Multi-Timescale Modelling of Ethernet Traffic

Patrik Carlsson

Karlskrona, December 2003
Department of Telecommunications and Signal Processing
Blekinge Institute of Technology
372 25 Ronneby, Sweden

Blekinge Institute of Technology
Licentiate Series No. 2003:09
ISSN 1650-2140
ISBN 91-7295-031-5

Published 2003
Printed by Kaserntryckeriet AB
Karlskrona 2003
Sweden

Dedicated to Bertil Carlsson

1932–2003

Abstract

Ethernet is one of the most common link layer technologies, used in local area networks, wireless networks and wide area networks. There is however a lack of traffic models for Ethernet that is usable in performance analysis.

In this thesis we describe an Ethernet traffic model. The model aims at matching multiple moments of the bit rate at several timescales. To match the model parameters to measured traffic, four methods have been developed and tested on real traffic traces. Once a model has been created, it can be used directly in a fluid flow performance analysis.

Our results show that, as the number of sources present on an Ethernet link grows, the model becomes better and less complex.

Keywords

Ethernet Traffic Model, Multi-Timescale, Multifractal, Bit Rate, Moments, Fluid Flow Analysis, Model Matching, Measurement.

Acknowledgments

First of all I would like to thank Professor Arne A. Nilsson, for accepting me as a PhD student. I would also like to thank Dr.-Ing. Markus Fiedler for planting the seed that inspired me to start this journey. Docent A. Popescu for the discussions that we had, and my colleagues at the University during these years. Without you the journey would not have been the same.

I would also like to thank my family, without your support I would not have made it.

Patrik Carlsson
Karlskrona, November 2003

Contents

Abstract vii

Acknowledgments ix

1 Introduction 1
- 1.1 IEEE 802.3 – Ethernet . . . 2
  - 1.1.1 History . . . 2
  - 1.1.2 Recent Features . . . 5
  - 1.1.3 Computers and Ethernet . . . 6
- 1.2 Traffic Measurement . . . 7
  - 1.2.1 Active Measurements . . . 7
  - 1.2.2 Passive Measurements . . . 9
  - 1.2.3 Timestamp Accuracy . . . 11
  - 1.2.4 Ethernet Measurements . . . 12
- 1.3 Traffic Modelling . . . 14
- 1.4 Optimization . . . 16
  - 1.4.1 Genetic Algorithms . . . 16
  - 1.4.2 Simulated Annealing . . . 19
  - 1.4.3 Comparison of GA and SA . . . 20
- 1.5 The Next Step – Fluid Flow Analysis . . . 20

2 State of the art 23
- 2.1 Bellcore Traces . . . 23
- 2.2 Network Measurements . . . 25
- 2.3 Traffic Modelling . . . 27

3 Measurements and Moment Estimation **29**
3.1 Measurements . . . 29
Measurement Point . . . 31
3.2 Bitrate Estimation . . . 32
3.2.1 Fractional Bits . . . 32
3.2.2 Timestamp Accuracy . . . 34
3.3 Moment Estimation . . . 36
3.4 Truncation . . . 37
3.5 Normalization . . . 38

4 Process Description **39**
4.1 Process . . . 39
4.2 Sub-process Analysis . . . 41
4.2.1 Moment Analysis . . . 41
4.2.2 Numerical Problems . . . 43
4.2.3 Parameters . . . 44
4.3 The Process . . . 47
4.3.1 Construction . . . 49
4.3.2 Moments . . . 50
4.3.3 Gradient . . . 50

5 Process Matching Methods **53**
5.1 Manual Matching . . . 53
5.2 Genetic Algorithm . . . 55
5.2.1 Fitness Function . . . 55
5.2.2 Boundary Values . . . 58
5.2.3 Example . . . 58
5.3 Simulated Annealing . . . 62
5.3.1 Cost Function . . . 62
5.3.2 Boundary Values . . . 62
5.3.3 Example . . . 62
5.4 Heuristic Method 1 . . . 66
Example . . . 67
5.5 Heuristic Method 2 . . . 70
Example . . . 71
5.6 Matching in Practice . . . 72
5.6.1 Number of Timescales . . . 72
5.6.2 Number of Sub-Processes . . . 72
5.6.3 Fitness . . . 73

5.6.4 Matching Problems 74

6 Results and Discussions **75**
6.1 Bellcore Traces 75
6.1.1 Measurements 75
6.1.2 Matching 78
6.2 Internet Access Link 82
6.2.1 Measurements 82
6.2.2 Matching 86
6.3 ADSL Link 90
6.3.1 Measurements 90
6.3.2 Matching 93
6.4 Discussion of Measurement Equipment 97
6.5 Observation on Measured traces 97
6.6 Discussion on Matching Methods 99

7 Conclusions and Outlook **105**

A Kronecker Algebra and Matrix Construction **107**

B Derivation of Moments and their limits **109**
B.1 First Moment Analysis 110
Limits 111
B.2 Second Moment Analysis 112
Limits 116
B.3 Third Moment Analysis 117
Limits 122

Bibliography **123**

List of Figures

1.1 LAN components 5
1.2 Example of an active measurement 7
1.3 Three ways to attach a packet monitor 9
1.4 10/100Base-T wiretap 11
1.5 Comparision of timestamp accuracy 13
1.6 Constraints on a model 15
1.7 GA Terminology 17
1.8 Fluid flow funnel 21

2.1 CDF frame sizes 25
2.2 CDF interarrival times 26

3.1 File structure 30
3.2 Capture header structure 30
3.3 Measuement point 31
3.4 Bit rate estimation 33
3.5 Fractional bit 33
3.6 Timestamp inaccuracy 34
3.7 Example of measurement error 35
3.8 Estimation of bitrates at different timescales. 36

4.1 State diagram for a 2-state MMRP 40
4.2 Example of output from a sub-process. 40
4.3 Sub-process with sub-process 45
4.4 Sub-process without numerical problems 45
4.5 Sub-process with fixed output rates and varying transition rates. . 46
4.6 Sub-process with fixed transition rates and varying output rates. . 47

4.7 Sub-process with asymmetrical transition rates. 48
4.8 Process based on three sub-processes 49
4.9 Example of a three slope process. 51

5.1 Artificial process 54
5.2 Fitness function 57
5.3 GA matching example, artifical data, default bounds 59
5.4 GA matching example, artifical data, tight bounds 61
5.5 SA matching example, artifical data, default bounds 63
5.6 SA matching example, artifical data, tight bounds 65
5.7 HM1 matching example 69
5.8 HM2 matching example. 71

6.1 Bellcore pOCt89 first three moments. 76
6.2 Bellcore OCt89Ext4 first three moments 77
6.3 BCpOct89 model 79
6.4 BCOct89Ext4 model 81
6.5 Measurement setup for the IAL scenario 82
6.6 IAL-1-IN first three moments 84
6.7 IAL-1-NI first three moments 85
6.8 IAL-1-IN model 87
6.9 IAL-1-NI model 88
6.10 Measurement setup for the ADSL scenario. 90
6.11 ADSL8RM first three moments 91
6.12 ADSL8MR first three moments 92
6.13 ADSL8RM model 94
6.14 ADSL8MR model 96
6.15 Comparision between timestamp accuracies 98
6.16 Better BCpOct89 model 102
6.17 Better IAL-1-NI model 103

B.1 State diagram for a 2-state Markov Modulated Rate Process. . . . 109

List of Tables

1.1 Important Ethernet standards 4
1.2 Data amount when capturing Ethernet 11

2.1 Network Application usage 1990 to 2003 24
2.2 Statistics for the BCOct89Ext4, IAL-1-IN and IAL-1-NI traces . . 27

3.1 Error in bit rate estimation 35

4.1 Gradient for the second and third moment 44

5.1 GA, artifical data, default bounds 58
5.2 GA, artifical data, tight bounds 60
5.3 SA, artifical data, default bounds 64
5.4 SA, artifical data, tight bounds 64
5.5 HM1 suggested solutions. 68

6.1 BCpOct89 models 78
6.2 BCOct89Ext4 models 80
6.3 IAL-1 traffic statistics 83
6.4 IAL-1-IN models 86
6.5 IAL-1-NI models 89
6.6 ASDL traffic statistics 91
6.7 ADSL8RM models 93
6.8 ADSL8MR models 95
6.9 Trace edge values 99
6.10 Matching methods summary of fitness values 100

Chapter 1

Introduction

In this work we will investigate the possibilities to model an Ethernet link on multiple timescale using a model that is simple, captures the scaling behaviour of the traffic and is usable in fluid flow analysis.

During the last decade an increasing amount of research showed that data traffic exhibits self-similar properties, and recently also indications that it is multifractal. During the same period many offices and residential areas became networked using Ethernet technology. Internet became a part of many businesses and private users' daily routine. All in all, this is a highly dynamic environment, in which it is hard to model a link using a single general model. Instead each link is unique, and requires detailed analysis in order to find a suitable model and capture its parameters.

A model proposed by Mannersalo, Norros and Riedi [MNR02] form the base for the model that we use. Their model exhibit multifractal properties:

> In its simplest form our model is based on the multiplication of independent rescaled stochastic processes $\Lambda^{(i)}(\cdot) \triangleq \Lambda(b^i)$ which are piecewise constant. ... In multiplying rather than adding re-scaled versions of a 'mother' process we obtain a process with novel properties which are best understood not in an additive analysis, but in a multiplicative one. Processes emerging from multiplicative construction ... exhibit typically a 'spiky' appearance [MNR02].

The mother- or base process is a Markov-modulated rate process (MMRP) with two activity states. The MMRP is well known, and used, in fluid flow analysis. Thus, with the exception of some decomposition results [SE91], fluid flow analysis

can be used straightforward. The model used here is a slighly modified version of the Mannersalo, Norros and Riedi model, modified in such a manner that we have no requirement on the scaling between the independent processes, nor do we require a infinite number of processes.

The model uses the link layer bit rate as the network parameter to match. The bit rate is a parameter that is easy to estimate (for both fixed and varying frame sizes) and it is available for all technologies and layers. Since the model operates on an Ethernet link, the bit rate is the obvious candidate. An alternative parameter could have been frame rate (frames per second, fps). But since a link does not differentiate bits from bits, i.e. bits that are part of the frame header receives the same treatment as the payload bits, the bit rate is more suitable. If we were to model an Ethernet switch, then both the bit rate and frame rate would be needed. Because a switch capabilities are limited in terms of bit rate, frame rates and switch fabric. To capture the scaling behaviour, the model matches the statistical moments of the bit rate at several timescales. An alternative to the statistical moments could be the probability density function (pdf). But this would make the analysis much more complex. Furthermore, we would be required to use a discrete version of the pdf which in itself can cause problems [LK91].

To match the models parameters to measured traffic, two well known optimization methods are used, namely Genetic Algorithms and Simulated Annealing. In addition, two heuristic methods have been developed.

In this chapter short introductions are given on Ethernet, traffic measurements, traffic modelling, optimization methods (Genetic Algorithms and Simulated Annealing) and the stochastic fluid flow analysis. In Chapter 2 a review of related works is given. In Chapter 3 we describe how we measure the network in order to obtain the parameters of interest, followed by a description of the model that we use in Chapter 4. In Chapter 5 we describe the matching methods, followed by a set of matching examples based on real-network traffic in Chapter 6. Conclusions and outlook are discussed in Chapter 7.

1.1 IEEE 802.3 – Ethernet

1.1.1 History

The history of IEEE 802.3 starts in 1983, with the definition of the first IEEE 802.3 Standard, 10BASE5. This was the culmination of approximately ten years of research, implementation and standardization discussions. It started in the

early 1970's with an experimental setup at the Xerox Palo Alto laboratory by Dr. Robert Metcalfe [MB76]. This experimental network ran at 3 Mbps. In 1979 the DIX Consortium (DEC-Intel-Xerox) was formed with the purpose to standardize, commercialize and promote the use of Ethernet technology. In 1980 a 10 Mbps standard was published[1]. In the same year the IEEE formed the 802 project to standardize LAN (Local Area Network) technology, with the goal of producing a single standard for all LANs. This failed and the result was that three different workgroups (WGs) were formed, 802.3 for Ethernet, 802.4 for Token Bus and 802.5 for Token Ring. IEEE 802.3 was published in 1983, supporting 10 Mbps and a wide range of cabling options.

During the 1980's the 10 Mbps Ethernet (from now on Ethernet refers to IEEE 802.3, unless otherwise stated) matured. It moved from thick and thin coaxial cables to twisted pair (TP) with the 10BASE-T standard in 1990. This included a paradigm shift from bus to star structure. In the same year IEEE 802.1D Bridge standard appeared. In 1995 100BASE-T Fast Ethernet was standardized, with proprietary products shipping already in 1992. In the same year, full-duplex products were available. In 1998 Gigabit Ethernet 1000BASE-X was standardized, with commercial shipping already in 1997. In 2002, a 10 Gbps Ethernet standard 10GBASE-X was released, with commercial products available already in 2001. Currently (2003) the industry is moving toward 40 Gbps as the next standard. A bit rate of 100 Gbps would require a new physical interface, whereas 40 Gbps can reuse the existing interface. A summary is shown in Table 1.1.

Initially Ethernet was used to connect computers to a LAN. As mentioned previously, the first Ethernets used thick or thin coaxial cables. This caused all computers on a LAN to share the same medium, thus the average sending speed of a computer would be a function of the link speed, i.e. speed at which the network operates, the number of attached hosts and their degree of activity. The use of coaxial cables also had cabling problems; the cable usually went from room to room in a shortest distance manner. With the 10BASE-T standard and bridges[2] the cables could follow the same cabling drums as the power and telephone cables. This transformed Ethernet from a bus to a star structure. Bridges also had the effect of increasing the computer's link speed since the medium is now shared only by two entities, the host and the recipient, given that no collisions occur within the bridge. Today Ethernet is *the* LAN technology; it has also started to gain usage in residential areas. With the sinking price of

[1] See http://standards.ieee.org for information about the IEEE standards.
[2] "A Switch is a Bridge is a Switch" [Sei00]

Table 1.1: Summary of important Ethernet standards (and other standards that affect Ethernet) and year of standardization.

Year of Standardization	**Standard**	**Speed [Mbps]**	**Comment**
1983	802.3	10	10BASE5 Yellow cable, etc..
1990	802.3i	10	10BASE-T Twisted Pair
1990	802.1D		Transparent Bridge
1995	802.3u	100	100BASE-T FastEthernet, FE
1997	802.3x		Full-Duplex Flow Control
1998	802.3z	1000	1000BASE-X GigabitEthernet, GE
1998	802.1Q		Virtual LAN Bridge/Switch
1999	802.3ae	1000	1000Base-T Gigabit over copper
2002	802.3ae	10 000	10GBASE-X, 10GE

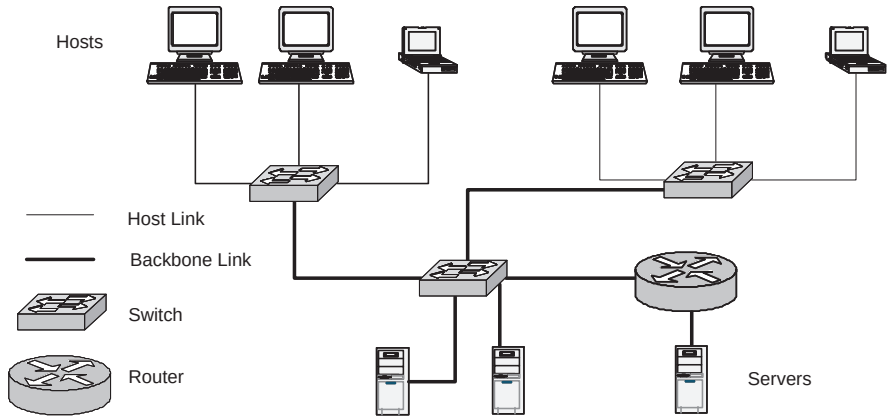

Figure 1.1: A small Ethernet LAN, specifying the different network parts.

computers and home electronics, the interconnection of such devices will mostly be done using Ethernet [EFM]. Furthermore, just the size of the install-base will ensure that Ethernet will be around for quite a while.

With the standardization of 100BASE-T in 1995 it was easy to provide servers better network access and to create a LAN backbone. A LAN backbone is a part of a network that in general operates at higher link speed than the links attached to the hosts/computers. Normally no hosts are directly attached to the backbone, only servers and network equipment that need the higher capacity. A typical example of a small Ethernet LAN is shown in Figure 1.1, here the different parts (backbone, hosts, switches and links) of the LAN are emphasized.

Gigabit Ethernet was introduced to handle the increasing traffic on the LAN backbone and the overall need for higher network capacity. With 10 Gigabit Ethernet, Ethernet moves into the MAN/WAN segment [GEA].

1.1.2 Recent Features

The changes in link speed, transmission mode and wiring has changed Ethernet from being a high-speed (10 Mbps), half-duplex, shared medium to an *very* high-speed (10 Gbps), full-duplex, dedicated medium. Obviously the technology has changed significantly since 1983. Recently a couple of interesting functions have been added to Ethernet.

- Virtual LAN
  This feature enables a unit to participate in more than one LAN, using only one physical interface. Initially this will probably be used as a software patch panel, but the possibilities are far greater. For example, from a host perspective this enables hosts at different locations to be on the same LAN segment. Another example, is the creation of temporary LANs.

- Flow Control
  Flow control allows a device to control how other devices send data to it. This can prevent data from being lost due to buffer unavailibility.

- Link Aggregation
  Instead of having to increase link capacity in factors of tens this feature allows for combining several interfaces to obtain higher throughput and redundancy. An aggregated link provides $n \times X$ Mbps, but the highest link speed is still X Mbps. This is like a highway, if the maximum speed is 90 km/h, then the road can handle Y cars/s. Adding a new lane will allow the road to carry $2 \times Y$ cars/s at 90 km/h, not $2 \times Y$ cars/s at 180 km/h.

- Priority
  This enables frames to receive different treatment due to their priority. The capability to give different treatement is a basic requirement for implementation of Quality of Service, QoS.

1.1.3 Computers and Ethernet

The standard computer of today has a PCI bus. This is a bus with a width of 32 bits and 33 Mhz frequency denoted PCI-32/33. The PCI-32/33 bus has a theoretical throughput of about 32 bit $\times$ 33 MHz = 1056Mbps. This capacity is shared among all devices that attach to the bus, hence it cannot fully utilize a 1 Gigabit Ethernet. A PCI-64/66 bus has a theoretical throughput of 4224 Mbps, which has enough capacity to enable a computer to keep up with a full-duplex Gigabit Ethernet. If higher capacity is needed, the new PCI-X [PSI] standard enables throughputs up to 33 Gbps (PCI-X 64/533 Mhz). However, with network speeds of 10 Gbps, it may take some time before a standard PC can fully utilize a 10 Gbps Ethernet link. Alternatives to PCI-X are: InfiniBand [IBD], RapidIO [RIO] and HyperTransport [HTC].

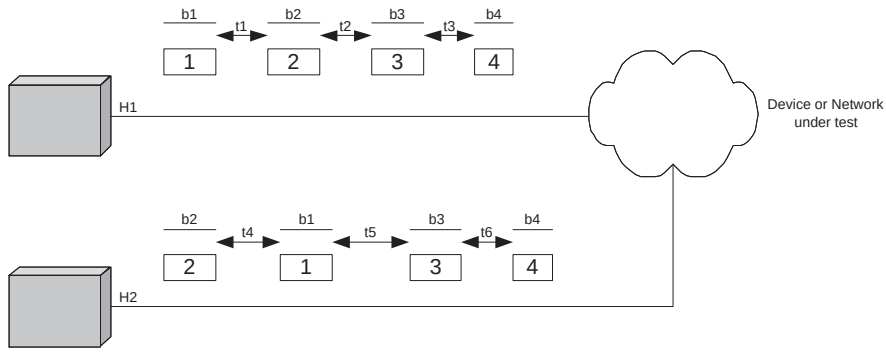

Figure 1.2: An example of an active measurement, H1 sends a stream of packets (1,2,3 and 4) to H2. H1 records the packet-order, sizes and transmission times, H2 does the same. The information can later be compared.

1.2 Traffic Measurement

When analyzing network performance, measurements of some sort are needed, ranging from the highly abstract user questionnaire: "Are you satisfied with the network?" to the detailed timing of symbol arrivals and bit error rate. Before performing any measurements we need to decide which parameter or parameters are of interest. As mentioned above, the parameter of choice is the bit rate. The bit rate has several appealing properties; easy to estimate and it is availibel for all technologies and layers. For instance, most downloads are visualized by a progress bar, in addition to displaying how much data that remains it usually also display a instantaneous download rate, either in byte/s or bit/s. This rate is usually the rate that the application perceives, i.e. bit rate at application level. The bit rate can also be used to estimate several other parameters that affect network performance [FTCN03]. Bit rates can be obtained by both active and passive measurements.

1.2.1 Active Measurements

Active measurements are based on the injection of traffic into a network, with the intent of measuring one or several parameters based on the treatment that this traffic receives from the network. The actual measurement can be done

directly from a traffic generator or by using passive methods like port mirroring, wiretapping, etc. Active measurements can be compared to impulse tests. This type of test is done by mounting accelerometers on the device to be tested; the device is then subjected to an impulse, usually from an impulse hammer. This hammer registers the amount of force that the device is subjected to, and the accelerometer registers the response at the measurement points. A modal analysis can then be performed to find the device's resonance frequency. When performing this test the operator has to make sure that the impulse is not too powerful. If so, it may be noticed if the device visually deforms. In a network environment this is not the case, furthermore a network is not a stable environment. In this case, active measurements that increase the load on the network might generate corrupt estimations of the measured parameter; furthermore they might also disrupt the network by causing overloads. Examples of some commonly used active measurement tools are `PING` and `TRACEROUTE`, these operate at the network layer. We are interested in link layer bit rates, and these are quite troublesome to observe by active measurements. Now, active measurements should not be disregarded. In fact, in some cases active measurements are the only option.

An example of an active measurement is shown in Figure 1.2. Here host H1 generates a stream of packets (1, 2, 3 and 4) and records the packet-order, sizes and transmission times. H2 records the same information. By comparing the information at H1 and H2 we can obtain information about: bit rate, delay (given that H1 and H2's clocks are synchronized), packet reordering, maximum transfer unit (MTU), etc. Alternatively H2 could return the packets to H1 immediately upon reception. H1 then records the same parameters as before when the packets return. Given this information H1 can estimate the round-trip-time (RTT). Other parameters like MTU, bit rate and delay could be estimated if H2 included some kind of information in the returning packets. What we need to consider is that the parameters estimated are valid for the entire network in-between H1 and H2. It is therefore not possible to distinguish the impact of a particular link in the network. Given the network technology, it might also be possible for one packet to follow one set of links (L1, L2 ... LN) whereas another packet follows a different set of links (L1, M2, M3 ... Mk). On the other hand, if there is only one link in-between H1 and H2, then the results would only be partially valid. This because the measurement stream we add will interfere with the already existing traffic. If no other traffic is present, we only measured the link capacity.

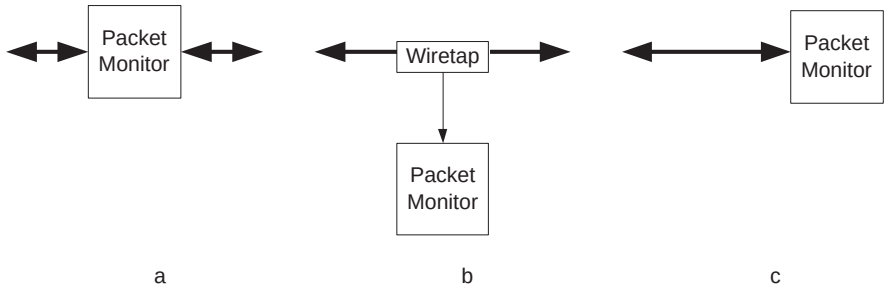

Figure 1.3: Three ways to attach a packet monitor to a link; a) intermediary b) wiretap and c) direct.

1.2.2 Passive Measurements

Passive measurements are based on the observation of the already existing traffic in the network. Hence, no additional traffic will be added to the network. The main advantage is that the parameter(s) under study will be untainted by the measurement. A passive measurement usually involves some form of passive measurement device, a device that copies the network traffic without interfering with it. As a parallel to the example provided in the active measurement case, a passive measurement could be compared to a voltage meter. Passive measurements are better suited for large scale deployment, since they do not add load to the network. On the other hand, depending on the desired level of detail, passive measurements can generate a huge amount of data, which can be troublesome to collect.

Furthermore, an important issue is the privacy. As the network traffic is copied, it is possible to read all the data within a frame. This requires that the captured data is treated with care. The data should by default be deprivatized[3] as much as possible, but not to the extent that the data is useless. This is a problem that active measurements largely avoid.

A commonly used tool for collecting passive measurements is SNMP (Simple Network Management Protocol) [CFSD90, HAN99, Sub00]. SNMP uses the manager/agent concept. The manager issues control actions to the agent. The agent processes these commands and returns the results. The information is exchanged via special SNMP Protocol Data Units (SNMP PDUs). The SNMP

[3]Removal of any contents that might identify a particular user

architecture uses an information model denoted as the Management Information Base II (MIB-II) [MR91]. This information base specifies syntax and semantics of the stored data. The variables in the information base are denoted as objects. The data types of the MIB-II objects comprise counters, addresses, and strings.

An alternative to SNMP based measurements is the use of packet monitoring systems like TCPDUMP [TC] and DAG cards [ENDa]. The packet monitoring software records packets for off-line, or on-line, analysis. These are usually referred to as traces. This allows for analysis on very small timescales, where the lower limit is set by the monitoring system. A monitoring system can be attached to a link in three ways (Figure 1.3); as an intermediary node (a), through a wiretap (b) or as an end node (c). Using the first alternative (a) requires the packet monitor to maintain wire speed even during measurements, which is quite difficult unless dedicated monitoring equipment is used. The third option (c) is not really interesting unless the other end of the link attaches to a network device that mirrors one of its other interfaces. This is especially interesting when it is impractical, or impossible, to install wiretaps on the links that connect to the network device. However, before port-mirroring is done, the properties of the mirroring device have to be evaluated since the mirroring operation can become a bottleneck and hence loose packets. The use of a wiretap (b) is probably best alternative of the three. The main problem with it is that it requires a small disruption of network services when the tap is installed. After this is done, the network will operate as normally, unaware of the wiretap. The wiretap will introduce some small transmission delay. In the order of one or two bit times, i.e. the time it takes to transmit a bit at the given link speed.

A wiretap works by copying the information on the tapped wire into a separate link that is attached to a packet monitor. A packet monitor can be a dedicated platform, a standard PC, or a combination of both, and this depends on the wire being tapped. For each physical medium, a special wiretap is needed. The simplest wiretap is the one that taps wireless communications, since it does not need to cut the wire. At the other end, there are high-quality wiretaps for optical cables, where the optical interface is extremely sensitive. In Figure 1.4 a schematic for a 10/100BASE-T wiretap is shown. To handle full-duplex operations the wiretap needs two output interfaces, one for each direction. Each interface connects to the interface of a packet monitor; usually this is the network interface of a computer.

Passive measurement can generate a large amount of data. A moderately loaded (50%) 100 Mbps full-duplex link generates approximately 12 MB[4] of data

[4] MB=1024^2,GB=1024^3, TB=1024^4 bytes.

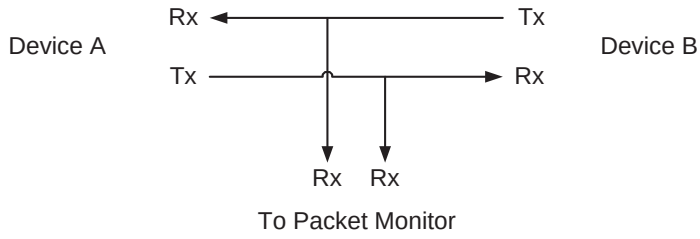

Figure 1.4: Sketch of a 10/100BASE-T wiretap, Tx=Transmit, Rx=Receive.

Table 1.2: Maximal amount of data when capturing full-duplex Ethernet links at various loads (frame size limitations and inter-frame are unaccounted).

Link Capacity	**100% load**		**50% load**		**10% load**	
Mbps	**GB/s**	**TB/day**	**GB/s**	**TB/day**	**GB/s**	**TB/day**
10	0.0023	0.1965	0.0012	0.0982	0.0002	0.0196
100	0.0230	1.9645	0.0116	0.9823	0.0023	0.1965
1 000	0.2328	19.6451	0.1164	9.8225	0.0233	1.9645
10 000	2.3283	196.45	1.1642	98.2254	0.2328	19.6451

every second, that is 0.98 TB over a day. In Table 1.2 a summary of the data sizes involved in capturing Ethernet is shown. The amount of data is estimated with

$$O = \frac{\rho_1 C}{8} + \frac{\rho_2 C}{8}$$

where ρ_x is the mean utilization in direction x and C is the link capacity.

With this said, a passive measurement cannot always be used. For instance; a passive measurement will have problems in verifying if a service (network or application) is operational, a simple task for an active measurement. Active measurements should however be used with caution, especially in a large network.

1.2.3 Timestamp Accuracy

The most commonly used packet monitoring software is TCPDUMP [TC], which is available for most operating systems. TCPDUMP uses the Packet Capture

Library (PCAP), which operates on a RAW socket. This socket receives copies of all the network traffic received by a network interface card (NIC). Each frame is timestamped, either by the NIC, the kernel or TCPDUMP. Using a vanilla LINUX 2.4 kernel and TCPDUMP 3.7, the timestamp give an accuracy in the order of μs. Furthermore, the timestamps are highly dependent on the load of the kernel, hence their accuracy might vary. For Windows the WinDump [WD] package from Politecnico di Torino is available online, which is similar to the TCPDUMP package. In the late 1990s, Waikato University developed the DAG card [ENDa, DAG], a high accuracy network monitoring card. It has a timestamp accuracy in the order of < 100 ns. Currently these cards are available for a wide range of network technologies (10Base-T, 100Base-T/X, 1000Base-X and OC-192). In Figure 1.5 a comparison between TCPDUMP, WINDUMP and DAG is shown. All three monitors were evaluated in parallel, and each was attached to a wiretap. The test was done by sending a stream of UDP packets, over the link that the wiretaps were attached to. Each monitor produced a packet trace containing the arrival time. From these traces, the interarrival time was calculated. The differences between TCPDUMP and DAG are not that large, but obvious. Worth mentioning is that the monitors only needed to monitor one direction, and that both TCPDUMP and WINDUMP reported that they dropped packets. See [Car] for more details.

1.2.4 Ethernet Measurements

When measuring a link layer property as the bit rate on an Ethernet, there are some issues that need to be addressed. First of all, since we are only interested in link layer properties we do not need to store all data that passes on the link. Actually we only need to store two items; frame arrival time and frame size. The first is however a bit tricky, since most generic Ethernet cards do not report the frame arrival time, but the time when a complete frame was received without collisions. The arrival time of the frame can be calculated from this value. However, most NICs do not report this time, they simply pass the frame to the kernel. It is then up to the kernel to set the timestamp for the frame. This can cause frames that arrived in sequence to be timestamped with the same value, or at least with incorrect interarrival times (see previous paragraph).

The position of the wiretap is crucial. If the link is a shared medium the wiretap behaviour has to be evaluated. For instance, if a frame is received correctly at the wiretap but destroyed at the recipient, what does the wiretap report? Does it report a successful frame or a collision. For a point-to-point link this problem is also present, but the risk of collisions are minimized. This is a problem that is

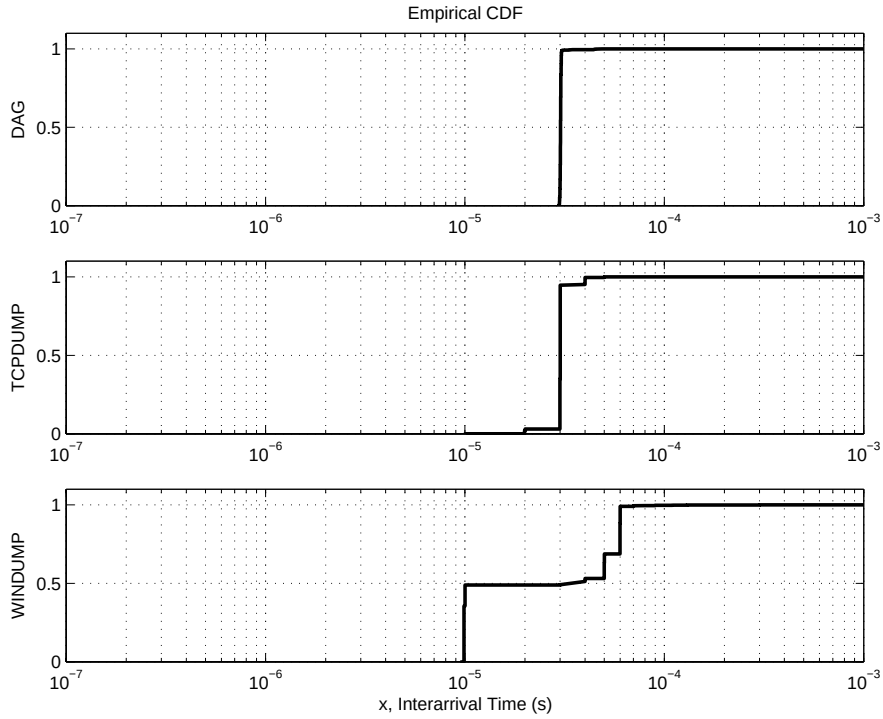

Figure 1.5: A comparison between DAG, TCPDUMP and WINDUMP timestamp accuracy; x is the interarrival time in seconds.

not easily solved. A possible solution would require some form of feedback from the receiving devices higher layers, but this is not yet practical.

1.3 Traffic Modelling

A network's performance is evaluated through measurements. To estimate the performance of a non-existing network (for example during the design phase), two possible methods are simulations and mathematical analysis. Both of them make use of traffic models; and hence traffic modelling is a vital component in network design and analysis.

When constructing a model, there are two types of constraints. The first type is regarding the entity (device,process,etc.) that is being modelled. Questions like what properties should be modelled and when is a model considered good or bad. The other type of constraints, focuses on the usability of the model. Let us consider that we wish to model a circle, but the system that will use the model only supports squares of varying sizes. Furthermore, there might be a constraint on the number of shapes that the system can handle. For example see Figure 1.6. In (a) the model is constructed from only one square. This is a very simple model, but there is hardly any similarity to the desired shape. In (b) there are 156 squares all of the same shape, here the model is a bit better. (c) uses 49 squares, of varying sizes. If more squares would have been used the match would have been even better. And finally, (d) there are approximately 700 cubes all of the same size. From the view of the system that will use the model, the accuracy grows from (a) to (d), as well as complexity. Thus, it might be possible to construct a perfect model, but the model will be useless if no system can use it.

During the 20th century the Poisson distribution was generally used to model telephone traffic. This model was verified in the 60's by the Holbaeck measurements [Ive73]. Initially the same models that were used to model classical telephone traffic were used to model data traffic. In 1993 it was shown in the paper *On the self-similar nature of Ethernet traffic* by W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson [LTWW93] that the old models did not hold for data traffic. This result was supported by several other studies [PF95, BTW95]. Later, it was also suggested that network traffic could in fact be considered as multifractal [FGW98, MN97].

We will model the link layer bit rate. Initial link layer models only considered modelling one parameter (e.g. the mean bit rate) at one timescale. Similarly to self-similar and multifractal models, we are looking on a range of timescales

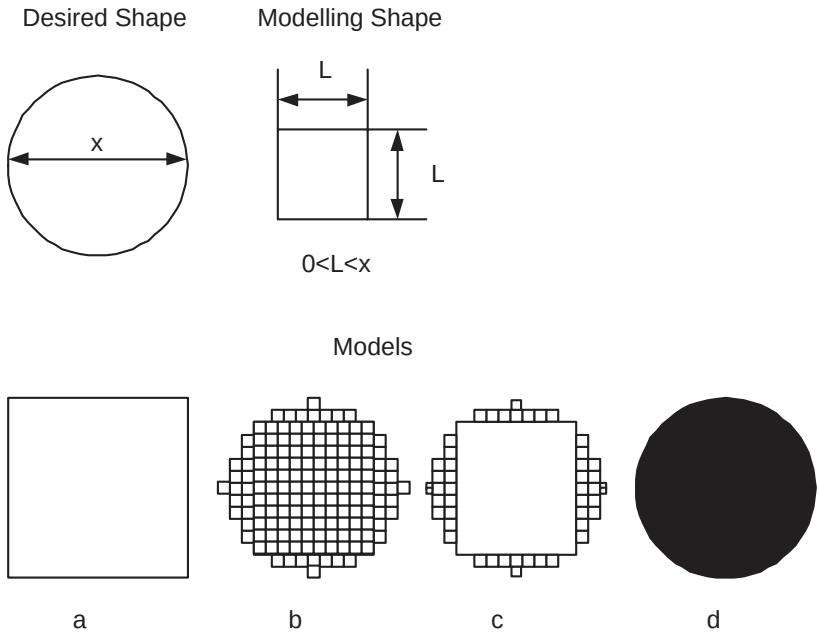

Figure 1.6: Example of type two constraints on a model. The desired shape is a circle, but the modeling only supports squares.

in order to capture the scaling properties of Ethernet traffic. To improve the model, we also model higher moments and not only the variance on which some self-similar and multifractal models focus. Our model is based on a mathematical process, which is described in detail in Chapter 4. The first modelling constraint we address by modelling the bit rate moments at several timescales, the second constraint sets an upper limit on the number of sub-processes that the model is allowed to use.

1.4 Optimization

In this section short reviews are given with regard to Genetic Algorithms and Simulated Annealing. The goal for any optimization algorithm is to find the best possible solution, with regard to some criteria. The two generic methods presented here have their strength in the ability to work on a wide range of problems. Compared to heuristic methods that require detailed problem knowledge.

1.4.1 Genetic Algorithms

In the 1950s and 1960s, evolutionary systems were evaluated in order to see if they could be used for optimization. Rechenberg [Rec65, Rec73] introduced *evolution strategies* in the 60s, the initial application was to optimize real-valued parameters for airfoils. Fogel, Owens and Walsh [FOW66] developed *evolutionary programming*, where possible solutions are represented in finite-state machines. These are evolved by randomly mutating their state-transition diagrams and choosing the fittest.

Genetic Algorithms (GA) were introduced by John Holland in 1975 in his book *Adaptation in Natural and Artificial Systems*[Hol75]. GA operates on populations; from this population, an individual is selected for reproduction as a function of its fitness. Fitness specifies how good the individual is at solving the particular problem. Once the potential parent(s) have been selected, children are formed. These can be formed by combining the genetic material of one or more parents. With a given probability a child will mutate. This is very similar to the normal evolution seen in biological systems.

Terminology

Since GA is influenced by biology, this is reflected in the terminology. A *population* is formed by a set of individuals. An individual or *chromosome* consists of

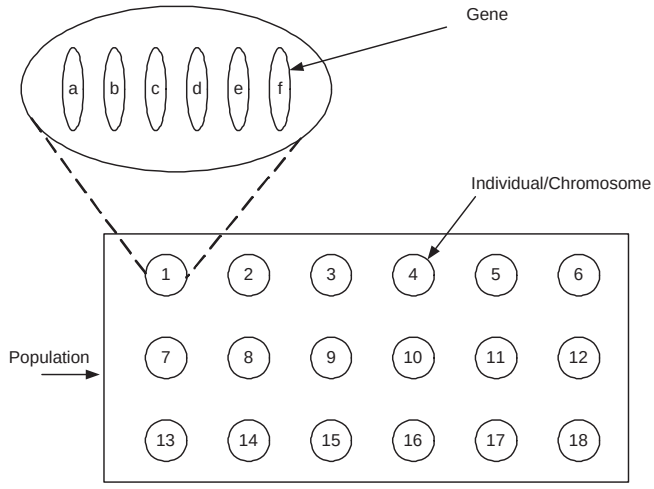

Figure 1.7: GA Terminology: a population is formed by individuals and each individual has a set of genes.

genes, see Figure 1.7. Depending on the selection of these genes the *fitness* of an individual can be evaluated. The actual algorithm operates on the chromosomes and their genes with genetic-inspired operators such as *crossover*, *mutation* and *inversion*. The assumption is the following: given a set of good parents, i.e. chromosomes with high fitness, the children that are created will have an even better fitness, at least in the long run. Most GA utilize *haploid* chromosomes, i.e. unpaired chromosomes. Most sexually reproducing species are *diploid*. In the *haploid* case crossover consists of exchanging genetic material between two single chromosome parents. Mutation consists of randomly changing the values of genes. Inversion rearranges the order of the genes, but it is rarely used.

Example

Given a simple organism with only one chromosome, which has two genes g_1 and g_2. The genes can only assume two values, 0 or 1. A chromosome can be described as (g_1, g_2). In this case the search space is only 2 by 2, a very limited space. Another important concept is the fitness landscape, which can be seen as a combination of the search landscape and the fitness value in the positions. In

the example above the search space can be represented by a 2D-square on the X- and Y-axis, and the fitness landscape would then be the value on the Z-axis. The goal of the fitness function, *FF*, is to evaluate the given solution's fitness. The use of a FF is very powerful, since it enables the GA to be totally oblivious to the problem at hand. On the other hand, it is extremely crucial that the fitness is correctly evaluated by the FF. How the fitness of a individual is evaluated is determined by the goal of the optimization. Continuing on the example from above, the fitness function could simply be defined as: $f() = g_1 \& g_2$. Considering another example, in order to maximize the function $f(x) = \cos(x) + \sin(x)$, the FF would just be chosen as $f()$, and the individual would only require one gene x. For a more detailed introduction see [Mit96, BNKF98].

Simple Genetic Algorithm

Given a well defined problem, where the candidate solutions are defined by strings of m bits, a simple GA operates as follows:

1. Generate a population based on n individuals, each with m genes, randomly.
2. Calculate the fitness $f(x)$ for each individual.
3. Repeat until n offsprings have been created:
   (a) Select a pair of parents, where the probability of being selected increase with fitness. Selection is done with replacement; which means that the same individual can be selected (to become a parent) several times.
   (b) With probability p_c (crossover probability), cross over the pair at a randomly given point. If no crossover occurs, form two offsprings that are exact copies of their parents.
   (c) Mutate the two offsprings at each gene with probability p_m (mutation probability), and place the resulting individuals in the new population.
   (d) If n is odd, one of the new population member should be discarded at random.
4. Replace the current population with the new.
5. Go to 2, and repeat until a stopping condition is reached.

The stopping condition can be either N iterations, callled *generations*, or until a certain fitness level is reached. A GA is typically run for 50 to 500 generations.

The entire set of iterations is called a *run*. Due to the randomness of the GA, runs will generally produce different results. Hence it is recommended that several runs are performed using different initial populations.

1.4.2 Simulated Annealing

Annealing is a term from metallurgy. When a metal is heated to a very high temperature, the metal enters a liquid state. In this state the atoms are quite mobile. When the liquid is cooled slowly, the atoms settle into a pattern rendering the metal much stronger than before. This principle can be employed as an optimization technique. The first application using *Simulated Annealing* (SA) for optimization was presented by Kirckpatrick [KGV83]. The algorithm itself was based on the use of statistical mechanics, demonstrated by Metropolis et al, [MRR+53].

Terminology

The two main terms are *annealing temperature* (β) and *annealing schedule*. The annealing temperature controls how fast the metal is cooled. The annealing schedule specifies how the annealing temperature should change during the search. Initially the search should allow solutions to be selected from a large area, and as the search continues the area should shrink in order to find an optimum.

Simple Simulated Annealing Algorithm

1. Select an initial solution $\vec{X_0}$, calculate the value at this point $f_0 = f(\vec{X_0})$.
2. Choose a random point and establish a search direction S.
3. Take a small step α in this direction and calculate the value at this point
$$f_1 = f(\vec{X_0} + \alpha S) \quad \delta f = f_0 - f_1$$
4. If $\delta f \leq 0$, then $p = 1$, else $p = e^{-\beta \delta f}$.
5. Generate a random number r, if $r \leq p$, then the step is accepted and the solution is updated. Otherwise no change is made to the solution. Go to 2.

For a more detailed discussion about simulated annealing see [Ven01, Per03].

1.4.3 Comparison of GA and SA

Genetic Algorithms are still in their infancy, only with systematically studies during the last 10−15 years. This places GA in a research and development phase; hence there are lots of opinions on the advantages and disadvantages of GA. To decide if a GA will perform good or bad, the first issue is to study the problem at hand. The main advantage of GA is their application to problems where there is little or no knowledge about it. This is also a weakness of GA. A GA can for instance find a suitable solution. However, this solution might not be the global optimum. The time to find this solution might also be longer than a problem-aware algorithm requires. In many cases, the solution produced by a GA can be worse than the solution produced by an algorithm especially designed for the problem. However, the solution from the GA might be sufficient for some applications. Thus the cost of developing a problem aware algorithm has to be weighted against the simplicity of the GA.

Simulated Annealing has been used for some time, and it is a mature and well developed optimization procedure. One advantage is its ability to move away from local optima. Thus, the ability to find the global optimum is not related to the initial conditions. One disadvantage for SA is the time consumption.

Both GA and SA are algorithms that come with a certain randomness, which can cause the algorithms to do random walks in the fitness landscape. Hence, runs will rarely produce exactly the same result. This is usually solved by increasing the number of iterations, or doing several independent runs.

1.5 The Next Step – Fluid Flow Analysis

The stochastic fluid flow model is an analytical tool for the calculation of performance parameters. Fluid flow analysis operates on flows, which means that is cannot handle individual bits or packets but flows or streams of bits or packets. This is usually not a problem, since most network traffic usually involves streams of many bits or packets. Fluid flow analysis is based on two matrices, the transition rate matrix $\mathbf{M}$ and the diagonal drift matrix $\mathbf{D}$, whose elements d_s show how the buffer content behaves in a particular state s:

- Positive drift $d_s > 0$: the buffer content rises until the buffer overflows; s is an overload state.
- Vanishing or zero drift $d_s = 0$: The buffer contents remain unchanged.

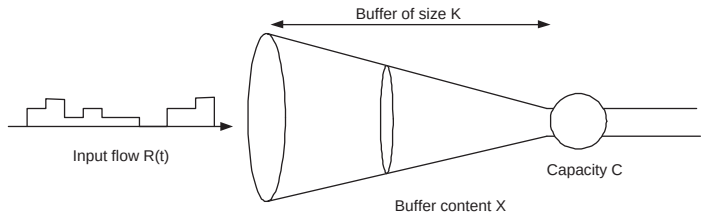

Figure 1.8: Fluid Flow Funnel.

- Negative drift $d_s < 0$: the buffer sinks until the buffer becomes empty; s is an underload state.

The matrices $\mathbf{M}$ and $\mathbf{D}$ are formed by Kronecker addition (see [SE91] and Appendix A) of the processes that are operate on the fluid system

$$\begin{aligned}\mathbf{M} &= \mathbf{M}_0 \oplus \mathbf{M}_1 \oplus \cdots \oplus \mathbf{M}_n \\ \mathbf{D} &= \mathbf{R}_0 \oplus \mathbf{R}_1 \oplus \cdots \oplus \mathbf{R}_n\end{aligned}$$

$\mathbf{M}_i$ is the transition matrix for the i:th process (which in turn can be constructed from other processes) and $\mathbf{R}_i$ is the corresponding rate matrix. The goal of this work is to construct these matrices for a process that describes Ethernet traffic. Furthermore, the transition matrix has to be a Markov process in order for the fluid flow analysis to operate.

Once these matrices are known, we can use fluid flow analysis to obtain the vector of joint buffer content distribution $\vec{F}(x)$ for a system with a buffer of size K and content X $(0 \leq X \leq K)$, see Figure 1.8. The elements in $\vec{F}(x)$, are $F_s(x) = \mathbf{Pr}\{\text{buffer content } X \leq x \wedge \text{State } S = s\}$, from which the performance parameters of interest can be derived. The system of σ differential equations to be solved is given by

$$\mathbf{D}\frac{\mathrm{d}}{\mathrm{d}x}\vec{F}(x) = \mathbf{M}\vec{F}(x) \tag{1.1}$$

The set of eigenvalues and right-hand-side eigenvectors

$$\{\{z_q\}, \{\vec{\phi}_q\}\}, q = 0 \ldots \sigma - 1$$

can be obtained by solving $z_q\mathbf{D}\vec{\phi}_q = \mathbf{M}\vec{\phi}_q$. Let $\vec{\pi}$ denote the vector of state probabilities ($\pi_s = \mathbf{Pr}\{S = s\}$) and $\{a_q\}$ a set of coefficients determined by the

boundary conditions. Then the solution to (1.1) and its complement $\vec{G}(x)$ is

$$\vec{F}(x) = \sum_{\forall q} a_q \vec{\phi}_q e^{z_q x} \tag{1.2}$$

$$\vec{G}(x) = \vec{\pi} - \vec{F}(x) \tag{1.3}$$

From these equations we can calculate the individual loss probabilities, delay quantiles [CF00, FCN01] and the bit rate distribution at the output [FT03]. All of these are essential for performance evaluation. A detailed description of fluid flow analysis [AMS82, SE91] and its numerical treatment can be found in [CF00, FV00, FV97].

Chapter 2

State of the art

In this chapter a short overview of previous work is given. This includes a short discussion on the Bellcore traces [BCT], a set of network traces that are available on the Internet and a review on network measurements. The chapter concludes with work on network modelling.

2.1 Bellcore Traces

Measurements should be treated as produces[1], which are best when fresh. Measurements that capture an entire link carrying the traffic from a mixture of applications should be treated carefully. The Bellcore traces were collected during 1989 at the Bellcore Morristown Research and Engineering facility. They were used as the base for [LTWW93, LTWW94], who introduced the self-similarity and long-range dependence (LRD) concepts in networking. These traces were made available to the public, and have since then been used extensively in research.

When these traces were made in 1989 the number of hosts on the Internet were approximately a quarter of a million; today the estimate is closer to 170 million (January 2003) [ISC]. The predominant traffic type in 1989 was FTP[2] and NNTP[3]; today the predominant traffic is HTTP[4] and Peer-2-Peer (P2P), see Table 2.1 for a comparison. Please notice that the statistics are collected at dif-

[1] Farm products, i.e. vegetables
[2] File Transfer Protocol
[3] Network News Transfer Protocol
[4] Hyper Text Transfer Protocol

Table 2.1: Percentage of network application usage, compared on the amount of bytes. The 1990−1995 data is based on the traffic on the NSF backbone. The 2000−2003 is based on IPmon taffic, sj-00.0-000809, sj-02.0-020806 and sj-20.0-030407, respectively.

Application	**1990**	**1993**	**1994**	**1995**	**2000**	**2002**	**2003**
P2P					2.4	25.4	46.6
HTTP		2.2	16.0	25.3	83.5	31.3	14.84
SMTP	7.6	6.0	5.6	4.6	1.6	5.20	2.01
FTP	44.6	40.9	31.7	21.5	1.0	1.51	0.46
TELNET	6.2	5.3	3.9	2.5			
NEWS	10.4	9.7	10.9	8.6			
Others		4.3	5.0				
Transport Protocol	**1990**	**1993**	**1994**	**1995**	**2000**	**2002**	**2003**
UDP					4.8	3.1	3.4
TCP					95.0	96.1	96.3

ferent sites; the 1990−1995 statistics were obtained from the NSF[5] backbone and describe the traffic that traversed the NSF backbone during one month [Mer]. The last three were collected from Sprints IPMON project [IPM]. Unfortunately the same information is not possibe to obtain from the Bellcore traces.

In addition to this change of usage, several new technologies (see Table 1.1 for some important Ethernet standards) have appeared. Thus, these traces (or any traces) should be used with care when drawing conclusions about the network behaviour of today.

In Figure 2.1 a comparison is done on the frame sizes. The comparision is done between the Bellcore Oct89Ext4 trace and two traces IAL-1-IN and IAL-1-NI (see Chapter 6). These traces were captured on a link that supplies approximately 300 students with Internet access. There are two traces, since the link was 100BaseT full-duplex. For the Bellcore trace the mean packet size was 258, for IAL-1-IN and IAL-1-NI it was 815 respectively 329 bytes. It is also interesting that the IAL-1-IN and IAL-1-NI links have captured frames smaller than the minimum frame size (64 bytes), which may be due to two reasons: either there is a device that sends frames of this size, even though it is against the standard, or these are

[5] US National Science Foundation

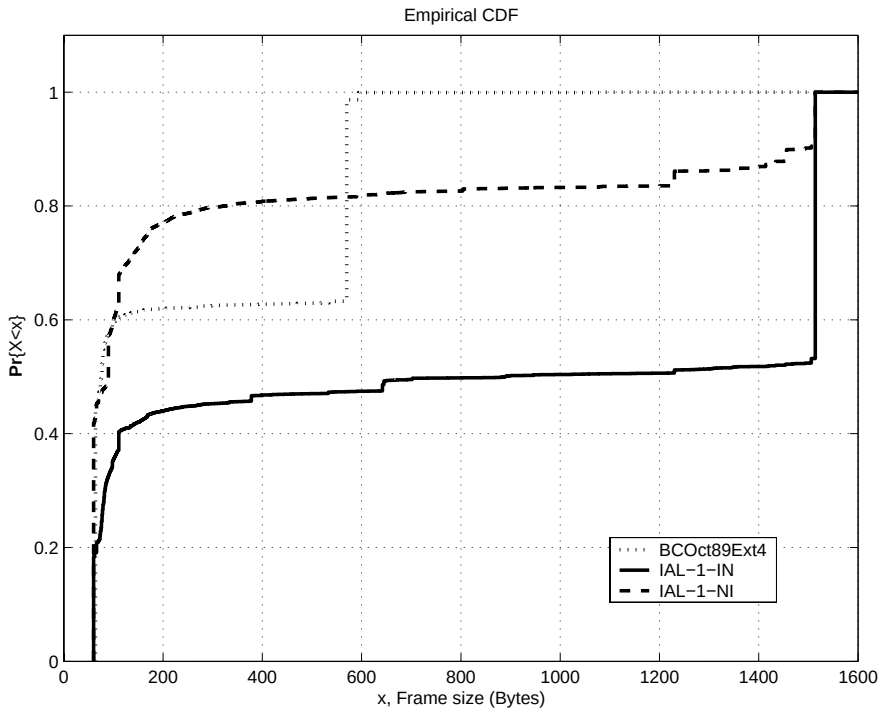

Figure 2.1: Cumulative distribution function of frame sizes for different traces.

remainders, (runts) of frames that have been damaged due to collisions. In Figure 2.2, the interarrival time for the same traces is compared. The mean interarrival time for the Bellcore trace was 76 ms, where as IAL-1-IN and IAL-1-NI had a mean interarrival time of 0.28 ms respectively 0.34 ms.

2.2 Network Measurements

Today, the many of measurements performed are based on TCPDUMP and PCAP derivatives. These are highly dependent on the hardware and operating system that is running them. In [LW91] a measurement system is described that has an timestamp accuracy of 10 μs. This was obtained when using a single

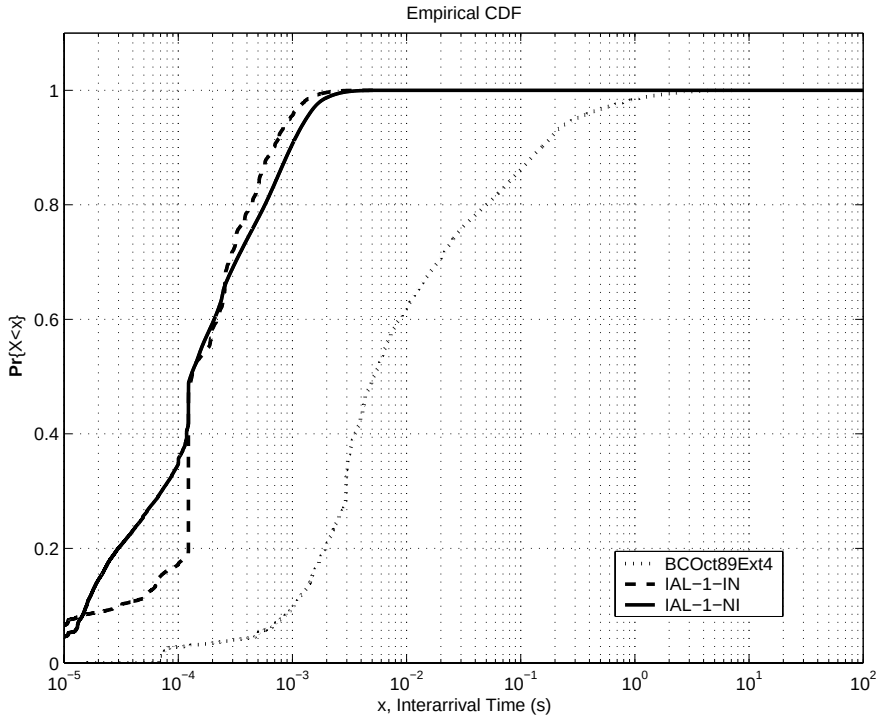

Figure 2.2: Cumulative distribution function of interarrival times for different traces.

Table 2.2: Statistics for the BCOct89Ext4, IAL-1-IN and IAL-1-NI traces used in Figure 2.1 and Figure 2.2.

	Frame Size [bytes]			**Interarrival Time [s]**		
	BC	IAL-1-IN	IAL-1-NI	BC	IAL-1-IN	IAL-1-NI
min	64	60	60	1.6×10^{-5}	6.6×10^{-6}	6.7×10^{-6}
max	1518	1514	1514	8.03	0.006	0.008
mean	258	815	329	0.076	2.8×10^{-4}	3.4×10^{-4}
median	77	888	90	0.005	1.3×10^{-4}	1.3×10^{-4}
std.dev	242	695	510	0.296	3.3×10^{-4}	4.7×10^{-4}

board computer with an onboard LANCE Ethernet interface, which was attached to a SUN-3 workstation. This high accuracy was only obtained when the computer was not saving the data to disk. During the phase of saving the data to disk the accuracy of the timestamp grew to 50 μs. Further, the accuracy was highly dependent on the frame rate. When the frame rate was 8000 packets per second, the accuracy reached 100 μs. In [MDG01] the authors discuss how to do precision timestamping of network packets, and in [Don02] the author gives a more general discussion on timing in passive measurements.

2.3 Traffic Modelling

In [HKS99] a good overview of different methods to model traffic sources is given. In [DSI99] traffic is modelled using universal multifractals. The authors model packets and arrival times and their model operates on the network level.

In [NAAM] a model for self-similar Ethernet traffic is presented. This model is used in SimATM, an ATM network simulator. The authors use the Bellcore traces [BCT] as a base for their Ethernet model. They model the mean packet size and variation, in addition to the upper and lower packet sizes. This is a self-similar model and it can only approximate a multifractal behaviour.

A similar class to the MMRP is the Markov Modulated Poisson Processes (MMPP). For MMPP a Poisson process replaces the rate process. The use of MMPPs in traffic modelling is quite large, see e.g. [SV01b, YKT01, AN98, SV01a, SHLN00]. A MMPP is used in [SVP03] to match the autocovariance and marginal distribution of a process. The parameter that the authors matches is the frame

rate, not the bit rate. Their model is a superposition of L 2-MMPPs and one M-MMPP. The 2-MMPPs are used to model each timescale, and the M-MMPP matches the marginal distribution, hence the resulting process is a $M2^L$ state process. This is a very interesting model, especially since [SVP03] presented a fitting procedure that seems to operate very well. However, since they use the MMPP, it is not easily translated into a model that is usable in fluid flow analysis.

In [PBF+02] a multiscale wavelet transform is used to analyse teletraffic in general. The authors show that the wavelet transform can handle several types of traffic including self-similar, fractal and multifractal. The applicatibility of this model to fluid flow analysis is however an unresolved issue.

Cumulants are used in [MT01] to model Internet traffic by a monofractal model. The authors argue that a monofractal model is flexible enough to capture the scaling behaviour. To capture traffic, they used a switch that support port mirroring; they mirrored the most loaded port of the switch and sent the traffic to a Linux computer running TCPDUMP. When this particular switch is instructed to do port mirroring, the mirrored port uses a store-and-forward; this prevents frames with CRC errors from being forwarded. On the other hand, if the mirrored port has a total load (input and output) that exceeds the recipient port there can be collisions within the switch, and frames will not reach the monitoring computer. Further, no details are given regarding the load on the mirrored port, nor is it specified what version of TCPDUMP that was used, which is necessary to find out the timestamp accuracy. Furthermore, the most loaded link does not have to be the link that carries the Internet traffic. In fact, based on their map it is clear that the most loaded link is the link between the switch and the router. This link is used to transport network traffic from one department group (DG) to another, in addition to DG to Internet traffic, hence they will capture both LAN traffic and WAN traffic. They model the measured TCP data in bytes in 100 ms intervals. But, similarly to [SV01b] it is unclear how to apply these results to a fluid analysis.

Chapter 3

Measurements and Moment Estimation

In this chapter, we discuss the process of collecting and calculating the moments of the bit rate at the link layer at multiple timescales. Furthermore, the bit rate estimation will be based on goodput bits, which implies that only bits that are a part of correctly received frames at the wiretap will be used in the estimations.

3.1 Measurements

The goal with the measurements is to estimate the bit rate with as few errors as possible. The analysing software operates off-line on files that contain packet traces This allows for almost arbitrary sample rates. In an on-line scenario the upper sample frequency would be hampered by the processing power of the measuring device. By working off-line we can analyse the measured data in more detail, without interfering with the measurements. The traces are stored to files, where the packets are stored according to a special format presented in Figure 3.1 and Figure 3.2. Each file starts with a file header (FH) containing information on where the trace was obtained and what version of the file formats is used. Each captured frame is stored together with a capture header (CH). This header contains information on where and when the frame was captured. Parts of the captured frame can be stored after the CH.

These files can be constructed from conversion or capturing. Conversion is used when traces are already available, for instance from TCPDUMP. Capturing

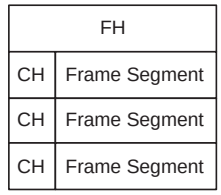

Figure 3.1: Structure of the files used to store the packet traces, FH=File Header, CH=Capture Header.

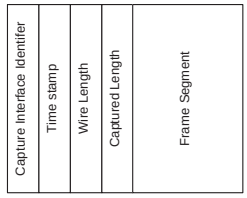

Figure 3.2: Structure of the capture header.

is done when it is possible to install a wiretap. The actual capturing is done by a measurement point (MP), to which a short overview is given below; for a more detailed description, see [Car03]. The MP is the key component in a passive measurement infrastructure (PMI). The PMI will not be described here since it will not be utilized, for more information about PMI see [CEF03].

Depending on how the traces were obtained, either from conversion or capture, the accuracy of the timestamp is determined by the original capturer. For instance, traces from DAG cards have an accuracy of less than 100 ns [ENDb], whereas traces from PCAP have an accuracy in the order of ms to μs depending on the system that ran PCAP [LW91, MDG01, Don02]. Furthermore, in a trace file produced from PCAP the timestamps indicate when the frame was completely received by the NIC or Kernel, whereas in traces based on DAG-card captures indicate when the frame started to arrive. The software accounts for this and recalculates the arrival time in the case of TCPDUMP measurements:

$$\text{Packet Arrival Time} = \text{Timestamp} - \frac{\text{Packet Length}}{\text{Link Capacity}}$$

We use passive measurements in order to get a complete and undistorted

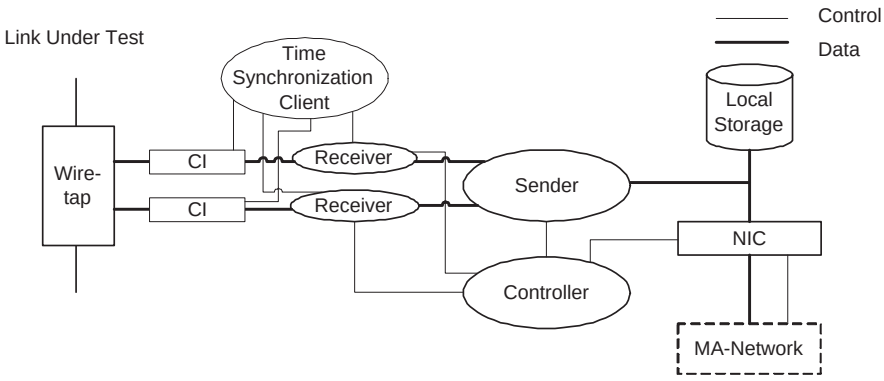

Figure 3.3: Schematic overview of a measurement point.

view of the properties of the measured link. SNMP could have been used, but the accuracy of SNMP based network measurements is not sufficient to capture short timescales [CFT+02].

Measurement Point

The main component of the PMI, and the only one that we utilized here, is the measurement point (MP) [Car03]. In Figure 3.3 the components of a schematic MP are shown. This device is responsible for the actual packet capturing. It is managed either from a MA-controller (not shown here) or from a local console. The data that it captures is sent to one or more consumers. A consumer can either be a device attached to the MA-network or a local file. An MP can be either logical or physical. A logical MP is simply a measurement process running on a host, whereas a physical MP uses a dedicated computer with a wiretap attached to it or custom hardware (ideally ASICs in order to create high-speed high-performance MPs).

A MP can tap one or more links. For each link a wiretap is needed. For full-duplex Ethernets the wiretap has two outputs, one for each direction. These are connected to separate capture interfaces (CI). A receiver listens to a CI and filters the packets according to filter rules stated by the controller. If the CI has not timestamped the frame then the receiver will do this. The packets are then delivered to the sender that is responsible for sending the captured frames to the correct consumers. Currently an MP supports two types of consumers: a local

consumer that stores the measurements to a disk located in the MP computer, and consumers attached to the MA-network. Each MP also has a controller that is responsible for the configuration of the MP and the communication with the MA-controller.

There is also a time synchronization client (TSC) that can use anything from a dedicated device (i.e. GPS, CDMA or optical) to a simple NTP server. It should receive the synchronization signal from a time synchronization device that is shared at least among all the MPs in a MA.

The filter rules specifies, in addition to traffic types of interest, a receiver and the amount of the frame to be captured. The MP can potentially do complete packet capturing, but actually it is fixed to an upper limit of 56 bytes, in order to capture link, network and transport headers. For each frame that passes the filter the MP attaches a capture header. In this header we store a CI identifier, a timestamp when the frame was captured (large enough to store a timestamp with an accuracy of pico seconds), the frame length and the number of bytes that actually were captured. The accuracy of the timestamp is decided by the capturing equipment.

3.2 Bitrate Estimation

The bit rate B_i in interval i is calculated as the number of bits that have arrived in sample interval i, divided by the sample interval duration T_s.

$$B_i = \frac{b_0 + \sum_{k=1}^{N} b_k + b_{N+1}}{T_s} \tag{3.1}$$

Here, b_0 are the bits belonging to interval i from a frame that started arriving prior to this interval. Similarly, b_{N+1} identify the bits of a frame that started arriving in this interval but was not completed. b_k are the bits of frames that were completed within the interval. See Figure 3.4 for an example. The sample interval T_s is determined by the desired base scale t_0.

3.2.1 Fractional Bits

Fractional bits occur when a sample instance is located within a bit, see Figure 3.5. This is caused by two reasons: Ethernet is asynchronous, and the timestamp accuracy is limited. Given that the accuracy is sufficient, the bit rate calculations can be adjusted to account for fractional bits simply by splitting them and adding

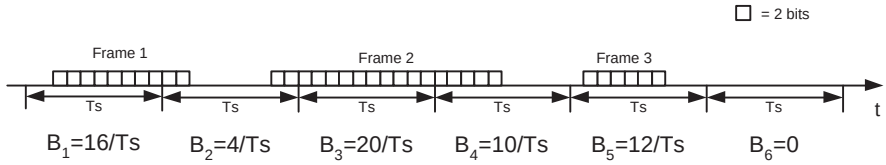

Figure 3.4: Estimation of the bit rate.

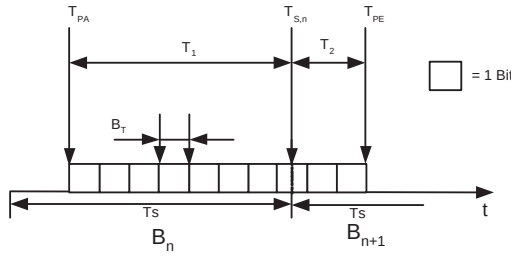

Figure 3.5: Visualization of a fractional bit. T_{PA} is the arrival time of the packet, T_{PE} is the ending time of the packet. $T_{S,n}$ is the arrival time of the n:th sample instance, and $B_T = 1/C$ is the bit transmission time where C is the link capacity.

the fractions to the bit counts in the related sample intervals. For the example shown in Figure 3.5, the bit rate would be calculated as

$$T_1 = T_{S,n} - T_{PA} \qquad T_2 = T_{PE} - T_{S,n}$$

$$B_n: \qquad b_0 = 0 \qquad \sum_{k=1}^{N} b_k = 0 \qquad b_{N+1} = T_1 C$$

$$B_{n+1}: \qquad b_0 = T_2 C \qquad \sum_{k=1}^{N} b_k = 0 \qquad b_{N+1} = 0$$

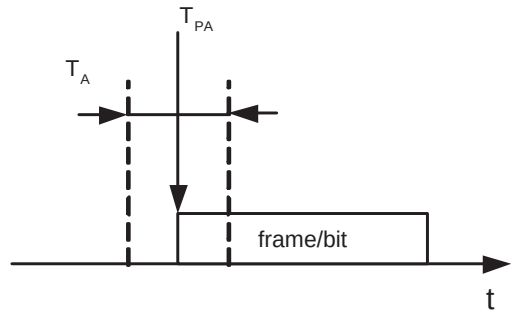

Figure 3.6: Timestamp inaccuracy. A packet is reported to have arrived at T_{PA}, but due to the accuracy of the timestamp we can not specify exactly when the packet arrived, but only within an interval of T_A seconds.

3.2.2 Timestamp Accuracy

Since all measurement equipment, both hardware and software, have a fixed timestamp accuracy, there will be errors in the bit rate estimation. The size of the error is related to the accuracy of the timestamp, T_A, and the sample interval, T_S, see Figure 3.6. The timestamp accuracy specifies how much a frame can be shifted, with regards to the timestamp. A frame can at most be shifted T_A seconds. In the worst case, this can cause $T_A \times C$ bits to be placed in incorrect interval(s). Thus, a rough estimate of the error is given by:

$$\text{Error} = \frac{\text{Maximal number of erroneously placed bits}}{\text{Maximal number of bits arriving in } T_S \text{ seconds}} = \frac{T_A C}{T_S C} = \frac{T_A}{T_S} \quad (3.2)$$

The real error actually larger, since in a real network environment the C in the denominator is smaller as it accounts for inter-frame gaps etc. For example in the case of a 10 Mbps Ethernet link this value is in the order of $7.5-9.7$ Mbps, depending on the frame size and their distribution, when the sample interval is 1 s. This would yield an error that is between 5% and 33% larger than the estimation above.

If the sample interval is relatively large these additional bits will only cause small errors. For instance, if $T_S = 1$ s and $C = 10$ Mbps, the maximal number of bits in a sample interval are (in theory) 10 million bits. Given a $T_A = 100$ ns, at most 1.0 bits will be placed in an incorrect interval. Using a PCAP-based

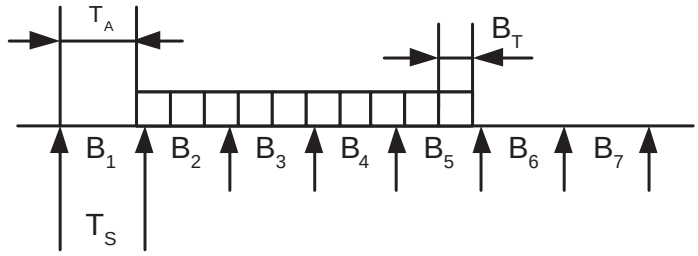

Figure 3.7: Example of how the error grows as the sample time decreases. Here at most 2.25 bits can be placed in an incorrect interval.

Table 3.1: Percentage of Error in Bit rate estimations, w.r.t. timestamp accuracy and sample interval.

T_S	$T_A = 100$ ns	$T_A = 10$ μs	$T_A = 1$ ms	$T_A = 10$ ms	$T_A = 1$ s
1 s	10^{-5}	10^{-3}	10^{-1}	1	10^{2}
0.1 s	10^{-4}	10^{-2}	1	10	10^{3}
10 ms	10^{-3}	10^{-1}	10	10^{2}	10^{4}
1 ms	10^{-2}	1	10^{2}	10^{3}	10^{5}
0.1 ms	0.1	10	10^{3}	10^{4}	10^{6}
10 μs	1	100	10^{4}	10^{5}	10^{7}
1 μs	10	1000	10^{5}	10^{6}	10^{8}

solution with $T_A = 10$ μs, at most 100 bits will be misplaced. In relation to the possible number of bits in an interval, this is an error of less than 0.001%. Obviously as the sample interval shrinks the error grows. This is shown in Figure 3.7, where the frame is shifted by T_A seconds, causing the last 2.25 bits to be placed in the next interval. In Table 3.1 a list of the error is given in percent, based on some typical timestamp accuracies. This error sets the lower limit on the timescales that we can use. Given a trace obtained from a DAG card, the base scale should not be smaller than 10 μs; for a PCAP-based trace, a lower bound of 1 ms applies.

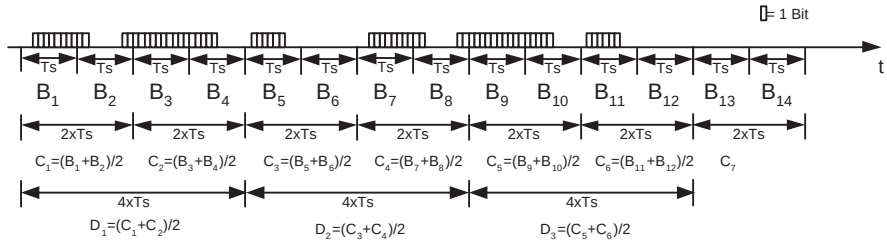

Figure 3.8: Estimation of bitrates at different timescales.

3.3 Moment Estimation

Based on the bit rate estimations we can evaluate the moments at different timescales. To do this we need to supply the desired base scale t_0 which decides $t_0 = T_s$, as well as a scaling factor. For simplicity, we have chosen to use the same scaling factor S, which is an integer larger or equal to 2. This factor specifies the distance between timescales $t_i = S \times t_{i-1}$, that is timescale i consists of S samples from timescale $i-1$.

Let us define X_k as the bit rate samples at the k^{th} timescale. Then the i^{th} moment at the k^{th} timescale, $\gamma_{i,k}$, is calculated from

$$\gamma_{i,k} = \frac{1}{M_k} \sum_{j=1}^{M_k} (X_{k,j})^i \tag{3.3}$$

where M_k is the number of samples that are present in X_k. This is formed from the samples of the lower timescales

$$X_{k,j} = \frac{1}{S} \sum_{m=(j-1)S+1}^{jS} X_{k-1,m} \tag{3.4}$$

with X_0 being the base scale obtained from the measurements.

In Figure 3.8 an example is shown, where the scale factor is $S = 2$. The measurements provide us with the $\vec{B}$ samples, which are equal to the $\vec{X_0}$ samples. From this we calculate two more timescales samples $\vec{X_1}$ and $\vec{X_2}$.

Using the definitions from above we obtain;

$$\vec{X_0} = [B_1, B_2, B_3, B_4, B_5, \ldots B_{14}]$$
$$\vec{X_1} = [C_1, C_2, C_3, C_4, \ldots C_7] = [(B_1 + B_2)/2, (B_3 + B_4)/2, \ldots (B_{14} + B_{13})/2]$$
$$\vec{X_2} = [D_1, D_2, D_3] = [(C_1 + C_2)/2, (C_3 + C_4)/2, \ldots (C_5 + C_6)/2]$$

$$\gamma_{1,0} = \frac{1}{14}\sum_{j=1}^{14} X_{0,j} = \frac{\sum_{j=1}^{14} B_j}{14} = \frac{66}{14} \qquad \gamma_{2,0} = \frac{1}{14}\sum_{j=1}^{14} X_{0,j}^2 = \frac{730}{14}$$
$$\gamma_{1,1} = \frac{1}{7}\sum_{j=1}^{7} X_{1,j} = \frac{\sum_{j=1}^{7} C_j}{7} = \frac{33}{7} \qquad \gamma_{2,1} = \frac{1}{7}\sum_{j=1}^{7} X_{1,j}^2 = \frac{202.5}{7}$$
$$\gamma_{1,2} = \frac{1}{3}\sum_{j=1}^{3} X_{2,j} = \frac{\sum_{j=1}^{3} D_j}{3} = \frac{16.5}{3} \qquad \gamma_{2,2} = \frac{1}{3}\sum_{j=1}^{3} X_{2,j}^2 = \frac{93.3750}{3}$$

From this example we also notice that the samples B_{13}, B_{14} and C_7 will not be used when estimating the moments on the third timescale.

3.4 Truncation

The number of timescales in the measurement is determined by the size of the trace. The largest timescale that can be used in moment estimation is specified by the length of a trace and the scale factor. Depending on the scale factor, the largest timescales will be based on only a few samples, (3.3) and (3.4). This can cause the moment estimations to be statistically highly unreliable. To minimize this, we truncate the result. We have found that a lower limit of approximately 30 samples for a timescale is usually good. Truncation is done by dropping entire timescales, for example

$$\begin{bmatrix} \gamma_{1,0} & \gamma_{2,0} & \gamma_{3,0} \\ \gamma_{1,1} & \gamma_{2,1} & \gamma_{3,1} \\ \gamma_{1,2} & \gamma_{2,2} & \gamma_{3,2} \\ \gamma_{1,3} & \gamma_{2,3} & \gamma_{3,3} \\ \gamma_{1,4} & \gamma_{2,4} & \gamma_{3,4} \end{bmatrix} \rightarrow \text{Truncation} \rightarrow \begin{bmatrix} \gamma_{1,0} & \gamma_{2,0} & \gamma_{3,0} \\ \gamma_{1,1} & \gamma_{2,1} & \gamma_{3,1} \\ \gamma_{1,2} & \gamma_{2,2} & \gamma_{3,2} \end{bmatrix}$$

where the fourth and fifth timescale contained less than 30 samples.

3.5 Normalization

We normalize the result in order to simplify the matching in a way that the first moment obtains an average of one over all the timescales. The normalization factor is then calculated as:

$$n_f = \frac{1}{N_T} \sum_{i=0}^{N_T-1} \gamma_{1,i} \tag{3.5}$$

Here N_T is the number of timescales remaining after the truncation. We update the moment estimations with this factor. Let $\vec{\gamma_l}$ denote a vector containing all timescales estimations for moment l, a normalized version is calculated as:

$$\vec{\gamma_l}_n = \frac{1}{n_f^l}\vec{\gamma_l} \tag{3.6}$$

Chapter 4

Process Description

This chapter is about the process and the sub-processes upon which the model of interest is built. First an overview of the process is given, followed by a detailed analysis of the sub-processes We conclude with an example of how a process is constructed.

4.1 Process

The model used here is based on a random process. To create a process that can capture multifractal properties we use a traffic model proposed by Mannersalo, Norros and Riedi [MNR02]. This model can exhibit multifractal properties:

> In its simplest form our model is based on the multiplication of independent rescaled stochastic processes $\Lambda^{(i)}(\cdot) \triangleq \Lambda(b^i)$ which are piecewise constant. ... In multiplying rather than adding re-scaled versions of a 'mother' process we obtain a process with novel properties which are best understood not in an additive analysis, but in a multiplicative one. Processes emerging from multiplicative construction ... exhibit typically a 'spiky' appearance [MNR02].

The mother- or base process is a Markov-modulated rate process (MMRP) with two activity states. The model presented by Mannersalo, Norros and Riedi uses a base process $\Lambda_0(t)$ and the other processes $\Lambda_i(t)$ are only rescaled versions of this base process, i.e. ${}^{(j)}\Lambda_i(t) = {}^{(j)}\Lambda_0(b^i t), b > 1$. Here, j stands for the j^{th} realisation of the process R, e.g. with random number seed Z_j. As the number of processes

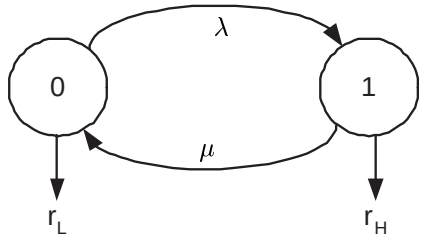

Figure 4.1: State diagram for a 2-state Markov Modulated Rate Process.

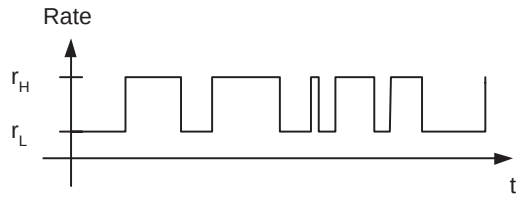

Figure 4.2: Example of output from a sub-process.

goes toward infinity an asymptotically self-similar process is obtained. In the following, we call the base process and the rescaled variants sub-processes.

Here a slightly modified process is used. First the number of sub-processes is limited. Second, the only requirement on the sub-processes is that they are independent, hence the scaling factor b is not used. The process is formed by multiplying the output of N independent sub-processes. Denote the output from sub-process i at time t with $R_i(t)$. The output from the process will be:

$$R(t) = \prod_{i=0}^{N-1} R_i(t) \quad \text{[bps]} \tag{4.1}$$

$R_0(t)$ can be considered to be modulated by other (unit-less) sub-processes $R_i(t), i \geq 1$. The key item is that $R(t)$ will be in bps. Since the process is formed by independent sub-processes, a detailed knowledge of the sub-process properties and parameters is desirable, hence a detailed analysis will follow.

4.2 Sub-process Analysis

The sub-processes are 2-state Markov-Modulated Rate Processes (MMRPs), each having four parameters: λ and μ are the transition rates in-between the states, r_L is the output rate in the 'low' state 0 and r_H is the output when the process is in the 'high' state 1. In Figure 4.1 a state diagram is shown, and in Figure 4.2 an example of a possible output is shown. Using this notation, we can express the transition matrix $\mathbf{M}$ and rate matrix $\mathbf{R}$ for sub-process i as:

$$\mathbf{M}_i = \begin{bmatrix} -\lambda_i & \lambda_i \\ \mu_i & -\mu_i \end{bmatrix} \qquad \mathbf{R}_i = \begin{bmatrix} r_{L,i} & 0 \\ 0 & r_{H,i} \end{bmatrix}$$

We define the cycle time as:

$$\tau_i = 1/\lambda_i + 1/\mu_i \tag{4.2}$$

This time specifies the mean time that it takes for the sub-process to go from one state to the other, and back to the original state. Thus, for a sub-process with a large cycle time the state changes will be rather infrequent, whereas a small cycle time indicates a sub-process with frequent state changes.

4.2.1 Moment Analysis

Using Figure 4.1, we can set up an equation to find the first moment. Since it is difficult to directly evaluate the moment for the output rate, we evaluate instead the counting process $N_i(T)$, which yields the rate process $R_i(T)$.

$$\mathbf{E}[R_i(T)] = \frac{\mathbf{E}[N_i(T)]}{T} \tag{4.3}$$

$N_i(T)$ is a process that describes how many bits that have arrived up to time T.

Below a sketch of the proof for the first moment is shown, and in Appendix B a detailed analysis is given for all three moments and their limits. $\mathbf{E}[N_i(T)]$ is obtained from;

$$\begin{aligned} \mathbf{E}[N_i(T)] = {} & \mathbf{Pr}\{\text{Process starts in state 0}\}\,\mathbf{E}[N_{i,0}(T)] \\ & + \mathbf{Pr}\{\text{Process starts in state 1}\}\,\mathbf{E}[N_{i,1}(T)] = \\ & \frac{\mu_i}{\lambda_i + \mu_i}\mathbf{E}[N_{i,0}(T)] + \frac{\lambda_i}{\lambda_i + \mu_i}\mathbf{E}[N_{i,1}(T)] \end{aligned} \tag{4.4}$$

The number of bits that have arrived in state 0 and 1, $N_{i,0}(t)$ and $N_{i,1}(t)$, are obtained from solving the following equation system:

$$\begin{cases} N_{i,0}(t) = & x_{i,0}(t) = r_{i,L}te^{-\lambda_i t} + \int_0^t \left[r_{i,L}\tau + N_{i,1}(t-\tau)\right]\lambda_i e^{-\lambda_i \tau}\mathrm{d}\tau \\ N_{i,1}(t) = & x_{i,1}(t) = r_{i,H}te^{-\mu_i t} + \int_0^t \left[r_{i,H}\tau + N_{i,0}(t-\tau)\right]\mu_i e^{-\mu_i \tau}\mathrm{d}\tau \end{cases} \tag{4.5}$$

In a similar manner the other two moments can be obtained. The first three moments are found to be:

$$\Gamma^1(T) = \mathbf{E}\left[R_i^1(T)\right] = \frac{r_{L,i}\mu_i + r_{H,i}\lambda_i}{\lambda_i + \mu_i} \tag{4.6}$$

$$\begin{aligned}\Gamma^2(T) = \mathbf{E}\left[R_i^2(T)\right] = \frac{2\lambda_i\mu_i(r_{L,i} - r_{H,i})^2}{(\lambda_i+\mu_i)^3}\left(\frac{1}{T} + \frac{e^{-(\lambda_i+\mu_i)T} - 1}{(\lambda_i+\mu_i)T^2}\right) \\ + \frac{(r_{L,i}\mu_i + r_{H,i}\lambda_i)^2}{(\lambda_i+\mu_i)^2}\end{aligned} \tag{4.7}$$

$$\begin{aligned}\Gamma^3(T) = \mathbf{E}\left[R_i^3(T)\right] = \frac{6C}{\lambda_i+\mu_i}\left(\frac{1}{T^2} - \left(\frac{1-e^{-(\lambda_i+\mu_i)T}}{(\lambda_i+\mu_i)T^3}\right)\right) + \frac{3D}{\lambda_i+\mu_i}\frac{1}{T} \\ + \frac{6F}{(\lambda_i+\mu_i)^3}\left(\frac{1-(\lambda_i T + \mu_i T + 1)e^{-(\lambda_i+\mu_i)T}}{T^3}\right) + \frac{G^3}{(\lambda_i+\mu_i)^3}\end{aligned} \tag{4.8}$$

where

$$\begin{aligned} A &= (r_{i,L}^3\mu_i + \lambda_i r_{i,H}^3) \\ B &= (2\mu_i^2 r_{i,L}^3 + 2\mu_i\lambda_i r_{i,L}^2 r_{i,H} + 2\lambda_i r_{i,L} r_{i,H}^2\mu_i + 2\lambda_i^2 r_{i,H}^3) \\ C &= \frac{A}{(\lambda_i+\mu_i)^2} - \frac{2B}{(\lambda_i+\mu_i)^3} + \frac{3G^3}{(\lambda_i+\mu_i)^4} \\ D &= \frac{B}{(\lambda_i+\mu_i)^2} - \frac{2G^3}{(\lambda_i+\mu_i)^3} \\ F &= \frac{B}{(\lambda_i+\mu_i)^2} - \frac{A}{\lambda_i+\mu_i} - \frac{G^3}{(\lambda_i+\mu_i)^3} \\ G &= r_{i,L}\mu_i + \lambda_i r_{i,H} \end{aligned}$$

In this work, we confine ourselves to matching the first three moments. Taking even higher moments into account and making them match to measured data would create an even better model, but also make the matching much more problematic.

4.2.2 Numerical Problems

Due to numerical issues that have their roots in the discretization of real values on computers, the equations above sometimes get into trouble. To handle this we calculate the limits of the moments as well.

$$\Gamma_0^1 = \lim_{T\to 0} \mathbf{E}\big[R^1(T)\big] = \frac{r_{L,i}\mu_i + r_{H,i}\lambda_i}{\lambda_i + \mu_i} \tag{4.9}$$

$$\Gamma_\infty^1 = \lim_{T\to \infty} \mathbf{E}\big[R^1(T)\big] = \frac{r_{L,i}\mu_i + r_{H,i}\lambda_i}{\lambda_i + \mu_i} \tag{4.10}$$

$$\Gamma_0^2 = \lim_{T\to 0} \mathbf{E}\big[R^2(T)\big] = \frac{\lambda_i\mu_i(r_{L,i} - r_{H,i})^2}{(\lambda_i + \mu_i)^2} + \frac{(r_{L,i}\mu_i + r_{H,i}\lambda_i)^2}{(\lambda_i + \mu_i)^2} \tag{4.11}$$

$$\Gamma_\infty^2 = \lim_{T\to \infty} \mathbf{E}\big[R^2(T)\big] = \frac{(r_{L,i}\mu_i + r_{H,i}\lambda_i)^2}{(\lambda_i + \mu_i)^2} \tag{4.12}$$

$$\Gamma_0^3 = \lim_{T\to 0} \mathbf{E}\big[R^3(T)\big] = \frac{r_{L,i}^3\mu_i + r_{H,i}^3\lambda_i}{\lambda_i + \mu_i} \tag{4.13}$$

$$\Gamma_\infty^3 = \lim_{T\to \infty} \mathbf{E}\big[R^3(T)\big] = \frac{(r_{L,i}\mu_i + r_{H,i}\lambda_i)^3}{(\lambda_i + \mu_i)^3} \tag{4.14}$$

Using this information, we can modify the moment equations so that they do not exhibit numerical problems. When a moment is calculated for a given timescale, each result is checked. Let $\gamma_{i,k}$ denote the i^{th} moment at the k^{th} timescale, T_k.

$$\gamma_{i,k} = \begin{cases} \Gamma_0^i & \text{if } T_k \leq \epsilon\tau \\ \Gamma_\infty^i & \text{if } T_k \geq \epsilon^{-1}\tau \\ \gamma_{i,k} & \text{else} \end{cases} \tag{4.15}$$

For an example see Figure 4.3 and Figure 4.4. To support this, a gradient analysis was performed, starting with the second moment.

$$\frac{\mathrm{d}\Gamma^2(T)}{\mathrm{d}T} = -\frac{2\lambda_i\mu_i(r_{L,i} - r_{H,i})^2}{(\lambda_i + \mu_i)^4 T^3} \cdot \Big(T(\mu_i + \lambda_i) + T(\mu_i + \lambda_i)e^{-(\lambda_i+\mu_i)T} + 2e^{-(\lambda_i+\mu_i)T} - 2)\Big) \tag{4.16}$$

From this it is clear that λ_i and μ_i have the same weights, i.e. their internal relation is almost irrelevant; the important value is τ_i. The second comment is

Table 4.1: Gradient behaviour for the second moment and third moment. To obtain these results we set $\lambda = \mu = 2$, $r_L = 0$ and $r_H = 2$

T	**Gradient** $\Gamma^2(T)$	**Gradient** $\Gamma^3(T)$
1τ	-0.26374	-0.79121
10τ	-0.00475	-0.01425
100τ	-4.975×10^{-5}	-0.00014925
$10^3\tau$	-4.9975×10^{-7}	-1.4993×10^{-6}
$10^4\tau$	-4.9997×10^{-9}	-1.4999×10^{-8}
$10^5\tau$	-5×10^{-11}	-1.5×10^{-10}
$10^6\tau$	-5×10^{-13}	-1.5×10^{-12}

that the influence of $r_{L,i}$ and $r_{H,i}$ is that of a constant factor that effects all timescales equally. Given this, we can rewrite (4.16):

$$\frac{\mathrm{d}\Gamma^2(T)}{\mathrm{d}T} = -\frac{2\lambda_i^2(r_{L,i} - r_{H,i})^2(2T\lambda_i + (2T\lambda_i)e^{-2\lambda_i T} + 2e^{-2\lambda_i T} - 2)}{2^4\lambda_i^4 T^3} = \\ -\frac{\lambda_i^2(r_{L,i} - r_{H,i})^2}{4} \cdot \frac{T\lambda_i - 1 + (T\lambda_i + 1)e^{-2\lambda_i T}}{\lambda_i^4 T^3} \quad (4.17)$$

Now, taking the limit as $T \to \infty$ we get 0. In Table 4.1 we evaluate the second and third moments gradients for a couple of $T \geq \tau_i$. We observe that the gradient goes to zero very rapidly as T moves away from τ_i. By using variable substitution ($U = 1/T$) we see that the gradient behaves the same when $T \to 0$. This is later confirmed when viewing the parameter plots. Obviously, the factor ϵ in Equation 4.15 should be larger than 100, and we use $\epsilon = 1000$.

4.2.3 Parameters

Now that we have expressions for the moments, their limits and the gradient, we turn our attention to the parameters. In order to simplify the comparision we start the analysis by evaluating parameter behaviour given that $\mathbf{E}\big[R^1(T)\big] \equiv 1$.

Assuming $r_{L,i}/r_{H,i}$ constant we vary λ_i and μ_i. As pointed out before, the gradient (Equation 4.16) depends only on $\lambda_i + \mu_i$, or τ_i. Thus, we only need to view the symmetrical case with $\lambda_i = \mu_i$. By increasing τ_i, the curve moves to the right, and when decreasing it, the curve moves to the left, see Figure 4.5.

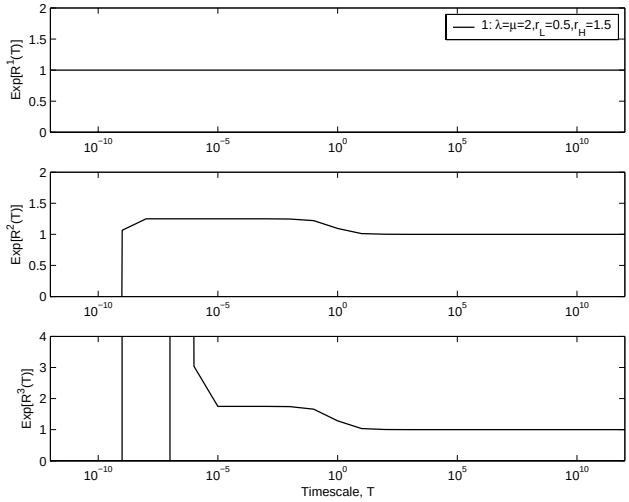

Figure 4.3: Example of moments when there are numerical issues in the moment equations.

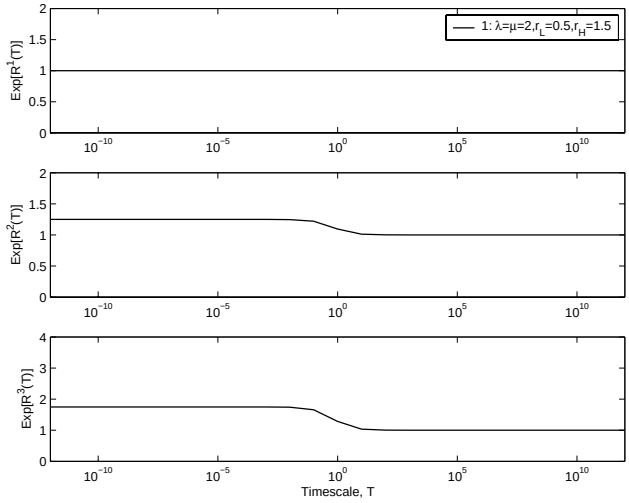

Figure 4.4: Example of moments when the numerical problems in the moment equations have been accounted for.

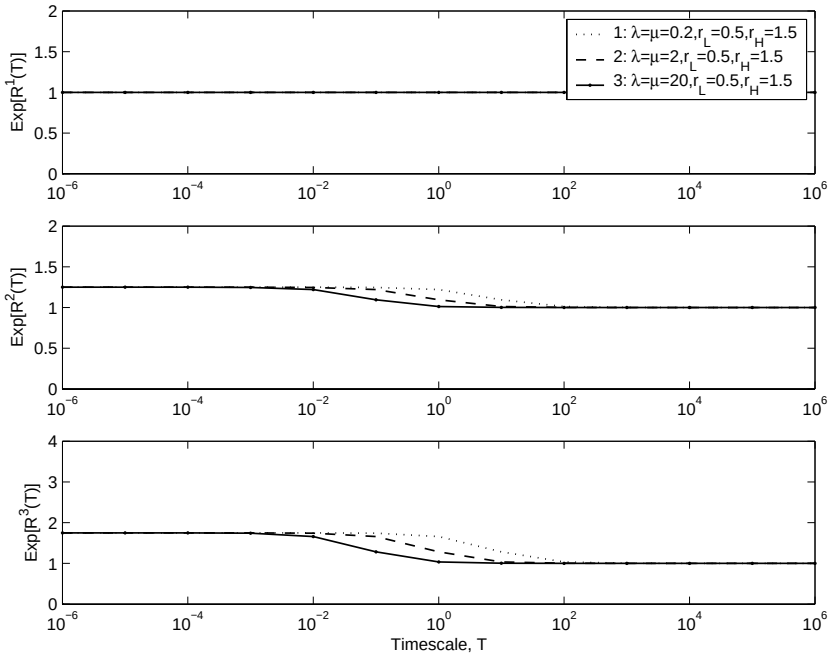

Figure 4.5: Sub-process with fixed output rates and varying transition rates.

One set of parameter values that are especially interesting is when $r_{L,i} = 0$. In this case the sub-process becomes an On-Off source, similarly to an Ethernet interface port. A source of this type exhibits the highest variability, and thus the highest second moment as well. Given symmetry ($\lambda_i = \mu_i$) and $\mathbf{E}\left[\Gamma^1(T)\right] = 1$, $r_{H,i}$ becomes 2. Then, the maximal value that the second moment can obtain is

$$\Gamma_0^2 = \frac{\lambda_i^2 4}{(2\lambda_i)^2} + \frac{4\lambda_i^2}{(2\lambda_i)^2} = 1 + 1 = 2 \tag{4.18}$$

Similarly, the third moments' maximum value becomes

$$\Gamma_0^3 = \frac{8\lambda}{2\lambda} = 4 \tag{4.19}$$

By varying $r_{L,i}$ and $r_{H,i}$ the shape can be made to move up and down. An example of this is given in Figure 4.6. In case the sub-process is symmetrical

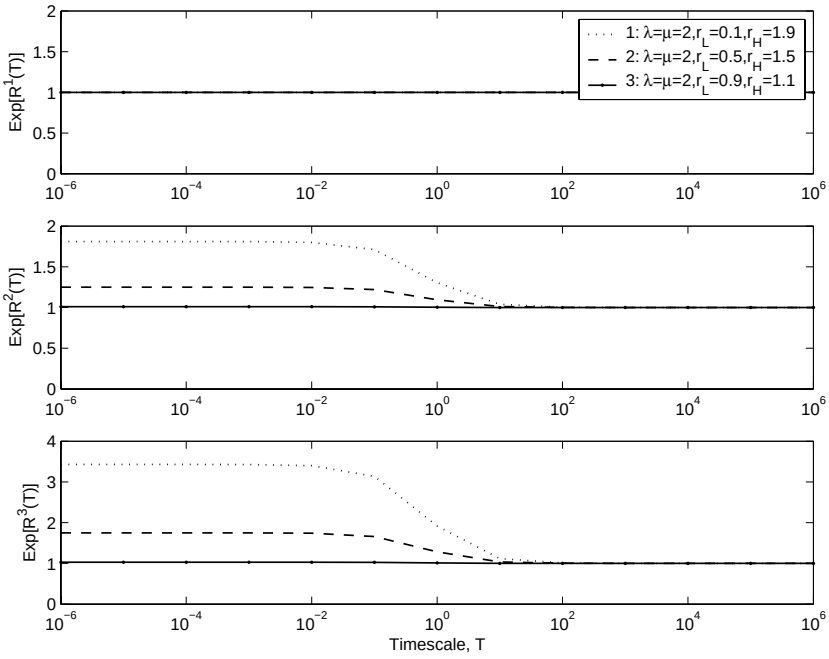

Figure 4.6: Sub-process with fixed transition rates and varying output rates.

w.r.t. transition rates ($\lambda_i = \mu_i$), we obtain even symmetry of the output rates as compared to the mean output rate ($r_{H,i} - \mathbf{E}\big[\Gamma^1(T)\big] = \mathbf{E}\big[\Gamma^1(T)\big] - r_{L,i}$). An example of processes with asymmetrical rates is shown in Figure 4.7. Here the cycle time τ_i is varied by a factor of 10.

4.3 The Process

Once we have obtained a set of sub-processes, we combine these to form the process. That will be matched to the measured data. If the match is acceptable, it will be used as a traffic model in a fluid flow analysis.

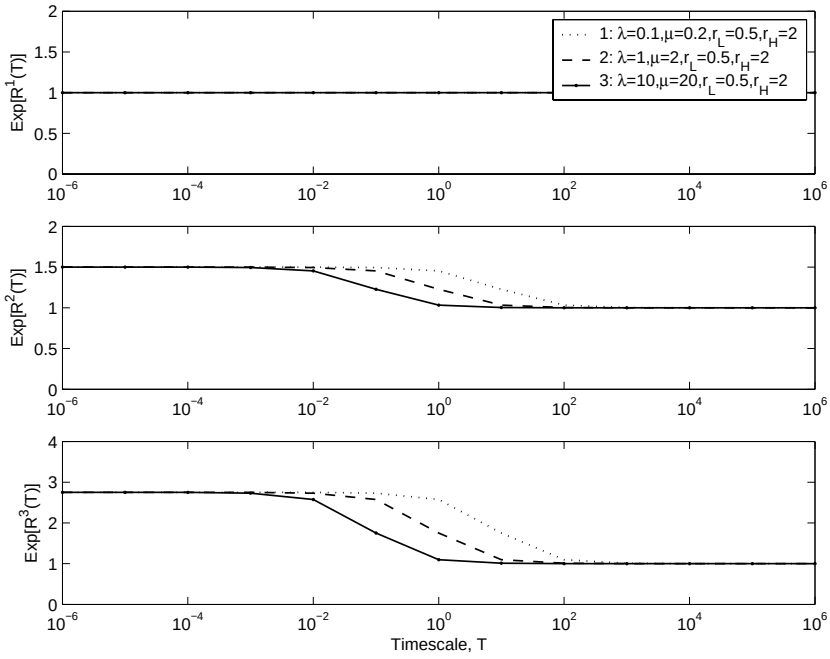

Figure 4.7: Sub-process with asymmetrical transition rates.

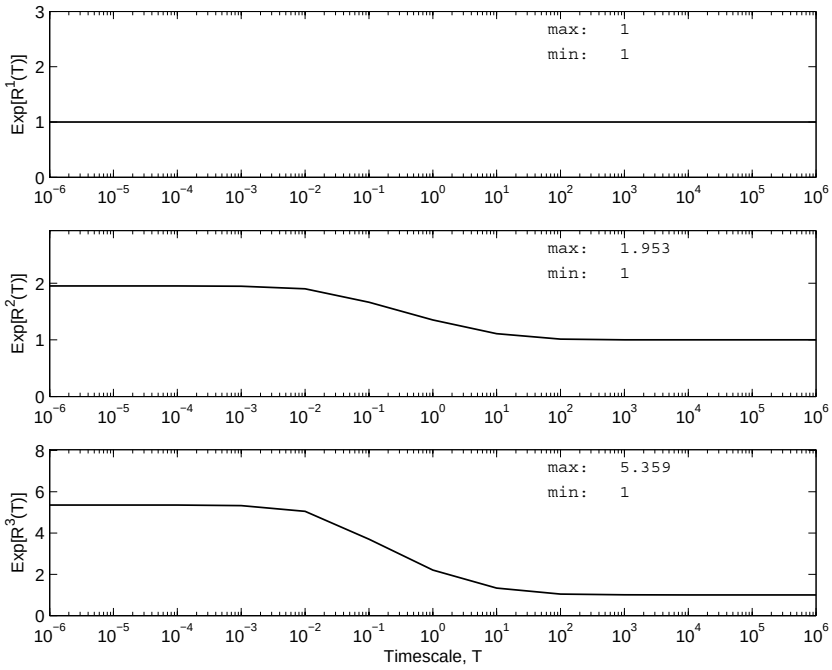

Figure 4.8: An example of a process combined of the three sub-processes shown in Figure 4.5.

4.3.1 Construction

As mentioned before the process is formed by multiplying the output from N sub-processes. The transition matrix is formed by using Kronecker addition:

$$\mathbf{M}_{\mathrm{D}} = \mathbf{M}_1 \oplus \mathbf{M}_2 \oplus \cdots \oplus \mathbf{M}_N \tag{4.20}$$

and the rate matrix as

$$\mathbf{R}_{\mathrm{D}} = \mathbf{R}_1 \odot \mathbf{R}_2 \odot \cdots \odot \mathbf{R}_N \tag{4.21}$$

See Appendix A for a short description of Kronecker algebra and the $\odot$ operator.

4.3.2 Moments

Since the process is formed by multiplication, the moment analysis becomes very simple: We multiply the moments of the independent sub-processes in order to obtain the moments for the process.

$$\mathbf{E}\left[R^1(T)\right] = \prod_{i=1}^{N} \mathbf{E}\left[R_i^1(T)\right] \tag{4.22}$$

$$\mathbf{E}\left[R^2(T)\right] = \prod_{i=1}^{N} \mathbf{E}\left[R_i^2(T)\right] \tag{4.23}$$

$$\mathbf{E}\left[R^3(T)\right] = \prod_{i=1}^{N} \mathbf{E}\left[R_i^3(T)\right] \tag{4.24}$$

An example is shown in Figure 4.8. This process is constructed from three sub-processes shown in Figure 4.5.

4.3.3 Gradient

Unlike the sub-processes that have clearly defined gradients of their higher moments, the process gradient is very different. The shape of a sub-process can be described as a slope; first there is a horizontal region (gradient close to zero), followed by a slope (maximal gradient at the cycle time of the sub-process), and a horizontal region at a lower value than the original. Each sub-process has this shape, and we refer to this as a *slope*. The gradient of the process is the product of the sub-process gradients. Instead of having a clearly defined cycle time, the process has several cycle times, one for each sub-process present. Due to this, the process can exhibit several slopes. If sub-processes share a cycle time, it will emphasize the shape of the process at that timescale. In Figure 4.9 we see three distinct slopes; the shape was created using three sub-processes.

$$\mathbf{P} = \begin{bmatrix} 10^4 & 10^4 & 0 & 2 \\ 1 & 1 & 0 & 2 \\ 10^{-4} & 10^{-4} & 0 & 2 \end{bmatrix}$$

In matrix $\mathbf{P}$ each row specifies a sub-process. The first column contains λ_i, the second μ_i, the third $r_{L,i}$ and the fourth $r_{H,i}$. From the shape of the second and third moment it is clear that these sub-processes' cycle times are present in the process.

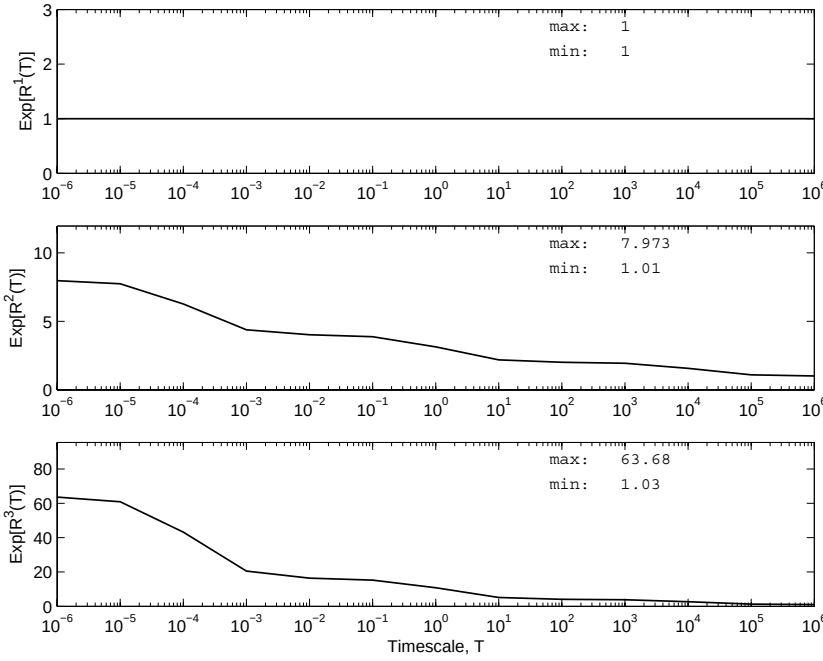

Figure 4.9: Example of a three slope process.

Chapter 5

Process Matching Methods

In this chapter we present five matching methods. These are used to select the sub-process parameters, in such a manner as to minimize the difference between the measured data and the process. One method is a manual method, i.e. the matching is done by manually selecting the parameter values. The second method uses a Genetic Algorithm, the third one uses Simulated Annealing, and the last two ones are heuristic methods. At the end of this chapter, some experience on matching is shared.

To be able to compare and evaluate the different methods w.r.t. their ability to find process parameters we use a data set produced by a known process rather than by measurements. This process consists of one single sub-process with $\lambda = \mu = 2, r_L = 0.75$ and $r_H = 1.25$, see Figure 5.1. This example is used for all methods except the manual, since for the manual one, the parameters are known beforehand.

5.1 Manual Matching

Manual matching can be done in several ways. One possible way is to start by finding a sub-process that matches the largest timescale, then trying to match at the smaller timescales. The choice of the number of sub-processes and their parameters is based on experience and knowledge of the sub-processes properties. This method has the advantages of being very simple and capable of producing models that have constraints other than pure error based.

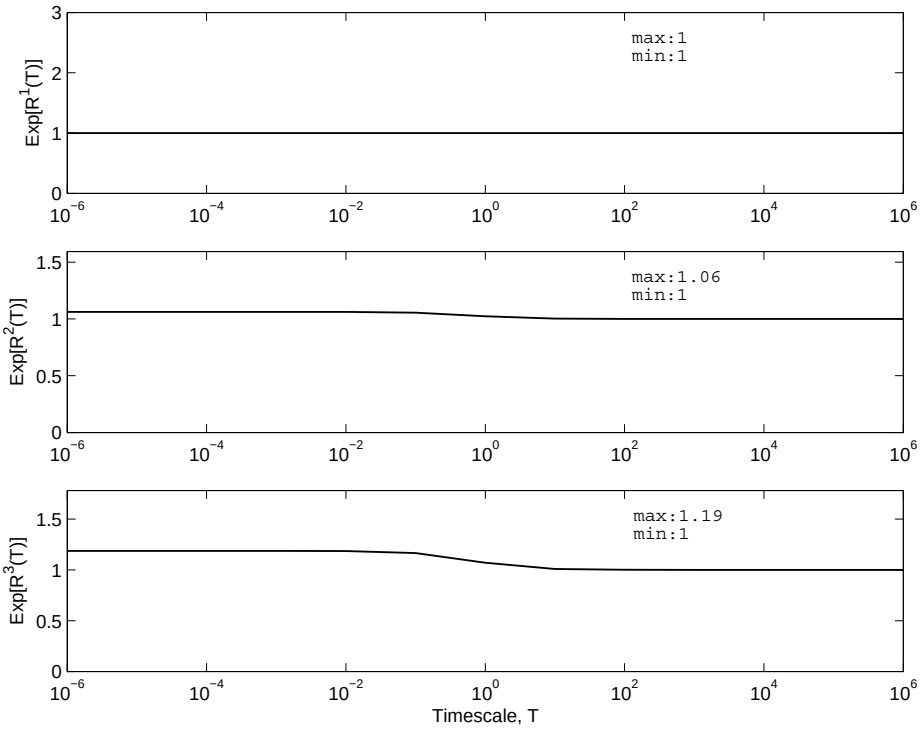

Figure 5.1: Artifical process used for comparison and evaluation of matching methods. The process was constructed from one sub-process with the following parameters: $\lambda = \mu = 2, r_L = 0.75$ and $r_H = 1.25$.

5.2 Genetic Algorithm

We are using a Matlab toolbox called Genetic Algorithms Optimization Toolbox (GAOT) [HJK]. This gives us the benefits of working in the Matlab environment and not having to implement a Genetic Algorithm (GA), however at the cost of processing speed. The two most important input parameters for a GA is the fitness function and the boundary values. In addition to these, there are other parameters that can be modified: crossover function, selection function, mutation function and their corresponding parameters. For a more detailed description of GAOT please see [HJK]. Below we focus on the fitness function and the boundary values are described in 5.2.2.

5.2.1 Fitness Function

To enable a GA to evaluate the fitness of a solution, it needs a fitness function. Our fitness function compares the proposed solutions' moments to the desired moments at the different timescales, where the desired moments are obtained from measurements. We evaluate the fitness value for each moment and timescale individually, and then combine all fitness values to obtain the total fitness of the solution. The fitness of a solution is evaluated as follows:

1. Based on the proposed solutions parameter matrix, $\mathbf{P}$, calculate the moment matrix $\mathbf{S}$ using the analytical expressions for the moments. The $\mathbf{S}$ matrix rows are based on the measured moments at the observed time intervals, $\widehat{\gamma}_{i,k}$ is the i^{th} moment at the k^{th} timescale. $\mathbf{P}$ holds the sub-process parameters with one sub process per row. If the method is allowed to use K sub-processes, and the measured data contains N_m moments and N_T timescales, then the $\mathbf{P}$ and $\mathbf{S}$ matrices look like this:

$$\mathbf{P} = \begin{bmatrix} \widehat{\lambda}_1 & \widehat{\mu}_1 & \widehat{r}_{L,1} & \widehat{r}_{H,1} \\ \widehat{\lambda}_2 & \widehat{\mu}_2 & \widehat{r}_{L,2} & \widehat{r}_{H,2} \\ \cdots & \cdots & \cdots & \cdots \\ \widehat{\lambda}_K & \widehat{\mu}_K & \widehat{r}_{L,K} & \widehat{r}_{H,K} \end{bmatrix} \tag{5.1}$$

$$\mathcal{F}(\mathbf{P}) \rightarrow \mathbf{S} = \begin{bmatrix} \widehat{\gamma}_{1,0} & \widehat{\gamma}_{2,0} & \cdots & \widehat{\gamma}_{n,0} \\ \widehat{\gamma}_{1,1} & \widehat{\gamma}_{2,1} & \cdots & \widehat{\gamma}_{n,1} \\ \cdots & \cdots & \cdots & \cdots \\ \widehat{\gamma}_{1,N_T-1} & \widehat{\gamma}_{2,N_T-1} & \cdots & \widehat{\gamma}_{N_m,N_T-1} \end{bmatrix} \tag{5.2}$$

   A matrix $\mathbf{D}$ containing the measured moments is constructed in the same way as $\mathbf{S}$. $\mathbf{D}_{i,j}$ identifies the desired value on the i:th timescale and j:th

moment, similarly $\mathbf{S}_{i,j}$ identifies the proposed solution.

$$\mathbf{D} = \begin{bmatrix} \gamma_{1,0} & \gamma_{2,0} & \cdots & \gamma_{n,0} \\ \gamma_{1,1} & \gamma_{2,1} & \cdots & \gamma_{n,1} \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_{1,N_T-1} & \gamma_{2,N_T-1} & \cdots & \gamma_{N_m,N_T-1} \end{bmatrix} \tag{5.3}$$

2. Calculate the error matrix as:

$$\mathbf{U} = |\mathbf{D} - \mathbf{S}|$$

From this the fitness for each timescale, i, and moment, j, is obtained as:

$$F_{i,j} = \begin{cases} \frac{1}{U_{i,j}} & U_{i,j} > 1 \\ \Delta \cdot U_{i,j} + a & 10^{-L} < U_{i,j} \leq 1 \\ F_{\max} & U_{i,j} \leq 10^{-L} \end{cases} \tag{5.4}$$

where L is the accuracy requirement or fitness levels. The value 10^{-L} specifies how small the error has to be in order to have a fitness of $F_{\max}$. Δ and a specify a line that goes from $(10^{-L}, F_{\max})$ to $(1, 1)$, simply calculated as

$$\Delta = \frac{F_{\max} - 1}{10^{-L} - 1}$$

$$a = F_{\max} - \Delta \cdot 10^{-L}$$

$F_{\max}$ is the maximum fitness that a moment can obtain in a single timescale, currently set to 100, see Figure 5.2 for a visualization of the shape of the fitness function. The reason not to use a linear error to fitness conversion is the following: When an error goes towards zero the fitness will grow rapidly, particular once the error is smaller than one. Such a value will dominate the total fitness value, i.e. if we for one particular timescale and moment found an almost perfect match with an error $U_{i,j} < 10^{-9}$, the corresponding fitness value will be very large (10^9). Such a fitness value will dominate the total fitness, even if the other fitness values are very small. This is why we have introduced an upper fitness level $F_{\max}$ and not a linear error to fitness relation.

3. Calculate the fitness for the individual moments as the weighted sum of the fitness at the different timescales;

$$\vec{F} = [F_1, F_2, \ldots, F_{N_m}] \quad \text{where} \quad F_i = \sum_{j=0}^{N_T-1} h_j F_{i,j} \tag{5.5}$$

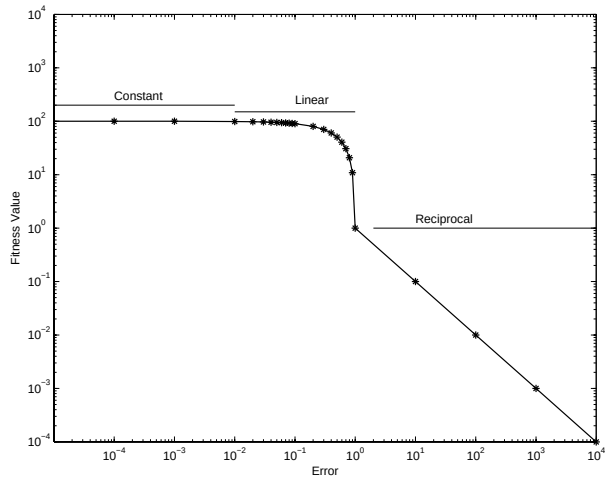

Figure 5.2: Shape of the fitness function, $L = 2$. Please note the logarithmic scales.

The weights h_j enable us to control the influence a particular timescale has on the total fitness of the moment. For instance, the larger timescales are formed from a smaller number of samples than the small timescales. Hence, their values might be less accurate, and should be weighted accordingly when added to the others. $\vec{H}$ contains the weights for the individual timescales, $\sum h_j = 1$.

4. Calculate the total fitness F_t as the weighted sum of the fitness of the moments;

$$F_t = \sum_{j=1}^{N_m} g_j F_j \quad \text{where} \quad \vec{G} = [g_1, g_2, \ldots, g_{N_m}] \tag{5.6}$$

$\vec{G}$ is a vector containing the weights for each moment, $\sum g_j = 1$. This gives us the same capabilities as for the timescale weighting, hence we can prioritze the first moment over the second that has priority over the third.

5. Return the fitness value F_t.

Table 5.1: GA-proposed processes to the artificial data when using default boundary values.

Run	$\hat{\lambda}$	$\hat{\mu}$	$\hat{r_L}$	$\hat{r_H}$	**Fitness**
1	0.766185	831354	0.0783854	999994	61.4481
2	306140	10.6679	7434.89	0.740959	73.302
3	10^{-6}	138531	1.00001	2947.57	97.2541
4	361.187	301867	0	836.769	75.4626
5	0.23073	537726	0.689758	723011	63.6249
6	538485	0.481116	767959	0.313848	61.7334
7	10^{6}	10^{-6}	6167.73	0.999725	96.0312
8	0.63606	670593	0.0514871	10^{6}	60.6358
9	90546.6	819895	1.11077	4.58687×10^{-5}	96.8222
10	10^{-6}	453453	1.00001	4377.47	97.2537

5.2.2 Boundary Values

Ideally the GA should have no boundaries, besides the number of parameters, but allowing a parameter to assume values in the range of $[-\infty, \infty]$ can cause problems. By bounding the parameters we reduce the fitness landscape that the GA has to search, without removing too many possible solutions. The boundary values should be set based on the measured data. In general the transition rates should be in the range of $10^{-6} \leq (\lambda, \mu) \leq 10^{6}$, and the output rates in the range of $0 < (r_L, r_H) \leq 10^{6}$, respectively.

5.2.3 Example

The GA is given the task to match the first three moments using only one subprocess, and the standard boundary values $10^{-6} \leq (\lambda, \mu) \leq 10^{6}$ for the transition rates, and $0 < (r_L, r_H) < 10^{6}$, within 1000 generations, each of 500 individuals. The weights were uniformly distributed over both moments ($g_i = 1/N_m$) and timescales ($h_i = 1/N_T$) and we used five fitness levels ($L = 5$), otherwise using the standard settings. The GA ran ten times, the results are listed in table 5.1. The proposed solution does not have that much in common with the artificial process. Taking in to account that the ranges for the parameters were 10^{12} for the transition rates and 10^{6} for the output rates, the fitness is farily good. The solution with the highest fitness is shown in Figure 5.3. This shows the strength

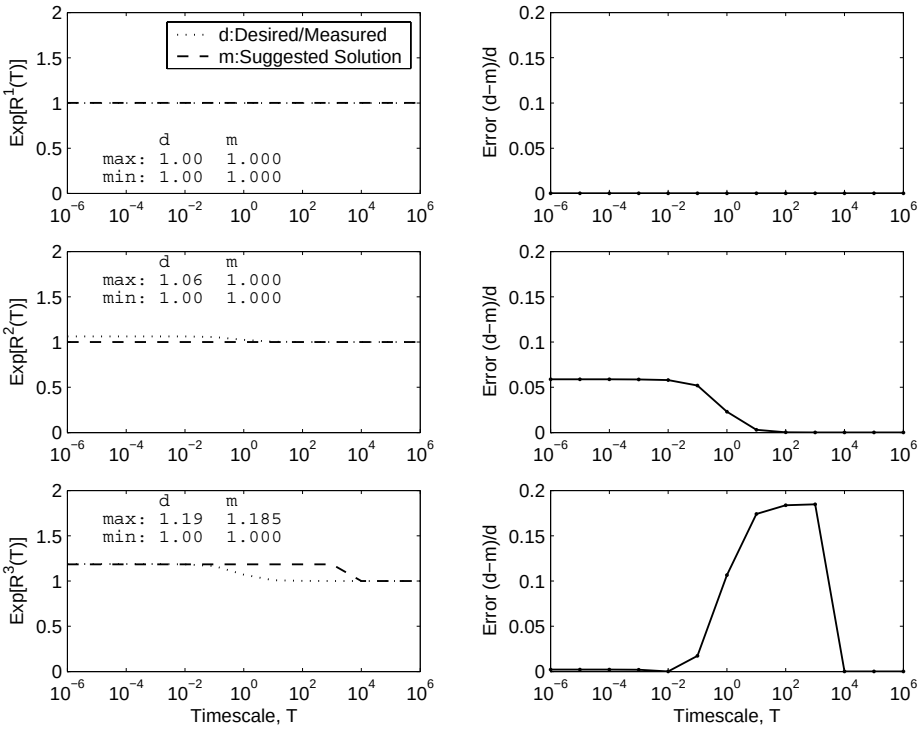

Figure 5.3: Example of a solution (left) and its relative errors (right) proposed by GA (Run 3 in Table 5.1) using default boundaries.

Table 5.2: GA-proposed processes when using tighter boundary values.

Run	$\hat{\lambda}$	$\hat{\mu}$	$\hat{r_L}$	$\hat{r_H}$	**Fitness**
1	32.0597	69.8445	0.833206	1.36344	99.0651
2	19.3102	77.2173	0.879989	1.48	99.0364
3	13.1701	17.9072	1.21724	0.70464	99.4334
4	45.2721	8.31049	0.357747	1.11791	99.2157
5	15.8139	19.4349	1.22749	0.720441	99.4007
6	20.8996	85.5972	1.133	0.455347	99.0553
7	16.9109	83.1436	1.12307	0.394974	99.0609
8	70.015	39.2801	0.657537	1.19215	99.0856
9	60.5431	31.4078	1.34298	0.822098	99.1009
10	81.8059	47.5841	1.32736	0.809616	99.0183

and weakness of the GA. It is possible to find an acceptable solution within a finite number of generations, but a perfect match might be troublesome. If we can find better bounds on the parameters, then we increase the fitness. The transition rates were bound to $10^{-2} \leq (\lambda, \mu) \leq 10^2$ and the output rates to $0 \leq (r_L, r_H) \leq 2$. With these new bounds, we obtained better results especially w.r.t. the higher moments, see Figure 5.4 and Table 5.2. We also explored the possibility of increasing the number of iterations and fitness levels, however the solutions that were obtained did not increase much in quality.

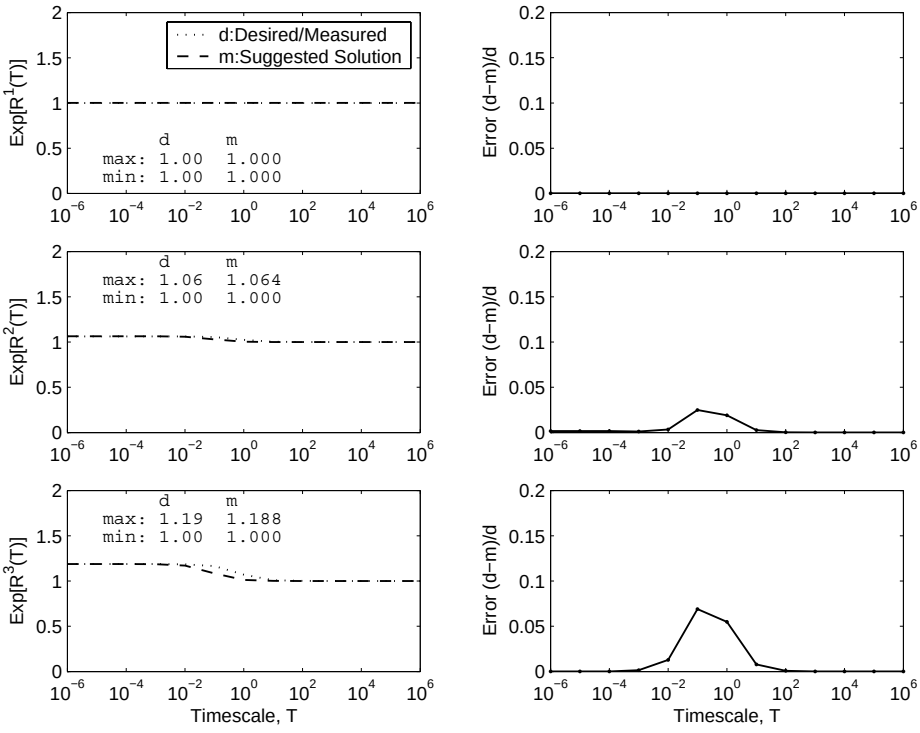

Figure 5.4: Example of a solution (left) and its relative errors (right) proposed by GA (Run 3 in Table 5.2) using tight boundaries.

5.3 Simulated Annealing

We have used Adaptive Simulated Annealing (ASA) [Ing]. ASA is a C-language implementation of simulated annealing. In conjunction with the ASA we used ASAMIN [Sak], which is a Matlab interface. Similar to the GA, it has two major input parameters; the cost function (CF) and the boundaries.

5.3.1 Cost Function

The goal of the SA is to minimize the error, which is done by minimizing the cost function as compared to the GA that tries to maximize the fitness. This enables us to create a cost function that is almost identical to the fitness function that GAOT used. The only difference is that instead of inverting the error in order to obtain the fitness, the error is used directly. We only need to change two equations from the fitness function, Equation 5.5 and 5.6. Thus, the weighted moment error (5.5) becomes

$$\vec{E} = [E_1, E_2, \ldots, E_{N_m}] \quad \text{where} \quad E_i = \sum_{j=0}^{N_T-1} h_j U_{i,j} \tag{5.7}$$

and the total error (5.6) is obtained as

$$E_t = \sum_{j=1}^{N_m} g_j E_j \quad \text{where} \quad \vec{G} = [g_1, g_2, \ldots, g_{N_m}] \tag{5.8}$$

5.3.2 Boundary Values

In order to give the SA the same fitness landscape, it uses the same default boundaries as the GA (see Section 5.2.2).

5.3.3 Example

The SA was given default settings, and the minimization ran for 1000 iterations. To compare the error to the fitness value from the GA, we simply let the proposed solution pass through the GA fitness function. This way we obtain a fitness value for the solution proposed by the SA, which provides us with a common value for comparison. The results the algorithms produced are given in Table 5.3. These solutions are roughly of the same accuracy as the GA. An example is shown in Figure 5.5. We do the same boundary reduction and repeat the matching, the results are shown in Figure 5.6 and Table 5.4, respectively.

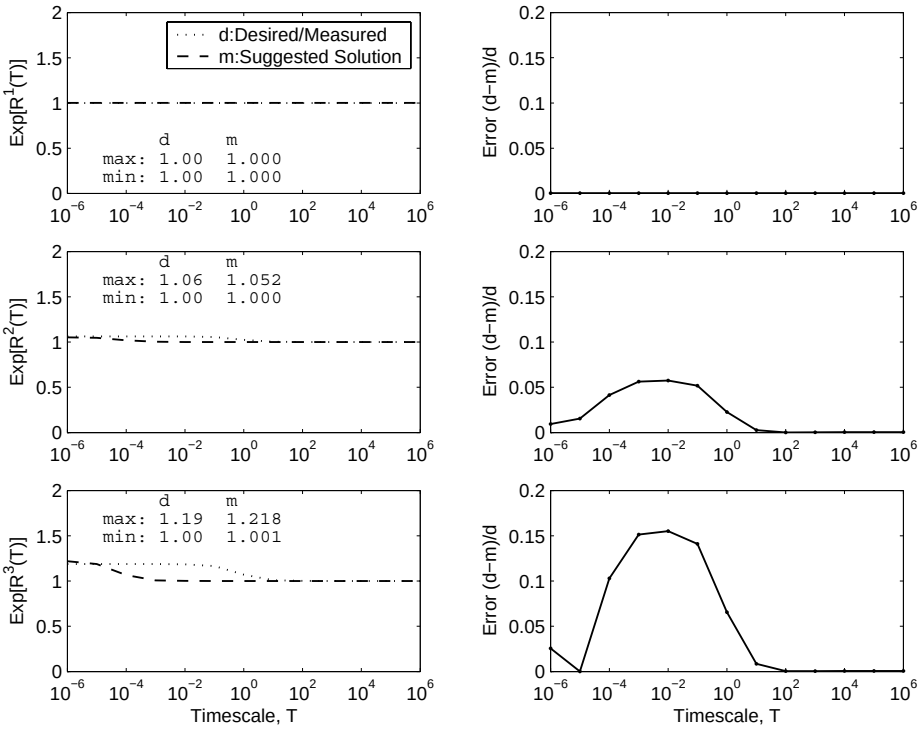

Figure 5.5: Example of a solution (left) and its relative errors (right) proposed by SA (Run 6 in Table 5.3) using default boundaries.

Table 5.3: SA-proposed solutions to the artificial data set, using default bounds.

Run	$\hat{\lambda}$	$\hat{\mu}$	$\hat{r_L}$	$\hat{r_H}$	**Fitness**
1	694225	873832	0.723206	1.34911	96.7738
2	413625	550906	1.25401	0.662323	96.8204
3	936185	335606	0.471071	1.18991	96.79
4	1739.48	249895	0.983902	3.35765	96.887
5	842486	53919.5	2.00518	0.936	96.7671
6	44278.8	1568.93	2.22232	0.956883	97.3674
7	132070	49728.3	1.41883	0.842524	97.0999
8	760129	35303.9	2.14158	0.947306	96.7714
9	393919	518957	1.25338	0.666916	96.8268
10	474909	471294	1.28981	0.713014	96.8231

Table 5.4: SA-suggested solutions when using tighter boundary values.

Run	$\hat{\lambda}$	$\hat{\mu}$	$\hat{r_L}$	$\hat{r_H}$	**Fitness**
1	39.4232	4.74581	1.664	0.920075	99.1857
2	7.75455	1.30142	1.56872	0.904554	99.6307
3	4.69925	20.8548	1.12851	0.429674	99.4212
4	67.1088	75.1583	1.23792	0.733538	99.0219
5	45.08	59.7667	0.782955	1.2878	99.0911
6	11.2371	30.9996	0.854549	1.40125	99.2804
7	46.8677	11.7225	0.462744	1.13438	99.2189
8	24.0269	48.0639	0.826179	1.34775	99.1622
9	84.5552	81.8098	0.744789	1.24693	98.9796
10	2.4913	15.1694	1.11265	0.314097	99.4848

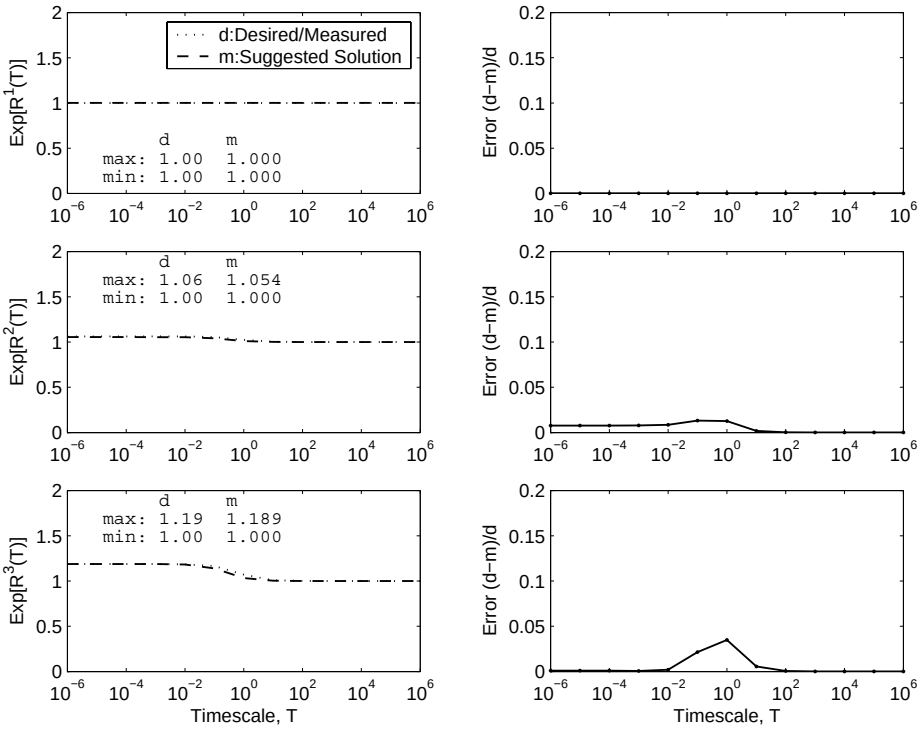

Figure 5.6: Example of a solution (left) and its relative errors (right) proposed by SA (Run 2 in Table 5.4) using default boundaries.

5.4 Heuristic Method 1

The fourth method of matching is a heuristic method called HM1. It utilizes the knowledge of how the sub-process behave, particularly in the limits. The operational principle is to match one sub-process to the tail part of the second moment, then remove the influence that this sub-process has on the desired moments.

The algorithm operates as follows:

1. The measured moments are stored in a matrix $\mathbf{D}$, similar to the SA and GA algorithms, while $\vec{T}$ stores a vector of the timescales. The user supplies a variable that specifies the number of iterations J the algorithm should perform. Further, we define an accuracy vector;

   $$\vec{F_r} = \left[10^0, 10^{-1}, \ldots, 10^{-L}\right]$$

   This is used to increase the accuracy of the process for each iteration. The first iteration will produce a rough estimation, and subsequent iterations will refine this estimation.

2. Initialize the number of unique sub-processes used $l = 0$.

3. Make a copy of the desired moment matrix, $\mathbf{D}^{(l)} = \mathbf{D}$.

4. Repeat for j=1 to J

   (a) Loop through all measured timescales $\vec{T}$, starting at the largest, $i = m, m-1, \ldots, 0$.

      i. Extract moments at timescale i and store them in

      $$\vec{M} = [D_{i,0}^{(l)}, D_{i,1}^{(l)}, \ldots] \quad \text{and let } t = T(i)$$

      ii. Estimate how many sub-processes that are needed to obtain the second moment at this timescale, using (4.18) as a rough estimation.

      $$k = \frac{\log(M_2)}{\log(2)}$$

      If $\max(k) \geq \vec{F_r}(i)$ then this timescale does not need one or more sub-process, go to 4(a). The number of sub-processes needed at this timescale is calculated from

      $$K_l = \max(1, \lceil k \rceil)$$

Where $\lceil \cdot \rceil$ denotes the smallest integer equal or larger than x, i.e. the ceiling operator.

iii. Estimate the parameters of this sub-process(es): Set

$$\begin{aligned} \lambda_l =& \mu_l = \frac{2}{\tau_l} = \frac{2}{T(i)} \\ O =& \sqrt[K]{M_2} \\ B =& |2\sqrt{O-1}| \\ r_{L,l} =& 1 - \frac{B}{2} \\ r_{H,l} =& 1 + \frac{B}{2} \end{aligned}$$

iv. Create a process based on these parameters that spans the same timescales $\vec{T}$,

$$\mathcal{F}(K_l, \lambda_l, \mu_l, r_{L,l}, r_{H,l}, T) \rightarrow \mathbf{S}^{(l)}$$

v. Append the sub-processes parameters to the solution matrix $\mathbf{P}$ (5.1).

vi. Check if the addition of this sub-process influences the total fitness in a positive manner. If $F_{T,\text{old}} > F_{T,\text{new}}$ then remove the influence of this sub-process(es);

$$D_{i,j}^{(l+1)} = \frac{D_{i,j}^{(l)}}{S_{i,j}^{(l)}} \quad \forall i,j$$

Otherwise drop the suggested sub-process(es) from the solution, i.e. remove the entry in the solution matrix $\mathbf{P}$.

(b) Go to 4.

Example

Using the standard example, the algorithm ran for 1, 3 and 5 iterations. The solutions are shown in Table 5.5 and with the fittest plotted in Figure 5.7.

Table 5.5: HM1 suggested solutions.

One Iteration			
$\hat{\lambda}$	$\hat{\mu}$	$\hat{r_L}$	$\hat{r_H}$
2×10^{-6}	2×10^{-6}	1.0	1.0
Fitness 96.0			

Three Iterations			
$\hat{\lambda}$	$\hat{\mu}$	$\hat{r_L}$	$\hat{r_H}$
2×10^{-6}	2×10^{-6}	1	1
2	2	0.84644	1.1536
20	20	0.81693	1.1831
0.2	0.2	0.95843	1.0416
2000	2000	0.94513	1.0549
Fitness 99.565			

Five Iterations			
$\hat{\lambda}$	$\hat{\mu}$	$\hat{r_L}$	$\hat{r_H}$
2×10^{-6}	2×10^{-6}	1.0	1.0
2	2	0.84644	1.1536
20	20	0.81693	1.1831
0.2	0.2	0.95843	1.0416
2000	2000	0.94513	1.0549
0.02	0.02	0.99035	1.0096
Fitness 99.567			

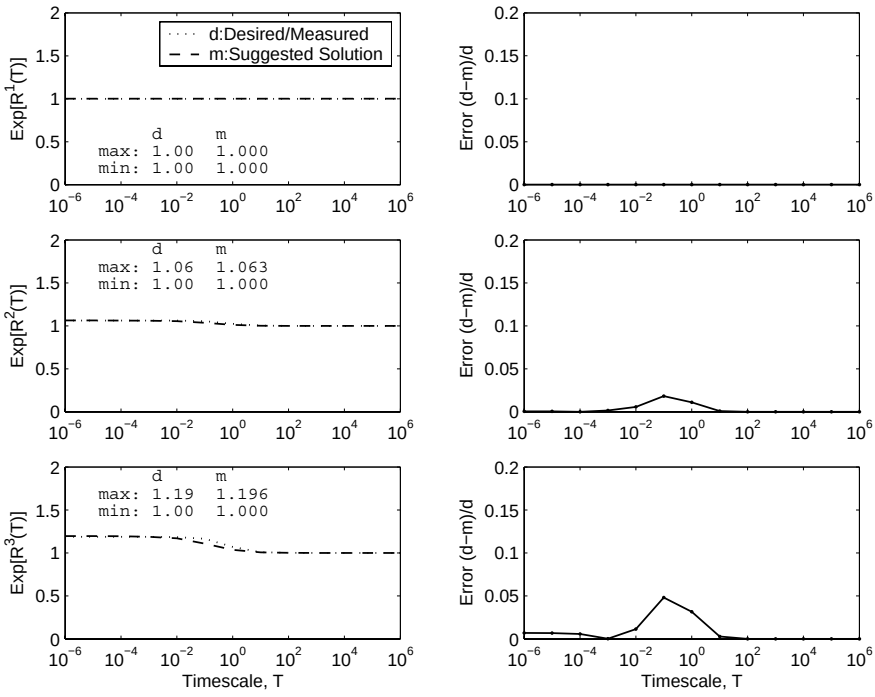

Figure 5.7: The result obtained by the HM1 method when using five iterations.

5.5 Heuristic Method 2

This method, called HM2, operates best if the measured data shows a linear region for the second moment as $T \to 0$ and $T \to \infty$, and if there is only one slope present. If so, it is possible to estimate how many sub-processes that are needed. From this, we can calculate the output rates, and the only uncertainty is τ, which we can get a fairly good estimate on. In order to improve the solution, a SA could be applied to find a better estimation of the transition rates.

1. Estimate the number of processes;

$$K = \left\lceil \frac{\log(D_{1,2})}{\log(2)} \right\rceil \tag{5.9}$$

   Using this information we only need to find the parameters of one sub-process since all will be assumed to be identical.

2. Estimate the output rates for one sub-process

$$\Delta = \left| \sqrt{\sqrt[K]{D_{1,2}} - D_{m,2}} \right|$$

$$\begin{aligned} r_L &= 1 - \Delta \\ r_H &= 1 + \Delta \end{aligned}$$

3. Estimate the cycle time τ.

$$\nabla_i = \frac{D_{i+1,2} - D_{i,2}}{T_{i+1} - T_i} \quad \forall i = 1, \ldots, N_{T-1} \tag{5.10}$$

   Then the maximum gradient can give a rough estimate of where the cycle time is located.

$$\tau = T_i \Big|_{i:\nabla_i = \text{maximal}} \tag{5.11}$$

4. Calculate the error matrix $\mathbf{U}$, and from it obtain the total error ε of the solution (using the SA cost function).

5. If $\varepsilon > 10^{-L}$, then use SA to find a better match. Give the SA tighter bounds than what would normally be used. Use the gradient estimation to find the span of the transition rates, let the transition bounds be slightly larger than τ, $(0.01\tau \leq (\lambda, \mu) \leq 100\tau)$ and fix the output rates to the suggested solution.

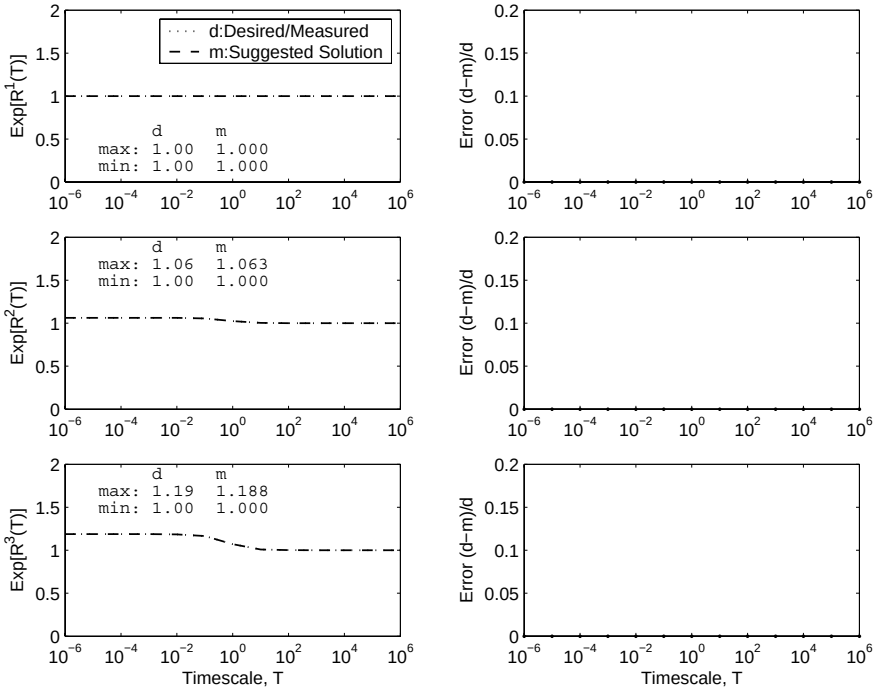

Figure 5.8: HM2 matching example.

Example

This method manages to find a perfect match, $\lambda = \mu = 2, r_L = 0.75$ and $r_H = 1.25$, for the artificial process. This is partially due to the fact that $\tau = 1/\lambda + 1/\mu = 1$, and this is one of the timescales that are present. If τ would have been shifted a bit, i.e. $\tau = 2 \rightarrow \lambda = \mu = 1$, then the match would not have been perfect, but would have required the use of the SA. With this alteration, the solution was found by the algorithm after applying the SA, ending up with a perfect match $\lambda = 1, \mu = 1, r_L = 0.75$ and $r_H = 1.25$. This was obtained using 1000 iterations. An illustration of the results is found in Figure 5.8.

5.6 Matching in Practice

The five methods that we have presented above are only a subset of all possible matching methods. Which method should be used? It is highly dependent on the data to be matched. For instance, if the data exhibits a linear portion as $T \to 0$ and $T \to \infty$ then the second heuristic method should be considered. Furthermore, by altering the algorithms' operational parameters better matches can be obtained. The methods described above should only be viewed as templates. In Chapter 6 we will give some examples for matching real data network traffic.

5.6.1 Number of Timescales

Once a data set has been obtained through measurments, the first issue to settle is to determine which timescales should be used by the model. By minimizing the number of timescales, the matching is made simpler from an optimization point of view. However, in this work we have not looked into this, thus it is left for further study.

5.6.2 Number of Sub-Processes

Both the GA and SA algorithms require that the user supplies the number of sub-processes that they are allowed to work with, hence an estimation of this value is desirable. The two main components that affect the number of sub-processes needed are the values of higher moments on the smallest timescale and the number of slopes for the same moments. A crude estimate was used in both heuristic methods:

$$K = \left\lceil \frac{\log(D_{1,2})}{\log(2)} \right\rceil \tag{5.12}$$

This estimation is based on that the second moment (4.18) approaches 2 as $T \to 0$, when $r_L = 0, r_H = 2$ and requiring $\mathbf{E}\left[R^1(T)\right] = 1$. This estimate will allow a solution to reach the desired value on the smallest timescale; it will not necessarily allow it to track all slopes. Each such slope requires at least one sub-process. In the example shown in Figure 5.1, there is only one slope present.

So, how to estimate the number of sub-processes? One alternative would be to use the crude estimation (5.12), count the number of slopes, add both numbers and use this as the number of sub-processes. However, for each sub-process we add, we quadruple ($\times 4$) the number of variables that need to be determined. When increasing the number of variables, we usually need to increase the number of iterations or generations as well. We also need to have the fluid

flow environment in mind, for which the state space grows with a factor 2 for each sub-process added. Hence a model with K sub-processes needs two $2^K \times 2^K$ matrices. The upper limit imposed on K is by the amount of memory present in the computer doing the analysis. Another issue is the time consumption for the fluid analysis. According to our experience K should be less than 10.

5.6.3 Fitness

What does, for instance, a total fitness of 80 really mean? Let us assume that we have three fitness levels, $L = 3$, and equal weights across all timescales and moments. Each moment supplies an equal amount of the total fitness, the maximum fitness that one moment can supply to the total fitness is $F_{\max}/N_m$. With $F_{\max} = 100$ and $N_m = 3$ a total fitness value of 80 can mean that two moments are almost perfect matches, and one has a moment fitness F_m of 14, or any combination in between. A moment fitness of 14 equals a timescale fitness of $F_T = F_m \cdot N_m = 42$.

$$80 = 33 + 33 + 14 = \frac{99}{3} + \frac{99}{3} + \frac{42}{3}$$

This in turn means that at least one of the timescales has an error smaller than one. If not, the timescale fitness would be less than one. Each timescale can have a fitness of $F_{\max}$, where a fitness of $F_{\max}$ means that the error in that timescale is less or equal to 10^{-L}. In order to obtain a fitness value larger than one on a certain timescale, the error has to be smaller than one according to (5.4). Each timescale can provide the timescale fitness with $F_{\max}/N_T$ fitness points. Thus, when $N_T = 10$, a $F_m = 14$ can imply four really good timescales and one with an error slightly less than one, and the remaining five have errors significantly larger than one.

$$42 \approx 10 + 10 + 10 + 10 + 2 + \varepsilon + \varepsilon + \varepsilon + \varepsilon + \varepsilon = 4\frac{100}{10} + 1\frac{20}{10} + 5\frac{\varepsilon}{10} \quad \varepsilon << 1$$

Impact of the Fitness Weights

In this thesis all results were obtained using uniformly distributed fitness weights. This is however not always the desired case. For instance, a possible scenario is that there are perfect matches on the second and third moment, but the first moment can be very wrong. To avoid situations like this we should not use uniform weights when summing the different moments' fitness. Instead, the weights

should identify which moment we like to match. One set of weights that emphasises the lower moments is

$$\vec{G} = [N_m/\bar{g}, (N_m - 1)/\bar{g}, \cdots, 1/\bar{g}] \quad \bar{g} = \frac{N_m + 1}{2}$$

A similar weighting can, and should, be done for the timescales, especially since the higher timescales are based on fewer samples. An example could be to have a staircase weight, where the smaller timescales are more important than the highest.

$$\vec{H} = \left[N_T/\bar{h}, (N_T - 1)/\bar{h}, \cdots, 1/\bar{h}\right] \quad \bar{h} = \frac{N_T + 1}{2}$$

However, in this work we will not use any of these capabilities, since we do not wish to give preferential treatment to any timescale or moment. How to select these fitness weights are left for future work.

5.6.4 Matching Problems

If we disregard the constraints that a fluid analysis incurs, there are still cases when the model will have problems, for instance when the measured data exhibits certain behaviour. If N_T is sufficiently large, so that the last slope has been passed and

$$\left(\Gamma^3_{N_T}\right)^{2/3} \approx \Gamma^2_{N_T} \tag{5.13}$$

does not hold, the model will have problems matching all three moments.

Chapter 6

Results and Discussions

In this chapter we model four different network links; one ADSL link, one Internet access link, and two of the Bellcore traces.

6.1 Bellcore Traces

Out of the four Bellcore traces [BCT], we have selected the BCpOct89 and BCOct89Ext4 traces. Both traces were collected at the Bellcore Morristown Research and Engineering facility. The BCpOct89 captured primarily LAN traffic, and BCOct89Ext4 collected WAN traffic.

6.1.1 Measurements

The BCpOct89 trace started at 11:00 October 5 1989 EDT and ended approximately 30 minutes later. The BCOct89Ext4 comes from a 307-hour long trace that begun on 14:37 October 10, 1989. The BCOct89Ext4 trace started about 215 hours into this 307 hour trace, and ended about 21 hours later. The timestamp accuracy is about 10 μs. Using a base scale of 1 ms, we get approximately 1% error in our bit rate estimations at the base scale, see Table 3.1 and page 34. If we had a base scale of 100 μs the error in the bit rate estimation could be as high as 10%. We use $S = 2$ as scaling factor. The first three moments for the BCpOct89 trace are shown in Figure 6.1, and similarily for the BCOct89Ext4 trace in Figure 6.2. From the figures we see that the BCpOct89 trace is quite well-behaved. At the highest timescale for instance, its second moment is almost

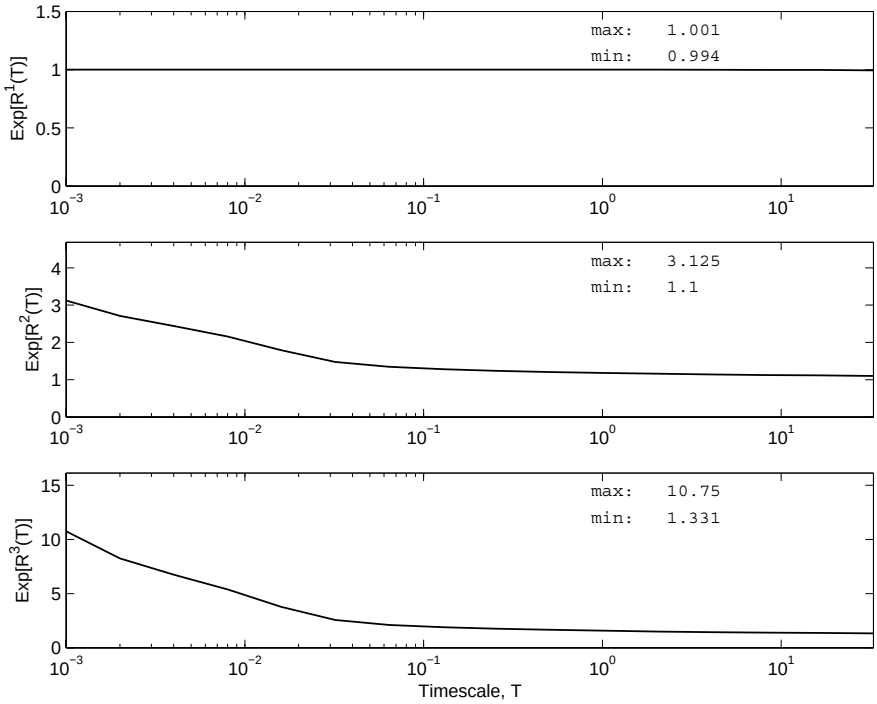

Figure 6.1: Bellcore pOCt89 first three moments.

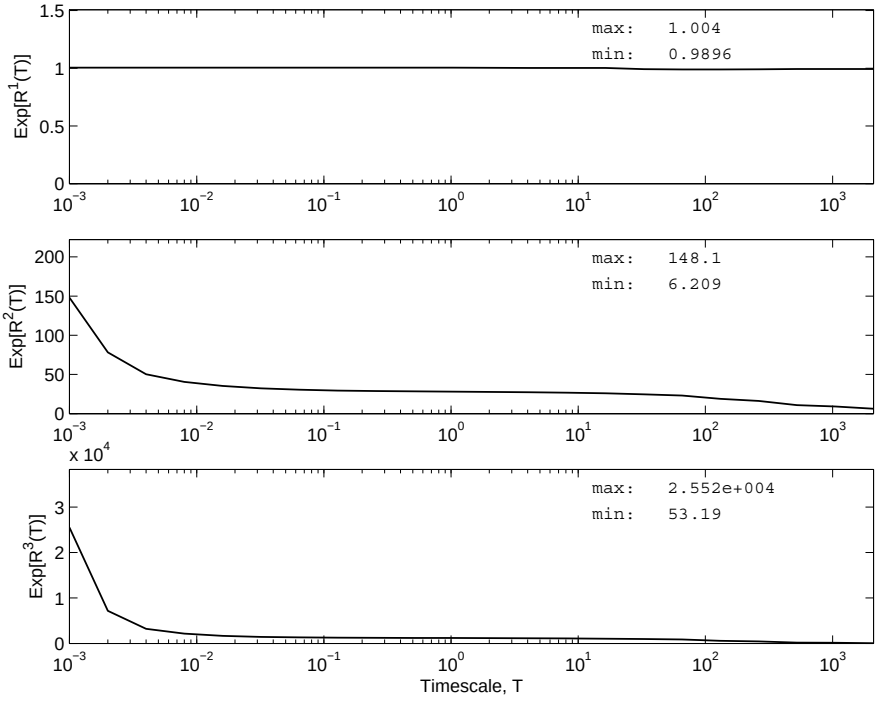

Figure 6.2: Bellcore OCt89Ext4 first three moments

Table 6.1: BCpOct89 Model Fitness values. The best solution is shown in Figure 6.3.

Method	Fitness	No. of sub-processes
GA	77.2270	2
SA	87.5855	2
HM1	88.159	6
HM2	88.7681	2
Manual	88.04	3
GA	67.6901	4
SA	88.0053	4
GA	67.6907	6
SA	87.5779	6

equal to the square of the first moment, a behaviour that we recognize from the process description in Chapter 4. The BCOct89Ext4 trace does not display this nice behaviour, in fact at the largest timescale the second moment is approximately six times higher than the squared first moment. This indicates that the model will have problems matching all three moments. Furthermore, the gradient on the two higher moments is significantly larger for the BCpOct89 trace. Altogether, this indicates that the BCOct89Ext4 is not a so well-behaved trace. The reasons for this could be that a process with a large timescale is present. As the trace does not contain more data, we unfortunately cannot account for this.

6.1.2 Matching

The GA and SA methods require us to supply a number of sub-processes that they are allowed to use. Using the guidelines from the previous chapter our initial estimation for the number of required sub-processes for the two traces became 2 (BCpOct89) and 8 (BCOct89Ext4). We used the default bounds, $10^{-6} \leq (\lambda, \mu) \leq 10^{6}$ and $0 \leq (r_L, r_H) \leq 2$, three fitness levels $L = 3$ and 1000 generations each of 500 individuals for the GA, while the SA ran for 1000 iterations.

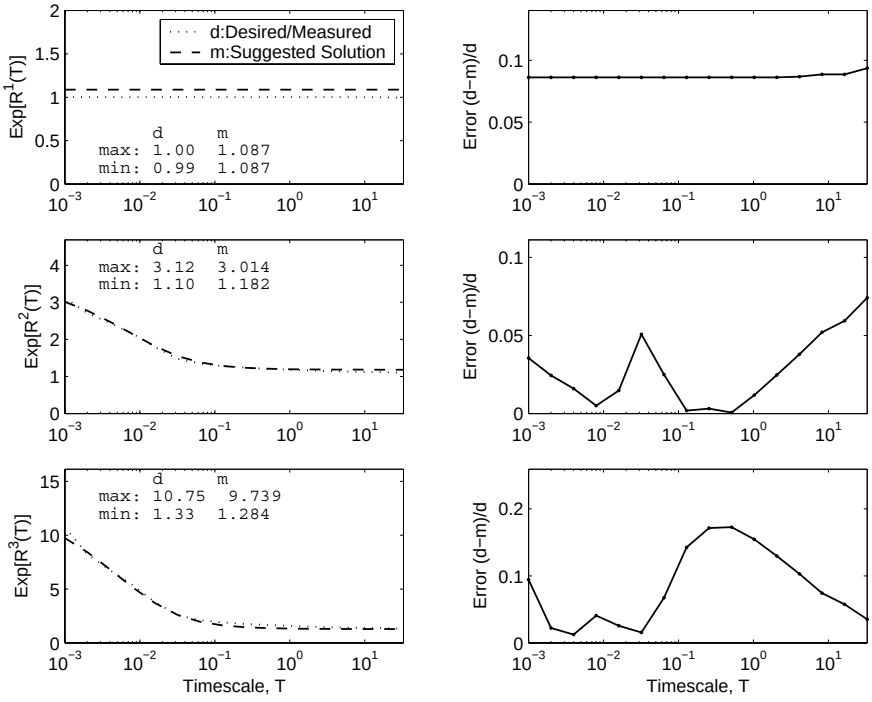

Figure 6.3: BCpOct89 Model with a fitness of 88.7681 obtained by the HM2 algorithm.

Table 6.2: BCOct89Ext4 Model Fitness values. The best solution is shown in Figure 6.4.

Model	Fitness	No. of sub-processes
GA	33.2364	8
SA	0.0474	8
HM1	34.3271	16
HM2	0.4973	8
Manual	33.20	9
GA	33.2363	10
SA	0.0474	10
GA	33.2363	16
SA	0.0568	16

BCpOct89

The fittest solution was obtained by the HM2 algorithm, see Table 6.1 and Figure 6.3, with a fitness of 88.77, using only two sub-processes:

$$\mathbf{P} = \begin{bmatrix} 81.684 & 143.13 & 0.12378 & 1.8762 \\ 999.84 & 342.26 & 0.12378 & 1.8762 \end{bmatrix}$$

This is very good compared to the HM1 and GA models. Even though the HM1 model obtained a high fitness, it required six sub-processes to do this. The GA managed to obtain a fitness of 77.2 when using two sub-processes, in comparison to SA that yielded 87.6. Since both the GA and SA were supplied with the number of sub-processes to use, we performed additional runs where we increased the number of sub-processes. The SA increased its fitness slightly, but the GA decreased it. The manual method obtained a fitness of 88.04 when using three sub-processes.

BCOct89Ext4

In Table 6.2 and Figure 6.4 the results are shown. The best match was obtained by the HM1 algorithm. When using 16 sub-processes it obtained a fitness of 34. The GA managed to find a fitness of 33 using only 8 sub-processes, which has a better fitness-per-sub-process value. Neither HM2 nor SA managed to obtain a fitness over one.

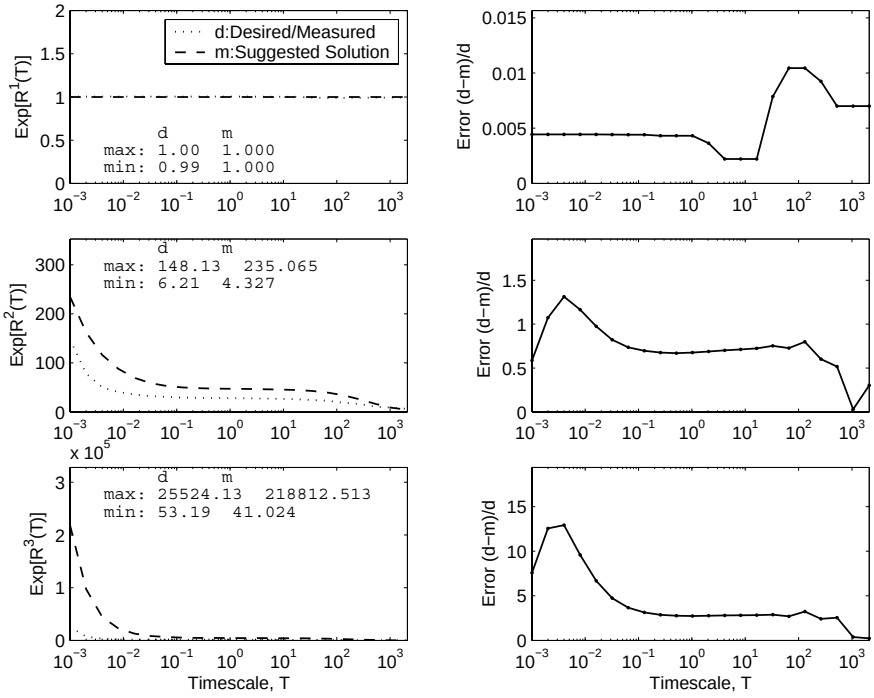

Figure 6.4: BCOct89Ext4 Model with a fitness of 34.3271 obtained by the HM1 method.

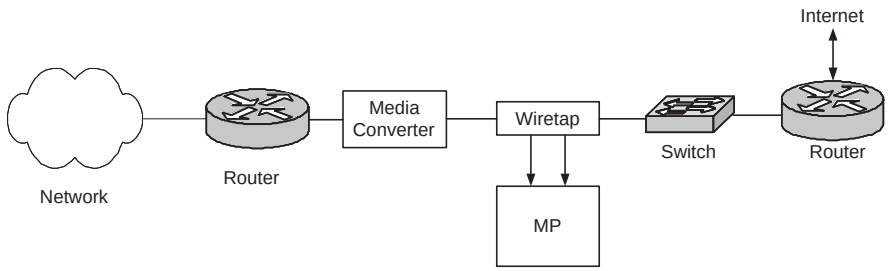

Figure 6.5: Measurement setup for the IAL.

One reason for this bad performance on behalf of the the SA method, is that it might have too many variables to control and too few iterations to evaluate all combinations. In fact, having eight sub-processes yields 32 variables, and 12 sub-processes 48 variables, hence using only 1000 iterations might be too few. By reducing the number of sub-processes to two:

$$\mathbf{P} = \begin{bmatrix} 199140 & 519530 & 0.19192 & 1.348 \\ 945210 & 22000 & 0.39923 & 1.9949 \end{bmatrix}$$

The GA algorithm managed to obtain the same fitness (33.2363) as when using eight. The SA algorithm obtained a fitness of 1.5 when using only two sub-processes.

6.2 Internet Access Link

This link gives Internet access to a student dormitory network that connects approximately 300 users. The link is a 100Base-T, and interconnects a media converter (100Base-X to 100Base-T) with a 100Base-T switch port. We call this link IAL. See Figure 6.5 for a view of the connection and measurement setup. The link operates at full-duplex.

6.2.1 Measurements

We performed several measurements at this location, but here we present the measurement we called IAL-1 since it exhibits an interesting shape. The IAL-1 started on 17:10:49 September 17, 2003 and it ran for one hour. For the

Table 6.3: IAL-1 Traffic Statistics.

Link Statistics		
	IAL-1-NI	IAL-1-IN
Protocol	No. of frames	No. of frames
IP	1.61×10^7	1.89×10^7
ARP	87	325
Other	–	1858
Network Statistics		
Protocol	No. of frames	No. of frames
ICMP	112381	60336
IGMP	–	60
TCP	1.01×10^7	1.08×10^7
UDP	2.32×10^6	2.86×10^6
Other	3.50×10^6	5.17×10^6

measurement we used a DAG3.5E card, which enables us to have a base scale of 10 μs. $S = 2$ was used for scale factor. Since the link is a full-duplex link we obtained two traces; IAL-1-IN and IAL-1-NI; I stand for Internet and N for Network. Hence, IN identifies traffic going from Internet to the Network. In Table 6.3 a summary of the protocol usage is shown. In Figure 6.6 the first three moments for the IAL-1-NI trace are shown. Based on this, we see that the second moment seems to be well behaved, in the sense that at the largest timescale the moment is almost equal to the squared first moment, similarly for the third moment. Turning our attention to the smallest timescale, we notice a value of approximately 3, indicating that two sub-processes are needed to reach it. Furthermore, we see that the shape seems to be leveling off which is nice. In the IAL-1-IN data, shown Figure 6.7, this flat region is not that obvious, but it shows some tendencies to level off. With a peak value of 7 at the smallest timescale, we will need three sub-processes. For the higher timescales, the higher moments are slightly smaller than the corresponding powers of the first moment. But in general, both traces seem to be well-behaved and display a shape similar to the shape of the matching sub-processes.

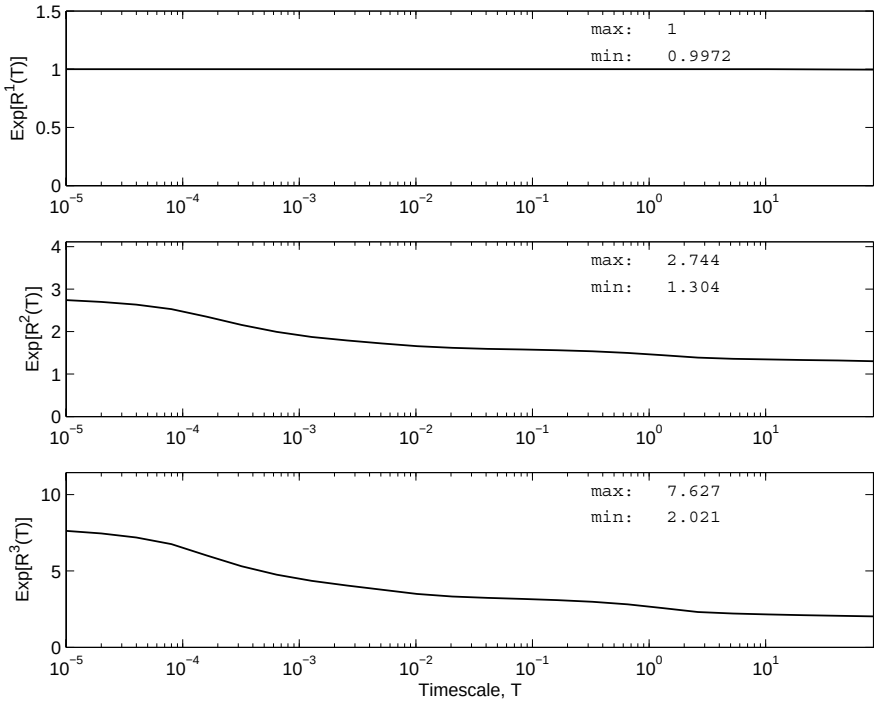

Figure 6.6: First three moments for the IAL-1-IN trace, which contains traffic from the Internet to the Network.

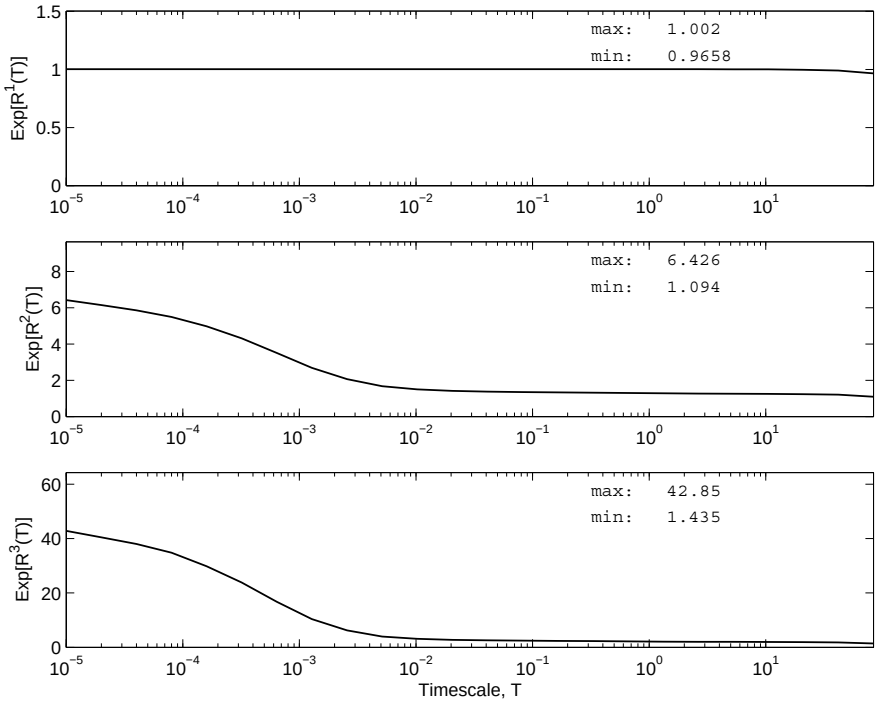

Figure 6.7: First three moments for the IAL-1-NI trace, which contains traffic from the Network to the Internet.

Table 6.4: IAL-1-IN Model fitness values. The best solution is shown in Figure 6.8.

Model	Fitness	No. of sub-processes
GA	67.0208	2
SA	64.1735	2
HM1	78.0643	6
HM2	62.8426	2
Manual	86.9	4
GA	60.8110	4
SA	76.7816	4
GA	74.7516	6
SA	75.4948	6

6.2.2 Matching

The matching was done using default boundaries, number of generations, iterations and population size. The number of sub-prcesses was determined to be two for the IAL-1-IN, and three for the IAL-1-NI trace. In the following section we describe the results of the matching.

IAL-1-IN

The best solution was obtained by the manual method; using four sub-processes, it reached a fitness of 89:

$$\mathbf{P} = \begin{bmatrix} 10^{-6} & 10^{-6} & 0.45 & 1.55 \\ 1 & 1 & 0.5 & 1.5 \\ 1000 & 1000 & 0.4 & 1.6 \\ 10000 & 10000 & 0.5 & 1.5 \end{bmatrix}$$

The second best fit was obtained by the HM1; using six sub-processes, it reached a fitness of 78. The SA algorithm reached a fitness of 76.8 using only four sub-processes, see Table 6.4. Figure 6.8 shows the fittest solution.

IAL-1-NI

The HM1 obtains a fitness of 72.7, using nine sub-processes. Here the GA reaches a fitness of 68.7; the HM2 reaches a fitness of 67.6. And finally the SA performs

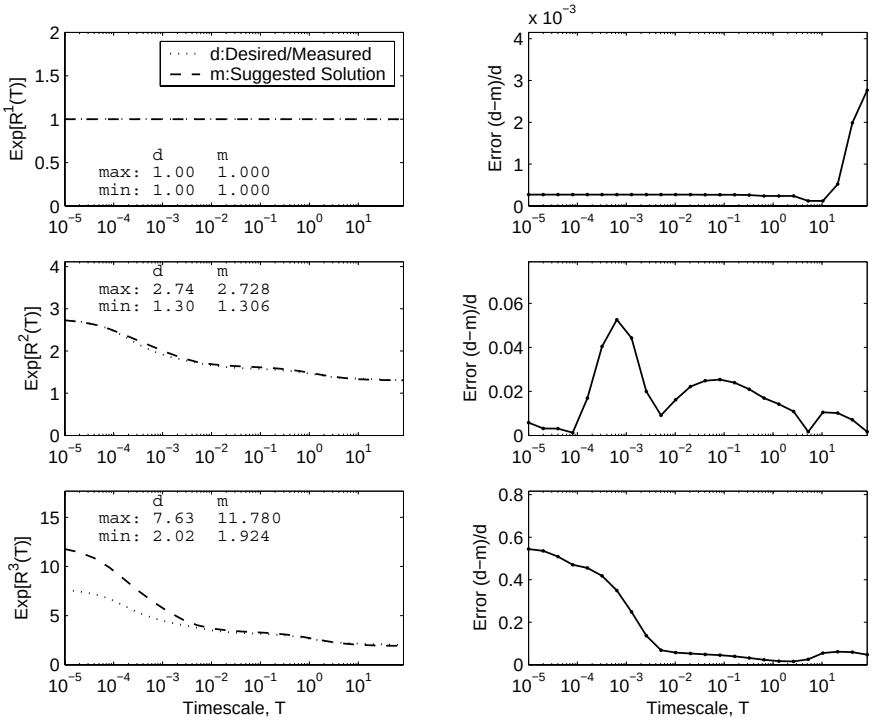

Figure 6.8: IAL-1-IN solution with a fitness of 86.9 obtained manually.

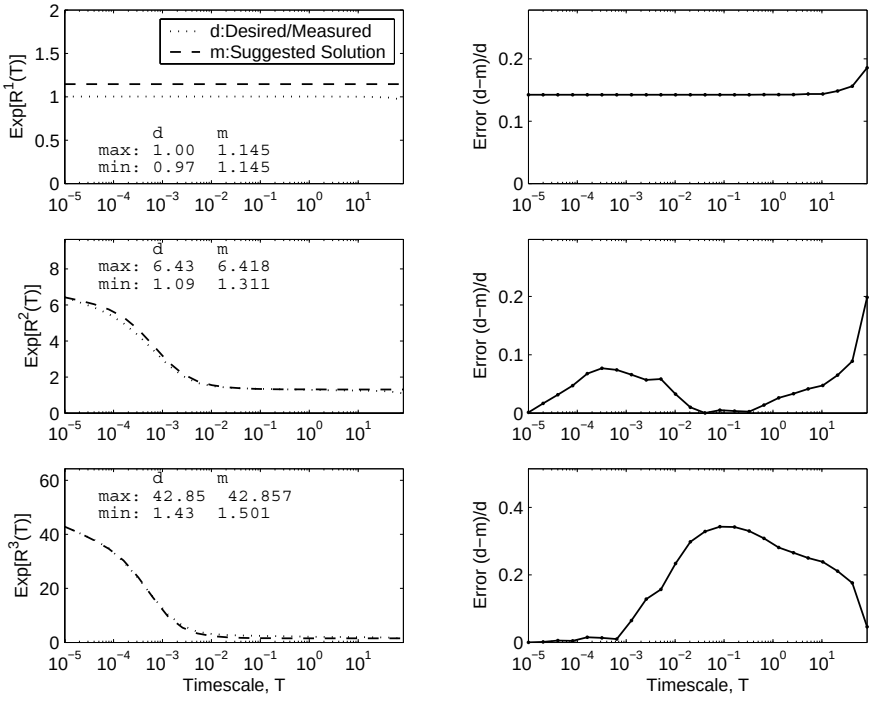

Figure 6.9: IAL-1-NI Model, this solution was obtained by SA with a fitness of 75.5058.

Table 6.5: IAL-1-NI Model fitness values. The best solution is shown in Figure 6.9.

Model	Fitness	No. of sub-processes
GA	68.6688	3
SA	75.5058	3
HM1	72.7274	9
HM2	67.598	3
Manual	65.2	7
GA	68.7700	5
SA	75.5328	5
GA	69.2943	7
SA	73.0489	7

really good, and using only three sub-processes it reaches a fitness of 75.5:

$$\mathbf{P} = \begin{bmatrix} 626.97 & 3300.2 & 0.057974 & 1.9379 \\ 44350 & 154590 & 1.9064 & 0.83501 \\ 11327 & 145350 & 1.9904 & 0.98145 \end{bmatrix}$$

The results are shown in Table 6.5 and the fittest solution is plotted in Figure 6.9. The manual solution managed 65.2 with seven sub-processes.

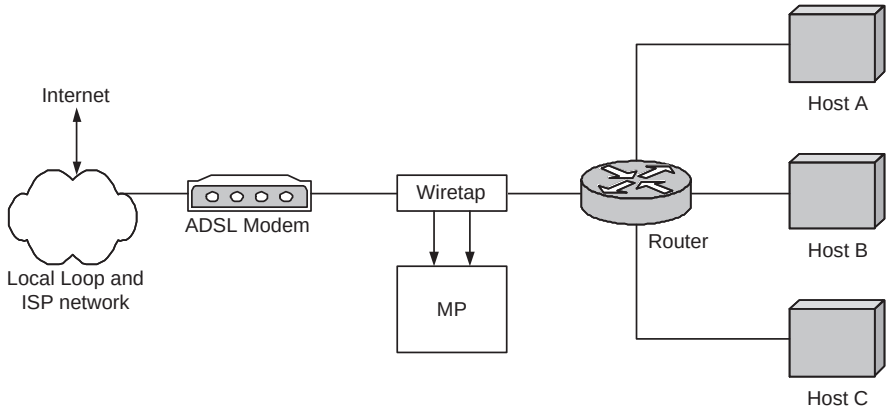

Figure 6.10: Measurement setup for the ADSL scenario.

6.3 ADSL Link

This link is located in an apartment, and connects a broadband router to an Asymmetric Digital Subscriber Line, ADSL, modem using 10Base-T/100Base-T. The router has four switch ports and a wireless access point (IEEE 803.11b). In normal operations, there are between two and three hosts attached to the switch and no wireless hosts, see Figure 6.10 for network map.

6.3.1 Measurements

We ran one measurement on the link, spanning eight hours. The measurement called ADSL8, ran from 09:30 to 17:30 September 14, 2003. During this period two hosts were active, one of which received streaming audio for the majority of the measurement. A summary of the traffic observed is given in table 6.6. Here M stands for Modem and R for router, so MR means traffic going from the modem to the router.

The measurement was done using a DAG3.5E card, hence our base scale was set to 10 μs, and we used a scale factor of $S = 2$. In Figure 6.11 and Figure 6.12 the different directions for the ADSL8 measurement are shown.

When looking on ADSL8MR (Figure 6.11) we notice that the first moment increases slightly instead of decreasing as both the Bellcore and IAL traces did. The increase is however very small. But turning our attention to the second

Table 6.6: ASDL Traffic Statistics

Link Statistics		
	ADSL8MR	ADSL8RM
Protocol	No. of frames	No. of frames
IP	13268	11374
ARP	30	6
Network Statistics		
Protocol	No. of frames	No. of frames
ICMP	2073	1493
TCP	11079	9691
UDP	116	190

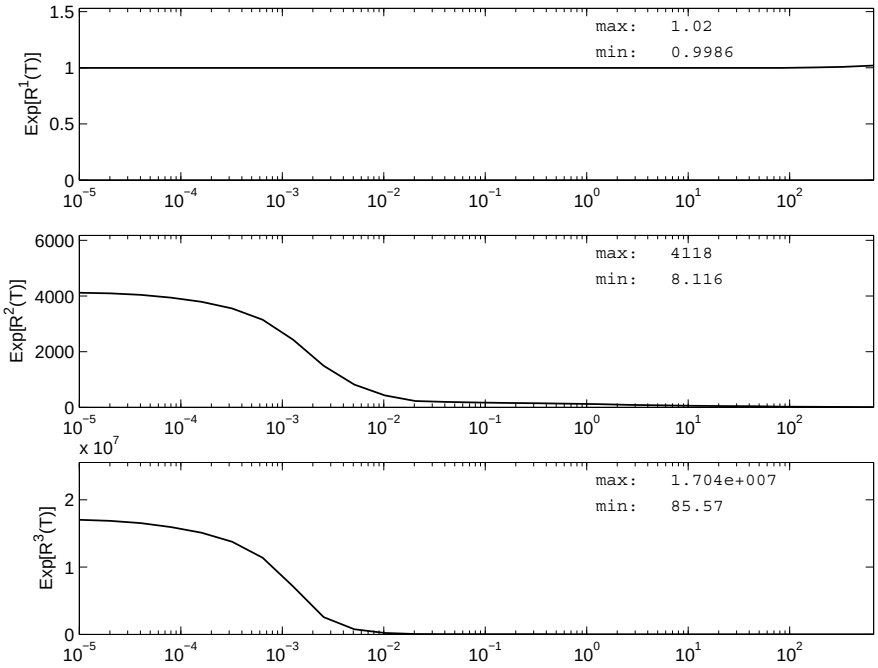

Figure 6.11: ADSL8RM trace, the first three moments containing router to modem traffic.

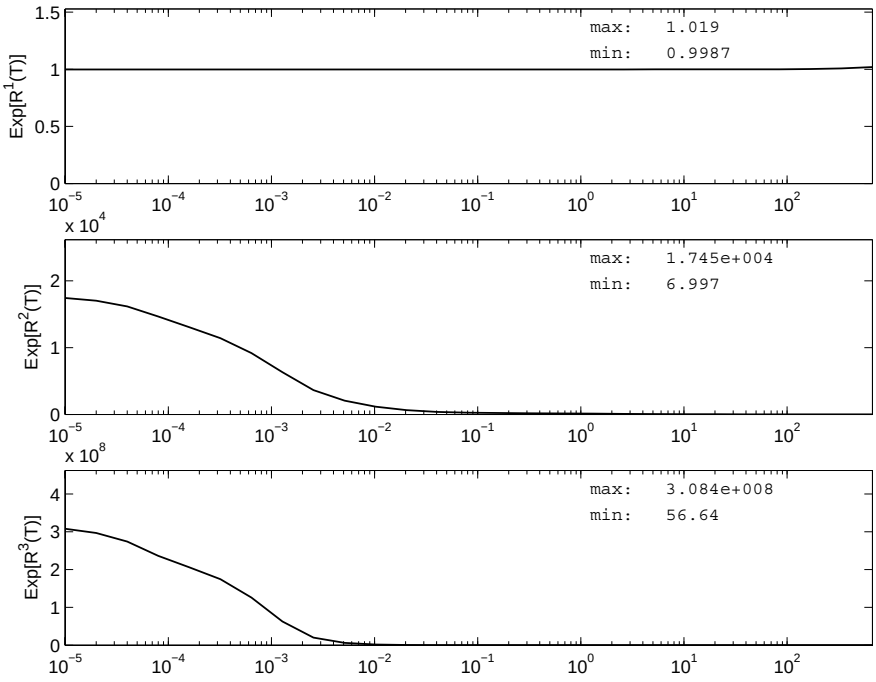

Figure 6.12: ADSL8MR trace, the first three moments containing modem to router traffic.

Table 6.7: ADSL8RM Model fitness values. The best solution is shown in Figure 6.13.

Model	Fitness	No. of sub-processes
GA	0.375113	12
SA	0.470198	12
HM1	34.1548	18
HM2	26.0981	13
Manual	34.4	12
GA	0.375113	14
SA	1.07734	14
GA	0.375113	16
SA	27.1533	16

moments, at the largest timescale it is almost thirteen times larger than the first moment. This could indicate that there is something that operates on a very long timescale. This is repeated in the ADSL8RM trace, seen in Figure 6.12, where the second moment is 8.4 times larger than the first moment. Except for these values, the shapes of both traces are well behaved, and look like the sub-process shape. This is a behaviour similar to the BCOct89Ext4 trace.

6.3.2 Matching

As usual default settings were used. The number of sub-processes for the ADSL8RM was estimated to 12 and 15 for the ADSM8MR trace.

ADSL8RM

The results are shown in Table 6.7, and obviously both SA and GA have failed totally. With twelve sub-processes there is a total of 48 variables to modify, which is probably too much for the methods. HM1 produces the second best solution, with a fitness of a mere 34 using 18 sub-processes. The manual method managed

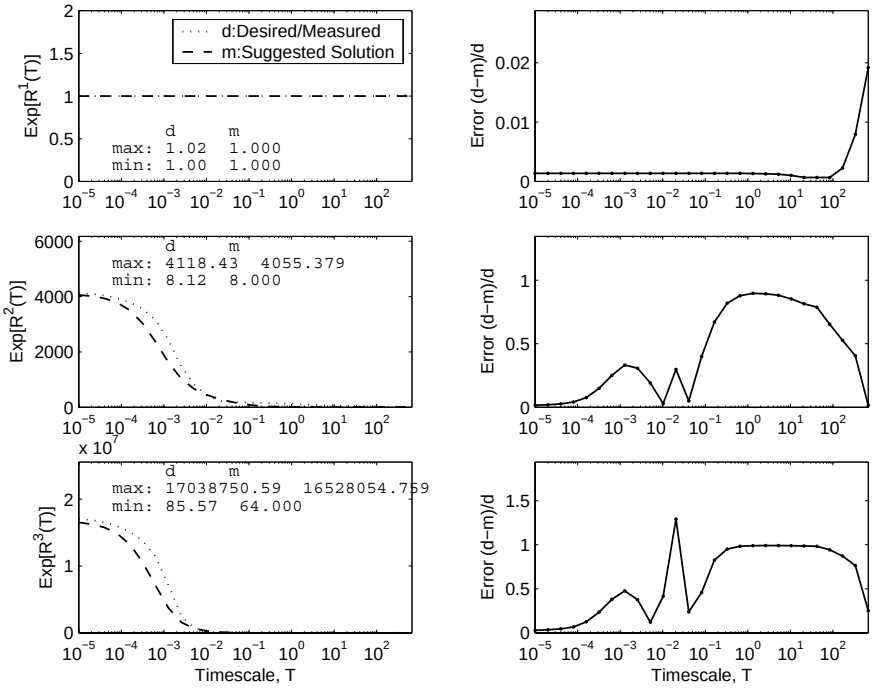

Figure 6.13: Manual methods suggested solution to the ADSL8RM trace, the fitness is 34.3949.

Table 6.8: ADSL8MR Model Fitness values. The best solution is shown in Figure 6.14.

Model	Fitness	No. of sub-processes
GA	0.376412	15
SA	0.568807	15
HM1	34.2393	23
HM2	27.8149	15
Manual	34.3	15
GA	0.376412	17
SA	0.26622	17
GA	0.376412	19
SA	0.988009	19

to obtain a fitness of 34.4 using twelve sub-processes:

$$\mathbf{P} = \begin{bmatrix} 10^{-6} & 10^{-6} & 0 & 2 \\ 10^{-6} & 10^{-6} & 0 & 2 \\ 10^{-6} & 10^{-6} & 0 & 2 \\ 10 & 10 & 0 & 2 \\ 10 & 10 & 0 & 2 \\ 10 & 10 & 0 & 2 \\ 10 & 10 & 0 & 2 \\ 10 & 10 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 1000 & 1000 & 0 & 2 \\ 1000 & 1000 & 0 & 2 \\ 1000 & 1000 & 0 & 2 \end{bmatrix}$$

ADSL8MR

The results are shown in Table 6.8 and Figure 6.14. Interestingly, the SA manages to obtain a fitness of 27 when it uses 16 sub-processes. The GA does not reach

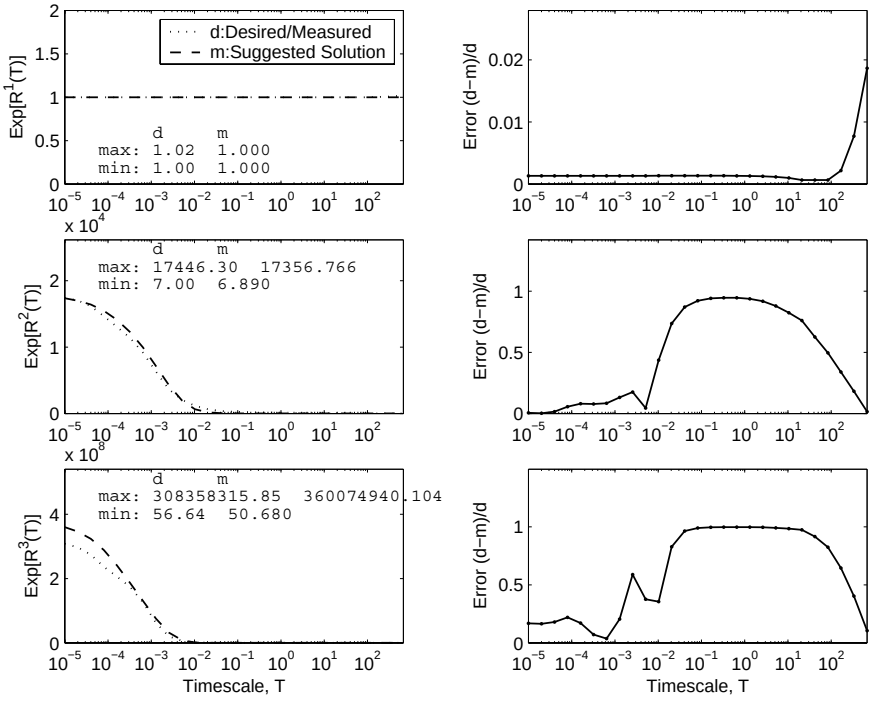

Figure 6.14: ADSL8MR model, this solution was obtained manually with a fitness of 34.4181 using 15 sub-processes.

above 0.37. The manual method obtains the best match using 15 sub-processes:

$$\mathbf{P} = \begin{bmatrix} 10^{-6} & 10^{-6} & 0 & 2 \\ 10^{-6} & 10^{-6} & 0 & 2 \\ 10^{-6} & 10^{-6} & 0.15 & 1.85 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 100 & 100 & 0 & 2 \\ 1000 & 1000 & 0 & 2 \\ 10000 & 10000 & 0.5 & 1.5 \end{bmatrix}$$

6.4 Discussion of Measurement Equipment

To model the really small timescales found on an Ethernet, TCPDUMP is insufficient. In Figure 6.15 we see the second moment for both a DAG and a TCPDUMP based measurement. The DAG card provides a 100 ns timestamp accuracy, whereas TCPDUMP 10 μs. Based on this, a base scale of 10^{-5} s can be obtained for the DAG and 10^{-3} s for the TCPDUMP, given that the base scale estimation error should be less or equal to 1%. It is clear from Figure 6.15 that a significant part of the second moment would be missing if TCPDUMP would have been used.

6.5 Observation on Measured traces

Out of the six traces that were evaluated, three obtained low total fitness values. None of the BCOct89Ext4, ADSL8RM and ADSL8MR traces obtained a total fitness above 35. In Table 6.9 the first three moments' upper edge values (i.e. moments the largest timescales) are presented, in addition to the number of timescales. In the same table the BCpOct89 trace is also included for comparison. From the table it is clearly visible that the three traces are quite different from the BCpOct89 trace. This indicates that there might be one or more processes that

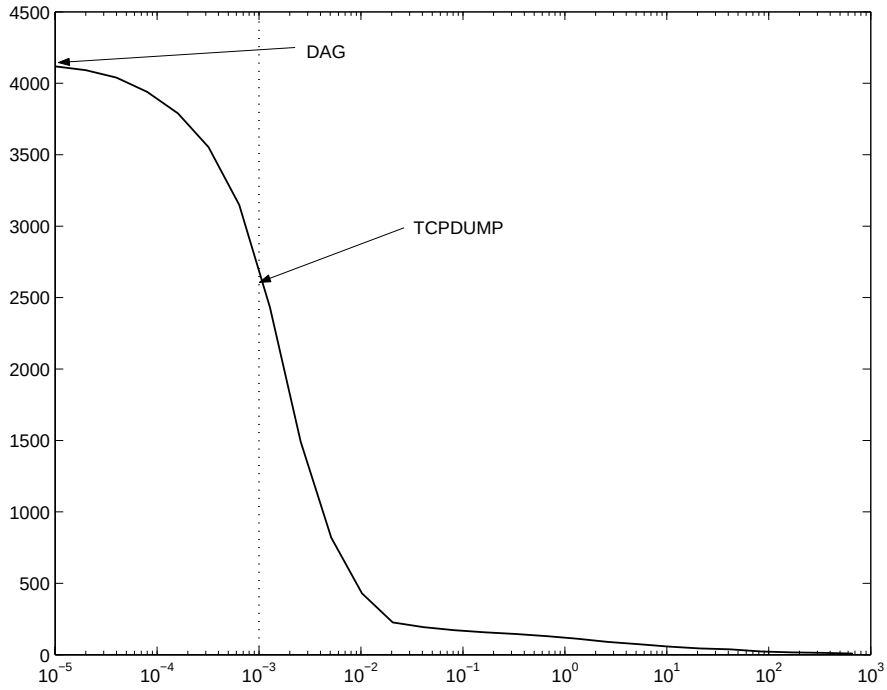

Figure 6.15: Comparison between usable timescales based on a measurement obtained either from TCPDUMP or a DAG card, when the bit rate estimation error on the base scale is required to be less than or equal to 1%, the DAG card can provide a base scale of 10 μs compared to TCPDUMPs' 1 ms.

Table 6.9: Comparision between traces w.r.t. moments' edge values.

	BCpOct89	**BCOct89Ext4**	**ADSL8RM**	**ADSL8MR**
N_T	16	22	27	27
$\Gamma^1_{N_T}$	0.994	0.9896	0.9986	0.9987
$\Gamma^2_{N_T}$	1.1	6.209	8.116	6.997
$\Gamma^3_{N_T}$	1.331	53.19	85.57	56.64
$(\Gamma^2_{N_T})^{(1/2)}$	1.05	2.49	2.85	2.65
$(\Gamma^3_{N_T})^{(2/3)}$	1.21	14.14	19.41	14.74

operates on larger timescales than measured, thus one way to correct this is to do longer measurements. Another reason can be that there is a long range-dependent process present, which is indicated by the fact that the variance does not tend towards zero as the timescales increase. In this case, self-similar or multi-fractal models should be taken into account, which is left for further studies.

6.6 Discussion on Matching Methods

In Table 6.10 we show a summary of the fitness values obtained by the matching methods. In the table we also show a fitness-per-sub-process value, it indicates how well a method utilized the sub-processes it used in order to get a value that enables us to compare the methods. The only method that is consistent is the HM1. This is probably due to that it resembles the manual matching. HM2 should in this case also perform reasonably. However, since it assumes that the shape contains only one slope, it fails when it contains more slopes.

Table 6.10: Matching Method Fitness values. For the SA and GA algorithms, we only show the cases where the number of sub-processes is equal to the calculated number ($K = \lceil \log M_2 / \log 2 \rceil$).

Fitness						
	BCpOct89	BCOct89Ext4	IAL-1-NI	IAL-1-IN	ADSL8RM	ADSL8MR
GA	77.2270	33.2364	67.0208	68.6688	0.375	0.376
SA	87.5855	0.0474	64.1735	75.5058	0.470	0.569
HM1	88.159	34.3271	78.0643	72.7274	34.15	34.24
HM2	88.7681	0.4973	62.8426	67.598	26.10	27.815
Manual	88.04	33.20	86.9	65.2	34.4	34.3
Fitness/K						
	BCpOct89	BCOct89Ext4	IAL-1-NI	IAL-1-IN	ADSL8RM	ADSL8MR
GA	38.6135	4.1546	33.5104	22.8896	0.0313	0.0251
SA	43.7927	0.0059	32.0868	25.1686	0.0392	0.0379
HM1	14.6932	2.1454	13.0107	8.0808	1.8972	1.4887
HM2	44.3841	0.0622	31.4213	22.5327	2.0077	1.8543
Manual	29.34	3.688	21.7250	9.31	2.64	2.29

The random algorithms' performance might be better when their optimization parameters are modified, e.g. increased number of generations or iterations, other selection criteria or mutation functions, etc. But the main issue for both algorithms are the boundary values. Even though the output rates were bounded to $0 \leq (r_L, r_H) \leq 2$, the transition rates were not. By giving the GA algorithm tighter bounds it managed to obtain a fitness of 89.3 using only two sub-processes and 1000 generations for the BCpOct89 trace, see Figure 6.16. The same behaviour was observed for the SA, which obtained a fitness of 90.32 when matching the BCpOct89 trace. A similar result is shown for the IAL-1-NI, where the SA obtains a fitness of 80.50 using 2 sub-processes, GA managed 84.5, seen in Figure 6.17. Thus, we should consider a two-stage matching method for both random algorithms. The first stage should only aim to find a number of sub-processes and rough boundaries for these. The next stage would run the matching algorithm using the result as input.

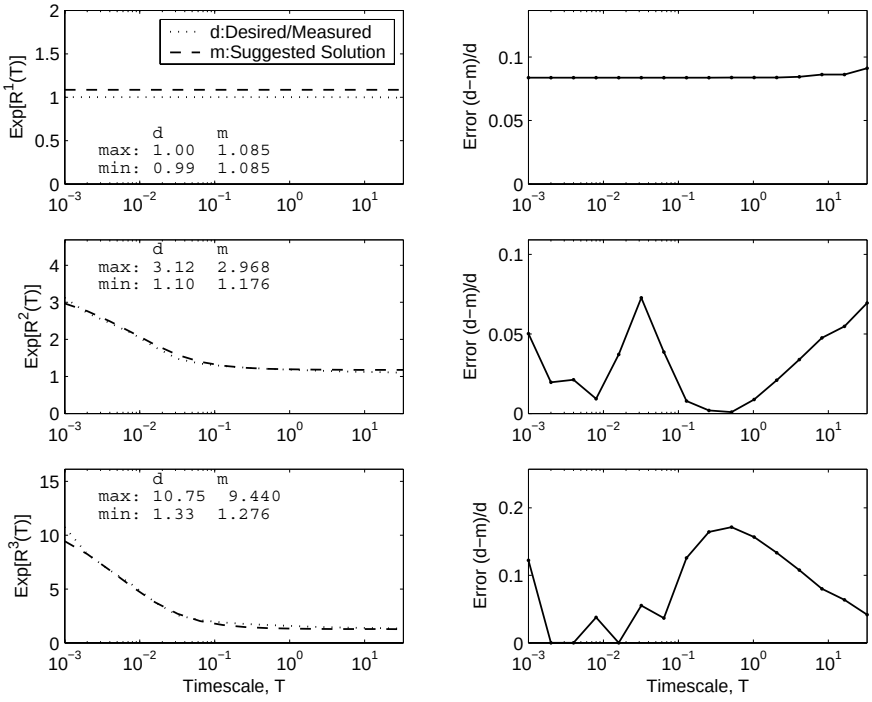

Figure 6.16: Result obtained by the GA when matching the BCpOct89 using tighter bounds. The fitness value is 89.3.

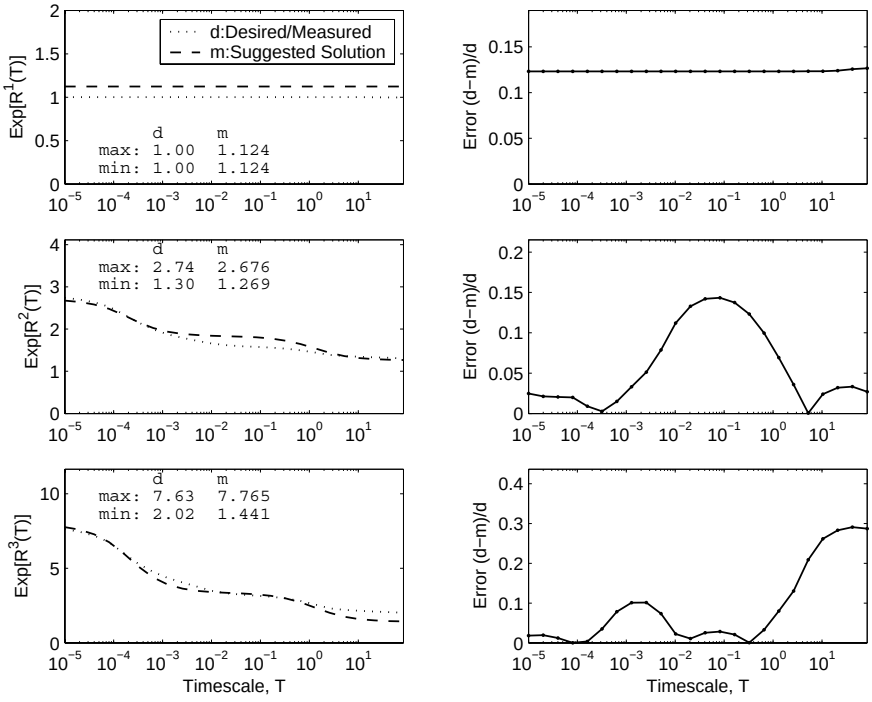

Figure 6.17: IAL-1-NI Model obtained by the GA when tighter bounds were applied. The fitness was 84.5.

Chapter 7

Conclusions and Outlook

In the introduction we set out to *"investigate the possibilities to model an Ethernet link on multiple timescale using a model that is simple, captures the scaling behaviour of the traffic and is usable in fluid flow analysis"* (page 1). The answer to this is positive, but with two constraints: the number of sub-processes needed and the number of moments to match. The number of sub-processes is a constraint set by the computer running the fluid flow analysis, about 10 for a 512MB RAM system. Some traces are less suitable for modelling all three moments. For instance, traces that indicate the possible presences of a long range-dependent process, or traces that only contain a small number of sources. How to model this is left for further study.

We evaluated five matching methods using both simulated traffic and real network traffic. The conclusion with regard to the matching methods is that all of them work, and they gain a lot if they are given good boundary values. Here future work will look into fitness weights and timescale selection. Futhermore, a two-stage matching method should be developed and the matching should be automated.

To model the small timescales found on an Ethernet, TCPDUMP is insufficient. In order to do measurements at the these timescales, the equipment needs to have a timestamp accuracy that is in the same order as the bit time on the measured link.

It would also be interesting if a model could be constructed which is continually updated. That is, in a network device, MIB entries could be present that presents the latest current model for a particular interface; one for input and one for output. This way, the model could be collected using normal SNMP.

Fluid models should be developed for some of the most common network elements like switches, routers, firewalls and combinations of these. It is possible to obtain results already using some of the existing models, but they can be quite coarse. Fluid flow analysis allows for calculation of performance parameters such as loss ratios, delay quantiles, etc. It can also be used to evaluate whether a network element behaves like a bottleneck, given a certain input.

Appendix A

Kronecker Algebra and Matrix Construction

Given two matrixes $\mathbf{A}$ and $\mathbf{B}$, let us define a identity matrix $\mathbf{I_X}$ that has the same dimension as the X matrix. Kronecker addition is defined as:

$$\mathbf{A} \oplus \mathbf{B} = \mathbf{A} \otimes \mathbf{I_B} + \mathbf{I_A} \otimes \mathbf{B} \tag{A.1}$$

and Kronecker multiplication as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \dots \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \dots \\ \dots & \dots & \dots \end{bmatrix} \tag{A.2}$$

Here $\mathbf{I_x}$ specify a identity matrix of the same size as the $\mathbf{x}$ matrix. We define the $\odot$ operator as:

$$\mathbf{A} \odot \mathbf{B} = \mathbf{A} \otimes \mathbf{I_B} \cdot \mathbf{I_A} \otimes \mathbf{B} \tag{A.3}$$

In the following, an example of how the matrices are constructed is provided:

$$\mathbf{M_0} = \begin{bmatrix} -2 & 2 \\ 2 & -2 \end{bmatrix} \quad \mathbf{M_1} = \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \quad \mathbf{M_2} = \begin{bmatrix} -0.125 & 0.125 \\ 0.125 & -0.125 \end{bmatrix}$$

$$\mathbf{R_0} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1.5 \end{bmatrix} \quad \mathbf{R_1} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1.5 \end{bmatrix} \quad \mathbf{R_2} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$

$$\mathbf{M}_{\mathrm{D}} = \mathbf{M}_0 \oplus \mathbf{M}_1 \oplus \mathbf{M}_2$$
$$\mathbf{R}_{\mathrm{D}} = \mathbf{R}_0 \odot \mathbf{R}_1 \odot \mathbf{R}_2$$

$$\mathbf{M} = \begin{bmatrix} -2.6250 & 0.1250 & 0.5000 & 0 & 2.0000 & 0 & 0 & 0 \\ 0.1250 & -2.6250 & 0 & 0.5000 & 0 & 2.0000 & 0 & 0 \\ 0.5000 & 0 & -2.6250 & 0.1250 & 0 & 0 & 2.0000 & 0 \\ 0 & 0.5000 & 0.1250 & -2.6250 & 0 & 0 & 0 & 2.0000 \\ 2.0000 & 0 & 0 & 0 & -2.6250 & 0.1250 & 0.5000 & 0 \\ 0 & 2.0000 & 0 & 0 & 0.1250 & -2.6250 & 0 & 0.5000 \\ 0 & 0 & 2.0000 & 0 & 0.5000 & 0 & -2.6250 & 0.1250 \\ 0 & 0 & 0 & 2.0000 & 0 & 0.5000 & 0.1250 & -2.6250 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 0.125 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.375 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.375 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.125 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.375 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.125 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.125 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3.375 \end{bmatrix}$$

Appendix B

Derivation of Moments and their limits

It is troublesome to directly evaluate the moment for the output rate, instead we evaluate the counting process $N(T)$ which yields the rate process $R(T)$. Let us start by defining $N_0(T)^k$ and $N_1^k(T)$ as:

$$N_0^k(T) = \begin{cases} (r_L)^k & \text{with probability } e^{-\lambda T} \\ (r_L\tau + N_1^1(T-\tau))^k & \text{with probability } \lambda e^{-\lambda\tau} \end{cases} \tag{B.1}$$

$$N_1^k(t) = \begin{cases} (r_H)^k & \text{with probability } e^{-\mu T} \\ (r_H\tau + N_0^1(T-\tau))^k & \text{with probability } \mu e^{-\mu\tau} \end{cases} \tag{B.2}$$

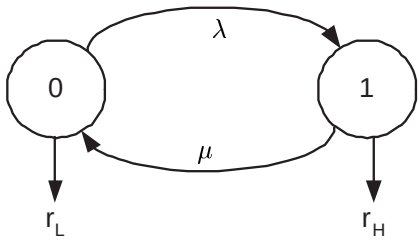

Figure B.1: State diagram for a 2-state Markov Modulated Rate Process.

where k specifies the moment that we are looking for. From this we can form $\mathbf{E}\big[N_0^k(T)\big]$ and $\mathbf{E}\big[N_1^k(T)\big]$ as

$$\begin{cases} \mathbf{E}\big[N_0^k(t)\big] = & (r_L t)^k e^{-\lambda t} + \int_0^t \big[r_L\tau + N_1^1(t-\tau)\big]^k \lambda e^{-\lambda\tau}\mathrm{d}\tau \\ \mathbf{E}\big[N_1^k(t)\big] = & (r_H t)^k e^{-\mu t} + \int_0^t \big[r_H\tau + N_0^1(t-\tau)\big]^k \mu e^{-\mu\tau}\mathrm{d}\tau \end{cases} \tag{B.3}$$

After solving that expression we can form

$$\begin{aligned} \mathbf{E}\big[N^k(T)\big] = \mathbf{Pr}\{&\text{Process starts in state } 0\}\,\mathbf{E}[N_0(T)] \\ &+ \mathbf{Pr}\{\text{Process starts in state } 1\}\,\mathbf{E}[N_1(T)] = \\ &\frac{\mu}{\lambda+\mu}\mathbf{E}\big[N_0^k(T)\big] + \frac{\lambda}{\lambda+\mu}\mathbf{E}\big[N_1^k(T)\big] \end{aligned} \tag{B.4}$$

And once this expression is obtained, we can find

$$\mathbf{E}\big[R^k(T)\big] = \frac{\mathbf{E}\big[N^k(T)\big]}{T^k} \tag{B.5}$$

This was the outline for how to calculate the kth moment. Below a detailed analysis for the first three moments is given.

B.1 First Moment Analysis

$$\begin{cases} \mathbf{E}\big[N_0^1(t)\big] = & x_0(t) = r_L t e^{-\lambda t} + \int_0^t \big[r_L\tau + \mathbf{E}\big[N_1^1(t-\tau)\big]\big]\lambda e^{-\lambda\tau}\mathrm{d}\tau \\ \mathbf{E}\big[N_1^1(t)\big] = & x_1(t) = r_H t e^{-\mu t} + \int_0^t \big[r_H\tau + \mathbf{E}\big[N_0^1(t-\tau)\big]\big]\mu e^{-\mu\tau}\mathrm{d}\tau \end{cases} \tag{B.6}$$

$$\begin{cases} x_0(t) = & r_L t e^{-\lambda t} + \int_0^t \big[r_L\tau + x_1(t-\tau)\big]\lambda e^{-\lambda\tau}\mathrm{d}\tau \\ x_1(t) = & r_H t e^{-\mu t} + \int_0^t \big[r_H\tau + x_1(t-\tau)\big]\mu e^{-\mu\tau}\mathrm{d}\tau \end{cases} \tag{B.7}$$

$$\frac{\mathrm{d}x_0(t)}{\mathrm{d}t} = r_L e^{-\lambda t} - r_L\lambda t e^{-\lambda t} + \int_0^t \frac{\mathrm{d}x_1(t-\tau)}{\mathrm{d}\tau}\lambda e^{-\lambda\tau}\mathrm{d}\tau + r_L t\lambda e^{-\lambda t} \tag{B.8}$$

Applying the Laplace (-Stieltjes) transform, we arrive at

$$\begin{cases} X_0^*(s) = & \mathcal{L}\left\{\frac{\mathrm{d}x_0(t)}{\mathrm{d}t}\right\} = \frac{r_L}{s+\lambda} + \frac{\lambda}{s+\lambda}X_1^*(s) \\ X_1^*(s) = & \mathcal{L}\left\{\frac{\mathrm{d}x_1(t)}{\mathrm{d}t}\right\} = \frac{r_H}{s+\mu} + \frac{\mu}{s+\mu}X_0^*(s) \end{cases} \tag{B.9}$$

Solving this equation yields

$$X_0^*(s) = \frac{r_L s + r_L\mu + r_H\lambda}{s(s+\lambda+\mu)} \tag{B.10}$$

$$X_1^*(s) = \frac{r_H s + r_L\mu + r_H\lambda}{s(s+\lambda+\mu)} \tag{B.11}$$

$$\tag{B.12}$$

Now we form $X^*(s)$;

$$X^*(s) = \frac{\mu}{\lambda+\mu}X_0^*(s) + \frac{\lambda}{\lambda+\mu}X_1^*(s) = \frac{\mu}{\lambda+\mu}\frac{r_L s + r_L\mu + r_H\lambda}{s(s+\lambda+\mu)} + \frac{\lambda}{\lambda+\mu}\frac{r_H s + r_L\mu + r_H\lambda}{s(s+\lambda+\mu)} = \frac{r_L\mu + r_H\lambda}{\lambda+\mu}\frac{1}{s} \tag{B.13}$$

Taking the inverse Laplace transform yields

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \mathcal{L}^{-1}\left\{X^*(s)\right\} = \frac{r_L\mu + r_H\lambda}{\lambda+\mu} \tag{B.14}$$

$$\begin{aligned} x(T) &= \int_0^T \frac{r_L\mu + r_H\lambda}{\lambda+\mu}\mathrm{d}t \\ &= \frac{r_L\mu + r_H\lambda}{\lambda+\mu}\int_0^T \mathrm{d}t \\ &= \frac{r_L\mu + r_H\lambda}{\lambda+\mu}T \end{aligned} \tag{B.15}$$

$$\mathbf{E}\left[R^1(T)\right] = \frac{\mathbf{E}\left[N^1(T)\right]}{T} = \frac{\frac{r_L\mu + r_H\lambda}{\lambda+\mu}T}{T} = \frac{r_L\mu + r_H\lambda}{\lambda+\mu} \tag{B.16}$$

Limits

As the time variable T is missing in (B.16), the limits are trivial:

$$\lim_{T\to 0}\mathbf{E}\left[R^1(T)\right] = \frac{r_L\mu + r_H\lambda}{\lambda+\mu} \tag{B.17}$$

$$\lim_{T\to \infty}\mathbf{E}\left[R^1(T)\right] = \frac{r_L\mu + r_H\lambda}{\lambda+\mu} \tag{B.18}$$

B.2 Second Moment Analysis

$$\begin{cases} \mathbf{E}\left[N_0^2(t)\right] = & y_0(t) = (r_L t)^2 e^{-\lambda t} + \int_0^t \left[r_L \tau + N_1^1(t-\tau)\right]^2 \lambda e^{-\lambda\tau} \mathrm{d}\tau \\ \mathbf{E}\left[N_1^2(t)\right] = & y_1(t) = (r_H t)^2 e^{-\mu t} + \int_0^t \left[r_H \tau + N_0^1(t-\tau)\right]^2 \mu e^{-\mu\tau} \mathrm{d}\tau \end{cases} \tag{B.19}$$

$$\begin{cases} y_0(t) = & (r_L t)^2 e^{-\lambda t} + \int_0^t \left[(r_L\tau)^2 + 2 r_L \tau x_1(t-\tau) + y_1(t-\tau)\right] \lambda e^{-\lambda\tau} \mathrm{d}\tau \\ y_1(t) = & (r_H t)^2 e^{-\mu t} + \int_0^t \left[(r_H\tau)^2 + 2 r_H \tau x_0(t-\tau) + y_0(t-\tau)\right] \mu e^{-\mu\tau} \mathrm{d}\tau \end{cases} \tag{B.20}$$

$$\begin{aligned} y_0(t) = (r_L t)^2 e^{-\lambda t} + & \\ \int_0^t (r_L\tau)^2 \lambda e^{-\lambda\tau} \mathrm{d}\tau + \int_0^t y_1(t-\tau) \lambda e^{-\lambda\tau} \mathrm{d}\tau & \\ + 2 r_L \int_0^t \tau x_1(t-\tau) \lambda e^{-\lambda\tau} \mathrm{d}\tau & \end{aligned} \tag{B.21}$$

$$\begin{aligned} \frac{\mathrm{d}y_0(t)}{\mathrm{d}t} = 2 r_L^2 t e^{-\lambda t} - (r_L t)^2 \lambda e^{-\lambda t} + & \\ + (r_L t)^2 \lambda e^{-\lambda t} + \int_0^t \frac{\mathrm{d}y_1(t-\tau)}{\mathrm{d}t} \lambda e^{-\lambda\tau} \mathrm{d}\tau & \\ + 2 r_L \int_0^t \tau \frac{\mathrm{d}x_1(t-\tau)}{\mathrm{d}t} \lambda e^{-\lambda\tau} \mathrm{d}\tau & \end{aligned} \tag{B.22}$$

$$\begin{cases} Y_0^*(s) = & \mathcal{L}\left\{\frac{\mathrm{d}y_0(t)}{\mathrm{d}t}\right\} = 2 r_L^2 \frac{1}{(s+\lambda)^2} + 2 r_L \lambda \frac{1}{(s+\lambda)^2} X_1^*(s) + \frac{\lambda}{s+\lambda} Y_1^*(s) \\ Y_1^*(s) = & \mathcal{L}\left\{\frac{\mathrm{d}y_1(t)}{\mathrm{d}t}\right\} = 2 r_H^2 \frac{1}{(s+\mu)^2} + 2 r_H \mu \frac{1}{(s+\mu)^2} X_0^*(s) + \frac{\mu}{s+\mu} Y_0^*(s) \end{cases} \tag{B.23}$$

Solving this equation yields

$$Y_0^*(s) = \frac{2\left(r_L^2 s^2 + (2 r_L^2 \mu + r_H^2 \lambda + \lambda r_L r_H) s + r_L^2 \mu^2 + 2\lambda\mu r_L r_H + \lambda^2 r_H^2\right)}{s^2 (s+\lambda+\mu)^2} \tag{B.24}$$

$$Y_1^*(s) = \frac{2\left(r_H^2 s^2 + (r_L \mu r_H + r_L^2 \mu + 2\lambda r_H^2) s + r_L^2 \mu^2 + 2\lambda\mu r_L r_H + \lambda^2 r_H^2\right)}{s^2 (s+\lambda+\mu)^2} \tag{B.25}$$

To reduce the writing, we define

$$\Delta = r_L \mu + r_H \lambda \tag{B.26}$$

$$A = r_L^2 \mu + \lambda r_H^2 \tag{B.27}$$

Now we form $Y^*(s)$

$$Y^*(s) = \frac{\mu}{\lambda+\mu}Y_0^*(s) + \frac{\lambda}{\lambda+\mu}Y_1^*(s) =$$
$$\frac{\mu}{\lambda+\mu}\frac{2\left(r_L^2 s^2 + (2r_L^2\mu + \lambda r_H^2 + \lambda r_L r_H)s + r_L^2\mu^2 + 2\lambda\mu r_L r_H + \lambda^2 r_H^2\right)}{s^2(s+\lambda+\mu)^2} +$$
$$\frac{\lambda}{\lambda+\mu}\frac{2\left(r_H^2 s^2 + (r_L\mu r_H + r_L^2\mu + 2\lambda r_H^2)s + r_L^2\mu^2 + 2\lambda\mu r_L r_H + \lambda^2 r_H^2\right)}{s^2(s+\lambda+\mu)^2} =$$
$$\frac{2\left((r_L^2\mu + \lambda r_H^2)s + r_L^2\mu^2 + 2\lambda\mu r_L r_H + \lambda^2 r_H^2\right)}{(\lambda+\mu)s^2(s+\lambda+\mu)} =$$
$$\frac{2\left((r_L^2\mu + \lambda r_H^2)s + \Delta^2\right)}{(\lambda+\mu)s^2(s+\lambda+\mu)} \quad \text{(B.28)}$$

$$Y^*(s) = \frac{2}{\lambda+\mu}\frac{\left((r_L^2\mu + \lambda r_H^2)s + \Delta^2\right)}{s^2(s+\lambda+\mu)} =$$
$$\frac{2}{\lambda+\mu}\frac{As + \Delta^2}{s^2(s+\lambda+\mu)} =$$
$$\frac{2}{\lambda+\mu}\left(\frac{As}{s^2(s+\lambda+\mu)} + \frac{\Delta^2}{s^2(s+\lambda+\mu)}\right) =$$
$$\frac{2}{\lambda+\mu}\left(\left\{\frac{A}{\lambda+\mu}\frac{1}{s} - \frac{A}{\lambda+\mu}\frac{1}{s+\lambda+\mu}\right\}\right.$$
$$\left. + \left\{\frac{\Delta^2}{\lambda+\mu}\frac{1}{s^2} - \frac{\Delta^2}{(\lambda+\mu)^2}\frac{1}{s} + \frac{\Delta^2}{(\lambda+\mu)^2}\frac{1}{s+\lambda+\mu}\right\}\right) =$$
$$\frac{2}{\lambda+\mu}\left(\frac{1}{s}\left(\frac{A}{\lambda+\mu} - \frac{\Delta^2}{(\lambda+\mu)^2}\right) + \frac{1}{s^2}\left(\frac{\Delta^2}{\lambda+\mu}\right)\right.$$
$$\left. + \frac{1}{s+\lambda+\mu}\left(\frac{\Delta^2}{(\lambda+\mu)^2} - \frac{A}{\lambda+\mu}\right)\right) =$$
$$2\frac{(\lambda+\mu)A - \Delta^2}{(\lambda+\mu)^2}\frac{1}{s} + 2\frac{\Delta^2}{(\lambda+\mu)^2}\frac{1}{s^2} + 2\frac{\Delta^2 - A(\lambda+\mu)}{(\lambda+\mu)^2}\frac{1}{s+\lambda+\mu} \quad \text{(B.29)}$$

We define

$$C = 2\frac{(\lambda+\mu)A-\Delta^2}{(\lambda+\mu)^3} \tag{B.30}$$

$$D = 2\frac{\Delta^2}{(\lambda+\mu)^2} \tag{B.31}$$

$$E = 2\frac{\Delta^2-A(\lambda+\mu)}{(\lambda+\mu)^3} \tag{B.32}$$

which allows us to $Y^*(s)$ as

$$Y^*(s) = C\frac{1}{s} + D\frac{1}{s^2} + E\frac{1}{s+\lambda+\mu} \tag{B.33}$$

Going back to the time domain, we obtain

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = \mathcal{L}^{-1}\left\{Y^*(s)\right\} = C + Dt + Ee^{-(\lambda+\mu)t} \tag{B.34}$$

$$\begin{aligned} y(t) = \int_0^T \frac{\mathrm{d}y(t)}{\mathrm{d}t}\mathrm{d}t = \int_0^T C + Dt + Ee^{-(\lambda+\mu)t}\mathrm{d}t = \\ CT + \frac{D}{2}T^2 - \frac{E}{\lambda+\mu}\left(e^{-(\lambda+\mu)T} - 1\right) = \\ CT + \frac{D}{2}T^2 + \frac{E}{\lambda+\mu}\left(1 - e^{-(\lambda+\mu)T}\right) = \\ 2\frac{(\lambda+\mu)(r_L^2\mu+\lambda r_H^2) - \Delta^2}{(\lambda+\mu)^3}T + \frac{2\frac{\Delta^2}{(\lambda+\mu)^2}}{2}T^2 \\ + \frac{2\frac{\Delta^2-(r_L^2\mu+\lambda r_H^2)(\lambda+\mu)}{(\lambda+\mu)^3}}{\lambda+\mu}\left(1 - e^{-(\lambda+\mu)T}\right) \end{aligned} \tag{B.35}$$

Here we use

$$\frac{2\left((\lambda+\mu)(r_L^2\mu+\lambda r_H^2)-\Delta^2\right)}{(\lambda+\mu)^3} =$$
$$\frac{2\left(r_L^2\lambda\mu+\lambda^2 r_H^2+\mu^2 r_L^2+\lambda\mu r_H^2-\Delta^2\right)}{(\lambda+\mu)^3} =$$
$$\frac{2\left(r_L^2\lambda\mu+\lambda^2 r_H^2+\mu^2 r_L^2+\lambda\mu r_H^2-(r_L^2\mu^2+2r_L r_H\lambda\mu+\lambda^2 r_H^2\right)}{(\lambda+\mu)^3} =$$
$$\frac{2\left(r_L^2\lambda\mu+\lambda^2 r_H^2+\mu^2 r_L^2+\lambda\mu r_H^2-r_L^2\mu^2-2r_L r_H\lambda\mu-\lambda^2 r_H^2\right)}{(\lambda+\mu)^3} =$$
$$\frac{2\left(r_L^2\lambda\mu+\lambda\mu r_H^2-2r_L r_H\lambda\mu\right)}{(\lambda+\mu)^3} =$$
$$\frac{2\lambda\mu(r_L^2+r_H^2-2r_L r_H\lambda\mu)}{(\lambda+\mu)^3} =$$
$$\frac{2\lambda\mu(r_L-r_H)^2)}{(\lambda+\mu)^3} \quad \text{(B.36)}$$

$$y(t) = 2\frac{(\lambda+\mu)(r_L^2\mu+\lambda r_H^2)-\Delta^2}{(\lambda+\mu)^3}T+\frac{\Delta^2}{(\lambda+\mu)^2}T^2$$
$$+2\frac{\Delta^2-(r_L^2\mu+\lambda r_H^2)(\lambda+\mu)}{(\lambda+\mu)^4}\left(1-e^{-(\lambda+\mu)T}\right) =$$
$$\frac{2\lambda\mu(r_L-r_H)^2}{(\lambda+\mu)^3}T+\frac{\Delta^2}{(\lambda+\mu)^2}T^2$$
$$+2\frac{(r_L^2\mu+\lambda r_H^2)(\lambda+\mu)-\Delta^2}{(\lambda+\mu)^4}\left(e^{-(\lambda+\mu)T}-1\right) =$$
$$\frac{2\lambda\mu(r_L-r_H)^2}{(\lambda+\mu)^3}T+\frac{\Delta^2}{(\lambda+\mu)^2}T^2$$
$$+2\frac{\lambda\mu(r_L-r_H)^2}{(\lambda+\mu)^3}\left(e^{-(\lambda+\mu)T}-1\right) =$$
$$\frac{\Delta^2}{(\lambda+\mu)^2}T^2+\frac{2\lambda\mu(r_L-r_H)^2}{(\lambda+\mu)^3}\left(T+\frac{e^{-(\lambda+\mu)T}-1}{\lambda+\mu}\right)$$

(B.37)

$$\mathbf{E}\big[R^2(T)\big] = \frac{\mathbf{E}\big[N^2(T)\big]}{T^2} =$$
$$\frac{1}{T^2}\left(\frac{\Delta^2}{(\lambda+\mu)^2}T^2 + \frac{2\lambda\mu(r_L-r_H)^2)}{(\lambda+\mu)^3}\left(T + \frac{e^{-(\lambda+\mu)T}-1}{\lambda+\mu}\right)\right) =$$
$$\frac{\Delta^2}{(\lambda+\mu)^2} + \frac{2\lambda\mu(r_L-r_H)^2)}{(\lambda+\mu)^3}\left(\frac{1}{T} + \frac{e^{-(\lambda+\mu)T}-1}{T^2(\lambda+\mu)}\right) \quad \text{(B.38)}$$

Limits

Let us define

$$C_1 = \frac{2\lambda\mu(r_L-r_H)^2}{(\lambda+\mu)^3} \quad \text{(B.39)}$$
$$C_2 = \frac{\Delta^2}{(\lambda+\mu)^2} \quad \text{(B.40)}$$

$$\lim_{T\to 0}\mathbf{E}\big[R^2(T)\big] = \lim_{T\to 0}\frac{2\lambda\mu(r_L-r_H)^2)}{(\lambda+\mu)^3}\left(\frac{1}{T} + \frac{e^{-(\lambda+\mu)T}-1}{T^2(\lambda+\mu)} + \frac{\Delta^2}{(\lambda+\mu)^2}\right) =$$
$$\lim_{T\to 0} C_1\left(\frac{1}{T} + \frac{e^{-(\lambda+\mu)T}-1}{T^2(\lambda+\mu)}\right) + C_2 =$$
$$\lim_{T\to 0}\frac{C_1}{T} + C_1\frac{e^{-(\lambda+\mu)T}}{T^2(\lambda+\mu)} - C_1\frac{e^{-(\lambda+\mu)T}}{T^2(\lambda+\mu)} + C_2 =$$
apply l'Hôpital's rule
$$\lim_{T\to 0}\frac{0}{1} + C_1\frac{(\lambda+\mu)e^{-(\lambda+\mu)T}}{2T(\lambda+\mu)} - \frac{0}{2T(\lambda+\mu)} + C_2 =$$
$$\lim_{T\to 0} -C_1\frac{e^{-(\lambda+\mu)T}}{2T} + C_2 =$$
apply l'Hôpital's rule
$$\lim_{T\to 0} -\frac{C_1}{2}\frac{-(\lambda+\mu)e^{-(\lambda+\mu)T}}{1} + C_2 =$$
$$\lim_{T\to 0}\frac{C_1(\lambda+\mu)}{2}e^{-(\lambda+\mu)T} + C_2 = \frac{C_1(\lambda+\mu)}{2} + C_2 =$$
$$\frac{2\lambda\mu(r_L-r_H)^2)}{(\lambda+\mu)^3}\frac{(\lambda+\mu)}{2} + \frac{\Delta^2}{(\lambda+\mu)^2} =$$
$$\frac{\lambda\mu(r_L-r_H)^2)}{(\lambda+\mu)^2} + \frac{(r_L\mu+r_H\lambda)^2}{(\lambda+\mu)^2} \quad \text{(B.41)}$$

$$\lim_{T\to\infty}\mathbf{E}\big[R^2(T)\big]=\lim_{T\to\infty}\frac{2\lambda\mu(r_L-r_H)^2)}{(\lambda+\mu)^3}\left(\frac{1}{T}+\frac{e^{-(\lambda+\mu)T}-1}{T^2(\lambda+\mu)}\right)+\frac{\Delta^2}{(\lambda+\mu)^2}=\frac{2\lambda\mu(r_L-r_H)^2)}{(\lambda+\mu)^3}(0+0)+\frac{(r_L\mu+r_H\lambda)^2}{(\lambda+\mu)^2}=\frac{(r_L\mu+r_H\lambda)^2}{(\lambda+\mu)^2} \quad \text{(B.42)}$$

B.3 Third Moment Analysis

$$\begin{cases} w_0(t)=\mathbf{E}\big[N_0^3(t)\big]= & (r_Lt)^3e^{-\lambda t}+\int_0^t\big[(r_L\tau)+N_1^1(t-\tau)\big]^3\lambda e^{-\lambda\tau}\mathrm{d}\tau \\ w_1(t)=\mathbf{E}\big[N_1^3(t)\big]= & (r_Ht)^3e^{-\mu t}+\int_0^t\big[(r_H\tau)+N_0^1(t-\tau)\big]^3\mu e^{-\mu\tau}\mathrm{d}\tau \end{cases} \quad \text{(B.43)}$$

$$\begin{cases} w_0(t)= & (r_Lt)^3e^{-\lambda t}+\int_0^t\big[(r_L\tau)^3+(r_L\tau)^2x_1(t-\tau) \\ & \quad +(r_L\tau)y_1(t-\tau)+w_1(t-\tau)\big]\lambda e^{-\lambda\tau}\mathrm{d}\tau \\ w_1(t)= & (r_Ht)^3e^{-\mu t}+\int_0^t\big[(r_H\tau)^3+(r_H\tau)^2x_0(t-\tau) \\ & \quad +(r_L\tau)y_0(t-\tau)+w_0(t-\tau)\big]\mu e^{-\mu\tau}\mathrm{d}\tau \end{cases} \quad \text{(B.44)}$$

$$\begin{aligned}\frac{\mathrm{d}w_0(t)}{\mathrm{d}t}&=3r_L^3t^2e^{-\lambda t}-r_L^3t^3\lambda e^{-\lambda t}\\&\quad+\int_0^t\left[3r_L^2\tau^2\frac{\mathrm{d}x_1(t-\tau)}{\mathrm{d}t}+3r_L\tau\frac{\mathrm{d}y_1(t-\tau)}{\mathrm{d}t}+\frac{\mathrm{d}w_1(t-\tau)}{\mathrm{d}t}\right]\lambda e^{-\lambda\tau}\mathrm{d}\tau\\&\quad+\big((r_L\tau)^3+3(r_Lt)^2x_1(0)+3r_Lty_1(0)+w_1(0)\big)\lambda e^{-\lambda t}\\&=3r_L^3t^2e^{-\lambda t}+3\int_0^t(r_L\tau)^2\frac{\mathrm{d}x_1(t-\tau)}{\mathrm{d}t}\lambda e^{-\lambda t}\mathrm{d}\tau\\&\quad+3\int_0^t(r_L\tau)\frac{\mathrm{d}y_1(t-\tau)}{\mathrm{d}t}\lambda e^{-\lambda t}\mathrm{d}\tau+\int_0^t\frac{\mathrm{d}w_1(t-\tau)}{\mathrm{d}t}\lambda e^{-\lambda t}\mathrm{d}\tau\end{aligned} \quad \text{(B.45)}$$

In the Laplace domain, we obtain

$$\begin{cases} W_0^*(s)= & \mathcal{L}\left\{\frac{\mathrm{d}w_0(t)}{\mathrm{d}t}\right\}=\frac{6r_L^3}{(s+\lambda)^3}+\frac{6r_L^2\lambda}{(s+\lambda)^3}X_1^*(s)+\frac{3r_L\lambda}{(s+\lambda)^2}Y_1^*(s)+\frac{\lambda}{s+\lambda}W_1^*(s) \\ W_1^*(s)= & \mathcal{L}\left\{\frac{\mathrm{d}w_1(t)}{\mathrm{d}t}\right\}=\frac{6r_H^3}{(s+\mu)^3}+\frac{6r_H^2\mu}{(s+\mu)^3}X_0^*(s)+\frac{3r_H\mu}{(s+\mu)^2}Y_0^*(s)+\frac{\mu}{s+\mu}W_0^*(s) \end{cases} \quad \text{(B.46)}$$

To solve this system of equations, we use the results from the first and second moments, and we create some new definitions:

$$X = A + BY \tag{B.47}$$
$$Y = C + DX \tag{B.48}$$

where $X = W_0^*(s), Y = W_1^*(s)$ and

$$\begin{aligned}
A &= \frac{6r_L^3}{(s+\lambda)^3} + \frac{6r_L^2\lambda}{(s+\lambda)^3}X_1^*(s) + \frac{3r_L\lambda}{(s+\lambda)^2}Y_1^*(s)\\
B &= \frac{\lambda}{s+\lambda}\\
C &= \frac{6r_H^3}{(s+\mu)^3} + \frac{6r_H^2\mu}{(s+\mu)^3}X_0^*(s) + \frac{3r_H\mu}{(s+\mu)^2}Y_0^*(s)\\
D &= \frac{\mu}{s+\mu}\\
X_0^*(s) &= \frac{r_L(s+\mu)+\lambda r_H}{s(s+\lambda+\mu)}\\
X_1^*(s) &= \frac{r_H(s+\lambda)+\mu r_L}{s(s+\lambda+\mu)}\\
Y_0^*(s) &= \frac{2\left(r_L^2s^2+(2r_L^2\mu+r_Lr_H\lambda+r_H^2\lambda)s+(r_H\lambda)^2+2r_Lr_H\lambda\mu+(r_L\mu)^2\right)}{s^2(s+\lambda+\mu)^2}\\
Y_1^*(s) &= \frac{2\left(r_H^2s^2+(2r_H^2\lambda+r_Lr_H\mu+r_L^2\mu)s+(r_L\mu)^2+2r_Lr_H\lambda\mu+(r_H\lambda)^2\right)}{s^2(s+\lambda+\mu)^2}\\
\Delta &= r_L\mu + \lambda r_H
\end{aligned}$$

If we simplify A and C we get

$$\begin{aligned}
A &= \frac{6r_L\left((r_Ls)^2+(r_H^2\lambda+r_Lr_H\lambda+2r_L^2\mu)s+\Delta^2\right)}{s^2(s+\lambda)(s+\lambda+\mu)^2}\\
C &= \frac{6r_H\left((r_Hs)^2+(r_L^2\mu+r_Lr_H\mu+2r_H^2\lambda)s+\Delta^2\right)}{s^2(s+\mu)(s+\lambda+\mu)^2}
\end{aligned}$$

Using this we can easily solve the equation;

$$W_0^*(s) = \frac{A+BC}{1-BD} \tag{B.49}$$
$$W_1^*(s) = \frac{C+DA}{1-BD} \tag{B.50}$$

Which after simplification yields;

$$\begin{aligned}
W_0^*(s) = \frac{6}{s^3(s+\lambda+\mu)^3}\big((r_L^3s^3 + (r_Lr_H^2\lambda + \lambda r_L^2r_H + 3r_L^3\mu + \lambda r_H^3)s^2\\
+(2r_L\lambda r_H^2\mu + 4\lambda r_H\mu r_L^2 + 3r_L^3\mu^2 + r_Lr_H^2\lambda^2 + 2\lambda r_H^3)s + \Delta^3\big)
\end{aligned} \tag{B.51}$$

$$W_1^*(s) = \frac{6}{s^3(s+\lambda+\mu)^3}\left((r_H^3 s^3 + (r_H r_L^2\mu + \mu r_H^2 r_L + 3r_H^3\lambda + \mu r_L^3)s^2 + (2r_H\mu r_L^2\lambda + 4\lambda r_H\mu r_L^2 + 3r_H^3\lambda^2 + r_H r_L^2\mu^2 + 2\mu r_L^3)s + \Delta^3\right) \quad \text{(B.52)}$$

From this we find

$$W^*(s) = \frac{\mu}{\lambda+\mu}W_0^*(s) + \frac{\lambda}{\lambda+\mu}W_1^*(s)$$

This equation is quite large, but after some simplifications, we end up with

$$W^*(s) = \frac{6\left((r_L^3\mu + \lambda r_H^3)s^2 + (2\mu^2 r_L^3 + 2\mu\lambda r_L^2 r_H + 2\lambda r_L r_H^2\mu + 2\lambda^2 r_H^3)s + \Delta^3\right)}{(\lambda+\mu)s^3(s+\lambda+\mu)^2} \quad \text{(B.53)}$$

Now we rewrite this as

$$W^*(s) = \frac{6(As^2 + Bs + \Delta^3)}{(\lambda+\mu)s^3(s+\lambda+\mu)^2} \quad \text{(B.54)}$$

where

$$\begin{aligned} G &= (r_L^3\mu + \lambda r_H^3) \\ H &= (2\mu^2 r_L^3 + 2\mu\lambda r_L^2 r_H + 2\lambda r_L r_H^2\mu + 2\lambda^2 r_H^3) \end{aligned}$$

$$W^*(s) = \frac{6}{\lambda+\mu}\left(\frac{Gs^2}{s^3(s+\lambda+\mu)^2} + \frac{Hs}{s^3(s+\lambda+\mu)^2} + \frac{\Delta^3}{s^3(s+\lambda+\mu)^2}\right) \quad \text{(B.55)}$$

Here we use some re-formulations

$$\begin{aligned} \frac{G}{s(s+\lambda+\mu)^2} &= \frac{G}{(\lambda+\mu)^2 s} - \frac{G}{(\lambda+\mu)^2(s+\lambda+\mu)} - \frac{G}{(\lambda+\mu)(s+\lambda+\mu)^2} \\ \frac{H}{s^2(s+\lambda+\mu)^2} &= -\frac{2H}{(\lambda+\mu)^2 s} + \frac{H}{(\lambda+\mu)^2 s^2} + \frac{2H}{(\lambda+\mu)^3(s+\lambda+\mu)} + \frac{H}{(\lambda+\mu)^2(s+\lambda+\mu)^2} \\ \frac{\Delta^3}{s^3(s+\lambda+\mu)^2} &= \frac{3\Delta^3}{(\lambda+\mu)^4 s} - \frac{2\Delta^3}{(\lambda+\mu)^3 s^2} + \frac{\Delta^3}{(\lambda+\mu)^2 s^3} - \frac{3\Delta^3}{(\lambda+\mu)^4(s+\lambda+\mu)} - \frac{\Delta^3}{(\lambda+\mu)^3(s+\lambda+\mu)^2} \end{aligned}$$

This allows us to write $W^*(s)$ as:

$$W^*(s) = \frac{6}{\lambda+\mu}\left(\frac{1}{s}\left(\frac{G}{(\lambda+\mu)^2} + \frac{3\Delta^3}{(\lambda+\mu)^4} - \frac{2H}{(\lambda+\mu)^2}\right) + \frac{1}{s^2}\left(\frac{H}{(\lambda+\mu)^2} - \frac{2\Delta^3}{(\lambda+\mu)^3}\right) + \frac{1}{s^3}\left(\frac{\Delta^3}{(\lambda+\mu)^2}\right) + \frac{1}{s+\lambda+\mu}\left(\frac{2H}{(\lambda+\mu)^3} - \frac{G}{(\lambda+\mu)^2} - \frac{3\Delta^3}{(\lambda+\mu)^4}\right) + \frac{1}{(s+\lambda+\mu)^2}\left(\frac{H}{(\lambda+\mu)^2} - \frac{G}{\lambda+\mu} - \frac{\Delta^3}{(\lambda+\mu)^3}\right)\right) \quad \text{(B.56)}$$

We add some more definitions:

$$\begin{aligned} K &= \frac{G}{(\lambda+\mu)^2} - \frac{2H}{(\lambda+\mu)^3} + \frac{3\Delta^3}{(\lambda+\mu)^4} \\ L &= \frac{H}{(\lambda+\mu)^2} - \frac{2\Delta^3}{(\lambda+\mu)^3} \\ O &= \frac{2H}{(\lambda+\mu)^3} - \frac{G}{(\lambda+\mu)^2} - \frac{3\Delta^3}{(\lambda+\mu)^4} = -K \\ P &= \frac{H}{(\lambda+\mu)^2} - \frac{G}{\lambda+\mu} - \frac{\Delta^3}{(\lambda+\mu)^3} \end{aligned}$$

This simplifies $W^*(s)$'s representation:

$$W^*(s) = \frac{6}{\lambda+\mu}\left(\frac{K}{s} + \frac{L}{s^2} + \frac{1}{s^3}\frac{\Delta^3}{(\lambda+\mu)^2} + \frac{O}{s+\lambda+\mu} + \frac{P}{(s+\lambda+\mu)^2}\right) \quad \text{(B.57)}$$

Now, we apply inverse transformation and obtain

$$\frac{\mathrm{d}w(t)}{\mathrm{d}t} = \frac{6}{\lambda+\mu}\left(K + Lt + \frac{\Delta^3}{2(\lambda+\mu)^2}t^2 + Oe^{-(\lambda+\mu)t} + Pte^{-(\lambda+\mu)t}\right) \quad \text{(B.58)}$$

$$w(T) = \int_0^T \frac{\mathrm{d}w(t)}{\mathrm{d}t}\mathrm{d}t =$$
$$\frac{6}{\lambda+\mu}\left[KT + \frac{L}{2}T^2 + \frac{\Delta^3}{6(\lambda+\mu)^2}T^3 + \frac{O}{\lambda+\mu}\left(1 - e^{-(\lambda+\mu)T}\right)\right.$$
$$\left. + \frac{P}{(\lambda+\mu)^2}\left(1 - (\lambda T + \mu T + 1)e^{-(\lambda+\mu)T}\right)\right] =$$
$$\frac{6K}{\lambda+\mu}T + \frac{3L}{\lambda+\mu}T^2 + \frac{\Delta^3}{(\lambda+\mu)^3}T^3 + \frac{6O}{(\lambda+\mu)^2}\left(1 - e^{-(\lambda+\mu)T}\right) +$$
$$\frac{6P}{(\lambda+\mu)^3}\left(1 - (\lambda T + \mu T + 1)e^{-(\lambda+\mu)T}\right) \quad \text{(B.59)}$$

From this, we obtain $\mathbf{E}\left[R^3(T)\right]$

$$\mathbf{E}\left[R^3(T)\right] = \frac{w(T)}{T^3} =$$
$$\frac{6K}{\lambda+\mu}\frac{T}{T^3} + \frac{3L}{\lambda+\mu}\frac{T^2}{T^3} + \frac{\Delta^3}{(\lambda+\mu)^3}\frac{T^3}{T^3} + \frac{6O}{T^3(\lambda+\mu)^2}\left(1 - e^{-(\lambda+\mu)T}\right) +$$
$$\frac{6P}{T^3(\lambda+\mu)^3}\left(1 - (\lambda T + \mu T + 1)e^{-(\lambda+\mu)T}\right) =$$
$$\frac{6K}{\lambda+\mu}\frac{1}{T^2} + \frac{3L}{\lambda+\mu}\frac{1}{T} + \frac{\Delta^3}{(\lambda+\mu)^3} +$$
$$\frac{6O}{(\lambda+\mu)^2}\left(\frac{1 - e^{-(\lambda+\mu)T}}{T^3}\right) + \frac{6P}{(\lambda+\mu)^3}\left(\frac{1 - (\lambda T + \mu T + 1)e^{-(\lambda+\mu)T}}{T^3}\right) =$$
$$\frac{6K}{\lambda+\mu}\left(\frac{1}{T^2} - \left(\frac{1 - e^{-(\lambda+\mu)T}}{(\lambda+\mu)T^3}\right)\right) + \frac{3L}{\lambda+\mu}\frac{1}{T}$$
$$+ \frac{6P}{(\lambda+\mu)^3}\left(\frac{1 - (\lambda T + \mu T + 1)e^{-(\lambda+\mu)T}}{T^3}\right) + \frac{\Delta^3}{(\lambda+\mu)^3} \quad \text{(B.60)}$$

Limits

$$\lim_{T\to 0} \mathbf{E}\big[R^3(T)\big] = \lim_{T\to 0} \frac{6K}{\lambda+\mu}\left(\frac{1}{T^2} - \left(\frac{1-e^{-(\lambda+\mu)T}}{(\lambda+\mu)T^3}\right)\right) + \frac{3L}{\lambda+\mu}\frac{1}{T} + \frac{6P}{(\lambda+\mu)^3}\left(\frac{1-(\lambda T+\mu T+1)e^{-(\lambda+\mu)T}}{T^3}\right) + \frac{\Delta^3}{(\lambda+\mu)^3} \quad \text{(B.61)}$$

$$\lim_{T\to \infty} \mathbf{E}\big[R^3(T)\big] = \lim_{T\to \infty} \frac{6K}{\lambda+\mu}\left(\frac{1}{T^2} - \left(\frac{1-e^{-(\lambda+\mu)T}}{(\lambda+\mu)T^3}\right)\right) + \frac{3L}{\lambda+\mu}\frac{1}{T} + \frac{6P}{(\lambda+\mu)^3}\left(\frac{1-(\lambda T+\mu T+1)e^{-(\lambda+\mu)T}}{T^3}\right) + \frac{\Delta^3}{(\lambda+\mu)^3} \quad \text{(B.62)}$$

Bibliography

[AMS82] D. Anick, D. Mitra, and M.M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *The Bell System Technical Journal*, 61(8):1871–1894, 1982.

[AN98] A. Andersen and B. Nielsen. A Markovian approach for modeling packet traffic with long-range dependence. *IEEE Journal on Selected Areas in Communications*, 16(5):719–732, June 1998.

[BCT] Bellcore traces. Available from World Wide Web: `http://ita.ee.lbl.gov/html/contrib/BC.html` [checked 11 November, 2003].

[BNKF98] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming: An Introduction.* Morgan Kaufmann Publishers, Inc., 1998.

[BTW95] J. R. Beran, M. S. Taqqu, and W. Willinger. Long-range dependence in variable-bit-rate video traffic. *IEEE Transactions on Communications*, 43:1566–1579, 1995.

[Car] P. Carlsson. A comparison of accuracy between tcpdump, windump and dag. Available from World Wide Web: `http://www.its.bth.se/staff/pca/` [checked 11 November, 2003]. Work in progress, please contact the author.

[Car03] P. Carlsson. Measurement point: Operational description and specification, 2003. Work in progress, please contact the author.

[CEF03] P. Carlsson, A. Ekberg, and M. Fiedler. Passive monitoring infrastructure, 2003. Work in progress, please contact the author.

[CF00] P. Carlsson and M. Fiedler. Multifractal products of stochastic processes: Fluid flow analysis. In *Proceedings of the 15th Nordic Teletraffic Seminar(NTS-15), LUND*, 2000.

[CFSD90] J.D. Case, M. Fedor, M.L. Schoffstall, and C. Davin. Simple network management protocol (SNMP), May 1990. RFC1157, STD0015.

[CFT+02] P. Carlsson, M. Fiedler, K. Tutschku, S. Chevul, and A. Nilsson. Obtaining reliable bit rate measurements in SNMP-managed networks. In *Proceedings of the 15th ITC Specialist Seminar, Würzburg*, 2002.

[DAG] The DAG project. Available from World Wide Web: `http://dag.cs.waikato.ac.nz` [checked 11 November, 2003].

[Don02] S. Donnelly. *High Precision Timeing in Passive Measurements of Data Networks.* PhD thesis, The University of Waikato, 2002.

[DSI99] N. Desaulniers-Soucy and A. Iuoras. Traffic modeling with universal multifractals. In *Proceedings of Globecom*, 1999.

[EFM] Ethernet in the first mile alliance, homepage. Available from World Wide Web: `http://www.efmalliance.org` [checked 11 November, 2003].

[ENDa] Endace. Available from World Wide Web: `http://www.endace.com` [checked 11 November, 2003].

[ENDb] Dag card specifications. Available from World Wide Web: `http://www.endace.com/products.htm` [checked 11 November, 2003].

[FCN01] M. Fiedler, P. Carlsson, and A. Nilsson. Voice and multifractal data in the internet. In *Proceedings of LCN2001, Tampa/FL*, 2001.

[FGW98] A. Feldmann, A. C. Gilbert, and W. Willinger. Data networks as cascades: Investigating the multifractal nature of internet wan traffic. In *Proceedings of the SIGCOMM 98*, pages 42–55, 1998.

[FOW66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* Wiley, 1966.

[FT03] M. Fiedler and K. Tutschku. Application of the stochastic fluid flow model for bottleneck identification and classification. In *DASD, Orlando*, 2003.

[FTCN03] M. Fiedler, K. Tutschku, P. Carlsson, and A. Nilsson. Identification of performace degradation in IP networks using throughput statistics. In *ITC 18, Berlin*, 2003.

[FV97] M. Fiedler and H. Voos. *Fluid flow-Modellierung von ATM-Multiplexern. Mathematische Grundlagen und numerische Lösungsmethoden.* Herbert Utz Verlag, 1997.

[FV00] M. Fiedler and H. Voos. New results on the numerical stability of the stochastic fluid flow model analysis. In *Proceedings of Networking 2000, Paris*, 2000.

[GEA] Available from World Wide Web: `http://www.10gea.com` [checked 11 November, 2003]. Expires January 1st 2004.

[HAN99] H. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems, Concepts, Architectures, and Their Operational Application.* Morgan Kaufmann Publishers, 1^{st} edition, 1999.

[HJK] C. R. Houck, J. A. Joines, and M. G. Kay. A Genetic Algorithm for Function Optimization: A MATLAB implementation.

[HKS99] Helmut Hlavacs, Gabriele Kotsis, and Christine Steinkellner. Traffic source modeling. Technical Report TR-99101, University of Vienna, 1999. Available from World Wide Web: `http://citeseer.nj.nec.com/hlavacs99traffic.html` [checked 11 November, 2003].

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 1975.

[HTC] Hypertransport consortium, homepage. Available from World Wide Web: `http://www.hypertransport.org/` [checked 11 November, 2003].

[IBD] Infiniband ta, homepage. Available from World Wide Web: `http://www.infinibandta.org/home` [checked 11 November, 2003].

[Ing] L. Ingber. Adaptive simulated annealing. Available from World Wide Web: `http://www.ingber.com` [checked 11 November, 2003].

[IPM] Sprint ipmon. Available from World Wide Web: `http://ipmon.sprint.com` [checked 11 November, 2003].

[ISC] Internet software consortium - number of internet hosts. Available from World Wide Web: `http://www.isc.org/ds/host-count-history.html` [checked 11 November, 2003].

[Ive73] V. B. Iversen. Analyses of real teletraffic processes based on computerized measurements. *Ericsson Technics*, 29:3–64, 1973.

[KGV83] S. Kirckpatric, G. .C. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 No. 4598, 1983.

[LK91] A. M. Law and W. D. Kelton. *Simulation, Modelling & Analysis*. McGRAW-HILL, second edition, 1991.

[LTWW93] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic. In *Proceedings ACM/SIGCOMM, San Fransisco*, pages 42–55, 1993.

[LTWW94] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[LW91] W. E. Leland and D. V. Wilson. High time-resolution measurement and analysis of lan traffic: Implications for lan interconnection. In *Proceedings of IEEE Infocom*, pages 1360–1366, 1991.

[MB76] R.M Metcalfe and D.R Boggs. Ethernet: Distributed packet switching for local computer networks. *Comminications of the ACM*, 19(7), July 1976.

[MDG01] J. Micheel, S. Donnelly, and I. Graham. Precision timestamping of network packets. In *Proceedings of ACM Sigcomm Internet Measurement Workshop*, 2001.

[Mer] Merit. NFSNET statistics 1988−1995. Available from World Wide Web: `http://www.mit.edu/people/mkgray/net/web-growth-summary.html` [checked 11 November, 2003].

[Mit96] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, 1996.

[MN97] P. Mannersalo and I. Norros. Multifractal analysis of real atm traffic: A first look. Technical report, COST-257, 1997.

[MNR02] P. Mannersalo, I. Norros, and R. Riedi. Multifractal products of stochastic processes: Construction and some basic properties. *Applied Probability Trust*, November 2002.

[MR91] K. McCloghrie and M.T. Rose. Management information base for network management of tcp/ip-based internets: Mib-ii, March 1991. RFC1213, STD0017.

[MRR+53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Telle, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[MT01] S. Molnar and G. Terdik. A general fractal model of internet traffic. In *Proceedings of 26th Conference on Local Computer Networks*, pages 492–499. IEEE Computer Society, 2001. Tampa/FL.

[NAAM] E.L. Andrade Neto, A.M. Alberti, D.S. Arantes, and L.S. Mendes. A realistic model for self-similar ethernet lan traffic in simatm - an atm network simulator: Design and performance implications.

[PBF+02] P.Abry, R. Baraniuk, P. Flandrin, R. Riedi, and D. Veitch. Multiscale nature of network traffic. *IEEE Signal Processing Magazine*, pages 28–46, 2002.

[Per03] P. Persson. *Annealing Based Optimization Methods for Signal Processing Applications.* PhD thesis, Blekinge Institute of Technology, 2003.

[PF95] V. Paxon and S. Floyd. Wide area traffic - the failure of poisson modeling. *ACM/IEEE Transactions on Networking*, 3:226–244, 1995.

[PSI] PCI SIG, homepage. Available from World Wide Web: `http://www.pcisig.com` [checked 11 November, 2003].

[Rec65] I. Rechenberg. *Cybernetic Solution Path of and Experimental Problem.* Royal Aircraft Establishment (U.K.), 1965.

[Rec73] I. Rechenberg. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution.* Frommann-Holzboog, 1973.

[RIO] Rapidio ta, homepage. Available from World Wide Web: `http://www.rapidio.org/home` [checked 11 November, 2003].

[Sak] S. Sakata. Asamin. MATLAB Interface to ASA.

[SE91] T.E. Stern and A. Elwalid. Analysis of separable markov-modulated rate models for information-handling systems. *Advances in Applied Probability*, 23:105–139, 1991.

[Sei00] R. Seifert. *The Switch Book*. Wiley, 1^{st} edition, 2000.

[SHLN00] S. Shah-Heydari and T. Le-Ngoc. MMPP models for multimedia traffic. *Telecommunications Systems*, 15(3-4):273–293, Jan 2000.

[Sub00] M. Subramanian. *Network Management, principles and practice*. Addison-Wesley, 1^{st} edition, 2000.

[SV01a] P. Salvador and R. Valadas. A fitting procedure for Markov Modulated Poisson Processes with an adaptive number of states. In *Proceedings of the 9th IFIP Working Conference on Performance Modelling and Evaluation of ATM & IP Networks*, June 2001.

[SV01b] P. Salvador and R. Valadas. Framework based on Markov Modulated Poisson Processes for modeling traffic with long-range dependence. In *Proceedings of Internet Performance and Control of Network Systems II, ITCom 2001*, pages 221–232, August 2001.

[SVP03] P. Salvador, R. Valadas, and A. Pacheco. Multiscale fitting procedure using Markov Modulated Poisson Processes. *Telecommunications Systems*, 23, June 2003.

[TC] Tcpdump public repository. Available from World Wide Web: `http://www.tcpdump.org` [checked 11 November, 2003].

[Ven01] P. Venkataraman. *Applied Optimization with MATLAB Programming*. Wiley, 2001.

[WD] Windump: tcpdump for windows. Available from World Wide Web: `http://windump.polito.it/` [checked 11 November, 2003].

[YKT01] T. Yoshihara, S. Kasahara, and Y. Takahasi. Practical timescale fitting of self-similar traffic with Markov-Modulated Poisson Processes. *Telecommunications Systems*, 17(1-2):185–211, May/June 2001.