közben megrongálja azokat, gondok merülnek fel. Végül, ha kifogy a memóriából, vagy rosszul végzi el az útvonalak kiszámítását, csúnya dolgok fognak történni. Ahogy az alhálózat a tíz- vagy százezres csomóponti nagyságrendbe nő, annak a valószínűsége, hogy alkalmanként egy router meghibásodik, már nem lesz elhanyagolható. A trükk az, hogy próbáljuk a kárt korlátozni, amikor az elkerülhetetlenül bekövetkezik. Perlman (1988) részletesen tárgyalja ezeket a problémákat és a megoldásaikat.

A kapcsolatállapot alapú forgalomirányítást széles körben használják a jelenlegi hálózatokban, ezért helyénvaló pár szót szólni az ezeket használó példaprotokollokról. Az OSPF protokoll, amelyiket egyre többen használnak az Interneten, egy kapcsolatállapot alapú algoritmust használ. Az OSPF-et az 5.5.5. részben írjuk le.

Egy másik fontos kapcsolatállapot alapú protokoll az **IS-IS (Intermediate System – Intermediate System)**, amelyet a DECnet számára terveztek, és amelyet később átvett az ISO is az összeköttetés nélküli hálózati rétegbeli protokolljába, a CLNP-be. Azóta módosították, hogy más protokollokat is kezeljen, különösen az IP-t. Az IS-IS-t számos Internet-gerinchálózatban (például a régi NFSNET gerinchálózatban is), és néhány digitális cellás rendszerben használják, mint amilyen a CDPD. A Novell NetWare az IS-IS egy kisebb változatát használja (az NLSP-t) az IPX csomagok forgalomirányításához.

Alapvetően az IS-IS egy képet oszt szét a routerek topológiájáról, amelyből számítják a legrövidebb utat. Minden router bejelenti a kapcsolatállapot információiban, hogy mely hálózati rétegbeli címeket tud közvetlenül elérni. Ezek a címek lehetnek IP, IPX, AppleTalk, vagy bármilyen más címek. Az IS-IS egy időben többféle hálózati rétegbeli protokollt is támogathat.

Az IS-IS számára tervezett újításokból sokat átvett az OSPF is (az OSPF-et évekkel az IS-IS után tervezték). Ezek közt olyanok vannak, mint a kapcsolatállapot frissítések elárasztásának egy önstabilizáló eljárása, a kijelölt router koncepciója egy LAN-on, valamint az utak kettéosztásának és többfajta mérték használatának számításai és támogatási módszere. Ennek következtében csak nagyon kicsi a különbség az OSPF és az IS-IS között. A legfontosabb különbség, hogy az IS-IS olyan módon kódolt, hogy azon egyszerű és természetes egyszerre többfajta hálózati rétegbeli protokoll információit szállítani, míg az OSPF ezzel a tulajdonsággal nem rendelkezik. Ez az előny különösen értékes nagy, többprotokollos környezetekben.

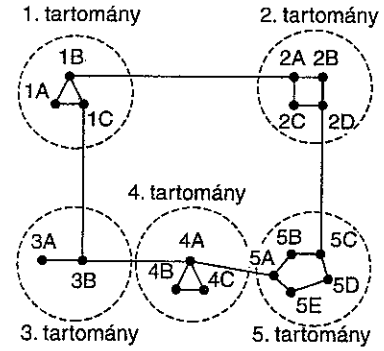
5.2.7. Hierarchikus forgalomirányítás

Ahogy a hálózat mérete nő, a routerek forgalomirányító táblázatai is arányosan nőnek. Nemcsak a routerek memóriáját fogyasztják az egyre növekvő táblázatok, hanem több CPU időre is van szükség az átvizsgálásukhoz, és nagyobb sávszélesség szükséges ahhoz, hogy elküldjük ezek állapotjelentéseit. Egyszer csak a hálózat akkorára nő, hogy már nem lehetséges minden egyes routerben minden más router számára egy bejegyzést fenntartani, ezért a forgalomirányítást hierarchikusan kell elvégezni, ahogy azt a telefonhálózatban is teszik.

Amikor hierarchikus forgalomirányítást használunk, a routereket úgynevezett **tartományokra (regions)** osztjuk. Minden router tudja, hogyan irányítsa a saját tartományán belüli célok felé a csomagokat, de nem tud semmit a többi tartomány belső szerkezetéről. Amikor különböző hálózatokat kapcsolunk össze, természetes, hogy mindegyiket különálló tartománynak tekintjük, és így az egyik hálózaton belüli routereknek nem kell tudniuk a többi hálózat topologikus szerkezetéről.

Hatalmas hálózatok esetén kétszintű hierarchia elégtelen lehet; szükség lehet arra, hogy a tartományokat kerületekbe, a kerületeket zónákba, a zónákat csoportokba szervezzük és így tovább, amíg csak ki nem fogyunk az egyesítésekre használt nevekből. A többszintű hierarchia egy példájaként nézzük meg, hogyan irányíthatnak egy csomagot a kaliforniai Berkeleyből a kenyai Malindibe. A berkeleybeli router ismerné a Kalifornián belüli részletes topológiát, de minden, az államból kifelé tartó forgalmat a Los Angeles-i routerhez küldene. A Los Angeles-i router képes lenne a többi belföldi routerhez irányítani a forgalmat, de a külföldi forgalmat New Yorkba küldené. A New York-i router úgy lenne felprogramozva, hogy minden forgalmat a célország külföldi forgalomért felelős routeréhez küldjön, mondjuk Nairobiba. Végül a csomag lefelé haladna a kenyai fán belül, amíg el nem éri Malindit.

Az 5.17. ábra egy mennyiségi példát ad a forgalomirányításra egy kétszintű hierarchiában, öt tartománnyal. Az 1*A* teljes forgalomirányító táblázata 17 bejegyzést tartalmaz, ahogy az az 5.17.(b). ábrán látszik. Amikor a forgalomirányítást hierarchikusan végezzük, mint az 5.17.(c) ábrán, akkor, mint előzőleg, vannak a helyi routerekre vonatkozó bejegyzések, de a többi tartományt egy-egy csomópontba sűrítettük össze. Így a 2. tartomány felé tartó összes forgalom az 1*B*–2*A* vonalon keresztül halad, de a



1*A* teljes táblázata

Cél	Vonal	Átugrások
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

1*A* hierarchikus táblázata

Cél	Vonal	Átugrások
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

5.17. ábra. *Hierarchikus forgalomirányítás*

távoli forgalom többi része az 1*C*–3*B* vonalat használja. A hierarchikus forgalomirányítás a táblázat méretét 17 bejegyzésről 7-re csökkentette. Ahogy a tartományok számának tartományonkénti routerek számához viszonyított aránya nő, úgy növekszik a megtakarítás a táblázathelyekben.

Sajnos, ez a helymegtakarítás nincs ingyen. Ezért büntetést kell fizetni, mégpedig a megnövekedett úthossz formájában. Például az 1*A* és 5*C* közti legjobb út a 2-es tartományon át vezet, de a hierarchikus forgalomirányításnál minden, az 5-ös tartomány felé tartó forgalom a 3-as tartományon keresztül zajlik, mert az 5-ös tartományon belül a legtöbb célhoz ez a jobb.

Amikor egy egyedülálló hálózat nagyon nagyra nő, egy érdekes kérdés merül fel: hány szintje legyen a hierarchiának? Vegyünk például egy 720 routerből álló alhálózatot. Ha nincs hierarchia, minden routernek 720 táblázatbejegyzésre van szüksége. Ha az alhálózatot felosztjuk 24 tartományra, amelyek mindegyikében 30 router van, minden routernek 30 helyi és 23 távoli bejegyzésre van szüksége, ez összesen 53 bejegyzés. Ha háromszintű hierarchiát választunk, nyolc kerülettel, amelyek mindegyike 9, tízrouteres tartományból áll, minden routernek 10 bejegyzésre van szüksége a helyi routerekhez, 8 bejegyzésre a saját kerületen belüli tartományokhoz, és 7 bejegyzésre a távoli kerületekhez, ez összesen 25 bejegyzés. Kamoun és Kleinrock (1979) felfedezték, hogy egy N routerből álló alhálózathoz a szintek optimális száma $\ln N$, amely $e^* \ln N$ bejegyzést igényel routerenként. Azt is megmutatták, hogy a hierarchikus forgalomirányítás által okozott átlagos valódi úthosszúság-növekedés elegendően kicsi ahhoz, hogy általában elfogadható legyen.

5.2.8. Forgalomirányítás mozgó hosztok esetében

Manapság emberek milliói rendelkeznek hordozható számítógéppel, és általában bárhol is vannak a világon, el akarják olvasni az elektronikus leveleiket, és el akarják érni a rendes fájlrendszerüket. Ezek a mozgó hosztok egy új bonyodalmat jelentenek: ahhoz, hogy egy csomagot egy mozgó hoszthoz irányíthassunk, a hálózatnak először meg kell találnia azt. A mozgó hosztok hálózatba való bevonásának témaköre még nagyon fiatal, de ebben a részben felvázolunk néhány idevágó kérdést és egy lehetséges megoldást adunk.

A hálózattervezők által tipikusan használt világmodellt az 5.18. ábra mutatja. Van egy WAN-unk, amely routerekből és hosztokból áll. A WAN-hoz LAN-ok, MAN-ok és olyan drót nélküli cellák kapcsolódnak, amelyeket a 2. fejezetben tanulmányoztunk.

A soha nem mozgó felhasználókat mozdulatlan (stationary) felhasználóknak nevezzük. Ők a hálózathoz rézvezetékeken vagy üvegszálakon keresztül kapcsolódnak. Ezzel ellentétben két másfajta felhasználót tudunk megkülönböztetni. A vándorló (migratory) felhasználók alapvetően mozdulatlan felhasználók, akik időről időre egyik rögzített helyről a másikra költöznek, de csak akkor használják a hálózatot, amikor fizikailag kapcsolódnak is hozzá. A kóborló felhasználók valóban mozgás közben használják a számítógépet, és fenn akarják tartani a kapcsolataikat, miközben mozognak. A **mozgó felhasználó (mobile user)** kifejezést mindkét utóbbi kategóriára használjuk, vagyis minden felhasználóra, aki nem otthon van.

Feltesszük, hogy minden felhasználónak van egy **állandó lakhelye (permanent home location),** amely soha nem változik. A felhasználóknak van egy állandó lakcímük is, amely felhasználható a lakhely meghatározására, hasonlóan, ahogy az 1-212-5551212 telefonszámból kiolvasható az Egyesült Államok (1-es országkód) és Manhattan (212). A forgalomirányítás célja a mozgó felhasználókkal rendelkező rendszerekben az, hogy lehetővé tegyük a mozgó felhasználók számára csomagok küldését a lakcímükre, és hogy a csomagok hatékonyan érjék el őket, bárhol is legyenek. Természetesen a trükk az, hogy meg kell őket találni.

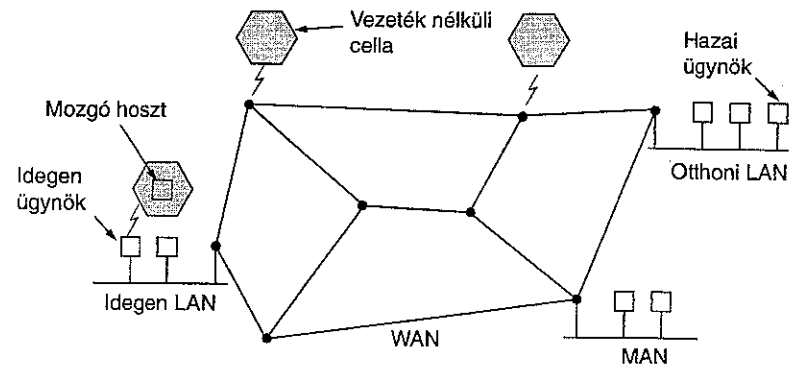
Az 5.18. ábra modelljében, a világ (földrajzilag) fel van osztva kis egységekre. Hívjuk ezeket körzetnek, ahol egy körzet tipikusan egy LAN vagy vezeték nélküli cella. Minden körzetnek van egy vagy több **idegen ügynöke (foreign agent),** amely nyilvántartja az összes, ebbe a körzetbe látogató mozgó felhasználót. Ezen kívül minden körzetnek van egy **hazai ügynöke (home agent)** is, amely azon felhasználókat tartja nyilván, akiknek itt van a lakhelyük, de most éppen más körzetbe látogattak el.

Amikor egy új felhasználó belép a körzetbe, vagy rákapcsolódással (például egy LAN-ba való bekötéssel), vagy egyszerűen a cellába lépés útján, a számítógépének regisztráltatnia kell magát az ottani idegen ügynöknél. A bejegyeztetési folyamat tipikusan a következőképpen működik:

1. Minden egyes idegen ügynök periodikusan szétküld egy csomagot, amelyben létezését és a címét közli. Egy újonnan érkezett mozgó hoszt várhat egy ilyen üzenetre, de ha egy ilyen sem érkezik elég gyorsan, a mozgó hoszt is szétküldhet egy csomagot, „Van itt idegen ügynök?" tartalommal.

2. A mozgó hoszt bejegyezteti magát az idegen ügynöknél, megadja a lakcímét, jelenlegi adatkapcsolati rétegbeli címét, és valamilyen biztonsági információt.

3. Az idegen ügynök kapcsolatba lép a mozgó hoszt hazai ügynökével, és azt mondja: „Az egyik hosztod itt van." Az üzenet, amely az idegen ügynöktől a hazai ügynökig megy, tartalmazza az idegen ügynök hálózati címét. Tartalmaz továbbá biztonsá-



5.18. ábra. *Egy WAN, amelyhez LAN-ok, MAN-ok, és vezeték nélküli cellák kapcsolódnak*

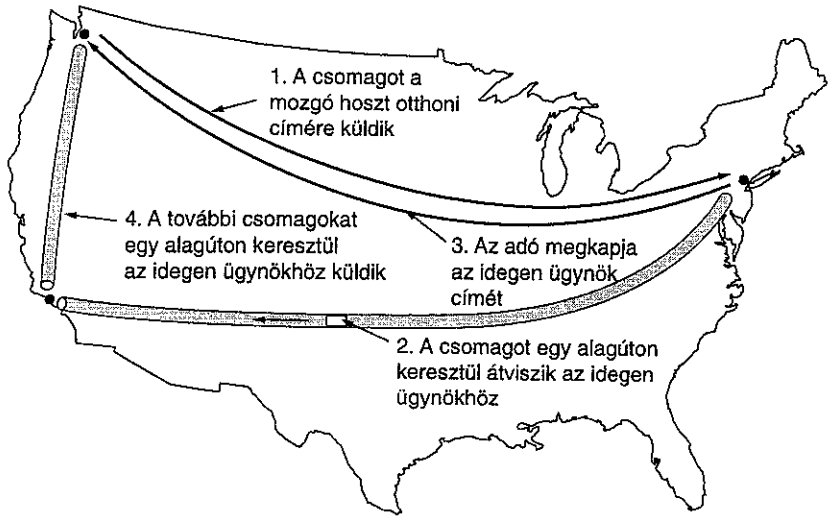
gi információt is, hogy meggyőzze a hazai ügynököt arról, hogy a mozgó hoszt valóban ott van.

4. A hazai ügynök megvizsgálja a biztonsági információt, amely egy időbélyeget is tartalmaz annak bizonyítására, hogy azt az elmúlt pár másodpercben hozták létre. Ha ezzel meg van elégedve, utasítja az idegen ügynököt a folytatásra.

5. Amikor az idegen ügynök megkapja a nyugtát a hazai ügynöktől, létrehoz egy bejegyzést a táblázataiban, és tudatja a mozgó hoszttal, hogy a regisztrációja megtörtént.

Ideális esetben, amikor egy felhasználó elhagy egy körzetet a regisztráció törlése végett, ezt be kellene jelenteni, azonban sok felhasználó csak kikapcsolja a számítógépét, amikor végzett.

Amikor egy csomagot küldenek a mozgó felhasználónak, azt a felhasználó otthoni LAN-jára irányítják, mert a cím szerint ezt kell tenni, ahogy azt az 5.19. ábra 1. lépése mutatja. A mozgó felhasználó számára az otthoni LAN-ra érkező csomagokat a hazai ügynök elfogja. A hazai ügynök ezután megkeresi a mozgó felhasználó új (ideiglenes) helyét, és megkeresi a mozgó felhasználót kezelő idegen ügynök címét. Ezután a hazai ügynök két dolgot tesz. Elsőként beágyazza a csomagot egy külső csomag adat mezejébe, és e csomagot elküldi az idegen ügynöknek (2. lépés az 5.19. ábrán). Ezt a mechanizmust alagút típusú továbbításnak hívják, később részletesebben is megnézzük. Miután megkapta a beágyazott csomagot, az idegen ügynök kiveszi az eredeti csomagot az adat mezőből, és adatkapcsolati keretként küldi el a mozgó felhasználónak.



5.19. ábra. *Csomagforgalom-irányítás mozgó felhasználók számára*

Másodikként, a hazai ügynök utasítja az adót, hogy ezek után a mozgó felhasználónak a csomagokat úgy küldje el, hogy ágyazza be azokat a kimondottan az idegen ügynöknek címzett csomagok adat mezejébe, ahelyett, hogy csak a mozgó felhasználó otthoni címére küldené (3. lépés). A további csomagokat már egyenesen a felhasználóhoz lehet irányítani az idegen ügynökön keresztül (4. lépés), teljesen kihagyva a lakhelyet.

A különböző javasolt megoldások több helyen is eltérnek egymástól. Először is az a kérdés, hogy a protokoll mekkora részét végezzék a routerek és mekkorát a hosztok, és a hosztok melyik rétege. Másodszor, néhány megoldásban az út menti routerek feljegyzik a leképzett címeket, így el tudják fogni és átirányítani a forgalmat, még mielőtt a lakhelyhez érne. Harmadszor, néhány megoldásban minden látogató egy egyedi ideiglenes címet kap; más megoldásokban az ideiglenes cím egy ügynökre hivatkozik, amely az összes látogatónak szóló forgalmat kezeli.

Negyedszer, a megoldások abban is különböznek, hogy hogyan oldják meg a gyakorlatban azt az esetet, amikor egy célcímre irányuló csomagot egy másik címre kell kézbesíteni. Egy választás, hogy változtassuk meg a célcímet és adjuk újra a módosított csomagot. Alternatívaként az egész csomagot, lakcímestül, mindenestül egy másik csomag adat mezejébe ágyazhatjuk be, amelyet az ideiglenes címre küldünk. Végül a megoldások a biztonsági megfontolások tekintetében is különböznek. Általában, amikor egy hoszt vagy router egy ilyen formájú üzenetet kap: „Mostantól kezdve kérlek, küldd el Cayla minden levelét nekem" lehet, hogy egy pár kérdése lesz arról, hogy kivel beszélt, és hogy ez vajon jó ötlet-e vagy sem. Számos mozgó hoszt protokollt tárgyal és hasonlít össze (Ioannidis és Maguire, 1993; Myles és Skellern, 1993; Perkins, 1993; Teraoka és mások, 1993; Wada és mások, 1993).

5.2.9. Adatszóró forgalomirányítás

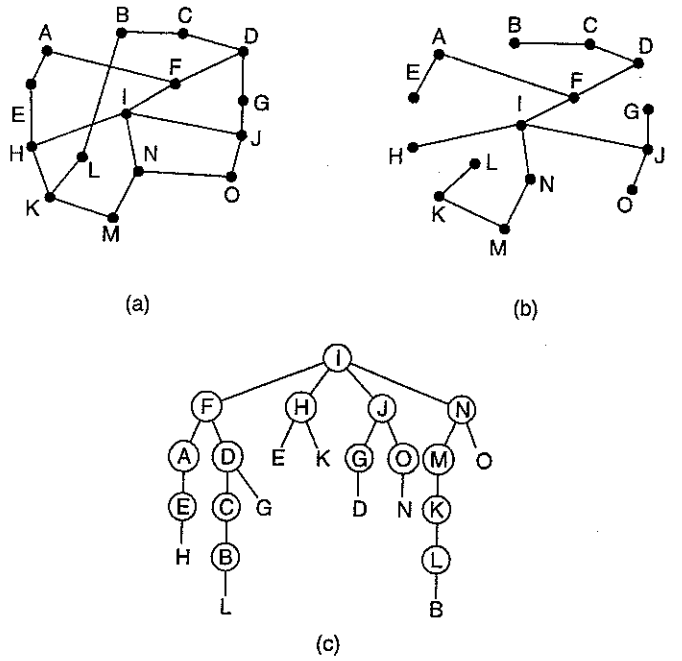
Néhány alkalmazásban a hosztoknak néhány vagy az összes többi hosztnak kell üzeneteket küldeniük. Például egy olyan szolgáltatás, amely időjárás-jelentést, tőzsdei híreket vagy élő rádióműsort oszt szét, a legjobban úgy működhet, hogy adatszórással minden géphez adatokat küld, és az érdeklődők elolvashatják azokat. Egy csomag mindenhová történő egyidejű elküldését **adatszórásnak (broadcasting)** nevezzük. Különböző módszereket javasoltak ennek megvalósítására.

Az egyik adatszóró eljárás, amely nem igényel semmi különleges tulajdonságot az alhálózattól, úgy működik, hogy a forrás egyszerűen külön csomagot küld minden egyes rendeltetési helyre. Ez a megoldás nemcsak sávszélesség-pazarló, hanem azt is megköveteli a forrástól, hogy rendelkezzen a rendeltetési helyek teljes listájával. A gyakorlatban lehet, hogy ez az egyetlen lehetőség, de a lehetséges eljárások közül ez a legkevésbé kívánatos.

Az elárasztás egy másik nyilvánvaló jelölt. Bár az elárasztás nem megfelelő a rendes kétpontos kommunikációhoz, az adatszóráshoz komolyan számításba vehető, különösen, ha az alább leírt módszerek közül egyik sem alkalmazható. A probléma az elárasztással, ha adatszórásra használjuk ugyanaz, mint a két pont közötti forgalomirányítás esetén: túl sok csomagot generál és túl nagy sávszélességet fogyaszt.

Egy harmadik algoritmus a **többcélú forgalomirányítás (multidestination routing)**. Ha ezt az eljárást használják, minden csomag tartalmaz vagy egy listát a rendeltetési helyekről vagy egy bit-térképet, amely a kívánt rendeltetési helyeket jelzi. Amikor egy csomag megérkezik egy routerhez, a router megnézi a rendeltetési helyek listáját, hogy eldöntse, mely kimeneti vonalakra lesz szükség. (Akkor van egy kimeneti vonalra szükség, ha az a legjobb út legalább egy rendeltetési hely felé.) A router előállítja a csomag egy új másolatát minden használandó kimeneti vonal számára, és csak azokat a célcímeket teszi bele, amelyeknek azt a vonalat kell használniuk. Ezzel a rendeltetési helyek halmazát felosztja a kimenő vonalak közt. Elegendő számú ugrás megtétele után minden csomag csak egy címet fog tartalmazni, és normális csomagként kezelhető. A többcélú forgalomirányítás olyan, mint a külön címzett csomagok esete, kivéve, hogy amikor több csomagnak is ugyanazt az útvonalat kell követnie, egyikük fizeti a teljes viteldíjat, a többi ingyen utazik.

Egy negyedik adatszóró algoritmus kifejezetten kihasználja az adatszórást kezdeményező routerhez tartozó nyelőfát, vagy bármely más kényelmes feszítőfát. Egy **feszítőfa (spanning tree)** az alhálózat egy részhalmaza, amelyben minden router benne van, de nem tartalmaz hurkokat. Ha minden router tudja, mely vonalai tartoznak a feszítőfához, egy bejövő adatszórásos csomagot minden, a feszítőfához tartozó vonalra kimásolhat, kivéve azt, amelyen érkezett. Ez az eljárás kitűnően használja ki a sávszé-



5.20. ábra. *Visszairányú továbbítás. (a) Egy alhálózat. (b) Egy feszítőfa. (c) A visszairányú továbbítás által felépített fa*

lességet, csak annyi csomagot hoz létre, amennyi mindenképpen szükséges a feladat elvégzéséhez. Az egyetlen probléma az, hogy minden routernek ismernie kell valamilyen feszítőfát, hogy alkalmazható legyen. Néha ez az információ rendelkezésre áll (pl. kapcsolatállapot alapú forgalomirányítás esetén), de néha nem (pl. távolságvektor alapú forgalomirányítás esetén).

Az utolsó adatszóró algoritmusunk egy kísérlet arra, hogy az előző algoritmus viselkedését megközelítsük, még akkor is, amikor a routerek nem tudnak semmit a feszítőfákról. Az ötlet figyelemre méltóan egyszerű, ha már egyszer megmutatták. Amikor egy adatszórásos csomag megérkezik egy routerhez, a router ellenőrzi, hogy azon a vonalon kapta-e meg, amelyen rendszerint ő szokott az adatszórás forrásához küldeni. Ha igen, akkor nagy esély van rá, hogy az adatszórásos csomag a legjobb utat követte a routertől, és ezért ez az első másolat, amely megérkezett a routerhez. Ha ez az eset, a router kimásolja minden vonalra, kivéve arra, amelyiken érkezett. Viszont, ha az adatszórásos csomag más vonalon érkezett, mint amit a forrás eléréséhez előnyben részesítünk, a csomagot eldobják, mint valószínű másodpéldányt.

Egy példa az algoritmusra, amelyet **visszairányú továbbításnak (reverse path forwarding)** neveznek, az 5.20. ábrán látható. Az (a) rész mutatja az alhálózatot, a (b) rész az alhálózat az *I* routerhez tartozó nyelőfáját, és a (c) rész mutatja, hogyan működik a visszairányú algoritmus. Az első ugrás során *I* csomagokat küld *F*-nek, *H*-nak, *J*-nek és *N*-nek, ahogy a fa második sora mutatja. Ezen csomagok mindegyike az *I* felé vezető előnyben részesített úton érkezett (feltételezve, hogy az előnyben részesített utak a nyelőfával egybeesnek), és ezt a betűk körüli kör jelzi. A második ugrás során nyolc csomag keletkezik, minden olyan router által kettő, amely az első ugrás során megkapta a csomagot. Amint kiderül, mind a nyolc még meg nem látogatott routerhez érkezik, és öt közülük az előnyben részesített vonalon. Abból a hat csomagból, amely a harmadik ugrás során keletkezik, csak három érkezik az előnyben részesített úton (*C*-hez, *E*-hez és *K*-hoz); a többi másodpéldány. Öt ugrás és 23 csomag után az adatszórás megszakad, ezzel összehasonlítva, a nyelőfa pontos követésekor négy ugrásra és 14 csomagra lett volna szükség.

A visszairányú továbbítás legfontosabb előnye, hogy ésszerűen hatékony és könnyen megvalósítható. Nem követeli meg a routerektől, hogy tudjanak feszítőfákról, és a céllista vagy bit-térkép által okozott többlet sincs jelen minden adatszórásos csomagban, mint a többcélú forgalomirányításnál. A folyamat megállításához sincs szükség külön mechanizmusokra, mint ahogy az elárasztás esetében szükség volt (vagy minden csomagban egy átugrásszámláló és az alhálózat átmérőjének előzetes ismerete, vagy forrásonként a már látott csomagok listája).

5.2.10. Többesküldéses forgalomirányítás

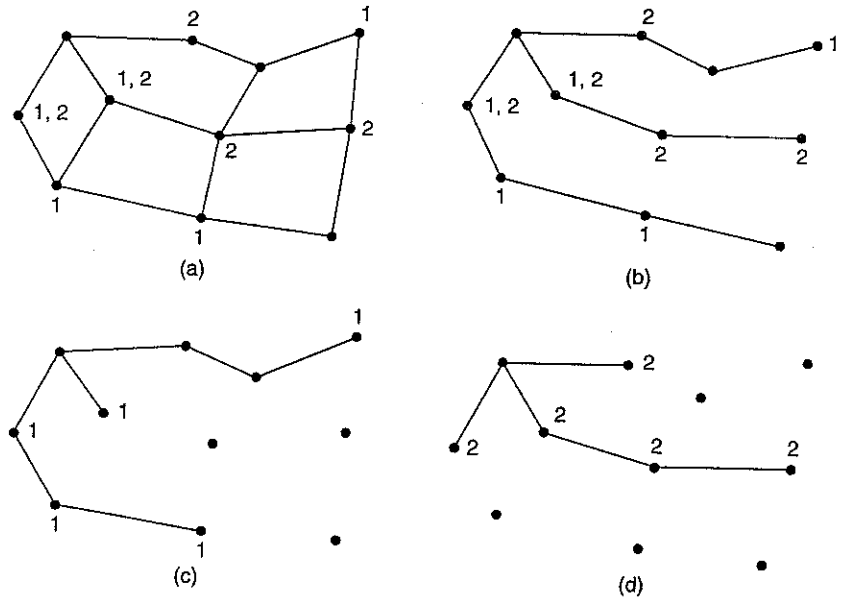
Néhány alkalmazásnál nagy távolságokra levő folyamatok működnek együtt meghatározott csoportosításban. Példa erre egy folyamatokból álló csoport, amely egy elosztott adatbázisrendszert valósít meg. Gyakran szükséges, hogy egy folyamat a csoport összes többi tagjának üzenetet küldjön. Ha a csoport kicsi, küldhet egy két pont közötti üzenetet. Ha a csoport nagy, ez a stratégia drága. Néha az adatszórás használható, de

az adatszórás használata arra, hogy 1000 gépet értesítsünk egy millió csomópontos hálózaton, nem hatékony, mert az elküldött üzenet a legtöbb vevőt nem érdekli (vagy ami még rosszabb, nagyon is érdekli, de nem szabad látniuk). Ezért szükségünk van egy olyan üzemmódra, amellyel jól meghatározott, számszerűleg nagy, de a teljes hálózathoz viszonyítva kis csoportoknak tudunk üzenetet küldeni.

Az ilyen csoportnak történő üzenetküldést **többesküldésnek (multicasting)** nevezzük, és az ehhez tartozó forgalomirányító algoritmust **többesküldéses forgalomirányításnak (multicast routing).** Ebben a részben leírunk egy módszert a többesküldés forgalomirányítására. További információkért lásd (Deering és Cheriton, 1990; Deering és mások, 1994; Rajagopalan, 1992).

A többesküldéshez a csoportok menedzselése is szükséges. Valami módon létre kell hozni csoportokat, majd ezeket meg kell szüntetni, a folyamatoknak csatlakozniuk kell a csoporthoz, és ki kell válniuk belőle. Hogy hogyan valósulnak meg ezek a feladatok, azzal a forgalomirányító algoritmus nem foglalkozik. Amivel viszont igen, az az, hogy amikor egy folyamat csatlakozik egy csoporthoz, tudatja ezt a tényt a hosztjával. Fontos, hogy a routerek tudják, mely hosztok mely csoportokhoz tartoznak. Vagy a hosztoknak kell a routereiket értesíteni a csoporttagságokban beállt változásokról, vagy a routereknek kell rendszeresen lekérdezni a hosztjaikat. Bárhogy is történik, a routerek megtudják, hogy mely hosztjuk mely csoportokba tartozik. A routerek értesítik a szomszédjaikat, így az információ tovaterjed az alhálózaton.

A többesküldés forgalomirányításához minden router kiszámít egy, az alhálózatban



5.21. ábra. *(a) Egy alhálózat. (b) Egy feszítőfa a legbaloldalibb routerhez. (c) Egy többesküldés-fa az 1. csoport számára. (d) Egy többesküldés-fa a 2. csoport számára*

levő összes többi routert lefedő feszítőfát. Például, az 5.21.(a) ábrán van egy alhálózat két csoporttal, 1-gyel és 2-vel. Néhány router olyan hosztokhoz csatlakozik, amelyek az egyik vagy mindkét csoportba tartoznak, ahogy az ábrán látszik. A legbaloldalibb router feszítőfája látható az 5.21.(b) ábrán.

Amikor egy folyamat egy többesküldéses csomagot küld egy csoportnak, az első router megvizsgálja a feszítőfáját és csonkolja, eltávolítván az összes vonalat, amely nem csoporttag hoszthoz vezet. Példánkban az 5.21.(c) ábrán látható az 1. csoport csonkolt feszítőfája. Hasonlóan az 5.21.(d) ábrán látható a 2. csoport csonkolt feszítőfája. A többesküldéses csomagokat csak a megfelelő feszítőfa mentén továbbítják.

Különféle módjai vannak a feszítőfa csonkolásának. A legegyszerűbbet akkor használhatjuk, amikor kapcsolatállapot alapú forgalomirányítást használunk, és minden router ismeri az alhálózat teljes topológiáját, beleértve azt is, hogy mely hosztok mely csoportokhoz tartoznak. Ekkor a feszítőfát minden út végénél kezdve, és a gyökér felé haladva csonkolhatjuk, eltávolítva mindazon routereket, amelyek nem tartoznak a kérdéses csoportba.

A távolságvektor alapú forgalomirányításnál más csonkolási stratégiát követhetünk. Az alapalgoritmus a visszairányú továbbítás. Viszont, amikor egy olyan router kap többesküldéses üzenetet, amelynek nincs abban a csoportban érdekelt hosztja és nincs más routerekhez kapcsolata, egy PRUNE üzenettel válaszol, utasítva a küldőt, hogy ennek a csoportnak szóló többesküldéseket ne küldjön többet. Amikor egy router, amelynek nincs csoporttag a hosztjai közt, minden vonalán ilyen üzeneteket kapott, ő is PRUNE üzenettel válaszolhat. Ily módon az alhálózat rekurzívan csonkolódik.

Ezen algoritmus egy lehetséges hátránya, hogy nagy hálózatokra nehezen méretezhető. Tegyük fel, hogy a hálózatban n csoport van, mind átlagosan m taggal. Minden csoporthoz m csonkolt feszítőfát kell tárolni, ez összesen mn fa. Amikor sok nagy csoport van, tekintélyes méretű tár kell a tárolásukhoz.

Egy alternatív terv **magalapú fákat (core-based trees)** használ (Ballardie és mások, 1993). Itt egy feszítőfát számolnak ki csoportonként, a gyökérrel (maggal) a csoport közepén. Hogy egy többesküldéses üzenetet küldjünk, a hosztnak a maghoz kell azt küldenie, amely aztán elvégzi a többesküldést a feszítőfa mentén. Bár ez a fa nem lesz optimális minden forrás számára, a tárolási költségek m fáról csoportonként egy fára csökkennek, ami jelentős megtakarítás.

5.3. Torlódásvédelmi algoritmusok

Amikor túl sok csomag van jelen az alhálózatban (vagy egy részében), a teljesítőképesség visszaesik. Ezt a helyzetet **torlódásnak (congestion)** nevezzük. Az 5.22. ábra festi le a tüneteket. Amikor a hosztok által az alhálózatba beadott csomagok száma a szállítási kapacitáson belül marad, akkor mind kézbesítésre kerül (kivéve egy párat, amely átvitel során hibákat szenvedett), és a kézbesítettek száma arányos az elküldöttek számával. Ahogy azonban a forgalom túl nagyra nő, a routerek már nem képesek ezt a forgalmat követni, és csomagokat kezdenek elveszteni. Ez általában tovább ront-

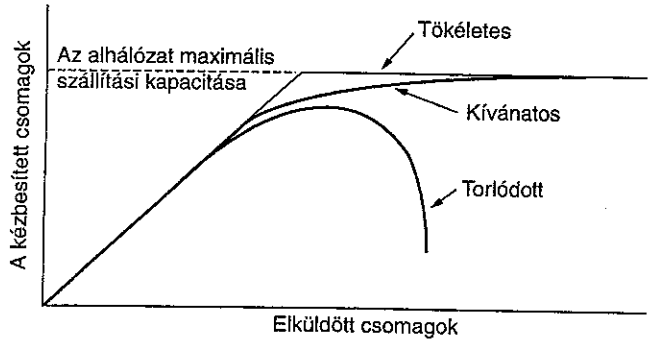
ja a helyzetet. Nagyon nagy forgalomnál a teljesítőképesség teljesen összeomlik, és szinte egy csomag sem kerül kézbesítésre.

A torlódást sok tényező okozhatja. Ha hirtelen nagy mennyiségű csomag kezd érkezni három vagy négy bejövő vonalon, és mindnek ugyanarra a kimenő vonalra van szüksége, egy várakozó sor fog felépülni. Ha a memória kevés ahhoz, hogy mindet befogadja, akkor csomagok fognak elveszni. Több memória hozzáadása egy adott pontig segíthet, de Nagle (1987) kimutatta, hogy ha a routereknek végtelen kapacitású memóriája van, a torlódás nem javul, hanem rosszabbá válik, mivel mire a csomagok a sor elejére kerülnek, az időzítésük már (többször is) lejárt, és így már másodpéldányok is továbbításra kerültek. A router mindezeket a csomagokat kötelességtudóan továbbítja a következő routerhez, megnövelvén a terhelést a rendeltetési helyig vezető út mentén.

A lassú processzorok is okozhatnak torlódást. Ha a routerek CPU-ja lassú a megkövetelt könyvelési feladatok (pufferek sorbaállítása, táblák frissítése stb.) elvégzéséhez, sorok képződhetnek, még ha van is fölös vonalkapacitás. Hasonlóan, a kis sávszélességű vonalak is okozhatnak torlódást. A vonalak kapacitásának növelése és a processzorok meghagyása, vagy a vonali kapacitás megtartása és a processzor megváltoztatása, gyakran javít a helyzeten, de sokszor csak a szűk keresztmetszetet helyezi át. A rendszer egy részét, de nem az egészét javítva, szintén sokszor csak a szűk keresztmetszetet helyezi máshova. A valódi probléma gyakran az, hogy a rendszer részei nem illenek össze. A probléma mindaddig megmarad, amíg a részek egyensúlyba nem kerülnek.

A torlódás hajlamos arra, hogy önmagát gerjessze és rosszabbodjon. Ha egy routernek nincs szabad puffere, az újonnan érkező csomagokat el kell dobja. Amikor egy csomag eldobásra kerül, az azt elküldő routernek (egy szomszédnak) lejárhat az időzítése, és a csomagot esetleg végtelen sokszor újraadhatja. Mivel nem dobhatja el a csomagot, amíg azt nem nyugtázták, a vevő oldali torlódás arra kényszeríti az adót, hogy ne szabadítsa fel a puffert, amit különben megtenne. Ily módon a torlódás visszafelé terjed, mint ahogy ez a fizetőkapu felé közeledő autóknál is tapasztalható.

Érdemes kifejezetten rámutatni a különbségre a torlódásvédelem és a forgalomszabályozás közt, mivel a kapcsolatuk nem magától értetődő. A torlódásvédelem azzal



5.22. ábra. *Amikor túl nagy forgalmat követelünk, bekövetkezik a torlódás, és a teljesítőképesség meredeken visszaesik*

foglalkozik, hogy az alhálózat képes legyen elszállítani a kért forgalmat. Ez egy átfogó, globális kérdés, amely magába foglalja minden hoszt, minden router viselkedését, a tárol-és-továbbít feldolgozást a routereken belül, és minden más tényezőt, amely hajlamos lerontani az alhálózat szállítási képességeit.

Ezzel szemben a forgalomszabályozás egy adott adó és egy adott vevő közti kétpontos forgalomra vonatkozik. Feladata megakadályozni, hogy egy gyors adó folyamatosan gyorsabban adjon, mint ahogy a vevő ezt fogadni tudja. A forgalomszabályozás majdnem mindig magába foglal egy közvetlen visszacsatolást a vevőtől az adóig, hogy megmondja az adónak, miképp állnak a dolgok a túlvégen.

Hogy lássuk a két fogalom közti különbséget, vegyünk egy 1000 gigabit/s kapacitású üvegszálas optikai hálózatot, amelyen egy szuperszámítógép próbál egy fájlt átvinni egy személyi számítógépre 1 Gb/s-mal. Bár nincsen torlódás (maga a hálózat nincs bajban), forgalomszabályozásra van szükség, hogy a szuperszámítógépet gyakori megállásra kényszerítsük, lélegzethez juttatva ezzel a személyi számítógépet.

A másik végletként vegyünk egy tárol-és-továbbít hálózatot 1 Mb/s-os vonalakkal és 1000 nagy számítógéppel, amelyek fele fájlokat próbál átvinni 100 kb/s-mal a másik feléhez. Itt a probléma nem az, hogy a gyors adók elnyomják a lassú vevőket, hanem egyszerűen az, hogy az összes kívánt forgalom meghaladja azt, amit a hálózat kezelni képes.

Az ok, amiért a torlódásvédelmet és a forgalomszabályozást gyakran összekeverik az, hogy néhány torlódásvédelmi algoritmus működés közben üzeneteket küld vissza bizonyos forrásoknak, utasítva őket, hogy lassítsanak, amikor a hálózat bajba kerül. Ezért egy hoszt kaphat „lassíts" üzenetet azért is, mert a vevő nem tudja kezelni a terhelést, vagy azért is, mert a hálózat nem bír vele. Erre a gondolatra később még visszatérünk.

A torlódásvédelem tanulmányozását először egy, a kezelésére szolgáló általános modell szemrevételével kezdjük. Ezután sokfajta megközelítést fogunk látni a torlódás megelőzésére, majd változatos dinamikus algoritmusokat, a torlódás bekövetkezése utáni helyzet lekezelésére.

5.3.1. A torlódásvédelem alapelvei

A komplex rendszerek, mint például a számítógép-hálózatok sok problémája megvizsgálható szabályozáselméleti szempontból. Ez a megközelítés a megoldásokat két csoportra osztja: nyílthurkú (open loop) és zárthurkú (closed loop) megoldásokra. A nyílthurkú megoldások a problémát jó tervezéssel kísérlik meg megoldani, más szavakkal, eleve nem engedik a torlódást kialakulni. Miután a rendszer életre kelt, nem végeznek futás közben korrekciókat.

A nyílthurkú szabályozás eszközei közé tartoznak azok a döntések, hogy mikor fogadunk el új forgalmat, mikor dobunk el csomagokat és melyeket, valamint az ütemezési döntések a hálózat különböző pontjain. Mindezekben közös, hogy a döntéseket a hálózat aktuális állapotától függetlenül hozzák.

Ezzel ellentétben, a zárthurkú rendszerek a visszacsatolt kör elvén alapulnak. Ennek a megközelítésnek három része van, ha torlódásvédelemre alkalmazzuk:

1. Figyelni a rendszert, hogy észrevegyük, hol és mikor következik be torlódás.

2. Továbbadni ezt az információt azokra a helyekre, ahol be lehet avatkozni.

3. Módosítani a rendszer működését, hogy helyrehozzuk a problémát.

Különböző mértékegységek alapján mérhetjük az alhálózatban a torlódást. Ezek közül a legfontosabbak a pufferhiány miatt eldobott csomagok százalékos aránya, az átlagos sorhosszak, a lejárt időzítésű és újraadott csomagok, az átlagos csomagkésleltetés, és a csomagkésleltetés szórása. Minden esetben a növekvő számok súlyosbodó torlódást jeleznek.

A visszacsatolt kör második lépése az információ továbbítása az észlelés helyétől oda, ahol valamit tudunk tenni ellene. Ennek a nyilvánvaló módja az, hogy a torlódást észlelő router egy csomagot küld a forgalom forrásához vagy forrásaihoz, bejelentve a problémát. Természetesen ezek a csomagok pont abban a pillanatban növelik a terhelést, amikor erre a legkevésbé van szükség, nevezetesen, amikor az alhálózatban amúgy is torlódás van.

Azonban léteznek más lehetőségek is. Például fenntarthatunk minden csomagban egy bitet vagy egy mezőt, amelyet a routerek kitölthetnek, amikor a torlódás valamilyen küszöbszint fölé emelkedik. Amikor egy router érzékeli ezt a torlódott állapotot, minden kimenő csomagban kitölti ezt a mezőt, hogy figyelmeztesse a szomszédokat a veszélyre.

Megint egy másik megközelítés az, hogy a hosztok vagy a routerek küldjenek ki rendszeresen próbacsomagokat, amelyek rákérdeznek a torlódásra. Ezt az információt arra lehet azután használni, hogy a problémás területeket megkerülve irányítsuk a forgalmat. Néhány rádióállomás helikoptert alkalmaz a városi forgalom figyelésére, abban a reményben, hogy a hallgatók az autóikkal a torlódások helyét elkerülik.

Minden visszacsatolást tartalmazó megoldás azon a reményen alapul, hogy a torlódás tudata ráveszi a hosztokat arra, hogy megfelelő lépéseket tegyenek a torlódás csökkentése érdekében. Hogy ez helyesen működjön, az időléptéket gondosan be kell állítani. Ha egy router ÁLLJ-t kiált, ahányszor csak két csomag érkezik egymás után, és mindig, amikor 20 μs-ig tétlen, INDULJ-t kiált, a rendszer vadul ingadozni fog és soha nem konvergál. Másfelől, ha a biztonság kedvéért 30 percet vár, mielőtt bármit is mondana, a torlódásvédelmi algoritmus túl lassan fog reagálni ahhoz, hogy bármilyen valódi haszna legyen. Hogy jól működjön, valamilyen átlagolás szükséges, de az időállandó helyes beállítása nem magától értetődő.

Sok torlódásvédelmi algoritmus ismeretes. Hogy értelmes módon lehessen ezeket rendszerezni, Yang és Reddy (1995) kifejlesztettek egy rendszertant a torlódásvédelmi algoritmusokhoz. Azzal kezdik, hogy az algoritmusokat nyílthurkúakra és zárthurkúakra bontják, ahogy fentebb leírtuk. A nyílthurkú algoritmusokat tovább osztják a forrásnál beavatkozókra és a célnál beavatkozókra. A zárthurkú algoritmusokat szintén két alkategóriára bontják: explicit visszajelzésesre és implicit visszajelzésesre. Az explicit visszajelzéses algoritmusokban csomagokat küldenek vissza a torlódás helyéről, hogy figyelmeztessék a forrást. Az implicit algoritmusokban a forrás kikövetkezteti a torlódás létezését helyi megfigyelésekből, mint amilyen a nyugták visszaérkezéséhez szükséges idő.

A torlódás megléte azt jelenti, hogy a terhelés (ideiglenesen) nagyobb, mint amit az erőforrások (a rendszer egy részében) kezelni képesek. Két megoldás juthat eszünkbe: megnövelni az erőforrásokat vagy lecsökkenteni a terhelést. Például az alhálózat elkezdhet telefonvonalakat használni, hogy ideiglenesen megnövelje a sávszélességet bizonyos pontok közt. Az olyan rendszerekben, mint az SMDS (l. 1. fejezet), kérhet a szolgáltatótól további sávszélességet egy időre. Műholdas rendszerekben az átviteli teljesítmény megnövelése gyakran nagyobb sávszélességet ad. A forgalom szétosztása több útvonal között mindig a legjobb (optimális) útvonal használata helyett szintén hatékonyan megnövelheti a sávszélességet. Végül, tartalék routerek is, amelyek rendszerint vésztartalékként szolgálnak (hogy a rendszert hibatűrővé tegyék), üzembe helyezhetők a nagyobb kapacitás érdekében, amikor súlyos torlódás lép fel.

Néha azonban egyáltalán nem lehet növelni a kapacitást, vagy azt már előzőleg a felső határig növeltük. Ekkor a torlódás leküzdésére az egyetlen mód az, hogy a terhelést csökkentjük. Sok módszer létezik a terhelés visszafogására, mint például megtagadni a szolgáltatást néhány felhasználótól, rontani a néhány vagy az összes felhasználónak nyújtott szolgáltatás színvonalát, továbbá a felhasználókat kényszeríteni arra, hogy igényeiket egy jobban előre látható módon ütemezzék.

Ezen eljárások közül néhány legjobban virtuális áramkörökre alkalmazható. Olyan alhálózatokra, amelyek belsőleg virtuális áramköröket használnak, ezek az eljárások a hálózati rétegben használhatók. Datagram alapú alhálózatoknál is használhatjuk ezeket még a szállítási réteg összeköttetéseiben. Ebben a fejezetben a hálózati rétegbeli alkalmazásaikra összpontosítunk. A következő fejezetben majd meglátjuk, mit tehetünk a szállítási rétegbeli torlódások kezelése érdekében.

5.3.2. Torlódásmegelőző módszerek

A torlódásvédelmi eljárások vizsgálatát kezdjük a nyílt hurkú rendszerekkel. Ezeket a rendszereket úgy tervezték, hogy eleve megpróbálják minimalizálni a torlódást, ahelyett, hogy kialakulása után reagálnának arra. Ezt a célt az egyes szinteken megfelelő politikák használatával próbálják elérni. Az 5.23. ábrán olyan adatkapcsolati, hálózati és szállítási politikákat láthatunk, amelyek befolyásolhatják a torlódást (Jain,1990).

Kezdjük az adatkapcsolati rétegnél és haladjunk fölfelé. Az újraadási politika azt határozza meg, hogy az adónak milyen gyorsan jár le az időzítése, és mit ad ekkor. Egy türelmetlen adó, amelynek gyorsan lejár az időzítése, és újraadja az összes elintézetlen keretet a visszalépés *n*-nel protokollt használva, nagyobb terhelést idéz elő, mint egy türelmes adó, amely a szelektív ismétlést használja. Ehhez kapcsolódik a tárolási politika. Ha a vevők automatikusan eldobják a sorrenden kívül érkező kereteket, akkor ezeket később újra kell adni, ami pluszterhelést képez.

A nyugtázási politika is befolyásolja a torlódást. Ha azonnal nyugtázunk minden keretet, a nyugtakeretek külön forgalmat jelentenek. Ha viszont a nyugtákat eltesszük, hogy a visszairányú forgalomra ültethessük rá azokat, emiatt külön lejárhatnak időzítők, és újraadhatják a keretet. Egy szoros forgalomszabályozási eljárás (pl. kisméretű csúszóablak) lecsökkenti az adatfolyam sebességét és segít elkerülni a torlódást.

Réteg	Politikák
Szállítási	• Újraadási politika • Sorrenden kívül érkezett csomagok tárolási politikája • Nyugtázási politika • Forgalomszabályozási politika • Időzítés meghatározása
Hálózati	• Az alhálózaton belül virtuális áramkörök vagy datagramok • Csomag-sorbaállítási és kiszolgálási politika • Forgalomirányító algoritmus • Csomagélettartam menedzselés
Adatkapcsolati	• Újraadási politika • Sorrenden kívül érkezett csomagok tárolási politikája • Nyugtázási politika • Forgalomszabályozási politika

5.23. ábra. *A torlódást befolyásoló politikák*

A hálózati rétegben a virtuális áramkörök és a datagramok közti választás szintén befolyásolja a torlódást, mivel számos torlódásvédelmi algoritmus csak virtuális áramkör alapú alhálózattal működik. A csomag-sorbaállítási és -kiszolgálási politika attól függ, hogy a routereknek bemenő vonalanként van-e egy soruk, kimenő vonalanként van-e egy soruk, vagy bemenő és kimenő soruk is van. Attól is függ, hogy milyen a csomagfeldolgozási sorrend (prioritásos vagy körforgó). A csomageldobási politika dönti el, mely csomagokat kell eldobni, ha nincs már hely. Egy jó politika enyhíthet a torlódáson, egy rossz ronthat rajta.

A forgalomirányítási algoritmus segíthet elkerülni a torlódást, ha az összes vonalra elosztja a forgalmat, ellenben ha rossz, túl sokat küldhet olyan vonalakon, ahol már amúgy is torlódás van. Végül a csomagélettartam menedzselés azt mondja meg, meddig élhet egy csomag, mielőtt eldobnák. Ha ez túl hosszú, az elveszett csomagok sokáig akadályt jelentenek, míg ha túl rövid, lejárhat a csomagok ideje, mielőtt céljukat elérnék, ami újraadásokat eredményezhet.

A szállítási rétegben ugyanazok a kérdések, mint az adatkapcsolati rétegben, hozzávéve, hogy az időzítési intervallum nehezebben határozható meg, mivel az átviteli idő egy hálózaton kevésbé jósolható, mint az átviteli idő egy vezetéken két router közt. Ha túl rövid az időzítés, felesleges csomagok átvitelére kerülhet sor. Ha túl hosszú, a torlódás csökken, de ennek csomagvesztés esetén a válaszidő látja kárát.

5.3.3. Forgalomformálás

A torlódás egyik fő oka, hogy a forgalom gyakran lökésszerű. Ha a hosztokat rá lehetne venni arra, hogy egyenletesen adjanak, kevésbé fordulna elő torlódás. Egy másik nyílthurkú eljárás arra irányul, hogy a torlódás kezelésére a csomagok előre megjósolhatóbb sebességgel kerüljenek továbbításra. A torlódásvédelemnek ezen megközelítése elterjedt ATM hálózatokban, és **forgalomformálásnak (traffic shaping)** hívják.

A forgalomformálás az adatátvitel átlagos *sebességének* (és lökéseinek) szabályozásával foglalkozik. Ezzel ellentétben a korábban tanulmányozott csúszóablakos protokollok csak az egyszerre úton levő adat mennyiségét határolják be, a sebességét nem. Amikor egy virtuális áramkör felépül, erre vonatkozólag a felhasználó és az alhálózat (vagyis az ügyfél és a szolgáltató) megegyeznek egy, erre az áramkörre vonatkozó forgalommintában (formában). Amíg az ügyfél betartja a megállapodás rá eső részét, és csak a szerződés szerint küldi a csomagokat, addig a szolgáltató ígérete szerint időben kézbesíti azokat. A forgalomalakítás csökkenti a torlódást és hozzásegíti a szolgáltatót, hogy betarthassa az ígéretét. Az ilyen megállapodások fájlátvitelnél nem túl lényegesek, annál fontosabbak viszont valós idejű adatoknál, mint pl. a hang- és képkapcsolatnál, amelyek nem jól tűrik a torlódást.

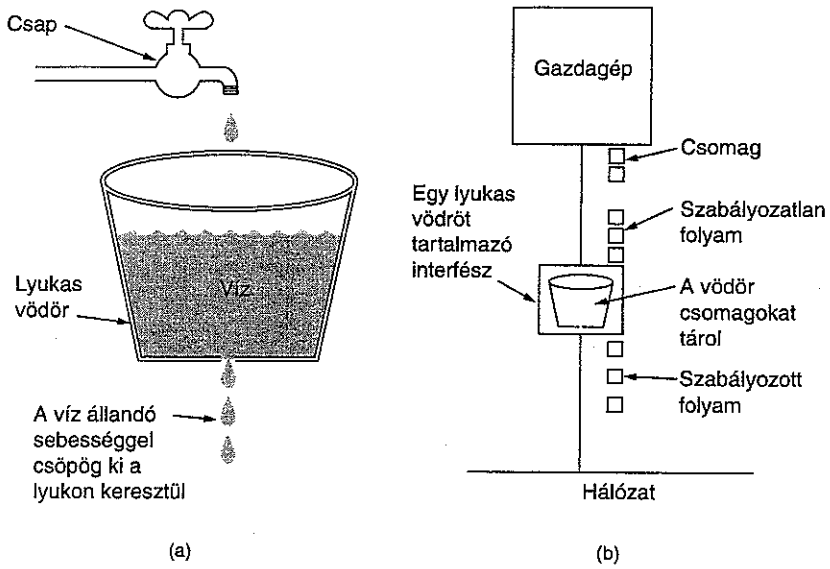
Gyakorlatilag a forgalomalakításnál így szól az ügyfél a szolgáltatóhoz: „Ilyen az átviteli mintám. Képes ezt kezelni?" Ha a szolgáltató beleegyezik, felmerül a kérdés, honnan tudja a szolgáltató, hogy az ügyfél betartja-e a szerződést, illetve mit tegyen, ha nem tartja be. A forgalom folyamának, haladásának figyelését **forgalmi politikának (traffic policing)** nevezzük. Egy forgalomalakban történő megegyezés és utána ennek felügyelete könnyebb virtuális áramkör alapú alhálózatokkal, mint datagram alapú alhálózatokkal. De ugyanezek az ötletek még datagram alapú alhálózatoknál is alkalmazhatók a szállítási rétegbeli összeköttetésekre.

A lyukas vödör algoritmus

Képzeljünk el egy vödröt, az alján egy kis lyukkal, ahogy az 5.24.(a) ábrán látható. Mindegy, hogy a víz milyen sebességgel érkezik a vödörbe, a kimenő folyam konstans ρ sebességű, amikor van víz a vödörben, és nulla, amikor a vödör üres. Ha a vödör megtelt, a további érkező víz kicsordul a vödörből és elveszik (vagyis nem jelenik meg a lyuk alatti kimeneti folyamban).

Ugyanez az ötlet alkalmazható csomagokra is, ahogy az 5.24.(b) ábra mutatja. A felfogás szerint minden hoszt egy lyukas vödröt, vagyis egy véges belső sort tartalmazó interfészen keresztül kapcsolódik a hálózathoz. Ha egy csomag akkor érkezik a sorba, amikor az tele van, a csomagot eldobják. Más szavakkal, amikor egy vagy több folyamat a hoszton belül akkor próbál csomagot küldeni, amikor már a sor betelt, az új csomagot minden felhajtás nélkül eldobják. Ez az elrendezés beépíthető a hardver interfészbe vagy szimulálható a hoszt operációs rendszere által. Ezt először Turner (1986) javasolta, és **lyukas vödör (leaky bucket) algoritmusnak** hívják. A gyakorlatban ez nem más, mint egy egykiszolgálós sorban állási rendszer állandó kiszolgálási idővel.

A hoszt óraütésenként egy csomagot tehet a hálózatra. Ezt megint csak az interfészkártya vagy az operációs rendszer kényszerítheti ki. Ez a mechanizmus a hoszton belüli felhasználói folyamatoktól eredő egyenetlen csomagfolyamot egyenletes csomagfolyamként adja a hálózatra, kisimítva a lökéseket és nagyban csökkentve a torlódás esélyét.



5.24. ábra. *(a) Egy lyukas vödör vízzel. (b) Egy lyukas vödör csomagokkal*

Amikor a csomagok mind azonos méretűek (pl. ATM cellák), akkor az algoritmust a leírt formájában használhatjuk. Viszont amikor változó méretű csomagokat használunk, gyakran jobb óraütésenként egy rögzített számú bájtot megengedni, mint egy csomagot. Így ha a szabály 1024 bájt/óraütés, egyetlen 1024 bájtos csomagot lehet elfogadni óraütésenként, két 512 bájtos csomagot, négy 256 bájtos csomagot és így tovább. Ha a fennmaradó bájtszám túl kicsi, a csomagnak meg kell várnia a következő óraütést.

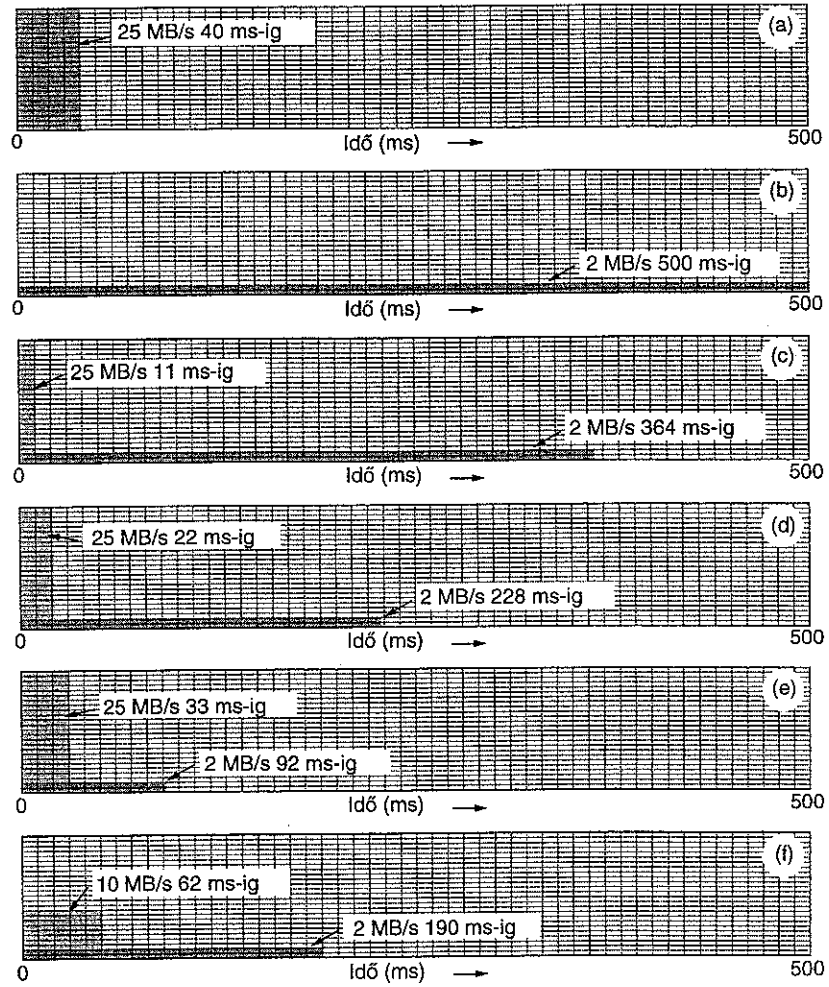
Az eredeti lyukas vödör algoritmust könnyű megvalósítani. A lyukas vödör egy véges sorból áll. Amikor egy csomag érkezik, ha van hely a sorban, hozzáillesztik; egyébként eldobják. Minden óraütéskor egy csomagot viszünk át (hacsak nem üres a sor).

A bájtszámolós lyukas vödröt majdnem ugyanígy valósítjuk meg. Minden óraütéskor a számlálót n-re állítjuk. Ha a sorban első csomagnak kevesebb bájtja van, mint a számláló jelenlegi értéke, átvisszük, és a számlálót a bájtok számával csökkentjük. További csomagokat is küldhetünk, amíg a számláló értéke elég nagy. Amikor a számláló a sorban következő csomag hossza alá csökken, az átvitel megáll a következő óraütésig, amikor is felülírjuk és elveszítjük a fennmaradó bájtszámot.

A lyukas vödörre egy példaként képzeljük el, hogy egy számítógép 25 millió bájt/s (200 Mb/s) sebességgel tud adatot termelni, és a hálózat is ezen a sebességen működik, ellenben a routerek ezt az adatsebességet csak rövid időn keresztül tudják kezelni. Hosszabb időn át legjobban 2 millió bájt/s-ot meg nem haladó sebességgel tudnak működni. Tegyük fel, hogy az adat 1 millió bájtos lökésekben érkezik, vagyis minden másodpercben egy 40 ms-os lökés. Hogy az átlagos sebességet lecsökkentsük 2 millió

bájt/s-ra, használhatunk egy lyukas vödröt ρ = 2 MB/s értékkel és C = 1 MB-os kapacitással. Ez azt jelenti, hogy a legfeljebb 1 MB-os lökéseket adatvesztés nélkül tudjuk kezelni, és az ilyen lökések mindig 500 ms-ra terülnek szét, mindegy, milyen gyorsan érkeztek.

Az 5.25.(a) ábrán látjuk a lyukas vödör bemenetét 25 MB/s-on 40 ms-ig. Az 5.25.(b) ábrán látjuk a kimenetet egyenletes 2 MB/s-mal áramlani 500 ms-ig.



5.25. ábra. *(a) Egy lyukas vödör bemenete. (b) Egy lyukas vödör kimenete. (c)–(e) Egy vezérjeles, 250 KB, 500 KB és 750 KB kapacitású vödör kimenete. (f) Egy 500 KB-os vezérjeles vödör kimenete, amely egy 10 MB/s-os lyukas vödröt táplál*

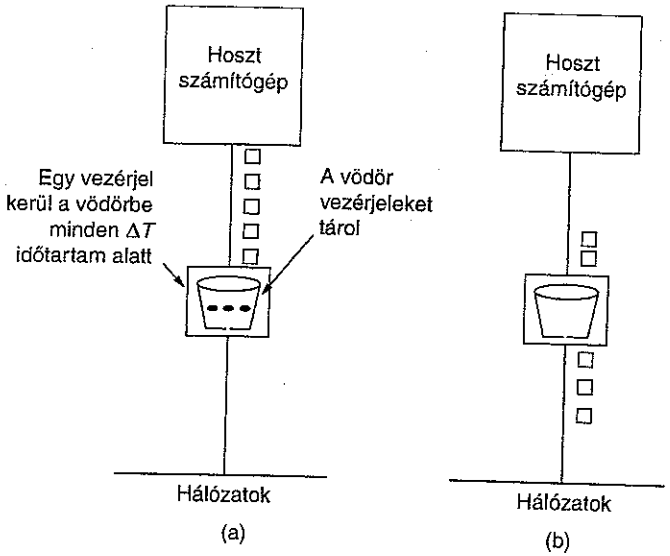
A vezérjeles vödör algoritmus

A lyukas vödör algoritmus merev kimeneti mintát kényszerít az átlagos sebességre, a forgalom lökéseitől függetlenül. Sok alkalmazásnak jobb, ha a kimenetet engedjük gyorsulni valamelyest, amikor nagy löketek érkeznek, így egy rugalmasabb algoritmusra van szükség, lehetőség szerint olyanra, amely soha nem veszt adatot. Egy ilyen algoritmus **a vezérjeles vödör (token bucket) algoritmus.** Ebben az algoritmusban a lyukas vödör vezérjeleket tartalmaz, amelyeket egy óra állít elő egy vezérjel/ΔT sebességgel. Az 5.26.(a) ábrán egy vödröt láthatunk, amelyben három vezérjel van, és öt csomag vár a továbbításra. Hogy egy csomag továbbítható legyen, el kell fognia és megsemmisítenie egy vezérjelet. Az 5.26.(b) ábrán láthatjuk, hogy három csomag keresztüljutott az ötből, de a másik kettő megrekedt, két további vezérjel előállítására várva.

A vezérjeles vödör algoritmus másfajta forgalomalakítást biztosít, mint a lyukas vödör algoritmusa. A lyukas vödör algoritmus nem engedi a tétlen hosztoknak, hogy az engedélyeket félretegyék, és később nagy löketeket küldjenek. A vezérjeles vödör algoritmus megengedi ezt a félretevést, a vödör maximális méretéig, n-ig. Ez a tulajdonság azt jelenti, hogy akár n csomagból álló lökések is küldhetők egyszerre, amely megenged némi lökésszerűséget a kimeneten és gyorsabb választ ad a bemenet hirtelen lökéseire.

Egy másik különbség a két algoritmus közt az, hogy a vezérjeles vödör algoritmus eldobja a vezérjeleket, amikor a vödör megtelik, de soha nem dob el csomagokat. Ezzel ellentétben a lyukas vödör algoritmus eldobja a csomagokat, ha a vödör megtelik.

Itt is lehetséges egy apróbb változtatás, ahol egy vezérjel nem egy csomagot, ha-



5.26. ábra. *A vezérjeles vödör algoritmusa. (a) Előtte. (b) Utána*

nem k bájtot enged küldeni. Egy csomagot csak akkor vihetünk át, ha elég vezérjel van arra, hogy lefedje a hosszát bájtokban. A töredék vezérjelek megmaradnak későbbi felhasználásra.

A lyukas vödör és vezérjeles vödör algoritmusok a routerek közti forgalom kisimítására is használhatóak, a hosztkimenetek példánkban szereplő szabályozásán kívül, de van egy különbség: egy hosztot szabályozó vezérjeles vödör leállíthatja a hoszt adását, amikor azt a szabályok szerint le kell állítani. Ha egy routert arra utasítunk, hogy álljon le, miközben a bemeneten továbbra is jön az adat, ez adatvesztést eredményezhet.

A vezérjeles vödör alapvető algoritmusának megvalósítása csak egy változó, amely a vezérjeleket számlálja. A számlálót eggyel növeljük ΔT időközönként, és eggyel csökkentjük, amikor csomagot küldünk. Amikor a számláló eléri a nullát, nem küldhetünk csomagot. A bájtszámlálásos változatban a számlálót k bájttal növeljük minden ΔT-ben, és mindegyik elküldött csomag hosszát kivonjuk belőle.

Alapjában véve, a vezérjeles vödör megenged lökéseket, de csak egy szabályozott legnagyobb hosszig. Nézzük például az 5.25.(c) ábrát. Itt van egy vezérjeles vödrünk 250 KB-os kapacitással. A vezérjelek egy 2 MB/s-ot megengedő sebességgel érkeznek. Feltéve, hogy a vezérjeles vödör tele van, amikor az 1 MB-os löket megérkezik, a vödör körülbelül 11 ms-ig tud a teljes 25 MB/s sebességgel adni. Ezután vissza kell állnia a 2 MB/s-ra, amíg a teljes bemeneti löketet el nem küldte.

A maximális sebességű löket hosszának kiszámítása kissé trükkös. Nem egyszerűen 1 MB osztva 25 MB/s, mert miközben kiadjuk a lökést, további vezérjelek érkeznek. Ha a lökethosszat S s-nak, a vezérjeleket tartalmazó vödör kapacitását C bájtnak, a vezérjelek érkezési sebességét ρ bájt/s-nak, és a legnagyobb kimeneti sebességet M bájt/s-nak vesszük, láthatjuk, hogy a kimeneti löket legfeljebb $C+\rho S$ bájtot tartalmaz. Azt is tudjuk, hogy a bájtok száma egy S s hosszú, maximális sebességű löketben MS. Így kapjuk a következőt:

$$C + \rho S = MS$$

Ezt az egyenletet megoldva kapjuk az $S = C/(M-\rho)$ eredményt. A mi paramétereinkre, ahol C=250 KB, M=25 MB/s, és ρ=2 MB/s, 11 ms körüli löketidőt kapunk. Az 5.25.(d) és (e) ábrák mutatják a vezérjeleket tartalmazó vödröt sorrendben 500 KB és 750 KB kapacitással.

Egy lehetséges probléma a vezérjeles vödör algoritmusával, hogy ismét megenged nagy löketeket, még ha a maximális löket időtartama szabályozható is ρ és M figyelmes megválasztásával. Gyakran kívánatos csökkenteni a csúcssebességet, de a lyukas vödör alacsony értékéhez való visszatérés nélkül.

A simább forgalom előállításának egy módja, hogy egy lyukas vödröt teszünk a vezérjeles vödör után. A lyukas vödör sebességének nagyobbnak kell lennie a vezérjeles vödör ρ-jánál, de kisebbnek a hálózat maximális sebességénél. Az 5.25(f) ábra egy 500 KB-os vezérjeles vödröt követő 10 MB/s-os lyukas vödör kimenetét mutatja.

Mindezekhez a sémákhoz politikát találni kissé trükkös lehet. A hálózatnak szimulálnia kell az algoritmust és meg kell arról győződnie, hogy nem küldenek több bájtot vagy csomagot, mint amennyi megengedett. A fölös csomagokat eldobják vagy visszaminősítik, mint ahogy azt később tárgyaljuk.

5.3.4. Folyammeghatározások

A forgalomalakítás akkor a leghatékonyabb, amikor mind az adó, mind a vevő és az alhálózat is megegyeznek egymással. Hogy megegyezést érjenek el, a fogalmi minta precíz meghatározása szükséges. Egy ilyen megegyezést **folyammeghatározásnak (flow specification)** nevezünk. Ez egy adatstruktúrából áll, amely mind a beadott forgalom mintáját, mind az alkalmazások által kívánt szolgálat minőségét leírja. Egy folyammeghatározást alkalmazni lehet mind a virtuális áramkörön küldött csomagokra, mind a datagramok egy forrás és egy cél (vagy akár több cél) közt küldött sorozatára.

Ebben a részben egy Partridge (1992) által tervezett példa folyammeghatározást írunk le, amit az 5.27. ábra mutat. Az ötlet az, hogy mielőtt egy virtuális áramköri összeköttetés felépülne, vagy egy sorozat datagramot elküldenénk, a forrás átadja a folyammeghatározást az alhálózatnak jóváhagyásra. Az alhálózat ezt elfogadhatja, visszautasíthatja, vagy előállhat egy ellenjavaslattal („Nem tudok 100 ms-os átlagos késleltetést biztosítani; megtenné 150 ms is?"). Amikor az adó és az alhálózat már megegyeztek, az adó megkérdezheti a vevőt, hogy ő is beleegyezik-e.

A bemenet jellemzői	A kívánt szolgálat
Maximális csomagméret (bájt)	Érzékenység a csomagvesztésre (bájt)
Vezérjeles vödör sebessége (bájt/s)	Csomagvesztési időköz (μs)
Vezérjeles vödör mérete (bájt)	Érzékenység a löketvesztésre
Maximális átviteli sebesség (bájt/s)	Minimális észlelt késleltetés (μs)
	Maximális késleltetésingadozás (μs)
	Garancia minősége

5.27. ábra. *Egy példa a folyammeghatározásra*

Vizsgáljuk most meg a példa folyammeghatározásunk paramétereit. Kezdjük a forgalom specifikációjával. A *Maximális csomagméret* megmondja, milyen nagyok lehetnek a csomagok. A következő két paraméter hallgatólagosan feltételezi, hogy a forgalmat a vezérjeles vödör bájtszámláló változata formálja. Ezek azt mondják meg, hogy hány bájt érkezik a vödörbe másodpercenként, és milyen nagy a vödör. Ha a sebesség r bájt/s, és a vödör mérete b bájt, akkor bármely tetszőleges ΔT időintervallumban a maximálisan elküldhető bájtok száma $b + r\Delta T$. Itt az első kifejezés a vödör az időintervallum kezdetén lehetséges legnagyobb tartalmát fejezi ki, a második pedig az intervallum alatt beérkező új vezérjeleket. A *Maximális átviteli sebesség* az a legnagyobb sebesség, amit a hoszt bármilyen körülmények közt nyújtani tud, és hallgatólagosan meghatározza a legrövidebb időintervallumot, amely alatt a vezérjeleket tartalmazó vödör kiürülhet.

A második oszlop azt határozza meg, hogy az alkalmazás mit akar az alhálózattól. Az első és második paraméter a számlálóját és a nevezőjét határozzák meg annak a törtnek, amely a maximális, még elfogadható elvesztési arányt adja meg (pl. 1 bájt óránként). Alternatívaként azt is jelezhetik, hogy a folyam érzéketlen a csomagvesztésre. Az *Érzékenység egy rövid sorozat csomag elvesztésére* azt mondja meg, hogy hány egymást követő elveszett csomagot tűrünk el.

A két következő szolgáltatás-paraméter a késleltetésre vonatkozik. A *Minimális észlelt késleltetés* azt mondja meg, milyen hosszan tarthatjuk fel a csomagot anélkül, hogy az alkalmazás azt észrevenné. Egy fájlátvitelnél ez egy másodperc is lehet, de egy audiofolyamnál 3 ms lehet a felső határ. A *Maximális késleltetésingadozás* azt a tényt próbálja meg számszerűsíteni, hogy néhány alkalmazás nem érzékeny a tulajdonképpeni késleltetésre, de igen érzékeny a **dzsitterre (jitter)**, vagyis, a vég-vég közötti csomagátviteli idők eltérésének mértékére. Ez kétszer annyi mikroszekundum, amennyit egy csomag késleltetése eltérhet az átlagostól. Ezért egy 2000-es érték azt jelzi, hogy egy csomag 1 ms-mal előbb vagy később érkezhet, de többel nem.

Végül a *Garancia minősége* paraméter azt jelzi, hogy az alkalmazás mennyire komolyan gondolja a fenti paraméterek teljesítését. Egyfelől az elvesztési és késleltetési jellemzők ideális célok lehetnek, de semmi kárt nem okoz, ha nem teljesülnek. Másfelől viszont olyan fontosak is lehetnek, hogy ha nem tudjuk ezeket elérni, az alkalmazás egyszerűen befejezi futását, terminálódik. E két véglet közötti esetek is lehetségesek.

Bár a folyammeghatározást úgy tekintettük, mint egy alkalmazástól jövő kérést az alhálózat felé, ez azonban visszatérő érték is lehet, amely megmondja, mire képes az alhálózat. Ezért a folyammeghatározás potenciálisan használható a szolgálat szintjéről történő kiterjesztett egyezkedésre is.

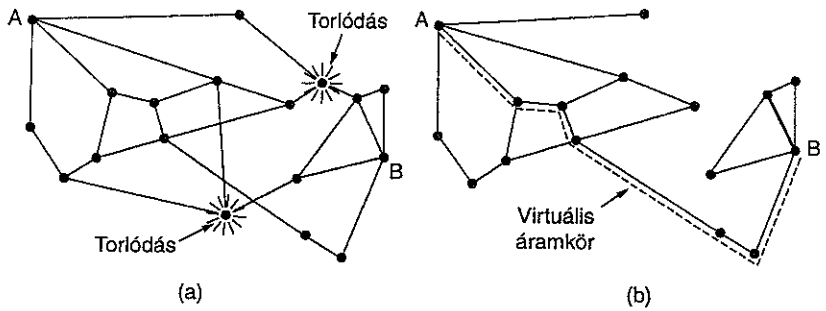
Minden folyammeghatározásnak veleszületett problémája az, hogy az alkalmazás nem biztos, hogy tudja, mit is akar valójában. Például egy alkalmazói program, amely New Yorkban fut, elégedett lehet Sydney felé egy 200 ms-os késleltetéssel, de igencsak elégedetlen ugyanezzel a 200 ms-os késleltetéssel Boston felé. Itt a „minimális szolgálat" nyilván annak a függvénye, hogy mit tartunk lehetségesnek.

5.3.5. Torlódásvédelem virtuális áramkör alapú alhálózatokban

A fentebb leírt torlódásvédelmi eljárások alapvetően nyílthurkúak: inkább eleve megpróbálják a torlódás kialakulását megelőzni, minthogy az esemény bekövetkezte után foglalkozzanak vele. Ebben a szakaszban néhány megközelítést írunk le a torlódás dinamikus szabályozására virtuális áramkör alapú alhálózatokban. A következő kettőben olyan módszereket fogunk látni, amelyek bármely alhálózatban alkalmazhatóak.

Egy módszer, amelyet széles körben használnak már kialakult torlódás további romlásának meggátlására, a **belépés ellenőrzése (admission control).** Az ötlet egyszerű: mikor már jelezték a torlódást, nem építünk fel több virtuális áramkört, amíg a probléma meg nem szűnik. Ezért az új szállítási rétegbeli összeköttetések felépítésére irányuló kísérletek sikertelenek lesznek. Még több ember beeresztése csak ront a helyzeten. Bár ez a megközelítés elég durva, egyszerű és könnyű kivitelezni. A telefonrendszerben, amikor egy kapcsolót túlterhelnek, az szintén belépésellenőrzést hajt végre azáltal, hogy nem ad tárcsahangot.

Egy alternatív megközelítés, hogy engedjünk meg új virtuális áramköröket, de figyelmesen úgy irányítsuk ezeket, hogy a problémás területeket elkerüljük. Példának vegyük az 5.28.(a) ábra alhálózatát, ahol két routerben van torlódás, ahogy jeleztük.



5.28. ábra. *(a) Egy torlódott alhálózat. (b) Egy újrarajzolt alhálózat, amely kiküszöböli a torlódást, és egy virtuális áramkör A-tól B-ig*

Tegyük fel, hogy egy, az *A* routerhez kapcsolódó hoszt egy, a *B* routerhez kapcsolódó hoszttal akar összeköttetést létrehozni. Rendesen ez az összeköttetés áthaladna az egyik torlódott routeren. Hogy ezt a helyzetet elkerüljük, újrarajzolhatjuk az alhálózatot, ahogy az 5.28.(b) ábrán látszik, kihagyva a torlódott routereket és a hozzájuk kapcsolódó vonalakat. A szaggatott vonal egy lehetséges útvonalat mutat egy, a torlódott routereket elkerülő virtuális áramkör számára.

Egy másik, virtuális áramkörökhöz kapcsolódó stratégia egy megegyezés kezdeményezése a virtuális áramkör felépülésekor a hoszt és az alhálózat közt. Ez a megegyezés rendszerint meghatározza a forgalom nagyságát, a forgalom alakját, a kívánt szolgálat minőségét, és más paramétereket. Hogy betarthassa a megegyezés rá eső részét, az alhálózat jellemzően erőforrásokat foglal le az út mentén, amikor a virtuális áramkör felépül. Ezek az erőforrások tartalmazhatják a tábla- és pufferterületet a routerekben és sávszélességet a vonalakon. Így valószínűtlen, hogy torlódás következzen be az új virtuális áramkörökön, mivel a szükséges erőforrások garantáltan rendelkezésre állnak.

Ezt a lefoglalást szabványos működési eljárásként mindig megtehetjük, vagy csak akkor, amikor az alhálózatban torlódás van. Ha mindig megtesszük, hátránya, hogy hajlamos az erőforrások elpazarlására. Ha hat virtuális áramkör, amelyek mindegyike 1 Mb/s-ot használhat, ugyanazon 6 Mb/s-os vonalon keresztül halad, a vonalat megteltnek kell jeleznünk, még akkor is, ha ritkán forgalmaz mind a hat virtuális áramkör egyszerre. Következésképpen, a torlódásvédelem ára a kihasználatlan sávszélesség.

5.3.6. Lefojtó csomagok

Térjünk most át egy megközelítésre, amely mind virtuális áramkör, mind datagram alapú alhálózatokban használható. Minden router könnyen megfigyelheti a kimeneti vonalai és egyéb erőforrásai kihasználtságait. Például hozzárendelhet minden vonalhoz egy valós változót, u-t, amelynek 0 és 1 közti értéke az utóbbi időben annak a vonalnak a kihasználtságát jelzi. Hogy u-t jól tudjuk becsülni, a pillanatnyi vonalkihasználtságból, f-ből periodikusan (0 vagy 1) mintákat vehetünk, és u-t a következő képlet szerint frissíthetjük:

$$u_{\text{új}} = au_{\text{régi}} + (1 - a)f$$

ahol az a konstans azt határozza meg, milyen gyorsan felejti el a router a közelmúlt történéseit.

Mindig, amikor u a küszöbszint fölé emelkedik, a kimeneti vonal „figyelmeztető" állapotba kerül. Minden újonnan érkező csomagot ellenőrzünk, hogy a kimeneti vonala figyelmeztető állapotban van-e. Ha igen, a router visszaküld egy **lefojtó csomagot (choke packet)** a forráshoszthoz, megadva a csomagban talált rendeltetési címet. Az eredeti csomagot megjelöljük (bebillentünk egy fejrészbitet), hogy ne hozzon létre újabb lefojtó csomagokat útja során, és azután a szokásos módon továbbítjuk.

Amikor a forráshoszt megkapja a lefojtó csomagot, köteles a megjelölt rendeltetési hely felé küldött forgalmát X százalékkal csökkenteni. Mivel valószínűleg más csomagok is útban vannak ugyanazon rendeltetési hely felé, és ezek további lefojtó csomagokat fognak létrehozni, a hosztnak figyelmen kívül kell hagynia az arra a rendeltetési helyre vonatkozó lefojtó csomagokat egy előre meghatározott ideig. Miután ez az idő lejárt, a hoszt figyeli a további lefojtó csomagokat egy másik időtartamig. Ha érkezik egy ilyen csomag, akkor a vonalon még mindig torlódás van, ezért a hoszt még jobban lecsökkenti az adott rendeltetési helyre irányuló forgalmát és megint kezdi figyelmen kívül hagyni a további lefojtó csomagokat. Ha a figyelési időn belül nem érkezik lefojtó csomag, a hoszt újra megnövelheti a forgalmat. Az ebben a protokollban meglevő visszacsatolás segíthet megelőzni a torlódást, mégsem fojtja le az adatáramlást egyik irányban sem, hacsak nincs valamilyen baj.

A hosztok a forgalmat a paramétereik, például az ablakméret vagy a lyukas vödör kimeneti sebességének megváltoztatásával csökkenthetik. Az első lefojtó csomag hatására az adatsebesség jellemzően a felére csökken le, a következő hatására a negyedére és így tovább. A növekedés kisebb lépésekben történik, hogy a torlódás ne alakuljon ki gyorsan újra.

Ennek a torlódásvédelmi algoritmusnak számos változatára van javaslat. Egy szerint a routerek számos küszöbszintet tarthatnának karban. Attól függően, hogy melyik küszöbszintet léptük át, a lefojtó csomag tartalmazhat egy enyhe figyelmeztetést, egy szigorú figyelmeztetést vagy egy ultimátumot.

Egy másik változat, hogy a sorhosszakat vagy a pufferkihasználtságot használjuk indítójelnek a vonalkihasználtság helyett. Természetesen ugyanaz az exponenciális súlyozás használható ezzel a mértékkel is, mint az u-val.

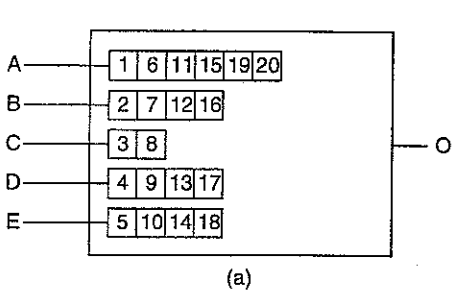
Súlyozott egyenlő esélyű sorba állítás

A lefojtó csomagok használatával az egyik probléma, hogy a forráshosztok által elvégzendő cselekedet önkéntes. Tegyük fel, hogy egy routert négy forrásból származó csomagok árasztanak el, és mindnek lefojtó csomagokat küld. Az egyikük visszavesz a sebességéből, ahogy kell, de a másik három erőlteti a dolgot tovább. Az eredmény az lesz, hogy a becsületes hoszt még kisebb részt kap a sávszélességből, mint annak előtte.

Hogy megkerüljük ezt a problémát, és ezáltal az alkalmazkodást vonzóbbá tegyük, Nagle (1987) javasolt egy **egyenlő esélyű sorba állítási (fair queueing)** algoritmust.

Az algoritmus lényege, hogy a routereknek minden kimeneti vonalhoz több várakozó soruk van, és minden bemeneti vonalhoz egy. Amikor a vonal egy előző adást befejezve tétlen lesz, a router körforgó (round robin) módon megvizsgálja a hozzá tartozó sorokat, és a következő sor első csomagját veszi el továbbításra. Ily módon, ha *n* hoszt versenyez egy adott kimeneti vonalért, minden hoszt az *n* csomagból csak egyet küldhet. Több csomag küldése nem javítja ezt az arányt. Néhány ATM kapcsoló ezt az algoritmust használja.

Bár kiindulásnak jó, az algoritmusnak van egy problémája: nagyobb sávszélességet ad a nagy csomagokat használó hosztoknak, mint a kis csomagokat használó hosztoknak. Demers és mások (1990) javasoltak egy továbbfejlesztést, ahol a körforgást úgy valósítják meg, hogy bájtonkénti körforgást szimulál csomagonkénti körforgás helyett. Ez úgy működik, hogy újra és újra végignézi a sorokat bájtról bájtra, amíg meg nem találja azt az óraütést, amikor minden csomag a végére ér. Ezután a csomagokat a végetérésük sorrendjébe rendezi és ebben a sorrendben küldi el. Az algoritmust az 5.29. ábra mutatja.



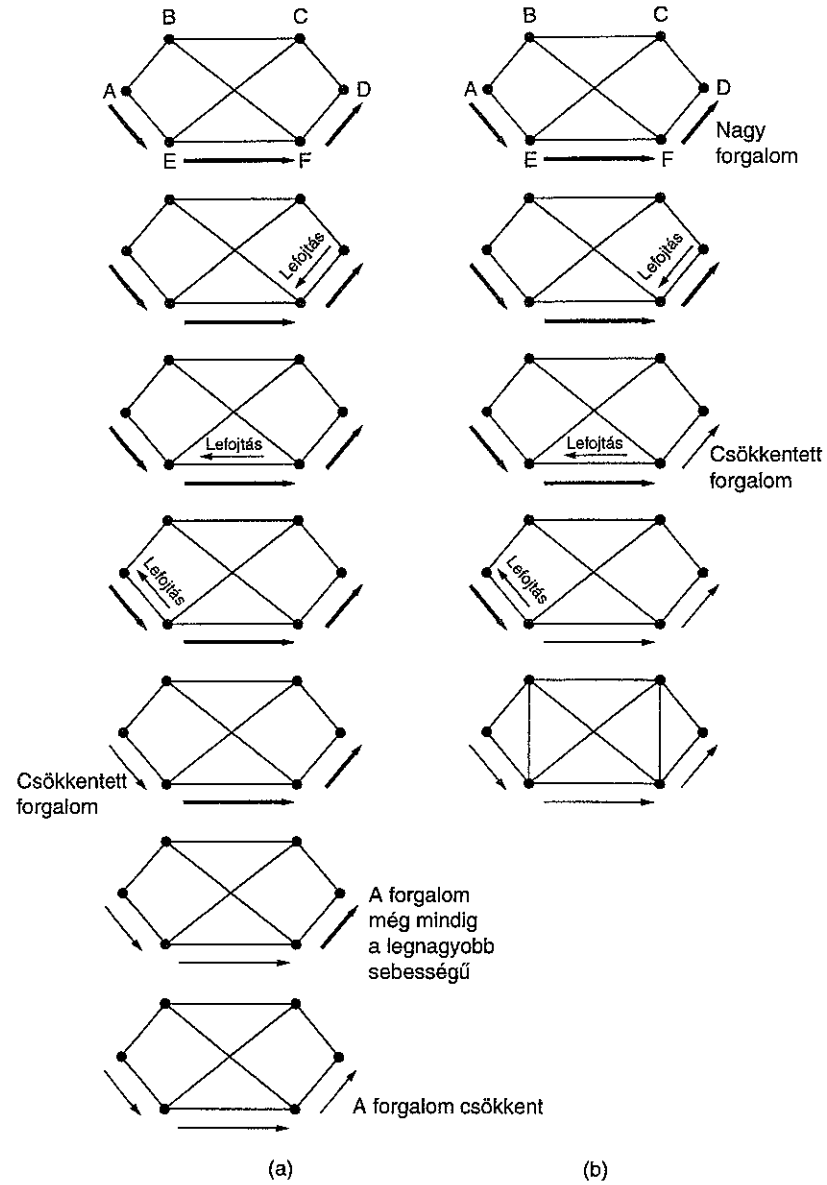
Csomag	Befejeződési idő
C	8
B	16
D	17
E	18
A	20

(b)

5.29. ábra. *(a) Egy router, amelyben az O vonal felé öt csomag áll sorban. (b) Az öt csomag befejeződésének ideje*

Az 5.29.(a) ábrán 2–6 bájt hosszú csomagokat láthatuk. Az első (virtuális) óraütéskor az *A* vonalon levő csomag első bájtját küldjük el. Ezután a *B* vonal csomagjának első bájtját és így tovább. Az első végetérő csomag a *C*, nyolc óraütés után. Az 5.29.(b) ábrán megadjuk a sorrendet. Új érkezők hiányában a csomagokat a felsorolt sorrendben küldjük el, *C*-től *A*-ig.

Ennek az algoritmusnak egy problémája, hogy minden hosztnak azonos prioritást biztosít. Sok helyzetben kívánatos a fájl- és egyéb kiszolgálóknak nagyobb sávszélességet adni, mint a klienseknek, tehát akár két vagy több bájtot is adhatunk nekik óraütésenként. Ezt a módosított algoritmust **súlyozott egyenlő esélyű sorba állításnak (weighted fair queueing)** nevezik, és elterjedten használják. Néha a súly egyenlő a gépből kijövő virtuális áramkörök vagy adatfolyamok számával, így minden folyamat egyenlő sávszélességet kap. Az algoritmus egy hatékony megvalósítását tárgyalja (Shreedhar és Varghese, 1995).



5.30. ábra. *(a) Egy lefojtó csomag, amely csak a forrásra van hatással. (b) Egy lefojtó csomag, amely minden olyan csomópontra hatással van, amelyen áthalad*

Lépésről lépésre ható lefojtó csomagok

Nagy sebességeknél és nagy távolságoknál a lefojtó csomagnak a forráshoszthoz való küldése nem működik jól, mert a reakció lassú. Vegyük például egy San Franciscó-i hosztot (az 5.30. ábrán az *A* router), amely 155 Mb/s-mal küld adatot egy New York-i hosztnak (az 5.30. ábrán a *D* router). Ha a New York-i hoszt kezd kifogyni pufferekből, kb. 30 ms-ba kerül egy lefojtó csomagnak visszajutni San Franciscóba, hogy utasítsa, lassítson le. A lefojtó csomag terjedése az 5.30.(a) ábra második, harmadik és negyedik lépésében látszik. Ezalatt a 30 ms alatt további 4,6 megabit (vagyis több mint 10 000 ATM cella) lesz már útban. Még ha a San Franciscó-i hoszt azonnal le is áll, a csőben levő 4,6 megabit továbbra is be fog folyni és foglalkozni kell vele. Csak az 5.30. ábra hetedik rajzán fog a New York-i router egy lassabb folyamot észlelni.

Egy alternatív megközelítés, hogy a lefojtó csomag minden állomásra hasson, amelyen keresztülhalad, ahogy az 5.30.(b) ábra sorozata mutatja. Itt amint a lefojtó csomag eléri *F*-et, *F*-nek csökkentenie kell a *D* felé tartó adatfolyamot. Ha így teszünk, akkor *F*-nek több puffert kell szentelnie az adatfolyamnak, mivel a forrás továbbra is teljes gőzzel ad, de *D*-nek azonnali enyhülést hoz, mint egy fejfájás-csillapító egy tv-reklámban. A következő lépésben a lefojtó csomag eléri *E*-t, és utasítja *E*-t, hogy csökkentse az *F* felé tartó adatfolyamot. Ez jobban igénybe veszi *E* puffereit, de *F*-nek azonnali enyhülést ad. Végül a lefojtó csomag eléri *A*-t, és az adatfolyam valóban lelassul.

Ennek a lépésről lépésre sémának a kiterjedő hatása gyors megkönnyebbülést biztosít a torlódás helyénél, azon az áron, hogy a folyam felsőbb részén több puffert használ. Így a torlódást csomagvesztés nélkül csírájában elfojthatjuk. Ezt az ötletet (Mishra és Kanakia, 1992) részletesebben tárgyalja, és szimulációs eredményeket is adnak.

5.3.7. Terhelés eltávolítása

Amikor a fenti módszerek közül egyik sem tünteti el a torlódást, a routerek bevethetik a nehéztüzérséget: a terhelés eltávolítását. A **terhelés eltávolítása (load shedding)** annak a szép megfogalmazása, hogy ha a routereket elárasztják olyan csomagok, amelyekkel nem tudnak megbirkózni, akkor egyszerűen kidobják azokat. A kifejezés az elektromos áram előállításának világából jön, ahol azt a gyakorlatot jelenti, hogy a közművek bizonyos területeken szándékosan áramszünetet idéznek elő, hogy az egész hálózat megmeneküljön az összeomlástól forró nyári napokon, amikor az áram iránti igény nagyban meghaladja a termelt mennyiséget.

Egy, a csomagoktól fuldokló router véletlenszerűen is kiválaszthatja az eldobandó csomagokat, de rendszerint ennél többet is tehet. Hogy melyik csomagot dobja el, az a futó alkalmazástól függ. Fájlátvitelnél egy régebbi csomag többet ér, mint egy új, mert a 6. csomag eldobása és a 7.-től 10.-ig terjedő csomagok megtartása egy hézagot okoz a vevőnél, amely miatt a 6.-tól 10.-ig terjedő csomagokat esetleg újra kell adni (ha a vevő üzemszerűen eldobja a sorrenden kívül érkező csomagokat). Egy 12 csomagból álló fájlban a 6. eldobása miatt esetleg újra kell adni a 7.-től 12.-ig terjedőket, míg a

10. eldobása miatt csak a 10.-től 12.-ig terjedőket kell esetleg újraadni. Ezzel ellentétben, a multimédiában egy új csomag fontosabb, mint egy régi. Az előbbi koncepciót (a régi jobb, mint az új) gyakran **bor (wine) politikának**, az utóbbit (az új jobb, mint a régi) gyakran **tej (milk) politikának** nevezik.

Az ennél eggyel magasabb intelligenciaszintre lépés együttműködést igényel az adótól. Sok alkalmazásnak egyes csomagok fontosabbak, mint mások. Például bizonyos mozgókép-tömörítő algoritmusok periodikusan visznek át egy egész képkeretet, és a következő képkereteket az utolsó teljes képtől való különbség formájában küldik. Ebben az esetben egy olyan csomag eldobása, amely a különbség része, előnyösebb, mint egy olyan eldobása, amely egy teljes képkeret része. Egy másik példaként vegyünk egy ASCII szöveget és képeket tartalmazó dokumentum átvitelét. Egy sor képpont elvesztése valamelyik képben sokkal kevésbé káros, mint egy sor olvasható szöveget elveszteni.

Hogy egy intelligens csomageldobó politikát valósítsunk meg, az alkalmazásoknak csomagjaikat prioritás-osztályoknak megfelelően kell megjelölniük annak jelzésére, hogy azok mennyire fontosak. Ha így tesznek, akkor amikor csomagokat kell eldobni, akkor a routerek először a legalacsonyabb osztályú csomagokat dobják el, majd a következő legalacsonyabb osztályúakat és így tovább. Persze, hacsak nincs valami jelentős ösztönzés arra nézve, hogy a csomagokat máshogy kelljen megjelölni, mint csak NAGYON FONTOS – SOHA NE DOBD EL, senki nem fogja a csomagokat osztályba sorolni.

Ösztönző eszköz lehet a pénz, mivel az alacsony prioritású csomagokat olcsóbb elküldeni, mint a magas prioritásúakat. Alternatívaként a prioritás osztályokat összekapcsolhatjuk a forgalomformálással. Például, lehet egy olyan szabály, amely azt mondja, hogy ha a vezérjeles vödör algoritmusát használjuk, és egy csomag akkor érkezik, amikor nincs a vödörben vezérjel, e csomag mégis elküldhető, feltéve, hogy legalacsonyabb prioritásúnak van jelölve, és így a baj első jelére eldobható. Kis terhelés esetén a felhasználók szerethetik ezt a működési módot, de ahogy a terhelés nő, és a csomagok tényleg eldobásra kerülnek, visszafoghatják magukat és csak akkor küldenek csomagokat, amikor van elérhető vezérjel.

Egy másik lehetőség az, amikor a hosztoknak megengedjük, hogy a virtuális áramkör felépítésekor egyeztetett megállapodásban szereplő határokat túllépjék (vagyis a megengedettnél nagyobb sávszélességet használjanak), azzal a feltétellel, hogy minden fölös forgalom alacsony prioritásúnak lesz megjelölve. Egy ilyen stratégia tulajdonképpen nem rossz ötlet, mivel jobban kihasználja a tétlen erőforrásokat, s megengedi a hosztoknak, hogy mindaddig használják ezen erőforrásokat, amíg más nem érdeklődik iránt uk, anélkül azonban, hogy a hosztok nehezebb időkben is jogot formálhatnának ezekre az erőforrásokra.

A csomagok osztályok szerinti megjelölése egy vagy több fejrészbitet igényel, amelyekbe elhelyezhetjük a prioritást. Az ATM cellákban 1 bit szolgál erre a célra, így minden ATM cella vagy alacsony, vagy magas prioritású. Az ATM kapcsolók valóban használják ezt a bitet, amikor eldobási döntéseket hoznak.

Néhány hálózatban a csomagokat nagyobb egységekbe fogják össze, amelyeket újraadási célokból használnak. Például ATM hálózatokban azok, amiket mi „csomagoknak" neveztünk, rögzített hosszúságú cellák. Ezek a cellák „üzenetek" töredékei. Ami-

kor egy cellát eldobnak, végül is az egész üzenetet újra kell adni, nemcsak a hiányzó cellát. Ilyen körülmények közt, az a router, amelyik eldob egy cellát, akár az üzenet többi celláját is eldobhatja, mivel ahhoz, hogy a maradékot átvigyük, sávszélességet igényel, és ennek átvitelével semmit nem nyerünk, még ha jól megérkezik is, mivel később az egészet újra át kell vinni.

A szimulációs eredmények megmutatták, hogy ha egy router bajt észlel a láthatáron, jobb, ha korán elkezdi eldobálni a csomagokat, minthogy várjon, amíg teljesen lelassul (Floyd és Jacobson, 1993; Romanow és Floyd, 1994). Ha így tesz, megelőzheti, hogy a torlódás megvesse a lábát.

5.3.8. Dzsitter szabályozás

Az olyan alkalmazásoknak, mint a hang- és mozgóképátvitel, nem nagyon számít, hogy a csomagoknak kézbesítéséhez 20 vagy 30 ms kell, amíg az átviteli idő állandó. Ha van néhány olyan csomagunk, amelyeknek 20 ms kellett, a többinek pedig 30 ms, ez a képet vagy a hangot egyenetlen minőségűvé teszi. Ezért a megegyezés az lehet, hogy a csomagok 99 százalékát 24,5 ms és 25,5 ms közé eső késleltetéssel továbbítják. A választott átlagos értéknek természetesen keresztülvihetőnek kell lennie. Más szavakkal, egy átlagos mennyiségű torlódást be kell számítani.

A dzsitterre korlátot adhatunk, ha kiszámoljuk a várható átviteli időt az út mentén minden átugrásra. Amikor egy csomag egy routerhez érkezik, a router ellenőrzi, hogy mennyivel siet vagy késik az ütemezéshez képest a csomag. Ha a csomag siet az ütemezéshez képest, éppen annyival tartja vissza, hogy pontos legyen. Ha késik az ütemezéshez képest, a router próbálja gyorsan kilökni az ajtón. A kimeneti vonalért versengő számos csomag közül a következőt kiválasztó algoritmus mindig választhatja az ütemezéséhez képest legtöbbet késő csomagot. Ily módon az ütemezéshez képest sietők csomagok lelassulnak, az ütemezéshez képest késő csomagok pedig felgyorsulnak, és mindkét esetben csökken a dzsitter nagysága.

5.3.9. Torlódásvédelem többesküldés esetén

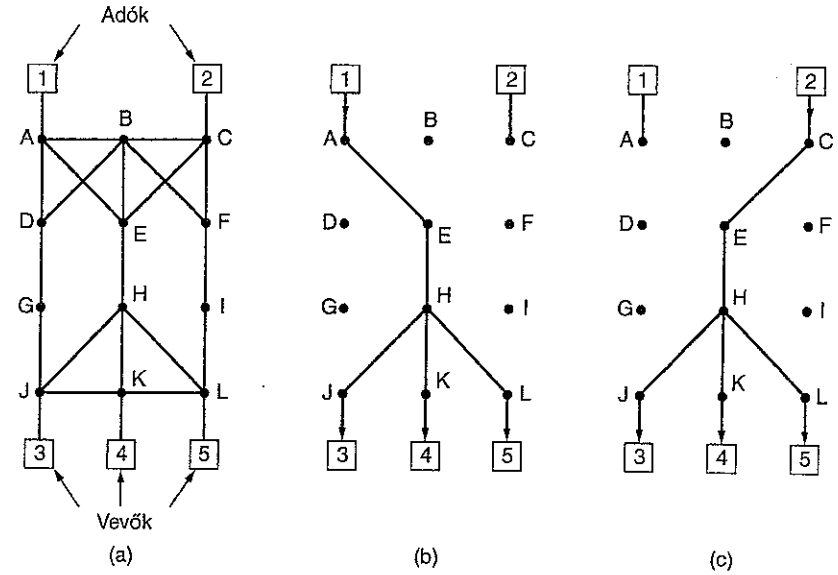
Az eddig tárgyalt torlódásvédelmi algoritmusok mindegyike egyetlen forrástól egyetlen cél felé tartó üzenetekkel foglalkozott. Ebben a részben leírunk egy módot a többesküldések több forrástól több cél felé tartó folyamainak kezelésére. Képzeljünk el például számos zártláncú televízióállomást, amelyek audio- és videofolyamokat visznek át vevők egy csoportjához, ahol mindegyik vevő egy vagy több állomást nézhet egyszerre, és kedve szerint kapcsolgathat egyik csatornáról a másikra. Ennek a technológiának az egyik alkalmazása lehet egy videokonferencia, ahol minden résztvevő az éppen beszélőre vagy a főnök arckifejezésére összpontosíthat kívánsága szerint.

Sok többesküldéses alkalmazásban a csoportok a tagságukat dinamikusan változtathatják, például amikor az emberek belépnek egy videokonferenciába, vagy megunják és átkapcsolnak egy szappanoperára. Ezen feltételek mellett az a megközelítés, hogy az adóknak előre le kelljen foglalniuk a sávszélességet, nem működik jól, mivel minden adótól megköveteli, hogy számon tartsa a közönségének minden egyes be- és kilépését, és minden változáskor újra létrehozza a feszítőfát. Egy olyan rendszerben, amelyet kábeltelevíziós átvitelre terveztek, előfizetők milliójával, az ilyen módszer egyáltalán nem is működne.

RSVP – erőforrás-foglalási protokoll

Egy érdekes megoldás, amely képes kezelni ezt a környezetet, az **RSVP (Resource reSerVation Protocol – erőforrás-foglalási)** protokoll (Zhang és mások, 1993). Ez lehetővé teszi több adó számára, hogy több vevőcsoport felé adjanak, és minden egyéni vevő szabadon átkapcsolhasson a csatornák közt, valamint optimalizálja a sávszélesség-használatot, ugyanakkor megszünteti a torlódást.

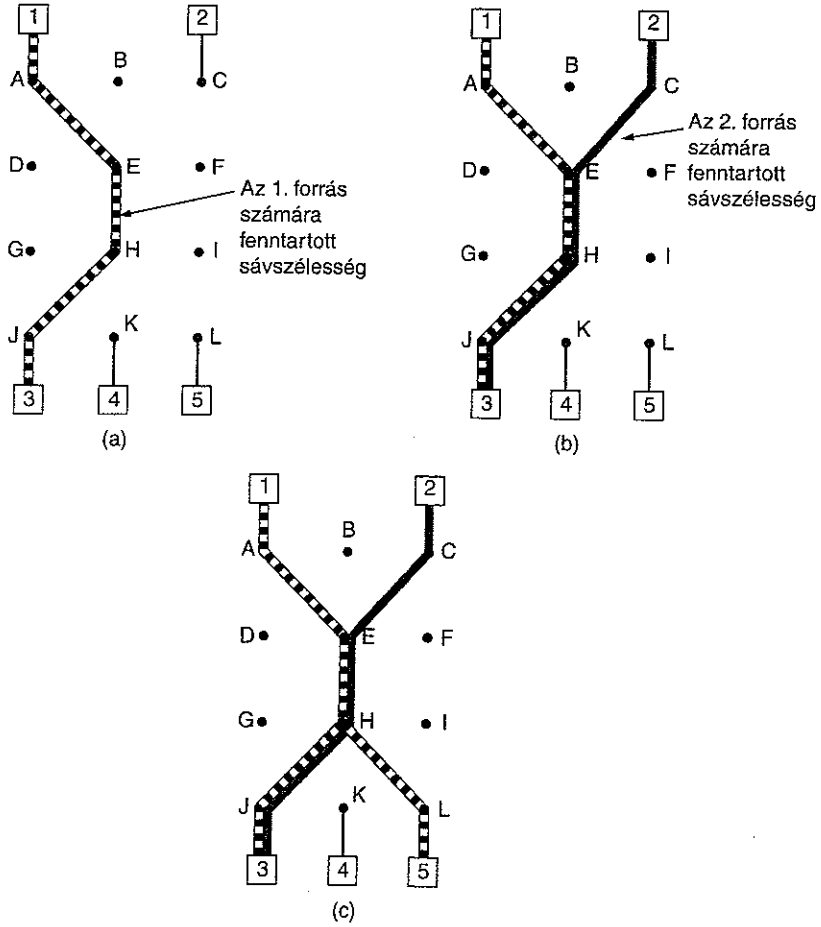
A legegyszerűbb formájában a protokoll a feszítőfát használó többesküldéses forgalomirányítást használja, ahogy azt korábban leírtuk. Minden csoporthoz hozzárendelünk egy csoportcímet. Hogy egy csoportnak küldjön, az adó a csoport címét teszi annak csomagjaiba. A szokásos többesküldéses forgalomirányítás ezután felépít egy feszítőfát, amely a csoport minden tagját lefedi. A forgalomirányító algoritmus nem az RSVP része. A rendes többesküldéstől való egyetlen eltérés az, hogy egy kevés kiegészítő információt kell a csoportnak periodikusan elküldeni, hogy a fa menti routereket utasítsuk bizonyos, a memóriájukban levő adatstruktúrák karbantartására.



5.31. ábra. *(a) Egy hálózat. (b) A többesküldéses feszítőfa az 1. hoszt számára. (c) A többesküldéses feszítőfa a 2. hoszt számára*

Hogy jobb vételt érjen el, és kiküszöbölje a torlódást, az egy csoportba levő vevők közül bármelyik küldhet egy foglalási üzenetet a fán felfelé az adóig. Az üzenetet a korábban tárgyalt visszairányú továbbítási algoritmus segítségével terjesztik. Minden lépésnél a router észreveszi a foglalást, és fenntartja a szükséges sávszélességet. Ha nem áll rendelkezésre elegendő sávszélesség, sikertelenséget jelez vissza. Amikorra az üzenet visszajut a forrásig, már le lesz foglalva a sávszélesség a feszítőfa mentén az adótól a foglaláskérést benyújtó vevőig.

Egy ilyen foglalásra látható egy példa az 5.32.(a) ábrán. Itt a 3. hoszt egy csatornát igényelt az 1. hoszthoz. Ha ez már felépült, a csomagok torlódás nélkül folyhatnak az



5.32. ábra. *(a) A 3. hoszt egy csatornát igényel az 1. hoszthoz. (b) Ezután a 3. hoszt egy második csatornát igényel a 2. hoszthoz. (c) Az 5. hoszt egy csatornát igényel az 1. hoszthoz*

1-től a 3-ba. Most vegyük szemügyre, mi történik, ha a 3. hoszt legközelebb a másik adóhoz, a 2. hoszthoz foglal csatornát, hogy a felhasználó egyszerre két televízióműsort tudjon nézni. Egy másig út is fenn lesz tartva, ahogy az 5.32.(b) ábra mutatja. Vegyük észre, hogy két külön csatornára van szükség a 3. hoszttól az *E* routerig, mert két független adatfolyam kerül továbbításra.

Végül az 5.32.(c) ábrán az 5. hoszt elhatározza, hogy az 1. hoszt által adott programot akarja nézni, és szintén foglalást végez. Először, saját célra sávszélességet foglal le a *H* routerig. Viszont ez a router látja, hogy már kapja a táplálást az 1. hoszttól, így ha a szükséges sávszélesség már le van foglalva, nem kell többet lefoglalnia. Vegyük észre, hogy a 3. és 5. hosztok már más sávszélességet kérhettek (pl. a 3.-nak fekete-fehér televíziókészüléke van, és így nincs szüksége a színinformációra), így a lefoglalt kapacitásnak olyan nagynak kell lennie, hogy kielégítse a legmohóbb vevőt is.

Amikor egy foglalást végez, a vevő (opcionálisan) megnevezhet egy vagy több forrást, amelyekből venni akar. Azt is megmondhatja, hogy ezek a választások rögzítettek a foglalás időtartama alatt, vagy a vevő meg akarja tartani a forrásváltoztatás lehetőségét későbbre. A routerek ezeket az információkat a sávszélesség-tervezés optimalizálásához használják fel, nevezetesen, két vevőt csak akkor állítanak be közös út használatára, ha mindkettő beleegyezett, hogy később ne változtasson forrást.

Ennek a stratégiának az oka a teljesen dinamikus esetben az, hogy a lefoglalt sávszélesség el van különítve a forrás választásától. Ha egy vevő már lefoglalt sávszélességet, átkapcsolhat egy másik forrásra, és megtarthatja a létező út azon részét, amely érvényes az új forrásra is. Ha a 2. hoszt például több videofolyamot is továbbít, a 3. hoszt tetszése szerint kapcsolgathat köztük, a foglalás változtatása nélkül: a routerek nem törődnek vele, milyen programot néz a vevő.

5.4. Hálózatok összekapcsolása

Egészen eddig hallgatólagosan feltételeztük, hogy csak egyetlen homogén hálózat van, amelyben minden gép minden egyes rétegben ugyanazt a protokollt használja. Sajnos ez a feltételezés túlzottan optimista. Sok különböző hálózat létezik, beleértve a LAN-okat, MAN-okat és WAN-okat is. Minden rétegben számos protokoll használata elterjedt. A következő szakaszokban gondosan szemügyre vesszük azokat a kérdéseket, amelyek akkor merülnek fel, ha kettő vagy több hálózatot egy **internetté** kapcsolunk össze.

Számottevő véleménykülönbség mutatkozik abban a kérdésben, hogy vajon a mai hálózati típusok sokfélesége egy átmeneti állapot, amely elmúlik, mihelyt mindenki rájön, hogy milyen csodálatos az [ide mindenki a saját kedvenc hálózatát írhatja], vagy vajon egy elkerülhetetlen, de állandó tulajdonsága a világnak, amely meg fog maradni. A különböző hálózatok létezése mindig különböző protokollok létezését is jelenti.

Úgy hisszük, hogy a különböző hálózatok választéka (és ezáltal a protokolloké is) mindig létezni fog, a következő okok miatt. Először is, a már meglevő különböző hálózatok nagyok és egyre nőnek. Majdnem minden UNIX rendszer TCP/IP-t futtat. Sok

nagyvállalatnak vannak még mindig SNA-t futtató mainframe-jei. A DEC még mindig fejleszti tovább a DECnet-et. A személyi számítógépekből álló LANok gyakran Novell NCP/IPX-et vagy AppleTalk-ot használnak. Végül különleges protokollokat használnak a műholdas, cellás és infravörös hálózatok. Ez az irányzat még évekig tovább fog folytatódni a létező hálózatok nagy száma miatt, és amiatt, hogy nem minden eladó tekinti ügyfelei érdekének azt, hogy egy másik eladó hálózatába történő átlépés könnyű legyen.

Másodszor, ahogy a számítógépek és hálózatok olcsóbbak lesznek, a döntések helye lefelé mozdul el. Sok társaságnál olyan értelmű politika van érvényben, hogy az egymillió dollárnál nagyobb értékű beszerzéseket a legfelső vezetésnek, a 100 000 dollárnál nagyobb értékű beszerzéseket a középszintű vezetésnek kell jóváhagynia, de a 100 000 dollárnál kisebb beszerzéseket a részlegvezetők minden felsőbb jóváhagyás nélkül elvégezhetik. Ez könnyen oda vezethet, hogy a pénzügyi osztály Ethernetet állít fel, a mérnöki osztály egy vezérjeles sínt, a személyzeti osztály pedig egy vezérjeles gyűrűt.

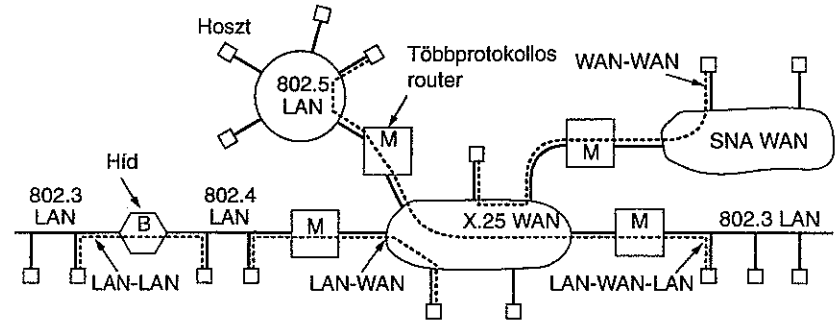
Harmadszor, a különböző hálózatoknak (pl. az ATM-nek és a vezeték nélkülinek) markánsan különbözik a technológiája, így nem lehet meglepő, hogy ahogy új hardvereket fejlesztenek ki, új szoftvereket is fognak készíteni, hogy illeszkedjenek az új hardverhez. Például az átlagos otthon ma olyan, mint az átlagos iroda tíz évvel ezelőtt: tele van számítógépekkel, amelyek nem beszélnek egymással. A jövőben teljesen természetes lehet majd az, hogy a telefon, a televíziókészülék és más eszközök egy hálózatban vannak, hogy távolról lehessen őket irányítani. Ez az új technológia kétségkívül új protokollokat fog hozni.

Egy példaként arra, hogyan működnek együtt különböző hálózatok, vegyük a következőt. A legtöbb egyetemen a számítástudományi és a villamosmérnöki tanszékeknek saját LAN-juk van, gyakran különböző. Emellett az egyetemi számítóközpontban gyakran vannak nagy számítógépek és szuperszámítógépek, az előbbiek a tanszékek humán beállítottságú dolgozói számára, akik nem kívánnak számítógép-karbantartással foglalkozni, az utóbbiak pedig a fizikusoknak, akik számokat akarnak feldolgozni. Ezen változatos hálózatok és intézmények következményeként az alábbi forgatókönyveket könnyen el lehet képzelni:

1. LAN-LAN: Egy számítástechnikus egy állományt tölt le a munkájához.
2. LAN-WAN: Egy számítástechnikus levelet küld egy távoli fizikusnak.
3. WAN-WAN: Két költő szonetteket cserél.
4. LAN-WAN-LAN: Különböző egyetemeken levő mérnökök kommunikálnak.

Az 5.33. ábra ezt a négyfajta összeköttetést szaggatott vonalakként ábrázolja. Minden esetben szükséges a hálózatok közti csomópontoknál egy „fekete dobozt" elhelyezni az egyik hálózatból a másikba haladó csomagok szükségszerű átalakítására.

A két hálózatot összekötő fekete dobozra használt név attól függ, melyik réteg végzi a munkát. Pár szokásos nevet megadtunk alább (bár nincs megegyezés ezen a területen a szóhasználatban).



5.33. ábra. *Hálózatok összehangolása*

1. réteg: Az ismétlők egyedi biteket másolnak kábelszegmensek közt.

2. réteg: A hidak adatkapcsolati kereteket tárolnak és továbbítanak LAN-ok közt.

3. réteg: A többprotokollos routerek eltérő hálózatok közt továbbítanak csomagokat.

4. réteg: A szállítási átjárók a szállítási rétegben kapcsolnak össze bájtfolyamokat.

4. felett: Az alkalmazási átjárók a 4. réteg feletti együttműködést teszik lehetővé.

Kényelmi szempontok miatt néha az „átjáró" kifejezésen bármilyen olyan eszközt fogunk érteni, amely kettő vagy több eltérő hálózatot kapcsol össze.

Az **ismétlők (repeaters)** alacsony szintű eszközök, amelyek csak erősítik vagy újragenerálják a gyenge jeleket. A hosszú kábelek meghajtásához szükséges áramot szolgáltatják. Például a 802.3-ban a MAC protokoll időzítési tulajdonságai (a választott τ érték) max. 2,5 km hosszú kábelt enged meg, de az adó-vevő chipek csak 500 méter meghajtásához elegendő teljesítményt tudnak biztosítani. A megoldás ismétlők használata, hogy a kábelhosszat a kívánt mértékben megnöveljük.

Az ismétlőktől eltérően, a **hidak (bridges)** tárol-és-továbbít típusú eszközök. A híd egy egész keretet fogad, és felfelé továbbítja az adatkapcsolati rétegnek, amely az ellenőrző összeget ellenőrzi. A keretet ezután leküldi a fizikai réteghez, hogy továbbítsa egy másik hálózaton. A hidak kisebb változtatásokat hajthatnak végre a kereten, mielőtt továbbítják. Például a keret fejrészében bizonyos mezőket hozzáadnak vagy törölnek. Mivel adatkapcsolati rétegbeli eszközök, nem foglalkoznak a 3. vagy afölötti rétegek fejrészeivel, és nem tudnak ezekre alapozott döntéseket hozni.

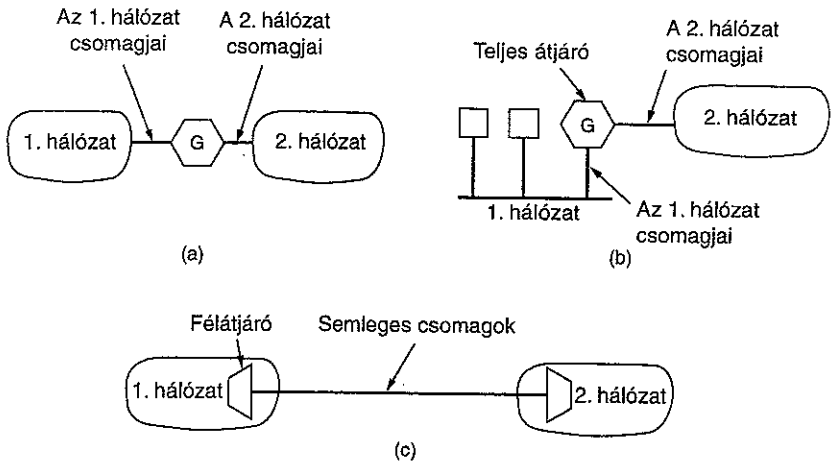
A **többprotokollos routerek (multiprotocol routers)** elvükben hasonlóak a hidakhoz, kivéve, hogy a hálózati rétegben működnek. Az egyik vonalukon veszik a beérkezett csomagokat és egy másikon továbbítják őket, mint ahogy azt minden router teszi, de a vonalak különböző hálózatokhoz tartozhatnak és különböző protokollokat használhatnak (pl. IP, IPX, és az OSI összeköttetés nélküli csomagprotokollja, a CLNP). Mint minden router, a többprotokollos routerek is a hálózati réteg szintjén működnek.

A **szállítási átjárók (transport gateways)** a szállítási rétegben teremtenek kapcsolatot két hálózat közt. Ezt a lehetőséget később tárgyaljuk, amikor az egymás után kapcsolt virtuális áramkörökhöz érünk.

Végül az **alkalmazási átjárók (application gateways)** egy alkalmazás két részét kapcsolják össze az alkalmazási rétegben. Például, ahhoz, hogy egy Internet gépről, az Internet levélformátumot használva levelet küldjön valaki egy ISO MOTIS postaládába, elküldi az üzenetet egy levelezési átjárónak. A levelezési átjáró kicsomagolja az üzenetet, átalakítja MOTIS formátumba, és továbbítja a második hálózaton az ott használt hálózati és szállítási protokollokat használva.

Amikor egy átjáró két olyan WAN között helyezkedik el, amelyeket különböző szervezetek, esetleg különböző országokban működtetnek, egy munkaállomás típusú gép mindkét fét által történő működtetése sok egymásra mutogatáshoz vezethet. Ezen problémák kiküszöbölése végett a dolog egy kicsit másfelől is megközelíthető. Az átjárót tulajdonképpen kettévágják középen, és a két részt egy vezeték köti össze. Mindkét felet **félátjárónak (half-gateway)** nevezik, és mind a kettőt egy-egy hálózatoperátor birtokolja és működteti. Az átjárás egész problémája lecsökken arra, hogy egy, a vezetéken használandó közös protokollban kell megegyezni, amelyik semleges és nem részesíti előnyben egyik felet sem. Az 5.34. ábrán láthatók a teljes és a félátjárók is. Akármelyik típust akármelyik rétegben használhatjuk (vagyis léteznek félhidak is).

Mindezt elmondva, a helyzet homályosabb a gyakorlatban, mint elméletben. Sok, a piacon levő termék kombinálja a híd és a router működését. Egy tiszta híd kulcsfontosságú tulajdonsága, hogy az adatkapcsolati keret fejrészét vizsgálja, és nem nézi vagy módosítja a hálózati rétegbeli csomagokat a kereteken belül. Egy híd nem tudja megmondani, és nem is érdekli, hogy az általa egy 802.x LAN-ról egy 802.y LAN-ra továbbított keret egy IP, IPX vagy CLNP csomagot tartalmaz-e az adat mezőben.



5.34. ábra. *(a) Egy teljes átjáró két WAN között. (b) Egy teljes átjáró egy LAN és egy WAN között. (c) Két félátjáró*

Ezzel szemben egy router nagyon is jól tudja, hogy ő egy IP router, egy IPX router, egy CLNP router, vagy mindhárom egyszerre. Megvizsgálja ezek fejrészeit és az ott talált címek alapján hoz döntéseket. Másfelől viszont, amikor egy router továbbad egy csomagot az adatkapcsolati rétegnek, nem tudja és nem is törődik vele, hogy ezt egy Ethernet keretben vagy egy vezérjeles gyűrű keretben fogják-e szállítani. Ez az adatkapcsolati réteg felelőssége.

Az iparban tapasztalható zavarodottság két forrásból jön. Először is, működés szempontjából a hidak és a routerek nem is annyira különbözőek. Mindketten bejövő PDU-kat (protokoll adategységeket) fogadnak, néhány fejrész mezőt vizsgálnak, és a fejrészbeli információk és belső táblázatok alapján eldöntik, hova küldjék a PDU-t.

Másodszor, sok kereskedelmi termék rossz címkével kerül eladásra, vagy a híd és a router feladatköreit kombinálja. Például a forrásirányított hidak egyáltalán nem is hidak, mivel a munkájuk elvégzéséhez tartalmaznak egy protokollréteget az adatkapcsolati réteg felett. A hidak és routerek vitájának megvilágosító tárgyalásáról lásd a (Perlman, 1992) 12. fejezetét.

5.4.1. Miben különböznek a hálózatok?

A hálózatok sok mindenben különbözhetnek egymástól. Az 5.25. ábrán felsorolunk pár különbséget, amelyek a hálózati rétegben fordulhatnak elő. Ezen különbségek elsimítása az, amely a hálózatok összekapcsolását nehezebbé teszi, mint az egy hálózaton belüli működést.

Amikor a csomagoknak, amelyeket az egyik hálózaton levő forrás ad, keresztül

Tétel	Néhány lehetőség
Felkínált szolgálat	Összeköttetés alapú vagy összeköttetés nélküli
Protokollok	IP, IPX, CLNP, AppleTalk, DECnet stb.
Címzés	Lapos (802) vagy hierarchikus (IP)
Többesküldés	Van vagy nincs (ugyanez adaszórásra is)
Csomagméret	Minden hálózat megszab egy legnagyobb értéket
Szolgálat minősége	Meglehet vagy hiányozhat; sokfajta létezik
Hibakezelés	Megbízható, sorrendhelyes vagy nem sorrendhelyes kézbesítés
Forgalomszabályozás	Csúszóablak, sebességszabályozás, más vagy nincs
Torlódásvédelem	Lyukas vödör, lefojtó csomagok stb.
Biztonság	Bizalmassági szabályok, titkosítás stb.
Paraméterek	Eltérő időzítések, folyammeghatározások stb.
Számlázás	Összeköttetési idő alapján, csomag alapján, bájtok alapján, vagy egyáltalán sehogy

5.35. ábra. *Néhány abból a sok dologból, amelyben a hálózatok eltérhetnek*

kell haladniuk egy vagy több idegen hálózaton, mielőtt elérnék a célhálózatot (amely szintén más lehet, mint a forráshálózat), sok gond léphet fel a hálózatok közti interfészeken. Először is, amikor egy összeköttetés alapú hálózatról származó csomagoknak át kell haladni egy összeköttetés nélkülin, sorrendjük megváltozhat. Ez olyasvalami, amire az adó nem számít és aminek a kezelésére a vevő nincs felkészülve. Gyakran protokollkonverziókra lesz szükség, amely bonyolult lehet, ha a kívánt funkciókat nem lehet megfogalmazni. Címkonverziókra is szükség lesz, amelyek valamiféle címtárrendszert igényelnek. Többesküldéses csomagok továbbítása olyan hálózaton keresztül, amely nem támogatja a többesküldést, azt igényli, hogy külön csomagot kell létrehozni minden rendeltetési helyre.

A különböző hálózatok által használt eltérő maximális csomagméretek ugyancsak nagy fejfájást okoznak. Hogyan továbbítana egy 8000 bájtos csomagot egy olyan hálózaton keresztül, amelynek a maximális mérete 1500 bájt? Gond a szolgálatok minőségének eltérése is, amikor egy valós idejű kézbesítési megkötésekkel rendelkező csomag egy olyan hálózaton halad keresztül, amely nem nyújt semmilyen valós idejű garanciát.

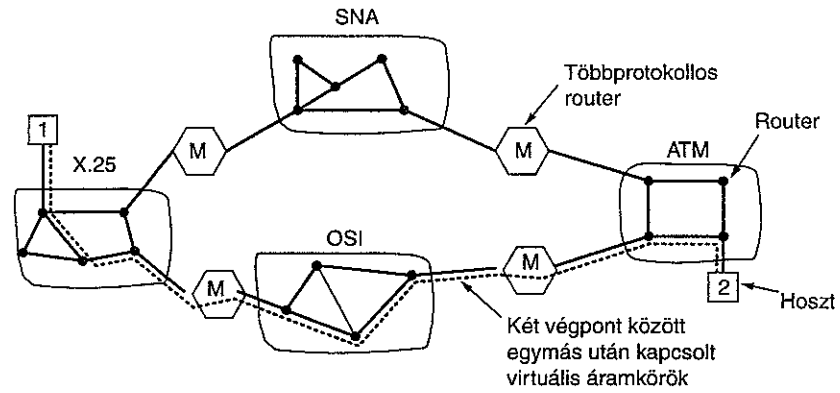
A hibavédelem, a forgalomszabályozás és a torlódásvédelem gyakran eltér különböző hálózatokban. Ha a forrás és a cél is arra számít, hogy minden csomag sorrendben hiba nélkül kézbesítésre kerül, de mégis egy közbenső hálózat eldob csomagokat, ahányszor csak torlódást lát feltűnni a láthatáron, vagy a csomagok céltalanul kószálhatnak egy ideig, majd felbukkanhatnak és kézbesítésre kerülhetnek, akkor sok alkalmazás el fog romlani. A különböző biztonsági mechanizmusok, paraméter-beállítások és számlázási szabályok, vagy akár a nemzeti személyiségi jogi törvények is okozhatnak problémát.

5.4.2. Egymás után kapcsolt virtuális áramkörök

A hálózatok összekapcsolásának kétféle megoldása lehet: a virtuális áramkör alapú alhálózatok összeköttetés alapú egymás után kapcsolása (konkatenációja), és egy datagram típusú internet. Vizsgáljuk meg ezeket részletesebben is. Az egymás után kapcsolt virtuális áramkör modellben, amit az 5.36. ábra mutat, egy távoli hálózatban levő hoszthoz hasonló módon épül fel az összeköttetés, mint ahogy azt normálisan felépítenénk. Az alhálózat látja, hogy a rendeltetési hely távoli, és egy virtuális áramkört épít fel a célhálózathoz legközelebb eső routerjéhez. Ezután felépít egy virtuális áramkört attól a routertől egy külső „átjáróig" (többprotokollos routerig). Ez az átjáró feljegyzi a virtuális áramkör létezését a táblázataiban, és továbblépve kiépít egy virtuális áramkört a következő alhálózatban levő routerig. Ez a folyamat folytatódik, amíg a célhosztot el nem érjük.

Amikor az adatcsomagok elkezdenek az útvonalon áramlani, minden átjáró átjátszsza a bejövő csomagokat, és szükség szerint konvertál a csomagformátumok és virtuális áramkör számok között. Világos, hogy minden adatcsomag ugyanazon az átjárósorozaton halad végig, és ezért sorrendben érkezik meg.

Ennek a megközelítésnek a lényeges tulajdonsága, hogy virtuális áramkörök egy sorozata épül fel egy vagy több átjárón keresztül a célig. Minden átjáró táblázatokat



5.36. ábra. *Hálózatok összekapcsolása virtuális áramkörök egymás után kapcsolásával*

tart karban, amelyek azt tartalmazzák, hogy milyen virtuális áramkörök haladnak át rajta, merre kell azokat irányítani, és mi az új virtuális áramkör száma.

Bár az 5.36. ábra a kapcsolatot egy teljes átjáróval megvalósítva mutatja, ugyanolyan jól el lehet ezt végezni félátjárókkal is.

Ez a módszer akkor működik a legjobban, ha az összes hálózatnak nagyjából ugyanolyan tulajdonságai vannak. Például, ha mindegyik megbízható kézbesítést garantál a hálózati rétegbeli csomagokra, akkor az adatfolyam a forrástól a célig szintén megbízható lesz, kivéve ha valami összeomlik az út mentén. Hasonlóan, ha egyikük sem garantálja a megbízható kézbesítést, akkor a virtuális áramkörök egymás után kapcsolása sem lesz megbízható. Másfelől, ha a forrásgép olyan hálózaton helyezkedik el, amely garantálja a megbízható kézbesítést, de a közbeeső hálózatok egyike elveszthet csomagokat, az egymás után kapcsolás alapvetően megváltoztatja a szolgálat természetét.

A virtuális áramkörök egymás után kapcsolása a szállítási rétegben is szokásos. Lehetőség van egy – mondjuk OSI-t használó – bitcsövet építeni, amely egy átjáróban végződik, és az átjárótól a következő átjáróig egy TCP összeköttetést használni. Ilyen módon egy különböző hálózatokat és protokollokat átívelő virtuális áramkört lehet a két végpont között felépíteni.

5.4.3. Hálózatok összeköttetés nélküli kapcsolódása

A másik összekapcsolt hálózati modell a datagram modell, amit az 5.37. ábra mutat. Ebben a modellben az egyetlen szolgálat, amit a hálózati réteg felkínál a szállítási rétegnek, az a lehetőség, hogy datagramokat adhat be az alhálózatba, és remélheti a legjobbat. A hálózati rétegben nem történik említés sem virtuális áramkörökről, sem ezek egymás után kapcsolásáról. Ez a modell nem követeli meg, hogy az egy összeköttetéshez tartozó összes csomag ugyanazon az átjárósorozaton haladjon végig. Az 5.37. ábrán az 1. hoszttól a 2. hoszt felé tartó datagramok különböző utakat követnek az

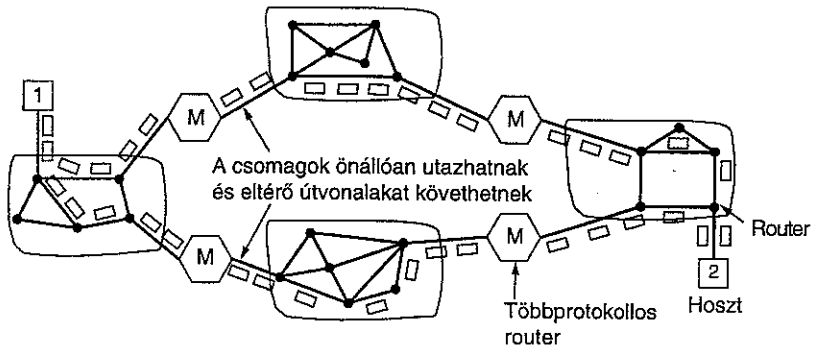
összekapcsolt hálózatokon keresztül. Minden csomagra külön hoznak forgalomirányítási döntést, esetleg a csomag elküldésének pillanatában fennálló forgalmat is figyelembe véve. Ez a stratégia több útvonalat is használhat, és ezáltal nagyobb sávszélességet érhet el, mint az egymás után kapcsolt virtuális áramkör modell. Másfelől viszont nincs garancia arra, hogy a csomagok a célhoz sorrendben érkeznek, már ha egyáltalán megérkeznek.

Az 5.37. ábra modellje nem is olyan egyszerű, mint amilyennek látszik. Például, ha minden hálózatnak saját hálózati rétegbeli protokollja van, nem lehetséges az egyik hálózatból származó csomagok számára, hogy egy másikon áthaladjanak. Elképzelhetjük, hogy a többprotokollos routerek valóban megpróbálják az egyik formátumot a másikba átfordítani. Ámde ha a két formátum nem közeli rokona egymásnak, azonos információs mezőkkel, az ilyen konverziók soha nem lesznek teljesek és gyakran kudarcra vannak ítélve. Ezen okok miatt az átalakítást ritkán kísérlik meg.

Egy másik és komolyabb probléma a címzés. Képzeljünk el egy egyszerű esetet: egy, az Internethez kapcsolódó hoszt megpróbál egy IP csomagot elküldeni egy olyan hosztnak, amely egy ezzel határos OSI hálózaton foglal helyet. Az OSI datagram protokoll, a CLNP, az IP-n alapul, és elég közel áll hozzá, hogy egy konverzió jól működhessen. A gond az, hogy az IP csomagok a fejrészükben hordozzák a célhoszt 32 bites Internet címét. Az OSI hosztoknak nincs 32 bites Internet címük. Decimális címeket használnak, hasonlókat, mint a telefonszámok.

Hogy a formátumok közti átalakítás lehetséges legyen a többprotokollos routerek számára, valakinek egy 32 bites Internet címet kellene rendelni minden egyes OSI hoszthoz. Ha a végsőkig elmegyünk, ez azt jelenti, hogy hozzá kellene rendelni egy Internet címet minden géphez a világon, amellyel egy Internet hoszt beszélni szeretne. Továbbá azt is jelenti, hogy a világon minden géphez, amellyel egy OSI hoszt beszélni szeretne, egy OSI címet is hozzá kellene rendelni. Minden más címtartománnyal (SNA, AppleTalk stb.) kapcsolatban is fellép ez a probléma. Ezek a problémák áthidalhatatlanok. Ezen kívül még valakinek egy adatbázist is fenn kellene tartania, amely mindent mindenhova leképez.

Egy másik ötlet, hogy tervezzünk egy egyetemes „internet" csomagot, és ezt ismer-



5.37. ábra. *Hálózatok összeköttetés nélküli összekapcsolása*

je fel minden router. Ez a megközelítés tulajdonképpen az IP, egy olyan csomag, amelyet arra terveztek, hogy sok hálózaton keresztülvihető legyen. Az egyetlen probléma, hogy az IPX, a CLNP és más „egyetemes" csomagok is léteznek, amely mindegyiküket kevésbé egyetemessé teszi. Nem lehetséges mindenkit egyetlen formátum elfogadására rávenni.

Foglaljuk röviden össze a két módot, ahogyan a hálózatok összekapcsolását megkezdhetjük. A virtuális áramkörök egymás után kapcsolása ugyanazokkal az előnyökkel bír, mint a virtuális áramkörök használata egyetlen alhálózaton belül: a puffereket előre lefoglalhatjuk, a sorrendhelyesség garantált, rövid fejrészeket használhatunk, és elkerülhetjük a feltartott és többszörözött csomagok által okozott problémákat.

Ugyanazok a hátrányai is: a routerekben minden megnyitott összeköttetés táblázathelyet igényel, nincs alternatív forgalomirányítás, hogy elkerülhessük a torlódott területeket, és sérülékeny az út menti forgalomirányító-meghibásodásokra nézve. Hátránya még az is, hogy bonyolult, ha nem egyenesen lehetetlen megvalósítani akkor, ha az egyik érintett hálózat egy megbízhatatlan datagram alapú alhálózat.

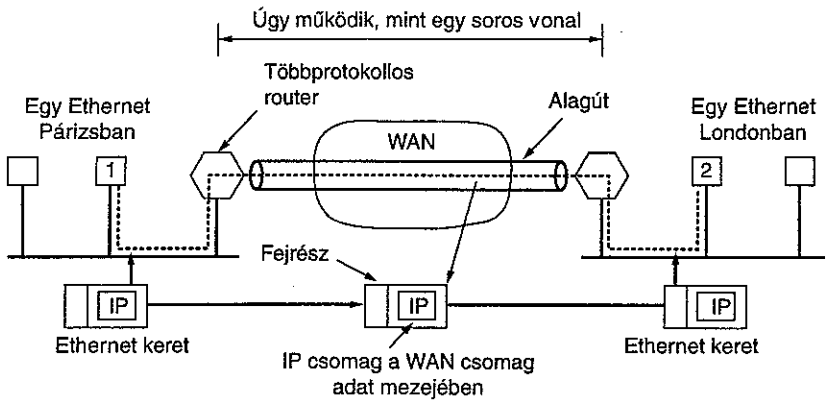
A hálózat-összekapcsolás datagram típusú megközelítésének tulajdonságai ugyanazok, mint a datagram alapú alhálózatoké: több lehetőség a torlódásra, de több lehetőség a hozzá való alkalmazkodásra is, robusztusság a router-meghibásodásokra nézve, és hosszabb fejrészek kellenek. Változatos adaptív forgalomirányítási algoritmusok lehetségesek egy interneten, mint ahogy egy egyedülálló datagram alapú alhálózaton belül is.

A hálózat-összekapcsolás datagram típusú megközelítésének egy nagy előnye, hogy olyan alhálózatok fölött is használható, amelyek belül nem virtuális áramköröket használnak. Sok LAN, mobil hálózat (pl. légi és tengeri flották), és néhány WAN is ebbe a kategóriába tartozik. Amikor egy internet ezek közül valamelyiket magába foglalja, súlyos problémák lépnek fel, ha a hálózatok összekapcsolásának stratégiája virtuális áramkörökre alapozott.

5.4.4. Alagút típusú átvitel

Két különböző hálózat együttműködését kezelni általános esetben túlságosan bonyolult. Ám van egy köznapi speciális eset, amely megoldható. Ez az eset az, amikor a forrás- és célhosztok azonos típusú hálózatokon vannak, de van köztük egy másfajta hálózat. Például gondoljunk egy bankra, amelynek van egy TCP/IP alapú Ethernete Párizsban, egy TCP/IP alapú Ethernete Londonban, és egy PTT WAN van köztük, ahogy az 5.38. ábra mutatja.

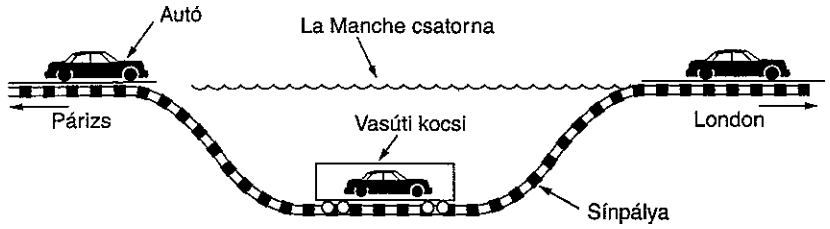
A megoldás erre a problémára egy olyan módszer, amelyet **alagút típusú átvitelnek (tunneling)** neveznek. Hogy egy IP csomagot küldjön a 2. hosztnak, az 1. hoszt összeállít egy csomagot, benne a 2. hoszt IP címével, behelyezi egy Ethernet keretbe, amelyet a párizsi többprotokollos routernek címez, és ráadja az Ethernetre. Amikor a többprotokollos router megkapja a keretet, kiveszi az IP csomagot, belehelyezi egy WAN hálózati rétegbeli csomag adat mezejébe, és ez utóbbit megcímzi a londoni többprotokollos router WAN címére. Amikor ez odaér, a londoni router kiveszi az IP csomagot, és egy Ethernet keret belsejében elküldi a 2. hosztnak.



5.38. ábra. *Alagút típusú átvitel Párizsból Londonba*

A WAN-t tekinthetjük úgy is, mint egy nagy alagutat, amely az egyik többprotokollos routertől a másikig tart. Az IP csomag egyszerűen az alagút egyik végétől a másikig utazik, kényelmesen a kis dobozában. Nem kell aggódnia amiatt, hogy foglalkoznia kell a WAN-nal. Ugyanígy a két Etherneten levő hosztoknak sem. Csak a többprotokollos routereknek kell megérteniük az IP és WAN csomagokat. Gyakorlatilag az egész távolság, ami az egyik többprotokollos router közepétől a másik közepéig terjed, egy soros vonalként működik.

Egy hasonlatként az alagút típusú átvitel világosabbá tételére, vegyünk egy embert, aki az autójával Párizsból Londonba utazik. Franciaországon belül az autó saját erejéből mozog, de amikor eléri a La Manche csatornát, berakják egy nagy sebességű vonatba és átszállítják Angliába a Csalagúton keresztül (az autóknak nem szabad behajtani a Csalagútba). Valójában az autót mint rakományt szállítják, ahogy az 5.39. ábrán látható. A másik oldalon az autót kiengedik az angol utakra és ismét saját erejéből fog mozogni. A csomagok alagút típusú átvitele egy idegen hálózaton keresztül ugyanígy működik.



5.39. ábra. *Egy autó alagút típusú átvitele Franciaországból Angliába*

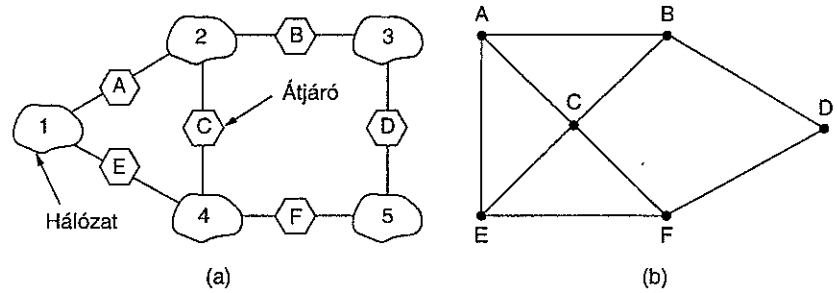
5.4.5. Forgalomirányítás összekapcsolt hálózatokban

Az összekapcsolt hálózatokon keresztüli forgalomirányítás hasonló az egyedülálló alhálózaton belüli forgalomirányításhoz, csak további bonyodalmakkal jár. Vegyük például az 5.40.(a) ábrán látható összekapcsolt hálózatokat, ahol öt hálózatot hat többprotokollos router köt össze. Ennek a szituációnak a gráfbeli modellezését bonyolítja, hogy minden többprotokollos router közvetlenül elérhet minden ahhoz a hálózathoz csatlakozó routert, ahova ő is csatlakozik (vagyis csomagokat tud küldeni annak). Például az 5.40.(a) ábrán *B* közvetlenül elérheti *A*-t és *C*-t a 2. hálózaton keresztül, és *D*-t is a 3. hálózaton keresztül. Ez az 5.40.(b) ábrán látható gráfhoz vezet.

Ha egyszer felépítettük a gráfot, ismert forgalomirányító algoritmusok, mint a távolságvektor és kapcsolatállapot alapú algoritmusok, alkalmazhatók a többprotokollos routerek halmazára. Ez egy kétszintű forgalomirányítási algoritmust eredményez: minden hálózaton belül egy **belső átjáró protokollt (interior gateway protocol)** használnak, de a hálózatok közt egy **külső átjáró protokollt (exterior gateway protocol)** használnak. (Az „átjáró" egy régebbi szóhasználat a „routerre".) Tulajdonképpen, mivel minden hálózat független, különféle algoritmusokat használhatnak. Mivel az összekapcsoltak közül mindegyik hálózat független az összes többitől, gyakran hivatkoznak rájuk **autonóm rendszerekként (Autonomous System, AS).**

Egy tipikus internet csomag a saját LAN-ján indul, a helyi többprotokollos routernek címezve (a MAC réteg fejrészében). Miután odaért, a hálózati réteg kódja eldönti a saját forgalomirányító tábláit használva, hogy melyik többprotokollos routernek továbbítsa a csomagot. Ha ez a router elérhető a csomag eredeti hálózati protokollját használva, közvetlenül oda továbbítja. Egyébként alagút típusú átvitelt használ, a közbeeső hálózat által megkövetelt protokollba beágyazva. Ezt a folyamatot addig ismétlik, amíg a csomag el nem éri a célhálózatot.

A hálózatok közti és hálózaton belüli forgalomirányítás közti egyik különbség, hogy a hálózatok közti forgalomirányításhoz gyakran át kell lépni nemzetközi határokat. Hirtelen különféle törvényeket kell figyelembe venni, mint pl. Svédország szigorú személyiségjogi törvényeit a svéd állampolgárok személyi adatainak Svédországból történő kivitelérőI. Egy másik példa az a kanadai törvény, amely kimondja, hogy a Kanadában képződő és végződő adatforgalom nem hagyhatja el az országot. Ez a tör-



5.40.ábra. *(a) Összekapcsolt hálózatok. (b) Az összekapcsolt hálózatok gráfja*

vény azt jelenti, hogy nem irányíthatjuk az ontariói Windsorból a Vancouverbe menő forgalmat a közeli Detroiton keresztül.

Egy további különbség a belső és külső irányítás közt a költségek területén mutatkozik. Egyetlen hálózaton belül általában egyetlen számlázási algoritmus érvényes. De különböző hálózatok más-más felügyeletek alá tartozhatnak, és az egyik út olcsóbb lehet, mint a másik. Hasonlóan a különböző hálózatok által felajánlott szolgálatok minősége eltérhet, és ez okot adhat arra, hogy az egyik út helyett a másikat válasszuk.

Nagy összekapcsolt hálózatokban a legjobb útvonal kiválasztása időigényes művelet lehet. Estrin és mások (1992) javasoltak egy megoldást ennek a problémának a kezelésére, mégpedig, hogy előre számítsunk ki útvonalakat népszerű (forrás-, cél-) párokra, és tároljuk ezeket egy adatbázisban, hogy az útvonal kiválasztásakor ezekre hivatkozhassunk.

5.4.6. Darabokra tördelés

Minden hálózat megszab valamilyen maximális csomagméretet. Ezeknek a határoknak különféle okai lehetnek. Néhány ezek közül a következő:

1. Hardver (pl. egy TDM időrés hossza).
2. Operációs rendszer (pl. minden puffer 512 bájtos).
3. Protokollok (pl. a csomaghossz mezőben a bitek száma).
4. Igazodás valamilyen nemzet(köz)i szabványhoz.
5. Az a kívánság, hogy a hibák által okozott újraadások valamilyen szintre csökkenjenek.
6. Az a kívánság, hogy egyetlen csomag se foglalhassa le a csatornát túl sokáig.

Mindezen tényezők eredményezik, hogy a hálózattervezők nem választhatnak tetszésük szerinti maximális csomagméretet. A maximális adatmező-hosszak 48 bájttól (ATM) 65 515 bájtig (IP csomagok) terjednek, bár az adatmező-hossz magasabb rétegekben gyakran nagyobb.

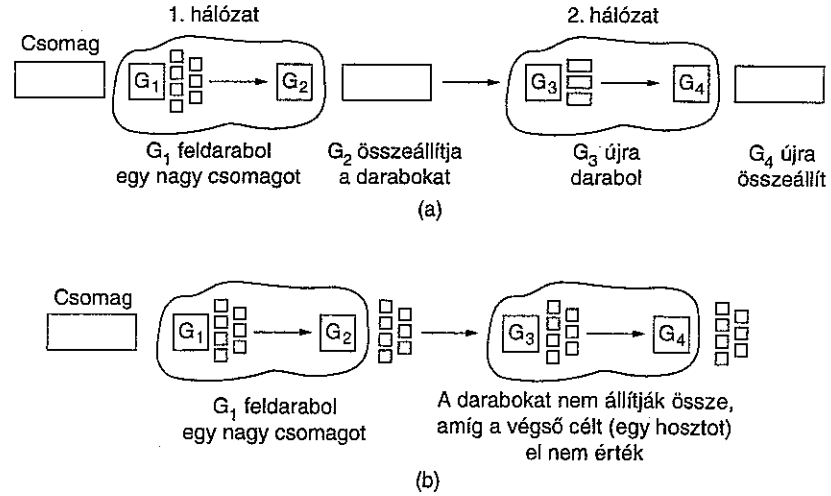
Egy nyilvánvaló probléma jelentkezik, amikor egy nagy csomag olyan hálózaton akar keresztülhaladni, amelynek a maximális csomagmérete túl kicsi. Egy megoldás az, hogy eleve elkerüljük a probléma felmerülését. Más szavakkal, az internetnek olyan forgalomirányító algoritmust kell használnia, amely nem küld olyan csomagokat olyan hálózatokon keresztül, amelyek nem tudják azokat kezelni. Viszont ez a megoldás egyáltalán nem megoldás. Mi történik, ha az eredeti csomag túl nagy ahhoz, hogy a célhálózat kezelje? A forgalomirányító algoritmus aligha tudja elérni a címzettet.

Alapvetően az egyetlen megoldás a problémára, hogy engedjük meg az átjáróknak,

hogy a csomagokat **darabokra (fragments)** tördeljék, és minden darabot mint önálló internet csomagot küldjék el. De mint ahogy azt minden kisgyermekes szülő tudja, egy nagy tárgy kisebb darabokká alakítása lényegesen könnyebb, mint a visszairányú folyamat. (A fizikusok még egy nevet is adtak ennek a jelenségnek: a termodinamika második főtétele.) A csomagkapcsoló hálózatoknak is gondjaik vannak a darabok újbóli összerakásával.

Két ellentétes stratégia létezik a darabok eredeti csomaggá történő visszaállítására. Az első stratégia, hogy a „kiscsomagos" hálózat által okozott darabolást átlátszóvá kell tenni a többi következő hálózat számára, amelyeken a csomagnak át kell haladnia a végső cél felé. Ezt a lehetőséget az 5.41.(a) ábra mutatja. Amikor egy túlméretes csomag érkezik egy átjáróhoz, az átjáró feldarabolja. Minden darabot ugyanannak a kimenő átjárónak címezi, ahol a részeket összeállítják. Ily módon a kiscsomagos hálózaton történő áthaladás átlátszó lesz. A következő hálózatok még csak nem is tudják, hogy a darabolás megtörtént. Az ATM hálózatoknak például speciális hardverjük van a csomagok cellákba való átlátszó darabolásához és a cellák csomagokká történő összeállításához. Az ATM világban a darabolást szegmentálásnak hívják; az elv ugyanaz, de néhány részlet különbözik.

Az átlátszó darabolás egyszerű, de van néhány problémája. Egyrészt a kimeneten levő átjárónak tudnia kell, mikor kapta meg az összes részt, tehát vagy egy számláló mezőt vagy egy „csomag vége" mezőt el kell helyezni minden csomagban. Másrészt az összes csomagnak ugyanazon az átjárón keresztül kell távoznia. Azzal, hogy nem engedjük meg, hogy egyes darabok más-más útvonalat kövessenek a végső cél felé, mint a többi, némi teljesítményt vesztünk. Egy utolsó probléma pedig a nagy csomagok ismételt összeállításához és feldarabolásához szükséges többletidő, amikor ezek egy sor kiscsomagos hálózaton haladnak keresztül.



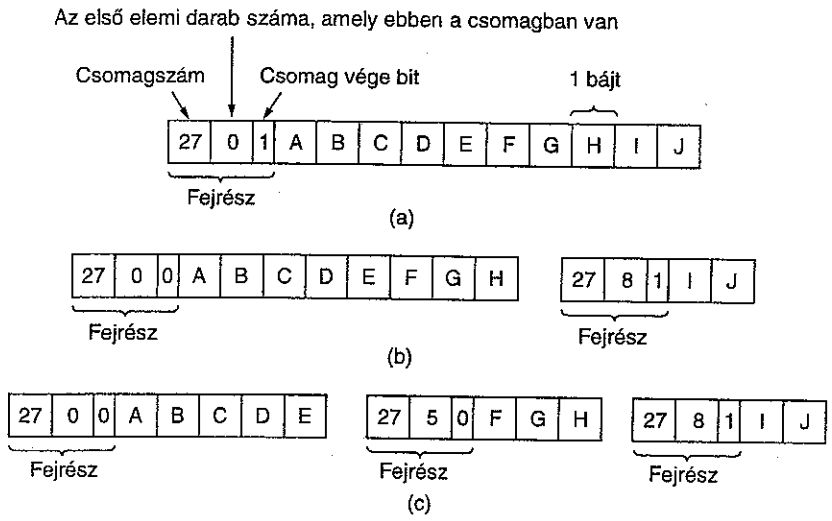
5.41. ábra. *(a) Átlátszó darabolás. (b) Nem átlátszó darabolás*

A másik darabolási stratégia, hogy tartózkodjunk a közbeeső átjáróknál az összerakástól. Ha egyszer egy csomagot feldaraboltunk, minden csomagot úgy kezelünk, mintha az egy eredeti csomag lenne. Minden darabot átjuttatunk a kimeneti átjárón keresztül, mint az az 5.41.(b) ábrán látszik. Az összeállítás csak a célhosztban történik meg.

A nem átlátszó darabolásnak is van néhány problémája. Például *minden* hoszttól megköveteli, hogy képes legyen az összeállítás elvégzésére. Még egy probléma, hogy amikor egy nagy csomagot feldarabolunk, a teljes átviendő adatmennyiség megnő, mivel minden darabnak kell legyen egy fejrésze. Míg az első módszerrel ez a többlet eltűnik, amint a csomag kilép a kiscsomagos hálózatból, ezzel az eljárással a többlet megmarad az út hátralevő szakaszára. Viszont egy előnye ennek az eljárásnak, az hogy több kimeneti átjárót is használhatunk, és nagyobb teljesítményt érhetünk el. Természetesen, ha az egymás után kapcsolt virtuális áramkör modellt használjuk, ezzel az előnnyel semmit nem érünk.

Amikor egy csomagot feldarabolunk, a darabokat oly módon kell megszámozni, hogy az eredeti adatfolyamot vissza lehessen állítani. A darabok számozásának egy módja a fa struktúra használata. Ha a 0-s csomagot fel kell hasítani, a darabokat nevezzük 0.0-nak, 0.1-nek, 0.2-nek stb. Ha ezeket a darabokat később is fel kell darabolni, a részeket így számozzuk: 0.0.0, 0.0.1, 0.0.2, ..., 0.1.0, 0.1.1, 0.1.2 stb. Ha elég mezőt tartottunk fent a fejrészben a legrosszabb esetre, és sehol nem keletkeznek másodpéldányok, ez a módszer elégséges ahhoz, hogy az összes részt helyesen össze lehessen állítani, mindegy, milyen sorrendben érkeznek.

Ellenben, ha akárcsak egy hálózat is elveszít vagy eldob csomagokat, akkor a végpontok közötti újraadásra van szükség, és ez sajnálatos hatással van a számozási rend-



5.42. ábra. *Darabolás, ha az elemi darabméret 1 bájt. (a) Eredeti csomag, amely 10 adatbájtot tartalmaz. (b) A darabok egy 8 bájtos maximális csomagméretű hálózaton való áthaladás után. (c) A darabok egy 5 méretű átjárón való áthaladás után*

szerre. Tegyük fel, hogy egy 1024 bites csomagot kezdetben négy egyenlő méretű darabra daraboltunk, és a darabokat a következőképpen számoztuk: 0.0, 0.1, 0.2 és 0.3. A 0.1 számú darab elveszik, de a többi rész megérkezik a célhoz. Nyilvánvalóan a forrásnak lejár az időzítője, és újraadja az eredeti csomagot. Csakhogy ezúttal a követett útvonal egy 512 bites határral rendelkező hálózaton halad keresztül, így két darab keletkezik. Amikor az új 0.1-es számú darab megérkezik a célhoz, a vevő azt fogja hinni, hogy most már mind a négy darab megvan és helytelenül fogja összeállítani a csomagot.

Egy teljesen más (és jobb) számozási rendszer az, hogy a hálózatközi protokollnak kell egy olyan elemi darabméretet meghatároznia, amely elég kicsi ahhoz, hogy az elemi darab minden hálózaton keresztül tudjon jutni. Amikor egy csomagot darabolnak, minden rész egyenlő lesz az elemi csomagmérettel, kivéve az utolsót, ami rövidebb lehet. Egy internet csomag számos darabot tartalmazhat hatékonysági okokból. Az internet fejrésznek tartalmaznia kell az eredeti csomagszámot, és a csomagban levő (első) elemi darab számát. Mint rendesen, kell egy olyan bit is, amely azt jelzi, hogy az internet csomagban levő utolsó elemi darab az eredeti csomag utolsó darabja.

Ez a megközelítés két sorszámmezőt igényel az internet fejrészben: az eredeti csomagszámot és a darabszámot. Nyilvánvalóan kompromisszum van az elemi darab mérete és a darabszámban levő bitek száma közt. Mivel az elemi darabméretről feltesszük, hogy minden hálózat számára elfogadható, egy sok darabot tartalmazó internet csomag további darabolása nem okoz problémát. A végső határ, hogy az elemi darab egyetlen bit vagy bájt legyen, amikor is a darabszám a bit vagy a bájt elhelyezkedését mutatja az eredeti csomagon belül, ahogy az az 5.42. ábrán látszik.

Néhány internet protokoll még tovább viszi ezt az eljárást, és a virtuális áramkörön történő egész átvitelt egy óriási csomagnak fogja fel, így minden darab tartalmazza a darab első bájtjának abszolút bájtszámát. Néhány további, a daraboláshoz tartozó kérdést tárgyal (Kent és Mogul, 1987).

5.4.7. Tűzfalak

Az a képesség, hogy bármely számítógépet, bárhol, bármely másik számítógéphez csatlakoztatni lehet, nem csak áldás. Azon egyéneknek, akik otthon ülve barangolnak az Interneten, jó móka. A vállalati biztonsági menedzsereknek egy rémálom. A legtöbb társaság nagy mennyiségű bizalmas információt tart on-line módon a hálózaton: üzleti titkokat, termékfejlesztési terveket, marketingstratégiákat, pénzügyi elemzéseket stb. Súlyos következményekkel járhat ezen információk felfedése egy versenytárs előtt.

Az információ kiszivárgásának veszélye mellett fennáll az információ beszivárgásának a veszélye is. Különösen a vírusok, férgek és más digitális kártevők (Kaufman és mások, 1995) lékelhetik meg a biztonságot, pusztíthatnak el értékes adatokat, és vesztegethetik el az adminisztrátorok idejének nagy részét, amikor azok összetakarítják a rendetlenséget, amit hagytak. Ezeket gyakran figyelmetlen alkalmazottak hozzák be, akik valami csillogó új játékot akarnak játszani.

Következésképpen, olyan mechanizmusokra van szükség, hogy a „jó" biteket bent, a „rossz" biteket pedig kinn tartsuk. Egy eljárás a titkosítás használata. Ez a megközelítés a biztonságos helyek között mozgó adatot védi, és a 7. fejezetben fogjuk tanul-

mányozni. Ámde a titkosítás semmit nem tesz azért, hogy kinn tartsa a digitális kártevőket és a hackereket. Hogy ezt a célt elérjük, a tűzfalakat kell szemügyre vennünk (Chapman és Zwicky, 1995; Cheswick és Bellovin, 1994).

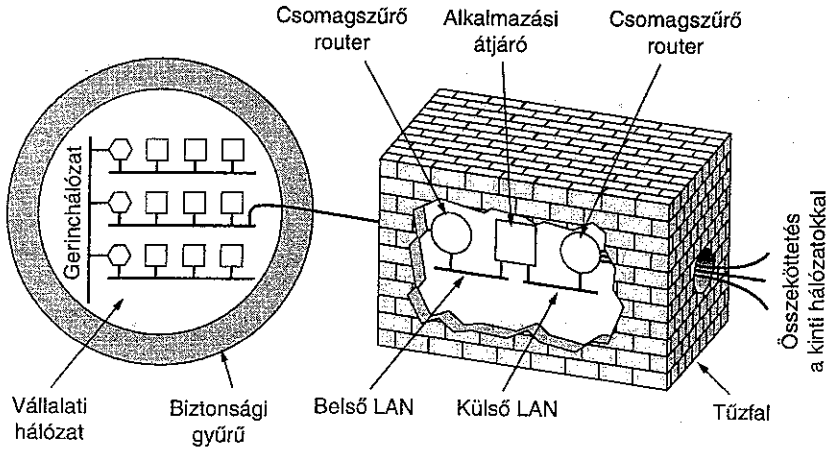
A **tűzfal (firewall)** egyszerűen egy modern adaptációja a régi középkori biztonsági intézkedésnek: egy mély árok ásásának a kastély körül. Ez a tervezés mindenkit, aki a kastélyba be- vagy onnan kilépett, arra kényszerített, hogy egyetlen felvonóhídon haladjon keresztül, ahol a kapuőrség meg tudta figyelni. A hálózatokkal is lehetséges ugyanez a trükk: egy társaságnak számos LAN-ja lehet, tetszőleges módon összekötve, de minden, a társaságtól ki- vagy befelé irányuló forgalom egy elektronikus felvonóhídon (tűzfalon) kell keresztülhaladjon, mint ahogy az 5.43. ábrán látszik.

Ebben a konfigurációban a tűzfalnak két eleme van: két router, amelyek csomagszűrést végeznek, és egy alkalmazási átjáró. Egyszerűbb konfigurációk is léteznek, de ennek a tervezésnek az előnye, hogy minden csomagnak két szűrőn és egy alkalmazási átjárón kell áthaladnia, hogy ki- vagy bejusson. Nem létezik más útvonal. Azon olvasók, akik úgy gondolják, hogy egy biztonsági ellenőrző pont elegendő, bizonyára nem repültek a közelmúltban menetrendszerű nemzetközi járaton.

Minden **csomagszűrő (packet filter)** egy szabályos router, pár külön feladatkörrel ellátva. A külön feladatkör megengedi, hogy minden kimenő vagy bejövő csomagot megvizsgáljon. A bizonyos feltételeket kielégítő csomagokat továbbítja. Amelyek nem mennek át a teszten, azokat eldobja.

Az 5.43. ábrán valószínűleg a belső LAN-on levő csomagszűrő a kimenő csomagokat ellenőrzi, a külső LAN-on levő pedig a bejövő csomagokat. Azon csomagok, amelyek vették az első akadályt, az alkalmazási átjáróhoz mennek további vizsgálatra. A két csomagszűrő külön LAN-okra helyezésének az az értelme, hogy egy csomag se juthasson ki vagy be anélkül, hogy át ne haladna az alkalmazási átjárón: nincs olyan út, amely ezt megkerüli.

A csomagszűrőket tipikusan a rendszeradminisztrátor által konfigurált táblázatok



5.43. ábra. *Egy tűzfal, amely két csomagszűrőből és egy alkalmazási átjáróból áll*

vezérlik. Ezek a táblázatok felsorolják azon forrásokat és rendeltetési helyeket, amelyek elfogadhatóak, azon forrásokat és rendeltetési helyeket, amelyeket blokkolni kell, és az alapértelmezett szabályokat arra, hogy mit kell tenni azon csomagokkal, amelyek más géptől jönnek vagy más géphez mennek.

Egy UNIX beállítás szokásos esete, hogy a forrás és a cél egy IP címből és egy portból áll. A portok azt mutatják, melyik szolgálatot igénylik. Például a 23-as port a Telneté, a 79-es port a Fingeré, és a 119-es port a USENET híreké. Egy társaság blokkolhatja azon bejövő csomagokat, amelyekben bármilyen IP cím ezen portok valamelyikével van kombinálva. Ily módon, a társaságon kívül senki nem tud Telnettel bejelentkezni, vagy a Finger démont használva embereket kikeresni. Ezen kívül a társaságot megkíméltük attól is, hogy az alkalmazottak az egész napot a USENET hírek olvasásával töltsék.

A kimenő csomagok blokkolása trükkösebb, mert bár a legtöbb helyen ragaszkodnak a szokásos portelnevezési konvenciókhoz, nincsenek rákényszerítve, hogy így tegyenek. Ezenkívül néhány fontos szolgálathoz, mint pl. az FTP (File Transfer Protocol), a portszámokat dinamikusan rendelik hozzá. Továbbá, bár a TCP összeköttetések blokkolása is bonyolult, az UDP csomagok blokkolása még nehezebb, mert nagyon kevés tudható előre abból, hogy mit fognak csinálni. Sok csomagszűrő egyszerűen az egész UDP forgalmat letiltja.

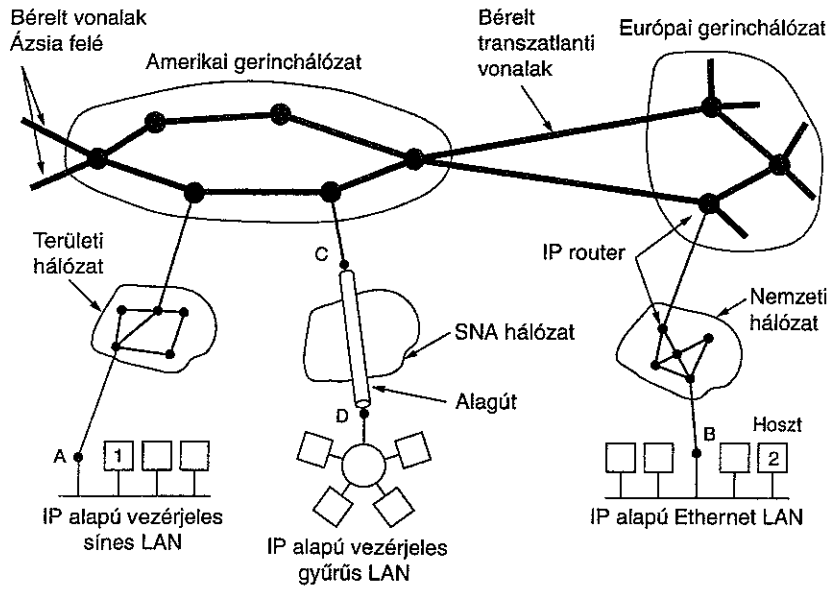
A tűzfal mechanizmusának második része az **alkalmazási átjáró (application gateway).** A nyers csomagok nézegetése helyett az átjáró az alkalmazási szinten működik. Például felállítható egy levelezési átjáró, hogy minden bemenő vagy kijövő üzenetet megvizsgáljon. Mindegyikre egy döntést hoz, hogy továbbítsa-e vagy eldobja, a fejrész mezőire, az üzenet méretére, vagy akár a tartalomra alapozva. (Például egy katonai környezetben a „nukleáris" és „bomba" szavak jelenléte valamilyen különleges tevékenység megtételét okozhatja.)

Az egyes installációkban fel lehet állítani egy vagy több alkalmazási átjárót egyedi alkalmazásokhoz, de nem szokatlan a gyanakvó szervezeteknél, hogy engedjék az e-levél forgalmat ki és be, és esetleg a World Wide Web használatát, de minden mást megtiltsanak. A titkosítással és csomagszűréssel kombinálva ez az elrendezés korlátozott biztonságot ajánl némi kényelmetlenség árán.

Egy végső megjegyzés a vezeték nélküli kommunikációról és a tűzfalakról. Könnyű olyan rendszert tervezni, amely elméletileg teljesen biztonságos, de amely gyakorlatilag lyukas, mint a szita. Ez fordulhat elő akkor, ha néhány számítógép vezeték nélküli csatornán kapcsolódik, és rádiós kommunikációt használ. Ez ugyanis átmegy a tűzfal felett mindkét irányban.

5.5. Hálózati réteg az Internetben

A hálózati réteg szintjén az Internet **autonóm rendszerek (Autonomous Systems, AS)** összekapcsolt együttesének tekinthető. Nincs valódi szerkezete, de számos főbb gerinchálózat létezik, amelyek nagy sávszélességű vonalakból és gyors routerekből állnak. A gerinchálózatokhoz csatlakoznak a területi (középszintű) hálózatok, és ezek-



5.44. ábra. *Az Internet sok hálózat egybekapcsolt összessége*

hez a hálózatokhoz csatlakoznak a sok egyetemen, társaságnál és Internet-szolgáltatónál levő LAN-ok. Ennek a kvázihierarchikus szerveződésnek egy vázlata látható az 5.44. ábrán.

Az a ragasztó, amely az Internetet egybetartja, az **IP (Internet Protocol – hálózatközi protokoll).** Sok régebbi hálózati rétegbeli protokolltól eltérően, ezt már a kezdetektől a hálózatok összekapcsolását szem előtt tartva tervezték. A hálózati rétegről egy jó elképzelés a következő lehet: az a feladata, hogy optimális szállítást biztosítson a datagramok forrásgéptől a célgépig történő eljuttatásához, függetlenül attól, hogy ezek a gépek ugyanazon a hálózaton vannak-e vagy sem, és vannak-e köztük más hálózatok vagy nincsenek.

Az Interneten a kommunikáció a következőképpen működik: a szállítási réteg veszi az adatfolyamokat, és datagramokra tördeli azokat. Elméletben a datagramok egyenként 64 Kbájt hosszúak lehetnek, de a gyakorlatban rendszerint 1500 bájt körüliek. Minden datagram átvitelre kerül az Interneten, esetleg menet közben kisebb egységekre darabolva. Amikor végül minden darab megérkezett a célgéphez, a hálózati réteg összeállítja azokat az eredeti datagrammá. A datagramot azután átadja a szállítási rétegnek, amelyik beilleszti azt a vételi folyamat bemeneti adatfolyamába.

5.5.1. Az IP protokoll

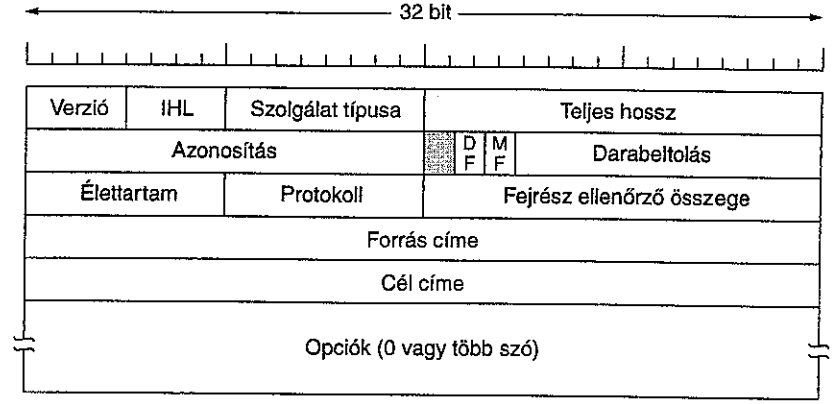
Az Internet hálózati rétegének tanulmányozásához jó kiindulási pont maguknak az IP datagramoknak a formátuma. Egy IP datagram egy fejrészből és egy szövegrészből áll. A fejrésznek van egy 20 bájtos rögzített része és egy változó hosszúságú opcionális része. A fejrész formátuma az 5.45. ábrán látható. Felsővég formában továbbítják: balról jobbra, a *Verzió* mező legmagasabb helyi értékű bitje megy elsőnek. (A SPARC felsővégű, a Pentium alsóvégű.) Az alsóvég-számítógépeken szoftveres átalakításra van szükség adáskor és vételkor is.

A *Verzió* mező azt tartja nyilván, hogy a protokoll melyik verziójához tartozik a datagram. Azzal, hogy a verzió helyet kapott a datagramban, lehetségessé vált, hogy a verziók közti átmenetek hónapokat, esetleg éveket vegyenek igénybe, amikor is az egyes gépeken a régi és más gépeken az új verzió fut.

Mivel a fejrész hossza nem állandó, a fejrész egy mezője, az *IHL*, szolgál arra, hogy 32 bites szavakban megadja a fejrész hosszát. A legkisebb érték 5, ez esetben semmilyen opció nem szerepel. Ennek a 4 bites mezőnek a maximális értéke 15, amely 60 bájtra korlátozza a fejrészt, ennélfogva az opciókat 40 bájtra. Néhány opcióhoz, mint pl. az, amelyik a csomag által megtett utat jegyzi fel, a 40 bájt túl kicsi, ezáltal az opció értelmét veszti.

A *Szolgálat típusa* mező lehetővé teszi, hogy a hoszt közölje az alhálózattal, milyen fajta szolgálatot akar. A megbízhatóságnak és sebességnek változatos kombinációi lehetségesek. A digitalizált hang számára a gyors kézbesítés fontosabb, mint a pontos kézbesítés. A fájlátvitel számára a hibamentes átvitel fontosabb, mint a gyors átvitel.

Maga a mező a következőket tartalmazza (balról jobbra): egy hárombites *Precedencia* mező, három jelzőbit, *D*, *T* és *R* és 2 kihasználatlan bit. A *Precedencia* mező egy prioritás, 0-tól (normál) 7-ig (hálózati vezérlő csomag). A három jelzőbit lehetővé teszi a hosztnak, hogy megmondja, mi neki a legfontosabb a {Késleltetés (Delay), Átbocsátás (Throughput), Megbízhatóság (Reliability)} halmazból. Elméletben ezek a



5.45. ábra. *Az IP (Internet Protocol) fejrész*

mezők lehetővé teszik a routerek számára, hogy válasszanak például egy nagy átbocsátóképességű és nagy késleltetésű műholdas kapcsolat és egy kis átbocsátóképességű és kis késleltetésű bérelt vonal közt. A gyakorlatban a mai routerek figyelmen kívül hagyják az egész *Szolgálat típusa* mezőt.

A *Teljes hossz*-ba a datagram minden része beleértendő, a fejrész és az adatrész is. A maximális hossz 65 535 bájt. Jelenleg ez a felső korlát még elmegy, de a jövőbeni gigabites hálózatoknál nagyobb datagramokra lehet majd szükség.

Az *Azonosítás* mező ahhoz szükséges, hogy a célhoszt eldönthesse, melyik datagramhoz tartozik az újonnan érkezett darab. Egy datagram minden darabja ugyanazt az *Azonosítás* értéket tartalmazza.

Egy kihasználatlan bit, majd két egybites mező következik. A *DF* jelentése: Ne Darabold (Don't Fragment). Ez egy, a routereknek szóló parancs, hogy ne darabolják fel a datagramot, mivel a cél képtelen a részek újbóli összerakására. Például amikor egy számítógép elindul, a ROM-ja kérheti, hogy küldjenek el neki egy memóriaképet egyetlen datagramban. Ezt a datagramot a *DF* bittel megjelölve, az adó tudja, hogy egy darabban fog megérkezni, még ha ez azt is jelenti, hogy el kell kerülnie egy kiscsomagos hálózatot a legjobb úton, és egy optimálisnál rosszabb útvonalat kell használnia. Minden gépnek kötelessége elfogadnia az 576 bájtos vagy kisebb darabokat.

Az *MF* jelentése: Több Darab (More Fragments). Minden darabban, kivéve az utolsóban, be kell állítani ezt a bitet. Ahhoz kell, hogy tudjuk, vajon egy datagram minden darabja megérkezett-e.

A *Darabeltolás* (Fragment Offset) megmondja, hova tartozik a mostani darab a datagramban. Egy datagram minden darabjának – kivéve az utolsót – 8 bájt többszörösének kell lennie, mert ez az elemi darabméret. Mivel 13 bit áll rendelkezésre, ez legfeljebb 8192 darabot jelent datagramonként, amely 65 536 bájtos maximális datagramhosszt ad ki, eggyel nagyobbat, mint amit a *Teljes hossz* mező lehetővé tesz.

Az *Élettartam* mező egy számláló, amelyet a csomag élettartamának korlátozására használnak. Ez elvileg az időt mérné másodpercekben, ezáltal maximálisan 255 másodperc hosszú életet tenne lehetővé. Minden átugrásnál csökkenteni kell, és ha hosszú ideig állt sorban egy routerben, akkor elvileg többször is csökkenteni kellene. Gyakorlatilag csak az átugrásokat számolja. Amikor eléri a nullát, a csomagot el kell dobni és egy figyelmeztető csomagot kell küldeni vissza a forráshoszthoz. Ez a tulajdonság megelőzi, hogy a datagramok a végtelenségig kóboroljanak, ami egyébként megtörténhetne, ha a forgalomirányító táblázatokba valamikor hiba csúszna.

Amikor a hálózati réteg összeállított egy teljes datagramot, tudnia kell, mit tegyen vele. A *Protokoll* mező mondja meg, melyik szállítási folyamatnak adja át. Lehetséges a TCP, de az UDP és pár másik is. A protokollok számozása az Interneten egységes, és az RFC 1700 definiálja.

A *Fejrész ellenőrző összeg* csak a fejrészt ellenőrzi. Egy ilyen ellenőrző összeg hasznos a routereken belüli rossz memóriaszavak által előidézett hibák kijelzésére. Az algoritmus az, hogy egyes komplemens aritmetikával adjuk össze a 16 bites félszavakat, ahogy érkeznek, és vegyük az eredmény egyes komplemensét. Ezen algoritmus miatt a *Fejrész ellenőrző összeget* a csomag érkezésekor nullának várjuk. Ez az algoritmus robusztusabb, mintha egy normális (aritmetikai) összeadást használna. Vegyük észre, hogy a *Fejrész ellenőrző összeget* minden átugrásnál újra kell számítani, mivel

legalább egy mező mindig változik (az *Élettartam* mező), de használhatók trükkök a számítás felgyorsítására.

A *Forrás címe* és *Cél címe* mutatja a hálózat számát és a hoszt számát. Az Internet címeket a következő részben tárgyaljuk. Az *Opciók* mező egy menekülési útvonal, amelyet arra terveztek, hogy a protokoll következő verzióinak lehetőségük legyen olyan információkat bevenni, amelyek az eredeti tervben nem voltak jelen, hogy kísérletezhessenek új ötletek kipróbálásával, és hogy ne kelljen olyan információk számára is fejrészbiteket lefoglalni, amelyekre csak ritkán van szükség. Az opciók változó hosszúságúak. Mindegyik egy egybájtos, az opciót azonosító kóddal kezdődik. Néhány opciónál ezt egy egybájtos hosszmező követi, majd egy vagy több adatbájt. Az *Opció* mezőt négy bájt többszörösére töltik ki. Eddig öt opciót definiáltak, ahogy az 5.46. ábrán látszik, de nem minden router támogatja mindegyiket.

A *Biztonság* opció azt mondja meg, milyen titkos az információ. Elméletben egy katonai router használhatná ezt a mezőt arra, hogy ne irányítson bizonyos, a katonaság szempontjából „rosszfiúnak" minősülő országokon keresztül. A gyakorlatban minden router figyelmen kívül hagyja, így az egyetlen gyakorlati funkciója az, hogy a kémek könnyebben megtalálhassák a jó dolgokat.

A *Szigorú forrás általi forgalomirányítás* opció IP címek sorozataként megadja a teljes utat a forrástól a célig. A datagramnak pontosan ezt az utat kell követnie. Ez nagyon hasznos rendszermenedzserek számára, hogy vészcsomagokat küldjenek a forgalomirányító táblák meghibásodása esetén, vagy időzítési méréseket végezzenek.

A *Laza forrás általi forgalomirányítás* opció megköveteli a csomagtól, hogy a megadott routereken, a megadott sorrendben áthaladjon, de útközben áthaladhat más routereken is. Rendesen ez az opció csak egy pár routert bocsát rendelkezésre, hogy egy bizonyos utat kényszerítsen ki. Például, hogy egy Londonból Sydneybe tartó csomagot kelet helyett nyugat felé kényszerítsünk, ez az opció például megadhat New York-i, Los Angeles-i és honolului routereket. Ez az opció nagyon hasznos, ha politikai vagy gazdasági megfontolásokból keresztül kell haladni bizonyos országokon, vagy el kell kerülni azokat.

Az *Útvonal feljegyzése* opció arra utasítja az útba ejtett routereket, hogy az IP címüket fűzzék hozzá az opció mezőhöz. Ez lehetővé teszi a rendszermenedzsereknek, hogy kinyomozzák a hibákat a router algoritmusokban. („Miért keresik fel a Houstonból Dallasba menő csomagok először mind Tokiót?") Amikor az ARPANET először

Opció	Leírás
Biztonság	Meghatározza, milyen titkos a datagram
Szigorú forrás általi forgalomirányítás	Megadja a teljes követendő utat
Laza forrás általi forgalomirányítás	Felsorolja a felkeresendő routereket
Útvonal feljegyzése	Minden router fűzze hozzá az IP címét
Időbélyeg	Minden router fűzze hozzá az IP címét és az időbélyegét

5.46. ábra. *IP opciók*

létrejött, egyik csomag sem érintett kilenc routernél többet, így az opció 40 bájtja bőségesen elegendő volt. Ahogy fentebb említettük, most már túl kicsi.

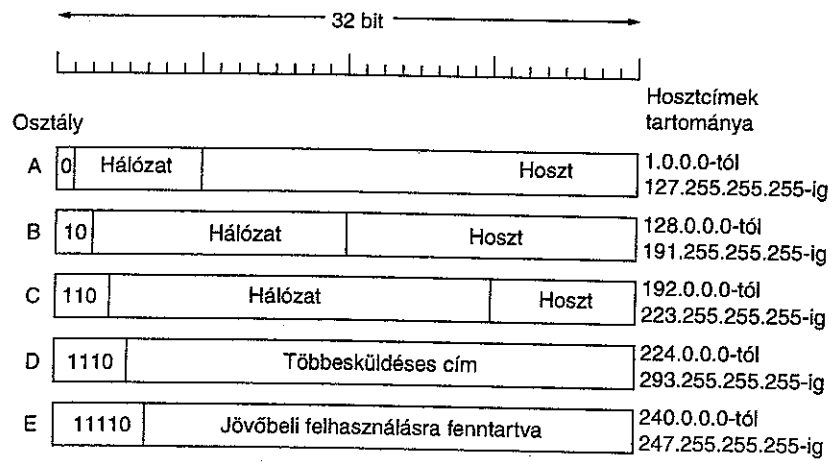
Végül az *Időbélyeg* opció olyan, mint az *Útvonal feljegyzése* opció, kivéve, hogy minden router a 32 bites IP címe mellé egy 32 bites időbélyeget is feljegyez. Ez az opció is főleg a router algoritmusok hibakereséséhez való.

5.5.2. IP címek

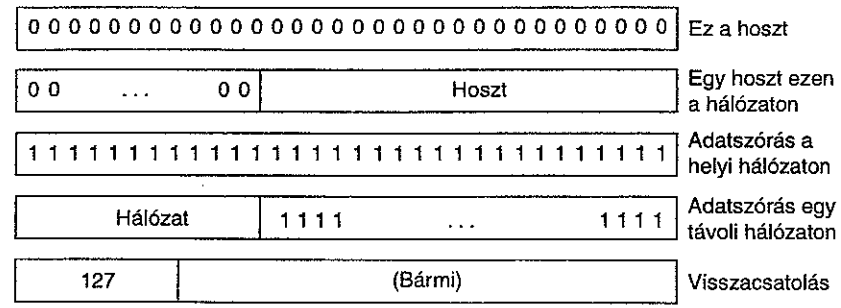
Az Interneten minden hosztnak és routernek van egy IP címe, amely a hálózat számát és a hoszt számát kódolja. A kombináció egyedi: nincs két gépnek ugyanaz az IP címe. Minden IP cím 32 bit hosszú és az IP csomagok *Forrás címe* és *Cél címe* mezőiben tartalmazzák. Az IP címekhez használt formátumok az 5.47. ábrán láthatók. A több hálózathoz is csatlakozó gépeknek mindegyik hálózaton külön IP címük van.

Az A, B, C és D osztályú formátumok lehetőségei: legfeljebb 126, egyenként 16 millió hosztból álló hálózat, 16 382 hálózat legfeljebb 64 K hoszttal, 2 millió hálózat (vagyis LAN) egyenként legfeljebb 254 hoszttal, és a többesküldés esete, ahol is egy datagramot több géphez is irányítunk. Az 11110-val kezdődő címek fenntartottak későbbi használatra. Most több tízezer hálózat kapcsolódik az Internethez, és ez a szám minden évben megduplázódik. A hálózatszámokat az **NIC (Network Information Center)** adja ki, hogy elkerüljék az ütközéseket.

A hálózati címeket, amelyek 32 bites számok, rendszerint **pontokkal elválasztott decimális jelölésrendszerben (dotted decimal notation)** írják. Ebben a formátumban minden 4 bájtot tízes számrendszerben írnak ki, 0-tól 255-ig, például a C0290614 hexadecimális címet 192.41.6.20-ként írják. A legkisebb IP cím a 0.0.0.0 és a legnagyobb a 255.255.255.255.



5.47. ábra. *IP címformátumok*



5.48. ábra. *Különleges IP címek*

A 0 és –1 értékeknek speciális jelentéseik vannak, ahogy az 5.48. ábrán látszik. A 0 érték jelentése: ez a hálózat vagy ez a hoszt. A –1-et adatszóró címként használják, és a jelzett hálózat összes hosztját értik alatta.

A 0.0.0.0 IP címet a hosztok használják, amikor elindulnak, de a későbbiekben nem használják. Az olyan IP címek, amelyeknek a hálózatszáma 0, az aktuális hálózatra vonatkoznak. Ezek a címek lehetővé teszik a gépeknek, hogy a saját hálózatukra hivatkozzanak anélkül, hogy tudnák a számát. (De az osztályát ismerniük kell, hogy tudják, hány 0-t tegyenek az elejére.) A csupa 1-ből álló cím az adatszórást teszi lehetővé a helyi hálózaton, jellemzően egy LAN-on. A helyes hálózatcímmel és csupa 1 hoszt mezővel rendelkező címek lehetővé teszik adatszóró csomagok küldését az Interneten bárhol elhelyezkedő távoli LAN-okra. Végül, az összes 127.xx.yy.zz formájú cím fenntartott a visszacsatolásos tesztelésre. Az erre a címre küldött csomagok nem kerülnek ki a vezetékre; helyileg kerülnek feldolgozásra és bejövő csomagokként kezelik azokat. Ez lehetővé teszi csomagok küldését a helyi hálózatra anélkül, hogy az adó ismerné annak számát. Ezt a tulajdonságot hálózati szoftverek hibakereséséhez is szokták használni.

5.5.3. Alhálózatok

Ahogy láttuk, egy hálózatban minden hosztnak ugyaz kell legyen a hálózatszáma. Az IP címzésnek ez a tulajdonsága gondokat okozhat, ahogy a hálózatok nőnek. Vegyünk példának egy társaságot, amely egy C osztályú LAN-nal indul el az Interneten. Ahogy múlik az idő, esetleg több mint 254 gépet is beszerez, és ezért szüksége lesz egy második C osztályú címre. Egy másik lehetőség az, hogy beszerez egy második, más típusú LAN-t, és ennek külön IP címet kér. (A LAN-ok összekapcsolhatók híddal, hogy egyetlen IP hálózatot alkossanak, de a hidaknak is megvannak a maguk problémái.) Végül is sok LAN-ja lesz, mindegyik saját routerrel és saját C osztályú címmel.

Ahogy a különálló helyi hálózatok száma nő, a menedzselésük komoly fejfájást okozhat. Minden alkalommal, amikor egy új hálózatot helyeznek üzembe, a rendszeradminisztrátornak kapcsolatba kell lépnie az NIC-vel, hogy egy új hálózatszámot szerezzen. Ezután ezt a számot világszerte ki kell hirdetni. Ezenkívül egy gép átköltözte-

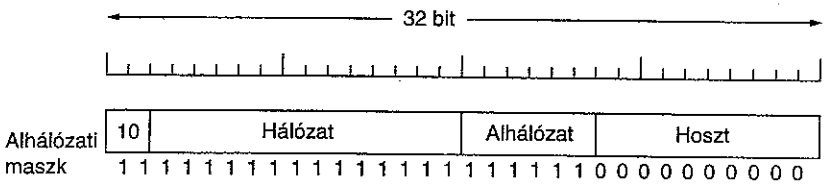
tése az egyik LAN-ról a másikra megköveteli, hogy az IP címe is megváltozzon, amely viszont a konfigurációs állományainak a módosítását és az új cím ismételt kihirdetését is maga után vonja. Ha valamilyen más gépnek adják a most felszabadított IP címet, az a gép, amíg a cím szét nem terjedt az egész világon, olyan e-leveleket és más adatokat fog kapni, amelyeket az eredeti gépnek szántak.

Ezekre a gondokra a megoldás az, ha lehetővé válik a hálózatnak több részre osztása belső használatra úgy, hogy az a külvilág felé továbbra is egyetlen hálózatként viselkedjen. Az Internet irodalmában ezeket a részeket **alhálózatoknak (subnets)** nevezik. Ahogy az 1. fejezetben megemlítettük, ez a szóhasználat ütközik azzal az „alhálózat" fogalommal, amely az összes routert és kommunikációs vonalat is jelenti egy hálózatban. Remélhetőleg világos lesz a szövegkörnyezetből, hogy a szó melyik jelentésére gondoltunk. Ebben a szakaszban az új meghatározást fogjuk használni. Ha a növekvő társaságunk egy C osztályú cím helyett egy B osztályú címmel indulna, akkor az induláshoz a hosztokat 1-től 254-ig számozhatná. Amikor a második LAN megérkezik, például dönthetne úgy, hogy a 16 bites hosztszámot egy 6 bites alhálózatszámra és egy 10 bites hosztszámra osztja, ahogy az 5.49. ábrán látszik. Ez a kettéosztás 62 LAN-ra (a 0 és a –1 fenntartottak) nyújt lehetőséget, mindegyiknél legfeljebb 1022 hoszttal.

A hálózaton kívül az alhálózatra felosztás nem látható, így egy új alhálózat elhelyezése nem igényli, hogy kapcsolatba lépjünk az NIC-vel, vagy külső adatbázisokat változtassunk meg. Ebben a példában az első alhálózat a 130.50.4.1-től kezdődő IP címeket használhatná, a második alhálózat a 130.50.8.1-nél kezdődhetne és így tovább.

Annak érdekében, hogy lássuk az alhálózatok működését, el kell magyarázni azt, hogyan dolgozza fel egy router az IP csomagokat. Minden routernek van egy táblázata, amely az alábbi bejegyzéseket tartalmazza: (hálózat, 0), IP címet leíró valamilyen szám és (ez a hálózat, hoszt) IP címet leíró valamilyen szám. Az első fajta azt mondja meg, hogyan lehet távoli hálózatokhoz eljutni. A második azt mondja meg, hogyan lehet a helyi hosztokhoz eljutni. Minden táblázathoz hozzá van rendelve a rendeltetési hely eléréséhez használandó hálózati interfész, és bizonyos egyéb információk.

Amikor egy IP csomag megérkezik, a célcímét kikeresik a forgalomirányító táblázatból. Ha a csomag egy távoli hálózatnak szól, továbbítják a táblázatban megadott interfészen a következő routernek. Ha az egy helyi (vagyis a router LAN-ján levő) hosztnak szól, akkor közvetlenül a célhoz küldik. Ha a hálózat nincs jelen a táblázatban, a csomagot egy alapértelmezett routerhez továbbítják, amelynek kiterjedtebb táblázatai vannak. Ennek az algoritmusnak az az értelme, hogy minden routernek csak a többi hálózatot és a helyi hosztokat kell nyilvántartania, nem a (hálózat, hoszt) párokat, amely nagyban csökkenti a forgalomirányító táblázat méretét.



5.49. ábra. *Egy mód egy B osztályú hálózat alhálózatokra való felosztására*

Amikor bevezetjük az alhálózatokat, a forgalomirányító táblázatok megváltoznak, megjelennek (ezen hálózat, alhálózat, 0) és (ezen hálózat, ezen alhálózat, hoszt) formájú bejegyzések. Így egy *k* alhálózaton levő router tudja, hogyan jusson el a többi alhálózathoz, és hogyan jusson el a *k* alhálózaton levő összes hoszthoz. Nem kell tudnia a részleteket a többi alhálózaton levő hosztokról. Tulajdonképpen csak annyit kell változtatni, hogy minden routernek végre kell hajtania egy logikai ÉS műveletet a hálózat **alhálózati maszkjával (subnet mask,** lásd az 5.49. ábrán), hogy megszabaduljon a hosztszámtól, és kikeresni az eredményül kapott számot a táblázataiból (miután eldöntötte, milyen osztályú címről van szó). Például egy, a 130.50.15.6-nak címzett és az 5-ös alhálózaton levő routerhez érkező csomagot ÉS kapcsolatba hozza az 5.49. ábra alhálózati maszkjával, amely a 130.50.12.0 címet adja. Ezt a címet kikeresi a forgalomirányító táblázatokból, hogy megtudja, hogyan kell a 3-as alhálózaton levő hosztokhoz eljutni. Ezzel az 5-ös alhálózaton levő router mentesül a nem az 5-ös alhálózaton levő hosztok adatkapcsolati címeinek nyilvántartása alól. Így az alhálózatok bevezetése csökkenti a táblázat helyigényét, azáltal, hogy egy háromszintű hierarchiát hoz létre.

5.5.4. Az Internet vezérlő protokolljai

Az adattovábbításra használt IP-n kívül az Internetnek számos, a hálózati rétegben használt vezérlő protokollja van, mint amilyen az ICMP, ARP, RARP és a BOOTP. Ebben a szakaszban ezeket mind áttekintjük.

ICMP

Az Internet működését routerek szorosan figyelemmel kísérik. Amikor valami váratlan esemény történik, ezt az eseményt az **ICMP (Internet Control Message Protocol – internet vezérlőüzenet protokoll)** segítségével jelentik. Az ICMP-t az Internet

Üzenet típusa	Leírás
Cél elérhetetlen	A csomagot nem lehetett kézbesíteni
Időtúllépés	Az élettartam mező elérte a 0-t
Paraméter probléma	Érvénytelen fejrész mező
Forráslefojtás	Lefojtó csomag
Átirányítás	Egy routert tanít meg a földrajzra
Visszhang kérés	Kérdés, hogy egy gép életben van-e
Visszhang válasz	Igen, életben vagyok
Időbélyeg kérés	Ugyanaz, mint a Visszhang kérés, csak időbélyeggel
Időbélyeg válasz	Ugyanaz, mint a Visszhang válasz, csak időbélyeggel

5.50. ábra. *A legfőbb ICMP üzenettípusok*

tesztelésére is használják. Körülbelül egy tucat ICMP üzenetet definiáltak. A legfontosabbakat az 5.50. ábra sorolja fel. Minden ICMP üzenetet egy IP csomagba ágyaznak be.

A CÉL ELÉRHETETLEN üzenetet akkor használják, ha az alhálózat vagy a router nem tudja megtalálni a célt, vagy egy *DF* bittel rendelkező csomagot nem lehet kézbesíteni, mert egy „kiscsomagos" hálózat az útjába állt.

Az IDŐTÚLLÉPÉS üzenetet akkor küldik, ha egy csomagot azért dobnak el, mert a számlálója elérte a nullát. Ez az esemény annak a tünete, hogy a csomagok hurokba kerültek, hogy hatalmas torlódás van, vagy hogy az időzítő értékét túl alacsonyra állították be.

A PARAMÉTER PROBLÉMA üzenet azt jelzi, hogy egy fejrész mezőbe érvénytelen érték került. Ez a probléma hibát jelez az adó hoszt IP szoftverében, vagy esetleg egy, az út során érintett router szoftverében.

A FORRÁSLEFOJTÁS üzenetet régebben a túl sok csomagot küldő hosztok visszafogására használták. Amikor egy hoszt ilyen üzenetet kapott, le kellett lassítania. Manapság már ritkán használják, mert amikor torlódás következik be, ezek a csomagok csak olajat öntenek a tűzre. A Interneten a torlódásvédelem most nagyrészt a szállítási rétegben történik, és a 6. fejezetben fogjuk tanulmányozni.

Az ÁTIRÁNYÍTÁS üzenetet akkor használják, ha egy router észreveszi, hogy egy csomag rosszul irányítottnak tűnik. A router használja, hogy a küldő hosztot értesítse a valószínű hibáról.

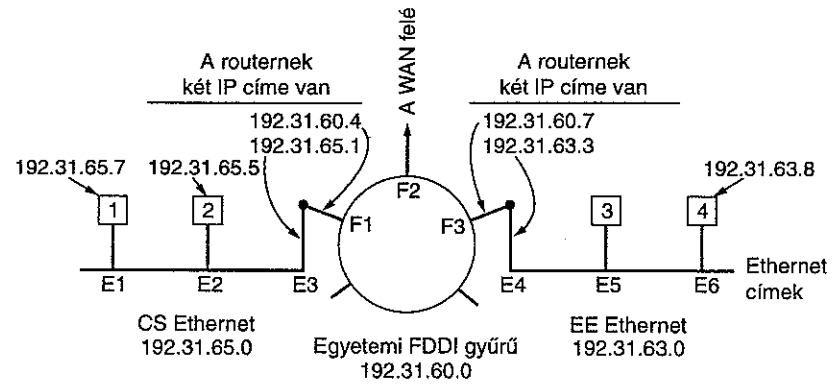
A VISSZHANG KÉRÉS és VISSZHANG VÁLASZ üzeneteket egy adott hoszt elérhetőségének és életben létének megvizsgálására használják. A VISSZHANG üzenetet megkapva, a célnak vissza kell küldenie egy VISSZHANG VÁLASZ üzenetet. Az IDŐBÉLYEG KÉRÉS és IDŐBÉLYEG VÁLASZ üzenetek hasonlóak, kivéve, hogy az üzenet érkezési ideje és a válasz indulási ideje is szerepelnek a válaszban. Ezt a tulajdonságot a hálózati teljesítmény mérésére használják.

Ezen üzeneteken kívül négy másik is van, amelyek az Internet címzéssel foglalkoznak, lehetővé téve a hosztoknak a hálózatszámuk kiderítését és azon eset kezelését, amikor több LAN osztozik egy IP címen. Az ICMP-t az RFC 792 definiálja.

ARP

Bár az Interneten levő összes gépnek van egy (vagy több) IP címe, ezeket voltaképpen nem használhatjuk csomagok küldésére, mivel az adatkapcsolati réteg hardverje nem érti az Internet címeket. Manapság a legtöbb hoszt egy olyan interfészkártyával kapcsolódik a LAN-hoz, amely csak a LAN címeket érti meg. Például minden eddig gyártott Ethernet kártya egy 48 bites Ethernet címmel felszerelve érkezik. Az Ethernet kártyák gyártói egy címtartományt igényelnek a központi hatóságtól, hogy biztosan ne legyen két kártyának ugyanaz a címe (hogy elkerüljék az ütközéseket, ha a két kártya valaha is ugyanazon a LAN-on tűnne fel). A kártyák a 48 bites Ethernet címekre alapozva küldenek és fogadnak kereteket. Semmit sem tudnak a 32 bites IP címekről.

Felmerül a kérdés: hogyan képzik le az IP címeket az adatkapcsolati rétegbeli címekre, mint amilyen az Ethernet cím is? Hogy elmagyarázzuk, hogyan működik ez, használjuk az 5.51. ábra példáját, ahol egy néhány C osztályú hálózatból álló kis egyetem látható. Két Ethernetünk van, egy a Számítástudományi Tanszéken, 192.31.65.0-s IP címmel, és egy a Villamosmérnöki Tanszéken, a 192.31.63.0-s IP címmel. Ezek egy egyetemi méretű, 192.31.60.0-s IP című FDDI gyűrűvel vannak összekapcsolva. Minden Etherneten levő gépnek van egy egyedi Ethernet címe, *E1*-től *E6*-ig jelölve, és minden FDDI gyűrűn levő gépnek van egy FDDI címe, *F1*-től *F3*-ig jelölve.



5.51. ábra. *Három összekapcsolt C osztályú hálózat: két Ethernet és egy FDDI gyűrű*

Kezdésként nézzük meg, hogyan küld az 1. hoszt egy felhasználója egy csomagot a 2. hoszt egy felhasználójának. Tegyük fel, hogy a küldő ismeri a szándéka szerinti vevő címét, valami ilyesmit: *mary@eagle.cs.uni.edu*. Az első lépés a 2., *eagle.cs.uni.edu*-ként ismert hoszt IP címének megkeresése. Ezt a felkutatást a Körzetnév Kezelő Rendszer (Domain Name System: DNS) végzi, amelyet a 7. fejezetben fogunk tanulmányozni. Most csak feltételezzük, hogy a DNS visszaadja a 2. hoszt IP címét (192.31.65.5).

Az 1. hoszt felsőbb rétegbeli szoftvere most felépít egy csomagot 192.31.65.5-tel a *Cél címe* mezőben, és átadja az IP szoftvernek átvitelre. Az IP szoftver megnézheti a címet, és láthatja, hogy a cél a saját hálózatán van, de valami módon meg kell találnia a cél Ethernet címét. Egy megoldás az, hogy legyen egy konfigurációs állomány valahol a rendszerben, amely leképezi az IP címeket Ethernet címekre. Ez a megoldás természetesen lehetséges, de a több ezer géppel rendelkező szervezetek számára ezeknek az állományoknak a naprakészen tartása hibára hajlamos, időrabló munka.

Egy jobb megoldás az, hogy az 1. hoszt kiad egy adatszóró csomagot az Ethernetre, kérdezvén: „Kié a 192.31.65.5-ös IP cím?" Az adatszórás minden géphez megérkezik a 192.31.65.0-s Etherneten, és mindegyik leellenőrzi az IP címét. Csak a 2. hoszt fog válaszolni a saját Ethernet címével (*E2*-vel). Ily módon az 1. hoszt megtudja, hogy a 192.31.65.5-ös IP cím az *E2*-es Ethernet című hoszton van. Az e kérdés feltevésére és a válasz megkapására szolgáló protokoll az **ARP (Address Resolution Protocol – címfeloldási protokoll)**. Az Interneten majdnem minden gép futtatja, és az RFC 826 definiálja.

Az ARP előnye a konfigurációs állományokkal szemben az egyszerűség. A rend-

szermenedzsernek nincs sok dolga, csak minden géphez egy IP címet kell hozzárendelnie és dönteni az alhálózati maszkok felől. Az ARP elvégzi a többit.

Ezen a ponton az 1. hoszt IP szoftvere felépít egy Ethernet keretet *E2*-nek címezve, belehelyezi a (192.31.65.5-nek címzett) IP csomagot az adat mezőbe, és ráadja az Ethernetre. A 2. hoszt Ethernet kártyája észleli a keretet, begyűjti, és egy megszakítást kezdeményez. Az Ethernet meghajtó kicsomagolja az IP csomagot az adat mezőből és átadja az IP szoftvernek, amely látja, hogy helyes a címzés, és feldolgozza.

Változatos optimalizálások lehetségesek az ARP hatékonyabbá tételére. Először is, ha egyszer egy gép futtatta az ARP-t, eltárolja az eredményt arra az esetre, ha rövid idő múlva ugyanazzal a géppel kell kapcsolatba lépnie. A következő alkalommal megtalálja a hozzárendelést a saját gyorstárában, ezáltal szükségtelenné válik a második adatszórás. Sok esetben a 2. hosztnak vissza kell küldenie egy választ, ezáltal arra kényszerül, hogy az adó Ethernet címének megállapításához futtassa az ARP-t. Ezt az ARP adatszórást el lehet kerülni, ha az 1. hoszt beleteszi az IP-Ethernet hozzárendelését az ARP csomagba. Amikor az ARP csomag megérkezik a 2. hoszthoz, a (192.31.65.7, E1) pár bekerül a 2. hoszt gyorstárába majdani felhasználásra. Tulajdonképpen az Etherneten levő összes gép bejegyezheti ezt a leképzést az ARP gyorstáraiba.

Még egy optimalizálás az, hogy minden gépnek a saját leképezését adatszórással közzé kell tenni, amikor elindul. Ezt az adatszórást általánosan egy, a saját IP címét kereső ARP formájában lehet megvalósítani. Normális esetben nem lehet válasz. Az adatszórás egy mellékhatásaként azonban egy bejegyzés keletkezik mindenkinek az ARP gyorstárában. Ha mégis lesz válasz, akkor két géphez van ugyanaz az IP cím hozzárendelve. Az újnak értesíteni kell a rendszermenedzsert és nem szabad elindulnia.

Hogy lehetőséget hagyjunk a leképzések megváltoztatására, például amikor egy Ethernet kártya tönkremegy és egy újra (és ezáltal új Ethernet címre is) cserélik, az ARP gyorstárban levő bejegyzéseknek néhány perces időzítés után le kell járniuk.

Most nézzük újra az 5.51. ábrát, csak most az 1. hoszt a 4. (192.31.63.8) hosztnak akar egy csomagot küldeni. Az ARP használata csődöt fog mondani, mivel a 4. hoszt nem látja az adatszórást (a routerek nem továbbítják az Ethernet szintű adatszórást). Két megoldás van. Először, a CS router beállítható úgy, hogy feleljen a 192.31.63.0 hálózatnak (és esetleg más lokális hálózatoknak is) szóló ARP kérésekre. Ebben az esetben, az 1. hoszt egy (192.31.63.8, E3) bejegyzést fog létrehozni az ARP gyorstárban, és boldogan küldi a 4. hosztnak szóló összes forgalmat a helyi routernek. Ezt a megoldást **helyettesítő ARP-nek (proxy ARP)** nevezik. A második megoldás, hogy az 1. hosztnak rögtön látnia kell, hogy a cél egy távoli hálózaton van, és az összes ilyen forgalmat egy alapértelmezett Ethernet címre küldje, amely az összes távoli forgalmat kezeli, ez esetben *E3*-ra. Ez a megoldás nem követeli meg a CS routertől, hogy tudja, mely távoli hálózatokat szolgál ki.

Bármelyik módon is jutottunk ide, ezután az történik, hogy az 1. hoszt az IP csomagot egy olyan keret adat mezejébe csomagolja bele, amelyet *E3*-nak címeztek. Amikor a CS router megkapja az Ethernet keretet, kiveszi az IP csomagot az adat mezőből és kikeresi az IP címet a forgalomirányító táblázataiból. Megtudja, hogy a 192.31.63.0 hálózatnak szóló csomagoknak a 192.31.60.7-es routerhez kell menniük. Ha még nem tudja 192.31.60.7 FDDI címét, egy adatszóró ARP csomagot helyez a

gyűrűre, és megtudja, hogy annak gyűrűcíme *F3*. Ezután elhelyezi a csomagot egy *F3*-nak címzett FDDI keret adat mezejében, és kirakja a gyűrűre.

Az EE routernél az FDDI vezérlő kiveszi a csomagot az adat mezőből és átadja az IP szoftvernek, amely látja, hogy 192.31.63.8-nak kell a csomagot küldenie. Ha ez az IP cím nincs az ARP gyorstárában, egy adatszórásos üzenetet bocsát ki az EE Ethernetre és megtudja, hogy a cél címe *E6*. Így egy *E6*-nak címzett Ethernet keretet épít, elhelyezi a csomagot az adat mezőben, és átküldi az Etherneten. Amikor az Ethernet keret megérkezik a 4. hoszthoz, a csomagot kiveszik a keretből és átadják az IP szoftvernek feldolgozásra.

Az 1. hosztól egy WAN-on át egy távoli hálózatig szükségképpen ugyanígy lehet eljutni, kivéve, hogy ez alkalommal a CS routert a táblázatai az *F2* FDDI című WAN router használatára utasítják.

RARP

Az ARP megoldja azt a problémát, hogy megtudjuk, melyik Ethernet cím felel meg egy adott IP címnek. Néha a fordított problémát kell megoldani: Adott egy Ethernet cím, mi a hozzá tartozó IP cím? Ez a probléma különösen akkor merül fel, amikor egy lemez nélküli munkaállomást indítunk el. Egy ilyen gép rendszerint egy távoli állománykiszolgálótól kapja meg az operációs rendszerének bináris képét. De hogyan tudja meg az IP címét?

A megoldás a (RFC 903 definiálja) **RARP (Reverse Address Resolution Protocol – fordított címfeloldási protokoll)** használata. Ez a protokoll lehetővé teszi egy újonnan indított munkaállomásnak, hogy adatszórással közölje Ethernet címét és azt mondja: „A 48 bites Ethernet címem 14.04.05.18.01.25. Tudja valaki az IP címemet?" A RARP szerver meglátja a kérést, kikeresi az Ethernet címet a konfigurációs állományaiban, és visszaküldi a megfelelő IP címet.

A RARP használata jobb, mint egy IP cím beágyazása a memóriaképbe, mert lehetővé teszi minden gépen ugyanannak a képnek a használatát. Ha az IP címek bele lennének égetve a képbe, minden munkaállomásnak saját képre lenne szüksége.

A RARP egy hátránya, hogy csupa 1-ből álló célcímet (korlátozott adatszórást) használ a RARP szerver eléréséhez, viszont az ilyen adatszórásokat nem továbbítják a routerek, így minden hálózaton szükség van egy RARP szerverre. Hogy ezt a problémát megkerüljék, feltaláltak egy másik induláskor használandó protokollt, amit **BOOTP**-nek hívnak (l. RFC 951, 1048 és 1084). A RARP-tól eltérően ez UDP üzeneteket használ, amelyeket a routerek továbbítanak. Ezen kívül további információkkal is ellátja a lemez nélküli munkaállomásokat, beleértve a memóriaképet tartalmazó állománykiszolgáló IP címét, az alapértelmezett router IP címét, és a használandó alhálózati maszkot. A BOOTP-t az RFC 951 írja le.

5.5.5. A belső átjáró protokoll: az OSPF

Ahogy már korábban is említettük, az Internet nagyszámú autonóm rendszerből (AS) tevődik össze. Minden AS-t más szervezet működtet, és belül a saját forgalomirányító algoritmusát használhatja. Például az *X*, *Y* és *Z* társaságok belső hálózata rendesen három AS-ként látszódna, ha mind a három az Interneten lenne. Mindhárom belül különböző forgalomirányító algoritmusokat használhat. Mindazonáltal az, hogy még a belső forgalomirányításhoz is léteznek szabványok, egyszerűsíti az AS-ek közti határfelületek megvalósítását, és módot ad a kód újrahasznosítására. Ebben a részben az egy AS-en belüli forgalomirányítást fogjuk tanulmányozni, a következőben pedig az AS-ek közti forgalomirányítást vesszük szemügyre. Az egy AS-en belüli forgalomirányító protokollt **belső átjáró protokollnak (interior gateway protocol)** nevezik; az AS-ek közti forgalomirányításra szolgáló algoritmus neve **külső átjáró protokoll (exterior gateway protocol).**

Az eredeti Internet belső átjáró protokollja egy, a Bellman–Ford-algoritmusra épülő távolságvektor alapú protokoll volt. Kis rendszerekben jól működött, de ahogy az AS-ek nagyobbak lettek, egyre kevésbé volt elfogadható. Szenvedett a végtelenségig számolás problémájától és általában a lassú konvergenciától is, így 1979 májusában felváltotta egy kapcsolatállapot alapú protokoll. 1988-ban az Internet Engineering Task Force elkezdte a munkát egy utódon. Ez az utód, amelynek a neve **OSPF (Open Shortest Path First),** 1990-ben szabvány lett. Ma már sok router-gyártó támogatja, és a közeljövőben ez lesz a fő belső átjáró protokoll. Alább egy vázlatot adunk arról, hogyan működik az OSPF. A teljes történetet lásd az RFC 1247-ben.

Mivel már sok tapasztalat gyűlt össze más forgalomirányító protokollokról, az új protokollt tervező csoportnak hosszú listája volt a teljesítendő követelményekről. Először is, az algoritmust a mindenki által hozzáférhető irodalomban kellett publikálni, innen az „O" (Open) az OSPF-ben. Nem felelt volna meg egy olyan megoldás, amelyet egyetlen vállalat birtokol. Másodszor, az új protokollnak mindenféle távolságmértéket támogatnia kellett, belértve a fizikai távolságot, a késleltetést és így tovább. Harmadszor, dinamikus algoritmusnak kellett lennie, méghozzá olyannak, amely a topológia változásaihoz automatikusan és gyorsan alkalmazkodik.

Negyedszerre, és ez új az OSPF-ben, támogatnia kellett a szolgálat típusára alapozott forgalomirányítást. Az új protokollnak képesnek kellett arra lennie, hogy a valós idejű forgalmat egy adott útvonalra, a másfajta forgalmat pedig másfelé irányítsa. Az IP protokollnak van egy *Szolgálat Típusa* mezeje, de semmilyen létező protokoll nem használta.

Ötödszörre, és a fentiekkel összefüggésben, az új protokollnak terheléskiegyenlítést is kellett végeznie, a terhelést több vonalra szétosztva. A legtöbb ezt megelőző protokoll minden csomagot a legjobb útvonalon küldött el. A második legjobb útvonalat egyáltalán nem használták. Sok esetben a terhelés több vonalra való szétosztása jobb teljesítményt ad.

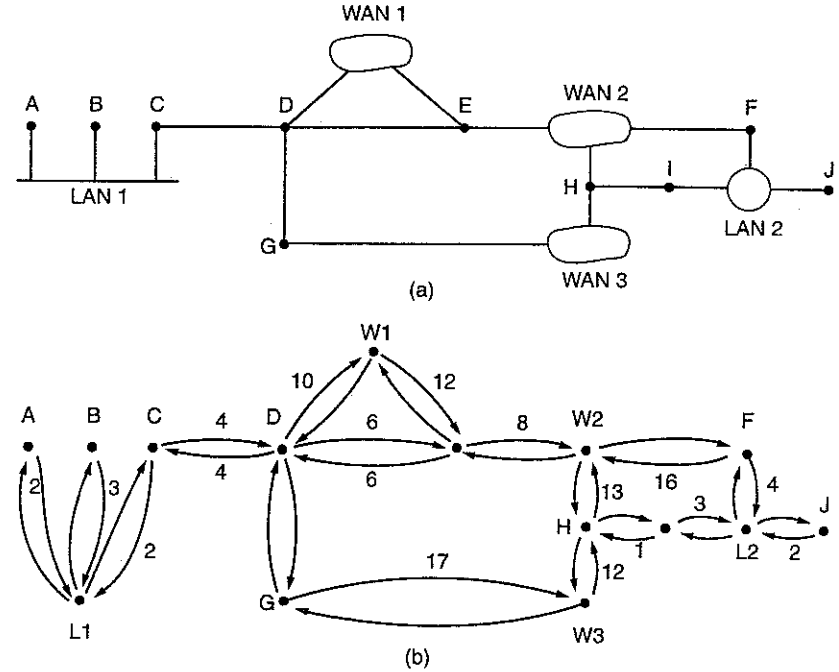
Hatodszor, a hierarchikus rendszerek támogatására is szükség volt. 1988-ra az Internet már olyan nagyra nőtt, hogy egyik routertől sem lehetett elvárni, hogy ismerje az egész topológiát. Az új protokollt úgy kellett megtervezni, hogy erre egyik routernek se legyen szüksége.

Hetedszerre, valami kevés biztonság is kellett, hogy a mókás kedvű hallgatókat meggátolják abban, hogy a routereket hamis forgalomirányítási információ küldésével félrevezessék. Végül valamilyen rendelkezés kellett az olyan routerek kezeléséhez, amelyek alagút típusú átvitelen keresztül kapcsolódtak az Internethez. Az előző protokollok ezt nem kezelték jól.

Az OSPF háromfajta összeköttetést és hálózatot támogat:

1. Kétpontos vonalak pontosan két router közt.
2. Többszörös hozzáférésű hálózatok adatszórási lehetőséggel (pl. a legtöbb LAN).
3. Többszörös hozzáférésű hálózatok adatszórási lehetőség nélkül (pl. a legtöbb csomagkapcsolt WAN).

A **többszörös hozzáférésű (multiaccess)** hálózat olyan, hogy több router is lehet rajta, és ezek mindegyike közvetlenül tud a másikkal kommunikálni. Minden LAN és WAN rendelkezik ezzel a tulajdonsággal. Az 5.52.(a) ábra egy olyan AS-t mutat, amely mind a háromfajta hálózatot tartalmazza. Vegyük észre, hogy a hosztok általában nem játszanak szerepet az OSPF-ben.



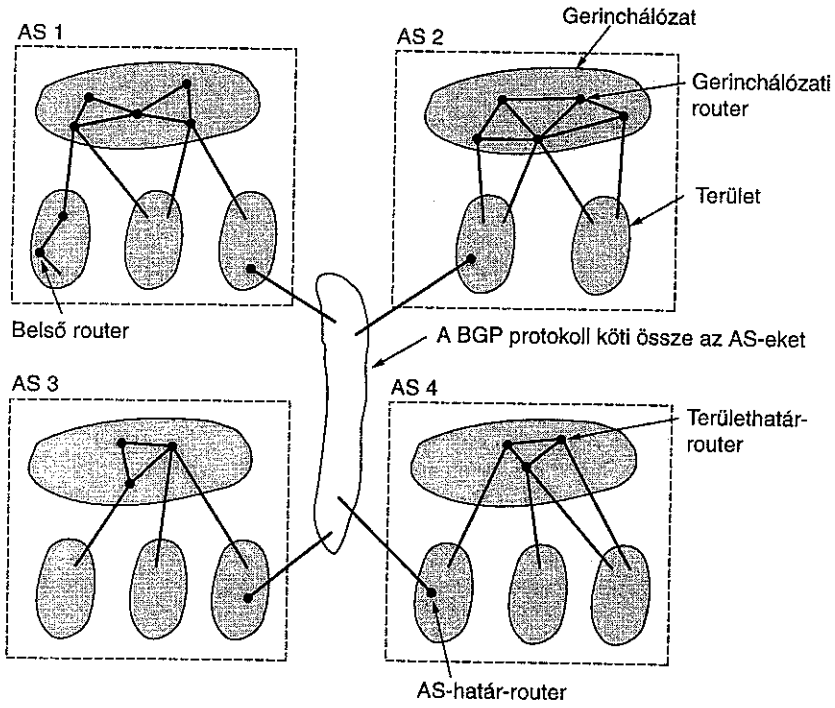
5.52. ábra. *(a) Egy autonóm rendszer. (b) (a) egy gráfreprezentációja*

Az OSPF úgy működik, hogy a valódi hálózatok, routerek és vonalak összességét egy irányított gráfba képzi le, ahol minden élhez tartozik egy költség (távolság, késleltetés stb.). Ezután az élek súlyaira alapozva kiszámítja a legrövidebb utat. Két router közti soros kapcsolatot egy élpár jelképez, mindkét irányba egy-egy éllel. Ezek súlyai különbözhetnek. Egy többszörös hozzáférésű hálózatot egy csomópont jelképez, minden routernek megfelel egy további csomópont. A hálózatnak megfelelő csomópontból a routerek felé vezető éleknek 0 súlyuk van, és ki is maradnak a gráfból.

Az 5.52.(b) ábra mutatja az 5.52.(a) ábra gráfreprezentációját. Alapjában véve az OSPF azt csinálja, hogy a valódi hálózatot egy ilyen gráfba képzi le, majd kiszámítja a legrövidebb utat minden routertől minden más routerig.

Az Interneten sok AS önmagában is nagy, és nem magától értetődő a menedzselésük. Az OSPF lehetővé teszi, hogy ezeket megszámozott **területekre (area)** osszuk fel, ahol egy terület egy hálózat vagy hálózatok összefüggő halmaza. A területek nem fedhetik át egymást, de nem kell kimerítőnek lenniük, vagyis néhány router esetleg egyik területhez sem tartozik. A terület az alhálózatok egy általánosítása. Egy területen kívülről annak topológiája és részletei nem láthatóak.

Minden AS-nek van egy **gerinchálózati (backbone)** területe, amit 0. területnek hívnak. Minden terület csatlakozik a gerinchálózathoz, esetleg alagutakon keresztül,



5.53. ábra. *Az AS-ek, gerinchálózatok és területek viszonya az OSPF-ben*

így az AS bármely területéről bármelyik másik területére el lehet jutni a gerinchálózaton keresztül. Az alagutat a gráfban egy él és egy költség jelképezi. Minden router, amely két vagy több területhez csatlakozik, a gerinchálózat része. Mint az más területekre is igaz, a gerinchálózat topológiája sem látszik a gerinchálózaton kívülről.

Egy területen belül minden router ugyanazzal a kapcsolatállapot adatbázissal rendelkezik, és ugyanazt a legrövidebb út algoritmust futtatja. A legfőbb feladata, hogy kiszámítsa a legrövidebb utat saját magától minden már routerig a területen belül, beléértve azt a routert is, amelyik a gerinchálózathoz kapcsolódik. Ilyenből legalább egy kell legyen. Egy olyan routernek, amely két területhez is kapcsolódik, mindkét terület adatbázisára szüksége van, és mindkettőre külön kell futtatnia a legrövidebb út algoritmust.

Az OSPF úgy kezeli a szolgálat típusára alapozott forgalomirányítást, hogy több gráfja is van: ezek közül az egyiket a késleltetés mértékének megfelelő költséggel címkézik, a másikat a teljesítőképesség mértékének megfelelő költséggel, a harmadikat pedig a megbízhatóság mértékének megfelelő költséggel. Bár ez megháromszorozza a szükséges számításokat, lehetőséget ad külön a késleltetést, átbocsátóképességet és megbízhatóságot optimalizáló útvonalak meghatározására.

Rendes működés közben háromfajta útvonalra lehet szükség: területen belülire, területek közöttire és AS-ek közöttire. A területen belüli útvonalak a legkönnyebbek, mivel a forrás-router már tudja a legrövidebb utat a cél-routerig. A területek közti forgalomirányítás mindig három lépésben megy végbe: menjünk a forrástól a gerinchálózatig; menjünk a gerinchálózaton keresztül a célterületig; menjünk a célig. Ez az algoritmus egy csillag elrendezést kényszerít az OSPF-re, ahol a gerinchálózat a csillag közepe és a többi területek az ágai. A csomagokat úgy irányítjuk a forrástól a célig, „ahogy vannak". Nem ágyazzuk be azokat és nem visszük keresztül alagutakon, hacsak nem egy olyan területre megy, amelyet a gerinchálózattal csak egy alagút köt össze. Az 5.53. ábra az Internet egy részét mutatja AS-ekkel és területekkel.

Az OSPF négy router osztályt különböztet meg:

1. A belső routerek teljes egészükben egy területen belül vannak.
2. A területhatár-routerek két vagy több területet kötnek össze.
3. A gerinchálózati routerek a gerinchálózaton vannak.
4. Az AS-határ-routerek a más AS-ekben levő routerekkel beszélnek.

Ezek az osztályok átfedhetik egymást, például minden határ-router automatikusan a gerinchálózat része is. Ezen kívül, egy olyan router, amely a gerinchálózaton van, de nem része semmilyen más területnek sem, belső router is. Az 5.53. ábra mind a négy router osztályra mutat példát.

Amikor egy router elindul, HELLO üzeneteket küld minden kétpontos vonalán, és ezeket többesküldéssel elküldi a LAN-okon az összes többi routerből álló csoportnak is. WAN-okon valamilyen konfigurációs információra van szüksége, hogy tudja, kivel lépjen kapcsolatba. A válaszokból minden router megtudja, kik a szomszédai.

Az OSPF azáltal működik, hogy információt cserél az **összefüggő (adjacent)** routerek közt, amely nem ugyanaz, mint a szomszédos routerek. Nem hatékony, hogy egy LAN-on minden router minden másikhoz beszéljen. Hogy ezt a szituációt elkerüljék, egy routert megválasztanak **kijelölt routernek (designated router).** Ezt nevezik az összes többi routerrel összefüggőnek, és ez cserél velük információt. Az olyan szomszédos routerek, amelyek egymással nem összefüggőek, egymás közt nem cserélnek információt. Mindig naprakészen tartanak egy tartalék kijelölt routert is, hogy az átállást megkönnyítsék, amennyiben az elsődleges kijelölt router összeomlana.

Rendes működés közben minden router periodikusan eláraszt KAPCSOLATÁLLAPOT FRISSÍTÉS üzenetekkel minden vele összefüggő routert. Ez az üzenet megadja annak állapotát, és biztosítja a topológiai adatbázisban használt költségeket. Az elárasztási üzeneteket nyugtázzák, hogy megbízhatóak legyenek. Minden üzenetnek van egy sorszáma, így a routerek láthatják, hogy egy bejövő KAPCSOLATÁLLAPOT FRISSÍTÉS régebbi vagy újabb annál, amelyik nekik megvan. A routerek akkor is elküldik ezeket az üzeneteket, amikor egy vonal tönkremegy vagy megjavul, vagy a költsége megváltozik.

Az ADATBÁZIS LEÍRÁS üzenetek megadják minden, a küldő által jelenleg birtokolt kapcsolatállapot bejegyzés sorszámát. A vevő a saját értékeit az adóéval összehasonlítva eldöntheti, kinek van a legújabb értéke. Ezeket az üzeneteket egy vonal megjavításakor használják.

Bármely partner kérhet kapcsolatállapot információt a másiktól a KAPCSOLATÁLLAPOT KÉRÉS üzeneteket használva. Ennek az algoritmusnak a tovaterjedő hatása az, hogy minden összefüggő router-pár ellenőrzi, hogy kinek vannak a legújabb adatai, és ily módon az új információ elterjed a területen belül. Mindezeket az üzeneteket nyers IP csomagokként küldik el. Az öt üzenettípus összefoglalása látható az 5.54. ábrán.

Végre összerakhatjuk a darabokat. Az elárasztást használva, minden router tudatja a területén belüli összes többi routerrel a szomszédait és költségeit. Ez az információ lehetővé teszi mindegyik router számára, hogy összeállítsa a területe(i) gráfját, és kiszámítsa a legrövidebb utat. Ezt a gerinchálózati terület is elvégzi. Ezen kívül a gerinchálózati routerek a területhatár-routerektől is elfogadnak információt, hogy kiszámolják a legjobb útvonalat minden gerinchálózati routertől minden más routerig. Ezt az információt visszafelé is elterjesztik a területhatár-routerekhez, amelyek kihirdetik ezt a körzeteikben. Ezt az információt használva egy router, amely egy területek közti csomagot készül küldeni, kiválaszthatja a legjobb kijáratot a gerinchálózat felé.

Üzenet típusa	Leírás
HELLO	A szomszédok felderítésére használatos
KAPCSOLATÁLLAPOT FRISSÍTÉS	Az adó költségeit adja meg a szomszédainak
KAPCSOLATÁLLAPOT NYUGTA	Nyugtázza a kapcsolatállapot frissítést
ADATBÁZIS LEÍRÁS	Bejelenti, mely frissítésekkel bír az adó
KAPCSOLATÁLLAPOT KÉRÉS	Információt kér a partnertől

5.54. ábra. *Az öt OSPF üzenettípus*

5.5.6. A külső átjárók router protokollja: a BGP

Az Interneten az egy AS-en belülre ajánlott router protokoll az OSPF (bár nem csak ezt használják). Az AS-ek közt egy másik protokollt, **BGP-t (Border Gateway Protocol – határátjáró protokoll)** használnak. Az AS-ek közt azért van másfajta protokollra szükség, mert egy belső átjáró protokoll és egy külső átjáró protokoll céljai nem ugyanazok. A belső átjáró protokollnak csak annyi a dolga, hogy a csomagokat a lehető leghatékonyabban mozgassa a forrás és a cél közt. Nem kell törődnie a politikával.

A külső átjáró protokolloknak viszont sokat főhet a fejük a politika miatt. Például egy vállalati AS-nek kellhet az a képesség, hogy csomagokat tudjon küldeni az Internet bármely helyére, és csomagokat tudjon fogadni az Internet bármely helyéről. Viszont nem biztos, hogy hajlandó egy idegen AS-ben keletkezett és egy másik idegen AS-be tartó csomagokat szállítani, még ha a saját AS a legrövidebb úton is lenne a két idegen AS közt. („Ez az ő problémájuk, nem a mienk.") Másfelől viszont hajlandó lehet átmenő forgalmat szállítani a szomszédai vagy esetleg külön AS-ekét is, amelyek fizettek ezért a szolgáltatásért. Például a telefontársaságok boldogan működnének szolgáltatóként az ügyfeleik számára, de más számára nem. Általában a külső átjáró protokollokat, így a BGP-t is úgy tervezték meg, hogy sokfajta forgalomirányítási politika kényszeríthető rá vele az AS-ek közti forgalomra.

A tipikus politikák nagypolitikai, biztonsági, vagy gazdasági megfontolásokat vonhatnak maguk után. Egy pár példa a forgalomirányítási korlátozásokra:

1. Ne legyen átmenő forgalom bizonyos AS-eken keresztül.
2. Soha ne helyezzük Irakot egy, a Pentagonnál kezdődő útvonalra.
3. Ne használjuk az Egyesült Államokat arra, hogy Brit Columbiából Ontarióba menjünk.
4. Csak akkor haladjunk át Albánián, ha nincs más út a célhoz.
5. Az IBM®-nél kezdődő vagy végződő forgalom ne menjen át a Microsofton®.

A politikákat kézzel konfigurálják minden BGP routerben. Ezek nem részei magának a protokollnak.

Egy BGP router szemszögéből a világ más BGP routerekből és az őket összekötő vonalakból áll. Két BGP routert összekötöttnek minősítünk, ha egy közös hálózaton osztoznak. A BGP átmenő forgalom iránti különleges érdeklődéséből adódóan, a hálózatok a következő három kategória közül az egyikbe tartoznak. Az első kategóriába a **csonka hálózatok (stub networks)** esnek, amelyeknek csak egy összeköttetésük van a BGP gráffal. Ezeket nem lehet az átmenő forgalom céljaira felhasználni, mivel nincs senki a másik oldalon. Ezután jönnek a **többszörösen bekötött hálózatok (multiconnected networks).** Ezeket használhatná az átmenő forgalom, csak éppen ők ezt megtagadják. Végül vannak az **átmenő hálózatok (transit network),** mint amilyenek a

gerinchálózatok, amelyek esetleg némi megkötésekkel készek kezelni egy harmadik fél csomagjait.

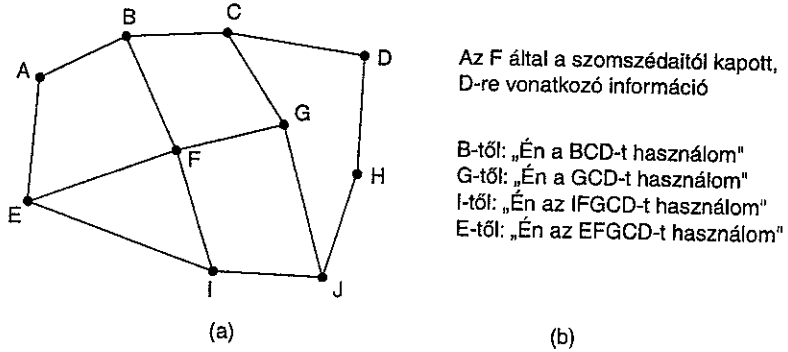
A BGP routerek páronként TCP összeköttetést létrehozva kommunikálnak egymással. Az ilyen működés megbízható kommunikációt biztosít és annak a hálózatnak a részleteit is elrejti, amelyiken áthalad.

A BGP alapvetően távolságvektor protokoll, de egészen eltér a legtöbb ilyentől, mint az RIP. Ahelyett, hogy csak a költségeket tartaná nyilván az egyes célokhoz irányítva, minden BGP router pontosan nyomon követi a használt útvonalat. Hasonlóan, ahelyett, hogy minden szomszédnak periodikusan megadná a becsült költséget minden lehetséges célig, minden egyes BGP router az általa használt pontos útvonalat mondja meg a szomszédainak.

Példának vegyük az 5.55.(a) ábrán látható BGP routereket. Pontosabban, vegyük *F* forgalomirányító táblázatát. Tegyük fel, hogy az *FGCD* utat használja, hogy eljusson *D*-hez. Amikor a szomszédok forgalomirányítási információt adnak át neki, a teljes útvonalaikat megadják, ahogy az 5.55.(b) ábra mutatja. (Az egyszerűség kedvéért csak a *D* célt mutatjuk.)

Miután minden szomszédtól beérkezett az útvonal-információ, *F* megvizsgálja, hogy ezek közül melyik a legjobb. Gyorsan elveti az *I* és az *E* útvonalát, mivel ezek áthaladnak magán *F*-en. Ezek után csak a *B* és a *G* között kell választani. Minden BGP router tartalmaz egy olyan modult, amely megvizsgálja az egy megadott célhoz vezető utakat, és pontozza őket, majd visszaad minden ezen cél felé vezető úthoz egy „távolság" értéket. Bármely olyan útvonal, amely egy politikai korlátozást sért, automatikusan végtelen értéket kap. A router ezután a legrövidebb távolságú útvonalat fogadja el. A pontozó függvény nem a BGP protokoll része, és bármilyen, a rendszermenedzserek által jónak tartott függvény lehet.

A BGP egyszerűen oldja meg a végtelenségig számolás problémáját, amely megfertőzte a többi távolságvektor alapú algoritmust. Példaként tegyük fel, hogy *G* összeomlik vagy az *FG* vonal megszakad. Ezután *F* a három megmaradt szomszédjától fog útvonalakat kapni. Ezek a *BCD*, *IFGCD* és *EFGCD* útvonalak lesznek. Azonnal láthatja, hogy a két utóbbi útvonal értelmetlen, mivel áthaladnak magán *F*-en, így



5.55. ábra. *(a) BGP routerek egy halmaza. (b) Az F-nek küldött információ*

FBCD-t választja új útvonalnak. Más távolságvektor alapú algoritmusok gyakran a rosszat választják, mert nem tudják, mely szomszédjuknak van független útvonala a célig és melyiknek nincs. A BGP mostani meghatározását az RFC 1654 tartalmazza. További hasznos információkat az RFC 1268 közöl.

5.5.7. Többesküldés az Interneten

A rendes IP kommunikáció egy adó és egy vevő közt zajlik. De néhány alkalmazás esetében hasznos, ha egy folyamat képes nagyszámú vevőnek egyszerre küldeni (multicasting). Ilyen például a többszörözött, elosztott adatbázisok frissítése, részvényárfolyamok átvitele több brókerhez, és a digitális konferencia (vagyis több résztvevős) telefonhívások kezelése.

Az IP támogatja a többesküldést a D osztályú címek használatával. Minden D osztályú cím egy hosztcsoportot azonosít. Huszonnyolc bit használható a csoportok azonosítására, így egy időben több mint 250 millió csoport létezhet. Amikor egy folyamat egy D osztályú címre küld egy csomagot, egy legjobb erőfeszítés típusú kísérlet történik arra, hogy a megcímzett csoport minden tagjának a csomag kézbesítésre kerüljön, de erre semmi garancia nincs. Egyes tagok esetleg nem kapják meg a csomagot.

Az IP kétfajta csoportcímet támogat: az állandó címeket és az ideiglenes címeket. Egy állandó csoport mindig létezik és nem kell felállítani. Minden állandó csoportnak egy állandó csoportcíme van. Néhány példa az állandó csoportokra:

224.0.0.1 Az egy LAN-on levő összes rendszer.
224.0.0.2 Az egy LAN-on levő összes router.
224.0.0.5 Az egy LAN-on levő összes OSPF router.
224.0.0.6 Az egy LAN-on levő összes kijelölt OSPF router.

Az ideiglenes csoportokat létre kell hozni, mielőtt használhatnánk őket. Egy folyamat megkérheti a hosztját, hogy csatlakozzon egy bizonyos csoporthoz. Arra is megkérheti a hosztját, hogy lépjen ki a csoportból. Amikor egy hoszt utolsó folyamata is elhagyja egy csoportot, az a csoport többé nem lesz jelen a hoszton. Minden hoszt nyilvántartja, hogy a folyamatai jelenleg milyen csoportokhoz tartoznak.

A többesküldést speciális többesküldéses routerek valósítják meg, amelyek vagy egybe vannak építve a szokásos routerekkel, vagy nem. Körülbelül percenként egyszer minden hoszt egy hardver (vagyis adatkapcsolati rétegbeli) többesküldést intéz a LAN-on levő hosztjaihoz (224.0.0.1-es cím), megkérve azokat, hogy jelentsék, milyen csoportokhoz tartoznak jelenleg a folyamataik. Minden hoszt az összes olyan D osztályú címre válaszol, amely érdekli.

Ezek a lekérdező és válasz csomagok egy **IGMP-nek (Internet Group Management Protocol – internet csoportfelügyeleti protokoll)** nevezett protokollt használnak, amely távolról hasonlít az ICMP-hez. Csak kétfajta csomagja van: lekérdezés és válasz. Mindkettő egyszerű, rögzített formátumú; az adat mező első szava némi vezérlőinformációt tartalmaz, a második szava pedig egy D osztályú címet. Ezt a protokollt az RFC 1112 írja le.

A többesküldés forgalomirányítása feszítőfák használatával történik. Minden többesküldéses router egy módosított távolságvektor protokollt használva cserél információt a szomszédaival, hogy minden csoporthoz felépítsenek egy feszítőfát, amely az összes csoporttagot lefedi. Változatos optimalizációk használatosak a fa csonkolásához, hogy kiküszöböljék az egyes csoportokban nem érdekelt routereket és hálózatokat. A protokoll nagyban kihasználja az alagút típusú átvitelt, hogy ne zavarja a nem a feszítőfában levő routereket.

5.5.8. Mobil IP

Az Internet sok felhasználójának van hordozható számítógépe, és akkor is összeköttetésben akarnak maradni az Internettel, amikor egy távoli Internet-helyet látogatnak meg, sőt még az utazás közben is. Sajnos az IP címzési rendszere miatt az otthontól távoli munkát könnyebb mondani, mint elvégezni. Ebben a szakaszban megvizsgáljuk a problémát és a megoldását. Egy részletesebb leírást ad (Johnson, 1995).

Az igazi intrikus maga a címzési séma. Minden IP cím három mezőt tartalmaz: az osztályt, a hálózatszámot és a hosztszámot. Vegyük példának a 160.80.40.20-as IP című gépet. A 160.80 megadja az osztályt (B) és a hálózatszámot (8272); a 40.20 a hosztszám (10260). A routereknek a világon mindenütt olyan forgalomirányító táblázataik vannak, amelyek megmondják nekik, hogy a 160.80-as hálózathoz melyik vonalat használva lehet eljutni. Valahányszor egy olyan csomag jön be, amelyiknek a célcíme 160.80.xxx.yyy formájú, az ezen a vonalon megy ki.

Ha egyszer csak az ezzel a címmel rendelkező gépet hirtelen elszállítjuk valami távoli helyre, a csomagjait továbbra is az otthoni LAN-jára (vagy routerjéhez) irányítják. A tulajdonos nem kap többet e-levelet és így tovább. Az, hogy a gépnek az új elhelyezkedésének megfelelő IP címet adunk, nem egy vonzó lehetőség, mivel nagyszámú emberrel, programmal és adatbázissal kellene tudatni a változást.

Más megközelítés az, hogy a routerek a teljes IP címet használják a forgalomirányításhoz, ne csak az osztályt és a hálózatot. Viszont ez a stratégia azt követelné meg, hogy minden routernek millió és millió táblázatbejegyzése legyen, ami az Internetre nézve csillagászati költséggel járna.

Amikor az emberek kezdték követelni, hogy lehessenek mozgó hosztjaik is, az IETF felállított egy munkacsoportot, hogy megoldást találjon. A munkacsoport gyorsan kialakított számos olyan célt, amelyet minden megoldásban kívánatosnak ítéltek. A főbb célok a következők voltak:

1. Minden mozgó hosztnak bárhol képesnek kell lennie az otthoni IP címe használatára.
2. A rögzített hosztokban nem engedélyezünk változtatásokat.
3. A router szoftverben és táblázatokban nem engedélyezünk változásokat.
4. A mozgó hosztokhoz menő legtöbb csomagnak nem szabad az úton kitérőket tennie.
5. Nem okozhat többletmunkát, amikor a mozgó hoszt otthon tartózkodik.

A választott megoldás az, amit az 5.2.8. részben leírtunk. Hogy röviden összefoglaljuk, minden helynek, amelyik lehetővé kívánja tenni a felhasználói számára a barangolást, egy hazai ügynököt kell létrehoznia. Minden helynek, amely látogatókat kíván megengedni, egy idegen ügynököt kell létrehoznia. Amikor egy mozgó hoszt megjelenik egy idegen helyen, kapcsolatba lép az ottani idegen hoszttal és bejegyezteti magát. Az idegen hoszt ezután kapcsolatba lép a felhasználó hazai ügynökével, és ad neki egy közvetett vagy **c/o címet (care-of address)**, rendszerint az idegen ügynök saját IP címét.

Amikor egy csomag megérkezik a felhasználó otthoni LAN-jára, valamilyen, a LAN-hoz csatlakozó routeren keresztül jön be. A router ekkor megpróbálja a szokásos módon meghatározni a hoszt helyét, egy ARP csomag szórása által, amely például azt kérdezi: „Mi a 160.80.40.20 Ethernet címe?" A hazai ügynök erre a kérdésre a saját Ethernet címének megadásával válaszol. Ezután a router a 160.80.40.20-nak szóló csomagokat a hazai ügynöknek küldi. Az ezeket egy alagúton keresztül elküldi az alcímre, beágyazván őket egy, az idegen ügynöknek címzett IP csomag adat mezejébe. Ezután az idegen ügynök ezeket kibontja és kézbesíti a mozgó hoszt adatkapcsolati címére. Ezen kívül a hazai ügynök megadja az alcímet az adónak, így a jövőben a csomagokat egyenesen az idegen ügynökhöz vihetik át egy alagúton keresztül. Ez a megoldás minden fentebb felsorolt követelménynek eleget tesz.

Egy kis részlet valószínűleg megérdemli, hogy megemlítsük. Amikor a mozgó hoszt mozog, a routernek valószínűleg a gyorstárában van a (rövidesen érvénytelenné váló) Ethernet címe. Hogy ezt felcseréljük a hazai ügynök Ethernet címével, egy **önkényes ARP-nek (gratuitous ARP)** nevezett trükköt használnak. Ez egy különleges, kéretlen üzenet a routernek, amely kicserélteti vele a gyorstár egy bizonyos bejegyzését, ebben az esetben a távozni készülő mozgó hosztét. Amikor a mozgó hoszt később visszatér, ugyanez a trükk használatos a router gyorstárának újbóli frissítésére.

A tervezésben semmi nincs, ami megtiltaná, hogy egy mozgó hoszt saját idegen ügynöke legyen, de ez a megközelítés csak akkor működik, ha a mozgó hoszt (idegen ügynök minőségében) logikailag kapcsolódik az Internethez a pillanatnyi helyén. Arra is képesnek kell lennie, hogy megszerezzen egy (ideiglenes) IP alcímet saját használatra. Ennek az IP címnek ahhoz a LAN-hoz kell tartoznia, amelyikhez pillanatnyilag kapcsolódik.

Az IETF mozgó hosztokra vonatkozó megoldása számos olyan problémát is megold, amelyet eddig nem említettünk. Például, hogyan találjuk meg az ügynököket? A megoldás az, hogy minden ügynök rendszeresen adatszórással közli a címét és a biztosítani kívánt szolgálatot (hazai, idegen vagy mindkettő). Amikor egy mozgó hoszt elérkezik valahova, egyszerűen figyelhet ezekre a **hirdetésnek (advertisement)** nevezett adatszórásokra. Alternatívaként adatszórással kiadhat egy csomagot, amely bejelenti az érkezését, és remélheti, hogy a helyi idegen ügynök válaszol rá.

Egy másik megoldandó probléma az, hogy mit tegyünk az udvariatlan mozgó hosztokkal, amelyek búcsúzás nélkül távoznak. A megoldás, hogy a bejegyzés csak egy rögzített időtartamig legyen érvényes. Ha nem frissítik rendszeresen, lejár, így az idegen hoszt kitisztíthatja a táblázatait.

Egy másik kérdés a biztonság. Amikor egy hazai ügynök egy olyan üzenetet kap, amely arra kéri, hogy Nóra minden csomagjait valamilyen IP címre továbbítsa, jobb,

ha nem engedelmeskedik, amíg meg nem győződött arról, hogy az üzenet forrása Nóra és nem olyasvalaki, aki csak megszemélyesíteni próbálja. Erre a célra kriptográfiai hitelesítő protokollokat használnak. Az ilyen protokollokat a 7. fejezetben fogjuk tanulmányozni.

Egy utolsó pont, amit a munkacsoport megcélzott, a mozgékonyság szintjeire vonatkozik. Képzeljünk el egy repülőgépet, fedélzetén egy Ethernettel, amit a navigációs és repülési számítógépek használnak. Ezen az Etherneten van egy rendes router, amely egy rádiókapcsolaton keresztül beszél a földön levő vezetékes Internettel. Egy szép napon valamelyik ravasz marketingfőnöknek az az ötlete támad, hogy minden karfába szereljenek egy Ethernet csatlakozót, és így a mobil számítógépekkel rendelkező utasok is bekapcsolódhatnak.

Most már két mobilitási szintünk van: a légi jármű saját számítógépei, amelyek az Ethernethez képest mozdulatlanok, és az utasok számítógépei, amelyek ehhez képest mobilok. Ezen kívül a fedélzeti router a földi routerhez képest mozog. Egy ilyen önmagában is mobil rendszeren belüli mobilitást, rekurzív alagút típusú átvitelekkel kezelhetünk.

5.5.9. Osztálynélküli körzetek közötti forgalomirányítás: CIDR

Az IP-t már több mint egy évtizede intenzíven használják. Nagyon jól működött, ahogy az Internet exponenciális növekedése is megmutatta. Sajnos az IP gyors tempóban lesz a saját népszerűségének áldozata: kezd kifogyni a címekből. Ez a fenyegető végzet sok tárgyalást és vitát szült az Internet közösségén belül arról, hogy mit is lehetne ezzel kapcsolatban tenni. Ebben a szakaszban mind a problémát, mind néhány javasolt megoldást tárgyalunk. Egy teljesebb leírás található (Huitema, 1996) könyvében.

1987-ben pár látnok megjósolta, hogy egy napon az Internet 100 000 hálózatig nőhet. A legtöbb szakértő ezt semmibe vette azzal, hogy ez évtizedekre van még, ha egyáltalán bekövetkezik. A 100 000. hálózatot 1996-ban kötötték be. Egyszerűen megfogalmazva, a gond az, hogy az Internet gyorsan fogy ki az IP címekből. Elméletben több mint 2 milliárd cím létezik, de a címtartomány osztályok szerinti szervezésének gyakorlata (l. az 5.47. ábrát) ebből milliókat elveszteget. Különösen a B osztályú hálózatok okozzák a gondot. A legtöbb szervezetnek egy 16 millió címmel rendelkező A osztályú hálózat túl nagy, és egy C osztályú hálózat, 256 címmel, túl kicsi. Egy 65 536 címmel rendelkező B osztályú hálózat pont jó lenne. Az Internet folklórjában ez a helyzet a **három medve problémája (three bears problem,** mint a hasonló című mesében) néven ismert.

A valóságban egy B osztályú cím nagyon nagy a legtöbb szervezetnek. Tanulmányok kimutatták, hogy az összes B osztályú hálózatnak több mint a fele kevesebb, mint 50 hoszttal rendelkezik. Egy C osztályú hálózat is megtette volna, de kétségkívül minden B osztályú címet kérő szervezet úgy gondolta, hogy egy napon kinőné a 8 bites hosztmezőt. Visszatekintve, lehet, hogy jobb lett volna, ha a C osztályú hálózatok nyolc helyett 10 bitet használtak volna a hosztszámra, amely hálózatonként 1022 hosztot tesz lehetővé. Ebben az esetben a legtöbb szervezet valószínűleg belenyugodott volna egy C osztályú címbe, és ezekből egy félmillió lehetne (szemben a csak 16 384 B osztályú hálózattal).

Viszont így egy másik gond merült volna gyorsabban fel: a forgalomirányító táblázatok robbanása. A routerek szemszögéből nézve az IP címtartomány egy kétszintes hierarchia, hálózatszámokkal és hosztszámokkal. A routereknek nem kell minden hosztot ismerniük, de minden hálózatot igen. Ha félmillió C osztályú hálózatot használnánk, az Interneten minden routernek olyan táblázatra lenne szüksége, amely félmillió bejegyzést tartalmaz, minden hálózathoz egyet, amely megmondja, melyik vonal használatával lehet eljutni ahhoz a hálózathoz, egyéb információk mellett.

A félmillió bejegyzéses táblázatok fizikai tárolása valószínűleg megoldható, bár költséges kritikus routereknél, amelyek a táblázatokat statikus RAM-ban, B/K kártyákon tartják. Egy komolyabb probléma, hogy a táblázatok kezeléséhez szükséges különféle algoritmusok komplexitása lineárisnál gyorsabban nő. Még rosszabb, hogy sok létező router szoftverét és firmware-jét akkor tervezték, amikor az Internethez 1000 hálózat csatlakozott és a 10 000 hálózat még évtizedekre levőnek tűnt. Az akkor hozott tervezési döntések ma már gyakran távolról sem optimálisak.

Ezen kívül különféle router algoritmusok megkövetelik minden routertől, hogy a táblázatait periodikusan átvigye. Minél nagyobbak a táblázatok, annál valószínűbb, hogy néhány rész útközben elveszik, és a másik végen nem teljes adatokhoz és esetleg forgalomirányítási instabilitáshoz vezet.

A forgalomirányító táblázat problémája megoldható lett volna mélyebb hierarchia használatával. Például, ha minden IP cím egy ország, állam, város, hálózat és hoszt mezőt is tartalmazna, az működne. Ekkor minden routernek csak azt kellene tudnia, hogyan jusson el minden egyes országba, a saját országán belüli államokba vagy megyékbe, az államában vagy megyéjében levő városokba, és a városán belüli hálózatokba. Sajnos ez a megoldás 32 bitnél számottevően többet igényelne az IP címek számára, és nem használná ki a címeket hatékonyan (Liechtensteinnek ugyanannyi bitje lenne, mint az Egyesült Államoknak).

Röviden, a legtöbb megoldás megold egy problémát, de létrehoz egy újat. Egy megoldás, amelyet most valósítanak meg, és amely egy kicsivel több lélegzethez juttatja az Internetet, a **CIDR (Classless InterDomain Routing, osztálynélküli körzetek közötti forgalomirányítás).** Az RFC 1519-ben leírt CIDR mögött az az alapötlet áll, hogy a maradék C osztályú hálózatokat, amelyekből majdnem kétmillió van, változó méretű blokkokban osszák ki. Ha egy helynek mondjuk 2000 címre van szüksége, egy 2048 címből álló blokkot kap (nyolc egybefüggő C osztályú hálózatot), és nem egy teljes B osztályú címet. Hasonlóan egy hely, amelyiknek 8000 címre van szüksége, 8192 címet kap (32 egybefüggő C osztályú hálózatot).

Azon kívül, hogy egységként egybefüggő C osztályú hálózatokból álló blokkokat használnak, az RFC 1519-ben a C osztályú címek kiosztási szabályait is megváltoztatták. A világot négy zónára darabolták fel, és mindegyik a C osztályú címtartomány egy részét kapta. A kiosztás a következő volt:

A 194.0.0.0-tól 195.255.255.255-ig terjedő címek Európához tartoznak.
A 198.0.0.0-tól 199.255.255.255-ig terjedő címek Észak-Amerikához tartoznak.
A 200.0.0.0-tól 201.255.255.255-ig terjedő címek Dél- és Közép-Amerikához tartoznak.

A 202.0.0.0-tól 203.255.255.255-ig terjedő címek Ázsiához és a Csendes-óceáni térséghez tartoznak.

Ily módon minden térség 32 millió kiosztható címet kapott, amellett, hogy másik 320 millió C osztályú címet 204.0.0.0-tól 223.255.255.255-ig tartalékoltak a jövőre nézve. Ennek a kiosztásnak az előnye, hogy most minden Európán kívüli router, amely egy 194.xx.yy.zz-nek vagy 195.xx.yy.zz-nek címzett csomagot kap, azt egyszerűen a szokásos európai átjárójához küldheti. Ezzel gyakorlatilag 32 millió címet sűrítettünk egy bejegyzésbe a forgalomirányító táblázatban. Hasonló a helyzet a többi térséggel.

Természetesen, ha már egy 194.xx.yy.zz csomag eljutott Európába, részletesebb forgalomirányító táblázatok kellenek. Egy lehetőség, hogy legyen a 194.0.0.xx -től 195.255.255.xx-ig terjedő hálózatokhoz 131 072 bejegyzés, de ez pontosan az a forgalomirányítótábla-robbanás, amelyet próbálunk elkerülni. Ehelyett minden bejegyzéshez hozzáveszünk egy 32 bites maszkot a forgalomirányító táblában. Amikor egy csomag bejön, először kicsomagoljuk a célcímét. Ezután (elméletileg) a forgalomirányító táblázatot bejegyzésről bejegyzésre végignézzük, a célcímet maszkolva és összehasonlítva a táblázattal, keresvén egy illeszkedést.

Hogy tisztázzuk ezt az összehasonlító folyamatot, vegyünk egy példát. Tegyük fel, hogy a Cambridge-i Egyetemnek 2048 címre van szüksége, és a 194.24.0.0-tól 194.24.7.255-ig terjedő címeket rendelik hozzá, a 255.255.248.0 maszkkal együtt. Következőnek az Oxfordi Egyetem kér 4096 címet. Mivel egy 4096 címből álló blokk egy 4096 bájtos határra kell kerüljön, nem kaphatja meg a 194.24.8.0-tól kezdődő címeket. Ehelyett a 194.24.16.0-tól 194.24.31.255-ig terjedő címeket kapja meg, a 255.255.240.0 maszkkal együtt. Most az Edinburghi Egyetem kér 1024 címet, és a 194.24.8.0-tól 194.24.11.255-ig terjedő címeket rendelik hozzá a 255.255.252.0 maszkkal együtt.

Most már Európában mindenütt felfrissítették a forgalomirányító táblázatokat három bejegyzéssel, amelyek mindegyike egy alapcímet és egy maszkot tartalmaz. Ezek a bejegyzések (binárisan):

Cím	11000010 00011000 00000000 00000000
Maszk	11111111 11111111 11111000 00000000
Cím	11000010 00011000 00010000 00000000
Maszk	11111111 11111111 11110000 00000000
Cím	11000010 00011000 00001000 00000000
Maszk	11111111 11111111 11111100 00000000

Most gondoljuk meg, mi történik akkor, amikor egy 194.24.17.4-nek címzett csomag jön be, amely binárisan

11000010 00011000 00010001 00000100

Először ezt logikai ÉS kapcsolatba hozzuk Cambridge maszkjával, és kapjuk a következőt:

11000100 00011000 00010000 00000000

Ez az érték nem illeszkedik Cambridge alapcímére, ezért az eredeti címet ÉS kapcsolatba hozzuk Oxford maszkjával, és a következőt kapjuk:

11000010 00011000 00010000 00000000

Ez az érték illeszkedik Oxford alapcímére, így a csomagot elküldjük az oxfordi routernek. A gyakorlatban a forgalomirányító bejegyzéseket nem sorban egymás után próbálják végig; indexelési trükköket használnak a keresés felgyorsítására. Az is lehetséges, hogy két bejegyzés is illeszkedik, amikor is az nyer, amelyiknek a maszkjában több 1 bit van. Végül, ugyanez az ötlet alkalmazható minden címre, nemcsak az új C osztályú címekre, így a CDIR-rel a régi A, B és C osztályú hálózatokat már nem használjuk a forgalomirányításhoz. Ezért hívják a CDIR-t osztály nélküli forgalomirányításnak. A CDIR-t részletesebben (Ford és mások, 1993; Huitema, 1995) írják le.

5.5.10. IPv6

Bár a CDIR még nyerhet egy pár évet, mindenki világosan látja, hogy az IP napjai a jelenlegi formájában (IPv4) meg vannak számlálva. Ezeken a technikai problémákon kívül bujkál egy másik kérdés is a háttérben. Egészen a közelmúltig az Internetet nagyrészben az egyetemek, a high-tech ipar, és a kormányzat (különösen a Honvédelmi Minisztérium) használták. Az Internet iránti érdeklődésben a 90-es évek közepén bekövetkezett robbanás miatt valószínű, hogy a következő évezredben az emberek sokkal nagyobb csoportja fogja használni, különösen más igényekkel rendelkező emberek. Egyfelől emberek milliói, akik vezeték nélküli hordozható számítógépekkel rendelkeznek, használhatják arra, hogy kapcsolatban maradjanak az otthoni bázisukkal. Másfelől a számítógép, távközlési és szórakoztató iparágak közelgő konvergenciája miatt lehet, hogy nemsokára a világon minden televíziókészülék egy Internet-csomópont is lesz, amely milliárdnyi hálózati videózásra használt gépet eredményez. Ezen körülmények miatt nyilvánvalóvá vált, hogy az IP-nek fejlődnie kell, és rugalmasabbá kell válnia.

Látván, hogy ezek a problémák feltűnnek a horizonton, 1990-ben az IETF elkezdte a munkát az IP egy új verzióján, egy olyanon, amely soha nem fogy ki címekből, mindenféle egyéb problémákat is megold, és ezek mellett rugalmasabb és hatékonyabb is. A fő célok a következők voltak:

1. Hosztok milliárdjainak támogatása, még nem hatékony címtartomány-hozzárendelés árán is.

2. A forgalomirányító táblázatok méretének csökkentése.

3. A protokoll egyszerűsítése, lehetővé téve ezzel a routereknek a csomagok gyorsabb feldolgozását.

4. A jelenlegi IP-nél jobb biztonság (hitelesítés és titkosság) biztosítása.

5. Nagyobb figyelem szentelése a szolgálat típusának, különösen a valós idejű adatoknál.

6. A többesküldés segítése, hatósugarak megadásának lehetővé tételével.

7. Lehetőség arra, hogy egy hoszt a címének megváltoztatása nélkül barangoljon.

8. A protokoll fejlődésének lehetővé tétele.

9. Az új és a régi protokoll még évekig egymás mellett létezhessen.

Az IETF egy javaslatokra és vitára való felhívást tett közzé az RFC 1550-ben, hogy egy olyan protokollt találjon, amely mindezen követelményeknek eleget tesz. Huszonegy válasz érkezett, közülük nem mind volt teljes javaslat. 1992 decemberére hét komoly javaslat feküdt az asztalon. Ezek az IP kisebb módosításaitól kezdve addig terjedtek, hogy az egészet ki kellene dobni és egy teljesen másmilyen protokollal helyettesíteni.

Egy javaslat az volt, hogy a TCP-t a CLNP felett kellene futtatni, amely a 160 bites címeivel mindörökre elegendő címtartományt biztosított volna, és egyesített volna két fő hálózati rétegbeli protokollt. De sok ember úgy érezte, hogy ez annak lett volna az elismerése, hogy az OSI világban tulajdonképpen valamit jól csináltak, ez az állítás pedig politikailag helytelennek minősül Internet-körökben. A CLNP-t közvetlenül az IP-ről mintázták, így a kettő valójában nem különbözik annyira egymástól. Tulajdonképpen a végül is kiválasztott protokoll sokkal jobban különbözik az IP-től, mint a CLNP. Egy másik csapás a CLNP-re az volt, hogy gyatrán támogatja azokat a szolgálattípusokat, amelyek a multimédia hatékony átviteléhez szükségesek.

A jobb javaslatok közül hármat közölt az IEEE Network (Deering, 1993; Francis, 1993; Katz és Ford, 1993). Sok vita, módosítás és pozícióharc után kiválasztották a Deering- és a Francis-féle javaslatok egy módosított kombinációját, amelyet ekkor már **SIPP-nek (Simple Internet Protocol Plus)** hívtak, és az **IPv6** jelölést adták neki. (Az IPv5-öt már egy kísérleti valós idejű folyamprotokollhoz használták.)

Az IPv6 egészen jól megfelel a célnak. Megtartja az IP jó tulajdonságait, elveti vagy kevésbé hangsúlyossá teszi a rosszakat, és újakat is hozzáad, ahol szükség van rá. Általánosságban, az IPv6 nem kompatíbilis az IPv4-gyel, de az összes többi Internet protokollal igen, beleértve a TCP, UDP, ICMP, IGMP, OSPF, BGP és DNS protokollokat, néhol úgy, hogy kisebb módosításokra van szükség (főleg a hosszabb címek kezelése miatt). Alább tárgyaljuk az IPv6 főbb tulajdonságait. További információ található az RFC 1883–1887-ben.

Először, ami a legfontosabb, az IPv6-nak hosszabb címei vannak, mint az IPv4-nek. 16 bájt hosszúak, amely megoldja azt a problémát, amelyet az IPv6-nak meg kell:

egy gyakorlatilag végtelen Internet címellátmányt biztosít. Nemsokára több mondanivalónk is lesz a címekről.

Az IPv6 második fő fejlesztése a fejrész egyszerűsítése. Csak 7 mezőt tartalmaz (ezzel szemben az IPv4 13-at). Ez a változás lehetővé teszi a routereknek, hogy gyorsabban dolgozzák fel a csomagokat és ezáltal javítsák az átbocsátást. A fejrészt is rövidesen megtárgyaljuk.

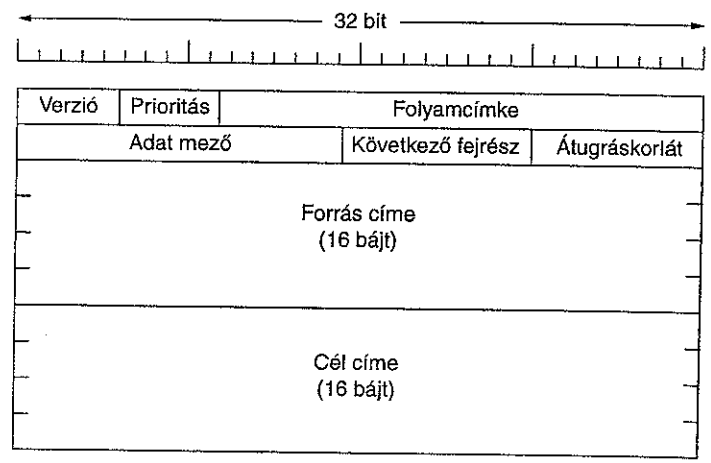
A harmadik fő fejlesztés az opciók jobb támogatása. Ez a változás szükségszerűen együtt jár az új fejrésszel, mert korábban megkövetelt mezők most opcionálisak lettek. Ezen kívül az opciók megjelenésének a módja is más, így a routereknek egyszerű átlépni a nem nekik szánt opciókon. Ez a tulajdonság a csomagfeldolgozási időt gyorsítja fel.

Egy negyedik terület, ahol az IPv6 jelentős előrelépést jelent, a biztonság. Az IETF-nek elege volt azokból az újságtörténetekből, ahol koraérett 12 évesek a személyi számítógépeikkel bankokba és katonai bázisokba törnek be az Interneten. Erősen érezhető volt, hogy valamit kell tenni a biztonság javítása érdekében. A hitelesítés és titkosság az új IP kulcstulajdonságai.

Végül, több figyelmet szenteltek a szolgálat típusának, mint a múltban. Az IPv4-nek tulajdonképpen van egy erre a célra szánt 8 bites mezője, de a multimédia forgalom várható jövőbeli növekedése miatt ennél sokkal többre van szükség.

A fő IPv6 fejrész

Az IPv6 fejrész az 5.56. ábrán látható. A *Verzió* mező IPv6-nál mindig 6 (és az IPv4-nél mindig 4). Azalatt az idő alatt, amíg átállnak az IPv4-ről, ami lehet, hogy egy évtizedet fog igénybevenni, a routerek megvizsgálhatják ezt a mezőt, hogy eldöntsék, mi



5.56. ábra. *A rögzített IPv6 fejrész (kötelező)*

lyen fajta csomagjuk van. Viszont ez mellékhatásként elveszteget pár utasítást a kritikus úton, így valószínűleg sok megvalósítás meg fogja próbálni ezt elkerülni, és az adatkapcsolati fejrészben fog használni valamilyen mezőt arra, hogy megkülönböztesse az IPv4 csomagokat az IPv6 csomagoktól. Ily módon a csomagokat közvetlenül a megfelelő hálózati rétegnek adhatják át. Viszont az, hogy az adatkapcsolati réteg tudjon a hálózat csomagtípusáról, teljesen megszegi azt a tervezési elvet, miszerint egyik réteg sem tudhat a felette levő réteg által átadott bitek jelentéséről. A „Csináljuk helyesen" és „Legyen gyors" táborok közt kétségkívül hosszú és parázs viták lesznek.

A *Prioritás* mező használatos azon csomagok közti különbségtételre, amelyeknek a forrásuk képes forgalomszabályozásra, és amelyeknek nem. A 0-tól 7-ig terjedő értékek azon átvitelekhez tartoznak, amelyek képesek torlódás esetén lassítani. A 8-tól 15-ig terjedő értékek az olyan valós idejű forgalomhoz tartoznak, amelyiknek a küldési sebessége állandó, még ha minden csomag el is veszik. A hang és a mozgókép az utóbbi kategóriába tartozik. Ez a megkülönböztetés lehetővé teszi a routerek számára, hogy a csomagokat jobban tudják kezelni torlódás esetén. Mindkét csoporton belül a kisebb számozású csomagok kevésbé fontosak, mint a nagyobb számozásúak. Az IPv6 szabvány például az 1-et a hírekhez, a 4-et az FTP-hez és a 6-ost a Telnethez javasolja, mivel az nem észrevehető, ha egy híreket tartalmazó csomagot pár másodpercig késleltetünk, de az biztosan, ha egy Telnet csomaggal tesszük ugyanezt.

A *Folyamcímke* mező még mindig kísérletek tárgya, de arra lehet majd használni, hogy egy forrás és egy cél felállíthasson egy álösszeköttetést, bizonyos tulajdonságokkal és igényekkel. Például egy bizonyos hoszt bizonyos folyamatától egy bizonyos célhoszt bizonyos folyamataig tartó csomagfolyamnak szigorú késleltetési igényei lehetnek, és ezért fenntartott sávszélességre van szüksége. A folyamot előre fel lehet állítani, és egy azonosítót adni neki. Amikor egy nem nulla *Folyamcímke* mezőjű csomag tűnik fel, minden router kikeresheti a belső táblázataiból, hogy milyen különleges elbánást igényel. Tulajdonképpen a folyamok bevezetése egy kísérlet arra, hogy mind a datagram alapú alhálózat rugalmassága, mind a virtuális áramkör alapú alhálózatok garanciái együtt legyenek.

Minden folyamot a forrás címe, a cél címe és a folyamszám azonosít, így sok folyam lehet egyidőben aktív két adott IP cím közt. Ilyen módon, ha más hosztoktól jövő, de ugyanolyan folyamszámmal rendelkező folyamok ugyanazon a routeren haladnak keresztül, a router meg tudja őket különböztetni a forrás- és célcímeket használva. Várhatóan a folyamszámokat véletlenszerűen fogják választani, ahelyett, hogy 1-től kezdve folyamatosan osztanák ki azokat, hogy a routerek könnyen tudják azokat hash-elni.

Az *Aadat mező hossza* mező megmondja, mennyi bájt következik az 5.56. ábra 40 bájtos fejrésze után. A név megváltozott az IPv4 *Teljes hossz* mezejéhez képest, mivel a jelentés is módosult: a 40 fejrészbájtot már nem számolják bele a hosszba, mint régebben.

A *Következő fejrész* mező kiengedi a zsákbamacskát. A fejrészt azon oknál fogva lehetett egyszerűsíteni, mert lehetnek további (opcionális) kiegészítő fejrészek. Ez a mező mondja meg, melyik kiegészítő fejrész következik a (jelenleg) hat közül, ha egyáltalán van ilyen. Ha a fejrész az utosó IP fejrész, a *Következő fejrész* mező azt mondja meg, melyik szállítási protokoll kezelőjének (TCP, UDP) kell a csomagot továbbítani.

Az *Átugráskorlát* mező gátolja meg a csomagokat abban, hogy azok örökké élhessenek. Ez gyakorlatilag ugyanaz, mint az *Élettartam* mező az IPv4-ben, vagyis egy olyan mező, amelyet minden átugrásnál csökkentenek. Elméletben az IPv4-ben ez idő volt másodpercekben mérve, de egy router sem használta így, ezért a nevet megváltoztatták, hogy arra utaljon, ahogy valójában azt használják.

Következnek a *Forrás címe* és *Cél címe* mezők. Deering eredeti javaslata, az SIP, 8 bájtos címeket használt, de a felülvizsgálási folyamat során sok ember érezte úgy, hogy 8 bájtos címekkel az IPv6 néhány évtizeden belül ki fog fogyni a címekből, míg a 16 bájtos címek soha nem fogynak el. Más emberek érvelése szerint a 16 bájt túlzás, megint mások pedig 20 bájtos címeket részesítettek volna előnyben, hogy az IPv6 kompatíbilis legyen az OSI protokollal. Egy másik frakció változó hosszúságú címeket akart. Sok vita után úgy határoztak, hogy a legjobb kompromisszum a rögzített hosszúságú 16 bites címek esete.

Az IPv6 címtartományt az 5.57. ábrán látható módon osztották fel. A 80 nullával

Előtag (binárisan)	Használat	Felhasznált törtrész
0000 0000	Fenntartott (beleértve az IPv4-et is)	1/256
0000 0001	Nem kiosztott	1/256
0000 001	OSI NSAP címek	1/128
0000 010	Novell NetWare IPX címek	1/128
0000 011	Nem kiosztott	1/128
0000 1	Nem kiosztott	1/32
0001	Nem kiosztott	1/16
001	Nem kiosztott	1/8
010	Szolgáltató alapú címek	1/8
011	Nem kiosztott	1/8
100	Földrajzi alapú címek	1/8
101	Nem kiosztott	1/8
110	Nem kiosztott	1/8
1110	Nem kiosztott	1/16
1111 0	Nem kiosztott	1/32
1111 10	Nem kiosztott	1/64
1111 110	Nem kiosztott	1/128
1111 1110 0	Nem kiosztott	1/512
1111 1110 10	Címek kapcsolat lokális használatára	1/1024
1111 1110 11	Címek helyszín lokális használatára	1/1024
1111 1111	Többesküldés	1/256

5.57. ábra. *Az IPv6 címek*

kezdődő címeket fenntartották az IPv4 címek számára. Ezekből két változatot is támogatnak, amelyeket a következő 16 bit különböztet meg. Ezek a változatok ahhoz kapcsolódnak, hogy hogyan viszik át az IPv6 csomagokat a létező IPv4 infrastruktúrán alagút segítségével.

Az, hogy a szolgáltató alapú és földrajzi alapú címekhez külön előtagot használnak, az Internet jövőjéről szóló két eltérő látomás közti kompromisszum eredménye. A szolgáltató alapú címeknek akkor van értelme, ha azt gondoljuk, hogy a jövőben néhány társaság fogja az Internet-szolgáltatást az ügyfeleinek biztosítani hasonlóan ahhoz, ahogy most az AT&T, az MCI, a Sprint, a British Telecom stb. biztosítják a telefonszolgáltatást. Ezen társaságok közül mindegyik a címtartomány egy töredékét fogja megkapni. A 010 előtag után következő első 5 bit jelzi, hogy melyik nyilvántartóból kell kikeresni a szolgáltatót. Ma három nyilvántartás működik: az észak-amerikai, az európai és az ázsiai. Legfeljebb 29 új nyilvántartást lehet később hozzáadni.

Minden nyilvántartás úgy osztja fel a maradék 15 bájtot, ahogy jónak látja. A várakozások szerint sokuk egy hárombájtos szolgáltatószámot fog használni, amely 16 millió szolgáltatót ad ki, és ezáltal a nagy társaságok saját szolgáltatójukként működhetnek. Egy másik lehetőség, hogy használjunk 1 bájtot a nemzeti szolgáltatók jelzésére, és engedjük, hogy a további kiosztást ők végezzék. Ezen a módon további hierarchiaszinteket vezethetnek be, ha erre szükség van.

A földrajzi modell ugyanolyan, mint a mostani Internet, ahol a szolgáltatók nem játszanak nagy szerepet. Így az IPv6 mindkét fajta címet kezelni tudja.

A kapcsolat és hely lokális címeinek csak helyi jelentőségük van. Ezeket minden szervezet ütközés nélkül újrahasznosíthatja. Ezek nem terjedhetnek szervezeti határokon túlra, ezért jól illeszkednek azon szervezetekhez, amelyek jelenleg tűzfalak használatával szigetelik el magukat az Internettől.

A többesküldéses címeknél egy 4 bites jelzőmező és egy 4 bites érvényességi kör mező következik az előtag után, majd egy 112 bites csoportazonosító. Az egyik jelzőbit különbözteti meg az állandó csoportokat az átmenetiektől. Az érvényességi kör mező lehetővé teszi, hogy az aktuális kapcsolatra, helyre, szervezetre vagy bolygóra korlátozzuk a többesküldést. Ez a négy érvényességi kör elszórva helyezkedik el a 16 értéken, hogy később további érvényességi körök is hozzávehetők legyenek. Például a bolygóra vonatkozó érvényességi kör a 14-es, így a 15-ös kód hozzáférhető az Internet más bolygókra, naprendszerekre és galaxisokra történő kiterjesztése esetén.

A szokásos egyesküldéses (kétpontos) és többesküldéses címek támogatása mellett az IPv6 egy újfajta címzést is támogat: a bárhova küldést. A **bárhova küldés (anycasting)** annyiban hasonlít a többesküldésre, hogy a célcím itt is címek egy csoportja, de ahelyett, hogy mindegyiküknek megpróbálná kézbesíteni a csomagot, csak az egyiknek próbálja meg kézbesíteni, azt rendszerint a legközelebbinek. Például, amikor egy kliens együttműködő állományszolgáltatók egy csoportjával lép kapcsolatba, a bárhova küldést használhatja, hogy a legközelebbit elérje anélkül, hogy tudná, melyik is az. A bárhova küldés rendes egyesküldéses címeket használ. A router rendszerre van bízva az, hogy kiválassza azt a szerencsés hosztot, amelyik megkapja a csomagot.

Egy új jelölésrendszert is kifejlesztettek a 16 bájtos címek leírásához. Nyolc, négy-négy hexadecimális számjegyből álló csoportként írjuk őket, a csoportok között kettősponttal, valahogy így:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Mivel sok cím sok nullát fog tartalmazni, három ésszerűsítést hitelesítettek. Először is, egy csoporton belül a bevezető nullák elhagyhatók, így a 0123 123-nak írható. Másodszor, egy vagy több 16 nullából álló csoport két kettősponttal helyettesíthető. Így a fenti címből a következő lesz:

8000::123:4567:89AB:CDEF

Végül, az IPv4 címek két kettőspont és a régi pontokkal elválasztott decimális szám formájában írhatók fel, például:

::192.31.20.46

Talán szükségtelen ennyire kihangsúlyozni, de sok 16 bájtos szám van. Egész pontosan 2^{128} darab van belőlük, amely körülbelül 3×10^{38}. Ha az egész Föld, szárazföld és víz is, számítógépekkel lenne befedve, az IPv6 7×10^{23} IP címet tenne lehetővé négyzetméterenként. A vegyész hallgatók észre fogják venni, hogy ez a szám nagyobb az Avogadro-számnál. Bár nem az volt a szándék, hogy a Föld felszínén minden molekulának saját IP címet adjunk, nem járunk annyira messze tőle.

A gyakorlatban a címtartomány nem lesz hatékonyan kihasználva, ahogy a telefonszámok címtartománya sem. (A manhattani területkód, 212, majdnem tele van, de a wyomingi, 307, majdnem üres). Az RFC 1715-ben Huitema kiszámolta, hogy a telefonszámok kiosztását irányadónak véve még a legpesszimistább forgatókönyv szerint is bőven több mint 1000 IP cím jut a Föld felszínének (szárazföld és víz egyaránt) minden négyzetméterére. Bármelyik valószínű forgatókönyv szerint trilliók jutnak egy négyzetméterre. Röviden, valószínűtlennek tűnik, hogy a belátható jövőn belül elfogynának. Érdemes azt is észrevenni, hogy a címtartománynak csak 28 százalékát osztották ki eddig. A többi 72 százalék elérhető a még fel sem merült jövőbeni célok számára.

Tanulságos összehasonlítani az IPv4 fejrészt (5.45. ábra) az IPv6 fejrésszel (5.56. ábra), hogy lássuk, mi maradt ki az IPv6-ból. Az *IHL* mező eltűnt, mert az IPv6 fejrész rögzített hosszúságú. A *Protokoll* mezőt is kivették, mert a *Következő fejrész* mező megmondja, mi jön az utolsó IP fejléc után (mármint UDP vagy TCP szegmens).

Minden, a darabolással kapcsolatos mezőt eltávolítottak, mert az IPv6 másképp közelíti meg a darabolást. Először is, minden, az IPv6-hoz igazodó hosztnak és routernek támogatnia kell 576 bájtos csomagokat. Ez a szabály eleve kevésbé valószínűvé teszi azt, hogy darabolás történik. Ezen kívül, amikor egy hoszt egy túl nagy IPv6 csomagot küld, az a router, amely képtelen azt továbbítani, darabolás helyett egy hibaüzenetet küld vissza. Ez az üzenet arra utasítja a hosztot, hogy minden ehhez a célhoz tartó csomagot törtdeljen fel a jövőben. Az, hogy a hoszt eleve jó méretű csomagokat küld, végül is sokkal hatékonyabb, mint ha a routerek darabolják azokat menet közben.

Végül az *Ellenőrző összeg* mező eltűnt, mert ennek a kiszámítása nagyban csökkenti a teljesítményt. A ma használt hálózatok megbízhatósága és azon tény mellett, hogy az adatkapcsolati és a szállítási rétegeknek rendszerint megvan a maguk ellenőr-

ző összege, még egy ellenőrző összeg értéke nem érte meg azt az árat, amelyet teljesítményben elvont. Mindezeknek a tulajdonságoknak az eltávolítása egy egyszerű és nagyszerű hálózati protokollt eredményezett. Ezáltal az IPv6 ezzel a tervezéssel elérte a céljait: egy gyors, mégis rugalmas protokoll lett, bőséges címtartománnyal.

Kiegészítő fejrészek

A hiányzó mezők némelyikére azért továbbra is szükség van, így az IPv6 bevezette az (opcionális) **kiegészítő fejrész (extension header)** fogalmát. Ezek a fejrészek használhatóak hatékony módon kódolt külön információ biztosítására. Jelenleg a kiegészítő fejrészeknek hat típusa definiált, ezeket az 5.58. ábra sorolja fel. Mindegyik opcionális, de ha több mint egy van jelen, akkor közvetlenül a rögzített fejléc után kell szerepelniük, és célszerűen a megadott sorrendben.

Kiegészítő fejrész	Leírás
Átugrás opciók	Különféle információk a routerek számára
Forgalomirányítás	A teljes vagy részleges követendő útvonal
Darabolás	A datagramdarabok kezelése
Hitelesítés	Az adó személyazonosságának ellenőrzése
Titkosított biztonsági hasznos teher	Információ a titkosított tartalomról
Célopciók	További információk a cél számára

5.58. ábra. *Az IPv6 kiegészítő fejrészei*

Némelyik fejrésznek kötött a formátuma; mások változó számú és hosszúságú mezőt tartalmaznak. Ez utóbbiaknál minden tétel egy (Típus, Hossz, Érték) hármasként van kódolva. A *Típus* egy egybájtos mező, amely azt mondja meg, hogy melyik opcióról van szó. A *Típus* értékeket úgy választották meg, hogy az első 2 bit megmondja a routernek, mit tegyen, ha nem tudja feldolgozni a csomagot. A lehetőségek: átugorni az opciót, eldobni a csomagot, eldobni a csomagot és visszaküldeni egy ICMP csomagot, ugyanaz, mint az előző, azzal a különbséggel, hogy a többesküldéses címekre ne küldjön ICMP csomagot (hogy egy rossz csomag ne generáljon több millió ICMP jelentést).

A *Hossz* is egy egybájtos mező. Azt mondja meg, milyen hosszú az érték (0 és 255 bájt közt). Az *Érték* 255 bájt erejéig bármilyen kívánt információ lehet.

Az ugrásról ugrásra (hop-by-hop) fejrészt olyan információk használják, amelyeket minden útba eső routernek meg kell vizsgálnia. Eddig egy ilyen opciót definiáltak, a

Következő fejrész	0	194	0
Jumbogram adat mezejének hossza			

5.59. ábra. *Az ugrásról ugrásra kiegészítő fejrész nagy datagramokhoz (jumbogramokhoz)*

64 K-nál nagyobb datagramok támogatásához. Ennek a fejrésznek a formátuma látható az 5.59. ábrán.

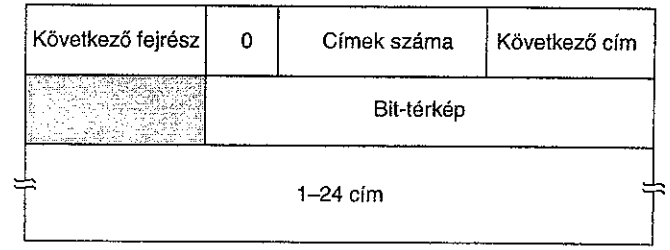
Ahogy minden kiegészítő fejrész, ez is egy olyan bájttal kezdődik, amely megmondja, milyen lesz a következő fejrész. Ezt a bájtot egy másik követi, amely megmondja, milyen hosszú az ugrásról ugrásra fejrész bájtokban, leszámítva az első 8 kötelező bájtot. A következő két bájt jelzi, hogy ez az opció a datagram méretét határozza meg (194-es kód) 4 bájtos szám formájában. Az utolsó 4 bájt adja meg a datagram méretét. A 65 536-s bájtnál kisebb méretű csomagok nem engedélyezettek. Ha előfordul, azt eredményezi, hogy az első router eldobja a csomagot és visszaküld egy ICMP hibaüzenetet. Az ezt a fejrész-kiegészítést használó datagramokat **jumbogramoknak** nevezik. A jumbogramok használata fontos a szuperszámítógépes alkalmazások számára, amelyeknek gigabájt nagyságrendű adatokat kell hatékonyan átvinni az Interneten.

A router fejrész egy vagy több routert sorol fel, amelyeket a célhoz vezető úton fel kell keresni. Mind a szigorú forgalomirányítás (az egész út meg van adva), mind a laza forgalomirányítás (csak bizonyos routereket adnak meg) lehetséges, de a kettőt összekombinálták. A router fejrész formátuma az 5.60. ábrán látható.

A router kiegészítő fejrész első 4 bájtja négy 1 bájtos egész számot tartalmaz: a következő fejrész típusát, a forgalomirányítás típusát (jelenleg 0), az ebben a fejrészben jelenlevő címek számát (1 és 24 közt) és a következő felkeresendő cím indexét. Ez utóbbi mező 0-ról indul, és minden egyes cím felkeresésénél eggyel növekszik.

Ezután egy fenntartott bájt következik, majd egy bit-térkép, amelyben minden ezt követő lehetséges IPv6 címhez egy bit tartozik. Ezek a bitek mondják meg, hogy mindegyik címet közvetlenül az azt megelőző után kell felkeresni (szigorú forrás általi forgalomirányítás), vagy más routerek közbejöhetnek (laza forrás általi forgalomirányítás).

A darabolási kiegészítő fejrész a darabolást hasonlóan kezeli, mint az IPv4. A fejrészben szerepel a datagramazonosító, a darabszám és egy bit, amely azt mondja meg, hogy jön-e még további darab. Az IPv6-ban, az IPv4-től eltérően csak a forráshoszt darabolhat fel egy csomagot. Bár ez egy nagyobb elgondolásbeli törést okoz a múlthoz képest, viszont egyszerűsíti a routerek munkáját és meggyorsítja a forgalomirányítást. Ahogy fentebb említettük, ha egy router olyan csomaggal találja magát szembe, amely túl nagy, eldobja a csomagot, és visszaküld egy ICMP csomagot a forrásnak. Ez az információ lehetővé teszi a forráshoszt számára, hogy ezt a fejrészt használva kisebb részekre darabolja a csomagot, és újra próbálkozzon.



5.60. ábra. *A kiegészítő fejrész forgalomirányításhoz*

A hitelesítő kiegészítő fejrész egy mechanizmust biztosít, ami által a csomag vevője biztos lehet abban, hogy a csomagot az küldte, aki a címben szerepel. Az IPv4-ben nincs ilyen garancia. A titkosított biztonsági adat mező lehetővé teszi, hogy egy csomag tartalmát úgy titkosítsuk, hogy csak a szándékaink szerinti vevő tudja elolvasni. Ezek a fejrészek titkosítási technikákat használnak küldetésük elvégzéséhez. Alább röviden leírjuk ezeket, de a modern titkosításban nem járatos olvasók esetleg nem fogják megérteni a teljes leírást. Ők a titkosítási protokollokkal foglalkozó 7. fejezet (és különösen a 7.1-es rész) elolvasása után térhetnek vissza ide.

Amikor egy adó és egy vevő biztonságosan akarnak kommunikálni, először meg kell egyezniük egy vagy több titkos kulcsban, amelyet csak ők ismernek. Hogy ezt hogyan intézik el, az kívül esik az IPv6 hatáskörén. Minden ilyen kulcshoz hozzárendelnek egy egyedi 32 bites kulcsszámot. A kulcsszámok globálisak, így ha Alice egy 4. kulcsot használ, hogy Bobbal beszéljen, akkor nem lehet egy másik 4. kulcsa, amelyikkel Carollal beszél. Minden kulcsszámhoz más paraméterek is tartoznak, mint pl. a kulcs élettartama és így tovább.

Hogy egy hitelesített üzenetet küldjön, az adó először összeállítja a csomagot az összes IP fejrészből és az adat mezőből, ezután az útközben változó mezőket (mint az *Ugráskorlát*) nullákkal helyettesíti. A csomagot ezután nullákkal tölti ki 16 bájt többszörösére. Hasonlóan a használandó titkos kulcsot is nullákkal töltik ki a 16 bájt többszörösére. Most egy titkosítási ellenőrző összeget számítanak ki a töltelékes titkos kulcs, a töltelékes csomag, és ismét a töltelékes titkos kulcs egymás után fűzött bit-sorozata alapján. A felhasználók definiálhatnak saját titkosítási algoritmusokat, de a titkosításban járatlan felhasználóknak inkább az alapértelmezett algoritmus, az MD5 javasolt.

Most érkeztünk el a hitelesítési fejrész szerepéhez. Alapvetően három részből áll. Az első rész 4 bájtból áll, és a következő fejrész számát, a hitelesítő fejrész hosszát és 16 nullás bitet tartalmaz. Ezután jön a 32 bites kulcsszám. Végül az MD5 (vagy más) ellenőrző összeget is beírják.

A vevő ezután a kulcsszámot használja, hogy megkeresse a titkos kulcsot. Ezután ennek a töltelékes változatát a töltelékes adat mező elé és mögé írják, a változó fejrész mezőket kinullázzák, és kiszámítják az ellenőrző összeget. Ha ez megegyezik a hitelesítő fejrészben szereplő ellenőrző összeggel, akkor biztos lehet benne, hogy a csomag attól az adótól jött, akivel megosztotta a titkos kulcsot, és afelől is biztos lehet, hogy a csomagot nem változtatták meg útközben. Az MD5 tulajdonságai a számításigény szempontjából lehetetlenné teszik, hogy egy támadó felvegye a küldő személyazonosságát, vagy a csomagot úgy módosíthassa, hogy azt ne vegyék észre.

Fontos megjegyezni, hogy egy hitelesített csomag adat mezeje titkosítatlanul kerül elküldésre. Bármelyik útba eső router elolvashatja, hogy mi van benne. Sok alkalmazásnak a titkosság nem igazán fontos, csak a hitelesítés. Például ha egy felhasználó utasítja a bankját, hogy fizesse ki a telefonszámláját, akkor valószínűleg nincs igazán szükség titkosságra, de a banknak nagyon is valós igénye, hogy teljes biztonsággal tudja, ki küldte a fizetési meghagyást tartalmazó csomagot.

Az olyan csomagoknál, amelyeket titkosan kell küldeni, a titkosított biztonsági adat mező kiegészítő fejrészét használják. Ez egy 32 bites kulcsszámmal kezdődik, amit a titkosított hasznos teher követ. A titkosítási algoritmus az adóra és a vevőre van

bízva, de alapértelmezésben ez a DES, a titkosított blokkokat egymás után láncoló üzemmódban. Amikor ezt a DES-CBC-t használják, az adat mező egy inicializáló vektorral kezdődik (4 bájt többszöröse), majd jön az adat mező, majd töltelék 8 bájt többszörösére. Ha a titkosítás és a hitelesítés is kívánatos, mindkét kiegészítő fejrészre szükség van.

A célopciók kiegészítő fejrészt olyan mezőkhöz szánták, amelyeket csak a célhosztban kell értelmezni. Az IPv6 kezdeti verziójában az egyetlen definiált opció a nulla opció, hogy ezt a fejrészt is kitöltsék 8 bájt többszörösére, így ezt kezdetben nem fogják használni. Azért került bele, hogy az új router és hoszt szoftverek biztosan tudják kezelni, arra az esetre, ha valaki egy napon kigondolna egy célopciót.

Viták az IPv6-tal kapcsolatban

Tudván a nyílt tervezési folyamatról és arról, hogy sok érintett ember makacsul ragaszkodott az álláspontjához, nem okozhat meglepetést az, hogy számos, az IPv6 esetében hozott döntés erősen vitatott volt. Ezek közül egy párat foglalunk össze az alábbiakban. A részleteket lásd (Huitema, 1996).

Már megemlítettük a vitát a címek hosszáról. Az eredmény egy kompromisszum lett: 16 bájtos rögzített hosszúságú címek.

Egy másik csata bontakozott ki az *Átugráskorlát* mező hossza körül. Az egyik tábor nagyon úgy érezte, hogy a maximális átugrásszám (a 8 bites mezőből adódó) 255-re való korlátozása öreg hiba. Végül is, a 32 ugrásból álló utak ma már elterjedtek, és 10 év múlva sokkal hosszabb utak is elterjedtek lehetnek. Ezek az emberek azzal érveltek, hogy egy hatalmas méretű cím használata előrelátó volt, de egy kicsi átugrásszámláló használata rövidlátásra utal. Álláspontjuk szerint a legnagyobb bűn, amit egy számítástechnikus elkövethet, hogy valahol túl kevés bitet hagy meg.

A válasz az volt, hogy minden mező megnövelésére akad érv, de ez egy felduzzadt fejrészhez vezetne. Egyébként is, az *Átugráskorlát* mező feladata, hogy a csomagok ne kószálhassanak hosszú ideig szerteszét, és 65 535 ugrás túlságosan hosszú. Végül, ahogy az Internet növekszik, egyre több és több nagytávolságú összeköttetés fog épülni, lehetővé téve, hogy bármely országból bármely más országba legfeljebb féltucat ugrással eljussunk. Ha 125 ugrásnál többe kerül eljutni a forrás és a cél számára, hogy eljussanak a megfelelő nemzetközi átjáróikhoz, valami nincs rendben a nemzeti gerinchálózatokkal. A 8 bitet támogatók nyerték meg ezt a csatát.

Egy másik hevesen vitatott téma a csomagméret volt. A szuperszámítógépes társadalom 64 KB-nál nagyobb csomagokat akart. Amikor egy szuperszámítógép átvitelbe kezd, akkor tényleg dolga van, és nem akarja, hogy 64 KB-onként megszakítsák. A nagy csomagok ellen az az érv merült fel, hogy ha egy 1 MB-os csomag elér egy 1,5 Mb/s-os T1 vonalat, akkor ez a csomag 5 másodpercre lefoglalja a vonalat, és egy jól észlelhető késleltetést okoz a szintén ezt a vonalat használó interaktív felhasználóknál. Itt egy kompromisszumot értek el: a rendes csomagok 64 KB-ra vannak korlátozva, de az ugrásról ugrásra kiegészítő fejrész használható jumbogramok engedélyezésére.

Egy harmadik forró téma az IPv4 ellenőrző összeg eltávolítása volt. Néhány ember ezt ahhoz hasonlította, mintha egy autóból kiszednénk a fékeket. Ha így teszünk, az

autó könnyebb lesz, és gyorsabban haladhat, de ha valami váratlan esemény történik, akkor gond van.

Az ellenőrző összegek elleni érv az volt, hogy bármely alkalmazásnak, amely valóban törődik az adatintegritással, amúgy is kell hogy legyen egy ellenőrző összege a szállítási rétegben, így túlzás az IP-be is berakni egyet (az adatkapcsolati réteg ellenőrző összegén felül).

A mozgó hosztok is vita tárgyát képezték. Ha egy hordozható számítógép félig körberepüli a világot, működhet-e tovább a célban ugyanazzal az IPv6 címmel, vagy egy hazai és idegen ügynökös sémát kelljen használnia? A mozgó hosztok aszimmetriát is okozhatnak a router rendszerben. Könnyen meglehet, hogy egy kis mobil számítógép könnyen meghallja a nagy mozdulatlan router által kibocsátott erős jelet, de a mozdulatlan router nem hallja a mozgó hoszt által kibocsátott erőtlen jelet. Következésképpen néhány ember a mozgó hosztok számára kifejezett támogatást akart beépíteni az IPv6-ba. Ez az erőfeszítés meghiúsult, amikor egyik javaslatban sem tudtak megegyezni.

Valószínűleg a legnagyobb csata a biztonság körül dúlt. Mindenki egyetértett, hogy szükség van rá. A harc afölött ment, hogy hol és hogyan legyen. Először nézzük a hol kérdését. A hálózati rétegbe helyezése mellett az az érv szólt, hogy ekkor ez szabványos szolgáltatássá válik, amelyet minden alkalmazás minden előzetes tervezés nélkül használhat. Az ellene szóló érv az, hogy a valóban titkos alkalmazások a végpontok közötti titkosításnál nem érik be kevesebbel, ahol is forrásalkalmazás végzi a titkosítást és a célalkalmazás fejti vissza. Bármivel, ami ennél kevésbé biztonságos, a felhasználó ki van szolgáltatva a hálózati réteg esetlegesen hibás megvalósításának, amelyek fölött nem bír ellenőrzéssel. Erre az érvre az a válasz, hogy ezek az alkalmazások tartózkodhatnak az IP titkosítási tulajdonságainak kihasználásától, és maguk végezhetik el a munkát. Erre pedig az a viszontválasz, hogy azok az emberek, akik nem bíznak meg abban, hogy a hálózat jól végezné ezt el, nem akarják megfizetni az ilyen képességgel rendelkező lassú és terjedelmes IP megvalósítások árát, még ha az ki is van kapcsolva.

A biztonság elhelyezésének egy másik vetülete ahhoz kapcsolódik, hogy sok (de nem minden) országnak szigorú kiviteli törvényei vannak a titkosításra nézve. Néhány ország, főképpen Franciaország és Irak, belföldön is nagyban korlátozzák a használatát, hogy az embereknek ne lehessenek titkaik a rendőrség előtt. Ennek eredményeként semmilyen olyan IP megvalósítást, amely elég erős titkosítási rendszert használ ahhoz, hogy valóban hasznos legyen, nem lehet kivinni az Egyesült Államokból (és sok más országból) világszerte a vásárlókhoz. A legtöbb számítógépes cég erőteljesen ellenzi azt, hogy két szoftverkészletet kelljen karbantartania, egyet belföldi használatra és egyet kivitelre.

Egy lehetséges megoldás, hogy minden gyártó a titkosítási műhelyeit olyan országba költözteti, amely nem szabályozza a titkosítást, mint például Finnország és Svájc. Ott azután erős titkosítási szoftverek tervezhetők és állíthatók elő, majd ezeket legálisan szállíthatják Franciaország és Irak kivételével minden országba. Ennek a megközelítésnek a problémája az, hogy ha a router szoftver egyik része egy országban készül, a másik része pedig másikban, az integrációs problémákhoz vezethet.

A végső, titkosítással kapcsolatos vita arról szólt, hogy melyek legyenek azok az

alapalgoritmusok, amelyeket minden megvalósításnak támogatnia kell. Bár az MD5-öt viszonylag biztonságosnak hitték, a titkosítási tudomány újabb eredményei meggyengíthetik. Egy komoly titkosító sem hiszi, hogy a DES ellenáll nagyobb kormányzatok támadásának, de valószínűleg elég jó ahhoz, hogy egyelőre még a legrafináltabb 12 évesek eszén is túljárjon. Ezért a kompromisszum az lett, hogy az IPv6-ban kötelezővé tették a titkosítást, egy korszerű ellenőrzőösszeg-algoritmust használnak a jó hitelesítés érdekében, és egy gyenge algoritmust a titkosításhoz, de megadják a felhasználóknak a lehetőséget, hogy ezeket az algoritmusokat a sajátjaikkal helyettesítsék.

Egy ponton nem volt vita, és ez az, hogy senki nem számít arra, hogy egy vasárnap kikapcsolják az IPv4 Internetet és hétfőn reggel mint IPv6 Internet indul újra. Ehelyett elszigetelt IPv6 „szigetek" fognak átállni, kezdetben alagutakon keresztül kommunikálva. Ahogy az IPv6 szigetek nőnek, úgy olvadnak össze nagyobb szigetekké. Végül az összes sziget össze fog olvadni, és az Internet teljesen át fog állni. Mivelhogy a ma üzemelő IPv4 routerekben hatalmas beruházás van, az átállási folyamat valószínűleg egy évtizedet fog igénybe venni. Ezen oknál fogva hatalmas erőfeszítések történtek annak biztosítása érdekében, hogy az átállás amennyire csak lehet, fájdalommentes legyen.

5.6. Hálózati réteg az ATM hálózatokban

Az ATM modell (1.30. ábra) rétegei nem képezhetők le igazán jól az OSI rétegekre, és ez kétértelműségekhez vezet. Az OSI adatkapcsolati réteg két azonos fizikai vezetéken (vagy fényvezető szálon) levő gép közti keretezési és átviteli protokollokkal foglalkozik. Az adatkapcsolati protokollok egyugrásos protokollok. Nem foglalkoznak a két végpont közötti összeköttetésekkel, mivel az adatkapcsolati rétegben nem fordul elő forgalomirányítás és kapcsolás.

A legalacsonyabb réteg, amely a forrástól a célig megy, és ezért forgalomirányítást és kapcsolást is tartalmaz (tehát többugrásos), a hálózati réteg. Az ATM réteg a cellák a forrástól a célig történő mozgatásával foglalkozik, és határozottan tartalmaz az ATM kapcsolókon belül router algoritmusokat és protokollokat. A globális címzéssel is foglalkozik. Ezért az ATM réteg feladata szerint ellátja a hálózati rétegtől elvárt munkát. Az ATM réteg nem ad garanciát a 100 százalékos megbízhatóságra, de ez nem is igény egy hálózati rétegbeli protokollal szemben.

Az ATM réteg az X.25 3. rétegére is hasonlít, amely mindenki szerint hálózati rétegbeli protokoll. A bitbeállításoktól függően az X.25 hálózati réteg protokoll lehet megbízható vagy nem megbízható, de a legtöbb megvalósítás megbízhatatlanként kezeli. Mivel az ATM réteg rendelkezik a hálózati rétegtől elvárt feladatkörrel, de nem rendelkezik az adatkapcsolati rétegtől elvárt feladatkörrel, továbbá nagyon hasonló a létező hálózati rétegbeli protokollokhoz, ezért az ATM réteget ebben a fejezetben tárgyaljuk.

Ezen a területen zűrzavar uralkodik, mivel az ATM közösségből sokan az ATM réteget adatkapcsolati rétegként, vagy LAN emuláció végzésekor akár fizikai rétegként tekintik. Az Internet-közösségből sokan szintén adatkapcsolati rétegnek tekintik, mi-

vel az IP-t ennek tetejére akarják rárakni, és ehhez az ötlethez az illene, ha az ATM réteg adatkapcsolati réteg lenne. (Bár ha ezt a gondolatmenetet végigkövetjük, akkor fizikai jellemzőitől függetlenül *minden* hálózat az adatkapcsolati rétegben működik.)

Az egyetlen gond az, hogy az ATM rétegben nincsenek meg az adatkapcsolati rétegbeli protokollokra jellemző tulajdonságok, vagyis nem egy egyugrásos protokoll, amelyet a vezeték két ellenkező végén levő gépek használnak. Nem olyan, mint a 3. fejezetben 1-től 6-ig leírt protokollok. Egy hálózati réteg tulajdonságaival rendelkezik: két végpont közötti virtuális áramkörök, kapcsolás és forgalomirányítás.

A szerzőt ez egy régi találós kérdésre emlékezteti:

Kérdés: Hány lába lenne az öszvérnek, ha a farkát lábnak neveznénk?
Válasz: Négy. Attól, hogy a farkát lábnak *nevezzük*, még *nem lesz* az.

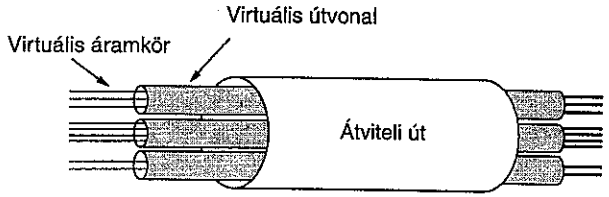
Legyen elég annyit mondani, hogy az olvasót itt figyelmeztetjük, óvakodjon a vitától, ugyanis egy nagy adag viharos érzelemmel jár együtt.

Az ATM réteg összeköttetés alapú, a felkínált szolgálat és a belső működés szerint is. Az ATM réteg alapeleme a virtuális áramkör, amelyet hivatalosan **virtuális csatornának (virtual channel)** neveznek. A virtuális áramkör rendesen egy forrás és egy cél közti összeköttetés, bár megengedettek többbesküldéses összeköttetések is. A virtuális áramkörök egyirányúak, de létrehozhatunk egyszerre egy áramkörpárt is. A pár mindkét tagját ugyanaz az azonosító címzi meg, így tulajdonképpen a virtuális áramkör duplexnek tekinthető. Ellenben a csatornakapacitás és más tulajdonságok eltérőek lehetnek a két irányban, sőt, az egyikre ezek értéke akár nulla is lehet.

Az ATM réteg azért szokatlan összeköttetés alapú protokoll, mert nem biztosít semmiféle nyugtázást. Ennek a tervezésnek az az oka, hogy az ATM-et fényvezető szálas hálózatokon való használatra tervezték, és ezek nagyon megbízhatóak. Megfelelőnek gondolták, hogy a hibavédelmet a felsőbb rétegekre hagyják. Végül is, az adatkapcsolati vagy hálózati rétegekben a nyugtázás valójában csupán optimalizálás. Mindig elegendő az, ha a szállítási réteg küld egy üzenetet, majd ha nem nyugtázzák időben, akkor még egyszer elküldi.

Továbbá, az ATM hálózatokat gyakran használják valós idejű forgalom továbbítására, mint amilyen a hang és a mozgókép. Az ilyesfajta forgalom számára egy esetleges rossz cella újraadása rosszabb, mintha azt figyelmen kívül hagynánk.

A nyugtázások hiánya ellenére az ATM réteg azért biztosít egy kemény garanciát: az egy virtuális áramkörön elküldött cellák soha nem fognak rossz sorrendben megérkezni. Az ATM alhálózatnak szabad eldobni cellákat, ha torlódás lép fel, de semmilyen körülmények között nem rendezheti át az egy virtuális áramkörön elküldött cellákat. Viszont a *különböző* virtuális áramkörökön elküldött cellákra nem adnak garanciát. Például, ha egy hoszt elküld egy cellát a 10. virtuális áramkörön, és később elküld ugyanahhoz a célhoz a 20. virtuális áramkörön is egy cellát, akkor a második cella érkezhet meg elsőnek. Ha a két cellát ugyanazon a virtuális áramkörön küldték volna, az első beérkező mindig az lenne, amelyiket elsőnek küldték el.

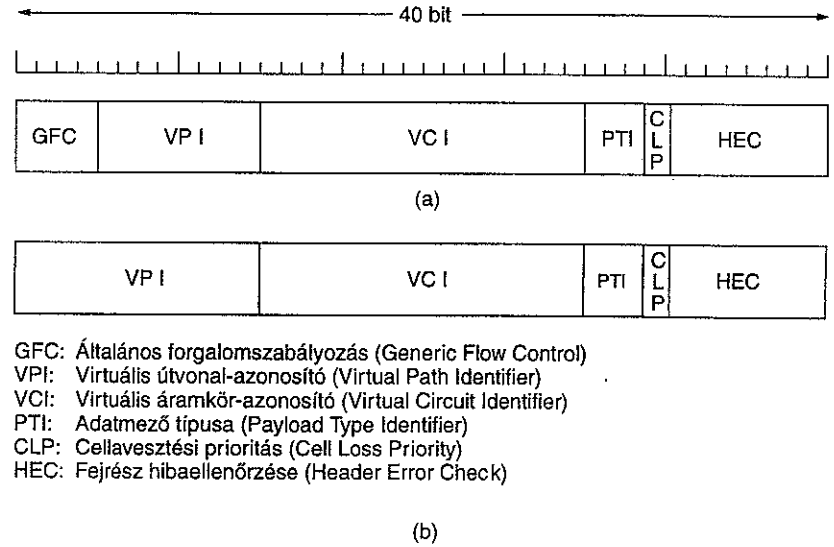


5.61. ábra. *Egy átviteli út több virtuális útvonalat is tartalmazhat, amelyek közül mindegyik több virtuális áramkört tartalmazhat*

Az ATM réteg egy kétszintes összeköttetés hierarchiát támogat, amely látható a szállítási réteg számára. Egy adott forrástól egy adott célig bármelyik átviteli út mentén a virtuális áramkörök egy úgynevezett **virtuális útvonalba (virtual path)** gyűjthetők össze, ahogy az 5.61. ábrán látszik. Elméletileg a virtuális útvonal hasonló egy sodort rézdrótköteghez: amikor újrairányítják, az összes párt (virtuális áramkört) együtt irányítják. Ennek a kétszintes tervezésnek a következményeit később részletesen végiggondoljuk.

5.6.1. Cellaformátumok

Az ATM rétegben két interfészt különböztetnek meg: az **UNI-t (User-Network Interface – felhasználó-hálózat interfész)** és az **NNI-t (Network-Network Interface – hálózat-hálózat interfész).** Az előbbi a hoszt és az ATM hálózat (sok esetben a vevő és a szolgáltató) közti határfelületet definiálja. Az utóbbi két ATM kapcsoló (ez az ATM szóhasználatban a routert jelenti) közti vonalra vonatkozik.



5.62. ábra. *(a) Az ATM réteg fejrésze UNI esetén. (b) Az ATM réteg fejrésze NNI esetén*

Mindkét esetben a cellák egy 5 bájtos fejrészből és 48 bájt adatrészből állnak, de a két fejrész kissé különbözik. A fejrészek az 5.62. ábrán láthatók úgy, ahogy az ATM Forum azokat definiálta. A cellákon belül a legbaloldalibb bájtot, a bájtokon belül a legbaloldalibb bitet viszik át először.

A *GFC* mező csak a hoszt és a hálózat közti cellákban szerepel. Az első kapcsoló amin áthalad, felülírja, így nincs a végpontok közötti forgalom szempontjából jelentősége, és nem kerül el a célhoz. Eredetileg úgy tervezték, hogy ennek esetleg valami szerepe lesz a forgalomszabályozásban vagy a prioritásban a hoszt és a hálózat közt, de nincsenek definiált értékei, és a hálózat figyelmen kívül hagyja. A legjobb, ha ezt a szabvány egy hibájának tekintjük.

A *VPI* mező egy kis egész szám, amely egy bizonyos virtuális útvonalat választ ki (l. az 5.61. ábrát). A *VCI* mező egy adott virtuális áramkört választ ki a megadott virtuális útvonalon belül. Mivel (UNI esetén) a *VPI* mezőnek 8 bitje, és a *VCI* mezőnek 16 bitje van, egy hosztnak legfeljebb 256 VP kötege lehet, amelyek közül mindegyik legfeljebb 65 535 virtuális áramkört tartalmazhat. Valójában mindegyikből ezeknél kicsivel kevesebb érhető el, mert néhány *VCI*-t fenntartottak vezérlő funkciókra, mint pl. a virtuális áramkörök felállítására.

A *PTI* mező határozza meg a cella által hordozott adatok típusát, az 5.63. ábrán megadott értékekkel összhangban. Itt a cellatípusokat a felhasználó, a torlódási információt azonban a hálózat szolgáltatja. Más szavakkal, egy 000-s PTI értékkel elküldött cella 010-val érkezhet meg, hogy a célt figyelmeztesse az útközben tapasztalt problémákra.

A *CLP* bitet a hoszt állíthatja be, hogy megkülönböztesse az alacsony prioritású forgalmat a magas prioritásútól. Ha torlódás lép fel, és a cellákat el kell dobni, a kapcsolók először az 1-be állított *CLP*-jű cellákat kísérlik meg eldobni, mielőtt bármely olyat eldobnának, amelyben ez a bit 0-ba van állítva.

Végül a *HEC* mező a fejrész ellenőrző összege. Ez nem ellenőrzi az adatrészt. Egy 40 bites szám Hamming-kódja csak 5 bitet igényel, így 8 bit esetén ennél kifinomultabb kódot is használhatunk. A kiválasztott kód minden egybites hibát ki tud javítani, és a több-bites hibák körülbelül 90 százalékát észleli. Több tanulmány is megmutatta, hogy az optikai kapcsolatokon a hibák túlnyomó többsége egybites hiba.

Adat mező típusa	Jelentés
000	Felhasználói adatcella, torlódás nincs, cellatípus 0
001	Felhasználói adatcella, torlódás nincs, cellatípus 1
010	Felhasználói adatcella, torlódást észleltek, cellatípus 0
011	Felhasználói adatcella, torlódást észleltek, cellatípus 1
100	Karbantartási információ összekötött kapcsolók között
101	Karbantartási információ a forrás- és célkapcsoló között
110	Erőforrás-kezelő cella (az ABR torlódásvédelemhez használják)
111	Fenntartott jövőbeni funkciók számára

5.63. ábra. *A PTI mező értékei*

A fejrész után következik az adatrész 48 bájtja. Nem mind a 48 bájt érhető el a felhasználó számára, mivel néhány AAL protokoll a fejrészét és a farokrészét az adatrész belsejébe helyezi.

Az NNI formátum ugyanolyan, mint az UNI formátum, csak a *GFC* mező nincs jelen és ezt a 4 bitet a *VPI* mező 8-ról 12-re való kibővítéséhez használják.

5.6.2. Összeköttetés létesítése

Az ATM mind az állandó, mind a kapcsolt virtuális áramköröket támogatja. Az állandó virtuális áramkörök mindig jelen vannak, és tetszés szerint használhatók, akárcsak a bérelt vonalak. A kapcsolt virtuális áramköröket létre kell hozni minden alkalommal, amikor használni kívánják őket, ugyanúgy, mint a telefonhívásoknál. Ebben a szakaszban leírjuk, hogyan jönnek létre a virtuális áramkörök.

Az összeköttetés létesítése nem része az ATM rétegnek. A vezérlési sík (l. 1.30. ábra) kezeli egy nagyon összetett ITU protokollt, a **Q.2931**-et használva (Stiller, 1995). Ettől függetlenül, egy hálózati összeköttetés létesítésének logikai helye a hálózati rétegben van, és mivel a hasonló hálózati rétegbeli protokollok is itt végzik el az összeköttetés létesítését, mi is itt fogjuk tárgyalni.

Sokféle módja lehet egy összeköttetés létesítésének. A szokásos módja az, hogy a jelzéshez először egy virtuális áramkört kell szerezni, és ezt használni. Hogy egy ilyen áramkör létrejöjjön, a kérést tartalmazó cellákat a 0. virtuális útvonalon, az 5. virtuális áramkörön elküldik. Ha ez sikeres, akkor egy új virtuális áramkör nyílik meg, amelyen az összeköttetés-felépítési kérések és válaszok küldhetők és fogadhatók.

Ennek a kétlépéses eljárásnak az oka az, hogy így lehet az 5. (alig használt) virtuális áramkör számára fenntartott sávszélességet rendkívül alacsonyan tartani. De egy másik módja is van virtuális áramkörök létrehozásának. Néhány szolgáltató lehetővé

Üzenet	Jelentése, amikor a hoszt küldi	Jelentése, amikor a hálózat küldi
LÉTREHOZÁS (SETUP)	Kérem, hozzon létre egy áramkört	Bejövő hívás
HÍVÁS FOLYAMATBAN (CALL PROCEEDING)	Láttam a bejövő hívást	Az Ön híváskérését megkíséreljük
KAPCSOLÁS (CONNECT)	Elfogadom a bejövő hívást	Az Ön híváskérését elfogadták
KAPCSOLÁS NYUGTA (CONNECT ACK)	Köszönjük, hogy elfogadta	Köszönjük, hogy hívott
ELENGEDÉS (RELEASE)	Kérem, szakítsa meg a hívást	A másik félnek elege van
ELENGEDÉS KÉSZ (RELEASE COMPLETE)	Nyugta az ELENGEDÉS-re	Nyugta az ELENGEDÉS-re

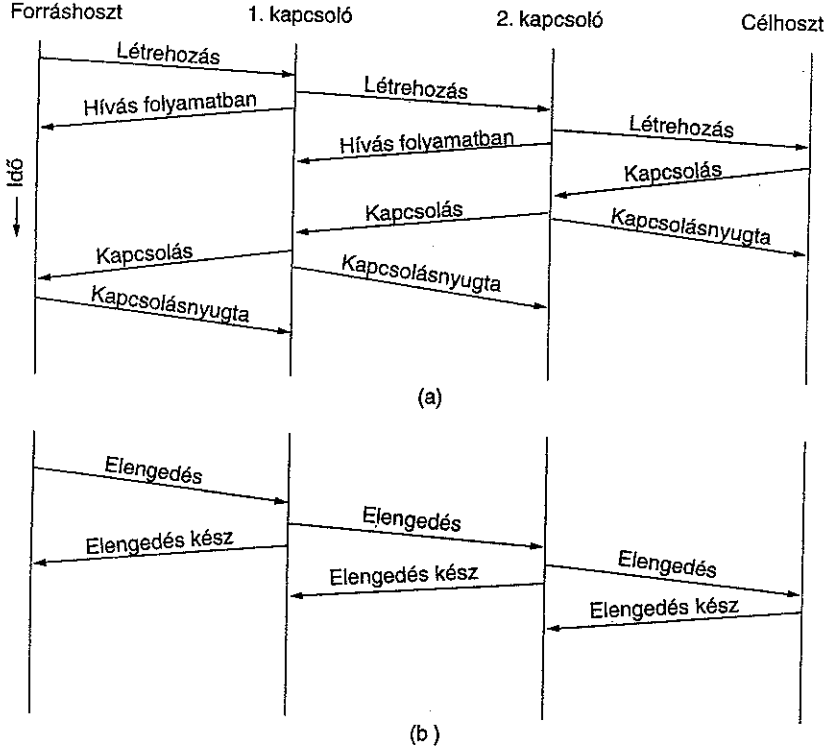
5.64. ábra. *Az összeköttetés felépítéséhez és lebontásához használt üzenetek*

teheti a felhasználói számára, hogy állandó virtuális útvonalaik legyenek előre meghatározott végpontok között, vagy hogy ezeket dinamikusan létrehozzák. Ha már egyszer egy hosztnak van egy virtuális útvonala valamely másik hosztig, saját maga helyezhet el ezen virtuális áramköröket, a kapcsolók bevonása nélkül.

A virtuális áramkörök létrehozása az 5.64. ábrán felsorolt hat üzenettípust használja. Minden üzenet egy vagy több cellát foglal el, és az üzenettípust, a hosszat, és a paramétereket tartalmazza. Az üzeneteket vagy egy hoszt küldheti a hálózatnak, vagy a hálózat küldheti (rendszerint egy másik hoszt üzenetére válaszul) egy hosztnak. Számos másfajta állapot- és hibabejelentő üzenet is létezik, de ezeket itt nem soroljuk fel.

Egy hívás felépítésének rendes módja az, hogy egy hoszt egy LÉTREHOZÁS (SETUP) üzenetet küld egy speciális virtuális áramkörön. A hálózat HÍVÁS FOLYAMATBAN (ALL PROCEEDING) üzenettel válaszol, hogy nyugtázza a kérés beérkezését. Ahogy a LÉTREHOZÁS (SETUP) üzenet terjed a cél felé, minden ugrásnál HÍVÁS FOLYAMATBAN (ALL PROCEEDING) üzenettel nyugtázzák.

Amikor a LÉTREHOZÁS (SETUP) üzenet végül megérkezik, a célhoszt egy KAPCSOLÁS (CONNECT) üzenettel válaszolhat, hogy elfogadja a hívást. Ezután a hálózat egy



5.65. ábra. *(a) Összeköttetés felépítése egy ATM hálózatban. (b) Az összeköttetés lebontása*

KAPCSOLÁSNYUGTA (CONNECT ACK) üzenetet küld, hogy jelezze, megkapta a KAPCSOLÁS (CONNECT) üzenetet. Ahogy a KAPCSOLÁS (CONNECT) üzenet visszafelé terjed a kezdeményező felé, minden ezt megkapó kapcsoló nyugtázza egy KAPCSOLÁSNYUGTA (CONNECT ACK) üzenettel. Az események ezen sorozata látható az 5.65.(a) ábrán.

A virtuális áramkör megszakításához szükséges sorozat egyszerű. A megszakítani kívánó hoszt egyszerűen egy ELENGEDÉS (RELEASE) üzenetet küld, amely tovaterjed a másik vég felé, és az áramkört bontja. Az úton minden ugrást nyugtáznak, ahogy az 5.65.(b) ábrán látszik.

Az ATM lehetővé teszi többesküldéses csatornák létrehozását. Egy többesküldéses csatornának egy adója és egynél több vevője van. Ezeket úgy építik fel, hogy a célok egyikéig létrehoznak a rendes módon egy összeköttetést. Ezután egy TÁRS HOZZÁADÁSA (ADD PARTY) üzenetet küldenek, hogy az előző hívás által visszaadott virtuális áramkörhöz egy másik célt is csatoljanak. Ezek után további TÁRS HOZZÁADÁSA (ADD PARTY) üzeneteket küldhetnek, hogy a többesküldéses csoport méretét növeljék.

Annak érdekében, hogy egy összeköttetés létesüljön egy célhoz, a LÉTREHOZÁS (SETUP) üzenetbe foglalt cím segítségével definiálni kell a célcímet. Az ATM címeknek három formájuk van. Az első formátum 20 bájt hosszúságú, és az OSI címeken alapul. Az első bájt azt jelzi, hogy a cím a három formátum közül melyikben van. Az első formátumban a 2. és 3. bájt az országot határozza meg, a 4. bájt a cím maradék részének formátumát adja meg. Ebben található egy 3 bájtos hatóság, egy 2 bájtos körzet, egy 2 bájtos terület, és egy 6 bájtos cím, plusz néhány további tétel. A második formátumban a 2. és 3. bájtok egy ország helyett egy nemzetközi szervezetet határoznak meg. A cím maradék része ugyanaz, mint az 1. formátumban. Alternatívaként egy régebbi címzésmód (CCITT E.164) is megengedett, amely 15 decimális jegyű ISDN telefonszámokat használ.

5.6.3. Forgalomirányítás és kapcsolás

Amikor egy virtuális áramkör létrejön, a LÉTREHOZÁS (SETUP) üzenet eljut a hálózaton keresztül a forrástól a célig. A router algoritmus határozza meg az ezen üzenet által követendő utat, és ezáltal a virtuális áramkör útvonalát is. Az ATM szabvány nem határoz meg egyetlen router algoritmust sem, így a szolgáltató szabadon választhat az ezen fejezetben korábban tárgyalt algoritmusok közt, vagy használhat egy másikat is.

A korábbi összeköttetés alapú hálózatokkal, mint például az X.25-tel való tapasztalat megmutatta, hogy a kapcsolókban tekintélyes mennyiségű számítási teljesítmény fordítódik arra, hogy a minden cellában levő virtuális áramkör információt felhasználva, a kimeneti vonalat meghatározzák. Az ATM tervezői el akarták kerüli ezt az utat, így az ATM réteget úgy tervezték, hogy lehetővé váljon a hatékony forgalomirányítás. Pontosabban, az ötlet az volt, hogy a forgalomirányítást ne a *VCI*, hanem a *VPI* mezőre alapozzák, kivéve minden irányban az utolsó ugrást, amikor a cellákat egy kapcsoló és egy hoszt között küldik át. Két kapcsoló között csak a virtuális útvonalat használják.

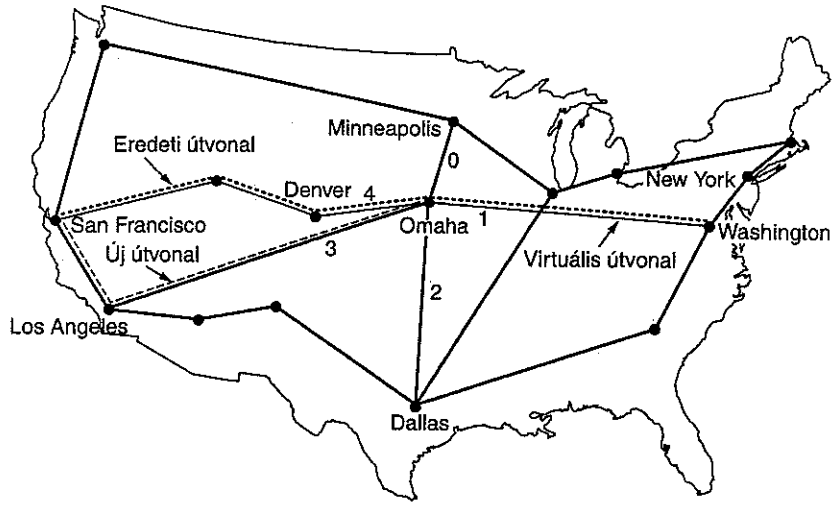
Számos előnye van annak, ha csak a *VPI*-ket használjuk a belső kapcsolók közt Először is, ha már egyszer létrehoztak egy virtuális útvonalat a forrástól a célig, az ezen útvonal menti további virtuális áramkörök egyszerűen ezt az útvonalat követhe-

tik. Nem kell új forgalomirányítási döntéseket hozni. Ez olyan, mintha egy érpárköteget már kihúztak volna a forrástól a célig. Egy új összeköttetés létrehozása csak egy kihasználatlan érpár kiutalását jelenti.

Másodszor, az egyes cellák irányítása könnyebb, ha az egy adott útvonalhoz tartozó virtuális áramkörök mindig egy kötegben vannak. A forgalomirányítási döntés csak egy 12 bites szám szemrevételét igényli, nem pedig egy 12 bites és egy 16 bites számét. Alább leírjuk, hogyan történik a cellakapcsolás, de még anélkül is, hogy a részletekbe mennénk, nyilvánvalónak kell lennie, hogy egy 2^{12} bejegyzésből álló táblázatba történő címzés keresztülvihető, de egy 2^{28} bejegyzésből álló táblázatba történő címzés nem.

Harmadszor, ha minden forgalomirányítást a virtuális útvonalakra alapozunk, az könnyebbé teszi egy egész csoport virtuális áramkör átkapcsolását. Vegyük például az Egyesült Államok feltételezett ATM gerinchálózatát, amely az 5.66. ábrán látható. Rendesen a New Yorkból San Franciscóba tartó virtuális áramkörök áthaladnak Omahán és Denveren. De tegyük fel, hogy zavar támad az Omaha–Denver vonalon. Az Omaha–Denver virtuális útvonalat átirányítva Los Angelesbe és onnan San Franciscóba, az összes virtuális áramkör (esetleg akár az összes 65 535 darab) egyetlen művelettel átkapcsolható, az esetleges több ezer művelet helyett.

Végül, a virtuális útvonalak könnyebbé teszik, hogy a szolgáltatók, zárt felhasználói csoportokat (magánhálózatokat) ajánljanak fel vállalati ügyfeleik számára. Egy társaság állandó virtuális útvonalakból álló hálózatot hozhat létre a különböző irodái közt, és ezután igény szerint helyezhet el ezeken az útvonalakon virtuális áramköröket. Kívülről egyetlen hívás sem jöhet be a magánhálózatba, és egyetlen hívás sem hagyhatja el a magánhálózatot, csakis speciális átjárókon keresztül. Sok társaság szereti a biztonságnak ezt a fajtáját.



5.66. ábra. *Egy virtuális útvonal átirányítása annak minden virtuális áramkörét átirányítja*

Hogy a kapcsolók valóban a *VPI* mezőt fogják a forgalomirányításhoz felhasználni, ahogy terveztük, vagy a *VPI* és a *VCI* mezők egy kombinációját fogják használni (és ezáltal a megtárgyalt előnyöket mind megsemmisítik), azt majd meglátjuk. A kezdeti tapasztalatok nem túl biztatóak.

Most nézzük meg, hogyan irányíthatják a cellákat egy belső kapcsolón (amely csak más kapcsolókkal van összeköttetésben, hosztokkal nem). Hogy egy konkrét esetet nézzünk, vegyük az 5.66. ábra omahai kapcsolóját. Mind az öt bejövő vonalához van egy táblázata, *vpi_table*, amelyet a bejövő *VPI*-k címeznek meg. E tábla megmondja, hogy az öt közül melyik kimenő vonalat kell használni és milyen *VPI* értéket kell a kimenő cellákba helyezni. Tegyük fel, hogy az öt vonalat Minneapolistól kezdve az óramutató járásával egyezően 0-tól 4-ig számozzák. Minden kimeneti vonalhoz a kapcsoló egy bit-térképet kezel, amely megmondja, milyen *VPI*-ket használnak jelenleg azon a vonalon.

Amikor a kapcsolót elindítják, a *vpi_table* struktúrában minden bejegyzést használaton kívülinek tüntetnek fel. Ugyanakkor minden bit-térkép azt jelzi, hogy minden *VPI* elérhető (kivéve a fenntartottak). Most tegyük fel, hogy a hívások az 5.67. ábra szerint jönnek.

Ahogy minden egyes virtuális útvonal (és virtuális áramkör) felépül, bejegyzések keletkeznek a táblázatokban. Feltételezzük, hogy a virtuális áramkörök duplexek, így minden létrehozás két bejegyzést eredményez, egyet az előreirányú forgalomnak a forrás felől és egyet a visszairányú forgalomnak a cél felől.

Az 5.67. ábra útvonalainak megfelelő táblázatok az 5.68. ábrán láthatók. Például az első hívás a (4, 1) bejegyzést hozza létre a washingtoni (DC) táblázatban az 1. *VPI* számára, mert ez hivatkozik az 1. vonalon 1. *VPI*-vel érkező és San Franciscóba tartó cellákra. Viszont a denveri táblázatban is létrejön egy bejegyzés, amely azt mutatja, hogy a Denver felől jövő, 1. *VPI*-vel rendelkező celláknak az 1. vonalon, 1. *VPI*-vel kell kimenniük. Ezek azok a cellák, amelyek ezen virtuális útvonalon másik irányba (San Franciscóból New Yorkba) utaznak. Vegyük észre, hogy néhány esetben két

Forrás	Bejövő vonal	Bejövő VPI	Cél	Kimenő vonal	Kimenő VPI	Útvonal
NY	1	1	SF	4	1	Új
NY	1	2	Denver	4	2	Új
LA	3	1	Minneapolis	0	1	Új
DC	1	3	LA	3	2	Új
NY	1	1	SF	4	1	Régi
SF	4	3	DC	1	4	Új
DC	1	5	SF	4	4	Új
NY	1	2	Denver	4	2	Régi
SF	4	5	Minneapolis	0	2	Új
NY	1	1	SF	4	1	Régi

5.67. ábra. *Néhány útvonal az 5.66. ábra omahai kapcsolóján keresztül*

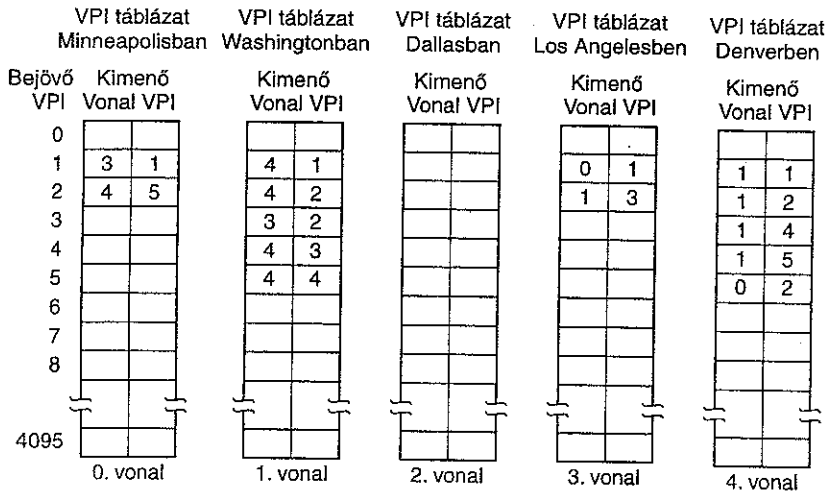
vagy három virtuális áramkör osztozik egy közös útvonalon. Nem kellenek új táblázatbejegyzések olyan további virtuális áramkörök számára, amelyek olyan forrást és célt kötnek össze, amelyekhez már rendeltünk útvonalat.

Most már elmagyarázhatjuk, hogyan dolgozzák fel a cellákat a kapcsolón belül. Tegyük fel, hogy egy cella az 1. vonalon (a washingtonin) érkezik, 3. *VPI*-vel. A kapcsoló hardverje vagy szoftverje a 3-sal megcímzi az 1. vonal táblázatát, és látja, hogy a cellának a 3. (Los Angeles-i) vonalon kell kimennie 2. *VPI*-vel. Felülírja a *VPI* mezőt egy 2-sel, és a kimenő vonal számát valahol elhelyezi a cellában, mondjuk a *HEC* mezőben, mivel azt később amúgy is újra kell számítani.

Most az a kérdés, hogy hogyan jusson a cella a jelenlegi bemeneti pufferéből a 3. vonalra. Ám ezt a kérdést már a 2. fejezetben részletesen megtárgyaltuk, és láttuk, hogyan megy végbe a kiütő és Batcher-banyan kapcsolókban.

Ezen a ponton könnyen belátható, hogyan lehet egy egész köteg virtuális áramkört átirányítani, ahogy az 5.66. ábrán is történt. Ha a washingtoni táblázatban az 1. *VPI* bejegyzését (4, 1)-ről (3, 3)-ra változtatjuk, a New Yorkból jövő és San Francisco felé tartó cellák el fognak térni Los Angeles felé. Természetesen a Los Angeles-i kapcsolót értesítenünk kell erről az eseményről, így a kapcsolónak elő kell állítani és elküldeni egy LÉTREHOZÁS (SETUP) üzenetet Los Angelesbe, hogy egy új útvonalat hozzon létre 3. *VPI*-vel. Ha már egyszer ez az útvonal létrejött, az összes New Yorkból San Francisco felé tartó virtuális áramkört átirányítottuk Los Angelesen keresztül, még ha több ezer is volt azokból. Ha nem lettek volna virtuális útvonalak, minden virtuális áramkörnek saját táblázatbejegyzése lett volna és külön kellett volna átirányítani.

Érdemes külön rámutatni, hogy a fenti tárgyalás arra vonatkozik, amikor az ATM-et WAN-okban alkalmazzák. Egy LAN-ban sokkal egyszerűbbek a dolgok. Például minden virtuális áramkör számára egyetlen virtuális útvonalat használhatunk.



5.68. ábra. *Az 5.67. ábra routereinek táblázatbejegyzései*

5.6.4. Szolgálati osztályok

Rengeteg tárgyalás és hibafelismerés eredményeképpen az ATM specifikáció 4.0-s verziójára már tisztázódott, hogy az ATM hálózatok milyenfajta forgalmat szállítanak és az ügyfelek milyen szolgálatokat akarnak. Ennek megfelelően úgy módosították a szabványt, hogy az kifejezetten felsorolja a szokásosan használt szolgálati osztályokat. Így lehetővé vált, hogy a gyártók optimalizálják az adapterkártyáikat és kapcsolóikat néhány vagy az összes ilyen osztályra. A fontosnak ítélt szolgálati osztályokat az 5.69. ábra sorolja fel.

Az **állandó bitsebességű (Constant Bit Rate, CBR)** osztály a szándékok szerint egy rézvezetéket vagy üvegszálat emulál (csak sokkal nagyobb költséggel). A biteket a vezeték egyik végén rárakják, és a másik végén azok kijönnek. Nem végeznek hibaellenőrzést, forgalomszabályozást vagy más feldolgozást. Ennek ellenére ez az osztály a jelenlegi telefonrendszer és a jövőbeni B-ISDN rendszerek közötti zökkenőmentes átálláshoz szükséges, mivel a beszédminőségű PCM csatornák, a T1 áramkörök és a telefonrendszer legnagyobb fennmaradó része állandó sebességű, szinkron bitátvitelt használ. A CBR osztállyal mindezt a forgalmat egy ATM rendszer közvetlenül elszállíthatja. A CBR szintén megfelelő minden más interaktív (vagyis valós idejű) hang- és mozgóképfolyamhoz.

A következő, a **változó bitsebességű (Variable Bit Rate, VBR)** osztály két alosztályra oszlik, valós idejűre és nem valós idejűre. Az RT-VBR-t olyan szolgálatokhoz szánják, amelyeknek változó bitsebessége és szigorú valós idejű követelményei vannak, mint amilyen az interaktív tömörített mozgókép (vagyis a videokonferencia). Az átviteli sebesség időben nagymértékben változik amiatt, ahogy az MPEG és más tömörítési eljárások működnek. Ezeknél ugyanis egy teljes alapképkeretet az alapképkeret és a jelenlegi képkeret közti különbségek sorozata követ (Pancha és El Zarki, 1994). Ezen szórás ellenére fontos, hogy az ATM hálózat ne dzsitteresítse a cellaérkezési mintát, mivel ez a megjelenítést szakadozottá teszi. Más szavakkal, mind az átlagos cellakésleltetést, mind a cellakésleltetés szórását szigorú ellenőrzés alá kell vonni. Másfelől viszont egy esetleges elveszett bit vagy cella itt elfogadható, és a legjobb, ha azt egyszerűen figyelmen kívül hagyjuk.

A másik VBR alosztály az olyan forgalom számára szolgál, amelynek fontos az időbeli kézbesítés, de az alkalmazás eltűr bizonyos fokú dzsittert. Például a multimédiás e-levelet tipikusan a vevő helyi lemezére pufferelik a megjelenítés előtt, így a cellakézbesítési idők bármilyen változását kiküszöbölik, mielőtt az e-levelet megnéznék.

Osztály	Leírás	Példa
CBR	Állandó bitsebességű	T1 áramkör
RT-VBR	Változó bitsebességű, valós idejű	Valós idejű videokonferencia
NRT-VBR	Változó bitsebességű, nem valós idejű	Multimédiás e-levél
ABR	Rendelkezésre álló bitsebességű	A Web böngészése
UBR	Specifikálatlan bitsebességű	Fájlátvitel a háttérben

5.69. ábra. *Az ATM szolgálati osztályai*

Az **ABR (Available Bit Rate, rendelkezésre álló bitsebesség)** szolgálati osztályt a lökéses forgalom számára tervezték, amelynek a sávszélesség-tartományát csak nagyjából ismerjük. A használatának egy tipikus példája az lehet, ha egy társaság bérelt vonalakkal köti össze az irodáit. A társaság dönthet arról, hogy vagy elég kapacitást helyez el ahhoz, hogy a csúcsterhelést kezelni tudja, amely azt jelenti, hogy néhány vonal a nap egy részében tétlen, vagy éppen a minimális terheléshez elegendő kapacitást helyez el, amely a nap legforgalmasabb részében torlódáshoz vezet.

Az ABR szolgálattal elkerülhető, hogy hosszú távon elkötelezzük magunkat egy rögzített sávszélesség mellett. Az ABR-rel például azt lehet mondani, hogy a kapacitásnak két pont közt mindig 5 Mb/s-nak kell lennie, de előfordulhatnak legfeljebb 10 Mb/s-os csúcsok is. A rendszer ekkor 5 Mb/s-ot mindig garantálni fog, és szükség esetén legjobb tudása szerint megpróbál 10 Mb/s-ot biztosítani, de ezt már ígéretek nélkül.

Az ABR az egyetlen osztály, amelynél a hálózat sebesség-visszajelzést biztosít az adóhoz, megkérvén azt, hogy lassítson, amikor torlódás lép fel. Feltéve, hogy az adó eleget tesz az ilyen kéréseknek, az ABR forgalom cellavesztése várhatóan alacsony lesz. Az ABR-rel utazni kicsit olyan, mint repüléskor a várólistára kerülni: ha vannak megmaradt ülések (fölös kapacitás), a várólistán levő utasokat késedelem nélkül elszállítják. Ha a kapacitás elégtelen, várniuk kell (hacsak nem elérhető a minimális sávszélesség egy része).

Végül eljutottunk az **UBR-hez (Unspecified Bit Rate – specifikálatlan bitsebesség)**, amely nem tesz ígéreteket és nem ad visszajelzést a torlódásról. Ez az osztály jól illeszkedik az IP csomagok küldéséhez, mivel az IP szintén nem tesz ígéreteket a kézbesítésről. Minden UBR cellát elfogadnak, és ha van még kapacitás, kézbesítik is azokat. Ha torlódás lép fel, az UBR cellákat eldobják anélkül, hogy az adónak visszajeleznének, és nem is várják el, hogy az adó lelassítson.

Hogy folytassuk a várólistás hasonlatunkat, az UBR-nél minden várólistán levő utas a fedélzetre kerül, de ha a cél felé félúton a pilóta azt látja, hogy kifogyóban az üzemanyag, a várólistán levő utasokat minden ceremónia nélkül kilökik a vészkijáraton. Hogy az UBR-t vonzóvá tegyék, a szolgáltatók valószínűleg olcsóbbá teszik ezt, mint a többi osztályt. Az olyan alkalmazásoknak, amelyeknek nincsenek kézbesítési megkötéseik, és amúgy is maguk akarják a hibavédelmet és a forgalomszabályozást

Szolgálat jellemzője	CBR	RT-VBR	NRT-VBR	ABR	UBR
Garantált sávszélesség	Igen	Igen	Igen	Opcionális	Nem
Megfelelő valós idejű forgalom számára	Igen	Igen	Nem	Nem	Nem
Megfelelő löketes forgalom számára	Nem	Nem	Igen	Igen	Igen
Visszajelzés a torlódásról	Nem	Nem	Nem	Igen	Nem

5.70. ábra. *Az ATM szolgálati osztályok jellemzői*

elvégezni, az UBR egy tökéletesen ésszerű választás. A fájlátvitel, az e-levél és a USENET hírek mind lehetséges jelöltek az UBR szolgálat számára, mivel ezen alkalmazások közül egyiknek sincsenek valós idejű jellemzői.

A különböző szolgálati osztályok jellemzőit az 5.70. ábra foglalja össze.

5.6.5. A szolgálat minősége

A szolgálat minősége ATM hálózatoknál fontos kérdés, részben azért, mert olyan valós idejű forgalomra használják, mint amilyen a hang és a mozgókép. Amikor egy virtuális áramkört létrehoznak, mind a szállítási rétegnek (vagyis tipikusan egy folyamatnak a hosztgépben, az „ügyfél"-nek), mind az ATM hálózati rétegnek (vagyis a hálózatot működtetőnek, a „szolgáltató"-nak) meg kell egyeznie egy, a szolgálatot definiáló szerződésben. Egy nyilvános hálózat esetében ennek a szerződésnek jogi következményei is lehetnek. Például, ha a szolgáltató megígéri, hogy csak egy cellát veszít el egymilliárdból, és ezek után kettő cellát veszít el, akkor az ügyfél jogi képviselője izgalomba jöhet, és „szerződésszegést" kiálthat mindenfelé.

Az ügyfél és a hálózat közti szerződés három részből áll:

1. A felkínálandó forgalomból.
2. A megállapodás szerinti szolgálatból.
3. A teljesítési szabályokból.

Érdemes megjegyezni, hogy a szerződés mindkét átviteli irányba eltérő lehet. Az egyik irányba egy hálózati video alkalmazás számára a felhasználó távirányítójától a videokiszolgáló felé az igényelt sávszélesség 1200 b/s lehet. A másik irányban ez esetleg 5 Mb/s is lehet. Meg kell említeni, hogy ha az ügyfél és a szolgáltató nem tudnak a feltételekről megegyezni, vagy a szolgáltató képtelen a kívánt szolgálatot biztosítani, a virtuális áramkör nem fog felépülni.

A szerződés első része a **forgalomleíró (traffic descriptor).** Ez a felkínálandó terhelést jellemzi. A szerződés második része az ügyfél által kívánt és a szolgáltató által elfogadott szolgálat minőségét specifikálja. Mind a terhelést, mind a szolgálatot mérhető mennyiségek alakjában kell megfogalmazni, hogy a teljesítés objektíven eldönthető legyen. Nem elegendő pusztán „mérsékelt terhelés" vagy „jó szolgálat" említése.

Hogy a konkrét forgalmi szerződéseket lehetővé tegye, az ATM szabvány számos **QoS (Quality of Service – szolgálat minősége)** paramétert definiál, amelyek értékeiről az ügyfél és a szolgáltató egyezkedhet. A szolgálat minőségének minden paraméterére a teljesítményt a legrosszabb esetre határozzák meg, és a szolgáltatónak kötelessége ezt elérni vagy túlhaladni. Néhány esetben a paraméter egy legkisebb érték; más esetekben ez egy legnagyobb érték. A szolgálat minőségét itt is mindkét átviteli irányban külön határozzák meg. Néhány fontosabb paramétert felsoroltunk az 5.71. ábrán, de ezek közül nem mindegyik alkalmazható minden szolgálati osztályra.

Az első három paraméter azt határozza meg, hogy milyen gyorsan akar a felhasz-

Paraméter	Rövidítés	Jelentés
Csúcs cellasebesség	PCR	Az a legnagyobb sebesség, amellyel cellákat fognak küldeni
Hosszú idejű cellasebesség	SCR	A hosszú időre érvényes átlagos cellasebesség
Minimális cellasebesség	MCR	A legkisebb elfogadható cellasebesség
Cellakésleltetés szórásának tűrése	CDVT	A legnagyobb elfogadható celladzsitter
Cellavesztési arány	CLR	A cellák elveszett vagy túl későn kézbesített hányada
Cellaátviteli késleltetés	CTD	Milyen hosszú ideig tart a kézbesítés (átlag és maximum)
Cellakésleltetés szórása	CDV	A szórás a cellák érkezési idejeiben
Cellahibaarány	CER	A hiba nélkül kézbesített cellák aránya
Hibás cellákat tartalmazó blokkok aránya	SECBR	Az összezavarodott blokkok aránya
Téves helyre kézbesített cellák aránya	CMR	A rossz helyre kézbesített cellák aránya

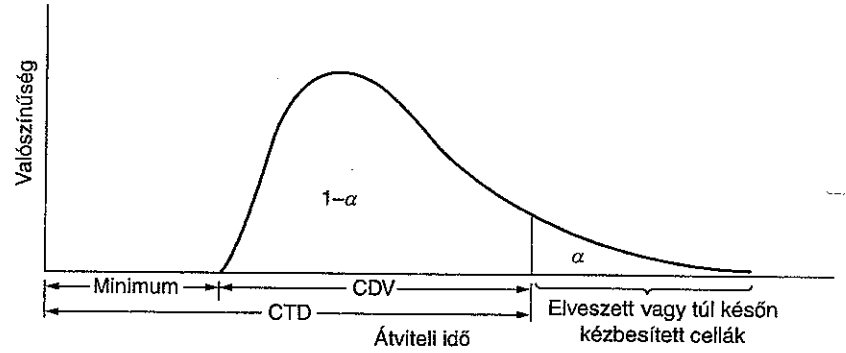
5.71. ábra. *Néhány paraméter a szolgálat minőségének paraméterei közül*

náló adni. A **PCR (Peak Cell Rate – csúcs cellasebesség)** az a maximális sebesség, amellyel az adó a cellákat tervezi adni. Ez a paraméter kisebb lehet, mint amit a vonal sávszélessége megenged. Ha az adó minden 4 μs-ban tervez egy cellát kibocsátani, a *PCR*-je 250 000 cella/s lesz, bár a valódi cellaátviteli idő 2,7 μs lehet.

Az **SCR (Sustained Cell Rate – hosszú idejű cellasebesség)** az elvárt vagy megkívánt átlagos cellasebesség egy hosszú időintervallumon átlagolva. Állandó kis sebességű (CBR) forgalomnál az *SCR* egyenlő lesz a *PCR*-rel, de minden más szolgálati osztályra az előbbi lényegesen alacsonyabb lesz. A *PCR*/*SCR* hányados mértéke a forgalom lökésszerűségének.

Az **MCR (Minimum Cell Rate – minimális cellasebesség)** az a minimális cella/s szám, amelyet az ügyfél elfogadhatónak ítél. Ha a szolgáltató képtelen ennyi sávszélességet biztosítani, vissza kell utasítania az összeköttetést. Amikor rendelkezésre álló bitsebesség (ABR) szolgálatot igényelnek, akkor az aktuális felhasznált sávszélességnek *MCR* és *PCR* közé kell esnie, de ez az összeköttetés élettartama alatt dinamikusan változhat. Ha az ügyfél és a szolgáltató megegyeznek, hogy az *MCR*-t 0-ra állítsák, akkor az ABR hasonlóvá válik a nem specifikált bitsebesség (UBR) szolgálathoz.

A **CDVT (Cell Variation Delay Tolerance – cellakésleltetés szórásának tűrése)** azt mondja meg, milyen szórás lesz jelen a cellaátviteli időkben. Ezt a *PCR*-től és *SCR*-től függetlenül határozzák meg. Egy *PCR* sebességgel működő tökéletes forrásnál minden cella pontosan 1/*PCR* idővel az előző után fog megjelenni. Egy cella sem fog soha sietni és egy sem fog soha késni, még egy pikoszekundummal sem. Egy *PCR* sebességgel működő valóságos forrásnál valamelyes szórás fog fellépni a cellaátviteli



5.72. ábra. *A cellaérkezési idők valószínűség-sűrűség függvénye*

időkben. A kérdés az: mekkora szórás fogadható el? Siethet-e egy cella 1 ns-ot? Hát 30 másodpercet? A *CDVT* az elfogadható szórás mértékét szabályozza egy lyukas vödör algoritmust használva, amelyet rövidesen leírunk.

A következő három paraméter a hálózat tulajdonságait írja le, és a vevőnél mérik őket. Mindháromban meg lehet állapodni. A **CLR (Cell Loss Ratio – cellavesztési arány)** egyértelmű. Az átvitt cellák azon hányadát méri, amelyeket egyáltalán nem vagy (valós idejű forgalom esetén) használhatatlanul későn kézbesítenek. A **CTD (Cell Transfer Delay – cellaátviteli késleltetés)** az átlagos átviteli idő a forrástól a célig. A **CDV (Cell Delay Variation – cellakésleltetés szórása)** azt méri, milyen egyenletesen kézbesítik a cellákat.

A *CDT* és *CDV* modellje az 5.72. ábrán látható. Itt t függvényében látjuk annak valószínűségét, hogy egy cella t idő alatt érkezik meg. Egy adott forrásra, célra és kapcsolókon átvezető útvonalra mindig létezik valamilyen minimális késleltetés, a terjedési és kapcsolási idő miatt. De nem minden cellának sikerül minimális idő alatt beérkeznie; a valószínűség-sűrűség függvénynek rendszerint egy hosszú farka van. Egy *CTD* értéket választva, az ügyfél és a szolgáltató valójában megegyeznek abban, hogy mennyit késhet egy cella kézbesítése ahhoz, hogy még mindig helyesen kézbesített cellának számítson. Rendesen CDV-t úgy választják meg, hogy az α, a késés miatt visszautasított cellák hányada a 10^{-10} vagy még kisebb nagyságrendbe essen. A *CDV* az érkezési idők szóródását méri. Valós idejű forgalomnál ez a paraméter gyakran fontosabb, mint a *CDT*.

Az utolsó három QoS paraméter a hálózat jellegzetességeit határozza meg, és ezeket rendszerint nem lehet egyeztetni. A **CER (Cell Error Ratio – cellahibaarány)** a cellák azon hányada, amelyek egy vagy több rossz bittel kerülnek kézbesítésre. Az **SBR (Severely-Errored Cell Block Ratio – hibás cellákat tartalmazó blokkok aránya)** az N cellából álló blokkok azon hányada, amelyben M vagy több cella tartalmaz hibát. Végül a **CMR (Cell Misinsertion Ratio – téves helyre kézbesített cellák aránya)** azon cellák száma másodpercenként, amelyeket rossz célhoz kézbesítettek egy, a fejrészben észre nem vett hiba miatt.

A forgalmi szerződés harmadik része azt mondja meg, mi jelenti a szerződés telje-