Model Checking LTL properties of High-Level Petri Nets with Fairness Constraints

Timo Latvala*
Helsinki University of Technology,
Laboratory for Theoretical Computer Science,
P.O.Box 9700, 02015 HUT, Finland
http://www.tcs.hut.fi/maria/

28th June 2001

*This research was financed by the Helsinki Graduate School on Computer Science and Engineering, the National Technology Agency of Finland (TEKES), the Nokia Corporation, Elisa Communications and the Finnish Rail Administration.

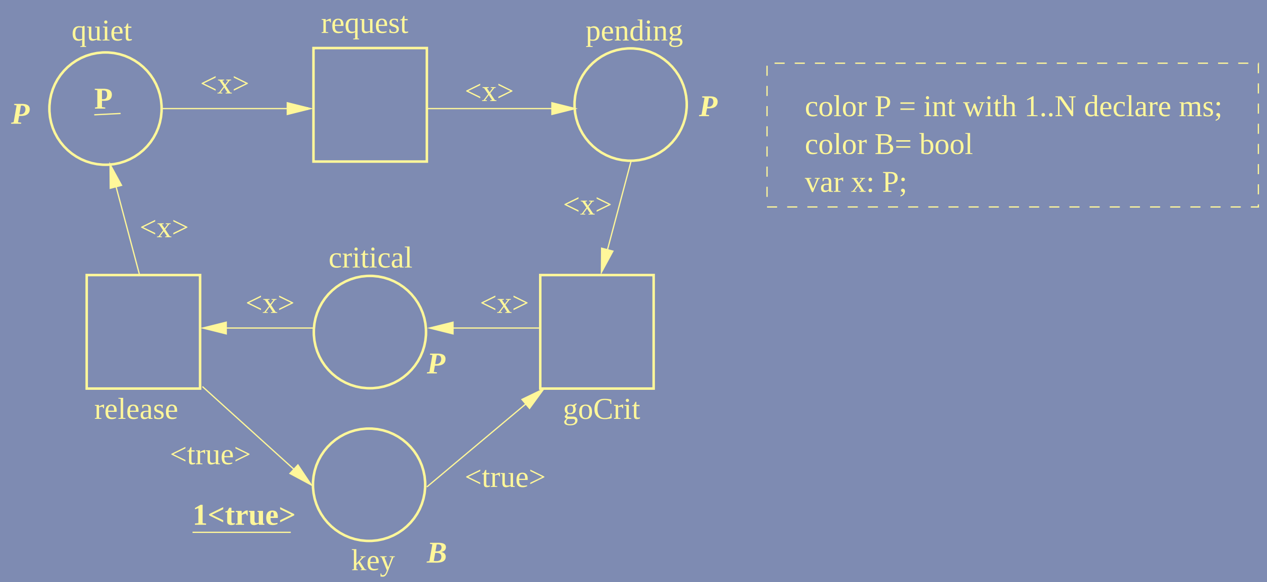
- Why is fairness important?
- The old solution
- A new approach
- Case study: Sliding window protocol
- Conclusions

Why is fairness important? (1/3)

- We usually distinguish between two classes of behavioural properties of distributed systems
  - Safety properties: "Something bad will never happen"
  - Liveness properties: "Something good will eventually happen"
- In many cases *liveness* properties cannot be proven without making some assumptions.
- *Fairness* is considered a reasonable and useful assumption

Why is fairness important? (2/3)

- Weak fairness: if an event is continuously enabled it will occur infinitely often
- Strong fairness: if an event is infinitely often enabled it will occur infinitely often
- Both weak and strong fairness can be expressed in LTL
- Weak fairness: $\Box\Diamond(\neg en \vee oc)$.
- Strong fairness: $\Box\Diamond(en) \Rightarrow \Box\Diamond(oc)$



- Accessibility does not hold if we do not assume that the transition *goCrit* is strongly fair w.r.t. each instance.

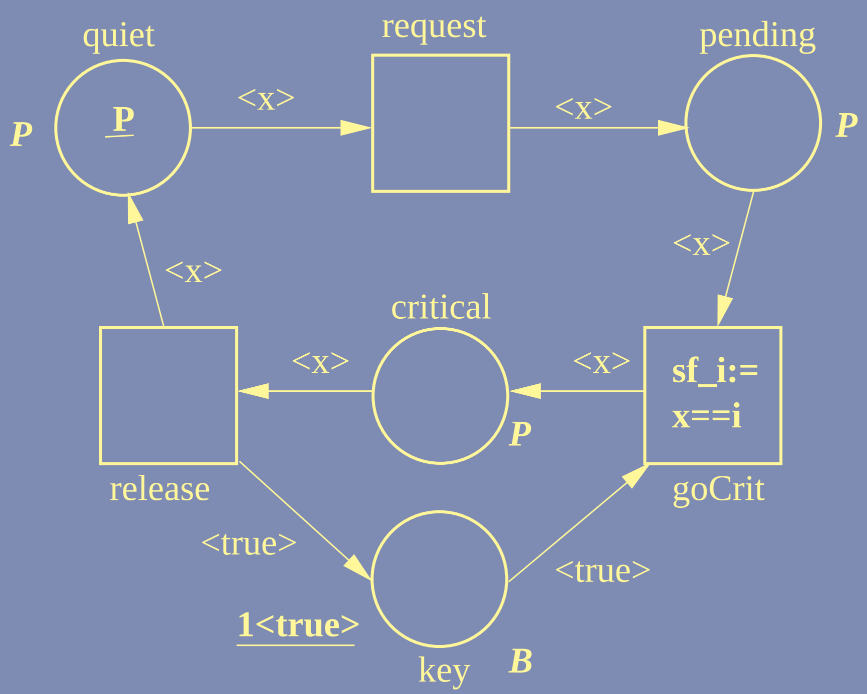
- We remember that fairness can be expressed in LTL
- Thus we verify the formula “$fairness \Rightarrow property$”
- Sometimes an explicit scheduler has to be modelled, in order for this to work.

- Model checking LTL is *PSPACE-complete* in the size of the formula
- May require changes in the model (adding scheduler)
- Adding scheduler can reduce the concurrency in the model, affecting some partial order methods.

A *fair CPN (FCPN)* is a triple $\Sigma_F = \langle \Sigma, WF, SF \rangle$, where Σ is a CPN, and $WF = \{wf_1, \ldots, wf_k\}$ is a set of weak fairness functions, where wf_i is function from transitions to boolean valued expressions. SF is the corresponding set of *strong fairness* functions.

- Fairness is made a part of the model
- The fairness functions singles out the instances which are to be treated fairly.

Example



color P = int with 1..N declare ms;
color B= bool
var x: P;

A **fair Kripke structure (FKS)** is a quintuple $K_F = \langle S, \rho, s_0, \mathcal{W}, \mathcal{S} \rangle$, where S is a set of states, $\rho \subseteq S \times S$ is a transition relation and $s_0 \in S$ is the initial state.

- The fairness requirements are defined by a set of *weak fairness* requirements $\mathcal{W} = \{J_1, J_2, \ldots, J_k\}$ where $J_i \subseteq S$, and a set of *strong fairness* requirements, $\mathcal{S} = \{\langle L_1, U_1 \rangle, \ldots, \langle L_m, U_m \rangle\}$ where $L_i, U_i \subseteq S$.

- An execution is an infinite sequence of states $\sigma = s_0 s_1 s_2 \ldots \in S^\omega$, where s_0 is the initial state, and for all $i \geq 0$, $(s_i, s_{i+1}) \in \rho$.

- Computations, i.e. fair executions of the system, are sequences that obey the fairness requirements $\bigwedge_{i=1}^{k} Inf(\sigma) \cap J_i \neq \emptyset$ and $\bigwedge_{i=1}^{m} (Inf(\sigma) \cap L_i = \emptyset \vee Inf(\sigma) \cap U_i \neq \emptyset)$.

- The constraints of FKS correspond to Generalised Büchi automata and Streett automata acceptance conditions respectively.

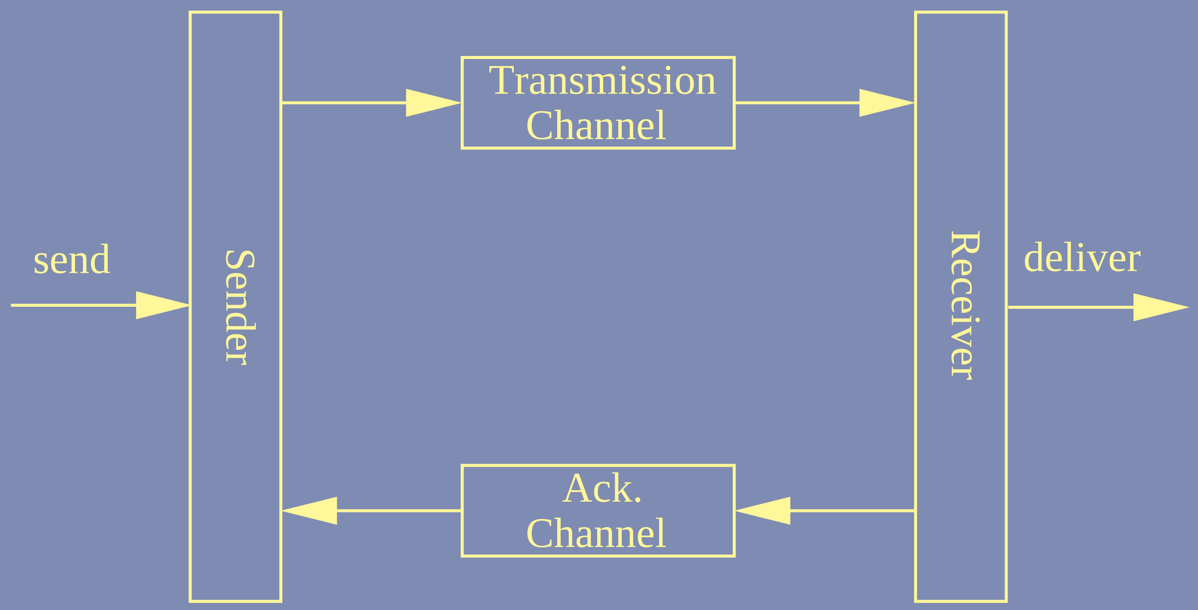
- The new procedure combines emptiness checking for Büchi and Streett acceptance conditions

- We try to avoid using the more time consuming Streett emptiness checking procedure if possible.

- The procedure has been implemented in the Maria tool.

- Emerson and Lei: Fair-CTL model checking
- Knesten, Pnueli and Raviv: Symbolic Fair LTL model checking
- Latvala and Heljanko: LTL model checking for P/T nets with fairness constraints on the transitions.

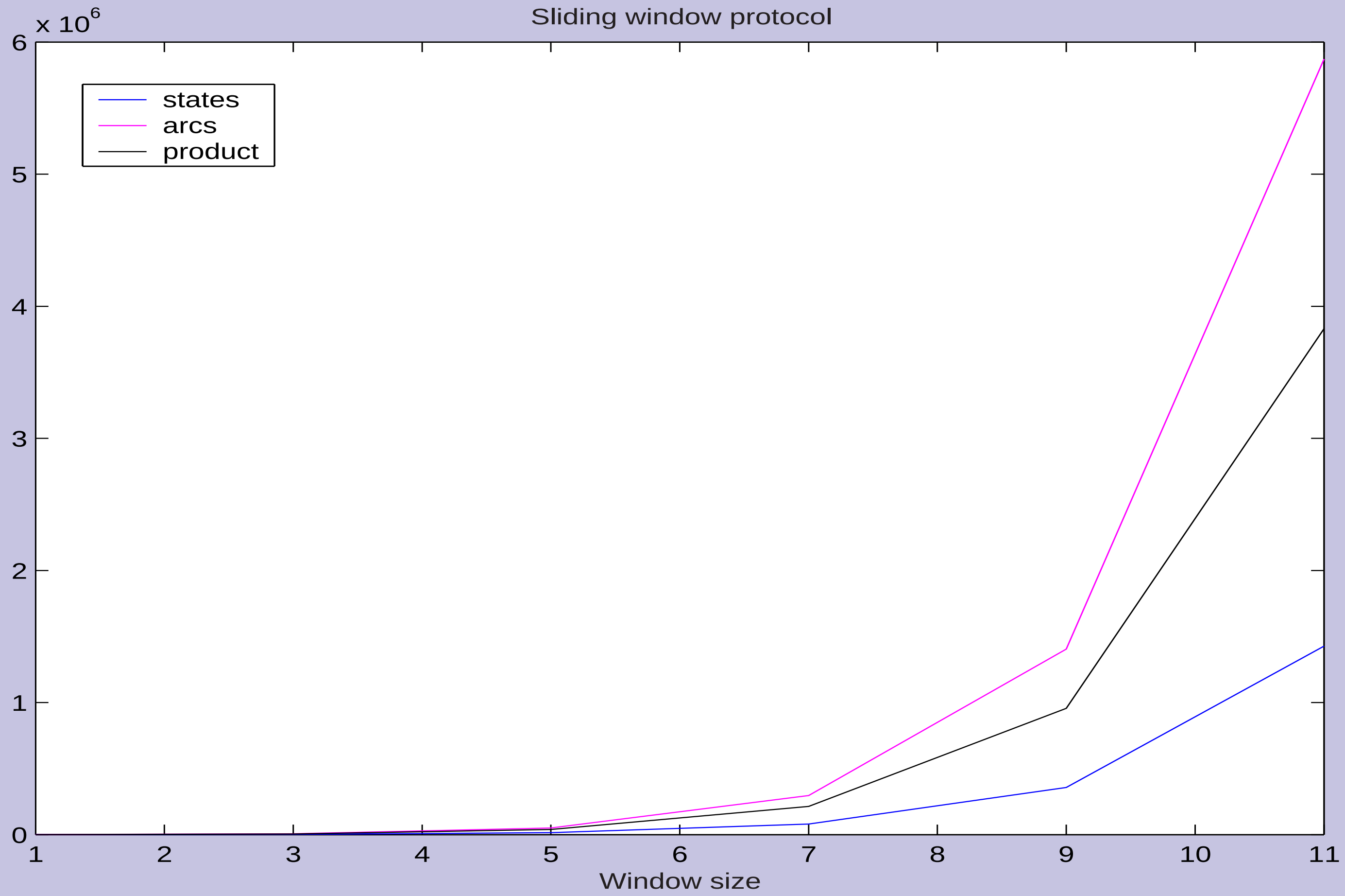
A sliding window protocol



- Provides reliable transmission over an unreliable medium
- This version is due N.V. Stenning
- The model follows closely the model presented by R. Kaivola
- We wish to verify that as many targets should be delivered to the target as are read from the data source. This holds only under a fairness constraint.

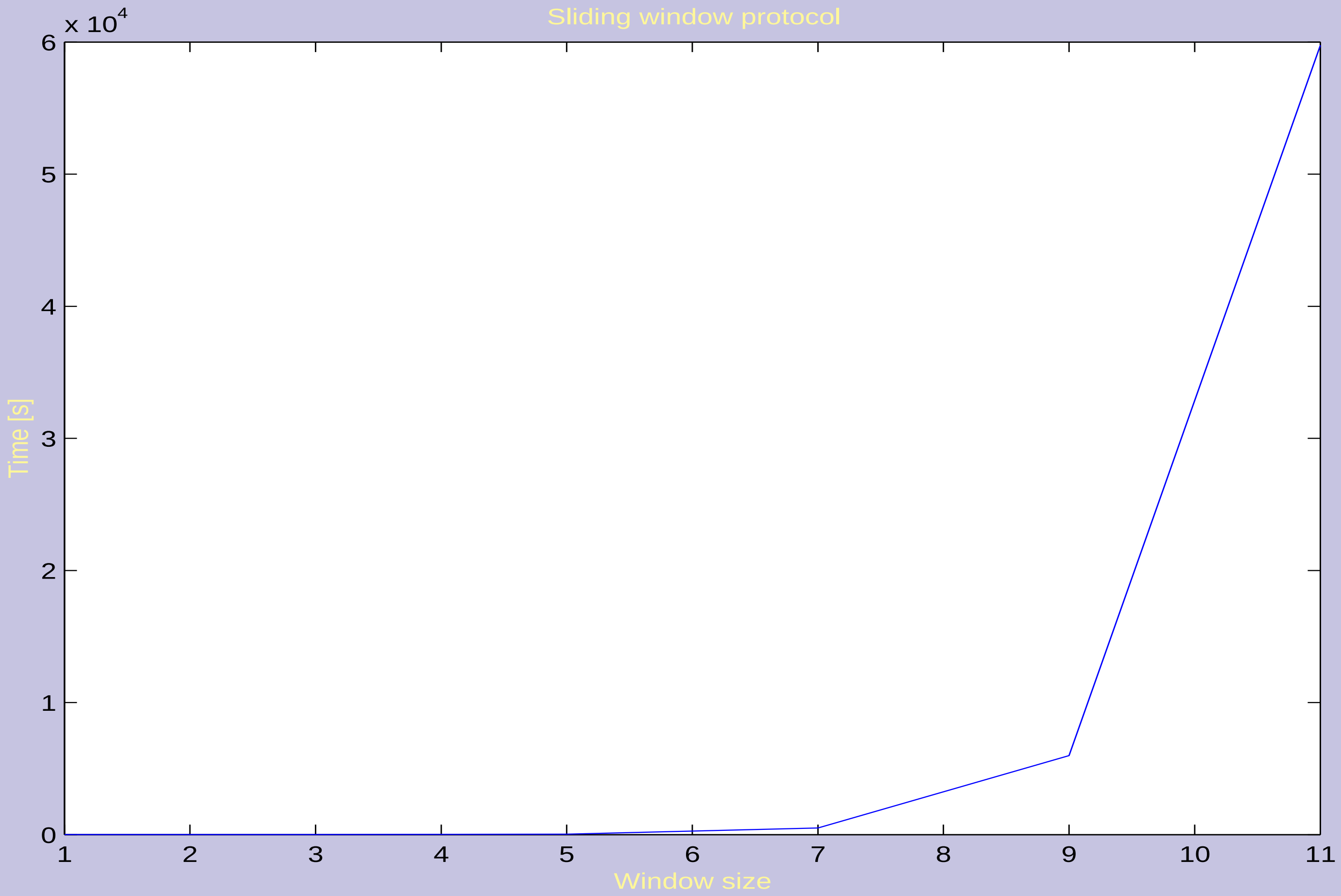
- Using the powerful type system and algebraic operations of Maria, modelling is straightforward.
- Complete model: 12 places and 9 high-level transitions.
- Strong fairness constraints on receive-transitions of the sender and the receiver processes.
- A weak fairness constraint is needed on the receiver side to guarantee progress in the sequential parts.

Results



Sliding window protocol
× 10^6
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
10
11
Window size
states
arcs
product

Results



Sliding window protocol
x 10^4
6
5
4
3
2
1
0
Time [s]
1
2
3
4
5
6
7
8
9
10
11
Window size

- We can do LTL model checking on high-level Petri nets with versatile fairness constraints on the transitions
- The procedure is much more efficient than specifying fairness as part of the property to be verified
- The procedure has been implemented in the Maria tool and found to scale fairly well
- Effect on partial order methods?