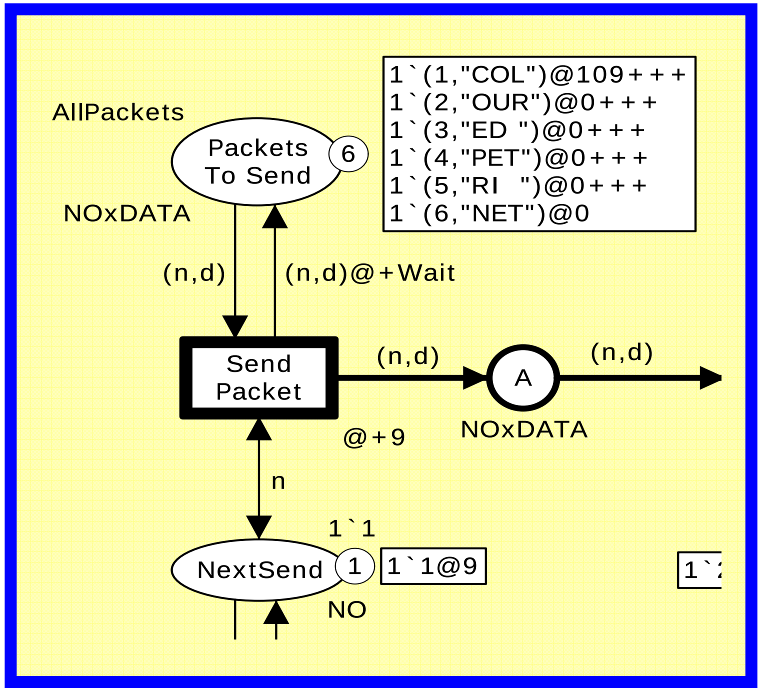
Coloured Petri Nets

Modelling and Validation of Concurrent Systems

Chapter 10: Timed Coloured Petri Nets

Kurt Jensen &
Lars Michael Kristensen

{kjensen,lmkristensen}
@daimi.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 time stamp.

- The time stamp is a non-negative integer belonging to the type TIME.

- The time stamp 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.

Multi-sets

Addition of
timed multi-sets

- Untimed multi-set:

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

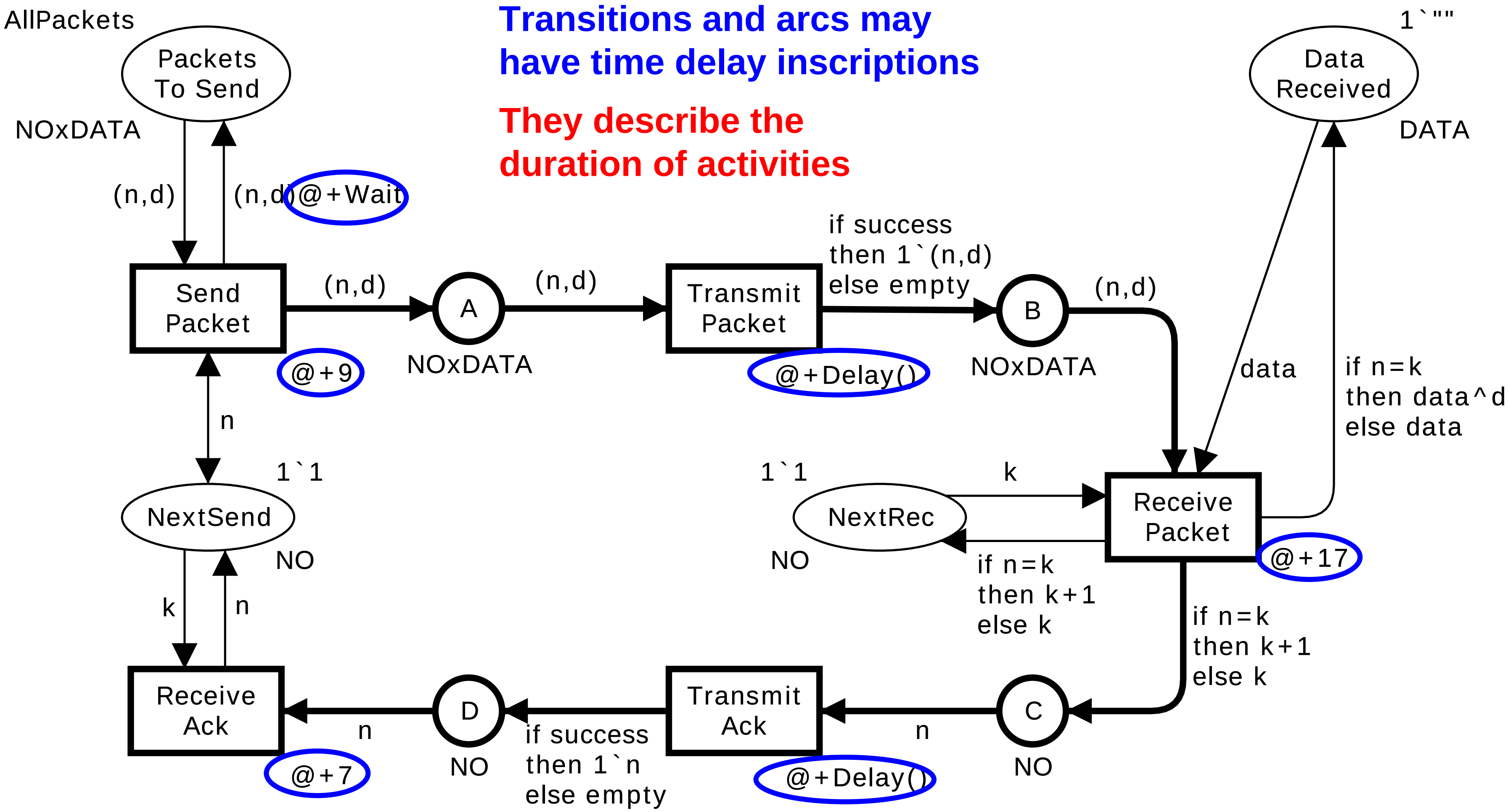
  Coefficient Token colour

- Timed multi-set:

  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 Time stamp

Timed CPN model for our protocol



Transitions and arcs may have time delay inscriptions

They describe the duration of activities

Modelling and Validation of Distributed Systems Group
Department of Computer Science

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 time stamps.

- 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

Modelling and Validation of Distributed Systems Group
Department of Computer Science

New function

- Definition of function:

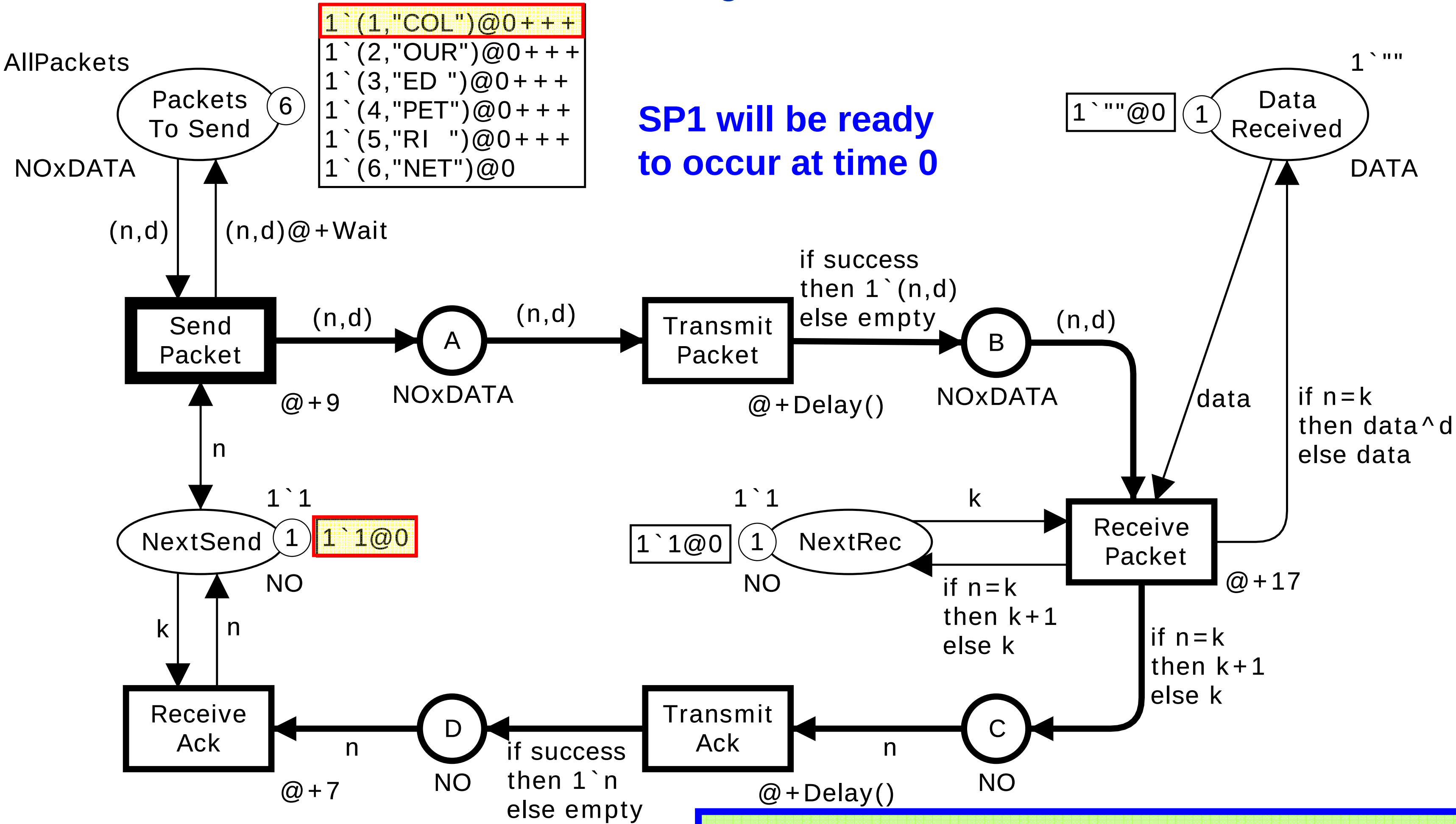
```
fun Delay() = discrete(25,75);
```

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

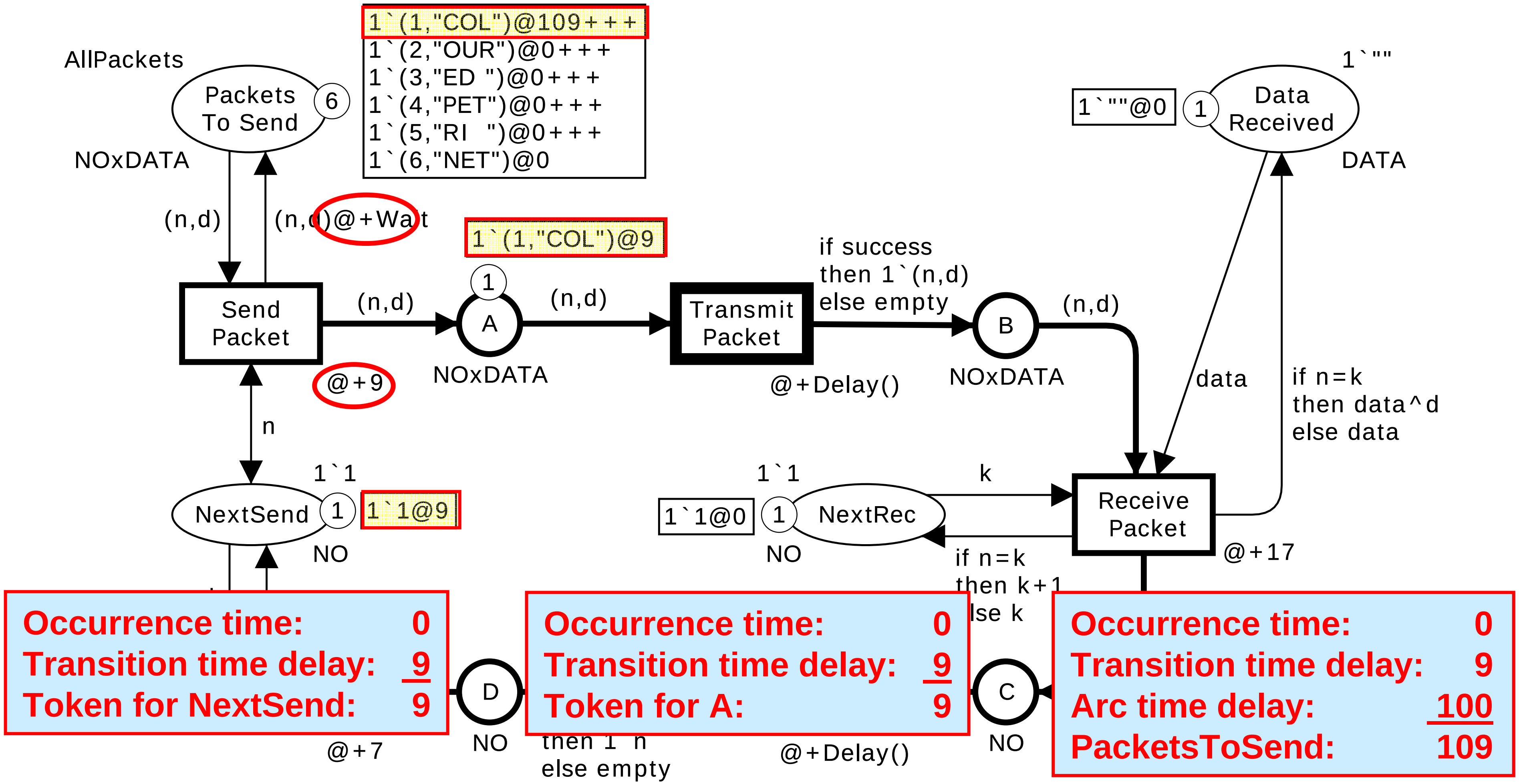


SP1 will be ready to occur at time 0

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

Marking M_1

- SP1 has occurred at time 0.



Occurrence time: 0
Transition time delay: 9
Token for NextSend: 9

Occurrence time: 0
Transition time delay: 9
Token for A: 9

Occurrence time: 0
Transition time delay: 9
Arc time delay: 100
PacketsToSend: 109

Time delays

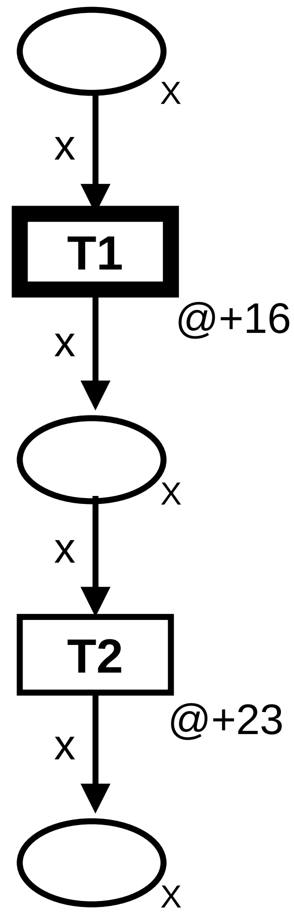
- Transitions and arcs may have time delay inscriptions.
- They are CPN ML expressions of type TIME – i.e. they evaluate to a non-negative integer.

- 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 short-hand 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 time stamp 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.

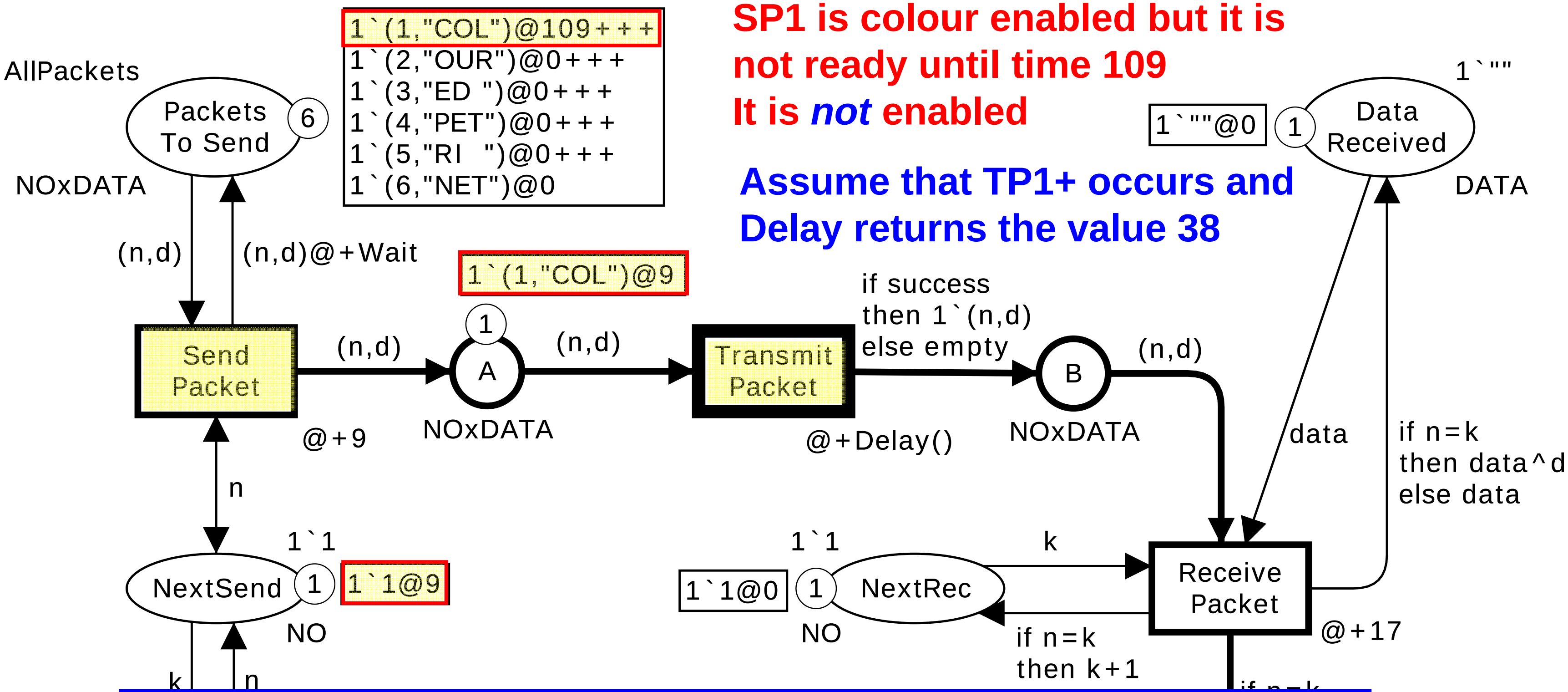
- This would imply that a timed CPN model has a number of intermediate markings with no counterparts in the corresponding untimed CPN model:
  - Some transitions have occurred partially.
  - Their input tokens have already been removed.
  - Their output tokens have not yet been added.

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



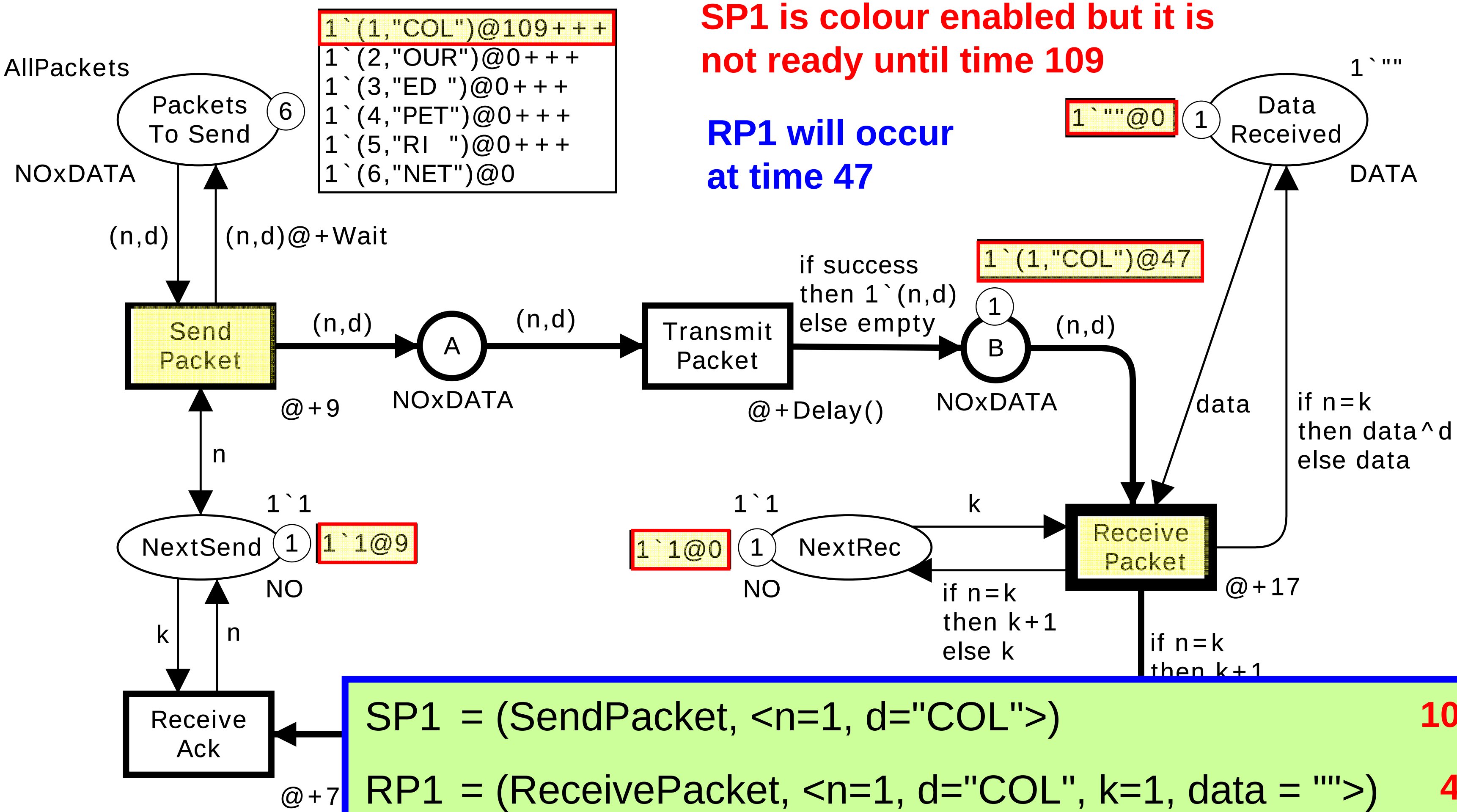
SP1 = (SendPacket, <n=1, d="COL">)	**109**
TP1+ = (TransmitPacket, <n=1, d="COL", success=true>)	**9**
TP1– = (TransmitPacket, <n=1, d="COL", success=false>)	**9**

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

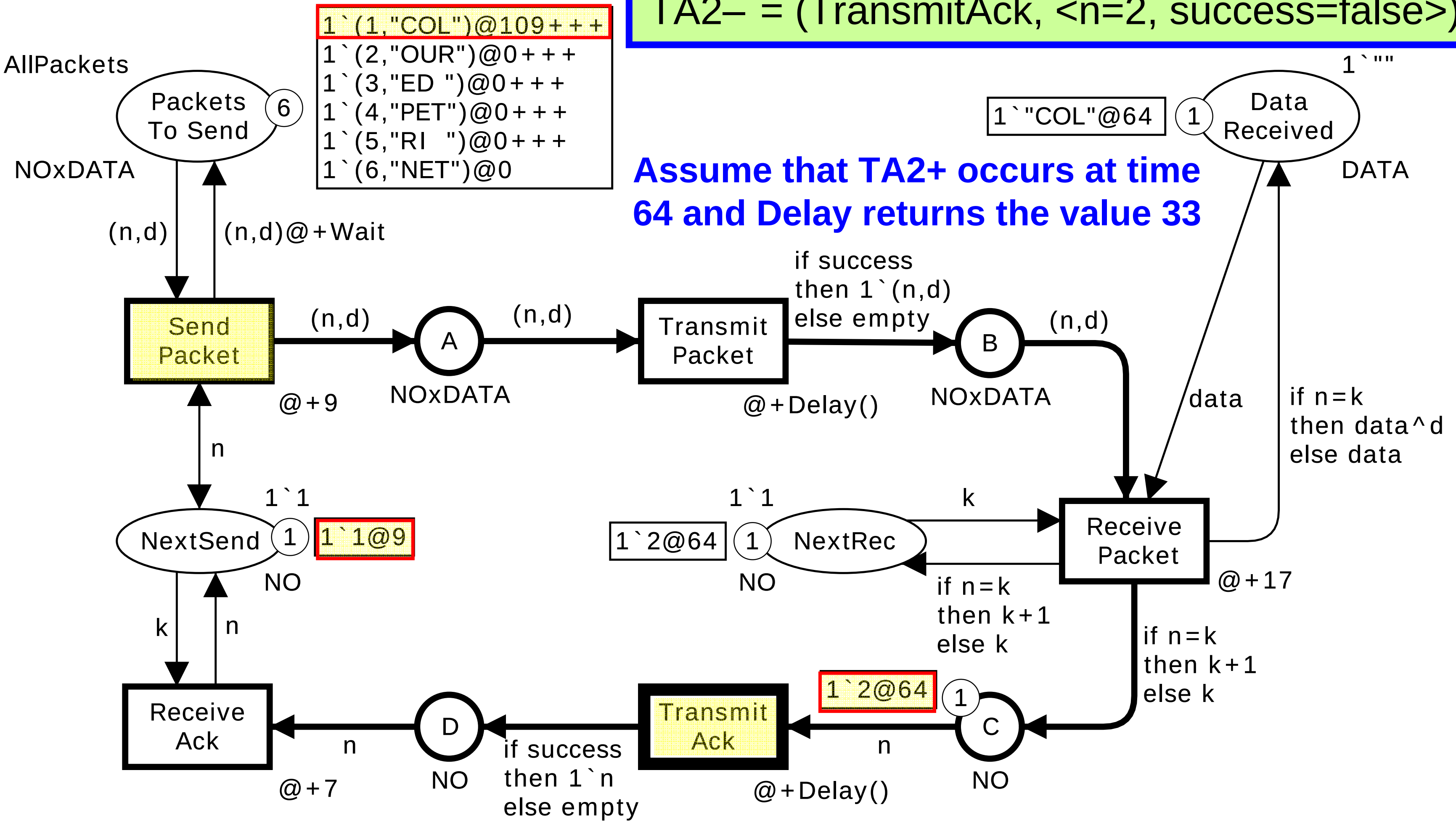
Marking M_3

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



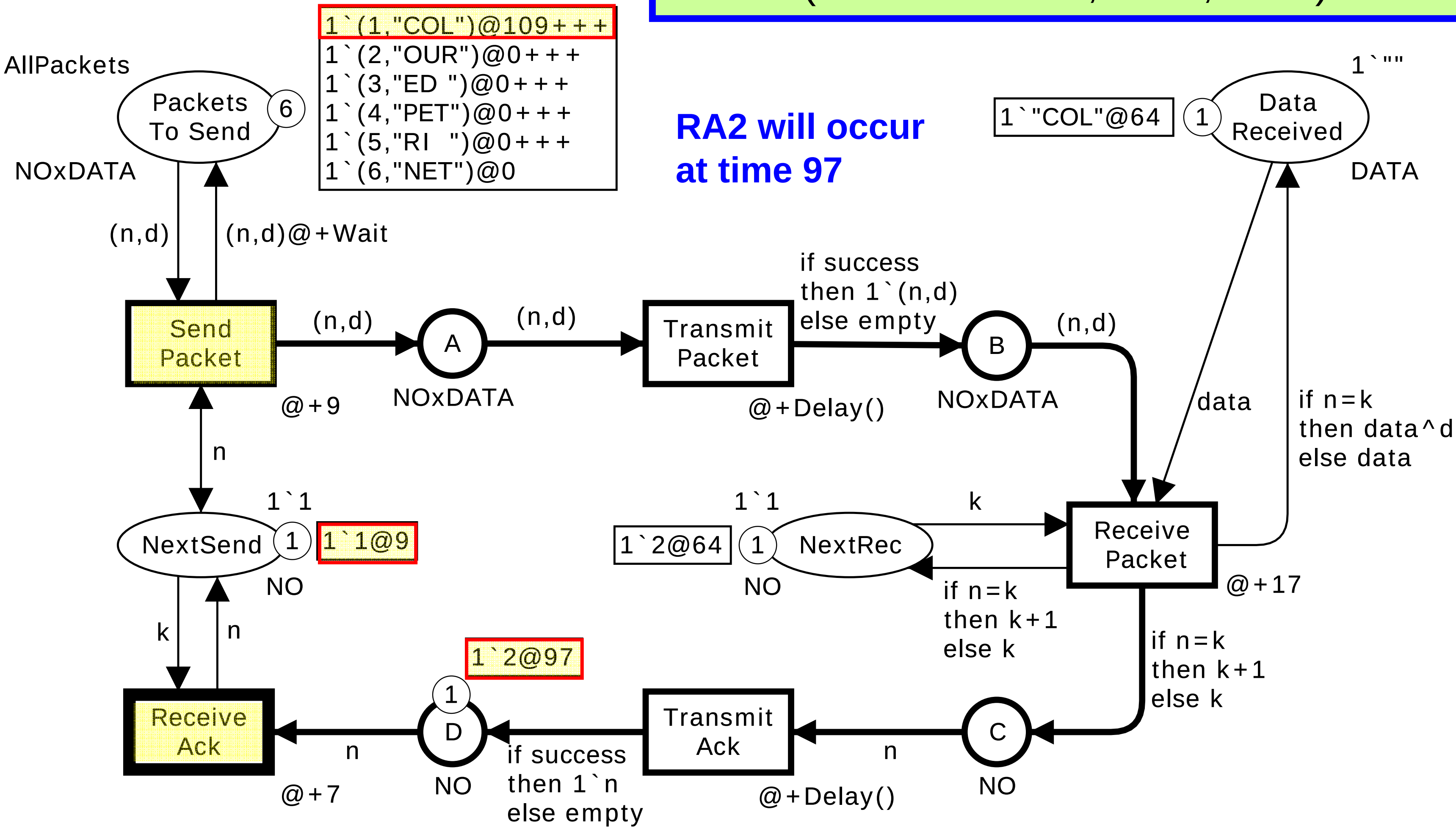
Modelling and Validation of Distributed Systems Group
Department of Computer Science

Kurt Jensen and Lars M. Kristensen

Marking M_4

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

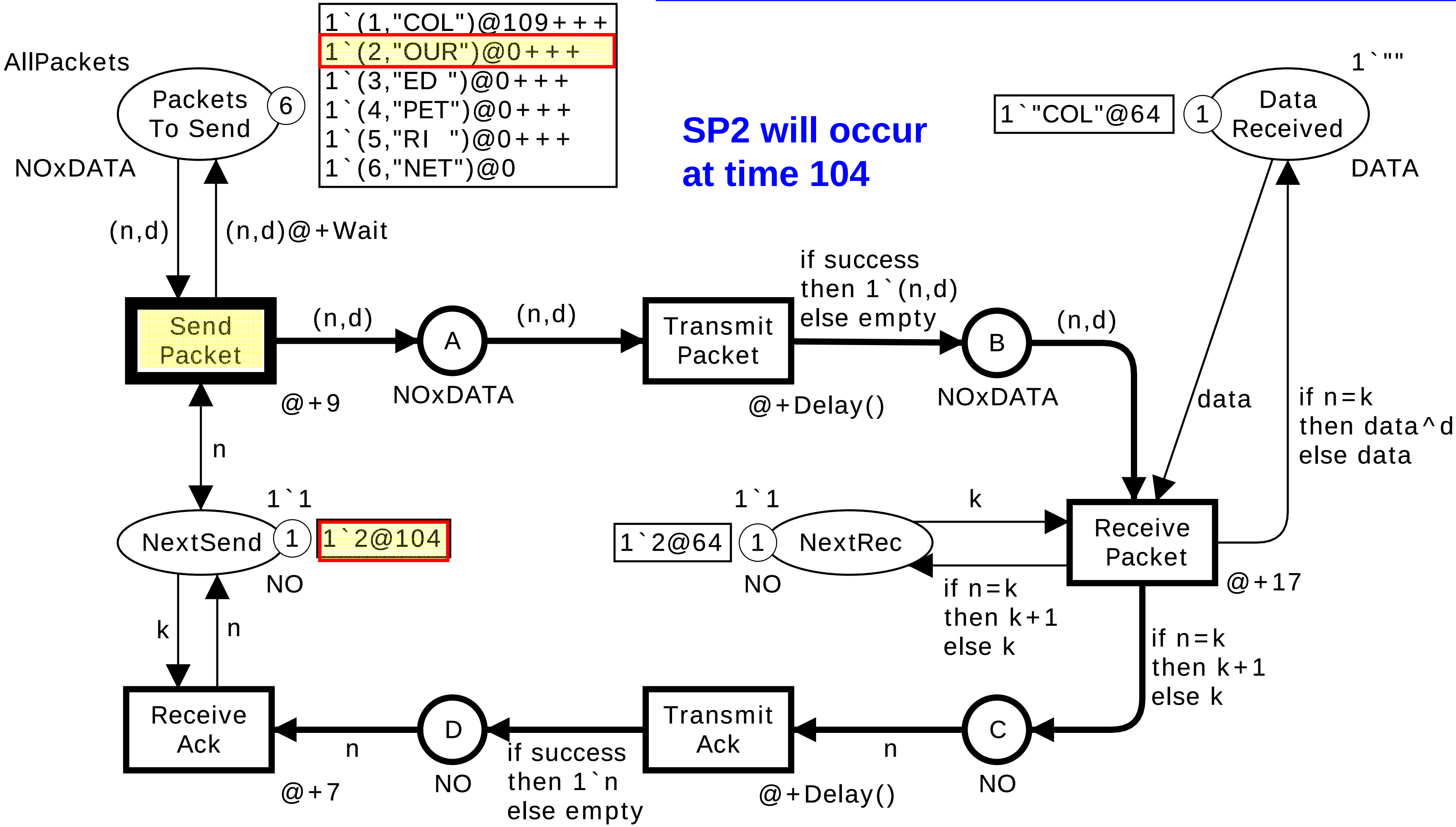
RA2 = (ReceivePacket, <n=2, k=1>) **97**



Marking M_5

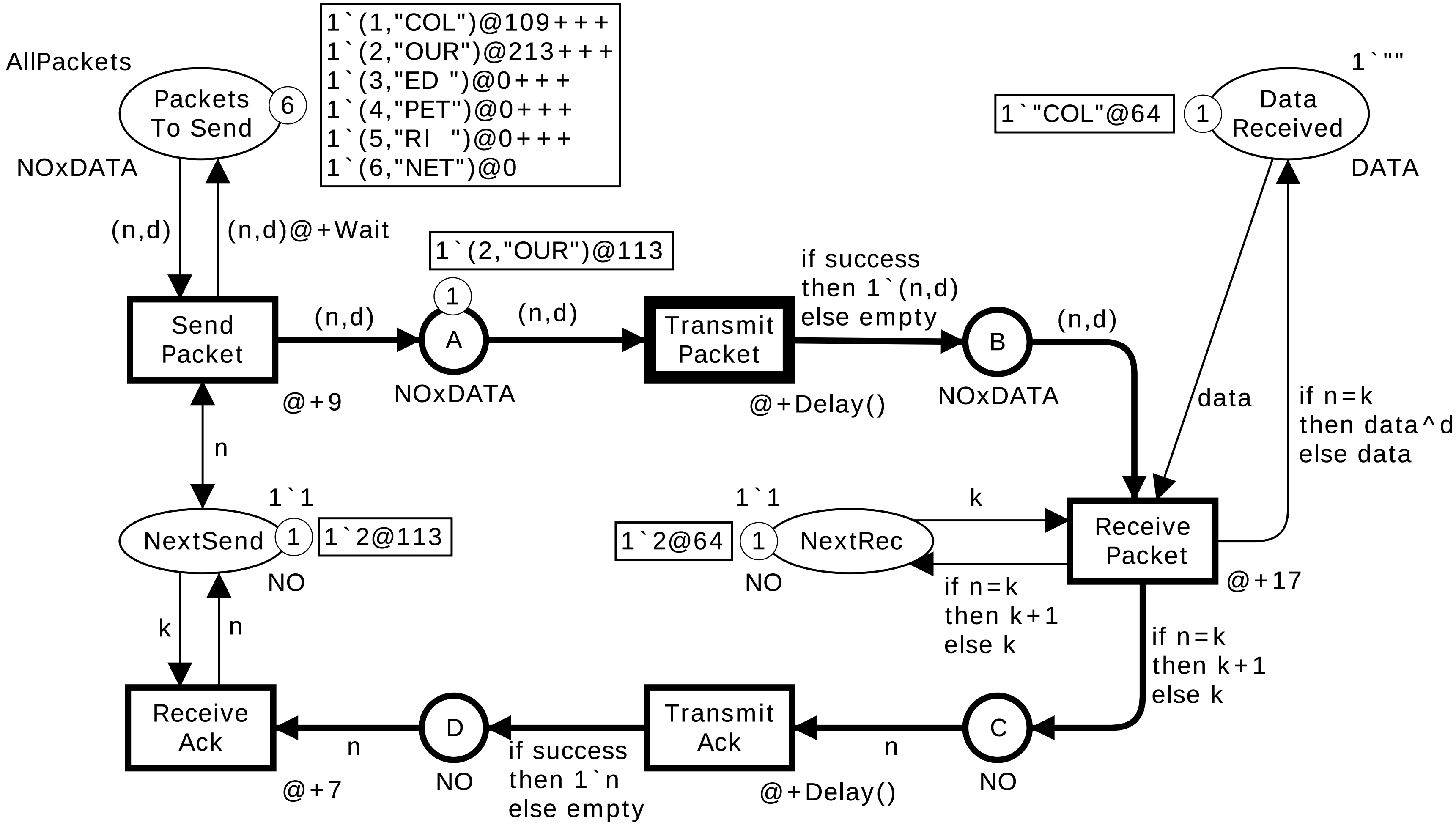
SP2 = (SendPacket, <n=2, d="OUR">) 104

SP2 will occur at time 104



Modelling and Validation of Distributed Systems Group
Department of Computer Science

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
1`""
1`"COL"@64
1
Data Received
DATA
(n,d)
(n,d)@+Wait
1`(2,"OUR")@113
if success
then 1`(n,d)
else empty
Send Packet
(n,d)
1
A
(n,d)
Transmit Packet
B
(n,d)
@+9
NOxDATA
@+Delay()
NOxDATA
data
if n=k
then data^d
else data
n
1`1
1`1
k
NextSend
1
1`2@113
1`2@64
1
NextRec
Receive Packet
NO
NO
if n=k
then k+1
else k
@+17
k
n
if n=k
then k+1
else k
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.

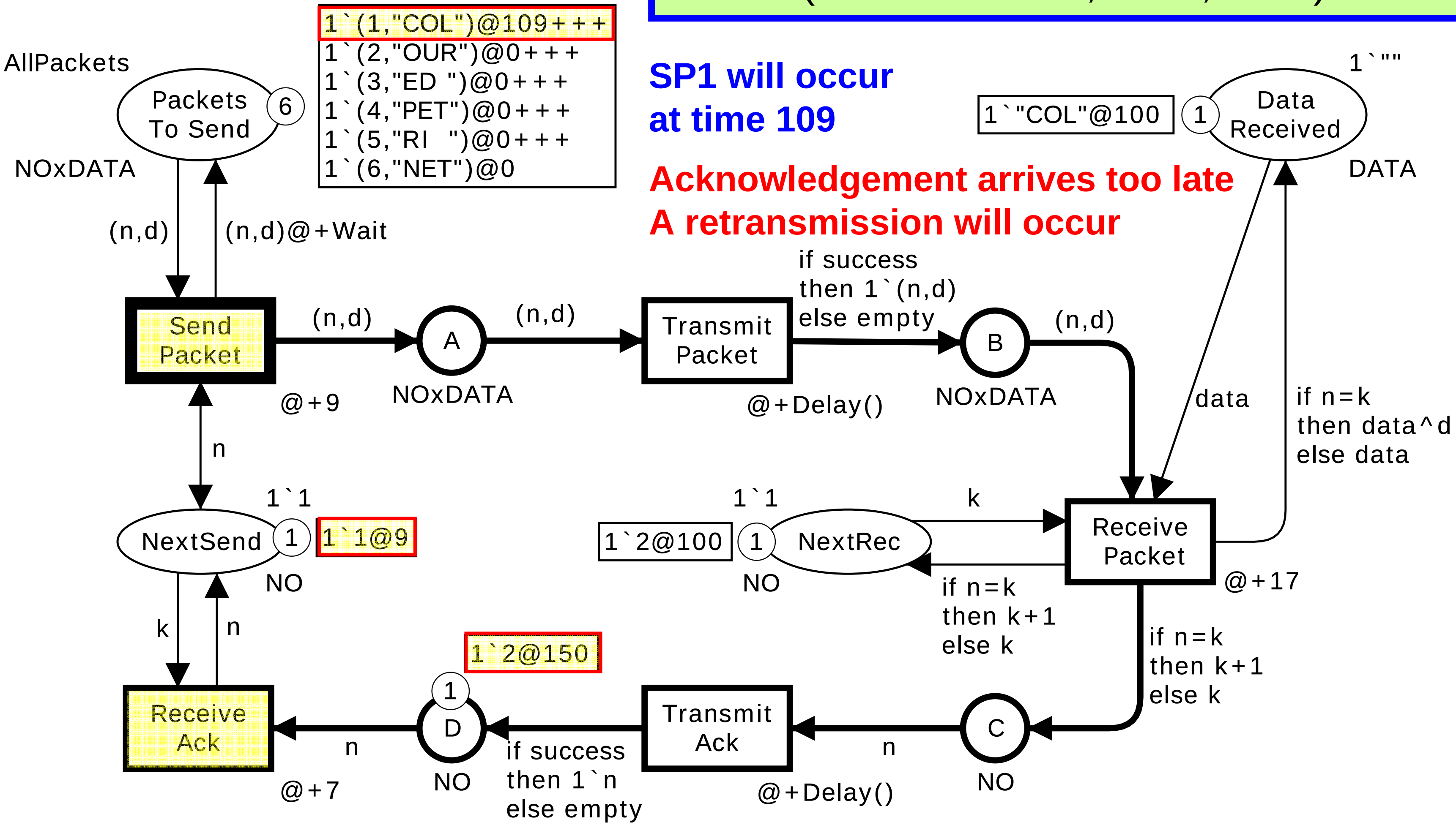
Marking $M_4{}^*$

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

RA2 = (ReceivePacket, <n=2, k=1>) **150**

SP1 will occur at time 109

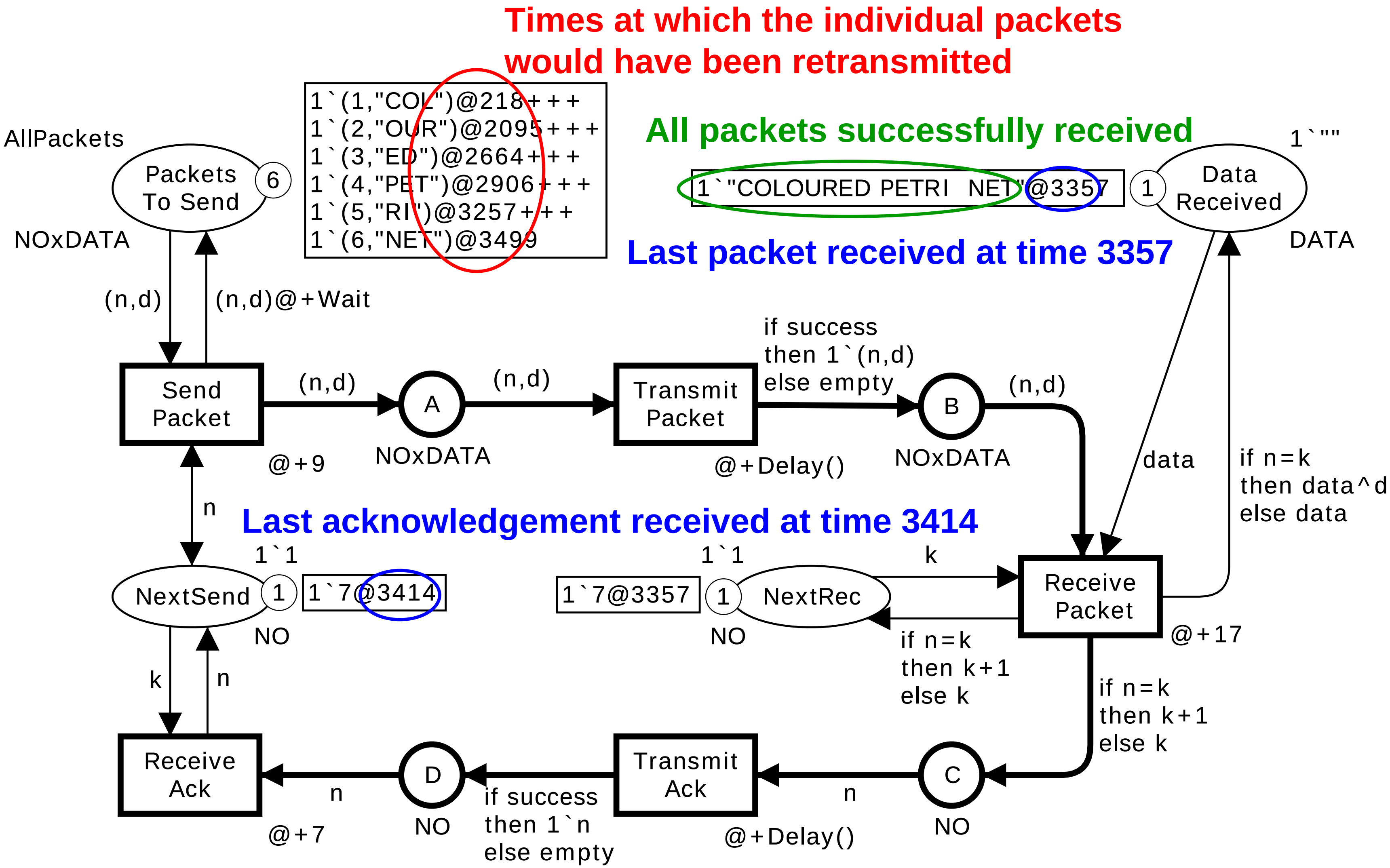
Acknowledgement arrives too late
A retransmission will occur



Modelling and Validation of Distributed Systems Group
Department of Computer Science

Kurt Jensen and Lars M. Kristensen

Dead marking at the end of simulation



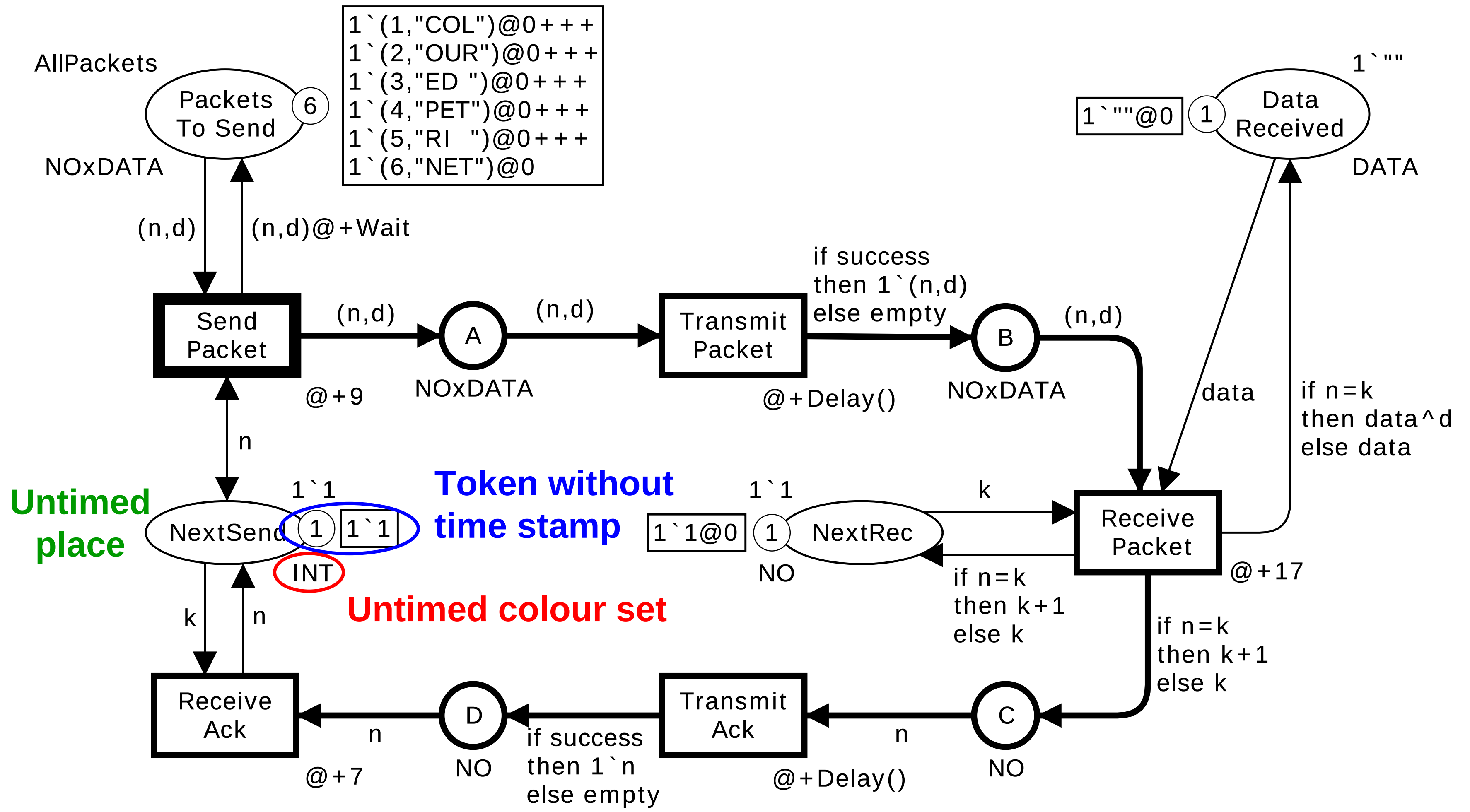
Non-deterministic simulation

- 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 time stamps will be different.

Slightly different model



Parallel sender operations

- The revised CPN model represents a system in which the sender can perform several SendPacket and ReceiveAck operations in parallel.
- The beginning of each operation is modelled by the occurrence of a transition.
- Several operations can start at the same moment of model time, or shortly after each other.

- In the original CPN model SendPacket and ReceiveAck have to wait for the timed token on place NextSend.
- Hence no other sender operation can begin until:
  - 9 time units after the beginning of a SendPacket operation.
  - 7 time units after the beginning of a ReceiveAck operation.

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.

Markings, conflicts and concurrency

- Each marking exists in a closed interval of model time – which may be a point, i.e., a single moment.

- As for untimed CPN models we may have conflicts and concurrency between binding elements (and binding elements may occur concurrently to themselves).
- This can only happen when the binding elements are enabled at the same moment of model time.

Behaviour of timed CPN models

- The behavioural properties of timed CPN models are defined in a similar way as in the untimed CPN case:
  - Occurrence sequences.
  - Reachability.
  - Integer and multi-set bounds.
  - Home markings, home spaces and home predicates.
  - Dead markings.
  - Dead and live transitions/binding elements.
  - Fairness properties.
- For multi-set bounds and home markings/spaces we consider the untimed markings – i.e. we ignore the time stamps.

Modelling and Validation of Distributed Systems Group
Department of Computer Science

Time delays are additional constraints

- Each timed CPN model determines an underlying untimed CPN model – obtained by removing all time delay inscriptions (and all time stamps).
- 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.

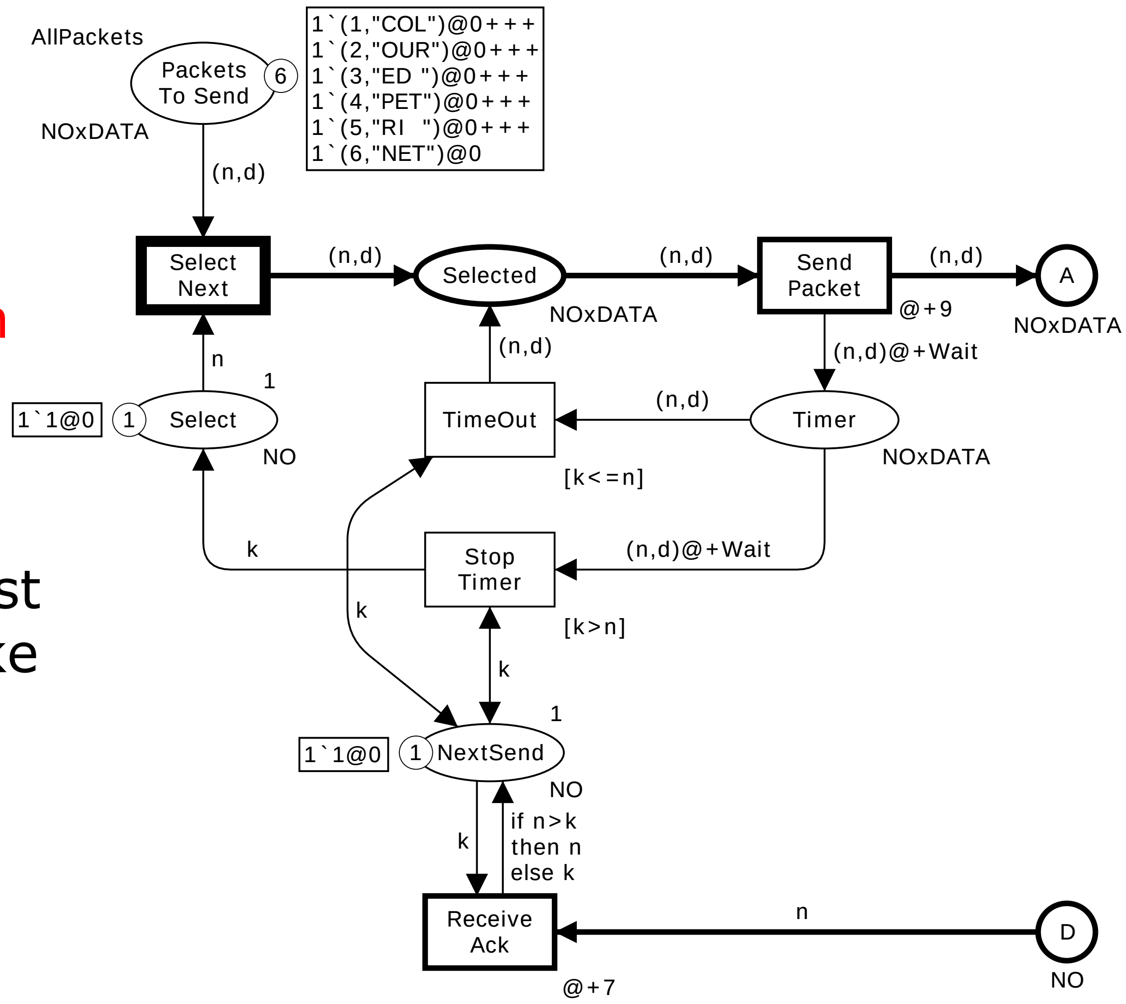
- This is done by adding time delays describing the duration of the individual operations.
- From our remarks above, it follows that these time delays cannot introduce new behaviour in the system – they merely restrict the possible occurrence sequences.

Revised protocol

- To illustrate other aspects of timed CPN models, we provide a modified version of the sender.
- We now explicitly model a timer controlling the retransmission of packets.
- We show how to use time delay inscriptions on input arcs.

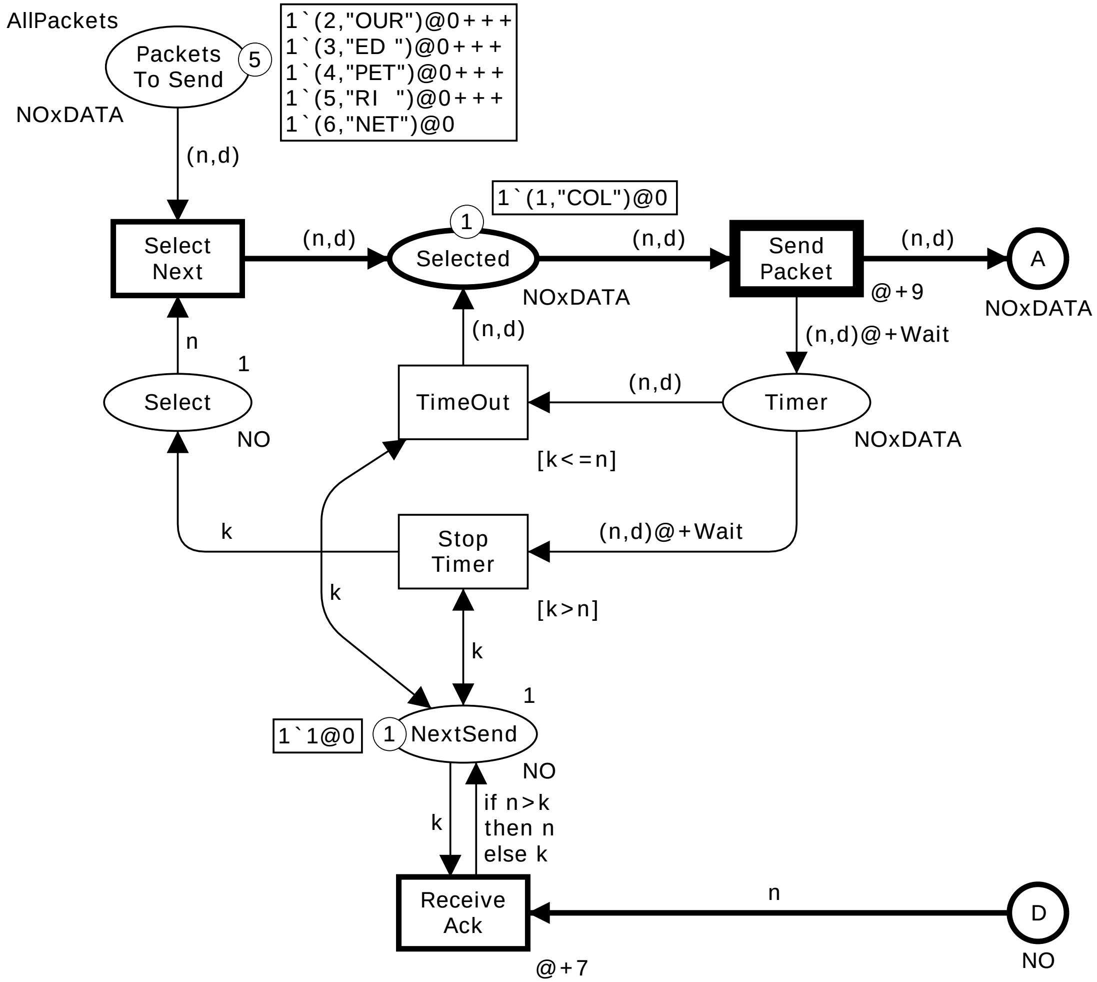
Modified sender – initial marking M_0

- SelectNext moves the next packet to be sent.
- The packet number is determined by the token colour on Select.
- SelectNext operation is instantaneous, i.e. so fast that it is modelled to take zero time.



Modified sender – marking M_1

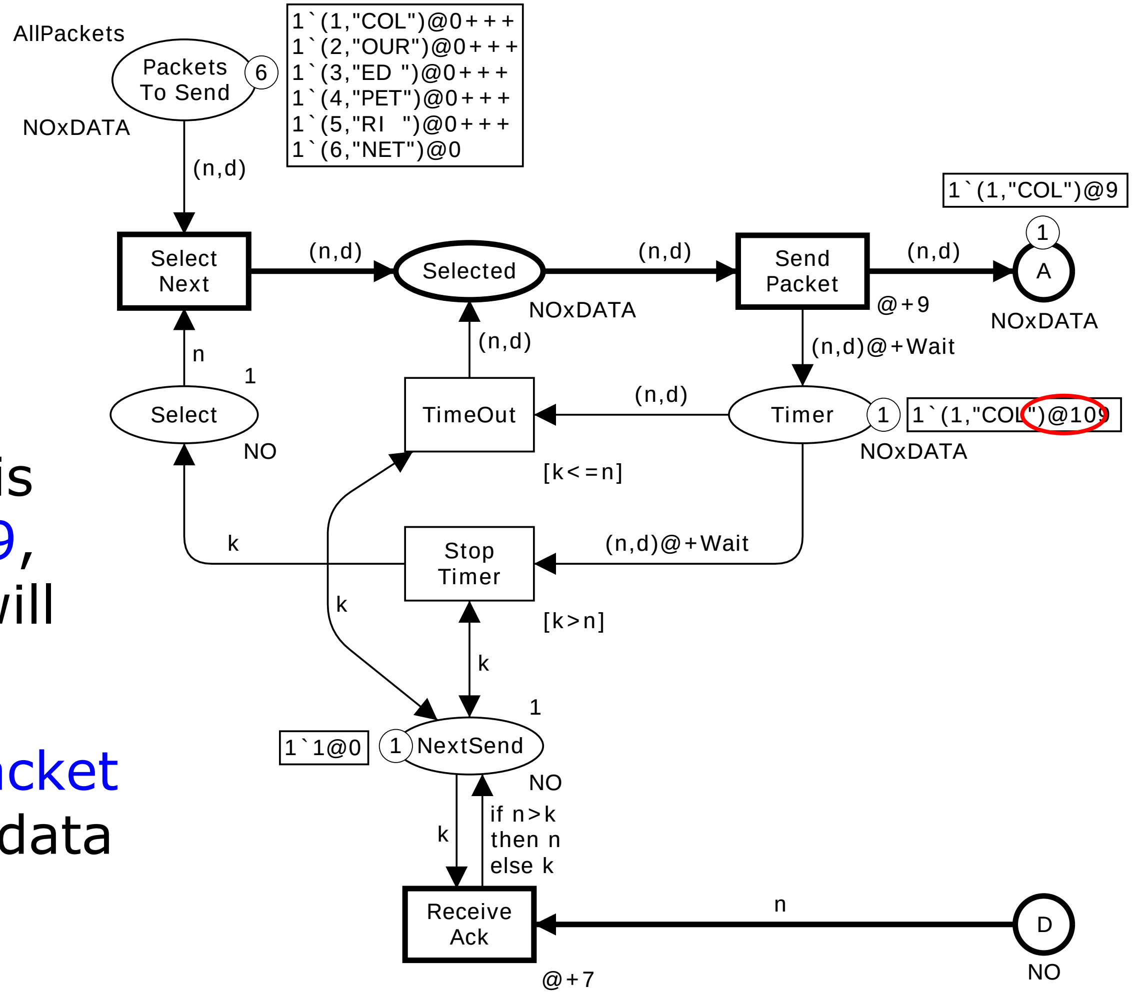
- Selected packet is ready to be sent by SendPacket.
- Simultaneously a token will be put on place Timer – modelling the start of a timer.



Modelling and Validation of Distributed Systems Group
Department of Computer Science

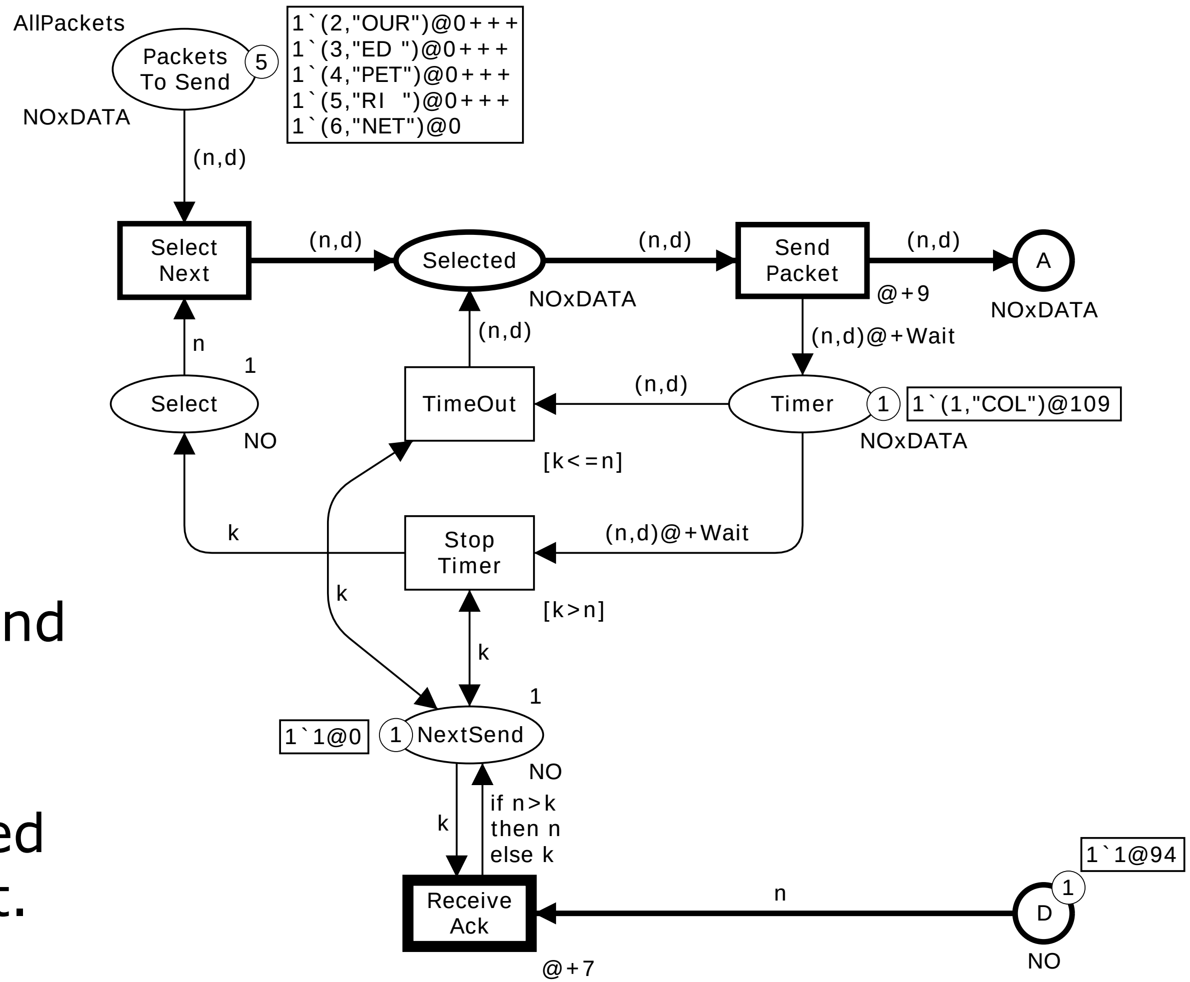
Modified sender – marking M_2

- Data packet no 1 has been sent.
- Timer has been started and it will expire at time 109.
- If no acknowledgement is received before time 109, the TimeOut transition will occur.
- This will enabled SendPacket and a retransmission of data packet no 1 will occur.



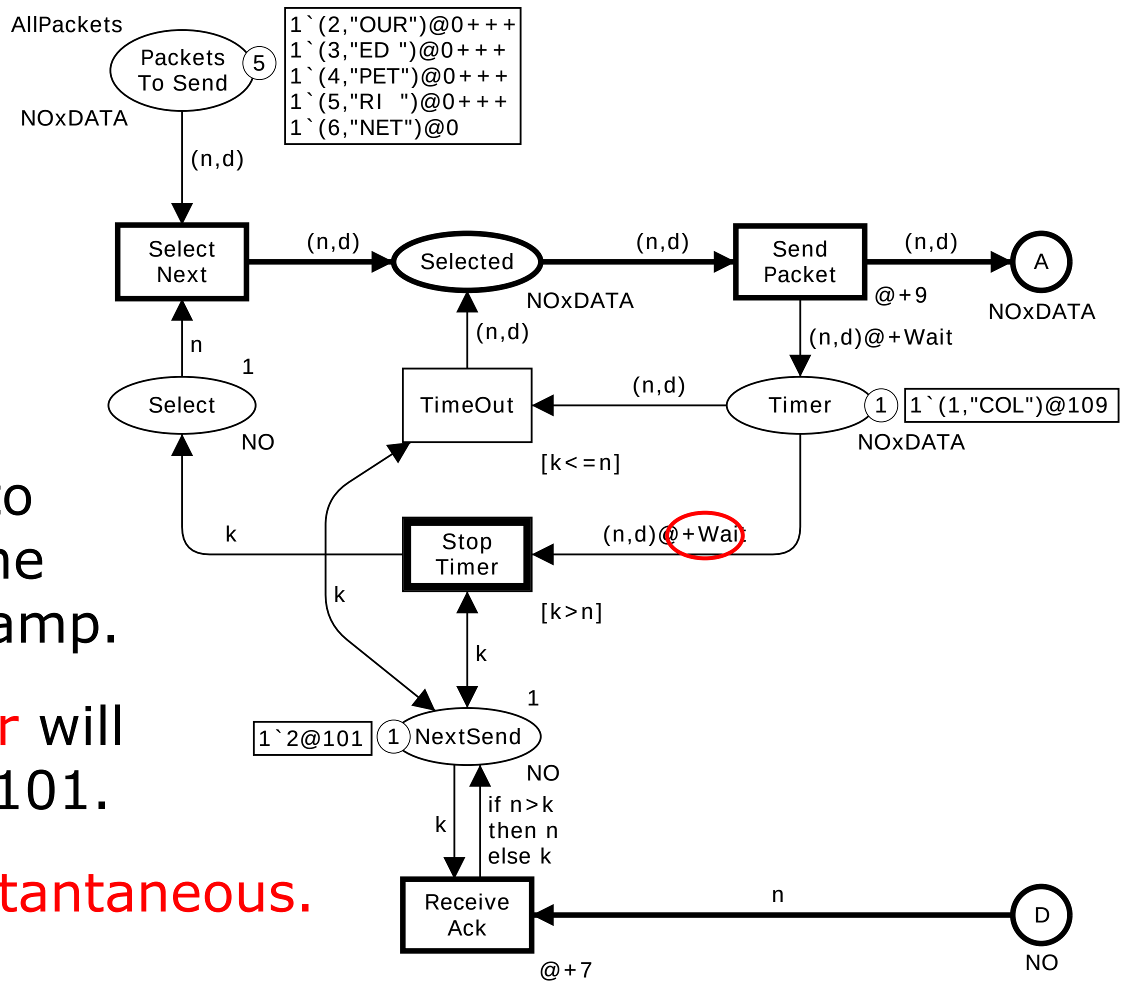
Modified sender – marking M_3

- Acknowledgement requesting packet no 2 is arriving at time 94.
- ReceiveAck will occur and update NextSend.
- The new value of NextSend is the highest value of n and k.
- This means that NextSend never is decreased.
- It holds the highest packet number requested by an acknowledgement.



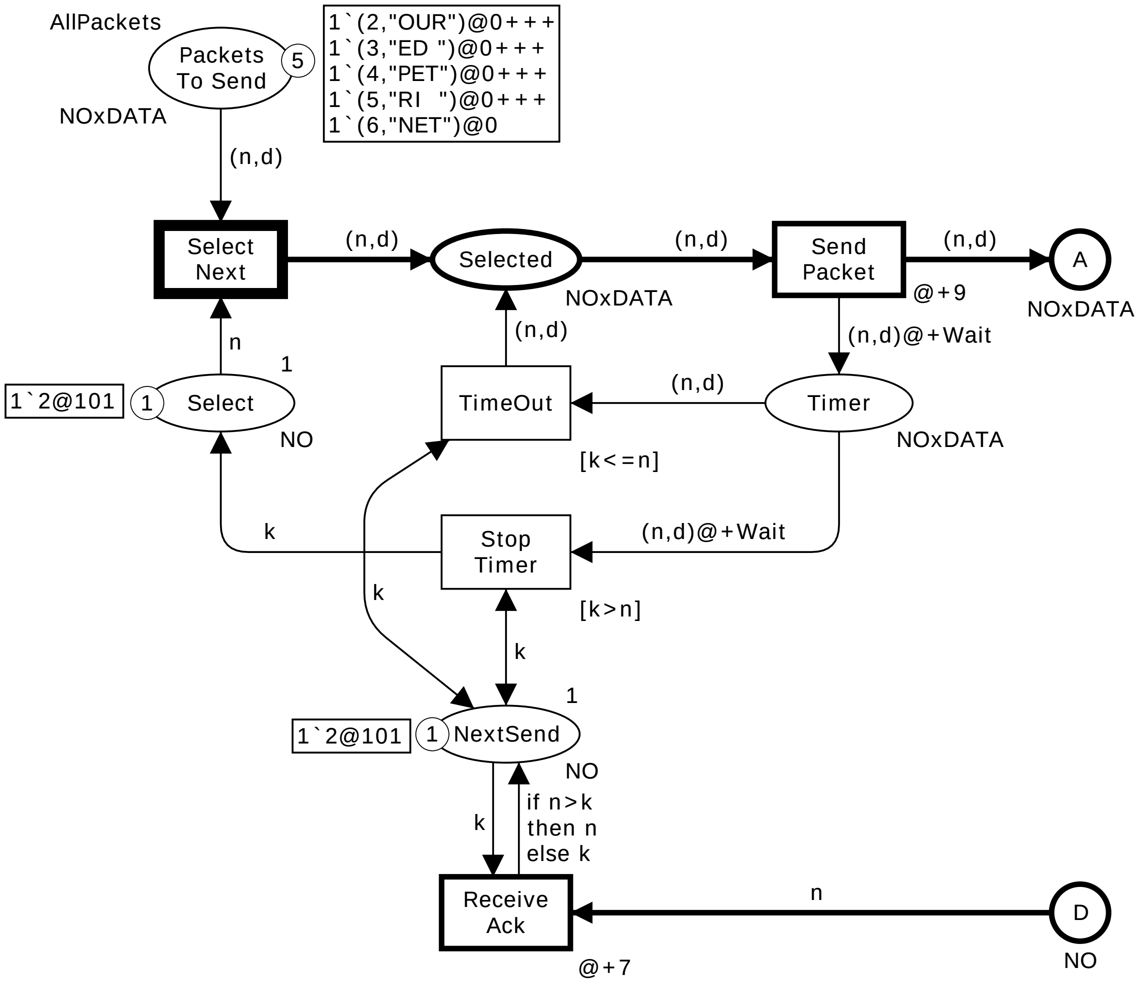
Modified sender – marking M_4

- StopTimer will remove a token with time stamp 109 from place Timer.
- The input arc from Timer has its own time delay inscription.
- This allows the transition to remove the token Wait time units ahead of the time stamp.
- This means that StopTimer will be ready to occur at time 101.
- StopTimer operation is instantaneous.



Modified sender – marking M_5

- Timer has been stopped.
- Select has been updated to the packet number specified by NextSend.
- Data packet no 2 can now be selected at time 101.



Modelling and Validation of Distributed Systems Group
Department of Computer Science
Kurt Jensen and Lars M. Kristensen

Time delay inscriptions on input arcs

- The use of time delay inscriptions on input arcs allow us to remove tokens ahead of time.
- This can be very useful to model the interruption of an ongoing operation – e.g. a running timer.
- Without such a facility one would need additional bookkeeping – to remember that a token should be removed when the model time becomes high enough to allow this.
- When a time delay inscription is used on a double arc, it is used for both the input arc and the output arc.
- If the time delay inscription only should be used for one of the arcs, it is necessary to draw two separate arcs.

State spaces for timed CPN models

- State spaces for timed CPN models are defined in a similar way as state spaces for untimed models.

- Each state space node represents a timed marking:
  - Timed multi-set for each timed place.
  - Untimed multi-set for each untimed place.
  - Global clock.

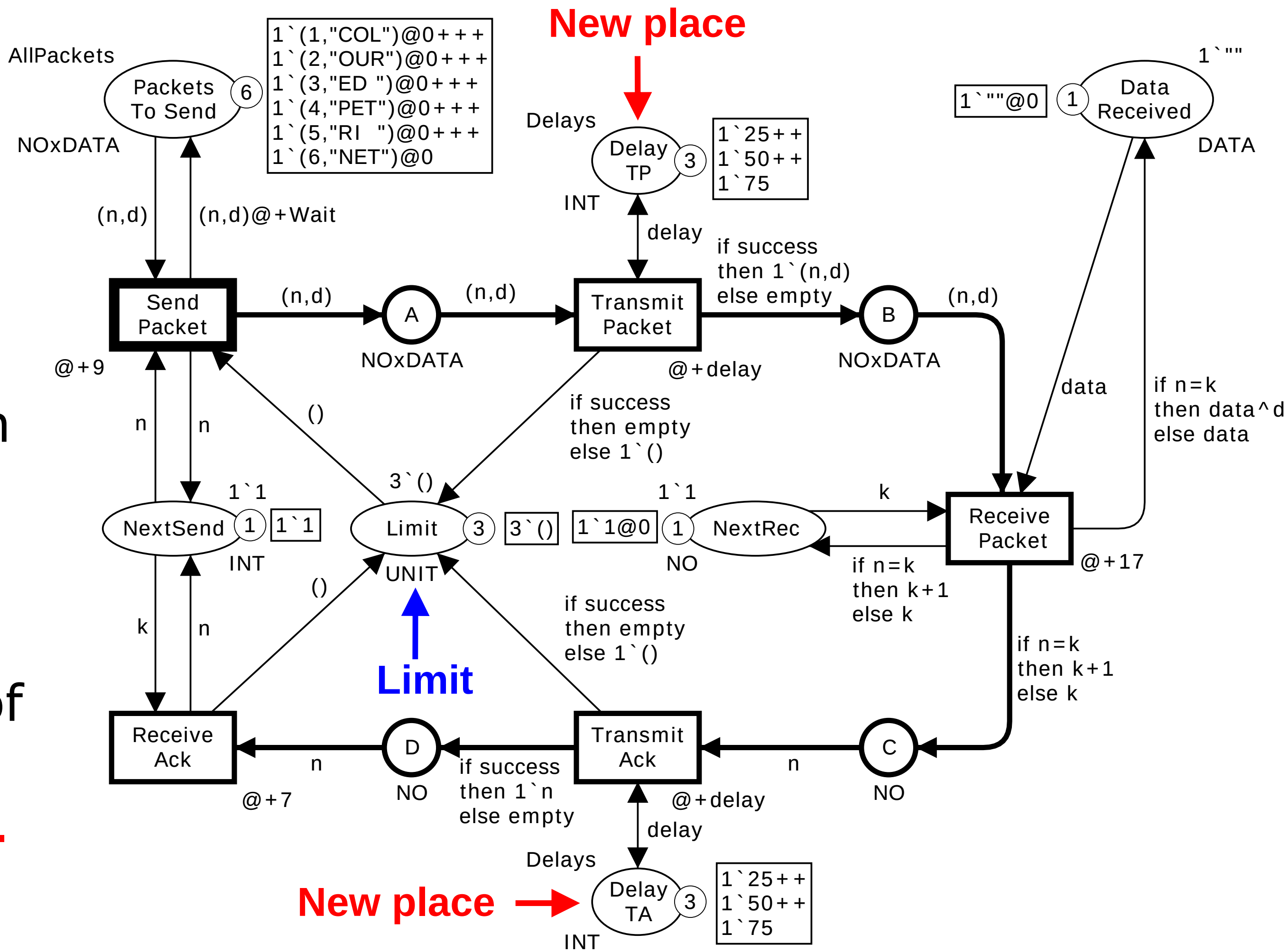
- CPN Tools supports construction and analysis of state spaces for timed CPN models.

State spaces for timed CPN models

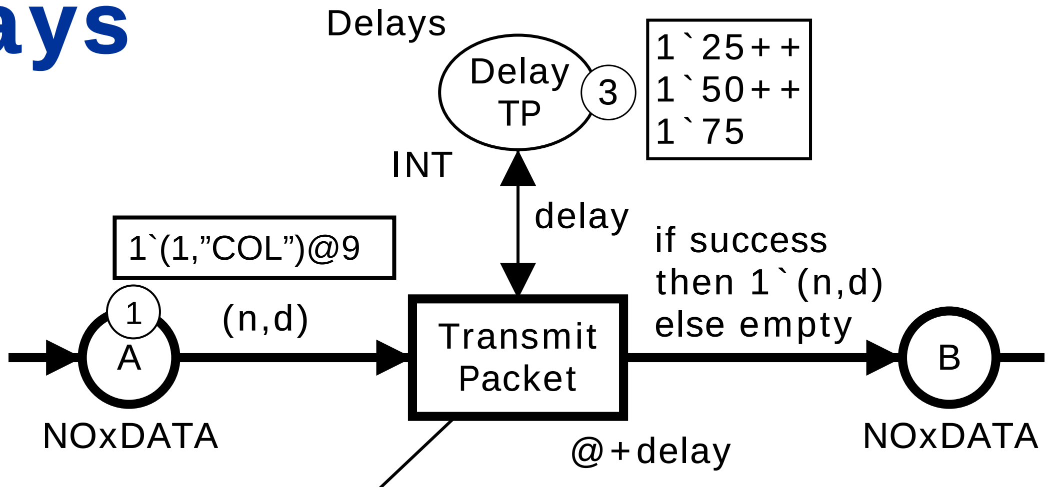
- $\text{OccSeq}_{\text{Timed}} \subseteq \text{OccSeq}_{\text{Untimed}}$
- This implies that the nodes in a timed state space have fewer out-going arcs (lower out-degree).
- Two timed markings may have:
  - Identical token colours.
  - Different time stamps and different values for the global clock.
- This implies that a timed state space may have more nodes.
- The timed state space may be infinite even though the untimed state space is finite.
- It is possible to use equivalence classes to reduce the number of nodes in a timed state space.

Timed protocol for state spaces

- Two new places specify the possible delays.
- For state spaces we cannot use the Delay function returning a random value.
- This would make the construction of the state space non-deterministic.



Modelling of delays



- To reduce the size of the state space we want fewer possible delays.
- In this case we only have three.

- At time 9 we have six enabled bindings.
- All of them are in conflict with each other.

<n=1, d="COL", success=true, delay=25>
<n=1, d="COL", success=true, delay=50>
<n=1, d="COL", success=true, delay=75>
<n=1, d="COL", success=false, delay=25>
<n=1, d="COL", success=false, delay=50>
<n=1, d="COL", success=false, delay=75>

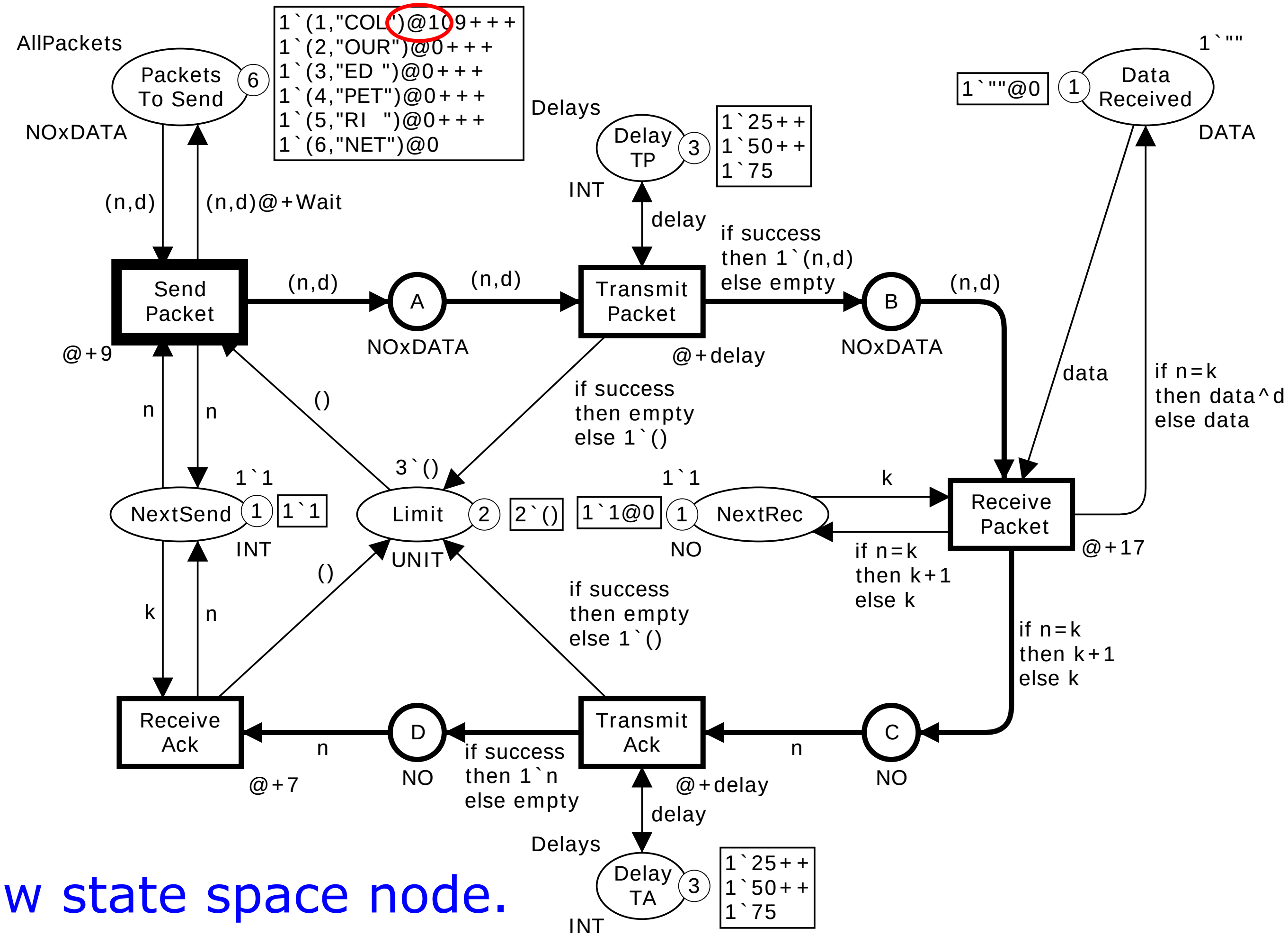
Success (first three bindings)

Failure (last three bindings)

- Copy of data packet no 1 on place B with time stamp 59.

New marking – similar to initial marking

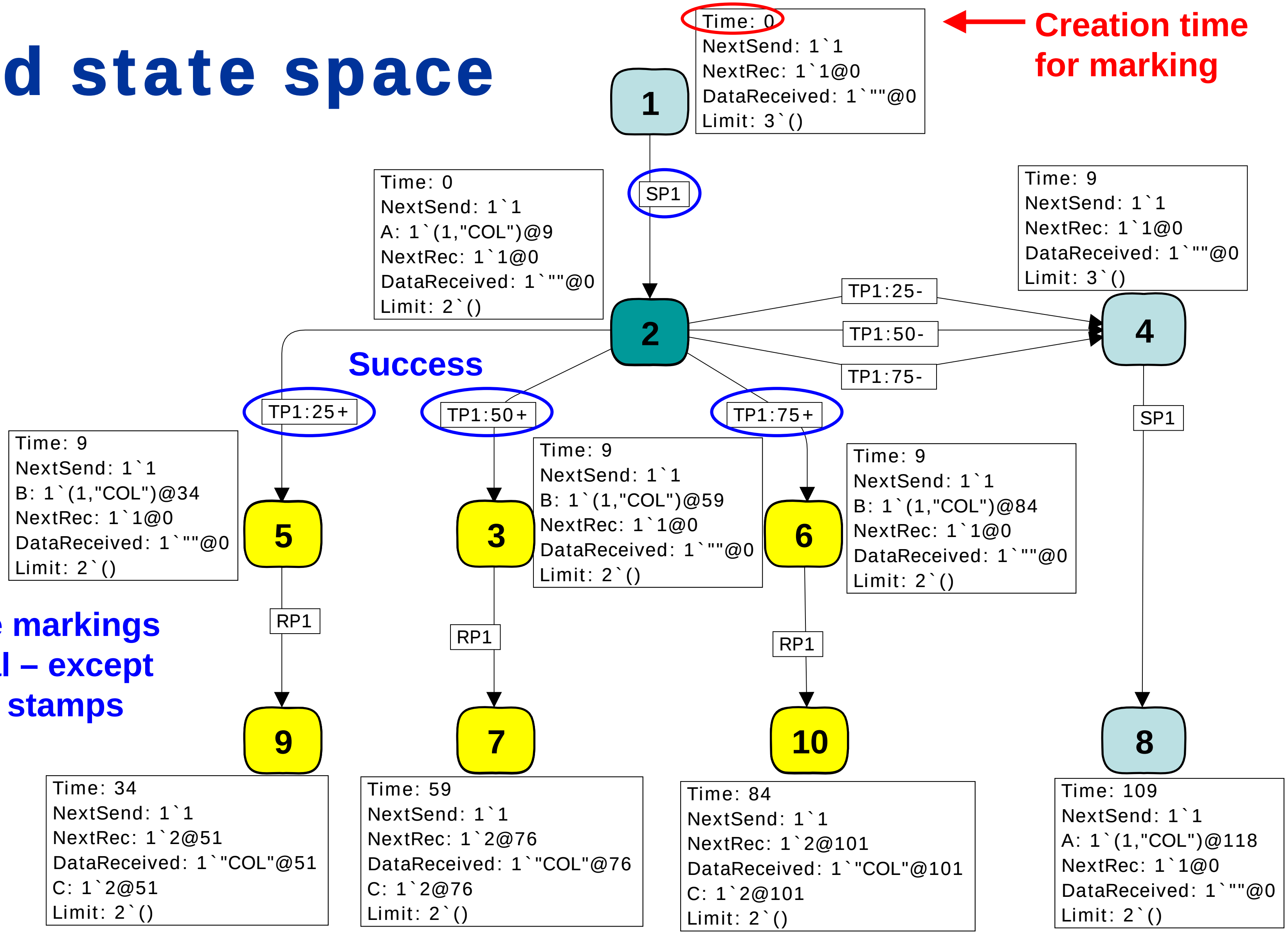
- One of the time stamps on PacketsToSend has been increased.



- New marking ⇒ New state space node.

Modelling and Validation of Distributed Systems Group
Department of Computer Science
Kurt Jensen and Lars M. Kristensen

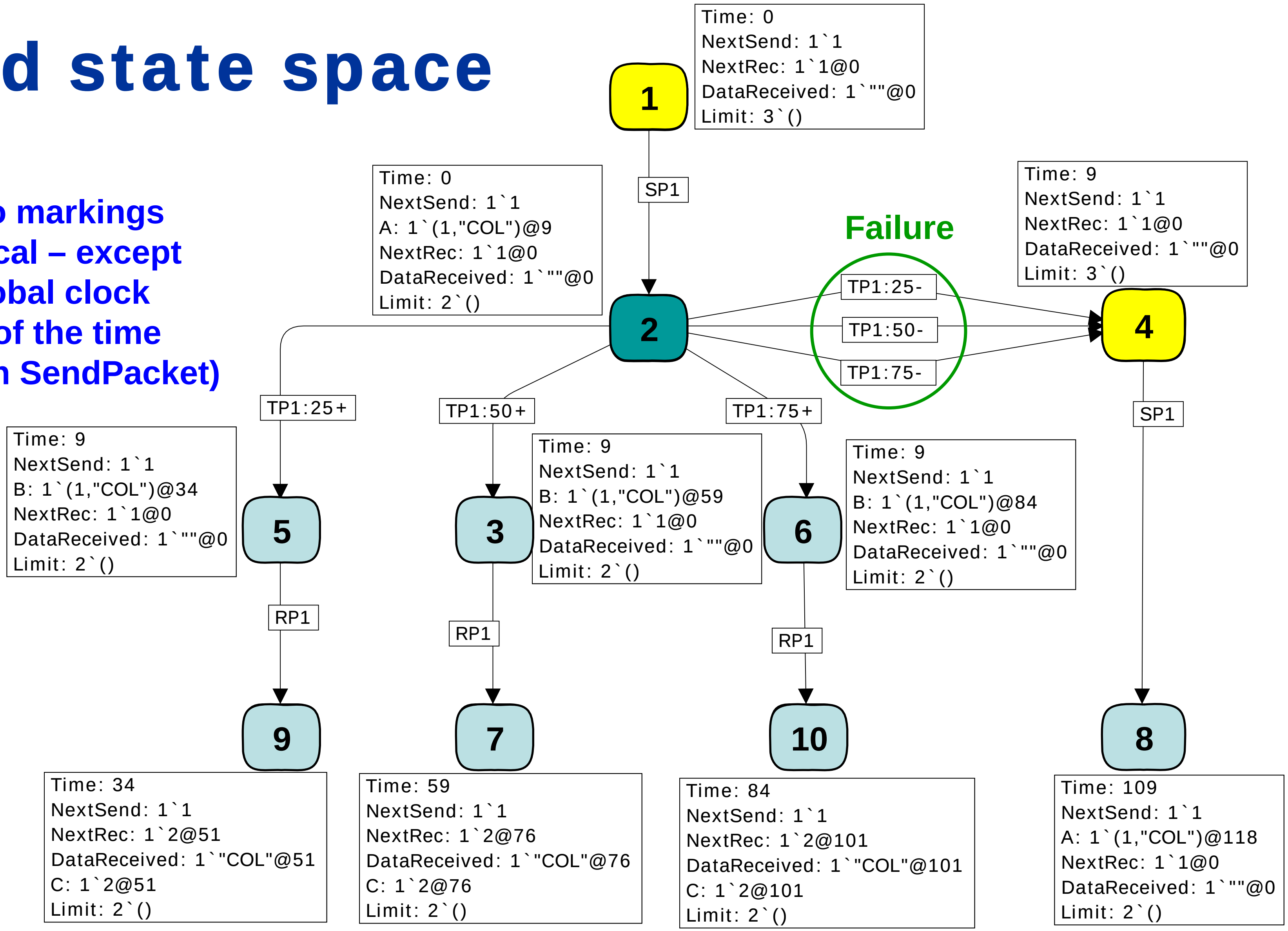
Timed state space



These three markings are identical – except for the time stamps

Timed state space

These two markings are identical – except for the global clock (and one of the time stamps on SendPacket)



Size of timed state space

- We may continue to send and lose the first data packet.
- Hence, the timed state space is infinite.
- To obtain a finite state space we put an upper bound on the value of the global clock.
- In this way we obtain a partial state space.
- We can, e.g., investigate whether a certain packet has arrived before a certain time.

Clock	Nodes	Arcs
10	12	19
20	48	87
30	156	269
40	397	644
50	814	1,273
60	3,005	4,583
70	7,822	12,154
80	17,996	28,002
90	49,928	79,224
100	103,377	165,798

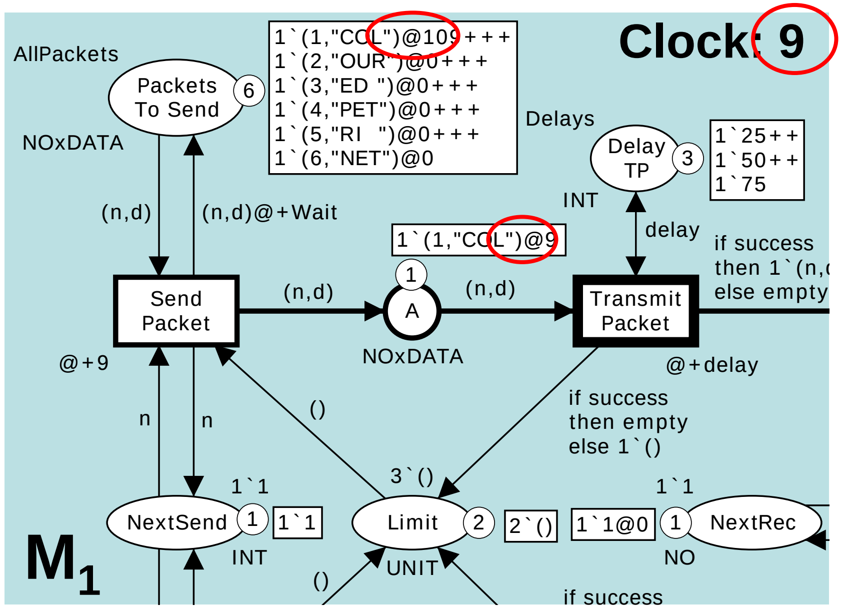
Time equivalence method

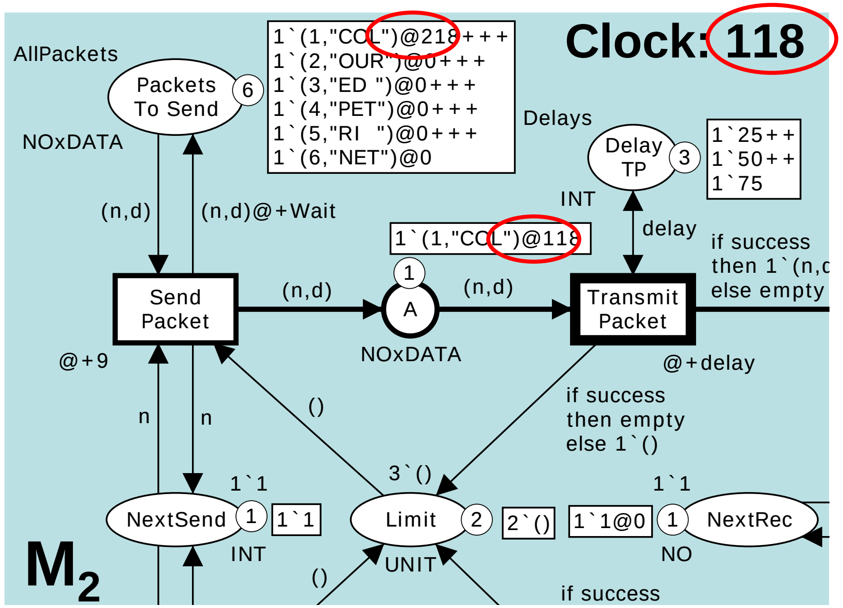
- Special case of the equivalence method from Chapter 8.

- The basic idea is to replace the absolute time values in the global clock and time stamps with relative values.
- The reduced state space is finite whenever the state space of the underlying untimed CPN model is finite.
- From the reduced state space it is possible to verify all behavioural properties of the timed CPN model.

Markings have different time values

- The state spaces of timed CPN models become infinite for models with cyclic behaviour.
- The time values (in the global clock and the time stamps) continue to increase.





Canonical representative

- We consider markings such as M_1 and M_2 to be equivalent and we compute a canonical representative for each equivalence class:

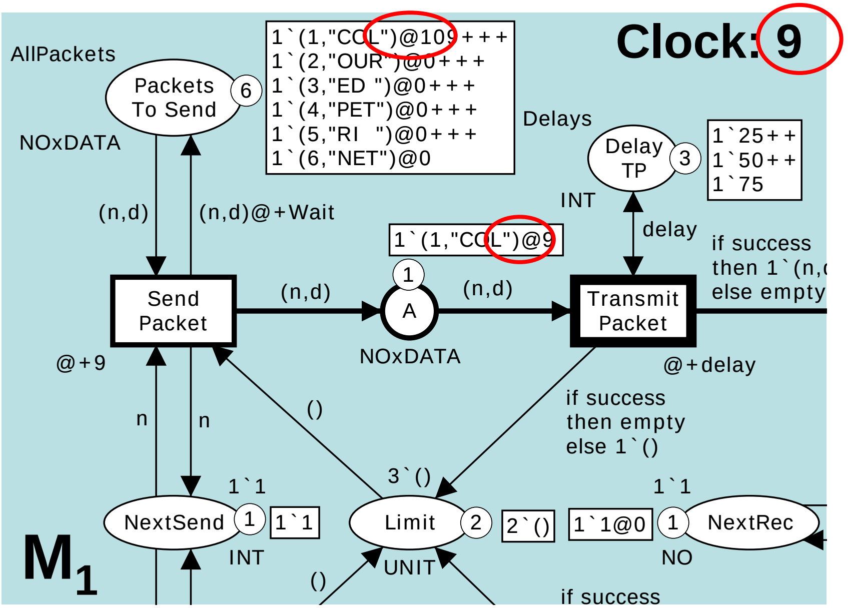
$$\text{time\_value} \rightarrow \max(\text{time\_value} - \text{current\_time}, 0)$$

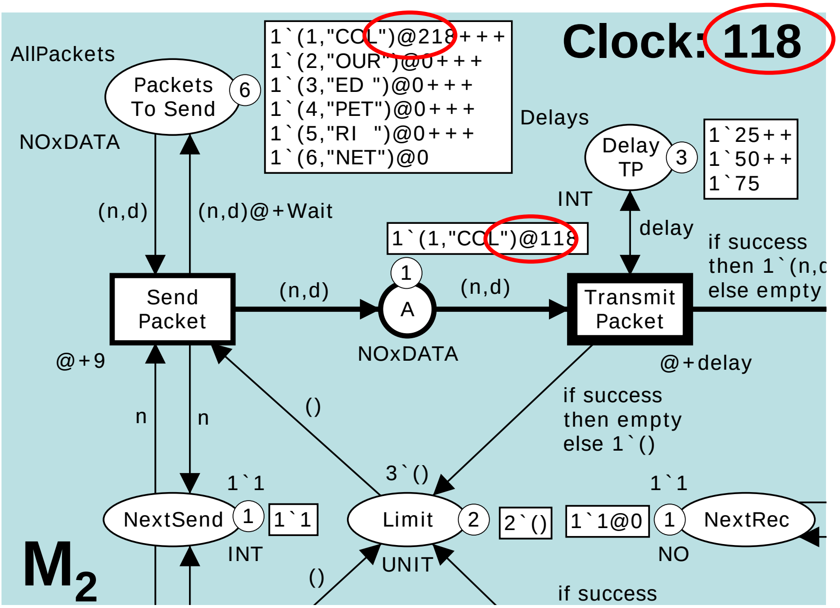
- All time stamps which are less than the current model time is set to zero (they cannot influence enabling).
- The current model time is subtracted from all time stamps which are greater than or equal to the current model time.
- The current model time i set to zero.

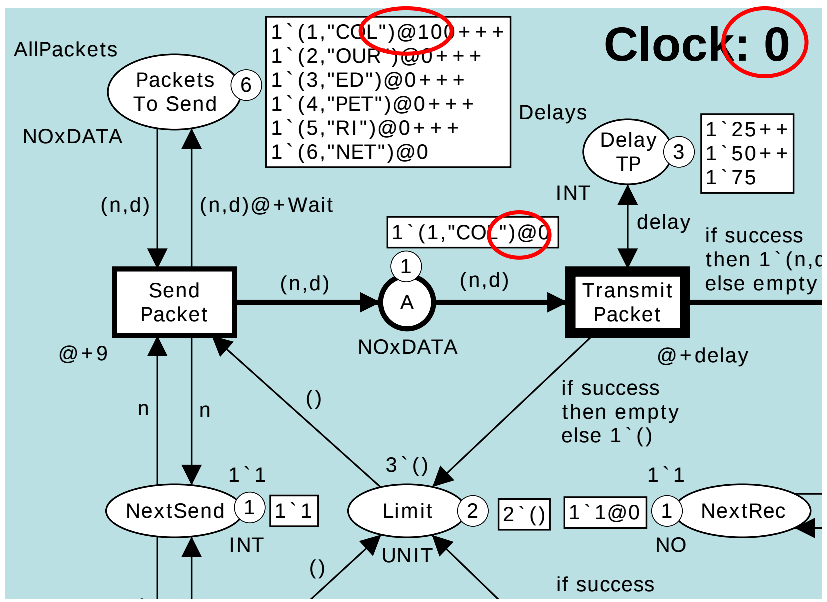
Modelling and Validation of Distributed Systems Group
Department of Computer Science

Kurt Jensen and Lars M. Kristensen

Canonical representative







The two markings have the same canonical representation – hence they are equivalent

Condensed state spaces

- Condensed state spaces for a timed CPN model can be computed fully automatically.
  - Consistency of the equivalence has been proven once and for all CPN models.
  - Predicates for the equivalence tests are implemented in CPN Tools once and for all CPN models.

- All properties expressible in the real-time temporal logic RCCTL$\ast$ are preserved in the time condensed state space.
- This includes all standard behavioural properties of CPN models.

Statistics for time equivalence method

Limit	Packets	Nodes	Arcs	Limit	Packets	Nodes	Arcs
1	10	81	110	5	2	88,392	158,351
1	20	161	220	5	4	307,727	556,212
1	50	401	550	7	1	13,198	23,400
1	100	801	1,100	7	2	145,926	285,848
2	5	3,056	4,424	7	3	323,129	657,650
2	10	6,706	9,694	10	1	20,062	36,250
2	20	14,006	20,234	10	2	244,990	502,916
2	50	35,906	51,854	12	1	24,630	44,802
3	1	2,699	4,126	12	2	335,651	697,127
3	5	85,088	129,858	13	1	26,914	49,078
3	15	306,118	466,408	13	2	391,743	818,021