# SZAKDOLGOZAT

Józsa László

Debrecen
2009

**Debreceni Egyetem**
**Informatikai Kar**

# Egy működő kommunikációs hálózat elemzése, tesztelése.

Témavezető:
Dr. Almási Béla
Egyetemi docens

Készítette:
Józsa László
Mérnök informatikus (B.Sc.)

Debrecen
2009

# Tartalomjegyzék

# 1. Bevezetés

Manapság minden fajta eszköz egy médiumot kezd el használni, ami azt eredményezi, hogy az eddig különálló hálózatok közötti határok elmosódnak, és minden mindennel kapcsolatban áll valamilyen módon. A kábeltelevíziós hálózat, a telefonhálózatok és az adathálózatok mind kezdenek összeolvadni egy nagy egésszé, amit mi csak Internetként emlegetünk. Az évek folyamán a hálózat egyre bonyolultabbá vált, gondoljunk csak a vezeték nélküli hálózat vagy akár az IP telefónia megjelenésére, amely számos új problémát vetett fel, és alapjaiban formálta át például a hálózat biztonságot. Ezek mellet a számítógépes hálózatok megjelenése emelte egy új szintre az úgynevezett távmunkásokat vagy akár az otthoni munkát. Az emberek sokkal hatékonyabban vihették haza a munkájukat és azokat otthonukból képessé váltak ellátni is. Immár elkerülhetetlenül is minden nap kapcsolatba kerülünk kisebb- nagyobb hálózatokkal. Ahol pedig a hálózatok megjelentek, ott a forgalomirányítás is elengedhetetlen.

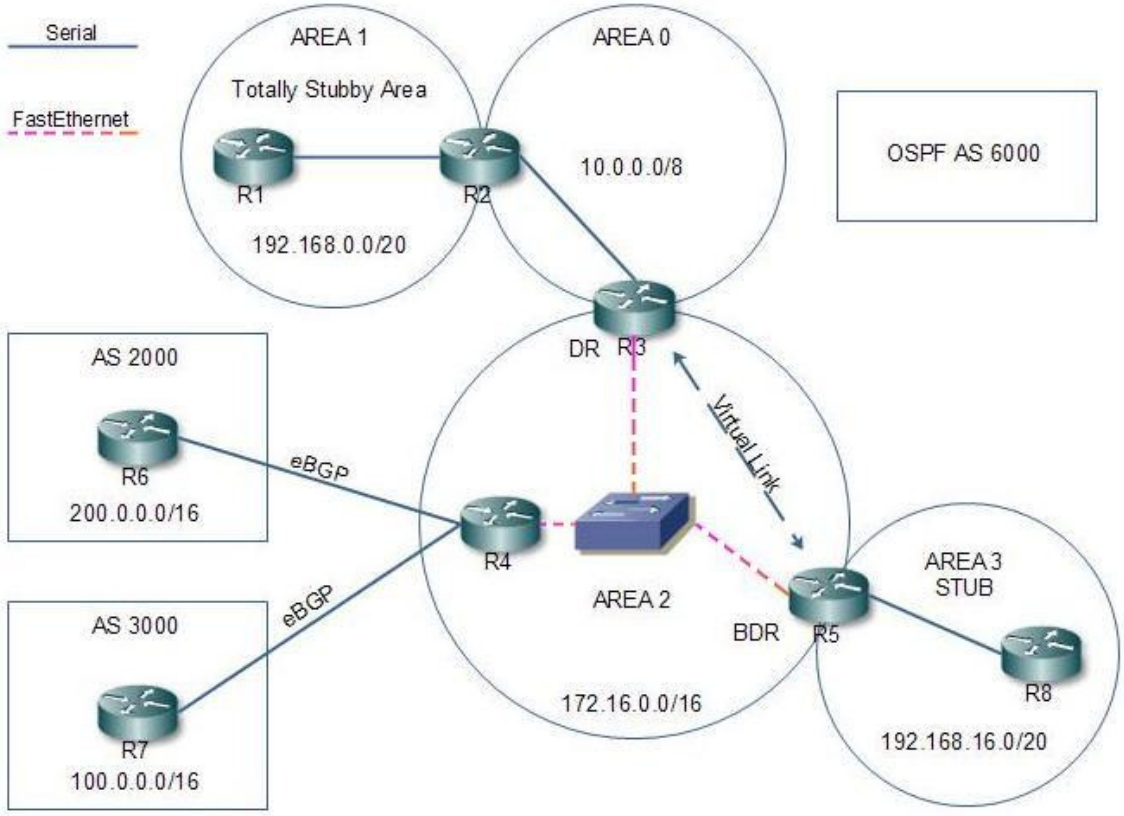
A forgalomirányítás nem más, mint különböző alhálózatok összekapcsolása valamilyen módon. Munkám során főleg forgalomirányítási hibafeltárás és elhárítás a feladatom, így napi szinten kapcsolatban állok nagy és komplex forgalomirányított hálózatokkal. Ismernünk kell a forgalomirányító protokollok működését, hogy azokat a legjobban ki tudjuk használni, és ott ahol erre szükség is van. Ha nem vagyunk tisztában a forgalomirányítás alapjaival, nem tudunk továbblépni és az ezeket használó megoldásokat kellő magabiztossággal implementálni. A legtöbbet előforduló protokoll szolgáltatói szemszögből a BGP, hiszen ennek nincs alternatívája, ez az egyetlen, ami képes autonóm rendszerek közötti forgalomirányításra. Dolgozatomban kitérek a BGP-n kívül, egy belső irányítóprotokollra is, ami a nyílt szabványból kifolyólag az OSPF-et alkalmazom. A dolgozat Cisco szemszögből közelíti meg a dolgokat. Ezek bemutatására és vizsgálatára egy teszthálózatot építettem, amiben a gyakorlati dolgok tárgyalásra kerülnek.

A szakdolgozat megértéséhez feltételezett egy alap szintű hálózatos tudás. Sok kifejezés angolul található meg a dolgozatban, mert ezek a magyar terminológiában nem találhatóak meg.

# 2. A topológia bemutatása

A szakdolgozatomban egy általam létrehozott teszthálózatot fogok bemutatni. Ki fogok térni a hálózatban használt gyakorlati dolgok elméleti hátterére, magyarázatára, hiszen ezek ismeretére szükségünk van, hogy megértsük a működését, és miért éppen azt kellett használnunk ott, ahol használtuk.

A topológia megvalósítását a GNS3[1] nevű szimulátor porgrammal valósítottam meg. Ez valódi IOS-t[2] szimulál, így a valósághoz nagyon is hasonló, körülbelül 99%-osan, azonos képet ad, mint ha igazi eszközöket használtam volna. A mellékletekben megtalálható mind a nyolc forgalomirányító általam készített konfigurációja. Az alapértelmezett dolgok nem kerülnek leírásra.




**1. ábra**

[1] http://www.gns3.net/
[2] http://en.wikipedia.org/wiki/Cisco_IOS

# 3. Manapság használatos protokollok

## 3.1. OSPF (Open Shortest Path First)

Az OSPF nyílt szabványú kapcsolat állapot alapú forgalomirányító protokoll. A protokoll az SPF algoritmust használja arra, kiszámítsa a legjobb utat a célhoz. Viszonylag gyorsan konvergál, másodperceken belül reagál a hálózat változásaira. Ennek ára van, több processzort és memóriát használ, mint távolságvektor alapú társai. Az OSPF bizonyos szituációkban sokkal bonyolultabb is lehet, ha figyelembe vesszük a topológiai és konfigurációs opciókat.

Az OSPF úgy lett tervezve, hogy flexibilisen kezelje a hálózatot, és mivel nyílt szabvány, alkalmazható különböző gyártók forgalomirányítóival is. Az OSPF-ről mindenre kiterjedő leírást az RFC 2328 dokumentum tartalmazza.

### 3.1.1. Kapcsolatállapot alapú protokollok koncepciója

A kapcsolatállapot alapú protokollok bonyolultak, Dijkstra SPF algoritmusát használják. Ha ezt szeretnénk egy távolságvektor alapúhoz hasonlítani, a kapcsolatállapot alapúak több információt dolgoznak fel a forgalomirányítóban, ezáltal csökkentve a hálózati forgalmat, viszont növelve az erőforrások (memória illetve processzor) felhasználását. Az egyiket beáldozzák a másik oltárán. A kapcsolatállapot alapú protokollok feljegyzik az összes lehetséges útvonalat, így sok olyan technikát itt nem kell használnunk, amit használunk a távolságvektor alapúaknál, hogy elkerüljük a forgalomirányítási hurkokat. Ilyen például a RIP-nél az útvonalmérgezés vagy a láthatármegosztás.

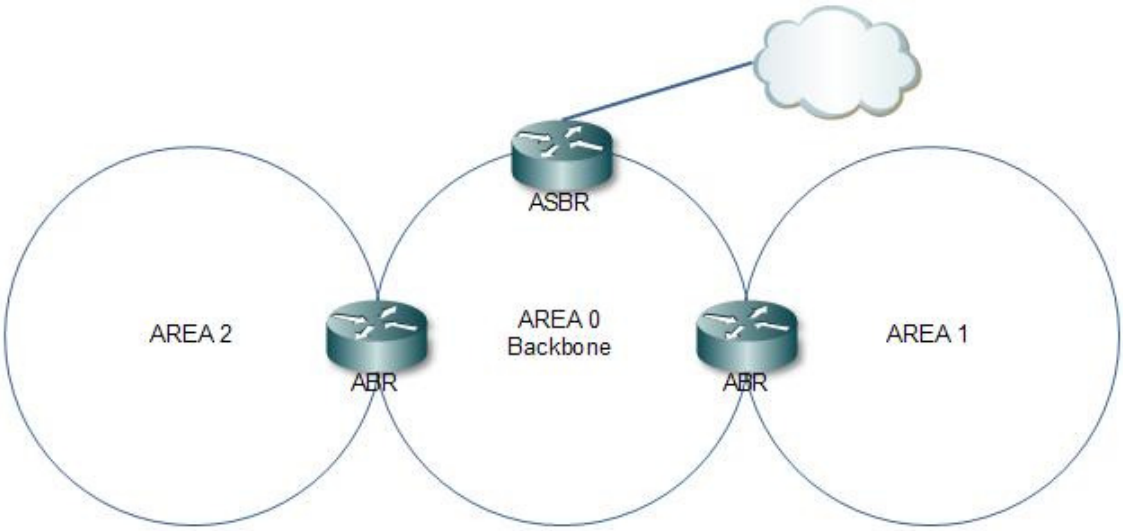
Az OSPF kapcsolatállapot alapú forgalomirányítási protokoll, autonóm rendszeren belül használjuk. Számos előnye van a távolságvektor alapú forgalomirányító protokollokkal szemben:

- Lehet summázni az útvonalakat
- Gyorsan konvergál
- Az OSPF szabvány, különböző gyártók termékei között is működik
- Hálózati sávszélességet használ
- Multicastot használ az üzenetszórás (broadcast) helyet

- Költséget használja, mint metrikát

A távolságvektor alapú forgalomirányítási protokollok tényleges utakat küldenek el a szomszédaiknak, nem úgy a kapcsolatállapot alapú társaik, amik egy listát továbbítanak a hozzájuk kapcsolódó hálózatokról. Ha egy kapcsolat állapotában változás történik (működik, nem működik), egy kapcsolatállapot hirdetés (LSA) generálódik. A távolságvektoros protokolloknál a szomszéd forgalomirányító nem más, mint egy közvetlenül csatlakoztatott eszköz vagy egy WAN (Wide Area Network) kapcsolat másik végén lévő forgalomirányító, és ezek egy hálózatba tartoznak. Minden forgalomirányító, amely egy tartományban van ilyen LSA-kal kommunikál. Minden forgalomirányító tárolja ezeket az LSA-kat egy táblában, amit kapcsolatállapot adatbázisnak (LSDB) nevezünk. Ezek az adatbázisok megegyeznek minden eszköznél egy tartományon belül. Ez után a forgalomirányítók lefuttatják az SPF algoritmust, hogy kiszámítsák az eddigi információikból a legjobb utakat, amik végül a forgalomirányítási táblába (routing table) is belekerülnek. (**[4]** Cisco.com, 2005)

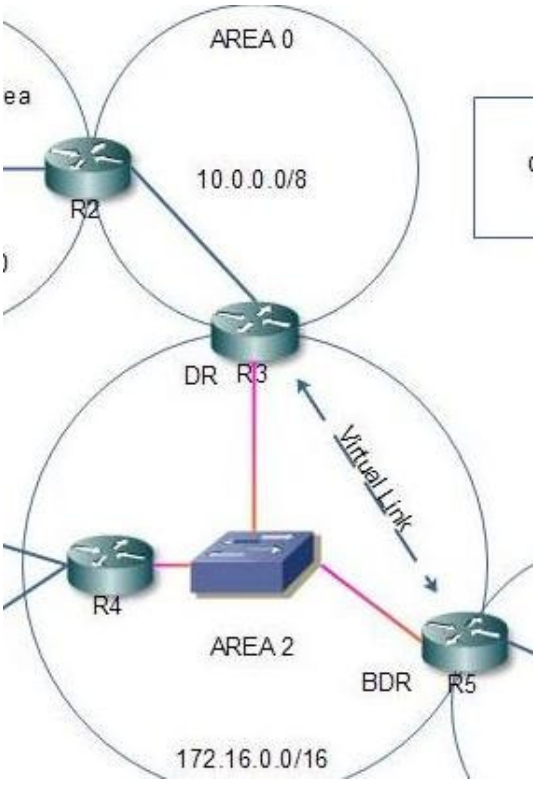
### 3.1.2. Az OSPF területek és ezek terminológiája




**2. ábra**

Alap koncepció az OSPF esetében, hogy minden terület a nullás, azaz gerinc területhez kell, hogy kapcsolódjon. Ha csak egy területről beszélünk, persze megengedett a nulláson kívül más használata is. A továbbiakban a több területű OSPF kerül előtérbe. Az egy területen (area) belül elhelyezkedő forgalomirányítóknak megegyezik a topológia táblájuk, azaz az egy területen elhelyezkedő forgalomirányítók ismerik a területen belül található összes hálózatot.

Ez minden azonos területen elhelyezkedő forgalomirányító esetében megegyezik. Ez azt jelenti, hogy ugyanazon információk szerint számolják ki a legjobb utakat. Maga a területes szerkezet a kezelhetőség miatt fontos. Ha minden forgalomirányító egy területhez tartozna, túl sok információt kellene feldolgozniuk. Erre gyengébb eszközök nem is lennének képesek, memória vagy processzor teljesítmény miatt. A területes szerkezetnél viszont elvi korlátja van, ami olyan 200. Így a frissítéseket lokalizálni lehet az egyes területeken. A területek határán álló forgalomirányítók az ABR-ek (Area Border Router) szűrik meg, hogy mely útvonalfrissítések kerüljenek be a területre, amit az ottani forgalomirányítók feldolgoznak. Az ABR-nek egy interfésze mindig a nullás területhez csatlakozik. A summázás miatt a több területes OSPF hierarchikus IP címzési sémát követel meg, hogy effektív legyen. Egy másik kinevezett forgalomirányító, ami különböző autonóm rendszereket köt össze, az ASBR (Autonomous System Boundary Router). Ennek egy interfésze az OSPF processzben vesz részt, míg másik interfésze egy másik forgalomirányító protokollt használ, mint például a RIP vagy a BGP. (**[7]** 3. fejezet), (**[1]** 187. oldal)




**3. ábra**

A topológiánkban az Area 1-et az R2 csatolja az Area 0-hoz. Az R3 pedig az Area 2-őt, valamint virtuálisan az Area 3-at is az R5-el, de erről a későbbiekben lesz szó. Az R2, R3 és

az R5 terület határ forgalomirányítók (ABR), míg az R4 egy autonóm rendszer határ forgalomirányító (ASBR), hiszen két másik autonóm rendszerhez csatlakozik.

### 3.1.3. Az OSPF szomszédsági viszony kialakítása

Ahhoz, hogy azonosítani tudjuk a felmerülő hibákat, majd megoldást találni rájuk, ismernünk kell a folyamat működését. Ilyen a szomszédsági viszony kialakítása is. Szomszédok egy területen belül alakulnak ki. Első lépésben szükségünk van egy állandó „névre" (rouer ID) amivel az adott forgalomirányítóra hivatkozni tudunk. Ez a legmagasabb IP számú aktív interfész, amennyiben nem található loopback interfész. Amennyiben több is található, akkor azok közül a legmagasabb IP című lesz az, amivel hivatkozni tudunk rá. A gyakorlati életben mindig loopback interfészt használunk. Ennek legfőbb oka, hogy loopback interfész csak abban az esetben nem elérhető, ha maga a forgalomirányító is elérhetetlen a hálózaton keresztül. Ha olyan interfészt használnánk, ami például vonalhiba miatt deaktiválódhat, a forgalomirányító kiesne az OSPF folyamatból. Ez az azonosító csak akkor változik meg, amennyiben az eszköz újraindul, vagy az OSPF folyamatot indítjuk újra.

#### A HELLO csomagok

Ezeket a csomagokat alapértelmezés szerint 10 másodpercenként küldi ki a forgalomirányító üzenetszórásos illetve pont-pont kapcsolat esetén. Ez az idő 30 másodpercre nő a nem üzenetszórásos, többes elérésű hálózatoknál, mint például a Frame-Relay vagy az ATM. Ezeket az intervallumokat természetesen tudjuk finom hangolni a legjobb eredmény elérése érdekében. Létezik egy úgy nevezett DEAD TIMER, ami a HELLO TIMER négyszerese. Ez azt jelenti, hogy 4 HELLO csomag kihagyás után a szomszédsági viszony megszűnik. Ahhoz, hogy kialakulhasson a szomszédsági viszony, első lépésben a forgalomirányítók HELLO illetve DEAD intervallumainak meg kell egyeznie. Szükséges még ugyanazon alhálózatba tartozniuk, és egy területben kell lenniük. Valamint ha állítottunk be jelszót, akkor azoknak is természetesen meg kell egyezniük. Ezek az információk mind a HELLO csomagban kapnak helyet.

Magát a folyamatot úgy aktiválhatjuk, hogy felvesszük azokat az interfészeket, amelyeken hirdetni szeretnénk az OSPF-et. Ezek így a kapcsolatállapot adatbázisba kerülnek. A későbbiekben ezeken az interfészeken fog a forgalomirányítónk HELLO csomagokat

kiküldeni. Ekkor még a szomszédsági viszony nem alakul ki, ugyanis ezekre válaszolni is kell valakinek. DOWN állapotban lesz az első HELLO csomag kiküldése után.

A folyamat első lépéseként, amint az egyik forgalomirányító kap egy HELLO csomagot, INIT állapotba kerül. Ekkor még nem alakult ki a szomszédsági viszony, csak ellenőrzi a megfelelő paraméterek egyezését.

Ha minden adat megegyezik, válaszol az üzenetre. Itt végre kell hajtania egy keresést. Először ellenőrzi a szomszédsági mezőt (NEIGHBOR) a HELLO csomagban, hogy a feladója szerepel-e már benne. Amennyiben megtalálja, nullázza a DEAD időzítőt. Ez azt jelenti, hogy a szomszéd még mindig elérhető és válaszol, tehát funkcionál.

Ha nem volt benne a mezőben, akkor új szomszédsági viszony kialakítás kezdődik el. Ilyenkor egy Mester- Szolga viszony áll fent a két forgalomirányító berendezés között. Ekkor kerül EXSTART állapotba. A Mester- Szolga viszonyra azért van csak szükség, hogy meghatározzák, ki küldi először az információt. Hogy ki legyen a Mester vagy a Szolga, az több dologtól függ. Először a prioritást ellenőrzi, ez alapállapotban 1, és nem nagyon szokták módosítani. Ha megegyezik a prioritás, akkor a forgalomirányító azonosító (ROUTER ID) dönt. Akinek nagyobb, az lesz a Mester. Miután kialakult a hierarchikus viszony, először a Mester majd a Szolga is elküld egy adatbázis leírást (DATABASE DESCRIPTION). Erre azért van szükség, mert lehet, hogy csak néhány dologról nem tud a forgalomirányító, így elég csak azokat elküldeni.

Ezután LOADING állapotba kerülnek. Ilyenkor történik az adatok szinkronizációja. Ezt a Szolga kezdi el, küld egy kérelmet, amire a Mester válaszol, majd ő kér. A kérelem az LSR (LINK STATE REQUEST), a válasz pedig az LSU (LINK STATE UPDATE). Ezek az LSU és LSR üzenetek nem egyesével történnek, hanem bennük sok LSA (LINK STATE ADVERTISEMENT) található. Az összes LSA, amit kér a Szolga az elején egy LSR-ben kerül kiküldésre, majd a Mester is egy olyan LSU-val válaszol, amiben benne van az összes előző LSR-re az LSA. (**[2]** 166. oldal)

Ez addig folytatódik, amíg a két forgalomirányítónak meg nem egyezik a kapcsolatállapot adatbázisa. Ekkor FULL állapotba kerülnek, és beszélhetünk szomszédsági viszonyról.

A gyakorlatban ezeket az állapotokat megfigyelhetjük a forgalomirányítókon is. A példában az R2 forgalomirányítón nézzük meg, amíg kiépíti a kapcsolatot az R1-el. A R2-n a ***debug ip***

***ospf events***, valamint a ***debug ip ospf adj*** parancsokat adtam ki. Az eredményt a log fájlban láthatjuk, amit a ***show logging*** paranccsal tekinthetünk meg.

```
R2#sh log | i OSPF
*Nov 14 17:52:27.451: OSPF: Interface Serial1/0 going Up
*Nov 14 17:52:27.455: OSPF: Send hello to 224.0.0.5 area 1 on Serial1/0 from 192.168.0.2
*Nov 14 17:52:27.627: OSPF: Rcv hello from 192.168.3.1 area 1 from Serial1/0 192.168.0.1
*Nov 14 17:52:27.631: OSPF: 2 Way Communication to 192.168.3.1 on Serial1/0, state 2WAY
*Nov 14 17:52:27.631: OSPF: Send DBD to 192.168.3.1 on Serial1/0 seq 0x1F3E opt 0x50 flag
0x7 len 32
*Nov 14 17:52:27.635: OSPF: Send immediate hello to nbr 192.168.3.1, src address
192.168.0.1, on Serial1/0
*Nov 14 17:52:27.639: OSPF: Send hello to 224.0.0.5 area 1 on Serial1/0 from 192.168.0.2
*Nov 14 17:52:27.639: OSPF: End of hello processing
*Nov 14 17:52:27.703: OSPF: Rcv DBD from 192.168.3.1 on Serial1/0 seq 0x19A4 opt 0x50 flag
0x7 len 32  mtu 1500 state EXSTART
*Nov 14 17:52:27.707: OSPF: First DBD and we are not SLAVE
*Nov 14 17:52:27.707: OSPF: Rcv DBD from 192.168.3.1 on Serial1/0 seq 0x1F3E opt 0x50 flag
0x2 len 52  mtu 1500 state EXSTART
*Nov 14 17:52:27.707: OSPF: NBR Negotiation Done. We are the MASTER
*Nov 14 17:52:27.707: OSPF: Send DBD to 192.168.3.1 on Serial1/0 seq 0x1F3F opt 0x50 flag
0x3 len 72
*Nov 14 17:52:27.731: OSPF: Rcv DBD from 192.168.3.1 on Serial1/0 seq 0x1F3F opt 0x50 flag
0x0 len 32  mtu 1500 state EXCHANGE
*Nov 14 17:52:27.731: OSPF: Send DBD to 192.168.3.1 on Serial1/0 seq 0x1F40 opt 0x50 flag
0x1 len 32
*Nov 14 17:52:27.731: OSPF: Send LS REQ to 192.168.3.1 length 12 LSA count 1
*Nov 14 17:52:27.763: OSPF: Rcv LS REQ from 192.168.3.1 on Serial1/0 length 48 LSA count 2
*Nov 14 17:52:27.763: OSPF: Send UPD to 192.168.0.1 on Serial1/0 length 68 LSA count 2
*Nov 14 17:52:27.767: OSPF: Rcv DBD from 192.168.3.1 on Serial1/0 seq 0x1F40 opt 0x50 flag
0x0 len 32  mtu 1500 state EXCHANGE
*Nov 14 17:52:27.771: OSPF: Exchange Done with 192.168.3.1 on Serial1/0
*Nov 14 17:52:27.779: OSPF: Rcv LS UPD from 192.168.3.1 on Serial1/0 length 100 LSA count 1
*Nov 14 17:52:27.783: OSPF: Synchronized with 192.168.3.1 on Serial1/0, state FULL
*Nov 14 17:52:27.787: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.3.1 on Serial1/0 from LOADING
to FULL, Loading Done
*Nov 14 17:52:27.955: OSPF: Build router LSA for area 1, router ID 192.168.4.1, seq
0x80000005
*Nov 14 17:52:28.327: OSPF: Rcv LS UPD from 192.168.3.1 on Serial1/0 length 112 LSA count 1
*Nov 14 17:52:35.171: OSPF: Rcv hello from 192.168.3.1 area 1 from Serial1/0 192.168.0.1
*Nov 14 17:52:35.175: OSPF: End of hello processing
*Nov 14 17:52:37.455: OSPF: Send hello to 224.0.0.5 area 1 on Serial1/0 from 192.168.0.2
*Nov 14 17:52:45.123: OSPF: Rcv hello from 192.168.3.1 area 1 from Serial1/0 192.168.0.1
*Nov 14 17:52:45.127: OSPF: End of hello processing
*Nov 14 17:52:47.455: OSPF: Send hello to 224.0.0.5 area 1 on Serial1/0 from 192.168.0.2
*Nov 14 17:52:55.155: OSPF: Rcv hello from 192.168.3.1 area 1 from Serial1/0 192.168.0.1
*Nov 14 17:52:55.159: OSPF: End of hello processing
*Nov 14 17:52:57.455: OSPF: Send hello to 224.0.0.5 area 1 on Serial1/0 from 192.168.0.2
R2#
```

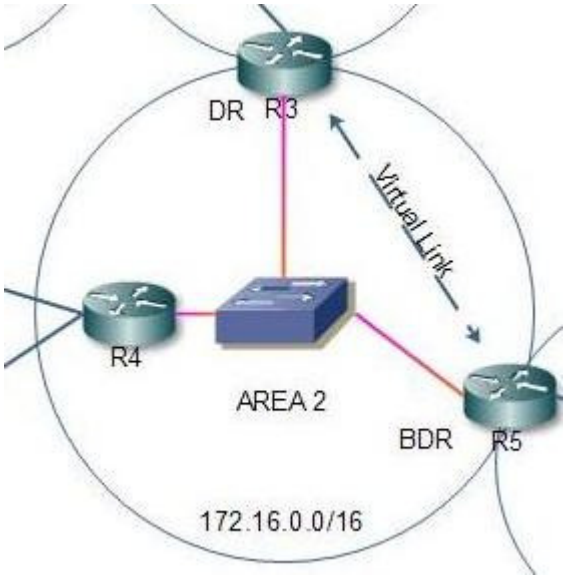
**4. ábra**

Amint az interfész, ami szerepel az OSPF folyamatban, FEL állapotba kerül, kiküldi az első HELLO csomagot rajta. A cél cím pedig a 224.0.0.5 multicast cím, ami minden forgalomirányítónak szól a szegmensben. Végigkövethetjük a Mester/Szolga (MASTER/SLAVE) viszony kialakítását és az információk cseréjének folyamatát. Végül pedig FULL állapotba kerül, amint az információk cseréje lezajlott. Az utolsó pár sorban pedig az Életben tartó hello csomagokat láthatjuk, amint kiküldésre kerülnek a multicast címre, majd pedig a szomszéd válaszol rá.

### 3.1.4. DR és BDR

Egy szegmens felesleges forgalom nélküli irányításához speciális forgalomirányítókra van szükségünk. Ezt a feladatot látja el a Designated Router (DR) és a Backup Designated Router

(BDR). Egy szegmensben csak egy DR és ennek a tartaléka, a BDR van. Hogy melyik forgalomirányítónk lesz a DR illetve a BDR, hasonlóan dől el, mint a Mester- Szolga viszony az előbb tárgyalt fejezetben. Először a prioritást ellenőrzi, ha nem tud dönteni, akkor a forgalomirányító azonosító (Router ID) alapján választ. A pont-pont kapcsolatoknál nincs szükség DR-re és BDR-re, mert ott csak egy eszköz lehet a másik oldalon. A szomszédsági viszony más állapotokon megy végig, mint ahol nincs DR és BDR. A DOWN és az INIT állapot itt is megtalálható, viszont a többi már nem. Itt nem minden forgalomirányítóval kerül FULL kapcsolatba. Ilyen teljes kapcsolatot csak a DR-el és a BDR-el épít ki. Az összes többivel úgynevezett 2WAY állapotba kerül. Ez jelen esetben nem jelenti azt, hogy rosszul működik, éppen ellenkezőleg, teljes kapcsolatot kell kiépíteni a DR-el és a BDR-el, mert az információkat tőlük kapják meg, nem közvetlenül attól a forgalomirányítótól, akivel a kapcsolatot akarnak kiépíteni. Így sok felesleges forgalmat lehet megspórolni, mert mindenki csak a kitüntetett forgalomirányítókkal kommunikál, és nem kell minden más eszköznek is elküldeni ugyanazt az információt, majd a DR ezt megteszi helyette. Erre multicast üzenetet használ. Amikor a kitüntetett forgalomirányítóknak küldünk üzenetet a 224.0.0.6 multicast címet használjuk. Amikor a DR küld információt, a 224.0.0.5 címet használja, hogy elérje a többi forgalomirányítót a szegmensben. Pont- pont kapcsolat esetében mindig a 224.0.0.5 címet használják. (**[1]** 133. oldal)




**5. ábra**

A DR/BDR választás a topológiánkban az Area 2-ben figyelhető meg. Itt nem pont- pont kapcsolatok vannak, mint a többi területnél, hanem egy kapcsolóhoz vannak csatlakoztatva a

forgalomirányítók. A felhasznált IP címtartomány a 172.16.0.0/16. A DR/BDR választást manipulálhatjuk a három forgalomirányító között. Ami eldönti, ki lesz a DR valamint a BDR, az első körben a forgalomirányító prioritása. Ez alap esetben 1, így utána a ***router-id*** vizsgálata következik. Ha nem változtattam volna meg a prioritásokat, akkor az R5 lenne a DR, az R4 pedig a BDR. A prioritások megváltoztatásával viszont az R3 lett a DR, 10-es, az R5 a BDR, 5-ös prioritással, az R4 pedig nem vesz részt ebben a választásban, hiszen a prioritása nulla. Ezeket az interfészeken lehet beállítani, amik abba a szegmensbe tartoznak ahol a DR/BDR választás végbemegy, az ***ip ospf priority 5*** paranccsal.

```
R4#sh run int fa 0/0
interface FastEthernet0/0
 description *** AREA 2 TO SW ***
 ip address 172.16.0.4 255.255.255.0
 ip ospf priority 0
 duplex auto
 speed auto
```

```
R5#sh run int fa 0/0
interface FastEthernet0/0
 description *** AREA 2 TO SW ***
 ip address 172.16.0.5 255.255.255.0
 ip ospf priority 5
 duplex auto
 speed auto
```

**6. ábra**

Ezt ellenőrizni a ***show ip ospf neighbor*** paranccsal tudjuk. Az R3 forgalomirányítón megfigyelhetjük, hogy. ***FULL/BDR*** kapcsolat alakul ki a BDR-el, aminek azonosítója 5.5.5.5, ***FULL/DROTHER*** pedig az összes többivel, amik a szegmensbe tartoznak.

```
R3#sh ip ospf neigh

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.4.1       0   FULL/  -        00:00:37    10.0.0.1        Serial1/1
5.5.5.5           0   FULL/  -           -        172.16.0.5      OSPF_VL0
4.4.4.4           0   FULL/DROTHER    00:00:33    172.16.0.4      FastEthernet0/0
5.5.5.5           5   FULL/BDR        00:00:31    172.16.0.5      FastEthernet0/0
```

**7. ábra**

Jelen esetben ez csak egy forgalomirányító, az R4, aminek azonosítója a 4.4.4.4. Mivel nem található ***FULL/DR***, ebből arra következtethetünk, hogy az R3 a DR. Ez be is bizonyosodik, amint az R4-en is kiadjuk a parancsot. Látható, hogy a 3.3.3.3 azonosítóval rendelkező forgalomirányítónk a DR, mivel ennek a legnagyobb a prioritása, 10.

```
R4#sh ip ospf neigh

Neighbor ID     Pri   State           Dead Time   Address         Interface
3.3.3.3          10   FULL/DR         00:00:38    172.16.0.3      FastEthernet0/0
5.5.5.5           5   FULL/BDR        00:00:34    172.16.0.5      FastEthernet0/0
```

**8. ábra**

### 3.1.5. OSPF csomagtípusok

Minden OSPF forgalom IP csomagok belsejében utazik. Akik megkapják, onnan ismerik fel, hogy OSPF csomagot kaptak, mert 89-es IP protokollnak van jelölve.

- HELLO csomag

  Közvetlenül kapcsolatban álló szomszédokkal alakít ki kommunikációs kapcsolatot.

- Adatbázis leíró csomag (DBD)

  Egy listában elküldi azoknak a forgalomirányítóknak az azonosítóját, akiktől LSA-t kapott, valamint a jelenlegi szekvencia számot. Ezt arra használják, hogy összevessék a tudásukat a hálózatról.

- Kapcsolatállapot kérelem (LSR)

  A DBD-k után rögtön ezeket küldi a forgalomirányító, ha valamelyik LSA hiányzik neki.

- Kapcsolatállapot frissítés (LSU)

  Ebben a formában válaszol az előbb kért adatokra.

- Kapcsolatállapot nyugtázás (LSAck)

  Megerősíti a kapott kapcsolatállapot információt.

Minden OSPF csomagnak ugyanaz a formátuma, és a következő 9 mezőből állnak:

<table>
<tr><td>Verzió<br>IPv4 esetében a verziószám 2. IPv6 esetében 3.</td><td>Típus<br>5 csomagtípust különböztetünk meg, amiket az előbb említettem.</td><td>Csomag hossza<br>Bájtokban megadva.</td></tr>
<tr><td colspan="3">Forgalomirányító azonosító<br>Egy 32 bites azonosító, ami a neve lesz a forgalomirányítónak.</td></tr>
<tr><td colspan="3">Területazonosító<br>Szintén egy 32 bites azonosító, ami a területet jelöli.</td></tr>
<tr><td colspan="2">Ellenőrző összeg<br>Szabványos, 16 bites ellenőrző összeg.</td><td>Hitelesítés típusa<br>3 fajtáját különbözteti meg az OSPF második verziója, amikről a későbbiekben lesz szó.</td></tr>
<tr><td colspan="3">Hitelesítő adat<br>64 bites adat, ami lehet üres, sima szöveg vagy egy leképezése (hash) a közös jelszónak.</td></tr>
<tr><td colspan="3">Adat<br>Változó hosszúságú lehet.</td></tr>
</table>

([**1**] 138. oldal)

### 3.1.6. OSPF hálózat típusok

Több féle hálózattípusról beszélhetünk, ahol megjelenik az OSPF.

- Üzenetszórásos, többes elérésű hálózatok

  Ilyen például az Ethernet vagy a Token- Ring. Megtörténik a DR és a BDR választás, 10 másodpercenként küldi a HELLO csomagokat, két multicast címet használ.

- Pont- pont hálózatok

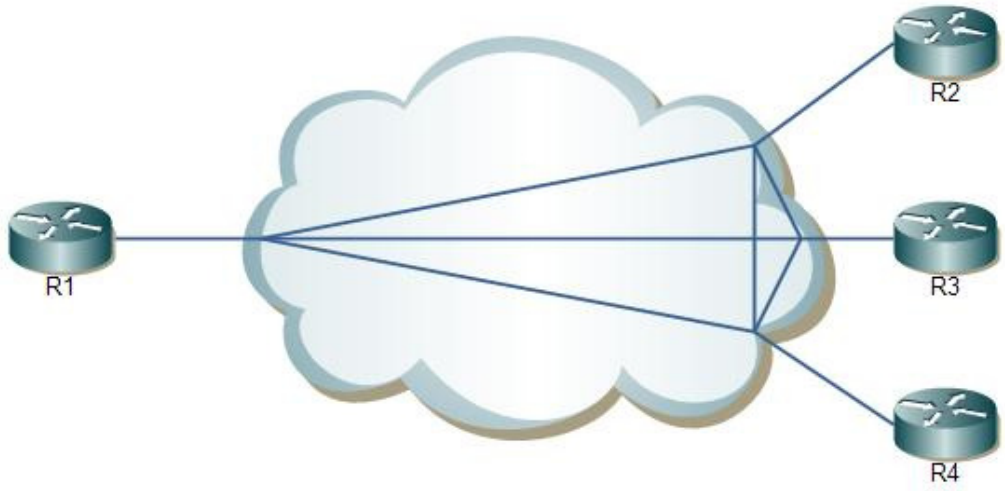
  Például a bérelt vonalak tipikusan pont- pont hálózatok. Nem történik DR, BDR választás, itt is 10 másodperces HELLO csomagok vannak, de csak egy multicast címet használ (224.0.0.5)

- Nem üzenetszórásos, többes elérésű hálózatok (NBMA)

  Frame-Relay és az ATM ilyen elven működnek. Öt féle módon történhet az implementációja, amire a későbbiekben térek ki.

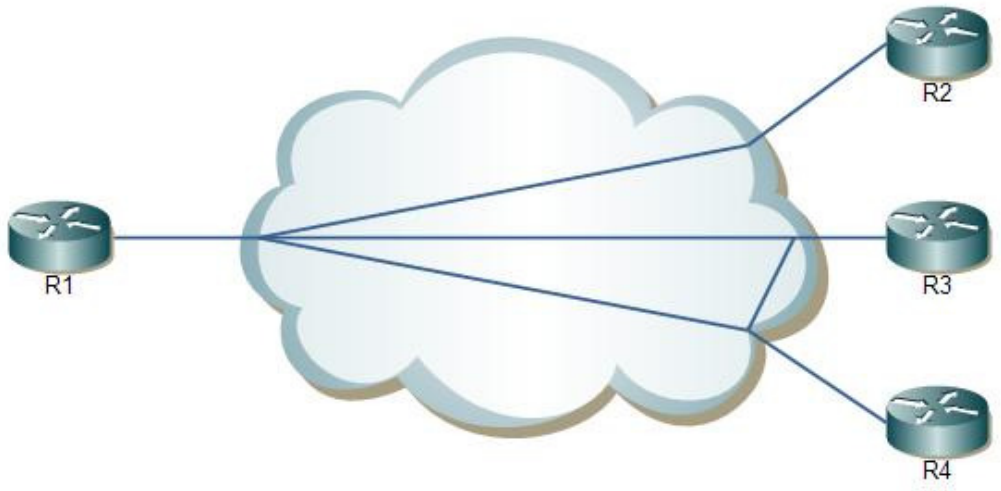
## OSPF NBMA hálózatokon

Nem üzenetszórásos, többes elérésű hálózatoknál (NBMA) több féle topológia fordulhat elő. Mindnek vannak természetesen előnyei és hátrányai.



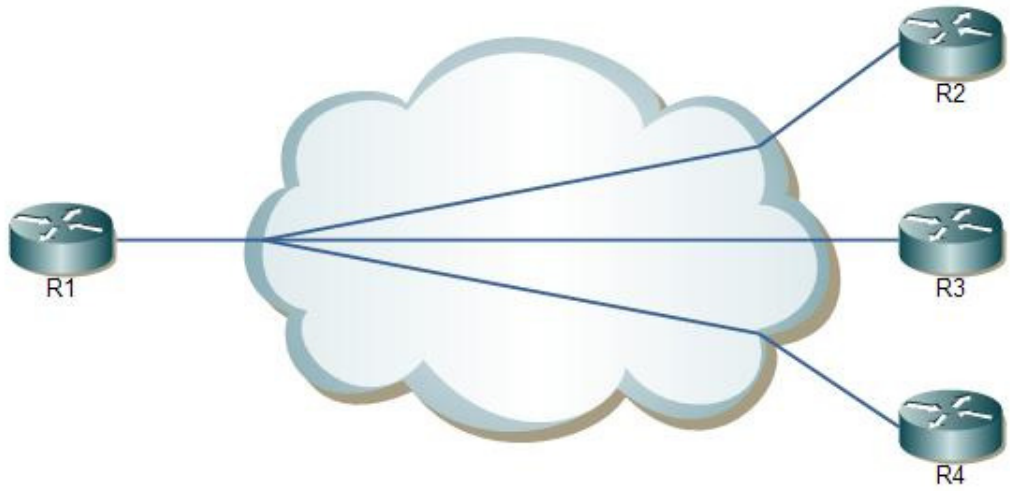

**9. ábra**

A 9. ábrán látható, a teljes összeköttetést biztosító topológia. Előnye, hogy mindenkinek mindenkivel közvetlen kapcsolata van, ezért roppant mód hibatűrő, a késleltetés ideális lehet például VoIP-ra. Hátránya viszont, hogy drága.




**10. ábra**

Az előbbi hátrányból, hogy drága, jelenthet megoldást a 10. ábrán látható topológia, a részleges összekapcsolás. Van benne hibatűrés és olcsóbb, mint a teljes összeköttetésű topológia.




**11. ábra**

Végül pedig a Hub.and-spoke, vagy más néven csillag topológia (11. ábra). Itt minden telephely egy központi irodához csatlakozik. Egymást is csak ezen keresztül érik el, ebből kifolyólag magas is lehet a késleltetés, ami alkalmatlanná teheti VoIP szempontjából. Teljesen redundancia mentes, amennyiben egy kapcsolat meghibásodik, akkor a hozzá kapcsolódó telephely teljesen elérhetetlenné válik. (**[2]** 177. oldal)

### 3.1.7. OSPF LSA típusok és területek

Ahhoz, hogy rendesen fel tudjunk építeni egy optimálisan működő, több területből álló OSPF-et, ismernünk kell a különböző LSA típusokat, és hogy melyik miért felelős. Fontosak a lesznek későbbiekben, amikor az utak summázásáról beszélünk majd, és főleg azért, mert bizonyos területtípusok ezek az LSA-k alapján szűrik meg, hogy mit engednek át, és mit nem. Persze magában az OSPF folyamat felállításában is részt vesznek, ezért is fontos szót ejteni róluk. (**[1]** 187. oldal)

- 1. típusú – Router LSA

  Ez a fajta egy hálózatot hirdet, lényegében egy forgalomirányító frissítés.

- 2. típusú – Network LSA

Ezt a kinevezett forgalomirányító (Designated Router - DR) generálja. Ebből kifolyólag egy Ethernet szegmensben lévő forgalomirányítók kapják meg, és azt tartalmazza, hogy abban a bizonyos szegmensben mely forgalomirányítók találhatóak meg.

- 3. típusú – Summary LSA

  Ez az ABR-től származó summázott útvonalfrissítéseket tartalmazza

- 4. típusú – Summary LSA

  Bár a nevéből arra következtethetnénk, hogy ez is valamiféle summázás, de nem. Ebben az LSA-ban csak és kizárólag az ASBR IP címe található meg.

- 5. típusú – External LSA

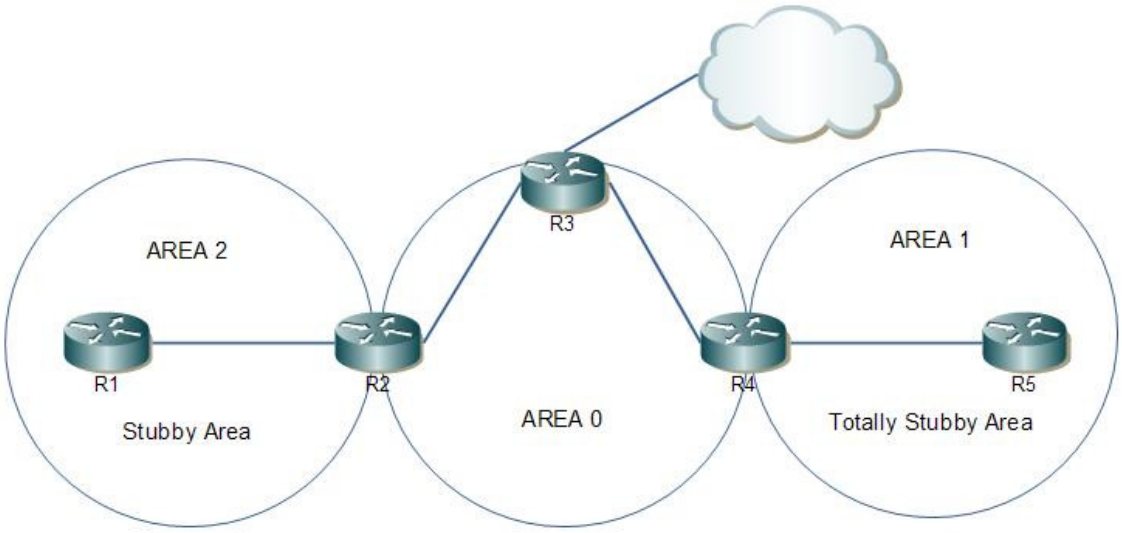
  ASBR küldi, és azokat a külső útvonalakat tartalmazza, amiket az ASBR az OSPF folyamatba hirdet.

A következőkben a nevesített OSPF területekről lesz szó. Tisztán CISCO környezetben 3 ilyen területtípusról beszélhetünk, mindegyiknek megvannak a sajátosságai, amiért érdemes lehet őket használni. Az utak summázása tulajdonképpen az előbb említett 5 LSA típussal történik. Bizonyos területek blokkolnak bizonyos típusú LSA-kat a belépéstől. Amiért jól használhatóak ezek a terület típusok, mert csak az ABR és ASBR esetében van szükségünk viszonylag erős, drága forgalomirányítókra, mert a kiépítésből kifolyólag ezek veszik le a terhet a mögöttük található forgalomirányítókról. Ez viszont kétélű fegyver, mert csak bizonyos esetekben, ha a területről csak egy forgalomirányítón keresztül juthatunk ki, működik úgy, ahogyan kellene. Ha egy területnek több kijárata is van, mert például meg szeretnénk osztani a forgalmat a még jobb kihasználtság érdekében, akkor nem fog rendeltetésszerűen működni. Ennek az az oka, hogy a belső forgalomirányítóknak nincs fogalmuk arról, hogy a „külvilágban” mi található. Minden ilyen döntést az ABR hoz meg helyettük.

### Stubby és Totally Stubby terület típusok

Amennyiben nem használnánk ezeket a területtípusokat, akkor minden egyes forgalomirányító, minden egyes útvonalról tudna. Ez egy nagy és komplex, sok forgalomirányítós hálózatban problémát is okozhatna. Az 12. ábrán látható R1

forgalomirányítónak teljesen felesleges tudni arról, mi van a saját területén kívül, mert azt csak és kizárólag az R2-n keresztül érheti el.

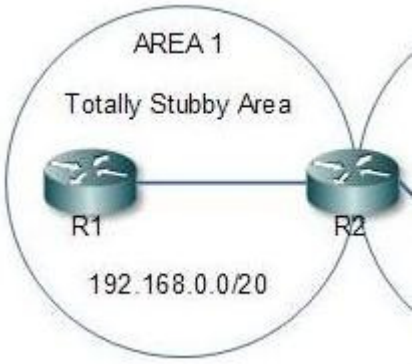



**12. ábra**

A Stubby area blokkolja az 5. típusú LSA-kat, hogy belépjenek a területre.

A Totally Stubby area pedig mind a 3.– 4.– 5. típusú LSA-kat blokkolja, amint belépni próbálnának a területre.

Tehát, a Stubby area tudni fog az OSPF folyamat összes útvonaláról, kivéve a külső útvonalakról, amik bekerültek. A nullás, azaz gerinc terület nem lehet Stubby area. A Totally Stubby area csak a saját területén belüli dolgokról fog tudni, semmi másról. Ez egyébként Cisco specifikus, tehát a Totally Stubby ABR-nek Cisco forgalomirányítónak kell lennie, viszont a mögötte lévőeknek nem, mert azok végül is akár egy Stubby területen is lehetnek, az ABR elvégzi a szűrést helyettük. (**[1]** 188. oldal)




**13. ábra**

A hálózatunkban az Area 1, ahogy az ábrán is látszik egy Totally Stubby Area, ami azt jelenti, hogy nem tudja, mi van a saját területén kívül, csak az Area 1-ről van fogalma. A hierarchikus dizájn miatt, ebben a területben az IP címek a 192.168.0.0/20-as hálózatból kerülnek ki. Ez majd a summázásnál lesz segítségünkre. Az R1 és R2 forgalomirányító között a 192.168.0.0/30-as hálózatot használtam, mivel pont- pont kapcsolatról van szó, így nem pazarolunk IP címet. Mind a két forgalomirányítón konfiguráltam még visszacsatoló (loopback) interfészeket, amik a belső hálózatot imitálják. Ezeket is természetesen hirdetem az OSPF folyamatban. Az ábrán láthatóak a konfigurált interfészek és a hozzájuk tartozó IP címek.

```
R1#sh ip int brief
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            unassigned      YES NVRAM  administratively down down
Serial1/0                  192.168.0.1     YES NVRAM  up                    up
Serial1/1                  unassigned      YES NVRAM  administratively down down
Serial1/2                  unassigned      YES NVRAM  administratively down down
Serial1/3                  unassigned      YES NVRAM  administratively down down
Loopback1                  192.168.1.1     YES NVRAM  up                    up
Loopback2                  192.168.2.1     YES NVRAM  up                    up
Loopback3                  192.168.3.1     YES NVRAM  up                    up
R1#
```

```
R2#sh ip int brief
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            unassigned      YES NVRAM  administratively down down
Serial1/0                  192.168.0.2     YES NVRAM  up                    up
Serial1/1                  10.0.0.1        YES NVRAM  up                    down
Serial1/2                  unassigned      YES NVRAM  administratively down down
Serial1/3                  unassigned      YES NVRAM  administratively down down
Loopback4                  192.168.4.1     YES NVRAM  up                    up
Loopback10                 10.1.1.1        YES NVRAM  up                    up
Loopback11                 10.2.2.1        YES NVRAM  up                    up
R2#
```

**14. ábra**

Ahhoz, hogy a terület egy Totally Stubby Area (TSA) legyen, mind a két forgalomirányítón ki kell adnunk az ***area 1 stub no-summary*** parancsot. Az R2 forgalomirányítón kell beállítanunk a summázást, mert az ABR, amit a többieknek hirdetni fog az Area 1-ről, ez pedig az ***area 1 range 192.168.0.0 255.255.240.0*** parancs eredménye. A ***network*** sorok azokat az interfészeket adják meg, amik részt vesznek az OSPF folyamatban. Itt inverz maszkot (wildcard mask) kell megadnunk, és a területet is meg kell adnunk, hogy melyikhez tartozik.

```
R1#sh run | b router ospf
router ospf 1
 log-adjacency-changes
 area 1 stub no-summary
 network 192.168.0.0 0.0.0.3 area 1
 network 192.168.1.1 0.0.0.0 area 1
 network 192.168.2.1 0.0.0.0 area 1
 network 192.168.3.1 0.0.0.0 area 1
```

```
R2#sh run | b router ospf
router ospf 1
 log-adjacency-changes
 area 1 stub no-summary
 area 1 range 192.168.0.0 255.255.240.0
 network 10.0.0.0 0.255.255.255 area 0
 network 192.168.0.0 0.0.0.3 area 1
 network 192.168.4.1 0.0.0.0 area 1
```

**15. ábra**

Amint ezekkel elkészültünk, a show ip route paranccsal bizonyosodhatunk meg arról, hogy mindent helyesen állítottunk be az R1-en.

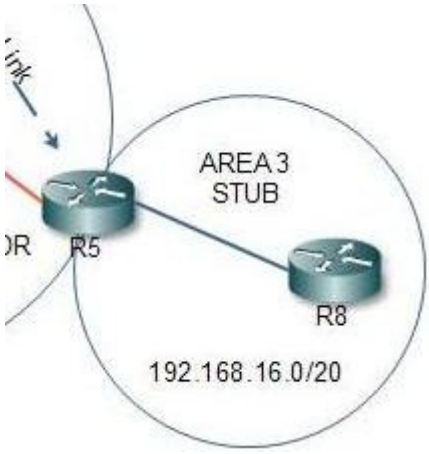
```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.0.2 to network 0.0.0.0

     192.168.4.0/32 is subnetted, 1 subnets
O       192.168.4.1 [110/65] via 192.168.0.2, 00:01:33, Serial1/0
     192.168.0.0/30 is subnetted, 1 subnets
C       192.168.0.0 is directly connected, Serial1/0
     192.168.1.0/32 is subnetted, 1 subnets
C       192.168.1.1 is directly connected, Loopback1
     192.168.2.0/32 is subnetted, 1 subnets
C       192.168.2.1 is directly connected, Loopback2
     192.168.3.0/32 is subnetted, 1 subnets
C       192.168.3.1 is directly connected, Loopback3
O*IA 0.0.0.0/0 [110/65] via 192.168.0.2, 00:01:33, Serial1/0
```

**16. ábra**

A táblában nem található a terület határain túlmutató hálózat, csak az R4 által hirdetett átjáró. Mindent az R2-n keresztül ér el, a 192.168.0.2 IP címen. Ez a Totally Stubby Area lényege, hogy területen belüli forgalomirányító, mint itt az R1, csak nagyon kevés információval rendelkezik. Mindent csak az R2-nek továbbit, amivel nem tud mit kezdeni, és hagyja, hogy az R2 hozza meg a döntést. Így a belső forgalomirányítóknak (R1) nem kell olyan nagy teljesítményűnek lenniük, így pénzt spórolhatunk meg.




**17. ábra**

A hálózatunkban a Stubby Area alkalmazása az Area 3-ban kerül elő. Itt a külső útvonalak nem találhatóak a forgalomirányító táblájában az R8-nak, csak az OSPF területekről tud. Ez annyiban különbözik az R1-étől, hogy itt találunk még plusz információkat, de csak az OSPF területekről (***IA*** bejegyzések). A Stub területnek az a sajátossága, hogy a külső utakat nem engedi meg belépni a területre. Így a táblából hiányzik az R5-nél például megtalálható sor.

***O E1 200.0.1.1 [110/445] via 172.16.0.4, 00:02:14, FastEthernet0/0***

```
R8#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.16.1 to network 0.0.0.0

O IA 172.16.0.0/16 [110/65] via 192.168.16.1, 00:07:22, Serial1/0
O IA 10.0.0.0/8 [110/66] via 192.168.16.1, 00:07:13, Serial1/0
     192.168.17.0/32 is subnetted, 1 subnets
C       192.168.17.1 is directly connected, Loopback17
     192.168.16.0/30 is subnetted, 1 subnets
C       192.168.16.0 is directly connected, Serial1/0
     192.168.19.0/32 is subnetted, 1 subnets
O       192.168.19.1 [110/65] via 192.168.16.1, 00:07:23, Serial1/0
     192.168.18.0/32 is subnetted, 1 subnets
C       192.168.18.1 is directly connected, Loopback18
O*IA 0.0.0.0/0 [110/65] via 192.168.16.1, 00:07:23, Serial1/0
O IA 192.168.0.0/20 [110/130] via 192.168.16.1, 00:02:47, Serial1/0
```

**18. ábra**

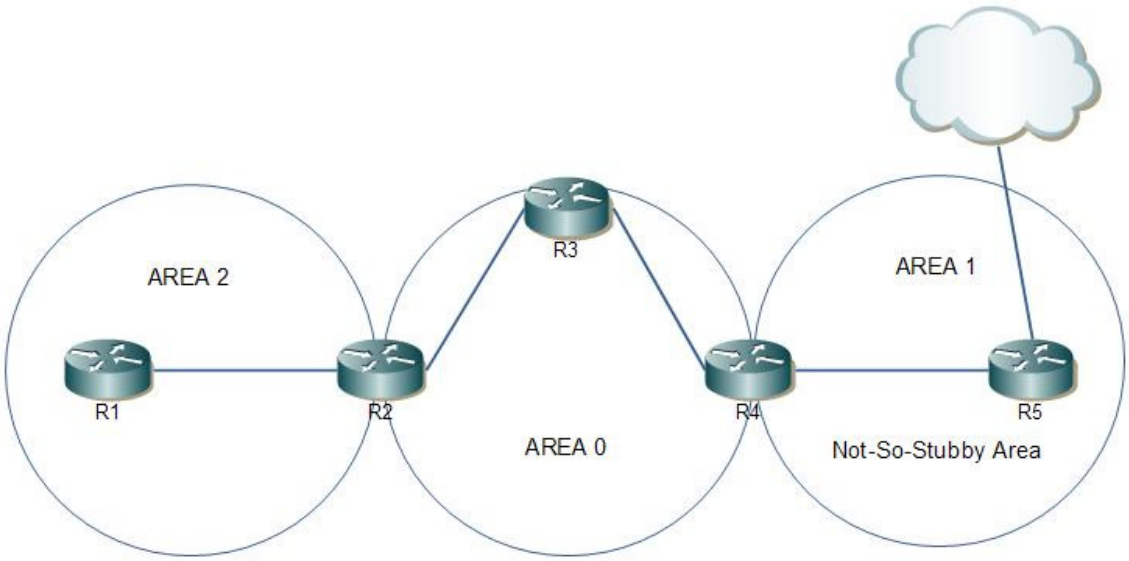
Konfigurációja hasonló a TSA-hoz, azzal a különbséggel, hogy itt a parancs csak annyi, hogy ***area 3 stub***. Hiányzik a ***no-summary*** kulcsszó, ami a TSA terület beállításához kellett. Ezt mind az R5 és az R8 forgalomirányítón is be kell állítanunk. Ez a terület a 192.168.16.0/20-as IP címtartományt használja.

```
R8#sh run | b router ospf
router ospf 1
 log-adjacency-changes
 area 3 stub
 network 192.168.16.0 0.0.0.3 area 3
 network 192.168.17.1 0.0.0.0 area 3
 network 192.168.18.1 0.0.0.0 area 3
!
```

**19. ábra**

## Not-So-Stubby Area (NSSA)

A példa szerint itt R5 gyakorlatilag egy ASBR, mert külső hálózatot kapcsol az OSPF folyamatba, de a területe Totally Stubby Area, ami megakadályozza, hogy bármi mást is beengedjen a területre, ami nem oda való (20. ábra). Erre ad megoldást a Not-So-Stubby Area. Ami nem csinál mást, mint továbbítja a külső útvonalakat. Ezt pedig úgy éri el, hogy az 5. típusú LSA-kat egy 7. típusú LSA-ba csomagolja, amik amint elérik a gerinc területet visszaalakulnak 5. típusúvá. (**[1]** 188. oldal)

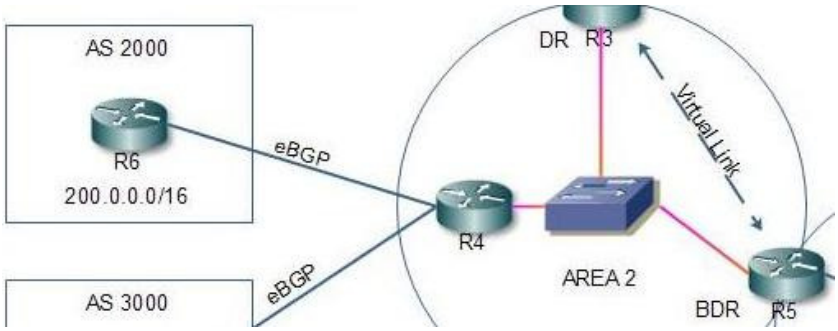



**20. ábra**

## Külső útvonal típusok

A külső útvonalak, amik az ASBR-en keresztül érik el az OSPF hálózatot, EXTERNAL ként lesznek megjelölve. Ennek is két fajtája van, az E1 és az E2. Az E1 típusnál az útvonal költsége folyamatosan növekszik, ahogy halad a hálózaton. Míg az E2-nél a költség változatlan marad. Ez az alapértelmezett. (**[4]** Cisco.com, 2005)

E1-et akkor érdemes használni, amikor több kijáratunk is van abba hálózatba, így valós döntést tud hozni, hogy melyik a jobb útvonal.




**21. ábra**

Az R4 forgalomirányító jóvoltából az OSPF folyamatban hirdetésre kerül a 2000-es autonóm rendszerben megtalálható utak. Ezek a megfelelő forgalomirányítók tábláiban meg is jelennek, mint E1 külső útvonalak. Ezek a pedig az ABR-ek, név szerint az R2, R3 és R5 forgalomirányítók.

```
R5#sh ip route

Gateway of last resort is 172.16.0.4 to network 0.0.0.0

     200.0.1.0/32 is subnetted, 1 subnets
O E1    200.0.1.1 [110/445] via 172.16.0.4, 00:02:14, FastEthernet0/0
     200.0.2.0/32 is subnetted, 1 subnets
O E1    200.0.2.1 [110/445] via 172.16.0.4, 00:02:14, FastEthernet0/0
     172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
O       172.16.1.1/32 [110/2] via 172.16.0.4, 00:02:51, FastEthernet0/0
C       172.16.0.0/24 is directly connected, FastEthernet0/0
O       172.16.0.0/16 is a summary, 00:02:51, Null0
O       172.16.3.1/32 [110/2] via 172.16.0.3, 00:02:51, FastEthernet0/0
C       172.16.2.1/32 is directly connected, Loopback2
     10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks
O       10.0.2.1/32 [110/2] via 172.16.0.3, 00:02:31, FastEthernet0/0
O       10.0.1.1/32 [110/2] via 172.16.0.3, 00:02:31, FastEthernet0/0
O       10.0.0.0/30 [110/65] via 172.16.0.3, 00:02:32, FastEthernet0/0
O       10.0.0.0/8 is a summary, 00:02:32, Null0
O       10.2.2.1/32 [110/66] via 172.16.0.3, 00:02:32, FastEthernet0/0
O       10.1.1.1/32 [110/66] via 172.16.0.3, 00:02:32, FastEthernet0/0
     192.168.17.0/32 is subnetted, 1 subnets
O       192.168.17.1 [110/65] via 192.168.16.2, 00:07:05, Serial1/0
     192.168.16.0/30 is subnetted, 1 subnets
C       192.168.16.0 is directly connected, Serial1/0
     192.168.19.0/32 is subnetted, 1 subnets
C       192.168.19.1 is directly connected, Loopback19
     192.168.18.0/32 is subnetted, 1 subnets
O       192.168.18.1 [110/65] via 192.168.16.2, 00:07:05, Serial1/0
O*E2 0.0.0.0/0 [110/1] via 172.16.0.4, 00:02:32, FastEthernet0/0
O IA 192.168.0.0/20 [110/66] via 172.16.0.3, 00:02:32, FastEthernet0/0
O    192.168.16.0/20 is a summary, 00:07:05, Null0
```

**22. ábra**

Ezek az útvonalak azért lettek E1 jelölésűek, mert a ***redistribute bgp 6000 metric 444 metric-type 1 subnets route-map BGP_TO_OSPF*** parancsnál, specifikáltam, hogy a ***metric-type 1*** legyen, ami az E1 jelű útvonalakat eredményezte. Láthatunk még E2 jelölésű

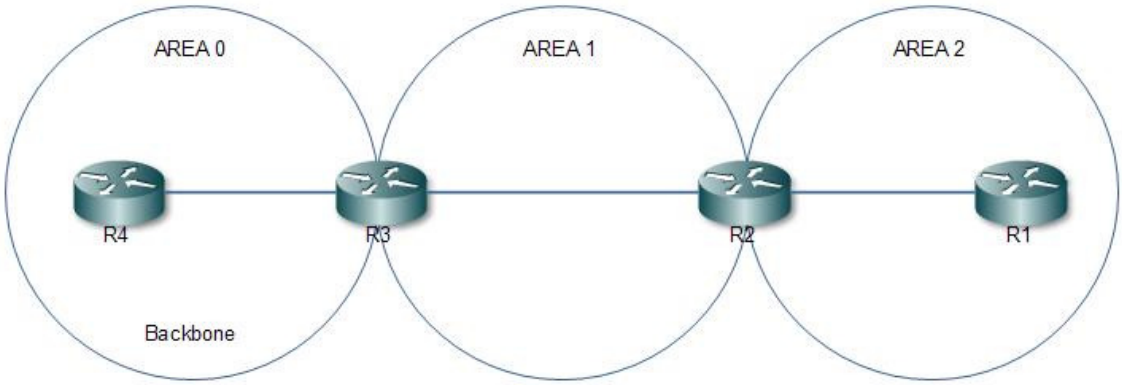
alapértelmezett útvonalat. Ez az OSPF kijáratát jelenti, ami nekünk az R4 –et jelenti. Mivel ez a kijáratunk a „külvilágba”, ezért a ***default-information originate always*** paranccsal a többi OSPF szomszédnak hirdetünk egy alapértelmezett útvonalat. Az ***always*** kulcsszó azt eredményezi, hogy akkor is fogja hirdetni a kijáratot, ha az ő forgalomirányító táblájában nincs.

```
R4#sh run | b router ospf
router ospf 1
 router-id 4.4.4.4
 log-adjacency-changes
 summary-address 172.16.0.0 255.255.0.0
 redistribute bgp 6000 metric 444 metric-type 1 subnets route-map BGP_TO_OSPF
 network 172.16.0.0 0.0.0.255 area 2
 network 172.16.1.1 0.0.0.0 area 2
 default-information originate always
```

**23. ábra**

### 3.1.8. Virtuális kapcsolat, a szabályok megszegése

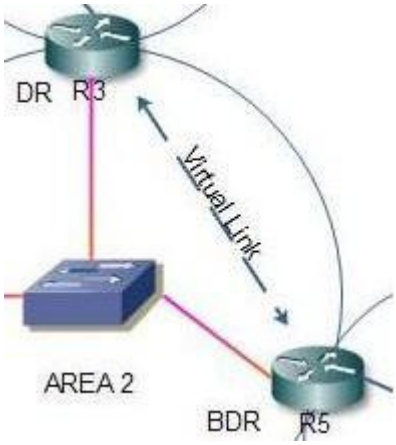
Akkor használjuk, ha be szeretnénk kapcsolni egy új területet az OSPF folyamatunkba, de nincs arra lehetőség, hogy a nullás, gerinc területhez kapcsoljuk. Ekkor jön szóba a virtuális kapcsolat, mint ideiglenes megoldás, amíg nem tudjuk megoldani a helyes csatlakozását a területnek, ahogy azt a szabvány megkövetelné. (**[7]** 15. fejezet)




**24. ábra**

A 24. ábrán, a 2. terület, mint látható az 1. területhez kapcsolódik, nem pedig a nullás, gerinc területhez. Ha csak így hagynánk, akkor nem működne, az új területet nem tudnánk elérni. Ezért van szükség a virtuális kapcsolatra (Virtual Link). Ezzel tulajdonképpen egy alagutat (tunnel) hozunk létre az R3 és R2 forgalomirányítók között. Fontos megjegyezni, hogy ha nem közvetlenül ennének összekapcsolva, mint az ábrán is látszik, hanem közöttük

több forgalomirányító is elhelyezkedne, akkor is csak a határ forgalomirányítókon kellene beállítanunk ezt az alagutat. Amint ezzel elkészültünk, azon az alagúton a 2. terület virtuális már közvetlenül kapcsolódik a gerinc területhez.




**25. ábra**

A példánkban a virtuális linket az „útban lévő" terület határán kell konfigurálnunk. Jelen esetben ez a kettes terület (Area 2), és az érintett forgalomirányítók az R3 és az R5. A parancs szintaktikája, hogy meg kell adnunk mely terület fölött szeretnénk létrehozni a virtuális kapcsolatot, majd pedig a távoli forgalomirányító azonosítója következik. A jobb átláthatóság érdekében használtam itt a ***router-id*** parancsot. Ez az a sor ***router-id 3.3.3.3***, amivel a forgalomirányító „nevét" állíthatjuk be az OSPF folyamatban. Ahol nincs ilyen ***router-id*** megadva, ott a legmagasabb aktív interfész címe lesz az azonosítója. Amennyiben loopback interfészek is találhatóak akkor abból a legmagasabb IP számú. Tehát az R5-ön kiadott parancs az ***area 2 virtual-link 3.3.3.3***, és az R3-on kiadott pedig az ***area 2 virtual-link 5.5.5.5***.

```
R5#sh run | b router ospf
router ospf 1
 router-id 5.5.5.5
 log-adjacency-changes
 area 0 range 10.0.0.0 255.0.0.0
 area 2 range 172.16.0.0 255.255.0.0
 area 2 virtual-link 3.3.3.3
 area 3 stub
 area 3 range 192.168.16.0 255.255.240.0
 network 172.16.0.0 0.0.0.255 area 2
 network 172.16.2.1 0.0.0.0 area 2
 network 192.168.16.0 0.0.0.3 area 3
 network 192.168.19.1 0.0.0.0 area 3
```

```
R3#sh run | b router ospf
router ospf 1
 router-id 3.3.3.3
 log-adjacency-changes
 area 0 range 10.0.0.0 255.0.0.0
 area 2 range 172.16.0.0 255.255.0.0
 area 2 virtual-link 5.5.5.5
 network 10.0.0.0 0.0.0.3 area 0
 network 10.0.1.1 0.0.0.0 area 0
 network 10.0.2.1 0.0.0.0 area 0
 network 172.16.0.0 0.0.0.255 area 2
 network 172.16.3.1 0.0.0.0 area 2
```

**26. ábra**

Több paranccsal is meggyőződhetünk arról, hogy a virtuális kapcsolatunk létrejött. Az egyik a ***show ip ospf neighbor***.

```
R3#sh ip ospf neigh

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.4.1       0   FULL/  -        00:00:37    10.0.0.1        Serial1/1
5.5.5.5           0   FULL/  -           -        172.16.0.5      OSPF_VL0
4.4.4.4           0   FULL/DROTHER    00:00:33    172.16.0.4      FastEthernet0/0
5.5.5.5           5   FULL/BDR        00:00:31    172.16.0.5      FastEthernet0/0
R5#sh ip ospf neigh

Neighbor ID     Pri   State           Dead Time   Address         Interface
3.3.3.3           0   FULL/  -           -        172.16.0.3      OSPF_VL0
3.3.3.3          10   FULL/DR         00:00:39    172.16.0.3      FastEthernet0/0
4.4.4.4           0   FULL/DROTHER    00:00:37    172.16.0.4      FastEthernet0/0
192.168.18.1      0   FULL/  -        00:00:38    192.168.16.2    Serial1/0
```

**27. ábra**

Itt láthatjuk, hogy valóban létrejött a kapcsolat. Az R5-nél az OSPF_VL0 sornál láthatjuk, hogy a szomszédunk a 3.3.3.3, ami nem más, mint az R3. Az R3-nál pedig az 5.5.5.5 van, ami az R5. A másik parancs a ***show ip ospf virtual-link***.

```
R3#sh ip ospf vir
Virtual Link OSPF_VL0 to router 5.5.5.5 is up
  Run as demand circuit
  DoNotAge LSA allowed.
  Transit area 2, via interface FastEthernet0/0, Cost of using 1
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:06
    Adjacency State FULL (Hello suppressed)
    Index 1/3, retransmission queue length 0, number of retransmission 1
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 1, maximum is 1
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

**28. ábra**

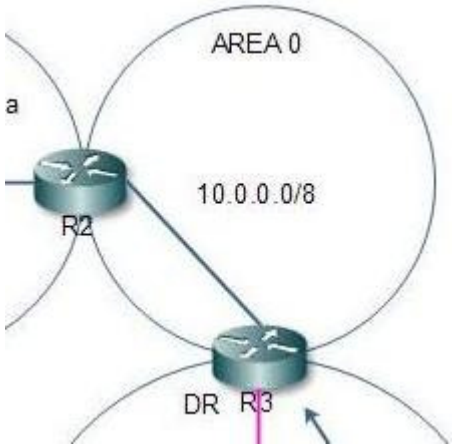
### 3.1.9. OSPF hitelesítés

Ha egy támadó képes létrehozni OSPF csomagokat, és ezeket a hálózatba tudja küldeni, akkor képes lehet eltéríteni a csomagokat az eredeti újukról, ahol ő könnyebben elkaphatja őket, vagy akár meg is béníthatja a hálózatot. Az OSPF folyamatban résztvevők alapértelmezetten megbíznak egymásban, ezért vezették be a hitelesítést. Képesek vagyunk hitelesíteni a forgalomirányítási frissítéseket, és minden más OSPF csomagot. Ezt pedig egy közös titok megosztásával teszik meg. Három fajtája létezik a hitelesítésnek:

- Null, ebben az esetben nem beszélhetünk hitelesítésről. Ez az alapértelmezett.

- Sima szöveghitelesítés. Jelen esetben a biztonság csekély, hiszen akárki, aki hozzáfér a médiumhoz, elkaphatja a csomagokat, majd könnyedén kiolvashatja belőle a hitelesítéshez használt jelszót
- MD5. Ez már viszonylag magas biztonságot ad, mert aki ezzel a hitelesítéssel küldi el a csomagot, tisztában van a közös jelszóval. Itt a jelszó sohasem kerül ki a médiumra közvetlenül, hanem annak egy leképezése (hash). Ezt a cél visszafejtve, ha ugyanazt a jelszót kapja, mint amivel ő is rendelkezik, akkor a hitelesítés sikerrel járt, a kommunikáció működik.

Fontos megjegyezni, hogy a hitelesítésnek területenként megegyezőnek kellett lenniük a régebbi IOS esetében (12.0 előtt). A frissebb verziókban a hitelesítést interfészekre lehet megadni, ami sokkal flexibilisebb, mint a régi rendszer. (**[4]** Cisco.com, 2005)




**29. ábra**

```
R3#sh run int s 1/1
interface Serial1/1
 description *** TO R2 ***
 ip address 10.0.0.2 255.255.255.252
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 7 13061E010803
 serial restart-delay 0
end
```

```
R2#sh run int s 1/1
interface Serial1/1
 description *** TO R3 ***
 ip address 10.0.0.1 255.255.255.252
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 cisco
 serial restart-delay 0
 clock rate 64000
end
```

**30. ábra**

Magát a hitelesítést két lépcsőben tudjuk bekapcsolni. Először meg kell adnunk, hogy milyen fajta hitelesítést szeretnénk használni (md5 vagy sima szöveg). Páronként meg kell egyezniük a paramétereknek, hogy a kapcsolat létrejöhessen. Az ***ip ospf authentication message-digest*** parancs jelen esetben az md5 hitelesítést kapcsolja be. Az ***ip ospf message-digest-key 1 md5 <jelszó>*** parancs pedig maga a jelszó. A két forgalomirányítónál látszólag

nem egyezik meg a jelszó, de ez nem így van. Az R3 esetében kiadtam a ***service password-encryption*** parancsot, ami annyit csinál, hogy a konfigurációs fájlban lévő összes sima szöveges jelszót titkosítja. Ez egyáltalán nem erős titkosítás, csak a szem elől fedi el a jelszavakat, hogy illetéktelenek ne tudják elolvasni. A jelszó mind a két esetben ***cisco***.

## 3.2. BGP (Border Gateway Protocol)

### 3.2.1. A BGP helye a mai hálózatos világban

Ezt a protokollt arra használják, hogy az Interneten a különböző szolgáltatókat összekapcsolják. A különböző szolgáltatók, különböző autonóm rendszereket (AS) jelentenek. Ezt a feladatot csak a BGP képes ellátni, ez az egyetlen protokoll, ami megbirkózik vele. A feladata egyszerű, megtalálni a legjobb utat az Interneten. Tervezéséből fakadóan, mivel nem belső irányítóprotokollnak készült, nagyon lassú. A leglassabb a földön, de ez jelen esetben nem róható fel hátrányként. Ha minden egyes változásra azonnal reagálna, összeomlana a változások alatt. Egy olyan hatalmas kiterjedésű hálózatban, mint maga az Internet, folyamatosan történnek változások. Egyik útvonal valamilyen okból kifolyólag használhatatlanná válik, míg más, eddig nem jól működőek kezdenek el funkcionalitásuknak megfelelően működni. Ha minden változást azonnal lereagálna, kész káosz lenne. Mivel ilyen nagy hálózatoknál használják, ezért a forgalomirányítási táblája sem olyan méretű, mint amit eddig megszokhattunk. Nem ritkán előfordulnak több száz megabájtos forgalomirányítási táblák is akár 100 000 bejegyzéssel. A BGP-ről részletesen az RFC 1771 dokumentumban olvashatunk.

### 3.2.2. Tények a BGP-ről

A belső irányító protokollokkal ellentétben, akik mindig a legújabb, legfrissebb utakat akarják a forgalomirányító táblájukban tudni, a BGP a stabil, nem állandóan újrahirdetett utakat részesíti előnyben. Miért? A BGP a hálózatot „csendben” szeretné tartani. Minél kevesebb a frissítés, annál jobb. Fontos még megemlíteni, hogy házirend alapú (policy based), a konfigurációnál szinte mindent meg lehet neki adni, ami a működését még jobbá teszi. Attribútumokon keresztül finom hangolható, úgy, mint semelyik más forgalomirányító protokoll. Az Interneten a megbízhatóság ugyanúgy kulcsfontosságú, mint az elérhetőség. A BGP ezért TCP-t használ, ami megbízhatóvá teszi. Ez nem magára az adatra vonatkozik, hanem csak a BGP frissítésekre. Összehasonlításképpen ez a megbízhatósági mechanizmus az OSPF protokollba van beépítve. Amikor két BGP-t futtató forgalomirányító kapcsolatot létesít a 179-es TCP porton, akkor az egy megbízható kapcsolatnak minősül a TCP következő szolgáltatásai miatt:

- Nyugtázás

  Miután elküldtünk egy szegmenst, egy számláló elkezd visszafelé számolni, amíg egy nyugtát nem kap a másik forgalomirányítótól. Ha a számláló eléri a nullát, a szegmens újra kiküldésre kerül.

- Szegmentálás

  A BGP adatok szegmentálásra kerülnek kisebb darabokra, amennyiben erre szükség van, hogy célba jussanak a hálózaton.

- Ellenőrző összeg

  Az ellenőrző összeg mind a TCP fejrészre, mind a BGP adat mezőjére is kiterjed, hogy hibamentes átvitelt biztosítson. Amennyiben az ellenőrzőösszeg különbözik, a szegmens eldobásra kerül. Nem küld nyugtát a feladónak, így az újraküldi a szegmenst.

- Adat sorrendbe rendezése

  A TCP képes arra, hogy a nem sorrendben érkező csomagokat újra sorrendbe tegye, amennyiben valamilyen okból kifolyólag nem sorrendben kapta meg.

- Áramlásvezérlés

  Minden BGP-t futtató forgalomirányító tudatja a másikkal, hogy mennyi elérhető puffere van még, így az csak megfelelő mennyiségű adatot küldjön.

([**1**] 399. oldal)

### 3.2.3. A BGP működésének alapjai

A BGP két forgalomirányító között hirdeti az utakat. A két forgalomirányító közötti kapcsolatot az határozza meg, hogy az autonómrendszer, amiben elhelyezkednek, megegyezik (iBGP), vagy különbözik (eBGP). Magát a viszony kialakítását és az aktuális útvonalfrissítéseket a különböző BGP csomagtípusok határozzák meg és irányítják. A helyi útválasztó a döntését a kapott információk alapján hozza meg, majd ezeket továbbítja a többi BGP forgalomirányító felé a hálózatban. A döntést befolyásolják a különböző attribútumok valamint a helyi házirend (policy) is.

Technikailag egy távolságvektor alapú protokoll a BGP, de útvonal-vektornak is ismert, ami talán jobban illik rá. Amikor továbbhirdet egy hálózatot, akkor mindig beleteszi a hirdetménybe, hogy milyen AS-ből indult. Ahogy halad tovább a hálózaton, egyre több AS szám lesz felfűzve egymás után, aszerint, hogy milyen autonóm rendszereken haladt már keresztül. Ha olyan útvonalat kap, aminek az AS útvonalában már szerepel a saját autonóm rendszer száma, akkor azt elutasítja. Így akadályozza meg, hogy hurok alakuljon ki. Ha nem állítanánk be a sok attribútumot a helyes működéshez, akkor gyakorlatilag ugyanúgy működne, mint a RIP, csak az ugrásszámot nézné, hogy milyen messze található a célhálózat. Ez így hamis képet adna, mert a különböző kapcsolatoknak más a sávszélessége és ezeket kihagyni a döntésből butaság lenne.

Olyan, mint metrika, nem olyan értelemben ismert, mint például az OSPF esetében. Helyette a BGP egy 10 lépcsőből álló tulajdonságokat vesz figyelembe, amik alapján kiválasztja az utat. A későbbiekben erről még lesz szó.

Belső forgalomirányító protokollok olyan környezetben működnek, ahol egy szigorúan hierarchikus dizájn szerint vannak kiosztva az IP címek, ebből kifolyólag a summázás lehetősége adott, hogy minél kisebb forgalomirányítási táblával dolgozzanak az eszközök. A BGP-nek ez nem adatott meg, neki olyan környezetben kell boldogulni, ahol az IP címek kiosztása nem a hierarchikus dizájnt követte, és a topológia sem feltétlenül ideális, mint a belső társainál. Természetesen a BGP-nél is van lehetőség summázásra, ugyanabból az okból, mint a többi protokollnál. Minél kisebb egy forgalomirányítási tábla annál gyorsabb lesz a folyamat, mert kevesebb memóriát és processzor időt igényel a feldolgozása. Egy optimalizált BGP hálózat erősen summázott, ugyanakkor nem hierarchikus felépítésű. (**[2]** 316. oldal)

### BGP üzenettípusok

Négy üzenettípust különböztetünk meg, amit a kapcsolat kiépítéséhez és fenntartásához használunk.

- Nyitás (Open)

  Ez kerül először kiküldésre. Itt egyeznek meg a kapcsolatot felépítő attribútumokról. Ilyenek például a BGP verziószám (4-es verziót használunk most), az időzítők vagy a hitelesítő adat.

- Életben tartás (Keepalive)

Ha az Életben tartó periódus alatt egy Frissítés kiküldésre kerül, a számlálót az is ki fogja nullázni. Tehát akár Frissítés, akár Életben tartó üzenetekkel is fent lehet tartani a kapcsolatot. Amennyiben egy sem érkezik a megadott időn belül, abban az esetben egy Értesítő üzenet kerül kiküldésre, és a kapcsolat megszakításra kerül.

- Frissítés (Update)

  A forgalomirányítási információk kiküldése és visszavonása is az Frissítés üzenettel történik.

- Értesítés (Notification)

  Amint egy BGP szomszéd hibát észlel a kapcsolatban, kiküld egy Értesítést a távoli forgalomirányítónak, és azonnal lezárja a TCP valamint a BGP kapcsolatot is.

Minden a Nyitás üzenettel kezdődik. Amint a BGP folyamat elindul, a 179-es TCP portot használja, hogy létrehozza és karbantartsa a kapcsolatot a Nyitás üzenettel. Ezentúl, bizonyos időközönként Életben tartó üzenetekkel ellenőrzi, hogy él-e még a kapcsolat. Ha egy szomszéd újraindul, kiküld egy értesítő üzenetet, hogy zárják le a kapcsolatot. Az értesítés soha nem jelent jót, mindig a kapcsolat megszűnését kezdeményezi. Amint először létrejött a kapcsolat, a szomszédok kicserélik a teljes forgalomirányítási táblájukat, erre a Frissítés üzenetet használják. Később csak akkor küldenek Frissítést, ha valami változik. (**[1]** 400. oldal)

### BGP táblatípusok

Három táblát kezel, de nem ugyanazokat, mint az OSPF.

- Szomszédsági tábla

  Ebbe a táblába az előre beállított szomszédok kerülnek, akikkel sikerült kiépíteni a szomszédsági viszonyt. A BGP-nél nem történik automatikusan meg a szomszédok felfedezése, ezért nekünk kell beállítanunk őket.

- BGP tábla

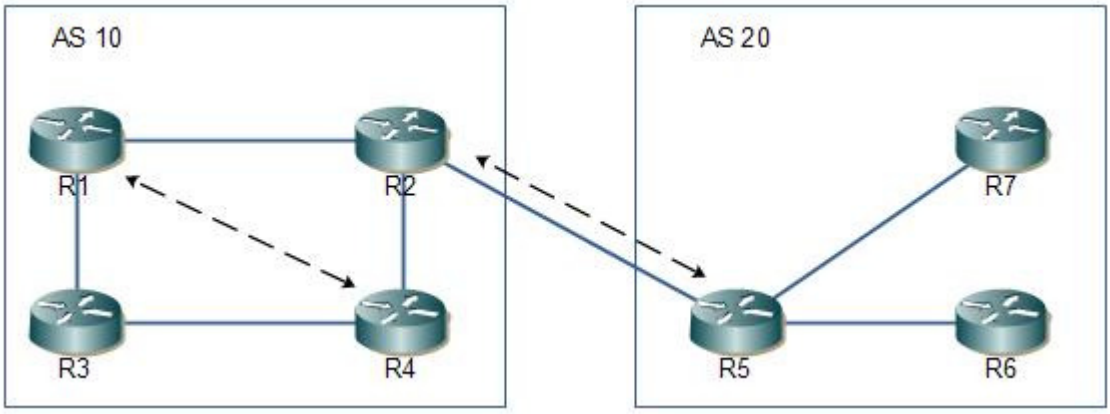
  Ez a tábla tartalmazza az összes BGP útvonalat. Hatalmas is tud lenni, de csak a legjobbak kerülnek belőle a forgalomirányítási táblába.

- Forgalomirányítási tábla

A legjobb utak a BGP táblából kerülnek bele ebbe a táblába.

### 3.2.4. A BGP szomszédsági viszony

A BGP információkat cserél két forgalomirányító között, amiket szomszédoknak nevezünk. Ez a kapcsolat pusztán logikai, nem kötelező, hogy valóban közvetlenül kapcsolódjanak. A TCP kapcsolatot használja arra, hogy olyan esetben is létrejöjjön a kapcsolat, ha közvetlenül vannak összekötve, vagy számos közbülső kapcsolat által.




**31. ábra**

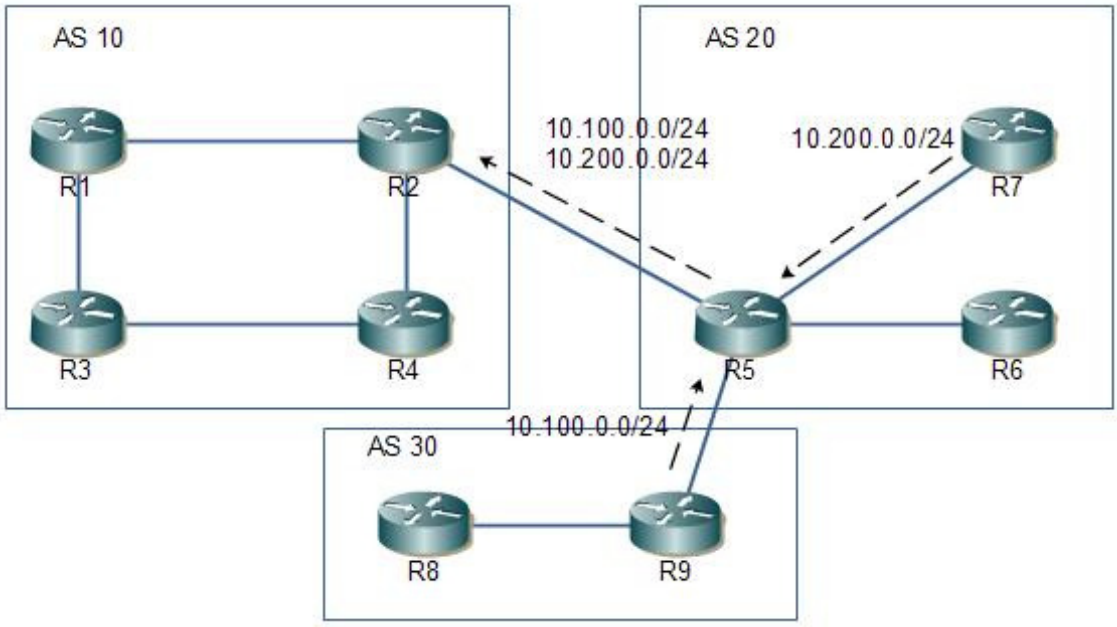
A 31. ábrán látható, hogy nem szükséges minden esetben a közvetlen kapcsolat a BGP szomszédsági viszony felállításához, mint az R2 és R5 forgalomirányítók között. Az R1 és R4 ugyanúgy lehetnek BGP szomszédok. Az R2 és R5 közötti közvetlen kapcsolatból eredően az IP kapcsolat létrejön, ami után a TCP is ki tud alakulni. Az R1 és az R4 esetében más a helyzet. Egy másik forgalomirányítón keresztül érik el egymást (R2 vagy R3), így itt az IP kapcsolat létrejöttét egy belső protokollnak kell szolgáltatnia vagy esetleg statikus út beállítása is megoldás lehet.

Az ábrán megfigyelhetünk még egy eltérést a két-két BGP szomszéd között. Az R2 és R5 között fennálló viszony, különböző autonóm rendszerek között történt. Ezt külső BGP-nek nevezzük (Exterior BGP, eBGP). Az R1 és az R4, ahhoz, hogy kiépítsék a szomszédsági viszonyt nem kell elhagyniuk az autonóm rendszert. Ezt belső BGP-nek (Interior BGP, iBGP) nevezzük.

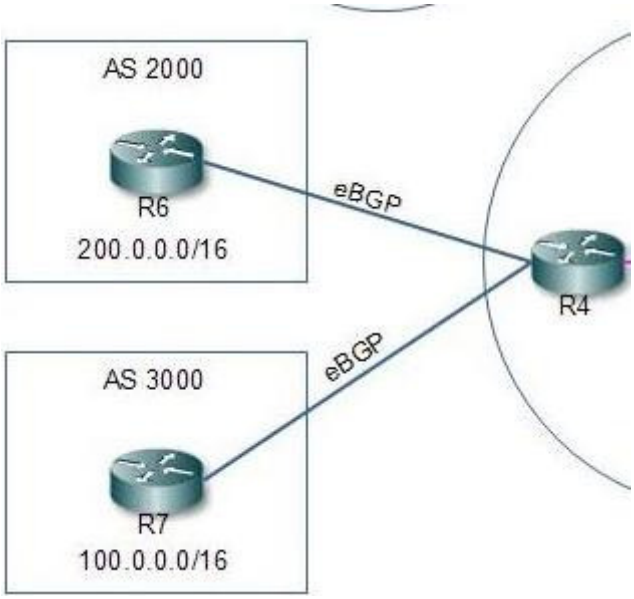
### 3.2.5. Külső BGP

Amikor két forgalomirányító BGP szomszédsági kapcsolatot hoz létre, úgy hogy különböző autonóm rendszerben helyezkednek el, az a külső BGP. A közvetlen kapcsolat itt lényeges, mert a csomag TTL mezője 1-re van állítva. Ha ennél több ugrásra érné el a szomszédot, akkor sohasem alakulhatna ki a kapcsolat. Ez persze meg is változtatható egy parancs segítségével. Amint az eBGP kapcsolat létrejött, a két szomszéd elkezdi kicserélni egymással a forgalomirányítási információikat. Minden aktív BGP útvonalat, amit más eBGP kapcsolatokon keresztül tanultak meg, azt hirdetik. Továbbá minden olyan BGP útvonalat, amit iBGP-n keresztül tanultak meg, szintén továbbadnak. (**[5]** Cisco.com, 2008)

Az ábrán láthatóak az alapértelmezett eBGP útvonal hirdetmények. Az R5 a 10.100.0.0/24-es hálózatot az R9-től kapta meg eBGP-n keresztül, a 10.200.0.0/24-et pedig az R7-től iBGP-n keresztül. Mind a két útvonal aktív jelen pillanatban az R5 forgalomirányítási táblájában, és tovább is adja ezeket az információkat az R2-nek az eBGP kapcsolaton keresztül.




**32. ábra**




**33. ábra**

A topológia szerint BGP-t futattunk az R4, R6 és R7-es forgalomirányítókon. A feladat az, hogy az OSPF folyamatban lévő forgalomirányítók elérjék a 2000-es és 3000-es autonóm rendszert, az R4-en keresztül. Erre eBGP-t használunk az R4 és R6, valamint az R4 és R7 között. Az R6 és R7 forgalomirányítók ismerik az OSPF területeken belül használt címeket. Ennek az oka, hogy az összes OSPF utat, amiről az R4-nek tudomása van, hirdeti tovább a BGP folyamatban a két szomszédjának. Az R4 a 6000-es autonóm rendszert használja a BGP-hez. Az OSPF folyamatba viszont nem fogom hirdetni mind az R6-tól és R7-től megkapott utakat, csak az R6-tól megtanultakat. A 3000-es autonóm rendszerbeli IP címeket is el fogják majd érni az OSPF területeken belül lévő forgalomirányítók, mert az R4-ig el tudnak jutni az alapértelmezett útvonallal, az R4-nek pedig tudomása van az R7 által hirdetett hálózatokról.

```
R4#sh run | b router bgp
router bgp 6000
 no synchronization
 bgp log-neighbor-changes
 redistribute ospf 1
 neighbor AS2000 peer-group
 neighbor AS2000 remote-as 2000
 neighbor AS2000 route-map OSPF_ONLY out
 neighbor 100.0.0.1 remote-as 3000
 neighbor 100.0.0.1 password 5 cisco
 neighbor 100.0.0.1 route-map OSPF_ONLY out
 neighbor 200.0.0.1 peer-group AS2000
 no auto-summary
```

**34. ábra**

Az R4 forgalomirányítón a BGP beállításának két féle módját is láthatjuk. Az egyik a 3000-es autonóm rendszerhez kapcsolja, a másik pedig a 2000-eshez. Működésben nincs különbség, csak például iBGP esetében a ***peer-group*** megkönnyíti a helyzetünket. Amikor is a paraméterek megegyeznek minden szomszédnál, csak az IP címek változnak, egyszerűbb ezt egyszer leírnunk, majd alkalmazni minden szomszédra. BGP szomszédok konfigurációjánál minden esetben a parancs a ***neighbor <IP cím>*** -el kezdődik. Ezután kell megadnunk, hogy melyik autonóm rendszerbe tartozik az a szomszéd, ***neighbor 100.0.0.1 remote-as 3000***, vagy például milyen autentikációt használunk, ***neighbor 100.0.0.1 password 5 cisco***. Természetesen a jelszavaknak itt is páronként meg kell egyeznie. Ha csak a jelszó egyezik meg, de a szint, az előtte lévő szám, nem akkor sem fog működni. Ez az egyik módja a szomszéd beállításának. A másik módja a ***peer-group*** parancs használat. Jelen esetben első lépésként létrehozunk egy ***AS2000*** nevű ***peer-group***-ot, ***neighbor AS2000 peer-group***. Majd ehhez a névhez beállítjuk az autonóm rendszert, a jelszót, és az egyéb beállításokat, csak úgy, mint ha egy valódi, IP címes szomszédhoz tennénk azt. Ha mindennel elkészültünk, akkor már csak a megfelelő IP címre kell alkalmaznunk a létrehozott ***peer-group***-ot, ***neighbor 200.0.0.1 peer-group AS2000***. Ezzel elkészültünk az egyik oldalon a BGP konfigurációjával.

```
R6#sh run | b router bgp
router bgp 2000
 no synchronization
 bgp log-neighbor-changes
 redistribute connected
 neighbor 200.0.0.2 remote-as 6000
 no auto-summary
```

```
R7#sh run | b router bgp
router bgp 3000
 no synchronization
 bgp log-neighbor-changes
 redistribute connected
 neighbor 100.0.0.2 remote-as 6000
 neighbor 100.0.0.2 password 5 cisco
 no auto-summary
```

**35. ábra**

A másik oldalon is ugyanezeket kell beállítanunk, csak természetes az IP címek változnak, mert a szomszéd ebből a szemszögből a másik forgalomirányító. Ebben a példában a BGP szomszédok közvetlenül kapcsolódnak, de lehetőség van loopback interfészeket használni, mint szomszéd IP címet. Ilyenkor viszont további beállításokra van szükségünk. Az eBGP a szomszédot egy „ugrásra” feltételezi. Ez viszont csak közvetlenül kapcsolt szomszédok esetében működik helyesen, loopbacknél nem. Erre van a ***neighbor <IP cím> ebgp-multihop <szám>*** parancs, amivel további „ugrásszámot” lehet beállítani. Ezzel áthidaljuk azt a problémát, hogy a szomszéd csak egy „ugrásra” lehet, de további problémák is felmerülnek. Ha loopback interfészek címeit használjuk szomszédnak, attól még a csomag feladó címe az elhagyott interfész IP címe lesz, nem pedig a loopback interfész IP címe. Tehát soha nem fog

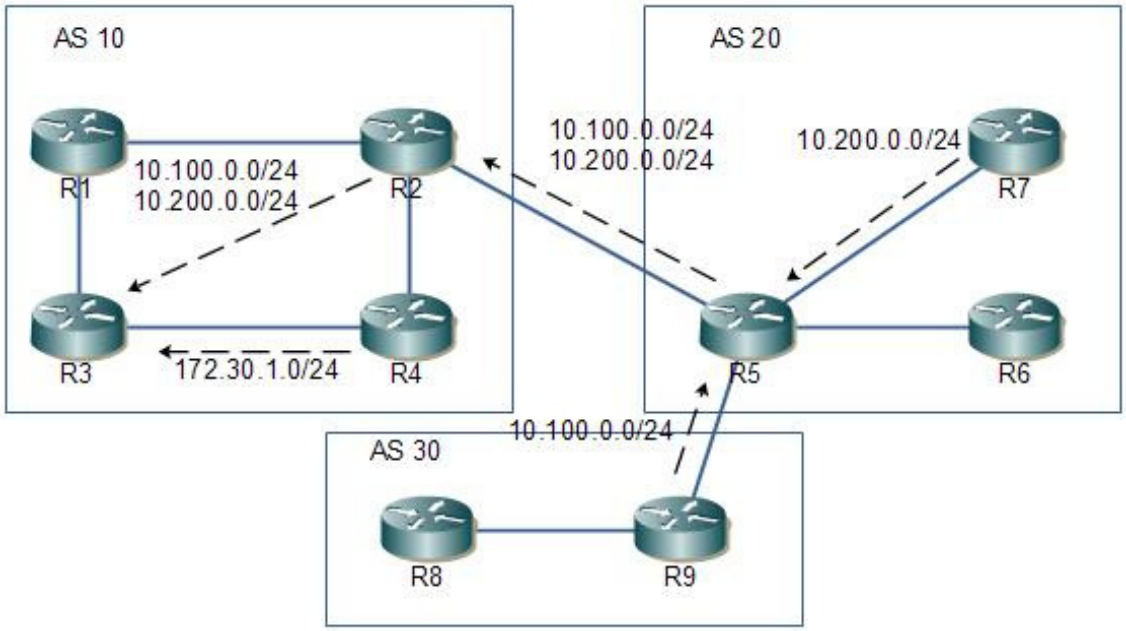
összeállni a szomszédsági viszony. Erre is van egy parancs, ***neighbor <IP cím> update-source <interfész neve>***. Így a kilépő csomagok forrás címében már a kívánt interfész IP címe lesz, és létrejöhet a kapcsolat.

Ezekkel az alapbeállításokkal az R4 tudni fog R6 és R7 által hirdetett utakról, viszont ha így próbálnánk meg elérni például az R2 forgalomirányítóról az R6 egyik loopback interfészét, akkor sikertelen lenne a próbálkozásunk. Az oda utunk rendben van, hiszen az R2 tudja, a forgalomirányítási táblájából, hogy a külvilágot az R4-en keresztül éri el, ezért el is küldi neki, az R4 pedig továbbítja azt az R6-nak. A hiba itt jelentkezik, ugyanis az R6-nak nincs információja arról az IP címről ahonnan a kérés érkezett. Ugyanez a helyzet az R7-es forgalomirányítóval is. Erre az a megoldás, hogy az R4-en hirdetjük az OSPF-es utakat a BGP szomszédainknak. Ezt a ***redistribute ospf 1*** paranccsal tehetjük meg, ahol az egyes szám az OSPF folyamatazonosító az R4 forgalomirányítón. Így már sikeresen el tudjuk érni a R6 valamint az R7 hálózatait is, hiszen a kérések immár vissza is találnak a feladóhoz.

### 3.2.6. Belső BGP

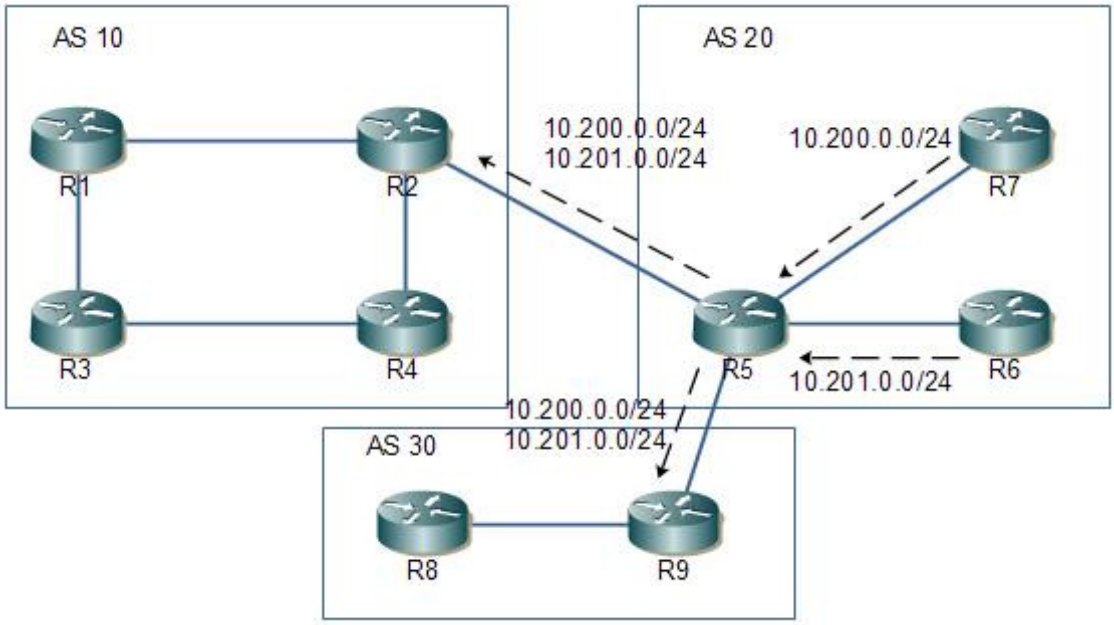
A kapcsolatot két forgalomirányító között, amikor azok ugyanabban az autonóm rendszerben helyezkednek el, belső BGP-nek (iBGP) nevezzük. Az eBGP-vel ellentétben itt nincs szükség közvetlen kapcsolatra, mert a csomag TTL mezője jelen esetben 64, ami megengedi, hogy a csomag keresztülutazzon az AS-en belül, hogy elérje az iBGP szomszédot.

Az iBGP szomszédoknak egy belső irányító protokollra kell támaszkodniuk az autonóm rendszeren belül. Általánosságban, a TCP kapcsolat az iBGP szomszédokkal úgy jön létre a hálózaton, hogy a közbülső forgalomirányítók forgalomirányító tábláját használja, hogy létrejöjjön maga a kapcsolat. Egészen konkrétan a loopback interfészeken keresztül alakul ki a szomszédsági viszony a hibatűrés és stabilitás miatt. Így az iBGP kapcsolat akkor is fent áll, ha az egyik kapcsolatban hiba keletkezik. Amint az iBGP kapcsolat létrejött, az utak kicserélésre kerülnek. Alapértelmezetten, csak az aktív BGP utakat tanulja meg az eBGP-t futtató szomszédjától.




**36. ábra**

Az alapértelmezett iBGP hirdetési szabályokat a 36. ábrán láthatjuk. Az R2 és R3 iBGP szomszédok. Az R2 megtanulja a 10.200.0.0/24 és az 10.100.0.0/24 hálózatot is az R5 forgalomirányítótól eBGP kapcsolat révén. Ezeket utána rögtön továbbadja az R3-nak. Az R3 ezen kívül megtanulja még a 172.30.1.0/24-es hálózatot is az iBGP szomszédjától, R4-től. Ezt az utat nem hirdeti tovább a 10-es AS-ben, mert ezzel megszegné az iBGP szomszédok szabályát. Addig nem fogja hirdetni és használni az iBGP-n keresztül tanult utat, amíg a belső forgalomirányító protokolltól meg nem tanulta ugyanazt. Erre azért van szükség, mert ha anélkül továbbítaná, hogy a többi forgalomirányító, akik csak a belső irányító protokollt futtatják és BGP-t nem, nem tudnának erről az útvonalról akkor úgyis csak eldobnák azt. Erről a 37-es ábrán láthatunk egy problémát.




**37. ábra**

Mind az R6 és R7 forgalomirányítónak is van iBGP kapcsolata az R5-el, de egymással már nincs. A 10.100.0.0/24 hálózatot az R7, és a 10.201.0.0/24 hálózatot pedig az R6 hirdeti az R5 felé. Az R5 mind a két kapott útvonalat tovább is hirdeti az eBGP szomszédai felé, R2-nek és R9-nek is. Alapértelmezésben ezeket nem hirdeti az iBGP szomszédainak. Ennek eredménye képen az R7-es és R6-os forgalomirányítóknak fogalmuk sem lesz egymás létezéséről. A megoldást az adja, hogy létre kell hoznunk egy iBGP kapcsolatot az R7 és R6 között. Ez lehetséges, hiszen iBGP szomszédoknál nem feltétel, hogy közvetlenül össze legyenek kapcsolva. Ha ezzel megvagyunk, akkor az R7-es forgalomirányító hirdetni fogja az útjait az R6-nak is és persze változatlanul R5 felé is. Ugyanez elmondható az R6-ról is. Ezt nevezzük teljes összeköttetésnek, amikor is az egy autonóm rendszerben lévő forgalomirányítók, minden egyes másik útválasztóval iBGP kapcsolatot létesítenek.

### 3.2.7. BGP szomszédok kialakulása

- Tétlen (Idle)

  Ebben az állapotban elutasít minden beérkező szomszédságot kialakító, vagy fenntartó üzenetet visszautasít. Miután a BGP folyamat elindult, és a TCP kapcsolat kialakítása is elkezdődött, a forgalomirányító Kapcsolt állapotba kerül.

- Kapcsolt (Connect)

Ebben az állapotban a helyi forgalomirányító vár, amíg a TCP kapcsolat be nem fejeződik. Amennyiben ez sikerrel lezárult, a helyi forgalomirányító egy Nyitás üzenetet küld a másik félnek, és a Nyitás elküldve állapotba kerül.

Ha a TCP kapcsolat sikertelen, a helyi forgalomirányító újraindít egy számlálót, és Aktív állapotba kerül.

- Aktív (Active)

  Jelen esetben a forgalomirányító TCP kapcsolatot próbál létesíteni a másik féllel. Ha sikerrel jár, egy Nyitás üzenet kerül kiküldésre, és Nyitás elküldve állapotba kerül.

  Amennyiben a távoli forgalomirányító rossz IP címmel próbál kapcsolatot létesíteni, nem azzal, amivel a várná a szomszéd, akkor azt vissza fogja utasítani, és Aktív állapotban marad.

- Nyitás elküldve (Open Sent)

  Ezt az állapotot akkor láthatjuk, amikor a TCP kapcsolat sikeres volt. A helyi forgalomirányító küld a párjának egy Nyitás üzenetet és várja rá a választ. Amint megérkezik egy ilyen Nyitás üzenet, Életben tartó (Keepalive) üzenetekben megegyeznek a paraméterekről, és Nyitás megerősítve állapotba kerül.

  Ha ebben a fázisban megszakad a TCP kapcsolat, akkor a BGP viszony is megszűnik, és Aktív állapotba kerül ismét a forgalomirányító.

- Nyitás megerősítve (Open Confirm)

  Ha az egyik fogalomirányító helyes Nyitás üzenetet kap, akkor Ebbe az állapotba kerül. Ezután Életben tartó üzeneteket küld, és várja, hogy erre érkezzen is vissza másik Életben tartó üzenet a szomszédtól.

- Létrejött kapcsolat (Established)

  Akkor kerül ebbe az állapotba, ha kapott Életben tartó üzenetet, amíg Nyitás megerősítve állapotban tartózkodott. Ez az utolsó állapot, ilyenkor minden rendben működik. Forgalomirányítási információkat csak Létrejött kapocsalt állapotban tudnak cserélni egymással a forgalomirányítók. Minden más nem helyes, vagy éppen kialakuló működésre utal.

(**[2]** 329. oldal)

### 3.2.8. Útvonalhirdetés BGP-ből OSPF-be

Lehetőségünk van a BGP szomszédoktól megtanult hálózatokat az OSPF folyamatban hirdetni. A parancs hasonló, mint az előbb már említett OSPF utak hirdetése a BGP folyamatba a ***redistribute ospf 1*** paranccsal, csak itt egy két finomhangolást hajtottam végre. Ezt már az OSPF folyamat alatt kell kiadnunk, ***redistribute bgp 6000 metric 444 metric-type 1 subnets route-map BGP_TO_OSPF*** és jóval hosszabb, mint elődje. Ezen attribútumok nagy része opcionális. A ***metric 444***, azt fogja csinálni, hogy az ezáltal az OSPF folyamatba kerülő utak metrikáját 444-re változtatja meg. Ez a szám tetszőleges, egy bizonyos kereten belül. A ***metric-type 1***, azt eredményezi, hogy az így bekerülő külső útvonalak E1 jelölést kapnak, ami azt jelenti, hogy a metrika növekszik, minél messzebb kell elküldenünk a hálózaton. Például az R3-nál már 445 lesz. Amennyiben itt kettest adunk meg, akkor E2 jelölést kapnak, és a metrika nem változik. A ***subnets*** kulcsszóval az alhálózatokat is hirdeti. A ***route-map BGP_TO_OSPF*** pedig megszűri a bekerülő útvonalakat.

```
ip access-list standard ONLY_AS_2000
 permit 200.0.0.0 0.0.255.255

route-map BGP_TO_OSPF permit 10
 match ip address ONLY_AS_2000
```

**38. ábra**

Ehhez szükségünk van egy hozzáférés vezérlési listára (access-list). Jelen esetben ez egy nevesített, egyszerű ACL, ami csak a 200.0.0.0/16-os hálózatot engedélyezi, minden mást tilt. Ezt az ACL-t felhasználva a ***route-map***-ben, ellenőrzi azokat az utakat, amiket a BGP folyamatból az OSPF-be szeretnénk hirdetni, de az R7 által hirdetett 100.0.0.0/16-os hálózatból kikerülő utakat nem fogja átengedni, hiszen ezek nem felelnek meg a feltételeknek. Ennek eredménye, hogy az OSPF folyamatban csak az R6 által hirdetett utak kerülnek be.

Ezt a fajta szűrést alkalmaztam még az R4 forgalomirányítón a szomszédokra is közvetlenül, csak egy másik ***route-map***-et, másik ACL-el.

```
neighbor 100.0.0.1 route-map OSPF_ONLY out

ip access-list standard NO_AS
 deny   100.0.0.0 0.0.255.255
 deny   200.0.0.0 0.0.255.255
 permit any

route-map OSPF_ONLY permit 10
 match ip address NO_AS
```

**39. ábra**

Amennyiben ezt nem alkalmaztam volna, az R4-en keresztül az R6 és az R7 kommunikálni tudna, ami jelen esetben nem megengedett. A hozzáférés vezérlési lista tiltja a 100.0.0.0/16 és 200.0.0.0/16-os útvonalak hirdetését, de minden mást megenged. Ezt majd a ***route-map***-re, amit pedig magára a szomszédra alkalmazva azt eredményezi, hogy az R6 forgalomirányítási táblájában nem lesz 100.0.0.0/16-os hálózatból érkező útvonal és fordítva. Ez abban az esetben lehet jó megoldás, ha két ISP-hez (Internet Service Provider) csatlakozunk, akkor akár igen nagy forgalomtól is megkímélhetjük a forgalomirányítónkat, azáltal hogy nem hirdetjük visszafele, hogy rajtunk el lehet érni a másik ISP-t, és így tranzit hálózattá válhatnánk a két ISP szemszögéből.

(**[3]** „The Route Maps and Policy Routing” című fejezet)

### 3.2.9. Ellenőrző parancsok BGP-nél

Az R4 forgalomirányítón a ***sh ip bgp summary*** paranccsal meggyőződhetünk arról, hogy mind a két BGP szomszédjával sikerült kialakítani a kapcsolatot. Láthatunk egy számlálót, hogy mióta is él a kapcsolat, valamint az utolsó oszlopban a számok azt jelentik, hogy annyi darab útvonalat kapott tőlük BGP-n keresztül. Látható még a szomszédok autonóm rendszer száma is, amivel meg tudjuk állapítani, hogy az aktuális szomszéd az iBGP vagy eBGP szomszéd-e.

```
R4#sh ip bgp summ
BGP router identifier 172.16.1.1, local AS number 6000
BGP table version is 63, main routing table version 63
17 network entries using 1989 bytes of memory
17 path entries using 884 bytes of memory
7/6 BGP path/bestpath attribute entries using 868 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 3789 total bytes of memory
BGP activity 34/17 prefixes, 38/21 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
100.0.0.1       4  3000      55      58       63    0    0 00:01:37        3
200.0.0.1       4  2000      61      70       63    0    0 00:02:06        3
```

**40. ábra**

A forgalomirányítási táblában láthatjuk, hogy az R3-al ellentétben, az R4 tud a 100.0.0.0/16-os és a 200.0.0.0/16-os hálózatról is. Ezeket a BGP szomszédoktól tanulta meg, nem is OSPF címkével vannak ellátva.

```
R4#sh ip route

Gateway of last resort is not set

     100.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
B       100.0.1.1/32 [20/0] via 100.0.0.1, 00:01:32
C       100.0.0.0/30 is directly connected, Serial1/1
B       100.0.2.1/32 [20/0] via 100.0.0.1, 00:01:32
     200.0.0.0/30 is subnetted, 1 subnets
C       200.0.0.0 is directly connected, Serial1/0
     200.0.1.0/32 is subnetted, 1 subnets
B       200.0.1.1 [20/0] via 200.0.0.1, 00:02:02
     200.0.2.0/32 is subnetted, 1 subnets
B       200.0.2.1 [20/0] via 200.0.0.1, 00:02:02
     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       172.16.1.1/32 is directly connected, Loopback1
C       172.16.0.0/24 is directly connected, FastEthernet0/0
O       172.16.3.1/32 [110/2] via 172.16.0.3, 00:02:40, FastEthernet0/0
O       172.16.2.1/32 [110/2] via 172.16.0.5, 00:02:40, FastEthernet0/0
     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
O IA    10.0.2.1/32 [110/2] via 172.16.0.3, 00:02:40, FastEthernet0/0
O IA    10.0.1.1/32 [110/2] via 172.16.0.3, 00:02:40, FastEthernet0/0
O IA    10.0.0.0/30 [110/65] via 172.16.0.3, 00:02:29, FastEthernet0/0
O IA    10.2.2.1/32 [110/66] via 172.16.0.3, 00:02:19, FastEthernet0/0
O IA    10.1.1.1/32 [110/66] via 172.16.0.3, 00:02:19, FastEthernet0/0
O IA 192.168.0.0/20 [110/66] via 172.16.0.3, 00:02:19, FastEthernet0/0
O IA 192.168.16.0/20 [110/2] via 172.16.0.5, 00:02:40, FastEthernet0/0
```

**41. ábra**

Ezeket az utakat tettük aztán bele az OSPF folyamatba, ám a ***route-map BGP_TO_OSPF*** hatására, a többieknél csak a 200.0.0.0/16-os hálózatbeli címek jelentek meg.

```
R3#sh ip route
Gateway of last resort is 172.16.0.4 to network 0.0.0.0

     200.0.1.0/32 is subnetted, 1 subnets
O E1    200.0.1.1 [110/445] via 172.16.0.4, 00:01:46, FastEthernet0/0
     200.0.2.0/32 is subnetted, 1 subnets
O E1    200.0.2.1 [110/445] via 172.16.0.4, 00:01:46, FastEthernet0/0
     172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
O       172.16.1.1/32 [110/2] via 172.16.0.4, 00:02:24, FastEthernet0/0
C       172.16.0.0/24 is directly connected, FastEthernet0/0
O       172.16.0.0/16 is a summary, 00:02:24, Null0
C       172.16.3.1/32 is directly connected, Loopback3
O       172.16.2.1/32 [110/2] via 172.16.0.5, 00:02:24, FastEthernet0/0
     10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks
C       10.0.2.1/32 is directly connected, Loopback11
C       10.0.1.1/32 is directly connected, Loopback10
C       10.0.0.0/30 is directly connected, Serial1/1
O       10.0.0.0/8 is a summary, 00:02:04, Null0
O       10.2.2.1/32 [110/65] via 10.0.0.1, 00:02:04, Serial1/1
O       10.1.1.1/32 [110/65] via 10.0.0.1, 00:02:04, Serial1/1
O*E2 0.0.0.0/0 [110/1] via 172.16.0.4, 00:02:04, FastEthernet0/0
O IA 192.168.0.0/20 [110/65] via 10.0.0.1, 00:02:04, Serial1/1
O IA 192.168.16.0/20 [110/2] via 172.16.0.5, 00:02:04, FastEthernet0/0
```

**42. ábra**

Az R6 és az R7 forgalomirányítók kimenetiben elvi eltérés nincs. Mind a kettőnél látható, hogy sikerült kialakítani a BGP szomszédsági viszonyt az R4 forgalomirányítóval.

```
R6#sh ip bgp summ
BGP router identifier 200.0.2.1, local AS number 2000
BGP table version is 37, main routing table version 37
14 network entries using 1638 bytes of memory
14 path entries using 728 bytes of memory
6/5 BGP path/bestpath attribute entries using 744 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 3134 total bytes of memory
BGP activity 21/7 prefixes, 26/12 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
200.0.0.2       4  6000      70      62       37    0    0 00:02:22        11
```

**43. ábra**

A State/PfxRcd oszlopban látható szám, a szomszédtól megkapott utak számát jelenti, ami jelen esetben 11, tehát az R4-től 11 útvonalat tanultak meg a BGP segítségével.

```
R7#sh ip bgp summ
BGP router identifier 100.0.2.1, local AS number 3000

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
100.0.0.2       4  6000      58      56       19    0    0 00:01:58        11
```

**44. ábra**

A forgalomirányítási táblában látszik, hogy BGP-n keresztül megtanulták az összes OSPF hálózatot, viszont az R7-hez kapcsolódó, 100.0.0.0/16-os címekről nem tud. Ezt a ***route-map OSPF_ONLY***-nak köszönhetjük, amit az előbb már megvizsgáltunk.

```
R6#sh ip route
Gateway of last resort is not set

     200.0.0.0/30 is subnetted, 1 subnets
C       200.0.0.0 is directly connected, Serial1/0
     200.0.1.0/32 is subnetted, 1 subnets
C       200.0.1.1 is directly connected, Loopback1
     200.0.2.0/32 is subnetted, 1 subnets
C       200.0.2.1 is directly connected, Loopback2
     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
B       172.16.1.1/32 [20/0] via 200.0.0.2, 00:02:21
B       172.16.0.0/24 [20/0] via 200.0.0.2, 00:02:21
B       172.16.3.1/32 [20/2] via 200.0.0.2, 00:02:21
B       172.16.2.1/32 [20/2] via 200.0.0.2, 00:02:21
     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
B       10.0.2.1/32 [20/2] via 200.0.0.2, 00:02:21
B       10.0.1.1/32 [20/2] via 200.0.0.2, 00:02:21
B       10.0.0.0/30 [20/65] via 200.0.0.2, 00:02:21
B       10.2.2.1/32 [20/66] via 200.0.0.2, 00:02:21
B       10.1.1.1/32 [20/66] via 200.0.0.2, 00:02:21
B    192.168.0.0/20 [20/66] via 200.0.0.2, 00:02:21
B    192.168.16.0/20 [20/2] via 200.0.0.2, 00:02:21
```

**45. ábra**

# 4. Összefoglalás

A forgalomirányítás megjelenik mindennapjainkban, mindenhol használjuk, még ha néha nem is tudunk róla. Dolgozatomban bemutatásra került az OSPF belső irányító protokoll, valamint a BGP, mint külső irányítóprotokoll elméleti bemutatása. Az elméletben tárgyalt részek aztán alkalmazásra is kerültek egy teszt hálózaton. Az itt leírtak tisztán Cisco környezetben kerültek megvalósításra.

Első lépésben megvizsgáltam a szomszédok kialakulásának folyamatát. Szó esett az OSPF területek fontosságáról, valamint fajtáiról. Megvizsgáltam miért is van értelme használni a nevesített területeket, hiszen ezekkel levehettem a terhet a gyengébb forgalomirányítók válláról. Bepillanthattam a DR/BDR folyamat lejátszódásába, és ennek befolyásolására is volt lehetőségem. Megnéztem, mi a teendő, ha nincs lehetőség minden területet a szabvány szerint a gerinc területhez csatolni. Ilyen helyzetekben egy virtuális kapcsolatot voltam kénytelen használni. Végül pedig a hitelesítést is bekapcsoltam, így a szomszédsági viszony már csak azon eszközök között alakulhat ki, amelyek ugyanazon hitelesítési beállításokkal vannak ellátva. A BGP irányítóprotokoll is megjelent a topológiában. Itt első sorban maga a konfigurációja kapott szerepet, két módszert is megnéztem erre. Szó volt a BGP szomszédok kialakulásáról és esetleges hibákról, amiért nem jöhet létre a kapcsolat. Természetesen itt is beszéltem hitelesítésről, csak úgy, mint az OSPF-nél. Legvégül magát a forgalomirányításhoz szükséges beállításokat konfiguráltam be, amivel lehetővé vált a kommunikáció immár nem csak az egy autonóm rendszeren belüli forgalomirányítók között. Itt olyan manipulatív megoldásokat használtam, amik egy működő, operatív hálózatban is fellelhetőek. A BGP protokoll használata elengedhetetlen, hiszen nincs más alternatíva, amit használni lehetne ilyen helyzetben.

Természetesen, ezek a megoldások nem állnák meg helyüket ebben a formában, egy produktív hálózatban, de a tesztkörnyezet jelen esetben, lehetőséget adott minden tárgyalt részlet bemutatására.

Munkám során főleg a BGP-nél megtanult elméleti dolgoknak veszem hasznát, hiszen sokszor csak a végberendezésekért vagyunk felelősek. Itt a működési sajátosságokat, amiket a teszthálózaton megfigyeltem hasznát veszem hibafelderítésnél és az elhárítási folyamatban egyaránt.

# 5. Irodalomjegyzék

**[1]** Brent D. Stewart and Clare Gough (2008), CCNP Exam preparation: CCNP BSCI Official Exam Certification Guide, Fourth Edition. Cisco Press, USA

**[2]** Todd Lammle, Carl Timm and Sean Odom (2002), CCIP: BSCI Study Guide. Sybex Inc., Alameda, CA

**[3]** Chris Bryant (2007), The Ultimate BSCI Study Guide. The Bryant Advantage

### Internetes hivatkozások:

**[4]** OSPF Design Guide (PDF)
http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094e9e.shtml

**[5]** BGP Case Studies (PDF)
http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a00800c95bb.shtml

**[6]** RFC 1771, Rekhter, Y. and T. Li, „A Border Gateway Protocol 4 (BGP-4)", RFC 1771, March 1995.

**[7]** RFC 2328, J. Moy, „OSPF Version 2", RFC 2328, April 1998

**[8]** RFC 4271, Rekhter, Y., T. Li and S. Hares, „A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006

# 6. Mellékletek

### R1

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R1
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$CwJ.$Jz34xVuy.xnfaiccBrCmT0
!
no aaa new-model
!
!
ip cef
!
!
interface Loopback1
 ip address 192.168.1.1 255.255.255.255
!
interface Loopback2
 ip address 192.168.2.1 255.255.255.255
!
interface Loopback3
 ip address 192.168.3.1 255.255.255.255
!
interface Serial1/0
 description *** TO R2 ***
 ip address 192.168.0.1 255.255.255.252
 serial restart-delay 0
!
router ospf 1
 log-adjacency-changes
 area 1 stub no-summary
 network 192.168.0.0 0.0.0.3 area 1
 network 192.168.1.1 0.0.0.0 area 1
 network 192.168.2.1 0.0.0.0 area 1
 network 192.168.3.1 0.0.0.0 area 1
!
ip forward-protocol nd
!
no ip http server
no ip http secure-server
```

```
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
!
!
end
```

**R2**

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R2
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$T/GK$ZT4iRDaY5Re2.2bvxGb1D.
enable password cisco2
!
no aaa new-model
!
!
ip cef
!
!
interface Loopback4
 ip address 192.168.4.1 255.255.255.255
!
interface Loopback10
 ip address 10.1.1.1 255.255.255.255
!
interface Loopback11
 ip address 10.2.2.1 255.255.255.255
```

```
!
interface Serial1/0
 description *** TO R1 ***
 ip address 192.168.0.2 255.255.255.252
 serial restart-delay 0
 clock rate 64000
!
interface Serial1/1
 description *** TO R3 ***
 ip address 10.0.0.1 255.255.255.252
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 cisco
 serial restart-delay 0
 clock rate 64000
!
router ospf 1
 log-adjacency-changes
 area 1 stub no-summary
 area 1 range 192.168.0.0 255.255.240.0
 network 10.0.0.0 0.255.255.255 area 0
 network 192.168.0.0 0.0.0.3 area 1
 network 192.168.4.1 0.0.0.0 area 1
!
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

**R3**

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname R3
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$fE8P$tU7Z.eBh9jWdDzZREvO7u0
enable password 7 1511021F072579
!
no aaa new-model
!
!
ip cef
!
!
interface Loopback3
 ip address 172.16.3.1 255.255.255.255
!
interface Loopback10
 ip address 10.0.1.1 255.255.255.255
!
interface Loopback11
 ip address 10.0.2.1 255.255.255.255
!
interface FastEthernet0/0
 description *** AREA 2 TO SW ***
 ip address 172.16.0.3 255.255.255.0
 ip ospf priority 10
 duplex auto
 speed auto
!
interface Serial1/1
 description *** TO R2 ***
 ip address 10.0.0.2 255.255.255.252
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 7 13061E010803
 serial restart-delay 0
!
router ospf 1
 router-id 3.3.3.3
 log-adjacency-changes
 area 0 range 10.0.0.0 255.0.0.0
 area 2 range 172.16.0.0 255.255.0.0
 area 2 virtual-link 5.5.5.5
```

```
 network 10.0.0.0 0.0.0.3 area 0
 network 10.0.1.1 0.0.0.0 area 0
 network 10.0.2.1 0.0.0.0 area 0
 network 172.16.0.0 0.0.0.255 area 2
 network 172.16.3.1 0.0.0.0 area 2
!
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

**R4**

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R4
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$UFFL$G.ula1nd284gM4zwwYL6C0
!
no aaa new-model
!
!
ip cef
```

```
!
!
interface Loopback1
 ip address 172.16.1.1 255.255.255.255
!
interface FastEthernet0/0
 description *** AREA 2 TO SW ***
 ip address 172.16.0.4 255.255.255.0
 ip ospf priority 0
 duplex auto
 speed auto
!
interface Serial1/0
 description *** TO R6 - AS 2000 ***
 ip address 200.0.0.2 255.255.255.252
 serial restart-delay 0
!
interface Serial1/1
 description *** TO R7 - AS 3000 ***
 ip address 100.0.0.2 255.255.255.252
 serial restart-delay 0
!
router ospf 1
 router-id 4.4.4.4
 log-adjacency-changes
 summary-address 172.16.0.0 255.255.0.0
 redistribute bgp 6000 metric 444 metric-type 1 subnets route-map BGP_TO_OSPF
 network 172.16.0.0 0.0.0.255 area 2
 network 172.16.1.1 0.0.0.0 area 2
 default-information originate always
!
router bgp 6000
 no synchronization
 bgp log-neighbor-changes
 redistribute ospf 1
 neighbor AS2000 peer-group
 neighbor AS2000 remote-as 2000
 neighbor AS2000 route-map OSPF_ONLY out
 neighbor 100.0.0.1 remote-as 3000
 neighbor 100.0.0.1 password 5 cisco
 neighbor 100.0.0.1 route-map OSPF_ONLY out
 neighbor 200.0.0.1 peer-group AS2000
 no auto-summary
!
ip forward-protocol nd
!
no ip http server
no ip http secure-server
```

```
!
!
ip access-list standard NO_AS
 deny   100.0.0.0 0.0.255.255
 deny   200.0.0.0 0.0.255.255
 permit any
!
ip access-list standard ONLY_AS_2000
 permit 200.0.0.0 0.0.255.255
!
route-map OSPF_ONLY permit 10
 match ip address NO_AS
!
route-map BGP_TO_OSPF permit 10
 match ip address ONLY_AS_2000
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

**R5**

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R5
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$bEUO$CtT7e1oZ8TWQUouKmjBch.
```

```
!
no aaa new-model
!
!
ip cef
no ip domain lookup
!
!
interface Loopback2
 ip address 172.16.2.1 255.255.255.255
!
interface Loopback19
 ip address 192.168.19.1 255.255.255.255
!
interface FastEthernet0/0
 description *** AREA 2 TO SW ***
 ip address 172.16.0.5 255.255.255.0
 ip ospf priority 5
 duplex auto
 speed auto
!
interface Serial1/0
 description *** TO R8 - AREA 3 ***
 ip address 192.168.16.1 255.255.255.252
 serial restart-delay 0
 clock rate 64000
!
router ospf 1
 router-id 5.5.5.5
 log-adjacency-changes
 area 0 range 10.0.0.0 255.0.0.0
 area 2 range 172.16.0.0 255.255.0.0
 area 2 virtual-link 3.3.3.3
 area 3 stub
 area 3 range 192.168.16.0 255.255.240.0
 network 172.16.0.0 0.0.0.255 area 2
 network 172.16.2.1 0.0.0.0 area 2
 network 192.168.16.0 0.0.0.3 area 3
 network 192.168.19.1 0.0.0.0 area 3
!
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
!
control-plane
!
```

```
!
gatekeeper
 shutdown
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

**R6**

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R6
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$KGCb$R.l83vqMFvbQ/HbkCU6wc/
!
no aaa new-model
!
!
ip cef
!
!
interface Loopback1
 ip address 200.0.1.1 255.255.255.255
!
interface Loopback2
 ip address 200.0.2.1 255.255.255.255
!
interface Serial1/0
 description *** TO R4 - AS 6000 ***
 ip address 200.0.0.1 255.255.255.252
 serial restart-delay 0
 clock rate 64000
!
```

```
router bgp 2000
 no synchronization
 bgp log-neighbor-changes
 redistribute connected
 neighbor 200.0.0.2 remote-as 6000
 no auto-summary
!
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

**R7**

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R7
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$g4aH$cH.mXvDLdGZKPIi3bBe7r1
!
no aaa new-model
!
!
```

```
ip cef
!
!
interface Loopback1
 ip address 100.0.1.1 255.255.255.255
!
interface Loopback2
 ip address 100.0.2.1 255.255.255.255
!
interface Serial1/1
 description *** TO R4 - AS 6000 ***
 ip address 100.0.0.1 255.255.255.252
 serial restart-delay 0
 clock rate 64000
!
router bgp 3000
 no synchronization
 bgp log-neighbor-changes
 redistribute connected
 neighbor 100.0.0.2 remote-as 6000
 neighbor 100.0.0.2 password 5 cisco
 no auto-summary
!
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

**R8**

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R8
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$2IDL$LTn9diOnkMKKTtMm47z14.
!
no aaa new-model
!
!
ip cef
!
!
interface Loopback17
 ip address 192.168.17.1 255.255.255.255
!
interface Loopback18
 ip address 192.168.18.1 255.255.255.255
!
interface Serial1/0
 description *** TO R5 ***
 ip address 192.168.16.2 255.255.255.252
 serial restart-delay 0
!
router ospf 1
 log-adjacency-changes
 area 3 stub
 network 192.168.16.0 0.0.0.3 area 3
 network 192.168.17.1 0.0.0.0 area 3
 network 192.168.18.1 0.0.0.0 area 3
!
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
!
control-plane
!
!
gatekeeper
 shutdown
!
```

```
!
line con 0
 exec-timeout 0 0
 logging synchronous
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

# 7. Köszönetnyilvánítás

A dolgozatom végén szeretnék köszönetet mondani témavezetőmnek, Dr. Almási Béla egyetemi docensnek, amiért segítségével, szakmai útmutatásával lehetővé tette számomra, hogy ez a dolgozat létrejöjjön. Valamint mindazoknak, akik hozzájárultak a munkámhoz.