Stochastic Well Formed Model of the UTO3 Example

Marina Ribaudo
Dipartimento di Informatica
Università degli Studi di Torino

1 Problem description

We consider a parallel reactive program that monitors a system by controlling the outcome of N sensors. The basic structure of the program is depicted in Figure 1:

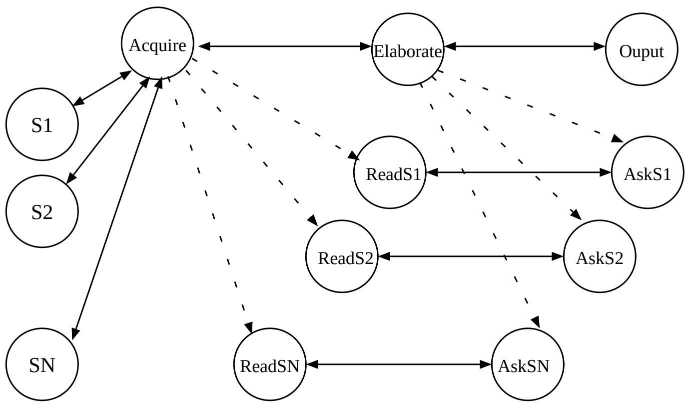

Figure 1: Structure of the program.

processes are represented by circles, communications are represented by arcs between processes and process activations are represented by dotted arcs. **S1**, **S2**, ..., and **SN** represent the devices of the system to be monitored: an anomaly in the i-th device corresponds to a message being sent from **Si** to process **Acquire**, through channel Sig_i. The main function of process **Acquire** is to listen to the channels Sig_i, $i = 1, \ldots, N$ and to forward any message received on Sig_i to process **Elaborate**. The processing triggered by the reception of a signal consists of the acquirement of the current values of all the other signals, and of some computation on the measured values. It thus follows that, upon reception of a message from process **Acquire**, process **Elaborate** activates $N-1$ processes **AskSi** with the special task of getting the current values of the other signals. In a similar way process **Acquire**, upon sending a signal to **Elaborate**, activates $N-1$ processes **ReadSi** with the specific task of reading the values and forwarding them to processes **AskSi**. There is also an **Output** process that prints the result of the computation. After a given number of signals is received, some statistics are computed and the monitoring system starts again.

2 Modelling assumptions

We assume that communications among processes are synchronous of the *rendezvous* type: input/output statements can be performed when both partners are ready to communicate.

The program is characterized assuming an unlimited number of physical resources; the analysis proposed is intended to be performed in the early stages of the design, when the full implementation may not yet be available and the target architecture may not have been chosen. Being machine independent, the analysis provides very general results that may be considered as theoretical maximum of the performance obtainable in the case of limited physical resources.

3 The SWN model of the example

In this section we show a simple SWN model corresponding to the concurrent program previously described. This model has been developed following an approach that has already been proposed in the literature for the translation of concurrent programs into GSPN models [1].

The program structure exibits a certain degree of symmetry. In fact processes **Si** behave in the same way and can be considered different instances of the same process (each instance is characterized by a different name). Processes **Acquire** and **Elaborate** have similar structures: synchronization and communication phases are followed by parallel activations of the same number of children. This regular structure gives rise to the compact and quite regular SWN model depicted in Figure 2. Only a basic colour class is used, it is defined as $SG = \{s_1, s_2, \ldots, s_N\}$ and it represents the N devices. No ordering relation exists among different devices and thus the basic color class SG is not ordered. Notice that the net is parametric in the number of devices: changing the cardinality of the class SG the global number of objects in the net changes but the net structure does not change. Places and transitions are either labeled by the colour domain SG or not coloured. Timed transition Par0 represents the parallel activation of the three processes **Acquire**, **Elaborate** and **Output**. Places i, j and k model control variables and are introduced for executing the body of these processes the same number of times; the firing of Par0 adds 20 tokens in places i, j, and k. Place Si models the N devices and in the initial marking it contains one token per basic colour of class SG (denoted by the diffusion function S inside the place). Hence transition gen_Si is enabled for any possible instantiation of the colour class SG. A token present in place Acquire represents the fact that process **Acquire** is suspended waiting for signals. If a message is sent from a process **Si** on channel Sig_i (a token is present in place SiaIt) and place i is not empty, a synchronization can occur and immediate transition SigAc_R [1] can fire. Then a communication phase takes place, modeled by timed transition SigAc_com. During this phase the message in the channel is actually read. As soon as **Acquire** has read the message it forwards it to process **Elaborate**: immediate transition AcEl_R represents the synchronization between these two processes and timed transition AcEl_com represents the communication phase. After this comunication is completed both processes activate $N-1$ children for the acquisition of the current

[1] The name of the transition is built using the first letters of the names of the processes involved in the synchronization while R stands for *rendezvouz*

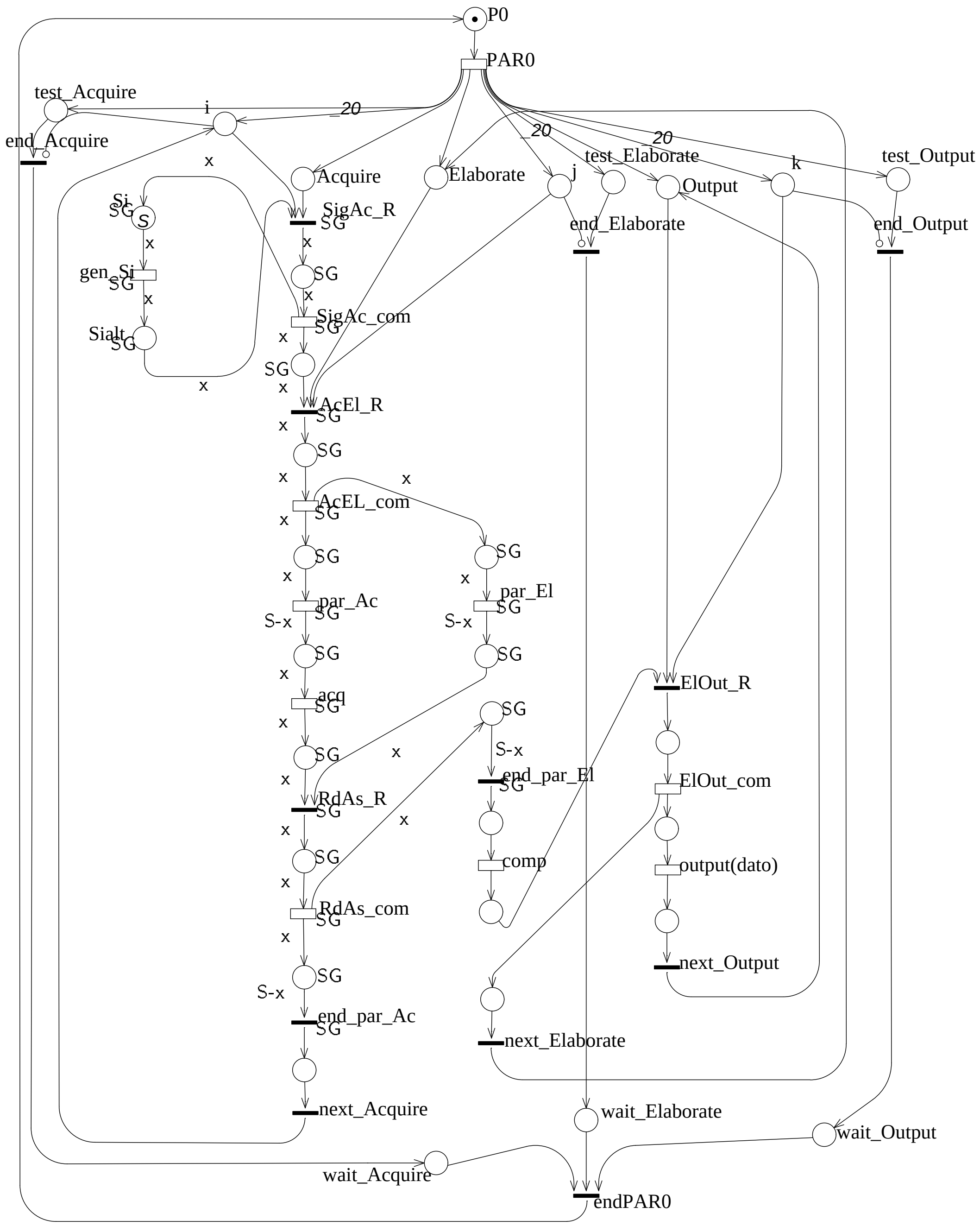

Figure 2: SWN model of the program.

values of all other signals. The parallel activation of processes **ReadSi**, children of **Acquire**, is modeled by timed transition par_Ac whose output arc is labelled by the colour function $S - x$. The effect of the firing of transition par_Ac is to remove one token of colour $\langle s_i \rangle$ from the input place and simultaneously to add all coloured tokens but one (the token $\langle s_i \rangle$ just removed) in the output place. Processes **AskSi**, children of **Elaborate**, are activated in the same way with the firing of timed transition par_El. Each **ReadSi** acquires the current value of the i-th device (timed transition acq is fired for different instances of the colour class SG) and then it synchronizes with the corresponding **AskSi** (immediate transition RdAs_R). It follows a communication between couples of corresponding processes (timed transition RdAs_com) and, when all **ReadSi** and all **AskSi** have terminated their executions, **Acquire** and **Elaborate** can resume. Immediate transition end_par_Ac can thus fire only if all the **ReadSi** processes have terminated. This is modelled using the colour function $S - x$ on the input arc of transition end_par_Ac that can fire only if the corresponding input place contains all the coloured tokens but one (corresponding to the signal $\langle s_i \rangle$ that has activated the whole process). Similarly, process **Elaborate** resumes its execution after the termination of all its children (transition end_par_El). A computation phase follows, modeled by timed transition comp. Finally a synchronization (immediate transition ElOut_R) and a communication (timed transition ElOut_com) between **Elaborate** and **Output** take place. Transition output(dato) models the printing of the computed results. When places i, j and k are empty immediate transitions end_ Acquire, end_Elaborate and end_Output are enabled. The effect of their firing is to add tokens in places wait_Acquire, wait_Elaborate and wait_Output enabling transition end_PAR0 that ends the whole program.

4 Performance parameters

Different performance parameters can be defined on the SWN model to quantify the behavior of the concurrent program. We can distinguish two different situations: (a) signals have the same quantitative behavior, i.e. they have the same arrival rate and processing time, (b) signals have different arrival rates and processing time. In the first case we can take advantage of the behavioral symmetries of the system to perform a more efficient analysis: more abstract performance results can be computed in this case from which the detailed results can be however derived. Otherwise the analysis complexity is similar to that of the unfolded net.

A parameter of interest is the communication rate between processes. Let us consider processes **Acquire** and **Si**: all communications between them take place through channel Sig_i. We can thus evaluate the amount of data exchanged between the two processes considering the throughput of transition Sig_Ac_com. In case (a) the communication rate is defined as the overall throughput of transition SigAc_com (independently of the specific instance) divided by the total number of signals in the system (which is the cardinality of class SG).

$$(a) \;\; communication(\text{Acquire, Si}) = \frac{X(SigAc\_com)}{\mid SG \mid}$$

In case (b) instead we need to introduce static subclasses in the model. If all signals have different behaviors each static subclass SG_i contains exactly one signal. Communication cost between **Acquire** and **Si** is thus evaluated considering the throughputs

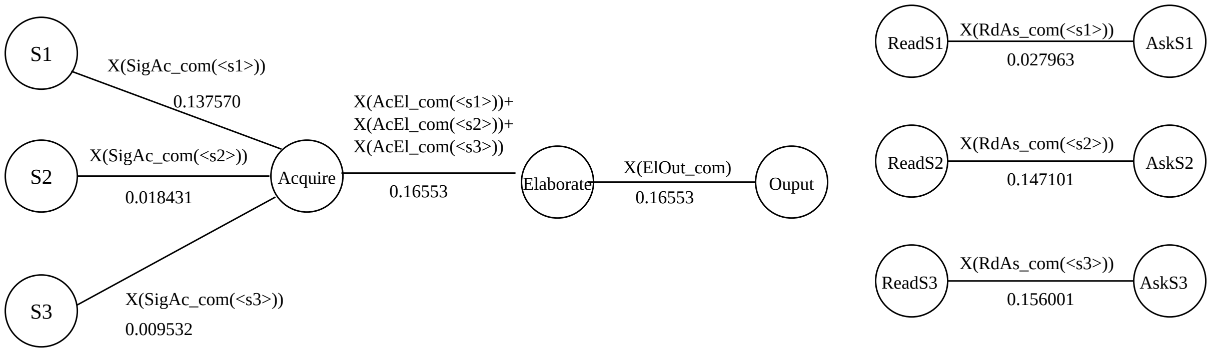

Figure 3: Communication graph.

of the transition instances Sig_Ac_com($\langle s_i \rangle$) with $s_i \in SG_i$ (these measures can be derived from the reachability graph). There is an intermediate case in which signals can be partitioned into static subclasses SG_i with $\mid SG_i \mid > 1$, each subclass containing signals that behave in the same way. Communication rate between **Acquire** and signal **Si** is evaluated considering the total throughput of all the transition instances SigAc_com($\langle x \rangle$) with $x \in d(S_i)$ [2] and dividing it by the cardinality of the static subclass.

This system can be classified as reactive, thus another performance measure of interest could be related to the program ability to serve external requests. In particular we are interested in finding the maximum arrival rate that the monitor program can manage without loosing too many signals. We thus consider the probability of having a signal waiting on channels Sig_i (because the acquisition process **Acquire** is not ready to accept it). In case (a) this probability can be computed considering the mean number of tokens in place Sialt divided by the total number of signals.

$$(a) \ \ Siwait = \frac{\overline{M}(Sialt)}{\mid SG \mid}$$

In case (b) we simply compute the probability of having the token $\langle s_i \rangle$ in place Sialt.

$$(b) \ \ Siwait = Prob\{M(Sialt) = \langle s_i \rangle\}$$

5 Results

The performance measures introduced in Section 4 have been computed solving a particular instance of the SWN model depicted in Figure 2 with three devices. We have thus defined the colour class SG as $SG = \{s_1, s_2, s_3\}$ and we associated the following arrival rates l_i to each signal: $l_1 = \lambda$, $l_2 = \frac{\lambda}{10}$, $l_3 = \frac{\lambda}{20}$ where λ ranges over the interval $[0.1, 0.8]$. We assumed that frequent signals (high values of the rates l_i) required quick computations, communications and visualizations.

In Figure 3 the communication graph of the program is shown. The arcs are labeled both with the formula used to compute the corresponding weight and with the values obtained when the parameter λ is set to 0.2.

Figure 4 shows the rate of communication between **Acquire** and **S1**, for λ ranging over the interval $[0.1, 0.8]$. Observe the curve of Figure 4: the rate of communication

[2] $d(S_i)$ is a function that returns the subclass S_i belongs to

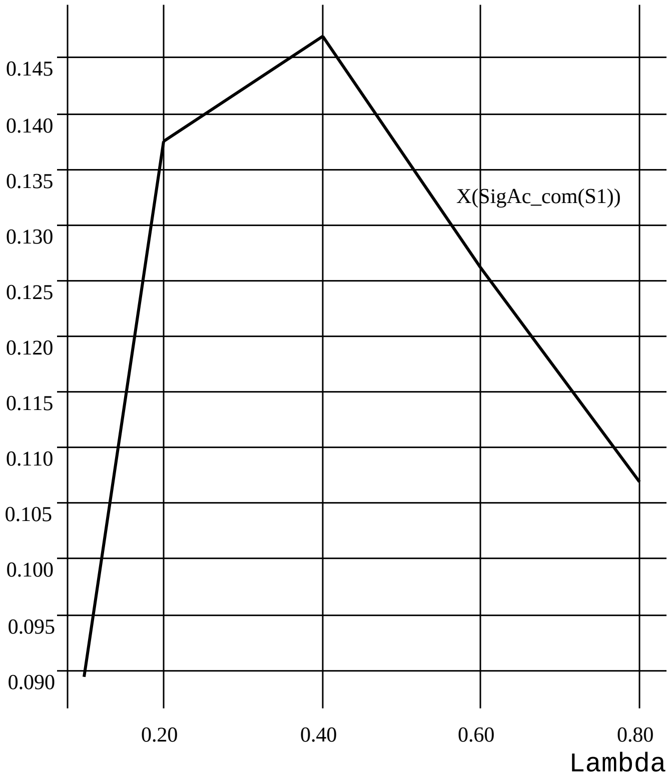

Figure 4: Communication between Acquire and S1.

increases until λ reaches a threshold value, for higher values of λ the curve starts decreasing. This behaviour can be physically explained as follows: for low values of λ the system spends more time in the processing of signals arriving from the first device (**S1**) that requires less computation. When λ grows, the probability of being in a situation in which all signals **S1**, **S2**, **S3** are available grows too. Under these circumstances the probability of analyzing a certain signal is not proportional to its interarrival frequency so that the number of signals of type **S2** and **S3** acquired by the monitor grows: thus the communication between process **Acquire** and **S1** is reduced. Finally observe that in this case the number of signals that the system can process per unit of time decreases for increasing values of λ since more time is needed for processing less frequent signals.

Figure 5 shows the probability of having a signal **S1** waiting on channel Sig_1 for different values of λ. We can observe that this probability increases for higher values of λ because in this case the probability of having the other signals available on channels Sig_2 and Sig_3 grows too. If we assume that a signal sent on a channel is lost if process **Acquire** does not immediately read it, we can also observe that the number of lost signals **S1** is larger for higher values of λ (i.e. when the probability of having the token s_1 in place Sialt is closer to one).

References

[1] G. Balbo, S. Donatelli, and G. Franceschinis. Understanding Parallel Programs Behaviour through Petri net models. *Journal of Parallel and Distributed Comput-*

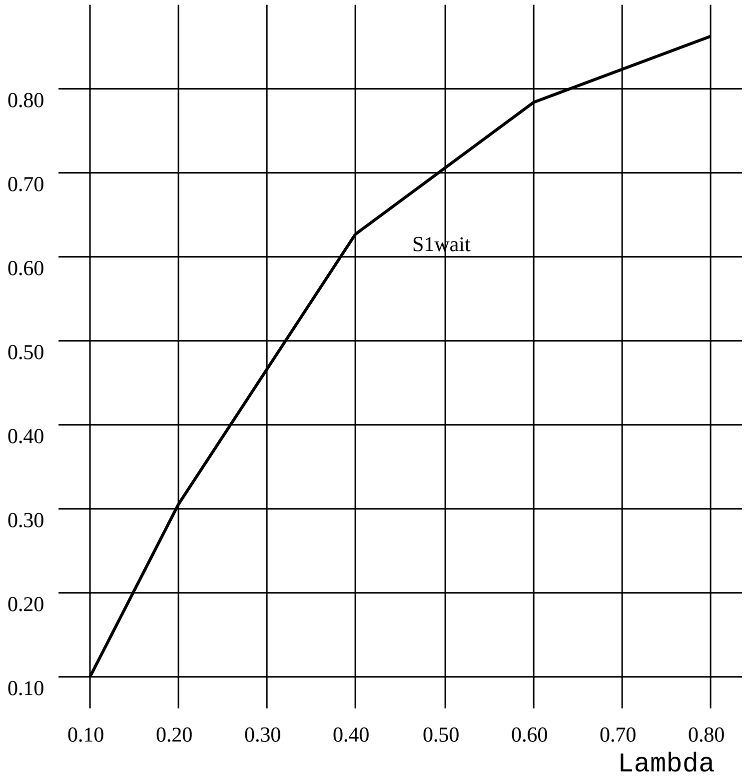

Figure 5: Waiting of signal S1.

ing, 15, October 1992.

[2] G. Balbo, S. Donatelli, G. Franceschinis, A. Mazzeo, A. Mazzocca and M. Ribaudo. On the computation of Performance Characteristics of Concurrent Programs using GSPNs. *Journal of Performance Evaluation*, to appear.