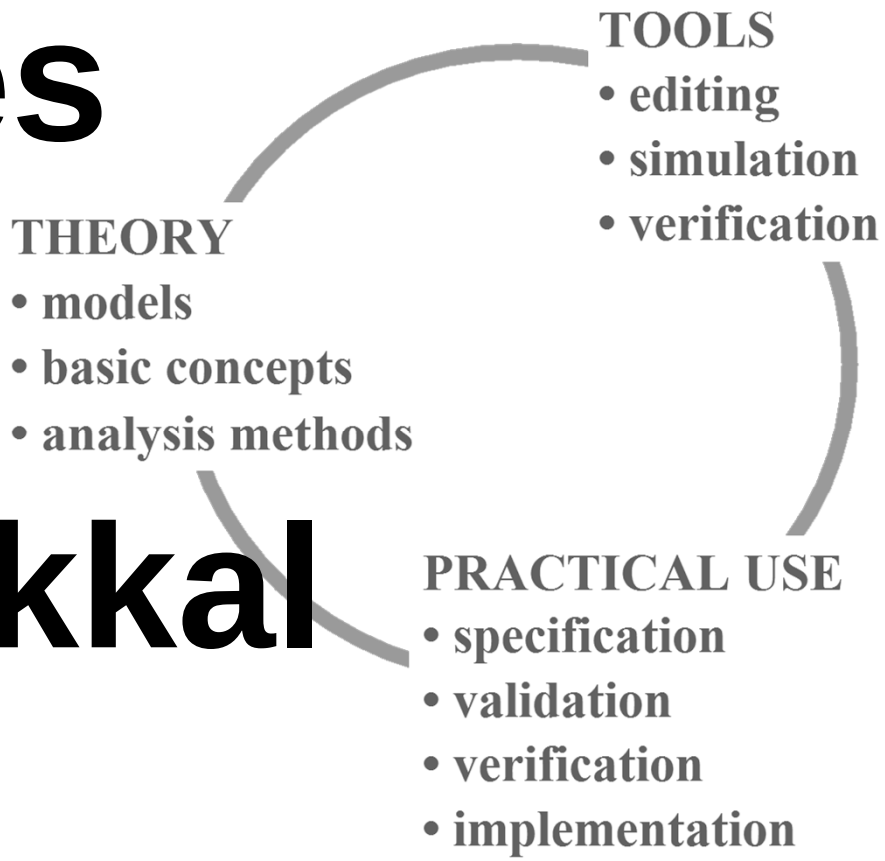
Modellezés Színezett Petri-hálókkal



© Kurt Jensen (részlet)

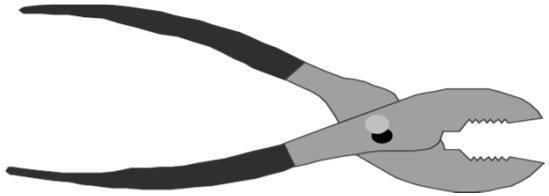
Fordította: Erdélyi Árpád

Department of Computer Science
Aarhus Egyetem, Dánia

kjensen@daimi.au.dk

Modellező eszközök

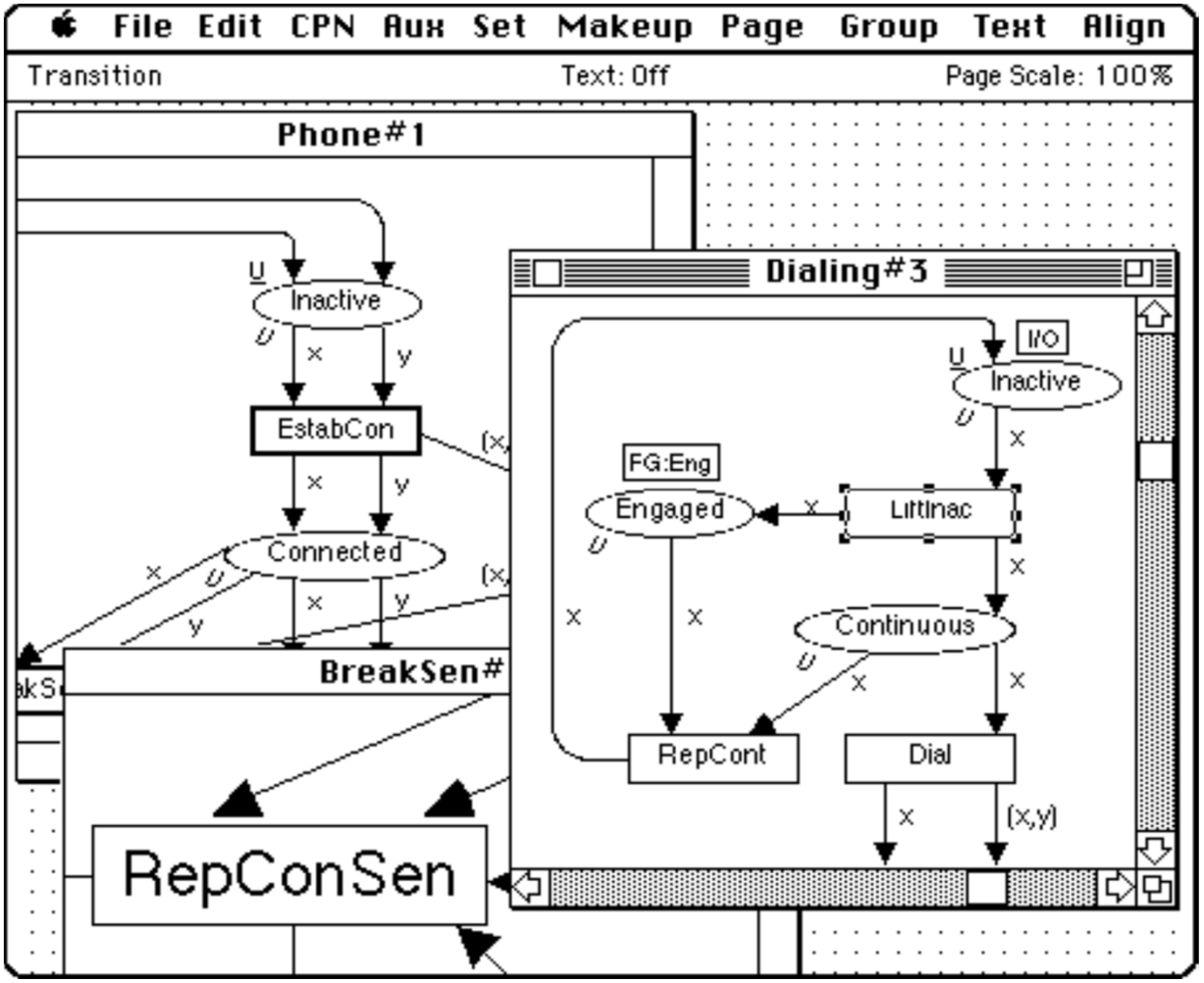
- *Design/CPN* eszközt a 80‘as évek végén és a 90‘as évek elején fejlesztették ki.
  - A maga idejében a legelterjedtebb Petri háló csomag volt.
  - *750 különböző szervezet 50 országban használta* – ezek között *200 kereskedelmi cég van.*
- *CPN Tools* már egy második generációs eszköz a Színezett Petri-hálókhoz.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



Standard ML

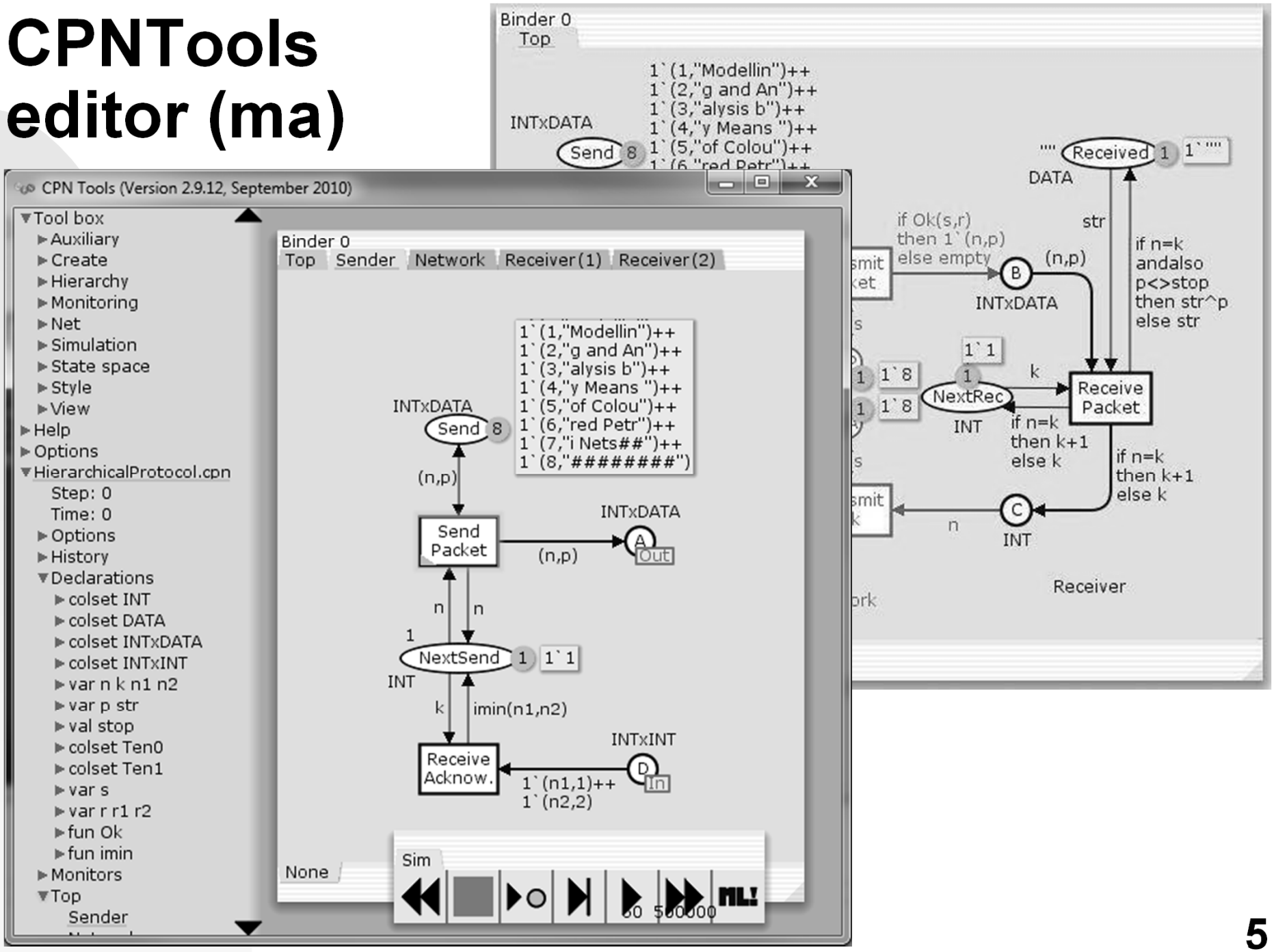
- A típusok, élkifejezések és őrfeltételek *Standard ML-ben* adhatók meg. Ez egy erősen típusos, funkcionális programozási nyelv (*Robin Milner* fejlesztette).
- *Adattípus lehet*:
  - *Atomi* (integer, string, boolean és felsorolás).
  - *Strukturált* (products, record, union, list és subsets).
- Tetszőlegesen 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)

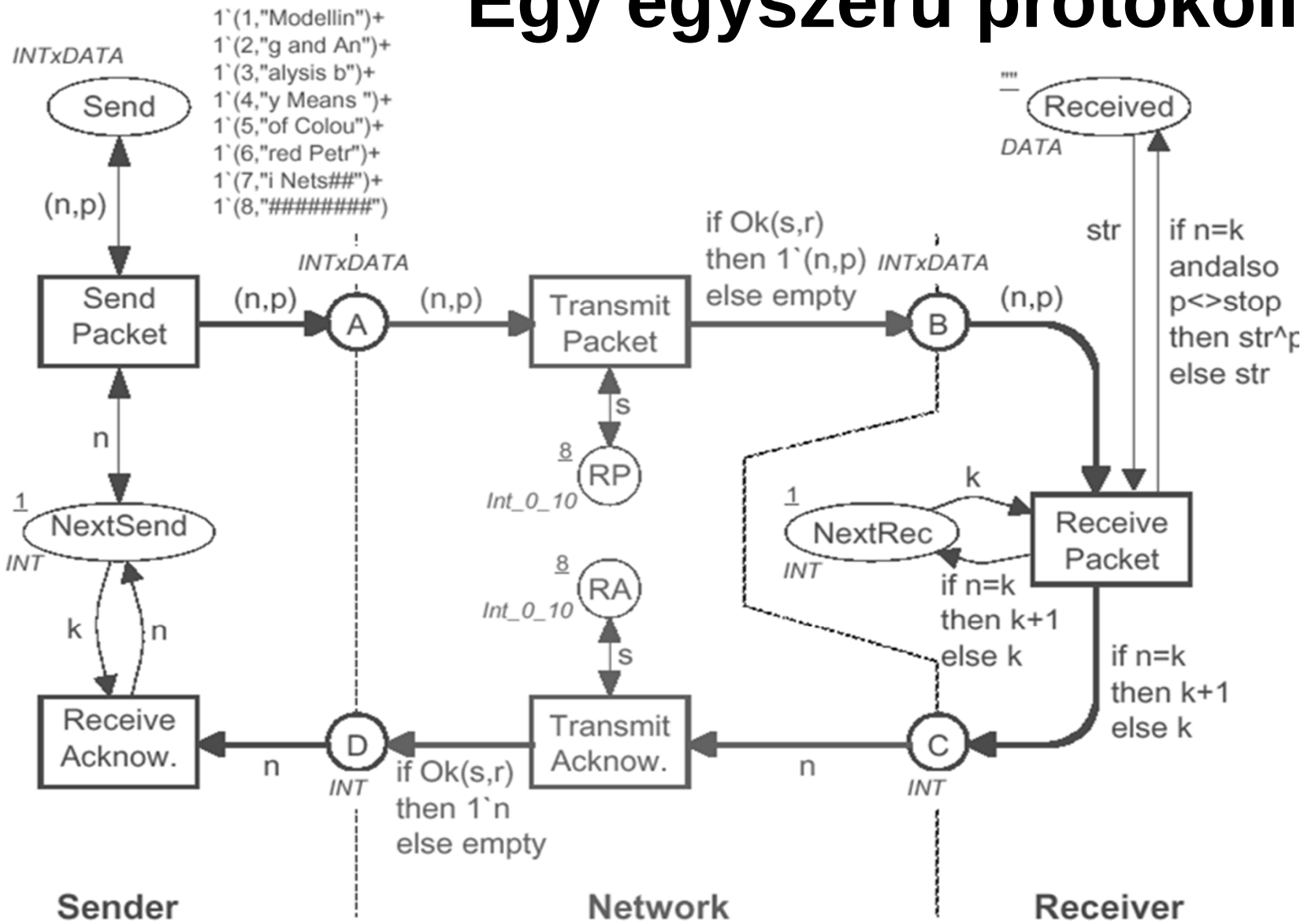


- *WYSIWYG felületről* könnyen módosíthatni tudjuk a modellünket.
- *Szintakszis-vezérelt* – sok *szintaktikai hiba* kiküszöbölhető.

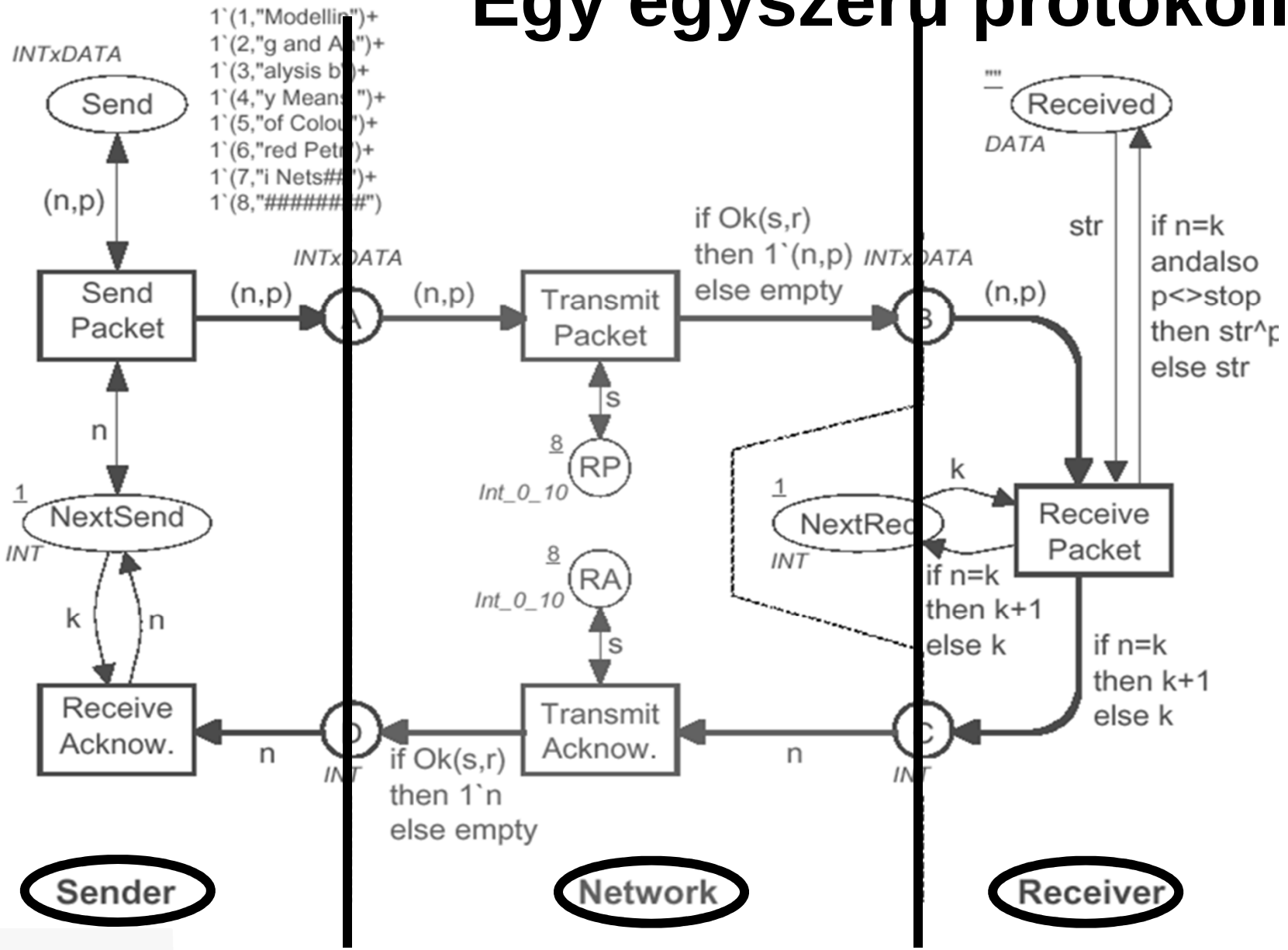
CPNTools editor (ma)



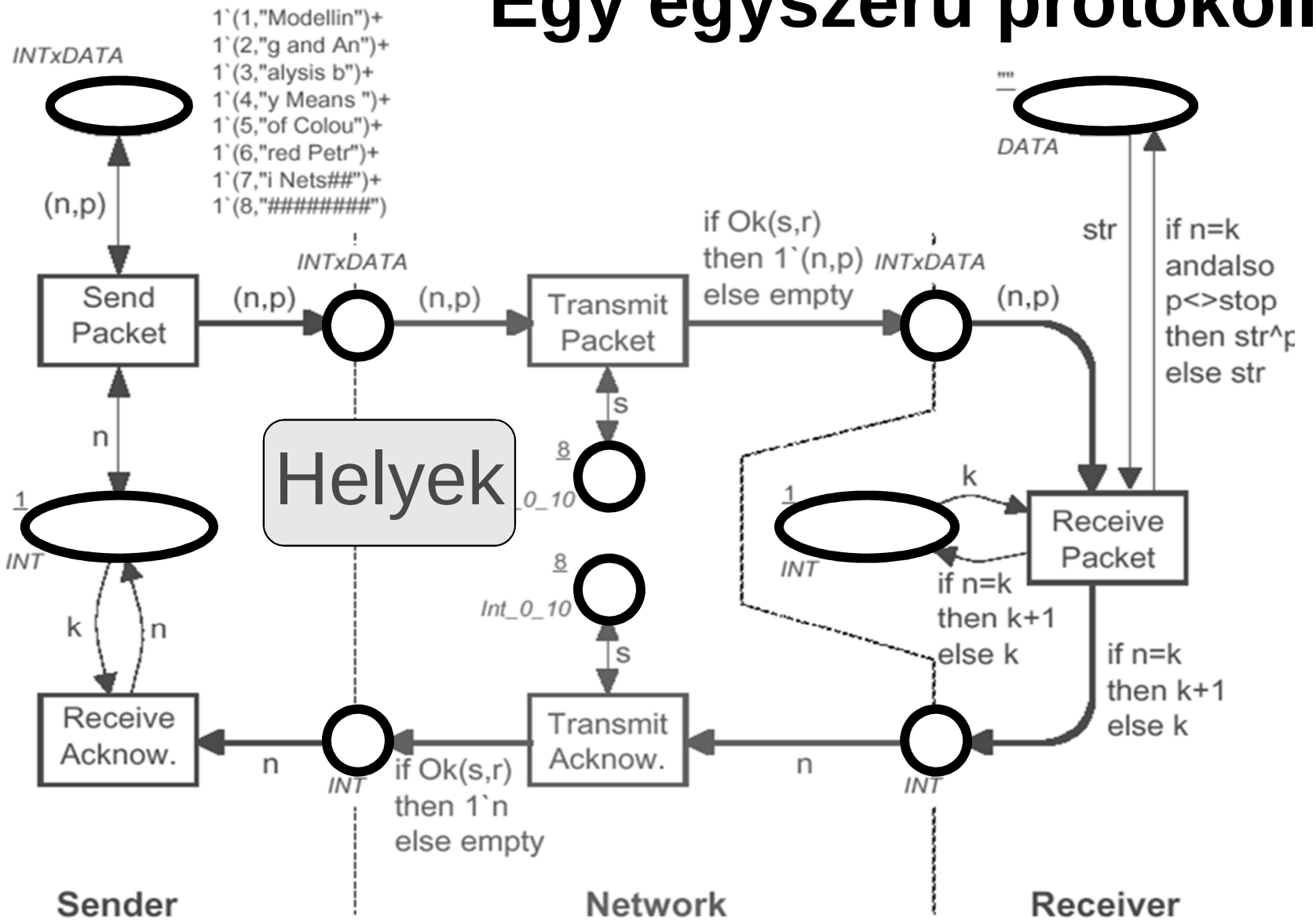
Egy egyszerű protokoll



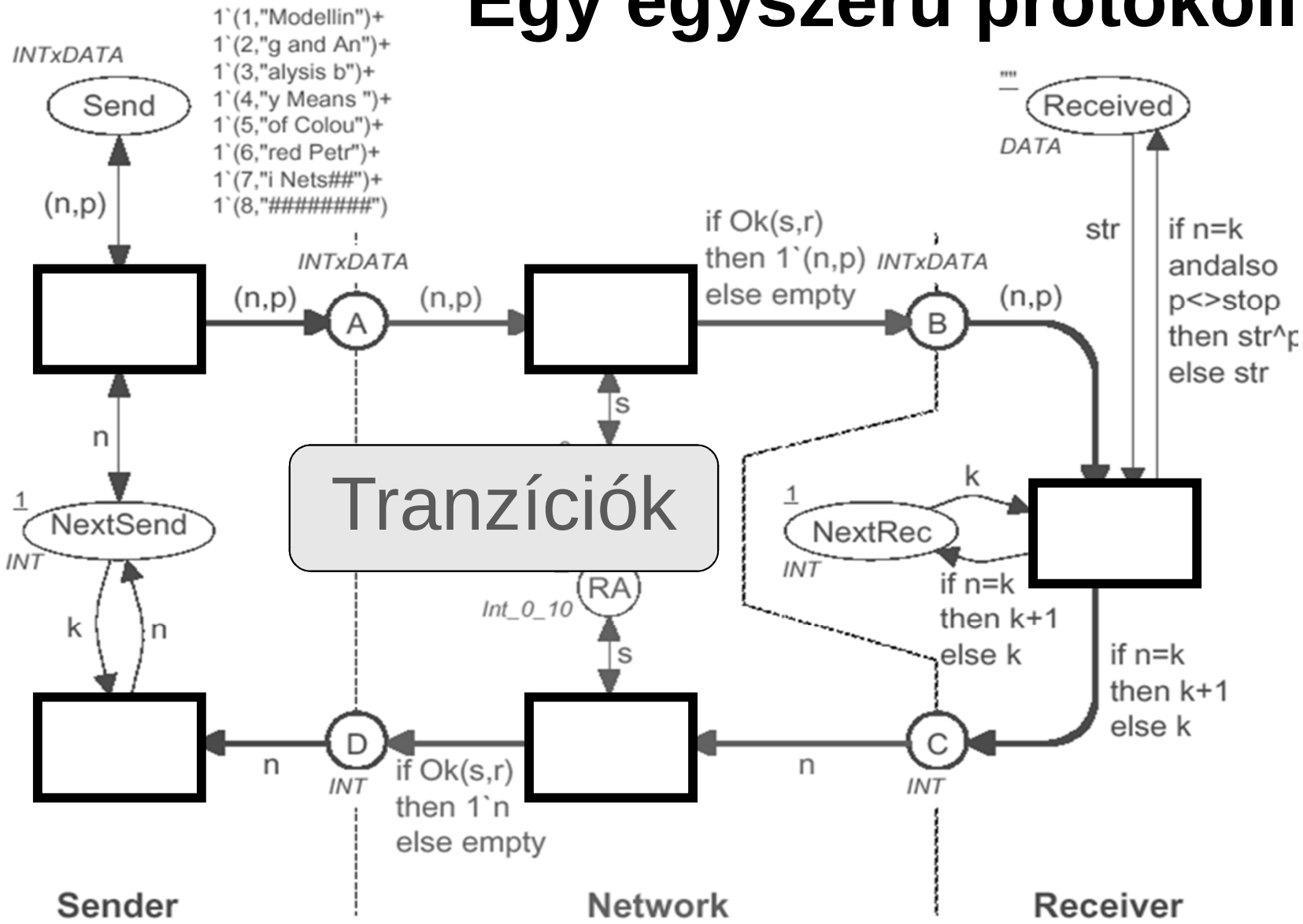
Egy egyszerű protokoll



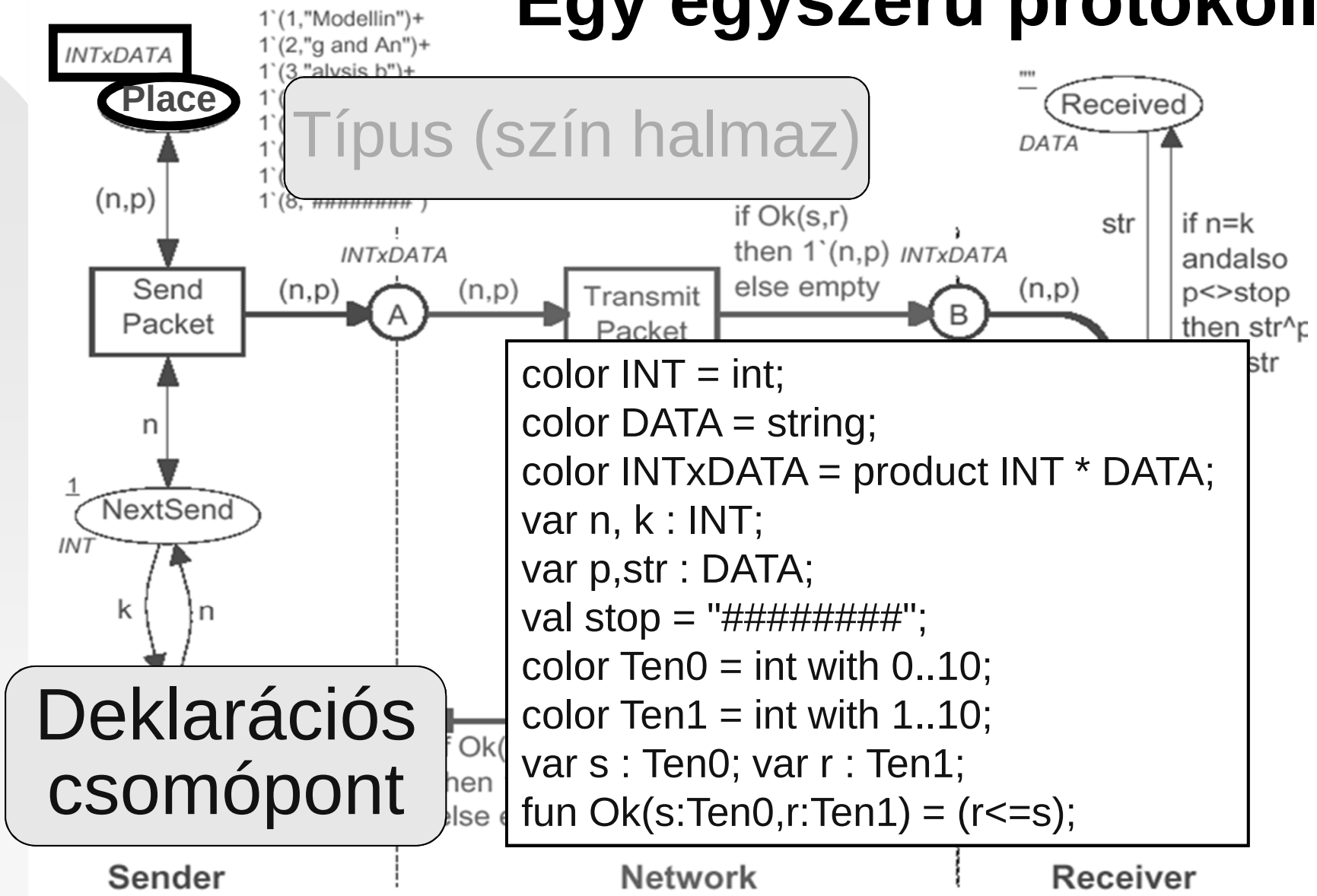
Egy egyszerű protokoll



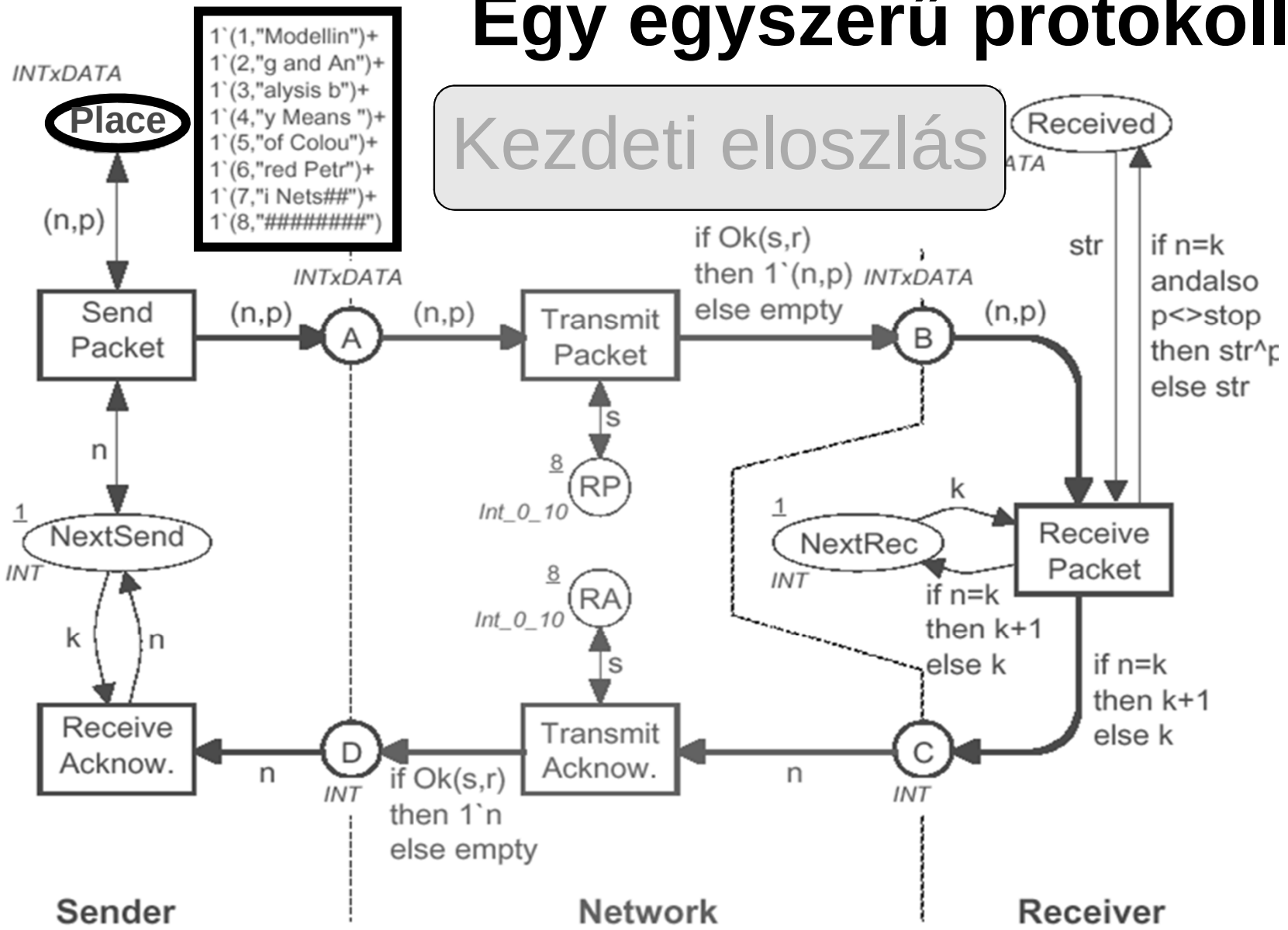
Egy egyszerű protokoll



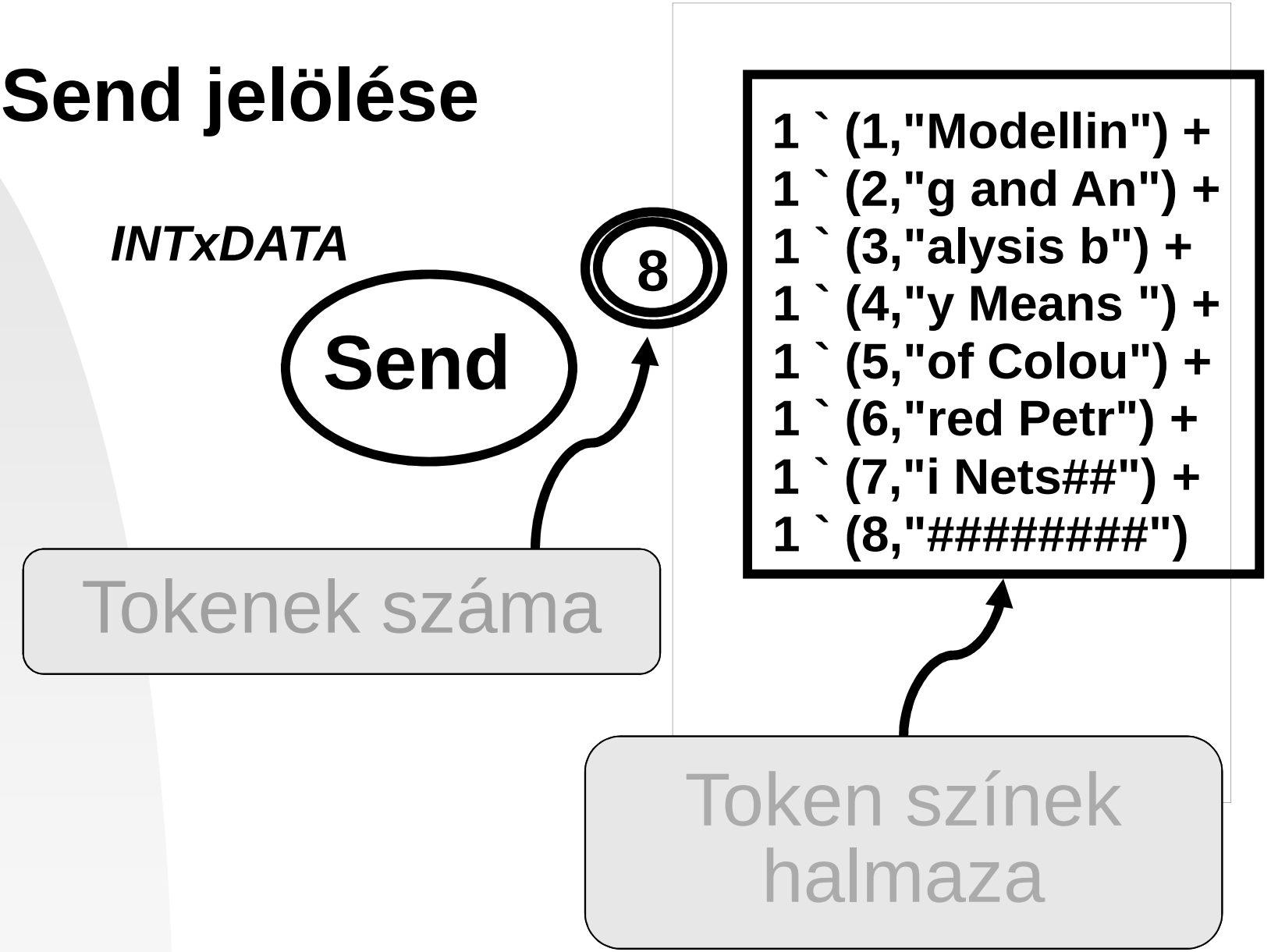
Egy egyszerű protokoll



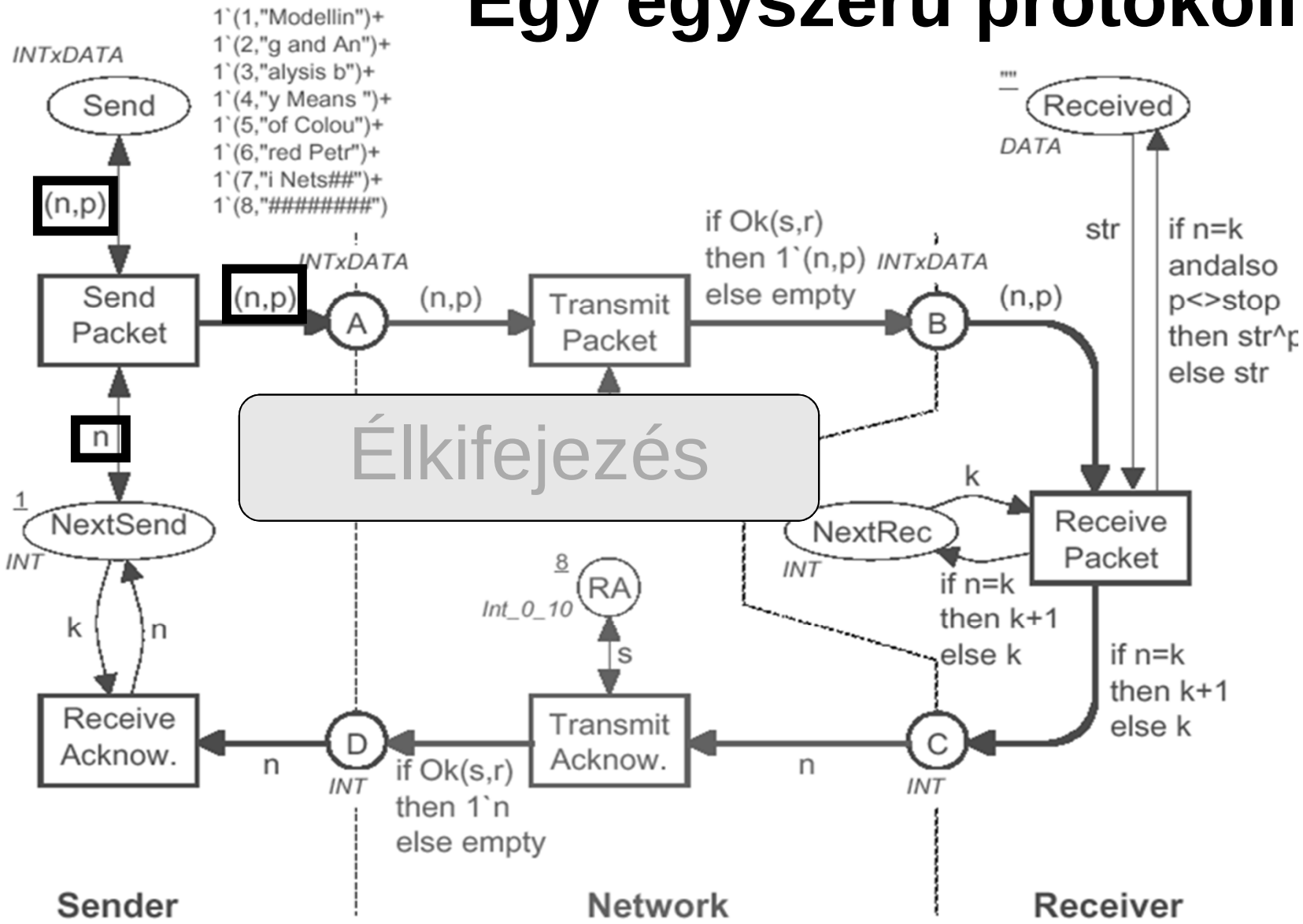
Egy egyszerű protokoll



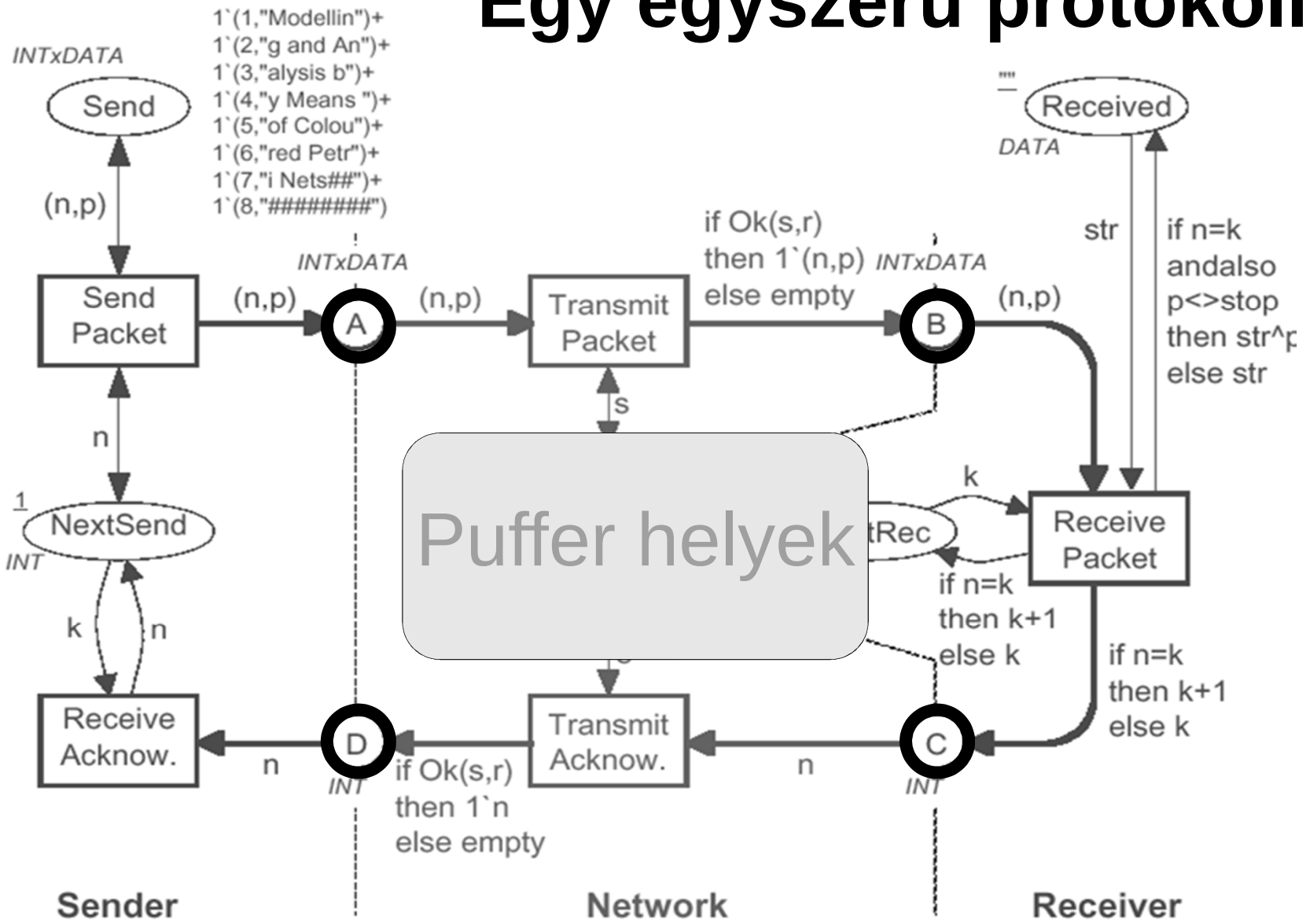
Send jelölése



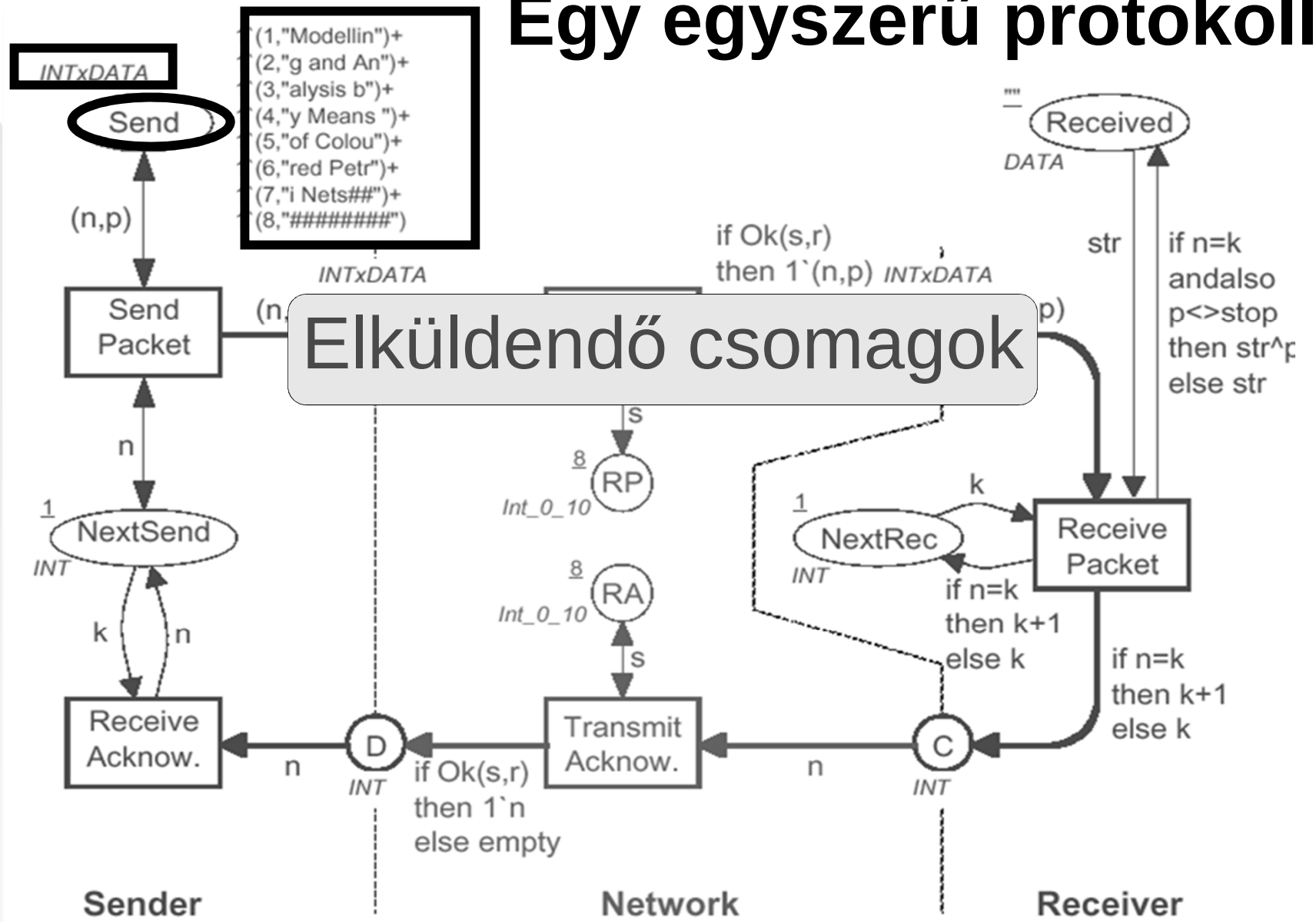
Egy egyszerű protokoll



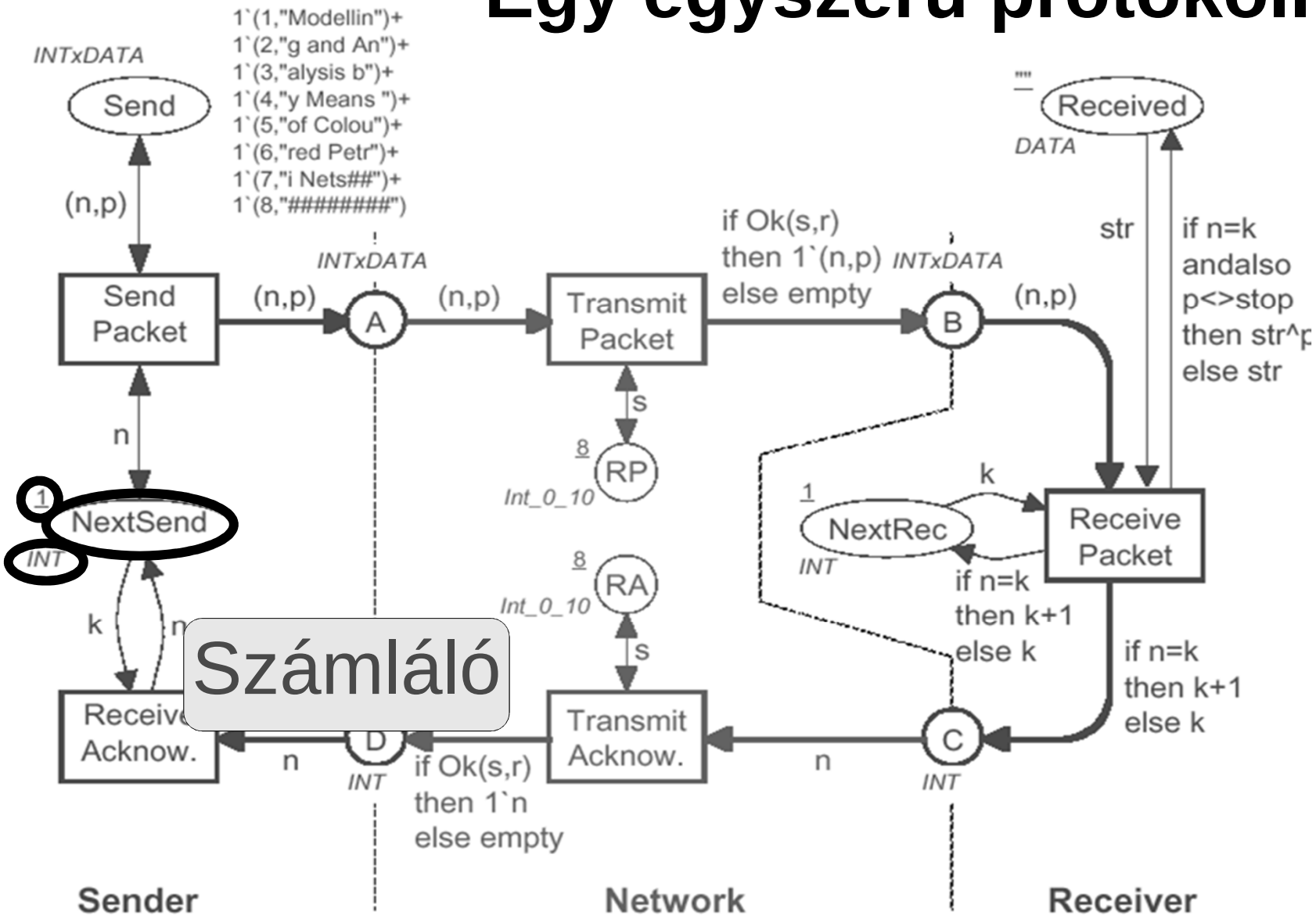
Egy egyszerű protokoll



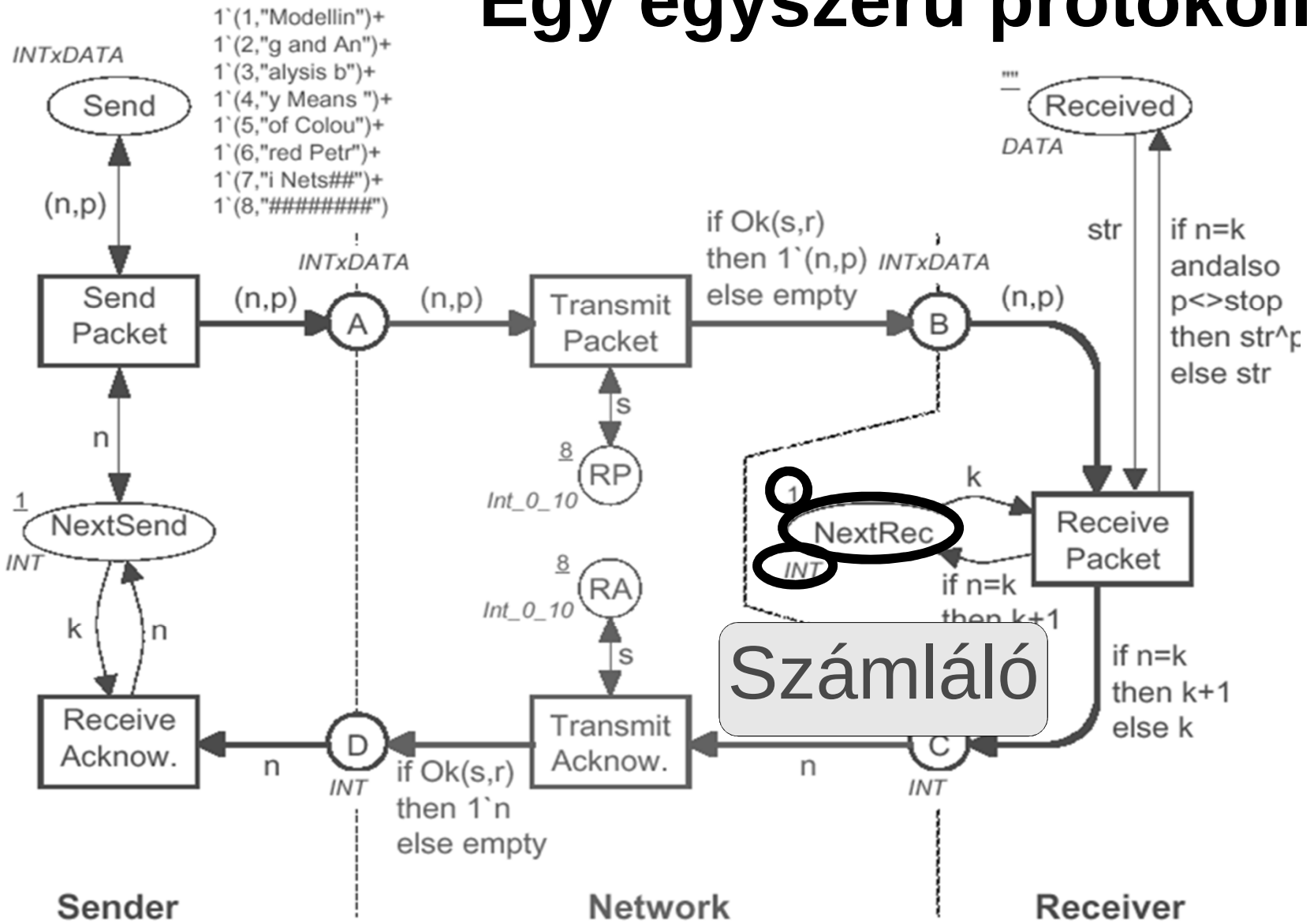
Egy egyszerű protokoll



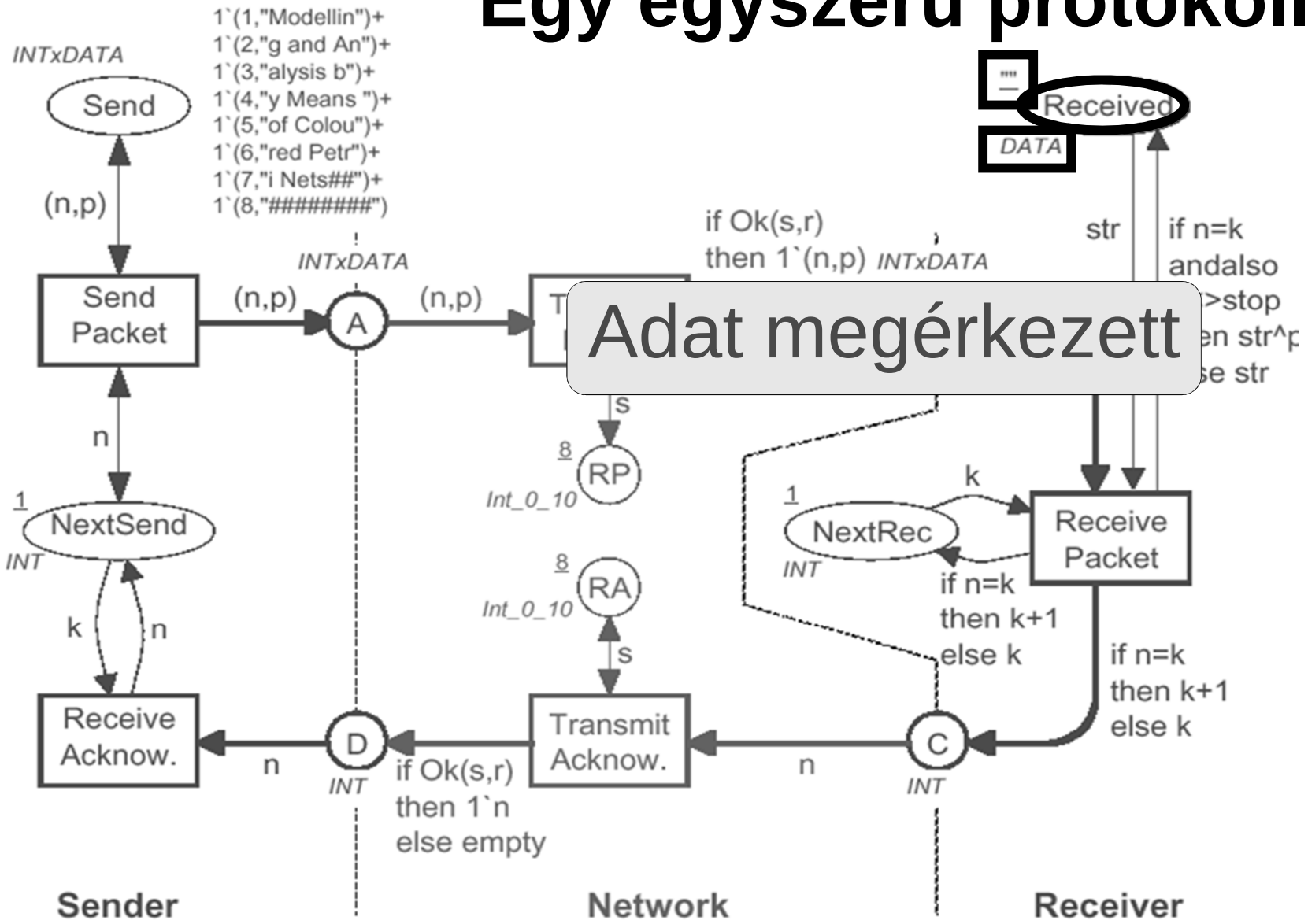
Egy egyszerű protokoll



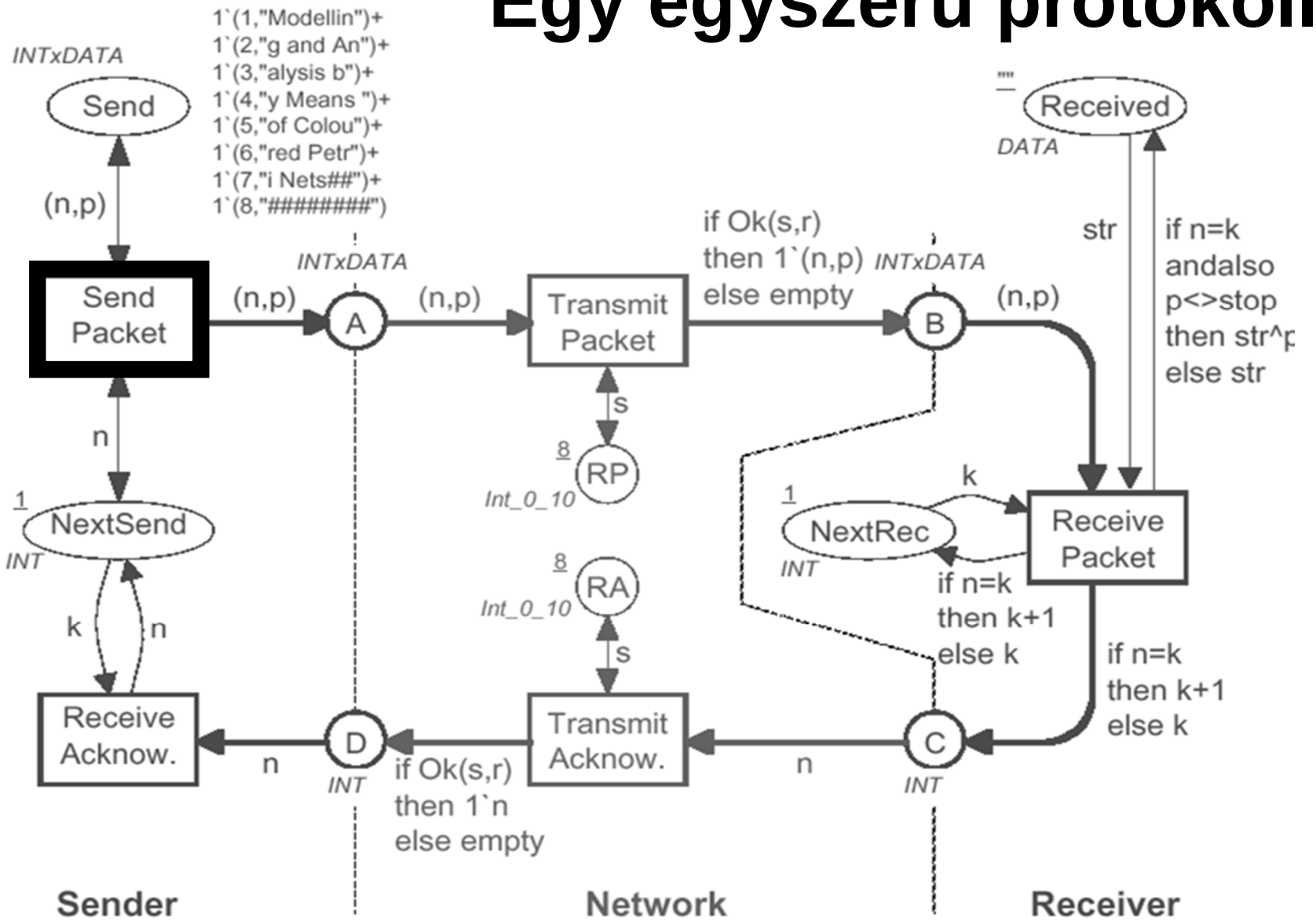
Egy egyszerű protokoll



Egy egyszerű protokoll



Egy egyszerű protokoll

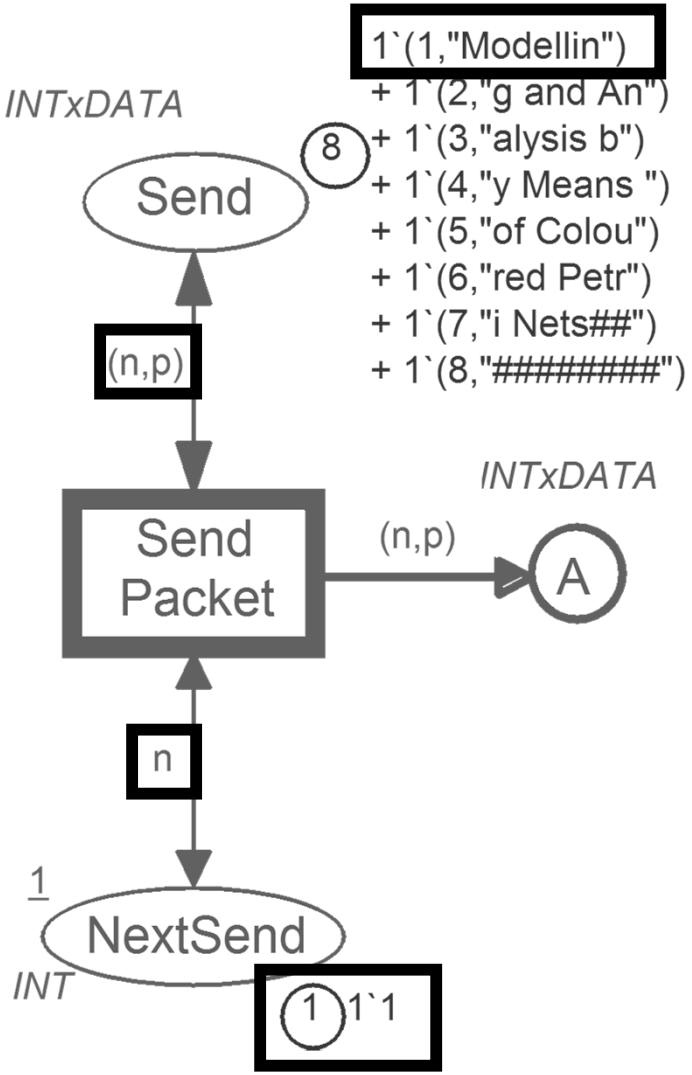


Csomagküldés

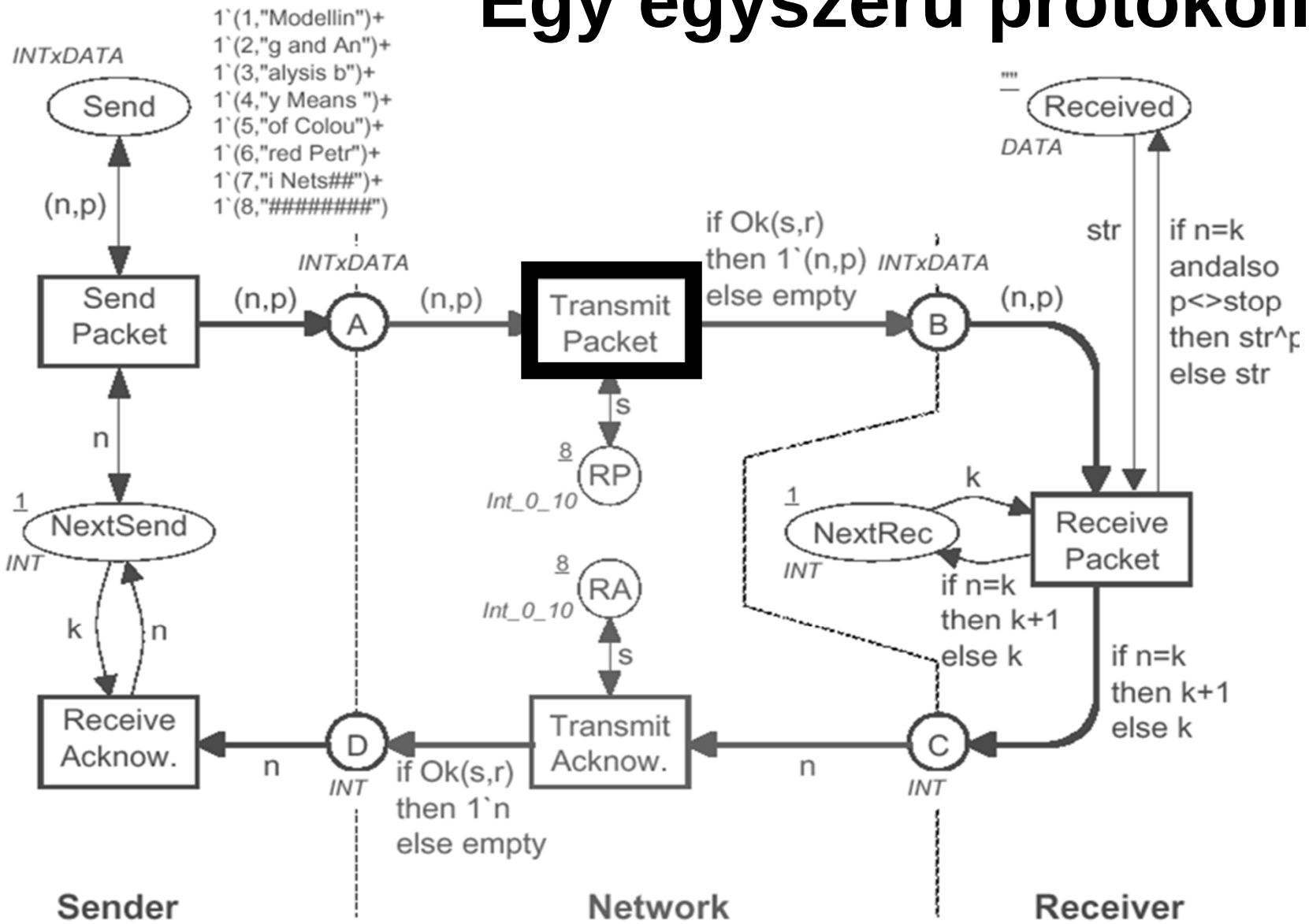
- Csak a

  <n=1,p="Modellin">

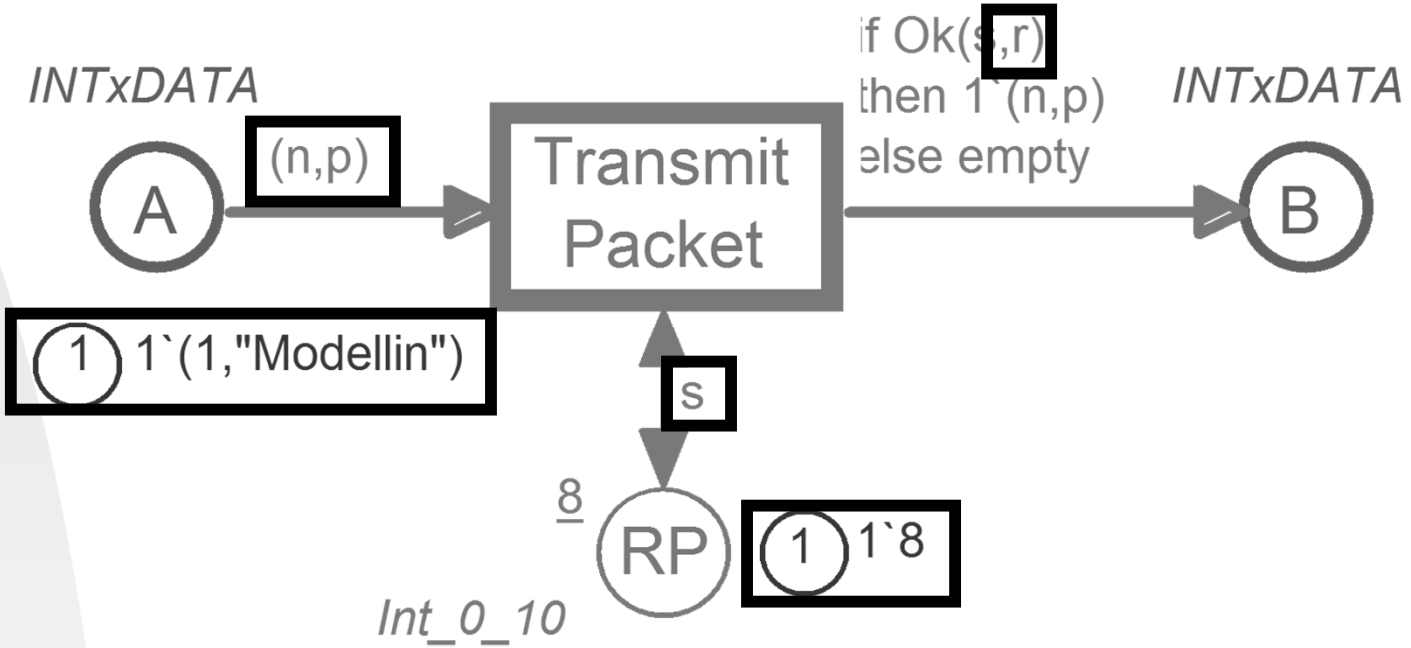
  pár *engedélyezett.*
- Ha található ilyen pár, akkor a *Send Packet* elküldi azt *A-nak*.
- Ez azt jelenti, hogy (1,"Modellin")-t elküldtük a hálózaton.
- A csomag nem törlődik a *Send* helyről és a *NextSend* sem változik.



Egy egyszerű protokoll



Csomag átvitele



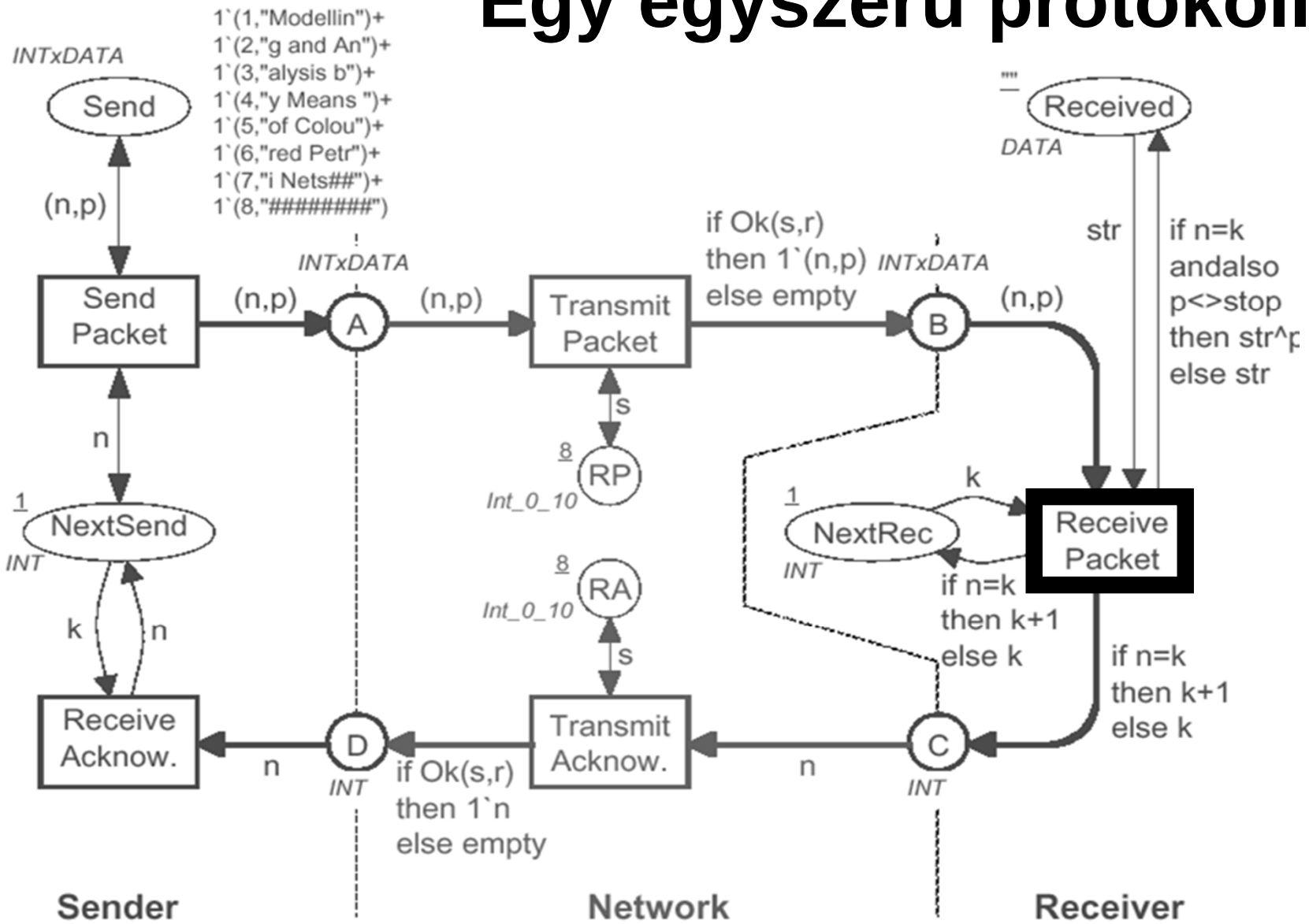
- Az engedélyezett pár így néz ki:
  - <n=1,p= "Modellin",s=8,r=...>
  - Típus: r ∈ 1. .10

Csomagvesztés

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

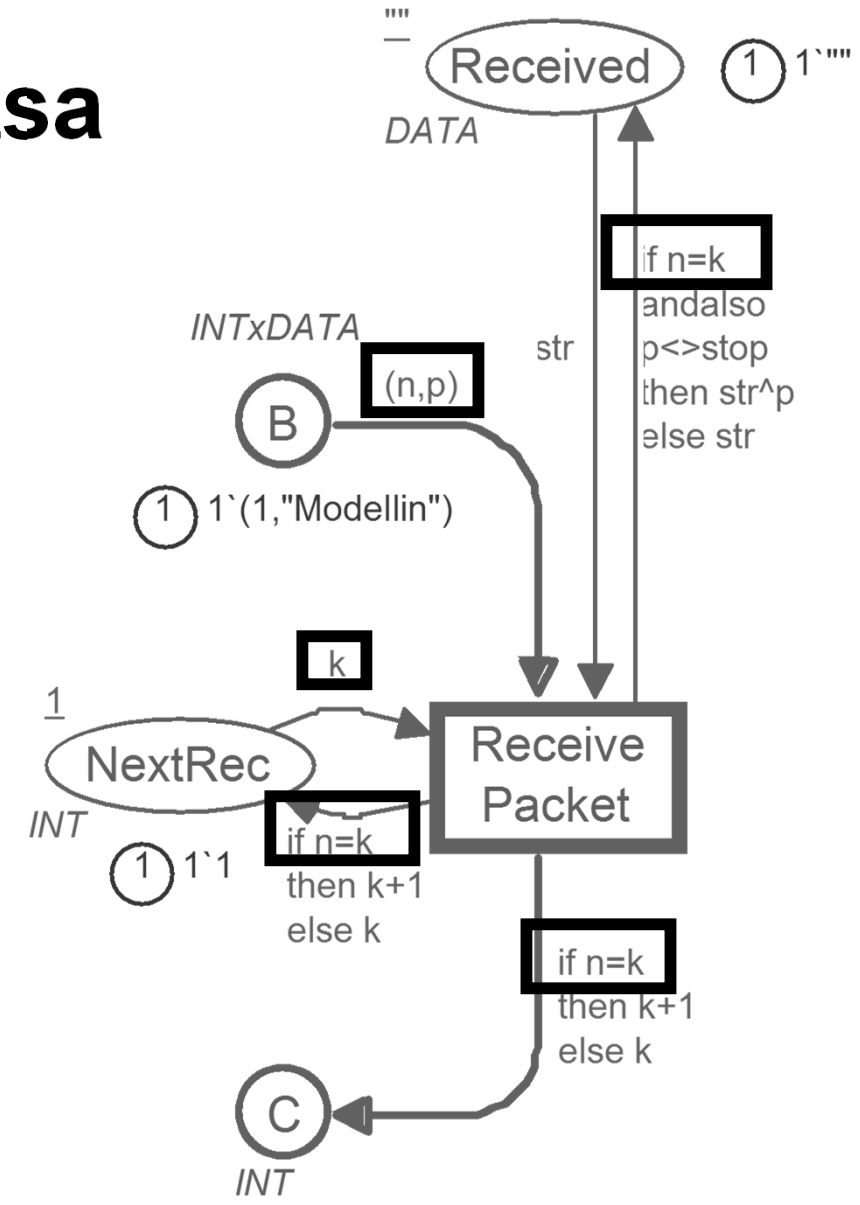
- A Ok(s,r) *függvény* ellenőrzi, hogy r ≤ s
  - For r ∈ 1. .8,*Ok(s,r)=true*.
    A tokent A-ból B-be tesszük. Ez azt jelenti, hogy a csomag *sikeresen átment* a hálózaton.
  - For r ∈ 9. .10,*Ok(s,r)=false.*
    Ekkor B-be nem került token. Ekkor *veszett el* a csomag.
- A CPN szimulátor *véletlenül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

Egy egyszerű protokoll



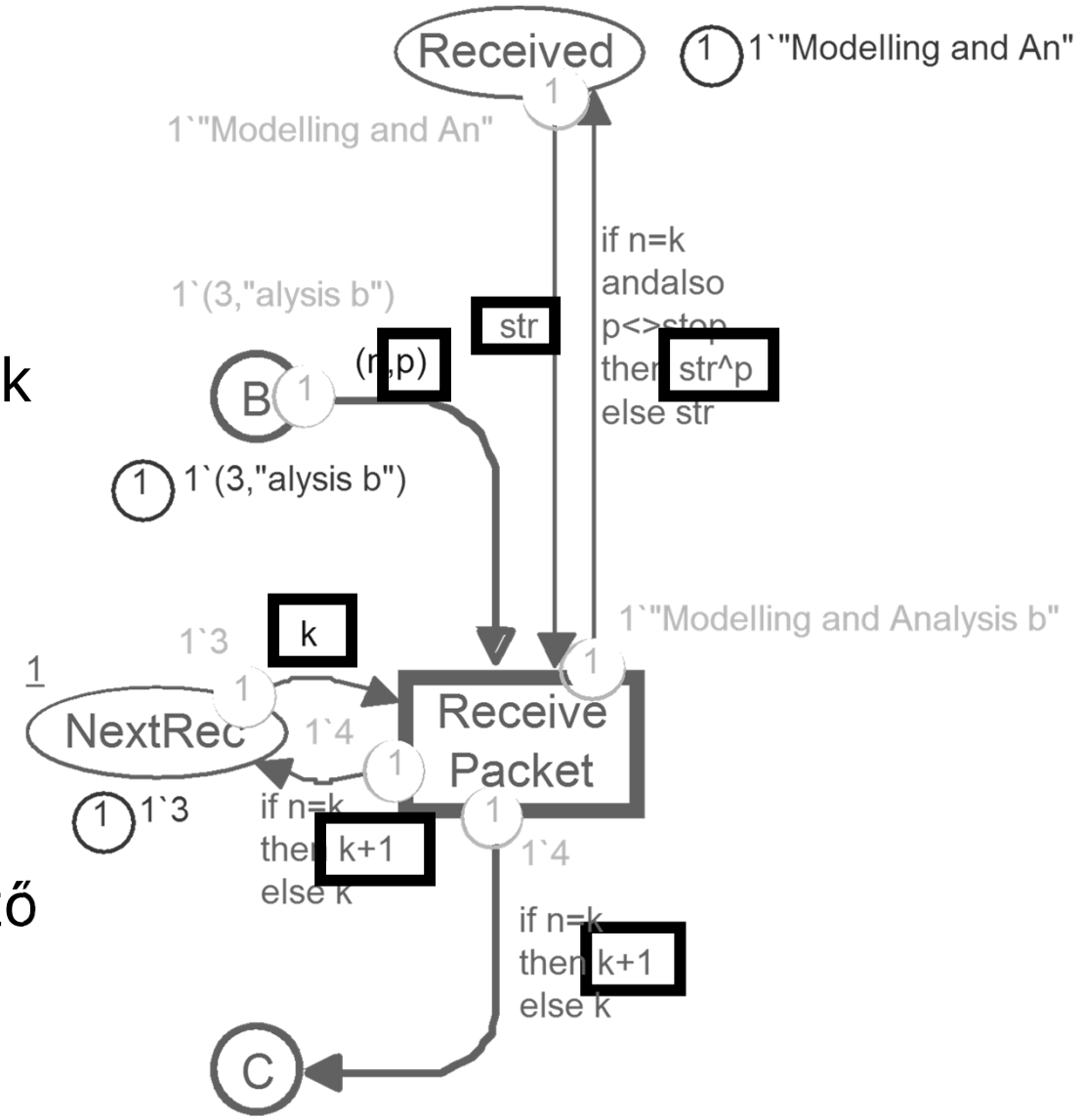
Csomag fogadása

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



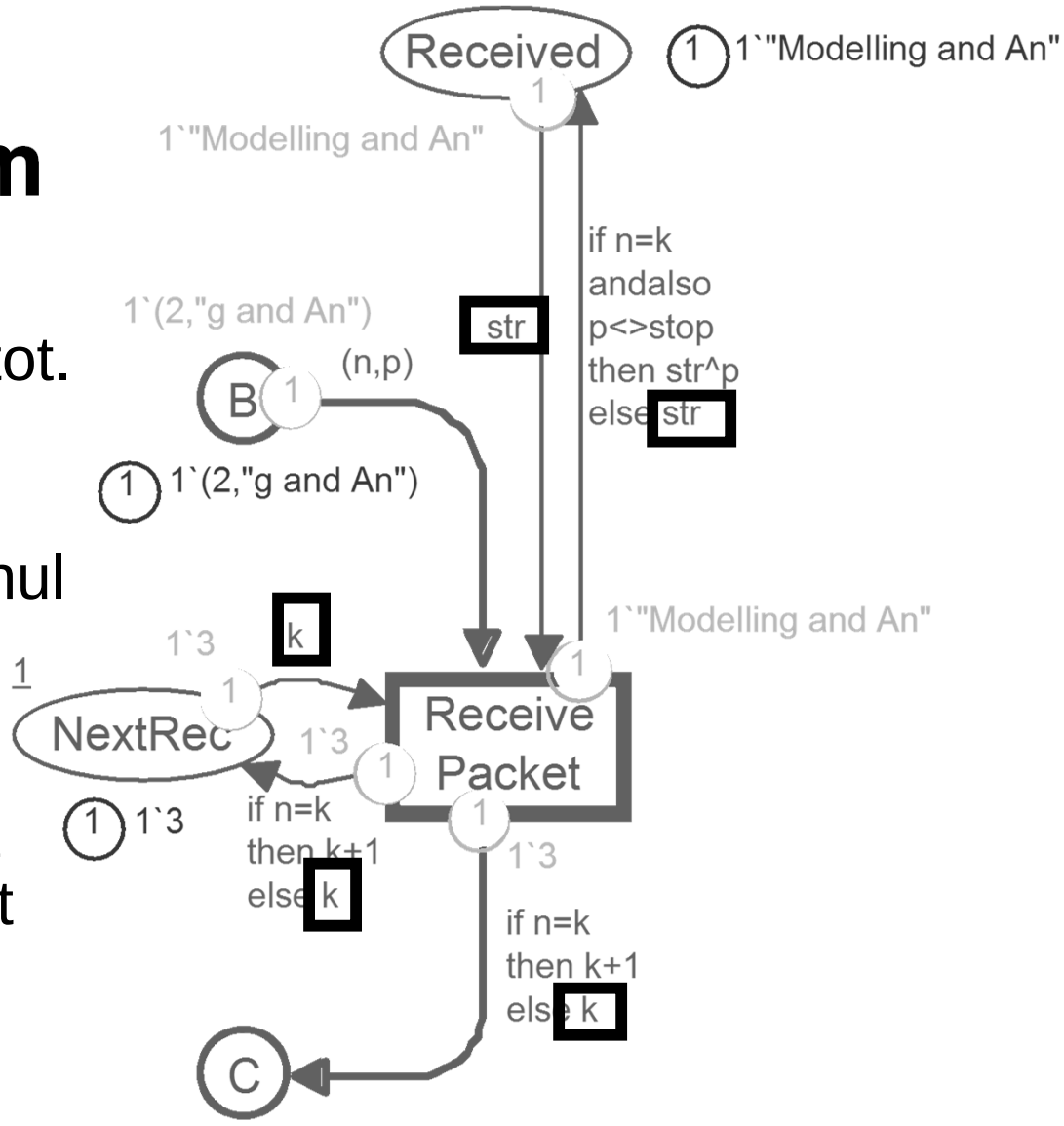
Csomagok sorszámának módosítása

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

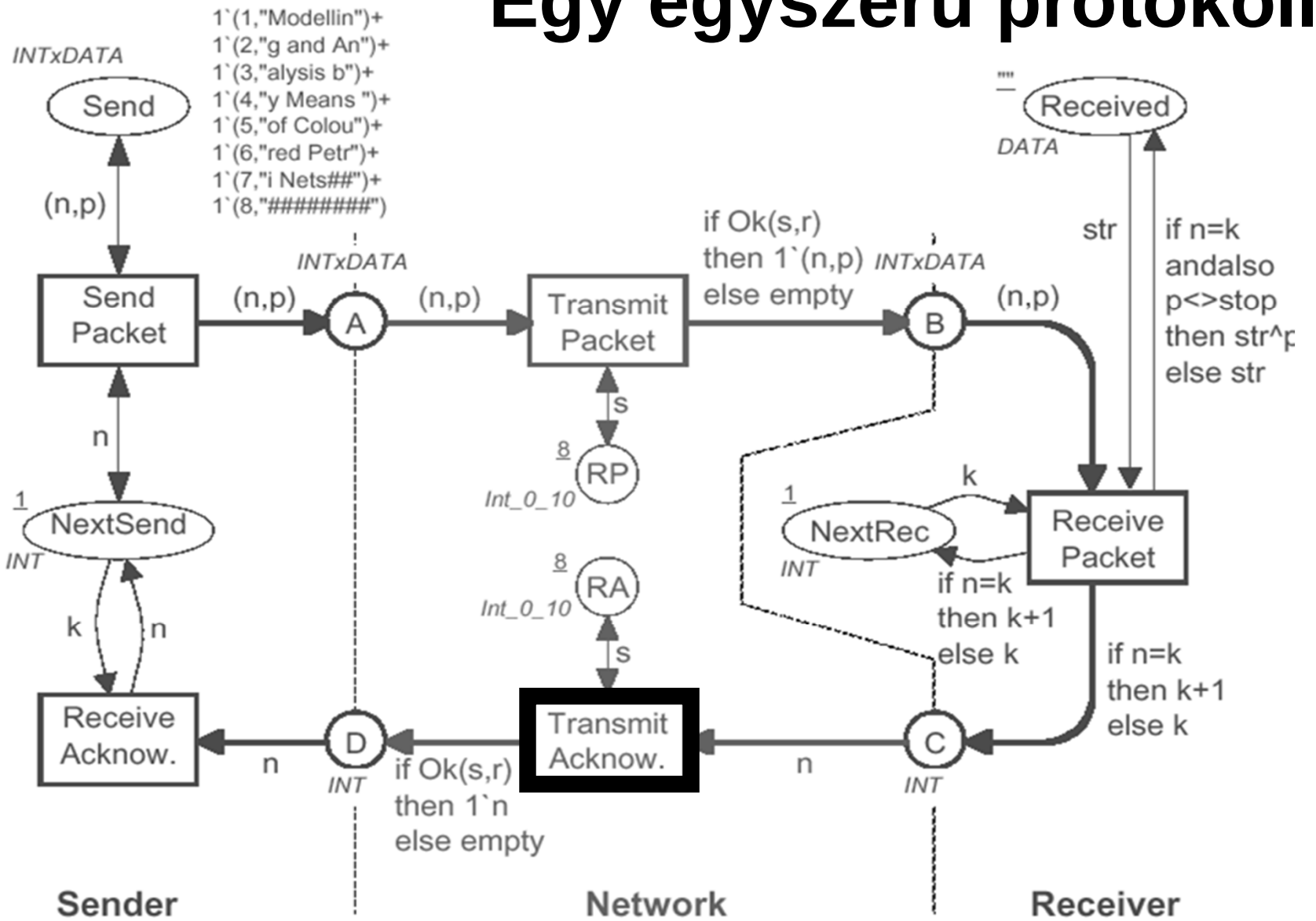


Rossz csomagsorszám

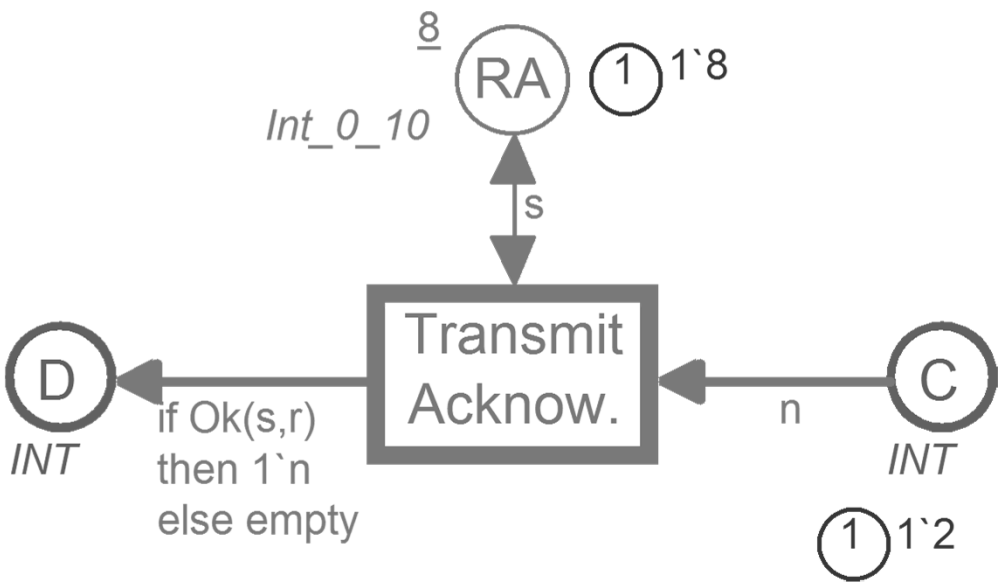
- *Eldobjuk* a kapott adatot.
- A *NextRec-t* változatlanul hagyjuk.
- Egy *acknowledgement üzenetet* küldünk. Ez a várt csomag sorszámát tartalmazza.



Egy egyszerű protokoll

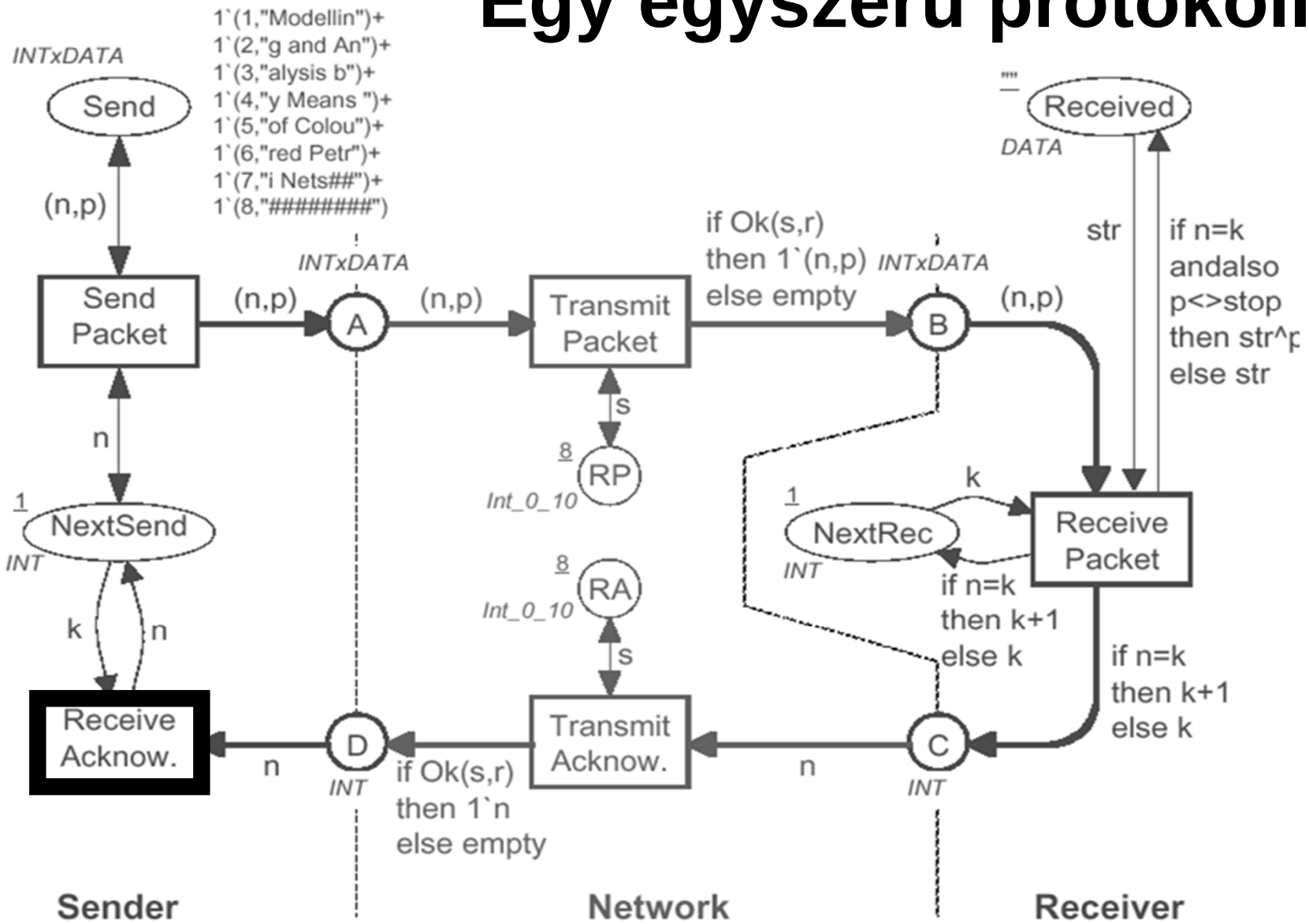


Átvitel visszajelzés

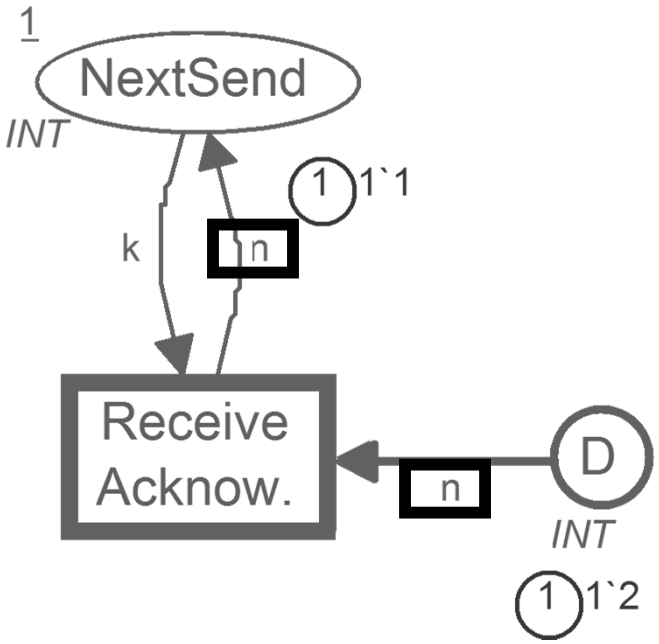


- Hasonlóan működik a *Csomag átvitelhez.*

Egy egyszerű protokoll

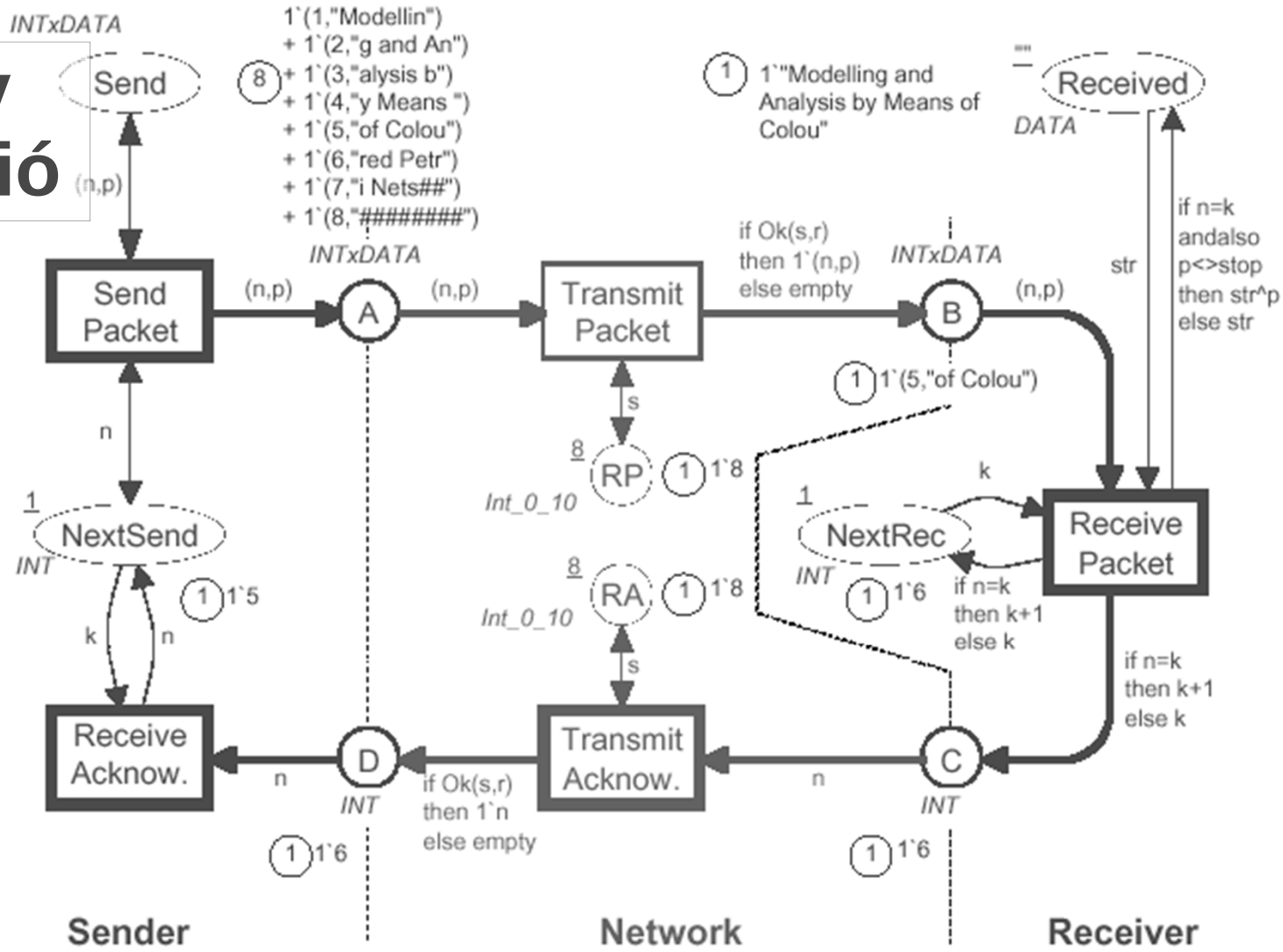


Fogadási visszajelzés



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

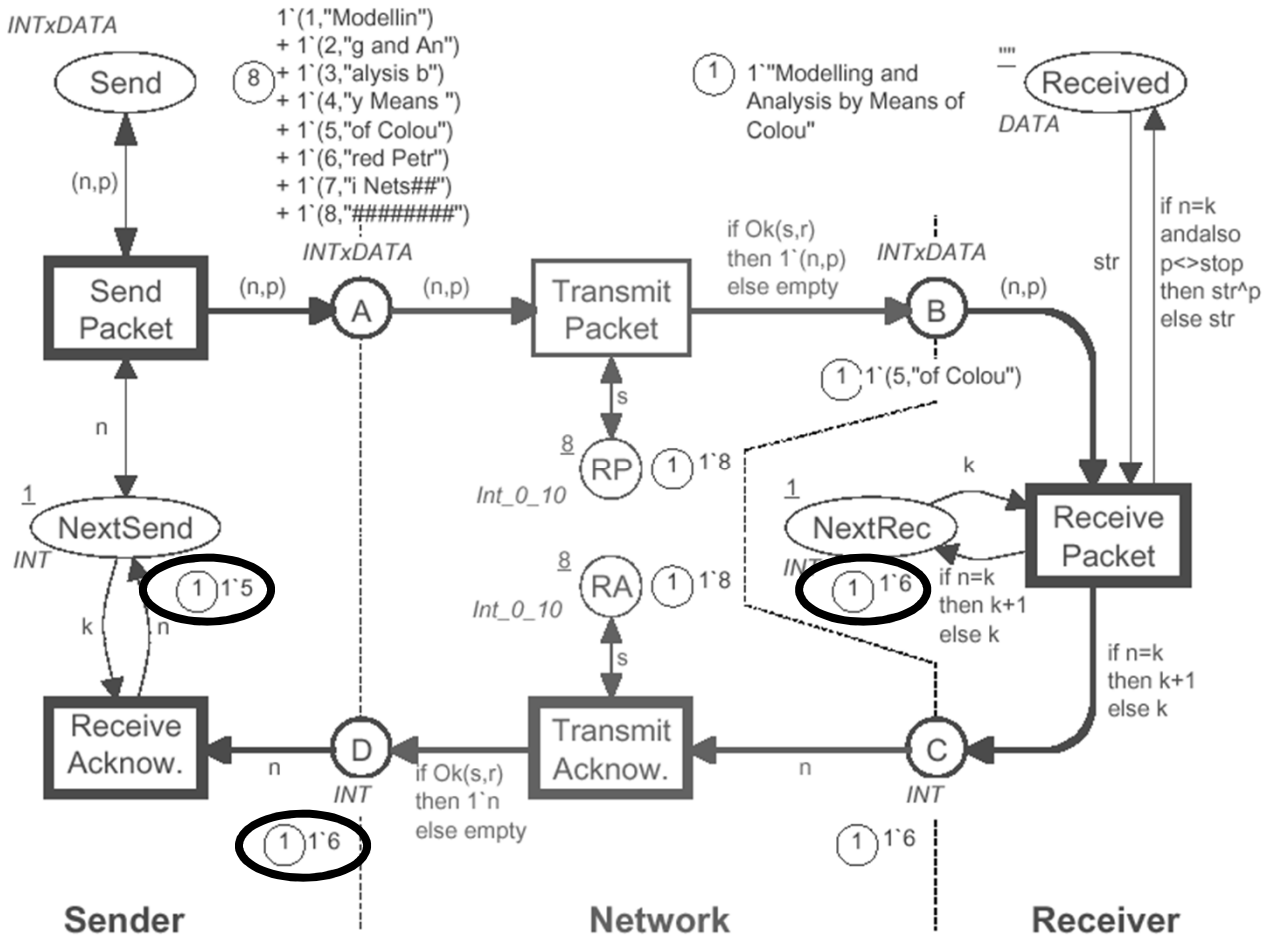
Interaktív szimuláció



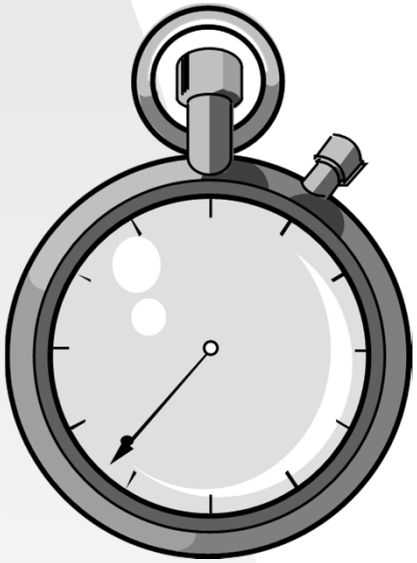
- *A Szimuláció* eredményét az ábrán láthatjuk.
  - *A következő tüzelést* manuálisan vagy automatikusan lehet kijelölni.
  - A felhasználó *watchpointot* és *breakpointokat* definiálhat.

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.

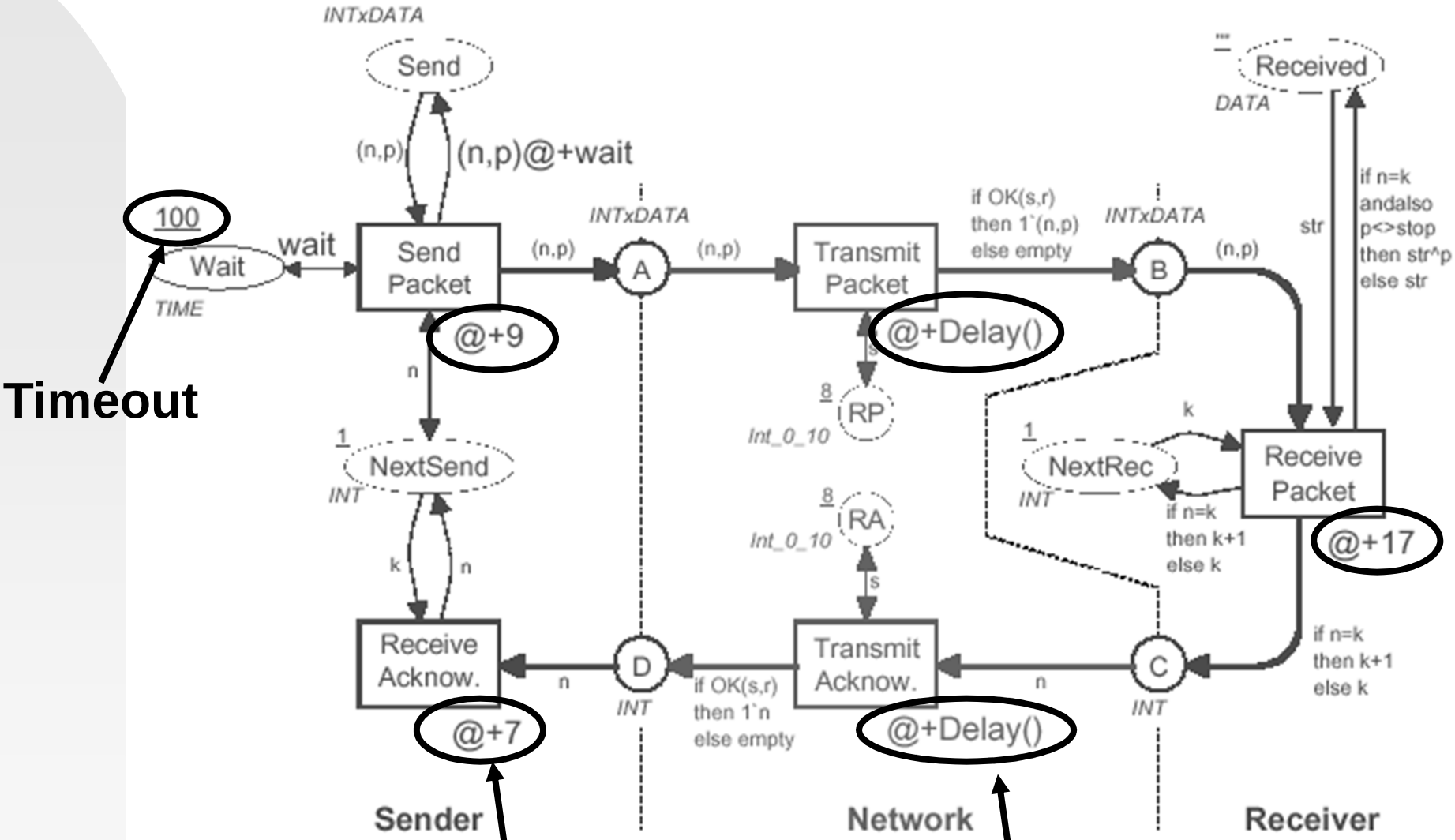


Idő analízis



- CPN-hez az *idő* fogalmát bevezethetjük. Ezzel ezeket nyertük:
  - *Logikai helyesség*.
    Megkívánt funkcionalitás, deadlock mentesség, stb.
  - *Teljesítmény*.
    Szűk keresztmetszetek.
    Várakozási idő és átlagos átbocsátó képesség előrejelzés.
- Egy időzített színezett Petri-hálóban minden tokennek *színe* (értéke) és egy *időbélyege* (mikor használható fel) van.

Egy időzített CPN a protokollunkhoz

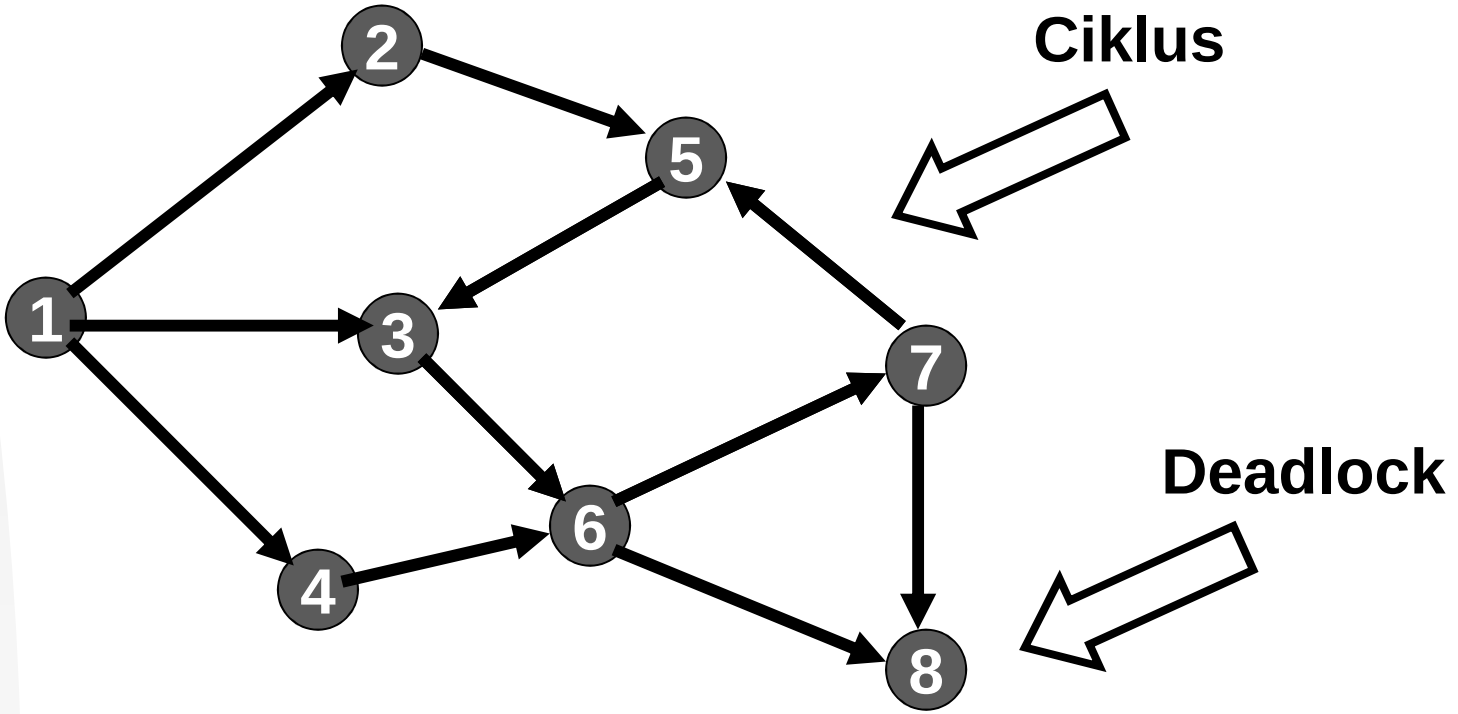


A CPN verifikációja

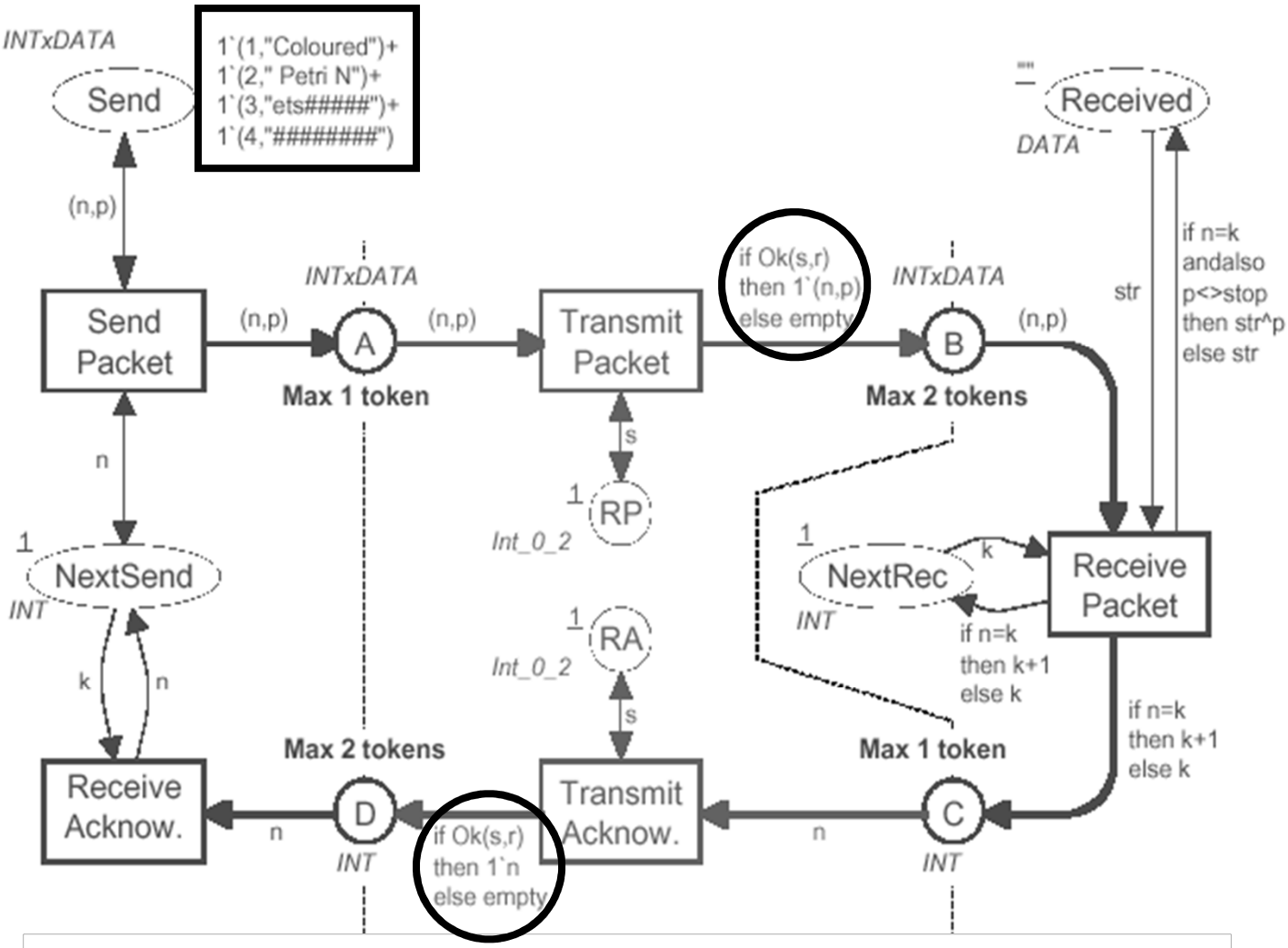
- A CPN verifikációja a következőkből áll:
  - *Állapottér*.
  - Hely és tranzíció *invariánsok.*
    - Hasonlóan a programozási nyelvek invariánsaihoz.

Állapottér

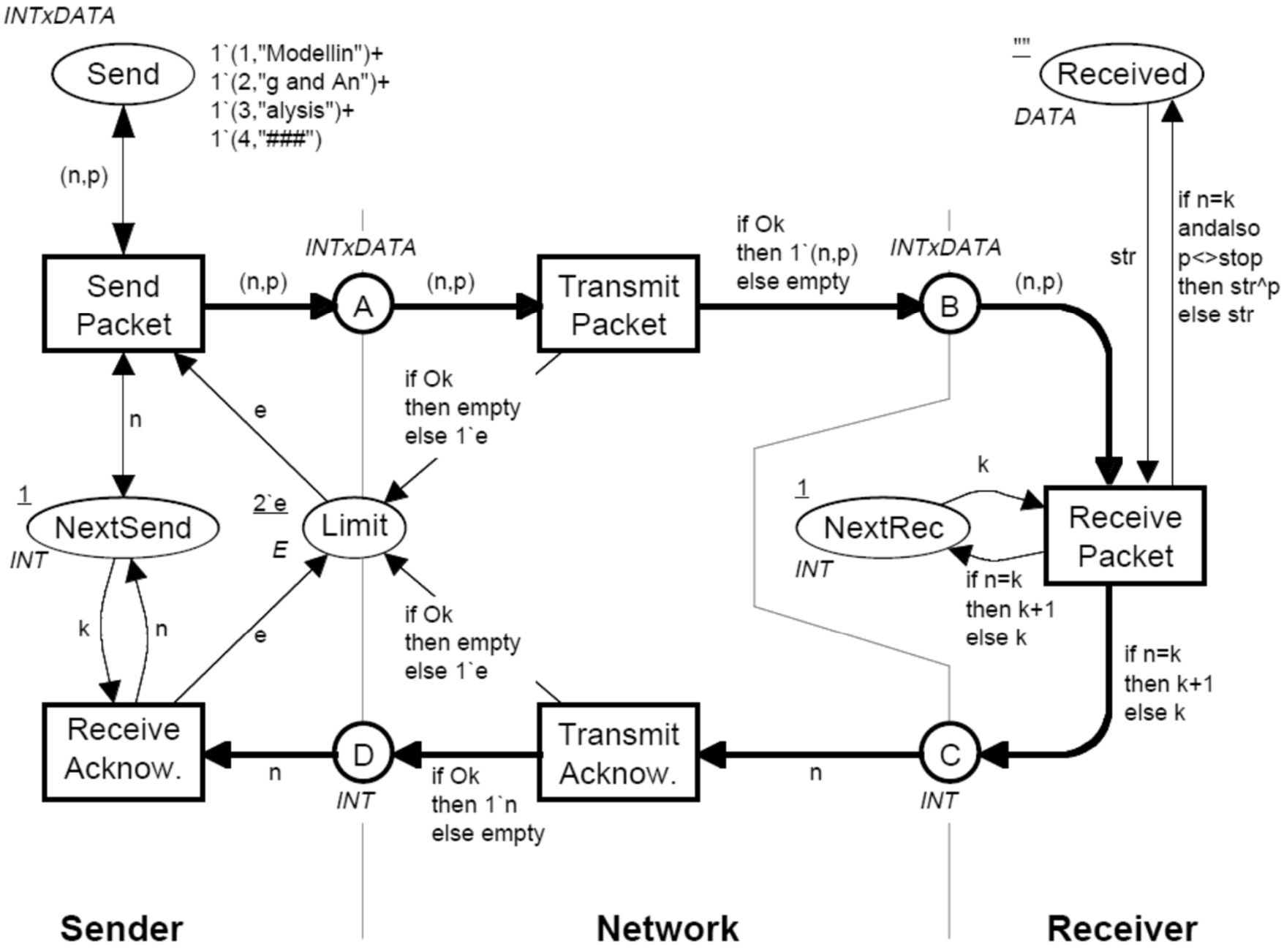
- Az *állapottér* egy irányított gráf, ahol:
  - A csomópont az *elérhető állapot*.
  - A nyíl a lehetséges tranzíciók (+ párosítás).



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
Send
1`(1,"Modellin")+
1`(2,"g and An")+
1`(3,"alysis")+
1`(4,"###")
(n,p)
Send Packet
INTxDATA
(n,p)
A
(n,p)
Transmit Packet
if Ok
then 1`(n,p)
else empty
INTxDATA
B
(n,p)
""
Received
DATA
str
if n=k
andalso
p<>stop
then str^p
else str
n
e
if Ok
then empty
else 1`e
1
NextSend
INT
2`e
E
Limit
k
1
NextRec
INT
Receive Packet
if n=k
then k+1
else k
k
n
e
if Ok
then empty
else 1`e
if n=k
then k+1
else k
Receive Acknow.
n
D
INT
if Ok
then 1`n
else empty
Transmit Acknow.
n
C
INT
Sender
Network
Receiver

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`(1,"Coloured")+ 1`(2," Petri N")+ 1`(3,"ets#####”)+ 1`(4,"########")**
B:	**2`(1,"Coloured")+ 2`(2," Petri N")+ 2`(3,"ets#####")+ 2`(4,"########")**
C:	**1`2+ 1`3+ 1`4+ 1`5**
D:	**2`2+ 2`3+ 2`4+ 2`5**
NextRec:	**1`1+1`2+1`3+1`4+1`5**
NextSend:	**1`1+1`2+1`3+1`4+1`5**
RA:	**1`1**
RP:	**1`1**
Received:	**1`""+ 1`"Coloured"+ 1`"Coloured Petri N"+ 1`"Coloured Petri Nets#####"**
Send:	**1`(1,"Coloured")+ 1`(2," Petri N")+ 1`(3,"ets#####")+ 1`(4,"########")**

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

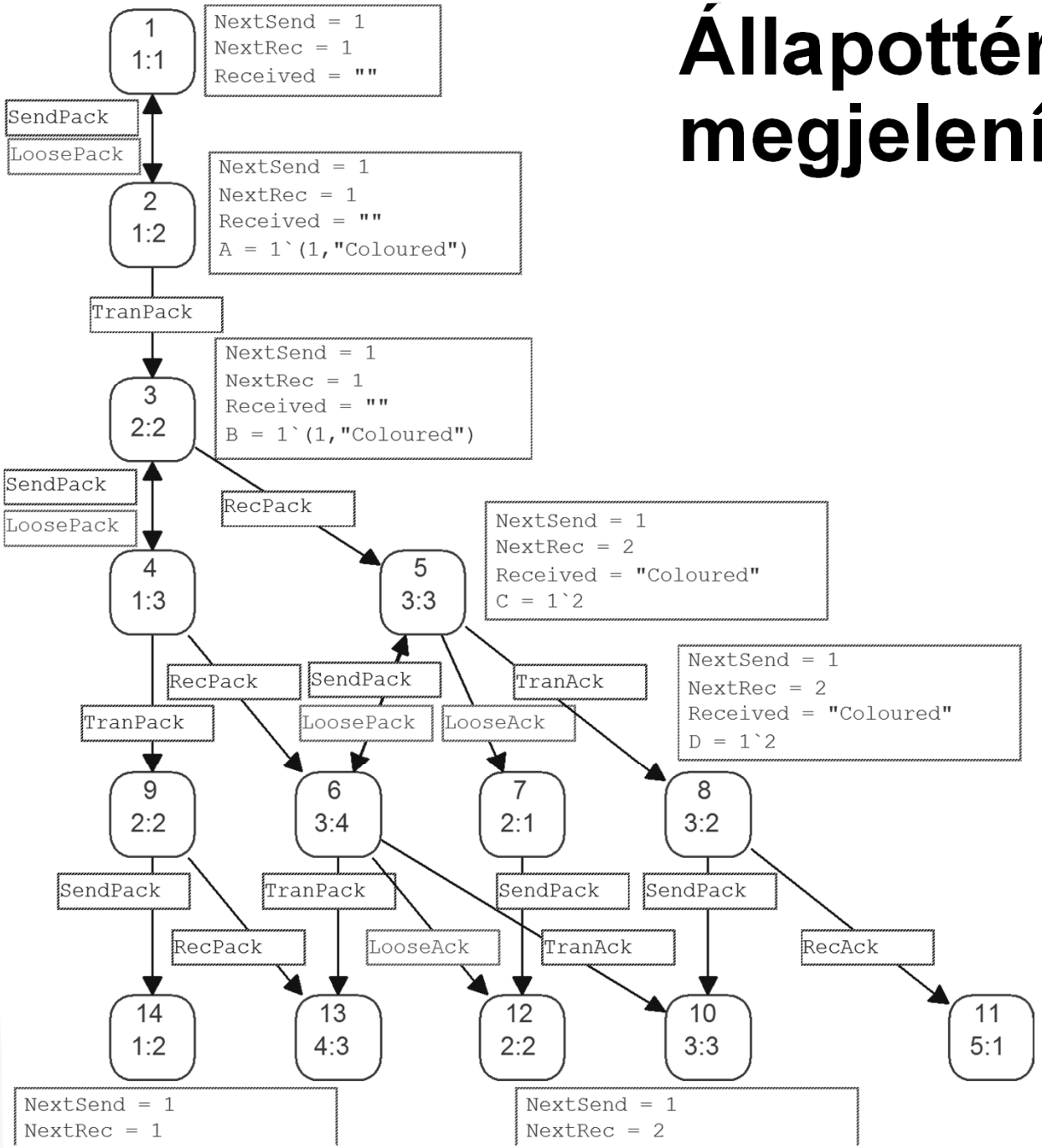
- A 452. tokeneloszlás a *dead*
  - Ez azt jelenti, hogy ez a protokoll *részlegesen korrekt* (ha lefut az algoritmus, akkor a végleges eloszlást veszi fel).
- Ez egyben *visszatérő állapot is*
  - Emiatt *mindig megvan a lehetőség arra, hogy helyesen fejezzük be* (az lehetetlen, hogy olyan állapotba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



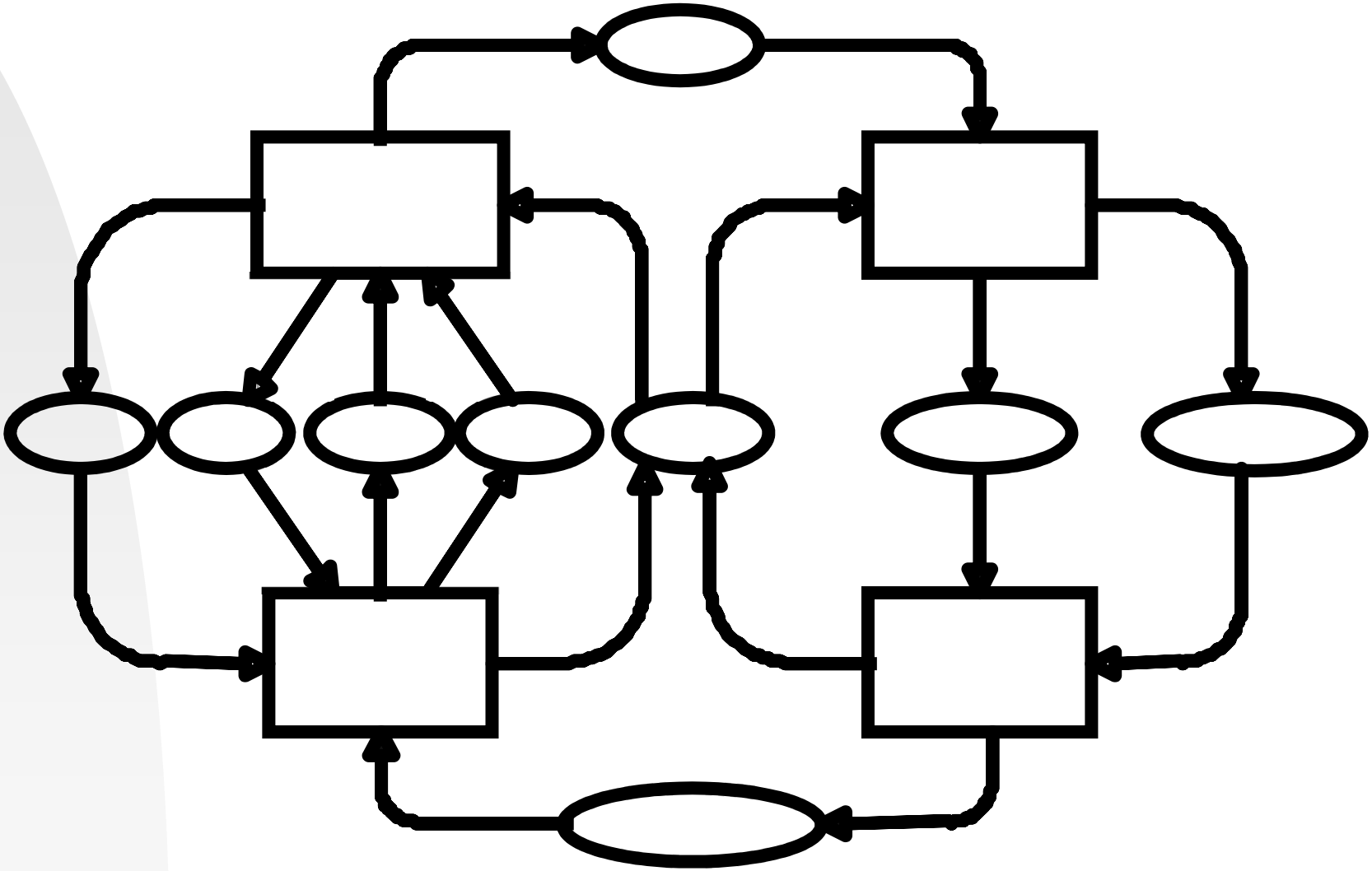
Temporális logika

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

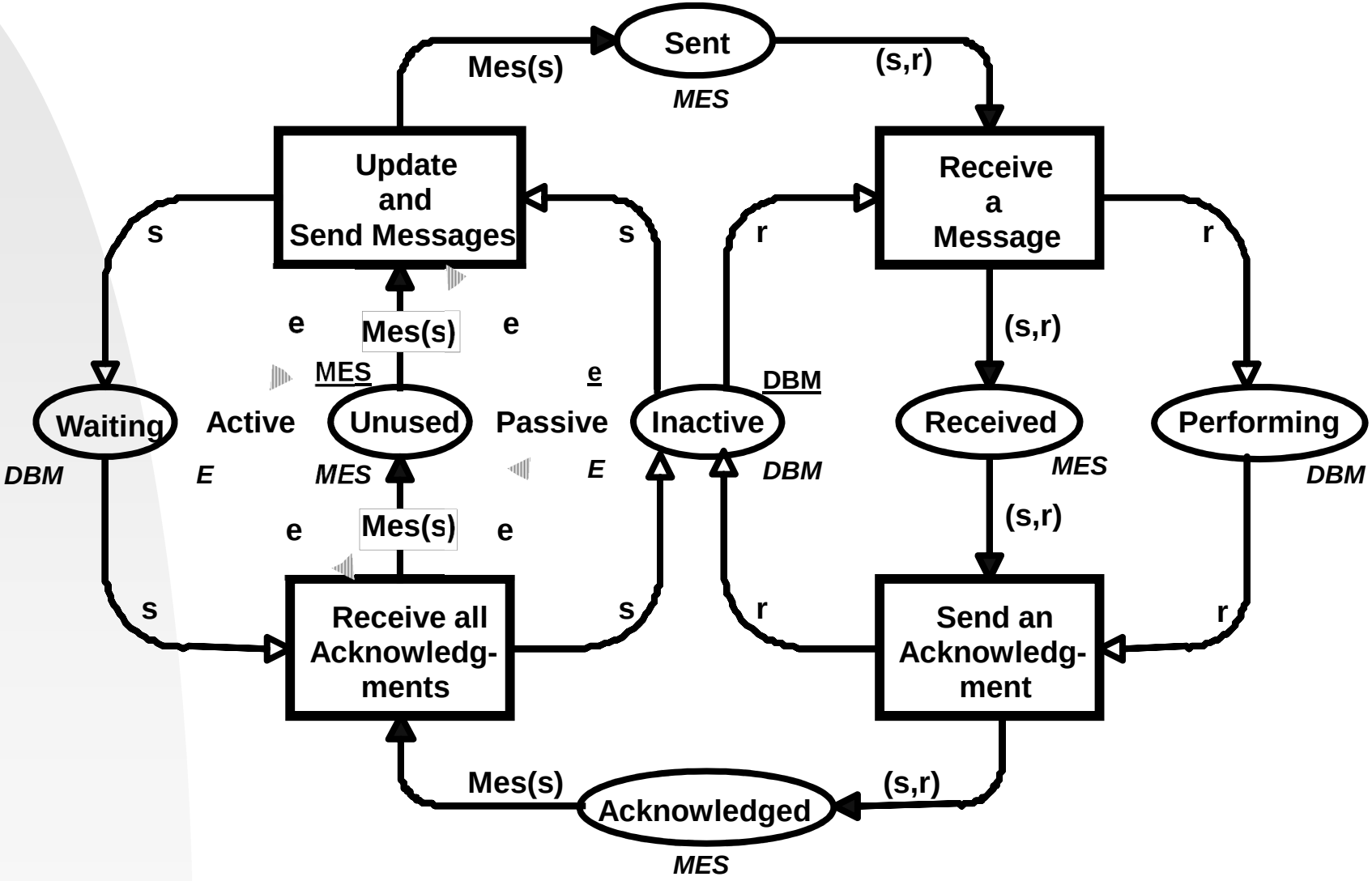
Állapottér analízis – előnyök/hátrányok

- 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*.

A Petri net közösség logója

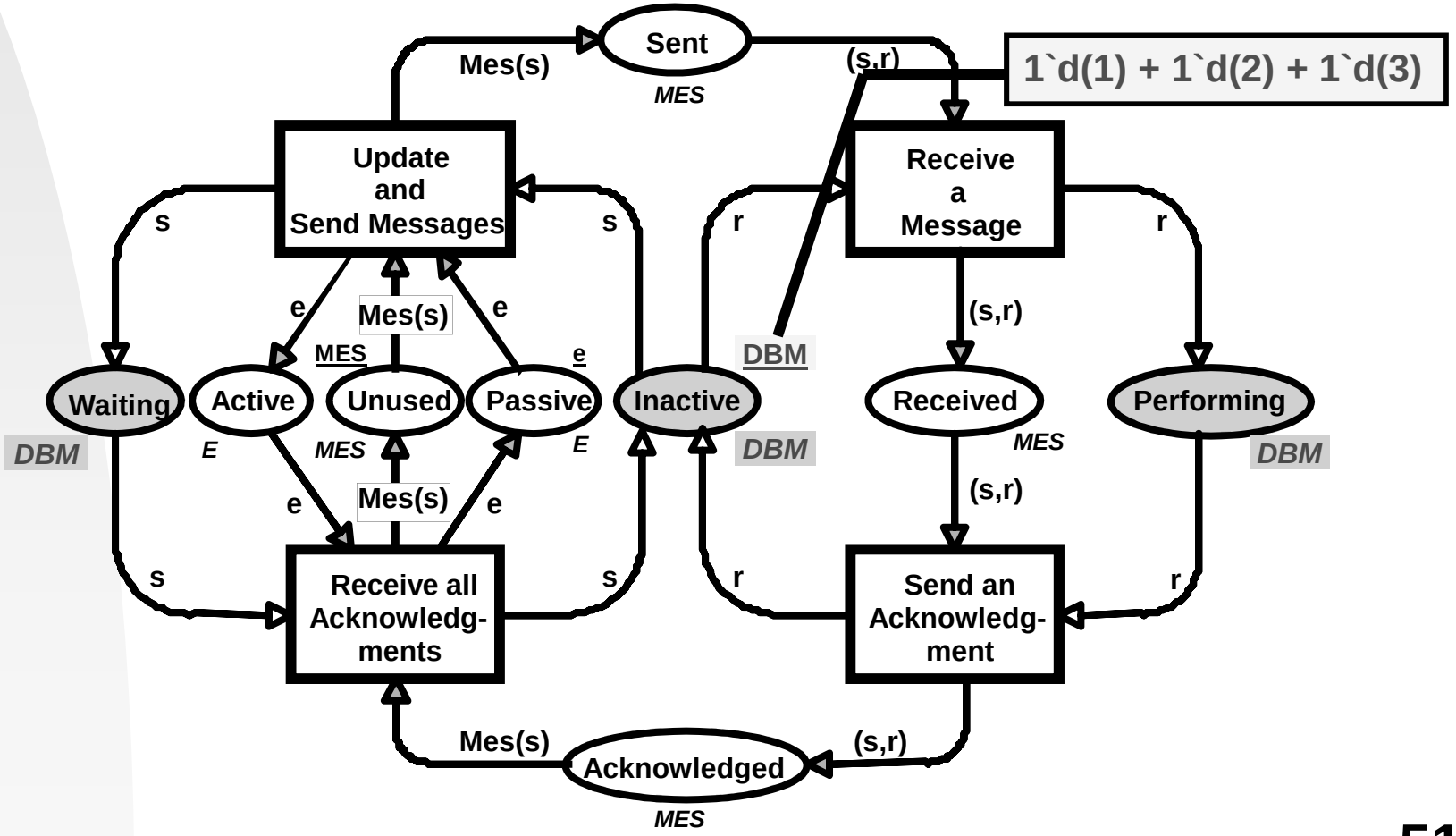


Elosztott adatbázis



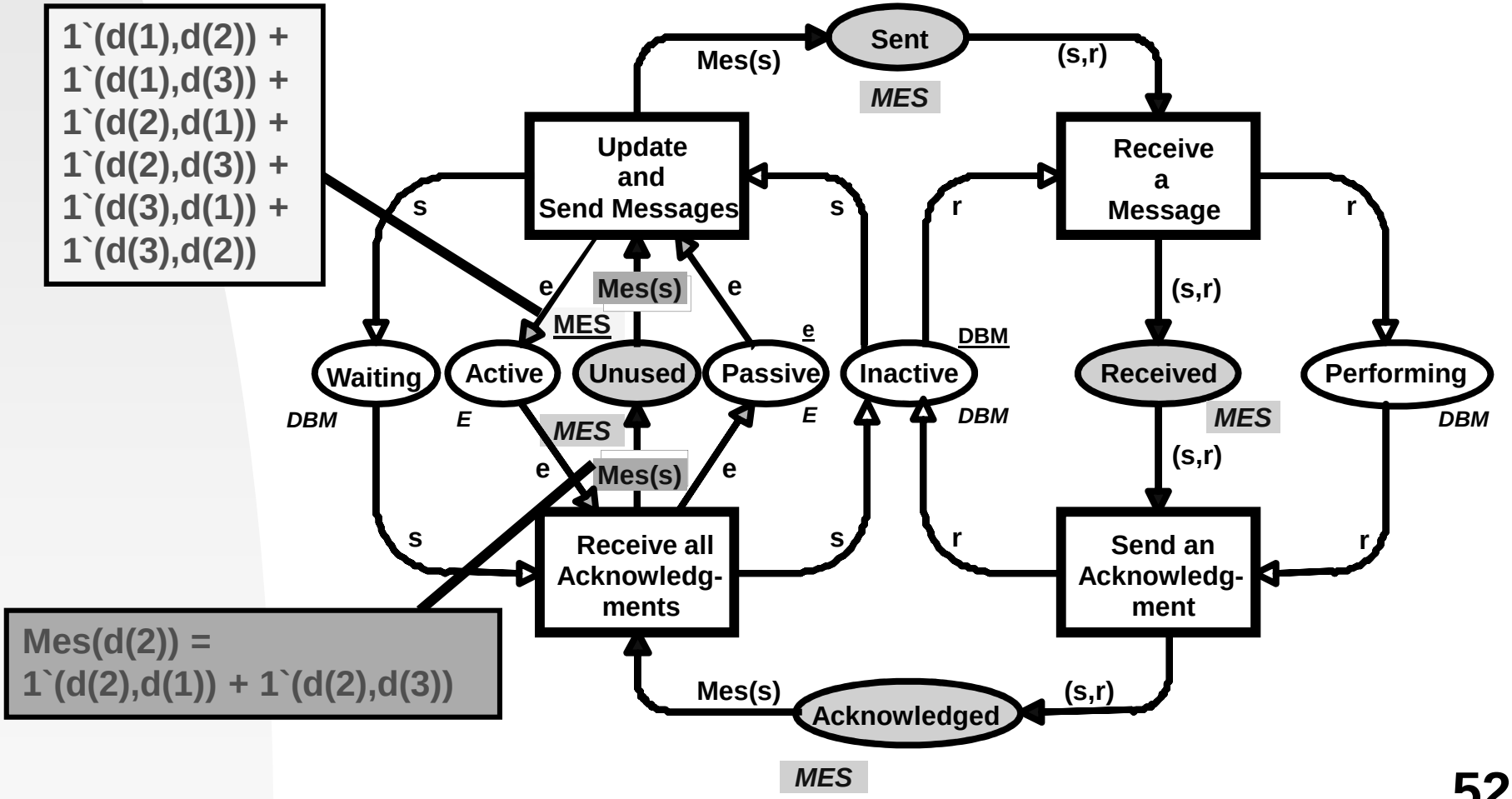
Adatbázis kezelők

- DBM = {d(1),d(2),d(3)}



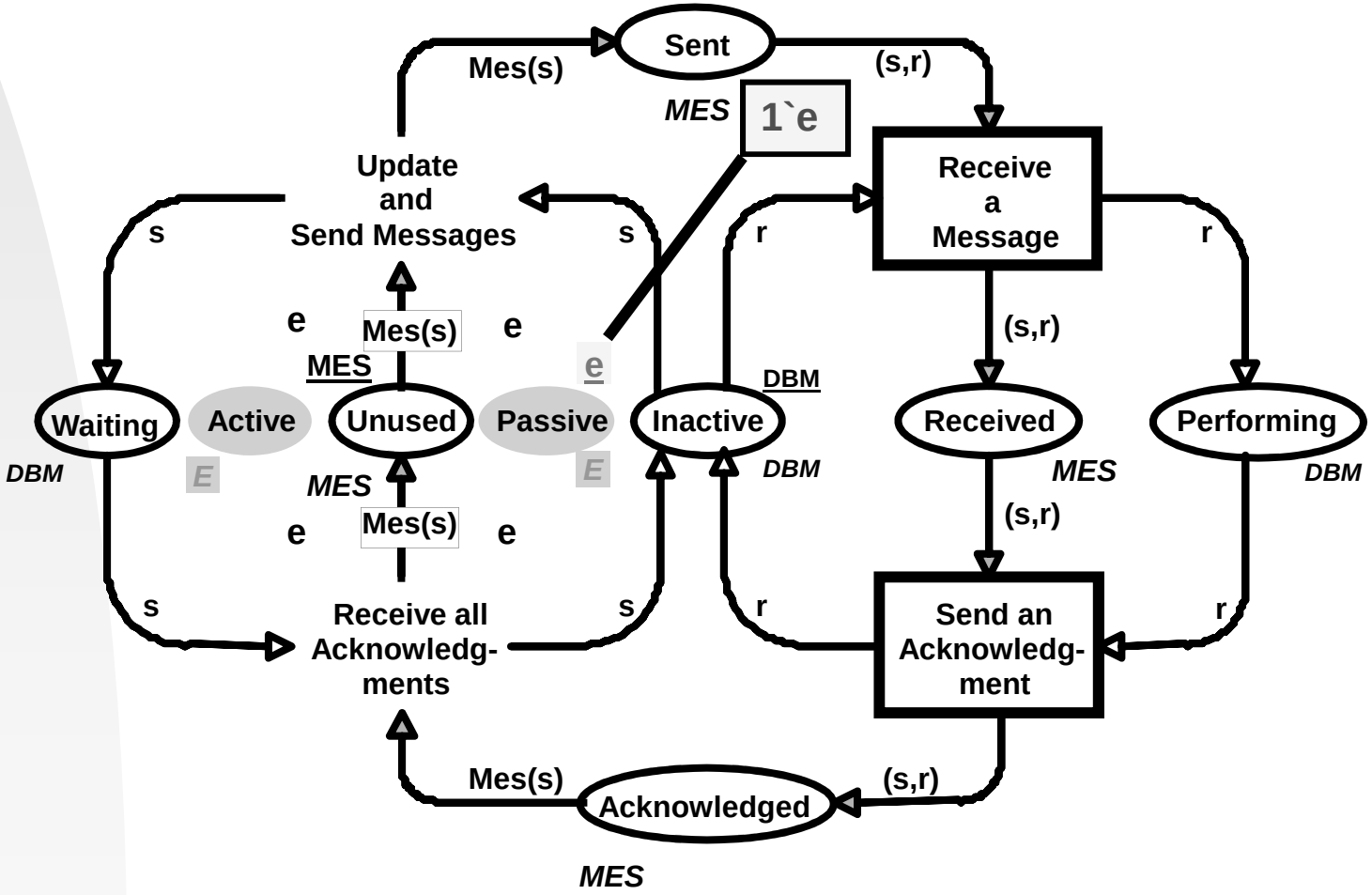
Üzenet bufferek

- MES = {(s,r)∈ DBM×DBM | s ≠ r}

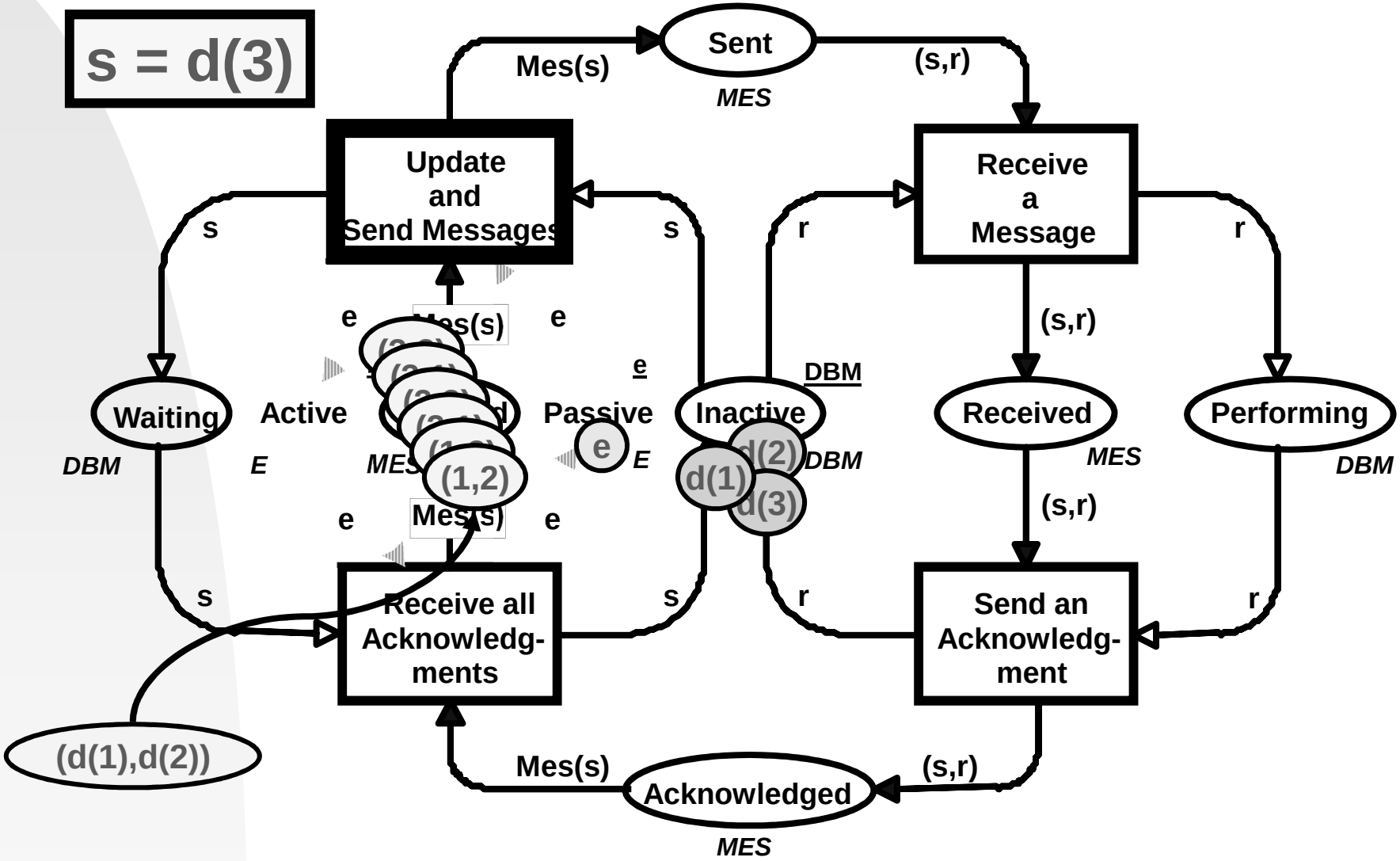


Kölcsönös kizárás

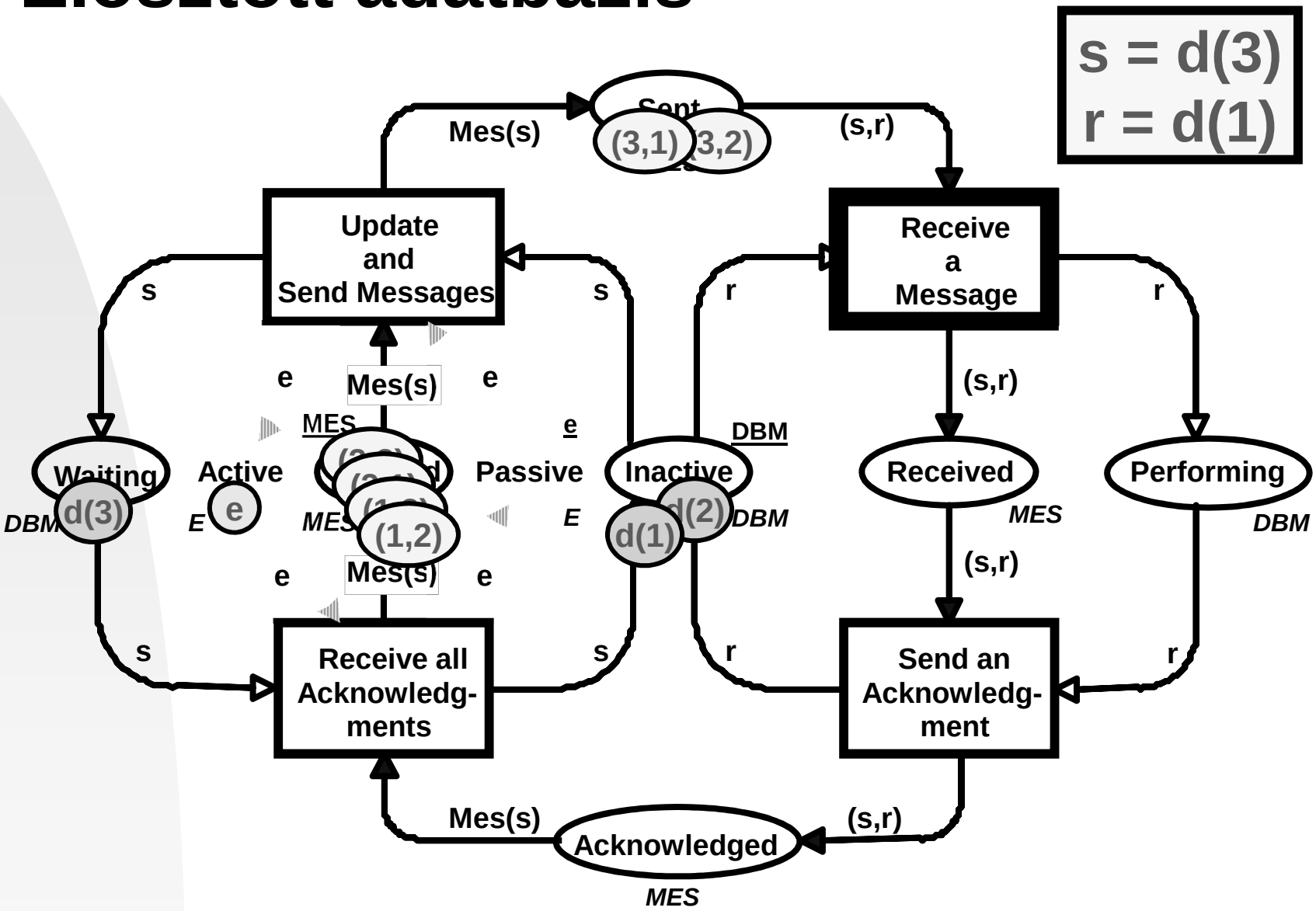
- E = {e}



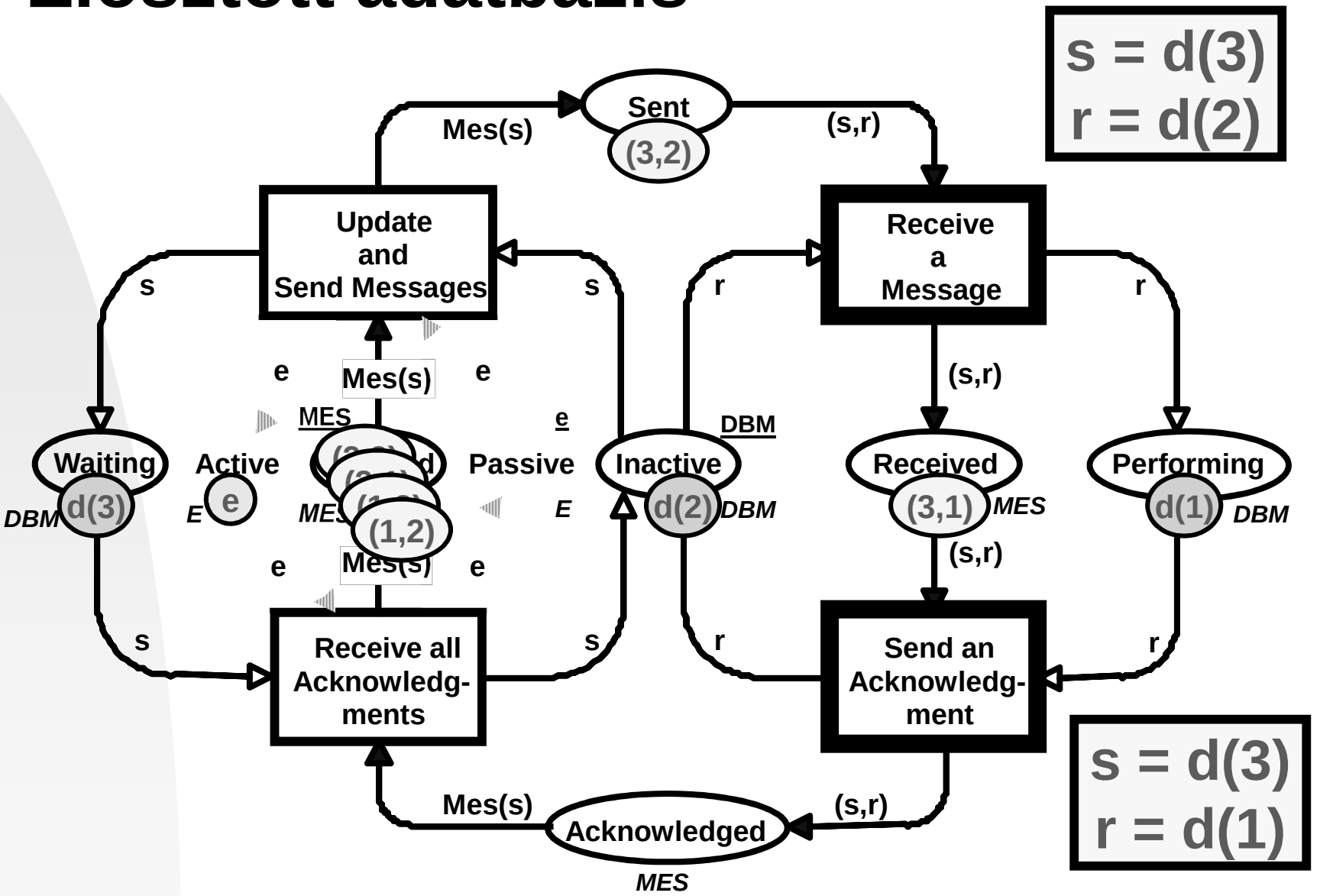
Elosztott adatbázis



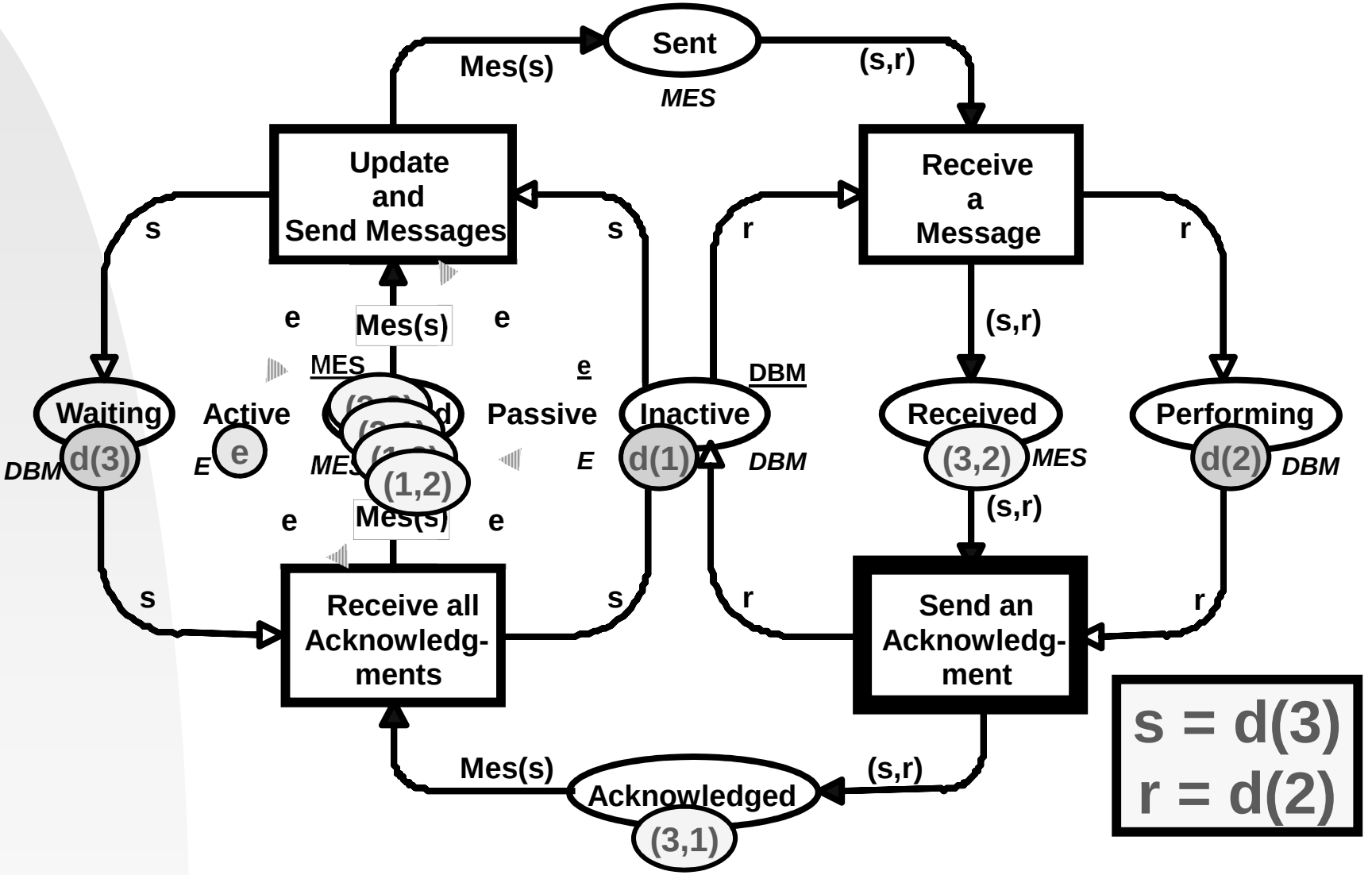
Elosztott adatbázis



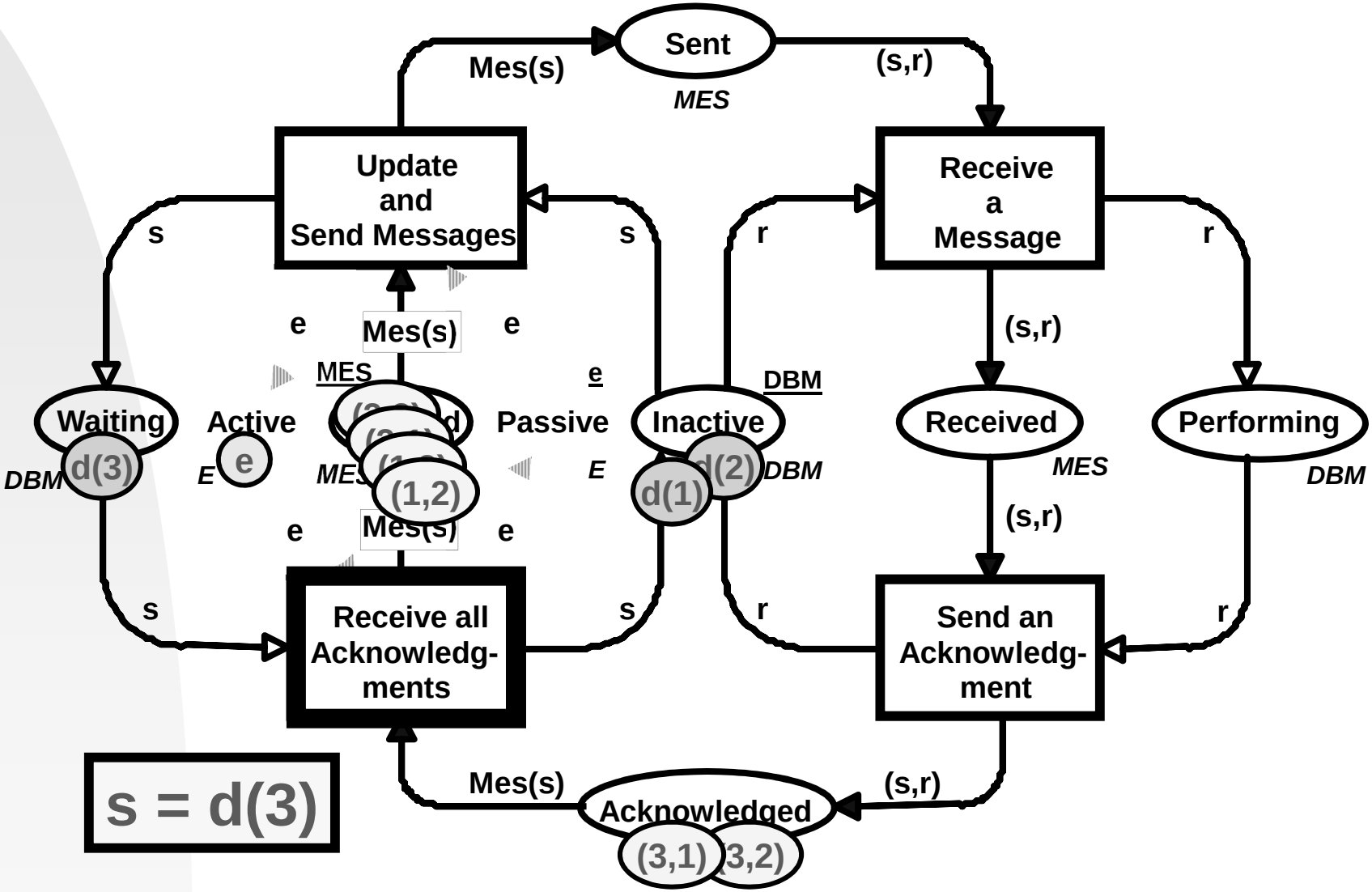
Elosztott adatbázis



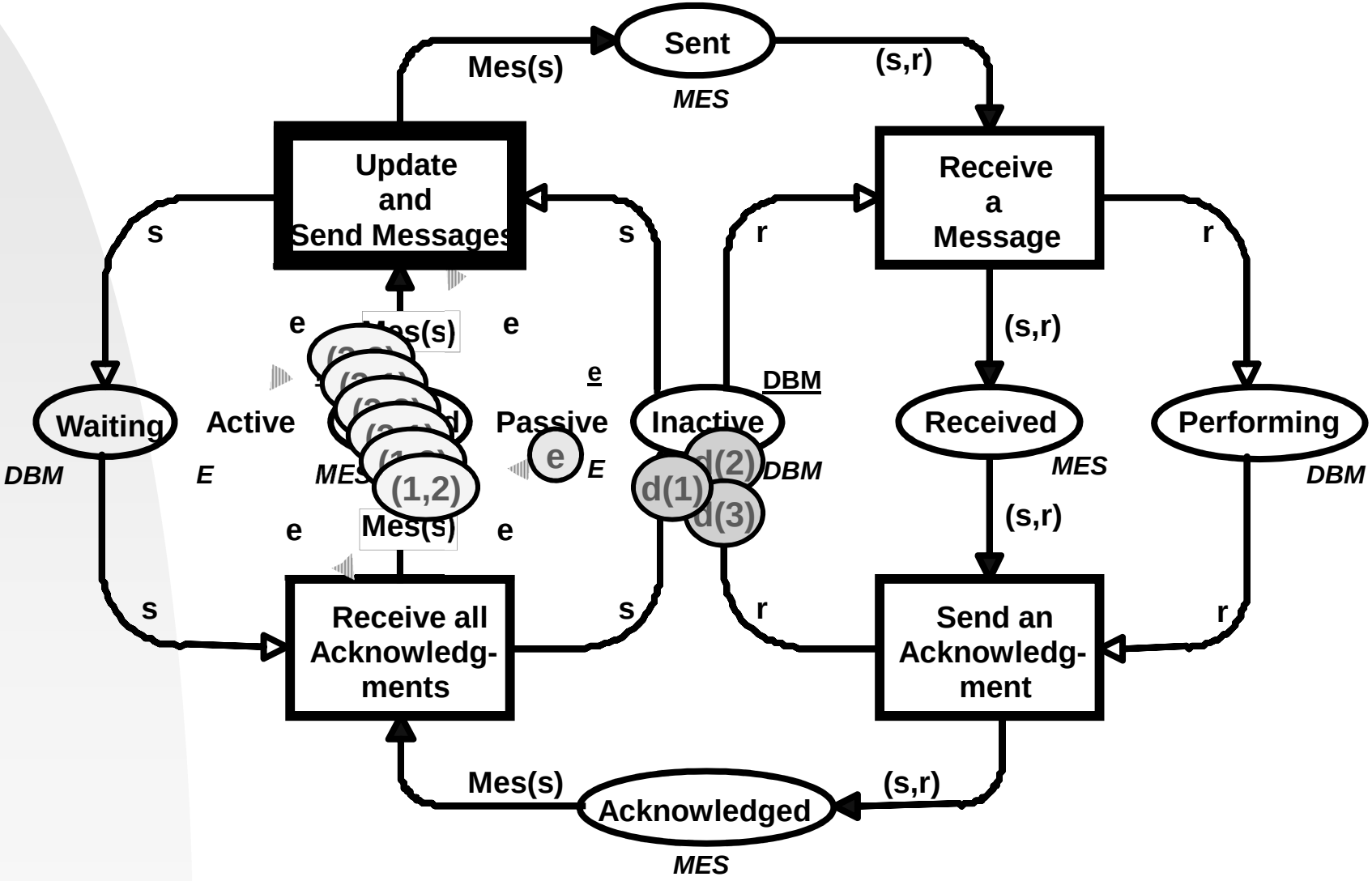
Elosztott adatbázis



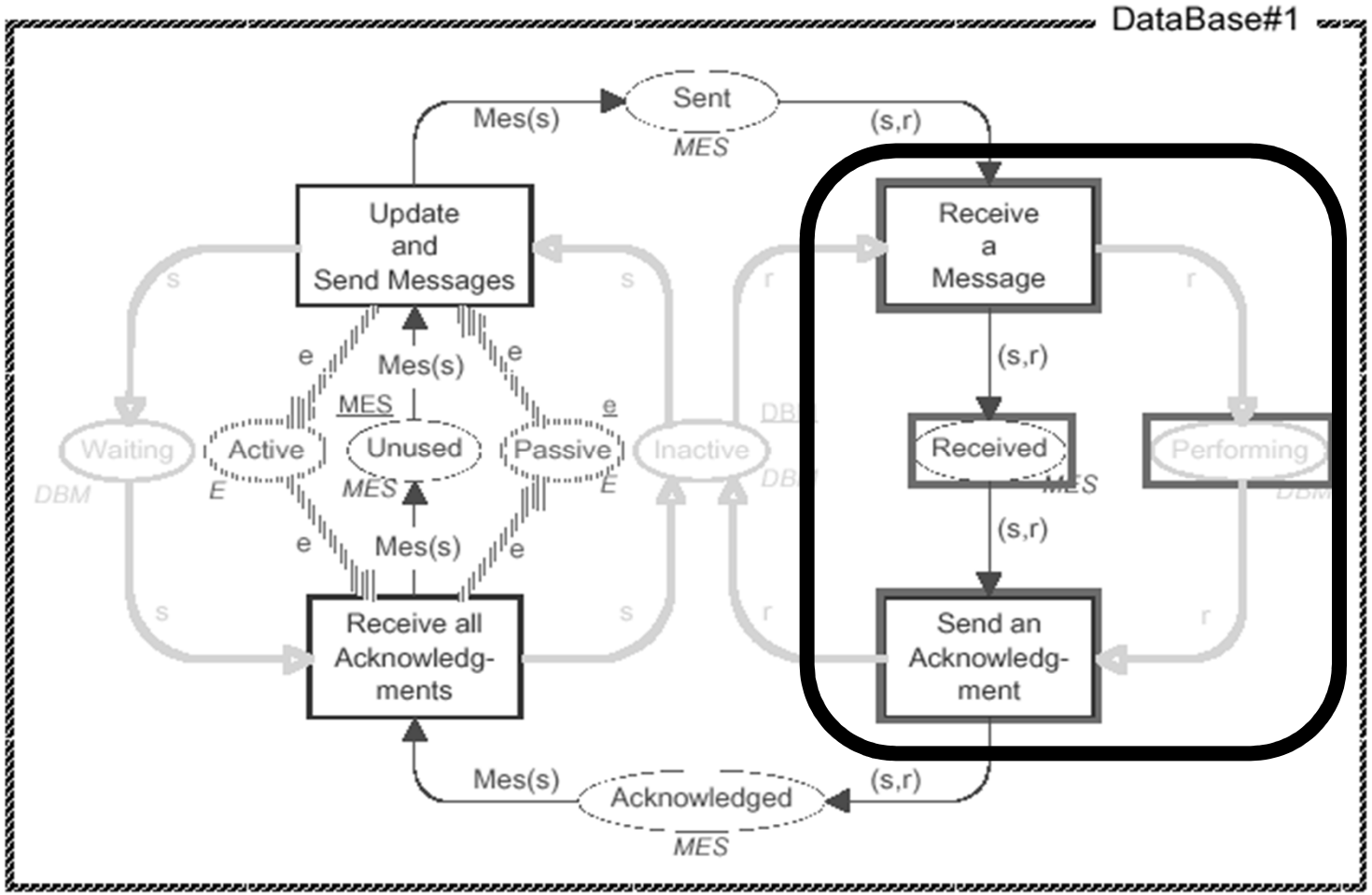
Elosztott adatbázis



Elosztott adatbázis

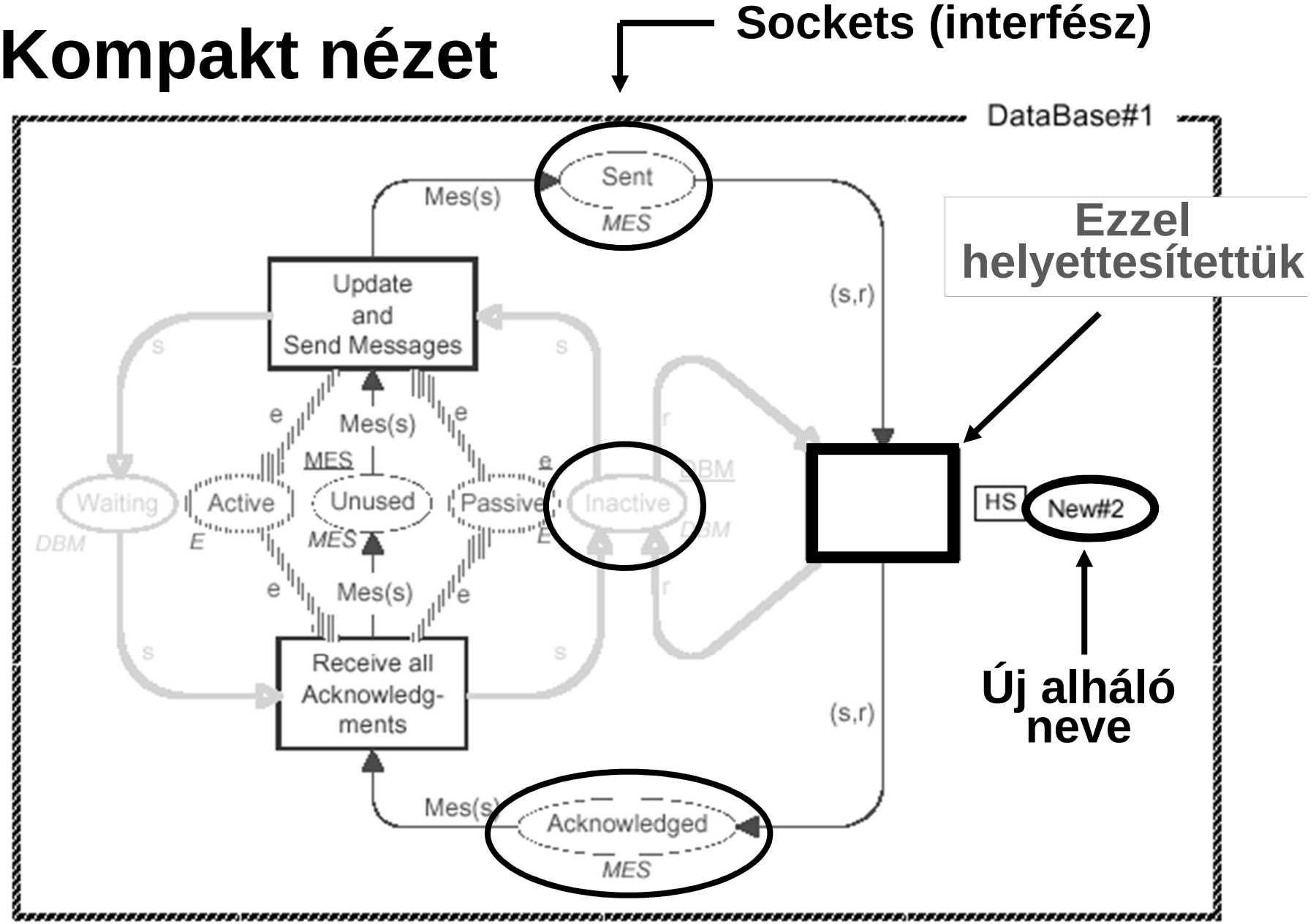


Hierarchikus modellek támogatása

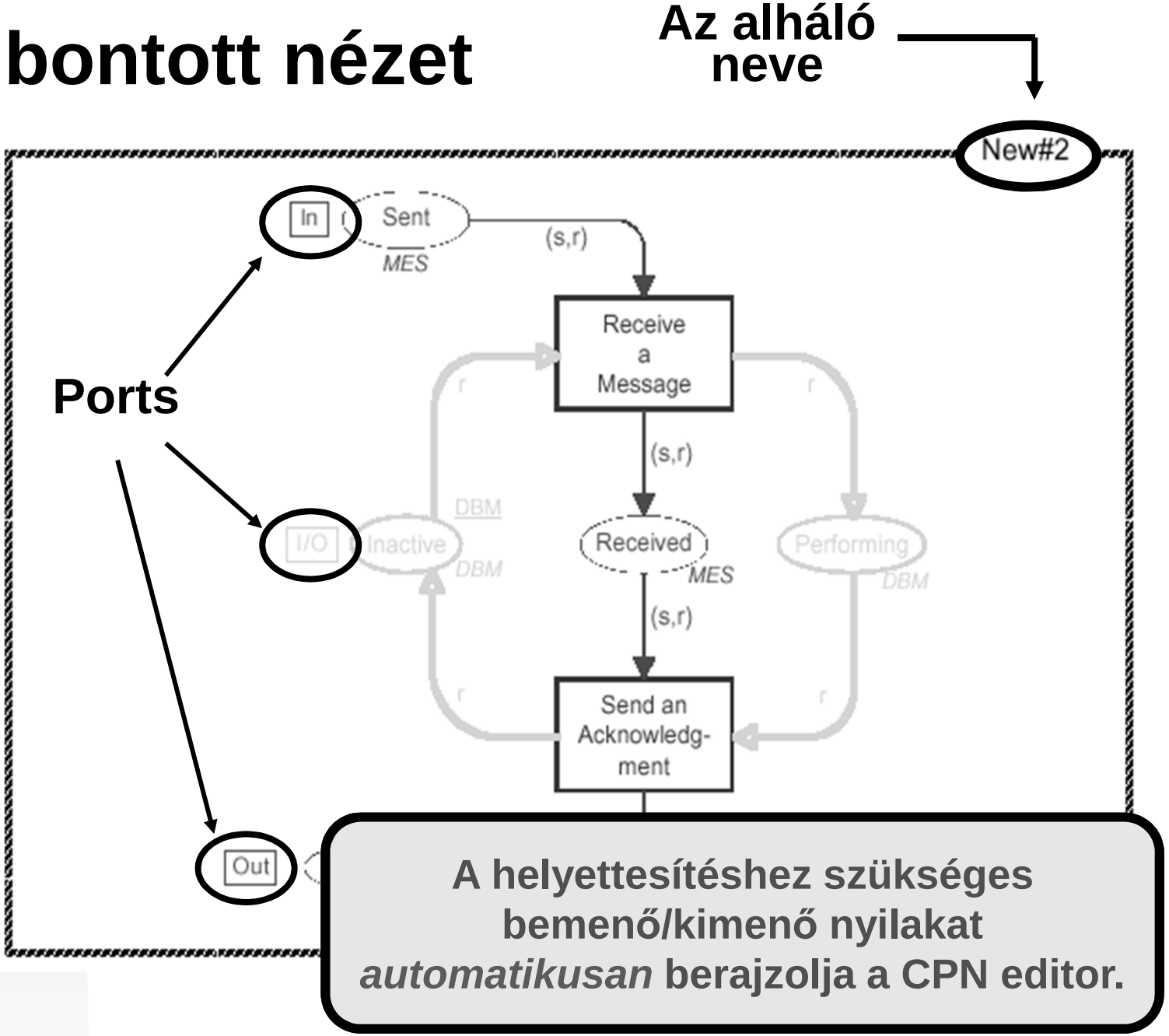


- A ***kijelölt részt*** könnyen helyettesíthetjük egyetlen alhálóval.

Kompakt nézet



Kibontott nézet



Hely invariánsok

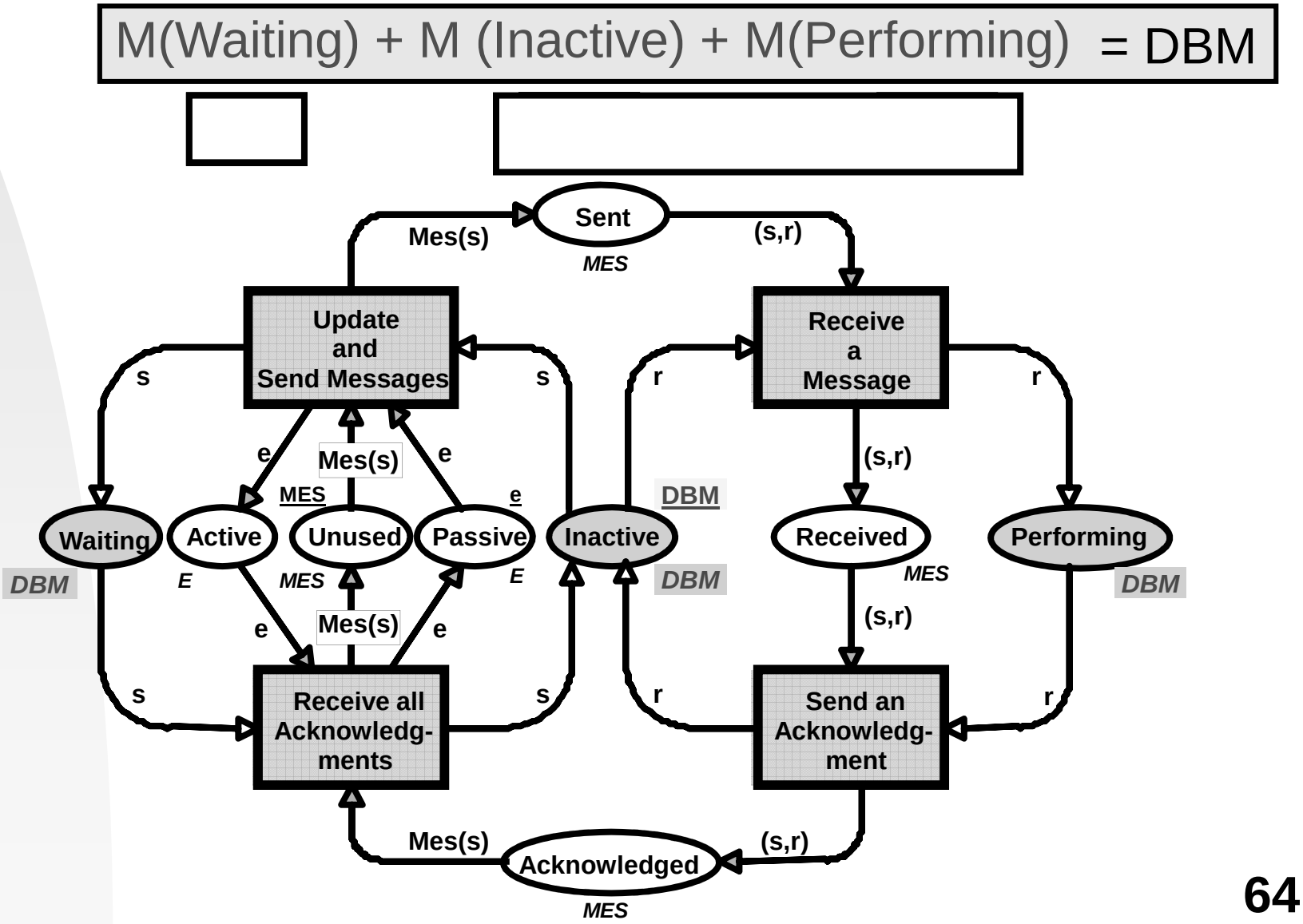
Place → M(Place)

- 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.

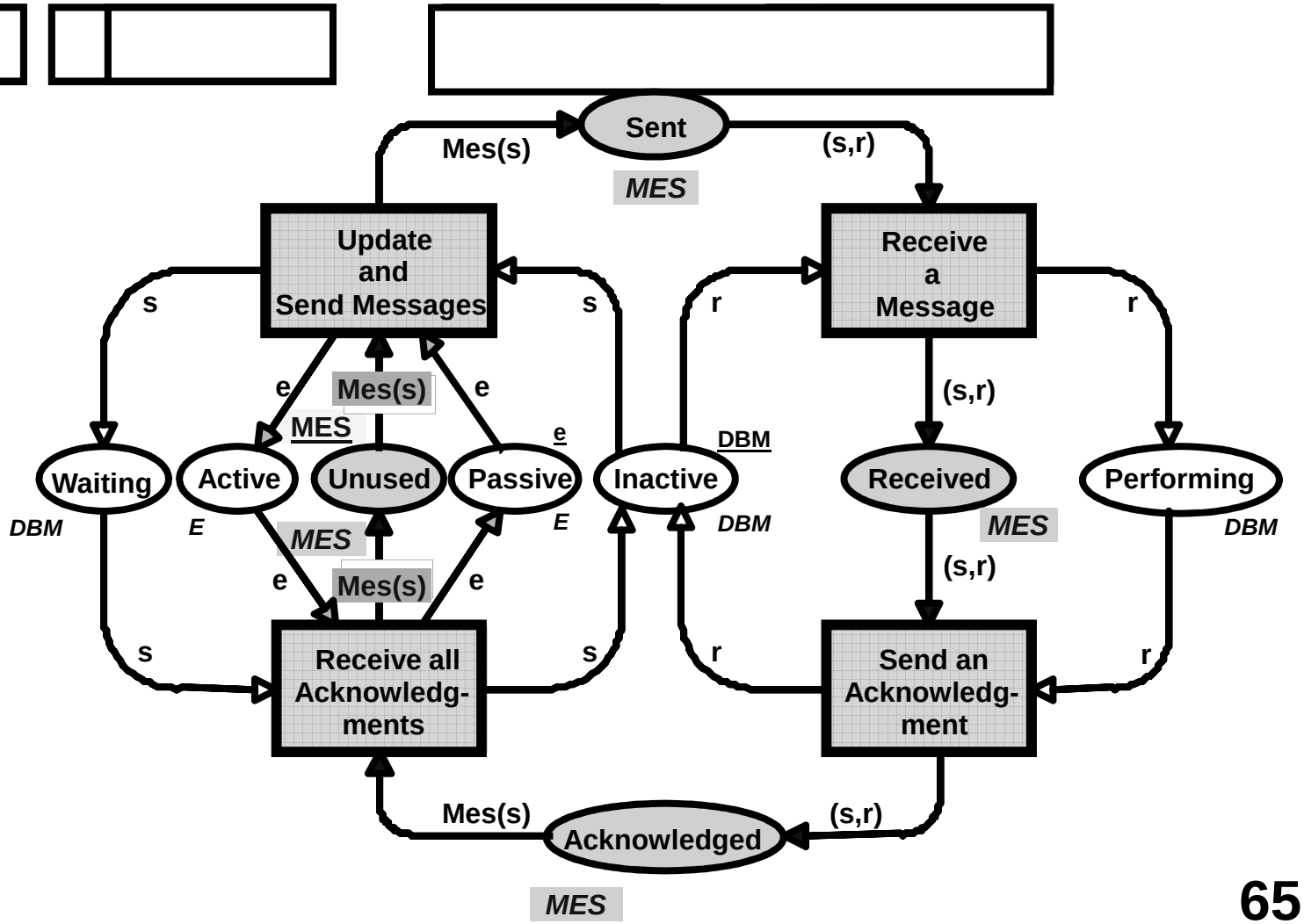
- *Ign*(Waiting) + Passive = E

Adatbázis kezelők

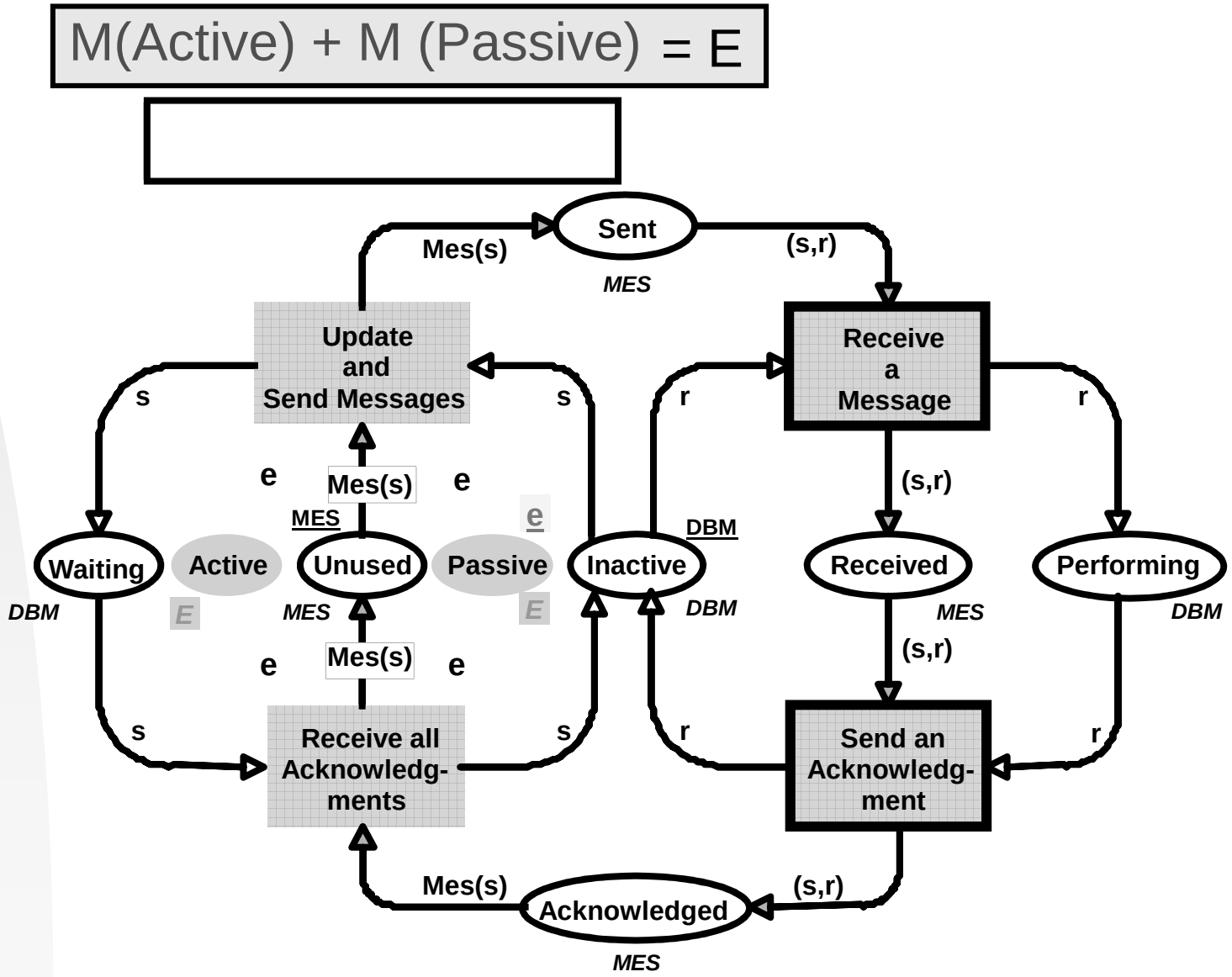


Üzenet bufferek

M(Unused) + M(Sent) + M (Received) + M(Acknowl) = MES



Kölcsönös kizárás



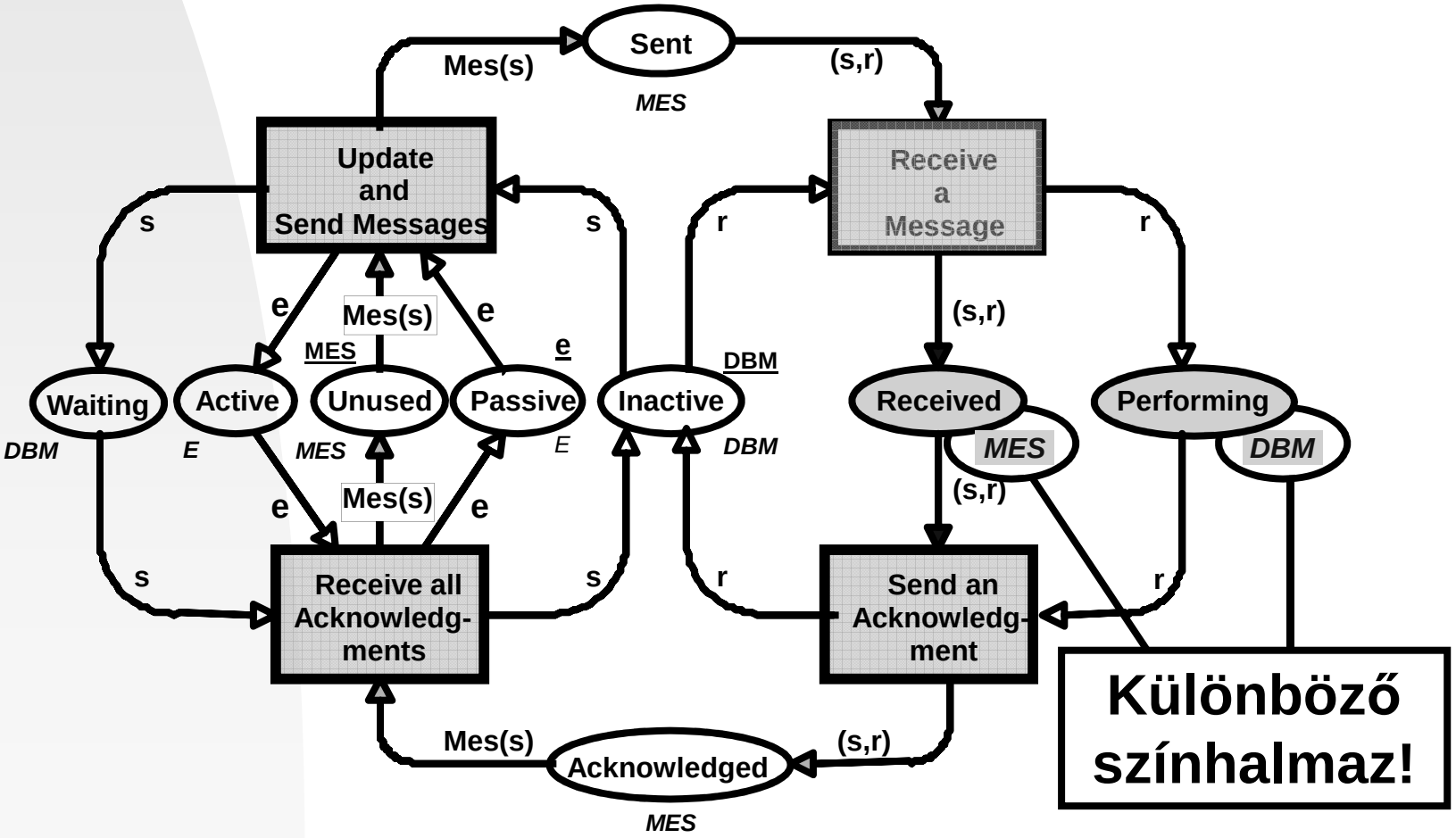
Fogadott üzenetek

Rec(s,r) = r

Súly függvény

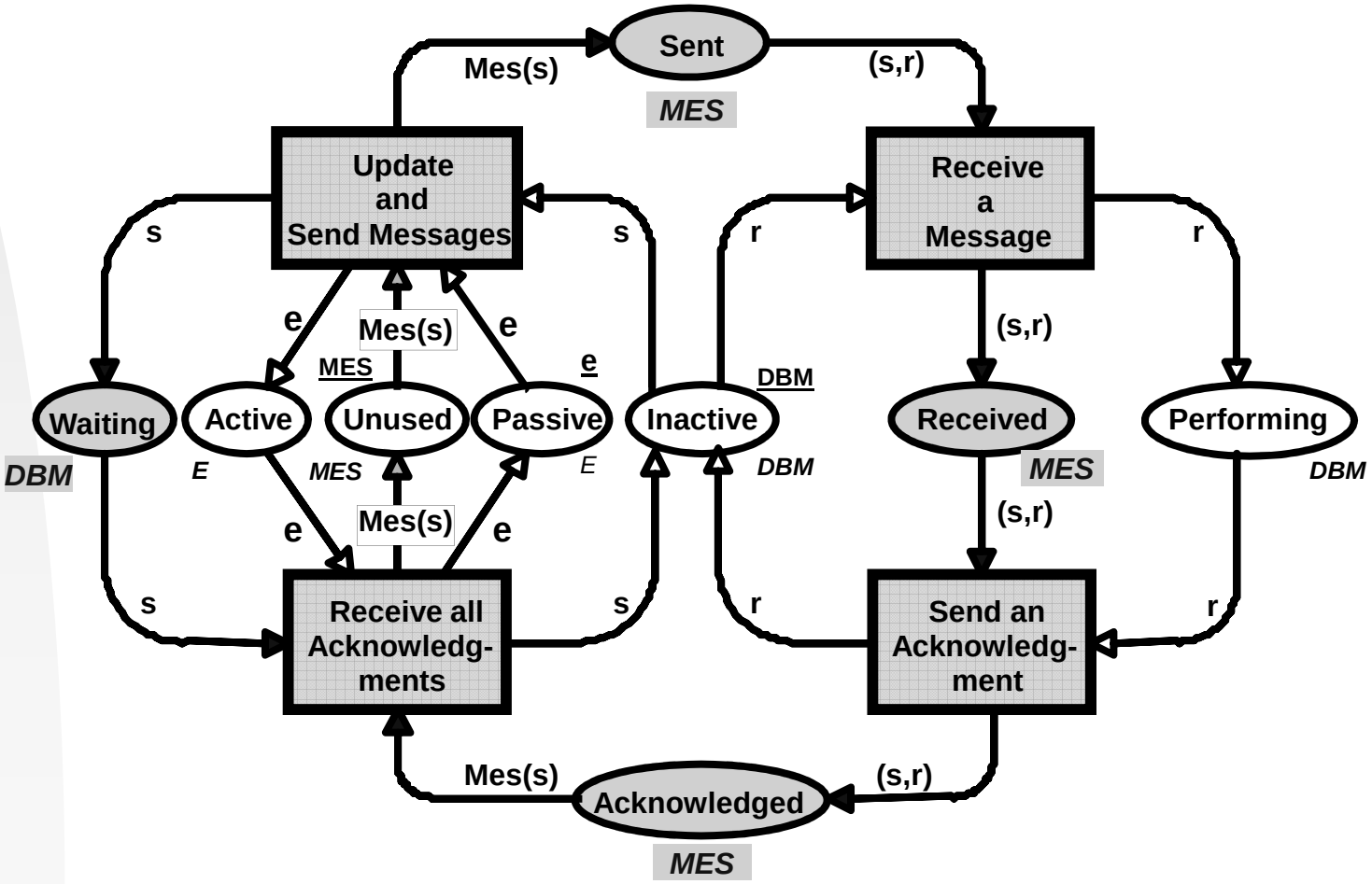
Rec(M(Received)) = M(Performing)

MES → DBM



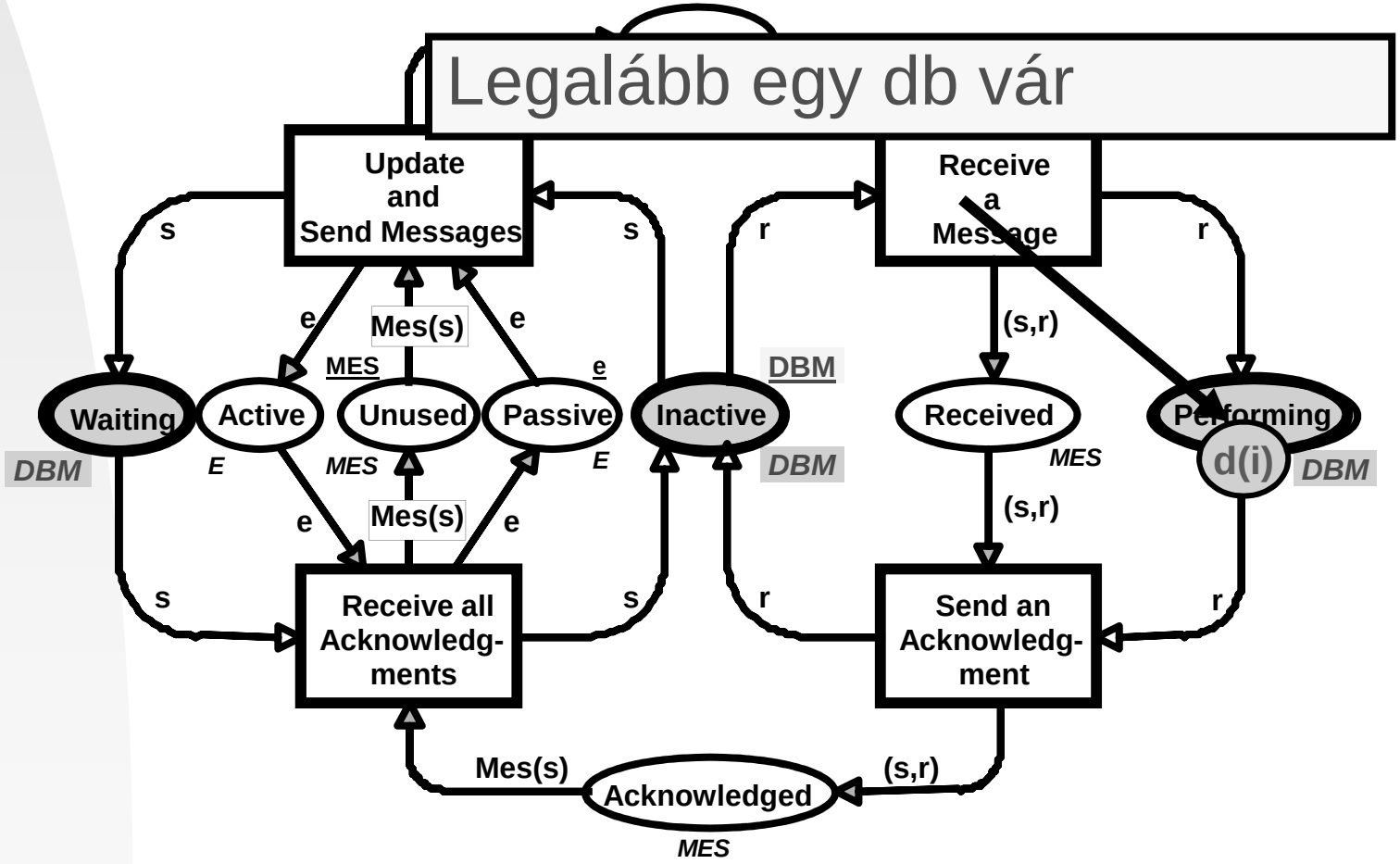
Használt üzenetek

Mes(Waiting) = M(Sent) + M (Received) + M(Acknowledged)



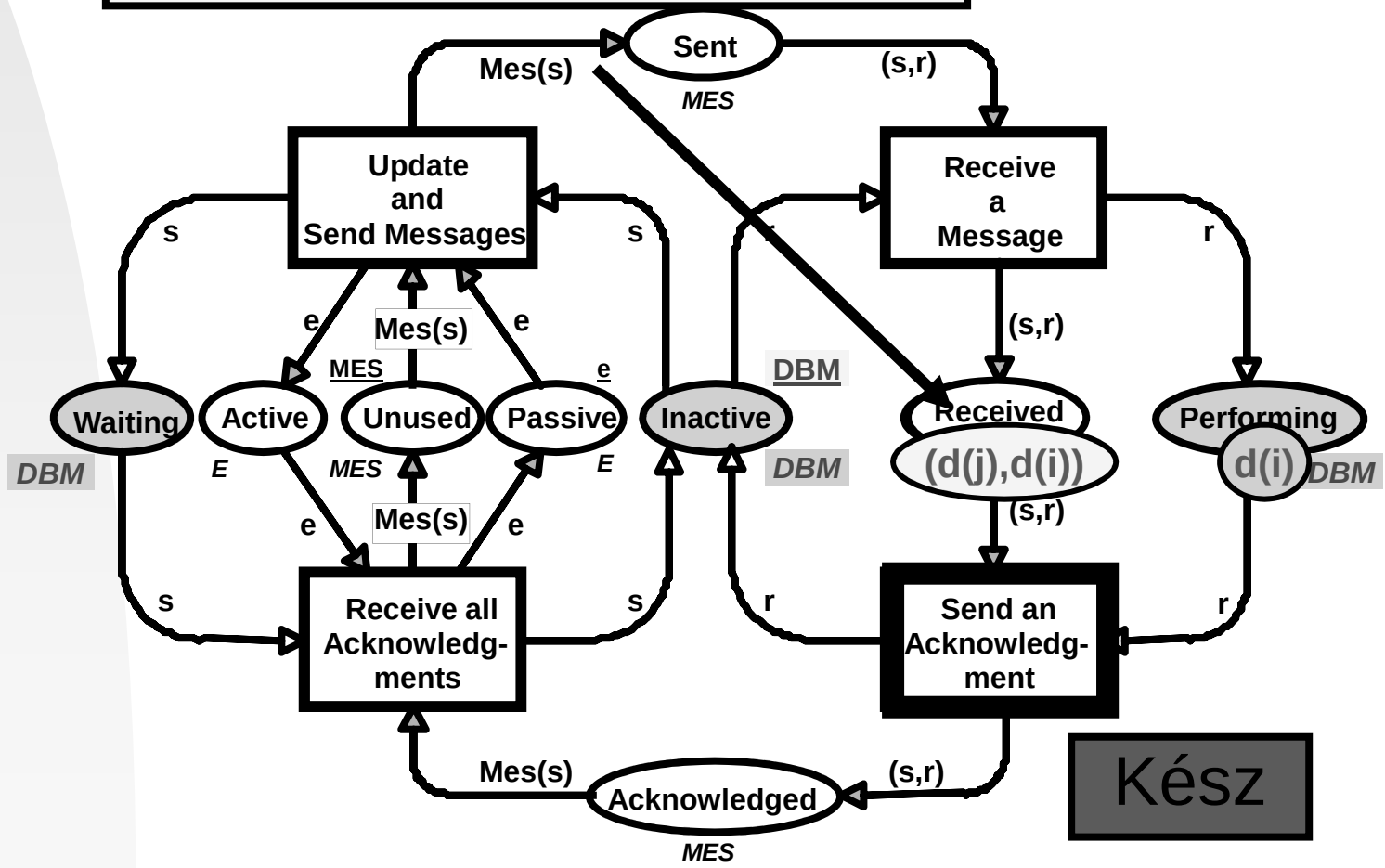
M(Waiting) + M (Inactive) + M(Performing) = DBM

Minden adatkezelő:



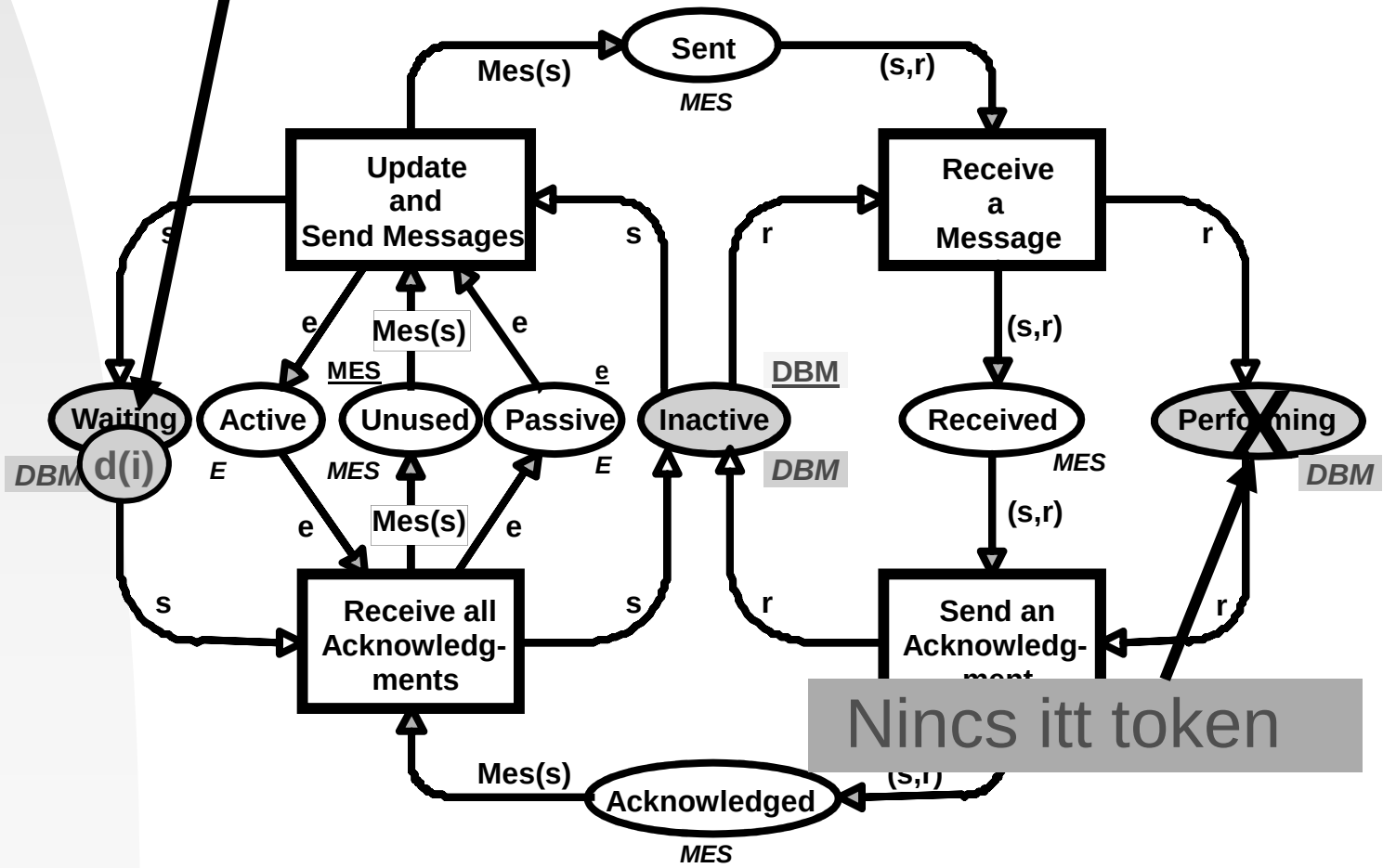
Rec(M(Received)) = M(Performing)

Ez az üzenet buffer:
Received with d(i)



Kész

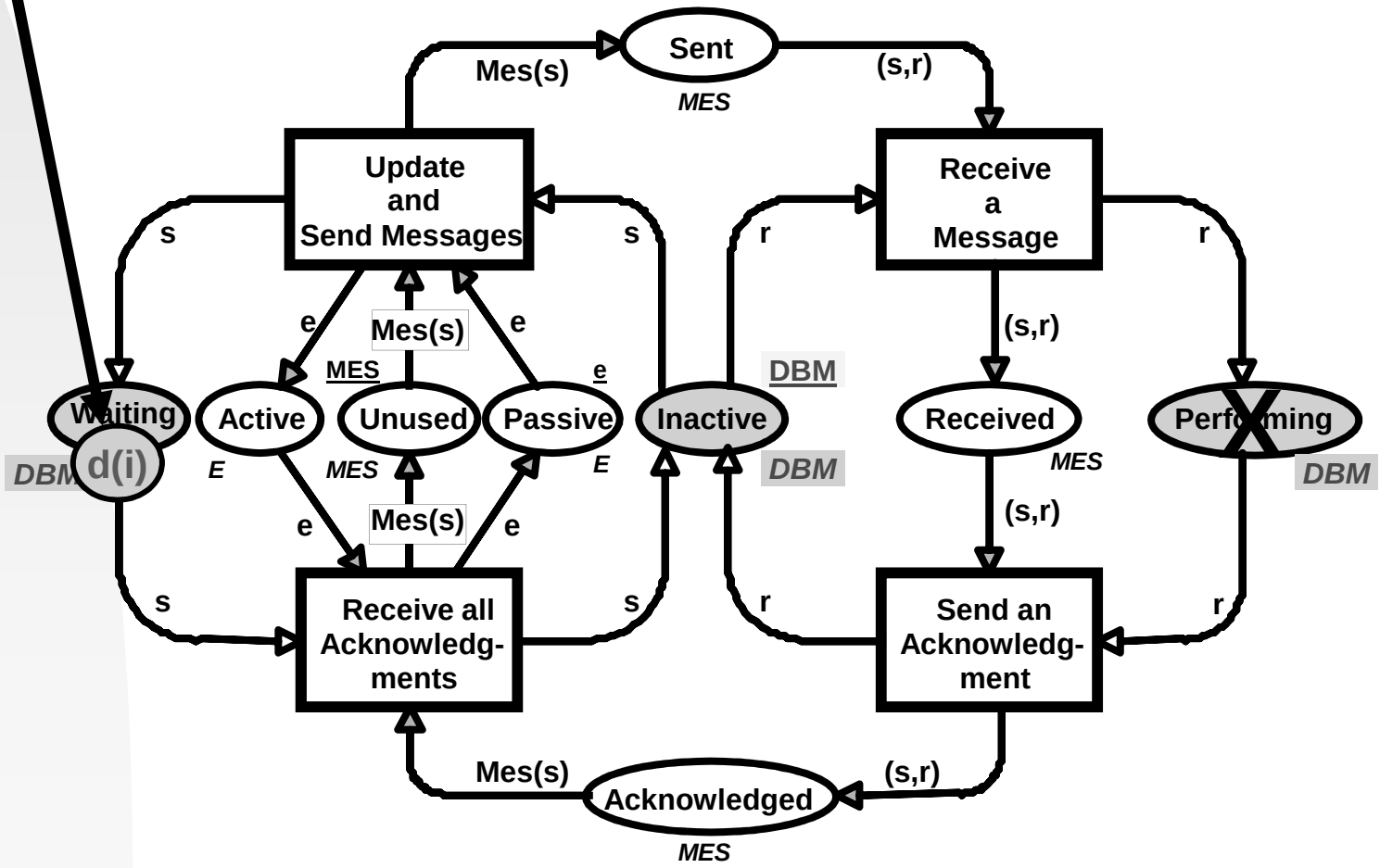
Most azt tesszük fel, hogy
legalább egy kezelő VÁR



Nincs itt token

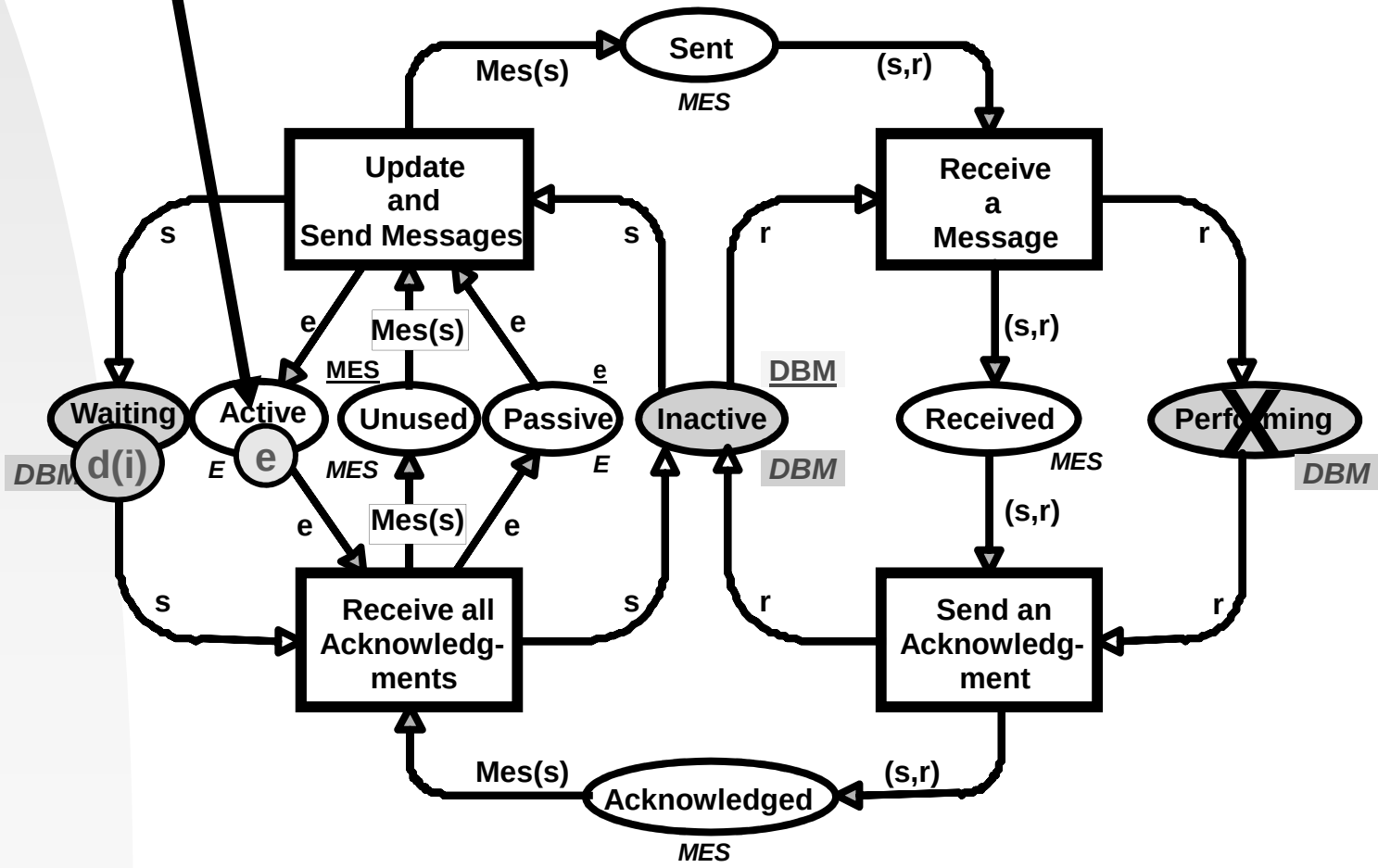
Ign(Waiting) + Passive = E

Pontosan egy token várakozik



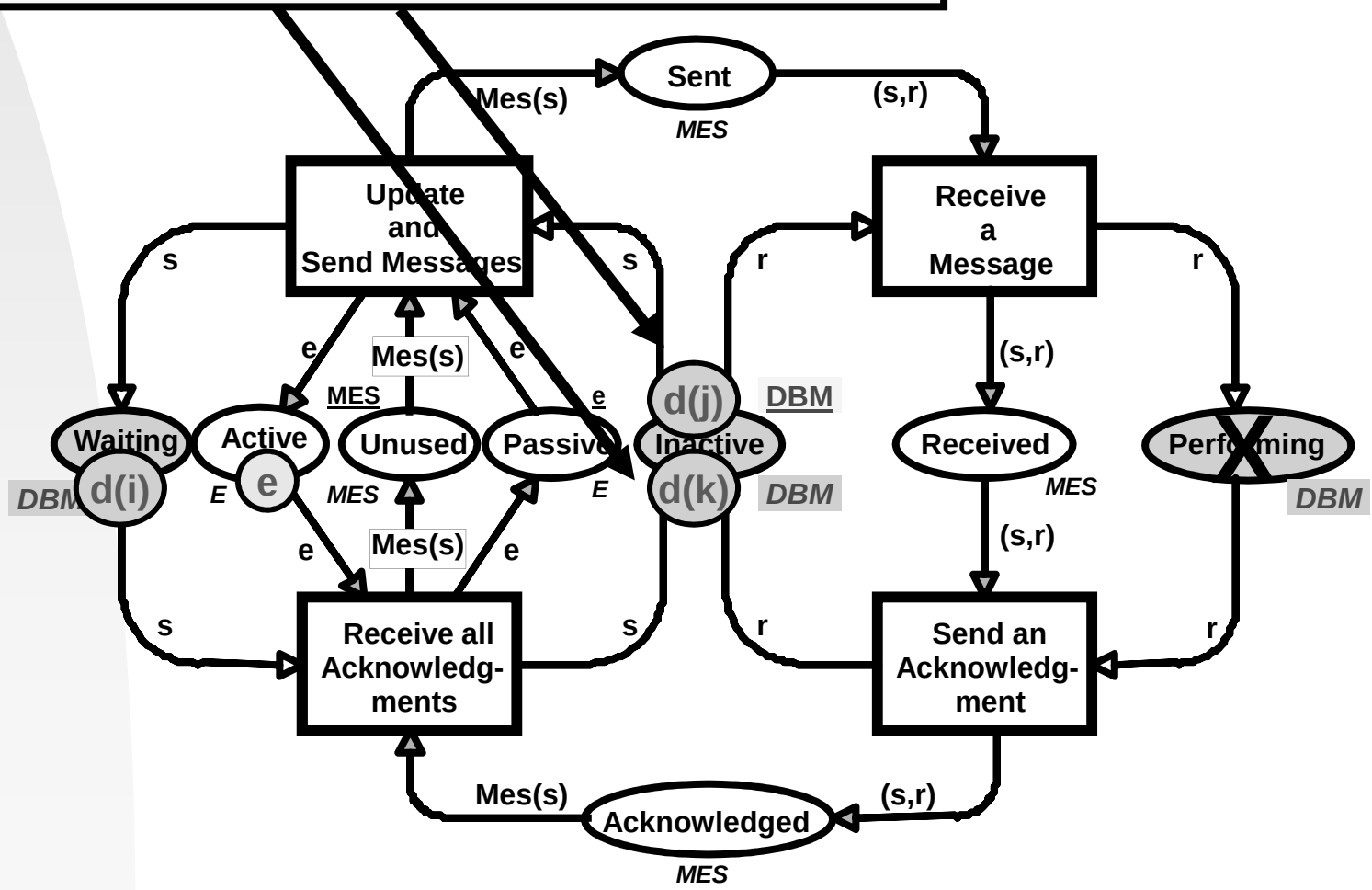
Ign(Waiting) = Active

Pontosan egy token aktív



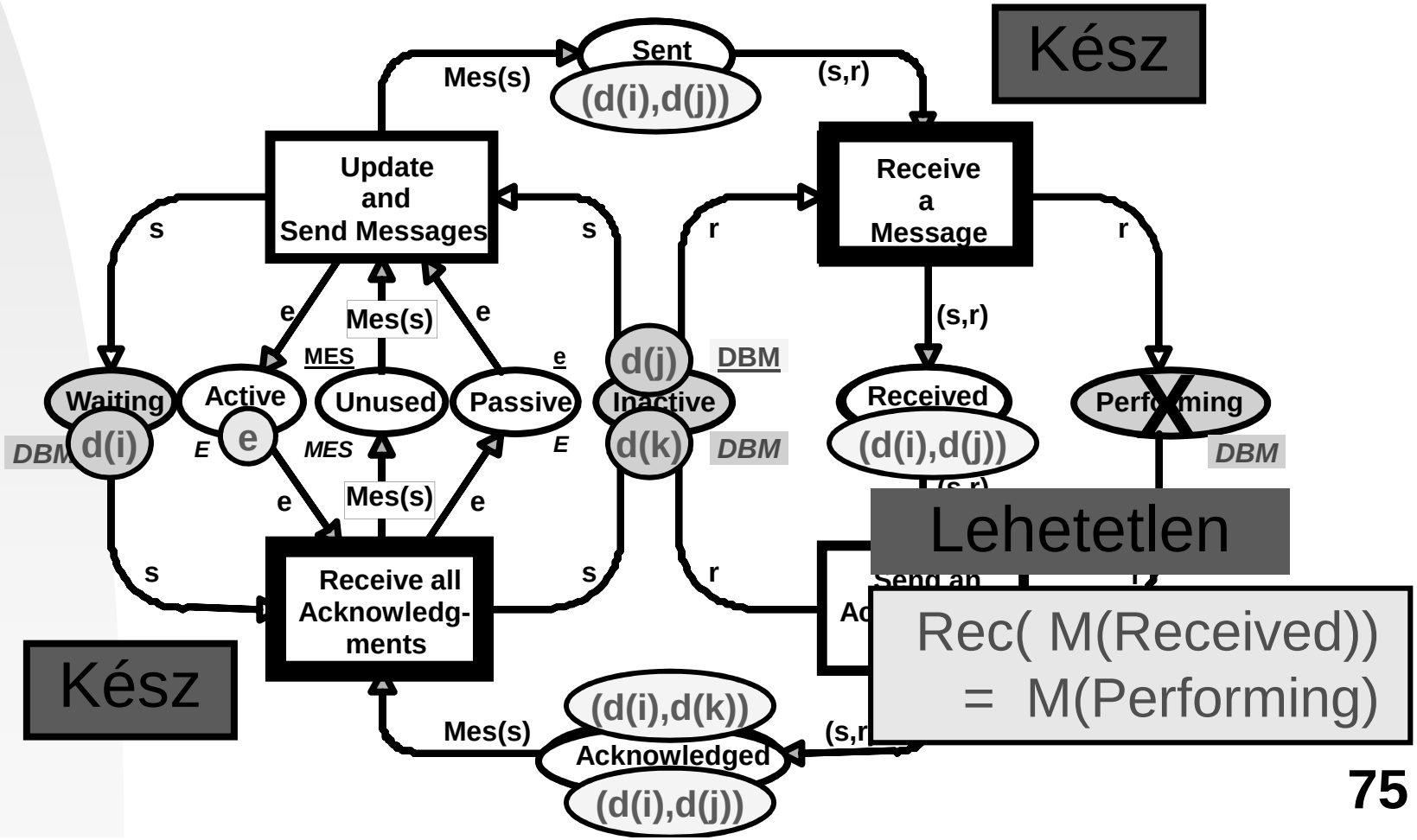
M(Waiting) + M (Inactive) + M(Performing) = DBM

Nincs több aktív adatbázis kezelő



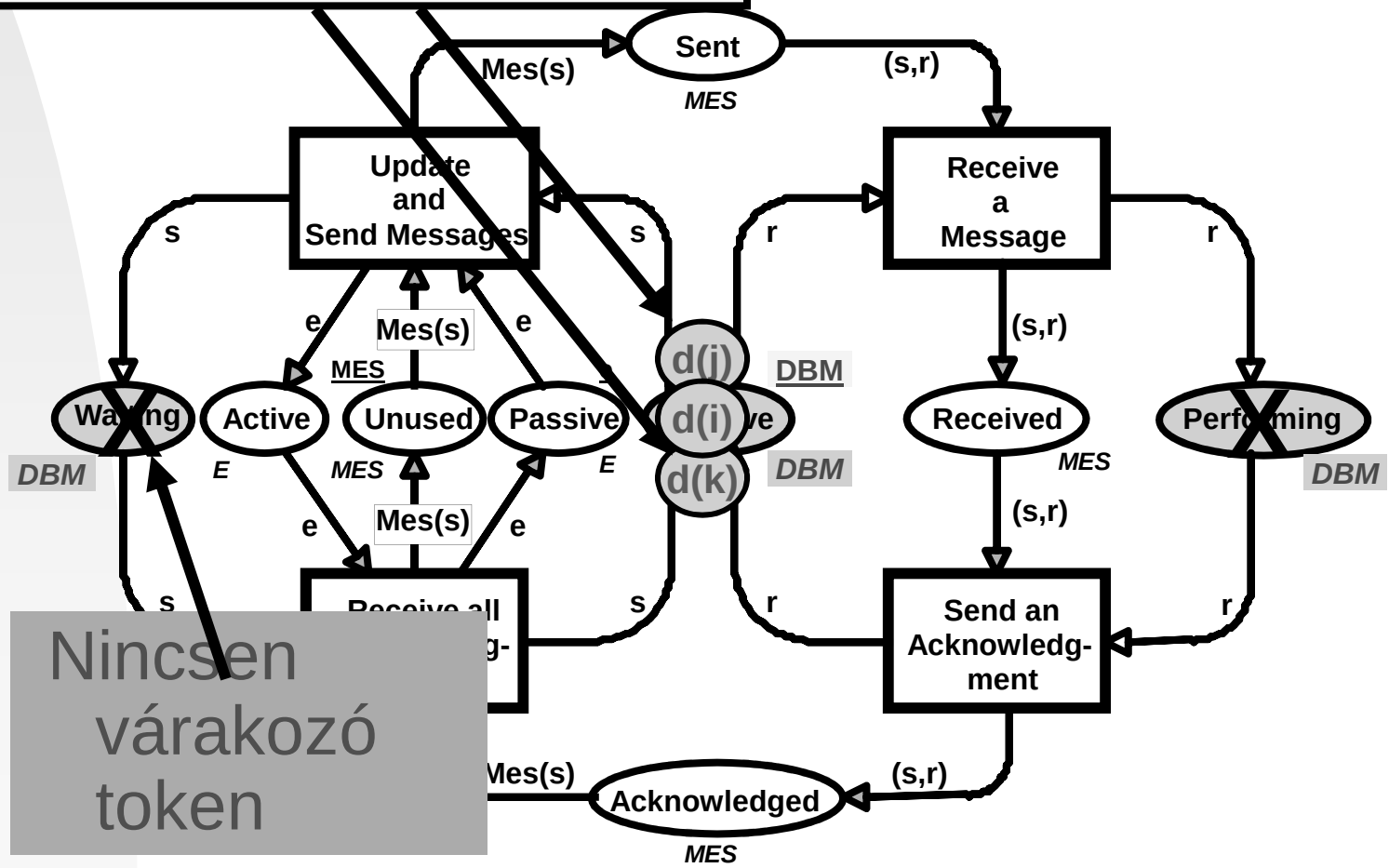
Mes(Waiting) = M(Sent) + M (Received) + M(Acknowledged)

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



M(Waiting) + M (Inactive) + M(Performing) = DBM

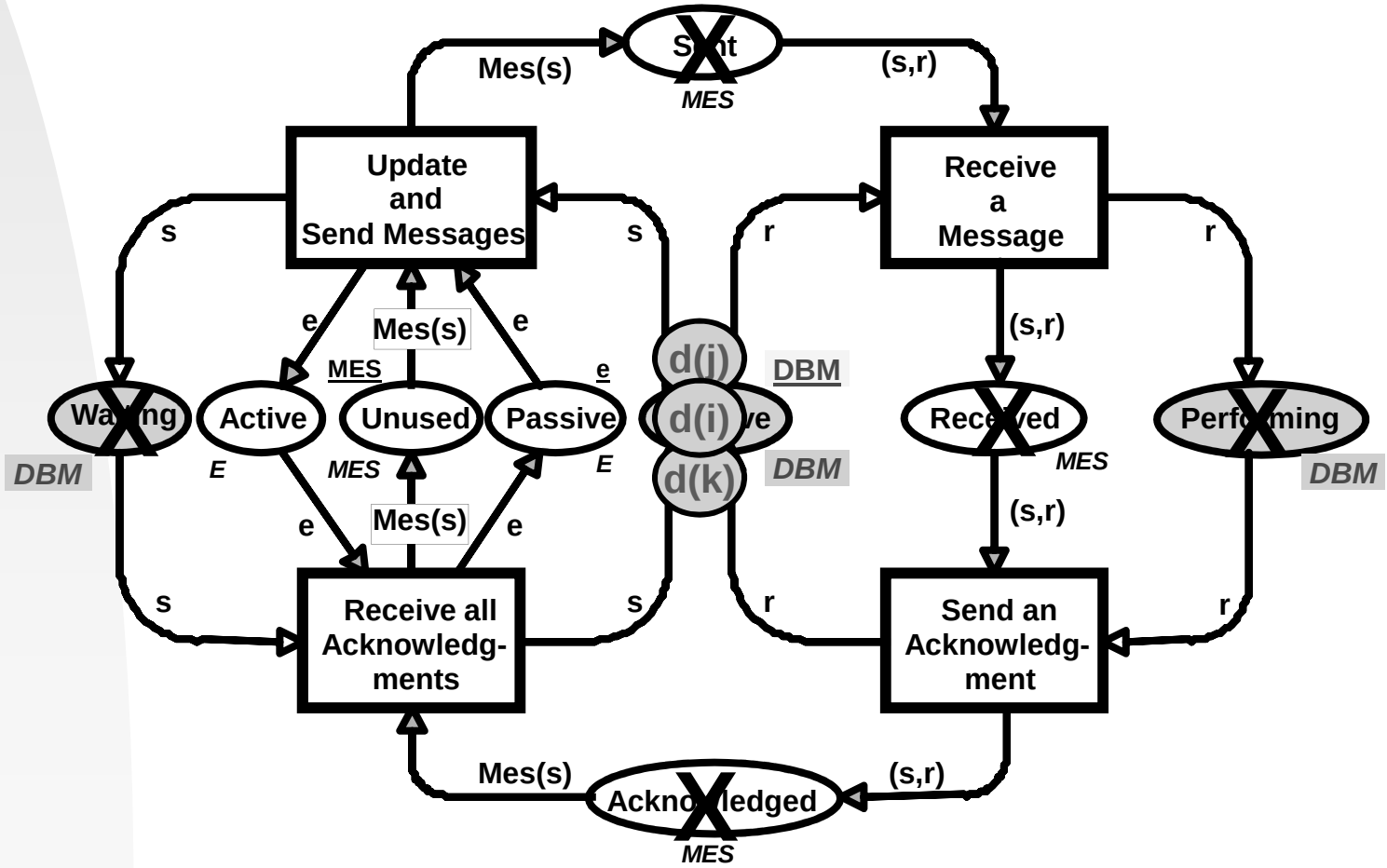
Nincsen aktív adatbázis kezelő



Nincsen várakozó token

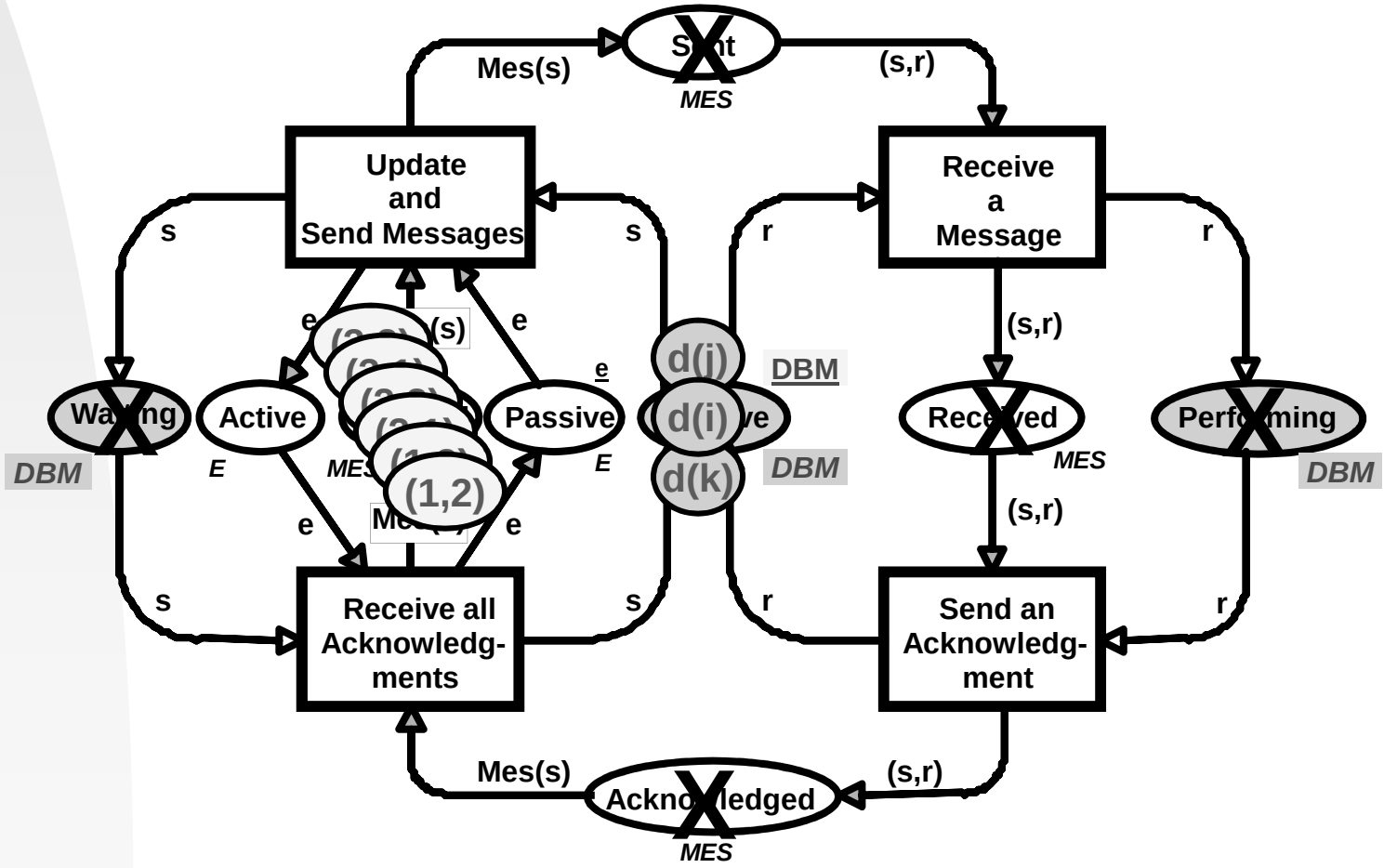
Mes(Waiting) = M(Sent) + M (Received) + M(Acknowledged)

Nincsen elküldött, várakozó és nyugta token



M(Unused) + M(Sent) + M (Received) + M(Acknowl) = MES

Az üzenetbufferek üresek:

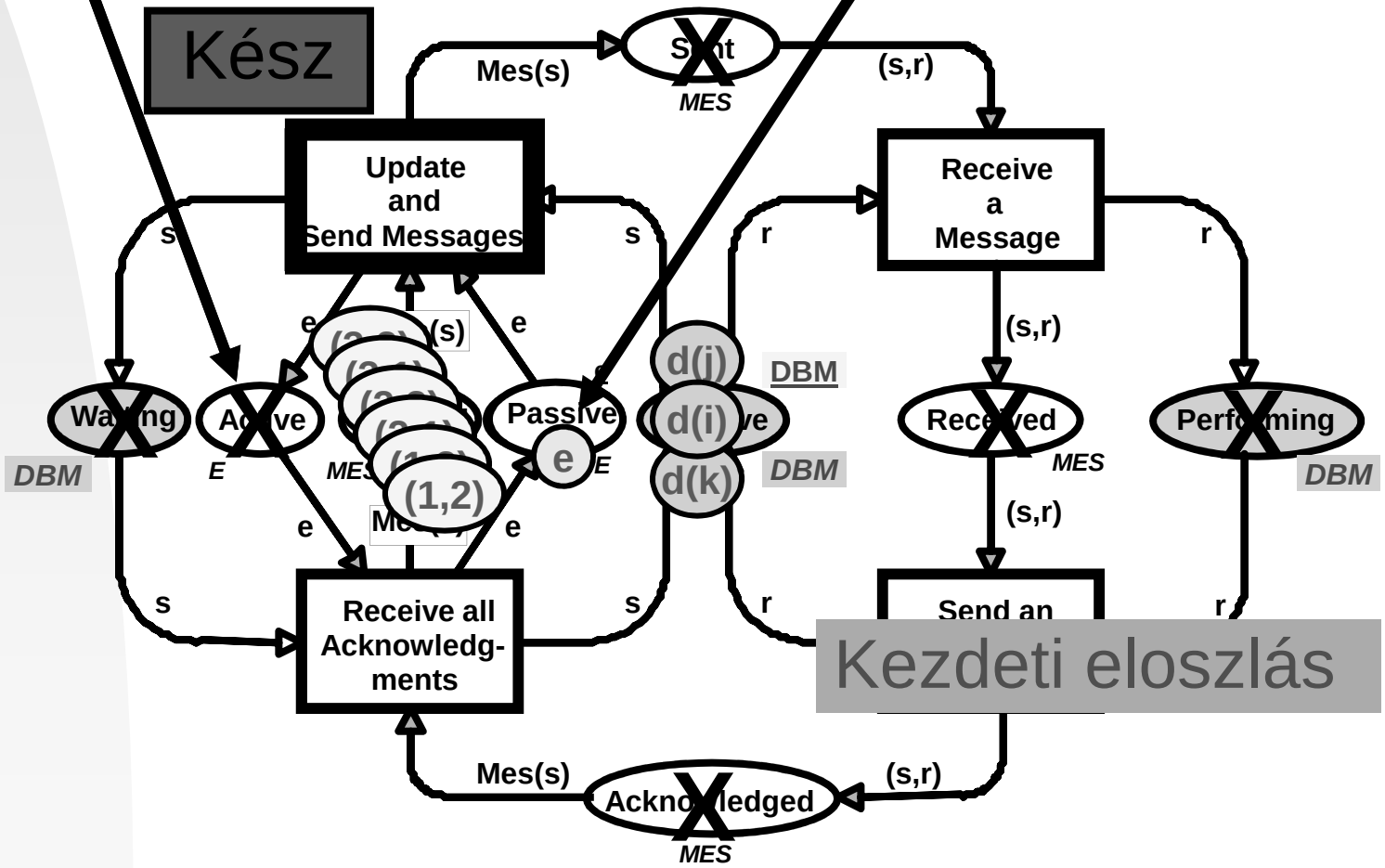


Ign(Waiting) = Active

Active + Passive = E

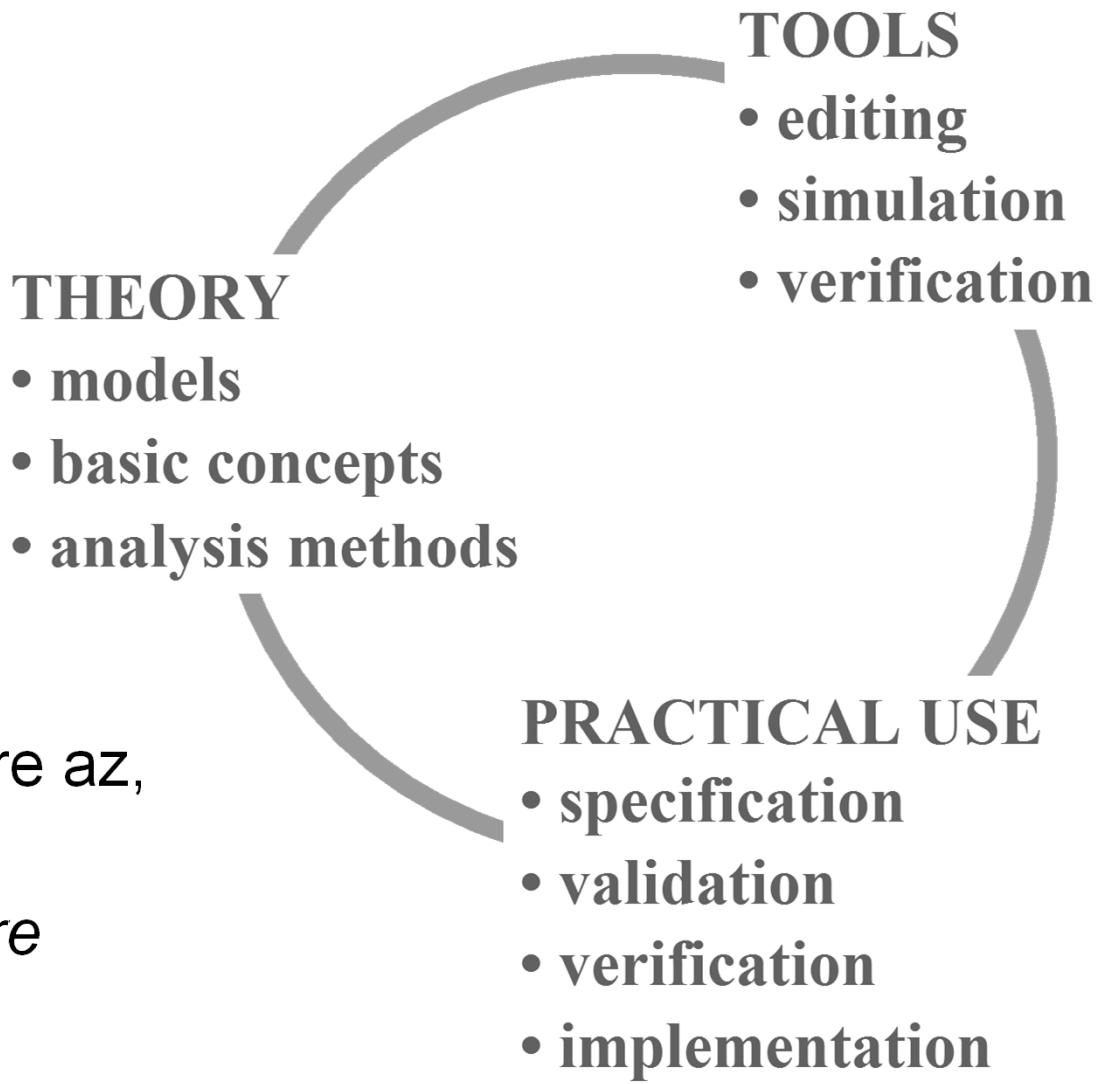
Nincsen aktív token

Egy e-token Passive



Kezdeti eloszlás

Konklúzió



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