adds the ESP components according to Figure 3.2. The resulting packet format, shown in Figure 3.3 can be captured at *eth0*.
The two relevant interfaces, *ipsec0 and eth0*, which collect all the available header information, i.e. old IP, GRE, ESP, new IP, and transport layer header are shown in Figure 3.6.

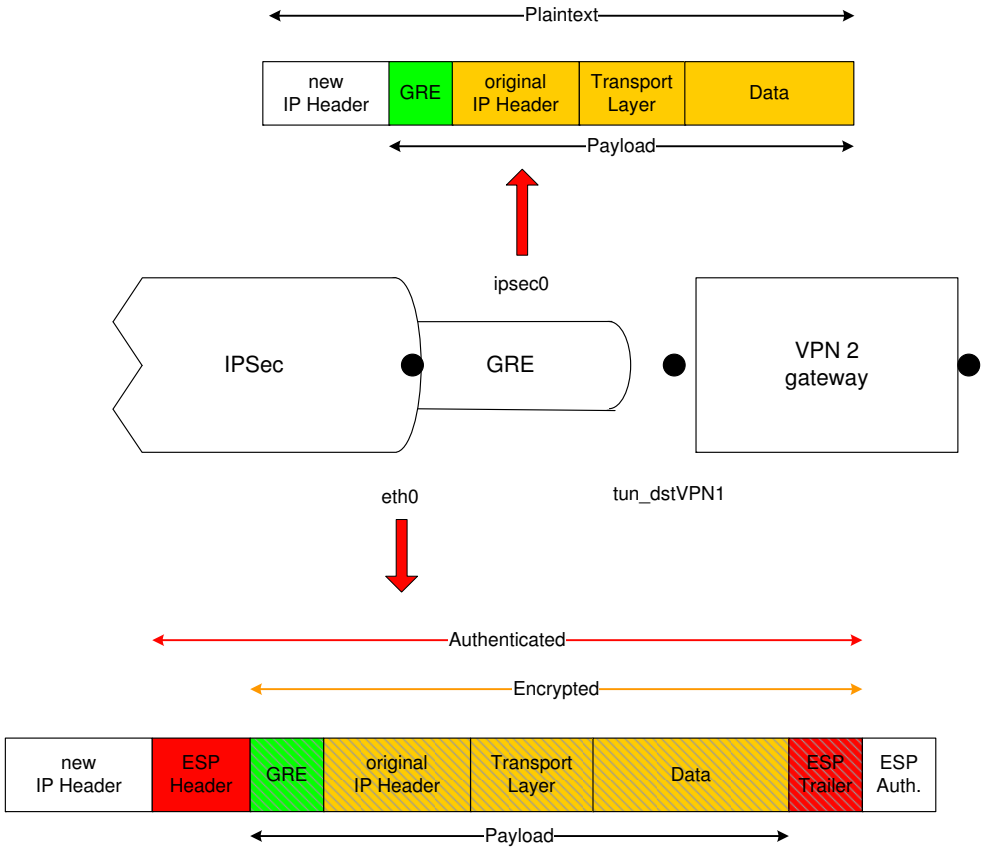

Figure 3.6: Algorithms Relevant Interfaces

The Figure points out that the new IP header is visible at both interfaces. Its payload, including the old IP header, is still visible at interface ipsec0, but encrypted at eth0 and therefore lost for analysis purposes. In exchange the unencrypted ESP header data compensates the information lack.
Nevertheless, data information of both interfaces can be combined by means of the new IP header. But this approach works only for incoming packets by the IP identification matching. Since the ID field of the IP header is set after the encryption process, this header field value is zero at *ipsec0* for all outgoing traffic.

3.1.3 Fragmentation

IPSec's encryption process takes place after interface ipsec0 and before eth0. The new IP header packet length value changes since this procedure enlarges the initial packet size. This is because of IPSec's protocol fields, authentication data and the padding which are all considered as the new IP's payload. The fragmentation process happens after encryption. Thus, if the payload at ipsec0, i.e. before encryption, is greater than 1458 Bytes it will be fragmented because of the Ethernet's Maximal Transfer Unit (MTU).
Figures 3.7 and 3.8 visualize this observation. The MTU is limited to 1500 Bytes. The size for the new IP header, the ESP Header and the Authentication Data doesn't vary. If you subtract this reserved space from the MTU, there are 1444 Bytes left.

12:15:27.026925
IP (tos 0x0, ttl 255, id 62786, offset 0, flags [DF], length: 108) pasv2 > pasv1:
[] IP (tos 0x0, ttl 63, id 7453, offset 0, flags [none], length: 84) 192.168.65.10 > 192.168.5.1:
icmp 64: echo reply seq 14

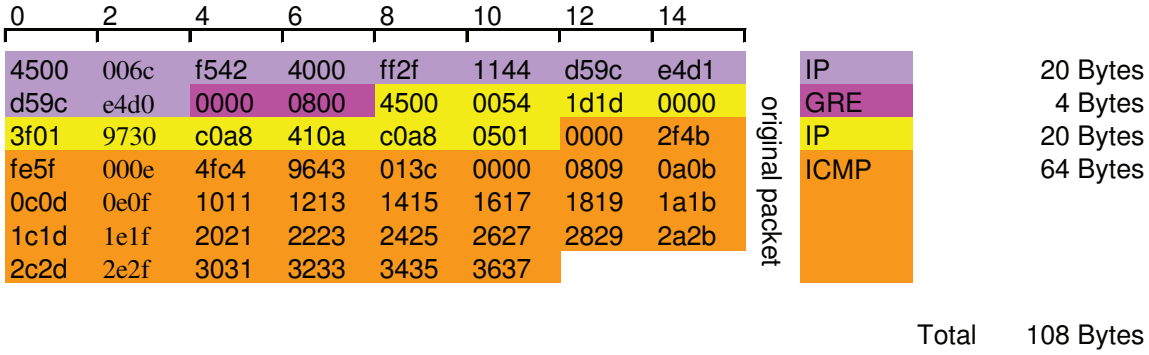

Figure 3.7: Data Packet at ipsec0 (GRE tunnel)

12:15:27.026925
IP (tos 0x0, ttl 255, id 62786, offset 0, flags [DF], length: 152) pasv2 > pasv1:
ESP(spi=0x4a3e6529,seq=0x158f)

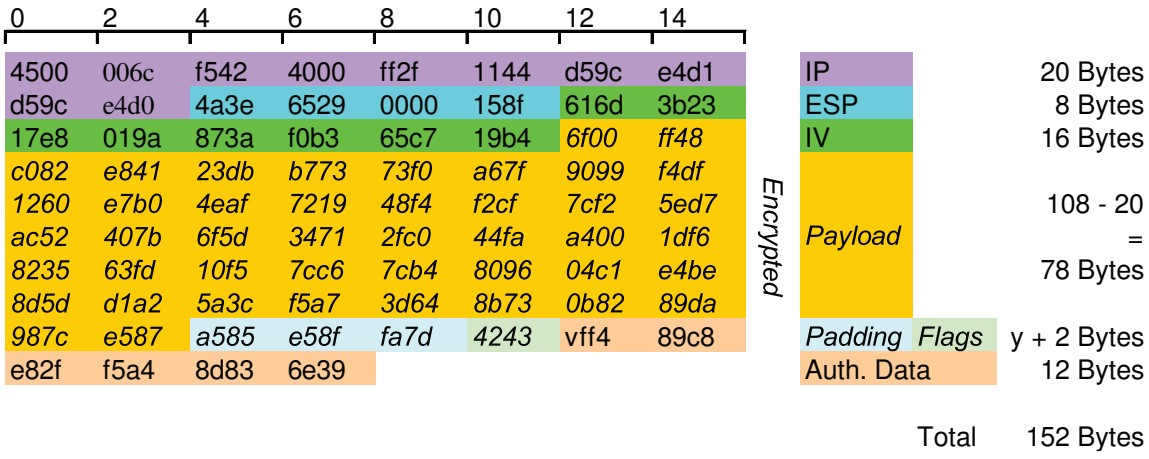

Figure 3.8: Data Packet at eth0 (IPSec tunnel)

The payload of the outer IP header[2] consists of the GRE header, the old IP header and it its payload with a variable size. The IPSec implementation uses AES-128 to produce the cipher text which requires

[2]See 3.5

that the input size is a multiple of 128 bits, or 16 Bytes. But before the AES-128 encryption the ESP Trailer is appended to the payload, which needs at least 2 Bytes for the Flags. To avoid fragmentation, the input size is limited to 1440 Bytes, or a payload of 1438 Bytes.
Figures 3.9 and 3.10 depict the tcpdump of a fragmentation for an incoming packet.

New IP					**Old IP**			
Source IP VPN	ID	offset	lenght	flags	ID	lenght	flags	Source IP
62.2.183.226	45753	0	72	DF	26334	52	DF	192.168.62.62:1414
62.2.183.226	45757	0	**1524**	**none**	26338	1500	DF	192.168.62.62:1414

Figure 3.9: Incoming Packet Information at ipsec0 (GRE tunnel)

New IP					**ESP**	
Source IP VPN	ID	offset	lenght	flags	spi	seq
62.2.183.226	45753	0	72	DF	0x20edca18	0x1d53e4
62.2.183.226	45757	0	**1500**	**+**	0x20edca18	0x1d53e8
62.2.183.226	45757	**1480**	**96**	**none**	sipp-esp	

Figure 3.10: Incoming Packet Information at eth0 (IPSec tunnel)

3.1.4 Metrics

The study of the network environment of Open Systems points out several possible metrics to evaluate the network quality in a totally passive way:

Round Trip Time The RTT is defined as the interval from the time when the VPN records an outgoing packet destined to an end to client to the moment it receives the according acknowledgement. Therefore, only approaches of Round Trip Time measurements are possible.

Retransmissions Retransmission is the number of attempts until a successful delivery of packet. This can be determined on the basis of the packet's sequence number of a certain connection.

The first two can be evaluated on the basis of the TCP protocol header and therefore not all customer traffic is considered. Since most of this traffic uses TCP the importance of this drawback is decreased. The next two metrics can be deduced from the IP and the ESP header information and therefore evaluated for all customer traffic.

Packet Loss Packet Loss is defined as the number of packets launched by the source VPN gateway but never received by the destination VPN gateway.

Fragment Loss If the packet size exceeds the MTU of a connection between two neighbored VPN gateways, it becomes fragmented. Fragment Loss occurs if only fragments and not the whole launched packet gets lost between source and destination.

Packet Loss is more accurate and reliable than Retransmissions. Retransmission can occur in spite of good network quality. Observing retransmissions could therefore lead to false positives.
Fragment Loss is very similar to Packet Loss. In some cases, it provides a little bit more information than Packet Loss. By means of Fragment Loss it is possible to deduce if the perceived problem already comes from the network or if it arises when it arrives at the VPN destination gateway. In the latter case the network quality is good in spite of announced packet loss.

Under certain circumstances these metrics provide too little information. One such case occurs if there is very low customer traffic between two VPN connections. Another occasion is when the monitoring mechanisms perceive bad network quality. This requires an active approach to receive

more accurate information. The former can be solved by actively sending a self-generated TCP packet to continue the RTT measurements. In the latter more information is needed. Two more metrics are possible:

One-Way Delay One-Way delay is defined as the time interval started at the moment when the packet is launched at the source and ended when it is received at the destination point. Therefore, it's possible to determine the direction for which the problem occurs.
To measure One-Way Delay several approaches are possible. One idea is to give a timestamp to already existing customer traffic. Another possibility is to actively generate a new packet which includes a timestamp.

Packet Size Variation Network quality may vary depending on the type of packets sent. Sending packets with variable sizes can help to deduce if either the quality is bad all the time, no matter how big the packet is, or the threshold size when network quality decreases.

3.2 Requirements

Section 3.1 presents the network environment where the monitoring mechanism will be employed. Subsection 3.1.4 summarizes the possible metrics which can be evaluated in this environment. This Section highlights the requirements which the algorithms should fulfill.

No False Positives The algorithm shouldn't generate false positives. False positives would only cost money and time.

Economical Use of System Resources The resource consumption and performance of the monitoring tool shouldn't handicap the gateway in offering its normal service at any time. This means that the algorithm must not influence other programs installed on the gateways.

No Influence to Customer Traffic Customer traffic shouldn't be influenced in any way. Neither a slow down of the traffic speed nor packet drops is allowed.

Scalability The customer network topologies vary very much in size. It could consist of up to 5000 hosts. This leads to big variation in the throughput of different VPN gateways.

Low Maintenance The installation on the VPN gateways shouldn't be time consuming. Further, it should run without much configuration and maintenance.

3.3 Algorithmic Design Aspects

This Subsection introduces the design of the two developed algorithms. They are based on the metrics described in Subsection 3.1.4 and fulfill the requirements defined in Subsection 3.2.

3.3.1 RTT Calculation

Algorithm This algorithm filters out all traffic that uses TCP as transport protocol at the virtual interface on which GRE encapsulated but unencrypted traffic is visible. It uses two characteristics of TCP to measure the RTT. The first is the reliability achieved by performing the Sliding Window Protocol. The second is the fact that each packet can be assigned to an unique connection. For each connection it categorizes its packets in incoming and outgoing traffic.

Outgoing packets For each outgoing packet, the algorithm records a timestamp, its TCP header flags and its expected acknowledgement number of incoming response in the future.

Incoming packets For each incoming packet, the algorithm checks if there is an entry with a matching acknowledgement number. If so, it updates the mean RTT for this connection and deletes the records. Since TCP allows multiple acknowledgements, the algorithm deletes all recorded entries with lower expected acknowledgement numbers as well.

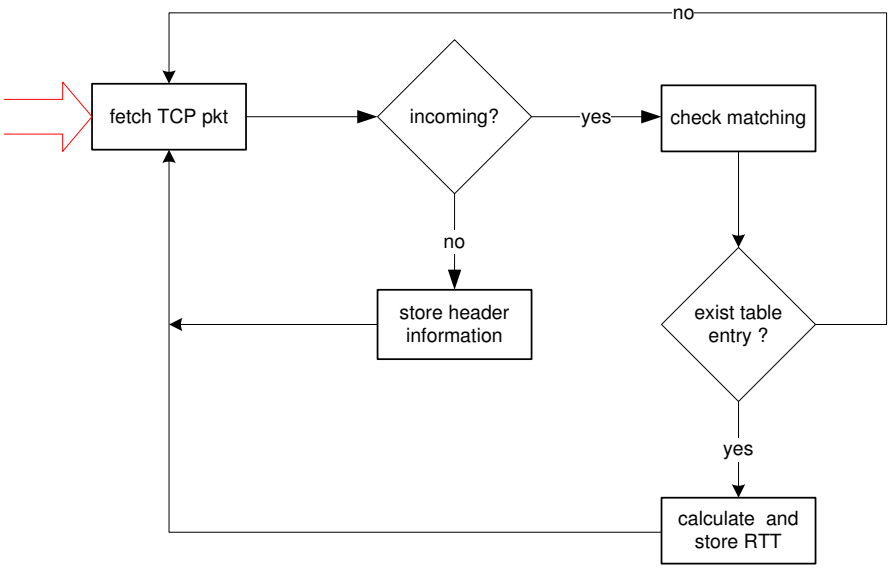

Figure 3.11: TCP RTT Calculation algorithm

Metrics To relieve the network load, TCP's Sliding Window Protocol introduced delayed ACKs which allow a TCP receiver to refrain from sending an ACK for each incoming segment. According to RFC 3465 ([27]) "a receiver should send an ACK for every second full-sized segment that arrives" or when a timeout occurs. The timeout time is not exactly defined. While [20] recommends 200 ms, RFC 1122 ([24]) defines 500 ms. Since only during connection setup, hence when the SYN flag is set, the absence of delayed ACKs is guaranteed, the algorithm differs between three different RTT values:

Overall RTT: The overall RTT is the mean value of all calculated RTTs, neglecting the flags of the outgoing packet.

SYN RTT: The SYN RTT is the mean value of all RTTs for which the SYN flag was set in the outgoing packet.

noSYN RTT: The noSYN RTT is the mean value of all RTTs which are not calculated during a connection setup.

Further, the algorithm saves the minimum RTT and the maximum RTT as well as their related TLL fields.

3.3.2 Packet Loss

The design of this algorithm arised during the evaluation of the target's VPN environment.
Packet Loss detection takes place at the external internet-directed interface. Since IP is connectionless, unreliable and guarantees neither that packets are delivered in order nor those packets are delivered at all, Packet Loss uses additional information offered by the IPSec protocol.

IPSec IPSec's Anti-Replay Service protects end users against so called man-in-the-middle attacks during which an intruder captures the message to subsequently replay it. This is achieved by the Sequence Number field and the Security Parameter Index (SPI) field of the ESP header and the IPSec authentication's requirement of the implementation of a receiver window of size W. The right edge of the receiver window represents the highest Sequence Number N of all packets received yet at the destination. Arriving packets with Sequence Numbers in the range between MIN (N-W+1) and MAX (N) are marked as valid. A packet with a higher Sequence Number, S, than N causes the receiver window to move right until H is at the window's right edge. Packets with lower Sequence Numbers than N-W+1, the value at the window's left edge, are invalid and therefore discarded.

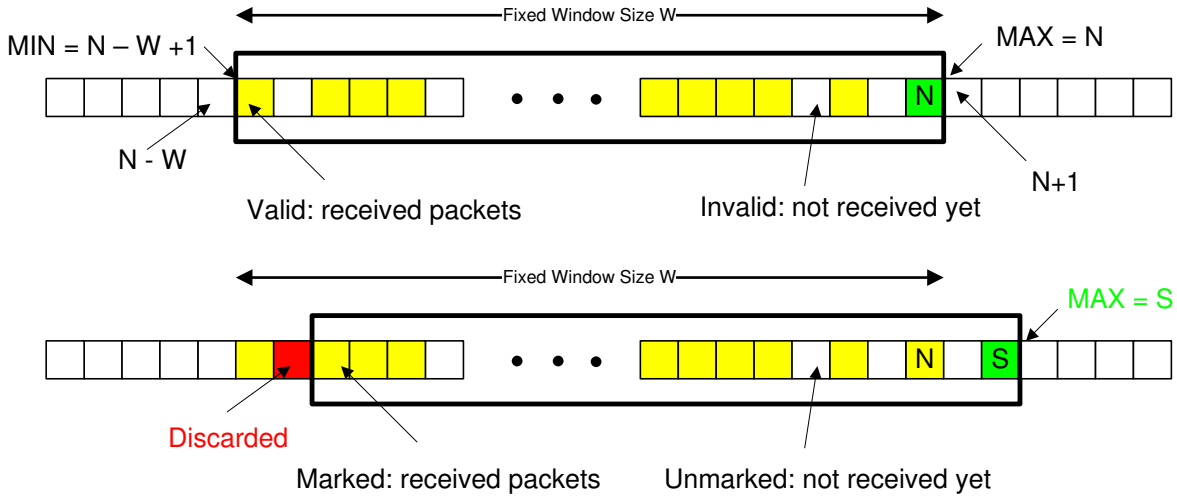

Figure 3.12: IPSec's Anti-Replay Service

Algorithm Packet Loss assigns each incoming packet to a connection, which is uniquely identified by the triple Source VPN, Destination VPN, SPI. For each open connection, it further tracks the receiver window with width W, MIN = N - W + 1 as the left and MAX = N as the right edge value.
For each incoming packet, Packet Loss roughly processes the following steps:

1. Check if it is a packet or a fragment. If it's a fragment, go to step (2) else go to step (3).
2. Store the necessary fragment information and check if all fragments are arrived to reassemble them to the original packet. If so, go to step (3).
3. Check the Sequence Number S of the arrived packet.
   (a) If (S < MIN) then the packet is invalid: increase the number of lost packets.
   (b) If (S > MAX) then count all invalid marked fields in [N-W+1 ... S-W] of the receiver window and add them to the number of lost packets. Move the receiver window S-N steps to the right.
   (c) Otherwise (N-W+1 < S < N) mark the packet as valid in the receiver window.

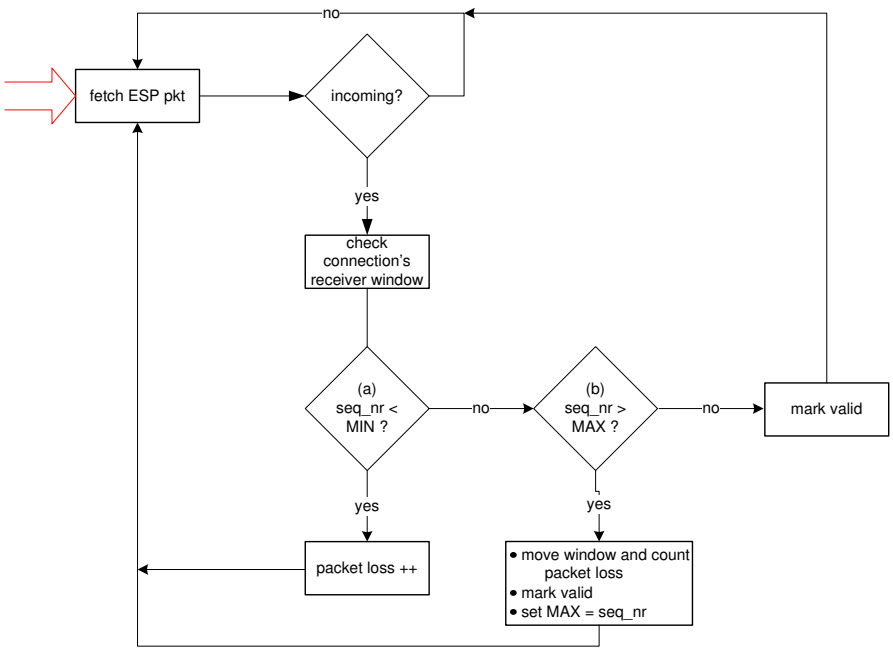

Figure 3.13: ESP Packet Loss algorithm for unfragmented packets

Metrics For each connection, the algorithm counts the number of lost packets and the number of expected packets. They are used to calculate the packet loss ratio. Further, it records the number of fragments and the number of successfully defragmented packets. By comparing them it is possible to roughly classify a perceived packet loss problem in three categories:

Loss of Large Packets: The difference between the number of successful defragmented packets and the number of observed fragments is equal to the number of lost packets. Hence, only packets with size larger than the MTU are lost.

Loss of Small Packets: The number of successful defragmented packets is equal to the number of observed fragments, hence packets of size smaller than the MTU are lost.

Loss of Both: The number of successfully defragmented packets is smaller than the number of observed fragments. Since additional packets are lost is detected, no conclusion about the packet size is possible.

3.4 Implementation

The following Subsections explain the two algorithms in detail. Both algorithms have to be installed on the VPN gateways and are started with the scripts *run_rtt* and *run_packet_loss*. These two scripts call the specific algorithms and pass the required parameters. The two algorithms use Perl's Net::Pcap to monitor the customer traffic passing the gateway. This module uses the standard libpcap code that is part of tcpdump and other packet sniffers. The program codes open up a capturing device, set up defined filters and whenever a packet is received, call the process_pkt routine. This routine uses NetPacket for decoding and encoding of network protocols. NetPacket provides a base class for a cluster of modules, each of them implementing one specific protocol.

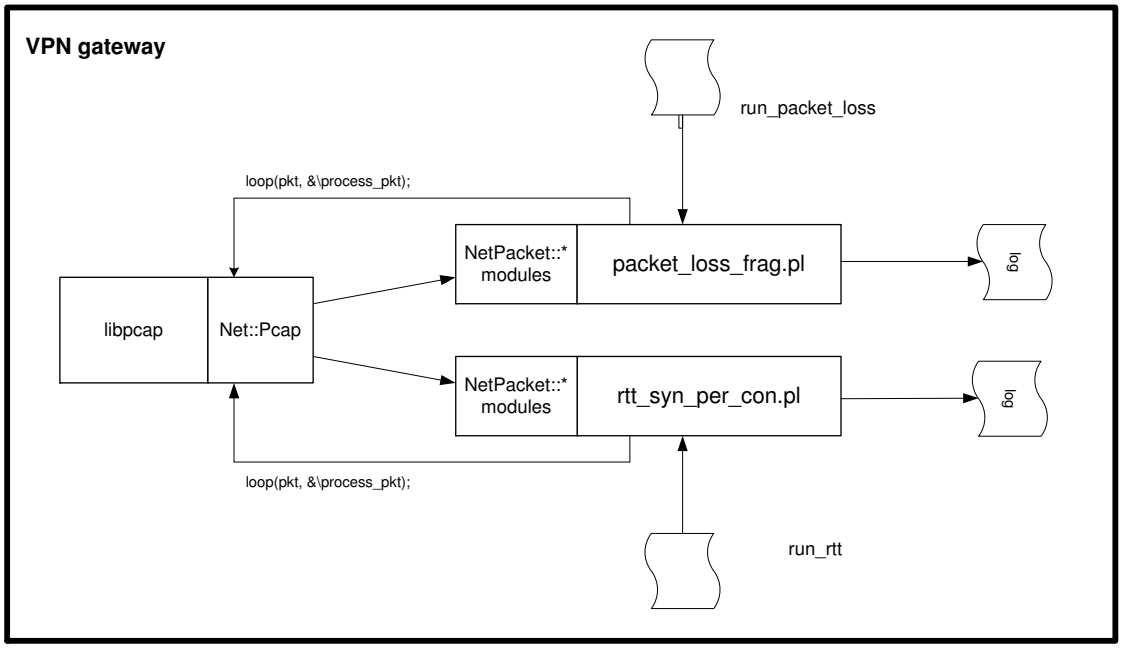

Figure 3.14: Modules and Libraries

3.4.1 RTT Calculation

The script run_rtt calls rtt_syn_per_con.pl and passes following parameters:

foldername The directory for the log files storage.

filename The name of the log files.

timeout_interval The time step in seconds between two log file entries.

interface The interface to observe.

SYN_threshold The threshold value to achive that only the SYN RTT is calculated.

The algorithm observes *ipsec0* interface to measure the RTT *per VPN connection*. But this RTT is not the same as Open System's active probing RTT. The passively calculated RTT is the time gap for the path *VPN src -> VPN dst -> client -> VPN dst -> VPN src*. This means that this RTT value is higher by the time interval *VPN dst -> client -> VPN dst*. After the defined time step *timeout_interval* the algorithm determines which value is assigned to the RTT. If the number of observed connection setups achieve *SYS_threshold* then the the mean value of all observed SYN RTTs per connection is logged to the file *"foldername/filename"*. Otherwise, it takes the value of all calculated RTTs. The format of the a log file entry looks like the following:

entry nr	VPN connection	mean RTT	min RTT	max RTT	total pkts	timestamp	min TTL	max TTL	standard deviation	standard error	Mode [ALL = 0; SYN = 1]
0	194.158.252.154:212.98.54.20	88097.5789473684	16962	221868	19	1143106422	127	127	1636.89599236959	397.005592628828	0
1	194.158.252.154:212.98.54.20	44948.0555555556	16971	111741	18	1143106705	127	127	2293.96787101223	556.368931409217	0
...											

NetPacket Modules The script rtt_syn_per_con.pl uses the information of three different protocol headers:

- Outer IP Header
- Inner IP Header
- TCP Header

To get the desired protocol information it uses the IP and TCP modules of NetPacket. Since the GRE protocol header is placed in between the Outer and Inner IP headers but no GRE module exists the algorithm decodes this information by itself.

Data Structure and Garbage Collection The algorithm makes use of three hash tables which can be accessed by a combination of TCP and IP protocol header information. They are listed in table 3.1.
The first table is a hash of hashes and it is used to match the outgoing packet with it's reply. Its outer hash uses the IP's client source and destination addresses as well as TCP's source and destination PORT numbers to create its key dst. The key of the inner hash, seq_entry, is the expected acknowledgment number of the reply packet.
The garbage collection for this hash is started every time when a match occurs.
The structure of the other two tables is similar. They store the following metrics per VPN connection, one for all RTTs and one for SYN RTTs:

- mean RTT
- maximum RTT
- minimum RTT
- number of observed RTTs
- maximum TTL
- minimum TTL
- standard deviation
- standard error
- timestamp

The entries of these two hashes are deleted every time interval when the metrics are written to the log file.

rtt {dst} {seq_entry}	=	[timestamp (sec), timestamp (ms), ACK nr, TCP flags]
rtt_all {vpn_con}	=	[mean RTT, min RTT, max RTT, total pkts, time_interval, min TTL, max TTL, standard deviation, standard error]
rtt_syn {vpn_con}	=	[mean RTT, min RTT, max RTT, total pkts, time_interval, min TTL, max TTL, standard deviation, standard error]

Table 3.1: Data Structure of rtt_syn_per_con.pl

Memory Usage The algorithm stores the outgoing packet's expected acknowledgement number and flags and timestamps this information. Thus, this requires a memory allocation of 16 bytes for each packet entering the VPN tunnel.
Additional 72 bytes are needed per VPN connection to save the calculated metrics.

3.4.2 Packet Loss

The script run_packet_loss calls packet_loss_frag.pl and passes following parameters:

foldername The directory for the log files storage.

filename The name of the log files.

timeout_interval The time step in seconds between two log file entries.

interface The interface to observe.

VPN_gateway The IP address of the VPN gateway

window_size The size of IPSec's receiver window. The usual size is 64.

The algorithm sets up a filter before it starts Net::Pcap to sniff packets passing the *eth0* interface. Due to this filter only packets are with he destination IP address *VPN_gateway* are considered. For each VPN connection it initialized a receiver window of size *window_size* and applies the rules described in 3.3.2. After each *timeout_interval* it writes the observed metrics in the log file *"foldername/filename"* which has the following format:

entry nr	srcIP:dstIP:SPI	nr of lost_pkts	nr of delayed_pkts	nr of pkts	nr of fragments	nr of suc_fragments	nr of late_frag	data_size	timestamp
1	213.173.179.58:212.203.89.115	0	0	3089	170	170	0	818432	1140594914
2	213.173.179.58:212.203.89.115	0	0	2838	32	32	0	441648	1140594974
...									
50	213.173.179.58:212.203.89.115	13	0	1211	13	13	0	197812	1140597854
...									

NetPacket Packets passing the *eth0* interface are already encrypted. Hence only the outer IP and the newly attached ESP protocol headers are readable. To access the IP header information NetPacket's IP module is used. There's no module to obtain the values of the ESP protocol fields and therefore extracted by the algorithm itself.

Data Structure and Garbage Collection The algorithm uses three hash tables to detect and store packet loss. The gateway implements for each VPN connection a receiver window. They are stored in the first hash and can be accessed through the triple Source IP, Destination IP, SPI. The second hash stores the calculated metrics per VPN connection. Its key is defined as the concatenation of Source IP and Destination IP addresses. Since the number of neighbored VPN gateways and therefore the number of VPN connections is constant, no garbage collection is needed for the first two hashes. To handle fragmented packets more information is needed. These particulars are hold in the third hash and accessible through the IP header fields "Source IP:Destination IP:Identification". Garbage collection takes place in two steps. The common way is to delete the entries when the packet is successfully defragmented. But this kind of memory deallocation doesn't work in case of fragment loss. Therefore, an additional garbage collector periodically removes such old entries.

seq_windows {src:dst:spi}	=	[Min, ..., Min + WindowSize = Max]
seq_statistics {src:dst}	=	[nr of pkts, nr of fragments, nr of lost_pkts, nr of delayed_pkts, nr of suc_fragments, nr of late_frag, data_size]
frag_properties {src:dst:id}	=	[src:dst:spi, ESP sequence_nr]

Table 3.2: Data Structure of packet_loss_frag.pl

The hash *seq_statistics* stores all the metrics that are written to the log file after the user-defined timeout_interval:

Packet Count Number of expected packets

Packet Loss Number of lost packets

Fragment Count Number of observed fragments

Successful Fragments Number of successfully defragmented packets

The hash further stores information about the number of delayed packets. These are packets which arrived too late at the destination gateway and therefore are discarded.

Memory Usage The algorithm stores the values either per Security Association[3] (SA), per connection or per fragmented packet:

SA: The VPN gateway has to implement a *receiver window* per SA. The window consists of 64 sequence numbers which results in 256 Bytes during the existence of the SA. Further, the algorithm provides a garbage collector for old SAs that needs additional 12 bytes per SA.

Connection: The metrics are stored per connection and require 28 bytes of space.

Fragment: Fragment handling requires additional information. Therefore, additional 16 bytes are temporarily allocated per fragmented packet until it is ready for defragmentation.

3.5 Approach to Alerting

Section 3.3 introduces the concepts of the two algorithms which are explained in Section 3.4 in more detail. After their implementation first evaluations of their reported metrics are done and documented in Chapter 4. This Section presents the approaches for an alerting system which were based on the observed results.

3.5.1 RTT Calculation Alerting

Case Study The Case Study of Table 3.3 on page 49 helps to predict the network performance and provides a basis for an alerting system.
The column attribute Possible Events in Table 3.3 lists the whole range of sources for metrics changes. Changes of either minimum RTT (MIN) or average RTT (AVG) falls in one of three categories:

Traffic Pattern The traffic pattern is influenced by the number of connections, number of connection setups, duration of connections, data packet sizes and type of application traffic. These parameters always vary and can cause changes in the measured metrics.

Routing Path The routing in a tunnel between two VPN gateways can change. There may be some routers added or dropped to a path which results in changes of the measured RTT. Rerouting can happen because of congestion in a link between two routers and can lead to longer paths and therefore longer RTTs. Although the new route has no congestion Routing Path Changes can influence the network quality.

Network Performance Congestion happens if a router is overloaded and a Routing Path Change is not possible. This leads to packet drops and TCP retransmissions. Congestion influences the network quality.

False Positives This classification outlines that only the *routing path* and *network performance* affects the network quality and therefore highlights the importance to identify the exact category. Thus, if the diagnosis is *attention* or *warning* further investigations are needed. Otherwise, the case of Traffic Pattern Change would alert a false positive. The idea of the analysis is to check the existence of OneHop RTTs: Their existence asserts the exclusion of this category. But this analysis requires active probing[4]. A simple active approach to distinguish between OneHop RTT and MultiHop RTT is to send a ping to the neighbor VPN gateway and measure the time delay to its reply. If this interval is equal to the Min RTT we can conclude network performance degradation with high probability.
Another proposal is an Inline Technique. Its idea is to modify the IP's TTL field: The sender sets TTL to 255 when the packet enters the VPN environment. The receiver then only considers RTT of packets with TTL value 254. Choosing this approach leads to the complete exclusion of the traffic pattern category. Then, no active probing is required.

[3] See the definition in 3.1
[4] See 2.5.2 for existing active probing techniques.

	MIN	AVG	Possible Event(s)	Diagnosis
1	↔	↓	**Traffic Pattern**: less MultiHop RTTs "Good" **Routing Path** Change after 1st hop **Network Performance**: No Congestion any more	IGNORE
2	↔	↔	no event (=> packet loss?)	
3	↓	↓	**Traffic Pattern**: no OneHop RTT before "Good" **Routing Path** Change in 1st hop **Network Performance**: No Congestion any more	
4	↓	↔	**Traffic Pattern**: no OneHop RTT before "Good" **Routing Path** Change in 1st hop **Network Performance**: No Congestion any more	
5	↔	↑	**Traffic Pattern**: more MultiHop RTTs "Bad" **Routing Path** Change after 1st hop **Network Performance**: Congestion	ATTENTION
6	↓	↑	**Traffic Pattern**:more MultiHop RTTs "Good" **Routing Path** Change after 1st hop **Network Performance**: Congestion	
7	↑	↔	**Traffic Pattern**: no OneHop RTTs attention "Bad" **Routing Path** Change in 1st hop **Network Performance**: Congestion in 1st hop	
8	↑	↑	**Traffic Pattern**: no OneHop RTTs anymore "Bad" **Routing Path** Change in 1st hop **Network Performance**: Congestion in 1st hop	
9	↑	↓	**Traffic Pattern**: only OneHop traffic and **Network Performance**: Congestion in 1st hop Network Performance: Congestion after 1st hop	WARNING

Table 3.3: RTT Case Study

3.5.2 Packet Loss Alerting

The Packet Loss algorithm measures the metrics. Its output provides the basis for an alerting system. Figure 3.15 on page 51 shows its state machine.

Algorithm The algorithm is an iterative process. It consists of the following steps:

1. Check Packet Count:
   If PC == 0 then send a ping and go to step 2
   else go to step 3
2. Check Packet Count:
   If PC == 0 then announce total packet loss and go to step 1
   else go to step 3
3. Check Packet Loss:
   If PL == 0 then announce connection ok and go to step 1
   else go to step 4
4. Check Fragment Count:
   If FC == 0 then send a ping and go to step 5
   else go to step 6
5. Check Fragment Count:
   If FC == 0 then announce packet loss with fragment loss and go to step 7
   else go to step 6
6. Compare Fragment Count with Succeeded Fragments:
   If FC == SC then announce packet loss but no fragment loss and go to step 7
   else packet loss with fragment loss and go to step 7
7. Calculate the ratio and compare to the threshold c:
   ratio = PL / PC
   If ratio > c then announce poor connection and go to step 1
   else announce connection ok and go to step 1

Explanation The states, and actions respectively, last one interval. Hence, a round of this algorithm lasts two to seven intervals. Some steps, or actions, require further explanation:

Step 1: As long as no packets arrive, the VPN gateway can't predict the link quality. Thus, if no traffic is observed, it pings its neighbor. With this action it assures that PC should not be zero at the next time interval.

Step 2: If the VPN gateway received a reply from its neighbor then packet count is greater than zero. Therefore the device can be sure that traffic with small packets is possible. Since there's the possibility of lost packet with bigger size, it still has to check the packet loss variable.

Step 4: The absence of fragments doesn't imply the absence of large data packet transfers. Therefore the gateway has to test if large packets can pass the tunnel. Pinging its neighbor with a large fragmented packet should result in a fragmented reply packet. Therefore the fragment count variable shouldn't be equal to zero at the next interval.

Differencing between packet loss with fragment loss and packet loss without fragment loss gives a first indication of the lost packet's data size. In case of poor connection, further analysis concerning this issue can be done manually by a security engineer at the mission control.

3.6 Conclusion

This Chapter introduced the target environment in detail and deduced its available metrics. After it listed the requirements which this Thesis's monitoring mechanism has to fulfill it describes the design and implementation of the developed approach. The designs of corresponding warning systems concludes Chapter 3.
The following Chapter reports the results of the tests run with the implemented mechanism.

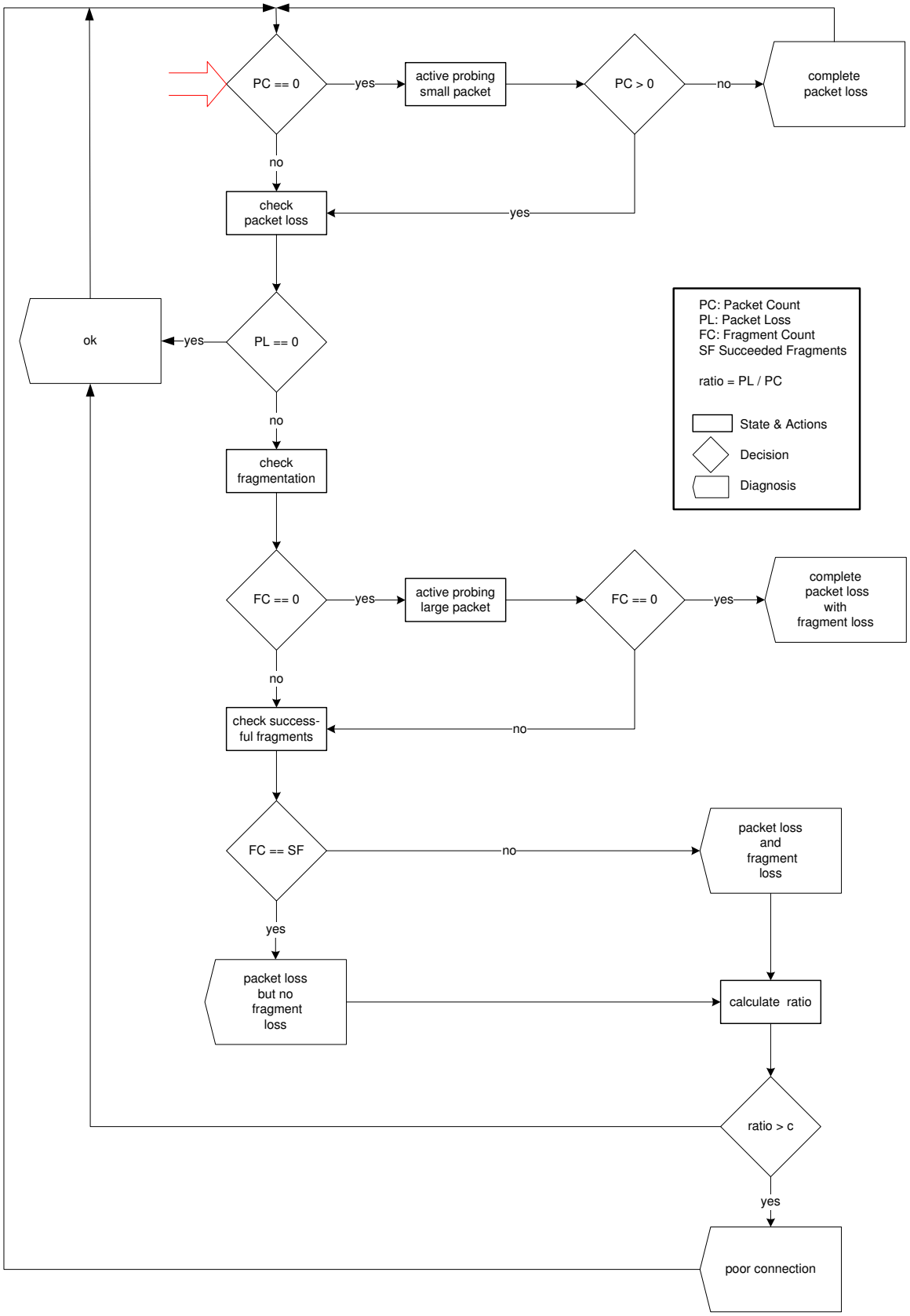

Figure 3.15: State Machine of Packet Loss Warn System

Chapter 4

Evaluation

Sections 3.3 and 3.4 presented the design and implementation of the two developed algorithms. This Chapter summarizes the evaluation of the reported metrics in Section 4.1. For this evaluation the implemented prototype was tested under different aspects and and many plots were done to visualize the results. Based on these evaluations the RTT Calculation algorithm was optimized. The improvements are documented in Section 4.2. The final Section addresses the comparison of Open System's active ICMP Probing with the passive RTT Calculation algorithm.

4.1 Evaluation

To test the monitoring algorithm, realtime traffic was recorded on different customer networks and at different time periods. Afterwards this traffic was rerun with tcpreplay on a test bench. This solution guaranteed on the one hand that the monitoring mechanism didn't slow down the VPN gateway's performance which would have influenced the network performance. On the other hand the replay of this traffic didn't falsify the gained results. However, the logged timestamps correspond to the network time taken at the replay time. That's why the scales of the Figures' timeline are minutes.

4.1.1 RTT Calculation

First evaluations of realtime traffic showed high discrepancies between the RTT of two neighbored VPN gateways A and B. Figures 4.1 and 4.2 show three different types of RTT measurements. While the Overall RTT uses all kind of traffic to calculate the metric, SYN RTT takes only packets into account which have the TCP SYN flag set. noSYN RTT is the opposite of SYN RTT.

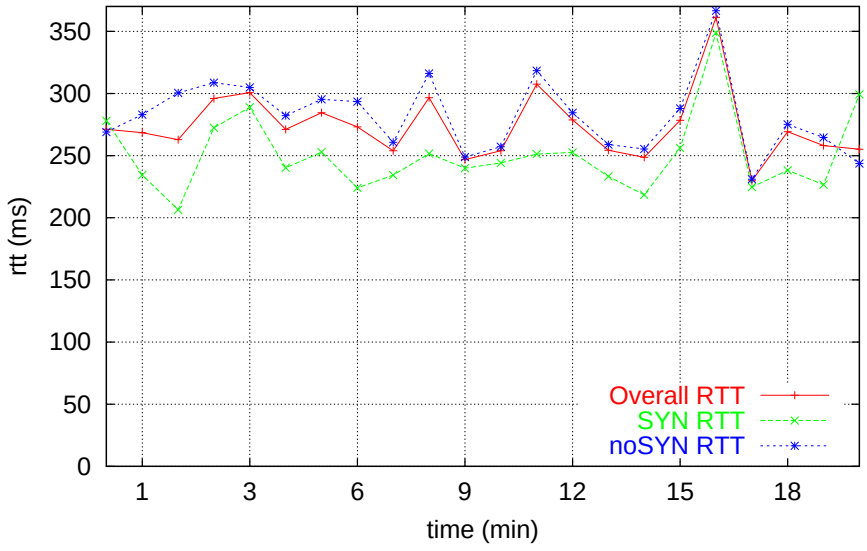

Figure 4.1: RTT observation at VPN gateway A

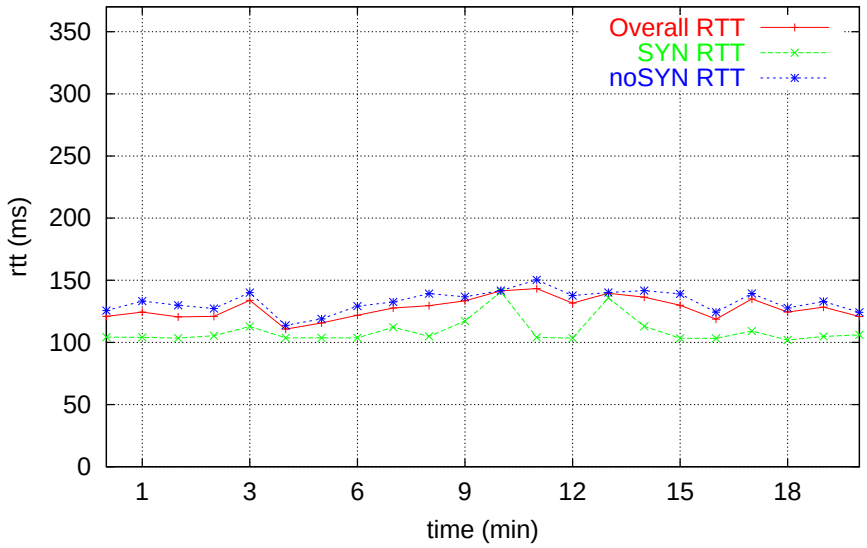

Figure 4.2: RTT observation at VPN gateway B

Thereupon a histogram for graphical summary of the RTT distribution was done at each VPN gateway:

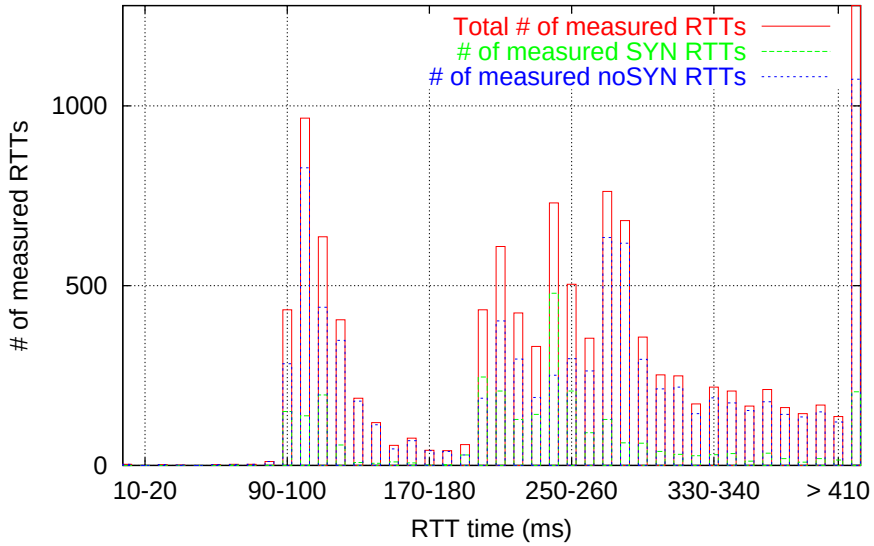

Figure 4.3: Histogram at VPN gateway A

Compared to Figure 4.4, the RTT distribution of Figure 4.3 has two peaks indicating that the RTT calculations depend on the network topology. While the gateway B measures RTTs of traffic between two direct neighbored sites the VPN A additionally observes RTTs of traffic passing multiple gateways.

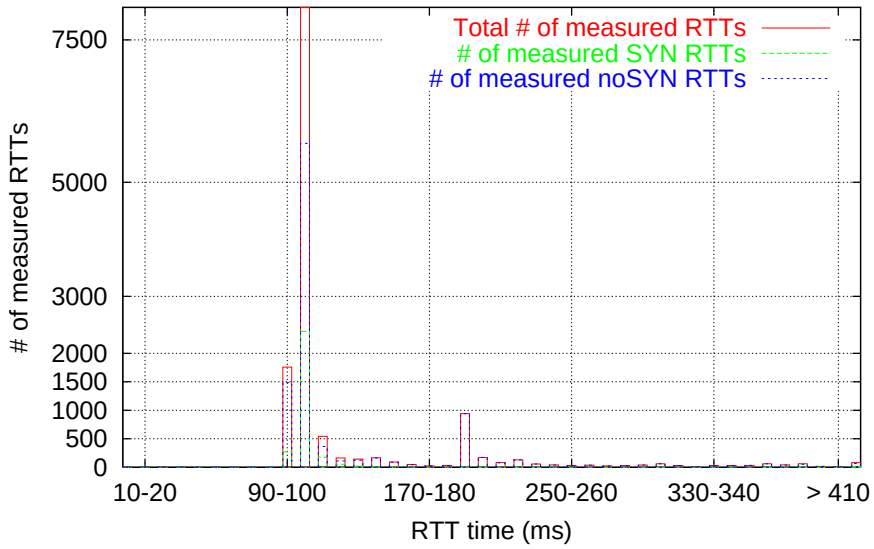

Figure 4.4: Histogram at VPN gateway B

The analysis of the known network topology confirms the network topology assumption showed in Figure 4.5.

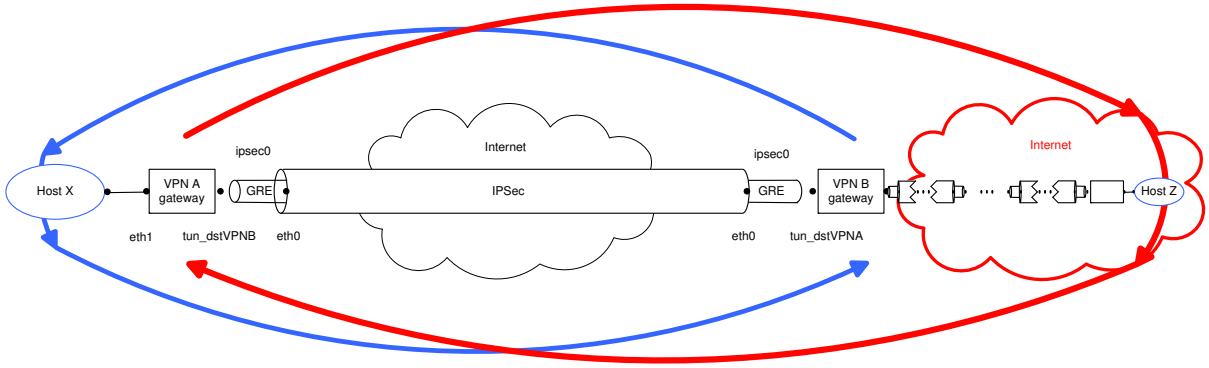

Figure 4.5: Network Topology Assumption

The main point of interest is the quality of the observed RTTs. Thus, subsequent to the network topology deductions, the three metrics of Figures 4.1 and 4.2 were compared. They show what was stated in Subsection 3.3.1: The RTTs of connection setups need less time than the other ones. Therefore SYN RTTs give more valuable estimations under consideration of two issues. First, by measuring SYN RTTs only small packets are considered. And second, SYN RTTs deliver more accurate estimations as long as the number of observed SYN RTTs is large enough. But Figures 4.6 and 4.7 show that a guaranteed amount of observed SYN RTTs is not available all the time.

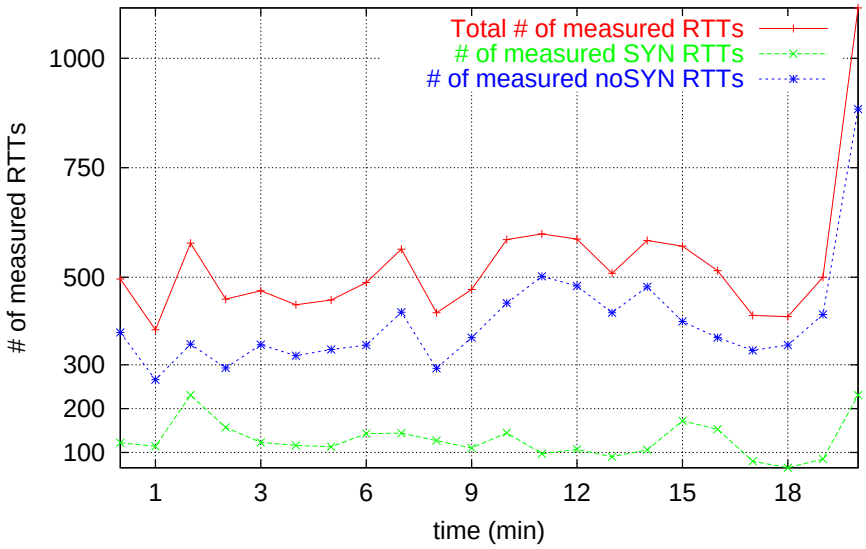

Figure 4.6: Counted RTT observations at VPN gateway A

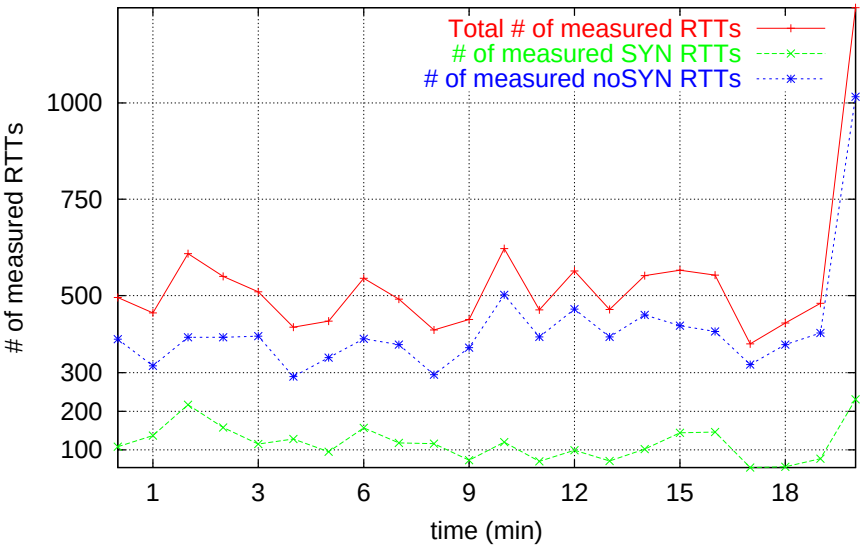

Figure 4.7: Counted RTT observations at VPN gateway B

These observations were used for further optimization[1].

[1] See Subsection 4.2.1

4.1.2 Packet Loss

First runs of Packet Loss analyzing network traffic showed that the customer's subjective perception agreed with the implementation's evaluation. During a voice over IP connection the customer complained about bad quality as soon as the number of lost packets exceeded the value of 60 packets.

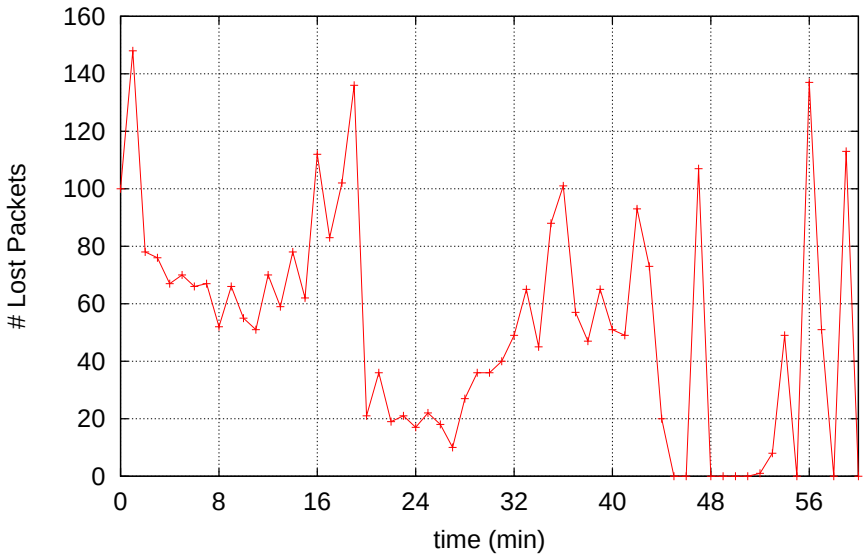

Figure 4.8: Packet Loss

Thereupon more traffic of VPN gateways was recorded and analyzed with the monitoring mechanism and their logs validate the observation of the first runs: the analysis of the reported metrics agreed with the customer's feedback at all times. Figure 4.9 shows a small extract of a customer network. The customer complained about bad link quality from VPN gateway C to VPN gateway B. The analyzed data of Packet Loss detected three bottleneck links out of the six connections.

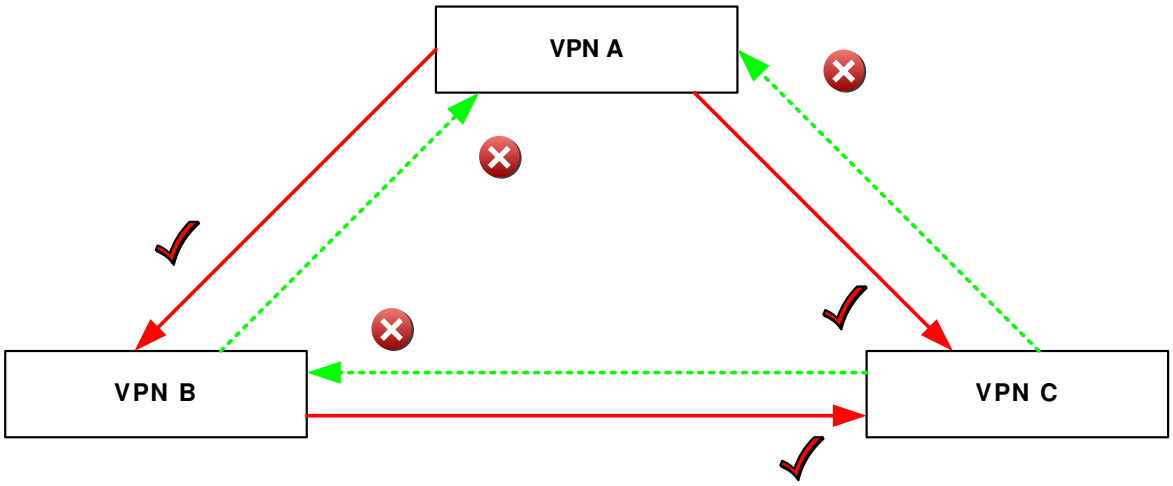

Figure 4.9: Extracted Setup of a Customer Network

The packet loss ratios of the connections are graphically visualized in figure 4.10 - 4.12.

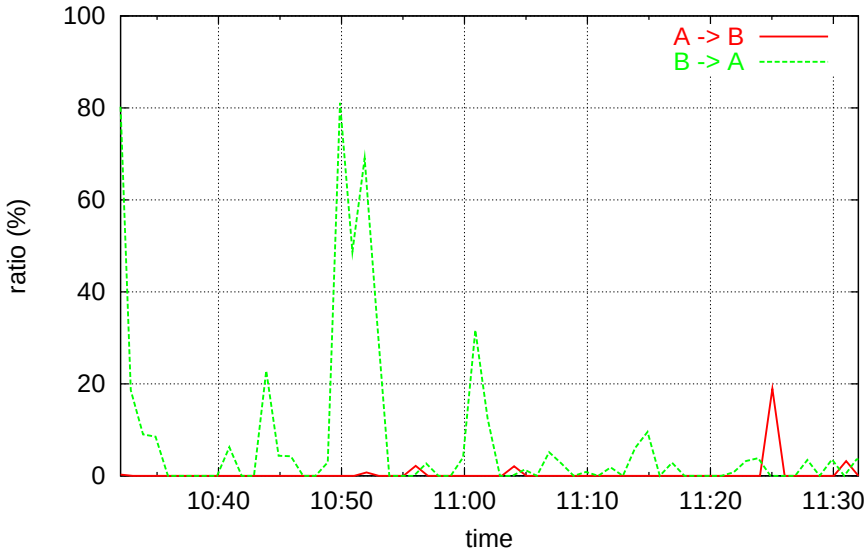

Figure 4.10: Packet Loss Ratios between the VPN gateways A and B

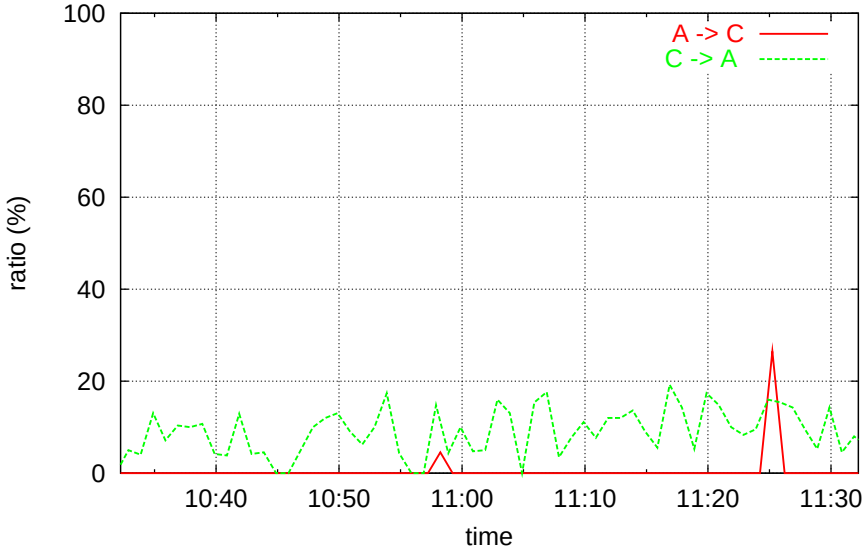

Figure 4.11: Packet Loss Ratios between the VPN gateways A and C

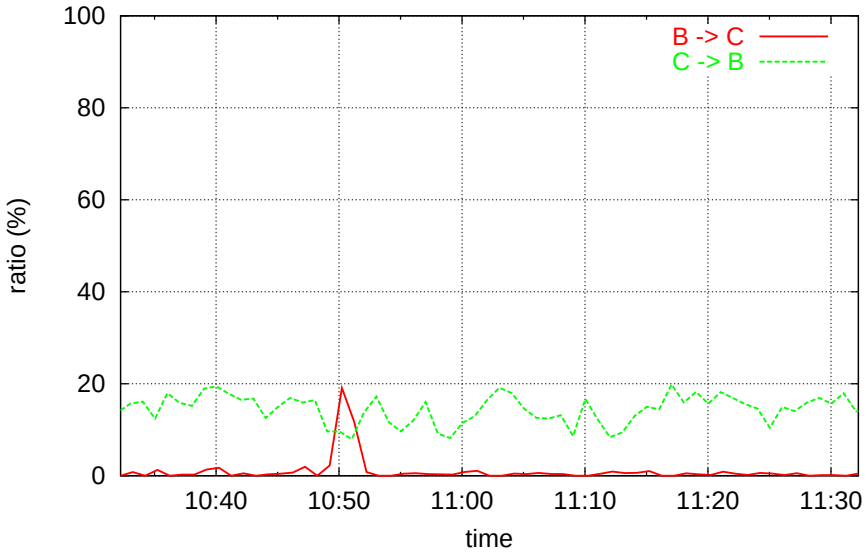

Figure 4.12: Packet Loss Ratios between the VPN gateways B and C

The positive results of the evaluations causes the decision to install the mechanism on several VPN gateways for further runs. They confirmed the results of the previous tests.

The evaluation done so far asserted that the algorithms are applicable in the target environment. Since Open Systems calculates the RTT metric too the RTT Calculation algorithm can be furhter tested against its accuracy. Thus a comparison of Open System's active ICMP Probing with the passive RTT Calculation algorithm completes the overall evaluations in Section 4.3.

4.2 Optimization

4.2.1 RTT Calculation

The focus for the optimization was set to the choice of the most accurate RTT metric and its deducible reports about the network behavior. To predict the link quality between two gateways, the *average RTT* is compared to the *minimum RTT*.

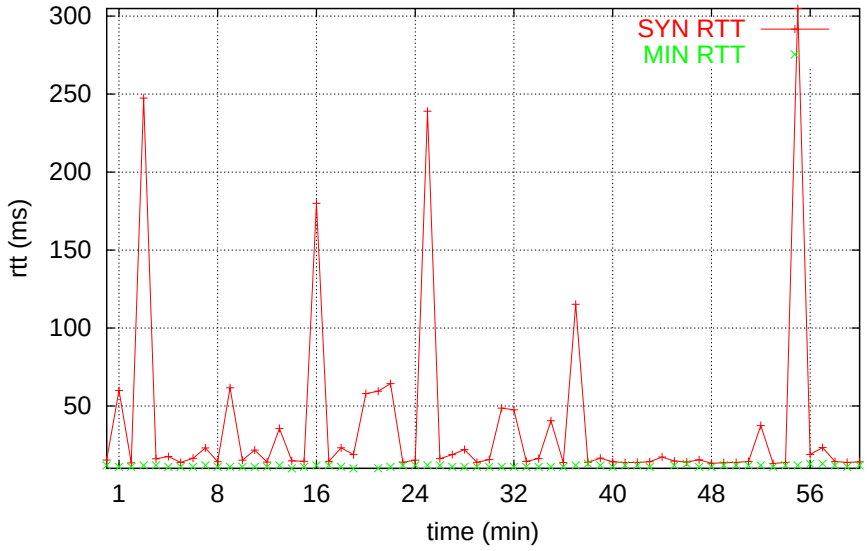

Figure 4.13: Comparison of the SYN RTT and MIN RTT at VPN gateway C

Figure 4.13 depicts the comparison of the *SYN RTT* to the *MIN RTT*. Figure 4.14 shows the relation between the *Overall RTT* and the MIN RTT. Both were simultaneously monitored at a VPN gateway of Open Systems.

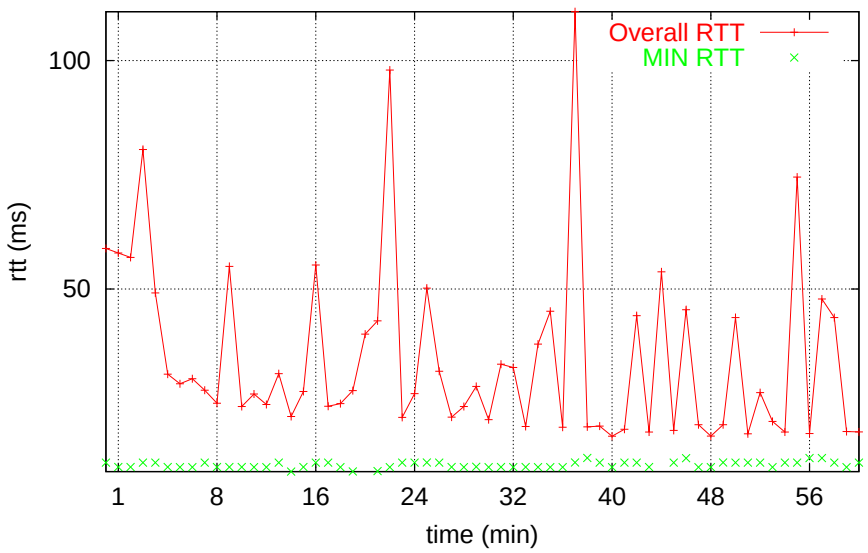

Figure 4.14: Comparison of the Overall RTT and MIN RTT at VPN gateway C

Both Figures lead to the same conclusion. While the MIN RTT shows a constant value, the average as well as the SYN RTT balances in a band of fixed width. The topology evaluation of Subsection 4.1.1 helps to explain this phenomenon and two definitions are introduced.

OneHop RTT: The measured RTT consists of two packets exchanged between neighboring VPN gateways.

MultiHop RTT: The packets used for measuring the RTT covers a distance with multiple VPN gateways in between.

Figures 4.15 and 4.16 visualize the definitions.

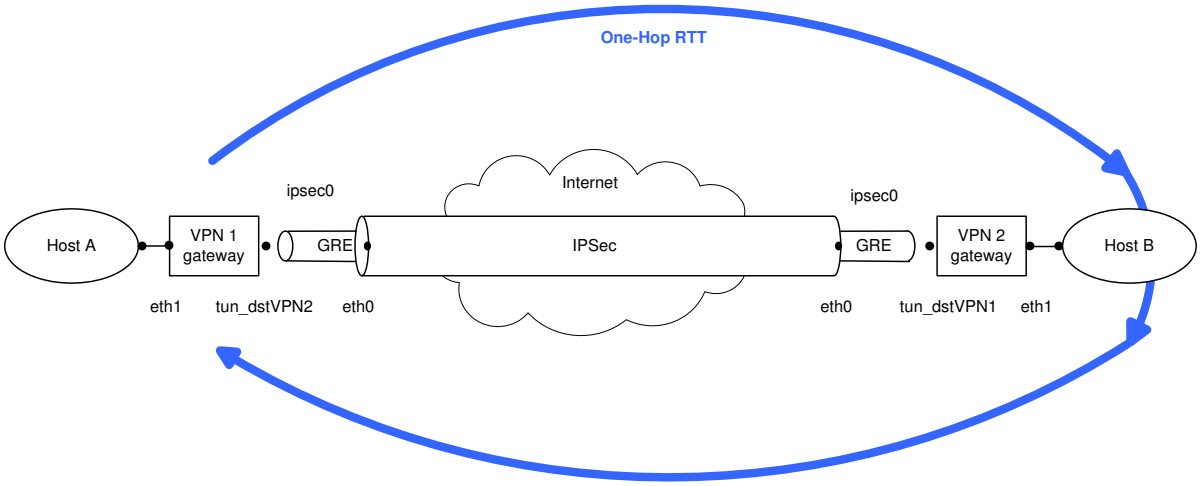

Figure 4.15: OneHop RTT

Since the MIN RTT is usually a OneHop RTT, its value is constant. But the average RTT, or SYN RTT respectively, depends on the traffic pattern during a measure interval: The ratio between the number of OneHop RTTs and MultiHop RTTs varies according to the customer traffic.
Comparing Figure 4.14 with Figure 4.13 brings up another observation: The oscillation band of the

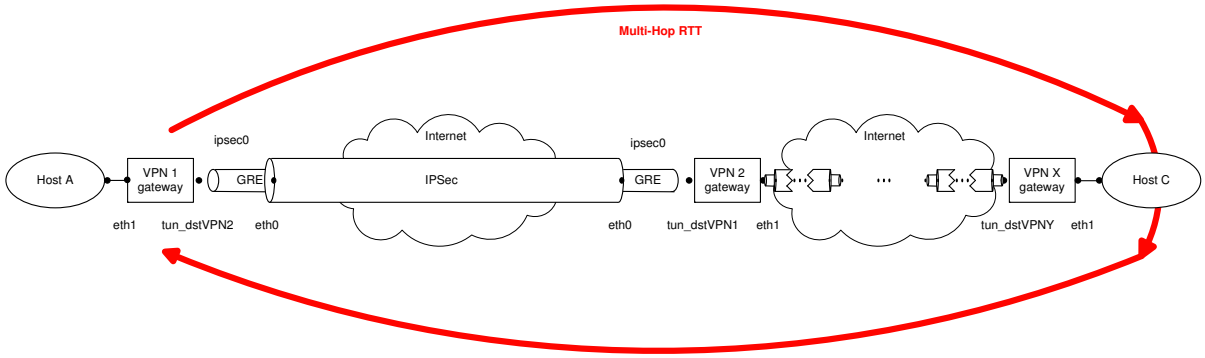

Figure 4.16: MultiHop RTT

Overall RTT is smaller than the SYN RTT's one what's the result of statistical uncertainties. [22] calls uncertainties statistical "if they arise not from a lack of precision in the measuring instruments but from overall statistical fluctuations in the collections of finite numbers of counts over finite intervals of time". Figure 4.17 which compares the number of observed SYN RTTs to the number of counted Overall RTTs outlines these uncertainties.

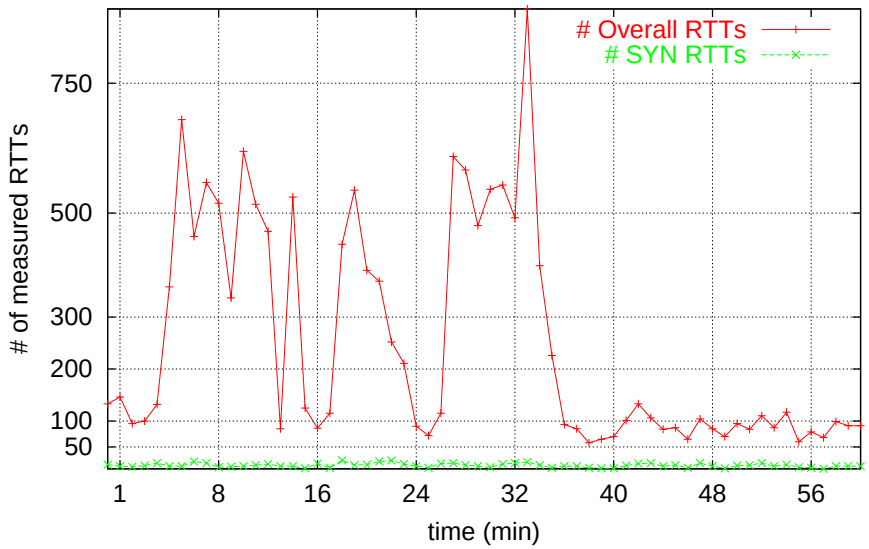

Figure 4.17: Number of observed SYN RTTs and Overall RTTs at VPN gateway C

The more RTT calculations per interval are measured the less weighted is a singular outlined calculation and hence the smoother are these oscillations. This, in turn, is desirable for the estimation quality of the algorithm and agrees with the latter observations of the previous Subsection.

Algorithm Improvement Since the RTT calculation algorithm of Subsection 3.3.1 is used for the analysis of the network behavior, it is optimized according to the results of the evaluation. It is restructured into two modes: *OBSERVE_SYN* and *OBSERVE_ALL*.

> In the OBSERVE_SYN mode, only SYN RTTs are monitored, as long as its observed amount larger than a threshold value. The threshold value is a parameter set at configuration time. If the number of observed SYN RTTs falls below this value, it switches in the OBSERVE_ALL mode. Then, all RTTs are observed until there is a constant appearance of SYN RTTs again.

The algorithm now announces exactly two metrics: *MIN RTT* and *AVG RTT*. But their quality depends on the number and type of measures taken during an interval. Thus, the metrics should be assigned to a worthiness value and be kept in mind for any further network predictions.

4.3 RTT Comparison: active ICMP Probing vs. passive RTT Calculation

The RTT Calculation algorithm measures nearly the same metric as Open Systems. While the active ICMP probing only measures the RTT of the VPN tunnel between two neighbored gateways the RTT metric logged by the passive approach adds to the tunnel RTT the delays of the remote customer site.

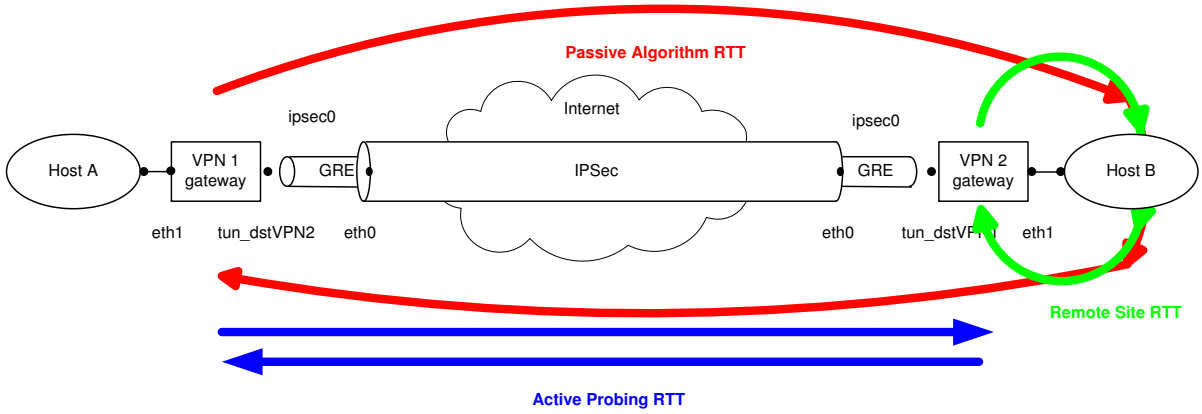

Figure 4.18: Measured RTTs

To test the accuracy and reliability of the passive metric in a direct comparison with the active one the algorithm was rewritten after the optimization. The modified algorithm still measures exactly the same RTT metric as the original one. It was extended by the additional metric Remote Site RTT. This metric corresponds to the additional part of the measured delay in the passive approach.

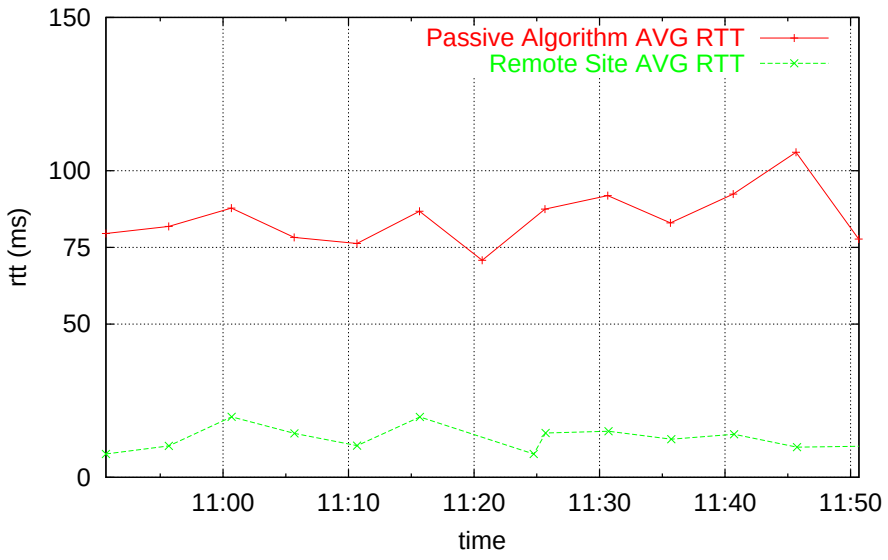

Figure 4.19: Remote Site and Passive Algorithm AVG RTTs

Figure 4.21 shows the direct comparison of the two algorithms. The Tunnel RTTs are the MIN and AVG RTTs of the passive algorithm after the subtraction of the Remote Site RTTs.
The active probing RTT is the mean of ten ICMP pings periodically done during five minutes. The plotted extracts of the evaluations highlight that such small packets can be delayed even though there are not many late arrivals in the customer traffic.

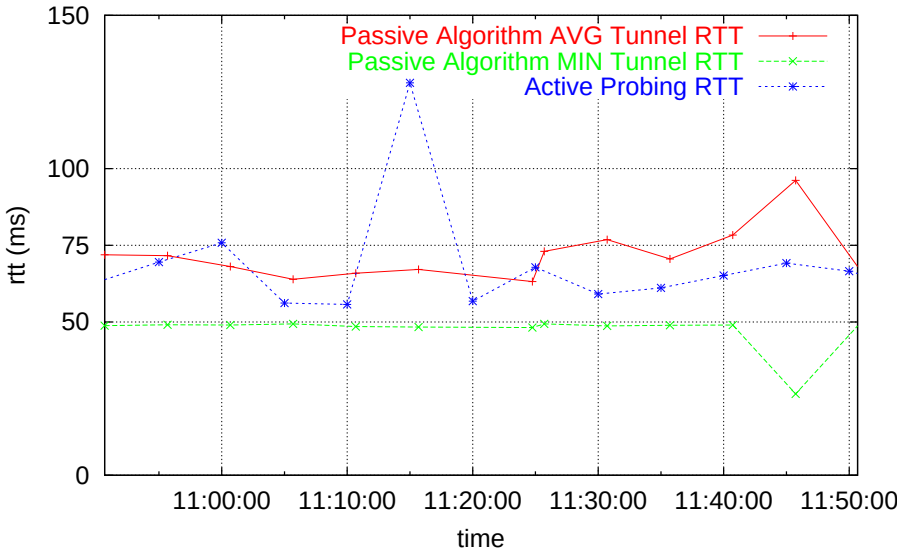

Figure 4.20: Comparison of active ICMP RTT and passive Tunnel RTT

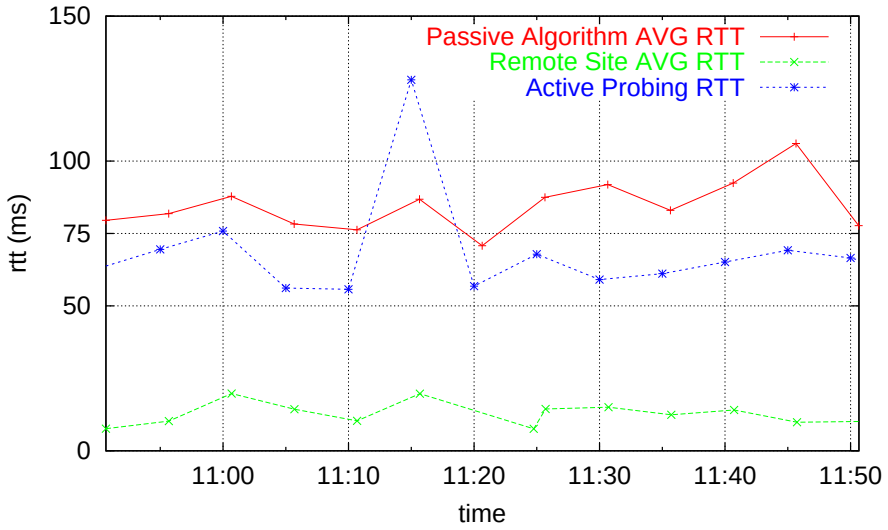

Figure 4.21: Remote Site, Passive and Active RTTs

4.4 Conclusion

The evaluations done show that the passive approach delivers a good approximation to the active one. But the original algorithm doesn't include the Remote Site RTT. Therefore it is also possible that it announces a high RTT value although there are no network problems inside the VPN environment.
This Chapter evaluated the two implemented algorithms under different perspectives. First analysis of the RTT Calculation algorithm pointed out some unexpected network behaviors. Thereupon the algorithm was optimized. This optimization is described in Section 4.2.1. All test done with the Packet Loss algorithm were very encouraging. The announced metrics always agreed with the reported network performance perception of Open Systems' customers.
The evaluation is completed by a comparison of the passive RTT Calculation algorithm with the active ICMP probing done by Open Systems. The positive results are documented in Section 4.3.

Chapter 5

Summary

5.1 Conclusion

The task to implement a passive monitoring mechanism is fulfilled. I designed two algorithms that calculate performance metrics. The tests in the Open System VPN environment have shown promising results. Open Systems plans to employ them for their network performance monitoring.
My prototype is implemented in Perl due to safety reasons. Perl's automatic memory management limits the risk of memory failures and is therefore preferred to C. The monitoring mechanism runs very stably and the reported metrics agree with the current network situation. The memory allocation of both RTT Calculation and Packet Loss is low and both memory and processing usage achieve the resource constraints.
Even though the passive monitoring system satisfies its requirements there are several issues which could be improved:

- The CPU utilization is dependent on the amount of traffic passing the VPN gateway. Since both algorithms need to capture all packets, the introduction of the prototype on VPN gateways with heavy traffic should be handled with care. Tests done showed that the utilization grows up to 10% on a 2.4 GHz intel processor at a traffic rate of 2 Mbit/sec. This is not acceptable since Open Systems runs up to five services on a gateway and these services shouldn't be handicapped in any case. Another point is that Open Systems plans to enlarge its service portfolio. This means that more services are installed on a VPN gateway require more processing resources.
- RTT Calculation as well as Packet Loss don't implement an alerting system. The current versions passively monitor the crossing traffic, calculate the metrics and log them. But the values of these metrics can differ between particular VPN gateways. Therefore, experience values must be individually evaluated at each gateway. The security engineer has then to decide if the performance metrics are higher than the threshold value and therefore network problems should be alarmed. An approach of such an alerting system is documented in Subsection 3.5.1.

Finally, I want to point out that the prototype is not the only and eternal solution. It is one solution among a large amount of existing passive monitoring mechanisms. There are many research groups that have been working on several monitoring projects for decades and they will carry on also in the future. Further, most of them combine passive with active approaches to achieve more and better results. Since the internet expansion won't stop in the near future, the complexity of the paths and the diversity of the performance will still increase. Further, new protocols and traffic types will appear. Therefore, there will be many future projects trying to optimize existing solutions.

5.2 Outlook

This section provides an outlook to describe further research topics and points out where improvements on the prototype can be done.

- The most preferred improvement is to speed up the existing prototype. The mechanism is currently implemented in Perl. Rewriting the algorithms in C would decrease the CPU utilization and therefore relieve the VPN gateways.

- Another open issue is that the prototype doesn't contain a warning or management system. However, the Subsections 3.5.1 and 3.5.2 describe a specification of such a system for the algorithms. The models incorporate active probing if required. The implementation of these specifications could be done in the future.
- The RTT Calculation algorithm includes measurements of both OneHop and MultiHop RTTs. Two approaches to measure OneHop only were proposed in 4.2.1. Particularly the suggested Inline Technique is worth to follow up. This technique doesn't generate additional traffic and hence doesn't fill up the customer's connection link.
- The RTT algorithm could be extended by the number of retransmissions. Since the rise of this metric doesn't imply bad network performance it should be regarded as complementary information to the RTT metric.
- The last paragraph of Subsection 3.1.2 addresses to the possibility to monitor the incoming packets at different interfaces and combine the different protocol information. It is worth noting, that the outer IP's identification field offers the possibility to relate the metrics collected on different interfaces.

There are more promising improvements which are beyond the scope of this Master Thesis though. The following points could be integrated in future projects.

- The first approach is once again an Inline Technique. As explained in 3.1.1, Open Systems uses the GRE protocol[1] for its packet processing. This protocol is extensible with optional header fields. One of them is the key field, which can be set by the encapsulator and interpreted by the receiver. The idea is that the sender assigns this field with a timestamp. Since the VPN gateways are synchronized via NTP, the receiver can calculate the OWD. The existing passive mechanism IMI uses OWD too. As already mentioned in its evaluation in Section 2.1.5, OWD is more significant than the RTT since it incorporates the direction of the time delay.
- Active Probing techniques can help in case of network performance problems. For example, packet size evaluation can be done by sending ICMP ECHO packets of different sizes through the troubled tunnel. With this technique it may be possible to filter out the packet size ranges which are disabled.
- To do customer traffic analysis as a supplementary helps in advanced network and service management. First, statistics about service and protocol types can highlight trends which support Open Systems in planning their future like Service Level Agreement (SLA) negotiation and policing. Second, the amount of traffic, the number of connection setups and the duration of a connection in relation to the time can help to improve the Quality of Service and to respect the SLAs.
- Statistical analysis provides information for ISP Topology analysis. It can filter out connections which show performance problems over and over again. In case any bottleneck ISPs are identified, the network can be optimized by setting up redundant VPN connections to this ISP.

Two additional propositions concern the area of Routing Topology analysis:

- The distribution of the calculated metrics among the VPN gateways would allow to locate network problems. For example, if the application traffic between two customer sites traverses multiple VPN gateways and a site complains about a bad network quality, the shared information would outline where the bottleneck link is located on the path.
- If the TTL field is manipulated according to the Inline Technique proposed in Subsection 4.2.1 it is possible to realize routing path length changes. For each hop between source and destination the TTL field decreases by one. To detect routing patch changes the destination VPN gateway stores for each MultiHop connection the TTL field it and compare this value with the TLL field of a new incoming packet for this connection. If the new TTL field value is lower than the stored one the length of the path increased. Otherwise, if it is higher a "good" routing path change took place.

[1] See RFC 1701 ([25])

Appendix A

Acknowledgments

I would like to thank all the people who supported me during this Master Thesis:

First of all, I thank Roel Vandewall for all his time he gave to me, the proofreadings and the many helpful discussions including critical and constructive statements. I also want to thank Stefan Lampart very much who was responsible for the organization of this thesis. The project plan could be maintained without any problems due to their support.

Special thanks for the support of Arno Wagner and Placi Flury from the ETH. Arno helped with his useful feedbacks that this thesis took a good way. I'm also grateful for the consent of Professor Bernhard Plattner which made this external thesis actually possible.

Finally, I would like to thank the whole Open Systems team. I spent a wonderful time here at Open Systems. The people here were very kind and helped me whenever needed.

Appendix B

CD-ROM Contents

This appendix describes the content on the compact disc which comes with this thesis.

- **Report:** A pdf- and a ps-version of the documentation. Additionally, all raw latex files and pictures of the report.
- **Final Presentation:** The slides of my final presentation at Open Systems AG and the ETH Zürich.
- **Task Description:** The formulation of my task as pdf- and ps-file.
- **Passive Monitoring Mechanism:** The monitoring algorithms as well as the scripts to configure and start them.
- **GNUPlot Scripts:** Perl scripts to visualized the metrics of the log files.

Appendix C

Scripts Configuration

The passive measurement mechanism consists of two algorithms: rtt_syn_per_con.pl for RTT Calculation and packet_loss_frag.pl for Packet Loss. The scripts *run_rtt* and *run_packet_loss* manage the configuration details.

RTT Calculation

For the configuration the parameters are passed in the following order:

1. foldername
2. filename
3. timeout_interval
4. interface
5. SYN_threshold

run_rtt starts rtt_syn_per_con.pl and writes two output files:

foldername/ALLfilename A short statistic about all VPN connections together.

foldername/filename Logged metrics per timeout_interval and per VPN connection.

Packet Loss

For the configuration the parameters are passed in the following order:

1. foldername
2. filename
3. timeout_interval
4. window_size
5. VPN_gateway
6. interface

run_packet_loss starts packet_loss_frag.pl and writes two output files:

foldername/ALLfilename A short statistic about all VPN connections together.

foldername/filename Logged metrics per timeout_interval and per VPN connection.

The mechanism has been tested on hosts running Open Systems Linux build OSIX 1.10.5 based on Linux kernel version 2.4.31 with libpcap 0.8.3. The algorithms are implemented with Perl version 5.8.3 and use the modules Net-Pcap 0.10 and NetPacket 0.04.

Appendix D

Schedule

This schedule has been planned at the beginning of this Thesis.

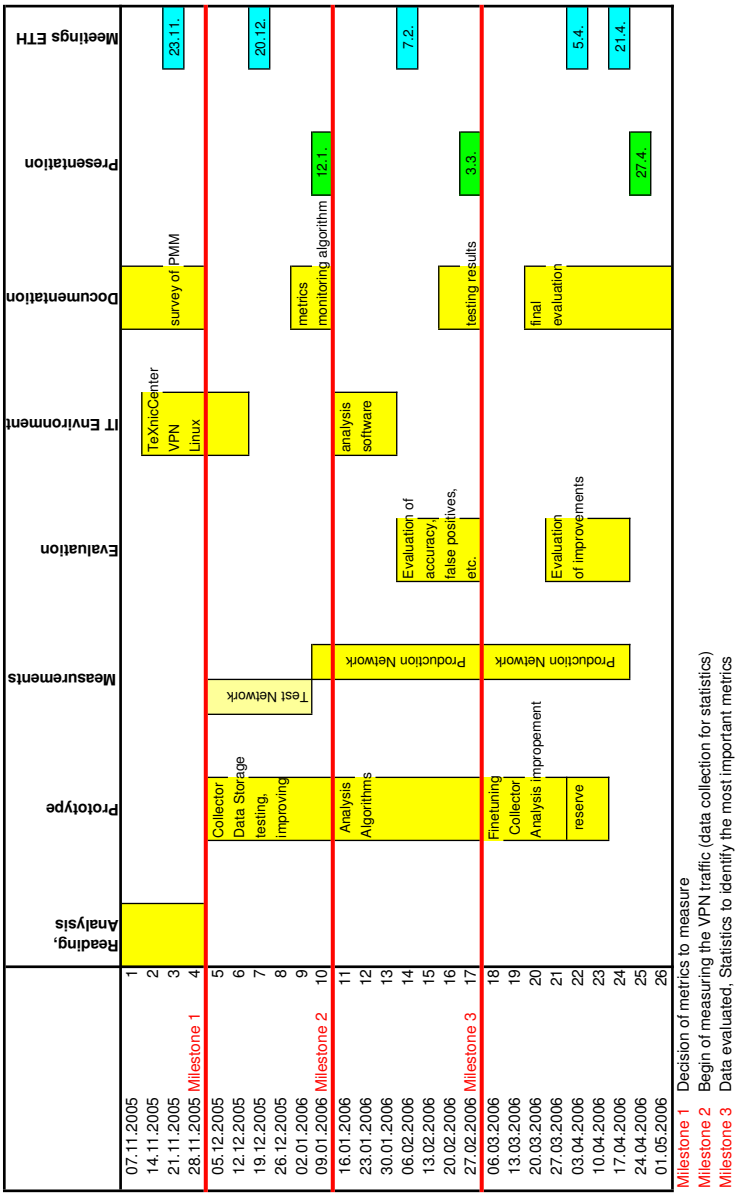

Figure D.1: Schedule

Bibliography

[1] *argus - auditing network activity*
http://www.qosient.com/argus/, 17.11.2005.

[2] *cisco - NetFlow*
http://www.cisco.com/en/US/tech/tk812/tsd_technology_support_protocol_home.html, 20.4.2006.

[3] Cristian Estan, Ken Keys, David Moore, George Varghese.
Building a Better NetFlow
http://www.cs.wisc.edu/ estan/publications/betternetflow.pdf, SIGCOMM 2004, Portland Oregon, USA, 2004.

[4] Dimitrios P.Pezaros, David Hutchison, Francisco J. Garcia, Robert D. Gardner, Joseph S. Sventek.
Service Quality Measurements for IPv6 Inter-networks
https://129.132.99.171/http/0/ieeexplore.ieee.org/iel5/9152/29055/01309369.pdf?tp=&arnumber=1309369&isnumber=29055, Twelfth IEEE International Workshop on Quality of Service (IWQOS), 2004.

[5] *Internet End-to-end Performance Monitoring*
http://www-iepm.slac.stanford.edu/, 21.03.2006.

[6] *Internet Protocol Suite - Wikipedia, the free encyclopedia*
http://en.wikipedia.org/wiki/Internet_protocol_suite, Wikipedia - The Free Encyclopedia, 27.02.2006.

[7] Les Cottrell, Matt Zekauskas, Henk Uijterwaal, Tony McGregor.
Comparison of some Internet Active End-to-end Performance Measurement projects
http://www.slac.stanford.edu/comp/net/wan-mon/iepm-cf.html, Stanford Linear Accelerator Center (SLAC), Menlo Park, CA, USA, 1999.

[8] Luca Deri.
nProbe: an Open Source NetFlow Probe for Gigabit Networks
http://luca.ntop.org/nProbe.pdf Proceedings of Terena TNC, Zagreb, 2003.

[9] Luca Deri, Stefano Suin.
Ntop: beyond Ping and Traceroute
http://luca.ntop.org/, Proceedings of the 10th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM), Zürich, Switzerland, 1999.

[10] Luca Deri, Stefano Suin.
Practical Network Security: Experiences with ntop
http://www.terena.nl/conferences/archive/tnc2000/proceedings/3A/3a1.pdf, TERENA Networking Conference, 2000.

[11] Luciano Paschoal Gaspary, Ederson Canterle.
Assessing Transaction-based Internet Applications Performance through a Passive Network Traffic Monitoring Approach
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/9481/30080/01378379.pdf?arnumber=1378379, In: IEEE Global Telecommunications Conference, Synopsium on Security and Network Management, Dallas, 2004.

[12] Luciano Paschoal Gaspary, Edgar Meneghetti, Fabricio Wendt, Lucio Braga, Roberto Storch, Liane Tarouco.
High-layer Protocol and Service management Based on Passive Network Traffic Monitoring: the Trace Management Platform
http://mutuca.metropoa.tche.br/PDF/2002_passive_13.pdf, In: IEEE Synopsium on computers and communications, Taormina, Proceedings of the Seventh IEEE Symposium on Computer and Communications, 2002.

[13] Luciano Paschoal Gaspary, Edgar Meneghetti, Fabricio Wendt, Lucio Braga, Roberto Storch, Liane Tarouco.
Trace: An Open Platform for High-layer Protocols, Services and Networked Applications Management
https://129.132.99.171/http/0/ieeexplore.ieee.org/iel5/7927/21855/01015630.pdf?tp=&arnumber=1015630&isnumber=21855, In: IFIP/IEEE Network Operations and Management Synopsium, Florence, 2002.

[14] Marcia Zangrilli, Bruce B. Lowekamp.
Using Passive Traces of Application Traffic in a Network Monitoring System
https://129.132.99.171/http/0/ieeexplore.ieee.org/iel5/9239/29284/01323495.pdf?tp=&arnumber=1323495&isnumber=29284, In Proceedings. 13th IEEE International Symposium on High performance Distributed Computing, 2004.

[15] Mark Stemm, Randy Katz, Srinivasan Seshan.
A Network Measurement Architecture for Adaptive Applications
https://129.132.99.171/http/0/ieeexplore.ieee.org/iel5/6725/17985/00832198.pdf?tp=&arnumber=832198&isnumber=17985, In Proceedings. IEEE Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2000.

[16] Marko Zec, Miljenko Mikuc, Mario Zagar.
Estimating the Impact of Interrupt Coalescing Delays on Steady State TCP Throughput
http://www.tel.fer.hr/zec/papers/zec-mikuc-zagar-02.pdf, University of Zagreb, Croatia, 2005.

[17] Matt Mathis, John Heffner, Raghu Reddy.
Web100: Extended TCP Instrumentation
http://www.web100.org/docs/mathis03web100.pdf, Pittsburgh Supercomputing Center, Pittsburgh PA, 2003.

[18] Myung-Kyun Kim, Jinsoo Kim.
Design and Implementation of a Web-Based RMON Agent System
https://129.132.99.162/http/0/ieeexplore.ieee.org/iel5/7691/21034/00975074.pdf?tp=&arnumber=975074&isnumber=21034, In: Proceedings. The Fifth Russian-Korean International Symposium on Science and Technology (KORUS), 2001.

[19] N. Brownlee.
Using NeTraMet for Production Traffic Measurement
https://129.132.99.171/http/0/ieeexplore.ieee.org/iel5/7332/19847/00918033.pdf?tp=&arnumber=918033&isnumber=19847, In. IEEE/IFIP International Symposium on Integrated Network Management Proceedings, 2001.

[20] *network lab - Netzwerkanalyse: Typische Problemsituationen Teil 2*
http://www.nwlab.net/guide2na/netzwerkanalyse-probleme-2.html, 07.03.2006.

[21] *NLANR - National Laboratory for Applied Network Research*
http://www.nlanr.net/, 21.03.2006.

[22] Philip R. Bevington, D. Keith Robinson.
Data Reduction and Error Analysis for the Physical Sciences
Mc Graw Hill, New York, NY, USA, 2003.

[23] Prasad Calyam, Dima Krymskiy, Mukundan Sridharan, Paul Schopis.
Active and Passive Measurements on Campus, Regional and National Network Backbone Paths
https://129.132.99.162/http/0/ieeexplore.ieee.org/iel5/10216/32582/01523933.pdf?tp=&arnumber=

1523933&isnumber=32582, Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN), 2005.

[24] *RFC 1122 - Requirements for Internet Hosts – Communication Layers*
October 1989.

[25] *RFC 1701 - Generic Routing Encapsulation (GRE)*
October 1994.

[26] *RFC 2401 - Securtiy Architecture for IP*
November 1998.

[27] *RFC 3465 - TCP Congestion Control with Appropriate Byte Counting (ABC)*
February 2003.

[28] R. Prasad, M. Murray, K. Claffy, C. Dovrolis.
Bandwidth Estimation: Metrics, Measurement Techniques, and Tools
http://www-static.cc.gatech.edu/fac/Constantinos.Dovrolis/Papers/NetDov0248.pdf, In: IEEE Network, 2003.

[29] S. Niccolini, M. Molina, F. Raspall, S. Tartarelli.
Design and implementation of a One Way Delay passive measurement system
https://129.132.99.171/http/0/ieeexplore.ieee.org/iel5/9208/29204/01317734.pdf?tp=&arnumber=1317734&isnumber=29204, In: IEEE/IFIP Network Operations and Management Symposium (NOMS), 2004.

[30] Srinivas Shakkottai, R. Srikant, Nevil Brownlee, Andree Broido.
The RTT Distribution of TCP Flows in the Internet and its Impact on TCP-based Flow Control
http://www.caida.org/outreach/papers/2004/tr-2004-02/tr-2004-02.pdf, Cooperative Association for Internet Data Analysis - CAIDA, San Diego Supercomputer Center, University of California, San Diego, University of Illinois, 2004.

[31] Steven Waldbusser.
Application Performance Measurement MIB
http://www3.ietf.org/proceedings/02mar/I-D/draft-ietf-rmonmib-apm-mib-06.txt, Internet Draft, 2002.

[32] Takeo Saitoh, Glenn Mansfield, Norio Shiratori.
Network Congestion Monitoring and Detection using the IMI infrastructure
https://www.webvpn.ethz.ch/http/0/ieeexplore.ieee.org/iel5/6466/17281/00797434.pdf?tp=&arnumber=797434&isnumber=17281, Proceedings of the International Conference on Parallel Processing, 1999.

[33] *Techniques of Observational Astronomy - Basic Statistics*
http://www.astro.ufl.edu/ oliver/ast3722/lectures/BasicStatistics/BasicStatistics.htm, 24.03.2006.

[34] Tony McGregor, Hans-Werner Braun, Jeff Brown.
The NLANR Network Analysis Infrastructure
https://129.132.99.162/http/0/ieeexplore.ieee.org/iel5/7803/21447/00994555.pdf?tp=&arnumber=994555&isnumber=21447, In Proceedings. Symposium on Applications and the Internet (SAINT) Workshops, 2002.

[35] *The CAIDA Web Site*
http://www.caida.org/home/, 21.03.2006.

[36] Venkata N. Padmanabhan; Sriram Ramabhadran; Jitendra Padhye.
Client-based Characterization and Analysis of End-to-End Internet Faults
http://research.microsoft.com/ padmanab/papers/msr-tr-2005-29.pdf, Mircrosoft Research, Redmond WA, 2005.

[37] William Stallings.
Network Security Essentials - Applications and Standards
Pearson Education, Upper Saddle River, New Jersey, USA, 2003.

[38] W. Richard Stevens.
TCP/IP Illustrated, Volume 1 - The Protocols
Addison-Wesley, Massachusetts, USA, 1997.

[39] W. Richard Stevens, Gary R. Wright.
TCP/IP Illustrated, Volume 2 - The Implementation
Addison-Wesley, Massachusetts, USA, 1997.

[40] Zhang Hui, Li Xing, Li Zimu.
A Scalable High Performance Network Monitoring Agent for CERNET
https://www.webvpn.ethz.ch/http/0/ieeexplore.ieee.org/iel5/8749/27719/01236277.pdf?tp=&arnumber=1236277&isnumber=27719, Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2003.

[41] Zhang Yibo, Lei Zhenuming.
Estimate Round Trip Time of TCP in a passive way
https://129.132.99.171/http/0/ieeexplore.ieee.org/iel5/9834/30995/01442141.pdf?tp=&arnumber=1442141&isnumber=30995, Proceedings of the 7th International Conference on Signal Processing (ICSP), 2004.