SERVICE ASSURANCE METHODS AND METRICS FOR PACKET SWITCHED NETWORKS

PhD thesis

Pál Varga

Supervisors:
Péter Tatai and Géza Gordos DSc.

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
DEPARTMENT OF TELECOMMUNICATIONS AND MEDIA INFORMATICS
2010.
Alulírott, Varga Pál kijelentem, hogy ezt a doktori értekezést magam készítettem és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelölt sem.

A dolgozat bírálatai és a védésről készült jegyzőkönyv a későbbiekben, a Budapesti Műszaki és Gazdaságtudományi Egyetem dékáni hivatalában lesznek elérhetőek.

Budapest, 2010. november

..................
To my family

Thank you for your patience
and for the everlasting encouragement
Contents

1 Introduction ........................................... 1
  1.1 Overview ......................................... 1
  1.2 Structure of the Thesis .......................... 2

2 Service Assurance for Networked Applications ....... 5
  2.1 Achieving Satisfactory Level of Service ........ 5
  2.2 Network Management Terminology ............... 6
  2.3 Related Work .................................... 8
    2.3.1 Performance Monitoring Tools and Architectures 10
    2.3.2 Service Level Monitoring and Inter-domain QoS Issues 11
  2.4 Service Assurance Framework .................... 13
    2.4.1 Collecting and Reporting Events in General . 15
    2.4.2 Event Notification Collector ................ 16
    2.4.3 Trend Analysis and Outlier Detection .......... 18
    2.4.4 Event Preprocessing ........................ 18
    2.4.5 Alarm Notification Presentation .............. 21
    2.4.6 Root Cause Analysis ........................ 21
    2.4.7 Advice and Modification .................... 21
    2.4.8 Event Notification Database ................ 22
    2.4.9 Topology Database .......................... 22
  2.5 The Philosophy of Connectivity Fault Management . . . . 22
    2.5.1 Measurements and Metrics - examples for Ethernet SA .. 25
  2.6 Conclusions ...................................... 25
3 Root Cause Analysis Method Modeling the Human Expert Behavior

3.1 Passive and Active Root Cause Analysis Methods ........................................ 29
3.2 Modeling Human Expert Behavior in Fault Localization ................................. 31
3.3 The Root Cause Analysis Framework ............................................................. 32
  3.3.1 Petri Net Based RCA Execution ............................................................. 33
  3.3.2 The Petri Net Representation of RCA Action Plans ................................. 34
  3.3.3 The RCA Scheduler .................................................................................. 35
  3.3.4 Elementary Diagnostic Checks .................................................................. 36
3.4 Case Studies .................................................................................................... 37
  3.4.1 Case 1: Ethernet Services, Connectivity Fault ........................................ 37
  3.4.2 Case 2: Ethernet Services, SLS Violation Detected ................................. 39
  3.4.3 Case 3: VoIP Services, High Packet Loss ................................................ 40
3.5 Conclusions .................................................................................................... 42

4 Localizing Network Bottlenecks

4.1 Passive and active approaches ................................................................. 45
4.2 Typical Bottleneck Properties ........................................................................ 47
4.3 Delay Factor of Flows ..................................................................................... 48
4.4 Packet Interarrival Times Distribution ......................................................... 51
  4.4.1 Higher Order Statistical Properties .......................................................... 52
  4.4.2 Packet Interarrival Times PDF of Links with No Congestion ................. 53
  4.4.3 Effects of the Bottleneck Behavior on Interarrival Times Distribution .... 54
4.5 Performance Evaluation Based on Simulations ............................................ 55
  4.5.1 Simulation Environment ............................................................................. 55
  4.5.2 Simulation and Measurement Results for Delay Factor ......................... 56
  4.5.3 Common Patterns of PIT PDFs ................................................................. 59
4.6 Evaluating Bottleneck Metrics by Real Measurements ............................... 62
  4.6.1 Real Measurement Results for Delay Factor .............................................. 62
  4.6.2 Using PIT Kurtosis to Detect Bottlenecks in Real, Operational Networks 63
4.7 Conclusions .................................................................................................... 66

iv
5 Service Quality from the End–user’s Perspective: Quality of Experience 69
5.1 Introduction ........................................................................ 70
5.2 Overview of the Metrics Chosen for Determining QoE ............. 70
  5.2.1 Throughput .................................................................... 70
  5.2.2 Loss Based Metrics ....................................................... 71
  5.2.3 Kurtosis of Packet Interarrival Times .............................. 72
  5.2.4 Delay Factor Calculus ................................................... 73
5.3 Measurement Environment .................................................. 74
5.4 Measurement Results ......................................................... 75
5.5 Conclusions ....................................................................... 78

6 Connecting the Knowledge Plane with the SA Framework 79
6.1 The Knowledge Plane Concept ............................................ 79
6.2 Related Work ..................................................................... 82
6.3 Corresponding Functions of the SA Framework and the KPlane Concept ......................................................... 84
6.4 The Monitor Plane ............................................................. 84
  6.4.1 The Advantage of Monitoring at Highly Aggregated Connections ......................................................... 85
  6.4.2 Distributed Processing .................................................. 85
  6.4.3 Essential Functionalities of Capture Probes ...................... 87
6.5 Traffic Matrix ..................................................................... 87
6.6 Traffic Mix ......................................................................... 88
6.7 Conclusions ....................................................................... 89

7 Summary .............................................................................. 91
7.1 Summary of the Theses ........................................................ 91
7.2 Results Applied in Practice .................................................. 93
List of Tables

2.1 Measurements and Metrics [97] ...................................................... 27

3.1 Elementary check function examples for end-to-end Ethernet services .... 37

4.1 PIT skewness values in simulated environment .............................. 61
4.2 PIT kurtosis values in simulated environment .............................. 62
4.3 PIT kurtosis values of measurements at the operator’s site ............ 64

5.1 Perceived quality of network services at different bottleneck scenarios 75
List of Figures

2.1 The management levels and tasks recommended by the Telecommunication Management Network framework ........................................ 7
2.2 The scope and functions of Service Assurance .......................................................................................................................... 8
2.3 Connection of elements of the event processing and alarm handling model .................................................................................. 14
2.4 The correlation window and the matching mechanism of event correlation rules ........................................................................ 19
2.5 Messages path in Connectivity Fault Management [97] .................................................................................................................... 23

3.1 The architecture of the RCA model using active checks scheduled by Petri nets ........................................................................ 33
3.2 Example of executing a simple Petri net ........................................................................................................................................ 35
3.3 Petri net to drive the "CFM connectivity fault" alarm ....................................................................................................................... 38
3.4 Petri net to drive the Root Cause Analysis of "SLS violation" alarm .............................................................................................. 40
3.5 Petri net to drive the Root Cause Analysis of "High VoIP packet loss" ............................................................................................ 41

4.1 Packet interarrival times PDF on an aggregated link with no queuing (left) and on an aggregated link, with eventual queuing (right) ........................................................................................................ 54
4.2 Network topology used during the simulations ............................................................................................................................. 55
4.3 Delay factor calculated by inter-arrival times (left) and sojourn times (right) in three scenarios ......................................................................................................................... 57
4.4 Packet interarrival time PDFs in simulation scenarios 700/600 (left) and 1000/300 (right) ......................................................................................... 60
4.5 Measurement topology at the operator’s site .................................................................................................................................. 64
4.6 Packet interarrival time PDFs calculated based on real network data ............................................................................................. 65

5.1 Typical packet interarrival probability density functions ............................................................................................................ 73
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Measurement architecture with demonstrating the available bandwidth</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>degradation of download volumes after the two bottlenecks</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>Varying of network performance metrics during the whole measurement</td>
<td>76</td>
</tr>
<tr>
<td>5.4</td>
<td>Average of bottleneck metrics for the whole measurement periods</td>
<td>77</td>
</tr>
<tr>
<td>5.5</td>
<td>Number of TCP retransmissions for two available bandwidth scenarios</td>
<td>78</td>
</tr>
<tr>
<td>6.1</td>
<td>Functions of the Knowledge Plane and its connections to Control and</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>Data Planes</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>The relation of the Monitor, Knowledge and Action Planes to the</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>Network</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>A scalable solution for Traffic Analysis of high-speed network links</td>
<td>86</td>
</tr>
<tr>
<td>6.4</td>
<td>VoIP Portion visualization of a Traffic Mix analysis</td>
<td>90</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Overview

Users take full advantage of network services nowadays, without even noticing the fact of accessing a computer network. Utilizing applications available at a remote host became as simple as running a program in the local desktop computer.

Operating and maintaining the servicing networks have remained the hidden functions of service providers and network operators. Network and service management has always been a complex task, even inside the operators’ own network. Furnishing satisfactory services in a multi-provider environment is even more challenging. Under real-life conditions the user data can traverse the territory of numerous network operators while reaching the servicing node.

Naturally, the user wishes to utilize network services without having to care about the background technical issues. Hence the service should be assured end-to-end, irrespective of the number, type and relation of the networks traversed by his/her traffic. Theoretically this service assurance can be covered by Service Level Agreements (SLA’s) between the service providers and network operators. Similar SLA’s – contracts with some technical content – are signed between the end-user and the provider of the accessed service. This is the usual code of practice nowadays, however, there are several issues with this approach.

1. The technical parameters or SLS’s – Service Level Specification – included in the SLA describe only statistical properties, possibly hiding unavailable servicing periods
for critical user-applications.

2. The SLS's defined at network-boundaries are related to the agreement between the corresponding operators. Meeting the SLS requirements at the network-borders does not necessarily guarantees similar level of end-to-end service quality for each affected service.

3. The statistical SLS values suggesting high quality at the operators' side gives "false feeling of safety". This can lead the operators to make false assumptions, and never notice that some key users are very dissatisfied with the service.

Assuring end-to-end service in a multi-provider environment requires appropriate methods and metrics for realizing service degradations and localizing their causes. This thesis aims to point out the issues of assuring service quality for networked applications by proposing a service assurance architecture, measurement methods, and advanced metrics. The proposed framework fits into the prevalent code of practice, in which the network operators and service providers manage their own territory. The advances lay in the integration of active and passive fault management methods; the enhancement of the usual QoS measures; and the idea of correlating the service quality perceived by the user (Quality of Experience, QoE) with the analysis results of core network measurements.

The advances of this dissertation lay in the integration of active and passive fault management methods; the enhancement of the traditional Quality of Service (QoS) measures; and the idea of correlating the service quality perceived by the user (Quality of Experience, QoE) with the analysis results of core network measurements. Furthermore, the new measurement and analysis methods and metrics are integrated into the current autonomous network management concept of knowledge plane, monitoring plane and action plane.

1.2 Structure of the Thesis

The body of this thesis is organized as follows.

Chapter 2 describes a general service assurance framework that takes into account both network and service management viewpoints. It describes the key elements of network man-
agement (NM), and refreshes the requirements of fault management (FM). Since most FM systems are based on analyzing data collected by passive measurements, the main metrics and methods of network performance measurement (PM) will be summarized here as well. The description of the proposed service assurance framework includes the description of data collection methods, data processing guidelines, and a short summary of fault localization requirements.

Once a possible fault or service degradation is identified, the key question in the operator’s point of view is whether the root cause of the case can be found inside the managed area. Chapter 3 introduces root cause analysis methods and algorithms based on passive alarm correlation techniques. The classic fault localization models, however, can be enhanced with the initiation of active checking routines. Since the main problem with the active checks appears to be their correct scheduling, this thesis proposes to use data driven architectures for root cause analysis.

The modeling of human expert behavior shows that experts initiate checking routines based on the available data patterns. The result of some checks provide input-data for further checks, which can lead to the root cause of the fault. Such data-driven routines can be described by Petri nets. Chapter 3 describes the usability and limitations of Petri nets for root cause analysis. To support the idea, practical examples are provided for typical faults appearing in a Voice over IP (VoIP) service network. Since the model can be applied for more foundational architectures, this chapter also outlines how does it fit in the practice of multi-provider overlay Ethernet networks’ operations and maintenance (OaM).

The most noticeable service degradations are caused by failures of critical network nodes. Network operators are prepared for such cases: the alarms raised by such events describe the place and type of the failure, hence the recovery process can start instantly. There are, however, cases where the signs of mass service degradation are not straightforward, and the actual place of a possible network bottleneck can not be determined immediately.

Passive measurement methods should have been able to predict such cases, but there were a lack of appropriate bottleneck detection metrics. Chapter 4 describes why traditional QoS metrics such as throughput and loss are not suitable for bottleneck detection. There are two novel metrics get introduced: the delay factor of flows and the kurtosis of packet interarrival times probability density function (PTT kurtosis, in short). The scalability and
limitations of these metrics are demonstrated throughout simulations as well as analyzing such properties of real network traffic.

One of the reasons why the existence of network bottlenecks is hard to determine is that a great deal of network-applications (Internet browsing, mass data download) are not sensitive to delay, packet delay variation, or temporal service unavailability. Until a certain threshold the users of these applications are more tolerant to service degradation than the ones using streaming services (i.e. video on demand) or time-critical applications (i.e. on-line gaming). The correlations between Quality of Experience and the metrics derived by passive measurement of core network links is described in Chapter 5. The presented evaluation also highlights the difference of effectiveness between traditional metrics (such as loss rate and throughput), and the previously introduced bottleneck detection measures (delay factor and PTT kurtosis).

The presented Fault Management- and Performance Management-oriented approach of the introduced Service Assurance framework hide the fact that the SA framework can actually be used as a practical implementation of certain autonomous networking concepts. One of the philosophies that answer the issues raised by the requirements of self-management, self-configuration and self-optimization is the Knowledge Plane (KPlane). The concept of the split Knowledge Plane - where Monitoring, Knowledge and Action planes cover different functions - is widely used in various levels of network and service management. Nevertheless, general traffic analysis is not yet utilized in order to support decision making in the KPlane. In Chapter 6, the elements of the SA framework are generalized, and mapped with the functions of the split KPlane. Furthermore, a scalable traffic analysis concept is introduced for the MPlane, which helps extracting valuable information and structure it for creating traffic mix and traffic matrix reports.
Chapter 2

Service Assurance for Networked Applications

2.1 Achieving Satisfactory Level of Service

In general, assuring that the service is available for the users in at least a certain, agreed level of quality is the responsibility of the service provider. The two ultimate aims regarding service assurance are:

- the advertised level of service should be made available for new applicants as well as for established users, and

- service degradation cases should be noticed and acted upon before the users notice it and feel dissatisfied.

This applies to services provided over computer networks, as well. The additional challenge with networked services lays in the underlying architecture: the full network path is usually not supervised by one single operator. The end-to-end current solution is not at all technical: separate service level agreements (SLA’s) are concluded along the service path between the network operators, service providers and the user. Keeping the agreement is up to the parties – who have defined the technical content of the SLA as a service level specification (SLS). The subject of the SLS is usually related to service availability rate over certain time periods and available bandwidth. Keeping an agreement for such general
measures clearly does not assure continuous high level service for time-critical applications although the end/user is sensitive to them. Furthermore, the SLS approach does not include the prevention, observation and elimination of service degradation cases.

According to the above description, service assurance should include service level as well as network level methods and metrics mainly for performance management and fault management. The requirements and possibilities are detailed in the following sections. A general service assurance framework is also proposed in this chapter.

2.2 Network Management Terminology

The traditional network management functions are defined by ITU-T in [1]. These may be grouped into five distinct management areas, namely fault, configuration, accounting, performance and security management. This model is lately called "FCAPS".

When focusing on service availability and quality, the misbehavior of various objects both in the managed network and in the service context is being analyzed constantly. In order to keep up-to-date status of the managed system, permanent evaluation of the network elements is unavoidable. This is the main task of performance management (PM). Keeping track of the hardware and software configuration changes in the managed network is the main function of configuration management (CM). Fault management (FM) encompasses the detection, isolation and correction of abnormal operation of the managed objects (network elements, service applications, etc.). This is efficient only if the operator has up-to-date status information (provided by PM) and interconnection information (provided by CM) about the objects. Intrusions and frauds are to be prevented by security management (SM), which also covers the detection, handling and logging of such activities. Accounting management addresses important issues, these, however, have minor effect on service quality and performance, hence I do not discuss them here.

The general TMN model is traditionally pictured as a triangle with a perspective. The triangle represents the various management levels from business processes to the network elements, whereas the perspective of the triangle suggests that the FCAPS tasks should be taken care of at each level. Figure 2.1 depicts this basic TMN view of the management functions.
Service Assurance (SA) is a de facto term of the industry, covering FM and PM functionalities in the network management and service management levels.

The current thesis proposes a broader scale for SA. The suggested SA framework utilizes fault management (FM), performance management (PM), service level specification (SLS) validation, as well as event- and alarm-processing regarding the notifications arriving from these management entities. Moreover, the framework comprises the processing of error reports logged by the operator, as well as event notifications arriving from the managed (network and service) entities themselves and security management (SM) alarms, too.

Figure 2.2 provides an overview of the scope of the SA model, as well as the covered TMN and non-TMN functionalities.

The SA model to be presented here was historically created to cover Voice over IP (VoIP) performance and fault management tasks. The model was created in order to help VoIP service providers maintaining the servicing nodes and interfaces of their own territory. Later this model was applied to Ethernet overlay services, supporting end-to-end service assurance efforts in a multi-provider Ethernet environment. This framework included connectivity fault management (CFM) and PM-based models specialized for Ethernet services. Based on these
experiences the SA model was generalized. The result is detailed in the following sections, after presenting the state of the art of operations and maintenance related research activities.

![Diagram of Service Assurance](image)

**Figure 2.2:** The scope and functions of Service Assurance

### 2.3 Related Work

Since Service Assurance is a term lacking of any standardization background, management system vendors have come up with their own interpretation of SA.

Company CA provides a solution for end-to-end service assurance for MPLS (Multi Protocol Label Switching) VPN (Virtual Private Network) services [2]. It utilizes the management
features of MPLS technology, and uses traditional QoS measures (throughput, packet loss, packet delay variation, latency) to determine the end-to-end service quality.

EMC Smarts and InfoVista have joined forces to provide a fault and performance management solution for global IT infrastructures. This solution combines performance monitoring and passive fault management functions on a network and service level [3]. The solution inherited the strong root cause analysis (RCA) engine of Smarts, features great performance and fault visualization; it lacks, however, of active fault localization methods and service level specification validation possibilities.

Similarly, Micromuse and Quallaby have merged to be able to provide an integrated fault and performance management solution [4]. The additional feature of this product is the possibility of monitoring the SLA of large size accounts. This also allows the operators to correlate the network faults with the impacted services, and weight the fault by the possible revenue-loss. A passive root cause analysis engine is also part of the system.

Spectrum by Concord utilizes performance management and SLS validation methods in a "technology relationship mapping" solution. According to their white paper [5] "SLAs are tied to the business processes, so issues can be prioritized based on the importance of the customers and services that are impacted. Real-time alarms are generated, warning of service outages and impending SLA violations (including the root cause), and allowing them to be addressed quickly before the business is affected." This solution includes the visualization of the pure alarms, however, it does not feature advanced RCA methods.

The common advantages of these systems are the advanced network and performance views, and the intention of monitoring service availability in order to increase customer satisfaction (as well as to minimize revenue loss). Nota bene, these solutions are commercial, well-established products. The drawback of these systems, however, is the limited usage of advanced performance metrics, and the missing utilization of any active fault localization method. Suggestions for such extensions are detailed in the following chapters.

Before the brief overview of the proposed, general service assurance framework, the state of the art research activities of performance and service monitoring is reviewed.
2.3.1 Performance Monitoring Tools and Architectures

The RIPE Test Traffic Measurement (TTM) [6] project of RIPE-NCC covers the implementation of metrics and measurement protocols defined by the IETF IP Performance Metrics (IPPM) working group. Utilizing the worldwide-installed RIPE-TTM probes allows the operator to test the connectivity of his/her network segment with other parts of the Internet. The defined metrics are based on active performance measurements such as one-way delay, loss, delay variation, round-trip delay, and link bandwidth.

National Internet Measurement Infrastructure (NIMI) [7] is a network measurement architecture based on active probes deployed network-wide. NIMI generates probe traffic sent between the measurement nodes and computes traffic characteristics based on the captured traffic patterns. NIMI calculates metrics such as available bandwidth, delay and packet loss.

NetFlow architecture from Cisco consists of passive meters (usually installed in routers) and a flow collector (providing data filtering and aggregation capabilities). It generates statistics on byte and packet counts per micro or aggregated flows. There exists several post-processing and visualizing applications for analysis and presentation of NetFlow-based measurements. Such applications include flow-tools [8], cflowd of CAIDA [9] and the Multi Router Traffic Grapher (MRTG) [10]. Latter generates HTML pages containing GIF images which provide a live visual representation of the measured traffic.

Remote Network Monitoring (RMON) uses the traditional client/server architecture of network management applications (clients) and the monitoring devices (servers). These remote probes run the RMON "agent" software program. RMON is closely tied to the Simple Network Management Protocol (SNMP) architecture: the calculated statistics are gathered into the RMON MIB (Management Information Base) [11, 12] that the client applications can inquiry. An other approach that is implemented in most probes is to set thresholds and, when a threshold is crossed, alert the client application with an SNMP trap. RMON version 1 provides the network management centre (NMC) with packet-level statistics about an entire LAN or WAN. RMON version 2 is an enhancement of RMONv1, offering the calculation and presentation of network- and application-level statistics.

The Measurement and Network Analysis Group of NLANR has developed the Network Analysis Infrastructure (NAI) [13]. This covers both passive and active network measurement
and analysis. The Passive Measurement and Analysis Project defines workload profiles for a number of measurement points in high-speed environments, whereas the Active Measurement Project addresses performance parameters such as round-trip time, packet loss, topology, and throughput between the deployed number of active monitors.

The Cooperative Association for Internet Data Analysis (CAIDA) supervises several measurement-related projects for monitoring Internet traffic. The long list of CAIDA tools [9] that can be utilized in Ethernet OAM includes

- NeTraMet - an implementation of the IETF Real-Time Traffic Flow Measurement (RTFM) architecture,
- cflowd - for NetFlow analysis,
- RTG (Real Traffic Grabber) - a flexible SNMP statistics monitoring system,
- skitter and scamp - for actively probing the Internet in order to analyze topology and performance,
- CoralReef - to collect and analyze data from passive Internet traffic monitors, in real time or from trace files.

The Sprint IP Monitoring (IPMON) project [14] uses passive, packet-level monitoring methods to be used up to OC-192 speeds. IPMON probes generate statistics for offline analysis.

2.3.2 Service Level Monitoring and Inter-domain QoS Issues

The European projects of IST-INTERMON, IST-MoMe and IST-SCAMPI focus on 1) interdomain QoS monitoring, analysis, and modeling, 2) enhancing real-time QoS architectures with integrated monitoring and measurement capabilities and 3) open and extensible network monitoring architecture and measurement tools to be used for denial-of-service detection, SLS auditing, QoS, traffic engineering (TE), traffic analysis, billing, and accounting [15].

Rondo [16] is an automated data collection and control system aiming to manage congestion of core networks by rerouting in near-real time. Rondo relies heavily on Multi Protocol Label Switching (MPLS).
NetScope [17] offers a unified set of tools in relation to service level monitoring and handling network performance degradation. Its TE solution works by measuring the network to derive traffic demands and then initiating network configuration updates in a non-real-time manner.

The authors of [18] declare TE guidelines and requirements of a scalable monitoring system, featuring event monitoring and in-service performance verification. The events of the distributed node monitors arrive at centralized network monitoring and SLS monitoring entities for evaluation. Node monitors can provide active measurement results on path or hop level, whereas measures of passive monitoring are provided relating to the router they are attached to. SLS management takes its input from the SLS repository (SLS scope, ingress-egress points) and from the monitoring repository (containing results from node monitors and the network monitor).

An other generic and scalable system for connection management and SLS monitoring of VPN services is presented in [19].

Since end-to-end service level agreements are hard to maintain in large networks, one solution is to reduce the information exchange between distributed nodes and a central network management system. The authors in [20] propose an aggregation and refinement method addressing these issues.

Mechanisms for gathering and processing information about available QoS parameters before purchasing another service are proposed by [21]. The authors also describe various metrics for inter-domain QoS measurements, together with proposing traffic prediction methods.

Various policies are proposed by [22] for distributing SLS elements across multiple domains to provide end-to-end QoS.

An other approach to support inter-domain QoS guarantees is to extend the underlying Border Gateway Protocol (BGP) with QoS attributes. Instances of such effort can be studied in [23, 24].

A comprehensive description of the issues and key requirements of inter-domain QoS provisioning is presented in [25]. The authors describe how can end-to-end QoS be achieved through centralized traffic engineering. Further QoS enhancements to BGP are also proposed and evaluated in this paper.
The author of [26] describes industrial efforts to provide multi-dimensional service aware management services for end-to-end Ethernet services. The description projects a system where the CFM functions on various service and network levels including SDP (Service Delivery Point), LSP (Label Switched Path), VPLS (Virtual Private LAN Services) would proactively find and diagnose network and service anomalies.

Alcatel provides IP/MPLS service management and operations support system described in [27]. This robust system offers a unified service provisioning and assurance solution, using simple tools such as ping, loop back, path trace, forward and backward defect indications. This system utilizes performance monitoring results as well, hence being very close to meet the Y.1731 [28] and 802.1ag [29] requirements set for Ethernet network management.

The above mentioned methods, tools and systems address performance monitoring, QoS or SLS measurement. They may also deal with deep analysis of the measured data and reaction based on the analysis. Nevertheless, none of the studied systems and tools integrate the data collection and analysis with advanced fault management functions such as event correlation or root cause analysis.

The SA framework described in the following – and first introduced by me and my colleagues in [52] and [97] – addresses these issues, by using active and passive performance and fault management methods focusing on the operations and maintenance issues of end-to-end networked services.

2.4 Service Assurance Framework

This section describes a general framework for Service Assurance (SA) functions, methods and activities. In this thesis a generalized meaning of Service Assurance is used, namely: it covers all the functions that help assuring fluent services over the managed network. SA systems are not mandatory when deploying a service, since in a perfect world services run without problems from the first moment. Once we begin to suspect that the world is not perfect, we realize the necessity of SA.

The proposed service assurance framework should be considered as an umbrella, utilizing – beside others – Connectivity Fault Management (CFM), Performance Monitoring (PM) and SLS Verification tools, methods and metrics. The event notifications generated by these
Figure 2.3: Connection of elements of the event processing and alarm handling model subsystems are analyzed by FM functions, which are also part of the SA framework.

This thesis describes the model from the event processing point of view (from the FM-prospective).

The input-signals of any fault management system are event notifications. These come from various sources, and each source has its own notification format. The FM system must recognize erroneous behaviour either presented or hidden by the event notifications, and find the root cause of these errors by initiating active testing and verification tasks. When the place and the nature of the fault are identified, the fault management system gives advice for corrections or (in special cases) takes corrective actions. Adaptive alarm handling (if implemented) must be supported by feedback mechanisms carrying the RCA results to the knowledge base of the event processing tasks.

Before describing the SA-framework, we should clarify the difference between event and alarm notifications. Events are generated by different parts of the managed system, notifying
about status changes minor, major and critical errors occurred at a given object. These events are sometimes alarming, but most of the time harmless for the overall system’s point of view. Alarm notifications, however, are clear, trouble ticket like objects, which in all cases need to be acted upon. There is always a fault behind an alarm notification.

As Figure 2.3 suggests, events arrive at the event processing and alarm handling model from various sources. The Event Notification Collector maps these to a standard format, then sends them to Event Preprocessing and stores them into the database. The Data Miner works on the stored events (seeing more historical events as well), and generates a new event when necessary. Once the events get correlated and filtered, only those alarms get presented to the NMC (Network Management Center), which should be acted upon. The Root Cause Analysis (RCA) for these alarms starts immediately. When the root cause is found, a fault description gets forwarded to the Advisor Module. This decides whether the corrective actions should be taken automatically, or only the advice (and the log about the tests carried out during RCA) should be forwarded to the human operator. The operator then decides what to do and he/she is also able to check if the automatic corrections taken were appropriate. The model includes learning mechanisms as well, which are indicated in Figure 2.3, too.

The following sections describe how the events (CFM, PM, SLS Verification, etc.) get generated and fed into this FM-flow, then detail the elements of the SA model.

2.4.1 Collecting and Reporting Events in General

Both connectivity fault and performance information can be gathered in a proactive or reactive manner. The proactive method performs continuous or periodical checks without prior solicitation – hence this method is also called unsolicited. The reactive method is activated by NMC via an explicit request – this method is referred to as solicited in the following sections.

Effective performance management can not be operated without a standalone performance monitoring system that is physically independent from the managed nodes. The main function of this equipment is to capture traffic at the monitored interfaces, process the data, and provide statistics on the traffic properties. Such performance monitoring tools can report statistical data as well as provide events of crossing some thresholds via SNMP.
SNMP

Most high speed network equipment support the Simple Network Management Protocol (SNMP) [30] as a management interface for remote applications. SNMP provides a way to verify the network element configuration, read basic statistics, and also provides an alarming functionality via configurable traps. The data for SNMP transmission is stored in a Management Information Base (MIB), located at the management entity. The managed network nodes have their own MIB specifications, since different vendors specify different statistics to report and different configuration parameters to store. Although a heterogenous network makes SNMP based management difficult, there are several standard bridge MIBs available [31] [32] that provide a guideline for equipment vendors.

Practical FM systems operated at ISPs (Internet Service Providers) often rely exclusively on SNMP traps and requests. This practice renders the active fault localization techniques more difficult. The solution lies in the solicited application of performance monitoring tools: the operator should be able to request special data capturing and processing functions in order to diagnose the faults interactively.

Syslog

Syslog is a de facto standard for forwarding log messages in an IP network. In a larger network the messages of various network elements are usually collected to a central server, where these messages are prioritized and then evaluated by the network administrator. Syslog messages often carry highly important event notifications that help to pinpoint exact failure conditions (for example interface failure notifications, software failures, etc.).

2.4.2 Event Notification Collector

The event sources are all addressing their notifications to this entity. It collects the events sent in different formats and reshape them to a standard notification format. Events will be stored in this standard format into the Event Notification Database and sent to the Event Processing module, too. The proposed event-generator entities are:

- *Connectivity Fault Management* - The proactive CFM function provides detailed information on connectivity issues. The philosophy of CFM is taken from Ethernet
OAM (refer to IEEE 802.1ag [29] and Y.1731 [28]), and summarized in Section 2.5. The advantages of CFM should be implemented for architectures other than Ethernet, as well.

- **Unsolicited Performance Monitoring** - Performance monitoring measurements are carried out at each node in an unsolicited manner (periodic checks without any external trigger). Such metrics are briefly summarized by Table 2.1.

- **SLS Verification** - SLS verification procedures are triggered periodically by verification agent instances. The metrics and checking routines are similar to the ones used in unsolicited performance monitoring, the initiation of these checks, however, is different. The SLS Verification framework includes a knowledge base about the SLS metrics to be met at different service levels (between different service entities). Solicited performance measurement routines are carried out based on this information.

- **Syslog** - Each node in the managed service area generates syslog messages. These should also arrive at the Event Notification Collector.

- **Security** - Events generated by the security management system should be used for fault management purposes as well. Examples for security management events are the MAC security related issues, unauthorized access attempt notifications, firewall alarms and intrusion detection alerts.

- **Other SNMP Traps** - There are various other types of SNMP traps that should arrive as event notifications, but do not fall into the above categories. Such traps arrive for example from the bridge nodes, notifying about exceeded capacity-limits, Spanning Tree Protocol (STP) changes or route changes.

- **User** - The operator should be able to log event (rather: alarm) notifications as well. This includes user complaints related to service quality, the notifications of the testing personnel, and the operator’s own notifications to help automatic fault localization.

- **Data Miner** - The source module of these events executes outlier detection and trend analysis algorithms on the complete set of event notifications arriving from other sources.
2.4.3 Trend Analysis and Outlier Detection

This module takes its input from the Event Notification Database. It continuously runs trend analysis algorithms and detects outliers that do not fit into periodic patterns on the data set and generates alarming events targeted to the Event Notification Collector.

The trend analysis (TA) function works on the raw event database, and looks for trends in event notifications. TA compares the events found in its "monitoring window" to the patterns stored in its knowledge-base and raises an alarm on a possible future event if the appearance of such an event can be assumed based on previous experiences. The basic analysis method is a rule-based approach, however a self-learning possibility of time-series data is also to be considered. [33, 34]

The outlier detection function also works on the raw (unfiltered) event database, categorizing and analyzing the events continuously. When parameters of an event are found to be significantly different from similar parameters of the neighboring events (in time and in category), the outlier detection function raises an alarm. [35]

Alarms generated by the Data Miner module arrives to the Event Notification Collector similarly to any other event notifications sent by "regular" sources.

2.4.4 Event Preprocessing

Event correlation (EC) is a term generally used in the literature for algorithms that enable gaining extra knowledge of correlating events arriving from various sources rather than independently investigating them. EC requires further clarification in the current fault management context: in here, EC algorithms take Events as input, pass them through, AND generate extra events based on the matching EC rule. This approach allows event filtering to work from an enlarged knowledge base. Furthermore, well defined and focused correlation rules provide a good foundation for later RCA activities, since their result can explicitly indicate the place of the fault.

Event filtering by default suppresses low severity events, passes high severity alarms. A refinement of this approach is described in the following sub-thesis. Nevertheless, even this basic definition enables the "important", original events and the EC-based events to become Alarms.
Root cause analysis (RCA) is generally mixed with event correlation, which is wrong. EC does not show the Root Cause of a fault, but a highly sophisticated symptom of it. Another way to describe their difference is that EC takes events for input, and provide sophisticated events or alarms for output – whereas RCA takes alarms from input and provides the root cause of the fault as the output.

In this context, the most effective way to connect EC, filtering and RCA modules is exactly in this order.

In the implementation, the Event Preprocessing module consists of a Correlator and a Filter sub-module. The Correlator looks for correlations between events. This sub-module passes all event notifications (either "correlated" or simple) to the Filter sub-module. The Filter suppresses unnecessary event notifications. The events that pass Event Preprocessing are the actual Alarms, and are taken care of the RCA.

Correlation

To reduce complexity, the Correlator in our SA framework is a rule based event correlation module. If a correlation rule is matched, a new event notification is being generated. Figure 2.4 depicts the matching mechanism of basic correlation rules.

Correlation rule: if during $t \in \tau$ events $(\hat{a}_1 | \hat{a}_2 | \hat{a}_3 | \langle b_1 \rangle)$ arrive then: report alarm $Z_1$

**Figure 2.4:** The correlation window and the matching mechanism of event correlation rules

Well defined and focused correlation rules provide a good foundation for later RCA
activities, since their result can explicitly indicate the place of the fault. This sub-module
passes all event notifications (either "correlated" or simple) to the Filter sub-module.

Filtering

There are four types of rule-based filtering methods carried out by the Filter sub-module of
our SA framework.

- **Counters** will pass an alarm through if the corresponding event arrives in a predefined
  number of times during the given time period. It is useful for low priority events that
  would be suppressed when arriving "alone", but should be acted upon when arriving in
  higher amounts (i.e., an unauthorized access attempt may indicate mistyped password,
  but hundred consecutive ones indicate a security problem);

- **Redundancy** filters pass the first occurrence of the alarm, and then suppress the other
  occurrences for a given duration. This type of filter is used for notifications of major
  and critical alarming events. They must be acted upon, but further notifications of
  the same kind are redundant, hence must be suppressed;

- **Dominance** filter needs two alarm IDs to be defined, A and B. It will pass A, and
  then (during a predefined period) suppress any occurrences of B. Network Connectivity
  alarms (A) usually dominate service availability notifications (B);

- **Suppress** filter simply suppresses event notifications. Low priority event notifications
  (i.e., a Syslog message about a harmless configuration change), base events of correlated
  alarm notifications, alarming events with ceasing arrived - these are the basic event
  types that should be included in suppress filter rules.

Filtering out transient network failures comes from the very own nature of the module.
It eliminates these transient conditions by means of a verification period. If the reported
condition goes away during the verification period, the module concludes that it is transient
and ignores it.
2.4.5 Alarm Notification Presentation

The output of Event Preprocessing are the alarm notifications. These will be presented to
the Network Management Center so the operator will see the distinguished, active alarms.
Each alarm notification will trigger a new fault localization (root cause analysis) entity.

2.4.6 Root Cause Analysis

Based on the descriptive parameters of the alarm notification, this module tries to find the
root cause of the problem. It fetches data from the Event Notification Database and the
Topology Database and initiates active checks based on this information.

RCA systems appeared in the last decade can be classified in many ways. One catego-
rization - from the point of view of implementation complexity - is the following:

- quasi-static techniques (codebook, rule based reasoning),

- methods with learning capabilities (Bayesian networks, neural networks, case based
  reasoning),

- mixed and other systems (model based, distributed, fuzzy, etc.).

The merge of these methods and approaches could also lead to useable and effective RCA
systems. [36], [37] and recently [38] also provided overviews of passive monitoring based
event correlation and RCA methods – researched and applied. Chapter 3 briefly describes
these methods and detail an advanced root cause analysis approach, modeling human expert
behavior.

2.4.7 Advice and Modification

The RCA output is a fault description. Based on the fault location and type, the module
presents the revealed fault to the NMC and may also give instructions for fault recovery. The
simplest way of providing such instructions is maintaining a knowledge base of fault descrip-
tions and the details of corresponding corrective actions. It depends on the operator’s policy
if he/she allows the system to make some of the corrective actions by itself. The corrective
actions certified to be done by the system automatically will be passed to the appropriate
Modification Agent. Configuration and software-related defects might be corrected this way, other problems, however, essentially require the actions of human operators. For these cases the proposed actions will appear at the Network Management Center, so the operator can carry out the necessary steps.

2.4.8 Event Notification Database

It stores all event notifications sent towards the management system. The notification description is standardized (regardless of the source) - it must contain all information to be used by the fault management system (alarm ID, priority, source, type, relevant connection addresses, short description where possible, other distinguishing parameters and values).

2.4.9 Topology Database

There are at least three important pieces of information stored here about each node. These are: Node address, Node type (functionality), List of connecting nodes (first hops, subnet addresses).

2.5 The Philosophy of Connectivity Fault Management

Connectivity Fault Management (CFM) is a term used in current Ethernet (ETH) network management recommendations. The field is currently under standardization at various international engineering organizations. Although CFM is a framework defined for managing interconnected Ethernet networks, its proactive methods can be applied in other architectures. This enables it for playing a key role in service assurance systems.[97]

802.1ag [29] CFM (Connectivity Fault Management) by IEEE introduces a scalable fault management system that fits into the multi-domain and multi-provider environment of interconnected Ethernet networks. Y.1731 [28] of ITU-T provides a slightly broader view, dealing with performance monitoring, too. The newly introduced SA framework intensively uses the methods and metrics defined in these recommendations.

CFM defines proactive and diagnostic fault localization procedures for point-to-point and multipoint EVCs (Ethernet Virtual Circuits) [39] that span one or more links. Its procedures
are defined end-to-end within an Ethernet network. A good overview of the Ethernet OAM can be found in [40].

The goal of CFM is to monitor a network that may be comprised of one or more Service Instances. An example for a Service Instance could be a LAN being made up by various means, including a VLAN or a concatenation of VLANs.

CFM has two types of key entities: Maintenance End Points (MEPs) and Maintenance Intermediate Points (MIPs) [97]. MEPs and MIPs are new software (or potentially hardware) entities created within a bridge for CFM only. MEPs and MIPs can be implemented per bridge or per port. MEPs initiate CFM messages and respond to CFM messages. MIPs passively receive these messages and respond back to the originating MEP. Also, MEPs prevent leaking of CFM messages between domains (i.e., among operators or between operators and customers). A Maintenance Association (MA) is a logical connection between any two MEPs. A Maintenance Domain consists of one or more MAs at the same level. There are eight levels defined in CFM. Entities at different levels have different responsibilities. Higher levels have a broader, but less detailed, view of the network.

![Diagram](image.png)

Figure 2.5: Messages path in Connectivity Fault Management [97]

Mechanisms supported by 802.1ag include Connectivity Check (CC), Loopback and Link trace. CFM allows for end-to-end fault management that is generally reactive (through Loopback and Link trace messages) and connectivity verification that is proactive (through
Connectivity Check (CC) messages are periodic "hello" messages multicast by a MEP within the maintenance domain, at the rate of X per second; where typical values of X can be e.g., 1, 10, 20, 100, etc. All MIPs and MEPs in the given domain will receive CC, they will, however, never respond to it. The receiving MEPs and MIPs will build a MEP database that has entries of the format [MEP DA, Port]. MEPs receiving this CC message will catalog it and know that the various maintenance associations (MAs) are functional, including all intermediate MIPs. MEPs will raise alarms in case of experiencing the loss of usually received CC messages. Beside detecting link failure or fabric failure, CC is able to indicate errors related to service cross-connect (MA ID mismatch), duplicate MEP configurations (MEP ID match), missing or unexpected MEPs (optional check: unexpected MEPs may not be an error), forwarding loops (duplicate sequence number), data loss (missing transaction IDs or lifetime expiration), and data corruption (bad optional data checksum) as well.

A Loopback message helps a MEP identify the precise fault location along a given MA. A Loopback message (manifested as a "ping" on the MAC level) is issued by a MEP to a given MIP along an MA. The appropriate MIP in front of the fault will respond with a Loopback reply. The MIP behind the fault will certainly not respond. For Loopback to work, the MEP must know the MAC address of the MIP to ping. This is done by means of configuration, discovery, or using a Link trace message after the connection is set up or reconfigured, well before the time to use a Loopback.

The Link trace message is used by one MEP to trace the path to another MEP or MIP in the same domain. It is needed for Loopback (MAC ping), too. All intermediate MIPs respond back with a Link trace reply to the originating MEP. After decreasing the TTL (Time-To-Live counter) by one, intermediate MIPs forward the Link trace message until the destination MIP/MEP is reached. If the destination is a MEP, every MIP along a given MA responds to the originating MEP. The originating MEP can then determine the MAC address of all MIPs along the MA and their precise location with respect to the originating MEP. Note the difference with IP, which does not support an IP trace route frame. IP uses the Ping frame and the ICMP reply (that has the TTL within it) to trace the route. In 802.1ag, the trace route function (via Link trace) is kept independent of the ping function (Loopback).
In case of failure, loss of CC is detected by one or more MEPs. If a MEP does not receive CC messages, it will issue a Loopback message to verify the failure.

A given level (i.e., customer) has no immediate way of knowing whether a fault originates at its level (link or fabric failure) and can therefore fix it, or whether it originates from a level below (link failure or fabric failure) and must wait for the problem to be fixed, and maybe impose penalties. Such issues raise the necessity of alarm suppression. The lower level failures should not be reported to the upper layer NMC. Otherwise, the upper layer NMC would be overwhelmed with indications of faults it cannot repair.[97]

### 2.5.1 Measurements and Metrics - examples for Ethernet SA

The various inputs used in our framework can be categorized by generating modules and collection methods. The data collection can be *unsolicited*, where periodic checks/polls are requested without further supervision, and the result of these checks/polls are immediately sent back, using some specific notification method. The other way for collection is *solicited*, which means that checks and measurements are initiated on request and their results are returned upon completion. Further metrics are the statistics collected automatically by the managed nodes. These can be inquired by SNMP. Table 2.1 summarizes the methods and metrics to be used for *Ethernet service assurance*.

### 2.6 Conclusions

Service assurance (SA) consists of fault management, performance management and service level specification validation at the network management and service management levels.

Current SA-related products have not yet integrated the functions of collecting advanced performance metrics, hence their usage in fault localization is limited. Similarly, none of the studied systems and tools integrate the data collection and analysis with advanced fault management functions such as event correlation or root cause analysis featuring active diagnostic analysis.

The proposed framework fills this gap, by integrating active and passive performance and fault management methods focusing on the operations and maintenance issues of end-to-end service assurance.
This chapter has categorized the input events of the service assurance framework by their source type. Various event sources have been suggested to form a complete, multi-domain SA, including connectivity fault management, performance monitoring, SLS validation, security applications, syslog, user-helpdesk, and general SNMP traps. Furthermore, the usage of trend analysis and outlier detection algorithms were proposed. This supports raising of further alarms by processing the event notification database-content.

The three main modules of our SA framework are *event preprocessing*, *root cause analysis* and *advice and modification*. During event preprocessing the input events get filtered and correlated in order to reduce their number and promote the actual alarms to carry out root cause analysis on. The RCA module aims at localizing the fault as well as providing description of its type of origin. The advice and modification module either provides a correction advice based on the root cause description, or initiates the actual corrections – depending on the nature of the fault, and the operator’s policy on allowing the SA system to reconfigure the network nodes.

The proposed general SA framework includes CFM and PM methods under standardization, furthermore, network and service management practices found to be useful in recent years. The aim of our SA framework is to ease the job of the operator’s personnel by integrating the service and network related OAM tasks in a multi-domain multi-provider environment.
<table>
<thead>
<tr>
<th>Source</th>
<th>Measure</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFM</td>
<td>Connectivity Check</td>
<td>Unsolicited</td>
<td>MEP Fault alarm notification</td>
</tr>
<tr>
<td>CFM</td>
<td>Loopback</td>
<td>Solicited</td>
<td>initiated by elementary checks</td>
</tr>
<tr>
<td>CFM</td>
<td>Link Trace</td>
<td>Solicited</td>
<td>initiated by elementary checks</td>
</tr>
<tr>
<td>PM</td>
<td>Frame Loss Ratio</td>
<td>Unsolicited</td>
<td>notification by SNMP</td>
</tr>
<tr>
<td>PM</td>
<td>Frame Delay measurement</td>
<td>Solicited</td>
<td>initiated by elementary checks</td>
</tr>
<tr>
<td>PM</td>
<td>Frame Delay Variation meas.</td>
<td>Solicited</td>
<td>initiated by elementary checks</td>
</tr>
<tr>
<td>PM</td>
<td>Availability</td>
<td>Statistics</td>
<td>statistics are usually</td>
</tr>
<tr>
<td>PM</td>
<td>Errored Frame Seconds</td>
<td>Statistics</td>
<td>gathered into MIBs,</td>
</tr>
<tr>
<td>PM</td>
<td>Service Status</td>
<td>Statistics</td>
<td>– otherwise collected</td>
</tr>
<tr>
<td>PM</td>
<td>Frame statistics</td>
<td>Statistics</td>
<td>with proprietary methods –</td>
</tr>
<tr>
<td>PM</td>
<td>Unavailable time</td>
<td>Statistics</td>
<td></td>
</tr>
<tr>
<td>SLS Validation</td>
<td>Frame Loss Ratio</td>
<td>Unsolicited</td>
<td>notification by SNMP</td>
</tr>
<tr>
<td>SLS Validation</td>
<td>Frame Delay measurement</td>
<td>Solicited</td>
<td>initiated by elementary checks</td>
</tr>
<tr>
<td>SLS Validation</td>
<td>Frame Delay Variation meas.</td>
<td>Solicited</td>
<td>initiated by elementary checks</td>
</tr>
<tr>
<td>SLS Validation</td>
<td>Availability</td>
<td>Statistics</td>
<td></td>
</tr>
<tr>
<td>SLS Validation</td>
<td>Errored Frame Seconds</td>
<td>Statistics</td>
<td></td>
</tr>
<tr>
<td>SLS Validation</td>
<td>Throughput at egress</td>
<td>Statistics</td>
<td></td>
</tr>
<tr>
<td>SLS Validation</td>
<td>Offered Load at ingress</td>
<td>Statistics</td>
<td></td>
</tr>
<tr>
<td>SLS Validation</td>
<td>Frame Rate at egress and ingr.</td>
<td>Statistics</td>
<td></td>
</tr>
<tr>
<td>SNMP</td>
<td>Spanning Tree bridge traps</td>
<td>Unsolicited</td>
<td>topology change, new root</td>
</tr>
<tr>
<td>SNMP</td>
<td>VLAN notification traps</td>
<td>Unsolicited</td>
<td>creation, deletion, membership</td>
</tr>
<tr>
<td>SNMP</td>
<td>RMON traps</td>
<td>Unsolicited</td>
<td>monitoring of MIB values</td>
</tr>
<tr>
<td>SNMP</td>
<td>MIB Query</td>
<td>Solicited</td>
<td>query of statistical MIBs</td>
</tr>
<tr>
<td>Syslog</td>
<td>System messages</td>
<td>Unsolicited</td>
<td>notification by Syslog service</td>
</tr>
<tr>
<td>Security</td>
<td>MAC security</td>
<td>Unsolicited</td>
<td>notification by SNMP</td>
</tr>
<tr>
<td>Security</td>
<td>Storm Control</td>
<td>Unsolicited</td>
<td>notification by SNMP</td>
</tr>
<tr>
<td>Security</td>
<td>Unauthorized access attempts</td>
<td>Unsolicited</td>
<td>notification by Syslog service</td>
</tr>
<tr>
<td>Security</td>
<td>Intrusion Detection System</td>
<td>Unsolicited</td>
<td>other – proprietary</td>
</tr>
<tr>
<td>User</td>
<td>Firewall Statistics</td>
<td>Statistics</td>
<td>SNMP or proprietary</td>
</tr>
<tr>
<td>User</td>
<td>Complaints</td>
<td>-</td>
<td>phone/mail</td>
</tr>
</tbody>
</table>

Table 2.1: Measurements and Metrics [97]
Chapter 3

Root Cause Analysis Method Modeling the Human Expert Behavior

3.1 Passive and Active Root Cause Analysis Methods

TMN (Telecommunications Management Network) was the first thorough network management architecture, described and recommended by ITU-T. TMN uses a philosophy of hierarchical and rigidly structured topology of alarm description and notification model, which is appropriate for the well controlled telecommunications systems, but rather un-flexible for the IP-based networks with various services, applications and overlays.

Internet service providers and network operators required a more flexible approach, namely root cause analysis methods able to find the ultimate cause of a fault that propagates various types of false and real alarms. Root cause analysis (RCA) is often referred to as fault isolation/localization or event/alarm correlation in the literature. In this thesis, and throughout my research fault localization means the process of identifying the place of the original fault. I refer to complex but passive filtering and correlation of the events as event correlation. Finally, I use root cause analysis as a more general term, for finding the place and the reason of the fault with utilizing both passive methods and active checking routines.

Various event correlation and root cause analysis methods appear in the literature.

- Codebook approach [41]
- Rule based reasoning [42]
• Case based reasoning [43, 44]

• Model based approaches [45, 46]

• Probabilistic approach: Bayesian networks [47, 36]

• Neural networks [48]

• Fuzzy logic [49]

• Distributed decision making [50]

These classic methods take passively collected event notifications to determine the root cause of active alarms.

There are commercial, highly scalable and flexible network and service management systems available that support event correlation and root cause analysis, too. The most well-known RCA engines are implemented in HP OpenView, IBM Tivoli NetView, EMC SMARTS InCharge and MicroMuse NetCool.

State of the art fault management systems, however, do not rely on passively collected event notifications only. At some points of the analysis, the RCA algorithm should be able to initiate active diagnostic tests. These tests can be often conducted by using the native capabilities of the network (i.e., CFM messages), there could be, however, special test equipment installed as well.

The difference of passive and active fault localization methods can be demonstrated by an example taken from human health care. A medical doctor who merely asks the patient about the problem and analyze the provided medical evidences is considered to use the "passive root cause analysis method". The other medical doctor who analyzes the previously given evidences, asks the patient, and orders a series of diagnostic checks – furthermore, based on the results, he/she would order further tests. This second doctor uses "active root cause analysis methods".

The general issues of RCA systems utilizing both active and passive approaches are detailed in [51], where the authors also describe a prototype system. A fault management system tailored for Voice over IP services are described in [52], where me and my colleagues have used an approach of utilizing active and passive fault localization methods very similar to the RCA framework described in this thesis.
The two ultimate questions in these systems are: when to initiate a test and how that test should look like. This chapter proposes a complete, flexible and scalable model based RCA method, addressing the issue of scheduling active tests as well.

3.2 Modeling Human Expert Behavior in Fault Localization

System specialists make either conceptual or ad-hoc plans to find the root cause of an alarm. They decide

- what measurements should be made,
- what should be searched for inside event logs or other databases, and
- what to do next depending on the result of these routines.

Consciously or not, experts fetch further input data (i.e., the IP address of a node, an interface number, etc.) to start some measurement or analysis processes. Simultaneous processes finish asynchronously, which forces the expert to make decisions on what to do next: what kind of further checks can be initiated using the gathered input data? Data driven architectures are actually designed to address such issues, hence they could be used to mimic the simultaneous data processing capabilities of a human specialist.

An RCA algorithm aiming to simulate human expert behavior should satisfy the following requirements:

- capability of extracting key parameters from the alarm description,
- using this data, it should initiate active diagnostic checks, search routines, and correlation processes,
- the initiation of independent checks, routines or processes should be made simultaneously,
- once a result of a check is available, it starts new routines using the new pieces of information until the root cause is found and no further checks are envisioned.
Flexibility is a further, important requirement for effective RCA systems. Model based RCA solutions model the network, their events and the connections between them. The model is flexible, so topology changes do not lead to RCA algorithm changes or rule changes. Beside the network model, the alarm model also exists in a template-manner. This means that the network entities are only referenced in the alarm model (as parameters) rather than being hard-coded in there.

Scalability issues of RCA systems arise due to the enormous number of possible event-types. In the proposed architecture the alarms arriving to the RCA system are generated based on a high number of event notifications, through an extensive alarm correlation and filtering process. The number of input alarm types can be limited by conscious typology of possible alarms, hence defining alarm type sets rather than specialized alarms for each fault.

This leads to the "best practices" to human specialists: if there is only a low number of alarm types exist, it is sensible to create separate "action plans" for each alarm type. This action plan is going to be the base for the data-driven RCA descriptions. The most well-known data driven architectures are the Petri nets. The following sections detail how does the Petri net-based RCA operate.

3.3 The Root Cause Analysis Framework

Alarm notifications arrive at the RCA module from the event preprocessing module. Figure 3.1 depicts the internal structure of the RCA framework.

The root cause analysis sub-module takes each alarm separately and unleashes the hidden fault by going through an "action plan" provided by the RCA descriptor. The diagnostic elementary checks are parametrized by data taken from the alarm description, as well as connection information gathered from the topology database. Once the fault is localized and the root cause of the alarm is found, its description is passed to the advice and modification module, which provides suggestions for corrective actions or initiates configuration changes depending on the information found in the advice-database.

The above mentioned sub-modules of this framework are detailed in the following sections.
3.3.1 Petri Net Based RCA Execution

The proposed RCA execution method schedules active checks, database searches and other processes depending on the availability of the input data of these routines.

Petri nets in this case should be considered as "action plans" to find the root cause of an alarm. Each alarm notification has its own Petri net associated with it. The passive filtering and alarm correlator modules are applied in the system in order to reduce the number of alarms and RCA executions. The event preprocessing module should merely feed critical alarm notification to the Petri net based RCA module. From the point of view of the RCA execution, alarms can be categorized in order to associate the most promising action plan with them.

The Basic Petri Net Theory

Petri net [54] can be considered as a modeling language to represent discrete distributed systems. It graphically depicts the structure of a distributed system as a directed bipartite graph with annotations.
A Petri net consists of places, transitions and directed arcs, which connect the places with transitions. A transition can have input and output places, the naming convention depends on the direction of the arc between the given place and the transition.

In general, places may contain any number of tokens – in the RCA model places can contain zero or one token. A distribution of tokens over the places of a net is called a marking. Places can contain tokens at the starting position of a Petri net – this is called the initial marking. A transition gets activated if there are tokens in its every input place. In this case the transition fires: three functions are taken during firing. The transition

1. consumes the tokens from its input places,

2. performs some processing task (which can take time), and

3. places a specified number (one in the RCA model) tokens into each of its output places.

These functions should be considered as taken in one, non-preemptible step. Due to the non-deterministic nature of the Petri net, it is well-suited for modeling human expert behavior during fault localization and root cause analysis.

Figure 3.2 depicts a simple Petri net with an example initial marking, and three execution steps. The Petri nets proposed for RCA execution work in a similar way. Although the research literature has produced various types of Petri nets, I proposed the simplest model (using Occam’s razor): there are no "weighted arches", "coloured nets" or "multiple tokened places".

3.3.2 The Petri Net Representation of RCA Action Plans

The active elementary diagnostic checks represent the transitions of the Petri net. Each elementary check has input parameters and output results. These represent the input and output places of the given transition. A token marking a place means that the data associated with that place is available. This way all those transitions (elementary checks) can fire, for which the input places are all "tokened". In other words a diagnostic check will be initiated when all required input parameters are available. Since getting the result of a measurement
a. Example Petri net with initial marking

b. Placing token to Place "1" initiates firing

c. Marking after Transition "A" has fired
d. Marking after Transition "B" has fired

Figure 3.2: Example of executing a simple Petri net

or database search takes time, there can be numerous processes belonging to one alarm-analysis running simultaneously.

There is, however, another level of parallelism included in the system. As alarms arrive at the RCA module asynchronously, there can be several alarms under investigation at a given time. As mentioned, there are different Petri net instances associated with these alarm-investigations. The task of the central scheduler module is to allow decision time and "stepping" time for each Petri net instance. During this time the arrived results are analyzed, tokens are redistributed according to the available data, and new elementary checks are initiated. If the place of the final result in a Petri net gets tokened, the investigation of the corresponding alarm is finished, and the results are passed to the Advice and Modification module.

### 3.3.3 The RCA Scheduler

Based on the alarm type, the scheduler chooses an appropriate RCA descriptor (Petri net) for scheduling the analysis. This is depicted by Figure 3.1.
As an initiation, the scheduler puts tokens to "Start Evaluation", and "Alarm Notification Parameters", showing that these data are available, so the transition "Triggering Function" can fire.

In the next step the scheduler checks all the other Petri nets – belonging to other active alarms. If there are data (results) arrived, it can start firing transitions: call appropriate elementary check routines. When it realizes, that our newborn alarm has a transition "Triggering Function", which can fire, it calls the function. As a result, the RCA method really starts rolling on: the parameters found in the alarm description are made available, so many transitions can fire.

When the scheduler realizes this, it opens new threads for each transition, and calls the belonging elementary check routine (having all required input parameters available).

3.3.4 Elementary Diagnostic Checks

Based on CFM, PM, SLS verification and other functionalities, me and my colleagues have defined a number of "elementary checks", which can be ranked into the following categories:

- **CFM** - The elementary checks are the Loopback and Link trace. Loopback is used for verification, while Link trace is for fault localization.

- **PM** - Performance management tests are defined in [28]. These include frame loss, frame delay and frame delay variation.

- **SLS** - Analyzing statistics or results of unsolicited tests against SLS of a given service instance between service endpoints. Furthermore, solicited checks can be initiated between nodes, against certain SLS criteria.

- **SNMP MIB request** - Such requests can be used for checking upon potentially suspicious statistics, misconfiguration verification, and analysis of service dependent settings, among others.

- **Event database search** is required for lookup of events in the event notification database. It is a parametric search to look up events that may relate to the root cause.
Elementary checks get logged, hence their frequent repetition can be avoided. A few elementary check examples – related to end-to-end Ethernet services – are shown in Table 3.1.

<table>
<thead>
<tr>
<th>Function</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFMloopback</td>
<td>src. &amp; dest. MEP ID</td>
<td>success/failure</td>
<td>CFM loopback test to verify the connectivity</td>
</tr>
<tr>
<td>CFMlinktrace</td>
<td>src. &amp; dest. MEP ID</td>
<td>fault location MIP</td>
<td>CFM link trace to localize the fault</td>
</tr>
<tr>
<td>PMframeloess</td>
<td>src. &amp; dest. MEP ID</td>
<td># lost frames</td>
<td>PM measurement to verify frame loss on path</td>
</tr>
<tr>
<td>PMframedelay</td>
<td>src. &amp; dest. MEP ID</td>
<td>delay value</td>
<td>PM measurement to verify frame delay</td>
</tr>
<tr>
<td>PMframedelayvar</td>
<td>src. &amp; dest. MEP ID</td>
<td>delay variation</td>
<td>PM measurement to verify frame delay var.</td>
</tr>
<tr>
<td>EventLookup</td>
<td>search criteria</td>
<td>Alarm ID</td>
<td>Search for specific events in the event database</td>
</tr>
<tr>
<td>SNMPVlanCheck</td>
<td>path, vlan ID</td>
<td>success/misconfig</td>
<td>Verification of the VLAN configuration</td>
</tr>
</tbody>
</table>

Table 3.1: Elementary check function examples for end-to-end Ethernet services

### 3.4 Case Studies

In the following I present three case studies for better understanding of the RCA execution. The first two cases are related to end-to-end Ethernet service management, whereas the third case is an example of VoIP service management.

#### 3.4.1 Case 1: Ethernet Services, Connectivity Fault

Let us consider a basic Ethernet connectivity error for the first case [97]. It may be reported by the CFM as a connectivity fault. The Petri net representation of the RCA associated to this alarm is depicted by Figure 3.3.
Figure 3.3: Petri net to drive the "CFM connectivity fault" alarm

The start of execution is triggered by a CFM alarm. The Petri net immediately branches to initiate active fault localization and passive database search. The unsolicited CFM functions may indicate the source of the error, however, it must be verified and if it persists, it must be localized. The Loopback and Link Trace functions provided by CFM are used as localization elementary checks in the upper branch of Figure 3.3. In the meantime – on the other branch – a search of a Syslog event (i.e. looking for "interface down" errors) is also initiated. If such events are found, the root cause is clearly indicated with a description of the error.

Once the exact location of the error is found by Link trace, we should check for possible error sources: VLAN misconfiguration and Spanning Tree protocol error, furthermore, if the Spanning Tree is not in reconfiguration phase. These checks can also be performed simultaneously.

Finally the results are evaluated: if appropriate Syslog notification events were found, the exact fault is described in the result (i.e. interface down). If the fault verification returned the location of the fault, the VLAN and STP check results may clarify the root cause. The final output itself is a rule based decision, providing the fault description depending on the
various test results.

3.4.2 Case 2: Ethernet Services, SLS Violation Detected

Let us consider a complex fault for the second case, namely, if an SLS is violated [97]. This case is a more complicated scenario, where many checks are required to find the real cause.

The fault is detected by the periodical SLS verification, and the RCA is triggered. SLS violation may cover higher delay, loss and packet delay variation parameters observed compared to the ones fixed in the SLS. The associated Petri net is depicted by Figure 3.4.

Firstly, the ingress and egress address pairs and the VLAN configuration of the service must be determined. Then, RCA may continue on three branches: VLAN settings checks, CFM/Performance verification and verification of denial of service attacks. The initiation of checking routines follow the problem solving steps of human experts – scheduled by the availability of the input data.

Based on the required VLAN settings information, the 802.1p priority check can be easily performed. Wrong priority setting may result in QoS degradation. Other VLAN errors would lead to connectivity loss (configuration management issue).

At the same time, a CFM Link trace can be performed to verify connectivity between the endpoints. If this function does not result in reporting an active connection between the endpoints, we can initiate further checks for connectivity failure:

- search for SNMP trap generated event notifications for discarded frames. This may indicate VLAN misconfiguration on a bridge port.

- verify VLAN to MEP mappings. If it does not comply, it results in connectivity error.

- STP check: spanning tree protocol may have an error or may be in topology change phase.

If the CFM link trace was successful, we have to evaluate the performance of the path. Thus, we search for relevant performance monitoring events and at the same time we also initiate performance checks based on Y.1731 definitions [28].

In parallel with the VLAN and CFM checks, the verification for a possible "Denial of Service" (DoS) attack (causing degraded service and SLS violation) is also initiated.

39
Then, we evaluate either the QoS, connectivity check or DoS verification results (note that only one of them will give positive result), and the RCA result is generated as a description of the most probable fault.

Figure 3.4: Petri net to drive the Root Cause Analysis of "SLS violation" alarm

3.4.3 Case 3: VoIP Services, High Packet Loss

This RCA case is taken from a complete fault management system created for VoIP service providers [52][53]. The solution features complete implementation of the event detection, event preprocessing, RCA and correction adviser modules. Regarding the RCA framework, a complete set of diagnostic checking routines have been implemented, together with a dozen of Petri nets – which cover the action plans for all the alarm types listed during the expert interview sessions.

One of the most complex RCA descriptions is related to the alarm notification of "high packet loss". Figure 3.5 depicts the corresponding Petri net. After the first execution step all the output places of the first transition get tokened, given the alarm description is defined to contain all the necessary parameters.
Figure 3.5: Petri net to drive the Root Cause Analysis of "High VoIP packet loss"

The execution takes three parallel routes:

1. "IP connectivity check" is a simple ping between the requested address-pairs. If the basic connection is working, then special utilities are triggered to check the packet drop rate, delay and delay variation characteristics of self-generated VoIP flows. The results get evaluated and the list of "bad connections" is passed to the final decision-maker function.

2. There is a search started for correlating alarms of the recent period in the alarm database. As a result, alarms with more precise fault localization parameters (i.e., a network error effecting high packet loss at the analyzed VoIP segment) can be found, supporting the RCA decision.

3. Some packet loss is usual for applications belonging to the "best effort" Class of Service (CoS). VoIP traffic should be handled with priority, hence wrong priority settings
of interim nodes could cause high packet loss for the VoIP service, as well.

The result of these diagnostic checks arrive at a specialized function, which evaluate the various results and provide a root cause description, together with the confidence level of the decision.

The simultaneous execution of elementary diagnostic checks clearly reduces the time of the RCA execution – which affects the duration of the fault correction. A study on the VoIP-tailored version of this RCA model has found that the average execution time of a complete, multi-step diagnostic process was cut to half due to the concurrent nature of the Petri net execution [55].

3.5 Conclusions

The processes of my SA framework are constructed to model the problem-solving behavior of human experts. These are sometimes realized by rule-based ordering and structuring the information (typically event detection and preprocessing), other times deep analysis is required of the environment, simultaneously with initiating and evaluating various tests (root cause analysis).

The classic root cause analysis and fault localization methods use passive event correlation and avoid the initiation of active diagnostic checks. The main questions to answer for introducing active checks in RCA systems are 1) when to initiate these checks and 2) how the actual check should be implemented. Human experts are naturally able to quickly order (schedule) the necessary diagnostic routines. The expert behavior of scheduling diagnostic checks to solve an RCA problem can be modeled with data-driven architectures. Petri net is one implementation of data-driven models.

After receiving an alarm notification, the RCA system starts with extracting key parameters from it. Based on this, elementary active checks, search routines, and correlation processes can be triggered. These tasks are scheduled depending on the input data availability. Since human experts tend to initiate independent checks, routines or processes simultaneously, a data driven RCA architecture would be the most satisfactory to model this behavior. I have used Petri nets to describe the root cause analysis steps. Petri nets allow starting new checking and searching routines depending on the availability of previous check results;
this feature is also similar to the human expert behavior while ordering check-sequences.

Due to the Petri net-based RCA description the execution time of the RCA processes can be cut to half, moreover, the execution order does not need to be determined in advance. The task of assigning checking routines to alarm types cannot be eliminated, but still this should be determined only once for each alarm-type. The checks will be executed following the connections of the Petri net, in order of the data availability.
Chapter 4

Localizing Network Bottlenecks

4.1 Passive and active approaches

The procedure of locating network bottlenecks needs complex analysis of the network properties. There are several measurement models to be used, both active and passive methods. The problem with active methods is that they introduce artificial traffic — changing the original network properties — and the measurement must be applied at several nodes to get the overall result. The most powerful passive monitoring based methods analyze data collected from every node by monitoring all their input and output links all the time. This truly gives a highly accurate result, however, it is very expensive computationally as well as financially. The method I propose is to monitor elastic traffic over a backbone link passively. The advantage of our passive method is its inexpensive feature: it monitors only one backbone link and still able to detect bottlenecks anywhere in the known network-segment. The method is only powerful if we have a certain knowledge of the network topology in addition to the measured backbone.

Passive monitoring-based bottleneck detection requires a complex collecting, pre-processing and evaluation system. Although the collection and pre-processing of the large amount of data are demanding tasks by themselves, it is also challenging to find appropriate measures for evaluating whether a link is bottleneck or not. In this chapter I introduce a novel metric, which seems to be powerful enough to distinguish traffic carried over a network bottleneck from "traffic flowing with its own pace".
There are various measures suggested by [56] and [57], each derived from transport-level flow analysis. These studies investigate several metrics (loss-rate, speed-averages, variance of flow-level throughput, delay-factor) to be used in bottleneck detection by passive monitoring, and found that they work with different accuracy under different conditions. The two most promising ones are the "X-measure" [57] (like peakedness [58], it is a coefficient of variation-type metric applied for the throughput) and the delay factor calculated from the inter-arrival times of flows (based on the M/G/R – PS model) [59, 56]. Nevertheless, a recent study [60] evaluating these metrics on live traffic showed that they are not yet accurate enough and need further fine-tuning.

Packet pair method [61, 62] is widely used for bottleneck bandwidth estimation. This technique includes, however, intrusion of active probe traffic to the network before analyzing the results [63]. My investigation focuses on non-intrusive bottleneck detection methods only. The entropy-based clustering method (introduced in [64]) seems to be powerful to find connections sharing a bottleneck; the referred paper, however, misses to provide a range for packet interarrival time entropy that suggests bottleneck behavior.

The traffic we carry out our analysis on has elastic nature. Its most important property is the guarantee for packet delivery, because lost packets are retransmitted. A typical elastic traffic is file transport (i.e., http- or ftp-download), using TCP over the Internet. The other important traffic class of today’s Internet is stream traffic, which has strict delay and/or delay variation requirements, and generally used in real-time applications such as audio- or video communication, transmitted typically by UDP over the Internet.

The metrics I describe for bottleneck link detection can be derived from data collected by passive monitoring of elastic flows. As there is no transport level retransmission for stream-traffic, it is very hard to analyze it by passive monitoring methods for these purposes. Nevertheless, in links carrying both traffic classes, stream-flows have effects on the transmission properties of the elastic flows, so analyzing elastic flows alone can after all give a realistic presentation of bottleneck behavior.

During my analysis I used a transport level flow-definition (source IP address, source port number, destination IP address and destination port number) starting with SYN and ending with FIN, RESET or (may they be lost during monitoring) a predefined, long timer. For more precise flow-definition, adaptive timeout methods (described in [65]) or an other
dynamic timeout strategy implemented in NeTraMet – the first open source implementation of an RTFM (Real Time Flow Management) system [66] – can be used.

4.2 Typical Bottleneck Properties

Making a decision on a network’s bottleneck is not an obvious task. I investigated several metrics to be used in bottleneck detection by passive monitoring, and found that they work with different accuracy under different conditions, so I recommend to use a combination of the following metrics to make the final decision more accurate.

Identification of the links with high loss rate is possible by counting retransmissions for each flow. The tighter the bottleneck is, the higher the loss rate will be. This relation is only linear if all connections are in congestion avoidance phase. Because of its additive nature, loss rate cannot be used alone as a bottleneck indicator. It could only be used in the decision of bottleneck detection, if a longer history of the aggregated traffic is known, and these values are used for calibrating a more complex, loss-based metric.

Low transmission speeds is an other property of the bottleneck. The interpretation of the average flow speed – the amount of data traversed during a certain period of time – is not suitable for bottleneck detection, because idle periods are not taken into account. Instead of the throughput-like metric, consider the definition for speed averages of flows: the actual mean of flow speeds measured on the link. It is important to understand the difference between throughput and speed averages of flows, and to realize that while the first metric is certainly inaccurate for bottleneck detection, the second is appropriate to prop up bottleneck behavior.

The Framework for IP Performance Metrics [67] defines – among other issues – the specification method for such metrics, allowing both analytical and empirical ways for defining them. The one-way delay [68] is an analytically described metric, describing the duration of the first bit of the packet being sent from the source to the last bit of the packet arriving to the destination. This should not be confused in any ways with the delay factor of flows described in this chapter. To measure the one-way delay, two reference points are required, and its measurement is related to one packet, as opposed to the delay factor, which is measured at one reference point and is related to a set of flows (each flow is a sequence of
packets). The IP packet delay variation [69] is, in short, the variation of many one-way delay measurements on the packet level, again, requiring two measurement points. Such scenario is not practical for locating bottlenecks in real networks, where the differences and skews of the node clocks brings in another level of complexity for the measurements.

In case of normal operation (if there is no severe congestion along the path), the variance of the flow-level throughput for elastic traffic could be high on a link. Different sending-rates for each flow are caused by the particular properties of TCP such as the advertised window size, RTT of the connection, loss rate along the path and other parameters. However, under bottleneck conditions I assume that elastic flows crossing the bottleneck share the bandwidth equally, thus their throughput would be similar, causing the throughput variance to be low. Also, having longer measurement periods, the variability of the throughput variance in time is expected to be especially low for bottleneck links. This is a promising, but still computationally expensive method.

Delay factor of a transport-level flow can be interpreted as the quotient of its measured sojourn time (average transmission time between source and destination) and its expected sojourn time. An other calculus for the delay factor of flows with a given size is provided by the M/G/R - Processor Sharing model (M/G/R - PS)[70]. This advanced technique simplifies the delay factor to be dependent merely on the link utilization. According to my network assumptions the M/G/R - PS model can be handled by only one server (R=1) and thus we can simplify one of the model's basic equations (see Equation 4.6).

In the following I focus only on the usability of delay-factor type metrics for bottleneck detection. The expectation is that normal links will have a delay factor around 1 and for bottleneck links this delay factor would be higher. I describe M/G/R - PS model in detail and compare its calculation-results for expected sojourn time to the measured mean sojourn time of flows with a given size.

### 4.3 Delay Factor of Flows

As mentioned in the previous section, the average delay factor of a flow with a given size can be calculated as

$$ f_S = \frac{\sum_{i=1}^{n} T_{m_i}}{n \cdot T_{ex}}, \quad (4.1) $$

48
where $T_{m}$ is the measured flow time and $T_{ex}$ is the expected sojourn time calculated for the given flow size, by the narrowest link’s capacity between the source and the measurement point. (Calculating an “ideal” sojourn time between all source/destination node-pairs is not feasible.) The calculus described above will be referred to as “sojourn time based delay factor”.

The delay factor can be derived from inter-arrival time of flows as well, with the help of $M/G/R - PS$ model. According to the model, the transmissions will be served by a number of servers ($R$), which could be derived by the following way:

$$R = \lfloor C/r_{peak} \rfloor,$$

(4.2)

where $C$ is the capacity of the specific aggregated links, and $r_{peak}$ is the maximum transfer rate of the flows, determined by the access rates of the users (e.g., modem speed). In the original fluid model, the $M/G/1 - PS$ service disciplines the server capacity being shared among the customers arrived, so every customer gets $1/X$ service capacity simultaneously – where $X$ is the number of customers on the link. A possible realization of this system can be performed in a round robin manner [71, 70]. A key property of Processor Sharing (PS) models is that the expected average sojourn time of a traffic flow does not directly depend on the distribution of the service time, only depends on the mean of this service time [72, 70]. The other important property of the fluid model is that every newly arriving customer gets service immediately. So there is no waiting time in the queue, the sojourn time of a customer, however, fluctuates because of their required different service time and the arrivals and departures of other customers. Insensitivity is a nice property of this model, which results in the same state probabilities for $M/G/R - PS$ and $M/M/R - PS$ systems [73, 72].

According to the model the expected sojourn time of elastic flows in the $M/G/R - PS$ system:

$$E\{T(x)\} = \frac{x}{r_{peak}} \left( 1 + \frac{E_2(R, R\rho)}{R(1 - \rho)} \right) = \frac{x}{r_{peak}} \cdot f_R,$$

(4.3)

where $x$ is the size of the transferred flow, $\rho$ denotes the utilization of the link, $R$ is the number of servers and $f_R$ is the delay factor. $E_2$ stands for Erlang’s second formula, which could be expressed as:

$$E_2(R, A) = \frac{A^{R} \cdot \frac{R}{R-A}}{\sum_{i=0}^{R-1} \frac{A^{i} \cdot \frac{R}{R-A}}{R-A}},$$

(4.4)
where $A = R \cdot \rho$. Utilization could be specially derived from the flow arrival rate ($\lambda_e$), the average flow size ($x_{\text{mean}}$), and the capacity ($C$) of that link [59]:

$$\rho = \frac{\lambda_e \cdot x_{\text{mean}}}{C}. \quad (4.5)$$

Unfortunately, Equation 4.3 does not present right values in our network environment for the expected sojourn time of the flows (i.e., the results do not match the average value of the measured sojourn times ($T_m$) for flows with a given size $x$). This is probably due to inhomogeneous traffic in the simulation and measurement environments, compared to traffic assumptions, where the M/G/R − PS model was successfully validated [59]. Nevertheless, I have found the value of delay factor indicates the occurrences of congestion clearly, thus the bottleneck can be discovered with it reliably: normal links will have a delay factor around 1 and for bottleneck links this delay factor would be higher.

The M/G/R − PS model can be reduced to M/G/1 − PS, if the $r_{\text{peak}}$ peak rate is larger than the aggregated link capacity [59]. In this case this assumption is satisfied since in the recent simulation environment I do not deal with the capacities of access links, assuming Local Area Networks of ISPs (Internet Service Providers) at the end of the aggregated links, so $r_{\text{peak}}$ is given. Thus the values of the maximum transfer rate are not bounded, they could extend as far as the capacities of the aggregated links. According to these assumptions the next expression − derived from Equation 4.3 − should be calculated for one server ($R = 1$) as:

$$f_R = 1 + \frac{E_2(R, R\rho)}{R(1 - \rho)}. \quad (4.6)$$

Thus Erlang’s second formula simplifies, and Equation 4.6 can be calculated as follows:

$$f_1 = 1 + \frac{\rho}{1 - \rho}. \quad (4.7)$$

This easily computable parameter shows the influence of congestion over a specific aggregated link. If its value is close to 1 for a specific link, it means the utilization is very low − not necessarily zero − so it is not a bottleneck link, while the higher its value is, the more considerable congestion there may be. It suggests that the bottleneck link could be directly distinguished by its utilization − I must stress, however, that the formula of $\rho$ (Equation 4.5), calculated by $\lambda_e$, $x_{\text{mean}}$ and $C$ − is different from the definition of utilization used by...
MIBs (Management Information Base). The exact relation of these two utilization-metrics should be studied further.

I found that this advanced metric suggests bottleneck behavior accurately, and traditional delay factor calculated by Equation 4.1 in fact fails to detect the bottleneck. Surprisingly, the traditional delay factor is usually much higher than 1, and sometimes even tight bottlenecks have lower delay factor than normal links – whatever short or long the chosen measurement periods are. The explanation is that the flow sojourn times can range on a broad scale and faster flows can not balance the effect of some extremely slow flows. After all, the average measured flow time for varied traffic can get longer than for flows with relatively slow speed but with small speed variance. This indicates that derivation of delay factor by measured and expected sojourn times is not suitable for bottleneck detection because of its high sensitivity to delay variance.

4.4 Packet Interarrival Times Distribution

This section introduces a novel metric to be used in bottleneck detection. It is based on the analysis of the probability distribution function (PDF) of packet interarrival times, which exhibit different shapes (with some common patterns) as seen during studying network bottlenecks using transport-level flow interarrival time distributions.

There is a good visual interpretation of packet interarrival times PDF patterns in [64]. The authors describe PDFs computed in scenarios with no experienced queuing, significant queuing and queued traffic influenced by cross-traffic. Spikes and "spike-trains" in the PDF are found to suggest bottlenecks – or at least significant queuing in the analyzed path.

Nevertheless, there is an important difference in naming conventions used in [64] and in my research. The referred study considered connections as "bottlenecks" where the packets experienced "significant queuing". This is a very loose definition, even though it is hard to give a firm description of network bottleneck (other definitions would be connected with loss, throughput limits, high utilization, significant delay [56]).

In my research a link is considered as "bottleneck" where packets experience continuous, severe queuing and even being dropped due to the finite queue-lengths. Let us take an other viewpoint: consider a user utilizing similar networked services on server a and on server b
(the servers offer similar processing performance). The user can reach server $a$ on route $A$, whereas he/she gets serviced by server $b$ on route $B$. In case this user is satisfied with the network performance towards server $a$, but he/she can notice performance problems towards server $b$, then route $B$ contains bottleneck link(s) – at least more of them, than scenario $A$ does.

Obviously this is not a precise definition of a bottleneck either. Finding an appropriate metric to distinguish bottleneck links from well-dimensioned ones could help clarifying the issues of having different definitions for the same underlying problem (which is ultimately reflected in user satisfaction of using networked services).

### 4.4.1 Higher Order Statistical Properties

The first and second central moment (mean and variance) of statistical distributions are widely used for briefly characterizing a distribution. Higher order statistical properties [74], as the third central moment (skewness) and the fourth central moment (kurtosis) are more rarely used in the engineering practice (although their applicability is wide-scale).

Skewness characterizes the degree of symmetry - or rather, the asymmetry - of a distribution around its mean. Positive skewness indicates a distribution with a probability-peak on the lower values and an extending tail towards the higher values. On the contrary, negative skewness indicates a distribution having a probability-peak on the higher values and an asymmetric tail extending towards the lower values.

Equation 4.8 shows the definition of skewness.

\[
\gamma_1 = \frac{E[(\xi - E(\xi))^3]}{\sigma^3},
\]

(4.8)

where $E(\cdot)$ stands for expectation, $\xi$ is the statistical variable (hence $E(\xi)$ is the mean of $\xi$) and $\sigma$ is the standard deviation. For measured data with finite number of measured entities, estimated skewness can be calculated as

\[
\gamma_1 = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \left( \frac{x_i - x_{\text{mean}}}{s^*} \right)^3,
\]

(4.9)

where $n$ is the number of entities, $x_i$ is the value of the actual item, $x_{\text{mean}}$ is the mean of the measured values of $x$ and $s^*$ is the standard deviation.
Kurtosis characterizes the relative peakedness or flatness of a distribution compared to the normal distribution. The distribution is leptokurtic (or more peaked than the standard normal distribution) if the kurtosis excess (see Equation 4.10) is positive. Negative kurtosis excess indicates a platykurtic (or relatively flat) distribution. The term "kurtosis" was first used in [75].

Equation 4.10 shows the definition of "kurtosis excess" [74], which is widely used in the practice of mathematical statistics. The outcome of this type of kurtosis is normalized for easier comparison with the normal distribution. "Kurtosis proper" is by definition the fourth central moment, and it misses the normalizing element of $-3$.

$$
\gamma_2 = \frac{E[(\xi - E(\xi))^4]}{\sigma^4} - 3. \quad (4.10)
$$

The statistical estimate of kurtosis with finite number of measured entities comes from the formula

$$
\gamma_2 = \frac{n(n-1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left( \frac{x_i - \bar{x}_{\text{mean}}}{s^*} \right)^4 - 3 \frac{(n-1)^2}{(n-2)(n-3)}. \quad (4.11)
$$

The next chapter introduces the features of PDFs derived from packet interarrival times of computer networks. We shall see that deriving skewness and kurtosis helps distinguishing bottleneck scenarios from un-congested measurement setup.

### 4.4.2 Packet Interarrival Times PDF of Links with No Congestion

The probability distribution function (PDF) of packet interarrival times (PIT) should be extremely flat in a non-queued aggregated network-link. This is because several links with various capacities can carry packets to the observed aggregated link. Independent sources generating traffic in such topology causes absolutely random interarrival times.

In case a node aggregates numerous links with various capacities (and no queuing), the PIT PDF at that node appears to be flat, as shown in Figure 4.1.a (showing probability and PIT (in microseconds) on the axes).

Once the link gets more busy, the relevant network node must queue some packets, and place them on the line right after the previous packet (back-to-back). The more of
Figure 4.1: Packet interarrival times PDF on an aggregated link with no queuing (left) and on an aggregated link, with eventual queuing (right)

This queuing is applied, the less "flat" the PDF becomes: spikes starting to appear at the interarrival times where queued packets has followed each other back-to-back. Figure 4.1.b depicts a scenario where several lower capacity links – having different peak rates – connect to an aggregation node. The packets have experienced some – not severe – queuing before arriving to the aggregated link. Skewness of such PIT distributions are close to zero, or negative, whereas their kurtosis is negative, emphasizing that these PDFs are relatively flat.

4.4.3 Effects of the Bottleneck Behavior on Interarrival Times Distribution

Theoretically what one should expect during the observation of a link getting congested is the following. As the observed link starts showing bottleneck behavior, most of the eventual spikes of the PDF gets less noticeable and the spike around the lowest possible interarrival times starts dominating the PDF. Under severe congestion this spike fully dominates the PDF, as packets arrive back-to-back during the whole measurement period. Both skewness and kurtosis exhibit more positive values. The more dominant the spike around the lower PIT values gets, the more skewed the PDF becomes. Similarly, as kurtosis is, as the definition suggests the "peakedness" or "spikeness" of the distribution [76], PIT kurtosis assumes higher values as the spike dominates the PDF. Since eventual queuing already makes the PDF skewed, one can expect that kurtosis should be more robust metric of bottleneck behavior than skewness. The following sections provide simulation results and analysis of real-life experiments to detail and support the above theory.
4.5 Performance Evaluation Based on Simulations

4.5.1 Simulation Environment

The network topology used during the simulation is shown by Figure 4.2. OPNET was chosen as a simulation tool, as previous work on bottleneck detection has proven its applicability [56]. During each simulation period, traffic was generated to traverse the network for 15 minute long measurements.

From the simulation's point of view the traffic sources were the ISP’s (aggregation nodes of Internet Service Providers). The characteristics of the traffic matches service types such as e-mail, ftp, http and database-access. There is also some asymmetry in the link capacities, since one of the ISP’s is connected to the backbone through a bottleneck link (link_2 between routers R2 and R5 in Figure 4.2).

![Network topology](image)

Figure 4.2: Network topology used during the simulations

The applied data collection and evaluation steps are as follows.

- the Probe captures traffic on an aggregated link (between R5 and R6 in Figure 4.2),
- traffic flowing from the same directions are distinguished by IP-address ranges belonging to the source ISP’s of a given direction,
- after separating the captured traffic by directions, the PTT-characteristics of each direction is analyzed.
To evaluate kurtosis and skewness as possible metrics for bottleneck detection, the bottleneck in this simulation environment has been set tighter in several steps. This way one can evaluate how much difference in utilization and also in available bandwidth (ABW) would result in a loose or tight bottleneck. The bottleneck was created in the simulation topology by defining link_2 having relatively low bandwidth, whereas link_1 and link_3 were equal, higher capacity connections.

The bottleneck was set tighter and tighter by increasing the maximum ABW of link_1 and link_3 (up to 1000 Mbps) and simultaneously decreasing the capacity of link_2 (down to 300 Mbps). The tightest bottleneck was set to be able to handle merely 30% of the traffic of the other links (loaded by the same amount of traffic). In the following this scenario will be referred to as "1000/300", suggesting the "higher/lower"-bandwidth values applied for "link_1,3/link_2".

4.5.2 Simulation and Measurement Results for Delay Factor

I had two aims during the analysis: the validation of expected sojourn time given by the M/G/R – PS model and the evaluation of the delay factor as a metric for bottleneck detection.

For thorough validation of expected sojourn time itself, I chose several values of $x$ for the flow size, but I have found that the expected sojourn time of flows calculated by Equation (4.3) does not result in realistic values for the transmission times, so in the following we rather focus on delay factor metric. To compare delay factors given by Equation (4.1), and Equation (4.7), I used data in simulation trace files. According to the M/G/R – PS model requirements, flows with a given size $x$ can only be considered. To get enough data to be processed either longer measurement periods should be used, or $x$ should be considered as a narrow range of flow sizes rather than a discrete value. I have investigated both possibilities: extending the measurement periods should make the mean-values more certain, although it requires much more time for data gathering, in which utilization of the network could vary too much, hence spoiling the results.

On the other hand using a narrow range of flow sizes around the most frequent flow size of the measurement provides better results during less time – I only present these results in here. Obviously, the range must be chosen so that it remains narrow enough (i.e., less than
±0.05x) not to violate M/G/R – PS’s requirements.

To meet the criterion about short analyzing-periods, a good approach can be to set fixed length time-intervals in which at least one flow with average-length is present (from beginning to the end) on the network. This means that each analyzing time-interval would be twice of the average sojourn time. These short intervals mean, however, that the number of flows belonging to a period is not sufficient for statistical analysis – in fact, there are not enough data collected from each link. The presented results are taken from the analysis, where the measuring interval was 4-times longer than the average measured sojourn time of the most recurrent flow size, although analysis with this factor being 8 and 16 provided very similar results.

![Graphs showing delay factor over time for different scenarios](image)

Figure 4.3: Delay factor calculated by inter-arrival times (left) and sojourn times (right) in three scenarios

Me and my colleagues analyzed the simulation- and measurement-output by several metrics including loss rate, throughput, speed averages of flows, and delay factors. In this document I present and analyze only the results for the delay factors based on inter-arrival times-
and sojourn time of flows. In general we can state that the results obtained from speed averages, variance-based and loss-based simulations give very similar results for whereabouts of the bottleneck – hence proving each other as well as the method based on inter-arrival times of flows. In contrast, sojourn time-based method provides uncertain values for bottleneck behavior.

According to my expectation the delay factor should be around 1 for non-bottleneck links and significantly higher for bottleneck links. In Figure 4.3 three scenarios can be seen for the two methods. The three graphs on the left side (Figure 4.3 a, b, c) present the delay factor values calculated by inter-arrival times of flows, while on the right side (Figure 4.3 d, e, f) can be seen the sojourn-time based results. The first scenario, in which the higher capacity links were 1000 kbit/s and the bottleneck was 300 kbit/s, will be referred to as "1000/300", is the tightest bottleneck scenario. The others will be referred to as 900/400 and 675/635, respectively. Each diagram has three curves (link_1, 2 and 3), representing the delay factor of the analyzed links connected to router R5 in Figure 4.2, where link_2 is always the bottleneck with lower capacity, while link_1 and 3 are the others with higher, but the same capacities. The delay factors are plotted in function of the elapsed time in seconds. In Figure 4.3 a and 4.3 d the bottleneck has 300 kbit/s and the other links have 1000 kbit/s capacity. The most frequent flow size was 2012 Bytes and flows between 1962 and 2062 Bytes were taken into account during the whole measurement – as suggested earlier. Comparison of the diagrams shows that the bottleneck behavior is easily noticeable in Figure 4.3 a: the delay factor value is floating around 2 on bottleneck, while faster links – link_1 and link_3 – have low utilization, thus delay factors are around 1. In contrast, d shows the narrow link having low delay factor compared to the faster links, which apparently have extremely high values. These results are inconsistent with my natural expectations for the range of delay factor.

Considering Figure 4.3 b the delay factor of the bottleneck link decreases as the difference of the link capacities lessens (as the bottleneck gets more loose). Figure 4.3 e shows the obvious failure of sojourn-time based method again: the bottleneck link has very low delay factor comparing to the non-bottleneck links.

In Figure 4.3 c the capacities of the links are very close to each other, in this case our method suggests no bottleneck – all delay factors are nearly 1 – although all the values are
slightly higher than in the previous scenarios. In Figure 4.3 f a link_3 still has significantly high value for delay factor, while the bottleneck-link and link_1 have similar, low values – detecting the bottleneck behavior absolutely wrong.

An important result of our studies is that Equation 4.3 is failed to be accurate in calculating the expected sojourning time for varied traffic; its delay factor, however, is a powerful measure in bottleneck detection.

### 4.5.3 Common Patterns of PIT PDFs

In order to validate the theory about PIT PDF structures, an exhaustive study was carried out using different source traffic characteristics, topologies and utilizations (the latter has ranged from scenario 675/625 to scenario 1000/300). Typical results for the two extreme scenarios are plotted in Figures 4.4.a-f.

The somewhat normal condition can be described as "packets experience queuing, but no loss" (scenario 700/600). The other extreme is a real bottleneck condition (scenario 1000/300).

In order to validate the theory about PIT PDF structures, an exhaustive study was carried out using different source traffic characteristics, topologies and utilizations (the latter has ranged from scenario 675/625 to scenario 1000/300). Typical results for the two extreme scenarios are plotted in Figures 4.4.a-f.

The somewhat normal condition can be described as "packets experience queuing, but no loss" (scenario 700/600). The other extreme is a real bottleneck condition (scenario 1000/300).

Comparing the resulting PDFs, there are numerous observations that can be made:

- the PDF keeps its general shape (number and place of the spikes) in a given link, as long as the same type of traffic flows through it,

- great spikes appearing in the PDF do not necessarily hide bottleneck condition,

- if the highest spike is not positioned at the lower PIT values, the packets are not following each other back-to-back (hence there is no severe congestion, see Figures 4.4.c.f),
Figure 4.4: Packet interarrival time PDFs in simulation scenarios 700/600 (left) and 1000/300 (right)

- the PDF loses some of its peakedness if the available bandwidth increases (compare Figures 4.4.a and d),
- as expected, the PDF gets extremely peaked and skewed under bottleneck conditions (see Figure 4.4.e),
- as expected, the spike at the lower PIT values are dominant under bottleneck conditions (see Figure 4.4.e).

Unfortunately, drawing conclusions from the comparison of plotted diagrams is not a feasible network maintenance practice. Even in a medium-sized network the high number of network connections makes such comparisons impossible. A more decent metric is needed.
that is able to distinguish PDFs of congested connections from other cases without significant queuing. As anticipated, skewness and kurtosis of PITs should be able to suggest whether a severe bottleneck exists or not. The following section evaluates the usability of these higher order statistical properties in bottleneck detection.

The statistical properties skewness and kurtosis of PITs were calculated using Equation 4.9 and Equation 4.11 respectively. During the simulations the same network topology (see Figure 4.2.) were loaded by similar source traffic. The only major difference between the simulation scenarios is the ABW assigned to links 1, 2 and 3 in Figure 4.2.

Table 4.1. provides skewness values of the three links for the different bottleneck scenarios. The following observations can be made by evaluating the results:

- as expected, skewness of a bottleneck link is positive,
- the tighter the bottleneck is, the more positive skewness is observed,
- skewness assumes positive values also for links with eventual queuing, hence it is difficult to distinguish these from severely congested links,
- connections with less traffic and almost no queuing can be characterized with negative or close to zero skewness value.

<table>
<thead>
<tr>
<th>Table 4.1: PIT skewness values in simulated environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>link_1</td>
</tr>
<tr>
<td>link_2</td>
</tr>
<tr>
<td>link_3</td>
</tr>
</tbody>
</table>

To conclude, skewness of PITs noticeably acquires positive values in bottleneck scenarios, although it may do so for links with eventual queuing.

Table 4.2. summarizes PIT kurtosis values observed in different simulation scenarios. There are merely two positive values appearing among the results, both observed on the bottleneck link in serious bottleneck scenarios (900/400 and 1000/300). It is also noticeable that the PIT distribution gets more platykurtic (flat) as ABW increases (link_1) and
changes its platykurtic character into leptokurtic (peaked) as ABW decreases as far as causing bottleneck behavior \((\text{link}_2)\). There is a less practical, but interesting feature of PIT kurtosis, namely, that it does not assume noticeably lower (more negative) values if the capacity of an underutilized link increases \((\text{link}_3)\).

<table>
<thead>
<tr>
<th>Name</th>
<th>675/625</th>
<th>700/600</th>
<th>800/500</th>
<th>900/400</th>
<th>1000/300</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{link}_1)</td>
<td>-0.96502</td>
<td>-0.87263</td>
<td>-1.04085</td>
<td>-1.23243</td>
<td>-1.45336</td>
</tr>
<tr>
<td>(\text{link}_2)</td>
<td>-0.59994</td>
<td>-0.35450</td>
<td>-0.28027</td>
<td>0.71819</td>
<td>1.31911</td>
</tr>
<tr>
<td>(\text{link}_3)</td>
<td>-1.62645</td>
<td>-1.63841</td>
<td>-1.63470</td>
<td>-1.63417</td>
<td>-1.61983</td>
</tr>
</tbody>
</table>

To summarize, PIT kurtosis seems to be an appropriate metric for detecting bottlenecks. This is further validated in the following section.

4.6 Evaluating Bottleneck Metrics by Real Measurements

4.6.1 Real Measurement Results for Delay Factor

To validate the usability of the inter-arrival time-based delay factor in bottleneck-link detection, me and my colleagues carried out some measurements at a local operator as well, by capturing continuous traffic at a router exchanging data at the rate of 640 Mbit/sec (64% utilization). Having only limited information about the monitored network, I processed the routing table of the measured router and used a simplified, one-hop measurement topology. Altogether we collected and analyzed data from 11 links connected to internal domains of the operator. I validated two methods based on inter-arrival time and speed averages of flows described earlier.

As we had limited memory allowance for the devices used at the measurement, the interval for capturing continuous traffic was limited as well. The measurement period was defined as the whole time of the measurement. In this way there was only one simple value for each link and for each method. Since there were no more detailed information on the operator’s network topology available, we can say there were some bottleneck behavior in two directions, where the delay factor values were 1.44 and 1.12 – these were the links routed towards MLLs.
(Managed Leased Lines). All the other links transmit data without any congestion, their
delay factor values were very close to 1.

4.6.2 Using PIT Kurtosis to Detect Bottlenecks in Real, Operational Networks

In order to validate metrics for passive bottleneck detection, a series of measurements have
been taken place at sites of a major Hungarian network operator. During the measurements
a passive network monitoring tool (Network Associates’ Sniffer and Snifferbook Ultra) have
been used, capturing continuous data traversed on Gigabit Ethernet interfaces. The mea-
surements covered normal and busy hour conditions, connections to the operator’s Internet
Data Center and to routers/switches at the edge of the core network. The measurements
were taken at several sites in Budapest, their continuous length was limited by the equipment
used, ranging from 5 minutes to 30 minutes. The conditions and the measurement results
are sensitive, hence only a portion of these are described here.

The measurements captured traffic during severe bottleneck conditions also. The topol-
yogy of the network segment that contained a bottleneck link during the measurements can be
studied in Figure 4.5. Traffic is flowing from the Digital Subscriber Line Access Multiplexers
(DSLAM) towards the core network (and back), going through internal ATM (STM-1) links.
The network monitoring tool has been connected to the aggregated link, and captured all
packet headers of the measurement period. As part of the data-processing, packet-headers
were sorted by direction (ATM sub-network). The moments of packet interarrival time dis-
tribution were calculated for each direction. Most of the measurements have been carried
out under normal network conditions. During one of the monitoring sessions, an ATM link
loading the $OpR_{-1}$ - $OpSW$ segment got overloaded. (this fact has been indicated by other
network analysis tools as well).

Skewness analysis resulted positive values for all cases. The bottleneck link provided
higher PIT skewness value comparing to the others, but still the difference was not significant,
leaving skewness to be unable to detect bottleneck by itself.

The results of PIT kurtosis calculations are demonstrated in Table 4.3. These kurtosis
values clearly meet the expectations set up earlier. Traffic measured under normal network
Figure 4.5: Measurement topology at the operator’s site

conditions (underutilized links) provide negative values; although $OpR_1 - OpSW$ shows kurtosis close to zero, suggesting some uncertainty. During the congestion on this route (remember that not the monitored link, but an ATM link, being two hops behind the monitoring unit has been congested), kurtosis reached the positive domain, suggesting severe bottleneck condition. In this period noticeable packet loss and over 90% link utilization was observed as well (using monitoring tools covering the targeted, lower capacity links).

Table 4.3: PIT kurtosis values of measurements at the operator’s site

<table>
<thead>
<tr>
<th>Link name</th>
<th>normal conditions</th>
<th>$OpR_1$ overloaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>$OpR_0 - OpSW$</td>
<td>-0.46420</td>
<td>-0.45571</td>
</tr>
<tr>
<td>$OpR_1 - OpSW$</td>
<td>-0.06267</td>
<td>1.15119</td>
</tr>
<tr>
<td>$OpR_2 - OpSW$</td>
<td>-0.11180</td>
<td>-0.13289</td>
</tr>
</tbody>
</table>

The PIT PDFs calculated for the three connected links under normal and high traffic conditions are shown in Figure 4.6.a-c and Figure 4.6.d-f respectively. After analyzing these PDFs based on the previously described assumptions, the following observations can be made. Figures 4.6.a, c and d suggest healthy traffic flow, with a normal amount of queuing considering the high aggregation level. Figure 4.6.b includes a noticeable spike, however it is found at the higher PIT values, which means there were no severe congestion. The visual comparison of Figures 4.6.e and f does not reveal any major difference, as in both figures the PDF spikes at the lowest possible PITs, although the spike is significantly higher at the real
bottleneck case (Figure 4.6.e).

\[\begin{array}{cc}
\text{a. } OpR_0 \text{ direction, normal cond.} & \text{d. } OpR_0 \text{ direction, higher traffic} \\
\text{b. } OpR_1 \text{ direction, normal cond.} & \text{e. } OpR_1 \text{ direction, high traffic} \\
\text{c. } OpR_2 \text{ direction, normal cond.} & \text{f. } OpR_2 \text{ direction, higher traffic}
\end{array}\]

Figure 4.6: Packet interarrival time PDFs calculated based on real network data

In fact, kurtosis – as a metric for bottleneck behavior – has clearly distinguished these situations as well. It provided the positive result of 1.15119 for the bottleneck case visualized by Figure 4.6.e as opposed to the otherwise normal traffic condition shown in Figure 4.6.f, where kurtosis was calculated to be −0.13289 (see Table 4.3).

During the various simulations and measurements it was noticed that PIT kurtosis calculated on the full PIT distribution depends slightly on link capacity. When two links have the same utilization (in percentage), the link having the higher capacity appear to have higher kurtosis. The reason behind this observation is not clearly identified yet. The consequences of this behavior, however, can lead far. In an extreme scenario the kurtosis of a higher capacity, less utilized connection can be higher than the kurtosis of a lower capacity, more
utilized link. This would weaken the accuracy of PIT kurtosis as a bottleneck metric.

To overcome the above obstacle, a correction of PIT kurtosis should be considered for bottleneck detection. This correction should be linked to the analyzed data volume somehow.

Current analysis shows that PIT kurtosis scales well and provides more accurate, less capacity-dependent results when it is calculated on a subset of the PIT-distribution. Leaving the 10 percent tail out of the analysis and calculating kurtosis up to the 90 percentile point of the PIT-distribution have provided satisfactory results on the available data sets. Future work should verify this observation, and clarify the issue of the slight capacity-dependency of PIT kurtosis.

4.7 Conclusions

In order to detect network bottlenecks, I suggest to monitor the backbone link passively. This measurement method eases the compilation of data collected from numerous nodes, avoids error occurrences during clock-synchronization and unlike active methods, it does not introduce artificial traffic. I focused on delay factor-type properties of network bottlenecks, and derived an advanced metric based on inter-arrival times of flows. $M/G/R - PS$ model gives an estimation of the expected sojourn time for a given flow size. My measurement results showed that this model is not accurate, as its result for expected sojourn time does not match the value for mean measured sojourn time for the given flow size. Nevertheless, the delay factor calculated by inter-arrival times of flows is found to be powerful for bottleneck detection. I have studied the metric’s behavior on different bottleneck scenarios, and found that it is well able to suggest the whereabouts of bottlenecks up to the level of a link being approximately 90% tighter than others. As the severity of the bottleneck behavior and the nature of its causes influence the suggested metrics differently, decision about a link being a bottleneck is best supported by the combination of methods based on delay-factor, loss rate, flow speed averages and throughput variance.

Bottleneck detection based on passive measurements can be supported by analyzing Packet interarrival time (PIT) distribution. The more packets experience queuing at a network node, the more of them leave the node back-to-back. Under normal conditions the PIT probability distribution function (PDF) is relatively "flat": spikes due to typical packet
lengths and minimal following times are visible, but appear to be small. As queuing turns into severe congestion, the spike around the lowest possible PIT value gets dominant. This fact is indicated by the fourth central moment – kurtosis –, too: it gets more positive. Beside this, the third central moment, skewness gets more and more positive, too. Distributions that are more flat than normal distribution have negative kurtosis, whereas peaked distributions characterize themselves with positive kurtosis. Applying this to packet interarrival times distributions, positive kurtosis suggests serious bottleneck behavior, while negative kurtosis is a property of underutilized links. Values being very close to zero is hard to evaluate, but probably hide serious queuing in the path.

Both skewness and kurtosis appeared to acquire higher values as the available bandwidth of a link decreased in the OPNET-based simulation environment. While both measures performed well as relative metrics, only kurtosis is powerful enough to distinguish bottleneck links from underutilized connections. Current studies show that kurtosis is slightly dependent on capacity, hence the metric should be refined for more accurate bottleneck detection: kurtosis should be calculated up to the 90 percentile of the PIT distribution.

Analysis of real measurement data has also been carried out, supporting that PIT kurtosis can be a powerful metric of detecting bottlenecks. The wider usability of PIT kurtosis in network performance analysis is for further study.
Chapter 5

Service Quality from the End–user’s Perspective: Quality of Experience

Describing service quality from the end-users’ point of view is fulfilled by using Quality of Experience (QoE) metrics. In a networking environment QoE metrics are very close to Quality of Service (QoS), but – because of subjective nature of QoE – they could not be derived directly from QoS. Collection and analysis of QoS and SLS (Service Level Specification) properties of networking services are daily tasks of the operators. User satisfaction, however, cannot be mapped directly to these measures related to a network path or aggregated access traffic. My ultimate aim is to find methods and metrics that correlate properly with QoE, and could be measured passively on aggregated network links [77] [78].

In this chapter I present some experimental results on correlating the severity of network overload and the experienced service quality by using our advanced metrics on bottleneck detection. The measurements were carried out in a real network environment by myself with supporting colleagues, in which we loaded the network with various kinds of service requests and made notes on the perceived quality. We captured packet level traffic, and I derived metrics based on packet interarrival times, packet size information and packet loss information.

My results show that the proposed bottleneck-detection metrics correlate much more to QoE than any common QoS metrics, thus could be used successfully as indicators of user satisfaction.
5.1 Introduction

QoE metrics are similar to QoS metrics, except the fact that end-user experience is subjective in nature. Moreover, QoE is also influenced by the access capabilities of end users and the used service path. The focus of this research is to determine the correlation between end-user experience and measurable traffic properties and also to find new appropriate metrics with high correlation. The following chapters provide experimental results of correlating usual QoS metrics as well as metrics of bottleneck severity with user perception of service quality.

There can be several scenarios where the experienced service quality becomes less than satisfactory. The roughest QoE degradation is the unavailability of a service. This actually happens more regularly than one might expect. Examples of temporal and/or regular service-unavailability include an unsuccessful paying procedure at a webshop, a broken download, or a "webpage unreachable" message [79, 80]. These cases are very hard to measure from a central site, since continuous application monitoring and data processing is not always feasible. Another type of QoE violation is tied to network QoS, hence QoE metrics usually can be correlated with some kind of QoS metrics. Network-related QoE-degradation can manifest itself as extended response time, decreased download speed, less enjoyable audio and video streaming. This is usually caused by one or more network bottlenecks, where packets get congested, queued, then delayed or dropped.

5.2 Overview of the Metrics Chosen for Determining QoE

There can be several link properties applied to characterize the presence of bottlenecks in packet data networks [56]. Since we carry out these measurements on aggregated links (rather than at the user’s end-terminal), we focus on measures that characterize mass traffic. The metrics that we analyze are throughput, properties of packet interarrival times, packet size information and packet loss information [81] [82].

5.2.1 Throughput

Throughput of the users’s access link may give valuable information about the provided networking service for applications such as ftp or peer-to-peer download. On an aggregated
link, however, throughput looses its QoE-like informative nature, since the throughput of such a link is an accumulated value. Various uncontrollable factors – including link capacity, number of users, active service-requests, average packet size, application distribution and many more – influences throughput, which makes it unsuitable for being an absolute metric for QoE. Nevertheless, comparing throughput of various links at different times could provide information of the network status, which can be correlated with user satisfaction.

5.2.2 Loss Based Metrics

Increased network loss clearly has an impact on QoE. In case of multimedia traffic the loss above a certain percent – depending of the traffic type – is noticeable, and shows service degradation. For TCP-like interactive traffic, however, a small loss-ratio is normal. This is resulting from the TCP protocol behavior, since TCP uses losses for indicating congestions. Nevertheless a higher loss ratio is noticeable, and the user can detect service degradation if:

- he/she does not get the accustomed bandwidth (possibly also specified in SLA),
- the interactive traffic gets slow or erratic.

A user who is only browsing and using e-mail will merely notice service degradation if his/her downloads require increased time or connections get dropped. A user making huge downloads or using peer-to-peer software will notice the decrease in available bandwidth, but will not necessarily notice increased delay. In today's typical network scenarios, however, user traffic is limited in the first mile. The full available bandwidth may be used up by TCP connections efficiently, at the price of a small percent of loss. If there is another bottleneck in the network, TCP connections crossing this bottleneck will suffer an additional loss percent, which may be used as measure for QoE. We found that in such situations the place of this additional bottleneck can be determined – at least to the extent of the bottleneck is situated 'in front’ or 'behind’ the measurement point. On the contrary we also found that the volume of service degradation cannot clearly be determined by means of analyzing various types of TCP-losses, regardless of the granularity of the time-lag. In the following chapters I provide some of the measurements results on this issue as well.

In [83, 84] the authors describe various types of retransmissions, making clear that not all retransmissions are really caused by packet loss. They found that there can be applications
where some retransmissions are caused by special types of errors (i.e., some radio channel misbehavior of GPRS networks may cause spurious timeouts). This paper, however, also states that analyzing retransmission-volume can be useful when done in a focused way, but misleading when applied in general.

To sum up, it is difficult to use the number of losses as a QoE measure. A very small increase in loss may introduce severe bandwidth decrease. Furthermore, it is difficult to estimate the number of active connections and their average RTT – this fact also makes the loss-based measures inaccurate.

5.2.3 Kurtosis of Packet Interarrival Times

Monitoring a network link passively and capturing the packets with correct timestamps would give the operator plenty of data to analyze. One way to detect bottlenecks is to order the packets by their arriving time, and analyze their interarrival times. Our investigations showed that the 4th central moment (kurtosis) of the packet interarrival times (PIT) probability distribution function (PDF) could provide valuable information about existence of bottlenecks and their severity [85].

The theory behind this observation is the following (see the earlier chapter on Packet Interarrival Times distribution). In case a node aggregates numerous links with various capacities (and there are actually no queuing), the PIT PDF at that node should appear to be flat. This is due to the fact that packets are travelling on network links with absolutely random interarrival times. Once the link gets busier, the relevant network node must queue some packets, and place them on the line right after the previous packet (back-to-back). The more of this queuing is applied, the less "flat" the PDF becomes: spikes starting to appear at the interarrival times where queued packets has followed each other back-to-back [64]. Generally it can be said, that if there is a narrow aggregated link on a path in front of that some queuing occurs, the shape of the PDF of interarrival times of packets will suggest it: the PDF will have spike(s) at typical time values.

There are a number of different queuing cases with corresponding PDF patterns are discussed in [64]. As a rule of the thumb we found that the examined link is congested if the first spike of the PDF is the highest (Figure 5.1.a), while the other lower spikes indicate queuing somewhere in front of that link (Figure 5.1.b). This kind of "spikeness" of the packet
interarrival times PDF can be characterized by its 4th central moment (kurtosis). My studies show that PITT kurtosis value is positive for traffic that crosses bottleneck links, and negative for traffic flowing with its own pace [86].

![Graph](image.png)

Figure 5.1: Typical packet interarrival probability density functions

### 5.2.4 Delay Factor Calculus

The end-user often decides a network-service unusable if he/she experiences long delays. By definition, delay factor (DF) could be calculated as the ratio of the ideal and the measured sojourn times ("travelling times") of a traffic flow that traversing on a network path.

Delay factor, however, can be derived from the interarrival time of flows as well, with the help of M/G/R-PS arrival model. According to the model, the transmissions will be served by a number of servers \( R \), which could be derived in the following way \( R = [C/r_{peak}] \), where \( C \) is the capacity of the specific aggregated links, and \( r_{peak} \) is the maximum transfer rate of the flows, determined by the access rates of the users. According to the assumptions and experiences (detailed in [70]), calculating the delay factor for one server \( (R = 1) \) still gives satisfactory result. The calculation of the M/G/R-PS-based delay factor then simplifies to: \( f_1 = \frac{1}{1-\rho} \), where \( \rho \) stands for the link utilization of the measured link. This could be specially derived from the flow arrival rate \( (\lambda) \), the average flow size \( (x_{mean}) \), and the capacity \( (C) \) of that link: \( \rho = \frac{\lambda x_{mean}}{C} \).

According to the original definition of delay factor in a non-congested situation the delay factor is around one (the ideal and the measured sojourn time is roughly equal), and reaches higher values if there is some congestion. Our flows supporting delay factor estimation (DFest) share the following attributes [87]: flows are identified merely by their source...
and target IP addresses, have a maximum size limit, furthermore have a limitation on the interarrival times of the packets they contain. Flows can be broken due to finished capture. Some flows started close to the end of the capture are not finished, hence usually undersized.

We have found delay factor results correlating with user perception of service quality in the overutilized measurement scenarios described in the following.

5.3 Measurement Environment

A configurable DSL-like environment has been set up in a segment of our university department, where the department staff was the suffering object of our QoE testing. Altogether about 50 end-user machines were involved in the test. We have degraded the usual 100 Mbps Internet access to 1 Mbps DSL-like lines. Moreover, we have introduced an artificial bottleneck for the department segment, which was changed from 7 Mbps to 2.5 Mbps in six steps. Each bottleneck scenario lasted 30 minutes, during which we have captured the traffic at two measurement points, and made notes on our experiences as common network application users. Figure 5.2 depicts the measurement scenario.

![Measurement Architecture Diagram]

Figure 5.2: Measurement architecture with demonstrating the available bandwidth degradation of download volumes after the two bottlenecks

During the test we have extensively used various networking applications and services. Applications included audio and video streaming, web-browsing, ftp and peer-to-peer downloads, Internet-telephony (Skype), database access and many more. Our experiences in
function of the bottleneck-severity are summarized in Table 5.1.

<table>
<thead>
<tr>
<th>Available Bandwidth</th>
<th>Video stream</th>
<th>Audio stream</th>
<th>Skype</th>
<th>P2P traffic</th>
<th>Web browsing</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0 Mbps</td>
<td>perfect</td>
<td>perfect</td>
<td>perfect</td>
<td>33-355 kbps</td>
<td>good</td>
</tr>
<tr>
<td>5.0 Mbps</td>
<td>occasional squares in picture</td>
<td>perfect</td>
<td>perfect</td>
<td>33-300 kbps</td>
<td>good</td>
</tr>
<tr>
<td>4.5 Mbps</td>
<td>playing pauses at key-frame changes</td>
<td>perfect</td>
<td>perfect</td>
<td>33-88 kbps</td>
<td>slow</td>
</tr>
<tr>
<td>4.0 Mbps</td>
<td>sec-long pauses at key-frame changes</td>
<td>perfect</td>
<td>perfect</td>
<td>12-14 kbps</td>
<td>slow</td>
</tr>
<tr>
<td>3.5 Mbps</td>
<td>bad; squares and pauses are regular</td>
<td>short pauses</td>
<td>scratching</td>
<td>3-10 kbps</td>
<td>unbearable</td>
</tr>
<tr>
<td>2.5 Mbps</td>
<td>very bad; not enjoyable</td>
<td>longer pauses</td>
<td>scratching</td>
<td>3 kbps</td>
<td>extremely bad</td>
</tr>
</tbody>
</table>

Table 5.1: Perceived quality of network services at different bottleneck scenarios

5.4 Measurement Results

This section presents results of a number of metrics commonly used in QoS context (throughput and losses) as well as the metrics introduced for bottleneck detection (PTT kurtosis and delay factor). We have analyzed these measures in different time scales, and from different points of view. The presented results are computed on the packet level data arriving from the hub placed between the two bridges (consider Figure 5.2).

Figure 5.3.a and b depict the measured packet-level throughput and the delay factor values in function of the elapsed time (seconds). The curves are broken because packet capture was stopped during the reconfiguration of network parameters from one scenario to the other.

Considering the graphs one could see that the more severe the bottleneck is, the higher value the delay factor becomes. In spite of that the measured throughput reflects only the amount of user-generated traffic, but there are no information about the load of the aggregated network link. Throughput changes, however explain the huge jumps of the delay factor values during e.g., the 4.5 Mbps period.
Figure 5.3: Varying of network performance metrics during the whole measurement

Figure 5.4 provides aggregated measurement values on the same traffic. The horizontal axes in the graphs show the ABW for the seven scenarios, while the vertical axes are the values of the specific metrics.

The average delay factor values are presented in Figure 5.4.a. The metric reflects the degree of congestion almost in all cases (100 to 3.5) properly, although at 2.5 Mbps ABW the relatively lower average delay factor values does not show the real situation: actually the network was halted. This forced the users to give up using the internet - causing the drop of DF.

Average PIT kurtosis values for specific ABW scenarios are depicted by Figure 5.4.b. As we described in [86], PIT kurtosis should be below zero in uncongested situations, while positive values suggest certain heavy loads. As one could expect the lower the ABWs are, the higher the kurtosis values become. The metric suggests properly the more and more congested situations, showing positive correlation with the service quality experienced by
the users. Nevertheless in the last three scenarios there could be no data presented because the artificial bottleneck realization (CBQ queuing method in a Linux-based bridge) biased the shape of the PIT PDF, which led to false kurtosis values. Figure 5.4.c presents the measured throughput in Mbps, calculated as the ratio of the sum of the transferred bits and the length of the whole measurement interval. According to this graph we can assume that in the first two cases the network link was under-utilized (below 50%) – of course this average hides the occasional higher throughput values.

Figure 5.4.d shows the average packet loss ratio (PLR). As in my previous works, I have found that the calculation of an average packet loss ratio cannot reflect the severity of the congestions. In the presented real case the relative PLR was the highest when the link was least of all used (100 Mbps ABW).

![Figure 5.4: Average of bottleneck metrics for the whole measurement periods](image)

Figure 5.5.a and b present the number of resent packets (used for loss calculation) in function of time, calculated for each 10 ms. Unlike the general expectation – there should be sporadic, high values of retransmissions in the congested case – the graphs show no significant difference. TCP packet loss type metrics and retransmission counts may suggest some network misbehavior only in very special cases, but in general these measures do not
provide really useful information, even if it is analyzed for short time periods.

![Graph](image)

Figure 5.5: Number of TCP retransmissions for two available bandwidth scenarios

5.5 Conclusions

This chapter discussed a number of metrics that could be appropriate for estimation of user perceived network properties, the so-called Quality of Experience based on packet-level measurements on aggregated network links. It was shown that throughput and loss measures fail to show correlation with QoE, whereas delay factor and PIT kurtosis could be used as passive measurement metrics to estimate some end-to-end QoE measures.

Many QoS-demanding application is used in best-effort manner on the Internet. User satisfaction primary depends on the operation of applications that they use. Therefore it is important to evaluate the QoE parameters that suggest user satisfaction for different types of traffic scenarios separately – from which the quality of specific application operation could be estimated.

The aim for my research in the near future is to examine the performance of VoIP, P2P, Video Streaming, and general best effort traffic properties in case of low network resources, and also the effects of these traffics on each other. My target is to find appropriate metrics to show correlation with end-to-end QoE measures for the different traffic types and the correspondent applications.
Chapter 6

Connecting the Knowledge Plane with the SA Framework

The presented Fault Management- and Performance Management-oriented approach of the introduced Service Assurance framework hide the fact that the SA framework can actually be used as a practical implementation of certain autonomous networking concepts. One of the philosophies that answer the issues raised by the requirements of self-management, self-configuration and self-optimization is the Knowledge Plane.

6.1 The Knowledge Plane Concept

The optimization of network and service resources and the maximization of end-user experience are not necessarily conflicting terms. The reason for the belief of these are conflicting lies in the fact that current network operators and service providers lack of up-to-date, usable information on their traffic. The questions of "how much" of "what" actually are traversed on the various network segments, where is that traffic "originated from" and where is it "distributed toward" are rarely answered. According to the main argument of [88], the users and the operators suffer from the lack of a serious, purposeful optimization effort in the traditional Internet. The transparent core has no knowledge about the data transported, and even if the intelligent edge nodes realize that there is a problem, the core might not be aware of what should be done. The low-level decisions (at the edge) are rarely relate to the higher-level goal (of the core). On the user side this results in meeting the service level
agreement only in coarse granularity: it is measured in long periods and more at a network level, rather than on a per-service basis.

The solution for gaining knowledge about network status and traffic characteristics is to gather and process such data, which then provide a basis to trigger corrective actions. The authors of [88] suggest to handle this knowledge in the Knowledge Plane (KP), an abstract entity that completes a triad together with Data Plane and Control Plane (see Figure 6.1.

![Diagram of Network Planes](image)

**Figure 6.1:** Functions of the Knowledge Plane and its connections to Control and Data Planes

In the original KPlane concept, the input is taken by sensors and the output is given by actuators. A practical variation of this architecture, detailed in [89], splits the KPlane into monitoring plane and knowledge plane. The separation of those is an obvious step: the actual "network monitoring units" (sensors) that capture and pre-process traffic data represent the "monitoring plane", similarly as depicted by Figure 6.1. There are further variations and additions to this architecture; we will review these in the section of Related Work, together with a short review of decision making methodologies and practical examples from the field.

Figure 6.1. depicts the relation between the Knowledge, Control, and Data Planes. The

80
probes/sensors take data from both the control and data planes, and report pre-processed information for the status processing module, where further analysis takes place. The actuator in the model is the decision maker module, which provides triggers for the control plane, completing the self-management cycle.

The main source of knowledge is the actual traffic of the Control and Data Planes. Although some traffic characteristics can be gathered by analyzing the Control Plane messages, many important applications such as Peer-to-Peer (P2P) downloads, Video Streaming, or interactive voice hide their control messages, hence their identification is only possible through Deep Packet Inspection (DPI) of the traversed traffic. The aim of Traffic Mix analysis is to determine the distribution of volumes for services and applications utilizing the network. Similarly, Traffic Matrix analysis provides results about traffic volumes and if possible, further characteristics broken down by route directions. The second part of this chapter discusses the proposed, unique method of Traffic Mix and Traffic Matrix analysis.

The IST-MUSE project resulted in many ideas and implementations in relation to KPPlane. Beside separating the Monitoring Plane from the KPPlane in [89], they further introduced the Action Plane (APlane). They also defined a Knowledge Base that is commonly reachable by KPPlane, MPlane and APlane. Figure 6.2 depicts their connection and relation to the network.

Figure 6.2: The relation of the Monitor, Knowledge and Action Planes to the Network
It is clear that the concept of the split Knowledge Plane is widely used in various levels of network and service management. Nevertheless, general traffic analysis is not yet utilized in order to support decision making in the KPlane. In the following theses (i) the relation between the elements of the SA framework and the split KPlane is described, (i) a scalable traffic analysis concept is introduced for the MPlane, and (iii) the development details of extracting valuable information for the traffic mix and the traffic matrix are also revealed.

6.2 Related Work

Kim et.al. summarizes the research and development ideas and efforts in management of the Future Internet in [90], specifically reviewing the research activity in the EU, USA and Korea. The authors emphasize the common interest and importance of measurements, monitoring, knowledge representation and reasoning. The original idea of introducing a higher level intelligence to the core about its traffic and general status first appeared in [88], where Clark et.al. introduced the concept of the Knowledge Plane. Besides providing very clear motivations, this groundbreaking paper suggested to solve networking issues by using methods devised in the field of Artificial Intelligence (AI). Since then, experts of both area Network Management, and AI elaborated various versions of the KPlane concept in great depth.

Li described a layered architecture in [91], where NetKP the network layer organizes agents to gather and provide valuable information to the higher-level entities, specKPs, which handle and act upon their own interest, i.e., routing optimization or intrusion detection. Another variant of splitting is suggested by [92], motivated by the need to get to the kernel of self-functions defined by autonomous networking drives. Hence, the processes in KPlane are based on two loops: a collaborative loop and an adaptation loop. The KPlane itself include a knowledge base, a reasoning engine, a knowledge sharing process, and a machine learning process. In this model, Monitoring functions remain outside the KPlane.

Dietterich et.al. found the the application of distributed, model-based reasoning agents is a feasible and successful approach for certain fault diagnostics tasks that involve the KPlane. In their report (see [93]) one of the main motivations was to involve Machine Learning in KPlane. Although their findings show that these methods can contribute to the KPlane,
they do not suggest to have machine learning as a key element of KPlane. Their paper also includes interesting reports on fault detection case studies, including DNS diagnosis, and a scenario where a typo in BGP (Border Gateway Protocol) tables was revealed.

The IST-MUSE project resulted in many ideas and implementations in relation to KPlane. Beside separating the Monitoring Plane from the KPlane in [89], [94], and [95], they further introduced the Action Plane (APlane). They also defined a knowledge Base that is commonly reachable by KPlane, MPlane and APlane. Figure 6.2 depicts their connection and relation to the network. The main motivation in these papers is to eliminate QoS (Quality of Service) and QoE (Quality of Experience) issues in the access network for VoIP, IPTV and other multimedia services. Instead of gathering knowledge from overall data plane traffic, these papers rely on designated protocols (i.e. RTP, Real-time Transport Protocol) and protocol analysis of the control messages.

The Monitor Plane is extensively used in [96] as well, where a complete, access control list-based VoIP service management system is described and evaluated. The KPlane in this paper is put in a different context: its functionalities include Call Data Record generation and visualization.

Although KPlane was not mentioned in [52] all of its features appear in the service management framework described in the paper: measurements, monitoring, data processing/mining, decision making, knowledge bases and machine learning. The presented framework has been effectively used for fault detection and elimination for Ethernet services and for VoIP services [97] as well.

A specialized KPlane is suggested in [98] in order to handle current QoS problems with protection routing algorithms in GMPLS over WDM (Generalized Multiprotocol Label Switching over Wavelength Division Multiplexing) networks. This is a clear example of using a variation of the KPlane concept to enhance concrete routing methods speed and effectiveness.

It is clear that the concept of Knowledge Plane is widely used in various levels of network and service management. Nevertheless, general traffic analysis is not yet utilized in order to support decision making in the KPlane. In the following sections we describe the suggested management architecture, traffic analysis concept and two methods to extract valuable information about the traffic mix and the traffic matrix.
6.3 Corresponding Functions of the SA Framework and the KPlane Concept

The input of the SA framework is traffic and status information, whereas its output is advice of modification or the actual execution of modification steps. The same applies for the Knowledge Plane concept.

The functions of the MPlane corresponds to the Event Notification Collector subsystem, its interfaces, and to the Event Preprocessing subsystem. The Root Cause Analysis and Data Miner subsystems are practical implementations of the actual KPlane functionalities. The APlane tasks correspond to the Advice and Modification subsystem and its interfaces. The Knowledge Base of the SA framework is the Event Notification Database and the Topology Database.

6.4 The Monitor Plane

In this thesis the architecture suggested in [95] (see Figure 6.2) is followed, especially examining the functions and requirements of the Monitor Plane. This function is crystallized at the original definition of autonomous networks, in [99], defining the foursome of "Monitor-Analyze-Plan-Execute" (MAPE) functions. The core function of the MPlane is to provide complete and detailed view of the network and its services. Probes at every element (access nodes, routers, switches, content servers, links, etc.) monitor the element status as well as traffic parameters.

Although built-in probe modules seem convenient, passive probing is more desirable to gather traffic/related information. Active network elements (such as routers or switches) keep their processing priorities for their main job, occasionally leaving the Knowledge Base without detailed information. These occasions of degradation in the status reporting function happen at the worst time from the KPlane’s point-of-view for practical reasons. It gets degraded at the time when the element is getting overloaded. Coincidentally, such detailed reports of overloading would be the most beneficial for the KPlane. This is why passive probing is more desirable to gather information on these elements.

After capturing the raw data, processed, grouped, and filtered traffic information gets
inserted into the Knowledge Base by the probes. Both packet- and flow-level analysis reveal important characteristics on losses, packet delays, and packet delay variations in the traffic, routing specialties, network structure changes and violations of the SLS (Service-Level Specification).

This thesis focuses on gathering these characteristics by passive monitoring. The following subsections briefly describe the basic requirements and mechanisms enabling this method.

6.4.1 The Advantage of Monitoring at Highly Aggregated Connections

The inevitable function of the network monitoring probes is catching, filtering, and pre-processing the traffic. These tasks should be completed for the whole network. Since installing and maintaining such a monitoring network could be an enormous effort for the operator, introducing the MPlane at the highest aggregation parts (i.e., monitoring the fastest links) can be a good decision. Monitoring these relatively few points allows gathering all packets that traverse the network, although some locally looping traffic could be left out of the analysis.

6.4.2 Distributed Processing

Depending on the traffic volume, and the depth of the analysis, detailed traffic analysis can be carried out by one or many processors. In order to keep up with the ever increasing traffic and the demand for complex analysis, the processing system must be distributed in order to support the requirement of high scalability. This distribution has two levels: at the first level the Capture probes receive the packets, time-stamp, filter them, and based on the distribution law in the given environment, pass their essential information (i.e., the truncated headers) to the distributed processing units called Monitors.

The time-stamped, filtered, truncated packets must be processed by the Monitors in order to reveal network and service statuses. After the analysis, traffic information must be inserted into the Knowledge Base by the Monitors.

Depending on the traffic volume, and the depth of the analysis, the processing jobs can
be fed into one or many processors. In order to keep up with the ever increasing traffic and the demand for complex analysis, the processing system must be highly scalable.

In cases where on-the-fly, complex analysis is required on highly utilized links, the SCALOPES C-board is a highly scalable solution. This equipment is closely related to my research, and has been developed during the ARTEMIS SCALOPES project [100], with the cooperation of engineers from AITIA and BME. It is a standalone, FPGA-based hardware, equipped with 2x 10 Gbps Ethernet interfaces and 16x 1 Gbps Ethernet interfaces. When used as part of the Monitor plane, it is also preprocessing the packets, but rather than passing their data to one CPU, it distributes them among many monitor units through its 1 Gbps Ethernet Interface. The standalone Monitor Units then carry out traffic analysis, and present the results to the knowledge base. Figure 6.3 depicts such a scenario. Detailed description of this system can be found in [100].

![Diagram](image)

Figure 6.3: A scalable solution for Traffic Analysis of high-speed network links

The distinct analysis tasks such as flow separation, application identification, QoS-related parameter calculation per flow/application/route are managed by separated modules, so the parallel tasks can be run on distinct processors in the same time. Moreover, the inactive modules can be turned off to save power.
6.4.3 Essential Functionalities of Capture Probes

- Creating timestamps for the packets. Time-stamping done by hardware (firmware) facilitates much more precision than by software, since it avoids possible latencies due to the operating system;

- Filtering on hardware level. High-speed traffic (i.e., currently 10 Gbps or above) presently allows no option for on-the-fly filtering in software. Clearly defined, low level filters are very useful: they can dramatically decrease the data to be analyzed;

- Truncating incoming packets. For the majority of the network analysis functions, statistics-counting, or fingerprint analysis, it is not necessary to use the whole IP packet, but the first portion of it;

- Encapsulation and presentation of preprocessed data. The resulting, digested packet information must be structured and packed when passed over to the Monitors.

The tasks of the Monitors in this architecture are the following:

- collection and decoding all the incoming information continuously (in 7/24 manner);

- checking filtering rules predefined by the network operator, execute conditional controlled orders/commands (i.e., conditional packet saving, alarming);

- structured data storage (i.e., raw data, statistics, assays, alerts);

- generation of packet- and flow-level counters on volume, loss, delay, packet delay variation;

- generation of specialized traffic reports, such as traffic mix and traffic matrix;

- database handling, remote access/query (i.e., Remote Capture, Session/Flow Trace).

6.5 Traffic Matrix

Traffic Matrix is a network planning and development tool. During Traffic Matrix analysis, basic QoS statistics are periodically created on flow-level, and matched to originating and
destination routes, network segments, or endpoint pairs (such as IP address(-range) pairs, MPLS tunnel endpoints, etc.). The first step of the analysis is determining the flows by an n-tuple (i.e., "5-tuple": from-IP, to-IP, from-port, to-port, protocol), and building/refreshing the flow-database. Once the targeted data structure is clarified, the algorithms of Traffic Matrix calculation are of low complexity. Such algorithms are described in [100]. The result of the measurement can be used to display periodical statistics that support network planning or service marketing activities.

The actual Traffic Matrix can easily contain endpoint-pairs in the magnitude of 10e5. It is challenging to display such huge amount of data in a way that humans understand. While the raw results should be made available for reference in the Knowledge Base, some kind of data grouping should also be applied for visual presentation. One example of a good solution is to group the matrix elements into network segments, based on their destination addresses. The aim of the grouping algorithm is never to display high, invisible amount of segments (e.g. more than 15). When the operator wishes to peek inside a segment's statistics, he/she get it displayed as a deeper layer of the matrix. This way the calculated QoS parameters show up in an aggregated manner in the segment-to-segment relation. If the system allows manual definition of segment-creation rules, operators can gather valuable information by grouping their endpoints into various segments.

6.6 Traffic Mix

Traffic mix analysis is the classification of traffic flows into application types, and then evaluating these for the service parameters important for the given application type. Flows are classified by means of statistical indicators and, if necessary, behavior heuristics. Currently operators are mostly interested in the determination of the application mix for video stream, video conference, audio stream, VoIP, and Peer-to-peer. (P2P) An application belonging to a traffic-class can be identified by using static identifiers (e.g. port-based), dynamic identifiers (e.g. changing ports, fingerprints) or by applying packet-level, temporal and spatial statistics-based evaluation methods. In my traffic analysis practice, the powerful identification methods for VoIP, video and P2P applications are described in [101], [102], and [103] respectively. We used these methods successfully during the CELTIC TIGER2 project see
Once a traffic flow is identified (i.e., based on 5-tuple), various metrics are calculated in order to help identifying the traffic-class. These metrics are the following:

- **throughput** - transferred data bytes per second;

- **packet loss** - the rate of received packets and total transmitted packets in a given time interval, or during the connection;

- **packet delay** - depending on the network topology and link load it takes a certain amount of time to receive a packet after it was sent; one-way packet delay is defined in [68];

- **packet delay variation** - network load is not always static: as conditions and usage changes over time, packet delay changes as well - this is called packet delay variation [68];

- **round-trip-time** - interactive applications require fast replies, which can be characterized with this parameter,

- **out of order/duplicated packets.**

Figure 6.4 depicts a partial result of one of our measurement at a major ISP. It visualizes the number of parallel VoIP sessions (upper diagram) and the traffic volume (in kbps). The different kind of VoIP traffic are represented with different colors, which are - from bottom to top - a) Skype over UDP, end-to-end; b) Skype over UDP, end-to-office; c) other type of VoIP, d) Skype over TCP.

## 6.7 Conclusions

Network efficiency and service quality are required to be kept at high standards for both the network operators and the users point of view. This can be achieved by keeping the network and service status under continuous monitoring. When inefficiencies become evident, or failures appear, corrective actions should be orchestrated. A recent concept to cover the autonomous loop of "Monitor-Analyze-Plan-Execute" (MAPE) is to utilize a Monitor Plane
Figure 6.4: VoIP Portion visualization of a Traffic Mix analysis

to gather and process information, introduce a Knowledge Plane to continuously process network and service status according to the requirements, and carry out commands for corrective actions by Action Plane entities. In this chapter I closely examined the tasks of the Monitor Plane, and suggested a scalable architecture to gather and process network traffic in a distributed manner. Since decisions at the Knowledge Plane should be partially based on traffic information, two important traffic analysis methods has been introduced to support decision making. Traffic Mix analysis requires a flow-based approach, where flows get classified into application types based on their characteristics, and then evaluated by related QoS metrics. Traffic Matrix analysis is important for both network and service planning, since it outputs the traffic volumes and characteristics correlated with the traffic endpoints. This information can efficiently support status processing and decision making at the KPlane, since currently these are the most sophisticated traffic-related analysis methods that human experts use during network/service evaluation and planning.
Chapter 7

Summary

My dissertation focuses on network and service quality measurement methods and metrics, and the utilization of the analysis results in network and service management.

This chapter includes my theses in their essential form, and a brief description of their application in practice.

7.1 Summary of the Theses

This section contains a list of the thesis groups and their sub-theses, briefly summarizing my research topics, my findings, and the architectures, methods, and metrics I created and validated.

The full description of these theses can be found in parts of this dissertation, and, in a more structured way, in the summary of my theses [105].

---

**Thesis 1** - Created and validated a unique, integrated service assurance framework for packet switched networks, which considers all types of incoming events for various networking service types; clearly defines and separates filtering, event correlation and root cause analysis functions; introduces a minimal set of event filtering methods that provides effective alarm generation [52] [97] [53]

**Thesis 1.1** - Defined a general and complete service assurance framework that covers event notification, preprocessing, root cause analysis, data mining and decision making func-
tionalities, which is general in the sense that it can be applied to any networking service, and complete so it detects and pro-actively supports the elimination of any service assurance-related issue.

**Thesis 1.2** - Created an event processing algorithm, which utilizes event correlation, event filtering, and root cause analysis functions and efficiently splits them – in order to populate events by correlation, allow filtering from a set of events with maximized population, and use root cause analysis merely for alarms rather than for events.

**Thesis 1.3** - Researched event filtering sets and algorithms and defined a small set of rule-based filters – consisting merely of *counters, redundancy filters, dominance filters* and *suppress function* – that, when applied together with event correlation, significantly decreases the number of generated false alarms while allowing the valid alarms to get through towards the RCA.

**Thesis 2** - Developed a new data-driven, parallelized, Petri net-based Root Cause Analysis method and applied it to various cases of service assurance [52] [97]

**Thesis 2.1** - Modeled the behavior of the human networking expert when searching for the root cause of a fault, and found that Petri-nets can be used for the representation of the parallelized action plan used by the human expert. [52] [97]

**Thesis 2.2** - Created and validated a unique data-driven, parallelized, Root Cause Analysis method that models the human expert behavior, and utilized Petri-nets for *actuating* the active RCA steps, as opposed to the well known practice of using them for *describing* event propagation in EC. [52] [97]

**Thesis 2.3** - Created completely new, Petri net-based Root Cause Analysis descriptors for Ethernet OAM services and Voice over IP OAM services, which validate the feasibility and the practical usability of the earlier described Petri-net-based RCA-method. [52] [97] [53]

**Thesis 3** - Defined and validated a new bottleneck-detection method and a unique metric, which are much more effective than others used by passive measurements on aggregated links

**Thesis 3.1** - Defined a packet-level metric called "PTT-kurtosis" for bottleneck detection by passive measurements, and showed that other metrics earlier introduced to detect network
bottlenecks are less effective for such purpose that PIT-kurtosis. [85] [86]

**Thesis 3.2** - Validated the effectiveness of the PIT-kurtosis metric through simulations and measurements in real, operational links at ISPs. [85] [86]

**Thesis 3.3** - Created an analysis method to detect network bottlenecks at segments by passively measuring a further placed, aggregated link. [56] [60]

**Thesis 3.4** - Created and validated a new bottleneck-detection metric called delay factor, based on the M/G/R–PS arrival model. [56] [87]

**Thesis 3.5** - Compared PIT-kurtosis to other possible bottleneck-detection metrics by passive measurements and found it the most effective. [81] [77]

**Thesis 3.6** - Described the relation of network bottlenecks and the Quality of Experience (QoE) and showed that high PIT-kurtosis values reveal poor QoE. [78] [81] [77] [82]

**Thesis 4** - Defined specialized monitoring services and researched new, effective traffic analysis methods – including Traffic Mix and Traffic Matrix – to serve the Knowledge Plane of autonomous networks [106] [107]

**Thesis 4.1** - Described the connection between the new service assurance framework and the current autonomous networking concepts of MPlane, KPlane, and APlane.

**Thesis 4.2** - Defined a highly scalable, distributed data capture and analysis architecture for the MPlane, to be used even in high-speed, core network connections.

**Thesis 4.3** - Researched, chosen and applied efficient algorithms to gather and present traffic mix and traffic matrix analysis data for the KPlane.

### 7.2 Results Applied in Practice

The Service Assurance Framework, which is described in Thesis 1, was applied and implemented as a prototype for VoIP by Ericsson, Kovax and BME-TMIT as part of the IKTA-00092-2002 program. The framework was generalized afterwards, and applied for multiprovider Ethernet services, as well. This latter work was part of the IST-MUSE project, an integrated research initiative, involving "almost all major players in Europe in the area of broadband access".

The Petri net based RCA method, which is described in Thesis 2, is a major component of the SA Framework. It was used similarly in the IKTA and IST-MUSE programs, both
for finding the root cause of VoIP service degradations and of multi-operator Ethernet connectivity faults and SLA violations. As part of their application, the routines for Petri-net execution were implemented, various Petri-net descriptors had to be developed and configured (to fit the actual RCA routine), and, naturally, all the customized elementary check routines related to VoIP and end-to-end Ethernet faults had to be implemented.

The passive monitoring-based method for bottleneck detection, which is described in Thesis 3, was first applied in BME-TMIT and in the NTT-DoCoMo research lab, in parallel. While simulations show the effectiveness of the M/G/R-PS-based delay factor metric, real life were not always convincing. Later, when techniques based on PIT skewness and most importantly, PIT kurtosis were developed and implemented, they were applied to reveal real-life network bottleneck detection issues. PIT kurtosis was found very effective during these set of measurements and analyses. Furthermore, the bottleneck detection method and the calculation algorithm of PIT kurtosis got integrated into the SGA1GEM, which is a network QoS testing environment with Gigabit Ethernet capabilities. This product was developed during the Jedlik research and development program, NKFP2-00015/2005.

An implemented example of the distributed Traffic Analysis concept at the MPlane includes the SCALOPES C-board (as Capture probe) and distributed Monitors. In this concept, the distinct analysis tasks – such as flow separation, application identification, QoS-related parameter calculation per flow/application/route – are managed by separated modules, so the parallel tasks can be run on distinct processors in the same time. Moreover, the inactive modules can be turned off to save power. The C-Board has been developed as part of the ARTEMIS SCALOPES research project, and it is a standalone, FPGA-based hardware, equipped with 2x 10 Gbps Ethernet interfaces and 16x 1 Gbps Ethernet interfaces. When used as part of the Monitor plane, it is also preprocessing the packets, but rather than passing their data to one CPU, it distributes them among many Monitor units through its 1 Gbps Ethernet Interface. The standalone Monitors then carry out traffic analysis, and present the results to the knowledge base. This traffic analysis includes traffic mix and traffic matrix calculations, which are also described in Thesis 4. This concept is also applied in the EUREKA project CELTIC TIGER2, where the correlated management of IP, GMPLS and Ethernet procedures are supported by decisions at the Knowledge Plane, which uses the traffic analysis results provided by the Monitor Plane.
Bibliography


J. Flick, RFC 3635 - Definitions of Managed Objects for the Ethernet-like Interface Types, IETF Std. RFC 3635, September 2003.

E. Bell, A. Smith, P. Langille, A. Rijhsinghani, and K. McCloghrie, RFC 2674 - Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions, IETF Std. RFC 2674, August 1999.


Ethernet Services Definitions - Phase 1, Metro Ethernet Forum, Technical Specification MEF 6, June 2004.


