Petri Nets with Time

Adding *Time* to Petri Nets

- Variation 1: **Transitions** have a delay time; firing takes a non-zero time from enabling. Time may be bounded from above or below.
- Variation 2: **Places** have a delay time: A token must dwell on a place a certain amount of time (determined by the place) before becoming usable in firing.
- Variation 3: Like 2, but **Tokens** have a delay time.

Variation 1 is Prevalent

- Ramchandani, MIT PhD Thesis, 1974.
- Richard Merlin, UCI Thesis, 1974.
- Berthomieu & Diaz, IEEE TSE, **17**, 3, pp. 259-273 1991.

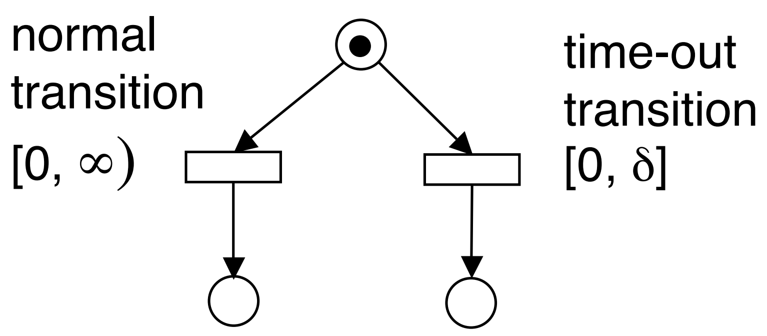
Berthomieu & Diaz

- A **Time Petri Net** is like a Petri Net with a time interval on each transition:

  $[t_1, t_2]$ or $[t_1, \infty)$

- From the time the transition is enabled, it cannot fire before t_1 and *must* fire by t_2 (unless disabled by firing another transition).

Example: Representing Time-out



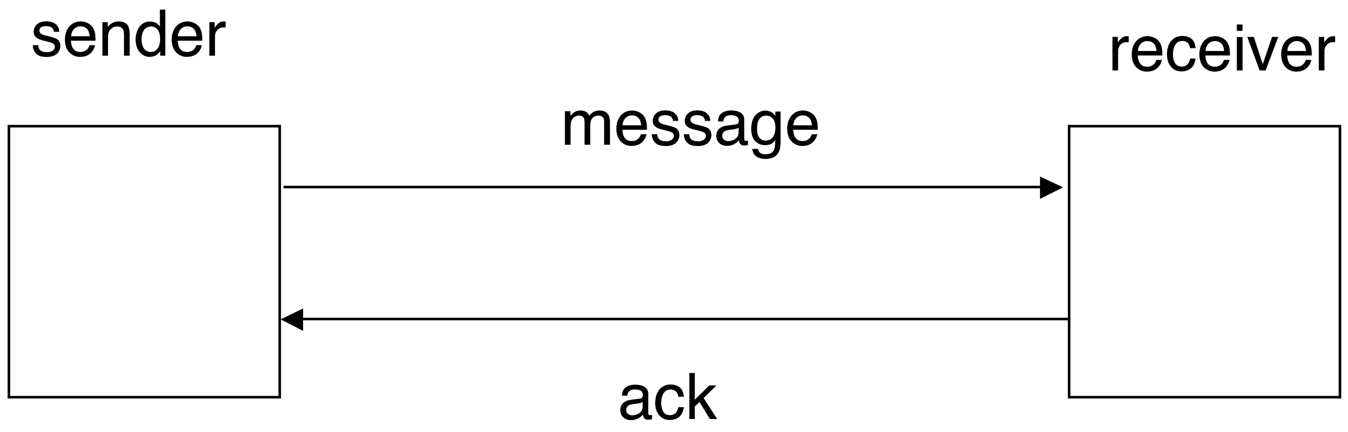
If the normal transition has not fired by time δ after being enabled, then the time-out transition will fire and the normal transition will be disabled.

Berthomieu & Diaz

IEEE Trans. on Software Eng., 1991, pp. 259--273

- Give sufficient conditions for boundedness for Time Petri Nets with rational time bounds.

Example: Sending messages between two sites.



Problem

- A message sent by a sender could get **lost or be garbled.**

- Thus the receiver must ack each message.

- If the ack is not received in a specified time, the transmission is regarded as having **timed out** and the sender must resend.

- The **ack** could also get lost or be garbled.

Problem, continued

- The sender might decide to **resend** although the original message has only been **delayed, not lost**.
- How can the receiver tell whether an incoming message is new or just a retransmission of an earlier message?

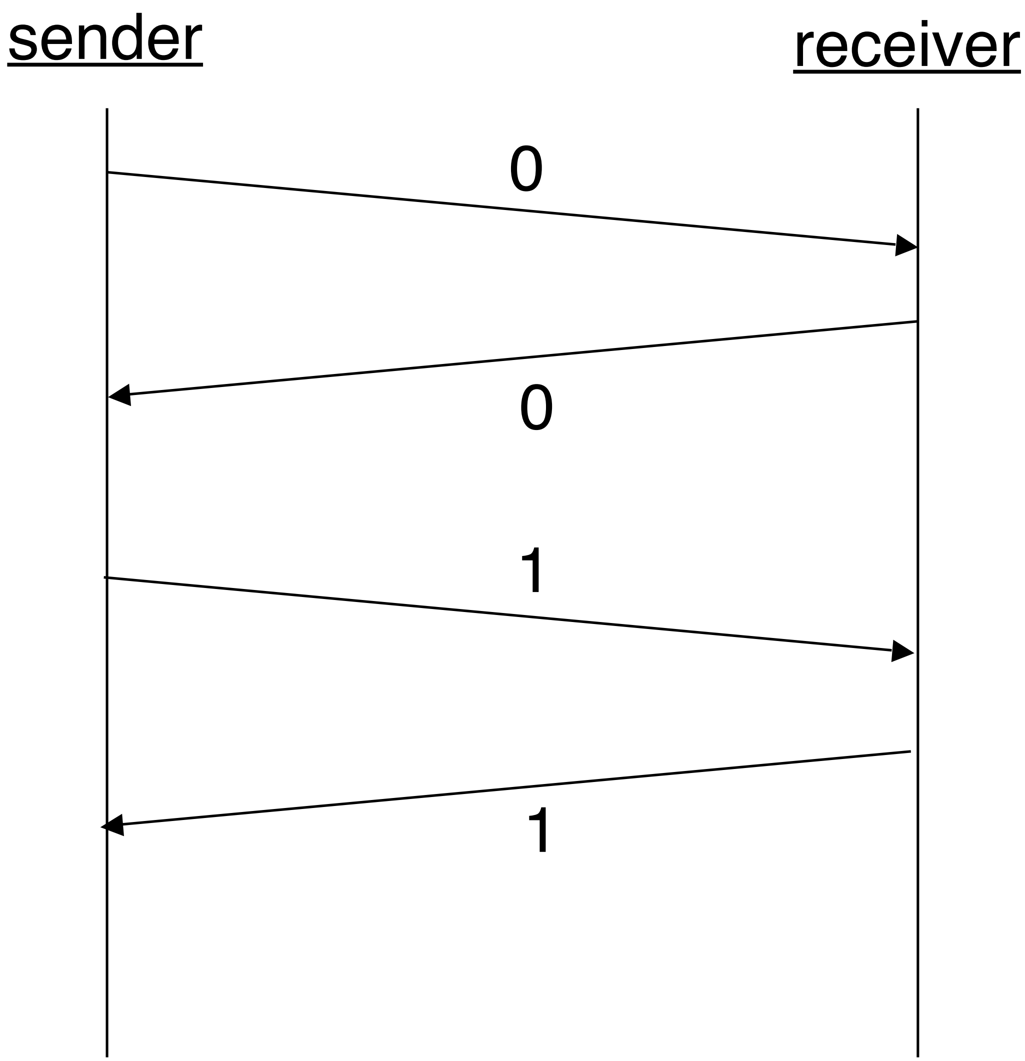
A Solution

- Each message is uniquely **timestamped** by a sequence number.
- The receiver only accepts and acknowledges the next number in sequence, not the replay of an earlier number.
- The acknowledgment indicates the number of the message being acknowledged.

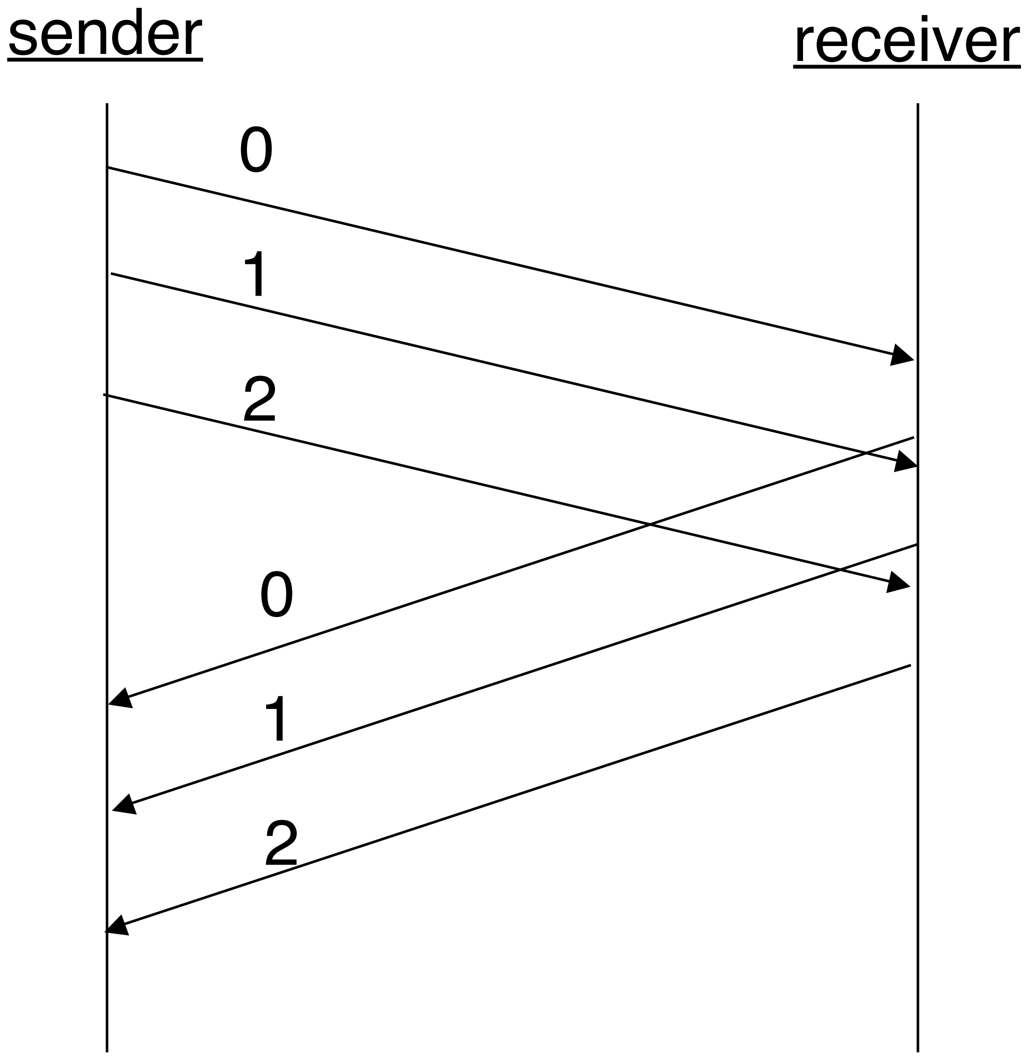
Problem & Fix

- The set of timestamps is not bounded.
- Alternating bit protocol (ABP):
  - Use only 0 and 1 as numbers.
  - Sender sends 1 only when 0 has been successfully acknowledged.
  - Sender sends 0 only when 1 has been successfully acknowledged.

Only one message can be in transit at a time.



More general possibilities (not ABP)



ABP Summary

- A message can be sent at an arbitrary time.
- Once sent, the message can be received or lost, within a bounded time.
- If the message is received, an ack is sent.
- The ack can be received or lost.
- If an ack is not received in a bounded time, the message is resent.
- If a resent message is received, it is ack'ed.

Pseudo-Code

(from http://www.fmi.uni-stuttgart.de/szs/teaching/ws0405/nets/nets.pdf)

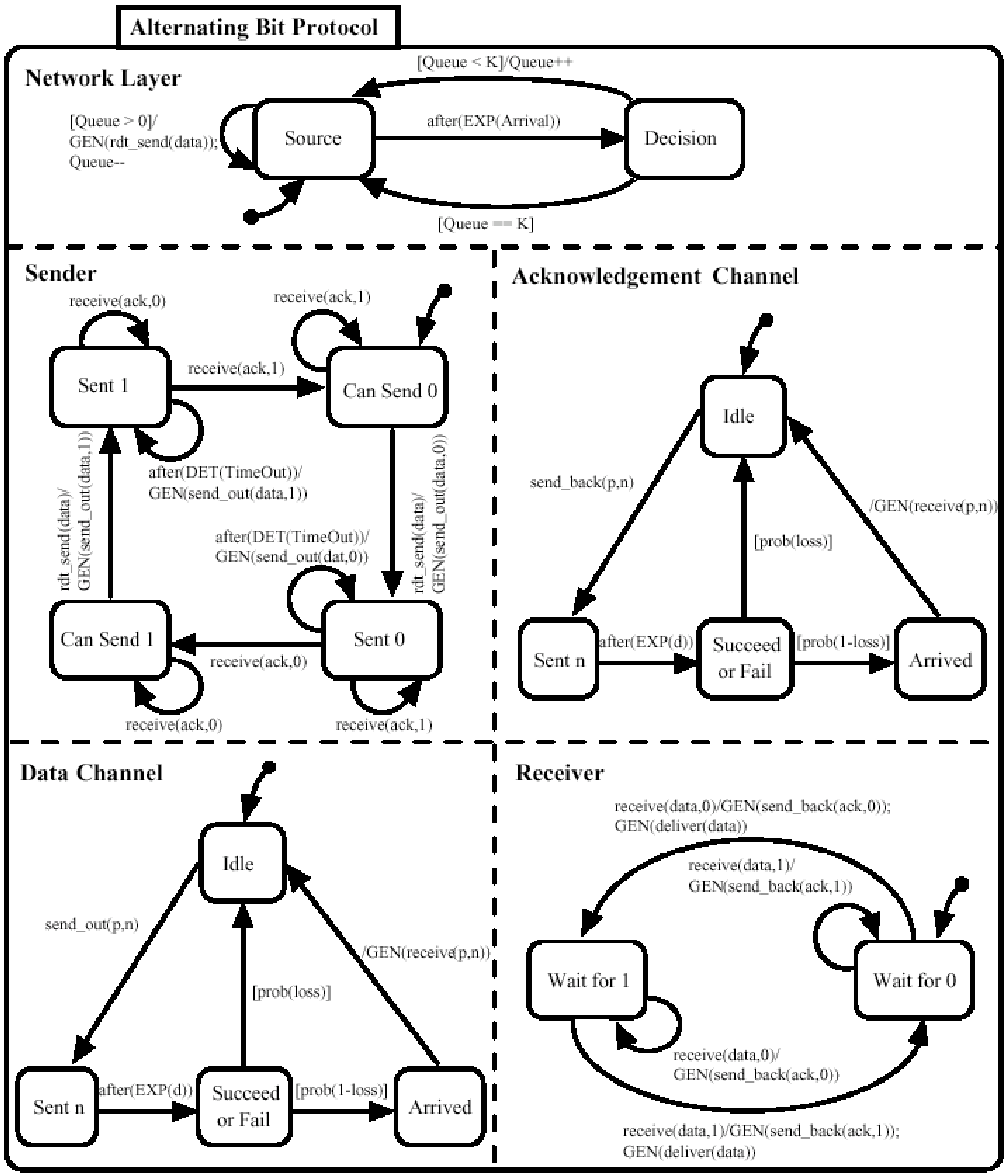
Assume that send and recv are primitives for sending and receiving a single piece of data over a channel.

```
void abp_send () {
  i = 0; s = 0;
  while (true) {
    i = i+1; s = !s;
    while (true) {
      send (data[i],s);
      wait (timeout);
      if (recv(ack) == s) break;
    }
  }
}
```

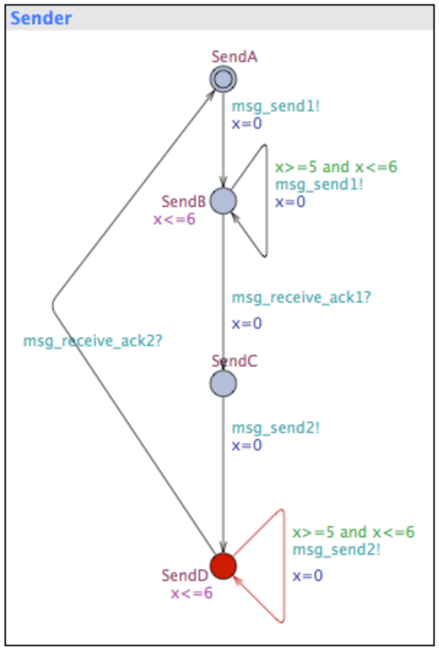
```
void abp_recv () {
  i = 0; r = 0;
  while (true) {
    i = i+1; r = !r;
    while (true) {
      recv (c,b);
      send (b);
      if (b == r) { data[i] = c; break; }
    }
  }
}
```

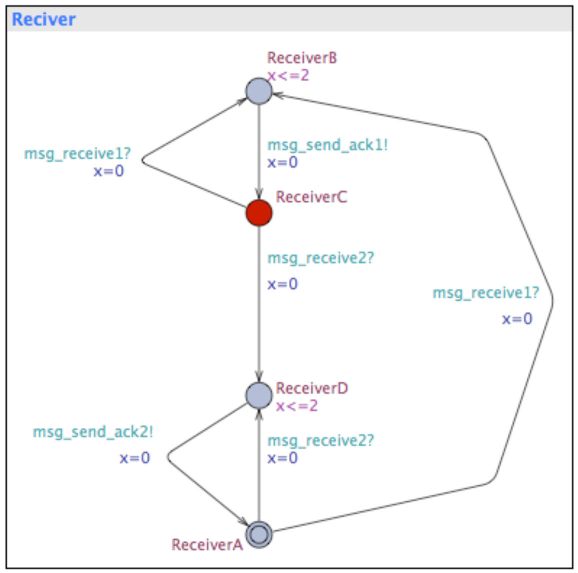
State Chart Version of ABP

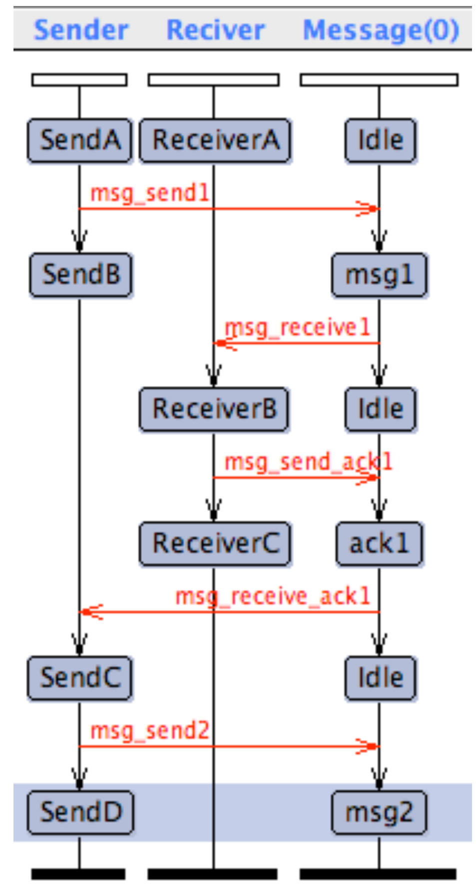
(from:http://ls4-www.cs.uni-dortmund.de/home/thummler/papers/WOSP00.pdf)



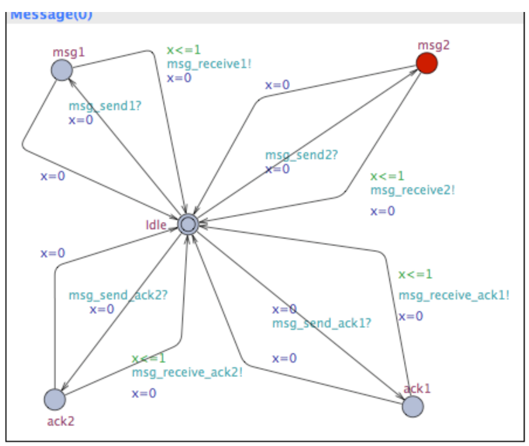
ABP in Uppaal





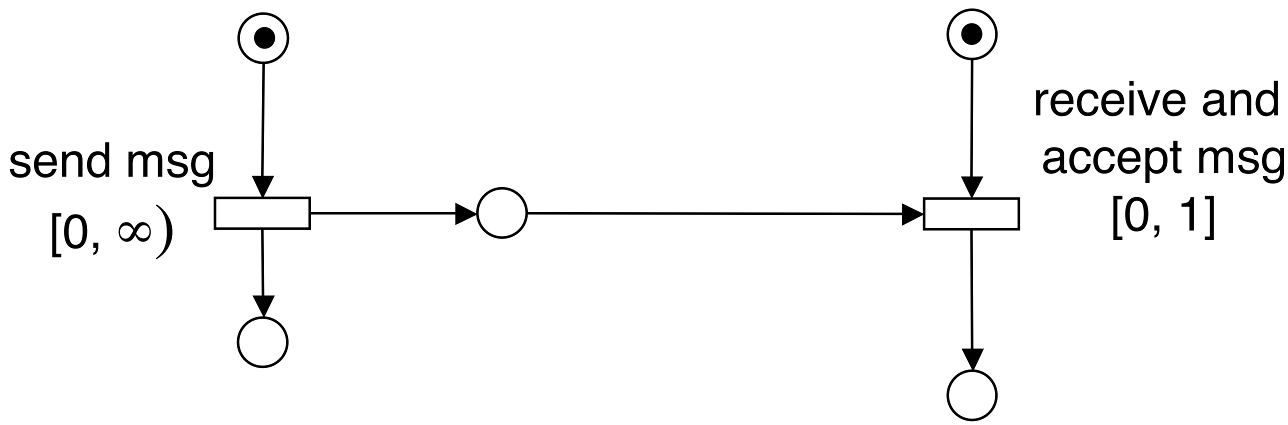


1 of 16 possible messages



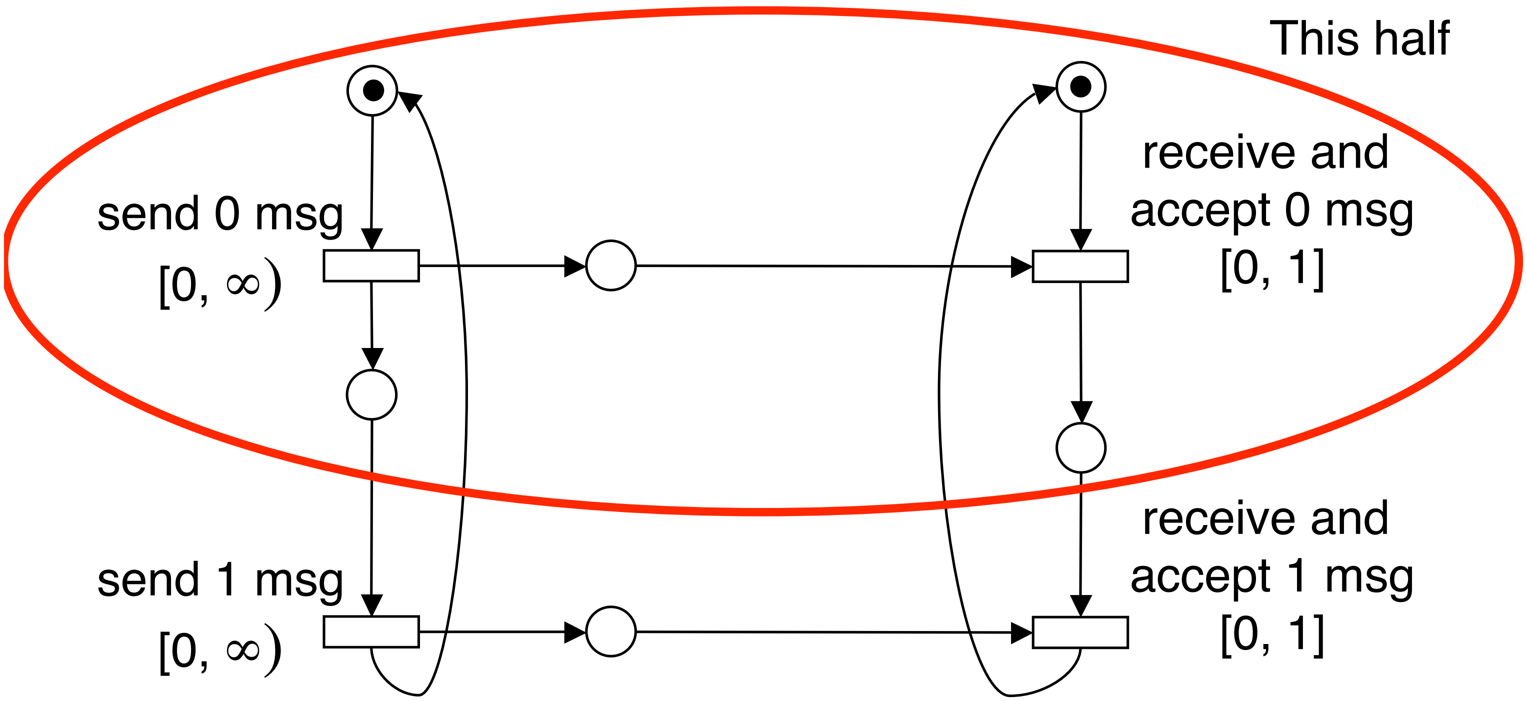
Example from Berthomieu & Diaz: Representing ABP in TPN

Basic

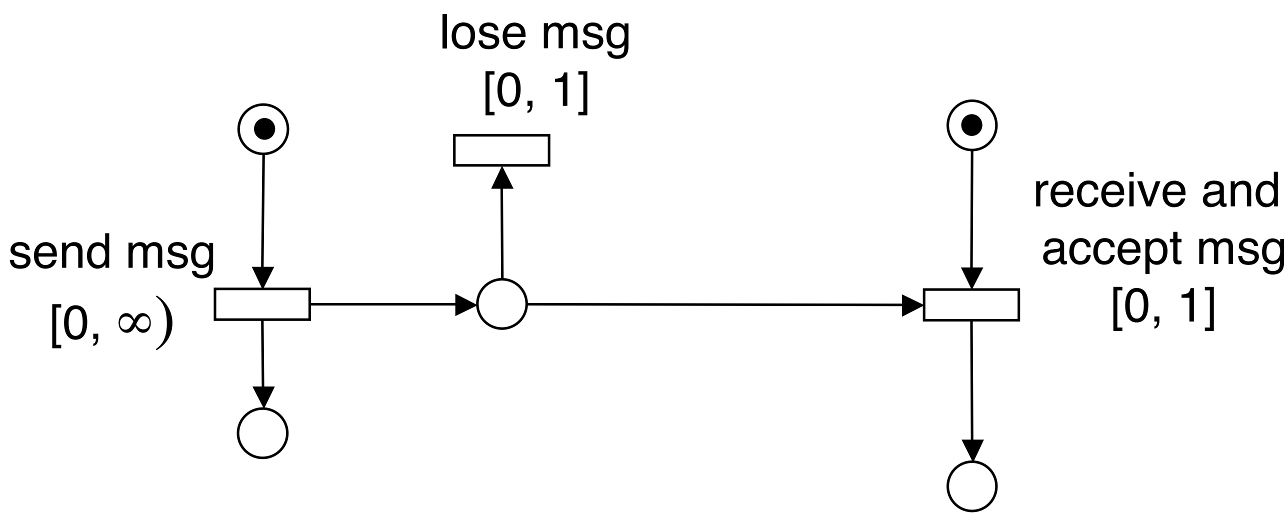


If the message is sent at time t, it will be received by time t+1.

We will focus on the 0 half of the protocol. The 1 half is a mirror image.

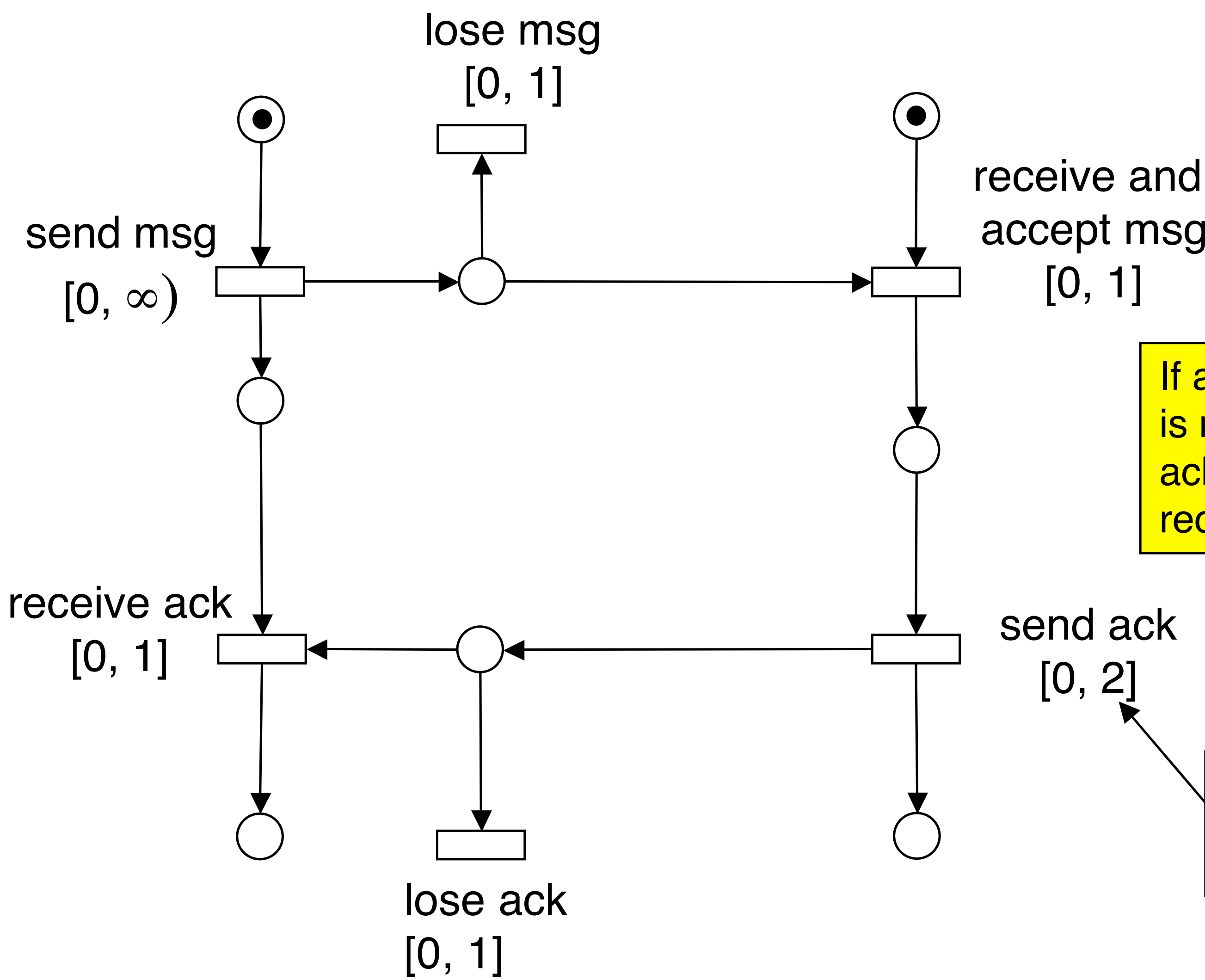


ABP:
Add possible loss



If the message is sent at time t, it will be received **or lost** by time t+1.

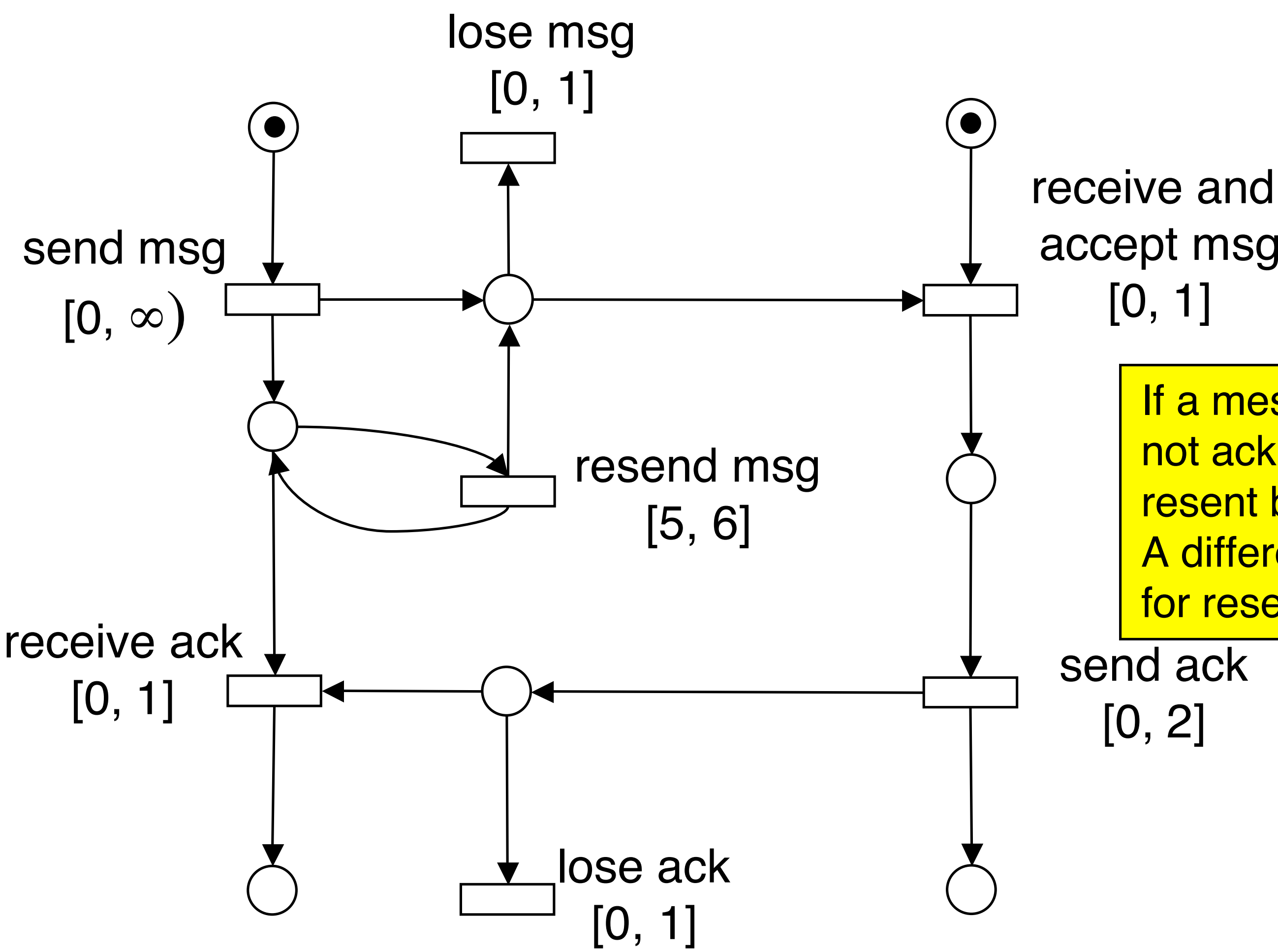
ABP:
Add Acknowledgement



If a message sent at time t is received by t+1, it will be ack'ed by t+3 and the ack received by t+4.

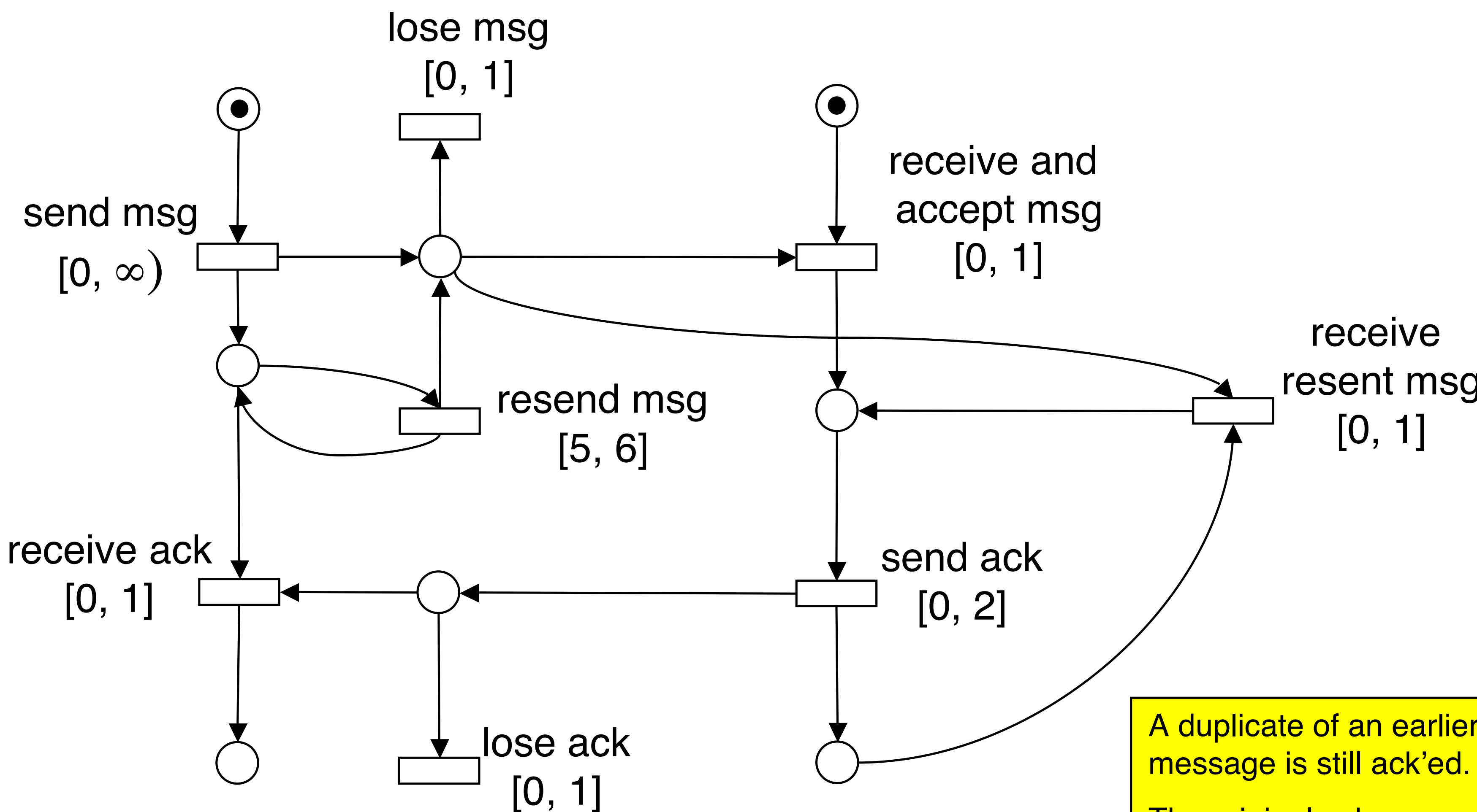
This is the ack time assumed by B&D.
Would 1 work just as well?

ABP:
Add Resend Capability



If a message sent at time t is not ack'ed by t+4, it will be resent by t+6.
A different transition is used for resend.

ABP:
Add receipt of resent messages



A duplicate of an earlier message is still ack'ed.

The original ack may have been lost.

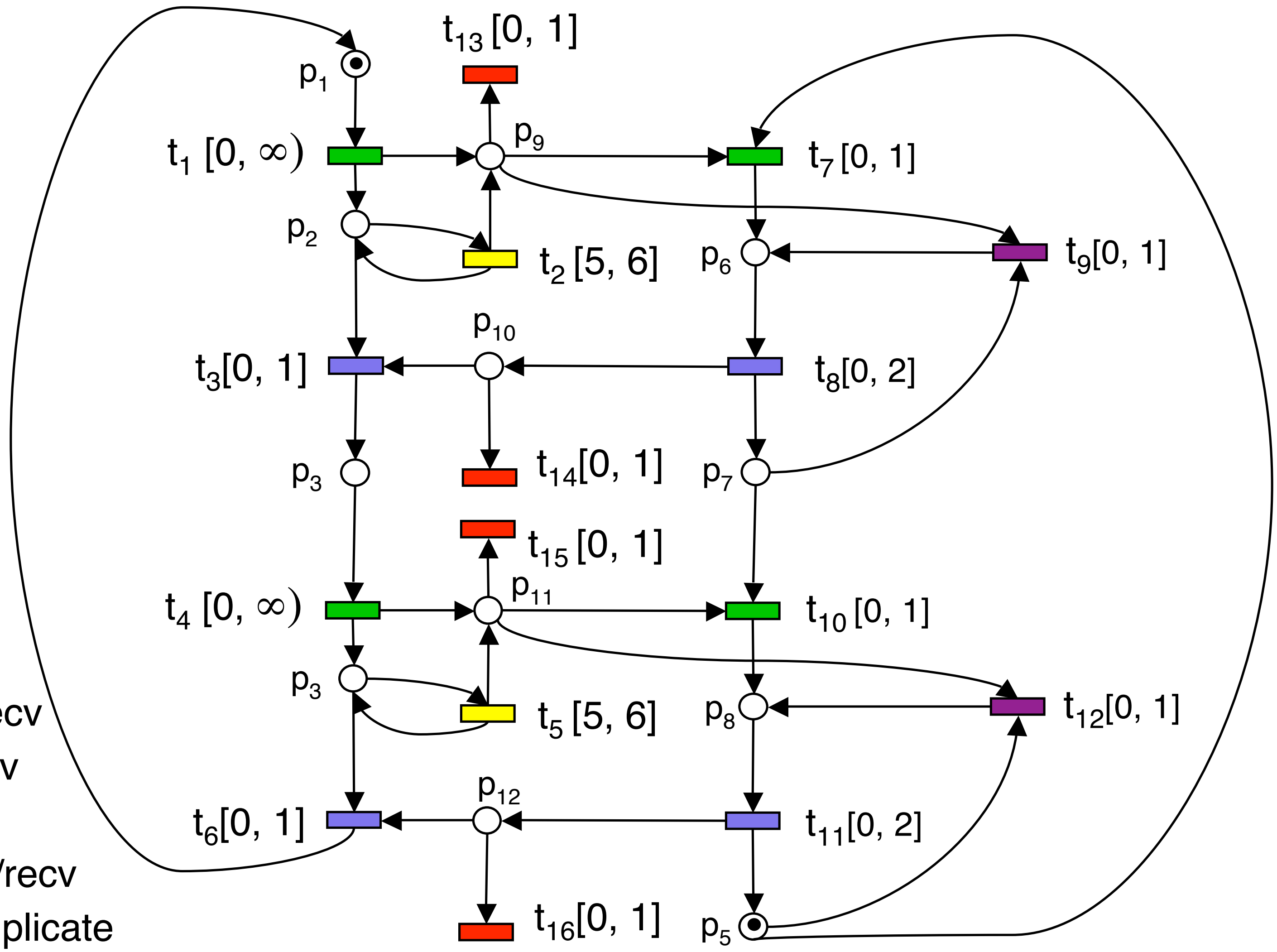
Desire to Show

- By showing that the system is **bounded**, in even the original cyclic context, it is established that **finite queuing capacity** suffices.
- B&D give a **sufficient condition for boundedness** and show that it is satisfied for the version of the ABP presented here.
- B&D develop a “Class Diagram” (not UML) representation for TPN and show that it is **finite** for this particular example. (In general, it won’t be.)

B&D Class Diagram

- Each node of the class diagram includes:
  - State (or “marking”), giving the distribution of tokens on places.
  - For each enabled transition, **upper and lower bounds** on when the transition can fire, relative to the time at which the class is entered.

ABP:
Full System

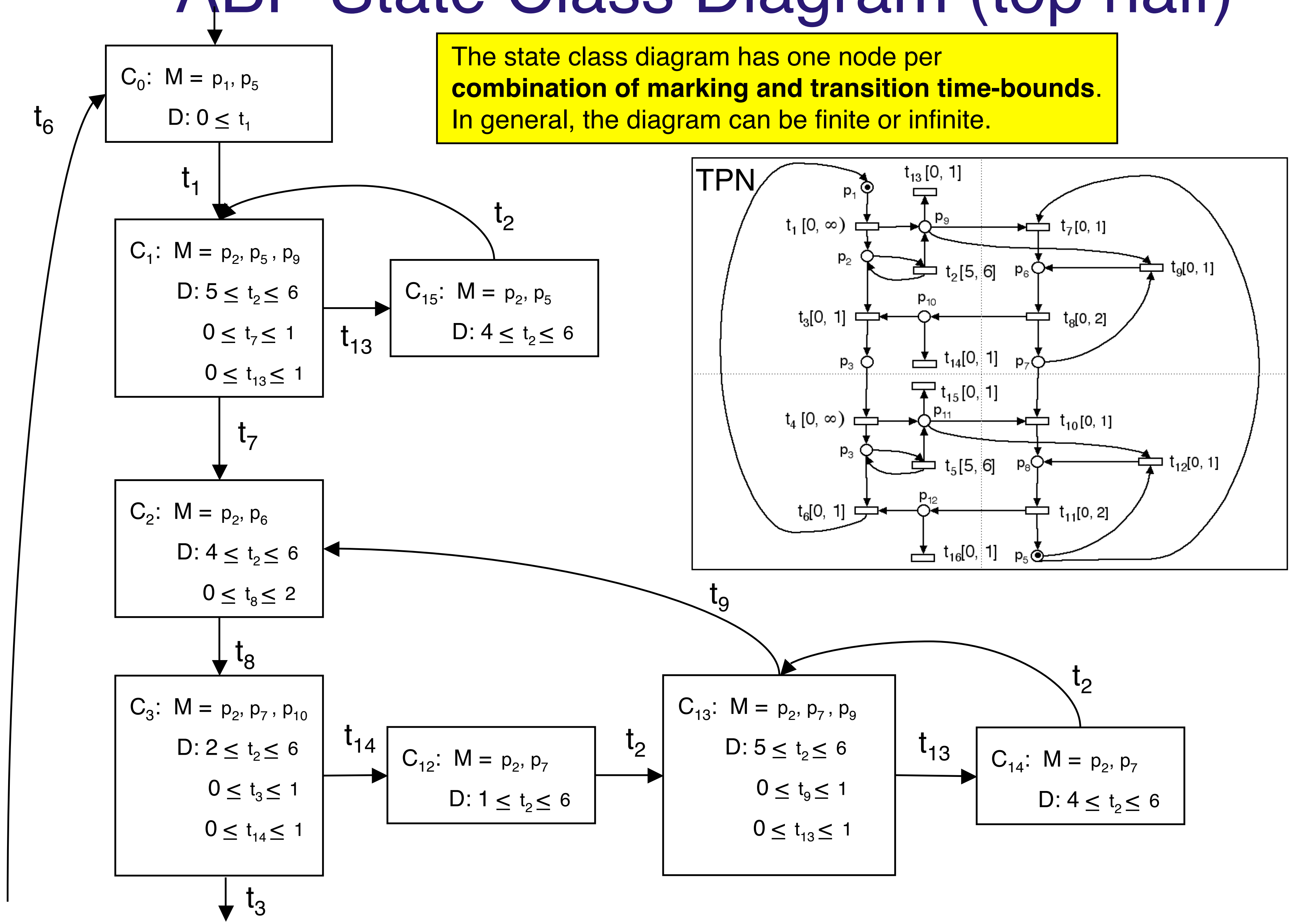


Coding

- send/recv
- ack/recv
- lose
- resend/recv
- recv duplicate

ABP State Class Diagram (top half)

The state class diagram has one node per **combination of marking and transition time-bounds**. In general, the diagram can be finite or infinite.



Other Uses of the State Class Diagram

- Determine various properties of the system, much as could be done by inspecting a finite-state-machine state diagram.

- For example,
  - Are two specific transitions ever enabled simultaneously?
  - Is mutual exclusion ever violated?
  - Is a buffer bound ever exceeded?
  - Can there be deadlock?

For the ABP

- We see that there is never more than one message outstanding on a channel.
  - If the retransmission time-out were set too low, this would not be the case.
- There is no deadlock.

Exercise: Develop a specification, such as a Time Petri Net specification, for the following real-time problem:

- A home alarm system has the following objects:
  - Siren
  - Entry switch (indicates when door is open)
  - Keypad (for arming/disarming)
- The keypad is located inside the home and one must use the door to get to it.

Specification continued

- If the system is not armed, the siren will not sound.
- When the system is set to be armed via the keypad, the user has 30 seconds in which to use the door before the system actually becomes armed.
- If the door is opened while the system is armed, the siren will sound, *unless* the combination is entered on the keypad within 15 seconds.
- Once the siren begins sounding, it will continue until disarmed at the keypad or until for 15 minutes have elapsed, at which time it will enter an advisory state that indicates an alarm occurred.
- The system will stay in the advisory state until disarmed at the keypad.

TAPAAL

- A front-end for Uppaal using Timed-Arc Petri nets
- Developed at Aalborg University

The TAPAAL tool offers a graphical editor for drawing TAPN models, simulator for experimenting with the designed nets and a verification environment that automatically answers logical queries formulated in a subset of CTL logic (essentially EF, EG, AF, AG formulae without nesting). It also allows the user to check whether a given net is k-bounded for a given number k. The verification algorithm translates the TAPAAL queries into UPPAAL ones and relies on the UPPAAL verification engine, but the user does not have to leave the TAPAAL GUI during any phase of the model verification and error traces are displayed directly in TAPAAL.

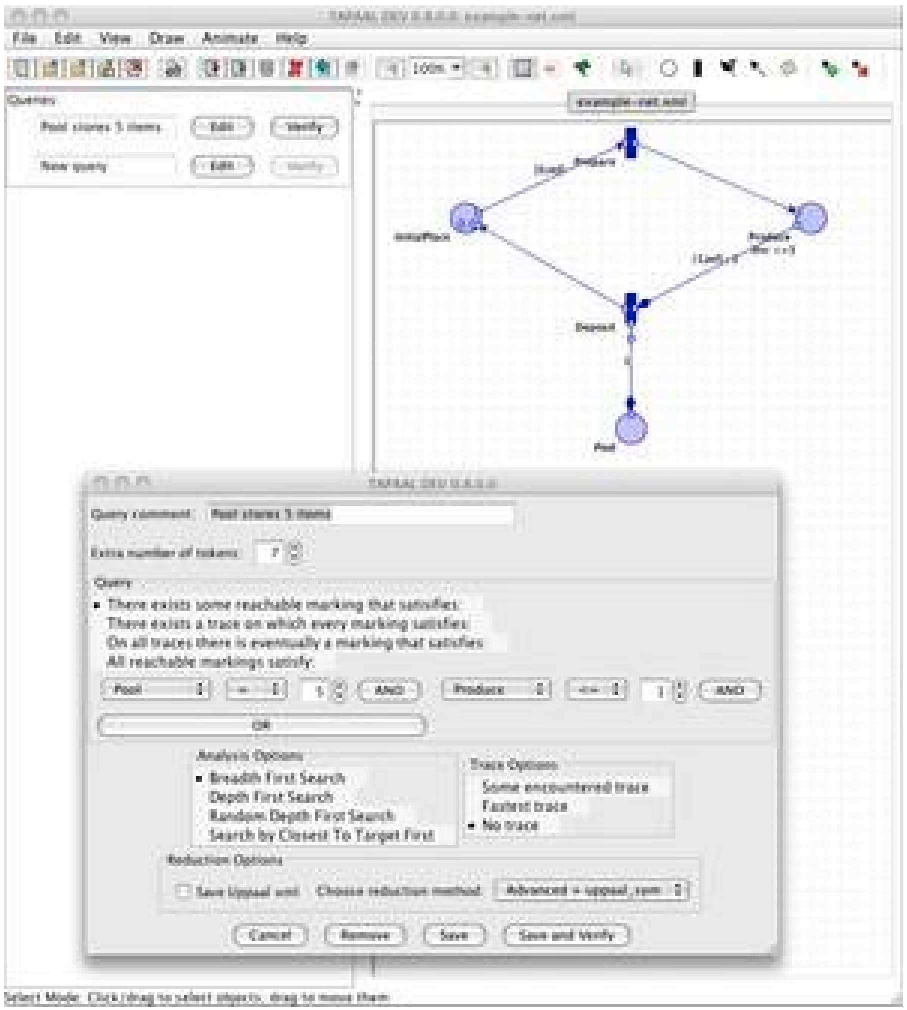
Source: http://www.tapaal.net/

Definition of Timed-Arc Petri Nets

- Tokens are annotated with an age (an integer value indicating the elapsed time from its creation).

- Arcs connecting places with transitions have an associated a time interval, which limits the age of the tokens consumed to fire the adjacent transition.

TAPAAL


File Edit View Draw Animate Help
Queries
Pool stores 5 items
Edit
Verify
New query
Edit
Verify
Query comment:
Pool stores 5 items
Extra number of tokens:
Query
There exists some reachable marking that satisfies:
There exists a trace on which every marking satisfies:
On all traces there is eventually a marking that satisfies:
All reachable markings satisfy:
Pool
AND
Produce
AND
OR
Analysis Options
Breadth First Search
Depth First Search
Random Depth First Search
Search by Closest To Target First
Trace Options
Some encountered trace
Fastest trace
No trace
Reduction Options
Save Uppaal xml
Choose reduction method:
Cancel
Remove
Save
Save and Verify
Select Mode: Click/drag to select objects; drag to move them

Origin of TAPN (Timed-Arc Petri Nets)

- T. Bolognesi and F. Lucidi and S. Trigila: From Timed Petri Nets to Timed LOTOS published in Proceedings of the IFIP WG 6.1 Tenth International Symposium on Protocol Specification, Testing and Verification in 1990 by North-Holland, Amsterdam, pages 1-14,

- H.M. Hanisch: Analysis of Place/Transition Nets with Timed-Arcs and its Application to Batch Process Control published in Proceedings of the 14th International Conference on Application and Theory of Petri Nets (ICATPN'93) in 1993, LNCS volume 691, pages 282-299, 1993.

Timed-Arc Petri Nets vs. Networks of Timed Automata

Jiří Srba*

BRICS**
Department of Computer Science,
Aalborg University, Fredrik Bajersvej 7B,
9220 Aalborg East Denmark
`srba@brics.dk`

Abstract. We establish mutual translations between the classes of 1-safe timed-arc Petri nets (and its extension with testing arcs) and networks of timed automata (and its subclass where every clock used in the guard has to be reset). The presented translations are very tight (up to isomorphism of labelled transition systems with time). This provides a convenient characterization from the theoretical point of view but is not always satisfactory from the practical point of view because of the possible non-polynomial blow up in the size (in the direction from automata to nets). Hence we relax the isomorphism requirement and provide efficient (polynomial time) reductions between networks of timed automata and 1-safe timed-arc Petri nets preserving the answer to the reachability question. This makes our techniques suitable for automatic translation into a format required by tools like UPPAAL and KRONOS. A direct corollary of the presented reductions is a new PSPACE-completeness result for reachability in 1-safe timed-arc Petri nets, reusing the region/zone techniques already developed for timed automata.

Other Variations on Petri Nets

- Add inhibiting arcs
- Add logic
- Add priorities to transitions
- Add stochastic behavior
- Coloured Petri Nets (Jensen)
  - Add types to tokens
  - Add complex operations to transitions
  - Add hierarchy
- HL (High-Level) Nets

For XML fans: PNML

http://www2.informatik.hu-berlin.de/top/pnml/about.html

- Petri Net mathematical description

```
<pn>
  <place id="input">
    <name>Input</name>
    <type>InItem</type>
  </place>

  <arc id="a1">
    <placeend idref="input">
    <transend idref="get-next-item">
    <expr>iitem</expr>
  </arc>

  <transition id="get-next-item">
    <name>Get Next Item</name>
  </transition>
```

```
  <place id="idle">
    <name>Idle</name>
    <type>M</type>
    <initmark>M</initmark>
  </place>

  <arc id="a2">
    <placeend idref="idle">
    <transend idref="get-next-item">
    <expr>m</expr>
  </arc>

  ...

</pn>
```

For XML fans: PNML

- Petri Net layout description

```
<layout>
  <node ref="machine.pn#input">
    <coord>
      <x>10.0</x>
      <y>0.0</y>
    </coord>
  </node>

  <conn ref="machine.pn#a1">
    <!-- no bend points -->

  <node ref="machine.pn#get-next-item">
    <coord>
      <x>10.0</x>
      <y>10.0</y>
    </coord>
  </node>
```

```
  <node ref="machine.pn#idle">
    <coord>
      <x>20.0</x>
      <y>20.0</y>
    </coord>
  </node>

  <conn ref="machine.pn#a2">
    <coord>
      <x>20.0</x>
      <y>10.0</y>
    </coord>
  </conn>

  ...

</layout>
```