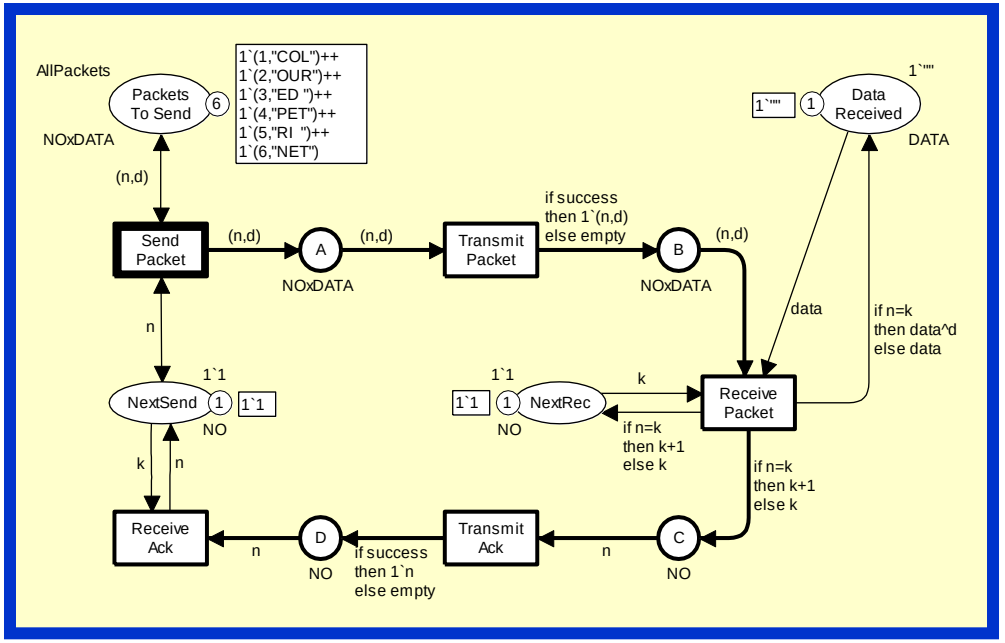
Coloured Petri Nets

Modelling and Validation of Concurrent Systems

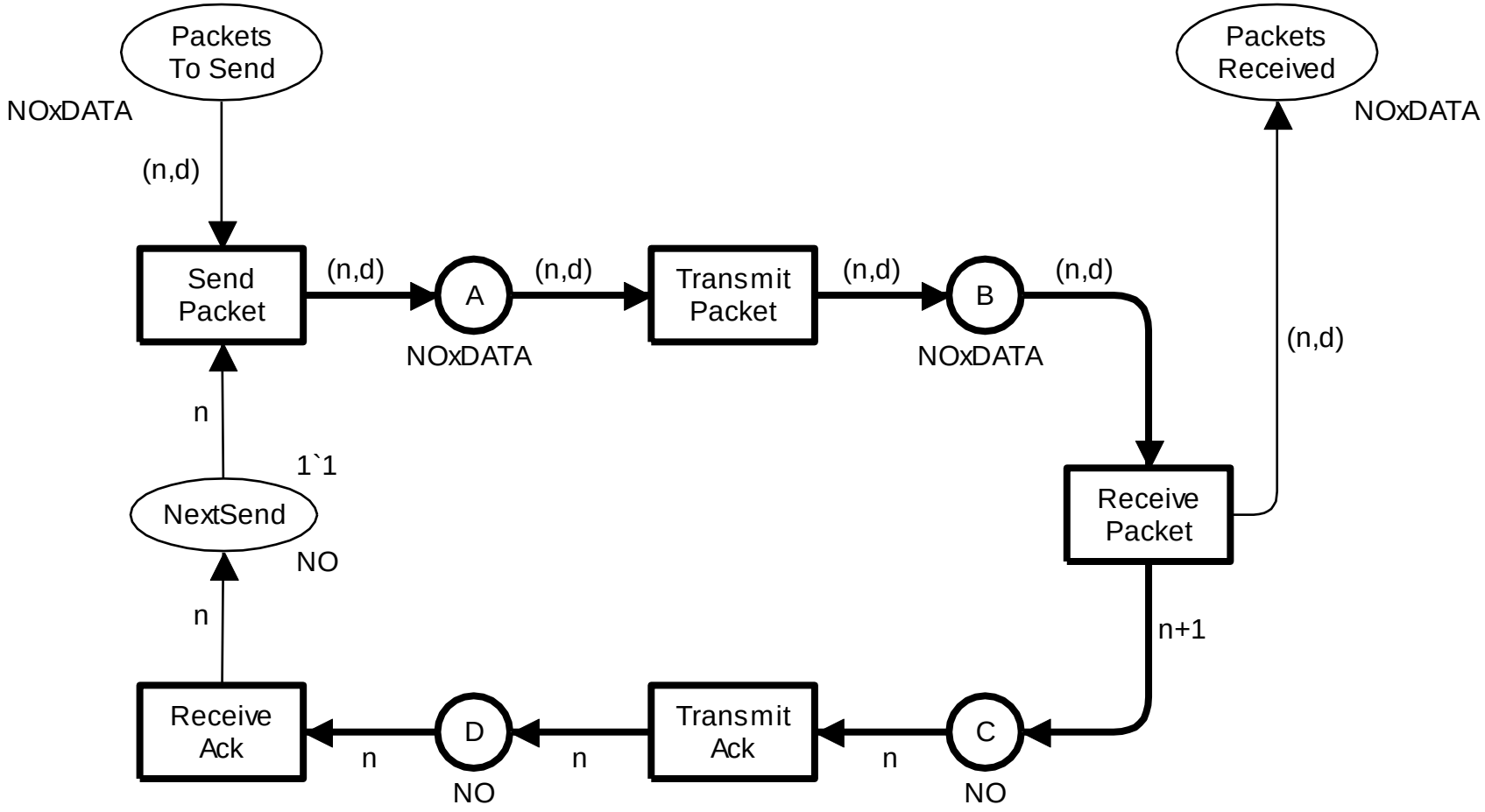
Chapter 2:
Non-hierarchical Coloured Petri Nets

Kurt Jensen &
Lars Michael Kristensen

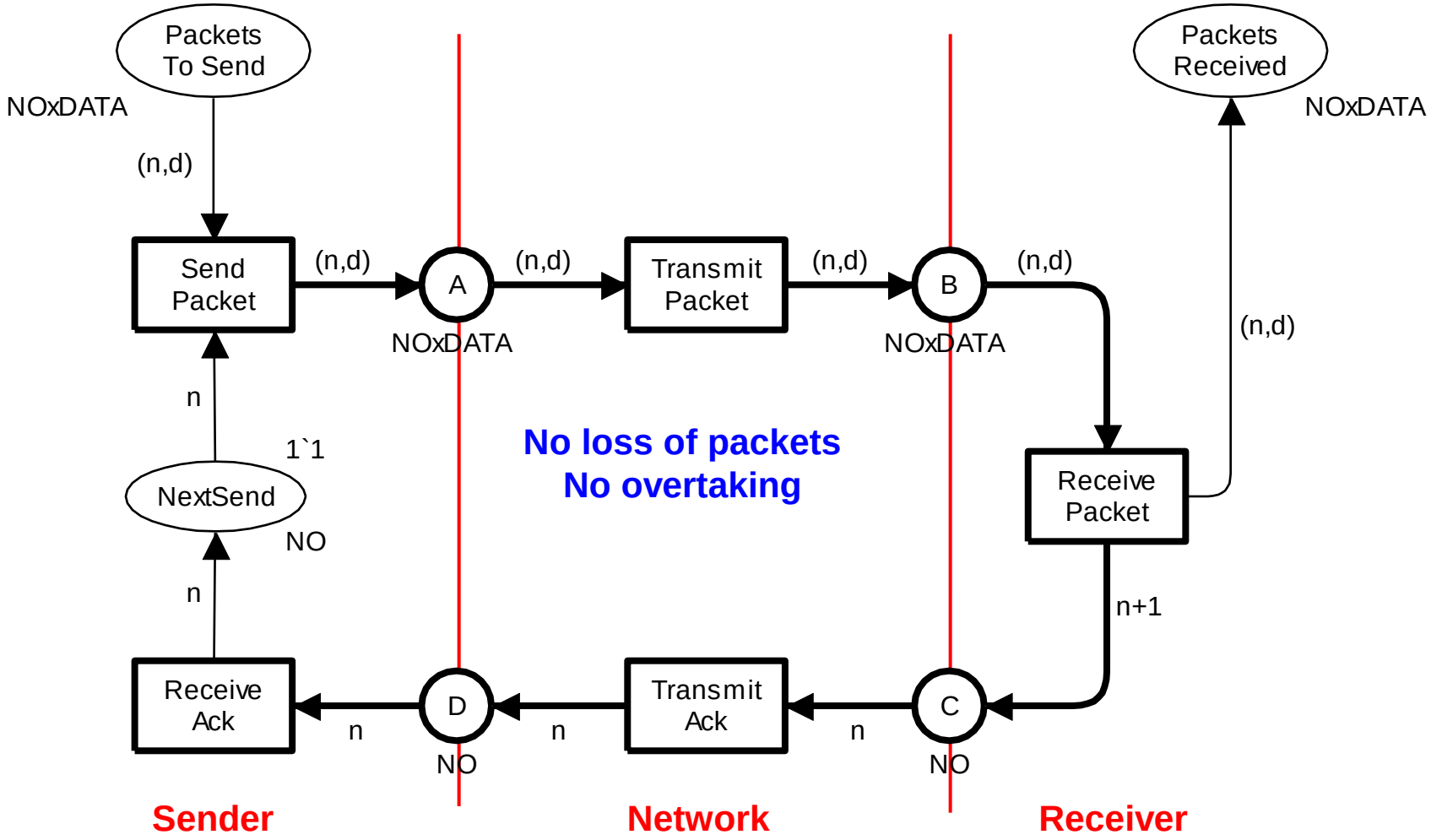
{kjensen,lmkristensen}
@cs.au.dk



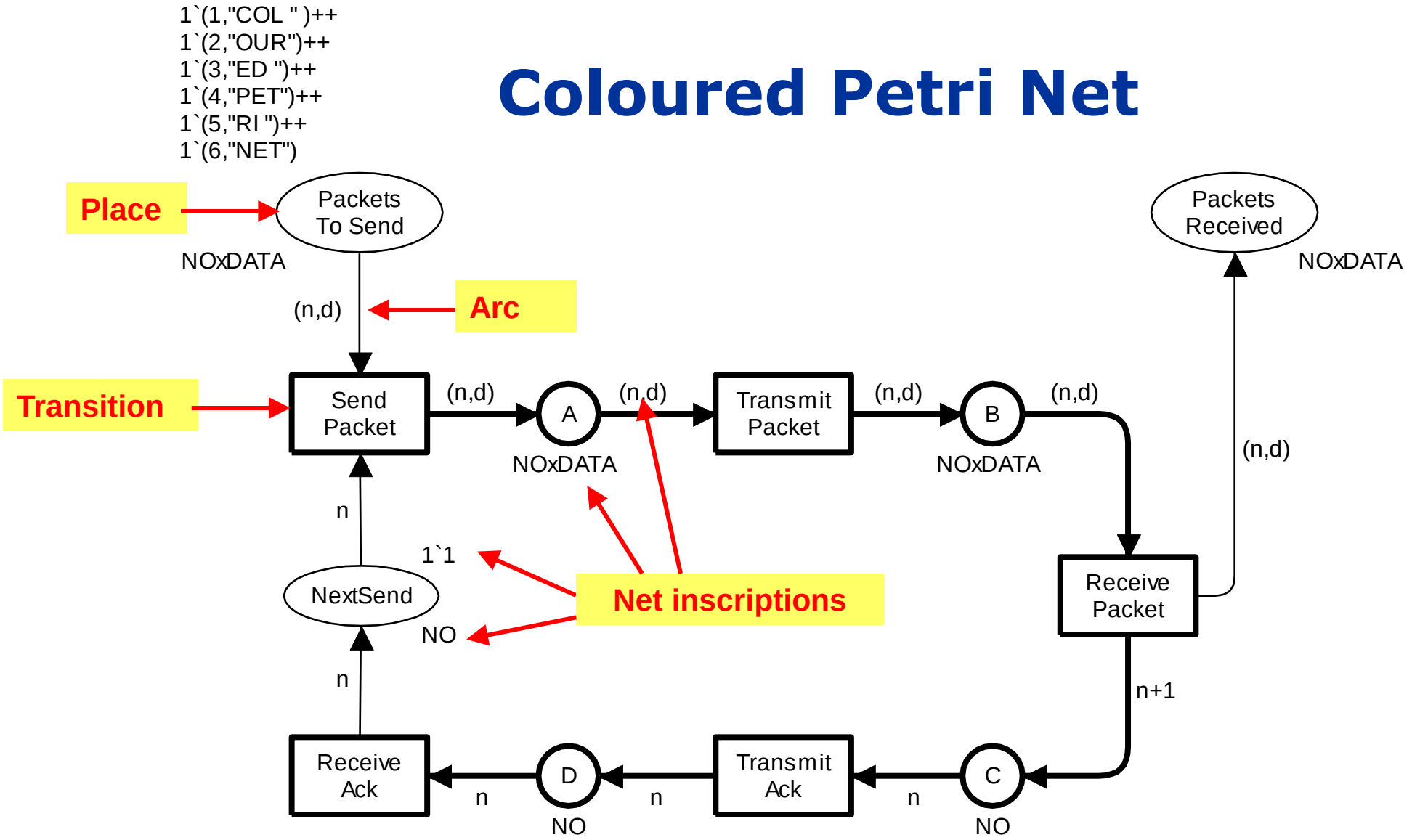
Simple protocol



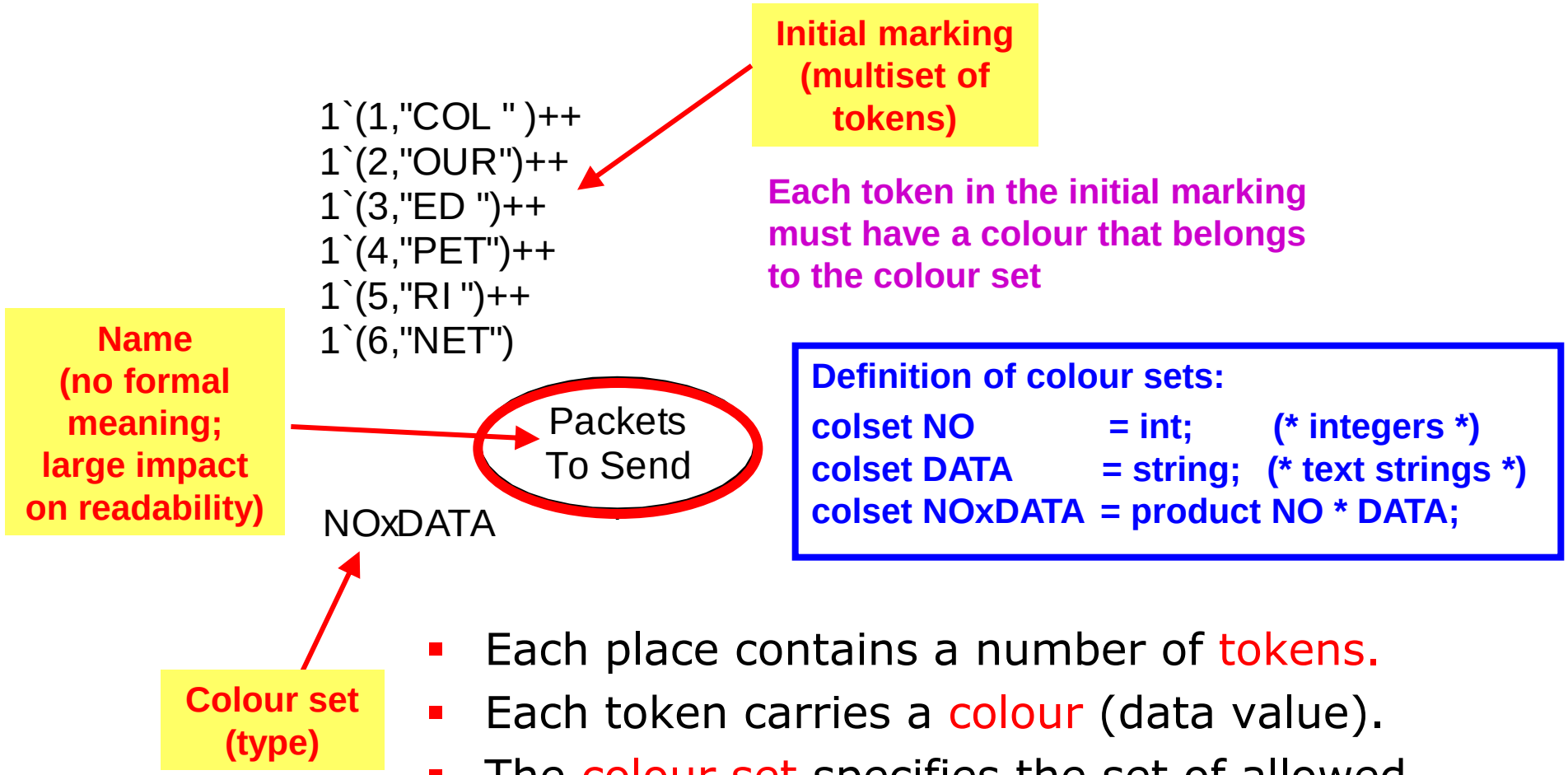
Informal description



Coloured Petri Net

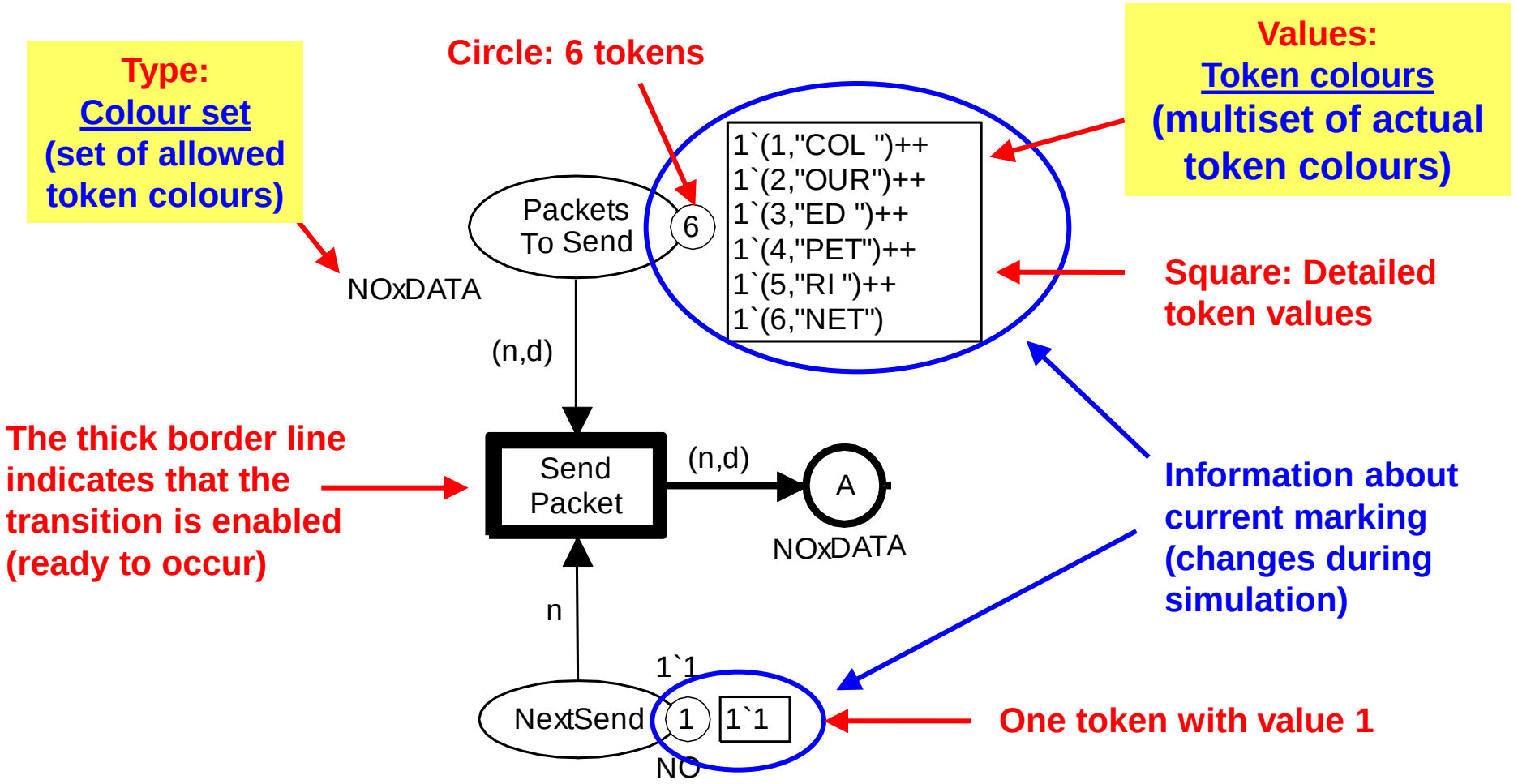


Places represent the state of the system

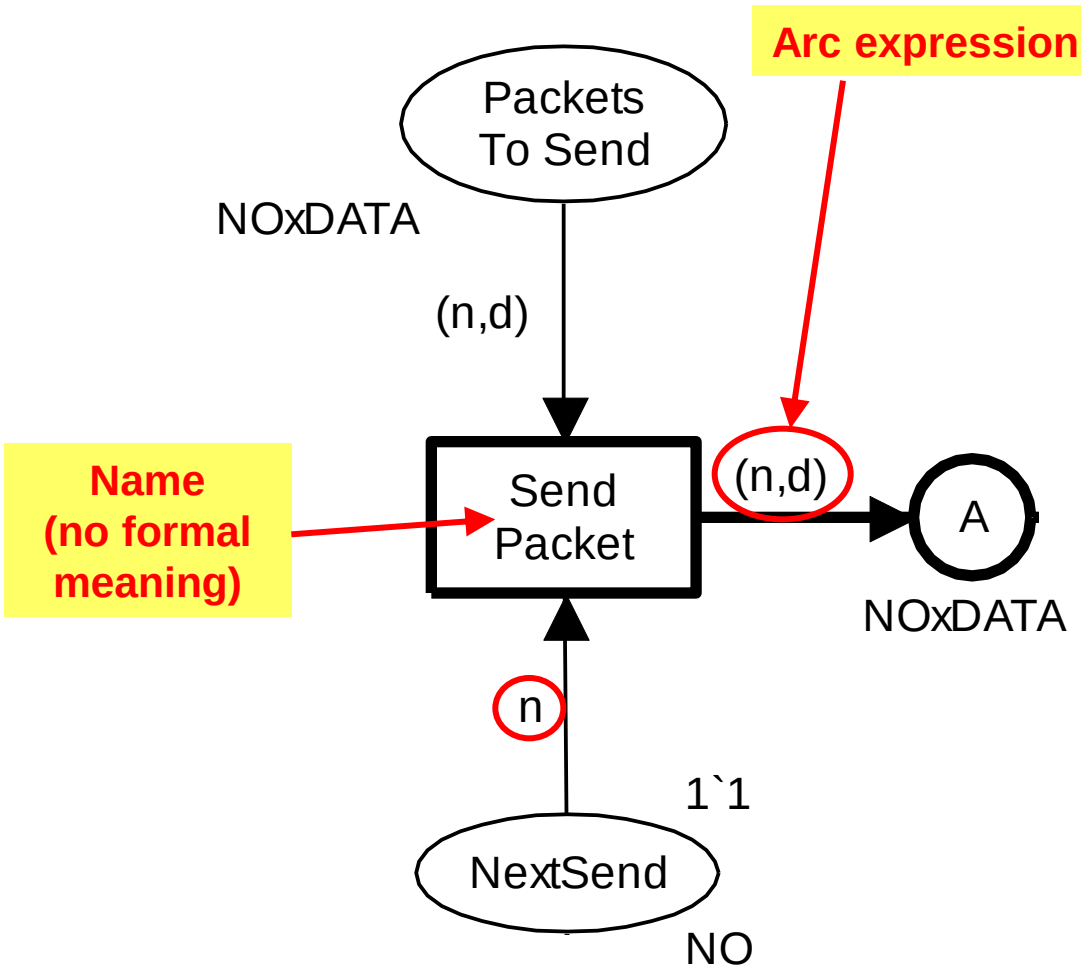


- Each place contains a number of tokens.
- Each token carries a colour (data value).
- The colour set specifies the set of allowed token colours.

Current marking during simulation



Transitions and arcs



The type of the arc expression must be equal to the colour set of the attached place (or a multiset over the colour set)

Declaration of variables:

```
var n : NO;      (* integers *)
var d : DATA;    (* strings *)
```

Binding of variables:

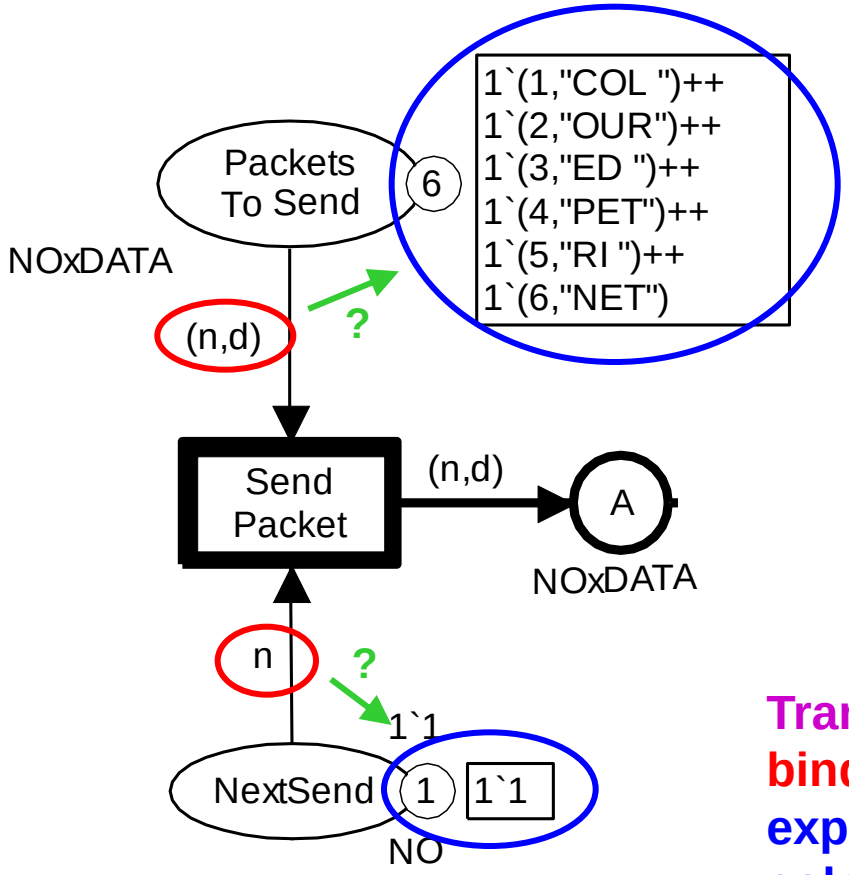
```
<n=3,d="CPN">
```

Evaluation of expressions:

(n,d) → (3,"CPN") : NOxDATA

n → 3 : NO

Enabling of transitions



Two variables:

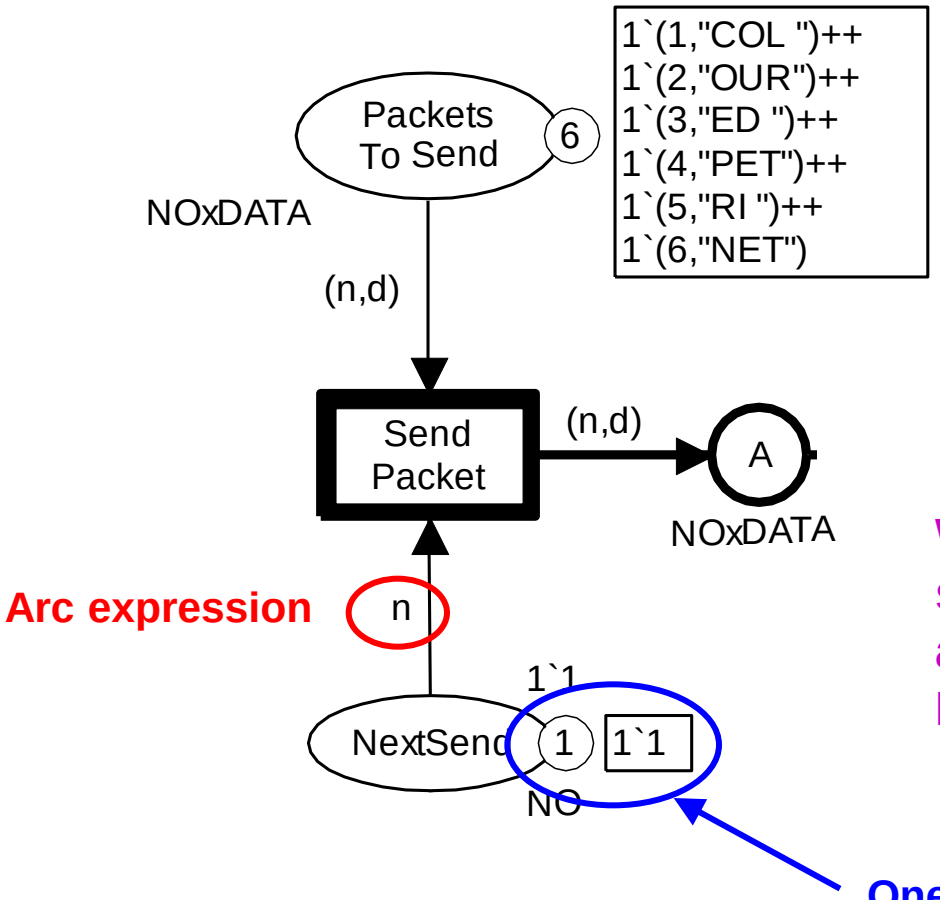
```
var n : NO;      (* integers *)
var d : DATA;    (* strings *)
```

Binding: < n=? , d=? >

NO DATA

Transition is enabled if we can find a binding so that each input arc expressions evaluates to a multi-set of colours that is present on the corresponding input place

Enabling of SendPacket

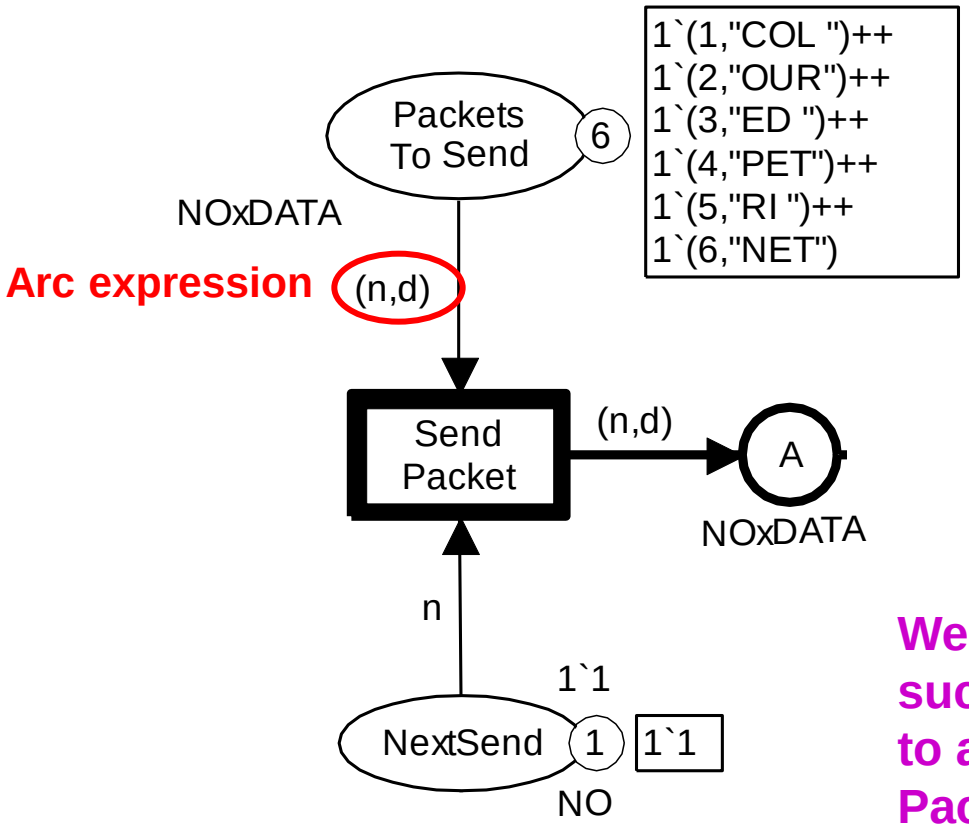


Binding: < n=1 , d=? >

We want to find a binding for the variable n such that the arc expression n evaluates to a colour which is present on the place NextSend

One token with value 1

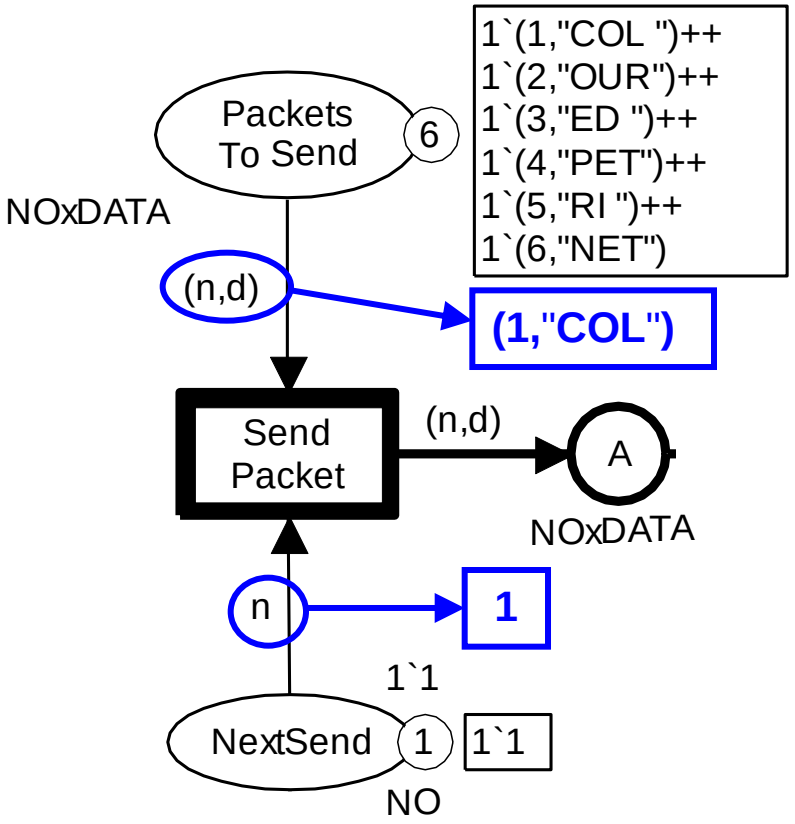
Enabling of SendPacket



Binding: < n=1 , d="COL" >

We want to find a binding for the variable d such that the arc expression (n,d) evaluates to a colour which is present on the place PacketsToSend

Enabling of SendPacket

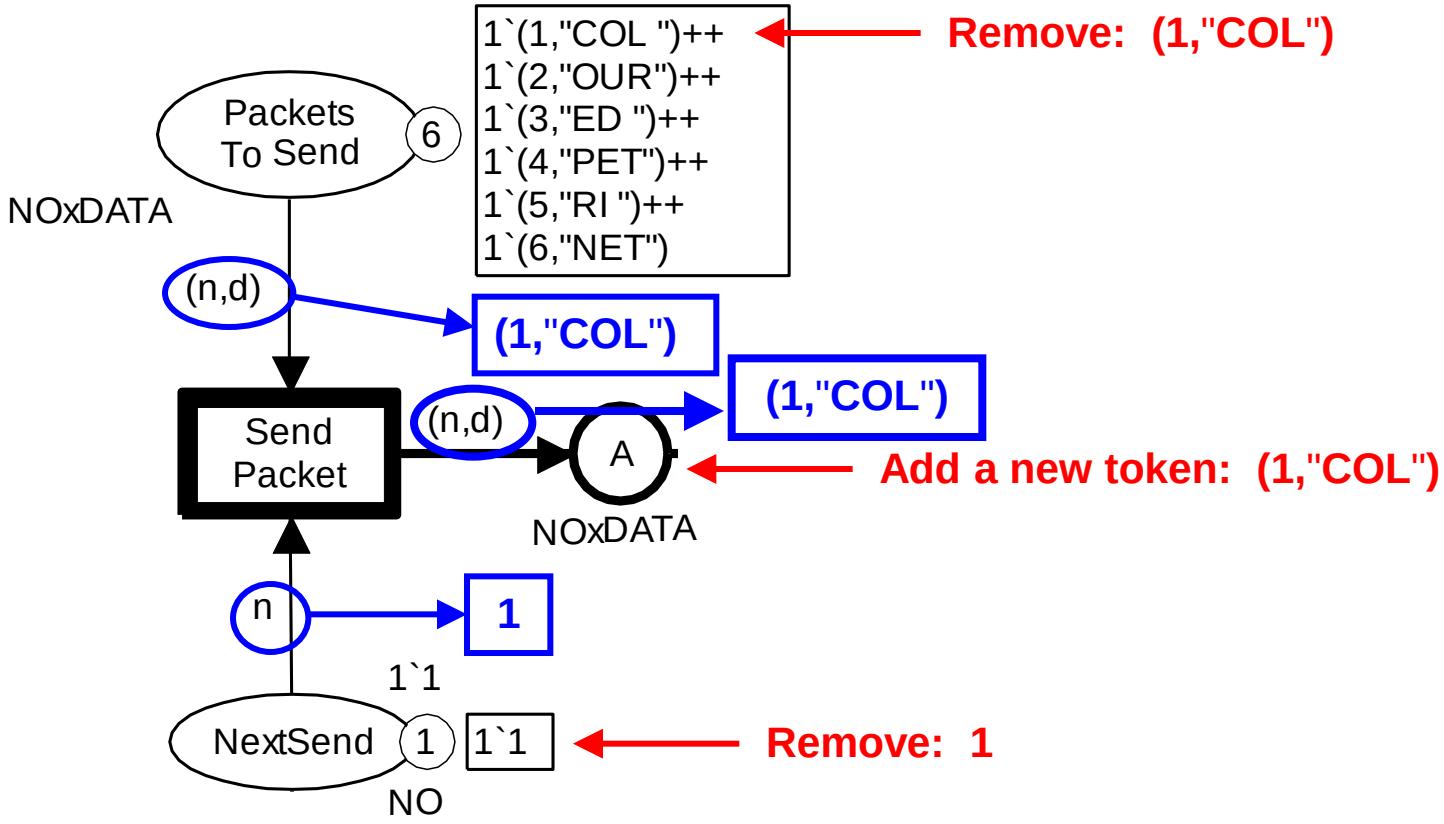


We have found a binding so that each input arc expression evaluates to a colour that is present on the corresponding input place

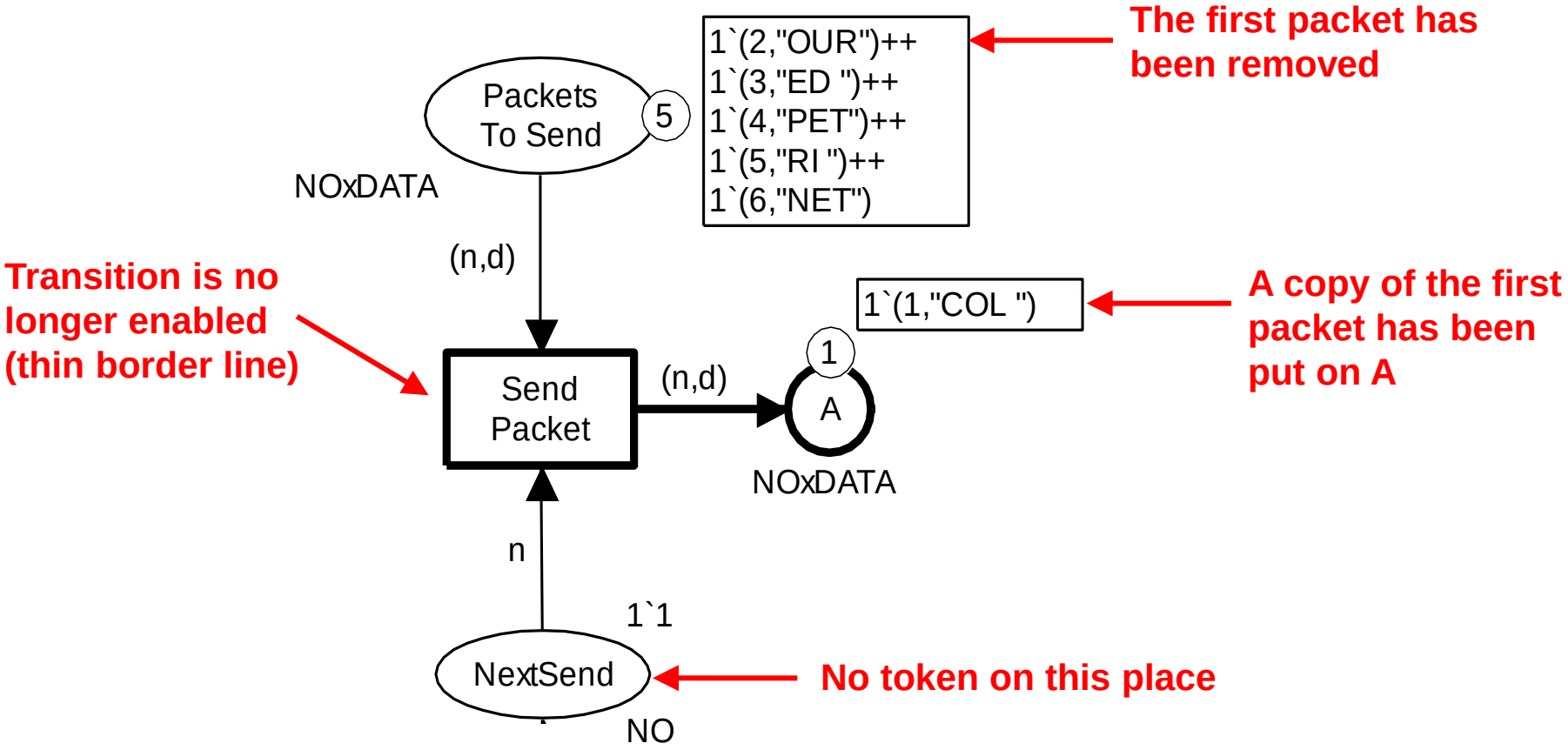
Binding: < n=1 , d="COL" >

Transition is enabled (ready to occur)

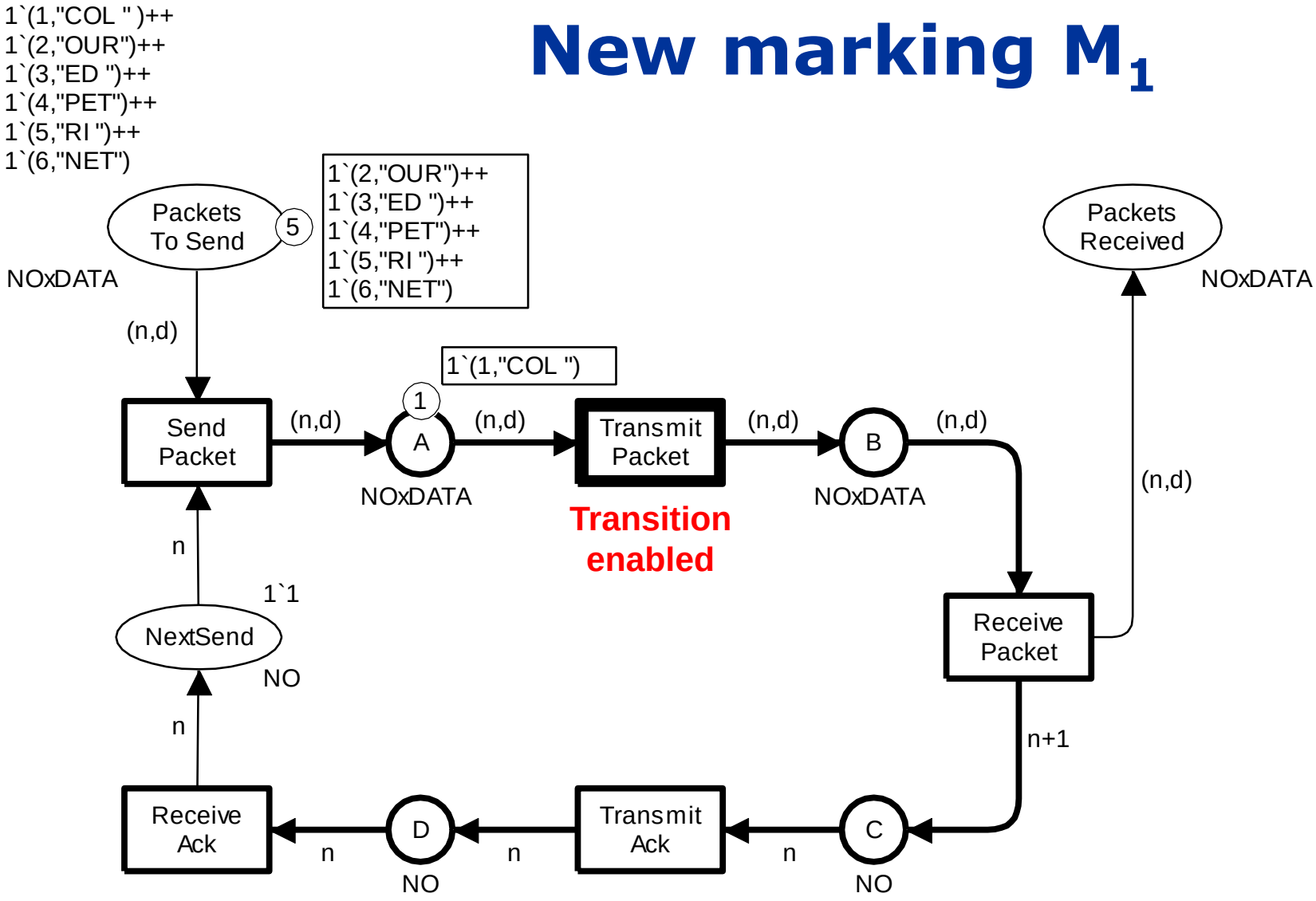
Occurrence of SendPacket in binding <n=1,d="COL">



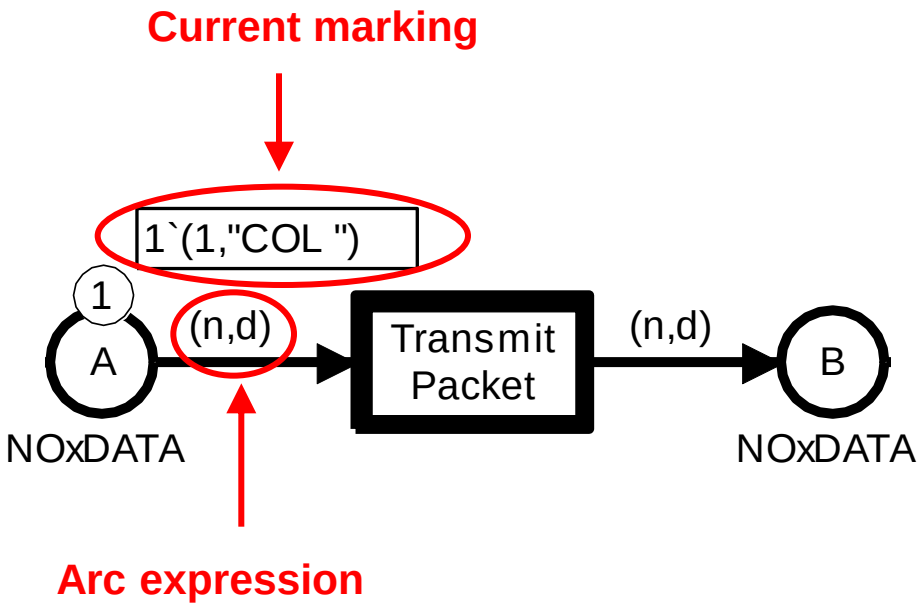
New marking after occurrence of SendPacket in binding <n=1,d="COL">



New marking M_1

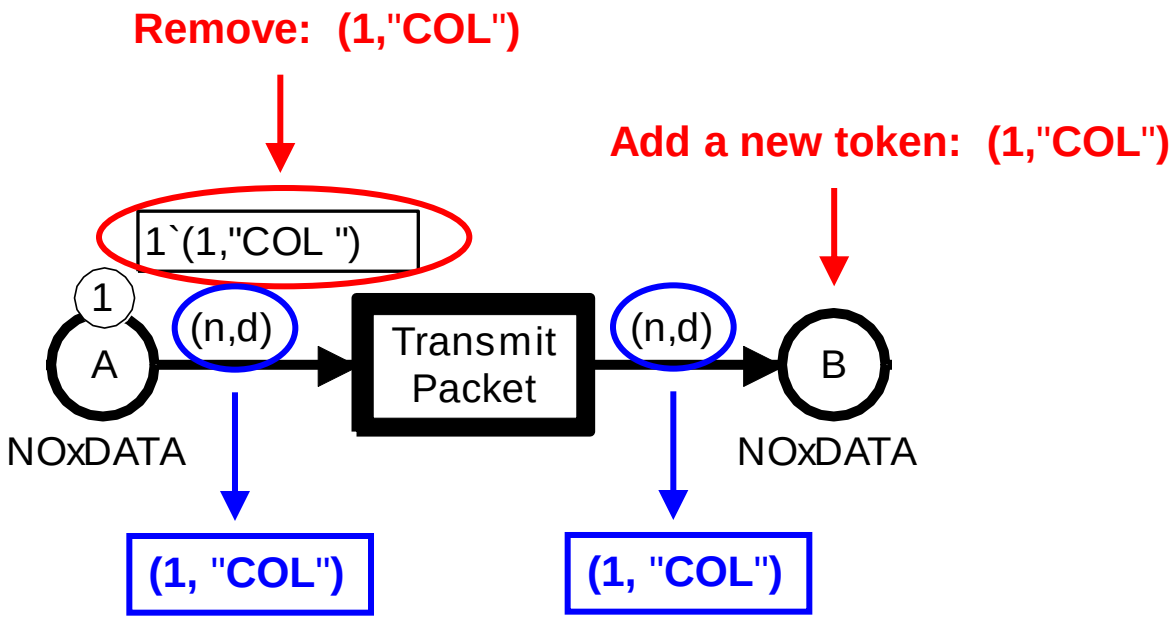


Binding of TransmitPacket

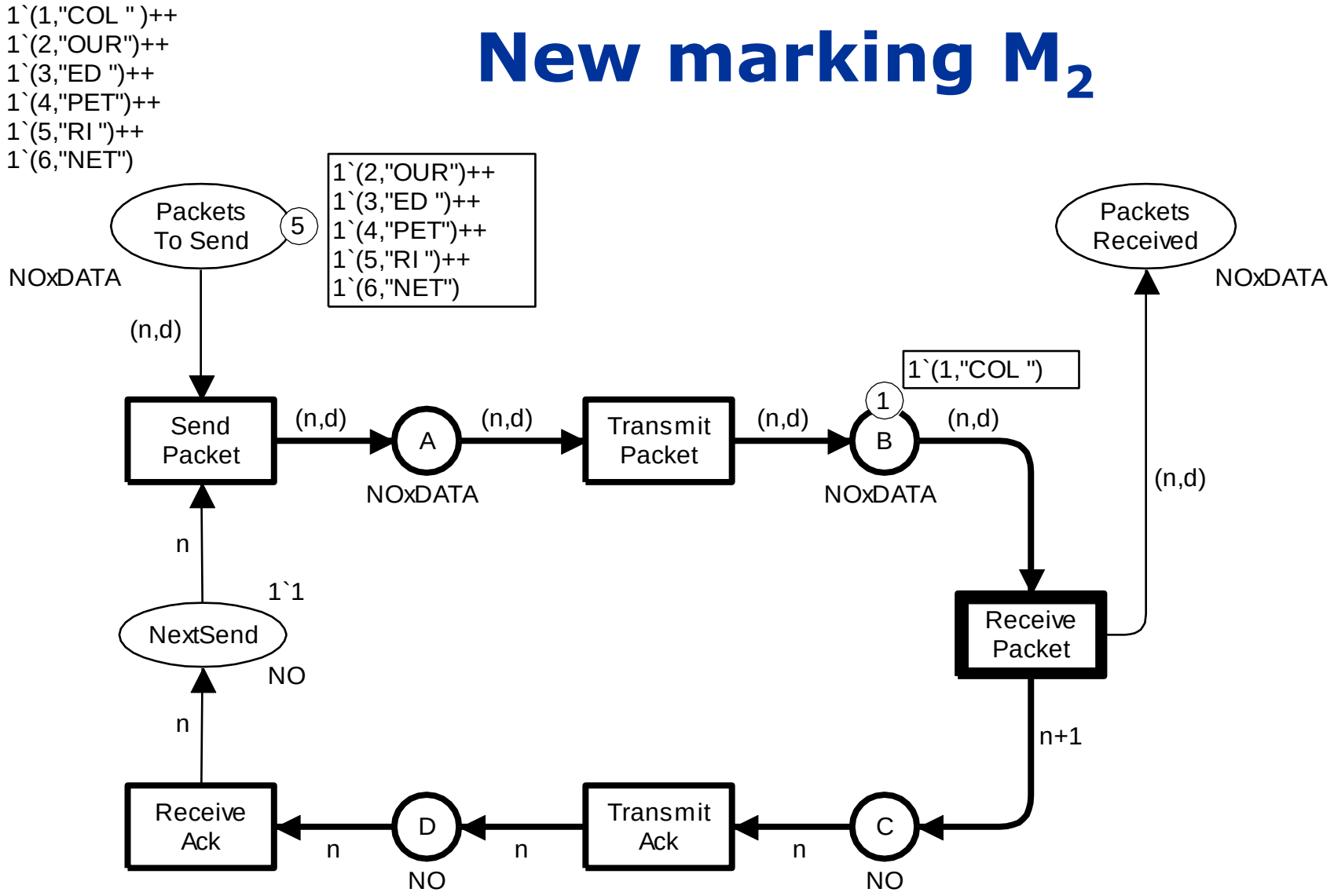


Binding: < n=1 , d="COL" >

Occurrence of TransmitPacket in binding <n=1,d="COL">

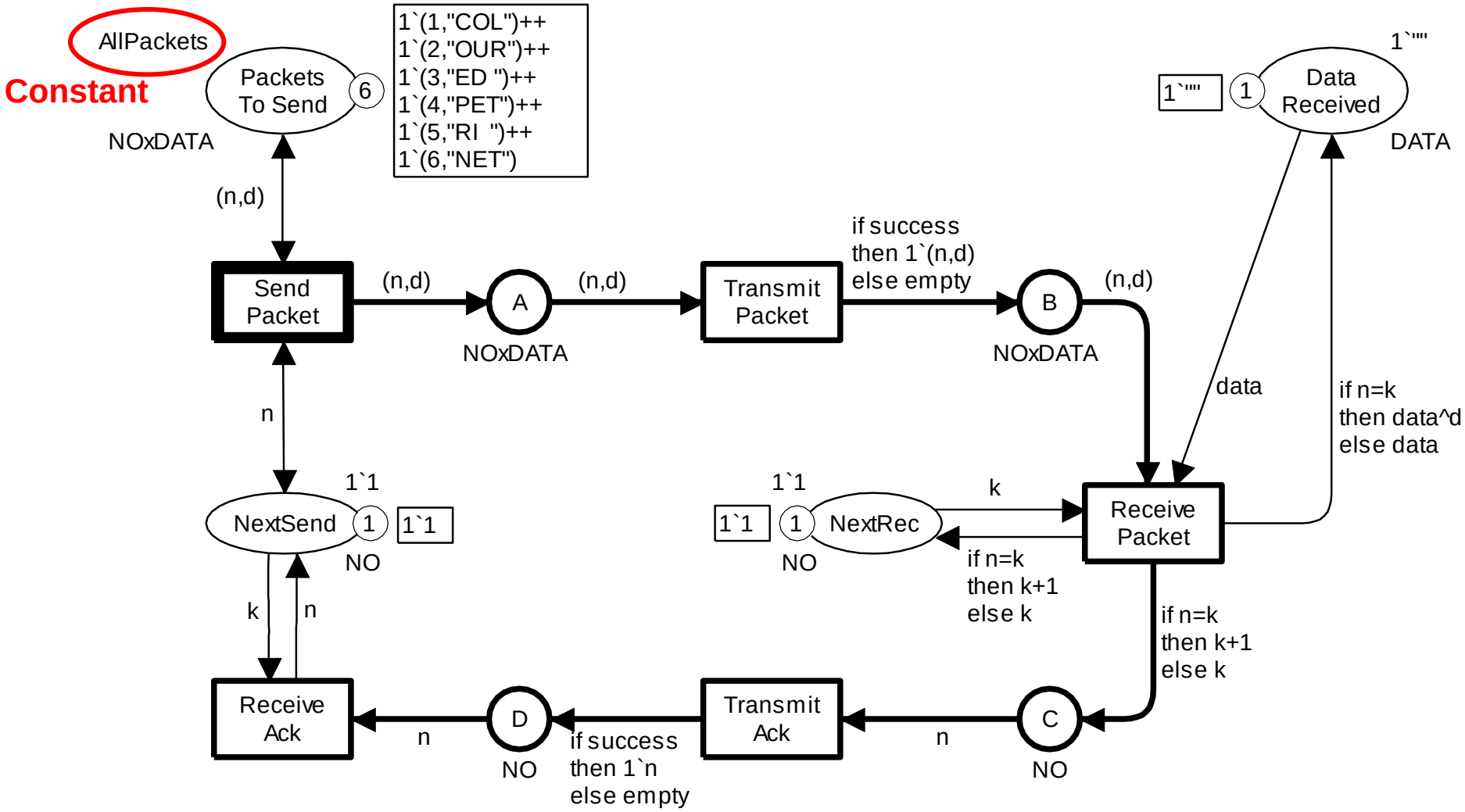


New marking M_2



Simulation Demo in CPN Tools

Second version of protocol



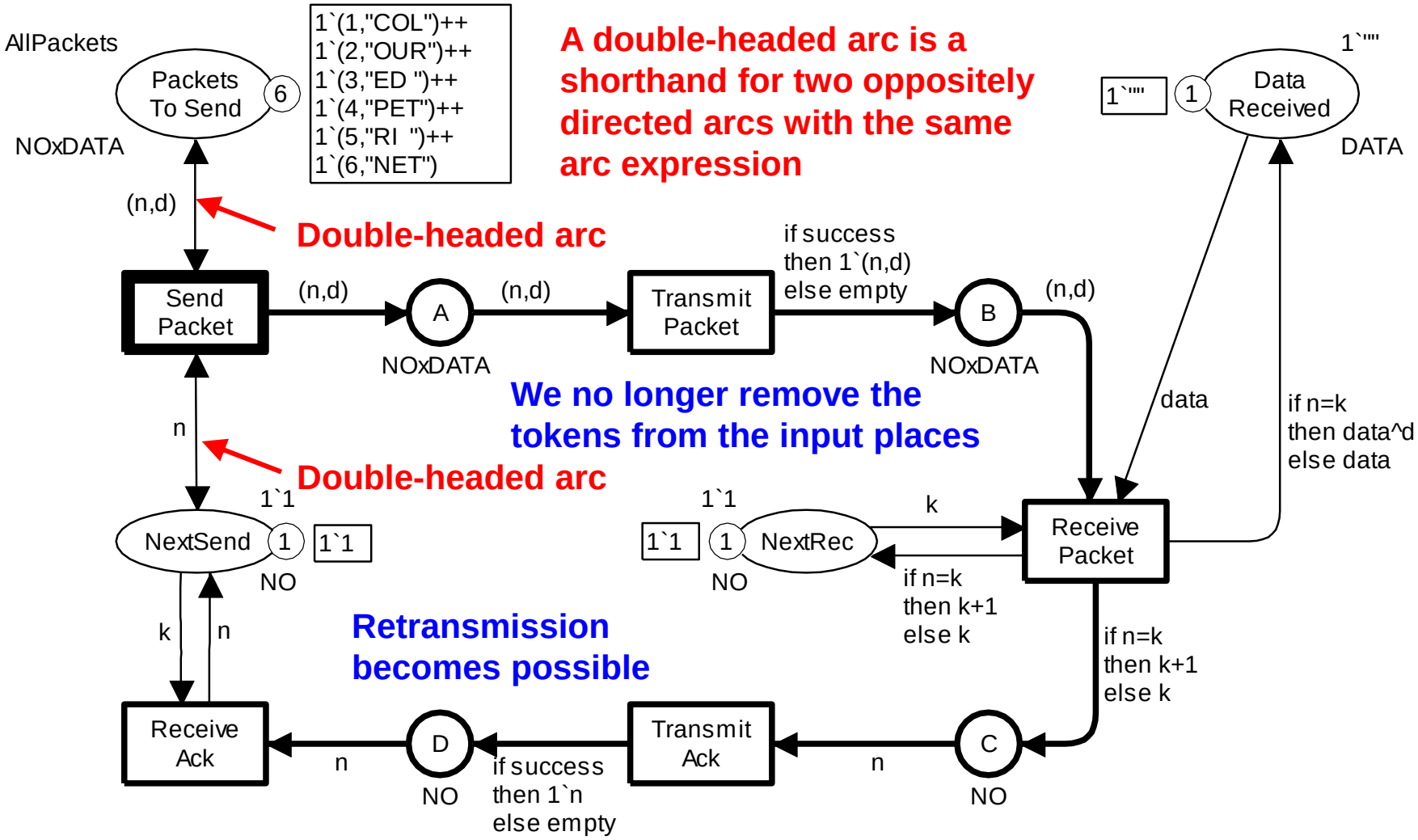
Declaration of constants

- We use the following constant to specify the initial marking of PacketsToSend.

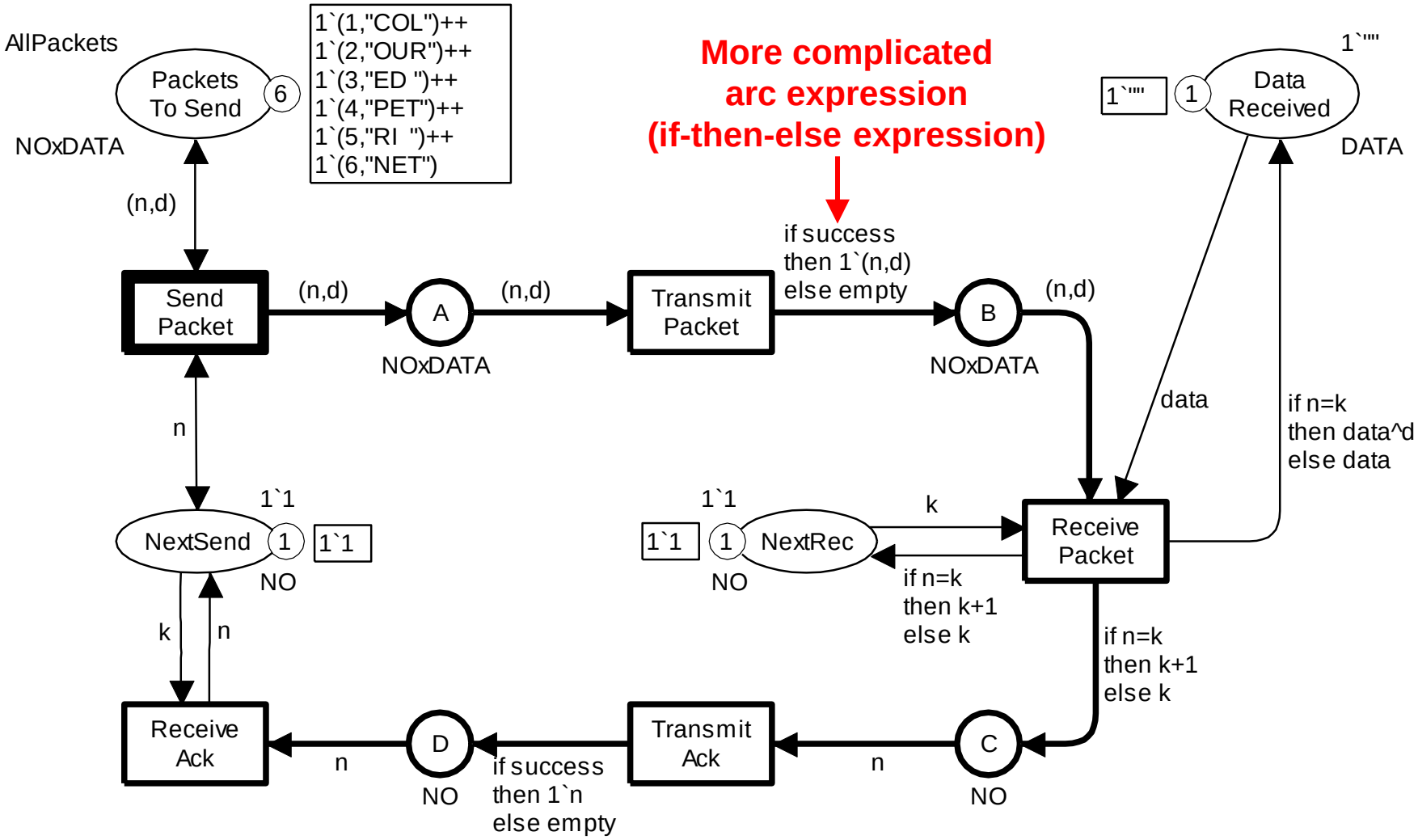
```
val AllPackets = 1`(1,"COL") ++ 1`(2,"OUR") ++
                 1`(3,"ED ") ++ 1`(4,"PET") ++
                 1`(5,"RI ") ++ 1`(6,"NET");
```

- Saves a little bit of space in the diagram.
- Enhances readability.
- Can be reused (at other places).

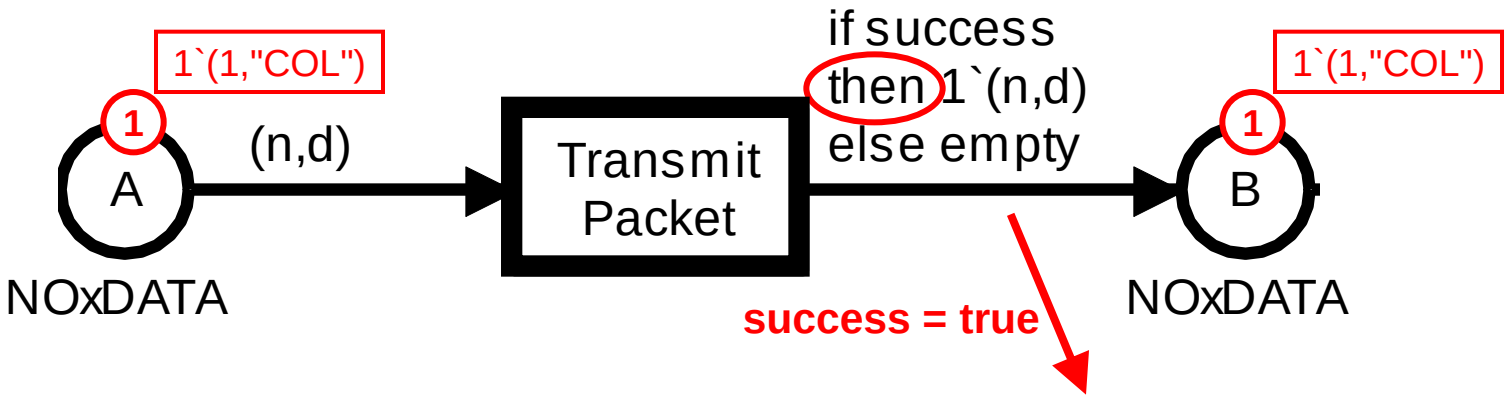
Double-headed arcs



More complicated arc expression



If-then-else expression



New variable:

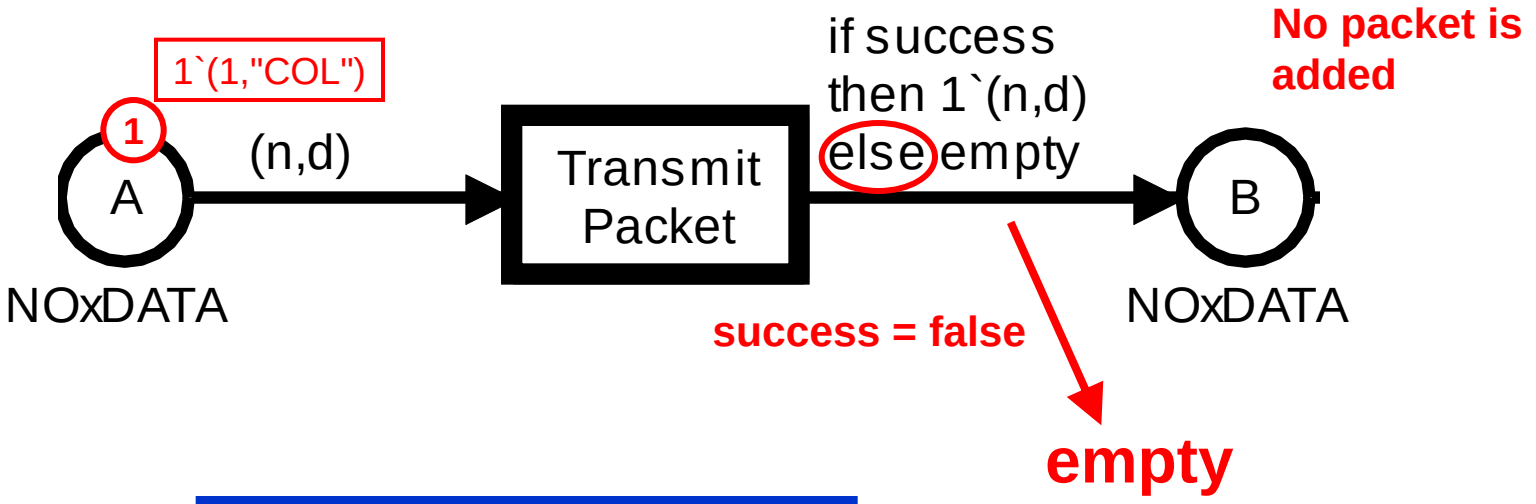
```
var success : BOOL;
```

1`(1,"COL")

Successful transmission over the network

b^+ = <n=1, d="COL", success=true>
b^- = <n=1, d="COL", success=false>

If-then-else expression

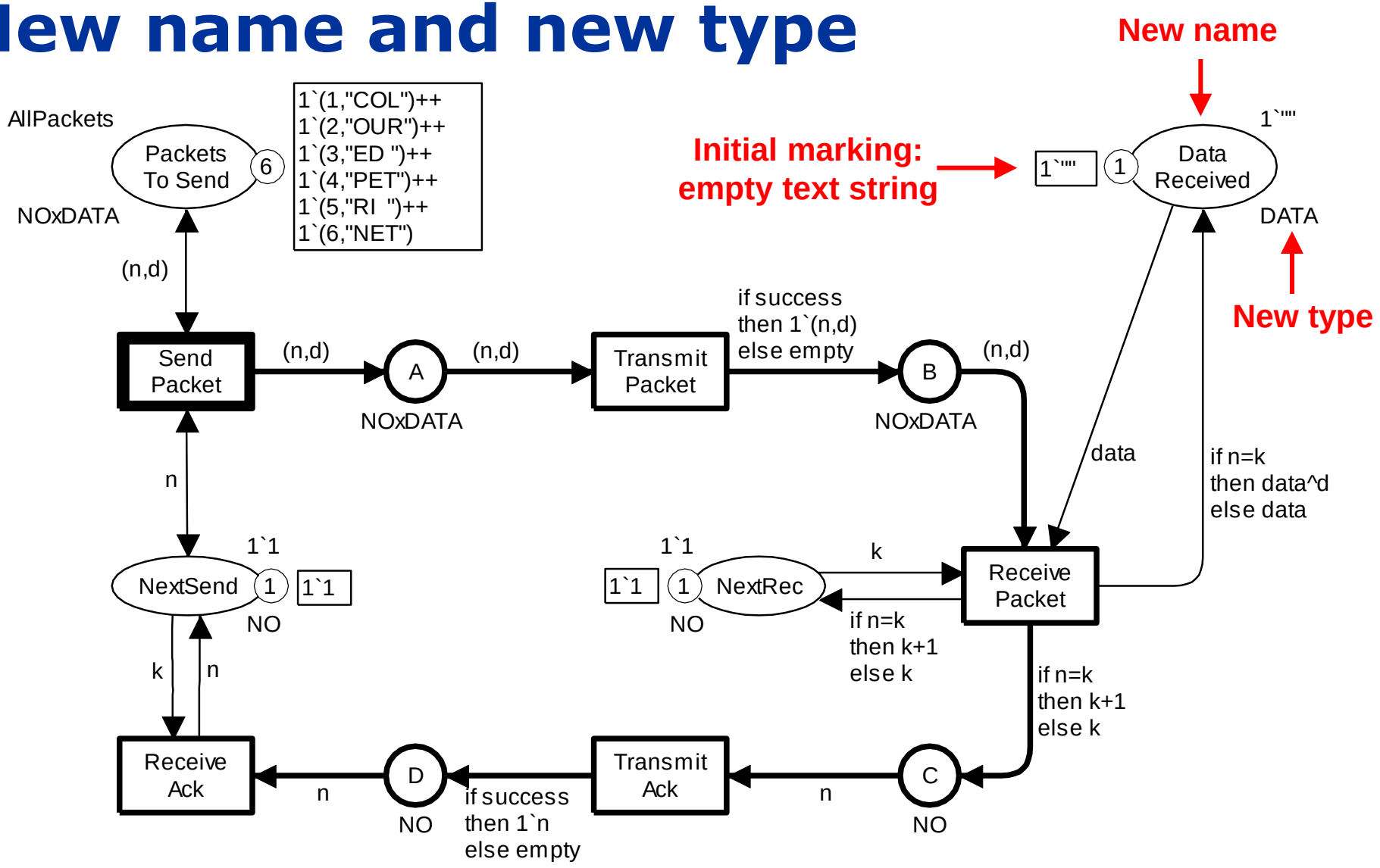



```
var success : BOOL;
```

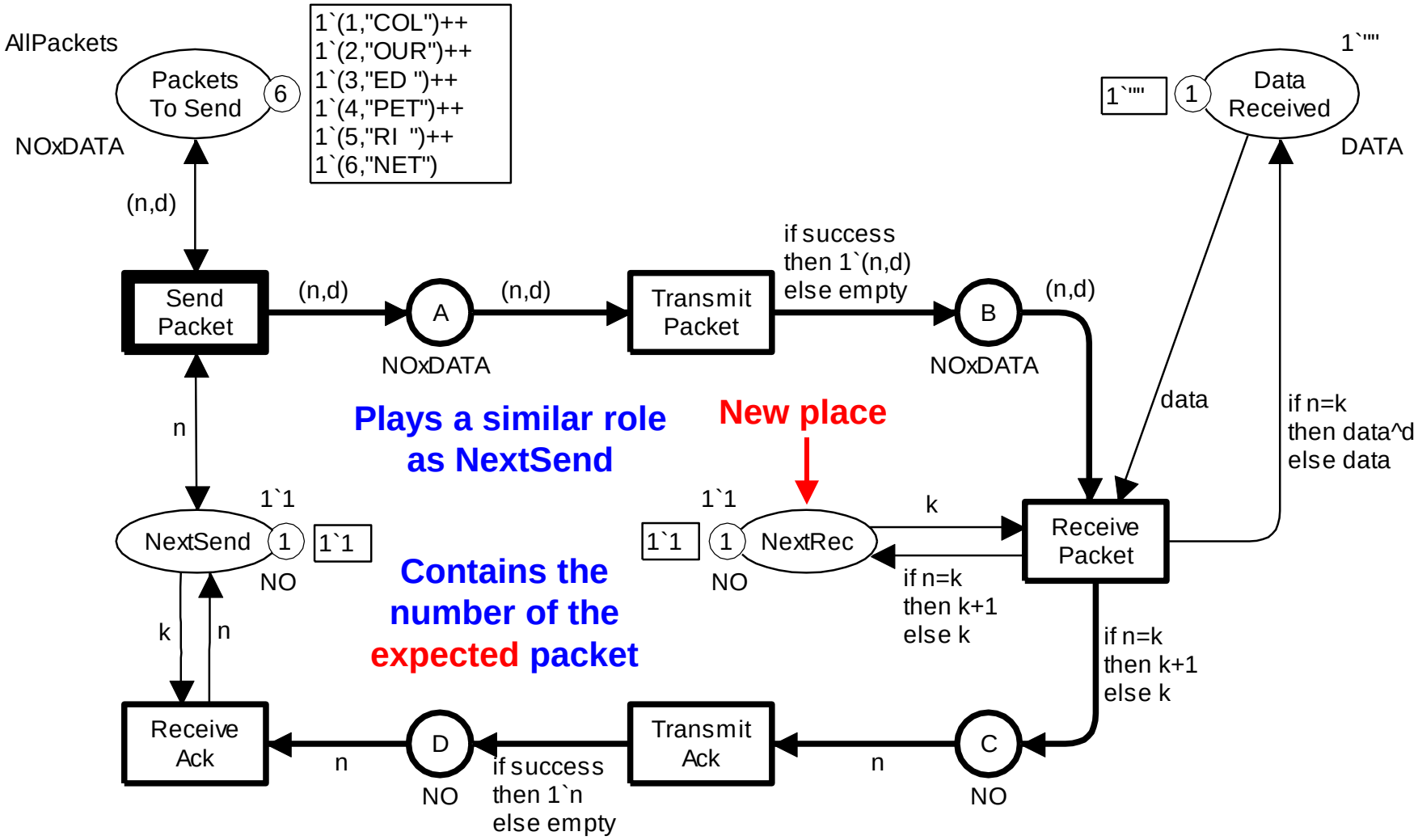
Packet is lost during transmission

b^+ = <n=1, d="COL", success=true>
b^- = <n=1, d="COL", success=false>

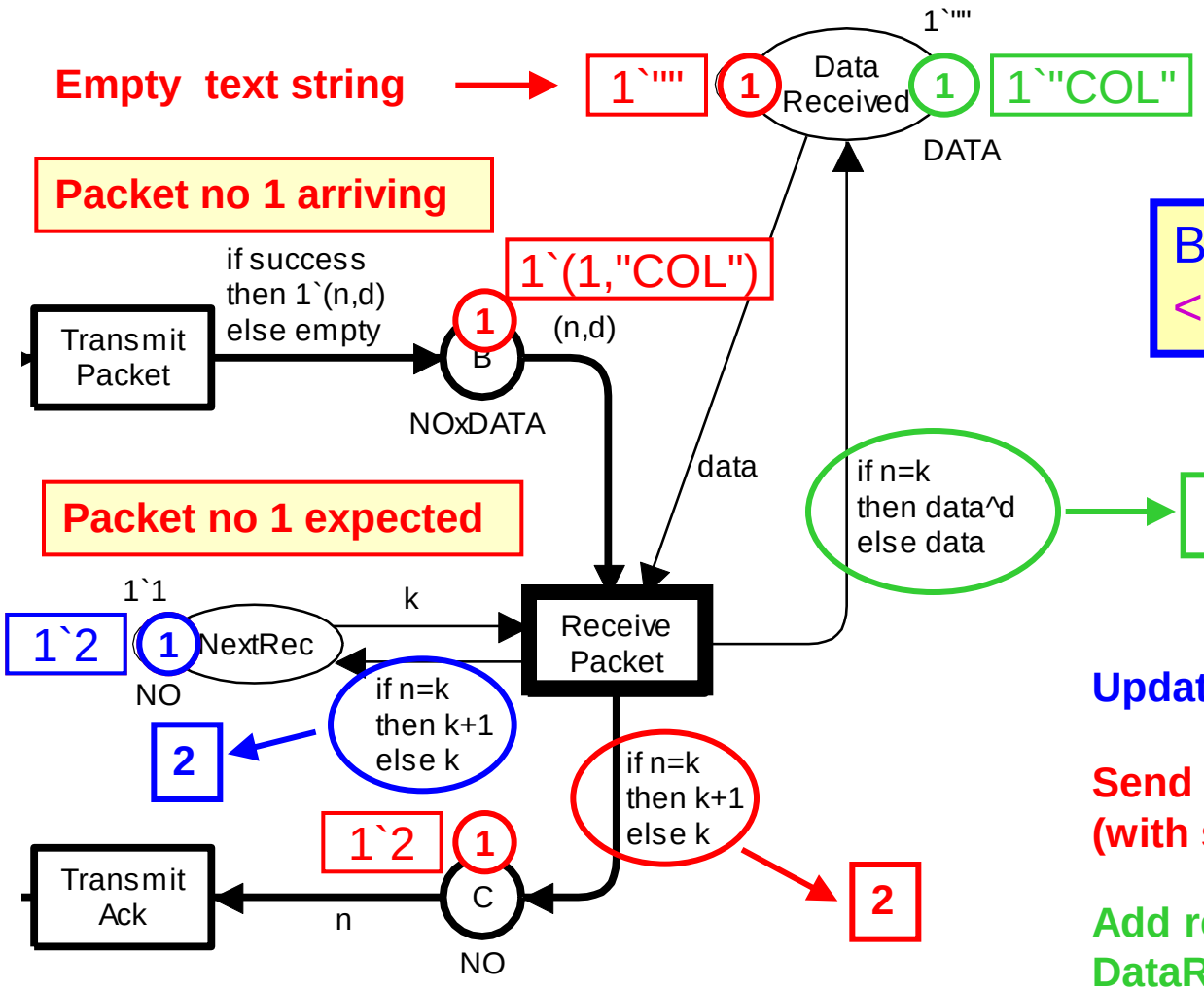
New name and new type



New place: NextRec



Correct packet arrives



Binding:
<n=1, d="COL", k=1, data="">

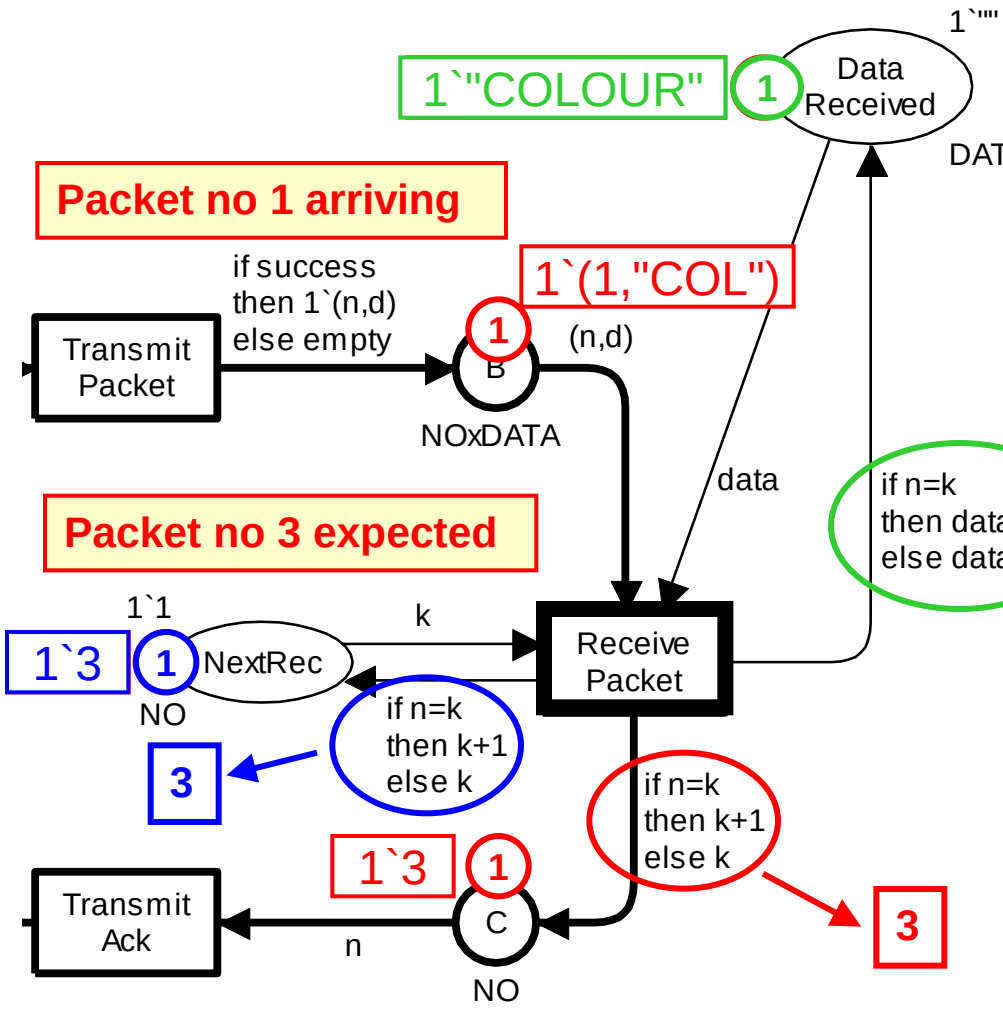
^ is the concatenation operator

Update NextRec (from 1 to 2)

Send acknoweledgement (with sequence number of next packet)

Add received data: "COL" to DataReceived

Wrong packet arrives



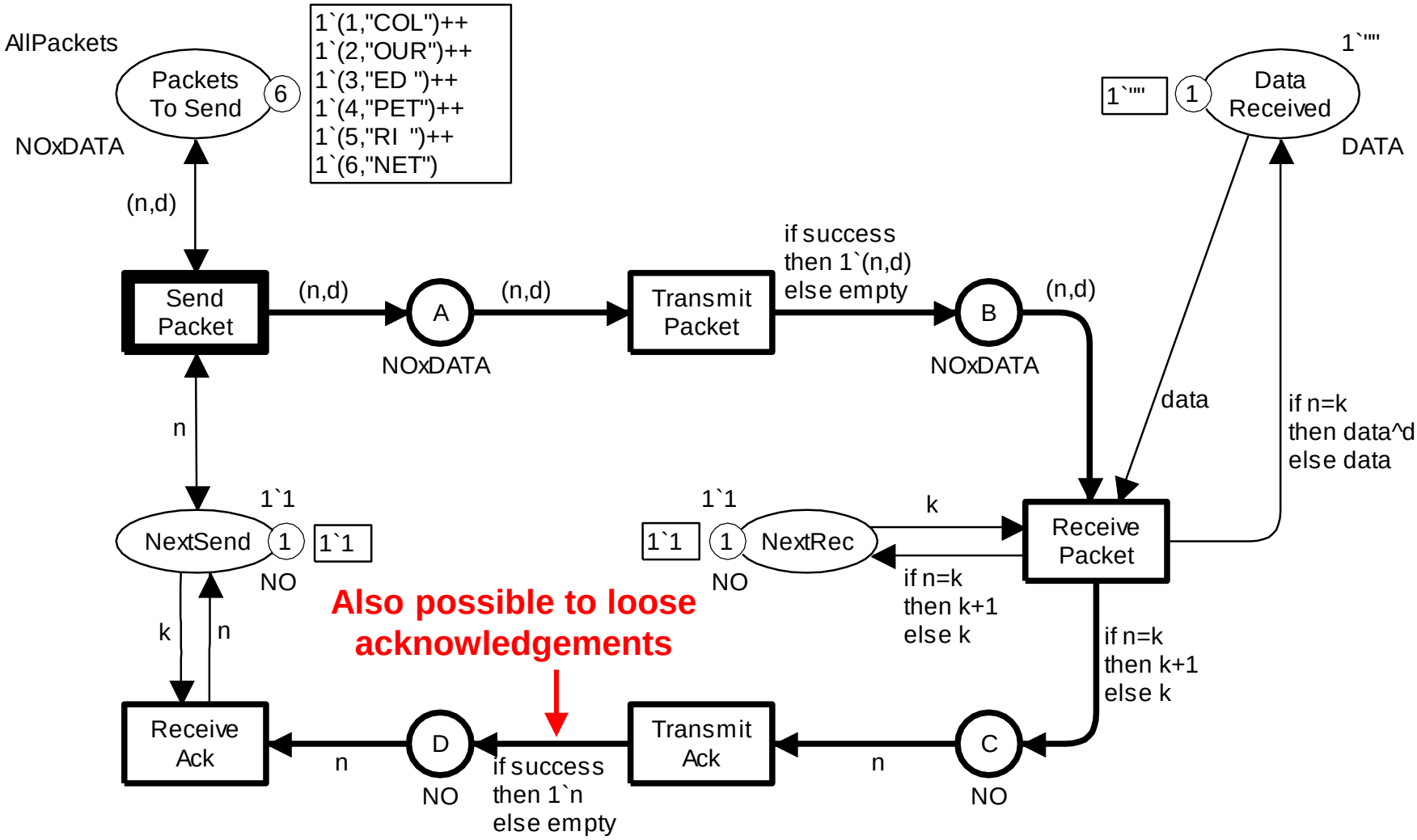
Binding:
<n=1, d="COL", k=3, data="COLOUR">

Do not change NextRec

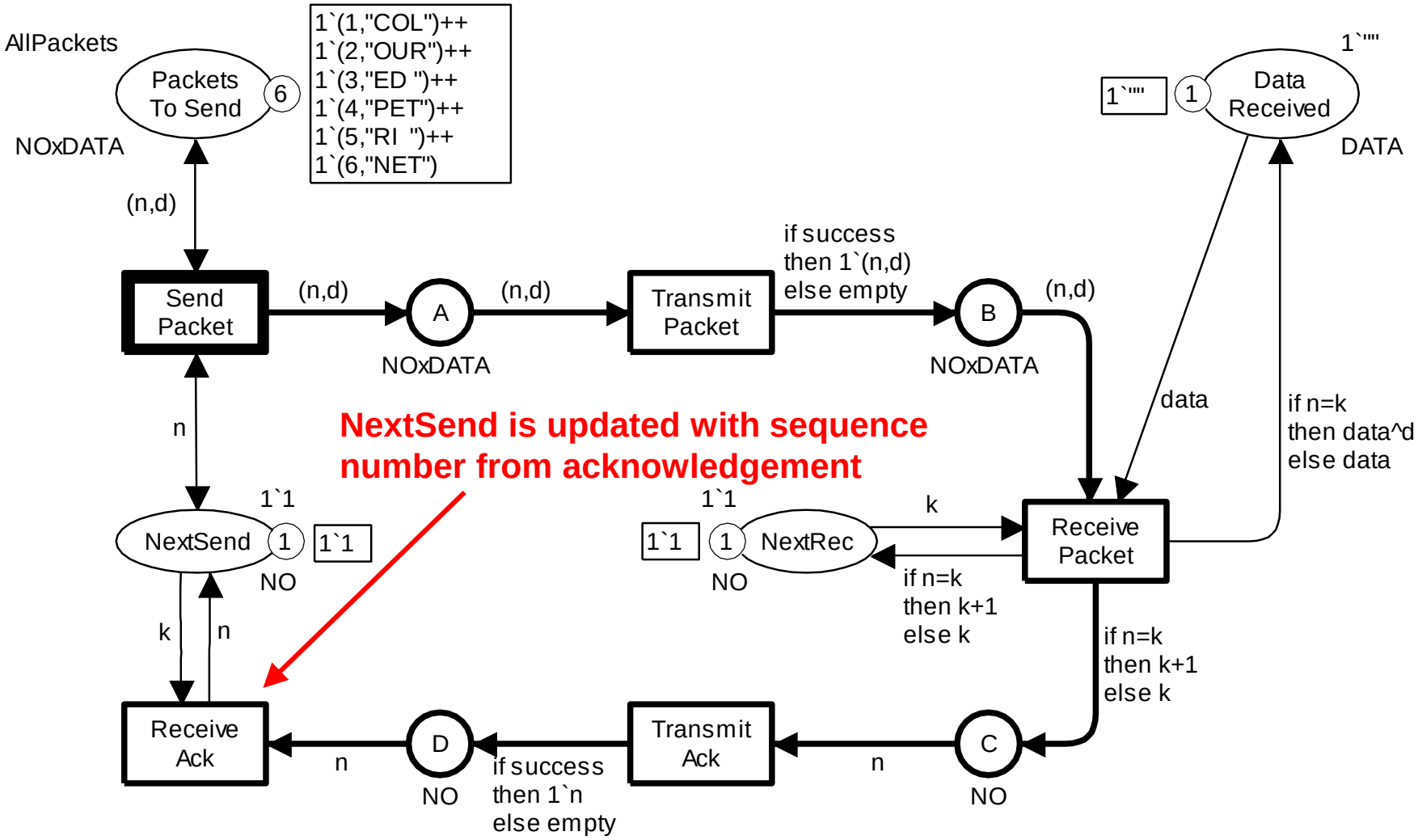
Send acknowledgement (with sequence number of next packet)

No data is added to DataReceived

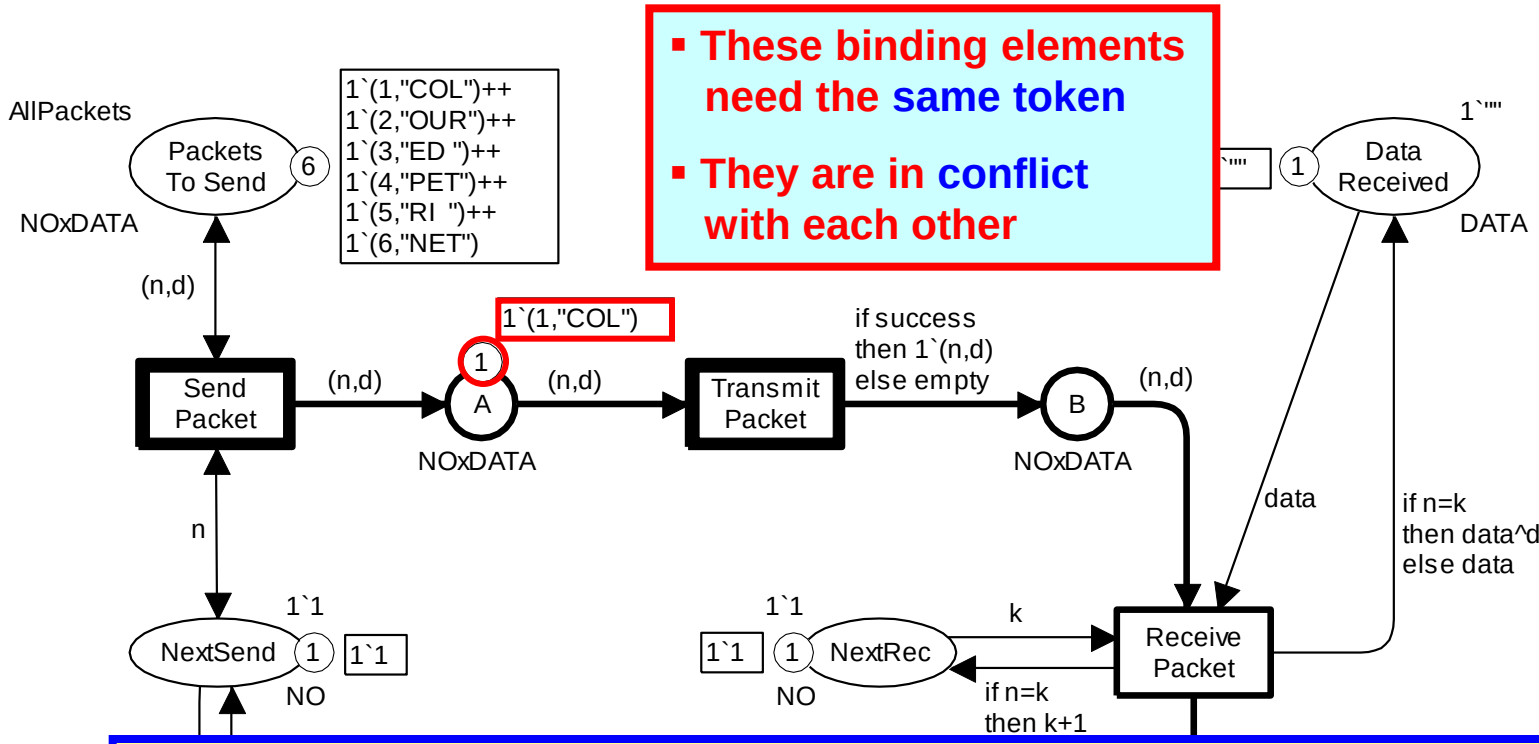
Acknowledgements can be lost



NextSend is updated



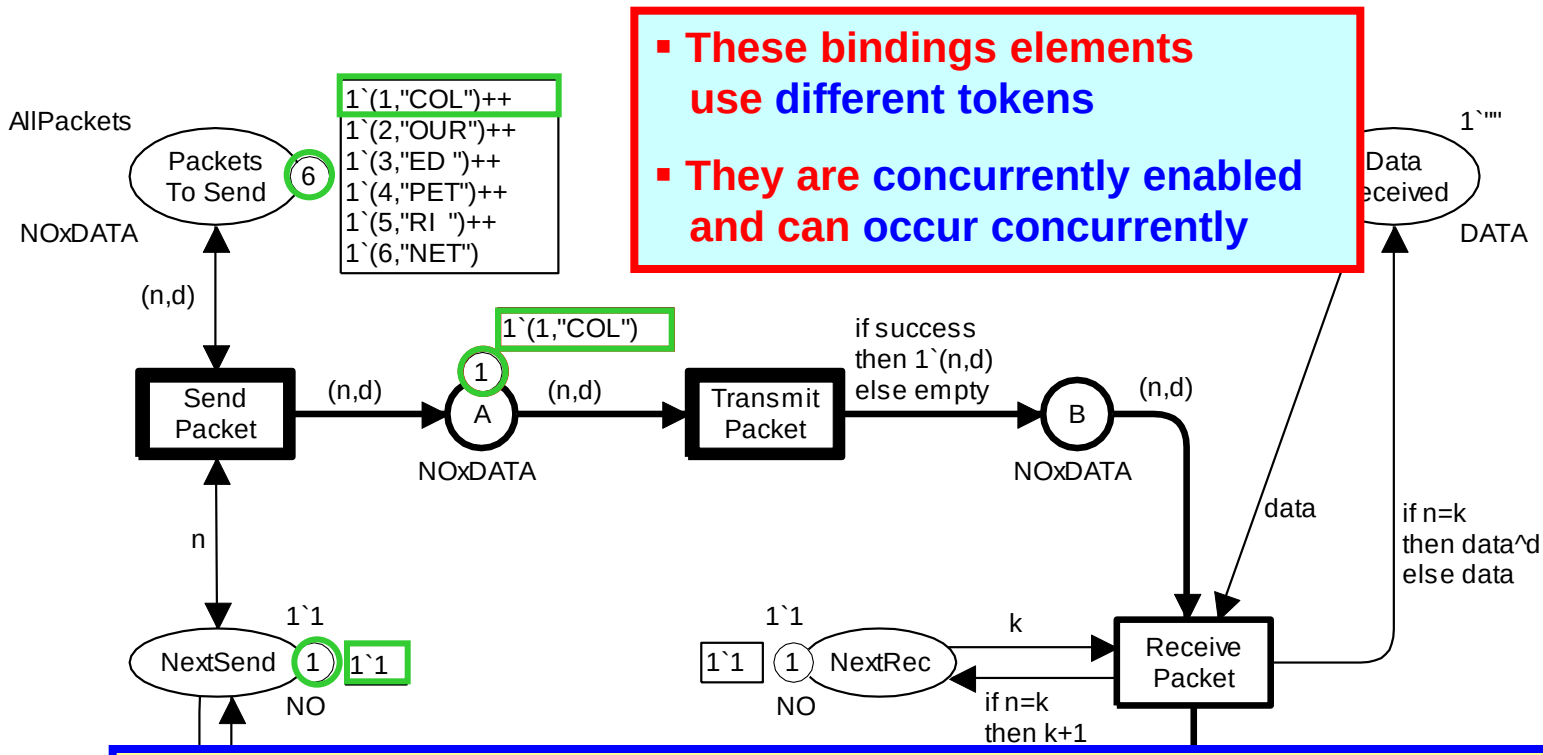
Two enabled transitions



SP = (SendPacket, <n=1, d="COL">)

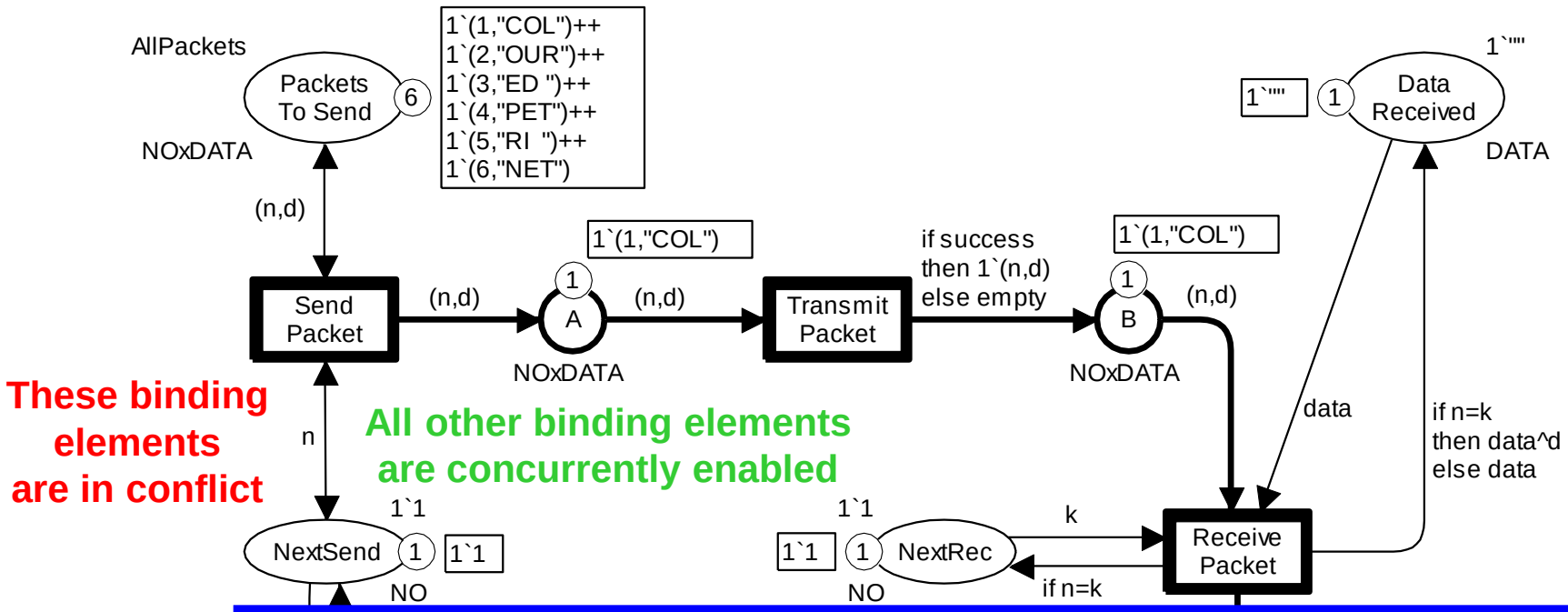
- TP^{+} = (TransmitPacket, <n=1, d="COL", success=true>)
- TP^{-} = (TransmitPacket, <n=1, d="COL", success=false>)

Two enabled transitions



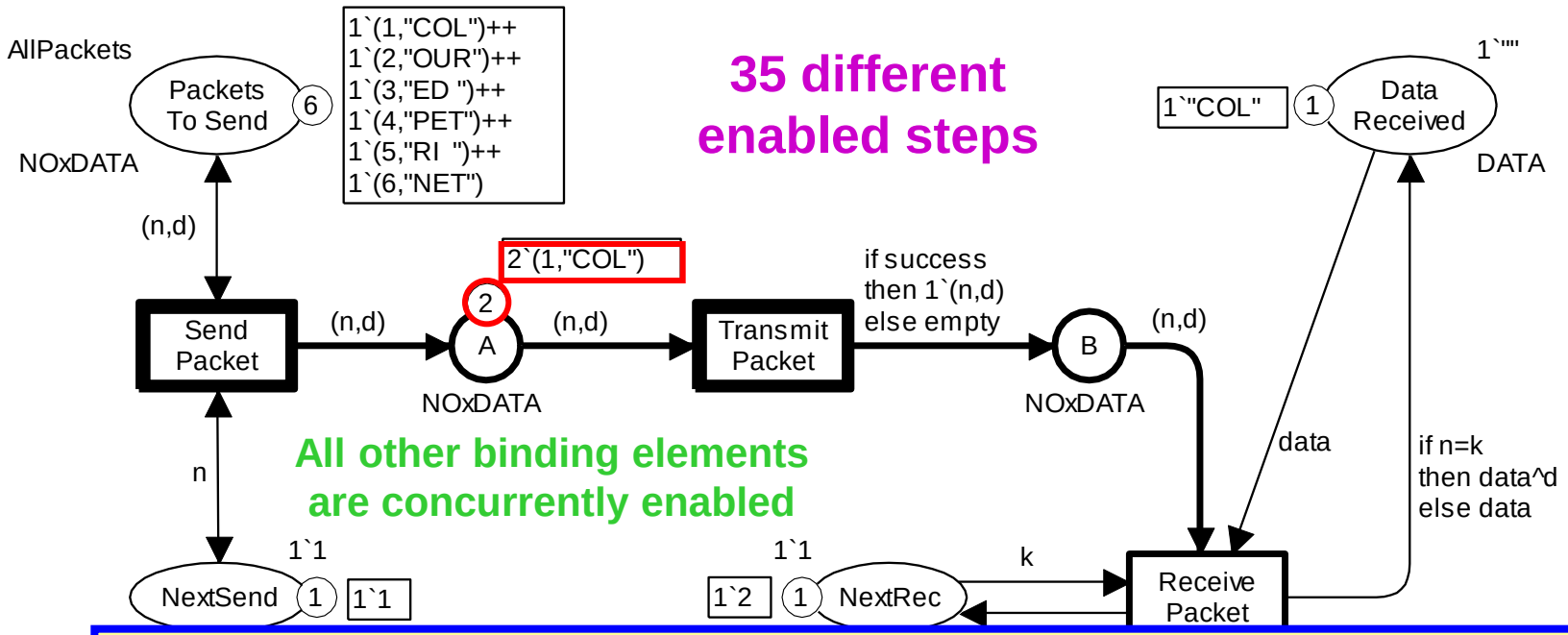
- SP = (SendPacket, <n=1, d="COL">)
- TP^{+} = (TransmitPacket, <n=1, d="COL", success=true>)
- TP^{-} = (TransmitPacket, <n=1, d="COL", success=false>)

Three enabled transitions



- SP = (SendPacket, <n=1, d="COL">)
- TP^+ = (TransmitPacket, <n=1, d="COL", success=true>)
- TP^- = (TransmitPacket, <n=1, d="COL", success=false>)
- RP = (ReceivePacket, <n=1, d="COL", k=1, data="">)

Three enabled transitions

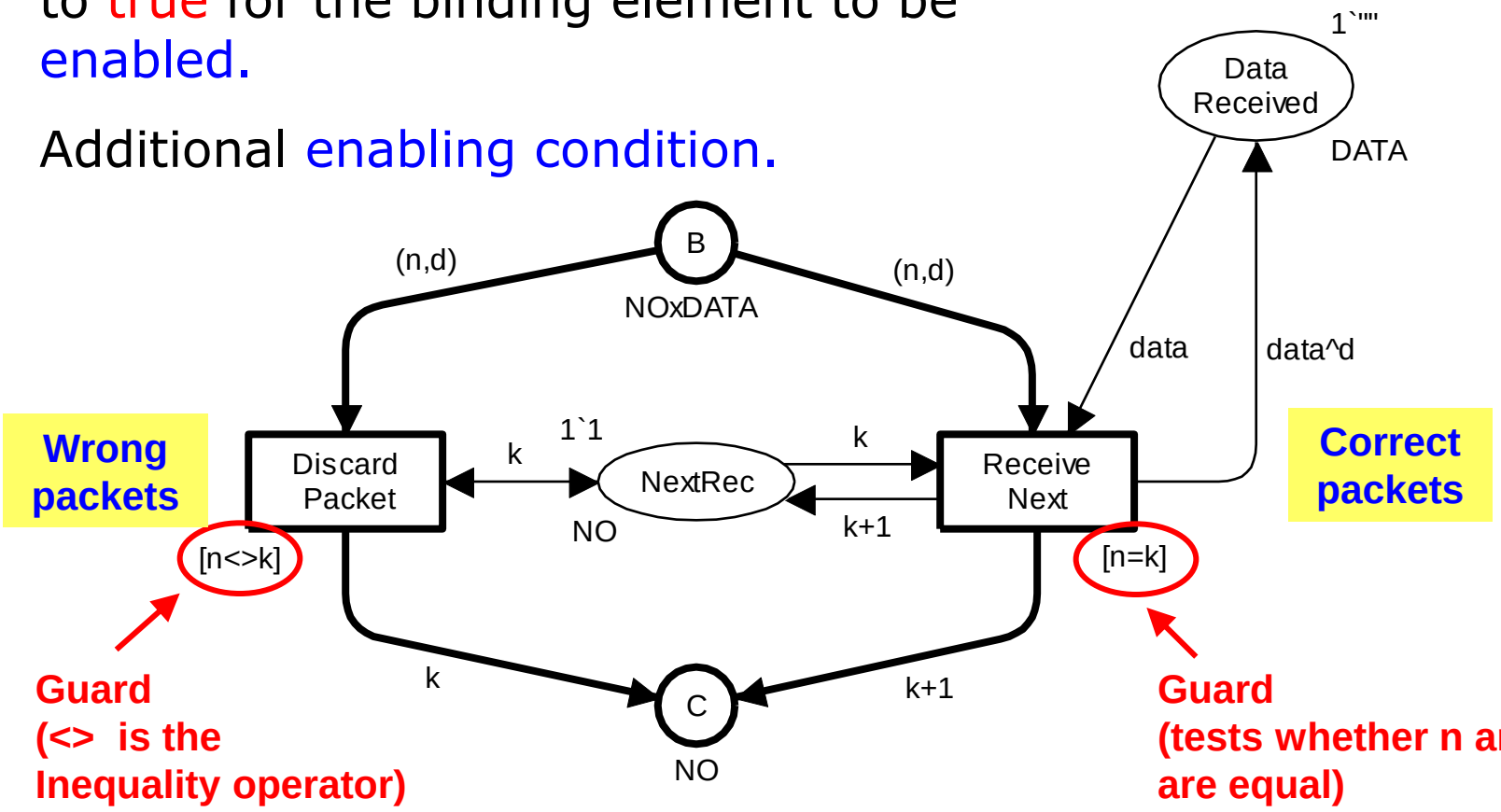


- SP = (SendPacket, <n=1, d="COL">)
- TP^+ = (TransmitPacket, <n=1, d="COL", success=true>)
- TP^- = (TransmitPacket, <n=1, d="COL", success=false>)
- TA^+ = (TransmitAck, <n=2, success=true>)
- TA^- = (TransmitAck, <n=2, success=false>)

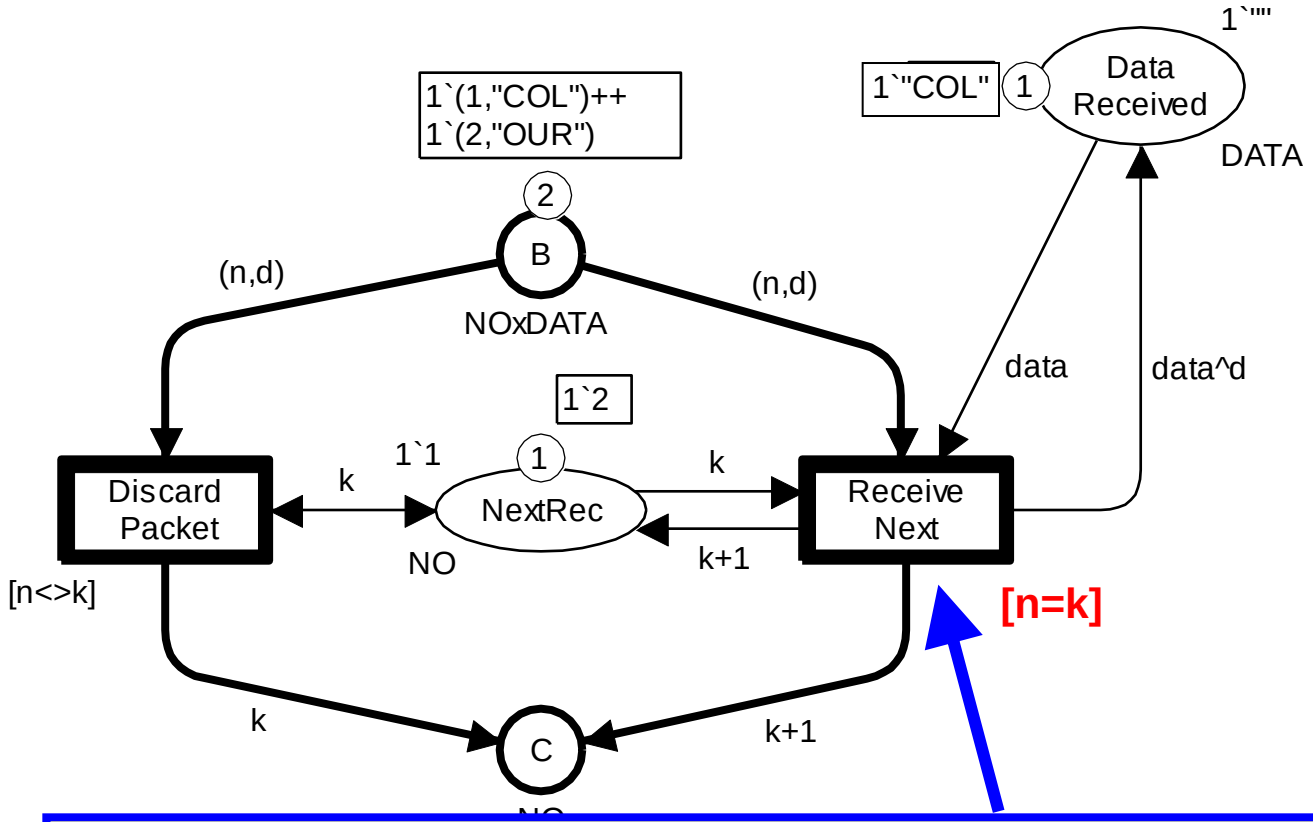
Simulation Demo in CPN Tools

Transitions can have a guard

- Boolean expression which must evaluate to true for the binding element to be enabled.
- Additional enabling condition.



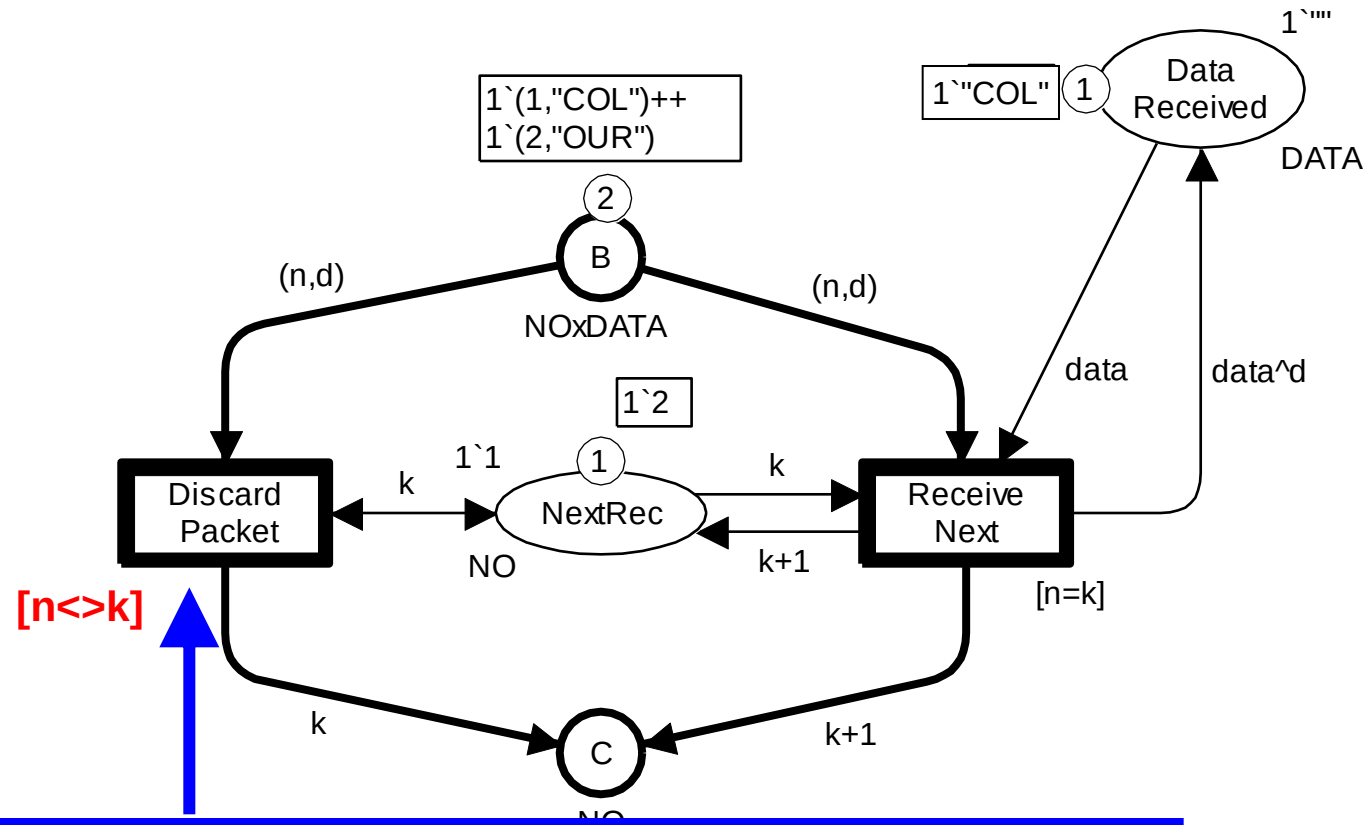
Guard must evaluate to true



false RN_1 = (ReceiveNext, <n=1, k=2, d="COL", data="COL">)

true RN_2 = (ReceiveNext, <n=2, k=2, d="OUR", data="COL">)

Guard must evaluate to true



Editing Demo in CPN Tools

Questions

Assignment 3 – Task 1

- In the lectures you have seen a stop-and-wait protocol. The sender keeps sending the same packet until a matching acknowledgement is received. In a sliding window protocol it is possible for the sender to transmit several packets to the receiver before receiving an acknowledgement. The sender has a window containing a number of data packets which are currently under transmission and for which acknowledgements have not yet been received.

Task 1: Modify the CPN model such that it models a Go-Back-N Window Protocol.

- In a Go-Back-N protocol, the sender sends all data packets in the current window and then waits for acknowledgements. If no acknowledgement is received (within a certain amount of time), the data packets in the window are all retransmitted.
- The CPN model of the stop-and-wait protocol can be downloaded from: http://www.cs.au.dk/~cpnbook/models/chapter2/2-10NondeterministicProtocol.cpn
- It should only be necessary to modify the Sender part of the CPN model. You should change the colour set and the initial marking of the NextSend place, so that it contains information about the start and end of the window. Having done this you should change the arc inscriptions of the surrounding arcs so that they implement a sliding windows strategy.
- Use simulation to validate the correctness of your protocol. It may be useful to change the initial marking of PacketToSend so that you get more packets to work with.