Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

IP forgalomválasztási táblák tömöríthetőségének időbeli vizsgálata

TDK dolgozat

Készítette:

Horváth Petra Rebeka

Konzulens:

dr. Rétvári Gábor

2020

Tartalomjegyzék

Kivonat

Az Internet világa az elmúlt években exponenciális növekedést mutat: egyre több eszköz és felhasználó található meg az online térben, egyre nagyobb forgalmat generálva, egyre több adatot keringetve a világhálón. Ez a rohamos növekedés kihívás elé állítja az egyre bővülő Internet hálózati infrastruktúráját. Feltevődik a kérdés, hogy nehézségekbe ütközik-e ez az egyre komplexebb és nagyobb rendszer a gyakorlatban? Előfordulhat-e, hogy a hálózati eszközök nem bírják el az emelkedő kapacitást és összeomlik az Internet? Hogyan oldható meg, hogy az egyre növekvő hálózatban található eszközök elbírják a terhelést, hogy a felhasználók közötti információk, adatok ugyanolyan gyorsan és olcsón közlekedjenek? Vajon hogyan változik időben ennek a feladatnak az alapvető elméleti és gyakorlati nehézsége, és vajon előrejelezhető-e egy pont, amikor ez a komplexitás már meghaladja a hálózati eszközök képességeit? A dolgozatban ezeket a kérdéseket vizsgáljuk egy teljesen új, információelméleti modell segítségével.

Az útválasztók (routerek) kulcsfontosságú építőelemei az Internet rendszerének, ugyanis ezek felelősek az itt közlekedő csomagok végpontok közti továbbításáért. A forgalomtovábbítási tábláikban található adatok alapján hoznak döntéseket arról, hogy adott csomagot milyen címre továbbítsanak a cél felé vezető úton. Ezen adatok tartalmazzák a világhálót használó összes eszköz lokális elérhetőségi információját. Az Internet címtér rohamos növekedése azonban jelentősen megnöveli a router tábláinak méretét, azzal fenyegetve, hogy az egyre nagyobb táblák nem fognak beleférni az útválasztók gyors elérésű memóriaegységeibe.

A dolgozat arra a kérdésre keresi a választ, hogy hogyan változott az évek során a rohamosan növekvő forgalomtovábbítási táblák valódi információtartalma. Elméleti becsléseket adunk meg a táblák entrópiájára vonatkozóan, továbbá speciális kódoló algoritmusokat alkalmazva bemutatunk egy hatékony tömörítési módszert. Az így kapott elméleti és gyakorlati eredmények által rálátást nyerünk a forgalomtovábbítási feladat információtárolási komplexitására. Módszereinket a BME-TMIT és a HBONE, a magyar akadémiai gerinchálózat fenntartója közti együttműködésben 2013 óta gyűjtött forgalomtovábbítási tábla adathalmazokon értékeljük ki. A bő hét évnyi adat elemzése után kapott statisztikák meglepő következtetésre vezetnek: a forgalomtovábbítási táblák méretének rohamos növekedését nem feltétlenül követi a tömörített méret hasonló arányú növekedése, azt sugallva, hogy az Internet útválasztási rendszere nagy valószínűséggel fenntartható lesz a közeljövőben is a hálózat gyors expanziója ellenére.

Abstract

The Internet has shown exponential growth in recent years: more and more devices and users can be found in the online space, generating more and more traffic, and increasing data circulation on the World Wide Web. This rapid growth poses a challenge to the ever-expanding Internet network-infrastructure. So, the question arises: are there difficulties appearing in this increasingly complex and large system? Is it possible that the network devices can't stand the rising traffic, and the Internet collapses? How can we succeed in making the network devices bear the load of the growing network, while keeping the information between users circulating as fast and as inexpensive as before? How does it change the theoretical and practical difficulty of this task over time, and is the moment when the complexity exceeds the capability of network devices predictable? In this study we answer these questions using a completely new information-theory model.

Routers are a key element in the Internet ecosystem as they are responsible for data packets reaching their destination. They choose which address to forward a package to, based on the data in the routers' FIB (Forwarding Information Base) tables. This data includes all local contact information of all devices using the Internet. However, the rapid growth of the World Wide Web address space significantly increases the size of router tables, raising the issue of the tables, growing in size, not fitting in the fast-access memory unit of routers.

This study seeks to answer how the actual information content of fast-growing FIB tables has changed over the years. We are providing theoretical estimates for the entropy of the tables. Furthermore we present an efficient compression method, using special coding algorithms. Through the obtained theoretical and practical results we gain insight into the information storage complexity of packet forwarding. These methods will be evaluated on a set of FIB table data that was being collected since 2013, by the cooperation between BME-TMIT department and HBONE, the maintainer of the hungarian academic backbone. The statistics obtained by analyzing seven years of data lead to a surprising conclusion: the rapid increase in the size of the FIB tables is not necessarily matched by a similar increase in the compressed size, suggesting that the Internet routing system will probably be sustainable in the near future despite the fast expansion of the network.

1. fejezet

Bevezetés

1.1. Háttér és motiváció

A jelenkor meghatározó részét képezi az internet világa, jóformán ez hálózza be a mindennapjainkat. Ez a rendszer, ami az évek során kialakult és nagyon hatékonyan működik, egy nagyon komplex, összetett hátterű hálózatrendszert követel meg, aminek bírnia kell a terhelést. Napjainkban még inkább láthatóvá vált, hogy az internetforgalom milyen mértékben emelkedik, és érezhető, hogy mennyire fontos az, hogy a hálózat elbírja ezt a terhelést. Az internetforgalom növekedése mellett növekedik az internetet használó eszközök száma is. Ezeknek a növekedéseknek a következményeként az útvonalválasztók (routerek) amik arról gondoskodnak, hogy az interneten közlekedő csomagok eljussanak a forrástól a célig, egyre leterheltebbek lesznek, egyre több számítási kapacitásra lesz szükségük, egyre nagyobb tárolási kapacitást igényelnek a tábláik, hogy eltárolhassák bennük a számítási eredményeiket. A tárolási kapacitásaink viszont végesek, és hamarosan már nem lesz elegendő tárhely minden fontos információ eltárolásra, aminek eredményeképp meg fog nőni a csomagok továbbítási ideje.

A dolgozat többek között azt a kérdéskört tárgyalja, hogy hogyan változott az útválasztó eszközökben található forgalomtovábbítási táblák információtartalma az elmúlt években, hiszen ezen táblákban található adatok alapján hoz döntést az útválasztó eszköz az Interneten közlekedő csomagok cél felé vezető útvonaláról. Belátható, hogy ezen táblák hatékony működése elengedhetetlen része a csomagtovábbításnak és ezáltal az Internet működésének.

A forgalomtovábbítási táblák méretei rohamos növekedésben vannak, és ez a gyors növekedés, amivel a kapacitások nem tudnak lépést tartani, aggodalomra ad okot. A FIB táblák méretnövekedésének a következtében az útválasztási folyamatok komoly skálázási problémákkal kerülnek szembe [11, 20]. Egy megtörtént ilyen eset volt az '512k day' néven elhíresült esemény, amikor 2014-ben jópár útválasztó eszköz elérte az 512 ezer soros memóriakorlátot, aminek hatására leállt az Internet egy egész napra jelentős nagyságú területen [8]. Látható tehát, hogy a skálázhatóság komoly problémákat okozhat az Internet világában. A felismerés azzal kapcsolatban, hogy ez a probléma valós, az *IAB - Internet Architecture Board* bizottság keretein belül is megfogalmazódott. 2007-ben workshop-ot tartottak az Internet skálázhatósági problémáiról amelyről külön RFC4984 is született [13].

A dolgozat megoldást keresve a forgalomtovábbítási táblák tárhelykapacitási problémáira, azt vizsgálja, hogy mennyire lehet ezen táblákat tömöríteni, továbbá, hogy a valóságban ténylegesen tömöríthetőek-e olyan mértékben, mint ahogy azt a becslések alapján gondolnánk.

1.2. Feladat megfogalmazása

A dolgozatban tárgyalt feladatkör két nagyobb egységre osztható. Az első rész egy olyan rendszer megtervezését és implementálását foglalja magába, amely képes forgalomtovábbítási táblákat feldolgozni, a bennük található adathalmazokon számításokat végezni, és szolgáltatja a táblák elméleti és gyakorlati tömöríthetőségével kapcsolatos eredményeit. A második rész az implementált rendszer eredményeit felhasználva több évnyi forgalomtovábbítási tábla adathalmazon végez elemzéseket, amelyekből kinyert információkat statisztikák készítésére használja.

1.2.1. Keretrendszer tervezése és implementációja

A bemeneti változóként megkapott forgalomtovábbítási táblákon akkor végezhetőek elemzések, készíthetőek statisztikák, ha rendelkezésre áll egy olyan rendszer, amely ezt az adathalmazt fel tudja dolgozni, algoritmusokat, számításokat tud rajta végrehajtani. Fontos szempont a rendszer tervezésénél, hogy a rendszernek képesnek kell lennie elméleti tömöríthetőségi becsléseket adni a feldolgozott táblákra, továbbá tömörítve a táblákat, gyakorlati tömörített méretkorlát értékeket kell tudnia rendelkezésre bocsátani.

A rendszer egységeinek elkészítését elméleti tájékozódás és információgyűjtés előzi meg, annak érdekében, hogy megfelelő és helyes legyen a működés, az egységek illeszkedjenek egymáshoz és dolgozni tudjanak egymással. Ha megvalósult a helyes működés, akkor feldolgozhatóak lesznek a forgalomtovábbítási táblák és ki lehet nyerni belőlük a statisztikák elkészítéséhez szükséges adatokat.

1.2.2. Adathalmazon való kiértékelés

Az implementált rendszert a HBONE, a magyar akadémiai gerinchálózat üzemeltetői által rendelkezésre bocsátott hét évnyi adathalmazon futtatjuk. A munka során közel 1300 forgalomtovábbítási tábla tartalmát dolgozza fel a rendszer, és készít elemzéseket az eredmények alapján. Statisztikák készülnek a tömörítetlen táblák tulajdonságairól, az elméleti tömöríthetőségi becslések értékeinek az évek során bekövetkezett változásáról, és az elméleti tömöríthetőségi és gyakorlati tömörített méretkorlát összefüggéséről.

Az elemzések által kapott eredmények bemutatják a forgalomtovábbítási táblákon bekövetkezett változásokat, és rálátást adnak arra, milyen irányba változtak ezek a táblák az évek alatt, továbbá mekkora a változás mérete. Statisztikák készítése által láthatóvá fog válni, hogy az adathalmazok tömörítetlen és tömörített változatai milyen viszonyban vannak egymással, megadva azt, hogy mennyire is tömöríthető a felhasznált eszközökkel egy forgalomtovábbítási tábla, illetve milyen előnyei vannak a tömörítésnek.

1.3. Struktúra

A dolgozat felépítése távolról indulva és fokozatosan egyre közelebb érve halad végig a feladat strukurális rendszerén. Kezdetben tárgyalja a munka során felhasznált elméleti komponenseket, majd bevezeti az olvasót az implementáció részleteibe, elmagyarázva a munka során felépített rendszer komponenseit és működését, végezetül pedig elemzi a kapott eredményekből készített statisztikákat és következtetéseket von le belőlük.

A 2. fejezet biztosítja a munka megértéséhez szükséges elméleti hátteret, tartalmazza azokat az információkat, amelyek kulcsfontosságú részei a feladatnak, és amely részek felhasználásra kerültek a rendszer implementálásában, elsajátításuk elengedhetetlen részét képezte a munka megvalósulhatóságának.

A feladat elméleti hátterének megismerése után következik annak a rendszernek a bemutatása, amely a munka során megtervezésre és implementálásra került. A 3. fejezet bemutatja a rendszer egységeit, működését, az elsajátított elméleti tudás gyakorlatban történő megvalósulását. A fejezetben az olvasó információt kap továbbá a munkához felhasznált környezeti eszközökről, amelyekben a rendszer megvalósult, és a tesztelés részleteiről is.

Végezetül az elméleti és gyakorlati részek megismerése után következik annak ismertetése a 4. fejezetben, hogy milyen és mekkora mennyiségű adathalmaz lett feldolgozva az implementált rendszerünk által, továbbá tárgyalásra kerülnek a kapott eredmények és az azokon végzett statisztikák, amelyek az elméleti és gyakorlati tömöríthetőségi jellemzőket mutatják be.

Az utolsó fejezetben a feladat összefoglalása és értékelése zárja le az elvégzett munkát, továbbá még megfogalmazódnak a jövőbeli tervek a téma folytatásával kapcsolatban.

2. fejezet

Elméleti háttér

2.1. IP forgalomtovábbítás menete

Kívülről az Internet világa elképzelhető küldő és fogadó felek közti információáramlások megvalósulásaként. De mi történik a háttérben? Hogyan tudnak az információhalmazok folyamatosan oda-vissza közlekedni akadálymentesen és gyorsan az Interneten? Csomagkapcsolt hálózatokban az információk csomagok sokaságaként haladnak a küldő és a fogadó felek között. A csomagok hordozzák magukkal azokat a meta-adatokat, amelyek alapján a rendszer tudja, hogy hova kell őket küldeni, hova tartanak. Ezen adatok közül az egyik legfontosabb a cél IP-cím, hiszen e nélkül nem lehetne tudni, hogy hova is kell megérkezzen az adott csomag. Abban az esetben amikor a kiinduló és a cél csomópont nem közvetlenül kapcsolódik egymáshoz, a küldő féltől elinduló csomag rendszerint ún. köztes hálózati csomópontokon halad át, hogy megérkezhessen a célhoz. Ezek a köztes csomópontok lehetnek routerek, switch-ek, bridge-ek, vagy más típusú eszközök is. A csomagok elküldésére a hálózat komplex felépítése és nagysága miatt rendszerint több útvonal is rendelkezésre áll.

Azt a folyamatot amely eldönti, hogy egy adott csomag a hálózatban milyen útvonalon haladjon végig, *útválasztásnak* (routing) nevezzük. Az útválasztás általában útválasztó táblák (RIB - Routing information Base) alapján történik. Ezek a RIB táblák olyanok, akár egy térkép: minden információt tartalmaznak a hálózat topológiájáról. A RIB táblákat az egyes útválasztó protokollok építik, minden protokoll a saját tábláját. Ezekben nagy mennyiségű protokoll-specifikus adat található meg minden címhez [10].

Azt, amikor minden útválasztó eszköz saját maga határozza meg lokálisan, önállóan hogy milyen döntéseket hoz egy csomag továbbításával kapcsolatban, *hop-by-hop* routingnak nevezzük. Ebben az esetben a routing tábla mindig tartalmazza az összes a router által elérhető célt. Ez a módszer napjaink legismertebb és legalkalmazottabb megoldása az IP hálózatok világában [10].

Ugyanakkor a RIB táblákat általában nem szokták direkt módon felhasználni csomagtovábbítási feladatok elvégzéséhez. Ehhez inkább egy ún. *forgalomtovábbítási táblát (FIB - Forwarding Information Base)* használnak, ami a RIB tábla alapján jön létre útválasztó algoritmusok segítségével [10]. Ha egy célhálózat felé több lehetséges útvonal is található a RIB táblában, akkor ezek közül mindig az algoritmus szerinti legjobbnak ítélt útvonalak fognak bekerülni a FIB táblába. A FIB tábla mondhatni a hozzá tartozó RIB tábla egyszerűsített, kicsinyített mása, ami mindig csak az ideális útvonal lehetőségeket fogja tartalmazni.

Az egyes routerek a FIB táblában minden, a hálózatban elérhető prefixhez tárolják annak a szomszédos routernek az IP címét (nexthop), amelyen keresztül a prefixbe tartó csomagok továbbításra kerülnek majd, és ezen adatokat felhasználva dől el, hogy egy csomag merre halad a hálózatban. Egy adott hálózat prefix értékét úgy lehet megkapni, hogy

a hálózati címet maszkoljuk a hozzá tartozó netmaszk értékkel. A netmaszk arra szolgál hogy felossza az IP-címeket kisebb címtartományokra a hálózatok számára és meghatározza az adott hálózathoz kapcsolható hosztok számát. Egy hálózatot az IP-cím és netmaszk páros határoz meg. A FIB táblák tartalmazzák a hálózat címeket, mindegyikhez hozzárendelve egy nexthop címet. Ezek a táblák minden útválasztó eszközben megtalálhatóak, és csomópontonként újra és újra felhasználásra kerülnek. Ezáltal tud a csomag eljutni a végponthoz. Ezen a ponton érdemes megemlíteni, hogy felmerülhet olyan probléma, hogy egy adott cél IP-cím több alhálózati prefixre is illeszkedik. Ez azt jelenti, hogy több lehetséges irány kínálkozik a csomag továbbküldésére, amelyekből ki kell választani egyet, nyilvánvalóan a legmegfelelőbbet. De hogyan is lehet megtalálni az optimális megoldást? Erre a kérdésre az *LPM (Longest Prefix Match)* segít megadni a választ. Ez egy olyan algoritmus, mely képes eldönteni, hogy a FIB táblában lévő melyik hálózat-nexthop összerendelés a legjobb döntés a csomag hatékony célba jutása érdekében. Ha a célcím több alhálózati prefixre is illeszkedik, az algoritmus kiválasztja a leghosszabb illeszkedést ezek közül, és az ehhez tartozó nexthopra fogja továbbítani a csomagot [6, 4]. Szemléltetésként egy rövid példa az LPM működésének megértéséhez: tegyük fel, hogy a FIB táblánk tartalmazza a következő két hálózati címet: 192.164.32.12/24 és 192.164.0.0/16. Továbbá érkezik egy csomag, aminek a célcíme: 192.164.32.4 . Ezen címek bináris formára hozzuk, amelynek eredménye a 2.2 ábrán látható.

Decimális forma	Bináris forma
192.164.32.12/24	11000000.10100100.00100000.00001100
192.164.0.0/16	11000000.10100100.00000000.00000000
192.164.32.4	11000000.10100100.00100000.00000100

2.1. ábra. IP címek decimális és bináris alakban

A 2.2 ábra első két sora a példában szereplő két hálózat, ezek alatt pedig a cél IP-címe található meg. A hálózatok esetében piros szín jelöli a netmaszk által elhatárolt részt, az IP első része az hálózati címet fogja azonosítani, a második rész pedig a hálózatban található eszközök számára lesz fenntartva. Ha a netmaszkot az n érték jelöli, akkor a hálózatban $2^n - 2$ eszköz számára lesz elérhető IP-cím. A 2.2 ábra utolsó sorában található célcímet ráillesztve a fölötte lévő sorokban található űhálózati IP-címekre, láthatjuk, hogy sokkal nagyobb részben illeszkedik az első hálózati prefixre, mint a másodikra. Az LPM algoritmus összehasonlítja az illeszkedéseket, és ezek alapján hoz döntést. A példában említett csomagot az első címhez tartozó nexthopra küldené tovább, hiszen ez a hosszabban illeszkedő prefix találat.

Röviden összefoglalva: az előző rész bevezetett az IP forgalomtovábbítás világába, ismertetve a működés alapköveit, ide sorolva az *útválasztó eszközök*et amelyek a hálózatot alkotó és összekötő csomópontok; a *hop-by-hop routing* módszert, amelyet ezen eszközök alkalmaznak csomagtovábbításra; a *RIB* és *FIB* táblákat, amelyek megtalálhatóak a routerekben; és az *LPM* algoritmust, ami segíti a csomagtovábbítás folyamatát a FIB táblákban.

2.2. Forgalomtovábbítási tábla tömöríthetőségére vonatkozó elméleti becslések

2.2.1. Prefix kódok szerepe a forgalomtovábbításban

Az Internet rohamos növekedésének hatására az útválasztó eszközökben található útválasztó és forgalomtovábbítási táblák rohamosabban bővülnek, mint ahogy azzal a tárhely-

kapacitások tartani tudnák a lépést. Ez problémákhoz vezethet a hatékony csomagtovábbítás terén. Ahhoz, hogy az esetlegesen előállható problémákat előre lehessen látni illetve, hogy fel lehessen rájuk készülni, szükséges számításokat, becsléseket végezni, mert ezek által lehet megismerni, hogy mennyire valósak a feltételezések a táblák jövőbeli tárhely-problémáira vonatkozóan.

Az Interneten közlekedő csomagok akkor tudnak hatékonyan és zavartalanul közlekedni, ha a FIB táblák megfelelően működnek: azaz gyorsan tudnak keresést végezni a tábláikon, és gyorsan tudnak választ adni arra, minden beérkező csomag esetében, hogy azt melyik nexthopra kell továbbítani. Ha a FIB táblák rendelkezésre álló tárhelyei kimerülnek, lassulni fog a csomagtovábbítás, hiszen nem fog beférni minden szükséges adat az útválasztók gyors elérésű memóriaegységeibe. Erre az egyik lehetséges megoldás: a táblák tömörítése olyan módszerekkel, amelyek nem rontják a táblák működésének hatékonyságát.

De milyen mértékben tömöríthető veszteségmentesen egy FIB tábla? A tömöríthetőség mértéke megkapható táblánként, az azokon számolt *entrópia* értékének kiszámításával. Ahhoz, hogy egy FIB táblán entrópiát lehessen számolni, először olyan formára kell hozni, hogy ez megvalósítható legyen. Ennek az első lépése, hogy a tábla soraiban található hálózatcímeket *prefix kód*okká kell átalakítani. A prefix kód egy olyan kódolása a címeknek, amely egyértelműen azonosít minden elemet, amit tartalmaz. Nem fordulhat elő olyan, hogy két elemre is illeszkedjen egyazon kód[1]. Ez azért nagyon fontos első lépése a folyamatnak, mert alapfeltevés, hogy a feldolgozott IP-címek, amik ki lettek emelve a FIB táblából, ne keveredjenek össze, illetve egyértelműen megtalálhatóak legyenek a feldolgozás bármely fázisában [12]. A FIB táblában található IP-címeknek a prefix kódja a netmaszk által meghatározott hálózat azonosító cím bináris formában. Ezt szemlélteti a 2.2 ábra, ahol jól látható, hogy a 192.164.0.0./16-os címnek a prefix kódja bináris formában a netmaszk által meghatározott hálózat cím lenne: 1100000010100100 [2].

2.2.2. Prefix fa fogalma

A FIB tábla hálózatcímeiből előálltak a prefix kódok. A folyamat következő lépése ezeket felhasználva előállítani a kapott kódokból egy *prefix fát*. A prefix fa a prefix kódoknak egy szemléletesebb ábrázolása és ez mellett könnyebb rajta számításokat végezni, műveleteket végrehajtani. A prefix fa egy olyan bináris fát jelent, ami hasonlóan a prefix kódhoz, egyértelműen azonosít minden elemet, amelyet tartalmaz. További előnye, hogy a FIB táblából prefix fába alakítás egy oda-vissza irányú kapcsolat, ami azt jelenti, hogy bármikor át lehet lépni egyikből a másikba anélkül, hogy adatvesztés lépne fel. Amikor egy FIB tábla prefix kódjaiból épül fel egy prefix fa, akkor az ágakon végighaladva mindig meg lehet találni egy adott IP-címet és a megfelelő csomópontban pedig a hozzá tartozó nexthop értéket.

De hogyan is lehet a FIB tábla IP-címeiből kapott prefix kódokból prefix fát építeni? A 2.2 ábra egy kis méretű FIB táblát szemléltet, és az ebben található hálózat-nexthop párosok alapján fog kialakulni egy prefix fa az alábbi módon: első lépésben minden hálózati címből prefix kódot kell készíteni, továbbá a nexthopokat az egyszerűség kedvéért azonosítókkal ellátva átláthatóbb lesz majd a prefix fa ábrázolása.

Ennek a lépésnek az eredményét szemlélteti a 2.3 ábra. Ezen a ponton rendelkezésre állnak az első oszlopban a prefix kódok, a másodikban pedig a nexthop azonosítók, tehát el lehet kezdeni felépíteni a fát. Kiindulási pont a fa gyökere, kezdetben ez az egy üres csomópont alkotja a fát, amely a prefix kódokon egyesével végighaladva, bitről bitre fog felépülni. Ha az adott prefix kód például a 01, akkor a gyökértől elindulva először a nullás

[1] Ahogy a neve is árulkodó, egyik kód sem folytatása egyetlen másik kódnak sem.

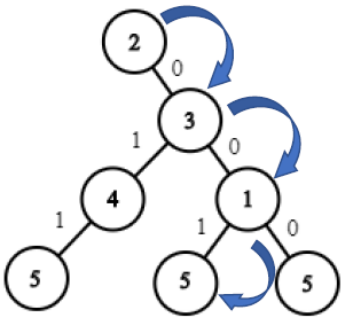
[2] Ezen a ponton már nem szükséges használni az IP-cimeknél használatos elválasztó pontokat, a prefix kód bináris számsorozatként értendő.

Destination	Nexthop
0.0.0.0/0	1.2.3.4
0.0.0.0/1	2.3.4.5
0.0.0.0/2	3.4.5.6
64.0.0.0/2	4.5.6.7
32.0.0.0/3	5.6.7.8
16.0.0.0/3	5.6.7.8
96.0.0.0/3	5.6.7.8

2.2. ábra. FIB tábla

Binary	NexthopID
-/0	2
0/1	3
00/2	1
01/2	4
001/3	5
000/3	5
011/3	5

2.3. ábra. Segédeszközök a FIB táblához



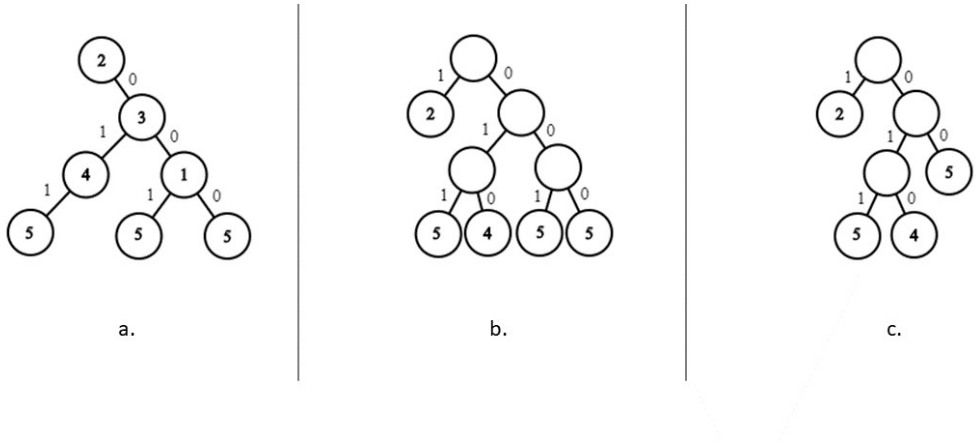
2.4. ábra. Prefix fa - a FIB tábla alapján

ágon, majd ezt követően az egyes ágon haladunk végig, létrehozva az adott ágat, ha az még nem része a fának, továbbá az adott prefix kód végéhez érve a fa megfelelő csomópontjába beírva a nexthop azonosító értékét. Ezt minden prefix kódra megismételve felépül a prefix fa, amely információtartalmilag megegyező erejű a FIB táblával, amiből épült. A 2.4 ábra egy ilyen prefix fát ábrázol, amely a 2.2 ábrán található FIB tábla tartalmát szemlélteti. A prefix fa helyessége könnyen ellenőrizhető a 2.3 ábra prefix kódjai alapján. Például kiválasztva a 001-es prefixet, majd a gyökértől elindulva kétszer a nullás, majd végül az egyes ágra lépve látható, hogy az 5-ös azonosítójú csomópontnál állt meg a keresés. Ennek helyessége a 2.3 ábrán látható táblázat NexthopID oszlopából kiolvasható: a 001-es prefix kódhoz tényleg az 5-ös azonosító tartozik.

2.2.3. Leaf-pushing algoritmus

Az előző fejezetekben tárgyaltak alapján már rendelkezésre áll egy FIB táblából felépített prefix fa, de még a tömöríthetőségre vonatkozóan nem állnak rendelkezésre adatok. Ahhoz, hogy entrópiát lehessen számolni, először még át szükséges alakítani a meglévő fát egy egyszerűbb formára. Ez az átalakítás egy *leaf-pushing* algoritmussal hajtható végre. A leaf-pushing egy preorder és egy postorder bejárást hajt végre a fán. A két bejárás kibővíti a fát úgy, hogy minden csomópont esetén, ahol csak egy levél van, oda létrehozásra kerül a másik, hiányzó levél is, továbbá az összes belső csomópontban levő érték 'letolódik' a fa leveleibe [9, 15]. A *preorder* bejárás először mindig a belső csomóponton fut le, és csak ezt követően a csomópont leszármazottain. A *postorder* pedig ennek az ellentéte: először az adott csomópont leszármazottain fut le, és csak azután magán a csomóponton. Az algoritmusok használata: először a preorderrel bejárásra kerül a fa, és kibővül új levelekkel. Ezt követően postorder bejárással törlésre kerülnek azok a levelek, amelyek azonos belső csomóponthoz tartoznak, és ez mellett a tárolt index értékeik is megegyeznek. Ezt követően pedig az index értéküket abba a csomópontba örököltetjük, amelyből ők leszármaztak. Ez

azért hasznos, mert így a táblában történő kereséskor lépések spórolhatóak meg azáltal, hogy a felesleges ismétlődések kiküszöbölésre kerültek.



2.5. ábra. FIB tábla alapján: (a) Prefix fa (b) Leaf-pushed fa (c) Egyszerűsített fa

A 2.5a ábrán, a 2.2.2 fejezetben már tárgyalt példában szereplő FIB tábla kerül elő, pontosabban az ebből felépített prefix fa. A 2.5b ábrán ugyanez a fa látható, a preorder bejárást követően. A preorder bejárással létrehozásra kerültek a hiányzó levelek[3] a fában, majd a fa belsejében lévő értékek kivétel nélkül a levelekbe kerültek. Azonosító ezután már csak a levelekben található. A 2.5b ábrát megfigyelve észrevehető, hogy a fa tovább egyszerűsíthető, hiszen azoknál a belső csomópontoknál, ahol mindkét leszármazott levélben megegyezik az azonosító, ott a leveleket törölni lehet, és a bennük levő értéket egy szinttel feljebb lehet vinni abba a belső csomópontba, amiből leszármaznak. Ez a belső csomópont a törlés után levéllé fog alakulni. Ezen rész működésének eredményét szemlélteti a 2.5c ábra.

2.2.4. Entrópia fogalma és kiszámítása

A 2.2.1 fejezet a FIB táblák tömöríthetőségét az entrópia értékével javasolta megbecsülni, viszont nem került bővebb kifejtésre, hogy ez a fogalom mit is takar. Az entrópia egy becslést ad arra, hogy adott értékhalmazból mekkora valószínűséggel érkezhet egy érték a forrástól a célhoz. A halmazban lévő minden értékhez tartozik egy valószínűség, ami azt határozza meg, hogy mekkora az esélye az adott érték előfordulásának. Ez a valószínűség egy 0 és 1 közötti szám, ahol a 0 fejezi ki a lehetetlen eseményt, az 1 pedig a biztos előfordulást. Minél kisebb eséllyel fordul elő egy érték, annál nagyobb az általa hordozott információ, és minél gyakoribb az előfordulása, annál kisebb információértékkel rendelkezik [3, 15].

Valós értékhalmazokon a különböző előfordulási valószínűségek miatt az entrópia értéke az egyes eltérő valószínűségekkel rendelkező értékek információmennyiségének összegzése lesz, amelyet súlyozni kell az értékek előfordulási gyakoriságával. Azaz, ha X az értékhalmaz, aminek x_1, x_2, ... , x_n értékei vannak, akkor ezek $p(x_1)$, $p(x_2)$, ... , $p(x_n)$ valószínűséggel fognak szerepelni a forrástól a cél felé vezető 'úton'. Ezzel a valószínűség-

[3]Hiányzó levele annak a belső csomópontnak van, amely csak egy levéllel rendelkezik.

gel súlyozzuk az elemek információmennyiségét. Ekkor a következő képletek egyike (a két képlet egyenértékű) fogja megadni a $H(X)$ entrópiát [9]:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2(p(x_i)) = \sum_{i=1}^{n} p(x_i) \log_2(1/p(x_i)) \tag{2.1}$$

Az entrópia fogalmának elméleti ismertetése után következhet ennek az alkalmazása a rendelkezésre álló FIB táblákon. Az előző fejezetek rávilágítottak arra, hogy entrópiát nem lehet a 'nyers' FIB táblákon számolni. Ehhez előtte még különböző átalakításokon, műveleteken kell átesnie a táblának. Az előzőekben már tárgyalásra kerültek ezen lépések, így csak röviden összefoglalva: első lépés a FIB táblában található hálózat címek prefix kódokká alakítása, ezt követően a prefix kódokból már felépülhet a prefix fa, végezetül pedig leaf-pushing algoritmussal egyszerűsödik a fa. Eljutva erre a formára, a létrejött fában tárolt adatokból ki lehet nyerni két darab értéket. Az egyik – legyen a neve S_α – sorfolytonosan kiolvasva fogja tartalmazni a levelekben tárolt azonosítókat, ami a 2.5.c ábrát tekintve '2545'. A másik – legyen a neve S_I – pedig sorfolytonosan fogja tárolni, hogy az adott csomópont levél-e[4] vagy sem. Esetünkben az S_I értéke '0011011'-nek felel meg. Ebből a két értékből már meg lehet határozni egy becslést arra, hogy a beolvasott forwarding táblának mennyi az entrópiája, azaz, ha tömörítjük, körülbelül mennyire fog tudni összetömörödni [9, 15].

A 2.2. ábrán látható FIB táblára számolt tapasztalati FIB entrópia 3.5 bit, ami azt jelenti, hogy egy levélben lévő adatot átlagosan ennyi biten kell tárolni. A tömörített FIB méretkorlát pedig 14 bit, azaz a FIB táblánkat ekkorára tudjuk majd összetömöríteni.

2.3. Tömörített adatstruktúrák

Az elméleti becslések helyességéről meggyőződést legegyszerűbben valós tömörítési eredmények alapján lehet kapni. A FIB táblák gyakorlatban történő tömörítése viszonyítási alapot ad az elméleti becslések alapján keletkezett eredmények helyességéről. Többféle módszerrel lehet ezeket a táblákat veszteségmentesen összetömöríteni, hogy közben ne veszítsenek a hatékonyságukból. A következőkben két féle módszer kerül bemutatásra: az egyik az S_I-t tömöríti, a másik az S_α-t. Ezek eredményeként pedig láthatóvá válik majd, hogy a valóságban egy FIB tábla mennyire tömöríthető, és eltér-e ettől az elméleti számítás eredménye.

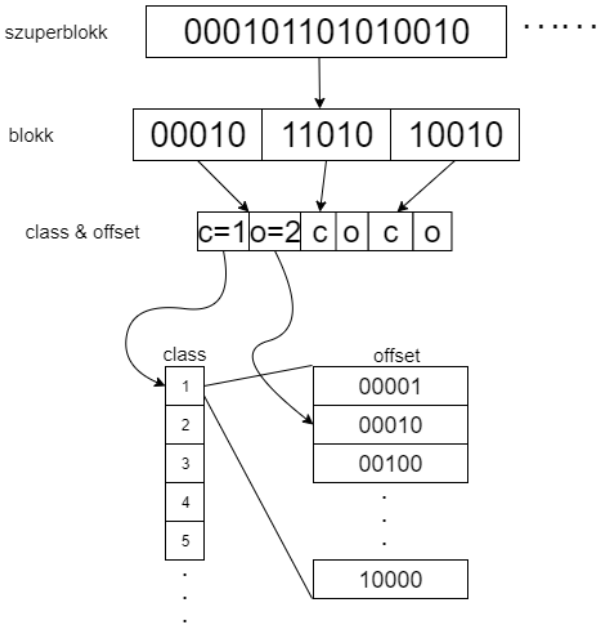
2.3.1. RRR struktúra

Az RRR struktúrák célja, hogy gyors bináris lekérést és tömöríthetőséget biztosítsanak az őket használó adathalmazok számára. Az alapgondolat a bitsorozatok olyan típusú kódolása, amely lehetővé teszi az azokon történő gyors keresést. Az RRR struktúrák $\mathcal{O}(1)$ időben tudnak lekéréseket és tömörítést végezni. A tömörített adatstuktúrákon anélkül lehet műveleteket végrehajtani (pl. keresni), hogy ehhez nem szükséges teljes mértékben kitömöríteni a fájlt [1, 5].

Az RRR a nevét a megalkotói után kapta: Raman, Raman és Rao [14]. A struktúra bitmap-ben tárolt bitsorozatokat tud tömöríteni, első lépésben blokkokra[5] és szuperblokkokra[6] osztva fel az adatot. A blokkokat olyan C osztály és O offset értékpárokkal helyettesítjük, amelyek kulcs értékként egyedileg tudják őket azonosítani a továbbiakban a táblában való keresés hatékonysága érdekében. Egy blokkhoz tartozó osztály és offset

[4] Ha a csomópont egy levél, az érték az adott csomópontra 1, másképp pedig 0 értéket vesz fel.
[5] A blokkok kis részegységei az adathalmaznak, pár bit nagyságúak.
[6] A szuperblokk több blokk által alkotott nagyobb egység.

azonosítókból egyértelműen vissza lehet kapni az adott blokkot. A 2.6 ábra egy kis példán szemlélteti az RRR struktúra szerkezeti felépítését [1, 5]. A példában a blokkok 5 bit nagyságúak és 3 blokk alkot egy szuperblokkot. Itt fontos megjegyezni, hogy a blokkokban található bitek határozzák meg azt, hogy egy-egy osztály-azonosítóhoz mekkora offset tábla tartozik. A blokkok az alapján kapják az osztály-azonosítójukat, hogy hány darab 1-es értéket tartalmaznak, az offset táblákban pedig az 1-esek adott biten történő összes előfordulása található meg. Ebből következik az, hogy amikor a blokkok sok bitet tartalmaznak, akkor az offset táblák nagyon sok permutációt kell eltároljanak, ezáltal nagyobb méretűek lesznek. Általában célszerű elkerülni a túl nagy blokkméretet, hiszen hatékonyabb, ha kisebb méretű blokkok vannak használva. Ha a példában szereplő első blokk bitjeinek felderítése a cél úgy, hogy csak a $c = 1$ és $o = 2$ kulcspár ismert, akkor a kulcspár alapján ez megtalálható: az 1-es osztály meghatározza, hogy melyik osztály azonosítóhoz tartozó offset táblát kell elővenni, a 2-es offset érték pedig megadja, hogy az offset táblában hányadik elem lesz a keresett bitsorozat, azon a pozíción pedig megtalálható a 00010 sorozat.



2.6. ábra. RRR struktúra példán szemléltetve [1]

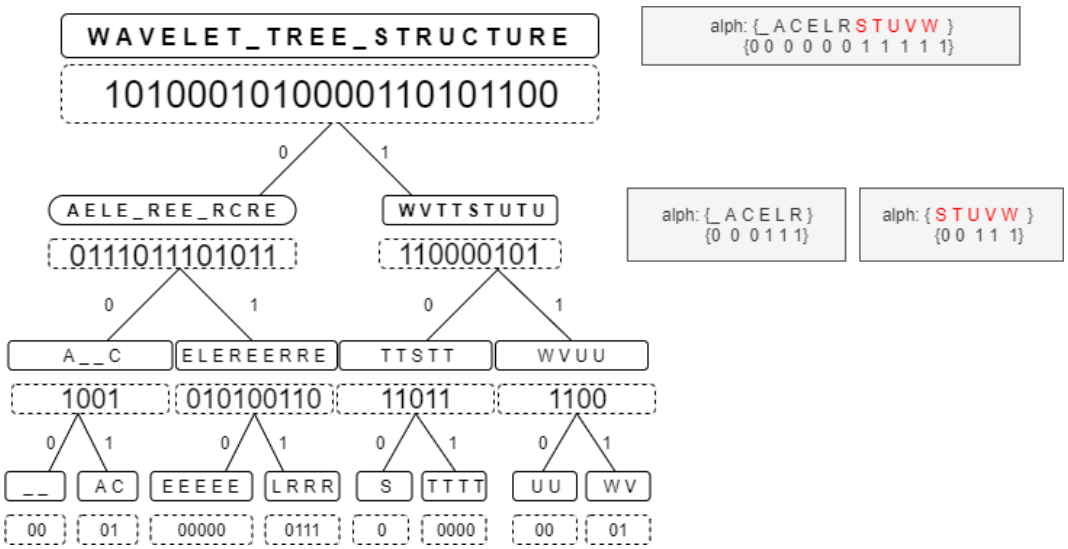
Ez egyelőre még csak a tömörítést biztosítja az elvártak szerint, a helyes működéshez azonban szükség van 3, az RRR adatstruktúrán elvégezhető műveletre is [1, 5]. Ezek közül az első a tetszőleges i. pozícióban levő bit kiolvasására szolgáló $access(i)$ művelet (véletlen elérés). Ennek megvalósítása úgy történik, hogy meghatározzuk, hogy a kiolvasandó i. pozíció melyik szuperblokk melyik blokkjába esik, az ott tárolt *class* és *offset* értékek alapján kikódoljuk a blokkot, majd visszaadjuk a blokk a lekért pozíciónak megfelelő bitjét. Ez a művelet $\mathcal{O}(1)$ időben megvalósítható, mivel nem szükséges az egész adatstruktúra, csak annak egy blokkjának a kitömörítése. Az RRR által támogatott másik fontos művelet a $rank(i)$, amely azt mondja meg, hogy az i. pozícióig hány darab 1-es volt a blokkokban. Ennek megvalósításához szükséges pár plusz adat letárolása: minden szuperblokk végén eltároljuk, hogy addig a pontig globálisan hány 1-es található a táblában, és minden blokkhoz eltároljuk, hogy a blokk kezdetéig a szuperblokkon belül hány 1-es van. Ezek után a $rank(i)$ megadása úgy történik, hogy megkeressük az i. pozícióhoz tartozó blokkot, kikódoljuk, majd összeadjuk a szuperblokk kezdetéig és a szuperblokkon belül a blokk kezdetéig előforduló 1-esek számát, és ehhez hozzáadjuk azt, hogy a blokkon belül hány

további 1-es érték szerepel az i. pozícióig. Ez szintén $\mathcal{O}(1)$ időben megvalósítható. Végül a harmadik fontos művelet az *select*, amely a *rank* inverze: a *select*(i) művelet megadja, hogy az i. 1-es érték hanyadik pozíción található. Viszonylag könnyen belátható, hogy a *rank* művelet ismételt hívásával a *select* $\mathcal{O}(\log n)$ időben megvalósítható (például bináris kereséssel), lásd [1, 5].

2.3.2. Wavelet tree

A wavelet tree struktúra abban különbözik az előbb tárgyalt RRR-től, hogy nem csupán 0 és 1 értékkel tud dolgozni, hanem sokkal szélesebb bemeneti halmazt (konkrétan sztringet) tud felhasználni bitvektorként tárolva azt. A lekérések időbeli komplexitása $\mathcal{O}(\log_2 n)$, ahol n jelöli a bemeneti értékhalmazt [2, 5].

A bemeneti adathalmazból kialakított bitvektorokat egy kiegyensúlyozott bináris fában tárolja el, ahol a 0 helyettesíti a szimbólumok egyik felét, az 1 pedig a másik felét. Ez miatt kezdetben nem lesz egyértelműen dekódolható az adat, de a fában a gyökértől egyre közelebb haladva a levelek felé, ez a kétértelműség csökken, majd teljesen feloldódik. Ennek megértésére szolgál a 2.7 ábra, amelyben a bemeneti adathalmaz a *wavelet tree structure* karaktersorozat, a 2.7 ábra jobb oldalán pedig az első két szinten a bemeneti ábécé betűi[7] láthatók abban a 0-1 felosztásban, amelyben a wavelet tree is használja az adott szinteken őket. A példa tökéletesen szemlélteti, hogy a fa leveleihez érkezve minden karakter egyedileg azonosítható, ehhez annak a feltételnek kell teljesülnie, hogy egy adott levélben legfeljebb két különböző karakter legyen. Az RRR struktúrához hasonlóan a wavelet tree is tud rank lekérést végezni a bináris fában (lásd 2.3.1) [2, 5].



2.7. ábra. Wavelet tree struktúra példán szemléltetve [2]

2.4. IP forgalomtovábbítási táblák tömörítése

A fejezet előző részei végigvezettek a FIB táblákon számolható elméleti tömöríthetőségi becslésekhez és magához a tömöríthetőséghez szükséges lépéseken. A hálózat és nexthop összerendeléseket tartalmazó FIB táblák címeiből prefix kódokat kialakítva, majd ezekből egy prefix fát felépítve és ezen algoritmusokat végigfuttatva, elméleti becsléseket lehet

[7] Az ábécé betűihez tartozik a szóköz is, ezek a típusú írásjelek a sorban a karakterek elé kerülnek.

számolni a fán, illetve tömöríteni is lehet azt. A folyamat egységeinek elméleti magyarázata a fejezet korábbi részeiben részletesen leírásra került, példákkal szemléltetve azokat.

A forgalomtovábbítási tábla tömörítéséhez elengedhetetlen a 2.2.4 fejezetben említett S_α és S_I értékek megléte. Ezeket a megfelelő formában tárolva könnyen végrehajtható a tömörítés. Az S_I-t egy RRR struktúrába alakítva, az S_α-t pedig bináris wavelet tree formába alakítva kapjuk meg annak tömörített méretét.

A számításokat elvégezve rendelkezésre áll a tömörített tábla konkrét mérete, továbba egy entrópiaérték, amely megbecsüli, hogy egy adott FIB tábla mennyire összetömöríthető. Ezen adatokat felhasználva következtetések vonhatóak le az adott FIB tábla tömöríthetőségével kapcsolatban. Megfigyelhető, hogy az entrópia által becsült érték megegyezik-e a valós tömörített értékkel, esetleg kisebb vagy nagyobb-e annál. Abban az esetben, ha eltérés lép fel, újabb kérdések merülnek fel: ha jobban/rosszabbul tömörödik a FIB tábla, mint ahogy az becsülve volt, az miért van, mi okozhatja az eltérést? A dolgozat válaszokat keres a felmerült kérdésekre a valós forgalomtovábbítási táblákra kiszámolt értékeket vizsgálva.

3. fejezet

IP forgalomtovábbítási táblák tömörítésére alkalmas rendszer tervezése és implementációja

Az előző fejezetben ismeretésre kerültek egységenként a forgalomtovábbítási táblákon végezhető becslésekhez és tömörítésekhez szükséges lépések, illetve azok elméleti háttere, működése is. Ezen fejezet az korábbi folytatásaként azt hivatott bemutatni, hogy az előzőekben felsorolt egységeket miként lehet megvalósítani, működő rendszert készítve az egyes részeket egybeépítve.

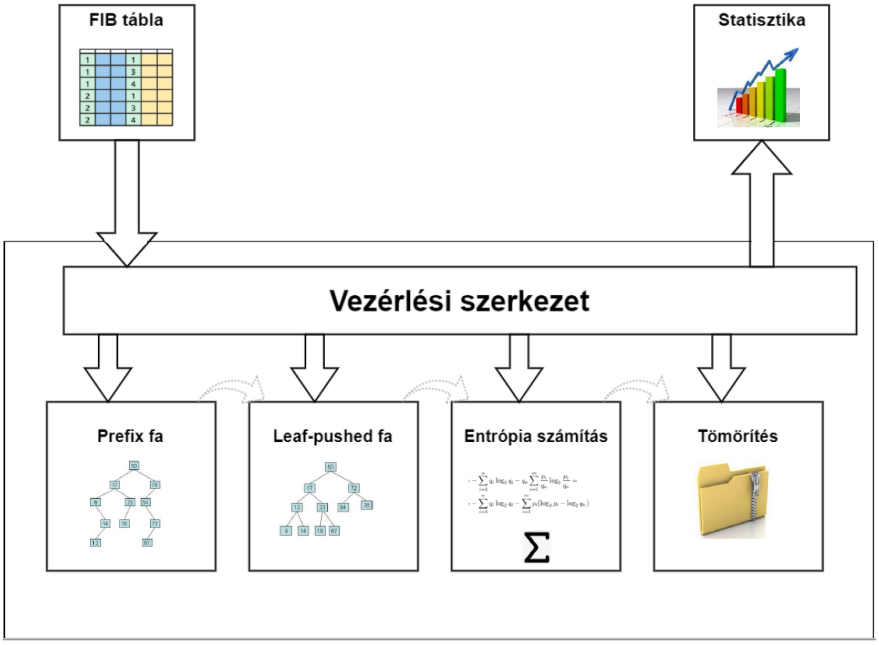
A dolgozat fő céljai közé tartozik bemutatni, hogy hogyan néz ki egy olyan saját megépített rendszer, amely bemenetként forgalomtovábbítási táblákat dolgoz fel, ezek tömöríthetőségére becslést ad, majd tömöríti is őket, hogy következtetéseket lehessen levonni, felhasználva az elméleti és gyakorlati eredményeket. Végül pedig a kapott értékek alapján statisztikákat állít fel. Ezen statisztikák célja szemléltetni a forgalomtovábbítási táblák időbeli változását, méretének és valós információtartamának növekedését, és ezek alapján megállapítani, hogy a FIB táblák milyen mértékben és irányba változtak az évek alatt, ez a változás okozhat-e problémát a jövőre nézve, előfordulhat-e, hogy összeomlik az Internet a forgalomtovábbítási táblák óriási mérete miatt.

3.1. Rendszerterv

Az egyes egységeket a 3.1 ábra mutatja be. Az ábrán végigkövethető a folyamat, amely során a FIB táblákból előállnak az elméleti és gyakorlati becslések, továbbá az ezekből készített statisztikák.

Az egész rendszert egy fő vezérlési szerkezet működteti. Ez kapja meg bemeneti forrásként a forgalomtovábbítási táblák adatait, feldolgozza azokat, és előállítja a kimenetére azon értékeket, ami alapján statisztikák készíthetőek. A 3.1 ábrán végighaladva látható részletesen bemutatva a folyamat és annak lépései. A bemenetre megérkező FIB táblát a vezérlő szerkezet első lépésben prefix fába alakítja, majd a fán algoritmusokat futtatva létrehozza a fa módosított, leaf-pushed verzióját. Ebből már következő lépésként elméleti becsléseket tud számolni a tömöríthetőségre vonatkozóan. Ezt követően pedig utolsó lépésként tömöríti a bemenetre érkezett táblát. A folyamat során a vezérlési szerkezet eltárolja az összes olyan lehetséges értéket, amely fontos lehet a statisztikák készítéséhez, illetve a következtetések megállapításához. Ezeket az adatokat kimenetként eltárolja egy fájlban, a könnyű felhasználhatóság érdekében.

A 3.1 ábrán látható nyilak segítenek annak a szemléltetésében, hogy a rendszer fő része a vezérlési szerkezet. Ez gondoskodik arról, hogy minden részegység megkapja a szá-



3.1. ábra. Vezérlési szerkezet

mára megfelelő adatot feldolgozásra, viszont arra nincs rálátása, hogy a részegységek hogyan működnek, hogyan oldják meg a feladataikat. Csupán abban segít, hogy az egységek a megfelelő sorrendben kövessék egymást, illetve összegyűjti az adatokat a statisztikák elkészítéséhez. A rendszer részegységei mind fontos szerepet töltenek be a rendszerben, egyikük se nélkülözhető, hiszen az egyik kimenete többek között a másik bemenete, és hibás működés esetén az összes egység rossz adatokkal fog dolgozni. A részegységek egy-egy nagyobb feladatrészt váltanak ki a folyamatból, és bár az ábra már nem szemlélteti, de további kisebb egységekre oszlanak fel, amelyek együttesen járulnak hozzá a részegységre osztott feladat elvégzéséhez.

3.2. Környezet és eszközök

Az implementált rendszer megvalósítása egy programkód segítségével történik. A program *Linux környezet*ben lett megvalósítva, virtuális gépen futó 64-bites, 20.04.01-es verziójú Ubuntu operációs rendszer alatt. A választott operációs rendszer segíti a hatékony munkavégzést, megkönnyítve a programkörnyezet előkészítését és a program használatát.

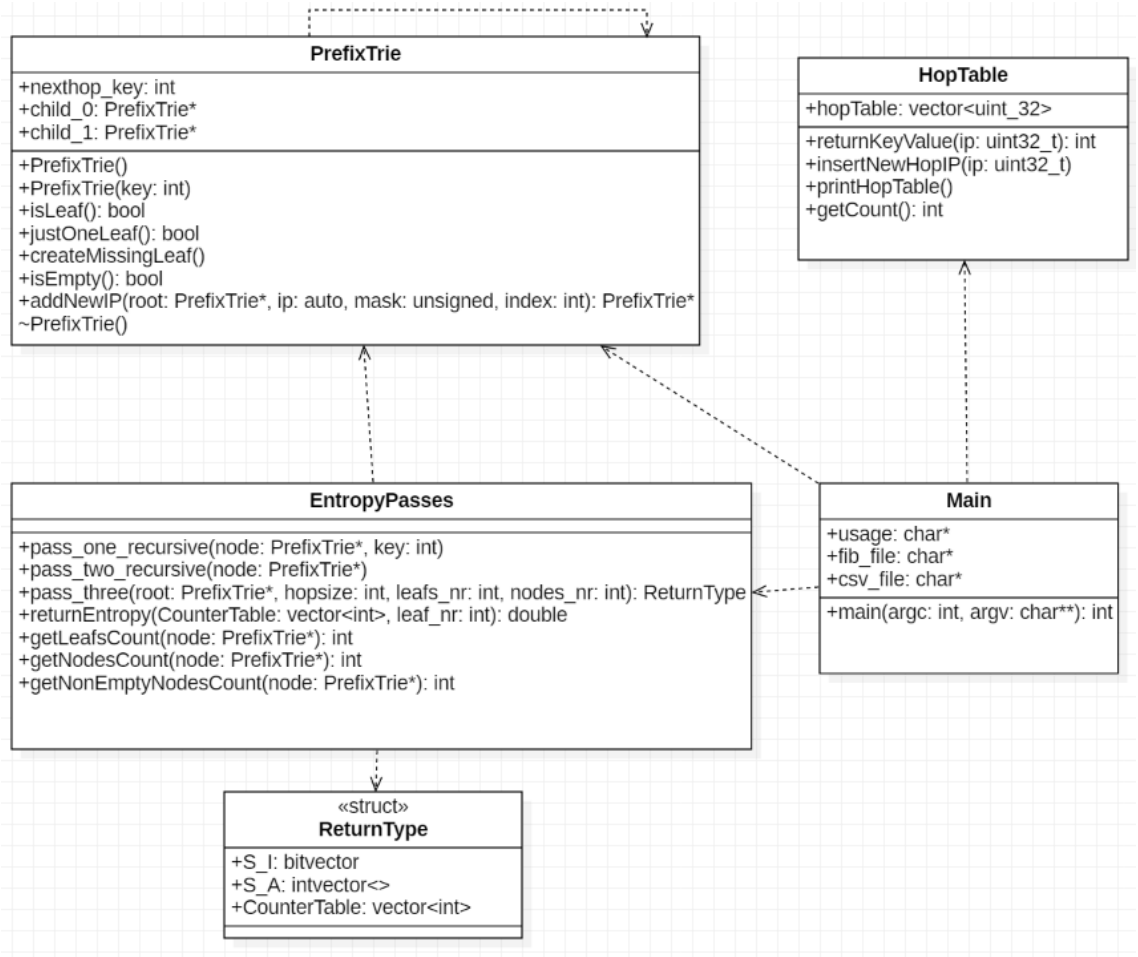
A programkód *C++11* programozási nyelven lett megvalósítva, *Code::Blocks* integrált fejlesztői környezet használva a fejlesztéshez. A program olyan speciális $C++11$ könyvtárakat használ, mint például az *SDSL - Succinct Data Structure Library*, amelyeket külön kell hozzáadni az IDE-hez. Mi is az SDSL? Az SDSL különböző tömörített adatstruktúrákat implementál komplex műveleteket megvalósítva rajtuk, bitvektorokat használva [16]. Ennek a könyvtárnak a gördülékeny, könnyű használatához nagy mértékben hozzájárult a virtuális gépen futtatott Ubuntu operációs rendszer, ugyanis Windows OS alatt rendszeres problémákba ütközött a könyvtár importálása, ellehetetlenítve a használatot.

A feladat megvalósításához szükséges részegységek külön-külön már implementálásra kerültek, az Internetet böngészve mindegyik részhez kínálkozik valamiféle megoldási lehetőség. Jelen megvalósítás viszont minden egységet saját megvalósításban implementál, az elméleti működés alapján: ide tartozik többek között olyan nagyobb része a rendszernek, mint a prefix fa építés folyamata, az azon végrehajtott algoritmusok implementálása, vagy

a tömöríthetőségi becslések kiszámítása. Könnyebb és átláthatóbb megoldás volt a teljes programkód önálló implementálása, mintha egy-egy 'idegen' részt kellett volna megfelelően beilleszteni a már meglévő működésbe.

Az implementált programkód által a kimenetre érkező adatok .csv fájlokban kerültek eltárolásra az elemzések könnyebb megvalósítása érdekében. A statisztikák elkészítéséhez a *Gnuplot* parancssorvezérelt függvényrajzoló program lett felhasználva. Használata könnyebbnek és átláthatóbbnak bizonyult a más ugyanerre a célra használt eszközökhöz képest.

3.3. Implementáció



3.2. ábra. Objektum modell

A programkód által megvalósított részek láthatóak a 3.2 ábrán. Az implementáció négy darab osztályt tartalmaz: ezek a *Main, PrefixTrie, EntropyPasses* és *HopTable* neveket kapták.

A Main osztály irányítja az egész programkód működését. Feldolgozza a bemenetére érkező táblák adatait, létrehozza az osztályok példányait, elvégzi rajtuk a megfelelő műveleteket, a kimenetre kiírja az eredményeket. A PrefixTrie osztály építi fel a prefix kódokból a prefix fát, eltárolja azt, és különböző műveleteket tud rajta végrehajtani, amelyek a fán végezhető módosításokhoz szükségesek. A HopTable osztály tárolja egy vektortömbben azokat a nexthop IP-címeket, amelyek a FIB táblából kerültek kiolvasásra. Ezen kívül ki tudja íratni az adatait és implementálja a tömb felhasználásához szükséges függvénye-

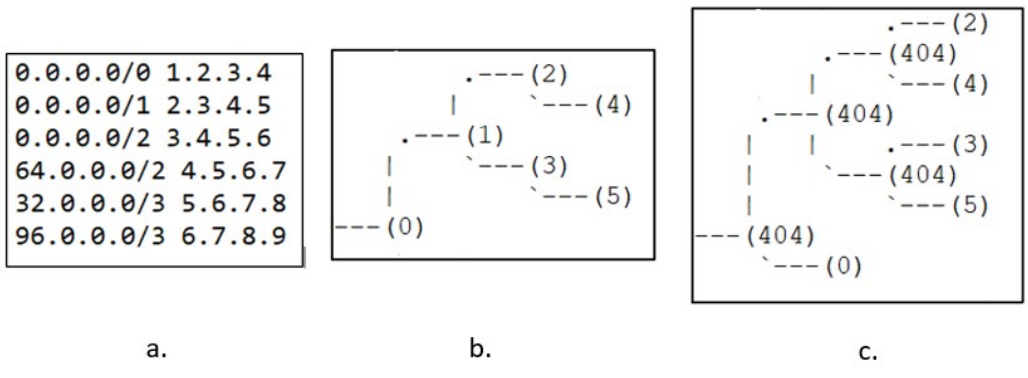
ket. Az EntropyPasses a legösszetettebb osztály az összes közül, szerepköre megosztozik a prefix fán történő módosítások végrehajtásában és az elméleti tömöríthetőség kiszámításában. Függvényei ezen szerepkörök feladatainak egységenként történő megvalósítása. Az EntropzPAsses osztály egy ReturnType struktúrát használ, annak érdekében hogy függvényében egyidőben többféle adatot lehessen visszatérési adatként megadni.

3.4. Tesztelés

Ahhoz, hogy valós adathalmazokon futtatva a programkódot, helyes számítási eredmények kerüljenek a kimenetre, ahhoz elengedhetetlen a program hibátlan működése. Ennek pedig alapkövetelménye a tesztelés, ami biztosítja, hogy a működés jól valósítja meg az adott funkcionalitásokat. Ennek tesztelhetőségében segített egyrészt a TMIT által megvalósított *The Internet Routing Entropy Monitor* weboldalon található FIB tábla adathalmaz, és az azokon végzett kiértékelések [19]. Fontos megjegyezni, hogy a fenti címen elérhető program képességei számos pontban elmaradnak a most fejlesztett rendszer képességeitől: mivel az eddigi kód magas szintű programozási nyelv használatával készült, ezért az lényegesen lassabb, és csak elméleti korlátokat számol, valós tömörített FIB méreteket nem. A jövőbeli tervek között szerepel a kód továbbfejlesztett változatának integrálása az Internet Routing Entropy Monitor rendszerbe, lecserélve annak lassú kódját a jelenlegi gyors implementációra.

A saját megvalósított program a bemenetén megkapta a weboldalon található FIB tábla adathalmazok egy részét, majd feldolgozva azokat, megadta a táblához tartozó eredményeket, amelyeket számolt. Ezen értékek összehasonlításra kerültek a weboldal által generált eredményekkel. Az eredmények tökéletes egyezése a program helyes működését hivatott alátámasztani.

Ahhoz, hogy biztosan elmondható legyen, hogy a programkód hibamentesen dolgozik, fontos a belső működés lépéseit is ellenőrizni. Ennek érdekében a programhoz tartozik egy részleges tesztelés, ami arra szolgált a kódolás során, hogy ellenőrizni lehessen a helyes működést a program minden része esetében, illetve az algoritmusok hibátlan implementálását is nyomon lehessen követni. A helyes működés először mindig kis méretű táblákon lett letesztelve. Egy ilyen saját gyártású tábla működésének tesztelése látható a 3.3 ábrán:



3.3. ábra. (a) FIB tábla (b) Beolvasott prefix fa (c) Leaf-pushed fa

A működés helyességének ellenőrzését szolgálja a fa strukturált kiíratása, hiszen ezáltal jól láthatóvá válik az összes, a fán végzett művelet. A 3.3.a ábra szemlélteti, hogy a beolvasott FIB tábla milyen adatokat tartalmaz. Ezt tekintve látható, hogy helyesen van

felépítve a 3.3.b ábrán látható prefix fa, továbbá a 3.3.c ábrán látható módosított fa[1] is. A 3.3. ábra pillanatképei között sok fontos lépés és algoritmus van jelen, amelyek nem láthatóak külön-külön jelenleg az ábrákon, de ezek helyes működése is tesztelésre került. Az első ilyen áttekintés akkor történt, amikor ezek a lépések új elemként implementálva lettek a programkódba, utána pedig folyamatosan akkor, amikor újabb elemekkel bővült a kód, figyelve ezáltal, hogy az új elemektől nem fog-e elkezdeni hibásan működni az addig jól funkcionáló program. A folyamatos hibátlan működés pedig az előbbiekben már említett tanszéki eredményekkel összevetve vált biztossá.

[1] A fa belső csomópontjai üresek, ezek ilyenkor `-1` értéket vesznek fel, de ez a kiíratáskor nehezítené az átláthatóságot, ezért itt ezek a belső csomópontok `404`-es értékkel kerültek kiíratásra; természetesen ez bármilyen más véletlenszerű szám is lehetne.

4. fejezet

Statisztikák kiértékelése

Ezen fejezet célja, hogy az elkészített rendszer által végzett számításokat feldolgozva statisztikákat készítsen a bemeneti adathalmazokról, illetve, hogy kiértékelje a kapott eredményeket. Ahhoz, hogy széles látószögben jellemezhetőek legyenek a forgalomtovábbítási táblák, elemzésre kerül az idő elteltével mutatott tömöríthetőségre vonatkozó változásuk, ami magába foglalja a méreteik növekedését, illetve elméleti és gyakorlati tömöríthetőségi jellemzőiket.

4.1. Felhasznált adathalmaz

Az elemzések valós, működő, a globális Internet útvonalválasztási rendszerében aktívan résztvevő útválasztó eszközökről letöltött forgalomtovábbítási táblákon kerülnek kiértékelésre. Ezeket a táblákat a HBONE, a magyar akadémiai gerinchálózat üzemeltetői bocsátották rendelkezésre a BME-TMIT és a HBONE együttműködése keretében [19].

A forgalomtovábbítási táblák a HBONE gerinchálózatának tíz legnagyobb forgalmú útválasztó eszközéről származnak. A táblák letöltése minden nap éjfélkor történik (a valós útválasztók terhelésének minimalizálása érdekében) az SNMP protokoll segítségével. Az adatgyűjtés kezdete 2013. A táblák tartalmazzák az Internet globális útválasztási rendszerébe a Border Gateway Protocol (BGP) segítségével meghirdetett összes alhálózati prefixet, továbbá több ezer "belső", a HBONE hálózatán belüli célpontokhoz tartozó útvonalat is. Ekkora mennyiségű adat esetén széleskörűen elemezhető a táblák méretének, információtartalmának változása. A könnyebb átláthatóság kedvéért ezek a változások diagramok segítségével kerülnek ábrázolásra.

Az alábbi elemzés céljából a teljes adathalmazból négy útválasztót emeltünk ki, amelyek jól reprezentálják a teljes adathalmazt: a $hbone-vh1$ és $hbone-vh2$ a HBONE Victor Hugo utcai adatközpontjában elhelyezkedő "központi" útválasztók, a $hbone-bme$ a Budapesti Műszaki Egyetem és a HBONE határponti átjárója, míg a $hbone-szeged$ a Szegedi Tudományegyetem és a HBONE határponti átjárója. A napi statisztikákat leszűrtük oly módon, hogy az adatgyűjtés kezdete óta minden hétből tartalmazzon az adathalmazunk egy-egy táblát. Végül az összes forgalomtovábbítási táblából eltávolítottuk az "alapértelmezett átjárót" (default gateway), mert kezdeti elemzéseink azt mutatták, hogy ezek jelentősen torzítják a statisztikákat.

4.2. Elméleti és gyakorlati tömöríthetőségi jellemzők

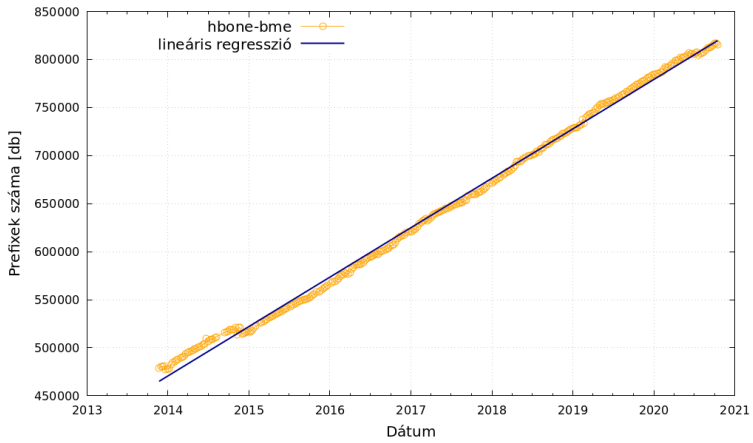
Az elkészített rendszer által feldolgozott táblák nagy mennyiségű adatot szolgáltatnak, amelyek felhasználhatóak statisztikák készítéséhez. A 4.2.1 fejezetben a még tömörítetlen táblák méreteinek növekedése kerül kielemzésre, hogy az évek során milyen változásokon

mentek keresztül ezek az értékek. Ezt követően a 4.2.2 fejezet a tapasztali FIB entrópia időbeli változását értékeli ki, a 4.2.3 fejezet az elméleti tömörítési korlát és a tömörített méret évek során alakult egymáshoz való viszonyulását vizsgálja, legvégül pedig a 4.2.4 fejezet a FIB táblák valós és tömörített méretének hányadosait elemzi.

4.2.1. Tömörítetlen táblák elemzése

A négy router hét évnyi FIB táblái fejenként több mint 300 darab táblát jelentenek, amelyekre az implementált programkód lefutott. Először a tömörítetlen táblákon végzett számítási eredmények kerülnek feldolgozásra. Ide tartozik az előző fejezetekben tárgyalásra került rendszer részegységei közül a prefix fába alakított FIB tábla (lásd 2.2.2). Ennek megépítése után rendelkezésre áll az az információ, hogy az adott FIB táblában hány különböző darab hálózat cím[1], illetve nexthop azonosító található.

Először elemezzük a forgalomtovábbítási táblák prefixeinek az évek során történt változását. A négy router tábláira kapott eredményeket szemléltetik a 4.1, 4.2, 4.3 és 4.4 ábrák. Mind a négy diagram esetében az x tengely szemlélteti a dátumot, az y tengely pedig a különböző prefixek számát. Az összes ábrán szembetűnő a felismerés, miszerint a hét évvel ezelőtti táblákban található prefixek száma szinte megduplázódott a legutolsó elemzett táblák prefix számaihoz képest. Ez azt jelenti, hogy a jelenlegi FIB táblák közel négyszázezerrel több hálózati címet tartalmaznak, mint hét évvel ezelőtt.

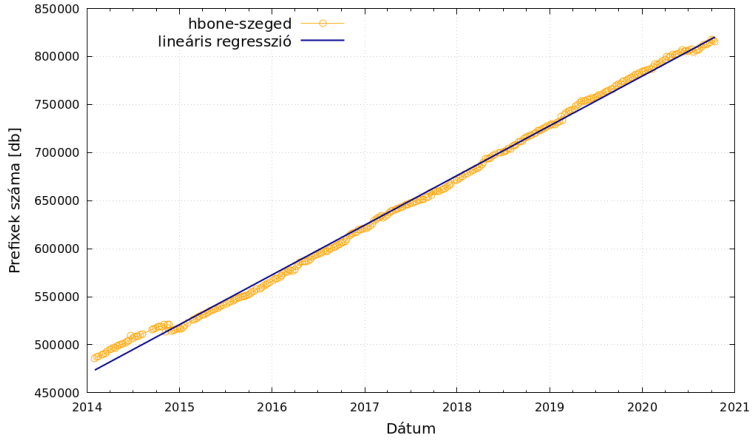


4.1. ábra. HBONE-BME prefix értékeinek változása (a lineáris regresszió alapján a változás üteme 140,3 darab/nap)

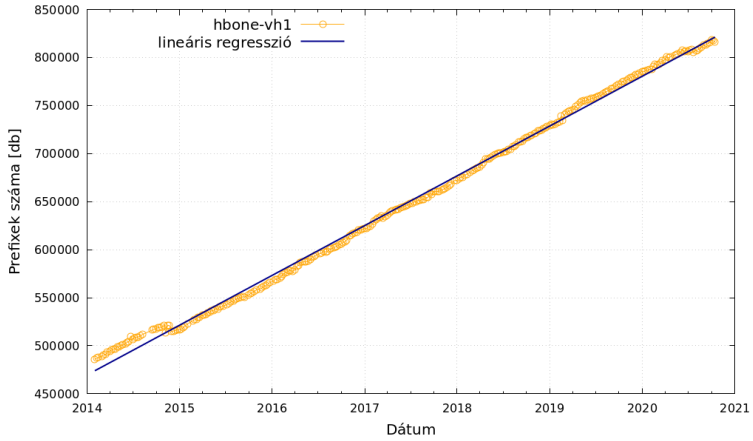
A prefix értékek növekedését szemléltető ábrákon megtalálható még továbbá egy egyenesekre illesztett lineáris regresszió. Ezen regressziók az y tengelyen található prefix számok változékonyságát hivatottak magyarázni az idő elteltével. A regresszió az adatokhoz legjobban illeszkedő egyenest adja meg [18]. Tökéletes egyezés esetén a regresszió által megadott egyenes teljes mértékben a prefix értékeket megadó pontokra illeszkedne. Azokon a helyeken ahol eltérés található, az az ún. regresszió hibája [7]. A diagramok prefix értékeinek időbeli változására számolt regresszió megadja, hogy naponta hány darab prefix értékkel növekedett a FIB táblák tartalma. Ez az érték mind a négy tábla esetében 140-es nagyságrend körül mozog, a pontos értékek az ábrák mellett vannak feltüntetve minden tábla esetében.

A tömörítetlen táblák elemzéséhez kapcsolódik a nexthopok számának időbeli alakulása is. Hasonlóan az előző ábrákhoz, a 4.5 ábrán is az x tengely szemlélteti az időt,

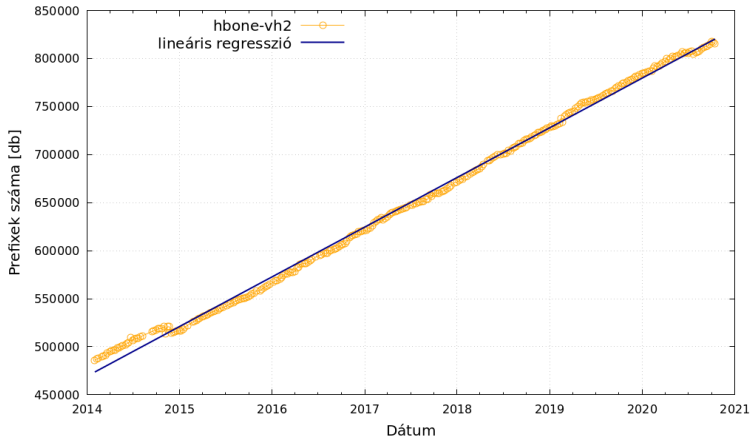
[1] A különböző hálózati címek számát a fában található címkével ellátott csomópontok száma fogja megadni, ez a prefixek száma.



4.2. ábra. HBONE-SZEGED prefix értékeinek változása (a lineáris regresszió alapján a változás üteme 141,5 darab/nap)



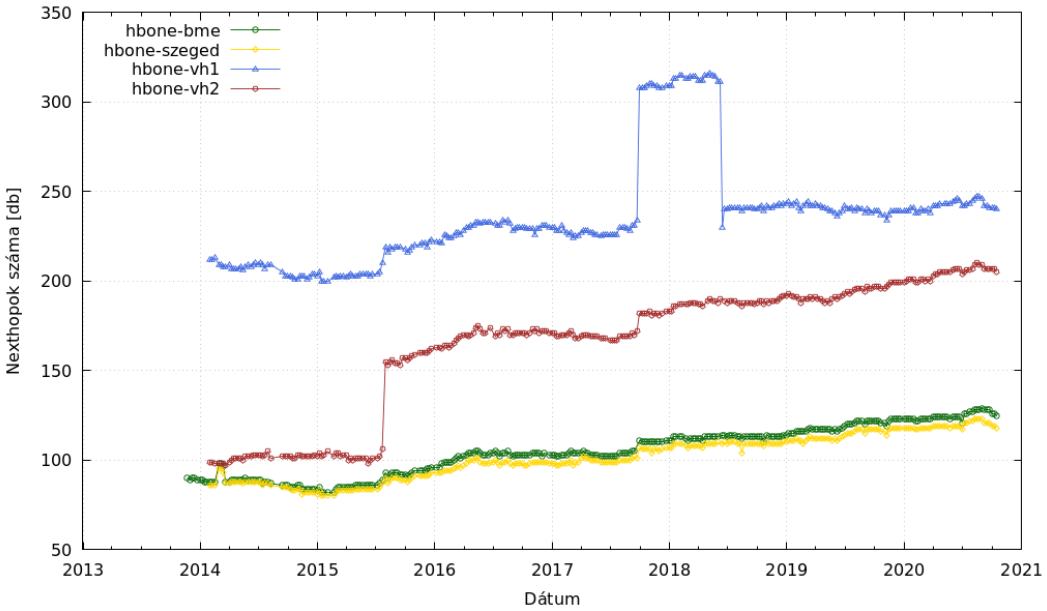
4.3. ábra. HBONE-VH1 prefix értékeinek változása (a lineáris regresszió alapján a változás üteme 140,9 darab/nap)



4.4. ábra. HBONE-VH2 prefix értékeinek változása (a lineáris regresszió alapján a változás üteme 143,9 darab/nap)

az y tengely pedig a nextop címek darabszámát. A nexthopok időbeli változása látható a 4.5 ábrán, ahol a négy routerből kinyert adatok összesítve találhatóak meg. A nexthop

címek azt adják meg, hogy adott csomagot a célcím felé milyen következő címre kell küldeni (részletesebben lásd: 2.1). Ezek esetében nem annyira jelentős a növekedés, mint a prefixeknél, de azért látható az ábrán hogy az idő múlásával ezek is fokozatosan bővülnek. A 4.5 ábrát tekintve észrevehető, hogy a *hbone-vh1* a 2018-as év környékén kiugrást mutat, az észrevételek jelzésre kerültek a HBONE felé.



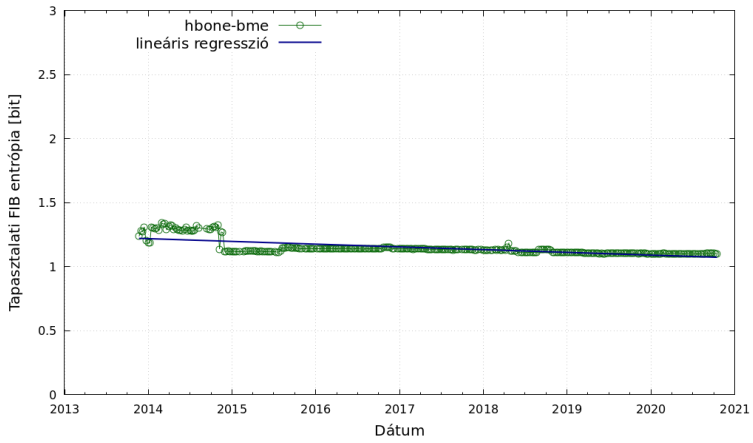
4.5. ábra. A nexthop címek számának alakulása az idő függvényében

Összefoglalva az előbbiekben tárgyaltakat: a tömörítetlen FIB táblák méretei az elmúlt hét év során erőteljes növekedésen mentek keresztül. Erről tanúskodik a prefix számok és nexthop értékek emelkedése. Amint látható, a forgalomtovábbítási táblákban található prefixek száma közel a duplájára emelkedett. Ez az évek során megvalósult nagyméretű növekedés a lineáris regresszió alapján azt mutatja, hogy ez az emelkedés várhatóan a jövőben sem fog megállni, ehelyett fel kell készüli arra, hogy naponta körülbelül 140, vagyis heti körülbelül 1000 új bejegyzés fog megjelenni az Internet forgalomtovábbítási tábláiban.

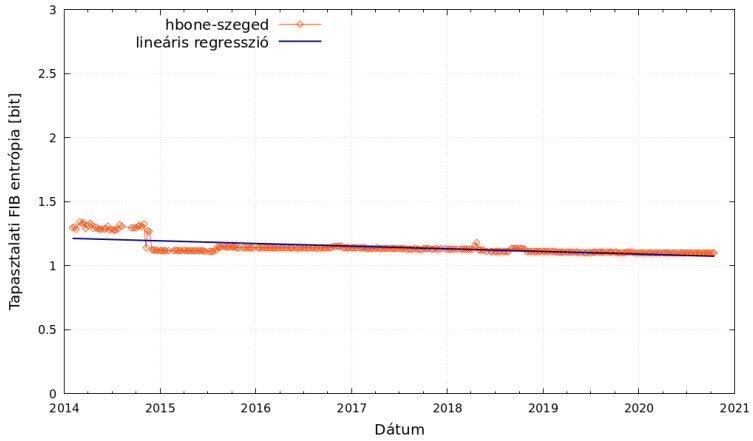
4.2.2. Elméleti tömöríthetőség

A forgalomtovábbítási táblákból előállított prefix fák, a rajtuk végrehajtott módosítások után, elérhetővé teszik azokat az adatokat amelyek az entrópiaszámításhoz szükségesek. A tapasztalai FIB entrópia megadja, hogy az adott fa egy csomópontjában található adatot átlagosan hány biten kell eltárolni a leghatékonyabb eredmény elérése érdekében (részletesebben lásd a 2.2.4 fejezetben).

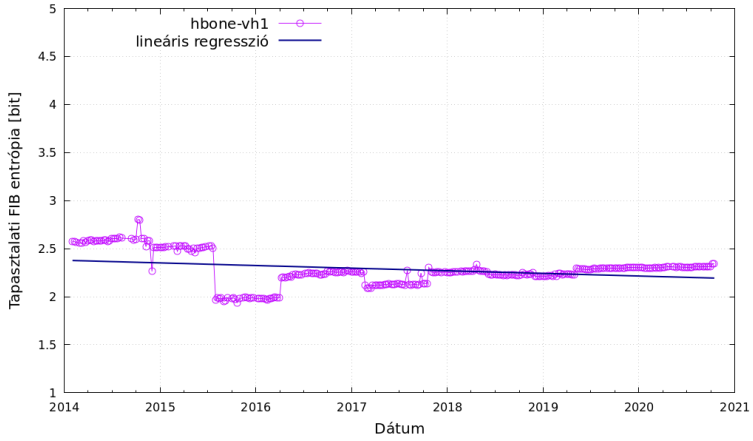
A négy adathalmazon végzett számítások eredményeként az látható a 4.6, 4.7, 4.8 és 4.9 ábrákon, ahol az x tengely szemlélteti az időt, az y tengely pedig a tapasztalati FIB entrópia értékét méri, a mértékegysége bit. Megfigyelve az ábrákat, látható, hogy a tapasztalati FIB entrópia értéke a hét év alatt nem mutat jelentős változást, végig közel konstans. A *hbone-vh1* kivételével a változások fél biten belül mozognak, és ez a változás a *hbone-vh1* esetében se haladja meg az 1 bitnyi mozgástartományt. A tapasztalati FIB entrópiát szolgáltató ábrákra is lineáris regresszió lett illesztve. Ez is megerősíti a felismerést, hogy milyen csekély mértékben változott az évek során jelentős mértékben növekvő táblák entrópiája.



4.6. ábra. HBONE-BME tapasztalati entrópiájának változása (a lin. regresszió alapján a változás üteme -10^{-4} bit/nap)

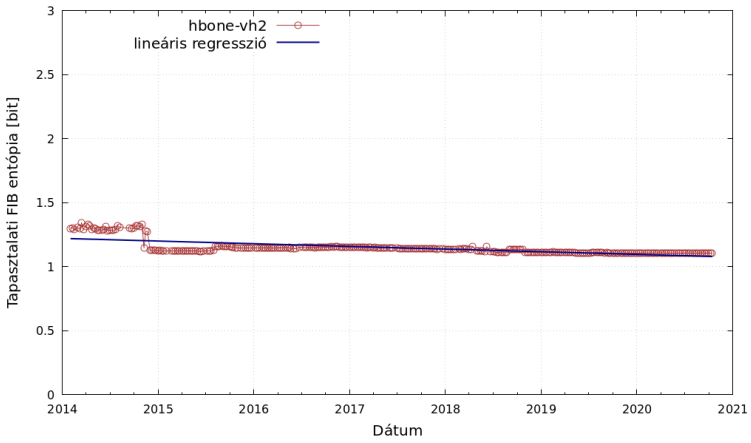


4.7. ábra. HBONE-SZEGED tapasztalati entrópiájának változása (a lin. regresszió alapján a változás üteme -10^{-4} bit/nap)



4.8. ábra. HBONE-VH1 tapasztalati entrópiájának változása (a lin. regresszió alapján a változás üteme -10^{-4} bit/nap)

A kis tartományon belüli változás arra enged következtetni, hogy a FIB táblák komplexitása nem növekszik jelentős mértékben, kifejezetten szűk határon belül mozog, nem



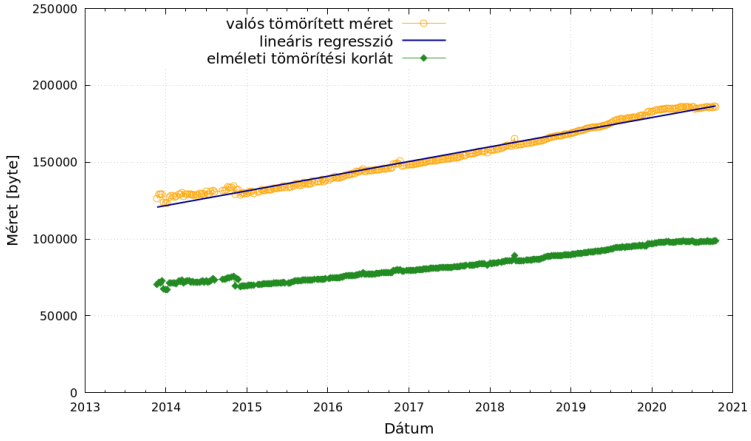
4.9. ábra. HBONE-VH2 tapasztalati entrópiájának változása (a lin. regresszió alapján a változás üteme -10^{-4} bit/nap)

tartja a lépést a 4.2.1 fejezetben látott prefixek rohamos növekedésével. A hét évre visszamenőleg elemzett adat viszonylag konstans entrópia értéke azt jelzi, hogy a jövőbeli értékek jól becsülhetők a már meglévőekből.

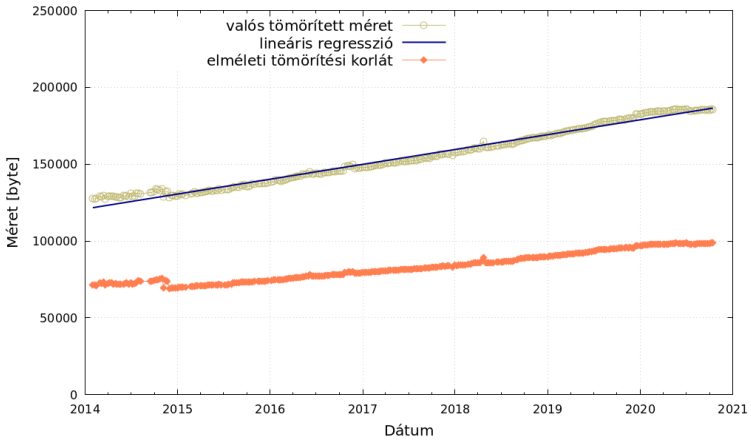
4.2.3. Elméleti tömörítési- és gyakorlati tömörített méretkorlát

A forgalomtovábbítási táblákon végzett számítások arra is rálátást nyújtanak, hogy ezen táblák elméleti tömöríthetőségi értéke milyen viszonyban van a táblák gyakorlati tömörített méretkorlátjával. Ezen fejezet ezeket az értékeket hivatott elemezni. Az ábrák együttesen tartalmazzák a két méretkorlátot mind a négy router esetében, továbbá a valós tömörített méretre lineáris regresszió is illesztve van.

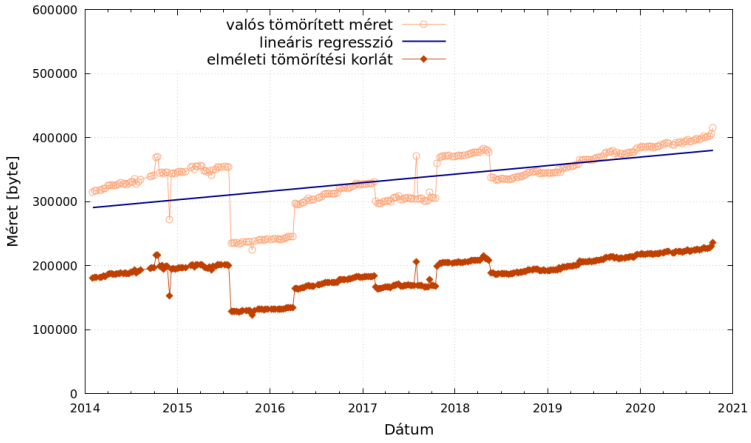
Emlékeztetőül: a valós tömörített méret a prefix fákból kinyert S_I és S_α értékek tömörítése után ezek méretének az összege. A két érték az ismertetett RRR és wavelet tree struktúrával lett tömörítve, ennek részleteit lásd a 2.3 fejezetben. A másik vizsgált érték pedig az elméleti tömörítési korlát. Ez azt adja meg, hogy a forgalomtovábbítási tábla az entrópia alapján mekkorára tömöríthető. Ezt az egy csomópontra eső tapasztalati FIB entrópia értékének és a fában található összes csomópont számának szorzata adja meg.



4.10. ábra. HBONE-BME elméleti és gyakorlati tömöríthetőség változása (a lineáris regresszió alapján a valós tömörített méret változásának üteme 26.11 byte/nap)



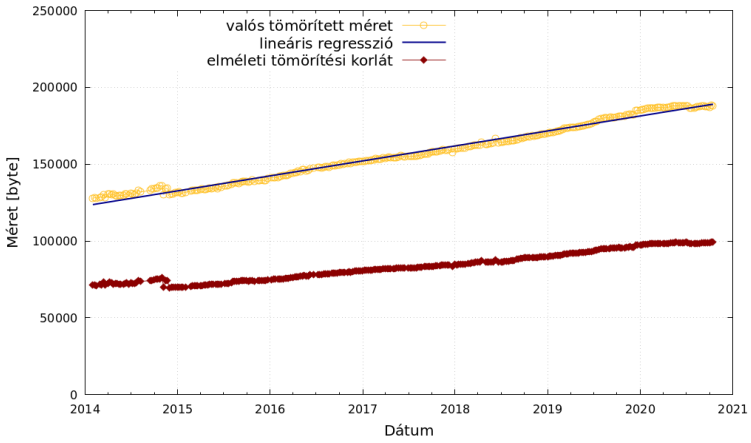
4.11. ábra. HBONE-SZEGED elméleti és gyakorlati tömöríthetőség változása (a lineáris regresszió alapján a valós tömörített méret változásának üteme 26.45 byte/nap)



4.12. ábra. HBONE-VH1 elméleti és gyakorlati tömöríthetőség változása (a lineáris regresszió alapján a valós tömörített méret változásának üteme 36.46 byte/nap)

A 4.10, 4.11, 4.12 és 4.13 ábrák szemléltetik a négy kiválasztott router forgalomtovábbítási tábláinak hét évre visszakövetett elmeleti és gyakorlati tömöríthetőségi értékét. Az ábrákon az x tengely az időt szemlélteti, az y tengely pedig a táblák méretét byte-ban mérve. A valós tömörített méretekre illesztett lineáris regresszió azt mutatja, hogy mennyivel változik naponta a táblák tömörített méretének nagysága. Ez az érték azt mutatja, hogy körülbelül, a *hbone-vh1* kivételével, 26 byte-tal emelkedik naponta a táblák tömörített mérete, a *hbone-vh1* esetében pedig kb. 36 byte-tal. A vh1 router tábláira számolt értékek viselkedése eltér a másik három tábla eredményeitől. A néhol inkonzisztens viselkedés miértjeivel kapcsolatban konzultálások folynak a HBONE-nal, közösen válaszokat keresve a kérdésre.

Az ábrákat megfigyelve észrevehető, hogy a valós és az elméleti tömöríhetőségi értékek szinte pontosan követik egymást. Az elméleti becslések a tömöríthetőségre mind a négy esetben kisebb értéket becsültek, mint amennyire a táblákat tömöríteni lehetett. A valós tömörített méretkorlátok körülbelül 1.5-2-szer több byte-on tömörödnek, mint ahogy azt az elméleti becslések megadják. Ennek persze az is oka lehet, hogy az RRR és a wavelet tree struktúra nem tömörít elég hatékonyan. A jövőben tervezve van más, hatékonyabb



4.13. ábra. HBONE-VH2 elméleti és gyakorlati tömöríthetőség változása (a lineáris regresszió alapján a valós tömörített méret változásának üteme 26.67 byte/nap)

struktúrákkal is megvizsgálni, hogy a táblák mennyire tömöríthetőek. A *Succinct Data Structure* könyvtár még felkínál különböző lehetőségeket a táblák tömörítésére [17], amelyek lehetséges, hogy akár hatékonyabb tömörítést is tudnak végezni.

4.2.4. Tömörítés hatékonysága

A kiválasztott routerek hét év alatt összegyűjtött forgalomtovábbítási tábláiból származó adatok az eddigiekben már elemzésre kerültek különböző szempontok alapján, a táblákból számolt értékek megfelelő részét felhasználva. Az utolsó szempont, ami alapján az adatok elemzésre kerülnek, a tömörítés hatékonyságának vizsgálata.

FIB tábla	Tömörítési ráta
hbone-bme	127.96
hbone-szeged	128.63
hbone-vh1	102.35
hbone-vh2	219.59

4.1. táblázat. Tömörítési ráta: tömörítetlen méret/valós tömörített méret

Mind a négy útválasztó eszköz esetében rendelkezésre áll a forgalomtovábbítási táblája valós tömörített és tömörítetlen mérete. A 4.1 táblázatban láthatóak ezeknek a tömörített és tömörítetlen méreteknek a hányadosaira számolt átlagok, a hét évre vonatkozóan. Mind a négy tábla esetében látható, hogy a tábla tömörítetlen mérete átlagosan több, mint százszor nagyobb a valós tömörített méret nagyságánál. A hbone-vh2 esetében pedig a kétszázat is meghaladja ez az érték. A 4.1 táblázat által szolgáltatott tömörítési ráta értékek nagyon hatékony tömöríthetőségről tesznek tanúbizonyságot, a kapott eredmények alapján az látható, hogy a választott tömörítési módszer nagyon jól tömörít, hiszen a tömörített méret több, mint százszor kisebb az eredetinél. Ezek az eredmények kedvezőek a táblák tömörítésére vonatkozóan, hiszen a számítások azt mutatják, hogy a hatékony tömörítés által jelentős memóriát takaríthatunk meg az útválasztó eszközök forgalomtovábbítási táblái számára.

Összefoglalva: megállapíthatjuk, hogy annak ellenére, hogy a valós forgalomtovábbítási táblák az évek során jelentős növekedésen estek át, ezt a növekedést a tömörítési algoritmusaink képesek jelentős mértékben „elfedni” az útválasztók elől. Ez egyrészt kö-

szönhető annak a meglepő felfedezésnek, hogy a forgalomtovábbítási táblák entrópiája az évek során kürülbelül konstans volt, illetve annak, hogy a tömörítés valamilyen okból váratlanul hatékony a forgalomtovábbítási táblákon. Ezeknek az okoknak a felderítése későbbi munkánk egyik legfontosabb irányát fogja képezni.

5. fejezet

Konklúzió

5.1. A munka értékelése

A dolgozat megvalósulását egy évnyi munka előzte meg, amely egy év első részében a szakterületben elmélyüléshez szükséges információgyűjtés kapott fő szerepet, továbbá implementálásra került a programkód első fele is, amely már képes volt becsléseket adni az elméleti tömöríthetőségre. A munkafolyamat második felének is fontos része volt az információgyűjtés, hiszen tanulmányozni kellett többek között azokat a tömörítési struktúrákat, amelyek felhasználásra kerültek, ezen kívül a programkód második felének leimplementálása is ekkor valósult meg.

Jelenleg rendelkezésre áll egy hatékony, gyors működést eredményező programkód, amely alkalmas FIB táblák tartalmának feldolgozására, szolgáltatva ezáltal rengeteg információt a táblák adataira vonatkozóan. Ilyen meghatározó információk többek között az elméleti tömörítési becslések, a gyakorlati tömöríthetőségi méretek, a tábla karakterisztikája. Azáltal, hogy ezen információk rendelkezésre állnak a program lefutását követően, lehetőség nyílt elemzések és becslések készítésére. Természetesen a valós értékekkel dolgozáshoz szükséges volt valós forgalomtovábbítási értékekre, amelyeket a munka elvégzéséhez a HBONE bocsátott felhasználásra. A hét évnyi adathalmaz, amelyen a program számításokat végzett, elegendő volt egy széles látószög kialakításához arra vonatkozóan, hogy a forgalomtovábbítási táblák hogyak változtak, illetve változnak az évek alatt..

Fontos megjegyezni, hogy az egy évnyi munkának a dolgozat nem lezárása, csupán egy pillanatképe. A forgalomtovábbítási táblák tömörítésére vonatkozó implementációk kipróbálása és az ezáltal kapott eredmények elemzése a dolgozat elkészítését követően is folytatódni fog, ezáltal tovább bővítve a témáról szerzett ismereteket.

5.2. Kitekintés

A hét évnyi forgalomtovábbítási tábla adatokon végzett statisztikák alátámasztják az állításokat, miszerint a FIB táblák mérete rohamos növekedésben van, és ennek a növekedésnek a következtében az útválasztási folyamatok komoly skálázási problémákkal kerülnek szembe [11, 20]. Egy megtörtént ilyen eset volt az '512k day' néven elhíresült esemény, amikor 2014-ben jópár útválasztó eszköz elérte az 512 ezer soros memóriakorlátot, aminek hatására leállt az Internet egy egész napra jelentős nagyságú területen [8]. Látható tehát, hogy a skálázhatóság komoly problémákat okozhat az Internet világában. A felismerés azzal kapcsolatban, hogy ez a probléma valós, az *IAB - Internet Architecture Board* bizottság keretein belül is megfogalmazódott. 2007-ben workshop-ot tartottak az Internet skálázhatósági problémáiról amelyről külön RFC4984 is született [13].

A dolgozatban elvégzett számítások ugyanakkor azt is bizonyítják, hogy a FIB táblák információtartalma az évek során nem követte táblák méretének rohamos növekedését, hatékonyan lehet tömöríteni őket. A tömörítés pedig nagyrészben megoldást kínálhat a skálázhatósági problémák okozta gondokra. A statisztikák azt mutatják, hogy a táblákban fellépő növekedések lineárisak, a lineáris regresszió segítségével becsléseket lehet adni a jövőre nézve, hogy hogyan fog alakulni a táblák méretnövekedése, ezáltal látva, hogy körülbelül mikor érik el a routerek újra a méretkorlátaik határát.

Számításaink azt mutatják, hogy az útválasztók forgalomtovábbítási tábláinak mérete évente tömörítetlen formában körülbelül 50 ezer új prefixszel, tömörített formában pedig körülbelül 10 Kbyte-tal nő. Ezek alapján a forgalomtovábbítási tábák mérete várhatóan 2023-ban átlépi majd az 1 milliót, amely akár újabb jelentős kieséseket okozhat majd a globális Internet szolgáltatásban, hiszen sok régebbi útválasztó memóriája 1M prefix tárolására képes csak ('1024k day'). Mindeközben a tömörített méret még mindig 250 KByte alatt marad majd, ami könnyen belefér akár a legrégebbi útválasztó eszközök gyors memóriájába is. Az eredményeink alapján levonhatjuk a következtetést, hogy az útválasztó táblák ebben a dolgozatban bemutatott tömörítése akár fontos eszközzé is válhat a jövőben az Internet útválasztás skálázhatósági problémáinak megoldásában.

5.3. Jövőbeli tervek

A korábban említettek megerősítéseként, jelen dolgozat egy pillanatképe a munka folyamatának. Számos további kitűzött cél vár megvalósításra, amelyek minél szélesebb körű rálátást adhatnak a forgalomtovábbítási táblák világára.

A tervek között szerepel további, sokkal nagyobb méretű adathalmazokra lefuttatni az elkészített programkódot, hogy mégtöbb információ álljon rendelkezésre ezek viselkedéséről. Továbbá a programkód további funkciókkal ellátása is fontos részét képezi a jövőbeli terveknek. Ilyen funkció lenne a keresések és a táblában történő módosítások időigényének és működésének megfigyelése. Ezen kívül cél a lineáris regressziónál magasabb fokszámú interpoláció illesztése a számításokra, hogy az azáltal adott becslések összehasonlíthatóak legyenek a jelenlegi becslésekkel, pontosabb számítási eredmények meghatározása érdekében.

Fontos kiemelnünk még, hogy a dolgozatban csak az IPv4 Internet útválasztó rendszerét vizsgáltuk, mivel jelenleg még mindig az IPv4 a meghatározó protokoll verzió az Interneten. Célunk az adatgyűjtést kiterjeszteni az IPv6 FIBek gyűjtésére is és az elemzéseinket kibővíteni IPv6 címekre is.

Irodalomjegyzék

[1] Alex Bowe: RRR – a succinct rank/select index for bit vectors. https://alexbowe.com/rrr/.

[2] Alex Bowe: Wavelet trees – an introduction. https://alexbowe.com/wavelet-trees/.

[3] Komenczi Bertalan: *Információelmélet*. 4.3.12. köt. 2011.

[4] BME: Mérnöki modellalkotás - az elmélettől a gyakorlatig. VITMMA003.

[5] Alex Bowe: Multiary wavelet trees in practice, 2010.

[6] Douglas Corner: *Computer Networks and Internets*. 2008, Prentice Hall.

[7] Csallner András Erik: Bevezetés az SPSS statisztikai programcsomag használatába. http://www.jgypk.hu/tamop15e/tananyag_html/spss/regresszi.html.

[8] Digital Wolf: 512k day. what happened to internet. https://medium.com/@digitalmindwolf/512k-day-what-happened-to-internet-3daaf44b138e.

[9] Rétvári Gábor: Hálózati problémák interdiszciplináris megközelítésben, 2020.

[10] Craig Hunt: *TCP/IP Network Administration*. 2002, OReilly.

[11] Varun Khare – Dan Jen – Xin Zhao – Yaoqing Liu – Dan Massey – Lan Wang – Beichuan Zhang – Lixia Zhang: Evolution towards global routing scalability. *IEEE JSAC*, 28. évf. (2010) 8. sz., 1363–1375. p. 13 p.

[12] Buttyán Levente – Györfi László – Győri S. Sándor – Vajda István: *Kódolástechnika*. 2006.

[13] D. Meyer – L. Zhang – K. Fall: Report from the IAB Workshop on Routing and Addressing. RFC 4984, 2007.

[14] Sattisrinivasa Rao Rajeev Raman, Venkatesh Raman: Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. https://dl.acm.org/doi/10.5555/545381.545411.

[15] G. Rétvári – J. Tapolcai – A. Kőrösi – A. Majdán – Z. Heszberger: Compressing IP forwarding tables: Towards entropy bounds and beyond. In *ACM SIGCOMM* (konferenciaanyag). 2013, 111–122. p. 12 p.

[16] Simon Gog: SDSL - succict data structure library. https://github.com/simongog/sdsl-lite.

[17] Simon Gog: Succint data structure library. https://simongog.github.io/assets/data/sdsl-slides/tutorial#1.

[18] SPSSABC: Lineáris regresszió. https://spssabc.hu/tobbvaltozos-elemzes/regresszio/linearis-regresszio/.

[19] The internet routing entropy monitor. http://lendulet.tmit.bme.hu/fib_comp/.

[20] Xiaoliang Zhao – Dante J. Pacella – Jason Schiller: Routing scalability: An operator's view. *IEEE JSAC*, 28. évf. (2010) 8. sz., 1262–1270. p. 9 p.