# Coloured Petri Nets

© Kurt Jensen
Department of Computer Science
University of Aarhus, Denmark

kjensen@daimi.au.dk
www.daimi.au.dk/~kjensen/

**TOOLS**
- editing
- simulation
- verification

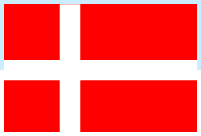
**THEORY**
- models
- basic concepts
- analysis methods

**PRACTICAL USE**
- specification
- validation
- verification
- implementation

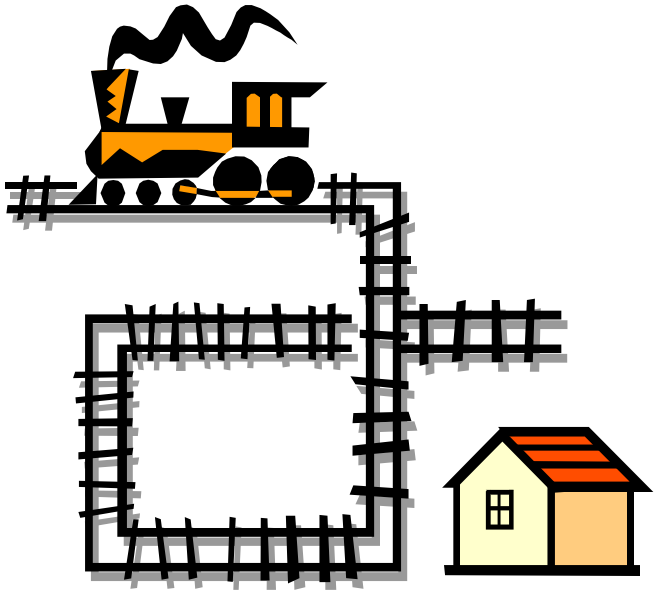
This slide set can be downloaded from:
http://www.daimi.au.dk/CPnets/slides/

# O que é Rede de Petri Colorida?

- *Linguagem de Modelagem para sistemas onde sincronização, comunicação e compartilhamento de recursos são importantes.*
- Combinação de *redes de Petri* e *linguagem de Progamação.*
  - ***Estruturas de controle, sincronização, comunicação*** e ***compartilhamento de recursos*** são descritas em *redes de Petri.*
  - ***Dados*** e ***manipulação de dados*** são descritos em *linguagem de programação funcional.*
- Modelos CPN são *validados* através de ***simulação*** e *verificado* através de ***espaço de estados****.*
- Redes de Petri Coloridas são desenvolvidas na ***University of Aarhus***, *Dinamarca* 25 anos atrás.



# Porque fazer modelos?



- Fazemos modelos para:
  - *Aprender novas coisas* sobre um sistema.
  - Checar se o sistema projetado possui as *propriedades esperadas.*

- *Modelos CPN* são *dinâmicos:*
  - Eles podem ser *executados* em um *computador.*
  - Isto possibilita simular e *investigar* diferentes *cenários.*

# Visão Geral

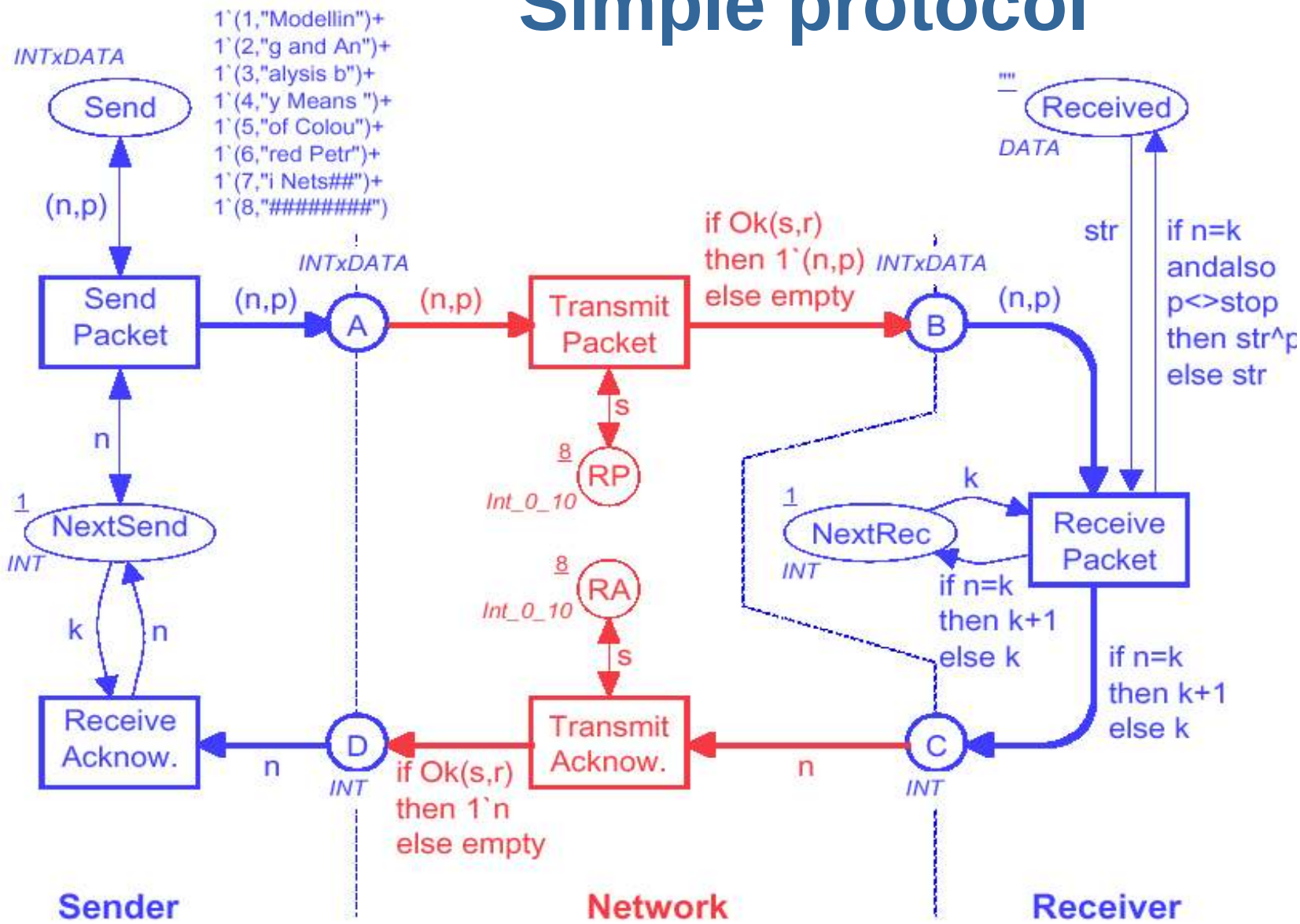
## Modelagem

- **Linguagem básica**
  - sintaxe
  - semântica
- Extensões
  - módulos
  - tempo
- Ferramenta de suporte
  - edição
  - simulação

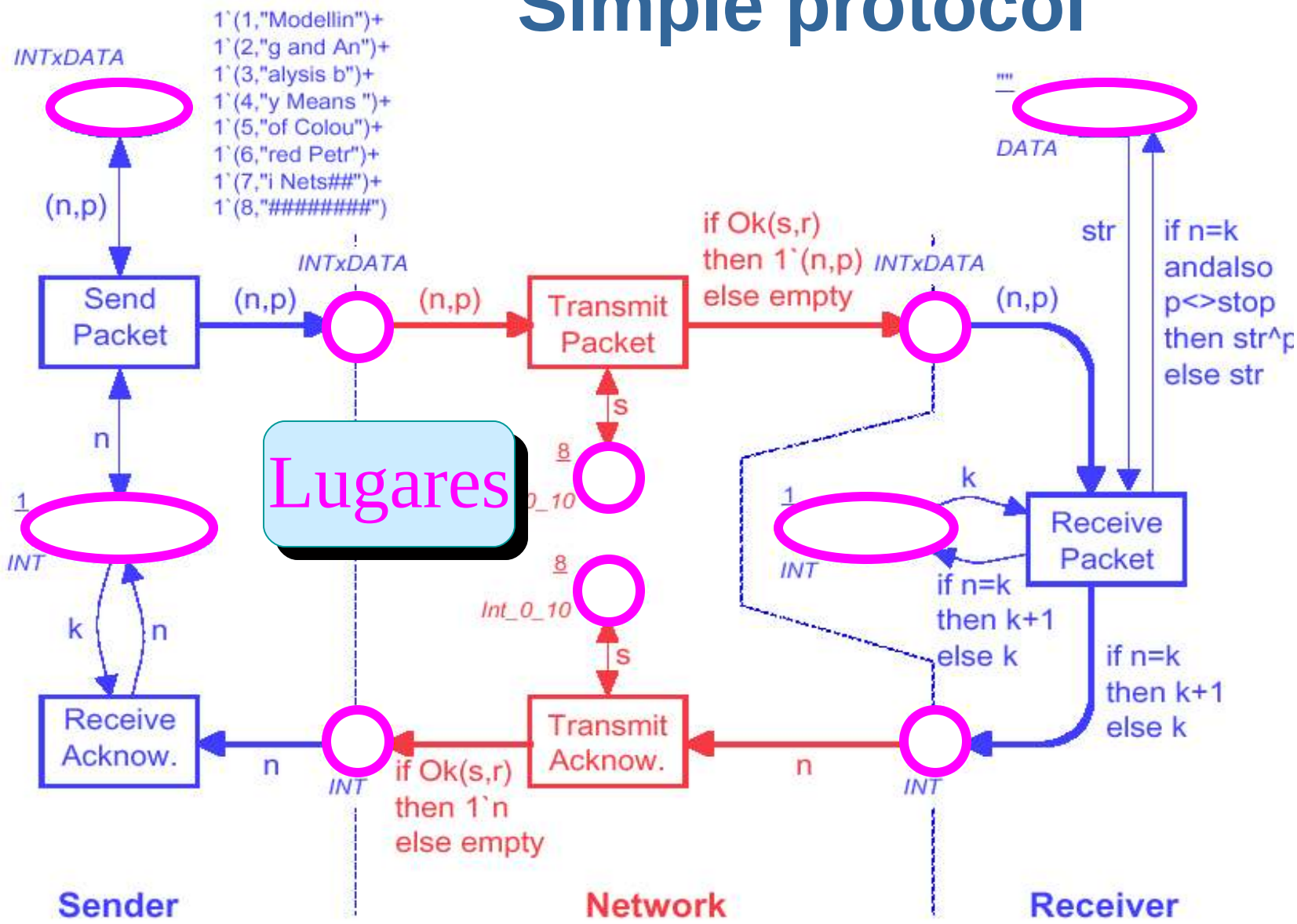
## Análise

- Espaço de Estado
  - completo
  - simetrias

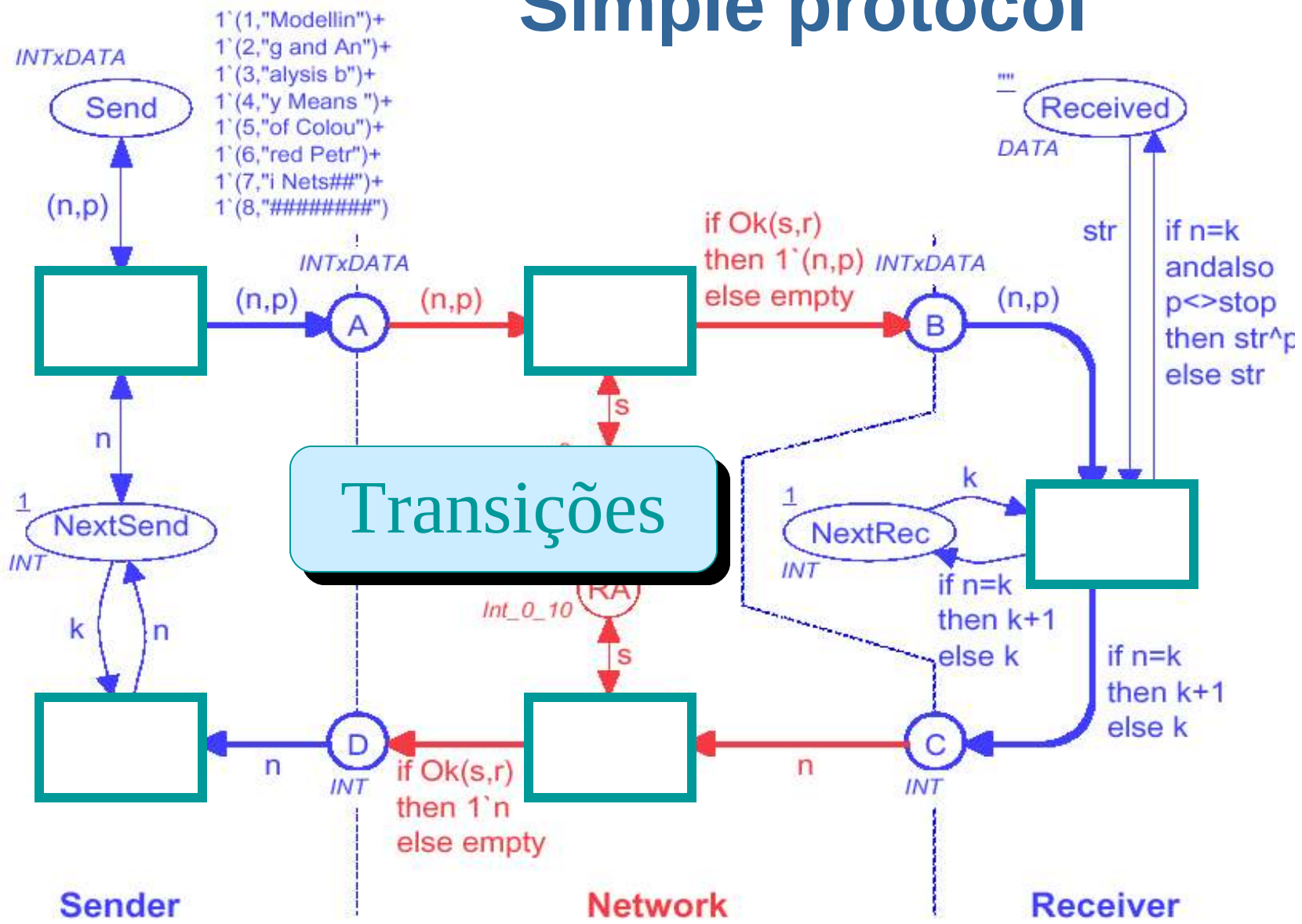
# Simple protocol



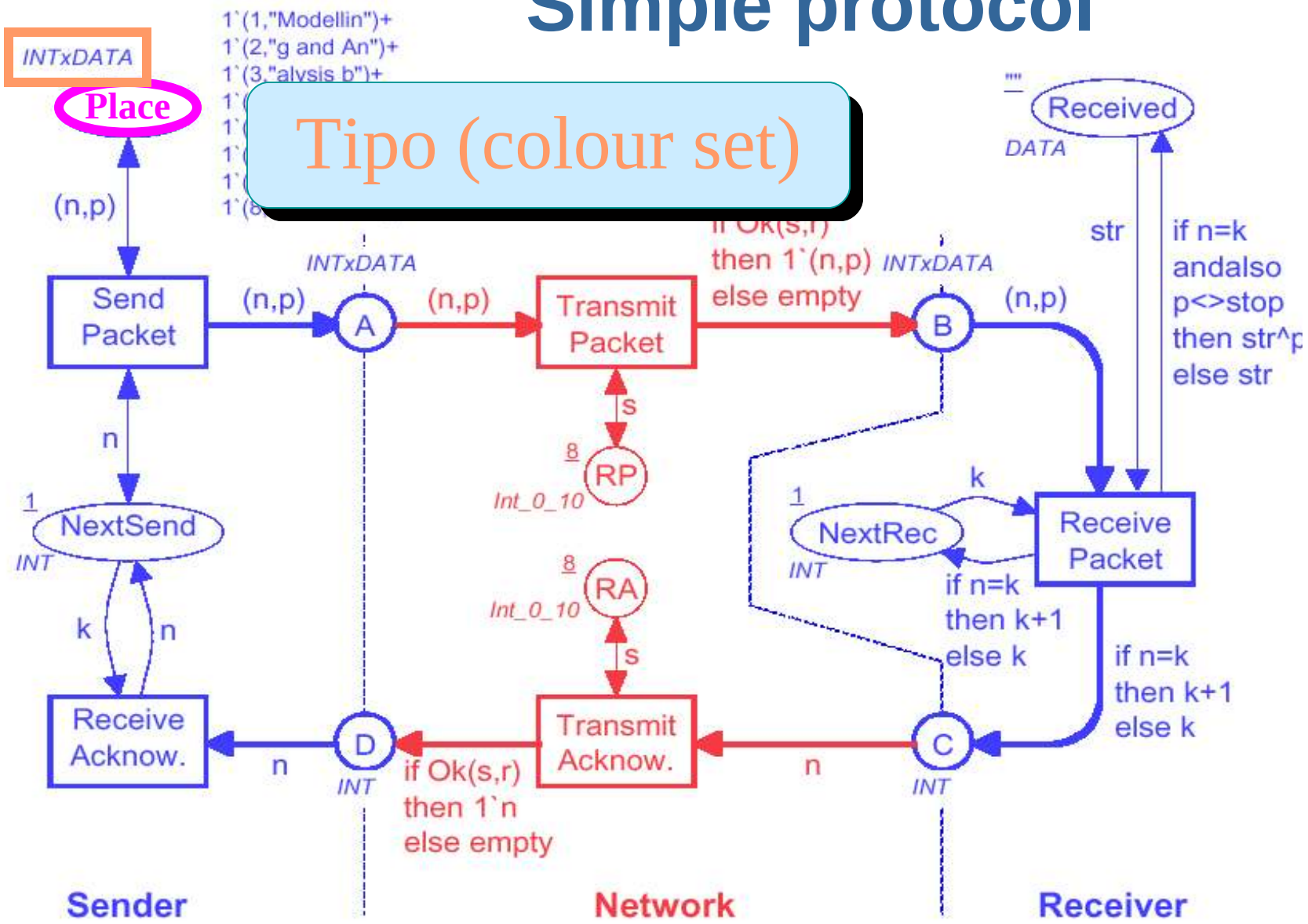
# Simple protocol



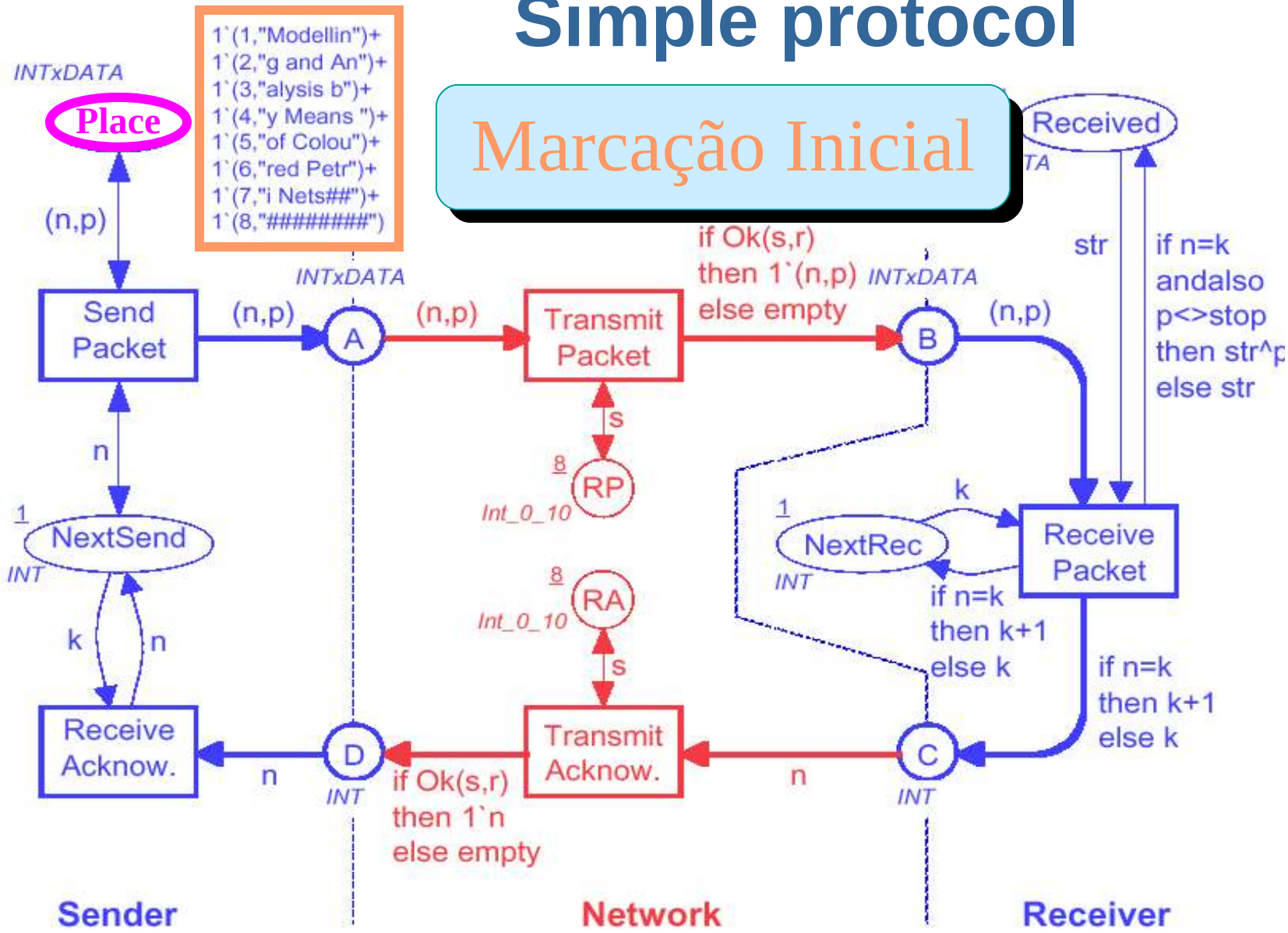
# Simple protocol



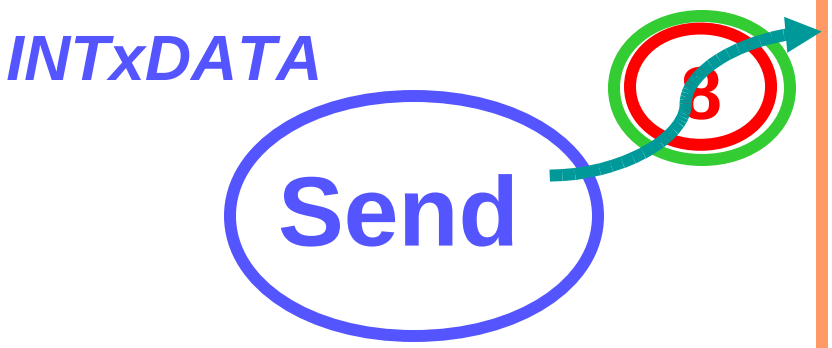
# Simple protocol



# Simple protocol



# Marcação de Send

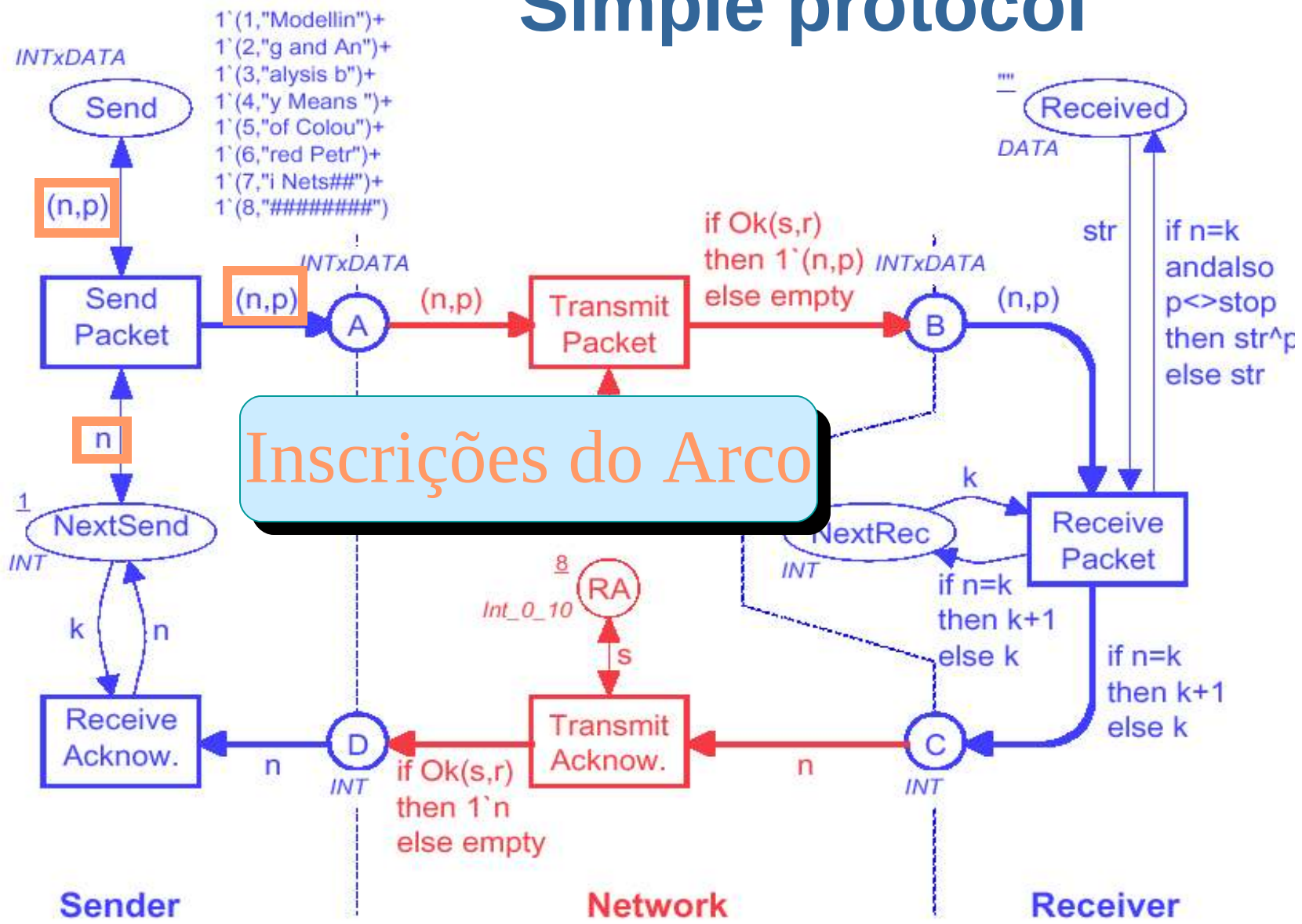



1 `(1,"Modellin") +
1 `(2,"g and An") +
1 `(3,"alysis b") +
1 `(4,"y Means ") +
1 `(5,"of Colou") +
1 `(6,"red Petr") +
1 `(7,"i Nets##") +
1 `(8,"########")

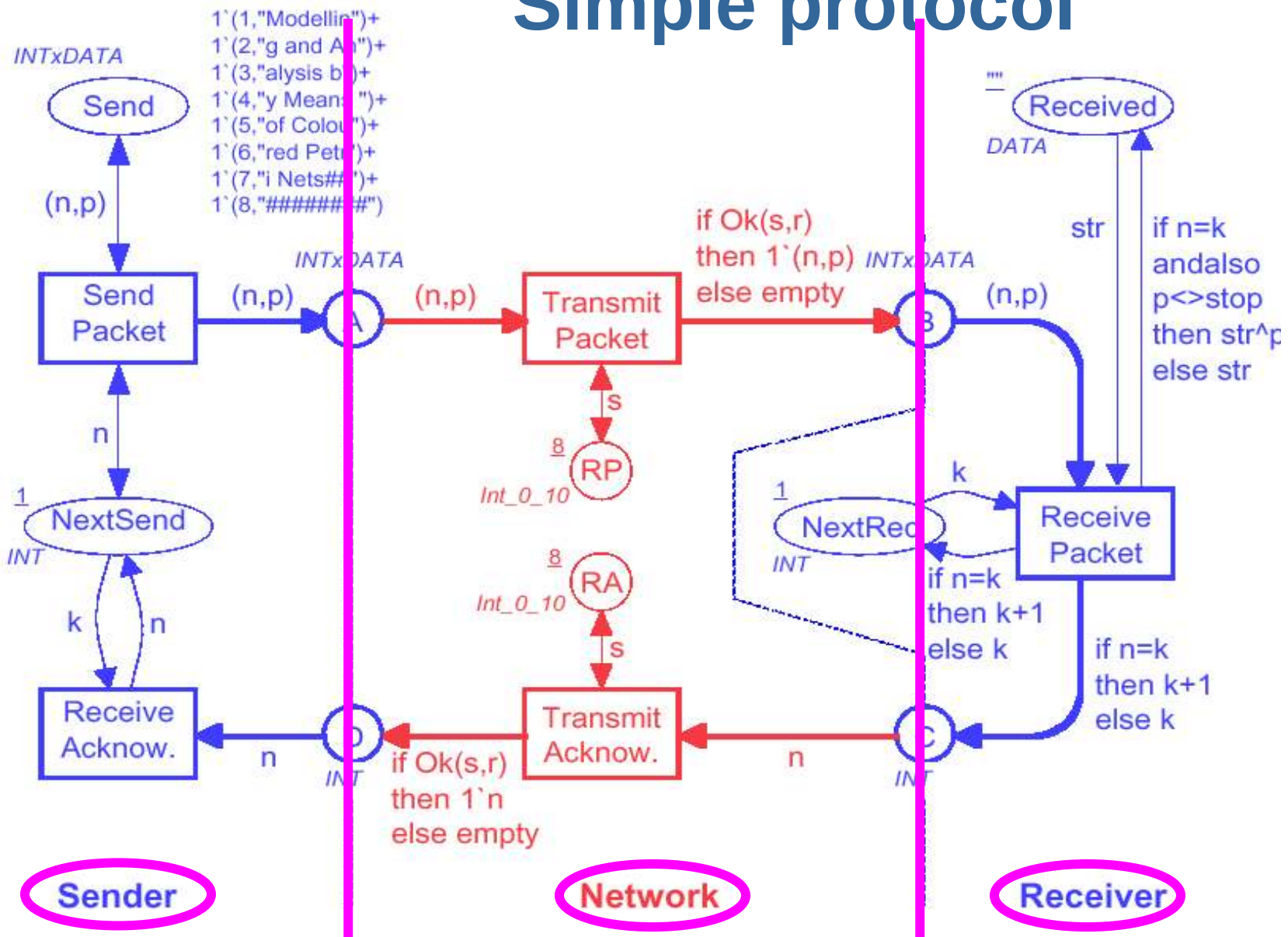
Número de fichas

cores

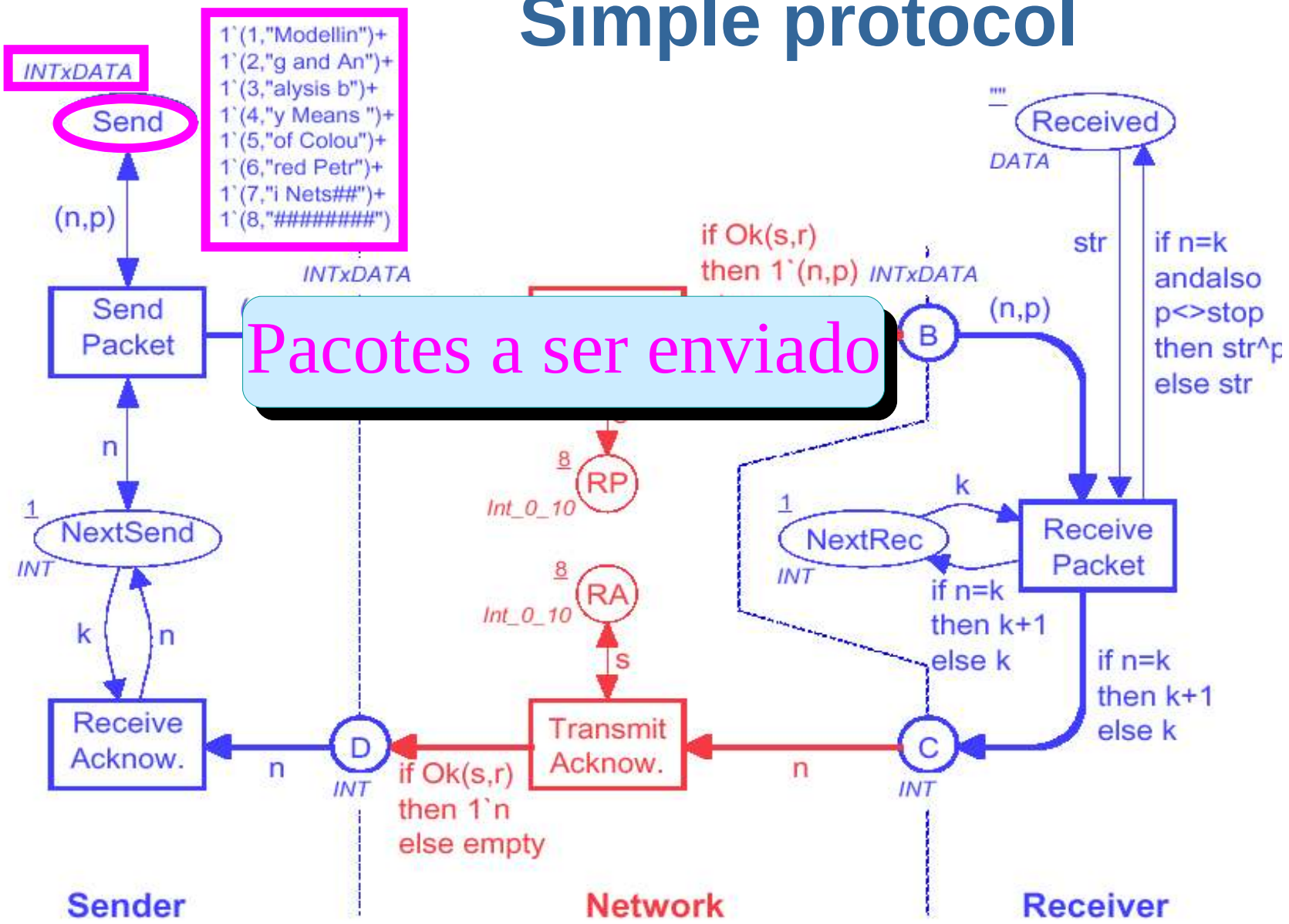
# Simple protocol



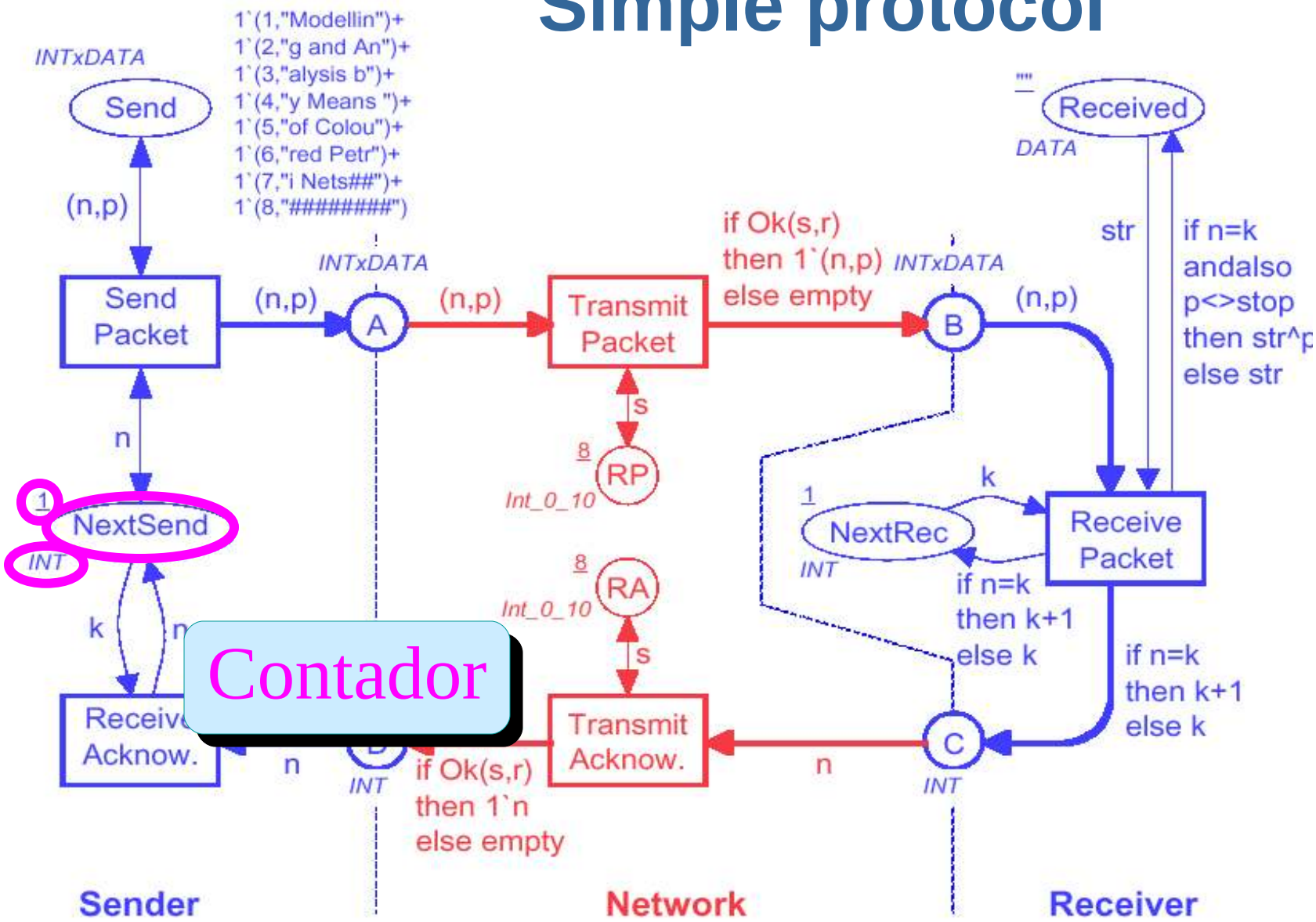
# Simple protocol



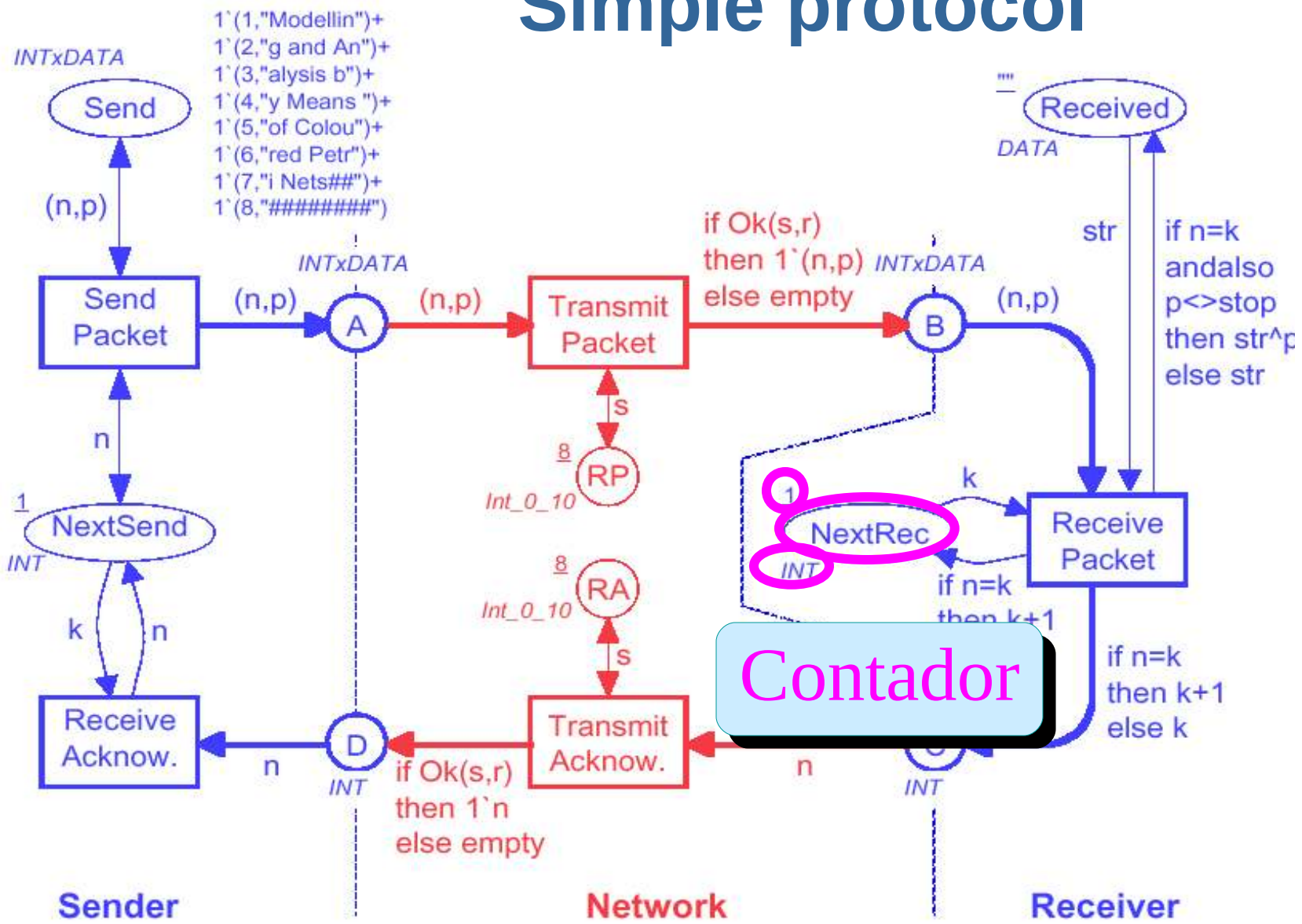
# Simple protocol



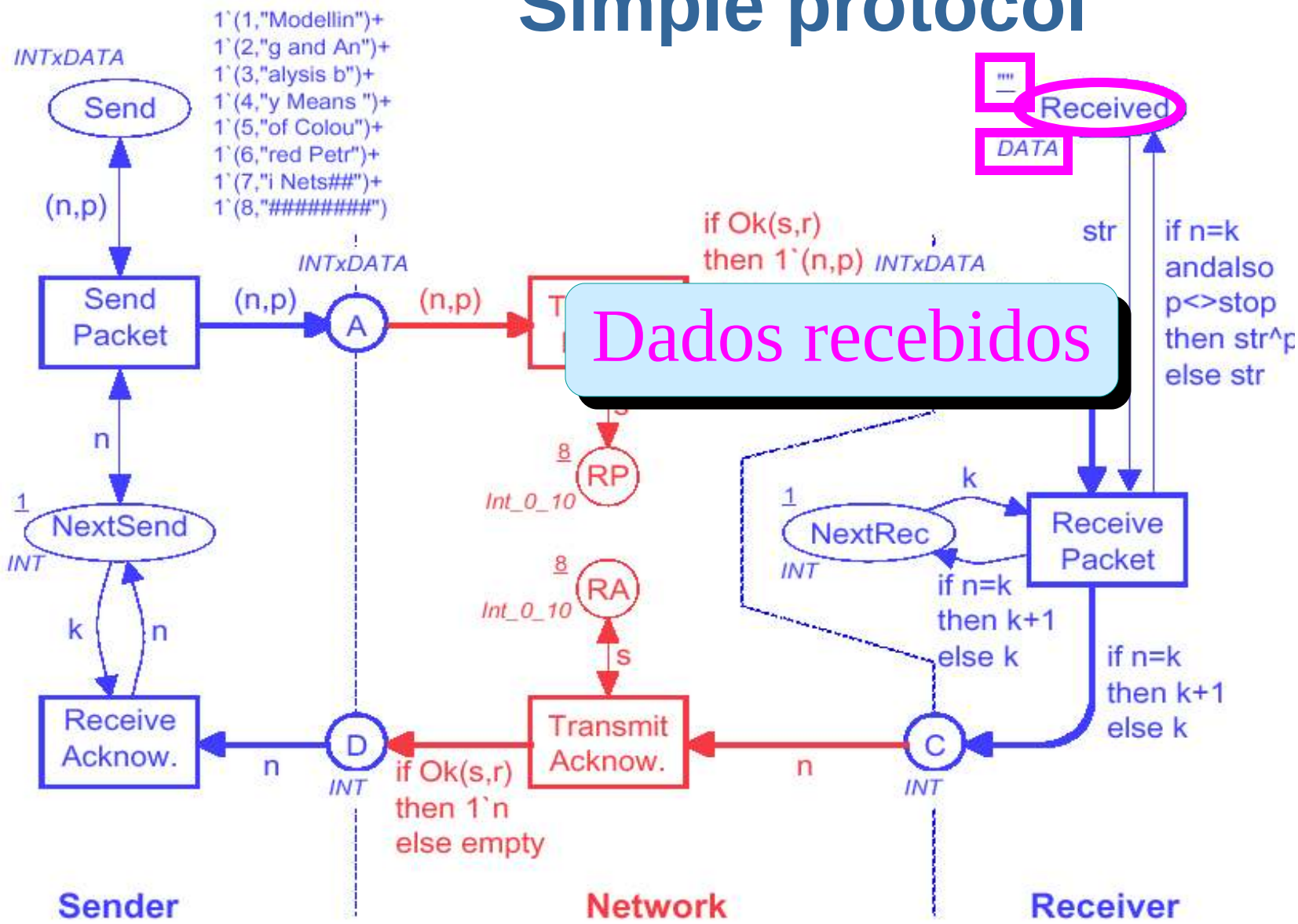
# Simple protocol



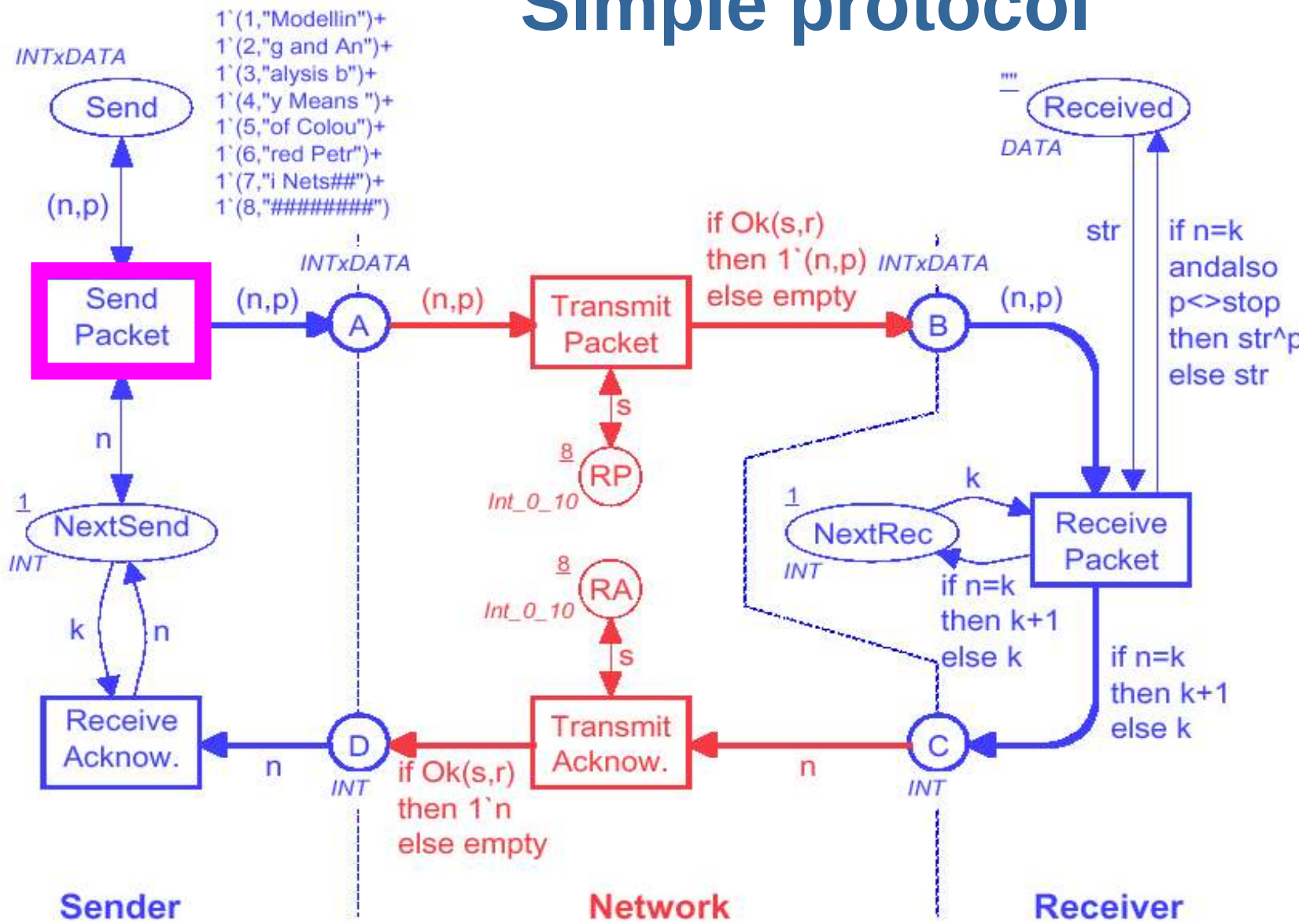
# Simple protocol



# Simple protocol

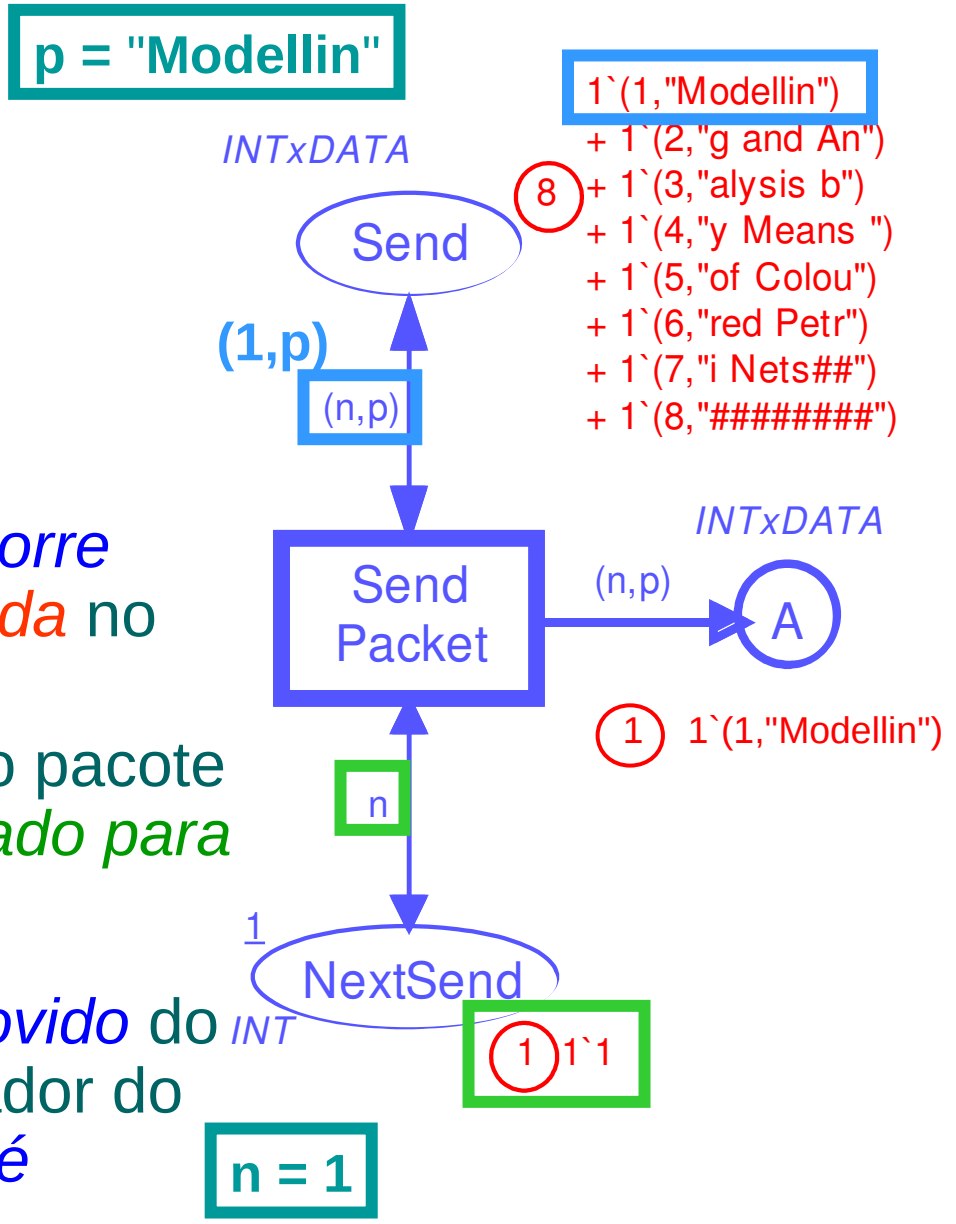


# Simple protocol

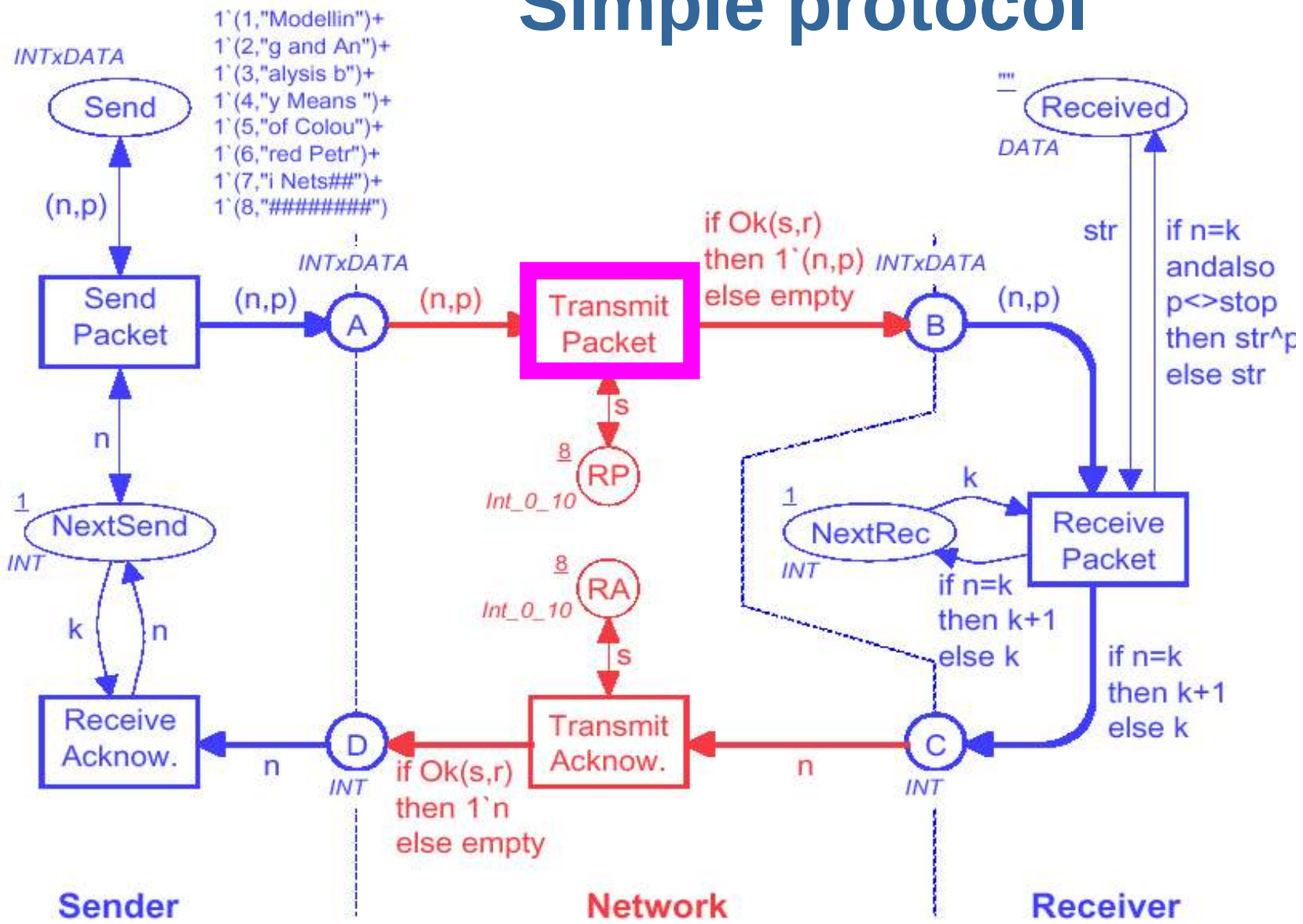


# Lugar Send

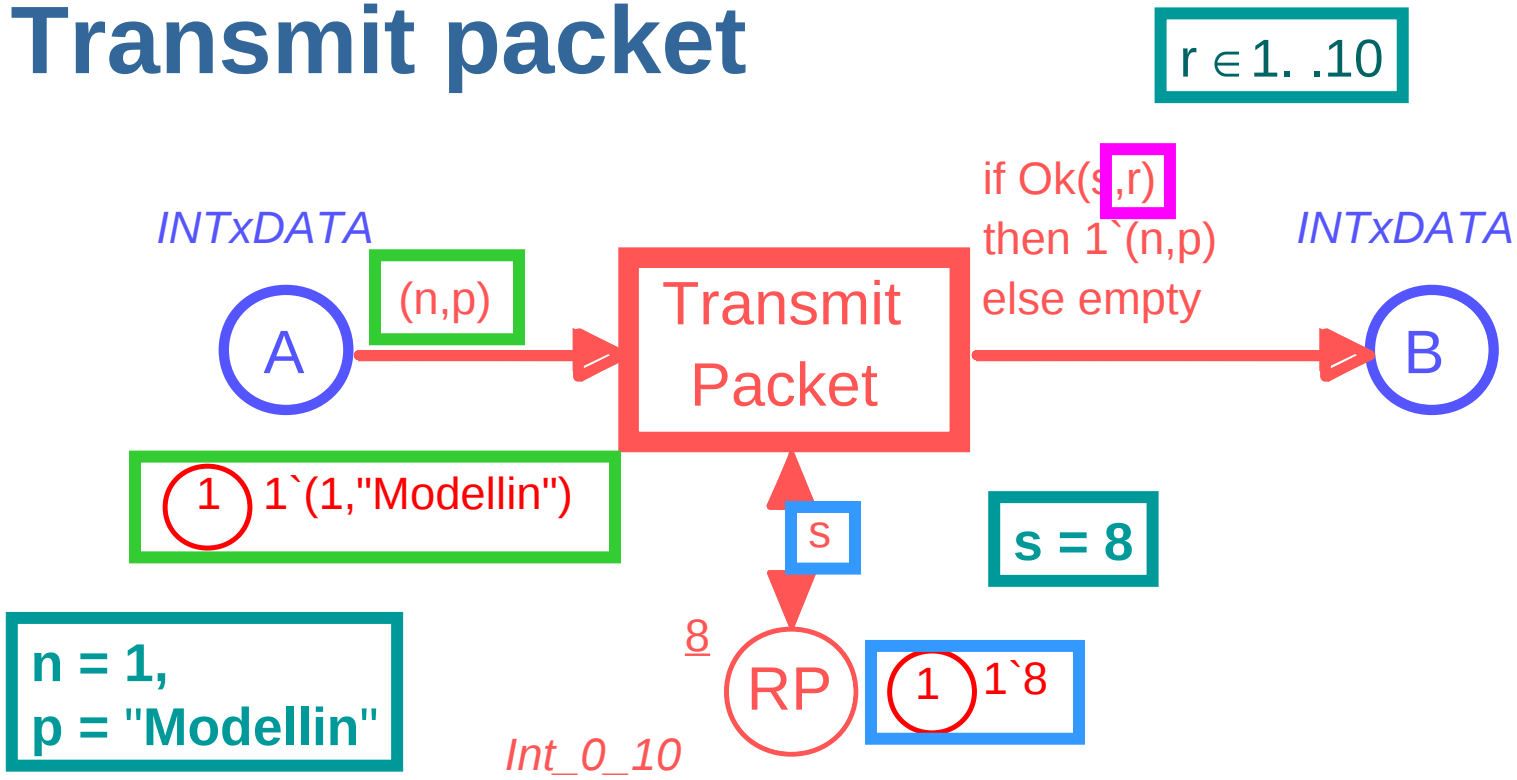
- A ligação <n=1,p="Modellin"> está *habilitada.*
- Quando a ligação *ocorre* *uma ficha é adicionada* no lugar A.
- Isto representa que o pacote (1,"Modellin") é *enviado para a rede.*
- O pacote *não é removido* do lugar *Send* e o contador do lugar *NextSend não é alterado.*



# Simple protocol



# Transmit packet



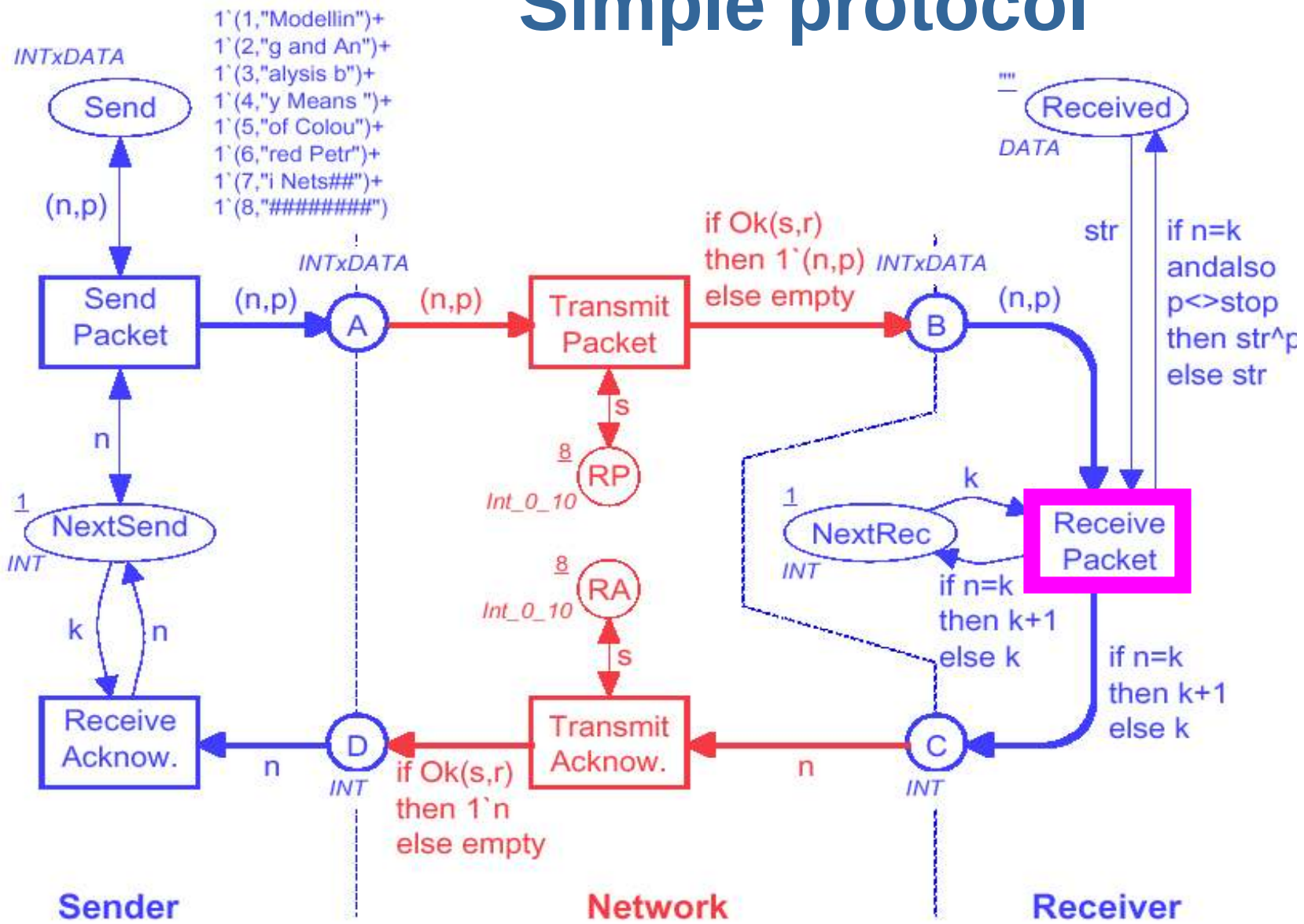

- Todas *ligações habilitadas* são:
  - <n=1,p= "Modellin",s=8,r=...>
  - onde r ∈ 1. .10

# Pacotes perdidos

```
if Ok(s,r)
then 1`(n,p)
else empty
```

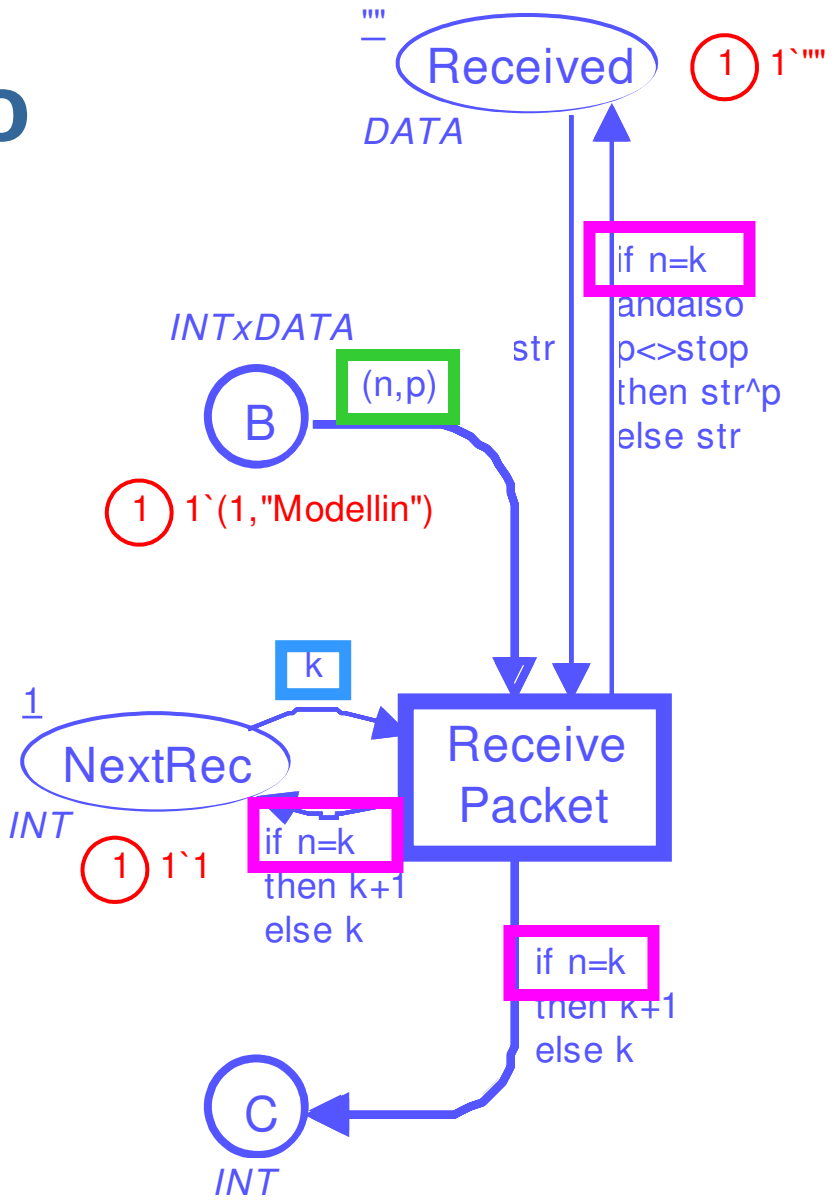
- A *função Ok(s,r)* checa se $r \leq s$.
  - *Para $r \in 1. .8$, Ok(s,r)=true.*
    A ficha é movida de A para B. Isto significa que o pacote é *transmitido com sucesso*.
  - *Para $r \in 9. .10$, Ok(s,r)=false.*
    Nenhuma ficha é adicionada em B. Isto significa que o pacote é *perdido.*
- O simulador CPN faz *escolhas aleatórias entre ligações*: 80% de chance para transferir com sucesso.

# Simple protocol



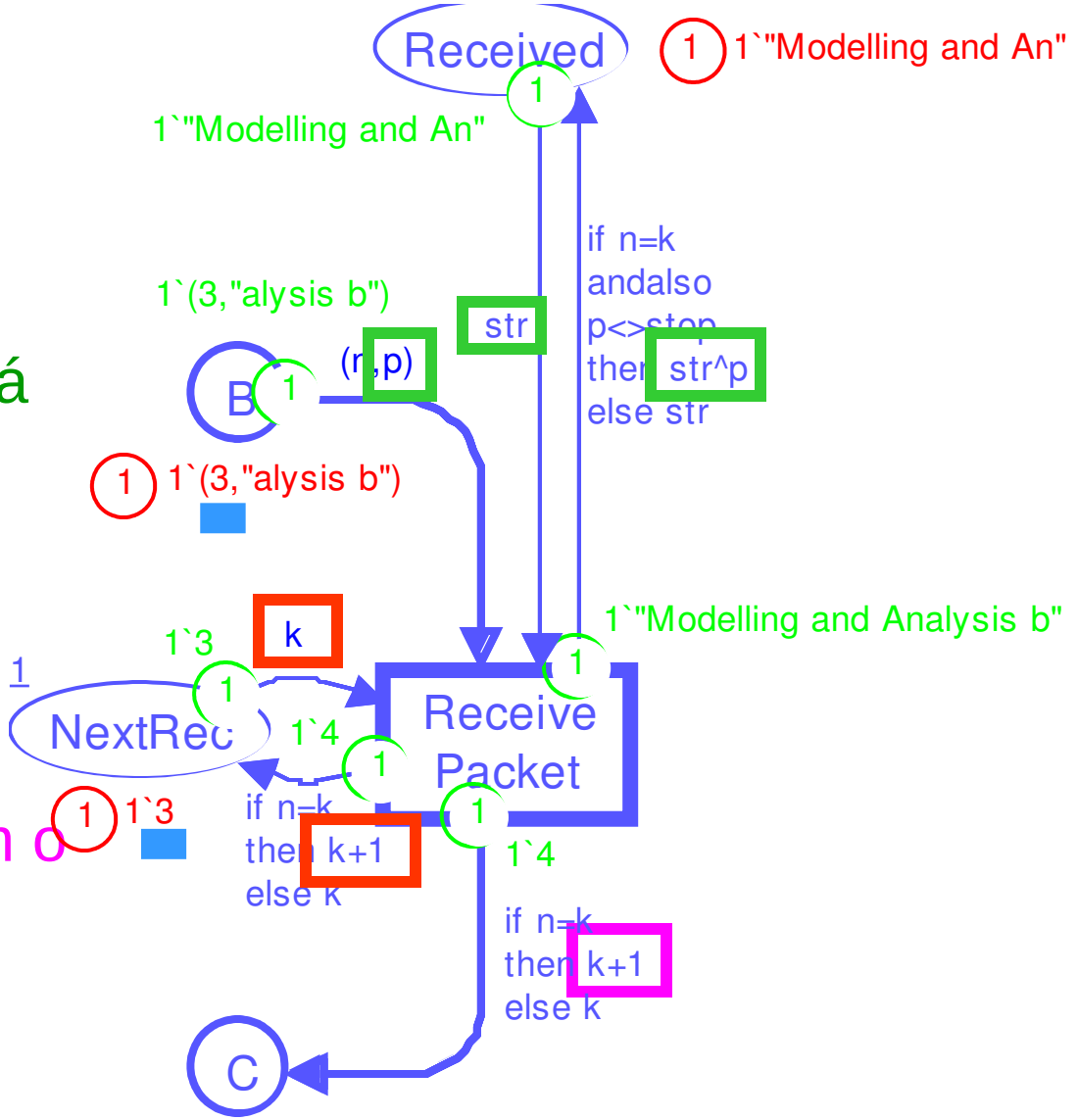
# Pacote Recebido

- O número do *pacote chegando* n
  e o número do *pacote esperado* k
  são *comparados.*



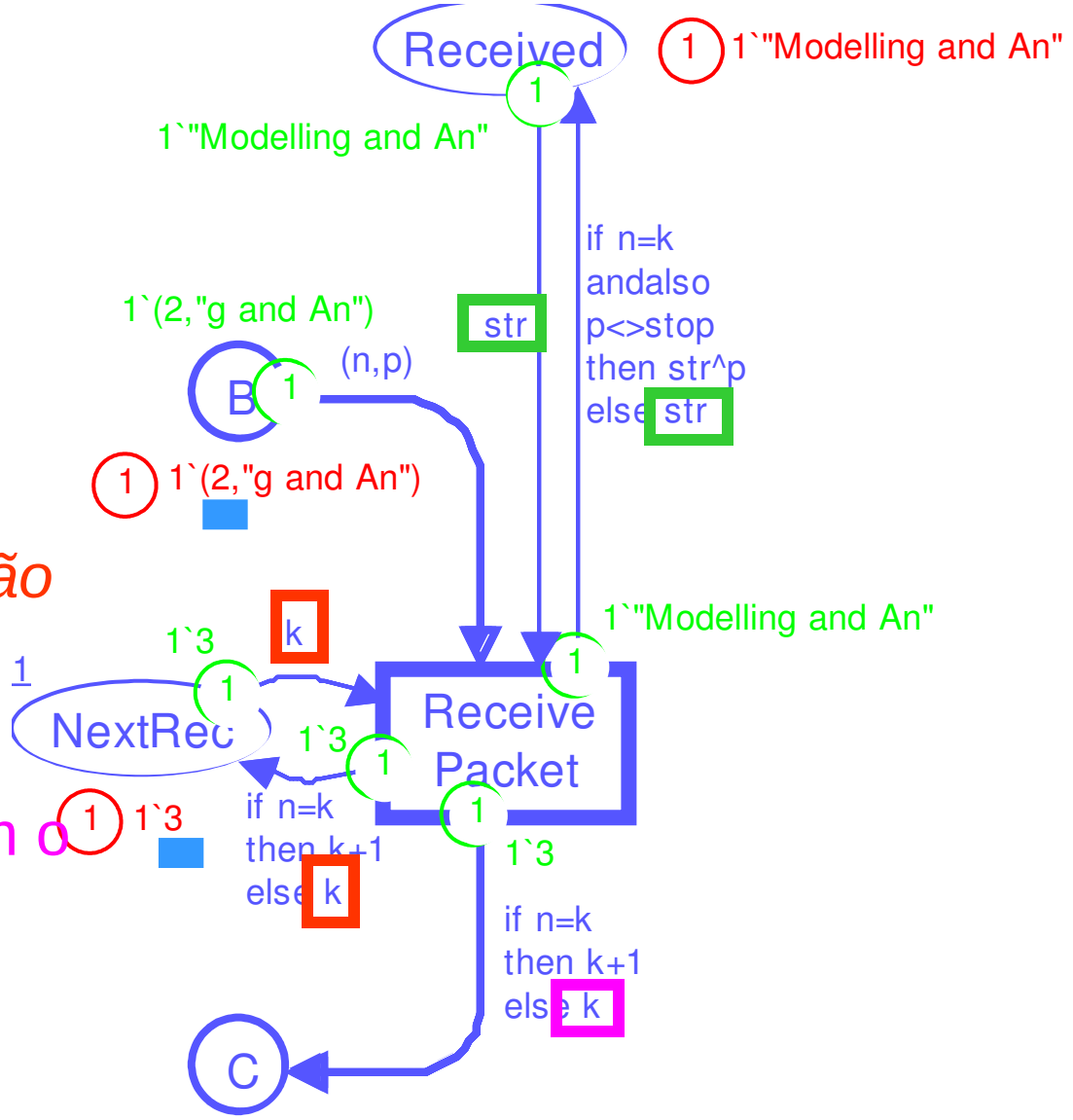
# Número do pacote correto

- O dado no pacote é *concatenado* ao dado já recebido.
- O contador *NextRec* é *incrementado em um.*
- Um reconhecimento é enviado. Este contém o número do *próximo pacote* a receber.

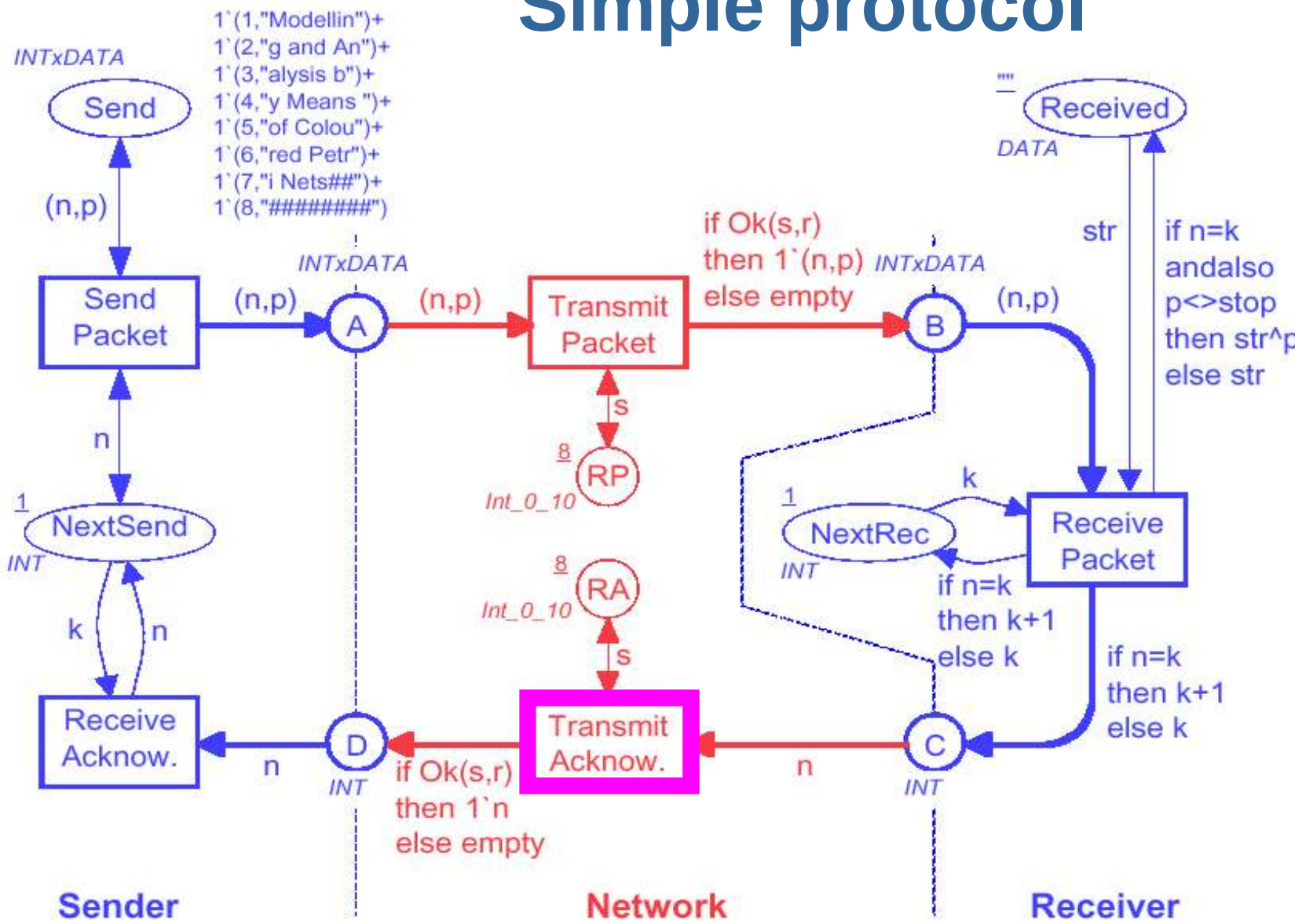


# Número do pacote errado

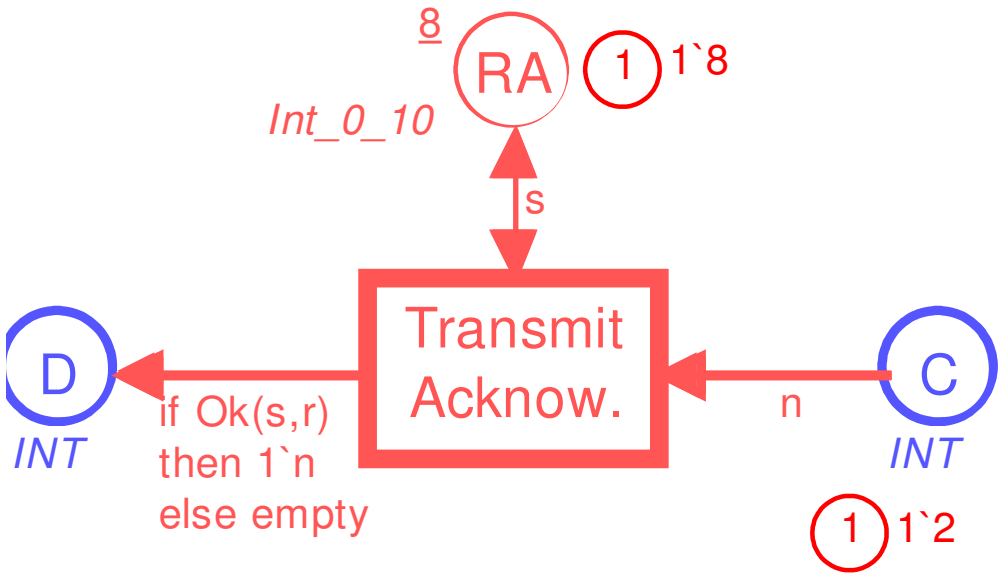
- O dado no pacote é *ignorado.*
- O contador *NextRec não é incrementado.*
- Um *reconhecimento* é enviado. Este contém o número do *próximo pacote* a receber.



# Simple protocol

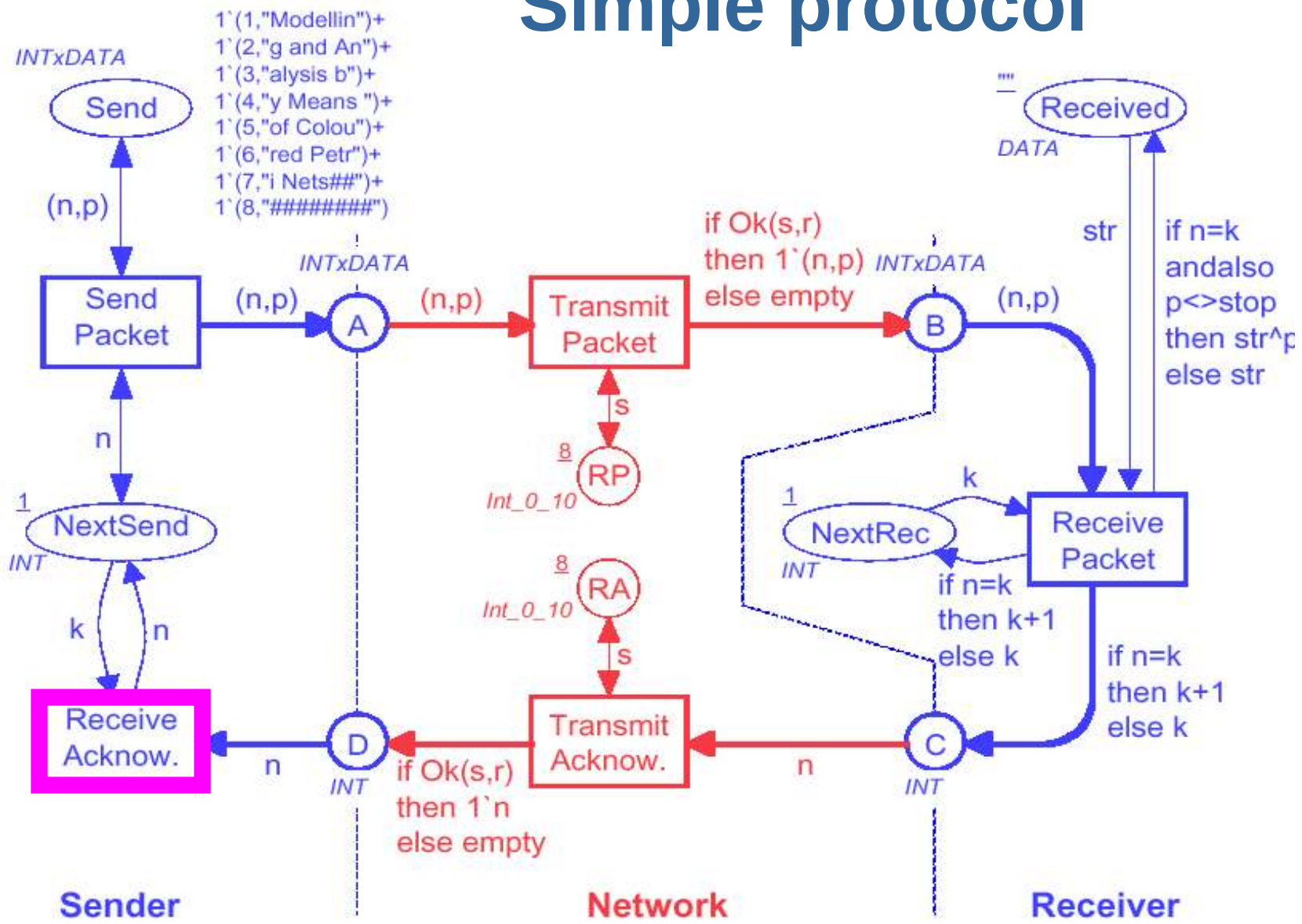


# Transmite reconhecimento

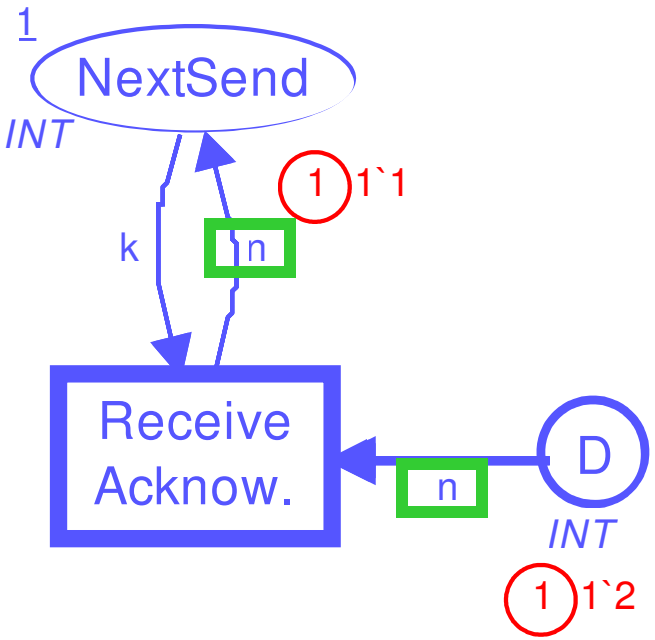



- Esta transição é similar a *Transmit Packet.*
- A marcação de *RA* determina o *sucesso do envio.*

# Simple protocol



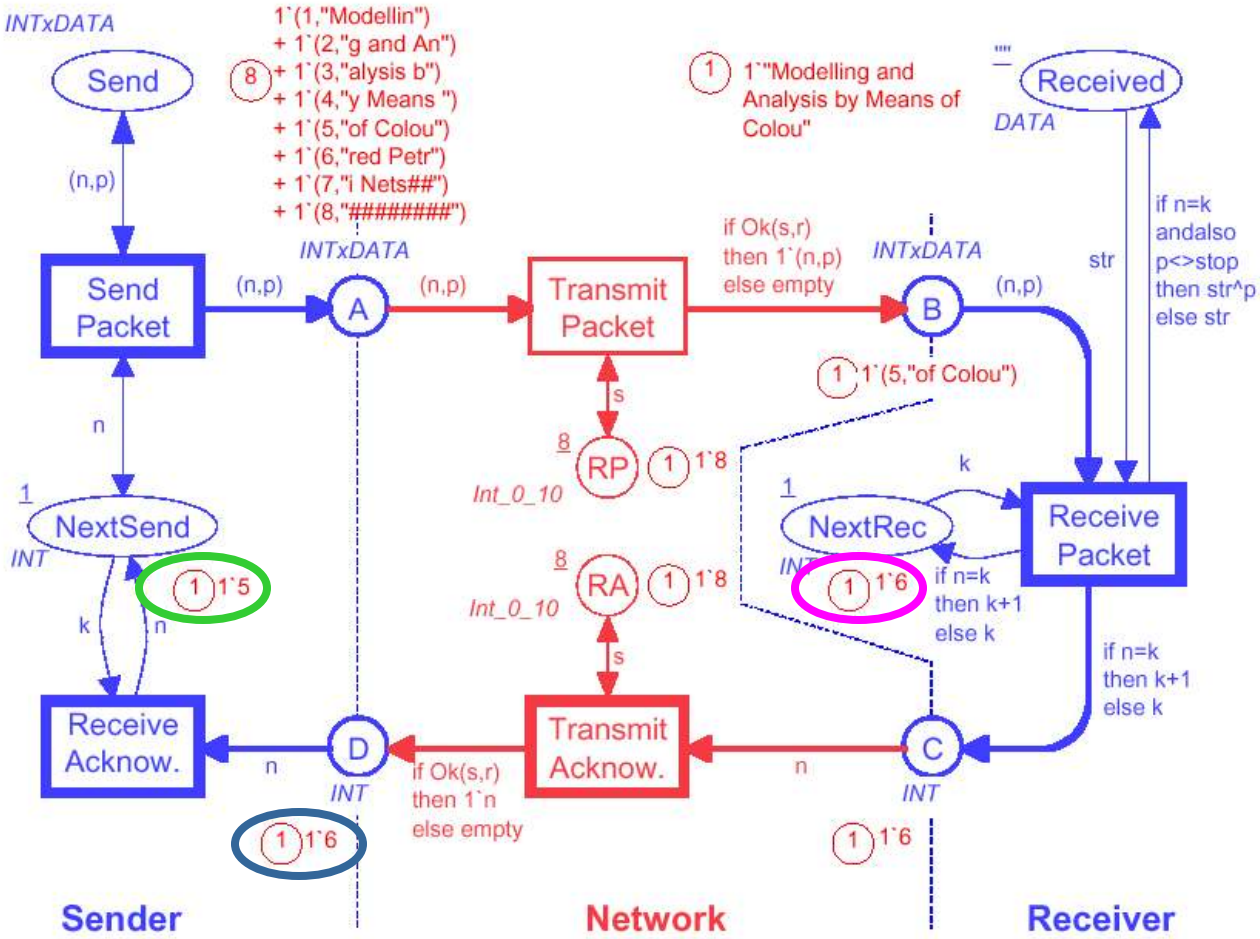
# Recebe reconhecimento




- Quando um reconhecimento chega para o *Sender* este é usado para atualizar o contador *NextSend*.
  - Neste caso o valor do contador se torna 2, e *Sender* enviará o *pacote número 2.*

# Estado intermediário

- *Receiver* aguarda pacote no. 6.
- *Sender* ainda envia pacote no. 5.
- Pacote no. 6 está chegando no reconhecimento.
- O *NextSend* é atualizado e *Sender* iniciará envio do pacote no. 6.



# CPN tem uma definição formal

- A existênca de uma *definição formal* é importante:
  - Base para *simulação*, i.e., execução da CPN.
  - Base para *verificação formal* (e.g., espaço de estados).
- *Não é necessário* um *usuário* conhecer a definição formal das CPN:
  - *Sintaxe* é checada pelo editor CPN.
  - *Semântica* é garantida pelo simulaor CPN e as ferramentas de verificação CPN.

# Rede de Petri Alto-Nível

- O relacionamento entre *CPN* e *redes de Petri lugar/transição* é *análogo* ao relacionamento entre *linguagem de programação de alto-nível* e *linguagem de máquina.*
  - Na *teoria,* os dois níveis tem exatamente o mesmo *poder computacional.*
  - na *pratica,* linguagens de alto-nível têm muito mais *poder de modelagem* – porque eles têm melhores estruturas, e.g., *tipos* e *módulos.*
- Muitos outros tipos de *redes de Petri de alto-nível* existem. No entanto, *redes de Petri coloridas* são as mais utilizadas – em particular para *trabalhos práticos.*

# Visão Geral

## Modelagem

- Linguagem básica
  - sintaxe
  - semântica
- **Extensões**
  - módulos
  - tempo
- Ferramenta de suporte
  - edição
  - simulação

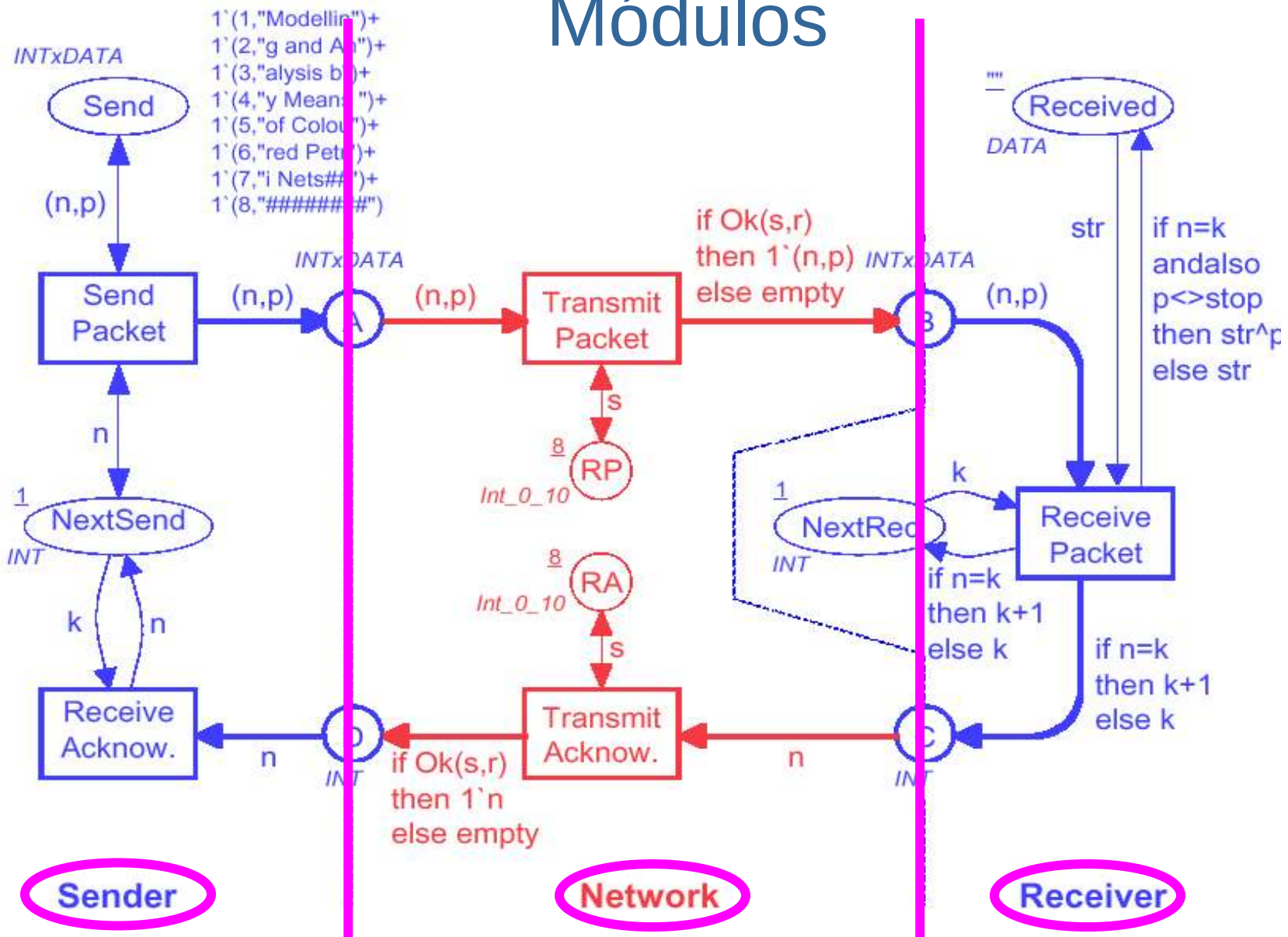
## Análise

- Espaço de Estado
  - completo
  - simetrias

# CPN são usadas para sistemas grandes

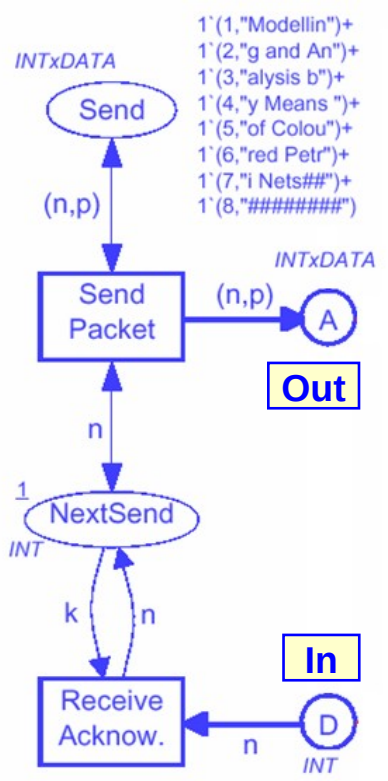
- Um modelo CPN consiste de vários *módulos*.
  - Também chamados *subredes* ou *páginas.*
  - *Interfaces* e *semânticas* bem-definidas.
- Uma *aplicação industrial* típica em CPN possui:
  - 10-200 módulos.
  - 50-1000 lugares e transições.
  - 10-200 tipos.
- Aplicações Industriais deste tamanho podem ser *totalmente impossível* sem:
  - Tipos de dados e valores de fichas.
  - Módulos.
  - Ferramenta de suporte.

# Módulos

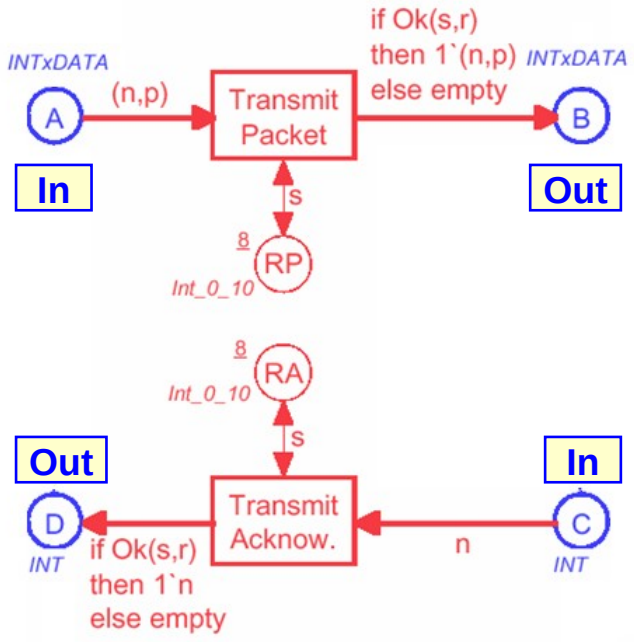

INTxDATA
Send
(n,p)
Send Packet
n
1
NextSend
INT
k
n
Receive Acknow.
n
INT
(n,p)
A
(n,p)
Transmit Packet
if Ok(s,r) then 1`(n,p) else empty
s
8
RP
Int_0_10
8
RA
Int_0_10
s
Transmit Acknow.
if Ok(s,r) then 1`n else empty
B
(n,p)
1
NextRec
INT
k
Receive Packet
if n=k then k+1 else k
if n=k then k+1 else k
C
n
INT
""
Received
DATA
str
if n=k andalso p<>stop then str^p else str
Sender
Network
Receiver

# Três diferentes módulos

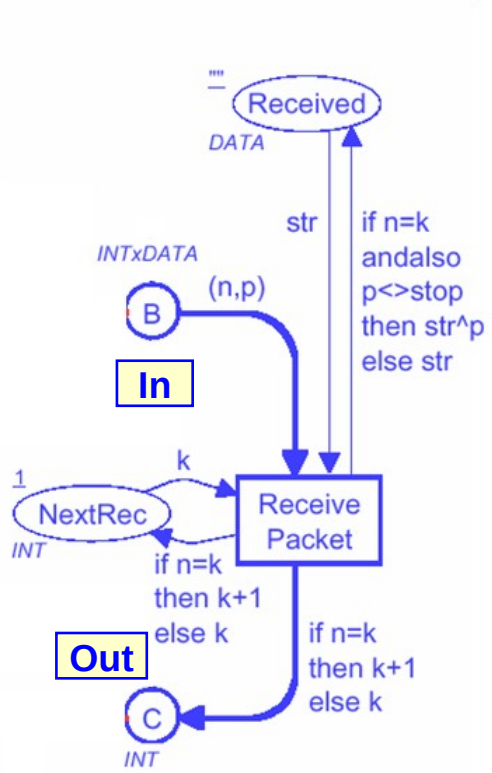
## Sender




## Network



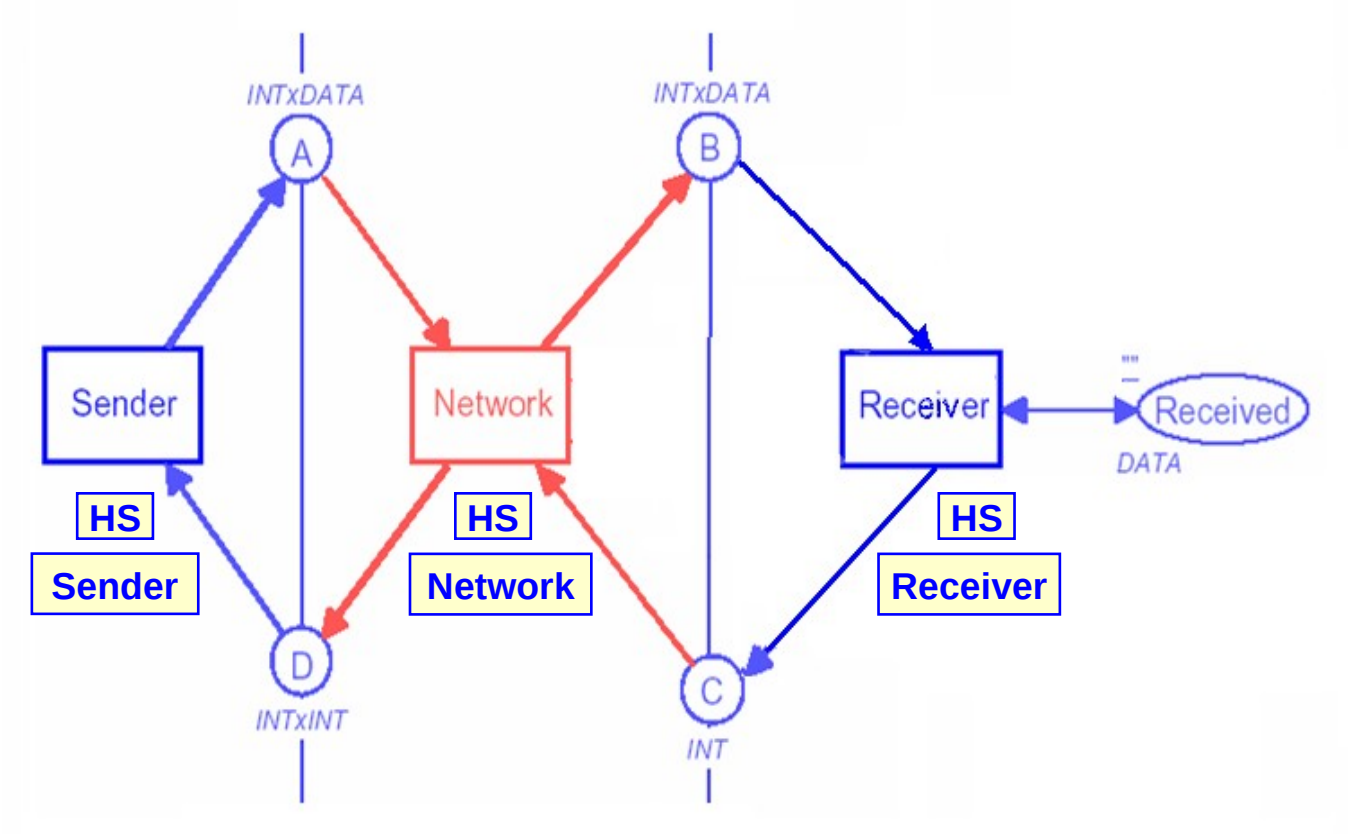

## Receiver




- *Lugares Porta* são usados para *mudar fichas* entre módulos.

# Visão Abstrata

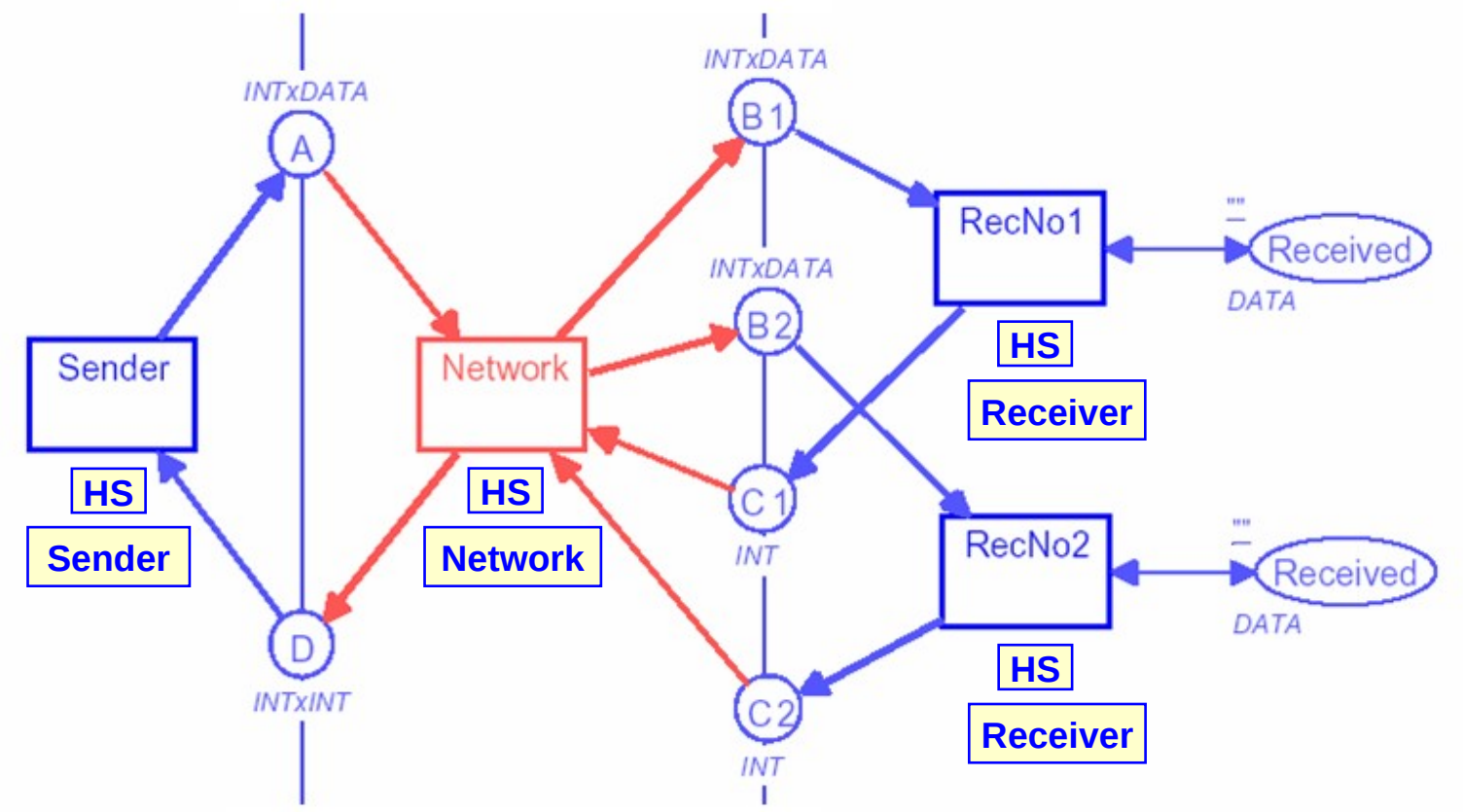
**Protocolo**




- *Transições de Substituição* referem-se aos *módulos.*
- *Lugares Socket* são relacionados aos *plugares porta.*

# Módulos podem ser reusados

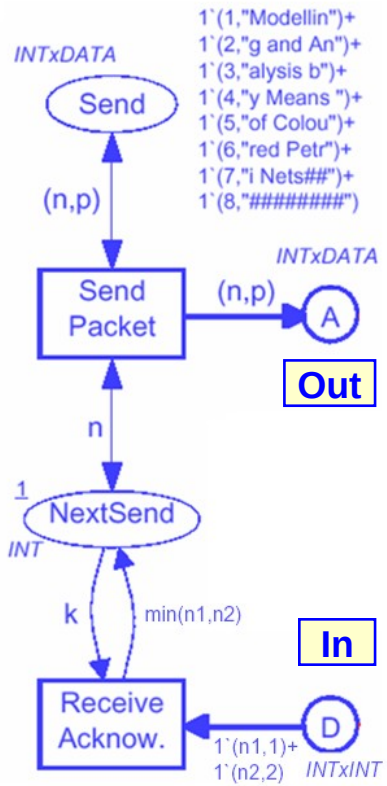
**Protocolo**

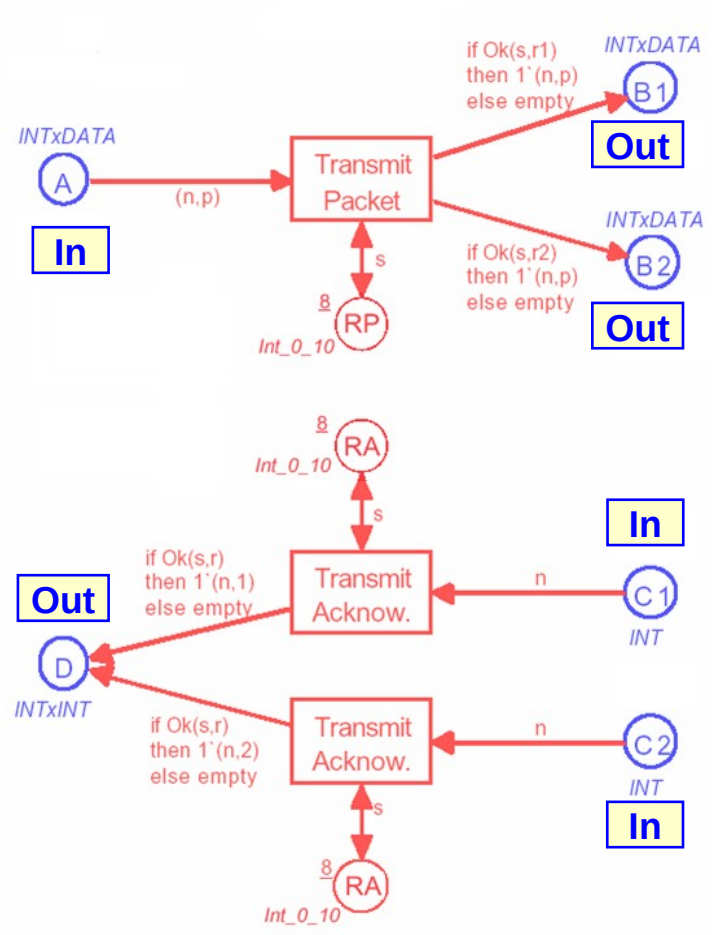


# Protocolo com vários receivers

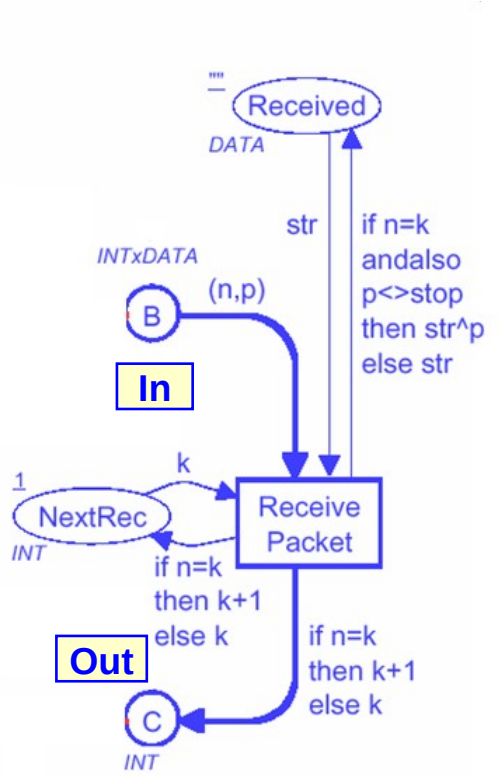
## Network

### Sender

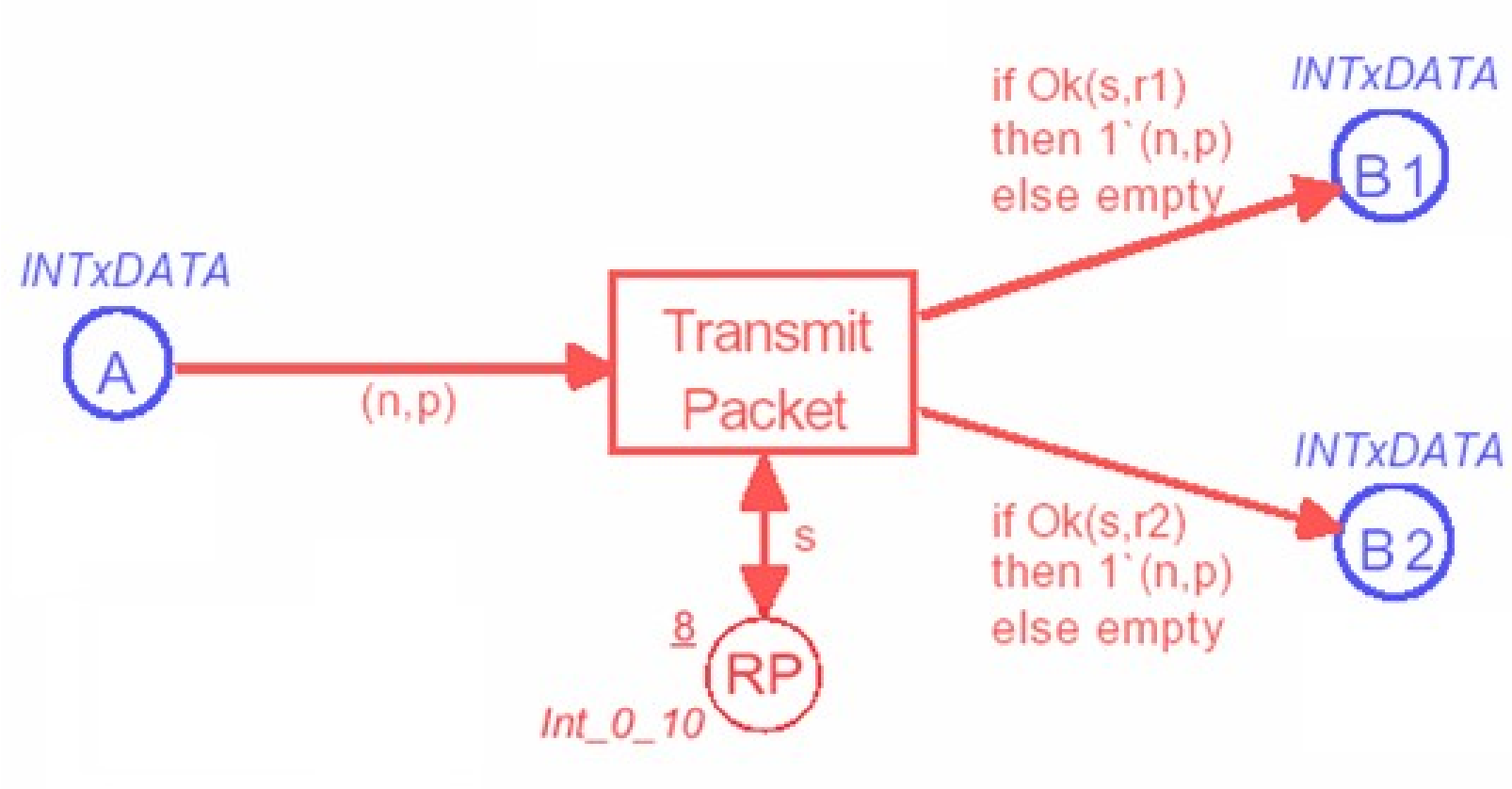



### Receiver

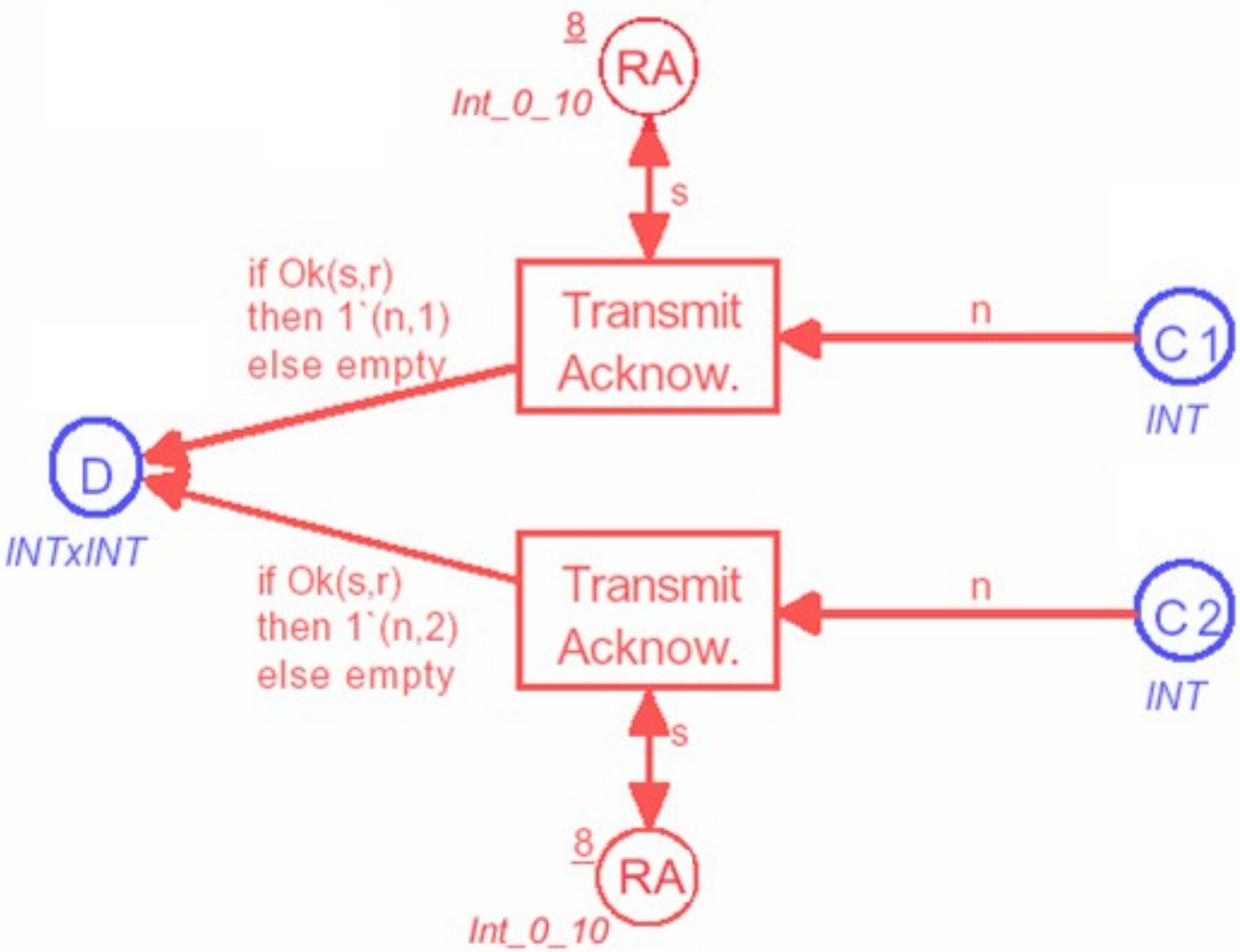


# Pacotes Transmitidos

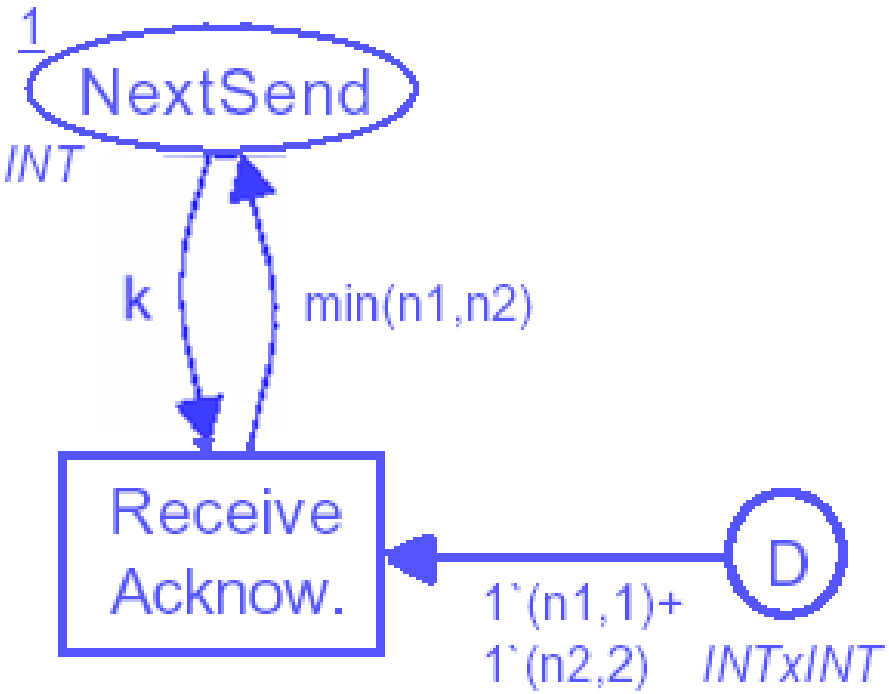



- Pacotes são *enviados* para os dois receivers.
  - *Alguns* dos pacotes podem *ser perdidos.*

# Reconhecimento transmitido



# Reconhecimento recebido



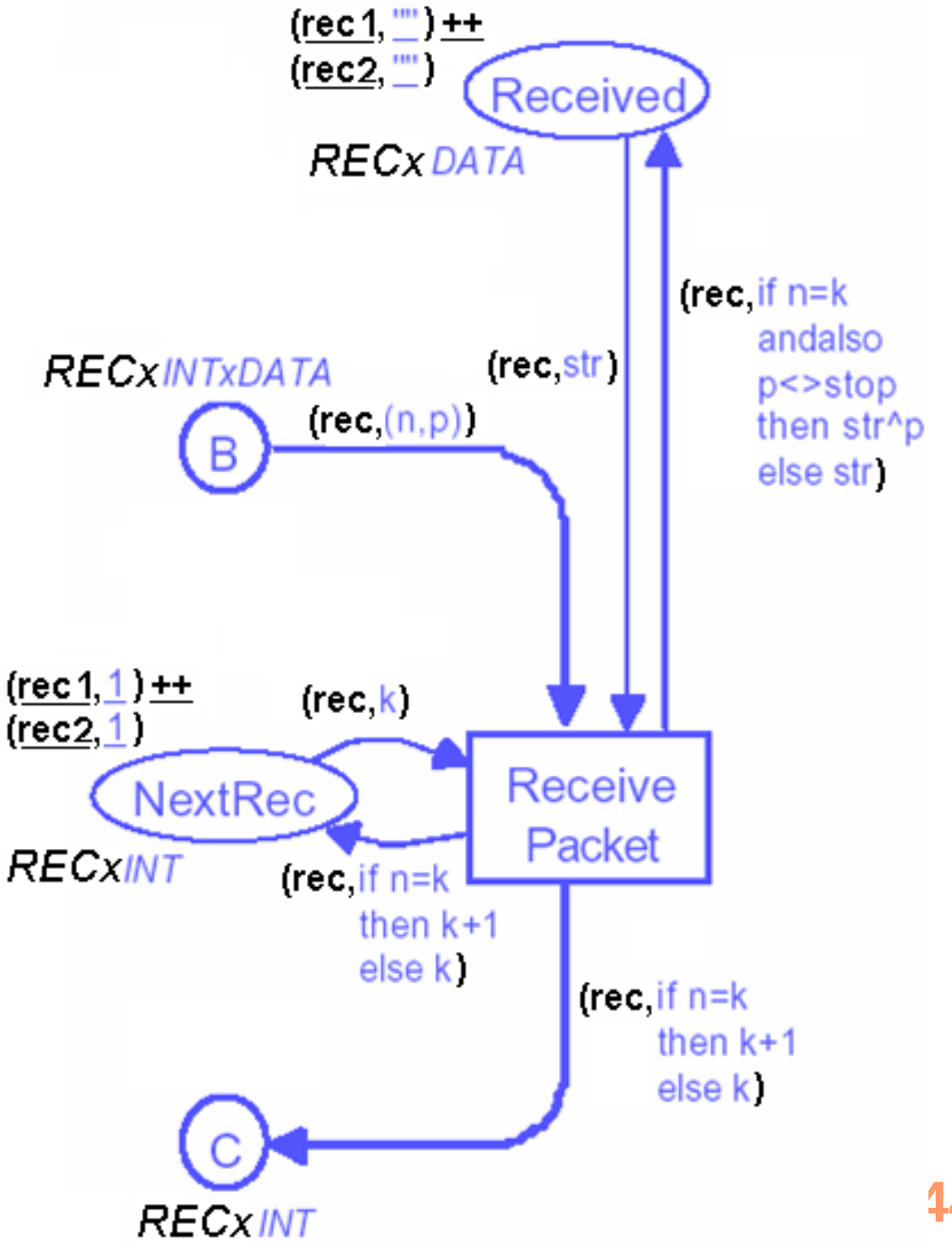

- O *sender* recebe o *menor* inteiro.

# Descrições Hierárquicas

- Nós usamos *módulos* para descrições de *estruturas grandes* e *complexas*.
- Módulos nos possibilita *ocultar detalhes* que não queremos considerar em um certo *nível de abstração*.
- Módulos possuem *interfaces bem-definidas*, consistem de *lugares socket* e *porta*, através do qual os módulos *trocam fichas* com outros.
- Módulos podem ser *reusados*.

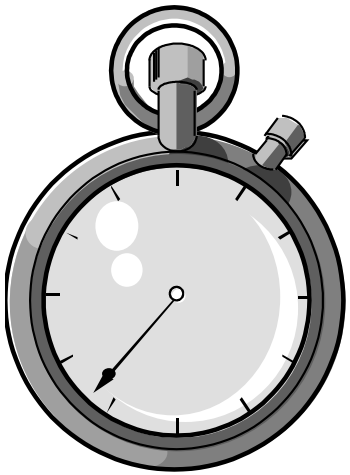
# Outra solução

- *Múltiplos receivers* podem também ser modelados adicionando um *novo componente* à *cor da ficha.*
- Mudanças similares devem ser feitas para a *transmição do pacote* e *transmição do reconhecimento.*



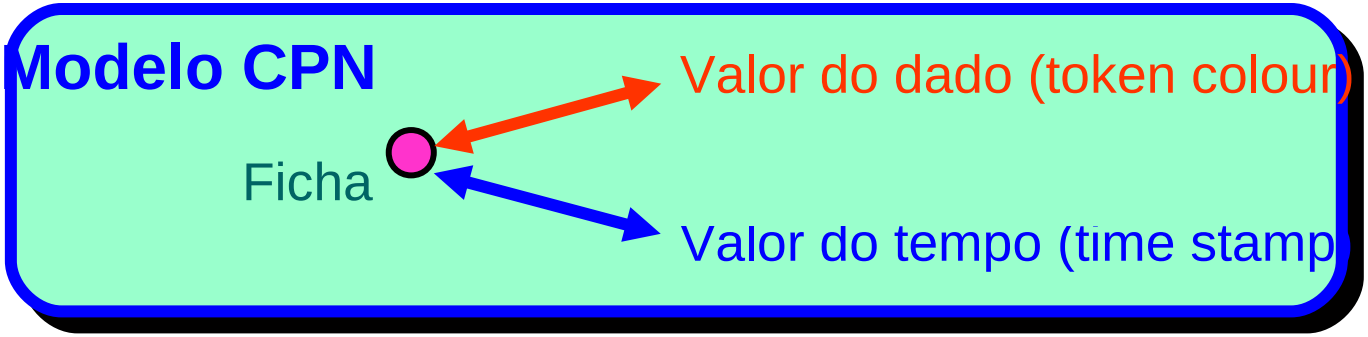
# Análise de Tempo

- CPN podem considerar o *conceito de tempo.* Isto significa que a *mesma linguagem de modelagem* pode ser usada para investigar:
  - *Corretude lógica.*
    *Funcionalidade desejada, sem deadlocks*, etc.
  - *Desempenho*.
    Quão rápido é o sistema e quantos recursos são usados.



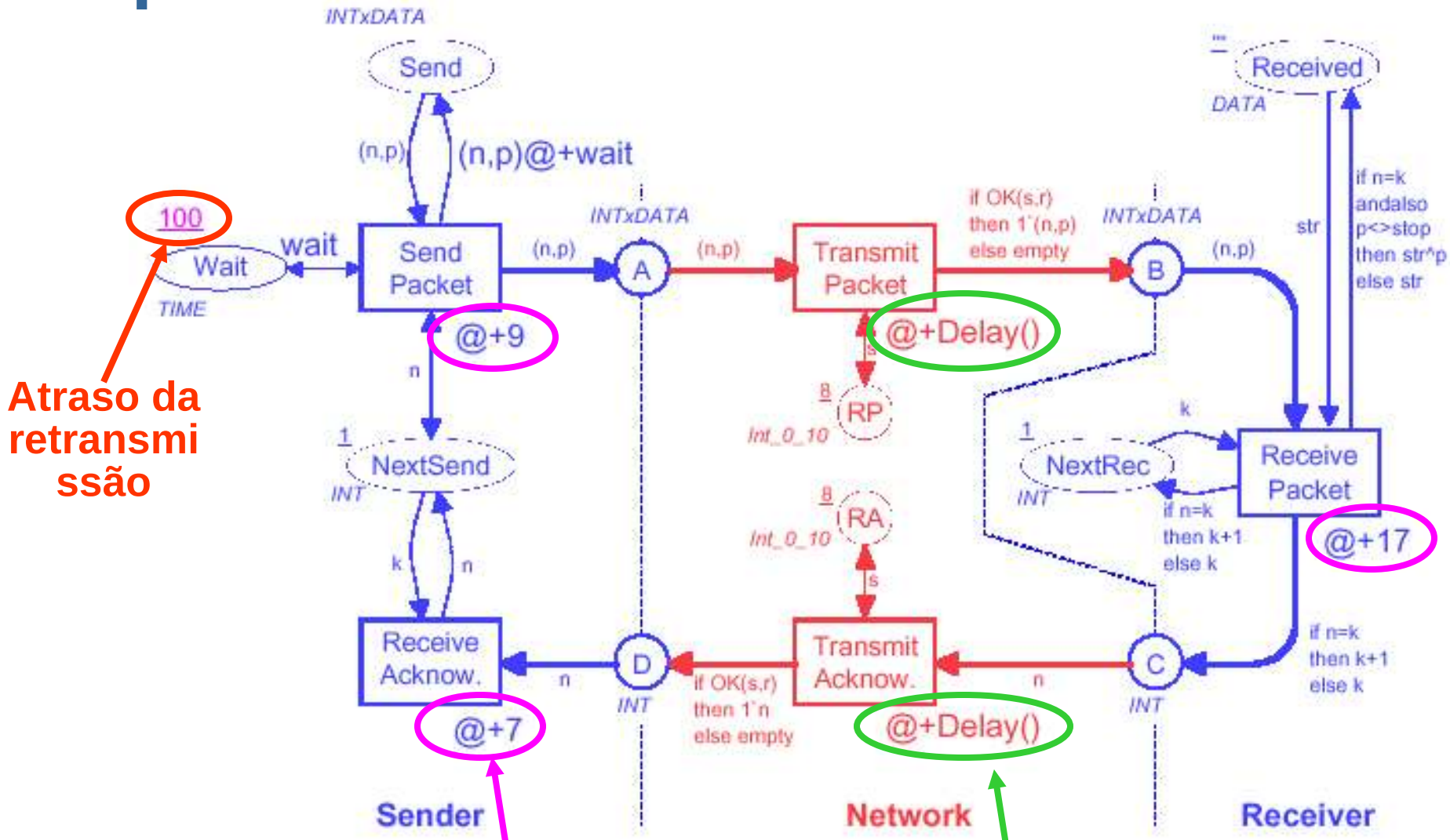
# Como adicionar tempo

- *Tempo* é adicionado em *modelos de redes de Petri* de várias formas – tipicamente especificando *atrasos* em *lugares* ou *transições*.




- *Time stamp* determina *quando* a ficha pode ser usada, i.e., *consumida por uma transição.*
  - *Atrasos* podem ser *fixados.*
  - Determinados por uma *distribuição arbitraria*.

# A CPN temporizada para o protocolo



# Áreas de Aplicações

## Protocolos and Redes

- Intelligent Networks at Deutsche Telekom
- IEEE 802.6 Configuration Control at Telstra Research Labs
- Allocation Policies in the Fieldbus Protocol in Japan
- ISDN Services at Telstra Research Laboratories
- Protocol for an Audio/Video System at Bang & Olufsen
- TCP Protocols at Hewlett-Packard
- Local Area Network at University of Las Palmas
- UPC Algorithms in ATM Networks at University of Aarhus
- BRI Protocol in ISDN Networks
- Network Management System at RC International A/S
- Interprocess Communication in Pool IDA at King's College

## Software

- Mobile Phones at Nokia
- Bank Transactions & Interconnect Fabric at Hewlett-Packard
- Mutual Exclusion Algorithm at University of Aarhus
- Distributed Program Execution at University of Aarhus
- Internet Cache at the Hungarian Academy of Science
- Electronic Funds Transfer in the US
- Document Storage System at Bull AG
- ADA Program at Draper Laboratories

## Controle de Sistemas

- Security and Access Control Systems at Dalcotech A/S
- Mechatronic Systems in Cars at Peugeot-Citroën in France
- European Train Control System in Germany
- Flowmeter System at Danfoss
- Traffic Signals in Brazil
- Chemical Production in Germany
- Model Train System at University of Kiel

## Hardware

- Superscalar Processor Architectures at Univ. of Newcastle
- VLSI Chip in the US
- Arbiter Cascade at Meta Software Corp.

## Sistemas militares

- Military Communications Gateway in Australia
- Influence Nets for the US Air Force
- Missile Simulator in Australia
- Naval Command and Control System in Canada

## Outros Sistemas

- Bank Courier Network at Shawmut National Coop.
- Nuclear Waste Management Programme in the US

# Visão Geral

## Modelagem

- Linguagem básica
  - sintaxe
  - semântica
- Extensões
  - módulos
  - tempo
- **Ferramenta de suporte**
  - edição
  - simulação

## Análise

- Espaço de Estado
  - completo
  - simetrias

# Ferramentas computacionais

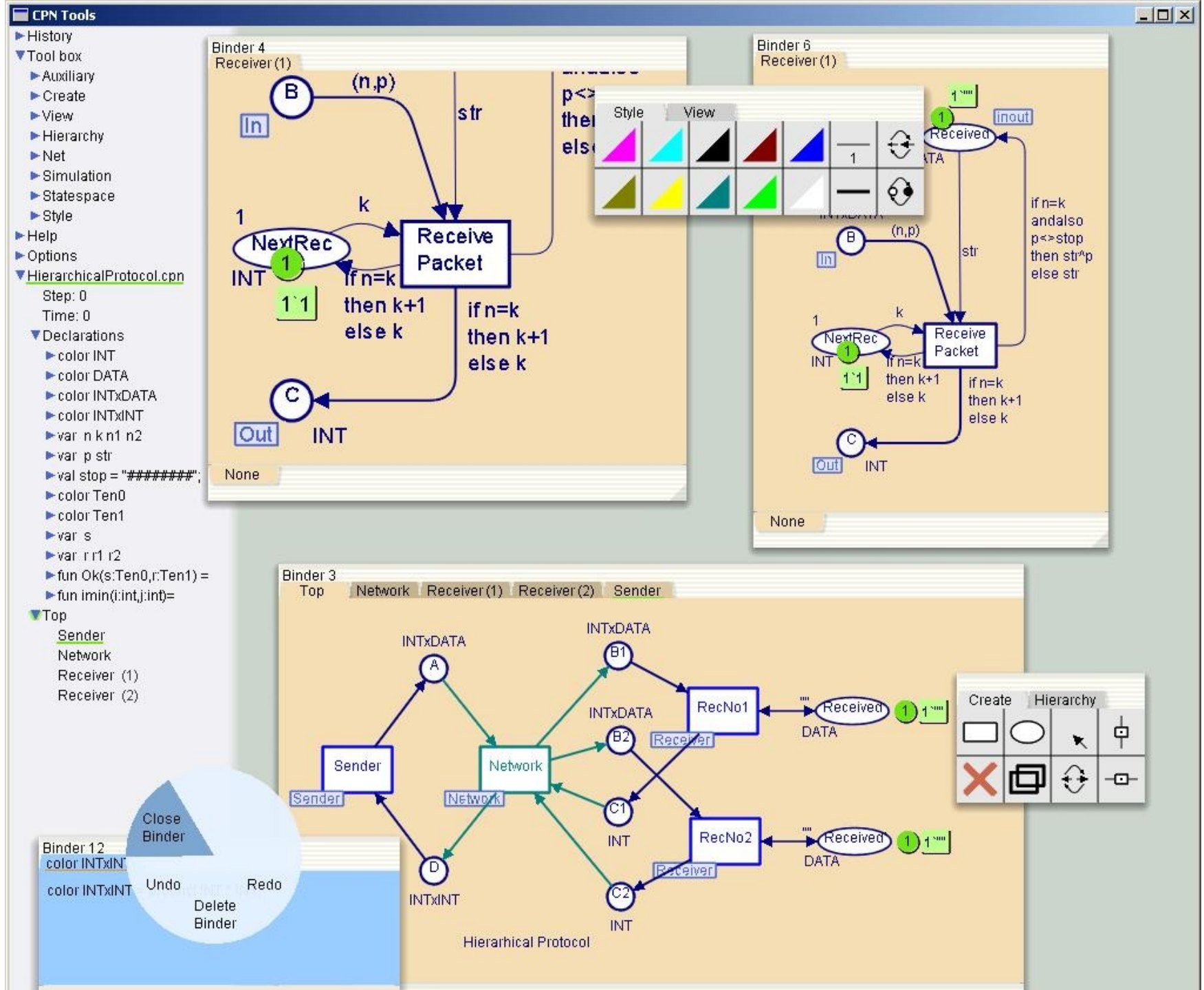
- *Design/CPN* foi desenvolvida nos anos 80e início dos anos 90.
  - Até recentemente, este foi o pacote de redes de Petri *mais usado*.
  - Usado por *1000 diferentes organizações* em mais de *60 países* – incluindo *200 empresas.*

- *CPN Tools* é a *próxima geração* de ferramentas de suporte para redes de Petri Coloridas.
  - Este tem *substituído* *Design/CPN* com *4000 licenças* em *120 países.*
  - Desenvolvimento *iniciado em 1999* com um total de *25 homens-ano*
  - Desenvolvimento *continua* com uma expectativa de esforço de *3-4 homens-ano*.

# CPN Tools e Design/CPN

A *funcionalidade* das duas ferramentas é a *mesma:*

- *Edicão* e *syntax check* de CPN.
- *Simulação Interativa* e *automática*.
- *Construção* e *análise* de *espaço de estados.*
- *Comunicação* com outras ferramentas.
- Simulation based *performance analysis.*
- *Animação Gráfica* do resultado das simulações.


CPN Tools
History
Tool box
Auxiliary
Create
View
Hierarchy
Net
Simulation
Statespace
Style
Help
Options
HierarchicalProtocol.cpn
Step: 0
Time: 0
Declarations
color INT
color DATA
color INTxDATA
color INTxINT
var n k n1 n2
var p str
val stop = "########";
color Ten0
color Ten1
var s
var r r1 r2
fun Ok(s:Ten0,r:Ten1) =
fun imin(i:int,j:int)=
Top
Sender
Network
Receiver (1)
Receiver (2)
Binder 4
Receiver (1)
B
In
(n,p)
str
k
1
NextRec
INT
1`1
if n=k
then k+1
else k
Receive
Packet
if n=k
then k+1
else k
C
Out
INT
None
Style
View
Binder 6
Receiver (1)
1`""
1
Received
inout
if n=k
andalso
p<>stop
then str^p
else str
Binder 3
Top
Network
Receiver (1)
Receiver (2)
Sender
INTxDATA
A
B1
B2
Sender
Network
RecNo1
RecNo2
Receiver
Received
DATA
C1
C2
INT
D
INTxINT
Hierarhical Protocol
Create
Hierarchy
Binder 12
color INTxINT
Close
Binder
Undo
Redo
Delete
Binder

# Standard ML

- Tipos, expressões de arco e guardas são especificadas em *Standard ML*, que é uma linguagem fortemente tipada, funcional e desenvolvida por *Robin Milner*.
- *Tipos de dados* podem ser:
  - *Atômicos* (inteiros, strings, booleanos).
  - *Estruturados* (produts, records, unions, lists, e subsets).
- *Funções* e *operações* podem ser definidas.
- Standard ML é bem conhecida, bem testada e muito geral. Muitos *livros* são disponíveis.