Redes de Petri Coloridas (Coloured Petri Nets - CPN)

Prof. Eduardo Tavares

Prof. Paulo Maciel

Centro de Informática (UFPE)

Disciplina: Modelos para Sistemas Comunicantes

Abstração

Foco naquilo que é essencial

Detalhes não proeminentes no contexto: descartados

Níveis de Abstração

Linguagens de Programação

```
00000000                push    ebp
00000001                mov     ebp, esp
00000003                movzx   ecx, [ebp+arg_0]
00000007                pop     ebp
00000008                movzx   dx, cl
0000000C                lea     eax, [edx+edx]
0000000F                add     eax, edx
00000011                shl     eax, 2
00000014                add     eax, edx
00000016                shr     eax, 8
00000019                sub     cl, al
0000001B                shr     cl, 1
0000001D                add     al, cl
0000001F                shr     al, 5
00000022                movzx   eax, al
00000025                retn
```

```
public void onModuleLoad() {
  final Button button = new Button("Click me");
  final Label label = new Label();

  button.addClickListener(new ClickListener() {
    public void onClick(Widget sender) {
      if (label.getText().equals(""))
        label.setText("Hello World!");
      else
        label.setText("");
    }
  });

  RootPanel.get("div1").add(button);
  RootPanel.get("div2").add(label);
 }
}
```

Qual é a mais poderosa?

Linguagens de Programação

```
00000000                push    ebp
00000001                mov     ebp, esp
00000003                movzx   ecx, [ebp+arg_0]
00000007                pop     ebp
00000008                movzx   dx, cl
0000000C                lea     eax, [edx+edx]
0000000F                add     eax, edx
00000011                shl     eax, 2
00000014                add     eax, edx
00000016                shr     eax, 8
00000019                sub     cl, al
0000001B                shr     cl, 1
0000001D                add     al, cl
0000001F                shr     al, 5
00000022                movzx   eax, al
00000025                retn
```

```
public void onModuleLoad() {
  final Button button = new Button("Click me");
  final Label label = new Label();

  button.addClickListener(new ClickListener() {
    public void onClick(Widget sender) {
      if (label.getText().equals(""))
        label.setText("Hello World!");
      else
        label.setText("");
    }
  });

  RootPanel.get("div1").add(button);
  RootPanel.get("div2").add(label);
 }
}
```

Qual é a mais poderosa?

R: Resolvem a mesma classe de problemas

Redes Petri

Apropriado para modelagem de sistemas com :

- Concorrência
- Comunicação
- Compartilhamento de recursos
- ...

Place/Transition nets (PT-nets)

- Componentes: Lugares, Transições, Arcos, Tokens (marcação)
- Ausência de tipos de dados e mecanismos de modularidade

Redes de Petri Coloridas (CPN)

Redes de Petri + Linguagem de Programação

Redes de Petri com:

- Tipos de Dados
- Modularidade

High-level Petri Net e Hierarchical Petri Net

Modelos sucintos e estruturados para determinadas situações

Token “carrega” um dado de um determinado tipo

Redes de Petri Coloridas (CPN)

Coloured por razões históricas:

- *Colour set* = Tipo
- *Token Colour* = *Token Value*

Possuem mesmo poder de expressividade de PT-nets

- Uma pode ser traduzida na outra
- Sem ganho teórico

Adoção da linguagem funcional ML (para a CPN considerada)

Técnicas de análise: grafo de alcançabilidade, invariantes, etc.

Exemplo

Pacotes

01010101 - 1
01011111 - 2
11110000 - 3
11112111 - 4

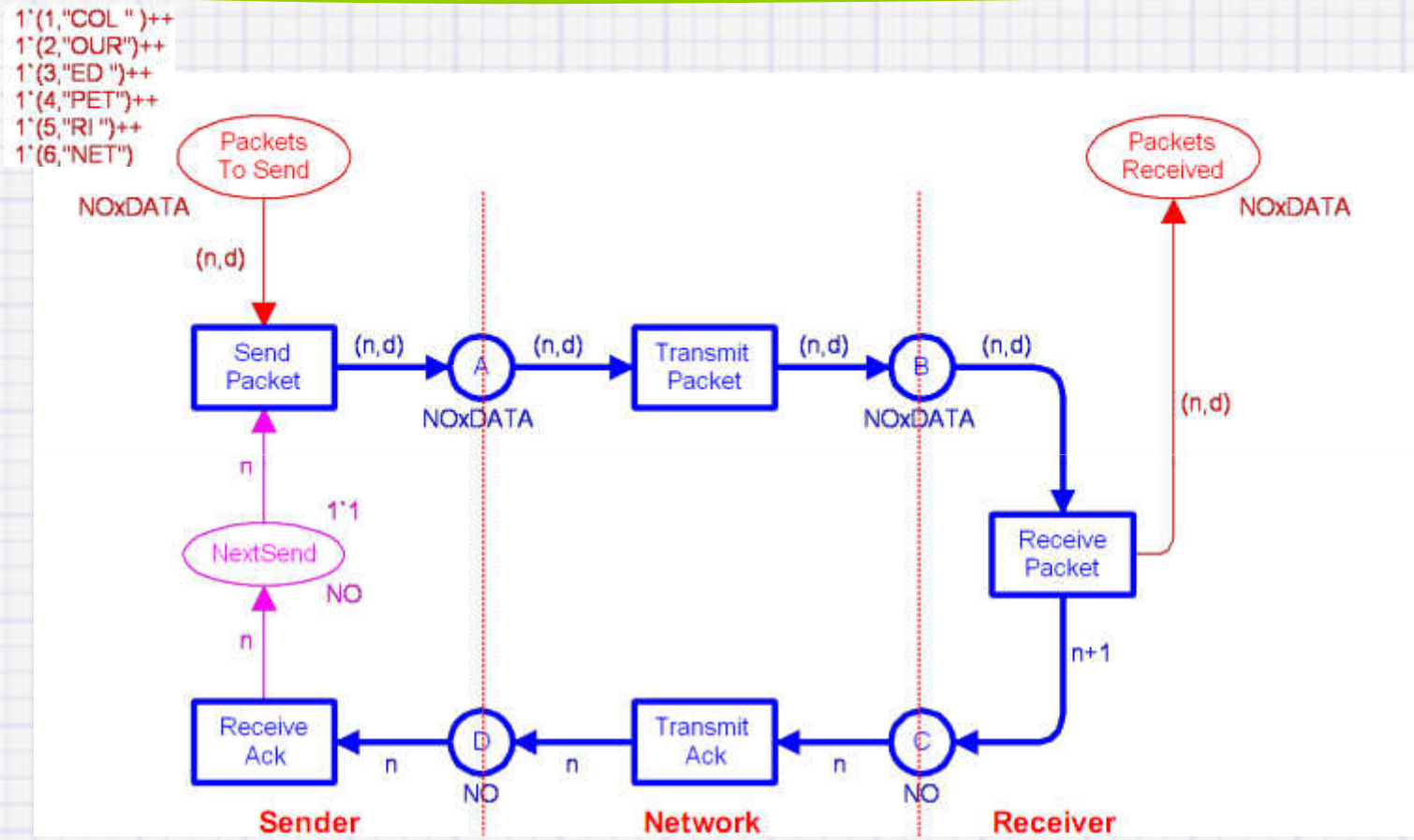
Receptor precisa juntar os dados recebidos

Protocolo *stop-and-wait* (i) transmite um pacote por vez;
(ii) Aguarda reconhecimento de recebimento

Assumir uma rede confiável

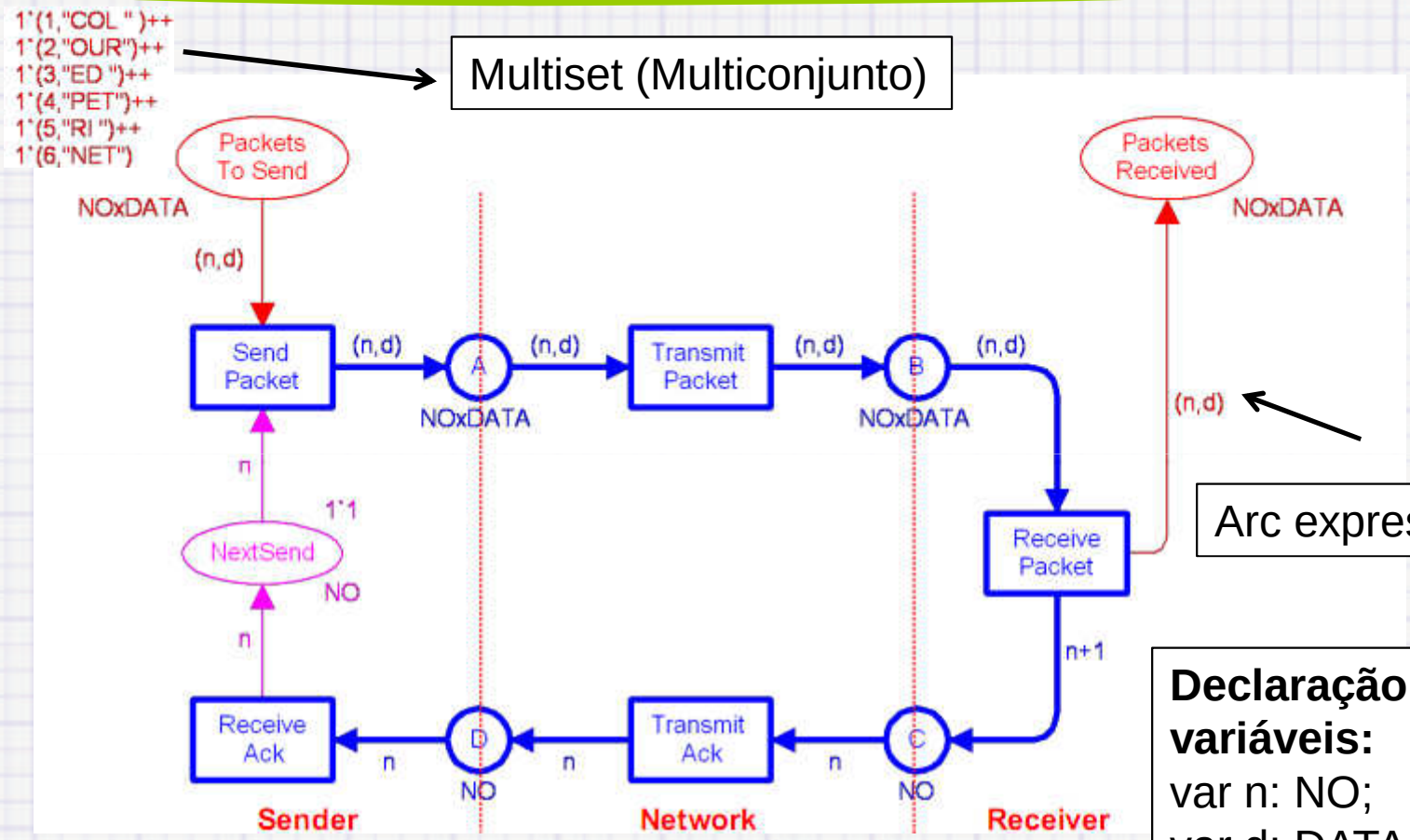
Baseado em ACPN-2010 por Lars Kristensen

Exemplo



Baseado em ACPN-2010 por Lars Kristensen

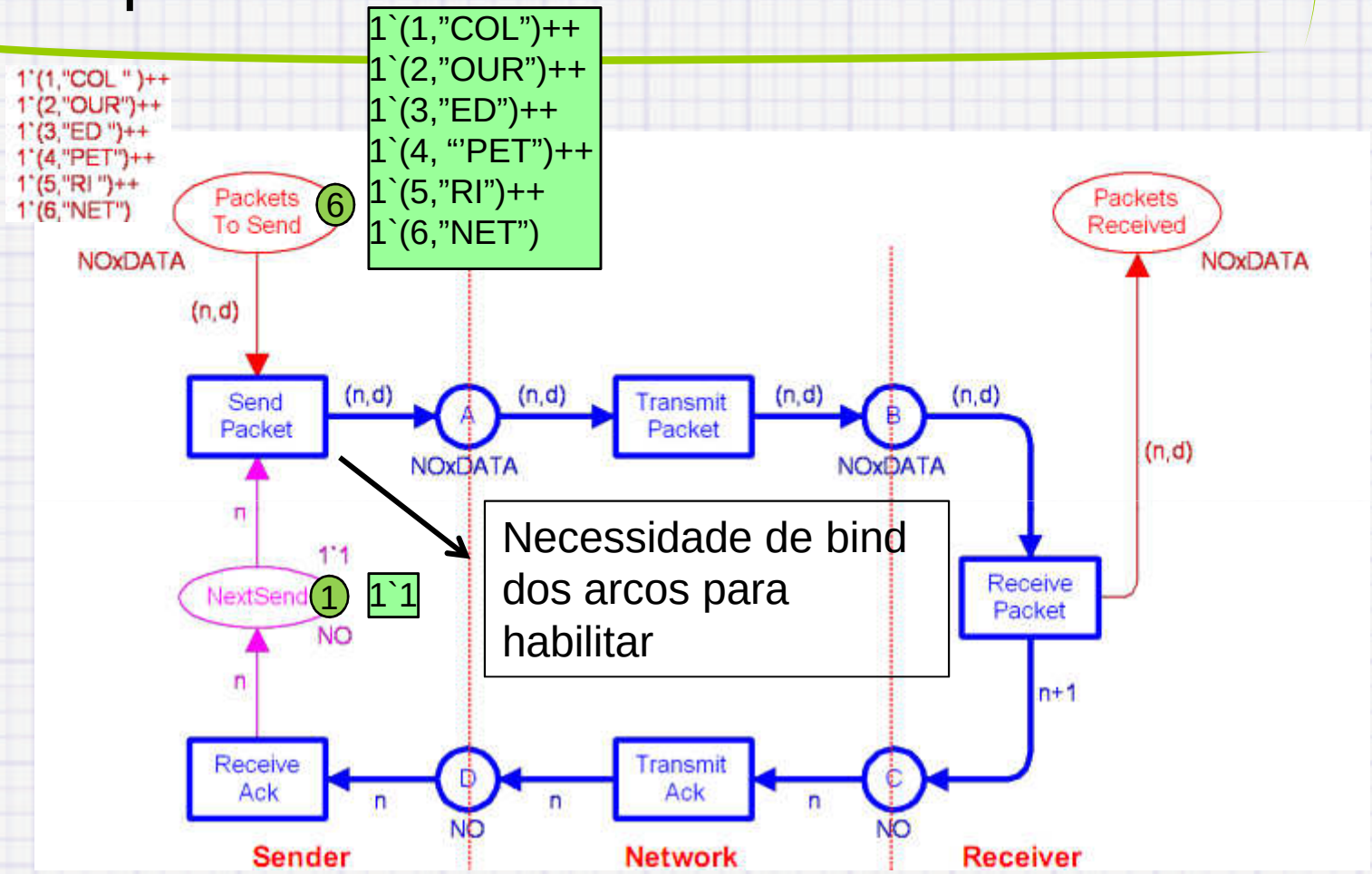
Exemplo



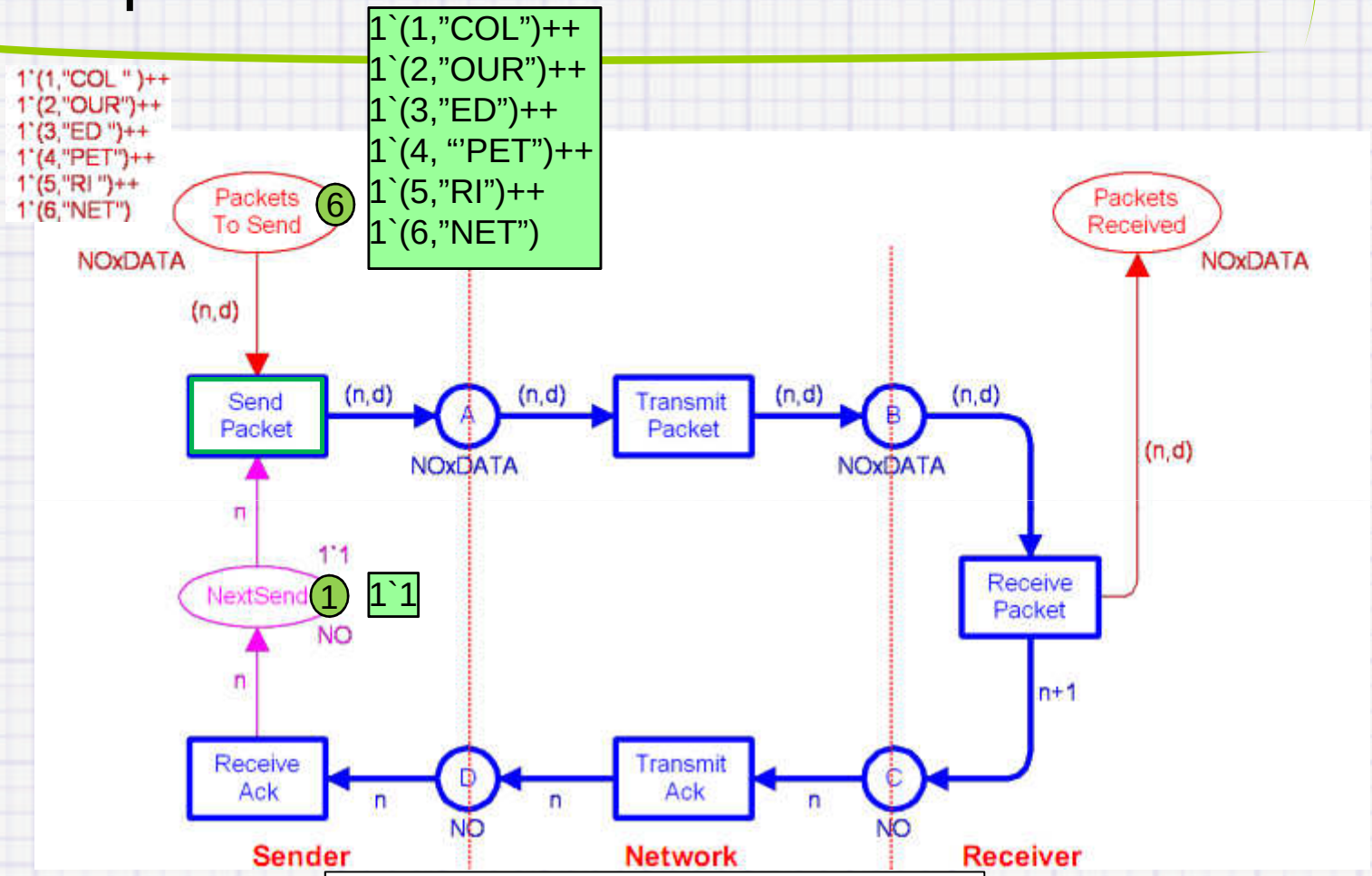
Declaração de variáveis:
var n: NO;
var d: DATA;

Definição de Colour Sets (Tipos):
colset NO = int;
colset DATA = string;
colset NOxDATA = product NO * DATA;

Exemplo


1`(1,"COL ")++
1`(2,"OUR")++
1`(3,"ED ")++
1`(4,"PET")++
1`(5,"RI ")++
1`(6,"NET")
1`(1,”COL”)++
1`(2,”OUR”)++
1`(3,”ED”)++
1`(4, “’PET”)++
1`(5,”RI”)++
1`(6,”NET”)
Packets To Send
6
NOxDATA
(n,d)
Send Packet
(n,d)
A
NOxDATA
(n,d)
Transmit Packet
(n,d)
B
NOxDATA
(n,d)
Packets Received
NOxDATA
(n,d)
Receive Packet
n+1
C
NO
n
Transmit Ack
n
D
NO
n
Receive Ack
n
NextSend
1
NO
1`1
1`1
n
Necessidade de bind dos arcos para habilitar
Sender
Network
Receiver

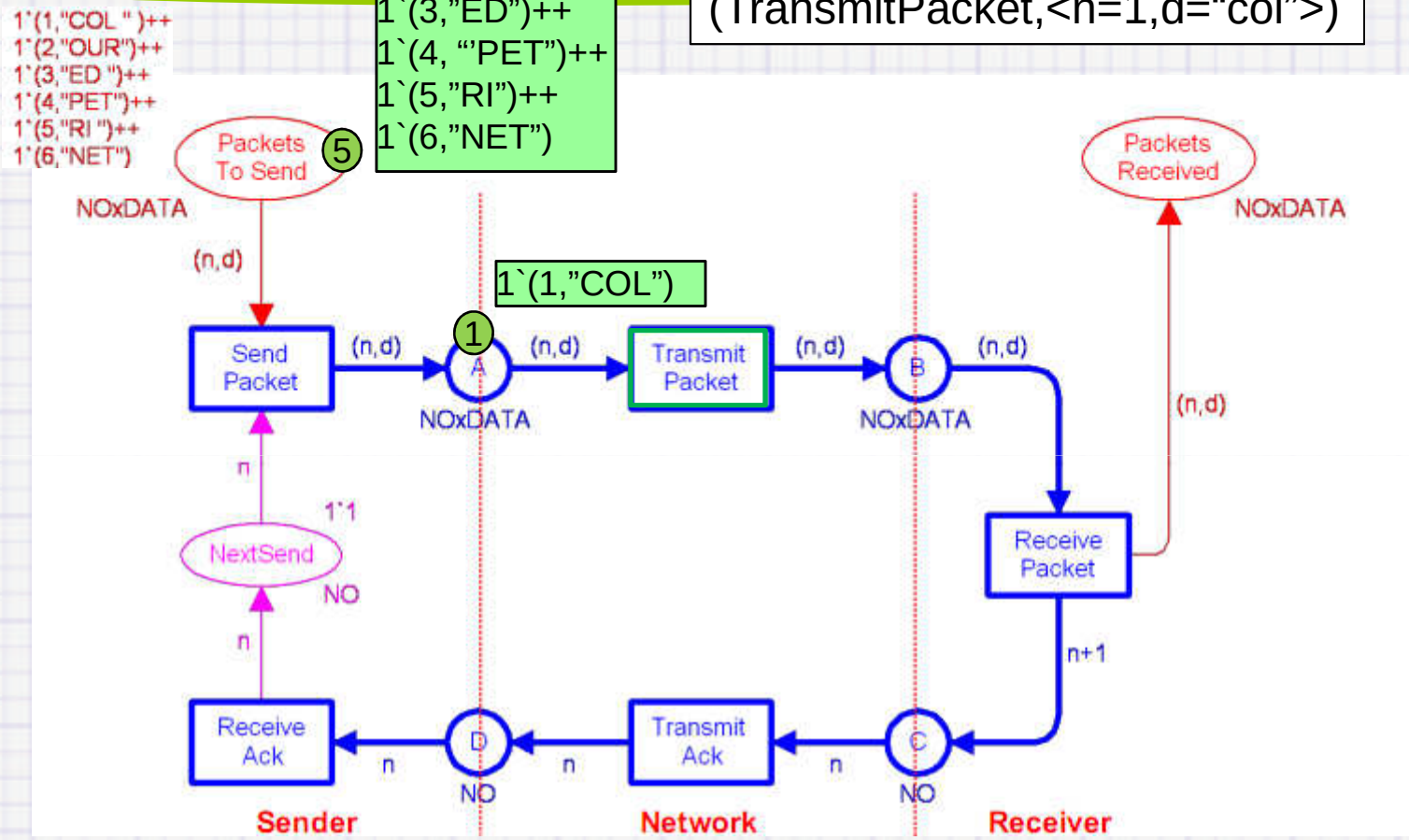
Exemplo



Binding Element
(SendPacket,<n=1,d=“col”>)
Avaliação de Arc Expressions
n -> 1
(n,d) -> (1,”col”)

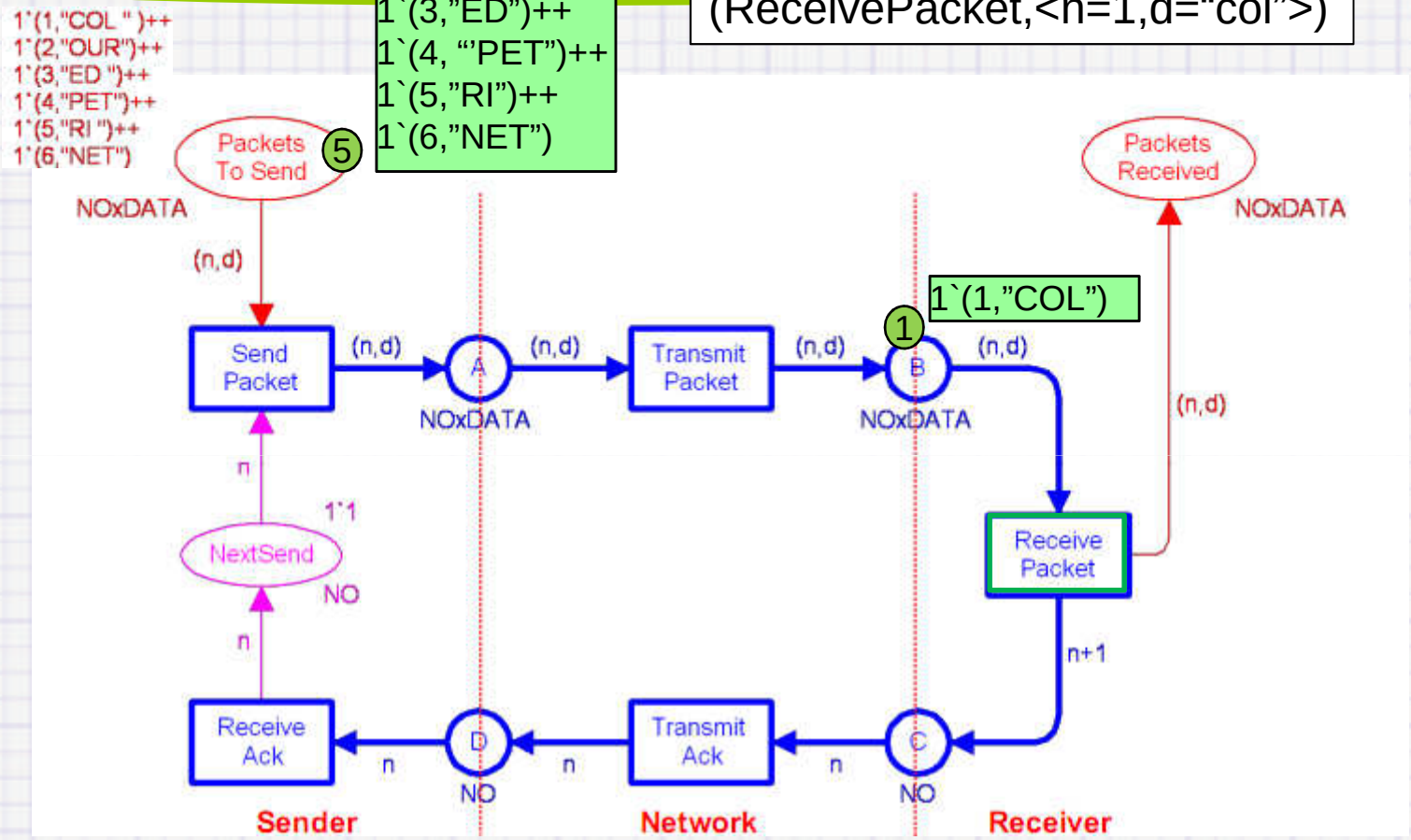
Exemplo

Binding Element
(TransmitPacket,<n=1,d="col">)

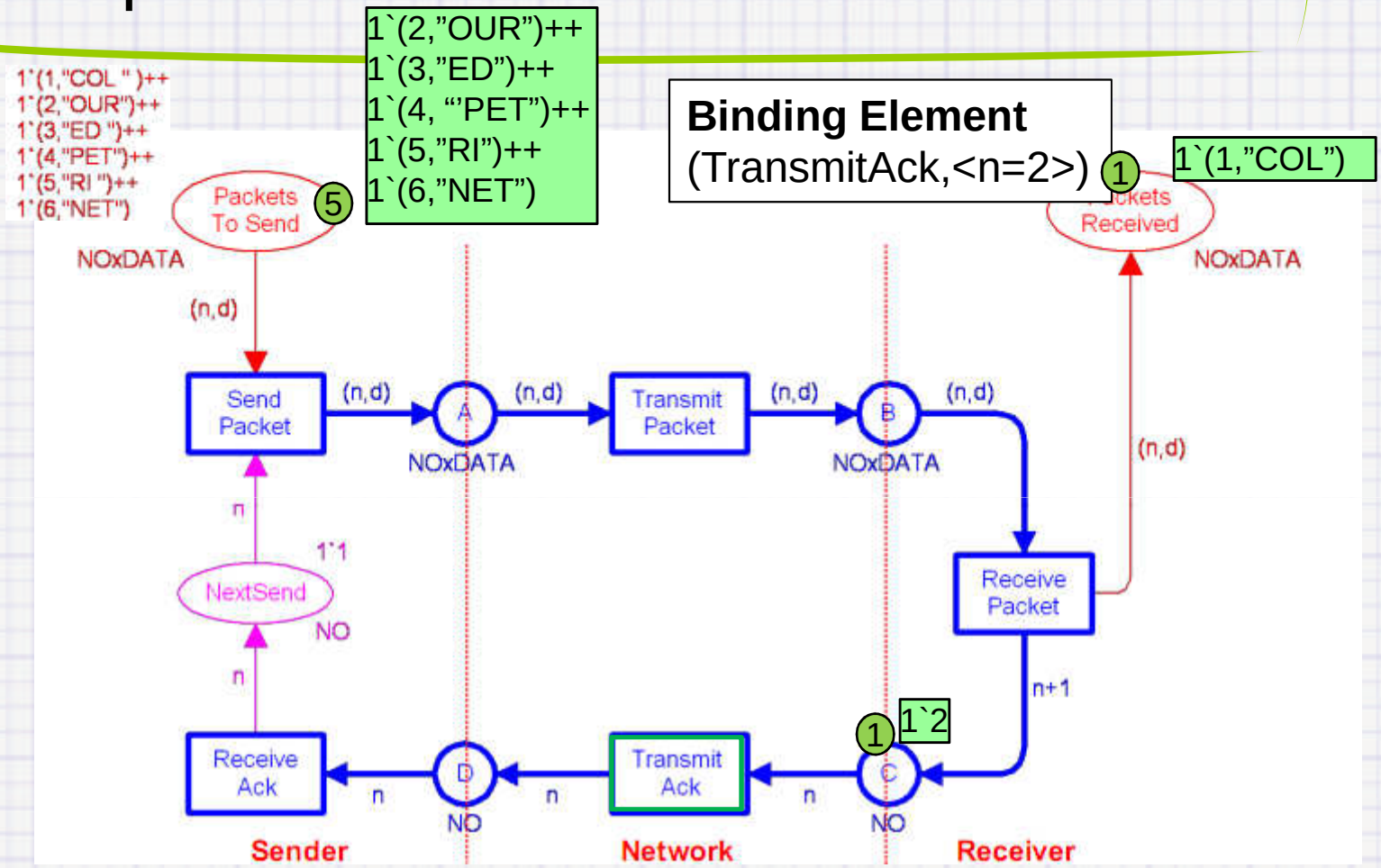


Exemplo

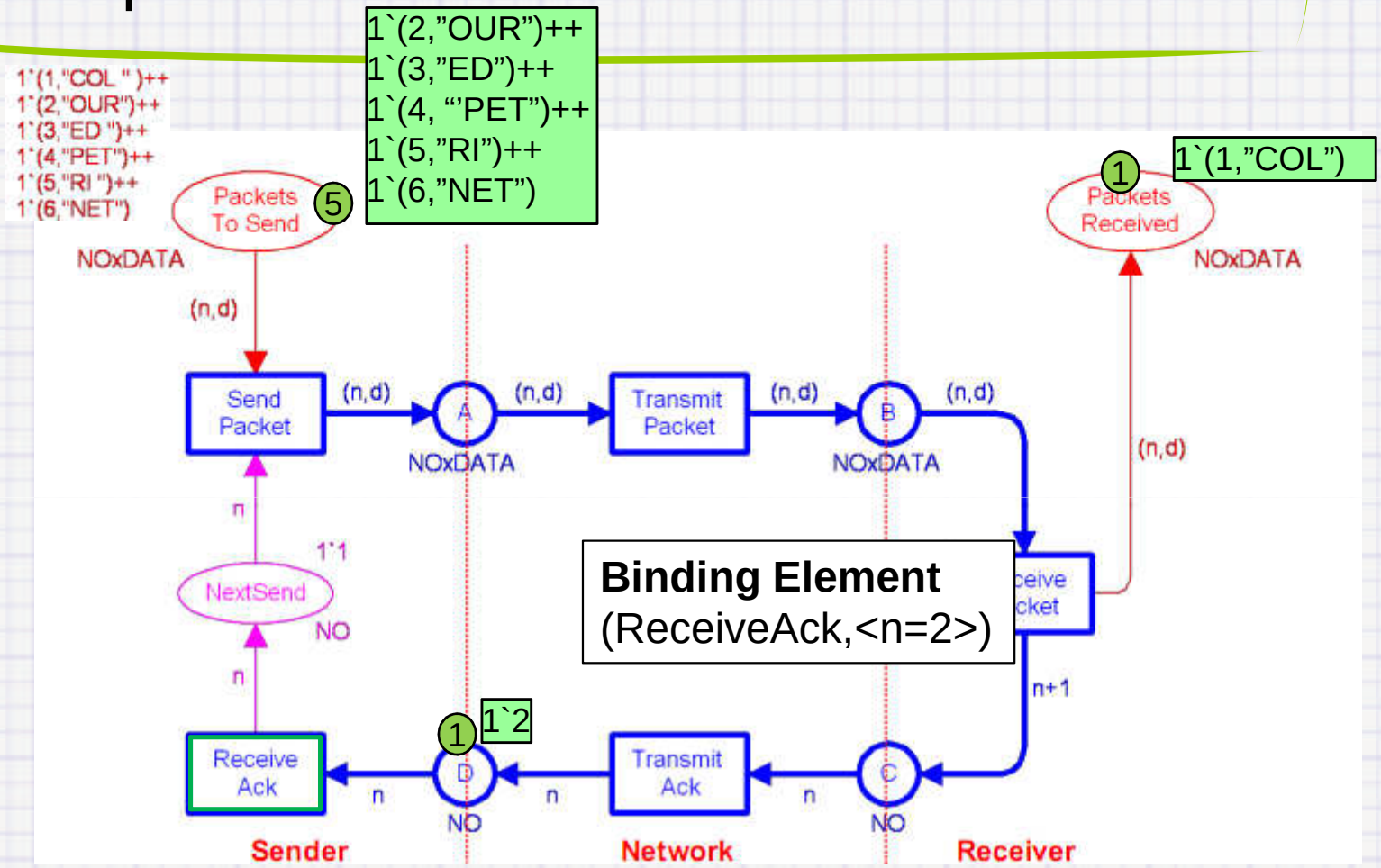
Binding Element
(ReceivePacket,<n=1,d="col">)



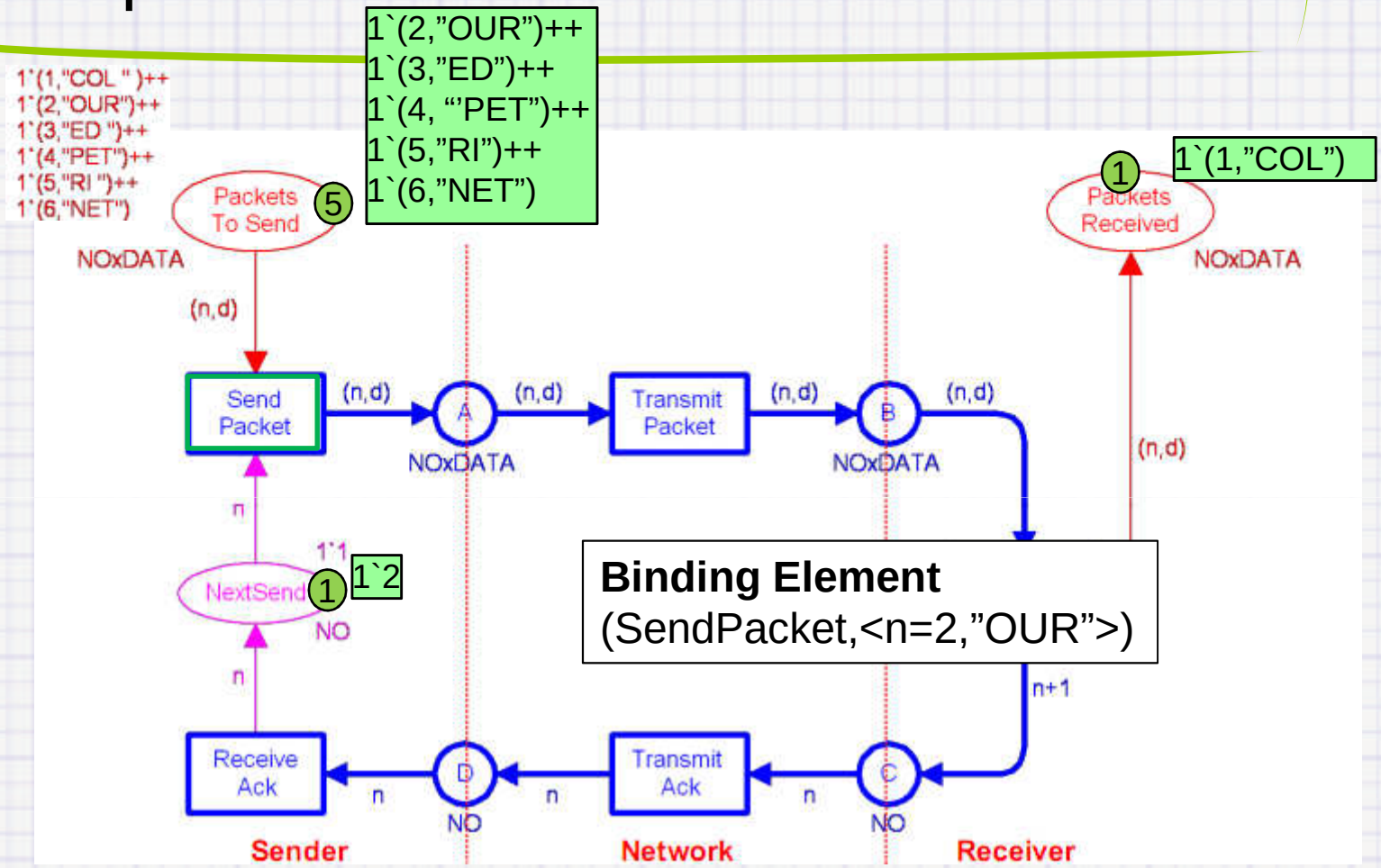
Exemplo


1`(1,"COL ")++
1`(2,"OUR")++
1`(3,"ED ")++
1`(4,"PET")++
1`(5,"RI ")++
1`(6,"NET")
1`(2,”OUR”)++
1`(3,”ED”)++
1`(4, “’PET”)++
1`(5,”RI”)++
1`(6,”NET”)
Binding Element
(TransmitAck,<n=2>)
1`(1,”COL”)
Packets To Send
5
NOxDATA
(n,d)
Send Packet
A
NOxDATA
Transmit Packet
B
NOxDATA
Receive Packet
Packets Received
NOxDATA
n
1`1
NextSend
NO
Receive Ack
D
NO
Transmit Ack
C
NO
1`2
n+1
Sender
Network
Receiver

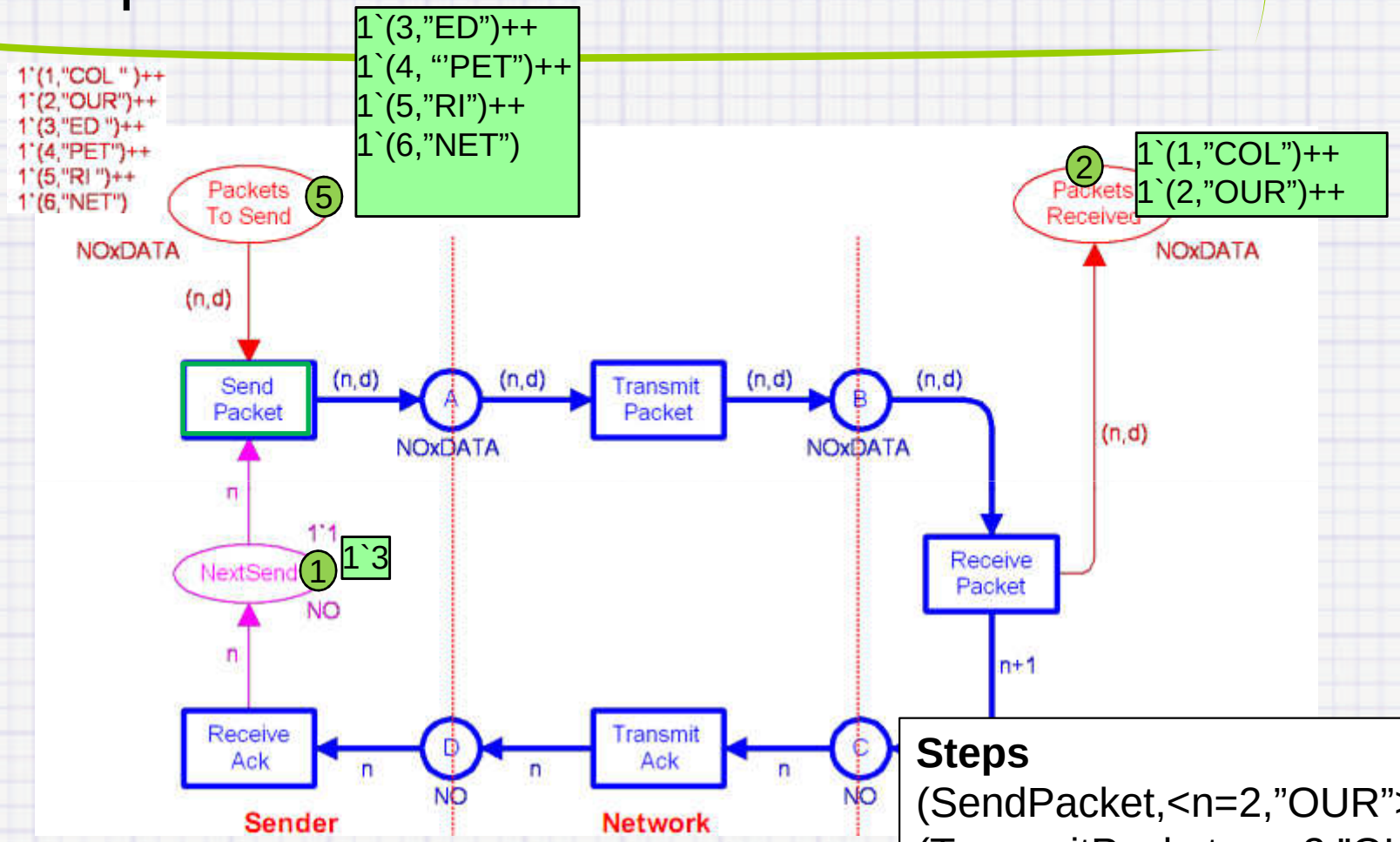
Exemplo


1`(1,"COL ")++
1`(2,"OUR")++
1`(3,"ED ")++
1`(4,"PET")++
1`(5,"RI ")++
1`(6,"NET")
1`(2,”OUR”)++
1`(3,”ED”)++
1`(4, “’PET”)++
1`(5,”RI”)++
1`(6,”NET”)
1`(1,”COL”)
Packets To Send
5
Packets Received
1
NOxDATA
NOxDATA
(n,d)
Send Packet
(n,d)
A
(n,d)
Transmit Packet
(n,d)
B
(n,d)
NOxDATA
NOxDATA
(n,d)
n
1`1
NextSend
NO
Binding Element
(ReceiveAck,<n=2>)
n+1
n
1`2
1
Receive Ack
n
D
n
Transmit Ack
n
C
NO
NO
Sender
Network
Receiver

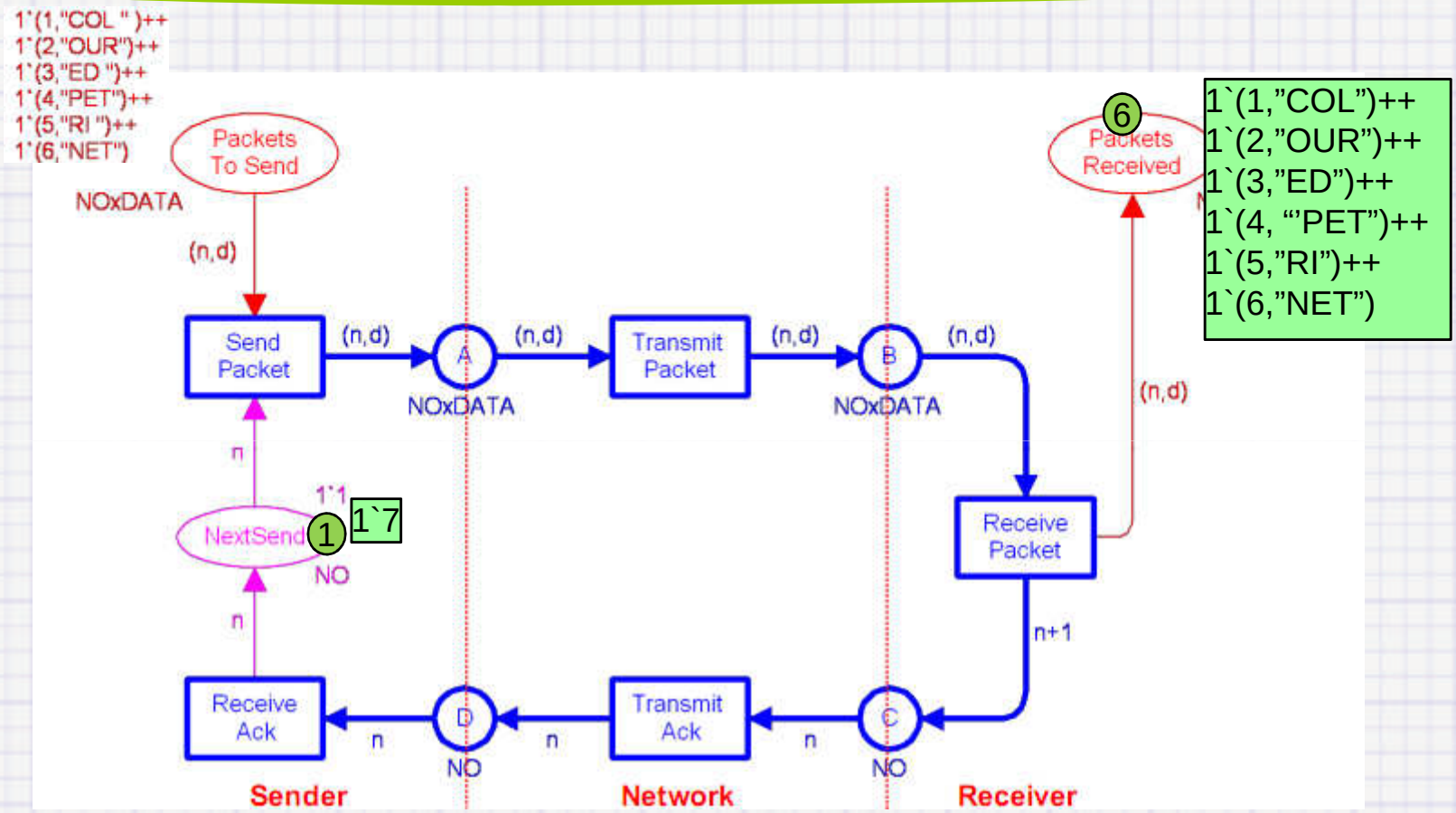
Exemplo


1`(2,"OUR")++
1`(3,"ED")++
1`(4, "'PET")++
1`(5,"RI")++
1`(6,"NET")
1`(1,"COL")
1`(1,"COL ")++
1`(2,"OUR")++
1`(3,"ED ")++
1`(4,"PET")++
1`(5,"RI ")++
1`(6,"NET")
Packets To Send
5
NOxDATA
(n,d)
Send Packet
(n,d)
A
NOxDATA
(n,d)
Transmit Packet
(n,d)
B
NOxDATA
(n,d)
Packets Received
1
NOxDATA
(n,d)
n
1`1
NextSend
1
1`2
NO
Binding Element
(SendPacket,<n=2,"OUR">)
n+1
n
Receive Ack
n
D
NO
n
Transmit Ack
n
C
NO
Sender
Network
Receiver

Exemplo


1`(1,"COL ")++
1`(2,"OUR")++
1`(3,"ED ")++
1`(4,"PET")++
1`(5,"RI ")++
1`(6,"NET")
1`(3,"ED")++
1`(4, "'PET")++
1`(5,"RI")++
1`(6,"NET")
1`(1,"COL")++
1`(2,"OUR")++
Packets To Send
5
NOxDATA
(n,d)
Send Packet
(n,d)
A
NOxDATA
(n,d)
Transmit Packet
(n,d)
B
NOxDATA
(n,d)
2
Packets Received
NOxDATA
(n,d)
Receive Packet
n+1
n
1`1
NextSend
1
1`3
NO
n
Receive Ack
n
D
NO
n
Transmit Ack
n
C
NO
Sender
Network
Steps
(SendPacket,<n=2,"OUR">)
(TransmitPacket,<n=2,"OUR">)
(ReceivePacket,<n=2,"OUR">)
(TransmitAck,<n=3>)
(ReceiveAck,<n=3)

Exemplo

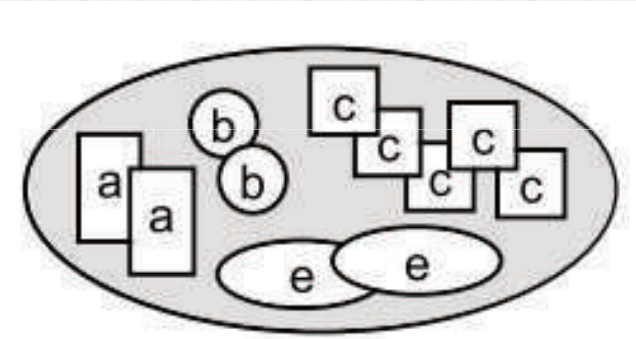


Depois de 30Steps

Multiset (Multiconjunto)

Assumindo S = {s1,s2,s3,...}, um multiset *m* *é* uma função $m:S \rightarrow \aleph$. Pode ser representado como:

$$^{++}\sum_{s \in S} m(s)`s = m(s1)`s1 ++ m(s2)`s2 ++ m(s3)`s3 ++ ...$$

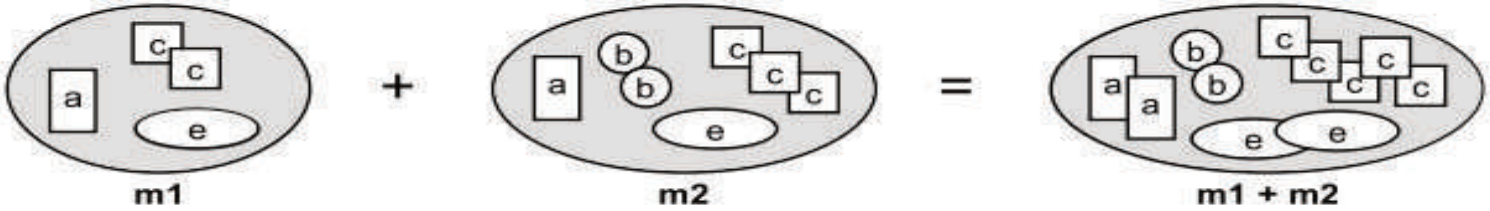


Ex: S={a,b,c,e}

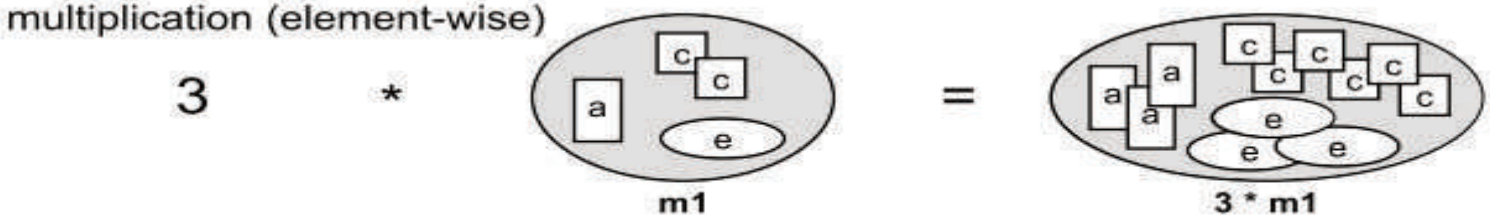
$$^{++}\sum_{s \in S} m(s)`s = 2`a ++ 2`b ++ 5`c ++ 2`e$$

$$m(s) = \begin{cases} 2 \text{ if } s = a \\ 2 \text{ if } s = b \\ 5 \text{ if } s = c \\ 2 \text{ if } s = e \\ 0 \text{ caso contrário} \end{cases}$$

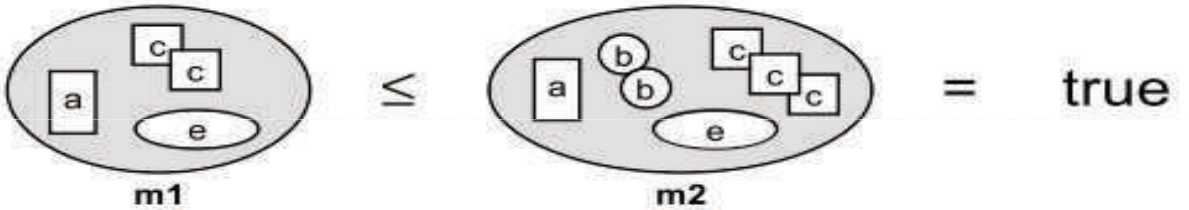
Addition (element-wise)



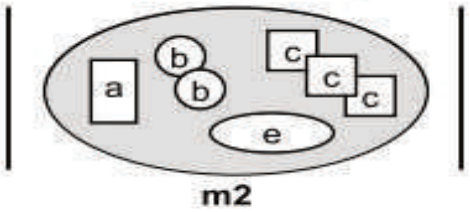
Scalar multiplication (element-wise)



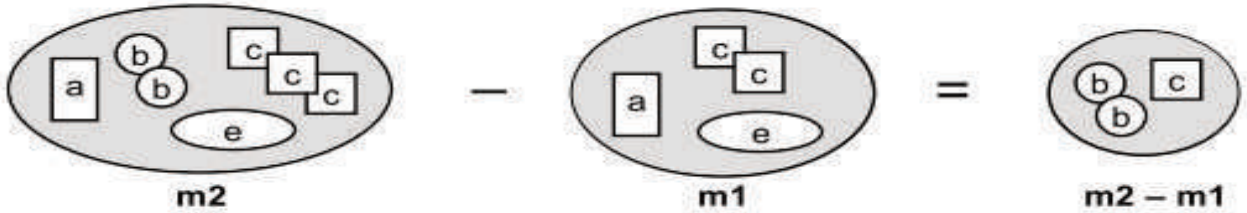
Comparison (element-wise)



Size (number of elements)



Subtraction (only if m2 ≥ m1)



CPN – Não hierárquica

Definition 4.2. A **non-hierarchical Coloured Petri Net** is a nine-tuple $CPN = (P, T, A, \Sigma, V, C, G, E, I)$, where:

1. P is a finite set of **places**.
2. T is a finite set of **transitions** T such that $P \cap T = \emptyset$.
3. $A \subseteq P \times T \cup T \times P$ is a set of directed **arcs**.

4. Σ is a finite set of non-empty **colour sets**.
5. V is a finite set of **typed variables** such that $Type[v] \in \Sigma$ for all variables $v \in V$.

6. $C : P \rightarrow \Sigma$ is a **colour set function** that assigns a colour set to each place.
7. $G : T \rightarrow EXPR_V$ is a **guard function** that assigns a guard to each transition t such that $Type[G(t)] = Bool$.
8. $E : A \rightarrow EXPR_V$ is an **arc expression function** that assigns an arc expression to each arc a such that $Type[E(a)] = C(p)_{MS}$, where p is the place connected to the arc a.
9. $I : P \rightarrow EXPR_\emptyset$ is an **initialisation function** that assigns an initialisation expression to each place p such that $Type[I(p)] = C(p)_{MS}$.

CPN – Não hierárquica

Definition 4.3. For a Coloured Petri Net $CPN = (P, T, A, \Sigma, V, C, G, E, I)$, we define the following concepts:

1. A **marking** is a function M that maps each place $p \in P$ into a multiset of tokens $M(p) \in C(p)_{MS}$.
2. The **initial marking** M_0 is defined by $M_0(p) = I(p)\langle\rangle$ for all $p \in P$.
3. The **variables of a transition** t are denoted $Var(t) \subseteq V$ and consist of the free variables appearing in the guard of t and in the arc expressions of arcs connected to t.
4. A **binding** of a transition t is a function b that maps each variable $v \in Var(t)$ into a value $b(v) \in Type[v]$. The set of all bindings for a transition t is denoted $B(t)$.
5. A **binding element** is a pair (t, b) such that $t \in T$ and $b \in B(t)$. The set of all binding elements $BE(t)$ for a transition t is defined by $BE(t) = \{(t, b) \mid b \in B(t)\}$. The set of all binding elements in a CPN model is denoted BE.
6. A **step** $Y \in BE_{MS}$ is a non-empty, finite multiset of binding elements.

CPN – Não hierárquica

Definition 4.4. A binding element $(t,b) \in BE$ is **enabled** in a marking M if and only if the following two properties are satisfied:

1. $G(t)\langle b\rangle$.
2. $\forall p \in P : E(p,t)\langle b\rangle \ll= M(p)$.

When (t,b) is enabled in M, it may **occur**, leading to the marking M' defined by:

3. $\forall p \in P : M'(p) = (M(p) -- E(p,t)\langle b\rangle) ++ E(t,p)\langle b\rangle$.

Definition 4.5. A step $Y \in BE_{MS}$ is **enabled** in a marking M if and only if the following two properties are satisfied:

1. $\forall (t,b) \in Y : G(t)\langle b\rangle$.
2. $\forall p \in P : \quad \overset{++}{\underset{MS}{\sum}}_{(t,b)\in Y} E(p,t)\langle b\rangle \ll= M(p)$.

When Y is enabled in M, it may **occur**, leading to the marking M' defined by:

3. $\forall p \in P : M'(p) = (M(p) -- \overset{++}{\underset{MS}{\sum}}_{(t,b)\in Y} E(p,t)\langle b\rangle) ++ \overset{++}{\underset{MS}{\sum}}_{(t,b)\in Y} E(t,p)\langle b\rangle$.

CPN – Não hierárquica

Definition 4.4. A binding element $(t,b) \in BE$ is **enabled** in a marking M if and only if the following two properties are satisfied:

1. $G(t)\langle b \rangle$.
2. $\forall p \in P : E(p,t)\langle b \rangle \ll= M(p)$.

When (t,b) is enabled in M, it may **occur**, leading to the marking M' defined by:

3. $\forall p \in P : M'(p) = (M(p) -- E(p,t)\langle b \rangle) ++ E(t,p)\langle b \rangle$.

Definition 4.5. A step $Y \in BE_{MS}$ is **enabled** in a marking M if and only if the following two properties are satisfied:

1. $\forall (t,b) \in Y : G(t)\langle b \rangle$.
2. $\forall p \in P : \overset{++}{\underset{(t,b)\in Y}{\underset{MS}{\sum}}} E(p,t)\langle b \rangle \ll= M(p)$.

Interleaving será assumido!

When Y is enabled in M, it may **occur**, leading to the marking M' defined by:

3. $\forall p \in P : M'(p) = (M(p) -- \overset{++}{\underset{(t,b)\in Y}{\underset{MS}{\sum}}} E(p,t)\langle b \rangle) ++ \overset{++}{\underset{(t,b)\in Y}{\underset{MS}{\sum}}} E(t,p)\langle b \rangle$.

Exemplo 2

Pacotes

01010101 - 1
01011111 - 2
11110000 - 3
11112111 - 4

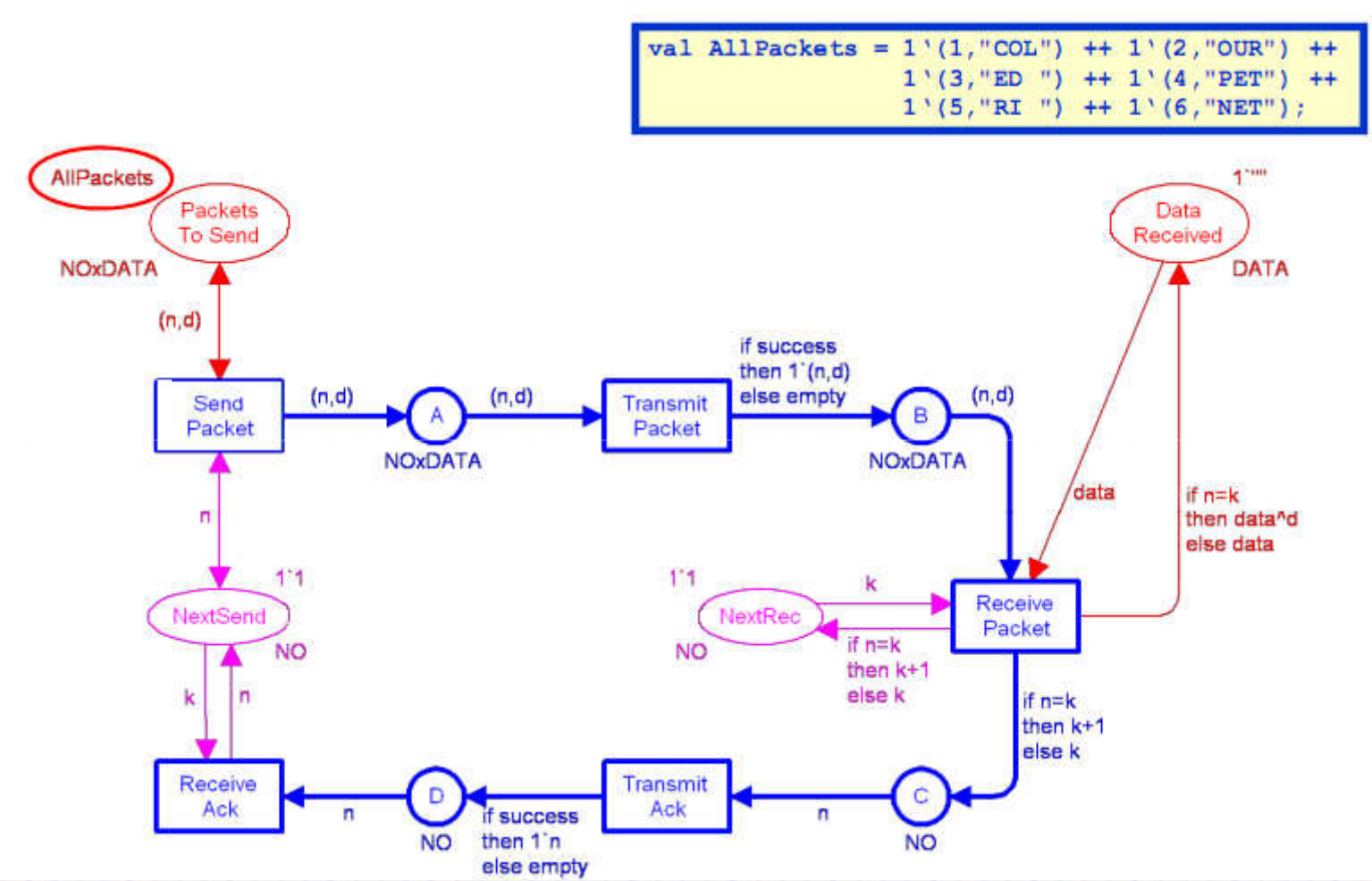
Rede não confiável (ex: perdas)

Transmissor pode retransmitir pacotes

Receptor aguarda o próximo pacote esperado

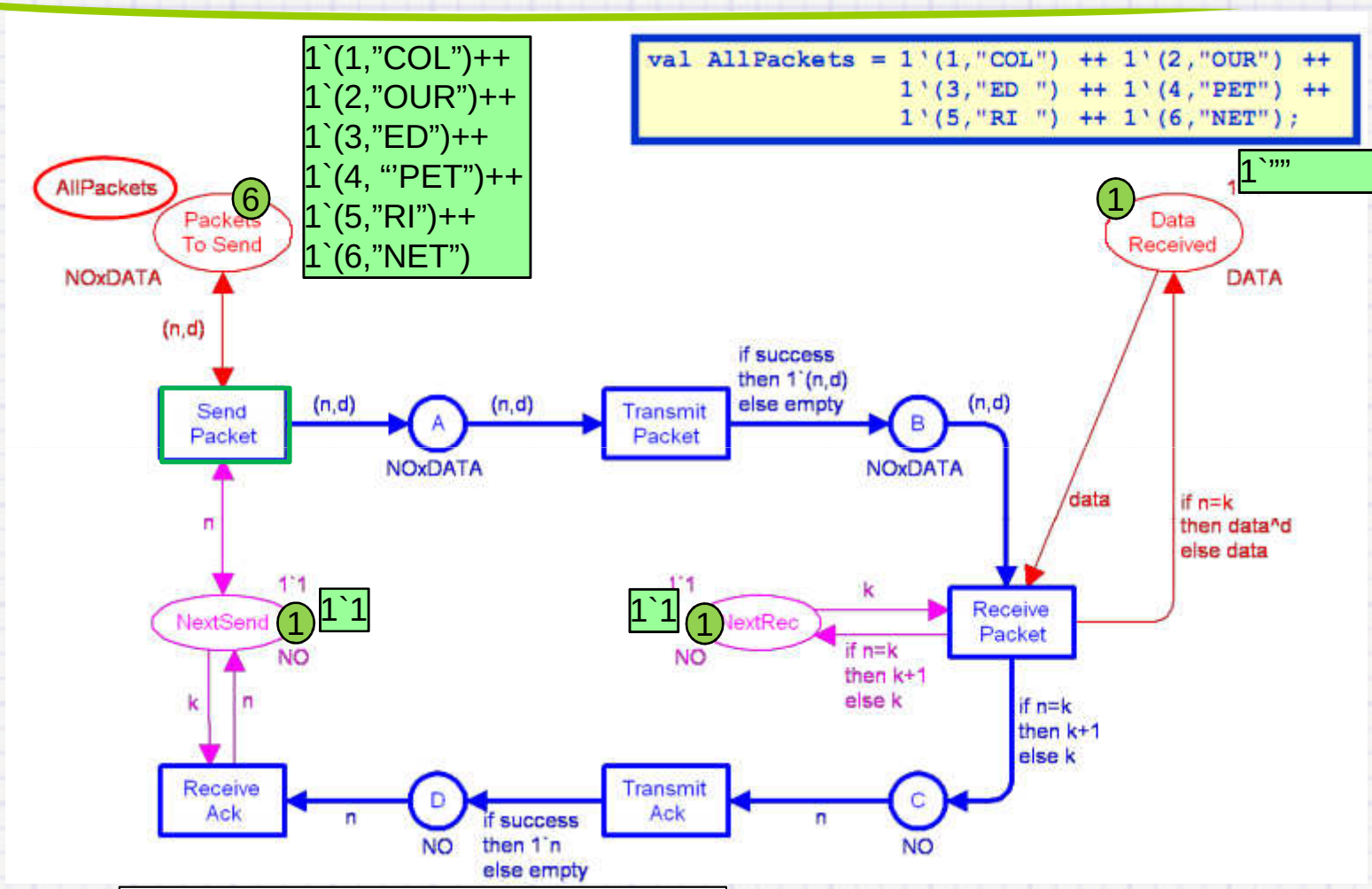
Baseado em ACPN-2010 por Lars Kristensen

Exemplo 2



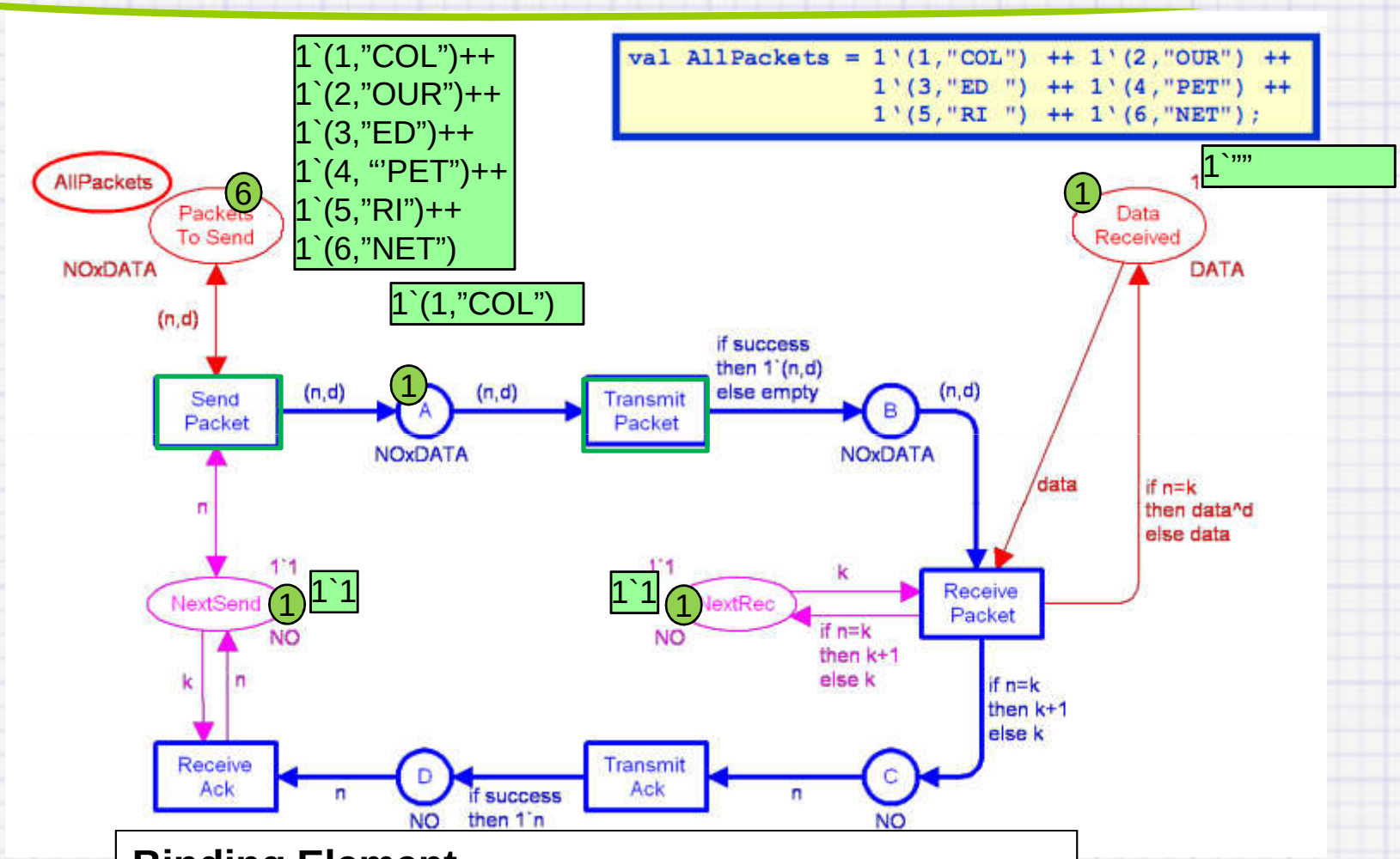
Baseado em ACPN-2010 por Lars Kristensen

Exemplo 2



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

Exemplo 2



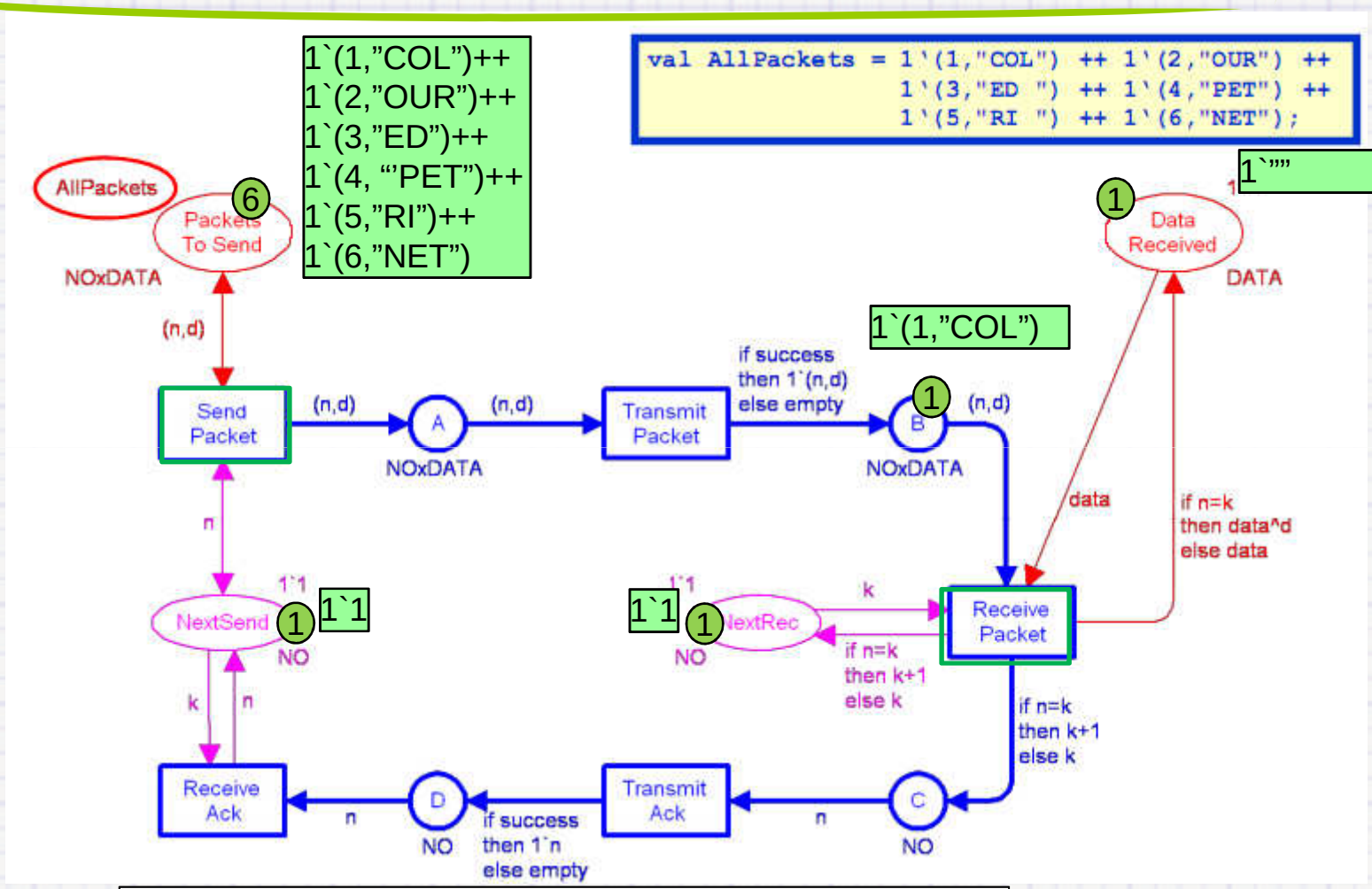
Binding Element

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

(TransmitPackage,<n=1,d="COL">,success=true)

(TransmitPackage,<n=1,d="COL",success=false>)

Exemplo 2

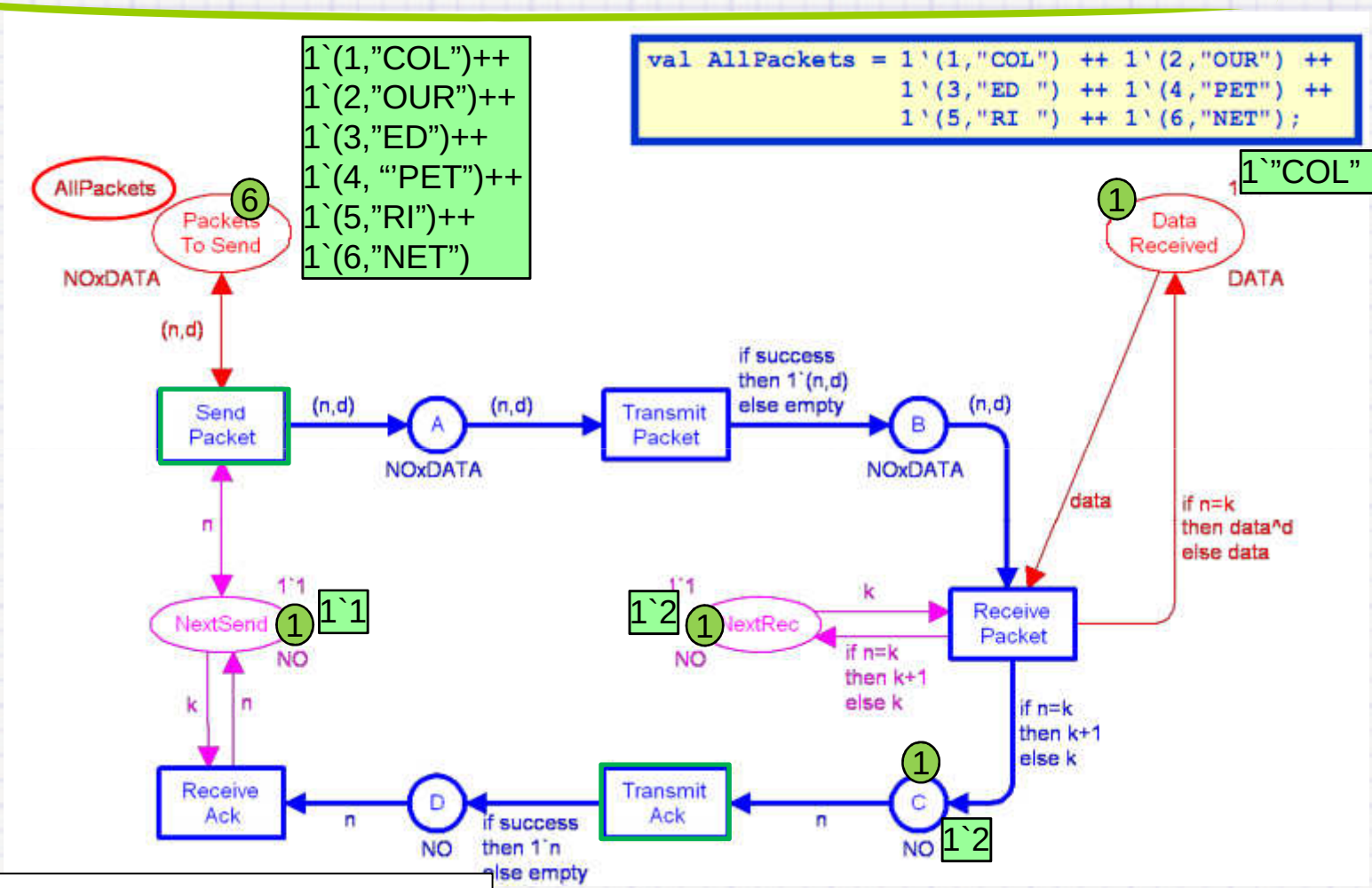


Binding Element

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

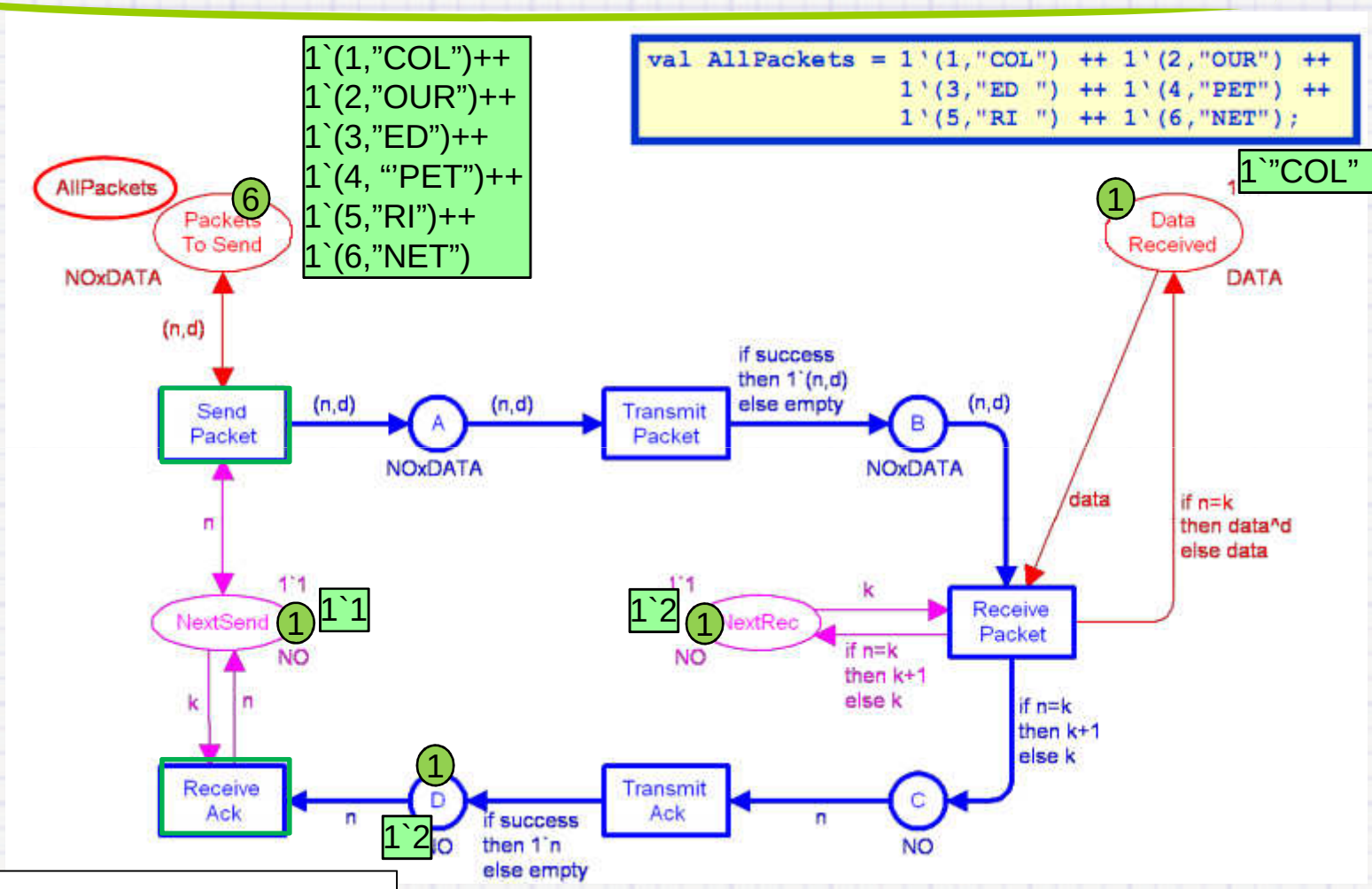
(ReceivePacket,<n=1,d="COL",k=1,data="">)

Exemplo 2



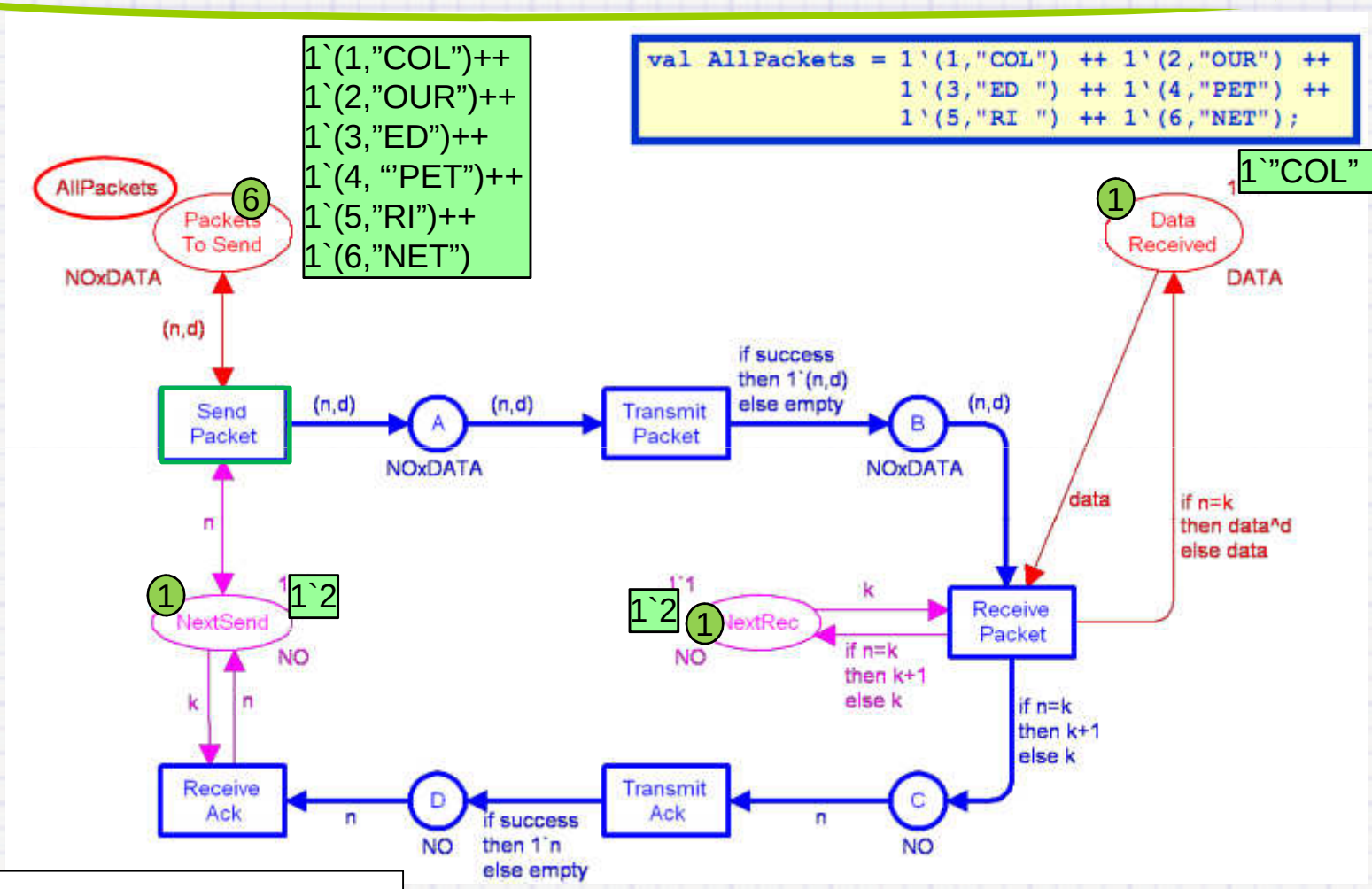
Binding Elements
(SendPacket,<n=1,d="COL">)
(TransmitAck,<n=2, success=true>
TransmitAck,<n=2, success=false>)

Exemplo 2



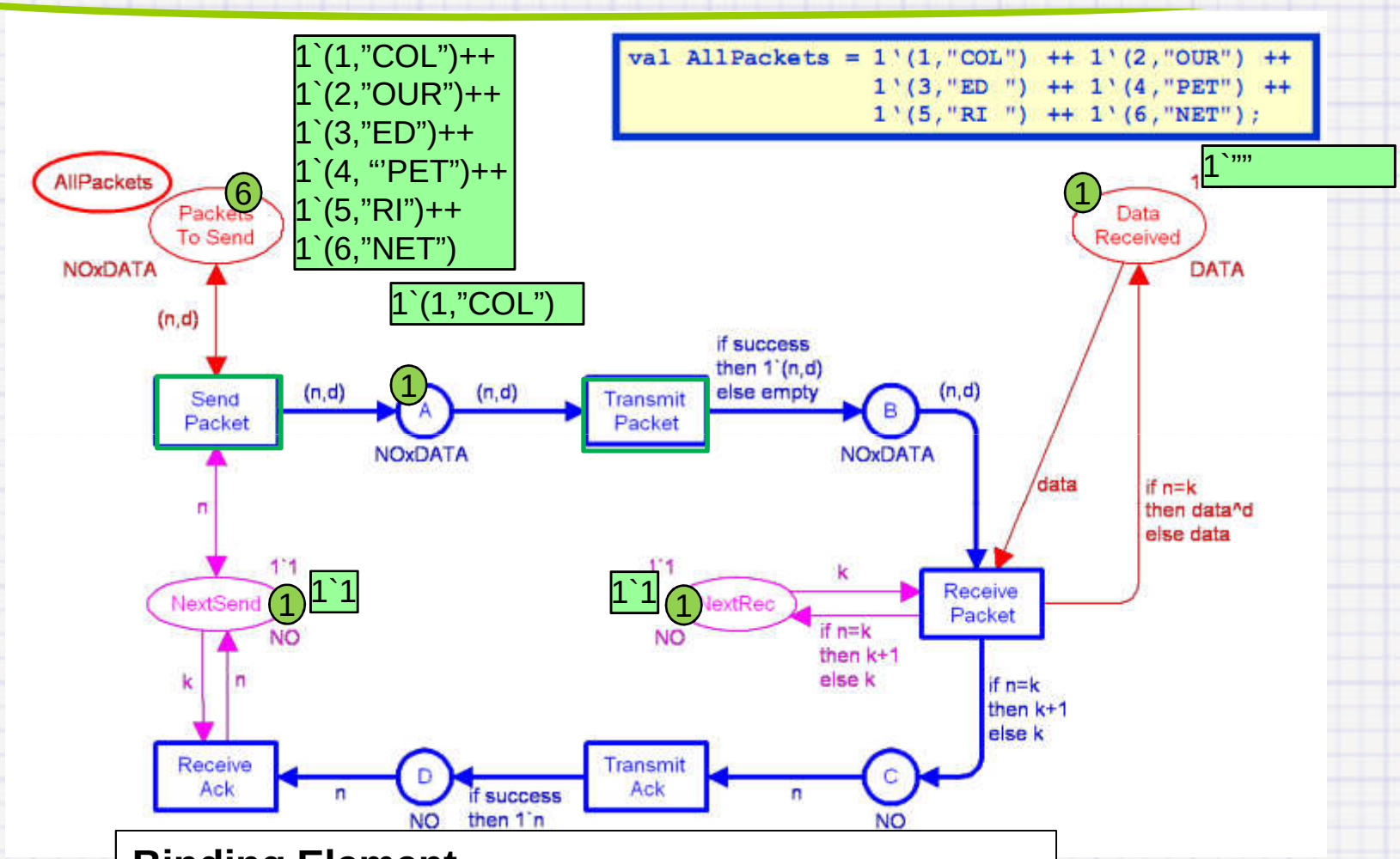
Binding Elements
(SendPacket,<n=1,d="COL">)
(ReceivePack,<n=2, k=1>)

Exemplo 2



Binding Elements
(SendPacket,<n=2,d="OUR">)

Exemplo 2 – Conflito e Concorrência



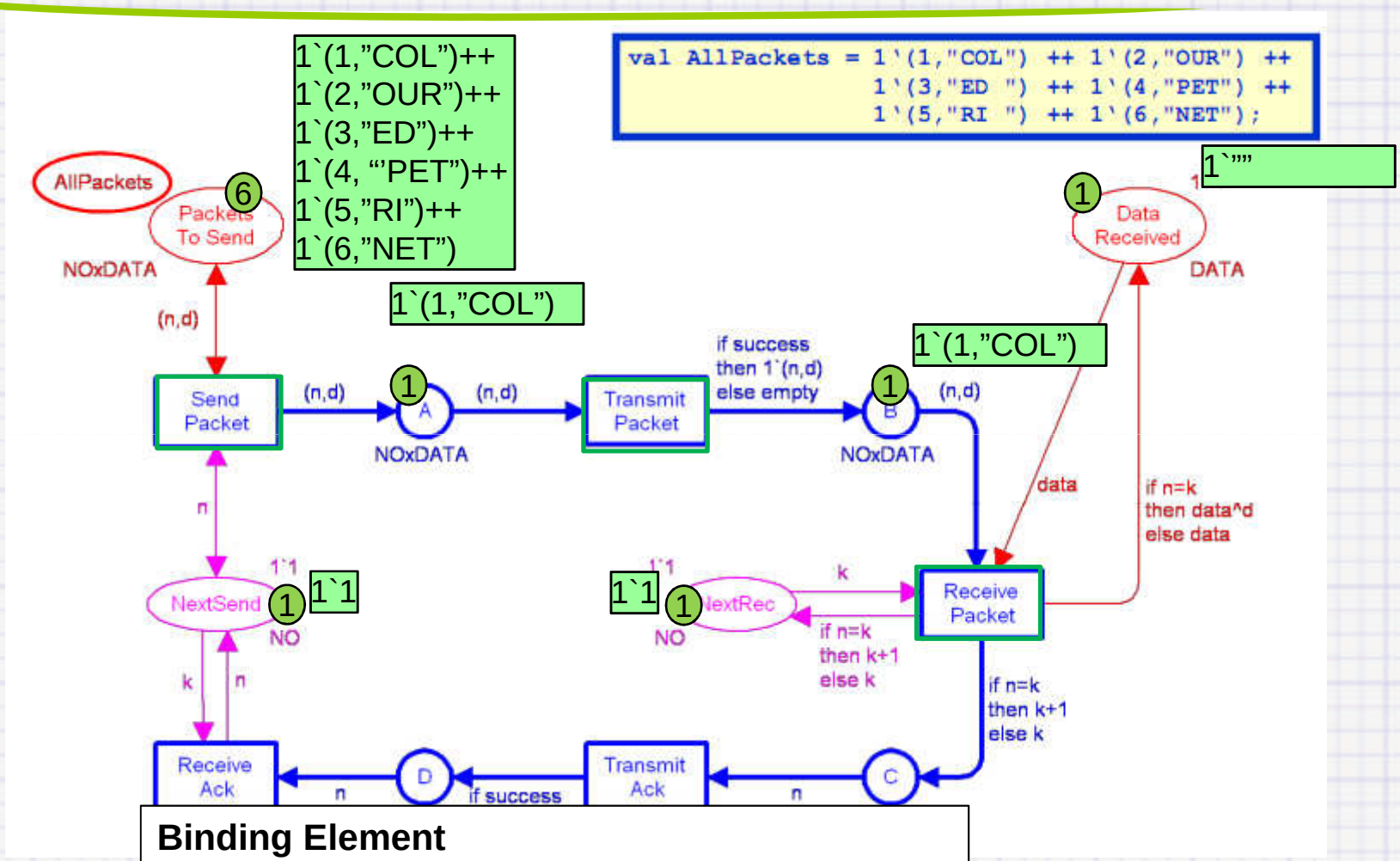
Binding Element

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

(TransmitPackage,<n=1,d="COL">,sucess=true)

(TransmitPackage,<n=1,d="COL",sucess=false>)

Exemplo 2 - Conflito e Concorrência



Binding Element

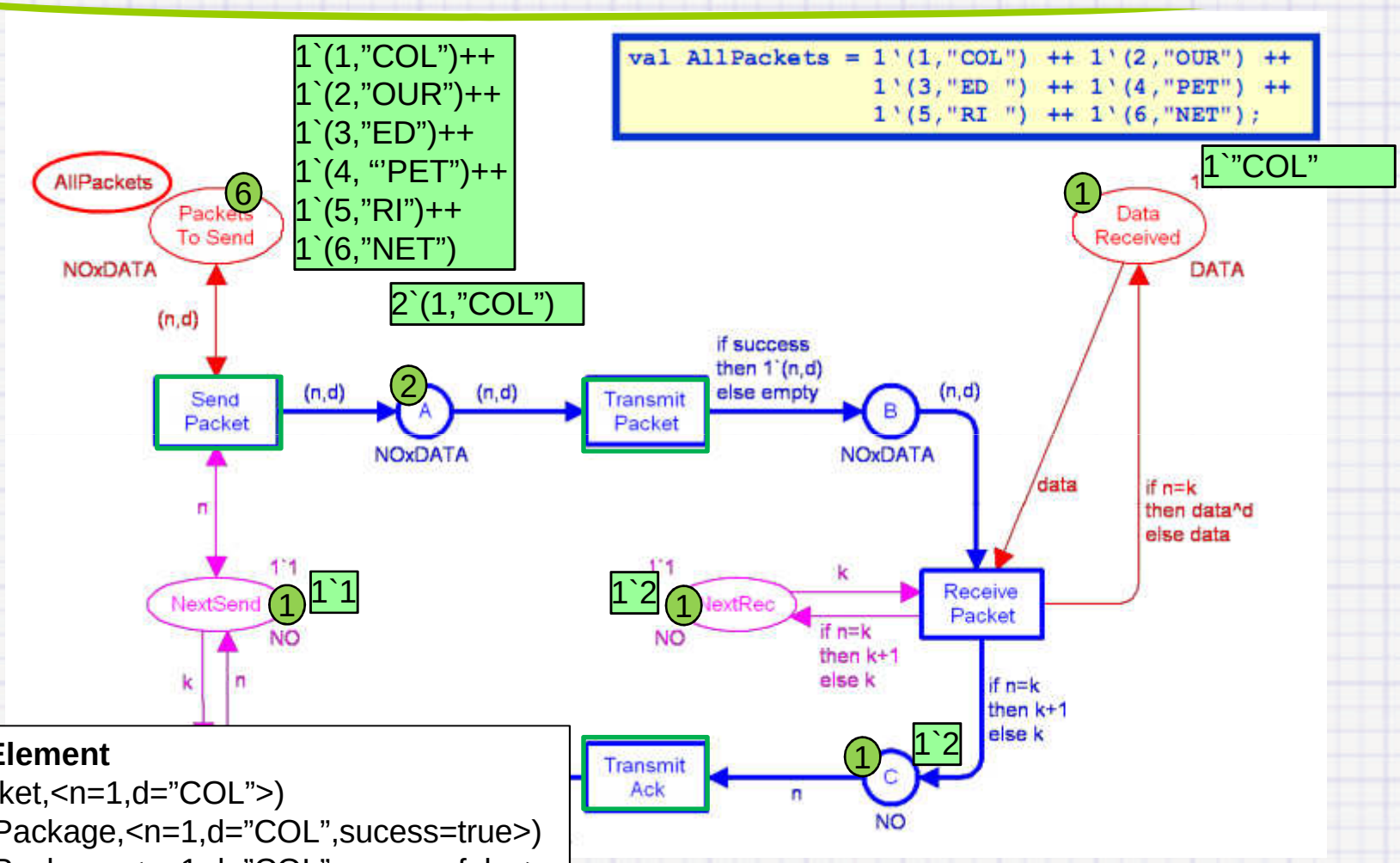
(SendPacket,<n=1,d="COL">)

(TransmitPackage,<n=1,d="COL",sucess=true>)

(TransmitPackage,<n=1,d="COL",sucess=false>

(ReceivePacket, ,<n=1,d="COL",data="",k=1)

Exemplo 2 - Conflito e Concorrência



Binding Element
(SendPacket,<n=1,d="COL">)
(TransmitPackage,<n=1,d="COL",sucess=true>)
(TransmitPackage,<n=1,d="COL",sucess=false>
)
(TransmitAck, ,<n=2, sucess=true>)
(TransmitAck, ,<n=2, sucess=false>)

Espaço de Estados

Os espaço de estados de uma CPN é um grafo dirigido SS = (N_{SS}, A_{SS}) com os arcos rotulados a partir de BE, no qual

1. $N_{SS} = R(M_0)$ é o conjunto de nós

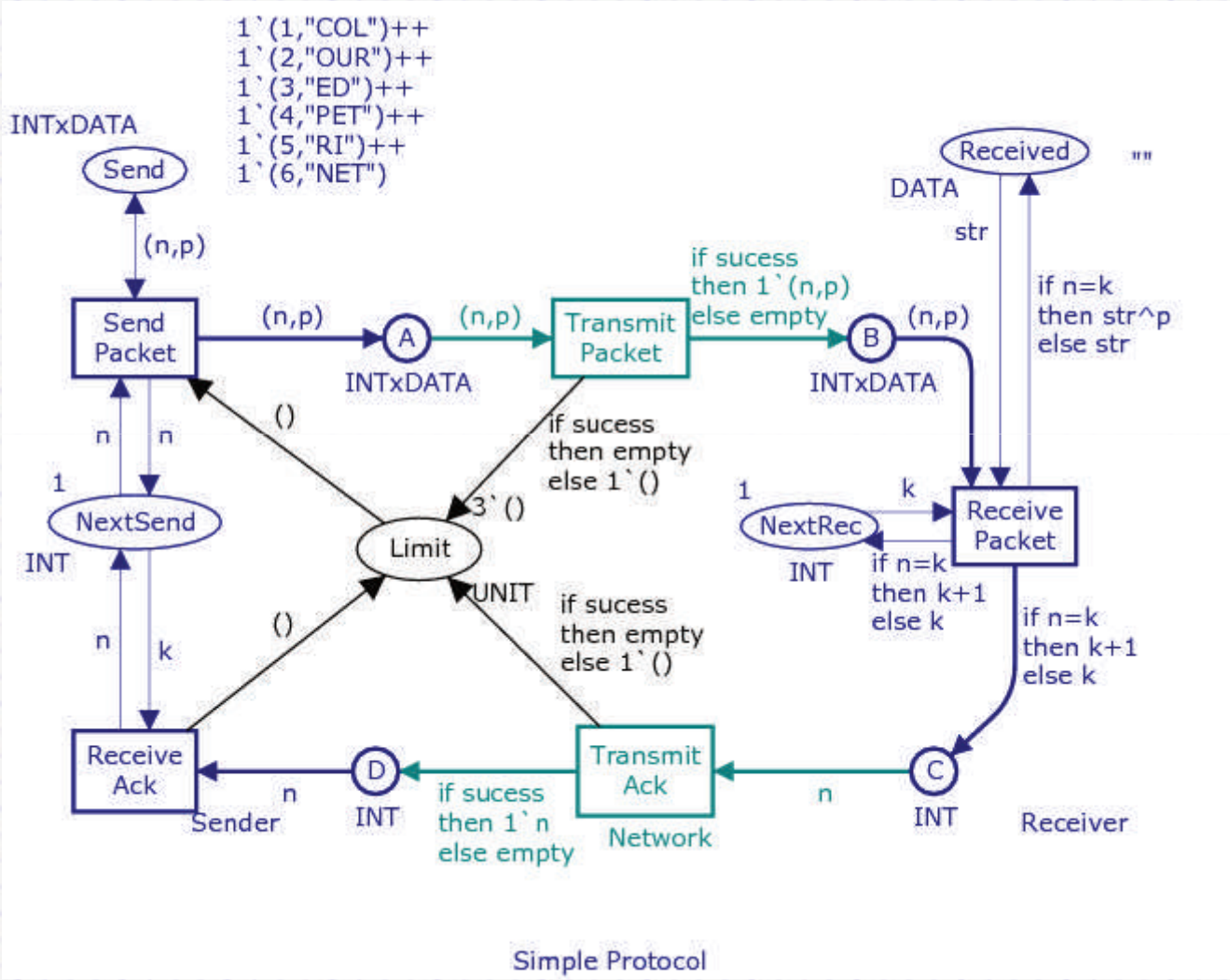
2. ASS = $\{(M,(t,b),M') \in N_{SS} \text{ X BE X } N_{SS} \mid M \rightarrow M'\}$ é o conjunto de arcos

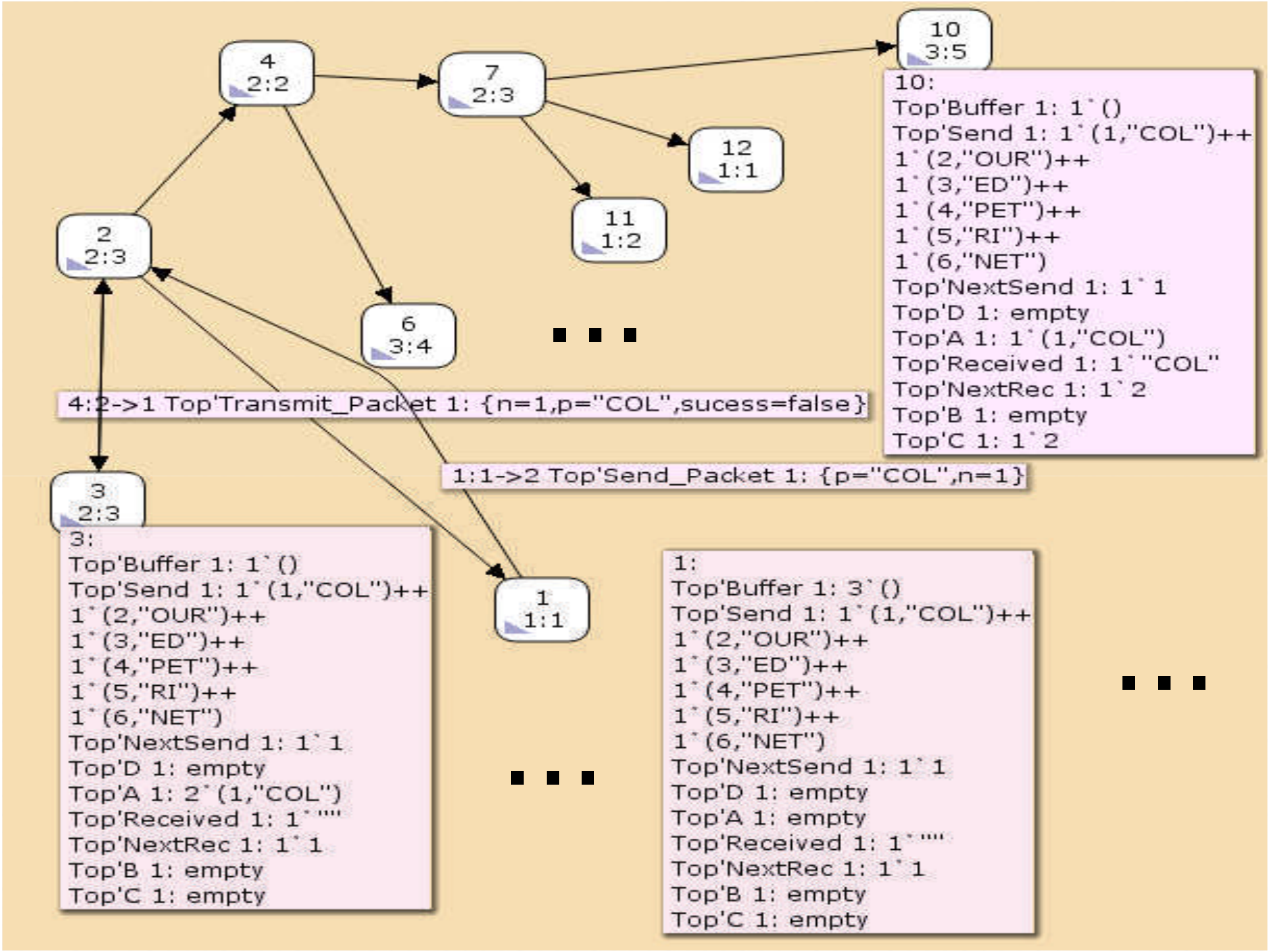
3. SS é finito e somente se N_{SS} e A_{SS} são finitos

Espaço de Estados

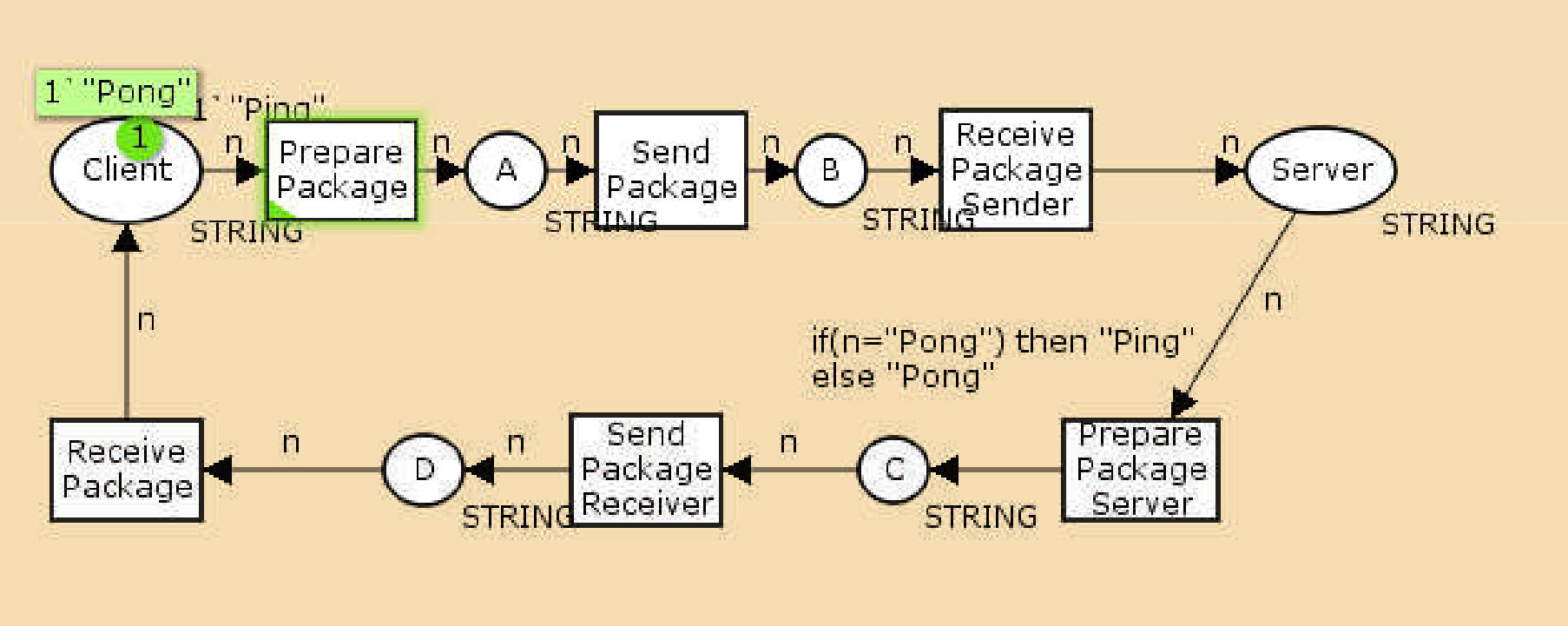
```
NODES ← {M0}
UNPROCESSED ← {M0}
ARCS ← ∅
while UNPROCESSED ≠ ∅ do
    Select a marking M in UNPROCESSED
    UNPROCESSED ← UNPROCESSED −{M}
    for all binding elements (t,b) such that M —(t,b)→ do
        Calculate M' such that M —(t,b)→ M'
        ARCS ← ARCS ∪ {(M,(t,b),M')}
        if M' ∉ NODES then
            NODES ← NODES ∪ {M'}
            UNPROCESSED ← UNPROCESSED ∪ {M'}
        end if
    end for
end while
```

Espaço de Estados

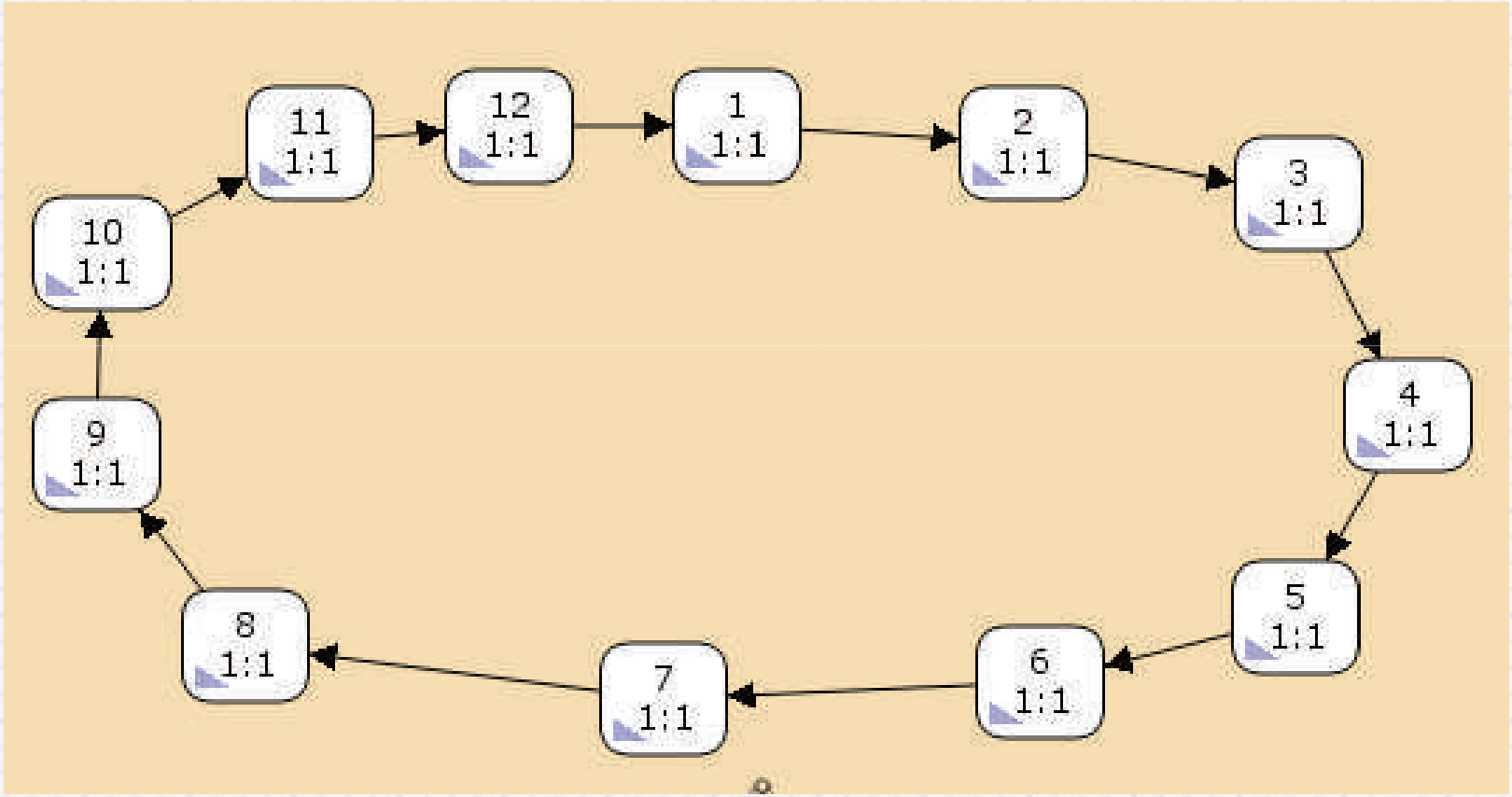



4
2:2
7
2:3
10
3:5
12
1:1
11
1:2
2
2:3
6
3:4
10:
Top'Buffer 1: 1`()
Top'Send 1: 1`(1,"COL")++
1`(2,"OUR")++
1`(3,"ED")++
1`(4,"PET")++
1`(5,"RI")++
1`(6,"NET")
Top'NextSend 1: 1`1
Top'D 1: empty
Top'A 1: 1`(1,"COL")
Top'Received 1: 1`"COL"
Top'NextRec 1: 1`2
Top'B 1: empty
Top'C 1: 1`2
4:2->1 Top'Transmit_Packet 1: {n=1,p="COL",sucess=false}
1:1->2 Top'Send_Packet 1: {p="COL",n=1}
3
2:3
1
1:1
3:
Top'Buffer 1: 1`()
Top'Send 1: 1`(1,"COL")++
1`(2,"OUR")++
1`(3,"ED")++
1`(4,"PET")++
1`(5,"RI")++
1`(6,"NET")
Top'NextSend 1: 1`1
Top'D 1: empty
Top'A 1: 2`(1,"COL")
Top'Received 1: 1`""
Top'NextRec 1: 1`1
Top'B 1: empty
Top'C 1: empty
1:
Top'Buffer 1: 3`()
Top'Send 1: 1`(1,"COL")++
1`(2,"OUR")++
1`(3,"ED")++
1`(4,"PET")++
1`(5,"RI")++
1`(6,"NET")
Top'NextSend 1: 1`1
Top'D 1: empty
Top'A 1: empty
Top'Received 1: 1`""
Top'NextRec 1: 1`1
Top'B 1: empty
Top'C 1: empty

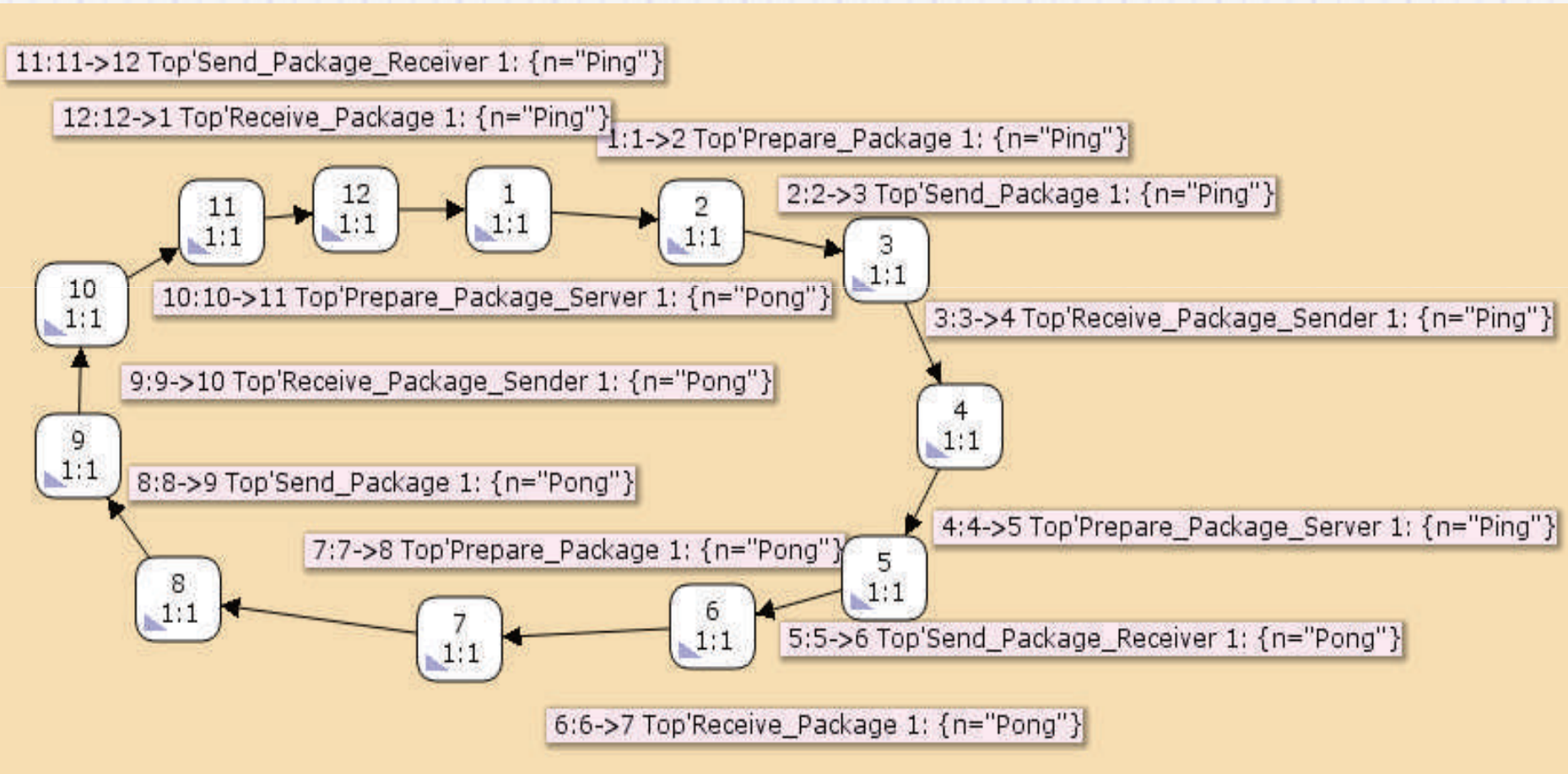
Espaço de Estados



Espaço de Estados



Espaço de Estados



Invariantes

Conceito similar em PTN, aplica-se para CPN

Definition 7.1: Let $A \in \Sigma$ be a type and let $W = \{W_p\}_{p \in P}$ be a set of linear functions such that $W_p \in [C(p)_{WS} \rightarrow A_{WS}]$ for all $p \in P$.

(i) W is a **place flow** iff:

$$\forall (t,b) \in BE: \sum_{p \in P} W_p(E(p,t)<b>) = \sum_{p \in P} W_p(E(t,p)<b>).$$

(ii) W determines a **place invariant** iff:

$$\forall M \in [M_0\rangle: \sum_{p \in P} W_p(M(p)) = \sum_{p \in P} W_p(M_0(p)).$$

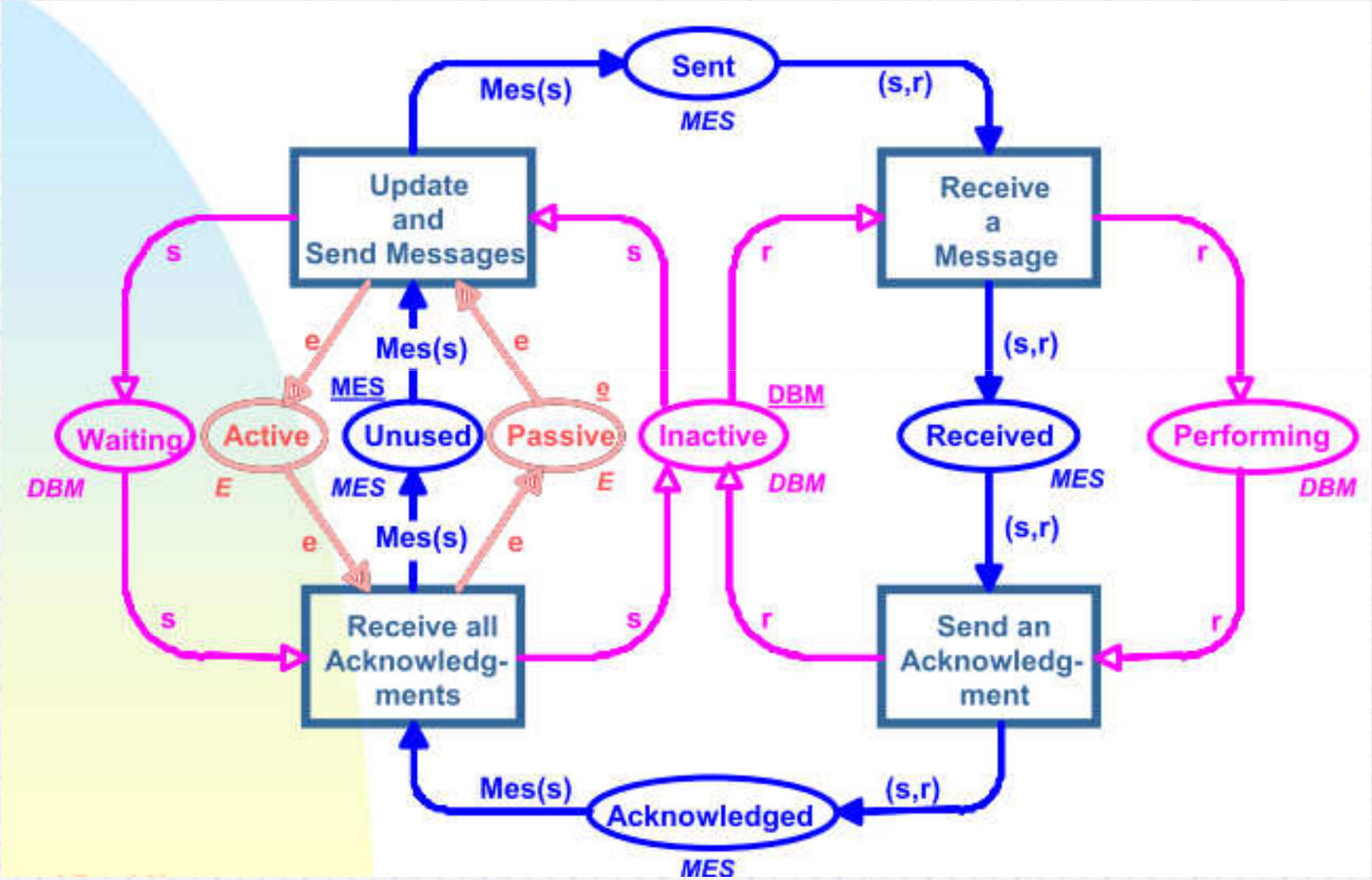
Theorem 7.2: W is a place flow $\Leftrightarrow$ W determines a place invariant.

$\Rightarrow$ is satisfied for all CP-nets.

$\Leftarrow$ is only satisfied when the CP-net does not have dead binding elements.

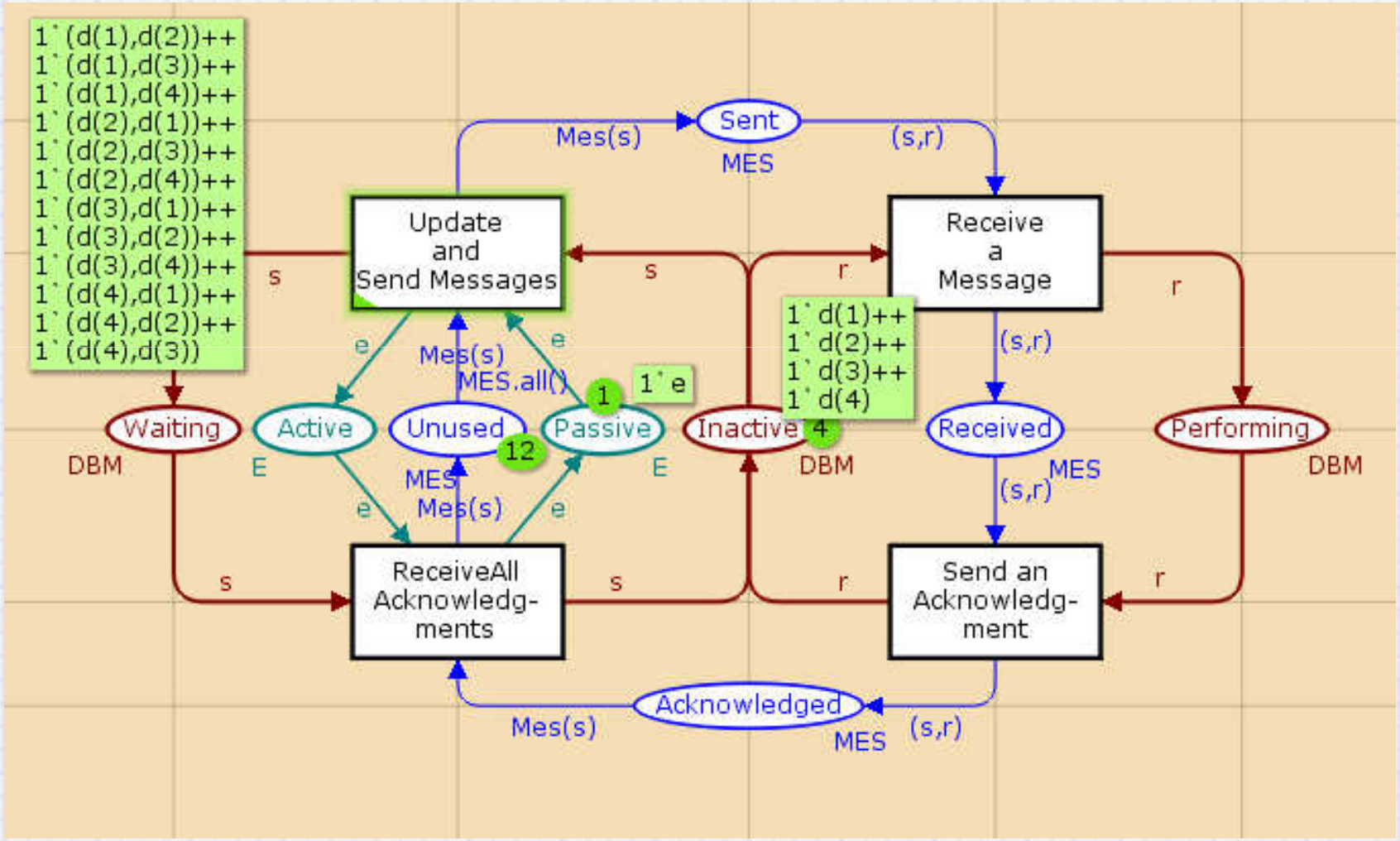
$$W * I = 0$$

Invariantes



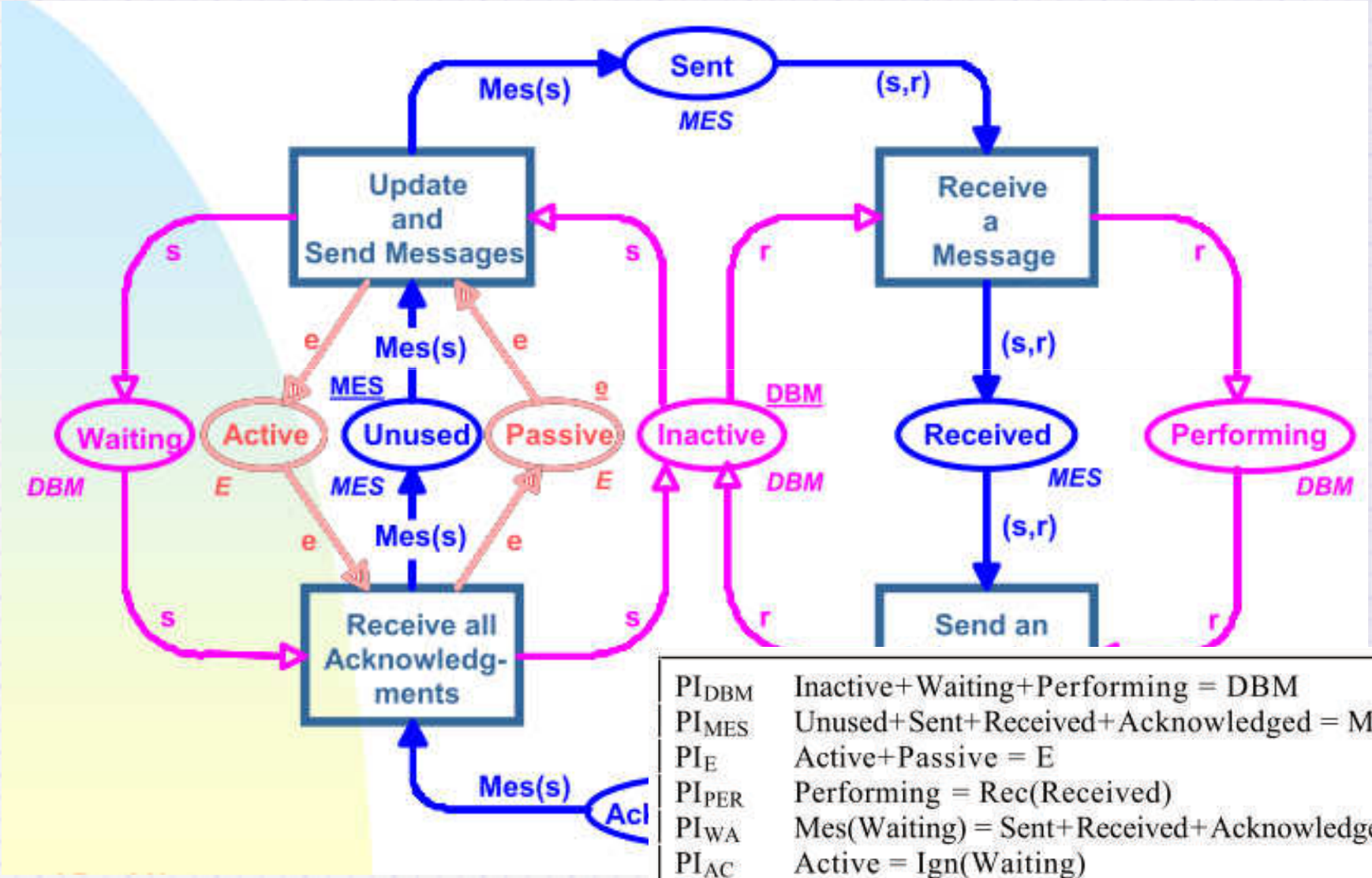
Baseado em ACPN-2010 por Lars Kristensen

Invariantes



Baseado em ACPN-2010 por Lars Kristensen

Invariantes



PI_{DBM}	Inactive+Waiting+Performing = DBM
PI_{MES}	Unused+Sent+Received+Acknowledged = MES
PI_{E}	Active+Passive = E
PI_{PER}	Performing = Rec(Received)
PI_{WA}	Mes(Waiting) = Sent+Received+Acknowledged
PI_{AC}	Active = Ign(Waiting)

Baseado em ACPN 2010 por Lars Kristensen

Invariantes

Invariantes de transições obtidos de forma similar

Pesos associados a transições

Transition flow: conjunto de ocorrências que não tem efeito total (marcação inicial e final são as mesmas)

A ferramenta mais representativa não tem suporte a invariantes

Redes de Petri Coloridas Hierárquicas (CPN)

Boas práticas na construção de programas: Módulos

- Reusabilidade
- Mais abstração
- Legibilidade
- Diminiuição de erros
- ...

Construção de uma CPN através de submodelos

Submodelos podem ser parametrizados e podem referenciar outros submodelos

Adoção de uma transição que representa a chamada ao submodelo

Redes de Petri Coloridas Hierárquicas (CPN)

Submodelo possui interface via lugares

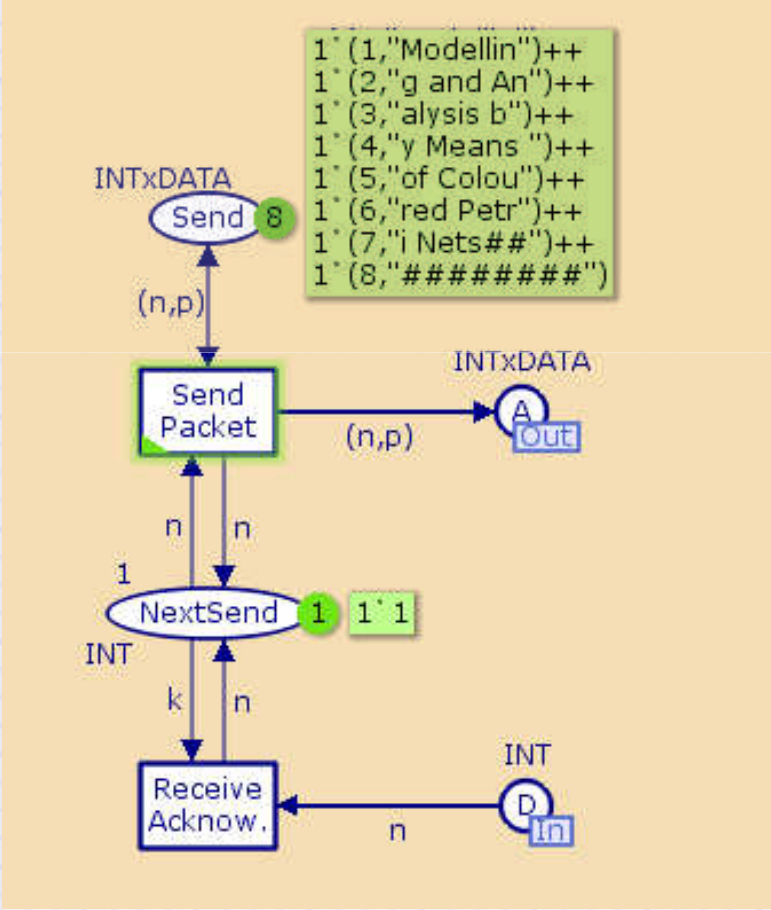
- Lugares de entrada (IN)
- Lugares de saída (OUT)
- Lugares de entrada/saída (I/O)

Ex: CPN com a rede não confiável

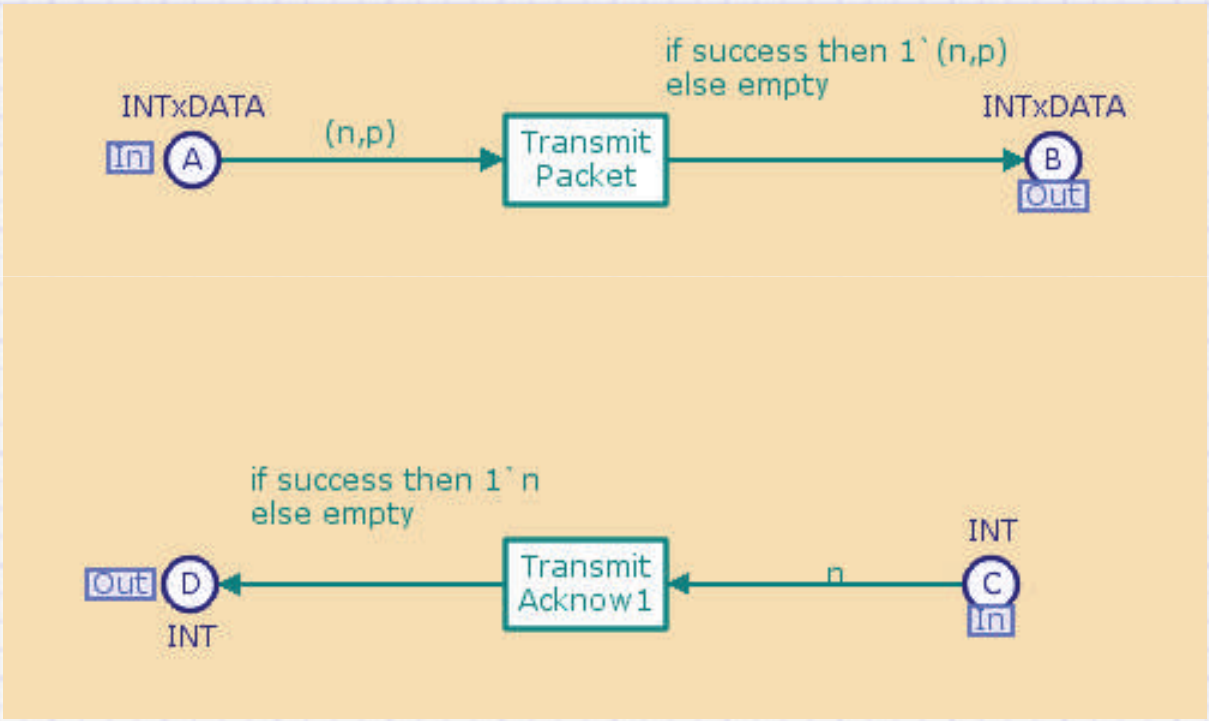
Considerar 3 submodelos:

- Transmissor
- Rede
- Receptor

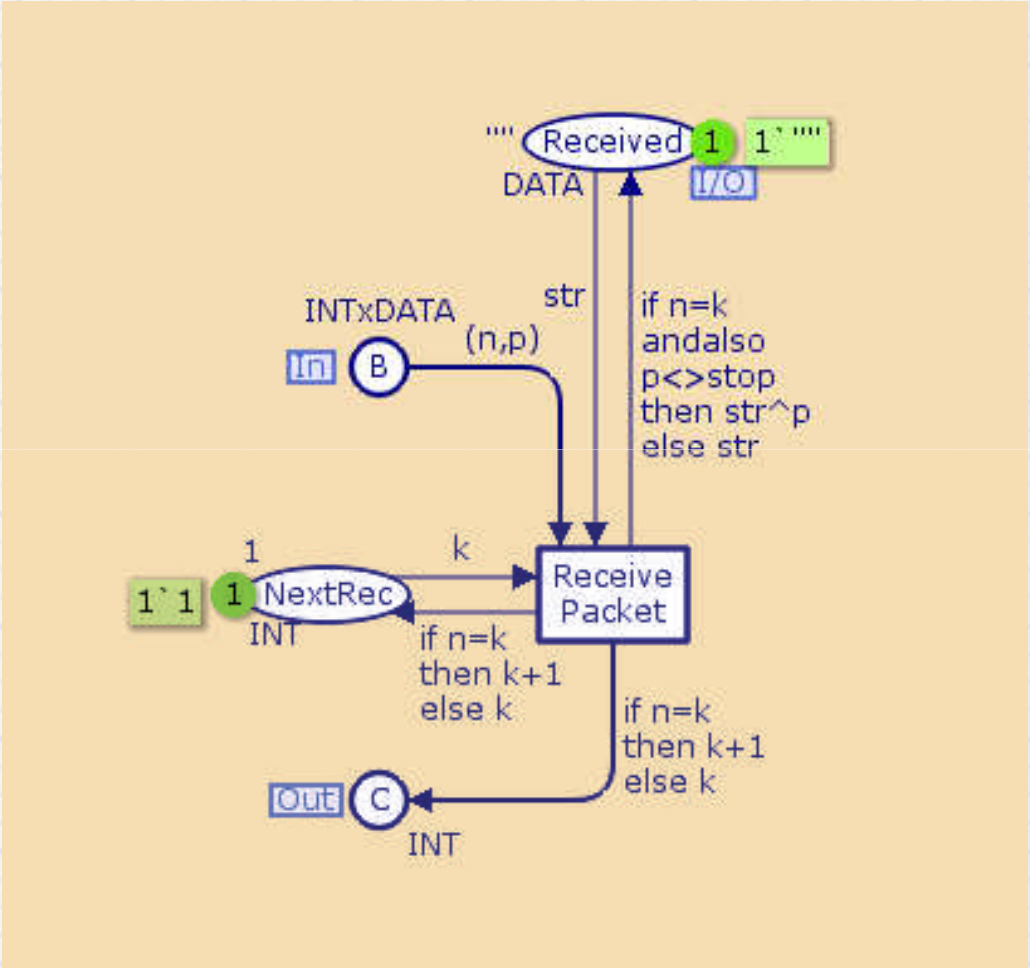
Redes de Petri Coloridas Hierárquicas (CPN)



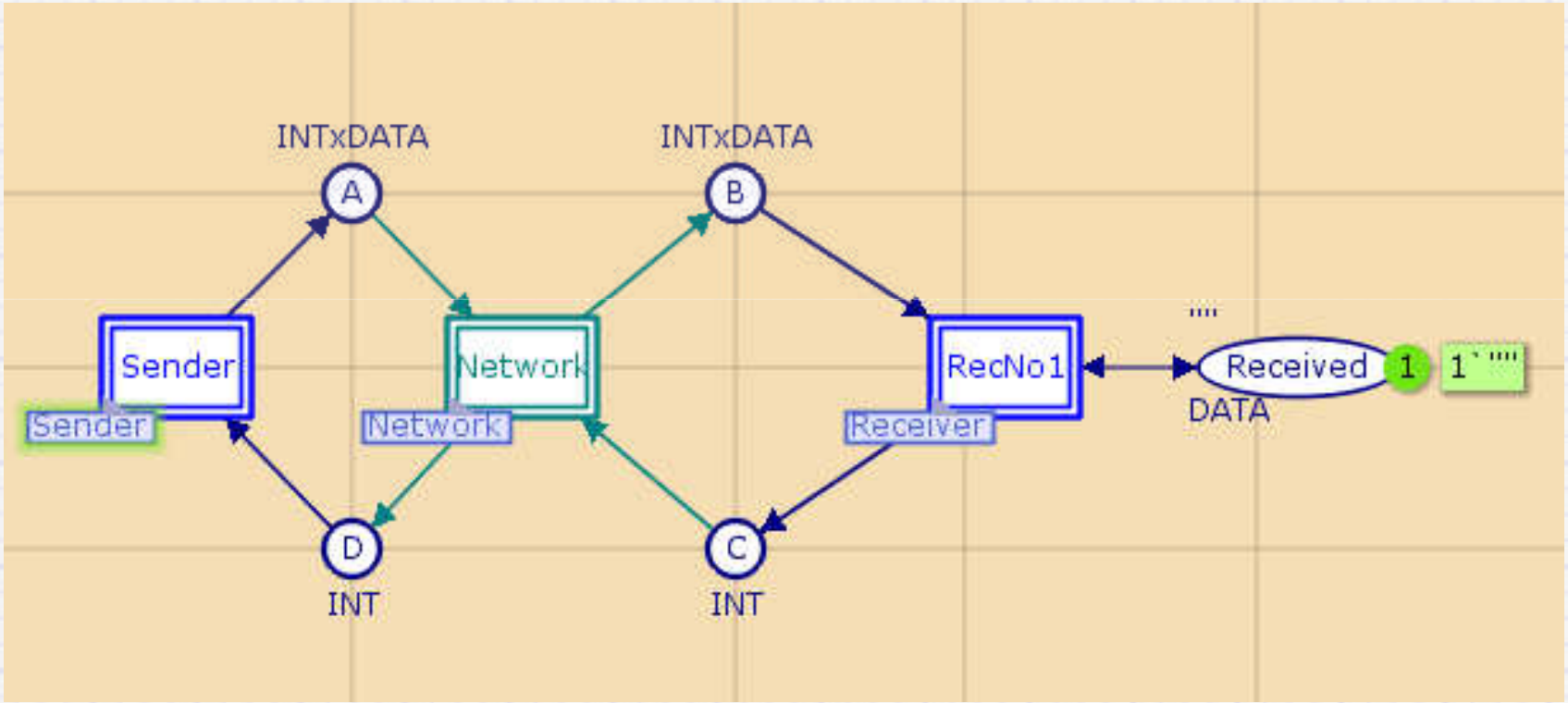
Redes de Petri Coloridas Hierárquicas (CPN)



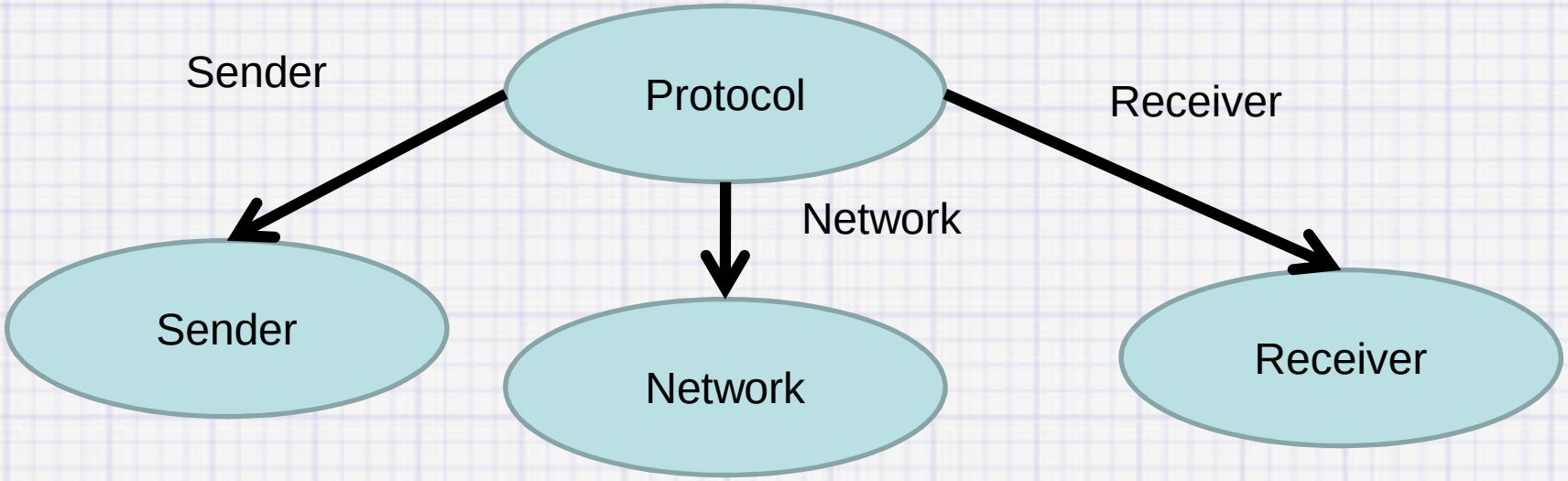
Redes de Petri Coloridas Hierárquicas (CPN)



Redes de Petri Coloridas Hierárquicas (CPN)



Redes de Petri Coloridas Hierárquicas (CPN)



Redes de Petri Coloridas Hierárquicas (CPN)

Fusion sets

- Submodelos compartilhando um mesmo lugar
- O lugar é o mesmo para as instâncias
- Similar ao conceito de variáveis globais
- Adoção precisa ser cuidadosa

Redes de Petri Coloridas Hierárquicas (CPN)

Definition 6.1. A **Coloured Petri Net Module** is a four-tuple $CPN_M = (CPN, T_{\text{sub}}, P_{\text{port}}, PT)$, where:

1. $CPN = (P, T, A, \Sigma, V, C, G, E, I)$ is a **non-hierarchical Coloured Petri Net.**
2. $T_{\text{sub}} \subseteq T$ is a set of **substitution transitions.**
3. $P_{\text{port}} \subseteq P$ is a set of **port places.**
4. $PT : P_{\text{port}} \rightarrow \{\text{IN}, \text{OUT}, \text{I/O}\}$ is a **port type function** that assigns a port type to each port place.

□

Redes de Petri Coloridas Hierárquicas (CPN)

Definition 6.2. A **hierarchical Coloured Petri Net** is a four-tuple $CPN_H = (S, SM, PS, FS)$ where:

1. S is a finite set of **modules**. Each module is a **Coloured Petri Net Module** $s = ((P^s, T^s, A^s, \Sigma^s, V^s, C^s, G^s, E^s, I^s), T^s_{\text{sub}}, P^s_{\text{port}}, PT^s)$. It is required that $(P^{s_1} \cup T^{s_1}) \cap (P^{s_2} \cup T^{s_2}) = \emptyset$ for all $s_1, s_2 \in S$ such that $s_1 \neq s_2$.
2. $SM : T_{\text{sub}} \rightarrow S$ is a **submodule function** that assigns a **submodule** to each substitution transition. It is required that the module hierarchy (see Definition 6.3) is acyclic.
3. PS is a **port–socket relation function** that assigns a **port–socket relation** $PS(t) \subseteq P_{\text{sock}}(t) \times P^{SM(t)}_{\text{port}}$ to each substitution transition t. It is required that $ST(p) = PT(p')$, $C(p) = C(p')$, and $I(p)\langle\rangle = I(p')\langle\rangle$ for all $(p, p') \in PS(t)$ and all $t \in T_{\text{sub}}$.
4. $FS \subseteq 2^P$ is a set of non-empty **fusion sets** such that $C(p) = C(p')$ and $I(p)\langle\rangle = I(p')\langle\rangle$ for all $p, p' \in fs$ and all $fs \in FS$.

□

socket places $P_{\text{sock}}(t)$ of a substitution transition $PT : P_{\text{port}} \rightarrow \{\text{IN}, \text{OUT}, \text{I/O}\}$ is a **port type function** *socket type function* $ST(t)$ that maps each socket place of t into its type

Redes de Petri Coloridas Hierárquicas (CPN)

Uma CPN hierárquica pode ser sempre transformada em um CPN não hierárquica

Na teoria, não adiciona maior poder de expressividade ao modelo

CPN hierárquica e não hierárquica são equivalentes

Da mesma forma CPN $\leftrightarrow$ PTN

- Lugares em CPN substituído por Lugares em PTN com a quantidade de cores no colour set
- Transições em PTN com as possibilidades de bindings satisfazendo as guardas das transições em CPN

Redes de Petri Coloridas Temporizadas (TCPN)

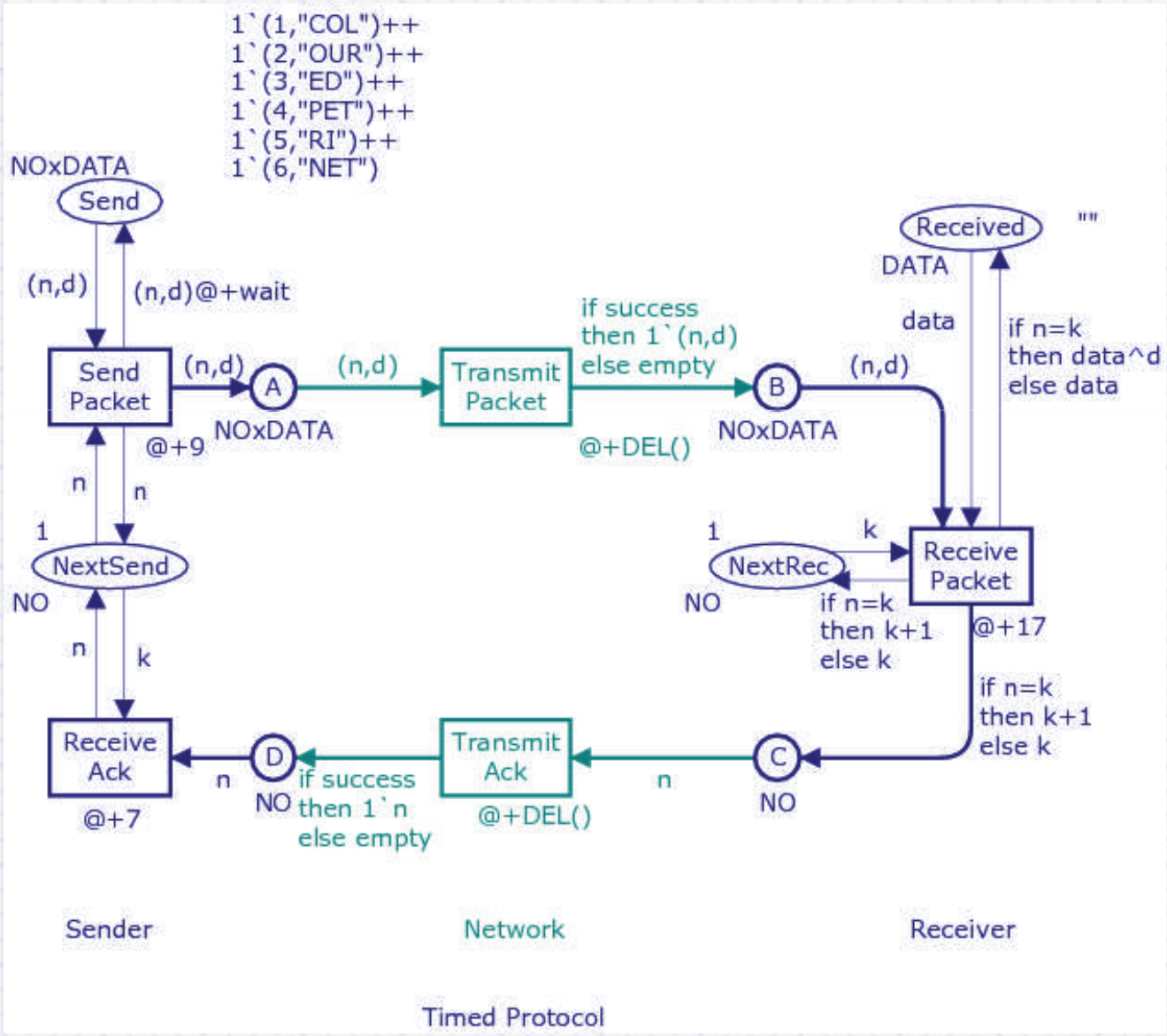
Importância do tempo para avaliação de desempenho/dependabilidade de sistemas

Modelagem de sistemas de tempo real

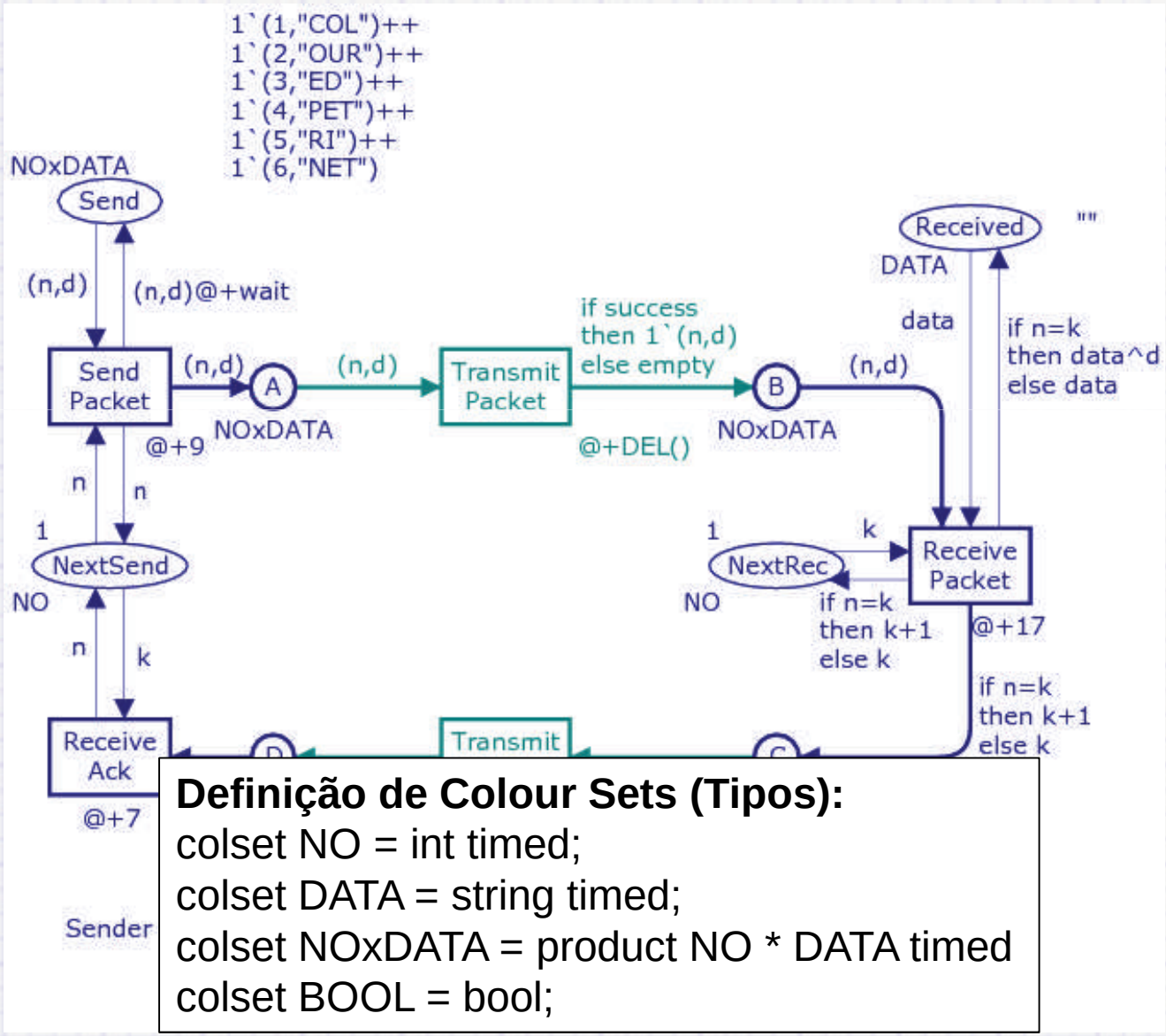
Conceitos

- Tempos associados às marcações(tokens)
- Adoção de um relógio global (global clock)
- O *timestamp* associado à marcação indica quando a mesma está pronta para uso
- O disparo de uma transição é imediato
- *Timed multiset* e *timed colour sets*
- *Timestamp* $\in$ TIME (conjunto dos inteiros não negativos)

Exemplo


1`(1,"COL")++
1`(2,"OUR")++
1`(3,"ED")++
1`(4,"PET")++
1`(5,"RI")++
1`(6,"NET")
NOxDATA
Send
(n,d)
(n,d)@+wait
Send Packet
(n,d)
A
NOxDATA
@+9
(n,d)
Transmit Packet
if success
then 1`(n,d)
else empty
@+DEL()
B
NOxDATA
(n,d)
Received
DATA
""
data
if n=k
then data^d
else data
n
n
1
NextSend
NO
n
k
1
k
NextRec
NO
if n=k
then k+1
else k
Receive Packet
@+17
if n=k
then k+1
else k
Receive Ack
@+7
n
D
NO
if success
then 1`n
else empty
Transmit Ack
@+DEL()
n
C
NO
Sender
Network
Receiver
Timed Protocol

Exemplo



Definição de Colour Sets (Tipos):

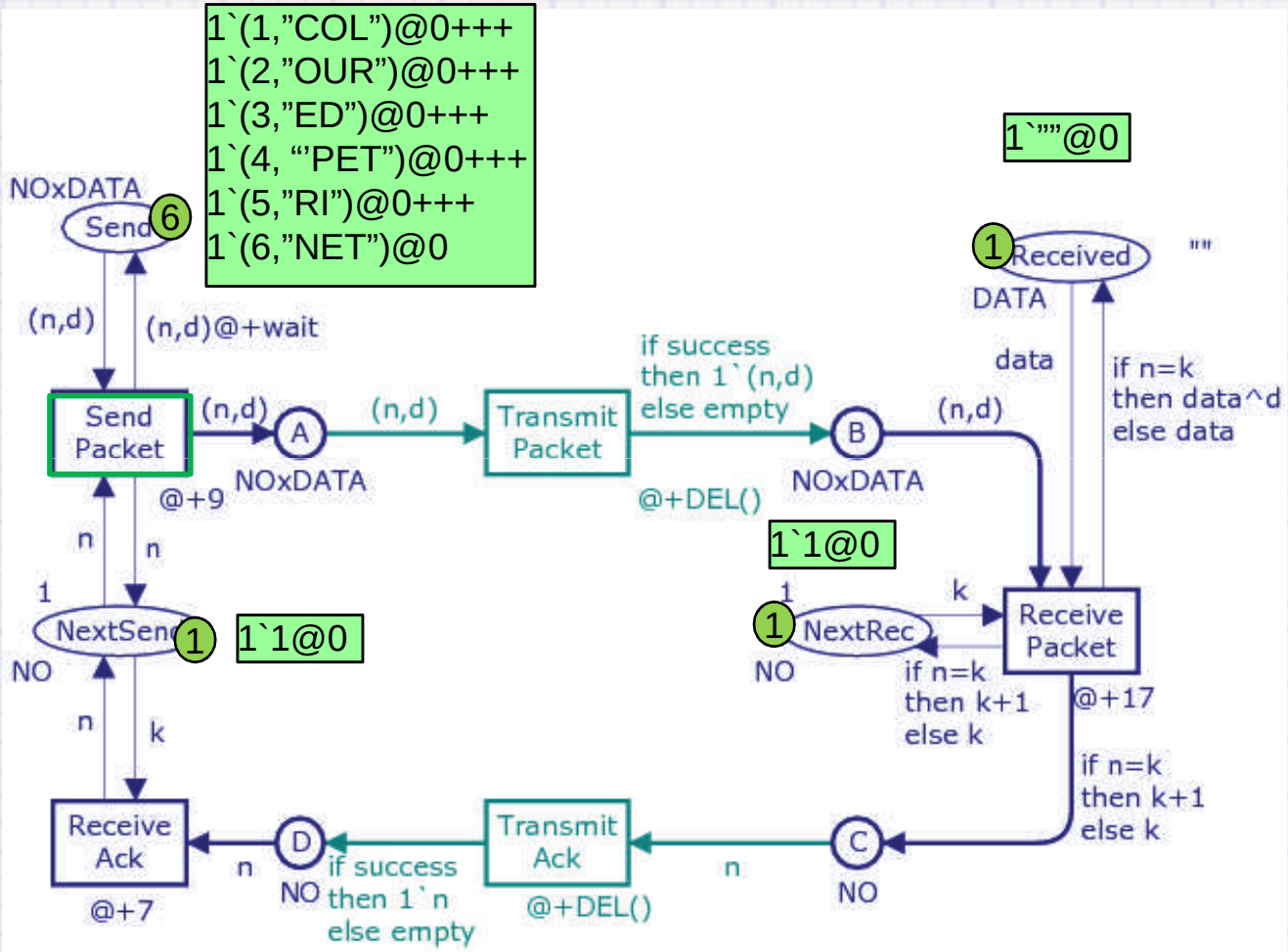
colset NO = int timed;

colset DATA = string timed;

colset NOxDATA = product NO * DATA timed

colset BOOL = bool;

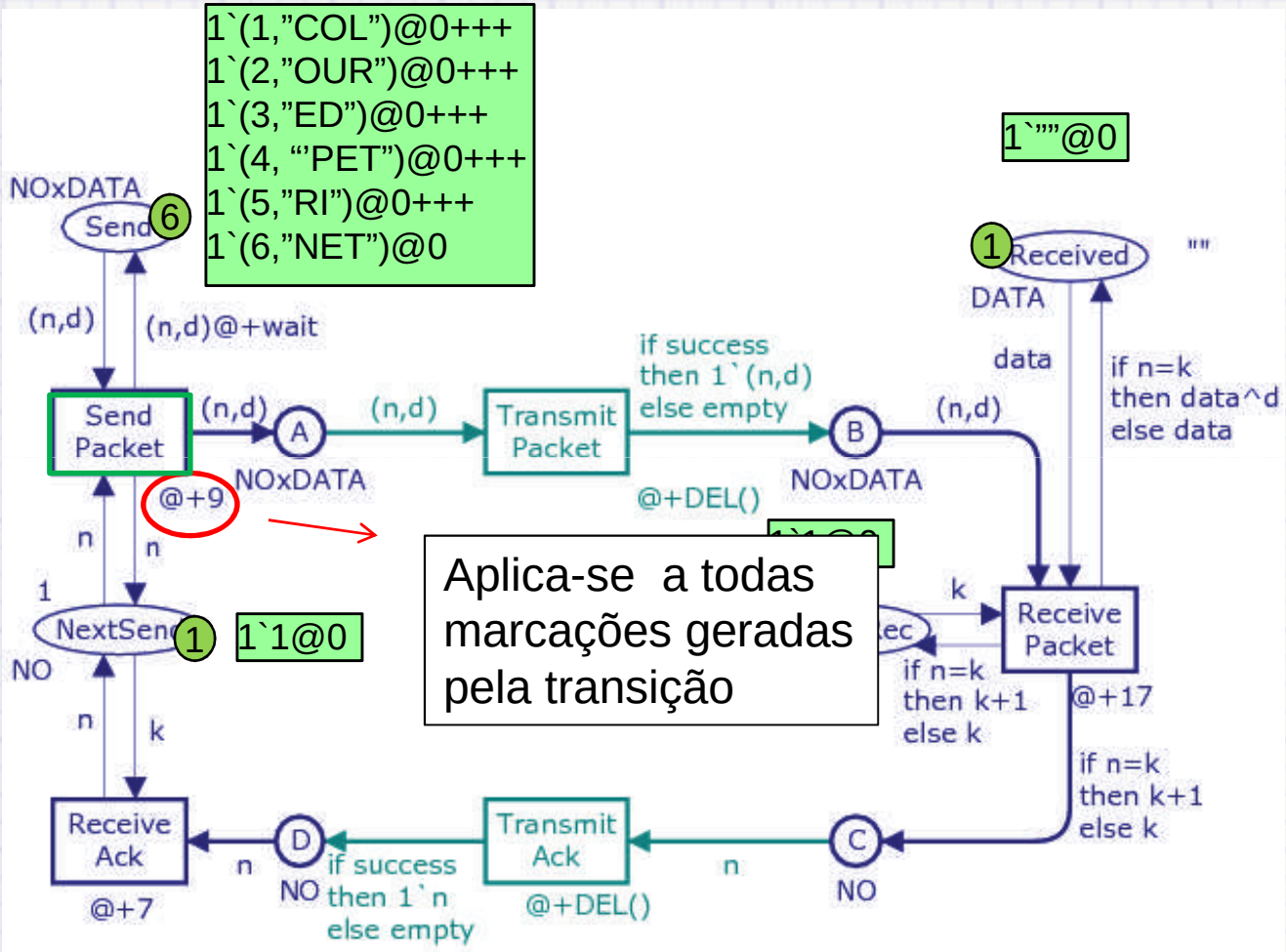
Exemplo



Binding Element
(SendPacket,<n=1,d="col">)
Pode ocorrer no tempo 0

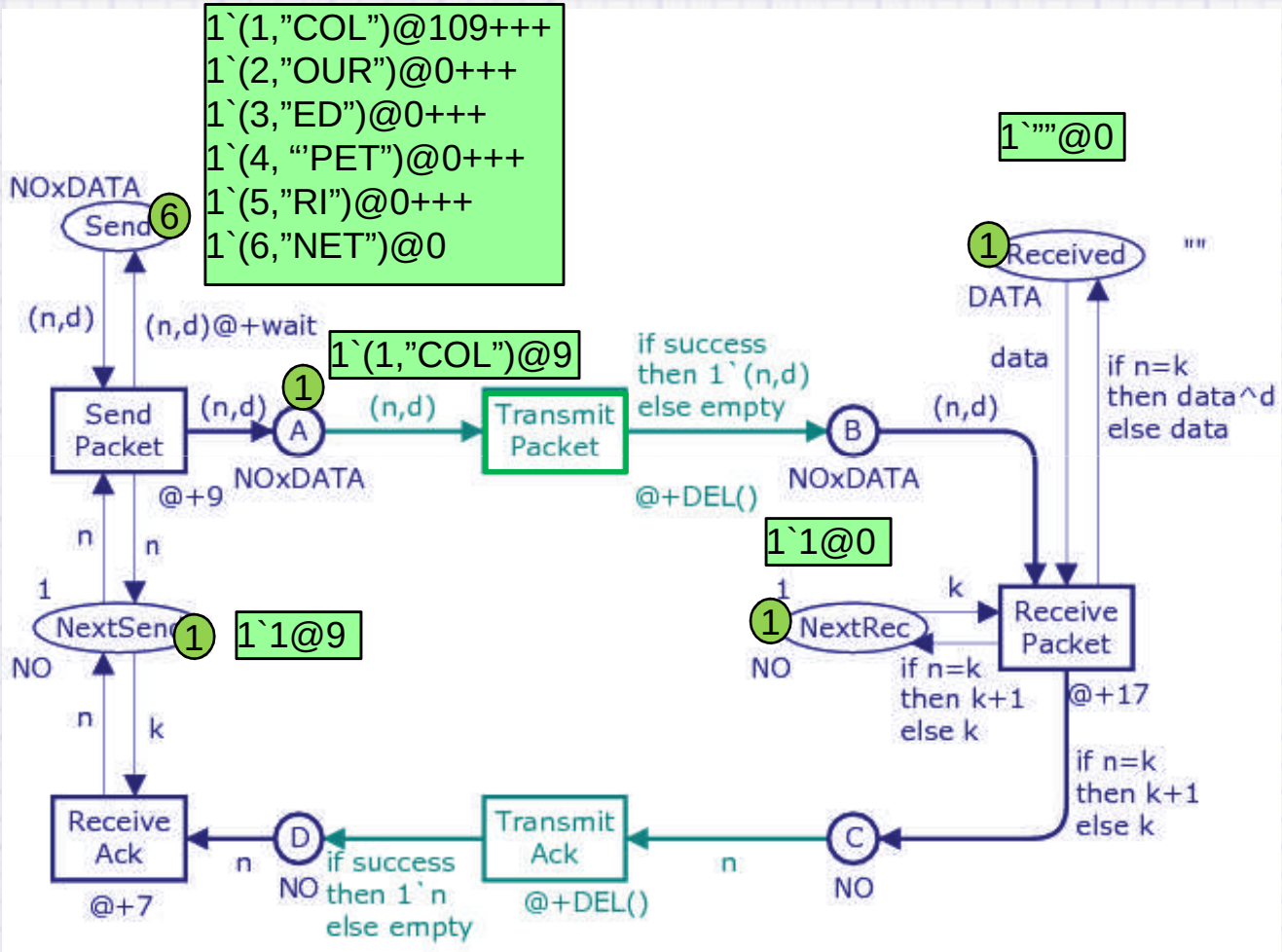
Global Clock=0

Exemplo



Global Clock=0

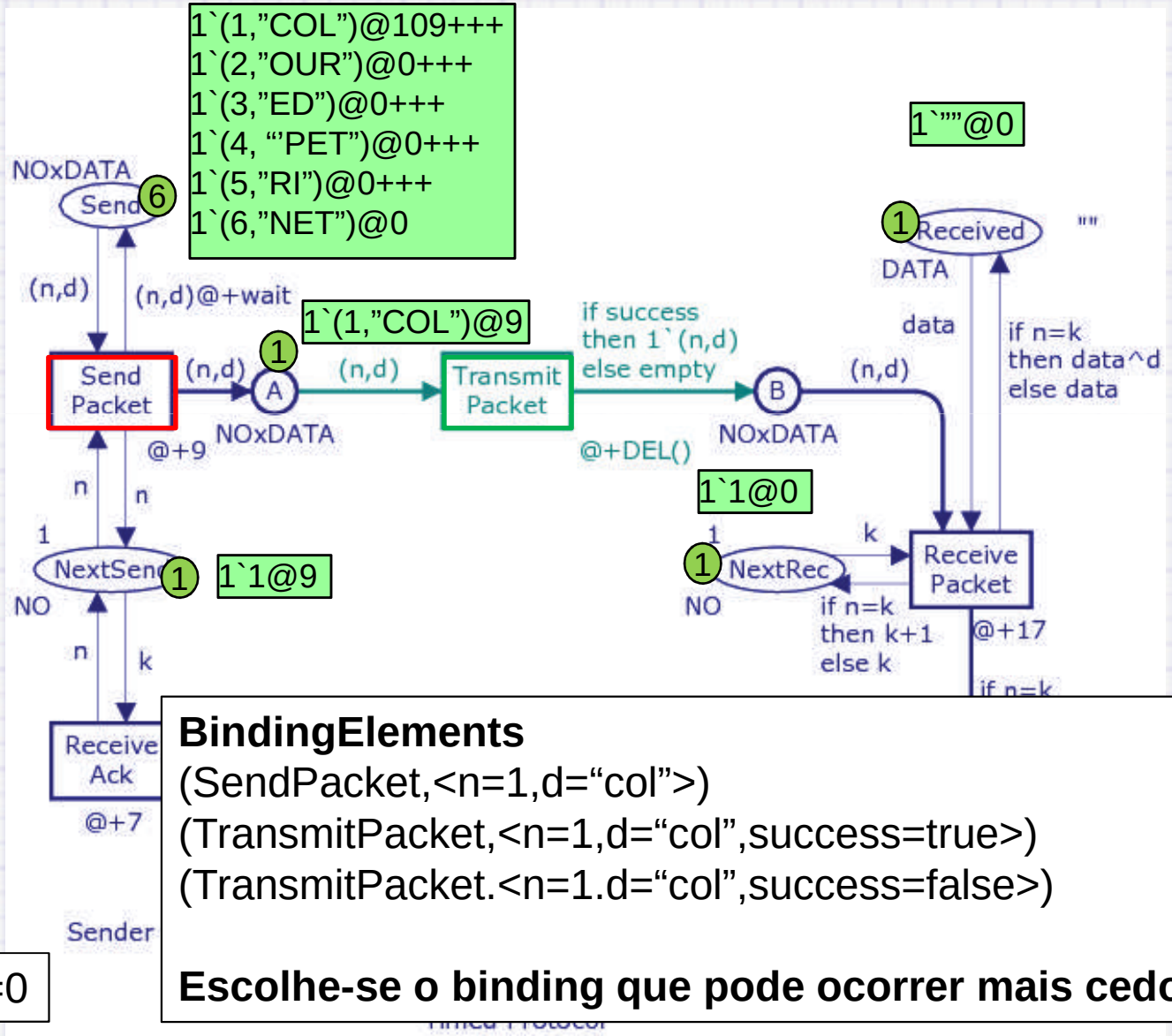
Exemplo



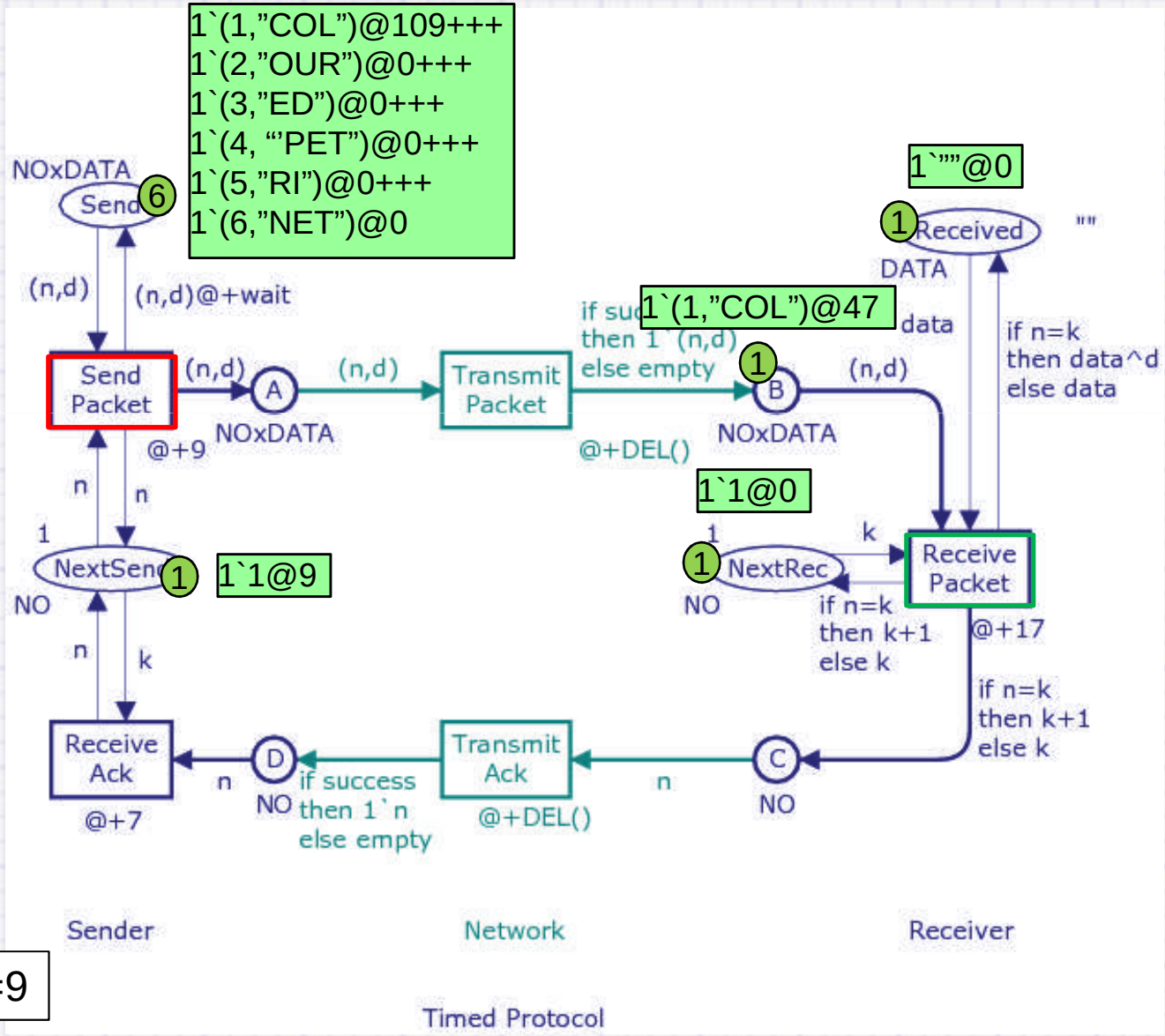
Global Clock=0

Timestamp das marcações geradas
Global Clock + Inscrição da Transição + Inscrição do arco de saída

Exemplo

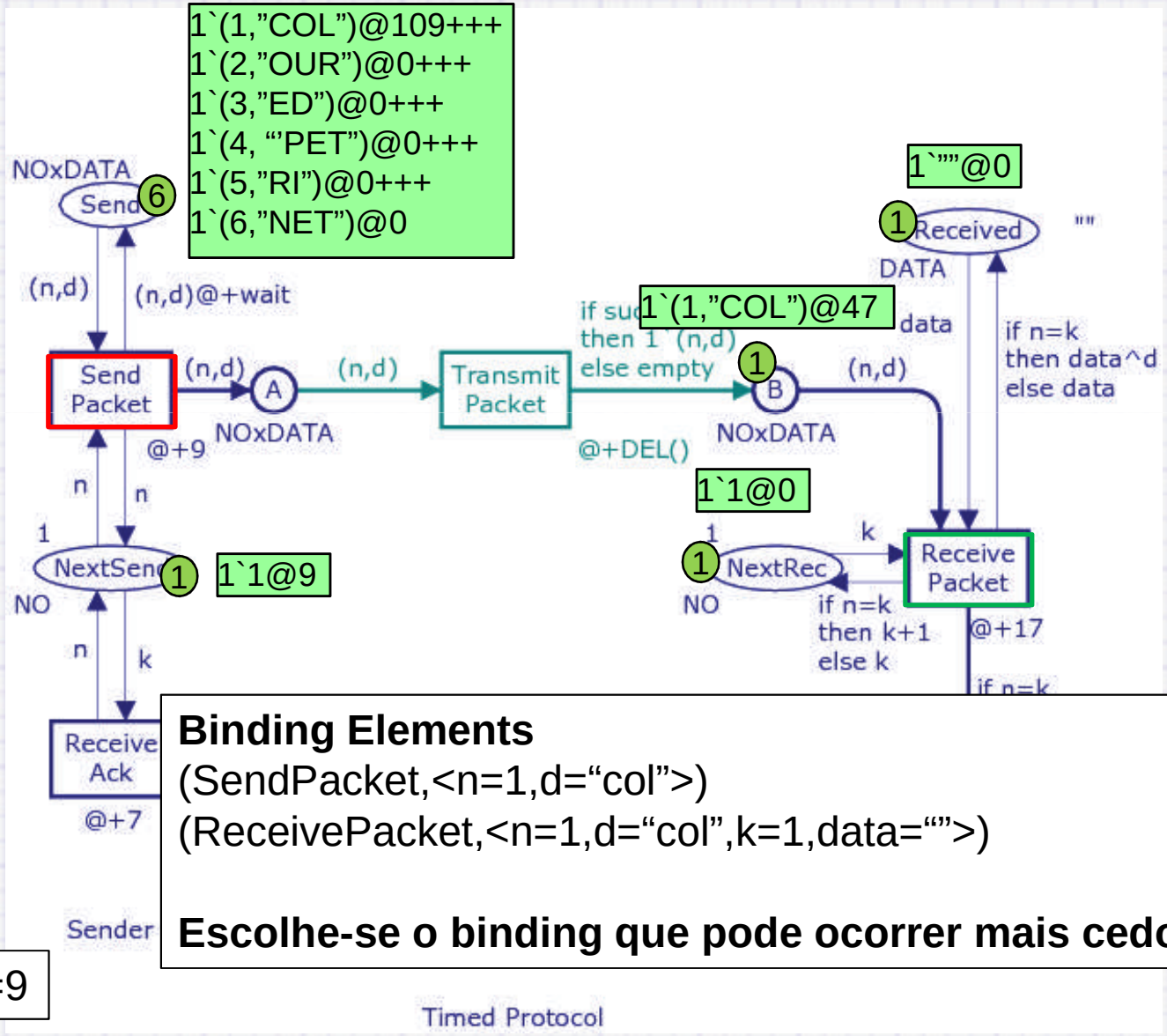

1`(1,"COL")@109+++
1`(2,"OUR")@0+++
1`(3,"ED")@0+++
1`(4, "'PET")@0+++
1`(5,"RI")@0+++
1`(6,"NET")@0
1`""@0
NOxDATA
Send
6
1
Received
""
DATA
(n,d)
(n,d)@+wait
1`(1,"COL")@9
if success
then 1`(n,d)
else empty
data
if n=k
then data^d
else data
Send
Packet
(n,d)
1
A
(n,d)
Transmit
Packet
B
(n,d)
NOxDATA
@+9
NOxDATA
@+DEL()
1`1@0
n
n
1
k
1
NextSen
1
1`1@9
1
NextRec
Receive
Packet
NO
NO
if n=k
then k+1
else k
@+17
n
k
Receive
Ack
@+7
Sender
BindingElements
(SendPacket,<n=1,d="col">)
(TransmitPacket,<n=1,d="col",success=true>)
(TransmitPacket.<n=1.d="col",success=false>)
Escolhe-se o binding que pode ocorrer mais cedo
Global Clock=0

Exemplo



Global Clock=9

Exemplo



Binding Elements

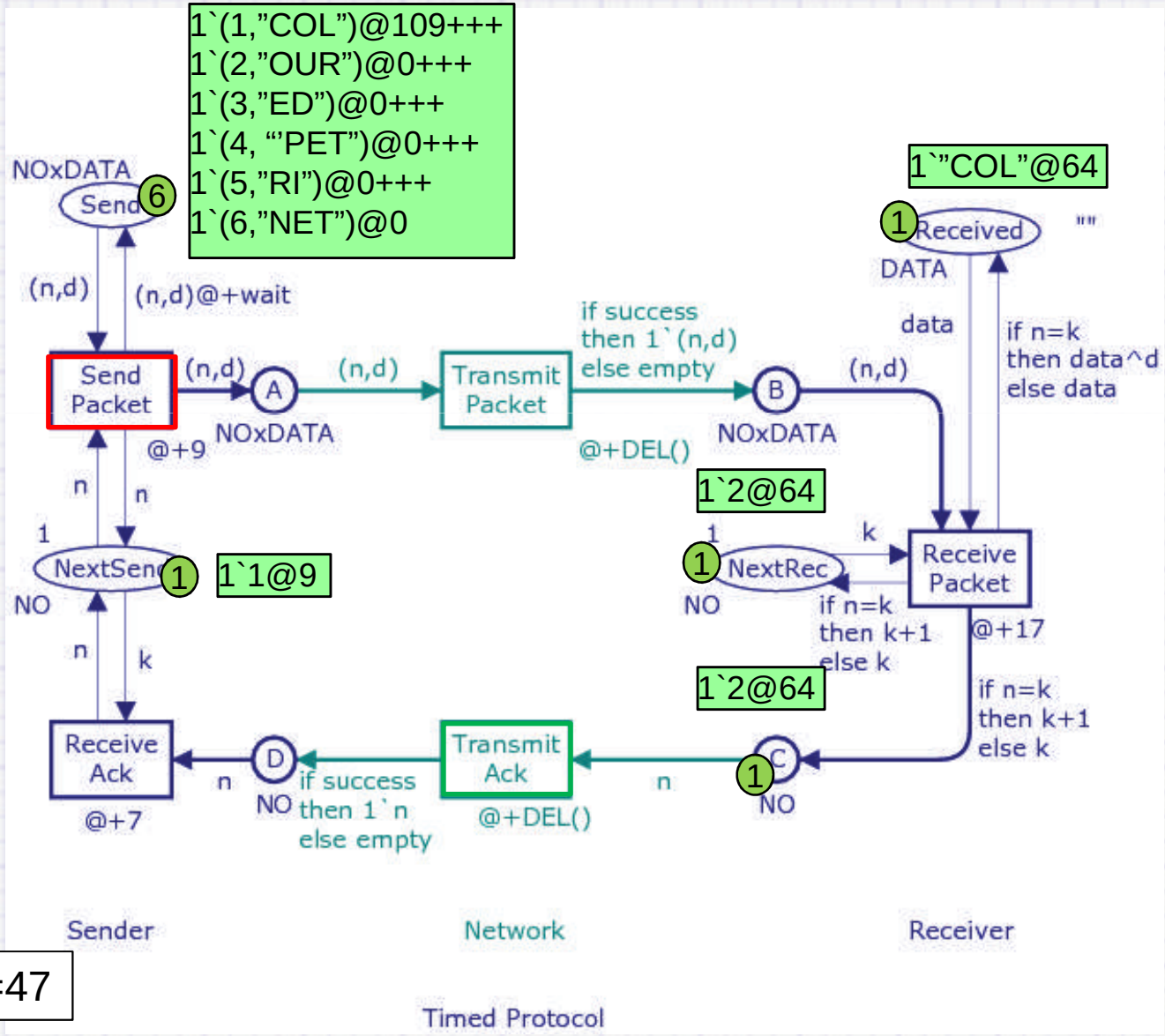
(SendPacket,<n=1,d="col">)

(ReceivePacket,<n=1,d="col",k=1,data="">)

Escolhe-se o binding que pode ocorrer mais cedo

Global Clock=9

Exemplo

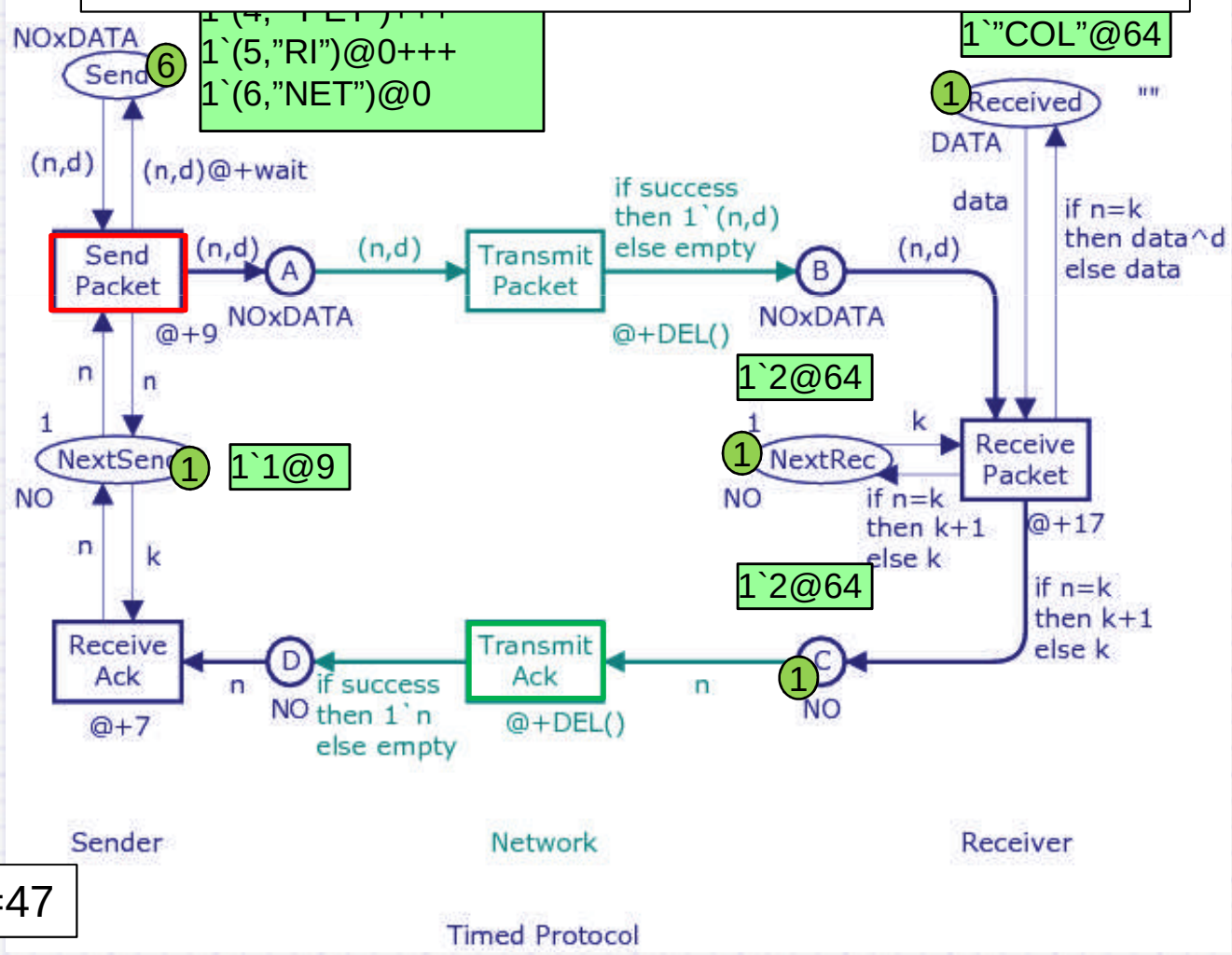


Global Clock=47

Exemplo

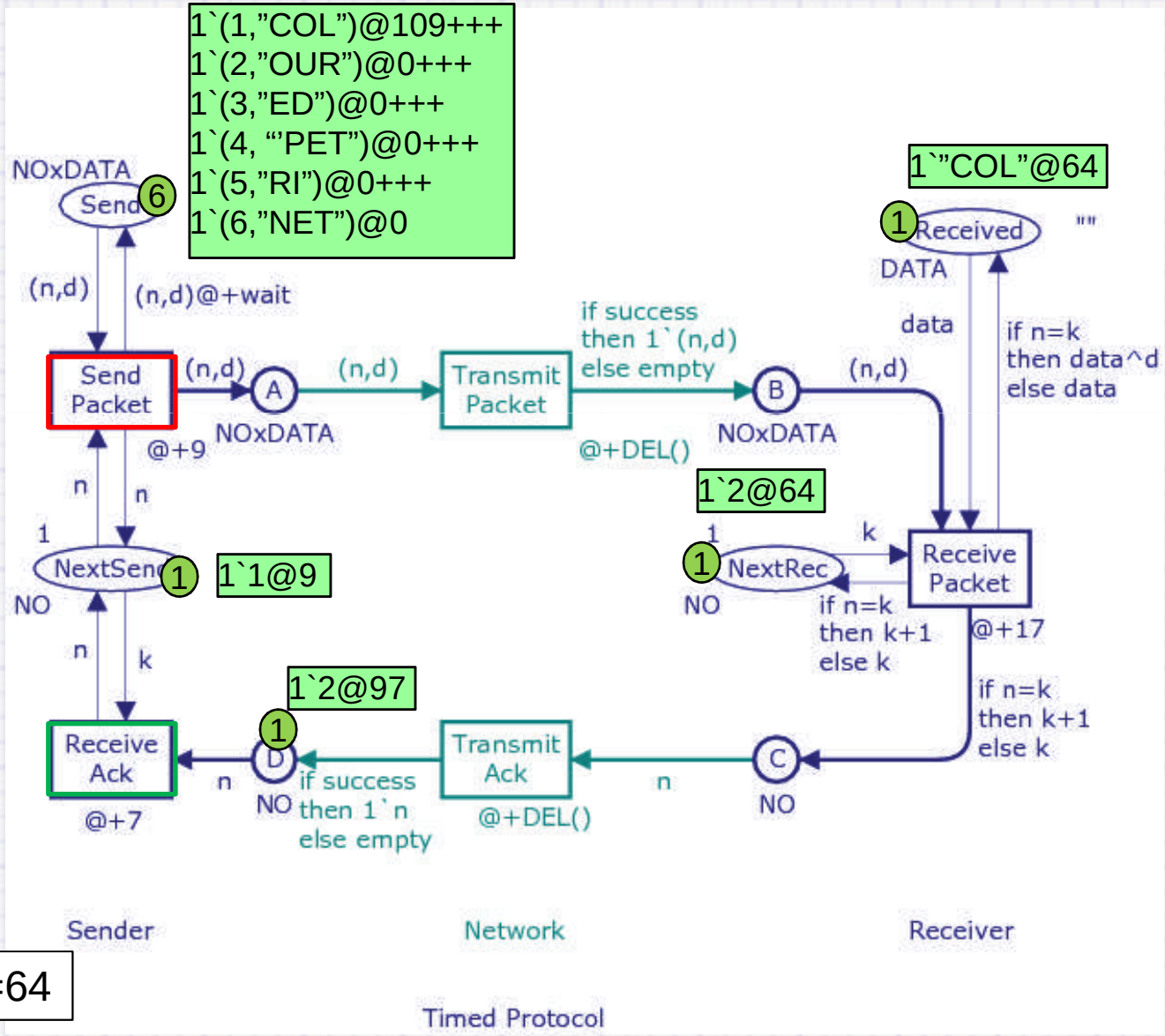
Binding Elements
(SendPacket,<n=1,d="col">)
(TransmitAck,<n=2,success=true>)
(TransmitAck.<n=2.success=false>)

Escolhe-se o binding que pode ocorrer mais cedo



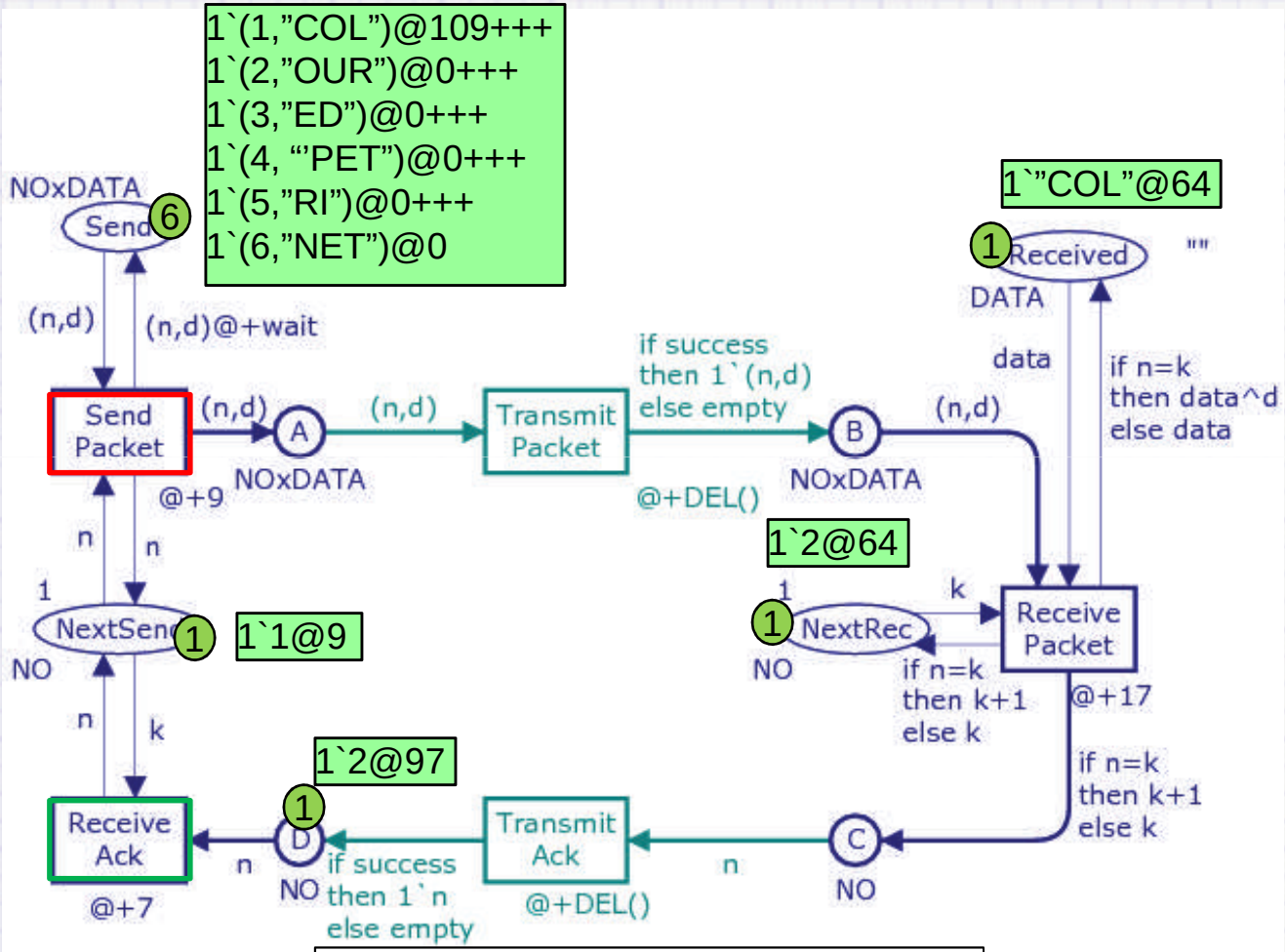
Global Clock=47

Exemplo

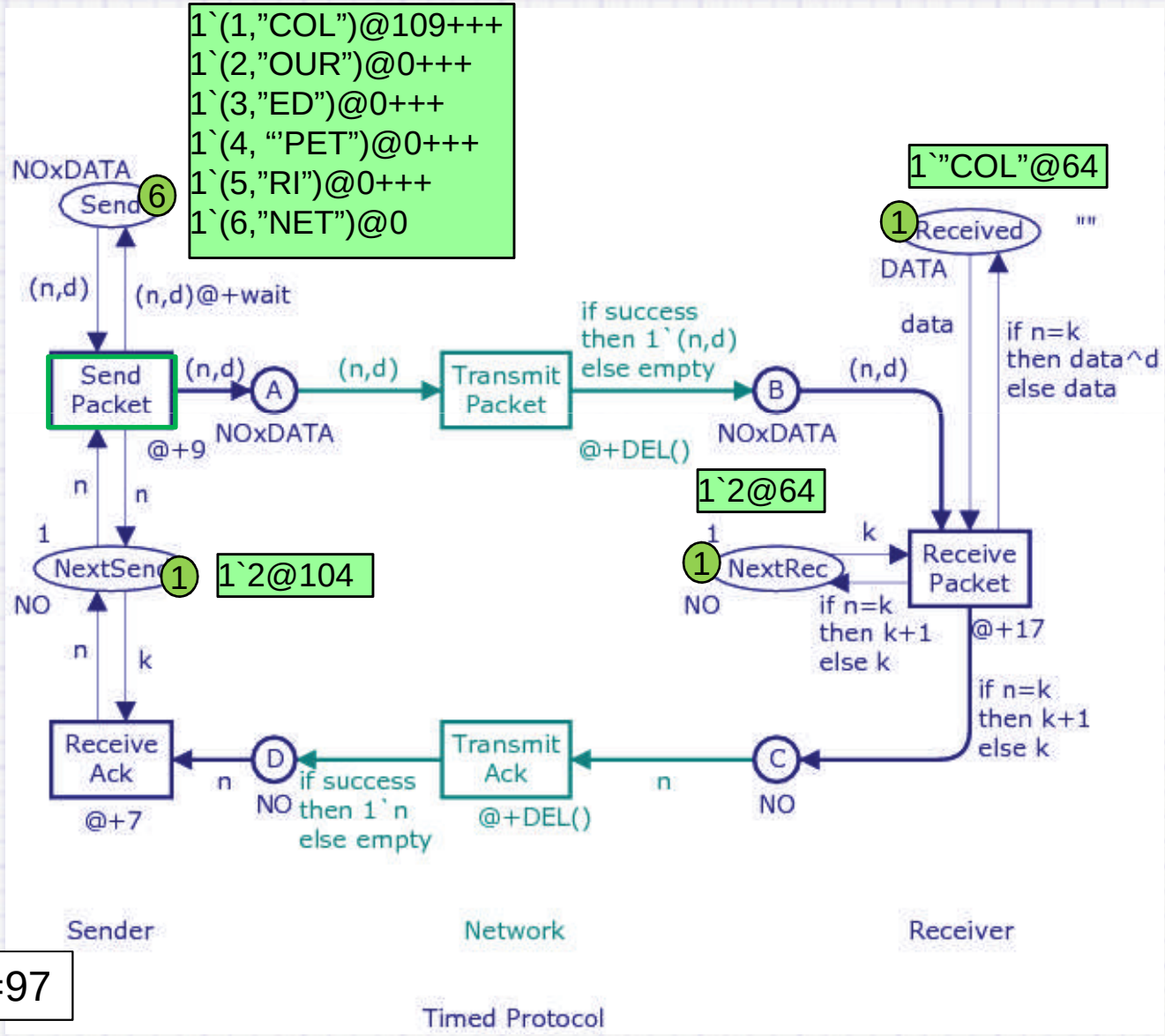


Global Clock=64

Exemplo

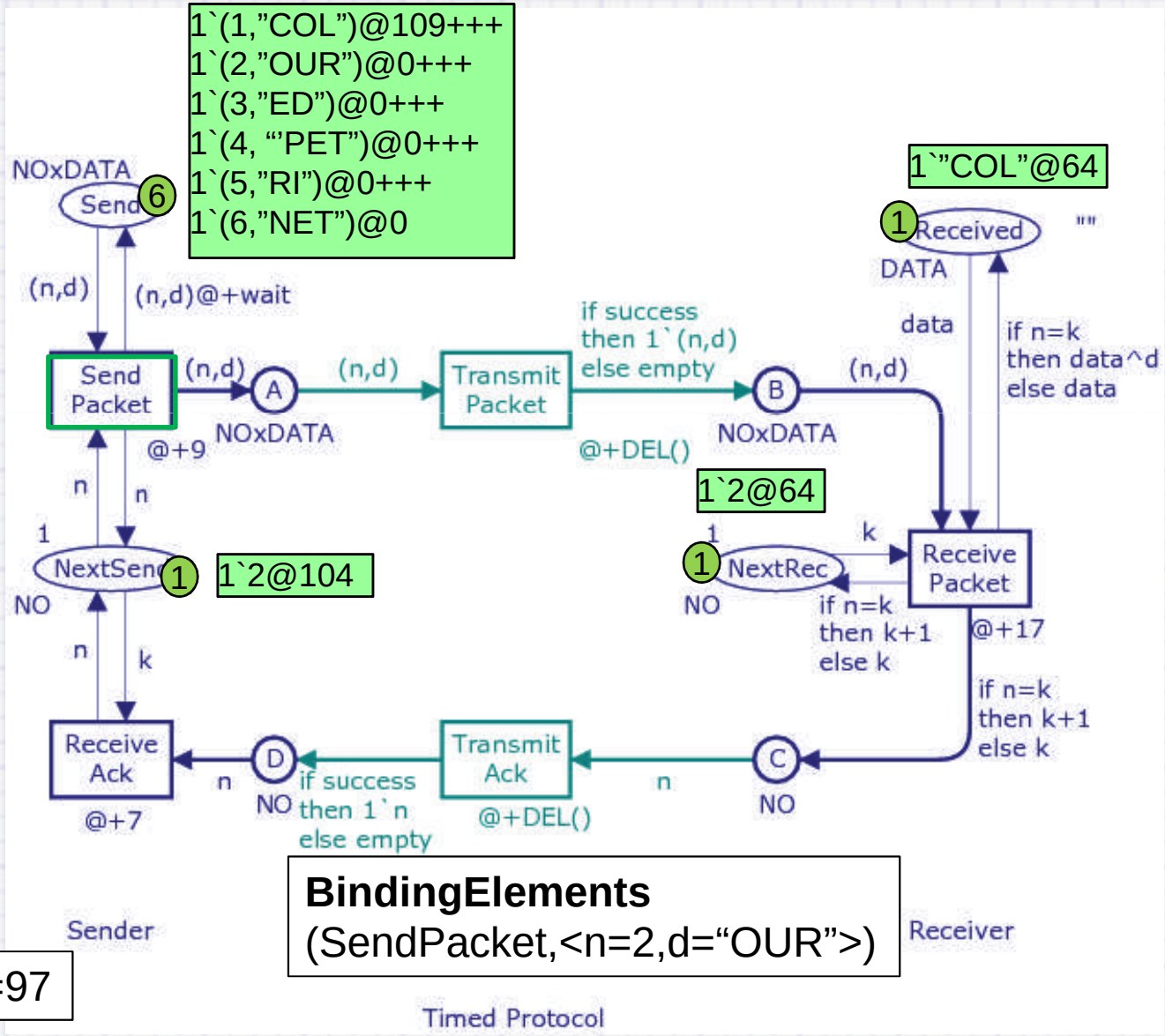

1`(1,"COL")@109+++
1`(2,"OUR")@0+++
1`(3,"ED")@0+++
1`(4, "'PET")@0+++
1`(5,"RI")@0+++
1`(6,"NET")@0
NOxDATA
Send
6
(n,d)
(n,d)@+wait
Send Packet
(n,d)
A
NOxDATA
@+9
(n,d)
Transmit Packet
if success
then 1`(n,d)
else empty
@+DEL()
B
NOxDATA
(n,d)
1`"COL"@64
1
Received
""
DATA
data
if n=k
then data^d
else data
1`2@64
1
k
1
NextRec
NO
if n=k
then k+1
else k
Receive Packet
@+17
n
n
1
NextSend
1
1`1@9
NO
n
k
1`2@97
1
D
NO
if success
then 1`n
else empty
Transmit Ack
@+DEL()
n
C
NO
n
if n=k
then k+1
else k
Receive Ack
@+7
Sender
Receiver
BindingElements
(SendPacket,<n=1,d="COL">)
(ReceiveAck,<n=2,k=1>)
Global Clock=64

Exemplo



Global Clock=97

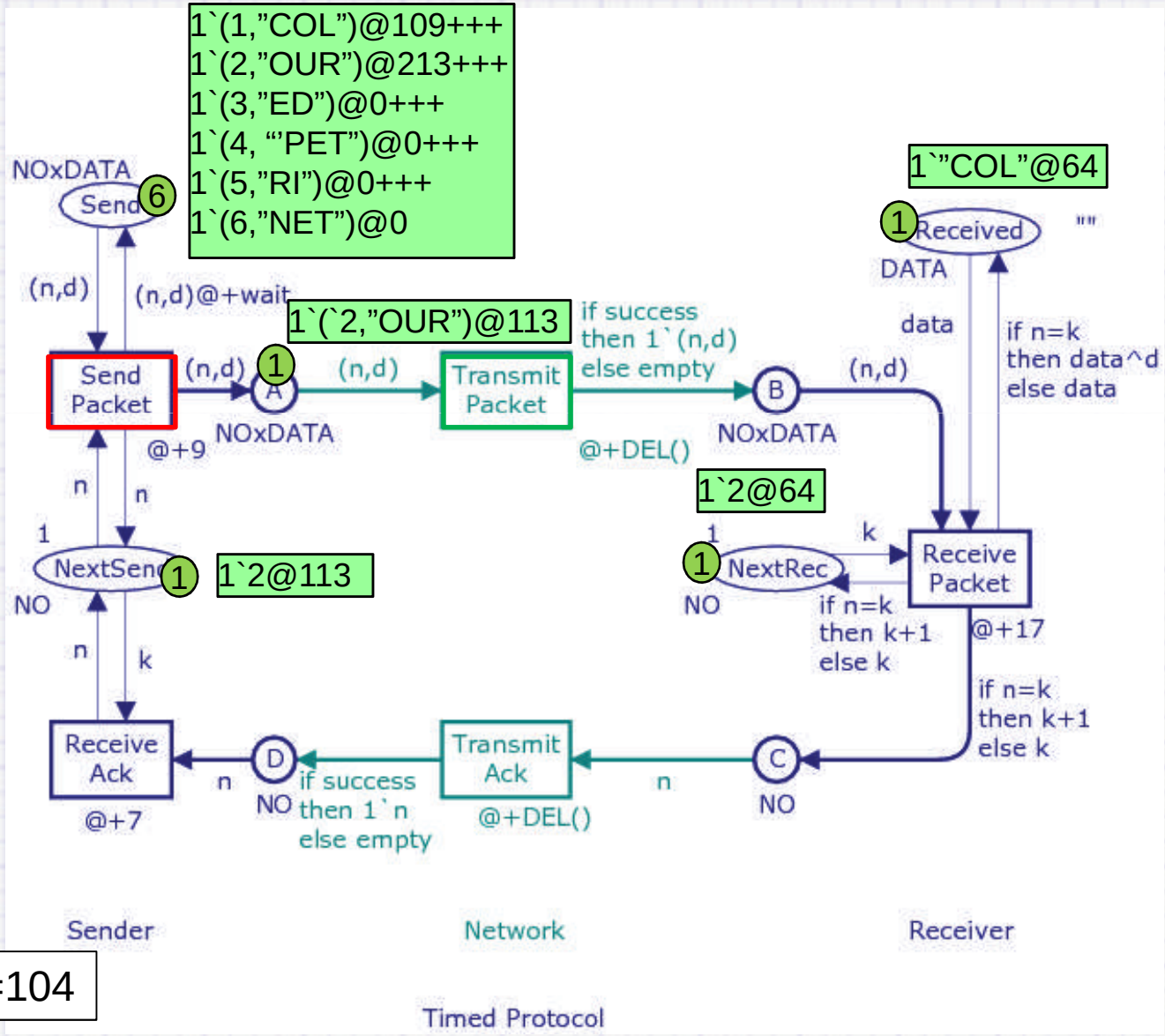
Exemplo



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

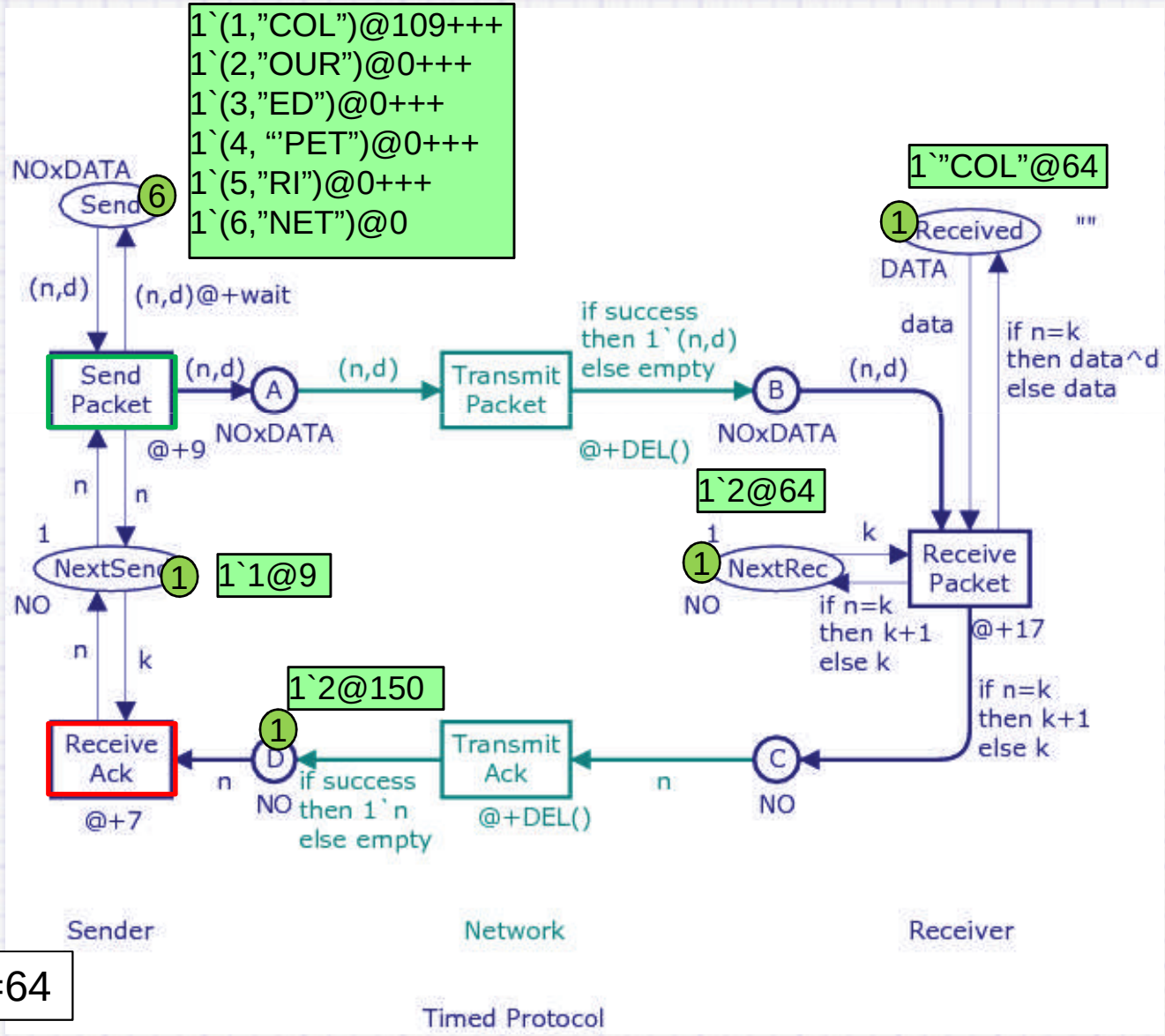
Global Clock=97

Exemplo



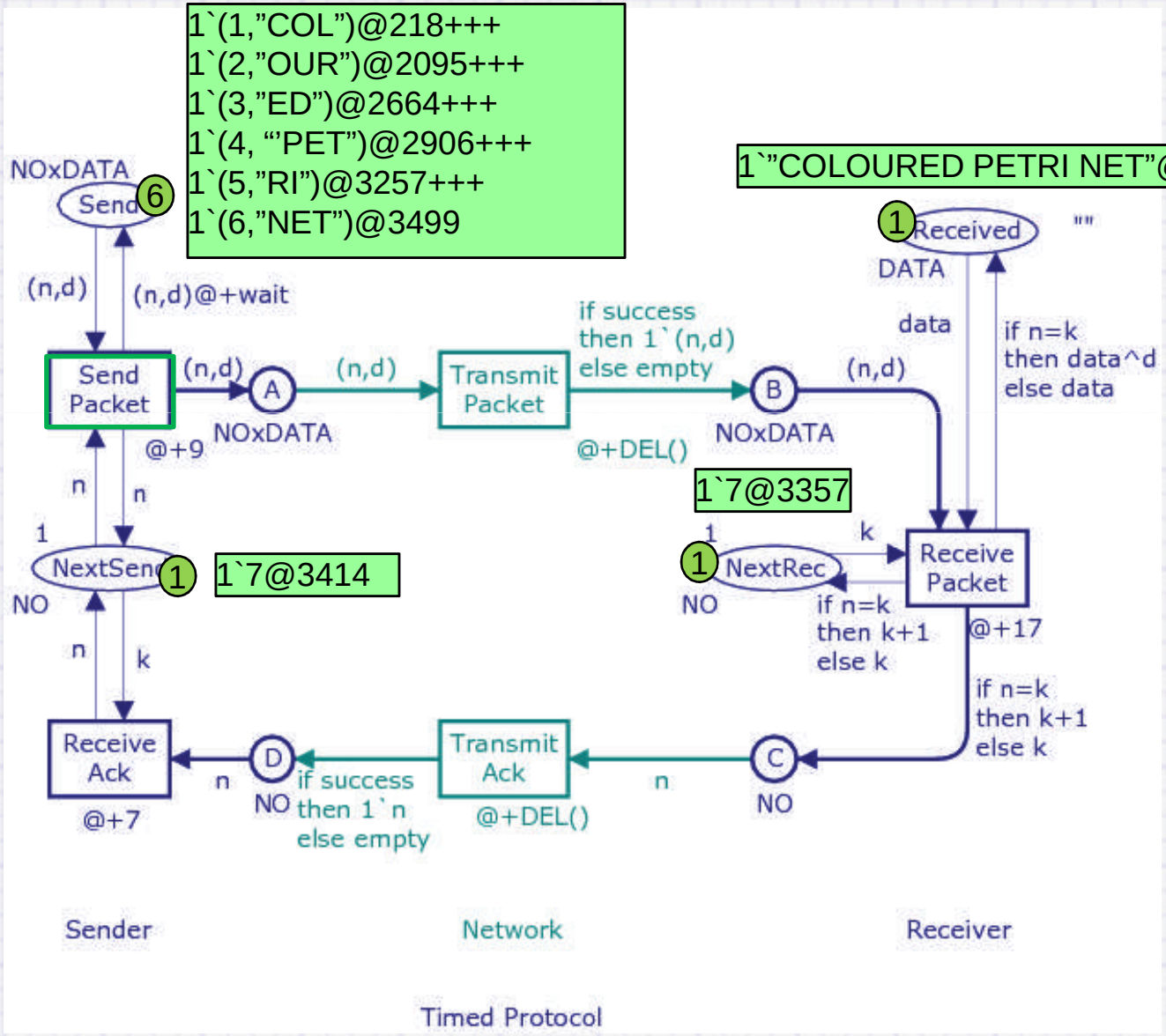
Global Clock=104

Exemplo (Retransmissão)



Global Clock=64

Exemplo (Dead Marking)



Redes de Petri Coloridas Temporizadas (TCPN)

m_d=1’2@405 +++ 2’2@409 +++ 1’2@411+++4’3@410

Assumir um binding element e global clock = 409. Irá remover um token com a cor 2

Remove-se um dos 2’2@409 (remoção do maior tempo possível)

Permite que a marcação alcançada usando semântica de passos possa ser alcançada com interleaving em um order arbitrária

Redes de Petri Coloridas Temporizadas (TCPN)

TCPN podem ter marcações com ou sem *timestamps*

Marcações sem *timestamps* sempre estão prontas para participarem de *binding elements*

Binding elements precisam ser *colour enabled* e *ready*

- *Colour enabled* = Marcação
- *Ready* = tempo (*timestamps* da marcação precisa ser menor ou igual ao valor atual do global clock)

Redes de Petri Coloridas Temporizadas (TCPN)

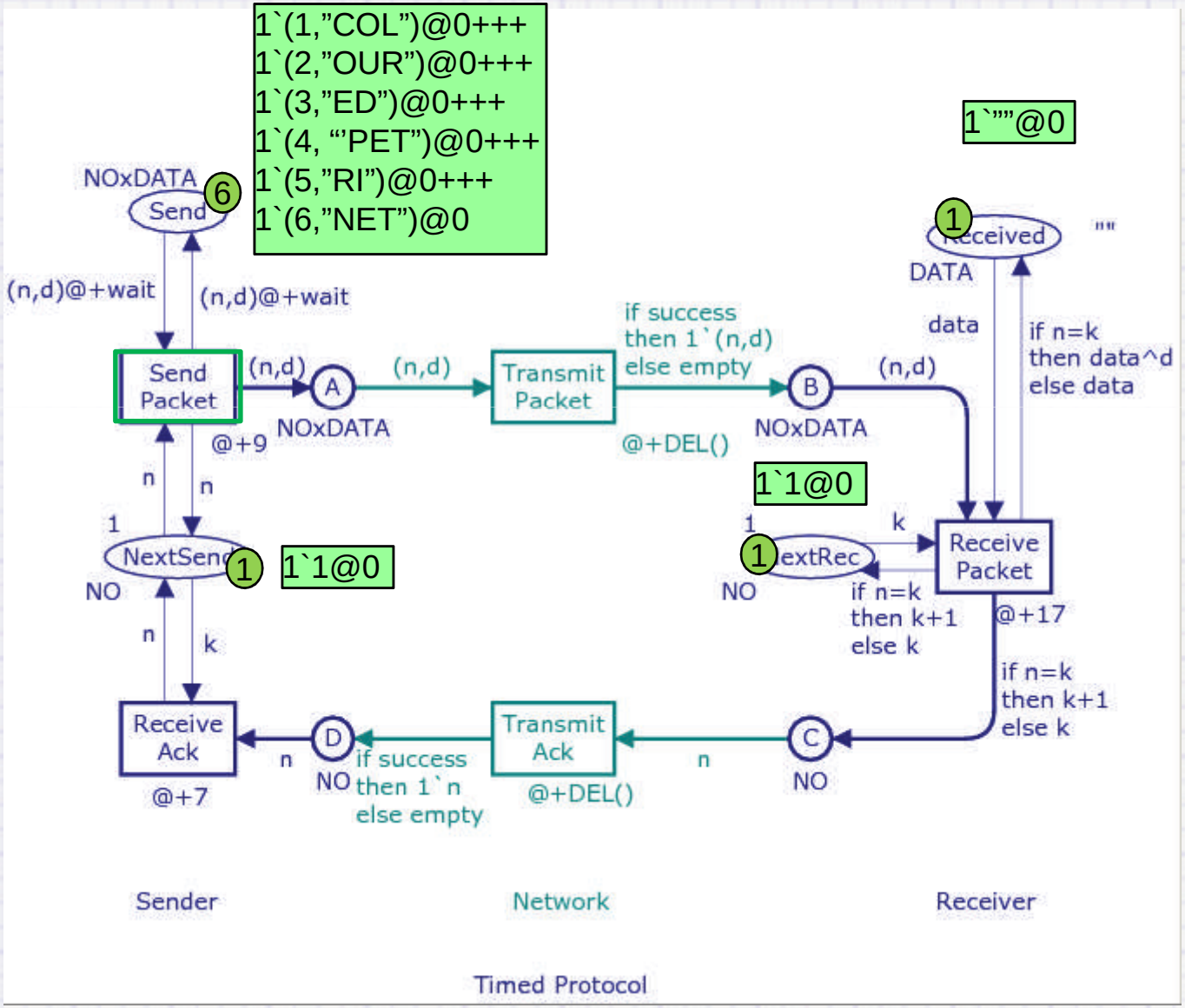
Caso não existam *binding elements* a serem executados, o simulador avança o *global clock* para o o próximo tempo mais cedo no qual *binding elements* possam executar

Cada marcação existe em um intervalo de tempo fechado que pode ser um ponto

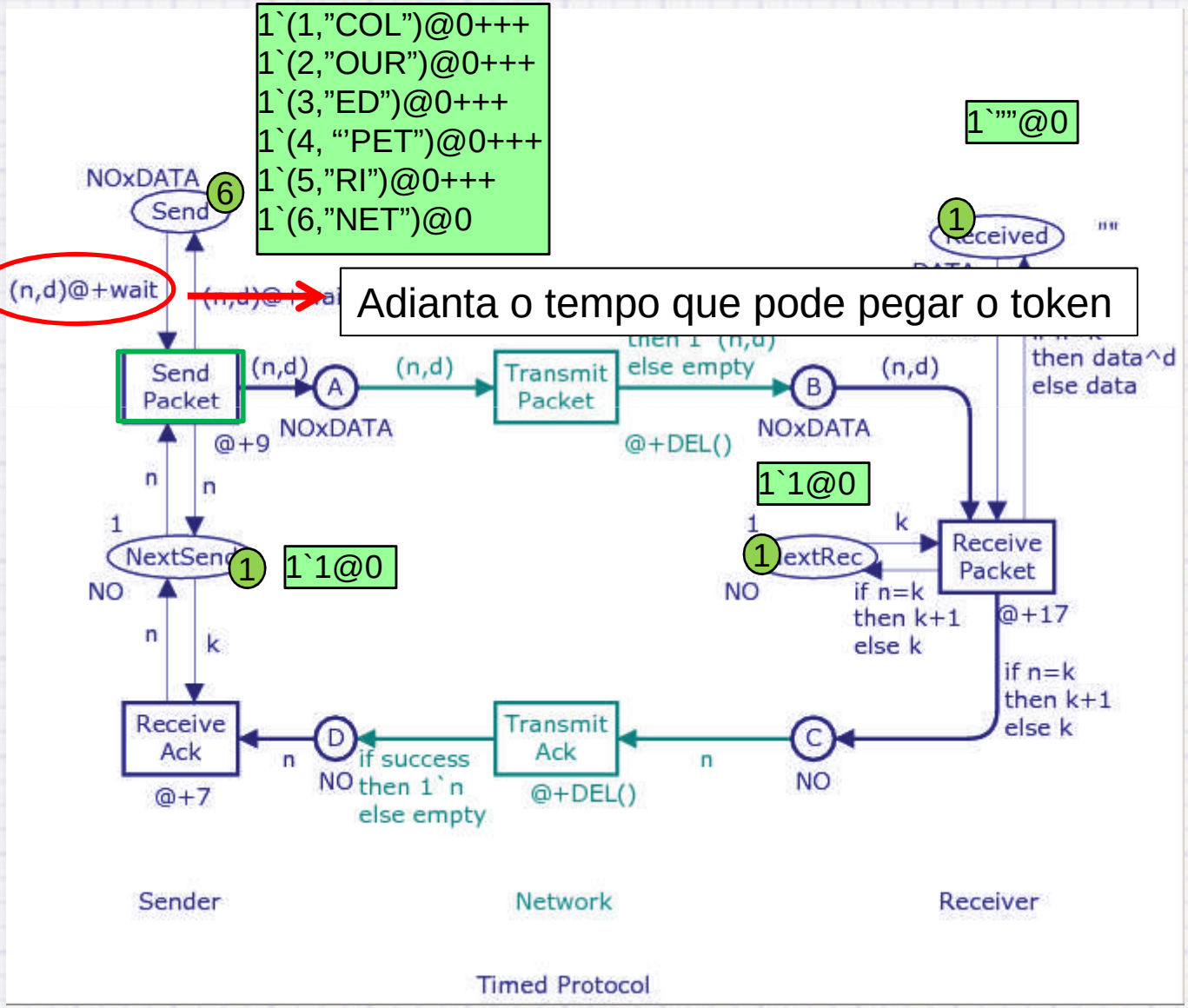
Similarmente, conflitos e concorrência podem acontecer com os *binding elements* no contexto temporizado

TCPN -> CPN (basta remover as inscrições envolvendo tempo). A sequência de ocorrências de uma TCPN é um subconjunto da CPN sem tempo. Tempo adiciona novas restrições

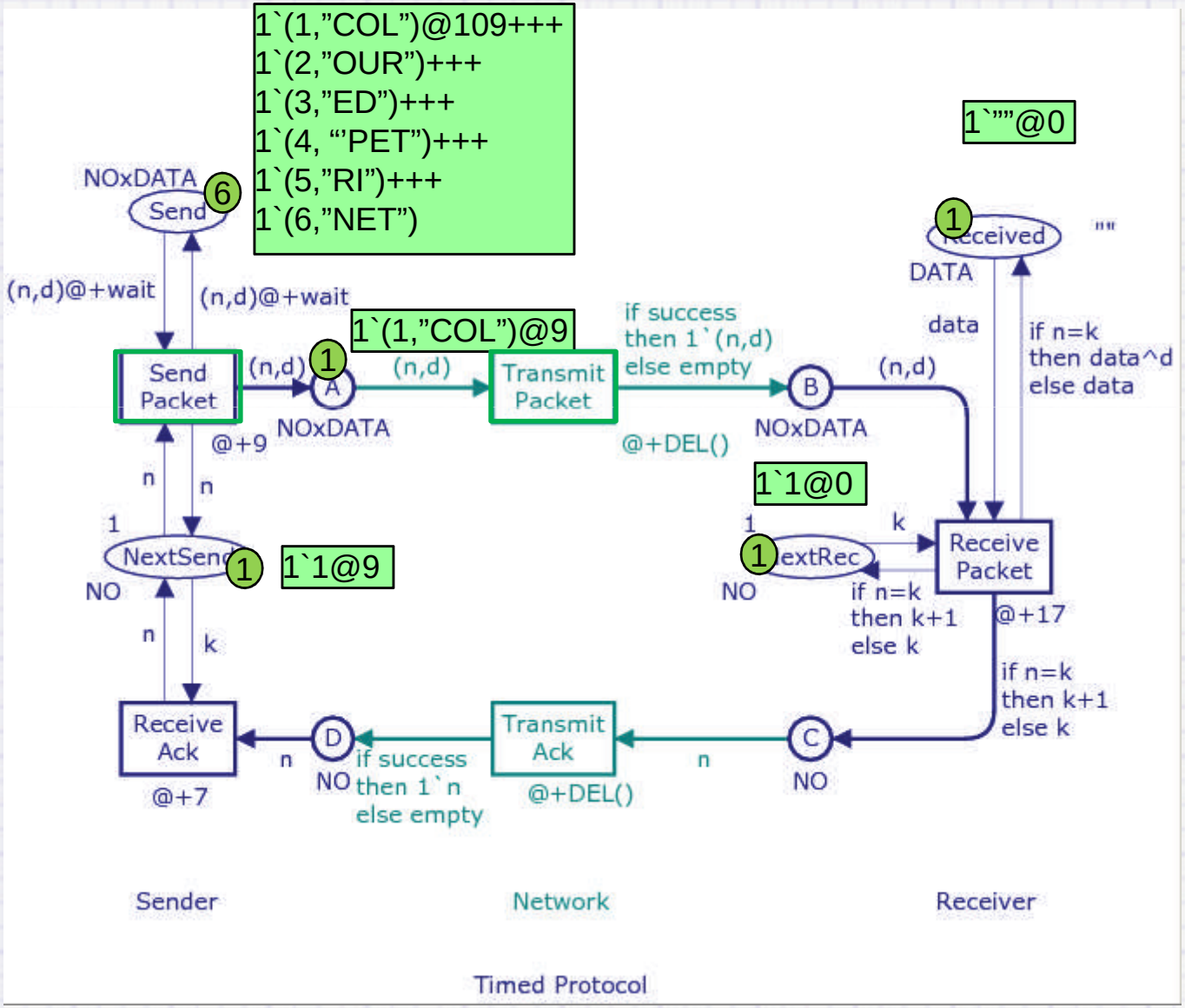
Exemplo


1`(1,"COL")@0+++
1`(2,"OUR")@0+++
1`(3,"ED")@0+++
1`(4, "'PET")@0+++
1`(5,"RI")@0+++
1`(6,"NET")@0
1`""@0
NOxDATA
Send
6
(n,d)@+wait
(n,d)@+wait
Send Packet
(n,d)
A
NOxDATA
@+9
(n,d)
Transmit Packet
if success
then 1`(n,d)
else empty
@+DEL()
B
NOxDATA
(n,d)
Received
DATA
data
""
if n=k
then data^d
else data
n
n
1
NextSend
1
1`1@0
NO
1`1@0
1
k
1
extRec
NO
Receive Packet
if n=k
then k+1
else k
@+17
n
k
if n=k
then k+1
else k
Receive Ack
@+7
n
D
NO
if success
then 1`n
else empty
Transmit Ack
@+DEL()
n
C
NO
Sender
Network
Receiver
Timed Protocol

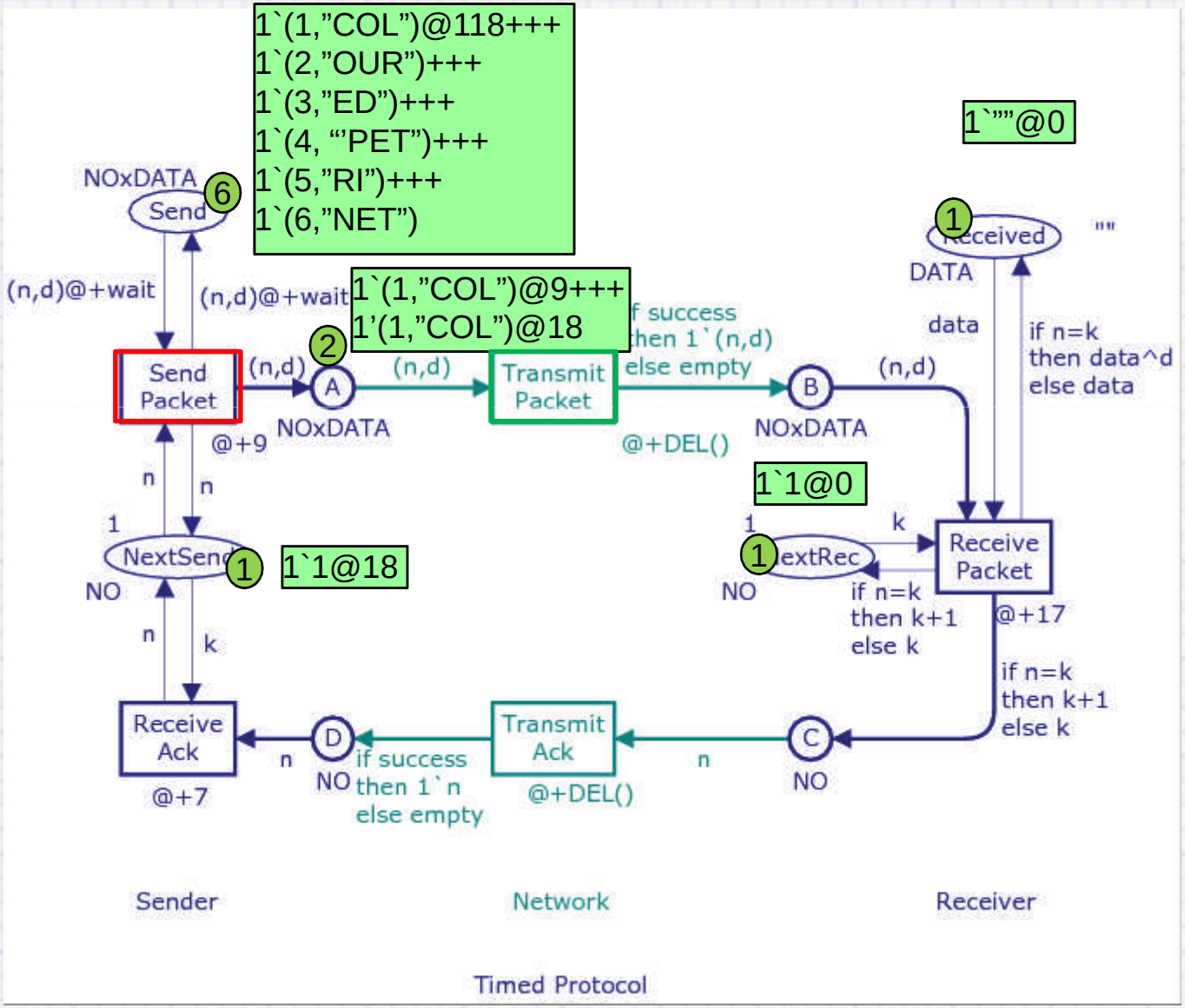
Exemplo


1`(1,"COL")@0+++
1`(2,"OUR")@0+++
1`(3,"ED")@0+++
1`(4, "'PET")@0+++
1`(5,"RI")@0+++
1`(6,"NET")@0
1`""@0
NOxDATA
Send
6
1
Received
""
(n,d)@+wait
Adianta o tempo que pode pegar o token
Send Packet
(n,d)
A
(n,d)
Transmit Packet
else empty
B
(n,d)
then data^d
else data
NOxDATA
@+9
NOxDATA
@+DEL()
1`1@0
n
n
1
k
NextSend
1
1`1@0
NO
1
extRec
NO
Receive Packet
if n=k
then k+1
else k
@+17
n
k
if n=k
then k+1
else k
Receive Ack
@+7
n
D
NO
if success
then 1`n
else empty
Transmit Ack
@+DEL()
n
C
NO
Sender
Network
Receiver
Timed Protocol

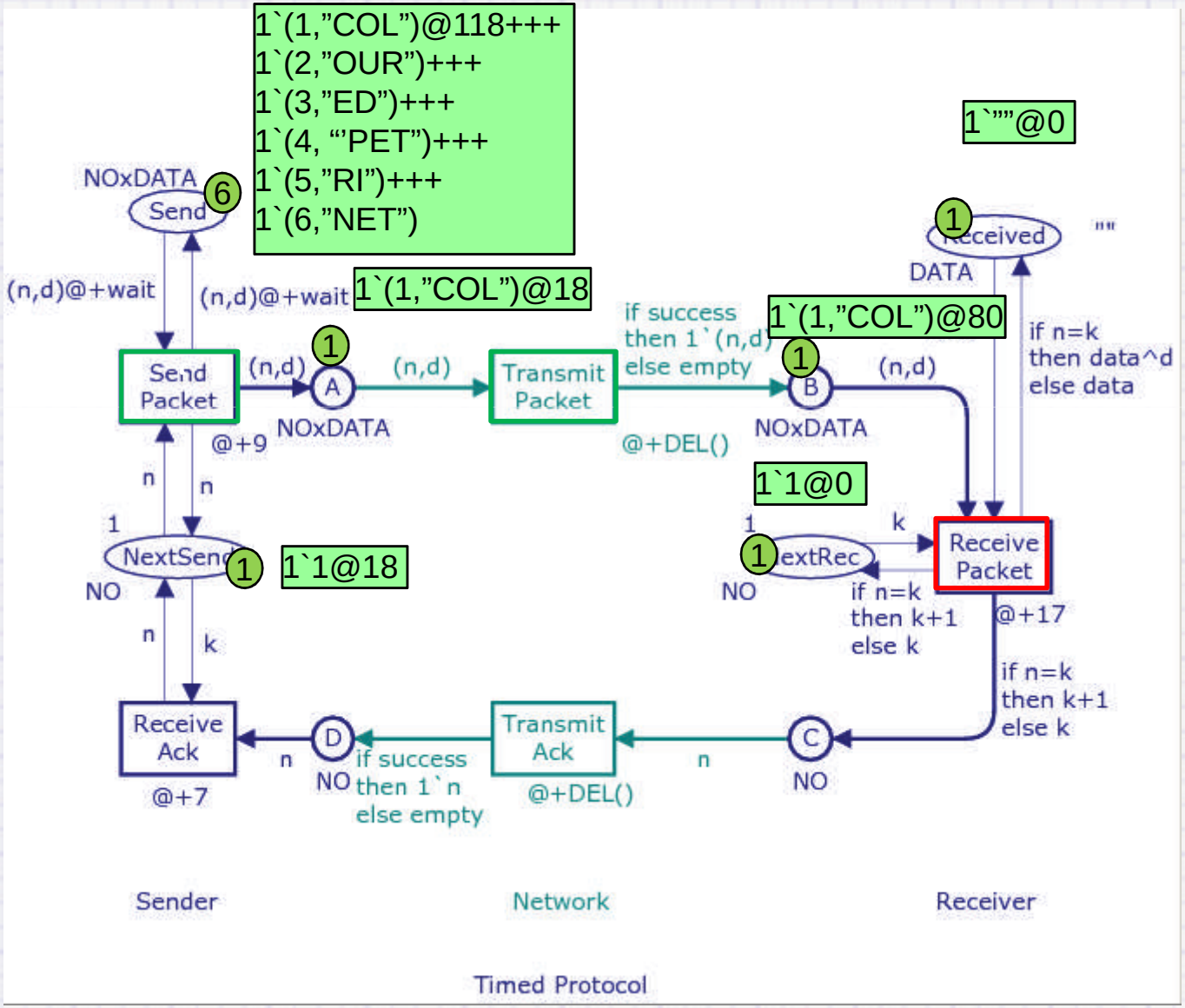
Exemplo


1`(1,"COL")@109+++
1`(2,"OUR")+++
1`(3,"ED")+++
1`(4, "'PET")+++
1`(5,"RI")+++
1`(6,"NET")
1`""@0
NOxDATA
Send
6
(n,d)@+wait
(n,d)@+wait
1`(1,"COL")@9
Send Packet
(n,d)
1
A
(n,d)
Transmit Packet
if success
then 1`(n,d)
else empty
B
(n,d)
NOxDATA
@+9
NOxDATA
@+DEL()
n
n
1
NextSend
1
1`1@9
NO
1`1@0
1
k
1
extRec
NO
Receive Packet
if n=k
then k+1
else k
@+17
if n=k
then k+1
else k
n
k
Receive Ack
@+7
n
D
NO
if success
then 1`n
else empty
Transmit Ack
@+DEL()
n
C
NO
1
Received
DATA
""
data
if n=k
then data^d
else data
Sender
Network
Receiver
Timed Protocol

Exemplo


1`(1,"COL")@118+++
1`(2,"OUR")+++
1`(3,"ED")+++
1`(4, "'PET")+++
1`(5,"RI")+++
1`(6,"NET")
1`""@0
NOxDATA
6
Send
(n,d)@+wait
(n,d)@+wait
1`(1,"COL")@9+++
1'(1,"COL")@18
2
Send Packet
(n,d)
A
(n,d)
Transmit Packet
f success
hen 1`(n,d)
else empty
B
(n,d)
NOxDATA
@+9
NOxDATA
@+DEL()
1
Received
""
DATA
data
if n=k
then data^d
else data
n
n
1
1`1@0
1
k
NextSend
1
1`1@18
1
extRec
Receive Packet
NO
NO
if n=k
then k+1
else k
@+17
n
k
if n=k
then k+1
else k
Receive Ack
@+7
n
D
NO
if success
then 1`n
else empty
Transmit Ack
@+DEL()
n
C
NO
Sender
Network
Receiver
Timed Protocol

Exemplo


1`(1,"COL")@118+++
1`(2,"OUR")+++
1`(3,"ED")+++
1`(4, "'PET")+++
1`(5,"RI")+++
1`(6,"NET")
1`""@0
NOxDATA
6
Send
(n,d)@+wait
(n,d)@+wait
1`(1,"COL")@18
Send Packet
(n,d)
A
(n,d)
Transmit Packet
if success
then 1`(n,d)
else empty
1`(1,"COL")@80
B
(n,d)
Received
DATA
""
if n=k
then data^d
else data
NOxDATA
@+9
@+DEL()
NOxDATA
1`1@0
n
n
1
NextSend
1
1`1@18
NO
1
NextRec
k
Receive Packet
NO
if n=k
then k+1
else k
@+17
n
k
if n=k
then k+1
else k
Receive Ack
@+7
n
D
NO
if success
then 1`n
else empty
Transmit Ack
@+DEL()
n
C
NO
Sender
Network
Receiver
Timed Protocol

Espaço de estados de TCPN

Espaço de estado similar, acrescentando:

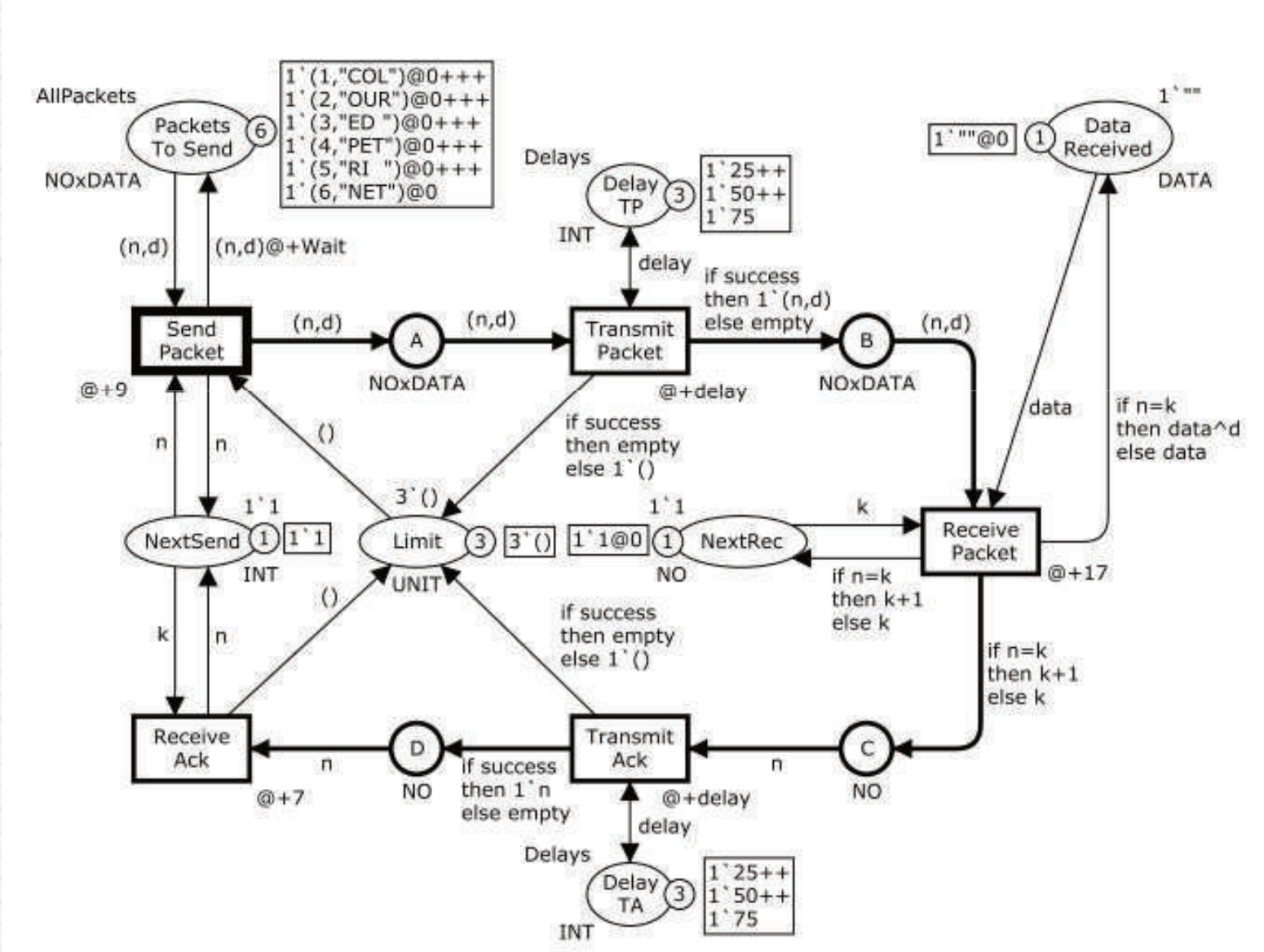
- *timestamps* dos tokens
- O valor do global clock

Duas marcações temporizadas podem ser diferentes mesmo que a correspondente marcação sem tempo sejam iguais

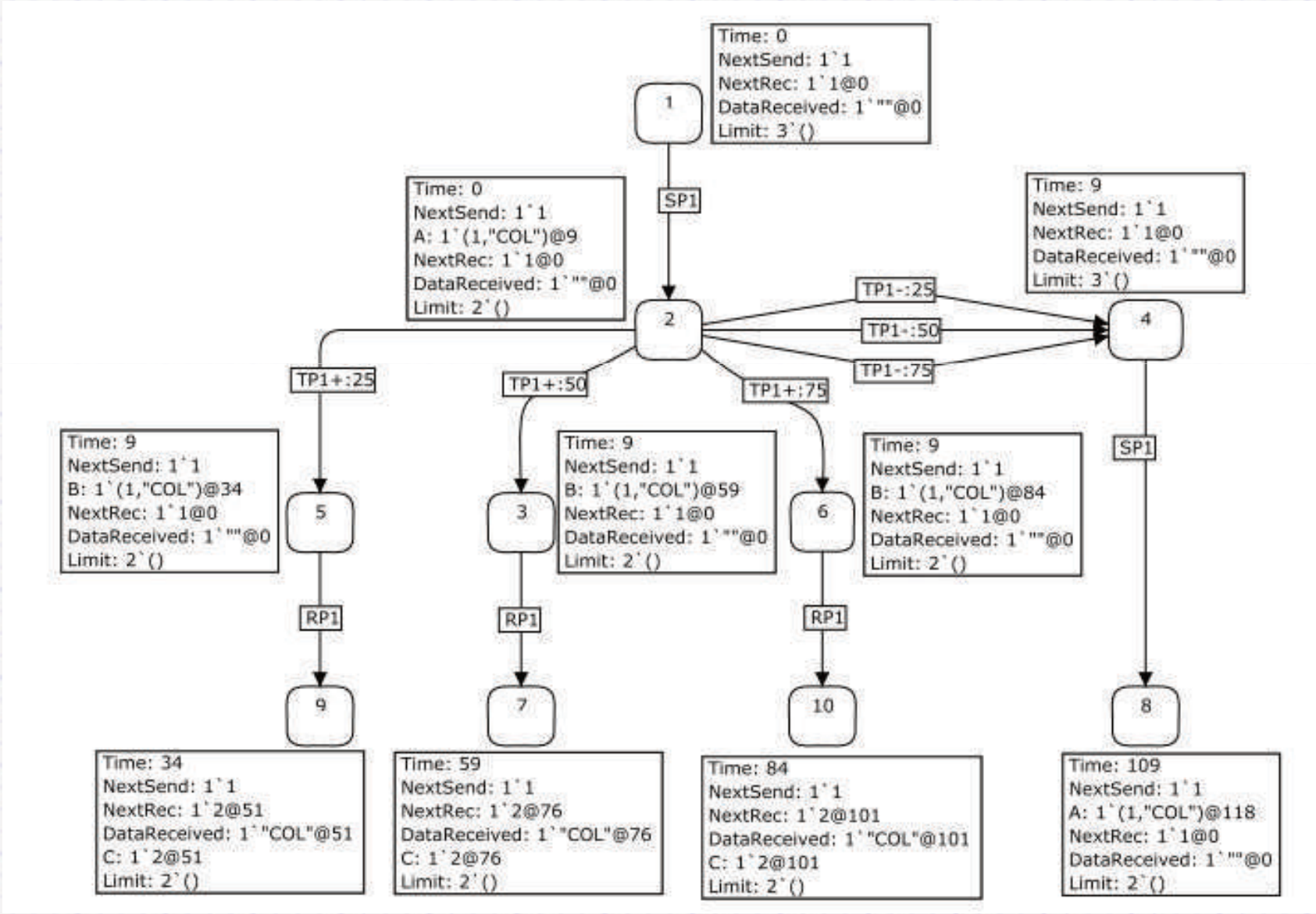
O espaço de estados temporizado pode ser infinito, mesmo que o correspondente espaço de estados sem tempo seja finito

O espaço de estados pode ser maior que o espaço sem tempo

Espaço de estados de TCPN



Espaço de estados de TCPN



Espaço de estados de TCPN

Adoção de *time equivalence method* na CPN tools

Um estado sem o valor absoluto de tempo pode ser adotado pra representar outros estados equivalentes

Construção de um *condensed state space.*

Condensed state space é finito, caso o espaço de estados da CPN sem tempo também seja finito.

Todas a propriedades verificada no espaço de estado completo também são verificadas no *condensed state space*

Timed Multiset (Multiconjunto Temporizado)

Definition 11.1. Let S be a non-empty set and let $\mathbb{T}$ be the set of time values. A **timed multiset** over S is a function $tm : S \times \mathbb{T} \to \mathbb{N}$ that maps each element $(s,t) \in S \times \mathbb{T}$ into a non-negative integer $tm(s,t) \in \mathbb{N}$. It is required that the sum

$$tm(s) = \sum_{t \in \mathbb{T}} tm(s,t)$$

is finite for all $s \in S$. The non-negative integer $tm(s)$ is the **number of appearances** (or coefficient) of s in tm. The **timestamp list** for an element s is a list

$$tm[s] = [t_1, t_2, \ldots, t_{tm(s)}]$$

satisfying $t_i \leq t_{i+1}$ for all $1 \leq i < tm(s)$. It contains the time values t for which $tm(s,t) > 0$ and a time value t appears $tm(s,t)$ times in the list. **Membership** and **size** are defined as follows, where tm is a timed multiset:

1. $\forall s \in S : s \in tm \Leftrightarrow tm(s) > 0$.
2. $|tm| = \sum_{s \in S} tm(s)$.

A timed multiset tm is **infinite** if $|tm| = \infty$. Otherwise tm is **finite**. The set of all timed multisets over S is denoted S_{TMS}. The empty multiset over S is denoted $\emptyset_{TMS}$ and is defined by $\emptyset_{TMS}(s,t) = 0$ for all $(s,t) \in S \times \mathbb{T}$.

□

Timed Multiset (Multiconjunto Temporizado)

tm_B=1`(1,”COUL”)@2030 +++ 1`(2,”OUR”)@2015 +++
2`(2,”OUR”)@`2005 +++ 1`(2,”OUR”)@1994

$tm_B(s,t)=$
- 1 if(s,t) = ((1,”COL”),2030)
- 1 if(s,t) = ((2,”OUR”),2015)
- 2 if(s,t) = ((2,”OUR”),2005)
- 1 if(s,t) = ((2,”OUR”),1994)
- 0 caso contrário

tm_B(2,”OUR”)=4

tm_B[(2,”OUR”)] = [1994,2005,2005,2015]

Formalização TCPN

Definition 11.2. For timed multisets over a set S and time values $\mathbb{T}$, **comparison**, **subtraction**, and **addition of time** on timestamp lists are defined as follows, where $tm[s] = [t_1, t_2, \ldots, t_{tm(s)}]$, $tm_1[s] = [t_1^1, t_2^1, \ldots, t_{tm_1(s)}^1]$, and $tm_2[s] = [t_1^2, t_2^2, \ldots, t_{tm_2(s)}^2]$ are timestamp lists of an element $s \in S$:

1. $tm_1[s] \leq_{[\mathbb{T}]} tm_2[s] \Leftrightarrow tm_1(s) \leq tm_2(s)$ and $t_i^1 \geq t_i^2$ for all $1 \leq i \leq tm_1(s)$
2. For $t \in \mathbb{T}$ such that $t \geq t_1$, $tm[s] -_{\mathbb{T}} t$ is the timestamp list
   - $tm[s] -_{\mathbb{T}} t = [t_1, t_2, t_3, \ldots, t_{i-1}, t_{i+1}, \ldots, t_{tm(s)}]$ where i is the largest index for which $t_i \leq t$.
3. When $tm_1[s] \leq_{[\mathbb{T}]} tm_2[s]$, $tm_2[s] -_{[\mathbb{T}]} tm_1[s]$ is the timestamp list defined by
   - $tm_2[s] -_{[\mathbb{T}]} tm_1[s] = ((([t_1^2, t_2^2, \ldots, t_{tm_2(s)}^2] -_{\mathbb{T}} t_1^1) -_{\mathbb{T}} t_2^1) \cdots -_{\mathbb{T}} t_{tm_1(s)}^1)$.
4. For $t \in \mathbb{T}$, $tm[s]_{+t}$ is the timestamp list defined by
   - $tm[s]_{+t} = [t_1 + t, t_2 + t, \ldots, t_{tm(s)} + t]$

□

Formalização TCPN

Definition 11.3. For timed multisets over a set S and time values $\mathbb{T}$, **addition**, **comparison**, **subtraction**, and **addition of time** are defined as follows, where tm, tm_1, and tm_2 are timed multisets:

1. $\forall (s,t) \in S \times \mathbb{T} : (tm_1 +\!+\!+\, tm_2)(s,t) = tm_1(s,t) + tm_2(s,t)$.
2. $tm_1 \lll\!= tm_2 \Leftrightarrow \forall s \in S : tm_1[s] \leq_{[\mathbb{T}]} tm_2[s]$.
3. When $tm_1 \lll\!= tm_2$, $tm_2 ---\, tm_1$ is the timed multiset defined by
   - $\forall s \in S : (tm_2 ---\, tm_1)(s) = tm_2(s) - tm_1(s)$;
   - $\forall s \in S : (tm_2 ---\, tm_1)[s] = tm_2[s] -_{[\mathbb{T}]} tm_1[s]$.
4. For $t \in \mathbb{T}$, tm_{+t} is the timed multiset defined by
   - $\forall s \in S : tm_{+t}(s) = tm(s)$ and $tm_{+t}[s] = tm[s]_{+t}$.

□

Timed Multiset (Multiconjunto Temporizado)

tm_{RP}=1`(2,"OUR")@2010

tm_B=1`(1,"COUL")@2030 +++ 1`(2,"OUR")@2015 +++
2`(2,"OUR")@`2005 +++ 1`(2,"OUR")@1994

tm_{RP}((2,"OUR"))=1

tm_B((2,"OUR"))=5

tm_{RP}[(2,"OUR")] = [2010]

tm_B[(2,"OUR")] = [1994,2005,2005,2015]

$(tm_B --- tm_{RP})(s)=$
- 1 if(s,t) = (1,"COL")
- 3 if(s,t) = (2,"OUR")
- 0 caso contrário

$(tm_B --- tm_{RP})$[(2,"OUR")]=[1994,2005,2015]

$tm_B --- tm_{RP}$=1`(1,"OUR")@2030 +++ 1`(2,"OUR")@2015 +++
1`(2,"OUR")@2005 +++ 1`(2,"OUR")@1994

Formalização TCPN

Definition 11.4. A **timed non-hierarchical Coloured Petri Net** is a nine-tuple $CPN_T = (P, T, A, \Sigma, V, C, G, E, I)$ where:

1. P is a finite set of **places**.
2. T is a finite set of **transitions** such that $P \cap T = \emptyset$.
3. $A \subseteq P \times T \cup T \times P$ is a set of directed **arcs**.
4. Σ is a finite set of non-empty **colour sets**. Each colour set is either untimed or timed.
5. V is a finite set of **typed variables** such that $Type[v] \in \Sigma$ for all variables $v \in V$.
6. $C : P \rightarrow \Sigma$ is a **colour set function** that assigns a colour set to each place. A place p is timed if $C(p)$ is timed, otherwise p is untimed.
7. $G : T \rightarrow EXPR_V$ is a **guard function** that assigns a guard to each transition t such that $Type[G(t)] = Bool$.
8. $E : A \rightarrow EXPR_V$ is an **arc expression function** that assigns an arc expression to each arc a such that

   - $Type[E(a)] = C(p)_{MS}$ if p is untimed;
   - $Type[E(a)] = C(p)_{TMS}$ if p is timed.

   Here, p is the place connected to the arc a.
9. $I : P \rightarrow EXPR_\emptyset$ is an **initialisation function** that assigns an initialisation expression to each place p such that

   - $Type[I(p)] = C(p)_{MS}$ if p is untimed;
   - $Type[I(p)] = C(p)_{TMS}$ if p is timed.

□

Formalização TCPN

Definition 11.5. For a timed Coloured Petri Net $CPN_T = (P, T, A, \Sigma, V, C, G, E, I)$, we define the following concepts:

1. A **marking** is a function M that maps each place $p \in P$ into a multiset $M(p)$ of tokens such that
   - $M(p) \in C(p)_{MS}$ if p is untimed;
   - $M(p) \in C(p)_{TMS}$ if p is timed.
2. A **timed marking** is a pair (M, t^*), where M is a marking and $t^* \in \mathbb{T}$ is the value of the global clock.
3. The **initial timed marking** is the pair $(M_0, 0)$, where M_0 is defined by $M_0(p) = I(p)\langle\rangle$ for all $p \in P$.

Formalização TCPN

Definition 11.6. A step $Y \in BE_{MS}$ is **enabled** at time t' in a timed marking (M,t^*) if and only if the following properties are satisfied:

1. $\forall(t,b) \in Y : G(t)\langle b\rangle$.
2. $\overset{++}{\underset{MS}{\sum}}_{(t,b)\in Y} E(p,t)\langle b\rangle \ll= M(p)$ for all untimed places $p \in P$.
3. $\overset{+++}{\underset{MS}{\sum}}_{(t,b)\in Y} (E(p,t)\langle b\rangle)_{+t'} \lll= M(p)$ for all timed places $p \in P$.
4. $t^* \leq t'$.
5. t' is the smallest time value for which there exists a step satisfying conditions 1–4.

When Y is enabled in (M,t^*) at time t', it may **occur** at time t', leading to the timed marking (M',t') defined by:

6. $M'(p) = (M(p) -- \overset{++}{\underset{MS}{\sum}}_{(t,b)\in Y} E(p,t)\langle b\rangle) ++ \overset{++}{\underset{MS}{\sum}}_{(t,b)\in Y} E(t,p)\langle b\rangle$

   for all untimed places $p \in P$.
7. $M'(p) = (M(p) --- \overset{+++}{\underset{MS}{\sum}}_{(t,b)\in Y} E(p,t)\langle b\rangle_{+t'}) +++ \overset{+++}{\underset{MS}{\sum}}_{(t,b)\in Y} E(t,p)\langle b\rangle_{+t'}$

   for all timed places $p \in P$.

□