A FORGALOM-FOLYAM ANALÍZIS ÉS AZ ESEMÉNYVEZÉRELT DISZKRÉT IDEJŰ SZIMULÁCIÓ KOMBINÁCIÓJA ÉS EGYÜTTMŰKÖDÉSE

Lencse Gábor
Távközlési Tanszék
Széchenyi István Egyetem
9026 Győr, Egyetem tér 1.
lencse@sze.hu

Kulcsszavak: *forgalom-folyam analízis, diszkrét idejű szimuláció, kommunikációs hálózatok, teljesítőképesség vizsgálat, forgalom modellezés*

A forgalom-folyam analízis kommunikációs rendszerek teljesítőképesség vizsgálatára alkalmazható új módszer. Közelítő eredményeket ad, és potenciálisan gyorsabb a diszkrét idejű szimulációnál. A két módszer kombinációja ígéretes módszernek tűnik abban az esetben, amikor egy rendszer teljesítőképességét a környezetével együtt kell megvizsgálni. A módszert különösen jól lehet alkalmazni kommunikációs hálózatok kritikus részeinek vizsgálatára.

BEVEZETÉS

A forgalom-folyam analízis (Traffic-Flow Analysis – TFA) [6] egy szimulációhoz hasonló módszer kommunikációs hálózatok forgalmi viszonyainak gyors vizsgálatára. A diszkrét idejű szimulációval ellentétben nem modellezi minden egyes csomag útját a hálózatok, hanem statisztikákat használ az alkalmazások hálózati terhelésének modellezésére. A módszer először szétosztja a forgalmat (a statisztikákat) a hálózatban és egy második lépésben számítja ki az összes vonalon és csomóponton kialakuló tényleges forgalmi viszonyokat. Az eredmények közelítőek, de kielégítően jellemezhetik a hálózat forgalmi viszonyait.

Az eseményvezérelt diszkrét idejű szimuláció (event-driven Discrete Event Simulation – DES) rendszerek részletes és pontos vizsgálatára alkalmazható, de ha a rendszer túl nagy és összetett, az események száma olyan nagy lehet, hogy a végrehajtás elfogadhatatlanul sokáig tart még szuperszámítógépeken is. Az eseményvezérelt DES algoritmusa olyan, hogy a párhuzamosítás nem könnyű feladat, és az elérhető gyorsulás gyakran erősen korlátos. [2]

Ez a cikk a két módszer kombinációjával foglalkozik. Sokszor kerülünk szembe a következő problémával: adott egy kommunikációs hálózat, amelynek van egy kritikus része. Például egy X.25 hálózat, amely ATM és POS terminálokat szolgál ki, és azt szeretnénk ellenőrizni, hogy mi történik, ha a szervernél egy fontos vonal meghibásodik. Ezeknek a feladatoknak a közös jellemzője, hogy van egy kritikus része a hálózatnak (mint a szerver közvetlen környezete a fenti példában), amelyet pontosan kell modelleznünk, és ott van a hálózat többi része, amelyet nem hagyhatunk el, mert abból jön a kritikus rész forgalmi terhelése. Ha diszkrét idejű szimulációt használunk az ilyen hálózatok vizsgálatára, a következő ellentmondással kerülünk szembe: bár bennünket a kritikus rész érdekel, a szimulációnkban az események döntő többsége a hálózat egyéb részén fordul elő (mert az tartalmazza majdnem az összes csomópontot, vonalat és a forgalom generátorokat). A javasolt megoldás alapötlete a következő: használjunk DES-t

a kritikus rész pontos vizsgálatára és TFA-t a hálózat többi részére. Ennek a megoldásnak létjogosultságát a következők támasztják alá: a kritikus rész modellezése kellően pontos, de a számítási kapacitást nem vesztegetjük olyan események végrehajtására, amelyek egyenként teljesen érdektelenek számunkra, csak bizonyos statisztikai jellemzőik befolyásolják a kritikus rész működését.

A két módszer kombinációjához „meg kell őket tanítani arra, hogy ugyan azt a nyelvet beszéljék", hogy képesek legyenek egymással információt cserélni. Ez azt jelenti, hogy szükségünk van TFA *statisztikái* és a DES *eseményei* közötti konverziós módszerekre mindkét irányban. Ezen kívül van egy további probléma is: egyrészt a virtuális idő (modell idő) nagyon fontos az eseményvezérelt diszkrét idejű szimulációban, másrészt a TFA nem foglalkozik az idő múlásával, a TFA arra használható, hogy a rendszer egy adott állapotáról pillanatfelvételt készítsen. A TFA implementációnk – ami az Iminet nevű hálózati szakértői rendszer része – egy eseményvezérelt DES szimulációs kernelre épült, és a szimulátor virtuális idejét a TFA belső céljaira használja (a forgalom térbeli eloszlásának meghatározása során – ezt később részletesen elmagyarázzuk).

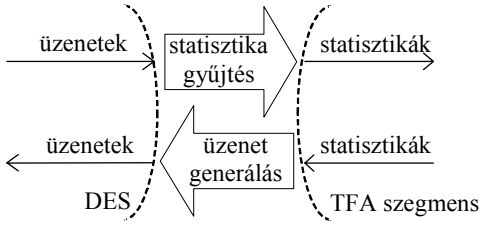
Először a statisztikák és az üzenetek közötti konverzióval foglalkozunk, és utána fogjuk megoldani a virtuális idő problémáját.

A STATISZTIKÁK ÉS AZ ÜZENETEK KÖZTI KÉTIRÁNYÚ KONVERZIÓ

Általános megközelítés

Tegyük fel, hogy a vizsgálandó hálózatunk két részre osztható: az egyikre a DES-t a másikra a TFA-t szeretnénk használni. Több DES és TFA szegmensből álló rendszer visszavezethető erre az egyszerű esetre.

Az 1. ábrán látható a szegmensek határán a statisztikák és üzenetek között szükséges konverzió alapötlete. Amikor a forgalom információ a DES szegmensből a TFA szegmens fele halad, a reprezentációs mód üzenetről statisztikára változik. Ez azt jelenti, hogy össze kell gyűjtenünk az üzenetfolyam megfelelő statisztikai jellemzőit, és szükség lehet a gyűjtött statisztikáknak a TFA szegmensben használt típusú statisztikákra való konverziójára is. Az ellenkező irányban, ahol a forgalom az TFA szegmensből az DES szegmens felé halad, üzeneteket kell generálnunk a TFA szegmens statisztikái (a TFA terminológiában *forgalom modell*) alapján. Az üzenetfolyam jellemzői, mint például az üzenetek forrás, cél, hossz, és érkezési időköz eloszlása, ugyan azt az információt fogják hordozni, ami a TFA szegmensből érkező statisztikákban van kódolva.



1. ábra - A DES és a TFA szegmensek összekapcsolásának alapötlete

Hogyan végezhetőek el ezek a konverziók a kétféle forgalom reprezentáció között? Nem sokat mondhatunk általánosságban, mert a TFA egy általános módszer, amelyet különböző forgalommodellekkel lehet használni, amelyek megfelelnek a TFA-ról szóló eredeti cikkben [6] a *forgalommodellel szemben támasztott követelmények*nek.

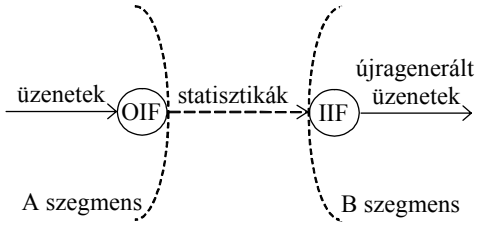
Először általánosságban vizsgáljuk meg, hogy milyen kérdéseket vet fel a TFA és a DES forgalom leíró módszerei közötti konverzió, aztán megtárgyalunk néhányat speciális esetekben.

Amikor rendszereket modellezünk, a rendszer azon tulajdonságaira koncentrálunk, amelyek fontosak a számunkra, és elhanyagolunk olyan részleteket, amelyek kevésbé fontosak a mi nézőpontunkból. Tekintsük a következő példát: egy hálózat teljesítőképességét szeretnénk megvizsgálni abból a célból, hogy megtaláljuk a lehetséges szűk keresztmetszeteket. A DES-hez való forgalommodell elkészítésénél valószínűleg alaposan megvizsgáljuk (az összes forgalom forrásra) a csomaghossz és érkezési időköz eloszlását és a kettő korrelációját, (és talán más jellemzőket is, mint a forgalom börsztössége), de nem igazán érdekel bennünket a csomagok tartalma. A TFA-hoz való forgalom modell készítéshez a forgalom forrásokat a *csomaggyakoriság eloszlás*sal (*packet throughput distribution*) jellemezzük a csomópontok kapcsológépei szempontjából, és a *bitmennyiség eloszlás*sal (*bit throughput distribution*) jellemezzük az átviteli vonalak szempontjából. Akár DES-t, akár TFA-t használunk, olyan modellt tervezünk, hogy a modellen végzett kísérleteink eredményei választ adjanak a kérdéseinkre. Azonban a munkák egy része arra irányul, hogy a modellünk megfeleljen az adott modellező rendszer követelményeinek. Például egy adott topológia leíró nyelvet kell használnunk, az egyes elemek működését pl. C++ nyelven kell lekódolnunk, stb. Ez igaz, abban az esetben is, ha a TFA-t és a DES-t szeretnénk kombinálni. ***Mindkét modellt úgy kell kialakítani, hogy kombinálhatók legyenek egymással.*** Ami a forgalomcserét illeti, ez azt jelenti, hogy mind a TFA mind a DES szegmens forgalommodelljének elegendő információt kell tartalmaznia a másik típusra való konverzióhoz, és a konverzió eredményének ki kell elégítenie a cél szegmensben a forgalommodellel szemben támasztott követelményeket. Ez az, amit általánosságban elmondhatunk.

Egy tipikus forgalommodell követelményei

A fenti példában a TFA lehetséges forgalommodelljeként említettük a csomaggyakoriság eloszlást és a bitmennyiség eloszlást. Ezeket javasolta és részletesen be is mutatta a TFA-ról szóló eredeti cikk is. [6] Mivel ezek tulajdonképpen *hisztogrammok*, ha ezeket választjuk, felhasználhatunk sok korábbi eredményt. Van egy kiváló könyv a nemparametrikus sűrűségfüggvény becslésről: [1]. Hisztogrammoknak statisztikagyűjtésre való használatát vizsgálta: [5]. Az a kutatás a statisztikai szinkronizációs módszer (*Statistical Synchronisation Method - SSM*) érdekében folyt, amely egy kevéssé közismert de ígéretes párhuzamos szimulációs szinkronizációs módszer. [9] Az SSM kutatás néhány eredményét fel tudjuk használni. Az SSM-et röviden összefoglaljuk, további információ [9]-ben található.

Más párhuzamos diszkrét idejű szimulációs módszerekhez hasonlóan a rendszer modelljét szegmensekre osztják. A szegmensek kommunikációja üzenetek küldésével és fogadásával reprezentálható. Az SSM céjából minden szegmenst kiegészítenek egy vagy több input és output interfésszel. A szegmensek között küldendő üzeneteket azonban nem továbbítják a cél szegmens felé, hanem a küldő szegmens *output interfésze* (OIF) statiszikát gyűjt róluk. A szegmensek *input interfészei* (IIF) a megfelelő output interfész által gyűjtött statisztikák alapján generálnak üzeneteket.



2. ábra Egy OIF - IIF pár

A szegmensek az input és output interfészeikkel együtt egymástól külön (külön processzoron) szimulálhatók, és a rendszer statisztikailag korrekt eredményeket ad. Természetesen az egyik szegmensben létrejövő eseményeknek nem pontosan olyan hatása van egy másik szegmensben, mint az eredeti modell esetén, így az SSM-t használó párhuzamos szimuláció eredményei nem pontosak. A pontosság függ a modell részekre bontásától, a statisztika gyűjtés és az üzenet újragenerálás pontosságától, valamint a processzorok közti információcsere gyakoriságától.

Az SSM viselkedését [3] és az elérhető gyorsulást [4] vizsgáló kísérletekben hisztogrammot használtunk a statisztikák reprezentációjára.

Az SSM-et használó párhuzamos DES valamint DES és TFA kombinációja közötti alapvető különbség ellenére mindkét esetben azonos vagy hasonló statisztika gyűjtési és üzenet generálási módszerek használhatók.

Ha a TFA forgalommodelljének implementációjára hisztogrammot használunk, akkor felhasználhatjuk [5] eredményeit. A cikk nemcsak a hisztogrammot, hanem számos más statisztika gyűjtési módszert is összehasonlít és alkalmaz különböző eloszlások esetén, hogy meghatározza, milyen nem-parametrikus sűrűségfüggvény becslési módszert érdemes használni statisztikagyűjtésre. A megvizsgált módszerek: Barron estimate, egyenlő osztásközű hisztogramm, egyenlő cellavalószínűségű hisztogramm, és diszkrét eloszlásokra a relatív gyakoriság módszere. Mind az erőforrás igényüket, mind a pontosságukat megvizsgáltuk. Az L_1 hibakritériumot alkalmaztuk. Megállapítottuk, hogy bár elméletileg az egyenlő cellavalószínűségű hisztogrammnak kellene a legkisebb L_1 hibát adnia, és ezt is tapasztaltuk, amikor túl kevés cellát használtunk, de elegendő számú cella esetén az egyenlő osztásközű hisztogramm volt a jobb. Folytonos eloszlásokra tehát az utóbbit javasoltuk. A relatív gyakoriság módszere elfogadható eredményeket adott valós életből vett diszkrét és kvantált eloszlásokra.

Valószínűleg a hisztogramm megfelel a céljainknak és a modellezett rendszer kellő ismeretében hisztogramgyűjtés tartománya és a cellaméret meghatározható, ha egydimenziós eloszlásaink vannak. De előfordulhat, hogy nincs elég a-priori információnk az eloszlásokról, vagy a megfigyelések száma túl alacsony a kellő pontosságú eredmények eléréséhez hisztogramm használatával. Ebben az esetben az üzenetekről statisztikákra való konverziót 2 lépésben végezhetjük: először egy másik módszert alkalmazunk statisztika gyűjtésre, majd annak eredményét transzformáljuk hisztogrammá. Egy esélyes jelölt a K-split [10] módszer. A K-split cella felosztások segítségével az eloszlásnak a megfigyelések számának megfelelően mindig optimális számú cellát használ. Ha több dimenziós eloszlásaink vannak, akkor ez a két lépéses megoldás (K-split használata, majd konverzió hisztogrammá) erősen javasolt.

A VIRTUÁLIS IDŐK ELTÉRŐ HASZNÁLATÁNAK KEZELÉSE

A kombináció és az együttműködés kérdései

A TFA és a DES *kombinációja* alatt azt értjük, hogy ugyan arra a hálózatra szeretnénk alkalmazni őket; egyiket a hálózat egyik részére a másikat pedig a másik részére. A kétféle forgalom reprezentáció közötti (előző fejezetben tárgyalt) kétirányú konverzió egy szükséges feltétel; nélküle a két rész nem tud egymással információt cserélni. Azonban ha azt szeretnénk, hogy a TFA és a DES *együtt is működjenek*, többre van szükségünk. Mielőtt a részletekre rátérnénk, vizsgáljuk meg, hogyan használják a virtuális időt – ez kulcsfontosságú lesz.

Az eseményvezérelt DES algoritmusa – amiben a virtuális idő kulcs szerepet játszik –, a következő (FES: Future Event Set – jövőbeli események halmaza):

```
inicializálás, bizonyos események berakása a FES-be;
repeat
  az első (azaz legkisebb időbélyegű) esemény kivétele (és törlése) a FES-ből;
  MOST := a FES-ből kivett esemény időbélyege;
  az esemény feldolgozása, közben esetleg új események berakása a FES-be;
until (FES kiürült) v (MOST > időkorlát) v (más okból meg kell állnunk)
```

A TFA pillanatfelvételt a vizsgált rendszer forgalmi viszonyairól, azaz, egy adott virtuális időpontban vizsgálja a rendszert. Bár a virtuális idő a DES-sel ellentétben nem halad a TFA során általában fontos és befolyásolja az eredményeket a rendszer paraméterein keresztül. Például ha POS vagy ATM terminálokat modellezünk egy hálózatban, a tranzakciók gyakorisága függ a hét napjától és a napon belüli órától is.

Definiáljuk a ***TFA és DES együttműködés***ét a következőképpen (egy vagy több alkalommal a következő történik):

1. DES beállítja a TFA virtuális idejét
2. DES bemeneti paramétereket ad TFA számára
3. DES meghívja TFA-t, hogy értékelje ki a rendszer rábízott részét
4. TFA fut
5. TFA paramétereket ad vissza DES-nek

A fenti lépéseknek a végrehajtásának módja erősen függ a program(ok) felépítésétől és a kommunikációs módszertől. Lássunk néhány lehetőséget:

A) A DES rész és a TFA rész két külön program és valamilyen inter-processz kommunikációs (IPC) módszert használnak (pl. PVM/MPI, named pipe-ok).

B) A TFA egy függvény a DES programon belül, amit néha meghívnak, de nem használja a DES kernel virtuális idejét saját belső céljaira.

C) A TFA egy függvény(halmaz) a DES programon belül, amit néha meghívnak, és használja a DES kernel virtuális idejét (esemény mechanizmusát és egyéb szolgáltatásait) saját belső céljaira.

Mindegyik hasznos lehet, de lehetnek hátrányaik is. Az "A" megoldás nem biztos, hogy kivitelezhető egy kereskedelmi DES programmal (például OPNET Modeler),

amelyet nem terjesztenek a teljes forráskóddal, és így nehézséget okozhat, vagy néha gyakorlatilag nem kivitelezhető az IPC használata.

A "B" lehet a választott megoldás, ha olyan szimulációs programcsomagot használunk, amely nem eléggé flexibilis a "C" használatához, vagy valamiért ezt szeretnénk használni. A "B" választásával elkerüljük azokat a problémákat, amikkel a cikk hátralevő részében foglalkozunk, de nélkülözzük a szimulátor által nyújtott szolgáltatások felhasználásával nyújtott előnyöket is. Ekkor a TFA részben mindent magunknak kell implementáltunk, beleértve valószínűleg a hálózati topológia leírását és az útvonalválasztást is. Nem kapunk segítséget a szimulátortól, nem vagy csak kis mértékben tudjuk felhasználni a kész elemeket. Ha egy olyan függvényt írunk, amit egyszer meghívnak, aztán végrehajtja a TFA-t, végül visszaadja az eredményeket, nem tudjuk használni a szimulátor eseménykezelő mechanizmusát, azaz: elveszítjük az összes eseményvezérelt modell felhasználásának a lehetőségét.

Akkor érdemes a "C" megoldást választanunk, ha a TFA részben így a szimulátor sok funkcióját fel tudjuk használni, és ez lényegesen csökkenti a programozásai (és modell építési) munkánkat. Ettől kezdve "C" esetén felmerülő kérdésekkel foglalkozunk.

Mit jelentenek az együttműködés lépései, ha a TFA-n belül felhasználjuk a rendszer virtuális idejét? (Azaz, használjuk a szimulátor eseménykezelését.) Nézzük meg egyenként!

1. A DES beállítja a TFA virtuális idejét a TFA indító eseményének a megfelelő virtuális időpontra való felidőzítésével. (Az esemény bekerül a FES-be.)
2. A DES a bemeneti paramétereket a TFA-nak a TFA-ért felelős modul(ok)nak való üzenetküldéssel adja át.
3. A DES egyáltalán nem hívja meg a TFA-t, akkor fut, ha elérkezik a virtuális ideje.
4. Amikor a TFA fut, használhatja az eseménykezelő rendszert, de (a virtuális idő ellentmondásos kezelése miatt) ezzel még foglalkoznunk kell.
5. A TFA a DES-ért felelős modul(ok)nak való üzenetküldéssel ad vissza paramétereket a DES-nek.

A 4. pont kivételével mindegyik világos. A virtuális idő kezelésével a következő alfejezetben foglalkozunk.

Hogyan tudjuk összeegyeztetni a DES és a TFA virtuális idő kezelését?

Idézzük fel, hogy hogyan osztja el a TFA a forgalmat a hálózatban. A következő eljárást hajtja végre minden forgalomforrásra: először is felosztja az adott forrás forgalmát leíró statisztikákat kisebb egységekre, aztán elküldi őket egyenként a céljuk felé. A méretüket az *útvonalválasztási egység méreté*nek (*size of routing unit* – S_{RU}) hívják, és ezeket egyenként viszik végig a hálózaton ugyan olyan útvonalválasztási algoritmus szerint, amit a csomópontok a DES esetén használnak az adatcsomagok továbbításakor. Természetesen S_{RU}-t az egyrészt úgy kell megválasztani, hogy a felbontás kellően finom legyen (hogy elég pontos eredményeket kapjunk); másrészt legalább egy nagyságrenddel kisebbnek kell lenni a statisztikákat hordozó útvonalválasztási egységek számának, mint az adatcsomagokat hordozó üzenetek száma lenne a DES esetén (hogy a TFA gyorsabb legyen, mint a DES).

A TFA-ban a DES infrastruktúrát felhasználva a statisztikákat a forgalomforrás üzenetbe csomagolva küldi el a cél felé, és a csomópontok és vonalak is üzenetként továbbítják őket. A csomópontok ugyan azt az útvonalválasztási algoritmust használhatják a statisztikákat hordozó üzenetek továbbítására, amit a DES használ az adatcsomagokat jelentő üzenetek továbbítására. Nagyon fontos, hogy nem kell mindent a semmiből létrehoznunk. Egy objektum orientált környezetben felhasználhatjuk a már adott elemeket, még ha esetleg egy kicsit módosítanunk kell is a viselkedésüket.

Nagyon fontos kérdés, hogyan lehet a különböző forrásból származó és ugyan azon a hálózaton áthaladó forgalom-folyamokat leíró statisztikákat jól összeszőni. Ez kritikus a forgalomnak a hálózatban való térbeli szétosztása szempontjából. Van két megoldás, amelyek mindegyike összeegyezteti a TFA és a DES virtuális idő használatát, de különböző módon. Egy kis iróniával az elsőt *modell hackelés*nek a másodikat *kernel hackelés*nek nevezhetjük.

Modell hackelés alatt azt értjük, hogy úgy kell megalkotnunk a TFA-val együttműködő DES modellt, hogy az hagyjon egy kicsi, de pozitív T virtuális időtartamot a TFA működésére. T-t elegendően kicsire kell választanunk, hogy a DES modell tolerálja azt, hogy bár a TFA-t t virtuális időpontban indítja el, csak $t+T$ virtuális időpontban kapja vissza az eredményeket. A különböző forrásból származó forgalom összeszövéséhez tekintsünk két forrást, **A**-t és **B**-t, amelyek rendre N és M számú statisztikát küldenek. **A** és **B** az i-edik statisztikát a következő virtuális időpontban fogja küldeni:

$$t_{A,i} = t + i\frac{T}{N}, \quad i = 0,\ 1,\ \ldots\ N-1 \qquad \text{és} \qquad t_{B,i} = t + i\frac{T}{M}, \quad i = 0,\ 1,\ \ldots\ M-1$$

A statisztikák szétosztásakor a TFA nem modellezi a feldolgozási vagy átviteli késleltetést, így a statisztikák az eredeti időbélyeggel utaznak végig a hálózaton. A TFA egy-egy befejező eseményt ütemez minden egyes csomópont és vonal számára $t+T$ időbélyeggel. Ennek az eseménynek a hatására a csomópontok és vonalak transzformálják a korábban összegzett (rajtuk áthaladó) statisztikákat úgy, hogy figyelembe veszik a saját véges kapacitásukat. Az algoritmus megtalálható az eredeti cikkben [6], konvergenciájának bizonyítása pedig [8]-ban.

A modell hackelés előnye, hogy (a kernel hackeléssel ellentétben) ez a megoldás nem kívánja a szimulációs kernel módosítását. Remélhetőleg a DES modellünk képes a TFA szegmens egy rövid, T idejű késleltetését tolerálni. Különben módosítanunk kell a DES modellünket, hogy képes legyen tolerálni – innen „modell hackelés” elnevezés.

A *kernel hackelés* egy tisztább módot nyújt a különböző források forgalmának összeszövésére. Azonban ehhez szükség van a DES kernel módosítására. Ez megtehető, ha a kernel a sajátunk, mint ahogyan DES kernelt tartalmazó Iminet hálózati szakértői rendszer az Elassys Consulting Kft. tulajdona, vagy a szimulátor nyílt forrású, mint például az OMNeT++ diszkrét idejű szimulációs rendszer [11]. A szimulációs kernel módosítása az *al-idő* bevezetését jelenti, azaz, egy második időbélyeget adunk minden eseményhez. Az eseményeket elsődlegesen az eredeti időbélyeg alapján rendezzük, de ha ez két esemény esetén megegyezik, akkor a sorrendjüket a második időbélyeg dönti el. Természetesen a második időbélyeg értéke a normál DES események esetén 0. A teljes időbélyeget úgy jelöljük, hogy az eredeti és a kiegészítő időbélyeget vesszővel elválasztva írjuk egymás mellé. Ekkor az előbbi példában szereplő TFA statisztika küldési eseményeket a következő virtuális időpontokra kell időzítenünk:

$t_{A,i} = t, i\frac{1}{N}, \quad i = 0, \quad 1, \quad \ldots \quad N-1$ és $t_{B,i} = t, i\frac{1}{M}, \quad i = 0, \quad 1, \quad \ldots \quad M-1$

Természetesen bármilyen más fix konstans is állhat az "1" helyén a számlálóban.

ÖSSZEFOGLALÁS

Megvizsgáltuk, hogy hogyan lehet a TFA-t és a DES-t kombinálni kommunikációs hálózatok még hatékonyabb teljesítőképesség vizsgálata céljából.

Megmutattuk, hogyan képes a TFA és a DES információt cserélni az eltérő forgalom reprezentációik (statisztikák és üzenetek) közötti kétirányú konverzióval. Ehhez felhasználhatók a statisztikagyűjtéssel kapcsolatos korábbi kutatási eredményeink.

Adtunk két megoldást a TFA és a DES különböző (és egymásnak ellentmondó) virtuális idő kezelésének összeegyeztetésére.

Megállapítjuk, hogy a TFA és a DES integrációja ígéretes, és a TFA csomagnak egy flexibilis DES kernel fölötti megvalósítása hatékony implementációt eredményezhet.

KÖSZÖNETNYILVÁNÍTÁS

A kutatást az MTA Bolyai János ösztöndíja támogatta. Ez a cikk [7] fordítása.

IRODALOM

[1] Devroye, L.; L. Györfi: *Nonparametric Density Estimation: The L1 view.* John Wiley, New York 1985.

[2] Fujimoto, R. M.: Parallel Discrete Event Simulation. *Communications of the ACM* Vol. **33,** no. 10, pp. 31-53, 1990.

[3] Lencse, G.: "Efficient Simulation of Large Systems - Transient Behaviour and Accuracy" *Proceedings of the 1997 European Simulation Symposium (ESS'97)* (Passau, Germany, Oct. 19-23. 1997.). SCS Europe, pp. 660-665.

[4] Lencse, G.: "Efficient Parallel Simulation with the Statistical Synchronization Method" *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation (CNDS'98)* (San Diego, CA. Jan. 11-14, 1998.). SCS International, pp. 3-8.

[5] Lencse, G.: "Statistics Collection for the Statistical Synchronisation Method" *Proceedings of the 1998 European Simulation Symposium (ESS'98)* (Nottingham, UK. Oct. 26-28, 1998.). SCS Europe, pp. 46-51.

[6] Lencse, G.: "Traffic-Flow Analysis for Fast Performance Estimation of Communication Systems" *Journal of Computing and Information Technology* **9**, No. 1, pp. 15-27, 2001.

[7] Lencse, G.: "Combination and Interworking of Traffic-Flow Analysis and Even-Driven Discrete Event Simulation" *Proceedings of the European Simulation and Modelling Conference (ESMc'2004)* (Paris, France, Oct. 25-27, 2004.). Eurosis, pp. 89-93.

[8] Lencse, G; L. Muka: "Convergence of the Key Algorithm of Traffic-Flow Analysis" *Journal of Computing and Information Technology,* - accepted for publication, expected to appear in Dec. 2005.

[9] Pongor, Gy.: "Statistical Synchronization: a Different Approach of Parallel Discrete Event Simulation". *Proceedings of the 1992 European Simulation Symposium (ESS'92)* (The Blockhaus, Dresden, Germany, Nov. 5-8, 1992.) SCS Europe, pp. 125-129.

[10] Varga, A.: "K-split – On-Line Density Estimation for Simulation Result Collection". *Proceedings of the 1998 European Simulation Symposium (ESS'98)* (Nottingham, UK. Oct. 26-28, 1998.). SCS Europe, pp. 41-45.

[11] Varga, A.: "OMNeT++ Discrete Event Simulation System" http://www.omnetpp.org