BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
TÁVKÖZLÉSI ÉS MÉDIAINFORMATIKAI TANSZÉK

MÓDSZEREK HÁLÓZATI FORGALOM HATÉKONY OSZTÁLYOZÁSÁHOZ

Szabó Géza

Tézisfüzet

Tudományos vezető

Dr. Molnár Sándor PhD
Nagysebességű hálózatok laboratóriuma (HSNLab)
Távközlési és Médiainformatikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Budapest
2010

1. Bevezetés

A hálózati forgalom részletes ismerete fontos a hálózati üzemeltetők és adminisztrátorok számára, mivel ez egy kulcsfontosságú információ számos hálózat menedzsmenttel kapcsolatos tevékenységhez. Például információt szolgáltathat hálózat tervezéshez, kapacitás becsléshez, a számlázási konstrukciók finomhangolásához vagy biztonsági monitorozáshoz. Továbbá, a hálózati forgalom aggregátum alapos megértése és a számos alkalmazás által generált hálózati forgalmi trendek megfigyelése is fontos a hálózati eszközök tervezéséhez.

A forgalomosztályozás célja, hogy meghatározzuk milyen alkalmazást futtatott a végfelhasználó, mi a különböző alkalmazások által generált forgalom aránya a teljes hálózati forgalomban.

Az IP hálózati csomópontok közötti kommunikációt adatfolyamokba lehet szervezni, és a forgalomosztályozás feladata az, hogy minden egyes adatfolyamhoz egy meghatározott alkalmazást rendeljen. Az adatfolyam olyan IP csomagok gyűjteményéből áll, amik egy adott portról, egy adott IP-címről, egy adott portra egy adott másik IP-címre, egy adott protokollt használnak. Az adatfolyamot az alábbi ötös azonosítja: forrás IP cím, a cél IP cím, forrás port, cél port, protokoll azonosító.

A legpontosabb forgalomosztályozás nyilvánvalóan a teljes protokoll feldolgozása lenne. Azonban számos protokoll biztonsági okokból titkosítást használ (SSH [8], SSL [7]). Emellett néhány protokoll saját célú felhasználásra tervezett, így nem áll rendelkezésre nyilvános leírásuk (Skype [9], MSN Messenger [5], World of Warcraft [10], stb.). Általában véve, nehéz lenne minden előforduló protokollt tökéletesen feldolgozni. Ezen kívül, még egy egyszerű protokoll állapotkövetés is olyan erőforrásigényes nagy számú felhasználó esetén, hogy gyakorlatilag megvalósíthatatlan.

- **Port alapú forgalomosztályozás:** Ebben a legegyszerűbb és leggyakoribb módszerben a besorolás azon alapul, hogy egy jól ismert portszámot, egy adott forgalmi típushoz rendel, pl., a web forgalmat a 80-as TCP porthoz. [4].

- **Minta alapú forgalomosztályozás - Mély csomagvizsgálat (Deep Packet Inspection – DPI) [36]:** Ahhoz, hogy megvalósíthatóvá tegyék a protokoll-felismerést, csak speciális bájt-mintákat keresnek a csomagokban állapotmentes módon. Ezek a bájt-minták előre definiáltak, hogy lehetségessé tegyék bizonyos forgalmi típusok azonosítását, pl. a web forgalom tartalmazza a '`GET`' mintát, az eDonkey – egyenrangú felek közötti fájlcserélés – peer-to-peer (P2P) forgalom tartalmazza a '`e3 38`' mintát.

- **Forgalomkarakterisztika alapú forgalomosztályozás [32], [44], [31], [20], [15]:** A forgalomkarakterisztika alapú forgalomosztályozás során a forgalom néhány statisztikai jellemzőjét vizsgálják, és segítségükkel beosztályozzák a hálózati forgalmat. Ahhoz, hogy automatikusan felfedezzék a forgalom jellemzőit, a statisztikai módszereket általában mesterséges intelligencia módszerekkel kombinálják. A leggyakrabban használt módszer a Bayes-osztályozó, ahogy a következő publikációkban is szerepel: [32], [44], [31], [20], [15].

- **Kapcsolati minta alapú forgalomosztályozás [25],[26]:** Az alapvető ötlet az, hogy meg kell nézni a kommunikációs mintát, amit egy adott gép létrehozott, és össze kell hasonlítani más viselkedésmintákkal, amik különböző tevékenységeket/alkalmazásokat ábrázolnak.

- **Információ elmélet alapú forgalomosztályozás:** Hasznos segítséget nyújtanak a forgalomosztályozásban a [42] és [28] publikációkban bemutatott módszerek, melyek információ elméleti megközelítést használnak, és tipikus viselkedés típusokba tudják csoportosítani a felhasználókat pl., szerverek, támadók. Az alapötlet az, hogy meg kell nézni az adatfolyamot azonosító ötös változékonyságát, illetve az értékkészletük véletlenszerűségét, melyek tartoznak egy adott forráshoz vagy a cél IP-címhez, forrás vagy célporthoz. Gyakorlatilag ez a módszer a kapcsolati minta alapú módszerben a gráf csomópontok számosságának egy tömörebb ábrázolása.

2. Kutatási célkitűzések

A disszertáció célja, az IP-hálózatok számára robusztus és hatékony forgalomosztályozási módszerek tervezése és fejlesztése.

Az **első részben a célom**, hogy meghatározzak egy keretrendszert, ami lehetségessé teszi az irodalomban rendelkezésre álló forgalomosztályozási módszerek összehasonlítását. Javasoltam egy módszert a forgalomosztályozási módszerek validációjára. További célom volt, hasznosítani a forgalomosztályozási módszerek pontossági metrika méréseinek az eredményeit, hogy javasolhassak egy új kombinált forgalomosztályozási módszert.

A forgalomosztályozási módszerek validációja során kiderült, hogy azok a feltételezések, amiket az irodalomban a forgalom karakterisztika alapú forgalomosztályozási módszereknél használtak, többnyire elavultnak bizonyultak a jelenleg népszerű játékok forgalmának azonosítása során. Az aktuális internetes játék forgalom jelentős

része valós idejű stratégiai játékokból (Real Time Strategy – RTS) és nagyon sok szereplős szerepjátékok (Massively Multiplayer Online Role Playing Game – MMORPG) forgalmát is tartalmazza a korábban modellezett sajátnézetes lövöldözős (First Person Shooter – FPS) játékok mellett. Az FPS forgalom állandó méretű csomagokból áll, amiket állandó sebességen küldenek a kommunikáló felek. A jelenleg népszerű játékforgalom különleges abban az értelemben, hogy a forgalmi jellemzőket a felhasználói viselkedések erősen befolyásolják. A **második** részben a **cél** ezeknek az új játéktípusok forgalmi jellemzőinek elemzése volt.

A játékok általában saját nem dokumentált protokollokat használnak – esetleg Protokoll Fejléc Titkosítást is –, hogy a mélycsomagvizsgálati módszerekkel végzett felismerést ezáltal megvalósíthatatlanná tegyék. Munkám során a játékforgalommal kapcsolatban több esetben is azt tapasztaltam, hogy a játék a kommunikációja során a játékosok pozícióit kommunikálja a felek között. Elemeztem az információ hasznosíthatóságának lehetőségét. A **harmadik** részben a **cél** az aktuális mély- csomagvizsgálati módszerek kiterjesztése azzal a képességgel, hogy nem-állandó bájt-mintákat tartalmazó mezőket is felismerjen egy protokoll-szerkezetben.

A munkám az első részében, forgalom osztályozási módszerek pontosságára és teljességére koncentráltam a forgalom- osztályozási módszerek sebességét figyelmen kívül hagyva. Mindazonáltal, a legtöbb esetben ez fontos kérdés a módszer alkalmazhatóságát tekintve, egy valós idejű forgalom osztályozási rendszerben. A **negyedik** részben a **cél** a valós idejű forgalom osztályozás megvalósíthatóságának vizsgálata és a forgalomosztályozó módszerek áttervezése párhuzamos architektúrákon való hatékony alkalmazásához.

3. Kutatási módszertan

A első és negyedik tézisekben a javasolt módszereket *implementáltam* és valós működő hálózatokban illetve onnan származó *mérések*en teszteltem.

A második tézisben bemutatott eredményeket aktív mérések *statisztikai elemzése* során szereztem. A harmadik tézisben a modellparamétereket valós hálózatokból származó mérések statisztikai elemzésén keresztül nyertem ki.

A harmadik tézisben az idősorok dekompozíciójának hatását és a negyedik tézisben az elérhető tömörítési arányt elemeztem az általam készített *szimulátor*ban, amiket Matlab-ban [11] valósítottam meg.

4. Új eredmények

4.1. Új keretrendszer forgalomosztályozásra és validációra

Először a forgalomosztályozási algoritmusok validációjának a problémájával foglalkoztam. Több különböző osztályozási módszert találhatunk az irodalomban. Mindazonáltal ezeknek a módszereknek a validációja gyenge és ad hoc, mert egyik sem egy megbízható és széles körben elfogadott validációs technika, illetve nincsenek elérhető egyértelműen osztályozott referencia mérések. A validáció általában egy adott másik forgalomosztályozási módszerrel történik a publikációkban.

Sen és társai [36] az általuk létrehozott bájt-minta adatbázisukat azáltal validálták, hogy kézzel ellenőrizték a módszerük hamis pozitív arányát. Karagiannis és társai [26] a módszerüket azáltal validálták, hogy mély csomagvizsgálati forgalomosztályozási módszert használtak. Mivel nincsenek közösen elfogadott és jól működő bájt-minták, a szerzők saját bájt-minta adatbázist konstruáltak. Crotti és társai [18] port alapú forgalomosztályozást használtak, amíg [21] szerzői mélycsomagvizsgálati módszert használtak ahhoz, hogy a módszerüknek tanító és teszt adatsort készítsenek az egyetemen gyűjtött adatokból. Zander és társai [44] a módszerük validációjára port alapú forgalomosztályozást használtak.

Erman és társai [20] publikus mérésekkel dolgoztak, de ezek a mérések csak csomagfejléceket tartalmaztak, ami így kizárja az olyan megbízható validációs módszereket, amik a csomagtartalmat vizsgálják. A [15] publikációban, a forgalomosztályozási módszert online alkalmazták a szerzők anélkül, hogy elmentették volna az eredeti adatot, mivel a nagy sebességű hálózat miatt a keletkező nagy mennyiségű adatot nem tudták volna tárolni. Ezáltal lehetetlenné vált, hogy mások által is validálható legyen a forgalomosztályozási módszerük eredménye.

1. Téziscsoport. *[J1, C1, C2, C3] Javasolok egy új eljárást forgalomosztályozási módszerek validációjára. Javasolok egy új kombinált forgalomosztályozási módszert. [Disszertáció 2. fejezet]*

4.1.1. Forgalomosztályozási módszerek validációja

Egy forgalomosztályozási módszert meggyőzően csak aktív teszt által lehet validálni, ami kielégíti a következő követelményeket:

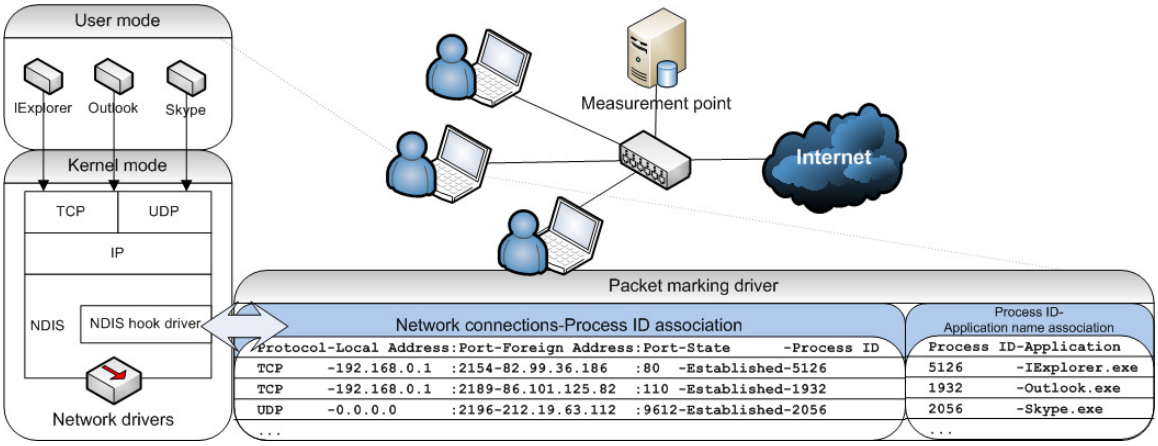
- A forgalomosztályozási módszerektől függetlennek kell lennie, azaz el kell azt kerülni, hogy egy forgalomosztályozási módszer validációját egy másik forgalomosztályozási módszerrel végezzük.
- Mindegyik hálózati csomagról a tesztnek referenciainformációt kell szolgáltatnia, amit össze lehet ezáltal vetni a vizsgált forgalomosztályozási módszerek eredményével.
- A tesztnek determinisztikusnak kell lennie, azaz nem épülhet véletlenszerű döntéseken.
- Megvalósíthatóság: lehetővé kell tenni nagy tesztek végrehajtását is erősen automatizálva.
- Az aktív tesztek környezetének valósághűnek kell lennie.

1.1. Tézis. *[C3] Javasolok egy eljárást forgalomosztályozási módszerek validációjára, ami a forgalom generáló felhasználói végberendezések funkcionalitásának kibővítésén alapul. A módszer egy alkalmazási rétegbeli információval egészíti ki a hálózati rétegbeli csomagfejléceket.*

A két fő követelmény a módszer megvalósításával kapcsolatban az, hogy nem szabad lerontania a terminál teljesítményét, és a hozzáadott információtöbblet méretének szintén elhanyagolhatónak kellene lennie. A preferált megvalósítás olyan meghajtóprogram, amit könnyen lehet telepíteni a terminálokra. A bemutatott meghajtóprogram architekturális pozíciója látható az 1. ábrán. Ez a meghajtóprogram a hálózati illesztő előtt helyezkedik el, így mindegyik csomagnak, amit a terminál és a hálózat között áramlik, át kell utaznia ezen a meghajtóprogramon. Megvalósítottam egy prototípust, ami egy Windows XP meghajtóprogram a Network Driver Interface Specification (NDIS) könyvtár alapján.

A kernel NDIS könyvtára a hálózati hardver képességeit teszi elérhetővé, illetve egy API-t nyújt, amin keresztül intelligens hálózati meghajtóprogramokat lehet hatékonyan programozni. Ha a küldő és fogadó funkcióit az NDIS IP protokoll meghajtóprogramnak keresztülvezetjük a javasolt meghajtóprogramon, akkor az összes TCP és UDP csomagot látjuk, megvizsgálhatjuk és műveleteket végezhetünk velük.

Ahhoz, hogy a követelményeknek megfeleljen a meghajtóprogram, a következő módon működik. Egy áthaladó csomag esetén a következő folyamat játszódik le (lásd. 2. ábrán):



1. ábra. A javasolt meghajtóprogram helye a terminálon belül

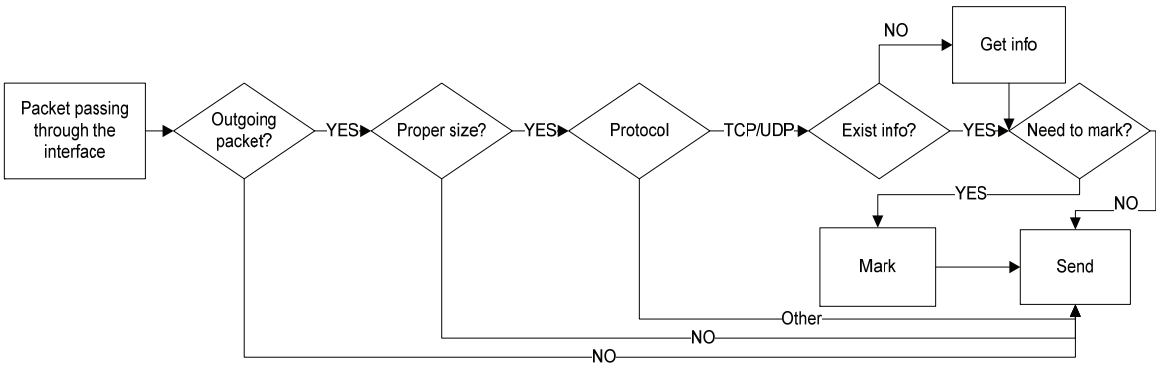
1. A csomagot megvizsgáljuk, hogy ez egy bejövő vagy kimenő csomag. Abban az esetben, ha ez egy bejövő csomag, a folyamat megáll anélkül, hogy jelölés történt volna, hiszen bejövő csomagokat felesleges megjelölni.

2. Abban az esetben, ha egy kimenő csomagról van szó, a csomag méretét vizsgálja meg. Ha jelen csomag mérete az átvihető maximális egységnek (Maximum Transmission Unit – MTU) felel meg, úgy csomagdarabolódáshoz vezetne a jelölés. Emiatt csak azokat a csomagokat jelöli meg, amik mérete MTU-nál kisebb a csomagjelölés nagyságát is ideértve.

3. Mivel az operációs rendszerben nem áll rendelkezésre a nem TCP és UDP csomagokból felépülő hálózati kapcsolatokról információ, így a folyamat csak TCP és UDP csomagok esetén folytatódik.

4. Az adatfolyam 5-ös azonosítója alapján a csomagot ellenőrzi, hogy rendelkezésre áll-e az adat korábbról eltárolva. Ha a gyorsítótárban még nem áll rendelkezésre információ, akkor az operációs rendszertől lekérdezi a nyitott hálózati kapcsolatokat, és a a futó alkalmazások folyamatazonosítóit.

5. Amikor minden információ rendelkezésre áll a csomagjelöléshez, van egy utolsó lehetőség még eldönteni, hogy a meghajtóprogram megjelölje-e a csomagot. A csomagjelölés az adatfolyam összes csomagjára is megtörténhet, véletlenszerűen választott csomagokra, csak az első csomagjára az adatfolyamnak, illetve ki is kapcsolható alkalmazásonként a jelölés.



2. ábra. A bevezetett meghajtóprogram működési elve

Kiértékelés. A validációt úgy végeztem el, hogy valós hálózati körülmények között használtam a módszert. A módszer transzparensen működött a kommunikációs felek között, mindazonáltal ennek lehetőségét további funkcionalitások segítségével valósítottam meg, ami kiterjesztése az alapötletnek (részletekért lásd. [C3]).

A rendszerrel szemben támasztott követelmények teljesülnek:

- A forgalomosztályozási módszerektől függetlennek kell lennie: ez teljesül, mivel a forgalomosztályozás a forgalomgenerálás ideje alatt megtörténik ahelyett, hogy egy ismeretlen forgalmat utólag dolgoznánk fel.
- Mindegyik hálózati csomagról a tesztnek referenciainformációt kell nyújtania: a forgalom jelölés csomagszinten történik, opcionálisan akár minden egyes csomagot is meg lehet jelölni.
- A tesztnek determinisztikusnak kell lennie: a rendszer csak determinisztikusan működő elemekből építkezik.
- Megvalósíthatóság: az egyedüli manuális feladat a rendszer telepítése, így elmondható, hogy a rendszer nagy mértékben automatizált.
- Az aktív tesztek környezetének valósághűnek kell lennie: a rendszer nem vezet be semmilyen korlátozást a felhasználó napi tevékenységébe, így a rendszert telepítve valós felhasználókhoz a mért forgalom is valóságos lesz.

A két kulcsmetrika, ami karakterizálja a forgalomosztályozó mechanizmusok teljesítményét: a pontosság és teljesség.[26]:

- *Pontosság*ot a forgalomnak az a része határozza meg, amit helyesen azonosítottak abban az értelemben, hogy azt az alkalmazástípust társítják ahhoz az

alkalmazási típushoz, ami valóban létrehozta azt. Egy vizsgált módszernek a forgalomosztályozási pontosság értéke azt mutatja, hogy mekkora a helyesen osztályozott forgalom aránya a forgalom egészéhez képest, az adott alkalmazásra és adott vizsgált módszert tekintve. Ezt adja meg a következő képlet: $\mathcal{A} = \frac{TP}{TP+FP}$, ahol TP és FP a igaz pozitív (true positive) és hamis pozitív (false positive) döntéseket jelöli egy adatfolyam esetében, egy adott forgalomosztályozási módszert tekintve.

- *Teljesség*et a forgalomnak az a része határozza meg, amit egy adott módszer kategorizál. Ez a forgalom azon része, amit egy adott módszer képes egy adott alkalmazástípushoz társítani. A forgalomosztályozási arányértékek egy adott módszer által, egy adott forgalomtípusnak beosztályozott forgalom és a validációs mérésben az összes adott forgalom arányát mutatja. Ezt a következő módszerrel számíthatjuk ki: $\mathcal{C} = \frac{TP+FP}{TP+FN}$ ugyanazokkal a jelölésekkel tekintve, mint $\mathcal{A}$.

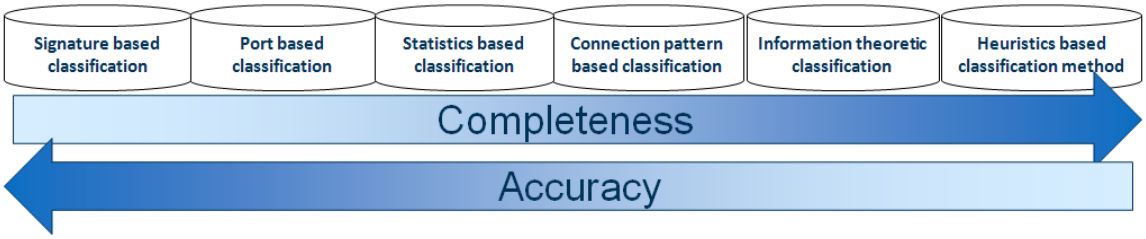
A tökéletes forgalomosztályozás az $(\mathcal{A}, \mathcal{C}) = (1, 1)$ esetben valósul meg. A forgalomosztályozás során ezeket a metrikákat kell magas szinten tartani. A következő példa két alkalmazás pontosságát és teljességét mutatja, amit egy adott módszerrel forgalomosztályoztak, (pl. `A`) összevetve a referencia méréssel (pl. `A*`).

```
A -> A, A -> A, A -> B, B -> B, ? -> B
Accuracy: A: 2/3 B: 1/1, Completeness: A: 3/2 B: 1/3
```

A forgalomosztályozási módszerek tapasztalt pontosságát és teljességét mutatja a 3. ábra. A legpontosabb módszer a szigorú protokoll-szintaktikai elemző- és ezek könnyűsúlyú implementációja, a mélycsomagvizsgálat alapú forgalomosztályozó. A legteljesebb módszerek, általában valamilyen gyenge heurisztikákon alapulnak javarészt. Így az ilyen módszerek mindig képesek osztályozni a forgalmat, azaz a teljességük magas lesz, mindazonáltal a pontosságuk alacsony. Fontos megjegyezni, hogy az 3. ábrán az aktuális módszerek pontosságának és teljességének egy általános áttekintése látható. Ugyanakkor nem biztos, hogy a metrikák közötti kompromisszumos választás mindig szükségszerű. Az alkalmazás típusokhoz tartozó pontos pontossági és teljességi értékeket lásd [C1]-ben részletesen.

4.1.2. Kombinált forgalomosztályozási módszer

Munkám előtt a publikációk általában egy bizonyos mindent megoldó forgalomosztályozási módszer után kutattak. Bebizonyítottam, hogy a forgalom osztályozási



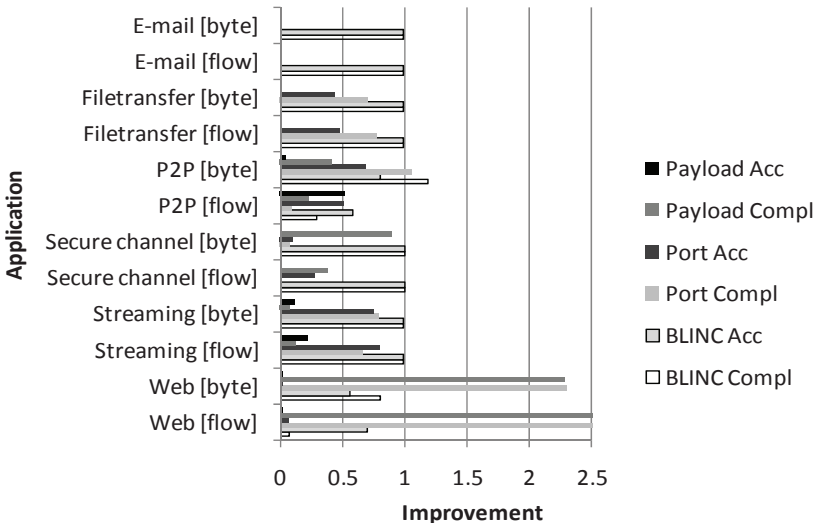
3. ábra. A pontosság és teljesség rangsora a különböző forgalomosztályozási módszereknek

módszerek pontossága változó az alkalmazás típus függvényében. Egy összetett döntési módszerre van szükség, hogy valós helyzetben magas pontosságot és teljességet nyújtson.

Az irodalomban létező forgalom osztályozási módszerek kombinálásának céljából a döntési fákat választottam ki, hogy kihasználhassam később a következő előnyeiket: a) a döntési fákat hatékonyan lehet megvalósítani és felhasználni b) a döntési fák egyszerűen megérthetőek és értelmezhetőek. Azt a gépi tanulási módszert, ami adatokból egy döntési fát képez, döntési fa tanításnak nevezik. A döntési fa tanító algoritmusok esetén diszkrét adattípusokra a célfüggvény általában az információ nyereséggel kapcsolatos, ami nem közvetlenül kapcsolódik a forgalomosztályozási problémához. A legfőbb probléma a kevés adatfolyammal rendelkező alkalmazástípusok elhanyagolása a tanulás folyamán.

1.2. Tézis. *[J1, C1, C2] Javasolok egy eljárást, ami a forgalomosztályozó módszerek eredményei közül a megfelelőt választja ki egy-egy forgalomtípushoz. A javasolt eljárás egy döntési mechanizmust és heurisztikákat foglal magában. Az eljárás pontossága az alkalmazott forgalomosztályozási módszerek mért pontosságán alapszik. A javasolt koncepciót egy automatikus döntési fa létrehozó algoritmusba beépítve alkalmazom. A döntési fa optimalizációs célfüggvénye forgalomosztályozási feladatra van specializálva.*

A javaslatom az, hogy közvetlenül forgalomosztályozási metrikákra optimalizáljon a döntési fa tanító algoritmus. A javaslatomban minden alkalmazástípust és forgalomosztályozási módszert azonos súllyal tekintem. Ahhoz, hogy pontosságra és teljességre is optimalizáljon az algoritmus, a javasolt célfüggvény a tökéletes osztályozástól $(\mathcal{A}, \mathcal{C}) = (1, 1)$ való euklideszi-távolsága az átlagos pontosságnak és teljességnek egy adott forgalomosztályozási módszert $m \in M$ tekintve egy adatfolyam csoportból F amik az A alkalmazás csoportot tartalmazzák.



4. ábra. A pontosság és teljesség növekedése a komibinált a bevezett módszernek a független módszerek eredményéhez képest

$$f(F,M,A) = \min_{m_i \in M}\{((1,1) - \frac{1}{|A|}\sum_{a\in A}(\mathcal{A}^a_{m_1}(F), \mathcal{C}^a_{m_1}(F))), ((1,1) - \frac{1}{|A|}\sum_{a\in A}(\mathcal{A}^a_{m_2}(F), \mathcal{C}^a_{m_2}(F))), ...\}$$

, ugyanazzal a jelöléssel ahogy a 4.1.1 szekcióban.

Kiértékelés. A módszer hálózati forgalmak alkalmazás komponenseinek vizsgálatára alkalmas. Az irodalomban felhasznált egyes módszerek teljesítménye főleg a nem valós idejű forgalomosztályzást támogatja.

A kombinált forgalomosztályozási módszert a 1.1. tézisben leírt módszerrel validáltam, azzal a kiegészítéssel, hogy a VoIP alkalmazások felismeréséhez a rendszert kiegészítettem [33]-ból vett ötletekkel. Az 4. ábrán látható, hogy az osztályozási pontosság és a teljesség is növekedett minden alkalmazástípus esetében. A részletek a [C3]-ban találhatóak.

4.2. A játékforgalom karakterisztikái

A játékforgalom vizsgálat fontos a forgalomosztályozás szempontjából, egyrészt amiatt, mert folyamatosan emelkedik a penetrációja, illetve a szimulációkkal támogatott játékszerver architektúra- tervezési módszerek miatt is.

A játékforgalom két fő tényezőtől, a játékprotokolltól és a játékosok viselkedésétől függ. Népszerű valós idejű sokjátékos játékok alapján ebben a téziscsoportban az utóbbi tényezőt vizsgáltam, ami azt mutatja, hogy egy tipikus játék különböző fázisai pl. játékos mozgás, a környezetben bekövetkező változások a különböző megfigyelési szinteken hogyan hatnak a forgalomra. Az emberi viselkedés természete olyan nagy hatással van ezen forgalmi jellemzőkre, hogy a forgalom mind makroszkopikus (forgalom ráta) mind mikroszkopikus (pl. csomagtartalom) szintjén jelentkezik a hatás. A játékos viselkedésének a játék szervertől jövő státuszfrissítő üzenetek érkezésének intenzitása miatt van fontos szerepe, amíg a környezetváltozások szerverváltást eredményezhetnek.

A [22] cikkben Fernandes és társai megmutatták, mi történik, amikor egy avatár[1] különböző tetteket hajt végre a virtuális világban különböző helyeken és különböző hálózati feltételek között. A munkám kiterjeszti a [22]-ben bemutatott játékforgalom vizsgálatot az idő függvényében történő vizsgálattal, a játékkörnyezetekkel kapcsolatos információt kinyerve a forgalomból, továbbá újradefiniálom a játékostevékenységet, amit [17]-ban használtak, hogy játékos viselkedés információt gyűjtsenek. A munkám célja, hogy részletes karaktermozgás és játékvilág környezet-információ gyűjtését tegyem lehetővé, ahogy pl. [22]-ban, mindezt passzív mérésekből származtatván, az adott protokoll részletes csomagtartalom-szintű ismerete nélkül, ahhoz képest, ahogy pl. [37] vagy [30]-ban oldották meg a problémát.

A feltörekvő sokszereplős online valós idejű stratégiai játékok összetett játékszerver architektúrát igényelnek, ahhoz, hogy a nagyszámú játékos által irányítható egységnek az állapot információjának a továbbítását megvalósíthatóvá tegye. Ezt az architektúra tervezést támogatják a játékos pontos jellemzése alapján működő hálózati forgalom-szimulátorok [43]. Mindazonáltal az RTS játékokban tanúsított játékosviselkedésből származó forgalomváltozás jellemzőit nem vizsgálták, így ez ennek a munkának a fő motivációja. Bemutatok egy olyan passzív méréseken alapuló módszert, ami azonosítja a háborús időszakokat valós idejű stratégiai játékokban, ezáltal lehetővé téve nagyszámú játék elemzését.

2. Téziscsoport. *[J2, J4, J5, C4, C5] Módszereket javasolok sokszereplős online szerepjátékokban és valós idejű stratégiai játékokban a játékokhoz tartozó akciók felismerésére. [Disszertáció 3. fejezet]*

A játékos viselkedése és a játékvilág környezete információinak összegyűjtése nehéz. Munkám előtt a publikációk nem foglalkoztak azzal a problémával, hogy MMORPG

[1] a játékos virtuális világi megfelelője

forgalomból származó információt gyűjtsenek passzív mérésekből. Más munkák vagy magát a játékot módosítják, bekapcsolnak terminál-oldali naplóvezetést, amit később összegyűjtenek, vagy feldolgozzák a protokoll-üzeneteket és elemzik a csomagokat. Az általam javasolt forgalom jellemzőkön alapuló módszer széleskörűen alkalmazható, mivel ez protokoll-formátum független, és ennek a játékműfajnak a közös forgalmi jellemzőire épít.

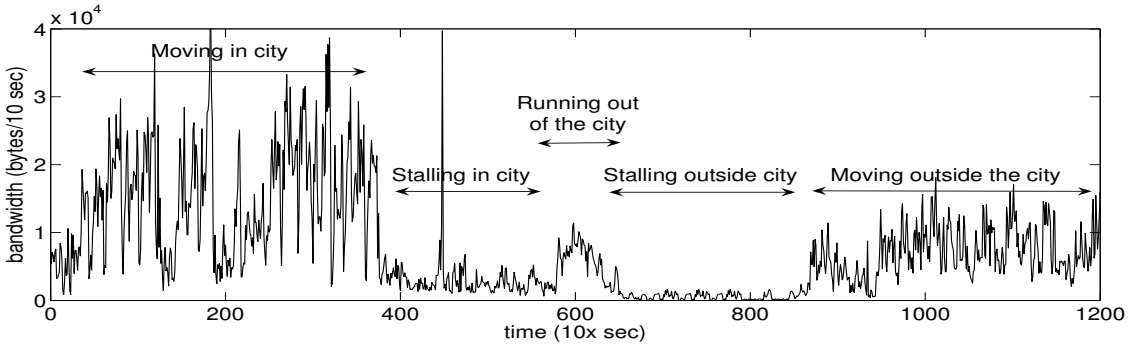
4.2.1. Játékos viselkedés felismerése MMORPG sokszereplős online szerepjátékokban

2.1. Tézis. *[J2, J4, J5, C4] Javasolok egy módszert sokszereplős online szerepjátékokban a játékosok viselkedésének és a játék környezetének felismerésére. A módszer a hálózati forgalmi ráta wavelet dekompozícióján alapszik és a környezet változásokat illetve a játékos viselkedést egy olyan állapotgép követi nyomon, ahol az állapotváltásokat a wavelet komponensek határozzák meg. A magasfrekvenciás-összetevő-változás állapotmátmenetet jelent az állás és mozgás között, miközben az alacsony-frekvenciás-összetevő-változás egy állapotátmenetet jelent a játék környezetek között. A javasolt módszer az MMORPG-k azon csoportján alkalmazható, amik a játékos környezetének frissítő információit a játékos közvetlen környezetéről küldi csak. Ez gyakorlatilag az összes ismert MMORPG-t magában foglalja.*

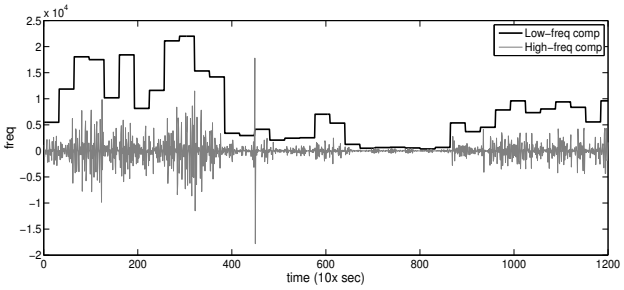
Az 6. ábra a 5. ábra forgalmának a wavelet dekompozícióját mutatja. Érdemes megjegyezni, hogy az álló és mozgó állapotok jól megkülönböztethetőek a nagyfrekvenciájú összetevőn: mozgás esetén mindez zajosabbá válik. A környezeti változások jól követhetőek a kisfrekvenciájú összetevőn. Az állapotgép követi, hogy a játékos városon belül vagy kívül van, és nem enged váratlan változásokat a környezetet tároló állapotban, ha rövid idejű tranziensek következnek be a bármelyik frekvenciaösszetevő esetén. A 7. ábra a definiált állapotokat és állapotátmeneteket mutatja a javasolt észlelési algoritmusban.

Az algoritmust úgy terveztem, hogy alkalmazkodjon az elemezett forgalomhoz, azaz ne kelljen tipikus sávszélesség tartományokat beállítani. Az algoritmus ezen tulajdonsága olyan előnyt ad, hogy bármilyen MMORPG játékforgalmat előzetes paraméterhangolás nélkül elemezni lehet. Fontos megjegyezni, hogy a módszer független a csomagirányoktól.

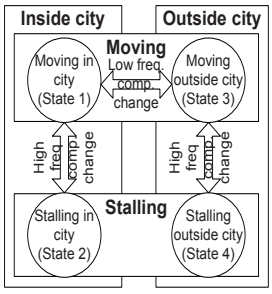
Kiértékelés. A módszer hálózati szimulátoroknak szolgáltathat játékosviselkedési paramétereket. A játékforgalom LRD tulajdonságát felfedtem és megmutattam, hogy



5. ábra. World of Warcraft mérés sávszélessége (bájts/10 másodperc)



6. ábra. World of Warcraft forgalom vizsgálata és a különböző játék környezetek és játékos mozgások hatásai



7. ábra. Állapotok és állapotátmenetek a javasolt módszerben

az a népszerű magyarázat, hogy a hosszútávú összefüggőség jelenlétét a forgalomban a játékos viselkedési periódusok nehéz-farkú tulajdonsága okozza [17] nem helytálló magyarázat.

Az első lépésben számos aktív mérés készült egy kontrollált környezetben felvéve a kliensek által generált hálózati forgalmat és manuálisan feljegyezve a játékos különböző állapotainak idejét. A mérés befejeztével a javasolt módszert lefuttattam a felvett hálózati forgalmon, majd az algoritmus által megjelölt állapotokat és a manuálisan feljegyzetteket összehasonlítottam. Egy állapot a játékos karakterének becsült tevékenysége egy adott környezetben egy 10 másodperces periódusban. A validáció ezen fázisában az állapotok 74% lett pontosan felismerve és az állapotok 14%-a egy szomszédos állapottal lett összecserélve. A szomszédos állapot azt jelenti, hogy egy állapot egy állapotátmeneten keresztül elérhető.

Második lépésben, a Silkroad Online [13] és World of Warcraft [10] játékokat választottam. A módszer pontosságát alaposan ellenőriztem egy automatizált mechanizmussal nagy számú mérésben videó feldolgozás és a hálózati forgalom korrelációja segítségével. A heurisztikus algoritmus megragadja a környezetnek és a játékos tevékenységének jellemzőit a képernyőn. A validációnak ebben a fázisában az állapotok 68% lett pontosan azonosítva és 19%-a az állapotoknak egy szomszédos állapottal lett összetévesztve. Fontos megjegyezni, hogy a sikerarány csökkenése a kézi validációval összehasonlítva a validációs heurisztikák pontatlanságai miatt is adódhat. Az első két validációs lépésben a megvizsgált forgalom összesen 9 órányi MMORPG játék volt.

A validációs eljárás harmadik fázisában különféle MMORPG-kal több aktív mérést is létrehoztam, hogy validáljam a módszer mennyire általánosan alkalmazható. Azt találtam, hogy a módszer jól működik, olyan esetekben ahol a játékos populáció magas és a karakter mozgatások jelentős forgalmat hoznak létre.

4.2.2. Csata felismerés RTS forgalomban

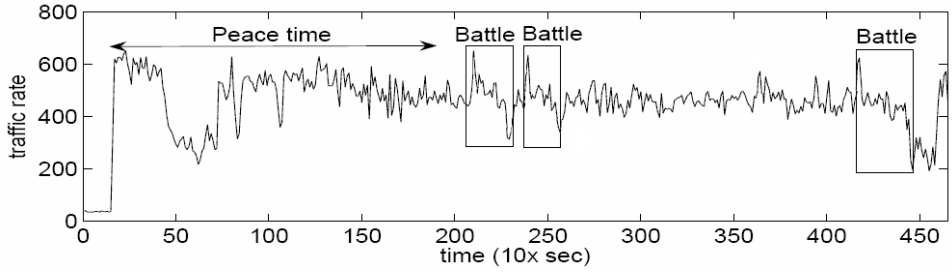
2.2. Tézis. *[C5] Javasolok egy módszert a csaták felismerésére a valós idejű stratégiai játékokban. A módszer a forgalom ráta heurisztikus vizsgálatán alapszik. A módszer a forgalom ráta lokális maximumait keresi, majd egy csökkenő trendet egy lokális minimum által behatárolva.*

Ahhoz hogy megragadjam ezeket a speciális jellemzőit a kliens oldali forgalomnak (lásd 8. ábra) kiszámítottam a lineáris regresszióját egy csúszó ablakban a szerver forgalmi rátájának. A csata indikátorai a következő tulajdonságok (lásd 9. ábra):

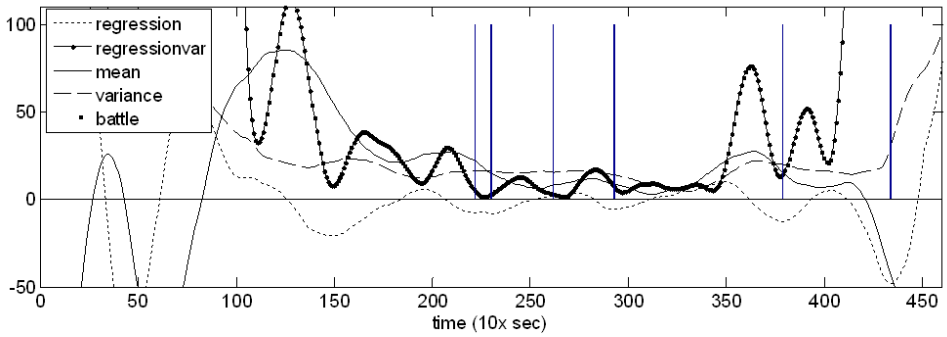
- *Van lejtő?* – Lokális minimum a gradiensben (p_1): Egy elsőfokú polinom $p(x)$ együtthatóit keresem, ami jól illeszkedik az adatokra, $p(x(i))$ az $y(i)$-ra, a legkisebb négyzetes értelemben. $x(i), i = n..n+k$ azt a megvizsgált időintervallumot jelenti, ahol az n a vizsgált időintervallum kezdete és az k a csúszó ablak hossza; $y(i)$ a forgalmi ráta az i^{th} időpillanatban. Az eredmény $p(x) = p_1x+p_2$ $\vec{p}$ egy két egység hosszú sorvektora a polinomegyütthatóknak csökkenő fok sorrendben.

- *Van növekedés a forgalom rátában?* – Lokális maximuma az átlag forgalom rátának egy csúszó ablakban

- *A lineáris illesztés jól sikerült?* – Lokális minimum az illesztett polinom és az eredeti idősor különbségének: $\min\{|y(i) - p(x)|\}, i = n..n + k$



8. ábra. Cossacks mérés forgalom intenzitása (csomagszám/10 másodperc)



9. ábra. Cossacks forgalom elemzése és a játékmenet alatti csaták hatása /a csatákat a függőleges vonalak jelzik/

- *Van egy nem szokványos ugrás a forgalom rátában?* – A gradiens normalizált szorása egy határ fölött van: $\tilde{\sigma}\{p_1^i x\} > c, i = n..n+k$

Kiértékelés. Az első lépésben számos aktív mérés készült egy kontrollált környezetben felvéve a generált hálózati forgalmát a kliensnek és manuálisan feljegyezve a csaták idejét a játékban. A mérés befejeztével, a javasolt módszert lefuttattam a felvett hálózati forgalmon. Az algoritmus által megjelölt csatákat és a manuálisan feljegyzetteket összehasonlítottam. A feljegyzett átfedő csataidőszakok a validációban és az algoritmus eredményeiben igaz pozitívnak tekintettem. A validációs fázisban RTS 15 órányi forgalmat gyűjtöttem és vizsgáltam meg. Az validáció ebben a fázisában 36-ból 33 csatát (92%) észlelt az algoritmus helyesen és 3 (8%) a jelzett csaták közül hamis pozitív volt.

A validációs eljárás második lépésében bemutattam egy automatikus módszert, hogy számszerűsítsem a csaták méretét és csatákat ismerjek fel automatikusan, nem csak a hálózati forgalomból, de magából a játékból is. Ezt a játék által lejátszott

hangok feldolgozásával valósítottam meg. A validáció ebben a fázisában a javasolt módszer 38-ból 33 csatát (87%) észlelt helyesen és 4 (11%) észlelt csata hamis pozitív volt.

A validációs eljárás harmadik fázisában különféle RTS-ekkel több aktív mérést is létrehoztam, hogy validáljam a módszer mennyire általánosan alkalmazható. Azt találtam, hogy a módszer annál megbízhatóbban működik minél nagyobb seregek vannak egy adott játékban.

Az eredményeket MMORTS játék szerver tervezésnél lehet *alkalmazni*. Nagy számú játékos egyidejű kiszolgálása MMORTS játékokban azt eredményezi, hogy a szerverterheléskiegyenlítés összetett problémává válik. A játékos több száz egységet tud irányítani, amik egy nagy virtuális világon vannak szétszóródva. Nem hatékony ha a játékos minden egyes egysége körüli összes információt közvetítjük. Cecin és társai [16] azt javasolják, hogy a szerver csak olyan adatokat közvetítsen a játékosoknak adott területekről ahova a játékosok ténylegesen koncentrálnak. A munkám segítségével az MMORTS játékok szimulációjához szükséges információt lehet gyűjteni.

Továbbá a megfigyelt RTS játékok jellemzőit összehasonlítottam valós háborúkból származó adatokkal. A való világ háborúinak az adatait a Correlates of War projektből [35] gyűjtöttem. Bemutattam, hogy a csatahosszak eloszlása hasonlóságokat mutat a játékvilágban és valós környezetben. Megmutattam, hogy a játékkörnyezet jól modellezi a való világot, de nem teljes másolata. Ez azt jelenti, hogy paramétereket, amiket valós világból gyűjthetőek bizonyos mértékig használhatóak de a legpontosabb eredményt a valós játékkörnyezet vizsgálata szolgáltatja.

4.3. Dinamikus mélycsomagvizsgálati minta

A jelenlegi mélycsomagvizsgálati módszerek fix bájt-minták után kutatnak a csomag tartalmakban. Ugyanakkor számos protokoll-mező van, pl., különösen a játékforgalommal kapcsolatban, mint amilyen a játékos mozgást közvetítő részek, amik folyamatosan változtatják az értéküket, azaz dinamikus minták. Más dinamikus minta típusokat is találhatunk pl., időbélyegek vagy sorozatszámok bármilyen általános protokollban.

Ismereteim szerint, játékprotokollokkal foglalkozó munkák nem léteznek még. Tan [37] és Liang [30] a játék protokolljának a szerkezetét a játék forráskódjából [37] vagy kézzel nyerték ki számos kísérlet eredményeképpen [30].

Pittman és társai [34] egy olyan szkript nyelvet [12] használtak amelyiket a World of Warcraft támogat, hogy bizonyos tevékenységeket automatizálni lehessen a játékban. Ezáltal kinyerték azon játékosok pozícióit akikkel a saját avatáruk mozgatása

közben találkoztak. A legjobban kapcsolódó publikációk az irodalomban automatikus protokoll minta előállítással foglalkoznak pl., [23], [27].

A téziscsoportban megvizsgálom, hogy az emberi viselkedés hatásai hogyan követhetőek a hálózati forgalomban csomagszinten. A játékosviselkedés és mozgással összefüggő csomagok a játékosok játékban elfoglalt pozícióit kódolják. Azt találtam, hogy ez a csomagszint-információ hasonló statisztikai tulajdonságokat mutat, azokhoz a statisztikai tulajdonságokhoz amik a valós világban az emberi mozgást jellemzik. A munkám egy olyan algoritmusnak a kifejlesztésére koncentrál, ami automatikusan megragadhatja a protokollok dinamikusan változó mezőit, különösen egy ismeretlen protokollnak a mozgással kapcsolatos mezőit. A követelmények szerint a játékosviselkedést az ismeretlen protokollokban kell tesztelni egy olyan hatékony módon, ami más publikációkhoz képest – pl. [37] FPS-ekhez vagy [30] MMORPG-khez – leegyszerűsített karaktermozgásmodell megalkotását tette szükségessé, a hatékony tesztelhetőséget előtérbe helyezve a pontos mozgás leíráshoz képest.

3. Téziscsoport. *[J2] Javasolok egy új mély csomagvizsgálati módszert, ami nemfix hosszúságú minták vizsgálatán alapszik. [Disszertáció 4. fejezet]*

4.3.1. Dinamikus minta létrehozása

A kihívás az volt, hogy a módszer automatikusan meghatározza a mezők hosszát, a számábrázolás irányát és megkülönböztesse őket egy egyszerű növekvő sorozatból. Lehetnek olyan felhasználások, amikor más módszerek nem tudnak megbirkózni a csomagtartalommal pl., csomagok Protokoll Fejléc Titkosítással majd egy mozgásmezővel. Olyan mélycsomagvizsgálati módszerekben amik statikus mezőkkel tudnak csak dolgozni nincs felismerhető minta.

3.1. Tézis. *[J2] Javasolok egy új dinamikus minta alapú forgalom osztályozó módszert. A bevezetett algoritmus a helyi és időbeli korrelációt használja ki és mintákat készít. A javasolt algoritmus egy adott adatfolyam csomagjait idősorként tekinti és megvizsgálja a különböző bájt-pozíciókban képzett idősorok H paraméterét.*

A modellem eredménye (lásd 3.2. tézis) az volt, hogy a koordinátákat frakcionális Brown mozgásként lehet tekinteni 1-hez közeli H paraméterrel. Emiatt ugyanazon mezők értékeinek erős korrelációjuk van. Ugyanezt a tesztet kell végrehajtani ezen korrelációs struktúra kereséséhez egy ismeretlen forgalomban.

Az algoritmus fBm-folyamat jelöltek után kutat. Megvizsgáltam, hogy a teszteredmények hogyan változtak ha az eredeti $fBm \sim \sum_{n=0}^{i}(2^8)^n X_n$ esetében, ahol az

i az érték bájt-ábrázolás hossza, ha szétbontják az eredeti idősort X_n összetevőkre. Szimulációkkal bebizonyítottam, hogy az eredeti eljárás H értéke a szétbontottnál mindig magasabb. Ez azt jelenti, hogy még akkor is, ha a tesztelt fBm-folyamat szórása annyira alacsony, hogy ez beleesik egy kisebb összetevő tartományába, akkor is az összes összetevőt figyelembe véve, a teszteredmények az eredeti folyamatnak magasabb H paramétert becsülnek. Ezeket a megfontolásokat valósítottam meg a javasolt algoritmusban.

Magas H paraméter úgy is kapható ha egy olyan idősort tesztelünk ami állandó méretű növekménnyel rendelkezik, mint pl. sorozatszámokat vagy időbélyegeket. A H paraméter tesztet ki kell egészíteni a jelölt bájt-pozíciók véletlenszerűségének vizsgálatával ahhoz hogy megkülönböztethessünk egy fBm-folyamatot és egy sorozatszámot. Ehhez a koordináták különbségeinek egy csúszó ablakban meghatározott relatív bizonytalanságát (lásd 3.2. tézis) kell kiszámolni, és ha az érték kevesebb 0.8-nál akkor ez egy sorozatszám állandó növekménnyel, ha a relatív bizonytalanság magas akkor az a mező mozgási csomagként tekinthető. A csúszó ablak amiatt szükséges, mivel a szomszédos értékek lehetnek állandóak is egy sétánál, (a lépés mérete). Ezzel szemben a távolság különbségek egy csúszó ablakban véletlenszerűséget mutatnak a több dimenziós mozgás miatt.

Ahhoz, hogy megkülönböztessük a sorozatszámokat az állandóértékektől egy speciális pozícióban, az eredeti idősorozatot intervallumokra osztjuk. A ládák száma a vizsgált tartománytól függ és a relatív bizonytalanságot meghatározzuk intervallumonként. Ha a relatív bizonytalanság alacsony, akkor a megvizsgált értékeket állandó értékeknek tekinthetjük.

Végül a még mindig üres mezőket a leghosszabb nem mozgás mezőkkel tölti fel az algoritmus az állapot és H paraméterteszt táblákból.

Kiértékelés. A módszert egy számos publikációban (pl. [23], [27]) gyakran használt módszerrel validáltam azáltal, hogy nyitott protokollokat elemeztem, pl., az RTP-t és a Gnutella protokollt, amiknek több dinamikus mezőjük is van. További bizonyítékként különféle saját célú felhasználásra írt játék forgalmat és más ismert forgalmi típusokat vizsgáltam meg a bemutatott algoritmussal és a fBm-hez hasonló szerkezet számos esetben létezett. Néhány példát lehet látni a 1. táblázatban.

4.3.2. Játékos mozgás modell

Nem egyértelmű, hogy az valós emberekre alkalmazott mozgás modellek pl., [29] játékvilágkörnyezetben tanúsított játékosviselkedésre is alkalmazhatóak lennének. A

Alkalmazás	Méret	Minta	Ábrázolási irány	Arány
Age of Mythology [1]	10	?{1} CCCCCCC{7} SS{2}	B	19%
Command & Conquer 3 [2]	12	FFFFCCCCCCC{11} ?{1}	L	25%
Command & Conquer 3	16	???W{4} CCCWW{5} CC{2} ?CCC?{5}	L	18%

1. táblázat. Néhány példa az alkalmazás mintákra /C-erős korreláció, F-flag, S-szekvencia szám, W-séta, ?-nem egyértelműen megadható/

vezérlőkörnyezet (botkormány, billentyűzet, egér) bizonyos korlátok közé szoríthatja a játékosviselkedést. A cél az volt, hogy egy olyan modellt alkossak, ami a) hatékonyan tesztelhető, b) az emberi mozgás által leírt koordináta viselkedés karakterisztikát jól leírja, c) kevés modell paramétert használ és d) nem játékspecifikus. Komplex többdimenziós modellek, amik az emberi viselkedés még pontosabb leírására törekszenek nem alkalmasak erre a feladatra.

3.2. Tézis. *[J2] Javasolok egy modellt, ami két független frakcionális Brown mozgás folyamat a játékos mozgásának leírására. A modell bemeneti paramétereket szolgáltat a dinamikus minta felismerő módszernek (3.1. tézis).*

A karaktermozgás legfőbb függőségjellemzői alapján a feltételezésem az volt, hogy a frakcionális Brown mozgás (fBm) le tudja írni a karaktermozgások változásait. Azt vártam, hogy az fBm-modellel jól leírható a karakterpozíció koordináták magas pozitív korrelációja. Azaz egy játékos várhatóan ugyanabba az irányba megy nagy valószínűséggel amerre elindult. Ezt a viselkedést egy egyszerű véletlen séta (pl. Brown mozgás) nem tudja leírni.

Két független fBm-modellt választottam, ahol egy fBm-et illesztek a mozgás X komponensére és egy másikat a mozgás Y koordinátájára. Az Z koordináta modellezését elhanyagoltam amiatt, hogy a terep ebben az esetben úgyis egy kényszerként szolgál, ebben az irányban a játékos a mozgását nem a játékos határozza meg.

Kiértékelés. A modell bemeneti paraméterekkel szolgál a dinamikus mélycsomagvizsgálati módszernek (lásd a 3.1. tézist).

Egy működő szélessávú hálózatból származó szűrt játék forgalmát elemeztem az R/S tesztel [40] és multifraktál wavelet analízissel [14]. Először az R/S tesztet [40] alkalmaztam a mérésből származtatott növekményeken. A 200 független idősoron végzett tesztek átlagban 0.73-as H-paramétert becsültek mind az X, mind az Y koordinátákra. Ezt a lépést ellenőriztem a H-paraméter wavelet alapú módszer becslésének megismétlésével ugyanazon az idősoron. A becsült átlagos H-paraméter 0.95-re

adódott az X koordináta és 0.96-ra az Y koordináta idősorára. Egy további validációs lépés a wavelet alapú H-paraméter becslők alkalmazása volt az eredeti kumuláns idősoron. A H-paraméter átlaga 0.89-re adódott az X koordináta és 0.9-re az Y koordináta idősorára

Azt a feltételezésemet, hogy a karaktermozgások jól modellezhetőek két független frakcionális Brown mozgással, az X és Y koordinátákra valós környezetből vett adatok statisztikai elemzésével validáltam. Egy megfelelő H paraméter a több mint 200 mintán végzett R/S becslő, növekményes logskála és kumuláns logskála elemző becsléseinek átlaga. Ez $H = 0.9$-ra adódott.

4.4. Nagy sebességű forgalomosztályozás párhuzamos architektúrán

Ebben a szekcióban megvitatom a Grafikus Feldolgozó Egységgel támogatott (GPU) forgalomosztályozáshoz szükséges rendszer komponenseket, ami képes sok Gbps sebességű forgalomosztályozásra átlagos személyi számítógépen.

A valós idejű forgalomosztályozás átlagos hardverrel számos korlátozásba ütközik, kezdve a csomagok hálókártyáról történő leemelésétől a forgalomosztályozásig bezárólag. Az aktuális személyi számítógépek nagy teljesítményűek – hálózati forgalommal kapcsolatos feladatoknál számos nagy kapacitású feldolgozó egység pl. a videokártya ki sincs használva –, de nehéz a hagyományos technikákkal munkába vonni őket. Az erőforrások teljes kiaknázásához az architektúrák pontos ismerete szükséges.

Goyal és társai [6]-ban, a determinisztikus véges automaták (DFA) és a kiterjesztett véges automaták (XFA) alapú mélycsomagvizsgálatot elemezték. A szerzők azt találták, hogy egy G80-as GPU alapú implementáció 9x gyorsabb volt a Pentium4 CPU implementációnál. Kihangsúlyozták azt a problémát, hogy az automatáknak az az adatstruktúrája (Mbájtos nagyságrendben van) nem illeszkedik az aktuális GPU architektúrák gyorsítótárába, ami alapvető lenne az optimális működés szempontjából. Ennek a megoldásnak a csomagfeldolgozási-sebessége 80,000 csomag/másodperc volt (FTP-vel, SMTP-pel és HTTP-mintákkal).

Vasiliadis és társai [38]-ban egy Behatolás Detektáló Rendszer (IDS) minta keresése volt a GPU-val támogatva, ami így 2.3 GBpbs sebességű rendszerátmenő teljesítményt nyújtott szintetikus forgalommal és 600 Mbps-es sebességre volt képes valós forgalommal meghajtva.

Vasiliadis és társai [39]-ban, a szerzők tovább fejlesztették a [38]-ban bemutatott

munkájukat a DPI motort CUDA architektúrára újaimplementálva [3]. A teljes rendszer áteresztőképesség 30%-al nőtt valós forgalmon mérve a [38]-hoz képest. [41]-ban a terhelés elosztás problémáját tovább finomították azáltal, hogy a keresendő nagy méretű szótárat szétosztották több processzor között. Mind a keresendő mintákat, mind az áttekintendő bementet particionálták a GPU teljes teljesítményének kihasználása érdekében.

Meg kell jegyezni, hogy [6, 38, 39, 41] publikációkban a fő feladat egy IDS megvalósítása volt a GPU-n. Ez azt igényli, hogy (a) hatalmas protokoll minta halmazokat kell felismerniük, (b) bárhol meg kell találni a bájt-mintát az adatfolyamban, és (c) hamis pozitív találat nem engedhető meg. Azaz az általam javasolt módszereket a forgalom osztályozási feladatra közvetlenül nem érdemes összehasonlítani a fenti módszerekkel. Ahogy az később látható, a módszereim gyorsabbak ezeknél a nyers csomag feldolgozási sebességet tekintve, de egy szűkebb problémakörön, a forgalomosztályozáson.

4. Téziscsoport. *[J3] Javasolok egy hatékony párhuzamos feldolgozáson alapuló forgalomosztályozó rendszert. [Disszertáció 5. fejezet]*

4.4.1. Mélycsomagvizsgálat grafikus feldolgozó egységgel

Azért, hogy egy GPU központú megoldással magas sebességet valósíthassak meg, a memória-hozzáférés többletköltséget minimalizálni kell. Ahhoz, hogy ezt elérjem, a célom az volt, hogy az bájt-mintákat egy minimalizált memórialábnyomon tároljam. Az állapotgépek processzor-hatékony módszerek a mintaillesztésre, de nem tárhatékonyak [6]. A hashek használatával adattömörítést lehet elérni, de nehéz helyettesítő jelek illesztését végrehajtani. Pl., tipikus megoldás ugyanazt az aláírást a lehetséges összes helyettesítőérték mindegyikével letárolni a hashben [19]. Ez a hash tábla méretének megsokszorozódásával és hamis pozitív találatokkal jár együtt. A GPU gyorsítótára kicsi, így ahhoz hogy nagy számú bájt-minta tárolása lehetségessé váljon adat tömörítés szükséges. A tömörítésnek a helyettesítő jelek használatát is támogatnia kell.

4.1. Tézis. *[J3] Javasolok egy memória optimizált mély csomagvizsgálati módszert. A hash algoritmus tömörítése segítségével a mély csomagvizsgálati módszerhez szükséges adatszerkezeteket a gyorsítótárakban helyezem el, aminek eredményeképpen nagy hatékonyságot lehet elérni olyan architektúrákon, amik támogatják a nagy mértékben párhuzamosított feldolgozást.*

(a) Bemeneti alkalmazás minta szótár (S)

App#1	a?b
App#2	aaa
App#3	?a?
App#4	??a

(b) Alkalmazás minták bitmaszkja (B)

101
111
010
001

(c) Abc-pozíció szótár (α)

	0	1	2
a	1100	1011	1000
b	1010	1110	1001

(d) Kódolt minta adatbázis (E)

0101
1111
1011
1000

2. táblázat. Bemeneti adat a mélycsomagvizsgálati algoritmusnak

Az alkalmazás mintákat tömörítéséhez egy Zobrist-hashelésen [45] alapú algoritmust javaslok. A szótárakat elraktározza a GPU gyorsítótárában és ugyanazt alkalmazza a bekódolásnál illetve amikor tesztelésre kerül a sor eldöntvén, hogy egy csomag melyik alkalmazáshoz tartozhat. A javasolt bájt-minta illesztés technika a következő bemenő adatot használja (illusztrációként lásd a 2. táblázatot). A kódolt mintáját egy adott alkalmazásnak úgy határozom meg, hogy a karakterek kódjait össze-xorolom abban az esetben ha a hozzátartozó bitmaszkban 1-es áll. Általánosan leírva: $E_i = \bigoplus_{j=0}^{|S_i|-1}(\alpha^j_{S^j_i} \bullet B^j_i)$. Illusztrációképpen, tekintsük a következő példát a 2. táblázat alapján.: pl., `a?b` a következőre kódolódik: $1100 \oplus 1001 = 0101$.

Mélycsomagvizsgálati adatstruktúrák a GPU memória architektúrában. Az *globális memória* (nem gyorsítótárazott) teret a hálózati csomagokkal töltöm fel. Mindegyik szál inicializálása alatt, a megfelelő tömbje a csomagoknak a globális memóriából a szál regisztereibe vagy az *osztott memóriájába* (gyorsítótárazott) másolódik. Így a globális memóriához való hozzáférés ugyanazoknál az adatoknál nem csökkenti a számítások sebességét.

Az *konstans memória* tér gyorsítótárazott, így egy olvasás a konstans memóriából csak egy memória olvasásba kerül a főmemóriából gyorsítótár hiba esetén, máskülönben ez éppen egy olvasásba kerül a konstans memóriából. Az előre kiszámított bemenő adatszerkezeteket a konstans memóriatérbe töltöm (az ábécé-pozíció-szótár, a bitmaszkok és a kódolt aláírások).

A foglalható állandó-memória-mérete az egész kernelnek a tesztkonfigurációban 64 kbájt volt. Ha a példa implementációmat vesszük, ahol az mintaadatbázis 4 bájt hosszú értékekből áll és akkor több 10 ezer minta illeszkedhetne a konstans memóriába.

Kiértékelés. A módszer képes reguláris kifejezés tömörítésére és nagy sebességű mélycsomagvizsgálatra.

	regexp	CPU	GPU	GPU_osztott
Átlag [thousand packets]	14	72	138	1844

3. táblázat. DPI teljesítmény összehasonlítás – Másodpercenkenként feldolgozott csomagszám

A tömörítési nyereség egyszerűen becsülhető, mivel az állandó méretű minták állandó méretű mintákra képződnek.

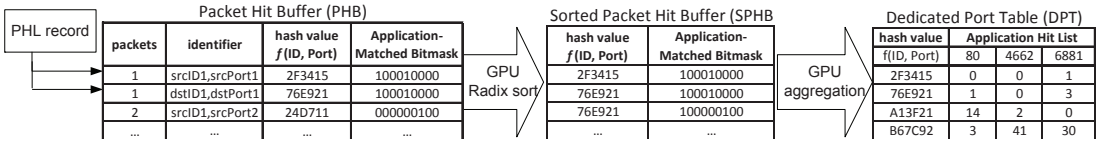
A 3. tábla összefoglalja a teljesítményvizsgálatokat, amik az átlagos csomag feldolgozási sebességre koncentrálnak. Összefoglalva az eredményeket, a javasolt módszer GPU központú megosztott memóriát használó verziója az eredeti reguláris kifejezés alapú módszerrel összehasonlítva két nagyságrend sebességnövekedését eredményezett. A GPU átlagos csomagfeldolgozás-sebessége kb. 1,800,000 csomag/másodperc, ami 500 bájtos átlagcsomagmérettel számolva 6.7 Gbps sebesség.

4.4.2. Csomag aggregáció GPU-val

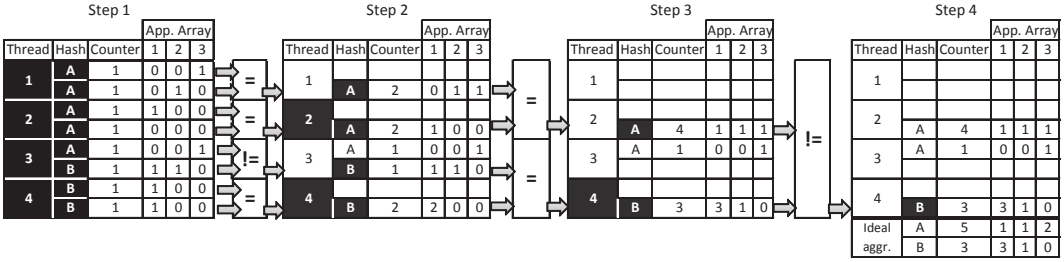
Néhány speciális forgalom típust nehéz egyszerűen mélycsomagvizsgálati módszer alapján azonosítani anélkül, hogy korábbi adatáramlási viselkedési csoportjait ne vennénk figyelembe. Ilyen forgalmak pl. a [26]-ben is ismertetett nemlétező társakkal való kapcsolat létrehozási kísérleteknél létrejövő rövid P2P adatfolyamok, a részben titkosított P2P forgalom vagy Skype. Ebben a részben javaslok egy módszert, mélycsomagvizsgálati és port alapú és kapcsolati minta alapú módszerek hatékony kombinálására a GPU-n.

4.2. Tézis. *[J3] Javasolok egy nagy sebességű összegző és frissítő módszert, ami a kapcsolat alapú forgalomosztályozást tudja támogatni párhuzamos feldolgozással. A módszer aggregálja a forgalomosztályozáshoz szükséges információt egy adatstruktúrába, ami egy tetszőleges pillanatban elérhető legpontosabb információt szolgáltatja.*

A javasolt módszer adatstruktúrája a Dedikált Port Tábla (DPT). A DPT célja az alkalmazás találatok tárolása a hoszt-port párok kommunikációs múltjának vonatkozásában az állandóan a legjobb elérhető forgalom osztályozási eredményt szolgáltatva. A DPT tárolja, hogy egy adott hoszt-port párt hányszor azonosítottak egy adott forgalom forrásának vagy céljának. A DPT egy olyan rekordokból álló lista ami ezen találatok számát tartalmazza. Azért, hogy elkerüljem a DPT csomagonkénti frissítését, ezen információ aggregációjához a GPU által a párhuzamos prefixum összeadás



10. ábra. Dedikált port keresés



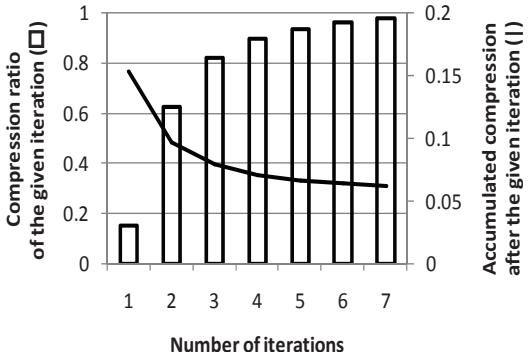
11. ábra. A csomaginfomáció aggregálása párhuzamos redukcióval

[24] egy kiterjesztését javaslom. Ez a módszer magas párhuzamosságot és nagy hatékonyságot ér el a GPU-val. A módszer nagymértékű párhuzamosíthatóságot és nagy hatékonyságot biztosít GPU-n megvalósítva (lásd 11. ábra).

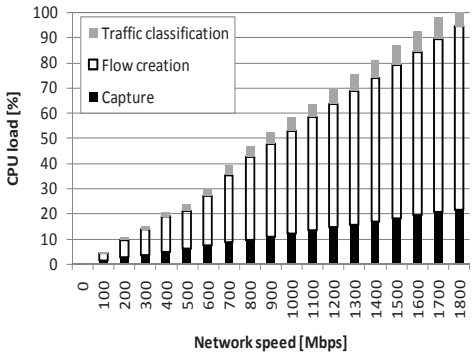
Az algoritmus bemenete az SPHB minden sorban egy számlálóval kiegészítve. Ez a számláló 1-re inicializálódik és ez mutatja a sikeresen aggregált csomagok számát (értsd. sorok számát). A 11. ábrán látható, hogy első lépésében minden szál összehasonlítja az általa vizsgálandó két sornak a hash kulcsát. Ha a kulcsok megegyeznek úgy megtörténik az aggregáció azáltal hogy az aggregációs számlálókat és a megfelelő alkalmazás találat számlálókat összegzi. Az eredményeket minden második sorban tárolja. A következő lépésben minden második szál végzi a munkát, összehasonlítva és összegezve az előbb feldolgozott sorokat, majd eltárolva az eredményeket minden negyedik sorban. Ez addig ismétlődik amíg nem marad feldolgozatlan sor. A párhuzamos redukció előnye a használt adat struktúrán az, hogy nincsen egyidejű, ütköző memória írás és olvasás. A szálakat minden lépés után szinkronizálni kell, hogy a memória tartalom konzisztenciáját biztosítsuk.

Kiértékelés. A javasolt felhasználás mellett a módszer egy nagy-sebességű multiprocesszor alapú adatfolyam-aggregációs-modul építőkockája is lehet a főprocesszor tehermentesítése érdekében.

Az aggregáció tömörítési arányát az APHB és az SPHB méretének hányadosaként értelmezhetjük. Egy valós forgalomból származó mérésből az SPHB-t feltöltöttem



12. ábra. A GPU alapú aggregáció tömörítési aránya az ismétlések számának függvényében



13. ábra. Rendszer összteljesítmény

256,000 sorral. Az ideális aggregáció által az APHB-t 4,560 sorba lehetett aggregálni, ami azt jelenti, hogy a tömörítési arány 1.78%. A GPU-t használva, az APHB-t 41,095 sorba lehetett aggregálni, ami 16%-os tömörítési arányt jelent.

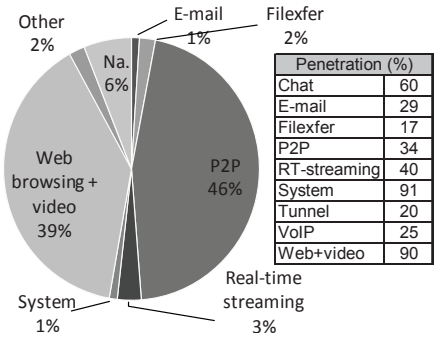
Mindazonáltal az aggregációarányt javíthatja az, hogy ha az aggregációt többször ismételjük. A tömörítési arányt az ismétlések számának funkciójaként ábrázolva látható a 12. ábrán. Azt lehet látni hogy a kezdeti tömörítési arány kb. megfeleződött a következő ismétlések által míg végül a végső tömörítési arány 8% lesz.

A következő kérdés az aggregáció teljesítménye, azaz mennyi ideig tart az aggregáció. A kísérletek az mutatják, hogy a GPU az aggregációs feladattal 4 nagyságrenddel gyorsabban birkózik meg mint a CPU a DPT frissítéssel, (hash beillesztést és frissítést tekintve). Vagyis az aggregáció nem okoz gyakorlati többlet költséget a teljes forgalom osztályozási eljárásban.

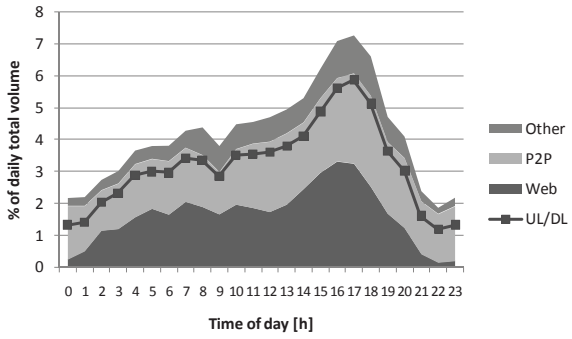
Figyelembe véve a teljes rendszer áteresztőképességét egy többprocesszoros környezetet tekintve, az elérhető sebesség a processzorok számával lineárisan skálázódik egészen 6-7 Gbps sebességig egy 4 magos rendszerben (lásd. az 13. ábrát).

5. Eredmények felhasználása

A bemutatott eredményeket felhasználhatjuk működő hálózatok forgalmának vizsgálatára. Az alábbiakban egy ilyen elemzés eredményét mutatom be. Az elemzett mérés három nap hosszan lett felvéve egy működő széles-sávú mobil hálózatban. A 14. ábra mutatja az átmenő forgalom mennyiségének arányát a felhasználók által



14. ábra. Alkalmazás forgalom mennyiség arány



15. ábra. Fel- és letöltési profil az idő függvényében

használt alkalmazások függvényében. Az előfizetők által használt alkalmazások típusa nagy mértékben befolyásolja a hálózati forgalmat. Látható, hogy a hálózatban a web böngészés és a P2P alkalmazások teszik ki a forgalom jelentős részét. Mivel a web böngészés letöltés irányú forgalmat generál főleg, emiatt a feltöltési forgalomhoz keveset járul hozzá.

Napközben a P2P és web forgalom állandó marad, míg éjszaka a P2P dominálja a forgalmat. A P2P forgalom megnövekedésének ideje egybeesik a felfelé menő forgalom arányának növekedésével. A web és a P2P forgalomtól mellett nagy mennyiségű streaming forgalom is van. Mivel a streaming forgalom letöltés domináns ezért csak a letöltési forgalomhoz járul hozzá. Más alkalmazások mint pl. az email vagy FTP forgalom csak néhány százalékot járulnak hozzá a teljes forgalomhoz.

Az 15. ábra a fel- és letöltési arányokat mutatja az idő függvényében. A feltöltési forgalom a teljes forgalom 1%-a fölött marad napközben és növekszik az éjszaka. Ez az éjszakai hiányzó web alkalmazások miatt van ami letöltés domináns, így a forgalma hiányzik az éjszakai teljes letöltési forgalomból.

Az alkalmazás forgalom arány mellett az is fontos hogy megvizsgáljuk, hogy a teljes felhasználó bázis mekkora része használ adott alkalmazásokat. A 14. ábra jobb oldalán az alkalmazás penetráció látható. A web böngészés meglehetősen elterjedt a felhasználók között és ezen alkalmazás rendszer üzeneteket is generál ahogy a web oldal látogatásoknál DNS lekérdezések generálódnak. Az üzenetküldő alkalmazások is nagyon elterjedtek a felhasználók között. Érdekes megfigyelni, hogy a felhasználóknak csak 30%-a használ email alkalmazásokat, ami egyrészt az üzenetküldő alkalmazások népszerűsége miatt van, amik képesek üzeneteket tárolni és elküldeni a címzettnek akkor amikor az online lesz, a másik ok pedig a webmail szolgáltatások népszerűsége miatt van. Ezen tényezők miatt a hagyományos email forgalom visszaesett. Az

ábrákból észrevehetjük, hogy a DNS forgalom (rendszer forgalom) korrelál a web böngészéssel, amíg a P2P forgalom nem generál jelentős DNS forgalmat.

Köszönetnyilvánítás

Köszönöm a felbecsülhetetlen segítségét a konzulensemnek, Dr. Molnár Sándornak és az Ericsson TrafficLab, illetve a Távközlési és Médiainformatikai Tanszék kollégáinak.

Hivatkozások

[1] Age of Mythology. http://www.microsoft.com/games/ageofmythology/.

[2] Command and Conquer 3. http://www.commandandconquer.com/default.aspx.

[3] CUDA Programming Guide 2.0. http://developer.download.nvidia.com/compute/cuda/2.0-Beta2/docs/Programming_Guide_2.0beta2.pdf.

[4] IANA port numbers, http://www.iana.org/assignments/port-numbers.

[5] MSN Messenger. http://join.msn.com/messenger/overview2000.

[6] N. Goyal, J. Ormont, R. Smith, K. Sankaralingam, C. Estan: Signature Matching in Network Processing using SIMD/GPU architectures. http://www.cs.wisc.edu/techreports/2008/TR1628.pdf.

[7] RFC 2246. http://www.ietf.org/rfc/rfc2246.txt.

[8] RFC 4251. http://www.ietf.org/rfc/rfc4251.txt.

[9] Skype. http://www.skype.com.

[10] World of Warcraft. http://www.worldofwarcraft.com/index.xml.

[11] Matlab, 1983-2009. http://www.mathworks.com/.

[12] Lua Programming Language, 1993-2008. http://www.lua.org/.

[13] Silkroad Online, 2005. http://www.silkroadonline.net/.

[14] P. Abry and D. Veitch. Wavelet Analysis of Long-Range-Dependent Traffic. *IEEE Transactions on Information Theory*, 44(1):2–15, 1998.

[15] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic Classification On The Fly. *SIGCOMM Comput. Commun. Rev.*, 36(2):23–26, 2006.

[16] F. R. Cecin, J. V. Barbosa, and C. F. R. Geyer. A communication optimization for conservative interactive simulators. In *IEEE Communications Letters Vol. 10, No. 9*, pages 686–688, 2006.

[17] K. Chen, P. Huang, C. Huang, and C. Lei. Game Traffic Analysis: An MMORPG Perspective. In *NOSSDAV '05*, New York, USA, 2005.

[18] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic Classification through Simple Statistical Fingerprinting. *SIGCOMM Comput. Commun. Rev.*, 37(1):5–16, 2007.

[19] L. Deri. High-speed dynamic packet filtering. volume 15, pages 401–415, New York, NY, USA, 2007. Plenum Press.

[20] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification Using Clustering Algorithms. In *Proc. MineNet '06*, New York, NY, USA, 2006.

[21] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Perform. Eval.*, 64(9-12):1194–1213, 2007.

[22] S. Fernandes, R. Antonello, J. Moreira, and C. Kamienski. Traffic Analysis Beyond This World: the Case of Second Life. In *NOSSDAV '07*, Urbana, Illinois, USA, June 2007.

[23] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In *MineNet '05*, New York, NY, USA, 2005.

[24] M. Harris, S. Sengupta, and John D. Owens. Parallel prefix sum (scan) with cuda. *Hubert Nguyen, editor, GPU Gems 3*, Aug 2007.

[25] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport Layer Identification of P2P Traffic. In *Proc. IMC*, Taormina, Sicily, Italy, October 2004.

[26] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *Proc. ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, August 2005.

[27] H. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *SSYM'04*, Berkeley, CA, USA, 2004.

[28] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. In *Proc. ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, August 2005.

[29] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong. SLAW: A Mobility Model for Human Walks. In *Infocom'09*, March 2009.

[30] H. Liang, I. Tay, M. F. N., W. T. Ooi, and M. Motani. Avatar Mobility in Networked Virtual Environments: Measurements, Analysis, and Implications. *CoRR*, abs/0807.2328, 2008. informal publication.

[31] A. McGregor, M. Hall, P. Lorier, and A. Brunskill. Flow Clustering Using Machine Learning Techniques. In *Proc. PAM*, Antibes Juan-les-Pins, France, April 2004.

[32] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *Proc. SIGMETRICS*, Banff, Alberta, Canada, June 2005.

[33] M. Perényi and S. Molnár. Enhanced Skype Traffic Identification. In *Proc. Valuetools07*, Nantes, France, 2007.

[34] D. Pittman and C. GauthierDickey. A Measurement Study of Virtual Populations in Massively Multiplayer Online Games. In *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for Games*, pages 25–30, New York, NY, USA, 2007. ACM.

[35] S. Reid and M. Reid. The correlates of war data on war: An update to 1997. In *Conflict Management and Peace Science, 18/1*, pages 123–144, 2000.

[36] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proc. Second Annual ACM Internet Measurement Workshop*, November 2002.

[37] S. A. Tan, W. Lau, and A. Loh. Networked Game Mobility Model for First-Person-Shooter Games. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for Games*, pages 1–9, New York, NY, USA, 2005. ACM.

[38] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In *RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, pages 116–134, Berlin, Heidelberg, 2008. Springer-Verlag.

[39] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis. Regular expression matching on graphics hardware for intrusion detection. In *RAID '09: Proceedings of the 12th international symposium on Recent Advances in Intrusion Detection*, pages 265–283, 2009.

[40] W. Willinger, M. Taqqu, and A. Erramilli. A Bibliographical Guide to Self-similar Traffic and Performance Modeling for Modern High-speed Network, 1996.

[41] C. Wu, J. Yin, Z. Cai, E. Zhu, and J. Chen. A hybrid parallel signature matching model for network security applications using simd gpu. In *APPT '09: Proceedings of the 8th International Symposium on Advanced Parallel Processing Technologies*, pages 191–204, Berlin, Heidelberg, 2009. Springer-Verlag.

[42] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proc. ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, August 2005.

[43] M. Ye and L. Cheng. System-performance modeling for massively multiplayer online role-playing games. *IBM Syst. J.*, 45(1):45–58, 2006.

[44] S. Zander, T. Nguyen, and G. Armitage. Automated Traffic Classification and Application Identification Using Machine Learning. In *Proc. IEEE LCN*, Sydney, Australia, November 2005.

[45] A. L. Zobrist. A hashing method with applications for game playing. *Tech. Rep. 88, Computer Sciences Department, University of Wisconsin*, 1969.

Publikációk

Folyóirat cikkek

[J1] A. Callado, **G. Szabó**, B. P. Gerö, J. Kelner, S. Fernandes, D. Sadok, : Survey on Internet Traffic Identification and Classification, *IEEE Communications Surveys and Tutorials*, 2009, Vol. 11, Num. 3, pp. 37–52.

[J2] **G. Szabó**, A. Veres, S. Molnár: On The Impacts of Human Interactions in MMORPG Traffic, *Journal of Multimedia Tools and Applications*, 2009, Vol. 45, Num. 1-3, pp. 133–161.

[J3] **G. Szabó**, I. Gódor, A. Veres, Sz. Malomsoky, S. Molnár: Traffic Classification over Gbit Speed with Commodity Hardware, *IEEE Journal of Communications Software and Systems*, 2010, Vol. 5, Num. 3.

[J4] **G. Szabó**, S. Molnár: Nagyon sok szereplős online szerepjátékok skálázási tulajdonságainak vizsgálata, *Híradástechnika*, 2007, Vol. LXII., Num. 11, pp. 20–26.

[J5] **G. Szabó**, S. Molnár: On The Scaling Characteristics of MMORPG Traffic, *Híradástechnika*, 2008, Vol. LXIII., Num. 1, pp. 40–47. (Selected Papers)

Konferencia cikkek

[C1] **G. Szabó**, I. Szabó, D. Orincsay: Accurate Traffic Classification. In *Proc., IEEE Wowmom 2007*, Jun 17-21, 2007, Helsinki, Finland.

[C2] **G. Szabó**, D. Orincsay, B. P. Gerö, Sándor Györi, Tamás Borsos: Traffic Analysis of Mobile Broadband Networks In *Proc., ICST Wicon 2007*, Oct 22-23, 2007, Austin, Texas, USA.

[C3] **G. Szabó**, D. Orincsay, Sz. Malomsoky, I. Szabó: On the Validation of Traffic Classification Algorithms In *Proc., PAM 2008*, April 29-30, 2008, Cleveland, Ohio, USA.

[C4] **G. Szabó**, A. Veres, S. Molnár: Effects of User Behavior on MMORPG Traffic In *Proc., ICC 2009*, June 14-18, 2009, Dresden, Germany.

[C5] **G. Szabó**, A. Veres, S. Molnár: Towards Understanding the Evolution of Wars in Virtual and Real Worlds In *Proc., ICSNC 2009*, September 20-25, 2009, Porto, Portugal.

Egyéb publikációk

[J6] **G. Szabó**, B Bencsáth: DHA támadás elleni védekezés központosított szűréssel, *Híradástechnika*, 2006, Vol. LIX. Num. 2, pp. 42–49.

[J7] A. Szentgyörgyi, **G. Szabó**, B. Bencsáth: Bevezetés a botnetek világába, *Híradástechnika*, 2008, Vol. LXIII. Num. 11, pp. 10–15. (Pollák-Virág díjas).