

Modellezés színezett Petri hálókkal

THEORY

- models
- basic concepts
- analysis methods

TOOLS

- editing
- simulation
- verification

PRACTICAL USE

- specification
- validation
- verification
- implementation

© **Kurt Jensen** (részlet)

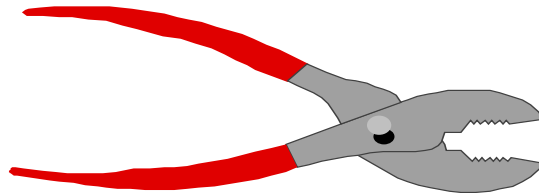
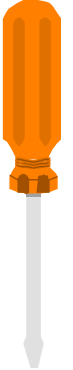
Department of Computer Science
Aarhus Egyetem, Dánia

kjensen@daimi.au.dk

Fordította: Erdélyi Árpád

Modellező eszközök

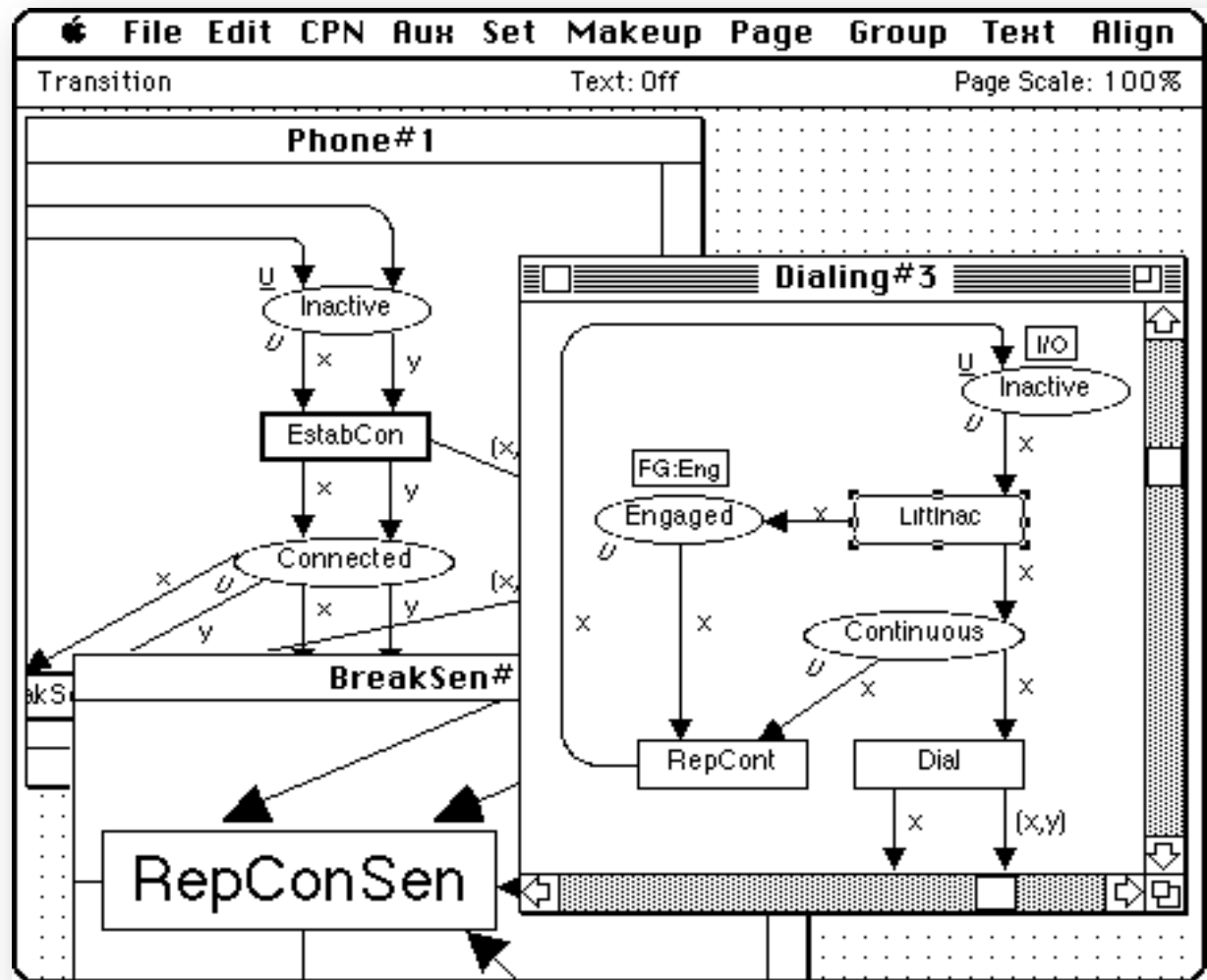
- ◆ *Design/CPN* eszközt a 80'as évek végén és a 90'as évek elején fejlesztették ki.
 - A maga idejében a legelterjedtebb Petri háló csomag volt.
 - *750 különböző szervezet* 50 országban használta – ezek között *200 kereskedelmi cég* van.
- ◆ *CPN Tools* már egy második generációs eszköz a Színezett Petri-hálókhöz.
 - Mára a CPN Tools *átvette a Design/CPN helyét*
 - A fejlesztés *1999-ben kezdődött* és több, mint *20 mérnökév* munka van benne.



Standard ML

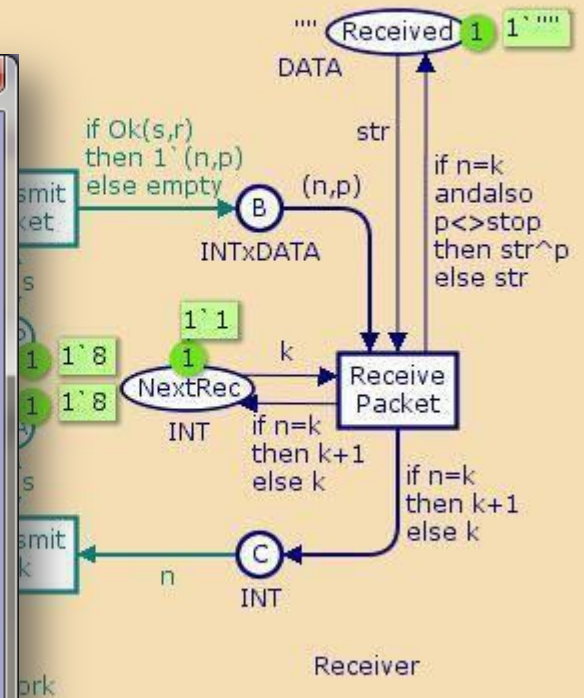
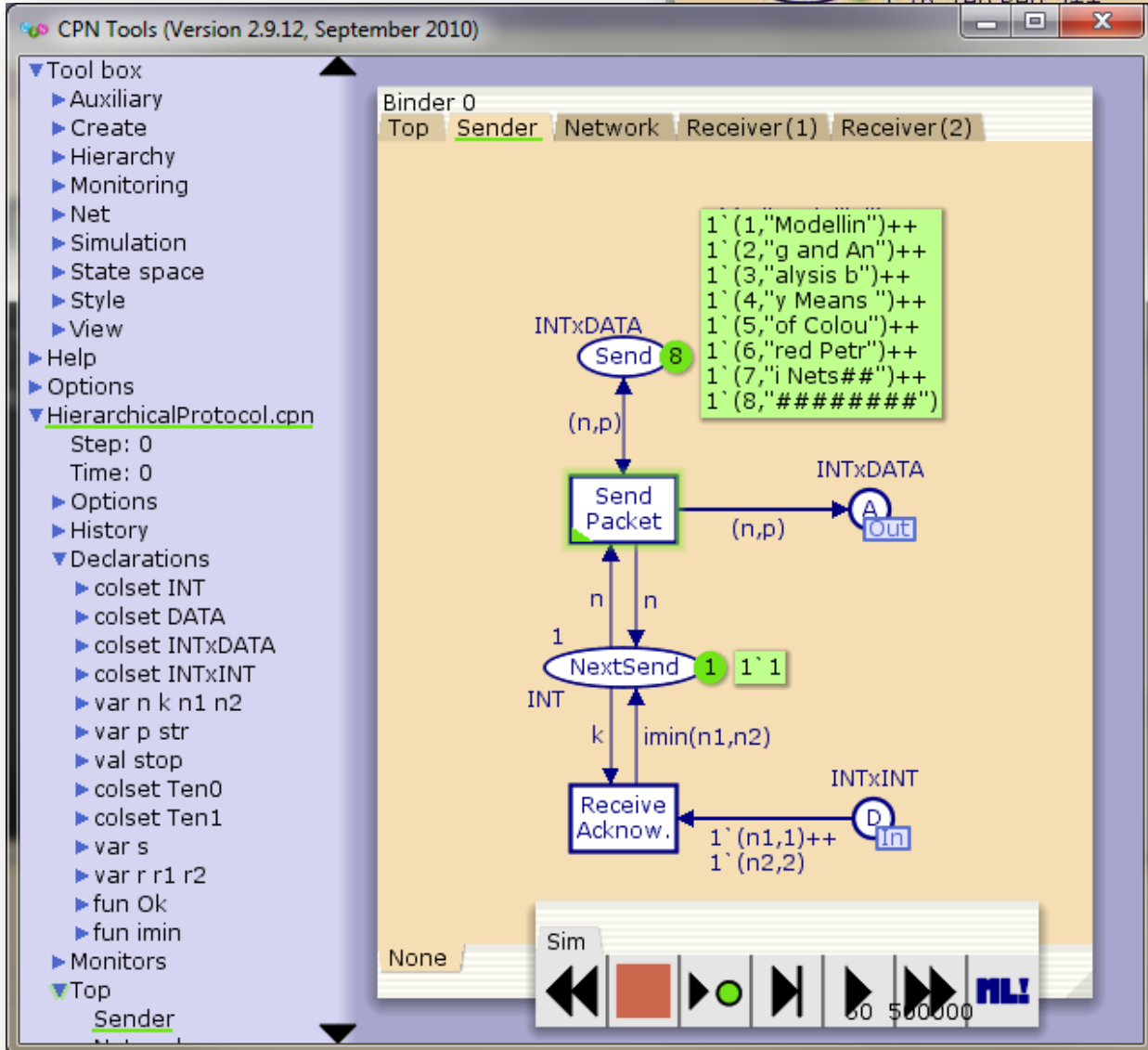
- ◆ A típusok, élkifejezések és őrfeltételek *Standard ML-ben* adhatók meg. Ez egy erősen típusos, funkcionális programozási nyelv (*Robin Milner* fejlesztette).
- ◆ *Adattípus lehet:*
 - *Atomi* (integer, string, boolean és felsorolás).
 - *Strukturált* (products, record, union, list és subsets).
- ◆ Tetszőlegesen bonyolult *függvényeket* és *műveleteket* lehet definiálni benne (pl. polimorfizmus).
- ◆ A Standard ML jól-ismert, tesztelt és nagyon általános. Számos *irodalma* van.

Design/CPN editor (egykor)

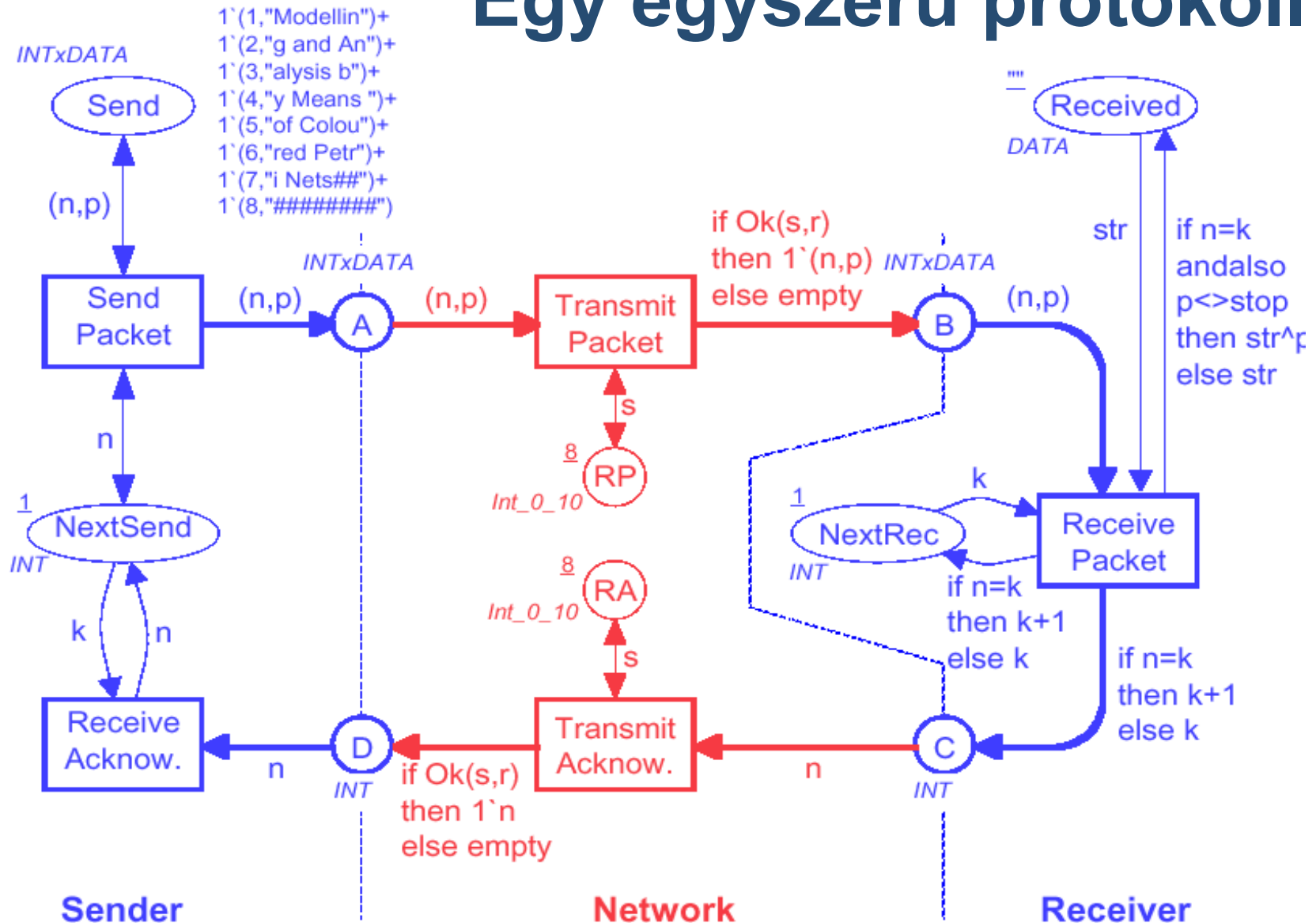


- ◆ *WYSIWYG felületről* könnyen módosíthatni tudjuk a modellünket.
- ◆ *Szintakszis-vezérelt* – sok szintaktikai hiba kiküszöbölhető.

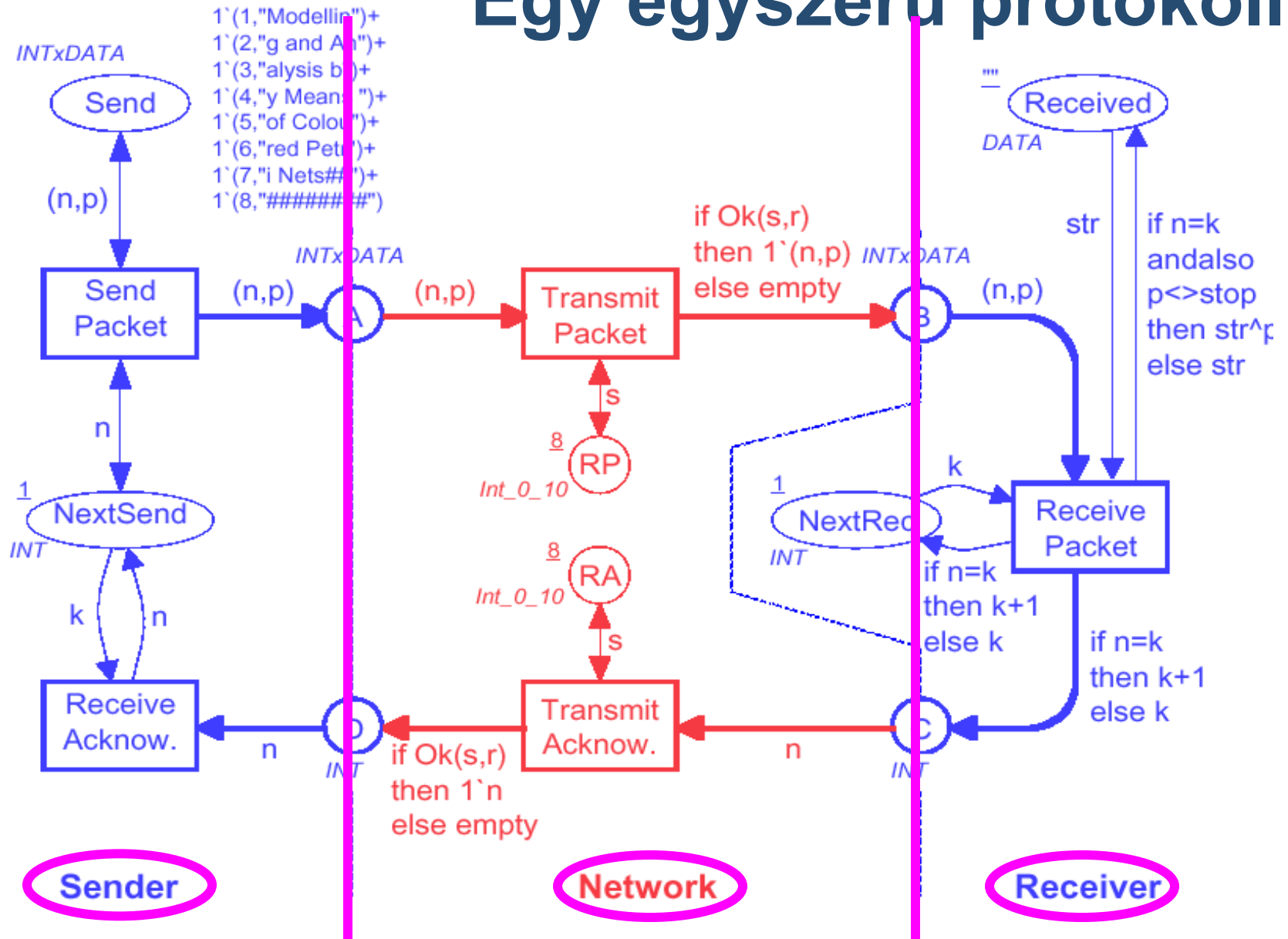
CPNTools editor (ma)



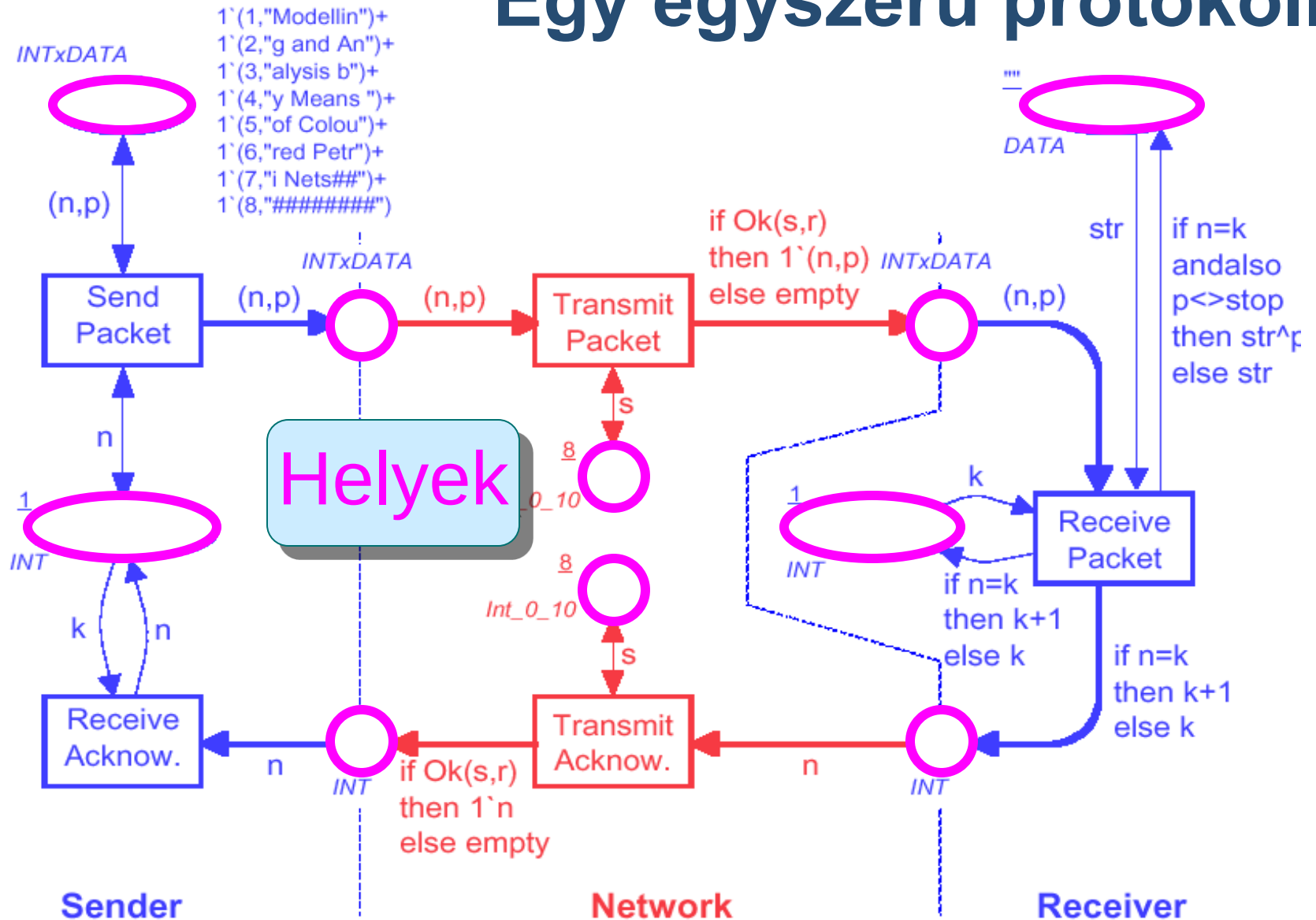
Egy egyszerű protokoll



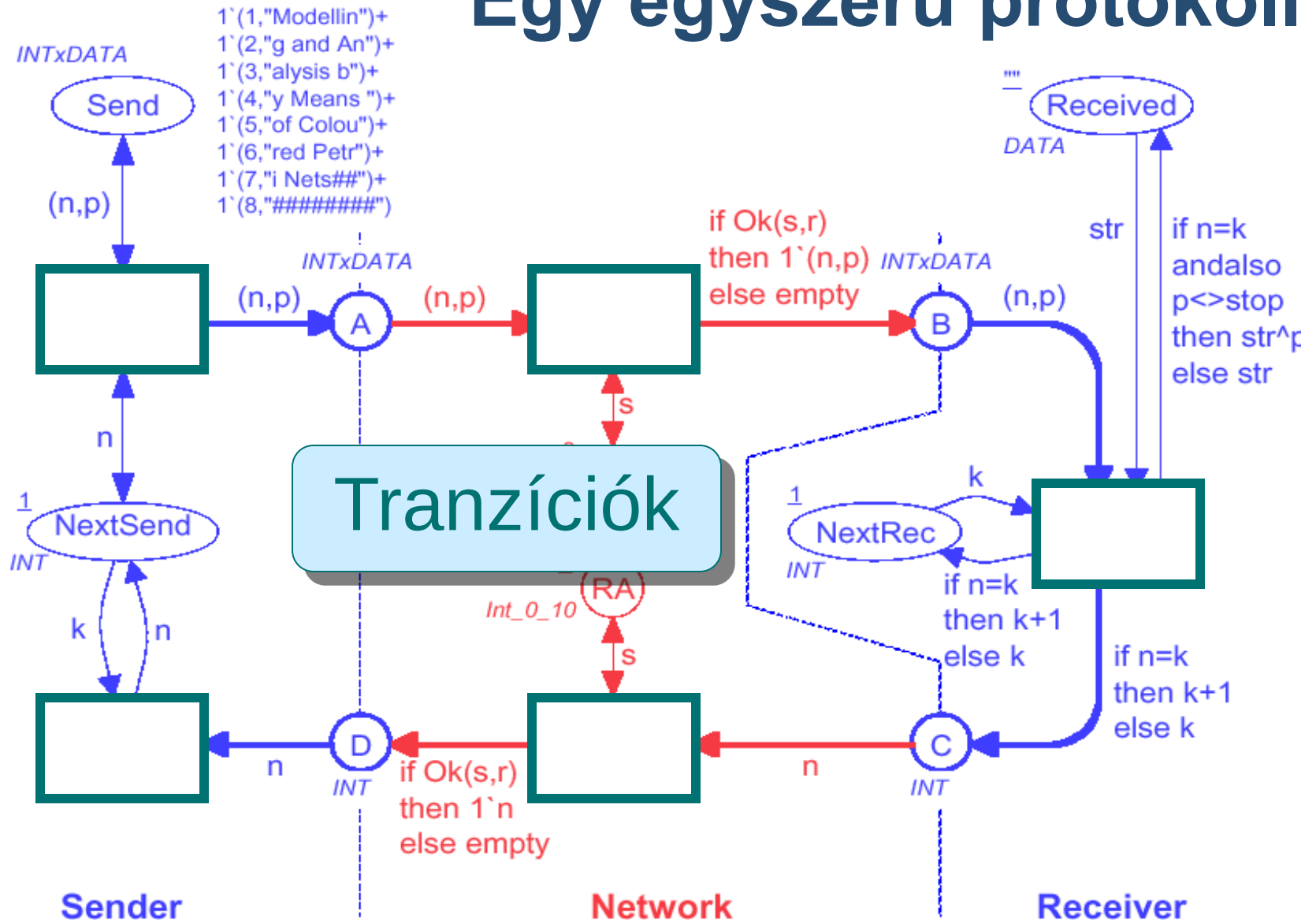
Egy egyszerű protokoll



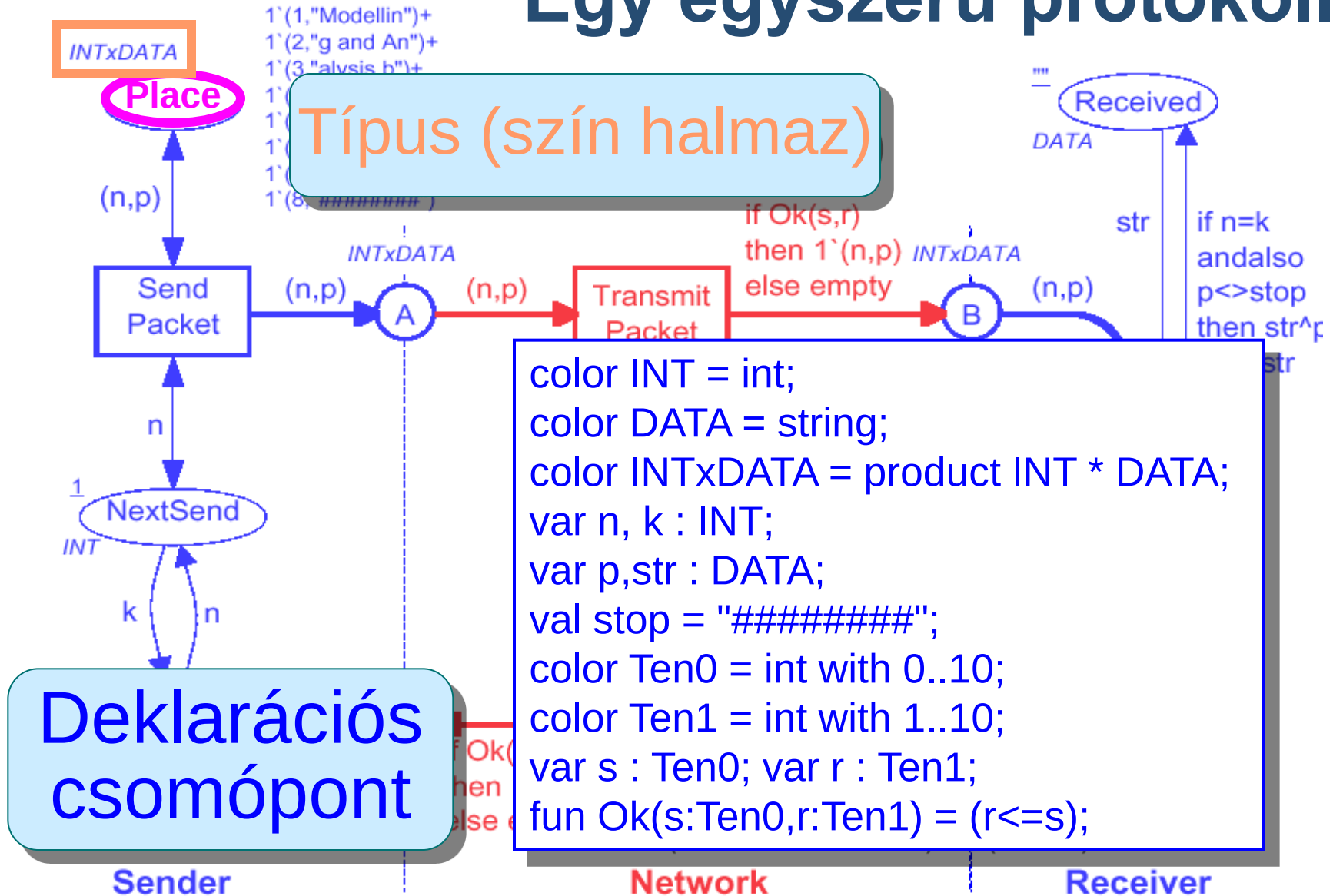
Egy egyszerű protokoll



Egy egyszerű protokoll

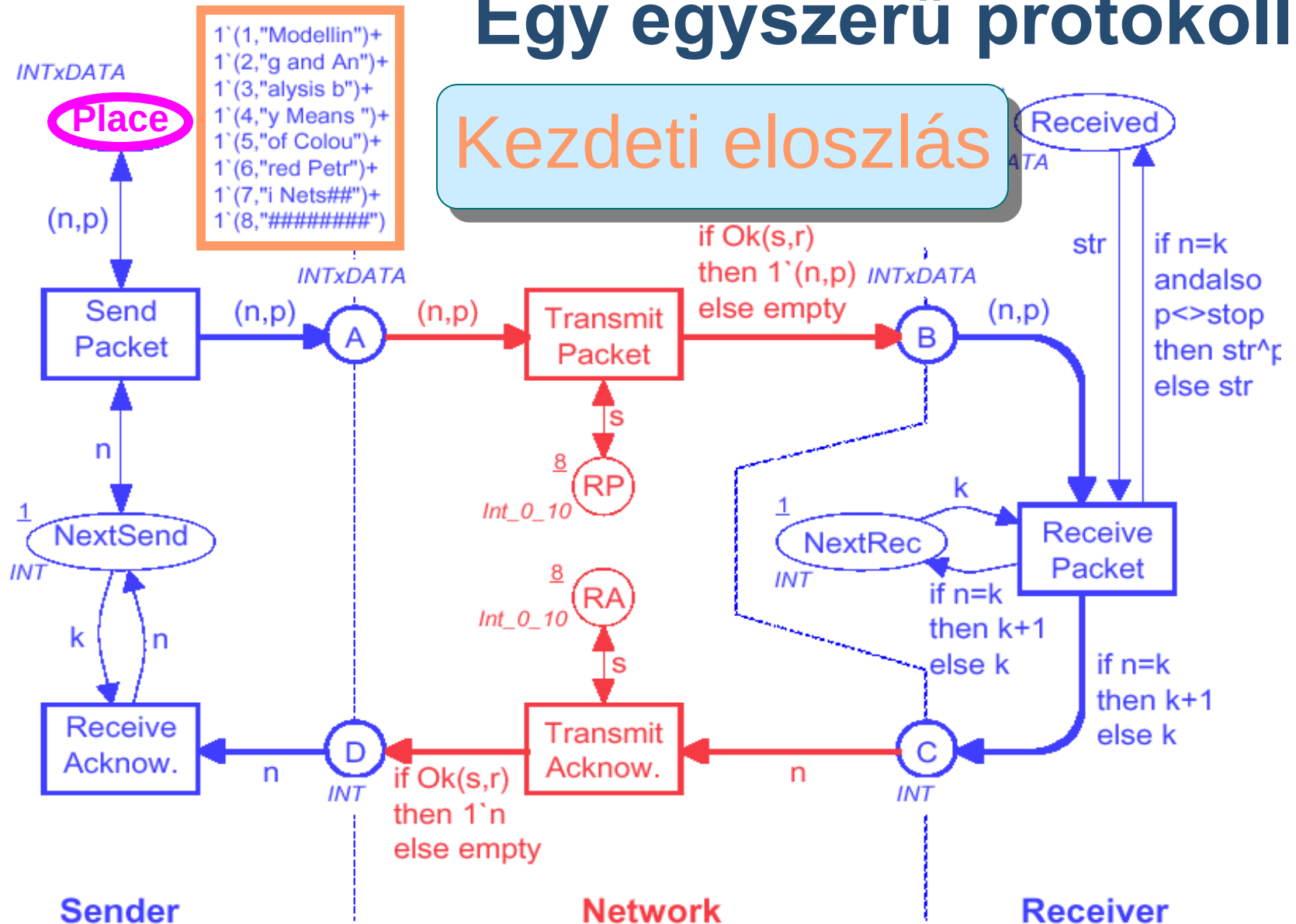


Egy egyszerű protokoll



Egy egyszerű protokoll

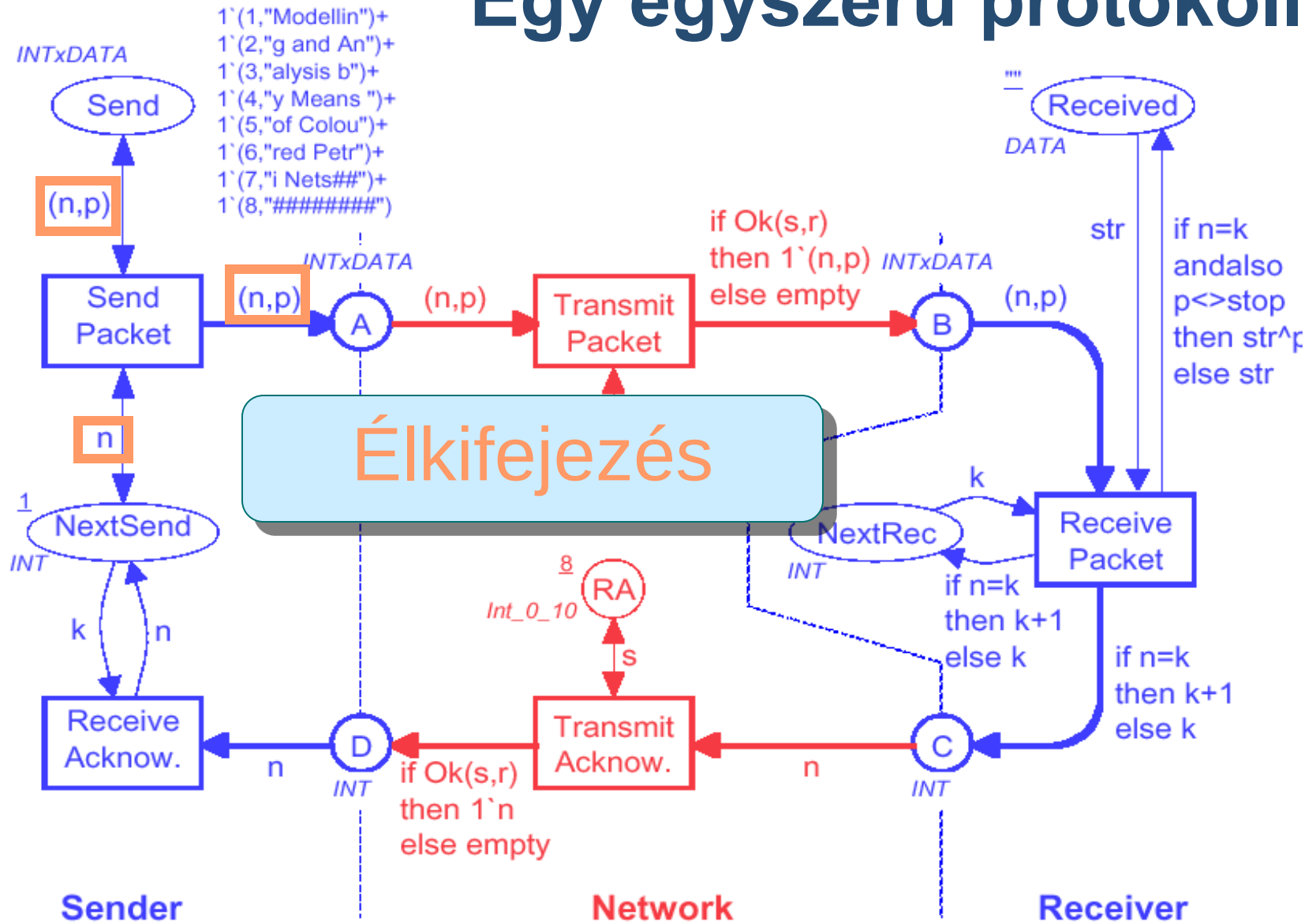
Kezdeti eloszlás



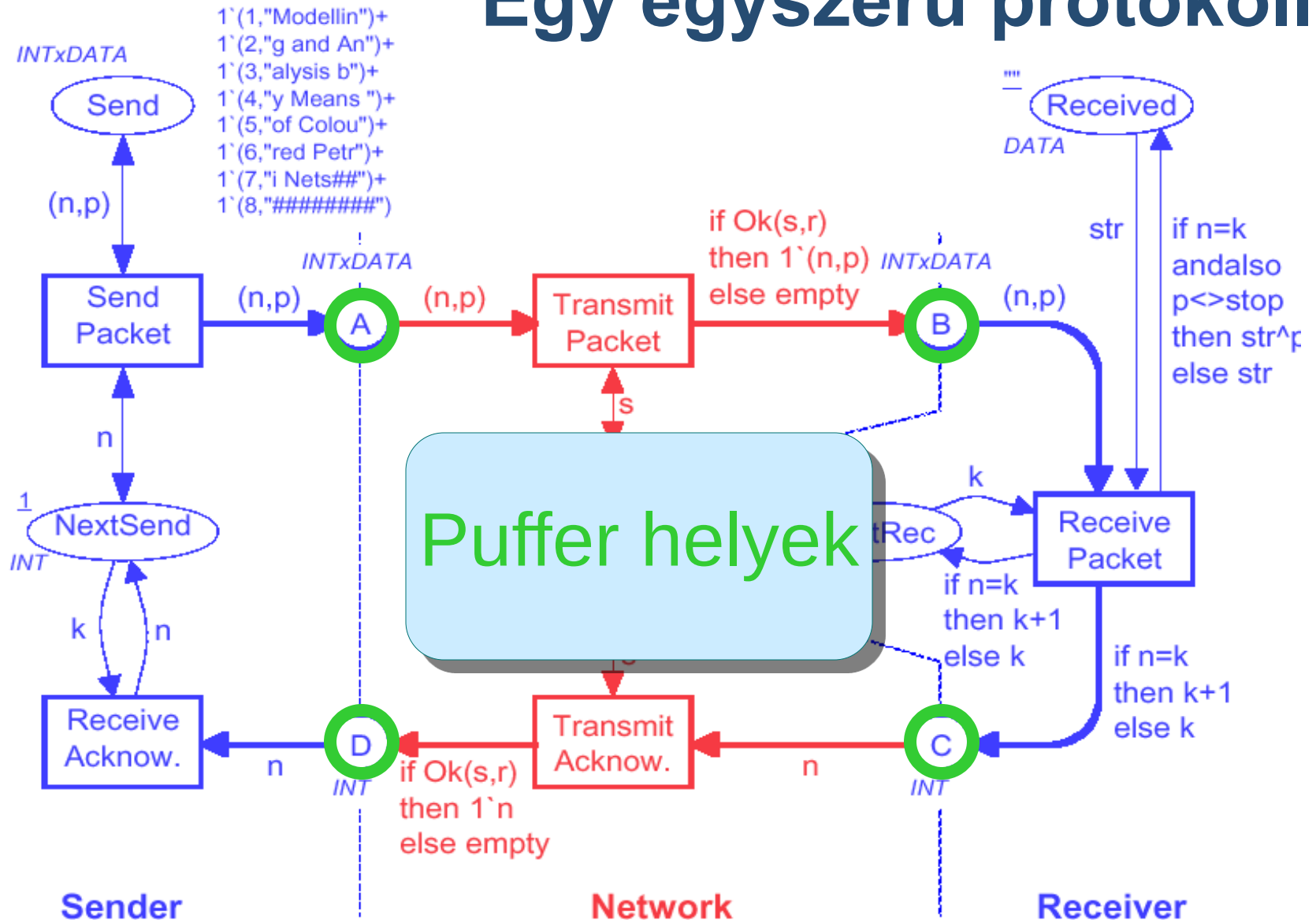
Send jelölése



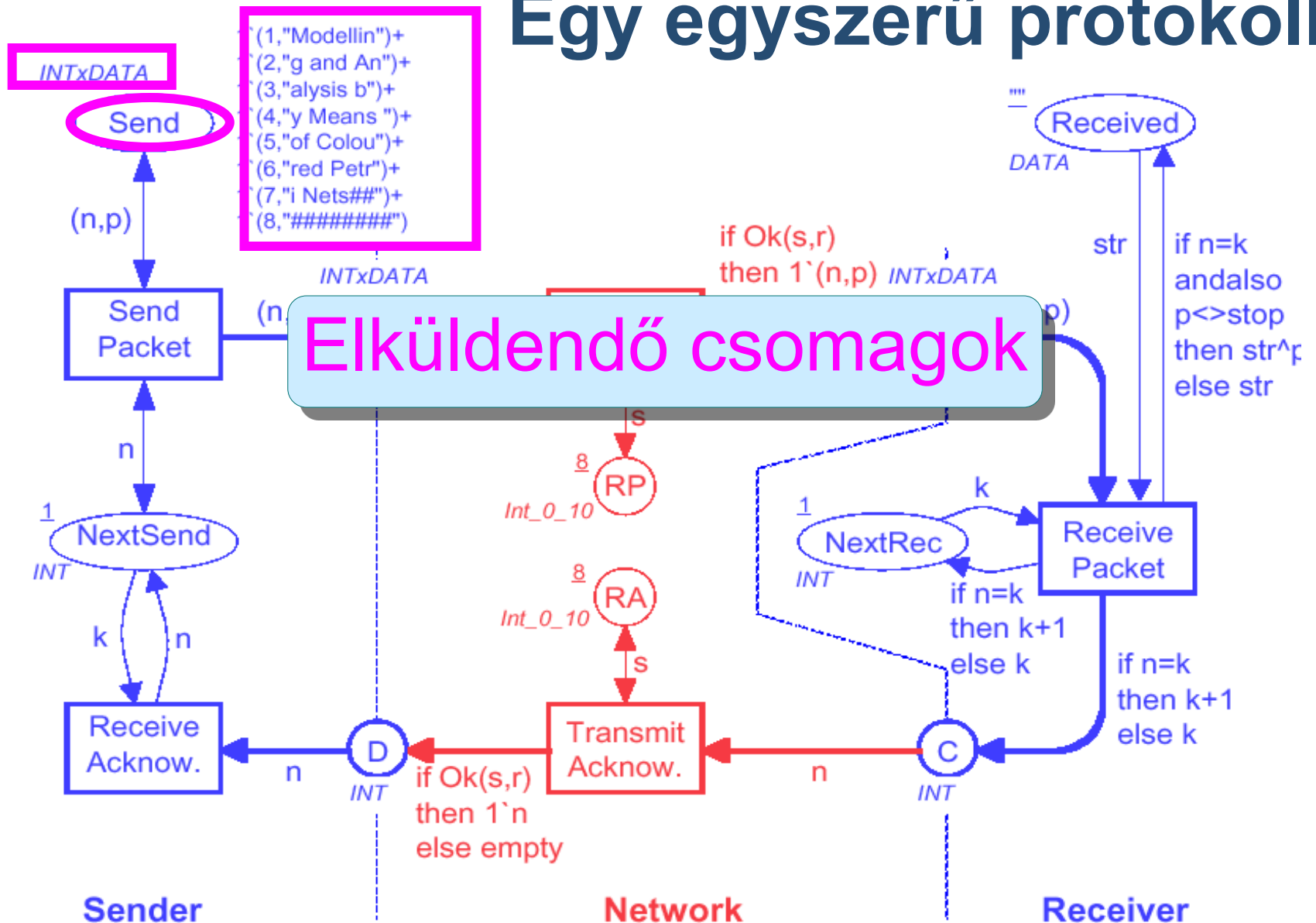
Egy egyszerű protokoll



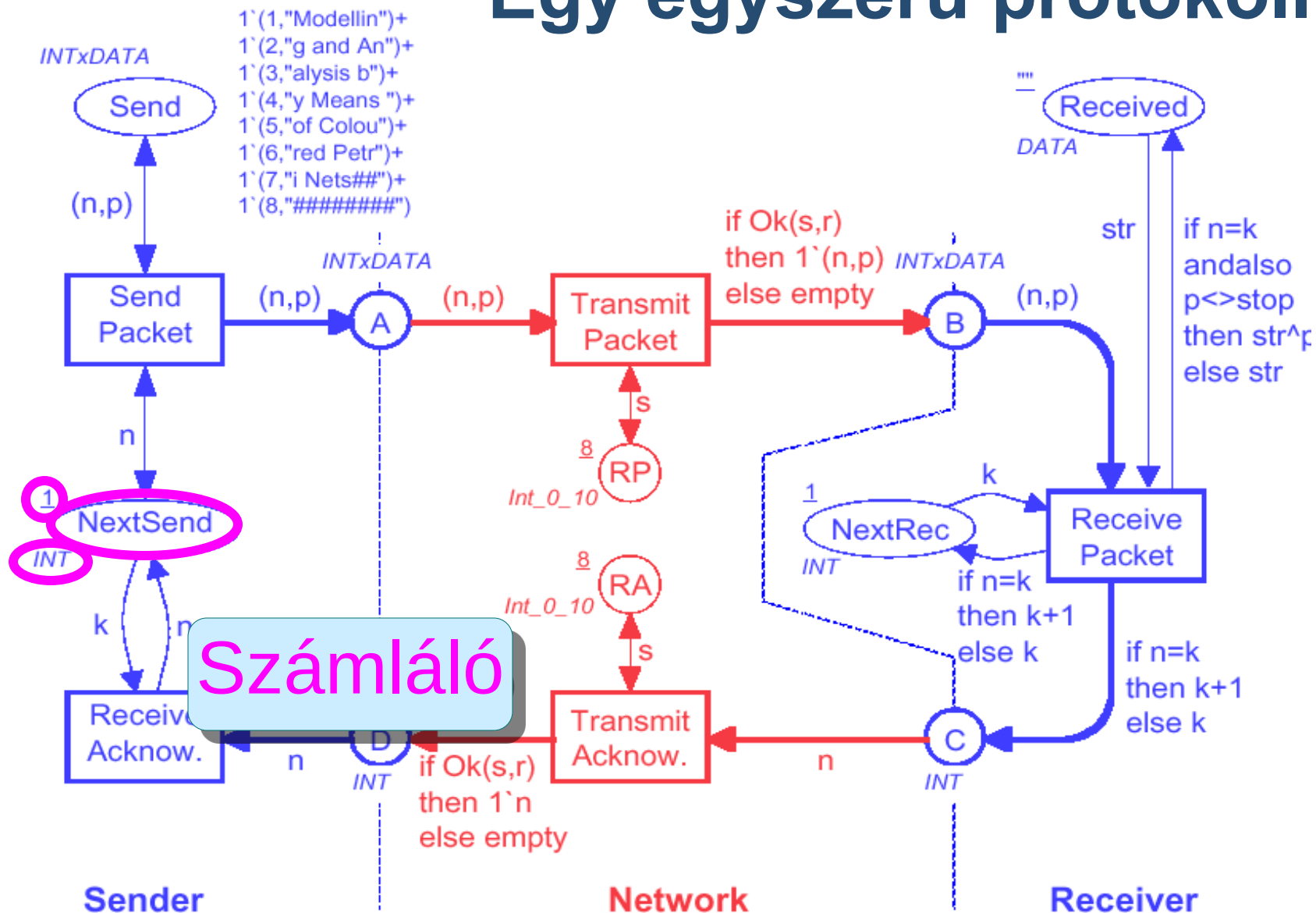
Egy egyszerű protokoll



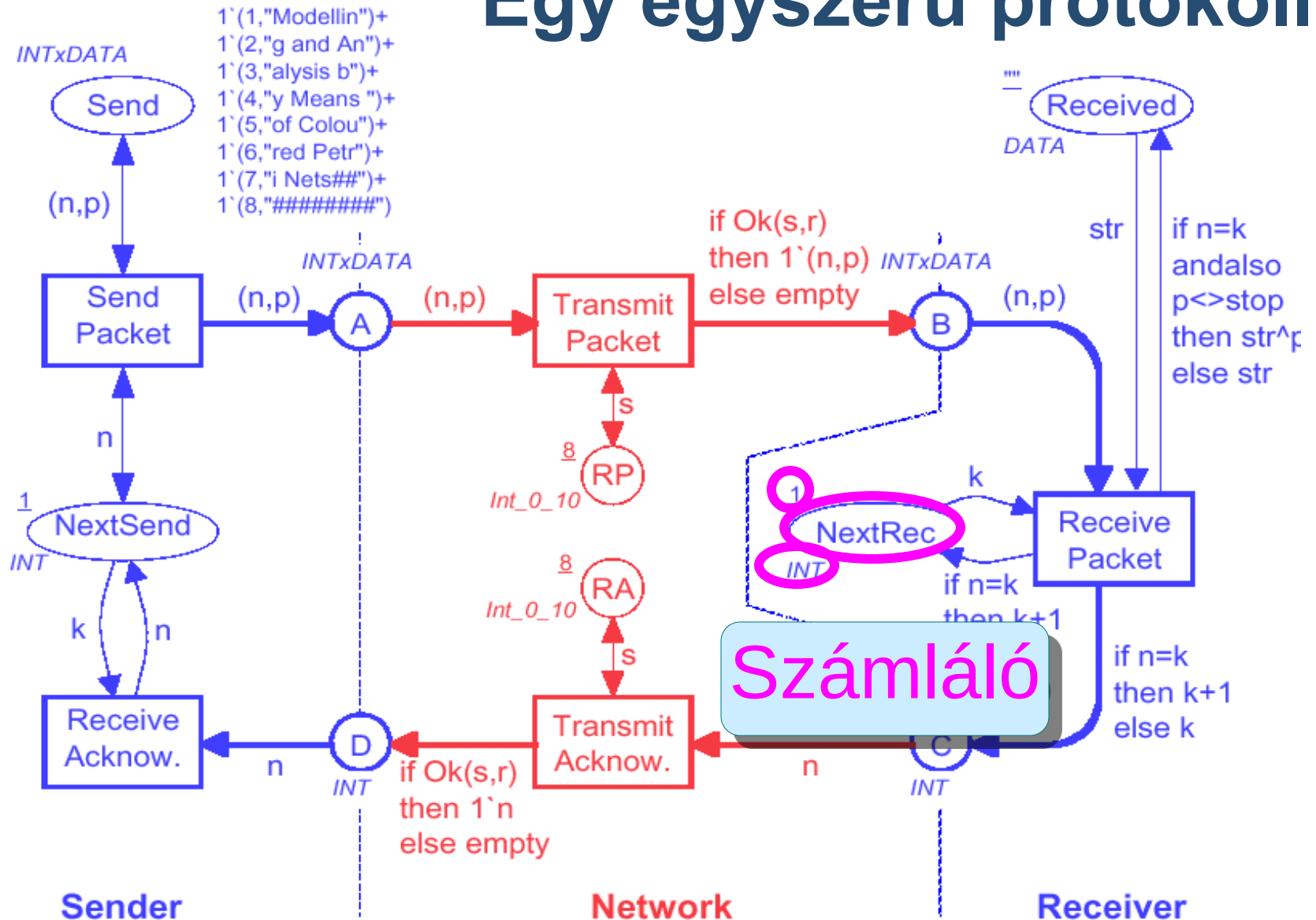
Egy egyszerű protokoll



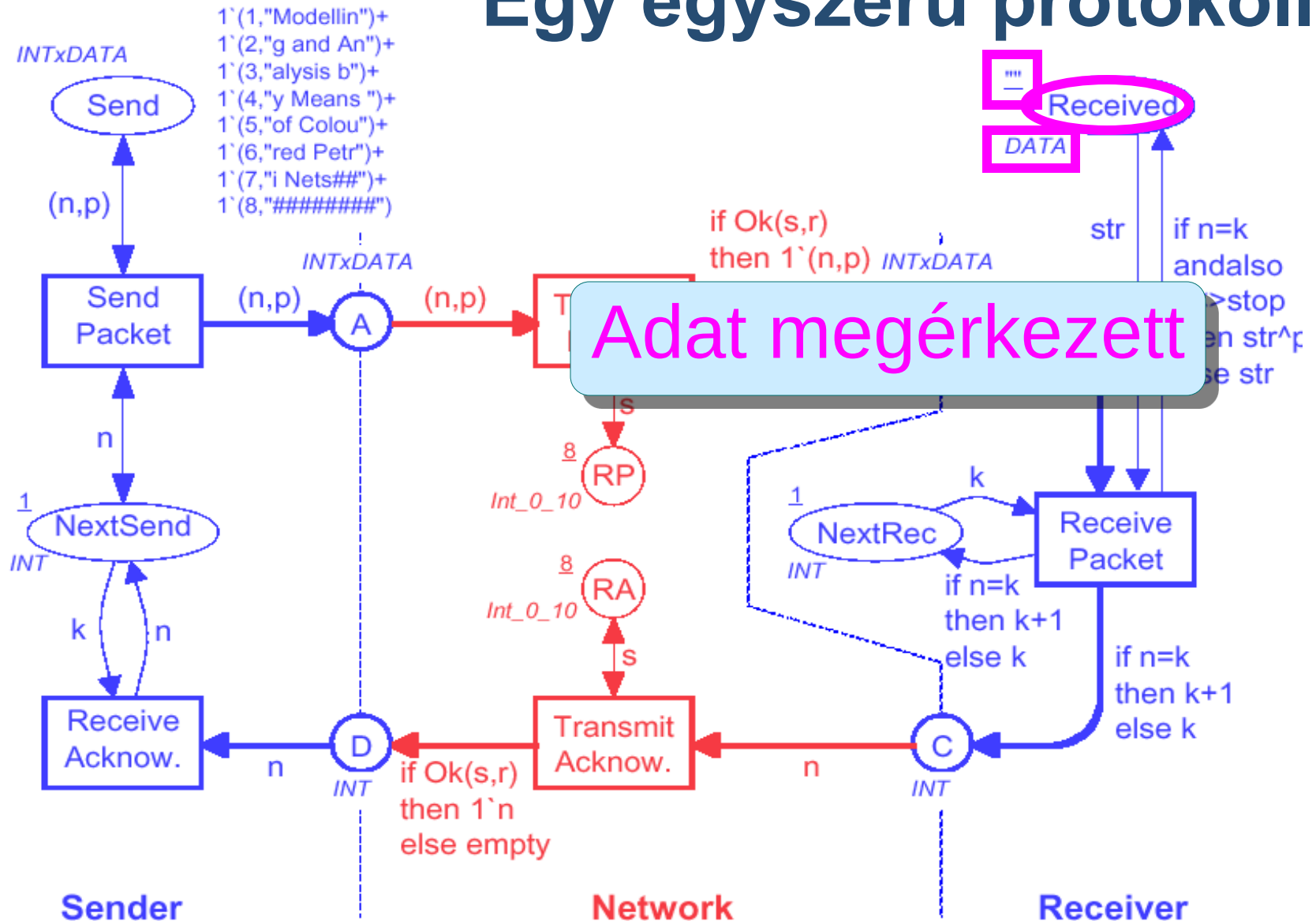
Egy egyszerű protokoll



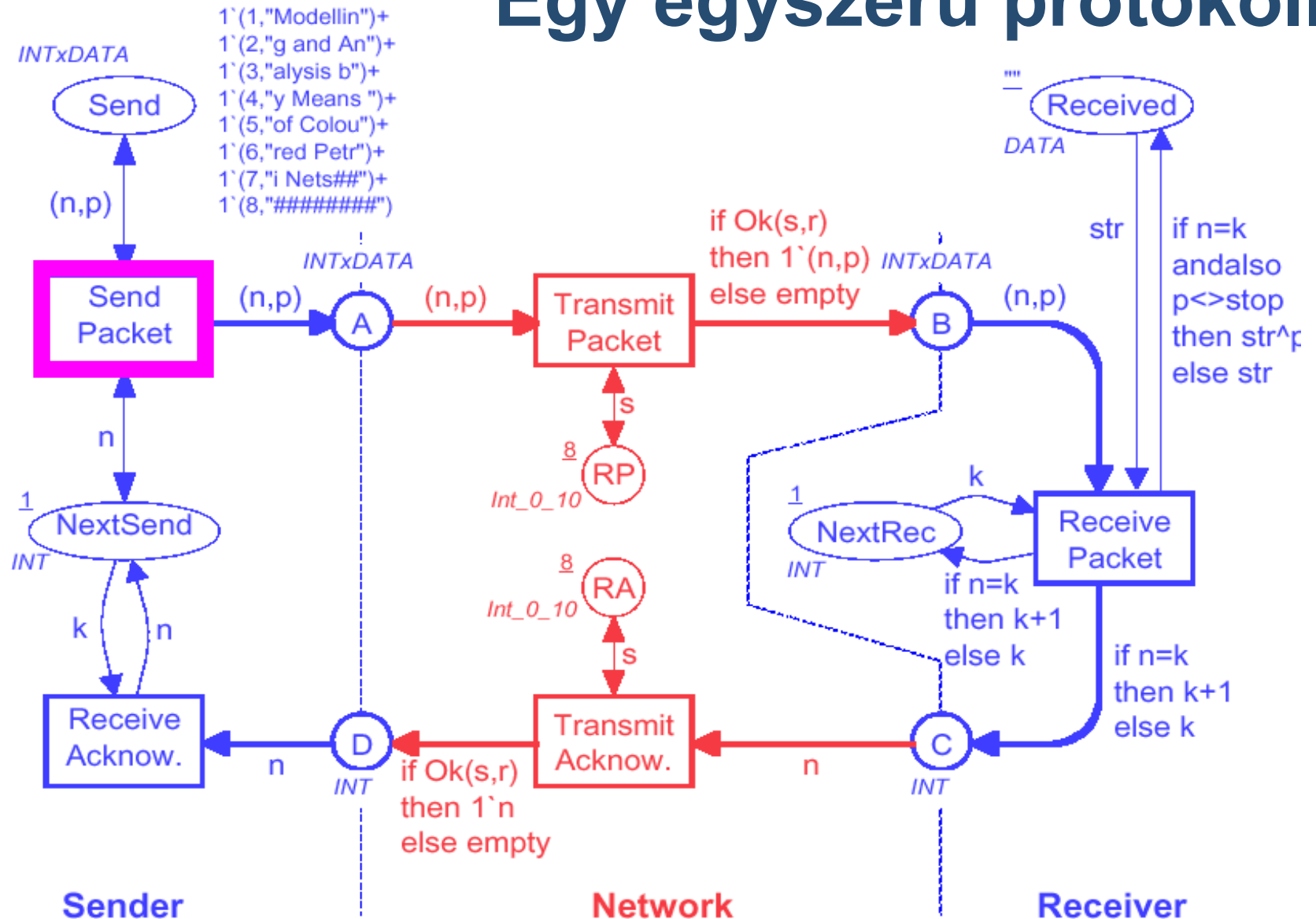
Egy egyszerű protokoll



Egy egyszerű protokoll



Egy egyszerű protokoll



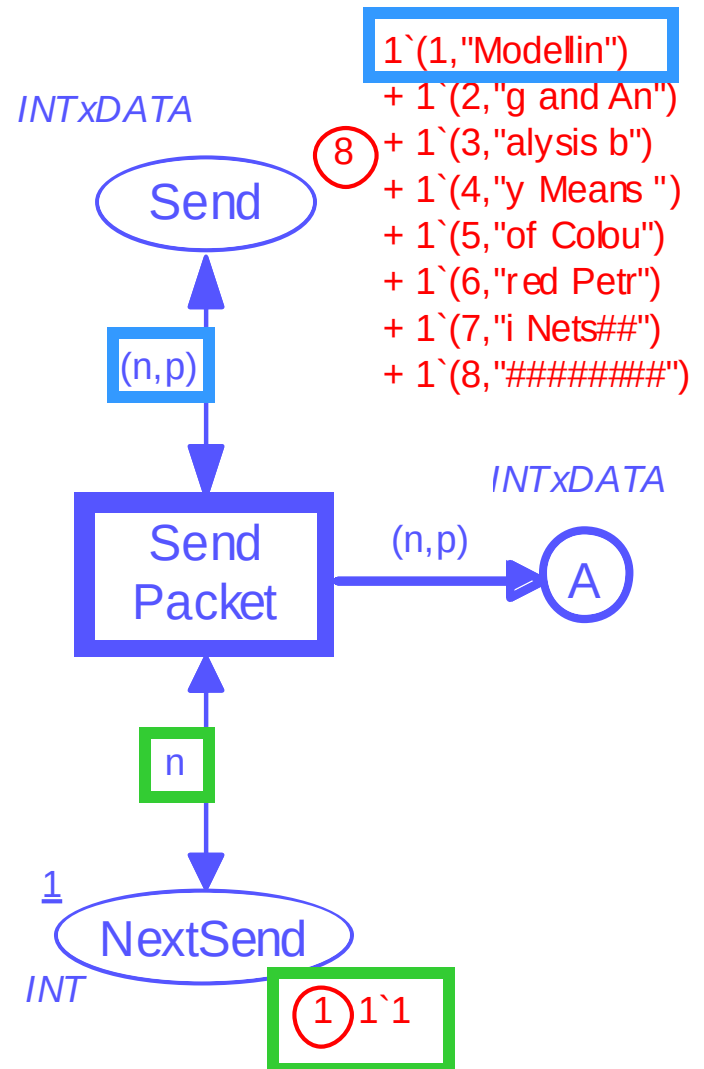
Csomagküldés

- ◆ Csak a

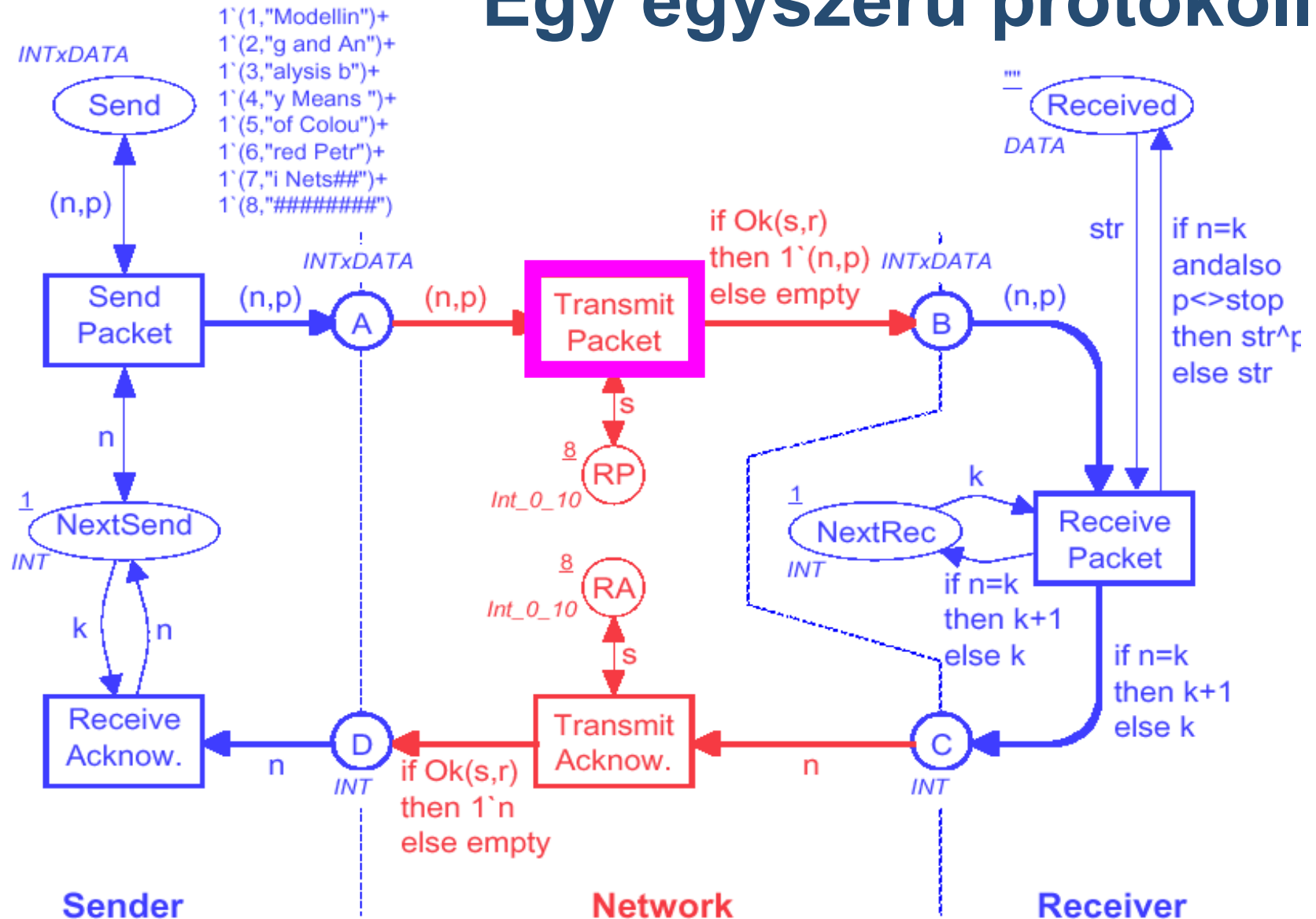
$\langle n=1, p="Modellin" \rangle$

pár engedélyezett.

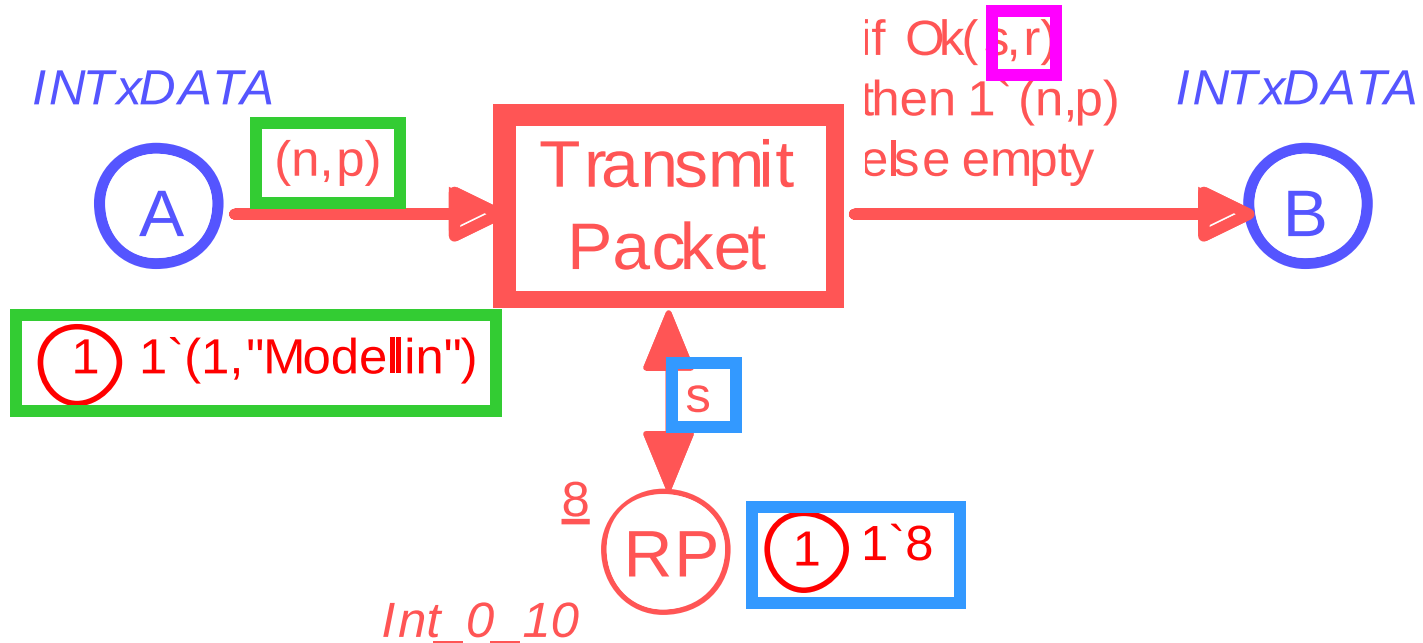
- ◆ Ha található ilyen pár, akkor a *Send Packet* elküldi azt A-nak.
- ◆ Ez azt jelenti, hogy $(1, "Modellin")$ -t elküldtük a hálózaton.
- ◆ A csomag nem törlődik a *Send* helyről és a *NextSend* sem változik.



Egy egyszerű protokoll



Csomag átvitele



◆ Az engedélyezett pár így néz ki:

■ $\langle n=1, p= "Modellin", s=8, r=... \rangle$

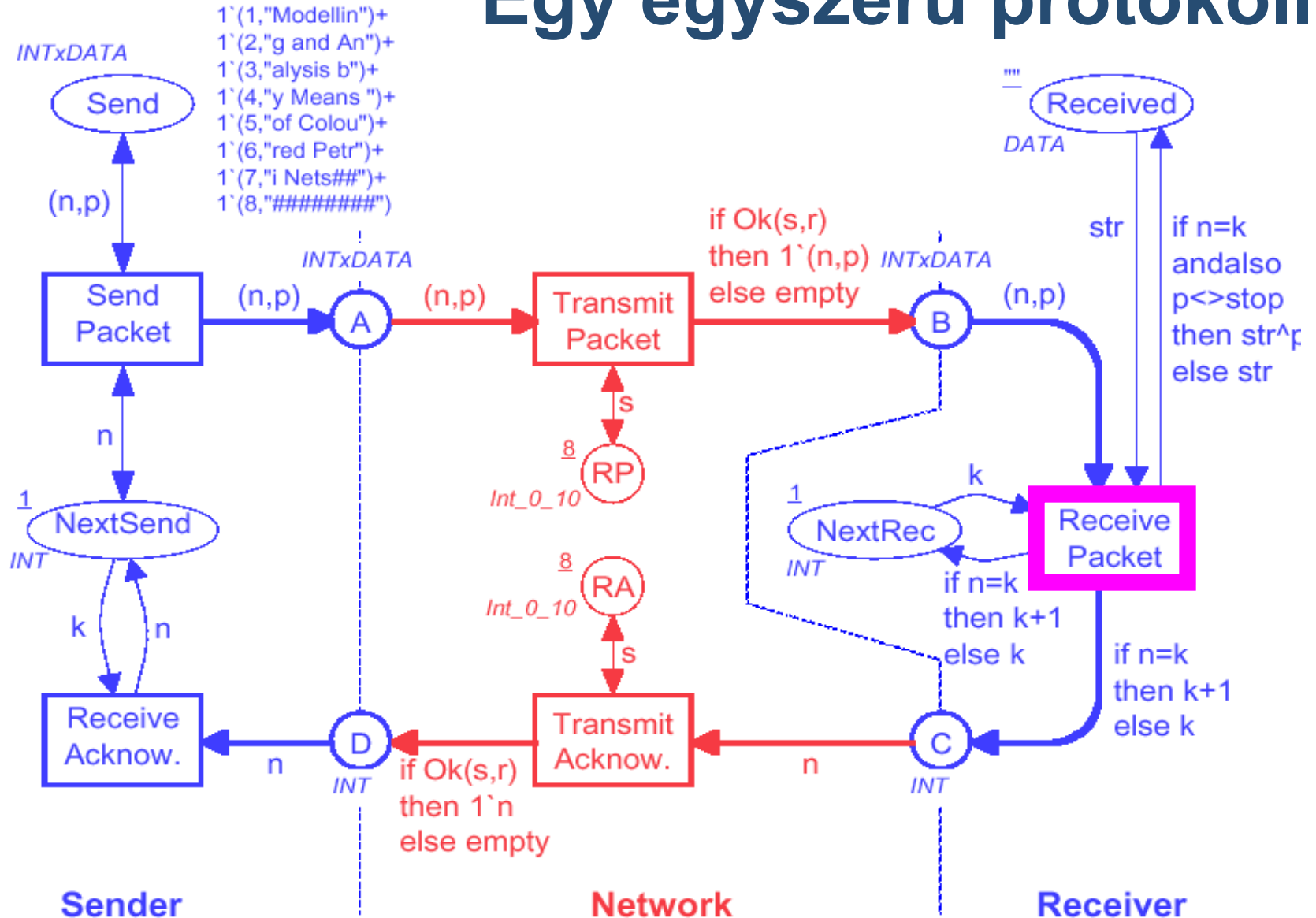
■ Típus: $r \in 1..10$

Csomagvesztés

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

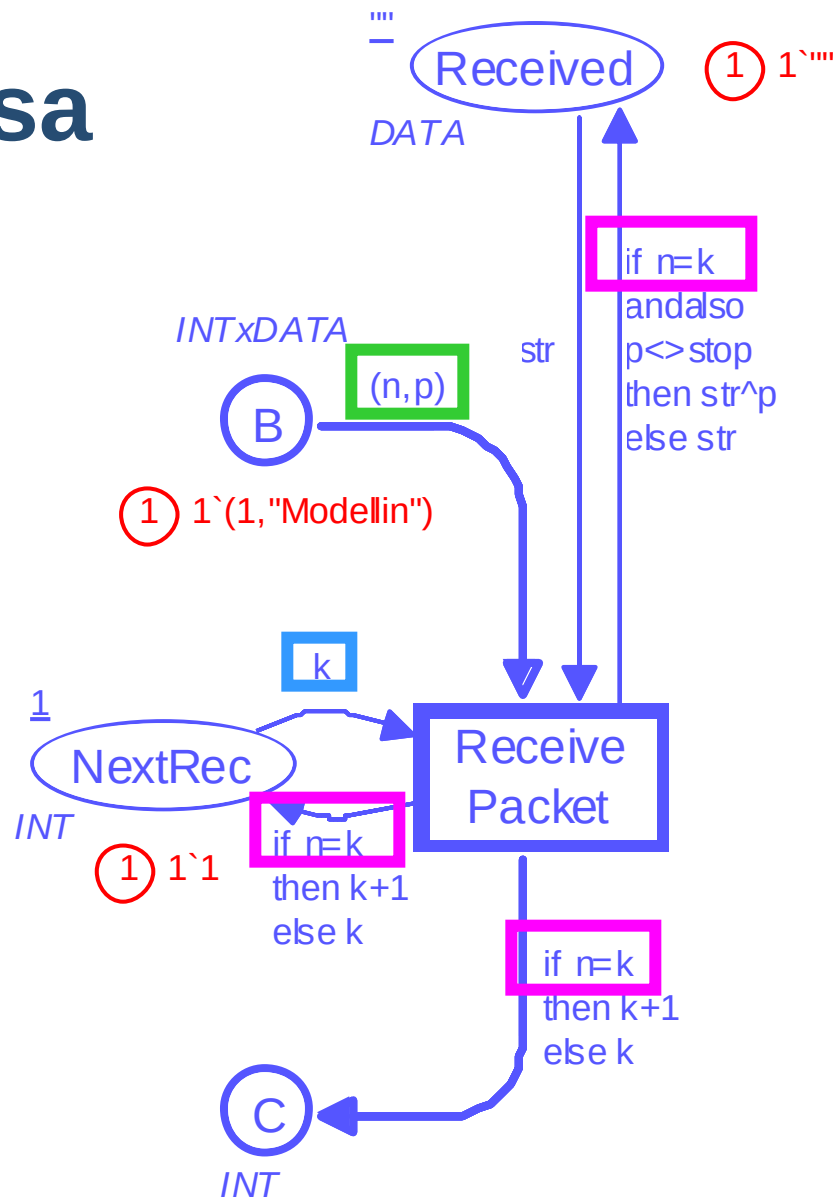
- ◆ A $Ok(s,r)$ függvény ellenőrzi, hogy $r \leq s$
 - For $r \in 1..8, Ok(s,r)=true$.
A token A-ból B-be tesszük. Ez azt jelenti, hogy a csomag *sikeresen átmegy* a hálózaton.
 - For $r \in 9..10, Ok(s,r)=false$.
Ekkor B-be nem került token. Ekkor *veszett el* a csomag.
- ◆ A CPN szimulátor *véletlenül választja* ki a párokat: 80% eséllyel sikeresen.

Egy egyszerű protokoll



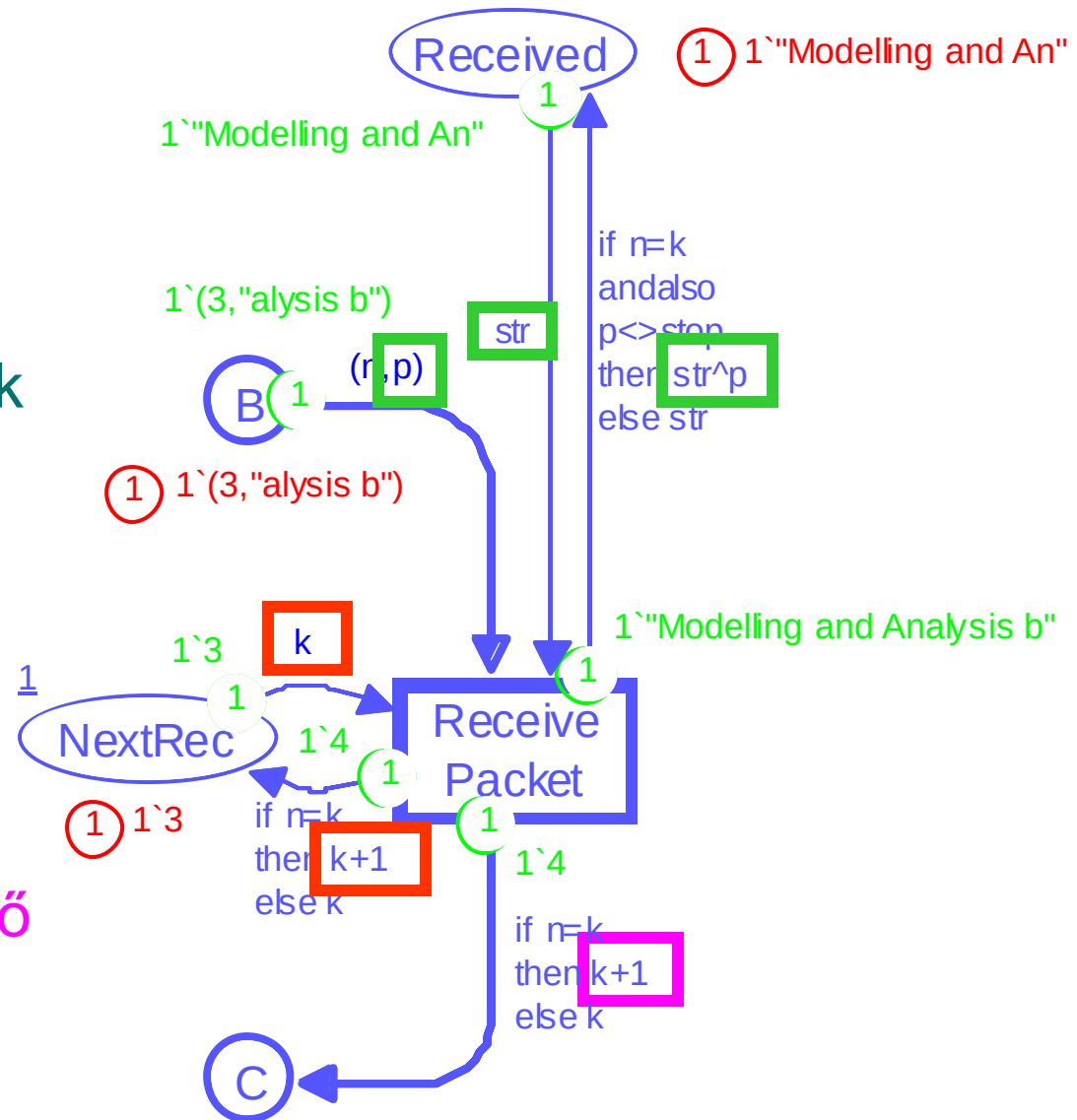
Csomag fogadása

- ◆ A bejövő (n) és a várt csomagok sorszámát (k) összehasonlítjuk.



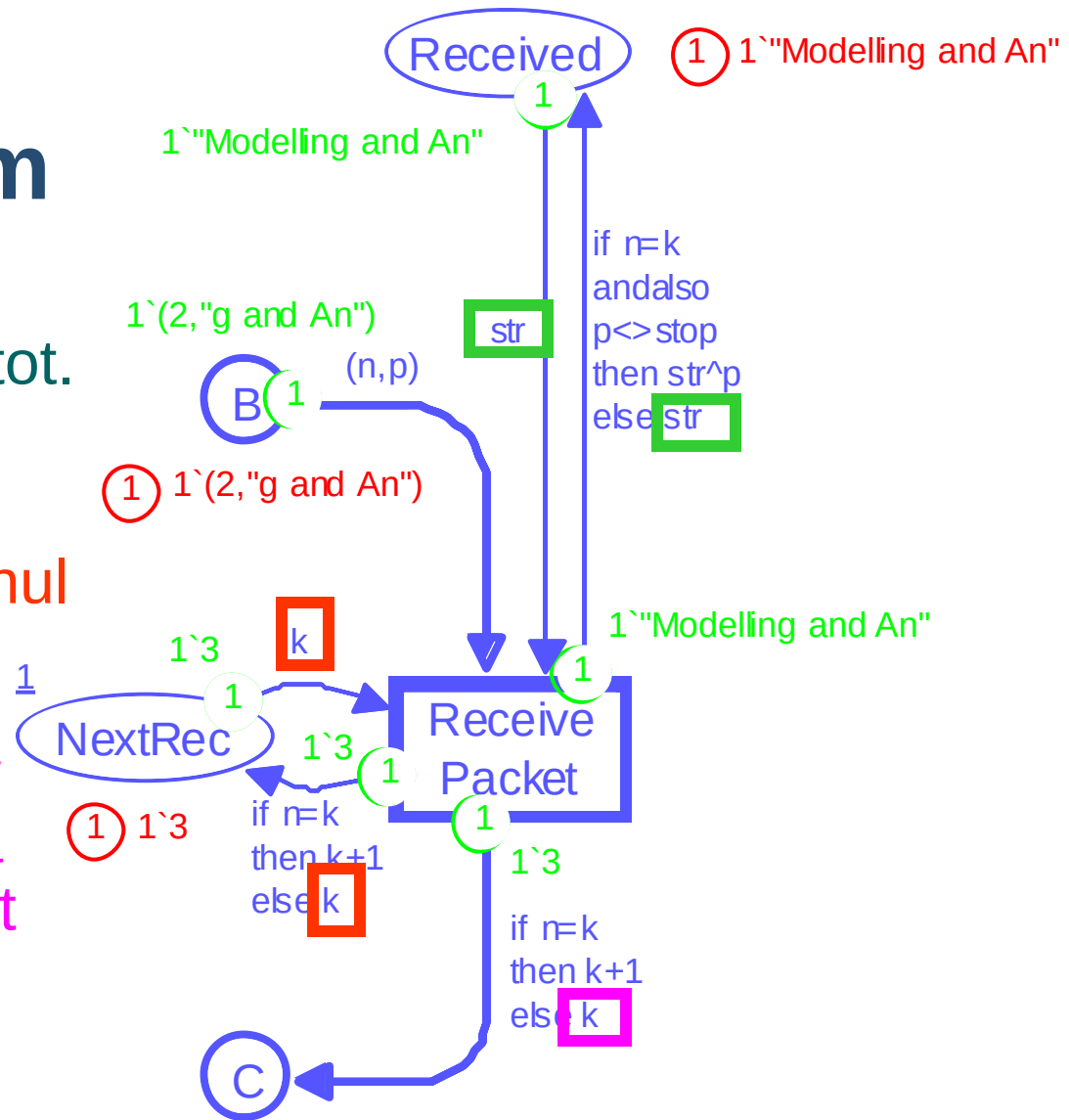
Csomagok sorszámának módosítása

- ◆ A már megkapott adatokhoz hozzáfűzzük az újonnan jöttet.
- ◆ A *NextRec* számlálót egyenként növeljük.
- ◆ *Acknowledgement* üzenetet küldünk. Mellékeljük a következő körben várt üzenet sorszámát.

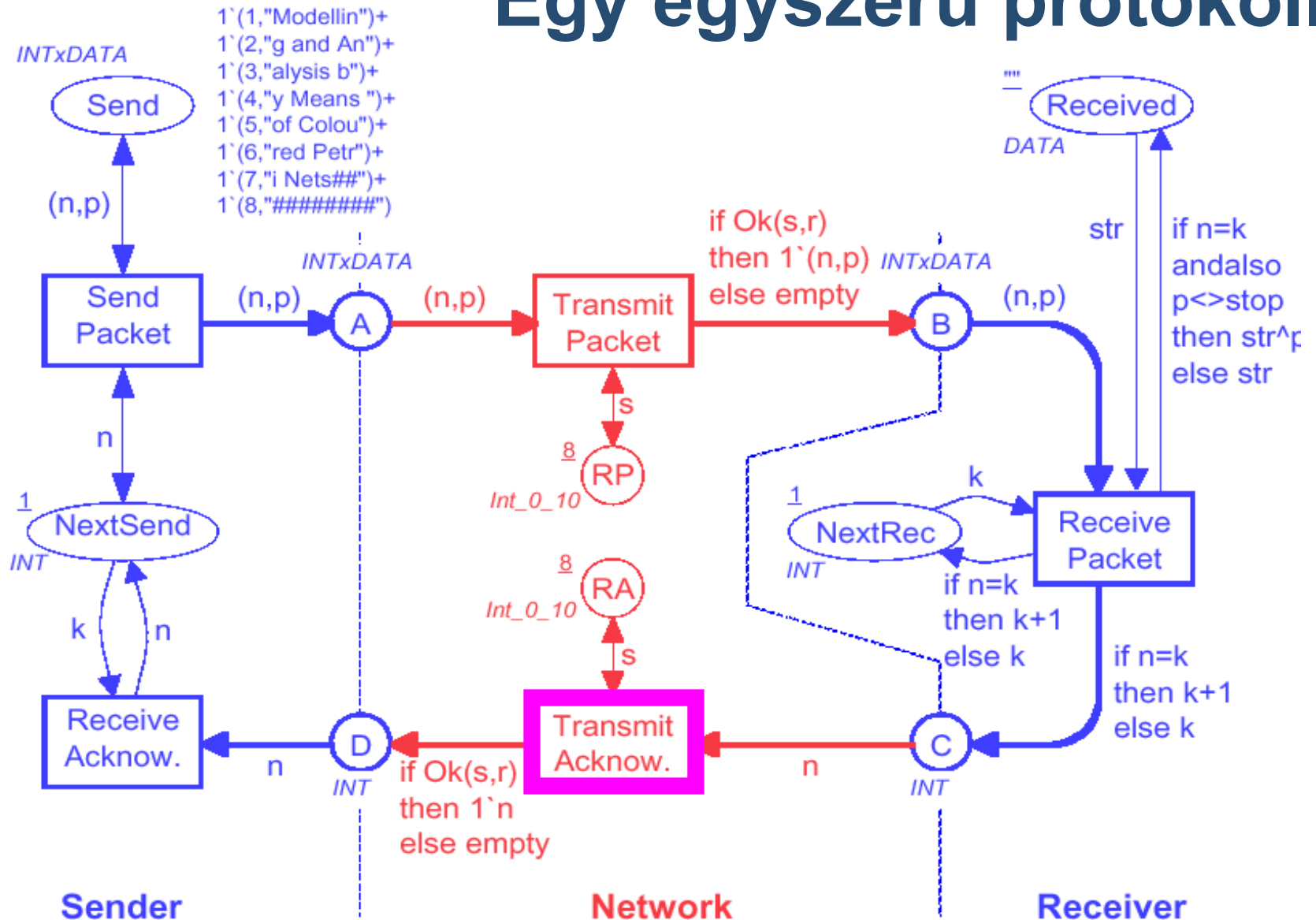


Rossz csomag sorszám

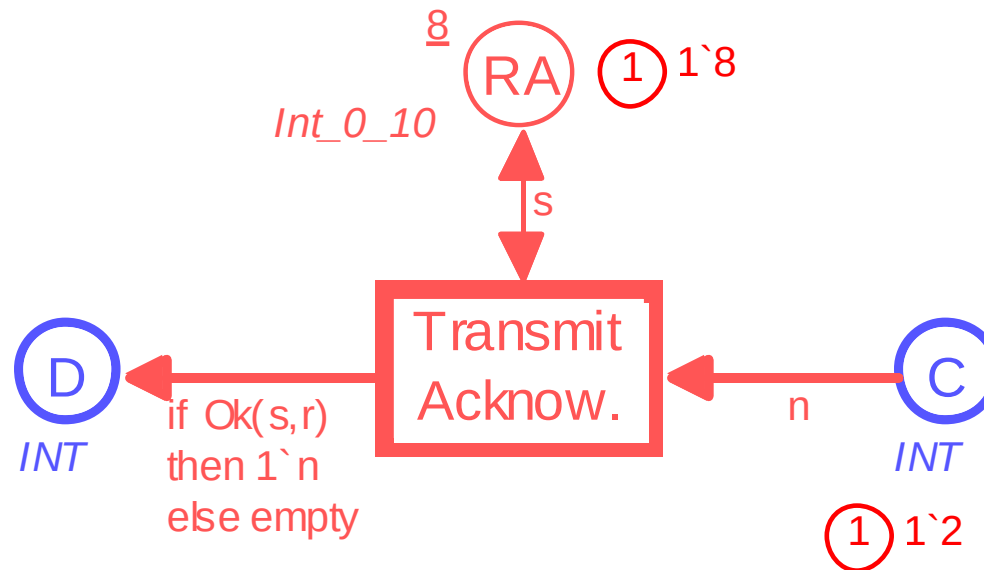
- ◆ *Eldobjuk a kapott adatot.*
- ◆ *A NextRec-t változatlanul hagyjuk.*
- ◆ *Egy acknowledgement üzenetet küldünk. Ez a várt csomag sorszámát tartalmazza.*



Egy egyszerű protokoll

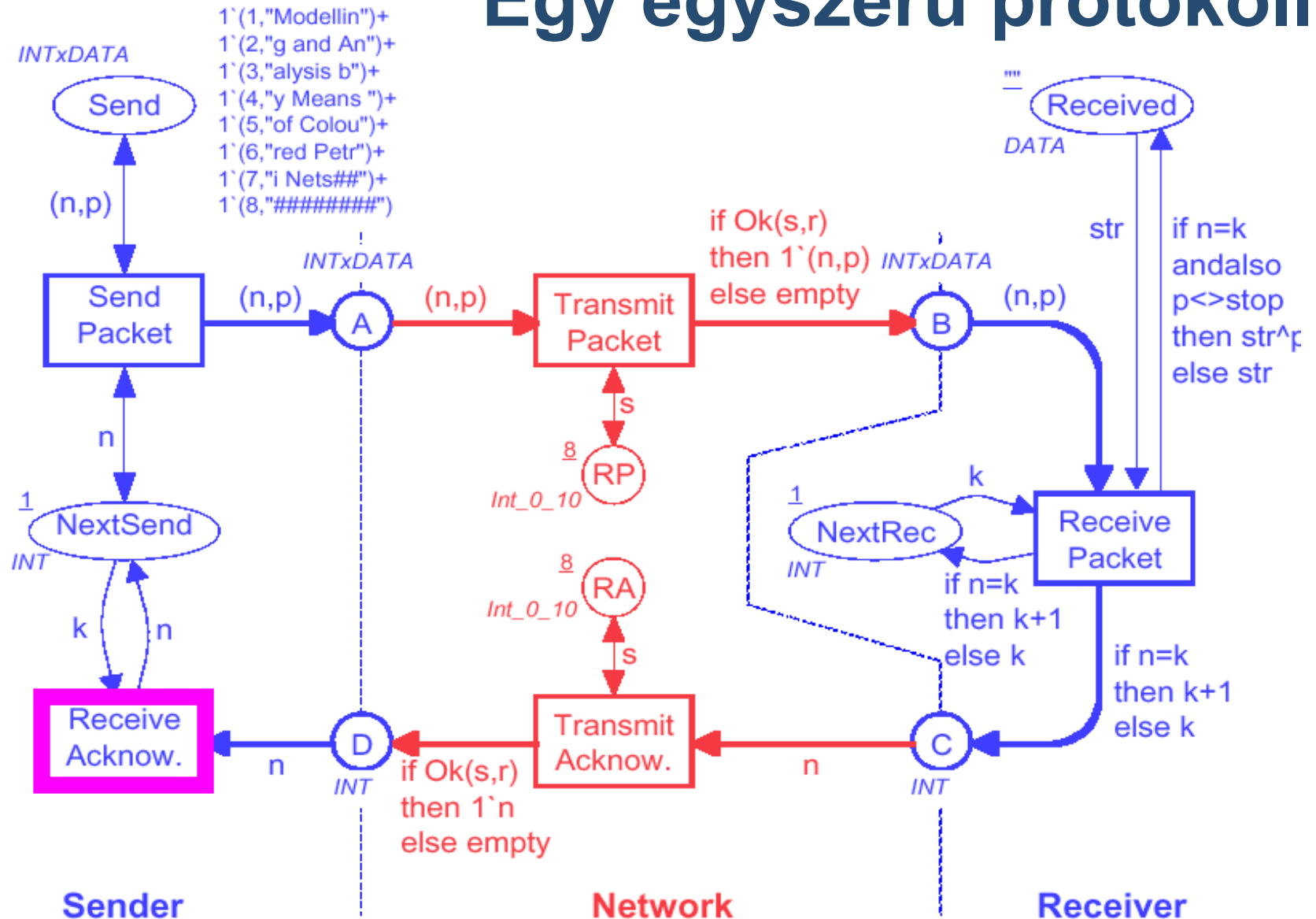


Átvitel visszajelzés

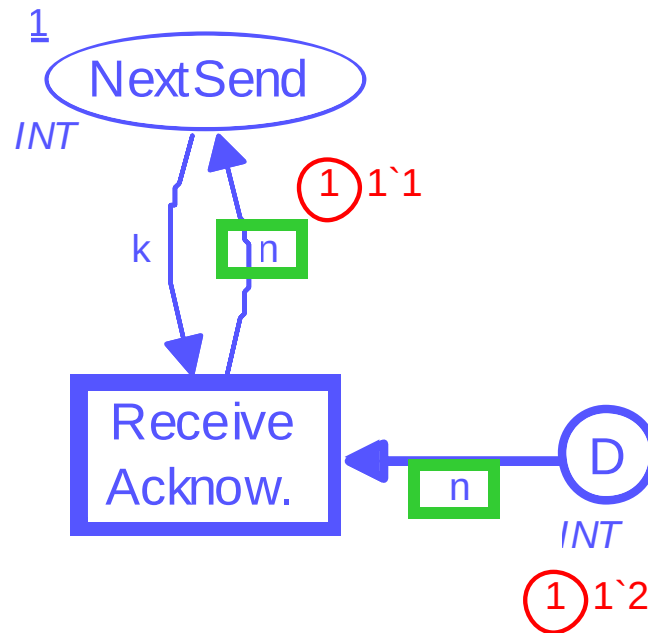


- ◆ Hasonlóan működik a *Csomag átvitelhez*.

Egy egyszerű protokoll

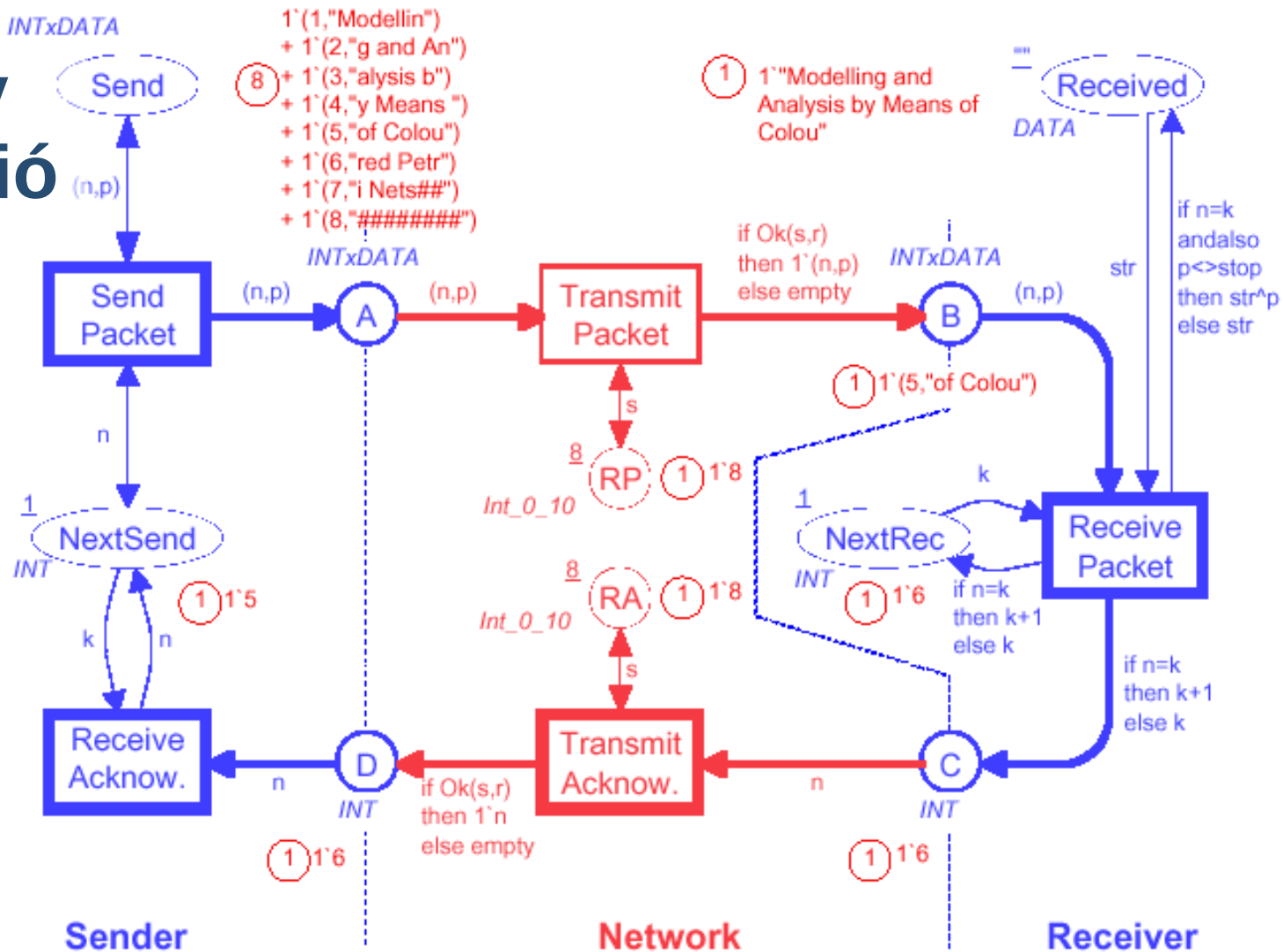


Fogadási visszajelzés



- ◆ Amikor az acknowledge megérkezik a *Küldő*höz, az azonnal frissíti a *NextSend* számlálót.
- ◆ Ebben az esetben a számláló értéke 2 lesz, így legközelebb a *Küldő* a 2-es csomagot fogja küldeni.

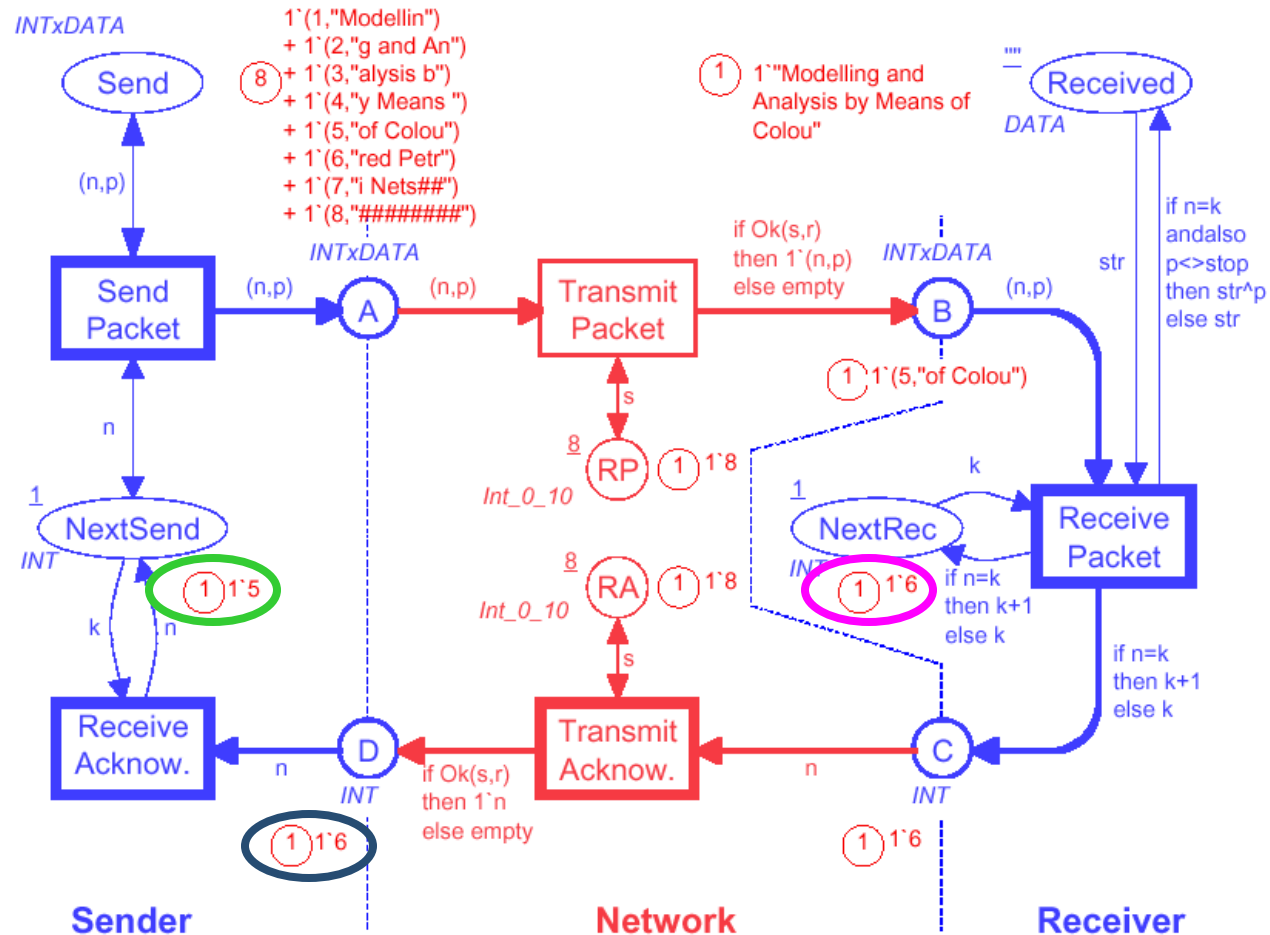
Interaktív szimuláció



- ◆ A Szimuláció eredményét az ábrán láthatjuk.
 - A következő tüzelést manuálisan vagy automatikusan lehet kijelölni.
 - A felhasználó *watchpoint*ot és *breakpoint*okat definiálhat.

Közbülső állapot

- ◆ *Fogadó várja a 6-os csomagot.*
- ◆ *Küldő még csak az 5-ös csomagot küldi.*
- ◆ A 6-os acknowledgement kérő csomag érkezik meg.
- ◆ Ezután a *NextSend* frissítődik majd a *Küldő* elindítja a 6-os csomagot.

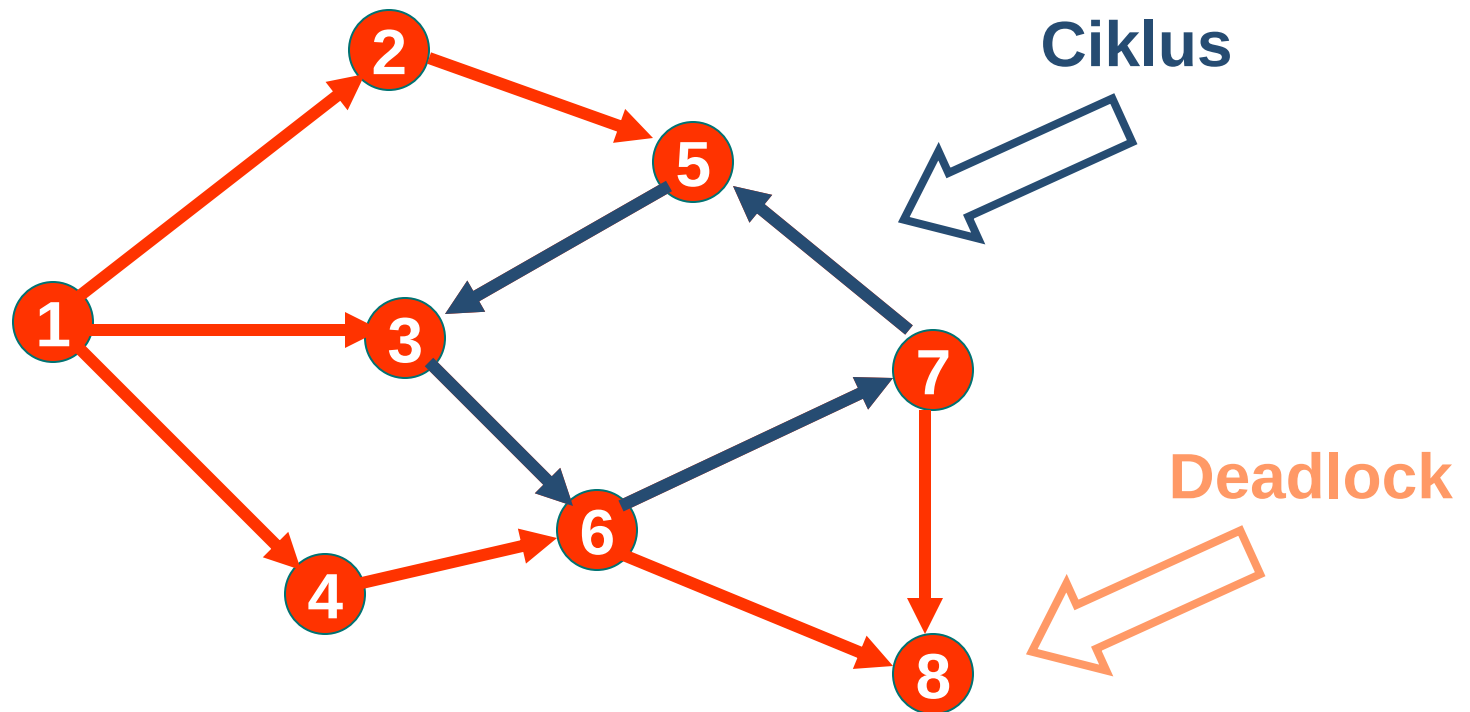


A CPN verifikációja

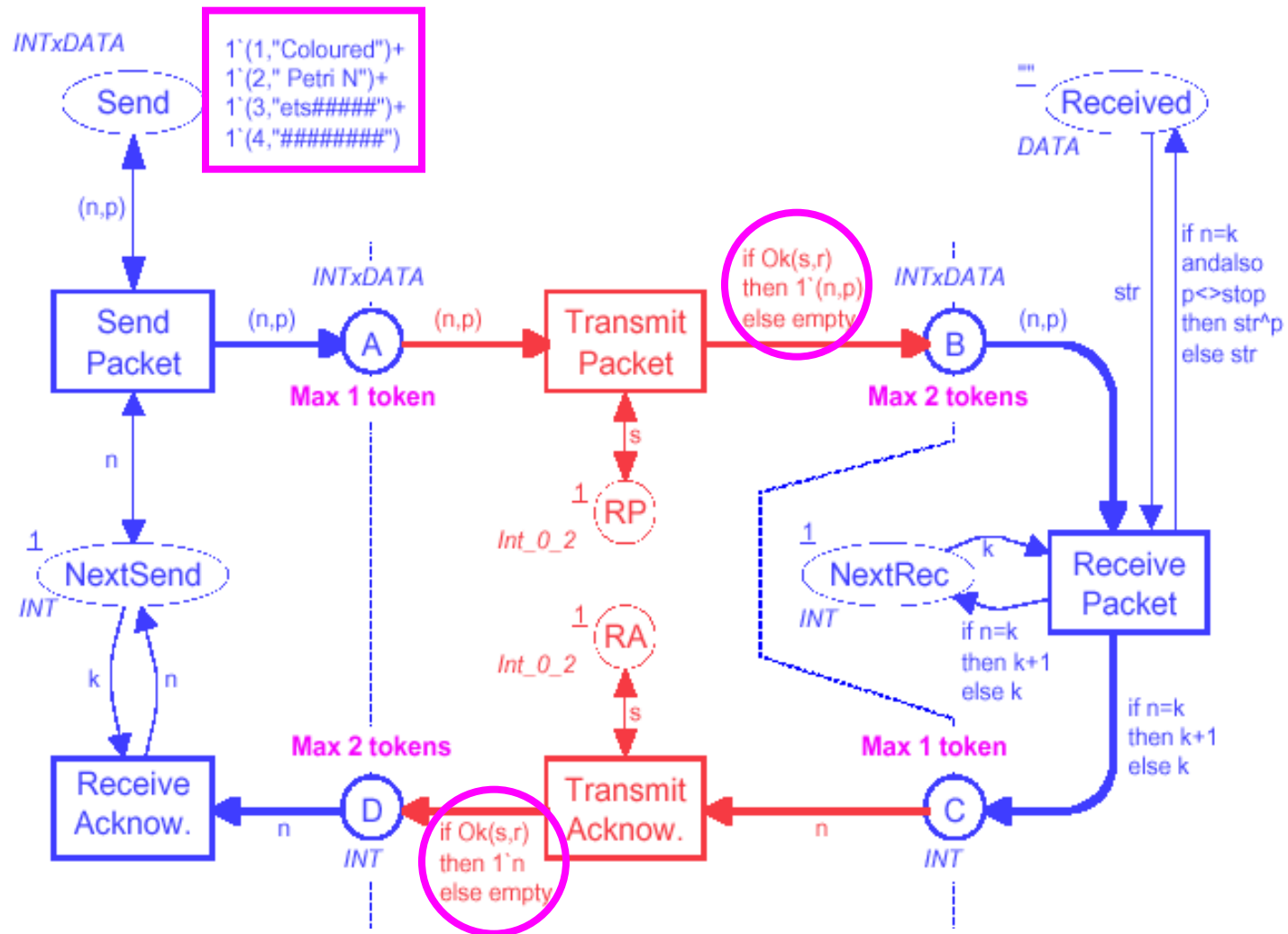
- ◆ A CPN verifikációja a következőkből áll:
 - *Állapottér.*
 - Hely és tranzíció *invariánsok.*
 - Hasonlóan a programozási nyelvek invariánsaihoz.

Állapottér

- ◆ Az állapottér egy irányított gráf, ahol:
 - A csomópont az *elérhető állapot*.
 - A nyíl a lehetséges tranzíciók (+ párosítás).



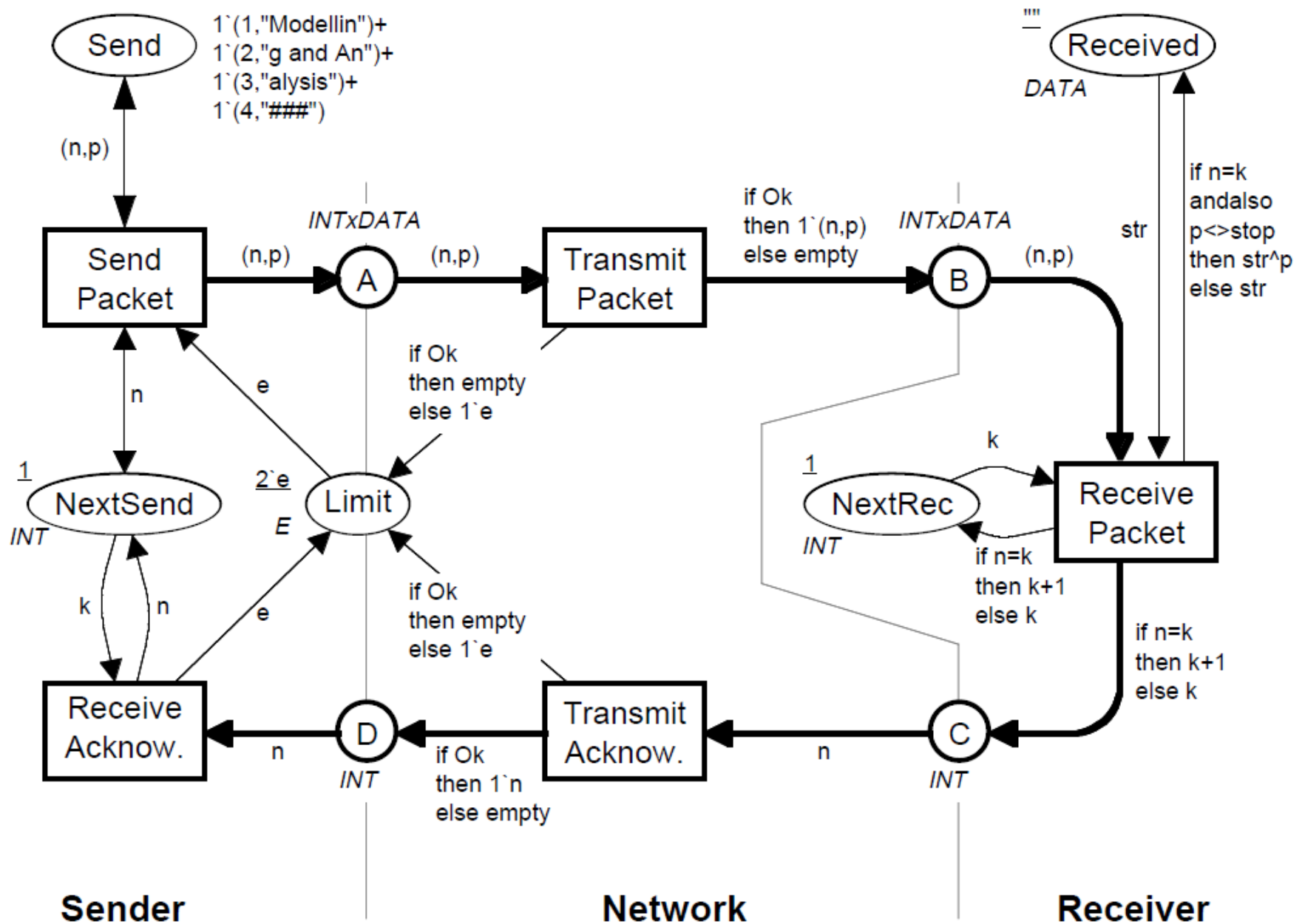
A protokoll állapottere



◆ Véges állapottér elérése:

- Tokenek számát korlátozzuk az A, B, C és D helyeken.
- Csak 4 csomagunk van.
- *Bináris választás, hogy sikeres vagy sem.*

INTxDATA



Állapottér információk a protokollunkról

Elérhetőségi gráf

Occurrence Graph Statistics

Nodes: 4298
Arcs: 15887
Secs: 16
Status: Full

Erősen összekötött komponensek:

Scc Graph Statistics

Nodes: 2406
Arcs: 11677
Secs: 5

Tokenszám korlátok

Upper Integer Bounds

A:	1
B:	2
C:	1
D:	2
NextRec:	1
NextSend:	1
RA:	1
RP:	1
Received:	1
Send:	4

Lower Integer Bounds

A:	0
B:	0
C:	0
D:	0
NextRec:	1
NextSend:	1
RA:	1
RP:	1
Received:	1
Send:	4

Multi-set korlátok

Upper Multi-set Bounds

A: $1^{\setminus}(1, \text{"Coloured"}) + 1^{\setminus}(2, \text{" Petri N"}) + 1^{\setminus}(3, \text{"ets#####"}) + 1^{\setminus}(4, \text{"#####"})$

B: $2^{\setminus}(1, \text{"Coloured"}) + 2^{\setminus}(2, \text{" Petri N"}) + 2^{\setminus}(3, \text{"ets#####"}) + 2^{\setminus}(4, \text{"#####"})$

C: $1^{\setminus}2 + 1^{\setminus}3 + 1^{\setminus}4 + 1^{\setminus}5$

D: $2^{\setminus}2 + 2^{\setminus}3 + 2^{\setminus}4 + 2^{\setminus}5$

NextRec: $1^{\setminus}1 + 1^{\setminus}2 + 1^{\setminus}3 + 1^{\setminus}4 + 1^{\setminus}5$

NextSend: $1^{\setminus}1 + 1^{\setminus}2 + 1^{\setminus}3 + 1^{\setminus}4 + 1^{\setminus}5$

RA: $1^{\setminus}1$

RP: $1^{\setminus}1$

Received: $1^{\setminus}"" + 1^{\setminus}\text{"Coloured"} + 1^{\setminus}\text{"Coloured Petri N"} + 1^{\setminus}\text{"Coloured Petri Nets#####"}$

Send: $1^{\setminus}(1, \text{"Coloured"}) + 1^{\setminus}(2, \text{" Petri N"}) + 1^{\setminus}(3, \text{"ets#####"}) + 1^{\setminus}(4, \text{"#####"})$

Visszatérőség és élőség

Visszatérőség

Home Markings: 1 [452]

Élőség

Dead Markings: 1 [452]

Live Transitions: None

NextSend = 5

NextRec = 5

Received = "Coloured Petri Nets####"

452

- ◆ A 452 token eloszlás azt jelenti, hogy minden csomag helyesen megérkezett.

A halott állapotok

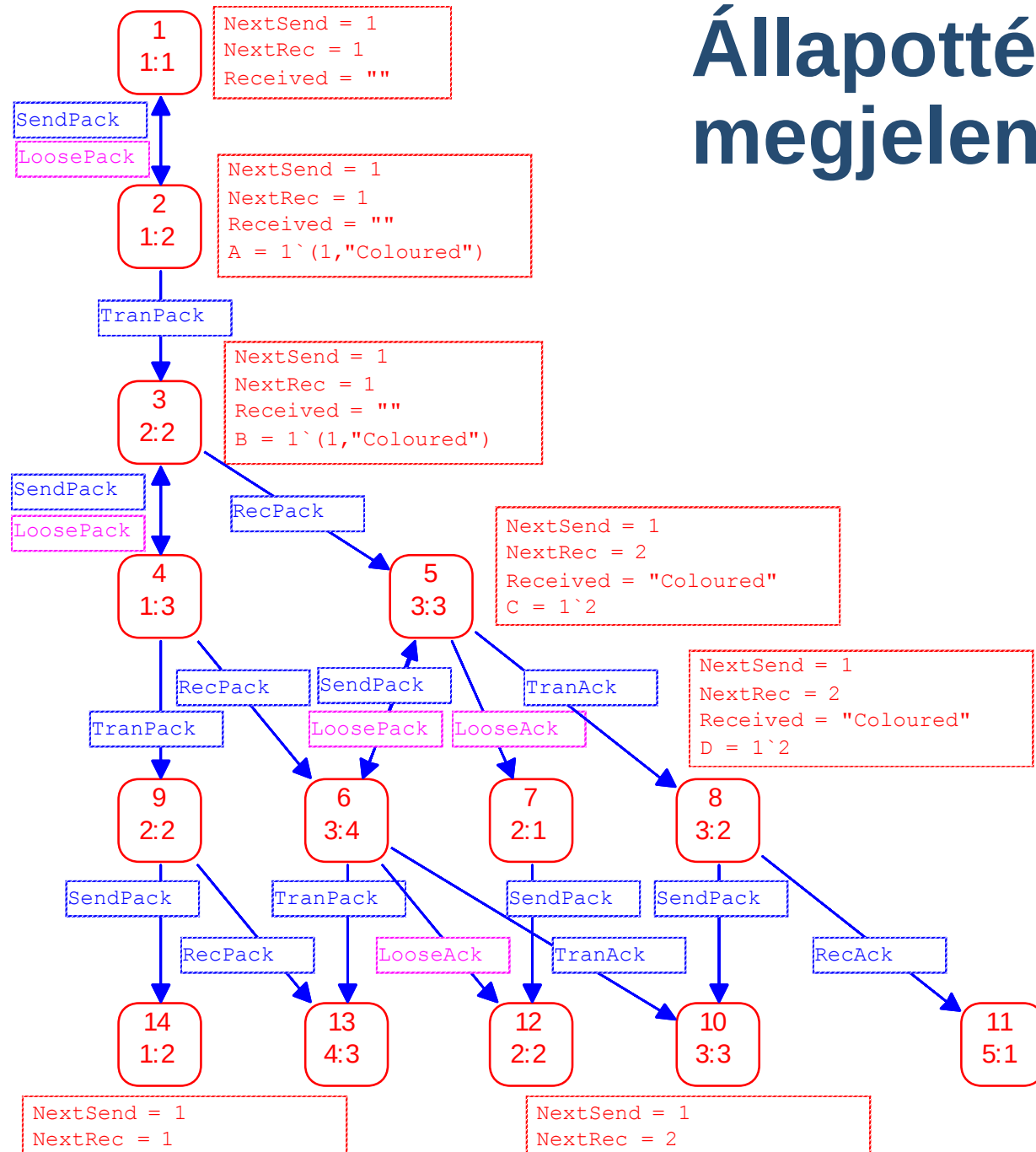
- ◆ A 452. tokeneloszlás a *dead*
 - Ez azt jelenti, hogy ez a protokoll *részlegesen korrekt* (ha lefut az algoritmus, akkor a végleges eloszlást veszi fel).
- ◆ Ez egyben *visszatérő állapot is*
 - Emiatt *mindig megvan a lehetőség arra, hogy helyesen fejezzük be* (az lehetetlen, hogy olyan állapotba tévedjünk, amiből nem tudunk a kívánt célállapotba jutni).

Fair tulajdonság

Send Packet:	Impartial
Transmit Packet:	Impartial
Receive Packet:	No Fairness
Transmit Acknow:	No Fairness
Receive Acknow:	No Fairness

- ◆ A fairség tulajdonság azt jelenti, hogy egy tranzíció milyen gyakran tüzelhet.

Állapottér megjelenítése



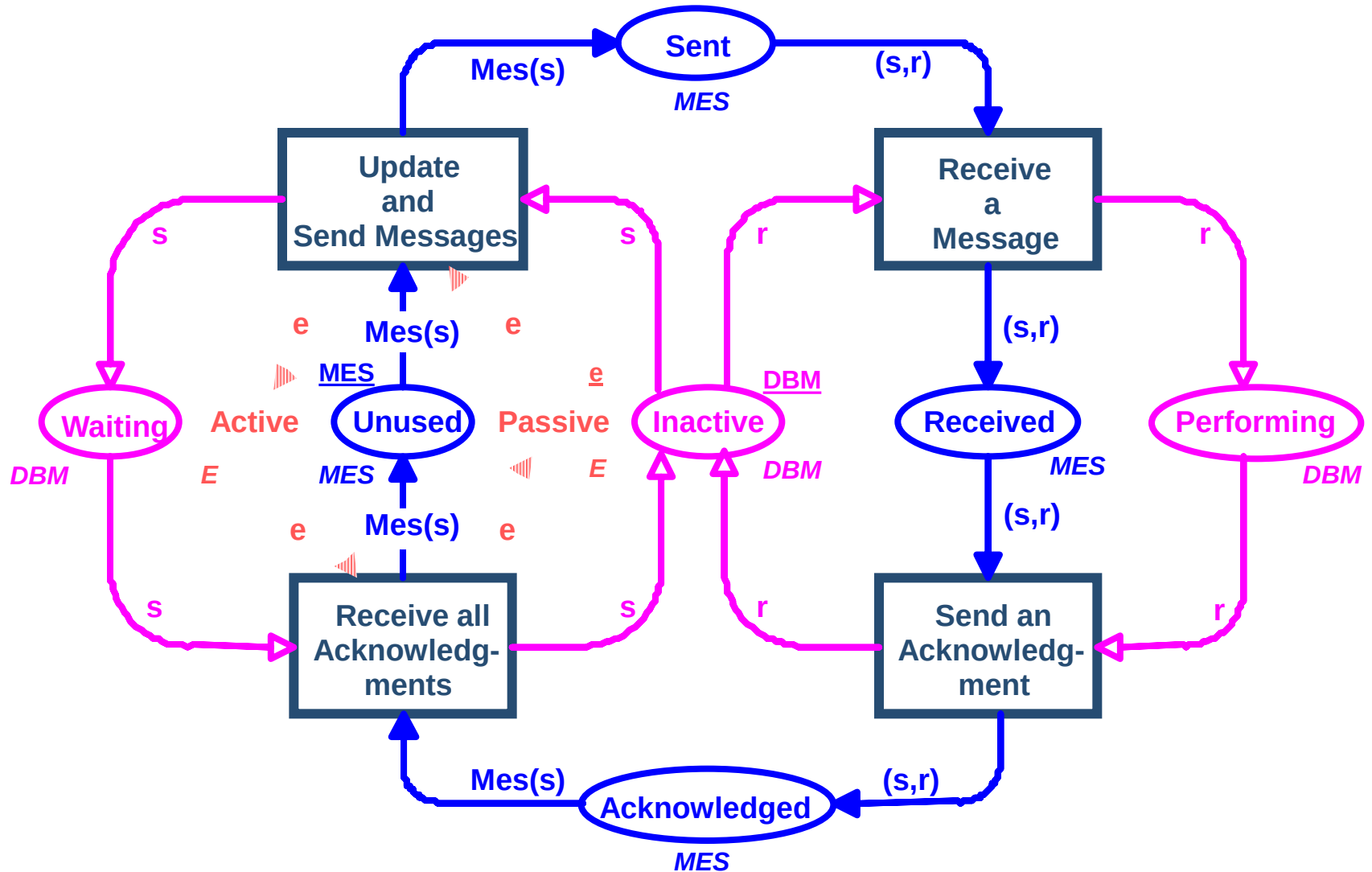
Temporális logika

- ◆ Lehetséges CTL-szerű kérdéseket is feltenni (*temporális logika*).
 - Állapotok
 - Átmenetek
 - Kötések elemei

Állapottér analízis – előnyök/hátrányok

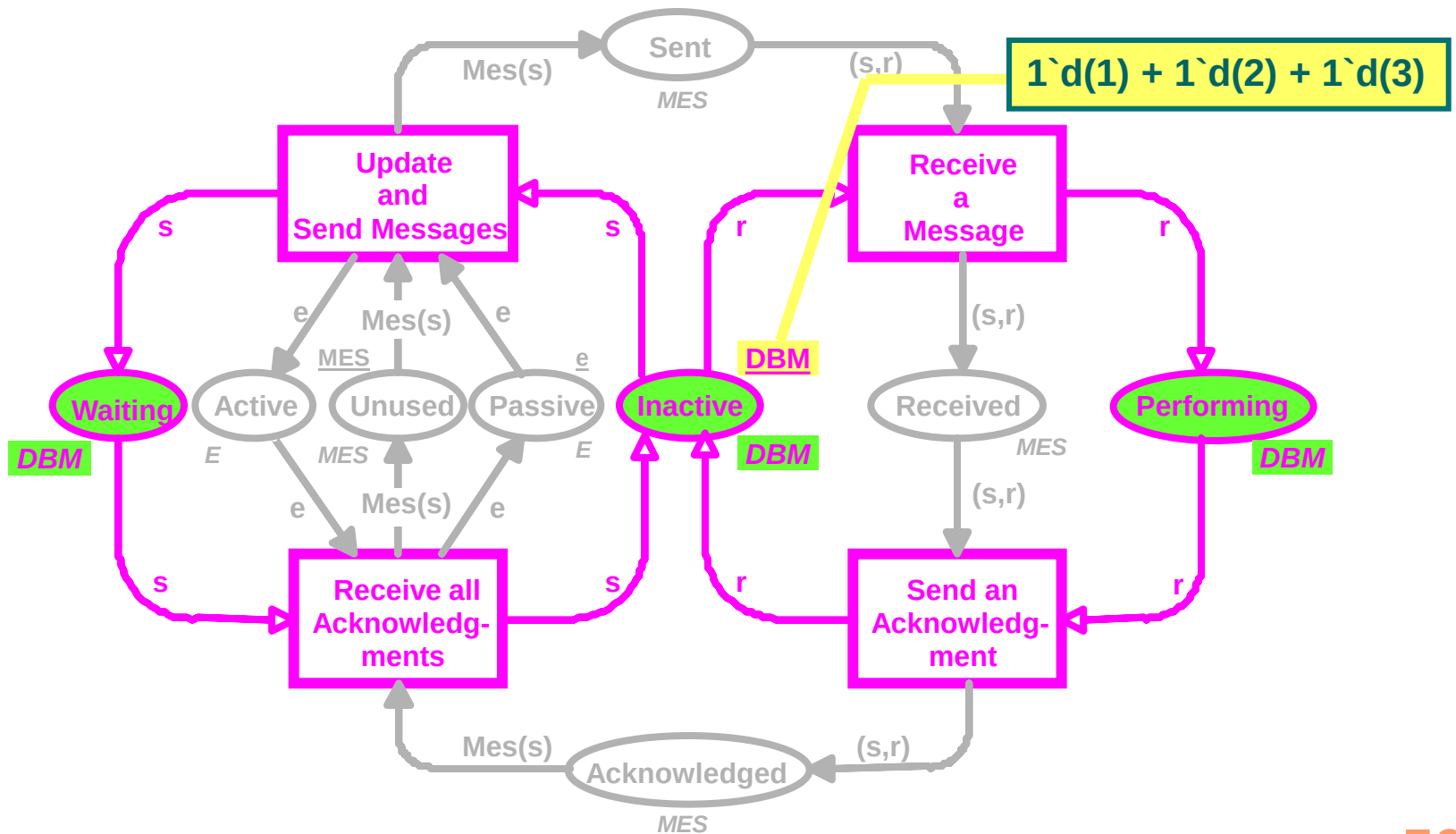
- ◆ Az állapottér *kifejezőereje nagy* és *könnyen* számolható.
 - Építése és analízise *automatizálható*.
 - *Nincs szükség* az analízis módszerek *matematikai* ismeretére
- ◆ Legnagyobb hátránya az *állapottér robbanás*
 - Az eszköz *egymillió* állapotot képes kezelni.
 - Ez sok esetben *nem elég*.

Elosztott adatbázis



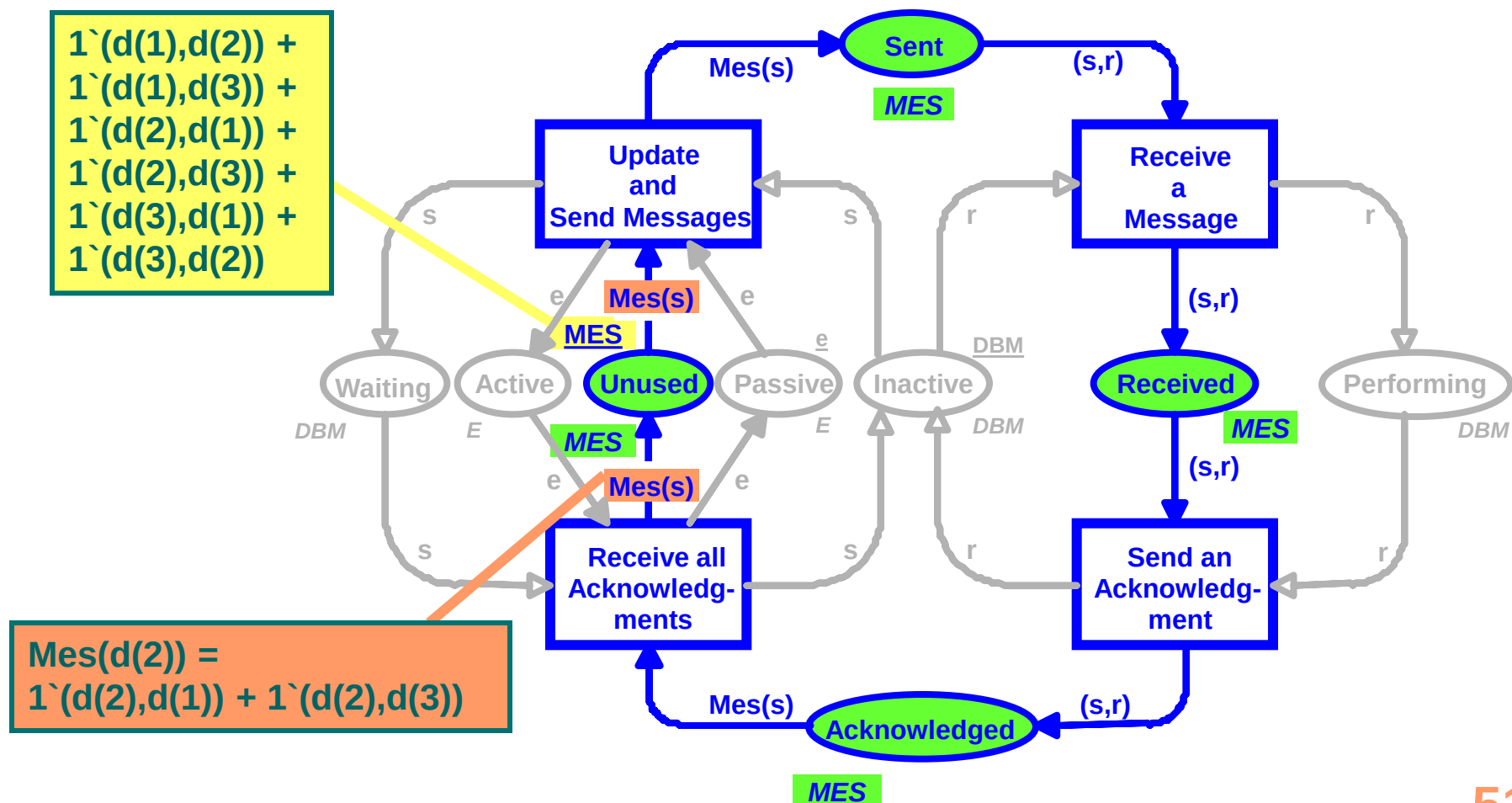
Adatbázis kezelők

◆ DBM = {d(1),d(2),d(3)}



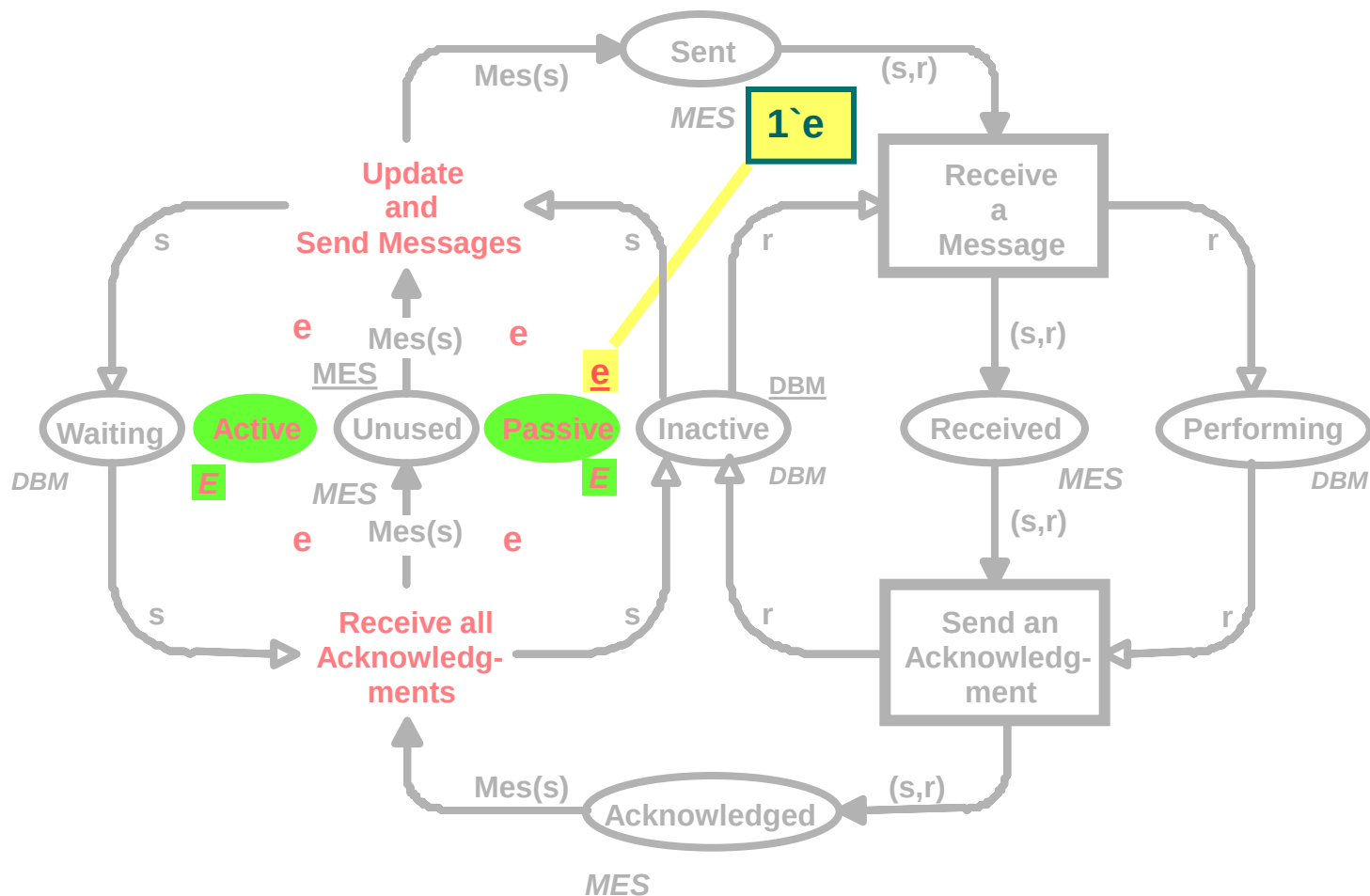
Üzenet bufferek

◆ $MES = \{(s,r) \in DBM \times DBM \mid s \neq r\}$

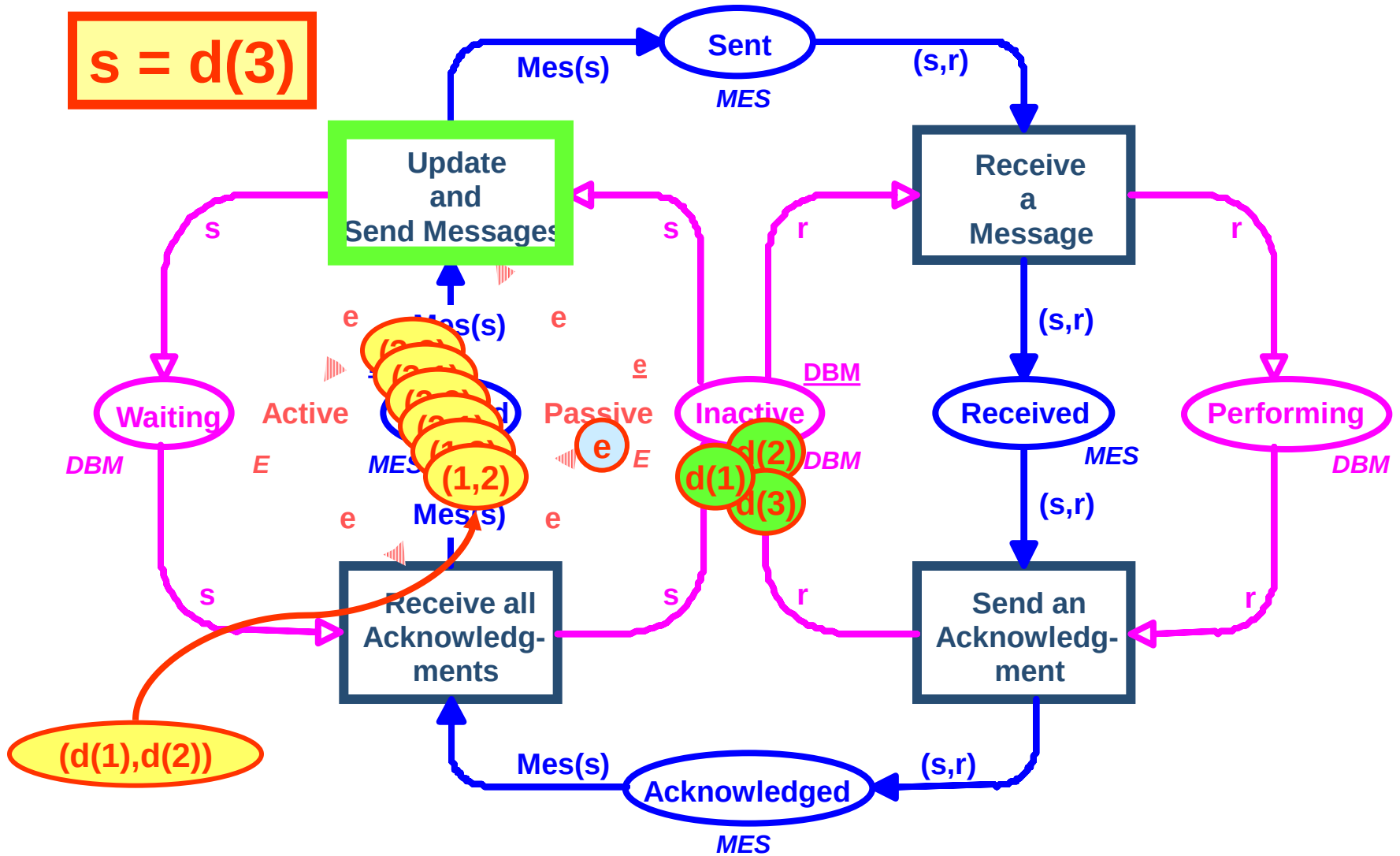


Kölcsönös kizárás

◆ $E = \{e\}$

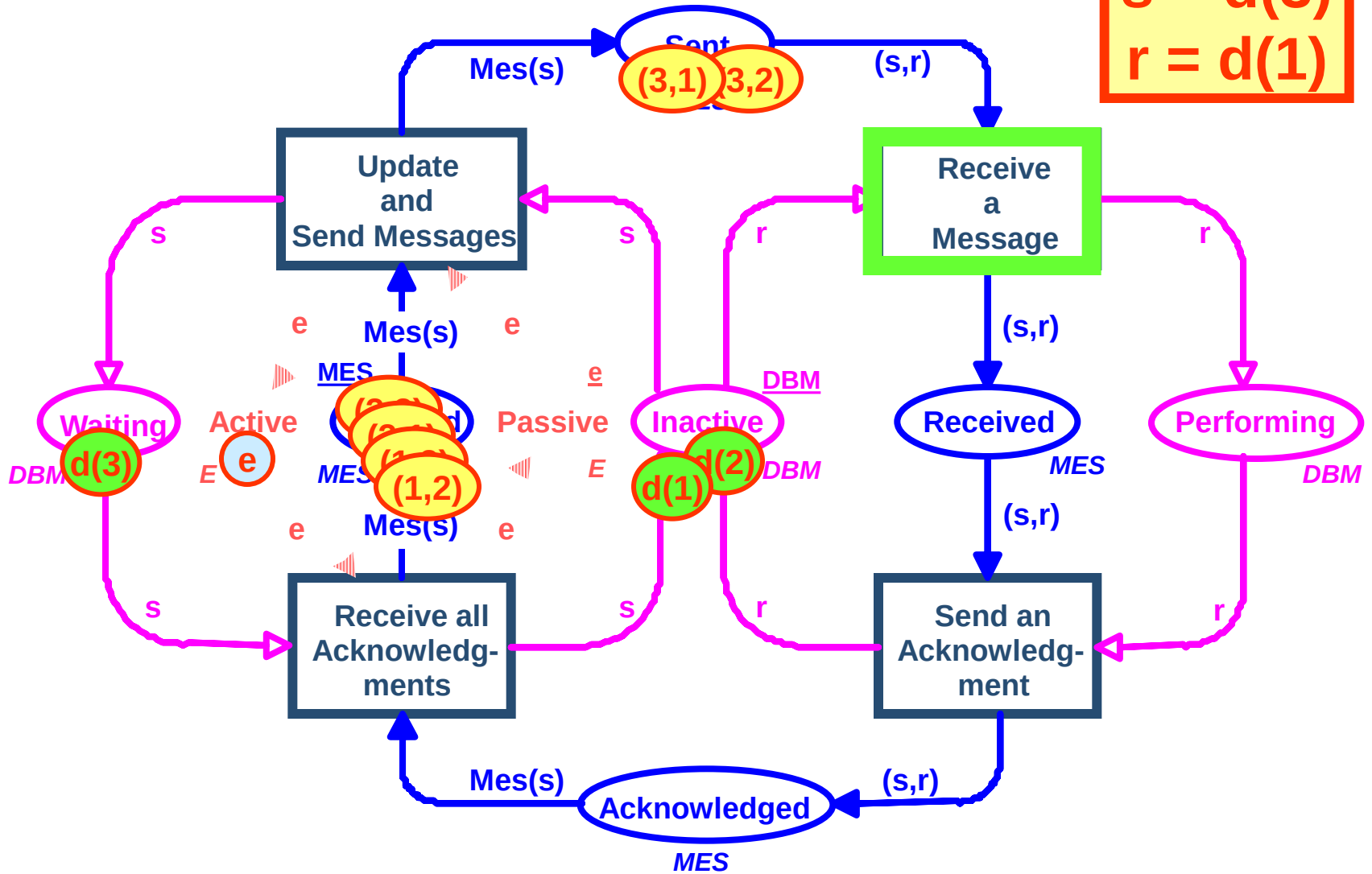


Elosztott adatbázis

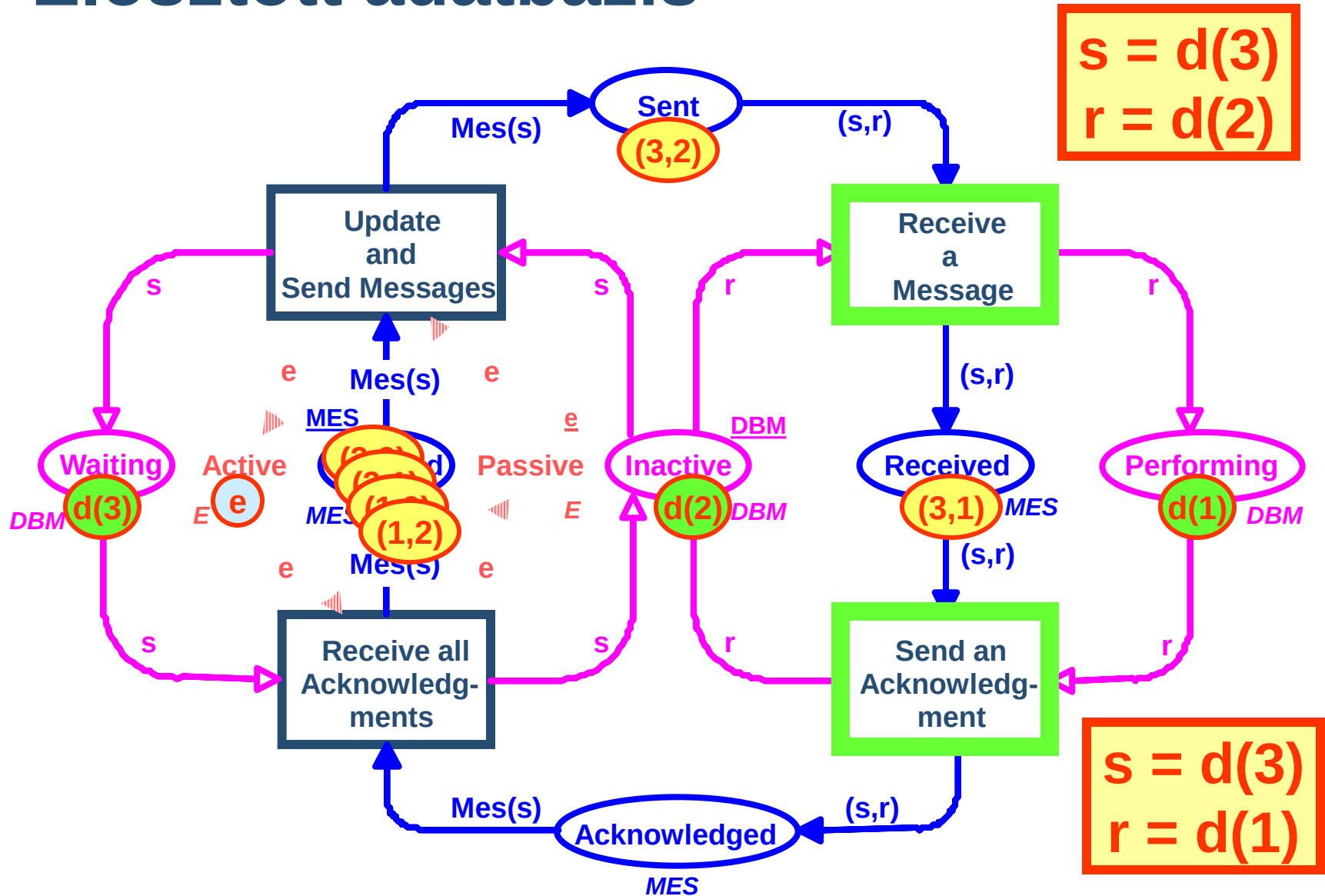


Elosztott adatbázis

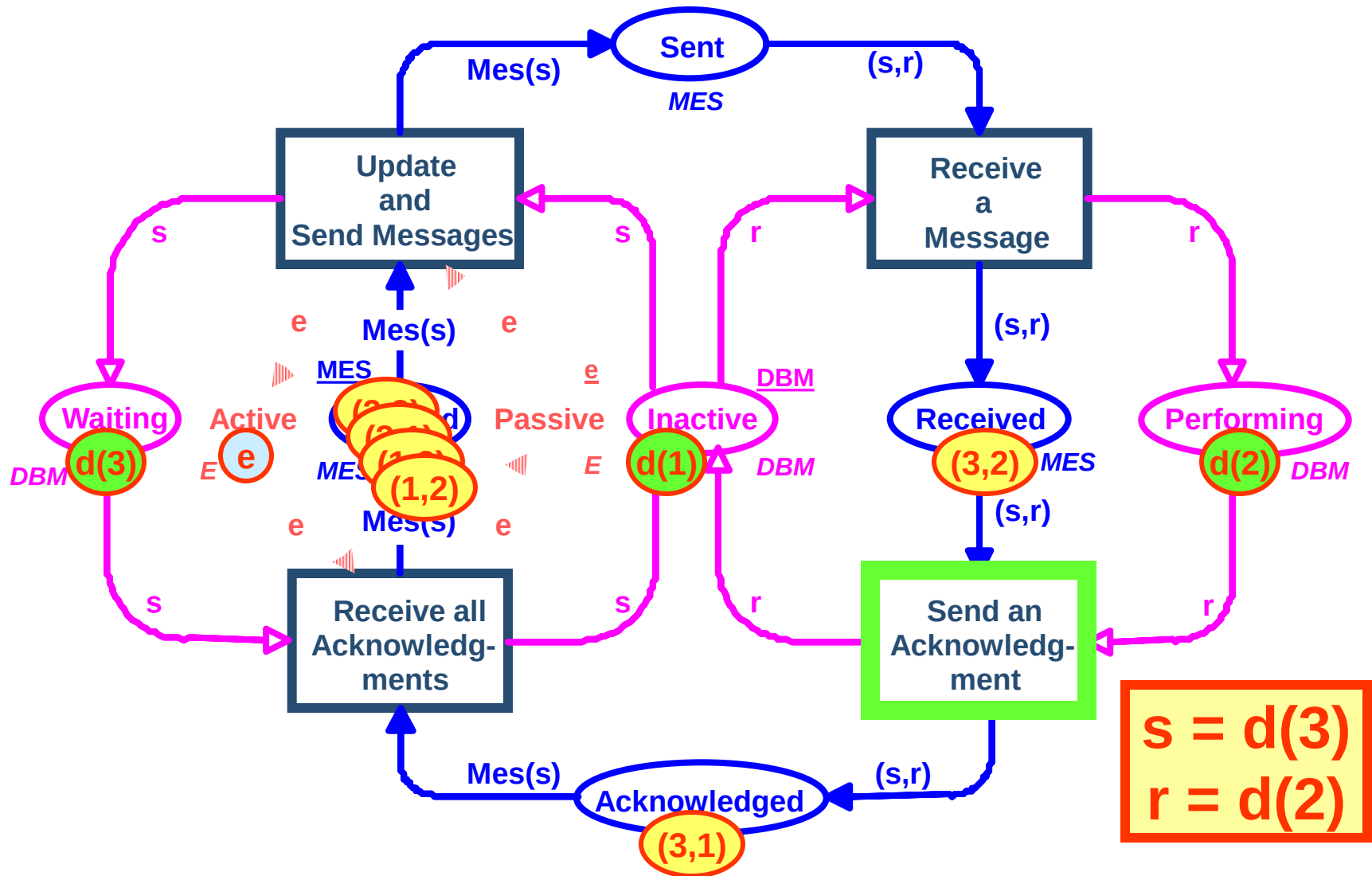
$s = d(3)$
 $r = d(1)$



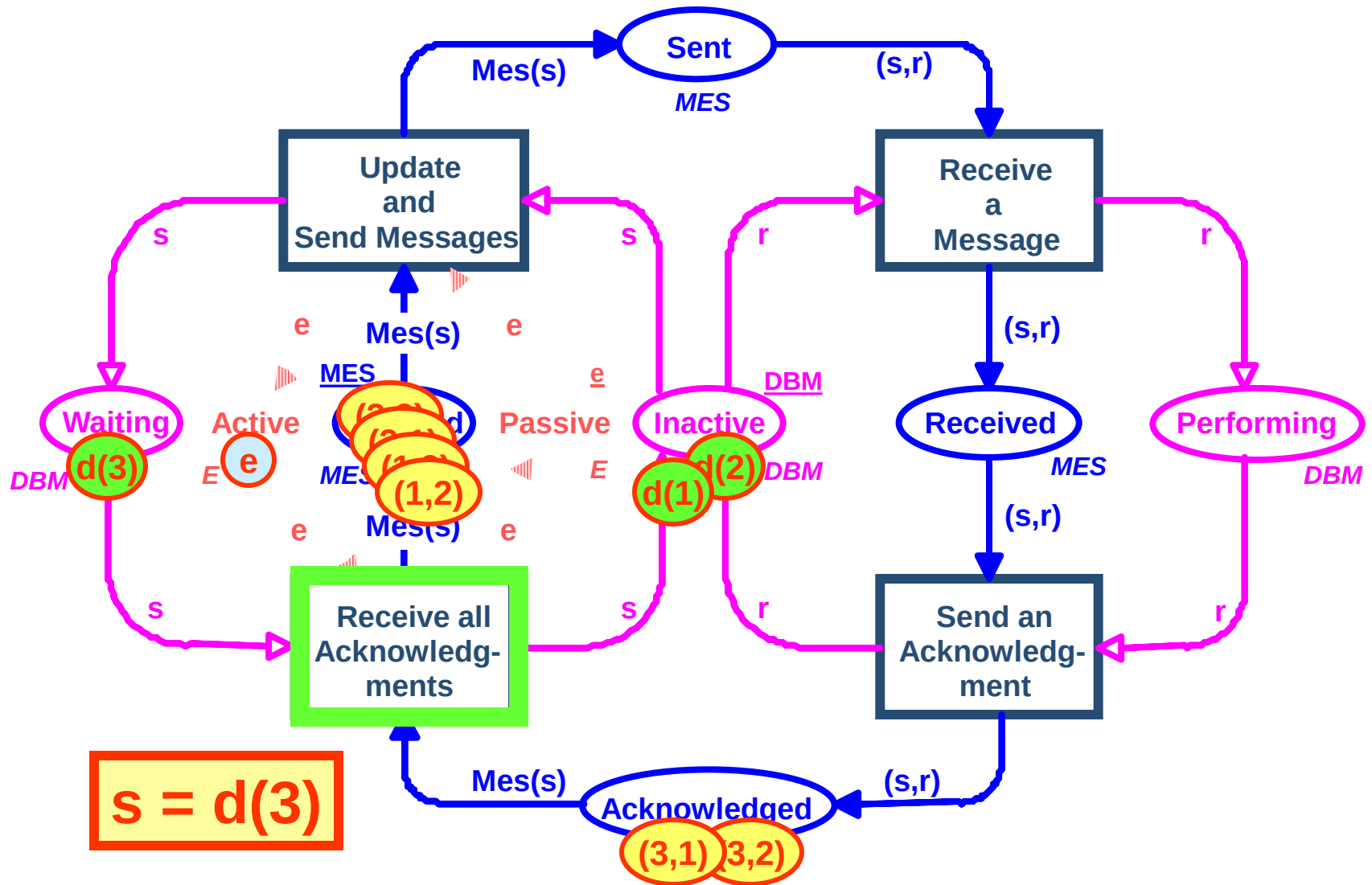
Elosztott adatbázis



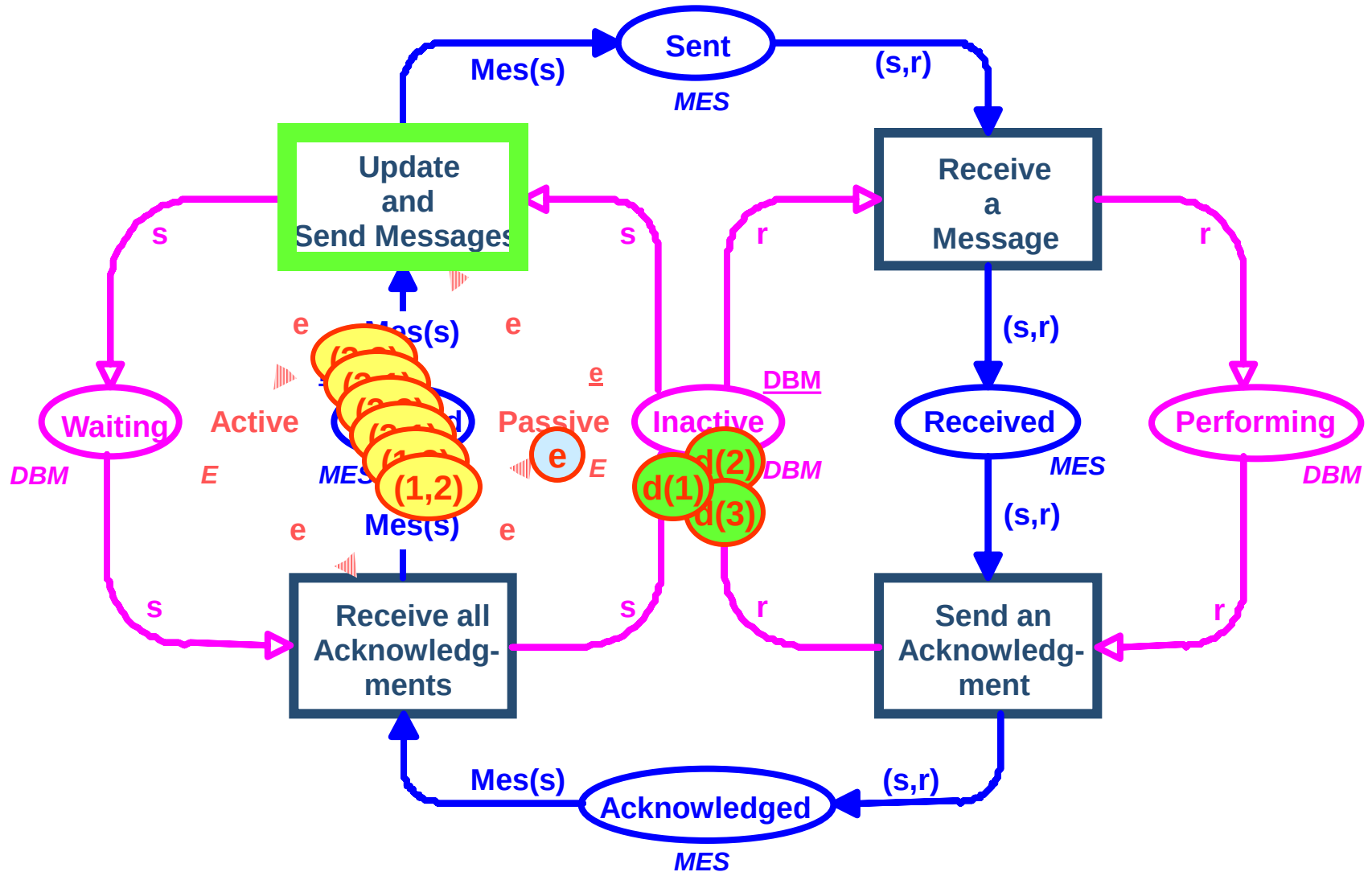
Elosztott adatbázis



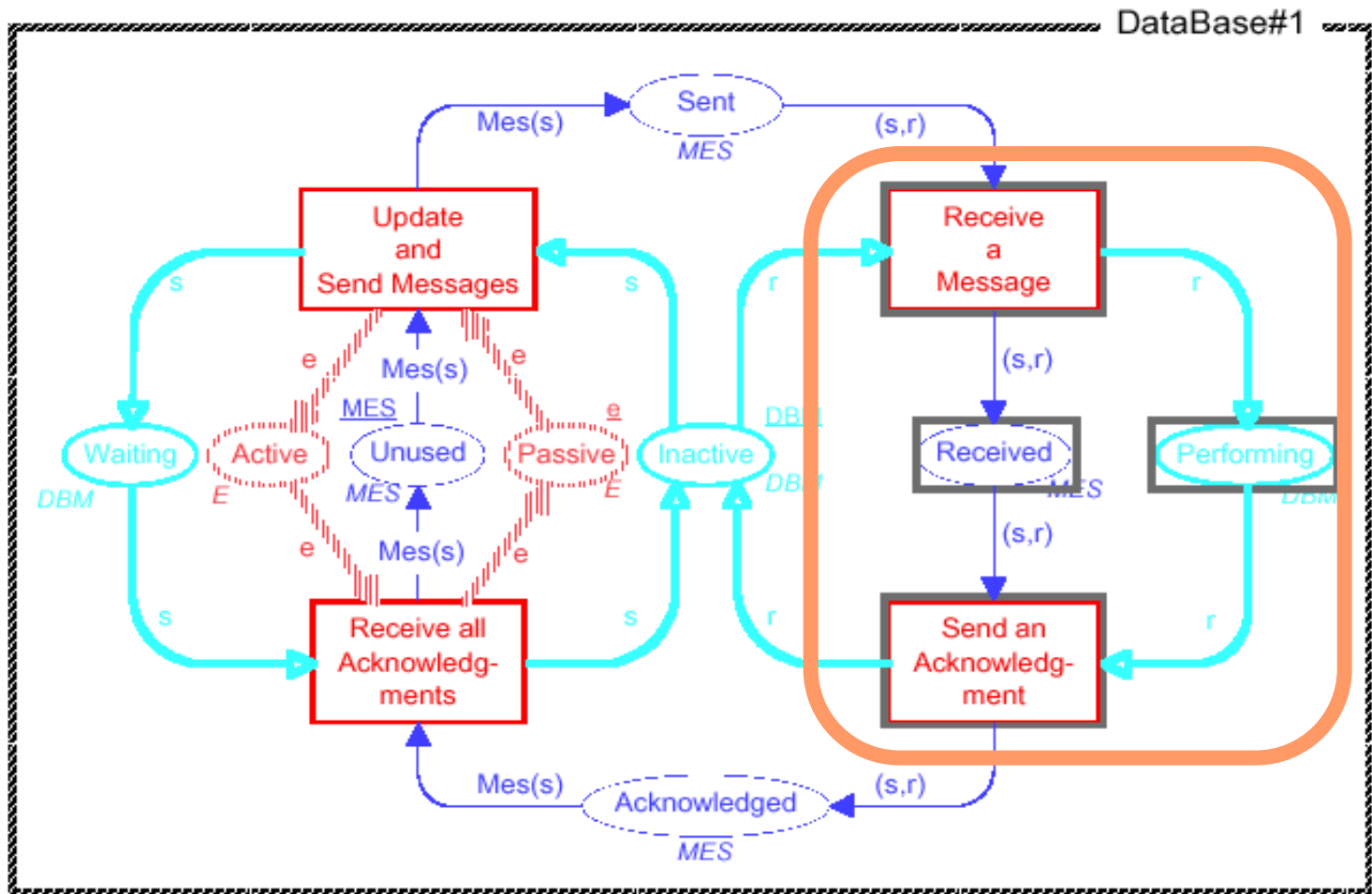
Elosztott adatbázis



Elosztott adatbázis



Hierarchikus modellek támogatása



- ◆ A *kijelölt részt* könnyen helyettesíthetjük egyetlen alhálóval.

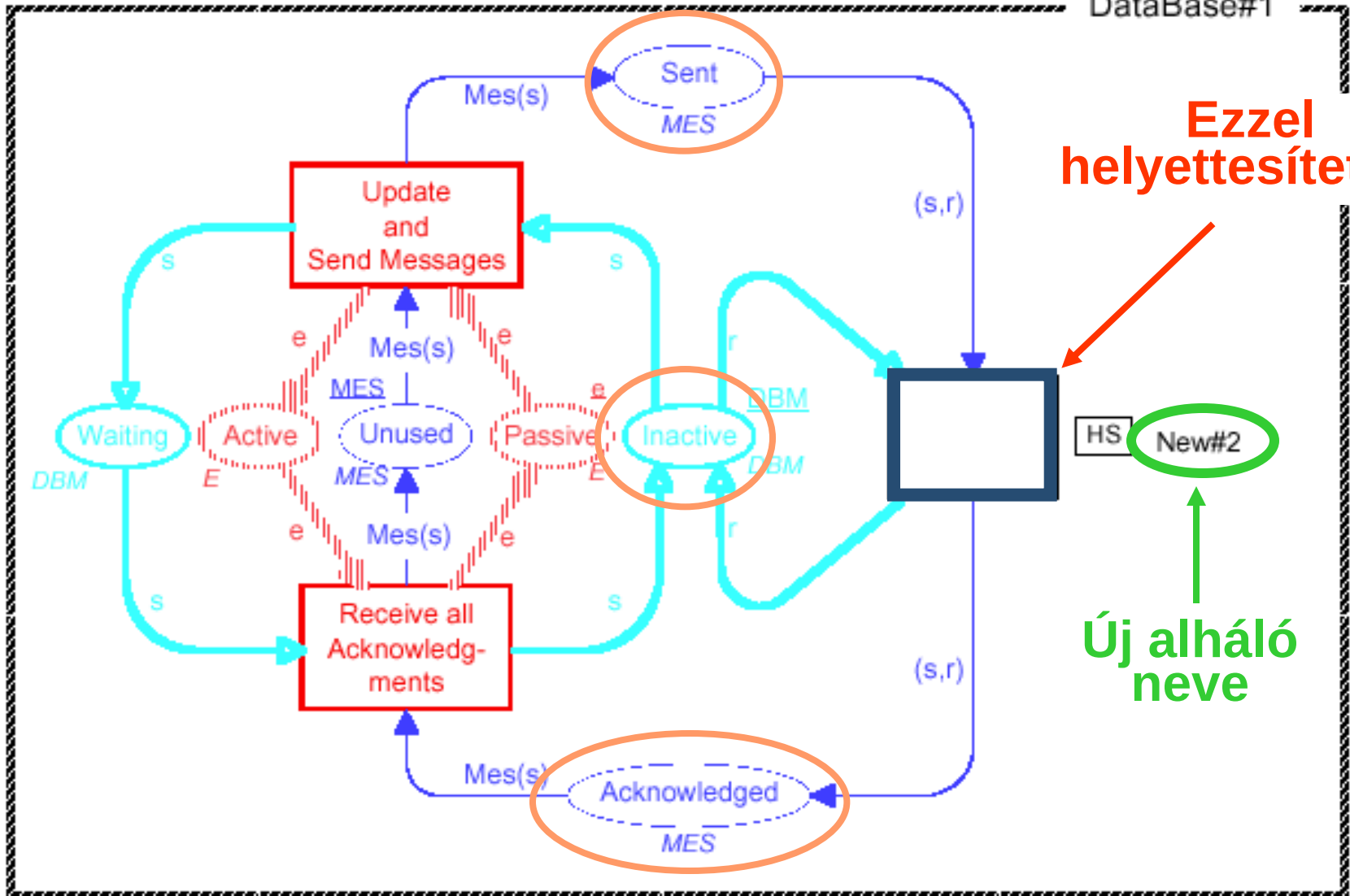
Kompakt nézet

Sockets (interfész)

DataBase#1

Ezzel helyettesítettük

Új alháló neve



Az alháló neve



Hely invariánsok

Place $\rightarrow$ M(Place)

◆ Waiting + Inactive + Performing = DBM

◆ Unused + Sent + Receive + Acknowledged = MES

◆ Active + Passive = E

◆ $Rec(Received) = Performing$

◆ $Mes(Waiting) = Sent + Received + Acknowledge$

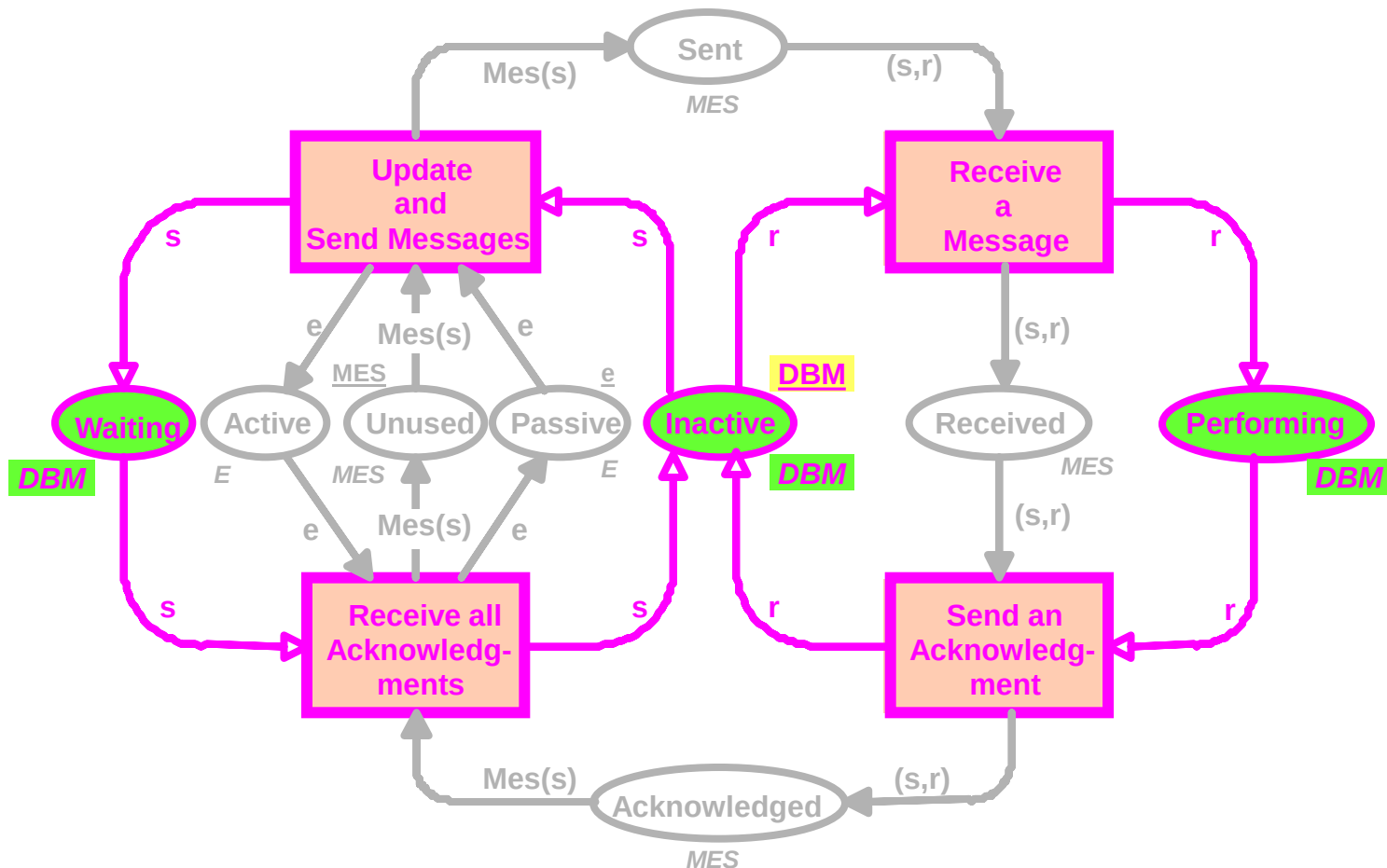
◆ $Ign(Waiting) = Active$

Ezek *lineáris kombinációival* gyárthatunk többet.

◆ $Ign(Waiting) + Passive = E$

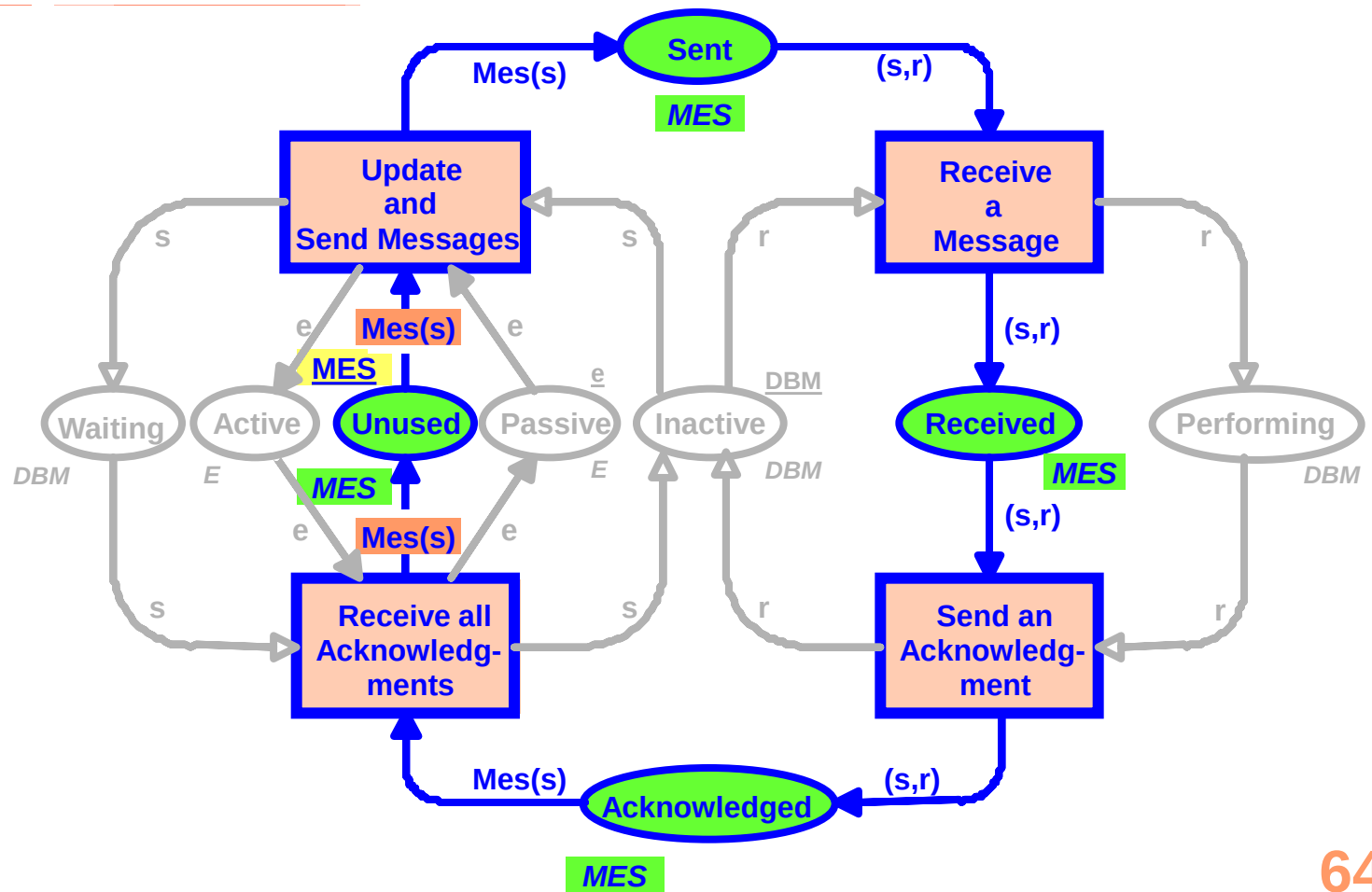
Adatbázis kezelők

$$M(\text{Waiting}) + M(\text{Inactive}) + M(\text{Performing}) = \text{DBM}$$



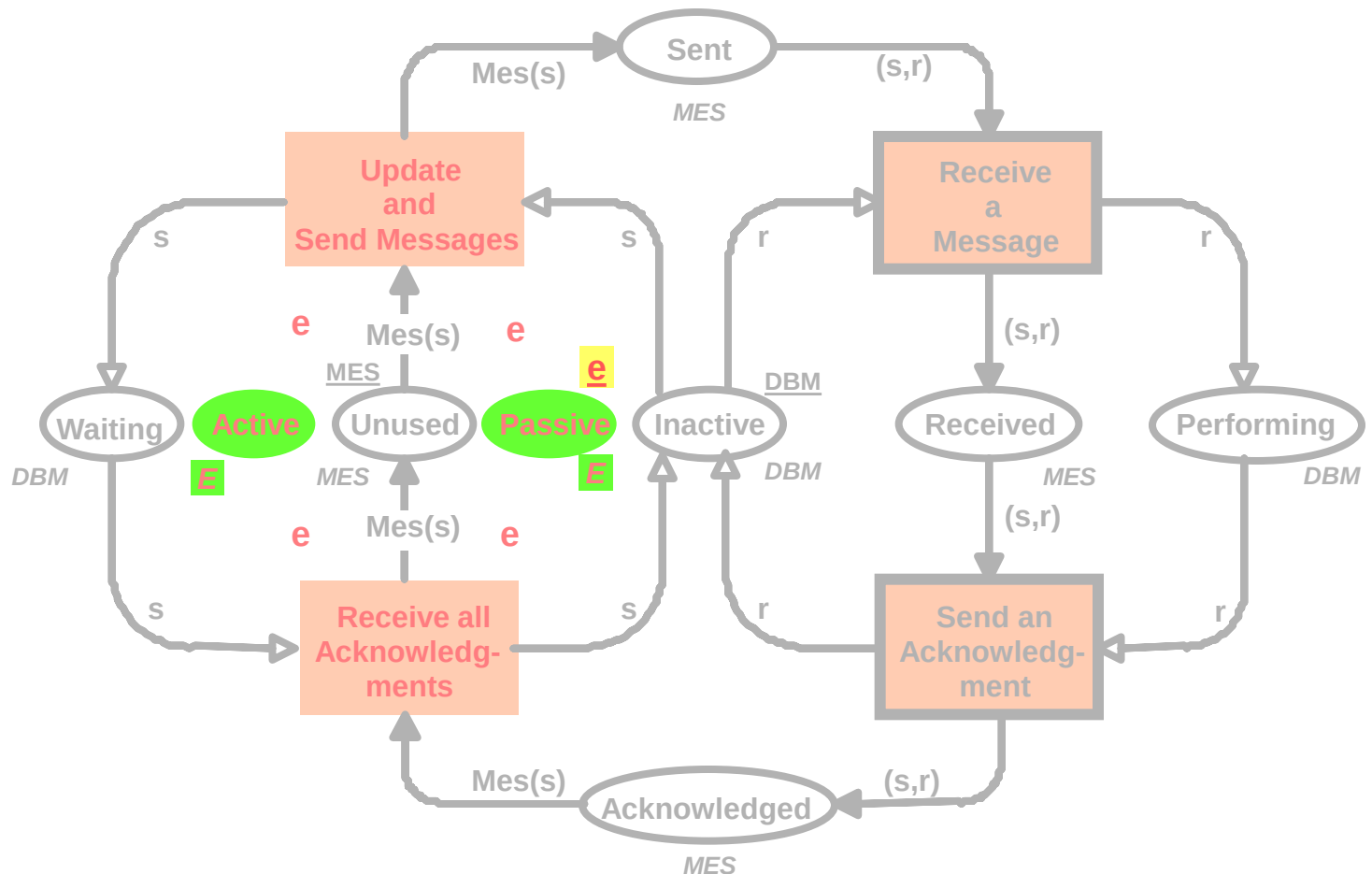
Üzenet bufferek

$$M(\text{Unused}) + M(\text{Sent}) + M(\text{Received}) + M(\text{Acknowledged}) = \text{MES}$$



Kölcsönös kizárás

$$M(\text{Active}) + M(\text{Passive}) = E$$



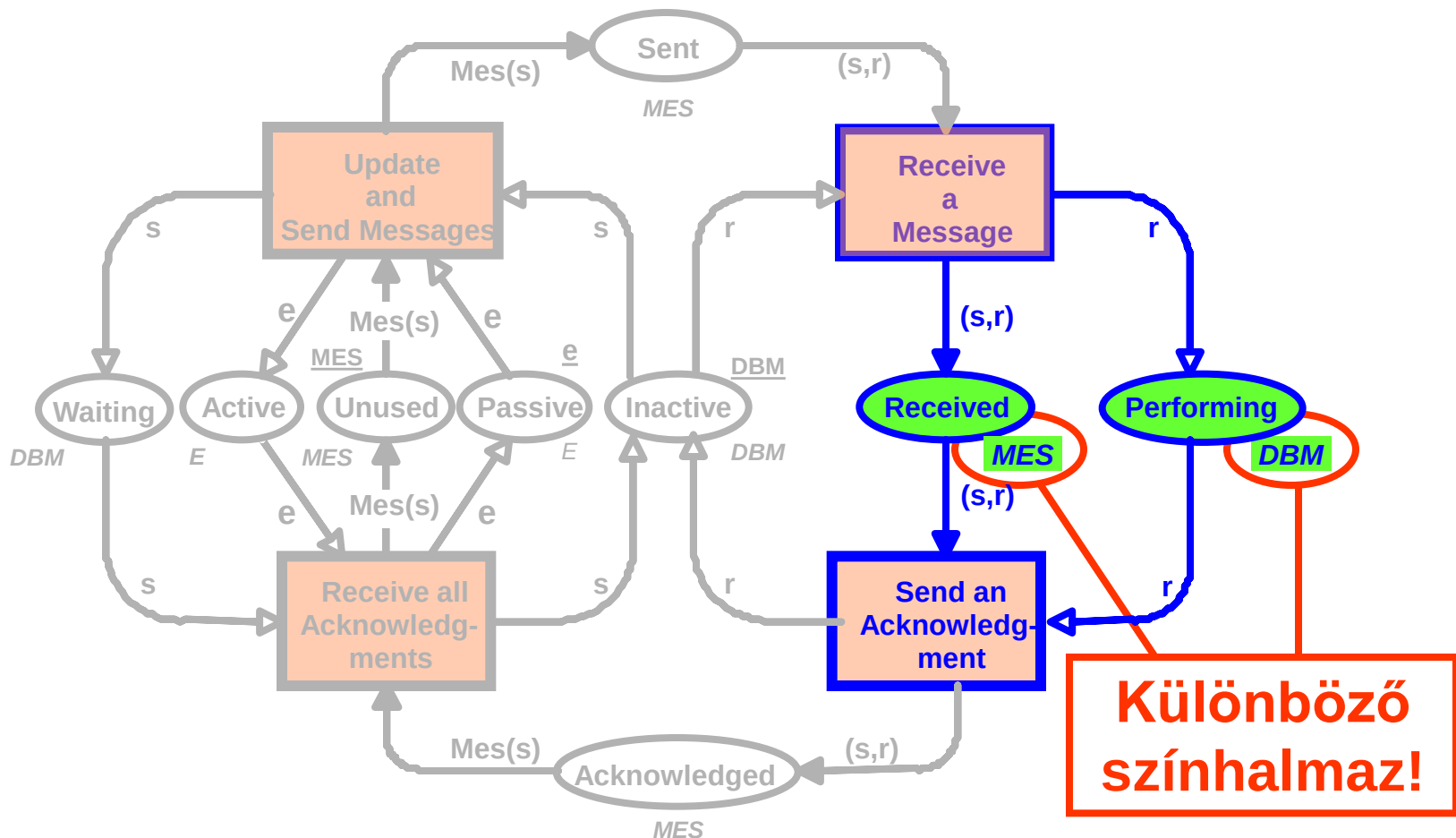
Fogadott üzenetek

$$\text{Rec}(s,r) = r$$

**Súly
függvény**

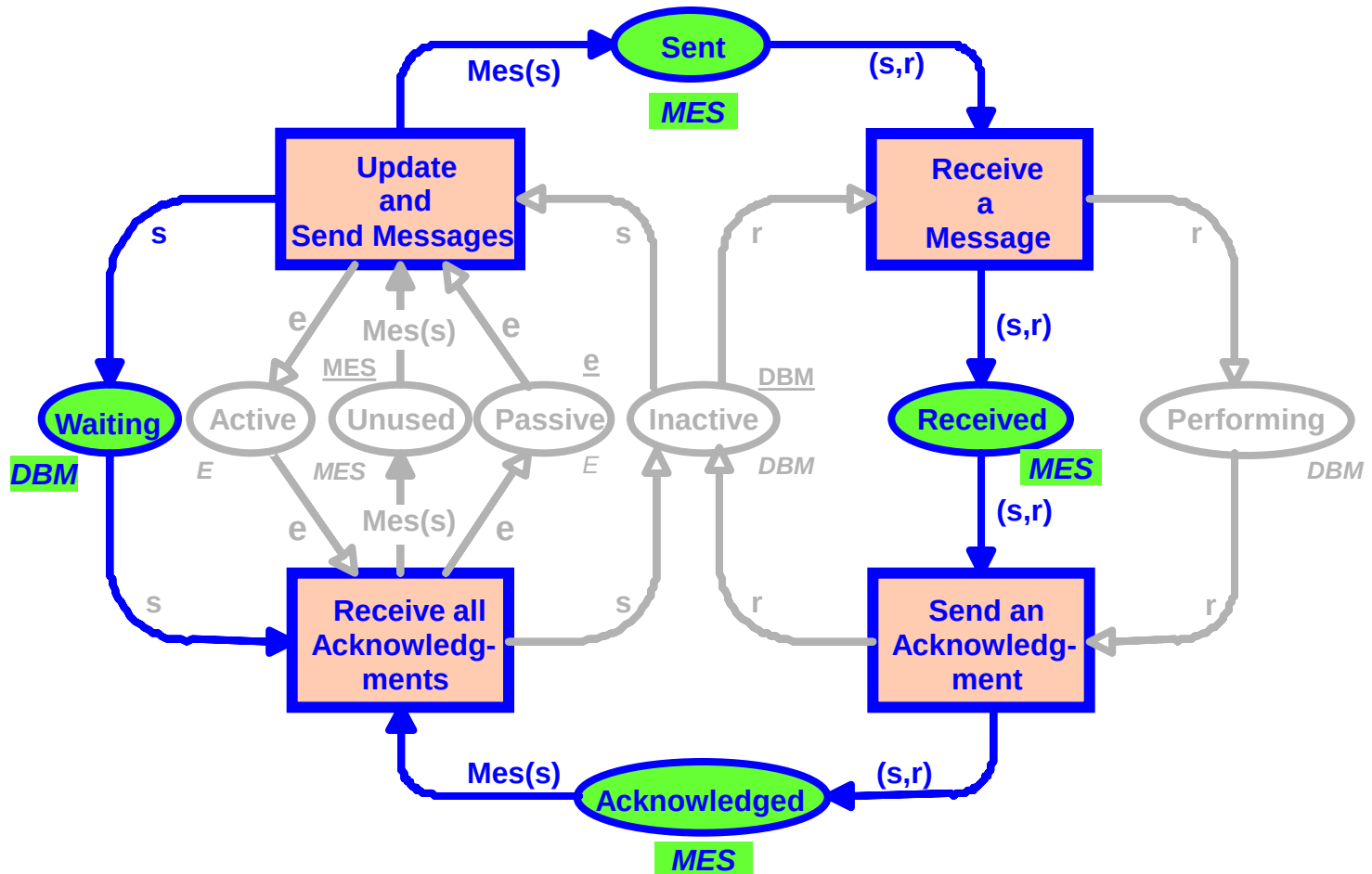
MES $\rightarrow$ DBM

$$\text{Rec}(M(\text{Received})) = M(\text{Performing})$$



Használt üzenetek

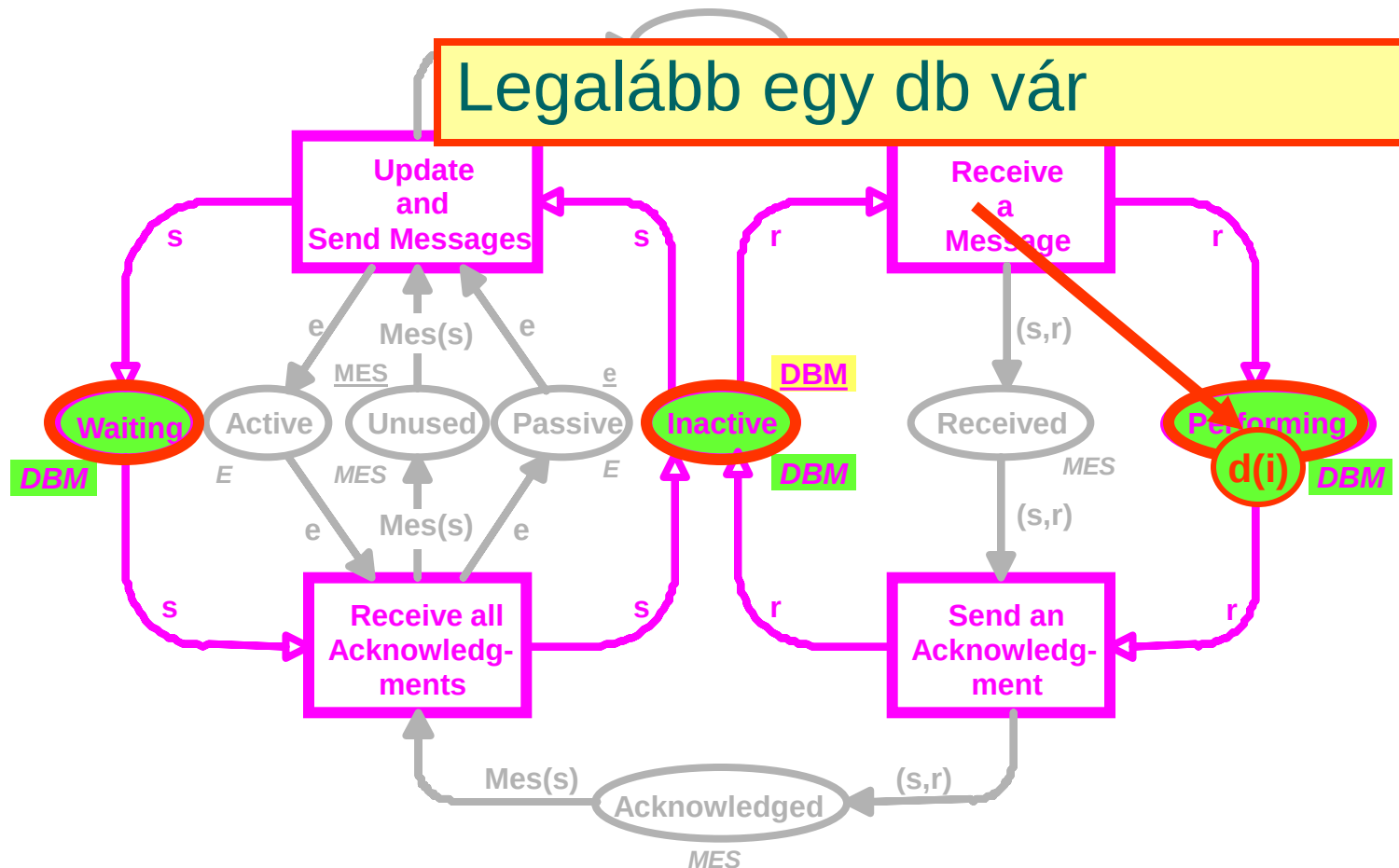
$$\text{Mes}(\text{Waiting}) = \text{M}(\text{Sent}) + \text{M}(\text{Received}) + \text{M}(\text{Acknowledged})$$



$$M(\text{Waiting}) + M(\text{Inactive}) + M(\text{Performing}) = \text{DBM}$$

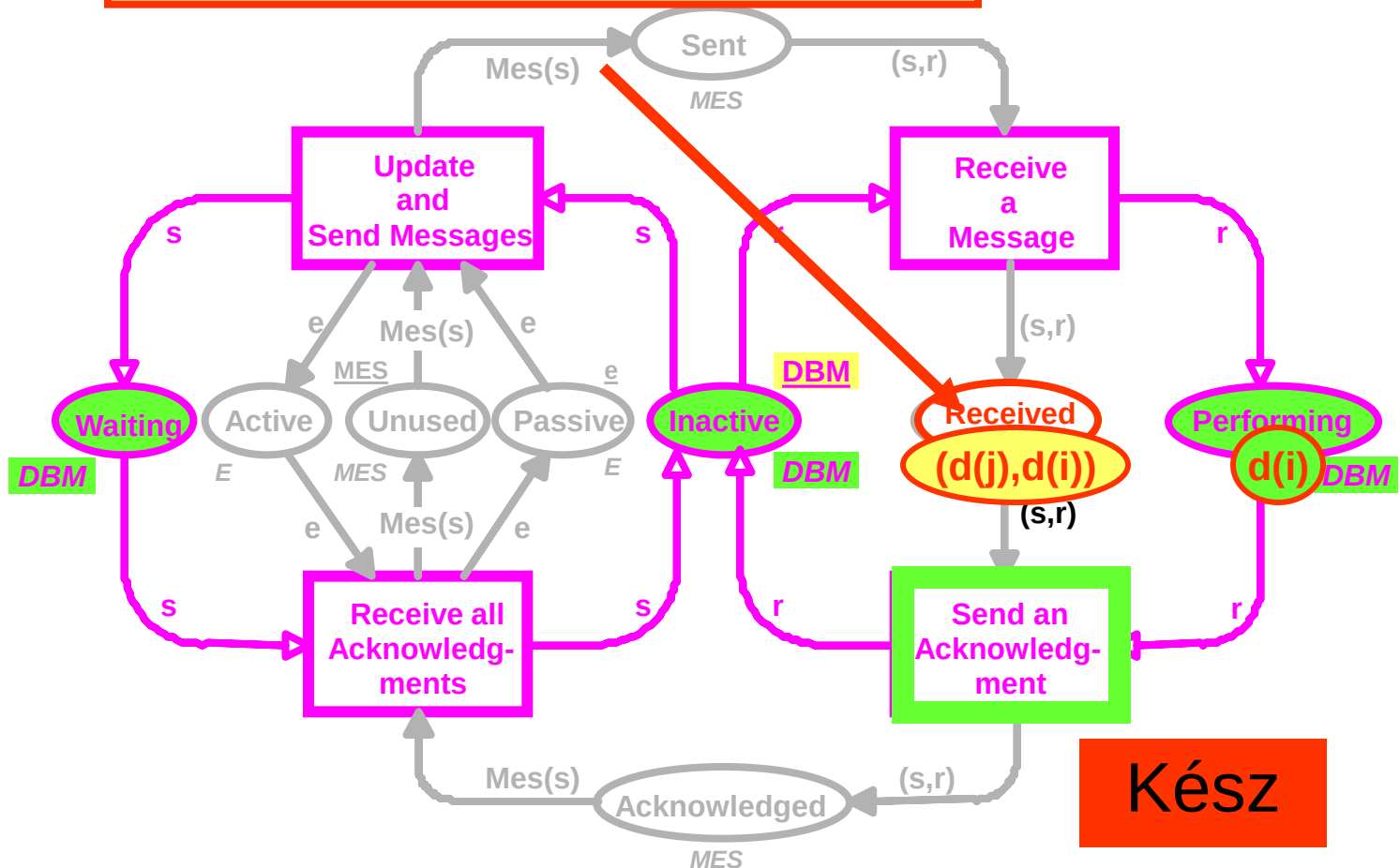
Minden adatkezelő:

Legalább egy db vár

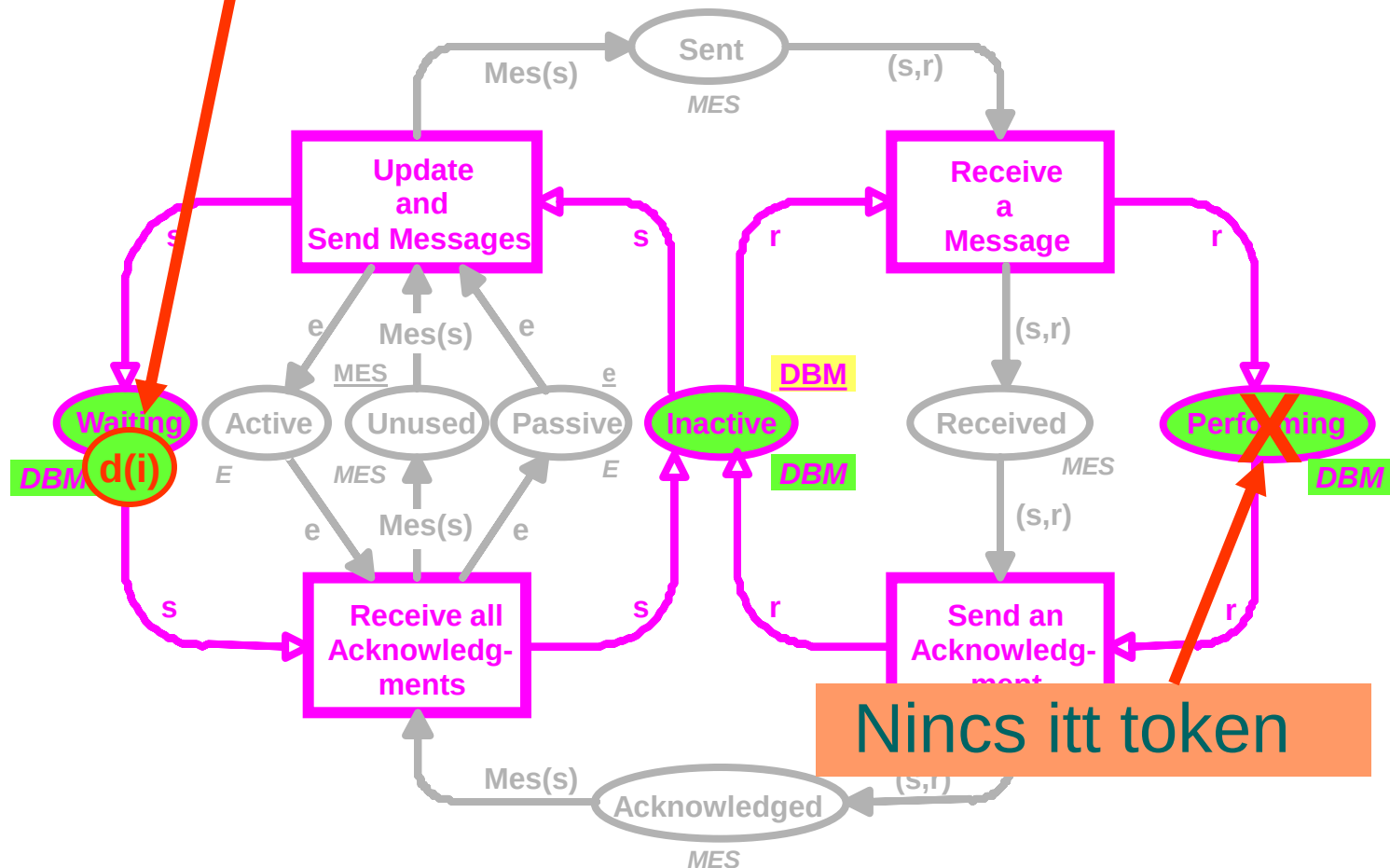


$\text{Rec}(M(\text{Received})) = M(\text{Performing})$

Ez az üzenet buffer:
Received with $d(i)$

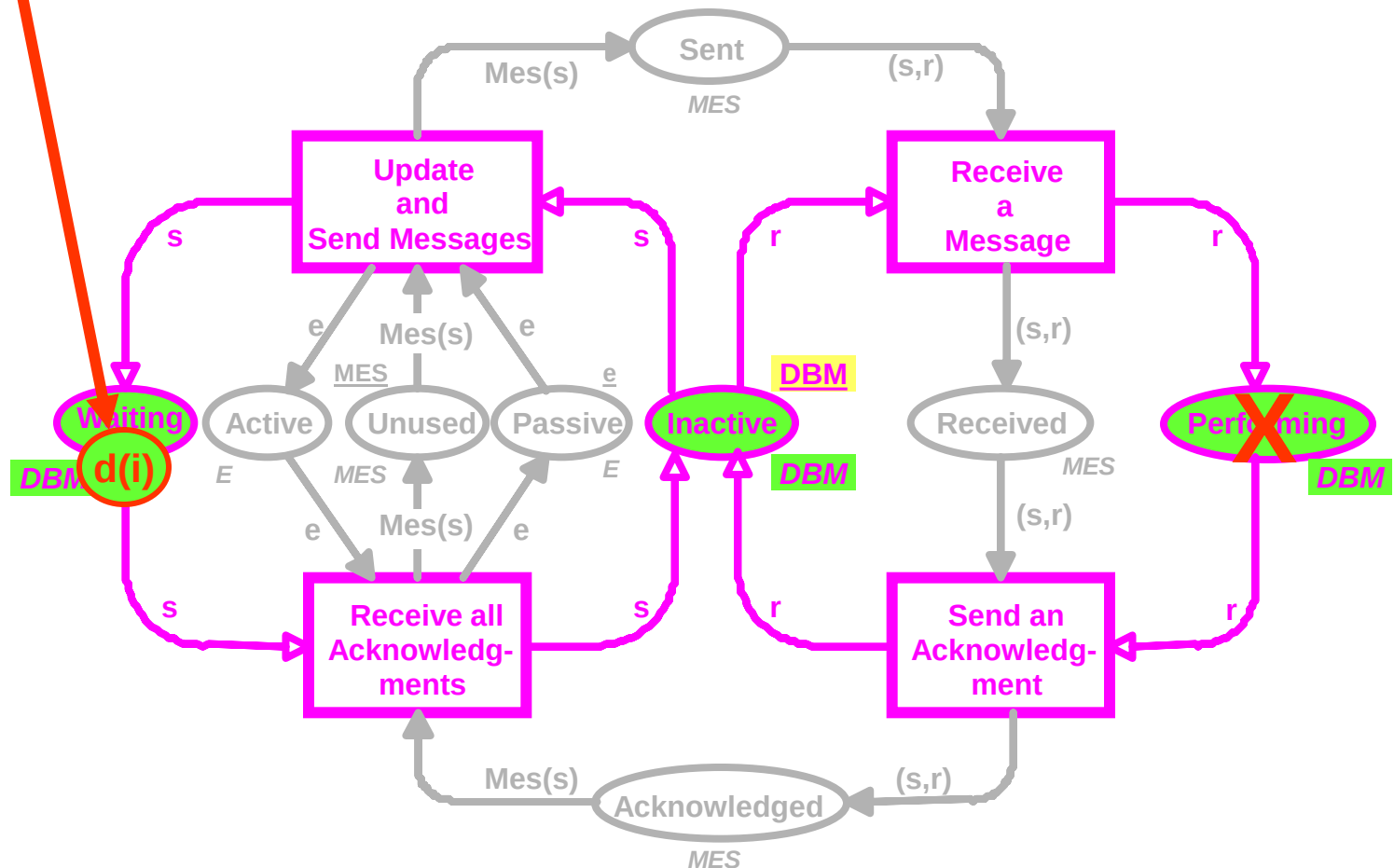


Most azt tesszük fel, hogy
legalább egy kezelő VÁR



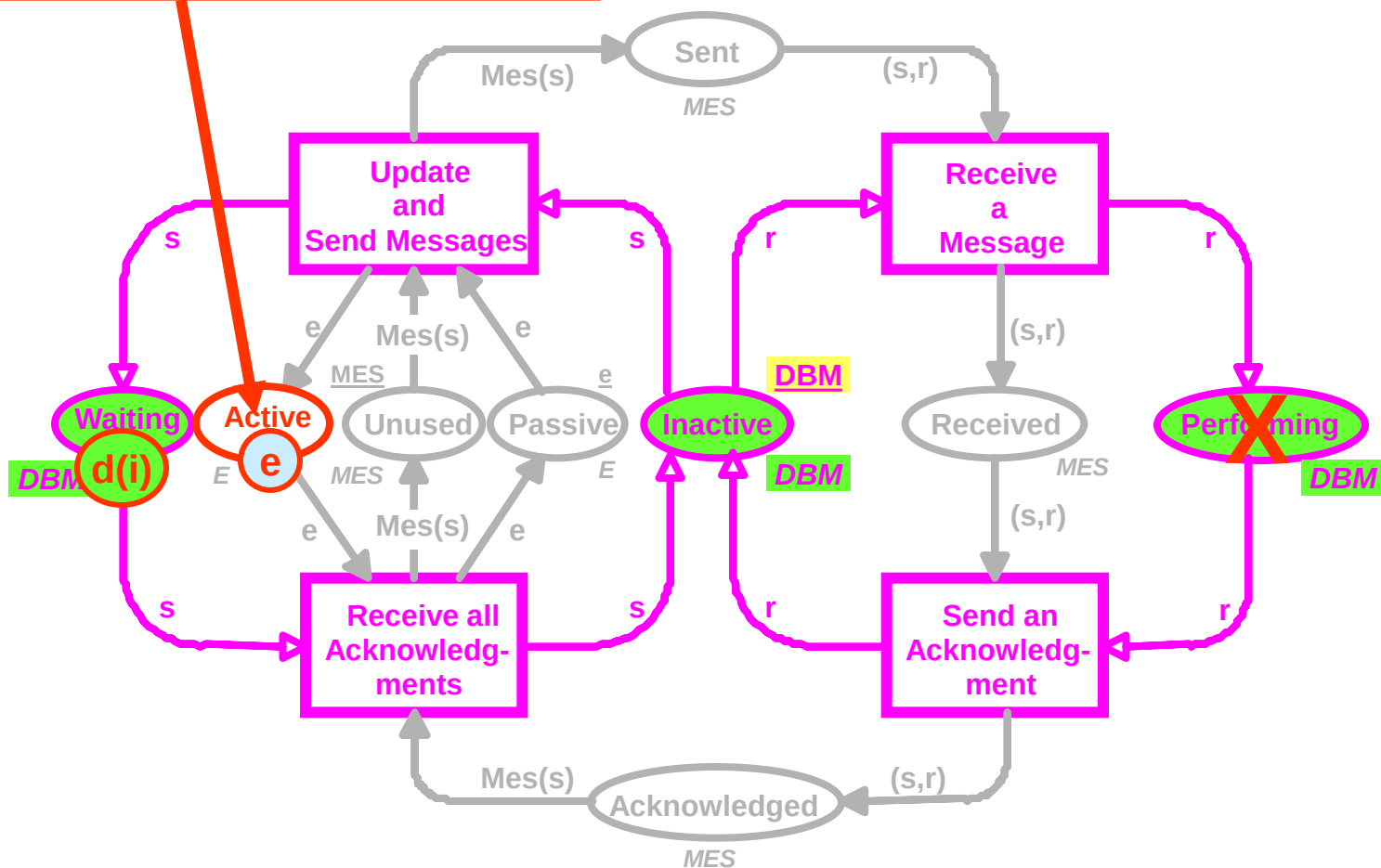
Ign(Waiting) + Passive = E

Pontosán egy token várakozik



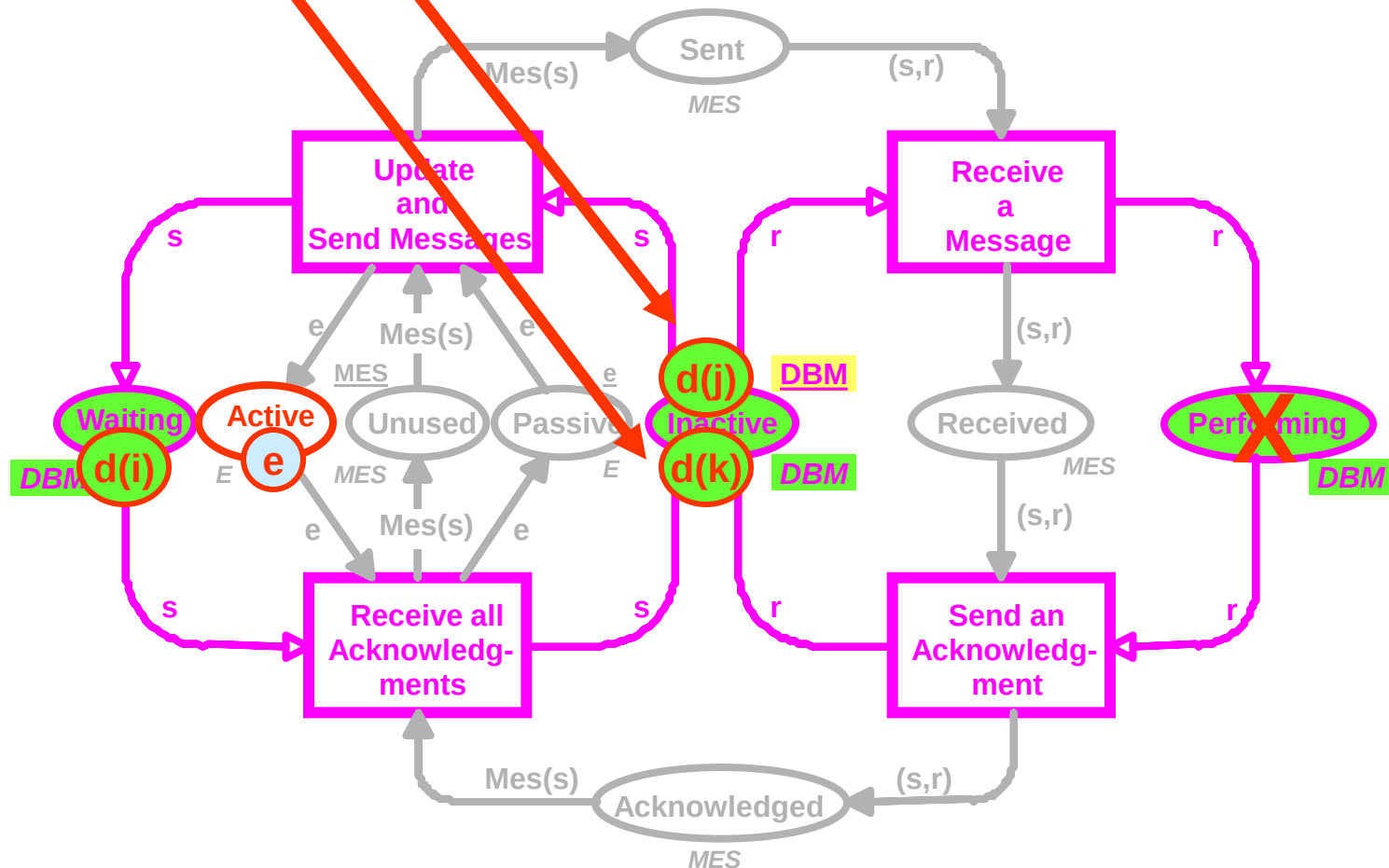
Ign(Waiting) = Active

Pontosan egy token aktív



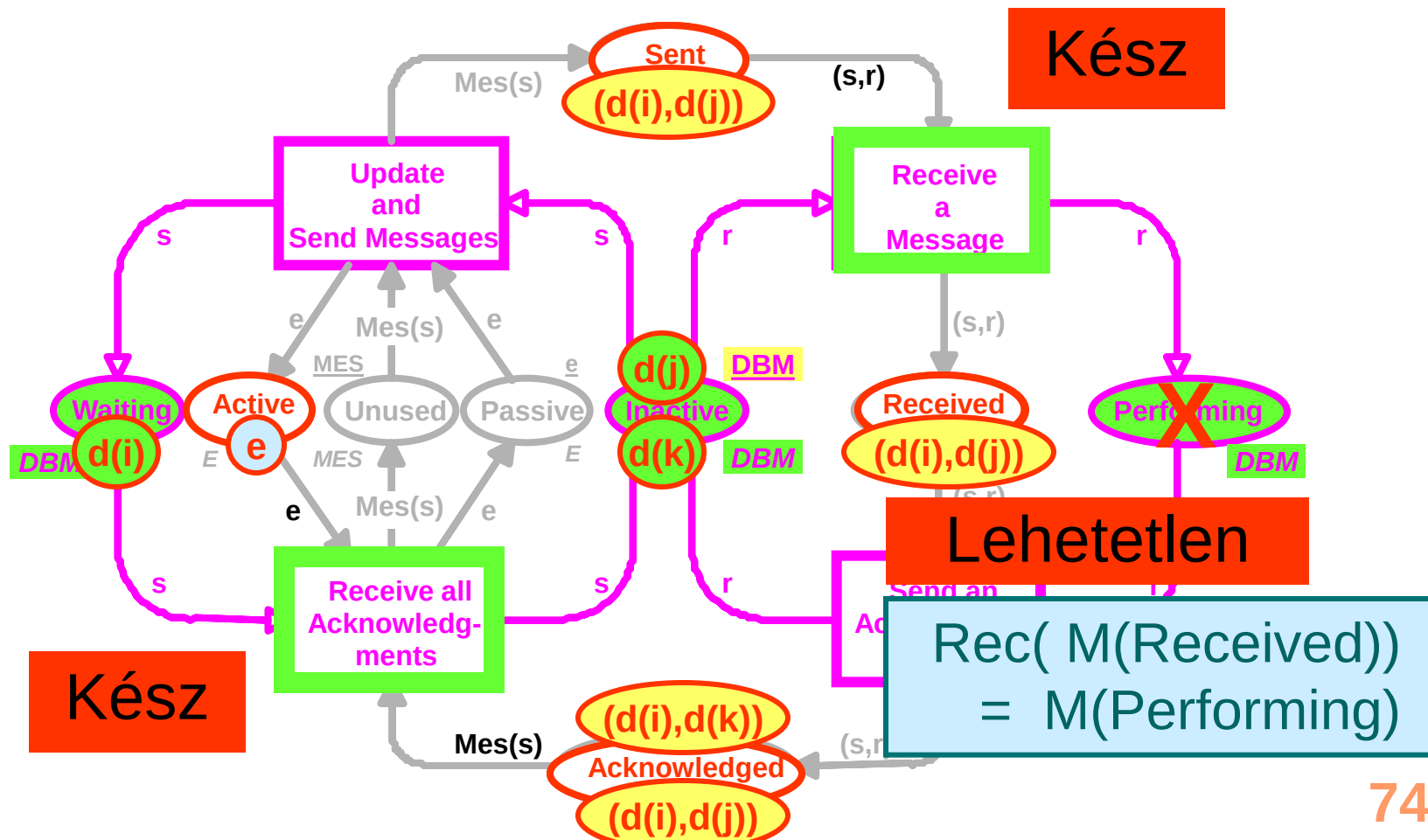
$$M(\text{Waiting}) + M(\text{Inactive}) + M(\text{Performing}) = \text{DBM}$$

Nincs több aktív adatbázis kezelő



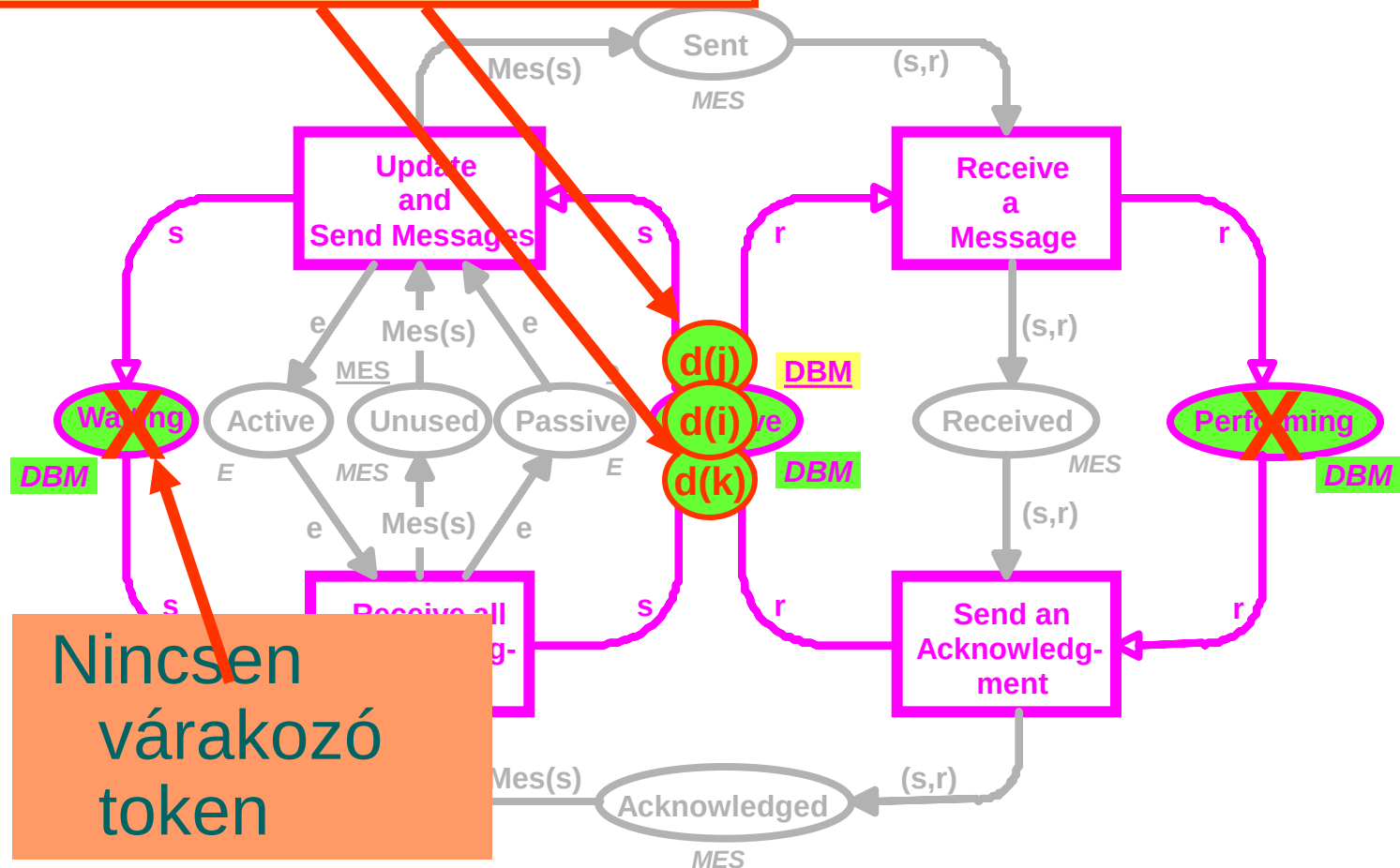
$$\text{Mes}(\text{Waiting}) = \text{M}(\text{Sent}) + \text{M}(\text{Received}) + \text{M}(\text{Acknowledged})$$

A d(i) által küldött üzeneteknek ennek kell lennie:



$$M(\text{Waiting}) + M(\text{Inactive}) + M(\text{Performing}) = \text{DBM}$$

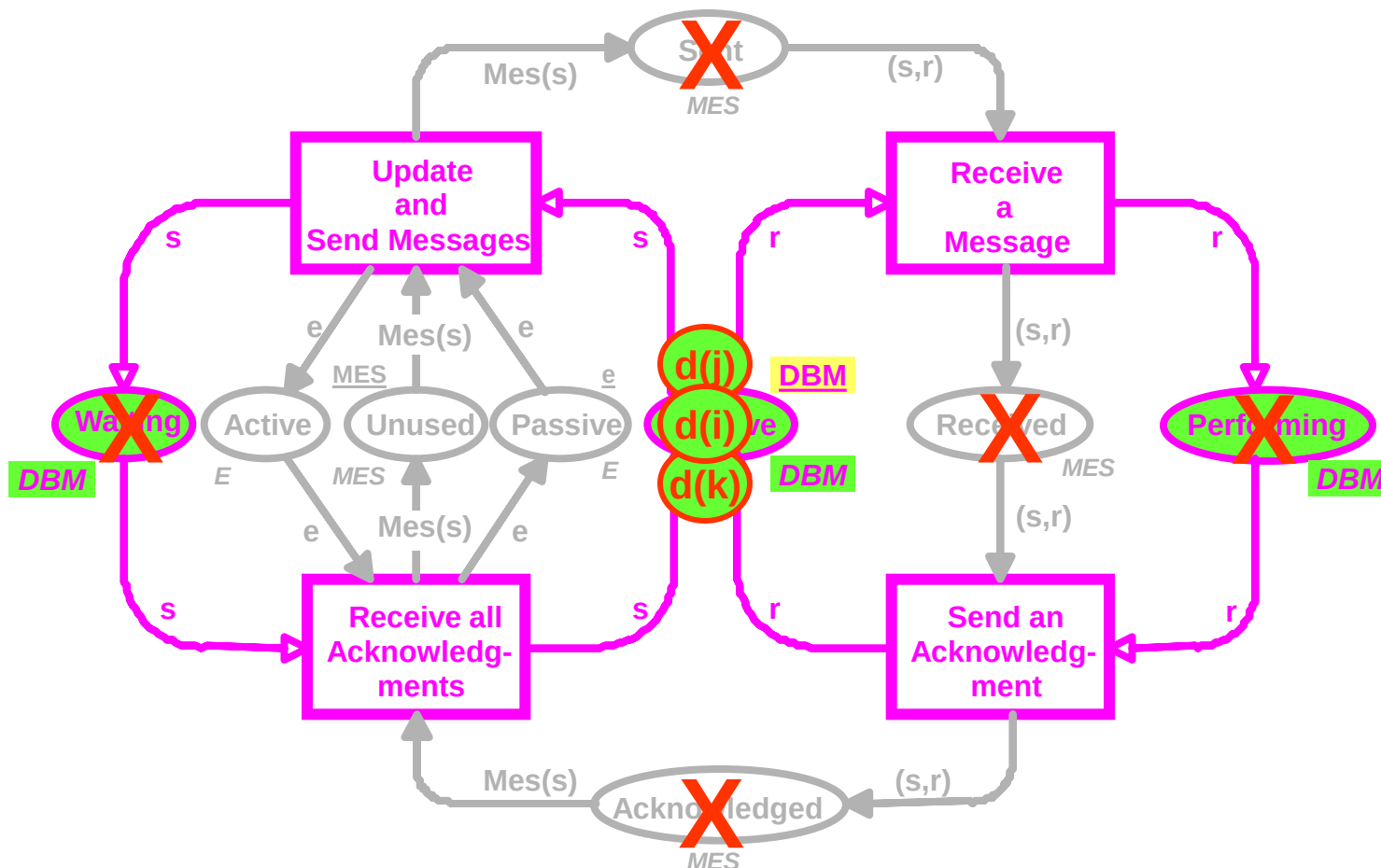
Nincsen aktív adatbázis kezelő



Nincsen várakozó token

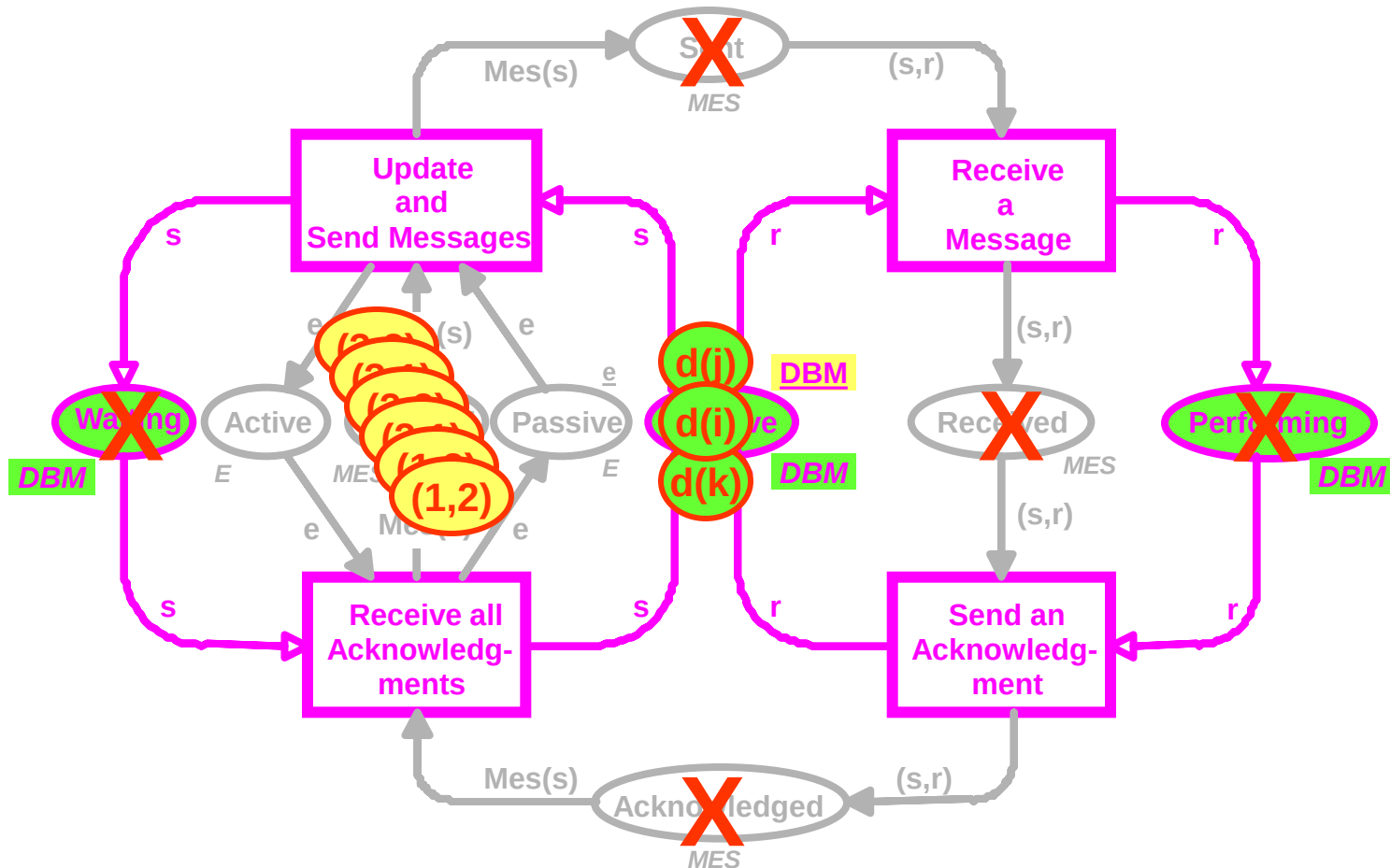
$$\text{Mes}(\text{Waiting}) = \text{M}(\text{Sent}) + \text{M}(\text{Received}) + \text{M}(\text{Acknowledged})$$

Nincsen elküldött, várakozó és nyugta token



$$M(\text{Unused}) + M(\text{Sent}) + M(\text{Received}) + M(\text{Acknowledged}) = \text{MES}$$

Az üzenetbufferek üresek:

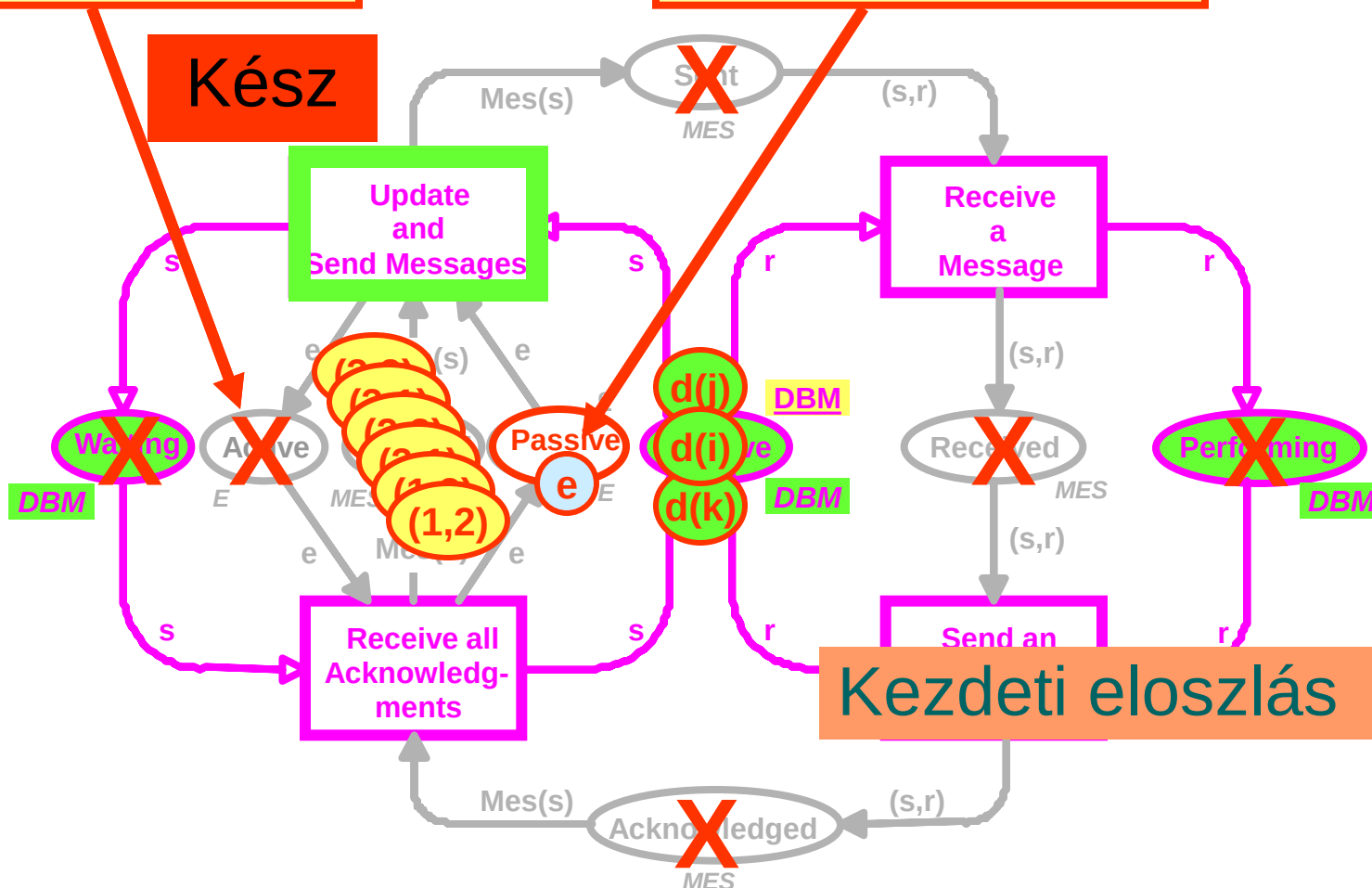


$\text{Ign}(\text{Waiting}) = \text{Active}$

$\text{Active} + \text{Passive} = E$

Nincsen
aktív token

Egy e-token
Passive



Konklúzió

THEORY

- models
- basic concepts
- analysis methods

TOOLS

- editing
- simulation
- verification

PRACTICAL USE

- specification
- validation
- verification
- implementation

- ◆ A CPN egyik **sikere** az, hogy mindhárom területen **egyszerre** használható.