Számítógép hálózatok (16. rész)
Kapcsolatállapot alapú forgalomirányítás, hierarchikus forgalomirányítás

Folytatjuk tovább a dinamikus forgalomirányító eljárásokkal, most egy gyakorlatban is széles körben használt algoritmust mutatunk be: a kapcsolatállapot alapú forgalomirányítást. Ezek után megnézzük, mi a teendő akkor, ha hálózatunk olyan nagyra nőtt, hogy a forgalomirányító táblázatok már nem férnek el útválasztóink memóriájában.

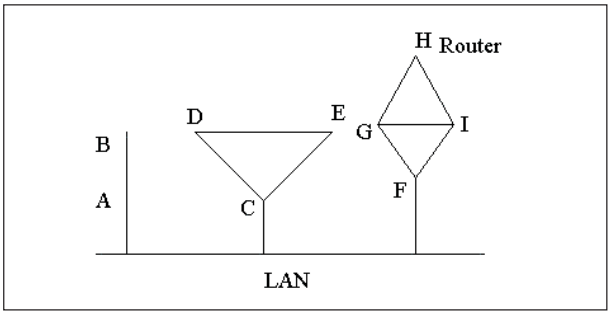
A távolságvektor alapú forgalomirányítás ugyan már egy dinamikus algoritmus volt, amit 1979-ig az internet elődje, az ARPANET is használt. A gyakorlatban azonban ez az eljárás megbukott. Leváltásában két dolog játszott szerepet. Először is az algoritmus döntéseinek meghozatalakor nem vette figyelembe az egyes vonalak sávszélességét. Ez a kezdetek kezdetén nem jelentett gondot, mivel az összes vonal 56 kb/s-os áteresztőképességgel rendelkezett, de miután egyes vonalak sávszélességét az eredetinek a többszörösére növelték, ez a hiányosság súlyos problémává kerekedett.

Az algoritmus bukásának valódi oka azonban nem ez volt, hiszen az eljárást könnyedén meg lehetett volna változtatni úgy, hogy figyelembe vegye az egyes vonalak közötti sávszélességbeli különbségeket is. Az igazi probléma inkább az volt, hogy az algoritmus a „rossz hírekre" továbbra is lassan reagált, és ezen még a megosztott látóhatár alkalmazása sem segített. Mivel a végtelenig számlálás problémájára nem sikerült elfogadható megoldást találni, egy teljesen új forgalomirányítási eljárást kellett kifejleszteni. Itt kezdődik a ***kapcsolatállapot alapú forgalomirányítás*** története.

Az algoritmus működése egyszerű: az útválasztók először a szomszédaikkal ismerkednek meg, majd megmérik a közöttük lévő késleltetéseket. Az így kapott információkat az alhálózat többi útválasztójának is elküldik, majd Dijkstra algoritmusának segítségével kiszámítják az alhálózat összes pontjához a legrövidebb utat.

Ismerkedés a szomszédokkal

Az útválasztó első feladata tehát a szomszédság feltérképezése. Ebben segít neki egy speciális, úgynevezett ***HELLO*** csomag, amelyre minden útválasztó válaszol, és a válaszban elküldi saját egyedi azonosítóját. (Az azonosítók egyedisége nagyon fontos. Tegyük fel, hogy egy útválasztó azt látja, hogy két másik útválasztó is szomszédja az X című útválasztónak. Ha az azonosítók nem egyértelműek, akkor nem lehet eldönteni, hogy a két útválasztó ugyanannak az útválasztónak a szomszédja-e, vagy két különböző X azonosítójú útválasztó van az alhálózatban).

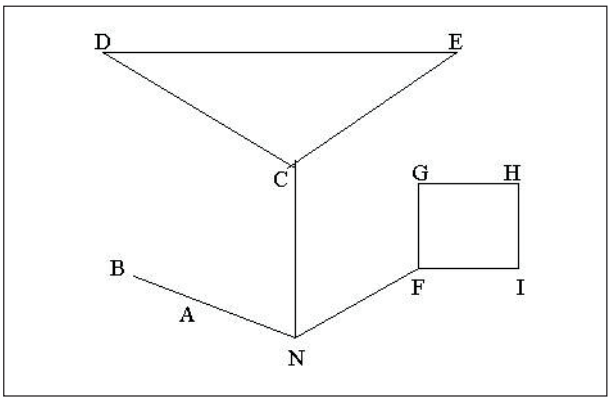


1. ábra Három router egy LAN-on

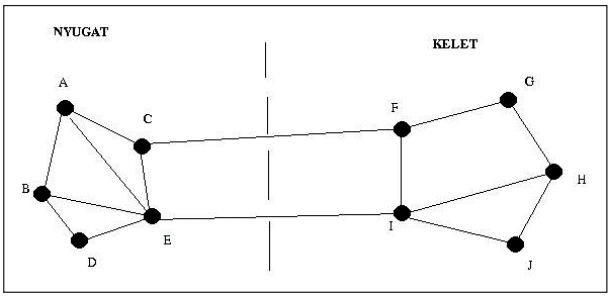
Amikor egy útválasztót bekapcsolunk, akkor az „becsönget" minden szomszédjához, azaz kiküld egy HELLO csomagot az összes kimenetén, majd vár a válaszok megérkezésére. Felmerül a kérdés, mi történik akkor, amikor egy kimeneten keresztül több útválasztót is el lehet érni, például két vagy több útválasztó egy LAN-ra van felfűzve. Az *1. ábrán* egy ilyen topológiát láthatunk, ahol A, C és F útválasztó közvetlenül, egy LAN-on keresztül kapcsolódnak egymáshoz. Ilyen esetekben célszerű a LAN-t helyettesíteni egy „virtuális" csomóponttal. A *2. ábrán* a LAN-t kicseréltük egy N címkéjű pontra. Ezek után például az A-ból a C-be az ANC úton keresztül juthatunk el.

Késleltetések mérése

A szomszédokat nem elég megismerni, hanem azt is tudni kell, hogy mekkora a – szomszédokhoz vezető – vonalak késleltetése (vagy költsége). Erre a feladatra is egy speciális csomagot alkalmaznak, amelynek neve ***ECHO*** csomag. Amikor egy útválasztó egy ilyen csomagot kap, azt soron kívül, azonnal visszaküldi azon a porton, amelyiken beérkezett. Miután az útválasztó kibocsátott egy ECHO csomagot,



2. ábra A LAN-t úgy is felfoghatjuk, mint az alhálózat egy csomópontját

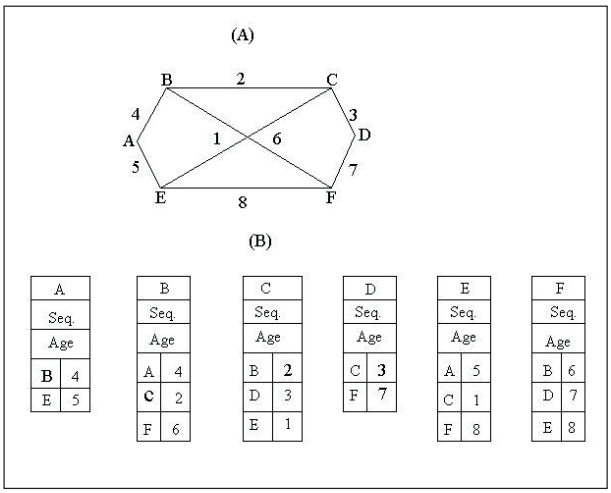


3. ábra Az ehhez hasonló toplógiájú hálózatokban nem feltétlenül előnyös, ha a vonalak költségeinek kiszámolásakor a terhelést is figyelembe vesszük

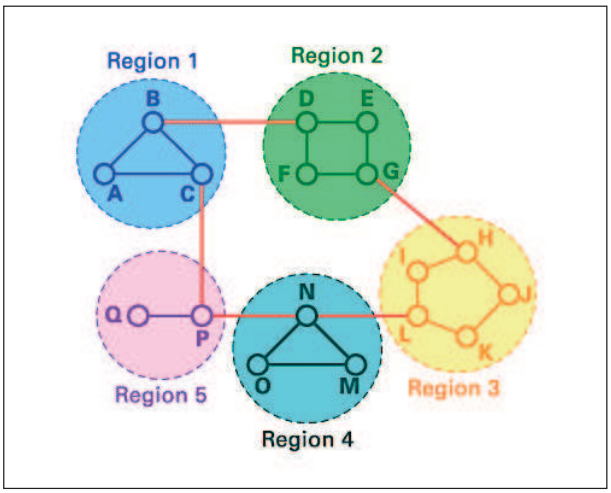
megméri, mennyi időbe telik, míg visszaér hozzá. Ha ezt az időt osztjuk kettővel, akkor egy becslést kaphatunk az útválasztó és a szomszédja között lévő vonal késleltetésére.
A pontosabb becslés érdekében ezt a mérést időnként érdemes megismételni. Ilyenkor a becsült költség mindig a mérések eredményeinek az átlaga.
Ha a vonal késleltetésének becslésekor figyelembe szeretnénk venni a terhelést is, akkor a mérés menete annyiban különbözik az előbbiektől, hogy az útválasztó az ***ECHO*** csomagot a várakozási sor végére teszi (ahol az útválasztón átmenő csomagok is várakoznak), és a „stoppert” onnantól indítja (nem pedig attól az időponttól, amikor az ***ECHO*** csomag távozik az egyik kimeneten).
Látszólag értelmetlennek tűnik az a kérdés, hogy figyelembe vegyük-e a terhelést, vagy sem. Mondhatnánk, hogy persze, vegyük figyelembe, hiszen ha van két ugyanolyan sávszélességű vonal, ahol az egyik teljesen leterhelt, a másikon pedig éppen semmi sem csordogál, akkor az utóbbi legyen része a kijelölt útnak. A logikus választás valóban ez lenne, hiszen így a csomag jóval hamarabb célba érhet.
Vannak azonban helyzetek, amikor kifejezetten hátrányos, ha a késleltetésbe a terhelést is beszámítjuk. Vessünk egy pillantást a *3. ábrára*! Láthatjuk, hogy az alhálózat nyugati részét a keleti résszel a CF és az EI vonal köti össze. Tegyük fel, hogy a legtöbb kelet felé menő csomagot a CF vonalon keresztül irányítjuk, így ott a terhelés sokkal nagyobb lesz, mint az EI-n. Ha a becsléskor a vonal terhelését is megvizsgáljuk, akkor azt fogjuk kapni, hogy a csomagok EI-n keresztül kisebb késleltetéssel jutnak célba. Ezek után a kelet felé tartó forgalom oroszlánrésze az EI-n kerül továbbításra, így az válik terhelté, míg a CF-en a forgalom ritkul. Ne csodálkozzunk, ha a következő becslés eredményeképpen a CF lesz része a legrövidebb utaknak.
A vonalak késleltetésének folyamatos változása miatt az útválasztók forgalomirányító táblázatai is állandóan módosulnak, méghozzá túl gyorsan. Ez nem túlzottan szerencsés dolog, ugyanis a forgalomirányítás nem lesz megbízható, és rengeteg nem várt problémával is szembesülhetünk. Ilyen esetekben tehát nem jó, ha a terhelést is nézzük az egyes vonalakon. Ehelyett megtehetjük például azt, hogy a forgalmat testvériesen megosztjuk a két vonal között, habár így nem biztos, hogy minden csomag a lehető legrövidebb úton fog célba jutni.

Kapcsolatállapot csomagok

Eddig sok újdonságról nem volt szó, a fentiekkel már találkozhattunk valamilyen formában a távolságvektor alapú forgalomirányításnál is. A kapcsolatállapot alapú forgalomirányítás lelke azonban az úgynevezett kapcsolatállapot csomagok, amelyek segítségével az útválasztó megoszthatja társaival az általa összegyűjtött adatokat. Ezek a csomagok tartalmazzák egyrészt a feladót, annak életkorát (lásd később), illetve annak szomszédait, és az azokhoz tartozó késleltetéseket. Ezeket az információkat az alhálózat összes útválasztója kicseréli egymással. A *4. ábrán* láthatunk példát az alhálózaton terjedő kapcsolatállapot csomagokra.
Nem egyszerű az a kérdés, hogy egy útválasztónak mikor érdemes összeállítani és szétküldenie a kapcsolatállapot csomagjait. Általában kétféle megközelítést alkalmaznak.
Az első esetben az útválasztók periodikusan, azaz szabályos időközönként küldik szét az általuk összegyűjtött adatokat.
A másik lehetőség az, hogy a kapcsolatállapot csomagok csak bizonyos események bekövetkeztében jönnek létre, például ha egy vonal megszakad, vagy egy szomszéd útválasztó elhalálozik, illetve megjavul.
A kapcsolatállapot csomagokat az útválasztók az elárasztás módszerével szórják szét az alhálózaton, tehát az átmenő kapcsolatállapot csomagokat az útválasztók az összes kimenetükön továbbküldik (kivéve azt a portot, amelyikről a csomag beérkezett). Mivel ez a módszer exponenciális léptékben gyártja a csomagok duplikátumait, a folyamatot valamiképp kordában kell tartani. Az előző részben bemutattunk erre egy megoldást, miszerint minden csomagnak kell rendelkeznie egy sorszámmal, amely minden csomagküldés alkalmával eggyel nő. Minden útválasztó számon tatja, hogy melyik portról milyen sorszámú csomagok érkeztek eddig. Ha egy olyan csomag jön, amelynek a sorszáma nincs ezen a listán, akkor azt az útválasztó elárasztja, és a sorszámát feljegyzi. A listán szereplő sorszámú csomagok viszont többé nem kerülnek elárasztásra. Az ilyen csomagokat az útválasztók egyből ki is dobják.
Ez ugyan egy jól használható módszer, mégis akadnak vele problémák. Az első és legkézenfekvőbb dolog az, hogy ha a sorszámok egyszer csak átcsordulnak, akkor az egész rendszer dugába dől. (Ha például egy bájttal ábrázolnánk a csomagok sorszámait, akkor a 255. sorszám után a 0 lenne a következő, amelyet egy útválasztó sem fog elárasztani). Erre igazából csak egy megoldás létezik, mégpedig az, hogy a sorszámokat minél több biten ábrázoljuk. Persze a probléma még így is fennáll, viszont például 32 bit esetén a sor-



4. ábra Kapcsolatállapot csomagok



5. ábra Kétszintű hierarchikus forgalomirányítás

számok csak 137 év múlva fordulnának át. (Feltéve ha az útválasztók átlagosan másodpercenként indítanak kapcsolatállapot csomagokat).
Persze az útválasztók nem arról híresek, hogy matuzsálemi korokat éljenek meg, előbb-utóbb mindegyik elromlik, kifagy vagy egyéb katasztrofális eseményekkel kell szembesülnie. Útválasztóink halandósága újabb problémákat szül: elfelejtik, hogy milyen sorszámú kapcsolatállapot csomagokat bocsátott már ki. Ha a sorszámozást a nullánál kezdi, akkor valószínű, hogy csomagjai nem jutnak messzire, hiszen a többi útválasztó elavultnak fogja tekinteni őket.
Az is problémát rejt, hogy az útválasztók lista helyett csak egy számlálót tartanak karban. A számláló aktuális értéke alatti sorszámok jelölik az elavult csomagokat. Ha egy nagyobb sorszámú csomag érkezik, az útválasztó azt elárasztja, majd a számláló értékét a csomag sorszámára növeli. Ez egy rendkívül memóriatakarékos módszer, viszont a bithibákra nagyon érzékeny. Ha ugyanis a sorszámban csak egy bit is megsérül, akár nagyságrendekkel nagyobb számot is kaphatunk. Például a 2-es és a 65538-as sorszám bináris alakja közötti eltérés csupán egy bit. Ha az útválasztó egy ilyen csomagot kapna, akkor 2-től 65538-ig minden csomagot el fog dobni. A probléma megoldásához szükséges, hogy a csomagba a sorszám mellé egy életkor mezőt is definiáljunk, amelynek értéke másodpercenként eggyel csökken. A csomagok életkorát az útválasztók is csökkentik elárasztáskor. Amikor egy csomag életkora eléri a nullát, akkor megsemmisül.
A gyakorlatban a beérkező kapcsolatállapot csomagok nem kerülnek egyből továbbításra, hanem először egy pufferbe kerülnek. Ha ugyanattól a forrásból még egy kapcsolatállapot csomag érkezik, akkor az útválasztó összehasonlítja a két csomag sorszámát. Ha ezek megegyeznek, akkor a második csomag már egy duplikátum, így az eldobható. Ha a sorszám különbözik, akkor a régebben érkezett csomag által hordozott információ már nem érvényes, így azt kell kidobni.
Hogy az algoritmus még robosztusabb legyen – azaz ellenállóbb legyen a nemvárt vonalhibákkal szemben – az útválasztók a megkapott kapcsolatállapot csomagokat nyugtázzák egymásközött. Hasonlóan a kapcsolatállapot csomagokhoz, a nyugták sem kerülnek azonnal elküldésre, először ők is a pufferben várakoznak.

A pufferben a csomagok mellett a forrásuk, a sorszámuk, a koruk és a küldési, illetve a nyugtázási jelzőik is tárolva vannak. Ezek közül az utóbbi kettő szorul még némi magyarázatra. A küldési jelző azt mondja meg, hogy a kérdéses csomagot mely kimeneteken kell továbbítani, a nyugtázási jelző pedig azt határozza meg, hogy a nyugtát kinek kell visszaküldeni.
Mit nyerünk ezzel? Tegyük fel, hogy a 4. ábrán feltüntetett alhálózati topológiában a B útválasztó kétszer is az E útválasztó kapcsolatállapot csomagot. Például először az EAB majd az EFB útvonalon keresztül. Ebben az esetben a csomagot felesleges minden porton elárasztani, elegendő csak a C felé vezető kimeneten. Így az útválasztó a pufferben átírja a csomag küldési jelzőjét úgy, hogy csak a C felé továbbítódjon (a nyugtát viszont A-nak és F-nek is vissza kell küldeni).
Ezzel a technikával csökkenthetjük a kapcsolatállapot csomagok által előidézett forgalomnövekedést.

Az új útvonal kiszámítása

Miután az útválasztók kicserélték információikat, felépíthetik az új forgalomirányító táblázatot. A legrövidebb utak kiszámítása a Dijkstra algoritmus segítségével történik.
Fontos megjegyeznünk, hogy a kapcsolatállapot csomagok révén megkapott információ mérete jóval meghaladja a forgalomirányító táblázat méretét. Gondoljuk csak meg: ha az alhálózatban N darab útválasztó van, és minden útválasztónak K darab szomszédja van, akkor a beérkező adatok tárolásához legalább K*N méretű memóriára van szükség. Ha nagyon nagy a hálózat, akkor ez komoly gond, mivel nem áll rendelkezésre elegendő memória (sőt, az igazán nagy hálózatok esetében a teljes forgalomirányító táblázat sem fér el az útválasztó memóriájában). Nem is beszélve a hálózat méretével arányosan növekvő számításiigényről, amely az új táblázat kiszámításához szükséges.
Ennek ellenére a kapcsolatállapot alapú forgalomirányítást a mai számítógép hálózatokban is széles körben használják. A későbbiekben részletesen megismerkedünk olyan protokollokkal, amelyek erre a forgalomirányítási eljárásra épülnie (például az OSPF protokoll).

Cél	Vonal	Ugrások
A	–	–
B	B	1
C	C	1
D	B	2
E	B	3
F	B	3
G	B	4
H	B	5
I	C	5
J	C	6
K	C	5
L	C	4
M	C	4
N	C	3
O	C	4
P	C	2
Q	C	2

6/a. ábra Az „A” útválasztó táblázata hierarchikus forgalomirányítás nélkül

Hierarchikus forgalomirányítás

Az imént említettük, hogy az alhálózat méretének növekedése maga után vonja a útválasztók forgalomirányító táblázatainak növekedését is. Előbb-utóbb alhálózatunk akkorára duzzadhat, hogy nem lesz olyan útválasztó, amely elegendő memóriával rendelkezne ahhoz, hogy az összes kapcsolóelemhez külön bejegyzést rendelhessen. Ilyenkor a forgalomirányítást ***hierarchikusan*** kell végeznünk, hasonlóan, mint a távbeszélőhálózatok esetében.

A hierarchikus forgalomirányítás esetén az útválasztók tartományokba vannak csoportosítva. Minden útválasztó csak a saját tartományába tartozó kapcsolóelemeket ismeri, a többi régió topológiája rejtve marad számára. Az *5. ábrán* láthatunk egy példát az alhálózat hierarchikus felépítésére. Az alhálózat összesen 17 kapcsolóelemet tartalmaz, ami azt jelenti, hogy ha nem lenne a forgalomirányítás hierarchikusan szervezett, akkor minden útválasztónak egy 17 bejegyzést tartalmazó útvonalválasztó táblázatot kéne észben tartania *(6/a ábra)*. Ellenkező esetben a tartományokat egy-egy csomópontba „sűríthetjük”, így az útválasztóknak a lokális kapcsolóelemeken kívül csak tartományonként egy-egy bejegyzést kell tárolniuk.

A *6. ábrán* láthatunk példát az első tartományban lakó Az útválasztó forgalomirányító táblázatára. Az első esetben nincs hierarchikus forgalomirányítás, így a táblázat 17 sort tartalmaz. A második esetben láthatjuk, hogy a második tartomány felé igyekvő csomagok mindig a BD vonalon keresztül haladnak, míg a többi régió felé a CP vonal vezet.

Cél	Vonal	Ugrások
A	–	–
B	B	1
C	C	1
2. tartomány	B	2
3. tartomány	C	4
4.tartomány	C	3
5. tartomány	C	2

6/b. ábra Az „A” útválasztó táblázata hierarchikus forgalomirányításnál

A táblázatok méretkülönbsége szembeötlő, a táblázat a hierarchikus forgalomirányítás esetében 17 helyett csupán 7 bejegyzést tartalmaz. Általánosan elmondható, hogy minél nagyobb a tartományok száma a kapcsolóelemek számához viszonyítva, annál kisebb méretűek lesznek az útválasztók forgalomirányító táblázatai.

Mivel semmi sincs ingyen, a helyspórolásnak is megvan a maga ára: le kell mondanunk arról, hogy a csomagok minden esetben a lehető legrövidebb úton keresztül jutnak célba. Maradva az 5. ábrán látható példánál, ha az Az útválasztótól a harmadik tartományban található H útválasztóhoz szeretnénk eljutni, akkor kicsit kerülni fogunk. Ugyan leghamarabb a második régión keresztül juthatnánk el, az A útválasztó mégis az ötödik tartomány felé irányít minket. Miért? A válasz egyszerű: lehet, hogy speciálisan a H útválasztó felé közelebb lenne, ha a BD vonalon indulnánk el, de a harmadik tartomány legtöbb útválasztója mégis a CP vonalon keresztül érhetőek el a leghamarabb. (Megmutatható azonban, hogy a hierarchikus forgalomirányításból adódó többletutak általában nem számottevőek, legalábbis bőven az elfogadható határon belül vannak. Persze kivételek mindig vannak).

Az 5. ábrán egy kétszintű hierarchiára láthattunk példát. Az igazán nagy hálózatok esetén azonban ez sem segít a dolgon, a forgalomirányító táblázatok mérete továbbra is elfogadhatatlanul nagyok maradnak. A tartományokat így tovább kell bontanunk zónákra, a zónákat kerületekre, a kerületeket csoportokra, és így tovább.

De vajon hány hierarchiai szintet érdemes definiálni? Ennek kiszámításához ismert egy összefüggés, amely szerint az optimális hierarchiai szintek száma ln N, ahol N az alhálózatban lévő útválasztók száma. Ebben az esetben minden útválasztónak e * ln N számú bejegyzést kell tárolnia a memóriájában.

A következő részben továbbra is a forgalomirányítással foglalkozunk, de most már azt vizsgáljuk, mi a helyzet, ha a gépek nem otthonülő életmódot folytatnak, hanem összevissza mozognak. Ezenkívül szó lesz még a többesküldéses, illetve az adatszórásos forgalomirányításról.

Garzó András
garzo@interware.hu