A tool support for automatic analysis based on the *tagged customer approach*

Levente Bodrog, Gábor Horváth, Sándor Rácz, Miklós Telek
Technical University of Budapest, 1521 Budapest, Hungary
e-mail: {bodrog,hgabor,raczs,telek}@webspn.hit.bme.hu

Abstract

The analysis of performance models based on the tagged customer approach *is composed by two main steps. The first one is the stationary analysis of a Markov chain and the second one is the transient analysis of another Markov (reward) model whose initial distribution is derived from the result of the first step. To the best of our knowledge none of the existing performance analysis tools allow the automatic evaluation of these performance models.*

This paper presents an extension of the MRMSolve tool for automatic execution of the two steps procedure. The theoretical background of this extension is well established and the implementation is built mainly on the existing Markov model analysis functions of MRMSolve, hence the contribution lies in the availability of the automatic analysis tool.

To demonstrate the modeling abilities and the practical importance of the new tool we present the MRMSolve models of [7] and [8]. Based on these MRMSolve models one can automatically generate the performance measures presented in [7] and [8].
Keywords: Markov reward models, Numerical analysis, Tagged customer approach.

1. Introduction

The application of discrete state continuous time Markov chains for performance analysis of computer and communication systems has a very long history. At the beginning the manual description of the Markov model of the system behaviour and the associated manual symbolic model analysis was the typical analysis approach. The applied modeling and analysis techniques improved a lot since that time.

The widespread use of powerful digital computers allowed the numerical evaluation of larger Markov models. The error-free manual generation of Markov models of such size is already too complex for humans. New model description methodologies were developed to support the automatic generation of system models. Depending on the typical application area and the analysis goals a large number of different model description languages were introduced (stochastic Petri net [14], stochastic process algebra [10], queueing network [23], stochastic activity network [21], etc.) and automatic software tools (GreatSPN [4], UltraSan [6], SHARPE [20], PEPSY [1], MOSEL [1], QNAT [11], etc.) were developed to

- interpret the model description,
- automatically generate the low level model,
- calculate the required performance parameters and
- translate them back to the model description context.

A convenient way to define complex performance measures in a wide range of model description languages is the use of reward variables. The value of a reward variable depends on the stochastic process through a simple function. The most common cases are the rate reward accumulation and the impulse reward accumulation. During the sojourn in a system state rate reward is accumulated at a constant rate characterized by the system performance in the particular state. Impulse reward is accumulated at state transitions of the stochastic process in an initial and final state dependent manner. The reward accumulated according to this simple function can represent almost all practically important performance measures. E.g., it can represent positive (negative) quantities like served (rejected) customers in a queueing system, or performed work (accumulated stress).

The majority of the performance analysis tools can perform both steady state and transient analysis. An essential difference between the transient and the steady state analysis is that (in case of ergodic systems) the steady state measures are independent of the initial system state (it does not need to be defined), while the transient measures depend on the initial distribution of the system (hence it has to be defined for transient analysis). In most of the practically important cases the transient analysis starts from a deterministic initial distribution, i.e., the system starts from a particular system state (e.g., empty queue, perfect working condition) with probability 1.

The definition of deterministic initial state distribution is quite easy in automatic performance analysis tools. Only a single initial state (in Markov chain based models) or an initial marking (in Petri net based models) needs to be defined.

The majority of the performance tools are restricted to the use of deterministic initial distribution or requires the listing of the potential initial states and the associated initial probabilities. Exceptions are, e.g., SPNP [5], METFAC2 [3]. This limitation practically inhibits the definition of initial distributions with a large number of potential initial states.

One of the practically important exceptions to deterministic initial distribution comes from the use of tagged customer approach. In this two steps analysis method the initial condition of a *tagged customer* is characterized first and the transient performance parameters of the tagged customer are analyzed next. The initial distribution of the second step contains all system states which are reachable at the arrival of the tagged customer. For larger systems it is infeasible to calculate the initial distribution without automatic tool support. This paper presents a performance analysis tool for the automatic analysis based on the tagged customer approach. Another practically important case with non-deterministic initial distribution is the consecutive transient analysis at time $t_1, t_2, \dots$ such that the initial distribution used for the analysis of the interval (t_i, t_{i+1}) is the transient distribution at time t_i.

The rest of the paper is organized as follows. Section 2 introduces the tagged customer approach. Section 3 presents the proposed description language for automatic analysis of this approach and Section 4 discusses the features of the MRMSolve tool that implements the automatic analysis. Two application examples from the field of telecommunication demonstrate the practical use and the automatic analysis of the tagged customer approach. The first one evaluates the throughput of elastic streams in a multi-service environment in Section 5 and the second one analyzes the performance of peer-to-peer (P2P) file transfer in Section 6. Some concluding remarks are given in Section 7.

2. Tagged customer approach

In the middle of the twentieth century the detailed analysis of basic queueing systems resulted in several effective intuitive explanations of their behaviour such as the concept of flow balance, the role of utilization, the PASTA property and the tagged customer approach [1]. According to the tagged customer approach, an ergodic queueing system can be analyzed by studying a single customer, referred to as tagged customer, that enters the system according to an initial distribution representing the state of the system just after a customer arrival. Indeed, it is just an application of the law of total probability. The required performance measure is calculated conditioned on the number of customer in the system at customer arrival. The introduction of Markov arrival processes extended the applicability of this approach to more complex arrival processes [12], where the condition is the number of customer in the system and the "phase" of the modulating Markov chain.

This concept was successfully applied for the analysis of queueing systems and networks, e.g., in [13, 16, 15]. In the mean time the evolution of stochastic modeling and automatic performance analysis of complex systems allow for the description and the evaluation of more complex systems with automatic performance analysis tools. This evolution resulted in the generalization of the tagged customer approach as well:

- General arrival pattern:
  The PASTA property describes the distribution of system state at an arrival instant of a homogeneous Poisson process. Unfortunately, the arrival processes of practically interesting systems are often non-Poisson and the distribution of the system state seen by a customer of a state dependent arrival processes is more complex to evaluate. State dependent arrival patterns characterize several practical systems. State dependency might be a result of finite population, load control, service differentiation, etc. Another recently typical reason for state dependent arrival processes is the widespread use of Markovian arrival process (MAP) models [12].

- Conditional measures of the tagged customer:
  Several performance measures of customers behaviour can be evaluated without using the tagged customer approach (e.g., the mean waiting time), but there are complex measures that cannot be extracted from the Markov chain describing the overall system behaviour. An example of this kind of complex measures is the conditional analysis of customers behaviour according to the following scenario. Consider a system with customer arrivals and departures. The performance measure of interest is the reward accumulated by a given customer during the interval $(0, t)$ supposing that the customer is in the system at time t. A convenient way to evaluate this kind of performance measures is to apply the tagged customer approach such that the behaviour of the tagged customer is analyzed over the $(0, t)$ interval and the departure of the tagged customer is prohibited. This approach was applied for the analysis of elastic streams in a multi-service environment in [2, 7] as it is discussed in the first example.

The initial distribution of the tagged customer is commonly given as the stationary distribution of another Markov chain describing the system without the tagged customer. In more complex cases (e.g., in case of state dependent arrival rates) the initial distribution can be defined as a stationary (reward) measure of the Markov chain describing the system without the tagged customer.

The sizes of the Markov chains describing the system with and without the tagged customer are usually different. Due to this difference a mapping of the stationary measures of the first chain to the states of the second one is required. This mapping is very simple in the classical queueing system examples, but it might be rather general in complex models (e.g., in case of multi-class queueing systems). The flexibility of this mapping step is indicated in the mapping sections of the examples presented below.

3. Model description language of MRMSolve

The model description language of MRMSolve is designed to be a compact and flexible formalism for easy description of finite, very large and well structured systems. For the details of this model description language we refer the reader to [18].

The model description consists of sections (see Tables 1, 2). The model constants are defined in the `Const` section, hence a model parameter (like the arrival rate) can be modified easily without modifying the descriptions of matrices and vectors. Variables are defined in the `Var` section. The difference between constants and variables is that only scalars are allowed on the right hand side of constant declarations, while expressions are also allowed on the right hand side of variable declarations. Complicated expressions (functions) can be defined in the `Code` section.

State space variables are given in `State` section. Using more than one state space variable the user can construct models with multi dimensional structure. The declaration of the state variables contains their ranges as well. The rules in the `Condition` section determine the valid combinations of state variables.

The generator matrix and the vectors corresponding to the model (like the reward rate vector and the initial probability vector) are described by the rules in `Vector` and `Matrix` sections. The vector rule syntax is:

```
    Condition : State = value;
```

Here `State` identifies an element of the vector, and `value` is assigned to this vector element (e.g., reward rate in `State`). The optional `Condition` is a boolean expression that can restrict the scope of the rule. The matrix rule syntax is the following:

```
    Condition : State1 -> State2 = value;
```

where `State1->State2` identifies the matrix element, and `value` is assigned to this matrix element. During the low level model construction phase the interpreter software generates all possible states, and fills in the vectors and matrices according to the rules with enabled condition.

To support the analysis of models based on the tagged customer approach, we introduced an extension for the automatic definition of the initial probability vector. If the `embedded` keyword is appended to the `Vector P0` line the initial probability vector is computed from the steady state probabilities of another Markov chain (Markov chain without the tagged customer). In this case, the mapping of the steady state probabilities of the Markov chain without the tagged customer and the initial probabilities of the Markov (reward) model with the tagged customer is provided in the `Mapping` section. The mapping rules have the following syntax:

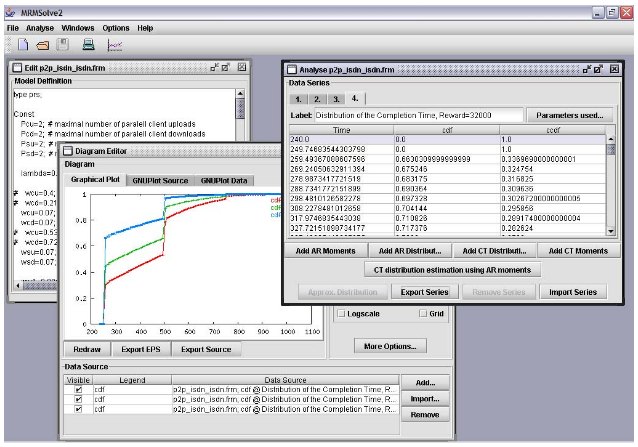
```
Condition :
State_init -{weight}-> State_rewardmodel;
```

At the execution of this line, the initial probability of `State_rewardmodel` of the Markov chain with the tagged customer is incremented by the product of the stationary probability of `State_init` of the Markov chain without the tagged customer and `weight` if `Condition` is true.

4. The MRMSolve reward model tool

With the continuous development of the MRMSolve package, we intend to collect, implement and integrate in a common environment as much computational functionality of Markov reward models as possible. MRMSolve consists of a set of command line tools and a graphical user interface. The command line tools take the model description as input and provide different performance measures. The command line tools implement the computationally intensive procedures. These tools are written in C++ to speed up the computation.

The graphical user interface (Figure 1) is written in Java. It allows to edit, check and visualize models, to investigate the effect of model parameters on the reward measure of interest.



```
type prs;

Const
  Pcu=2;  # maximal number of paralell client uploads
  Pcd=2;  # maximal number of paralell client downloads
  Psu=2;  # r
  Psd=2;  # r

  lambda=0.

#  wcu=0.4;
#  wcd=0.21
  wcu=0.07;
  wcd=0.07;
#  wcu=0.53
#  wcd=0.72
  wsu=0.07;
  wsd=0.07;
```

Label: Distribution of the Completion Time, Reward=32000

Time	cdf	ccdf
240.0	0.0	1.0
249.74683544303798	0.0	1.0
259.49367088607596	0.6630309999999999	0.3369690000000001
269.24050632911394	0.675246	0.324754
278.9873417721519	0.683175	0.316825
288.7341772151899	0.690364	0.309636
298.4810126582278	0.697328	0.30267200000000005
308.2278481012658	0.704144	0.295856
317.9746835443038	0.710826	0.28917400000000004
327.72151898734177	0.717376	0.282624

Visible	Legend	Data Source
✓	cdf	p2p_isdn_isdn.frm; cdf @ Distribution of the Completion Time, R...
✓	cdf	p2p_isdn_isdn.frm; cdf @ Distribution of the Completion Time, R...
✓	cdf	p2p_isdn_isdn.frm; cdf @ Distribution of the Completion Time, R...

Figure 1. MRMSolve 2.0 screenshot

The graphical user interface offers the following model verification methods:

- syntax check of a selected model,
- show state space (list the states),
- draw state space (using the tools of the Graph Visualization Project (of the ATT Research Lab.) [19]).

The graphical interface provides flexible analysis possibilities for the reward models. The following measures can be evaluated:

- the distribution of the accumulated reward and of the completion time (if supported for the model),
- the moments of the accumulated reward and of the completion time (if supported for the model),
- moment based estimation for the accumulated reward and for the completion time,
- moment based estimation for the completion time from the moments of the accumulated reward.

Series of runs can be generated automatically, where the moments are computed as the function of a model parameter appearing in the 'Const' section of the model description.

With the built-in diagram editor it is possible to draw and compare all available analysis results of all opened models. The tool calls 'gnuplot' to create the required plots. The user can set and change the most important plot options, like the range of the x and y axis, the style of lines, the linear and logarithmic scaling of the axes, or the position of the legends. The graphical results can be exported to encapsulated postscript files. The standard C++ and Java implementation makes MRMSolve platform independent. We have installed it on Unix and Windows.

The new functionality of MRMSolve, the support for the tagged customer approach, is implemented using some existing elements of MRMSolve (like the stationary analysis command line tool) and a new feature for mapping the stationary probabilities to the initial probabilities of another Markov chain.

5. Analysis of elastic streams in a multi-service environment

Multi-service systems supporting fixed bandwidth and elastic flows arise in various telecommunication scenarios. The analysis of these models appeared in several research papers [2, 7]. We evaluate a multi-rate system supporting peak allocated stream and elastic flows based on [7] using the MRMSolve tool. This model allows us to evaluate systems where non-adaptive stream traffic with strict QoS requirement, like voice traffic, and rate-adaptive elastic traffic, like packet switched traffic, share a common resource, e.g., a transmission link. For stream traffic the system guarantees the required bandwidth to fulfill the QoS requirement using priority over elastic traffic. The elastic flows use some kind of flow control mechanisms (like TCP) to adapt their actual bandwidth to the bandwidth left available by the stream traffic. Figure 2 illustrates the system behaviour. During the lifetime of stream flows their required bandwidths are provided by the system. However, elastic flows adapt their bandwidth demands to the available bandwidth. As Figure 2 shows, if there is enough bandwidth for the elastic flows they receive their peak bandwidth. If there is not enough capacity for providing the peak bandwidth for each elastic flow, then the elastic flows reduce their bandwidth uniformly.

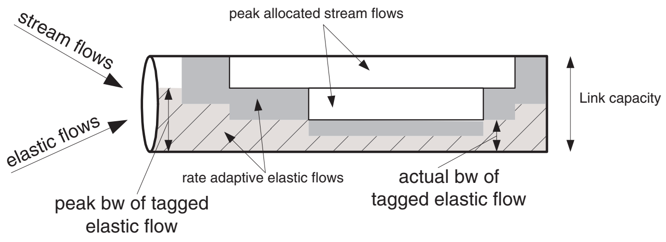

stream flows
elastic flows
peak allocated stream flows
Link capacity
rate adaptive elastic flows
actual bw of tagged elastic flow
peak bw of tagged elastic flow

Figure 2. Bandwidth sharing in the multi-service environment

The performance measure of interest is the throughput of elastic flows. The throughput of an elastic flow is the transmitted data divided by the length of the transmission. For example, when an elastic flow transmits $1,000$ Kbit during 10 sec, then its throughput is 100 Kbps. Formally, the throughput of an elastic flow of length t is the integral of its instantaneous bandwidth function:

$$\text{Throughput}(t) = \frac{1}{t}\int_0^t r(t) b_{el} dt$$

where b_{el} is the peak bandwidth demand of the elastic flow and $r(t)$ denotes the compression of elastic flow at time t, such that, if $r(t) = 1$ then elastic flows receive their peak bandwidths at time t. Throughput(t) is a random variable and we use the tagged customer approach to study its distribution. We define a Markov reward model whose accumulated reward is identical with Throughput(t).

5.1. Model description

First we introduce the input parameters of the Markov reward model of the studied system and then precisely formalize the system behaviour.

The link of capacity C serves stream and elastic flows that are characterized as follows:

- The *stream* flows are characterized by their peak bandwidth demand b_{st} (`b_st`), flow arrival rate λ_{st} (`l_st`) and the mean length of flows t_{st} (`t_st`).
- The *elastic* flows are characterized by their peak bandwidth demand b_{el} (`b_el`), minimal bandwidth requirement $b^{el}_{min} = r_{min} b_{el}$, flow arrival rate λ_{el} (`l_el`), and by the mean of exponentially distributed flow time when the peak bandwidth is available, t_{el} (`t_el`). r_{min} (`rmin`) is the maximal bandwidth reduction of elastic flows. The elastic flows experience the ideal condition when the peak demand is always available throughout the whole time of flows. If the elastic flow receives less bandwidth than its peak bandwidth demand then its length will be longer.

All arrival processes are independent Poisson processes and both the service times of stream flows and the carried load of elastic flows are exponentially distributed. The resource sharing policy is as follows:

- The stream flows always get their peak bandwidths. The elastic flows are only allowed to use the capacity left by the stream flows.
- The sum of the peak bandwidths of ongoing stream flows and the minimal bandwidth of ongoing elastic flows has to be less than or equal to the link capacity.
- If the sum of the peak bandwidth requirements of all ongoing elastic flows does not exceed the capacity left by the ongoing stream flows each flow gets its required peak bandwidth.
- Otherwise, when the link is serving at its total capacity, the remaining bandwidth is shared between individual elastic flows equally.

To describe the time evolution of a tagged elastic flow, we build up a Markov reward model. The reward variable describes the bandwidth received by the tagged elastic flow. Table 1 provides the model description used in the MRMSolve tool.

A two-dimensional Markov chain describes the behaviour of the model while the tagged elastic flow is in the system. The pair of numbers of ongoing stream flows (n_{st}) and ongoing elastic flows (n_{el}) identify a state of the model:

$$\mathcal{S} = \{(n_{st}, n_{el}) : n_{st} b_{st} + n_{el} b^{min}_{el} \leq C \ \text{ and } \ n_{el} \geq 1\}$$

The first condition guarantees the capacity limitation and the second condition guarantees that the tagged elastic flow is always present in the model. The first `State` and `Condition` section formally describes the structure of the state space.

In state (n_{st}, n_{el}) the bandwidth share of an elastic flow (including the tagged elastic flow as well) can be calculated as follows:

$$b(n_{st}, n_{el}) = \min\left(b_{el}, \frac{C - n_{st} b_{st}}{n_{el}}\right).$$

The first `Code` section defines this relation.

The state transition intensities are described in the `Matrix Q` section. There are four types of transitions:

1. arrival of a stream flow,
2. arrival of an elastic flow,
3. departure of a stream flow and
4. departure of an elastic flow excluding the tagged flow.

The reward assigned to states are described in the `Vector R` section. The reward variable describes the bandwidth received by the tagged elastic flow.

To determine the initial distribution of the Markov reward model we build up another Markov chain. This Markov chain describes the system behaviour without the tagged elastic flow. The state space of this Markov chain is

$$\mathcal{S}' = \{(n_{st}, n_{el}) : n_{st} b_{st} + n_{el} b^{min}_{el} \leq C\}$$

where we do not exclude states where the number of ongoing elastic flow is zero. Right after the tagged customer arrival the number of elastic flows increases by one. This translates to the following mapping of the state-spaces:

$$\mathcal{S}' \to \mathcal{S} : (n_{st}, n_{el}) - 1 \to (n_{st}, n_{el} + 1)$$

Thus, we use the steady-state probability of state (n_{st}, n_{el}) in the first Markov chain as the initial probability of state $(n_{st}, n_{el} + 1)$ in the second Markov reward model. The section `Vector P0 embedded` specifies the second Markov chain and the section `Mapping` the mapping of the state-spaces.

5.2. Analysis

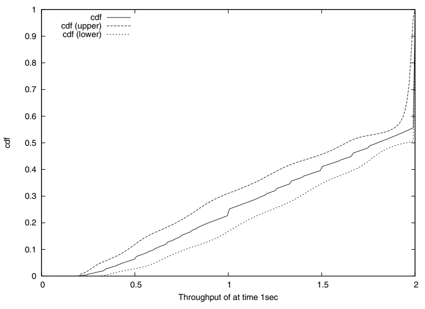
The model parameters are provided in the first `Const` paragraph in Table 1. Figure 3a provides the distribution of the throughput at time $t = 1sec$, i.e., the amount of data transmitted in $1sec$ divided by $1sec$. MRMSolve also provides upper and lower limits for this measure. The bounds are derived using the first 21 moments of the measure. These bounds are useful when the computational complexity of direct distribution calculation is infeasible.

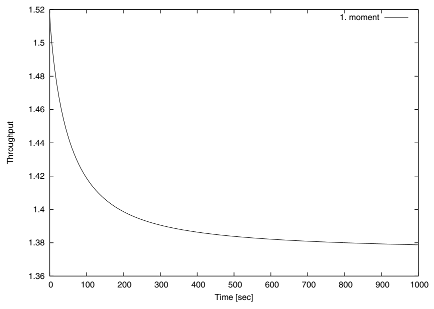
Figure 3b shows how the mean value of the throughput depends on the time. As we can see the throughput is smaller for larger t. The reason of this time dependent behaviour of the throughput is that the Markov reward model does not start from its steady-state distribution.

```
# Markov reward model (MRM) that describes the time evolution
# of bandwidth received by the tagged elastic customer
type prs;
Const # System description (Input parameters)
    C =         10;  # Link capacity [BU]
    b_st =      1;   # Bandwidth demand of stream flows [BU]
    b_el =      2;   # Peak bandwidth of elastic flows [BU]
    rho_st =    5;   # Offered load of stream flows
    rho_el =    1.5; # Offered load of elastic flows
    log_t_st =  1;   # Log of mean length of stream flows [TU]
    log_t_el =  1;   # Log of (ideal) mean length of elastic flows [TU]
    rmin =      0.1; # Maximal allowed compression of elastic flows
Var # Derived parameters
    r = 0;                        # Actual compression of elastic flows
    mu_st = 1.0/pow(10,log_t_st); # Departure rate of stream flows
    mu_el = 1.0/pow(10,log_t_el); # Ideal departure rate of stream flows
    l_st  = rho_st*mu_st;         # Arrival rate of stream flows
    l_el  = rho_el*mu_el;         # Arrival rate of elastic flows
State # State variables (a two-dimensional state-space)
    n_st : 0 To floor(C/b_st);        # Number of ongoing stream flows
    n_el : 0 To floor(C/(rmin*b_el)); # Number of ongoing elastic flows
Condition  # Valid combination of state variables
    n_st*b_st+n_el*b_el*rmin <= C;        # Capacity limitation
    n_el > 0;                             # Tagged elastic flow is always present in the system
Code # Calculating the compression of elastic flows in the state (n_st,n_el)
    r = if(n_el>0,(C-n_st*b_st)/(n_el*b_el),1); # Actual compression of ongoing elastic flows
    r = min(1,r);                               # Compression is always less than or equal to 1
Matrix Q  # State transitions
    [n_st,n_el]->[n_st+1,n_el]   = l_st;             # Arrival of a new stream flow
    [n_st,n_el]->[n_st,n_el+1]   = l_el;             # Arrival of a new elastic flow
    [n_st,n_el]->[n_st-1,n_el]   = n_st*mu_st;       # Departure of an ongoing stream flow
    [n_st,n_el]->[n_st,n_el-1]   = r*(n_el-1)*mu_el; # Departure of an ongoing elastic flow
Vector R # Bandwidth received by the tagged elastic customer
    [n_st,n_el] = b_el*r;

# Markov chain (MC) whose steady state solution provides
# initial distribution of MRM defined above
Vector P0 embedded
    Const
     C =         10;  # Link capacity [BU]
     b_st =      1;   # Bandwidth demand of stream flows [BU]
     b_el =      2;   # Peak bandwidth of elastic flows [BU]
     rho_st =    5;   # Offered load of stream flows
     rho_el =    1.5; # Offered load of elastic flows
     log_t_st =  1;   # Log of mean length of stream flows [TU]
     log_t_el =  1;   # Log of (ideal) mean length of elastic flows [TU]
     rmin =      0.1; # Maximal allowed compression of elastic flows
    Var
     r = 0;                        # Actual compression of elastic flows
     mu_st = 1.0/pow(10,log_t_st); # Departure rate of stream flows
     mu_el = 1.0/pow(10,log_t_el); # Ideal departure rate of stream flows
     l_st  = rho_st*mu_st;         # Arrival rate of stream flows
     l_el  = rho_el*mu_el;         # Arrival rate of elastic flows
    State
     n_st : 0 To floor(C/b_st);           # Number of ongoing stream flows
     n_el : 0 To floor(C/(rmin*b_el));    # Number of ongoing elastic flows
    Condition
     n_st*b_st+n_el*b_el*rmin <= C;       # Capacity limitation
    Code
     r = if(n_el>0,(C-n_st*b_st)/(n_el*b_el),1);  # Actual compression of ongoing elastic flows
     r = min(1,r);                                # Compression is always less than or equal to 1
    P0generator
     [n_st,n_el]->[n_st+1,n_el]   = l_st;         # Arrival of a new stream flow
     [n_st,n_el]->[n_st,n_el+1]   = l_el;         # Arrival of a new elastic flow
     [n_st,n_el]->[n_st-1,n_el]   = n_st*mu_st;   # Departure of an ongoing stream flow
     [n_st,n_el]->[n_st,n_el-1]   = r*n_el*mu_el; # Departure of an ongoing elastic flow
    # Transforming the steady-state distribution of MC to initial distribution of MRM
    Mapping
     [n_st;n_el]-{1}->[n_st;n_el+1];
```

Table 1. Model description in the MRMSolve


a) Throughput distribution at time $t = 1sec$


b) Mean throughput vs time

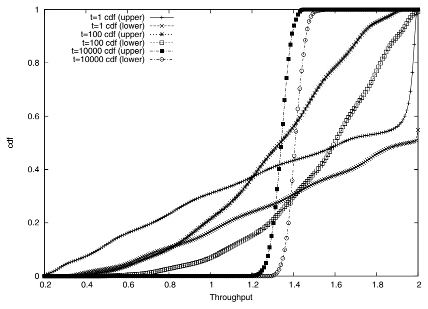

c) Upper/lower bounds of the distribution

Figure 3. Throughput of elastic streams in the multiservice environment

Finally, Figure 3c shows the bounds of the throughput distribution as a function of t. According to our expectation, for larger t the throughput will be more deterministic, since it gets closer and closer to the stationary throughput.

6. Peer-to-peer example

This example demonstrates the application of the tagged customer analysis feature of MRMSolve for the performance analysis of P2P file sharing. We take the model of P2P download from [8, 9].

These papers assume that the bottleneck in file transfer is located either at the client side, or at the server side, but not in the network backbone. The client and server relation refers to the direction of the tagged file transfer. The available bandwidth during the tagged file transfer depends on how many downloads and uploads proceed in parallel at the client and the server side. The number of parallel file transfers is changing during the tagged transfer, new request may arrive and ongoing requests may finish. The system behaviour is modeled by a Markov chain, that modulates the available bandwidth during the tagged file transfer.

The common performance measure of interest in P2P networks is the distribution of the transfer time. This performance measure was obtained by a fluid stochastic Petri net in [8]. However, the bandwidth of the tagged flow is a positive quantity, hence this model can be analyzed by MRMSolve, such that the *completion time* represents the file transfer time.

6.1. Model Description

Both the client and server side allow only a given number of file downloads and uploads in parallel. If this limit is not reached yet, new download or upload requests can arrive according to a Poisson process with parameter λ. The completion time of a request is approximated by a 2-phase hyper-exponential distribution.

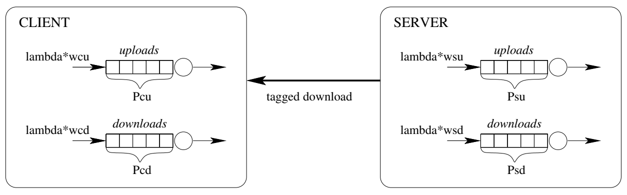


Figure 4. Model of the peers

The model description is presented by Table 2. The meaning of the constants is as follows:

- `Pcu`, `Pcd`: the maximal number of parallel upload and download requests at the client side.
- `Psu`, `Psd`: the maximal number of parallel upload and download requests at the server side.
- `lambda`: the arrival rate of file transfer requests.
- `wcd` (`wcu`): the portion of download (upload) request to the client having a given network connection. This parameter ensures that the arrival rate to the client (popularity) is depending on its network connection.
- `wsd` (`wsu`): the portion of download (upload) request to the server having a given network connection.
- `mu1`, `mu2`, `alpha1`, `alpha2`: the parameters of the hyper-exponential completion time of the background transfers. `mu1`, `m2` are intensities; `alpha1` and `alpha2` are the initial probabilities of the distribution (`alpha2`$=1-$`alpha1`).
- `modem_bw`, `isdn_bw`, `dsl_upld`, `dsl_dwld`: the total bandwidth of the given network technology.

The state space is the product of the state space of the four queues of the model (Figure 4), thus the state variables in the model are:

- `a1` (`a2`): the number of upload request being in phase 1 (2) in the hyper-exponential service time at the client. As indicated in the `Condition` section, the number of background uploads (`a1+a2`) plus the tagged transfer must be less than the maximal number of download requests allowed (`Pcu`).

- `b1` (`b2`): the number of download request being in phase 1 (2) in the hyper-exponential service time at the client. In the `Condition` section, `b1+b2<=Pcu` ensures that the number of background downloads is bounded by `Pcd`.
- `c1`, `c2`, `d1`, `d2`: their meaning is the same as `a1`, `a2`, `b1`, `b2`, but at the server side.

The section corresponding to the generator matrix (`Matrix Q`) describes the behaviour of the 4 independent M/H2/Pcd/Pcd (Pcu,Psu,Psd, accordingly) queues.

The `Vector R` section defines the bandwidth available for the tagged transfer. The "+1" in the expression corresponds to the tagged customer.

In `Vector P0` the `embedded` keyword means that the initial state probabilities of the reward model are taken from the steady state distribution of a Markov chain that is embedded into the model description.

The `Mapping` section defines how the steady state distribution of the embedded model is mapped to the initial probability vector of the reward model. In this example we compute the steady state distribution of the background process of the file transfer, therefore the generator of the embedded model is very similar to the one of the analyzed model. The difference is that the tagged customer is a normal customer in the embedded model. The mapping of the probabilities is such that it does not map probabilities to states where the tagged customer cannot arrive (thus, when the download queue of the client or the upload queue of the server is full).

6.2. Analysis

There are two ways in MRMSolve to evaluate the distribution of the completion time. There is a direct distribution solver, with 3 selectable algorithms, and there is a moment based estimation method [22]. Among the 3 direct solution algorithm only the one by Sericola [17] gave numerically stable results for this problem. It was slow (1-2 minutes per point) compared to the moment based estimation (1-2 seconds per point). In this section – where not indicated – all the plots are generated using the direct solver with the Sericola algorithm. The values of the constants of the model are taken from [8]. The size of the downloaded file is 4 MB.

In Figure 5a the transfer time distribution of the file is depicted, where both the client and the server have a 56 kbps modem connection. The plots on the figure compare the results obtained by the direct distribution solver and the moment based estimation (using 5 and 17 moments). This figure demonstrates that the tool offers a choice between analysis speed and accuracy. With the moment based estimations it is possible to get an initial idea about the shape of the distribution quickly, before executing the time consuming exact distribution solver.

In Figure 5b we investigate the effect of the server side connection on the transfer times. On the client side an ISDN connection is assumed (128 kbps), and the server is connected with a modem (56 kbps), ISDN (128 kbps), and ADSL (256 kbps up). If the server is connected by a modem, then it will be the bottleneck of the transfer; in the other two cases the 128 kbps ISDN bandwidth will be the tightest connection. The DSL performs better because, while the client side load conditions are the same, the server side has more bandwidth, and thus will be the bottleneck less frequently.

Figure 5c depicts the transfer time distribution as the function of `lambda`, thus, as the function of the incoming transfer request rate. Both the client and the server have ISDN connection in this example. This means that the minimum and maximum download times are the same for all `lambda` ($32000/128 = 250$ seconds and $32000/(128/4) = 1000$ seconds). With increasing `lambda` the probabilities move toward the maximum download time.

7. Conclusions

The analysis of complex systems based on the tagged customer approach is theoretically known, but usually too complex to perform manually. To support this analysis model description languages and associated automatic analysis tools are required. This paper presents a tool with this functionality which was not available before according to the authors knowledge.

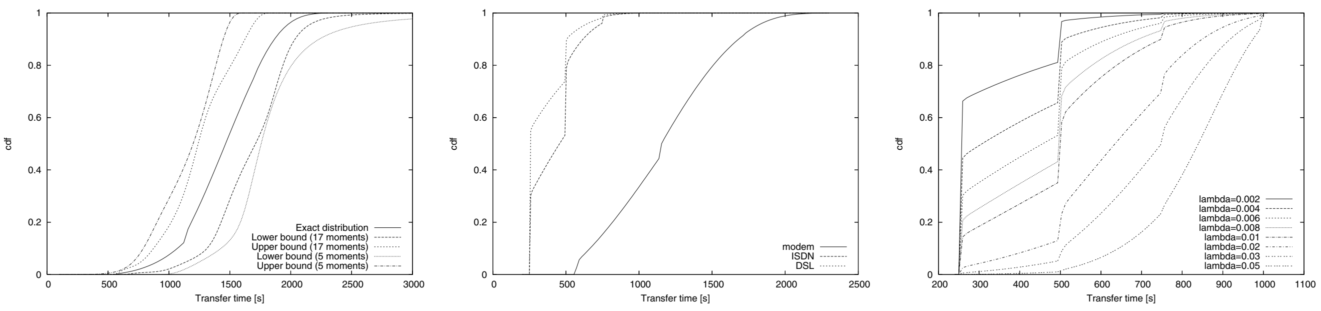
The main elements of the analysis of the tagged customer approach are the mapping of the Markov chain describing the system without the tagged customer to the one describing the system with the tagged customer, the definition of the initial distribution of the second Markov chain based on the stationary behaviour of the first one, and the construction of the second Markov chain according to the required performance measure. The proposed model description and analysis allows a flexible definition of these main elements.

Two telecommunication examples demonstrate the modeling and analysis abilities of the proposed approach. The examples are taken from the literature, are described by one-page description files, and are automatically analyzed by the MRMSolve tool.

The computational complexity of the analysis is mainly determined by the applied reward analysis method. Apart of some very special cases the general automatic analysis based on MRMSolve has the same complexity as the anal-

```
type prs;
Const
    Pcu=2;  # maximal number of parallel client uploads
    Pcd=2;  # maximal number of parallel client downloads
    Psu=2;  # maximal number of parallel server uploads
    Psd=2;  # maximal number of parallel server downloads
    wcu=0.4;   # portion of upload requests on a client with modem connection
    wcd=0.21;  # portion of download requests on a client with modem connection
    wsu=0.07;  # portion of upload requests on a server with isdn connection
    wsd=0.07;  # portion of download requests on a server with isdn connection
    lambda=0.006; mu1=0.001; mu2=0.1; alpha1=0.6; alpha2=0.4;
    modem_bw = 56; isdn_bw  = 128; dsl_upld = 256; dsl_dwld = 1024;
State
    a1 : 0 To Pcu; a2 : 0 To Pcu; b1 : 0 To Pcd; b2 : 0 To Pcd;
    c1 : 0 To Psu; c2 : 0 To Psu; d1 : 0 To Psd; d2 : 0 To Psd;
Condition
    a1+a2 <= Pcu; b1+b2 <= Pcd-1; c1+c2 <= Psu-1; d1+d2 <= Psd;
Matrix Q
    # uploads to the client
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1+1,a2,b1,b2,c1,c2,d1,d2] = wcu*lambda*alpha1;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2+1,b1,b2,c1,c2,d1,d2] = wcu*lambda*alpha2;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1-1,a2,b1,b2,c1,c2,d1,d2] = mu1*a1 ;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2-1,b1,b2,c1,c2,d1,d2] = mu2*a2 ;
    # downloads from the client
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1+1,b2,c1,c2,d1,d2] = wcd*lambda*alpha1;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2+1,c1,c2,d1,d2] = wcd*lambda*alpha2;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1-1,b2,c1,c2,d1,d2] = mu1*b1 ;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2-1,c1,c2,d1,d2] = mu2*b2 ;
    # uploads to the server
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2,d1+1,d2] = wsu*lambda*alpha1;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2,d1,d2+1] = wsu*lambda*alpha2;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2,d1-1,d2] = mu1*d1 ;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2,d1,d2-1] = mu2*d2 ;
    # downloads from the server
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1+1,c2,d1,d2] = wsd*lambda*alpha1;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2+1,d1,d2] = wsd*lambda*alpha2;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1-1,c2,d1,d2] = mu1*c1 ;
    [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2-1,d1,d2] = mu2*c2 ;
Vector R
    # client is modem, server is isdn
    [a1,a2,b1,b2,c1,c2,d1,d2] = min(modem_bw/(a1+a2+b1+b2+1),isdn_bw/(c1+c2+d1+d2+1));
Vector P0 embedded
    Const
        Pcu=2;  # maximal number of parallel client uploads
        Pcd=2;  # maximal number of parallel client downloads
        Psu=2;  # maximal number of parallel server uploads
        Psd=2;  # maximal number of parallel server downloads
        wcu=0.4;   # portion of upload requests on a client with modem connection
        wcd=0.21;  # portion of download requests on a client with modem connection
        wsu=0.07;  # portion of upload requests on a server with dsl connection
        wsd=0.07;  # portion of download requests on a server with dsl connection
        lambda=0.006; mu1=0.001; mu2=0.1; alpha1=0.6; alpha2=0.4;
        modem_bw = 56; isdn_bw  = 128; dsl_upld = 256; dsl_dwld = 1024;
    State
        a1 : 0 To Pcu; a2 : 0 To Pcu; b1 : 0 To Pcd; b2 : 0 To Pcd;
        c1 : 0 To Psu; c2 : 0 To Psu; d1 : 0 To Psd; d2 : 0 To Psd;
    Condition
        a1+a2 <= Pcu; b1+b2 <= Pcd; c1+c2 <= Psu; d1+d2 <= Psd;
    P0generator
        # uploads to the client
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1+1,a2,b1,b2,c1,c2,d1,d2] = wcu*lambda*alpha1;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2+1,b1,b2,c1,c2,d1,d2] = wcu*lambda*alpha2;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1-1,a2,b1,b2,c1,c2,d1,d2] = mu1*a1 ;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2-1,b1,b2,c1,c2,d1,d2] = mu2*a2 ;
        # downloads from the client
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1+1,b2,c1,c2,d1,d2] = wcd*lambda*alpha1;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2+1,c1,c2,d1,d2] = wcd*lambda*alpha2;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1-1,b2,c1,c2,d1,d2] = mu1*b1 ;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2-1,c1,c2,d1,d2] = mu2*b2 ;
        # uploads to the serer
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2,d1+1,d2] = wsu*lambda*alpha1;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2,d1,d2+1] = wsu*lambda*alpha2;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2,d1-1,d2] = mu1*d1 ;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2,d1,d2-1] = mu2*d2 ;
        # downloads from the server
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1+1,c2,d1,d2] = wsd*lambda*alpha1;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2+1,d1,d2] = wsd*lambda*alpha2;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1-1,c2,d1,d2] = mu1*c1 ;
        [a1,a2,b1,b2,c1,c2,d1,d2]->[a1,a2,b1,b2,c1,c2-1,d1,d2] = mu2*c2 ;
    Mapping
       (b1+b2)<Pcd & (c1+c2)<Psu : [a1;a2;b1;b2;c1;c2;d1;d2]-{1}->[a1;a2;b1;b2;c1;c2;d1;d2];
```

Table 2. Model description of the P2P example



a) Distribution and its estimates b) Effect of server side connections c) Effect of background load

Figure 5. Transfer time distribution in the P2P example

ysis based on specific softwares developed for the analysis of a particular performance measure of a given model.

References

[1] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor Trivedi. *Queueing Networks and Markov Chains*. John Wiley and Sons, 1998.

[2] S.C. Borst, R. Nunez-Queija, and M.J.G. van Uitert. User-level performance of elastic traffic in integrated-services networks. *Performance Evaluation*, pages 507–519, 2002.

[3] J. A. Carrasco. Markovian dependability/performability modelingof fault-tolerant systems. In H. Pham, editor, *Reliability Engineering Handbook*, pages 613–642. 2003.

[4] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. Greatspn 1.7: Graphical editor and analyzer for timed and stochastic petri nets. *Performance Evaluation*, 24(1-2):47–68, November 1995.

[5] G. Ciardo, J. Muppala, and K.S. Trivedi. SPNP: stochastic Petri net package. In *PNPM89*, pages 142–151. IEEE Computer Society, 1989.

[6] Joseph A. Couvillion, Roberto Freire, Ron Johnson, W. Douglas Obal, M. Akber Qureshi, Manish Rai, William H. Sanders, and Janet E. Tvedt. Performability modeling with ultraSAN. *IEEE Software*, 8(5):69–80, 1991.

[7] G. Fodor, S. Rácz, and M. Telek. On providing blocking probability- and throughput guarantees in a multi-service environment. *International Journal of Communication Systems*, 15:4:257–285, May 2002.

[8] R. Gaeta, M. Gribaudo, D. Manini, and M. Sereno. Fluid stochastic petri nets for computing transfer time distributions in peer-to-peer file sharing applications. *Electronic Notes in Theoretical Computer Science*, 128:79–99, 2005.

[9] R. Gaeta, M. Gribaudo, D. Manini, and M. Sereno. Analysis of resource transfers in peer-to-peer file sharing applications using fluid models. *Perform. Eval.*, 63(3):149–174, 2006.

[10] H. Hermanns, U. Herzog, and V. Mertsiotakis. Stochastic process algebras as a tool for performance and dependability modelling. In *Proc. of IPDS*, Erlangen, Germany, 1995. IEEE Computer Society Press.

[11] H. T. Kaur, D. Manjunath, and S. K. Bose. The queuing network analysis tool (QNAT). In *Proceedings 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, volume 8, pages 341–347, 2000.

[12] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. American Statistical Association and the Society for Industrial and Applied Mathematics, 1999.

[13] Benjamin Melamed and Micha Yadin. Numerical computation of sojourn-time distributions in queuing networks. *Journal of ACM*, 31(4):839–854, 1984.

[14] M.K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Tr. on Computers*, C-31:913–917, 1982.

[15] J. K. Muppala, K. S. Trivedi, V. Mainkar, and V. G. Kulkarni. Numerical computation of response time distributions using stochastic reward nets. *Annals of Oper. Res.*, 48(1-4):155–184, 1994.

[16] J.K. Muppala, S.P. Woolet, and K.S. Trivedi. Composite performance and dependability analysis. In *Proc. of PMCCS* , pages 131–161, Enschede (NL), 1991.

[17] H. Nabli and B. Sericola. Performability analysis: a new algorithm. *IEEE Transactions on Computers*, 45:491–494, 1996.

[18] S. Rácz, B. P. Tóth, and M. Telek. MRMSolve: Numerical analysis of large Markov reward models. In *Tools 2000*, pages 337–340. Springer, LNCS 1786, 2000.

[19] AT&T Labs Research. Graphviz - open source graph drawing software. http://www.research.att.com/sw/tools/graphviz.

[20] R. Sahner, K.S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems*. Kluwer Academic Publisher, 1996.

[21] Willaim H. Sanders and John F. Meyer. Stochastic activity networks: formal definitions and concepts. In *first EEF/Euro summer school on trends in computer science*, pages 315–343. Springer-Verlag New York, Inc., 2002.

[22] A. Tari, M. Telek, and P. Buchholz. A unified approach to the moments based distribution estimation - the unbounded case. In *EPEW*, pages 79–93, Versailles, France, Sept 2005.

[23] J. Walrand. *An Introduction to Queueing Networks*. Prentice-Hall, 1988.