[HeSchSt02] Oliver Heckmann, Jens Schmitt, Ralf Steinmetz; **Multi-Period Resource Allocation at System Edges;** 10th International Conference on Telecommunication Systems Modelling and Analysis (ICTSM10) Monterey, USA, October 2002.

Multi-Period Resource Allocation at System Edges

Oliver Heckmann, Jens Schmitt, and Ralf Steinmetz
Multimedia Communications (KOM), Darmstadt University of Technology, Merckstr. 25 • 64283 Darmstadt • Germany
http://www.kom.e-technik.tu-darmstadt.de
Email:{Oliver.Heckmann, Jens.Schmitt, Ralf.Steinmetz}@kom.tu-darmstadt.de

Abstract

The Internet consists of a variety of interconnected heterogeneous networks managed by independent providers. At the edges between two networks resources are allocated. In this paper, we present a framework and taxonomy for problems at these edges like admission control and provider selection; we call them multi-period resource allocation problems at system edges (MPRASE). We look at several problem incarnations of this framework and show that many of those problems - including well-known problems in the are of networking - are sub- or dual problems of each other and that it is useful to treat them in an integrated fashion.[1]

1. Introduction

The Internet consists of a variety of interconnected heterogeneous networks (autonomous systems, AS), managed by multiple independent providers. Both the number of ASes as well as the average number of ASes a given AS is peering with is increasing at a fairly high rate. The number of ASes rose from 909 in 9/95 to 4427 in 12/98 and 7563 in 10/00 [17, 18]. Similarly, the average peering degree, i.e., the number of providers a certain provider has peering agreements with, rose from 2.99 in 9/95 to 4.12 in 12/98. It is also very notable that a single provider may peer with up to 1000 other providers [17].

The highest cost factors of ISPs are peering costs and line costs [33]. For the increasing number of peering agreements resources have to be provisioned. In particular, an optimization of resource allocations becomes a competitive factor for Internet providers. In this paper, we deal with a general problem class called multi-period resource allocation at system edges (MPRASE) for which the peering providers have been the motivating scenario, although some other network Quality of Service (QoS) problems also fit into this problem class as we will discuss later on.

[1] This work has been partly sponsored by the DFN (german research network provider, www.dfn.de) as part of the LFTSQoS project (www.letsqos.de).

At first we present an overview of the MPRASE problem framework and a taxonomy for the MPRASE problems that can be used to efficiently identify, classify and mathematically describe a resource allocation problem at a provider's edge.

The MPRASE problems include problems in the area of admission control, reservation in advance, renegotiable services, token bucket fitting, provider selection and RSVP/IntServ [6] over DiffServ/Bandwidth-Broker [3, 2] problems. We present a number of these problems in varying detail and show that it is very useful to treat them in an integrated manner as algorithms can be reused and complex problems relaxed towards simpler problems of the framework.

In section 2 we present the MPRASE framework and its taxonomy. In section 3 we present two selected abstract problem incarnations from the framework, the first is very complex and basically encompasses all the problems discussed later in this paper. The second is the smallest non-trivial MPRASE problem incarnation which is discussed in detail as it is the basis for many of the other problems discussed in this paper. Section 4 then presents some uncertain MPRASE problem incarnations while in section 5 several deterministic ones are presented. We conclude with a summary and an outlook in section 6.

2. The MPRASE Framework and Taxonomy

In this section, we introduce a general structural model which tries to capture all the different facets of MPRASE problems. This model allows us to derive a taxonomy along its components.

2.1 Generalized Problem Structure Model

Figure 1 shows the overall structural model of the general class of MPRASE problems. On one side, there are *customers* that have a certain demand for network resources. These network resources are provided by the opposite side,

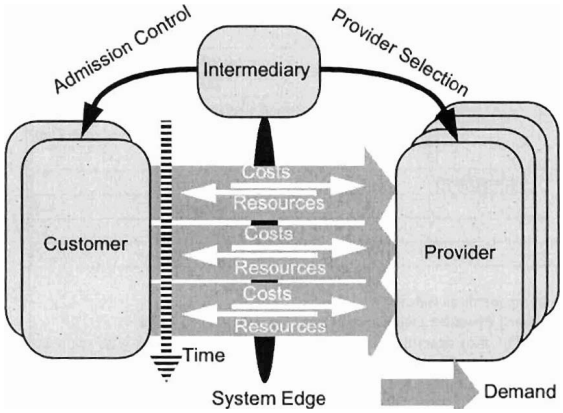

Figure 1: MPRASE problem structure

which we call the *providers*. Note that the customers can be end-users or themselves providers for other customers (at another edge). There is a third party involved, the *intermediary* instance that is located at the edge. The intermediary tries to mediate between the two by selecting providers on the one hand and enforcing admission control of the customers on the other hand. Note that the logical separation of the intermediary instance from customer and provider does not necessarily imply that it may not belong to either customer or provider premises, in fact, this will usually be the case. If the intermediary is, e.g., imposing admission control, he will usually belong to the provider's premises.

Allocating the resources to satisfy the customers demand incurs certain *costs* which need to be accounted for by the customers. These costs can be real (monetary) costs, computational costs, purely fictive / calculatory costs or a mix of those.

The demand changes over the *time* so that resources might be reallocated. Let us now look at the different components / submodels of the structural model for MPRASE.

2.2 The 6 Submodels

2.2.1 Customer. The customer model of the MPRASE model captures the number of customers, i.e., whether a single or multiple customers are considered, and the flexibility of the demand, i.e., whether demand may be dissatisfied or be served with a degraded quality. In the case that multiple customers exist the total demand D is the sum of the individual customers' demand that is $D = \sum_{i=1}^{n} d_i$. With an admission control mechanism the number of served customers n is becoming variable while with degraded quality the amount of the demand d_i of a customer i that is satisfied by the provider becomes flexible itself.

Parameter	Value	Abbrev.
Number of Customers	*single Customer*	*1*
	multiple Customers	*N*
Flexibility of Demand	*inflexible (satisfied 100%)*	-
	dissatisfied/admission control (satisfied 0 or 100%)	*AC*
	degraded quality (satisfied between 0 and 100%)	*DQ*

Table 1: Customer Model

The taxonomy for the customer model is displayed in Table 1. We describe the customer submodel by specifying both parameters of Table 1. The abbreviation "-" in that table means that this value does not need to be specified, it is the default. A simple customer model consisting of a single customer with inflexible demand would therefore be expressed by "1" while a model containing multiple customers that accept degraded quality are identified with "N_{DQ}" or "N, DQ".

2.2.2 Provider. The provider model encompasses the number of providers and whether they are modelled as having limited or unlimited capacity. While the latter is unrealistic it can be a simplifying, yet valid assumption for the case where supply exceeds demand with very high probability.

Parameter	Value	Abbrev.
Number of Providers	*single Provider*	*1*
	multiple Providers	*N*
Capacity	*unlimited*	
	limited	*Cap*

Table 2: Provider Model

The taxonomy for the provider model is displayed in Table 2. A simple provider model with a single provider that has unlimited capacity would therefore be expressed by "1" while a model containing multiple customers with limited resources are identified with "N_{Cap}" or "N, Cap".

2.2.3 Resource. This component models the resources, i.e., whether they are one- or multidimensional or whether they are provided on a deterministic or statistical basis.

Parameter	Value	Abbrev.
Dimensions	*one-dimensional Resource*	*1*
	multi-dimensional Resource	N^{Type}
Stochastic Behaviour	*deterministic*	-
	statistical	*Stat*

Table 3: Resource Model

A one-dimensional deterministic resource like guaranteed bandwidth is expressed by "1", a token bucket would be described by "N^{TB}". The taxonomy is summarized in Table 3, we specify abbreviations for some well-known multi-dimensional resource models in Table 4.

Parameter	Value	Abbrev.
Buffer + Rate	*Token Bucket*	*TB*
	n-Level Token-Bucket (n=2 equals a TSPEC)	*n-TB*
	Leaky Bucket	*LB*

Table 4: Some Multi-Dimensional Resource Models

2.2.4 Cost. The cost model seizes the cost structure for allocation requests, i.e., whether these incur certain setup or transactional costs or whether the number of requests is bounded and how variable costs for resource allocations are modelled, e.g., linearly or non-linearly. Please note that costs do not have to be monetary costs, they can also reflect imputed or fictive/ calculatory costs. Please note, too, that profit is in effect negative costs and is thus included in the cost model. The term "cost" is also used if we refer to purely technical constraints. [2]

Table 5 specifies the different types of costs that can be used, Table 6 specifies the properties of those cost types. A budget constraint means the following: for the related cost term that only a limited budget is available which can not be exceeded. This can also be used in a plain technical context: If all setup costs are 1 and the budget is N we only allow a maximum of N reallocations/allocations. With a time constraint we describe that - again using the setup costs as example - there has to be a certain time interval between two reallocations.

[2] This is because if we model the problem mathematically we need the same kind of variable to measure the number of reallocations independent of whether we use it for calculating real fixed costs or as a technical constraint; see e.g. M1.

Parameter	Value	Abbrev.
*Fixed Costs per **allocation / reallocation** (= Setup Costs)*	*Non-Existent*	
	Existent	*F*
	Infinite Setup Costs[a]	F_∞
*Variable Costs per amount of **allocated** resources per time*	*Non-Existent*	
	Existent	*V*
*Variable Costs per amount of **used** resources per time*	*Non-Existent*	-
	Existent	*U*
*Variable Costs per amount of **requested but not satisfied** resources per time*	*Non-Existent*	
	Existent	*R*
*Variable Costs per **served customer**[b]*	*Non-Existent*	
	Existent	*P*

Table 5: Cost Model Elements

a. Finite fixed costs for the first period and infinite fixed costs for all other periods. This effectively prohibits reallocations and thus simplifies the resulting problem. We introduce this special notation because this simplification will be used quite often in the MPRASE problems below.
b. This will in many problems be a negative term modelling the profit per served customer

Parameter	Value	Addition
Linearity	*Linear*	
	Non-Linear and Convex	*cx*
	Non-Linear and Concave	*cv*
	Otherwise Non-Linear	*nl*
Time dependent costs	*Costs can vary between different periods*	
	Costs remain equal for all periods	=
Cost-Constraint	*Costs are unconstrained*	
	Budget constraint	*budg*
	Time constraint	*time*

Table 6: Cost Model Additions

To specify the cost model in the taxonomy we list all existent cost terms plus the necessary additions for each cost term. Linear fixed and variable allocation costs are described by "FV" while "$F_{nl}V_{nl}$" would denote linear fixed setup costs that are equal for all periods and non-linear changing variable costs.

2.2.5 Intermediary. Note that the intermediary is the component where solution techniques towards MPRASE problems are conceptually located. Mathematically speaking, it captures the target function of the optimization problem that is described by the taxonomy.

Parameter	Value	Abbrev.
Part of the Target Function	*All Cost Terms*	*
	Individual Cost Terms of the Cost Model	*F,V,U,R,C...*

Table 7: Intermediary Model

If all cost terms of the cost model are to be optimized (minimized) this is indicated by "*". A combination of the cost model "$F_{budg}V$" with intermediary model "V" means that only the variable costs are to be minimized, the fixed setup costs only have to remain below their budget constraint.

2.2.6 Edge. The edge model encompasses the nature of knowledge about the problem parameters at the system edge. Deterministic knowledge means that we know the exact values of the parameter for all periods. If the knowledge is stochastic, we do not know the exact value of the parameter for the future periods but have some knowledge of statistical nature about it, e.g., the probability distribution. Discrete stochastic means that the parameter set is chosen from a number of known scenarios. And we speak of total uncertainty if no assumptions about the parameter can be made.

For the taxonomy we specify the parameters that are not deterministic and describe their uncertainty with a small index (S, D or T). So if every parameters is deterministic instead of the future demand which is totally uncertain we would write "D_T". If all parameters are deterministic we write "*".

2.3 The Complete Taxonomy

We can now describe each MPRASE problem incarnation by describing all of the six components as follows:

Parameter	Value	Abbrev.
Parameter	*All*	*
	Cost Term	*F,V,U,R,C...*
	Demand	*D*
	Budget / Technical Constraint(s)	*Budg / Tech*
	Provider's capacity	*Cap*
	...	
Uncertainty	*Deterministic*	-
	Stochastic	*S*
	Discrete Stochastic	*D*
	Total Uncertainty	*T*

Table 8: Edge Model

Customer | Provider | Resource | Cost | Intermediary | Edge.

1|1|1|FV|*|* thus describes the MPRASE problem incarnation with one customer, one provider, a one-dimensional resource, linear fixed setup and variable costs that are to be minimized under deterministic knowledge.

3. Selected Abstract MPRASE Problems

We now present two abstract problem incarnations from the MPRASE framework. The first is very complex and encompasses all problems discussed later in this paper while the second is the smallest non-trivial problem of the framework. We concentrate on the discussion of the second problem as the results come in handy later for the other MPRASE problems.

3.1 General Model: Maximizing Social Welfare at the Edge

3.1.1 Problem Formulation. The overall goal at an edge between a number of customers and providers is to maximize social welfare. Thus we first look at a very general but rather complex MPRASE problem incarnation which models an edge between a number of customers and a number of providers where the intermediary's goal is to maximize social welfare.

We assume that there is a considerable number of customers, the intermediary performs admission control on them. Additionally the providers are allowed to dissatisfy a part of the customers' demand (degraded quality), although doing this imposes costs on them. Thus the customer model is „$N_{AC,DQ}$".

There are multiple providers with limited capacity; the provider model is there-

fore „N_{Cap}". We use a token bucket as resource model: „N^{TB} ". There are fixed setup costs for each (re)allocation. As a technical limit for the reallocations there is a minimum time that has to pass between to reallocations at the same provider. There are variable costs imposed for the token bucket parameters, degraded quality leads to costs as well as rejecting customers leads to lost profit. The cost model is thus „$FF_{time}VRC$". The intermediary tries to maximize social welfare and thus embraces all costs („*") and for ease of description we look at the deterministic version of the problem, leading to „*" as edge model.

In terms of our taxonomy the problem is described by

$N_{AC,DQ} \mid N_{Cap} \mid N^{TB} \mid FF_{time}VRC \mid * \mid *$.

It is formulated in MIP (Mixed Integer Programming, [32]) form as M1. The social welfare is the total utility of the providers and the customers, it is maximized in (1) and consists of the profit for accepting customer i minus the costs for the resource allocation, consisting of the fixed setup costs and the variable costs for the token bucket rate and depth, minus the costs for degraded quality.

For M1 we assume that all the cost terms are non-negative and that the profit p_i of customer i is lower than $\sum_i \sum_t c^u_{it} b_{it}$ so that there is an incentive to impose admission control.

In M1 the constraints (3) to (6) force s_{ijt} to 1 whenever a reallocation is made, indicated by a change in r_{ijt} and/or d_{ijt}. Constraints (7) and (17) set the variable u_{it} to the unsatisfied demand but not smaller than zero. (8) updates l_{ijt}, the tokens in the bucket at the end of period t are the ones left from last period plus the current rate minus the tokens used to satisfy demand as expressed by v_{ijt}. (9) makes sure that there are never more tokens in the bucket than the bucket depth at the end of the period. The provider's maximum rate and bucket depth is accounted for by (10) and (11). (12) is the technical constraint that makes sure that reallocations can only occur once every ΔT periods. (13) to (19) are the non-negativity and binary constraints for the variables.

3.1.2 Solution. As M1 is a MIP problem it can be solved with standard MIP solving techniques like branch and bound with LP relaxation [32]. This is however not necessary. A huge system edge between customers and providers as assumed in M1 cannot be solved centrally in the Internet because of the scalability issues involved. There is no central intermediary in the Internet that could ever manage all requests from the customers. Proposals like [46] that rely on a central intermediary (there called broker) are generally regarded as unrealistic approaches.

The goal of M1 must be aimed at with distributed algorithms. Therefore we do not intend to look for algorithms that solve M1, instead we use M1 to show that a number of problems in the literature are actually subproblems of M1. Thus we prove the generality of the MPRASE framework and show that it is sensible to look at these problems at systems edges in an integrated fashion.

3.1.3 Modelling Subproblems. To change the customer model to „1" parameter I in M1 has to be set to 1. Dropping the „AC" (admission control) in the customer model is reflected by forcing all a_i to 1 in (19). Dropping the „DQ" (degraded quality) is reflected by setting all c^u_{it} to infinity. Changing the provider model to providers with unlimited capacity is done by setting all C^r_{jt} and C^d_{jt} to infinity. To change the provider model to „1" J has to be set to 1.

If the one-dimensional resource model „1" shall be used instead of a token bucket model all C^d_{jt} have to be set to zero. To drop „F_{time}" from the cost model ΔT has to be set to zero. If we only allow one allocation in the first period and forbid all reallocations „F" in the cost model has to be replaced with F_∞. This is done by setting the setup costs c^s_{jt} to infinity for all but the first period.

Many of the possible subproblems of M1 are modeled and solved in the following parts of the paper.

3.2 The Single Provider Problem (SPP)

3.2.1 Problem Formulation. While M1 is the most complex and comprehensive MPRASE problem discussed here, the single provider problem is the most simple non-trivial MPRASE problem. In terms of the MPRASE taxonomy it is **1|1|1|FV|*|***. There is one customer that has one-dimensional capacity demands b_t that must be fully satisfied at every discrete time interval $t = 1,...,T$. The edge is deterministic. Capacity is requested from a single provider who is charging a fixed setup cost c^s_t for each allocation and variable allocation costs c^r_t per reserved capacity unit and period. A new allocation is constituted by a change in the allocated capacity. Allocated capacity is available in the period the allocation is made and in all subsequent periods until the next allocation is made. Note that the allocated and not the actually used capacity causes the costs.

3.2.2 Exact Solution Algorithm. At first we want to look at techniques that guarantee to produce an optimal solution for the SPP.

3.2.2.1 Branch and Bound with Linear Programming (LP) Relaxation.

A standard approach to solve the single provider problem SPP is to use a mixed integer problem solver in order to solve M2. A typical algorithm for solving a mixed integer LP model is a branch and bound algorithm that uses the LP relaxed problem M3 of M2:

The resulting problem can be easily solved with the simplex algorithm. The solution of M2' is a lower bound to the optimal solution of M2. Branching can be

M1 Basic MPRASE Model (Variables and Parameters)

Indices:

- i Index for customers $i = 1, ..., I$
- j Index for providers $j = 1, ..., J$
- t Index for periods $t = 1, ..., T$

Variables:

- r_{ijt} Allocated token bucket rate for customer i by provider j in period t.
- d_{ijt} Allocated token bucket depth for customer i by provider j in period t.
- l_{ijt} Amount of buckets left in token bucket of customer i at provider j at the end of period t.
- s_{ijt} Auxiliary binary variable for accounting the setup costs. Set to 1 if customer i reallocates resources (bucket rate and/or depth) at provider j in period t and to 0 otherwise.
- a_i Binary variable, set to 0 if customer j is rejected by the admission control and to 1 otherwise.
- v_{ijt} Amount of demand by customer j in period t that is satisfied by provider j.
- u_{it} Amount of unsatisfied demand of customer j in period t.

Parameters:

- b_{it} Demand of customer i in period t.
- c^s_{jt} Setup costs of provider j in period t.
- c^r_{jt} Costs per allocated rate of provider j in period t.
- c^d_{jt} Costs per allocated bucket depth of provider j in period t.
- c^u_{it} Costs per unsatisfied demand (degraded quality) of customer i in period t.
- C^r_{jt} Maximum total rate available at provider j in period t.
- C^d_{jt} Maximum total bucket depth available at provider j in period t.
- p_i Profit for accepting customer i.
- $r_{ij0} = 0$. Rate allocated for customer i at provider j before the first period.
- $d_{ij0} = 0$. Bucket depth allocated for customer i at provider j before the first period.
- $l_{ij0} = 0$. Tokens in the bucket of customer i at provider j before the first period.
- M M is a sufficiently high number ($\max\{b_{it} \mid \forall i, t\}$).
- ΔT Time interval that must pass between two (re)allocations.

M1 Basic MPRASE Model

$$\text{Maximize } \sum_i p_i a_i - \sum_i \sum_j \sum_t c^s_{jt} s_{ijt} - \sum_i \sum_j \sum_t c^r_{jt} r_{ijt} \quad (1)$$

$$- \sum_i \sum_j \sum_t c^d_{jt} d_{ijt} - \sum_i \sum_t c^u_{it} u_{it} \quad (2)$$

subject to

$$r_{ijt} - r_{ijt-1} \le M \cdot s_{ijt} \quad \forall i, \forall j, \forall t \quad (3)$$

$$r_{ijt-1} - r_{ijt} \le M \cdot s_{ijt} \quad \forall i, \forall j, \forall t \quad (4)$$

$$d_{ijt} - d_{ijt-1} \le M \cdot s_{ijt} \quad \forall i, \forall j, \forall t \quad (5)$$

$$d_{ijt-1} - d_{ijt} \le M \cdot s_{ijt} \quad \forall i, \forall j, \forall t \quad (6)$$

$$u_{it} \ge a_i b_{it} - \sum_j v_{ijt} \quad \forall i, \forall t \quad (7)$$

$$l_{ijt} \le l_{ijt-1} + r_{ijt} - v_{ijt} \quad \forall i, \forall t \quad (8)$$

$$l_{ijt} \le d_{ijt} \quad \forall i, \forall j, \forall t \quad (9)$$

$$\sum r_{ijt} \le C^r_{jt} \quad \forall j, \forall t \quad (10)$$

$$\sum_i d_{ijt} \le C^d_{jt} \quad \forall j, \forall t \quad (11)$$

$$\sum_{\tau=t}^{t+\Delta T} s_{ij\tau} \le 1 \quad \forall i, \forall j, \forall t = 1, ..., T-\Delta T \quad (12)$$

$$r_{ijt} \ge 0 \quad \forall i, \forall j, \forall t \quad (13)$$

$$d_{ijt} \ge 0 \quad \forall i, \forall j, \forall t \quad (14)$$

$$s_{ijt} \in \{0, 1\} \quad \forall i, \forall j, \forall t \quad (15)$$

$$l_{ijt} \ge 0 \quad \forall i, \forall j, \forall t \quad (16)$$

$$u_{it} \ge 0 \quad \forall i, \forall t \quad (17)$$

$$v_{ijt} \ge 0 \quad \forall i, \forall j, \forall t \quad (18)$$

$$a_i \in \{0, 1\} \quad \forall i \quad (19)$$

M2 Single Provider Problem - SPP

Variables:

r_t Amount of reserved capacity in period $t = 1,..., T$.

s_t Binary variable, 1 if a allocation setup is made at beginning of period $t = 1,...,T$ and 0 otherwise.

Parameters:

b_t Demanded capacity in period $t = 1,...,T$. Demand is assumed to be greater than 0.

c_t^s Setup costs in period t.

c_t^r Costs per allocated rate in period t.

r_0 Allocation level before the beginning of the first period.

M M is a sufficiently high number (e.g., max $\{b_t\}$).

$$\text{Minimize} \sum_{t=1}^{T} c_t^s s_t + \sum_{t=1}^{T} c_t^r r_t \qquad (1)$$

subject to

$$r_t \geq b_t \qquad \forall t = 1, ..., T \qquad (2)$$

$$r_t - r_{t-1} \leq M \cdot s_t \qquad \forall t = 1, ..., T \qquad (3)$$

$$r_{t-1} - r_t \leq M \cdot s_t \qquad \forall t = 1, ..., T \qquad (4)$$

$$s_t \in \{0, 1\} \qquad \forall t = 1, ..., T \qquad (5)$$

M3 LP Relaxation of M1 (SPP)

The binary condition (5) is dropped from M1 and replaced by

$$0 \leq s_t \leq 1 \qquad \forall t = 1, ..., T \qquad (6)$$

done by fixing the highest not yet fixed s_t to 1 in the first and to 0 in the second subproblem.

Even for this very simple MPRASE problem incarnation an example with only 50 periods took already 33 minutes to be solved[3]. Problems with more than 100 periods could not be solved within several days. The reason for this is that the structure of the problem does not make it very amenable to branch and bound algorithms since s_t are often set to very low values greater 0 resulting in a vast underestimation of fixed costs which leads to very loose bounds. Therefore, we strived for more efficient, yet still exact algorithms for the SPP.

[3] All experiments have been performed on a 400 MHz Pentium II processor using the commercial MIP Solver CPLEX [11].

3.2.2.2 Dynamic Programming (DP).

Let (t_1, t_2) be the edge from vertex t_1 to vertex t_2. The costs (or length) of edge (t_1, t_2) are defined as

$$C(t_1, t_2) = c_{t_1}^s + \sum_{\tau=t_1}^{t_2} c_\tau^r \cdot max(b_t | t \in \{t_1, ..., t_2\}) . \qquad (7)$$

With this, the problem can be solved efficiently with the following algorithm which uses the dynamic programming paradigm [1] and has a complexity of $O(T^2)$ (see Figure 2).

Preparation:

Prepare an empty array *cMin* and an empty array *pred*, each with T entries.

Start:

$cMin(1) = C(t_1, t_1)$

$pred(1) = 1$

Iteration $t = 2, ..., T$:

$cMin(t) = \min\{C(i, t) + cMin(i\text{-}1) \mid i = 1, ..., t\}$

$pred(t) = \text{argmin}\{C(i, t) + cMin(i\text{-}1) \mid i = 1, ..., t\}$

Result:

$cMin(T+1)$ contains the minimal costs while array *pred* stores the hops towards that solution.

Figure 2: Basic Dynamic Programming (DP) Algorithm.

3.2.2.3 Assessment of Execution Times. Table 9 shows the execution times for all of the exact algorithms for two differently sized problem instances.

Algorithm	B&B	DP
$T=50$	1920.7	0.0026
$T=1000$	n.a.	9.0

Table 9: Execution times for exact SPP algorithms (in sec).

3.2.3 Heuristic Solution Algorithms. The last section introduced exact solutions for the SPP, which while they provided fairly good performance still required a certain computational effort that might be prohibitive in scenarios where there is either a large number of periods to be planned for or where there is only an extremely limited amount of time available for computation as, e.g., if the resource allocation is done in response to signalling messages and thus affects setup latencies. Therefore, we now want to investigate heuristic techniques which do not guarantee an optimal solution but allow very fast allocation decisions. A further reason for investigating heuristics becomes obvious when we extend the SPP techniques towards other MPRASE problems later in this paper when we sometimes end up having to solve huge numbers of SPPs.

3.2.3.1 LP Heuristic (LH). The LP heuristic is solving the LP relaxation M3 of Section 3.2.2.1 to determine the amount of allocated capacity. After solving M3 (using the simplex algorithm), any $s_t \neq 0$ is set to 1 wherever necessary (that is, where r_t and r_{t-1} differ). This leads to a relative high number of allocations since fixed costs are systematically underestimated by allowing continuous s_t.

3.2.3.2 Merge Heuristic (MH). The merge heuristic starts with a separate allocation for each period and then tries to merge two successive allocations into one if the saved fixed costs of the allocation are less than the waste of variable costs (see Figure 3 for an illustration of this).

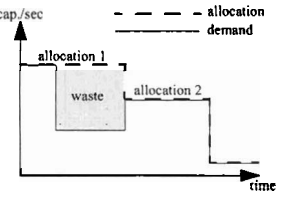

Figure 3: Waste of variable costs.

3.2.3.3 Split Heuristic (SH). The split heuristic starts with a single allocation and then tries for all periods to split existing allocations if the fixed costs for the new allocation are less than the saved waste of variable costs.

3.2.3.4 Combined Heuristics (CH[x,y]). The merge and split heuristics can also be used to further improve the results of other heuristics. In our simulations we therefore iterated through merge and split in sequence until no further improvement could be achieved (CH[MH, SH]). Moreover, we also tried the combination of merge and split based on the result of the LP heuristic (CH[LP,MH,SH]).

3.2.4 Evaluation. In order to evaluate the performance of the heuristics we ran a simulation over 100 random problem instances, each with T=1000, fixed costs $c_t^s \in [200,800]$ drawn from a uniform random distribution once and then set equal for all T periods. Variable costs c_t^r are drawn from [3,5] and remain equal for p periods; p is drawn from [10,20].

The demand b_t is calculated by superposing a number of requests (for example representing individual requests **from several users) with their interar**rival time modelled by a Poisson distribution ($\lambda = 4$) and their duration modelled by an exponential distribution ($\mu = 20$)[4]. For calculating the requests' capacity demand we draw from a uniform random distribution from one out of three possible intervals [2,8], [10,20] and [35,50] representing small, medium and high capacity requests. The interval itself is selected for each request with a probability of 40%, 30% and 30%. Figure 4 shows a sample problem generated in this way.

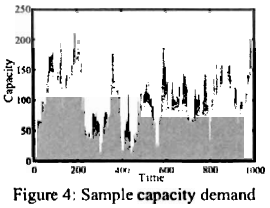

Figure 4: Sample capacity demand

Table 10 shows the results generated by the simulations. Here, allocation length denotes the average duration of a single allocation and waste is the total waste of variable costs for a single SPP instance (as illustrated in Figure 3).

As a very simple alternative heuristic and to have a reference value we also used what we called the peak heuristic (PH) which makes a single allocation with the highest capacity demand over all periods. Expectedly, PH performed very poorly compared to the other techniques. A much better performance at very low execution time is achieved by the merge heuristic (MH): on average it imposes less than 5% additional costs relative to the optimum and reduces execution time by a factor of 4500. The conceptually very similar split heuristic (SH) is considerably less effective. Looking at the allocation length shows the reason: it overdoes its job by splitting too often, resulting in too short allocation lengths and thus incurring fixed costs more often although waste of variable costs is roughly equal to MH.

The LP heuristic performs only marginally better than SH, although it consumes considerably more time. This is due to its characteristic of underestimating fixed costs which is also expressed in a very low waste and small allocation lengths.

Next, let us see how these results may be improved by the combination of heuristics as described in Section 3.2.3.4. The combination of MH and SH leads expectedly to better results than the techniques in isolation. Yet, even better

[4] We have to admit that parameter choice is rather arbitrary (albeit sensible) due to lack of empirical data. However, we have experimented with other values without changing the results in a significant manner.

results can be achieved by integrating LP with MH and SH.

In conclusion, the best results are achieved by CH[LP,MH,SH], yet the most attractive trade-off between cost performance and execution time is probably achieved by MH or CH[MH,SH].

3.2.5 Related Work. The deterministic single provider problem is treated in more detail in [27] and [26]. The algorithms for the single provider problem are extremely useful for many other MPRASE problems and are reused several times in the following chapters.

4. Selected Uncertain MPRASE Problems

4.1 Background on Uncertain Optimization Models

Many decisions and optimizations in the areas of network design, traffic engineering and other resource allocation problems are based on uncertain data due to the relatively long timescales on which these mechanisms operate. In this section we derive several fairly general strategies for dealing with uncertain problems of the MPRASE framework.

4.1.1 Stochastic Programming. We will use methods from stochastic programming in this section. Stochastic programming deals with optimization under uncertainty and was introduced in 1955 by Dantzig [12]. Good overviews on stochastic programming are given in [38, 63, 73, 24]. Many economical problems are solved using stochastic programming; e.g., a case study that uses stochastic programming for capacity planning in the semiconductor industry can be found in [39].

4.1.2 Modeling Uncertainty. If there is no uncertainty with regard to a parameter the value of that parameter is known at the time the decision is made. We then call that parameter *deterministic*. M2 was an example for a model which has only deterministic parameters.

4.1.2.1 Types of Uncertainty. Parameters like future bandwidth demand which form the basis for a decision or optimization process can be and in practice often are uncertain. Several degrees of uncertainty can be distinguished for a parameter:

- *Total uncertainty*: Nothing is known about which values the parameter will take. The best thing one can do in this case is to try to react flexibly and learn from past values the parameter took. Section 4.3 gives an example for an MPRASE problem under total uncertainty and presents an efficient and flexible self-learning algorithm.
- *Stochastic uncertainty*: The exact value the parameter will take is not known but the decision maker knows the probability distribution of the parameter and can thus make some predictions about the parameter. [16] and [54] are typical works that deal with stochastic uncertainty for bandwidth allocation problems from a provider's point of view by assuming sources with on-off traffic.

  Discrete stochastic uncertainty: The parameter is drawn from a discrete set of values, each value has a certain probability. The set is typically modelled as a number of scenarios. This approach is discussed below in more detail as it is the approach taken in Section 4.2.

Algorithm	Costs	Relative deviation from optimum costs				Allocation length	Waste	Time (sec)
	av	*av*	*stddev*	*min*	*max*	*av*	*av*	*av*
Optimum (DP)	*452304*	*n.a.*	*n.a.*	*n.a.*	*n.a.*	*9.43*	*36515*	*9.000*
PH	*1010199*	*123.81%*	*32.96%*	*58.97%*	*221.73%*	*1000.00*	*645804*	*< 0.001*
MH	*474027*	*4.79%*	*1.07%*	*2.05%*	*7.15%*	*10.65*	*64257*	*0.002*
SH	*568759*	*25.93%*	*10.43%*	*12.96%*	*73.65%*	*3.72*	*63295*	*0.010*
LP	*554317*	*22.34%*	*8.37%*	*6.12%*	*39.07%*	*2.62*	*424*	*0.452*
CH[MH, SH]	*469723*	*3.85%*	*0.74%*	*1.80%*	*5.34%*	*9.81*	*56064*	*0.005*
CH[LP, MH, SH]	*460404*	*1.77%*	*0.70%*	*0.39%*	*3.75%*	*8.93*	*41918*	*0.452*

Table 10: SPP simulation results.

4.1.2.2 Modeling Uncertainty with Scenarios. The idea of modeling uncertainty with scenarios has its roots in scenario analysis [49, 47]. Scenario analysis is a method for long-range planning under uncertainty. Conformant and plausible combinations of the realizations of all uncertain parameters yield a number of scenarios. These scenarios form the basis for the following decision process (e.g., a production plan is based on the assumption that one of the three scenarios will occur: "prices and demand go up", "prices fall slightly and demand remains equal", "demand goes back and prices fall heavily"). An application example and literature overview is given in [39].

However, describing uncertainty with a range of scenarios also makes sense for short- and mid-range planning and is often used for stochastic programming [38, 12, 63] as it has some crucial advantages over using a parametrized probability distribution:

- It is easy and intuitive for the decision maker to create the scenarios, they could also be created automatically [22].
  Scenarios are easy to analyze, their plausibility can be approved easier than by creating a mathematical probability distribution.
- Scenarios are flexible, every kind and number of possible events can be easily accounted for in the scenarios.
- Finally, scenarios can be used as a discretization of probability distributions for numerical algorithms.

4.1.3 Robustness. The notion of robust plans stems from decision theory [63]. Decision makers are typically evaluated ex post by how good their proposed plan performed in reality (i.e., in the scenario that actually occurred). As they can loose their job and career when their plan performs badly in the occurring scenario and this typically outweighs the praise if the plan performs well, clever decision makers are risk-averse to a certain degree and biased towards robust plans. A robust plan is a plan that is judged positive in most of the scenarios and does not perform too badly in any of the scenarios.

4.1.4 Strategies for Dealing with Uncertainty.

In the following two sections we examine two MPRASE problems that act under varying degree of uncertainty to demonstrate various strategies that deal with uncertainty.

4.2 VPN Provisioning

4.2.1 Problem Formulation. In this section, we look at a customer that needs a considerable, varying amount of network resources (e.g., bandwidth) over long timescales, for example for a provider provisioned virtual private network (see IETF working group ppvpn, [9, 21]), potentially in support of business-critical applications. The demand fluctuates heavily over the course of a day with peaks in the late morning and afternoon hours and far lower demand in the night hours as well as over the course of the week with ups on the weekdays and downs on the weekend.

Previous research work [23, 43, 74, 25] has shown that it is generally favorable for both customer and provider to allow renegotiation of bandwidth allocations. The customer saves costs during phases of low demand and the provider can make better use of the capacity of the network. Among other findings, the simulations in this section confirm that without renegotiation the costs increase considerably (at least by a factor of 3 in our settings). A lot of research in the area of virtual private networks is done to increase the flexibility of VPNs [14, 41, 36, 35, 48], a trend which will make renegotiations easy and common.

On the downside, for business critical applications renegotiation can be a dangerous mechanism because customers are given no guarantees that they obtain the higher amount of bandwidth they need for their peak demands as the provider could run out of resources in such times leading to a rejection of the request.

This problem can be avoided if renegotiation is combined with reservation in advance. Customers can now request their increased bandwidth ahead of time. They can thus avoid the risk of running out of bandwidth for business critical applications. We show in this section that they will usually still save costs. So there are strong arguments for customers to use reservation in advance.

On the other hand with reservation in advance the provider has a better prognosis of the utilization of the network in advance which may allow him in turn to potentially allocate bandwidth more efficiently at further providers, yet the latter recursion is not in the scope of this paper. We assume that if there is not enough bandwidth for a reservation in advance that either the provider allocates the missing bandwidth at another provider or the customer changes providers.

In the VPN provisioning problem we take the viewpoint of a (e.g., VPN) customer that reserves bandwidth (e.g., for one of the trunks of his VPN) in advance at a provider (e.g., offering a bandwidth-assured VPN service). The problem for the customer is that its demand forecast is necessarily uncertain.

The VPN provisioning problem is the MPRASE problem incarnation $\mathbf{1|1|1|FV|^*|D_D}$ as it deals with an uncertain edge (discrete stochastic demand) between one customer and one provider, uses a one-dimensional resource model and a linear cost model with fixed and variable costs. Our prior discussion of the SPP M2 comes in handy now, as the VPN provisioning problem is quite similar to it. The difference is the uncertain parameter b_t for period $t = 1,...,T$. Using the scenario model from Section 4.1.2 we assume that we have a number S of scenar-

ios with the demand forecast b_{ts} for period t and scenario s, each scenario has a probability p_s with

$$\sum_{s=1}^{S} p_s = 1. \tag{8}$$

4.2.2 Strategies for Dealing with Uncertainty. Because the demand b_t is now uncertain, we can no longer use the algorithms of the SPP. We now derive strategies that can deal with the uncertain parameters b_{ts} and evaluate their robustness later in simulations.

In general, uncertain parameters can occur in the objective function and the constraints of an optimization problem. If the objective function is affected the decision maker runs the risk of not achieving optimal results because of the uncertainty. If, however, the constraints are affected the decision maker risks creating plans that are not valid or realizable in reality. Dealing with uncertainty in the constraints is usually harder and more complex, yet more important than dealing with uncertainty in the objective function [63]. In our problem constraint (2) of M2 is affected by the uncertain parameters b_{ts}. **We now present some general strategies how to deal with problems that have uncertain constraints.**

4.2.2.1 Deterministic Substitution Strategies. For the deterministic substitution strategies we substitute the uncertain (scenario dependent) parameter b_{ts} with a deterministic (scenario independent) parameter $\hat{b}_t$ and then solve the resulting deterministic problem **M2 with the algorithm presented in Section 3.2.2.2.**

Several substitutions can be used. An obvious one is to use the expected value

$$\hat{b}_t = \frac{1}{S}\sum_{s=1}^{S} p_s b_{ts} \tag{9}$$

as substitute, we call this strategy DED (deterministic with expected demand). To avoid underestimating the demand a surcharge α can be added to the substitute. We call this strategy surcharge strategy (DSUα):

$$\hat{b}_t = (1+\alpha)\cdot\frac{1}{S}\sum_{s=1}^{S} p_s b_{ts} \tag{10}$$

For the deterministic worst-case strategy DWC we use the highest value of all scenarios as substitute:

$$b_t = max\{b_{ts} | \forall s\} \tag{11}$$

A plan based on the worst case values yields a solution that satisfies all constraints for all scenarios, this is why such a strategy is also called fat solution strategy [38, 63].

4.2.2.2 Chance Constrained Strategies. The deterministic strategies have no real control over the chance that their plan violates the uncertain constraints with the exception of DWC which makes sure that the plan is valid for 100% of the scenarios. The chance constrained strategy CC allows finer control over the chance that a plan is valid by introducing a factor α and forcing the uncertain constraint to be satisfied in at least α percent of the scenarios.

The chance constrained strategy is much harder to implement than the deterministic substitution strategies. The MIP model and an efficient algorithm to implement the chance constrained strategy CC are presented in [28], there also a simplified version of the CC strategy is presented, the so called separated chance constrained strategy SCC.

4.2.2.3 Recourse Strategies. The CC strategy controls the risk that a solution is invalid to some extent. Recourse strategies control the risk in a different way. In M4 a recourse strategy with expected recourse (RER) is given.

M4 Bandwidth Allocation with Expected Recourses (RER)

Variables see M2 and

f_{ts} Recourse for scenario s = 1,...,S for period t = 1,...,T.

Parameters see M2 and

c_t^f Recourse costs for scenario s = 1,...,S for period t = 1,...,T.

b_{st} Demanded capacity in scenario s = 1,...,S for period t = 1,...,T.

p_s Probability of scenario s = 1,...,S

$$\text{Minimize } \sum_t c_t^s s_t + \sum_t c_t^r r_t + \sum_t \sum_s p_s c_t^f f_{ts} \tag{12}$$

subject to (3), (4), (5) and

$$r_t + f_{ts} \geq b_{ts} \qquad \forall t, \forall s \tag{13}$$

$$f_{ts} \geq 0 \qquad \forall t, \forall s \tag{14}$$

In constraint (13) the new variable f_{ts} measures by which amount the demand remains unsatisfied in scenario s for the resulting planned allocation in period t, r_t. The CC strategy only takes into account that demand is unsatisfied or not, the recourse strategy also takes into account how much demand is unsatisfied in a given scenario.

The recourse f_{ts} has to be penalized in the objective function. The RER does this by weighting f_{ts} with c_t^f and adding the expected value over all scenarios to the objective function(12).

In order to implement the recourse strategy the algorithm of Section 3.2.2.2 can be reapplied with some modifications. It uses as new cost function

$$C(r_{12}, t_1, t_2) = c_{t_1}^s + \sum_{t=t_1}^{t_2} c_t^r r_{12} + \sum_{t=t_1}^{t_2}\sum_{s=1}^{S} p_s c_t^f f_{ts}(r_{12}, t_1, t_2), \tag{15}$$

the optimal rate r_{opt} (that leads to minimal costs

$C_{opt}(t_1, t_2) = C(r_{opt}(t_1, t_2), t_1, t_2)$ between t_1 and t_2)

$$r_{opt}(t_1, t_2) = r | C(r, t_1, t_2) = \min\{C(r, t_1, t_2) | \forall r \in [0, max\{b_t | t \in [t_1, t_2]\}]\} \tag{16}$$

and the recourse $f_{ts}(r, t_1, t_2)$ which is defined as

$$f_{ts}(r, t_1, t_2) = max\{0, b_{ts} - r\} \tag{17}$$

As c_{t1} is fixed, the minimum costs $C(r_{12}, t_1, t_2)$ from (15) can also be written as

$$\tilde{C}(r, t_1, t_2) = \sum_{t=t_1}^{t_2} c_t^r r + \sum_{t=t_1}^{t_2}\sum_{s=1}^{S} p_s c_t^f max\{0, b_{ts} - r\} \tag{18}$$

which can be rewritten as

$$\tilde{C}(r, t_1, t_2) = \left(\sum_{t=t_1}^{t_2} c_t^r\right) r - \sum_{t=t_1}^{t_2}\sum_{s=1}^{S} p_s c_t^f min\{0, r - b_{ts}\} \tag{19}$$

$$\tilde{C}(r, t_1, t_2) = \tilde{C}_1 - \tilde{C}_2 \tag{20}$$

$$\text{Function } \tilde{C}_1 = \left(\sum_{t=t_1}^{t_2} c_t^r\right) r \tag{21}$$

is a linear strictly monotonic increasing function of r.

$$\text{Function } \tilde{C}_2 = \sum_{t=t_1}^{t_2}\sum_{s=1}^{S} p_s c_t^f min\{0, r - b_{ts}\} \tag{22}$$

is a wide-sense increasing piecewise linear function that starts with negative values. Its slope is decreasing and becomes zero for all $r > max\{b_{ts} | s=1,...,S, t \in [t_1, t_2]\}$. **For a local minimum the slope of the difference of these two functions $\tilde{C}$ has to be zero[5].** As the slope of $\tilde{C}$ is the difference between the constant positive slope of $\tilde{C}_1$ and the decreasing slope of $\tilde{C}_2$ it is zero only for a single point t_a or a single interval $[t_a, t_b]$. $\tilde{C}$ therefore only has one local minimum which is then at the same time the global minimum. If there is only a single minimum it can be easily found with a binary search over all $r = b_{ts}$ with $t \in [t_1, t_2]$ and $s=1,...,S$. This results in a worst-case complexity of $O(T^2 log(TS))$.

4.2.3 Evaluation. A simulative comparison is used in [28] to assess the merits of the different strategies. Realistic demand patterns for 20 scenarios with peaks in the late morning and afternoon and downs during the night in accordance with [57] and [37] are used to describe empirically found traffic patterns. The exact method is described in [28]. The possible demand of one scenario is depicted in Figure 5 together with the allocation made by the RER strategy. As can be seen, it is possible that a plan does not allocate sufficient bandwidth for the demand of some periods for a given scenario. To account for such failures of the bandwidth allocation strategies the unsatisfied demand is penalized with penalty costs that are 10 times as high as the variable costs.

In [28] the robustness and general performance of the strategies are evaluated in detail, we summarize the results here, the minimal, average and maximum rel-

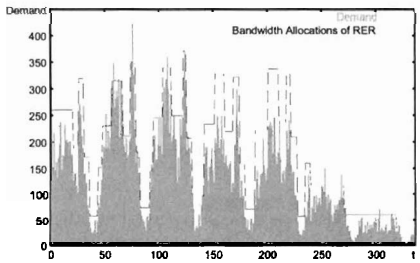

Figure 5: Real demand for one week and the allocations made by the RER strategy

[5] The slope in a local minimum or maximum is zero. The difference function here obviously has no maximum.

ative deviation from the optimal costs are depicted in Figure 6:

The robustness was evaluated based on the worst case performance of the strategies. The RER strategies show the best worst-case behavior, followed by SCC and $DSU_{0.2}$. The RER and SCC strategies are more robust concerning the variation of their parameters α respectively c than DSU.

DED and DSU with lower or higher surplus perform very badly, as does DWC and CC. Those strategies cannot be considered robust, this is important for the DWC strategy which is based on the worst case demands and thus never leads to penalty costs. But its basic plan is still much more expensive than the combination of penalty and the planned costs of the other strategies. Only when the penalty costs are set higher than 100 times the variable costs the DWC strategy performs acceptably. Thus the DWC strategy cannot be recommended for a wide range of parameter sets of the bandwidth allocation problem.

DED and DSU with low surplus factor are also not robust. Only if the surplus factor of DSU is set correctly its performance is acceptable; it can thus not really be considered robust.

SCC and RER can be considered robust. SCC bases its calculations on quantiles of the demand distribution and thus uses more information from the demand distribution than the surplus strategies DSU which explains the better performance. RER performs very good, obviously the fine-grained control over the risk makes it more robust than the deterministic strategies.

Next, the general performance is evaluated based on the average performance over a number of simulation runs with higher uncertainty. The ranking in performance is quite similar to the ranking regarding robustness above. The RER and SCC strategies perform best and can be recommended.

DSU again only performs well if the surplus factor is set correctly. DED and DWC as well as CC perform relatively badly and cannot be recommended.

The conclusions from the experiments are that the RER strategy should be used. The recourse costs should be set similar to the estimated (calculatory) penalty costs of unsatisfied demand for best performance. However, the strategy is robust against a wrong setting of the recourse costs, it still performs very good as long as the recourse costs are in the same order of magnitude as the estimated penalty costs.

If the computational complexity of RER is too high, SCC can alternatively be used, it performs a little worse but still better than all other strategies and is easy to compute.

The experiments allocating resources once per week without renegotiation lead to about 3 times higher costs than those yielded by RER or SCC. This shows again that renegotiation can save a considerable amount of costs. We have

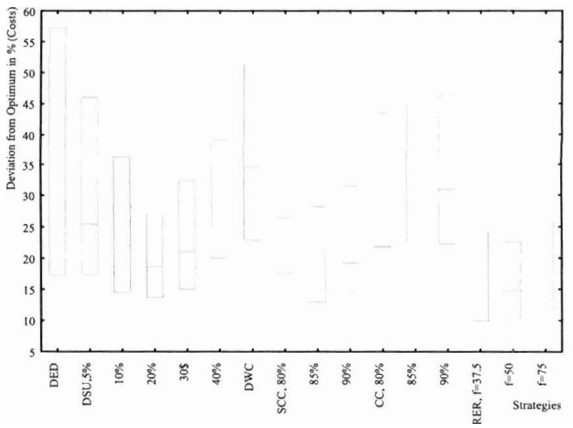

Figure 6: Relative Deviation from the minimal costs

explained why reservation in advance is vital to avoid the risk of not getting enough bandwidth in peak periods. Even if that is not the case reservation in advance can be better than short-term reservations: short term reservations will typically be priced higher because they leave the provider with a much higher planning uncertainty and the risk of underutilizing his resources. The results show that if short-term allocations are priced only 15 to 20% higher than long-term reservations the latter combined with a robust algorithm are cheaper than the optimal short-term allocations.

4.2.4 Related Work. In [72], service provisioning for distributed communication networks with uncertain data is studied. Several service provisioning models are presented that account for several types of uncertainty. However, no efficient solution algorithms are presented and no simulations are carried out. Another related work is [15], here a service provider offers computational services and tries to maximize profits. In our work we consider a network service and take the perspective of the customer. Some of the methods presented in this paper were also successfully applied to a different problem domain, the planning of a production program [29].

4.3 Decoupling of Time-Scales

While the last problem was an example for a problem with (discrete) stochastic uncertainty we now discuss one with total uncertainty. We present a framework to solve such problems that is useful beyond MPRASE.

4.3.1 Problem Formulation. Different time scales of QoS systems may arise due to different QoS architectures like RSVP/IntServ (Resource reSerVation Protocol/ Integrated Services) [6], DiffServ (Differentiated Services) [3], or ATM (Asynchronous transfer Mode) [4] being used but may also be due to different QoS strategies followed by providers even if they employ the same QoS architecture. Choosing different QoS architectures as well as different strategies results from serving different needs, e.g., for an access and backbone provider. An access provider that has a comparatively moderate load and directly connects to end-systems may favor a fast time scale system responding immediately to the end-systems requests. A backbone provider that connects access providers respectively offers transit services is generally faced with a drastically higher load of individual transmissions, so that reaction on the time scale of individual requests is usually not possible and a slower time scale system needs to be enforced.

When different time scales are in operation in heterogeneous network QoS systems, it is simply not possible to query the underlying QoS system each time an overlaid system is altering its state. Here, the system operating on a faster time scale needs to be smoothed when overlaying it onto a system that operates only on slow time scales.

A realistic configuration for access and backbone providers may be, e.g., that access providers use RSVP/IntServ to suit their customers' needs while a backbone provider uses DiffServ with a Bandwidth Broker (DiffServ/BB) to allow for some dynamics but on a slower time scale. This scenario is shown in Figure 7.

Here it is also very obvious why a BB is generally not able to react to individual RSVP requests that are arriving at edge devices between access and backbone provider. Because if it did, the BB would need to operate at a throughput of requests that is proportional to the square of the number of access providers it serves - that is not scalable. Here a decoupling of the different time scales is necessary. The decoupling can be achieved by building "depots" of capacity which stabilize the fluctuations of the "nervous" demand curve for backbone capacity by individual requests. From another perspective, the decoupling technique can also be viewed as introducing a combined local and global admission control for

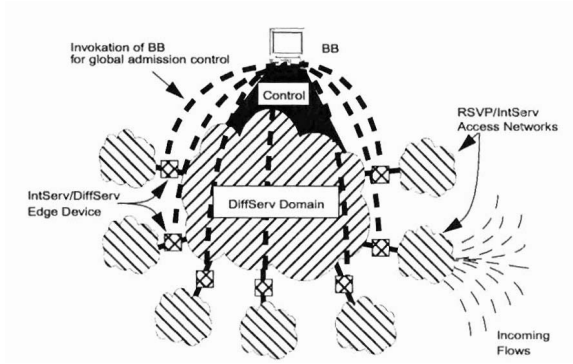

Figure 7: Combined local and global admission control.

the DiffServ/BB network. Global admission control is only invoked whenever local admission control at an edge device runs out of resources in its capacity depot. In such a case, local admission control on an edge device tries to obtain more resources from the global admission control represented by the BB. This scheme allows to trade off resource efficiency for a more stable and long-term capacity demand presented to the BB.

This problem of decoupling different timescales is the MPRASE problem incarnation $\mathbf{1|1|1|FV|*|D_D}$, it is similar to the SPP M2 but acts under total uncertainty. In the next section we present a flexible, self-learning, and powerful heuristic scheme.

4.3.2 Solution Strategies. Acting under total uncertainty we propose the use of an adaptive heuristic as a way to learn the statistical properties of the system in an on-line fashion. The scheme we propose is highly useful in an environment where there are unpredictable, but rather long-term fluctuations in the demand for capacity. In general, the adaptation to behavior that would have been "good" in the past is the best a heuristic technique can do under complete uncertainty.

The question what is "good" behavior can be assessed by comparing the outcome of an on-line heuristic with the optimal solution of the SPP that results by looking back at the past n periods.

We now first present the adaptive algorithm framework and next a heuristic

$h(\theta)$ that fits into this framework. The combination of both is then evaluated by means of simulation.

4.3.2.1 The Adaptive Framework. Let us assume that we have a parametrized heuristic $h(\theta)$ for the on-line decoupling problem and that we use an exact algorithm for the off-line decoupling problem, which is the SPP with the now deterministically known demands of the last n periods. There are essentially two different modes of adaptation that can be directed by good behavior as achieved by the cost-minimal cover of the past CDC:

Adaptation in Action Space. In this mode, the heuristic's parameter (vector) θ is adapted such that the behavior of the CDC cover produced by applying the heuristic deviates as little as possible from the optimal cover with respect to some characteristic as, e.g., the number of reallocations.

Adaptation in Performance Space. In this mode the heuristics parameter (vector) θ is adapted such that the cost of the solution produced by applying the heuristic $h(\theta)$ deviates as little as possible from the optimal cost obtained for the SPP.

Both adaptation modes have three parameters:

- The *frequency of adaptation* determines how often the adaptation of the heuristics parameter is carried out.
- The *time window of adaptation* determines the length of the past period that is taken into account for the adaptation.
- The *accuracy of adaptation* determines how thoroughly the parameter space is searched during the optimization problem for the adaptation.

We call this adaptation scheme ODAH (Optimum-Directed Adaptive Heuristic).

4.3.2.2 An On-line Heuristic. Now, a very simple, yet reasonable heuristic is introduced that deals with the problem under total uncertainty at each period. It is called thresholded depot excess (TDE) as it ensures that the capacity depot held for decoupling is never above a certain threshold. It is applied in each period:

If the demand level rises above the current allocated capacity the change is always followed (assuming that there is enough capacity at the underlying QoS system). Whenever demand decreases, TDE checks whether the step is smaller than a certain fraction $\alpha \in [0,1]$ of the old allocation level and if that is the case, TDE follows this step.

Of course, the value of parameter α is crucial for the success of TDE. If α is set too high, then TDE is too "nervous", and will produce too many changes in the level of the depot and if it is set too low, TDE is too "lazy", and will waste a lot of capacity.

4.3.2.3 Combining Both. We now integrate TDE into the ODAH framework so that the parameter α is adapted automatically. We call the resulting heuristic ODAH-TDE.

As discussed above, there are two modes of adaptation in the ODAH scheme: adaptation in performance space and in action space. In principle, both kinds of adaptation are possible for ODAH-TDE. The adaptation in performance space works by simply adjusting TDE's parameter α such that the difference in costs between TDE(α) and the optimal solution of the SPP (see Section 3.2) is minimized. This minimization is done by a simple recursive grid search [19] through the interval [0,1] for parameter α as there is no simple relationship between α and c for a more intelligent search to exploit. See [61] for details.

For the adaptation in action space, it was decided to use the number of reallocations as basis for the similarity relation between covers, so that in this case the difference in the number of reallocations is to be minimized. We can use an interpolation search [19] since α and n have a simple relationship: $n^{TDE,\alpha}$ is monotonically increasing in α. This is, of course, much more efficient than the recursive grid search for the adaptation in performance space mode. See [61] for details.

4.3.3 Evaluation. In simulations (described in detail in [61]) we experimented with both adaptation modes. Both modes performed very similar but adaptation in action space is more efficient due to the less compute-intensive adaptation step.

The simulation results show that ODAH-TDE generally achieves a good and robust performance over all types of requests. In particular we experimented with different lifetimes of requests, where ODAH-TDE was able to achieve over 90% of the cost saving performance of a hypothetical optimal scheme which operates under certainty, i.e., solves the SPP exactly.

ODAH thus represents a robust scheme for heuristically dealing with the sequential decoupling problem under total uncertainty. In particular, it should work well even if flow characteristics as the lifetime of requests change since it shows good performance for all types of requests in the simulations.

4.3.4 Related Work. [70] deals with a two-tier model which consists of an intra- and interdomain resource management. BBs are representing each administrative domain in the interdomain resource management. Based on measurements, a watermark heuristic at edge devices is used to trigger inter-domain signalling. In contrast to our work, the triggers are based on traffic measurements instead of control path events. Furthermore, the introduction of the watermark technique is rather ad hoc, and resembles the TDE algorithm without any adaptation.

One piece of work that explicitly deals with different time scales of access and backbone networks on the control paths is [51]. Here a backbone QoS signalling is proposed which integrates mechanisms in order to dampen the faster time scales of access networks. This mechanism is based on hysteresis and quantization for traffic aggregates which are based on sink trees towards destinations. The applied algorithm is to always reserve capacity in multiples of a certain quantity Q. Whenever the reserved capacity level of $k \times Q$ is no more sufficient, it is increased to $(k+1) \times Q$ and the new quantum is only relinquished when the reserved capacity falls below $(k-1) \times Q$. This is very comparable to the simple strategy of the TDE algorithm, and uses no adaptation.

4.4 Admission Control Problems

Admission control is a widely recognized problem at system edges. The basic admission control problem is $N|1_{Cap}|1|P|*|D$ or with an n-dimensional resource model (e.g., token buckets) $N|1_{Cap}|N|P|*|D$ and consists of maximizing the profit (from the accepted customers) from a provider's point of view or the total utility from a user's point of view. Admission control is discussed broadly in literature, e.g., [7, 16, 42, 43, 45, 55, 60].

5. Selected Deterministic MPRASE Problems

In this section we discuss several selected deterministic MPRASE problems.

5.1 Provider Selection

The basic provider selection problem $1|N|1|FV|*|*$ and $1|N_{Cap}|1|FV|*|*$ could be regarded as the dual problem of the basic admission control problem (Section 4.4). Unlike the latter it is not treated broadly in literature. Because of this, we treat it here in more detail than the admission control problems.

5.1.1 Problem Formulation. Let us assume that there are a number of providers offering capacity to a single customer. The customer has to decide which or which combination of providers to select and if and when to change the providers.

We assign index $j = 1, ..., J$ to the different providers. We can model this problem with M5. This model mainly differs from the SPP M2 in the additional index j. Furthermore, we now have to model the case that in a certain period no capacity is allocated at a certain provider. This is captured by the introduction of demand defect variables, d_{jt}, and the constraints (27) and (28). Here, ε needs to be chosen small, e.g., $\varepsilon = 1/(\max\{b_t\})$, whereas M and L need to be chosen large, e.g., $M = \max\{b_t\}$ and $L = 1/(\min\{b_t | b_t > 0\})$.

M5 Provider Selection Problem - PSP

Variables:

- r_{jt} Amount of allocated capacity in interval t from provider j.
- s_{jt} 1 if an allocation for provider j is made at the beginning of period t and 0 otherwise.
- d_{jt} 1 if allocation for provider j drops to 0 in interval t and 0 otherwise.

Parameters:

- b_t Demanded capacity in interval $t = 1, ..., T$. Demand must be fully satisfied in each period.
- c^s_{jt} Setup costs, i.e., cost for an allocation in period t from provider j, we assume $f_{jt} > 0$.
- c^r_{jt} Variable capacity costs, i.e., costs per capacity unit per period (specific per provider and period).
- r_{j0} Allocation level before the beginning of the first planning period.

$$\text{Minimize} \sum_{j=1}^{J}\sum_{t=1}^{T} c^s_{jt}(s_{jt} - d_{jt}) + \sum_{j=1}^{J}\sum_{t=1}^{T} c^r_{jt} r_{jt} \qquad (23)$$

subject to

$$\sum_{j=1}^{J} r_{jt} \geq b_t \qquad \forall t = 1, ..., T \qquad (24)$$

$$r_{jt} - r_{j(t-1)} \leq M \cdot s_{jt} \quad \forall j = 1, ..., J, \forall t = 1, ..., T \qquad (25)$$

$$r_{j(t-1)} - r_{jt} \leq M \cdot s_{jt} \quad \forall j = 1, ..., J, \forall t = 1, ..., T \qquad (26)$$

$$d_{jt} + \varepsilon r_{jt} \leq 1 \qquad \forall j = 1, ..., J, \forall t = 1, ..., T \qquad (27)$$

$$L(r_{jt} + r_{j(t-1)}) \geq d_{jt} \quad \forall j = 1, ..., J, \forall t = 1, ..., T \qquad (28)$$

$$d_{jt} \in \{0, 1\} \qquad \forall j = 1, ..., J, \forall t = 1, ..., T \qquad (29)$$

$$s_{jt} \in \{0, 1\} \qquad \forall j = 1, ..., J, \forall t = 1, ..., T \qquad (30)$$

$$r_{jt} \geq 0 \qquad \forall j = 1, ..., J, \forall t = 1, ..., T \qquad (31)$$

In the next step we use additional parameters, k_{jt}, to model by (32) that each provider j can offer only a limited amount of resources k_{jt} in period t. This leads to model M6, the capacitated PSP (cPSP).

Capacitated Provider Selection Problem - cPSP

Minimize (23)

subject to (24)-(31) and

$$x_{jt} \le k_{jt} \qquad \forall j = 1, ..., J, \forall t = 1, ..., T \qquad (32)$$

5.1.2 Solution Algorithms. The uncapacitated PSP represents a situation where a customer's demand is relatively small compared to the provider's supply such that the resulting problem consists mainly in the selection of the cheapest provider. The capacitated PSP (cPSP), on the other hand, rather deals with a good mixing of providers to achieve low total costs.

Note that the problem complexity of PSP is much higher than that of SPP (see Section 3.2). First, the demand of each period can be satisfied by 2^{J-1} different combinations of providers and second, if two or more providers are selected to satisfy the demand of one period there is a high number of sensible shares between these. This higher complexity is also illustrated by the execution times of applying the standard branch and bound solver to model M5. A small PSP with T=20 and J=4 already took 1920.8 seconds to solve while the corresponding SPP with T=20 only took 1.2 seconds. For any larger PSPs execution **times** were no longer reasonable. With this complexity in mind we go directly for heuristics and try to exploit our knowledge about the SPP.

5.1.2.1 Static Cheapest Provider Heuristic (SCPH). A rather straightforward approach to tackle the uncapacitated PSP is to transform it into J SPPs, one for each provider and each with the full demand. The SPPs can then be solved by any of the SPP algorithms discussed in Section 3.2. After solving the J SPPs we select the provider of the SPP with the least costs. That means we obtain a solution where one provider is used for all periods.

5.1.2.2 Dynamic Cheapest Provider Heuristic (DCPH). One drawback of SCPH is that it does not allow provider changes. Using a technique similar to the DP algorithm from Section 3.2.2.2 we can eliminate this characteristic of the SCPH. The resulting algorithm is called dynamic cheapest provider heuristic (DCPH). This is also illustrated in Figure 8.

We use the DP algorithm from Section 3.2.2.2, but the minimal costs $C(t_1, t_2)$ for satisfying the demand between two periods t_1 and t_2 are obtained by solving J independent SPPs for the interval $[t_1, t_2]$ and choosing the cheapest provider. Unlike the DP algorithm from Section 3.2.2.2, this algorithm does not necessarily lead to the optimal result as it does not allow for a constellation as depicted for the optimal solution in Figure 8. Again, we have the freedom of selecting any of the SPP algorithms for solving the sub-SPPs.

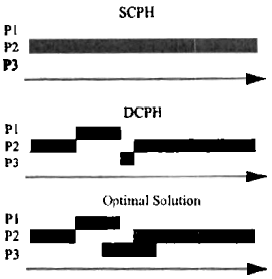

Figure 8: Provider usage of the different algorithms for the PSP.

5.1.2.3 Adaptation of the Heuristics for the Capacitated PSP. If the capacity of one provider is not enough to satisfy the whole demand we can no longer simply select a single provider in SCPH and DCPH but have to combine several providers. We do this by first cropping the demands in each SPP to the capacity of the according provider. We then solve the SPPs for all J providers and select the provider that has the minimum costs per satisfied demand. The overall demand is then reduced by the capacity served by the selected provider and the procedure is repeated until no more demand remains unsatisfied. Example allocations are shown in Figure 9.

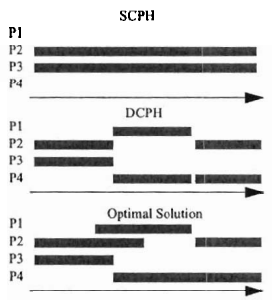

Figure 9: Provider usage of the different algorithms for the cPSP.

Please note that the non-zero demand assumption in Section 3.2 can now no longer be held and model M2 as well as the heuristics of Section 3.2 had to be adapted to cope with periods of no demand.

5.1.2.4 Other Heuristics for the PSP. Of course, we can again use the results of the LP relaxation for M5 and M6 to obtain a solution for PSP/cPSP.

We also adapted the merge heuristic to the multi-provider case and to the capacity constraints and combined it with DCPH and LP in order to investigate whether it can improve their solutions.

5.1.3 Evaluation. In order to evaluate the PSP heuristics described above we ran a simulation over 50 PSP instances similar to the simulations in Section 3.2 with 100 periods. We used 10 providers and different levels of capacity. The average different costs the used strategies yielded are depicted in Figure 10 for an uncapacitated and a capacitated PSP. In the latter problem, 2.58 providers were used on average at the same time. We use the DP algorithm from Section 3.2.2.2 for the SPP subproblems.

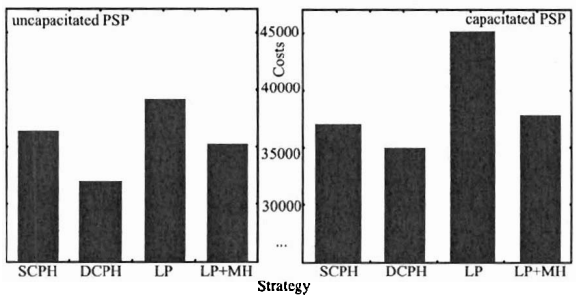

Figure 10: Some Results for the PSP

The results for the uncapacitated PSP show that DCPH is expectedly significantly better than SCPH. This, however, comes at a drastically increased execution time (243 s contra 0.4 s per instance). While the LP heuristic alone does not perform good, it performs well if combined with the merge heuristic (roughly 2.3 s execution time). Please note that running merge on the solution of DCPH was ineffective because within its range (i.e., only one provider at a time) the DCPH solution is already optimal.

In the capacitated case the results are similar but SCPH comes closer to the results yielded by DCPH. This can be explained by the fact that now because of the limited capacities also the modified SCPH can and has to use more than one provider.

Summarizing, DCPH leads to good results if the execution time does not matter, otherwise SCPH and the combination of LP and MH can be recommended. The results from the SPP came in handy and the good heuristics for the SPP could be adapted and perform well here again.

5.1.4 Related Work. The provider selection problem is discussed in more detail in [27]. As we mentioned, there are not many works about the provider selection problem. [65] analyses dynamic provisioning in a multi-provider environment and gives very interesting insights into the global behavior of such a system by game-theoretic observations.

5.2 Token Bucket Dimensioning

5.2.1 Problem Formulation. For any kind of QoS guarantees traffic has to be regulated. Traffic shapers and policers are common elements in both IntServ [6] and DiffServ [3]. Token buckets are the most popular traffic regulating mechanism, especially as they are easy to implement, see, e.g., [34, 30, 31, 67, 59] for the role of token buckets in a DiffServ environment. A token bucket is specified by two parameters, the rate r and the bucket depth B. The sender accumulates tokens in the bucket with a rate of r. Unused tokens are stored in the bucket, there can never be more than B tokens in the bucket, surplus tokens are lost. In order to send data tokens are spent (e.g. per byte or per packet). The bucket starts with $\delta \cdot B$ tokens ($0 \le \delta \le 1$). We assume that this parameter δ is fixed.

Consider the following problem: A single token bucket (r, B) has to be dimensioned for a flow x_t ($t = 1, ..., T$) which is known in advance as when streaming a pre-recorded video from a server towards a client. We assume that the allocation of r and B imposes certain (real or fictive) costs C_r and C_B, the relation between those two coefficients expresses the trade-off between rate r and buffer B. Our aim is to find the optimal token bucket (r_{opt}, B_{opt}).

We call the problem the single token bucket dimensioning problem (STBD), it is the problem incarnation $1|1|N_{TB}|F_{\infty}V|*|*$ of the MPRASE framework. To some extent this problem has already been discussed in literature:

According to [68] the first work to efficiently calculate the minimal bucket depth of a token bucket for a given token rate - and that is a subproblem of the STBD - was done by Partridge and Garrett in 1994 [52]; their algorithm Send-Now is also described in [68]. An algorithm for the same problem which is more flexible as it does not rely on a full bucket in the first period is also derived in [68]. Both papers also deal with calculating the minimal bucket depth for a given rate when a certain queue is added before the token bucket in which the stream

can be hold while it is waiting for enough tokens to be accumulated.

However, these works look at the optimal B for a given r but do not calculate the optimal r.

Keshav [40] proposes as a heuristic for token bucket dimensioning to choose the "knee area" that the $B_{opt}(r)$ curve shows, outside which small changes in r resp. B can only be compensated by greater changes in the other parameter. However, Keshav does not propose a trade-off function with which the preference of r and B can be weighted and influenced and he proposes no algorithm to find the area. Also other works [54] show that the "knee area" is not straightforward to find for long range dependent traffic.

5.2.2 Exact Algorithm. The optimal token bucket (r_{opt}, B_{opt}) for a given stream can be calculated as follows:

The optimal B for a given rate r and $\delta \neq 0$ is:

$$B_{opt} = max(B_{opt1}, B_{opt2}) \tag{33}$$

$$\delta B_{opt1} = \max_{1 \le v \le T} \left(\sum_{i=1}^{v} x_i - rv \right) \tag{34}$$

$$B_{opt2} = \max_{2 \le u \le v \le T} \left(\sum_{i=u}^{v} x_i - r(v-u+1) \right) \tag{35}$$

The proof is given in [25].

Next, the optimal r has to be found. We use a cost function to describe the trade-off between rate and bucket depth. With a linear cost function the bucket costs are a function

$$P_{opt}(r) = c_r \cdot r + c_B \cdot B_{opt}(r) . \tag{36}$$

We can find the minimal costs of this piecewise linear function using a search algorithm similar to regula falsi which is described in [25].

5.2.3 Related Work. On the first view the static token bucket dimensioning problem resembles lot sizing, lot scheduling and related problems [34]. Unfortunately, the nature of the resources involved is fundamentally different and the mathematical structure is different enough that the algorithms and methods do not fit.

Apart from the works mentioned above in 5.2.1 there are some works in the area of Quality of Service (QoS) dealing with **I|1|NTB|F_∞ V|*|D_S** and **I|1|NTB|F_∞ V|*|***. Glasmann et.al. present in [20] a simple heuristic for guessing the token bucket parameters for video conferencing flows. The heuristic consists of setting r to the mean transmission rate of the video and B to the number of tokens that are then required to avoid packet drop. This work does not consider the potential trade-off between r and B but shows some realistic values for video streams.

Dovrolis et.al. [13] analytically derive from the empirical envelope the optimal token bucket parameters. It considers the trade-off between r and B and tries to minimize the reserved rate R of an IntServ guaranteed service flow given a delay bound. This problem can be seen as a subproblem of the STBD problem in this paper with a fixed trade-off which minimizes R.

Falkner et. al. [16] use a cost function for token bucket dimensioning with minimum costs from the perspective of a single user. They, however, assume an ATM network and on-off traffic which is not known in advance. They solve the resulting non-linear optimization problem with the Lagrangean method.

Bruno et. al. [8] study token bucket dimensioning for aggregate VoIP sources for the DiffServ Expedited Forwarding service class. Their LBAP is an aggregation of independent fluid on-off sources. They analyze the effect of token bucket parameters on the non-conformance probability. They, however, do not use a cost function or something similar and do not present an algorithm to derive the optimal pair of token bucket parameters.

Kulkarni and Gautam study in [44] the sizing of K token buckets with admission control resp. network utilization in mind. They also formulate and solve token bucket dimensioning as an explicit optimization problem but their perspective is fundamentally different to ours. While we consider minimizing the costs of one customer and expect the customer to choose his/her token bucket parameters they do not look at costs but try minimizing the sum of the rates of K customer's token buckets at the same time, taking the network's point of view.

Procissi et. al. analyse in [54] the influence of long range dependence in traffic on the dimensioning of token buckets. They use two cost models, one of them similar to the one used in this work, to derive an analytical model for estimating the token bucket parameters. This model explicitly takes into account the long range dependency of traffic, the $B_{opt}(r)$ curve is obtained for traffic modeled as a Fractional Brownian Motion process. As a result they can quite well estimate good token bucket parameters for Internet traffic. They, however, show no algorithm for calculating the optimal parameters for a given trace as we did.

Naudts [50] describes an efficient algorithm for calculating the optimal cell rate $r^*(\tau)$ for a given τ for the ATM generic cell rate algorithm (GCRA). As the GCRA can also be described as a continuous-state leaky bucket this is equivalent to calculating the bucket rate for a given bucket depth.

In [59] a token bucket marker is used for TCP streams and the effect of the token bucket parameters on the achieved sending rate are analysed. That paper operates with different assumptions (TCP instead of real-time traffic) and is thus complementary to the algorithms in Section 5.2.2.

5.3 Renegotiable Services

5.3.1 Problem Formulation. In Section 5.2 we have shown how to calculate the optimal token bucket (r_{opt}, B_{opt}) for a given flow of length T. Video streams often have longer scenes with a relatively high or low transmission rate. Fitting a single token bucket usually leads to a high resource waste during the times with a rather low transmission rate. For example, the cost minimal single token bucket for the Asterix movie of [58] with c_r=1, c_B=0.1 leads to a solution where the bucket is only used in 89 of 40000 periods[6]!

For a longer video stream it thus makes sense to allocate a series of token buckets instead of a single token bucket. But we have to assume that there is a certain reservation overhead involved for the setup of each new token bucket and we want to avoid that a token bucket is used for a too short time period. We account for this again by introducing setup costs which are applied whenever a new token bucket is used. Another possibility would have been to allow a new token bucket only every n periods. The latter, however, is less flexible and can usually be achieved by choosing setup costs adequately, as our results show.

Please note again that we do not necessarily mean real costs, they can also be fictive / calculatory:

- **For each allocation, independent of its duration, fixed setup costs c^s are** incurred.
  The token rate r induces costs proportional to height and duration:
  $p_r(r, \tau_i) = c^r \cdot r \cdot \tau_i$.
- The costs per bucket depth B are similar: $p_B(B, \tau_i) = c^B \cdot B \cdot \tau_i$.
- The $\delta \cdot B$ tokens in the bucket at the beginning of an allocation induce the following costs: $p_\delta(\delta \cdot B) = c^\delta \cdot \delta \cdot B$.

The DTBD can be formulated as a quadratic optimization problem (see M7) and is thus generally extremely hard to solve exactly with standard optimization techniques [32].

Target function (37) of M7 minimizes all costs consisting of the setup costs, the costs for the rate, the bucket depth and the tokens the bucket is filled with

M7 Dynamic Token Bucket Dimensioning (DTBD)

Variables:

- r_t rate in period $t = 1,...,T$.
- B_t bucket depth in period $t = 1,...,T$.
- y_t number of tokens in the bucket at the beginning of the period $t = 1,...,T$.
- s_t binary variable, set to 1 if the token bucket parameters (r_t, B_t) were changed at the beginning of the period $t = 1,...,T$ and 0 otherwise. This variable is necessary to account for the setup costs F.

Parameters:

- x_t tokens used in period $t = 1,...,T$ to send data.
- c^r cost coefficient for the rate r_t.
- c^B cost coefficient for the bucket depth B_t.
- c^δ cost coefficient for each token in the bucket at the beginning of a new allocation period.
- c^s fixed setup costs per redimensioning.
- δ bucket starting factor ($\delta \in [0, 1]$).
- M big enough constant to resemble infinity numerically,e.g. $M = \sum_{t=1}^{T} x_t$

$$\text{Minimize } c^s \sum_{t=1}^{T} s_t + c^r \sum_{t=1}^{T} r_t + c^B \sum_{t=1}^{T} B_t + c^\delta \delta \sum_{t=1}^{T} (z_t B_t) \tag{37}$$

subject to

$$r_t + y_t \ge x_t \quad \text{for all } t = 1,...,T \tag{38}$$

$$y_t \le (1 - s_t) B_t + s_t \delta B_t \quad \text{for all } t = 1,...,T \tag{39}$$

$$y_t \le (1 - z_t)(y_{t-1} + r_{t-1} - x_{t-1}) + s_t M \quad \text{for all } t = 2,...,T \tag{40}$$

$$B_t - B_{t-1} \le M s_t \quad \text{for all } t = 1,...,T \tag{41}$$

$$B_{t-1} - B_t \le M s_t \quad \text{for all } t = 1,...,T \tag{42}$$

$$r_t - r_{t-1} \le M s_t \quad \text{for all } t = 1,...,T \tag{43}$$

$$r_{t-1} - r_t \le M s_t \quad \text{for all } t = 1,...,T \tag{44}$$

$$r_t, b_t, y_t \ge 0 \quad \text{for all } t = 1,...,T \tag{45}$$

$$s_t \in \{0, 1\} \quad \text{for all } t = 1,...,T \tag{46}$$

[6] For higher C_B the number of periods increases but still remains on a very low level. For C_B=C_r the number of periods only increases to 203 periods.

after redimensioning.

Constraint (38) makes sure there are enough tokens available each period. (39) makes sure there are no more tokens in the bucket than the bucket depth (if no redimensioning was performed that period - indicated by s_t=0) rsp. the bucket starting factor (after redimenioning). Similarly, (40) accounts for the new and used tokens if s_t=0, that is no redimensioning was performed in that period. After redimensioning (40) imposes no additional limit to y_t.

(41) to (44) force s_t to one if the bucket was redimensioned. Redimensioning equals a change in B_t and/or r_t.

(45) and (46) are the non-negativity and binary constraints for the variables.

5.3.2 Solution Algorithms. The MPRASE algorithms from Section 3.2 can again be adapted to also solve this problem. The solution is a series of token buckets which are themselves again the result of a STBD process. The different buckets are decoupled. We can solve the single token bucket dimensioning STBD problems (see Section 5.2 and [25]) between each couple of periods u,v with $1 \le u \le v \le T$ and store the optimal TB parameters (r, B) and related costs of these $T(T+1)/2$ problems. We then have to find the optimal combination of those token buckets with a modified DP algorithm (Section 3.2.2.2), the algorithm is described in more detail in [25].

Because of the relatively high complexity of the modified DP algorithm we also strive for heuristics. A possible heuristic is to use the exact algorithm above and change it so that before we solve the STBD between periods u and v, we have a look at the previous solution found for u and v-1:

- If the rate $r(u, v\text{-}1)$, the token bucket size $B(u, v\text{-}1)$ and the number of tokens remaining at the end of the period $v-1$ $y_{v-1}(u, v\text{-}1)$ are high enough to satisfy the demand of the new period v, then we extend the previous solution by one period to include v. This way the parameters are not always optimal but we do not have to solve the STBD for each sub-problem.
- Only if the previous parameters and tokens left are not sufficient we solve the STBD for (u, v).

We call this heuristic the dynamic programming heuristic DPH. We also adapted MH, SH and CH[SH+MH][7] from Section 3.2.3 to this problem.

5.3.3 Evaluation. Our basic simulation[8] uses the video traces patterns of [58]. These 21 traces are from MPEG versions of different types of video sequences (movies, cartoons, TV, sport, ...). One period represents one group of pictures (12 frames, 0.5 seconds), 2000 periods equal little more than 15 minutes of a movie. The average bit rate of the movies is 0.536 Mbps, the average peak rate of the movies is 3.54 Mbps. The cost coefficients are $c^r = c^\delta = 1$; $c^B = 0.1$ and $c^f = 10^5$, the bucket starting factor is set to $\delta = 0.5$.[9]

The DTBD was solved for different values of T ranging between 50 and 2000. We tested the exact algorithm DP and the heuristic DPH, MH, SH and CH. We also fitted a single token bucket (STB) instead of a token bucket series using an exact algorithm from [25]. We measured the CPU time, the numbers of allocations and the relative difference between the calculated cost and the optimal costs (yielded by the exact algorithm)

$$\Delta = \frac{P - P_{opt}}{P_{opt}} \tag{47}$$

The following figures 11 to 13 are based on the average over the results from each of the 21 traces.

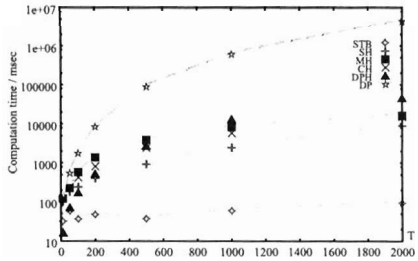

Figure 11. Computation times (logarithmic)

By looking at the computation times in Figure 11 one first notices that the exact DP algorithm takes by far the longest time to solve as can be expected as it has the highest computational complexity. The DPH heuristic is much faster than the DP algorithm and scales a little better. This indicates that in practice it can avoid solving a lot of STBDs because it can extend the previous token bucket by just one period in most of the cases. The fastest way is of course to solve the

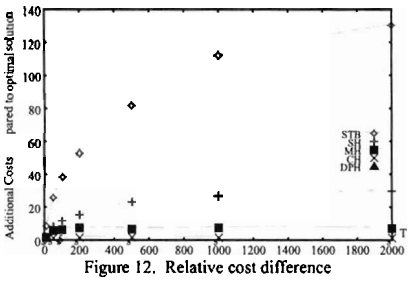

Figure 12. Relative cost difference

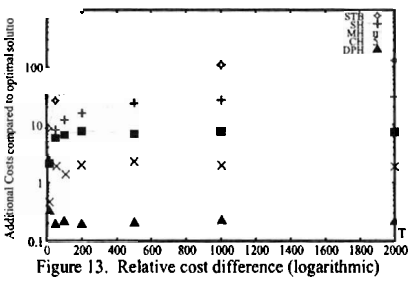

Figure 13. Relative cost difference (logarithmic)

DTBD without renegotiation as a STBD. MH, SH and CH are slower than DPH for small T but scale better and are thus faster than DPH for higher T. CH can never be faster than SH as the first step of CH is to execute SH. MH always takes longer than SH and most of the times even than CH.

Presumably more important than the execution time is the quality of the results measured by the relative difference in costs compared to the optimal costs returned by the exact algorithm as depicted in Figure 12 and 13. For a single token bucket the additional costs are far higher than for a series of token buckets. The difference increases with the number of periods T which is obvious as the potential benefit of being able to change token bucket parameters increases with T. This also clearly shows that it generally makes sense to use a series of buckets and to look at the token bucket redimensioning problem DTBD as it can very significantly reduce costs by a factor of 2 and more.

The DPH algorithm on the other hand is always extremely close to the optimal solution, resulting in less than 0.25% higher costs. SH performs quite bad, MH is better but as can be expected CH is better than SH and MH and roughly 2% away from the optimal solution.

When increasing the setup costs by a factor of 10 the number of allocations goes down by a factor of roughly 3 to 5 as can be seen in Figure 14 which shows that the setup cost are an effective way of influencing the number of used token buckets. Even with 10 times higher setup costs, using a series of token buckets instead of a single one the total costs can still be reduced by a factor of 2, the ranking of the algorithms in computation time and performance remains the same, for more details see [25].

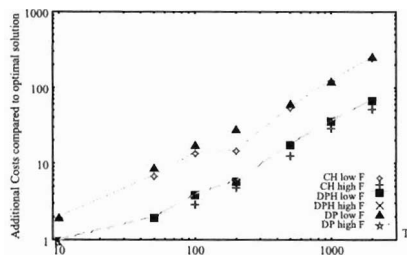

Figure 14. Number of allocations for different setup costs (logarithmic)

Instead of using the MPEG traces we now use randomly generated traffic using the *ffi_fgn* traffic generator [53, 64] generating three patterns following a fractional brownian motion process. The first pattern is a pure brownian motion pattern (Hurst parameter H=0.5), the second a fractional brownian motion pattern with a low autocorrelation of the values (Hurst parameter H=0.7) and the third is one with a strong autocorrelation of the values (Hurst parameter H=0.9).

The performance of CH and DPH is depicted in Figure 15 and 16. First of all, one notices that the performance of the algorithms degrades the lower the Hurst parameter is. The performance drop is higher for DPH than for MH, if there is no autocorrelation in the traffic (H=0.5) CH even yields better results than the DPH heuristic. This can be explained as follows: DPH extends the token bucket of a

[7] Test experiments showed that starting with SH yielded slightly better results.

[8] The simulation was done with Java as programming language on a PC with a 700MHz Pentium III Processor and a 256 MB RAM.

[9] Variation of these parameters showed no significant influence on the basic results, see [25].

previous calculation by one period $t+1$ if the bucket is big enough. This extension is the better the more the traffic of $t+1$ depends on the values t, $t-1$, ... that is the higher the autocorrelation is.

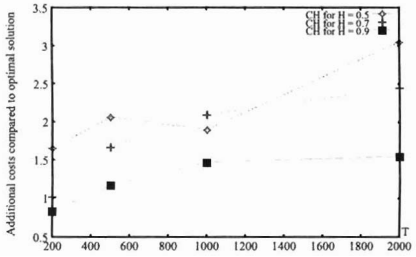

Figure 15. Performance of CH for different Hurst parameters

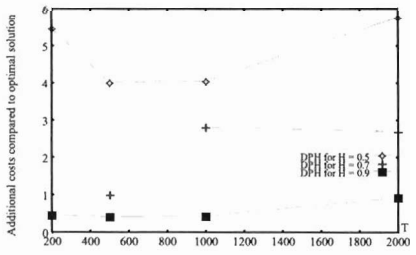

Figure 16. Performance of DPH for different Hurst parameters

In summary, DPH has the most attractive trade-off between computation time and the quality of the solution. As it is extremely close to the optimum for long-range dependent traffic and orders of magnitudes faster than the exact algorithm it can be used instead of the exact algorithm.

For very high T CH might be attractive, too, as it scales better than DPH. For short-range dependent traffic it is better than DPH, too.

5.3.4 Related Work.

Renegotiable services are also popular in literature. The $1|1|N_{TB}|FV|*|*$ problem is discussed broadly in [25].

There are some works, e.g. [5], [71], and [66], that consider optimal smoothing for guaranteed service streams. These works are different from this work in that in principle they smooth a given stream to fit into a token bucket by adding a playback delay and using buffers while this work tries to fit a single rsp. multiple serial token buckets to a given stream.

While this section tries to fit a series of token buckets to a given stream [56] tries to fit a multi-level token bucket (multiple token buckets starting at the same moment) to a video stream.

There are also a number of works on renegotiable services [23, 74, 43]. Grossglauer et. al. [23] propose the renegotiable constant bit rate service and show how it can be used to increase total network utilization. Knightly and Zhang [74, 43] extend this work to the renegotiable variable bit rate service (RED-VBR). They also consider sending an MPEG movie known in advance. They show that without renegotiation for certain MPEG streams only an average utilization of 25% can be achieved. They propose a heuristic called off-line algorithm to calculate a series of token buckets for the ATM VBR service that achieve a far higher average utilization. This heuristic needs an input parameter that controls how often to segment the stream. This parameter is difficult to set. Our work presents an exact algorithm and an extremely close yet much faster heuristic instead. Knightly and Zhang also present a second heuristic (on-line algorithm) that does not require the traffic to be known in advance and they propose an admission control scheme for renegotiable VBR services.

6. Conclusion & Outlook

This paper has described a framework and taxonomy for a class of optimization problems related to resource allocation at system edges over multiple time periods (MPRASE). The taxonomy consists of six submodels describing the individual facets of the different problem incarnations: customer, provider, resource, cost, edge and intermediary. Each submodel can be described by a short abbreviation, the combination of them then identifies the problem incarnation exactly.

We have presented two abstract MPRASE problems including the single provider problem (SPP) - the smallest non-trivial MPRASE problem.

After that we presented two uncertain and three deterministic MPRASE problem incarnations and showed that MPRASE problems occur often - although yet unrecognized - in literature which we showed in a lot of related work for the individual problems. We also showed that it makes sense to look at these problems in an integrated manner as they have many similarities, allow the reuse of algorithms and the simplification towards easier already solved MPRASE problems.

We encourage readers to make use of the framework and taxonomy and plan to further investigate interesting MPRASE problems.

References

[1] R. Bellmann and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, N.J., 1962.

[2] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine. A framework for integrated services operation over diffserv networks., 11 2000.

[3] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services, 12 1998.

[4] Ulysses Black. ATM: Foundation for Broadband Networks, 1995. Prentice Hall, Englewood Cliffs.

[5] J.-Y. Le Boudec and O. Verscheure. Optimal smoothing for guaranteed service. pages 689–696, 2000.

[6] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview, 6 1994.

[7] L. Breslau, S. Jamin, and S. Shenker. Comments on the performance of measurement-based admission control algorithms. In *Proceedings of IEEE INFOCOM 2000*, 3 2000.

[8] R. Bruno, R. G. Garoppo, and S. Giordano. Token bucket dimensioning for aggregate VoIP sources. In *Proceedings of IEEE ATM Workshop 2000*. IEEE, 6 2000.

[9] R. Callon, M. Suzuki, J. De Clercq, B. Gleeson, A. Malis, K. Muthukrishnan, E. Rosen, C. Sargor, and J. J. Yu. A framework for provider provisioned virtual private networks. http://draft-ietf-ppvpn-framework-03.txt, 1 2002. Internet Draft.

[10] Jorge A. Cobb. Preserving Quality of Service Guarantees in spite of Flow Aggregation. In *Proceedings of the 6th International Conference on Network Protocols (ICNP'98)*, pages 90–97. IEEE, November 1998.

[11] Ilog Cplex. Mathematical programming optimizer, http://www.ilog.com/products/cplex/.

[12] G. B. Dantzig. Linear programming under uncertainty. pages 197–206, 1955.

[13] K. Dovrolis, M. Prasad Vedam, and P. Ramanathan. The selection of the token bucket parameters in the IETF guaranteed service class. Technical Report Technical Report TB98, University of Delaware, http://www.cis.udel.edu/ dovrolis/Papers/tokbuck.ps, 6 1998.

[14] N. G. Duffield, P. Goyal, A. Greenberg, K. K. Ramakrishnan, and J. E. van der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of SIGCOMM*. IEEE, 8 1999.

[15] S. Dye, L. Stougie, and A. Tomasgard. The stochastic single node service provision problem. http://edoc.hu-berlin.de/speps/. 2 2002.

[16] M. Falkner, M. Devetsikiotis, and I. Lambadaris. Minimum cost traffic shaping: A user's perspective on connection admission control. pages 257–259, 9 1999. Volume 3.

[17] W. Fang and L. L. Peterson. Inter-AS traffic patterns and their implication. In *Proceedings of Global Internet Workshop at GLOBECOMM 1999*. IEEE, 12 1999.

[18] Cooperative Association for Internet Data Analysis (CAIDA). Visualizing internet topology at a macroscopic scale, http://www.caida.org/analysis/topology/as_core_network/.

[19] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press Inc., New York, USA, 1981. ISBN 0-12-283952-8.

[20] J. Glasmann, M. Czermin, and A. Riedl. Estimation of token bucket parameters for videoconferencing systems in corporate networks. In *Proceedings of SoftCOM 2000*, 10 2000.

[21] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A framework for IP based virtual private networks, draft-gleeson-vpn-framework-03.txt, 2 2000. Informational RFC 2764.

[22] T. J. Gordon and H. Haywood. Initial experiments with the cross-impact matrix method of forecasting. pages 100–116, 12 1968.

[23] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A simple and efficient service for multiple time-scale traffic. In *Proceedings of SIGCOMM 1995*, pages 219–230, 9 1995.

[24] W. K. Haneveld and M. H. van der Vlerk. Stochastic integer programming: general models and algorithms. pages 39–57, 1999.

[25] O. Heckmann, F. Rohmer, and J. Schmitt. The token bucket allocation and reallocation problems (MPRASE token bucket). Technical Report Technical Report TR-KOM-2001-12, Darmstadt University of Technology, Multimedia Communications Lab (KOM), ftp://ftp.kom.e-technik.tu-darmstadt.de/pub/papers/HRS01-1-paper.pdf, 12 2001.

[26] O. Heckmann and J. Schmitt. Multi-period resource allocation at system edges (MPRASE). Technical Report Technical Report TR-KOM-2000-05, Darmstadt University of Technology, Multimedia Communications Lab (KOM), ftp://ftp.kom.e-technik.tu-darmstadt.de/pub/papers/HS00-2-paper.pdf, 10 2000.

[27] O. Heckmann, J. Schmitt, and R. Steinmetz. Multi-Period Resource Allocation at System Edges - Capacity Management in a Multi-Provider Multi-Service Internet. In *Proceedings of IEEE Conference on Local Computer Networks (LCN 2001)*, http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/HSS01-1.html, November 2001.

[28] O. Heckmann, J. Schmitt, and R. Steinmetz. Robust Bandwidth Allocation Strategies. In *Proceedings of IWQoS 2002*, pages 138–147, http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/HSS02-1.html, May 2002.

[29] O. Heckmann and A. Scholl. Rollierende robuste Planung von Produktionsprogrammen unter systematischer Unsicherheit. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/HS00-1.html. *Schriften zur Quantitativen Betriebswirtschaftslehre*, 4(04/00), July 2000.

[30] J. Heinanen and R. Guerin. A single rate three color marker, RFC 2697, 9 1999.

[31] J. Heinanen and R. Guerin. A two rate three color marker, RFC 2698, 9 1999.

[32] F. S. Hillier and G. J. Lieberman. *Operations Research*. McGraw-Hill, 1995.

[33] G. Huston. *ISP Survival Guide: Strategies for Running a Competitive ISP*. John Wiley & Sons, 1st edition edition, 11 1998. ISBN: 0471314994.

[34] J. Ibanez and K. Nichols. Preliminary simulation evaluation of an assured service. draft-ibanez-diffserv-assured-eval-00.txt, 8 1998. Internet Draft.

[35] R. Isaacs. Lightweight, dynamic and programmable virtual private networks. In *IEEE OPENARCH*, pages 3–12. IEEE, 3 2000.

[36] R. Isaacs and I. Leslie. Support for resource-assured and dynamic virtual private networks. 2001 19(3).

[37] Internet Traffic Archive (ITA). http://ita.ee.lbl.gov/html/traces.html.

[38] P. Kall and S. W. Wallace. *Stochastic Programming*. Wiley, New York, 1994.

[39] S. Karabuk and S. D. Wu. Strategic capacity planning in the semiconductor industry: A stochastic programming approach. Technical Report Report No. 99T-12, Strategic Capacity Planning in the Semiconductor Industry: A Stochastic Programming Approach, http://citeseer.nj.nec.com/karabuk99strategic.html, 1999.

[40] S. Keshav. *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, 1997.

[41] I. Khalil and T. Braun. Implementation of a bandwidth broker for dynamic end-to-end resource reservation in outsourced virtual private networks. In *Proceedings of IEEE Conference on Local Computer Networks (LCN)*. IEEE, 11 2000.

[42] E. Knightly and N. Shroff. Admission control for statistical QoS: Theory and practice. pages 20–29, 1999. Vol. 13, No. 2.

[43] E. Knightly and H. Zhang. Connection admission control for RED-VBR, a renegotiation-based service. In *Proceedings of IWQoS 1996*, 1996.

[44] G. Kulkarni and N. Gautam. Leaky buckets: Sizing and admission control. In *Proceedings of the 35th IEEE Conference on Decision and Control*, http://citeseer.nj.nec.com/kulkarni96leaky.html, 1996

[45] C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek. On quality of service optimization with discrete QoS options. In *Proceedings of the IEEE Real-time Technology and Applications Symposium*, 6 1999.

[46] J. K. MacKie-Mason and H. Varian. Pricing congestible network resources. pages 1141–49, 9 1995. Vol. 13, No. 7.

[47] D. Meadows and J. Randers. *The Limits to Growth*. Universe books, New York, 1972.

[48] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan. VPN DESIGNER: a tool for design of multi-service virtual private networks. pages 15–31, 10-12 1998.

[49] K. Nair and R. K. Sarin. Generating future scenarios - their use in strategic planning. pages 57–66, 6 1079.

[50] J. Naudts. Towards real-time measurement of traffic control parameters. pages 157–167, 2000.
[51] Ping Pan, Ellen Hahne, and Henning Schulzrinne. BGRP: A Tree-Based Aggregation Protocol for Inter-domain Reservations. *Journal of Communications and Networks*, 2(2):157–167, June 2000.
[52] C. Partridge. Manual page of TB program, 1994.
[53] V. Paxson. Fast approximation of self-similar network traffic. Technical Report Technical Report LBL-36750, Lawrence Berkeley Laboratory, 4 1995.
[54] G. Procissi, M. Gerla, J. Kim, S. S. Lee, and M. Y. Sanadidi. On long range dependence and token buckets. In *Proceedings of SPECTS 2001*, 7 2001.
[55] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A resource allocation model for QoS management. In *Proceedings of the IEEE Real-Time Systems Symposium 1997*, http://citeseer.nj.nec.com/rajkumar97resource.html, 12 1997.
[56] A. M. Ramasamy. On policing of MPEG video streams using multi (o,p) envelopes, http://citeseer.nj.nec.com/38284.html, 1997.
[57] J. Roberts. Traffic theory and the internet. pages 94–99, 1 2001.
[58] O. Rose. Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems. Technical Report Technical Report No. 101, University of Wuerzburg, Institute of Computer Science, http://citeseer.nj.nec.com/rose95statistical.html, 2 1995.
[59] S. Sahu, P. Nain, T. Towsley, C. Diot, and V. Firoiu. On achievable service differentiation with token bucket marking for TCP. In *Proceedings of ACM SIGMETRICS 2000*. ACM, 6 2000.
[60] J. Schlembach, A. Skoe, P. Yuan, and E. Knightly. Design and implementation of scalable admission control. In *Proceedings of the International Workshop on QoS in Multiservice IP Networks 2001*, 1 2001.
[61] J. Schmitt, Oliver Heckmann, M. Karsten, and R. Steinmetz. Decoupling Different Time Scales of Network QoS Systems. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/SHKS02-1.htm% 1. *Computer Communications*, 25(11-12):1047–1057, July 2002. ISSN 0140-3664.
[62] Jens Schmitt, Martin Karsten, and Ralf Steinmetz. On the Aggregation of Deterministic Service Flows. *Computer Communications*, 24(1):2–18, January 2001.
[63] A. Scholl. *Robuste Planung und Optimierung: Grundlagen . Konzepte und Methoden . Experimentelle Untersuchungen*. Physica, Heidelberg, 1 edition, 2001.
[64] C. Schuler. fft_fgn. Research Institute for Open Communication Systems, GMD FOKUS, Hardenbergplatz 2, D-10623 Berlin, Germany.
[65] N. Semret, R. R. Liao, A. T. Campbell, and A. A. Lazar. Peering and provisioning of differentiated internet services. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, pages 414–420. IEEE, 3 2000.
[66] S. Sen, D. Towsley, Z.-L. Zhang, and J. Dey. Optimal multicast smoothing of streaming video over an internetwork. 2002.
[67] H. Su and M. Atiquzzaman. Performance modeling of differentiated service network with a token bucket marker. In *Proceedings of the 11th IEEE Workshop on Local and Metropolitan Area Networks*, pages 81–82, 3 2001.
[68] P. Tang and T. Tai. Network traffic characterization using token bucket model. In *In Network Traffic Characterization Using Token Bucket Model*, pages 51–62, 1999.
[69] Andreas Terzis, John Krawczyk, John Wroclawski, and Lixia Zhang. RSVP Operation over IP Tunnels. Proposed Standard RFC 2746, January 2000.
[70] Andreas Terzis, Lan Wang, Jun Ogawa, and Lixia Zhang. A Two-Tier Resource Management Model for the Internet. In *Proceedings of Global Internet Workshop at GLOBECOMM'99*, pages 1779–1791. IEEE, December 1999.
[71] P. Thiran, J.-Y. Le Boudec, and F. Worm. Network calculus applied to optimal multimedia smoothing. In *Proceedings of INFOCOM 2001*, pages 1474–1483, 2001.
[72] A. Tomasgard, S. Dye, S. W. Wallace, J. A. Audestad, L. Stougie, and M. H. van der Vlerk. Stochastic optimization models for distributed communication networks. Technical Report Working paper #3-97, Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, 1997.
[73] H. Vladimirou, S. A. Zenios, and R. J-B. Wets (editors). *Models for planning under uncertainty*. Annals of Operations Research, Vol. 59, J.C. Baltzer AG Scientific Publishers, 1995.
[74] H. Zhang and E. Knightly. RED-VBR: A renegotiation-based approach to support delay-sensitive VBR video. pages 164–176, 5 1997. Vol. 4, No. 3.

Oliver Heckmann. Oliver Heckmann finished his degree in economics and electrical engineering (Wirtschaftsingenieur) in March 2000 at the Darmstadt University of Technology. During his studies he focused on hardware verification, optimization techniques and electronic commerce.

Since June 2000 he is a researcher in the Multimedia Distribution & Networking group of the Multimedia Communications Lab (KOM) of Prof. Steinmetz at the Darmstadt University of Technology. His research area is the Internet Service Provider in a Multiservice Internet. Within that area he is interested especially in quality of service, robust network design, traffic engineering, network simulation and the influence of peer-to-peer networking. He was working in the M3I project (www.m3i.org) and is now working in the LETSQoS project (www.letsqos.de).

Jens Schmitt. Jens Schmitt is research group leader of the Multimedia Distribution & Networking group at the Multimedia Communications Lab (KOM) of the Darmstadt University of Technology, where he works in the fields of Quality of Service (QoS) provisioning in distributed systems, in particular in heterogeneous network scenarios, QoS for mobile communications, and scalable distribution of multimedia content with an emphasis on high availability systems. Further research interests are in network traffic modelling, real-time scheduling, and evolutionary programming.

Jens Schmitt received his Master degree (Dipl.) from the University of Mannheim in Joint Business and Computer Sciences in 1996. In 1994, during a stay at the University of Wales, Swansea, he also did a European Master of Business Sciences degree. In 2000 he received his Ph.D. degree (Dr.-Ing.) from Darmstadt University of Technology, his thesis was about "Heterogeneous Network Quality of Service Systems".

Ralf Steinmetz. Ralf Steinmetz received the M.Sc. (Dipl.-Ing.) degree and the Ph.D. (Dr.-Ing.) degree, working in the area of Petri nets and concurrent programming languages with focus on communications, from the Darmstadt University of Technology, both in electrical engineering, in 1982 and 1986, respectively.

He was research engineer at Philips and IBM. He became responsible for the establishment, definition and realization of multimedia projects at IBM, Heidelberg. In January 1993 he was nominated as IBM "Chief Designer" for IBM Germany. From 1997 to 2001 he has been the Director of the GMD (German - National Research Center for Information Technology) institute IPSI (Integrated Publications- and Information Systems Institute) in Darmstadt. In 1999 he founded the hitc (Hessian Telemedia Technology Competence Center) with focus on applied networked multimedia services being applied to education projects.

He is an ICCC Governor, IEEE Fellow, and ACM Fellow.

Since 1996 Ralf Steinmetz is Professor at the Departments of "Electrical Engineering and Information Technology" and "Computer Science" at the Darmstadt University of Technology and head of the "Multimedia Communications" (KOM) Lab. His research interests are the areas of multimedia distribution, multimedia networking and multimedia semantics.