

# A Generalized Analysis Technique for Queueing Networks with Mixed Priority Strategy and Class Switching

Stefan Greiner  
Duke University  
Dept. of Electrical Engineering  
Durham, NC 27708-0291, USA  
Box 90291  
greiner@ee.duke.edu

Gunter Bolch  
University Erlangen-Nuernberg  
IMMD IV  
Martensstrasse 1  
D – 91058 Erlangen  
bolch@informatik.uni-erlangen.de

## Abstract

A very well known and popular technique for the performance evaluation of current computer and operating systems is analytical modeling based on queueing networks because it is easy to understand and use compared to other modeling techniques. For so called product form queueing networks efficient exact and approximate analysis algorithms exist but most queueing networks of realistic systems do not fulfill the requirements of product form queueing networks. To solve this kind of networks Markovian analysis or discrete event simulation is used. The disadvantage of these techniques is that they are very time consuming to prepare, implement and run, especially when we want to analyze the system on a wide range of parameter values. Because of these disadvantages our goal is to transform queueing networks that can not be solved with standard analysis techniques into networks that can be solved with standard analysis techniques (like MVA, SCAT or Convolution). Problems that appear very often in realistic systems are priorities with and without preemption, mixed priorities and class switching. These problems appear especially in the field of operating system modeling [BoGr94, GBT94, Jung91]. The paper is organized as follows. In Section 1 we introduce the notation used in this paper and present already existing techniques to solve networks with priorities. Then we show how to combine the presented techniques to solve networks with a certain type of mixed priority classes. In Section 2 we introduce the concept of chains to handle open, closed and mixed networks with class switching. The goal is to derive a solution technique for any type of mixed priority strategy. Therefore the well known shadow technique is introduced and extended for open, closed and mixed networks with class switching and priorities. In Section 3 we apply these techniques to a simple non product form network with the problems mentioned above. It is shown how this network can be transformed into a product form network and solved using a standard analysis technique. In Section 4 the presented techniques are demonstrated on the example of a UNIX based multiprocessor operating system kernel.

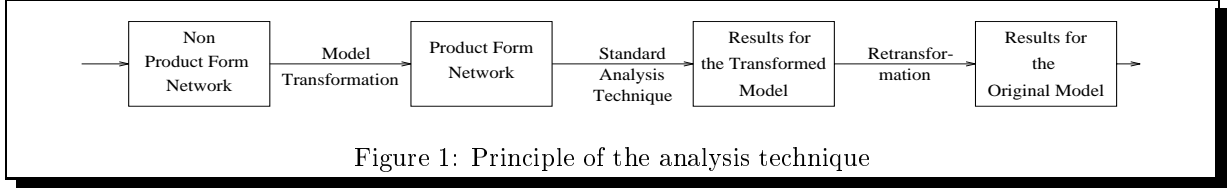
**Keywords:** Product and non product form queueing networks, class switching, shadow technique, preemptive and non preemptive priority classes, UNIX operating system.

## 1 Introduction

### 1.1 Problem

In todays computer and software environment problems get more and more complex. If a mistake is made in the design process then this can go through the whole project and reduce the system performance dramatically. Therefore it is desirable to get performance measures *before* the system is built. This is the point when system modeling comes in. Usually a mathematical model of the system which has to be

analyzed is made. A very well known technique is modeling by using queueing networks because they are easy to understand and use compared to other modeling and analysis techniques. For so called product form queueing networks efficient analysis algorithms are known but most queueing networks of realistic systems do not fulfill the requirements of a product form network. Especially in realistic systems we have priorities with and without preemption, mixed priority strategies, class switching and heterogeneous nodes. These networks can not be solved with the standard analysis techniques. The goal therefore is to transform such non product form networks in a way so that they can be analyzed using a standard analysis technique (like MVA, SCAT or Convolution). The principle of this technique is shown in Fig. 1. It should



be noted that we can use any unmodified standard analysis technique to analyze the transformed model. To demonstrate how this new technique works we use a simple non product form queueing network and the three models of a UNIX based multiprocessor operating system [Jung91] (see section 4.3).

## 1.2 Notation

- $N$  : number of nodes in the network.
- $R$  : number of job classes.
- $\underline{k}$  : population vector  $(k_1, k_2, \dots, k_R)$  where  $k_i$  is the number of jobs in class  $i$  ( $1 \leq i \leq R$ ) in the network.
- $K$  : number of jobs in the closed network with

$$K = \sum_{i=1}^R k_i$$

- $(\underline{k} - 1_r)$  : population vector with one class  $r$  job removed.
- $\mu_{ir}$  : service rate of a class  $r$  job at node  $i$ .
- $s_{ir} = \frac{1}{\mu_{ir}}$  : service time of a class  $r$  job at node  $i$ .
- $\lambda_{ir}$  : arrival rate of a class  $r$  job at node  $i$ .
- $m_i$  : number of servers at node  $i$ .
- $p_{ir,js}$  : probability that a class  $r$  job moves to class  $s$  at node  $j$  after completing service at node  $i$ .
- $e_{ir}$  : mean number of visits of a class  $r$  job at node  $i$ .

$$e_{ir} = \sum_{j=1}^N \sum_{s=1}^R e_{js} \cdot p_{js,ir}$$

- $\rho_{ir}$  : utilization of node  $i$  by class  $r$  jobs  $\rho_{ir} = \frac{\lambda_{ir}}{m_i \cdot \mu_{ir}}$ .
- $\bar{w}_{ir}$  : mean waiting time of a class  $r$  job at node  $i$ .
- $\bar{t}_{ir}$  : response time of a class  $r$  job at node  $i$ .
- $\bar{k}_{ir}$  : mean number of class  $r$  jobs at node  $i$ .
- $\bar{q}_{ir}$  : mean number of class  $r$  jobs that are in the queue of node  $i$ .

## 1.3 Queueing Networks without Class Switching

### 1.3.1 Description

If class switching is not allowed in the network then the number of jobs in a class is always constant. For this type of network a large number of exact and approximate analysis methods is known. Many of these techniques are implemented in the tool PEPSY-QNS (**P**erformance **E**valuation and **P**rediction **S**ystem for **Q**ueueing **N**etwork**S**) at the Institute for mathematical machines and dataprocessing IV (IMMD IV) Erlangen Nuernberg. The rest of the paper is based on the well known mean value analysis (MVA) [ReLe80] as standard analysis technique but our new technique is not only restricted to this analysis technique.

### 1.3.2 Mean Value Analysis

The MVA was developed by Reiser and Lavenberg [ReLa80] and can be used for the exact solution of queueing networks which consist of the following types of nodes (so called product form nodes) :

- Type 1 : M / M / m - FCFS
- Type 2 : M / G / 1 - PS (RR)
- Type 3 : M / G /  $\infty$  (Infinite Server)
- Type 4 : M / G / 1 - LCFS PR

The following two laws are fundamental to this analysis technique :

- Little's law :  $\bar{k} = \lambda \cdot \bar{t}$
- Arrival theorem by Reiser and Lavenberg :

$$\bar{t}_i(K) = \frac{1}{\mu_i} \cdot (1 + \bar{k}_i(K - 1))$$

The following formulas are central for the MVA and can be derived from the arrival theorem. All further modifications are based on these formulas.

$$\bar{t}_{ir}(k) = \begin{cases} \frac{1}{\mu_{ir}} \cdot \left( 1 + \sum_{s=1}^R \bar{k}_{is}(\underline{k} - 1_r) \right) & \begin{matrix} \text{Type-1,2,4} \\ m_i = 1 \end{matrix} \\ \frac{1}{\mu_{ir} \cdot m_i} \cdot \left( 1 + \sum_{s=1}^R \bar{k}_{is}(\underline{k} - 1_r) + \sum_{j=0}^{m_i-2} (m_i - j - 1) \cdot p_i(j | \underline{k} - 1_r) \right) & \begin{matrix} \text{Type-1} \\ (m_i > 1) \end{matrix} \\ \frac{1}{\mu_{ir}} & \text{Type-3} \end{cases}$$

### 1.3.3 Priorities

If priority scheduling is used in the network then the product form assumptions are no longer valid. Therefore one needs an approximate solution. We assume that the priority classes are ordered linearly where class 1 has the highest priority. In the derivation of the formulas at first only one node is considered. The results are then integrated into the closed network using the mean value analysis technique.

**Preemptive Resume (PR)** The characteristic of this strategy is that as soon as a high priority job arrives at the system this job is served if no other job with higher priority is in service at the moment. If a job with lower priority is in service when the high priority job arrives then this low priority job is preempted. If all jobs with higher priority are served then the preempted job can be served again at the point where it was preempted. In this case the mean response time of a job consists of

- the service time

$$\frac{1}{\mu_r}$$

- the time one has to wait until all jobs with higher or the same priority are served

$$\sum_{s=1}^r \frac{\bar{k}_s}{\mu_s}$$

- the time for serving all jobs with a higher priority that arrive during the response time  $\bar{t}_r$

$$\sum_{s=1}^{r-1} \frac{\bar{t}_r \cdot \lambda_s}{\mu_s}$$

For the priority classes  $r$  ( $1 \leq r \leq R$ ) we get [Bolc89]

$$\bar{t}_r = \sum_{s=1}^r \frac{\bar{k}_s}{\mu_s} + \sum_{s=1}^{r-1} \frac{\bar{t}_r \cdot \lambda_s}{\mu_s} + \frac{1}{\mu_r}$$

By factorizing this term and using the arrival theorem one gets for node  $i$  [BKLC84]

$$\begin{aligned} \bar{t}_{ir}(\underline{k}) &= \frac{\frac{1}{\mu_{ir}} + \sum_{s=1}^r \frac{\bar{k}_{is}(\underline{k}-1_r)}{\mu_{is}}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{is}} \\ \tilde{\rho}_{is} &= \rho_{is}(\underline{k} - \bar{k}_{is}) \end{aligned}$$

**Non preemptive (HOL)** Here an arriving job with higher priority has to wait until the job that is processed at the moment is ready to leave the node even if its priority is lower than the priority of the arriving job. In this case the mean response time of a job consists of

- the service time

$$\frac{1}{\mu_r}$$

- the time that is needed to serve a job with higher priority that arrives during the waiting time  $(\bar{t}_r - \frac{1}{\mu_r})$

$$\sum_{s=1}^{r-1} \left( \bar{t}_r - \frac{1}{\mu_r} \right) \cdot \frac{\lambda_s}{\mu_s}$$

- the time that is needed for the job that is served at the moment

$$\sum_{s=1}^R \frac{\rho_s}{\mu_s}$$

- the service time for jobs with the same or higher priority that are still in the queue

$$\sum_{s=1}^r \frac{\bar{k}_s - \rho_s}{\mu_s}$$

For the priority classes  $r$  ( $1 \leq r \leq R$ ) one gets [Bolc89]

$$\bar{t}_r = \sum_{s=1}^r \frac{\bar{k}_s - \rho_s}{\mu_s} + \sum_{s=1}^R \frac{\rho_s}{\mu_s} + \sum_{s=1}^{r-1} \left( \bar{t}_r - \frac{1}{\mu_r} \right) \cdot \frac{\lambda_s}{\mu_s} + \frac{1}{\mu_r}$$

By factorizing this term and using the arrival theorem one gets for node  $i$  [BKLC84]

$$\begin{aligned} \bar{t}_{ir}(\underline{k}) &= \frac{1}{\mu_{ir}} + \frac{\sum_{s=1}^r \frac{\bar{k}_{is}(\underline{k}-1_r)}{\mu_{is}} + \sum_{s=r+1}^R \frac{\rho_{is}(\underline{k}-1_r)}{\mu_{is}}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{is}} \\ \tilde{\rho}_{is} &= \rho_{is}(\underline{k} - \bar{k}_{is}) \end{aligned}$$

If we use these formulas instead of the original  $\bar{t}_{ir}$  in the MVA we get an approximate technique for analyzing priority queueing networks with a PR or HOL strategy. These formulas are well suited for utilizations lower than 0.7. For higher utilizations, the accuracy of these formulas is not very high for class 2... $R$  jobs but for class 1 jobs the results are very good.

### 1.3.4 Combination of PR and HOL

Now we want to show how to combine the two presented priority types to analyze priority structures of the form shown in Fig. 2. The new idea behind this grouping strategy is used later on in the paper to allow for any type of mixed priority strategy. Let us look now at a mixed strategy like the one in the

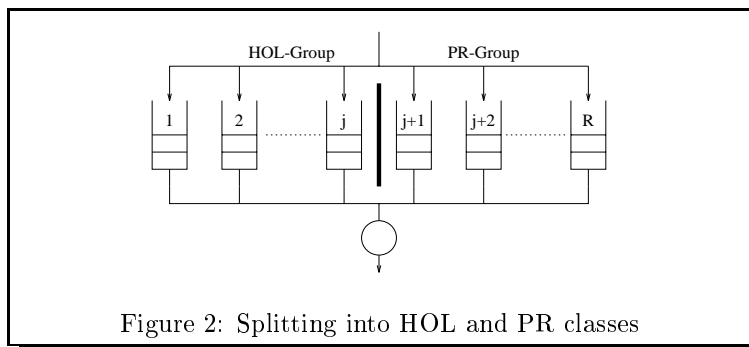


Figure 2: Splitting into HOL and PR classes

UNIX-multiprocessor models of [Jung91]. That means that some jobs are processed by HOL and some jobs are processed by PR. It is assumed that HOL-jobs have always higher priority than PR jobs (see Fig. 2). This means that we split the jobs in the two classes HOL and PR with the following assumptions :

- an arriving HOL job can not preempt a HOL job that is still in progress, but it can preempt a PR job that is still in progress. Therefore the response time of a HOL job is not influenced by the arrival of PR jobs.

$$\bar{t}_r = \sum_{s=1}^r \frac{\bar{k}_s - \rho_s}{\mu_s} + \sum_{s=1}^j \frac{\rho_s}{\mu_s} + \sum_{s=1}^{r-1} \left( \bar{t}_r - \frac{1}{\mu_r} \right) \cdot \frac{\lambda_s}{\mu_s} + \frac{1}{\mu_r}$$

By factorizing this term one gets for node  $i$

$$\begin{aligned}\bar{t}_{ir}(\underline{k}) &= \frac{1}{\mu_{ir}} + \frac{\sum_{s=1}^r \frac{\bar{k}_{is}(\underline{k}-1_r)}{\mu_{is}} + \sum_{s=r+1}^j \frac{\rho_{is}(\underline{k}-1_r)}{\mu_{is}}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{is}} \\ \tilde{\rho}_{is} &= \rho_{is}(\underline{k} - \bar{k}_{is})\end{aligned}$$

- a PR-job can always be preempted by a job with higher priority. For priority class  $r$  we therefore get

$$\bar{t}_r = \sum_{s=1}^r \frac{\bar{k}_s}{\mu_s} + \sum_{s=1}^{r-1} \frac{\bar{t}_r \cdot \lambda_s}{\mu_s} + \frac{1}{\mu_r}$$

By factorizing this term one gets for node  $i$

$$\begin{aligned}\bar{t}_{ir}(\underline{k}) &= \frac{1}{\mu_{ir}} + \frac{\sum_{s=1}^r \frac{\bar{k}_{is}(\underline{k}-1_r)}{\mu_{is}}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{is}} \\ \tilde{\rho}_{is} &= \rho_{is}(\underline{k} - \bar{k}_{is})\end{aligned}$$

If these formulas for  $t_{ir}(\underline{k})$  are used in the MVA instead of  $\bar{t}_{ir}$ , then we get an approximate MVA for the analysis of closed queueing networks with a mixed priority strategy.

## 2 Queueing Networks with Class Switching

### 2.1 Description

Each node has its own set of transition probabilities which characterize the transition to other nodes and classes. But this means that the number of jobs is not constant any longer. For a queueing model without class switching the following four conditions must hold for a feasible state [BrBa80]:

1. the number of jobs in each class at each node is always non negative, i.e.

$$k_{ir} \geq 0 \quad \forall 1 \leq r \leq R \quad \wedge \quad 1 \leq i \leq N.$$

2. for all jobs the following condition must hold :

$$k_{ir} > 0 \iff \text{there exists a way for class } r \text{ jobs} \\ \text{to node } i \text{ with a probability bigger than zero.}$$

3. the number of jobs at a node is given by

$$K = \sum_{r=1}^R \sum_{i=1}^N k_{ir}$$

4. the sum of class  $r$  jobs in the network is constant at any time

$$k_r = \sum_{i=1}^N k_{ir} = \text{const.} \quad \forall 1 \leq r \leq R$$

If class switching is allowed then the conditions 1 - 3 are fulfilled, but condition 4 can not be fulfilled because the number of jobs within a class is not longer constant but depends on the time when we look at the system and can have the values  $x \in \{0..K\}$ . In order to prevent this situation the concept of chains [BrBa80] is introduced.

## 2.2 The Concept of Chains

### 2.2.1 The Idea behind the Chains

Look at a general transition probability matrix  $P$  with a finite state space,  $S$ . This state space can be partitioned into disjoint sets

$$S = \pi_1 + \pi_2 + \dots + \pi_k$$

with  $\pi_i$  : set of recurrent states. With a possible relabeling of states, we consider the transition probability

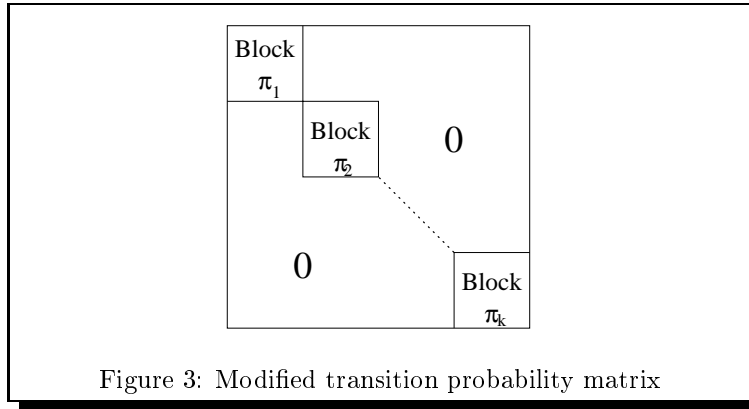


Figure 3: Modified transition probability matrix

matrix of Fig. 3. Here the single chains  $\pi_i$  are disjoint, ergodic and form a new transition probability matrix. Because of the disjointness of the chains  $\pi_i$  it is impossible for a job to switch from one chain to another. If a job starts in a recurrent block it will never leave this block. With the help of this partitioning technique the number of jobs in each chain is always constant. Therefore condition 4 for queueing networks is fulfilled for the chains. Muntz [Munt73] proved that a closed queueing network with  $U$  chains is equivalent to a closed queueing network with  $U$  job classes. If class switching is not possible in the network then the number of chains is equal to the number of classes. But if class switching is possible then the number of chains is smaller than the number of classes. This means that the dimension of the population vector is reduced.

### 2.2.2 How to find the Chains

The transition probability matrix  $P = [p_{ir,js}]$  defines a discrete-time Markov chain whose states are pairs of the form  $(i, r)$  with

- $i$  : node number
- $r$  : class number

1. define  $R$  sets that satisfy the following condition

$$E_r = \{s : \text{the state } (j, s) \text{ can be reached in } \geq 0 \text{ steps if one starts with the state } (i, r)\}.$$

$$\forall 1 \leq r \leq R \quad \wedge \quad 1 \leq s \leq R$$

2. eliminate all subsets and identical sets so that we have

$$U \leq R \quad \text{sets} \quad \pi_1 \dots \pi_U$$

3. compute the number of jobs in each chain

$$K'_s = \sum_{r \in \pi_s} K_r \quad \forall 1 \leq s \leq U$$

### 2.2.3 Example

Let the following transition probability matrix be given

P	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(3,3)
(1,1)	0	0.4	0	0	0.3	0	0	0.3	0
(1,2)	0.3	0	0	0	0	0	0	0.7	0
(1,3)	0	0	0	0	0	0.3	0	0	0.7
(2,1)	0	0	0	0	1	0	0	0	0
(2,2)	0.3	0	0	0	0	0	0.7	0	0
(2,3)	0	0	0.3	0	0	0	0	0	0.7
(3,1)	1.0	0	0	0	0	0	0	0	0
(3,2)	0	0	0	1.0	0	0	0	0	0
(3,3)	0	0	0.4	0	0	0.6	0	0	0

By using the chain technique we get

$$E_1 = \{1,2\}$$

$$E_2 = \{1,2\}$$

$$E_3 = \{3\}$$

By eliminating the subsets and identical sets we get the two chains

$$\pi_1 = \{1,2\}$$

$$\pi_2 = \{3\}$$

The new reorganized transition probability matrix has then the form

P	(1,1)	(1,2)	(2,1)	(2,2)	(3,1)	(3,2)	(1,3)	(2,3)	(3,3)
(1,1)	0	0.4	0	0.3	0.3	0	0	0	0
(1,2)	0.3	0	0	0	0	0.7	0	0	0
(2,1)	0	0	0	1.0	0	0	0	0	0
(2,2)	0.3	0	0	0	0.7	0	0	0	0
(3,1)	1.0	0	0	0	0	0	0	0	0
(3,2)	0	0	1.0	0	0	0	0	0	0
(1,3)	0	0	0	0	0	0	0	0.3	0.7
(2,3)	0	0	0	0	0	0	0.3	0	0.7
(3,3)	0	0	0	0	0	0	0.4	0.6	0

If a job starts in one of the blocks  $\pi_1$  or  $\pi_2$  then it can never leave that block.

## 2.3 Extension of the MVA to Handle Class Switching

An extension of the MVA to deal with networks with class switching is very easy if one has the concept of chains and one gets an algorithm that is isomorphic to the original MVA algorithm. Because of the switch from classes to chains it is necessary to transform class measures into chain measures [BrBa80].

- number of visits.  
In a chain a job can reach different states  $(i, j)$  where  
 $i$  = node number  
 $j$  = class number

$$e_{iq}^* = \frac{\sum_{r \in \pi_q} e_{ir}}{\sum_{r \in \pi_q} e_{1r}}$$

- the number of jobs per class is constant in a chain but within the chain jobs can switch to another class. If therefore a job of chain  $q$  visits node  $i$  it is impossible to know from which class that job is because we look at a chain as a homogeneous construct. If we make the transformation

$$\text{class} \quad \Longrightarrow \quad \text{chain}$$

the information about the class of a job is lost. Because of the fact that different job classes have different service times, this missing information is substituted by the scale factor  $\alpha$ . For the service rate in a chain one gets:

$$s_{iq} = \frac{1}{\mu_{iq}} = \sum_{r \in \pi_q} s_{ir} \cdot \alpha_{ir} \quad \alpha_{ir} = \frac{e_{ir}}{\sum_{s \in \pi_q} e_{is}}$$

### 2.3.1 The Modified MVA for Closed Queueing Networks

1. calculate the number of visits  $e_{ir}$  in the original network.
2. partition the transition probability matrix  $P$  in chains  $\pi_1 \dots \pi_U$  and calculate the number of jobs  $k_i^*$  in each chain.
3. calculate the number of visits  $e_{iq}^*$  for each chain.
4. calculate the scale factors  $\alpha_{ir}$ .
5. calculate the service times  $s_{iq}$  for each chain and node.
6. calculate the performance parameters for each chain using the extended MVA where we treat chains in the same way as classes in a system without class switching.
7. retransform the performance measures per chain into the equivalent performance measures per class.

$$\begin{aligned} \bar{t}_{ir}(\underline{k}_U) &= s_{ir} \cdot (1 + k_i^*(\underline{k}_U - 1_q)) \\ \lambda_{ir}(\underline{k}_U) &= \alpha_{ir} \cdot \lambda_{iq}^*(\underline{k}_U) \quad r \in \pi_q \\ \rho_{ir}(\underline{k}_U) &= s_{ir} \cdot \lambda_{ir}(k_U) \\ \bar{k}_{ir}(\underline{k}_U) &= \lambda_{ir}(k_U) \cdot \bar{t}_{ir}(\underline{k}_U) \end{aligned}$$

### 2.3.2 The Modified MVA for Open Queueing Networks

The mean value analysis technique can not be applied to open queueing networks but if we transform the open network into a closed network using the so called *closing method* [BGJ92] then our technique can also be applied to open networks.

#### The Closing Method

This method was developed by [BGJ92] and enables us to use algorithms for closed queueing networks also for open queueing networks. The idea of the closing method is to substitute in an open queueing network with a general distributed source the source by a  $\cdot/G/1$ -node (see Fig. 4). The new node values are given by the open network:

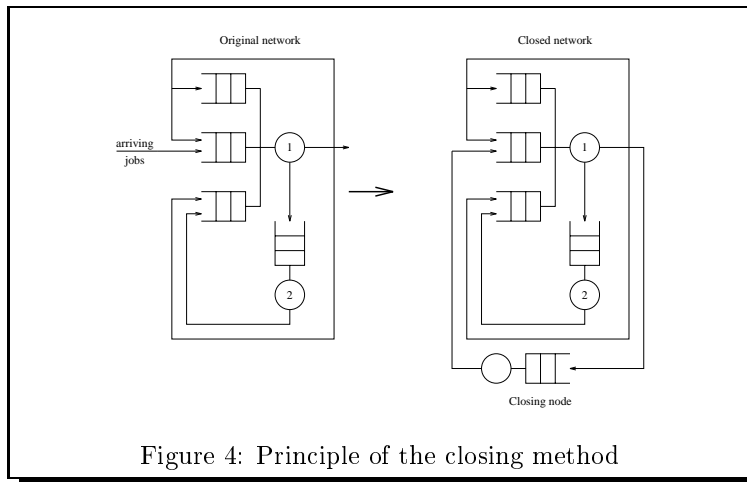


Figure 4: Principle of the closing method

- the service rate  $\mu_x$  of the closing node is given by the arrival rate  $\lambda_{01}$ . For multiple class queueing networks we have  $\mu_{xr} = R \cdot \lambda_{01,r}$  (with  $r = 1 \dots R$  and  $R =$  number of job classes).
- the coefficient of variation  $c_{B_x}$  of the service process is given by the arrival process of the open network. The utilization of the new node should be as close to 1 as possible, so that this node simulates the arrival process of the given open network. Therefore one usually needs a very high number of jobs in the closed network ( $k > 100$ ), with the most jobs being in the closing node.

After this transformation the standard mean value analysis can be used to analyze the system.

For mixed queueing networks we can also use the closing method to obtain performance measures. Here the closing node is used only for the open classes (only jobs from the open classes can enter the closing node). After that transformation we get a closed queueing network which can be analyzed using any suitable analysis algorithm for closed queueing networks.

**Notes:**

Remember that

- the dimension of the population vector is  $U \leq R$ .
- in the case of  $U = R$  class switching is not possible.
- existing algorithms need not be changed but only extended in the following way :
  - determine the chains of the system.  
If the number of chains equals the number of classes then class switching is not allowed and the MVA can be used with the original parameters of the network. If the number of chains is smaller than the number of classes then class switching is allowed and one has to carry out the MVA on chain values.

With the help of the chain-based technique it is possible to have an exact analysis of networks with class switching (but without priorities).

## 2.4 Networks with Class Switching and Priorities

The first opinion might be to use the formulas that were presented in the case of networks without class switching (see section 1.3.3, 1.3.3) but this is not possible because

- when we make the transition from classes to chains the dimension of the population vector is usually reduced and therefore an expression like

$$\underline{k}_{ir} - 1_s$$

does not make any sense if one wants to use a component  $s > U$ .

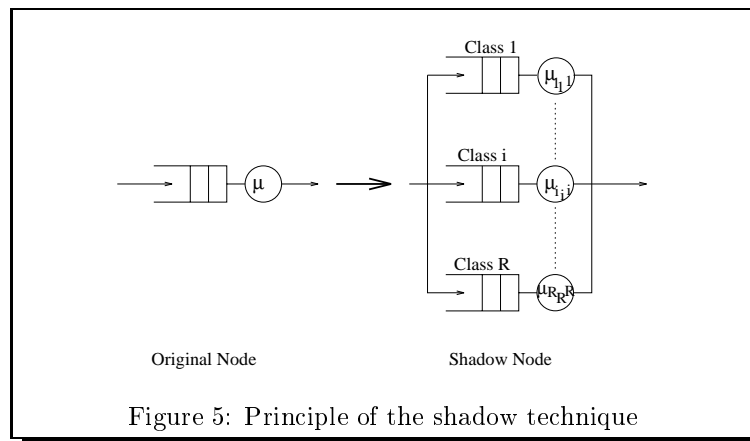
- a chain can contain jobs of different priorities and therefore a summation over all classes with higher priorities is not longer possible because we have to sum over chains.

We therefore need a technique to analyze priority networks where it is not necessary to sum over all jobs with higher priorities. For this reason the idea of shadow server is chosen. This technique is then extended to handle networks with class switching and priorities.

### 2.4.1 The Shadow Technique

This technique, shown in Fig 5, was introduced by [Sevc77] for networks with a pure PR-strategy. The principle of this technique is to split each node in the network into  $R$  parallel nodes, one for each priority class. The problem then is that in this extended node jobs can run in parallel whereas this is not possible in the original node. To deal with this problem and to simulate the effect of preemption one iteratively increases the service time for each job class in the shadow node. The lower the priority of a job the higher is the service time for the job. After all iterations we have

$$s_{i,j} \geq s_{i,j}$$



The fact that for the computation of the service rates no summation over other priority classes is necessary makes this technique suitable for priority networks with class switching. To differentiate between a shadow node and the original node a notation of the form  $(i_j, r)$  is used for the shadow nodes with

- $i$  : original node,
- $j$  : shadow node and
- $r$  : job class.

## 2.4.2 Shadow Algorithm

1. transform the original model into the shadow model.
2. set  $\lambda_{i_j,r} = 0$
3. iterate
  - (a) compute the utilization of each shadow node

$$\tilde{\rho}_{i_j,r} = \begin{cases} \lambda_{i_r,r} \cdot s_{i_r,r} & \text{if } j = r \\ 0 & \text{otherwise} \end{cases}$$

- (b) compute the shadow service times

$$s_{i_j,r} = \begin{cases} \frac{s_{i,r}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{i_s,s}} & \text{if } j = r \\ 0 & \text{otherwise} \end{cases}$$

Here  $s_{i,r}$  is the original service time of a class  $r$  job at node  $i$  in the original network. For nodes with  $s_{i_j,r} = 0$  the corresponding visit ratio must be set to 0. All other visit ratios remain unchanged.

- (c) compute the performance measures of the shadow model using the MVA, i.e. use the MVA with the  $s_{i_j,r}$  and compute the throughputs  $\lambda_{i_j,r}$  that are needed for the next iteration. If the  $\lambda_{i_j,r}$  differ less than  $\varepsilon$  in two successive iterations then stop the iteration. Otherwise go back to step 3a.

## 2.5 Extended Shadow Technique

Measurements on real systems and comparisons with simulation results have shown that the technique of shadow server is not very accurate. Because of this, [Kauf84] suggested an extension of the original shadow technique. This extended method differs from the original method in the way that the expression

$$\sum_{s=1}^{r-1} \tilde{\rho}_{i_s,s}$$

in the iteration is multiplied with a correction factor. This correction factor prevents the iteratively computed service times from growing too fast. For this reason the correction factor  $\delta$  is defined as follows [Kauf84]:

$$\delta_{i_j,r} = \begin{cases} \frac{\rho(r)[1 - \rho(r)] + \alpha(r) \cdot \rho(r+1)}{\rho(r)[1 - \rho(r)]^2 + \alpha(r) \cdot \rho_{i_r,r}} & \text{for } r = 2 \dots R \quad \text{and} \quad j = r \\ 0 & \text{otherwise} \end{cases}$$

with

$$\begin{aligned} \alpha(r) &= \sum_{k=1}^{r-1} \frac{\rho_{i_j,r}}{\omega_{r,k}} \\ \omega_{r,k} &= \frac{s_{i,k}}{s_{i,r}} \\ \rho(r) &= \sum_{k=1}^{r-1} \rho_{i_k,k} \end{aligned}$$

This correction factor  $\delta$  changes only step 3b in the original shadow algorithm:

$$s_{i_j,r} = \begin{cases} \frac{s_{i,r}}{1 - \sum_{s=1}^{r-1} \frac{1}{\delta_{i_s,s}} \cdot \tilde{\rho}_{i_s,s}} & \text{if } j = r \\ 0 & \text{otherwise.} \end{cases}$$

If the utilizations of the shadow nodes are very close to 1 then the expression

$$1 - \sum_{s=1}^{r-1} \frac{1}{\delta_{i_s,s}} \cdot \tilde{\rho}_{i_s,s}$$

can be negative because of the approximate technique. In this case the iteration must be stopped.

### 2.5.1 Extended Shadow Technique with Bisection

- Problem :

In the original and extended shadow method the expression

$$\sum_{s=1}^{r-1} \tilde{\rho}_{i_s,s}$$

can be very close to 1. Therefore the service time  $s_{i_r,r}$  becomes very big and the throughput through this node is very low. But then the value of

$$\sum_{s=1}^{r-1} \tilde{\rho}_{i_s,s}$$

is very small in the next iteration step and the shadow technique does not converge. This problem was not considered in the techniques until now. Therefore the shadow technique is extended to a new variant which leads to the shadow technique with bisection.

- Solution

For the solution of this problem one has to take into account the fact that the real values for the performance parameters are somewhere between the values that move back and forth. The idea now is to extend the MVA by bisection but this technique is used only when the values in the normal analysis technique start to move back and forth. In this case the new response time is computed as the mean value of the two response times that move back and forth. This response time is then used as the new response time for the MVA. With this technique it is possible to get out of the unstable state within a few steps and the technique converges.

### 2.5.2 Extended Shadow Technique with Class Switching

1. determine the chains in the network.
2. transform the original model into the shadow model.
3. set  $\lambda_{i_j,r} = 0$
4. iterate
  - (a)

$$\tilde{\rho}_{i_j,r} = \begin{cases} \lambda_{i_r,r} \cdot s_{i_r,r} & \text{if } j = r \\ 0 & \text{otherwise} \end{cases}$$

(b)

$$s_{i_j,r} = \begin{cases} \frac{s_{i,r}}{1 - \sum_{s=1}^{r-1} \frac{1}{\delta_{i_s,s}} \bar{\rho}_{i_s,s}} & \text{if } j = r \\ 0 & \text{otherwise} \end{cases}$$

Here  $s_{i,r}$  is the original service time of a class  $r$  job at node  $i$  in the original network. For nodes with  $s_{i_j,r} = 0$  the visit ratios must be zero. The rest of the visit ratios remain unchanged.

- (c) transform the class values into chain values (to differentiate between class values and chain values the chain values will be marked with \*).
- (d) compute the performance parameters of the transformed shadow model with the MVA. If the model parameters move back and forth then set

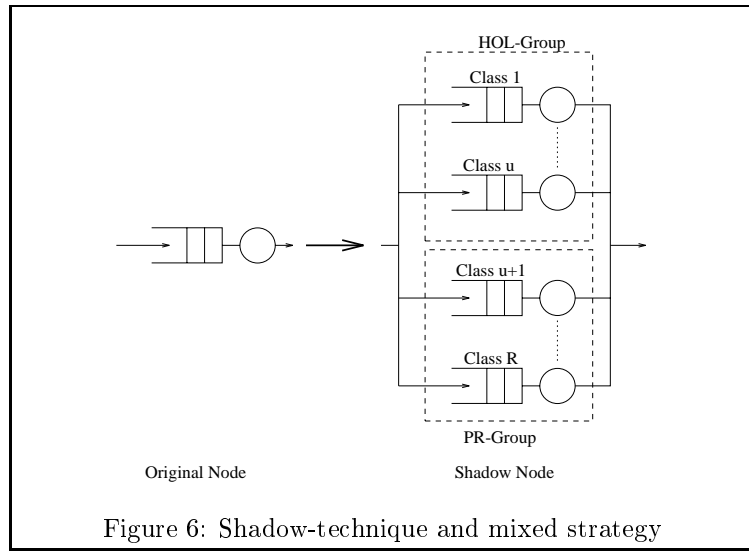
$$t_{i_j r} = \frac{t_{i_j r}^{old} + t_{i_j r}^{new}}{2}$$

and compute the performance measures with this new response time.

- (e) transform chain values into class values.
- If the  $\lambda_{i_j,r}^*$  (chain value) in two following steps differ less than  $\varepsilon$  then stop the iteration. Otherwise go back to step 4a.

### 2.5.3 Class Switching and Shadow Technique with Mixed Priority Strategy

The original shadow technique [Sevc77] can only be used for networks with a pure PR-strategy but in the models of a UNIX based multiprocessor operating system a mixed priority strategy is used. For this reason the original shadow technique has to be extended to deal with this mixed priority strategy. Therefore the suggested grouping strategy (see section 1.3.4) is used. In Fig. 6 this concept is shown. One has to note that an arriving PR-job does not affect the waiting time of a HOL-job in the situation of Fig. 6. On the other side a PR-job can always be preempted by an arriving HOL-job. This situation is taken into account



by modifying the correction factor  $\delta$  in the extended shadow technique [GBT94]

$$\psi_{i_j,r} = \begin{cases} \frac{\varrho(r)[1 - \varrho(r)] + \beta(r) \cdot \tilde{\varrho}(r)}{\varrho(r)[1 - \varrho(r)]^2 + \beta(r) \cdot \varrho_{i_r,r}} & \text{for } r = 2 \dots R \text{ and } j = r \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(r) = \begin{cases} \sum_{k=1, k \neq r}^u \frac{\rho_{i_j,r}}{\omega_{r,k}} & \text{if } r \in \text{HOL} \\ \sum_{k=1}^{r-1} \frac{\rho_{i_j,r}}{\omega_{r,k}} & \text{if } r \in \text{PR} \end{cases}$$

(For the meaning of the other variables see section 2.5.4) 5

#### 2.5.4 Class Switching and any Mixed Priority Strategy with the Shadow Technique

Until now we assumed that the classes  $1, \dots, u$  are pure HOL-classes and the classes  $u + 1, \dots, R$  are pure PR-classes. In the derived formulas we can see that in the HOL-case the summation in the correction factor is over all HOL-classes except the actual considered class. In the PR-case the summation is over all classes with higher priority. With this relation in mind it is possible to extend the formulas in such a way that every mixing of PR and HOL-classes is possible. Let  $\xi_r$  be the set of jobs that can not be preempted by a class  $r$  job. In addition it is assumed that  $r \notin \xi_r$ . For the computation of the correction factor this mixed priority strategy has the consequence that

- for a HOL class  $r$  the summation is not from 1 to  $u$  but over all priority classes  $s \in \xi_r$ .
- for a PR class  $r$  the summation is not from 1 to  $r - 1$  but over all priority classes  $s \in \xi_r$ .

For any mixed priority strategy we get the following formulas

$$s_{i_j,r} = \begin{cases} \frac{s_{i,r}}{1 - \sum_{s \in \xi_r} \frac{1}{\psi_{i_s,s}} \cdot \tilde{\rho}_{i_s,s}} & \text{if } j = r \\ 0 & \text{otherwise} \end{cases}$$

$$\psi_{i_j,r} = \begin{cases} \frac{\varrho(r)[1 - \varrho(r)] + \beta(r) \cdot \tilde{\varrho}(r)}{\varrho(r)[1 - \varrho(r)]^2 + \beta(r) \cdot \varrho_{i_r,r}} & \text{if } j = r \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(r) = \sum_{k \in \xi_r} \frac{\rho_{i_j,r}}{\omega_{r,k}}$$

$$\omega_{r,k} = \frac{s_{i,k}}{s_{i,r}}$$

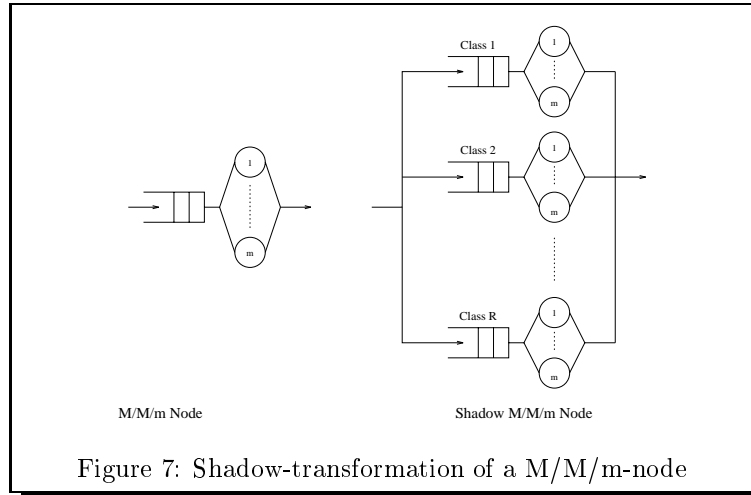
$$\varrho(r) = \sum_{k \in \xi_r} \rho_{i_k,k}$$

$$\tilde{\varrho}(r) = \sum_{k \in \{\xi_r \cup r\}} \rho_{i_k,k}$$

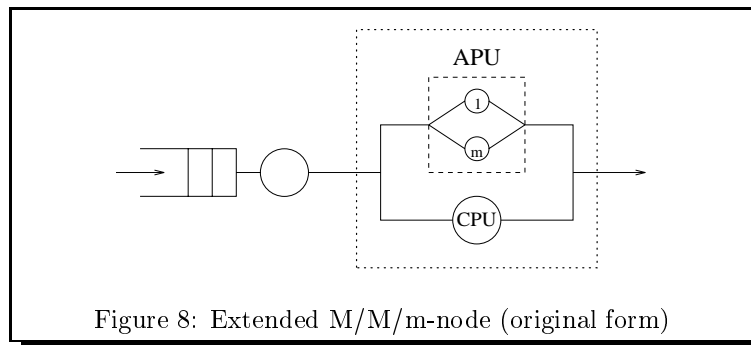
#### 2.5.5 Shadow Technique and Extended M/M/m-node

Now we introduce a new type of node - the so called extended M/M/m-node. This new node type plays a major role in the UNIX multiprocessor models. In the case of a M/M/m-node it is very easy to analyze this

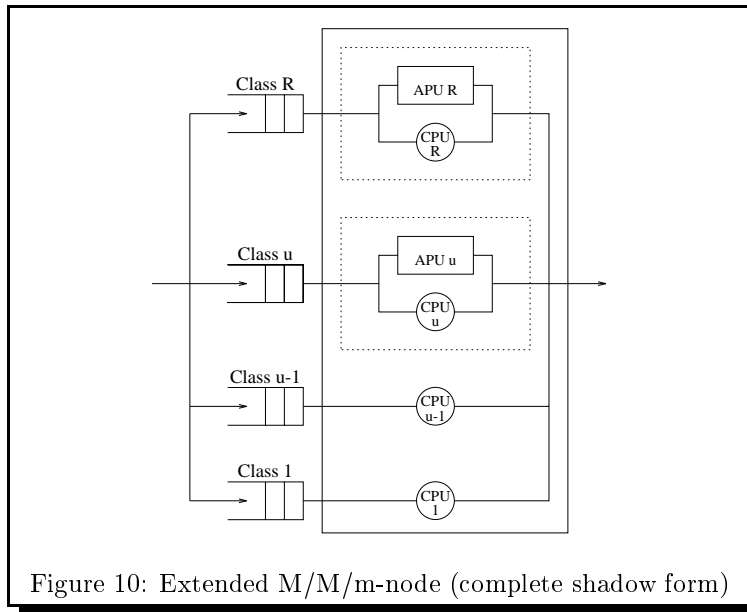
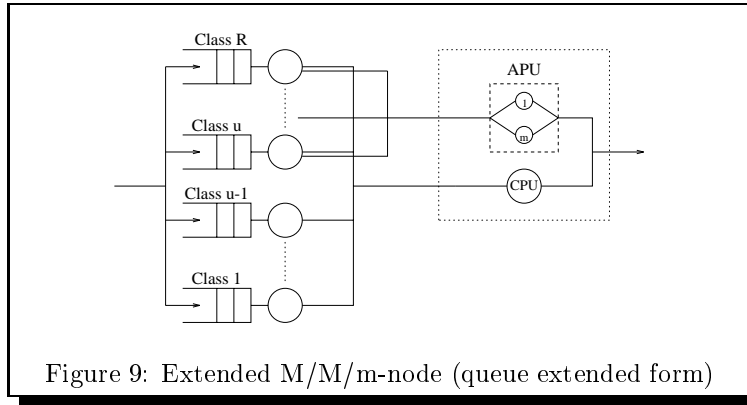
node. One only has to transform the M/M/m-node into its shadow form (see Fig. 7) and use the formulas for M/M/m-nodes when analyzing the node with the mean value analysis in each shadow iteration. The



difference between the M/M/m-node shown in Fig. 7 and the extended M/M/m-node in Fig. 8 is that in the extended M/M/m-node one has a node with  $R$  job classes which are ordered in priority order. The CPU can be entered by all job classes while the so called APU (associate processing unit) can only be entered by the job classes  $u \dots R$  ( $1 \leq u \leq R$ ). In the regular M/M/m-node each job of each class can use any free server. For a better overview a queue is introduced for each job class. The result is shown



in Fig. 9. If for example  $u = R = 3$  then the job classes 1, 2, 3 can enter the CPU while the APU can only be entered by job class 3. In the CPU class 3 jobs can always be preempted by class 1 and 2 jobs while in the APU class 3 jobs can not be preempted (in this situation). This means that the priority of the job class depends on which node it enters. In general preemption is also possible at the APU. To deal with this problem one has to use the idea of the shadow server again. The result is shown in Fig. 10. With this transformations done on the original network, it is possible to use the extended MVA for networks with priority strategies. Because of the fact that the shadow iterations are done separately on the CPU and APU, the service times of the different service stations become different and therefore we get an asymmetric node. Let  $\zeta$  be the set which contains the highest priority of the job that can enter the service station (CPU, APU). In a general case if one has a so called *generalized extended M/M/m node* [GBT94]  $\zeta$  contains normally more than two elements. In [BoGr94] it is shown that in the shadow algorithm only



the step for computing the mean service time  $s_{i_j,r}$  has to be changed in the following way :

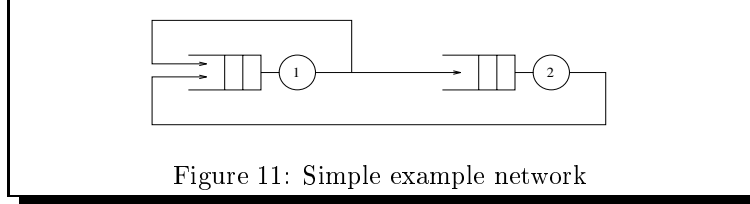
$$\forall c \in \zeta : s_{i_j,r} = \begin{cases} \frac{s_{ir}}{1 - \sum_{s \geq c, s \in \xi_r} \frac{1}{\psi_{i_s,s}} \cdot \tilde{\rho}_{i_s,s}} & \text{if } j = r \\ 0 & \text{otherwise} \end{cases}$$

All the results and techniques, shown here, can be applied to

- closed queueing networks (standard mean value analysis, see section 2.3.1).
- open queueing networks (transform the open queueing network into a closed queueing network, using the closing method, see section 2.3.2).
- mixed queueing networks (closing method for mixed queueing networks, see section 2.3.2).

### 3 Simple Example of a Network with Class Switching and Mixed Priority Strategy

To demonstrate how to use the new technique , we consider the following example: Node 1 is a M/M/1-



PRIORITY node and node 2 is a M/M/1-FCFS node. The network contains two job classes

**class 1:** PR

**class 2:** HOL

and class switching is allowed. The transition probabilities are given as follows:

$$\begin{aligned}
 p_{11,11} &= 0.5 & p_{11,12} &= 0.4 & p_{11,21} &= 0.1 \\
 p_{12,11} &= 0.5 & p_{12,12} &= 0.4 & p_{12,22} &= 0.1 \\
 p_{21,11} &= 1.0 & p_{22,11} &= 1.0 & &
 \end{aligned}$$

All other parameters are given as :

$$\begin{aligned}
 s_{11} &= 0.1 & s_{12} &= 0.1 \\
 s_{21} &= 1.0 & s_{22} &= 1.0 \\
 K &= 3 & \epsilon &= 0.001
 \end{aligned}$$

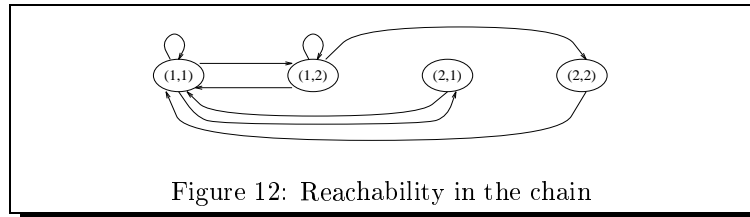
**Step 1:** compute the visit ratios  $e_{ir}$  in the network.

For this we use the formula

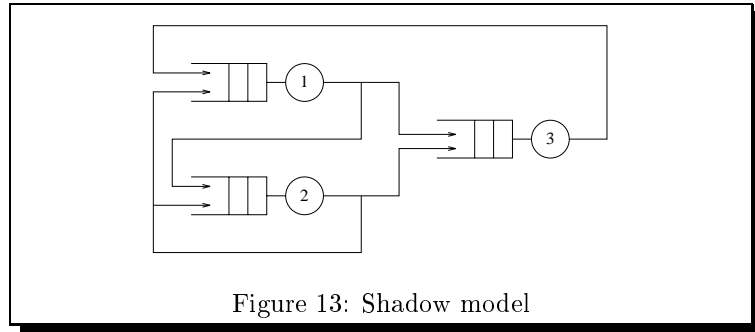
$$e_{ir} = \sum_{j=1}^2 \sum_{s=1}^2 e_{js} p_{js,ir}$$

$$\text{and get } e_{11} = 1.5 \quad e_{12} = 1.0 \quad e_{21} = 0.15 \quad e_{22} = 0.1$$

**Step 2:** determine the chains in the network and compute the number of jobs in each chain. In the following picture the "reachability graph" for the chains is shown and it can easily be seen, that one can reach each state from each other state (in a state  $(i, j)$   $i$  denotes the node number and  $j$  denotes the class number). Therefore we have only one chain in the network.



**Step 3:** transform the network into the shadow network



The corresponding parameters are

$$\begin{aligned}
 s_{11} &= 0.1 & e_{11} &= 1.5 & e_{31} &= 0.15 \\
 s_{22} &= 0.1 & e_{12} &= 0.0 & e_{32} &= 0.1 \\
 s_{31} &= 1.0 & e_{21} &= 0.0 & & \\
 s_{32} &= 1.0 & e_{22} &= 1.0 & & 
 \end{aligned}$$

**Step 4:** compute the visit ratios  $e_{iq}^*$  per chain and remember that we have only one chain.

$$e_{iq}^* = \frac{\sum_{r \in \pi_q} e_{ir}}{\sum_{r \in \pi_q} e_{1r}}$$

Then we get:

$$\begin{aligned}
 e_{11}^* &= \frac{e_{11} + e_{12}}{e_{11} + e_{12}} &= & 1 \\
 e_{21}^* &= \frac{e_{21} + e_{22}}{e_{11} + e_{12}} &= & 2/3 \\
 e_{31}^* &= \frac{e_{31} + e_{32}}{e_{11} + e_{12}} &= & 1/6
 \end{aligned}$$

**Step 5:** compute the scale factors  $\alpha_{ir}$ .

For this we use the equation

$$\alpha_{ir} = \frac{e_{ir}}{\sum_{s \in \pi_q} e_{is}}$$

and get

$$\begin{aligned}
 \alpha_{11} &= \frac{e_{11}}{e_{11} + e_{12}} &= & 1.0 \\
 \alpha_{12} &= \frac{e_{12}}{e_{11} + e_{12}} &= & 0 \\
 \alpha_{21} &= \frac{e_{21}}{e_{21} + e_{22}} &= & 0 \\
 \alpha_{22} &= \frac{e_{22}}{e_{21} + e_{22}} &= & 1.0 \\
 \alpha_{31} &= \frac{e_{31}}{e_{31} + e_{32}} &= & 0.6 \\
 \alpha_{32} &= \frac{e_{32}}{e_{31} + e_{32}} &= & 0.4
 \end{aligned}$$

**Step 6:** compute the service times per chain which are given by the equation

$$s_{iq}^* = \frac{1}{\mu_{iq}} = \sum_{r \in \pi_q} s_{ir} \cdot \alpha_{ir}$$

$$\begin{aligned} s_{11}^* &= s_{11} \cdot \alpha_{11} + s_{12} \cdot \alpha_{12} &= & 0.1 \\ s_{21}^* &= s_{21} \cdot \alpha_{21} + s_{22} \cdot \alpha_{22} &= & 0.1 \\ s_{31}^* &= s_{31} \cdot \alpha_{31} + s_{32} \cdot \alpha_{32} &= & 1.0 \end{aligned}$$

Now we can analyze the shadow network, given in Fig. 13 with the following parameters:

$$\begin{aligned} e_{11}^* &= 1.0 & e_{21}^* &= 2/3 & e_{31}^* &= 1/6 \\ s_{11}^* &= 0.1 & s_{21}^* &= 0.1 & s_{31}^* &= 1.0 \\ \alpha_{11} &= 1.0 & \alpha_{21} &= 0.0 & \alpha_{31} &= 0.6 \\ \alpha_{12} &= 0.0 & \alpha_{22} &= 1.0 & \alpha_{32} &= 0.4 \\ K &= 3 & N &= 3 \\ \xi_1 &= \{2\} & \xi_2 &= \{1\} \end{aligned}$$

**Step 7:** start the shadow iterations.

**1. Iteration:**

$$\begin{aligned} \rho_{11}^* &= 0.505 & \rho_{21}^* &= 0.337 & \rho_{31}^* &= 0.841 \\ \lambda_{11}^* &= 5.049 & \lambda_{21}^* &= 3.366 & \lambda_{31}^* &= 0.841 \\ s_{11}^{(1)*} &= 0.117 & s_{21}^{(1)*} &= 0.145 & s_{31}^{(1)*} &= 1.000 \end{aligned}$$

**2. Iteration:**

$$\begin{aligned} \rho_{11}^* &= 0.541 & \rho_{21}^* &= 0.447 & \rho_{31}^* &= 0.766 \\ \lambda_{11}^* &= 4.597 & \lambda_{21}^* &= 3.065 & \lambda_{31}^* &= 0.766 \\ s_{11}^{(2)*} &= 0.125 & s_{21}^{(2)*} &= 0.142 & s_{31}^{(2)*} &= 1.000 \end{aligned}$$

**3. Iteration:**

$$\begin{aligned} \rho_{11}^* &= 0.569 & \rho_{21}^* &= 0.430 & \rho_{31}^* &= 0.756 \\ \lambda_{11}^* &= 4.533 & \lambda_{21}^* &= 3.022 & \lambda_{31}^* &= 0.756 \\ s_{11}^{(3)*} &= 0.122 & s_{21}^{(3)*} &= 0.147 & s_{31}^{(3)*} &= 1.000 \end{aligned}$$

**4. Iteration:**

$$\begin{aligned} \rho_{11}^* &= 0.556 & \rho_{21}^* &= 0.447 & \rho_{31}^* &= 0.755 \\ \lambda_{11}^* &= 4.531 & \lambda_{21}^* &= 3.012 & \lambda_{31}^* &= 0.755 \\ s_{11}^{(4)*} &= 0.124 & s_{21}^{(4)*} &= 0.144 & s_{31}^{(4)*} &= 1.000 \end{aligned}$$

**5. Iteration:**

$$\begin{aligned} \rho_{11}^* &= 0.565 & \rho_{21}^* &= 0.437 & \rho_{31}^* &= 0.754 \\ \lambda_{11}^* &= 4.528 & \lambda_{21}^* &= 3.018 & \lambda_{31}^* &= 0.754 \end{aligned}$$

Now we stop the iteration because the difference between the  $\lambda^*$  in two successive iteration steps is smaller than  $\epsilon$ .

**Step 8:** retransform the chain values into class values

$$\begin{aligned}\lambda_{11} &= \alpha_{11} * \lambda_{11}^* = 4.528 \\ \lambda_{22} &= \alpha_{22} * \lambda_{21}^* = 3.018 \\ \lambda_{31} &= \alpha_{31} * \lambda_{31}^* = 0.452 \\ \lambda_{31} &= \alpha_{32} * \lambda_{31}^* = 0.301\end{aligned}$$

If we retransform the shadow network into the original network we get the following results:

$$\begin{aligned}\lambda_{11} &= 4.528 & \lambda_{12} &= 3.018 \\ \lambda_{21} &= 0.452 & \lambda_{22} &= 0.301\end{aligned}$$

**Step 9:** Validation of the results.

To validate this results we use the Markov-analysis tool MOSES [BGJZ94]. The exact results for this network are

$$\begin{aligned}\lambda_{11} &= 4.52 & \lambda_{12} &= 3.01 \\ \lambda_{21} &= 0.45 & \lambda_{22} &= 0.30\end{aligned}$$

## 4 Performance Evaluation of the Kernel of a UNIX Multiprocessor Operating System

The use of the presented techniques is now shown on the models of a UNIX based multiprocessor operating system [Jung91]. For these models the parameters were varied over a wide range of parameter values. We compare the results obtained from our approximation techniques with the one obtained from real system measurements. The UNIX models are described in the following section.

### 4.1 Model of a UNIX Job

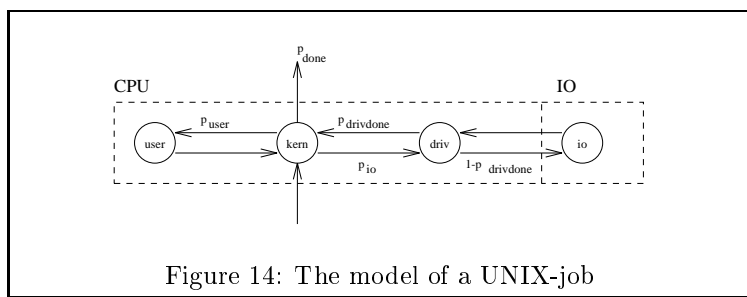


Figure 14: The model of a UNIX-job

In Fig. 14 the model of a UNIX job [Jung91] during its life cycle in the system is shown. A job always starts in the kernel context *kern* and changes into the user context *user* or driver context *driv* with probability  $p_{user}$  and  $p_{io}$  respectively. A job can leave the system only from the kernel context and is replaced by a new job (closed queueing network).

## 4.2 System assumptions

- the network is closed with  $K$  jobs in the system.
- the service times are exponentially distributed.
- the peripheral device system consists of three magnetic tapes and is modeled as a load dependent station.
- the system has three job classes *user*, *kern*, *driv* with the priority order  $driv > kern > user$ .
- jobs in the context *user* can always be preempted by *driv* and *kern* jobs. Other preemptions are not possible in the system.
- class switching is allowed.
- APU's (associated processing units) are used as coprocessors.
- the system is in steady state.
- the base model parameters are

$K = 10$	$p_{io} = 0.05$	$s_{user} = 0.25...20.0$
$\# \text{ Apu} = 2$	$p_{done} = 0.01/0.005$	$s_{kern} = 1.0$
	$p_{drivdone} = 0.4$	$s_{driv} = 0.5$

- a state in the transition probability matrix (network routing matrix) is described as a pair

(node number,class number)

node number 1 : CPU    node number 2 : IO  
class number 1 : driv    class number 2 : kern  
class number 3 : user

and the transitions between the states take place with the following probabilities

	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)
(1,1)	0	$p_{drivdone}$	0	$1 - p_{drivdone}$	0	0
(1,2)	$p_{io}$	$p_{done}$	$p_{user}$	0	0	0
(1,3)	0	1	0	0	0	0
(2,1)	1	0	0	0	0	0
(2,2)	0	0	0	0	0	0
(2,3)	0	0	0	0	0	0

The following difficulties occur when solving these models:

- different job classes have different priorities.
- a mixed priority strategy is used in the system so that some job classes can be preempted and others not.
- class switching is allowed.

### 4.3 The Monoprocessor Model

The model shown in Fig. 15 is called monoprocessor model. In an abstract way we have a closed queueing network with three job classes where class switching is allowed. At the CPU we have a mixture of a preemptive- and non preemptive service strategy. This means that *user* jobs can always be preempted by *kern* or *driv* jobs. Other preemptions are not possible in the system. If the CPU is free and *kern* and *driv* jobs are in the queue then *driv* jobs have higher priority.

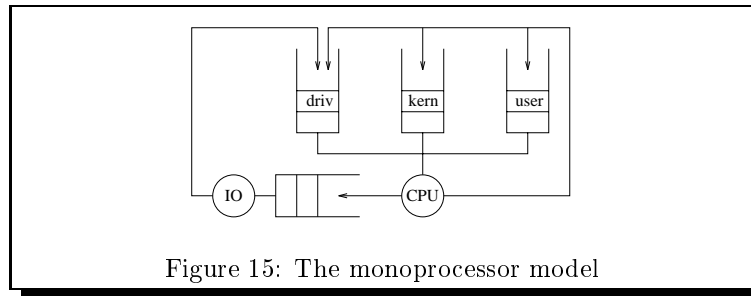


Figure 15: The monoprocessor model

### 4.4 The Master Slave Model

For this model, shown in Fig. 16, we have the same assumptions as in the monoprocessor model but now we have two so called APU's. These are additional coprocessors that work together with the main CPU. We assume that only jobs in the *user* context can be processed at the APU. If APU and CPU are both free and a user job is in the queue then it is decided randomly on which processor (CPU or APU) the job is served.

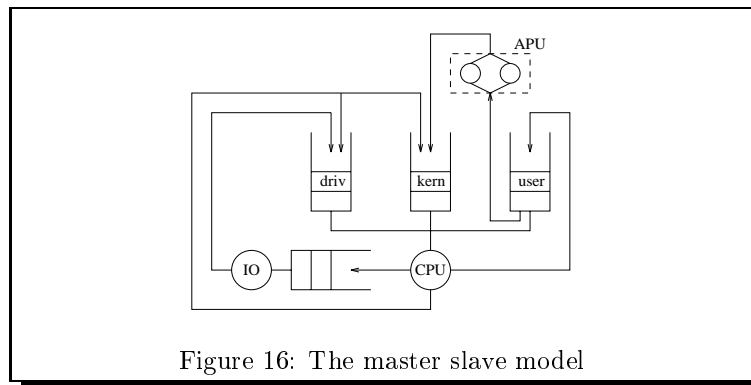
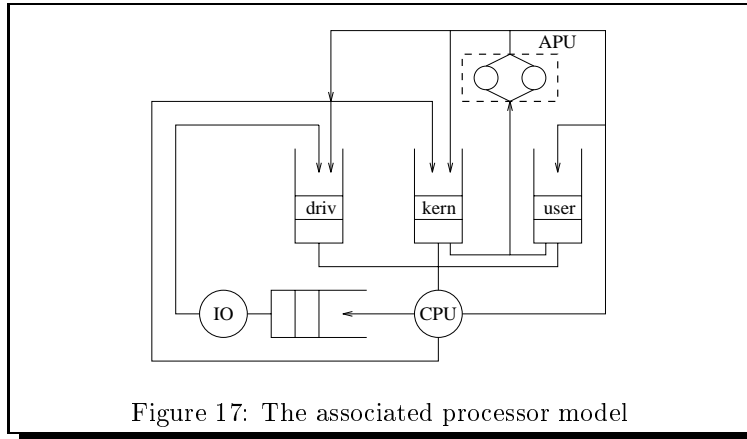


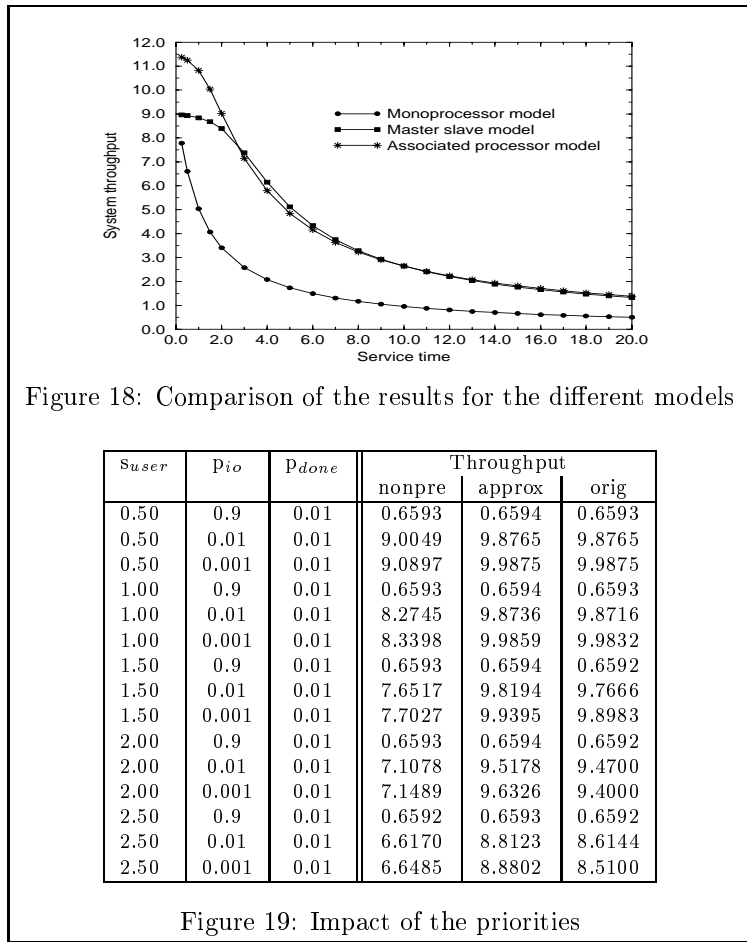
Figure 16: The master slave model

### 4.5 The Associated Processor Model

The assumptions are the same as in the master slave model but now kernel jobs are also allowed to be processed on the APU. Because of the fact that kernel jobs have higher priority than user jobs, preemption is also possible at the APU.



#### 4.6 Graphical Representation of the Results



As can be seen in Fig. 18 the the associated processor model has the highest throughput for jobs with short service times ( $s_{user} < 2$ ). For service times  $s_{user} \geq 2$  the associated processor model and the master slave model behave nearly identical. That means that for long running user jobs it is not worth to parallelize the

system kernel. In more detailed examinations it was found that it is not worth to add more than 2 APU's in the system because by adding another APU the system throughput increases only a little while the costs for adding an extra APU are very high. In Fig. 19 the impact of priorities to the system throughput is shown. It can be seen that for certain parameter values these impacts can not be neglected.

## 5 Conclusion

Our presented technique can be applied to open, closed and mixed queueing networks and therefore it is now possible to analyze any static mixed priority strategy with class switching and heterogeneous nodes with a standard analysis technique (like MVA). With this new technique computation time can be reduced considerably especially when the system has to be analyzed on a wide range of parameter values. In the case of the UNIX operating system the computation time for simulation was about two days, the computation time for Markovian analysis about 25 minutes and the computation time with our new technique was less than one second. Because of the very good results that we got from the UNIX-operating system models we plan to develop queueing models for other operating systems (like BS2000, real time systems, distributed systems). At the moment our new technique is restricted to systems with static priorities, but it is planned to extend the new technique to have also dynamic priorities and a mixture of static and dynamic priorities.

## References

- [BKLC84] Bryant R.M.; Krzesinski A.E.; Lakshmi M.S.; Chandy K.M.: *The MVA Priority Approximation*, ACM Transactions on Computer Systems, Vol. 2, No. 4, pp. 335 - 359, Nov. 1984.
- [BoGr94] Bolch G., Greiner S.: *Performance Evaluation of Operating Systems Using Approximate Analytical Methods*, Conference Proceedings of the ESS94-European Simulation Symposium 94.
- [Bolc89] Bolch G.: *Leistungsbewertung von Rechensystemen*, Teubner , 1989.
- [BrBa80] Bruell S.C. , Balbo G.: *Computational Algorithms for Closed Queueing Networks*, North Holland , 1980.
- [BGJ92] Bolch G. , Gaebell M. , Jung. H.: *Entwicklung und Validierung der Schliessmethode zur Analyse offener Warteschlangennetze*, Operations Research Proceedings, Aachen, Springer, 1992.
- [BGJZ94] Bolch G., Greiner S., Jung H., Zimmer R.: *The Markov Analyzer MOSES*, Technical Report TR-I4-10-94, Institute of Mathematical Machines and Data Processing IV University Erlangen-Nuernberg, June 1994.
- [GBT94] Greiner S., Bolch G., Trivedi K.: *Approximate Analytical Performance Evaluation Of A UNIX Based Multiprocessor Operating System* Technical Report TR-I4-94-27, Institute of Mathematical Machines and Data Processing IV University Erlangen-Nuernberg, June 1994.
- [Jung91] Jung H.: *Leistungsbewertung Unix basierter Betriebssysteme für Multiprozessoren mit globalem Speicher*, Dissertation at IMMD IV University Erlangen, 1991.
- [Kauf84] Kaufmann J.S.: *Approximation Methods for Networks of Queues with Priorities*, Performance Evaluation Vol 4 , pp. 183 - 198 , 1984.
- [Munt73] Muntz R.R.: *Poisson Departure Process and Queueing Networks*, Proc. of the 7th Annual Princeton Conf. on Information Sciences and Systems , Princeton University pp 435 - 440 March 1973.
- [ReLa80] Reiser M. , Lavenberg S.S.: *Mean-Value-Analysis of Closed Multichain Queueing Networks*, Journal of the ACM , Vol 27 , No. 2 , pp 313 - 322 . April 1980.
- [Sevc77] Sevcik K.C.: *Priority Scheduling Disciplines in Queueing Network Models of Computer Systems*, Proc. IFIP Congress , North Holland , pp 565 - 570 , 1977.