Advanced architecture for INTER-domain quality of service MONitoring, modelling and visualisation

InterMON-IST-2001-34123

Modelling and Simulation Specification (Deliverable 6)

Deliverable

Work-package No. / Title	WP5: Deliverable 6
Planned Issuing Date	23-12-02
Distribution	InterMON Consortium (Project Officer, Evaluators, Peer Reviewers)
Document Identifier	Im-wp5-v100-UniBe-DS6-pf
File name	im-wp5-v100-UniBe-DS6-pf.doc
Version	V 1.00
Editor/Author	**University of Bern (UniBe)**
Contact Person(s)	Matthias Scheidegger <mscheid@iam.unibe.ch>
Authors	Florian Baumgartner (UniBe) Tímea Dreilinger (BUTE) Gianluca Foddis (TILAB) Pedro A. Aranda Gutiérrez (TID) Tamas Mahr (BUTE) Ilka Miloucheva (SRA) Maurizio Molina (NEC) Fausto Saluta (TILAB) Matthias Scheidegger (UniBe) Carsten Schmoll (FOKUS) Joern Seger (UniDo) Cristina Soto (TID) Attila Vidacs (BUTE)

Change History

V0.1	Adapted SAR template
V0.2	Merged Load, Delay and VoIP models
V0.3	Merged goal and ns-2 section
V0.3b	Changed goal section according to TID's comments
V0.4	Included SAR and NEC/CINI contributions
V0.8	Post-Turin version, including almost everything
V0.82	Structural changes and first version of executive summary
V0.90	First Release Candidate, content-wise
V1.00	Last edits and layout

Table of Contents

List of Figures

List of Tables

Executive Summary

This document contains the specification of the modelling and simulation environment, as done by the work package 5 of the InterMON project. The aim of this work was to define novel models and modelling techniques to describe the properties and behaviour of large scale inter-domain networks that possibly consist of multiple ISP networks, autonomous systems (ASs) and the like. Using measurements from live networks these models can then be parameterized and combined to form a rendering of the conditions in the real world. This enables us to simulate and predict network behaviour — especially QoS parameters like delay, jitter and packet loss — or to perform what-if analysis by deliberately changing the simulation scenario. This concept is called *measurement based modelling and simulation*.

Being a rather specialised part of the project, work package 5 bases on the work done earlier, especially the architecture from work package 3. The interfaces to other parts of the project have significant similarities to the ones used in the work packages 4 (network measurement toolkit) and 6 (visual data mining) due to the common dependency on the InterMON inter-domain QoS database. Section 2 explains how the modelling and simulation toolkit fits into the global InterMON architecture by discussing the form and semantics of input parameters (especially SLAs), the interaction with other parts of the framework — especially the database — and the internal division of the work flow into tools, programs and the like.

Traditional network simulators suffer from scalability problems when we try to apply them to large inter-domain networks. This is true for classical packet-based simulators like ns-2 or Opnet, but also for simulators using other approaches like fluid flow simulation. A viable way to reduce the complexity of simulating large scale networks is to find suitable abstractions. Instead of considering single nodes and packets or flows we then look at their aggregates. Concepts like packet trains or session based simulation are examples for such aggregations that have already been proposed. On one hand, the drawback of such abstractions is of course lesser accuracy. On the other hand however, the performance gain may more than compensate for it. Moreover, the exact details of a simulation scenario are often unknown, in which case a well-chosen abstraction is better suited than a “wild guess” at the exact structure of the network area in question.

Based on this observation, Section 3 of the modelling and simulation toolkit specification consists of a number of modelling systems, each being an attempt to find suitable abstractions to make simulating large scale inter-domain networks more efficient.

The basic approach defines models for two abstractions: network domains (or ASs) and inter-domain links. For each abstraction the two aspects of traffic load and delay behaviour are separately modelled, using a mix of approaches from packet based and fluid flow simulation and analytical queuing theory. So called transit matrices describing the routing behaviour of a domain form the basis of this approach. Those models can be combined to form a complete multi-domain model. Furthermore, by including them into a packet based simulator, a hybrid of packet based simulation and analytical modelling is made possible. This is particularly suited to cases where inter-domain effects influence a well-known small network, like intranets based on VPNs. Future developments can also be included into this approach by predicting the changes to the single models, using ARIMA predictions for example.

Since it is unclear how much information can be expected to be published by ISPs, a reduced and in this sense more realistic approach complements the above approach in Section 3.3.

Another aspect of network modelling that can be aggregated is the area of application traffic aggregates, e.g. the combination of a significant number of similar flows originating at a common source. Modelling this aggregate instead of modelling the sources one by one can result in another performance boost. Moreover, this approach is better suited to situations where only the aggregate performance of a set of flows can be measured. Section 3.4 specifies models for VoIP, Video and web traffic.

While predictions based on time series are only a feature of the above approaches they are the basis of the approach introduced in 3.5. All properties of the simulated network are modelled by calculating stochastic models (like ARIMA models) from time series. To study the QoS of a given flow — in the present or the future — we can now combine the affected models mathematically and get an exact statistical description of the predicted behaviour.

Neural networks are a popular approach to find approximations to problems where analytical solutions are impossible or very difficult to find. The behaviour of a network domain certainly qualifies as such a problem. Section 3.6 describes a way to model this behaviour using neural networks.

Section 3.7 concludes the modelling part of this document with a specialised modelling approach named Planisfero who comes from an actual running system at Telecom Italia.

As described above the models defined here can be combined and integrated into simulations. Again there are several possible approaches, each having its own merits. Section 4 specifies these approaches and shows which models can be used in a specific approach, how they are integrated in the specialised simulator environments and how they are used in a simulation scenario.

One of the simulation approaches discussed is the integration of analytical models for domains, inter-domain links and aggregate traffic sources into the packet-based ns-2 simulator. Collapsing possibly multiple networks domains into a single node gives us the both, the possibility to efficiently study large-scale network and to combine this with fine grained, packet-based simulations of specific applications, e.g. a VoIP flows or a VPN distributed over several small sites.

Measurements are usually time series of some sort and are used to parameterise models in the toolkit. The time series simulator approach however extends this point of view by actually simulating time series based on measured ones. If we consider a flow of packets going through a domain, their cumulative delays from their source to the ingress point form a time series. When they leave again we get another time series by adding a random variable to each element of the original time series. On the whole we can statistically describe the delay experienced by the flow's packets using the resulting time series.

Classical fluid flow simulators are event driven and experience the so-called ripple effect, which can make them even less scalable than packet-based simulators in certain scenarios. The Rate and Time Continuous Fluid Simulator (RTC-FSIM) developed in the framework of the InterMon project attempts to eliminate this effect by describing the system as a set of differential equations. RTC-FSIM is a modular and extendable inter-domain simulation technology including traffic classes and priority services models to describe QoS issues. See Section 4.3 for further details.

Classical fluid flow simulators are event driven and experience the so-called ripple effect, which can make them even less scalable than packet-based simulators in certain scenarios. A novel approach implemented in the Bergholz simulator attempts to eliminate this effect by describing the system as a set of differential equations. See Section 4.3 for further details.

One could expect that models only work with one kind of simulation approach. This is not the case. Some of the models can be applied to more than one approach; ARIMA models for example can be integrated in ns-2 and be used in the time series simulator without any conceptual changes. Ultimately, the flexibility to choose from multiple models and simulation approaches greatly enhances the power of the toolkit. Like in weather forecast there is no single best model suited to all situations, but rather a combination of different models, each having its strengths and weaknesses, that gives a good and differentiated view.

1 Introduction

The InterMON inter-domain modelling and simulation environment is based on the integration of advanced techniques for modelling and simulation providing different levels of abstractions required by simulation of inter-domain applications. The main focus of the InterMON modelling and simulation environment is its usage for inter-domain traffic engineering, network planning and QoS prediction. It is based on the integration of

- different simulation approaches (packet based, fluid and time series data simulators)
- measurement based modelling techniques for automated model generation and integration into the simulation environment
- a common measurement and modelling data base.

This chapter shows the general concepts of the InterMON modelling and simulation environment and compares the InterMON concepts with the state-of-the art of the inter-domain modelling and simulation.

1.1 General Concepts of the Modelling and Simulation Environment

InterMON WP5 is aimed at the development of a scalable measurement based modelling and simulation environment for the analysis of inter-domain network behaviour. Possible application areas are the QoS analysis of application classes (VoIP, real-time multimedia, elastic traffic, etc.) in large interconnected domain infrastructures and support for inter-domain traffic engineering. Several simulation techniques are used to reach this goal (packet based simulation, continuous time fluid simulation and time series simulation).

The modelling and simulation environment is based on general abstractions for inter-domain traffic, QoS, topology and routing and different simulation techniques, which can be selected according to the requirements.

Simulation scenarios will be based on automated parameterization of models from measurements stored in the InterMON database. Different approaches for automated parameterization of models (quasi static, predictive) focussing on modelling of application classes (VoIP, Web) and aggregative traffic modelling as well as modelling of inter-domain and end-to-end QoS (delay) will be considered dependent on the simulation technology.

Also of interest are models based on rate and time continuous fluid simulation (RTC-FSIM) technology [Ber02a] using differential equations. RTC-FSIM is a modular and extendable inter-domain simulation technology for QoS based inter modelling considering traffic classes and priority services models to describe QoS issues (for instance DiffServ). The RTC-FSIM approach provides efficient measurement based modelling using MATLAB with interfaces to inter-domain models implemented in SIMULINK. The efficiency of RTC-FSIM, which is free from ripple effect (see Section 4.3) but may have significant computational overhead, is to be proved for InterMON applications, QoS analysis of applications in interconnected environments dependent on traffic load and simulation of different flows for inter-domain traffic engineering.

In addition to quasi static modelling and simulation, a special focus is the development of modelling and simulation framework for inter-domain traffic and QoS based on ARIMA prediction models.

The modelling and simulation environment is based on the integration of inter-domain routing and topology obtained from InterMON topology discovery and measurement tools interacting using the common measurement and modelling data base.

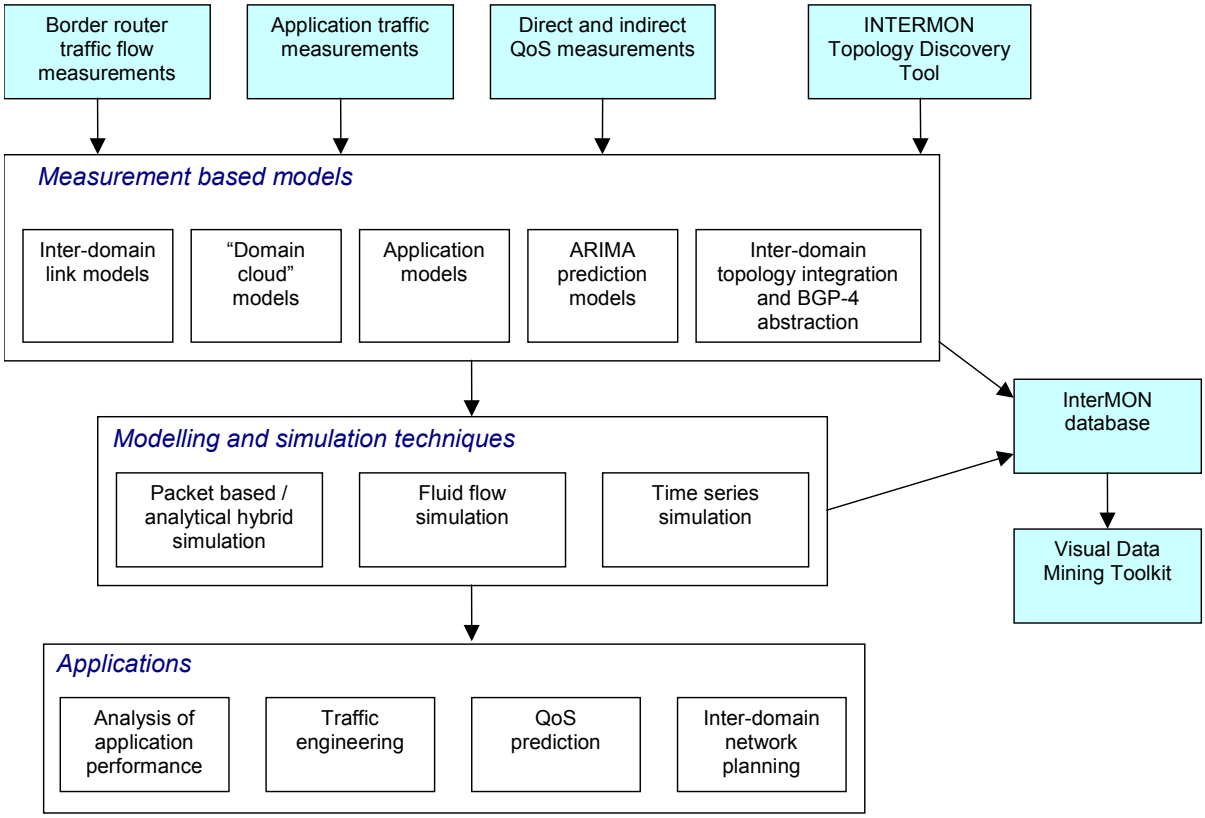

Figure 1.1 - Components of the InterMON modelling and simulation environment

1.2 Goals and Methodology

The aim of this chapter is to give a brief and high-level overview for the Modelling and Simulation Toolkit included in the InterMON Platform.

The purpose of the modelling & simulation toolkit is mainly to provide the ISPs (and possibly large corporate end users) with tools for performing a wide set of tasks. In particular, the modelling & simulation toolkit must allow predictive performance analysis, which is relevant in the short-term QoS optimization and long-term network planning and resource optimization.

1.2.1 Network and Load / Delay Models

Figure 1.2 provides a high level description of the steps followed in the modelling and simulation process.

The starting functionality, for the modelling viewpoint, is given by the design of a network model. A network model can be created in two ways:

- Importing an existent topology from the InterMON Database
- Using network topology generator functionalities (from a green field starting point);

In general, when a user wants to perform a short term QoS optimization, a good starting point is a network model imported from the InterMON database and, therefore, representative of a real network. On the contrary, when a user wants to realize a long time network planning and resource optimization, a green field scenario[1] can be a good starting model.

[1] By green field scenario we mean a scenario where the current interconnection configuration of a given domain X in question is completely neglected, intending to build a new interconnection structure from scratch.

How the network model looks like heavily depends on the user of the Simulation toolkit and on his purposes. We envisage that the network model will be in general more “detailed” around the “region” the user administers/uses and coarser further away from it. For instance, if the user is an ISP interested in evaluating end to end performances from himself to a given destination, the network model it may include details on the internal routers of its AS, details on the Border Routers of peering domains and on the SLS it has with them (but no or fewer details about the internal structure of these domains); “distant” domains may be even grouped and modelled as coarser “domain clouds” if details about all their boundary routers aren’t relevant for the purpose of the simulation.

The second starting point is given by measured Load / Delay data. Measured traffic data are useful in order to characterize the traffic profile in terms of dynamic, activity, asymmetry, classes and statistical properties.

Given the network and the Load / Delay model, the modeling and simulation toolkit performs the first step of the process computing the network state. The network state is obtained by the relationships between each network element and traffic data corresponding to the network element.

From the network state, the next step is the performance evaluation, which calculates the QoS values for the network under examination.

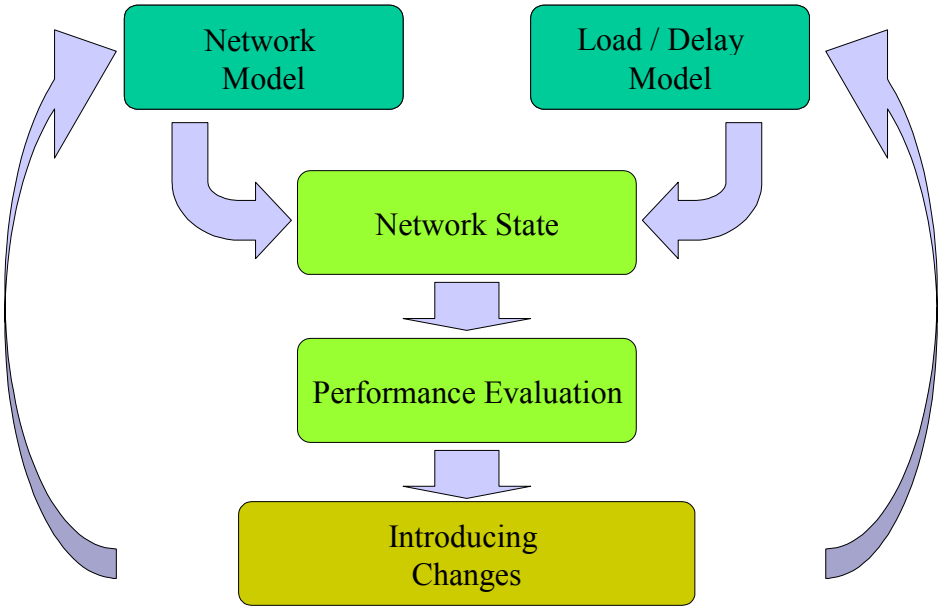

Figure 1.2 - Modelling & Simulation Toolkit Flow Chart

At this point, the user of the Simulation toolkit should have a picture of the performances he’s interested in as they are “now” (i.e. with the current a topology/routing/loads/delays) and can proceed with a what-if analysis, understanding what happens in the network state once a perturbation occurs. The changes introduced in the network and Load / Delay model can be very drastic, even bringing to a completely different model with respect to the previous one. We envisage that the main “changes” that can be introduced belong to one or more of the following categories:

- Topological / Administrative changes (like adding-removing Border Routers to the user’s domain, adding/removing/modifying SLSs between domains, modifying the inter-domain routing);
- Changes in the Load / Delay models derived by a statistical analysis of the short term past history, projecting them into the future with appropriate predictive models (for example, daily cycles or longer terms growing trends in the load or delay variation). In the following of this document predictive models to achieve such estimates are described (e.g. ARIMA models).

- Changes in the Load / Delay models not derived from statistical analysis, but rather from other type of knowledge the user has. For instance, an increase of load in a certain domain's ingress Router because the user/ISP is about to host a new "large" customer.

1.2.2 Why Load / Delay Models and How To Combine Them

As mentioned before, the Load / Delay models are the second input of the Simulation toolkit (beyond the network model). We give here motivation about why we need both types of models and how to combine them.

1.2.2.1 Delay Only Models

Let's assume, as a starting example, that a user wants to predict network End to End (E2E) performances between two network Border Routers (BR). These BR are, in general, owned by two different administrative entities and thus we speak about Inter-Domain E2E performances. We'll focus initially on delays and thus call this generic performance metric D_{E2E} , but it's possible to extend the reasoning to other performance metrics.

The user may think about directly establishing active measurement between the two end points of interest or access directly passive end to end measurements collected by some third party. In both cases, beyond administrative issues (Who is authorized to perform end to end active measurements? Who can reliably collect end to end passive measurements?) there's an obvious scalability concern: the number of possible end points grows with the square of the number of border routers, and it's unlikely to scalably repeat the procedure for each possible couple of interest.

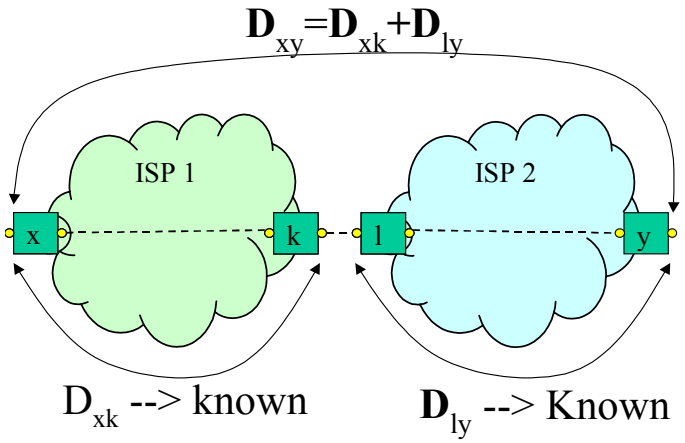

Figure 1.3 - Delay only model

In such a situation the InterMON Simulation toolkit may be of help. Suppose that instead of the end to end performance figure D_{E2E} a sequence of D_{a2b} performance figures are available (i.e. published by each domain), where a and b delimit a generic "portion" along the end to end path. D_{E2E} needs then to be estimated as a composition of these D_{a2b} . As a simple example (see picture below), think about the case where D_{E2E} is between two BR x and y, and the path between them passes through two domains, with edge points k and l, respectively. Then, $D_{x2y} = D_{x2k} + D_{l2y}$. Note that, as we're speaking about "the future", the D_{a2b} are already in general intended to be the result of some prediction based on time series analysis.

This approach can be applied in nearly all scenarios, provided the performance parameter D_{E2E} can be measured using probes.

As a very particular case, the same approach could be used to estimate, by difference, an unknown single domain delay if the end to end delay and all the other single domain delays are known.

1.2.2.2 Combined Load / Delay Models

Beyond requiring the availability of per-domain delay measurements along the whole path of interest, relying on delay models only has a second, severe limitation: it does't allow predicting performances in case the "evolution" of the traffic in the network goes beyond what can predicted by statistical prediction models. Think, for example, to the already introduced case of an ISP wanting to host a new customer, or "double home" and load balance the traffic coming from an existing, single homed customer. Then the evolution of traffic (and of delays, that depend on traffic) will face a "discontinuity" that cannot just be predicted looking at past delays. The following picture depicts how the simulation toolkit can assist in such situations where either a per domain delay is unknown or the load evolves "beyond" statistical predictions. In some domain the per domain delay is evaluated with the aid of Load models and a model of one or more network element that "translate" the load into a delay. In the picture, the two elements are the ingress and the egress router of a domain (i.e. in this case the intra domain delay in the second domain is considered negligible).

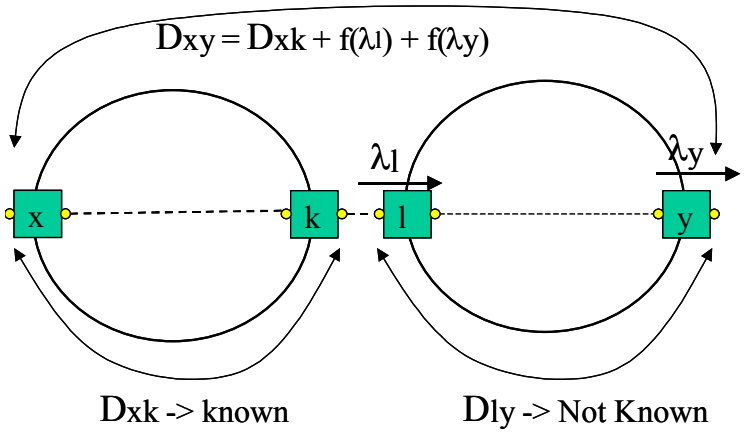

Figure 1.4 - Combined Load/Delay model

In case Load models are used in some portion of the path, a model is needed in order to translate the load on a specific "network element" to a delay. What these models are is detailed in the following of the document. They may be analytical models or simulation models, and the "network element" doesn't necessarily map one to one with a physical device. It may be, for example, the sequence of all the internal router of a domain (in case the user has access and is interested in this level of derail) or a "summary model" for the generic interconnection of multiple ISPs.

1.2.2.3 Load Transit Matrix

Note finally, always referring to the above picture, that being the focus of the InterMON platform the inter domain performance evaluation, in many cases of interest the end to end performances of a generic path are dominated by the delays at the output boundaries of a domain. This, because these are the points where the bandwidth is likely to be limited by logical SLSs between domains. This means that, in the example above, the dominating delay component in the transit of the righmost domain would be $f(\lambda_y)$. We envisage that in many cases of interest a user may want on the contrary to estimate the performances in a new situation where there'a a significant change in the *input* load (in a certain point of the domain, or in the domain as a whole). I.e. λ_l may be known, while λ_y not. Thus, a significant effort is put in finding a "transit load matrix" between input and output loads on the boundary routers of a domain. This is also described in detail in the rest of the document.

1.3 Application Framework

1.3.1 Analysis of Application Level QoS in Inter-Domain Environment s

Packet based simulation techniques are used to analyse application performance. The advantage is that an application's reactions to network effects on the packet level can be analysed. RAMP [Lan02] addresses real time simulation by using structure based modelling[2] of web services and automated integration of models into the packet based NS-2 environment. Analysis of specific applications is also meaningful for inter-domain environments, especially where the focus is on the behaviour of application QoS dependent on the inter-domain load and delay. Open challenges addressed by InterMON in this area are:

- Structural modelling and integration of models for real time services, especially VoIP and elastic services, in packet based simulator environments for detailed analysis of how performance depends on inter-domain load and QoS (delay)
- Analysis of the mutual impact of application flow aggregates on QoS parameters.

Another challenge is the integration of prediction models (ARIMA) in packet based simulator techniques and predictive modelling and simulation of applications based on structural modelling. The following figure depicts the different concepts for analysis of applications in inter-domain environment using packet based simulation:

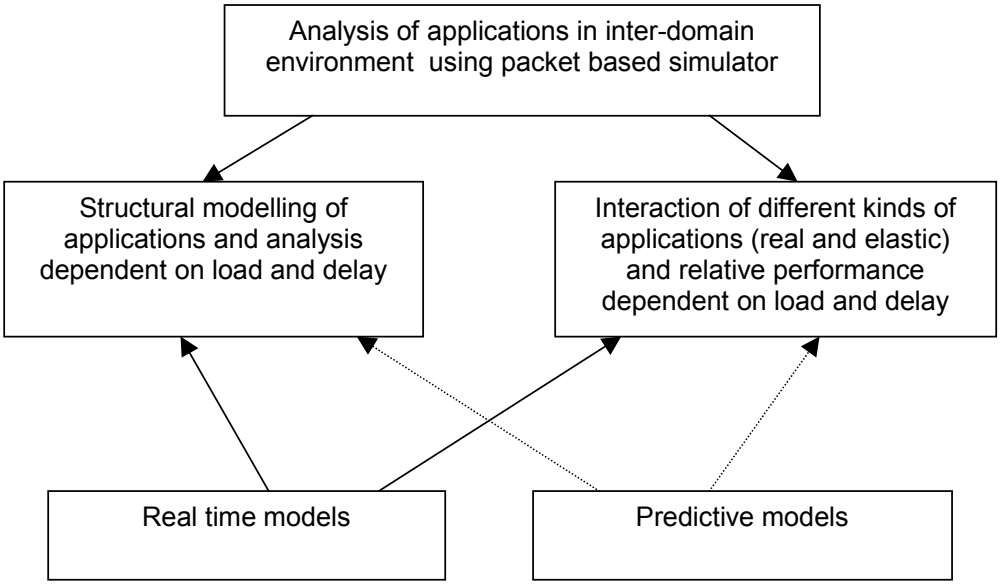

Figure 1.5 - Analysis of application QoS in inter-domain networks

The basic approach uses traffic load and delay models for the abstractions of domains (or ASs) and inter-domain links. Details are given in 3.2.

Possible prediction of performance of applications in interconnected domain environments are: Which inter-domain route is optimal to a given destination; which inter-domain route must be taken in case of network failure; is multihoming appropriate.

[2] Structure in this context means the characteristics of application flows such as the inter-arrival times of packets and the distribution of their sizes, for example. In InterMON especially the structure of application *aggregates* is of interest.

1.3.2 Inter-domain Traffic Engineering and Fluid Simulation techniques

The inter-domain traffic engineering is aimed at optimisation of border routing traffic based on flow measurements. InterMON uses the IPFIX concept for flow measurements, based on which flows are characterised by rate and duration, amongst others. An efficient way to model and simulate based on flow rate measurements is to use fluid flow modelling and simulation. The purpose of this simulation will be to study how different kinds of flows interact and influence each other depending on the load. Real-time and predictive modelling and simulation could also be addressed.

The inter-domain traffic engineering based on a border router traffic matrix is aimed at an optimal mapping of ISP border router traffic to different destinations. Simulations may assist in predicting possible traffic shifting due to the creation of peering connections or the use of other possible inter-domain connections to the destination. They may even lead to the conclusion that the inter-domain connection over this border router is not stable and another routing choice is better.

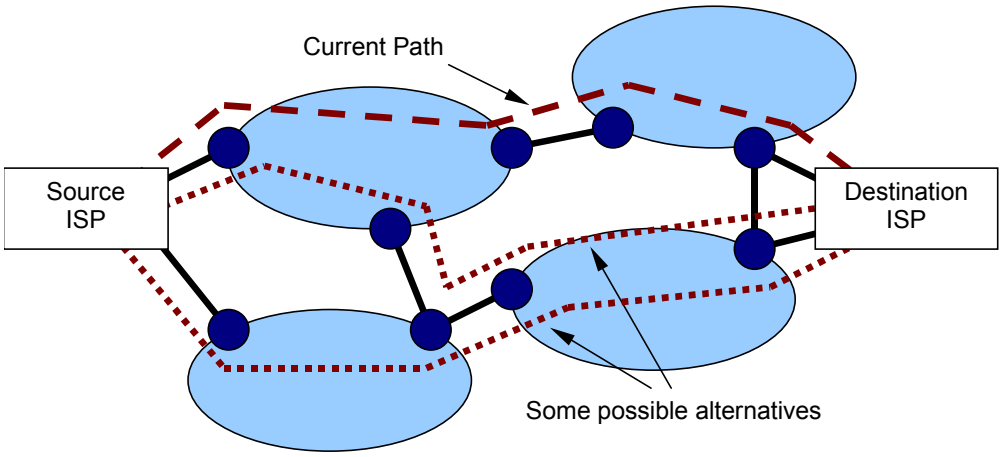

Figure 1.6 - Simulation for inter-domain traffic engineering

1.4 Comparison with the State of the Art

Abstraction is the key point in the design of modelling and simulation environments for communication networks and services based on measurements. Many research efforts discuss the performance trade-offs in selecting different abstraction levels and techniques to map real-life networks and applications to a simulated environment and back again ([Ahn96], [Bre00], [Cow99], [Hua98], [Liu01], [Nic99]). Choosing an appropriate level of abstraction is the main task required to ensure continued accuracy of results while enabling the simulator to scale to communication environments of realistic size. Modelling and simulation techniques for communication protocols and applications are characterised by their abstraction granularity and their key abstraction, i.e. their approach to mapping network structures and application services.

Examples of abstractions for modelling communication services and protocols are packet based, fluid flow, hybrid, etc. Modelling and simulation of inter-domain communication protocols and services brings further levels of abstraction, e.g. the abstraction of some intra-domain details, called domain abstraction. Each domain can be represented by a simulation node that communicates with other domains through inter-domain links.

There are several simulation techniques using different abstractions. The following table compares the most important ones and is mostly based on a table from [Gad02]:

	Simulation granularity	Key abstraction	Target domains
Ns [Ns-2]	packet	packet	Application structure, transport layer protocols and below

Packet trains [Ahn96]	Groups of packets	Treating closely spaced series of packets as large packets	Transport layer protocols
Selective abstraction [Hua98]	packet	Selectively move transient details from simulated world to the simulator	Transport level protocols and below
Fluid network simulation [Liu01], [Kum98], [Yan97], [Liu99] [Liu01], [Ber02a]	Flow changes	Map network as pipes, packet streams as fluid flows	Transport level protocol
Time stepped hybrid simulation (TSHS) [Guo00]	Packet/Time steps	Packet smoothing	Transport layer protocols

Table 1.1 - Comparison of different abstractions of simulation techniques

Simulation techniques can yield better efficiency by using

- Higher abstraction layers – for instance packet-train simulation models a cluster of closely spaced packets as a single packet train [Ahn96]
- Simulation algorithms, for instance event list algorithms are addressed in RESTART mechanism to explore rare event simulation [Vil94].

[Nic99] and [Gad02] are two proposals to solve the conflict between the need for efficiency of simulation and the need for adequate result accuracy in measurement based inter-domain simulation. Possible criteria for comparing existing inter-domain modelling and simulation environments are complexity, supported modelling and simulation techniques and the efficiency and speed of use.

As a result of a study of the state-of-the art of existing inter-domain modelling and simulation environments, following research directions for enhanced efficiency of inter-domain modelling and simulation environment could be summarised:

- Parallel and distributed simulation
- Coarse grained network simulation (abstracting fine grained network elements, like nodes and links, into coarser grained models, like domain clouds etc.)
- Application and flow abstractions for dedicated analysis of inter-domain routing, transport services and applications (looking at connections instead of single packets, for example).

The two main aspects of parallel and distributed simulation are:

- **Space decomposition**: Partition large network into disjoined individual domains, each simulated independently and concurrently with others.
- **Time decomposition**: Partition simulation time into separate intervals.

Rapid simulation based on parallelism has been used for many years to improve simulation performance ([Cha81], [Jef85], [Amm95]). Several parallel network simulators are currently available: Parsec [Bag98], SSFNET [Cow99], [Nic99] and parallel versions of ns-2 [Amm95], [Ril01]. The Scalable Simulation Framework (SSF) is an example for a large modelling and simulation environment intended to solve the scalability problem in inter-domain modelling based on

- modelling with open API for parallel discrete event simulators,
- inter-domain and intra-domain topology generation facilities
- support of large Internet protocol models BGP4, OSPF, TCP, UDP, sockets, miscellaneous client/server apps.

The Scalable Simulation Framework (SSF) is a Java and C++ based API for the description of network models. Three implementations of this API exist, one in Java and two in C++. The objectives of SSF are to provide a public domain standard of discrete-event simulation of large complex systems. However, only the core of SSF is in the public domain while most of the important application and

service models are proprietary. SSF models are compact, flexible, portable, and transparently parallelizable. At the base level the SSF API defines only five base classes: Entity, Process, Event, InChannel, and OutChannel. An Entity is a container of state, of Processes, and of communication endpoints. SSF is process-oriented; a Process is a bit of C++ code that executes when triggered by a time-out, or the arrival of an Event on an InChannel it has specified. The process may modify Entity state variables, and createEvents which are written to OutChannels. A simple SSF topology might consist of traffic sources, each of which is connected to traffic shapers. Traffic leaving a shaper enters a switch, and is routed to a host.

[Gad02] focuses on coarse-grained network simulation for wide area services including content distribution networks (CDNs), replicated Internet services, grid computing, peer-to-peer file sharing networks and other wide area storage infrastructures. The authors develop abstractions for modelling, simulation and comparative evaluation of wide-area services on large scale networks abstracting from transport level and focussing on application issues of services whose traffic consists primary of bulk transfers over reliable transport protocols like TCP. The idea is that for comparison of applications, which use common transport services and network infrastructure in large scale networks, more efficient simulators based on higher layer abstractions than standard packet based simulators could be used.

To sum up, three different approaches could be taken for efficient inter-domain modelling and simulation environment:

- parallelism and distributed simulation
- scenario prefiltering
- efficient abstraction appropriate for applications.

The InterMON inter-domain modelling and simulation environment targets these new research areas, i.e. specifically the development of efficient modelling and simulation abstractions deriving from measurements in large inter-domain infrastructures. Both packet and fluid simulation approaches are considered. .

1.5 Related Research on Automated Parameterisation and Configuration

1.5.1 Basic Internet Traffic Modelling

It is difficult to simulate and model the Internet due to its scale, heterogeneity and dynamics [Flo01]. Internet traffic is constantly changing over time both in volume and statistical properties, even observed at the same location. These changes are hard to predict, even if some characteristics are known. For example, Zhang et al. [Zha01] found that certain path properties are constant at the scale of hours. Cao et al. [Cao01] showed that, due to the lower link utilization and higher degree of multiplexing, the traffic in backbone-style links tends to have higher non-stationarity than traffic in the access links. Floyd and Paxson [Flo01] characterized the problems: The constantly-changing and decentralized nature of the Internet results in a poor understanding of traffic characteristics and makes it difficult to define a typical configuration for simulating the Internet.

1.5.2 Measurement Based Traffic Modelling and Parameterisation Approaches

The utility of simulations and analysis depends on good models of network traffic. In the InterMON context, two traffic models are particularly important:

- Aggregate border router traffic
- Traffic models considering application classes.

Because network traffic constantly changes over time, existing approaches typically take years from collecting traces and analysing the data to finally generating and implementing models [Bar98]. Three stages are involved in this time-consuming process of model creation:

- trace collection,
- design of traffic model and

- model parameterisation from measurement.

In prior work, these stages have typically been combined, with each new experiment requiring development and parameterisation of new models.

The research on Internet workload generation has typically focused on creating generative models based on packet traces of various applications. Several studies have adopted this approach to develop workload generators for web traffic including SURGE [Bar98], IPB [Mah98] and work at RPI [Yuk00]. Their work focused on fitting statistics derived from a set of traces to some widely-used distributions which are then used to generate synthetic traffic workload.

However, given that Internet traffic is changing constantly, it is not generally applicable to characterize the current traffic simply based on statistics collected years ago from different parts of the network. Moreover, even if the existence of some universal statistical property (e.g. heavy-tailed distribution of file size) is assumed, parameterisation is still a non-trivial job.

Considering the Internet's great technical and administrative diversity and immense variations over time regarding how applications are used, it is not obvious that one can model *his* traffic accurately based on the models derived from measurements taken previously and/or from other parts of the network.

Recently, a rapid model parameterisation tool based on measurements was developed. Motivated by the observation that it is important to quickly parameterise models from new data to account for the diversity of network characteristics, a tool called *RAMP* ([Lan02]) shows how models can be automatically and rapidly parameterised from traces, allowing a user to quickly instantiate models that represent current, local traffic. This approach employs a trace-analysis tool that infers traffic and topology characteristics, and a CDF[3]-based traffic model that can capture widely varying web traffic. RAMP supports rapid parameterisation of live network traffic for generating realistic application-level simulation models. It considers Floyd and Paxson's arguments [Flo01] by showing that network characteristics not only change over time but also vary in other dimensions such as locations and flow directions. RAMP can also convert live measurements into simulation models, which may then be used to generate realistic synthetic traffic. The approach is to automatically generate statistics that model user behaviour and network path characteristics by analysing TCP/IP header information captured in the measurements. The resulting model is built into the widely-used NS network simulator [Ns-2] and validated against the original trace via wavelet-based analysis. The effectiveness of the RAMP approach is the capability of generating simulation models that capture traffic dynamics in a timely fashion even when facing the ubiquitous heterogeneity of the Internet. The input is a tcpdump-format file, recorded at a single tap point in the network, that contains only TCP/IP header information. The output is a set of CDF files that model the corresponding traffic. Specifically, these CDF files consist of two types. Files of the first type model user/application level statistics of the traffic, such as user session arrival, page/file size etc. (Currently only web and FTP traffic are supported, which are among the most dominant types of traffic of the present Internet [Mcc00]). Files of the other type model path characteristics of the network. Typically it takes tens of minutes for RAMP to process a trace file with the size of several hundreds megabytes.

1.5.3 Structural Modelling for Application Classes

Traditional black box approaches typically treat the measurement as a time series. They focus on capturing the statistical characteristics (autocorrelation and marginal distribution in particular) of empirical data to model network traffic, based on various approaches such as Markov processes, ARIMA, TES etc. ([Hef86], [Mag88], [Luc94], [Gur91]). Although being able to reproduce the measured traffic correctly, as trace replay techniques, these approaches generally ignore the underlying network structure and hence provide little or no insight about the observed characteristics of measured traffic and its underlying causes.

Structural modelling, first discussed by [Wil98], proposes to implicitly take into account the complex hierarchical structure of application and intertwined networking mechanisms in order to accurately reproduce the traffic while still providing a physical explanation for observed phenomena. It addresses modelling of user/application behaviour based on source-level patterns. The traditional trace-replay techniques typically ignore the fact that traffic frequently reacts to the network's current properties, therefore source-level patterns of the data already sent are used.

[3] CDF - Cumulative Distribution Function

Opposed to trace-replay, there are several advantages to the structural modelling approach:

- Some protocols must be modelled as end-to-end entities in order to capture the feedback effect such as TCP congestion control, while trace-replay techniques typically ignore the fact that traffic is frequently *shaped* by the network's current properties,
- Internet protocols present very rich, multi-fractal behaviour across a range of time scales. Simple trace-replay approaches fail to capture this richness.
- By capturing the details of data transfer in an algorithm, traffic with much less storage requirements than trace-replay can be reproduced.

The RAMP modelling methodology as example for structural modelling and automated model parameterisation is emphasizing the source-level pattern in which data is sent. In this aspect, the models should capture the application structure *invariant* in the traffic. This helps to cope with the constantly changing nature of Internet traffic [Flo01].

Rather than treating measured traffic as a time series of statistics, traces are utilised to estimate end-user behaviour and network conditions to generate application-level simulation models. The web traffic heuristics used by RAMP were originally developed by Mah [Mah97] and Barford and Crovella [Bar98]. They imply that there are multiple levels of feedback effect within the hierarchical structure of web traffic, and each level operates at different time scales. For example, TCP has its own congestion control mechanism operating at the time scale of seconds or below, while HTTP has a request-response loop functioning at the time scale of tens of seconds. Hence, it is important to reproduce the structure of the application in the model in order to accurately reproduce the traffic. Statistics adequate for validation include the distributions of flow arrival, flow size, flow duration, packet inter-arrival time and the application properties, such as page size, page arrival, object size (for web traffic), file size, file arrival (for FTP traffic), request arrival and session duration. The CDF plots of model parameters such as page/file size, request arrival etc. also match closely, which is not surprising since the model is directly driven by those parameters.

1.5.4 Challenges for the Automated Generation of Models

The following challenges still exist in the area of automated generation of models for measurement based simulation environments:

- Inter-domain and backbone-style traffic models are required. Passive measurements at edge nodes provides sufficient information. When backbone-style traffic is captured, extra information may be obtained for bottleneck bandwidth using existing active probing techniques [patchar], [Dow99], [Car96].
- Real-time model parameterisation depends on temporal relationships between different types or characteristics of traffic. These relationships have to be investigated.
- Long-term traffic prediction should be integrated in simulations. Current models only yield correct results at the time scale of tens of minutes.
- Integration of distributed measurements and distributed network monitoring such as SCAN [Gov97] and recent work in network tomography ([CAIDA], [Mat96], [Var96], [Cao00a], [Cao00b]) require new algorithms and tools to merge distributed data into a coherent model.
- Distributed measurements are required in order to get a network-wide view of traffic and to correctly model the behaviour of cross traffic while keeping the size of collected data maintainable.

2 Modelling & Simulation toolkit

2.1 Toolkit Design Overview

The toolkit for the modelling and simulation components in the InterMON system provides a wrapper for the applications (tools) that implement the algorithms for network traffic modelling and simulation. This toolkit entity provides the included tools with a framework for control communication and data exchange. It enables the data exchange between the tools one the one hand and other work packages on the other hand via the central InterMON controller. Components from other work packages are for instance the user GUI, the visual data mining (VDM) toolkit and the InterMON database. The specified toolkit also manages the incoming computation tasks for the modelling and simulation tools in a generic way.

The following figure shows an overview of the toolkit design:

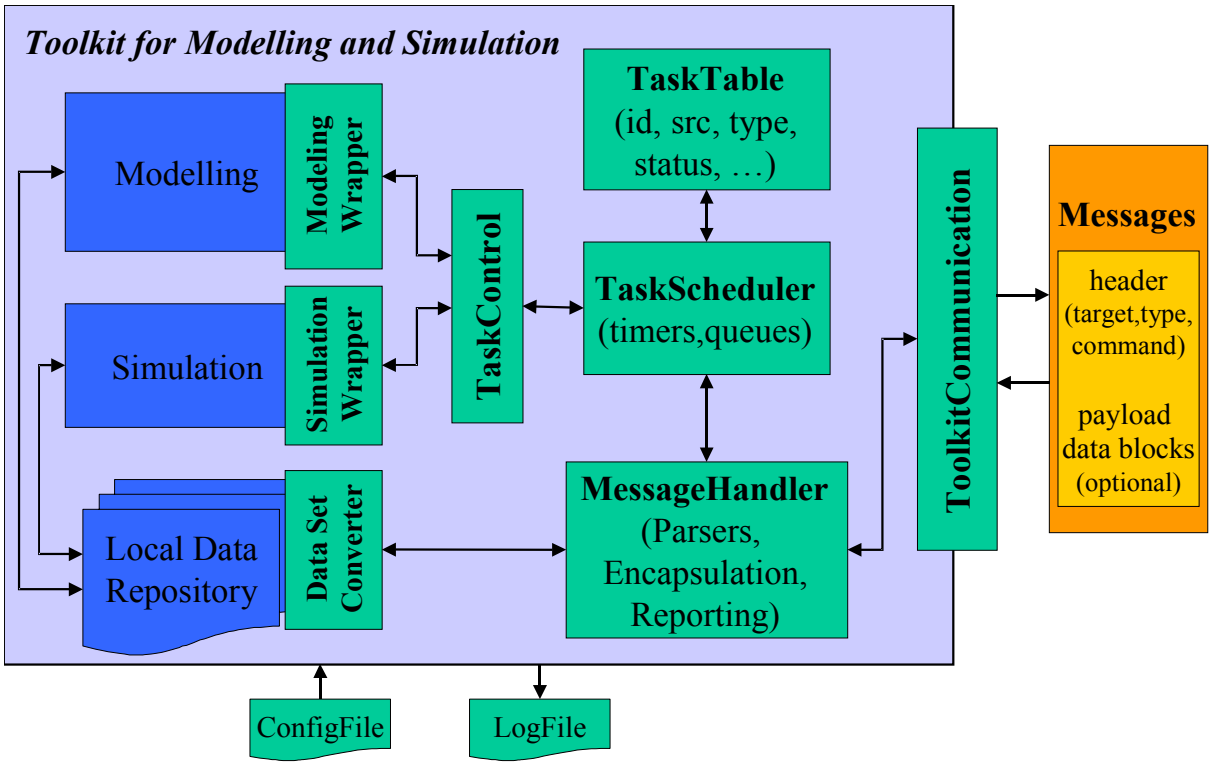

Figure 2.1 – Overview of toolkit design

The grey box of the figure shows the different components that logically build the toolkit entity. The blue boxes show the tools used in this specific toolkit, whereas the green boxes cover the toolkit-generic components. An external message data block is depicted in orange.

The tools in this figure build the worker processes in this toolkit. They perform the computational tasks for modelling and simulation and have access to data-repositories in a tool-specific format. These repositories may consist of files in a directory structure, or a tool-specific data base. Wrappers and a data set converter are required as part of the toolkit to adapt to the tools and the repository. A toolkit setup may include support for multiple tools at once. Alternatively there can be a separate setup of the toolkit for every tool instance.

The toolkit provides the management of multiple tasks (processing jobs) and communication facilities to the InterMON system for the tools. The following section gives a more detailed description of the toolkit internal components (green boxes).

2.1.1 Toolkit Components

The *ToolkitCommunication* component provides low-level methods for external communication of the toolkit entity with the rest of the InterMON system. Its functions perform data encapsulation, message framing, and network-level data transmission. This component will make use of a wide-spread middleware solution (e.g. CORBA,SOAP,JMS). It interacts with the central entities of the InterMON system in order to exchange messages from/to the toolkit. Such messages may contain control data, input data for the tools and result data from tools, as well as status queries and answers.

The *MessageHandler* component implements the InterMON-specific toolkit communication functions. It analyses incoming messages and reacts by invoking the other toolkit components dependant on the type of the message (e.g. task start command, or status request). When the toolkit needs to send messages (e.g. simulation results), this component builds a proper message from the supplied information pieces. The message handler is also responsible for syntax- and consistency-checking of the messages. It will make use of the logging functionality provided by the toolkit framework to store notifications about its progress and about observed (message) errors.

A *Message* (orange box in Figure 2.1) is not a component of the toolkit but a data entity that can be received or sent by a toolkit. A message consists of different parts. Firstly is has an envelope that is determined by the used middleware solution. Secondly it has an InterMON data header that describes what type of message it is (e.g control, query, result data, etc.) as well as information about sender and recipient of this message. And thirdly it may contain blocks of additional data that are relevant to its type (e.g. input data for a starting a task on some modelling tool in the toolkit).

The *TaskTable* component inside the toolkit manages the set of tasks which are currently known by a toolkit instance. Every task has a unique identification and a set of attributes including its type (e.g. Delay-Estimation) and status (waiting, running, finished, etc.). Each task is only known by the toolkit as long as is has not yet finished and is expunged after its results have been transmitted to the InterMON system's central database.

The *TaskScheduler* component manages timed and delayed tasks. A timed task is one that is planned to be executed a some particular time in the future. A reference for such a task is stored temporarily in this component, and the task is executed at the desired time. Therefore this component implements alarm timers and storage containers for timed tasks. A normal task might be delayed if there are currently no processing resources (e.g. simulators) available. This can happen if the setup of a simulation toolkit only allows for a limited number of tasks to be active at once. In this case the task will be queued temporarily inside this component. The TaskScheduler will start such a task immediately if resources become available.

The *TaskControl* component builds a bridge to the wrappers for the specific tools used in a particular toolkit setup. Its task is to select the appropriate wrapper(s) and forward messages and replies to/from the tool wrappers.

The *ModellingWrapper* component translates commands (e.g. start/stop/check task) from generic to tool specific syntax (and vice versa) and invokes an instance of the specific tool. This process might involve tool-specific communication to a modelling server or the explicit starting of a modelling tool. The wrapper also supplies additional parameters and references to input data (e.g files) for the tool.

The *SimulationWrapper* component translates commands (e.g. start/stop/check task) from generic to tool specific syntax (and vice versa) and invokes an instance of the specific tool (e.g. ns2). This process might involve tool-specific communication to a simulation server or the explicit starting of a simulation tool. The wrapper also supplies additional parameters and references to input data (e.g files) for the tool.

The *DataSetConverter* component translates input and output data entities to/from the legacy formats that are needed by the worker tools. This process might involve access to local files or to some sort of data base. This component is invoked by the message handler at the start end at the end of a task to provide and collect data for modelling and simulation tools and exchange these with the central InterMON controller and the central data base.

The toolkit *ConfigurationFile* stores the current setup of a specific toolkit instance. It is stored locally at the toolkit setup location but may be changed by the toolkit when configuration changes are indicated by an InterMON user (with administrative permissions for this toolkit). The toolkit configuration file describes the available tools, their types and interfaces (needed wrappers) as well as toolkit-specific parameters (e.g. identity, maintainer, capabilities, etc).

The toolkit *LogFile* will be extended by the toolkit at runtime with notifications about the progress of the toolkit communication, about task execution on the tools and when error conditions occur. The amount of logging (i.e. the verbosity) depends on the current setup of the toolkit instance.

A more detailed description of components, functionality and their associated behaviour (example sequence for job initiation and control) will be included in the deliverables D8 and D10 which define the generic toolkit specification.

2.2 Input Parameters

The modelling and simulation toolkit receives from the INTERMON platform control messages which include commands for the toolkit. Such messages can include different blocks of input data. These can be used for instance to design correctly the network and traffic models that are used in the simulation phase. If the modelling and simulation toolkit is requested by the INTERMON platform to execute a specific task then the associated control message(s) must indicate what kind of task this is and must supply all necessary input parameters.

The kind of task that is to be executed is called a simulation scenario. A scenario defines the set and the granularity of the input data, the required procedures, and the goal (i.e. type of output data sets) of a specific type of simulation and/or modelling task.

In the following, the set of information to be used in the modelling phase is described. Each listed information can be mandatory, optional or arbitrary depending on the kind of the analysis to be performed.

In this document, it is assumed that each ISP which once agrees to belong to the INTERMON platform, shares a set of information about itself. Some subset of information should be mandatory; others information could be optional.

The first input acquired from the INTERMON database is the network topologies of the ISPs. Each ISP have a degree of flexibility when sharing own data. A minimum set is constituted by: name and type of the ISP, geographical information, list of traffic ingress/egress point; list of interconnected ISPs.

The second input is the list of capacity allocated in each point of interconnection between ISPs, even in the case of connection through an IXP.

The third is constituted by the list of SLAs that an ISP have with other ISPs. The knowledge of SLAs is important in order to assign the correct ability to exchange traffic with other domains and the correct virtual capacity perceived on each interconnection link. The policies applied to the ingress traffic are considered to be included in the SLAs.

The fourth input is given by the traffic data measured in the interconnection points of the INTERMON ISPs.

The fifth input concerns the information related to the performance perceived by the traffic traversing the domain. This information could be in the form of edge-to-edge delay/jitter/loss matrices or in a more aggregate form. The export of not all the mentioned metrics may be mandatory but an ISP may choose to export the delay only, for instance. Furthermore, if for some reason this information is not available, a good estimation of the performance can be made using SLA information (see further).

Almost all the information deemed to be needed to the modelling and simulation toolkit can be gathered from the other components of the INTERMON platform. It is imported via the Global controller of the INTERMON system. On the contrary, the third input concerning SLAs is an administrative information that can't be automatically retrieved by a monitoring component, rather it should be made available with an explicit notification by ISPs participating to the INTERMON service. Furthermore, not all the information necessary to completely define a SLA is useful for the modelling and simulation toolkit. The next section provides a deeper insight on the information contained in an SLA contract.

2.2.1 SLA Information

A Service Level Agreement (SLA) is a contract that regulates relationships between a customer and a Service Provider. According to the definition of SLA for the IP networks world, given in [RFC2475], a SLA specifies the forwarding service the customer should receive. A customer may be any user (organisation owning a local network, another Internet Service Provider) willing to exploit services the provider can offer. A SLA may contain general information (e.g. security information, etc.) and application specific information (details about the service for a specific purpose, e.g. for an Internet access service information about how much traffic should be carried, QoS, etc.). The second group of information forms the so-called Service Level Specification (SLS); a formal description of a SLS for an ISP is described in [Goder].

The SLS template provided by this document contains the following information:

1. **SLS Identification**: an identifier used by the customer and the provider to identify the SLS and the service the SLS is related to;
2. **Scope**: identifies the geographical/topological region over which the QoS is to be enforced (e.g. network interfaces);
3. **Flow Identification**: identifies the IP packets allowed to exploit the service (e.g. source/destination/application information, class of service);
4. **Traffic Envelope and Conformance**: the traffic envelop describes the traffic characteristics of the traffic stream identified by the flow ID using traffic conformance parameters that, applied to a conformance test algorithm, allow to distinguish in-profile packets from excess traffic;
5. **Excess Treatment**: is the treatment reserved for the excess traffic;
6. **Performance Guarantees**: the performance that in-profile traffic should experiment (may be described in quantitative terms using performance parameters like delay, jitter, loss, throughput);
7. **Service Schedule**: the definition about when the service is offered (e.g. start/end date, etc.);
8. **Reliability**: out-of-service probability and duration;
9. **Monitoring**: parameters that have to be monitored and reported to the customer.

Information 3, 4, 5 and 6 deserve a deepen description because they are useful for the inter-domain modelling and simulation studies and have an impact on the QoS experimented by the traffic crossing domain boundaries and subject to some SLA drawn up by the administrators of the adjacent network domains.

Information 3 is in principle any specification of a traffic stream but the routing scheme adopted may put restriction on flow identification capabilities. For example BGP-4 bases its behaviour on addresses reachability announcements so that the flow identification under this routing algorithm cannot be, in general, selective on a source address basis. Depending on the flow identification, SLAs can be subdivided in peering and transport agreement. Peering agreements allow two directly interconnected domains to exchange traffic originated or terminated in the domains; transit agreements allow full accessibility to the traffic coming from two interconnected domains even if the traffic is not generated nor terminated in the domains in question. The information on AS routing resides in the BGP databases of the routers participating to the routing protocol and can be directly learnt from the routers without requiring to be explicitly stated.

The traffic envelop for conformance description proposed in the template presented in [Goder] uses a set of conformance parameters for IP traffic that have been specified in the document describing the services in the IntServ framework ([RFC2211], [RFC2212]). They are referring to the well known token bucket paradigm and include: peak rate p, token bucket rate r, bucket depth b, maximum transfer unit M, minimum packet size m. The traffic sent by a given source is said to be in-profile if the amount of data sent over all time periods T does not exceed the quantity:

$$M + \min(pT + M, rT + b)$$

Figure 2.2 shows the profile of a generic traffic (blue line) compared with the profile allowed by a token bucket (piecewise linear black envelope), they respectively represent the total amount of traffic transmitted by a given source as a function of time and the maximum amount of traffic allowed to be transmitted by the token bucket algorithm.

This token bucket algorithm is a very common mechanism used to test traffic conformance and it is also implemented in commercial devices.

Another interesting information for SLS is the treatment reserved for excess traffic. Three possible treatments can be envisaged: dropping, shaping and marking. If the dropping policy is selected, all the excess traffic is dropped. If shaping is selected, the excess traffic is buffered and delayed until its transmission is allowed by the token bucket algorithm. If marking is chosen, out-of-profile traffic is marked and carried by the network, if its transport does not affect the performance of the in-profile traffic (e.g. transported as best effort).

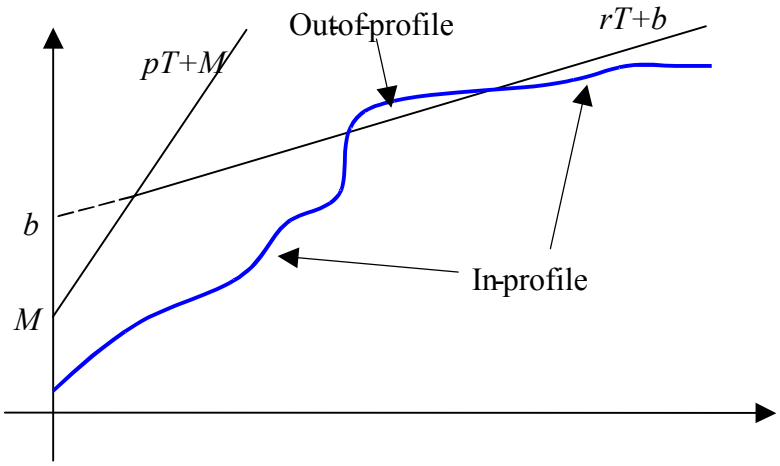

Figure 2.2 - Token bucket profile as a function of time

Both token bucket parameters and excess treatment are useful for a correct modelling.

Finally, performance guarantees can be expressed quantitatively, using delay, jitter, loss and throughput parameters, or qualitatively, expressing relative guarantees (e.g. real time service has low loss, jitter and delay guarantees while premium service has medium delay and jitter guarantees and low loss). Performance guarantees are not strictly required for modelling if performance monitoring information for all the domains is made available.

In conclusion, the information concerning SLAs that should be publicised by ISPs includes:

- Token buckets parameters,
- Excess treatment,
- Performance guarantees information if actual measurements results are not available.

2.3 Models and Simulators

The modelling and simulation toolkit contains many different models and multiple simulation approaches, which can be parameterized with measurement data from the InterMON database and combined to build a rendering of the real network, possibly with deliberate changes to perform a what-if study.

A typical session with the modelling and simulation toolkit would look like in Figure 2.3. We start with measurements stored in the database, possibly requesting further measurements from the architecture if needed. Once the measurement data has been gathered, it is fed into user-selected or automatically chosen model parameterization tools. As a result the user gets a set of models defined by their types and parameters (e.g. domain, inter-domain link and application traffic models), which are then combined to build a simulation scenario. This combination depends on a priori knowledge and/or on a measurement-based model of the inter-domain topology. The models and parameters calculated during this step can also be stored in the database and reused later in similar simulation

scenarios. This is especially useful if the user wants to run a series of simulation scenarios that only differ in specific points. Instead of recalculating all of the model parameters each time, most of them can then be taken directly from the database.

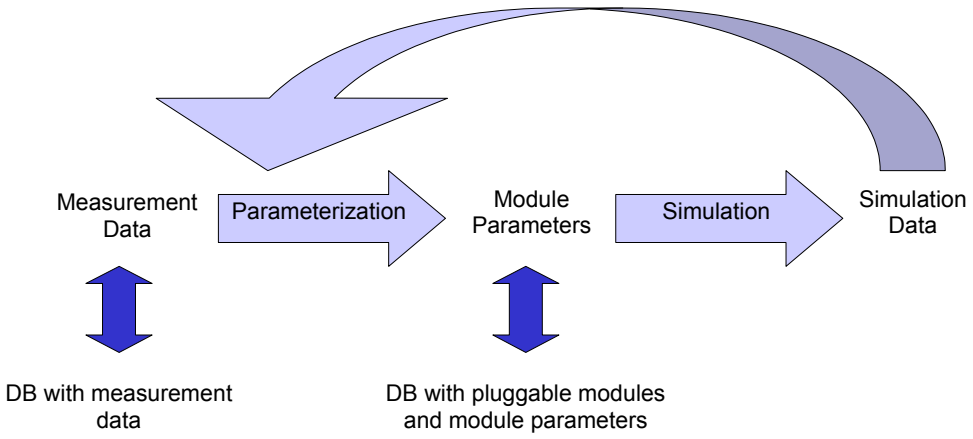

Figure 2.3 - Modelling and simulation process

Depending on the user's needs and the models chosen, the simulation scenario will then be run using one of the several simulators available. It is important to note here that some of the models can be used in multiple simulation approaches while others are specialised to a specific one. ARIMA models, for example, can be used in ns-2 based hybrid simulations as well as in the time series simulator.

Once a simulation run is complete, the output can again be stored in the database. Its structure is similar to measurements from the real network but probably much more detailed. Entries in the database originating from simulation runs should therefore be clearly marked as such. One big advantage of the similar structure of measurements and simulation output is the ability to compare the measurements of a real network setup to the results of its simulated counterpart. This makes manual or automated iterative refinement of the models possible. Especially the neural network modelling approach will profit greatly from this.

Based on the above work process the following division of the toolkit into tasks and tools appears reasonable:

- A *service front end* receives service requests from the "outside", divides them into single tasks and delegates them to other tools.
- For each model a parameterization tool (its *modeller*) collects the measurements and returns a model description in XML format, ready to be stored in the database. This tool also decides what kinds of measurements are required to create a model of the given kind.
- The *simulators* accept a scenario description in a format depending on the underlying simulation tool (e.g. TCL scripts for ns-2). However, this format must be adapted or extended to the point where it is possible to introduce and parameterize the models from above. Preferably this should be done using a plug-in mechanism. In ns-2 this is easily possible. In fact even a hot-plug mechanism can be realised.
- While modellers compute a model's parameters, another program must simulate its behaviour depending on some input when used inside a simulator. These programs are called *predictors* and take the form of dynamically loadable libraries or DLLs for short and conform to a common API (see also Section 1.1.1 for more details). When used in a simulation they simulate the behaviour of the network entity or property they abstract. A domain delay model could be plugged into a single node in the ns-2 simulator, for example.
- Common computational tasks often require similar functionality. Publicly available function libraries such as the one from the R-Project (a free implementation of S-Plus, see [SPlus] and [R]) concentrate this functionality in the *computational backend*. These libraries can be used by both, modellers and predictors.

3 Modelling

3.1 Overview

In traditional packet-based simulators the "world" is modelled in terms of nodes and links with individual capacity and delay characteristics. When simulating whole Internet domains this approach quickly becomes problematic, due to the sheer amount of events to be processed. A simplified view of the network can significantly reduce the complexity of large scale simulations, but one must give great care not to oversimplify things. Here we propose a model that we hope will result in far more efficient simulations than traditional approaches, but still give a good approximation of real network behaviour.

In our modelling view the network is divided into domains and inter-domain links. For each domain, the set of edge nodes and their links to other domains are known, but the topology inside domains is of no concern (i.e. we have a so called black box model). The connections between such domains are modelled by inter-domain link models, which implement properties like link capacity, queuing behaviour or Service Level Specifications (SLSs). Figure 3.1 gives a graphical rendering of this modelling view.

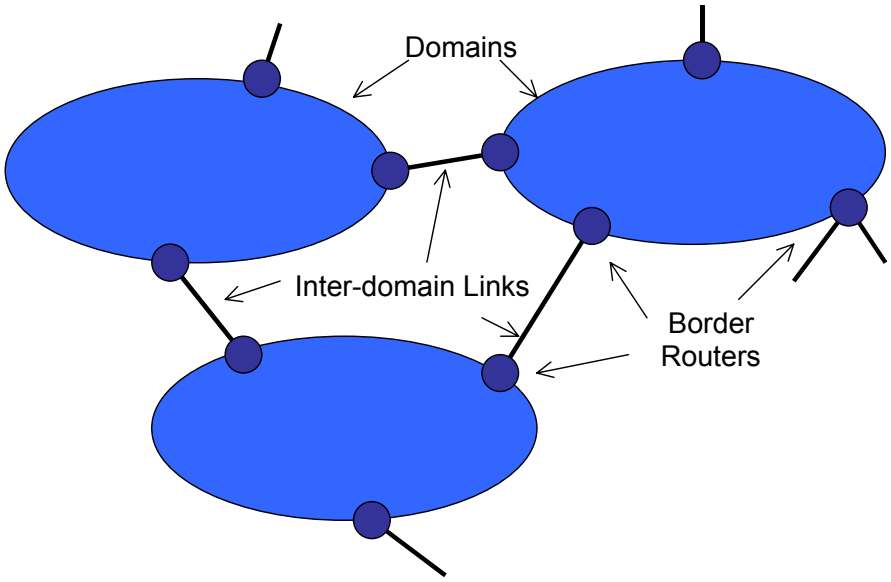

Figure 3.1 - The basic modelling view

Domain and inter-domain link models implement different aspects of network load behaviour. On one hand, domains are concerned with load distribution, i.e. they model the load on their output links depending on the load on their input links. Inter-domain link model on the other hand model the effects caused by this load between domains, which are packet loss due to link overload and SLS enforcement, amongst others. This also has consequences on the delay models of both kinds: While queuing delay in domains can be nearly neglected it is of great importance in inter-domain links, where congestion can actually occur.

Further components of this model system are the application traffic models. They serve to simulate large aggregates of application traffic (VoIP, Video, Web, etc.) and their specific properties in a scalable way. Moreover, if these models are derived from network measurements, their future properties can be predicted by applying statistical means like ARIMA models, for example.

Since the models described here represent a situation in the real world, measurements form the network are needed to configure them. For traffic load models, these measurements consist of the loads on the input and output links, respectively. Additionally, information about the distribution of inbound traffic from a given ingress node to all the outbound links can be included, where available, Furthermore, knowledge of inter-domain topology (in contrast to intra-domain topology), inter-domain link capacities and SLSs is required for realistic network modelling.

3.2 Basic Approach

3.2.1 Traffic Load Models

3.2.1.1 Domain Model

There are two main aspects to the proposed domain model. One is the traffic matrix describing the dependencies of the load parameters of inbound and outbound links, which is described in the following section. The other aspect is the continuous adaptation of this matrix based on measurements done on the live network. This is covered in 3.2.1.1.2.

3.2.1.1.1 Traffic Matrices

As mentioned above, the purpose of the domain model is to describe the distribution of traffic flowing into a domain to other domains. Depending on the quality of the available information different domain models are adequate tot fulfil this purpose. We will investigate two scenarios: In the first one only the inbound and outbound loads are considered. In the second one, the distribution of inbound traffic to the output links replaces knowledge about inbound loads.

However, before we can look at those two possibilities we should go into further detail about the possible inbound and outbound link load parameters. A domain D has a non-empty set E^D of edge nodes, each of which is connected over one or more inter-domain links to other domains. It appears reasonable to assume these links to be duplex. In our model, the number of inbound links of a domain would therefore be equal to the number of outbound links. However, it may not be possible to tell between the loads of the different links of an edge node. In such a case we consider the whole of the inbound or outbound links of the edge node in question to be one link. The numbers of inbound and outbound links thus cannot be assumed to be equal if we want to keep that flexibility.

The Transit Matrix

If we only have measurements of the loads on the inbound and outbound links, a correlation matrix is a good way to describe the dependencies of input and output loads of the domain. Let $I_t = (I_{1,t},...,I_{n,t})$ be the input bandwidths on the input links $1,...,n$ at time t and $O_t = (O_{1,t},...,O_{m,t})$ the bandwidths on the output links $1,...,m$ at time t. We seek a matrix T such that, ideally, $T \cdot I_t = O_t$, i.e. T can be used as a predictor of the output bandwidths depending on the input bandwidths. Using multivariate linear regression through the origin we get a model

$$O_{i,t} = (\beta_1,...,\beta_n) I_t + \varepsilon_{i,t}$$

of the dependencies between $O_{i,t}$ and I_t, where $\varepsilon_{i,t}$ is the random error of measurement. The combination of these models results in the domain model given by the matrix

$$T = \begin{pmatrix} \tau_{11} & \cdots & \tau_{1n} \\ \vdots & \ddots & \vdots \\ \tau_{m1} & \cdots & \tau_{mn} \end{pmatrix}$$

which is called the *transit matrix*. The domain model itself thus becomes

$$O_t = T \cdot I_t + \begin{pmatrix} \varepsilon_{1,t} \\ \vdots \\ \varepsilon_{m,t} \end{pmatrix}$$

or — for a single output load value:

$$O_t = (\tau_{i1},...,\tau_{in})I_t + \varepsilon_{i,t}.$$

The total traffic going from an input link to output links is the same as the link load itself, thus

$$I_{j,t} = \sum_{i=1}^{m} \tau_{ij} \cdot I_{j,t}$$

$$I_{j,t} = I_{j,t} \cdot \sum_{i=1}^{m} \tau_{ij}$$

$$1 = \sum_{i=1}^{m} \tau_{ij}$$

In other words, the total of the elements per column vector of T equals 1. Furthermore, given a well-behaving network domain without packet losses, the input load is equal to the output load, thus

$$\sum_{i=1}^{m} O_{i,t} = \sum_{j=1}^{n} I_{j,t}$$

If the assumption of a well-behaving network domain shows valid, these observations will help quantifying any measurement errors.

Using Knowledge of Traffic Forking

In the previous section only the load on the inbound links was known and we had to find out which outbound links were affected by this load by means of the transit matrix T. Alternatively, information about the "traffic forking", i.e. the distribution of traffic from an inbound link to the outbound links could be used. Gathering this information from the ingress nodes is not easy in all cases because this might require knowledge of intra-domain topology and routing. It is often far easier to determine how much of the load on an outbound link comes from a specific inbound link.

Instead of the input and output loads we would then know the load of each outbound link and the shares of the contributing inbound links, and the load going from inbound link j to outbound link i would be $\sigma_{ji} O_{i,t}$. Given the assumption "input load = output load" holds it follows that the load on a given inbound link j is

$$I_{j,t} = \sum_{i=1}^{m} \sigma_{ji} \cdot O_{i,t}$$

so the whole system can be written as

$$\begin{pmatrix} I_{1,t} \\ \vdots \\ I_{n,t} \end{pmatrix} = \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1m} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_{nm} \end{pmatrix} \begin{pmatrix} O_{1,t} \\ \vdots \\ O_{m,t} \end{pmatrix}$$

Again, the sum of elements of the column vectors equal 1, which can be shown analogously to above.

In a simulation scenario the given values are usually the input loads. A way to convert the above notation to the transit matrix notation is necessary to accommodate for that. As seen above the input load $I_{j,t}$ is the total of m terms of the form $\sigma_{ji} O_{i,t}$, for some j. The elements τ_{ij} of the transit matrix can thus be calculated by

$$\tau_{ij} = \frac{\sigma_{ji} O_{i,t}}{\sum_{k=1}^{m} \sigma_{jk} O_{k,t}}$$

or, if I_t is known, by

$$\tau_{ij} = \frac{\sigma_{ji} O_{i,t}}{I_{j,t}}$$

given that $I_t \neq 0$. Otherwise we can assume $\tau_{ij} = 0$. Again, the resulting transit matrix has the property that the element sum of the column vectors equals 1.

SLSs and Traffic Classes

The transit matrix model discussed only considers one load parameter per input or output link. However, there often are several load parameters for various traffic classes defined in service level agreements. Based on the assumption of a "well-behaving network domain" as mentioned above, these traffic classes can be considered as independent traffic aggregates. This allows us to describe multiple traffic classes by using one transit matrix per traffic class and domain. Instead of a single transit matrix T the domain model then uses the set of matrices $T_1,...,T_C$ where C is the number of separate traffic classes.

3.2.1.1.2 Matrix Adaptation

Network domains have constantly changing characteristics. Accordingly, a domain's traffic matrix also changes over time. While the calculation of transit matrices already requires a time series of input and output vectors, modelling the changes of domain behaviour requires a time series of traffic matrices.

From a time series of input and output load vectors I_t and O_t ($t = 0,...$) we thus have to calculate a time series of transit matrices T_u ($u = 0,...$). Let s be the number of observations required to get a good estimate of the momentary transit matrix of a domain. Then the calculation of matrix T_u is based on the vectors I_v and O_v where $v = u \cdot s,...,u \cdot (s+1) - 1$. I.e. we make s sized groups of input and output vectors and calculate a transit matrix from each. Using this time series of matrices we can predict future transit matrices with well-known prediction methods. The only problem is to make the predicted matrices conform to the properties described above.

3.2.1.2 Inter-Domain Link Model

When choosing an analytical model for inter-domain links one has to make the decision whether to use the traditional queuing theory approach or a fluid simulation approach. While the former may become difficult to calculate in the case of complicated SLSs, the latter may be expected to perform better in such situations but also to yield inferior exactitude.

3.2.1.2.1 Fluid Approach

Unlike domain models the inter-domain link model has only one load source, which can usually be described with a single scalar I_t. If multiple traffic classes have to be distinguished — in DiffServ environments for example — this parameter becomes $I_t = (I_{1,t},...,I_{C,t})$, where C is the number of different traffic classes. The vector elements each describe the bandwidth used by the traffic aggregate of a single traffic class. Models of multiple levels of complexity can be defined based on this input parameter and additional knowledge, such as priorities of traffic classes.

Simple Capacity Division

The simplest approach only takes the link bandwidth capacity B and a number of traffic classes with fixedly assigned bandwidth shares s_i into account. These shares are combined in the vector $S = (s_1,...,s_C)$ (the sum of elements is 1). The output load can them be written as

$$O_t(I_t) = (\min(I_{1,t}, s_1 B),..., \min(I_{C,t}, s_C B)).$$

The applicability of this approach is very limited since the interaction between multiple traffic classes cannot be modelled by it.

Traffc Class Model Tree

Interrelations between traffic classes can take various forms. Expedited Forwarding for example has absolute priority over other traffic classes, as long as the used capacity remains below a fixed value. A more complex example is Assured Forwarding with its three subclasses called dropping precedences, which influence themselves while the whole of them influences other traffic aggregates.

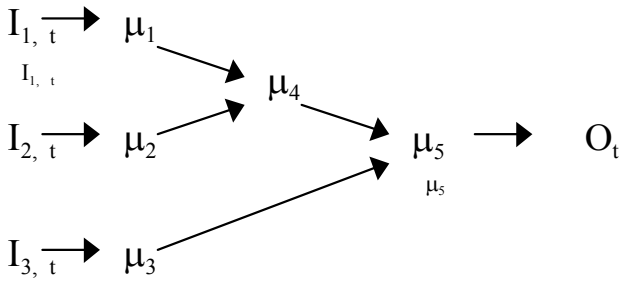

Figure 3.2 - Example of a model hierarchy

Using a model hierarchy these interrelations can easily be modelled. The leafs of this hierarchy tree model the behaviour of single traffic classes. Nodes higher in the hierarchy model the interrelations between multiple traffic aggregates below them. Figure 3.2 shows an example with three leaf nodes and one intermediate node, and of course one root node. In algebraic form the model would be

$$O_t = \mu(I_t) = \mu_5(\mu_4(\mu_1(I_{1,t}),\mu_2(I_{2,t})),\mu_3(I_{3,t}))$$

Below a few functions useful to construct such model hierarchies will be defined. In order to make the functions freely combinable they all have to be of a certain algebraic form, which is

$$\mu(B,\vec{P},\vec{M},\vec{A}).$$

Parameter B is the maximum available bandwidth, $\vec{P}$ is a vector of parameters specific to μ, $\vec{M}$ is a vector of n submodels and $\vec{A}$ is a vector of argument vectors for these submodels. The elements of $\vec{A}$ again have the form $(\vec{P},\vec{M},\vec{A})$, although with other vector sizes. The input bandwidths $I_{i,t}$ are considered as constant functions in this context. Furthermore, the following function will help readability:

$$m(b,\mu,\vec{a}) = \min(b,\mu(b,\vec{a}))$$

which according to the above algebraic form can also be written

$$M(B,(b_1,...,b_n),\vec{M},\vec{A}) = \begin{pmatrix} M(b_1,\mu_1,\vec{a}_1) \\ \vdots \\ M(b_n,\mu_n,\vec{a}_n) \end{pmatrix}$$

if and only if $B \geq \sum_{i=1}^{n} b_i$ holds.

Fixed Bandwidth Shares We can take a first basic function from 0. Incoming bandwidth from the models in $\vec{M}$ is simply truncated to a maximum bandwidth B according to bandwidth shares in $\vec{P} = (s_1,...,s_n)$. We get *fixed share* function

$$F(B,\vec{P},\vec{M},\vec{A}) = \begin{pmatrix} m(s_1 B, \mu_1, \vec{a}_1) \\ \vdots \\ m(s_n B, \mu_n, \vec{a}_n) \end{pmatrix}$$

Priority Multi-Queue Multi-queue systems often implement a system of fixed priorities where queues with small priorities are only allowed to send if all queues with higher priority are unable to. This property can be modelled by the function

$$P(B,(),\vec{M},\vec{A}) = \begin{pmatrix} m(\beta(1), \mu_1, \vec{a}_1) \\ \vdots \\ m(\beta(n), \mu_n, \vec{a}_n) \end{pmatrix}$$

The name P stands for the priority dependent behaviour of the function. The helper function β yields the bandwidth available to μ_i and is defined as

$$\beta(i) = \begin{cases} B, & i = 1 \\ B - \sum_{j=1}^{i-1} P_i, & otherwise \end{cases}$$

Dropping Precedences Assured Forwarding (AF) defines three dropping precedences — low, medium and high — which cannot be modelled with the functions above. AF packets are normally marked with low dropping precedence when they are sent and only change that status to a higher dropping precedence if they are detected to be “out of profile” by a router on their path. Then, in case of congestion, routers drop packet with higher dropping precedence first. The following functions models a generalized variant of this approach with an arbitrary number of dropping precedences:

$$D(B,(b_1,...,b_{n-1}),\vec{M},\vec{A}) = \begin{pmatrix} D_1 \\ \text{M} \\ D_n \end{pmatrix}$$

Note the parameters $b_1,...,b_{n-1}$: In contrast to the bandwidth share parameters above they stand for absolute bandwidths. Additionally we assume $b_1 \le B$. Again, we need helper functions to describe the elements of the result vector. First, function r calculates the amount of bandwidth of a given dropping precedence that has to be retagged to a higher dropping precedence:

$$h(i) = \begin{cases} \max(\mu_1(b_1, \vec{a}_1) - b_1, 0), & i = 1 \\ \max(\mu_i(b_i, \vec{a}_i) + h(i-1) - b_i, 0), & i > 1 \end{cases}$$

Using this function we can see how much bandwidth remains for a given dropping precedence aggregate by computing

$$r(i) = \begin{cases} m(b_1, \mu_1, \vec{a}_1), & i = 1 \\ \mu_n(\infty, \vec{a}_n) + h(n-1), & i = n \\ m(b_i, \mu_i, \vec{a}_i) + h(i-1), & otherwise \end{cases}$$

so we can finally write

$$D_i = \begin{cases} \min(r(1), B), & i = 1 \\ \min(r(i), B - \sum_{j=1}^{i-1} D_j, & otherwise \end{cases}$$

Fair Queuing Fair Queuing and Weighted Fair Queuing are very popular approaches to manage QoS. Our proposed system should therefore provide a rendering of their behaviour, or rather just of Weighted Fair Queuing since it is a generalization of Fair Queuing. The parameter vector has again

the form $\vec{P} = (s_1,...,s_n)$ here. If not all of the submodels completely use up their bandwidth share, the remaining bandwidth can be used by the other submodels, according to their share. The calculation of the resulting bandwidth shares is best done algorithmically. We need the following definition:

$$\alpha(i,b) = \begin{cases} 1, & \mu_i(b,\vec{a}_i) \geq b \\ 0, & otherwise \end{cases}$$

To begin let $\iota = \{1,...,n\}$, $W = 1$ and $b_1 = ... = b_n = 0$. The sum of the unused bandwidths of the single flows is calculated by

$$U = \sum_{i \in \iota} \alpha\left(i, \frac{s_i}{W} B + b_i\right)\left(\frac{s_i}{W} B + b_i - \mu_i\left(\frac{s_i}{W} B + b_i, \vec{a}_i\right)\right)$$

Then, for every $i \in \iota$, reassign

$$b_i = \begin{cases} \frac{s_i}{W} B + b_i, & \alpha\left(i, \frac{s_i}{W} B + b_i\right) = 1 \\ \mu_i\left(\frac{s_i}{W} B + b_i, \vec{a}_i\right) & otherwise \end{cases}$$

In the next round only the “unsatisfied” models participate. We reassign

$$\iota = \left\{ i : i \in \iota \wedge \alpha\left(i, \frac{s_i}{W} B + b_i\right) = 1 \right\}$$

The sum of weights must also be adjusted, thus we reassign

$$W = \sum_{i \in \iota} s_i$$

From here go to the calculation of U until ι is the empty set. The variables b_i then contain the bandwidths assigned to the models μ_i. The complete model is then given by

$$W(B, \vec{P}, \vec{M}, \vec{A}) = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

An Example An example will help clarifying the use of the above system: Consider an inter-domain link with the queuing system shown in Figure 3.3.

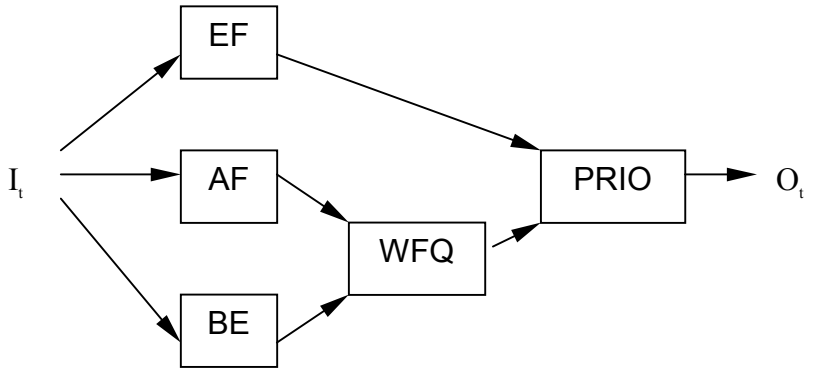

Figure 3.3 - Inter-Domain Link Example

Let the total link bandwidth be 100Mbit/s, and the bandwidths for the AF dropping precedences be 40Mbit/s, 8Mbit/s and 2Mbit/s, for low, medium and high dropping precedence, respectively. Further, let the WFQ weights be 0.5 for both inputs and 25Mbit/s be the maximum bandwidth allowed for EF traffic. Using the prototypical functions from above we get

$$M\left[\mathrm{X},(25),(I_{1,t}),()\right]$$

for the EF queue and

$$D\left[\mathrm{X},(40,8,2),(I_{2,t},I_{3,t},I_{4,t}),()\right]$$

for the AF queue (X will be replaced by the function higher in the hierarchy). The Best Effort queue simply uses as much bandwidth as it can get, so we can directly take $I_{5,t}$ as a model for it. Combining these we get

$$W\left[\mathrm{X},(0.5,0.5),(D,I_{5,t}),\vec{A}_W\right]$$

for the WFQ queue with

$$\vec{A}_W = [\underbrace{((40,8,2),(I_{2,t},I_{3,t},I_{4,t}),())}_{D},\underbrace{()}_{I_{5,t}}]$$

Using a P function to combine these, we get the final model

$$O_t = P[100,(),(M,W),\underbrace{(((25),(I_{1,t}),()),}_{M} A_W)]$$

3.2.1.2.2 Queuing Theory Approach

Traditionally, analytical network models have always been based on the queuing theory originating from operations research and the like. Although creating such a model for a given system is often non-trivial, the results are both accurate and efficient. On the other hand, larger systems become very complicated to model.

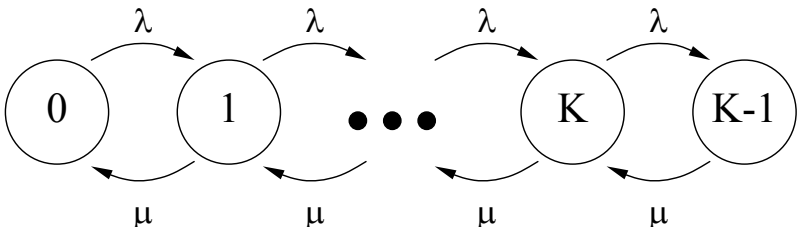

Figure 3.4 - Birth and death process

In the simple case of one queue per inter-domain link we can use a classic M/M/1/K queue, that is, a queue with exponentially distributed inter-arrival time τ and service time s, a single "processing station" (the physical link) and a system capacity K. The arrival and service rates are given by $\lambda = 1/E(\tau)$ and $\mu = 1/E(s)$, respectively. This system is a birth and death process as shown in Figure 3.4. For a birth and death process of this kind the probability p_i of the system to be in state i is given by

$$p_i = \begin{cases} \dfrac{1-\lambda/\mu}{1-(\lambda/\mu)^{K+1}}, & i = 0 \\ (\lambda/\mu)^i \, p_0, & i > 0 \end{cases}$$

if $\lambda \neq \mu$, and

$$p_0 = \ldots = p_K = \frac{1}{K+1}$$

if $\lambda = \mu$. Because here we are only concerned with the changes to the traffic load caused by the queuing system, there is now a very simple way to simulate the dropping behaviour. Arriving packets will only be dropped if the queue is full, which is the case with probability p_K. It is therefore sufficient to randomly drop a corresponding fraction of the arriving packets, or in the case of an input load parameter I_t to write

$$O_t = (1 - p_K) I_t$$

Assured Forwarding As mentioned above Assured Forwarding defines three dropping precedences. The differences in behaviour towards these precedences are usually implemented by beginning to drop packets at different fill levels of the queue. This can again be modelled by a birth and death process, although a more complicated one (see Figure 3.5). The arrival rate λ consists of three rates λ_l, λ_m and λ_h, for low, medium and high dropping precedence, respectively, with $\lambda = \lambda_l + \lambda_m + \lambda_h$.

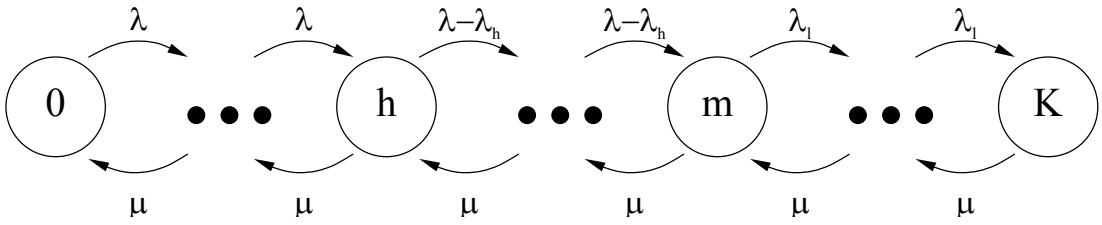

Figure 3.5 - AF birth and death process

The system capacity is again K. Medium precedence packets can only be queued if the system contains less than m packets, and high precedence packets only if it contains less than h. The system state probabilities for $i > 0$ are

$$p_i = \begin{cases} (\lambda/\mu)^i \, p_0, & 0 < i \le h \\ ((\lambda - \lambda_h)/\mu)^{i-h} \, p_h, & h < i \le m \\ (\lambda_t/\mu)^{i-m} \, p_m, & m < i \le K \end{cases}$$

State 0 consequently occurs with probability

$$p_0 = \left[\sum_{i=1}^{h} \left(\frac{\lambda}{\mu} \right)^i p_0 + \left(\frac{\lambda}{\mu} \right)^h \sum_{i=h+1}^{m} \left(\frac{\lambda - \lambda_h}{\mu} \right)^{i-h} p_0 + \left(\frac{\lambda}{\mu} \right)^h \left(\frac{\lambda - \lambda_h}{\mu} \right)^{m-h} \sum_{i=m+1}^{K} \left(\frac{\lambda_l}{\mu} \right)^{i-m} p_0 \right]$$

After some transformations and using the terms

$$A = \left(1 - \frac{\lambda}{\mu}\right), B = \left(1 - \frac{\lambda - \lambda_h}{\mu}\right), C = \left(1 - \frac{\lambda_l}{\mu}\right)$$

we can write

$$p_0 = ABC / \left[1 - \left(\frac{\lambda}{\mu}\right)^{h+1} BC - \left(\frac{\lambda}{\mu}\right)^{h} \left(\frac{\lambda - \lambda_h}{\mu}\right)^{m-h+1} AC - \left(\frac{\lambda}{\mu}\right)^{h} \left(\frac{\lambda - \lambda_h}{\mu}\right)^{m-h} \left(\frac{\lambda_l}{\mu}\right)^{K-m+1} AB\right]$$

Analogous to the simple queue above the output loads are then calculated using

$$O_{h,t} = I_{h,t} \cdot \sum_{i=0}^{h-1} p_i$$

$$O_{m,t} = I_{m,t} \cdot \sum_{i=0}^{m-1} p_i$$

$$O_{l,t} = I_{l,t} \cdot (1 - p_K)$$

Note that the probabilities used also change for every t. Due to the rather heavy calculations involved the above model is not suited to small sampling intervals.

Schedulers Queuing systems with multiple queues and a single outgoing interface need one or more schedulers to decide which queue is allowed to send when the interface is done sending a packet. Some of the most frequently used schedulers are the Weighted Fair Queuing and Priority schedulers.

To model WFQ we can almost immediately use the Fair Queuing approach from 0. Instead of nesting functions to determine the output loads of submodels we can directly use the service rates $s_i\mu$ ($i = 1,...,n$), and instead of the output bandwidth B we use a known service rate μ. Going through the algorithm yields the adjusted service rates for the queues. By recalculating the models with these rates we get the final output loads for every queue.

Priority schedulers can be modelled with a slightly modified version of the approach in 0. The system consists of n queues with arrival rates $\lambda_1,...,\lambda_n$ and service rates $\nu_1,...,\nu_n$. The service rates $\nu_1,...,\nu_{n-1}$ are fixed and have the property

$$\sum_{i=1}^{n} \nu_i \leq \mu$$

where μ is the service rate of the priority scheduler itself. ν_n is given by

$$\nu_n = 1 - \sum_{i=1}^{n} \nu_i .$$

The output loads of the queues $1,...,n-1$ does not change. That of queue n is obtained by evaluating it with service rate ν_n.

3.2.2 Delay Models

3.2.2.1 Multi-Domain Model

In accordance with the general modelling view, the delay model divides into a model for intra-domain delay and a model for inter-domain delay. Before having a closer look at their detailed structure we first study how they relate to each other.

The delay caused by a domain or and inter-domain link is not exactly the same for each of a series of consecutive packets of a flow. In fact, the delays of a series of packets can be more adequately modelled with probability distributions, rather than with a single value like mean delay. This is also the case for delay variations and the variations in packet interarrival times. As a consequence it is a

promising approach to study the delay behaviour of a packet stream of packets going through a series of network domains by adding the delay random variables of the domains and inter-domain links the stream goes through. Let the random variable of delay along a path P be

$$D^P = \sum_{i=1}^{n} D_i^P + \sum_{i=1}^{n-2} D_i^I$$

where D_i^D and D_i^I are the random delays caused by the n domains and $n-2$ inter-domain links along the path. The mean end-to-end delay of the path is then $E(D^P)$ and jitter is the mean interarrival time of two packets, given the distribution of D^P. It can be calculated by taking the difference of two random variables $d_1, d_2 \sim D^P$.

$$Jitter = E\left(\sqrt{(d_2 - d_1)^2}\right)$$

While the approach is very simple from the multi-domain viewpoint, finding fitting delay models for domains and inter-domain links requires more detailed analysis. In the following sections we will have a closer look at these details.

3.2.2.2 Domain Model

A well-known way of looking at end-to-end delay is to divide it into link, processing and queuing delay. These effects appear in both, domains and inter-domain links. It is a viable approach to build the delay models by deciding whether and how to model them.

When looking at the case of a path going through a domain it appears clear that all three effects are strongly dependent on the number of hops inside the domain involved. The following formula describes a very general model based on this assumption:

$$D_i^I = \sum_{j=1}^{m} l_j + p_j + q_j$$

Here, l_j are the link delays, p_j the processing delays and q_j the queuing delays of a hop. Figure 3.6 shows this graphically for the case $m = 3$.

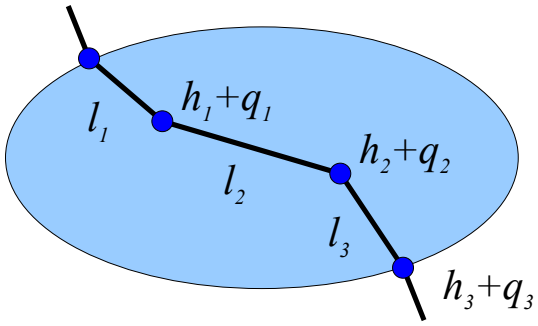

Figure 3.6 - General domain delay model

The link delays l_j are probably not equal to each other but all of them are constant. We can therefore replace these terms in the formula by the constant L. Processing delays on the other hand are not constant and often depend on small timing variations inside the routers. Experiments performed at the University of Bern have shown that processing delays are approximately poisson distributed.

Because we assume that network domains are well-behaving and congestion only occurs in inter-domain links, queuing delays can only be caused by the burstiness of traffic, which gets smaller the more “backbone characteristics” the domain has. Traffic in backbone domains tends to be smooth because of the great number of microflows it consists of. During validation we will try to show that queuing delays inside domains are really negligible or at least describable by simple means. If the above proves true the domain delay model can be reduced to the simpler formula

$$D_i^D = L_i \sum_{j=1}^{m} p_j$$

where L_i is constant and p_j are poisson distributed random variables.

It is important to note that the black box domain model does not allow to store delay characteristics per node in the domain. Each path between two edge nodes must have an own model, include hop count as well as link delay and processing delay characteristics. The adequate model for a path can then be selected by looking at routing information.

3.2.2.3 Inter-Domain Model

Inter-domain delays are mainly due to a fixed link delay L and a queuing delay Q. Obviously, an analytical queuing model is a natural approach to the latter. In the trivial case, a single Best-Effort queue, an M/M/1/K queue, can be used, with arrival and service rates derived from the average observed packet size S, the input load I and the output bandwidth $B$of the link. The system capacity K (i.e. the queue length plus one) can be set to a typical value if it is not known a priori.

Calculating the delay distribution for an M/M/1/K queue is rather easy: Let $\lambda = I/S$ and $\mu = B/S$ be the arrival and service rates. The system state probabilities are computed analogous to 3.2.1.2.2. When the system holds i packets, an arriving packet experiences a delay of i/μ. The delay distribution of the whole inter-domain link is thus

$$\begin{pmatrix} p_0 & p_1 & \cdots & p_K \\ L & 1/\mu + L & \cdots & K/\mu + L \end{pmatrix}$$

3.2.2.3.1 Service Differentiation

While the above model is simple and efficient it is not suited to areas where the routers differentiate between multiple classes of packets. Well known examples are the Expedited and Assured Forwarding services and schedulers like Round Robin, Weighted Fair Queuing and Priority Scheduling. In the paragraphs below we develop analytical delay models for some of these. Note that Expedited Forwarding is trivial to model using a queuing model as above, with fixed service rate.

Assured Forwarding The three dropping precedence levels defined by Assured Forwarding do not allow us to use the simple M/M/1/K queue from above. Nevertheless, by using the model defined in 3.2.1.2.2, a similar method is possible. Again, the variables $p_1,...,p_K$ designate the probability that the system contains i packets at a given moment in time, i.e. at the arrival of a new packet. Since calculating these probabilities is significantly more complex than in the M/M/1/K case we do not repeat the equations here. L and μ take the same roles as above, so the result is again the discrete distribution shown above, although with other probability values.

Priority Scheduler When multiple queues are combined using a priority scheduler the delay of packets in low priority queues in heavily influenced. To model this we define the following model:

A set of queues $Q_1,\ldots,Q_n$, all M/M/1/K queues with arrival rates λ_i and service rates μ_i is combined with a priority scheduler with service rate $\mu = \mu_1 + ... + \mu_n$. The service rates of the queues are controlled using a token bucket with a bucket size of one packet. The delays for packets in Q_1 are the same as in a single M/M/1/K queue. Packets in lower priority queues get the same queuing delay *plus* a delay when waiting for higher priority packets to leave the system. A packet in queue i must wait for a packet in queue *j* if and only if Q_j has a packet to send (probability $1 - p_0$, p_0 coming from the occupation distribution of Q_j) and is allowed to send by its token bucket, which is true in μ_i/μ cases.

The random variables $s_1,\ldots,s_n$ give the number of packets a queue can send at a given moment

$$s_i = \begin{cases} 1, & with\ probability\ (1 - p_{i,0})\frac{\mu_i}{\mu} \\ 0, & otherwise \end{cases}.$$

Let now

$$S_i = \sum_{j=1}^{i} s_j .$$

The delay of packet entering queue i is consequently the queuing delay plus S_i / μ.

3.2.3 Multi-Domain Models

All the models defined so far can and will be used as single entities in a simulator. Routing of traffic along the domains and inter-domain links is straightforward in this case: The routing information used to connect the model topology can directly be used by the simulator to determine the path a packet will take. This information may be derived from the BGP tables of the network to be simulated, for example.

It may be more efficient to combine them into a single entity from the simulator's point of view, however. Such a multi-domain model cannot use the simulator's routing capabilities and therefore needs knowledge about the routing inside the modelled network area. It appears reasonable to assume that the number of connections from and to the multi-domain model is "small", i.e. the complexity of storing all possible paths is manageable. Given n links to the "outside" the system would have to store $n(n-1)/2$ paths. These paths are uniquely identified by the affected egress links of the first to the second last domain. This approach is also applicable to the models concerned with delay, jitter, etc.

A multi-domain model is thus defined by the definitions of the included domain and inter-domain link models and a table of paths between all links connected to the "outside" of the model.

When used in a simulator, multi-domain models yield data about the input and output bandwidths of certain paths and probability distributions of delay. How can these be used?

3.3 Reduced, Realistic Model

3.3.1.1 Introduction

The aim of this contribution is to describe what is and what is not possible in real life networks and the impact on delay models. The design of a network model is necessary for Modelling and Simulation Toolkit included in the InterMON Platform. In this contribution, we argue that the model can be simplified to domains and inter-domains links. Obtaining data from ISP's is complicated, because there are no useful models (from ISP's point of view) and deriving models from network data is nearly impossible, because of the lack of real life network data. We aim to break this chicken and egg situation with this simplified model. Once and if it proves to be useful, we expect ISP's to publish data which should help enhance this model.

We have to consider the delay in the domains and in the links between them. In this context a domain consists of a number of nodes of different types (probably we won't know this information). Some of this nodes will are edge nodes and we will try to guess some useful information through this elements.

We have to assume that it will be difficult to get from ISP's its intra-domain topology information (i.e. nodes, links and technological information about both). Moreover there is to take into account that this topology can to change in the future. For this reasons InterMON have to be able to define a network model as independent of this matters as possible.

3.3.1.2 Delay Model

The intra-domain delays are private data and for this reasons we have to obtain it by measuring without the collaboration of the ISP. A way to collect this data for the network model is using the traceroute process.

Traceroute, though, has several drawbacks: Some nodes do not respond to traceroute packets:

- Many Internet exchange points use Layer 2 technology and traceroute is Layer 3 specific
- Many ISP's have MPLS backbones and response to traceroute packets is optional in this case
- Traceroute response might be intentionally blocked because of
  - Security holes in the node's OS
  - A network policy obscuring the network topology
- Traffic engineering techniques applied in the network may force packets to follow several alternate paths

These reasons favour the 'black box' approach of InterMON.

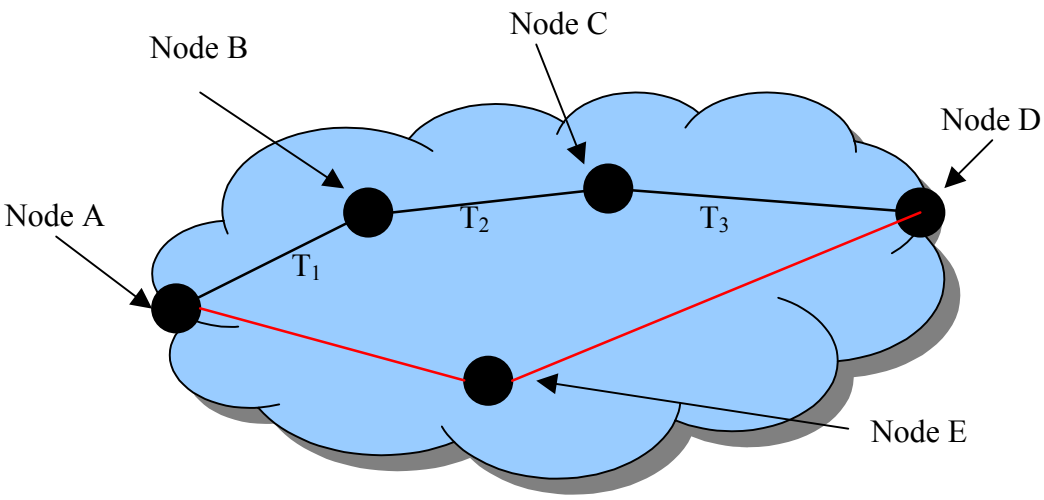

Figure 3.7: Intra-domain delay

Traceroute or similar methods condense the delay in one single point we identify as 'the node'. Lines can be then made 'ideal', i.e. entities with null delay. In Figure 3.7 we assume that the delay T_1 resides in node B, the delay T_2 resides in node C and the delay T_3 resides in node D, respectively. This is an intuitive approach derived directly from the output of the traceroute command:

nodeA$ *traceroute <Node D>*

```
traceroute to <Node D> (d.d.d.d), 30 hops max, 40 byte packets
 1  nodeB.domain (b.b.b.b)  1.211 ms  1.053 ms  0.976 ms
 2    * * *
 3  nodeD.domain (d.d.d.d)  4.070 ms  2.589 ms  2.777 ms
```

This way, we become independent to the ISP and the technology used in the intra-domain and inter-domain links, something fundamental for the delay model.

3.3.1.3 Working Thesis

There are two essential working thesis we have to take into account for the models desing. The first has relationship with the previous matter. We want to get a network model that not depend on the transmission technology. The model consists of a node which has a delay (can be estimated with traceroute or equivalent tools) and links between nodes in the network with null delay.

With this approach, each ISP can be modelled like a node separate and we can forgot the links between them, from the view point of the delay, because in this links we have null delay.

The second work thesis considered from the start of this project consists on consider each Autonomous System like a “Black Box”. That idea is just the proposed delay model try to get. Each ISP was a “box” with a delay equal to the sum of the delays of all its nodes.

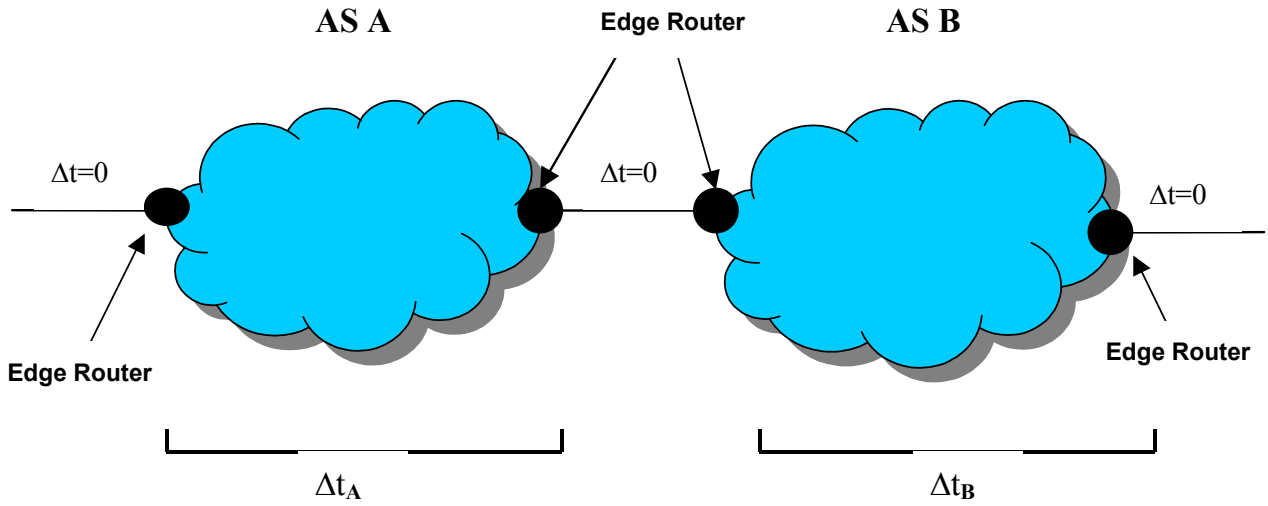

Figure 3.8 - Inter-domain delay

The advantage of this model is that we only need the IP address of the edge router since we can measure the other necessary parameters and derive from them models and trends for the Modelling and Simulation Toolkit included in the InterMON Platform.

3.3.1.4 Conclusion

We have presented a simplified delay model which makes the InterMON system technology independent, thus overcoming several limitations found in today's networks. The only restraints we impose for this model is that the edge or border devices are layer 3 devices, i.e. routers, which respond to the probing method we select (i.e. traceroute) to measure inter-node delays.

We hope to derive first behavioural models for autonomous systems from this approach. Proving the usefulness of these models should trigger a process, whereby ISP's recognise the value of the InterMON approach and start publishing other network data (i.e. interdomain traffic). This would break the current chicken and egg situation, whereby no data are published because this doesn't have any direct return and hence no models can be built because there are no data to confirm and refine them.

3.4 Application Traffic Models

3.4.1 Voice over IP

3.4.1.1 Single Call Analysis

The voice over IP communication includes a wide variety of available protocols. The most common used standard for bi-directional communication on non-guaranteed networks is given by the H.323 (ITU). The newer SIP (Session Initiation) protocol is introduced by the IETF.

The H.323 is called an umbrella standard, because it covers transport of multimedial data (refer to Figure 3.9)

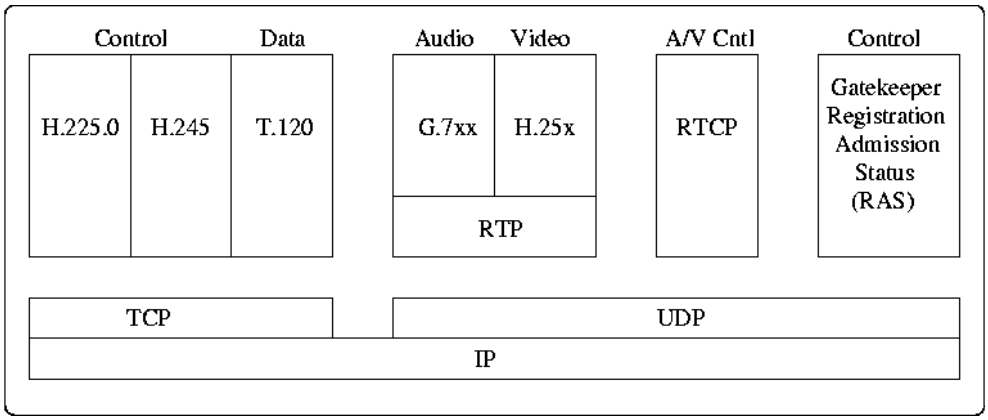

Figure 3.9 - H.323 protocol stack

For simulate voice over IP traffic, the most important parameters are the different audio codecs. Available codecs are:

Algorithm	Description	Voice BW (kb/s)	Packets per sec	Total BW on Ethernet (kb/s)
G.711	Pulse code modulation of voice frequencies	64	100	107.2
G.723.1	Dual rate speed coders multimedia communication transmitting at 5.3kb/s	5.3	22	14.8
G.723.1	Dual rate speed coders multimedia communication transmitting at 6.3kb/s	6.3	26	17.6
G.728	Coding of speech at 16 kb/s using low delay code excited linear prediction	16	n.a.	n.a.
G.729	Coding of Speech at 8 kb/s using conjugate-structure algebraic-code-excite linear-prediction	8	50	29.6

Table 3.1 - Available codecs in H.323

To modulate this voice over IP traffic within an ON-OFF flow model the following considerations are needed:

- Equidistant packets
- All packets have the same size

Therefore there is only one ON-process per call with an flow-rate \alpha of the codec bandwidth.

3.4.1.2 Voice Activity Detection (VAD)

Voice activity detection was created for a better bandwidth utilisation. Gap periods within the voice stream will not be transmitted to avoid sending empty packages. There are two main implementation of VAD within the VoIP traffic: G.729B and NeVoT. Both behave in a slightly similar way, so that there is only G.729B is mentioned from now on.

[Jia00] analyses VAD influence on traffic aggregation and shows test with many different parameters like hangover and threshold. By this traces it's getting obvious that traditional exponential distributions for gaps and spurts do not always fit well with real traces.

[Bel02] stated, that Gamma distribution fits better for the ON-period and Weibull distribution for the OFF-period. Additionally they found out, that in some real life traces the gabs follow the Pareto distribution of high variability.

Parameter	Mean	Typical Deviation	PDF
On Period	0.4011	0.3637	Gamma
OFF Period	0.5775	1.1774	Weibull

Table 3.2 - Analysis of real VoIP traces

3.4.1.3 Traffic Analysis

For simulation of traffic trunks, the call scenarios described in the last paragraph needed to be aggregated. These aggregations were described in [Cis02] and should be discussed here.

Traffic Model

To give a short introduction to traffic models, some terms needed to be specified:

The *call hold time* is the time of a phone call and is often specified as T_H. The *call (inter-)arrival time* T_A is the time duration between two phone calls. The number of sources is often only divided into finite and infinite. One source was considered in the previous section.

The call hold time is not examined for special derivation functions normally, because it has proven that it will have a negative exponential distribution in almost all cases.

Some different traffic patterns should be discussed here, which are mainly divided by the inter-arrival time:

- **Hypo-exponential traffic pattern**

  The hypo-exponential traffic is also called smooth arrival pattern and occurs where there is not a great amount of variation in the traffic. The hold time and inter-arrival time is easily predictable. Example: a Call-Centre expects 600 calls with a duration of 2 minutes an hour.

$$X = Hypo(n) = \sum_{i=1}^{n} X_i$$

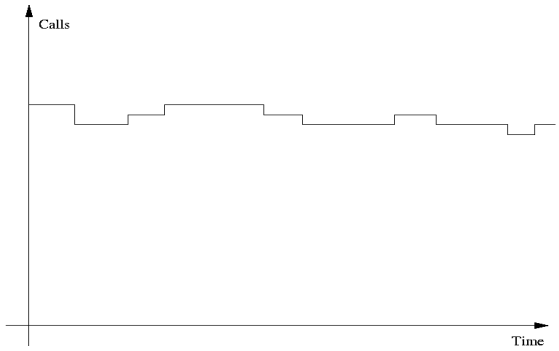

Figure 3.10 - Hypo-exponential traffic pattern

- **Hyperexponential traffic pattern**

  The hyperexponential traffic is also called peak arrival pattern. The variation within a peak arrival pattern is very high. Example of heavy peak traffic is Christmas day and New Year's eve.

$$f_x = f_{Hn} = \sum_{i=1}^{n} p_i \cdot f_{E\lambda i}$$

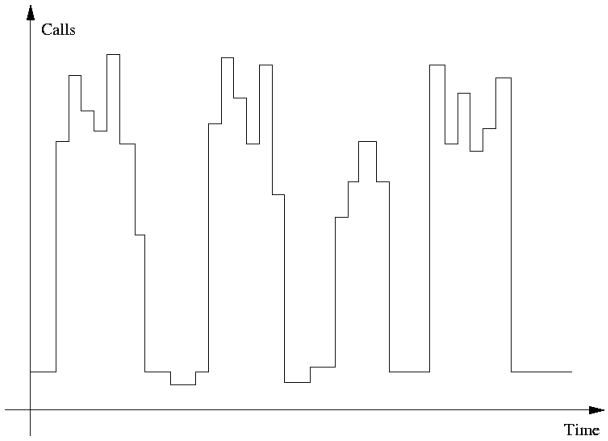

Figure 3.11 - Hyperexponential traffic pattern

- **Random traffic pattern**

  The random traffic pattern is also known as Poisson or exponential distribution. This distribution is generally seen in private branch exchange (PBX) environment with circuits varying from 1 to 30.

$$F_{E_\lambda}(x) = 1 - e^{-\lambda x}$$

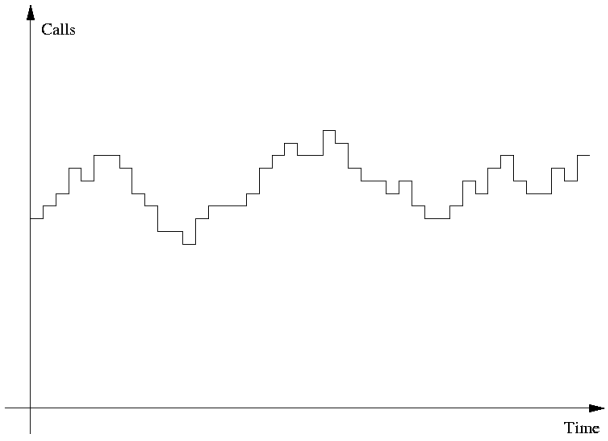

Figure 3.12 - Random traffic pattern

Especially in VoIP environments based on class of service aggregations, there is no mechanism to *block calls*. Therefore, all block call estimations don't work in this environment. How to handle congested (VoIP-)connection trunks is out of scope of this section.

Sampling Methods

The International Telecommunication Union Telecommunication Standardization Sector (ITU-T) makes recommendations on how to accurately sample a network to dimension it properly:

- Measurement / read out periods be 60 minutes and/or 15 minutes intervals
- Finding peaks with

- Daily peak period (DPP)
- Fixed Daily Measurement Interval (FDMI)

- Dividing daily measurements into groups: workdays, weekend days and yearly exceptional days
- Normal load traffic intensity for the month is defined as the fourth highest daily peak traffic (ITU-T recommendation E.492)

3.4.2 Video Streaming

3.4.2.1 Study of the Traffic Associated with Streaming Content

According to a recent industry study [EITO02] the ongoing deployment of an array of broadband last mile access technologies (such as DSL, cable and high speed wireless links) will ensure that a growing segment of population will have sufficient bandwidth to receive streaming video and audio in the near future. However, due to the high bandwidth requirements and the long-lived nature of digital video, server and network bandwidths are providing to be major limiting factors. (Audio and video files tend to be large in size, e.g. 4.8MB for a five minutes ling 128 Kbps MP3 audio clip, 450MB for a two-hour long MPEG-4 video clip encoded at 500 Kbps.)

[MSK] analyzes the traffic along a number of dimensions using streaming logs from a commercial service. The results can be summarized as it follows:

- Requests for Windows Media dominate those for Real where content is available in both formats. This relative ratio is quite stable across the different months. It suggests a widespread prevalence and use across the Internet of the Windows Media software.
- Requests for content encoded at a higher bitrate are dominant where high and low encodings are available. The statistics seem to indicate that a large majority of requests for streaming content are sourced by clients with good end-to-end broadband connectivity.
- Both Windows Media and RealNetworks recommended that the video be streamed using their respective proprietary streaming protocols running preferably over UDP. To overcome firewall restrictions, the protocol can also run over TCP. There is also the option to stream the clip using standard HTTP or some variant of it. [MSK] states that sessions using transport protocols running over TCP dominate those using UDP. This seems to indicate that even for high bandwidth streams the quality of media streamed using TCP is considered good enough by a large proportion of the end-users.
- Request and traffic volumes are highly skewed at different levels of aggregation (IP address, routing prefix and AS).
- For a tier-1 ISP a significant percentage of streaming clients are within two AS-hops of the ISP.
- Selective arrangements with a modest number of consistently high contributing ASs yield significant gain in improving coverage to streaming clients.
- Streaming traffic exhibits regular daily patterns with very high variability in terms of request, traffic volume and concurrent number of connections.
- Ramp up to daily peaks can be gradual over several hours or very sudden over tens of minutes.
- Streaming traffic exhibit very high variability in terms of daily peaks.
- A small number of streaming objects is responsible for significant portions of the request and traffic volume.
- Where the same content is encoded in high and low bitrates, the higher bitrate clients tend to watch more of the content.

3.4.2.2 Picture (Frame) Coding

The following picture (also called frame) coding methods are available:

- Intracoded
- Predictive
- Bidirectional

Intracoded (**I**) pictures are encoded without using any information from other pictures. Blocks of 8x8 pixels are transformed into frequency domain using two-dimensional Discrete Cosine Transform (DCT). Each element of the resulting 8x8 matrix is then quantized by the corresponding value of the 8x8 quantiser matrix. DC coefficients are then coded with a Differential Pulse Code Modulation in which only the difference from the previous DC coefficient is stored. These differences are stored using a Variable Length Code. AC coefficients are Run Length Coded, taking advantage of the fact that most AC coefficients are zero. 4 8x8 blocks constitute a 16x16 macroblock. I pictures are built up of 16x16 macroblocks, called Intracoded (**I**) blocks.

Temporal compression is achieved by using reference blocks. A 16x16 pel area is matched against a past or future block. If the block is found on the reference picture than only its motion vector is sent. If not an exact, but only a close match is found than the difference between the two blocks and the motion vector is encoded.

Reference blocks can be chosen from past frames, future frames, or mathematical combination (usually the average) from both a future and a past frame. There are several algorithms to search the reference block. This search is the most time-consuming process of the encoding.

Predictive (**P**) frames use only intracoded (**I**) blocks, or blocks encoded using blocks from past frames, called **P** blocks. Bidirectional (**B**) frames may contain any kind of blocks. Using **B** frames even better compression can be achieved, but on the other hand it introduces extra delay. **B** frames can only be encoded and transmitted after the succeeding **I** or **P** frame is encoded. (The expression **I**/**P** frame will stand for either **I** or **P** frame.)

We have to distinguish between the **order of display** and the **order of transmission**. Since **B** frames may only be encoded/decoded after the succeeding **I**/**P** frame, it would be illogical to send the **B** frames before the corresponding **I**/**P** frame. So if the order of display is for example: IBBPBBPBBI, then the order of transmission is: IPBBPBBIBB. The usage of B frames introduces extra delay, so they are usually not used for real-time video conferencing. In this case the order of display is the same as the order of transmission.

The three types of pictures follow each other in given periodic structure. (e.g. IBBPBB). One period is called a Group of Pictures (GOP). The sequence describing the periodic structure is called the GOP pattern.

The periodically appearing **I** frames increase the size of compressed video, but their usage can not be avoided because of the following reasons:

1. resynchronization after a failure or watching a movie from the middle would be impossible
2. backward, fast backward and fast forward playing would be extremely difficult without their usage.

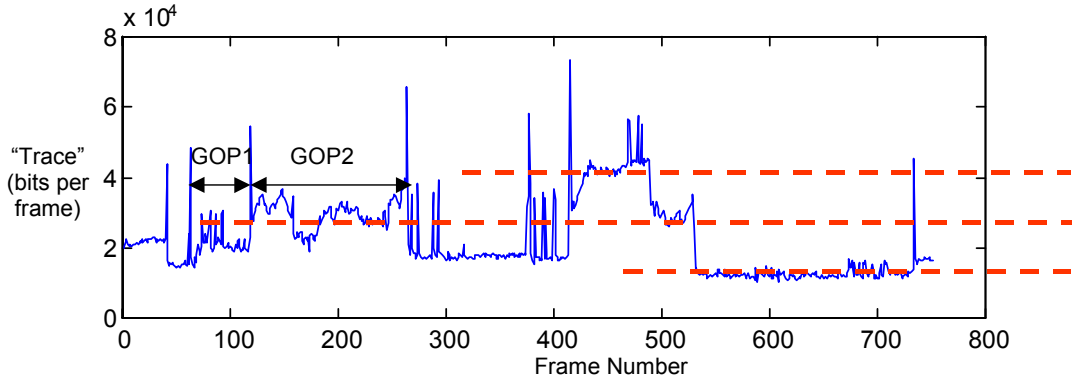

Figure 3.13 - Video traffic characteristic

The traffic characteristic of a video is illustrated on Figure 3.13. It is clear, that different activity levels can be distinguished. The statistic of video traffic can be seen on Figure 3.14.

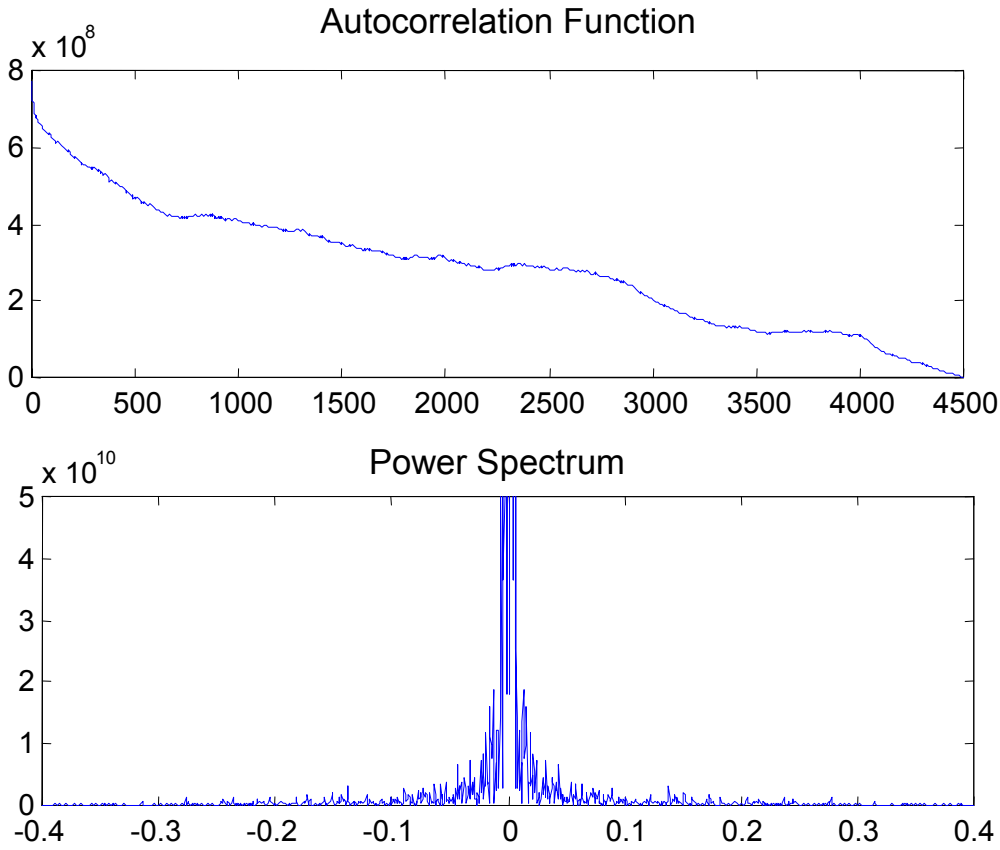

Figure 3.14: Statistic of video traffic

3.4.2.3 Survey of Video Models

The video models published up to now are usually built by the following way: one specific trace of a video source is investigated, then a model is given, which models that particular source. The model is verified by comparing the results gained by applying the original source and the model to the same network.

3.4.2.3.1 Autoregressive Methods

The main disadvantage of non-statistical video modelling techniques (such as [Xio98] and [Lia01]) are usually more difficult to analyze. This is not valid for autoregressive methods. However, the main drawback of this model is that it does not capture the dynamic nature of video traffic nicely.

Interleaved Autoregressive Model

A conventional interleaved autoregressive process, x_n can be formalized as follows:

$$x_n = \rho x_{n-1} + \sigma\sqrt{1-\rho^2}e_n \qquad \text{n=1,2,…}$$

where n is the time index of the process, ρ describes the dependency of the sample at time n with the previous sample, σ^2 is the variance of the process, x_n and e_n is a Gaussian random variable with mean 0 and variance 1 to characterize the random nature of the process.

This method does not take into account the timing information in both the training and synthesis stages. To illustrate this, consider the case where more than one AR processes are interleaved together. At time instant 1 AR process x_n takes place, at time instant 2 AR process y_n takes place, and so on. The method to train two sets of AR parameters of sequences x_n and y_n is by splitting the single process to two separate processes. Each one of the processes represents the training sequence of x_n or y_n regardless of the time index associated with each sample.

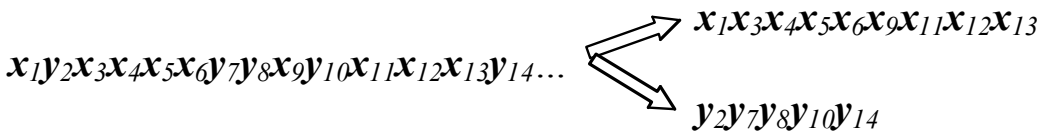

Figure 3.15 - Two interleaved autoregressive processes – autoregressive model

Interleaved Punctured Autoregressive Model

To consider the timing information missing from the conventional interleaved autoregressive model, [Che02] proposes to train and synthesize the AR processes as follows. First split the single process to two separate processes. Notice that the timing information is utilized while leaving the sample of x_n blank is at some particular time instance the original process is with the other process. It is illustrated on Figure 3.16.

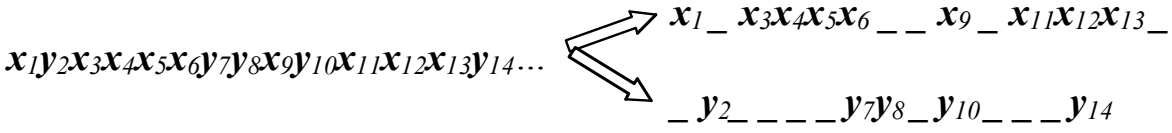

Figure 3.16 - Two interleaved autoregressive processes – punctured autoregressive model

3.4.2.3.2 Markov-Chain Based Models

In general, it is preferred to use a Markov chain like process to model the dynamic nature of the video traffic.

Simple Markov Chain Model

We define a Markov chain with *N* states. A given value is associated with each state of the Markov chain. In each state the corresponding value is the output of the model. The number of states, transition probability matrix and the corresponding values are calculated from the original source.

[Ros95] gives a model for generating GOP size traces. It suggests to set N to $\frac{G_{max}}{\sigma_G}$, where G_{max} is the maximum size of GOPs and σ_G is the standard deviation of GOP sizes. Each state represents an interval. The corresponding value is the mean value and the transition probabilities are calculated by the following equation:

$$P_{ij} = \frac{n_{ij}}{n_i}$$

where P_{ij} is the transition probability from state *i* to state *j*, n_{ij} is the number of transitions in the original source form interval *i* to interval *j* and n_i is the number of GOPs belonging to interval *i*.

Using Markov chain model correlation from one value to the next is introduced, but no direct correlation over larger lags. This results in an exponentially decaying autocorrelation function.

Doubly Markov Modulated AR Process

The simple Markov chain models (such as [Kru98]) are too complex.

[Tur01] describes the conventional doubly Markov modulated AR process, which models the variable bit rate video traffic as a doubly Markov process with AR processes inside each Markov state. It is detailed below.

The model comprises of two layers of Markov process. Without loss of generality, one can consider two frame types, I and P. The doubly Markov process models I and P frame transitions, as well as different frame activities. The outer Markov process describes how I and P frames transit. The frame type can be further categorized into different activity levels. Frames of higher activity level consume more number of bits, while frames of lower activity level consume less number of bits. The inner Markov process describes how the frames of different activity levels transit.

This model is not constrained by a fixed GOP structure. There are six states in total, namely, I frame in high activity level, I frame in medium activity level, I frame in low activity level, P frame in high activity level, P frame in medium activity level, and P frame in low activity level. Each state is modeled as an AR process with different AR parameters. It is illustrated on Figure 3.17.

This model does not use the timing information between frames of the same Markov state. It leads to periodicity in the power spectrum, as it can be seen on Figure 3.18.

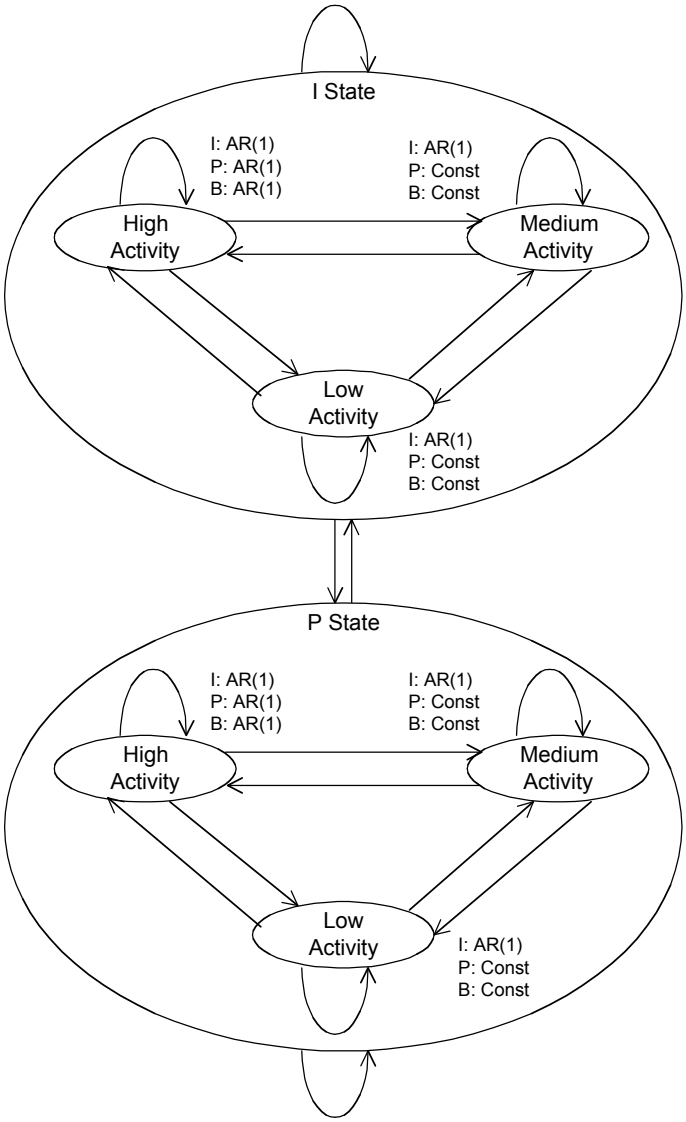

Figure 3.17 - Doubly modulated AR process

Doubly Markov Modulated Punctured AR Process

[Che02] describes the doubly Markov modulated punctured AR process. It is detailed below.

To simply the model, the inner Markov process of I frames is characterized by initial probabilities of three activity levels only. Since I frames are usually far apart in a video sequence, they do not need to modeled as a Markov process. Such simplification will have similar performance as the full model.

They propose to model the VBR video traffic as a doubly Markov modulated punctured AR process. This new model explicitly considers the timing information between two frames of the same state. The AR parameters are trained in the way punctured manner. It is illustrated on Figure 3.20.

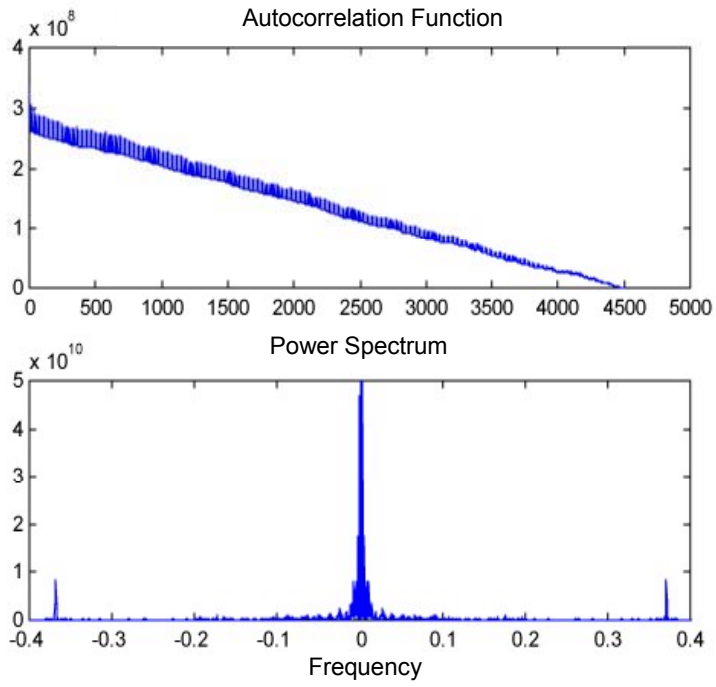

Figure 3.18 - Statistic of doubly markov modulated AR process

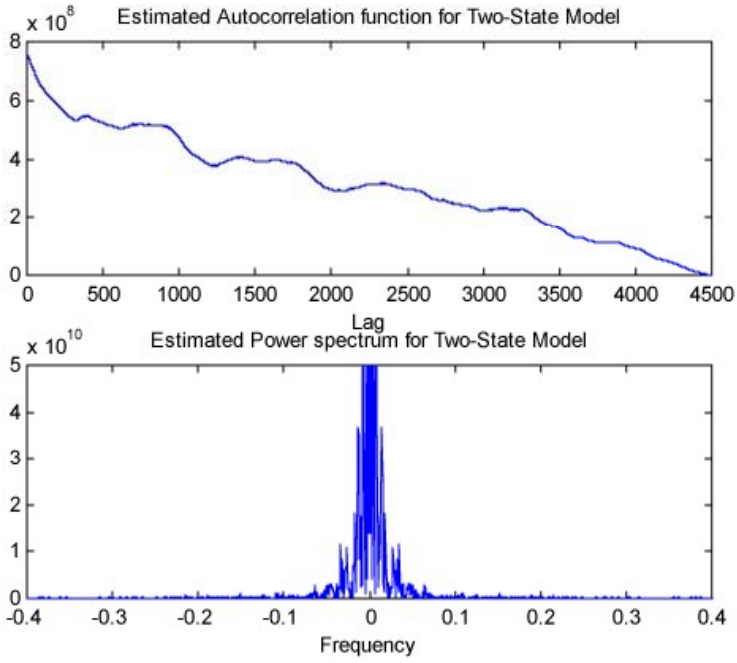

Figure 3.19 -Statistic of doubly markov modulated punctured AR process

3.4.3 HTTP Source Modelling

HTTP is the protocol used by the WWW service. Due to its relevance, it's important to study, within the InterMON modelling and simulation work package, what are the most established models in the literature for this type of traffic, and which model implementations already exist or need to be developed from scratch.

We recall that HTTP is a client-server protocol. The most common functioning consists of a user issuing through its client browser a "Get" request towards a server, specifying the resource he wants to download. Typically, the requested resource is a web page, uniquely identified by an URL (Universal Resource Locator). The server sends a response, which typically is the page main body. This first response normally is followed by a parsing activity on the client side, and a set of subsequent requests with which the client browser completes the download of all the objects "embedded" in the page main body. An example of an object is a graphic file.

The underlying communication protocol to exchange HTTP requests, responses and associated data is TCP. The number and the way TCP connections are open between the client and the server depends on the HTTP protocol version and on the browser implementation. For the InterMON simulator, however, it's not necessary to dwell into this level of detail, also because most of the models already available in existing simulator (for instance, in ns-2) already take care of this.

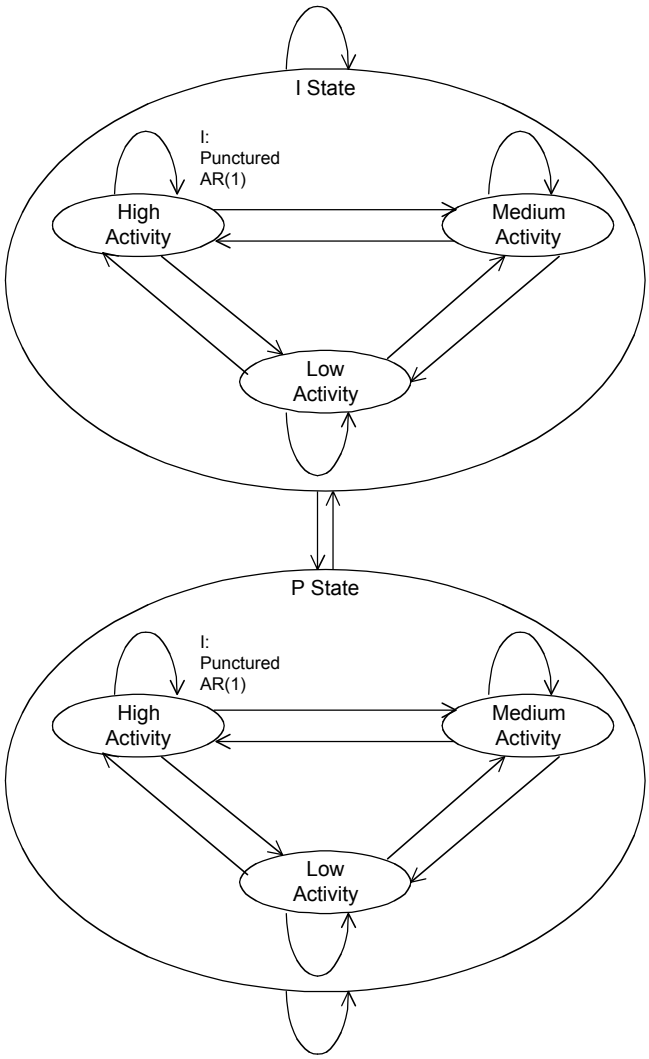

Figure 3.20 - Doubly markov modulated punctured AR process

What is important, on the contrary, is to understand that an HTTP interaction is characterized by "human" activities (basically, download requests) that immediately trigger the automatic download of the requested information. For a high level modelling, it's thus necessary at least to model the process of the request arrival, and the size of the downloaded "page", where page is in a broad sense the page main body and its the set of embedded object. Modelling the size of the requests is less important, being the traffic heavily asymmetric (in terms of volume, [Mol02] reports a ratio of 1/13 between the cli->ser and the ser->cli traffic).

In the following, we briefly explain what are the fundamental parameters that must be instantiated in such a model, considering two possible cases: a single source modelling or an aggregate source modelling.

3.4.3.1 Single HTTP Source Modelling

When considering a single source, i.e. a single browser<->web server interaction, it's important to understand that in a web browsing session, the time the user waits to issue a new requests normally starts from the time the previous page download is completed. Then, as regards the request process, all what needs to be modelled is the so called "think time" between the termination of a page download and the beginning of a new one. [Mol02] proposes, based on experimental observations, a Lognormal distribution as the best approximation for the think times. For the page size, also a Lognormal distribution results as the best fit. The Lognormal distribution belongs to the cathegory of the so-called "hevy tailed distributions" (i.e. very "big" page sizes are not unfrequent...). A lognormal distribution is also the best fit for the size of the "Get" requests. The model is summarized in the following picture (get "sizes" are not reported as they're much smaller than page sizes, as explained before). It's important to note that a model based on the "think time" doesn't depend from the time needed to download the pages. This allows to safely apply the same model, with the same parameters, to very different network scenarios where the bandwidth availability may heavily change the page download time, or when the bandwidth availability that the user experieces during a download, as shown in the picture, varies over time.

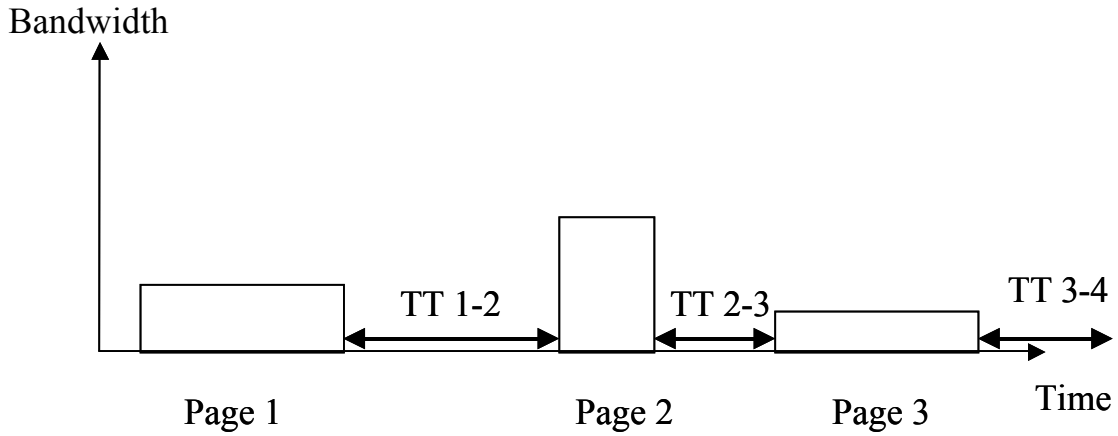

Figure 3.21 - http source model of a single client server couple

The experimental values reported in [Mol02] for the average of the mentioned parameters are:

- Think time: 61.35s
- Page size: 30 Kbytes
- Get (http request) size: 2.6 Kbytes

3.4.3.2 Aggregate HTTP Source Modelling

The model previously described can be straightforwardly used to model a multitude of client/server couples. However, as the number of clients increases, the aggregate page request process (i.e. the time superposition of the requests coming from all the users) tends to a Poisson process. This is suggested by the statistical theory and confirmed by experimental observations. There isn't a precise threshold beyond which the "Poissonian" hypothesis starts to hold, but in [Mol02], for example, a good match with a Poisson distribution was observed with an average number of active web users ranging between 100 and 200. The size of pages and of the get requests of course doesn't change with the single source model. An advantage of this model is that if the "page" is considered as an atomic object (i.e. the fragmentation in objects is neglected, etc), in many cases of interest the queuing theory can be applied and predictions can be extracted with the use of a simple M/G/1 model.

A model similar to this one (i.e. Poisson page request process and heavy tailed page sizes) is available in the standard ns-2 library. The main limit of this model is its scalability (i.e. it cannot be used to reliably simulate user populations greater than around 30).

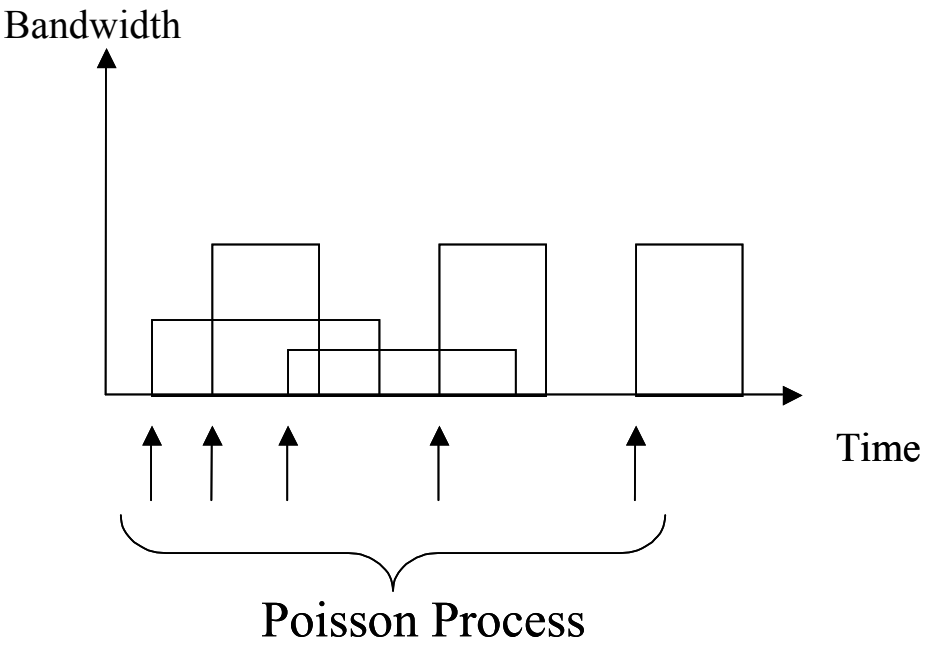

Figure 3.22 - http source model of a large number of client server couples

3.5 Time Series and ARIMA Models

3.5.1 Second-Order Summaries

The theory of time series is based on the assumption of second-order stationarity.

Let X_t be a stationary time series with mean μ and variance σ_x^2. The autocovariance function of X_t at lag *k* is defined as

$$\gamma(k) = E\{(X_t - \mu)(X_{t+k} - \mu)\}.$$

The autocorrelation function at lag *k* is defined as

$$\rho(k) = \frac{\gamma(k)}{\gamma(0)},$$

and is simply a standardized version of the autocovariance function.

The autocovariance function estimate at lag *k* is:

$$\hat{\gamma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}(x_t - \bar{x})(x_{t+k} - \bar{x}), \quad \text{(xx)}$$

where

$$\bar{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

is the mean of time series and *n* is the length of the observed series. (Notice that the divisor *n* is used, even though there are only *n-k* terms!)

3.5.2 AR(p) Models

Autoregression provides a good approximate (linear) model which has the virtue of extreme simplicity. However, one should be careful not to insist on using an AR model where, for example, an MA component is needed, nonstationarity must be dealt with, or a nonlinear model is needed.

Let ε_t dote a series of uncorrelated random variables with mean zero and variance σ_ε^2. An autoregressive process of order *p* (AR(*p*)) is defined by

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \varepsilon_t \, .$$

Note that not all values of the autoregression coefficients $\alpha_1,\ldots,\alpha_p$ result in a stationary process.[4]

3.5.2.1 The Yule-Walker Equations

Let $\gamma(k)$ be the autocovariance function for the AR(p) process. Then the coefficients $\alpha_1,\ldots,\alpha_p$ satisfy the Yule-Walker equations

$$\sum_{k=1}^{p} \gamma(k-i)\alpha_k = \gamma(i), \quad i = 1,2,\ldots,p \qquad \text{(x)}$$

In addition,

$$\sigma_x^2 = \sum_{k=1}^{p} \gamma(k)\alpha_k + \sigma_\varepsilon^2 \, .\text{(xxx)}$$

There is a natural way to obtain estimates $\hat{\alpha}_1,\hat{\alpha}_2,\ldots,\hat{\alpha}_p$ based on a finite sample $x_1,x_2,\ldots,x_n$. Namely, replace the $\gamma(k)$ in (x) and (xxx) by the estimates of (xx).

3.5.2.2 Order Selection

In practice, the order of the autoregression is not known. Hence, we wish to solve the sample-based Yule-Walker equations for a variety of values of *p* from 1 up to p_{max}, where p_{max} is sometimes 10 or even 15 (or even larger).

A way of selecting the order of the AR process is to find an order that balances the reduction of estimated error variance with the number of parameters being fit. One such measure is Akaike's Information Criterion (AIC). For an order *k* model, this criterion can be written as

$$AIC(k) = n\log\left(\hat{\sigma}_{\varepsilon,k}^2\right) + 2k \, .$$

If the series is an AR process, then the value of *k* which minimizes AIC(*k*) is an estimate of the order of the autoregression.

3.5.3 MA(q) Models

Let ε_t dote a series of uncorrelated random variables with mean zero and variance σ_ε^2. A moving average process of order *q* (MA(*q*)) is defined by

$$X_t = \sum_{j=0}^{q} \beta_j \varepsilon_{t-j} \, ,$$

[4] The condition for stationarity is that the (complex) roots of $\Phi(z) = 1 - \alpha_1 z - \alpha_2 z^2 - \cdots - \alpha_p z^p$ lie outside the unit circle.

where $\beta_0 = 1$.

The autocovariance function for this process is

$$\gamma(k) = \sum_{t=0}^{q-k} \beta_t \beta_{t+k}, \qquad k = 1,2,\ldots,q,$$

and is zero for $q > k$.

3.5.4 ARMA(p,q) Models

A stationary autoregressive moving-average process is obtained by combining the equations for an MA process and AR process. An ARMA(*p*,*q*) process is defined by

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{j=0}^{q} \beta_j \varepsilon_{t-j}.$$

The process ε_t is sometimes called the *innovations* process. A frequently used assumption is that the innovations ε_t are Gaussian, uncorrelated (thus they are also independent).

3.5.5 ARIMA(p,d,q) Models

Many time series encountered in practice are *nonstationary*. For these series, simple ARMA models are typically inadequate. However, the *differenced* series may be stationary. These are known as autoregressive integrated moving-average (ARIMA) models.

An ARIMA(*p*,*d*,*q*) process is a process whose *d*th difference $\nabla^d X_t$ is an ARMA(*p*,*q*) process. For example, with *d* = 1, the differenced series $W_t = \nabla X_t = X_t - X_{t-1}$ is assumed to follow an ARMA(p,q) process. When *d* = 2, the twice differenced series is

$$W_t = \nabla^2 X_t = \nabla(X_t - X_{t-1}) = X_t - 2X_{t-1} + X_{t-2}.$$

3.5.6 Identifying and Fitting ARIMA Models

The paradigm for fitting ARIMA models is to iterate through the following steps:

1. *Model identification*: Determination of the ARIMA model orders (*p*,*d*,*q*).
2. *Estimation of model parameters*: The unknown model parameters are estimated.
3. *Diagnostics and model criticism*: The residuals are used to validate the model and suggest potential alternative models which may be better.

These three steps are repeated until a satisfactory model is found.

3.5.6.1 Model Identification

Initial identification is done using the autocorrelation function.

An alternative procedure for selecting the model order is use of a penalized log-likelihood measure. One such measure is Akaike's Information Criterion (AIC). For AR models, see 3.5.2.2. For general ARIMA models see below.

3.5.6.2 Estimation of Model Parameters

The log-likelihood for an ARMA model can be computed using the *prediction error decomposition*. Consider an ARMA process X_t and assume the innovations ε_t are independent Gaussian random variables. Let

$$\hat{x}_t^{t-1} = E\{x_t, | x_1, \ldots, x_{t-1}, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q\}$$

denote the conditional mean one-step-ahead prediction of x_t based on the data $x_1, x_2, \ldots, x_{t-1}$, and let

$$\sigma_\varepsilon^2 f_t = \mathrm{var}\{x_1, \ldots, x_{t-1}, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q\}$$

denote the conditional variance of $\hat{x}_t^{t-1}$. Defining the *prediction errors* by $e_t = x_t - \hat{x}_t$ and letting $L(x_1, \ldots, x_n)$ denote the likelihood, one can show that

$$-2 \log L(x_1, \ldots, x_n) = n \log(2\pi\sigma_\varepsilon^2) + \sum_{t=1}^{n} f_t + \frac{1}{\sigma_\varepsilon^2} + \sum_{t=1}^{n} \frac{e_t^2}{f_t}. \quad \text{(Lx)}$$

Fitting an ARMA(p,q) model by Gaussian maximum likelihood involves finding the estimates $\hat{\alpha}_1, \ldots, \hat{\alpha}_p$ and $\hat{\beta}_1, \ldots, \hat{\beta}_q$ that yield a minimum in (Lx). The estimate of σ_ε^2 is $\sum_{t=1}^{n} e_t^2 / f_t$, which can be concentrated out of the likelihood. The likelihood is, in general, nonlinear in $\hat{\alpha}_1, \ldots, \hat{\alpha}_p$ and $\hat{\beta}_1, \ldots, \hat{\beta}_q$ and so a nonlinear optimizer must be used.

Estimating ARIMA models by Gaussian maximum likelihood is a straightforward extension from estimating ARMA models. The likelihood for a nonstationary series is obtained by differencing the data and computing the likelihood for the differenced process.

3.5.6.3 AIC and Model Selection

One method of model selection is based on Akaike's information criterion (AIC). The best model is given by the model with the lowest AIC value. AIC is a penalized version of the conditional log-likelihood function and is defined by

$$AIC = -2 \log L(x_{m+1}, \ldots, x_n | x_1, \ldots, x_m) + 2r,$$

where the conditional log-likelihood function is given by

$$-2 \log L(x_{m+1}, \ldots, x_n | x_1, \ldots, x_m) = (n-p) \log(2\pi\sigma_\varepsilon^2) + \sum_{t=p+1}^{n} \log f_t + \frac{1}{\sigma_\varepsilon^2} \sum_{t=p+1}^{n} \frac{e_t^2}{f_t}$$

and *r* is the total number of parameters estimated. Specifically, *r* is the number of AR, MA, and the regression coefficients. For example, for an ARIMA(1,1,1) model, *r* = 2.

When comparing the AIC values for different models, it is important to condition the likelihood on the same number of observations. In other words, *m* should be the same for all models.

3.5.6.4 Diagnostics and Model Criticism

The third stage in fitting ARIMA models consists of validating the model through examination of the one-step prediction residuals e_t.

The single most important diagnostic is a plot of the *standardized residuals* $\tilde{e}_t \equiv e_t / \sqrt{f_t}$ over time. If the correct ARIMA model is fit and the data are Gaussian, then $\tilde{e}_t$ should behave approximatelly like a

Gaussian white noise process with zero mean and unit variance. Problems to look for in the plot of $\tilde{e}_t$ include outliers, nonhomogenity of variance, and obvious structure in time.

Another basic technique is to examine the autocorrelation function of the residuals e_t. Let $\hat{\gamma}_k$ be the autocorrelations of the residuals e_t. If the model is adequate, then $\hat{\gamma}_k$ should be uncorrelated and approximatelly Gaussian random variables with mean zero and variance 1/*n*. Hence, the presence of large autocorrelations indicates that the model may be inadequate.

In addition to examining the $\hat{\gamma}_k$'s individually, it is useful to base a diagnostic on the autocorrelations taken as a whole. Define the *portmanteau* test statistic *Q* by

$$Q = n\sum_{k=1}^{K}\hat{\gamma}_k^2 ,$$

where *K* is a fixed maximum number of lags and *n* is the number of observations used to compute the likelihood. Typically, *K* should be between 10 and 20. If the correct ARIMA model is fit and the data are Gaussian, then *Q* is approximatelly distributed as a χ^2 random variable on *K-r* degrees of freedom, where *r* is the number of parameters fit to the model.

3.5.7 Forecasting using ARIMA models

An important application of ARIMA models is to forecast beyond the end of a series. Typically, one would first fit an ARIMA model. The resulting model can then be used to produce forecasts for the series.

3.5.8 Long memory time series modeling

Long memory time series have autocorrelations that decay slowly as lag increases; typically the autocorrelations tend to zero hyperbolically (that is, $\rho(k) \sim k^{-\alpha}$, with $\alpha > 0$) so that the sum of the autocorrelations is infinite (that is, $\sum_{k=0}^{\infty}\rho(k) = \infty$).

The ARMA models (with no differencing) are, by contrast, short memory models. For them, the autocorrelations decay exponentially, the sum of the autocorrelations is finite. *Fitting a (short memory) ARMA model to the data can give very misleading results if the long memory property holds, even if the fitted model matches the lower-lag autocorrelations well.*

3.5.8.1 Fractionally differenced ARIMA models

The *fractionally differenced* ARIMA(p,d,q) model can represent long memory time series quite well. It is defined as in 3.5.5 except that now *d* may take any value in the unit interval [0,1]. For this model $\rho(k) = k^{-(1-2d)}$. The process is stationary only for $0 \le d < 1/2$ and reduces to the usual short memory ARMA(p,q) model when *d* = 0.

The log-likelihood for the fractionally differenced ARIMA(p,d,q) model can be computed exactly using the prediction error decomposition as before. A major practical problem with maximum likelihood estimation based on this likelihood is that the required CPU time is $O(n^2)$. Therefore an approximation can be used that essentially approximates the dependence of x_t on x_{t-j} for *j* > *M* by assymptotic values.

3.5.9 ARIMA Database Entries

In InterMON Deliverable 2, in section 8.2 (IM -DB Logical Data Base Description) the entities “QoSP prediction model” (in sec. 8.2.6.4) and “Border Router Flow Prediction Model” (in sec. 8.2.8.7) contain the attributes for ARIMA model methodology and model parameters as follows:

Attribute	Data type	Description
Prediction Model methodology option	INTEGER	Identifies the prediction model methodology of AutoRegressive Integrated Moving Average (ARIMA)
Number Model parameters	INTEGER	Number of model parameters
Parameter values Model Distribution	LIST	Description of model parameter values

Table 3.3 - Originally planned ARIMA storage format

As the modification of the above attributes, the following entity for storing an ARIMA(p,d,q) model is proposed instead:

Attribute	Data type	Description
p	INTEGER UNSIGNED	order of autoregressive operator
d	FLOAT	the number of differences
q	INTEGER UNSIGNED	order of the moving average operator
period	INTEGER UNSIGNED	the period of ARIMA operators (for modeling seasonality)
ar	LIST	vector of autoregressive coefficients (length *p*)
ma	LIST	vector of moving average coefficients (length *q*)

Table 3.4 - Proposed updated ARIMA storage format

Notes:

- The parameter *period* makes it possible to use the model for time series data exhibiting seasonal cycles.
- For *fractionally differenced* ARIMA(p,d,q) (also called as fARIMA) models the number of differences *d* may take any value in the unit interval [0,1].

3.6 Modeling by Artificial Neural Networks

3.6.1 Introduction

The world of models can be divided into two parts. Hard-computing uses analytical methods for modelling that describes the real world more or less precisely but sometimes are hard or impossible to compute. Soft-computing uses approaches that are applicable even in such situations where no analytical model is possible or it is very difficult to find it but they approximate the truth with worse

efficiency. One of the main approaches of soft-computing methods is the artificial neural network ([Gur97], [Hay99]).

As a consequence of the InterMON architecture (periodically measured traffic patterns) the multi-layer perceptron (MLP) structures seems to be applicable. The applicability condition of MLP networks is that the output has to depend only on the appropriate input and must not depend on the earlier values of the input or the output. In the InterMON case it means that a measured output load has to be the function only of the measured input load. In inter-domain environment this property holds if the sampling interval is significantly greater then the overall delay of the domain.

3.6.1.1 Architecture

Many kinds of neural network structures are known. One of the most frequently used is the multi-layer perceptron (MLP [Min69]). A multi-layer perceptron is a feed-forward neural network consisting of a number of units (neurons) that are connected by weighted links (see Figure 3.23). The units are organised in several layers, namely an input layer, one or more hidden layers, and an output layer. The input layer receives an external activation vector, and passes it via weighted connections to the units in the first hidden layer. These compute their activations and pass them to neurons in succeeding layers.

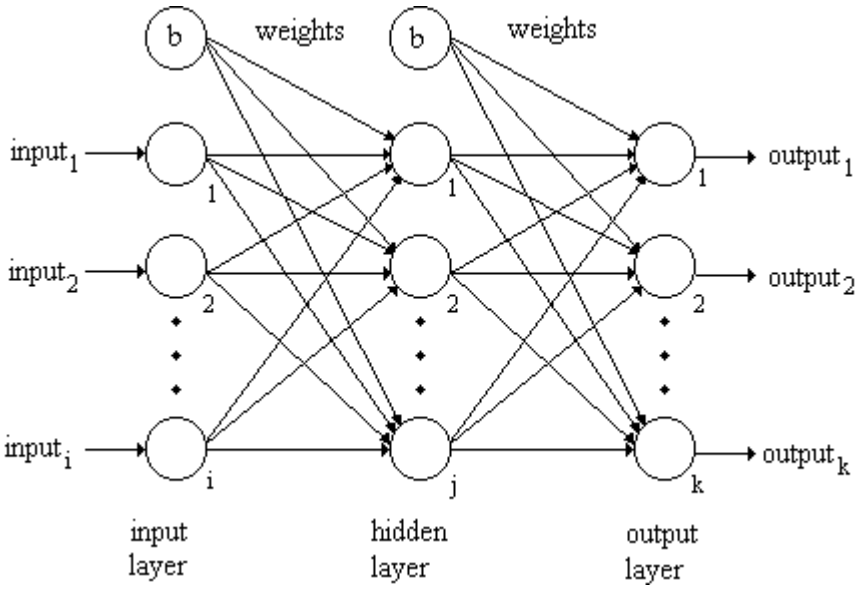

Figure 3.23 - Topology of an MLP with one hidden layer

From a distal point of view, an arbitrary input vector is propagated forward through the network, finally causing an activation vector in output layer. The entire network function that maps the input vector onto the output vector is determined by the connections weights of the net. Each neuron in the network is a simple processing unit that computes its activation with respect to its incoming excitation. The input of the i^{th} neuron in the l^{th} layer (see Figure 3.24):

$$s_i^{(l)} = \sum_{j \in pred(i)} y_j^{(l-1)} w_{ji}^{(l-1)} + b_i^{(l-1)}$$

where pred(i) denotes the set of predecessors of unit i, w_{ji} denotes the connection weight from unit j to unit i, and b_i is the unit's bias value. For the sake of homogenous representation, b_i is often substituted by a weight to a 'bias unit' with a constant output 1.

The activation of unit i is computed by passing the input through a non-linear activation-function. Usually, the sigmoid function is used:

$$y_i^{(l)} = \frac{1}{1+e^{-s_i^{(l)}}}$$

A nice property of this function is its easily computable derivative:

$$\frac{\partial y_i}{\partial s_i} = y_i(1-y_i)$$

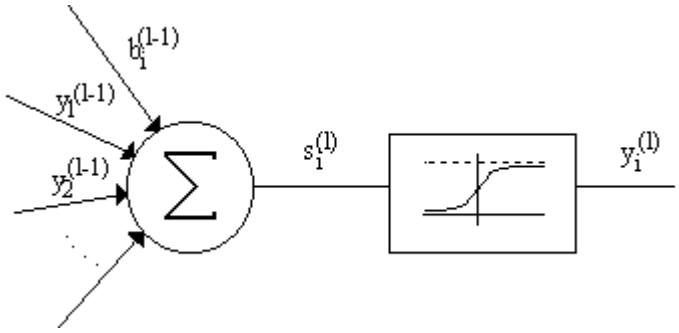

Figure 3.24: Inputs and output of a neuron

3.6.1.2 Training

In training, the objective is to tune the weights in the network such that the network performs a desired mapping of input to output activations. The mapping is given by a set of examples of this function, the so-called pattern set. Each pattern pair *p* of the pattern set consists of an input activation vector x^p and its target activation vector t^p. After training the weights, when an input activation x^p is presented, the resulting output vector y^p of the network should equal the target vector t^p. The distance between the target and the actual output vector, in other words the fitness of the weights, is measured by the following error or cost function:

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{n=1}^{k}(t_n^{(p)} - y_n^{(p)})^2$$

where k is the number of units in the output layer. Fulfilling the learning goal now is equivalent to finding a global minimum of *E*. The weights in the network are changed along a search direction *d(t)*, driving the weights in the direction of the estimated minimum:

$$w(t+1) = w(t) + \eta d(t)$$

where the learning parameter η scales the size of the weight-step. To determine the search direction *d(t)*, first order derivative information, namely the gradient $\nabla E = \partial E / \partial w$ is commonly used [Rum86].

The most used method to minimize the error is the Error-Back-Propagation (BP [Wer74]) algorithm, which is a steepest descent algorithm. It is a first-order method as it only uses derivatives of the first order of the error. If no line-search is used, then it has no guarantee of convergence and the convergence rate obtained is usually very slow. If a second-order method is to be employed, the best algorithm to use is the Levenberg-Marquardt (LM [Mar63]) algorithm, which explicitly exploits the underlying structure (sum-of-squares) of the underlying optimization problem.

Which of these to algorithm is to be used is not clear in advance. The size of the problem may cause the LM algorithm to be very complex numerically and then we will have to use the BP algorithm. We can make the decision only after some testing.

3.6.2 Router and domain models

Neural networks represent a nonlinear function between its inputs and outputs. They are trained by patterns to approximate the function. In the InterMON architecture periodic measurements can be carried out to supply the various applications with information about the state of the network. These measured samples can be used to train the neural network continuously, 'on line'. Since the training has a high cost it is not effective to train models for each simulation. Instead, permanent models should be used that get the periodic measurement data, thus they can adapt to the changes of the monitored network element. Then if a simulation was requested, the simulation environment can use the permanent models – maybe more simulation can use it simultaneously – to act as simulation elements.

It depends on the measurements what the neural network models. If we sample the inputs and the outputs of a router than the neural network will model that router. If we sample the edges of a domain, then it will model the domain and so on. Theoretically any part of the network can be modeled with neural networks; it only depends on what we measure.

Anyhow, the model should have one input and more outputs (see Figure 3.25). The input is a load value measured or predicted on the input of the modeled entity. The output consists of an output load value and, maybe, some QoS parameters. The model should be implemented as an MLP with a first layer of one perceptron and a last layer of as many perceptrons as many outputs are needed. The number of the internal (hidden) layers is hard to determine in advance. It depends on the properties of the modeled entity (router, domain) and may change from entity to entity. However a good approximation may be applicable but it is an open question now and it should be further researched. (Maybe we can determine different numbers for different type of entities.)

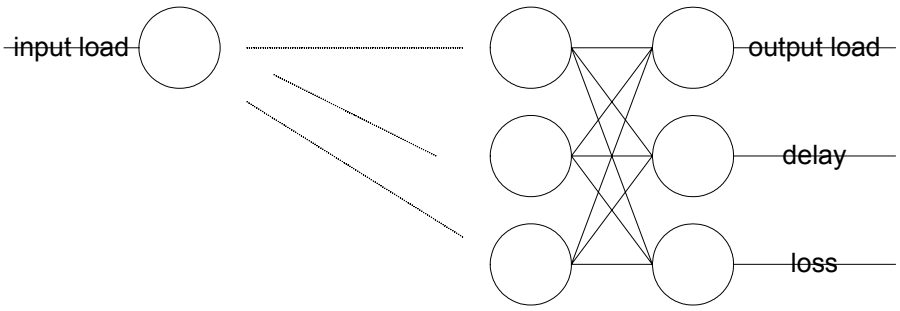

Figure 3.25: Neural network that models a router or a domain

3.6.2.1 Database Storage

Storing a neural network in a database is quite simple, since we only have to store the weights of the network and the positions for the weight. Based on this information any neural network can be restored.

3.6.3 Conclusion

Neural networks could be good tools for modelling functions for that it is difficult to find an analytical solution. Therefore they can be applicable for modelling routers or domains. However there are difficulties:

- It is not clear that the number of hidden layers and the number of neurons in a hidden layer can be computed automatically. For practical reasons one hidden layer is appropriate and we can use some heuristics or we can overestimate the number of hidden neurons, but an optimal solution may require human interaction.

- It is not clear either which training algorithm can be used to train the network. Several algorithms exist with different accuracy, speed of convergence and complexity.

These problems are not trivial to answer but there exists solution. The difficulties, however, have to be taken into account at the decision

3.7 Planisfero Modelling Approach

The aim of this chapter is to provide a general overview of the modelling and performance evaluation approach performed by Planisfero.

Planisfero is a TILAB tool specifically designed to satisfy the intra-domain network planning needs in a DiffServ context of TelecomItalia. In particular, Planisfero is addressed to ATM and IP DiffServ networks for the purposes of:

- Topology planning
- Traffic and MPLS tunnels routing
- Network branches sizing
- Network nodes sizing
- Network performance evaluation
- Network elements failure simulation and what-if analysis

The modelling approach, although designed for intra-domain networks, could be, with some care, extended to the inter-domain context in which the InterMON platform is planned to work. More specifically, the items that could be exported are the last two of the previous list: network performance evaluation and what-if analysis. The following description is focused on these aspects.

As shown in Figure 3.26, three modules compose Planisfero:

- Routing module;
- Capacity planning module;
- Performance evaluation module.

The routing module contains routing algorithms able to simulate real network protocols. The currently implemented routing protocols are OSPF (with Equal-Cost MultiPath option) and MPLS. The inputs for the routing modules are constituted by:

- Network Topology information:
  - Node list;
  - Link (intended as monodirectional) list;
- OSPF information:
  - OSPF link metrics.
- MPLS information:
  - Tunnel list (in terms of end-point pair)
  - Attributes of the tunnels (bandwidth, static route, etc.)

The routing module is used in the first phase of the network planning, network performance evaluation and what-if analysis. It simply computes and stores the paths that each traffic component follows inside the network. In addition, the routing module computes the traffic routing, on a static base, among MPLS tunnels. It's important to highlight that this is the only utilization of traffic volume information inside the routing module.

As it is, the routing module needs some further enhancement in order to be adapted to the InterMON inter-domain context. In particular, BGP-4 implementation is a required step.

The second component of Planisfero is the capacity-planning module. This module is reported here only for shake of completeness. Basically, given the topology and routing information passed by the routing module, the capacity-planning module computes the bandwidth required for each link in order to achieve the performance constraints imposed as input for the module. The main input needed by the capacity-planning module is constituted by the information about the amount of traffic exchanged between each end-point pair, detailed on a per-class basis. The analytical models used by the capacity-planning module are the same used in the performance-evaluation module and, for this reason, are described in the following.

The third Planisfero component, the most interesting from the InterMON viewpoint, is the performance evaluation module. The inputs required by the module are:

- The outputs of the routing module (routing configuration, MPLS traffic and tunnel routing configuration);
- The Traffic matrix (per-class traffic is provided for each E2E relation and MPLS tunnel)
- The capacity of the links (and in a DiffServ networks the scheduler weights)

The network behaviour is modelled as follows. The node models perform the routing decisions establishing the output link for each traffic subset entering in the node. The scheduling is implemented directly in the link model. In this way, each link contains a certain number of queues depending of the planned traffic classes. Typically, one queue is reserved to the strict priority class (which carries traffic with strict requirements in terms of delay and jitter, e.g. VoIP traffic). Other queues are dedicated to the rest of traffic. The queues are emptied following the MDRR scheduling algorithm. So, an additional required input for the performance evaluation module is constituted by the list of the MDRR weights (provided per-class and per-link).

The priority M/G/1 queuing model is therefore used to evaluate the performances as long as the link is not in the congestion state (total offered traffic greater than its capacity). If the link is globally congested the performances of the non-congested traffic classes only are evaluated using a heuristic method that assumes that each traffic class experiments the performance that it would see in isolation. Matching these results with the routing information, the model applies an E2E composition rules (mainly relying on the additional property for the delay) computing the E2E performances:

- E2E average delay (per-class based for each E2E relation);
- E2E jitter information based on the use of percentiles (per-class based for each E2E relation).

Working on this way, the PEM allows obtaining delay information from load information. This approach can be considered quite realistic especially thinking what happens in real networks. Actually, the routers are able, via implemented MIBs, to provide local traffic information (that are, for this reason, easily available), but have more problems for providing delay information. Using M/G/1 queuing algorithms, as done by the PEM, realizes a compromise between the purpose of obtain e2e delay and maintaining a certain degree of simplicity avoiding direct or intrusive measurements.

InterMON extensions for the performance evaluation module should take into account the modelling of the interconnected AS. In other words, the Performance Evaluation Module is already able to estimate the delays introduced by the edge routers located at the borders of the AS domain. It reaches, also, the objective to perform what if analysis. Actually, it is possible, after a first run of the PEM, to analyse the performance results and introduce changes in all the inputs of the PEM. This means that the user can change, also in a drastically manner, the amount of traffic (at whatever level the user likes) or the routing parameters (in order to find a different way to carry the desired amount of traffic). The user can also add or remove MPLS tunnels or change the class assignment for the traffic component. In this way, the PEM can help to reach the InterMON goal of understanding what happens in the network state once a perturbation occurs.

Some effort is needed to obtain a model to estimate the delays introduced on the E2E performance by the internal crossing of the domains. The amount of required effort depends on the modelling approach for the AS cloud. Actually, a reduced effort should be possible modelling the intra-AS delay as fixed or measurement derived. In this case, only the existent module E2E delay composing rule should be modified. Alternatively, additional effort has to be planned in order to develop a model that can estimate intra-domain delays.

Furthermore, classical M/G/1 model for border routers can be enhanced to better reflect the MDDR or WFQ behaviour implemented in routers. A simulation derived empirical model was already defined in [Salta].

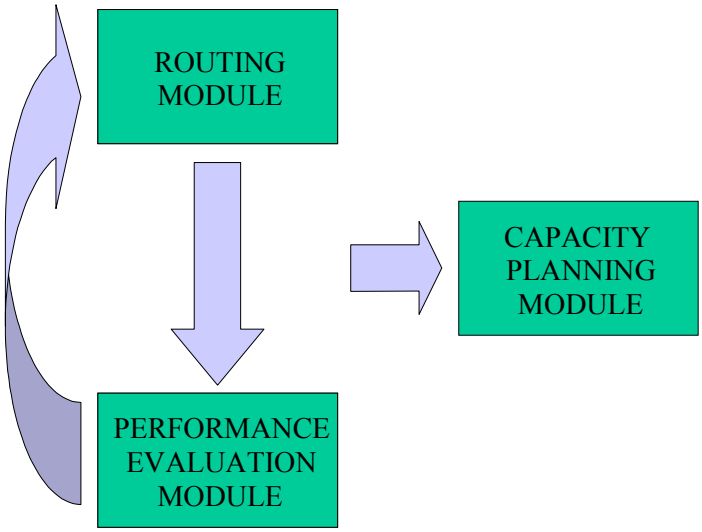

Figure 3.26 - Planisfero Modules

4 InterMON Simulation and Modelling Environment

4.1 Measurement Based Modelling and Integration into Simulation Environments

A goal of the InterMON architecture is generation of inter-domain simulation scenarios using traffic and QoS models derived and parameterised automatically from measurements. Models can be separated into operational (real time, quasi static) and predictive ones. They are integrated into different simulation environments (NS-2, fluid and time series simulation). The following figure is used to explain the main structure of the measurement based modelling and simulation environment of InterMON.

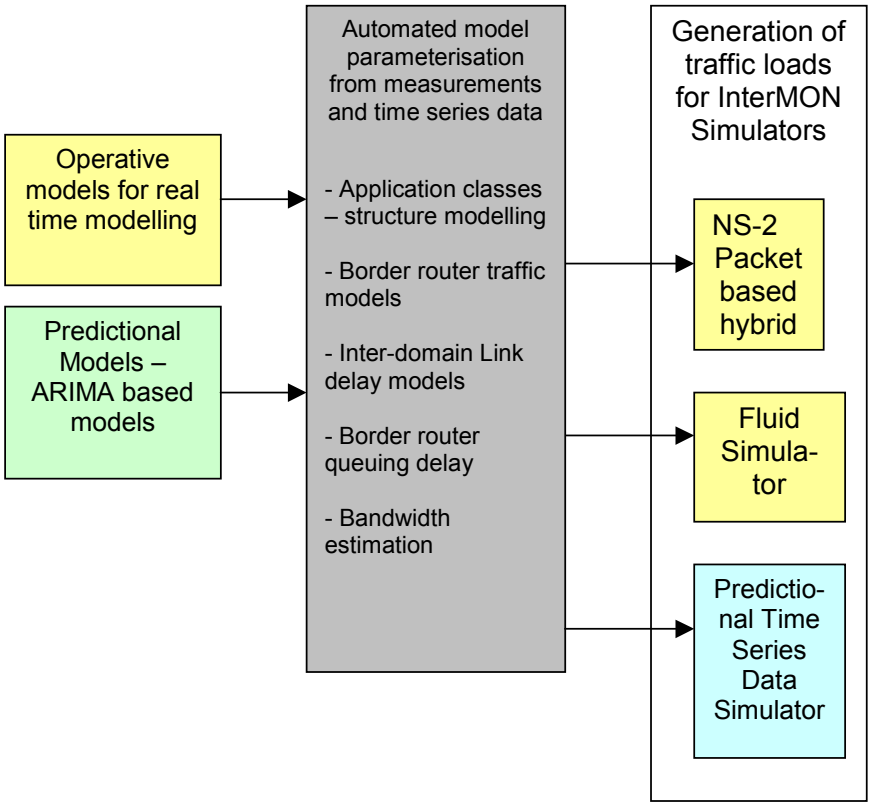

Figure 4.1 -Modelling and simulation framework in InterMON

4.2 Packet-Based Approach

4.2.1 Simulation Environment

Packet-based simulation environments are based on packet treatment models for ISP networks. There are several disadvantages of this modelling approach:

- A detailed simulation of entire ISPs network is not feasible due to the size of ISPs networks and the high number of flows. Especially within a planning tool long computation times are not practicable.

- Information about the intra ISP topology are not available and there is little interest of ISPs to share this competition-critical information.

Therefore an environment is required, which allows to replace the ISP networks within a simulation by ISP models. These ISP models describe the behaviour of entire ISP networks including packet loss, delay and jitter. Such a model can be used to model a single ISP or a set of multiple ISP networks without explicitly calculating the effects of all nodes inside the modelled domains.

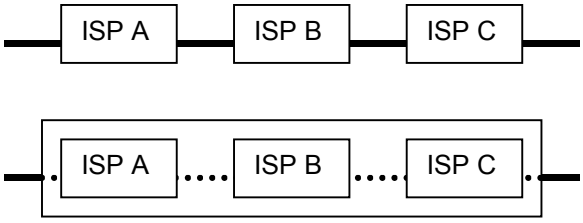

Figure 4.2 - Chain of ISP models and a model of multiple ISPs

If a single ISP is represented, the inter ISP packet treatment is not provided by the model, but is left to a network simulator, connecting the ISP models. The inter ISP delays between ISPs, modelled by a single black box model, have to be provided by the model itself. In this case, the black box model also has to reflect any kind of interconnection between the modelled ISPs.

4.2.2 Simulator Integration

We propose a strict separation of the packet-based simulation toolkit and the ISP models. This simplifies the development and implementation of new models without the need of modifying the network simulator itself. For the interaction between the simulator and the ISP models an interface has to be defined, which allows the simulator to retrieve information about packet loss and delay from the ISP model.

As a basis for the simulation toolkit we propose ns2. Ns2 is a discrete event simulator, which provides support for various protocols, traffic sources, and traffic conditioning components. Ns2 is well known and also supports Differentiated Services, which is essential to simulate inter ISP links. Ns2 uses a rather small set of core components, implemented in C++, which can be accessed, extended and combined through a scripting language front-end based on TCL.

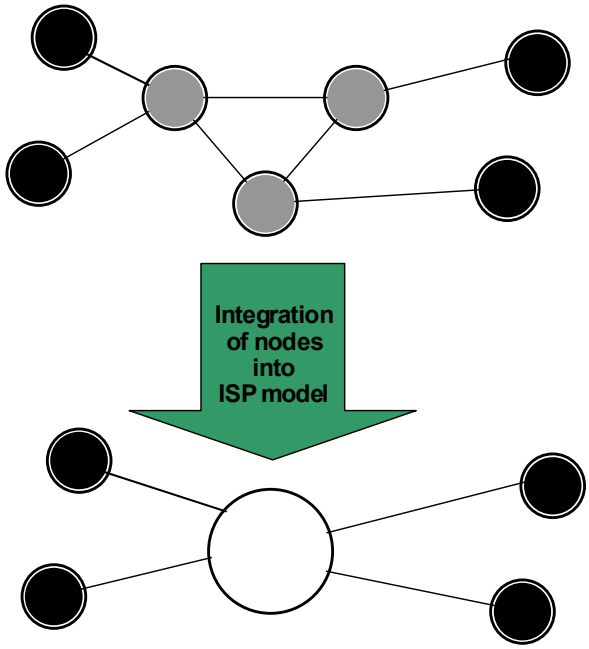

Figure 4.3 - Integrating a set of nodes (an ISP network) into an ISP model

4.2.2.1 Integration into NS

Since ns2 is a multi purpose simulator also supporting wireless and satellite networks, amongst others, the representation of the network is rather abstract and the graph-theory-like edges and nodes approach can not be directly mapped to real world links and routers. While ns2 links can provide the functionalities required to implement inter ISP links, the ns2 nodes have to be extended to simulate the behaviour of a single or a set of ISP networks.

Combining nodes representing entire ISPs (see Figure 4.3) with normal ns2 nodes allows to simulate the treatment of packets depending on the applied models. Several necessary modifications to ns2 have been identified.

Since ns2 nodes only provide routing mechanisms and act as traffic sources and sinks only, packet delays and losses only occur at the ns2 link components. Nodes have to be extended to provide support for a varying forwarding delay and packet loss.

A plug-in mechanism has to be implemented to allow the attachment of ISP modules to ns2 nodes. This mechanism has to support a parameterization of the modules as well as the usage of different modules at different nodes.

An interface has to be defined for the communication between the network simulator and the attached module. The model will mainly provide information about packet delay, a possible packet loss or header modifications. The network simulator can provide:

- ID of the current node (ISP).
- ID of the node (ISP) the packet was received from.
- ID of the next node (ISP) the packet will be send to.
- Header information of the processed packet.
- Current simulation time

Since ns2 is rapidly evolving, the interface between ns2 and the external modules should be kept simple, and additional mechanisms should be implemented rather within the external modules than in ns2 itself. The module itself can be implemented externally as an executable object and attached dynamically to the appropriate nodes.

4.2.2.2 Flow & Packet Based Approach

Even if ns2 is packet based only, the combination with the ISP modules allows to calculate a packet's per domain behaviour by investigating the aggregate level behaviour within an ISP or a set of ISPs. Whether intra ISP links are handled on a per packet base within ns2, or are handled on an aggregate level within a multi ISP model depends on the available data, the topology size and the required outcome of the simulation. For ns2 there is no difference whether a module attached to a node simulates a single ISP or a set of ISPs. Additionally this model-nodes support the integration of aggregate based traffic sources. This allows to simulate and investigate the treatment of packets, dependent on additional load, without having to implement packet-based traffic sources.

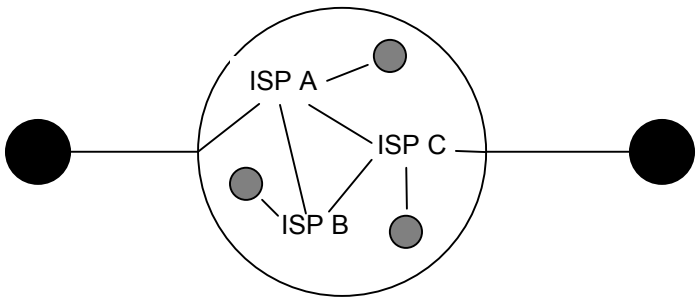

Figure 4.4 – Two ns2 nodes connected via a "model-node", modelling three ISPs including aggregate based traffic sources

4.3 Fluid-based Modelling and Simulation

Part of the InterMON modelling and simulation environment is a novel modular Rate and Time Continuous Fluid SIMulation technology (RTC-FSIM) based on differential equations described in technical reports [Ber02a], [Ber02b], [Ber02c].

RTC-FSIM technology bases on different layers of abstractions (border router, autonomous systems, clouds of autonomous systems, etc) using differential equations to describe continuous rate traffic behaviour.

Because the flow path decomposition represented by differential equations from node to node is a general one and is appropriate to represent any kind of queuing system, there are a variety of models based on differential equations integrated for descriptions of different kinds of queuing systems. The different kinds of models could be flexibly combined by the user dependening on the selected abstraction (border router, autonomous system, etc), complexity of inter-domain systems, inter-domain routing and topology abstractions.

Briefly summarised, the purpose of the InterMON RTC-FSIM technology in the InterMON project is to build inter-domain modelling and simulation environments with the following facilities:

- Large scale inter-domain modelling and simulation based on high level abstractions using the differential equation paradigm
- Efficient measurement based modelling for generation of continuous rate input models (described with mean, variance and autocorrelation function) using MATLAB with interfaces to RTC-FSIM inter-domain models implemented in SIMULINK [Simulink]. An important reason for choosing these tools to build the RTC-FSIM environment are their powerful modelling and simulation facilities combined with visualisation functionalities. Furthermore these tools support comfortable possibilities for analysis in the frequency domain.
- A modular and extendable simulation environment integrating different kinds of inter-domain simulation models (basic service model, multiservice and priority model, etc)
- Inclusion of inter-domain models describing traffic classes and priority services aimed to describe QoS issues (for instance DiffServ) in inter-domain environments (the multiservice and priority model [Ber02b]).

Currently implemented in RTC-FSIM is the general service model [Ber02a] for fluid flow approximation using an aggregate traffic model as input (i.e. input continuous rate model obtained by superposition of ON/OFF sources with identical distribution and autocorrelation). This general service model does not differentiate between traffic classes and could be used for traffic engineering considering resource capacity of inter-domain node abstractions (border routers). Another implemented RTC-FSIM model is the multiservice model based on priority processing [Ber02b] which could be used for simulation studies of different traffic classes such as user application traffic classes (VoIP, MPEG video) in QoS based inter-domain node environments (for instance DiffServ border router processing traffic classes with different priorities).

RTC-FSIM is a large scale environment because it is designed for flexible integration of topology and inter-domain routing models (i.e. BGP-4 protocol), as for instance multihoming heuristics (see [Ara02]).

The following sections give a brief introduction into the fluid simulation and RTC-FSIM modelling and simulation environment. For more details on working with RTC-FSIM as well as comparison with existing modelling and simulation technologies please consider the technical reports [Ber02a], [Ber02b] and [Ber02c].

4.3.1 Introduction to Fluid Based Simulation

Fluid-based models of communication network traffic consider traffic behaviour in terms of *rates* (i.e. rate changes), rather than packet instances. The fluid model was first proposed by Anick et al. in [Ani82] to model data network traffic. Fluid modelling and simulation techniques for communication networks and services are discussed in different works ([Ahn96], [Kes96], [Kum98], [Yan99], [Liu99]).

In the fluid simulation paradigm, network traffic is modelled in terms of a (time) discrete or continuous rate based models, rather than discrete packet instances. A fluid simulator technique keeps track of the fluid rate changes at traffic sources and network nodes. An equivalent packet-level simulator would keep track of all individual packets in the network. Issues and trade of fluid simulation is addressed in [Liu99]. [Nic99] compared the performance of fluid-based and packet-based SSF models that differentiate in the generation and handling of events.

In fluid simulation, the higher level of abstraction suggests that less processing might be needed to simulate network traffic. Intuitively, this is not surprising as a large number of packets can be represented by a single fluid chunk. However, for event driven fluid simulation, a rate change event at one node triggers rate updates in all downstream nodes. More specifically, the change of one or more

departure rates from a node can change the arrival rates and station state in the set of nodes directly connected to it via continuous flows, which in turn can change their departure rates and so on. Thus, a rate change in a given node can potentially ripple to all downstream nodes causing multiple rate change event processing and performance degradation due to the ripple effect [Kes96], [Liu99], [Liu01]. The ripple effect depends on the complexity of topology, node queuing mechanisms and interconnections [Liu01] as well as used simulation technique (some techniques are reported to deal efficiently with ripple effect such as time stepped event simulation [Yan98]). The rate and time continuous fluid simulator hides this performance problem behind the numerical effort for differential equation solving ([Mel02], [Gar01], [Ber02c]).

Fluid-flow models were exploited in the current research, to achieve reductions in the computational complexity of simulation runs, for powerful modeling based on rate paradigm, as for instance:

- simulation of ATM rate based transport services ([Kes93], [Kes95], [Kes96])
- TCP fluid based flow analysis ([Bon98])
- Fluid Methods for Modelling Large, Heterogeneous Networks ([Liu99], [Gon00]).

[Nic99], [Ros99a], [Ros99b] discuss the computation of performance measures such as end-to-end delay and loss characterization by the fluid simulations. Nicol et al. [Nic99] claim that despite the high level of abstraction of fluid models, the error of estimated measures obtained with fluid simulation is very small compared to the results of packet-level simulation. Evolution of rate and time continuous fluid modelling and simulation covering also other possible application areas is given in [Ast98].

To relate efficiency and trade off of InterMON RTC-FSIM technology (especially for the inter-domain modelling and simulation context) to known packet and fluid modelling and simulation research, a comparison table for simulation technologies is provided below. The table differentiates between:

- source model : packet, rate discrete, rate continuous
- simulation technique (service process): event driven, time slice driven, time continuous.

SIM_TECHNIQUE (service process) **MODEL (source model)**		**Event based** **(rare, complex event processing)**	**Time slice T (fixed time increment) event** **(frequently, simple event processing)**	**Time Continuous** **(computational effort hidden behind the numerical algorithms of digital computers)**
	Packet based model	For: rare packet arrivals, medium complex network Not for: high rate packet arrival, complex networks	For: high rate packet arrival, complex networks Not for: rare packet arrivals, medium complex network	
	Fluid Rate dis-crete	For: rare rate changes, medium complex network Efficiency depends on possible ripple effect (rate change propagation) [Nic99], [Liu00]	For: high frequency of rate changes, complex networks Not for: rare rate changes, medium complex network Efficiency dependent on possible ripple effect (rate changes propagation) Time-slice driven fluid simulation scheme [Yan98]. Time stepped fluid [Guo00]	

Fluid Rate continuous			**Differential equations** For high frequency of rate changes and complex networks Hybrid Discrete Continuous Flow (HDCF) simulator [Mel02] Distributed hybrid simulation [Gar01] **RTC-FSIM InterMON** Efficiency depends on the computation complexity of solving differential equations

Table 4.1 - Efficiency and trade off of basic simulation techniques and models

The RTC-FSIM environment as seen in the table is focussed on a high level of abstraction i.e. continuous rate modelling, which is also the background for simple and efficient modelling of large complex inter-domain infrastructures. RTC-FSIM efficiency depends on the computational complexity of solving differential equations, which is considering the complexity of inter-domain node models, their complexity of interconnections and topology.

RTC-FSIM is based on one modelling and simulation paradigm, i.e. differential equations, in order to have a simple (rate based) modelling and simulation approach for QoS studies in a large scale network (considering traffic classes, QoS processing disciplines). The differential equation paradigm of the basic model is similar to the continuous flow model (CFM) used in the Hybrid Discrete Continuous Flow (HDCF) simulator [Mel02]. HDCF is aimed at integrated discrete packet and continuous flow simulation.

While RTC-FSIM is intended to provide the continuous rate abstraction for traffic behaviour in large scale networks, hybrid packet and fluid simulation are aimed at different abstraction levels. For instance, hybrid packet and fluid simulation in Project Maya (i.e. Next Generation modelling and simulation tools) integrates packet level simulation for subnets and fluid for heterogeneous large scale abstraction.

The distributed hybrid simulation concept for transient and stationary performance evaluation of large scale telecommunication networks proposed by the ESPRIT project STAR [Gar01], combines queuing theory models and packet based simulation in a global decomposed traffic modelling environment. The difference to the RTC-FSIM basic model and HDCF differential paradigm is that the differential equation in the [Gar01] model describe the mean and not the actual buffer occupation.

4.3.2 Introduction to RTC-FSIM Modelling and Simulation Environment

RTC- FSIM is based on simulation of a queuing system as signal flow model. The basic RTC-FSIM simulation model describes simple queuing system with one service station.

The basic model is given in the next figure:

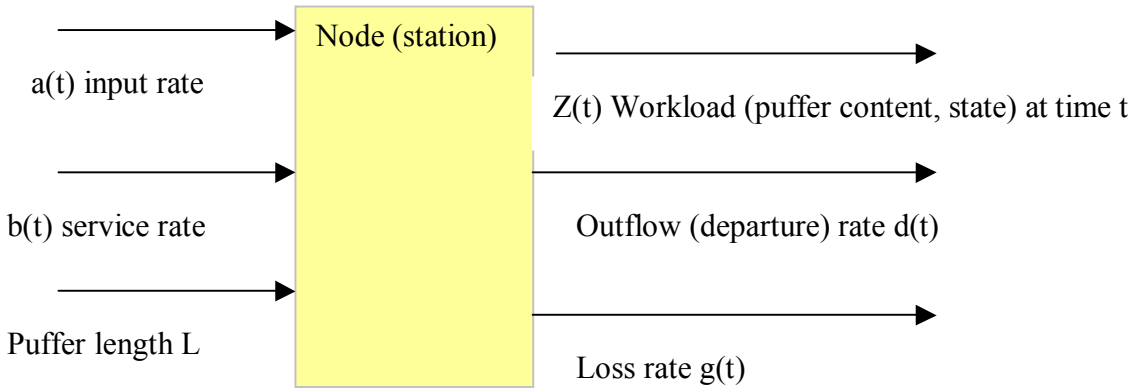

Figure 4.5 - Basic RTC-FSIM station model

The basic station model could be used for abstractions of border router considering actual buffer occupation. The differential equations describing the general RTC-FSIM node (station) model are given below. The buffer contents (workload) at time t is denoted by z(t) and given by

$$\frac{dz(t)}{dt} = \begin{cases} 0, & (z(t) = 0 \wedge a(t) - b(t) \le 0) \quad \vee \\ & (z(t) = L \wedge a(t) - b(t) \ge 0) \\ a(t) - b(t), & otherwise \end{cases} \qquad (1)$$

The outflow (departure) rate from the station at time t is denoted by d(t) and given by

$$d(t) = \begin{cases} \min(a(t), b(t)) & z(t) = 0 \\ b(t) & z(t) > 0 \end{cases} \qquad (2)$$

Finally, the spillover (loss) rate due to finite buffer space is denoted by L and given by

$$g(t) = \begin{cases} a(t) - b(t) & z(t) = L \wedge a(t) - b(t) \ge 0 \\ 0, & otherwise \end{cases} \qquad (3)$$

While the basic RTC-FSIM model is aimed to describe inter-domain node (i.e. border router) without consideration of traffic classes and QoS based processing, the multiservice and priority signal flow model is aimed to model the inter-domain border router based on different traffic classes with consideration of different priorities [Ber02b].

The multi service model assumes, that each class has an own unlimited buffer space. A fluid flow K_{ij} is specified by its source i and sink node j.

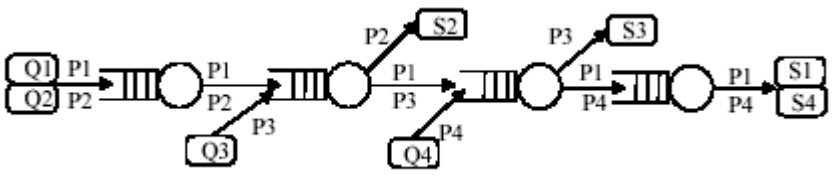

Figure 4.6 - Network with multiple fluid flows as base of multiservice model

Service disciplines without priorities (processor sharing) and with priorities are differentiated. The transient behaviour is given by the following equations:

Without priorities:

node state or workload (buffer content) equation:

$$\frac{dz_{ij}(t)}{dt} = \begin{cases} 0, & z_{ij}(t) = 0 \wedge a_{ij}(t) - b_{ij}(t) \le 0 \\ a_{ij}(t) - b_{ij}(t), & otherwise \end{cases}$$

departure rate equation:

$$d_{ij}(t) = \begin{cases} \min(a_{ij}(t), b_{ij}(t)), & z_{ij}(t) = 0 \\ b_{ij}(t), & z_{ij}(t) > 0 \end{cases}$$

i: node identifier

j: fluid flow identifier

i=1,2,..,M j=1,2,…, K_i

RTC-FSIM modelling and simulation is based on flexible interconnection of the different kinds of models (basic, multiservice, further models) representing inter-domain nodes (border router, Autonomous Systems) considering topology and inter-domain routing information (e.g. BGP-4 routing information and abstraction for BGP-4 services, as for instance multi homing heuristics).

The measurement based source traffic modelling is based on the continuous rate models obtained automatically by MATLAB from traffic sources. The input continuous rate model is described by the mean input rate, the input rate variance and the parameter of the input autocorrelation function.

The rate based quasi static traffic source modelling of RTC-FSIM could be enhanced by ARIMA techniques (see Section 3.5) for the purpose of traffic prediction modelling.

Focus of further investigations is the integration of techniques for merging of one or more ON/OFF packet or fluid traffic source models (for instance VoIP) with the continuous rate input traffic models as well as merging of superposition of many ON/OFF packet or fluid traffic sources with the continuous rate input traffic models.

4.3.3 Implementation of RTC-FSIM Based on SIMULINK and MATLAB

The modular design of the RTC-SIM is based on the usage of SIMULINK as basic simulation technology. Simulink [Simulink] provides an interactive block-diagram environment for modeling and simulating dynamic systems. It includes an extensive library of predefined blocks that can be used to build graphical models of your systems using drag-and-drop operations. Supported model types include linear, nonlinear, continuous-time, discrete-time, multirate, conditionally executed and hybrid systems. Models can be grouped into hierarchies to create a simplified view of components or subsystems. High-level information is presented clearly and concisely, while detailed information is easily hidden in subsystems within the model hierarchy. Simulink has many features that allow customization, especially with regard to incorporating existing user C code. In addition, simulations can be run interactively or in batch mode from the MATLAB command line.

Through its integration with the design process flow via optional products like Real-Time Workshop® and Stateflow®, Simulink is the foundation for a family of design solutions, spanning DSP, communications, control, and power system design. You can also extend your simulation environment even further with Simulink libraries for specialized applications, such as the DSP Blockset, the Fixed-Point Blockset, the Power System Blockset, and the Communications Toolbox. SIMULINK is used to generate signal flow plans describing the time continuous signal flow models used by RTC-FSIM. The facilities for hierarchical model design of SIMULINK are used for the implementation of RTC-FSIM modules. The implementation of the basic model described by differential equations for workload, departure rate and loss of is discussed in [Ber02a]. The multiservice class implementation is described

in [Ber02b]. The following figure shows the hierarchical design of the basic model implementation of the RTC-FSIM technology.

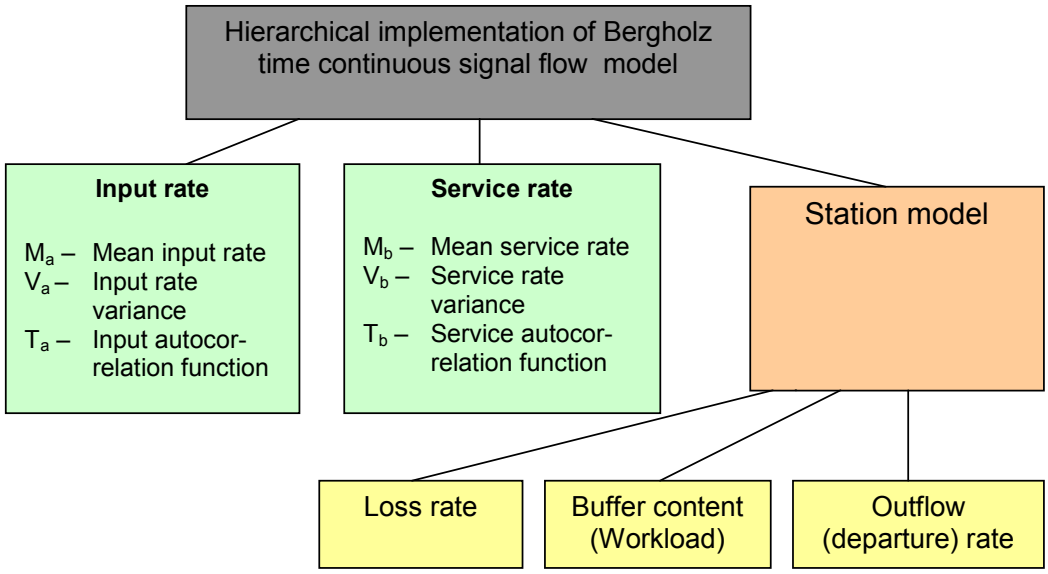

Figure 4.7 - Hierarchical design of the basic model implementation of RTC-FSIM

MATLAB is used for automated generation of RTC-FSIM model input to the SIMULINK environment, based on measurements and parameter evaluation of the input and output model processes.

In the case of basic RTC-FSIM model, MATLAB is used for the generation of the input rate models described by M_a – Mean input rate, V_a – Input rate variance and T_a – Parameter of input autocorrelation function, as well as for presentation of the simulation output parameters. Further parameter of the basic RTC-FSIM model are

L – buffer size

M_b – Mean service rate

V_b – Service rate variance

T_b – Parameter of service autocorrelation function

4.4 Time Series Simulation

4.4.1 Introduction

One of the basic requirements on the simulation environment is that it should integrate the measurement data and generate the simulation scenario automatically. It means that the toolkit should map the different kind of data to different simulation entities. According to the scenarios the toolkit should schedule the computations on these entities (models) and execute them. This section tries to clarify how this could be accomplished. We propose a new architecture of simulator called stochastic simulator.

Inputs

1. Topology (edge router connections)
2. Capacity of inter-connections (parameter of the edge router model)
3. SLA (to realize overflows)

4. Load (maybe the distribution too)
5. E2E delay, jitter and loss

Remark: Input 4 and 5 is in the form of time series.

The only problem corresponds to the e2e loss. If we assume the inside of domains congestion free, than probably no loss will occur between the edges. Despite, the edge models should handle losses. In addition of these inputs some control input is needed: which scenario to investigate and scenario specific parameters.

About the Simulation

In our vision the InterMON simulation works on edge routers. These are the entities of the real network that are characterized by the measurement data; moreover our selected domain model (static intra-domain conditions – dynamic inter-domain conditions) implies this too. Hence topology information means the declaration of the edge routers of the domains and the connections between them.

From the measurements the load is known on each edge router and based on the time series (input 4) prediction is possible. As a consequence we don't need to connect the edge routers the way like in an ordinary simulator and forwarding packets (or flow rates) is not necessary. It is pointless because we already know the loads on every (edge) router. Indeed, simulation is done only in the edge routers to evaluate the router model (if necessary).

To calculate the change of the QoS parameters of a path in such a simulator, first we need the series of edge routers (the path), and then we have to evaluate the router models step-by-step (router-by-router) using the previous step's (router's) output as input of the current step. The input of the first router model is the predicted time series of measured load plus some new traffic. The output is a couple of new time series that reflects the effect of the first router. The output series is forked based on the measured traffic distribution and modified according to the 'static' intra domain behavior. In the end we get a new time series that is the input of the next router model.

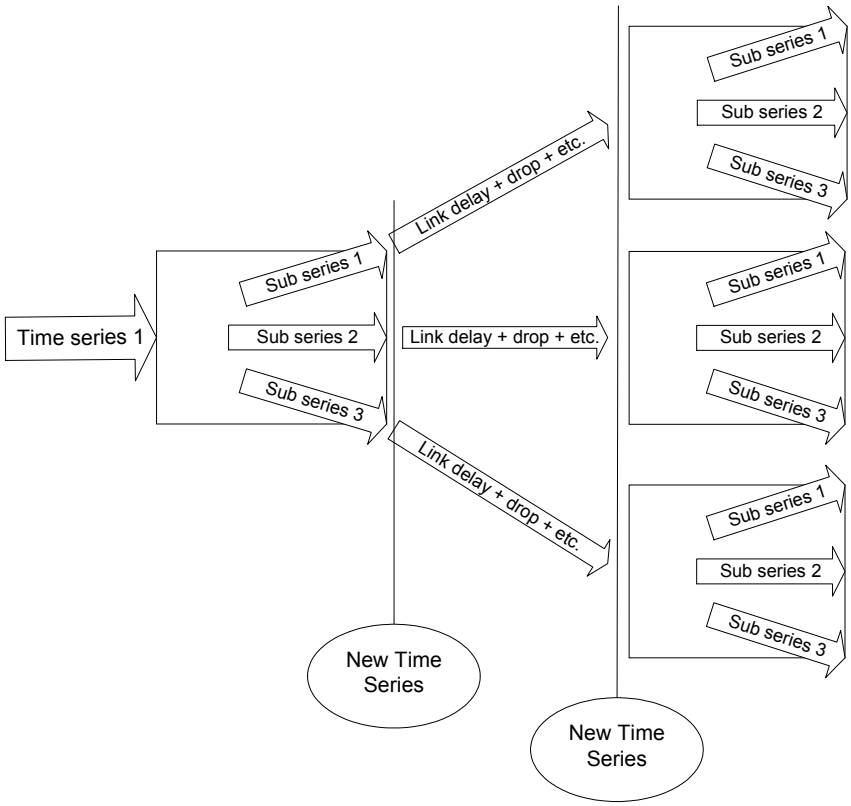

Figure 4.8 - Simulator architecture that works on time series

As seen on Figure 4.8 the router and link models take a time series as input and generate another as output. Thus they can be concatenated. Note that the models of the routers and links are pluggable. This enables the use of different models.

The architecture has two critical points: traffic forking and traffic merging. Traffic merging is needed when a router has more then one inputs. It is not too difficult -- we can simply add the load values. Traffic forking is the process that creates the sub series from the input series (see Figure 4.8). For now consider this as a task of the pluggable model.

The execution of the simulator differs from the usual way. It is divided into several runs. In one run it evaluates a whole time series (as input load) on a router. In the next run it takes another router. The selection of the next router to evaluate is important. If we want to check a network path, then it is obvious that we don't have to check (and compute) all routers only those are on the path and those that have some effect on the path. This fits into the 'data driven' processing paradigm as a step can be executed only if all the inputs are available. Fortunately we always will be able to start the processing since we will have all input for an ingress router from the measurements. To compute the outgoing load on an egress node first we have to do the computation on all other ingresses.

4.4.2 Models

4.4.2.1 Router Model

A router model in this simulator is a processing entity that takes basically two time series as input and generates as many time series as output as many connections the router has.

Inputs:

- Load: multicolumn matrix, describes the load in every service class (or in aggregate) at different time points.
- Distribution: multicolumn matrix: describes the distribution of the traffic at the same time points as the load matrix.

The internal behaviour of the model can be anything but it has to be able to produce the outputs according to the distribution matrix.

As mentioned above, forking the traffic is a task of the router model. It is done based on the distribution matrix but some extra attention is needed by the directed traffic. Directed traffic is a part of the overall load on the router but its destination is known. (E.g. new traffic request.) Therefore this traffic shouldn't be distributed according to the distribution matrix, it simple has to be 'forwarded' to the appropriate output. (First it has to be subtracted from the overall load, the overall load should be distributed then the directed traffic has to be added to the appropriate output.) To handle these traffic flows an interface of the model is needed through which these flows can be described for the model.

4.4.2.2 Link Model

From the viewpoint of the architecture a link model differs from a router model only in one attribute: it has only one output (time series). As a consequence it doesn't need the distribution matrix as input. Indeed we can include it in the end of the router model.

4.4.2.3 Generating the Simulation

Generation and execution such a simulator can be done quite automatically in a few steps.

1. Generating the routers and links based on the topology information. – Input 1
2. Selecting the appropriate models (if more than one exists). – Input 2, 5
3. Calibrating the models (computing the parameters).
4. Calculating (predicting) the input time series (load and load distribution). – Input 4
5. Scenario specific tuning. – E.g. new traffic configuration
6. Run the simulation.
7. Evaluate the results. – Input 3

4.4.2.4 Generating the Topology

We can identify the edge routers that are the nodes in the simulation based on user input. The user should describe the scope of the examination by defining a set of routers -- a domain. Each node should be connected to all the others.

4.4.2.5 Selecting Models

Different models may model the operation of an edge router or a link. The selection of the models may depend on the scenario, the time scale, the load or any other parameter.

4.4.2.6 Calibrating the Models

After we selected the appropriate models we have to define the parameter values of the model. This may be done according to input 2 and 5 or to the input time series (or both). The models should work so that the state of the simulated network should be the same as the state of the measured one expressed by the time series.

4.4.2.7 Prediction

Prediction of time series is needed if we have to simulate future scenarios. This can be done using sophisticated analyzing tools that can tell the trends, the seasonality, etc. of the time series.

4.4.2.8 Scenario Setup

Some scenario specific actions are needed to set up the simulator. E.g. we have to define the directed flows (see 4.4.2.1) to examine the effect of new traffic; or changing the load distributions can simulate route changes that has not happened in the real network.

4.4.2.9 Running and Evaluation

First we have to select which nodes are of interest then the simulator must compute the output time series for these nodes.

Evaluation means the processing of the output time series: compute the QoS parameters and set it against the SLA parameters.

4.5 Inter-Domain Topology and Routing Integration in the Simulation Environment

4.5.1 BGP Modelling and Simulation Approaches

The Border Gateway Protocol (BGP) ([RFC1771], [Ste98]) is currently the inter-domain routing protocol used to maintain connectivity between autonomous systems. BGP is a policy-based, path-vector protocol that distributes route information between Autonomous Systems (ASs).

Consideration of BGP-4 abstractions in the inter-domain simulation environment is important because BGP-4, through routing and topology changes (due to routing policies, failures, etc.), is able to change the inter-domain QoS and traffic and to change performance of applications.

A main concern of inter-domain routing performance is the rather loose BGP specification [RFC1771], which allows the application of different tuning techniques and parameters (e.g. timer settings) but leaves their implementation and the settings unspecified. Since the introduction of BGP, its convergence properties have been investigated. A theoretical proof of BGP-4 routing divergence was published in [Var00], where Varadhan et al. show that independent route selection protocols like BGP may exhibit persistent oscillations and never converge to a single route due to conflicting policies. Characteristics of routing convergence and stability have a high impact on the network's ability to

perform route repairs quickly and to provide good performance in the presence of network faults. Currently inter-domain route repair convergence in the Internet may have a latency on the order of several minutes. During the latency period end-user traffic quality is highly degraded.

Large scale simulations based on BGP-4 models have been done to study performance issues such as interaction of different BGP parameter settings for dynamic topologies and effects of different techniques on BGP's convergence properties. In order to study BGP-4 performance issues, different BGP-4 simulation models have been proposed, typically including some fundamental mechanisms of the BGP-4 protocol but excluding underlying mechanisms such as TCP/IP [Lab00]. [Lab01] observed that high levels of route fluctuation during delayed convergence have an adverse effect on end-to-end traffic delay, resulting in packet loss and intermittent disruption of connectivity. [Gri01] presented simulation results on the relationship between the BGP rate-limiting timer, various BGP implementation techniques and router workload delay, and how they affect convergence time in simple network topologies. [Gri99a] presented a sufficient condition for a convergent routing system and proposed BGP-like Simplified Path-Vector Protocol [Gri00] based on it that is proven to be convergent. The [Gov00] approach is to use simulation of BGP using SSFNet [SSFNet] to study convergence in a small number of simple model topologies.

We give a detailed state-of-the-art of the BGP-4 modelling and simulation approaches in the technical report [Ara02]. Considering the state-of-the-art, the BGP-4 protocol could be integrated in the simulation environments based on different layers of abstraction:

- Quasi static inter-domain routing and topology model without considering BGP-4 in the simulation environment, i.e. no inter-domain routing and topology changes due to BGP-4 model integration.
- BGP-4 models based on abstraction of specific features. For instance, the discrete-event driven BGP-4 simulator [Nyk02] is proposed to study the BGP-4 convergence features.
- Detailed BGP-4 modelling for realistic studies of transport protocols and applications in large scale inter-domain topology and routing environments. An example is the complete BGP-4 model integration in the SSF simulation environment ([Nic99], [Cow99]) in order to simulate different transport protocols in combination with a realistic BGP-4 environment. The BGP implementation of SSFNet [SSFNet] is compliant with the BGP-4 specification in [RFC1771], although it currently does not include some of the optional extensions commonly seen in commercial implementations.

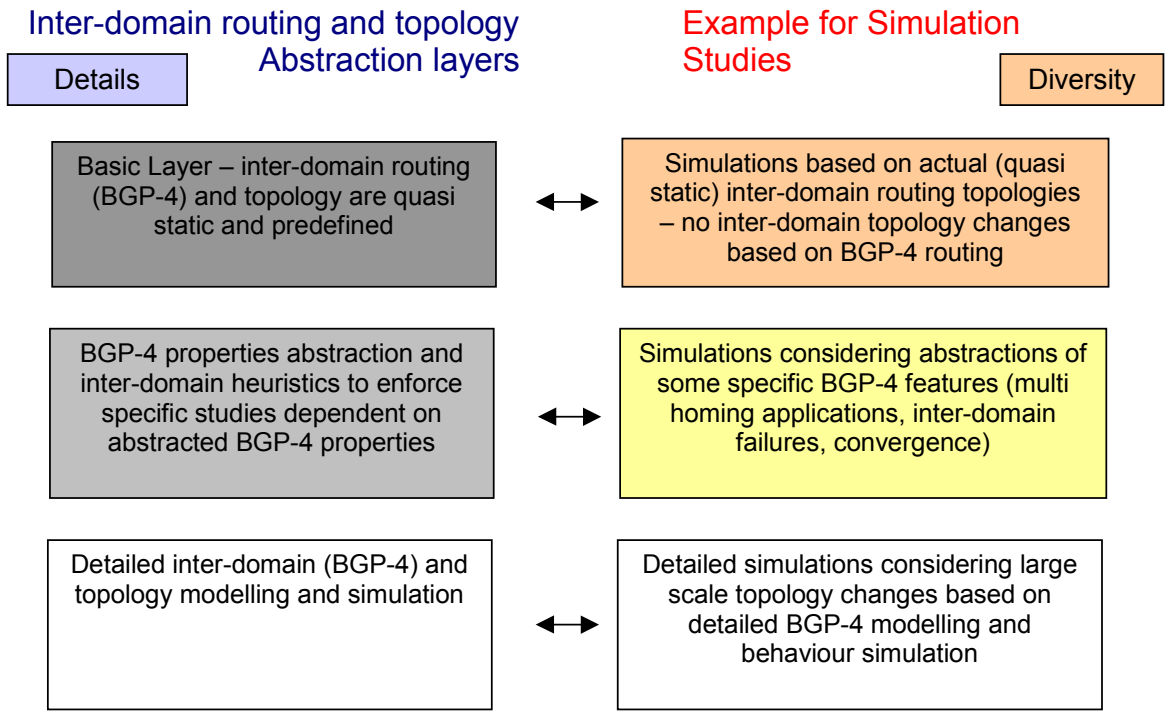

Figure 4.9 -Stepwise approach for abstraction of inter-domain BGP-4 routing and topology

4.5.2 Integration of Inter-Domain Routing and Topology in the InterMON Simulation Environment

4.5.2.1 Possible Workflow Models for the BGP4 Data Collector

The BGP-4 data collector proposed for the InterMON project consists of two programs. The BGP-4 probe *sbgp* collects BGP-4 routing data from the AS routing infrastructure and stores them in binary files. The second program, *route_btoa*, translates these binary files into human readable ASCII files for further processing.

There are two possible ways to use these programs because it is possible to connect both programs through a UNIX pipe and produce human readable ASCII output directly. They are:

- **Time limited routing data collection**: In this model, the SBGP program is launched and programmed to store the routing updates it detects in an intermediate file, which is transmitted to the local database when the measurement is finished.
- **Launch and push**: In this model, the SBGP and ROUTE_BTOA programs are launched and connected through a UNIX pipe. During the time the routing probe is working, routing updates are pushed to the local data base when they are received.
-

4.5.2.2 View of the Internet

The data collected by the BGP4 tools yields the following view of the Internet.

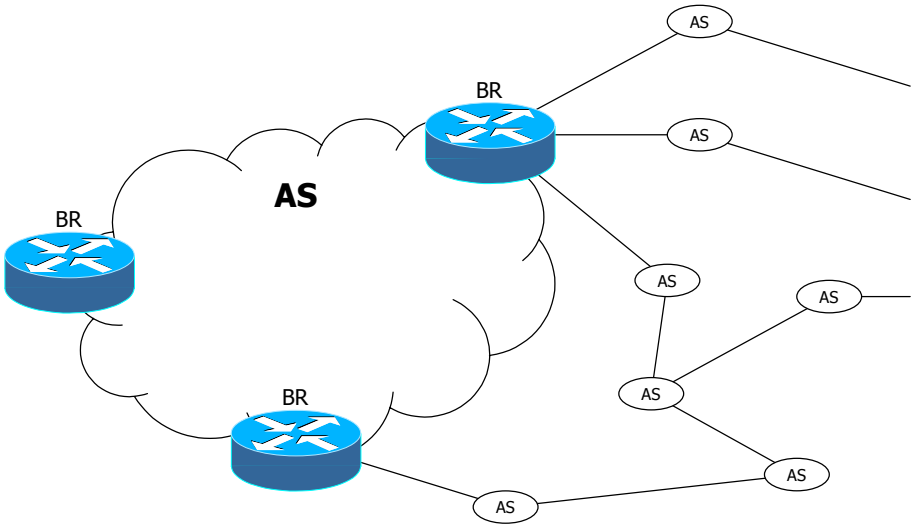

Figure 4.10 - View of the Internet of BGP-4 data collection tools

Common practice design rules mandate that the border routers appear as the next hop for external networks in the BGP4 data. Routes injected by the AS into the Internet (AS-path = ^$) will behave differently, because the router injecting the route into the BGP4 protocol doesn't necessarily have to be the router which has the network directly connected to it. This behaviour is compatible with the black box model and the proposed traffic matrix with entries reserved to AS internal traffic sources and sinks.

5 References

[Ahn96] Jong Suk Ahn and Peter B. Danzig, *Packet network simulation: speedup and accuracy versus timing granularity*, IEEE/ACM Transactions on Networking, Volume 4, No. 5 (Oct. 1996) Pages 743-757

[Amm99] M. H. Ammar, G. F. Riley, R. M. Fujimoto, *A generic framework for parallelization of network simulations*, in Proceedings of the Seventh International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, (MASCOTS)College Park, MD. October 1999, October 1999

[Ani82] D. Anick, D. Mitra, and M.M. Sondhi, *Stochastic theory of a data-handling system with multiple sources*, The Bell System Technical Journal , vol. 61, no. 8, pp. 1871 - 1894, Oct. 1982

[Ara02] P. A. Aranda Gutierrez, I. Miloucheva, U. Hofmann, *Efficient large scale simulation considering inter-domain routing (BGP-4) modelling*, Technical Report SR-ANC-M0I1, December 2002

[Ast98] K.J. Aström, H. Elmqvist, S.E. Mattsson, *Evolution of continuous time modelling and simulation*, 12th European Simulation Multiconference , ESM 98, June 16-19, 1998, Manchester, UK

[Bag98] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng,J. Martin, H. Y. Song, *PARSEC: A parallel simulation environment for complex systems*, IEEE Computer, 31(10):77-85, October 1998

[Bal97] Hari Balakrishnan, Mark Stemm, Srinivasan Seshan, Randy H. Katz, *Analyzing stability in wide-area network performance*, in SIGMETRICS/Performance, 1997

[Bar98] Paul Barford and Mark E. Crovella, *Generating representative Web workloads for network and server performance evaluation*, in Proceedings of Performance'98/ACM SIGMETRICS '98, pages 151160, June 1998

[Bar98] Paul Barford and Mark E. Crovella, *Generating representative Web workloads for network and server performance evaluation*, in Proceedings of Performance'98/ACM SIGMETRICS '98, pages 151160, June 1998

[Bel02] Boris Bellalte, Miquel Oliver, David Rincon, *Capacity and traffic analysis of voice services over GPRS mobile networks*, Technical University of Catalonia, University Pompeu Fabra, Spain 2002

[Ber02a] G. Bergholz, *Signalflußsimulation für Nachrichtenverkehrsmodelle*, Technical Report SR-ANC-B0I1, September 2002

[Ber02b] G. Bergholz, *Mehrklassen-Signalflußsimulation für Nachrichtenverkehrsmodelle*, Technical Report SR-ANC-B0I2, December 2002

[Ber02c] G. Bergholz, U. Hofmann, I. Miloucheva, , P. Haber, C. Brandauer, *InterMON Rate and Time Continuous Fluid Simulation Technology (RTC-SIM) compared with current fluid simulation research*, Technical Report SR-ANC-B0I3, December 2002

[Bon98] T. Bonald, *Comparison of TCP Reno and TCP Vegas via fluid approximation*, Technical Report 3563, INRIA, 1998

[Bre00] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John S. Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, Haobo Yu. *Advances in network simulation*, IEEE Computer,33(5):5967, 2000

[CAIDA] *CAIDA : Internet measurement infrastructure*, http://www.caida.org/analysis/performance/measinfra

[Cao00a] J. Cao, D. Davis, S. Wiel, B. Yu, *Time-varying network tomography : Router link data*, Journal of the American Statistics Association, 95(452):1063–1075, February 2000

[Cao00b] J. Cao, Scott Vander Wiel, Bin Yu, Zhengyuan Zhu, *A scalable method for estimating network traffic matrices*, Bell Labs Tech. Report, 2000

[Cao01] Jin Cao, William S. Cleveland, Dong Lin, Don X. Sun, *On the nonstationarity of internet traffic*. In SIG-METRICS/Performance, pages 102–112, 2001

[CAR] Cisco white paper, *Committed Access Rate*, http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/car.htm

[Car96] Robert Carter and Mark Crovella, *Measuring bottleneck link speed in packet-switched networks*, in *PERFORMANCE '96*

[Cha81] K. M. Chandy and J. Misra, *Asynchronous distributed simulation via a sequence of parallel computations*, Communications of the ACM, 24(11):198-205, April 1981

[Che02] T. P. Chen, T. Chen, *Markov Modulated Punctured Autoregressive Processes for Traffic and Channel Modeling*, April 2002

[Che95] Stuart Cheshire and Mary Baker, *Experiences with a wireless network in MosquitoNet*, in Proceedings of the IEEE Hot Interconnects Symposium '95, August 1995

[Cis02] Cisco Document Server, *Traffic analysis for Voice over IP*, posted September 2002

[Cow99] James H. Cowie, David M. Nicol, Andy T. Ogielski, *Modelling the global internet*, Computing in Science and Engineering , 1(1):4250, January/February 1999

[Dow99] Allen B. Downey, *Using Pathchar to estimate internet link characteristics*, in SIGCOMM, pages 222–223, 1999

[Dut01] Debojyoti Dutta, Ashish Goel, John Heidemann, *Faster Network Design with Scenario Prefiltering*, ISI-Technical-Report-550, November 15, 2001

[EITO02] *European Information Technology Observatory – EITO 2002*

[Flo01] Sally Floyd and Vern Paxson, *Difficulties in simulating the Internet*, ACM/IEEE Transactions on Networking, 9(4):392–403, February 2001

[Gad02] S. Gadde, J. Chase, A. Vadat, *Coarse-Grained Network Simulation for Wide-Area Distributed Systems*, Duke University, Technical Report, 2002

[Gao00] Lixin Gao, *On inferring autonomous system relationships in the Internet*, in Proc. IEEE Global Internet Symposium, November 2000

[Gar01] J-M. Garcia, D. Gauchard, O. Brun, P. Bacquet, J. Sexton, E. Lawless, *Differential Traffic Modelling and Distributed Hybrid Simulation*, Calculateurs paralleles, Vol 18-n 3/2001

[Goder] D. Goderis et al, *Service Level Specification Semantics and Parameters*, Internet Draft, draft-tequila-sls-02.txt

[Gon00] W. Gong, J. Kurose, V. Misra, D. Towsley, *Fluid Methods for Modeling Large, Heterogeneous Networks* , Slides m U. Massachusetts http://gaia.cs.umass.edu/fluid

[Gov00] Ramesh Govindan and Hongsuda Tangmunarunkit, *Heuristics for Internet map discovery*, In Proc. IEEE INFOCOM, March 2000

[Gov97] Ramesh Govindan, Cengiz Alaettinoglu, Deborah Estrin, *Self-configuring active network monitoring (SCAN)*, White paper, February 1997

[Gri00] Timothy G. Griffin and Gordon Wilfong, *A Safe Path Vector Protocol*, in Proc. of INFOCOM, pages 490499. IEEE, March 2000

[Gri01] Timothy G. Griffin and Brian J. Premore, *An Experimental Analysis of BGP Convergence Time*, in Proc. of ICNP, November 2001

[Gri99a] Timothy G. Griffin, F. Bruce Shepherd, Gordon Wilfong, *Policy Disputes in Path Vector Protocols*, in Proc. of ICNP, November 1999

[Gri99b] Timothy G. Griffin and Gordon Wilfong, *An Analysis of BGP Convergence Properties*, in Proc. of ACM SIGCOMM , pages 277288, September 1999

[Guo00] Yang Guo, Weibo Gong, Don Towsley, *Time-stepped hybrid simulation (TSHS) for large scale networks*, in Proceedings of IEEE INFOCOM 2000 , March 2000

[Gur91] R. Gurenefelder, J. P. Cosmas, S. Manthrope, A. Odinma-Okafor, *Characterization of video codecs as autoregressive moving average processes and related queuing system performance*, IEEE Journal on Selected Areas in Communications, 9:284–293, 1991

[Gur97] K. Gurney, *An Introduction to Neural Networks*, UCL Press, 1997

[Hay99] S. Haykin, *Neural Networks*, Prentice-Hall, 1999, 2nd Edition

[Hef86] H. Heffes and D. M. Lucantoni, *A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance*, IEEE Journal on Selected Areas in Communications, 4:856– 868, 1986

[Hei00] John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun-chan Lan, Ya Xu, Wei Ye, Deborah Estrin, Ramesh Govindan, and John Heidemann, *Effects of Detail in Wireless Networks*

[Hua98] Polly Huang, Deborah Estrin, John Heidemann, *Enabling large-scale simulations: selective abstraction approach to the study of multicast protocols*, in Proceedings of the International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, pages 241248. IEEE, July 1998

[Hus99a] Geoff Huston, *Interconnection, peering and settlements: Part I*, Internet Protocol Journal, 2(1), June 1999

[Hus99b] Geoff Huston, *Interconnection, peering and settlements: Part II*, Internet Protocol Journal, 2(2), June 1999

[Jac97] Van Jacobson, *Pathchar*, MSRI talk, April 1997

[Jef85] D. R. Jefferson, *Virtual time*, ACM Transactions on Programming Languages and Systems, 7(3):404-425, July 1985

[Jia00] Wenyu Jiang, Henning Schulzrinne, *Analysis of On-Off Patterns in VoIP and their effect on voice traffic aggregation,* Department of Computer Science, Columbia University 2000

[Kes93] G. Kesidis and J. Walrand, *Quick Simulation of ATM Buffers with On-off Markov Fluid Sources*, ACM TOMACS, V3, N3, pp. 269-276, Jul 1993

[Kes95] G. Kesidis and A. Singh, *An Overview of Cell-Level ATM Network Simulation*, Proc. High Performance Computer Systems, Montreal, PQ, pp. 202-212, July 1995

[Kes96] G. Kesidis, A. Singh, D. Cheung, W. Kwok, *Feasibility of Fluid-event-driven simulation for ATM Networks*, Proc. IEEE Globecom 1996

[Kru98] M. M. Krunz, A. M. Makowski, *Modeling Video Traffic using M*G/∝ Input Processes: A Compromise between Markovian and LRD Models*, IEEE Journals on Selected Areas in Communications, 1998

[Kum98] K. Kumaran and D. Mitra, *Performance and Fluid Simulations of a Novel Shared Buffer Management System*, Proc. IEEE INFOCOM 98, March 1998

[Lab00] Craig Labovitz, Abha Ahuja, Abhijit Bose, Farnam Jahanian, *Delayed Internet Routing Convergence*, in Proc. of ACM SIGCOMM, volume 30, pages 175187, October 2000

[Lab01] C. Labovitz, R. Wattenhofer, S. Venkatachary, A. Ahuja, *The impact of Internet policy and topology on delayed routing convergence*, in Proc. IEEE INFOCOM, April 2001

[Lab01] Craig Labovitz, Ahba Ahuja, Roger Wattenhofer, Srinivasan Venkatachary, *The Impact of Internet Policy and Topology on Delayed Routing Convergence*, in Proc. of INFOCOM , pages 537546. IEEE, April 2001

[Lab99] Craig Labovitz, G. Robert Malan, Farnam Jahanian, *Origins of Internet Routing Instability*, in Proc. of INFOCOM, IEEE, June 1999

[Lai00] Kevin Lai and Mary Baker, *Measuring link band-widths using a deterministic model of packet delay*, in SIGCOMM, pages 283–294, August 2000

[Lai01] Kevin Lai and Mary Baker, *Nettimer: A tool for measuring bottleneck link bandwidth*, in Proceedings of the USENIX Symposium on Internet Technologies and Systems, March 2001

[Lai99] Kevin Lai and Mary Baker, *Measuring bandwidth*. In INFOCOM (1), pages 235–245, 1999

[Lan02] K. Lan, J. Heidemann, *Rapid model parametrization from traffic measurements*, August 2002, Technical Report ISI-TR-561

[Lia01] Q. Liang, J. M. Mendel, *MPEG Video Traffic Modeling and Classification Using Fuzzy Techniques*, IEEE Transactions on Fuzzy Systems, February, 2001

[Liu01] Benyuan Liu, Daniel R. Figueirido, Yang Guo, Jim Kurose, Don Towsley, *A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation*, in Proceedings of IEEE Infocom 2001, April 2001

[Liu01] Benyuan Liu, Daniel R. Figueiredo, Yang Guo, Jim Kurose, Don Towsley, *A Study of Networks Simulation Efficiency: Fluid Simulation vs. Packet-level Simulation*, Proceedings of IEEE Infocom 2001

[Liu99] B. Liu, Y. Guo, J. Kurose, D. Towsley, and W. Gong, *Fluid simulation of large scale networks: issues and tradeoff*, in PDPTA '99, Las Vegas, NV, June 1999, pp. 2136 - 2142

[Liu99] B. Liu, Y. Guo, J. Kurose, D. Towsley, W. Gong, *Fluid Simulation of Large Scale Networks: Issues and Tradeoffs*, Proc. International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV, June 1999

[Luc94] D. Lucantoni, M. Neuts, A. Reibman, *Methods for performance evaluation of VBR video traffic models*, IEEE/ACM Trans. Networking, 2:176–180, 1994

[Mag88] B. Maglaris, D. Anastassiou, G. Karlsson P. Sen, J. D. Robbins, *Performance models of statistical multiplexing in packet video communications*, IEEE Trans. on Comm., 36(7):834–844, 1988

[Mah97] B. Mah, *An empirical model of HTTP network traffic*, in Proceedings of the IEEE Infocom, pages 592–600, Kobe, Japan, April 1997

[Mah98] B. Mah, P. Sholander, L. Martinez, L. Tolendino, *LPB; An internet protocol benchmark using simulated traffic*, in Proceedings of MASCOTS '98, Montreal, Canada, August 1998. IEEE

[Mah99] Bruce A. Mah, *Pchar: Child of pathchar*, presented at the DOE NGI testbed workshop, Berkeley, ca, 21 July 1999

[Mar63] Marquardt, D., *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, SIAM J. Appl. Math., 11, 1963, pp. 431-441

[Mat96] M. Mathis and J. Mahdavi, *Diagnosing internet congestion with a transport layer performance tool*, in Proceedings of INET '96, Montreal, June 1996

[Mcc00] Sean McCreary and K. Claffy, *Trends in wide area IP traffic patterns: A view from ames internet exchange*, 13th ITC Specialist Seminar, pages 1–11, September 2000

[Mel02] Benjamin Melamed, Shuo Pan, Yorai Wardi, *Hybrid discrete-continuous fluid-flow simulation*, 2002

[Min69] M. Minsky, S. Paper, *Perceptrons*, MIT Press, Cambridge MA, USA, 1969

[Mol02] Maurizio Molina, Paolo Castelli, Gianluca Foddis , *Web Traffic Modeling Exploiting TCP Connections' Temporal Clustering through HTML-REDUCE,* IEEE Network, May 2002, pp. 46-55

[MSK] J. van der Merwe, S. Sen, C. Kalmanek, *Streaming Video Traffic: Characterization and Network Impact*

[Nic99] David Nicol, Michael Goldsby, Michael Johnson, *Fluid-based simulation of communication networks using SSF*, in Proceedings of the 1999 European Simulation Symposium, October 1999

[Nic99] David Nicol, Michael Goldsby, Michael Johnson, *Fluid-based simulation of communication networks using SSF*, in Proc. 1999 European Simulation Symposium, Erlangen-Nürnberg, Germany, Oct. 1999

[Ns-2] *The Network Simulator ns-2*, http://www.isi.edu/nsnam/ns

[Nyk02] Johan Nykvist and Lenka Carr-Moty ckov´a, *Simulating Convergence Properties of BGP*, Division of Computer Science and Networking, Department of Computer Science and Electrical Engineering Luleå University of Technology, Luleå, Sweden, 2002

[Pax97] Vern Paxson, *Measurements and analysis of end-to-end internet dynamics*, Ph.D. thesis, University of California, Berkeley, April 1997

[Pax99] Vern Paxson, *End-to-end internet packet dynamics*, IEEE/ACM Transactions on Networking, 7(3):277–292, 1999

[PaxAr] Vern Paxson, *Internet Traffic Archive*, http://www.acm.org/sigcomm/ita

[R] R-Project, www.r-project.org

[RFC1771] Y. Rekhter and T. Li, *A Border Gateway Protocol, RFC 1771 (BGP version 4)*, March 1995

[RFC2211] J Wroclawski, *Specification of Controlled Load Network Element Service*, RFC2211

[RFC2212] S. Shenker et al, *Specification of Guaranteed Quality of Service*, RFC2212

[RFC2439] C. Villamizar, R. Chandra, R. Govindan, *BGP route flap damping*, RFC 2439, 1998

[RFC2475] S. Blake et al, *An Architecture for Differentiated Services*, RFC2475

[Ril01] G. Riley, M. Ammar, R. Fujimoto, D. Xu, K. Perumalla, *Distributed network simulations using the dynamic simulation backplane*, 2001

[Ros95] O. Rose, *Simple and Efficient Models for Variable Bit Rate MPEG Video Traffic*, Report No. 120, July 1995. http://ww-info3.informatik.uni-wuerzburg.de/TR/tr120.ps.gz

[Ros99a] D. Ros and R. Marie, *Estimation of end-to-end delay in high-speed networks by means of fluid model simulations*, in 13th European Simulation Multiconference, Warsaw, June 1999

[Ros99b] D. Ros and R. Marie, *Loss characterization in high-speed networks through simulation of fluid models*, in SPECTS'99, Chicago, IL, USA, July 1999

[Rum86] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Learning Internal Representations by Error Propagation*, Volume 1 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 318--362. MIT Press, Cambridge MA, USA, 1986

[Salta] EURESCOM Project P1115 (Saltamontes), *Traffic Engineering in Differentiated Services Networks*, (vol. 3)

[Simulink] *SimuLink*, http://www.sciencesoftware.com/simulink.asp

[Simulink] www.sciencesoftware.com/simulink.asp

[SPlus| S-Plus, http://www.insightful.com

[SSFNet] www.ssfnet.org

[Ste98] John W. Stewart. *BGP4: Inter-Domain Routing in the Internet*, Addison-Wesley, 1998

[Tur01] D. S. Turaga, T. Chen, *Hierarchical Modeling of Variable Bit Rate Video Sources*, Packet Video 2001

[Var00] Kannan Varadhan, Ramesh Govindan, Deborah Estrin, *Persistent Route Oscillations in Inter-Domain Routing*, Computer Networks 32(1):116, 2000

[Var96] Y. Vardi, *Network tomography : Estimating source-destination traffic intensities from link data*, Journal of the American Statistics Association, 91(433):365– 377, March 1996

[Var96] K. Varadhan, R. Govindan, D. Estrin, *Persistent route oscillations in inter-domain routing*, ISI technical report 96-631, USC/Information Sciences Institute, 1996

[Vil94] M. Vill´en-Altamirano and J. Vill´en-Altamirano, *RESTART: a straight- forward method for fast simulation of rare events*, in Proceedings of the 1994 Winter Simulation Conference, Lake Buena Vista, Florida - USA, Dec. 1994, pp. 282 - 289

[War00] Y. Wardi and B. Melamed, *Loss Volume in Continuous Flow Models: Fast Simulation and Sensitivity Analysis*, Proceedings of 8-th IEEE Mediterranean Conference on Control and Automation (MED-2000), Patras, Greece, July 17-19, 2000

[War01] Y. Wardi and B. Melamed, *Variational Bounds and Sensitivity Analysis of Continuous Flow Models*, J. of Discrete Event Dynamic Systems, 2001

[War99] Y. Wardi and B. Melamed, *Continuous Flow Models: Modeling, Simulation and Continuity Properties*, Proceedings of 38-th IEEE CDC, 3439, Phoenix, Arizona, December 7-10, 1999

[Wer74] Werbos, P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences,* PhD. Dissertation, Appl. Math., Harvard University, USA, 1974

[Wil98] W. Willinger, V. Paxson, M. Taqqu, *Self-similarity and heavy-tails: Structural modeling of network traffic*, in *Self-Similarity and Heavy-Tails: Structural Modeling of Network Traffic, in A Practical Guide To Heavy Tails: Statistical Techniques and Applications,* edited by R.J. Adler, R.E. Feldman and M.S. Taqqu. ISBN 0-8176-3951-9. Birkh auser, Boston, 1998

[Xio98] Xiong Simin, Liang Jinsong, Yang Zhimin and Lei Zhengming, *Video Traffic Modeling Based on RBF Networks*, Proceedings of 1998 International Conference on Communication Technology

[Yan97] Anlu Yan and Wei-Bo Gong, *Fluid Simulation for High Speed Networks*, Technical Report TR-96-CCS-1, Dept. of ECE, Univ. of Massachusetts. Proceedings of the 15th International Teletraffic Congress, Washington, D.C., June, 1997

[Yan98] A. Yan and W.B. Gong, *Time-driven fluid simulation for high-speed networks with flow-based routing*, in Proc. of the Applied Telecommunications Symposium, Boston, MA, Apr. 1998, pp. 153 - 158

[Yan99] A. Yan and W.B. Gong, *Fluid Simulation for High Speed Networks with Flow-Based Routing*, IEEE Trans. on Information Theory 45, 1588-1599, 1999

[Yuk00] M. Yuksel, B. Sikdar, K. S. Vastola, B. Szymanski, *Workload generation for NS simulations of wide area net works and the internet*, in Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS) part of Western Multi-Conference (WMC), pages 93–98, San Diego, CA, 2000

[Zha01] Y. Zhang, N. Duffield, V. Paxson, S. Shenker, *On the constancy of internet path properties*, in Proceedings of ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco Bay Area, November 2001