Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DYNAMO

HS 2023

Prof. Dr. Lana Josipović and Jiahui Xu
based on Prof. Lothar Thiele's material

Discrete Event Systems

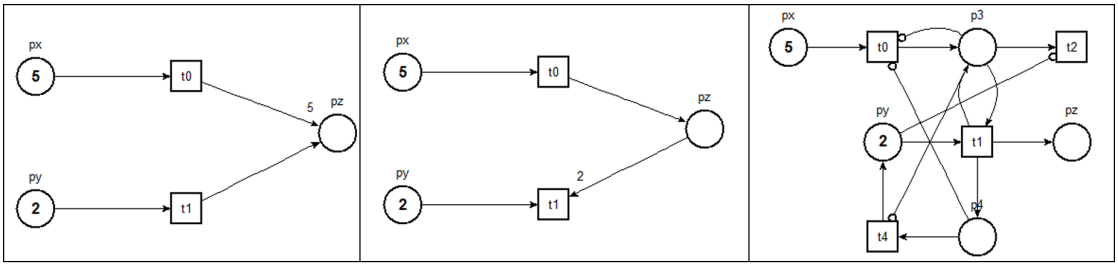
Solution to Exercise Sheet 14

1 Time Petri Net

step	τ	t_{fired}	M^τ	L^τ
0	0	-	$[0, 1]$	$(t_3, 2)$
1	2	t_3	$[2, 1]$	$(t_1, 3)$, $(t_3, 4)$
2	3	t_1	$[0, 2]$	$(t_3, 4)$, $(t_2, 5)$
3	4	t_3	$[2, 2]$	$(t_1, 5)$, $(t_3, 6)$, $(t_2, 6)$
4	5	t_1	$[0, 3]$	$(t_3, 6)$, $(t_2, 6)$
5	6	t_2	$[2, 1]$	$(t_3, 8)$, $(t_1, 7)$

2 Calculating with Petri nets

There are multiple options here. I just present one set of solutions. These are captures of TINA models of these nets. The models are available with the solution.



a) b) c)

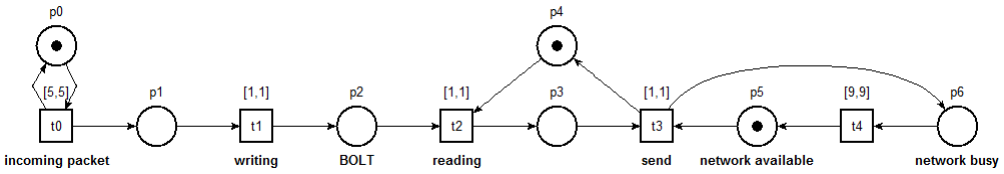
3 Simulate your Petri nets with TINA

No solutions.

4 Queue sizing and overflow management of BOLT

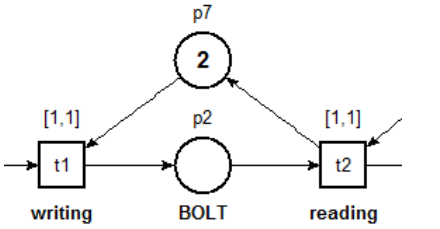
4.1 The nice and pretty deterministic world

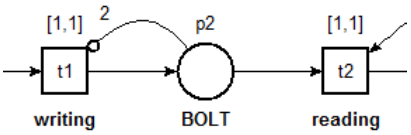
a) It should be rather easy to obtain the following model:



b) This net is obviously not bounded because it generates one packet every 5 time units on one end, and disposes of one every 10 time units only on the other end. Hence, packets pile up in the middle.

c) There are two easy ways of doing this. Either directly add a capacity to the `BOLT` place or use a weighted inhibitor arc from `BOLT` over the `writing` transition.





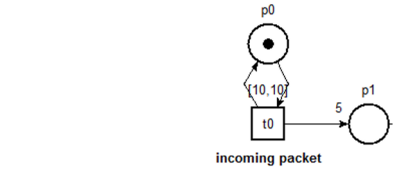
However, now BOLT is safe from overflow, but messages still pill up somewhere. In this case, "somewhere" is in place $p1$, which represents the sensor memory. Having this overflowing is not satisfactory either.

d) *No solutions.*

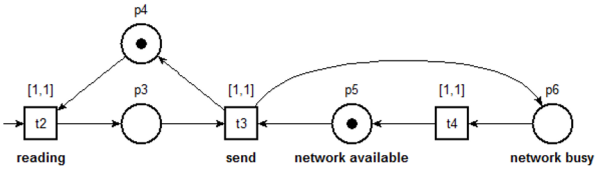
e) Option **c)** does not work because of the mechanism of delay reset in time Petri nets. When there are two tokens in `network busy` and `t4` fires, this creates an evolution of the enabling of `t4`, hence the delay is recomputed (i.e., the clock is reset, as explained in the lecture). Since the delay cannot be anything else than 9 time units, it starts over from there, regardless how long the second token has already spent in the place. This implies that the resource has in fact a cycle time of 9 time units, and not 5 as we wanted.

Furthermore, we will emphasis with the next question that solution **b)** is more flexible, as you can send 2 messages anytime during a 10 time units window, while for **a)**, you must wait 5 time units between two messages.

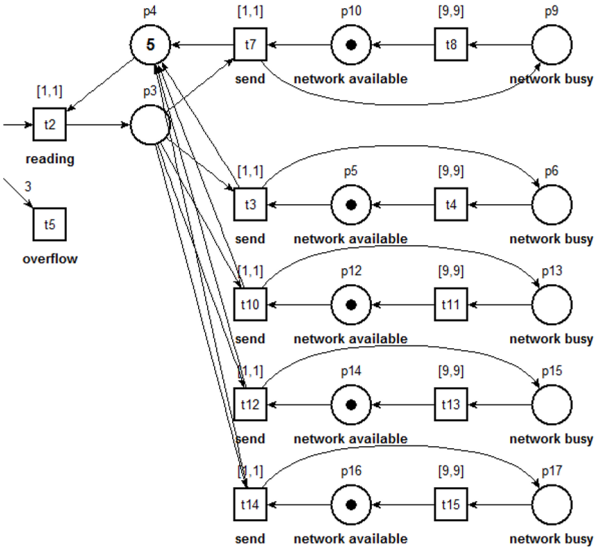
f) As mentioned before, option **b)** is more flexible, it can absorb bursts that **a)** cannot. The reason is that the communication processor cannot read out fast enough the messages from BOLT, one every two time units at the most. If you allow multiple reads (e.g., 5 tokens in p4), it makes options **a)** and **b)** equivalent.



Burst



a)



b)

4.2 Real-world is non-deterministic

a) *No solutions.*

b) $\Diamond\, t5 \equiv$ "Whatever happens, eventually transition $t5$ will fire."

The initial model verifies this property. But this is the opposite of what we want! It means there will always be an overflow of BOLT at some point, even in the best-case scenario...

c) We want that "Whatever happens, there is no overflow of BOLT". This can be verified with the LTL query "$\Box - t5$" evaluating to TRUE. This is equivalent to the CTL "$AG\, not(t5)$".

d) The place $p4$ directly correlates to the memory of the communication processor. Observe that places $p4$ and $p5$ always contain the same number of tokens together (as every transition either moves a token from $p4$ to $p5$, or the other way around).

e) *No solutions.*

f) First, observe that when we currently own the network resource (place $p7$), then the fastest speed of getting new packets (combining transition intervals from $t0$, $t1$, $t2$) matches the slowest speed of sending them out (transition interval $t3$). This means, that overflow can only happen if place $p7$ currently has no tokens. The longest time during which place $p7$

does not own any token is 50 (the upper limit of transition $t4$). Hence, our first piece of information is that we at least need as much space as messages can be generated during the time we don't have the network resource.

However, with a capacity of 25 (with the highest frequency of incoming packets being 2), it is still possible to overflow. Consider the worst case, where place $p2$ has 2 tokens at the time where the network resource is released. Now, 25 new messages can arrive until we get the network resource again (with two tokens still in place $p2$). Then, it again takes 2 time units until we can send the first message, during which a new message can come in that causes the overflow. In fact, it is possible that up to two messages can come in during that time, given a specific firing sequence of $t1$. This shows that we need a capacity of 27, which is two plus the number of messages that can be received during the longest time the network resource is not available.

g) Now, we need a capacity of 17 since 15 messages can be received during the time it takes for the resource to be available. This number can be verified using the LTL model-checker of TINA.