SWN Models of the UTO2 Example

C. Anglano and G. Chiola
Dipartimento di Informatica
Università degli Studi di Torino

1 Modelling assumptions

The description of the problem is ambiguous about the scheduling of tasks on the nodes. On the one hand no hypothesis is explicitly stated in case of preemptive interaction, besides the priority of communication service tasks over local computation tasks: either processor sharing or single server FCFS would be logically acceptable for the scheduling of the N computation tasks. On the other hand, non-preemptive interaction implicitly assumes a non interruptible, single server scheduling of computation tasks (so that, e.g., processor sharing and round robin would be logically incompatible). This lack of specification is reflected in (and was actually pointed out by) our Petri net models. As we show by reasoning on the PN models such lack of information does not affect the performance average performance results that we consider, but this consideration is made a-posteriori.

Also nothing is said about the presence of communication delays in the interconnection network. We assume that communication delays are negligible in our model. The addition of two delay transitions to take them into account would be however straightforward.

2 Modelling non-preemptive interaction

The first interaction policy considered is the non-preemptive one, in which external requests are not preemptive, but are served at the end of the local operation with higher priority with respect to computation tasks. As already pointed out, this model assumes a non interruptible, single server type scheduling of computation tasks.

In Fig. 1 the SWN model corresponding to the non-preemptive policy for a $n \times m$, 2-dimensional (2D) mesh is depicted. This SWN model contains two **ordered colour class** Px and Py of processor identities that allows the use of **successor functions** $\oplus x$ to model connections with nearest neighbour processors. Colour class Px contains $|Px| = n$ objects, while class Py contains $|Py| = m$ objects. Each processor is uniquely identified by a pair, representing its coordinates in the 2D mesh.

The colour class Px (respectively Py) is partitioned in two **static subclasses** Px_0 and Px_+, with $|Px_0| = 1$ (respectively Py_0 and Py_+, with $|Py_0| = 1$). This allow the use of the two constant functions S_0 and S_n such that $\oplus S_n = S_0$ in order to identify the two boundary elements of the first dimension of the mesh (respectively, S_0 and S_m for the second dimension). The identification of first and last element in each

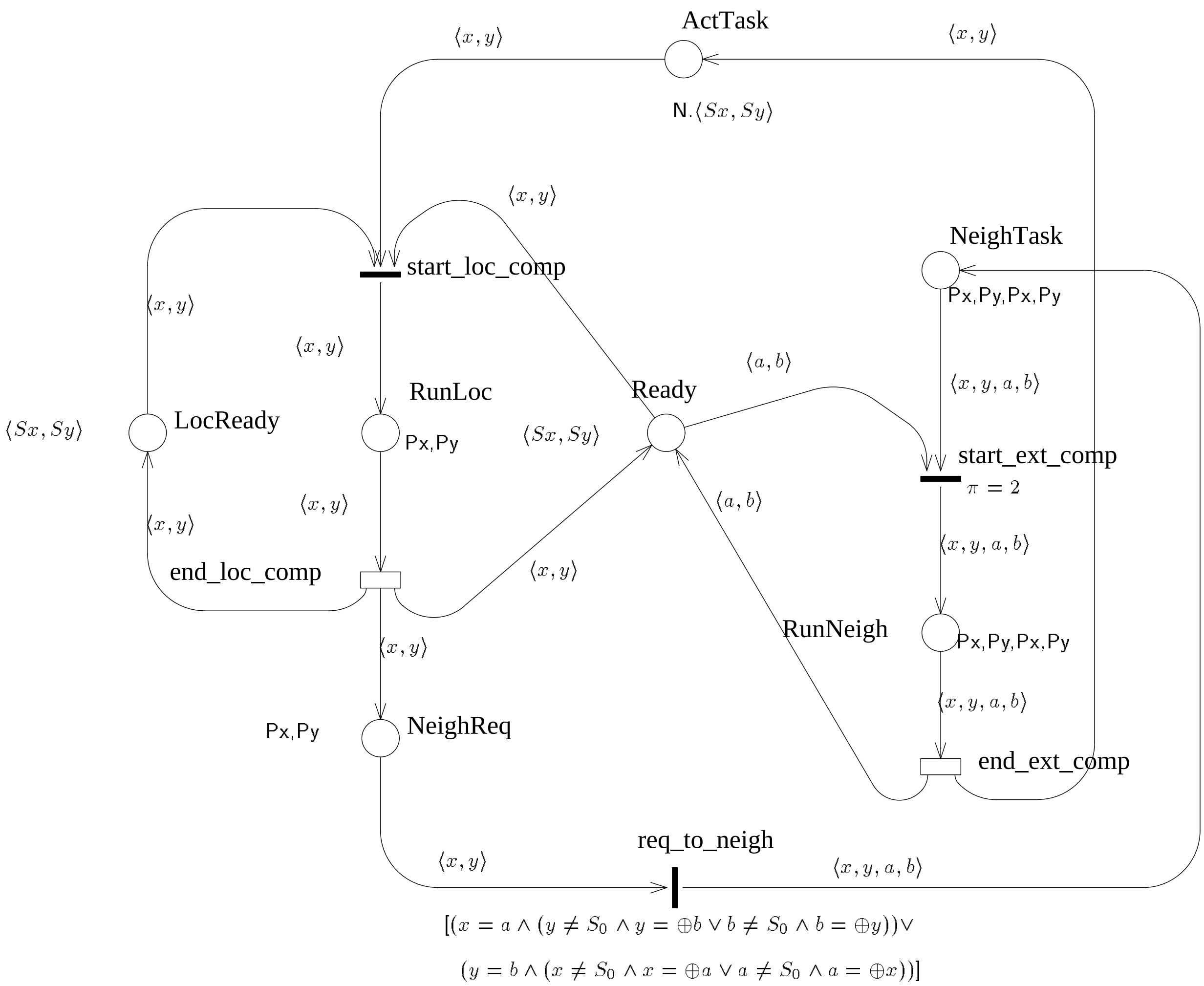

Figure 1: $n \times m$, 2D mesh with non-preemptive policy

dimension in the enabling predicate associated with transition req_to_neigh implements the boundary conditions for the interconnection topology (all internal processors are connected to the two nearest neighbours in each dimension except for the first and last one that are connected with only one neighbour).

Place ActTask models the presence of N indistinguishable tasks on the different processors, each one characterized by a pair representing the two coordinates of the processor on which it is allocated. Immediate transition start_loc_comp models the scheduling of such tasks one at a time on each processor by synchronizing the tokens in place ActTask with the ones in place LocReady, initially containing only one token per coordinate pair. Place Ready initially containing one token per coordinate pair represents the availability of processors for the execution of computation and server tasks. Transition end_loc_comp (whose rate is λ) models the activity of processors that are not currently involved in a communication. Its firing represents issuing a communication request to another processor. The selection of the destination processor is made at random through the firing of immediate transition req_to_neigh, that "invents" the identity of the destination by means of the additional components a and b in the function labelling its output arc. The values that can be bound to functions a and b

are restricted by the transition predicate in order to correctly implement the nearest neighbours connection on the 2D mesh topology. Immediate transition start_ext_comp models the scheduling of the activity related to the message handling at the destination processor. The completion of the activity related to the communication is modelled by the timed transition end_ext_comp (whose rate is μ), The non-preemptive priority policy is simply obtained by giving immediate transition start_ext_comp higher priority than immediate transition start_loc_comp. Indeed once a processors has chosen to start a local computation phase, the token with processor identity is removed from place Ready and it is not available for the enabling of transition start_ext_comp until the local operation completes.

2.1 Structural model analysis and validation

By checking the incidence functions of the net and the initial marking, one may easily realize that place LocReady is behaviourally as well as structurally implicit. Indeed its marking may never reduce the enabling degree of its only output transition start_loc_comp since it is always greater than or equal to the marking of the implicant place Ready. It may then be deleted together with its input and output arcs without affecting the behaviour of the model in any way.

After the erasure of the implicit place LocReady the net exhibits two coloured, minimal support P-semiflows, namely:

1. $\forall \langle x, y\rangle \in Px \times Py$, $\exists a \in Px$, $\exists b \in Py$ such that

$$\langle x, y\rangle \cdot M[\mathsf{ActTask}] + \langle x, y\rangle \cdot M[\mathsf{RunLoc}] + \langle x, y\rangle \cdot M[\mathsf{NeighReq}] +$$

$$\langle x, y, a, b\rangle \cdot M[\mathsf{NeighTask}] + \langle x, y, a, b\rangle \cdot M[\mathsf{RunNeigh}] = N \geq 1$$

and $(a = x) \wedge ((y \neq S_0) \wedge (\oplus b = y) \vee (b \neq S_0) \wedge (b = \oplus y)) \vee (b = y) \wedge ((x \neq S_0) \wedge (\oplus a = x) \vee (a \neq S_0) \wedge (a = \oplus x))$

2. $\forall \langle x, y\rangle \in Px \times Py$, $\exists a \in Px$, $\exists b \in Py$ such that

$$\langle x, y\rangle \cdot M[\mathsf{RunLoc}] + \langle x, y\rangle \cdot M[\mathsf{Ready}] + \langle a, b, x, y\rangle \cdot M[\mathsf{RunNeigh}] = 1$$

and $(a = x) \wedge ((y \neq S_0) \wedge (\oplus b = y) \vee (b \neq S_0) \wedge (b = \oplus y)) \vee (b = y) \wedge ((x \neq S_0) \wedge (\oplus a = x) \vee (a \neq S_0) \wedge (a = \oplus x))$

The fact that all places are covered by P-semiflows is sufficient to guarantee boundedness of the model, while an intelligent use of the two invariant relations is sufficient to prove liveness and reversibility (i.e., the ergodicity of the corresponding Markovian process). The details of the formal proofs are omitted for the sake of brevity. Only a sketch of the liveness proof is reported here.

- Consider all possible cases satisfying invariant number 1 for any pair $\langle x, y\rangle$:

  $j \geq 1$ **tokens** in places RunLoc, NeighReq, or RunNeigh:
  the output transitions are enabled since they have only one input arc labelled with identity functions; their firing produce markings with at least one token in places NeighReq, NeighTask, or ActTask, respectively.

$j \geq 1$ **tokens** in place NeighTask:
consider all possible cases allowed by invariant number 2:

1 token in place Ready:
transition start_ext_comp is enabled, and its firing yields one token in place RunNeigh;

1 token in place RunNeigh:
transition end_ext_comp is enabled, and its firing yields one token in place Ready;

1 token in place RunLoc:
transition end_loc_comp is enabled, and its firing yields one token in place Ready.

$j \geq 1$ **tokens** in place ActTask and 0 tokens in place NeighTask:
considerations on all possible cases allowed by invariant number 2 similar to the one made in the previous case show the liveness of transition start_loc_comp.

$j \geq 1$ **tokens** in place ActTask and $k \geq 1$ tokens in place NeighTask:
Transition start_loc_comp cannot be enabled until place ActTask becomes empty, due to the higher priority of transition start_ext_comp. However, each firing of start_ext_comp (that is live) decreases the marking of NeighTask by 1 token. The causal connection chain from start_loc_comp to end_loc_comp to req_to_neigh shows however that the only way of putting new tokens in NeighTask is to fire transition start_loc_comp eventually, the worst case being $j = 1$ and $k = N - 1$, in which start_loc_comp becomes enabled only after k firings of start_ext_comp. Hence also transition start_loc_comp is live.

Notice that all these formal proofs are parametric with respect to the cardinality of the colour classes (i.e., are valid $\forall n, m \geq 1$).

3 Modeling preemptive interactions

The second model proposed is relative to the preemptive interaction policy. Please remember that in the preemptive interaction policy requests coming from the neighbours are served with higher priority than the local activity, immediately preempting any computation task. However nothing is assumed concerning the scheduling of computation tasks.

In Fig. 2 the model for a $n \times m$ 2D mesh is depicted. The definition of the basic colour class is exactly the same as for the model of the non-preemptive case. The main difference with respect to the SWN depicted in Figure 1 is that the arc from place Ready to transition start_loc_comp has been moved to transition end_loc_comp. In this way the token representing the processor is used for the enabling of timed transition end_loc_comp, but is not removed during the execution of computation tasks. Immediate transition start_ext_proc may be enabled by an incoming communication request (if no other communication request is currently being served) even during the execution of a computation task. The fact that start_ext_proc is immediate forces its firing right away, thus possibly disabling end_loc_comp. The memoryless property of the firing time distribution of transition end_loc_comp makes it unnecessary the specification

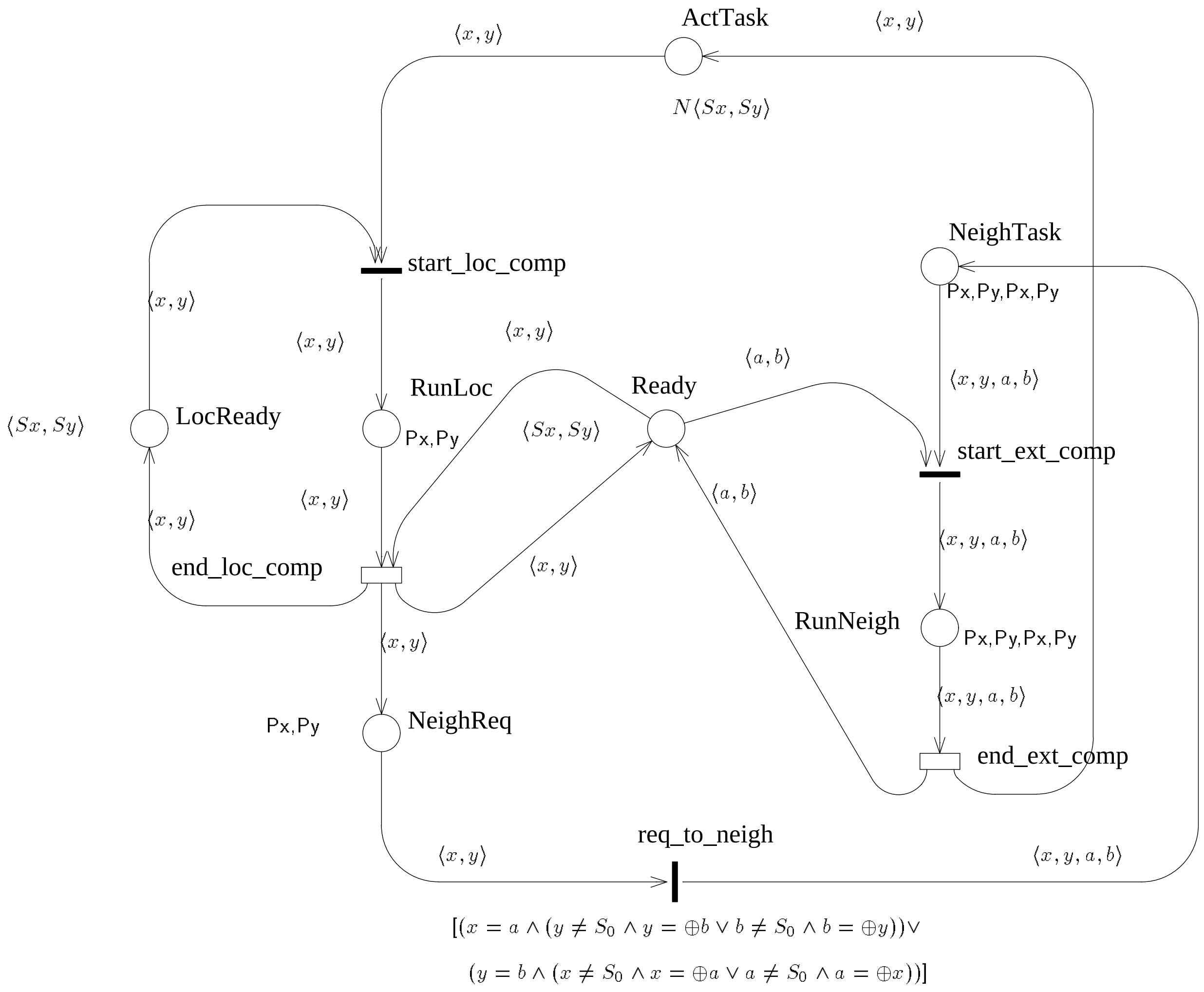

Figure 2: $n \times m$ 2D mesh with preemption

of an age memory policy to correctly handle the resume of activity after preemption. The completion of the activity related to the communication is still modelled by the timed transition end_ext_comp (whose rate is μ). Its firing may resume the enabling of transition end_loc_comp by returning the corresponding processor identity token in place Ready (if no other communication requests are pending).

In this case, place LocReady is not behaviourally redundant with respect to the enabling degree of transition start_loc_comp. However, the latter is an immediate transition, for which the enabling degree is not semantically relevant. The enabling degree of timed transition end_loc_comp that is semantically crucial in the behaviour of the timed model, instead is not affected by the presence of place LocReady, since it is already limited to the maximum value 1 by the marking of place Ready. Therefore, removing place LocReady does not change the timed measurable behaviour of the model. The net resulting from the erasure of place LocReady from the net depicted in Figure 2 differs from the original in the the model can be interpreted. In the original model, a single server, non interruptible scheduling (besides preemption for the service of higher priority tasks) is represented. Indeed a new token representing a different computation task may arrive in place RunLoc only after the service of the previous one is complete.

In the transformed net, all tasks are potentially executable by the processor at the same time, so that, e.g., a round robin scheduling might be assumed. Due to the memoryless property of the task completion time, the distribution of the firing time of transition end_loc_comp is still exponential, independently of the scheduling discipline. These considerations made on the SWN model prove that the lack of specification of the model description cannot affect the computed results in case of exponential distributions.

From the model point of view, after the erasure of place LocReady the net may be further reduced without affecting its qualitative and performance properties by removing immediate transition start_loc_comp and merging places ActTask and RunLoc into a single place.

Boundedness, liveness, and reversibility may be formally proven in a parametric way as in the non-preemptive case. The details of the proofs are omitted here.

4 Changing the number of dimensions

The models depicted in Figures 1 and 2 model a 2D mesh architecture. To model higher dimensional mesh connections, higher order tuples for the colour domains representing processor identities may be used. Please note that the predicate associated with transition req_to_neigh needs to be modified to correctly handle all possible boundary situations.

To extend the above models to a 3D mesh architecture, for instance, we introduce a third basic colour class Pz (as before partitioned into two static subclasses Pz_0 and Pz_+) containing k objects. The obtained model is a $n \times m \times k$ 3D mesh.

Figure 3 depicts the model for a $n \times m \times k$ 3D mesh architecture with non-preemptive interaction. The model for the preemptive interaction can be obtained exactly in the same way starting from the corresponding 2D case.

5 Performance measures

Different performance indices can be defined for the above described models.

The first of them is the utilization of the processors for different values of N. The utilization can be specialized in the utilization for the internal operations and the utilization for the external requests. From the classical utilization law it is possible to define:

$$\begin{cases} U_{loc} = \dfrac{x_{end\_loc\_comp}}{\lambda} \\ U_{ext} = \dfrac{x_{end\_ext\_comp}}{\mu} \end{cases}$$

This definition holds for both preemptive and non-preemptive cases. The global utilization is given by $U = U_{loc} + U_{ext}$.

Another interesting parameter is the average time spent waiting for synchronization. This parameter can be estimated applying Little's law to place Ready, based on the explicit computation of the throughputs and of the average markings.

Both Markovian numerical analysis (for small configurations, the size being limited by the total number of states in the tangible reachability graph of the order of 10^5–10^6) and discrete event simulation can be used to obtain "exact" performance estimate. Both techniques may be quite expensive to apply in terms of CPU time. In case of non

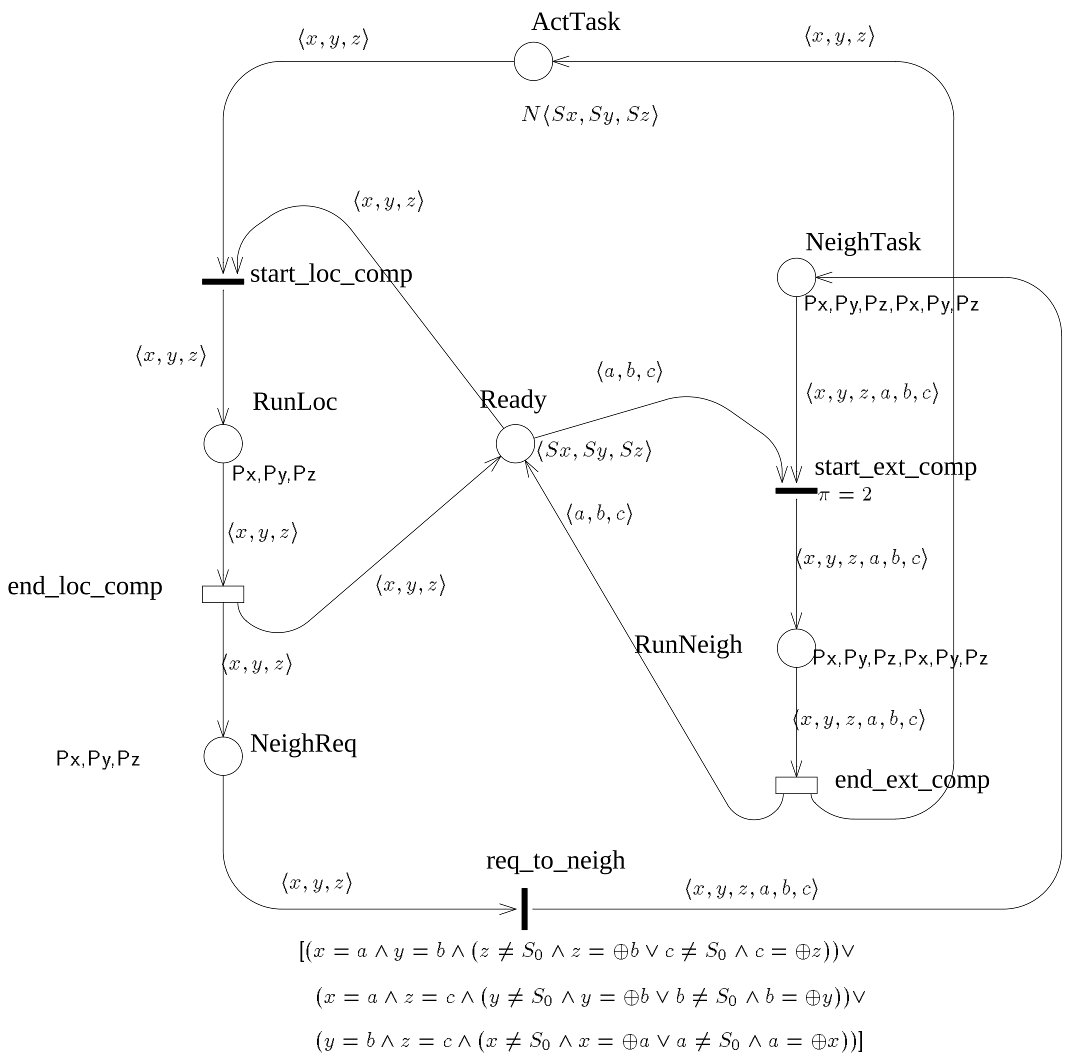

Figure 3: 3D $n \times m \times k$ mesh with non-preemptive interaction

stiff problems the numerical Markovian Technique may provide numerical accuracy of the order of 10^{-5}–10^{-7}, while simulation runs usually produce confidence intervals of the order of 1–10%.