A HTB várakozásisor-kezelési elv elemzése

Vajon képes a Linux úgy garantálni a szolgáltatásminőséget, hogy egyszerre álljon rendelkezésünkre a kívánt sávszélesség, a megadott határt mégse lépjük túl? Kipróbáltuk.

A ***HTB (Hierarchical Token Buckets, hierarchikus zseton vödrök)*** várakozásisor-kezelő elv – a linuxos forgalomszabályozó megoldások egyik eleme – olyan eszköz, mely egyrészt ***szolgáltatásminőségi (Quality of Service, QoS)*** funkciókat biztosít, másrészt alkalmas a ***TCP*** alapú forgalom kifinomult szabályozására. Írásomban áttekintem a sorkezelő elv elemeit, valamint több előzetes teljesítményteszt eredményeit is ismertetem. Ezek elvégzéséhez különféle linuxos környezeteket állítottunk össze, a forgalomkeltést pedig egy Ixia készülékre bíztuk. A próbák során kimutattuk, hogy az átviteli sebesség szabályzásának pontossága befolyásolható, illetve a sávszélesség körülbelül egy 2 Mbps-os tartományon belül szabályozható. A próbák igazolták a ***HTB*** sorkezelő algoritmusok teljesítményét és pontosságát, valamint rámutattak bizonyos, a forgalomkezelés javítására alkalmas módszerekre.

A forgalomszabályozási eljárások akkor jutnak szerephez, amikor az adott ***IP***-csomag már bekerült a továbbításra váró csomagok adott kimenő felülethez rendelt várakozási sorába, ám az illesztőprogram még nem kezdte meg tényleges továbbítását. Az ***1. ábrán*** látható, hogy a forgalomszabályozási döntések meghozatalának helyszíne hogyan viszonyul a fizikai Ethernet összeköttetésen való továbbításhoz és a szállítási rétegbeli protokollokhoz, mint az ***UDP*** és a ***TCP***.

A ***Linux*** rendszermag ***Alexey Kuznetsov*** által megvalósított forgalomszabályozó szolgáltatása négy fő összetevőből áll: várakozásisor-kezelő elvek, szolgáltatási osztályok, szűrők és házirendkezelés.

A várakozásisor-kezelő elvek a sorba állított ***IP***-csomagok kezelésének módját megadó szoftveres eljárások. Minden hálózati készülék ismer valamilyen sorkezelő elvet, ezek közül a legelterjedtebb talán a ***FIFO (First In, First Out, elsőként érkező elsőként távozik)*** algoritmusra alapuló elv, melynél a csomagok érkezésük sorrendjében kerülnek be a várakozási sorba, mégpedig olyan sebességgel, amilyennel a sorhoz tartozó készülék képes elküldeni őket. A ***Linux*** jelenleg is számos sorkezelő elvet támogat, és újabbak hozzáadására is biztosít lehetőséget.

A sorkezelő algoritmusok részletes leírását az interneten ***„Iproute2+tc Notes”*** (lásd a forrásokat) címmel lehet megtalálni. A ***HTB*** elv a hozzárendelt szolgáltatási osztályok soraiba helyezett csomagok kezelésére a ***TBF*** algoritmust alkalmazza. A ***TBF*** algoritmus forgalmi házirendkezelési és forgalomformálási (traffic-shaping) képességekkel rendelkezik. A ***TBF*** algoritmus részletes leírása a ***Cisco IOS Quality of Service Solutions Configuration Guide*** útmutatójában (lásd: ***„Policing and Shaping Overview”***, A házirendkezelés és a forgalomformálás áttekintése) szerepel.

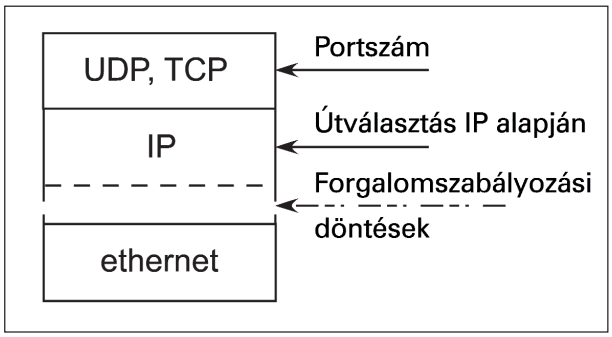
Egy-egy szolgáltatási osztály bizonyos házirendszabályokat határoz meg, ilyen a maximális sávszélesség vagy a maximális löket, és a sorkezelő elvet alkalmazza ezen szabályok betartatására. Egy-egy sorkezelő elv és osztály összetartoznak. Az adott osztály által előírt szabályokat egy előre megadott várakozási sorral kell összerendelni. A legtöbb esetben minden osztálynak saját sorkezelő elve van, de előfordulhat, hogy több osztály is osztozik ugyanazon a várakozási soron. A sorkezelő megoldásoknál az adott osztályhoz tartozó házirend összetevők általában eldobják a megadott határértéken túli csomagokat (lásd: ***„Policing and Shaping Overview”***).

A szűrők a sorkezelő elv által alkalmazott szabályokat adják meg. A sorkezelő elv ezen szabályok alapján dönti el, hogy melyik osztályhoz kell rendelnie a bejövő csomagokat. Minden szűrőnek megadott fontossága, prioritása van. A szűrők fontosságuk szerint növekvő sorrendbe vannak rendezve. Amikor egy sorkezelő elv kap egy csomagot sorba állításra, akkor megpróbálja egyeztetni azt a megadott szűrők valamelyikével. Az egyezés keresése a lista szűrőinek megvizsgálásával történik, kezdve a legnagyobb fontosságúval. Minden osztályhoz vagy sorkezelő elvhez egy vagy több szűrő tartozhat.

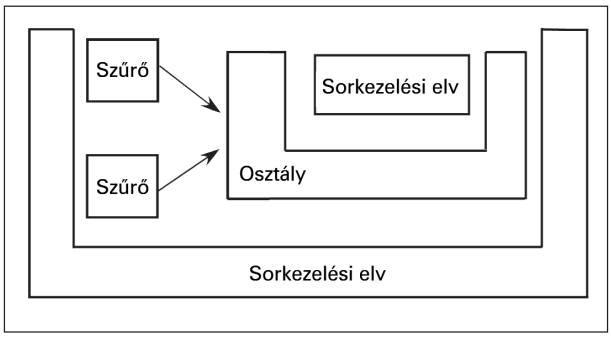
A házirendkezelő összetevők biztosítják, hogy a forgalom nagysága ne haladja meg a megadott sávszélességet. A házirendkezelő döntések a szűrőkön és az osztály alapon megadott szabályokon alapulnak. A ***Linux*** forgalomszabályozó alrendszerének összetevői között fennálló kapcsolatokat a ***2. ábra*** tartalmazza.

A TC segédprogram és a HTB definíciók

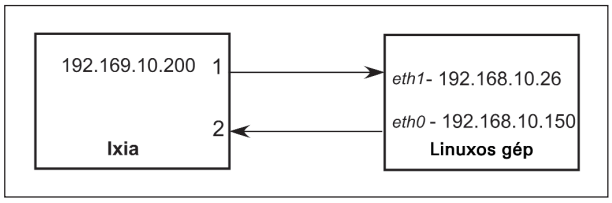
A ***TC*** egy felhasználói program, segítségével várakozási sorokat, osztályokat és szűrőket lehet létrehozni, illetve hozzá lehet ezeket rendelni a megfelelő kimenő felülethez (lásd: ***„tc-Linux QoS control tool”*** a források között). A szűrőket az irányítótábla, ***u32*** osztályozók és ***TOS*** osztályozók alapján lehet összeállítani. A program ***netlink*** foglalatokon



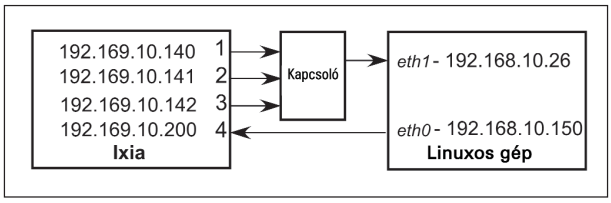
1. ábra A rendszermag azt követően hozza meg forgalomszabályozási döntéseit, hogy a csomag bekerült a küldési várakozási sorba



2. ábra A Linux forgalomszabályzó alrendszerében sorkezelő elveket, osztályokat, szűrőket és házirendkezelést találunk



3. ábra Az 1. tesztmodell összeállítása



4. ábra Az 2. tesztmodell összeállítása

keresztül tartja a kapcsolatot a rendszermag hálózatkezelő alrendszerének eljárásaival. Az ***1. táblázatban*** a három fő műveletet és a megfelelő *TC* parancsokat foglaltam össze. A *TC* használatáról a ***HTB Linux Queuing Discipline User Guide*** útmutató tartalmaz részletesebb tájékoztatást.
A *HTB* használata az egyik mód arra, hogy kézben tartsuk az egyes összeköttetések kimenő sávszélességének használatát.

1. táblázat *A TC segédprogrammal elvégezhető műveletek és a megfelelő parancsok*

TC művelet	Parancs
tc qdisc	Sorkezelő elv létrehozása
tc filter	Szűrő létrehozása
tc class	Osztály létrehozása

A *HTB* használatához azt osztályként és a sorkezelési elv típusaként kell megadni. A *HTB* a forgalmat a *TBF* algoritmus segítségével alakítja, mely független az alsóbb réteg sávszélességétől. Kizárólag a gyökér sorkezelő elvet szabad *HTB* típusként megadni, az összes többi osztálypéldány a *FIFO* sort (alapértelmezett) használja. A sorkezelő folyamat mindig a gyökér szinten indul, majd a szabályok alapján dönti el, hogy melyik osztálynak kell megkapnia az adatokat. Az osztályfa bejárása egészen addig folytatódik, míg egyező levélosztályt nem sikerül találni (lásd: *„Hierarchical Token Bucket Theory”*).

Tesztelés

A *HTB* pontosságát és teljesítményét a következő hálózati készülékek segítségével próbáltuk ki: egy darab ***Ixia 400*** forgalomkeltő készülék, egy darab ***10/100 Mbps Ethernet*** terhelő modullal ***(LM100TX3)***, továbbá egy darab ***Pentium 4 PC*** (1 GB RAM, 70 GB-os merevlemez), ezen a Linux rendszermag 2.6.5-ös változata futott. Két próbamodellt terveztünk, egyet a házirendkezelés pontosságának, a másikat a sávszélesség megosztásának ellenőrzésére.
Az első modellt (***3. ábra***) meghatározott osztály házirendkezelési pontosságának ellenőrzésére használtuk. Az ***Ixia 1***-es kapuján keltett forgalom egy vagy több folyamot alkotott, célállomása a ***192.168.10.200 IP***-cím volt. A linuxos gép a csomagokat egy statikus útvonallal az eth0 felületre irányította, majd visszaküldte őket az ***Ixia 2***-es kapujára.
A forgalomszabályozási előírásokat az eth0 felületre léptettük életbe. Az elemzéseket teljes egészében az ***Ixia 2***-es kapuján kapott forgalom alapján végeztük.
A második modellt (***4. ábra***) annak vizsgálatára használtuk, hogy két, azonos osztályba tartozó adatfolyam sávszélességének megosztása hogyan történik. Ebben az esetben további két ***Ixia*** kaput vettünk igénybe az adatok küldésére.
Az ***Ixia*** 1., 2. és 3. kapuja – egy-egy adatfolyamot használva – egyaránt mesterségesen keltett forgalmat továbbított a ***192.168.10.200***-as *IP*-címre. A linuxos gép ezeket a csomagokat egy statikus útvonal alapján az eth0 felületre irányította, majd visszaküldte őket az ***Ixia*** 2-es kapujára. A forgalomszabályozási szabályok ebben az esetben is az eth0 felületre vonatkoztak. Minden elemzést az ***Ixia*** 2-es kapujára beérkezett forgalom alapján végeztünk.

Az Ixia készülék beállításai és korlátai

A küldő kapuk minden próbánál összefüggő csomaglöketeket küldtek, megadott sávszélességgel. Az ***Ixia 10/100 Mbps*** sebességű ***Ethernet (LM100TX3 modellszámú)*** terhelő modulja négy különböző kapuval rendelkezik, ezek mindegyike legfeljebb 100 Mbps sebességű küldésre képes. Az ***Ixia*** terhelőmodulja képes volt ugyan arra, hogy egy-egy kapun több adatfolyamot állítson elő, de nem tudta összekeverni

egymással az adatfolyamokat, és egyszerre csak egy adatfolyamot szolgáltatott. A korlátozás oka, hogy ütemezője körbeforgó elven működik. Először elküldi az X adatfolyam egy bájtlöketét, majd továbblép a következő adatfolyamra, és az Y adatfolyamon belül is küld egy löketet.
Ha egy adatfolyamból meghatározott sávszélességet akarunk előállítani, mely ráadásul az adott kapuhoz hozzárendelt adatfolyamcsoport tagja, akkor finomhangolnunk kell az Ixia bizonyos beállításait. Ezek a beállítások – illetve szerepük – következők:

- ***Burst (löket)***: az egyes folyamokban a következő folyamra való továbblépés előtt elküldött csomagok száma
- ***Packet size (csomagméret)***: a folyamot alkotó egyes csomagok mérete
- ***Total bandwidth (teljes sávszélesség)***: az összes adatfolyam által felhasznált sávszélesség

Az ***Ixia*** beállításait a ***2. táblázatban*** foglaltuk össze.
A cél az volt, hogy meghatározzuk, mi az a löketméret, amelynél az egyes adatfolyamokhoz kívánt sávszélességet el tudjuk érni. Mivel mindhárom adatfolyam ugyanazt a fizikai vonalat használta, az adatok továbbításának módja az ***5. ábrán*** láthatóhoz volt hasonló.
A beállítások közötti összefüggéseket az alábbi egyenletekkel írhatjuk le:

$$T_c = \mathrm{Sum}(B_{s\text{-}i} * 8 * P_{s\text{-}i}) / T_b$$
$$N_c = 1/T_c$$
$$B_{n\text{-}i} = N_c * P_{s\text{-}i} * 8 * B_{s\text{-}i}$$

Az egyenletekben használt változók jelentését a ***3. táblázat*** tartalmazza.
Feltételezve, hogy a csomagméret minden adatfolyam esetében azonos, ahogy a példában is, csak a löketméret meghatározása marad hátra.
Mivel minden adatfolyam ugyanazon a sávszélességen osztozik, az elvárt löketértékeket az elvárt sávszélességek közötti arányok vizsgálatával kaphatjuk meg, a $B^{s\text{-}i} = B^{n\text{-}i}$ egyenlet alapján. Ez az érték meglehetősen nagy is lehet, szükség esetén osztásokkal hozhatjuk kezelhetőbb formára. Ha különböző csomagméreteket akarunk hozzárendelni az egyes adatfolyamokhoz, akkor a löketméretek szükség szerint ismételt módosításával kaphatjuk meg az egyes adatfolyamokban a kívánt sávszélességeket. Táblázatkezelő program segítségével egyszerűbben, könnyebben végezhetjük el a több sávszélességre is kiterjedő számításokat.

Tesztek és eredmények

A HTB beállításainak megadásakor a következő tc class parancsokat használtuk az elvárt eredmények eléréséhez:

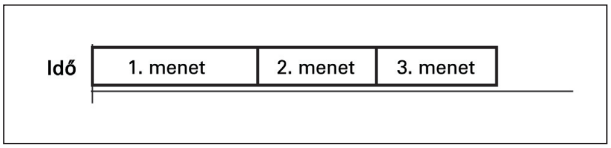
- *rate* = adott osztály legfeljebb ennyi sávszélességet használhat el úgy, hogy nem más osztályoktól veszi azt kölcsön

2. táblázat ***Az Ixia készülék beállításai***

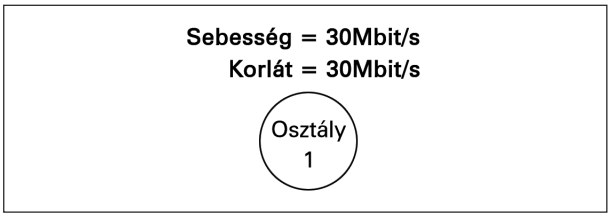
Adatfolyam	Mesterségesen keltett sávszélesség	Csomagméret	Löketméret
1	15 Mbps	512 B	150
2	10 Mbps	512 B	100
3	2 Mbps	512 B	20
Összes	27 Mbps	–	–

3. táblázat ***A beállítások összefüggéseit jellemző egyenletekben használt változók***

Beállítás	Jelentés
T_c	Az 1-i sorszámú löketek elküldéséhez szükséges idő összege másodpercben mérve ($T_c1 + T_c2 + T_c3+...$)
$B_{s\text{-}i}$	Az i adatfolyam löketeit alkotó csomagok száma
$P_{s\text{-}i}$	Az i adatfolyamban elküldött csomag mérete
T_b	Az összes adatfolyam által lefoglalt sávszélesség (bit/másodperc)
N_c	A T_c löketek másodpercenkénti száma
$B_{n\text{-}i}$	Az i adatfolyam elvárt sávszélessége (bit/másodperc)



5. ábra Az adatok áramlása az átviteli vonalon az Ixia készülék és a tesztelt Linux rendszer között



6. ábra 1. teszt – egy-egy be- és kimenő adatfolyam

- *ceiling* = adott osztály legfeljebb ennyi sávszélességet használhat el; így szabályozható, hogy az osztály mekkora sávszélességet vehet kölcsön másoktól
- *burst* = maximális sebességgel legfeljebb ennyi adatot lehet elküldeni, mielőtt a következő osztály kiszolgálására lépnénk tovább
- *cburst* = az átviteli közeg sebességével legfeljebb ennyi adatot lehet elküldeni, mielőtt a következő osztály kiszolgálására lépnénk tovább

4. táblázat *Az 1. tesztmodellel kapott eredmények*

Burst (bájt)	Cburst (bájt)	Csomagméret (bájt)	Bejövő sávszélesség (Mbps)	Kimenő sávszélesség (Mbps)
Alapértelmezett	Alapértelmezett	128	40	33,5
Alapértelmezett	Alapértelmezett	64	40	22
Alapértelmezett	Alapértelmezett	64	32 (Max)	29,2
15 k	15 k	64	32 (Max)	30
15 k	15 k	512	32 & 50 & 70	25,3
15 k	15 k	1500	32 & 50 & 70	25,2
18 k	18 k	64	32 (Max)	29,2
18 k	18 k	512	32 & 50 & 70	30,26
18 k	18 k	1500	32 & 50 & 70	29,29

5. táblázat *A Burst/Cburst és Rate értékek közötti kapcsolat*

Burst (bájt)	Cburst (bájt)	Csomagméret (bájt)	Bejövő sávszélesség (Mbps)	Kimenő sávszélesség (Mbps)	Hozzárendelt sebesség (Mbps)
9 k	9 k	64	32	17,5	15
9 k	9 k	512	32	15,12	15
9 k	9 k	1500	32	15,28	15
4,8 k	4,8 k	64	32	8,96	8
4,8 k	4,8 k	512	32	8,176	8
4,8 k	4,8 k	1500	32	8	8
3 k	3 k	64	32	17,5	15
3 k	3 k	512	32	15,12	15
3 k	3 k	1500	32	15,28	15

Az internetes világ forgalmának túlnyomó része ***TCP*** alapú, ezért a próbák során is a datagramokra jellemző – 64 bájtos (TCP Ack), 512 bájtos (FTP) és 1500 bájtos – csomagméreteket választottunk.

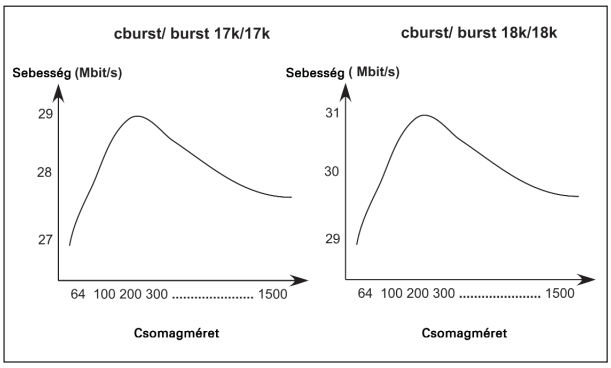
Az első tesztmodell

A ***4. táblázatban*** foglalt eredmények alapján a következőket jelenthetjük ki:

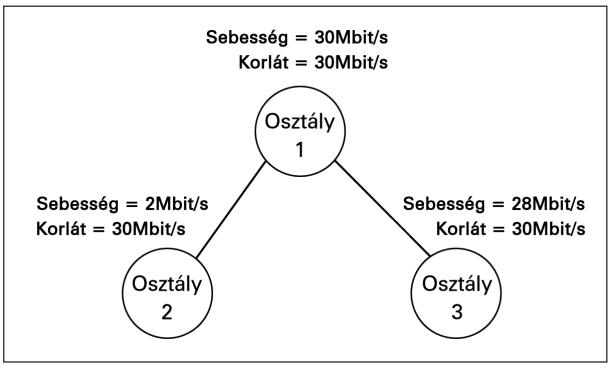
- Egy linuxos gép 64 bájtos csomagokból álló, összefüggő adatfolyamok esetén körülbelül 34 Mbps sebességgel képes továbbítani (egyik interfészen fogadni, a másikon küldeni) a forgalmat.
- A legjobb átlagos szabályzási pontosságot 18 k / 18 k burst/cburst érték mellett kapjuk.
- A löket (burst) érték és az elvárt sebességérték között lineáris kapcsolat van, ez a tesztek során egyértelművé válik.
- A kimenő felületre erőltetett sávszélesség nem befolyásolja az eredmények pontosságát.

A ***7. ábra*** a csomagméret és a kimenő sávszélesség közötti, különféle csomagméretekkel elvégzett tesztek során kapott eredmények alapján megállapított kapcsolatot szemlélteti. A 4. táblázatban és a 7. ábrán szereplő eredményekből két tanulságot vonhatunk le: egyrészt az átvitel szabályzásának pontosságát a ***cburst/burst*** értékek módosításával lehet befolyásolni, másrészt a pontossági sávszélesség-tartomány 2 Mbps széles, amennyiben a csomagméret 64 és 1500 bájt közötti. Azt, hogy a ***burst/cbursts*** értékek és az átviteli sávszélesség (rate) között lineáris összefüggés van, többféle ***burst*** és ***cburst*** érték vizsgálatával igazoltuk. Az ***5. táblázat*** a próbák során kapott adatminták fontosabbjait tartalmazza. A kiindulási érték 18 k / 18 k volt. A ***burst/cburst*** értékeket a ***cburst/burst*** (Kb) = 18/(30M/hozzárendelt sebesség) képlet alapján kaptuk. Az 5. táblázatban szereplő eredmények szerint a ***cburst/burst*** értékeket dinamikusan lehet megadni a sebességértékhez, mint amikor lineáris kapcsolatot feltételezünk.

A ***6. táblázat*** az egyszintű öröklődésre mutat példát. A 2. és a 3. osztály örökli az 1. osztály sebességkorlátját (30 Mbps).



7. ábra A csomagméret és a kimenő sávszélesség kapcsolatának grafikus elemzése



8. ábra 2. teszt – három bejövő és egy kimenő adatfolyam

6. táblázat *A 2. teszt eredményei*

Adatfolyam	Burst (bájt)	Cburst (bájt)	Csomagméret (bájt)	Bejövő sávszélesség (Mbps)	Kimenő sávszélesség (Mbps)	Osztály
1	17 k	17 k	64	15	12,7	3
2	17 k	17 k	512	20	17,1	3
3	1 k	1 k	512	4	2,01	2
Összes	18 k	18 k	–	39	31,8	–

7. táblázat *A 3. teszt eredményei*

Adatfolyam	Burst (bájt)	Cburst (bájt)	Csomagméret (bájt)	Bejövő sávszélesség (Mbps)	Kimenő sávszélesség (Mbps)	Osztály
1	1 k	1 k	512	5	2,04	2
2	6 k	6 k	6	15	11,326	4
3	3 k	3 k	64	10	5,67	5
4	7,8 k	7,8 k	512	20	13,02	6
Összes	18 k	18 k	–	50	32,05	–

Ennél a tesztnél a gyermekosztályok felső sebességkorlátja megegyezik a szülő korlátjával, vagyis a 2. és a 3. osztály legfeljebb 30 Mbps sávszélességet vehet kölcsön. A többi osztályhoz tartozó ***cburst/burst*** értékeket lineáris összefüggést feltételezve, az elvárt sávszélességek alapján számítottuk ki.

A 6. táblázat azt szemlélteti, hogy egyszintű öröklésnél hogyan jellemezhető a lineáris összefüggés. A teszt során a bejövő adatfolyam 39 Mbps sávszélességű, folyamatos forgalmat ad, a teljes kimenő sávszélesség pedig 31,8 Mbps.

A *7. táblázat* a kétszintű öröklést szemlélteti. A 2. és a 3. osztály örökli az 1. osztály sebességkorlátját (30 Mbps). A 4., az 5. és a 6. osztályok a 3. osztály sebességkorlátját öröklik (28 Mbps), és saját sebességhatáraik szerint osztoznak rajta. A tesztben a gyermekosztályok felső sebességkorlátja megegyezik a szülő korlátjával, vagyis a 4., az 5. és a 6. osztály legfeljebb 28 Mbps sávszélességet vehet kölcsön. A többi osztályhoz tartozó ***cburst/burst*** értékek kiszámítása lineáris összefüggést feltételezve, az elvárt sávszélességek alapján történt.

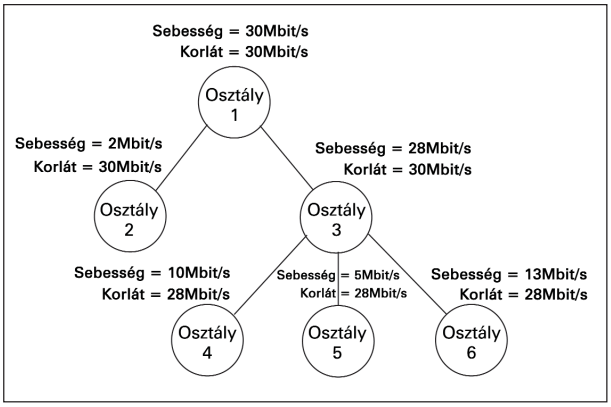
A 7. táblázatban foglalt eredmények alapján megállapíthatjuk, hogy a lineáris kapcsolat kétszintű öröklésnél is fennáll. A teszt során a bejövő kapu 50 Mbps sávszélességű, folyamatos forgalmat fogad, a teljes kimenő sávszélesség pedig 32,05 Mbps.

A második tesztmodell

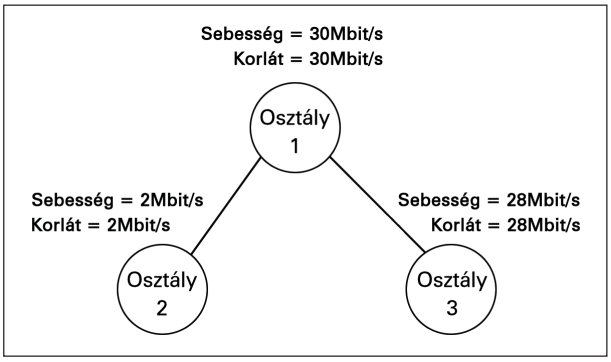
Ahogy a *8. ábrán* is követhető, a sávszélesség egyenlően oszlik meg az adatfolyamok között, amennyiben ezek azonos számú bájtot küldenek el, illetve azonos osztályhoz tartoznak. Egy másik kísérlet, amelynél az 1. adatfolyam bejövő sávszélessége nagyobb volt, mint a 2. és a 3. adatfolyamé, kimutatta, hogy az 1. adatfolyamnak a kimenő sávszélessége is nagyobb volt, mint a 2. és a 3. adatfolyamé. Az eredményekből arra következtethetünk, hogy ha több ada-

8. táblázat *Az 1. teszt eredményei*

Adatfolyam	Burst (bájt)	Cburst (bájt)	Csomagméret (bájt)	Bejövő sávszélesség (Mbps)	Kimenő sávszélesség (Mbps)	Osztály
1	1 k	1 k	512	5	0,650	2
2	1 k	1 k	512	5	0,600	2
3	1 k	1 k	512	5	0,568	2
Összes	18 k	18 k	–	15	1,818	–



9. ábra 3. teszt – négy bejövő és egy kimenő adatfolyam



10. ábra 1. teszt – három bejövő, egy kimenő adatfolyam

tot küldünk egy meghatározott adatfolyamban, akkor az az azonos osztályba tartozó egyéb folyamokhoz képest több csomag továbbítására lesz képes.

Összefoglalás

Az ismertetett kísérletek elvégzése az egyik lehetőség a *HTB* pontosságának és teljesítményének vizsgálatára. Bár a folyamatosan érkező, adott forgalmat jelentő löketek nem feltétlenül jellemzőek a valós életbeli adatforgalomra, vizsgálatukkal mégis megkaphatjuk a kiindulási pontokat a *HTB* osztályok és jellemzőik megadásához.
A kísérletek eredményeit a következő kijelentésekkel összegezhetjük:

- Egy linuxos gép 64 bájtos csomagokból álló, összefüggő adatfolyamok esetén körülbelül 34 Mbps sebességgel képes továbbítani (egyik interfészen fogadni, a másikon küldeni) a forgalmat. A korlát azért jelentkezik, mert az *Ethernet* illesztőprogram által fogadott vagy elküldött csomagok mindegyike egy megszakítást vált ki. A megszakítások kezelése processzoridőt igényel, ami értelemszerűen akadályozza a rendszer egyéb folyamatainak működését.
- Ha a forgalmat 30 Mbps sebességre állítjuk, 18 k / 18 k ***cburst/burst*** érték mellett kapjuk a legjobb átlagos pontosságot.
- A ***löket (burst)*** érték és az elvárt sebességérték között lineáris kapcsolat van. A 30 Mbps sebességhez tartozó ***cburst/burst*** értékek megfelelő kiindulási alapot szolgáltatnak a más sebességekhez tartozó ***burst*** értékek meghatározásához.
- Az átvitel pontosságát a ***cburst/burst*** értékek módosításával lehet szabályozni. A pontos szabályozás sávszélesség-tartománya 64-1500 bájt közötti méretű csomagoknál körülbelül 2 Mbps.
- A sávszélesség egyenlően oszlik meg az adatfolyamok között, amennyiben azok azonos számú bájtot küldenek el, illetve azonos osztályhoz tartoznak.

Linux Journal 2005. március, 131. szám

Yaron Benita az izraeli Jeruzsálemben született, jelenleg San Franciscoban, Kaliforniában él. A Prediwave CMTS szoftverigazgatója, munkája során elsősorban hálózatokkal és beágyazott rendszerekkel foglalkozik. Nős, van egy hat hónapos, gyönyörű kislánya. A yaronb@prediwave.com vagy az ybenita@yahoo.com címen érhető el.

KAPCSOLÓDÓ CÍMEK

- snafu.freedom.org/linux2.2/iproute-notes.html
- tcng.sourceforge.net
- luxik.cdi.cz/~devik/qos/htb/manual/userg.htm
- luxik.cdi.cz/~devik/qos/htb/manual/theory.htm
- www.cisco.com
- www.rns-nis.co.yu/~mps/linux-tc.html