

Színezett Petri-háló

© Kurt Jensen (részlet)

Fordította: Erdélyi Árpád

Department of Computer Science
Aarhus Egyetem, Dánia

kjensen@daimi.au.dk

THEORY

- models
- basic concepts
- analysis methods

TOOLS

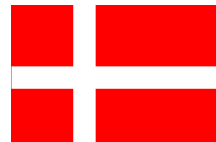
- editing
- simulation
- verification

PRACTICAL USE

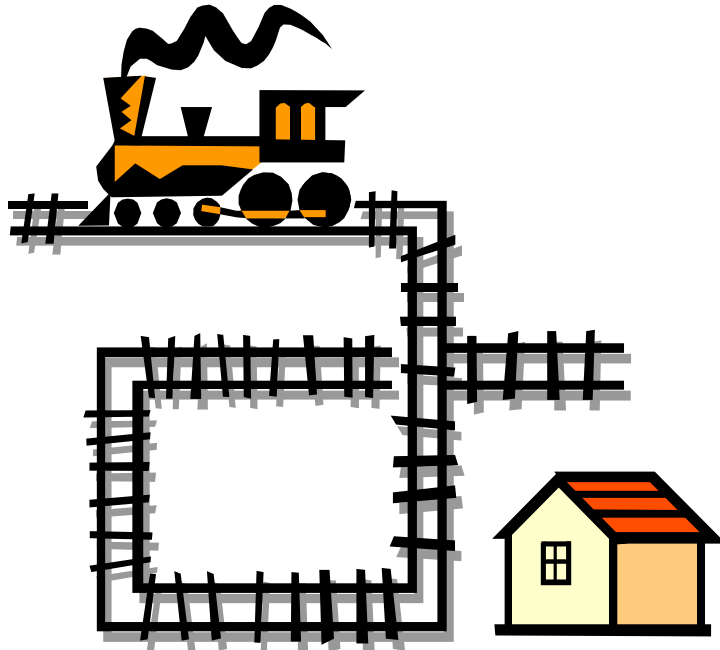
- specification
- validation
- verification
- implementation

Mi is az a Színezett Petri háló?

- ◆ *Modellező nyelv* rendszerekhez, ahol szinkronizáció, kommunikáció, és erőforrás megosztás fontos.
- ◆ Egyben *Petri Háló* és *Programozási Nyelv*.
 - *A vezérlési struktúrát, szinkronizációt, kommunikációt*, és az *erőforrás megosztást* a Petri háló írja le.
 - *Az adatokért* és az *adat kezelésért* a (Standard ML) funkcionális programozási nyelv a felelős.
- ◆ CPN modelleknél *validálás* alatt *szimulációt* értünk, *verifikálás* alatt pedig *állapotteret* és *állapot-invariánsokat*.
- ◆ *Dániában* az *Aarhus egyetemen* már közel 25 éve kutatják a Színezett Petri hálókat.



Miért készítünk modelleket?



- ◆ A *modellezés* célja:
 - *Új dolgokat ismerhessünk meg* a rendszerről.
 - A rendszerrel szemben támasztott *követelmények* teljesülésének ellenőrzése.

- ◆ CPN modellek *dinamikusak*, azaz *végrehajthatóak*:
 - *Futtathatóak* számítógépen.
 - *Forgatókönyvek* lejátszására van lehetőség.

Magas-szintű Petri-háló

- ◆ A *CP-háló* és a *normál Petri-háló* közti kapcsolat hasonló a *magas-szintű programozási nyelvek* és az *assembly kód* kapcsolatához.
 - Elméletben a két szint *kifejezőereje* azonos.
 - Gyakorlatban a magas szintű nyelvek sokkal nagyobb *modellező erővel* rendelkeznek, mivel jobbak a struktúra-leíró képességük, pl. vannak modulok, típusok.
 - Több más magas-szintű Petri háló létezik, de a *Színezett Petri-háló* a legelterjedtebb a *gyakorlatban*.

CPN-eket nagy rendszerekhez használják

- ◆ Egy CPN modell számos *alhálóból* áll.
 - Hasonlóak a *modulokhoz*.
 - Jól-definínált *interfészek* és tiszta *szemantika*.
- ◆ A CPN tipikus *ipari alkalmazása*:
 - 10-200 alháló.
 - 50-1000 hely és tranzíció.
 - 10-200 típus.
- ◆ Ilyen méretű ipari alkalmazások *elképzelhetetlenek* ezek nélkül:
 - Adattípusok és token értékek.
 - Modulok.
 - Modellező eszköz támogatás.

Felhasználási területek

Protokollok és Hálózatok

- ◆ 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

Szoftver

- ◆ 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

Vezérlő rendszerek

- ◆ 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

Hardver

- ◆ VLSI Chip in the US
- ◆ Arbiter Cascade at Meta Software Corp.

Katonai rendszerek

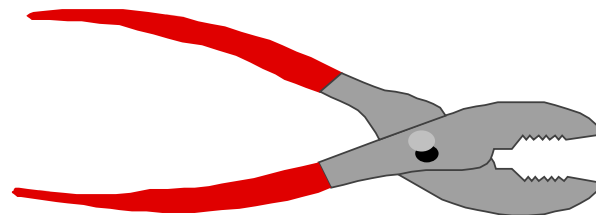
- ◆ Military Communications Gateway in Australia
- ◆ Influence Nets for the US Air Force
- ◆ Missile Simulator in Australia
- ◆ Naval Command and Control System in Canada

Egyéb

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

Modellező eszközök

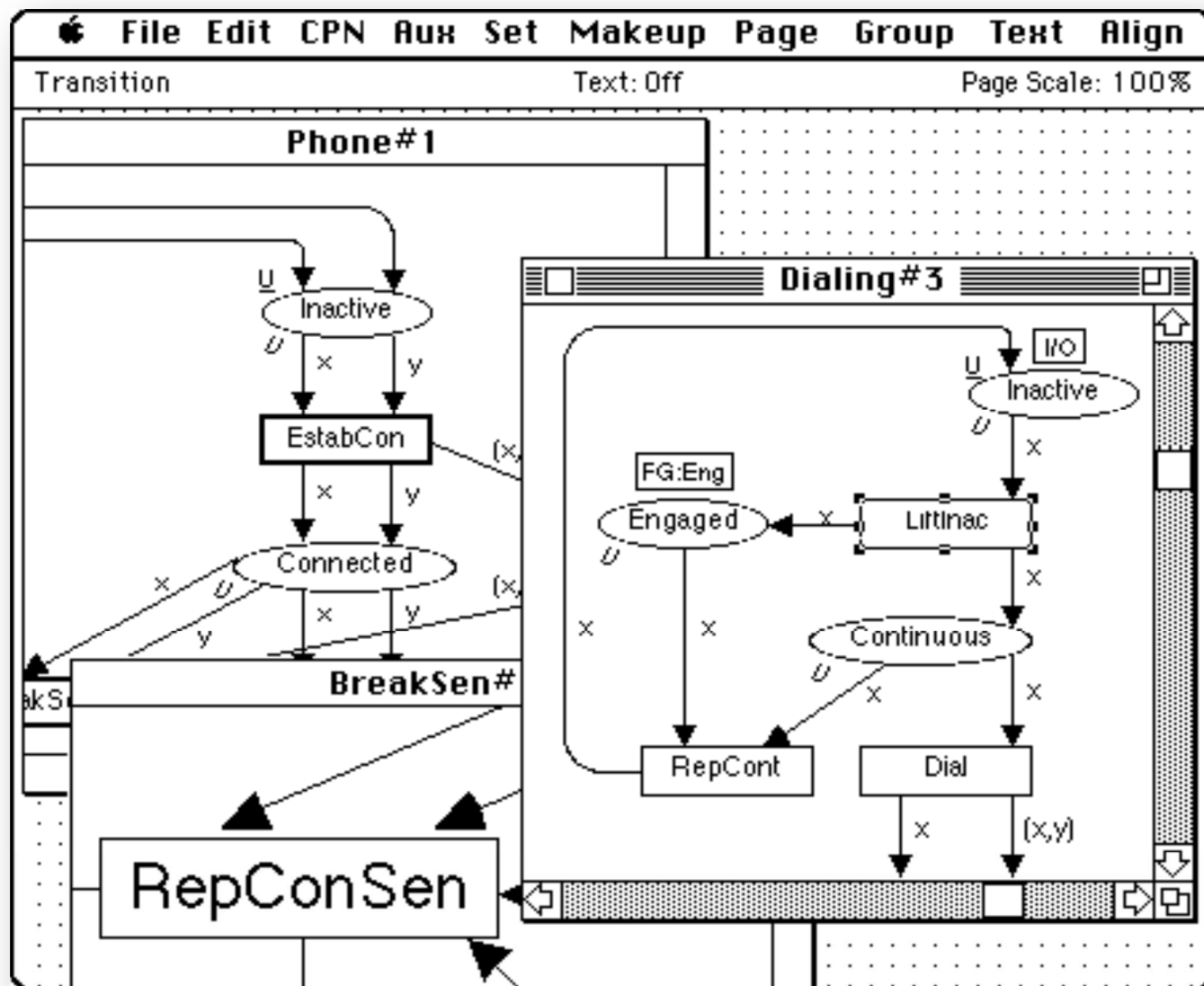
- ◆ *Design/CPN* eszközt a 80'as évek végén és a 90'as évek elején fejlesztették ki.
 - A maga idejében a legelterjedtebb Petri háló csomag volt.
 - *750 különböző szervezet 50 országban használta* – ezek között *200 kereskedelmi cég van.*
- ◆ *CPN Tools* már egy második generációs eszköz a Színezett 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)

CPN Tools (Version 2.9.12, September 2010)

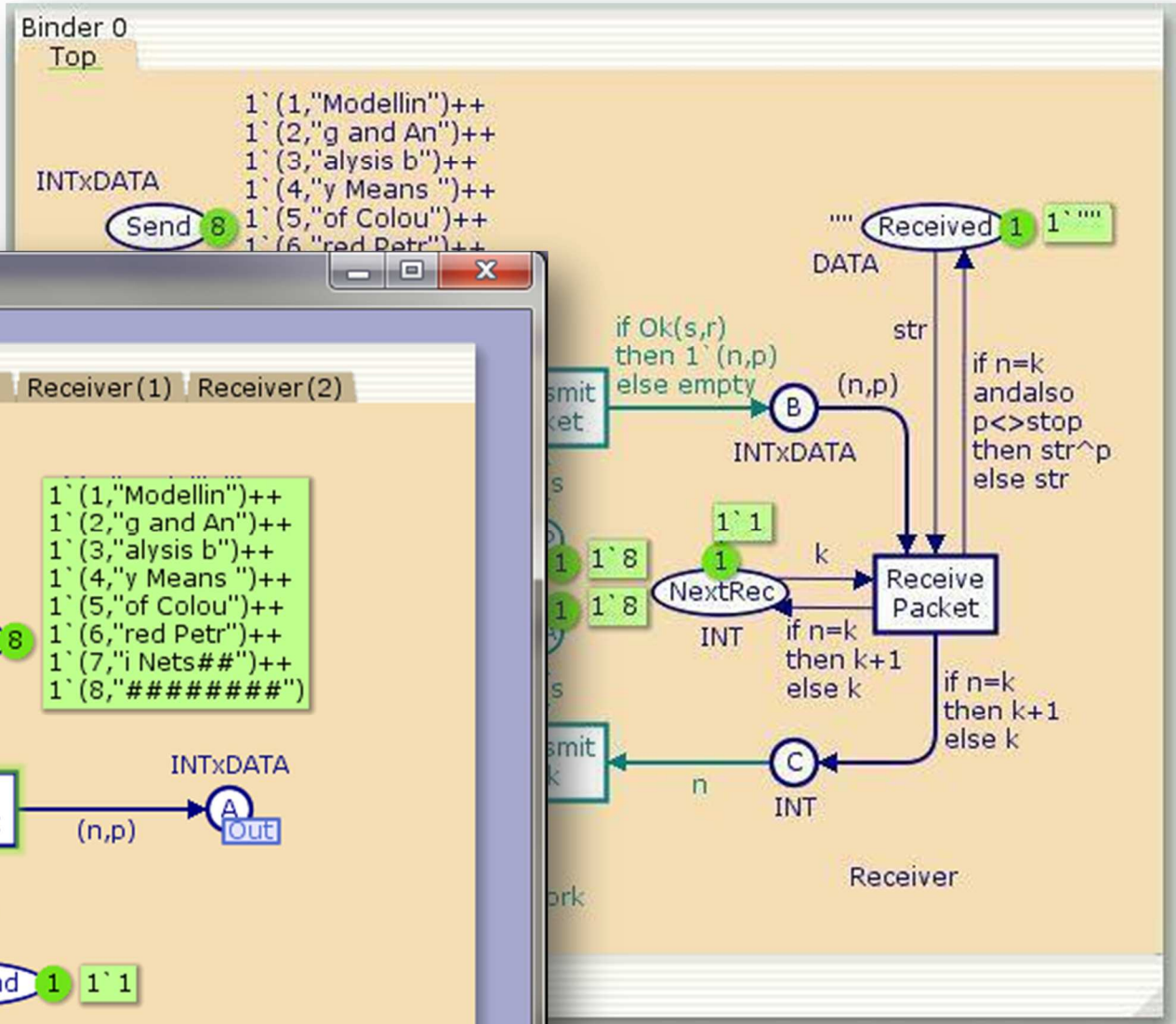
- Tool box
 - Auxiliary
 - Create
 - Hierarchy
 - Monitoring
 - Net
 - Simulation
 - State space
 - Style
 - View
- Help
- Options
- HierarchicalProtocol.cpn
 - Step: 0
 - Time: 0
 - Options
 - History
 - Declarations
 - colset INT
 - colset DATA
 - colset INTxDATA
 - colset INTxINT
 - var n k n1 n2
 - var p str
 - val stop
 - colset Ten0
 - colset Ten1
 - var s
 - var r r1 r2
 - fun Ok
 - fun imin
 - Monitors
 - Top
 - Sender

Binder 0
Top Sender Network Receiver(1) Receiver(2)

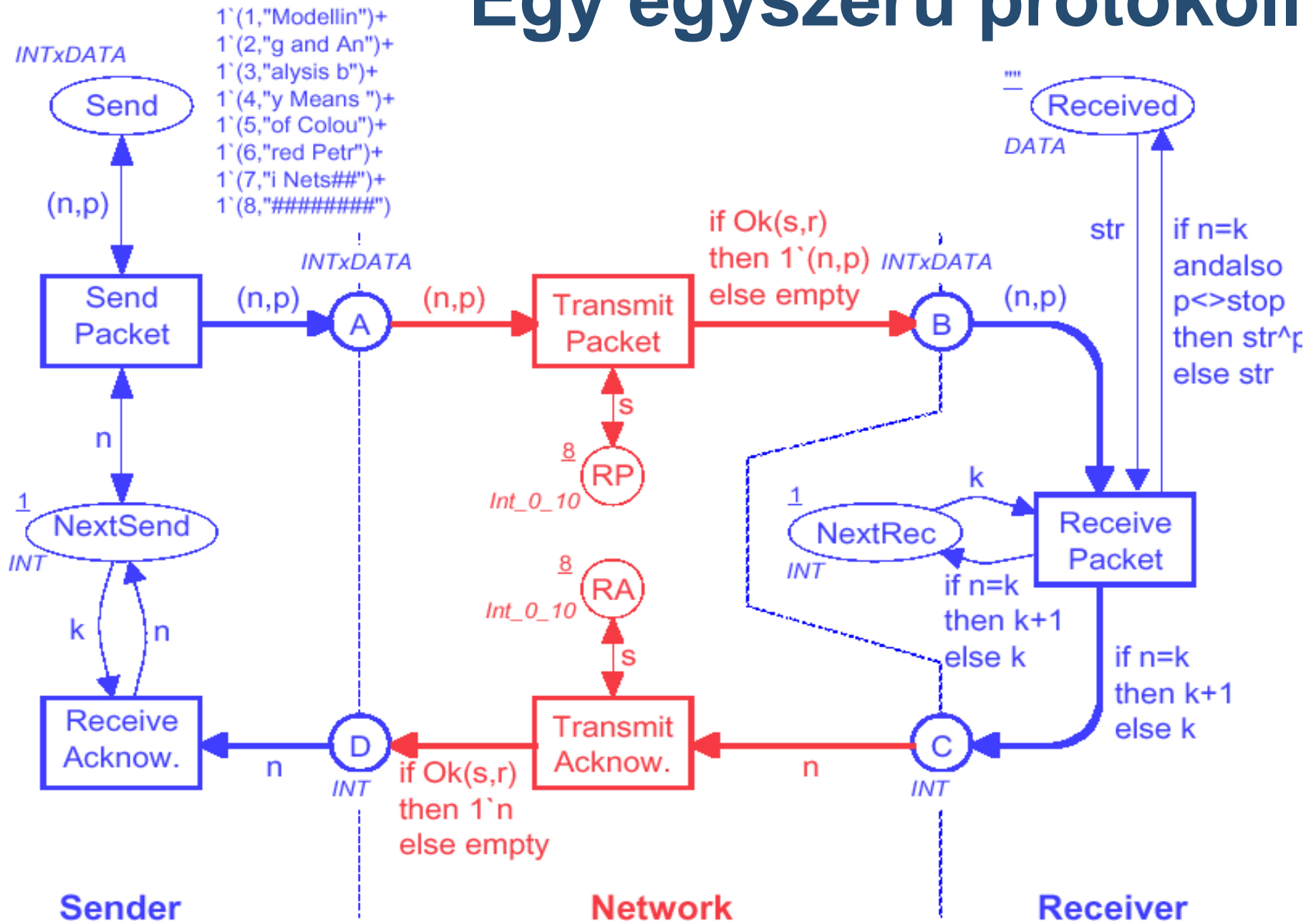
```

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,"#####")
    
```

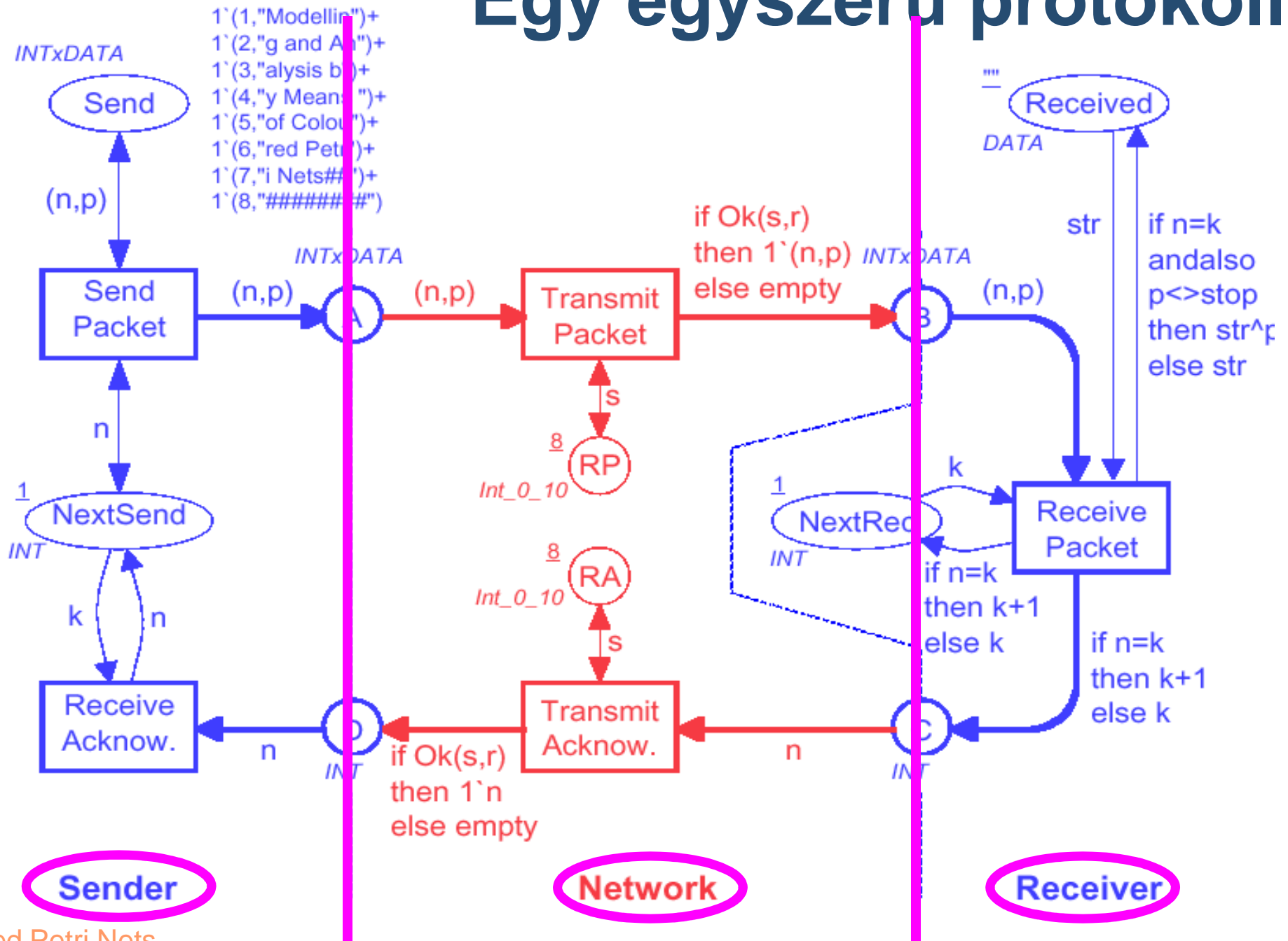
Simulation controls: None, Sim, Stop, Play, Step, Step, Step, 60, 500, 000, ML!



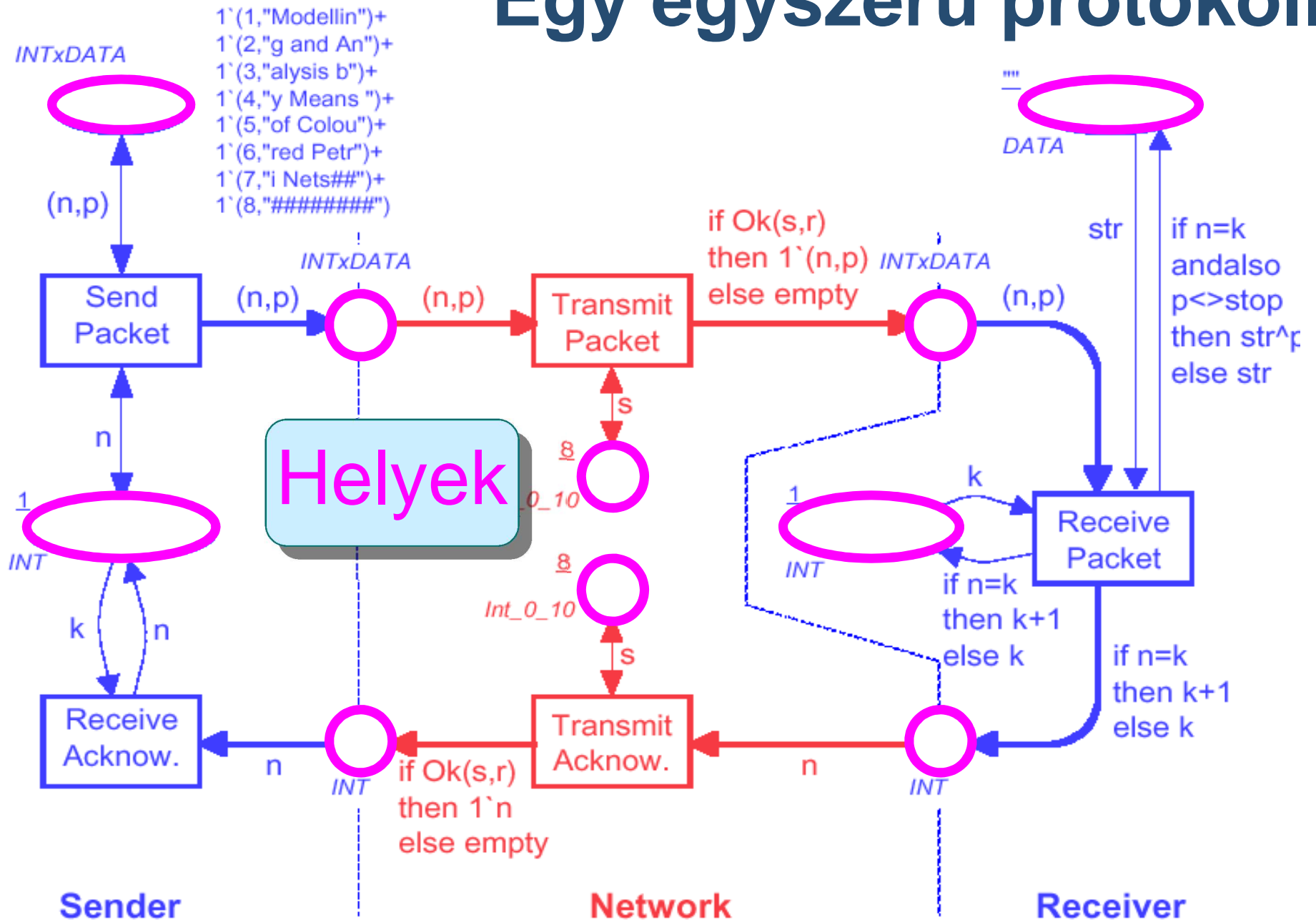
Egy egyszerű protokoll



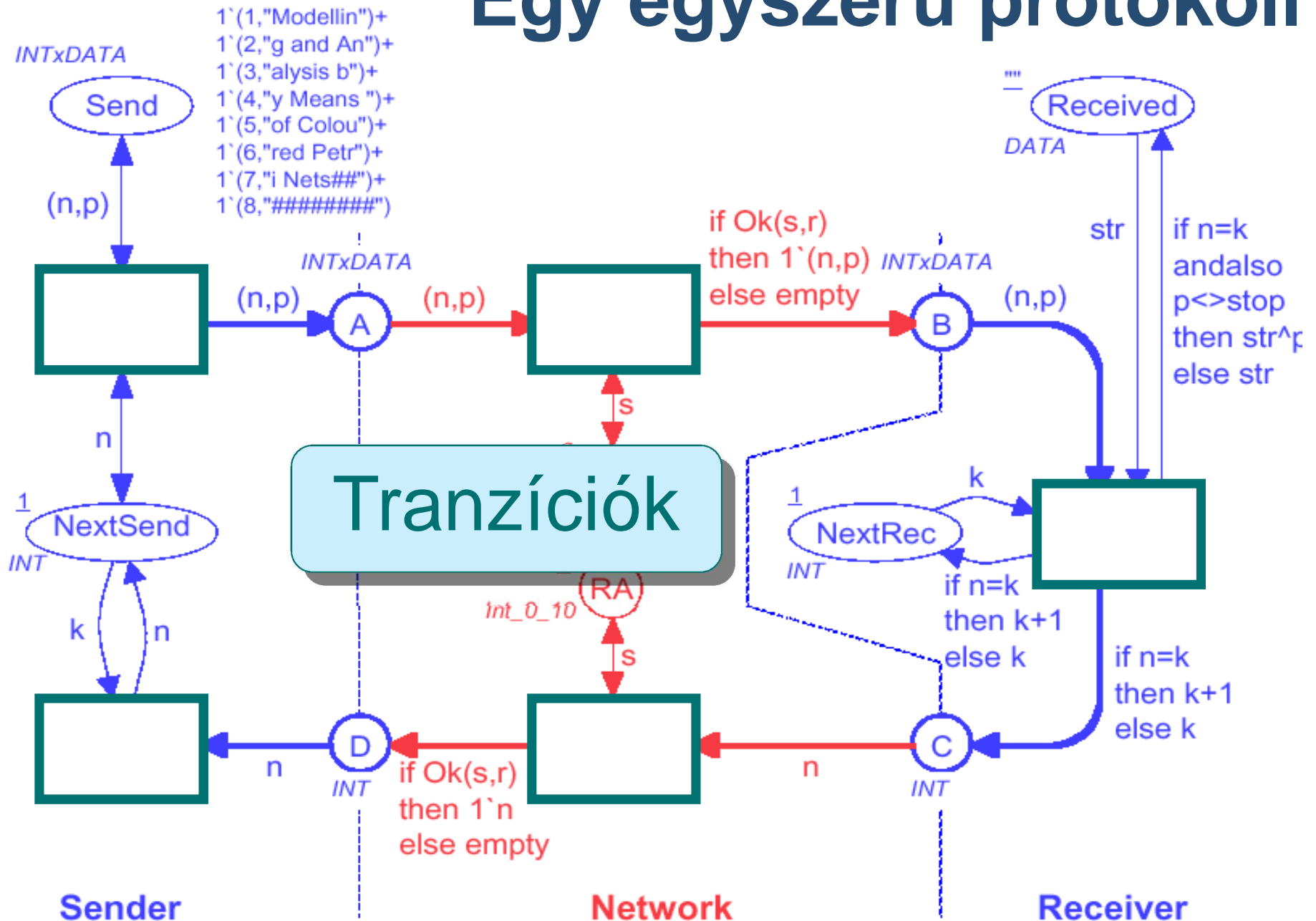
Egy egyszerű protokoll



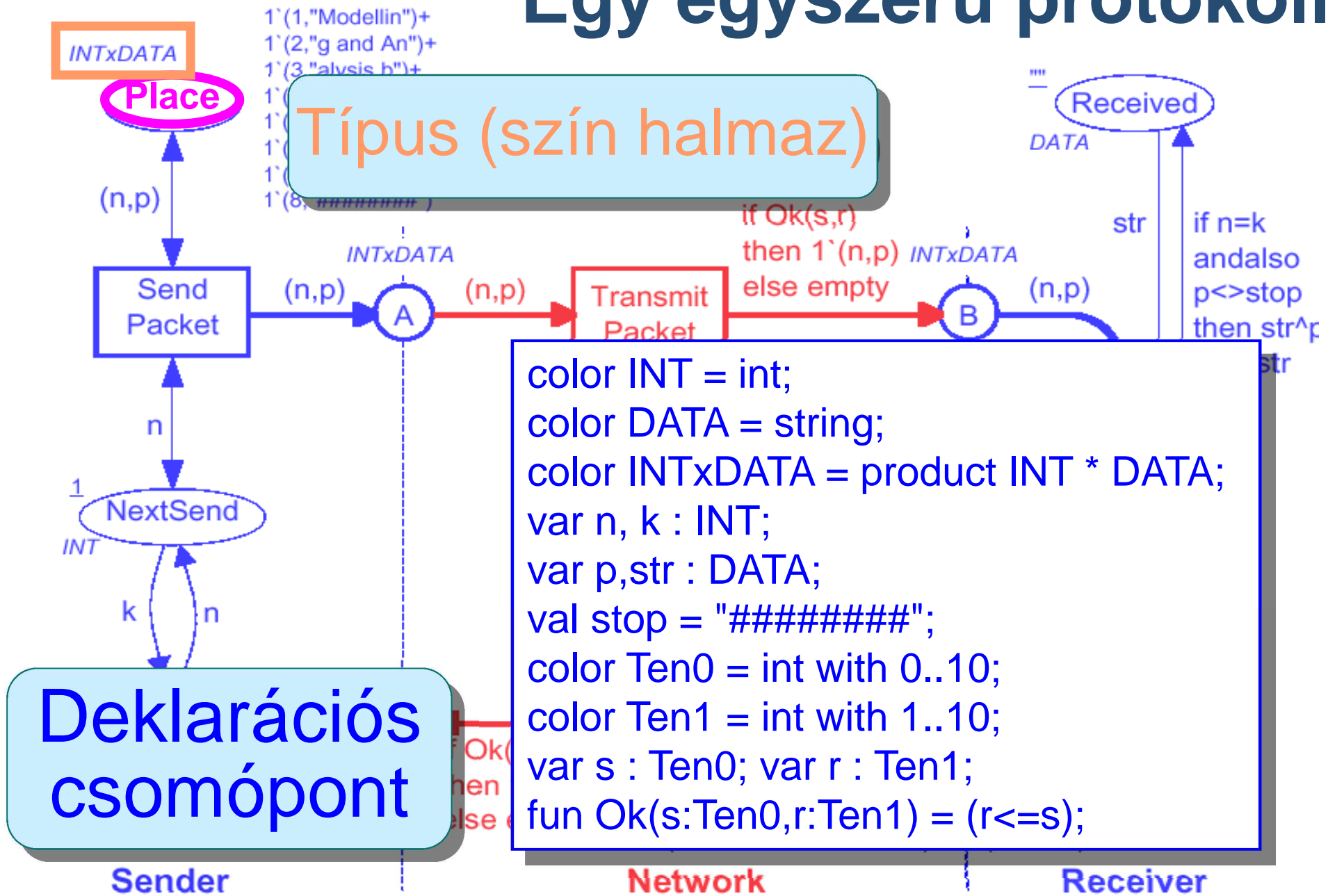
Egy egyszerű protokoll



Egy egyszerű protokoll

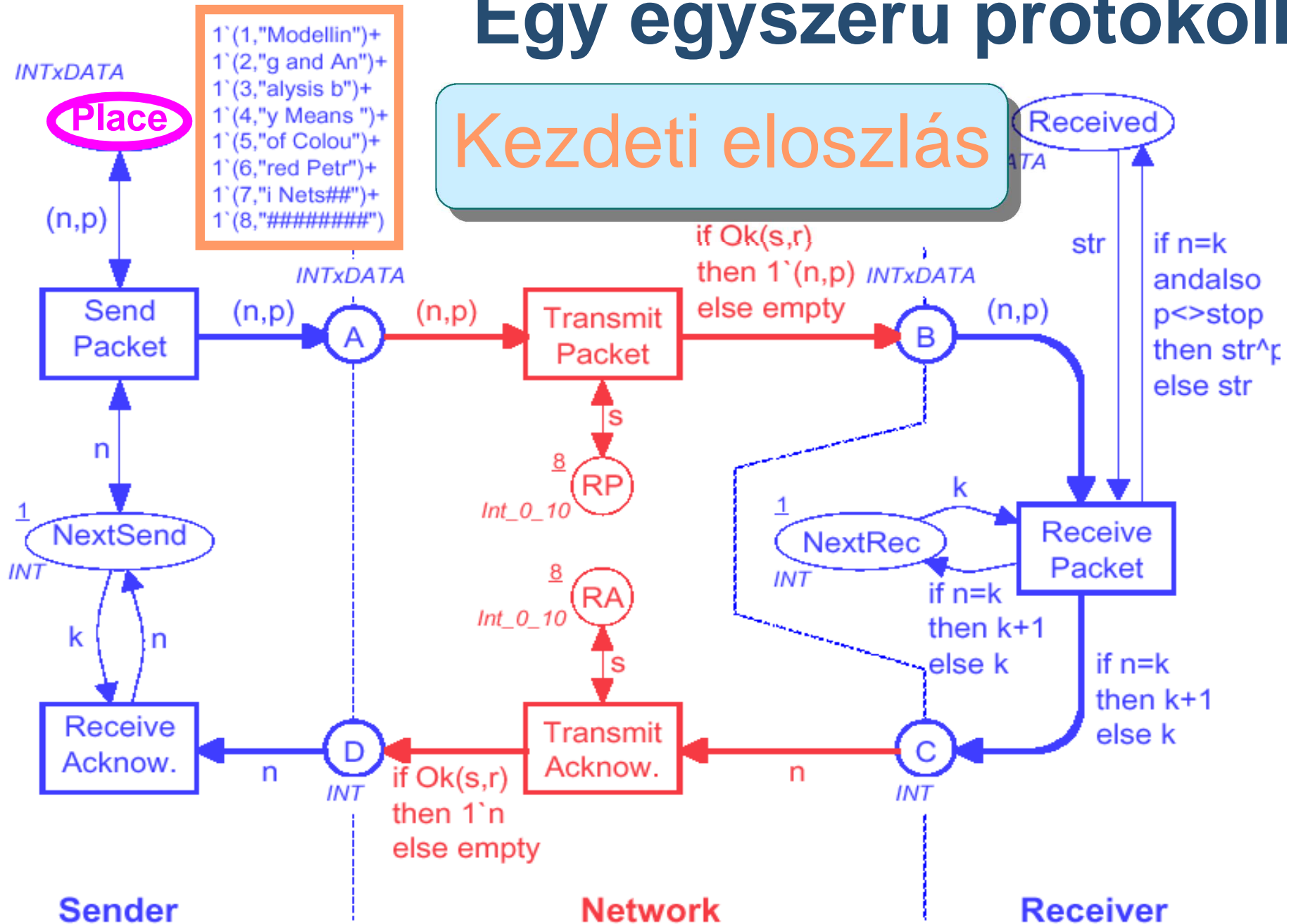


Egy egyszerű protokoll



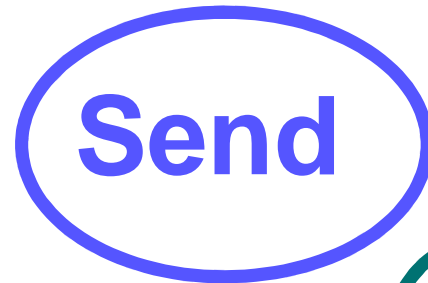
Egy egyszerű protokoll

Kezdeti eloszlás



Send jelölése

INTxDATA

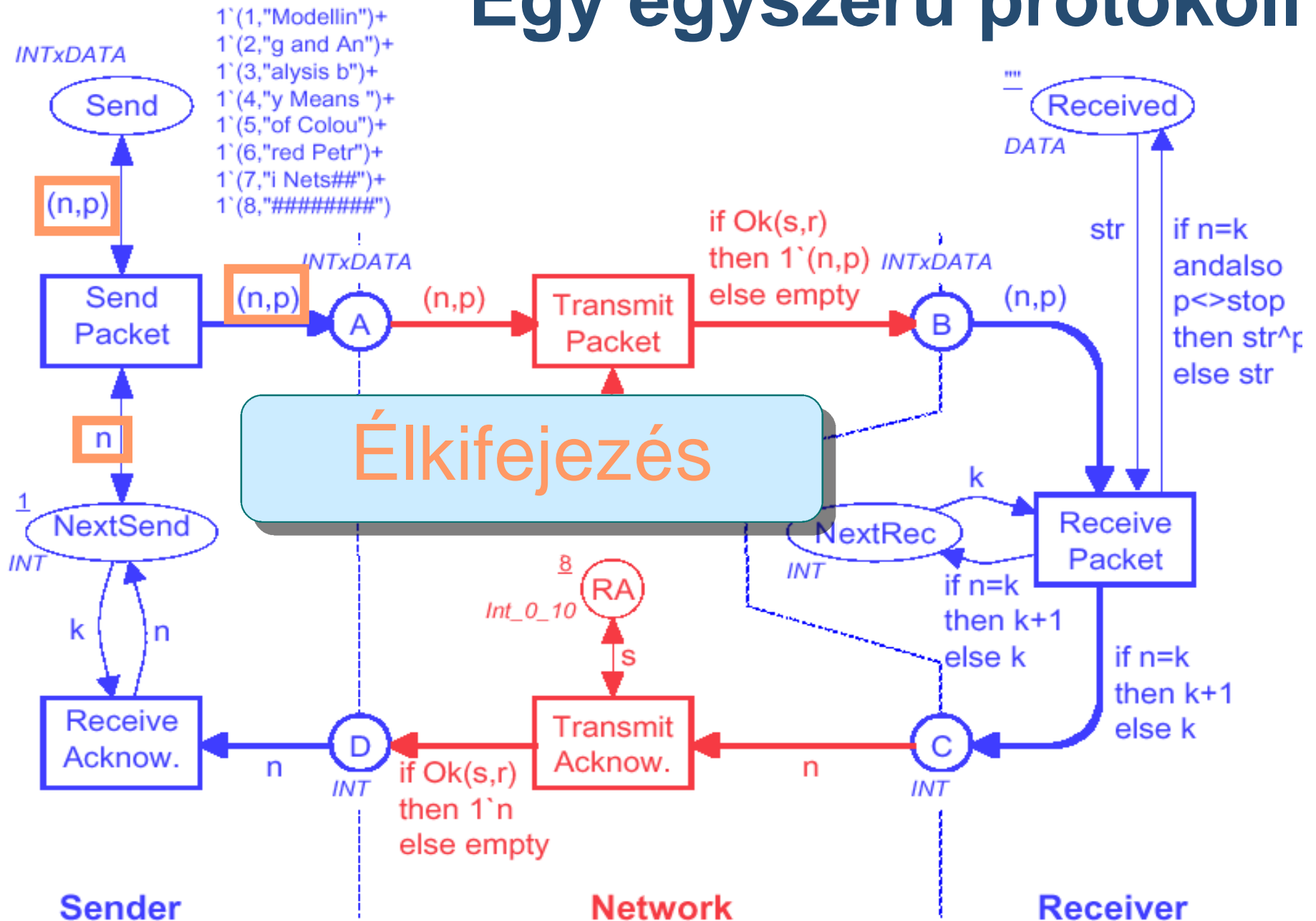


Tokenek száma

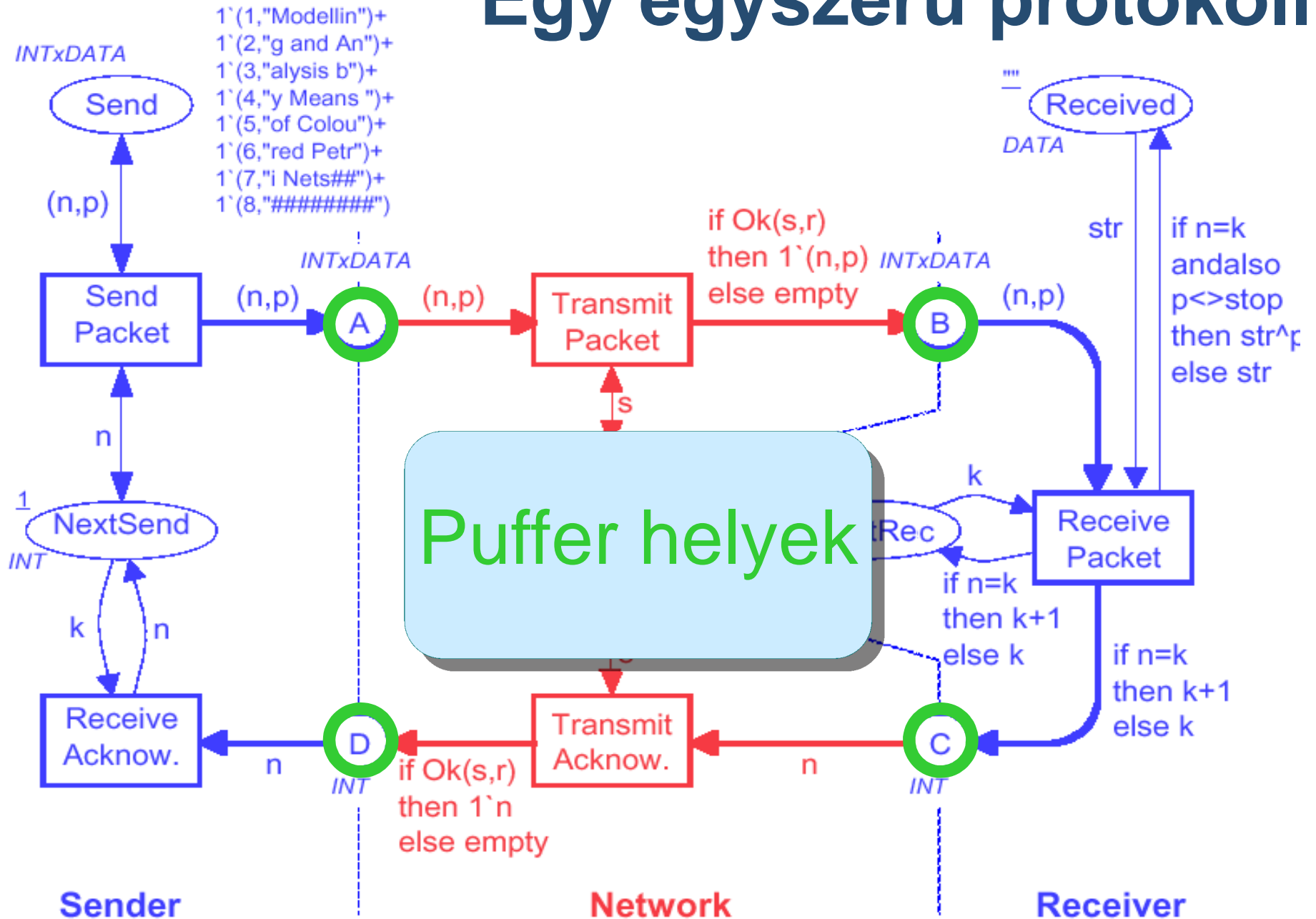
1 ` (1, "Modellin") +
1 ` (2, "g and An") +
1 ` (3, "alysis b") +
1 ` (4, "y Means ") +
1 ` (5, "of Colou") +
1 ` (6, "red Petr") +
1 ` (7, "i Nets##") +
1 ` (8, "#####")

Token színek
halmaza

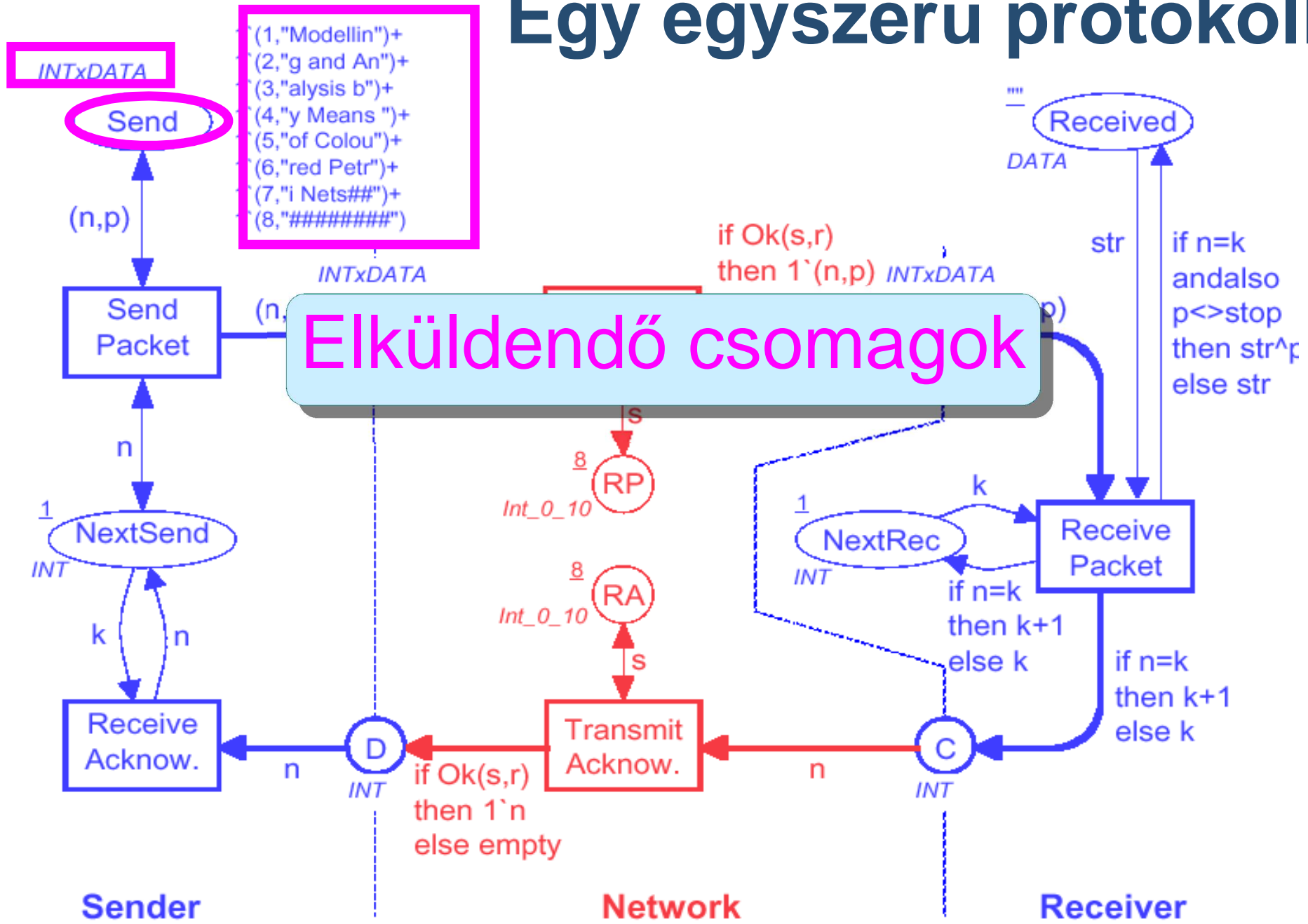
Egy egyszerű protokoll



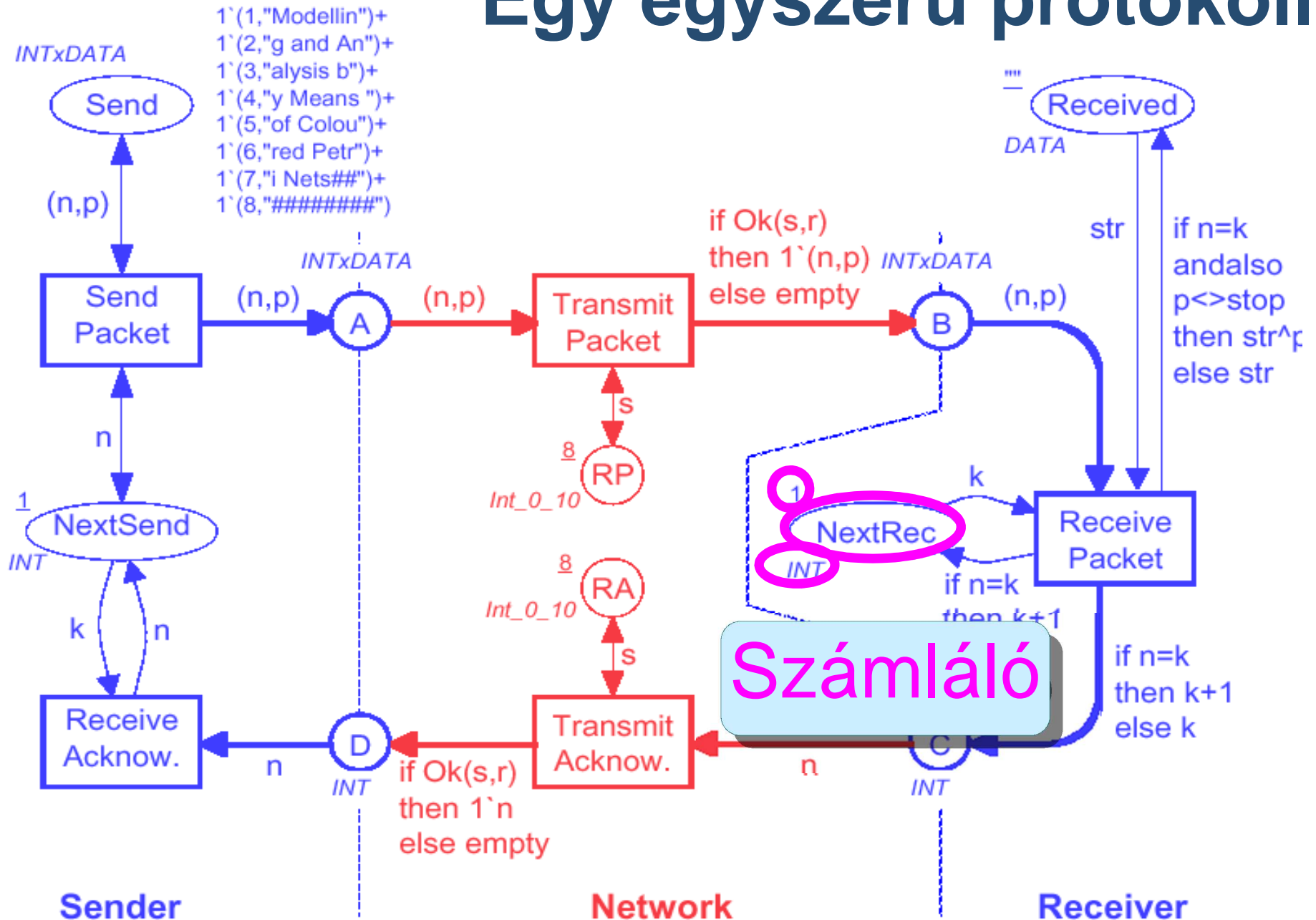
Egy egyszerű protokoll



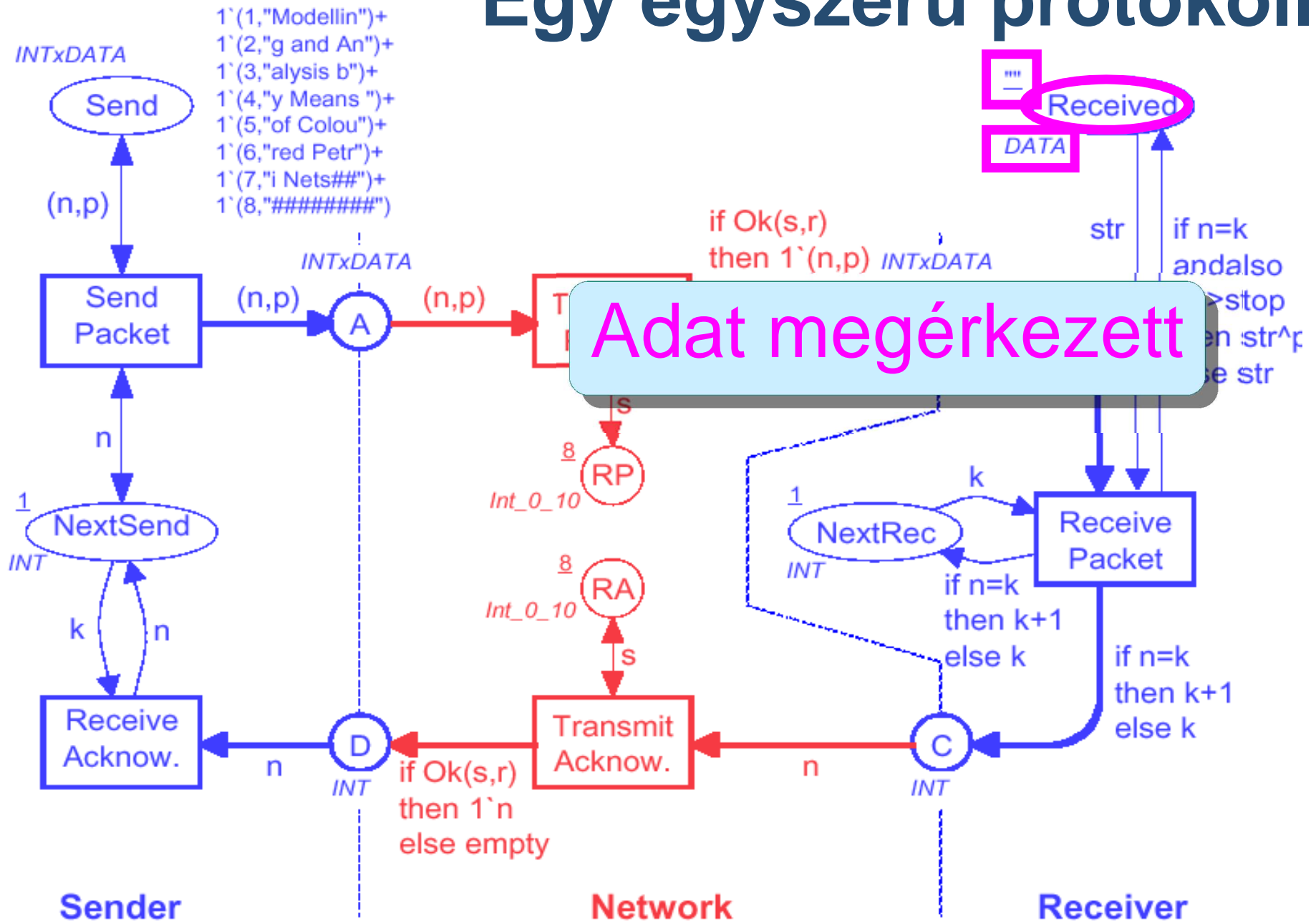
Egy egyszerű protokoll



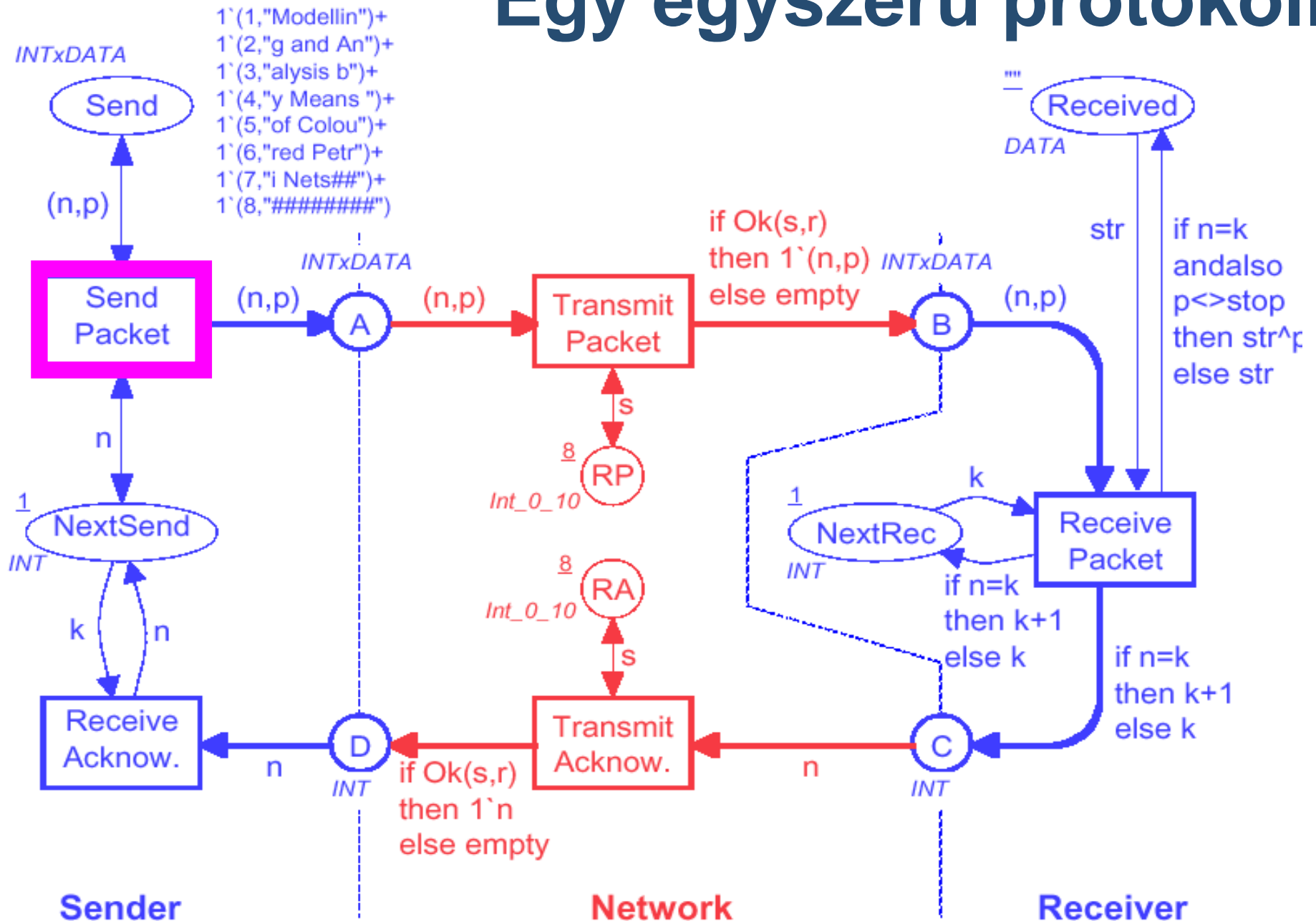
Egy egyszerű protokoll



Egy egyszerű protokoll

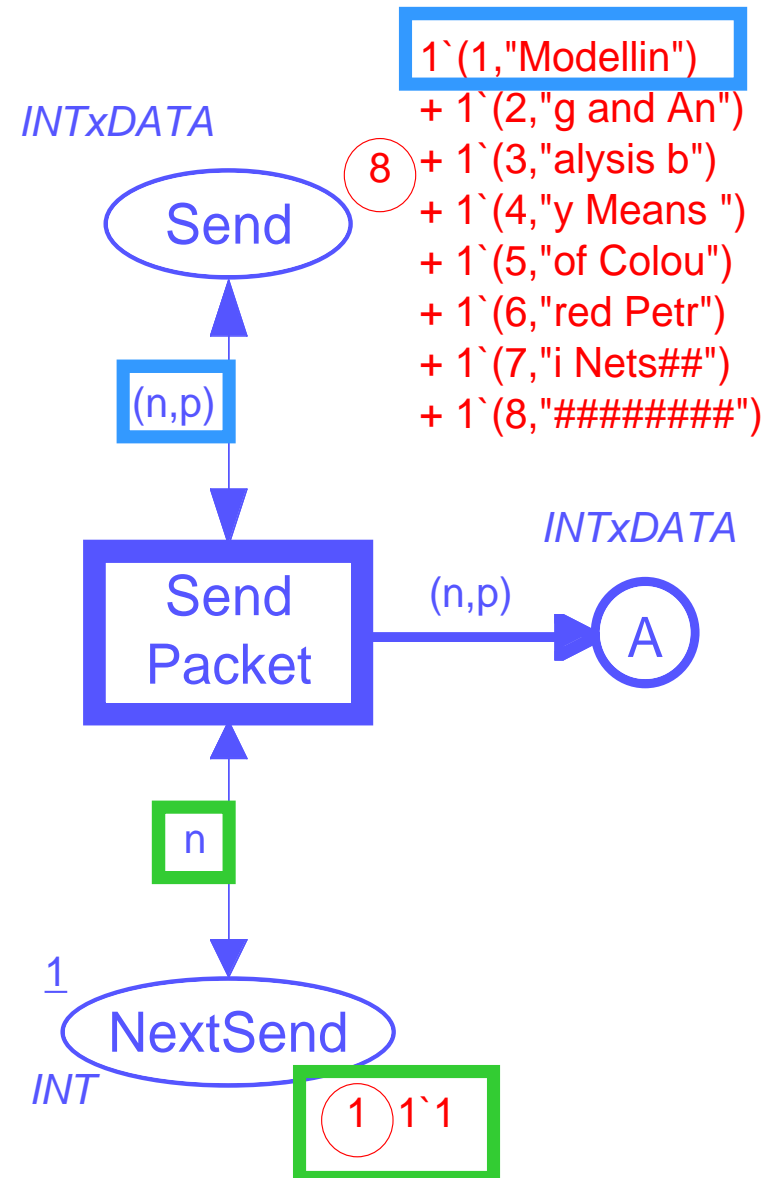


Egy egyszerű protokoll

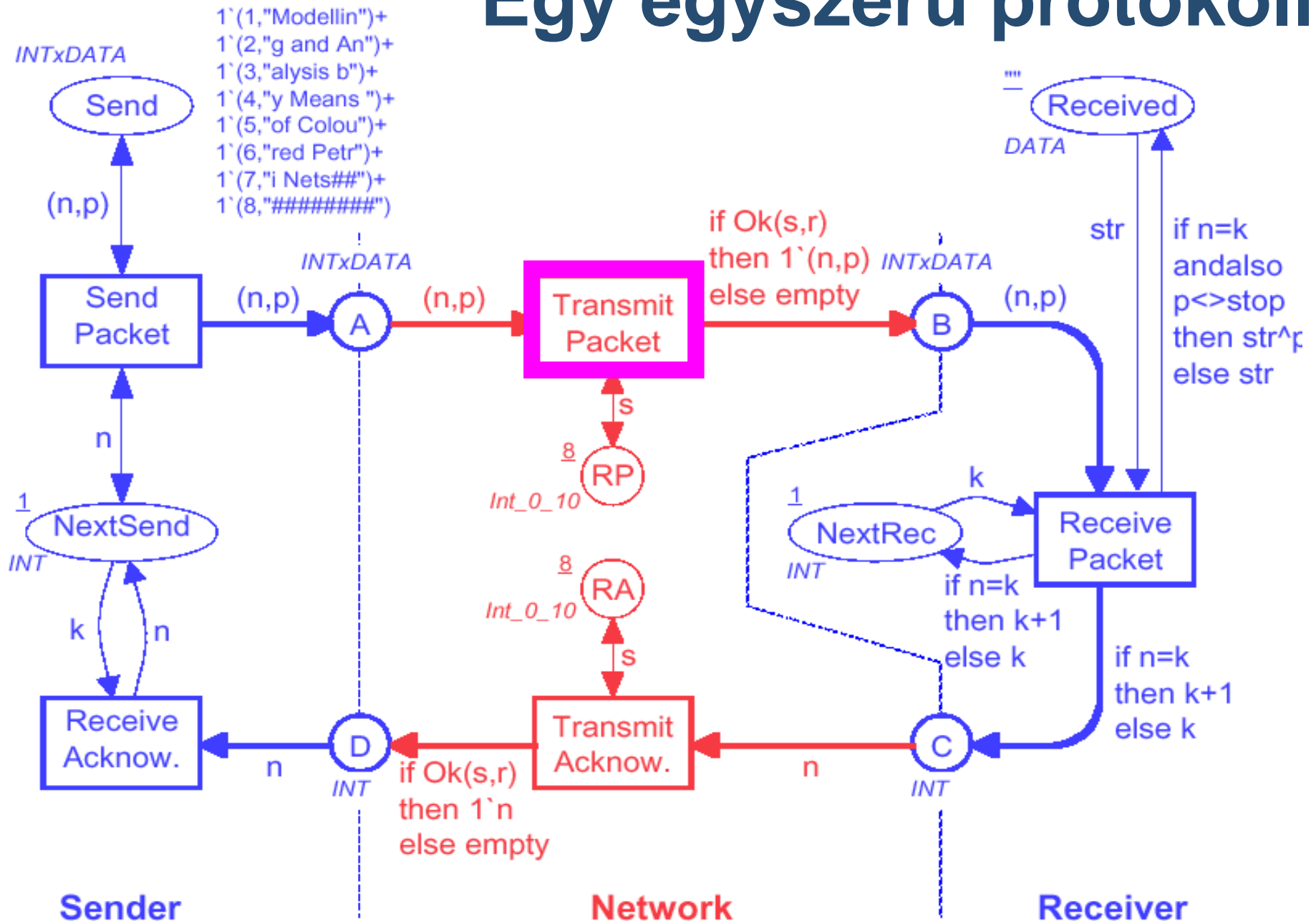


Csomagküldés

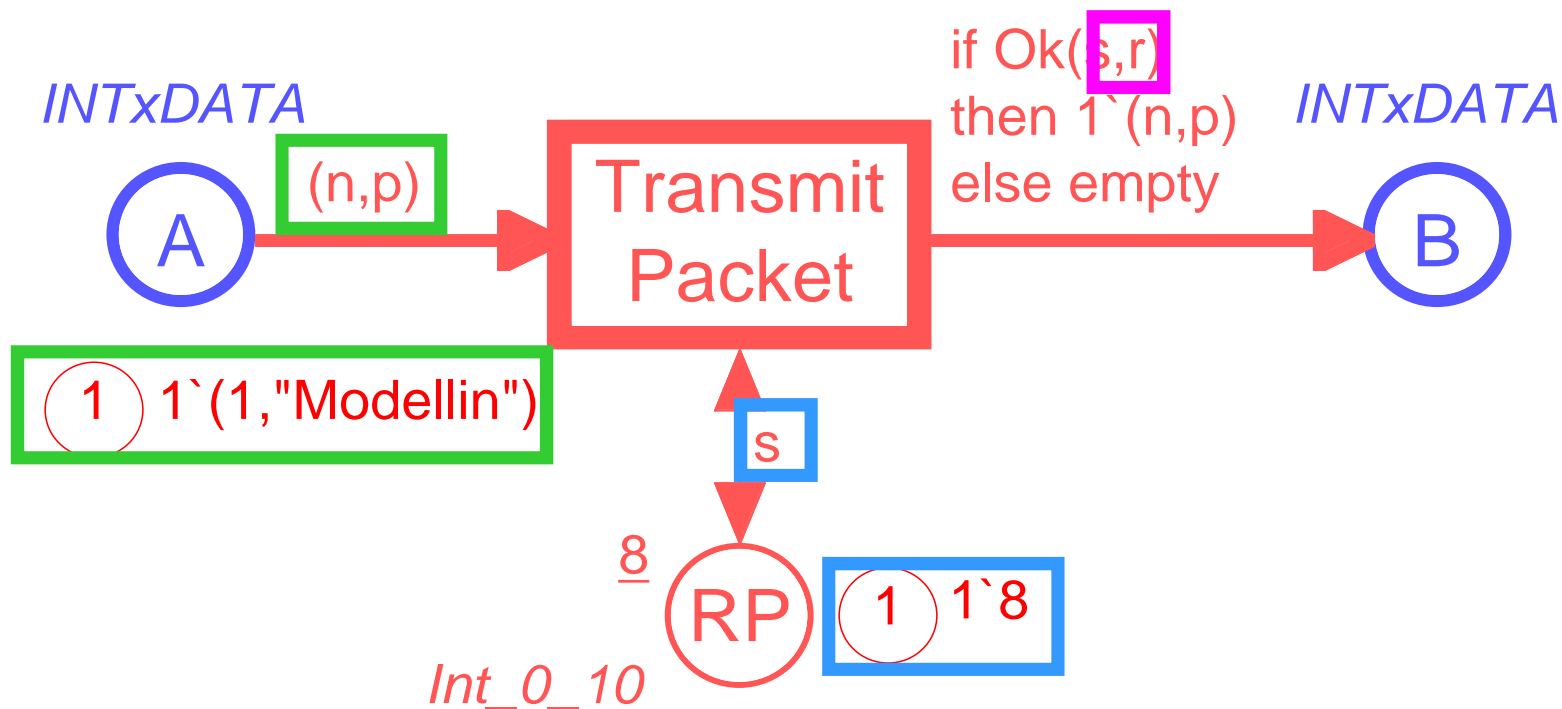
- ◆ Csak a $\langle n=1, p="Modellin" \rangle$ pár engedélyezett.
- ◆ Ha található ilyen pár, akkor a *Send Packet* elküldi azt *A*-nak.
- ◆ Ez azt jelenti, hogy $(1, "Modellin")$ -t elküldtük a hálózaton.
- ◆ A csomag nem törlődik a *Send* helyről és a *NextSend* sem változik.



Egy egyszerű protokoll



Csomag átvitele



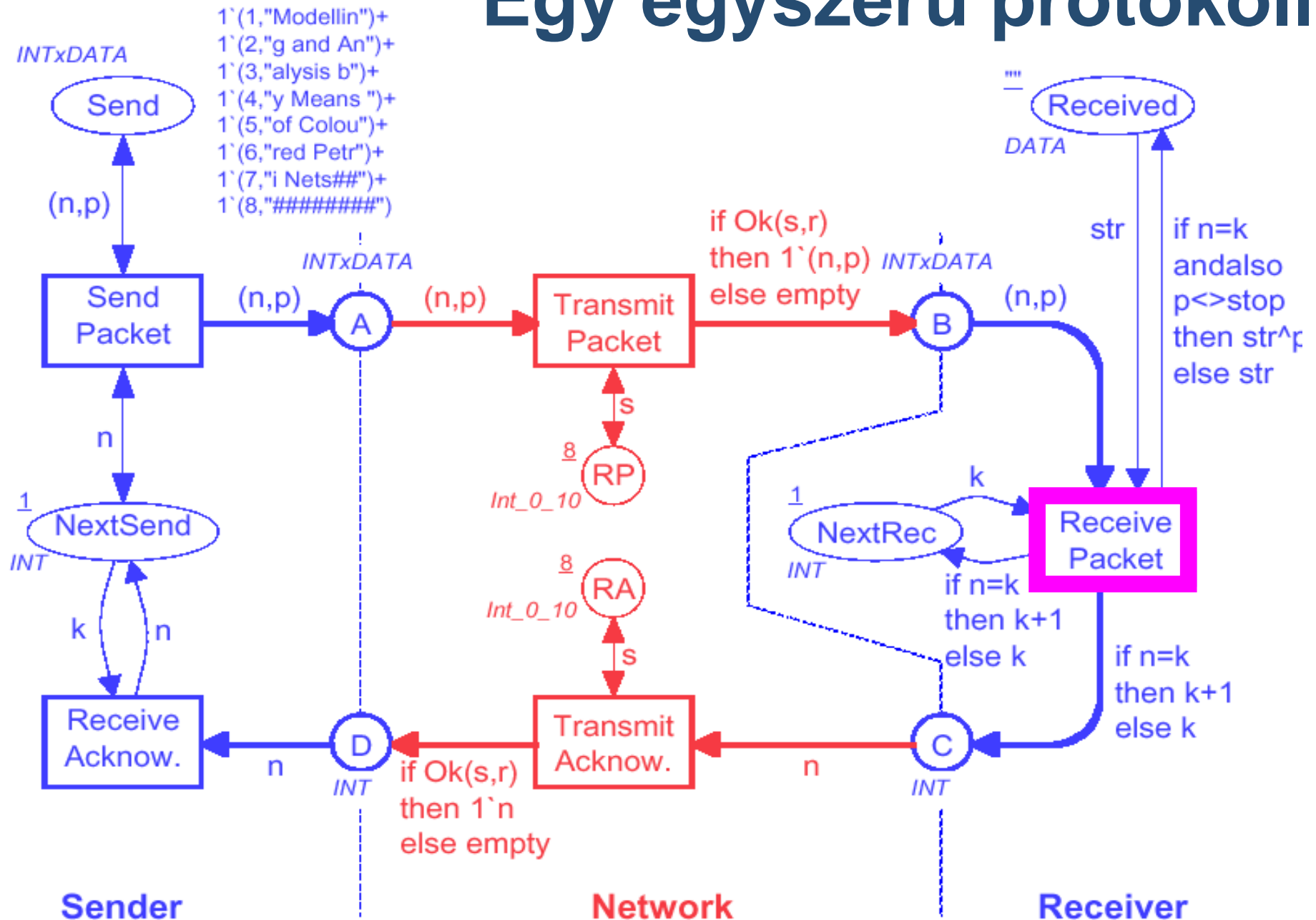
- ◆ Az engedélyezett pár így néz ki:
 - $\langle n=1, p= \text{"Modellin"}, s=8, r=... \rangle$
 - Típus: $r \in 1..10$

Csomagvesztés

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

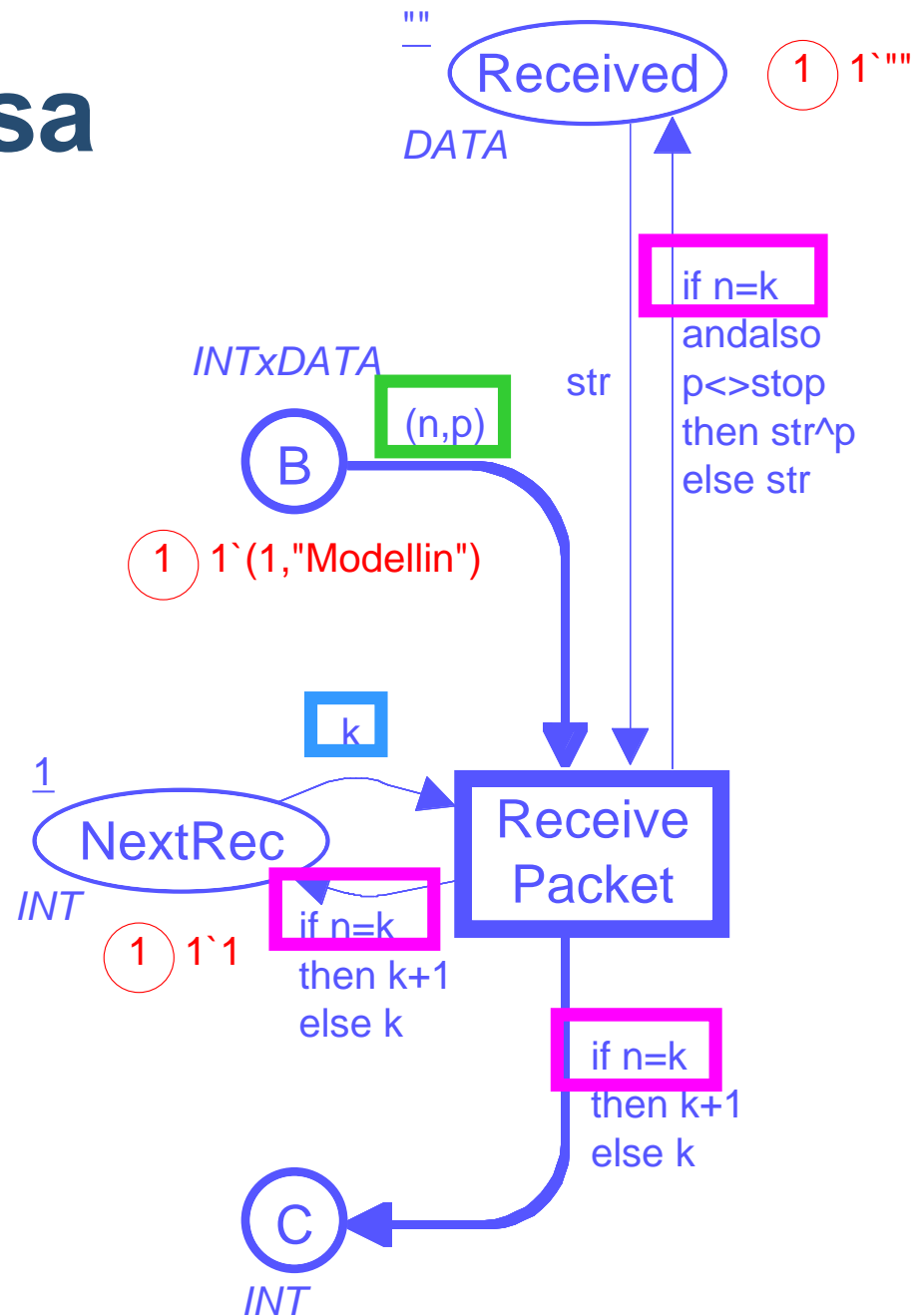
- ◆ A $Ok(s,r)$ függvény ellenőrzi, hogy $r \leq s$
 - For $r \in 1..8, Ok(s,r)=true$.
A token A-ból B-be tesszük. Ez azt jelenti, hogy a csomag *sikeresen átment* a hálózaton.
 - For $r \in 9..10, Ok(s,r)=false$.
Ekkor B-be nem került token. Ekkor *veszett el* a csomag.
- ◆ A CPN szimulátor *véletlenül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

Egy egyszerű protokoll



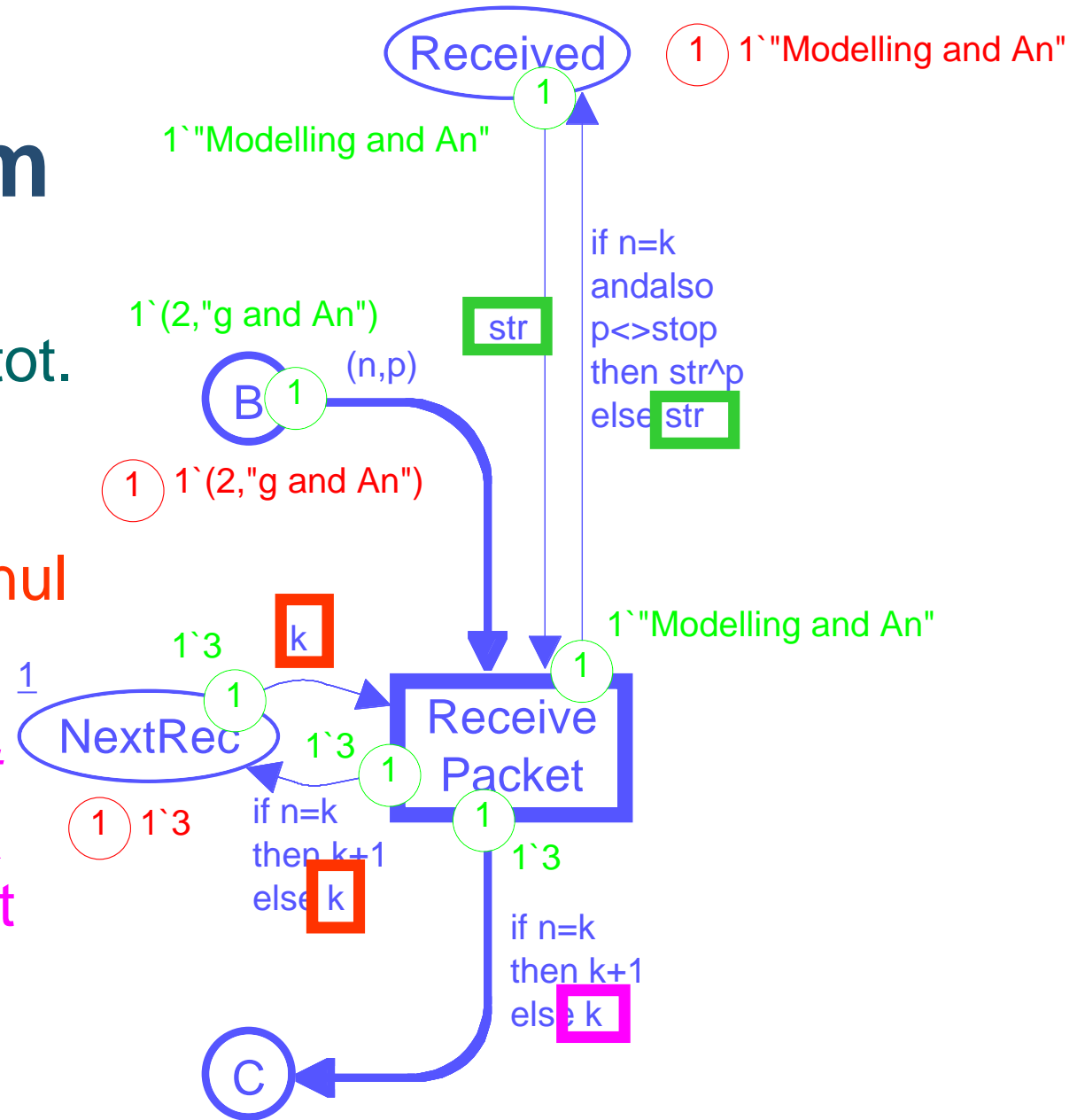
Csomag fogadása

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

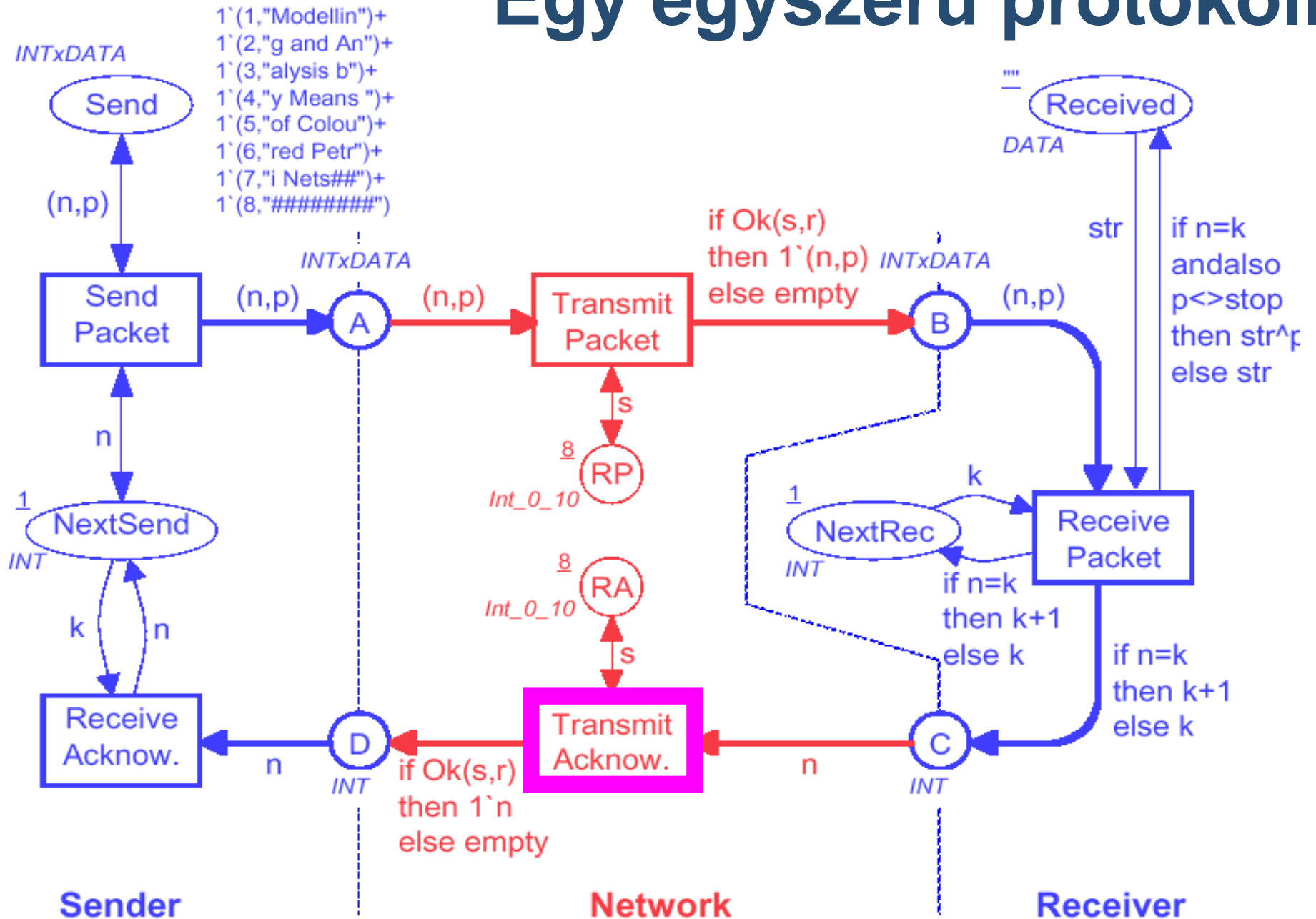


Rossz csomagorszám

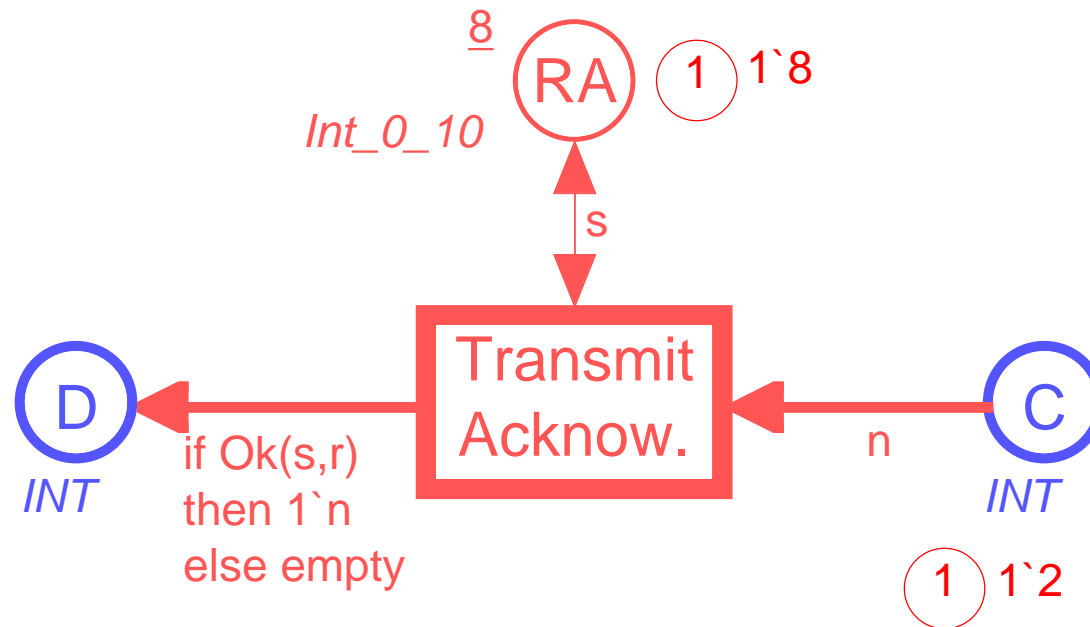
- ◆ *Eldobjuk a kapott adatot.*
- ◆ *A NextRec-t változatlanul hagyjuk.*
- ◆ *Egy acknowledgement üzenetet küldünk. Ez a várt csomag sorszámát tartalmazza.*



Egy egyszerű protokoll

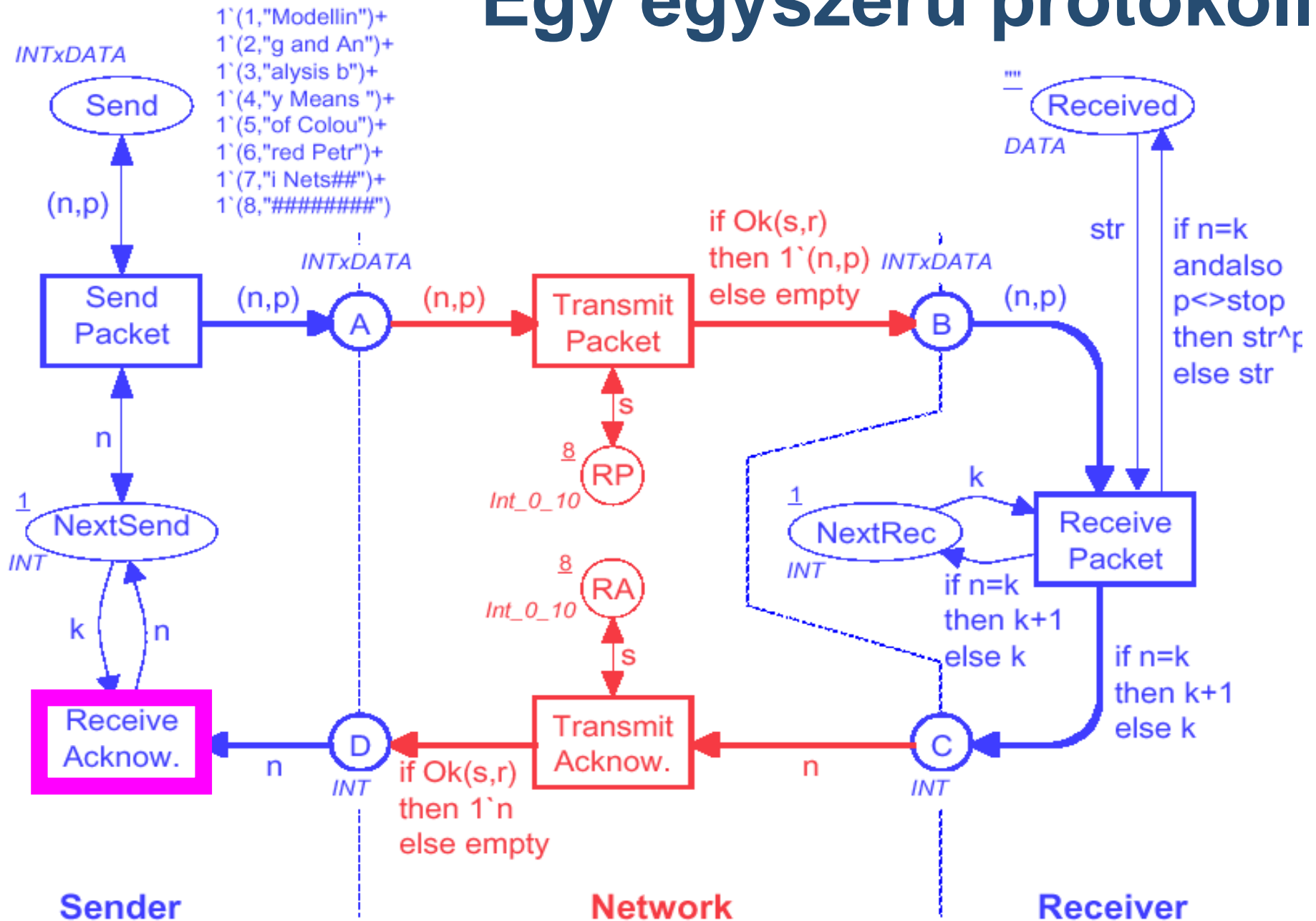


Átvitel visszajelzés

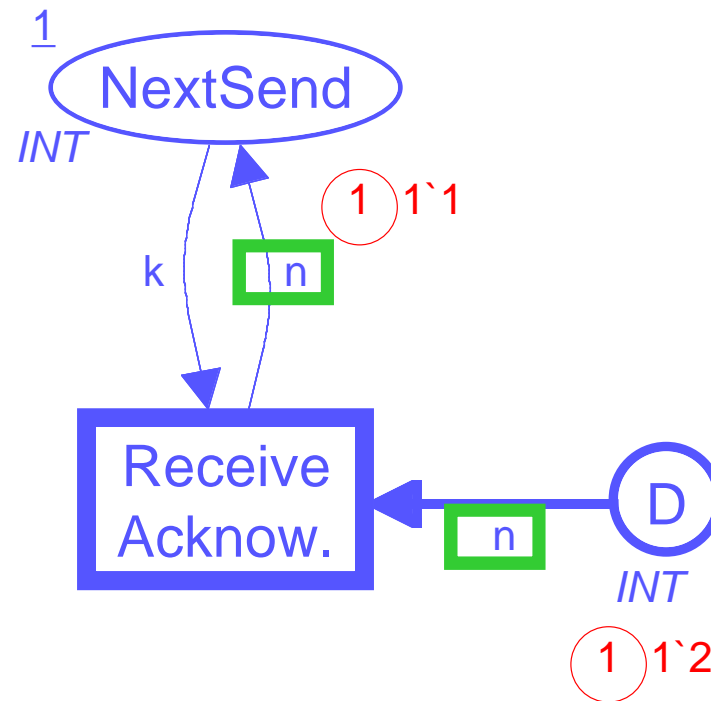


- ◆ Hasonlóan működik a *Csomag átvitelhez*.

Egy egyszerű protokoll



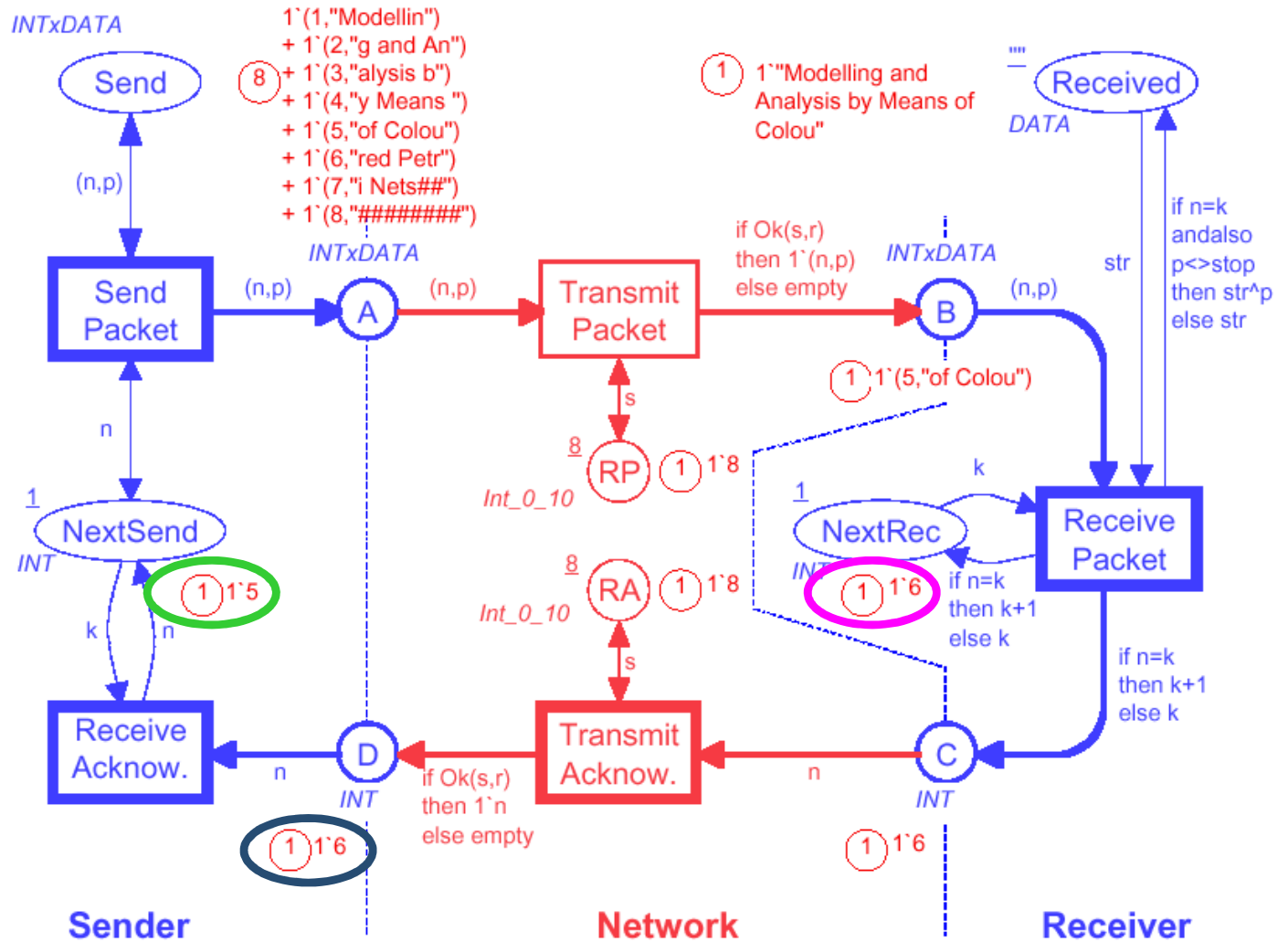
Fogadási visszajelzés



- ◆ 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.

Közbülső állapot

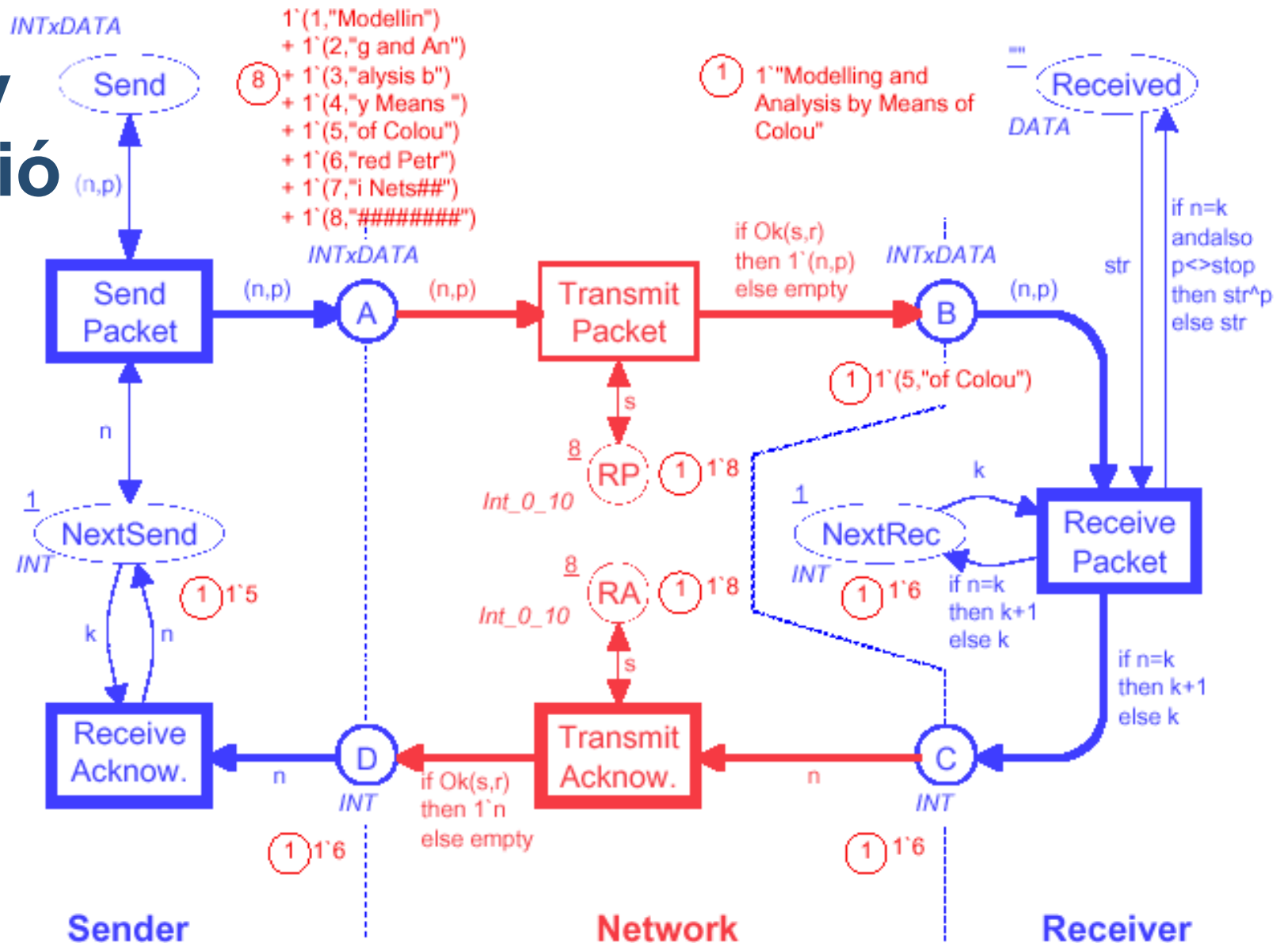
- ◆ Fogadó várja a 6-os csomagot.
- ◆ Küldő még csak az 5-ös csomagot küldi.
- ◆ A 6-os acknowledgement kérő csomag érkezik meg.
- ◆ Ezután a *NextSend* frissítődik majd a *Küldő* elindítja a 6-os csomagot.



Színezett Petri háló szimulációja

- ◆ A szintaktikailag helyes CPN ábrából a CPN tool *legenerálja* a szükséges *kódot*.
 - Kiszámítja, hogy az egyes tranzíciók *engedélyezettek-e*.
 - Majd *végrehajtja* a tranzíciókat.
- ◆ A szintaxis ellenőrzés és a kódgenerálás *inkrementális*. Emiatt a kis módosítások könnyen végrehajthatóak.
- ◆ Két különböző szimulációt különböztetünk meg:
 - *Interaktív* szimuláció: a felhasználó beavatkozhat, de nagyrészt a rendszer dolgozik.
 - *Automata* szimuláció: a rendszer mindent megcsinál.

Interaktív szimuláció



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

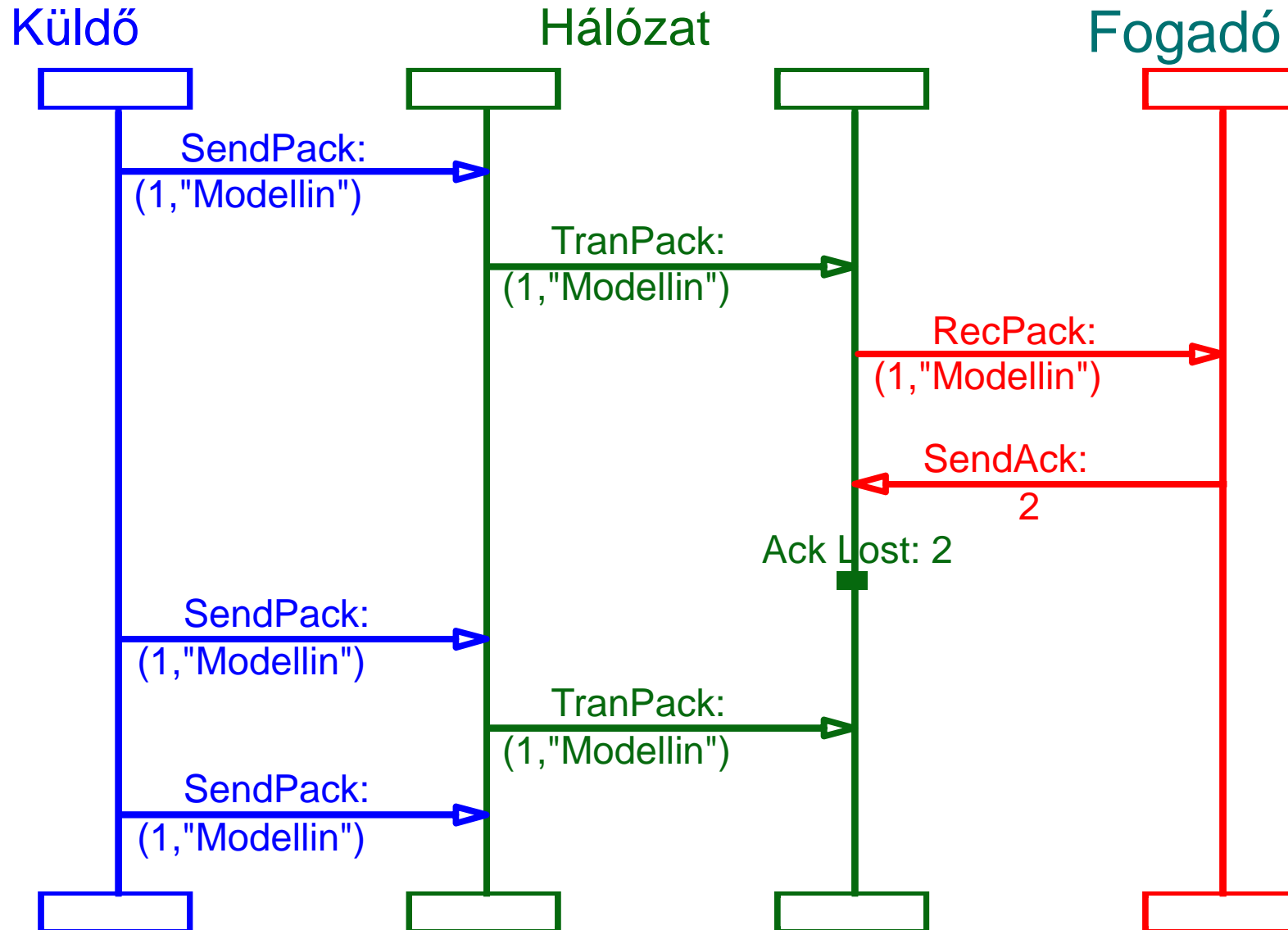
Automatikus szimuláció

- ◆ Amennyiben *nem* kívánjuk nyomon követni magát a szimulációt:
 - ez *nagyon gyors lehet* – sokezer lépés/mp.
 - Lehet *megállási kritériumokat* definiálni
 - Megállásnál a *grafikus képernyő* frissül
 - Ekkor a kialakult helyzetet tanulmányozhatjuk
- ◆ Az automatikus és interaktív szimuláció *hibrid is lehet.*
- ◆ Az automatikus szimuláció eredményének megismeréséhez a felhasználó számára számos eszköz áll rendelkezésre.

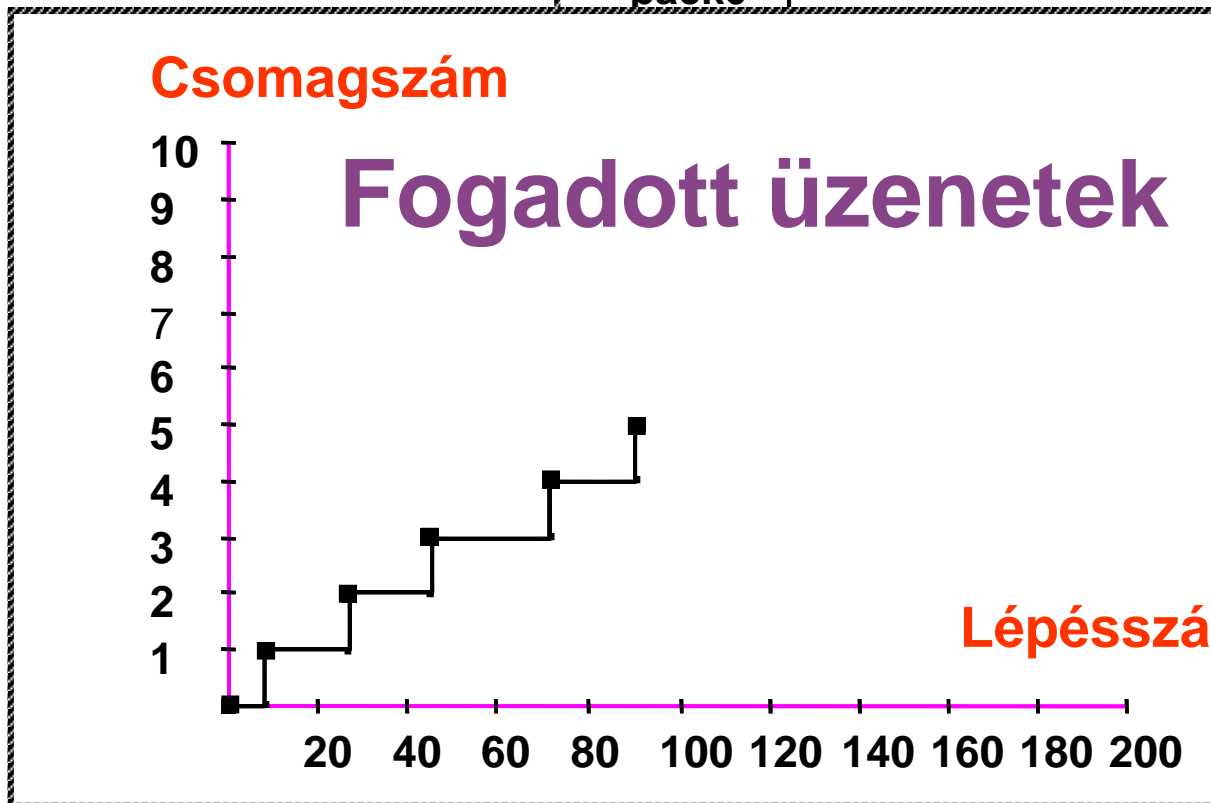
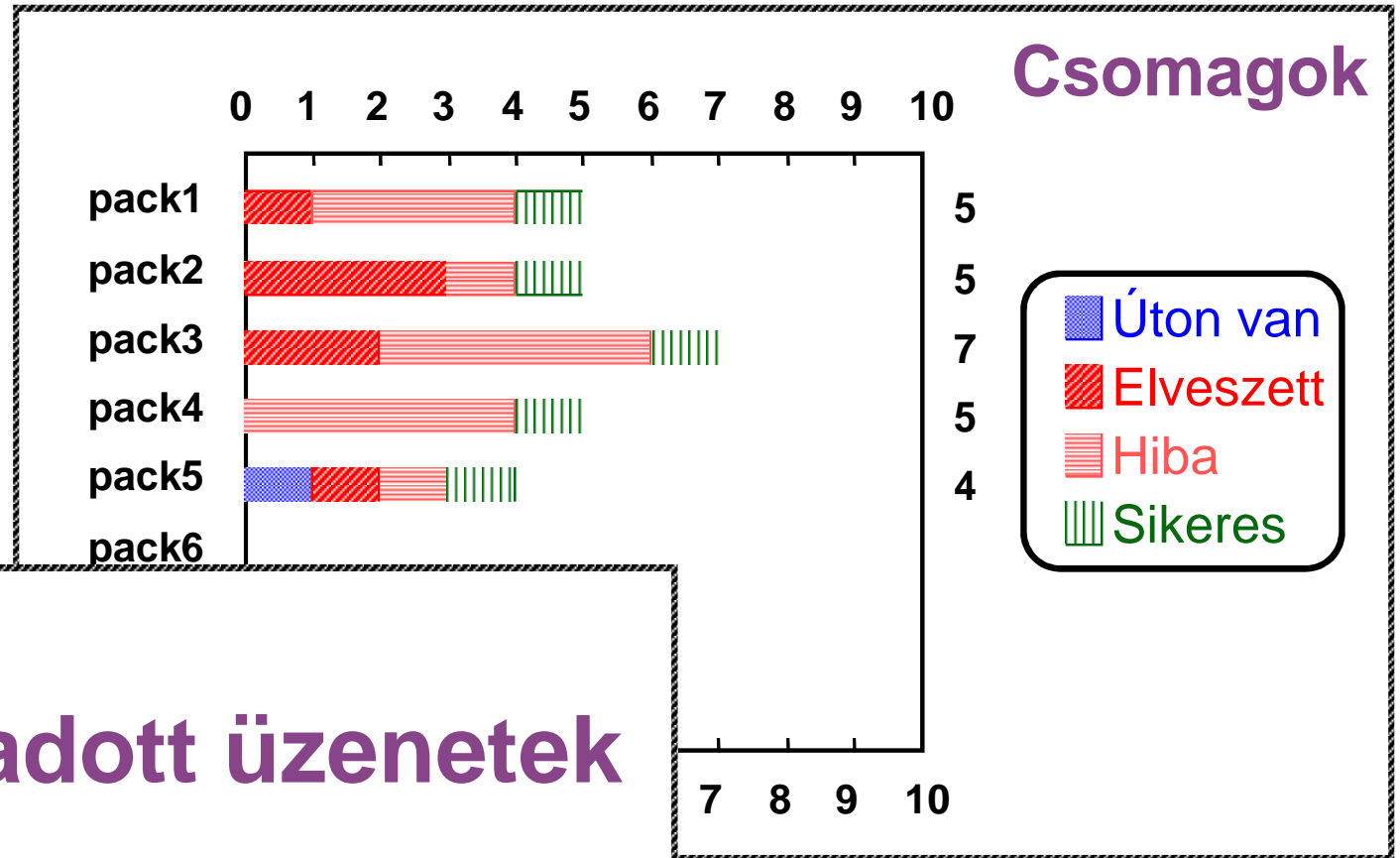
Szimuláció report

- 1 `SendPack@(1:Top#1) {n=1,p="Modellin"}`
- 2 `TranPack@(1:Top#1) {n=1,p="Modellin",r=6,s=8}`
- 3 `SendPack@(1:Top#1) {n=1,p="Modellin"}`
- 4 `TranPack@(1:Top#1) {n=1,p="Modellin",r=3,s=8}`
- 5 `RecPack@(1:Top#1) {k=1,n=1,p="Modellin",str=`
- 6 `SendPack@(1:Top#1) {n=1,p="Modellin"}`
- 7 `SendPack@(1:Top#1) {n=1,p="Modellin"}`
- 8 `TranAck@(1:Top#1) {n=2,r=2,s=8}`

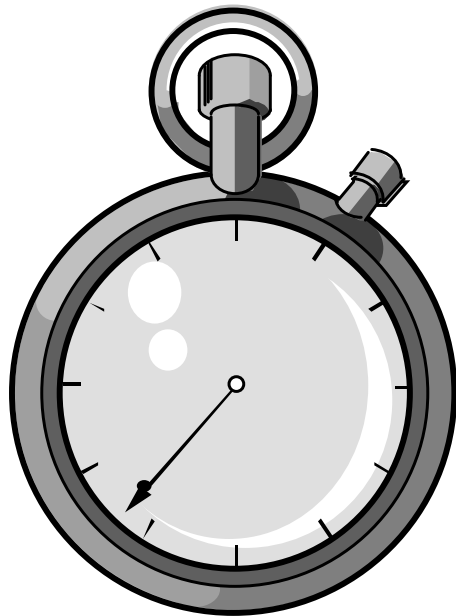
Szekvencia diagram



Üzleti diagramok

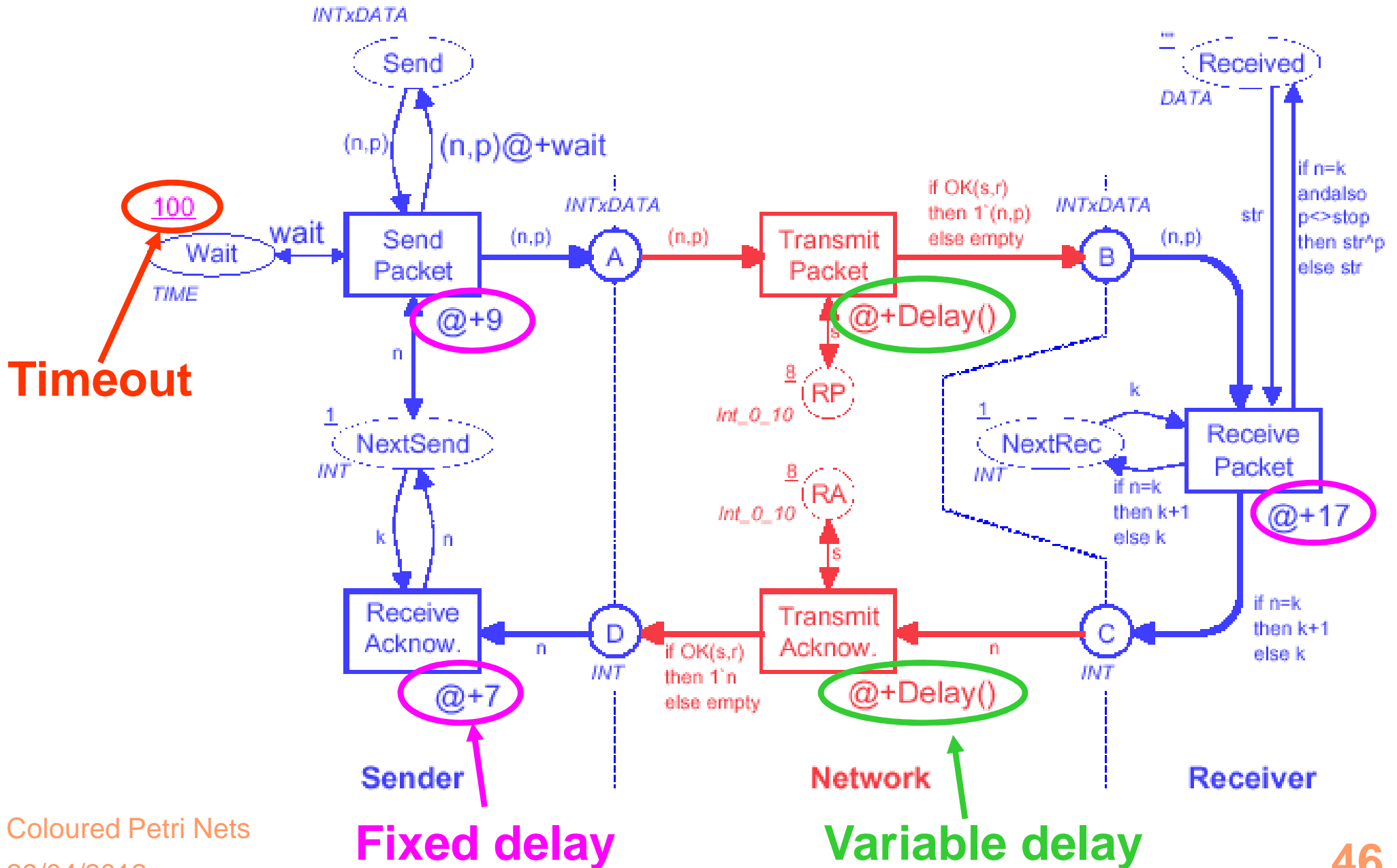


Idő analízis



- ◆ CPN-hez az *idő* fogalmát bevezethetjük. Ezzel ezeket nyertük:
 - *Logikai helyesség.*
Mégkívánt 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

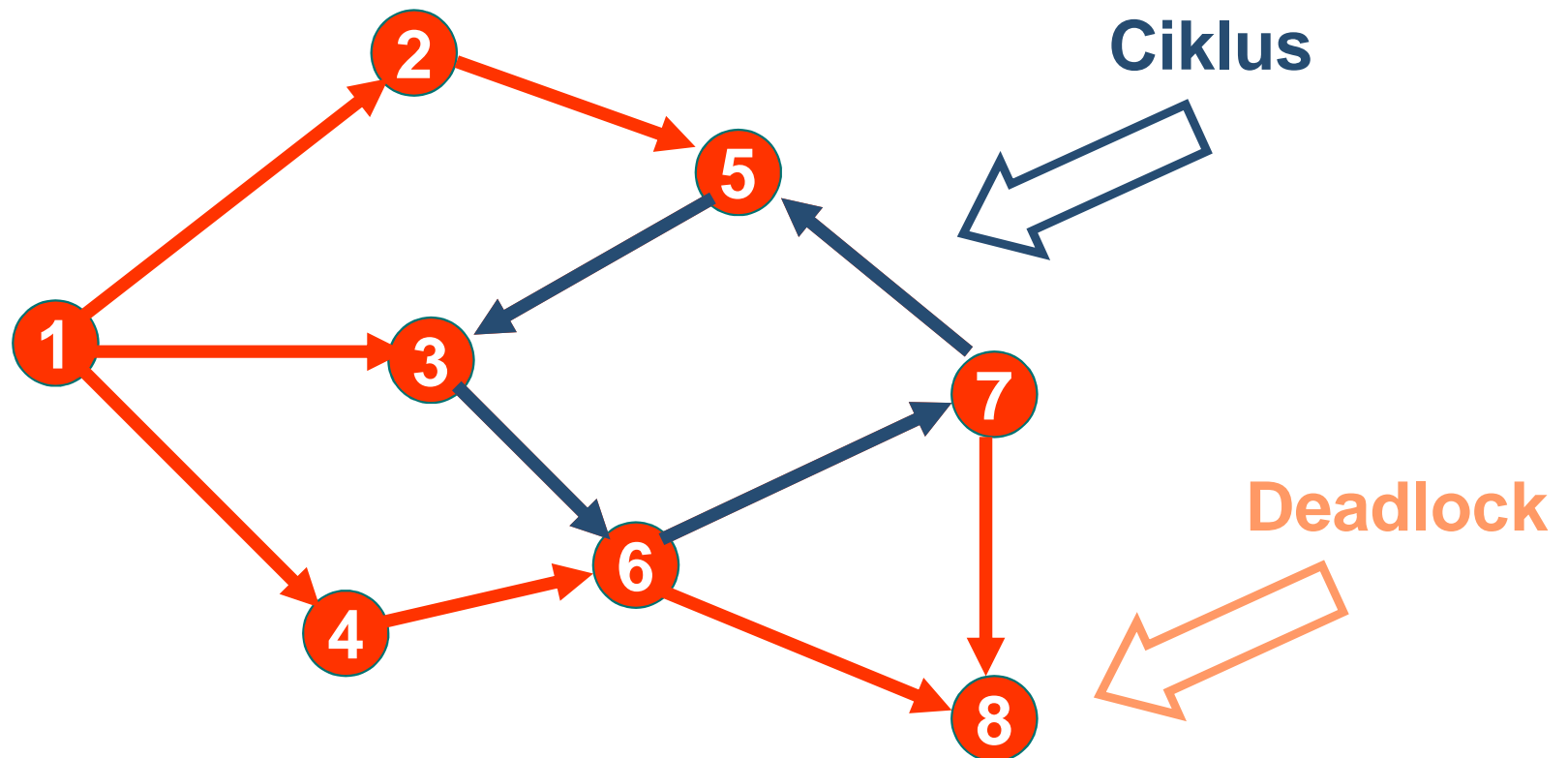


A CPN verifikációja

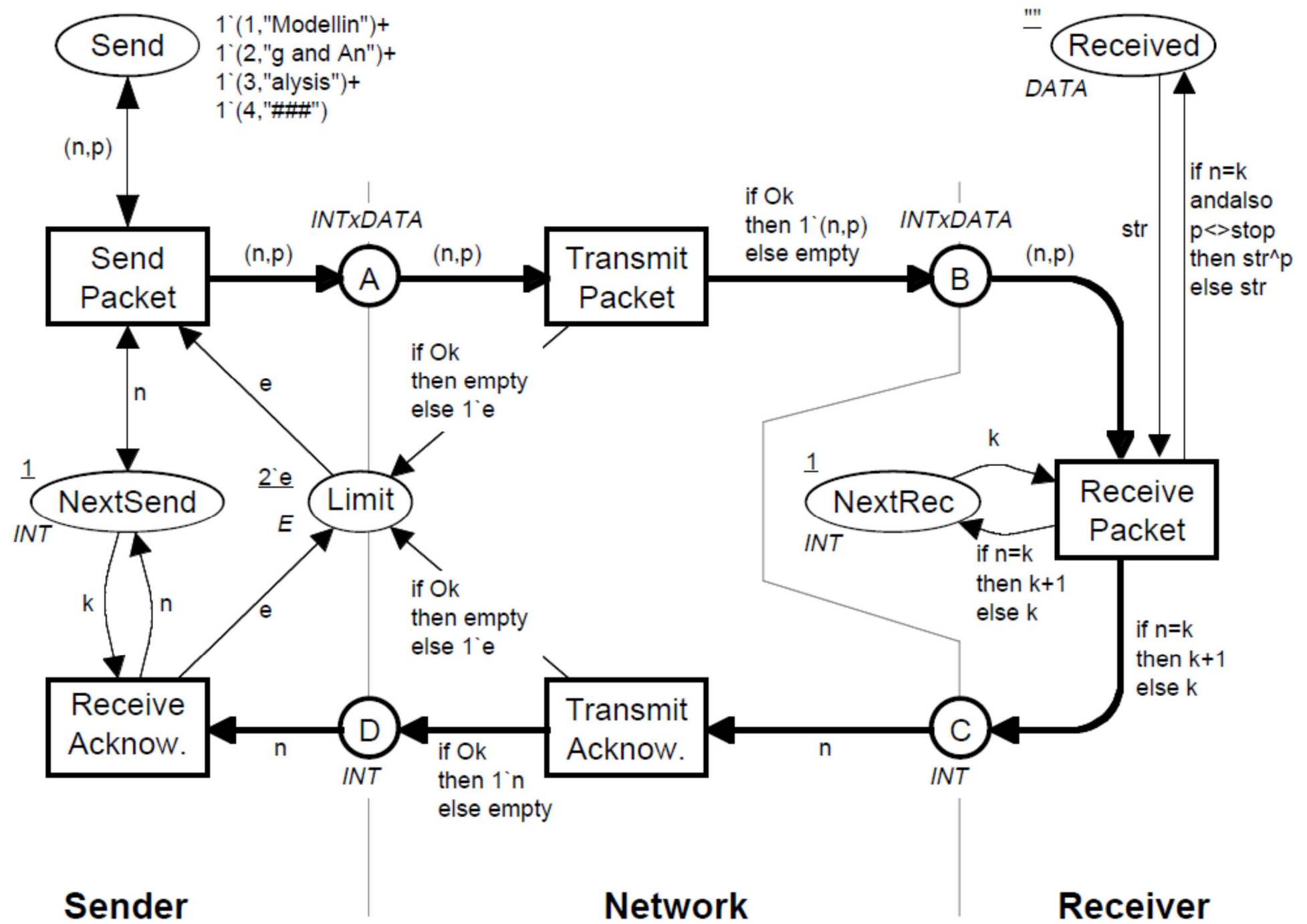
- ◆ A CPN verifikációja a következőkből áll:
 - *Állapottér.*
 - Hely és tranzíció *invariánsok.*
 - Hasonlóan a programozási nyelvek invariánsaihoz.

Állapottér

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



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

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 fairness tulajdonság azt jelenti, hogy egy tranzíció milyen gyakran tüzelhet.

Legrövidebb út megkeresése

- ◆ A *legrövidebb utat* szeretnénk megtalálni a *kezdeti eloszlásból* a *végeloszlásba*.

```
val path =  
NodesInPath(1,452);
```

```
Length(path);
```

↑
Lekérdezés

```
> val path =  
[1,2,3,5,8,11,15,20,27,38,50,  
64,80,102,133,164,199,243,  
301,375,452] : Node list
```

```
> 20 : int
```

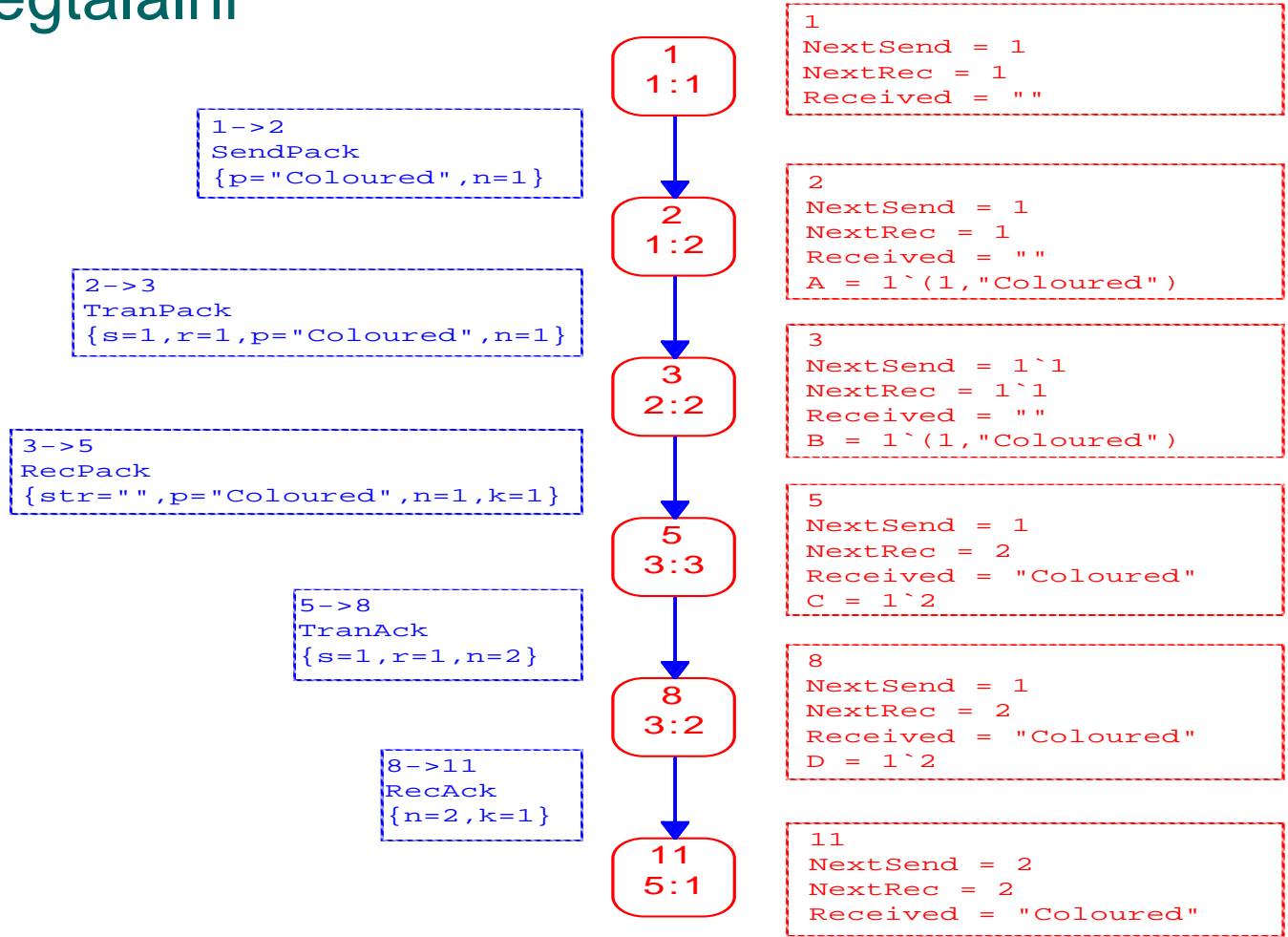
↑
válasz

A legrövidebb út kirajzolása

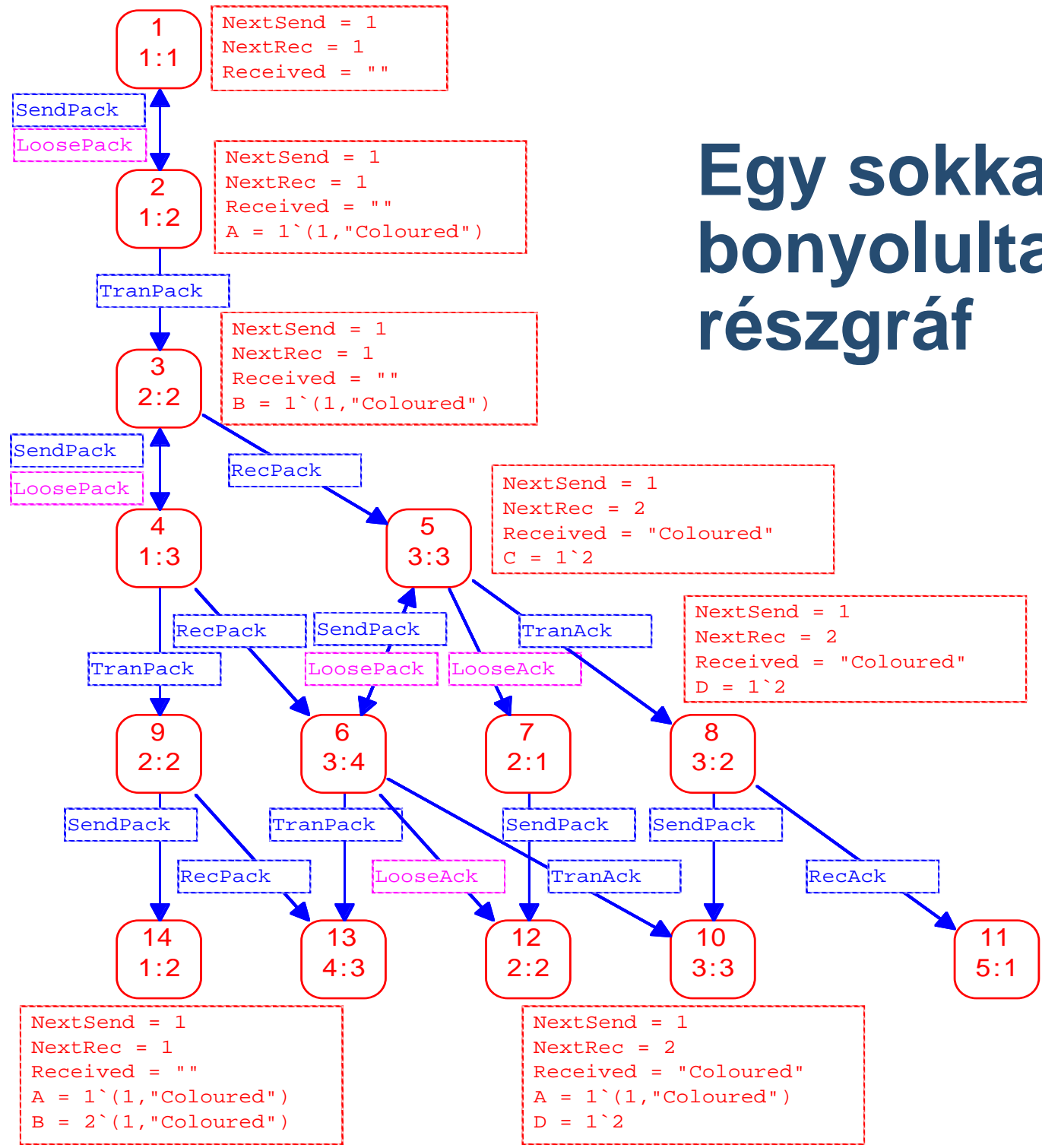
DisplayNodePath [1,2,3,5,8,11]; > () : unit

- ◆ A legrövidebb út kezdetét kívánjuk megtalálni

DisplayNodePath; [1, 2, 3, 5, 8, 11]; > () : unit



Egy sokkal bonyolultabb részgráf



Nem szabványos lekérdezések

- ◆ Lehetséges-e a NextSend számlálót csökkenteni?

PredArcs

```
(EntireGraph,  
fn a => ((ms_to_col(Mark.NextSend 1  
                (SourceNode a))) >  
        (ms_to_col(Mark.NextSend 1  
                (DestNode a))))),  
10);
```

>[10179,10167,10165,10159,10055,10052,10035,
10031,10019,10007] : Arc list

IGEN!

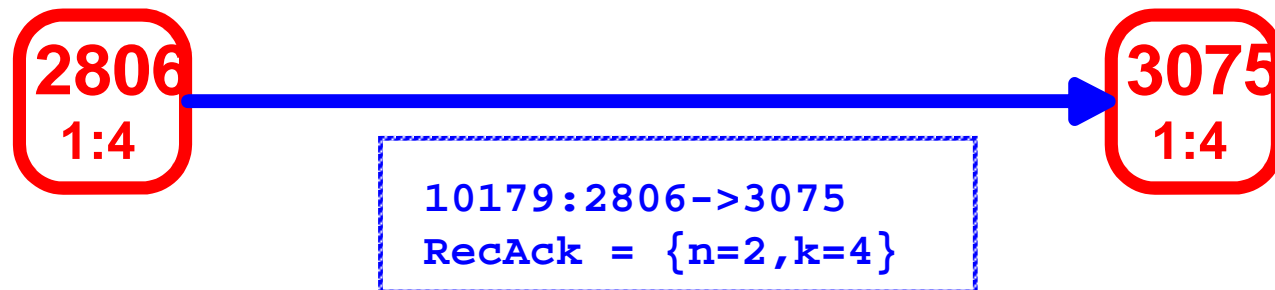
Számlálós példa

```
DisplayArcs [10179];
```

```
> () : unit
```

```
NextSend = 4  
NextRec = 5  
Received = "Coloured Petri  
Nets####"  
A = 1^(4, "#####")  
B = 2^(4, "#####")  
C = 1^5  
D = 1^2+ 1^5
```

```
NextSend = 2  
NextRec = 5  
Received = "Coloured  
Petri Nets####"  
A = 1^(4, "#####")  
B = 2^(4, "#####")  
C = 1^5  
D = 1^5
```



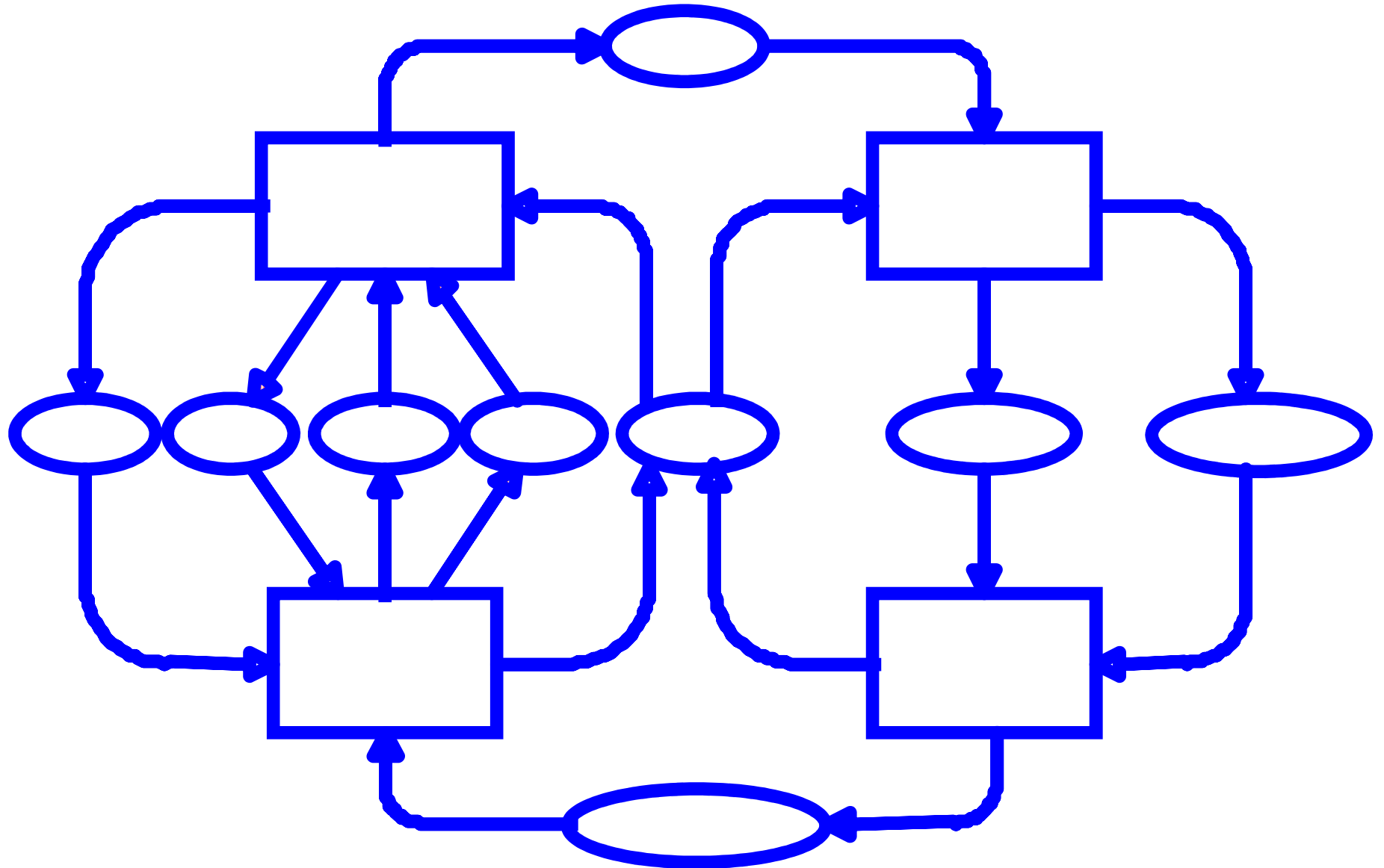
Temporális logika

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

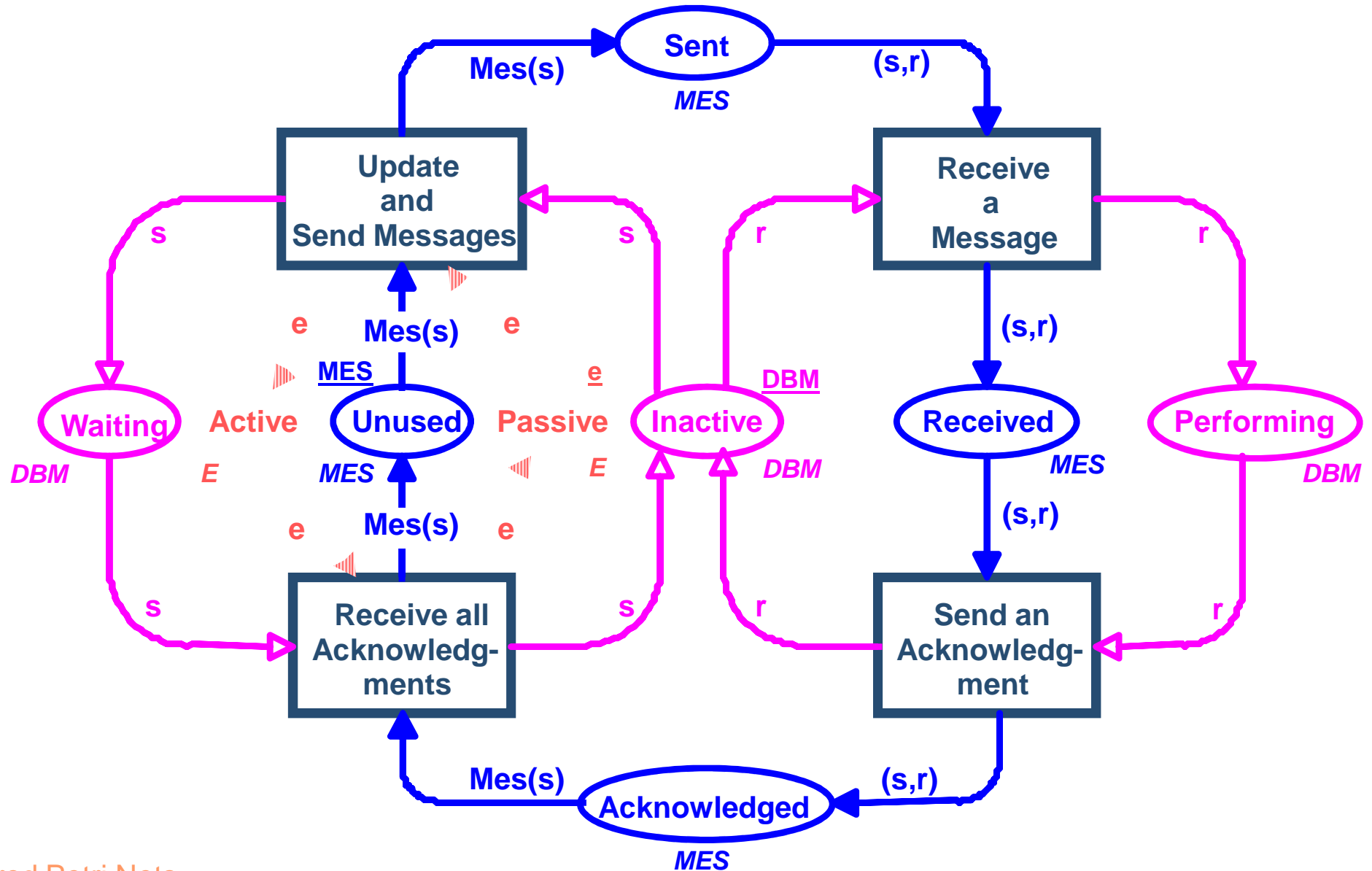
Állapottér analízis – előnyök/hátrányok

- ◆ 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

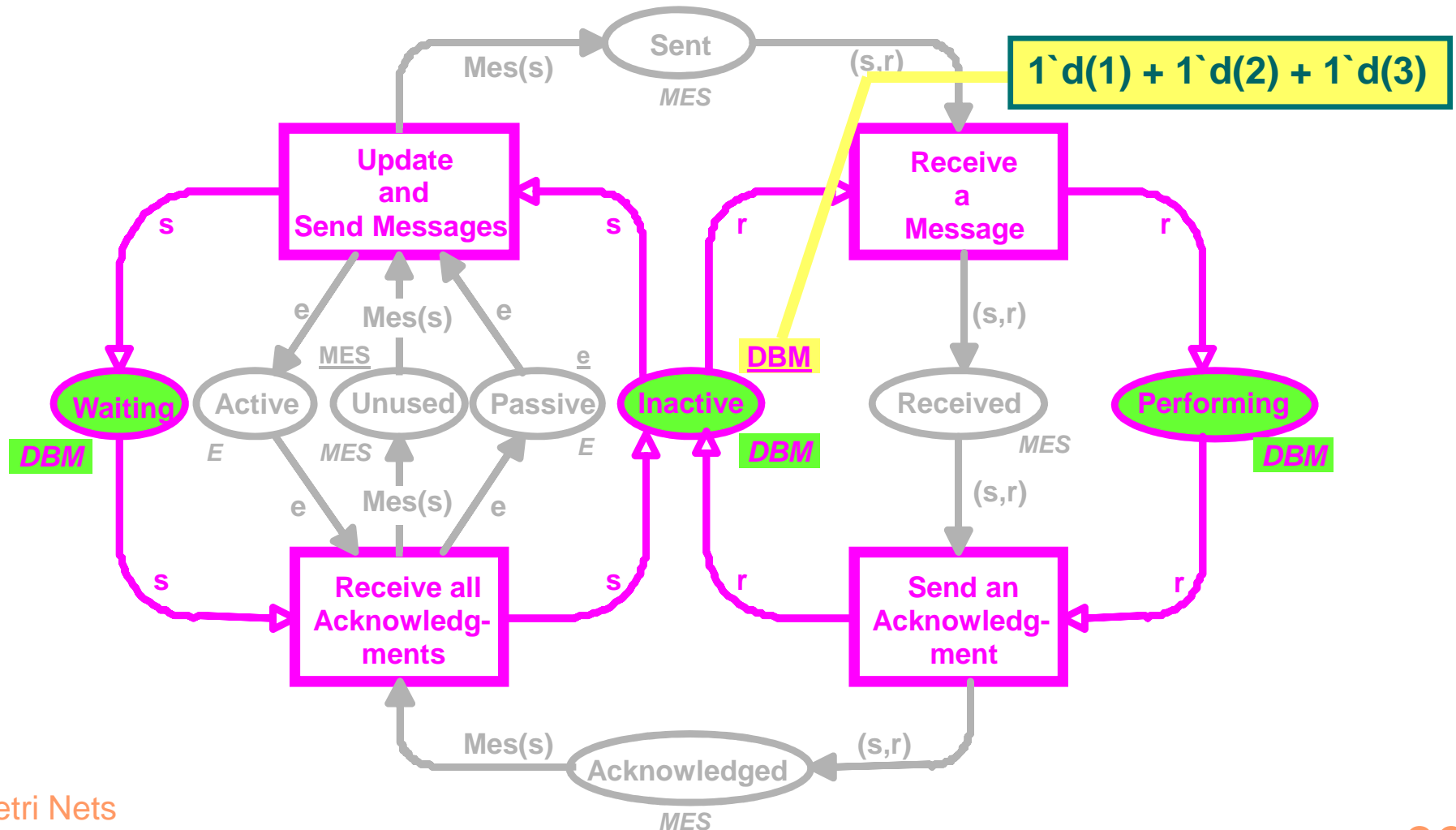


Elosztott adatbázis



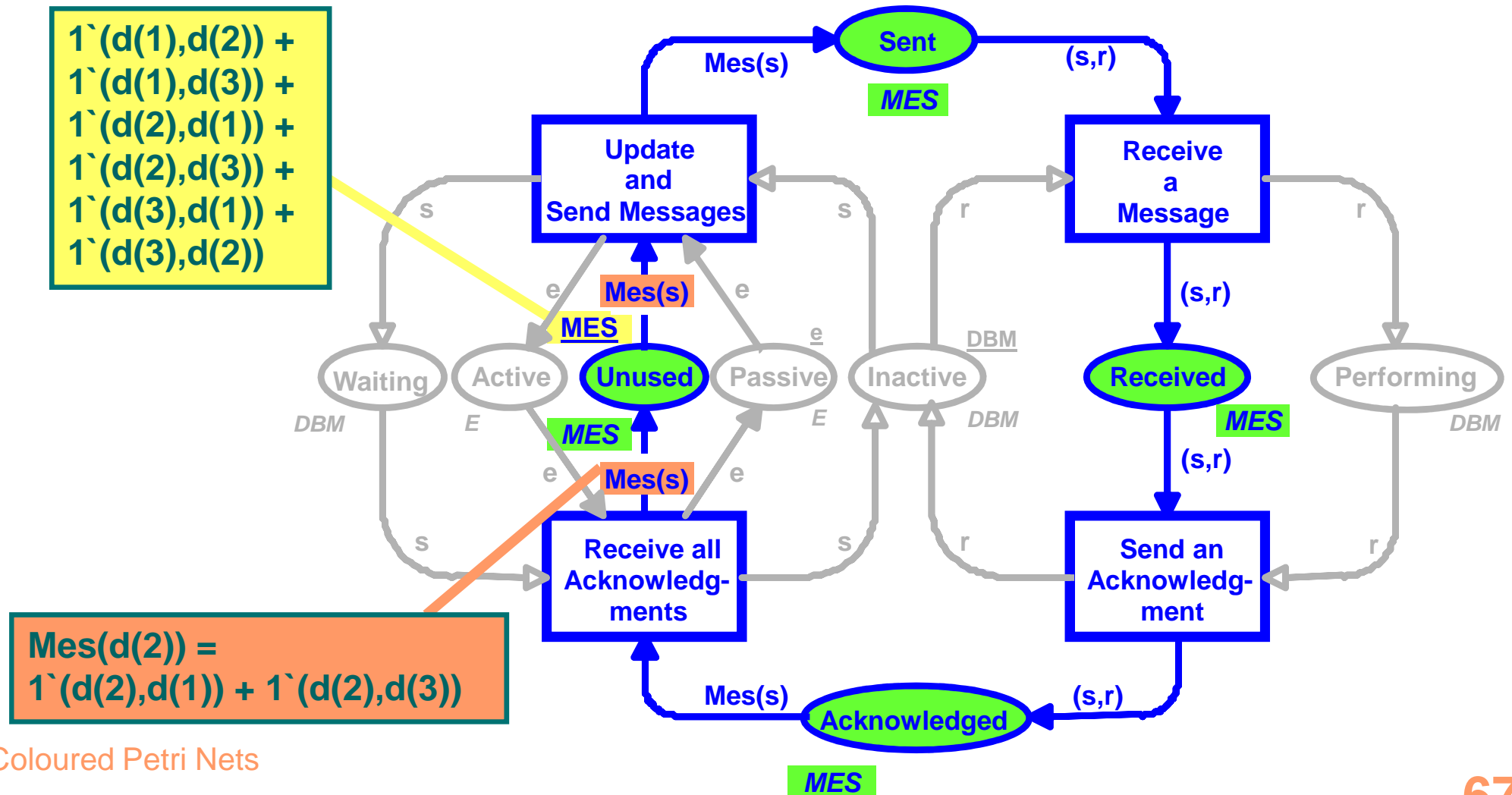
Adatbázis kezelők

- ◆ DBM = {d(1),d(2),d(3)}



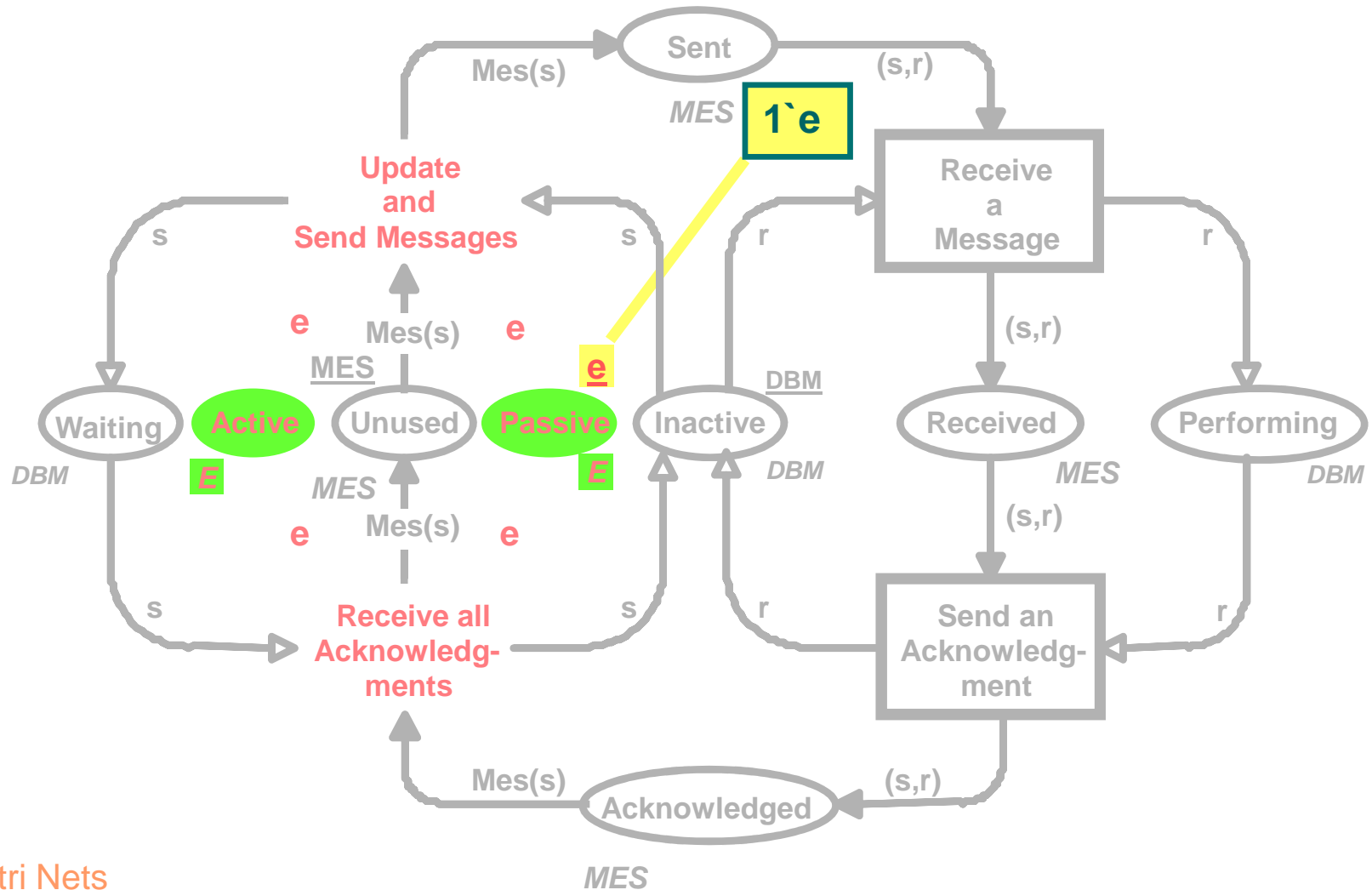
Üzenet bufferek

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

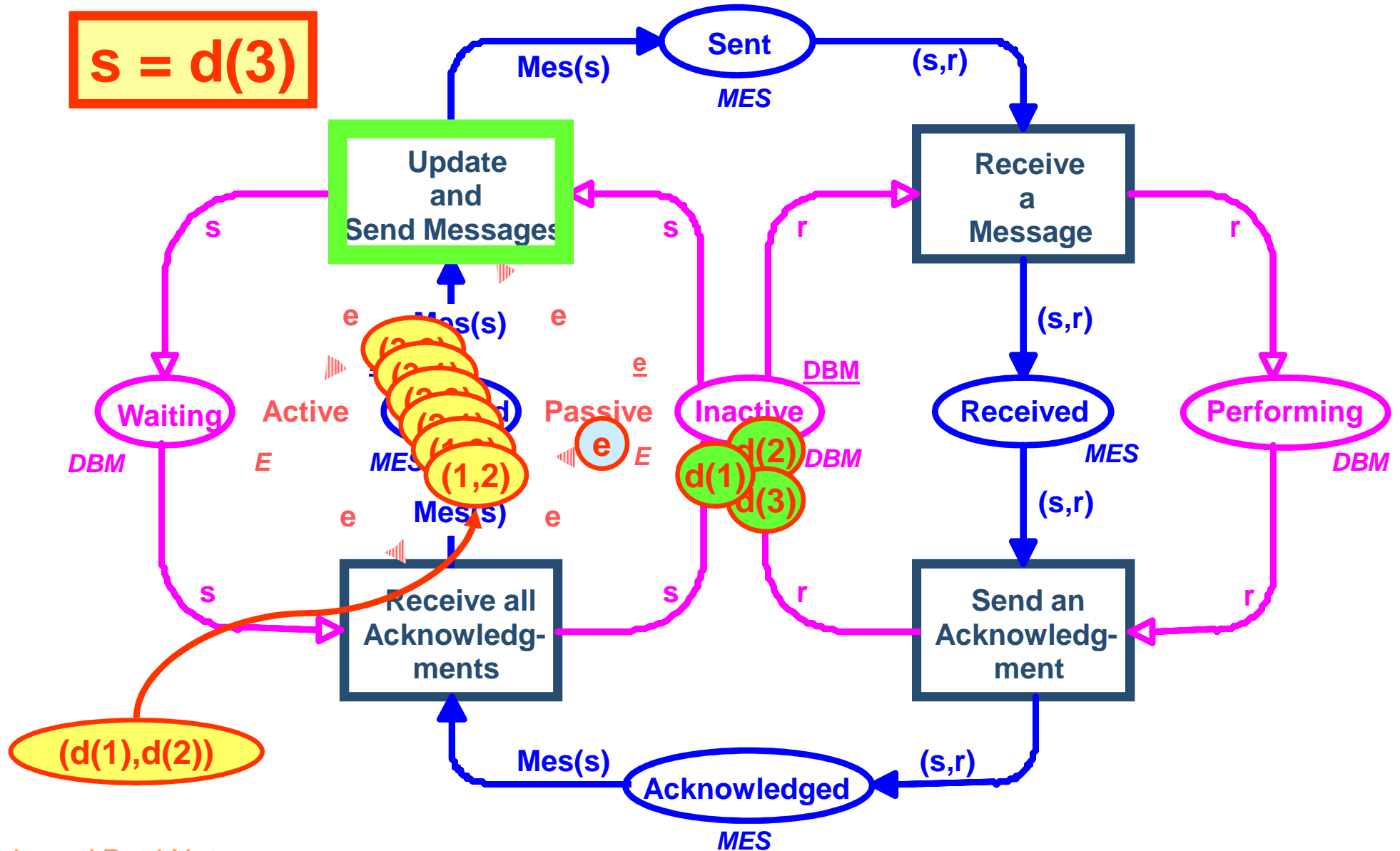


Kölcsönös kizárás

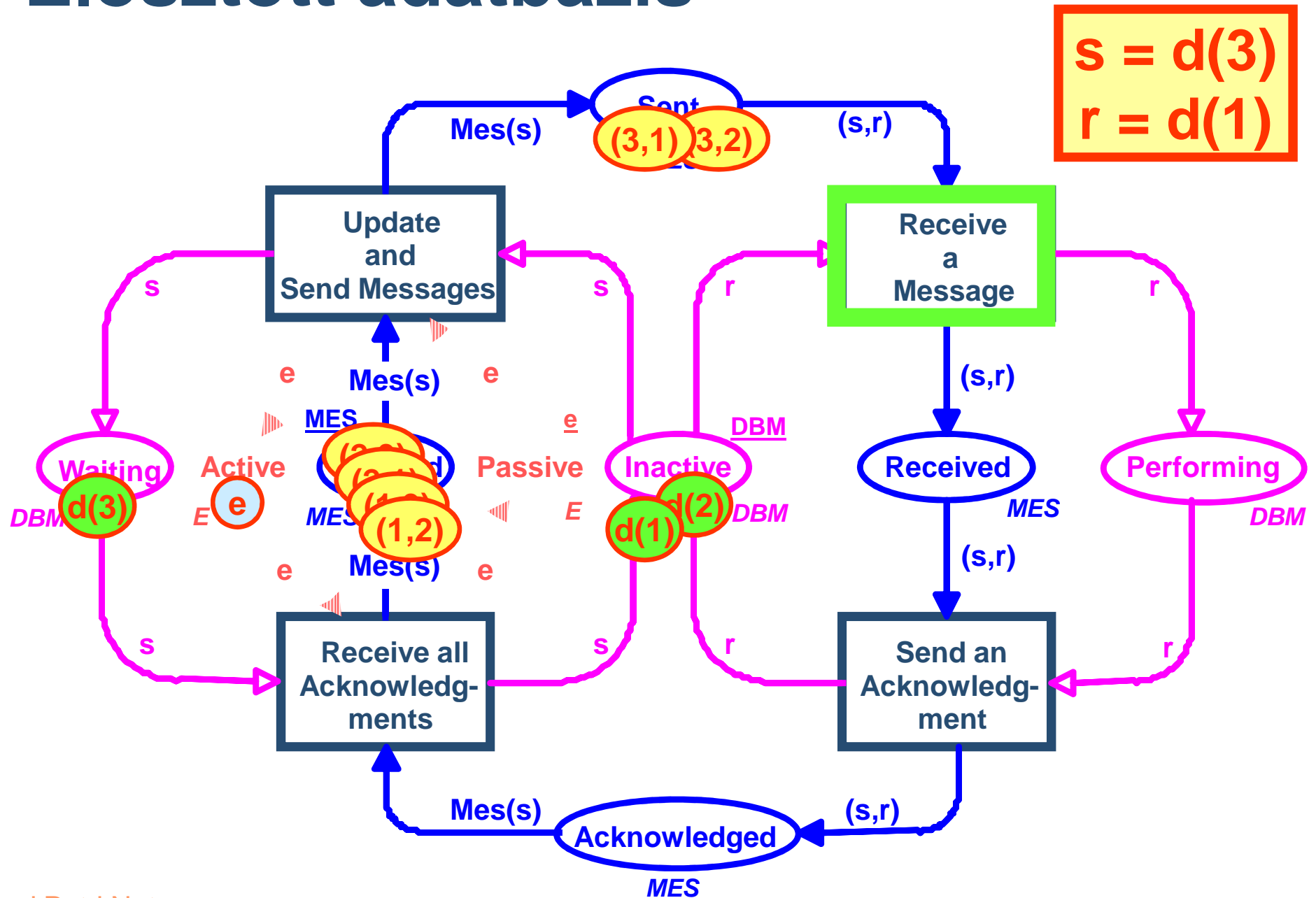
◆ $E = \{e\}$



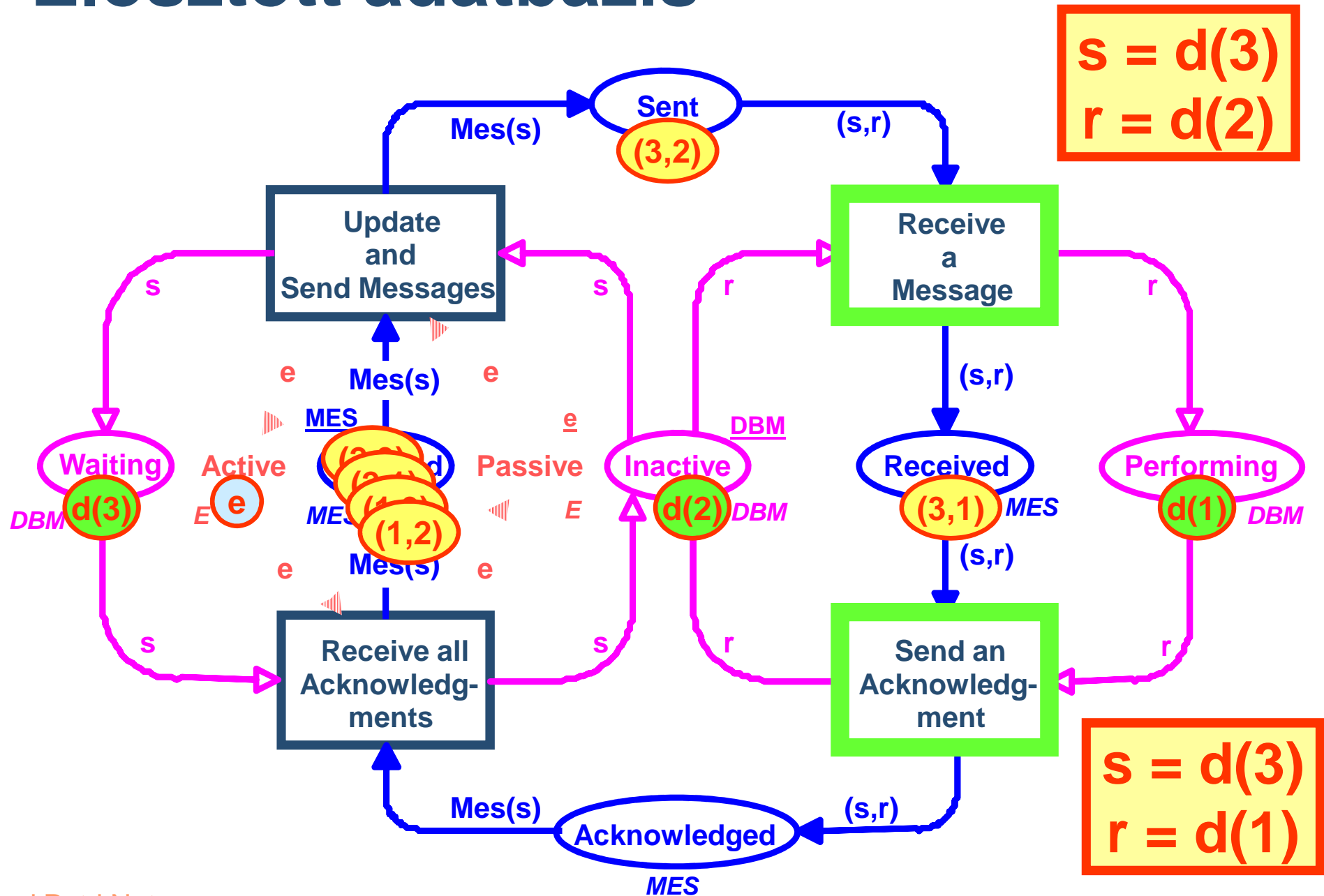
Elosztott adatbázis



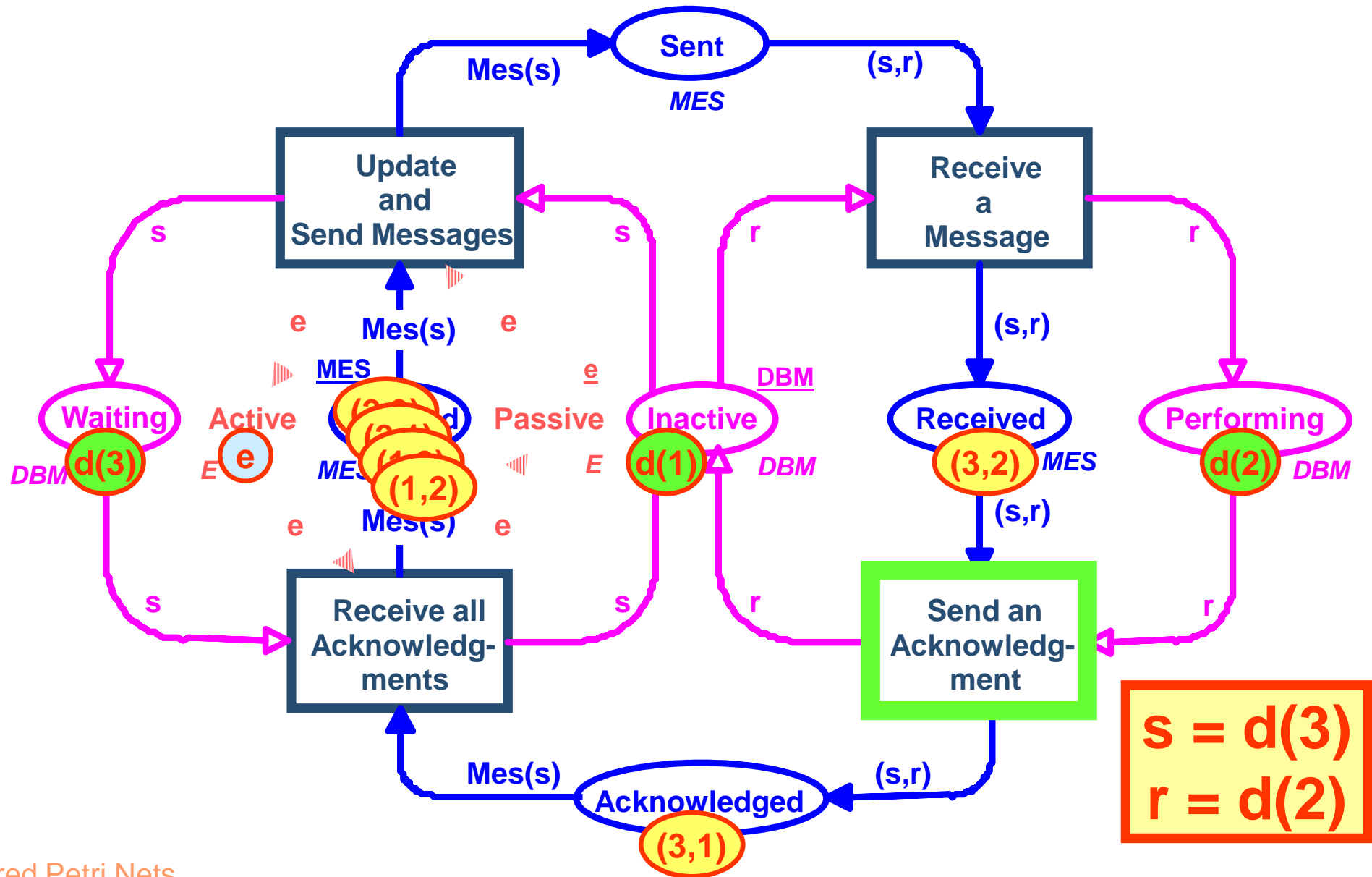
Elosztott adatbázis



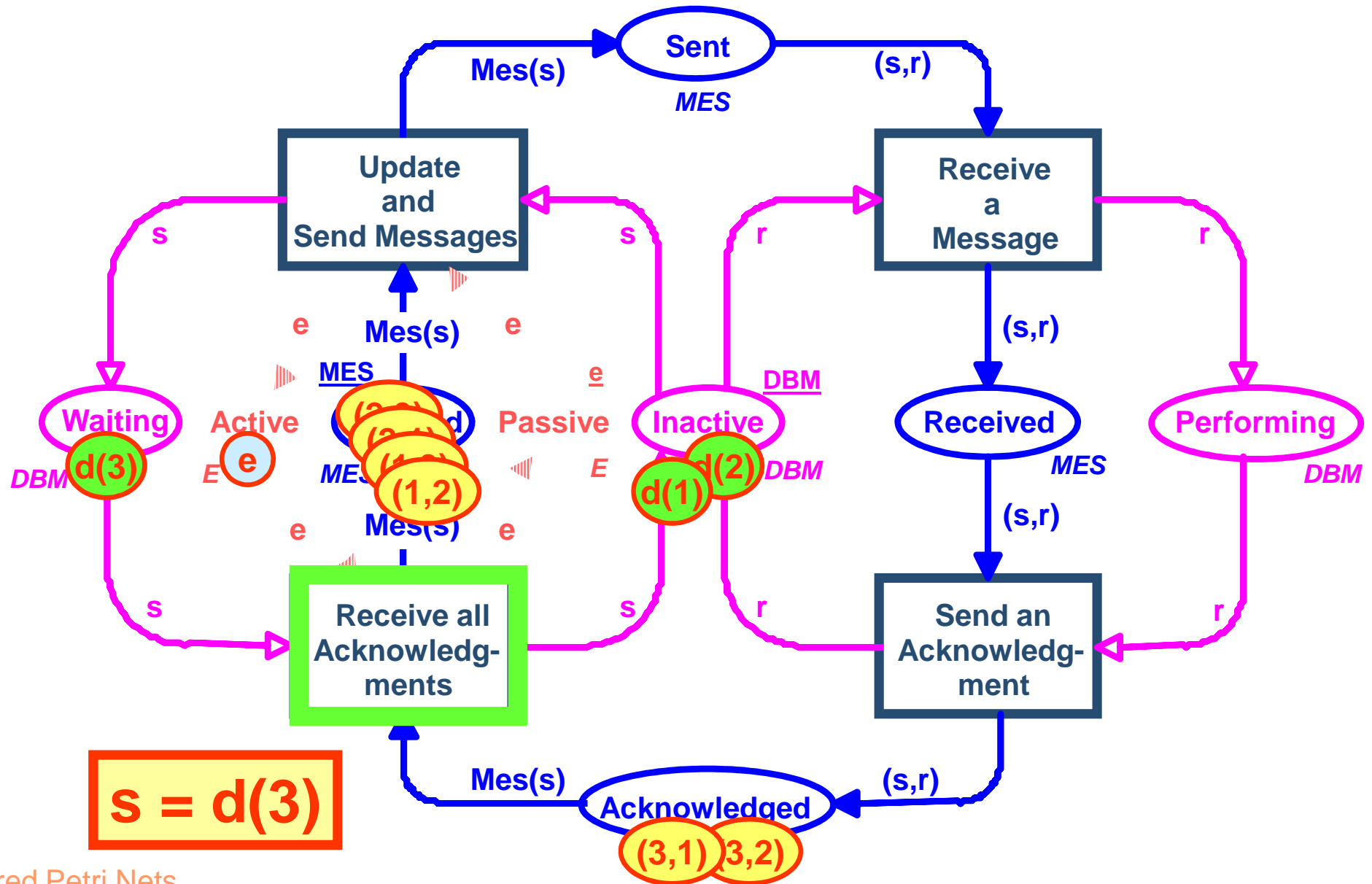
Elosztott adatbázis



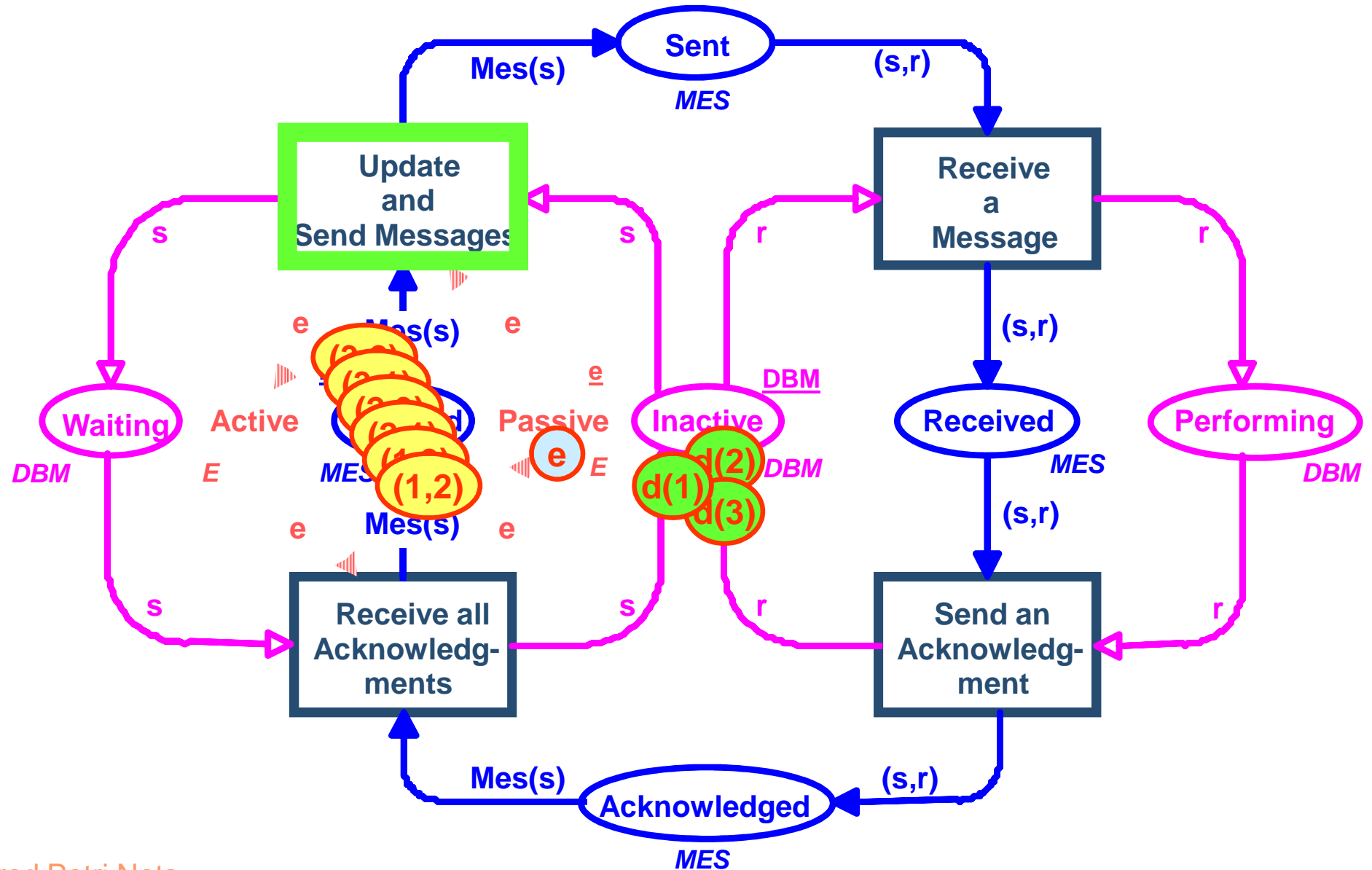
Elosztott adatbázis



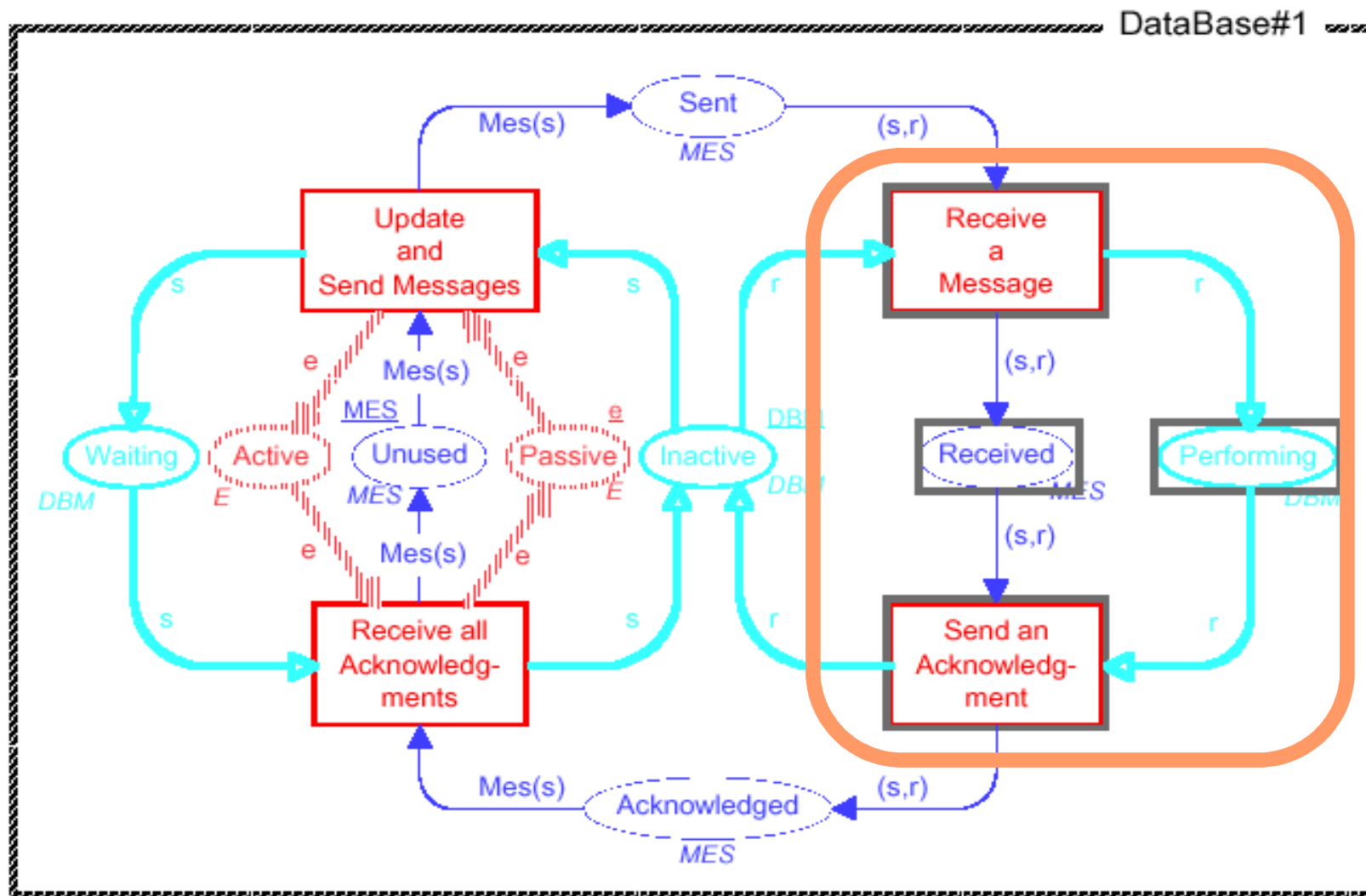
Elosztott adatbázis



Elosztott adatbázis



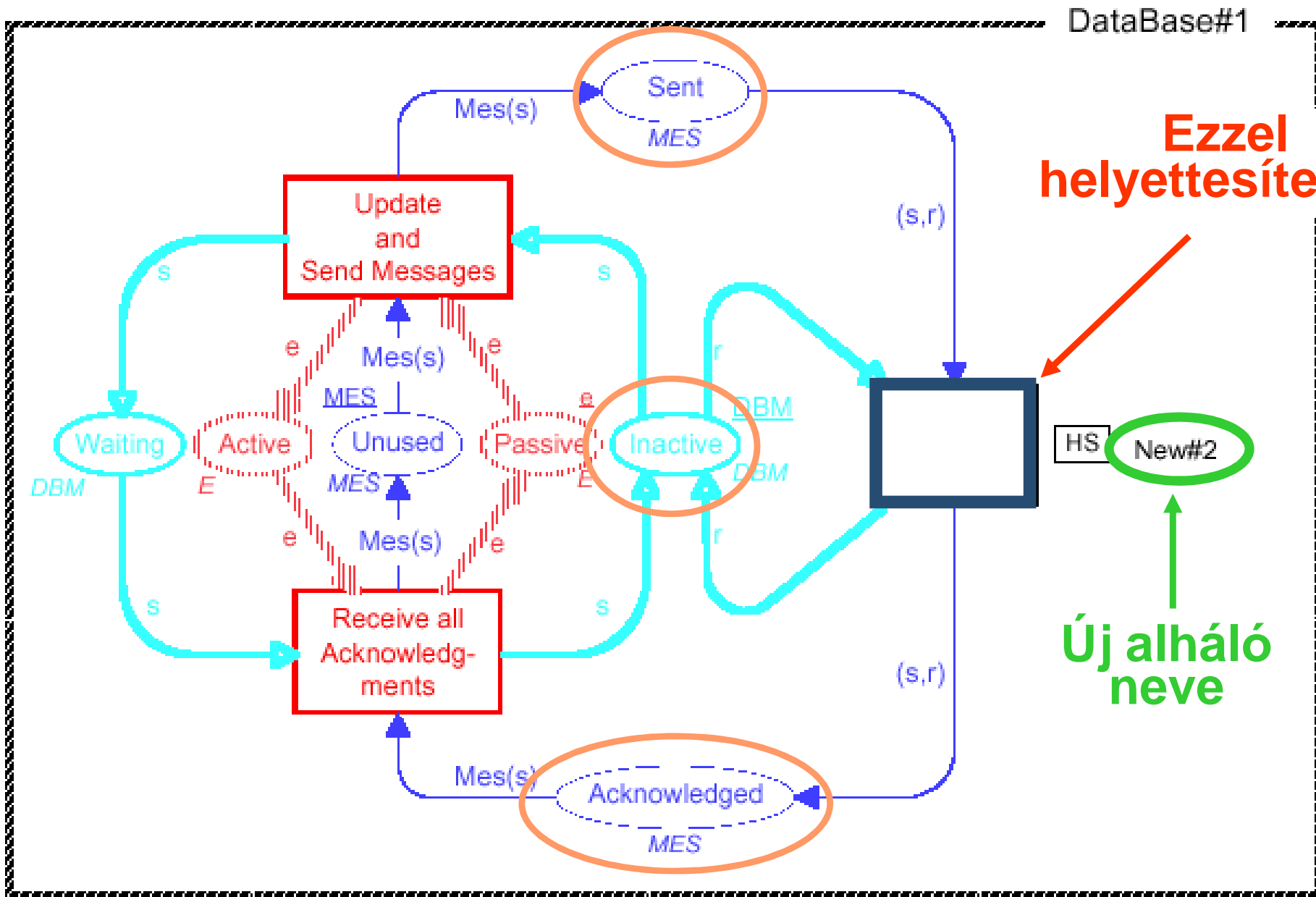
Hierarchikus modellek támogatása



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

Kompakt nézet

Sockets (interfész)



Ezzel helyettesítettük

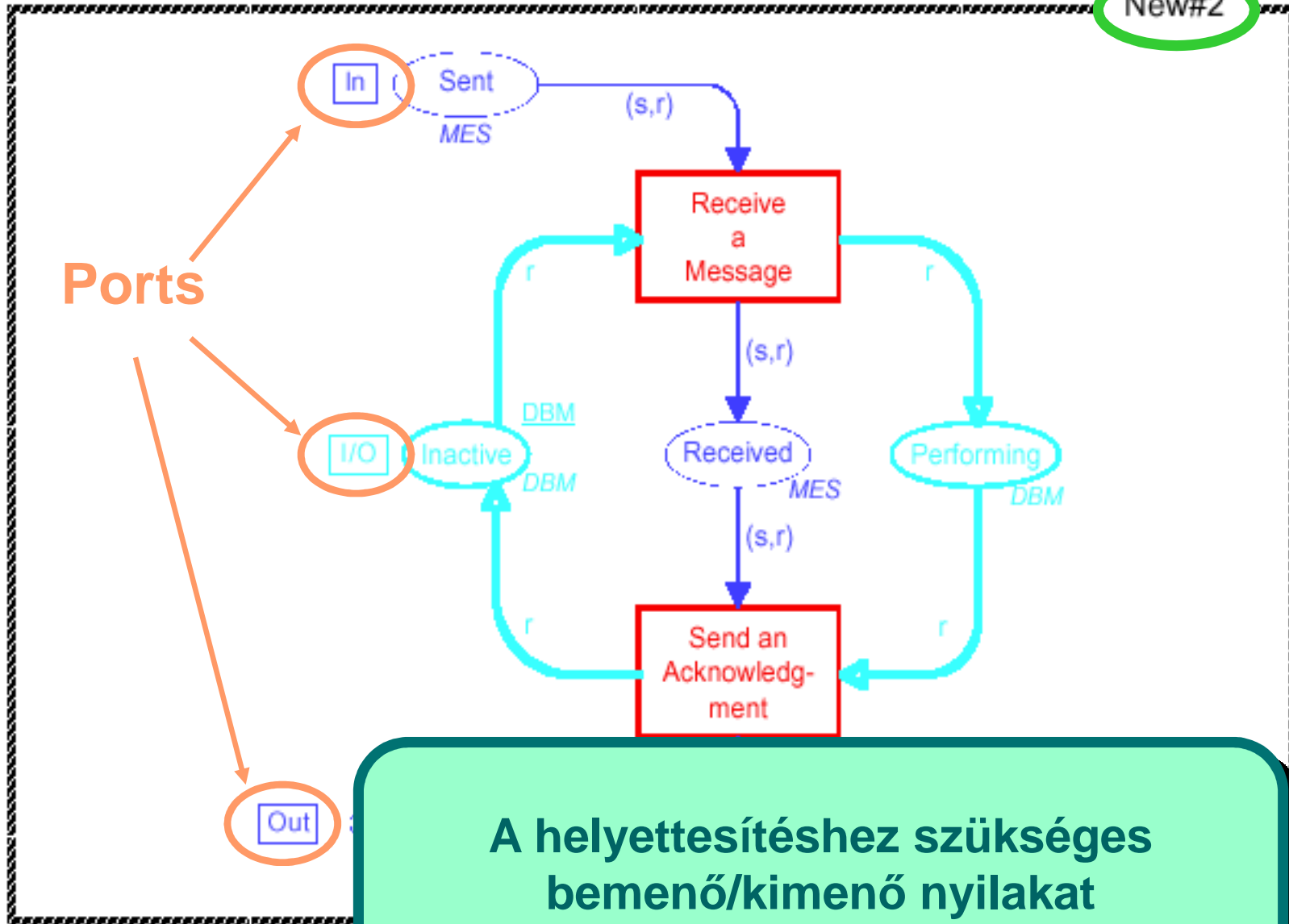
Új alháló neve

Kibontott nézet

Az alháló neve



New#2



A helyettesítéshez szükséges bemenő/kimenő nyilakat *automatikusan* berajzolja CPN editor.

Hely invariánsok

Place \rightarrow M(Place)

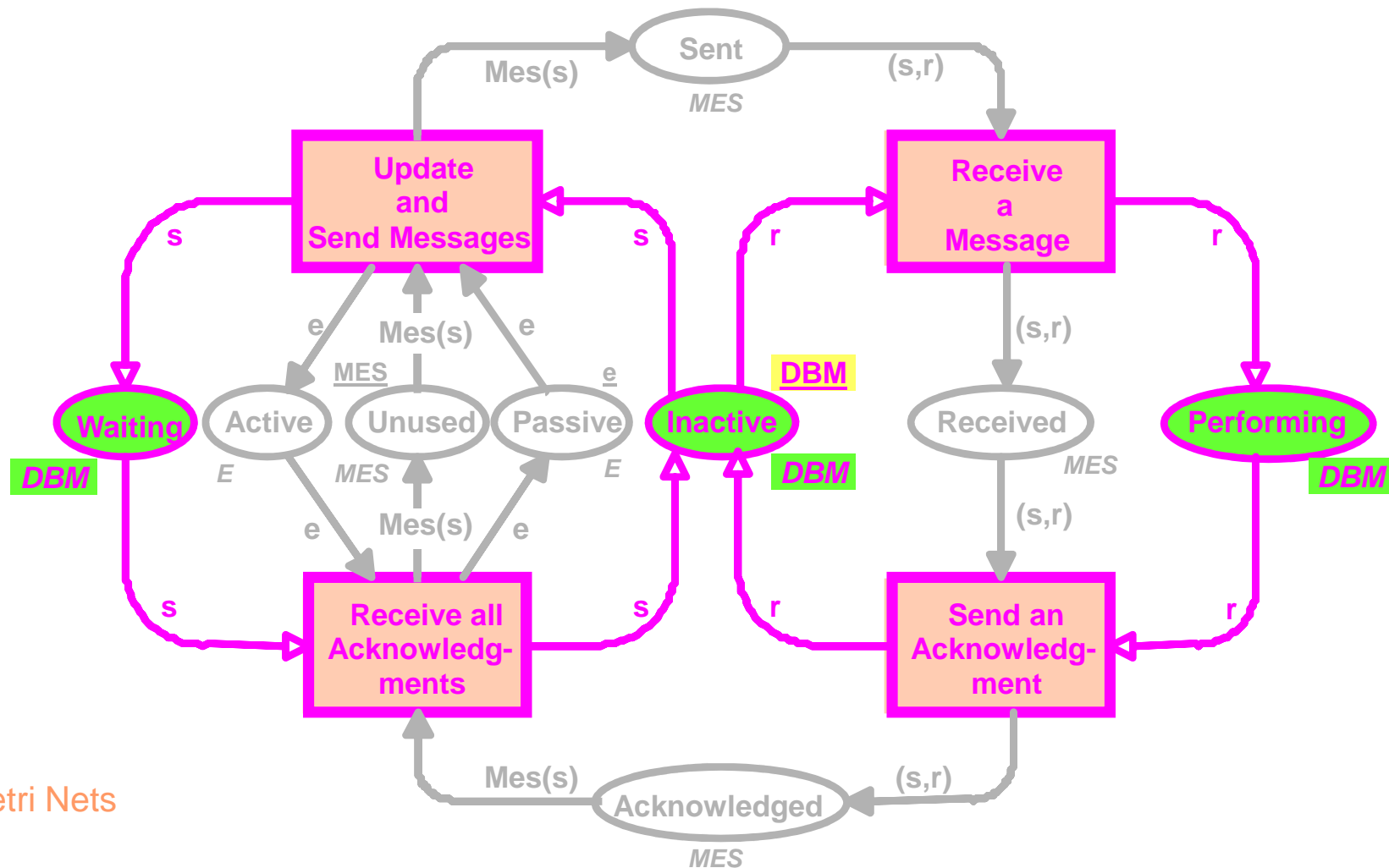
- ◆ $\text{Waiting} + \text{Inactive} + \text{Performing} = \text{DBM}$
- ◆ $\text{Unused} + \text{Sent} + \text{Receive} + \text{Acknowledged} = \text{MES}$
- ◆ $\text{Active} + \text{Passive} = \text{E}$
- ◆ $\text{Rec}(\text{Received}) = \text{Performing}$
- ◆ $\text{Mes}(\text{Waiting}) = \text{Sent} + \text{Received} + \text{Acknowledge}$
- ◆ $\text{Ign}(\text{Waiting}) = \text{Active}$

Ezek *lineáris kombinációival* gyárthatunk többet.

- ◆ $\text{Ign}(\text{Waiting}) + \text{Passive} = \text{E}$

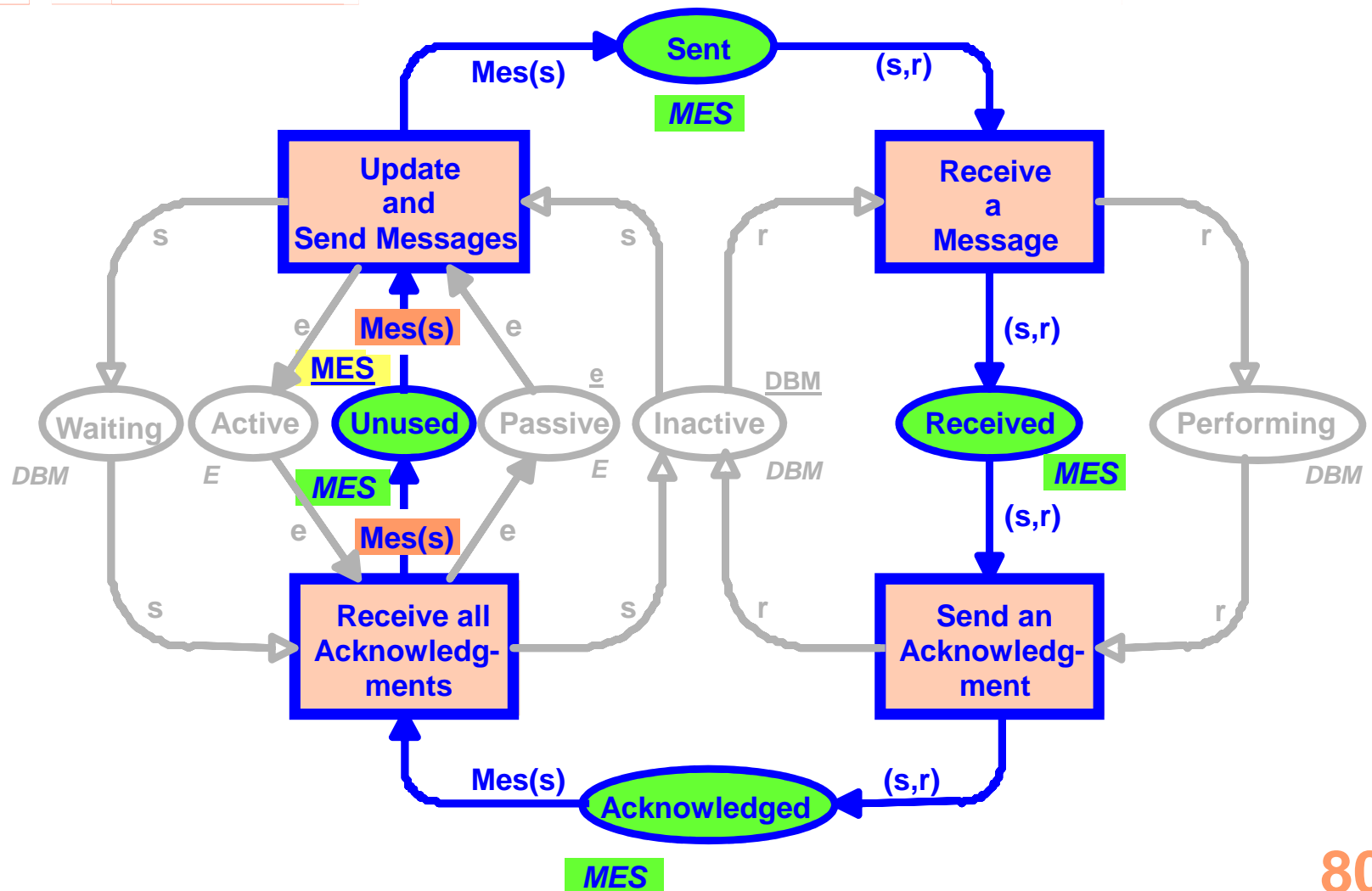
Adatbázis kezelők

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



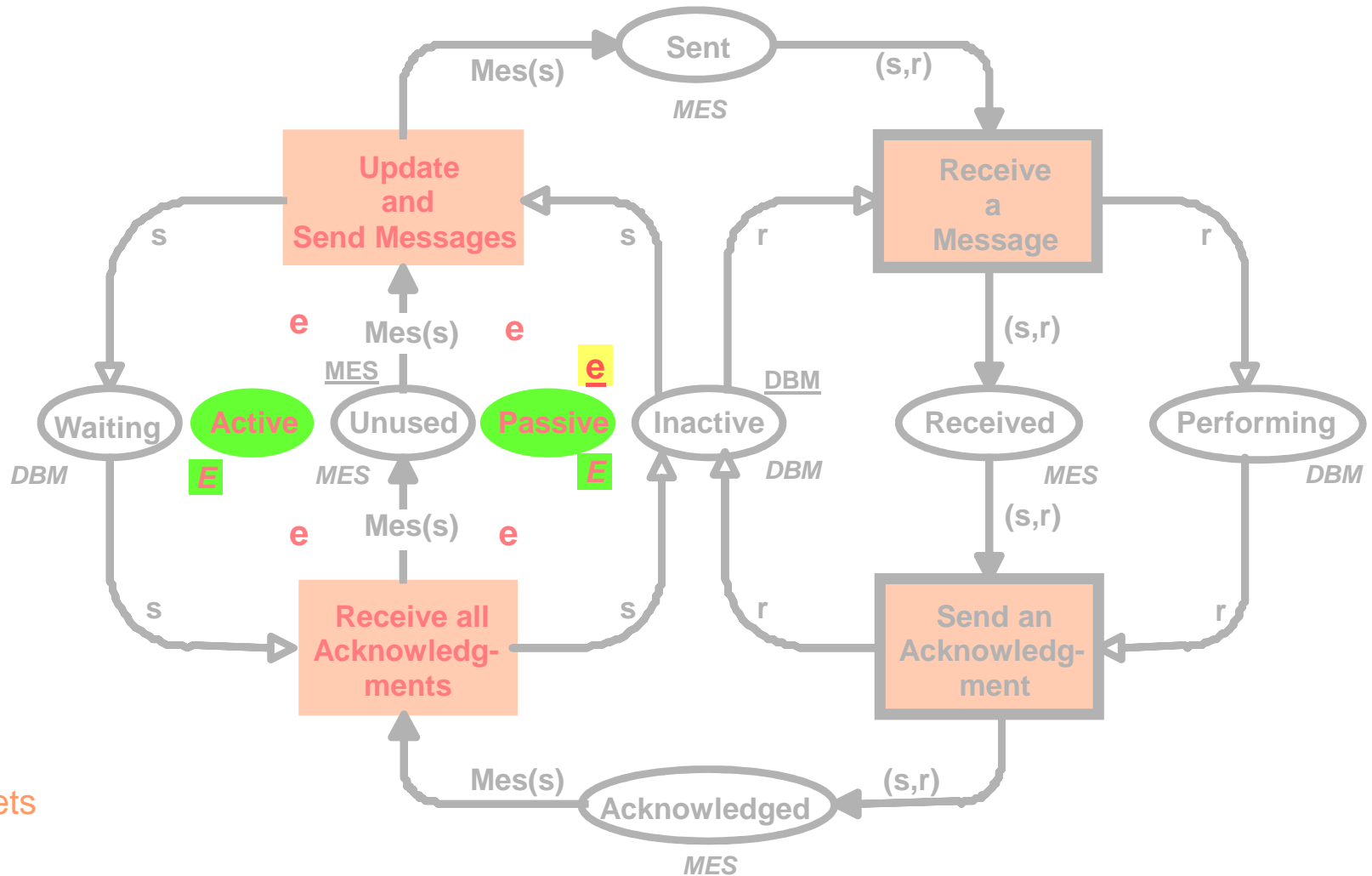
Üzenet bufferek

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



Kölcsönös kizárás

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



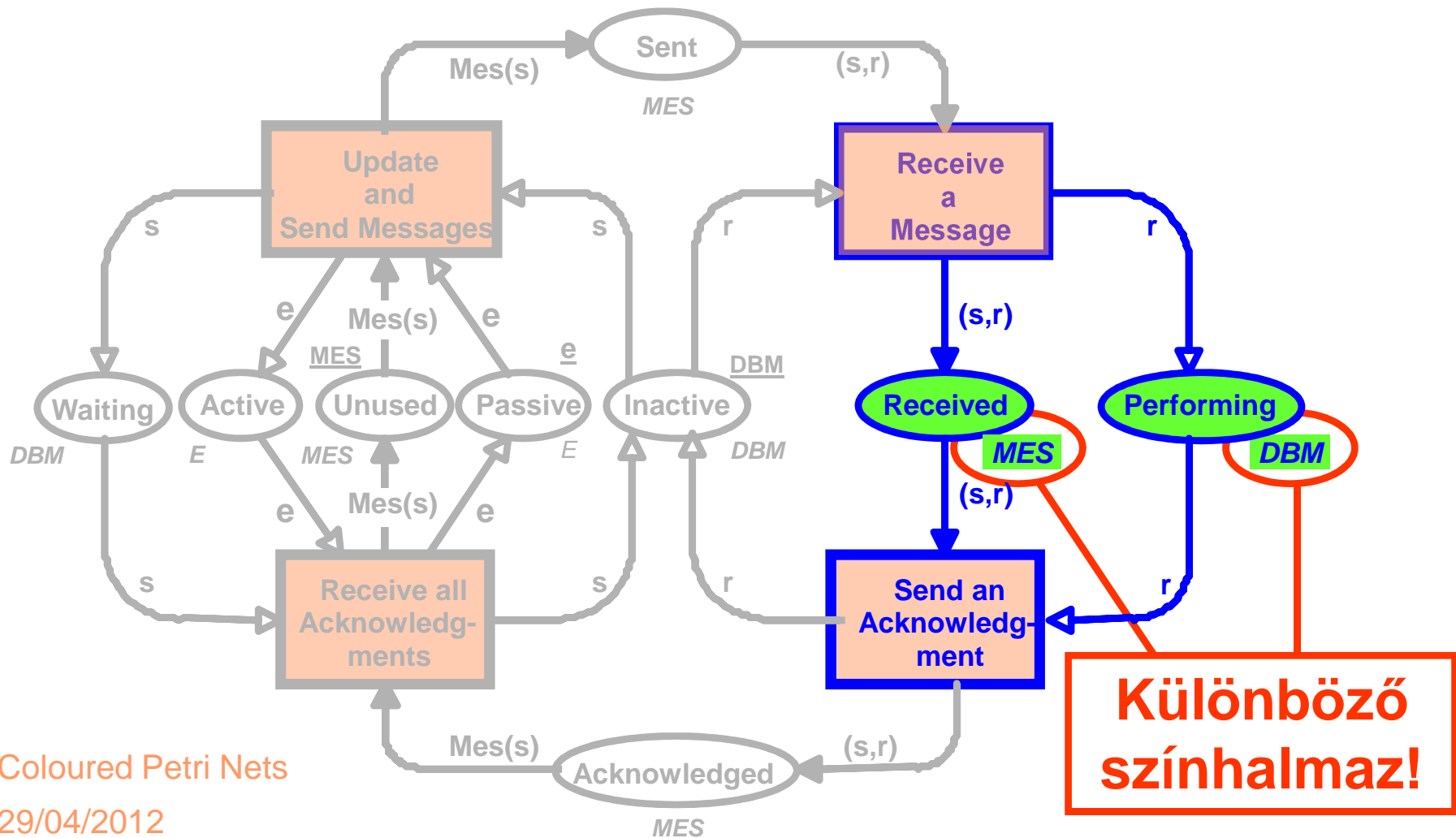
Fogadott üzenetek

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

Súly függvény

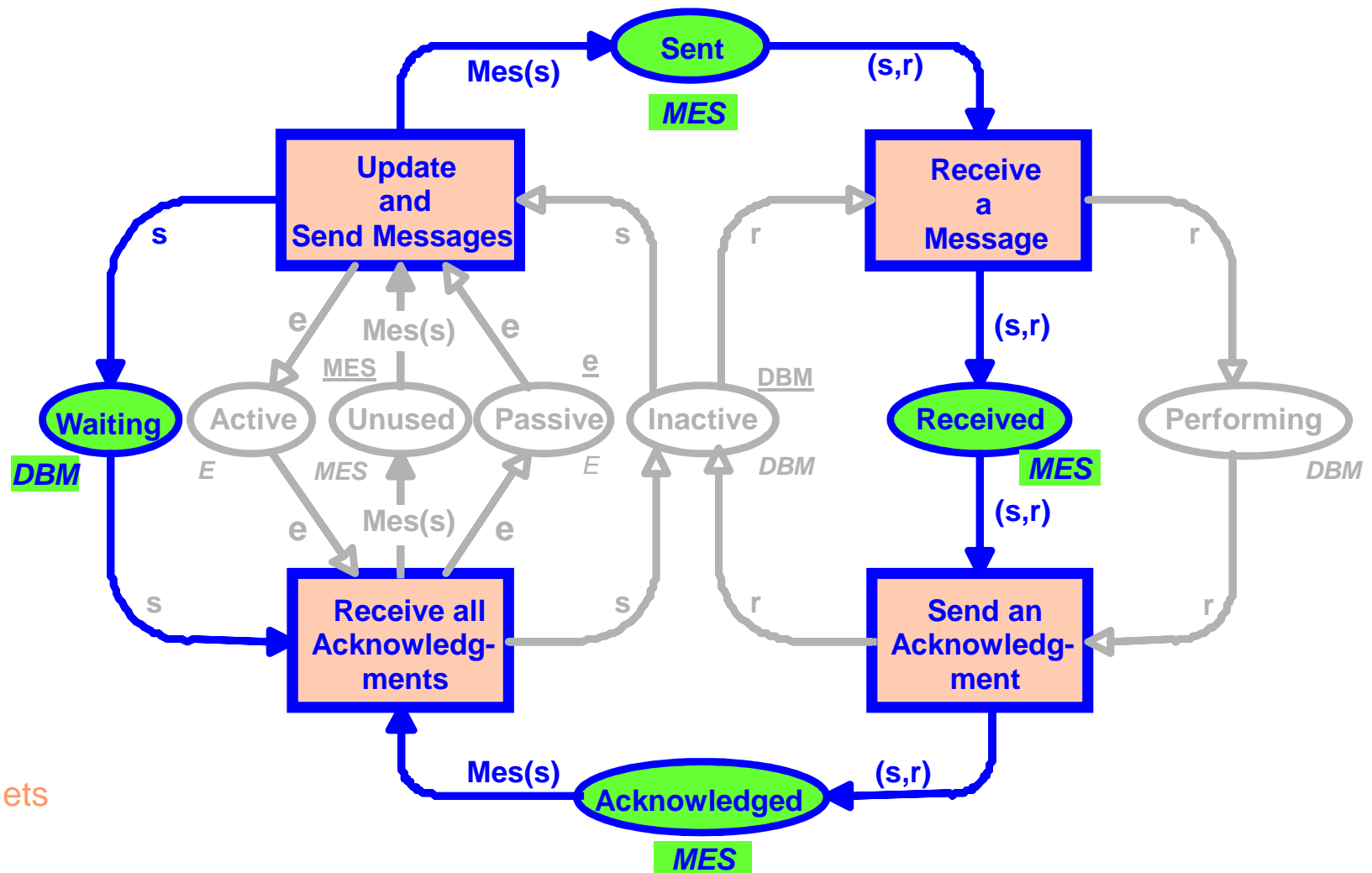
$$\text{Rec}(M(\text{Received})) = M(\text{Performing})$$

MES \rightarrow DBM



Használt üzenetek

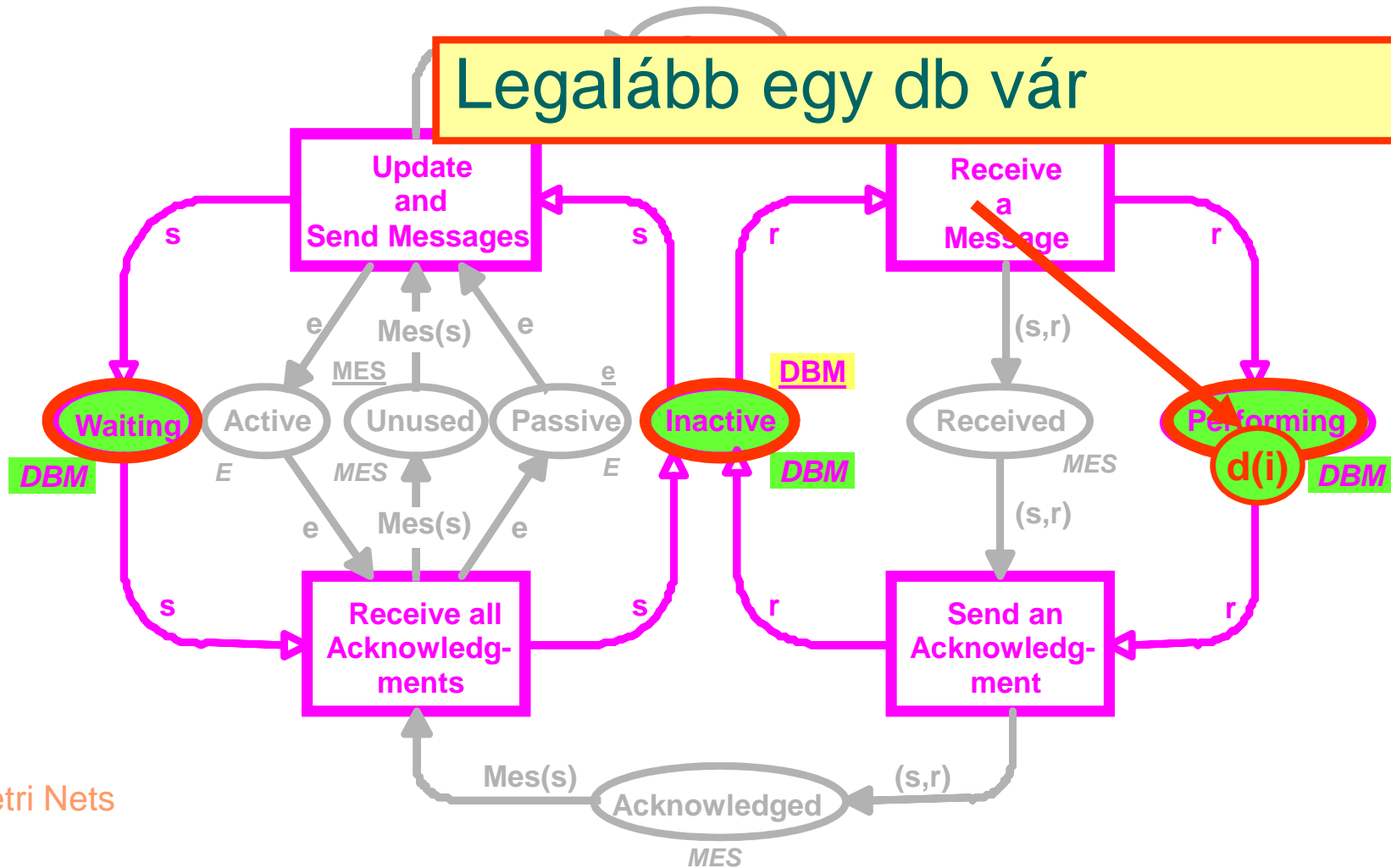
$$\text{Mes}(\text{Waiting}) = M(\text{Sent}) + M(\text{Received}) + M(\text{Acknowledged})$$



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

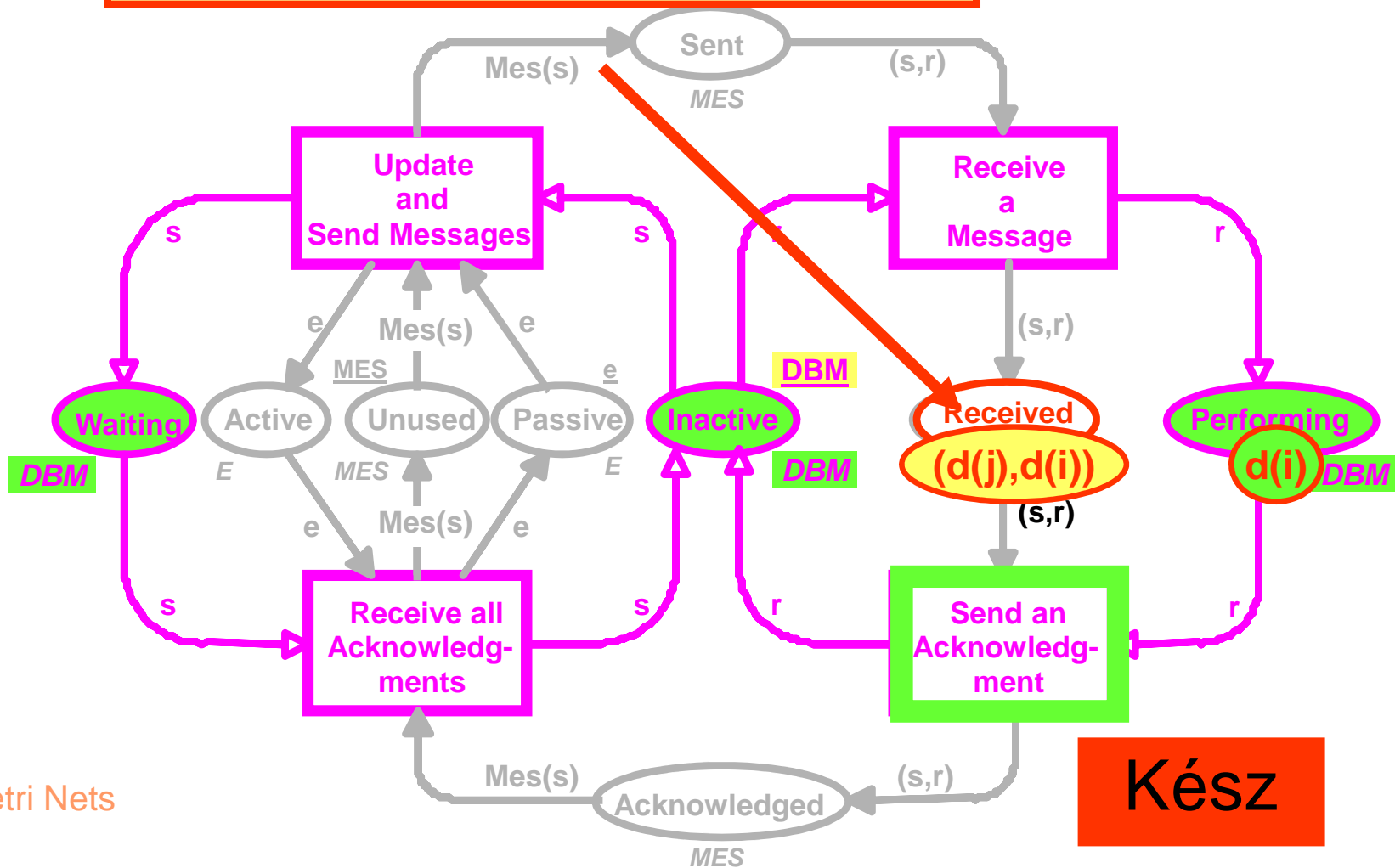
Minden adatkezelő:

Legalább egy db vár



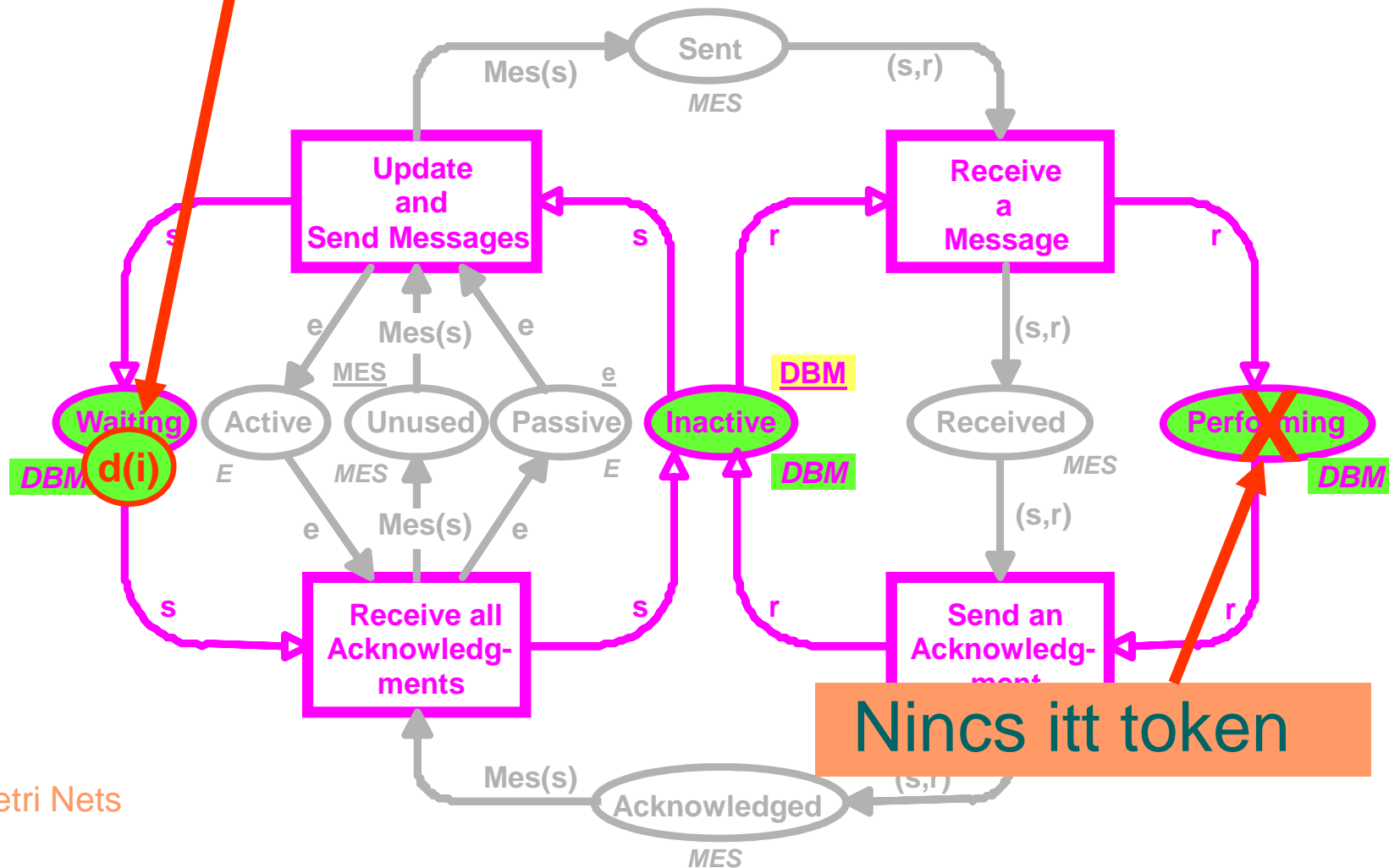
$$\text{Rec}(M(\text{Received})) = M(\text{Performing})$$

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



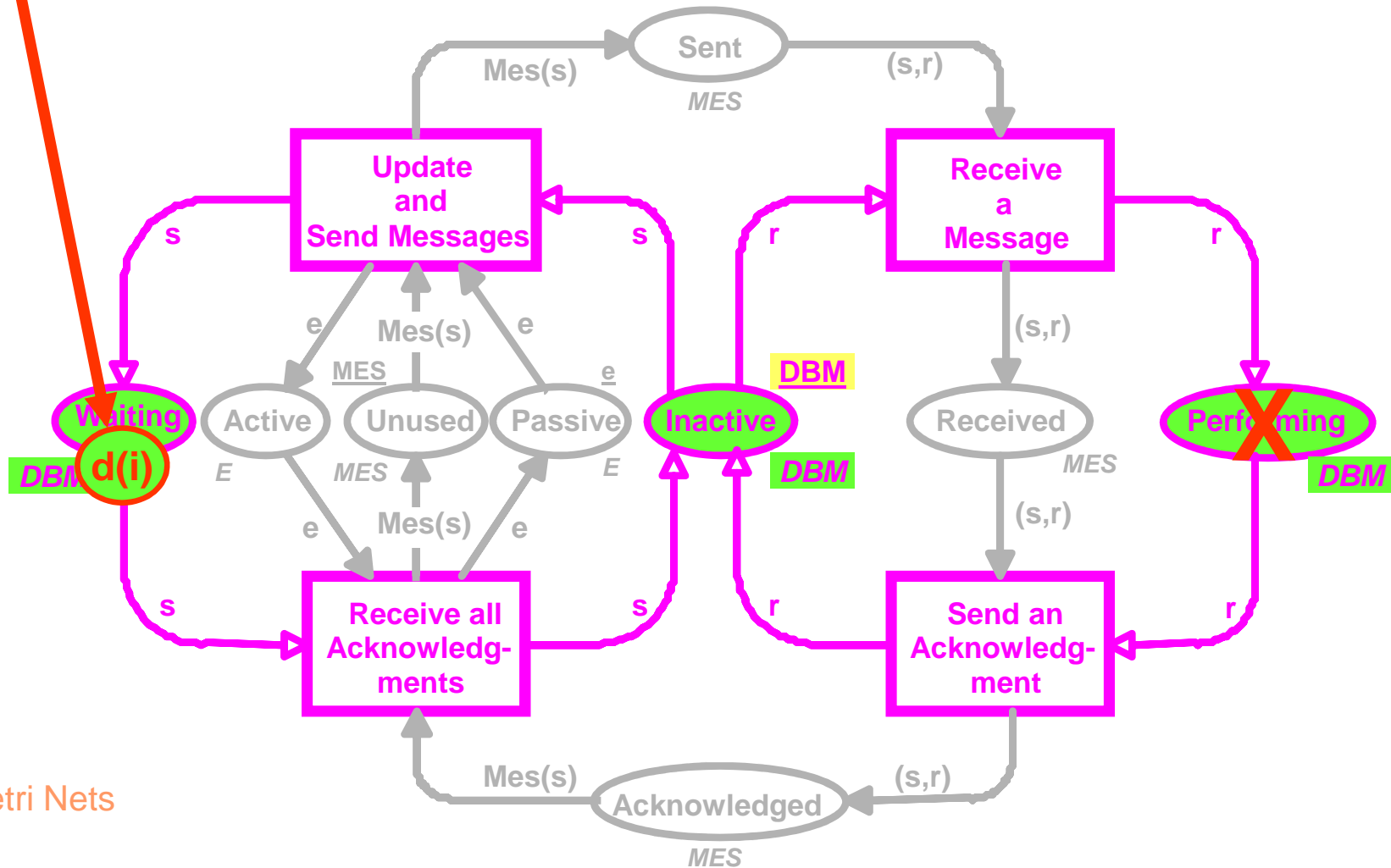
Kész

Most azt tesszük fel, hogy legalább egy kezelő VÁR



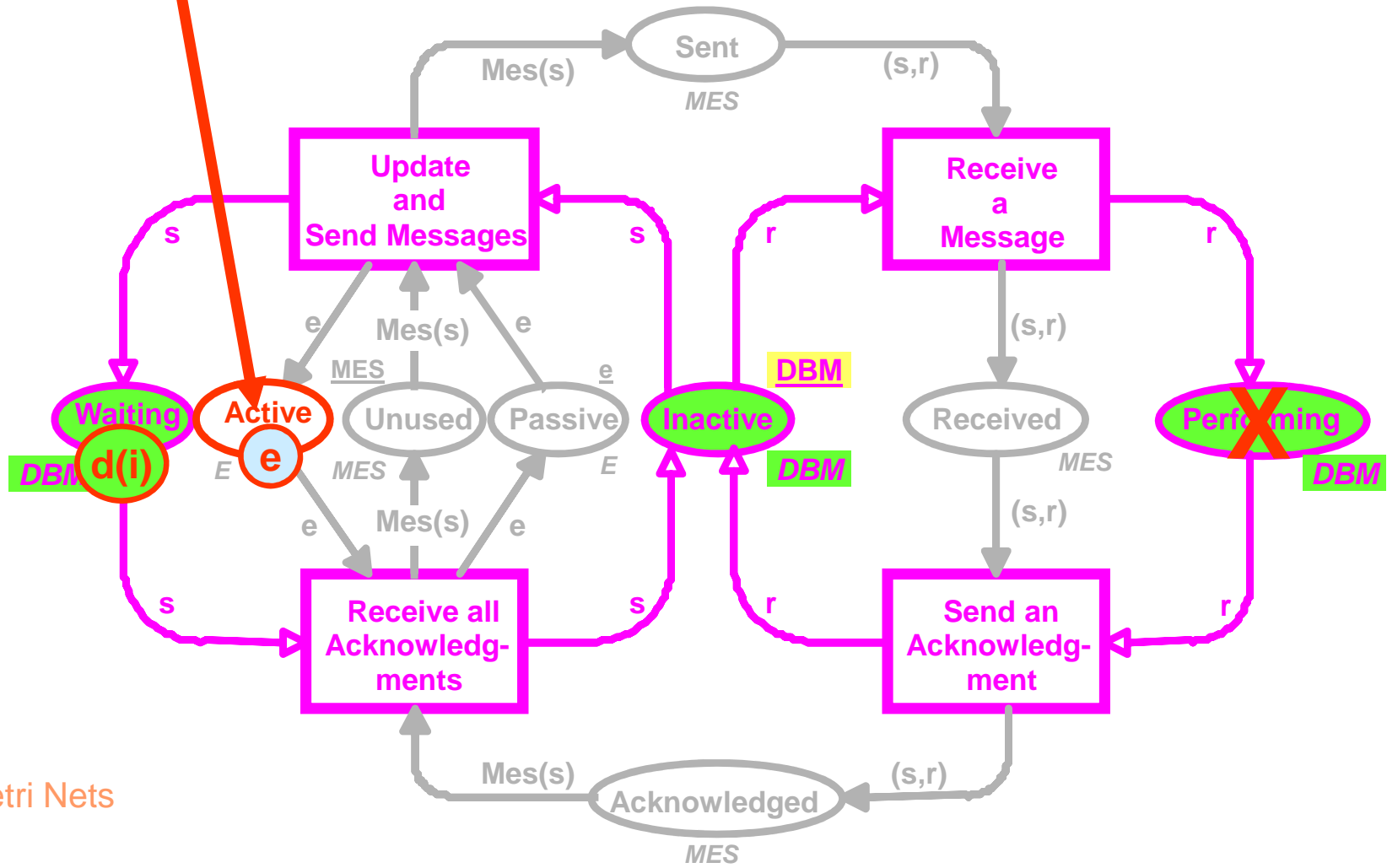
Ign(Waiting) + Passive = E

Pontosan egy token várakozik



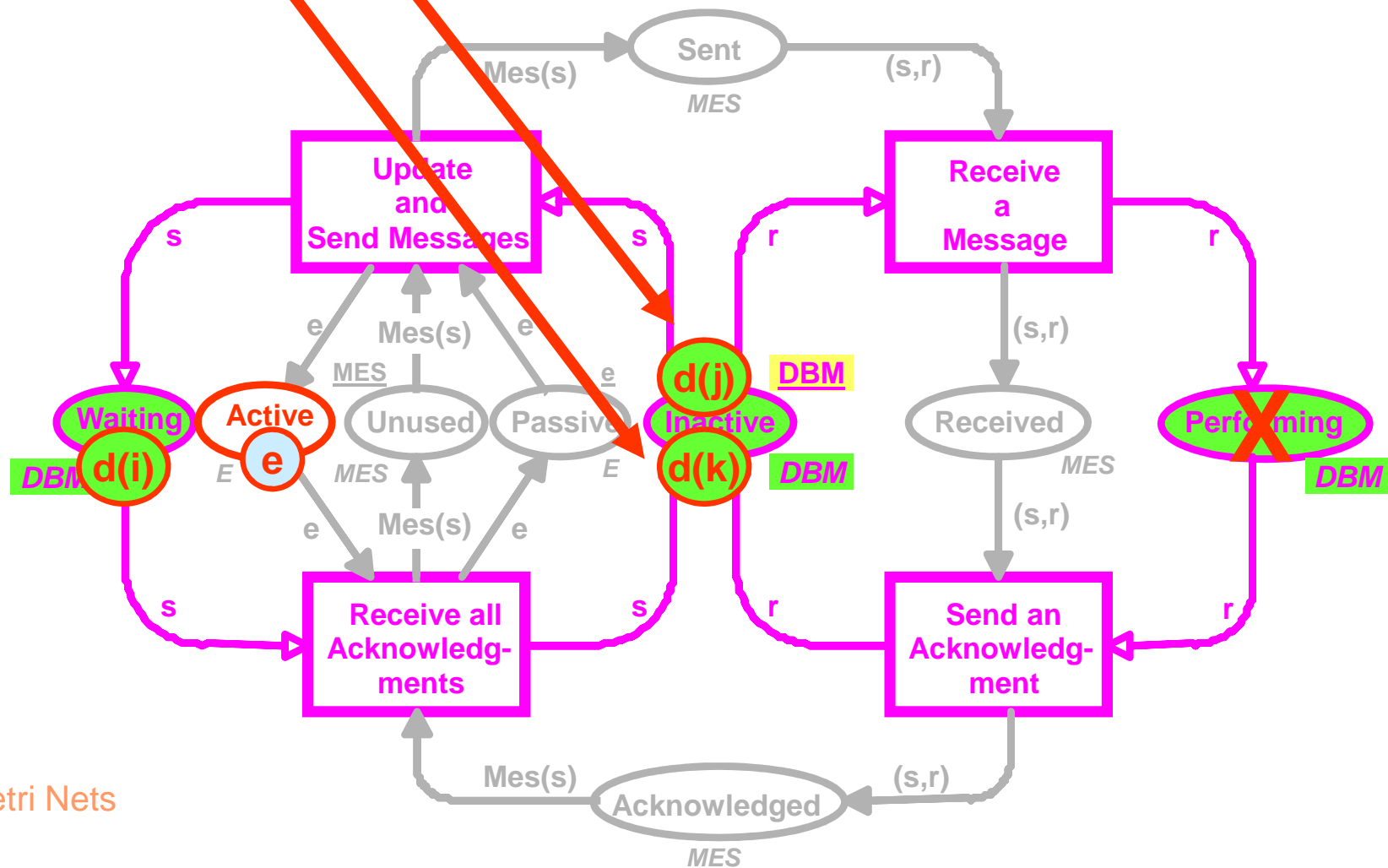
Ign(Waiting) = Active

Pontosan egy token aktív



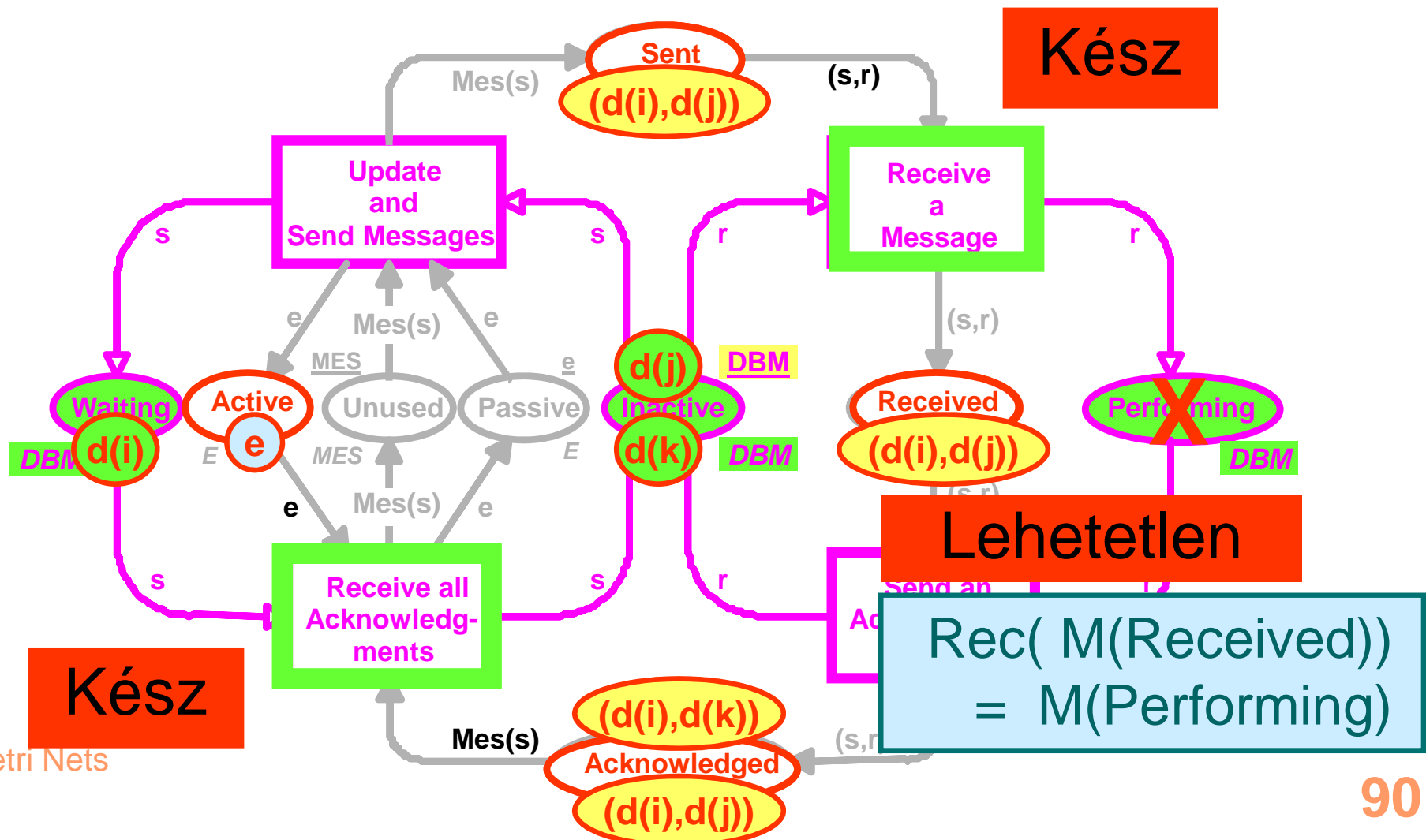
$$M(\text{Waiting}) + M(\text{Inactive}) + M(\text{Performing}) = \text{DBM}$$

Nincs több aktív adatbázis kezelő



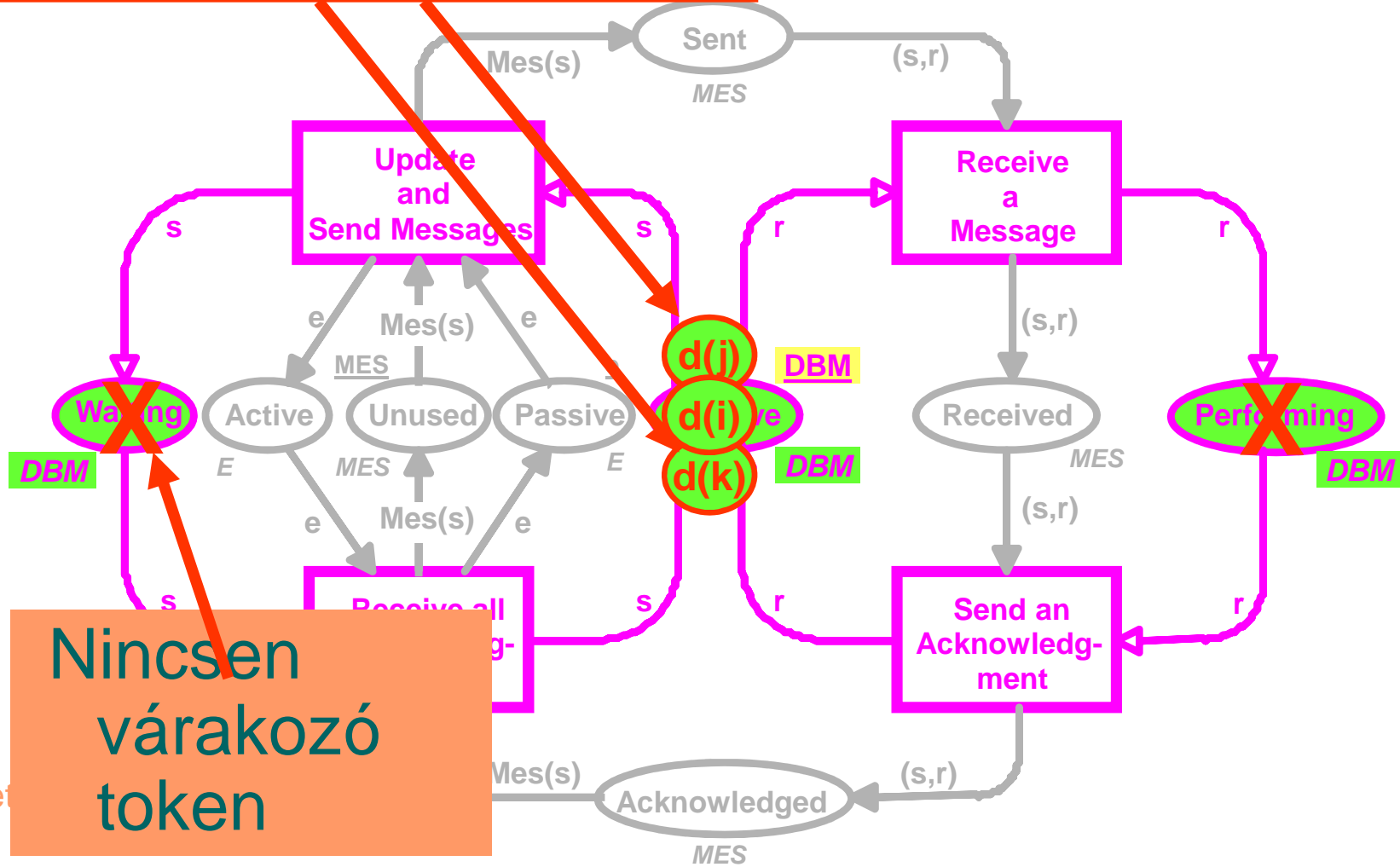
$$\text{Mes}(\text{Waiting}) = \text{M}(\text{Sent}) + \text{M}(\text{Received}) + \text{M}(\text{Acknowledged})$$

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



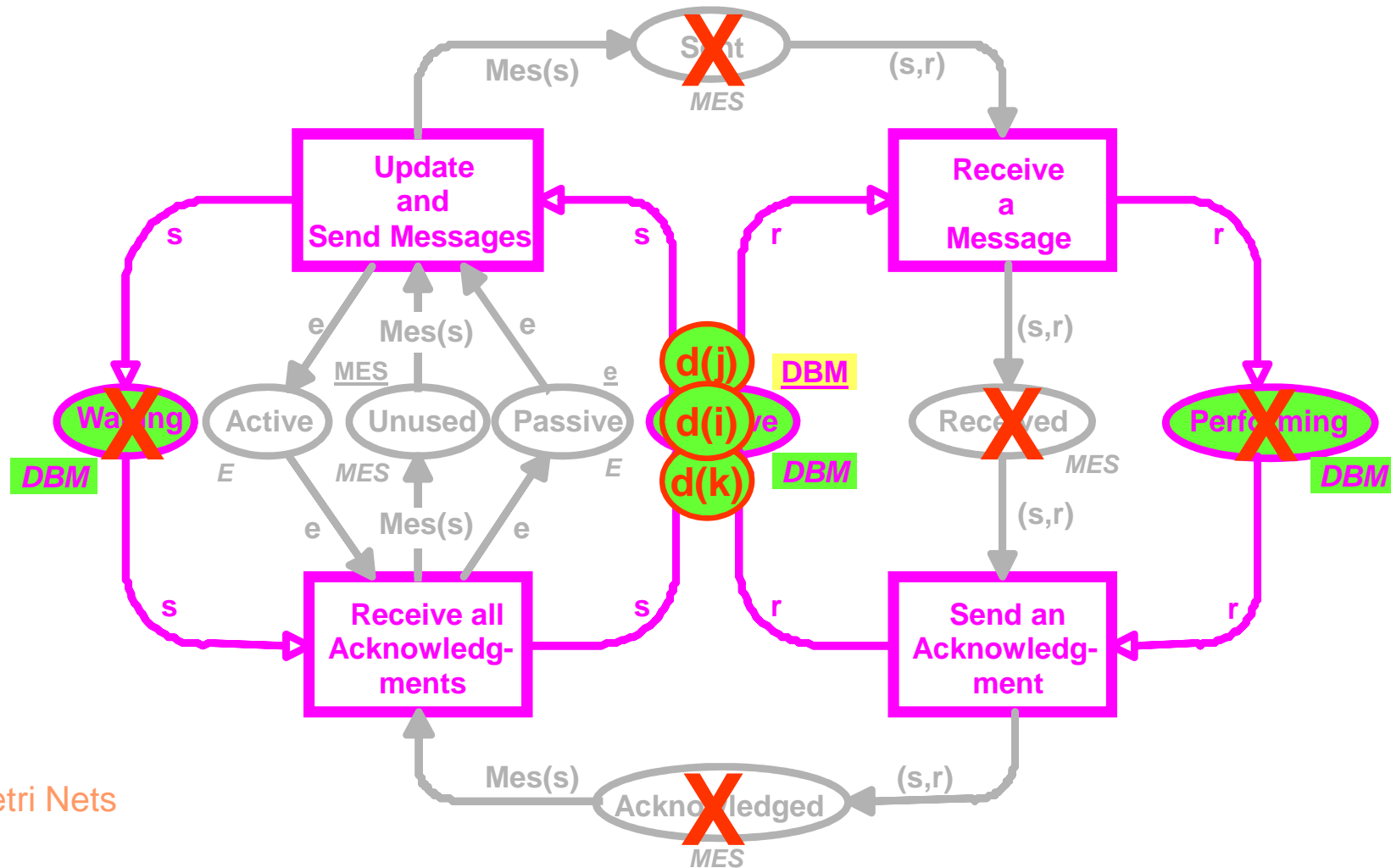
M(Waiting) + M (Inactive) + M(Performing) = DBM

Nincsen aktív adatbázis kezelő



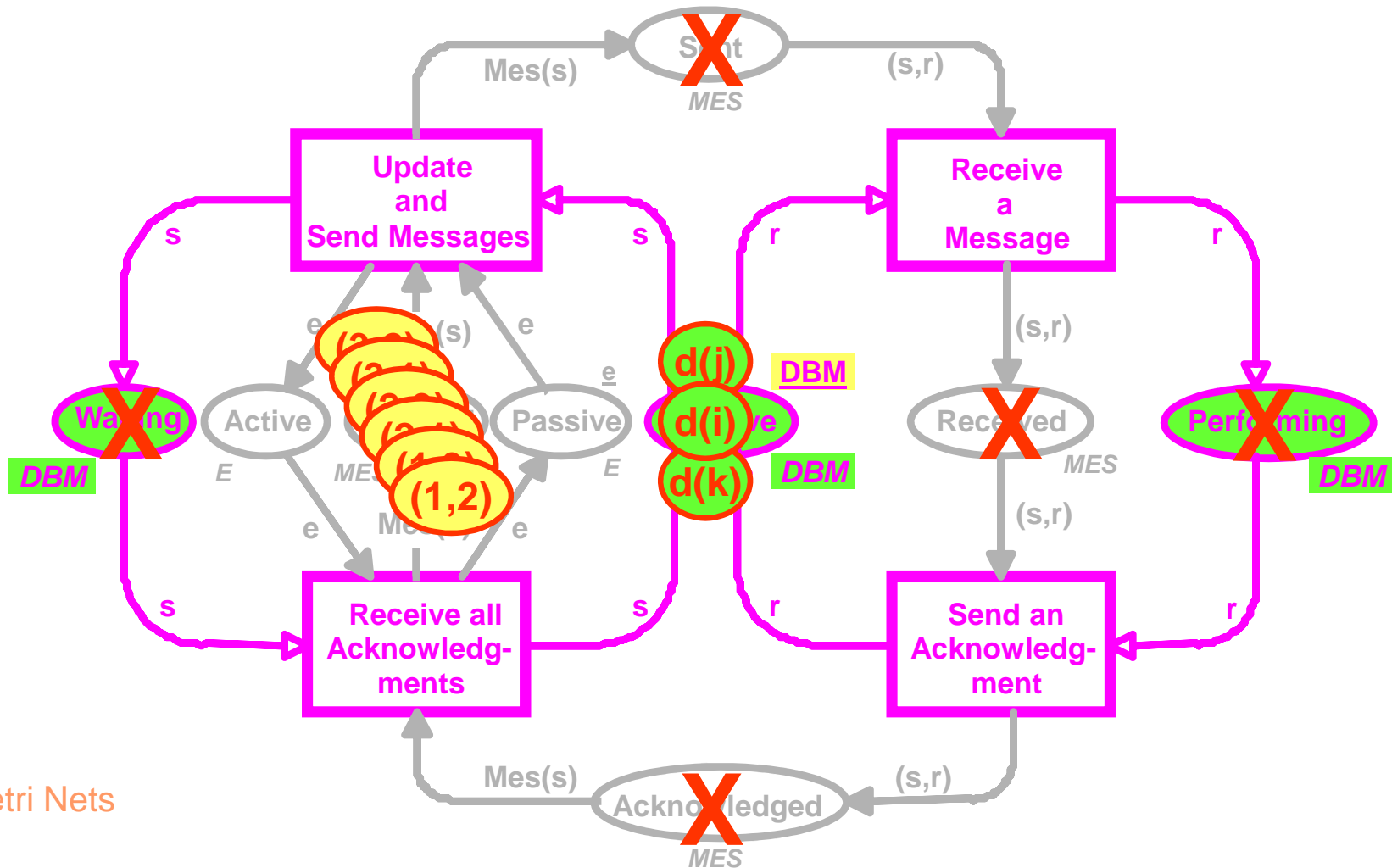
$$\text{Mes}(\text{Waiting}) = \text{M}(\text{Sent}) + \text{M}(\text{Received}) + \text{M}(\text{Acknowledged})$$

Nincsen elküldött, várakozó és nyugta token



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

Az üzenetbufferek üresek:



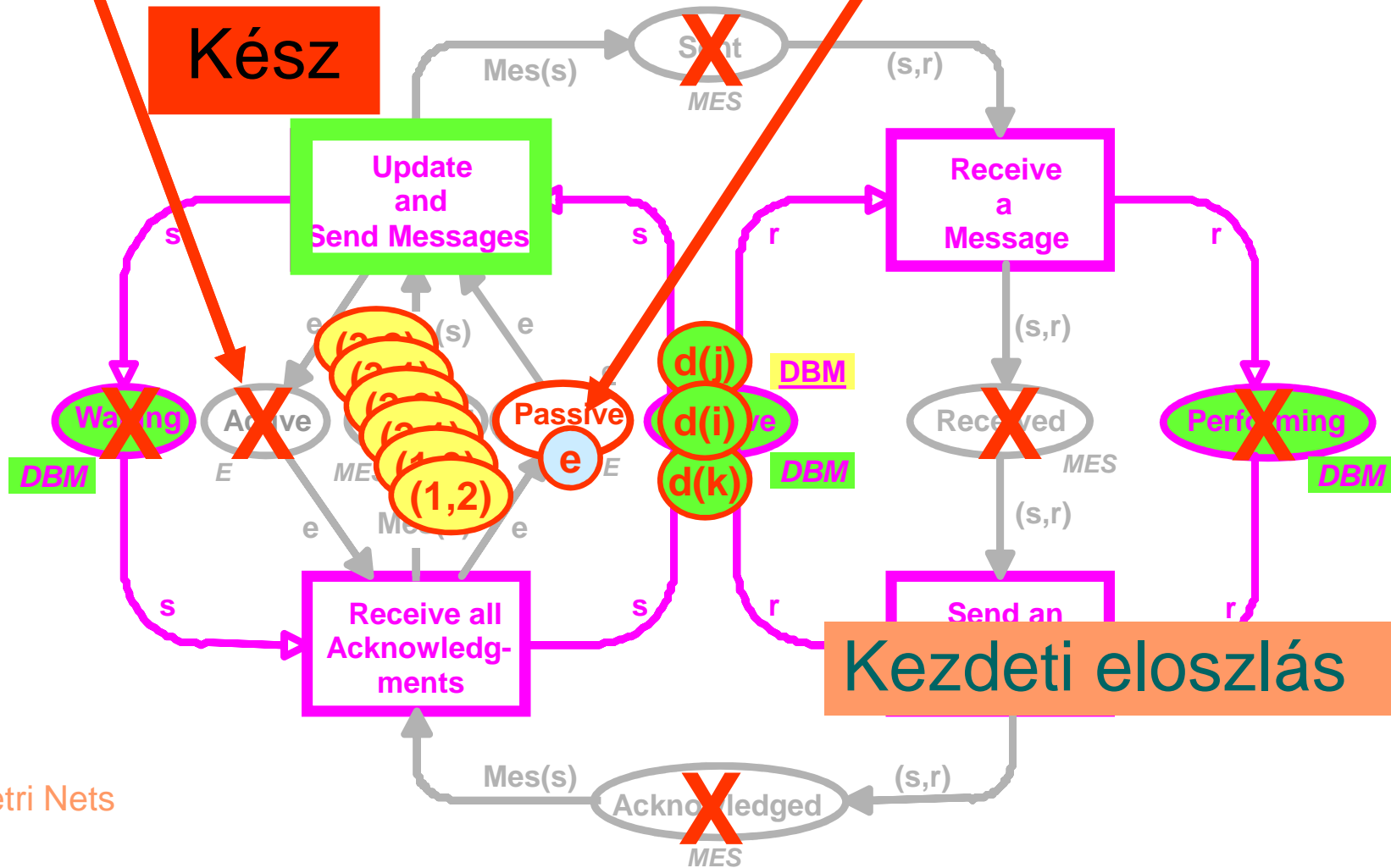
Ign(Waiting) = Active

Active + Passive = E

Nincsen aktív token

Egy e-token Passive

Kész



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.

További információk CPN-ről

- ◆ Az alábbi *WWW oldalakon* nagyon sok hasznos információt találhatunk CPN-ről és a tool-uktól:
<http://www.daimi.au.dk/CPnets/>
- ◆ *Bevezetés a CPN-be*, itt rengeteg példát találhatunk.
- ◆ Manual *Design/CPN-hez* és *CPN Tools-hoz*.
 - Ezek az eszközök még *szabadon használhatóak* ipari célokra is.
- ◆ Több mint 50 publikáció található amelyek különböző *ipari megoldásokat* mutat be.
- ◆ Létezik egy 3-kötetes *CPN könyv*.