Subnetwork Time Distributions as a Means for Multilevel Evaluation of Generalized Stochastic Petri Nets

Guenter Klas, Reinhard Matuschka
Siemens Corporate R&D, ZFE IS SYS 13
Otto-Hahn Ring 6, 8000 Munich 83, Germany

Abstract
In this paper a new hierarchical evaluation procedure for Generalized Stochastic Petri Nets (GSPN) is being presented. It is based on extensions of Flow Equivalent Aggregation (FEA). At every level of hierarchy, subnets are approximated by substitute networks. As criterion for the similarity of the networks, the Subnetwork Time Distribution (STD) is used which is the sojourn time distribution of a token \( X \) in a subnet conditioned on the token distribution at the epoch of arrival of \( X \) and the context into which the net is embedded for determining the STD. The computation of a substitute network's type and parameters is outlined. The performance of this technique called FEAD (FEA based on Subnetwork Time Distribution) is discussed by means of examples.

1 Introduction

1.1 Background
Generalized Stochastic Petri Nets (GSPN) \([14]\) are often used besides queueing networks in performance and performability evaluation of computers and communication networks. Since GSPNs are usually evaluated by solving the global balance equations, their numerical treatment soon gets unmanageable with an increasing state space. Two basic approaches exist to cope with the state space explosion.

First, the problem may be shifted to finding a powerful numerical solution method on matrix level. Then, usually the whole state space is constructed and the global balance equations are approximately solved \([7,18,5,10,15]\). The approximation quality often depends on the degree of nearly completely decomposability of the nets.

Second, a homogeneous hierarchical solution of GSPNs may be performed by replacing submodels at every level of hierarchy by less complex substitute nets. This avoids the construction of the whole state space. The basic aggregation step consists of finding a substitute net which behaves similar to the original submodel. This may be done by analyzing a submodel under certain conditions, e.g. open subnet with a predefined token input process \([12]\) or a directly fed back, closed submodel with a fixed token population. The latter approach stems from Norton's theorem or Flow Equivalent Aggregation \([6,16]\) which is only exact for nets with local balance. FEA has been applied to non-product form nets on a heuristic basis \([2,9,1,17]\). It was extended by simulative preanalysis of subnets \( e.g. \) in \([3]\).

In \([11]\) FEA was considered for GSPN submodels with internal resources and blocking of output transitions. The authors of \([11]\) propose a special aggregate network which takes into account (i) the delay which a token experiences until it is accepted by the submodel's input and (ii) the token's processing time in the submodel. Therefore, the aggregated net is similar to a Generalized Branching Erlang net \([16]\) with 1 and 2 stages. These considerations naturally lead to the approach outlined in the following sections when the structure of the GSPN submodel is allowed to be quite general. The original submodel is replaced by a substitute network which shows a similar stochastic behaviour under the same stimulating conditions. A crucial point is, therefore, the definition of similarity.

1.2 Contribution
We propose a hierarchical extension of standard FEA called FEAD (FEA based on Subnetwork Time Distribution STD). FEAD is an analytical
hierarchical solution method for computing approximately the steady state solution of a large GSPN net. It differs from former approaches, e.g. [3], in four respects:

1) The types of substitute networks (or aggregates) are different.
2) As criterion for the similarity of original subnet and aggregate, a token's load dependent sojourn time in the net under varying token distributions at the epoch of the token's arrival is used.
3) The STD conditioned on the submodel's GSPN neighbourhood at every level of hierarchy is considered.
4) The aggregation is applied at multiple levels of hierarchy. The model hierarchy seems to be more general then the one recently proposed in [4].

With FEAD the submodels are replaced by substitute networks by using the subnetwork time distribution as matching criterion. The STD is computed in dependence on the submodel's context and its arrival point token distribution.

The rest of the paper is organized as follows: in chapter 2 some definitions are introduced. Then in chapter 3 the STD is defined and its computation discussed. The estimation of a substitute network's type and parameters is shown in chapter 4. The FEAD algorithm is summarized in chapter 5. After providing numerical results in chapter 6 we will give some conclusions in chapter 7.

2 Definitions

A GSPN model G is hierarchically decomposed into subnets. G is represented by a tree whose vertices are labelled by subnetwork names. The root of the tree, G, is of hierarchy level 0. Define for a network N \( S(N) \) as set of networks that are direct successors of N. Define for all T \( \in S(N) \) \( P(T) = \{N\} \), which means that N is direct predecessor of T. Since we use an "is part of" semantic for the edges of the tree, all nets T \( \in S(N) \) are part of the net P(T), but \( \cup T \in S(N) \), T is not necessarily P(T). A subnet N has to adhere to the following restrictions:

1) Tokens enter subnet N via one entry place \( e \in N \) and leave net N via one exit transition.
2) N does neither produce nor consume tokens.
3) There is no direct feedback of N.

Definition 2.1 Neighbourhood

Let N be a subnet. The neighbourhood v(N) of N is defined as

\[ v(N) = P(N) \cup S(P(N)) \setminus \{N\}. \]

Definition 2.2 Neighbourhood test network

A network M is called a neighbourhood test network if it is closed and consists of the series connection of a subnet N and a load dependent server L with exponentially distributed service times.

Definition 2.3 Isolation

A net N is considered under isolation as net Ni if it is separated from the original net and if a direct feedback is applied from its exit transition to its entry place.

Definition 2.4 R-, T-tokens, population

Tokens which model local resources in a net are defined R-tokens. Tokens that enter and leave a net are denoted T-tokens. Since, with GSPN, tokens are indistinguishable, the distinction refers only to the population of a net which is counted in T-tokens.

We refer to T-tokens simply as tokens and mention R-tokens explicitly. The state space of a closed net N with p tokens is denoted \( Z_p \) or \( Z_p(N) \). The restriction of a set of markings I to the places of a net N is denoted \( [I]_N \). Places are annotated with there capacity c if \( c \neq \infty \), e.g. place1.c. \#S denotes the number of tokens in place S. In the sequel the STD is defined and its computation is outlined.

3 The Subnetwork Time Distribution

We are interested in the time it takes for a token X to proceed through a subnet N that holds k-1 tokens when X enters the subnet.

3.1 Definition of the Subnetwork Time Distribution (STD)

Definition 3.1 Subnetwork Time Distribution

Let N be a subnetwork of a given GSPN model. \( D_{N,M}(t,s) \) is defined as the distribution of time that a token X, initially placed in the entry place of N, needs to proceed through the network N when k-1 tokens are in the subnet N at the time of the entrance of token X and when N is embedded in a GSPN context. The strategy s fixes the set \( I_{F,k-1} \) of arrivals instant states with k-1 tokens in N and their distribution \( F_{k-1} \).

Definition 3.2 F-net and STD-net

An F-net serves for the computation of an arrival instant distribution \( F_{k-1} \) and has a state space \( Z_{k-1} = I_{F,k-1} \). A STD-net is an augmentation of the F-net and is isomorphic to an absorbing Markov chain. It serves for the computation of the STD. For its initial markings \( [I_{STD}]_N = I_{F,k-1} \), the

\[ v(N) = P(N) \cup S(P(N)) \setminus \{N\}. \]
markings $I_{F,k-1}$ with 1 additional token (the kth one) in the entry place of net $N$.

Since $F$ is a priori unknown (although in principle computable from the original net) we use heuristics for establishing a distribution $F$. For a fixed value of $k$ we consider the following strategies:

**WA** Worst Case Analysis
Set $I_{F,k-1}$ is fixed by a marking with k-1 tokens initially located in the entry place of $N$. The support of $F_{k-1}$ is a singleton.

**SY** Stationary Distribution Analysis
The F-net consists of net $N_i$ with k-1 tokens circulating in it. This yields $F_{k-1}$ of the markings of a state space $Z_{k-1}$. Before computing $D_{N,k}(t,SY)$, k-1 tokens initially are located in the places of $N$ according to $F_{k-1}$.

**ND** Neighbourhood Dependence Analysis
The F-net consists of the neighbourhood test network $M$ which is constructed of subnet $N$ and a load dependent exponentially distributed server $L$ that has been gained by means of FEA of the neighbourhood $v(N)$. The following preliminary analysis is performed in order to fix the initial distribution $F_{k-1}$ of the i=k-1 tokens in $N$:

Given $p$ tokens in $M$, there will be, on an average, $n(p) \in \mathbb{R}$ tokens in $N$ and $p-n(p)$ tokens in $L$ after a stationary analysis of $M$. Before computing $D_{N,k}(t,SY)$, k-1 tokens initially are located in the places of $N$ according to $F_{k-1}$.

3.2 Computation of the Subnetwork Time Distribution $D_{N,k}(t,s)$

$D_{N,k}(t,s)$ is computed from an absorbing continuous time Markov chain (CTMC). For its initial states $I_{STD}$ holds: $I_{STD}N = I_{F,k-1}$. The absorbing states are defined by the fact that token $X$, initially located in the entry place of net $N$, has just left $N$. Since a close tracking of token $X$, which is indistinguishable from other tokens in GSPN net $N$, is quite complex, we consider the following approximation for the computation of $D_{N,k}(t,s)$: an absorbing state is reached after the kth firing of the exit transition of $N$.

$D_{N,k}(t,s)$ is computed at equidistant time points $t_i = i \Delta t$ for $i = 0...c$ as time to absorption of the absorbing CTMC by means of the randomization technique [8] and, in the case of an acyclic Markov chain, by means of the algorithm ACE [13] which yields the distribution also in a symbolic representation. Now we show the constructions needed to compute $D_{N,k}(t,s)$ for the strategies WA, SY and ND with $k=2$ for an example. Figure 3.1a shows the submodel $N$ considered.

All exponential transitions have rate 1.

**Strategy WA:**
The initial marking seen by the arriving second token is determined by 1 token in place $S_1$.

**Strategy SY:**

Since the stationary distribution of k-1 tokens in the closed submodel is used as the arrival point distribution which is seen by the arriving kth token, stationary analysis for k-1 = 1 circulating token of $N_i$ supplies the stationary distribution $F_1$ of the states $F_k$ seen by the second token: $P(a_1,a_2,a_3,a_4) := P(a_i \text{ tokens in } S_i, i = 1...4)$
\begin{align*}
P(O,1,1,0) &= 0.25 \quad P(1,1,0,0) = 0.25 \\
P(O,0,1,1) &= 0.25 \quad P(0,1,0,1) = 0.25
\end{align*}

Dimension of state space \( Z_1 \): 4. Figure 3.1c shows the STD-net. For each state \( j \in I_{F,1} \) an immediate transition \( t_j \) has to be produced with probability \( P(a_1 j, a_2 j, a_3 j, a_4 j) \) for the state \( (a_1 j, a_2 j, a_3 j, a_4 j) \), whose multiple arcs with weights \( a_{ij} \) lead to places \( S_i \). Moreover, a place Start and arcs from Start to all transitions \( t_j \) are constructed. \( D_{N,2}(t, SY) \) is then computed as the time to absorption of a MC with initial states (3.1), but with 1 additional token in place \( S_1 \).

Below: example for state \((0,1,1,0)\)

![Image](image.png)

**Figure 3.1c:** Strategy SY, STD-net

**Strategy ND:**
The neighbourhood of net \( N \) is aggregated to a load dependent server (figure 3.1d). The initial distribution \( F_1 \) of 1 token in net \( N \) is determined by stationary analysis for one and two tokens of the neighbourhood test network \( M \) which supplies the stationary distribution of states of the state spaces \( Z_1 \) and \( Z_2 \).

1 token: mean number of tokens in \( N \): 0.78
2 tokens: mean number of tokens in \( N \): 1.69.

\[ F_{1}=Z_1 \cup Z_2 \]

Next, we proceed analogously to Strategy SY, but for both state spaces \( Z_m \), \( m = 1, 2 \). For each state \( j_m \) an immediate transition \( t_{jm} \) is appended with probability \( g(m)P(j_{jm}) \). The multiple arcs of these transitions with weights \( a_{ij_m} \) lead to places \( S_i \). A place Start and arcs from Start to all transitions \( t_{jm} \) have to be produced. This constitutes the STD-net. \( D_{N,2}(t, ND) \) is computed analogously to \( D_{N,2}(t, SY) \) using the initial states from spaces \( Z_1 \) and \( Z_2 \) augmented by one token in place \( S_1 \).

So far, a GSPN subnet \( N \) is characterized by means of its Subnetwork Time Distribution. Using this STD as matching criterion we compute an optimum substitute network which is less complex than the original subnet.

![Image](image.png)

**Figure 3.1d:** Strategy ND, part of the STD-net

**4 Selection of the substitute network**

**Problem statement:**
Given subnet \( N \) and STD \( D_{N,d}(t_{is}), k=1...K=\text{capacity}(N), t_i = i^*\Delta t, i=0...c \) under a strategy \( s \), we have to find a substitute network \( SN \) that shows similar stochastic behaviour and has a smaller state space than \( N \).

Since we use STDs as matching criterion, substitute network types should feature small model complexity and practical computability of these distributions. The subnetwork time distribution \( D_{E_1,k}(x,t,s) \) for a substitute network type \( E_i \) is given symbolically or numerically in dependence on the parameters \( x = [x_1, ..., x_{\text{max}}] \) with \( \text{max} \) as number of parameters of the network type \( E_i \).

The subnetwork time distribution of both the original net \( N \) and the aggregate net \( E \), \( D_{N,d}(t_{is}) \) and \( D_{E,d}(t_{is}) \) respectively, vary over the range of number of tokens in the net. Starting from the STDs \( D_{N,d}(j^*\Delta t), j=0...c \) of a token when arriving as the \( d \)th customer at net \( N \), we determine an optimum aggregate network type \( E_0 \) from a set of types \( \{E_i\} \), \( i=1...m \). The optimum type \( E_0 \) is determined only from the equivalent service time distribution of submodel \( N \), i.e. \( D_{N,i}(j^*\Delta t), j=0 \). After this type is fixed, the aggregate net's parameters are adjusted to account for load dependence, such that \( D_{N,d}(j^*\Delta t) = D_{E_0,d}(j^*\Delta t), d=2...K \).
The type of aggregate network \( E_0 \) is determined by sequencing through the set \( \{ E_i \} \), recording new minimum types and stopping even before the end of the set is reached if the Euclidean norm of the two vectors \( D_{N,1} \) and \( D_{E,1} \) is less than a prescribed minimum. We start the ordered set \( \{ E_i \} \) with \( E_1 \) as a load dependent exponential server. This yields the equivalent mean service time of submodel \( N \). This parameter is used for types \( E_i \), \( i \geq 2 \) to get a close initial guess \( x^i \) in algorithm 4.1. We use the following sequence:

\( \{ E_i \} = \{ E_1, E_2, E_3, E_4, B_2, B_3, H_2, H_3, C_2 \} \)

with \( E_r \) denoting an Erlang-k server, \( B_2 \) denoting a Branching Erlang-k server, \( H_k \) denoting a hyperexponential server with \( k \) branches and \( C_2 \) a Cox-2 server.

**Algorithm 4.1:**

begin
for all types \( E_i \in \{ E_i \} \)
minimize \( \| D_{N,1} - D_{E,i} \| \)
\( \rightarrow \) optimum type \( E_0 \).

for \( d = 2 \) to \( K \)

end

The optimal parameters of the aggregate net \( E_i \) for load \( d \times x_1, x_2, \ldots, x_{\text{max}} \) are computed by minimizing the following functional:

\[
(4.1) \quad \min_{(x_1, \ldots, x_{\text{max}})} f(x_1, \ldots, x_{\text{max}}) = \sum_{j=0}^{c-1} \left[ D_{N,d}(i^* \Delta t) - D_{E,i,d}(x_1, \ldots, x_{\text{max}}, i^* \Delta t) \right]^2.
\]

This constitutes a non-linear optimization problem. Let \( F_d^k \) denote the Jacobian matrix for net type \( E_k \) and a load of \( d \) tokens with

\[
F_d^k(i_j) = \frac{\partial D_{E_k,d}(x_1, x_2, \ldots, x_{\text{max}}, i^* \Delta t)}{\partial x_j}
\]

and \( i = 0, \ldots, c-1, j = 1, \ldots, \text{max} \). The flow time distribution \( D_{E_k,d}(x) \) is linearized as

\[
D_{E_k,d}(x + \Delta x) = D_{E_k,d}(x) + F_d^k(x) \Delta x + O(\Delta x^2)
\]

The solution of (4.1) is then iteratively possible by the following algorithm:

**Algorithm 4.2**

\[
x = x^i \quad / * \text{initial parameter guess} */
\]

do
solve the following linear optimization problem
(4.2) for \( \Delta x \):

\[
\min f(\Delta x) = \sum_{j=0}^{c-1} \left[ D_{N,d}(i^* \Delta t) - D_{E,i,d}(x_1, \ldots, x_{\text{max}}, i^* \Delta t) - F_d^i(j, \Delta x) \right]^2
\]

\( x = x + \Delta x \)

while \( \| \Delta x \| > \epsilon \) AND \( f > \eta \). For problem (4.2) the normal equations are established and solved for \( \Delta x \)

\[
F_d^i \Delta x = D_{N,d} - D_{E_i,d}(x).
\]

In the next section, the complete algorithm is discussed.

5 Summary of the algorithm

Two variants of FEAD are presented. Let \( N_h \) be the set of networks of hierarchy level \( h \).

**Algorithm 5.1: Variant One**

FEAD without dependence on neighbourhood of subnetworks, Strategy \( s = \text{WA} \) or \( s = \text{SY} \).

begin
for hierarchy level \( h = H = \max(h) \) to 1 do

end

**Algorithm 5.1: Variant Two**

FEAD with dependence on neighbourhood of subnetworks, Strategy \( s = \text{ND} \).

begin
for hierarchy level \( h = H \) to 1 do

end

5 Summary of the algorithm

Two variants of FEAD are presented. Let \( N_h \) be the set of networks of hierarchy level \( h \).

**Algorithm 5.1: Variant One**

FEAD without dependence on neighbourhood of subnetworks, Strategy \( s = \text{WA} \) or \( s = \text{SY} \).

begin
for hierarchy level \( h = H = \max(h) \) to 1 do

end

**Algorithm 5.1: Variant Two**

FEAD with dependence on neighbourhood of subnetworks, Strategy \( s = \text{ND} \).

begin
for hierarchy level \( h = H \) to 1 do

end
6 Numerical results

Two examples that are kept simple for the sake of clarity are analyzed. Although they do not justify any aggregation due to their small complexity, they provide some insight into the algorithm's performance.

Example 1:
Consider the GSPN model in figure 6.1.

Figure 6.1: Example of a GSPN model which is hierarchically solved by FEAD

The following algorithms are compared:
1) exact solution (E)
2) Flow Equivalent Aggregation (F)
3) FEAD Variant One WA (WA)
4) FEAD Variant One SY (SY)
5) FEAD Variant Two ND (ND).

As performance criterion we regard the deviation
\( e = \frac{\text{result}_{\text{approximate solution}} - \text{result}_{\text{exact solution}}}{\text{result}_{\text{exact solution}}} \)

in
a) mean number of tokens in place \( Q \),
b) mean throughput of transition \( T \),
e) probability of finding \( i \) tokens in place \( Q \)
\( \text{prob}(\#Q=i) \) for \( i = 0 \ldots 3 \).

Moreover, the rate \( r \) of transition \( T \) which influences the coupling degree of the submodels is varied. The results are presented in figures 6.2 and 6.3. In this example, we find that Flow Equivalent Aggregation shows a bad performance, especially in the mean number of tokens in place \( Q \) and in \( \text{prob}(\#Q=1) \), in comparison to the other aggregation algorithms. This indicates that the subnetwork time distribution is a good criterion for the characterization of subnets at several levels of hierarchy. Furthermore, the different strategies with FEAD may be discussed. We find that FEAD-SY behaves similarly to FEAD-ND, although ND is slightly better than SY. Surprisingly FEAD-WA shows the best performance in computing the throughput of \( T \) and \( \text{prob}(\#Q=0) \) and is nearly as good as ND and SY in the other values. This indicates that the positive effect gained by using the STD as criterion for network similarity is much greater than the supplementary advantage stemming from a consideration of the network's neighbourhood.

Next, the aggregation results are considered with respect to the coupling degree of the models which is heavily influenced by the firing rate of transition \( T \). Figures 6.2 and 6.3 show that standard FEA performs worst with respect to the other algorithms when the ratio of the submodel input rate \( r \) to the submodels' internal rates (\( = 1.0 \)) is in the interval \([0.1, \ldots, 1]\). This effect is alleviated when FEAD with various strategies is used.

From a computational point of view FEAD-WA is the most efficient FEAD variant.

Example 2:
Figure 6.4 shows the simple model of a multiprocessor system with 3 processors. The model is decomposed into processor and cache system and the bus and memory subsystem SNA. Model SNA, in turn, holds a submodel SNB which accounts for the memory system. Transitions and the model's parameters are defined in table 6.1.

The application of algorithm 5.1 Variant One with strategy WA yields the following results for the throughput of transition \( T_1 \):

\[
\begin{array}{l|c|c}
\text{Algorithm} & \text{Exact result} & \text{Error in \%} \\
\hline
\text{FEAD-WA} & 0.1024 & 4.1 \\
\text{FEA} & 0.1124 & 5.2 \\
\end{array}
\]

47
The subnets are approximated by the following substitute nets where $\lambda(i)$ denotes the rate parameter of an exponential phase for a load of $i$ tokens:

\[
\begin{array}{c|ccc}
N & SN & \lambda(i): i = 1 & 2 & 3 \\
SNB & Erlang-2 & 0.10 & 0.104 & 0.112 \\
SNA & Erlang-4 & 0.13 & 0.156 & 0.174 \\
\end{array}
\]

Figure 6.5 compares the subnetwork time distributions $D_{\alpha}(t)$ of the original subnet SNA and the substitute nets for token populations $d \in \{1, 2, 3\}$.

7 Conclusions

We have presented an approach for the hierarchical solution of Generalized Stochastic Petri Nets using a step by step aggregation of GSPN submodels. Subnetworks are approximated by substitute networks using the Subnetwork Time Distribution STD introduced in chapter 3 as criterion for the stochastic equivalence of subnet and aggregate. An STD equals the sojourn time distribution of a token in a subnet dependent on the token distribution $F$ seen by the arriving token and dependent on the subnet's GSPN neighbourhood.

A problem consists in the computation or efficient estimation of the a priori unknown arrival point token distribution $F$. We approached this by using heuristic strategies. In chapter 4 we discussed the determination of the optimum substitute network type and the estimation of the load dependent parameters. In our prototype implementation we computed the optimum parameters of a series of substitute network types by nonlinear optimization. For nontrivial substitute nets we compute e.g. the Jacobian of an STD by numerical differentiation and the STD by numerical convolution. Although this approach of sequencing through a series of net types is readily extended to new substitute network types, which is suitable in an experimental phase, it seems to be more appropriate to restrict the aggregate network types to a single class, e.g. 2 stage Coxian networks for which the parameter estimation problem is directly solved either numerically or analytically.

The algorithm presented in chapter 5 was implemented in a tool. Since major parts of this tool like compiler, reachability generator, steady state solver and algorithms for the transient analysis were taken from a general GSPN analyzer developed at the authors' institute, modules with well defined interfaces were available. For this reason and for the sake of easy testing, most constructions necessary in algorithm 5.1 as e.g. applying a direct feedback to a subnet were performed on GSPN level by manipulating GSPN specification files. For a more efficient implementation a direct manipulation of the generator matrices is recommended.

The additional improvement introduced by taking into account a subnet's neighbourhood was shown to be relatively small with respect to the dominating improvement achieved by accounting for the second moment in a subnet's equivalent service time distribution. In the considered example, standard Flow Equivalent Aggregation yields worse results than the aggregation algorithms presented in this paper which motivates further research in this area.

Acknowledgement

Thanks are due to the referees and Helen Hoffmann for reviewing the final paper, to Roland Lepold from Siemens Labs Munich and to Prof. Elke Jessen from Technical University Munich.

References


Figure 6.2: a) Absolute error of mean number of tokens in place Q, b) of mean throughput of T
Figure 6.3: Absolute error of the aggregation algorithms in the probability of a fixed number of tokens in place $Q$ (see the model in figure 6.1).
Figure 6.4: Example 2: multiprocessor

Table 6.1: Parameters of example 2

<table>
<thead>
<tr>
<th>trans.</th>
<th>rate</th>
<th>probability</th>
<th>signification</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0.1</td>
<td>#S1</td>
<td>rate for access to cache</td>
</tr>
<tr>
<td>T2</td>
<td>0.7</td>
<td></td>
<td>cache hit rate</td>
</tr>
<tr>
<td>T3</td>
<td>0.3</td>
<td></td>
<td>cache miss rate</td>
</tr>
<tr>
<td>T4</td>
<td>0.25</td>
<td></td>
<td>delay time on cache bus</td>
</tr>
<tr>
<td>T5</td>
<td>1.0</td>
<td></td>
<td>access to memory bus, 3 slots</td>
</tr>
<tr>
<td>T50</td>
<td>0.66</td>
<td></td>
<td>delay time of a request to mem.</td>
</tr>
<tr>
<td>T6</td>
<td></td>
<td></td>
<td>if(#S7=0) then 1.0 main memory access else 0.0</td>
</tr>
<tr>
<td>T7</td>
<td></td>
<td></td>
<td>if(#S7=1) then 0.2 disk I/O 1 else 0.0</td>
</tr>
<tr>
<td>T8</td>
<td></td>
<td></td>
<td>if(#S7=1) then 0.8 disk I/O 2 else 0.0</td>
</tr>
<tr>
<td>T90</td>
<td>1.0</td>
<td></td>
<td>access time in main memory</td>
</tr>
<tr>
<td>T10</td>
<td>0.01</td>
<td></td>
<td>access time disk 1</td>
</tr>
<tr>
<td>T11</td>
<td>0.005</td>
<td></td>
<td>access time disk 2</td>
</tr>
<tr>
<td>T12</td>
<td>10.0</td>
<td></td>
<td>data reply on memory bus</td>
</tr>
<tr>
<td>T13</td>
<td>0.25</td>
<td></td>
<td>data reply on memory bus</td>
</tr>
<tr>
<td>T14</td>
<td>0.25</td>
<td></td>
<td>data reply on cache bus</td>
</tr>
<tr>
<td>T13, T14</td>
<td></td>
<td></td>
<td>model the scheduling of data replies w.r.t. requests</td>
</tr>
<tr>
<td>T15</td>
<td>0.25</td>
<td></td>
<td>data reply on cache bus</td>
</tr>
</tbody>
</table>

Table 6.1: (cont.)

<table>
<thead>
<tr>
<th>arc</th>
<th>weight</th>
<th>resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>2</td>
<td>number of active processors</td>
</tr>
<tr>
<td>w2</td>
<td>if #S40 = 1 then 1 else 0</td>
<td></td>
</tr>
<tr>
<td>w3</td>
<td>2</td>
<td>number of slots in the packet switched memory</td>
</tr>
<tr>
<td>w4</td>
<td>if #S70 = 1 then 1 else 0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.5: STDs for subnet SNA and aggregate: d = number of tokens for FEAD-WA, m = mean of the subnetwork time distribution