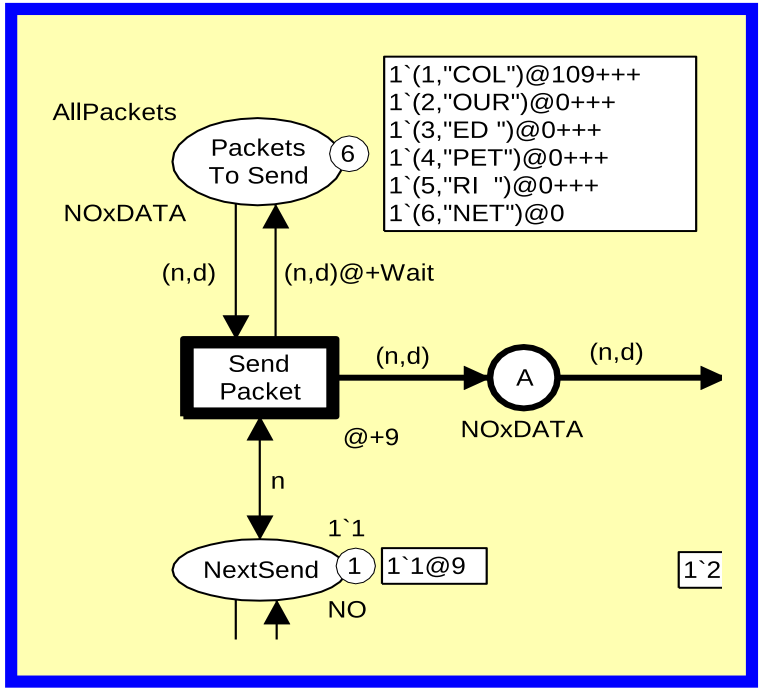
Coloured Petri Nets

Modelling and Validation of Concurrent Systems

Chapter 10: Timed Coloured Petri Nets

Kurt Jensen &
Lars Michael Kristensen

{kjensen,lmkristensen}@cs.au.dk



Two kinds of properties

- Up to now we have concentrated on the functional/logical properties of the modelled system such as deadlocks and home markings

- It is also important to be able to analyse how efficient a system performs its operations
- This is done by means of timed CPN models

- A timed CPN model allows us to investigate performance measures such as queue lengths and waiting times

CPN language can be used for both

- The CPN modelling language can be used to investigate both
  - Functional/logical properties
  - Performance properties

- Most other modelling languages can only be used to analyse either functional/logical properties or performance properties

- It is an obvious advantage to be able to make both kinds of analysis by means of the same modelling language

- Usually we have two slightly different but closely related CPN models

Timed CPN models

- In a timed CPN model tokens have
  - A token colour
  - A timestamp

- The timestamp is a non-negative integer belonging to the type TIME
- The timestamp tells us the time at which the token is ready to be removed by an occurring transition

- The system has a global clock representing model time
- The global clock is shared by all modules in the CPN model

Multisets and timed multi-sets

Addition of timed multisets

- Untimed multiset:

```
1`(1,"COL") ++
1`(1,"OUR") ++
1`(1,"ED ") ++
1`(1,"PET") ++
1`(1,"RI ") ++
1`(1,"NET")
```

Coefficient

Token colour

- Timed multiset:

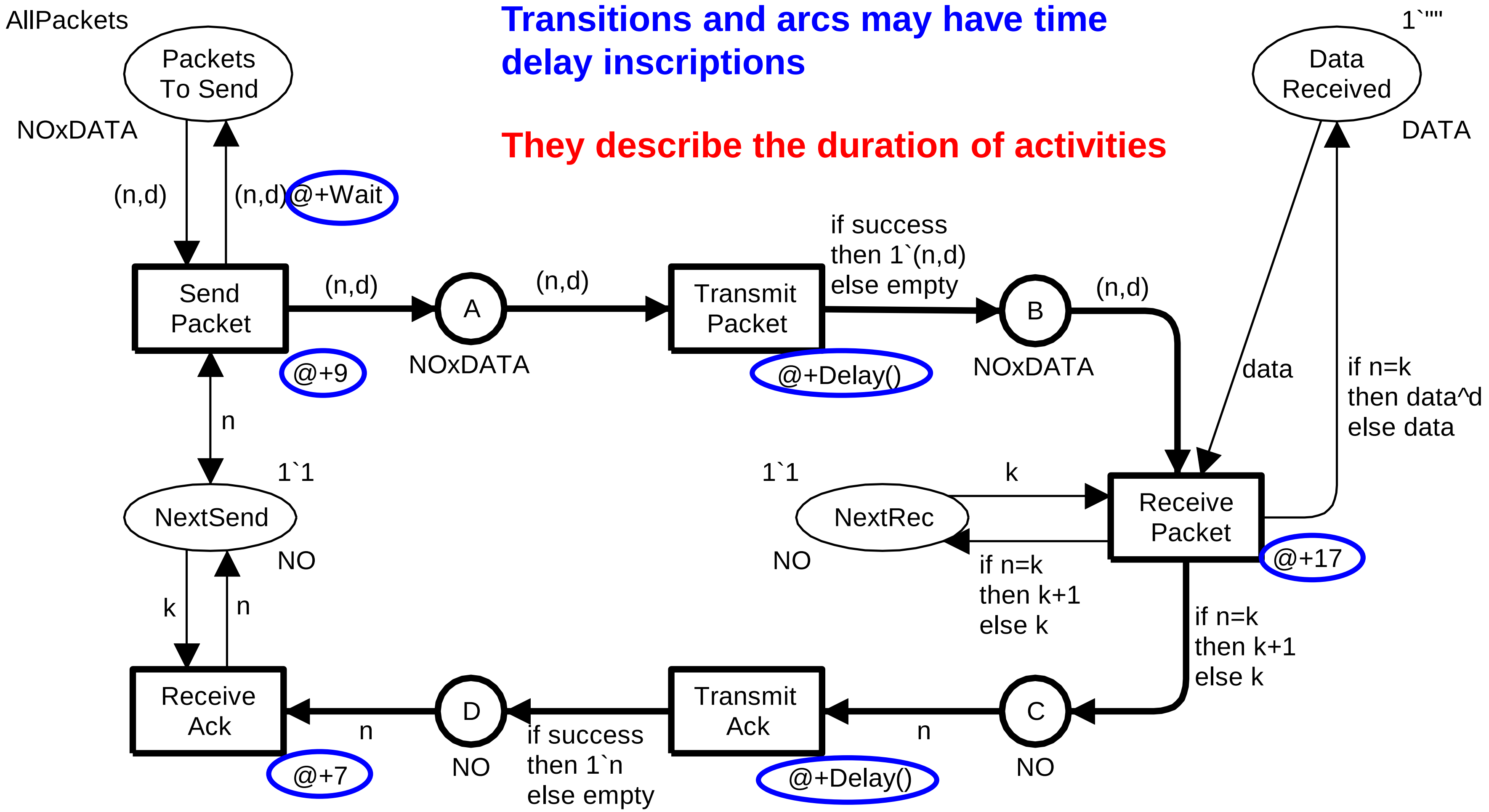
```
1`(1,"COL")@218 +++
1`(1,"OUR")@2095 +++
1`(1,"ED ")@2664 +++
1`(1,"PET")@2906 +++
1`(1,"RI ")@3257 +++
1`(1,"NET")@3499
```

Coefficient

Token colour

Timestamp

Timed CPN model for our protocol



Transitions and arcs may have time delay inscriptions

They describe the duration of activities

AARHUS UNIVERSITY
Coloured Petri Nets
Department of Computer Science
Kurt Jensen
Lars M. Kristensen

Definitions and declarations

- Definitions of colour sets

CPN ML keyword

```
colset NO      = int timed;
colset DATA    = string timed;
colset NOxDATA = product NO * DATA timed;
colset BOOL    = bool;
```

- Tokens of type NO, DATA, and NOxDATA will carry timestamps

- Declarations of variables and constants

```
var n,k : NO;
var d, data : DATA;
var success : BOOL;
val Wait = 100;
```

New symbolic constant specifying the delay between retransmissions

New function

- Definition of function

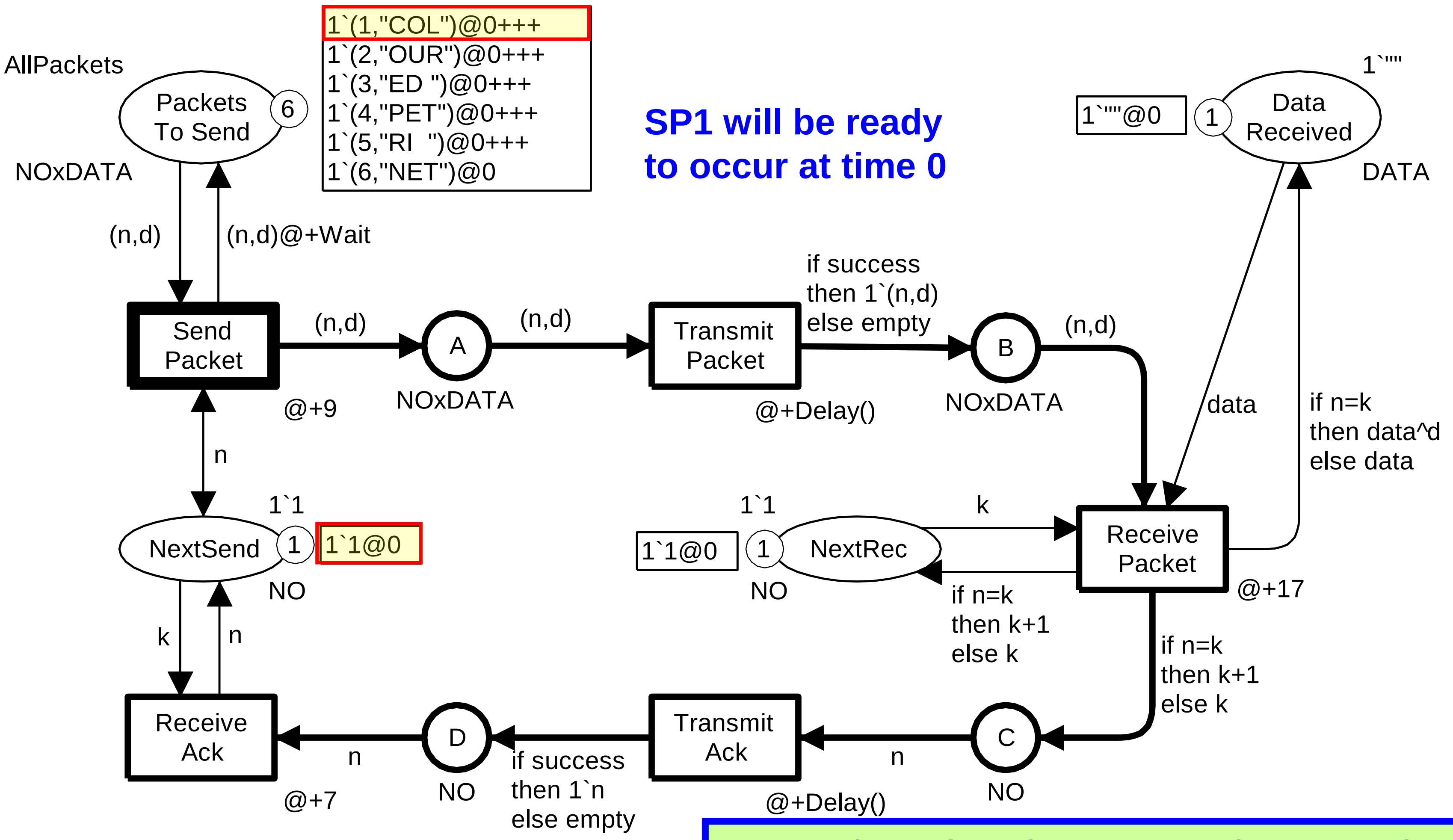
```
fun Delay() = discrete(25,75);
```

Predefined function returning an arbitrary integer in the interval specified by its argument

- Delay returns an integer between 25 and 75
- All 51 values have the same probability to be chosen
- The choice is made by a random number generator

- The Delay function will be used to model the transmission time for data packets and acknowledgements
- The transmission time may vary between 25 and 75 time units – e.g. depending on the network load

Initial marking M_0

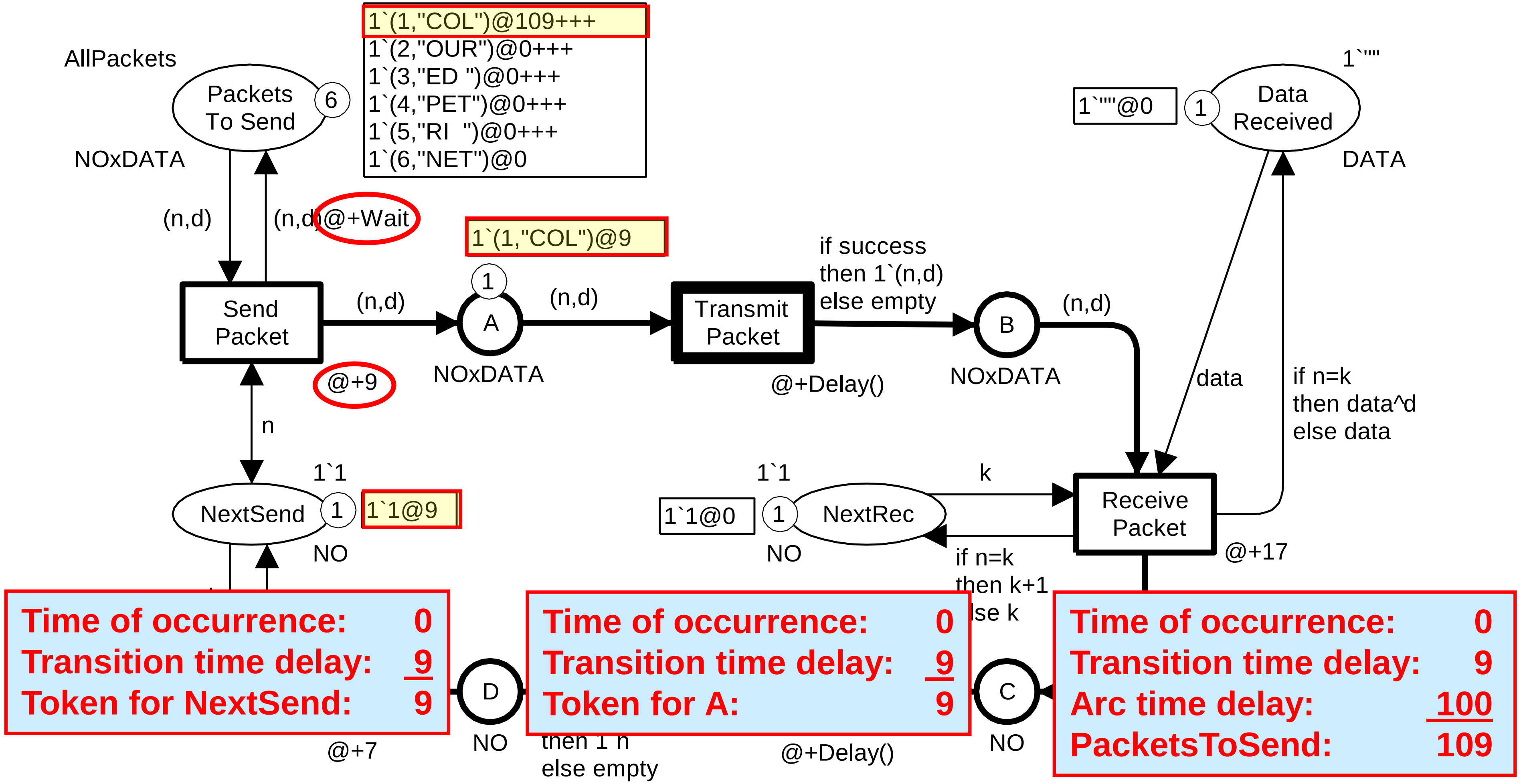


SP1 will be ready to occur at time 0

SP1 = (SendPacket, <n=1, d="COL">) **0**

Marking M_1

- SP1 has occurred at time 0.



Time delays

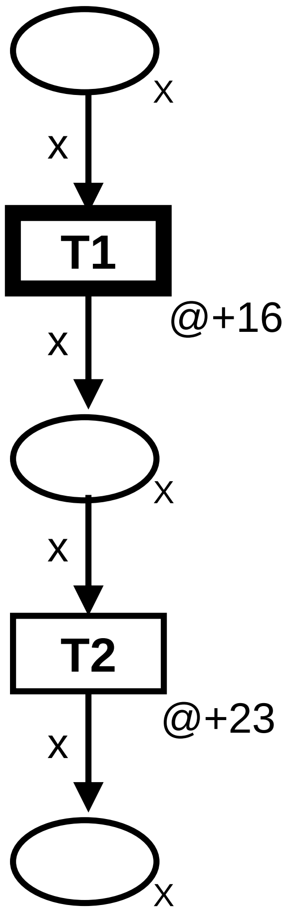
- Transitions and arcs may have time delay inscriptions
- They are CPN ML expressions of type TIME – i.e. they evaluate to a non-negative integer

- A time delay at a transition applies to all tokens produced by the transition
- A time delay at an output arc applies to all tokens produced by that arc

- An omitted time delay is a shorthand for a zero time-delay

Transitions occur instantaneously

- The occurrence of a transition is always instantaneous, i.e. takes no time
- The time delay Δ of a transition can be interpreted as the duration of the operation which is modelled by the transition
- The output tokens of the transition will not be available for other transitions until Δ time units later



- If T1 occurs at time 45 it will produce a token with timestamp 61
- This implies that T2 cannot occur until 16 time units after the beginning of the operation modelled by T1

Another possibility

- At a first glance, it may look simpler to define the occurrence of a transition with time delay Δ to take Δ time units
  - Removing input tokens when the occurrence starts
  - Adding output tokens when the occurrence ends

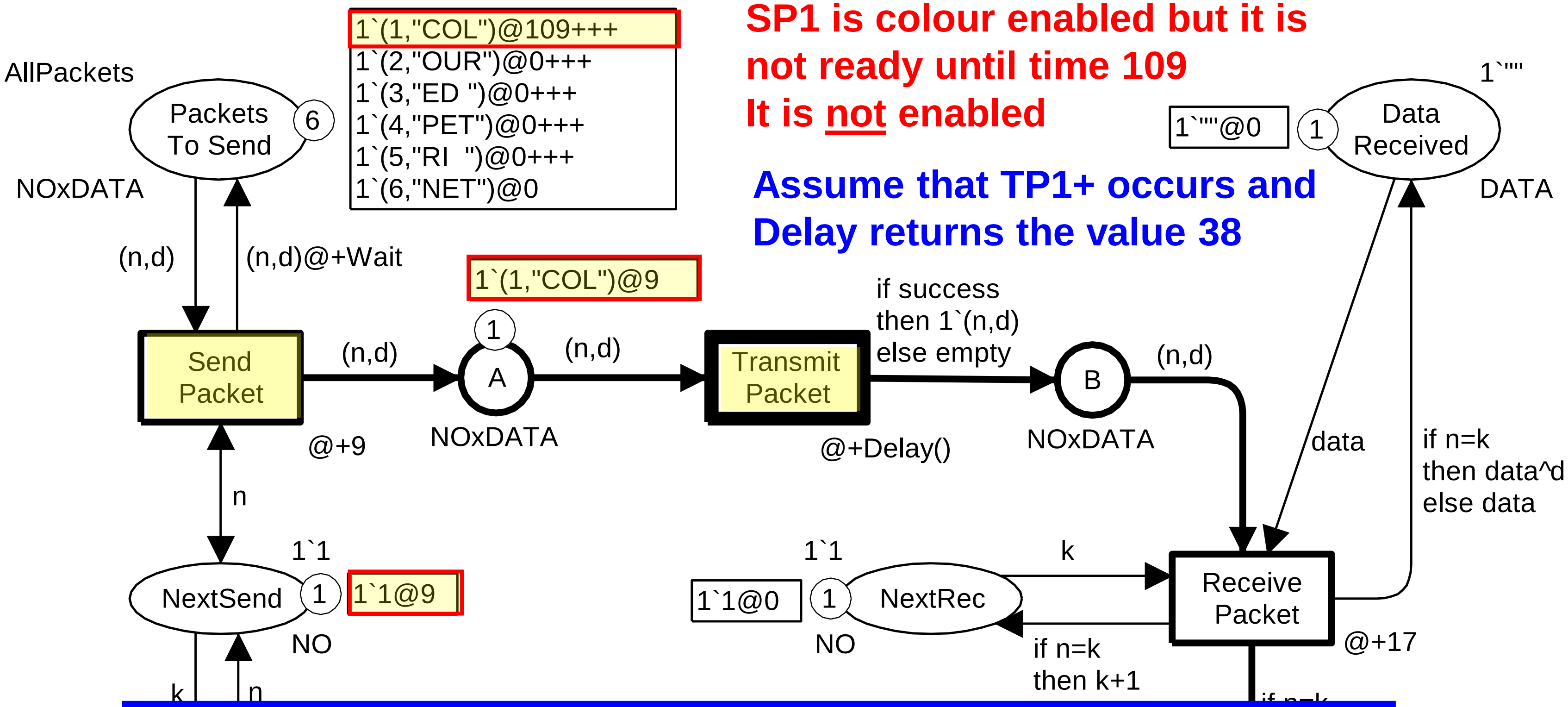
- This would imply that a timed CPN model has a number of intermediate markings with no counterparts in the corresponding untimed CPN model
  - Some transitions have occurred partially
  - Their input tokens have already been removed
  - Their output tokens have not yet been added

Marking M_1

TP1+ and TP1– are ready to occur at time 9.
They are in conflict with each other

SP1 is colour enabled but it is not ready until time 109
It is not enabled

Assume that TP1+ occurs and Delay returns the value 38



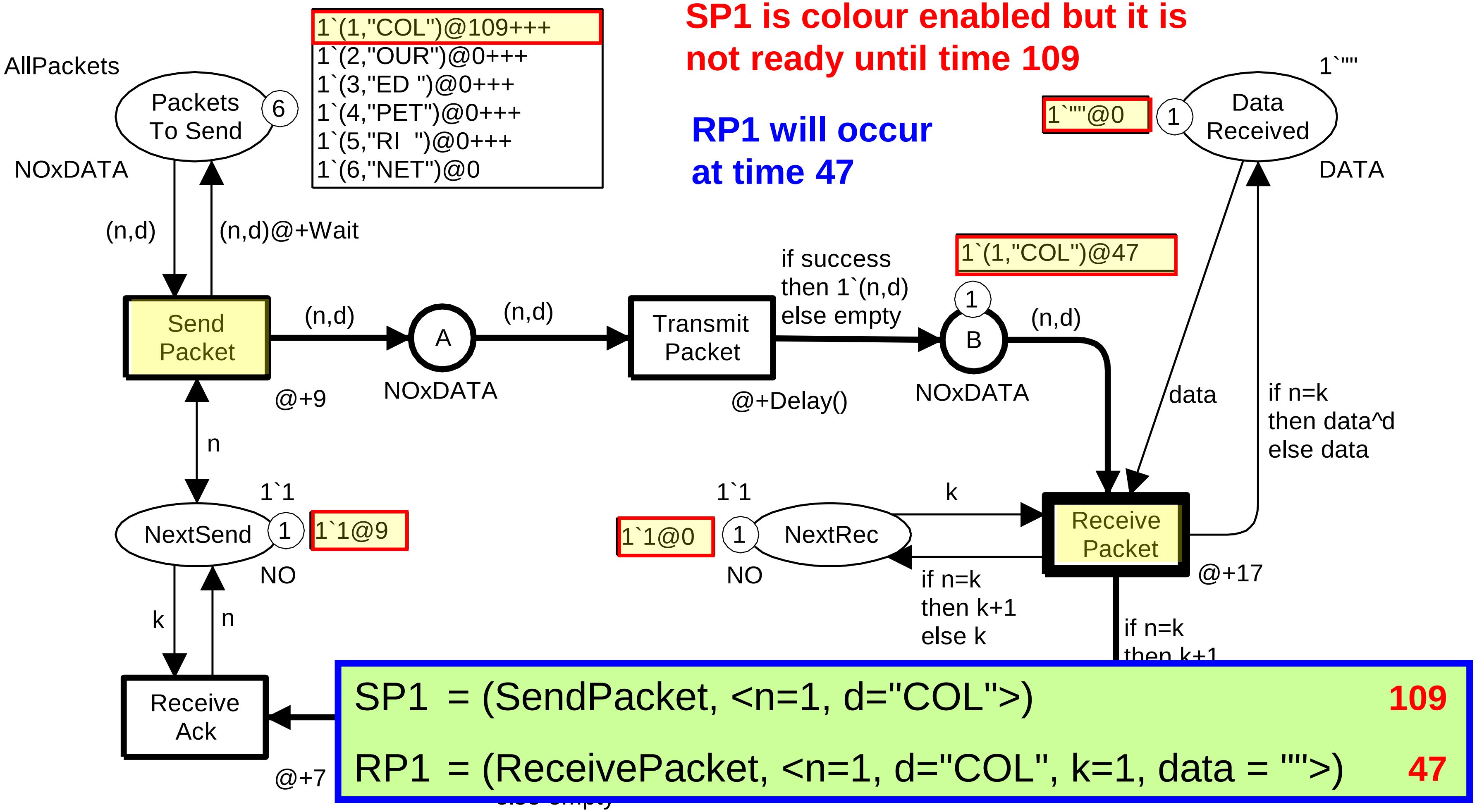
SP1 = (SendPacket, <n=1, d="COL">)		109
TP1+ = (TransmitPacket, <n=1, d="COL", success=true>)		9
TP1– = (TransmitPacket, <n=1, d="COL", success=false>)		9

Marking M_2

RP1 is ready to occur at time 47

SP1 is colour enabled but it is not ready until time 109

RP1 will occur at time 47



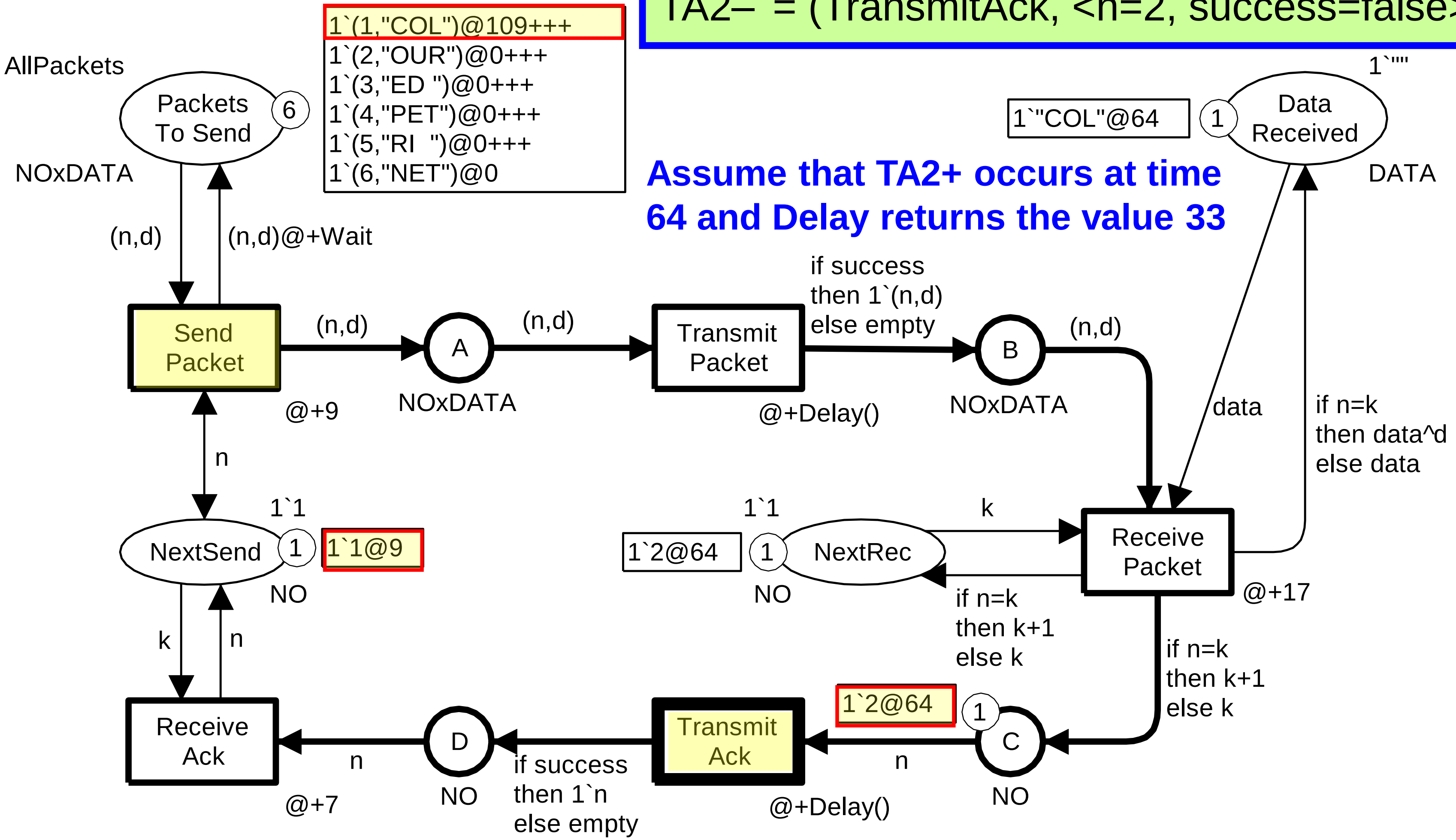
SP1 = (SendPacket, <n=1, d="COL">) **109**

RP1 = (ReceivePacket, <n=1, d="COL", k=1, data = "">) **47**

Marking M_3

SP1	= (SendPacket, <n=1, d="COL">)	109
TA2+	= (TransmitAck, <n=2, success=true>)	64
TA2–	= (TransmitAck, <n=2, success=false>)	64

Assume that TA2+ occurs at time 64 and Delay returns the value 33

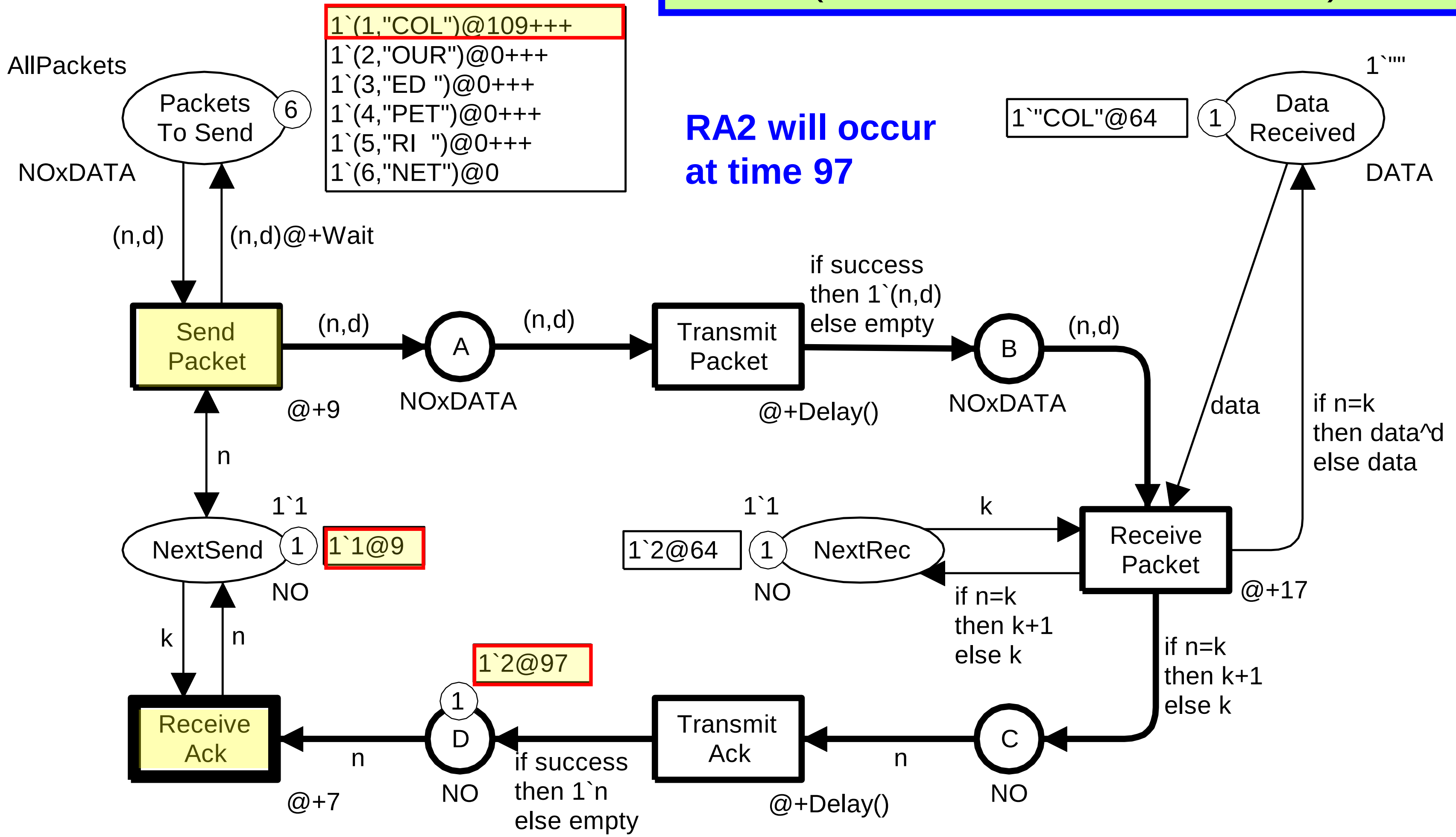


Marking M_4

SP1 = (SendPacket, <n=1, d="COL">) **109**

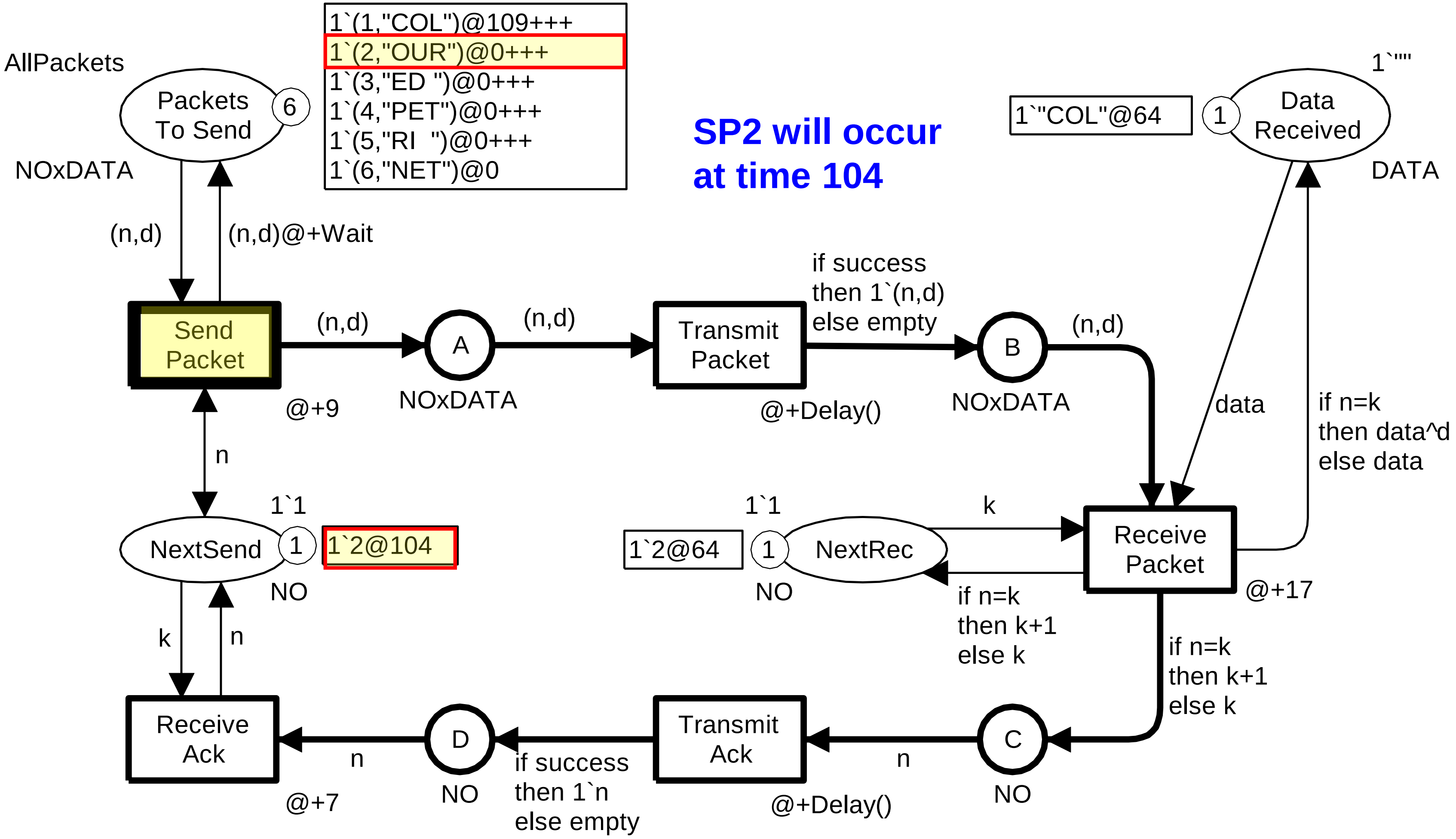
RA2 = (ReceivePacket, <n=2, k=1>) **97**

RA2 will occur at time 97



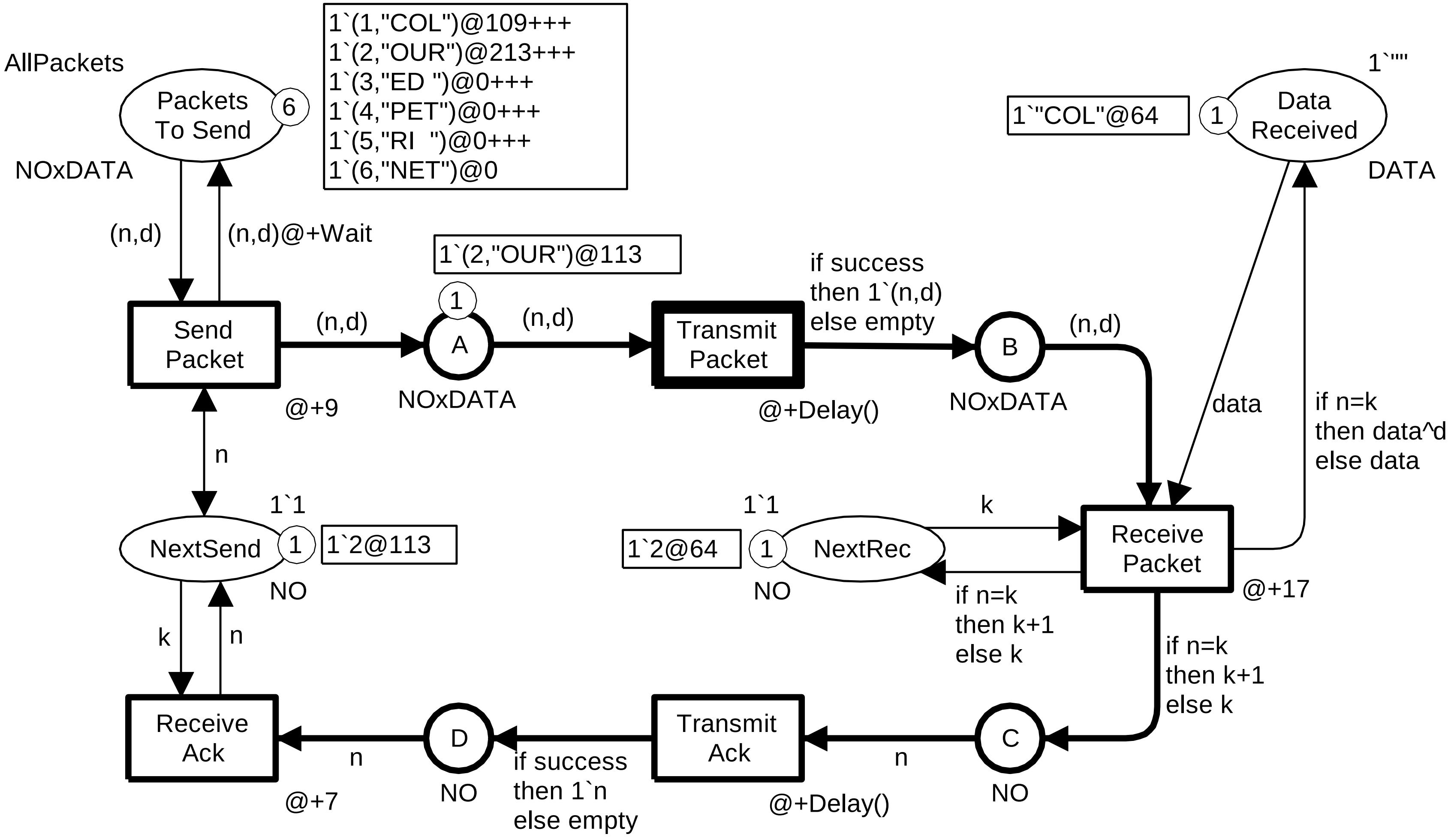
Marking M_5

SP2 = (SendPacket, <n=2, d="OUR">) **104**



SP2 will occur at time 104

Marking M_6



AllPackets
Packets To Send
6
NOxDATA
1`(1,"COL")@109+++
1`(2,"OUR")@213+++
1`(3,"ED ")@0+++
1`(4,"PET")@0+++
1`(5,"RI ")@0+++
1`(6,"NET")@0
(n,d)
(n,d)@+Wait
1`(2,"OUR")@113
1
Send Packet
(n,d)
A
(n,d)
Transmit Packet
if success
then 1`(n,d)
else empty
B
(n,d)
@+9
NOxDATA
@+Delay()
NOxDATA
1`""
1`"COL"@64
1
Data Received
DATA
data
if n=k
then data^d
else data
n
1`1
NextSend
1
1`2@113
NO
1`1
1`2@64
1
NextRec
NO
k
Receive Packet
@+17
if n=k
then k+1
else k
if n=k
then k+1
else k
k
n
Receive Ack
n
D
if success
then 1`n
else empty
Transmit Ack
n
C
@+7
NO
@+Delay()
NO

Retransmissions

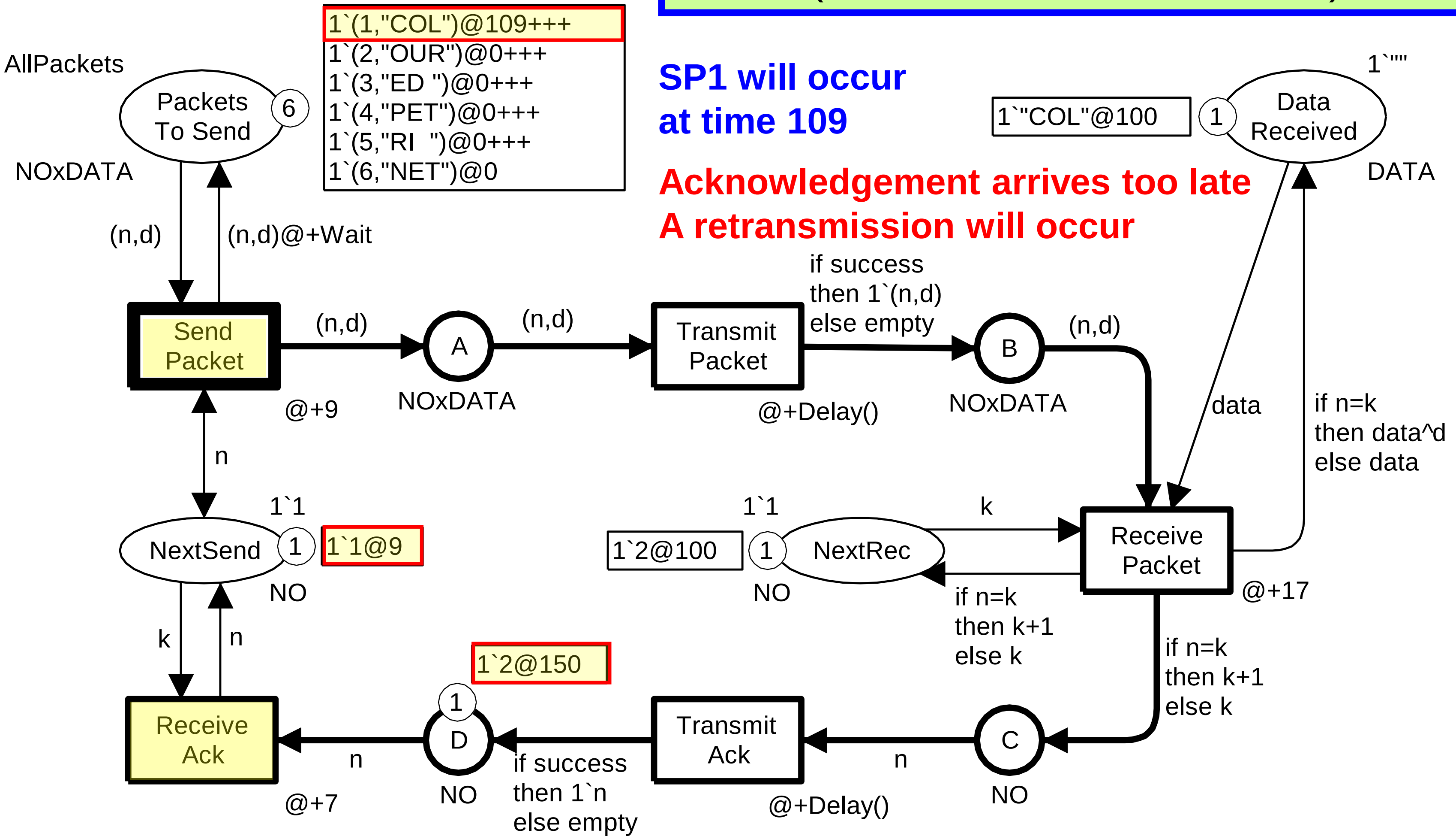
- In the investigated occurrence sequence it turned out to be unnecessary to retransmit data packet no 1
- The acknowledgement requesting data packet no 2 arrived at time 97 while the retransmission would have started at time 109
- When the delays on the network are larger the situation is different and we may, e.g., reach the following marking

Marking $M_4{}^*$

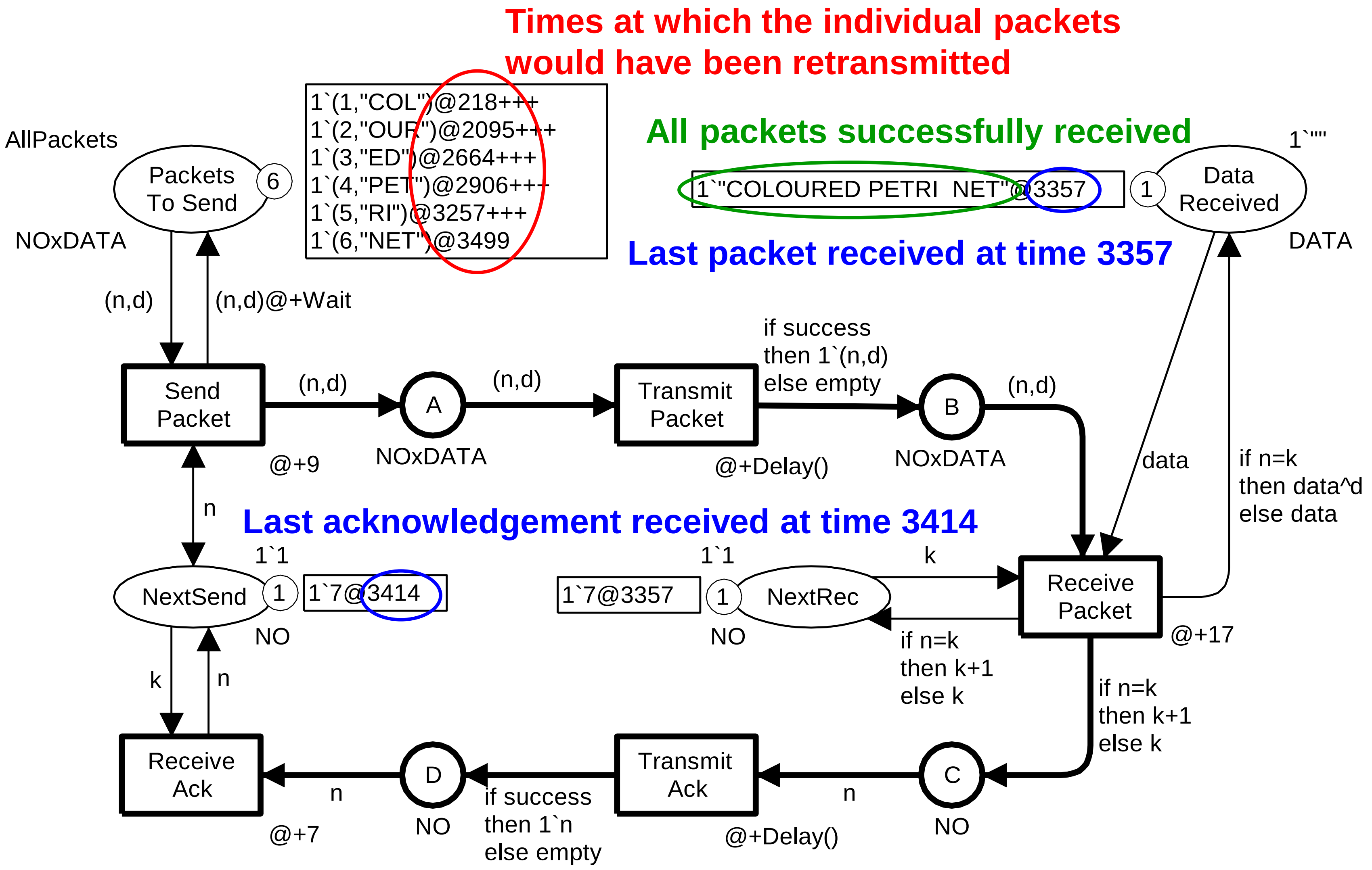
SP1	= (SendPacket, <n=1, d="COL">)	109
RA2	= (ReceivePacket, <n=2, k=1>)	150

SP1 will occur at time 109

Acknowledgement arrives too late
A retransmission will occur



Dead marking at the end of simulation



Non-deterministic simulation

- The chosen occurrence sequence depends on
  - The values returned by the Delay function
  - The choices between conflicting binding elements

- In an automatic simulation both sets of choices are made by means of a random number generator

- A new simulation will result in a new dead marking
  - The token colours will be the same
  - The timestamps will be different

Event queue

- The execution of a timed CPN model is controlled by the global clock
- This is similar to the event queue found in many simulation engines for discrete event simulation

- The model remains at a given model time as long as there are binding elements that are enabled, i.e., are both
  - Colour enabled (have the necessary input tokens)
  - Ready for execution (the required input tokens have time-stamps that are smaller than or equal to the global clock)

- When there are no more enabled binding elements, the simulator advances the global clock to the earliest next model time at which one or more binding elements become enabled
- If no such model time exists the marking is dead

Time delays are additional constraints

- Each timed CPN model determines an underlying untimed CPN model – obtained by removing all time delay inscriptions (and all timestamps)
- The occurrence sequences of a timed CPN model form a subset of the occurrence sequences for the corresponding untimed CPN model
- The time delay inscriptions enforce a set of additional enabling constraints – forcing the binding elements to be executed in the order in which they become enabled

- CPN Tools use the same simulation engine to handle timed and untimed CPN models
- For untimed CPN models the global clock remains at 0

Start with an untimed CPN model

- It often beneficial for the modeller to start by constructing and validating an untimed CPN model
- In this way the modeller can concentrate on the functional/logical properties of the system
- The functional/logical properties should as far as possible be independent of concrete assumptions about execution times and waiting times
- It is possible to describe the existence of time-related system features, such as retransmissions and the expiration of a timer, without explicitly specifying waiting times or the duration of the individual operations

Continue with a timed CPN model

- When the functional/logical properties of a system have been designed and thoroughly validated, the user may analyse and improve the efficiency by which the system performs its operations

- This is done by adding time delays describing the duration of the individual operations
- From our remarks above, it follows that these time delays cannot introduce new behaviour in the system – they merely restrict the possible occurrence sequences

Questions