




Project Number:	IST-1999-10077
Project Title:	 Adaptive Resource Control for QoS Using an IP-based Layered Architecture
Deliverable Type:	PU – public

Deliverable Number:	IST-1999-10077-WP2.3-SPU-2303-PU-R/b1
Contractual Date of Delivery to the CEC:	December 31, 2001
Actual Date of Delivery to the CEC:	December 21, 2001: version b0 March 9, 2003: version b1
Title of Deliverable:	Report on the development of measurement utilities for the second trial
Workpackage contributing to the Deliverable:	WP 2.3
Nature of the Deliverable:	R – Report
Editor:	Felix Strohmeier (SPU)
Author(s):	Heinz Dörken, Cornelius Heidemann, Joachim Mende, Ralf Widera (DTA); Tero Kilkanen (ELI); Bernhard Hechenleitner, Martin Herfurt, Ulrich Hofmann, Felix Strohmeier (SPU)

Abstract:	<p>This deliverable describes the enhancements of the measurement utilities, enhanced traffic generators on application level and the interface to network information from routers and how they can be exploited in the second trial. It also describes methods how the measurement results can be analysed for the evaluation and validation of the AQUILA QoS architecture.</p>
Keyword List:	AQUILA, IST, IP, QoS, measurements

Executive Summary

This deliverable summarises the results regarding the measurement utilities of the first trial of AQUILA, states the goals for the second trial and describes the enhancements for the second trial.

The main feedback for the measurement utilities from the first trial was:

- System integration has to be simplified
- Graphical user interface (GUI) should be enhanced
- More sophisticated load generators for measurement flows are needed
- Number of simultaneously running flows must be increased

Out of this feedback and because of the development of the AQUILA architecture for the second trial, the following goals are targeted for the second trial:

- Easier system integration of the components at the trial sites
- Simplification and improvement in the functionality of the GUI
- Provide support for MBAC
- Implementation of new load generator models for more flexibility in load generation
- Automatic signalling to the AQUILA QoS architecture to perform reservations for measurement flows
- Provide tools for the analysis of measurement results

To reach these goals, the distributed measurement architecture (DMA) designed for the first trial and its components have been enhanced for the second trial.

Useful measurement components from other projects will be integrated to enhance the second trial with

- additional application-like load generators
- passive measurements for the validation of MBAC

Table of Contents

1	INTRODUCTION.....	9
2	CONCLUSIONS OF THE FIRST TRIAL	10
2.1	SUMMARY OF THE RESULTS OF THE FIRST TRIAL	10
2.2	GOALS FOR THE SECOND TRIAL	11
3	ENHANCEMENTS FOR THE SECOND TRIAL.....	12
3.1	MBAC SUPPORT & MBAC VALIDATION	13
3.2	MEASUREMENT CONTROL WITH QoS HYPOTHESIS TESTING.....	13
3.2.1	QoS Mean Value Estimation.....	14
3.2.2	QoS Gradient Estimation	16
3.2.3	Hypothesis Testing between Two Loss Rates.....	16
3.3	DMA GUI.....	17
4	DMA COMPONENTS FOR THE SECOND TRIAL	19
4.1	SYNTHETIC FLOW GENERATOR.....	20
4.1.1	Management Process for Application-like Measurement Flows (MPa).....	20
4.1.1.1	Resource Reservation	20
4.1.1.2	Performance Enhancements.....	22
4.1.2	Application-like Measurement Agent (MAa).....	22
4.1.2.1	Time Synchronisation.....	22
4.1.2.2	Minimum Packet Inter-departure Time.....	22
4.1.2.3	Result Aggregation.....	23
4.1.2.4	Traffic Traces	24
4.1.2.5	General Traffic Sources	24
4.1.2.6	Path Discovery	25
4.1.2.7	Windows Version	25
4.2	ACTIVE NETWORK PROBING TOOL.....	25
4.2.1	Measurement Agent (MAp).....	25

4.2.1.1	Additional Load by Active Probing.....	25
4.2.1.2	Aggregated Flow Load Generator.....	26
4.2.2	Master (MPp).....	27
4.2.3	Measurement Agent CONTroller (MACON)	28
4.3	ROUTER QOS MONITORING (RM).....	30
4.3.1	The Launcher Component	30
4.3.2	The Edge Router Monitor (DMAAgent).....	30
4.3.3	The Core Router Monitor.....	30
4.3.4	Interface to the AQUILA RCL.....	30
4.3.5	Monitoring Influence to the Network	31
4.4	MEASUREMENT DATABASE	31
4.4.1	Database Model	31
4.4.2	Description of Changes	32
4.4.2.1	Entity “user”	32
4.4.2.2	Entity “test”	32
4.4.2.3	Entity “flow”	33
4.4.2.4	Entity “hop”	33
4.4.2.5	Entity “packet”	33
4.4.2.6	Entity “reservation”	34
4.4.2.7	Entity “state”	34
4.5	GRAPHICAL USER INTERFACE.....	34
4.5.1	Navigation	34
4.5.2	User	35
4.5.3	Quick Start	36
4.5.4	Advanced Configuration	37
4.5.5	Multiple Flow Generator.....	39
4.5.6	Start/Stop.....	40

4.5.7	Monitoring	40
4.5.7.1	Hop Status Monitoring.....	40
4.5.7.2	Flow Monitoring	41
4.5.8	Flow Search Function.....	41
4.5.9	Measurement Results	42
4.5.9.1	Result Browsing by Tests	43
4.5.9.2	Result Browsing by Flows.....	44
4.5.9.3	Results of Router Monitoring.....	44
4.5.10	Outlook for the Second Trial.....	44
5	APPLICATION-LEVEL LOAD GENERATORS	45
5.1	VOICE LOAD GENERATOR AND PERCEPTUAL EVALUATION	45
5.1.1	Functionality	45
5.1.2	Perceptual Evaluation.....	46
5.1.3	System requirements.....	47
5.2	VIRTUAL WEB USER.....	48
5.2.1	Architecture	48
5.2.1.1	Information Server.....	49
5.2.1.2	Data Server	49
5.2.1.3	Streaming Server	49
5.2.1.4	Client	49
5.2.2	Functionality	49
5.2.3	System Requirements	50
6	MBAC VALIDATION BY PASSIVE MEASUREMENTS	51
6.1	MOTIVATION	51
6.2	ARCHITECTURE	53
7	STATE OF THE ART.....	55
7.1	ONE-WAY ACTIVE MEASUREMENT PROTOCOL (OWDP).....	55

7.2	ROUND-TRIP DELAY VERSUS ONE-WAY DELAY	56
8	ABBREVIATIONS.....	59
9	REFERENCES	60
10	ANNEX A: ACCURATE ESTIMATION OF THE LOSS RATE.....	63
10.1	INDEPENDENT LOSS EVENTS $P(\text{LOSS}=P)$	63
10.1.1	Calculation of the Number of Measurements for $\alpha\%$ Right Side Confidence Interval.....	63
10.1.2	Calculation of the Number of Measurements for $\alpha\%$ Left Side Confidence Interval.....	65
10.2	DEPENDENT LOSS EVENTS.....	66
11	ANNEX B: HYPOTHESIS TESTING BETWEEN TWO LOSS RATES	67
11.1	MAXIMUM LIKELIHOOD ESTIMATION OF THE CROSSING OF QOS LEVELS.....	67
11.1.1	Simulation Example of the Maximum Likelihood Approach	68
11.1.1.1	QoS Change Detection with Active Monitoring	69
11.1.1.2	QoS Change Detection with Passive Monitoring.....	71
11.1.2	Maximum Likelihood Estimation for Aggregated Event Information.....	72
11.1.2.1	Algorithm of Papantoni-Kazakos.....	72
11.1.2.2	General Approach for Aggregated Measurements	74
11.2	SEQUENTIAL TESTING.....	75
11.2.1	Simulation Example of the Sequential Testing Approach.....	76
11.2.2	Conclusions.....	80
12	ANNEX C: DETAILS OF THE DAG CAPTURE CARD.....	81
12.1	DAG CAPTURE CARD.....	81
12.2	TECHNICAL DESCRIPTION.....	81
12.3	FRAME AND TRACE FORMAT.....	82
12.4	SYSTEM REQUIREMENTS.....	83

Table of Figures

FIGURE 3-1: AQUILA DISTRIBUTED MEASUREMENT ARCHITECTURE.....	12
--	----

FIGURE 3-2: CONVERGENCE OF MEAN ESTIMATION	14
FIGURE 4-1: COMPONENTS OF THE DMA	19
FIGURE 4-2: PACKET INTER-DEPARTURE TIME.....	23
FIGURE 4-3: RESULT DIFFERENTIATION	23
FIGURE 4-4: EXAMPLES OF GENERATED PARETO TRAFFIC	27
FIGURE 4-5: STRUCTURE OF THE "MASTER" SOFTWARE MODULE	28
FIGURE 4-6: FULLY MESHED MEASUREMENT WITH MACON	29
FIGURE 4-7: MACON-SCREENSHOT	29
FIGURE 4-8: CONCEPTUAL MEASUREMENT DATABASE MODEL.....	32
FIGURE 4-9: DMA NAVIGATION MENU.....	35
FIGURE 4-10: MANAGING DMA USERS.....	36
FIGURE 4-11: DMA QUICK START	37
FIGURE 4-12: ADVANCED CONFIGURATION – SELECTING TRAFFIC	38
FIGURE 4-13: ADVANCED CONFIGURATION – MODIFYING TRAFFIC	39
FIGURE 4-14: START / STOP OF PROBING FLOWS.....	40
FIGURE 4-15: HOP STATUS MONITORING.	40
FIGURE 4-16: EVENTLOG.....	41
FIGURE 4-17 STRUCTURE OF FLOW SEARCH FUNCTION	42
FIGURE 4-18: SCREENSHOT OF DMA RESULTS.....	43
FIGURE 5-1: PROCESS OF A H.323 CONNECTION	46
FIGURE 5-2: VIRTUAL WEB USER ARCHITECTURE	48
FIGURE 6-1 TWO APPROACHES FOR MEASUREMENTS TO SUPPORT MBAC.....	51
FIGURE 6-2: SINGLE FLOW TIME SCALES	52
FIGURE 6-3: SINGLE ON/OFF FLOW	52
FIGURE 6-4: AGGREGATED ON/OFF SOURCES.....	53
FIGURE 6-5: PASSIVE MEASUREMENTS FOR MBAC VALIDATION.....	53
FIGURE 7-1: OWDP ARCHITECTURE	56
FIGURE 7-2: MEASUREMENT CONFIGURATION	56
FIGURE 7-3: MEASUREMENT RESULTS.....	57
FIGURE 7-4: ONE-WAY DELAY (HIGH-RESOLUTION VIEW)	58
FIGURE 10-1: LEFT AND RIGHT SIDE INTERVALS OF TARGET AND TOLERANCE LOSS RATES.....	63
FIGURE 10-2: PROBABILITY FUNCTION	64
FIGURE 10-3: DISTRIBUTION FUNCTION	64
FIGURE 10-4: LEFT SIDE 5% CONFIDENCE INTERVAL OVERLAPS WITH RIGHT SIDE INTERVALS.....	65
FIGURE 10-5: DECREASED 5% LEFT SIDE OVERLAPPING.....	66
FIGURE 11-1: GENERAL STRUCTURE OF THE CONTROL PROCESS.....	67

FIGURE 11-2: SIMULATION MODEL	69
FIGURE 11-3: RATE OF INPUT FLOW AND LOSS EVENTS OF PROBING FLOW	70
FIGURE 11-4: DETECTION OF QoS CHANGES; S_k : MLR MONITOR FUNCTION GOOD \rightarrow BAD.....	70
FIGURE 11-5: DOUBLE-SIDED ALGORITHM; S_k : GOOD \rightarrow BAD MONITOR, T_k : BAD \rightarrow GOOD MONITOR.....	71
FIGURE 11-6: ANALYSING ROUTER MIB INFORMATION CONCERNING LOSSES.....	72
FIGURE 11-7: EXAMPLE LOG-COMPLEMENTARY DISTRIBUTION FUNCTION OF DURATION OF LOSS-FREE EPOCHS.....	76
FIGURE 11-8: SEQUENTIAL TESTING.....	77
FIGURE 11-9: MEAN SAMPLING NUMBER FOR H1 HYPOTHESIS TESTING $H_0: P = P_0 = 0.00001$; $H_1: P = P_1 = 0.001$	78
FIGURE 11-10: α -FUNCTION: $\alpha(10^{-5}, 10^{-5}) = 0.5$; $\alpha(10^{-5}, 10^{-3}) = 0.05$	78
FIGURE 11-11: REDUCED NUMBER OF SAMPLES $E1_KORR$ FOR $\alpha(P_0, P_1)$, $P_0 = 10^{-5}$	79
FIGURE 11-12: SIMULATION RESULTS: NUMBER OF NECESSARY SAMPLES FOR HYPOTHESIS ADMISSION	79
FIGURE 12-1: DAG CARD COMPONENTS.....	81
FIGURE 12-2: DAG FRAME FORMAT	82
FIGURE 12-3: DAG CARD TRACE FORMAT	83

Table of Tables

TABLE 3-1: MEAN VALUE ESTIMATION.....	15
TABLE 3-2: DIFFERENTIATION BETWEEN CROSS POINT ESTIMATION AND SEQUENTIAL TESTING	17
TABLE 4-1 MEASUREMENT PROCEDURE WITH RESERVATION REQUEST	21
TABLE 4-2: ADDITIONAL LOAD BY ACTIVE PROBING.....	26
TABLE 5-1: MOS TESTING.....	47
TABLE 5-2: OVERVIEW OF METHODS FOR SPEECH QUALITY MEASUREMENT	47
TABLE 10-1: α -INTERVALS FOR DIFFERENT LOSS PROBABILITIES AND SAMPLE SIZES.....	65
TABLE 12-1: DAG PC SYSTEM REQUIREMENTS.....	84

1 Introduction

To enable validation and evaluation and to support the operation of the QoS architecture developed within the AQUILA project [D1202][D1302], test and measurement tools for the AQUILA trial are developed. For the first trial a distributed measurement architecture (DMA) has been designed, which integrates three different measurement techniques (active network probing, passive monitoring and synthetic flow generation) by using the same measurement information database [D2301]. For the second trial this architecture has been enhanced, the tools of the DMA have been extended and also additional load generators are provided to generate application-like flows and aggregated flow loads.

This deliverable describes the DMA and the prototypical measurement utilities and tools provided for the second trial of the AQUILA project with a limited number of hosts. Main focus of the deliverable is to provide an update of its predecessor [D2301] for the second trial.

The AQUILA architecture for the second trial has new approaches and now e.g. supports also mechanisms for measurement based admission control (MBAC) in addition to the declarative based admission control (DBAC) as it was used in the first trial. To fulfil this task measurement methods are required, which are integrated into the resource control layer (RCL) of AQUILA.

The extensions of the measurement utilities for the second trial are mainly focused to:

- provide **enhanced load generators** for the evaluation and validation of the QoS architecture and to
- **collect data from the routers** to support admission control on the edge devices as well as the monitoring of the core network

The document is structured as follows: After the introduction section 2 gives an overview about the results regarding the measurement tools of the first trial of AQUILA and the main goals defined for the measurement tools that will be provided for the second trial. Section 3 focuses on the enhancements in the design of the distributed measurement architecture. After that section 4 describes the DMA for the second trial, its integrated tools and the enhanced measurement information database. Section 5 describes additional application-level load generators provided for the second trial, which are not covered by the DMA but can co-exist and used with it in parallel. Section 6 introduces the plans about the evaluation and validation of the MBAC algorithms by passive measurements using network capture cards.

The document is concluded by a state of the art section, which points out international projects and work done in the area of performance measurements in the Internet.

The annexes A, B and C appended at the end of the document contain more detailed information for some of the chapters.

2 Conclusions of the First Trial

2.1 Summary of the Results of the First Trial

During the first trial the correct working of the AQUILA architecture, that uses a separate resource control layer to provide QoS-capable network services, was extensively tested. The tests were carried out both with the Distributed Measurement Architecture (DMA) developed within AQUILA and commercially available test equipment.

The DMA used in the first trial is described in [D2301]. Details of the integration process of the DMA in the trial sites can be found in [D3101] and details about the first trial results and the usage of the DMA in [D3201]. All in all the different measurement tools of the DMA were deployed successfully.

During the installation of the DMA and the integration process the following reasons for difficulties were identified:

- Installing the components of the DMA on different hardware (PCs, network cards, etc.).
- Using different versions of Linux as underlying OS (different versions of SUSE or even other Linux distributions than SUSE).
- Installing the active network probing tool as well as the synthetic flow generator on one machine (necessary to save PCs, but originally not designed like this).
- To specific installation descriptions (i.e. without concerning different Linux versions).

All problems concerning installation and integration of the DMA could finally be solved. During the first trial the following suggestions for improvement were made:

- The operation of the DMA can be simplified by restructuring its graphical user interface (GUI).
- The available load generators were not fully satisfying the trial requirements.
- The number of flows running simultaneously should be increased.

Especially improvements concerning the GUI could already be implemented in the first trial extension. The started simplification of the operation of the DMA will be continued until the second trial. The identified issues for improvement will also be taken care of in the second trial.

2.2 Goals for the Second Trial

The following goals concerning the DMA for the second trial were identified. Details concerning the components of the DMA for the second trial can be found in section 4.

- Easy operation of the DMA by simplification of the GUI (details in section 4.5):
 - Improvement of the navigation (details in section 4.5.1)
 - Easier configuration for groups of flows (“multiple flow generator”, details in section 4.5.5)
 - Online-monitoring of measurement flows and hops within the GUI (details in section 4.5.7)
 - Implementation of a flow search function (details in section 4.5.8)
 - Display of aggregated results as graphs for both, flow monitoring and result evaluation (details in section 4.5.9)
- MBAC support (details in section 3.1)
- Analysis of loss estimation (details in section 3.2)
- Active network probing tool and synthetic flow generator on one machine (details in section 4)
- Reservation interface for measurement flows to AQUILA architecture (details in section 4.1.1)
- Refined load generators for the DMA (details in section 4.1.2, 4.2.1)

To reach these goals, enhancements are necessary for the distributed measurement architecture itself and its measurement components. Also the GUI needs some enhancements to fulfil the requirements.

In addition to the enhancements of the DMA, useful measurement components from other projects will be integrated to enhance the second trial with

- additional application-like load generators
- passive measurements for the validation of MBAC

3 Enhancements for the Second Trial

The distributed measurement architecture (DMA) designed for the first trial [D2301] integrates three different measurement methods: The generation of *application-like traffic*, the generation of *active probes* into the network to measure path performance characteristics and passive measurements by *router monitoring* to get status information about the network from the routers.

To implement the architecture, three tools have been developed. As depicted in Figure 3-1, for the second trial a similar approach with the following tools is used:

- The **Synthetic Flow Generator** uses application-like measurement agents (MAa) to produce synthetic flows that follow different Internet applications like FTP, VoIP and audio/video streaming. As the application-like measurement agents are emulating end-user traffic, they are located near the end-user hosts.
- The **Active Network Probing Tool** uses probing measurement agents (MAp) to inject probing packets into the network. With these probing packets the path performance characteristics of the network are evaluated. As they are designed to support the network operation for ISPs, they are located at the providers network edges.
- The **Router QoS Monitoring Tool** (RM) monitors QoS related parameters from the output interfaces from the core and from edge routers to get a view of the network situation and to detect possible bottlenecks. The router monitor is a distributed application and is located near the routers.

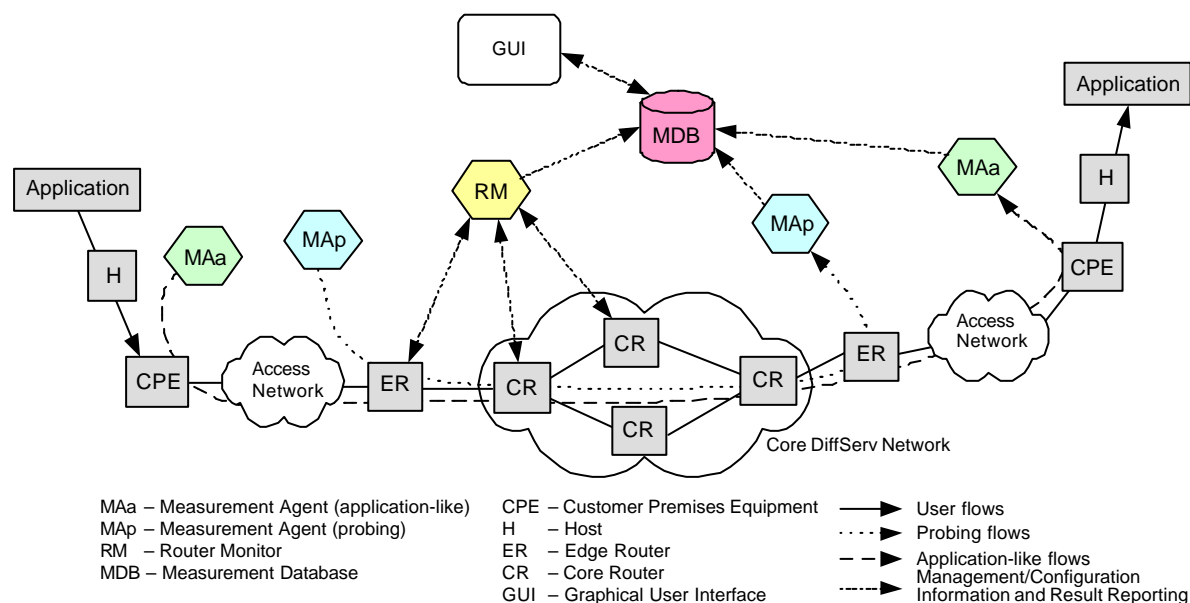


Figure 3-1: AQUILA Distributed Measurement Architecture

To integrate the single tools into the DMA, a measurement database (MDB) and a graphical user interface (GUI) were developed.

Some enhancements to the architecture have been developed, which are pointed out in this section. Furthermore the components itself got some enhancements, which are described in section 4.

3.1 MBAC Support & MBAC Validation

The admission control in the second trial is based on measuring the actual traffic on the edge device links to the core network. If the traffic rate to the core network exceeds a threshold, no new flows are admitted.

All of the measurement based admission control algorithms as specified for the second AQUILA trial [D1302] use the mean traffic rate on the network interfaces as input parameters. Each of the traffic classes that support MBAC uses different rules for the admission control. To measure the mean traffic rate of an interface, the edge router counts the number of bytes sent and received via the link to the core network. These numbers are regularly polled by the ACA with defined intervals, and from these numbers the mean rate for the interval is calculated. The mean rate is reported to the ACA, which uses the number to perform MBAC.

In the first trial, all router monitoring software was located in the database server. This concept cannot be used with MBAC, since signalling traffic to edge devices and ACAs would be far too high. For scalability reasons the router monitoring component is located within the ACA in the second trial implementation.

The limitation of measuring the mean rate on the output interfaces by requesting the router is the minimum time interval between two measurements. This approach hides the peaks of the traffic rate, which can occur between two consecutive measurements. To get the exact mean rate on the link at every time, passive measurements on the link are becoming necessary. With this approach all packets can be captured and the current traffic rate and its variances can be measured accurately. Major drawback of this approach is that the effort for deploying the necessary components into the network is very high, as every link between the edge devices and the core network has to be traced by special packet capturing equipment. But this approach is useful to validate the MBAC algorithms within the trial network. Details on this approach can be found in section 4.5.10.

3.2 Measurement Control with QoS Hypothesis Testing

The loss rate will be measured in the 2nd trial to validate the resource control and admission control. The following loss rates will be measured

- e2e loss rate of the flows sent by the load generators
- loss rate of the probing flows

- loss rates of the router interfaces

The duration of the measurement interval for a stochastic process is important for the feedback control mechanisms like AQUILA MBAC and for the minimisation of the effort for validation of measurements. Figure 3-2 shows the convergence of the estimated mean dependent from the measurement burst length (according to the law of large numbers). For hypothesis testing, the necessary length of the measurement interval can be determined.

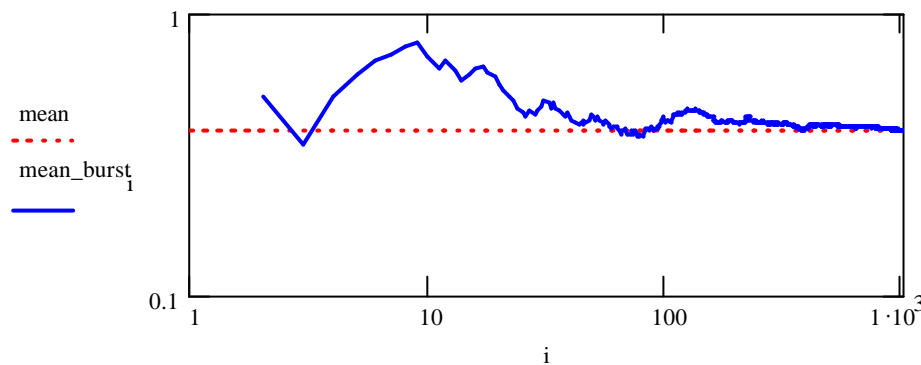


Figure 3-2: Convergence of mean estimation

A measured process (M) represents all stochastic events of the observed process, e.g. the one-way delay or loss of all packets. Probing (P) means the generation of additional packet flows. The experienced QoS values of the probing flow's packets are assumed to be the same as the QoS values of the flow P+M. Sampling (S) means the measurement of selected "samples" of a packet flow. The experienced QoS values of the sampled packets are assumed to be the same as the QoS values of the flow M (or M+P). Sampling can be active (= probing for M+P) and passive for M.

The following chapters describe methods for the measurement of QoS parameters.

3.2.1 QoS Mean Value Estimation

Accurate estimation of the loss rate: The mean loss rate will be estimated for independent and dependent loss events:

- independent losses: The number of losses is binomial distributed and the accuracy of the mean loss rate estimation can be calculated. For more details please refer to annex A (section 10.1)
- dependent losses: The aggregation of a sufficient large number of losses into independent batches results in (Lindeberg-Levy Lemma) normal distributed batch mean losses and the confidence interval can be calculated. For more details please refer to annex A (section 10.2)

The mean value estimation of a QoS value can be done in both, a stationary and a non-stationary environment. In a non-stationary environment older values are less weighted by applying smoothing algorithms.

Assuming a stationary environment the mean value can be estimated by using the Chi-square (χ^2) test (in case of a known variance) or by using the t-distribution (in case the variance is unknown).

Table 3-1 gives an overview on this approaches. The loss measurements of the flows are given by single loss events: $\text{loss_meas} = \{0,0,0,1,0,1,0,\dots\}$; 0 = packet sent, 1 = packet lost. By aggregation of loss measurements we mean the sequence of batches of summarised single loss events: $\{0,0,1,0,1,0\} \Rightarrow \frac{2}{6}$. Such values are the result of the router loss statistics monitoring.


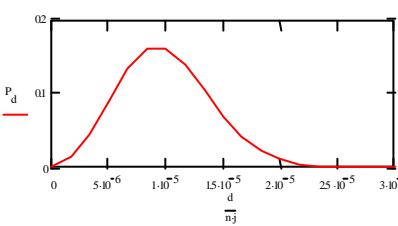
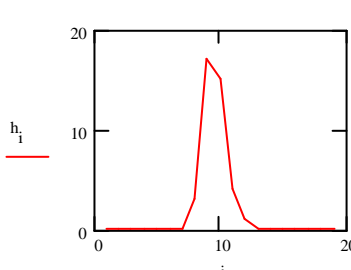
analysis type sample type	mean estimation $M[\text{parameter}] = ?$
0,1 (e.g. loss events)	aggregation to batches $\{0,0,1,0,0,1\} \Rightarrow M[\text{loss event}] = \frac{2}{6}$ independent  dependent
Independent aggregated losses: binomial distributed (e.g. collected from router loss statistics)	
Dependent aggregated losses: normal distributed (e.g. collected from router loss statistics)	variance known: χ^2 test variance unknown: t-test  i ... number of loss events per interval h_i ... number of intervals with i loss events

Table 3-1: Mean Value Estimation

The objective of measurement control is to get a predefined level of accuracy with a minimum number of measurements. Further work about this topic can be found in [Hube99].

3.2.2 QoS Gradient Estimation

Especially for feedback systems the gradient estimation method allows to find the right direction and values for the control decision. E.g. for MBAC: “If the load would be increased by Δb the delay time T will increase (= direction) by ΔT (= value)”. The gradient estimation can be done by the calculation of the following difference equation:

$$F = \frac{T(b + \Delta b, N) - T(b, N)}{\Delta b}$$

Two measurement processes have to be run, each with N measurements for the two different parameter values $b + \Delta b$ and b . The perturbation analysis uses the knowledge about the structure of the system to derive perturbation algorithms for a fast gradient estimation. The convergence of the perturbation algorithms is in the order of $mse = 1/N$ (mse = mean square error of gradient estimation)

[Suri89] compared with the classical difference equation approach with $mse = \frac{1}{\sqrt{N}}$. This approach

can be used to estimate the “crash-time” = $\text{mean}\{\text{time until } P(\text{buffer overflow})=1\}$. This is exactly the mean time in that a feedback controller like MBAC has to react on.

3.2.3 Hypothesis Testing between Two Loss Rates

Hypothesis testing between two loss rates: The accurate loss rate is difficult to calculate because loss events are “rare events”. For operational purposes the accurate loss rate is often not needed, a lower level of knowledge about the “truth” of one of the two hypothesis

- Hypothesis 0 = QoS level 0: the target QoS e.g. loss rate $p_0 = 10^{-5}$ and
- Hypothesis 1 = QoS level 1: the tolerance QoS level, e.g. loss rate $p_1 = 10^{-3}$

is sufficient. Such a test needs less samples compared with the accurate loss rate estimation. The hypothesis testing between two loss rates will be realised for two approaches:

- Maximum likelihood estimation of the crossing of QoS Level 0 or QoS Level 1 (Crosspoint estimation [Papa79]). For more details please refer to annex B (section 11.1)
- Wald’s sequential testing [Fisz78]. For more details please refer to annex B (section 11.2)

The measurement architecture provides the data needed for this approach in the measurement database.

Table 3-2 shows the differentiation between these two approaches.

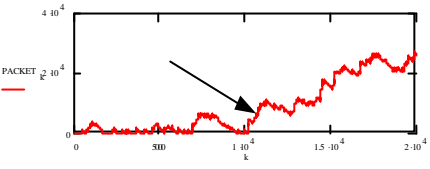
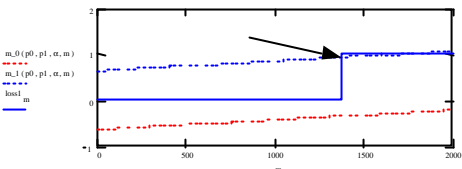
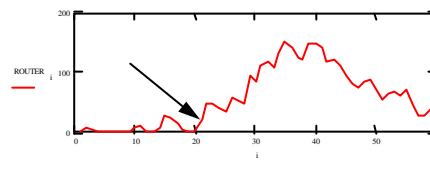
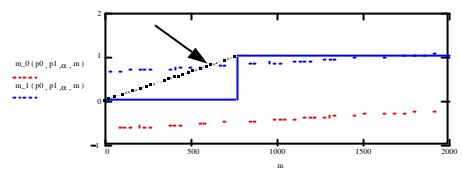
analysis type	<u>cross point estimation</u> assumption: $p \rightarrow p+\tau, p+\tau \rightarrow p$ MLR estimation of the cross point	<u>(p or p+τ) - testing</u> assumption: p or $p+\tau$ sequential testing
sample type		
0,1 (e.g. loss events)	 <p>MLR monitor function (0,1) measurements \rightarrow [Papa79]</p>	 <p>Sequential testing (0,1) measurements \rightarrow [Fisz78]</p>
aggregated losses (e.g. collected from router loss statistics)	 <p>MLR monitor function (lost / sent) measurements \rightarrow annex B (section 11.1.2)</p>	 <p>Sequential testing (lost / sent) measurements \rightarrow annex B (section 11.2.1)</p>

Table 3-2: Differentiation between cross point estimation and sequential testing

3.3 DMA GUI

One very important point for any tool is the way how the user communicates with the system. It must be easy to understand and easy to use. For measurement systems three important steps must be done. The system must be configured, the measurements must be started and the results must be shown. These are the same steps for the flow generator and the active measurement tool. It is uncomfortable to do nearly the same steps twice and use nearly the same interface twice. Also both tools use the same database and nearly the same things have to be configured.

The third integrated measurement tool of the DMA is the router QoS monitoring tool. The configuration of this tool is mostly done via configuration files, as it is assumed to have a static configuration for the whole measurement process. This tool also uses the measurement database to report the measurement results. The display of these measurement results has also been integrated to the GUI of the DMA.

The conclusion for the second trial is, that all integrated tools of the DMA work with the same GUI, which communicates with the clients via the measurement database.

4 DMA Components for the Second Trial

To implement the distributed measurement architecture, the components illustrated in Figure 4-1 are necessary.

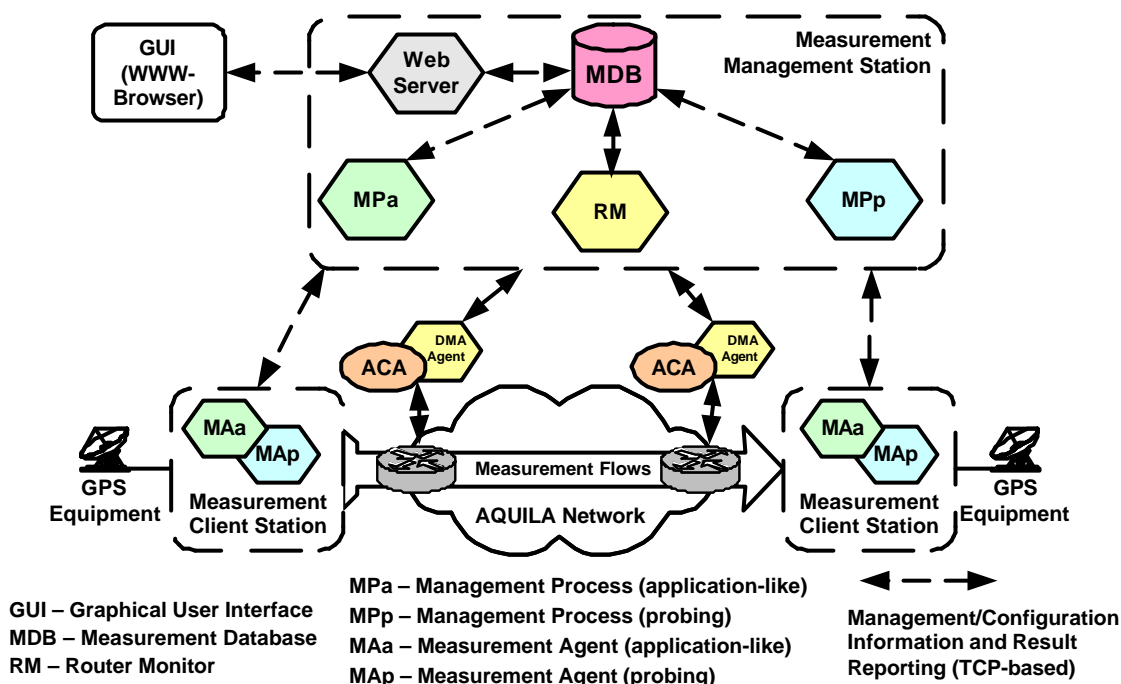


Figure 4-1: Components of the DMA

To enable the exact measurement of one-way delay, the active agents (MAa & MAp) are intended to be equipped with GPS hardware. If GPS hardware is not available, the system clocks can be synchronised (with a loss of accuracy) via NTP.

Because of their different tasks, the MAa and the MAp are located in different regions of the network (Figure 3-1). But to save the number of PC's (and the necessary GPS hardware) to be used in the trials of AQUILA, the two kinds of measurement agents have been installed on the same machines. In the first trial, some problems with the co-existence of the MAa and the MAp occurred, when both agents tried to access the GPS hardware simultaneously. To overcome these difficulties, some adaptations on the code were necessary. Now the system clock is used as time reference, which is directly synchronised to the GPS hardware via NTP.

This section describes the single components of the distributed measurement architecture. It focuses on the enhancements for the second trial, while the basic functionality of the components is already described in [D2301].

4.1 Synthetic Flow Generator

As described in [D2301], the synthetic flow generator (or CM Toolset) performs measurements by generation of application-like flows and the measurement of their performance parameters.

In the first trial some limitations of the synthetic flow generator were discovered. Therefore some extensions and changes are provided for the second trial, which are described in this section.

As depicted in Figure 4-1 the synthetic flow generator consists of

- a management process which polls the measurement database in regular intervals for new measurement scenarios and
- distributed measurement agents, which are executing the tests and reporting the results back to the management process.

For the second trial, the management process is also responsible for requesting reservations from the AQUILA resource control layer.

4.1.1 Management Process for Application-like Measurement Flows (MPa)

This section describes the enhancements on the distributed measurement agents for application-like measurement flows (CMCaller). The CMCaller is the centralised process that controls the distributed measurement agents (section 4.1.2). Because of performance reasons, it is intended that the CMCaller runs on the same machine as the measurement database. The management process of the synthetic flow generator has the following tasks:

- check the measurement database in regular intervals for new measurement flows
- optional: perform resource reservation by sending the configured reservation request to the AQUILA RCL via the EAT.
- distribute the measurement flow configuration to the specified sending and receiving agent
- awaiting the results of the distributed measurement agents and writing the results back to the measurement database.

4.1.1.1 Resource Reservation

For the passing of resource reservations from the users or from the end-user applications to the AQUILA resource control layer (RCL) an entity called end-user application toolkit (EAT) is responsible. The EAT provides the following interfaces for end-users or applications to specify resource reservation parameters:

- **Reservation GUI:** For the manual configuration of reservation requests by end-users a web-based graphical user interface is provided. This interface has two different abstraction levels for the users. On the one hand it provides a GUI for a detailed specification of single reservation parameters and on the other hand it provides a more user-friendly GUI, which has already a mapping between the reservation parameters and some selected legacy applications.
- **Internal API:** The internal API is based on CORBA and can be used by QoS-enabled applications, preferably written in Java.
- **Script Interface:** The script interface provides an easy way to set up reservation requests by using XML documents. It reads the reservation parameters from an XML file, which is passed to the class called EATScript.
- **Proxies:** Several proxies are provided for different applications or protocols. The proxies concept was introduced to support also legacy applications, that are either not aware of doing reservation requests or using other reservation mechanisms as specified in AQUILA.

The different application interfaces of the EAT are described in more detail in the “User Guide for End-user Application Toolkit” [D2203].

For performing reservation requests from the synthetic flow generator, it has been decided to use the script interface for resource reservation.

The resource reservation is done by the measurement management station before the measurement flow is started. Table 4-1 shows the process of a measurement flow with reservation.

Step	Description
0	Start-up from the CMCaller with login to the AQUILA RCL via EAT Script
1	new measurement flow with reservation found in database
2	request reservation via EAT Script performed by measurement management station
3	distribute measurement flow information to receiving and sending daemons
4	measurement flow starts/ends with the specified parameters and reports measurement results
5	reservation release via EAT Script performed by measurement management station

Table 4-1 Measurement procedure with reservation request

4.1.1.2 Performance Enhancements

During the first trial performance problems of the management process in combination with database have been discovered. The limitations occurred when a larger number of measurement agents are reporting results at the same time. The bottleneck was the connection from the management process to the database. To overcome this problem, the database connection is now limited to one writing process at a time. The other result processes are scheduled and have to wait until they get a semaphore.

4.1.2 Application-like Measurement Agent (MAa)

This section describes the changes and enhancements on the distributed measurement agents for application-like measurement flows (CMDaemons). These agents are distributed along the network, which should be measured. They retrieve the parameters for their measurement flows and also report the measurement results back to the management process (CMCaller).

4.1.2.1 Time Synchronisation

In the first trial, some difficulties in running the application-like measurement agent together with the probing agent occurred because of the limitations of the GPS hardware. Both measurement agents wanted to retrieve the current time directly from the GPS card each time a packet has been sent or received. Each time an agent requested the GPS card, it has been blocked for a short time. Within this time, the GPS card was not accessible by the other agent. The same problem happened, when multiple measurements with application-like flows were started simultaneously.

For the second trial the measurement agents do not access the GPS hardware directly, but getting the time from the current system time. To achieve time synchronisation between two measurement hosts, NTP in combination with the GPS hardware is used. NTP is used to synchronise the local clocks of the hosts by using the GPS hardware as reference time server. If the host does not have its own GPS card, the synchronisation via NTP can be done using another time-synchronised machine. Using this approach the timestamp can become inaccurate, depending on the distance between the time server and the client.

To achieve reliable measurement results, the result of each measurement packet indicates whether the local clock was synchronised by using an internal GPS hardware or by using an external NTP server.

4.1.2.2 Minimum Packet Inter-departure Time

The packet inter-departure time is the waiting time for the sending measurement process between one packet has been departed and the next packet is sent (Figure 4-2).

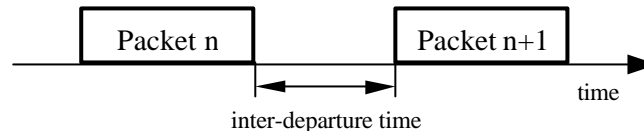


Figure 4-2: Packet Inter-departure Time

For the first trial, the minimum inter-departure time between two consecutive measurement packets was limited to multiples of 10ms. This limitation, which was introduced because of the Linux kernel granularity, was a drawback when generating e.g. Poisson distributed traffic sources. The characteristic of Poisson distributed measurement flows allows the theoretical time distance between two consecutive packets to converge to zero. Therefore the calculated time values can be also between the multiples of 10ms.

To enhance the accuracy, the new approach was to wait a number of CPU-ticks instead of sending the process into sleep. This solution has the drawback, that the system will be overloaded, especially when several measurement flows are started from the same sender machine. To reduce the load, the waiting time between two packets is now separated into two phases. The first phase is implemented by sending the process into a sleep, which is 25 ms smaller than the calculated inter-departure time. During the rest of the inter-departure time (second phase) the process waits a number of CPU-ticks until the departure time of the next packet is reached, which is sent afterwards.

It has to be noticed that the accuracy of the packet inter-departure time is limited by the end-system performance.

4.1.2.3 Result Aggregation

In the first trial all results from the application-like measurement agents were stored back to the measurement database when the measurement flow was ready. Dependent from the result options specified by the user either raw data (i.e. detailed per packet data) and overall result data or only the overall result data was stored.

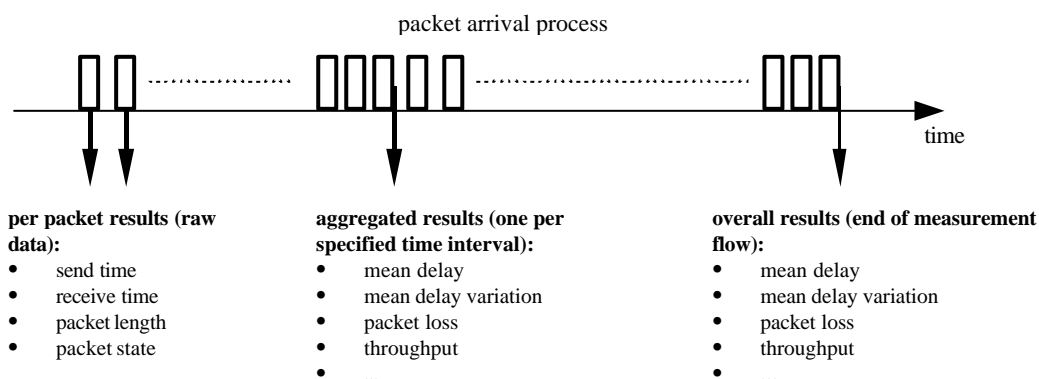


Figure 4-3: Result Differentiation

For the second trial a function for the aggregation of measurement results is provided. If an aggregation time (in seconds) for a measurement flow is specified, aggregated results are produced and reported to the measurement database in this constant time interval already during the measurement (Figure 4-3). With this feature it is possible to reduce the amount of measurement data without losing much information about the measurement results. Like before, also the per packet details can be reported back to the database, if specified. The raw data for this part of the measurement flow is reported together with the aggregated results. It allows also the monitoring of the measurement flow during run-time (see also section 4.5.7.2, which describes flow monitoring part of the GUI).

A flow then has several result entries in the database, one for each aggregate and a final overall result of the flow. The GUI can display graphs of both, the aggregated result data and the raw data.

4.1.2.4 Traffic Traces

A flexible way to generate traffic with the synthetic flow generator is the use of traffic traces. Tracefiles specify the packet size and the packet inter-departure time within a plain text file with two columns. The first column specifies the packet length in bytes, the second column represents the waiting time until the next packet is sent. By using tracefiles, arbitrary traffic sources can be simulated, but the generation of tracefiles and their distribution to the daemons is laborious. Tracefiles are preferably used, when having a packet trace from real applications, which should be re-generated by the synthetic flow generator. The tracefiles can be generated by using e.g. tcpdump. The following example shows a command for tracing UDP packets between a specified sender and receiver using a specified portnumber:

```
tcpdump -tt src source-ip and dst dest-ip and udp and port portnumber > dumpfile
```

To get the required tracefile format from the dumpfile it can be further processed e.g. by using the well-known Unix tool 'awk':

```
awk '{print prev,($1*1e+6-a);a=$1*1e+6;prev=$6}' dumpfile | tail +2 > tracefile
```

The output file must be made available at the sending host and can be specified with its absolute path via the GUI.

4.1.2.5 General Traffic Sources

Several state driven traffic sources (e.g. ON/OFF sources) are commonly used in simulating end-user traffic. Current parameter settings don't allow to generate arbitrary state driven sources, because the synthetic flow generators for the first trial were not state oriented, but was parameterised by the distribution of the measurement packet size and the distribution of the inter-departure time of the measurement packets. Transitions between different distributions of packet size and/or inter-departure time were possible by specifying transition probabilities using a Bernoulli random number (perform the transition or don't perform the transition).

Introducing a general model, which also supports e.g. ON/OFF sources needs the specification of several states. Each state can be either an ON state (packets are sent) or an OFF state (no packets

are sent). The duration of the state can be specified either by a time duration or by a number of packets. If the state is an ON state, the following parameters are additionally necessary:

- the distribution of packets with its parameters
- the distribution of the inter-departure time of the packets with its parameters.

It is planned to support constant, uniform and exponential distributions for all, i.e. for the state duration, the packet size and for the inter-departure time.

The support of the general traffic sources will be available for the second trial.

4.1.2.6 Path Discovery

To enable the router monitor performing the monitoring of the routers, which are involved in a measurement flow its path through the network can be discovered prior to the execution of the measurement flow. A “traceroute” [TRACEROUTE] is performed by the sending host using the receiving host as destination. Afterwards the path information is reported to the measurement database.

When using a QoS based network, packets with different TOS-Byte (or DSCP) values can take different routes between two hosts. Therefore the packets of traceroute are marked with the same TOS-Byte (or DSCP) as the measurement flow, that will be sent. Changes of the route during the measurement flow are not considered.

4.1.2.7 Windows Version

For the second trial the measurement agent of the synthetic flow generator is also available for Windows platforms. The implementation has been tested using Win9x/NT/2000. Currently not supported is the use of GPS-cards in the Windows environment. It is intended that the time synchronisation is done by the user (e.g. by using NTP).

4.2 Active Network Probing Tool

4.2.1 Measurement Agent (MAp)

4.2.1.1 Additional Load by Active Probing

By active probing additional load is generated. As can be seen from Table 4-2 the additional load is negligible in most practical cases.

packet size (incl. Header, IP+UDP) [bytes]	inter packet time [ms]	additional load [kbps]	additional load (64 kbps) [%]	additional load (128 kbps) [%]	additional load (768 kbps) [%]	additional load (2 Mbps) [%]	additional load (Ethernet) [%]	additional load (Fast Ethernet) [%]
60	10	48,000	75,000%	37,500%	6,250%	2,344%	0,480%	0,048%
60	100	4,800	7,500%	3,750%	0,625%	0,234%	0,048%	0,005%
60	1000	0,480	0,750%	0,375%	0,063%	0,023%	0,005%	0,000%
60	60000	0,008	0,013%	0,006%	0,001%	0,000%	0,000%	0,000%
150	10	120,000	187,500%	93,750%	15,625%	5,859%	1,200%	0,120%
150	100	12,000	18,750%	9,375%	1,563%	0,586%	0,120%	0,012%
150	1000	1,200	1,875%	0,938%	0,156%	0,059%	0,012%	0,001%
150	60000	0,020	0,031%	0,016%	0,003%	0,001%	0,000%	0,000%
400	10	320,000	500,000%	250,000%	41,667%	15,625%	3,200%	0,320%
400	100	32,000	50,000%	25,000%	4,167%	1,563%	0,320%	0,032%
400	1000	3,200	5,000%	2,500%	0,417%	0,156%	0,032%	0,003%
400	60000	0,053	0,083%	0,042%	0,007%	0,003%	0,001%	0,000%
1500	10	1200,000	1875,000%	937,500%	156,250%	58,594%	12,000%	1,200%
1500	100	120,000	187,500%	93,750%	15,625%	5,859%	1,200%	0,120%
1500	1000	12,000	18,750%	9,375%	1,563%	0,586%	0,120%	0,012%
1500	60000	0,200	0,313%	0,156%	0,026%	0,010%	0,002%	0,000%

Table 4-2: Additional load by active probing.

4.2.1.2 Aggregated Flow Load Generator

For testing the behaviour of the AQUILA architecture under "real" conditions it is important to have a background traffic generator that mimics a population of real network users. So for the second trial the functionality of the generator part of the AQUILA distributed measurement system will be extended accordingly.

In the last 10 years several studies/measurements were done (e.g. [PKC96]), where a self-similar relationship was observed for local and wide area networks. The heavy-tailed file size distributions and the operation of the reliable transmission and flow control mechanism of TCP cause the long-range dependency structure. Aggregated flows can be generated by multiplexing several sources of Pareto-distributed ON and OFF periods:

- Transmissions or traffic flows are initiated as a memoryless (M or Poisson) process with arrival rate λ
- The data volume V of each transmission is sent in a series of packets in constant time intervals at a considered access speed C
- Data volumes V of transmissions are independent and Pareto-distributed:

$$\Pr\{V > x\} = (x / \alpha)^{-b} \quad \text{where } b \text{ is a degree of long-range dependence}$$

$$E(V) = a / (1 - b) \quad \text{is the mean data volume per transmission,}$$

$E(N) \approx 1 a / (1 - b)$ is the mean number of transmissions in parallel

$E(R) \approx 1 C a / (1 - b)$ is the mean transmissions rate

- The aggregated traffic is self-similar with Hurst parameter $H = (3 - b)/2$
- $b \rightarrow 2$ or $H \rightarrow 0.5$ indicates weak, whereas $b, H \rightarrow 1$ indicates strong long-range dependence; $H \approx 0.85$ is typically measured for Internet traffic

Figure 4-4 shows some examples of generated Pareto traffic.

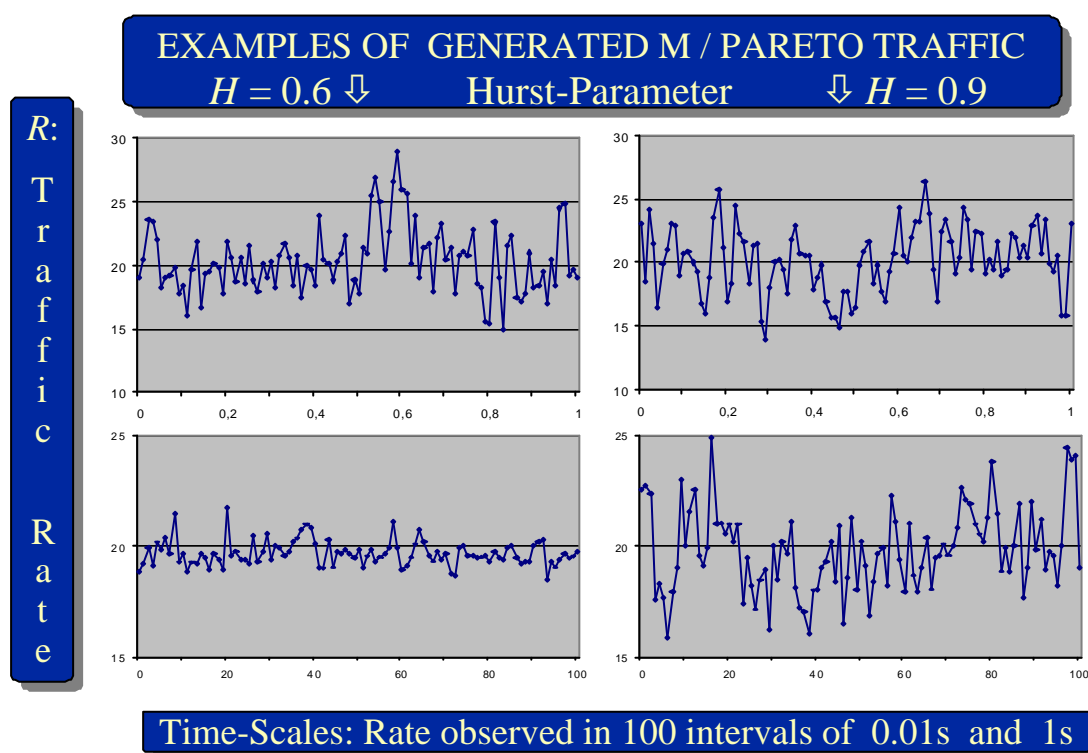


Figure 4-4: Examples of generated Pareto traffic

For the implementation of this distribution no changes in the database are necessary. Adaptations must be made in the GUI, the master station and the measurement agent.

4.2.2 Master (MPp)

The measurement agents are controlled by the master software module called “master”. During the startup the “master” reads from the configuration file “master.conf” the information how to access the measurement database and starts the processes (P1-Pn) which control the measurement agents. These processes (“Agent threads”) read the configuration data from the database, start / stop the measurement flows and store the results in the database. Errors and warnings are stored in the event-

log table in the database and / or the Logfile “master.log”. The Software module “MMon” provides a GUI for testing / debugging the “master” module.

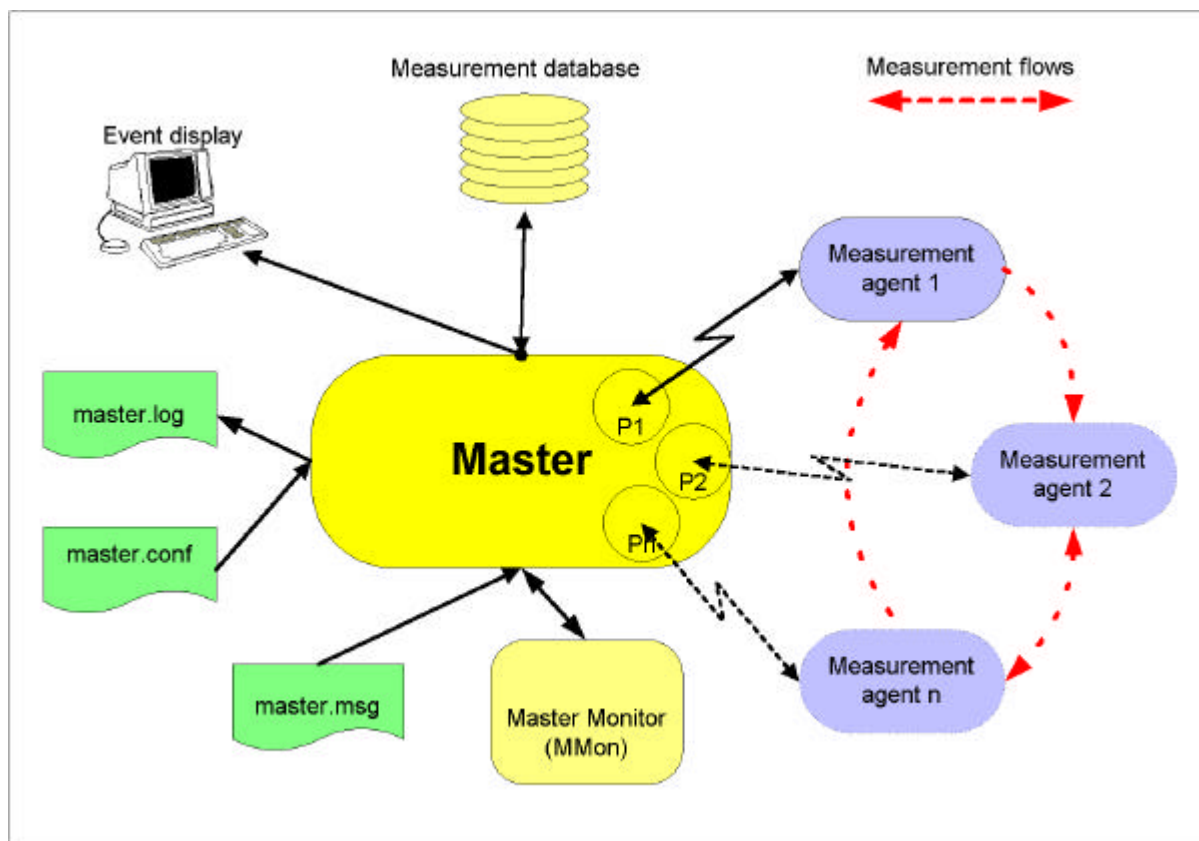


Figure 4-5: Structure of the "Master" software module

4.2.3 Measurement Agent CONTroller (MACON)

MACON (Measurement Agent CONTroller) is a simple to use GUI for DTA measurement agents. The controller is a Java2 application, and so executable on any Computer, where the Sun JDK 1.3 is installed.

MACON can control up to 10 measurement agents and 5 traffic classes. The Controller (see Figure 4-7) makes it easy to configure a measurement between the agents. MACON starts automatically a full meshed measure between the listed agents and traffic classes. Therefore no data base or web server is needed. Every information is stored locally.

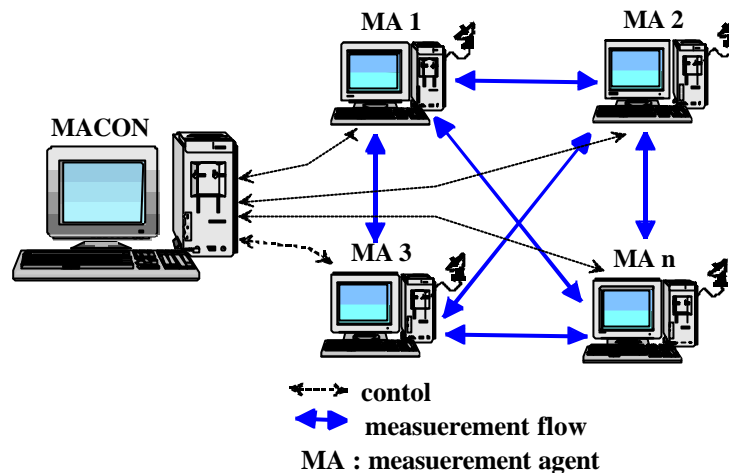


Figure 4-6: Fully meshed measurement with MACON

During the measurement the status of the agents and the current results are monitored. The online monitoring of the results can show the values of one-way delay, delay-variation and packet-loss. It is possible to store the results in csv-files and analyse them offline.

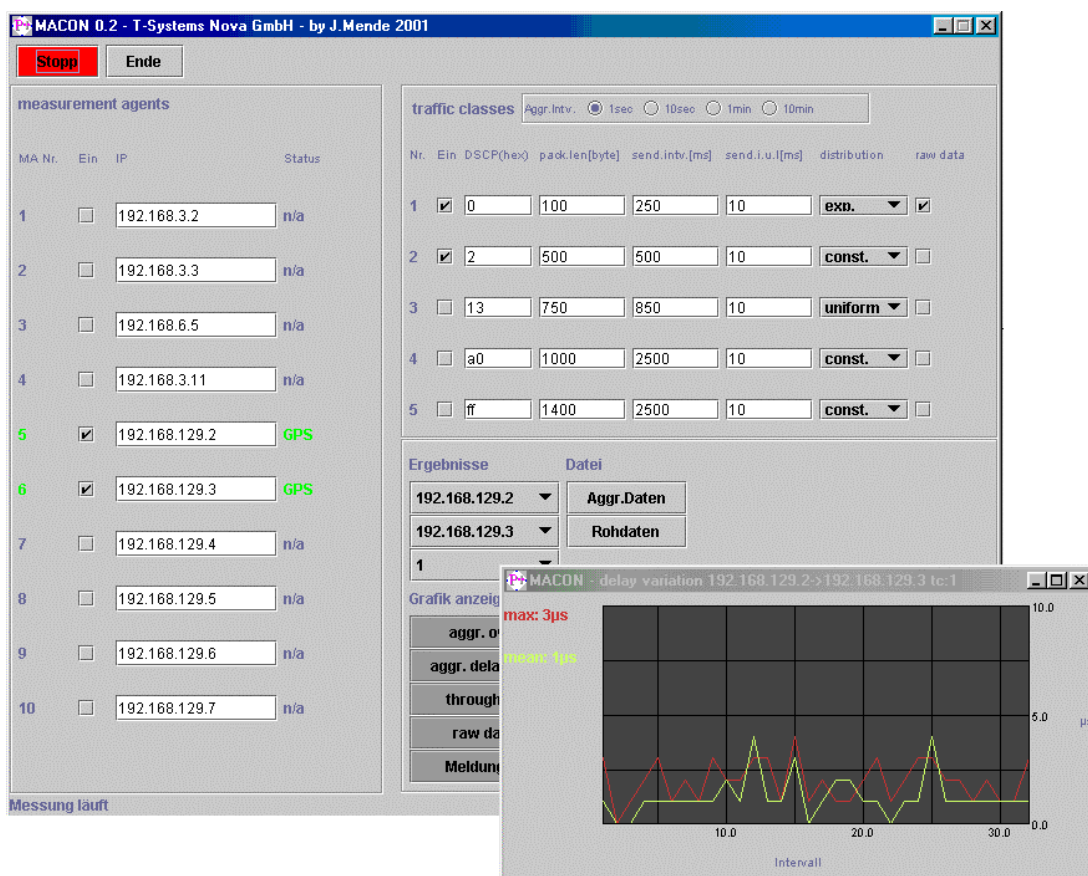


Figure 4-7: MACON-Screenshot

4.3 Router QoS Monitoring (RM)

The router QoS monitoring tool consists of three separate parts: edge router monitor (DMAAgent), core router monitor and the launcher for the monitors. The edge router monitors are within the corresponding ACAs that manage the edge routers. The launcher and core router monitor are located within the database server.

4.3.1 The Launcher Component

The launcher checks the database for new flows in regular intervals. When a new measurement flow is detected, it sends a request to start monitoring of the routers to the edge router monitors and core router monitors in the path of the flow. After the flow has finished, the edge and core router monitors send the results back to the launcher, which then saves the results to the database.

4.3.2 The Edge Router Monitor (DMAAgent)

The edge router monitor is integrated to the Router package of ACA. There are two types of parameters that are monitored with the edge router monitor. The first type is parameters related to one particular flow. These parameters contain the total number of packets/bytes for the flow, the number of dropped packets/bytes in the policer, the time since the last packet arrival, and size of current burst.

The second type of parameters is the parameters related to traffic classes for the outgoing interface and general router performance. These numbers include the total number of packets/bytes in the traffic class, the number of packet/byte drops and CPU usage.

The monitor listens to requests from the launcher, and starts monitoring the edge router, when the monitoring request is received. The monitor uses the interface provided by the Router package to access the edge router. The interface between the DMAAgent and the launcher is described in chapter 4.4.4.

4.3.3 The Core Router Monitor

The core router monitor runs in the database server machine. Otherwise the core router monitor will use the DMAAgent design for the monitoring. The monitor listens to requests from the launcher, and starts monitoring the core router or a set of core routers, when the monitoring request is received. The set of parameters to monitor is fixed, the measured parameters are the number of packets/bytes dropped, total number of packets/bytes for each traffic class and CPU usage for the router. The core router monitor accesses the router statistics using CLI or SNMP.

4.3.4 Interface to the AQUILA RCL

The interface between launcher and the DMAAgent is a CORBA interface. The interface provides methods to start the router monitoring in specified time intervals, to collect the monitoring data and to

stop the monitoring. When the monitoring is started, the DMAAgent starts retrieving data from the router to its local storage. When the collectData() method is called, the collected data will be sent to the launcher. This way the interference caused by the monitoring is minimised.

4.3.5 Monitoring Influence to the Network

Router monitoring produces additional overhead traffic in the network. The overhead is due to the polling of statistics from the router.

In the second trial, the monitoring of edge routers is integrated into the ACAs, that are near the edge routers. This way, the communication between the DMAAgent and the router has only local effect. Communication between the DMAAgent and Launcher component is done using a more efficient protocol, therefore the load of router monitoring is decreased. The caching of monitored data in the DMAAgent further reduces the load to the network.

The monitoring of core routers produces more traffic in the network than the edge router monitoring. This happens, because the core router monitoring component is located at the measurement server in the second trial. Depending on the network to be monitored, the core router monitors can be distributed also in the network, so that the overhead of core router monitoring will be also minimised.

4.4 Measurement Database

The measurement database has roughly the same structure as already described in [D2301]. Only some adaptations were necessary to fulfil the needs of the measurement system. Like in the first trial, MySQL [MYSQL] will be used as DBMS for the second trial, as it is free for non-commercial use.

4.4.1 Database Model

Figure 4-8 shows the conceptual database model (entity-relationship diagram) of the measurement database for the second trial.

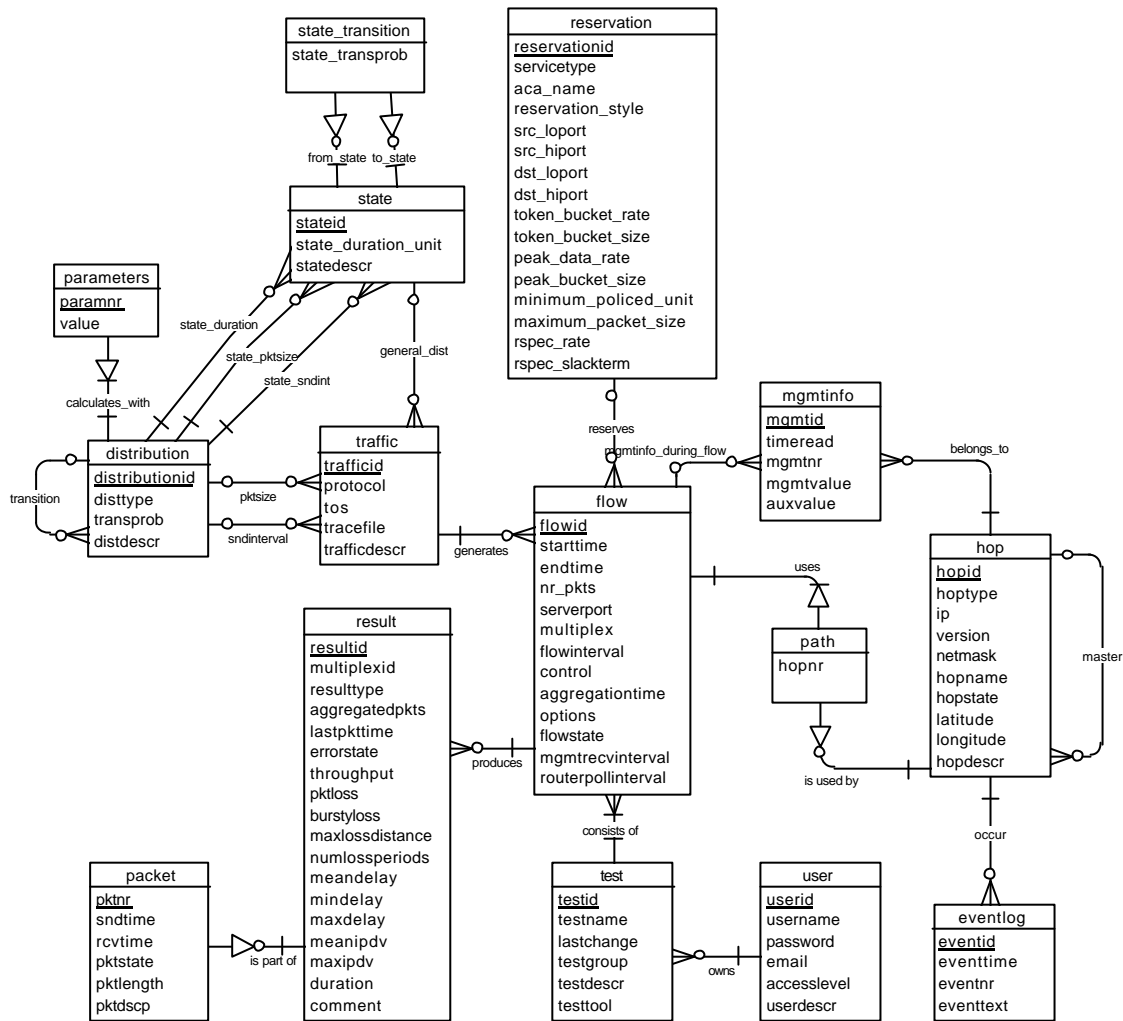


Figure 4-8: Conceptual Measurement Database Model

4.4.2 Description of Changes

This section describes the changes of the measurement database, which were defined since the delivery of [D2301].

4.4.2.1 Entity “user”

The email address of the user was introduced to enable automatic reporting of his tests (which is currently not implemented).

4.4.2.2 Entity “test”

The new introduced test name allows a short description of the test.

The test state will no longer be stored in the database as it is possible to calculate the test state out of the states of the flows in this test.

4.4.2.3 Entity “flow”

To enable the configuration of the router monitor, the following two fields are used:

- **routerpollinterval:** Specifies, how often data is collected from the router by the DMAAgent. This does not specify, whether the data is retrieved by the DMA.
- **mgmtrecvinterval:** Specifies, how often data is retrieved by the router monitor (RM) from the DMAAgent. In this intervals the collected data is written to the database.

4.4.2.4 Entity “hop”

To this entity the following attributes were added:

- **Latitude/longitude:** As the measurement clients should be equipped with GPS cards for time synchronisation, they are also aware of there geographic location. This information can be stored to the measurement database.
- **Netmask:** The netmask of the hop (or interface) is necessary to perform AQUILA reservations.
- **Hopstate:** The hop state defines the state of the hop as well as the daemons running on the system. The different operating levels of hops are
 - **Ping-able:** The hop is alive, i.e. the interface is up and reachable
 - **Clock synchronised:** The system clock of the machine is synchronised (e.g. via NTP, the network time protocol).
 - **GPS-equipped:** The hop has direct access to GPS-equipment (e.g. a GPS card is installed in the system).

4.4.2.5 Entity “packet”

The packet state now also stores information about the accuracy of its timestamps. Like the hop state the packet state can be either NTP synchronised or synchronised to a GPS source.

Also the receiving DSCP of the packet is now supported. As the routers possibly change the Differentiated Services Code Point from the packet until it is received, this value can be interesting to see how the packet was threatened.

4.4.2.6 Entity “reservation”

The entity reservation (formerly RSVP) has been extended to be able to support resource reservations within the AQUILA network. In addition to the RSVP services (for both guaranteed service and controlled-load service) the AQUILA network services PCBR (Premium Constant Bit Rate), PVBR (Premium Variable Bit Rate), PMM (Premium Multimedia) and PMC (Premium Mission Critical) are supported.

4.4.2.7 Entity “state”

For the support of traffic sources that can be general parameterised this new entity has been introduced. With this feature, the traffic sources can have n different states, each state has a duration (state_duration), a packet size (state_pktsize) and a sending interval (state_sndint) specified by a distribution and its parameters.

4.5 Graphical User Interface

The graphical user interface (GUI) is integrated for the three measurement tools of the AQUILA DMA. The GUI is web based, i.e. the user only needs a web browser for the communication with the measurement system. As server the Apache web server [APACHE] with the server side script language PHP4 [PHP] is used.

4.5.1 Navigation

In discussion with the partners and with two redesigns the GUI was implemented with a “tree structured” navigation. The design is now “Windows Explorer-like” with three levels. Every functionality of the system is reflected with one link of the navigation and leads to a screen in the main window (see Figure 4-9).

The navigation allows switching between the different screens with a maximum of three mouse clicks.

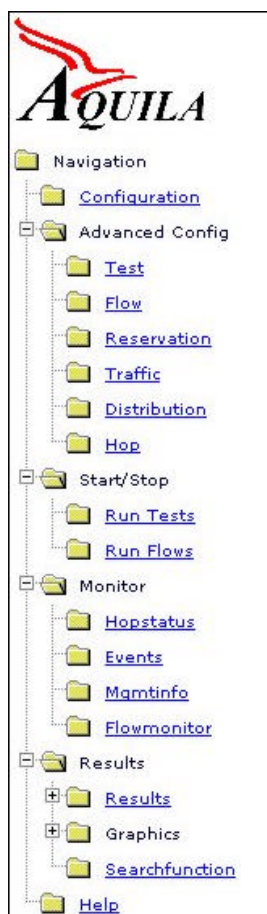


Figure 4-9: DMA navigation menu.

Some features for an easy handling in every screen where implemented. There is a sorting mechanism for the data fields in every screen, which handles tables from the database.

4.5.2 User

The GUI has a security mechanism that only authorised users can enter the GUI. In the screen in Figure 4-10 users can be generated, modified and deleted. Every field is explained with an online help on the bottom of the mask. There are necessary fields to fill in.

New user

Parameter	Input
Help	
Username (1)	<input type="text"/>
Password (2)	<input type="password"/>
E-Mail (3)	<input type="text"/>
Accesslevel (4)	<input checked="" type="radio"/> User <input type="radio"/> Administrator
Userdescription (5)	<input type="text"/>
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	
	Back
(1)	A significant name for the new user. It must be unique, if this name is used by anybody else you will get an error message after pressing 'Submit'. Choose a new name or use a suffix, e.g. name_1. Minimum 4 characters, maximum 50 characters.
(2)	A password for the new user. Minimum 4 characters, maximum 50 characters. The password should be a mix out of characters and numbers.
(3)	Enter a valid E-Mail address of the new user.
(4)	The accesslevel specifies if the new user will be an administrator. Administrators can modify all configurations in every menu. Only administrators can configure the hops and work with the usertables. As a regular user you can only work with 'your' tests and flows.
(5)	Fill in a significant description of the user.

Figure 4-10: Managing DMA users.

4.5.3 Quick Start

Using an interface must be intuitive. So a “Quick Start” was implemented to start a measurement without knowing every detailed feature of the GUI (Figure 4-11). The user clicks on configuration and is lead to a beginning screen, where he can configure a complete measurement in two or more steps. Every necessary step to generate a flow is opened and at the end the user is redirected to the result graphics of the flows, he configured and started with the “Quick Start”. The main window is the screen where the flows are configured. Old or new tests, reservations, traffic or distributions can be chosen. It is also possible to mesh more than one pair of measurement clients. With jokers and wildcards in the input fields sub-nets or parts of a network can be meshed.

Quick Starting Measurement
Define Flows -> Run -> Results

Parameter	Input
Testid (1)	?Funktionstest
Reservation (2)	-No Reservation-
Trafficname (3)	-New Traffic-
Serverport (4)	-New Traffic-
Multiplex (5)	1. Best effort
Aggregation time (6)	2.TC 1 3.TC 2 4.TC 3 5.TC1 last 6.TC2 last 7.lesverkehr 10 prozent 8.messverbindung losttest 9.lesverkehr 2 prozent 10.lostgenerator 10 prozent 11.Losspattern
Options (7)	<input checked="" type="checkbox"/> Delay <input checked="" type="checkbox"/> IP delay variation <input checked="" type="checkbox"/> Duration <input type="checkbox"/> Path discovery
Hopselection (8)	Senderhops (use * and ? as wildcards) Hop P 192.168.* Receiverhops (use * and ? as wildcards) Hop IP 192.168.*
Managementinterval (9)	400
Managementfile (10)	
Timecontrol (11)	<input type="radio"/> Limited time(SPU) Starttime (ddmmww) (hhmmss) 15 11 2001 09 52 33 or start in minutes Endtime (ddmmww) (hhmmss) 15 11 2001 09 52 33 or stop in minutes <input type="radio"/> Limited packet(SPU) Starttime (ddmmww) (hhmmss) 15 11 2001 09 52 33 or start in minutes Flowinterval (hhmmss) 000 05 00 Number of packets 3000 <input type="radio"/> Unlimited manual(DTA)
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

Note: Starttimes should be valid after you are ready with your input

Figure 4-11: DMA quick start.

4.5.4 Advanced Configuration

The “Advanced config” is a possibility to configure all parameters in detail.

These parameters are:

- Test
- Flow
- Traffic
- Distribution
- Reservation

- Hop

An interface should allow a quick handling of the standard functions. The parameters can be generated, modified, deleted and copied. A selection of used flows must be possible and also the possibility to generate new parameters. Generating means choosing different values in the input fields of every parameter. For example a flow needs a source and a destination address and a traffic class. The kind of results must be filled in by the user. If changes are necessary, a single field can be marked and the values can be changed and written back to the database. The parameters can be deleted and as a comfort function new parameters with nearly the same values can be copied. As an example there are two screenshots, one for the selection (Figure 4-12) and one with a mask, where the inputs can be made (Figure 4-13).

Configure traffic

D	M	C	<Trafficid>	<Trafficdescription>	<Packetsize Distribution>	<Sendinterval Distribution>	<Protocol>	<Type of service>	<Tracefile>
<input type="button" value="Submit"/> <input type="button" value="Reset"/>			<input type="checkbox"/> New Traffic						
(D=Delete, M=Modify, C=Copy)									
<input type="radio"/>			15	TC1 fast klein II	60 byte	40 ms / const	UDP	1	
<input type="radio"/>			14	TC1 fast klein	60 byte	100 ms	UDP	1	
<input type="radio"/>			13	lastgenerator 4 prozent	20000 byte/const	40 ms / const	UDP	0	
<input type="radio"/>			12	lastgenerator 2.5 prozent	12500 byte/const	40 ms / const	UDP	0	

Figure 4-12: Advanced configuration – selecting traffic.

Modify traffic	
Parameter	Input
Packetsize-Distribution ID (1)	21.constant:0.000
Sendinterval-Distribution ID (2)	6.constant:100 ms:0.000
Protocol (3)	<input type="radio"/> TCP <input checked="" type="radio"/> UDP <input type="radio"/> TCP_NoDelay
Type of Service (4)	MSB <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> LSB
Tracefile (5)	
Trafficdescription (6)	testverkehr 2 prozent
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

(1) Choose an ID of the distribution of the packet size. This field can be NULL, if a path to a tracefile is defined [Choose Tracefile].

(2) Choose an ID of the distribution of the sending interval. This field can be NULL, if a path to a tracefile is defined [Choose Tracefile].

(3) Choose the protocol that will be used for the traffic.

(4) By clicking the fields you can set the ToS byte at the sender (IP Header) field Default is 0.

(5) Fill in a pointer to a tracefile. Use the right fileformat of the platform. You must fill in a pointer if no Distribution is choosen.

(6) Fill in a short description of the traffic properties. Use a senseful description to recognize your traffic in the table.

Figure 4-13: Advanced configuration – modifying traffic.

4.5.5 Multiple Flow Generator

In the first trial, each reservation for measurement flows were established separately from the measurement flow definition. The implementation of the synthetic flow generator for the second trial is able to perform automatic AQUILA reservation, if specified. This feature allows to validate the admission control algorithms easier than in the first trial.

A new feature from the GUI is, that multiple flows can be configured in a single step. This can be used to have flows in the database, which permanently ask for reservations and starts the measurement flows, if the reservation was successful. If not, the flow won't start and a reservation request for the next flow is performed.

The multiple flow generator is parameterised as follows:

- number of flows to be started
- interval between two flows to be started in seconds (and its distribution)
- usual flow specification (start time, end time, traffic model etc.)

In addition a function for copying whole test scenarios (i.e. including the measurement flows) will be provided for the second trial.

4.5.6 Start/Stop

To start or stop a measurement a control flag must be set in the database. This is done in the GUI with a flow or a test selection. The flows / tests, which should start or stop the measurement can be marked with a checkbox (Figure 4-14).

It has to be noticed, that this functionality is only provided for probing flows. The start and stop times of flows for the synthetic flow generator are specified during the configuration of the measurement flows.

Run tests

Stop / Start	<Testid>	<Userid>	<Testname>	Flowstates	<Testdescription>	<Testtool>	<Lastchange>	<Testgroup>
Submit	Reset							Help
<input checked="" type="checkbox"/>	18	2	test berlin 7	All flows ready	3-Tage Messung	Measurement probes (DTA)	29-10-2001 11:48:21	0
<input checked="" type="checkbox"/>	17	2	test berlin 6	All flows ready	wochenendmessung	Measurement probes (DTA)	26-10-2001 15:49:58	0
<input checked="" type="checkbox"/>	16	2	test berlin 5	All flows ready	test rechner berlin mit 2 Mbit	Measurement probes (DTA)	25-10-2001 12:56:26	0
<input checked="" type="checkbox"/>	15	2	test berlin 4	All flows ready	test rechner berlin mit Video	Measurement probes (DTA)	23-10-2001 13:40:57	0
<input checked="" type="checkbox"/>	14	2	test berlin 3	All flows ready	test rechner berlin 3	Measurement probes (DTA)	23-10-2001 12:57:30	0
<input checked="" type="checkbox"/>	13	2	test berlin 2	All flows ready	test rechner berlin 2	Measurement probes (DTA)	23-10-2001 12:19:18	0
<input checked="" type="checkbox"/>	12	2	test berlin 1	All flows ready	test rechner berlin 1	Measurement probes (DTA)	23-10-2001 11:23:52	0
<input checked="" type="checkbox"/>	11	2	test berlin	All flows ready	test rechner berlin	Measurement probes (DTA)	23-10-2001 08:13:08	0
<input checked="" type="checkbox"/>	10	1	Funktionstests	All flows ready	Funktionstests fuer neue Testnetzrechner	Measurement probes (DTA)	18-10-2001 07:14:10	0
<input checked="" type="checkbox"/>	9	1	Demo1	All flows ready	Demonstration 1	Measurement probes (DTA)	11-10-2001 11:43:56	0

Figure 4-14: Start / stop of probing flows.

4.5.7 Monitoring

4.5.7.1 Hop Status Monitoring

It is necessary for the user to get some information about the running system. With the hopstatus the user gets some information about the synchronisation and the availability of the measurement clients (Figure 4-15).

Show hopstatus

<Hopname>	<Hop IP>	<Hopstatus>	<Tooltype>
MC-6-2	192.168.6.2	GPS card available and synchronised via GPS	DTA
MC-7-2	192.168.7.2	GPS card available and synchronised via GPS	DTA
MC-3-2	192.168.3.2	GPS card available and synchronised via GPS	DTA
MS-3-4	192.168.3.4	Pingable but not synchronised	No Tool/Daemon running
MC-6-3	192.168.6.3	GPS card available and synchronised via GPS	DTA
MC-3-3	192.168.3.3	GPS card available and synchronised via GPS	DTA
MC-3-11	192.168.3.11	GPS card available and synchronised via GPS	DTA
MC-3-13	192.168.3.13	GPS card available, but not synchronised	DTA

Figure 4-15: Hop status monitoring.

Another screen shows information generated by the measurement system. The so called “eventlog” shows errors or warnings if the system is disconnected or a misconception was done (Figure 4-16).

Events			
Hop IP	Eventtime	Eventnumber	Eventtext
Submit Reset		<input checked="" type="radio"/> Last 10 <input type="radio"/> Last 100 <input type="radio"/> All <input type="radio"/> Delete	
	21-08-2001 07:59:57	0	Do_Query: try reconnecting to database (2)
192.168.3.4	16-08-2001 15:41:29	0	Read DB: Error storing flow info in internal tables (1)
192.168.3.4	16-08-2001 15:41:25	0	Read DB: Error storing flow info in internal tables (1)
192.168.3.4	16-08-2001 15:41:19	0	Read DB: Error storing flow info in internal tables (1)
192.168.3.4	16-08-2001 15:41:15	0	Read DB: Error storing flow info in internal tables (1)

Figure 4-16: Eventlog.

4.5.7.2 Flow Monitoring

The flow monitor (available at Monitor → Flowmonitor) lists all running flows, which have an aggregation time specified. With the flow monitor it is possible to view the aggregated results as a graph, which were produced during the runtime of the flow. This is especially useful for probing flows. The aggregation time of results can be selected during the flow configuration and specifies the constant intervals, where the receiving daemons are reporting a result aggregate over the last aggregation period.

From the list of running flows, the monitor can be either configured within the current frame (integrated) or within an extra window (popup).

The flow monitor automatically updates the graph in specified intervals, which defaults to the specified aggregation period.

4.5.8 Flow Search Function

With the flow search function it is possible to select flows matching specific criteria. All selected criteria are AND-connected. Figure 4-17 shows the structure of the search criteria:

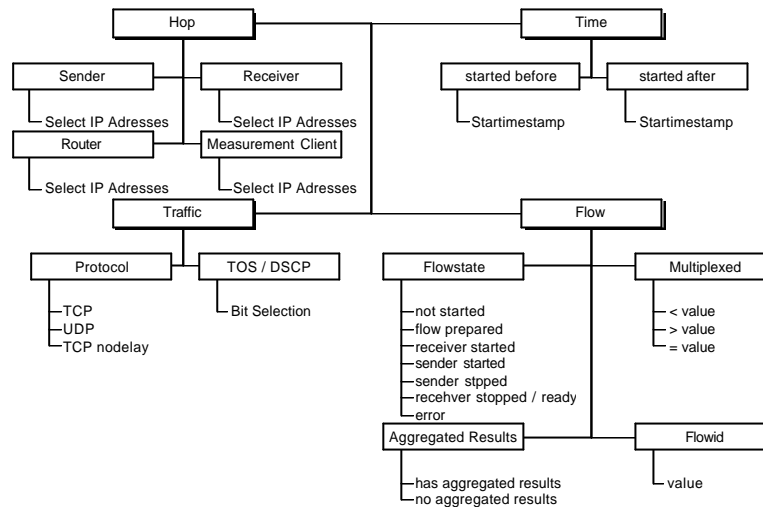


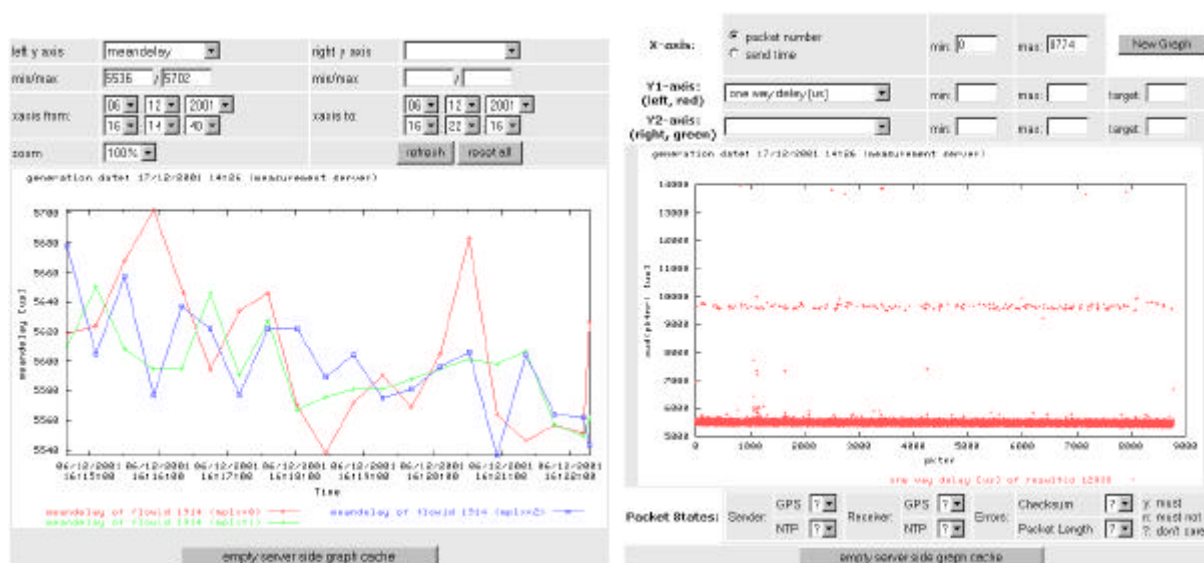
Figure 4-17 Structure of Flow Search Function

4.5.9 Measurement Results

Three types of measurement results are produced by the active measurement part of the measurement system:

- **Overall results** are produced to have a final result for the measurement flow. A flow has usually one overall result, except the flow has been specified as a multiplexed flow. Multiplexed flows have several overall results, one for each single flow.
- **Aggregated results** are produced to get an overview about the measurement results within specific time periods.
- **Raw data results** are produced to get a detailed view of the measurement results. Raw data means, that the sending and receiving timestamp as well as the packet size and a packet state are reported for every single measurement packet.

E.g. Figure 4-18 shows two graphs of the delay of a measurement flow. The measurement flow consisted of three multiplexed UDP flows, each with mean a sending rate about 150 kbps. Figure 4-18 (a) displays the mean delay of the aggregated data of the three UDP flows. Figure 4-18 (b) displays the according raw data (per packet) of one of the three multiplexed UDP flows.



(a) aggregated data result display

(b) raw data result display

Figure 4-18: Screenshot of DMA results

The results are only produced if they are requested by the user.

For overall and aggregated results the reported results are:

- Throughput
- Packet Loss
- Packet Loss Patterns (maximum loss burst, maximum loss distance, number of loss periods)
- Delay (mean delay, maximum delay, minimum delay)
- Delay variation (mean delay variation, maximum delay variation)

Except by using the flow monitor (see section 4.5.7.2) the results can be viewed when the measurement flows are ready. In general, the results of the active measurements can be browsed grouped by tests or by searching single flows. Also the results of router monitoring can be graphically displayed.

4.5.9.1 Result Browsing by Tests

Using this function (Results → Graphics → Ready / Running / Error / All / Empty Tests), the selected tests of the current user will be displayed in a table. From here the single flows of the tests can be viewed by using the '+' button at a specific test, which lists all the containing flows. With a further '+' the according overall results to the flow are shown. Now links are provided to view the graphs

or the data lists (as comma separated value list) of the flows. Also some statistics can be calculated and viewed, if raw data is available for this flow.

4.5.9.2 Result Browsing by Flows

In principle, this function (Results → search function) shows the same results as above, with the difference, that the flows are not grouped by their according tests, but can be selected individually. Flows are selected by using the search function of the GUI (see section 4.5.8). The list of matching flows are then displayed in a table. Each flow can be selected by using the checkbox. Afterwards the following functions can be applied:

- **Zoom to selected:** With this function, the same table is displayed but only the selected flows are listed.
- **Show selected:** This function displays the aggregated results of the selected flows as graph.
- **Delete selected:** This function deletes the selected flows with all according results and the path information.

4.5.9.3 Results of Router Monitoring

The results of router QoS monitoring include packet statistics for each specific flow for the ingress edge router, and traffic class statistics for all routers specified for monitoring. Also CPU usage of the routers is monitored.

The results can be viewed as follows:

- Cumulative number of total / dropped packets for a flow as a function of time
- Cumulative number of total / dropped packets for a traffic class as a function of time
- Cumulative number of in- and out- packets dropped in TCL3 and TCL4 as a function of time
- Mean queue length of TCL3/4 as a function of time
- CPU usage as a function of time

4.5.10 Outlook for the Second Trial

There are some features needed for the 2nd trial. To handle large tables it is necessary to reduce the output in the browser. Otherwise performance problems will occur. This problem will either be solved with a filter mechanism or with a “leaf through” mechanism.

5 Application-level Load Generators

Besides the enhancements of the integrated DMA load generators for the second trial two additional load generators are provided, which are described in this section. The load generators are not integral parts of the DMA, but are provided separately and are designed to co-exist with the DMA on the same machines simultaneously.

5.1 Voice Load Generator and Perceptual Evaluation

An additional measurement utility provided for the second trial is a voice load generator and the perceptual evaluation of the voice quality.

A sender establishes the connection to the corresponding receiver component by the use of H.323 and then sends the audio data via the Real-time Transport Protocol RTP. Both the sender and the receiver are executable programs which run separately.

5.1.1 *Functionality*

The functionality of this additional measurement utility is as follows: First of all, the user defines a new „Speech Quality Measurement Test Scenario“, for example by choosing a voice sample (wav-file). The signalling between sender and receiver is done by H.323. After having established the connection and having opened the logical channel, the RTP transmission of the voice sample takes place. Figure 5-1 gives an overview of this technical concept. A background net load can be provided by the DMA in order to get a degraded file.

When all the audio data is transmitted, the connection is terminated (again by H.323). Finally, the result process stores the transmitted and degraded voice sample and it can be compared with the original file. Here, one of the many algorithms for speech quality measurement is applied (see section 5.1.2).

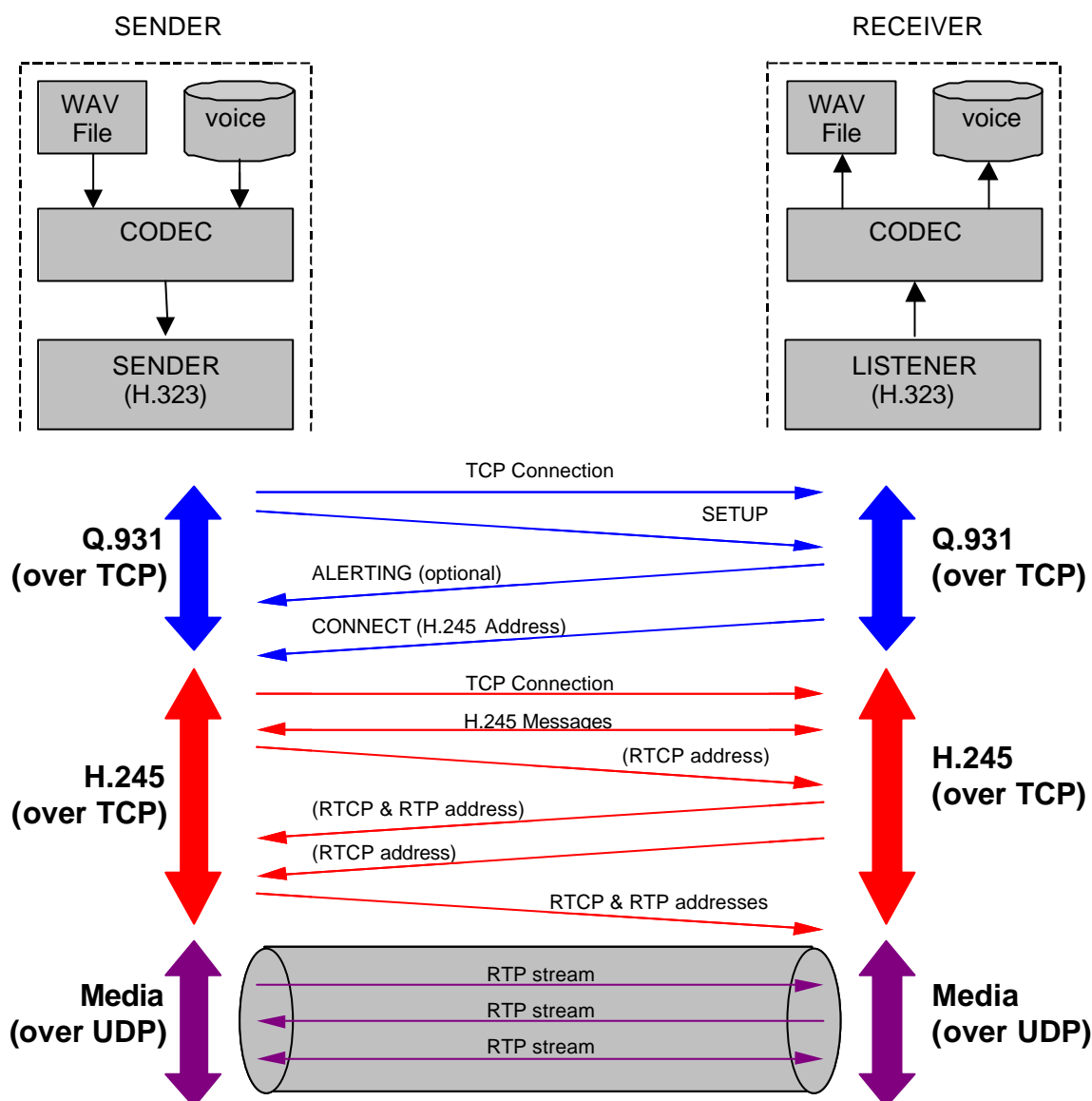


Figure 5-1: Process of a H.323 Connection

5.1.2 Perceptual Evaluation

There are many factors that influence voice quality, e.g. voice clarity, echo, time delay between spoken phrases, time clipping and so on. Of all these factors, *clarity* and *time delay* are often considered as the most important. Most of these factors (like time delay) are very objective parameters and can be effectively be measured and expressed (time delay in milliseconds), but voice clarity is a very subjective parameter.

There are two ways to “measure” voice clarity. The first significant technique used to measure voice quality was to actually use large numbers of human listeners to produce statistically valid subjective clarity scores. This technique is called Mean Opinion Scoring (MOS), where the mean value of large

volumes of human opinion scores is calculated. Scores reach from 5 (excellent voice quality) to 1 (bad voice quality), see Table 5-1:

Score	Quality of Speech	Effort required to understand the meaning of sentences
5	Excellent	Complete relaxation possible, no effort required.
4	Good	Attention necessary; no appreciable effort required.
3	Fair	Moderate effort required.
2	Poor	Considerable effort required.
1	Bad	No meaning understood with any feasible effort.

Table 5-1: MOS testing

MOS testing is subjective, expensive and inefficient. These drawbacks suggest that objective, automated and repeatable testing methods are needed for measuring subjective speech clarity. These techniques should take into account clarity's subjective and human perception and their results should show a high correlation to the real MOS results.

In the last years, several methods for objective speech clarity measurement were investigated. All of them try to figure out the clarity by calculating the perceptual distance between the input (original) and output (transmitted, degraded) signals. Table 5-2 gives an overview of current methods:

Name	Description	Company
PSQM	Perceptive Speech Quality Measurement	KPN
PAMS	Perceptual Analysis Measurement System	British Telecom (BT)
PESQ	Perceptive Evaluation of Speech Quality	KPN, BT
TOSQA	Telecommunication Objective Speech Quality Assessment	Deutsche Telekom

Table 5-2: Overview of methods for speech quality measurement

The voice quality component is going to use TOSQA as evaluation algorithm.

5.1.3 System requirements

The following system requirements have been agreed on:

- the program runs in an IP based environment

- the platform is LINUX
- The programming is done in C/C++
- the audio data is transmitted via RRP in a compressed mode
- the co-existence with the DMA is possible (background load)

5.2 Virtual Web User

In this section the implementation of an application, that generates web traffic in a network simulating real web usage behaviour patterns is shown. This application was implemented by a group of students within the framework of a project at the Polytechnic University of Salzburg. Linux was chosen as the operating system for the implementation. The application itself consists on the one hand of a small C++ program and on the other hand the apache web server in connection with various CGI-scripts. The necessary steps to determine the user behaviour are shown. Finally, the way of simulating user behaviour by the software is described.

5.2.1 Architecture

As depicted in Figure 5-2, the VWU architecture consists out of three main components. These components are described in the following paragraphs.

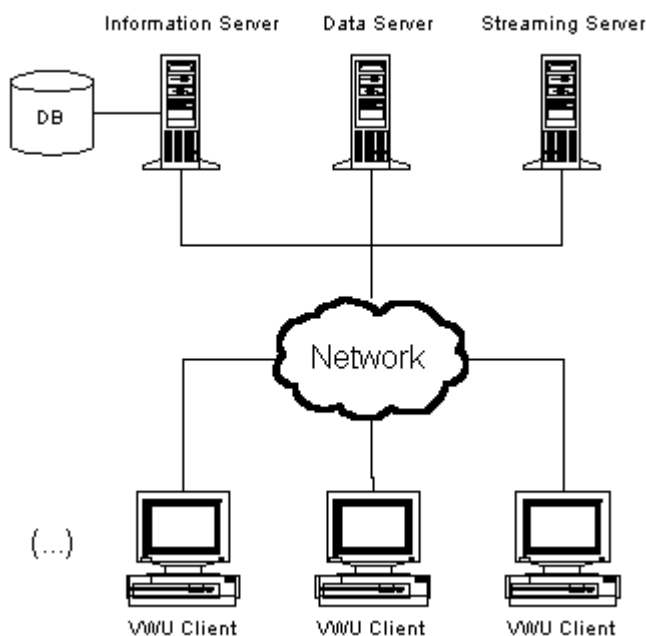


Figure 5-2: Virtual Web User Architecture

5.2.1.1 Information Server

The Information Server is central counterpart of the VWU system. Clients continuously connect to this server for many reasons. The database which the Information Server accesses, holds the configuration for each client within the VWU system. According to this configuration

5.2.1.2 Data Server

As might have expected, the Data Server generates HTTP traffic closely similar to an ordinary Web Server. The Client produces an HTTP request including the amount of data to be transmitted to. This information is retrieved from the Database respectively from the Information Server. However, the Data Server generates a HTTP packet including a small header and the demanded amount of random data. Besides, the sent data does not originate from a random number generator, but is loaded from the file 'data' in the current directory. This enables testing of various compression algorithms.

5.2.1.3 Streaming Server

The Streaming Server represents the functionality of an UDP-Streaming engine just like most Multimedia Servers in the internet environment. The Client produces a HTTP request including the desired bandwidth, the amount of data to be transmitted and additional parameters. This information is retrieved from the Database respectively from the Information Server. The Streaming Server on his part generates a HTTP acknowledge and initialises an UDP streaming thread. The desired bandwidth is achieved by a continuous loop of single packet transmission followed by a waiting period. The relationship of packet size and delay determines the resulting bandwidth. The maximum size of the UDP packets is limited to 8kBytes. Besides, the data transmitted to the client does not originate from a random number generator, but is loaded from the file 'data'. This enables testing of various compression algorithms.

5.2.1.4 Client

The client implements the end-host-part of the VWU project. Every client represents a 'virtual user' acting the way specified in the profile, which is stored on the information server.

5.2.2 Functionality

After successfully establishing a connection with the Information Server, the client announces itself as a new VWU client to this server, which in turn then provides information for the client about the next command to execute according to the assigned profile; e.g. it gives the client the command to sleep, or make a new data request etc.

In the next step, the client requires the information from the data/streaming server as specified in the afore-sent information from the Information Server. After finishing the request, the Client connects to the Information Server and requires new tasks to perform.

5.2.3 System Requirements

The Virtual Web User application is implemented in C++ and JAVA 1.1.8.

The client, that runs on many hosts in the network is programmed in C++. It does not have to run on high-performance machines.

All servers are written in JAVA and were tested with JRE 1.1.8. The performance for the servers should be high, since the generation of random data with specified properties for many clients is a challenging job for the machine.

6 MBAC Validation by Passive Measurements

For the MBAC validation it is proposed to install DAG capture cards to the trial sites, which are able to trace all network traffic on the links. The captured per-packet information can be compared to the measurement data gathered by router monitoring (to measure the mean rate on the outgoing interface), which is currently used to implement measurement-based admission control (MBAC).

To generate traffic, a new function of the synthetic flow generator will be provided (the “multiple flow generator”). With this function, reservation requests will be sent to the AC. If a reservation was successful, the flow will be started (or not, as configured) and the next request will be sent immediately.

6.1 Motivation

The measurements can be done directly by measuring the QoS values, which are of interest. But these direct measurements are difficult for QoS parameters that generate rare events, e.g. the packet loss process. Typical target values are $10^{-3} \dots 10^{-7}$. Maintaining a loss rate of 10^{-7} with 95% confidence interval requires 154 million packets, e.g. in case of a flow sending rate 3Mbit/s it takes 616 000 s (= 7 days). Some authors propose to measure other values, e.g. buffer occupancy [SiWa98] which are correlated with the loss rate to get information about the loss rate. Most of the models use the effective bandwidth formulas to estimate the loss probability and the number of flows, which can be admitted.

Figure 6-1 shows the two approaches. Both approaches are targeting at the loss probability of the QoS network.

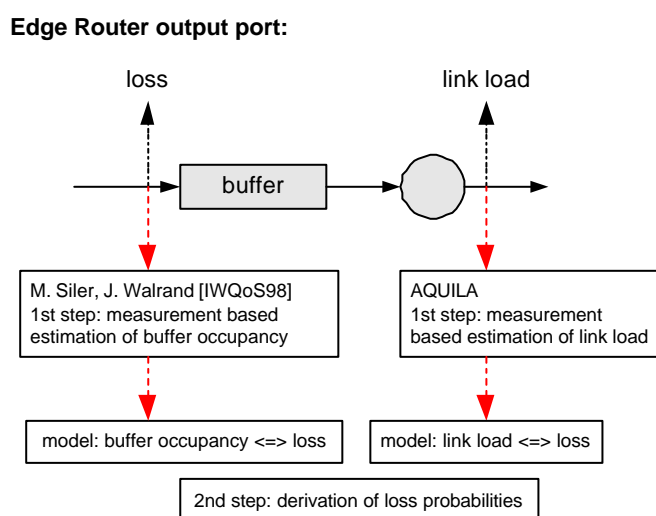


Figure 6-1 Two approaches for measurements to support MBAC

Measuring the load:

First the different time scales of a single flow is analysed:

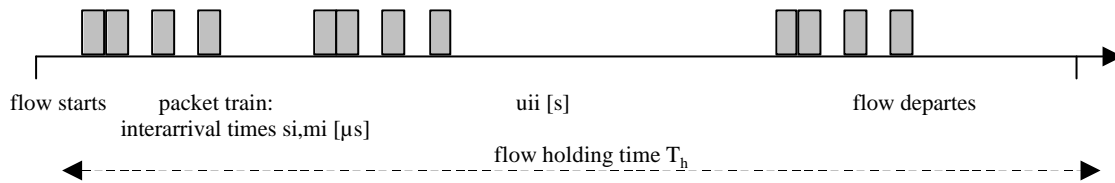


Figure 6-2: Single Flow Time Scales

- **uui**: user interaction inter-arrival time (e.g. time between the user got a web-page and his next click). Typical values [s...min]
- **mi**: medium interval between packet trains (e.g. tcp-window packet trains). Typical values [...ms]
- **si**: short interval between a packet pair of a packet train [t(last bit), t(first bit)] (e.g. caused by bottleneck links, multiplexing in routers). Typical values [ns...μs...ms] dependent from the link capacity

The flows can be modelled as ON/OFF sources. Figure 6-3 shows an example of such a flow with the following parameters (time unit 100 ms):

- constant + exponential distributed on-intervals: const = 10, mean_exp = 30 => mean = 40
- constant + exponential distributed off-intervals (uui): const = 30, mean_exp = 90 => mean = 120
- generated ON traffic is constant with a Gaussian noise: $20 + N(0,2)$

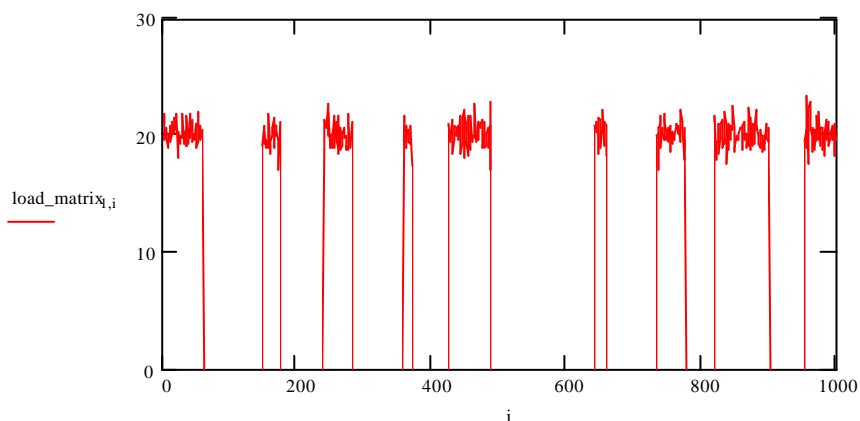


Figure 6-3: Single ON/OFF flow

Figure 6-4 shows the aggregated load and the mean load for 10 single ON/OFF flows.

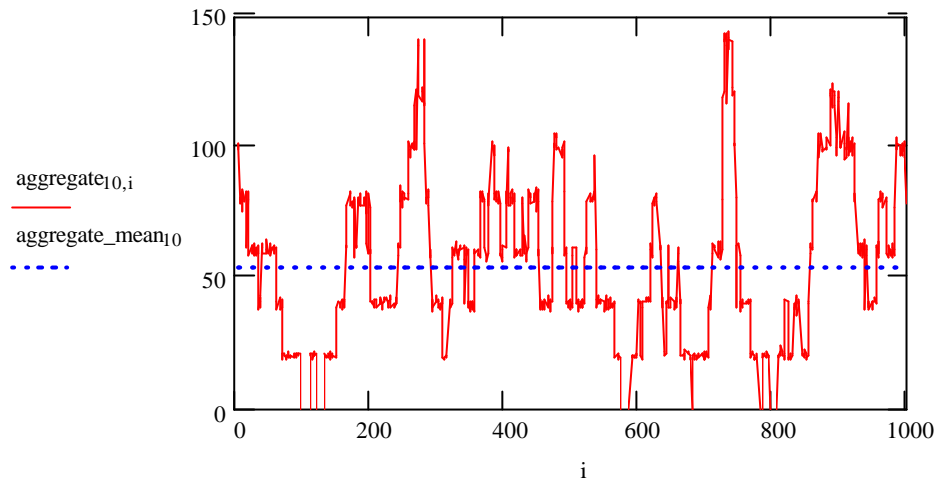


Figure 6-4: Aggregated ON/OFF sources

The aggregated flow figure shows the short-term peaks where the packet losses will occur. The AQUILA load measurements will be realised with time slices of $n \cdot 100$ ms, $n=1,2,\dots$. Therefore in the second trial passive monitoring hardware will be integrated for measuring also the short-term peaks.

6.2 Architecture

Figure 6-5 shows the architecture.

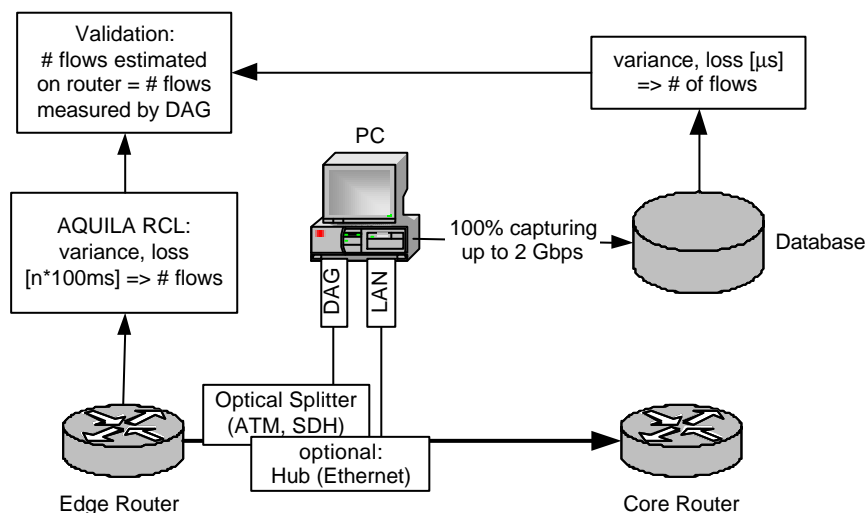


Figure 6-5: Passive measurements for MBAC validation

During the second trial the passive measurement will export the load measurements for off-line analysis. Parameter: time-slice.

For the validation of the MBAC algorithms a more detailed view on the network traffic of the access link than can be reached with the periodic polling of aggregated status information from routers is helpful or even necessary. Therefore a mechanism has to be found which is able to reliably capture all detail information of the network traffic for further processing. We propose to use the DAG capture card [DAG] for this detailed tracing of traffic. See section 12 for some details of the DAG capture card.

7 State of the Art

7.1 One-way Active Measurement Protocol (OWDP)

The IETF IP Performance Metrics (IPPM) working group has proposed draft standard metrics for one-way packet delay [RFC2679], loss [RFC 2680] and delay variation [IPDV] across Internet paths. There are now several measurement platforms that implement collection of these metrics:

- “SURVEYOR” of Advanced Network & Services Inc. [SURVEYOR],
- “TTM Test box” of the RIPE NCC [RIPE],
- “Distributed measurement architecture (DMA)” of the AQUILA-Project.

Currently there exists no standard that would allow initiation of test flows, exchange of test packets or results in an interoperable manner. With the increasingly wide availability of affordable global positioning system (GPS) and NTP-based time sources, very accurate time sources are available for the hosts (either directly or through their proximity to NTP primary stratum-1 time servers), by standardising a technique for collecting IPPM measurements, it could be possible to create an environment where IPPM metrics may be collected across a far broader mesh of Internet paths than is currently possible. One vision is of widespread deployment of open OWDP measurement systems that would make measurement of IPPM as commonplace as measurement of round-trip time using an ICMP-based tool like ping.

The suggested OWDP actually consists of two inter-related protocols:

- OWDP-Control and
- OWDP-Test.

OWDP-Control is responsible for Test-Connection setup and management. It sets up a TCP connection between the source and destination. Control messages are exchanged between the source and destination over the TCP connection. OWDP-Test is responsible for sending test packets from the source to the destination. UDP is used to send test packets. The architecture of the OWDP protocol is shown in Figure 7-1.

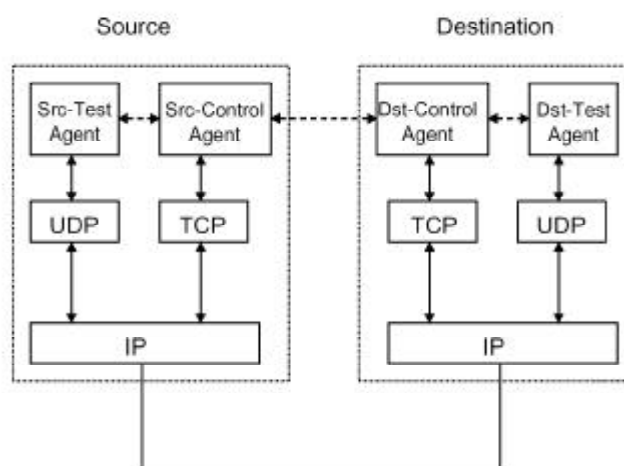


Figure 7-1: OWDP architecture

In the Surveyor Project [SURVEYOR] a first version of an OWDP based measurement system has been implemented.

7.2 Round-trip Delay versus One-way Delay

Due to the rapid growing of the Internet in all sectors of life, it is increasingly becoming important to accurately measure the communication performance between two Internet sites. Among others, delay, delay variation / jitter and packet loss are the three most important performance measures.

Traditionally, PING was used to measure these delay and loss related variables between two sites. The problem with PING is that it only measures the round-trip delay as opposed to one way delay. Dividing the round trip delay by two gives a crude estimate of one-way delay and grossly inaccurate estimate for asymmetric paths. DTA has performed several measurements with the AQUILA-Distributed Measurement System in the Internet. Figure 7-2 shows the measurement configuration.

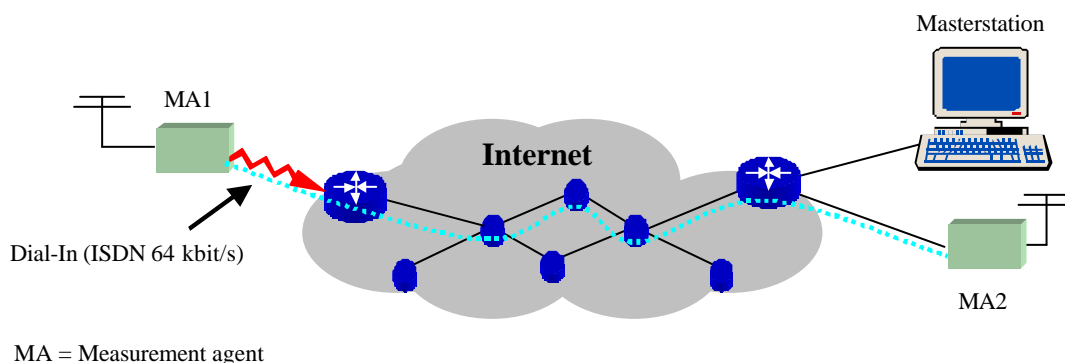


Figure 7-2: Measurement configuration

Measurement parameters:	
MA1:	Dial in in Munich
MA2:	Darmstadt
Packet length:	150 Byte/constant
Mean sendinterval:	50 ms /exponential
Mean throughput:	~24000 bit/s
Number of packets:	12.000

The following figures show the measurement results of the described scenarios.

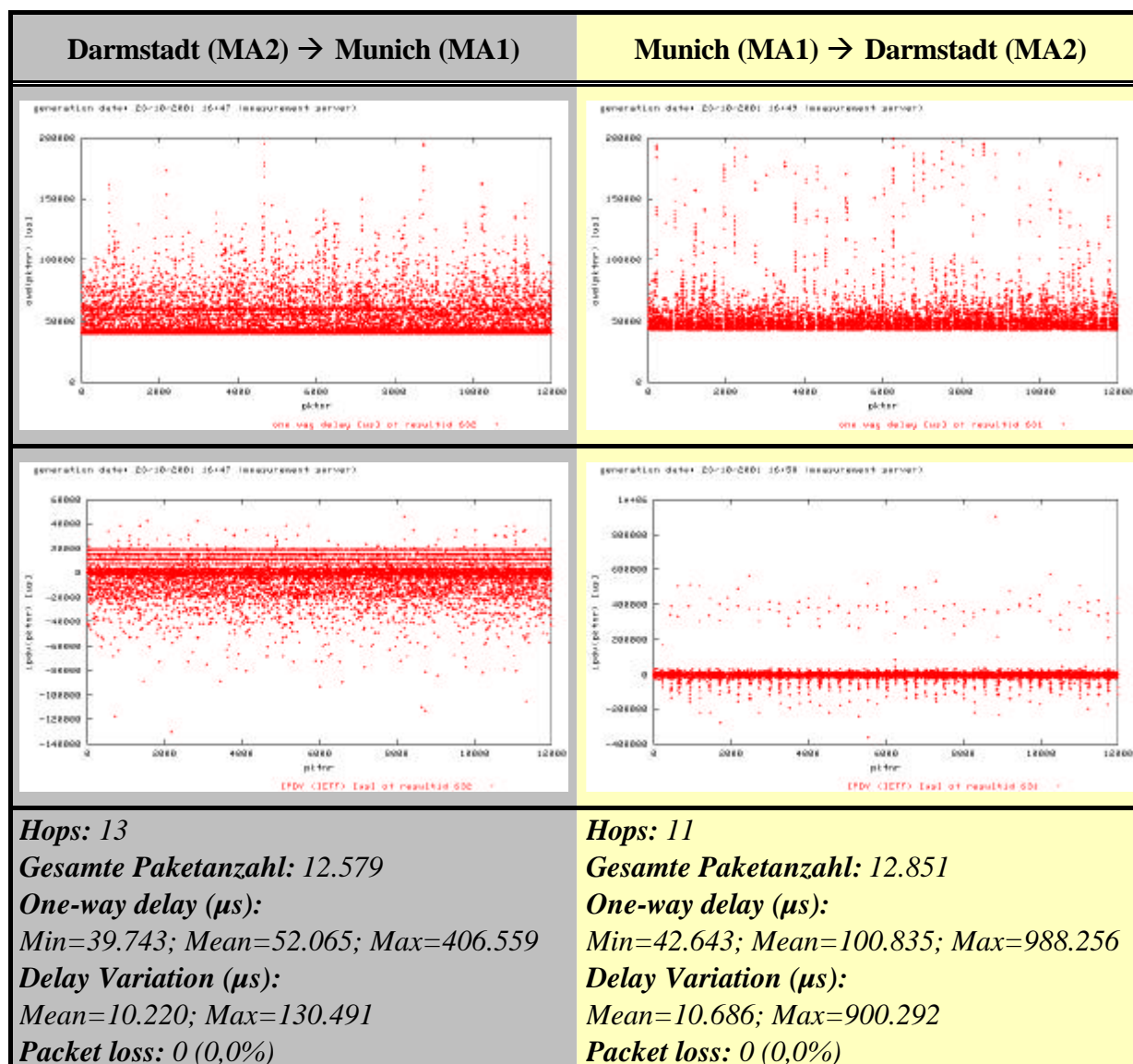


Figure 7-3: Measurement results

All measurements showed that the bi-directional connections have been routed through different routes. In the direction Darmstadt → Munich 13 hops were involved and in the opposite direction only 11 hops. This and the different network load in each direction leads to a mean one-way delay of **52 ms** for the direction Darmstadt → Munich and **100 ms** for the opposite direction.

→ The results show that it is essential for QoS monitoring to use one way measurement results instead of the roundtrip (“ping”) based results.

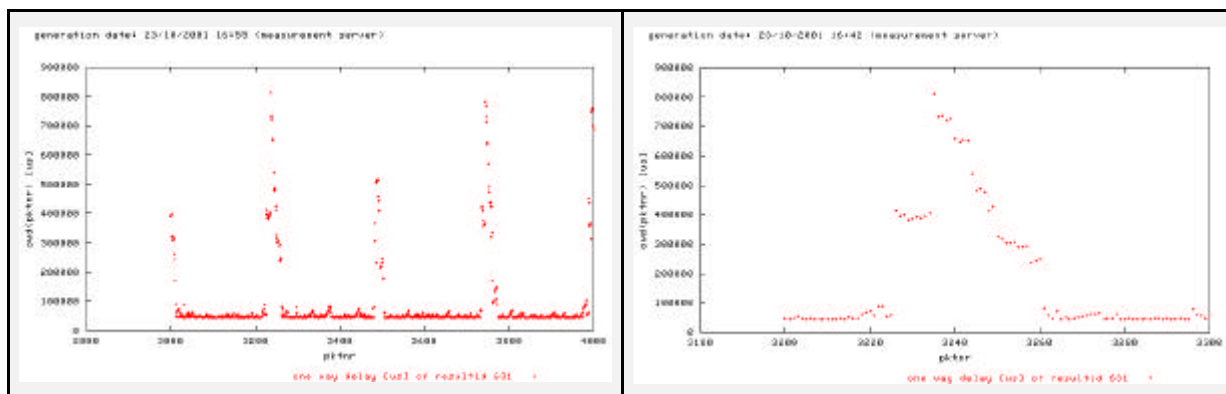


Figure 7-4: One-way delay (high-resolution view)

Another interesting outcome of the measurements were the regular patterns in the IP delay-variation and One-way delay graphs for the direction Munich → Darmstadt (see Figure 7-3 and Figure 7-4). If the one-way delay measurement results are displayed at a higher resolution you can also see a similar pattern. The reason for this pattern could be the behaviour of network components in (temporary) overload situations, but this assumption has to be verified with additional measurements.

8 Abbreviations

AQUILA	Adaptive Resource Control for QoS Using an IP-based Layered Architecture
DBAC	Declarative Based Admission Control
DMA	Distributed Measurement Architecture
EAT	End-user Application Toolkit
GPS	Global Positioning System
GUI	Graphical User Interface
IP	Internet Protocol
MBAC	Measurement Based Admission Control
NTP	Network Time Protocol
QoS	Quality of Service
RCL	Resource Control Layer
XML	Extensible Mark-up Language

9 References

- [APACHE] The Apache Software Foundation. Web page: <http://www.apache.org>
- [BSUB98] M. S. Borella, D. Swider, S. Uludag, G. B. Brewster: Internet Packet Loss: Measurement and Implications for End-to-End QoS. Proceedings, International Conference on Parallel Processing, Aug. 1998.
- [Clear00] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson. Design principles for accurate passive measurement. Proceedings of PAM2000, Hamilton, New Zealand, April 2000.
- [D1202] M. Winter et al: IST-1999-10077-WP1.2-SAG-1202-PU-O/b1, System architecture and specification for the second trial
- [D1302] S. Salsano et al: IST-1999-10077-WP1.3-COR-1302-PU-O/b1, Specification of traffic handling for the second trial
- [D2203] F. Fünfstück et al: IST-1999-10077-WP2.2-TUD-2203-PU-R/b0, User Guide for End-user Application Toolkit
- [D2301] F. Strohmeier et al: IST-1999-10077-WP2.3-SPU-2301-PU-R/b0, Report on the development of measurement utilities for the first trial
- [D3101] C. Tsetsekas et al: IST-1999-10077-WP3.1-NTU-3101-PU-R/b0, First Trial Integration Report
- [D3201] Z. Kopertowski et al: IST-1999-10077-WP3.2-TPS-3201-PU-R/b0, First Trial Report
- [DAG] The DAG Project. <http://dag.cs.waikato.ac.nz/>
- [DAGIG] DAG 3 Installation Guide.
http://dag.cs.waikato.ac.nz/dag/docs/dig_v2.1.pdf
- [Fisz78] M. Fisz: Wahrscheinlichkeitsrechnung und mathematische Statistik. Akademie Verlag Berlin, 1978
- [Hube99] A. Huber: The use of sampling in network measurements. Master Thesis, University of Salzburg, March 1999
- [IPDV] C. Demichelis, P. Chimento: Instantaneous Packet Delay Variation Metric for IPPM. Internet Draft, Februar y 2001 <expired>. <http://www.ietf.org/internet-drafts/draft-ietf-ippm-ipdv-07.txt>

- [KiWo56] J. Kiefer, J. Wolfowitz: Sequential tests of hypotheses about the mean occurrence time of a continuous parameter Poisson process. Naval Research Logistics Quarterly 3, No. 3, 1956.
- [Mich01] J. Micheel, H-W. Braun, and I. Graham. Storage and bandwidth requirements for passive Internet header traces. Proceedings of the NRDM Workshop, Santa Barbara, California, May 2001.
- [MYSQL] The MySQL web page. <http://www.mysql.com>
- [Papa79] P. Papantoni-Kazakos: Algorithms for Monitoring Changes in Quality of Communication Links. IEEE Transactions on Communications, Vol. COM-27, No. 4, April 1979
- [PHP] PHP Project of the Apache Software Foundation. A Hypertext Pre-processor. Web page: <http://www.php.net>
- [PKC96] Park, K., Kim, G., Crovella, M. „On the relationship between file sizes, transport protocols, and self-similar network traffic“, Proceedings of the Fourth International Conference on Network Protocols, pages 171-180, October 1996.
- [RFC2679] G. Almes, S. Kalidindi, M. Zekauskas: A One-way Delay Metric for IPPM. RFC 2679, September 1999. <http://www.ietf.org/rfc/rfc2679.txt>
- [RFC2680] G. Almes, S. Kalidindi, M. Zekauskas: A One-way Packet Loss Metric for IPPM. RFC 2680, September 1999. <http://www.ietf.org/rfc/rfc2680.txt>
- [RIPE] RIPE-NCC/Test Traffic Measurements. Information available at: <http://www.ripe.net/ripenncc/mem-services/ttm/>
- [SiWa98] M. Siler, J. Walrand: On-line Measurement of QoS for Call Admission Control. Proc. IWQoS'98, May 1998. <http://www-networking.eecs.berkeley.edu/~siler/papers/siler-iwqos98.ps>
- [Suri89] R. Suri: Perturbation analysis: The state of the art and research issues explained via the GI/G/1 Queue. Proc. IEEE, Vol 77, No1, January 1989, 114-137
- [SURVEYOR] Advanced Network & Services and the Common Solutions Group (CSG)'s Surveyor project web page, June 1999. <http://www.advanced.org/surveyor/>
- [TRACEROUTE] LBNL's Network Research Group's Traceroute: <http://www-nrg.ee.lbl.gov/>
- [ZDPS01] Y. Zhang, N. Duffield, V. Paxson, S. Shenker: On the Constancy of Internet Path Properties. ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco, USA. November 2001.

- [ZPS00] Y. Zhang, V. Paxson, S. Shenker: The Stationarity of Internet Path Properties: Routing, Loss, and Throughput. ACIRI Technical Report May 2000.
<http://www.aciri.org/vern/papers/stationarity-May00.ps.gz>

10 Annex A: Accurate Estimation of the Loss Rate

10.1 Independent Loss Events $P(\text{Loss}=p)$

The following analysis shows how the confidence interval can be calculated. Results about the sample size are derived. We differentiate between "left" and "right" side interval because it is less important to find

- the left side interval for the target loss rate e.g. 10^{-6} (better is not critical)
- the right side interval for the tolerance loss rate e.g. $2 \cdot 10^{-6}$

In case this differentiation is not required both sides will be considered.

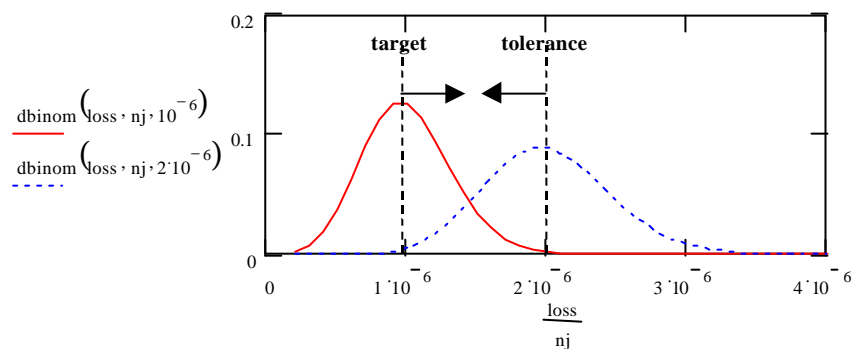


Figure 10-1: Left and Right Side Intervals of Target and Tolerance Loss Rates

10.1.1 Calculation of the Number of Measurements for a % Right Side Confidence Interval

E.g.: measured {lost, sent}: {4,500}, {6,355}, {5,550}, {0,655}

j number of aggregated loss event statistics $j = 4$

n_i number of monitored packets per statistics {500,355,550,655}

r_i lost packets 4,6,5,0

p loss probability e.g. 0.001

r_i is binomial distributed in $[0, n_i]$

[Fisz78]: The sum of losses $r_\Sigma = r_1 + r_2 + \dots + r_j$ is binomial distributed in $[0, \Sigma n_i = n_1 + n_2 + \dots + n_j] \Rightarrow$ mean_loss_rate = r/n is the transformed binomial distribution in $[0,1]$.

Without simplification we set $n = n_i = \text{constant}$; $i = 1, \dots, j$

E.g.:

$p := 0.00$ $q := 1 - p$ $n := 2000$ $j := 300$ $\Sigma n_i := 0 \dots n \cdot j$

We assume that all aggregates have the same length n ; it can be shown, that this is not a simplification, because in the formulas all n_i and r_i will be added. The target loss rate is the lower value which has to be compared with the higher tolerance value, which is equal to the right side of the low target value.

We have: true unknown mean number of losses: $m_{\text{mean}} = n \cdot j \cdot p$

Estimation of the right side α interval:

Binomial distribution for $p=10^{-5}$ and $2000 \cdot 300$ sent packets (Figure 10-2 and Figure 10-3 show the binomial distribution)

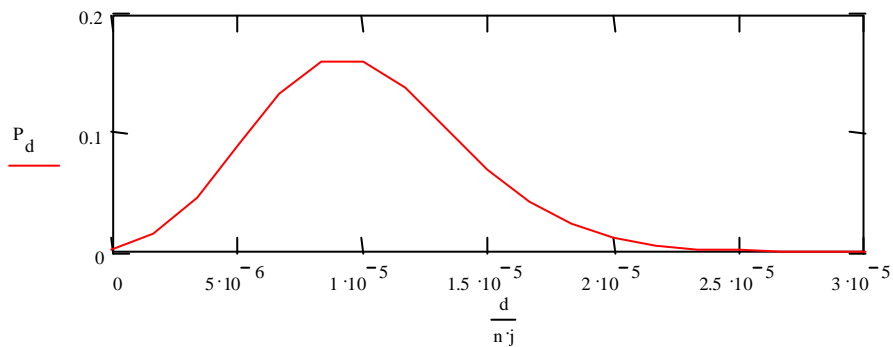


Figure 10-2: Probability Function

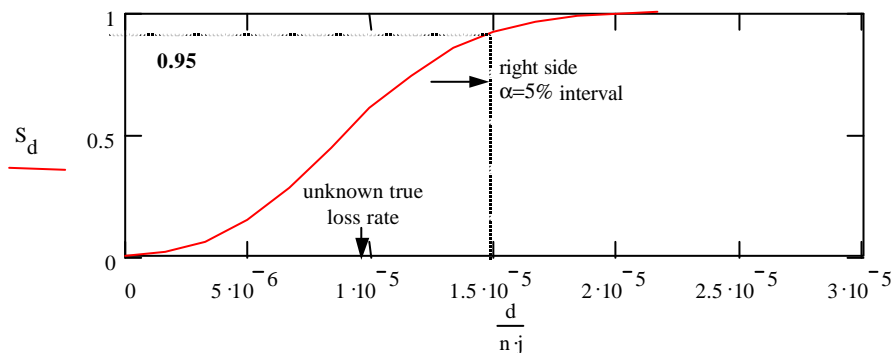


Figure 10-3: Distribution Function

Table 10-1 shows the right side α -intervals for different loss probabilities p and sample sizes $n \cdot j$.

$n \cdot j$	$2/p$	$3/p$	$4/p$	$5/p$
right side α -interval	$2p$	$1.9p$	$1.75p$	$1.7p$

Table 10-1: α - Intervals for Different Loss Probabilities and Sample Sizes

Example: For a loss probability of 10^{-5} we need $5 \cdot 10^5$ samples to get the $1.7p$ right side confidence interval.

10.1.2 Calculation of the Number of Measurements for a % Left Side Confidence Interval

Figure 10-4 shows the probability distribution of the measured loss rate for a true tolerance loss probability $p = 0.001$ and $n \cdot j = 2/p = 2000$. The probability for a measured zero-loss rate $r = 0$ is 0.135:

- $P(\text{measured loss rate} = 0) = 0.135$

Obviously the left side 5% confidence interval of this measurement overlaps with all possible right side confidence intervals of each target loss. Therefore we will analyse the possibilities to decrease the left side α -interval by increasing the sample size.

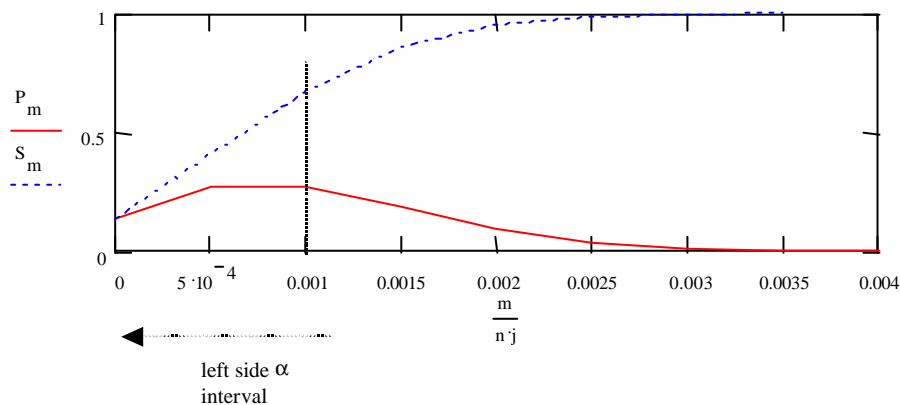


Figure 10-4: Left Side 5% Confidence Interval Overlaps with Right Side Intervals

We increase the sample size to $4/p = 4000$. Figure 10-5 shows the distribution of the measured loss rate:

- $P(\text{measured loss rate} = 0) = 0.018$ (decreased!)
- $P(\text{measured loss rate} < 2.5 \cdot 10^{-4}) = 0.09$

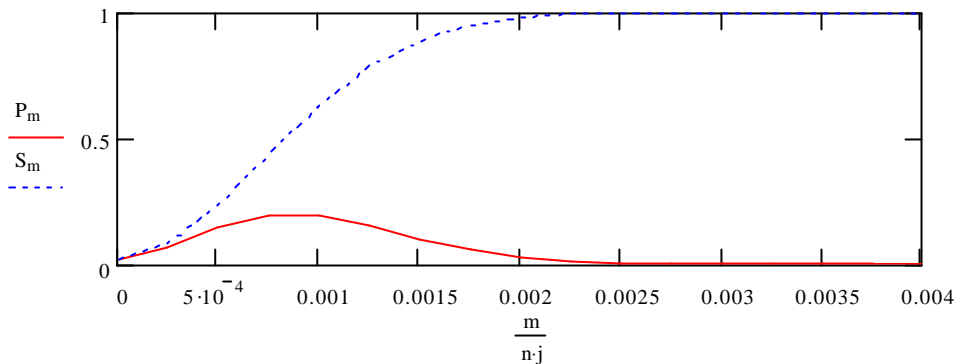


Figure 10-5: Decreased 5% Left Side Overlapping

10.2 Dependent Loss Events

For dependent loss events the classical well known approach of aggregation of measurements into batches should be used. AQUILA measurements will be done to analyse the loss patterns to choose the appropriate method.

Summary: The accurate calculation of loss rates needs a large number of samples. In case of trials in a laboratory or measurements in high-speed links this may be a realistic scenario. For operational management purposes under changing system load conditions we need fast algorithms to estimate the actual loss rate. Therefore another approach is analysed in annex B.

11 Annex B: Hypothesis Testing between Two Loss Rates

11.1 Maximum Likelihood Estimation of the Crossing of QoS Levels

We assume that the transmission link can be in one of the two states $P(\text{loss}) = p$ or $P(\text{Loss}) = p + \tau$.

The process "under control" is quasi-static. This means that there is enough time to detect the change (decrease) in quality in order to influence the network (e.g. resource allocation, rejection of new flows) and to detect the improvement of the quality. Figure 3-2 shows the general structure of the control process. The system (random environment) switches between two QoS-values: the target value p and the value $p + \tau$ (QoS decrease).

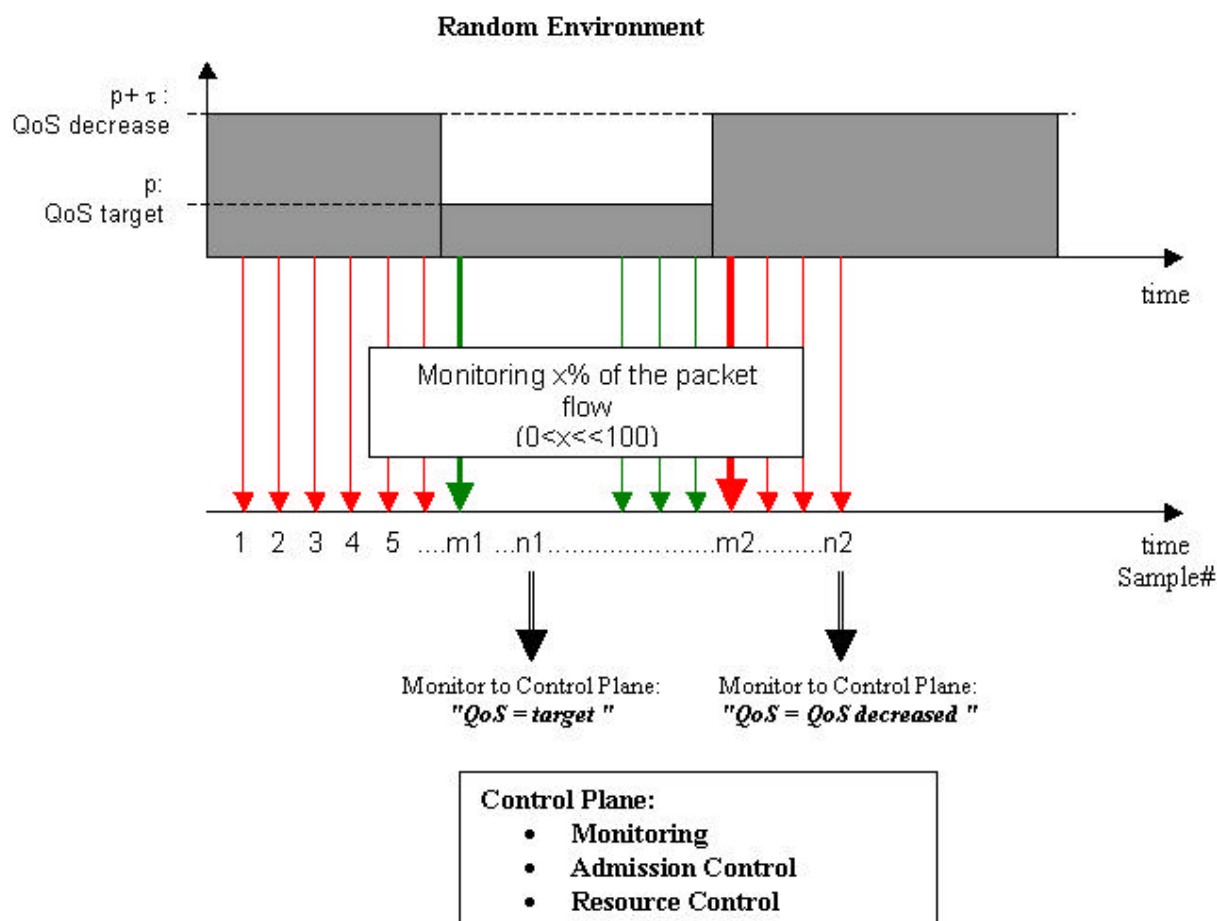


Figure 11-1: General structure of the control process

The following events are depicted in Figure 11-1:

- $m1$: the first QoS shift occurred before $m1$
- $n1$: the first QoS shift was detected by the monitor at $n1$
- $m2$: the second QoS shift occurred before $m2$
- $n2$: the second QoS shift was detected by the monitor at $n2$

The validation of the proposed algorithms must be done by using the following parameters:

- probability for "false alarm": a QoS change is monitored without QoS shift
- latency of alarm: the time between the shift occurred and this event is monitored, e.g. $latency1 = n1 - m1$.

Obviously the quasi-static approach is well suited for such control systems, where the latency of the detection of a shift is relatively small in comparison with the duration of a static interval.

This model assumes that loss events are independent [ZPS00]. See section 11.2 for more details.

The basic model works with single event information (e.g. "successful" or "unsuccessful" transmission of each single packet). The basic algorithm as well as its expansion for aggregated event information is described in section 11.1.2.

The problem will be analysed first for the case of monitoring changes between two possible QoS-values p and $p+\tau$. After this the more realistic case of detecting changes between QoS intervals $(0,p]$ and $[p,p+\tau)$ will be investigated.

11.1.1 Simulation Example of the Maximum Likelihood Approach

This problem was analysed by [Papa79]. There is the assumption that we have a sample of size n .

$$x^n = \{x_i ; i=1, \dots, n\}$$

This sample can be seen as a binary sequence of monitoring results:

$x_i = 0$ for "successful" transmission (no packet loss, delay < target_delay, ...)

$x_i = 1$ for "unsuccessful" transmission (packet loss, delay > target_delay, ...)

The simulated system started with a low loss probability $p = 0.1$, after 10000 packets the loss probability increased to $p+\tau = 0.15$ because of additional load, after further 10000 packets the loss rate felt back to the initial low loss probability.

Figure 11-2 shows the used simulation model.

The analysis of QoS change detection was done for the two monitoring methods used within AQUILA:

- Active monitoring (probing)
- Passive monitoring

11.1.1.1 QoS Change Detection with Active Monitoring

The density of the probing flow was set to 10% of the application flow.

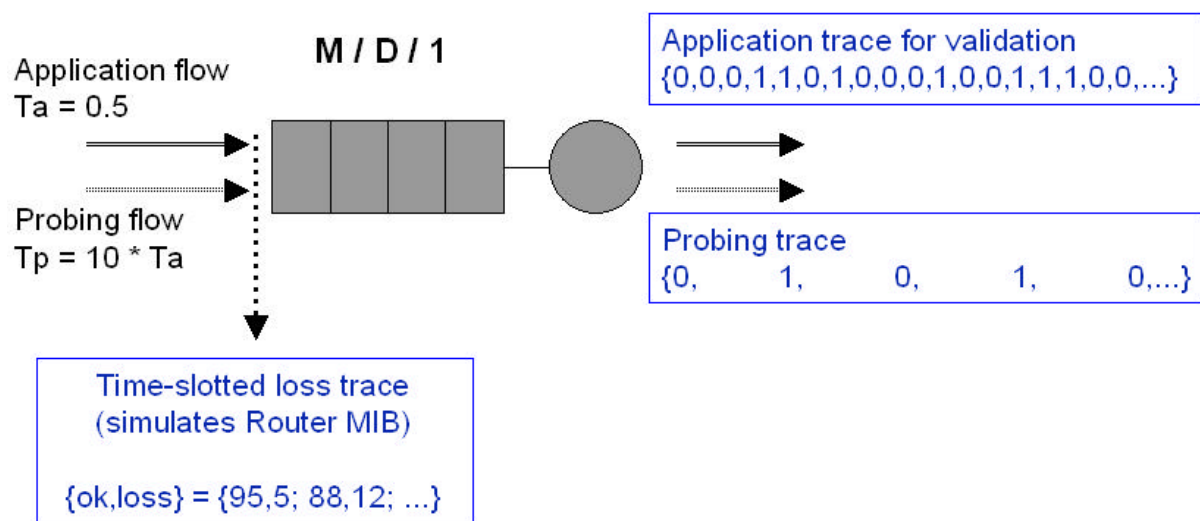


Figure 11-2: Simulation model

The shift of the two loss probabilities p and $p+\tau$ was achieved by increasing and decreasing the rate of the incoming Poisson application flow (see Figure 11-3).

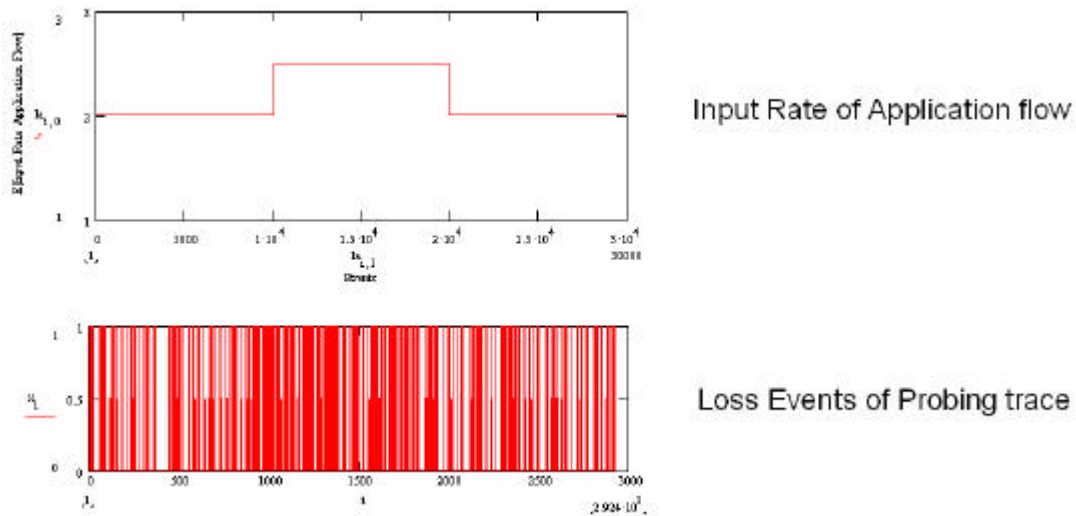


Figure 11-3: Rate of input flow and loss events of probing flow

To detect changes of the QoS level, the method of Papantoni-Kazakos [Papa78] was used. This method uses a metric, which increases when a change from the “good” QoS level (p) to the “bad” QoS level ($p+\tau$) happens and decreases when a shift in the other direction happens. The progression of this metric (S_k) for our simulation is shown in Figure 11-4.

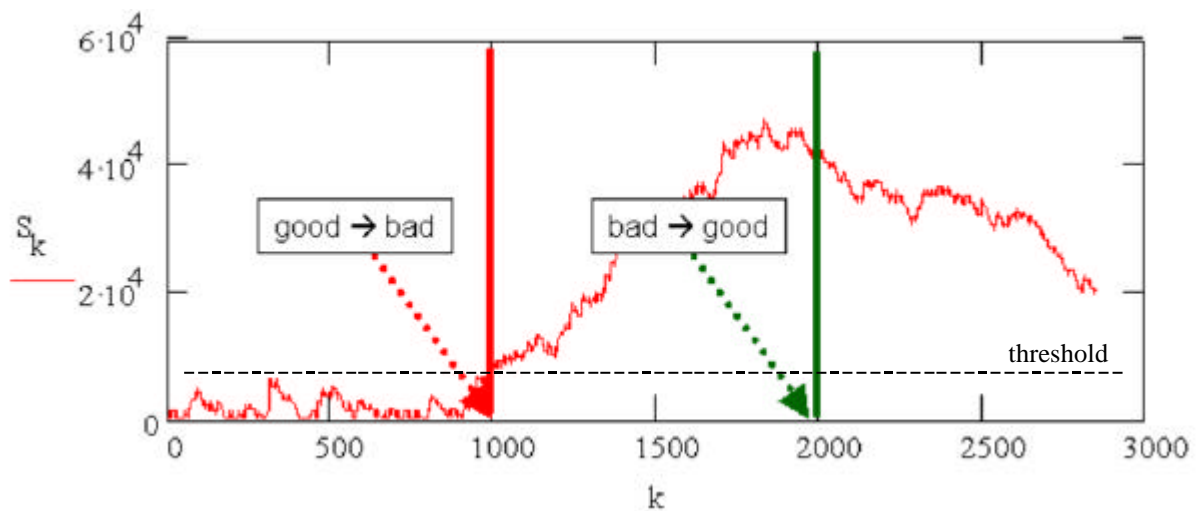


Figure 11-4: Detection of QoS changes; S_k : MLR monitor function good → bad

The thresholds should be optimised to avoid late or early alarm:

- threshold high: late alarm
- threshold low: early alarm

To unbiased the detection of a QoS shift from “bad” to “good”, a double-sided algorithm can be used. Figure 11-5 shows the application of the double-sided algorithm for the detection of QoS changes.

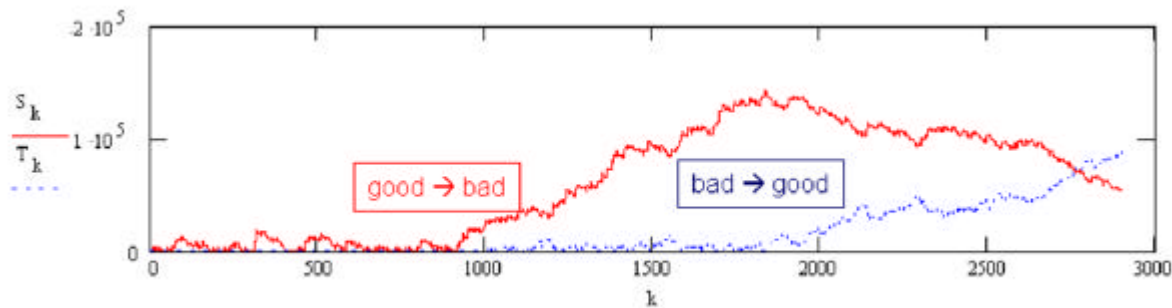


Figure 11-5: Double-sided algorithm; S_k : good \rightarrow bad monitor, T_k : bad \rightarrow good monitor

11.1.1.2 QoS Change Detection with Passive Monitoring

A second step in our simulation was the analysis of losses as they are reported from the Router monitoring tool which gathers this information either from the Router MIB or Router CLI (passive monitoring).

The upper graphics of Figure 11-6 shows the loss values periodically reported from the passive monitoring. It is rather hard to find exact points where a change in the QoS level could have been happened. Of course we know that the change happened at sampling point 50 from good \rightarrow bad and at sampling point 100 in the other direction. The metric of the adapted algorithm (W_k) clearly detects this point.

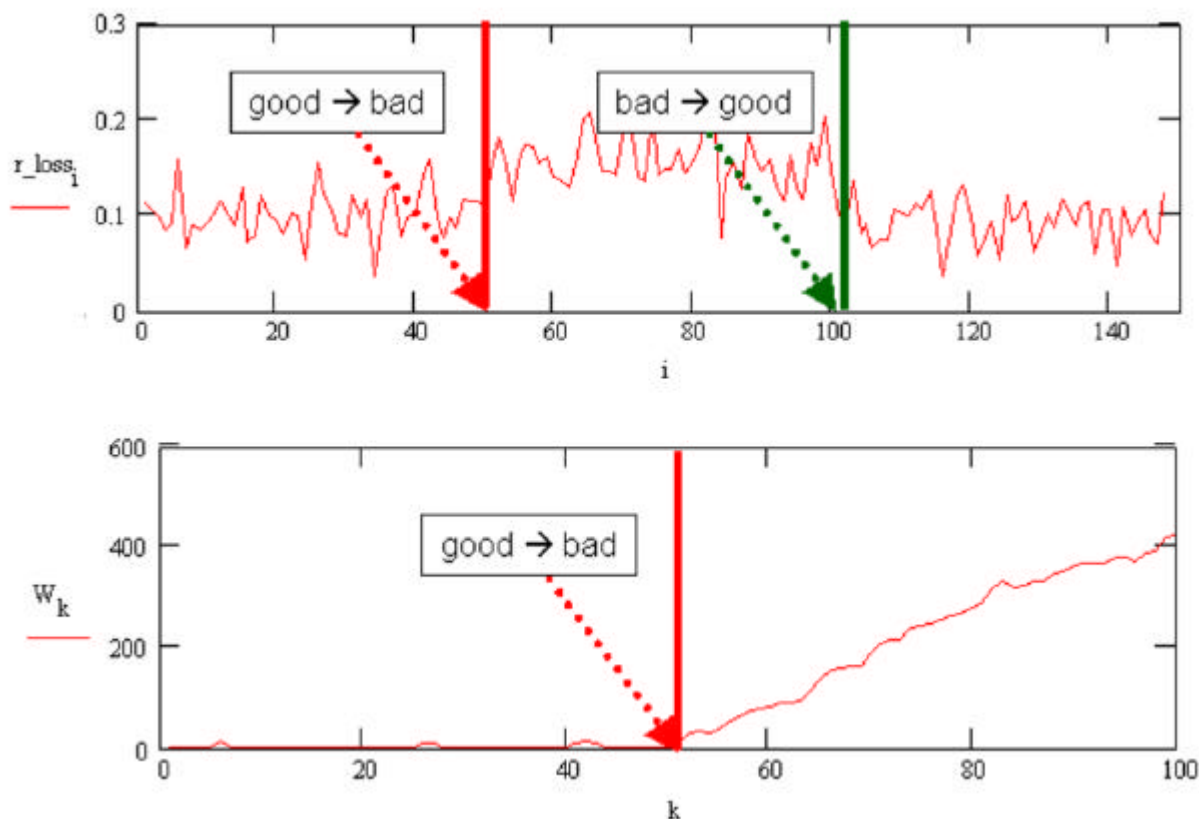


Figure 11-6: Analysing Router MIB information concerning losses

11.1.2 Maximum Likelihood Estimation for Aggregated Event Information

11.1.2.1 Algorithm of Papantoni-Kazakos

There is the assumption that we have a sample of size n .

$$x^n = \{x_i; i=1, \dots, n\}$$

This sample can be seen as a binary sequence of monitoring results:

$x_i = 0$ for "successful" transmission (no packet loss, delay < target_delay, ...)

$x_i = 1$ for "unsuccessful" transmission (packet loss, delay > target_delay, ...)

Now the $n+1$ hypotheses can be stated and the decision has to be made in favour for one of them:

H_i : the shift occurred just before the $(i+1)^{\text{th}}$ sample; $i = 0, 1, \dots, n$

For

$P(x^n | p, m, p+\tau)$: conditional probability of observing the sample sequence under the condition that the $p \rightarrow p+\tau$ shift occurred just before the $(m+1)^{th}$ sample

the probability for a sample, where the shift occurred before m is

$$\prod_{i=0}^m p^{x_i} (1-p)^{1-x_i} \prod_{i=m+1}^n (p+\tau)^{x_i} (1-p-\tau)^{1-x_i}$$

$x_i = 0,1$; before m the probabilities for x_i are $p, 1-p$ after m they are $p+\tau, 1-p-\tau$

We get the MLR-function

$$\ln P\{x^n | p, m, p+\tau\} = \sum_{i=0}^m \ln p^{x_i} (1-p)^{1-x_i} + \sum_{i=m+1}^n \ln (p+\tau)^{x_i} (1-p-\tau)^{1-x_i}$$

The ideal observer determines the m , where the maximum of the MLR function is reached:

H_m :

$$MLA = \sum_{i=0}^m [x_i - h(p, p+\tau)] = \min_{0 \leq k \leq n} \sum_{i=0}^k [x_i - h(p, p+\tau)]$$

where

$$h(p, p+\tau) = \frac{\ln \frac{1-p}{1-p-\tau}}{\ln \frac{(1-p)(p+\tau)}{p(1-p-\tau)}}$$

The algorithm for the calculation of the alarm event at sample point n is :

Stop at the first step n such that : $W_n \geq d > 0$

where $W_0 = 0$, $W_k = \max (0, W_{k-1} + z_k)$

and $z_i = x_i - h(p, p+\tau)$

For more practical application the algorithm can be mapped to an algorithm with natural numbers.
We define:

$\eta(p, p+\tau) = q/s$, q, s are natural numbers

$y_i = s z_i = s [x_i - \eta(p, p+\tau)]$

The range of y_i is $\{s-q, -q\}$

Now the modified stopping algorithm is

Start at level zero

Stop at the first step n such that : $S_n \geq t > 0$

where $S_0 = 0$, $S_k = \max (0, S_{k-1} + y_k)$; $k \geq 1$

11.1.2.2 General Approach for Aggregated Measurements

The original algorithm has been modified so that it can be applied to aggregated measurement information like router statistics loss rate obtained by the Router Monitoring tool.

The algorithm is extended for the use of batches of size t , considering the number of “successful” and “unsuccessful” packet transmissions within these batches. For this reason, a binomial distribution is assumed. All events within the batches are assumed to be independent Bernoulli-type events.

The known ML approach is being applied to n batches, and it is assumed that the QoS shift from $p \rightarrow p + \eta$ happened at batch m (τ was replaced with η).

We use

x_i number of “unsuccessful” packet transmissions within batch i

$t - x_i$ number of “successful” packet transmissions within batch i

$$\begin{aligned}
 MLA &= \max_{0 \leq k \leq n} \left[\sum_{i=0}^k \log \frac{t!}{x_i!(t-x_i)!} p^{x_i} (1-p)^{t-x_i} \right. \\
 &\quad \left. + \sum_{i=k+1}^n \log \frac{t!}{x_i!(t-x_i)!} (p+\eta)^{x_i} (1-p-\eta)^{t-x_i} \right] \\
 &= \max_{0 \leq k \leq n} \left[\sum_{i=0}^k \log \frac{t!}{x_i!(t-x_i)!} p^{x_i} (1-p)^{t-x_i} \right. \\
 &\quad \left. + \sum_{i=0}^n \log \frac{t!}{x_i!(t-x_i)!} (p+\eta)^{x_i} (1-p-\eta)^{t-x_i} \right. \\
 &\quad \left. - \sum_{i=0}^k \log \frac{t!}{x_i!(t-x_i)!} (p+\eta)^{x_i} (1-p-\eta)^{t-x_i} \right]
 \end{aligned}$$

$$\begin{aligned}
MLA &= \max_{0 \leq k \leq n} \left[\sum_{i=0}^k \log \frac{\frac{t!}{x_i!(t-x_i)!} p^{x_i} (1-p)^{t-x_i}}{\frac{t!}{x_i!(t-x_i)!} (p+\eta)^{x_i} (1-p-\eta)^{t-x_i}} \right] \\
&= \max_{0 \leq k \leq n} \left[\sum_{i=0}^k (x_i \log p + (t-x_i) \log (1-p) - x_i \log (p+\eta) - (t-x_i) \log (1-p-\eta)) \right] \\
&= \max_{0 \leq k \leq n} \left[\sum_{i=0}^k \left(x_i \log \frac{p(1-p-\eta)}{(1-p)(p+\eta)} + t \log \frac{(1-p)}{(1-p-\eta)} \right) \right] \\
&= \min_{0 \leq k \leq n} \left[\sum_{i=0}^k \left(x_i \log \frac{(1-p)(p+\eta)}{p(1-p-\eta)} - t \log \frac{(1-p)}{(1-p-\eta)} \right) \right] \\
&= \min_{0 \leq k \leq n} \left[\sum_{i=0}^k \left(\frac{x_i \log \frac{(1-p)(p+\eta)}{p(1-p-\eta)}}{\log \frac{(1-p)(p+\eta)}{p(1-p-\eta)}} - \frac{t \log \frac{(1-p)}{(1-p-\eta)}}{\log \frac{(1-p)(p+\eta)}{p(1-p-\eta)}} \right) \right]
\end{aligned}$$

When using the batch method we get the following for the maximum of the MLR function:

$$\begin{aligned}
MLA &= \min_{0 \leq k \leq n} \left[\sum_{i=0}^k (x_i - t \cdot \gamma(p, p+\eta)) \right] \\
\gamma(p, p+\eta) &= \frac{\log \frac{(1-p)}{(1-p-\eta)}}{\log \frac{(1-p)(p+\eta)}{p(1-p-\eta)}}
\end{aligned}$$

The only difference to the original equation is the additional multiplicand t (batch size).

11.2 Sequential Testing

The following model assumes independent loss events: each packet is lost independent from the previous packet loss with the same probability p . In the research literature different models can be found:

- [ZDPS01]: measurements in WAN show that the “arrival” of lost packet can be modelled by loss free epochs; the length of the epochs is exponentially distributed (Figure 11-7)
- [BSUB98]: correlation models of packet loss

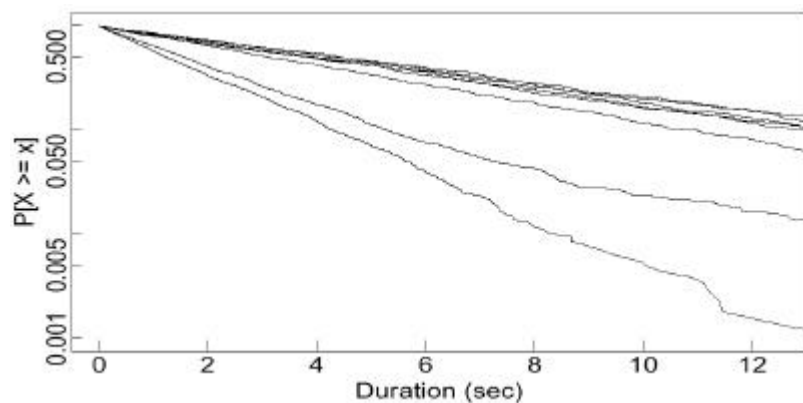


Figure 11-7: Example log-complementary distribution function of duration of loss-free epochs

11.2.1 Simulation Example of the Sequential Testing Approach

The simulation used the following parameters.

p0, p1	Packet loss probabilities
H0: p=p0 H1: p=p1	P(Loss)-Hypothesis
m_i	number of necessary loss packets for Hi: p=pi, i=0,1;
m	number of transmitted testpackets (= successfull + lost)
lossi	simulated number of lost packets
α, β	type1 errors: $\alpha = P(\text{decision}=H1 \text{true}=H0)$, $\beta = P(\text{decision}=H0 \text{true}=H1)$ (e.g. typical value $\beta=\alpha=0.05$)

Figure 11-8 illustrates the algorithm [Fisz78, pp. 690-699]. The lines m_1 and m_0 are calculated from the formulas

$$m_0(p_0, p_1, \alpha, m) := \frac{\ln(B(\alpha))}{\ln\left(\frac{p_1}{p_0}\right) - \ln\left(\frac{1-p_1}{1-p_0}\right)} - m \cdot \frac{\ln\left(\frac{1-p_1}{1-p_0}\right)}{\ln\left(\frac{p_1}{p_0}\right) - \ln\left(\frac{1-p_1}{1-p_0}\right)}$$

$$m_1(p_0, p_1, \alpha, m) := \frac{\ln(A(\alpha))}{\ln\left(\frac{p_1}{p_0}\right) - \ln\left(\frac{1-p_1}{1-p_0}\right)} - m \cdot \frac{\ln\left(\frac{1-p_1}{1-p_0}\right)}{\ln\left(\frac{p_1}{p_0}\right) - \ln\left(\frac{1-p_1}{1-p_0}\right)}$$

and represent the limits of uncertainty. The lines *loss0* and *loss1* show the simulated number of losses. If the number of losses is between the lines each of the hypothesis *H0* and *H1* has to be rejected, the system remains in an undefined state. If one of the lines *m_1*, *m_0* is crossed, the hypothesis *H1*, *H0* can be accepted.

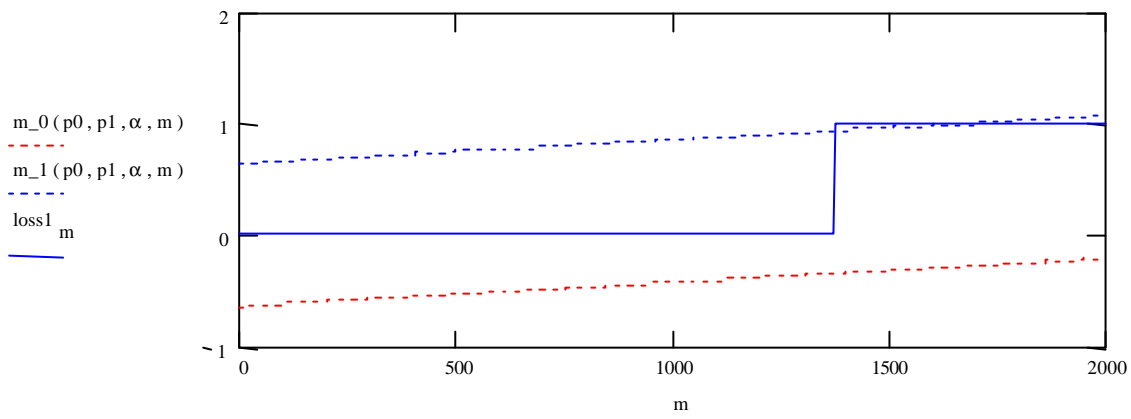


Figure 11-8: Sequential Testing

For operational purposes the mean number of sampled packets in the following formulas is of interest:

$$El(\alpha, p_0, p_1) := \frac{L1 \cdot \ln(B(\alpha)) + (1 - L1) \cdot \ln(A(\alpha))}{Elz(p_0, p_1)}$$

where

$$Elz(p_0, p_1) := \ln \left[\left(\frac{p_1}{p_0} \right)^{p_1} \cdot \left(\frac{1-p_1}{1-p_0} \right)^{1-p_1} \right]$$

and

$$L0 := 1 - \alpha \text{ and } L1 := \beta.$$

Figure 11-9 shows the mean number of sampled packets for *H1* hypothesis admission.

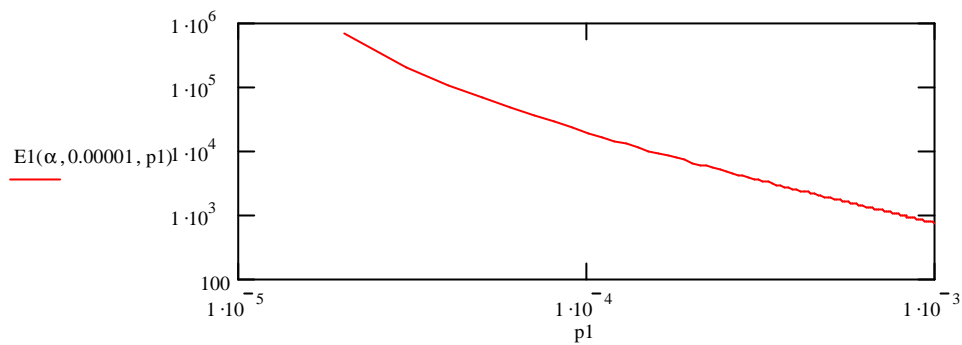


Figure 11-9: Mean sampling number for $H1$ hypothesis testing
 $H0: p = p0 = 0.00001; H1: p = p1 = 0.001$

Obviously the necessary number of samples increases for $p1 \rightarrow p0$, this is the reason why the differentiation between “near” values is more “difficult”.

This leads to a heuristic approach: The differentiation between “near” (e.g. 10^{-5} and 10^{-4}) values is not so important for QoS management than the differentiation between values like 10^{-5} and 10^{-3} .

So a α -type1-error-function can be assumed. Figure 11-10 shows a linear α -function for $p0 = 10^{-5}$

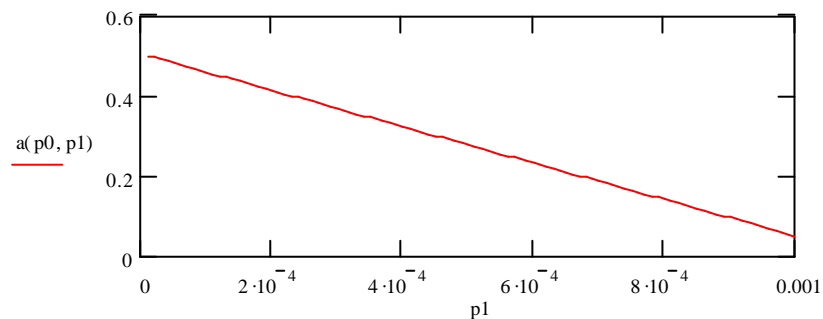


Figure 11-10: a -function: $a(10^{-5}, 10^{-5}) = 0.5; a(10^{-5}, 10^{-3}) = 0.05$

Figure 11-11 shows the reduced mean number of necessary samples.

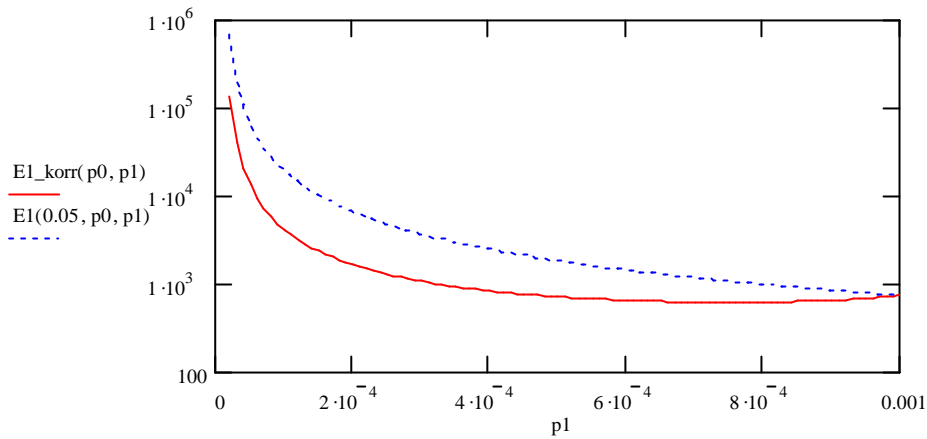


Figure 11-11: Reduced Number of Samples $E1_{korr}$ for $a(p0, p1)$, $p0=10^{-5}$

In the following simulation 200 experiments were made and the necessary number of samples until crossing the 10^{-3} threshold was determined. The solid line in Figure 11-12 shows the results of this simulation which is the number of samples c_n for the 200 experiments $n=1, \dots, 200$. The average number of necessary samples is 953.

A second simulation was realised with the loss-epoch model c_{n_exp} . The dashed line in Figure 11-12 shows the results when the series of lost packets shows exponential gaps. Using this model of losses the average number of necessary samples is 696.

For Poisson processes with independent arrivals the formulas are similar to the above formulas for the binomial loss process [KiWo56].

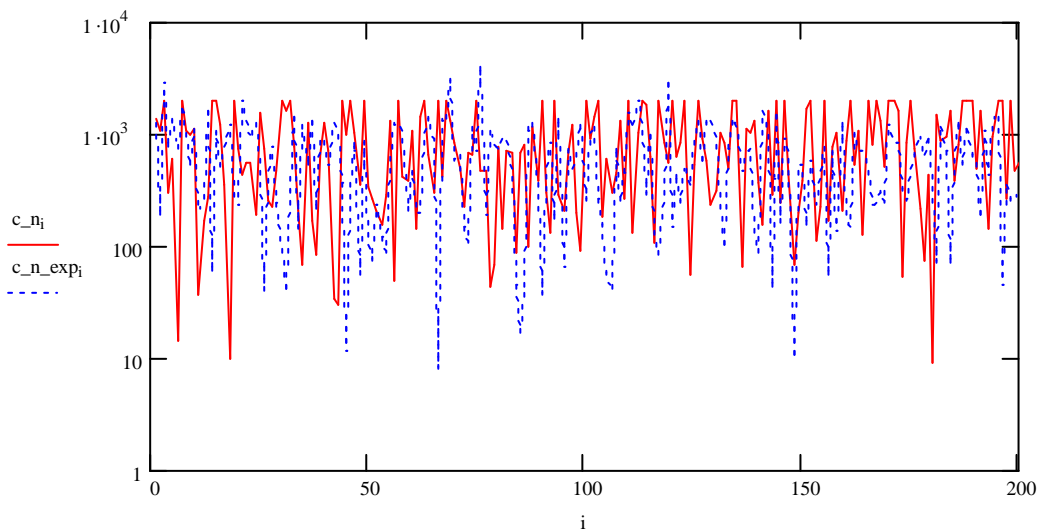


Figure 11-12: Simulation Results: Number of necessary Samples for Hypothesis Admission

11.2.2 Conclusions

Sequential tests have benefits for operational applications because of their fast convergence. E.g. [Fisz78] shows that for $\alpha=\beta=0.05$, $p_0=0.05$, $p_1=0.1$ the sequential test needs 144 samples and the Neyman-Pearson test needs 292 samples.

12 Annex C: Details of the DAG Capture Card

12.1 DAG Capture Card

The DAG capture card has been developed by the University of Waikato, New Zealand. This capture card is highly sophisticated and therefore it is the ideal candidate for executing passive measurements.

Generally there are three types of DAG capture cards:

- DAG 3.5 for ATM/PoS of OC-3c and OC-12c links
- DAG 4.2 for ATM/PoS of OC-48c links
- DAG 4.2e for Ethernet/Fast Ethernet links

12.2 Technical Description

The DAG capture card is a PCI-card for PCs with SC (ATM/PoS) or RJ-45 (Ethernet) connectors. It contains FPGAs for Framing and Time Stamping as well as a StrongARM processor for filtering records or collecting statistics. All records that are of interest are written to PC memory via the PCI bus and afterwards are stored on a PC's hard drive.

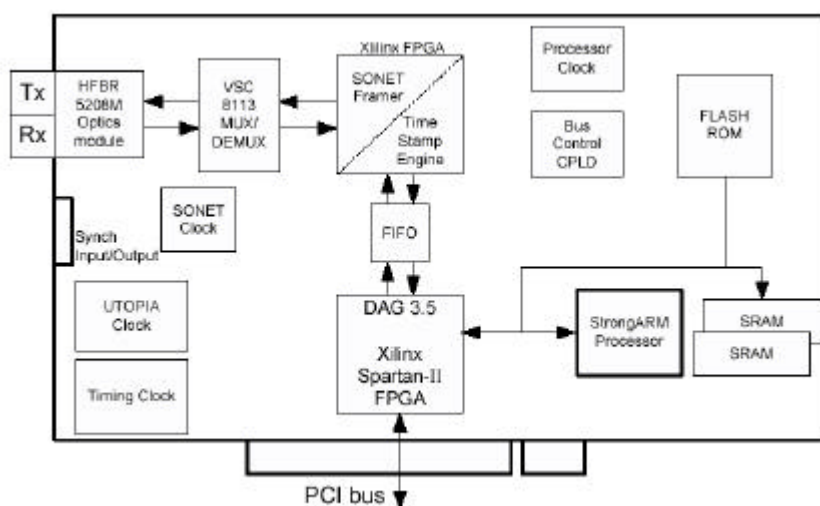


Figure 12-1: DAG card components

Further technical information about the DAG capture card can be found in [DAG].

12.3 Frame and Trace Format

By default the DAG capture card records 64 bytes of each incoming frame/cell. In a future release of the card it should be possible to choose an arbitrary number of bytes for recording.

Figure 12-2 shows the data formats of the DAG records when using different networking technologies for which a DAG capture card is available.

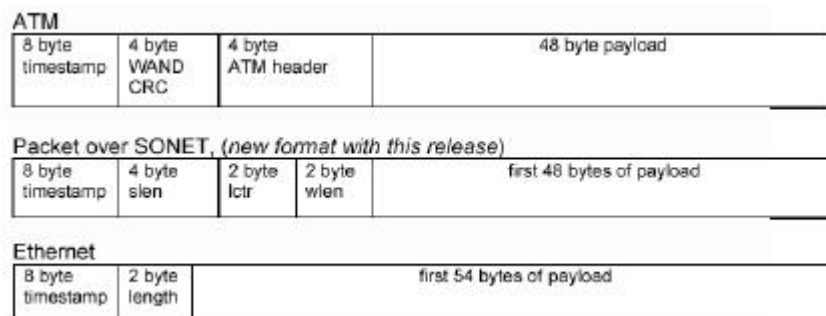


Figure 12-2: DAG Frame Format

The timestamp produced by the FPGA of the DAG capture card is 64 bits in length and is in little-endian (Pentium native) byte order. The most significant 32 bits of the timestamp represent the number of seconds since midnight, January 1, 1970. The least significant 32 bits form a binary fraction, representing the fractional part of the timestamp in the specified second. The entire 64 bit timestamp can be considered to be a fixed point number in seconds, or it can be considered an integer count of time in units of 2^{-32} seconds. Time deltas can be computed directly by subtraction of timestamps. This can even be done between different trace files if the clocks are synchronised, as timestamps are absolute, not relative to the start of the measurement trace.

The field WAND CRC which can only be found in the ATM mode of operation is a 32 bit CRC over the 48 bytes of cell data of the recorded ATM cell. The ATM header does not include the standard 8-bit HEC.

All timestamps and the 2 byte Ethernet length field are in little-endian byte order. WAND CRC, ATM header, slen, wlen and lctr fields are in big-endian (network) byte order. All payload data is captured as a byte stream, no byte re-ordering is applied.

Figure 12-3 shows some examples of common records assuming that no IP/UDP/TCP options are present.

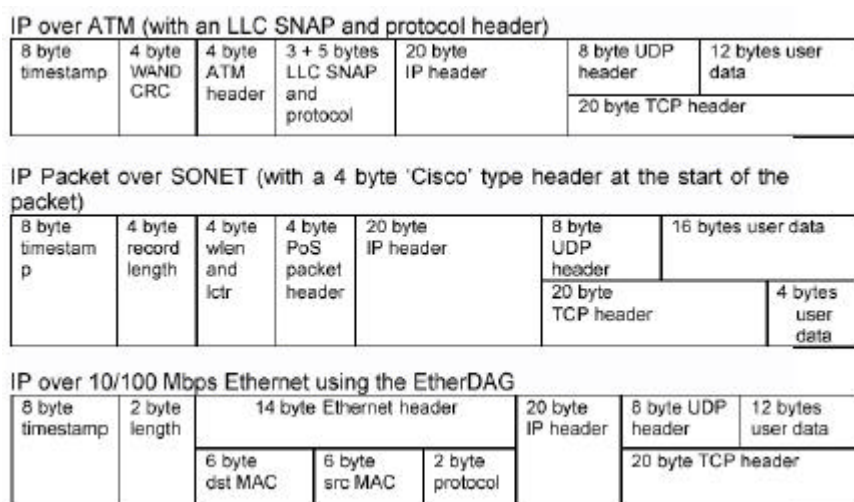


Figure 12-3: DAG Card Trace Format

12.4 System Requirements

The DAG 3 Installation Guide [DAGIG] gives the following recommendations concerning the PC which can host several DAG capture cards¹:

- PC, at least Pentium II 400 MHz, Intel BX or GX chip set
- Minimum of 128 MB RAM
- At least two free PCI slots with 3.3V and 5V power
- At least 2 GB free disk, more if large data sets are to be captured
- Linux operating system with 2.2.x or 2.4.x kernel
- Debian testing/unstable Linux system for those users requiring the handling of files larger than 2 GB in size

If more than one DAG capture card is used within on PC, each card should get its own hard drive for storing the records.

Considering the statements of papers dealing with storage and bandwidth requirements [Clear00, Mich01] and some personal communication with the DAG development team we would recommend the following components for the PC:

¹ When capturing traffic on an optical link, two DAG cards are required – one for each direction. Both cards may be hosted by the same PC.

CPU	AMD Athlon 1.4 GHz / 133 MHz Socket A
Motherboard	MSI K7T Turbo, 3 SDRAM, 2 UDMA-100
RAM	2 x SDRAM 512 MB 133 MHz, 168 Pin, 3.3 V
Hard Drives	2 x Die-HD Western Digital WD1000BB 100 GB, 7200 min-1, UDMA-100 ²

Table 12-1: DAG PC System Requirements

Besides the PC and the DAG capture cards, when capturing data transmitted over optical links, optical splitters are needed to split off the signals and feed them into the DAG cards.

A detailed description of how to install the DAG capture card can be found in [DAGIG].

If more than one measurement point is used, it could be necessary to have timely synchronised records. The DAG capture card provides a RS-422 differential serial port for the connection to an external timing source. This port can be connected to the output of another DAG card, or to an external reference, such as the PPS output of a GPS receiver.

² Optional the following hard drive could be used as well: Western Digital HD: 2 x IDE-HD Seagate Barracuda IV ST380021A 80GB, 7200 min-1, UDMA-100