# CrashCourse — Time PetriNets

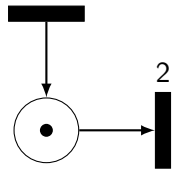
Tibor Schneider

TIK

# Time Petri Nets

Timer on Transitions, that restart when:

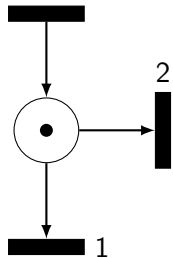
1. Transition becomes **active**
2. A token from any input place is **removed**



# Time Petri Nets

Timer on Transitions, that restart when:

1. Transition becomes **active**
2. A token from any input place is **removed**



# Tina: Time Ranges

1. $[a, a]$: Exactly after $a$ time units (like in the lecture).

# Tina: Time Ranges

1. $[a, a]$: Exactly after $a$ time units (like in the lecture).
2. $[a, b]$: Somewhere between $a$ and $b$ time units (at most $b$, at least $a$)

# Tina: Time Ranges

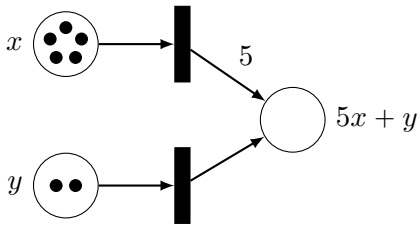
1. $[a, a]$: Exactly after $a$ time units (like in the lecture).
2. $[a, b]$: Somewhere between $a$ and $b$ time units (at most $b$, at least $a$)
3. $[a, \infty]$: At least $a$ time units, no upper bound.

# Tina: Time Ranges

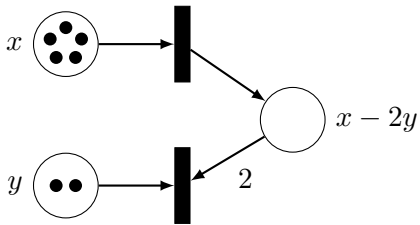
1. $[a, a]$: Exactly after $a$ time units (like in the lecture).
2. $[a, b]$: Somewhere between $a$ and $b$ time units (at most $b$, at least $a$)
3. $[a, \infty]$: At least $a$ time units, no upper bound.
4. $[0, \infty]$: The transition can fire anytime (just like in the normal Petri net).

Your turn to work!

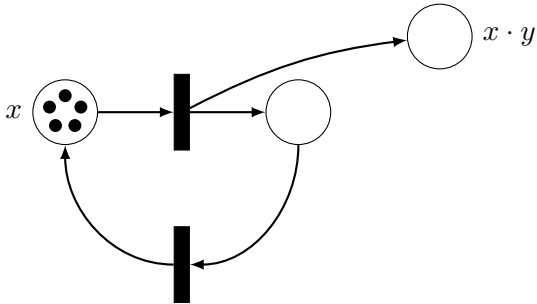
# Ex1 a) $5x + y$



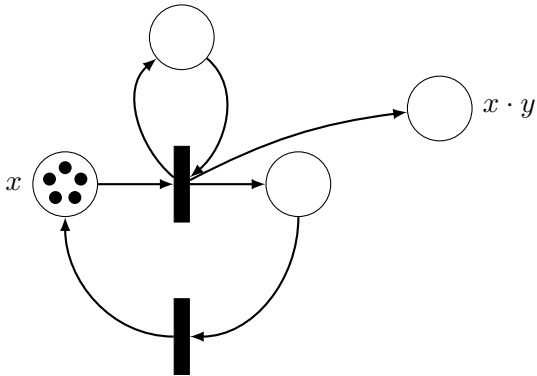
# Ex1 a) $x - 2y$



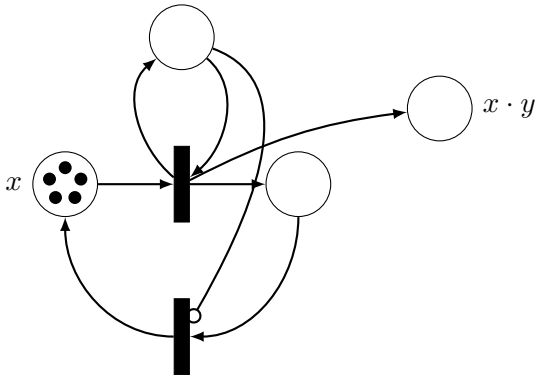
# Ex1 a) $x \cdot y$: Start by duplicating tokens in $x$



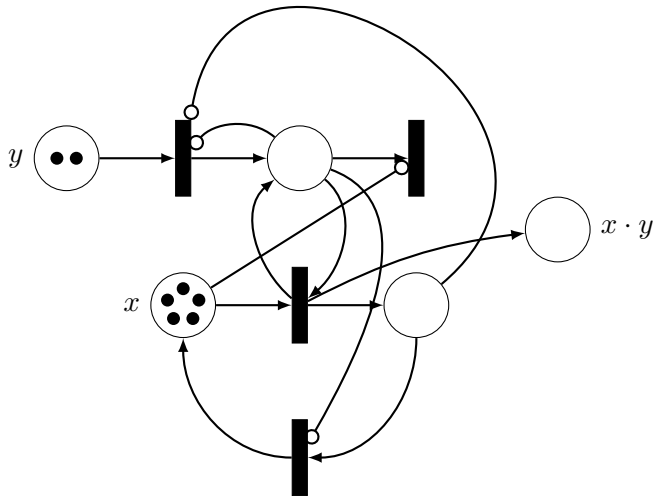
# Ex1 a) $x \cdot y$: Start by duplicating tokens in $x$



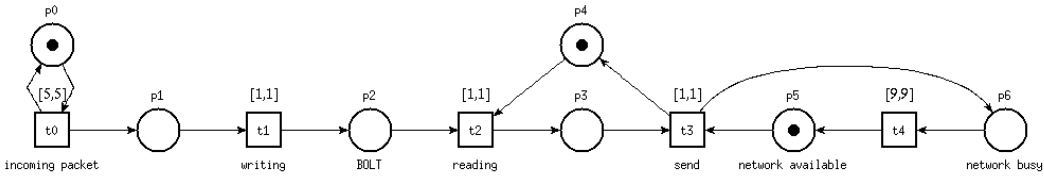
# Ex1 a) $x \cdot y$: Start by duplicating tokens in $x$



# Ex1 a) $x \cdot y$: Start by duplicating tokens in $x$

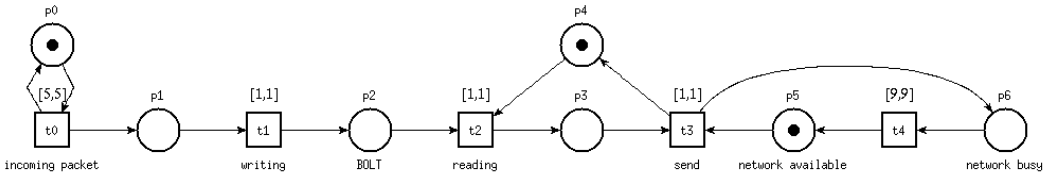


# Ex3.1 a)



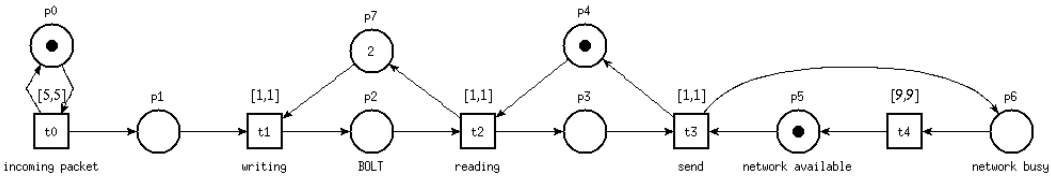

- one message every 5 time units → **t0**.
- Reading / writing from/to BOLT takes 1 time unit each → **t1** and **t2**
- Sending a message in the network takes 1 time unit → **t3**
- BOLT the network 10% of the time → **t4** (9 time units)

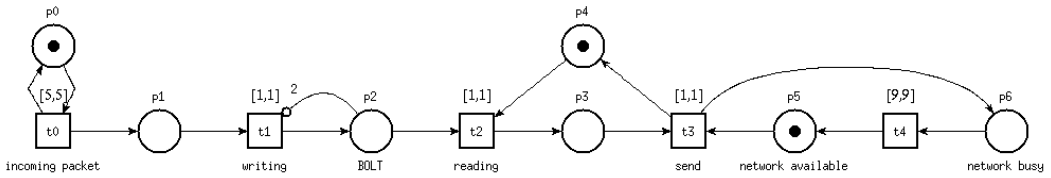
# Ex3.1 b)



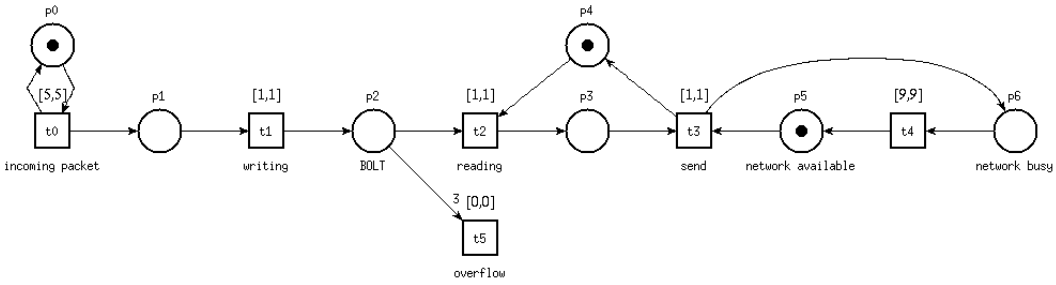

Network is not bounded!

# Ex3.1 c) BOLT has capacity of 2

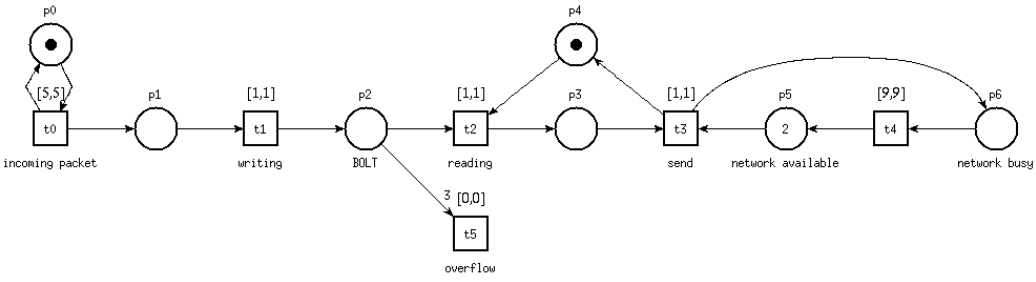


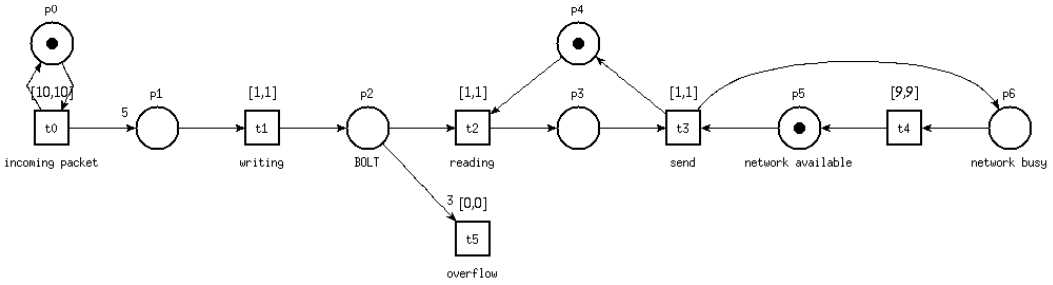
# Ex3.1 d) Overflow transition **t5**



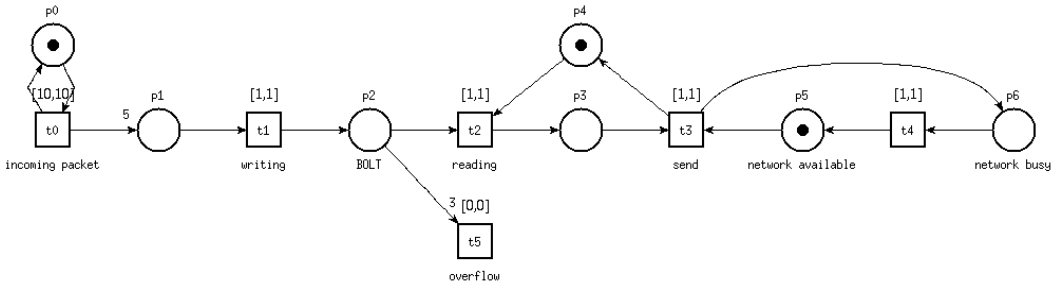
# Ex3.1 e) Why does adding another token to p5 solve the problem?



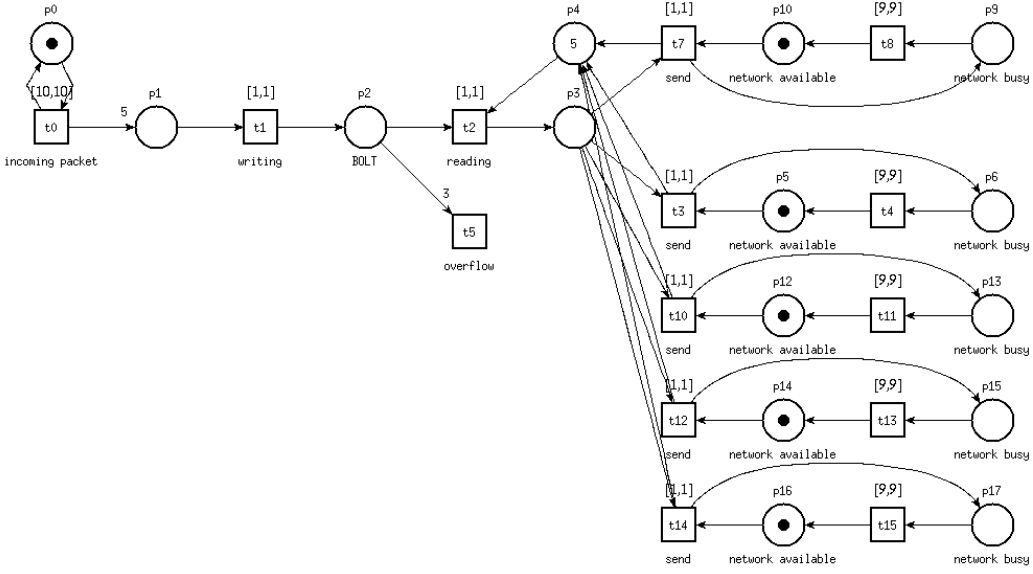
# Ex3.1 f) Make the input come in bursts



# Ex3.1 f) Reduce time on transition t4



# Ex3.1 f) duplicate the network multiple times



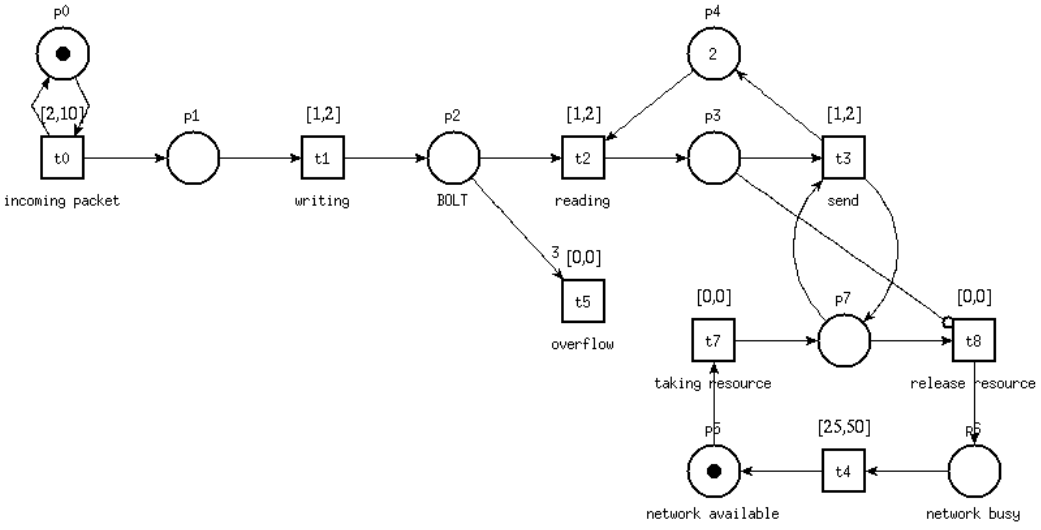
# Ex3.2 b) From LTL to CTL

$\diamond t5 \iff \mathsf{AF}\, t5 \iff$ No matter what happens, t5 will eventually fire.

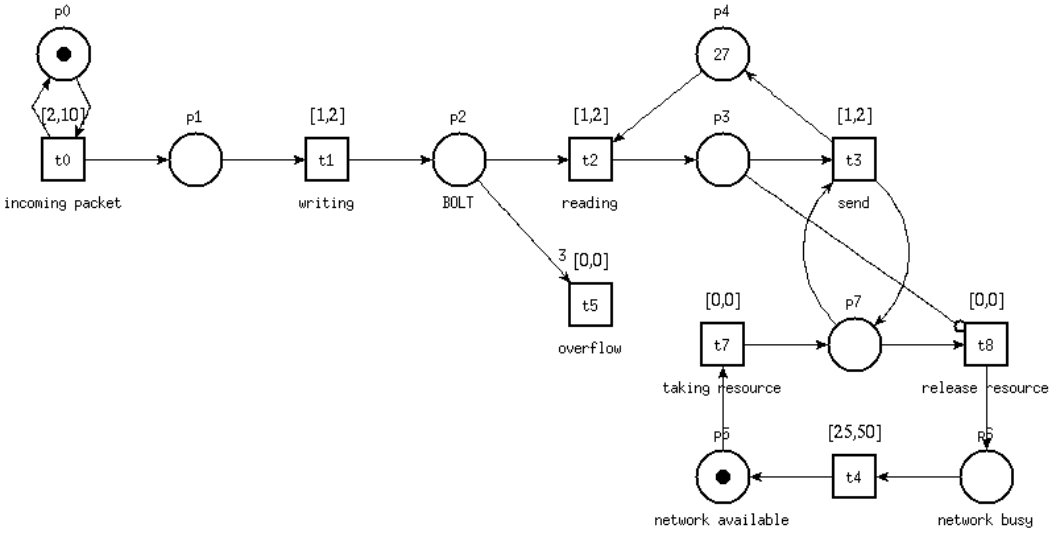
# Ex3.2 c) From Specification to CTL and LTL

No matter what happens, there is no overflow. $\iff$ AG $\neg t5$ $\iff$ $\Box \neg t5$

# Ex3.2 d) Memory Place p4



# Ex3.2 f) Why 27


p0
[2,10]
t0
incoming packet
p1
[1,2]
t1
writing
p2
BOLT
[1,2]
t2
reading
p4
27
p3
[1,2]
t3
send
3
[0,0]
t5
overflow
[0,0]
t7
taking resource
p7
[0,0]
t8
release resource
p6
network available
[25,50]
t4
R6
network busy