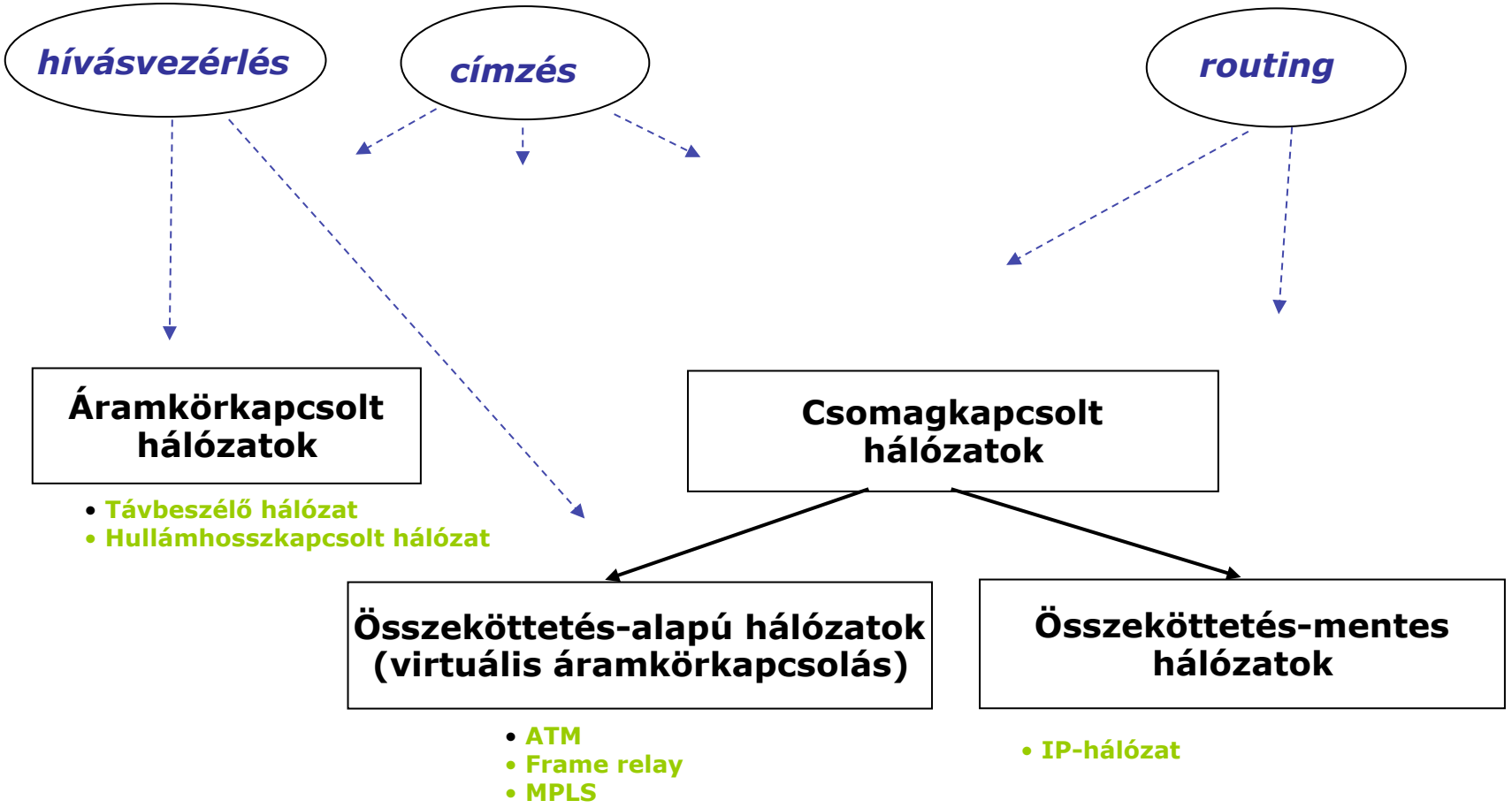
ROUTING

Útvonalválasztás

Dr. Simon Vilmos
docens
BME Hálózati Rendszerek és Szolgáltatások Tanszék
svilmos@hit.bme.hu

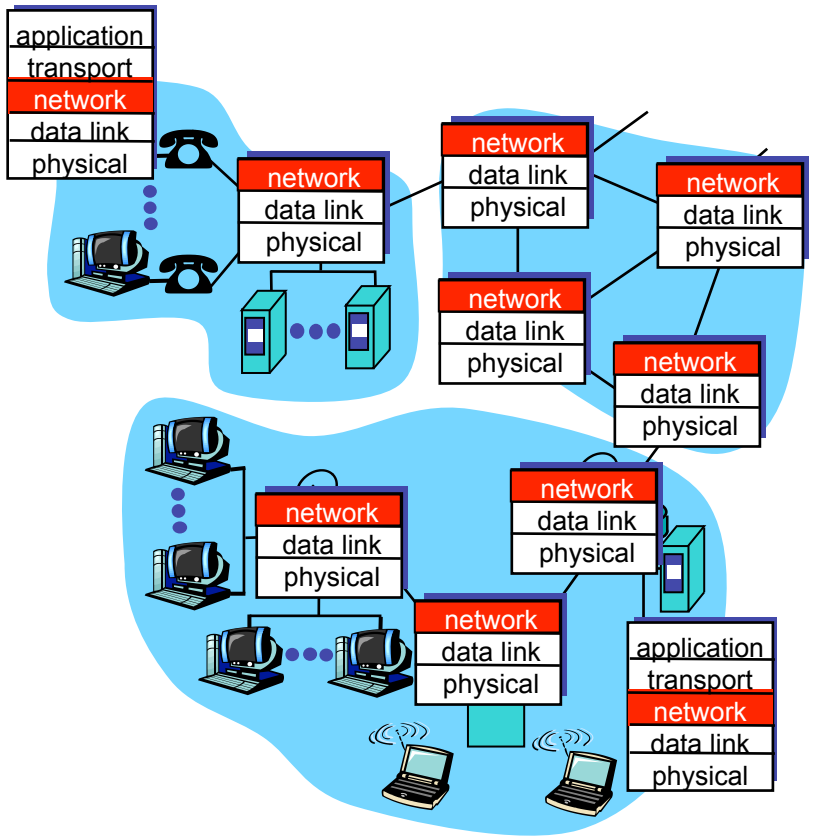
2016.október 26.

További nagyon fontos funkció a kapcsolt hálózatok működtetéséhez



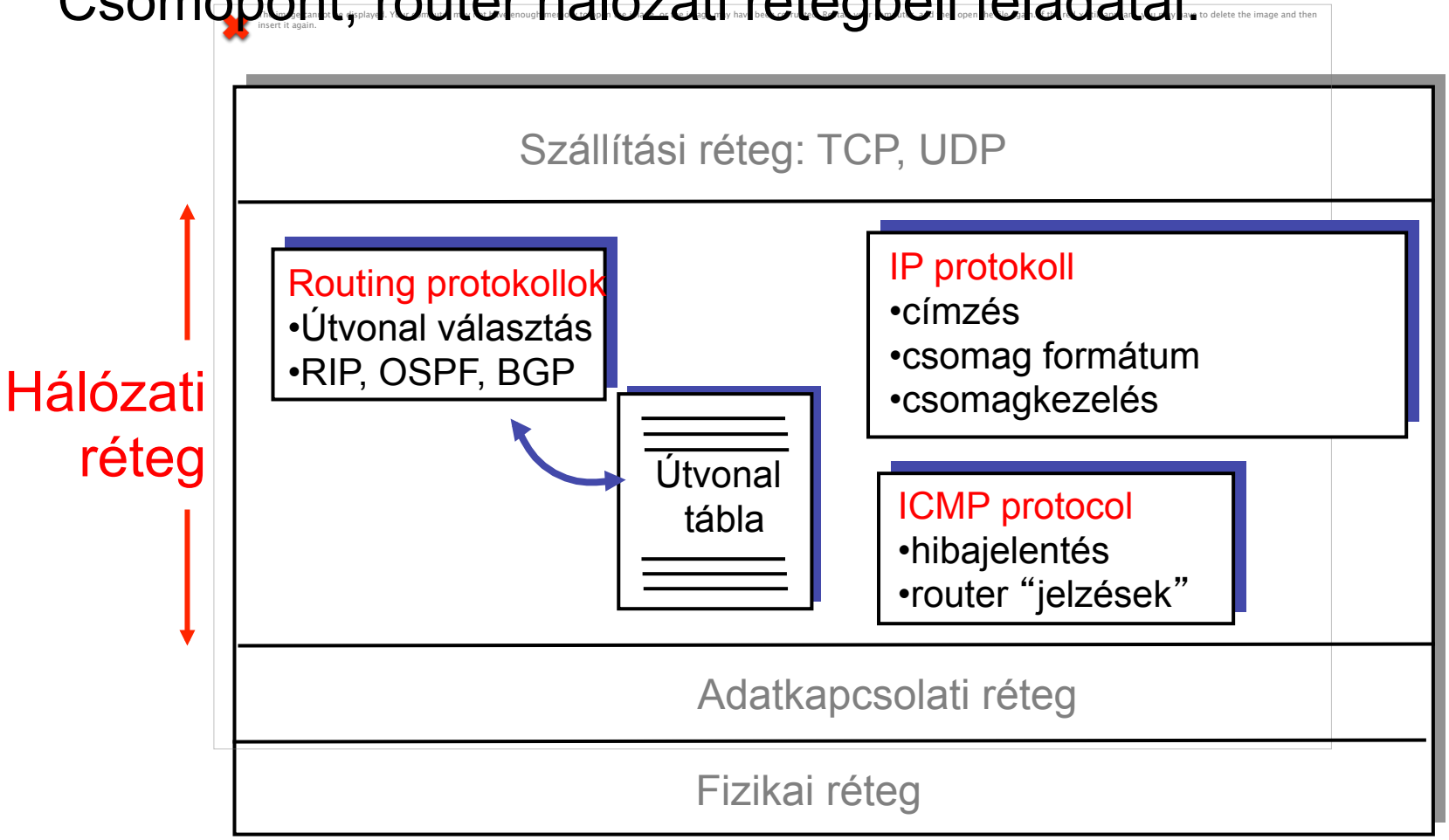
Hálózati réteg

- Szegmens átvitele a feladótól a címzettig
- Küldő oldalon: szegmensből csomag (IP fejléc kerül rá)
- Címzettnél: szállítási rétegnek kézbesíti a szegmenst
- Hálózati rétegbeli protokoll minden csomópontban (routerben)
- Router megvizsgálja az IP fejlécet, ez alapján dönt



Hálózati réteg feladatai

Csomópont, router hálózati rétegbeli feladatai:



Különbség bridging és routing között

- Bridging:
  - Adatkapcsolati rétegben MAC címek alapján
  - Nem tud különbséget tenni hálózatok között
  - Elárasztást használ: csak helyi hálózatokban

- Routing:
  - Hálózati rétegben, IP címek alapján
  - Skálázhatóbb megoldás
  - Erről lesz szó a továbbiakban!

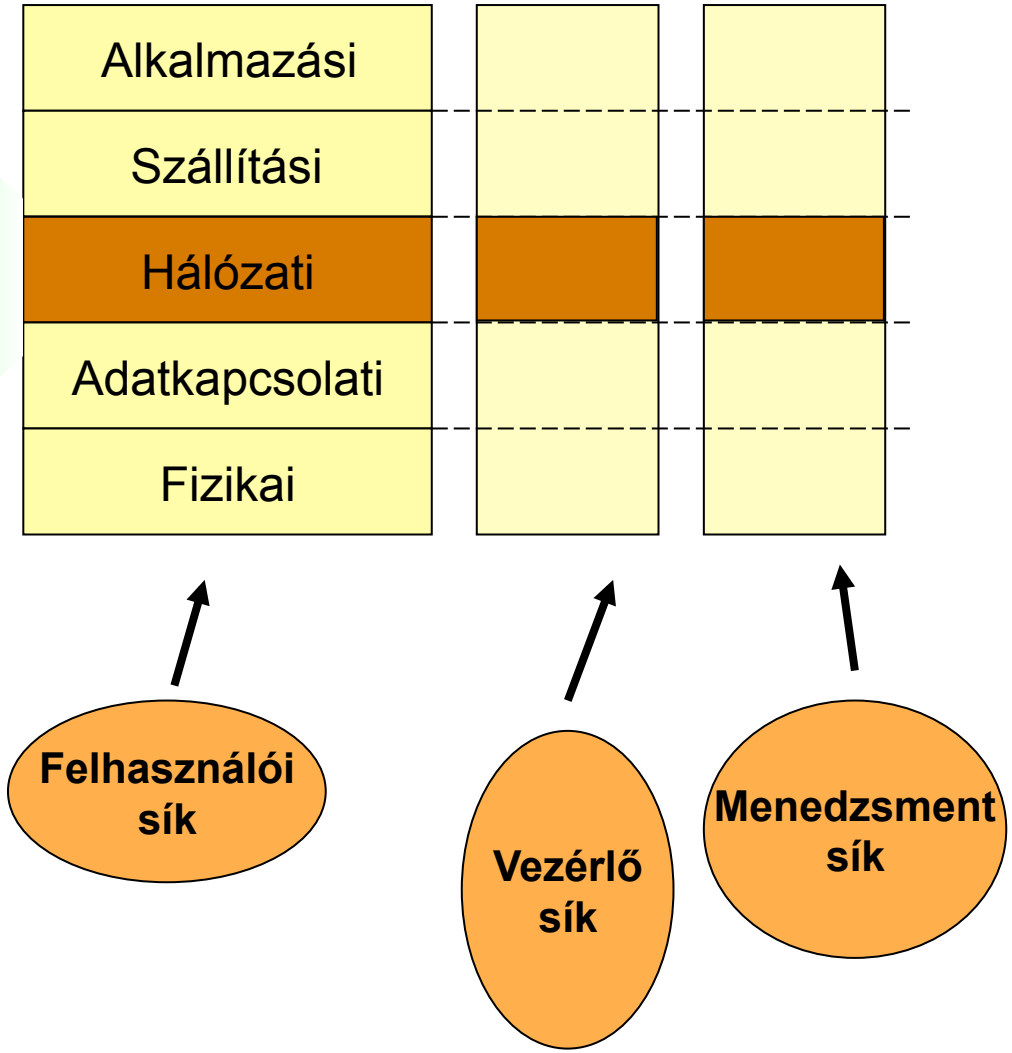
Elv, módszer, protokoll

Elv:

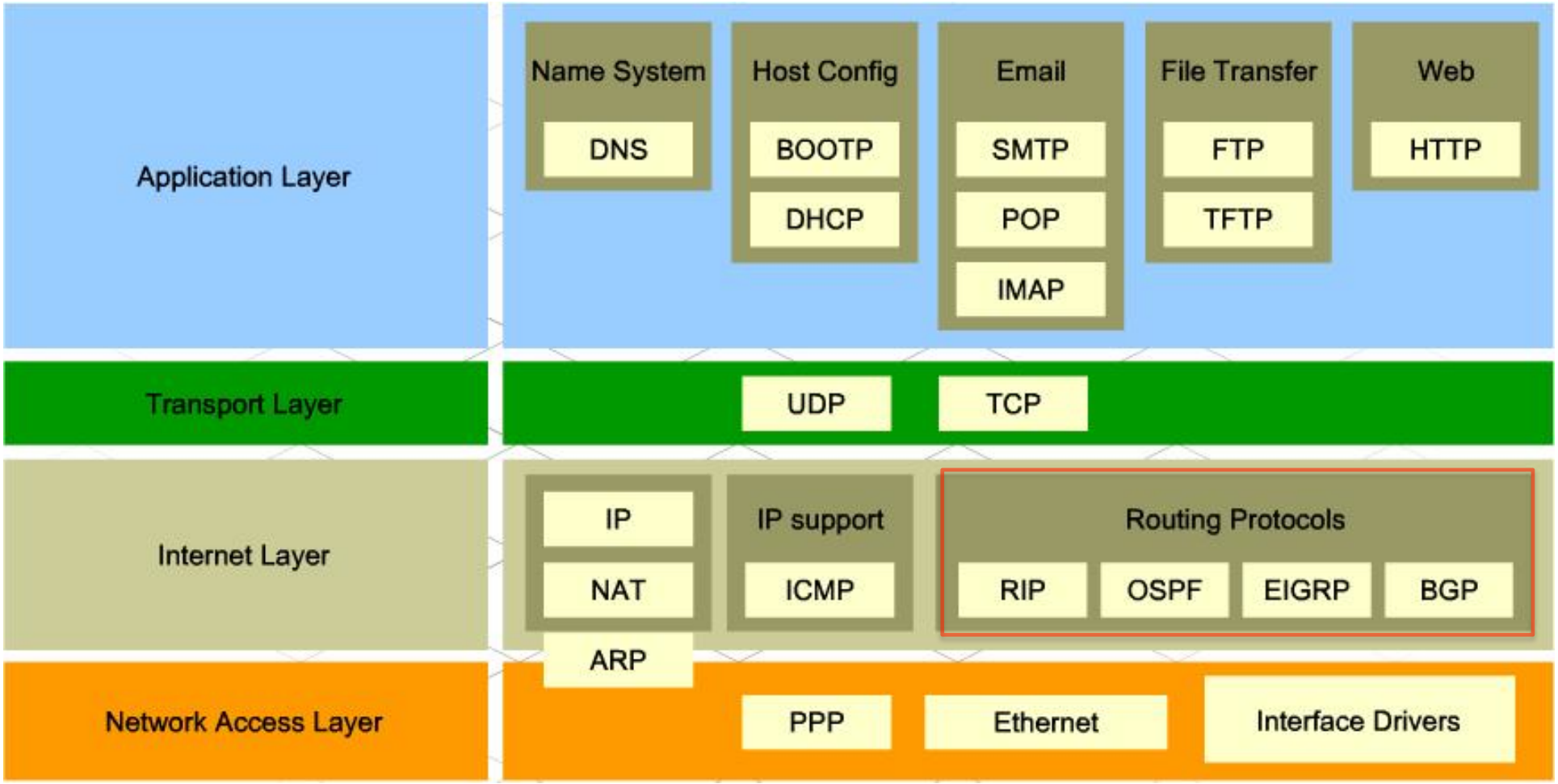
Routing

Módszerek:
Távolságvektor
Linkállapot

Protokollok: későbbiekben



Routing protokollok



Az elnevezésről...

.fr Routage

.de Routing

.ru Маршрутизация (marshrutizatsiya)

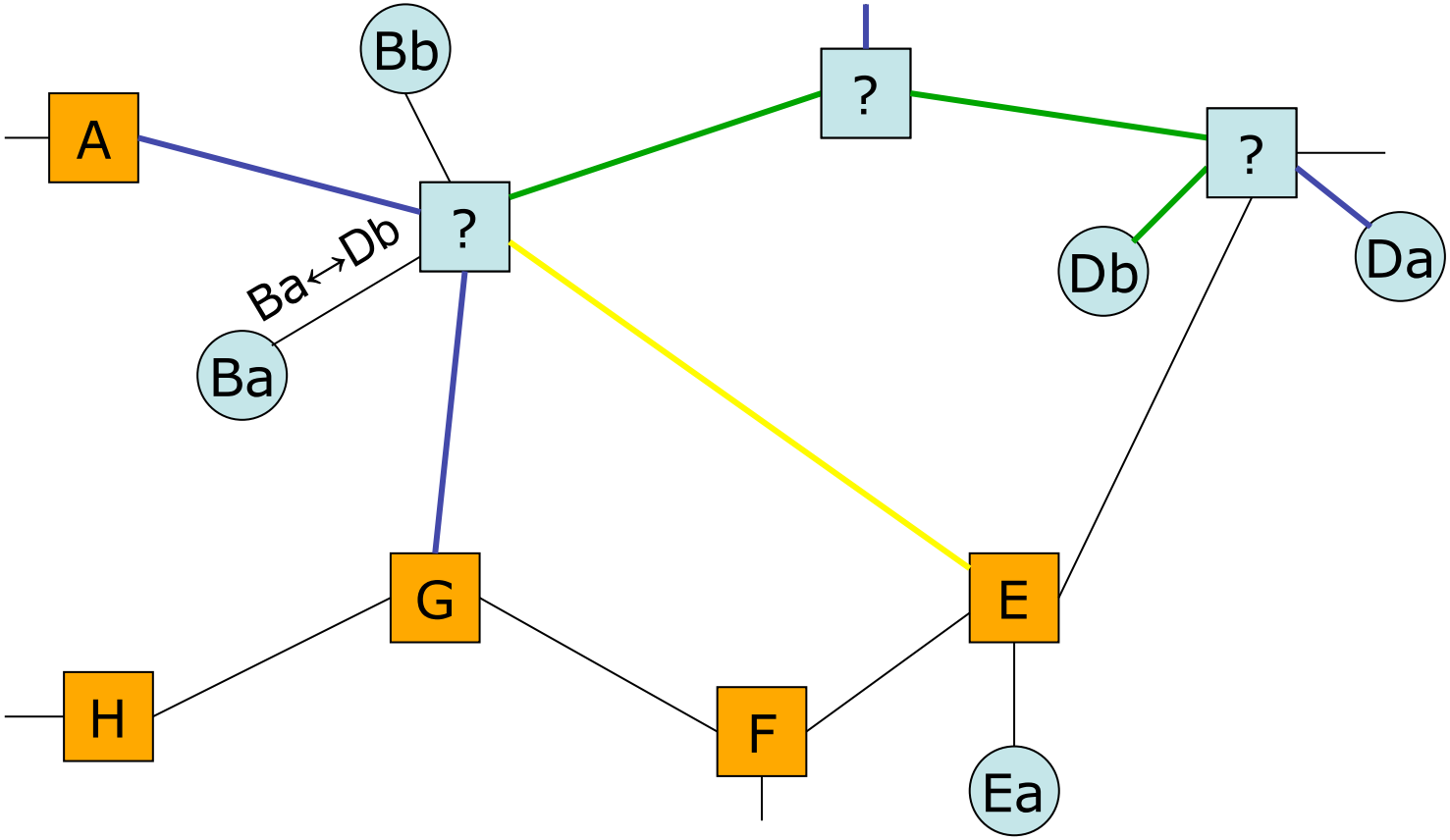
A Tanenbaum: Computer Networks könyv magyar fordításában: „forgalomirányítás”

Nálunk (általában): routing

Mi a routing?

- *Az a mechanizmus, mely segítségével a szállítandó információ a **megfelelő útón** kerül továbbításra a végpontok között*
  - *szűkebb értelemben: csak az útvonalválasztás/útvonalkijelölés*
  - *tágabb értelemben: ideértjük a csomópontokban a csomagok **továbbítását** is*
- A feladat érdekessége:
  - Általában ***nem állandó*** a hálózat felépítése: adaptív módon
  - Gyakran **„nagy" a hálózat mérete**: nincs pontos és aktuális információ a hálózat állapotáról.
- Ez a feladat egyaránt felmerül az összeköttetés-alapú és –mentes hálózatokban
  - ö. a.: útvonalkijelölés
  - ö. m.: útvonalválasztás
- Ide tartoznak az útvonalválasztó **módszerek**, **algoritmusok** és az azokat megvalósító **protokollok**

Bevezetés: a feladat értelmezése

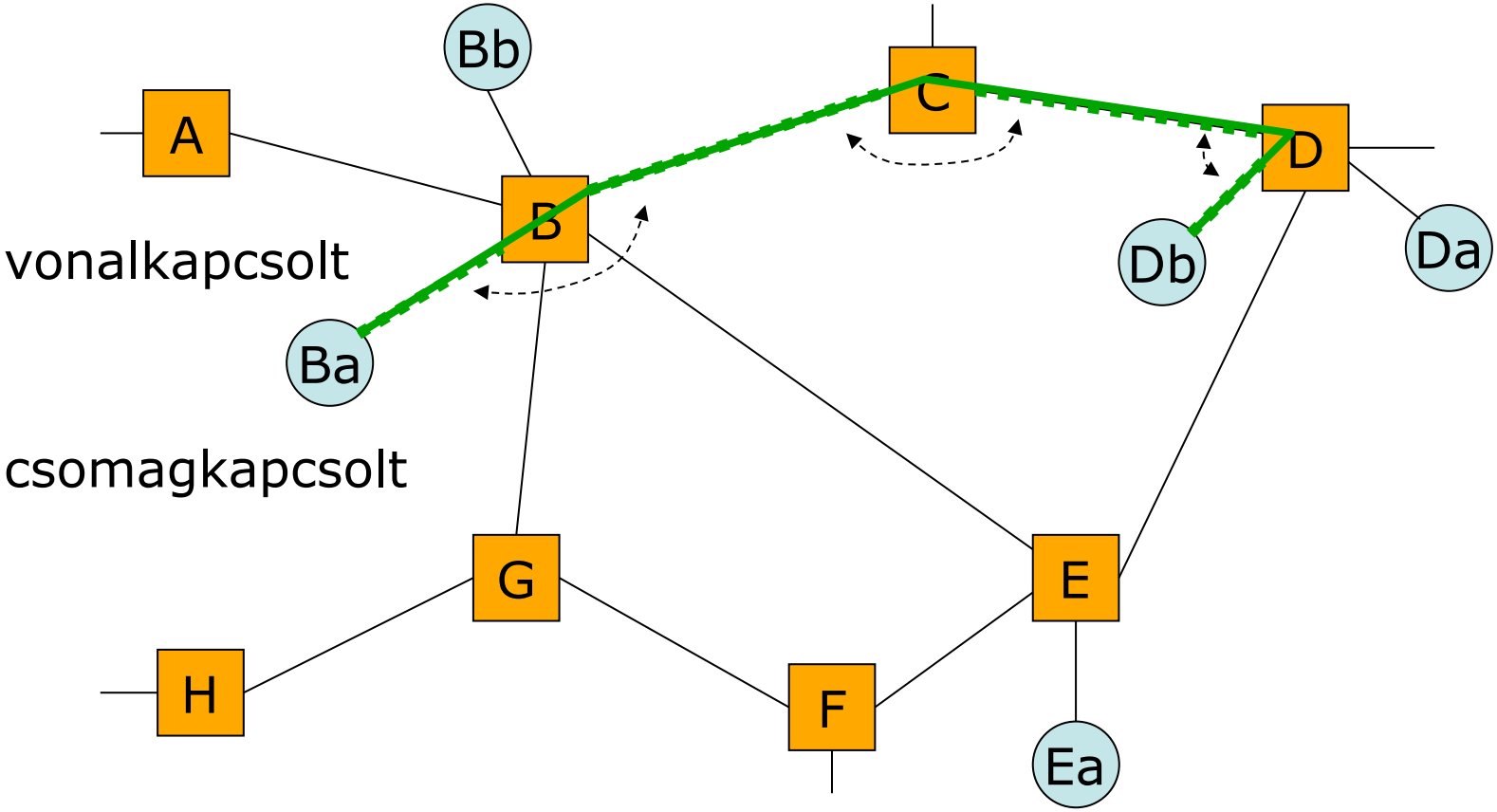


Mi tehát a feladat?

- **Megismerni, nyilvántartani** a hálózat felépítését:
  - csomópontok
  - összekötések (link)
  - lehetséges útvonalak

- **Összeköttetés-alapú eset:**
  - Csomópontokban felhalmozott ismeretek alapján
    - az útvonal kiválasztása lépésről-lépésre
    - **feljegyzése** az érintett csomópontokban
    - **értesítés** a kommunikáló végekhez, hogy elkészült az útvonal, megkezdhető a kommunikáció.
- **Összeköttetés-mentes**:
  - nincs külön útvonalkialakítás előre
  - a csomópontok a megfelelő útvonalat a csomag továbbításával **egyidejűleg** választják meg

Összeköttetés-alapú /-mentes



Mi tehát a feladat?

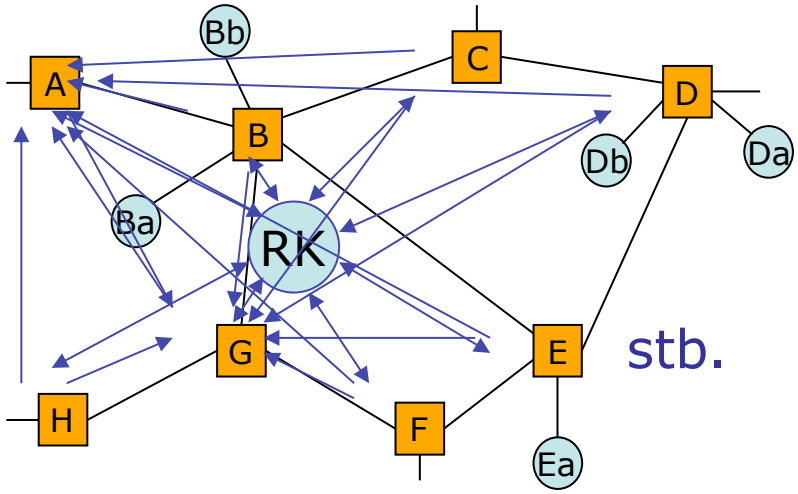
- **Áramkörkapcsolt eset:**
  - állandó átviteli csatorna létrehozása az összeköttetés időtartamára
- **Összeköttetés-alapú eset**
  - virtuális csatorna létrehozása a csomópontokban történt feljegyzésekkel
    - honnan hová kell tenni a csomagot
- **Összeköttetés-mentes eset:**
  - a csomópontok minden egyes csomagra újból elvégzik az útválasztást
  - nincsenek állandó feljegyzéseik
- *Utóbbi esettel foglalkozunk, a teljesség kedvéért megemlítve az előző kettőt*
- Mi kell az útválasztáshoz?
  - A hálózat felépítésének (csomópontjainak és linkjeinek) megismerése és nyilvántartása
  - Ezen ismeretek biztosítása a csomópontok számára

Útvonaltáblák (forwarding table)

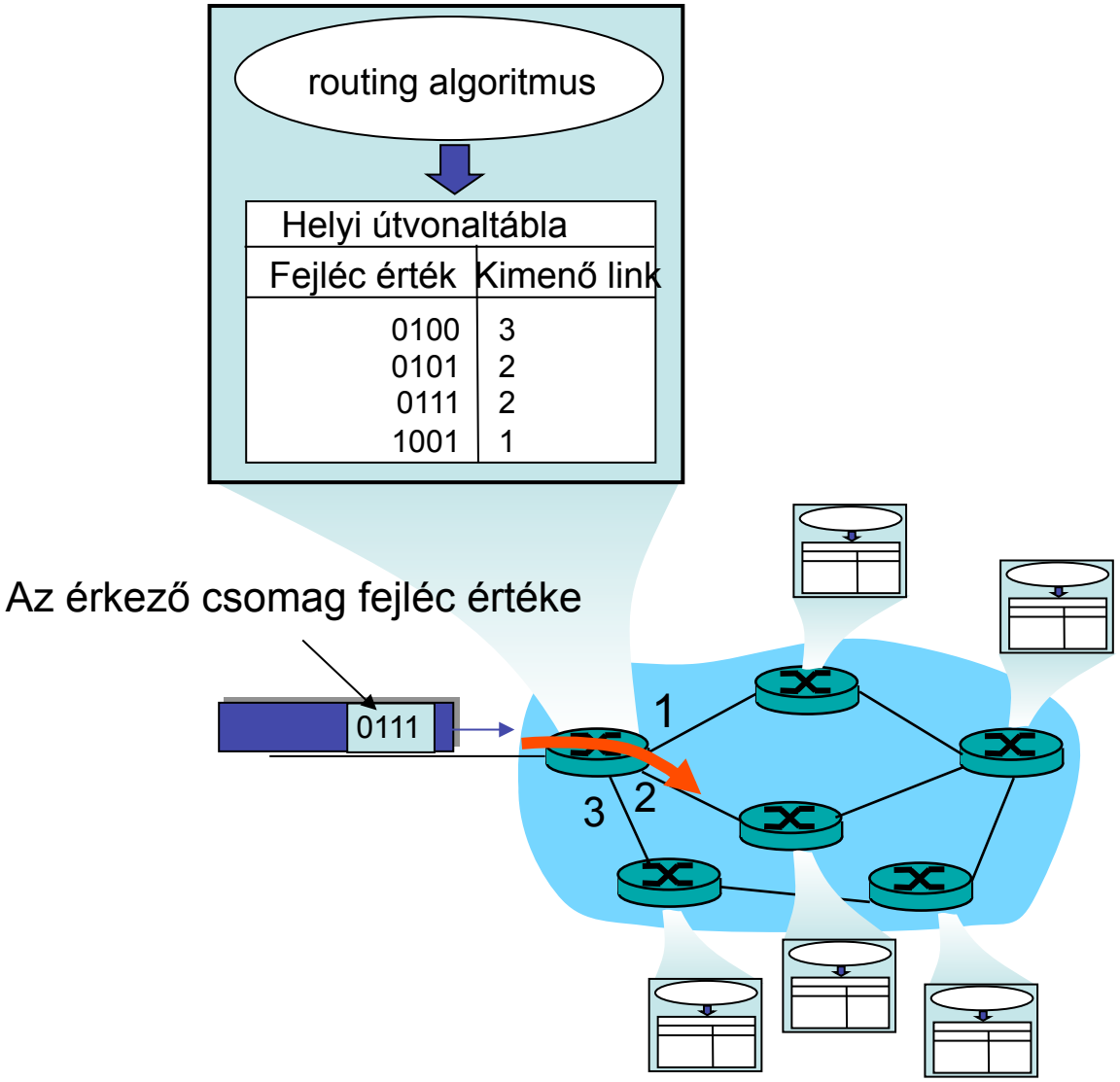
- Útvonalkijelölés ill. útvonalválasztás:
  - a csomópontok táblázatai alapján

Cél-végpont	következő csomópont
…	…

- A táblák kitöltése:
  - manuálisan
  - automatikusan:
    - centralizáltan
    - elosztottan



Csomagtovábbítás az útvonaltábla alapján



Centralizált **kontra** *elosztott* **módszer**

- **Centralizált:**
  - egy ponton összegyűjtjük a hálózatról megszerezhető valamennyi ismeretet
  - meghatározzuk az egyes csomópontok útvonaltábláinak bejegyzéseit
  - továbbítjuk azokat a csomópontokhoz.

- **Elosztott esetben**
  - valamennyi csomópont egyénileg gyűjt információt a hálózatról
  - saját útvonaltábláját készíti el

Centralizált előnyei-hátrányai

\+

- Egyetlen kép a hálózatról, minden útvonal-tábla „**konzisztens**” lesz

\-

- Nagyobb hálózatnál közben megváltozhat az állapota
- Sérülékenység *(„single point of failure”)*
  - megoldás lehet: tartalékolás

Elosztott előnyei-hátrányai

+

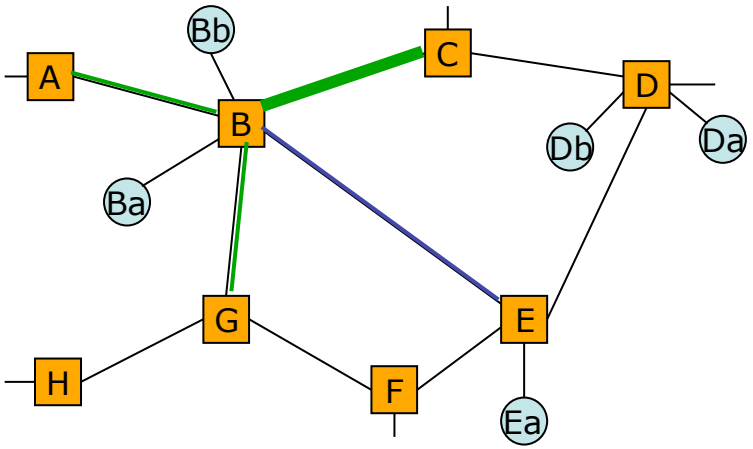
- Kevésbé sérülékeny

-

- Nincs egységes kép a hálózatról
- Eltérő „nézőpontok” a csomópontokban

A gyűjthető információ

- Csomópontok közötti összekötések, linkek:
  - pl.: *B-A, B-C, B-E, B-G*
    - nem működik: pl.: *B-E*
    - *B->Ba,->Bb*
- Linkek jellemzői:
  - *átv. seb., pl.*
    - *B-A:* 2 Mbit/s
    - *B-C:* 10 Mbit/s
    - *B-E:* 2 Mbit/ s
  - *Forgalom (sorhossz):*
    - *B-A:* 2
    - *B-C:* 0
  - *Terhelés %-ban*
  - *Kiszolgálási díj*



A routing módszerrel szembeni követelmények

- ***Minél kisebb méretű útvonaltábla:***
  - Kisebb tár, olcsóbb csomópont
  - Gyorsabb keresés a táblában, gyorsabban működő csomópont
  - Kisebb routingforgalom

- ***Robosztusság:***
  - Hibás tábla esélyének minimalizálása, mert a hibás tábla okozhat:
    - „fekete lyukat”
    - hurkot
    - oszcillációt

- ***Optimális útvonalak kijelölése:***
  - az út optimalitása az igénytől függő, lehet optimális a
    - legrövidebb *(miben mérjük?)*
    - legmegbízhatóbb
    - legolcsóbb

Routing a kommunikációs hálózatokban

- „Routing az összekötöttség – **connectivity** – **maximalizálására**”: katonai célok eleinte!

- Legyen a módszer **elosztott**!
  - Az útvonaltáblák kialakítását végezzék maguk a csomópontok

- Két alapvető módszer:
  - „distance-vector” – *Bellman-Ford algoritmus*
  - „link-state” – *Dijkstra algoritmus*
- A módszerek különböznek abban, hogy:
  - milyen információt gyűjtenek
  - hogyan gyűjtik az információt
  - hogyan terítik ezt később

Routing algoritmusok csoportosítása

Globális vagy decentralizált információ?

Globális:

- Minden routernek megvan a teljes topológia, linkköltség infó
- De nem centralizált, mindenki maga gyűjti és számolja ki
- "linkállapot" algoritmusok

Decentralizált:

- Router a fizikai szomszédjai felé vezető linkeket, költségeket ismeri
- Iteratív számolás: a szomszédoktól hallott infókra alapozva
- "távolságvektor" algoritmusok

Statikus vagy dinamikus?

Statikus:

- Lassan változnak az útvonalak

Dinamikus:

- Gyorsan változnak
  - Periodikus frissítés
  - Ahogy változás van
    - pl. topológiában vagy linkköltségben

A módszerek lényege

- Információgyűjtés, és -terítés történik
- „Kinek, mit mondunk?”:
  - „Távolságvektor” módszer (*distance-vector*):

A csomópontok elmondják a **hálózatról alkotott *elképzeléseiket* a szomszédaiknak**

  - „Linkállapot” módszer (*link-state*):

A csomópontok elmondják **mindenkinek a szomszédaikról nyert *tapasztalataikat***

A „távolságvektor” módszer

- A gyűjtött információ és a gyűjtés módja:

  A csomópontok elmondják a hálózatról alkotott elképzeléseiket a szomszédaiknak

- Az „elképzelések”:
  - melyik csomópont milyen távol van
  - egy lista (vektor) melynek elemei
    - csomópontazonosító – távolság párok

- A távolságvektorát közli mindegyik csomópont **valamennyi szomszédjával**
- A távolság mérése?

A „távolságvektor” módszer: a távolság mérése

- Legegyszerűbb esetben ez lehet *a **lépések** (ugrások, hop-ok) száma*

- Súlyozás:
  - *a link átviteli sebességével*
  - *a sorbanállási hosszal*
  - *a költséggel*

- *Szóhasználat: távolság = költség*
- Így a távolságvektor elemei az adott csomópont által elképzelt költségek a hálózat többi (ismert) csomópontjához

Távolságvektor algoritmus

Bellman-Ford egyenlet

Ha

$d_x(y)$:= a legkisebb költségű útvonal költsége *x* és y között

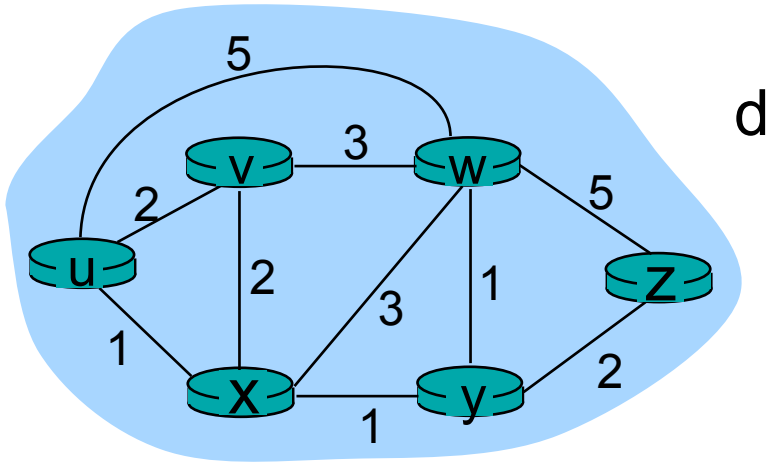
Akkor

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

ahol a minimum függvény *x* összes *v* szomszédjára van értelmezve

Bellman-Ford példa

Keressük az *u* és *v* közötti legkisebb költségű útvonalat:



$d_v(z) = 5,\ d_x(z) = 3,\ d_w(z) = 3$

B-F egyenlet szerint:

$$d_u(z) = \min \{ c(u,v) + d_v(z),\ c(u,x) + d_x(z),\ c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

Amelyik szomszédra jön ki a minimum ➜ útvonaltáblába kerül

Távolságvektor algoritmus

Iteratív, aszinkron

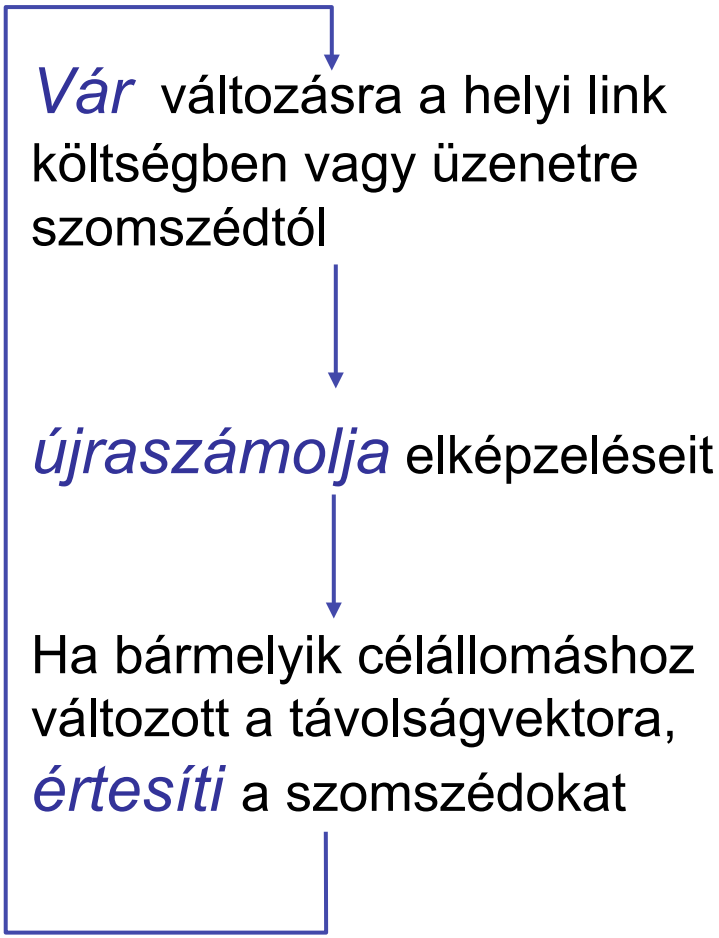
Helyi iterációt kiválthat:

- Helyi link költségének változása
- Új távolságvektor szomszédtól

Elosztott:

- Minden csp **csak akkor** értesíti a szomszédait, ha változik a távolságvektora
  - A szomszédok aztán értesítik az ő szomszédaikat ha szükséges

Minden csp:



$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

x tábla

költség -ig

-tól \	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

költség -ig

from \	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

y tábla

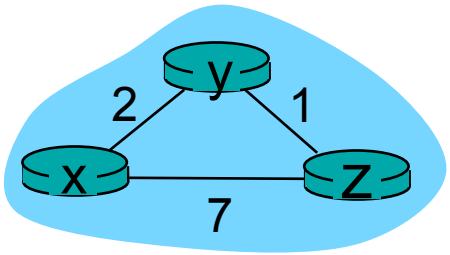
költség -ig

-tól \	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

z tábla

költség -ig

-tól \	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



idő

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

x tábla

költség -ig

-tól	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

költség -ig

-tól	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

költség -ig

-tól	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

y tábla

költség -ig

-tól	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

költség -ig

-tól	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

költség -ig

-tól	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

z tábla

költség -ig

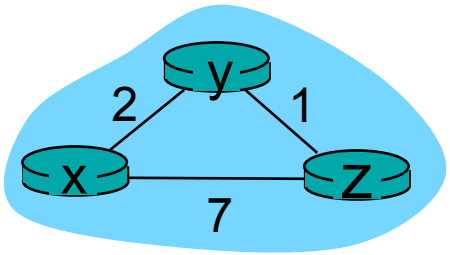
-tól	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

költség -ig

-tól	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

költség -ig

-tól	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



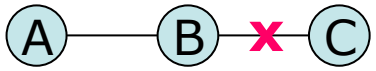
idő

A távolságvektor módszer alapproblémája (1)

- **A végtelenig számolás**
  - meghibásodások léphetnek fel
  - a csomópontok tévesen korrigálhatják távolságvektorukat
    - nem tudják, hogy a kapott távolságvektorokat a szomszédaik milyen módon, lépésekkel hozták létre
      - ‘routing by rumor’
  - egyre növelik egy adott csomóponthoz való távolságukat
    - elvileg végtelenig
    - konkrét megvalósításoknál véges számig
- Következmény: egymásnak irányítják az el nem érhető csomóponthoz tartó forgalmat, ezzel túlterhelnek linkeket

A távolságvektor módszer alapproblémája (2)

- Példa a *végtelenig számolásra*:



	C táv	Köv. lépés
A	2	B
B	1	C

- A *B*-*C* link megszakad

A	2	B
B	∞	-

- *B* kijavítja a bejegyzését

A	∞	-
B	3	A

- *B* és *A* kicserélik elképzeléseiket és korrigálnak:

A	4	B
B	∞	-

- Ha ismét cserélnek és korrigálnak:

A távolságvektor módszer javítási lehetőségei

- Split horizon: megtiltja a csomópont számára az útvonal hirdetését azon interfész felé, ahonnan tanulta
- Route poisoning: hirdeti a csomópont, hogy elérhetetlen az útvonal, törölni kell a táblákból
  - Pl. ha maximális hopszám 15, 16-ra állítja
- Kettőt együtt alkalmazzák: split horizon with poison reverse
- Holddown timer: amikor értesül a csomópont, hogy elérhetetlen egy útvonal, indít egy időzítőt
  - amíg le nem jár, nem foglalkozik azon útvonal hirdetményeivel
  - vagy megjavul ez idő alatt az útvonal, vagy új útvonalat keres
- Növekszik a konvergencia idő, további komplexitás!

A „linkállapot” módszer

- A gyűjtött információ és a gyűjtés módja:
  A csomópontok elmondják **mindenkinek a szomszédaikról nyert tapasztalataikat**

- A „tapasztalatok”:
  - a **szomszédokhoz vezető linkek aktuális állapota**: ez pontosan ismerhető!
  - a link költsége valamilyen mérték szerint

- Az **információ elküldése mindenkinek „*elárasztással*”:**
  - elküldik a szomszédokhoz, amelyek továbbadják
    (kivéve azon a linken, amelyen érkezett)

- *A linkállapot-információk alapján a **legrövidebb utak kiválasztása***
- *Minden csomópont rendelkezik a **globális link-állapot információkkal** az elárasztásnak köszönhetően*

Linkállapot routing algoritmus

Dijkstra algoritmus

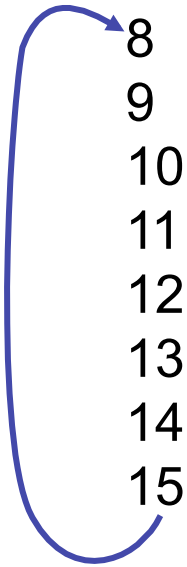
- Input: minden csp ismeri a hálózati topológiát, link költségeket
  - Link állapot broadcast-nek köszönhetően (sorszámozott hirdetések!)
  - Minden csp.-nak ugyanaz a hálózati képe van
- Kiszámolja a legrövidebb útvonalat **a forrástól minden másik csp-ig**
  - Kitölti a forrás útvonaltábláját
- Iteratív: *k* iteráció után, ismeri a legrövidebb utat *k* célállomás felé

Jelölés:

- c(x,y): költség *x*-től *y* csp-ig,
  - = ∞ ha nem szomszédok
- D(v): a forrástól a *v* célállomásig vezető útvonal pillanatnyi költsége
- p(v): forrástól *v* célállomásig vezető útvonalon, a *v* előtti csp
- N': azon csp-ok halmaza, amelyekhez megvan a legkisebb költségű útvonal

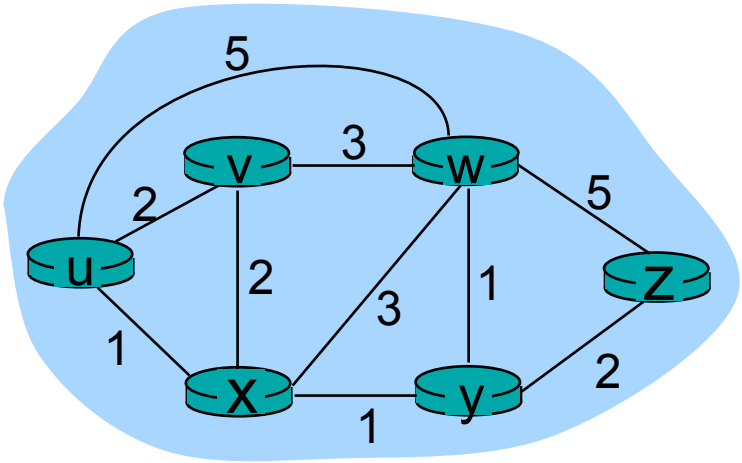
Dijkstra algoritmus

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until all nodes in N'
```



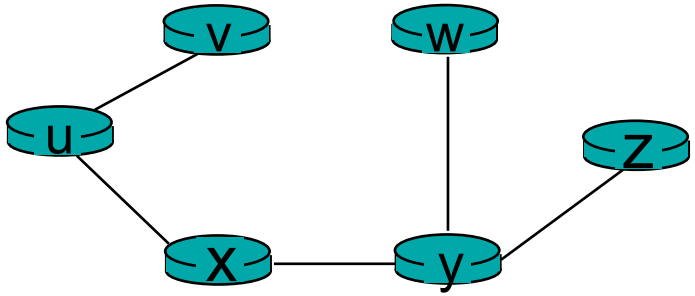
Példa a Dijkstra algoritmus működésére

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Példa a Dijkstra algoritmus működésére

Eredménye: legrövidebb útvonalakból alkotott fa (*u*-ból nézve)



Eredménye: útvonaltábla *u*-ban:

célállomás	kimenő link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Dijkstra komplexitása

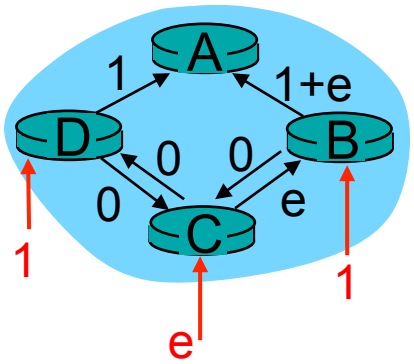
Az algoritmus komplexitása: *n* csomópont esetén

- Minden iterációban: minden csp-ra megnézni, melyik a legkisebb költséggel elérhető, ami nincs benne N' halmazban
- n(n+1)/2 összehasonlítás: O(n^2)
- Hatékonyabb implementációval: O($n\log n$)
- Gyakorlatban: hierarchikus linkállapot routing protokollok

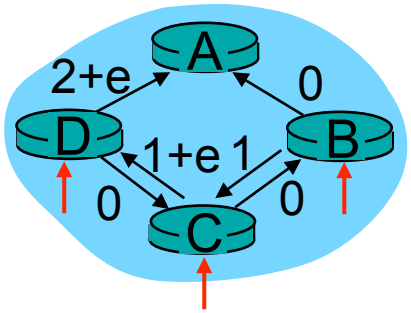
Problémák

Oszcillációk lehetségesek:

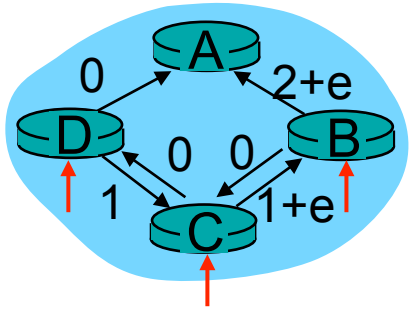
- Pl. ha link költsége = pillanatnyi forgalom a linken
- Megoldás:
  - időben randomizálni a csp-ok által kiküldött linkállapot hirdetéseket



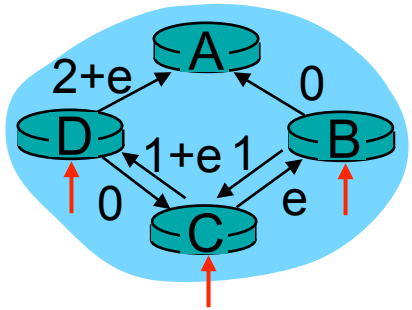
A-nak küldik



B és C jobb útvonalakat fedez fel A-ba



B,C és D jobb útvonalakat fedez fel A-ba



B,C és D jobb útvonalakat fedez fel A-ba

Távolságvektor vs. Linkállapot

Üzenet komplexitás

- LÁ: *n* csp, *E* link, O(nE) üzenet kiküldése
- TV: *üzenetcsere csak szomszédok között*
  - konvergencia idő hullámzik

Konvergencia idő

- LÁ: O(n^2) komplexitású algoritmushoz O(nE) üzenet
  - lehetnek oszcillációk
- TV: k.i. hullámzik
  - lehetnek routing hurkok
  - végtelenig való számolás

Robusztusság: mi történik ha egy router rosszul működik?

LÁ:

- a csp hirdethet téves **link** költséget
- minden csp csak a saját tábláját számolja

TV:

- csp hirdethet téves **útvonal** költséget
- Minden csp tábláját használja a többi
  - Hiba végig terjedhet a hálózaton

Távolságvektor vs. Linkállapot

- Távolságvektor alapú protokollok:
  - Rosszabbul skálázódnak
  - Lassabb konvergencia
  - Kisebb hálózatokban használják őket
    - Pl. egy ISP hálózatán belül

  - Manapság a linkállapot alapú módszerek dominálnak tartományon belüli routing esetén

Routing tábla metrikák

- A metrika az adott routing algoritmustól függ
  - Linkkapacitás
  - Késleltetés
  - Hopszám
  - Megbízhatóság
  - Csomagvesztési ráta
  - MTU stb. kombinációja

A hierarchikus routing

- Kiterjeszthetőek („skálázhatóak”) a routing algoritmusok egyszintű (flat) hálózat esetén?
- *N* csomópontja esetén kimutatható:
  - A legrövidebb út számítása: $O(N \log N)$
  - Az útvonaltábla mérete: $O(N)$
  - Nem lehet hierarchia nélkül egy Világháló:

# csomópont	Táblahely	Számítási idő
1	1	$O(0)$
1.000	1.000	$O(3.000)$
1.000.000	1.000.000	$O(6.000.000)$
100.000.000	100.000.000	$O(800.000.000)$
10.000.000.000	10.000.000.000	$O(100$ milliárd$)$

Az *autonóm rendszerek*

- Megoldás: a hálózat szegmentálása

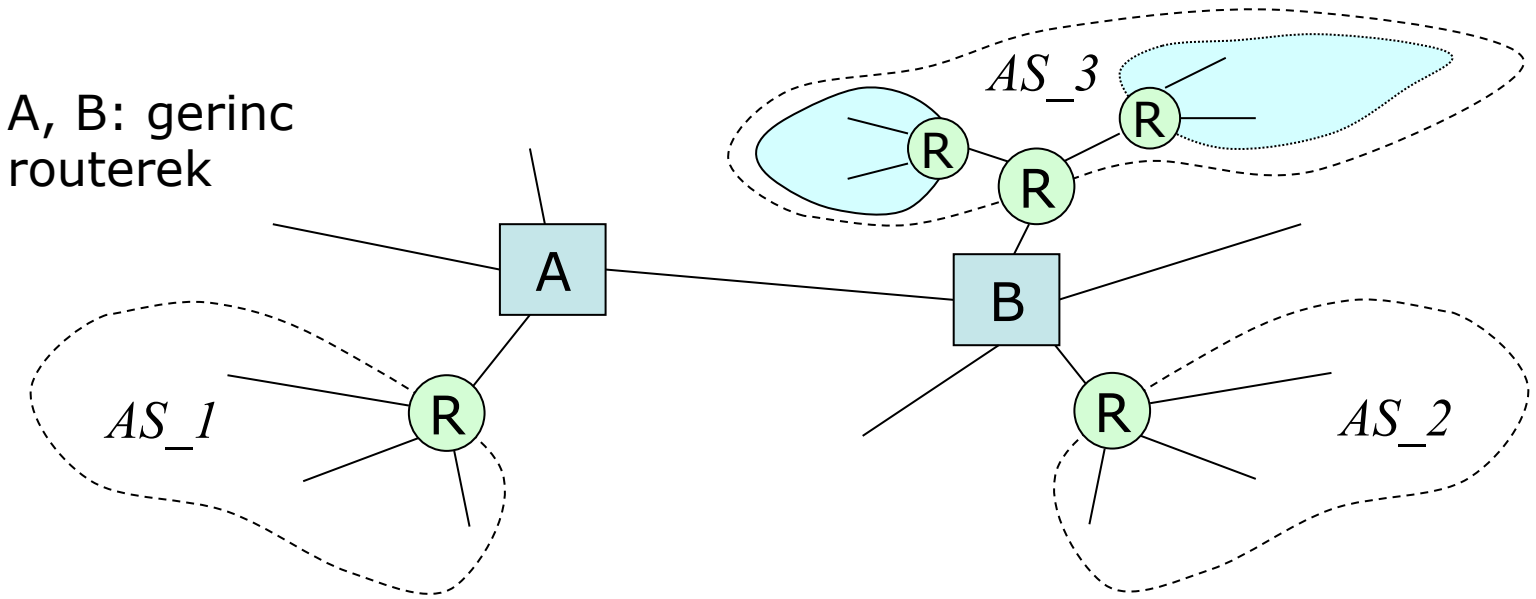
- ***Autonóm rendszerek*** kialakítása
  - *önállóan oldanak meg routingfeladatokat*
  - *a külvilág felé egyetlen elérési pontként szerepelnek*

- Az autonóm rendszer (AS)
  - olyan egység, amelyen belül **egységes routing policy** érvényesül
  - egyetlen hálózat, vagy hálózatok csoportja
  - közös hálózatadminisztrátor kezeli
  - egyetlen szervezeti egység (pl. egyetem, üzleti vállalkozás) megbízásából
- Gyakran **routing-domain**-nek is nevezzük
- Globálisan egyedi azonosító: **ASN (Autonomous System Number),** a IANA-tól

Autonóm rendszerek

- Egy autonóm rendszer a többi autonóm rendszer számára képviseli az általa képviselt csoporto(ka)t
  - az ún. **határ-routereken** keresztül érhető el (gateway router angolul)

- Az egész csoport **egyetlen bejegyzés** lesz az útvonaltáblákban



AS típusok

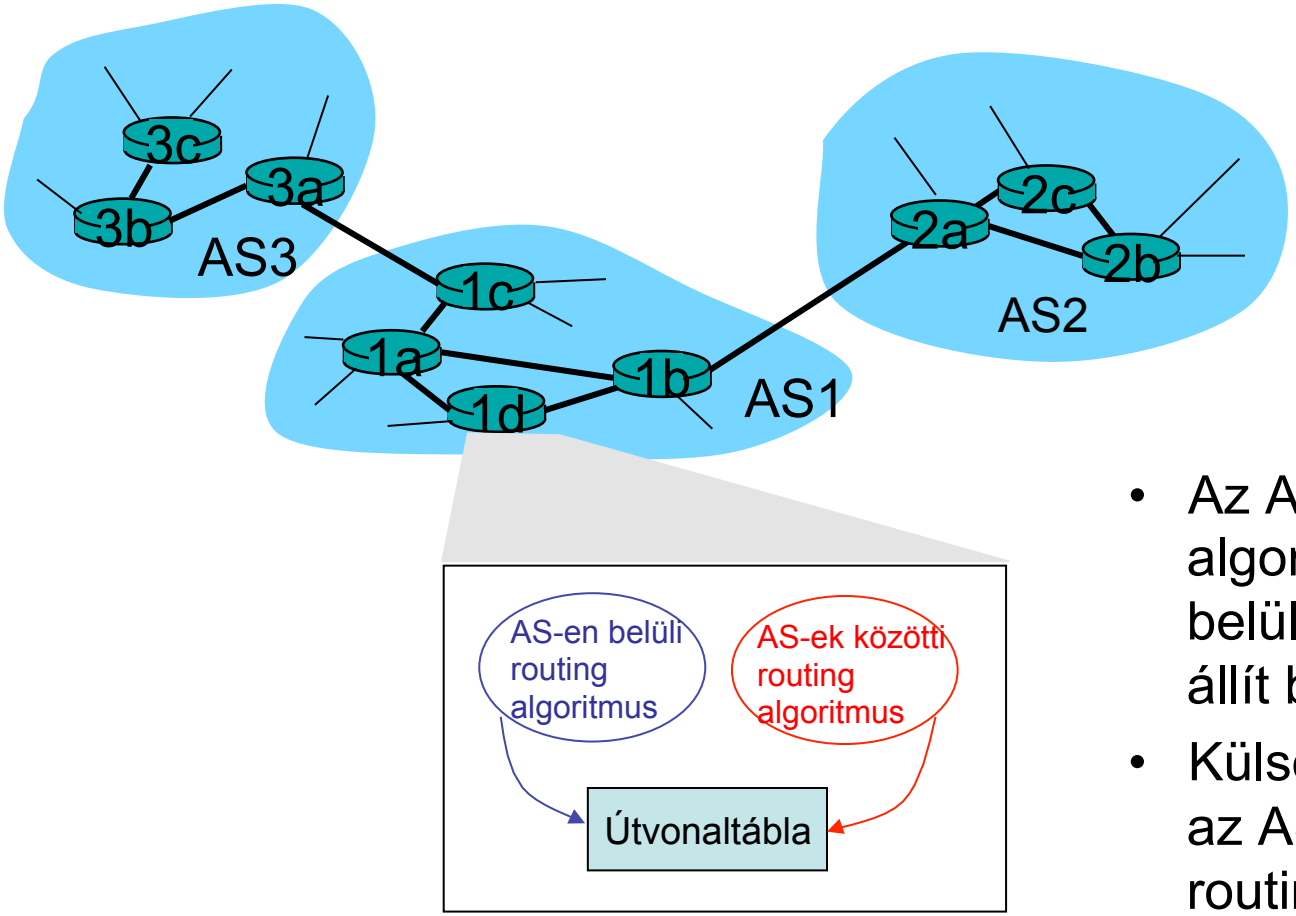
- Multihomed AS
  - Több mint egy AS-el van összekötve
  - Meghibásodás esetén sem lesz elvágva
  - Viszont nem enged átmenő forgalmat más AS-ek között
- Stub (csonk) AS
  - Csak egy másik AS-el van összekötve
    - Gyakorlatban többel is van peering, csak nem nyilvános
- Tranzit AS
  - Csak átmenő forgalmat szolgálnak ki
  - Összes ISP ilyen

AS-k közötti útvonalválasztás

- Multihomed és tranzit AS esetén több útvonal közül választhat más AS-ek felé
- Sok esetben nem a linkkapacitás vagy késleltetés számít, hanem az üzleti kapcsolatok a szomszédos AS-el

- Hot-potato (forró krumpli) elv:
  - minél rövidebb legyen az út a saját AS-én belül
  - annak ellenére, hogy lehet összeségében így hosszabb lesz az út
- Cold-potato:
  - minél hosszabb ideig a saját hálózatán belül akarja tartani a forgalmat, szinte a felhasználóig
  - szolgáltatás minőségi garanciákat nyújthat

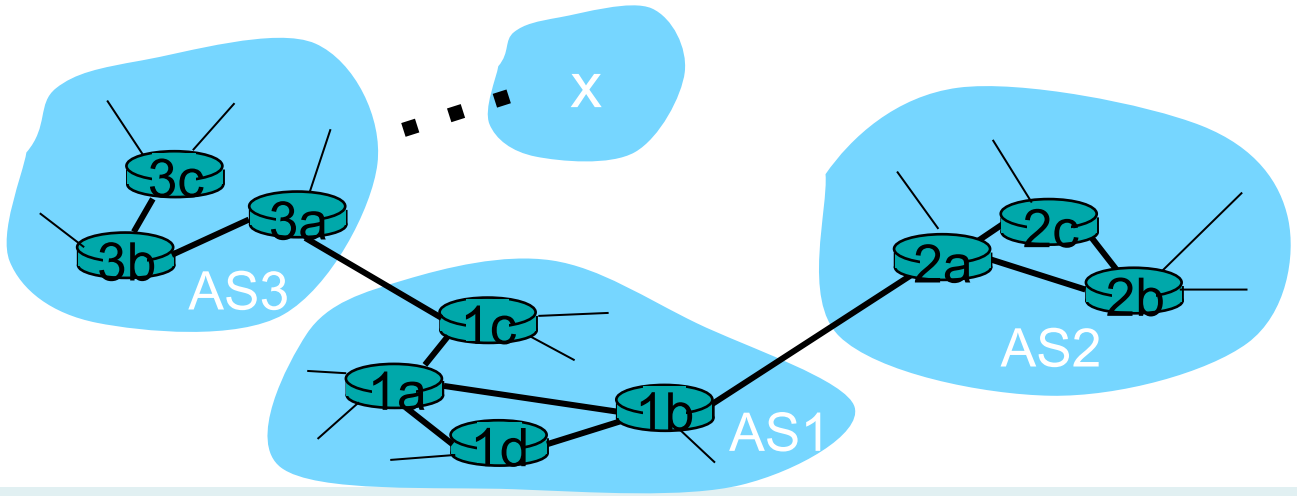
AS-k közötti útvonalválasztás



- Az AS-en belüli routing algoritmus a tartományon belüli célállomásokhoz állít be útvonalakat
- Külső célállomásokhoz az AS-en belüli és közötti routing algoritmus is

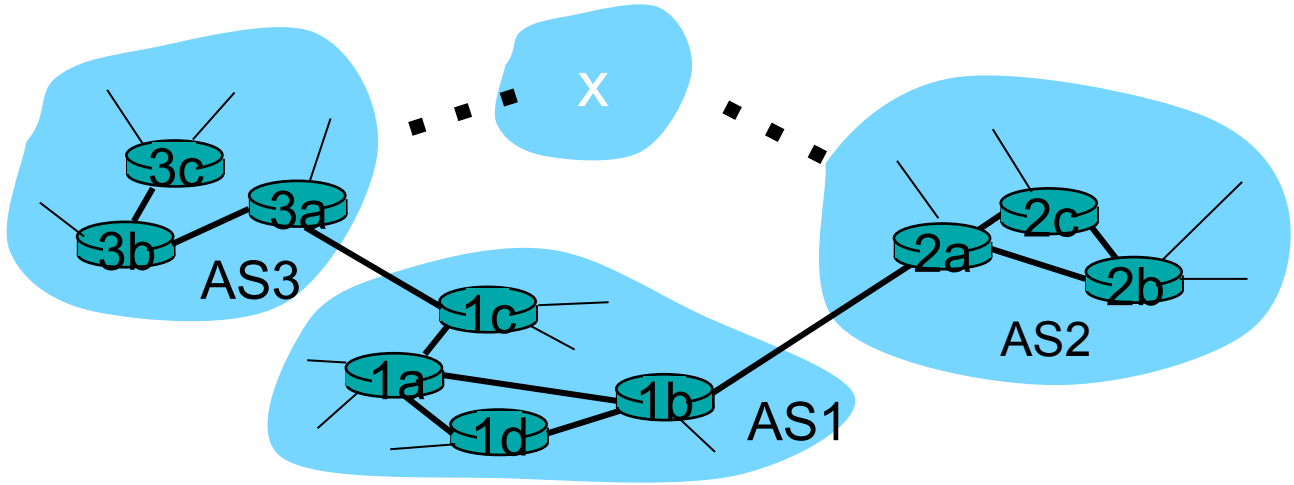
Példa: 1d router útvonaltábla beállítása

- AS1 “megtanulja (AS közötti routing protokollal):
  - *x* alhálózat az AS3-on (1c gateway) keresztül elérhető, de AS2-ön keresztül nem
- AS közötti routing protokoll elterjeszti ezt minden belső routernek (AS1-en belül)
- Az 1d router AS-en belüli routingal meghatározza, hogy *I* interfészén keresztül vezet a legkisebb költségű út 1c-hez.
  - Új bejegyzés: *(x,I)*



Példa: több AS közötti választás

- AS1 “megtanulja” AS közötti r. p.-al
  - *x* alhálózat AS3-n *és* AS2-n keresztül is elérhető
- 1d routernek meg kell mondani melyik gateway-en keresztül küldje a csomagjait x alhálózatnak
  - **AS közötti** r. p. dolga!
  - Amikor megvan a gateway: akkor **AS-n belüli** r.p.-al kell meghatározni a hozzá vezető legkisebb költségű utat!

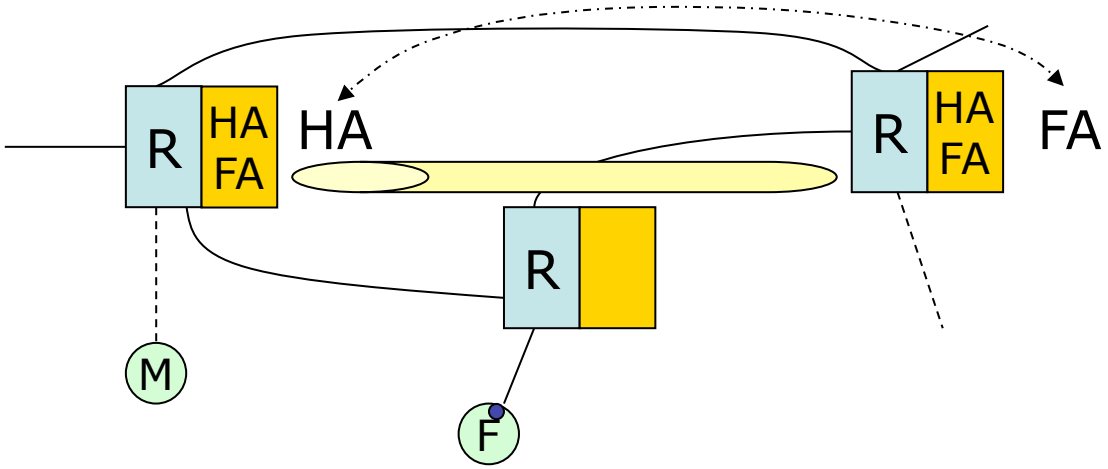


Mobil végpontok kezelése

- Eddig: egy végpontot a csomóponthoz való kapcsolata definiálja
  - pl. Ba, Bb: a B csomóponthoz kapcsolódó végpontok
- A végpontok átmozoghatnak más csomópont környezetébe, más autonóm rendszerbe
  - mozgás: vándorló (nomádikus) végpont, mobil végpont
- A többi végpont csak a mozgó csomópont eredeti (pl. Ba) címét ismeri
  - jó lenne, ha nem is kellene tudnia, hol van aktuálisan Ba
- Ezért a hálózatnak kell kezelnie Ba mozgását – ez a mobil routing feladata

Konkrét megvalósítás az IP protokollnál: mobil IP

- Elv: „kiegészítések” a csomópontokon:
  - Mobil ügynök:
    - Home agent = hazai ügynök
    - Foreign agent = idegen ügynök

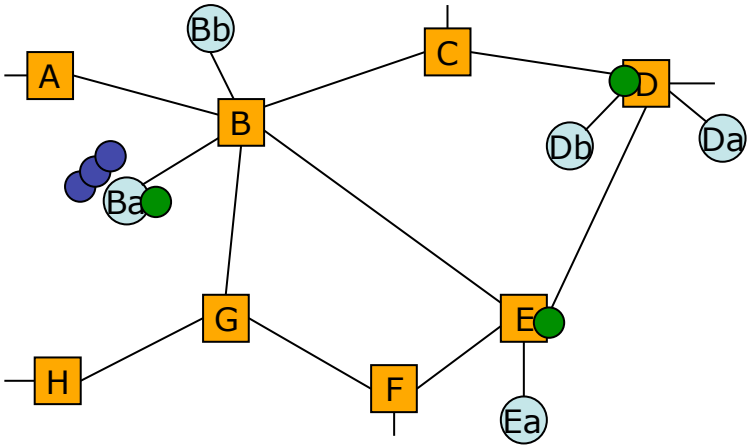


Multicast routing

- Egy másik érdekes járulékos feladat az alap routinghoz képest: ***multicast routing***
- Nem egyetlen végponthoz/címre kell eljuttatni a csomagokat, hanem egyidejűleg többre
  - a gyakorlatban: pl. adat-beszéd-videókonferencia
- *Elnevezés:*
  - *broadcast - (műsor)szórás mintájára, mindenkinek a hálózaton*
  - *multicast: egy csoportnak*
  - *anycast: egy valakinek a csoportból (pl. aki a legközelebb van a forráshoz)*
  - *unicast: egy adott végpontnak*
- Magyarul: *többesadás*

A multicast routing: példa és megoldandó feladatok

- Példa:
  - *Ba -> Da, Db* és *Ea*
    - Három unicast
    - Multicast



- Megoldandó feladatok:
  - Címzés
  - Csoportkezelés
  - Útválasztás segítése

A többescímzés, csoportazonosítás

- Azonosítsuk a csoportot!

- A hálózat végpontjait azonosítják a
  - Fizikai címeik
  - Logikai címeik
    - ezt tartják a csomópontok nyilván az útvonaltábláikban
- A csoport azonosítására használhatnánk a csoport tagjainak címlistáját is...
- Jobb: a csoportot **egy**, az egyedi logikai címhez hasonló **azonosító** különbözteti meg: **csoport-cím**!
- A csoport létezésének feltétele: legalább egy aktív információforrása legyen

Multicast routing, módszerek

- A csoport résztvevői a hálózatban:
  - Sűrű elhelyezkedés:
    - Elárasztás + lemondás (flood-and-prune)
      - Reverse path forwarding (utak megfordítása)
        - » Legrövidebb útvonalfa alkalmazása
      - Csoportlemondás
      - Forráslemondás
  - Ritka elhelyezkedés:
    - Explicit csatlakozás
      - Gerincalapú fa alkalmazása (Core Based Tree)
- Elhelyezkedés-független
  - A kettő ötvözete

Összefoglalás

- Routing:
  - **Az a mechanizmus, mely segítségével a szállítandó információ a megfelelő úton kerül továbbításra a végpontok között**
  - szűkebb ételemben: az a folyamat, amelynek eredményeként a csomópontokban létrejönnek az útvonaltáblák

- Két fő módszer és két algoritmus az útvonaltáblák létrehozására:
  - Távolságvektor (distance-vector) – Bellman-Ford
  - Linkállapot (link-state) - Dijkstra

- Kiegészítések szükségesek:
  - Mobilitás biztosítására
  - Pont-multipont, multipont-multipont kommunikáció esetén

- A routingot megvalósító *protokollokat* később tárgyaljuk

Kérdések?

KÖSZÖNÖM A FIGYELMET!

Dr. Simon Vilmos
docens
BME Hálózati Rendszerek és Szolgáltatások Tanszék
svilmos@hit.bme.hu