Non-Ambiguous Petri Nets and their Application to the Modeling of ATM Switches

–

Bruno Gaujal Günther Siegel Alain Jean-Marie

SLOOP **MISTRAL**
INRIA Sophia-Antipolis
CNRS I3S
UNSA

WODES'98 Cagliari
26 August 1998

Objectives

- Introduce and define formally a new class of timed Petri Nets with
  - ∗ true competition
  - ∗ simultaneous events
  - ∗ priorities
  - ∗ well defined unique evolution

  $\Longrightarrow$ **Non-Ambiguous Petri Nets**
- Discuss its aptness at
  - ∗ detail modeling of complex systems
  - ∗ fast “equational” simulation
  - ∗ parallel/mixt simulation

  $\Longrightarrow$ **ATM Switch Models**

Overview

- Evolution Equations and Simulation
- Timed Petri Nets and their Semantics
- Non Ambiguous Petri Nets
- Discussion
- Models of ATM Networks

Equational Techniques

An **equational simulation** consists in using recursive evolution equations for computing the successive states of a system.

If the n-th state is $X(n)$ and there exists a function F such that

$$X(n+1) = F(X(n)) .$$

A simulation uses this equation to compute the states: $X(0), X(1), \cdots X(n), \cdots$ from which the statistical measures can be derived.

The function F can be computed so far for

– Event Graphs, arbitrary times, FIFO,

– Free Choice Nets, integer times.

Timed Petri Nets with Routing

$\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{A}, M_0, \Phi, H, \Pi)$,

- $(\mathcal{P}, \mathcal{T}, \mathcal{A}, M_0)$ standard Petri Net,
- $\Phi = \{\phi_i, i \in \mathcal{T}\}$ a the set of constant, integer firing times.
- $H = \{H_{p,i}(n,t), p \in \mathcal{P}, i \in p^\bullet, n \in \mathbb{N}, t \in \mathbb{N}\}$ a set of routing functions.

$$H_{p,i}(n,t) = \#\{\text{ number of tokens routed to transition } i \in p^\bullet \text{ when } n \text{ tokens are present in place } p \text{ and at time } t.$$

- $\Pi = \{\pi_p(i), p \in \mathcal{P}, i \in p^\bullet\} \subset \mathbb{N}$ a the set of priorities for the output transitions of each place.

Semantics

enabling A transition q is *enabled* at time t if for each p in ${}^\bullet q$, p contains a token which is routed to p at time t.

token reservation Tokens participate to at most one firing.

firing sequence At any stage in its evolution, the net performs the following sequence:

- Fire the next transition to be enabled under the current marking.
- If several transitions are enabled under the current marking, fire one of the transitions with the highest priority.

A "behavior" is correct if there exists a firing sequence which realizes it, and complies with the rules above.

Orderings among transitions

Priority Ordering.
If two transitions, t_1 and t_2 are output transitions of a place p, and if t_1 has higher priority than t_2, then

$$t_1 \ <_p \ t_2$$

Timing ordering.
If two *immediate* transitions t_1 and t_2 have a place between them, then

$$t_1 \ \prec \ t_2$$

Non-Ambiguous Petri Nets

A NAPN is a Timed Petri Net with routing with the following properties:

(C_1) The Petri Net does not contain any cycle made of immediate transitions $\Longrightarrow$ $\prec$ is a partial order on $\mathcal{T}$.

(C_2) For each place p,

- $<_p$ is total on the subset $p^\bullet$,
- all orders $\{<_p, p \in \mathcal{P}\}$ are compatible $\Longrightarrow$ their union $<_\mathcal{P}$ is a partial order on $\mathcal{T}$.

(C_3) The orders $\prec$ and the order $<_\mathcal{P}$, are compatible.

Simulation Engine

```
Behave(N)
 Order 𝒯 with a linear extension of ≪.
 for i from 1 to T do X_i(0) := 0 od
 for n from 1 to N do
  for i from 1 to T do X_i(n) := X_i(n-1) od
  for i from 1 to T do
    for all p in •i do
      M_p := Σ_{j∈•p} X_j(n - φ_j) - Σ_{j∈•p} X_j(n)
      A_p := H_{p,i}(M_p, n)
    od
    X_i(n) := min_{p∈•i}{A_p} + X_i(n)
  od
 od
```

Where:

- $X_i(n)$ (behavior): number of firings *initiated* at transition i at time n
- A_p: the number of tokens which are available in p for i at current time.

Results

Theorem 1. *The algorithm* `Behave(N)` *computes a correct behavior of the net up to time* N.

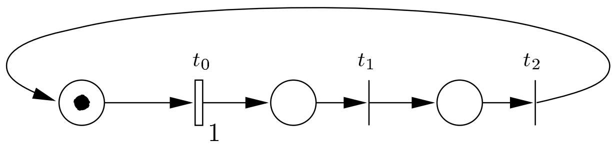
Corollary 1. *A Non-Ambiguous Petri Net has a unique correct behavior.*

Remark 1. *At each time step, each transition is "considered" only once.*

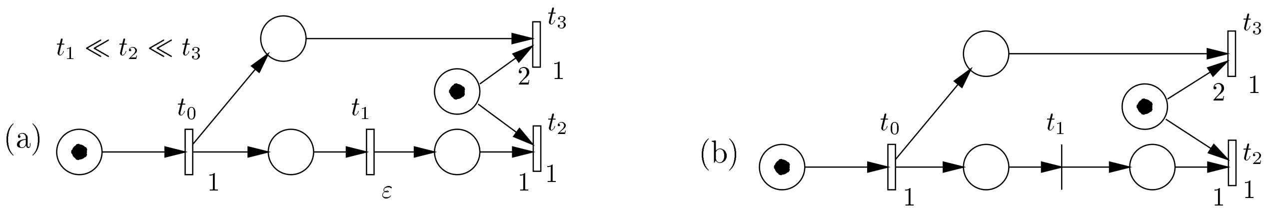
Remark 2. `Behave(N)` *is a simulation of the net, of the time driven family.*

Discussion

- The order in which the variables $X_i(n)$ are computed is indeed relevant.



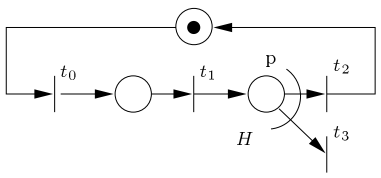
- Immediate $\neq$ almost immediate



(a) t_3 fires $\forall \varepsilon > 0$

(b) t_2 fires

- Immediate cycles could be tolerated



Application of NAPNs

- Perform a *detailed* modeling of an ATM switch
- Cell level / Virtual Circuit level
- Assess
  * Power of modeling
    express basic constructs of ATM switches with PNs
    construct a complex model out of building blocks
  * Power of simulation
    is equational simulation efficient?
    simulation of larger models
    parallel simulation
    mix equational and conventional simulation

Model of ATM Switches

Petri nets for the building blocks of the ATM switch fabric.

- Leaky Bucket/Token Bucket/Traffic shaping devices
- Backpressure mechanisms
- Jitter control
- Scheduling/Traffic classes (UBR, CBR, ABR)
- Source Models and Random Delays (Finite Discrete, Geometric, Markov Modulated, etc.)

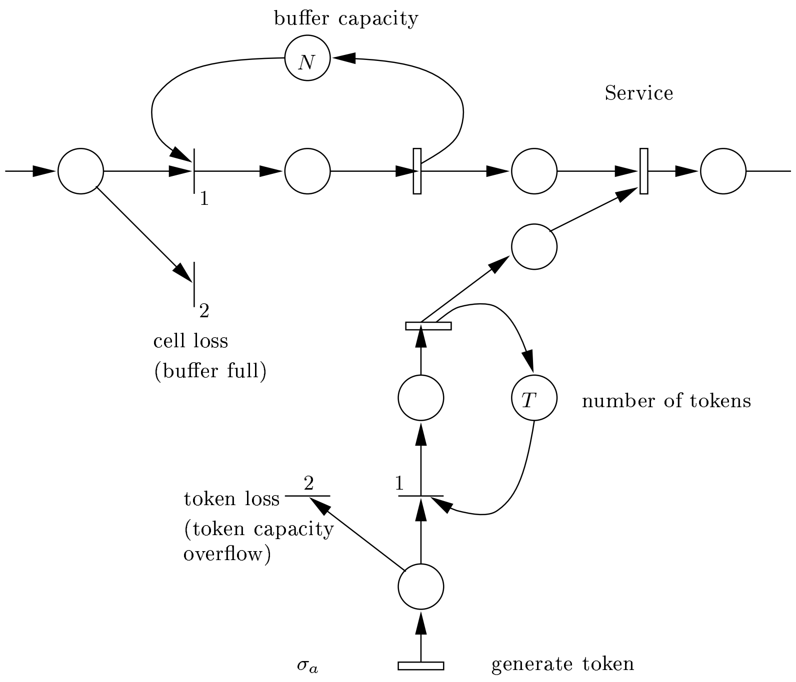


Figure 1: Leaky Bucket Model

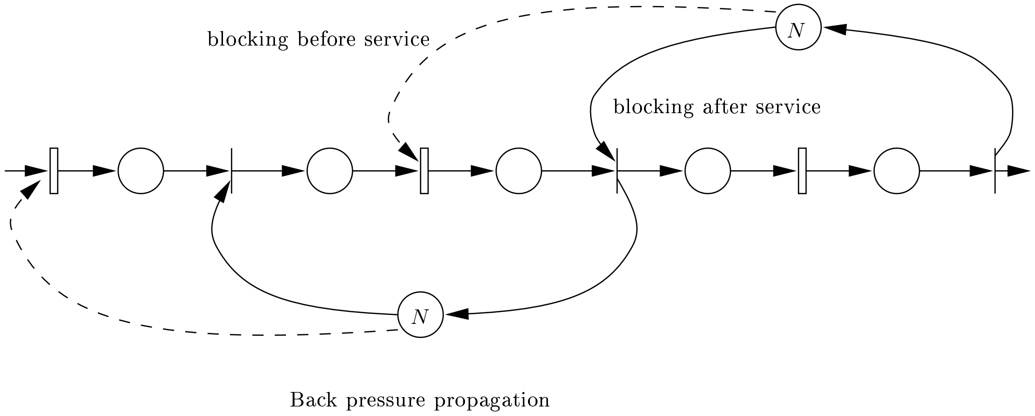


Figure 2: Back Pressure Protocol

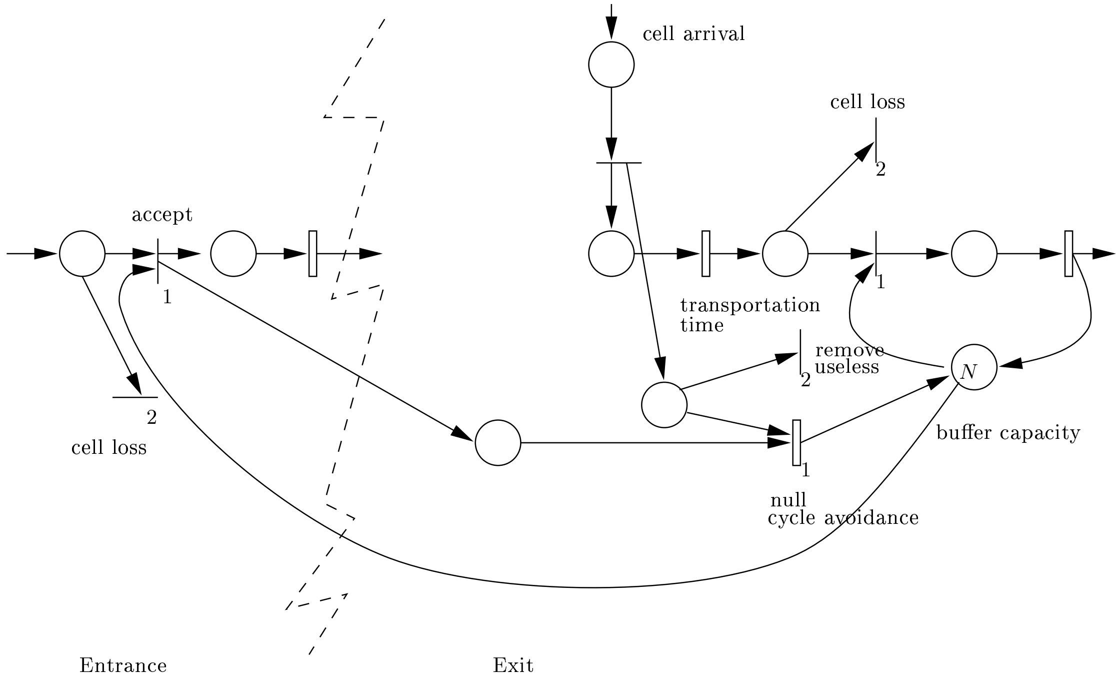


Figure 3: Admission control based on a threshold in output buffers

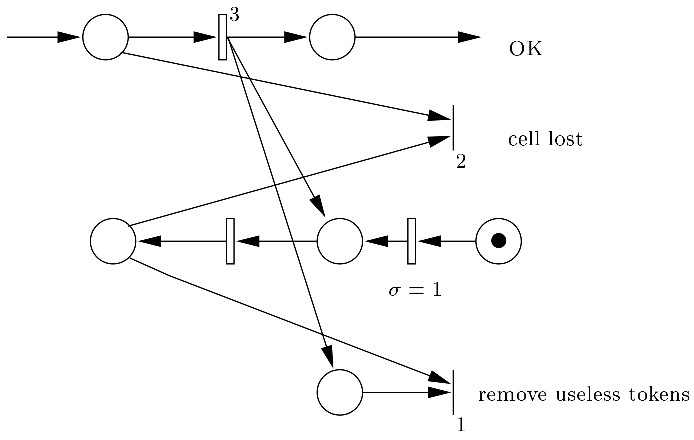


Figure 4: Jitter Control

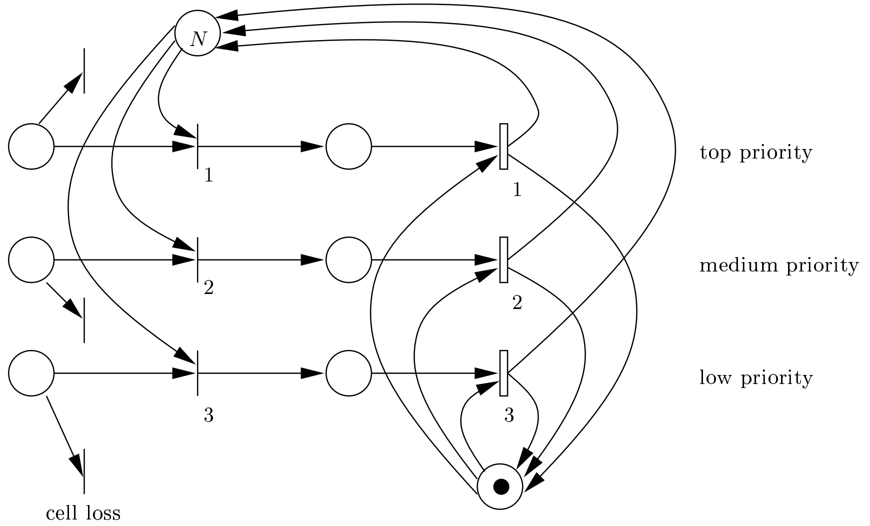


Figure 5: Three different traffic in a single finite queue with three levels of priority.

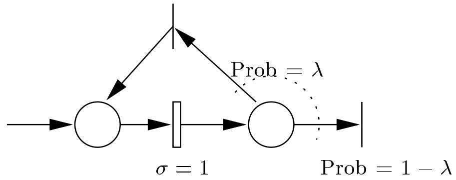


Fig. a

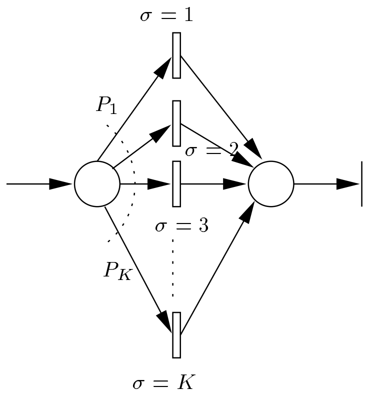


Fig. b

Figure 6: Construction of discrete firing times transitions

Model of ATM Network

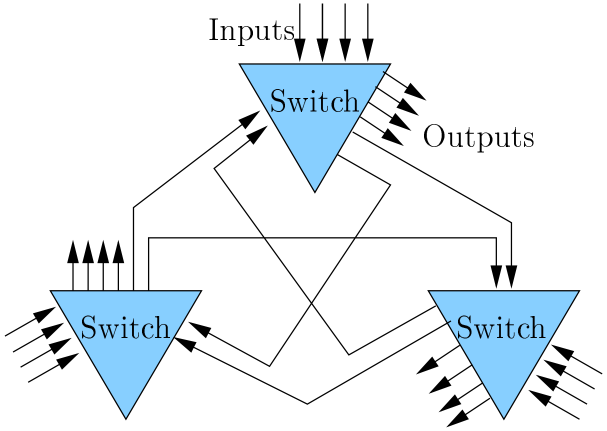


Figure 7: General view of the network

Three 6×6 swithes with:

– two classes of traffic (CBR and UBR),

– input streams are regulated by *leaky buckets*,

– admission controled by a *back pressure* protocol,

– *time out* losses are possible in the output buffers.

Model of ATM Network (*continued*)

- approx. 1500 places and transitions.
- sources modelled as independent objects, with event-driven evolution (mixt simulation)
- distribution on 3 processors
- several experiments with different offered loads

Experiments

Comparison between the mixed approach with a pure discrete event simulation within PROSIT:

Input rate	Mixed	Pure
$\lambda_0 \times 2$	875	12490
λ_0	140	9580
$\lambda_0 \div 10$	26	4098
$\lambda_0 \div 100$	14	599

Comparison between sequential and distributed simulations:

Input rate	$\lambda_0 \div 100$	$\lambda_0 \div 10$	λ_0	$\lambda_0 \times 2$
Sequential simulation	14	26	140	875
Distributed simulation	44	68	125	387
Distributed simulation with buffering	44	64	97	359

Conclusion

+ A class of PNs with a fairly good power of modeling, and a fairly simple simulation engine
+ Implementation in several forms
+ Mixt and distributed simulation

- Need for a better understanding of dynamics
- Need for transformation/simplification of models
- Need for theoretical results (closed nets, deterministic or saturated inputs)
- What happens when timing goes to 0?