



Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Kovács Bertalan

2021. évfolyam

ÁLLAPOTTARTÓ NAT44/NAT64 ÁTJÁRÓK SKÁLÁZHATÓSÁGÁNAK VIZSGÁLATA

TDK dolgozat

KONZULENS

Dr. Lencse Gábor

tudományos főmunkatárs

BUDAPEST, 2024. 12. 12.

Tartalomjegyzék

Összefoglaló	4
Abstract.....	5
1 Bevezetés	6
2 A technológiák és implementációk bemutatása.....	7
2.1 Az állapottartó NATxy átjárók.....	7
2.2 Az FD.io VPP.....	7
2.3 A DPDK.....	8
2.3.1 A keretrendszer rövid ismertetése.....	8
2.3.2 A keretrendszer fontossága	9
2.4 A siitperf.....	9
2.4.1 A szabványok követelményei	10
2.4.2 A siitperf állapotmentes működése	11
2.4.3 Mérési módszertan az állapottartó átjárók vizsgálatára	12
2.4.4 A siitperf állapottartó működése	14
2.4.5 A siitperf öntesztje	15
2.5 Az nftables	15
2.6 A Jool	16
3 A mérési környezet.....	16
3.1 A StarBED ismertetése	16
3.2 A BME-n található mérőrendszer ismertetése	17
3.3 A StarBED szerverek előkészítése.....	18
3.4 A rendszerek összehasonlító-táblázata.....	19
3.5 A BME-s szerverek előkészítése.....	19
3.6 Az implementációk beüzemelése, mérés a gyakorlatban.....	19
3.6.1 A siitperf mérőprogram beállításai	20
3.6.2 A Jool beállításai és mérése	21
3.6.3 A VPP beállításai és mérése.....	22
3.6.4 Az nftables beállításai és mérése	22
4 A mérési eredmények értékelése.....	23
4.1 A VPP NAT44 eredményei.....	23
4.1.1 Maximális kapcsolat kialakítás	23
4.1.2 Átbocsátóképesség	24

4.2 A VPP NAT64 eredményei.....	26
4.2.1 Maximális kapcsolatkiakítás	26
4.2.2 Átbocsátóképesség	27
4.3 A Jool eredményei.....	29
4.3.1 Maximális kapcsolatkiakítás	29
4.3.2 Átbocsátóképesség	31
4.4 Az nftables eredményei.....	33
4.4.1 Maximális kapcsolatkiakítás	33
4.4.2 Átbocsátóképesség	35
4.5 Az eredmények összegzése	36
4.6 Alkalmazási területek.....	37
5 Jövőbeli tervek.....	38
6 Köszönetnyilvánítás	38
Irodalomjegyzék.....	39
Függelék	41
A.1 A siitperf szkriptjei és konfigurációja	41
A.2 A Jool szkriptjei	42
A.3 A VPP szkriptjei.....	43
A.4 Az nftables szkriptjei.....	45
A.5 A VPP NAT64 plugin hibája	46

Összefoglaló

A csomagfeldolgozás és csomagtovábbítás sebessége és megbízhatósága alapvető fontosságúvá vált az internet működésében, mivel az egyre növekvő felhasználói igények és digitális szolgáltatások gyors és stabil hálózati infrastruktúrát igényelnek. Az IPv4 címek kimerülésének problémája miatt szükségessé vált az IPv6-ra való áttérés, amely azonban bonyolult folyamat, ezért a probléma ideiglenes kiküszöbölésére különböző állapottartó átjárókat, mint a NAT44 és NAT64, fejlesztettek ki. Ezek az állapottartó átjárók kulcsfontosságúak a jelenlegi hálózatokban, így teljesítményük vizsgálata létfontosságú a hatékony és megbízható internetes forgalom fenntartásához.

A dolgozatomban több állapottartó átjárót megvalósító implementációt vizsgálok meg a maximális kapcsolat kialakítás, átbocsátóképesség és skálázhatóság szempontjából, valamint bemutatom ezen technológiák működését, és foglalkozom az állapottartó átjárók vizsgálatának módszertanával és a mérésekhez használt siitperf szoftverrel is. A technológiák között foglalkozom az FD.io VPP, a Jool és az nftables implementációkkal, melyekből az FD.io VPP polling alapú csomagfeldolgozást, míg a Jool és az nftables pedig megszakítás alapú csomagfeldolgozást használ, így a dolgozatban kitérek a két technológia közötti különbségekre is. A VPP polling alapú csomagfeldolgozása nagyságrendekkel nagyobb teljesítményt ígér a DPDK technológia használatának köszönhetően, így egy erős versenytárs lehet a Jool és az nftables programokkal szemben.

Az implementációk mérését egyrészt a processzor magszám, másrészt pedig a kapcsolatok száma szerinti skálázódás vizsgálatával végzem, hogy egy átfogóbb képet kapjak az egyes implementációk képességeiről és, hogy szimulálhassam a jelenlegi hálózatokban megtalálható átjárókat, hiszen ezek sok maggal, és – az adott hálózati terheléstől függően – változó kapcsolatszámokkal működnek. Mindkét előbbi esetben elengedhetetlen a maximális kapcsolat kialakítási sebesség (maximum connection establishment rate) és az átbocsátóképesség (throughput) vizsgálata. (Előbbi a másodpercenkénti maximálisan kialakítható kapcsolatok számát, utóbbi pedig a másodpercenkénti átbocsátható csomagok számát adja.) Ezek a mérőszámok jól jellemzik egy adott technológia képességét.

A dolgozat végén kielemezem a mért eredményeket, különös figyelmet fordítva az egyes technológiák erősségeire és gyengeségeire, valamint azok lehetséges alkalmazási területeire a hálózati infrastruktúrákban. Az egyes implementációk által nyújtott eredmények kulcsfontosságú információval szolgálhatnak a jelenlegi hálózat teljesítményének optimalizálásához és a jövőbeni fejlesztési irányok meghatározásához.

Abstract

The speed and reliability of packet processing and delivery have become crucial to the operation of the Internet, as growing user demands and digital services require a fast and stable network infrastructure. The problem of the exhaustion of IPv4 addresses has necessitated the migration to IPv6, but this is a complex process, so various stateful gateway solutions such as NAT44 and NAT64 have been developed to temporarily overcome this problem. These stateful gateways are key aspects of current networks, so testing their performance is vital to maintaining efficient and reliable Internet traffic.

In this thesis, I examine several stateful gateway implementations in terms of maximum connection establishment rate, throughput and scalability. I describe the operation of these technologies, and discuss the methodology for testing stateful gateways and the siitperf software used for measurements. Among the technologies, I cover the implementations of FD.io VPP, Jool and nftables, where FD.io VPP uses polling-based packet processing whereas Jool and nftables use interrupt-based packet processing, so I also consider the differences between the two technologies. VPP's polling-based packet processing promises orders of magnitude higher performance due to its use of DPDK technology, making it a strong competitor to Jool and nftables.

I analyse the implementations by measuring the scaling by the number of cores and the number of connections to get a more comprehensive picture of the capabilities of each implementation and to simulate the gateways found in current networks, which operate with many cores and, depending on the network load, with a varying number of connections. In both cases, it is essential to examine the maximum connection establishment rate and the throughput (the former is the maximum number of connections that can be established per second and the latter the maximum number of packets that can be transmitted per second).

At the end of the thesis, I analyse the measured results, paying particular attention to the strengths and weaknesses of each technology and their potential applications in network infrastructures. The results provided by each implementation can supply key information for optimizing current network performance and defining future directions for improvement.

1 Bevezetés

Napjainkban már szinte minden az Interneten keresztül történik, kezdve a kereskedelmi és pénzügyi információáramláson keresztül, a multimédia és szórakoztató tartalmak kiszolgálásáig. Ezért nyíltan kijelenthetjük, hogy az internetes forgalom gyorsasága és megbízhatósága alapvető fontosságú mind a gazdasági, mind a társadalmi élet szempontjából. Az egyre növekvő felhasználói igények, a folyamatosan bővülő online szolgáltatások, valamint a digitális gazdaság térnyerése megköveteli a hálózatok magas szintű teljesítményét és stabilitását. Tehát a modern világban a késleltetés csökkentése, a sávszélesség növelése és a hálózatok skálázhatósága kulcsfontosságú tényezők a felhasználói élmény javításához és a hálózati torlódások elkerüléséhez.

Amikor az Internet Protokoll 4-es verzióját megtervezték, még nem számoltak azzal, hogy az Internet terjedése milyen gyors folyamat lesz. 2010 körül már majdnem teljesen kimerítette az emberiség a publikusan elérhető IPv4 címek halmazát. Több próbálkozás is született a címek „kibővítésére”, de az igazi megoldást csak az IPv6-ra való áttérés jelenthette. Sajnos az áttérés igen nehéz és komplikált, így a napjaink kutatói, jelenleg is aktívan dolgoznak a probléma megoldásán. A különféle megoldások között szerepel a NAT44 (NAPT) és a NAT64, amelyek olyan logikát megvalósító, ún. állapottartó átjárók, amik az előbbi esetben bizonyos IPv4 címeket fordítanak megfelelő IPv4 címekre, utóbbi esetben pedig megfelelő IPv6 címekre. Több ilyen átjáró is szerves részét képezi a jelenlegi internetes hálózatnak, így nagyon fontos kérdés, hogy ezen eszközök a lehető leghatékonyabban és legmegbízhatóbban működjenek.

Egy ígéretes csomagfeldolgozási módszert mutatott be az FD.io. A Vector Packet Processor (VPP, <https://fd.io>) egy nagyteljesítményű, moduláris és rugalmas szoftveres csomagfeldolgozási platform. A VPP-t arra tervezték, hogy hatékonyan kezelje a nagy forgalmú hálózatokban előforduló csomagokat, például adatközpontokban és távközlési hálózatokban [1].

A VPP implementáció mellett viszont több másik technológia is létezik, amelyek jelentős csomagfeldolgozási teljesítményekkel bírnak. Például az nftables és a is jó implementációk egy állapottartó NAT44 vagy NAT64 átjáró megvalósítására.

A dolgozat fő célja a különböző technológiák teljesítményének vizsgálata skálázódás szempontjából, egy előre jól definiált, korszerű módszerrel és az így kapott eredmények összehasonlítása és elemzése.

Dolgozatomban áttekintem az alkalmazott technológiákat, és a mérési módszertant. Bemutatom a mérési környezetet és a mérések paramétereit. Mérési eredményeimet részletesen elemzem, és kitérek a tervezett/lehetséges további mérésekre is.

2 A technológiák és implementációk bemutatása

2.1 Az állapottartó NATxy átjárók

Mivel az IANA¹ a kezdetekben a nagyobb cégeknek 10 milliós nagyságrendű IPv4-címtartományokat adott, és főként mivel az Internetre csatlakoztatott eszközök száma meghaladja az IPv4-címtér méretét, ezért az IPv4-címek kimerülése mára megnehezítette az Internet használatát. A probléma kiküszöbölésére számos megoldás volt, de a legelterjedtebb megoldást az RFC 1631 szabványban felvetett Network Address Translator (NAT) eszköz mutatta, mely bizonyos tartománybeli IPv4-címeket cserélt le egy publikus IPv4-címre, így egy publikus IPv4-cím mögé rengeteg eszközt rejtve. Később az RFC 3022 szabvány ezt a megoldást kiegészítette portszámokkal is, és megszületett a Network Address and Port Translation (NAPT), ami már TCP/UDP portszámokat is nyilvántartotta [2].

A kimerülés problémáját igazán az IPv6-címekre való áttéréssel lehet megoldani, de sajnos az átállás nem történhet meg egyik napról a másikra. Ezért jöttek létre a különböző IPv6 áttérési technológiák. Az állapottartó átjáró (NAT44, NAT64) megoldások kulcsfontosságú szerepet játszanak az Internet használhatóságában. A NAT44 és NAT64 működése majdnem teljesen azonos az előző bekezdésben említett megoldásokkal. Azért nevezik állapottartónak ezeket az eszközöket, mert az internet-szolgáltató rendszerében működnek állapottartó módon és nem pedig a felhasználói oldalon okoz gondot a nyilvántartás rezsiköltsége.

2.2 Az FD.io VPP

Maga a VPP igazából egy keretrendszer, mely alapból már nagyon sok hálózati funkciót támogat, például alapvető kapcsolási és útvonalválasztási protokollok, virtuális LAN-ok, DHCP szerver funkciók és - ami a dolgozat témájához a legközelebb áll - a NAT, pontosabban NAPT.

Ahogy a neve is mutatja, a VPP vektoros csomagfeldolgozást használ, szemben a hagyományos skaláris csomagfeldolgozással. A skaláris csomagfeldolgozás azt jelenti, hogy egyszerre csak egy csomag feldolgozásával foglalkozik a számítógép. Az ilyenkor a keletkező rendszermegszakítást feldolgozása körülményes és költséges, hiszen a megszakítás kezdetekor teljesen végig kell járni a híváshalmazon és csak ezután lehet visszatérni a megszakításból [1]. A vektoros csomagfeldolgozás ezzel szemben több csomagot dolgoz fel egyidejűleg, vagyis az

¹ Internet Assigned Numbers Authority

eszközök egyszerre több csomagot tudnak átvizsgálni és kezelni. Mivel a csomagok feldolgozása vektorosan történik, így nagyobb késleltetéssel jár, mint a hagyományos, csomagfeldolgozás, hiszen meg kell várni, míg kellő számú csomag érkezik a feldolgozandó vektorba. A vektoros csomagfeldolgozás lehetővé teszi a nagyobb teljesítményt és a jobb skálázhatóságot, még olyan helyzetekben is, ahol nagy terhelés éri a hálózatot.

A VPP ún. „worker” és „main” megnevezésű szálakat használ a csomagfeldolgozáshoz. A main szál a karmester, ez a mag koordinálja a workereket. A worker szálak pusztán a csomagfeldolgozásért felelősek. Ha nincs kitüntetett worker, akkor a main mag egymagában dolgozza fel a csomagokat. A VPP alapvetően két módszerrel konfigurálható. A `vpp-cli` (parancssoros értelmező) és a VPP konfigurációs fájljának szerkesztésével[1].

A VPP különböző Linux disztribúciókat támogat, köztük a legfontosabbak a Debian és az Ubuntu. A méréseket az Ubuntu és Debian implementáción végeztem, mert a StarBED rendszergazdái az Ubuntut javasolták, valamint a BME szerverein a Debian bizonyult stabilabbnak.

Összességében a projekt nagyon ígéretes és a VPP hatalmas átbocsátóképességet garantál. Az akár 9 MPPS² magonkénti teljesítmény [1] nagyon jól skálázódik a processzor magok számának növelésével.

2.3 A DPDK

2.3.1 A keretrendszer rövid ismertetése

A VPP „lelke” a Data Plane Development Kit (DPDK), hiszen ez a technológia teszi lehetővé, hogy a már önmagában gyors, párhuzamos csomagfeldolgozás még nagyobb teljesítményre legyen képes. A DPDK fő célja, hogy egyszerű, teljes keretrendszert és fejlesztői könyvtárat biztosítson az adatsíkbeli alkalmazások gyors csomagfeldolgozásához. A keretrendszerben megtalálható Environment Abstraction Layer (EAL) segítségével létrehozható egy olyan standard interfész, amely elrejtí a környezetre specifikus tényezőket. Így a programozók könnyen alapul vehetik a DPDK által nyújtott interfészt, és a dokumentációban feltüntetett példakódok segítségével megérthetik annak működését, és létrehozhatják a saját szoftverüket [3].

Mivel a projekt nyílt forráskódú és az Intel kezdeményezésére jött létre, ezért nagyon sok hálózati kártyához készült DPDK kompatibilis illesztőprogram. A dolgozat során felhasznált `siitperf` mérőprogram [4] is egy ilyen illesztőprogramot használ a működéséhez.

² Million packets per second - (millió csomag / másodperc)

2.3.2 A keretrendszer fontossága

A DPDK azért játszik fontos szerepet a hálózati kihívások megoldásában, mert kiküszöböli a klasszikus, megszakítás alapú csomagfeldolgozást, ezzel gyorsítva a teljes rendszer csomagfeldolgozását. Ezt úgy teszi lehetővé, hogy kiemeli a kernel térből a felhasználói térbe a csomagfeldolgozást.

2.3.2.1 Csomagfeldolgozás a Linux kernelben

A hagyományos Linux kernelben a csomagfeldolgozás úgy működik, hogy ha a hálózati eszköz kap egy keretet a fizikai interfészen, akkor az eszköz illesztőprogramja egy rendszermegszakítást okoz. Innen a csomag az életének majdnem az összes idejét kernelhívásokban tölti. A keret eközben átmásolásra kerül a rendszermemóriában megtalálható csomagsorba, majd pedig ismét több rendszerhíváson keresztül feldolgozásra kerül (például netfilter kernel modul). A csomag ezután vagy továbbításra kerül vagy a felhasználói térbe.

Ennek a folyamatnak a teljesítményét különböző skálázási technikákkal lehet könnyen növelni: Receive Side Scaling (RSS), Receive Packet Steering (RPS), Receive Flow Steering (RFS) stb. [5].

Az RSS működésének a lényege, hogy párhuzamos csomagsorokat engedélyez, így elosztva a terhelést az egyes processzor magok között, így növelve a keretek feldolgozásának teljesítményét. Az RPS és az RFS is hasonló az RSS-hez, csak mindezt szoftveresen oldják meg, és a különbség köztük az általuk használt csomagsor címezési módszer [5].

2.3.2.2 Csomagfeldolgozás a DPDK segítségével

Mivel a DPDK a hálózati interfészekhez polling módú³ illesztőprogramokat is biztosít, ezért a csomagok feldolgozása az operációs rendszer kerneléből a felhasználói térben futó folyamatokba átemelhető. Így nem kellene rendszermegszakítások, hiszen elég az adott hálózati interfész bemenetét folyamatosan lekérdezni. Ez a módszer sokkal nagyobb teljesítmény elérését biztosítja, mint a hagyományos kernel alapú csomagfeldolgozás.

2.4 A siitperf

A siitperf mérőprogram Dr. Lencse Gábor által fejlesztett nyílt forráskódú IPv6 áttérési technológiai mérőprogram. A szoftver képes a legtöbb „IPv4aaS” technológia teljesítményének lemérésére [6]. A legfontosabb ilyen technológiák: 464XLAT, lw4o6, MAP-T és MAP-E, DS-Lite.

³ lekérdezés alapú

Ezen technológiák mellett természetesen a teljesítmény méréseket IPv4 és IPv6 protokollal történő csomagtovábbítás esetén is képes elvégezni a program.

A program az RFC 8219-ben, az IETF Benchmarking Working Group által, leírt Stateless IP/ICMP Translation (SIIT) mérési-módszertan alapján készült el. Jelenleg ez az első, RFC 8219 szabványnak megfelelő, szabad szoftver mérőprogram.

2.4.1 A szabványok követelményei

A fentebb említett szabvány több követelményt is felsorol. Az RFC-ben különböző mérőszámokat definiáltak egy átjáró teljesítőképességének vizsgálatához. Ezekből a mérőszámokból a siitperf program az alábbiakat támogatja [4][7]:

- **Throughput**, magyarul: áteresztőképesség - Az a leggyorsabb sebesség, ahol a Device Under Test (DUT, a mérendő eszköz) által továbbított teszt keretek száma megegyezik a tesztelő berendezés által továbbított teszt keretek számával.
- **Latency**, magyarul: késleltetés - Az az időérték, mely úgy számolandó, hogy az ún. megjelölt keretek mindegyikére rögzíteni kell azt az időpontot, amikor a keret teljes egészében ki lett küldve (első időpont), illetve azt az időpontot, amikor a keret teljes egészében megérkezett (második időpont). A késleltetés e két időpont között eltelt idő nagysága.
- **Packet Delay Variaton**, magyarul: csomagkésleltetés-ingadozás - Az az időmennyiség, amelyet egy adott időintervallumon belül tapasztalt 99.9 percentilis késleltetési idő és minimális késleltetési idő közötti különbség ad meg.
- **Frame Loss Rate**, magyarul: csomagvesztési ráta - Az az arány mely a következő képlet alapján számolandó:

$$\frac{(n_{ki} - n_{be})}{n_{ki}} \cdot 100\%,$$

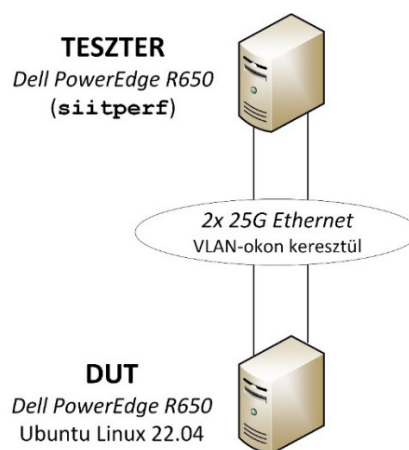
ahol n_{ki} a TESZTER által, adott sebességgel kiküldött csomagok számát, n_{be} pedig a TESZTER visszaérkezett csomagok számát jelenti.

Az RFC 2544, ami alapján az RFC 8219 is létrejött, egy átfogó mérési módszertant definiált a hálózati összekötő eszközök képességének a tesztelésére. Az RFC 5180 az IPv6 technológiával kapcsolatos specifikációkra tért ki, az RFC 8219 pedig az IPv6 áttérési technológiák esetét definiálta. Sajnos ezen RFC-k közül egy sem követelte meg az RFC 4814-ben megfogalmazott pszeudorandom portszámok használatát [8]. A siitperf az első olyan program, mely képes ezeket a portszámokat támogatni és ilyen beállítások mellett is méréseket végezni, és így megállapítani, egy átjáró teljesítményét.

2.4.2 A siitperf állapotmentes működése

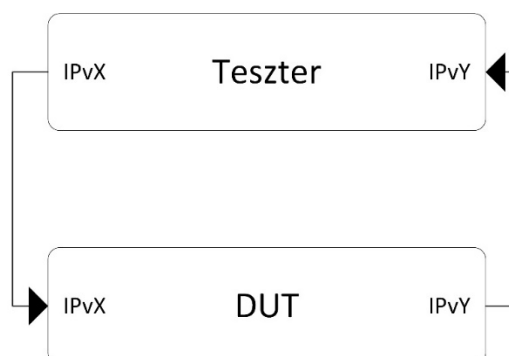
A program alapvetően három alrészből áll, melyek egyenként önmagukban is futtatható binárisok. A `siitperf-tp`, ahogy a nevéből is következtetni lehet, az áteresztőképesség és a csomagvesztés mérésével foglalkozik. A `siitperf-lat` a késleltetés mérését hivatott ellátni. Végül pedig a `siitperf-pdv`, ami a csomagkésleltetési ingadozást méri. Mindhárom program felparamétrezhető a parancssorból, illetve a `siitperf.conf` fájlból is kötelező bizonyos konfigurációs kulcs-érték párok felvétele a megfelelő működés eléréséhez [4]. A programhoz csomagolva megtalálható még néhány, a mérést segítő, `bash` szkript is, mely például a bináris keresést hajtja végre adott mérési értékek mellett.

A program dióhéjban úgy mér, hogy a konfigurációs fájlban beállított interfészén a TESZTER kiküldi a csomagokat, a szintén ott beállított DUT bemeneti interfészére, majd a DUT a feladata elvégzése után, a kimeneti interfészén kiküldi a módosított csomagot és beérkezik a TESZTER bemeneti interfészére. A 2.1. ábra egy példát mutat arra, hogy az előző folyamatban milyen utat járt be a tesztkeret.



2.1. ábra: Egy példakörnyezet a siitperf tesztek elvégzésére

A 2.2. ábra pedig a leegyszerűsített tesztösszeállítás vázát mutatja be. Az ábra úgy értelmezendő, hogy az Y és X tagok az IP verziókban a {4,6} halmaz elemei.



2.2. ábra: A tesztelés menetének sematikus ábrája [4]

Az alábbi példa azt szemlélteti, hogy hogyan is épül fel egy állapotmentes `siitperf-tp` parancs paraméterezése és azt, hogy mit jelentenek az egyes paraméterek [4]:

```
./siitperf-tp 84          3750000 60          2000          2 2  
a program    a keret mérete sebesség időtartam globális időtúllépés n m
```

A paraméterek értelmezése [4]:

- **keret mérete:** Az IPv6 keret méretére (84-1518 byte) utal. Az IPv4 keretek automatikusan 20 byte-tal rövidebbek.
- **sebesség:** A keretek átküldési sebessége. Keret per másodpercben értendő.
- **időtartam:** A tesztelés időtartama, tehát, hogy mekkora ideig tartson a keretküldés folyamata. Másodpercben értendő.
- **globális időtúllépés:** A tesztelés automatikus leállítása ennyi idő után, vagyis, ha ennyi ideig nem érkezik meg az összes kiküldött csomag a program leáll. Milliszekundumban értendő.
- **n és m:** Két relatív prím, amelyek meghatározzák az elő- és háttérforgalom⁴ arányát.

2.4.3 Mérési módszertan az állapottartó átjárók vizsgálatára

Az állapottartó NAT átjárók teljesítményének mérése az RFC 4818 által meghatározott módon, pszeudorandom portszámok használatával a valósághoz hasonló forgalmat szimulálva, nem triviális. Több mérési szcenárió is létezik [8]:

- A teszteket **egyetlen** IP-cím párral is lehet végezni. Ez pontosan az előbb említett NAT átjárók esetét írja le, amikor egy publikus címre fordítjuk a privát címünket.
- A teszteket lehet egyetlen IP-cím pár helyett egy egész IP-cím **tartománnyal** is végezni. Ez a megoldás elterjedtebb a gyakorlatban, hiszen több céleszközt is ki kell szolgálni.

Egyik megoldásban sem szabad azt elfeledni, hogy a fordítás történhet IPv4-IPv4, IPv4-IPv6 és IPv6-IPv6 között is. Problémát jelent az átjárókban megtalálható kapcsolatkövető táblázat kezelése is, mivel annak kezelése jelentős módszertani megfontolásokat igényel. Lehetséges, hogy az átjárókban a kapcsolatkövető tábla menedzselésének rezsiköltsége nagyobb, mint a címfordítás művelete, ezért ekkor a menedzselés lesz a szűk keresztmetszet. A teljesítmény mérés emiatt két fázisból tevődik össze [8].

⁴ Az RFC 8219 megköveteli, hogy a mérések forgalmának különböző aránya legyen címfordítva (előtérforgalom) és nem címfordítva, azaz natív IPv6 (háttérforgalom).

2.4.3.1 Az első fázis

Az első fázisban a DUT kapcsolatkövető táblázatát kapcsolatokkal kell feltölteni, hogy a menedzselés által okozott szűk keresztmetszetet kiküszöböljük, illetve az így kialakított kapcsolatviszonyoknak köszönhetően a TESZTER fordított irányban is képes csomagokat küldeni. Pontosan emiatt kötelező minden csomagtovábbítási teljesítmény mérésnél az első fázis futtatása [8].

A fázis elején a kezdeményező (kimeneti interfész a TESZTEREN) a DUT-on keresztül a válaszadó (bemeneti interfész a TESZTEREN) eszköznek kereteket küld adott sebességgel. A DUT ennek hatására végrehajtja az állapottartó címfordítást és eltárolja az új kapcsolatot a táblájában. A válaszadó megkapja a címfordított teszt kereteket és eltárolja a saját állapottáblájában [8].

Ha az első tesztfázis a második tesztfázis előkészítéseként kerül végrehajtásra, az alkalmazott keret sebességnek alacsonyabbnak kell lennie a maximális kapcsolatépítési sebességnél [8].

Az első fázis önmagában is végrehajtható egy állapottartó átjáró maximális kapcsolatépítési sebességének a kiderítéséhez.

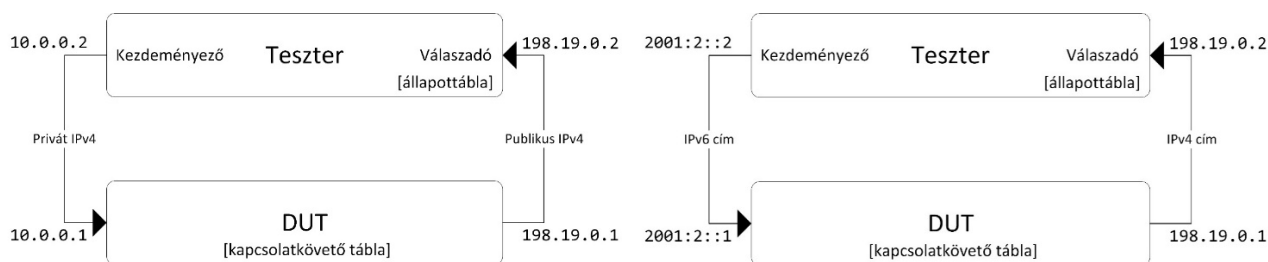
2.4.3.2 A második fázis

Ez a fázis felelős az állapottartó átjárók áteresztőképességének méréséért. Ebben a fázisban három különböző forgalomirányt különböztetünk meg mérési módszertani szempontból [8]:

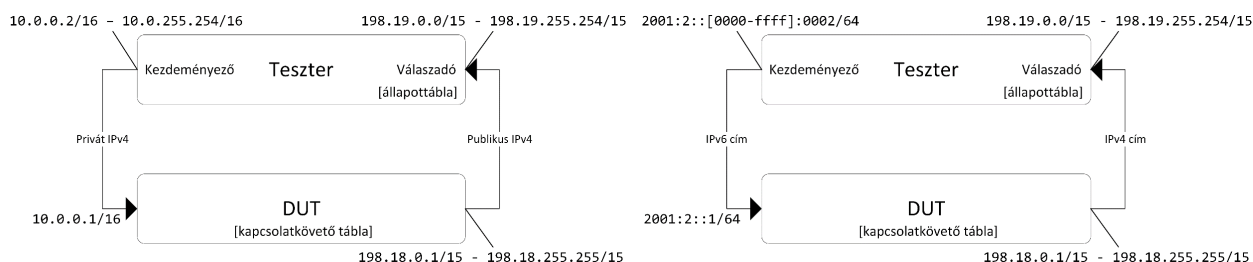
- **Kétirányú keretforgalom** - A kezdeményező elküldi a kereteket a válaszadónak és a válaszadó erre válaszolva elküldi a válasz-kereteket a kezdeményezőnek.
- **Egyirányú keretforgalom a kezdeményezőtől** - Csak a kezdeményezőtől a válaszadóig történik keretküldés.
- **Egyirányú keretforgalom a válaszadótól** - Csak a válaszadótól a kezdeményezőig történik keretküldés.

Ezekből a scenáriókból következő átbocsátóképesség mértékek az RFC 8219-ben leírtak szerint számolandók.

A mérési módszer által felállított tesztkörnyezet megértését a következő ábrák segítik [8] (az ábrákon csak az oda irány van ábrázolva, de a keret vissza irányt is bejárhat). A 2.3. ábra az egyszerű állapottartó NAT44 (NAPT) és NAT64 átjáró sematikus rajzát ábrázolja. A 2.4. ábra az IP címtartománnyal (többszörös IP-címmel) működő NAT44 és NAT64 átjáró mérési rajzát.



2.3. ábra: Tesztkörnyezet az állapotartó NAT44 (bal) és állapotartó NAT64 (jobb) méréséhez [8]



2.4. ábra: Tesztkörnyezet az IP-cím tartománnyal rendelkező állapotartó NAT44 (bal) és NAT64 (jobb) méréséhez [8]

2.4.4 A siitperf állapotartó működése

A siitperf mérőprogram képes az állapotartó átjárók mérésére is. Az eredeti program `send()` és `receive()` függvényének módosításával már elérhető volt az újabb működés. A programhoz csomagolt bash segédskript képes bináris kereséssel megállapítani a maximális kapcsolatkiépítési sebességet. Ez a lépés elengedhetetlen, hiszen a második fázis méréséhez szükséges ez az érték, mert ebben a fázisban csak az átbocsátóképesség teljesítményének mérése a cél, nem pedig a kapcsolatkiállítás sebességének. [9]. A program paraméterezése is más állapotartó működési módban. A második fázis mérése hasonló a siitperf állapotmentes átjáró méréséhez, de a mérés elvégzéséhez az előbb említett plusz paraméterek elengedhetetlenek.

A `siitperf.conf` konfigurációs fájlban megadhatók a legkisebb és legnagyobb port számértékek, melyek között az RFC 4814 szabványnak megfelelően egy pszeudorandom port érték kerül minden csomagba. Emellett képes növekvő és csökkenő port számértékek generálására is.

Az alábbi példa azt szemlélteti, hogy hogyan is épül fel egy állapotartó méréshez felparaméterezett `siitperf-tp` parancs paraméterezése és azt, hogy mit jelentenek az egyes paraméterek [9]:

```
./siitperf-tp 84          3750000 60          2000          2 2 1 1 1 10 90
a program      keret mérete sebesség időtartam globális időtúllépés n m N M R T D
```

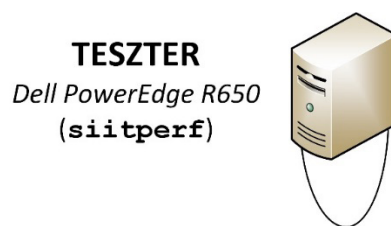
A paraméterek értelmezése [9]:

- **keret mérete - m paraméterek:** Hasonlóan a siitperf állapotmentes méréséhez.

- **N:** A teszt keretek száma az első fázisban
- **M:** A TESZTER állapot-táblájában megtalálható bejegyzések száma.
- **R:** A keretsebesség, amivel az első fázisban a keretek küldésre kerülnek. Keret per másodpercben értendő.
- **T:** A globális időtúllépés az első fázisban. Milliszekundumban értendő.
- **D:** A teljes késleltetés, amit a program az első fázis miatt vár. Milliszekundumban értendő.

2.4.5 A siitperf öntesztje

Ez a mérés abból áll, hogy mivel egy TESZTER két fizikai hálózati interfésszel is rendelkezik, önmagába hurkolható és megmérhető az eszköz saját teljesítménye. Persze érdemes arra figyelni, hogy ilyen esetben a lehető legazonosabbak legyenek a hálózati interfészek, például a maximális átviteli sebességük. Alább a 2.5. ábra egy ilyen tesztkörnyezetet szemléltet, ahol az látható, hogy egy eszköz egyik interfészét összekötöttük a másikkal és a siitperf mérőprogrammal teszteljük.



2.5. ábra: Egy példakörnyezet az öntesztre

2.5 Az nftables

Az nftables a sikeres iptables programcsalád (iptables, ip6tables, arptables, ebtables) utódja. A dolgozat azért nem foglalkozik az iptables teljesítményével, mert az iptables a 5-ös Linux kernel verzió óta elavultnak tekinthető és nagy részben leváltotta a szerepét az nftables. A szoftver egy új, a kernelen belül, csomagosztályozási keretrendszert biztosít a felhasználónak. Ez a keretrendszer egy hálózatspecifikus virtuális gépen egy új, **nft** nevű, felhasználói térbeli parancssori eszközön alapul. Az nftables újrahasznosítja a meglévő netfilter kernelrendszereket, mint például a hook infrastruktúrát, a NAT-ot és a naplózási alrendszert [10]. Tehát az nftables igazából egy új csomagolóprogram⁵ a már meglévő netfilter hook API-hoz.

Új szabály hozzáadásakor az nft parancssori eszköz lefordítja a szabálykészletet a virtuális gép bytekódjára és az nftables API segítségével átadja a kernelnek. Már létező szabálykészlet

⁵ wrapper program

elérésekor az API az előző bytekódot visszafordítja az eredeti szabálykészlet reprezentációjára [10]. A teljesítmény növelése érdekében jobb szabálykeresési algoritmusokat tartalmaz, mint az iptables.

Működésben tehát nagyon hasonló az iptables programhoz. A legnagyobb eltérést a két program között a szintaxis, a táblák és láncok konfigurálhatósága és a szabályok működése adja [10]. Az nftables szoftverben nincsenek alapértelmezett táblák. Ezeket manuálisan kell hozzáadni és személy szerint testre szabni. A láncokkal is hasonlóan kell eljárni, ismét a teljesítmény növelés jegyében. A másik különbséget a szabályok működése adja, hiszen egy szabályhoz nem csak egy feltételt lehet hozzáadni, hanem akár többet is [10]. A feltételek kiértékelése a szabályhoz való hozzáadásuk sorrendjében történik. Emellett egy szabály több akciót képes végrehajtani egy csomaggal, ismét a hozzáadási sorrendjükben.

2.6 A Jool

Több megoldás született az IPv4-IPv6 átmenet megvalósítására. A [11] jól bemutatja ezeket a technológiákat. Az RFC 9313 által megvizsgált lényegesebb technológiák és a róluk szóló RFC-k felsorolása [12]-ben olvashatóak.

A technológiák implementálása nem egyszerű feladat, és a Network Information Centre México (NIC-Mexico) arra vállalkozott, hogy ezekből az „IPv4aaS” technológiákból, a MAP-T és 464XLAT variációkra, egy szoftvercsomagot valósít meg. A Jool eredetileg egy nyílt forráskódú SIIT és NAT64 implementáció Linux rendszerekre [13]. A 4.2 verziótól már a MAP-T technológiát is támogatja. A program nemcsak felhasználói szoftverszámokat csomagol, hanem egy speciális kernel modult is.

A Jool konfigurálása a jool felhasználói térbe települő programmal végezhető. A Jool két módban képes üzemelni. Az egyik a „netfilter”, a másik pedig az „iptables” mód. Az előbbi módban a Jool, minden a kernelbe beérkező csomagot elkap és feldolgozza, míg utóbbiban iptables szabályokon keresztül lehet megadni, hogy melyik csomagokat dolgozza fel az implementáció [13]. A dolgozatomban a „netfilter” módot alkalmaztam.

3 A mérési környezet

3.1 A StarBED ismertetése

A konzulensemnek, Lencse Gábornak, és a National Institute of Information and Communication Technology (NICT) „StarBED5” projektjének köszönhetően a Japánban található általános felhasználású Internet tesztkörnyezet segítségével végeztem a méréseket.

A "StarBED" megépítésének alapkonceptiója az volt, hogy az Internet kutatását és fejlesztését támogassa azáltal, hogy olyan tesztkörnyezetként szolgál, amely többszörös, nagyszabású szimulációk elvégzésére használható, és állandó szimulációs létesítményt biztosít [14]. Ebben a tesztkörnyezetben kaptam két vállalati minőségű szervert a dolgozat elkészüléséhez szükséges mérések elvégzésére. A teljes paramétereikről bővebben a 3.1. táblázat értekezik.

A magas minőségű szerverekre azért van szükség, hogy egy internetszolgáltató gyakorlatban használt eszközeihez legjobban hasonlító hardveren lehessen elvégezni a méréseket. A szerverben megtalálható nagy teljesítményű processzorok segítségével könnyen lemérhető az egyes implementációk skálázhatósága és a redundáns SSD tárolóknak köszönhetően a mérési adatok biztonsága is garantált.

A StarBED infrastruktúrában megtalálható csomagkapcsoló (switch) eszközöket direktben nem volt jogom konfigurálni, de a webes node-konfiguráló felületen lehetőségem volt az egyes szerverek, egyes portjait bizonyos VLAN-okba kapcsolni, így biztosítani a két komponens közti kommunikációt.

A mérőkörnyezethez való kapcsolódás meglehetősen körülményes folyamat. Magát az egész StarBED környezetet csak Secure Sockets Layer Virtual Private Network (SSL VPN) program segítségével lehet elérni. SSL VPN szoftverként a FortiClient VPN⁶ ingyenes verzióját használtam.

3.2 A BME-n található mérőrendszer ismertetése

Ismét a konzulensem jóvoltából a BME is használatra bocsátott kettő Dell PowerEdge R730 típusú szervert, melyek hasonló hálózati topológiában voltak elrendezve, mint a StarBED-ben található szerverek, de közvetlen kábelkapcsolattal voltak összekötve. A teljes paramétereikről bővebben a 3.1. táblázat értekezik.

A szerverekhez egy jump-szerveren⁷ keresztül tudtam csatlakozni, illetve a távoli konfiguráláshoz még segítségemre volt egy VPN szerver, amelyen a szerverek Intelligent Platform Management Interface (IPMI) rendszerét tudtam elérni.

⁶ <https://www.fortinet.com/support/product-downloads>

⁷ A jump-szerver egy közvetítő eszköz, amely a forgalomnak a tűzfalakon keresztül történő átirányításáért felelős egy felügyelt biztonságos csatorna segítségével.

A BME-s szerverek sokkal kevesebb maggal rendelkeznek, mint a StarBED-eszközök, viszont magonként nagyobb órajel frekvenciájúak. A StarBED szerverei 2,2GHz-en, a BME szerverei pedig 3,2GHz-en futnak.

3.3 A StarBED szerverek előkészítése

A kapott két szerveremre Ubuntu 22.04.3 LTS operációs rendszert telepítettem. A telepítéseket a StarBED webes felületén az Integrated Dell Remote Access Controller⁸ (iDRAC) távoli server menedzsment eszköz segítségével végeztem el. A nagy távolságból adódó késleltetés és a VPN miatt a telepítőképek felmásolása hosszadalmas feladat volt. Az iDRAC segítségével könnyedén állíthattam a Unified Extensible Firmware Interface (UEFI) beállításokat is.

A két szerverből az egyik a TESZTER, a másik pedig a DUT szerepet tölti be a mérés során. Az FD.io VPP, az nftables és a Jool implementációkat a DUT futtatja. Az egyes implementációkat a szerverek beüzemelése után telepítettem fel a rendszerekre.

A siitperf mérőprogram forrását a konzulensem által szolgáltatott tömörített fájlból csomagoltam ki, majd pedig a *make* segédprogram segítségével fordítottam a TESZTER gépen. A DUT gépekre az nftables programot nem kellett feltelepítenem, hiszen az az Ubuntu operációs rendszerrel csomagolva érkezett. A Jool programot a hivatalos oldalról⁹ az ott megtalálható leírás alapján telepítettem fel. A VPP telepítése pedig ismét a hivatalos oldal¹⁰ alapján történt. Az egyes technológiák beállításait a 3.6. fejezet taglalja.

⁸ <https://www.dell.com/en-us/lp/dt/open-manage-idrac>

⁹ <https://nicmx.github.io/Jool/en/install.html>

¹⁰ <https://s3-docs.fd.io/vpp/24.06/gettingstarted/installing/>

3.4 A rendszerek összehasonlító-táblázata

	BME		StarBED	
	Tesztter	DUT	Tesztter	DUT
Szerver modell	Dell PowerEdge R730		Dell PowerEdge R650	
Processzor	2db Intel Xeon E5-2667 v3		2db Intel Xeon Gold 6330N	
Processzor magszám	8		28	
Processzor órajel	3,2 GHz (fix)		2,2 GHz (változó)	
Processzor turbo	-	-	engedélyezve	letiltva
Hálózati kártya	Intel X540-AT2 10 Gbit/s NIC		Intel E810-XXV 25 Gbit/s NIC	
Rendszermemória	128 GB 2133 MT/s DDR4		512 GB 2667 MT/s DDR4	
Hyperthreading	letiltva		letiltva	
Operációs rendszer	Debian 11.7	Debian 11.11	Ubuntu 22.04.4 LTS	
Kernel verzió	5.10.0-27-amd64		5.15.0-122-generic	
VPP verzió	-	v24.06-release	-	v24.06-release
Jool verzió	-	4.1.11.0	-	4.1.11.0
nftables verzió	-	v1.8.7	-	v1.8.7

3.1. táblázat: A mérési környezetekben található rendszerek adatai

3.5 A BME-s szerverek előkészítése

A két BME-s, az I épületben megtalálható, szerveremre a StarBED-ről átklónozott Debian 11-es képfájlját telepítette előzőleg a konzulensem. Azért jutottunk erre a döntésre, mert egy megbízható rendszert már előzőleg összeállított a konzulensem a StarBED-en, és így már előre telepítve volt minden szükséges könyvtár és csomag.

Itt is hasonlóan jártam el, mint a StarBED esetében és az egyik szerver a TESZTER, a másik, pedig a DUT szerepét töltötte be. Fontos beállítás volt a GRUB bootloader GRUB_CMDLINE_LINUX_DEFAULT paraméterébe az intel_iommu=off flag beillesztése, mert a DPDK illesztőprogramjai csak így működtek. Ezt mind a DUT, mind pedig a TESZTER szervereken meg kellett tennem, hiszen mind a siitperf, mind a VPP DPDK-t használ.

A siitperf mérőprogramot a képfájl „hozta” magával, így ennek a telepítésével és fordításával nem foglalkoztam.

3.6 Az implementációk beüzemelése, mérés a gyakorlatban

Célom, hogy megvizsgáljam, az egyes technológiák skálázhatóságát a processzor magszám, és a kapcsolattartó tábla méretének függvényében. Az így kapott metrikák alapján már mért adatokkal lehet alátámasztani egy technológiáról, hogy jobb-e, mint a másik. Az eredmények megállapításában hatalmas segítséget nyújtott a siitperf mérőprogram. A mérések konzisztenciájának érdekében, a BME szerverein fix processzor órajelet használtam. A StarBED szerverein nem tudtam fix órajelet állítani a tlp segédprogrammal. A StarBED szerverein, a DUT eszközökön kikapcsoltam, a

TESZTER eszközökön pedig bekapcsoltam a turbo funkciót, a BIOS rendszerben. Előbbire azért volt szükség, hogy az egyes magokon konzisztens legyen az órajel, utóbbira pedig azért, hogy a TESZTER teljesítménye a mérés során nagyobb legyen, mint a DUT teljesítménye.

Mérések során a forrás portszámok száma mindig 40 000 és a célportszámok száma változik. Ez azért van, mert az internetes forgalomban megfigyelhető forrás és cél portszámok nem szimmetrikusak. Amíg forrás portszámok értékei véletlenszerűek, egy általános esetben, addig a cél portszámok értékei előre definiáltak [8].

A következő alfejezetek az egyes technológiák beállításával kapcsolatos teendőkkel foglalkoznak.

3.6.1 A siitperf mérőprogram beállításai

A siitperf mérőprogram sok hasznos szkriptet tartalmaz a mérések elvégzéséhez, így például egy bináris keresés során nem kell manuálisan felparaméterezni és meghívni a mérőprogramot. A dolgozat során főleg a `binary-rate-alg.sh-sf` és a `binary-rate-alg.sh-R` szkriptekkel fogok teljesítményt mérni. Előbbi az állapottartó átjárok átbocsátóképességét, míg utóbbi az átjárok maximális kapcsolatfelépítési sebességét¹¹ méri.

Minden technológia mérése előtt el kell végezni a maximális kapcsolatfelépítési sebességet meghatározó mérést, majd ennél a maximumnál egy kisebb számot választva, el lehet végezni egy átbocsátóképesség mérést. Erre azért van szükség, mert az átbocsátóképesség mérés során okozhatná így a szűk keresztmetszetet a kapcsolatok kiépítési sebessége [8].

A siitperf maga is a DPDK keretrendszert használja csomagok küldéséhez, így egy, a mérőprogramhoz csomagolt szkripttel beállítottam a TESZTER gépen, hogy a siitperf konfigurációs fájljában feltüntetett interfészek a DPDK által biztosított illesztőprogramot használják.

Mivel a siitperf nem válaszolja meg sem az ARP, sem az NDP feloldás kéréseket, így az egyes implementációk során, az azoknak megfelelő módon, beállítottam a DUT interfészeihez a megfelelő IP-cím – MAC-cím összerendeléseket, hogy a TESZTER interfészeivel való kommunikáció zökkenőmentesen működhessen. Emellett az interfészek megfelelő IP-címeit is felvettem a TESZTER gépen, melyeket a [15] forrás alapján javasolt értékeknek választottam. Ezt a siitperf `siitperf.conf` konfigurációs fájljában tettem meg. A DUT gépen még egy fontos dolgot kellett beállítanom, ez pedig a [6] alapján a fentebb említett RSS bekapcsolása volt az egyes interfészekre. Erre azért volt szükség, mivel a siitperf UDP csomagokkal végzi a mérést és ebből következően a

¹¹ maximum connection establishment rate [8]

Linux kernel csomagelosztó hash-függvénye ne csak az IP-címeket, hanem a portszámokat is figyelembe vegye a csomagok elosztásakor az egyes processzor magok között is.

A siitperf megfelelő működéséhez, a TESZTER rendszereken, a GRUB bootloader GRUB_CMDLINE_LINUX_DEFAULT paraméterébe a hálózati kártyával azonos NUMA node-on lévő processzor magokat izoláltam, hogy az operációs rendszer ütemezője ne tudjon az adott magokra ütemezni. Ezt az `isolcpus=2,4,6,8` és `isolcpus=4,8,12,16` beállításokkal végeztem el, előbbit, a BME, utóbbit, a StarBED esetében alkalmazva.

Az egyes szkriptek megtalálhatóak a függelék A.1 szakaszában.

3.6.2 A Jool beállításai és mérése

A dolgozat megírásakor a legfrissebb Jool verzió a 4.1.11-es volt, és ezt a verziót választottam a mérések elvégzéséhez. A Jool beállításához több szkriptet is használtam. Az első szkript (`set-jool`), betöltötte a Jool kernel modulját és hozzáadott egy IPv6-címtartományt a Jool nyilvántartásához. Analóg módon hozzáadtam egy IPv4-címtartományt is a Jool nyilvántartásához. A második szkript (`del-jool`), kitörölte a Jool kernel modulját, így kikapcsolva a Jool programot. A jool megfelelő működéséhez fel kellettennem a helyes interfészek IP-címeit is, valamint az ARP és NDP feloldások hiánya miatt keletkezett IP-cím – MAC-cím összerendeléseket is. Az egyes szkriptek megtalálhatóak a függelék A.2 szakaszában.

A maximális kapcsolatfelépítési sebesség mérésekor a siitperf programmal csomagolt bináris kereső szkript a fent említett két szkriptet hívta minden elemi mérés előtt és után. Értelemszerűen a `set-jool` szkript a mérés megkezdése előtt, a `del-jool` szkript pedig a mérés befejezése után hívódott meg. Erre azért volt szükség minden méréskor, mert a Jool egy mérés folyamán megtöltötte az állapottábláját és ezt ki kellett üríteni, mivel a következő mérési eredményét javította volna a már megtöltött kapcsolattábla jelenléte [6].

Az áteresztőképesség mérésekor az előző bekezdésben meghatározott maximális kapcsolatfelépítési sebesség segítségével meghatározott csomagátviteli rátánál egy kisebb értéket választottam a tesztek futtatásakor. Erre a 3.6.1 fejezet ad magyarázatot. A mérések folyamán ismét minden elemi siitperf mérés előtt és után kellett használnom a `set-jool` és `del-jool` szkripteket, hogy az egyes mérések üres állapottáblával induljanak.

A Jool mérésekor a használt magszám befolyásolását a GRUB bootloader GRUB_CMDLINE_LINUX_DEFAULT paraméterében a `maxcpus` flag megfelelő felparaméterezésével tettem, így elérve az 1,2,4,8 ... magszámot.

3.6.3 A VPP beállításai és mérése

A VPP „v24.06-release” verzióját használtam a mérések elvégzéséhez. Mivel az FD.io VPP is a DPDK keretrendszert használja, így itt is elengedhetetlen lépés engedélyezni a megfelelő DPDK illesztőprogramok használatát a DUT gépen. Ebben a siitperf mérőprogrammal csomagolt szkript volt a segítségemre. A VPP beállításához a parancssori értelmezőjét alkalmaztam, a vppctl-t. Ismét két szkriptet használtam a mérések elvégzéséhez, az egyik a set-vpp, illetve a set-vpp-64 a másik pedig a del-vpp volt. Az előbbi beállította az egyes interfészeket a DUT gépen, adott technológiának megfelelően (NAT44 vagy NAT64), az utóbbi pedig kikapcsolta a VPP működését így kiürítve az állapottábláját. Itt a siitperf mérőprogramhoz hasonlóan be kellett állítani az IP-cím – MAC-cím összerendeléseket, hasonló okok miatt, mint a 3.6.1 fejezetben [16].

A VPP konfigurációs fájljában megadtam, hogy 50 GB memóriát használhat, illetve felsoroltam az egyes méréseknél a megfelelő worker szálak számát. A szálakat úgy választottam meg, hogy azonos NUMA node-on helyezkedjenek el a hálózati kártyával, hogy minél kisebb legyen az adatok mozgatásának a rezsiköltsége. Fontos beállítás volt még, hogy a kernel ne ütemezzen a VPP által használt magokra, így ezeket a GRUB bootloader GRUB_CMDLINE_LINUX_DEFAULT paraméterében az isolcpus flag megfelelő felparaméterezésével tettem meg.

Az egyes szkriptek és konfigurációs fájlok megtalálhatóak a függelék A.3 szakaszában.

3.6.4 Az nftables beállításai és mérése

Az nftables méréséhez is két fő szkriptet használtam (set-nftables és del-nftables), hasonlóan a Jool méréséhez. Az első szkript (set-nftables), létrehozott egy nftables szabályt és a hozzá szükséges táblát is. A szabály az egyik hálózati interfészen érkező csomagok IPv4-címét és forrás portszámát cseréli ki egy IPv4-cím és egy portszám kettesre. Tehát definíció szerint egy állapottartó NAT44 átjáró szerepét tölti be. A második szkript (del-nftables), egyszerűen törölte az összes nftables szabályt ezzel ismét azt elérve, hogy üres legyen az állapottáblája az átjárónak. Az egyes szkriptek megtalálhatóak a függelék A.4 szakaszában.

A maximális kapcsolatfelépítési sebesség és az áteresztőképesség mérésekor hasonlóan jártam el, mint a Jool esetében. Az egyes mérőszkriptek ismét a beállító és törlő szkripteket hívták meg az elemi mérések előtt és után.

Az nftables esetében is GRUB bootloader GRUB_CMDLINE_LINUX_DEFAULT paraméterében állítottam át a rendszer által használható maximális magok számát.

4 A mérési eredmények értékelése

4.1 A VPP NAT44 eredményei

4.1.1 Maximális kapcsolatkialakítás

Mérési környezet	BME			StarBED		
Kapcsolatok (db)	400 000	4 000 000	40 000 000	400 000	4 000 000	40 000 000
Workerek (db)	1 main core és 1 worker			1 main core és 1 worker		
Forrás portszám (db)	40 000	40 000	40 000	40 000	40 000	40 000
Cél portszám (db)	10	100	1 000	10	100	1 000
Minimum (cps)	1 076 131	967 772	898 191	334 707	299 803	178 709
Maximum (cps)	1 085 991	985 611	914 551	338 136	304 932	180 732
Medián (cps)	1 080 785	980 801	907 133	335 934	303 301	180 219
Diszperzió ¹² (%)	0,91	1,82	1,80	1,02	1,69	1,12

4.1. táblázat: A VPP NAT44 maximális kapcsolatkialakítási sebessége egy main core és egy dedikált worker használatával a kapcsolatok skálázódása szerint

Mérési környezet	BME				StarBED			
Workerek (db)	0	1	2	4	0	1	2	4
Kapcsolatok (db)	4 000 000				4 000 000			
Forrás portszám (db)	40 000				40 000			
Cél portszám (db)	100				100			
Minimum (cps)	935 541	967 772	935 483	929 686	296 354	299 803	330 925	301 748
Maximum (cps)	947 816	985 611	949 280	951 172	299 816	304 932	334 596	304 718
Medián (cps)	945 755	980 801	945 672	945 643	298 073	303 301	333 086	303 561
Diszperzió (%)	1,30	1,82	1,46	2,27	1,16	1,69	1,10	0,98

4.2. táblázat: A VPP NAT44 maximális kapcsolatkialakítási sebessége 4 000 000 kapcsolat mellett a workerek skálázódása szerint

¹² A 99. percentilis és az 1. percentilis különbségének a mediánnal vett hányadosa [7].

A VPP NAT44 kapcsolat kialakítási sebessége (ld. 4.1. táblázat), a BME szerverein, jól skálázódik a kapcsolatszámok tekintetében, mivel a kapcsolatok számának tízszeresítésének ellenére a teljesítménye csupán csak nagyjából 10%-ot romlott méretenként. A diszperzió mértékéből az állapítható meg, hogy kellően stabil technológiáról beszélünk.

A StarBED rendszerein kisebb teljesítmény és rosszabb skálázódás figyelhető meg. Ennek egyszerű az oka, a BME szerverein a CPU sokkal magasabb frekvencián működik, mint a StarBED processzorai, így sokkal több csomagot képes feldolgozni. A másik ok az, hogy a dolgozat írása során a StarBED rendszerein nem tudtam fix órajelet beállítani, így sokszor változott mérések közben is az órajel, nem minden esetben a maximum teljesítményt nyújtva.

A méréseknél, a workerek skálázódásának vizsgálatánál (ld. 4.2. táblázat), 4 000 000 kapcsolatot választottam a kapcsolattábla méretének, mert egy komolyabb internetszolgáltató hálózatában nagyjából akkora kapcsolatszámmal kell számolni [18].

Mindkét mérési környezetben azt figyeltem meg, hogy először 0 és 1 worker esetén a teljesítmény javult, bár igen rossznak mondható skálázódással. Viszont, amikor 2 és 4 workert használtam a teljesítmény nem, hogy nem javult semennyit, hanem még romlott is.

Különösen fontosnak tartom megjegyezni, hogy az átbocsátóképesség mérési eljárása olyan, hogy egy adott elemi mérés már egyetlen egy csomag elvesztése esetén is sikertelen. A mérési naplófájl szerint a StarBED esetén viszonylag kis számú csomag elvesztése miatt voltak sikertelenek a bináris keresés egyes lépései.

4.1.2 Átbocsátóképesség

Mérési környezet	BME			StarBED		
Kapcsolatok (db)	400 000	4 000 000	40 000 000	400 000	4 000 000	40 000 000
Workerek (db)	1 main core és 1 worker			1 main core és 1 worker		
Forrás portszám (db)	40 000	40 000	40 000	40 000	40 000	40 000
Cél portszám (db)	10	100	1 000	10	100	1 000
Minimum (pps)	2 820 554	2 540 988	2 265 368	949 192	702 748	368 954
Maximum (pps)	2 825 332	2 543 532	2 270 020	965 118	716 444	398 438
Medián (pps)	2 823 896	2 542 696	2 267 374	955 416	712 765	392 218
Diszperzió (%)	0,17	0,10	0,21	1,67	1,92	7,52

4.3. táblázat: A VPP NAT44 átbocsátóképessége egy main core és egy dedikált worker használatával a kapcsolatok skálázódása szerint

A BME szerverein az átbecsítőképességek kapcsolatszámok szerinti skálázódása (ld. 4.3. táblázat) is jónak mondható, mivel ismét csak 300 000 pps, nagyjából 10%, degradáció látható az egyes kapcsolatértékek növekedésével. A diszperzió mértéke megint kiváló, stabilnak mondható a technológia.

A StarBED rendszerein kisebb teljesítmény figyelhető meg, az előzőekben tárgyalt okok miatt. A diszperzió csak a 40 000 000 kapcsolatszám esetén volt nagyobb, de ez az érték is elfogadható.

Mérési környezet	BME				StarBED			
Workerek (db)	0	1	2	4	0	1	2	4
Kapcsolatok (db)	4 000 000				4 000 000			
Forrás portszám (db)	40 000				40 000			
Cél portszám (db)	100				100			
Minimum (pps)	2 515 132	2 540 988	2 514 978	2 515 560	609 192	702 748	598 860	591 670
Maximum (pps)	2 519 562	2 543 532	2 520 020	2 523 438	646 240	716 444	668 508	665 076
Medián (pps)	2 517 934	2 542 696	2 517 693	2 518 760	642 390	712 765	644 880	617 920
Diszperzió (%)	0,18	0,10	0,20	0,31	5,77	1,92	10,80	11,88

4.4. táblázat: A VPP NAT44 átbecsítőképessége 4 000 000 kapcsolat mellett a workerek skálázódása szerint

A workerek skálázódásának helyzete (ld. 4.4. táblázat) hasonlóan alakult, mint a maximális kapcsolat kialakítás esetén, hiszen itt is az tapasztalható, hogy a dedikált worker alkalmazásával stabilabb és nagyobb teljesítmény érhető el, viszont több worker alkalmazása esetén rosszul skálázódik az implementáció.

A StarBED esetében egy dedikált worker alkalmazásával érhető el a legstabilabb csomagtovábbítás. Ez a BME szervereinél is megfigyelhető, hiszen mindkét esetben ott volt a legkisebb a diszperzió értéke.

A fix és magasabb órajelnek köszönhetően a BME ismét kimagaslóan szerepelt a két rendszer közötti teljesítményversenyben.

4.2 A VPP NAT64 eredményei

4.2.1 Maximális kapcsolatkiakítás

Mérési környezet	BME			StarBED		
Kapcsolatok (db)	400 000	4 000 000	40 000 000	400 000	4 000 000	40 000 000
Workerek (db)	1 main core és 1 worker			1 main core és 1 worker		
Forrás portszám (db)	4 000 ¹³	4 000	4 000	4 000	4 000	4 000
Cél portszám (db)	100	1 000	10 000	100	1 000	10 000
Minimum (cps)	823 387	75 137	8 151 ¹⁴	71 426	12 254	1 048
Maximum (cps)	910 938	81 119	8 174	83 541	13 587	1 053
Medián (cps)	885 638	79 692	8 157	77 313	12 516	1 049
Diszperzió (%)	9,89	7,51	0,28	15,67	10,65	0,48

4.5. táblázat: A VPP NAT64 maximális kapcsolatkiakítási sebessége egy main core és egy dedikált worker használatával a kapcsolatok skálázódása szerint

A VPP NAT64 implementációjának teljesítményvizsgálatakor, a maximális kapcsolatkiakítási sebesség, kapcsolatszám szerinti skálázódásánál (ld. 4.5. táblázat) egy main core és egy worker párost alkalmaztam.

Könnyen megfigyelhető, hogy a kapcsolatok számának növelésével jelentős teljesítménybeli degradáció fedezhető fel (nagyjából 85% kapcsolatléptékenként a BME esetében). Ez leginkább a BME szervereinek értékeinél látható. Kijelenthető, hogy az implementáció nagyon rosszul skálázódik.

A BME eredményei stabilabbak voltak, mint a StarBED eredményei, ez a fix órajel értéknek köszönhető. A BME szervereinek a teljesítménye, hasonló okok miatt, sokkal jobb volt, mint a StarBED rendszereinek a teljesítménye.

¹³ A VPP NAT64 implementációjának mérésekor egy hibát találtam. Emiatt nem 40 000 db forrásportot, hanem 4 000 forrásportot használtam. (Bővebben: a függelék A.5 szakaszában)

¹⁴ Mivel a VPP NAT64 implementációja körülbelül 10 000 kapcsolat kialakítására, valamint hasonló értékű átbocsátóképességre alkalmas, ezért a 40 000 000 kapcsolatszám esetében csak öt mérést végeztem.

Mérési környezet	BME				StarBED			
Workerek (db)	0	1	2	4	0	1	2	4
Kapcsolatok (db)	4 000 000				4 000 000			
Forrás portszám (db)	4 000				4 000			
Cél portszám (db)	1 000				1 000			
Minimum (cps)	80 563	75 137	81 367	81 232	12 455	12 254	13 172	13 133
Maximum (cps)	81 676	81 119	81 741	81 724	13 574	13 587	13 550	13 580
Medián (cps)	81 600	79 692	81 564	81 525	12 845	12 516	13 410	13 281
Diszperzió (%)	1,36	7,51	0,46	0,6	8,71	10,65	2,82	3,37

4.6. táblázat: A VPP NAT64 maximális kapcsolatkiakítási sebessége 4 000 000 kapcsolat mellett a workerek skálázódása szerint

A workerek skálázódásánál (ld. 4.6. táblázat), ismét 4 000 000 kapcsolatot alkalmaztam [18]. A VPP NAT64 implementációjánál minimális teljesítményjavulást fedeztem fel a workerek számának növelésével. Érdekes anomália, hogy amikor egy dedikált workert használt az implementáció, akkor volt a mérési eredmények között a legnagyobb diszperzió, mint a BME, mind pedig a StarBED esetében is.

A BME ismét sokkal jobb teljesítményt nyújtott, mint a StarBED, mivel több kapcsolatot tudott másodpercenként kialakítani, illetve sokkal kisebb diszperzióval is dolgozott. Az implementáció stabilnak mondható a kis diszperziós értékek miatt.

4.2.2 Átbocsátóképesség

Mérési környezet	BME			StarBED		
Kapcsolatok (db)	400 000	4 000 000	40 000 000	400 000	4 000 000	40 000 000
Workerek (db)	1 main core és 1 worker			1 main core és 1 worker		
Forrás portszám (db)	4 000	4 000	4 000	4 000	4 000	4 000
Cél portszám (db)	100	1 000	10 000	100	1 000	10 000
Minimum (pps)	2 812 498	793 748	20 042	789 056	30 346	3987
Maximum (pps)	2 837 654	812 556	20 446	801 402	31 260	4003
Medián (pps)	2 833 362	803 224	20 290	799 034	30 850	4012
Diszperzió (%)	0,89	2,34	1,99	1,55	2,96	0,62

4.7. táblázat: A VPP NAT64 átbocsátóképessége egy main core és egy dedikált worker használatával a kapcsolatok skálázódása szerint

A VPP NAT64 átbocsátóképesség méréseinél, a kapcsolatok szerinti skálázódás méréseknél (ld. 4.7. táblázat) már jobb teljesítményeredmények születtek. Az eredmények skálázódása nagyon rossznak mondható, ez megint a BME szerverein figyelhető meg jobban. Itt első esetben körülbelül 70%-os, második esetben nagyjából 95%-os teljesítményromlással járt.

A StarBED rendszerein mért eredményeiből is könnyen látható, hogy a kapcsolatok számának növelésével súlyosan romlott az implementáció teljesítménye.

A BME szerverei ismét jobban teljesítettek, mint a StarBED szerverei, de ez a már megszokott indokok miatt van így.

Mérési környezet	BME				StarBED			
Workerek (db)	0	1	2	4	0	1	2	4
Kapcsolatok (db)	4 000 000				4 000 000			
Forrás portszám (db)	4 000				4 000			
Cél portszám (db)	1 000				1 000			
Minimum (pps)	785 914	793 748	793 336	786 708	30 346	34 112	34 270	33 590
Maximum (pps)	806 642	812 556	812 508	806 326	31 260	35 164	35 356	35 986
Medián (pps)	795 090	803 224	799 756	800 736	30 850	34 512	34 521	34 358
Diszperzió (%)	2,61	2,34	2,40	2,45	2,96	3,05	3,15	6,97

4.8. táblázat: A VPP NAT64 átbocsátóképessége 4 000 000 kapcsolat mellett a workerek skálázódása szerint

Az átbocsátóképesség, workerek skálázódása szerinti, méréscsoportjánál (ld. 4.8. táblázat) azt tapasztaltam, hogy itt is egy minimális javulást lehet felfedezni, egy dedikált worker használata során, azonban, ha a workerek száma növekszik, itt is teljesítménybéli stagnálás, illetve minimális romlás észlelhető.

A StarBED rendszerein nagyságrenddel gyengébb eredmények születtek, mint a BME szerverein. Itt is az figyelhető meg, hogy egy dedikált worker alkalmazásával javul a teljesítmény, viszont utána egyre több worker bevonásával romlik a teljesítmény.

A diszperzió értékek a BME esetében alacsonyabbak voltak, mint a StarBED esetében, viszont az eltérés marginális. Az implementáció kellően stabilnak bizonyult, viszont a teljesítménye alacsony.

4.3 A Jool eredményei

4.3.1 Maximális kapcsolatkialakítás

Mérési környezet	BME			StarBED		
Kapcsolatok (db)	400 000	4 000 000	40 000 000	400 000	4 000 000	40 000 000
CPU magok (db)	16			56		
Forrás portszám (db)	40 000	40 000	40 000	40 000	40 000	40 000
Cél portszám (db)	10	100	1 000	10	100	1 000
Minimum (cps)	781 249	585 936	410 218	525 722	496 876	338 508
Maximum (cps)	813 492	625 001	460 938	571 880	526 465	365 644
Medián (cps)	809 716	589 635	418 493	562 500	507 747	352 622
Diszperzió (%)	3,98	6,63	12,12	8,21	5,83	7,70

4.9. táblázat: A Jool NAT64 maximális kapcsolatkialakítási sebessége a maximális magszám mellett a kapcsolatok skálázódása szerint

A Jool NAT64 implementációja a kapcsolatok száma szerint (ld. 4.9. táblázat) elfogadhatóan skálázódott. Mind a BME, mind a StarBED rendszerein a maximális magszám mellett mértem az egyes átbocsátóképeség értékeket, mert úgy gondoltam, úgy van a legnagyobb esélye az implementációknak érvényesülni.

A BME szerverein, csak a 40 000 000 kapcsolat mellett volt magasabb a diszperzió, de ott ez elfogadható érték, mivel a mért rendszer nagy terhelés mellett volt használva. Az egyes kapcsolatléptékek megtételével a teljesítmény nagyjából 25%-ot romlott.

A StarBED rendszerein is az volt megfigyelhető, hogy a kapcsolatok számának növelésével csökkent a teljesítmény, de 400 000 kapcsolatról 4 000 000 kapcsolatra ugorva csak nagyjából 10%-os teljesítmény romlást lehetett észlelni. A 4 000 000 kapcsolatról a 40 000 000 kapcsolatra való növelés pedig körülbelül 31%-os romlást jelentett.

A BME szerverei jobban teljesítettek, mint a StarBED rendszerei. Ez nagy valószínűséggel a magasabb órajelnek köszönhető.

Mérési környezet	BME				
CPU magok (db)	1	2	4	8	16
Kapcsolatok (db)	4 000 000				
Forrás portszám (db)	40 000				
Cél portszám (db)	100				
Minimum (cps)	261 668	397 231	562 499	576 902	585 936
Maximum (cps)	261 668	427 740	578 126	625 001	625 001
Medián (cps)	269 904	417 385	567 164	578 568	589 635
Diszperzió (%)	9,07	7,31	2,76	8,31	6,63

4.10. táblázat: A Jool NAT64 maximális kapcsolatkiakítási sebessége 4 000 000 kapcsolat mellett a CPU magok skálázódása szerint (BME)

Mérési környezet	StarBED						
CPU magok (db)	1	2	4	8	16	32	56
Kapcsolatok (db)	4 000 000						
Forrás portszám (db)	40 000						
Cél portszám (db)	100						
Minimum (cps)	121 578	156 732	161 592	206 305	297 945	410 534	496 876
Maximum (cps)	134 766	167 579	174 437	227 348	311 522	452 344	526 465
Medián (cps)	128 002	162 854	168 975	220 906	302 520	432 172	507 747
Diszperzió (%)	10,30	6,66	7,60	9,53	4,49	9,67	5,83

4.11. táblázat: A Jool NAT64 maximális kapcsolatkiakítási sebessége 4 000 000 kapcsolat mellett a CPU magok skálázódása szerint (StarBED)

A Jool NAT64 kapcsolatkiakítási méréseinél (ld. 4.10. táblázat és 4.11. táblázat), a CPU magszám skálázódás méréseknél azt tapasztaltam, a BME szerverein, hogy egy érdemleges javulás csak egy pontig látható, utána pedig csak marginális teljesítményjavulás tapasztalható. Ez a pont a BME szervereinél 4-8 mag környékén látható. A StarBED rendszerein pedig egy folyamatosan javuló tendencia látható.

A BME szerverein 10% a diszperzió mértéke, de még az elfogadható határon belül van. Ennek ellenére a fix órajel miatt a BME szerverein mért értékek stabilabbak, mint a StarBED rendszereié.

A StarBED rendszereinél mért diszperzió értéke nem elhanyagolható, de még mindig a normális határon belül mozog. Az implementáció nagyon jól skálázódott a magszám növelésével.

A BME szervereinek végleges teljesítménye még 16 mag mellett is nagyobb volt, mint a StarBED rendszereié 56 mag mellett, ennek valószínűleg a magasabb órajel az oka.

4.3.2 Átbocsátóképesség

Mérési környezet	BME			StarBED		
Kapcsolatok (db)	400 000	4 000 000	40 000 000	400 000	4 000 000	40 000 000
CPU magok (db)	16			56		
Forrás portszám (db)	40 000	40 000	40 000	40 000	40 000	40 000
Cél portszám (db)	10	100	1 000	10	100	1 000
Minimum (pps)	874 220	631 176	474 598	350 110	322 044	284 072
Maximum (pps)	918 752	675 002	505 378	389 998	371 102	318 760
Medián (pps)	875 923	637 058	489 044	381 356	341 258	299 446
Diszperzió (%)	5,08	6,88	6,29	10,46	14,38	11,58

4.12. táblázat: A Jool NAT64 átbocsátóképességé a maximális magszám mellett a kapcsolatok skálázódása szerint

A Jool átbocsátóképességénél mért kapcsolatszám-beli skálázódás (ld. 4.12. táblázat) hasonlóan alakult, mint a kapcsolatkiakítási sebességénél. Itt is hasonló romlást lehet tapasztalni a kapcsolattábla növelésének rezsiköltségével, mint az előző esetben.

A BME szerverein az előzőekhez hasonló teljesítménycsökkenést tapasztaltam, mint a kapcsolatfelépítési sebesség mérések esetében. A mért diszperzió értékek nagyobbak, mint a kapcsolatkiakításnál, de így is elfogadhatóak. A StarBED rendszerein látható, hogy sokkal gyengébb teljesítményt produkált, mint a BME szerverei. Ez egyrészt az alacsonyabb elvi maximális órajel mértéke, illetve az ingadozó órajel miatt van. Utóbbi könnyen alátámasztja az egyes diszperziók értékét.

A BME-vel ellentétben a StarBED esetében a magok száma miatt sokkal kisebb teljesítményromlást szenvedett az implementáció a kapcsolatok skálázásával, hiszen ebben az esetben 56 mag állt rendelkezésre. A BME szerverein mért teljesítmény, minden kapcsolattábla méret mellett sokkal magasabb volt, mint a StarBED rendszerein.

Mérési környezet	BME				
CPU magok (db)	1	2	4	8	16
Kapcsolatok (db)	4 000 000				
Forrás portszám (db)	40 000				
Cél portszám (db)	100				
Minimum (pps)	325 234	493 324	605 872	627 188	631 176
Maximum (pps)	337 624	515 716	635 002	653 112	675 002
Medián (pps)	327 858	495 015	635 002	647 369	637 058
Diszperzió (%)	3,78	4,52	4,78	4,00	6,88

4.13. táblázat: A Jool NAT64 átbecsítőképességé 4 000 000 kapcsolat mellett a CPU magok skálázódása szerint (BME)

Mérési környezet	StarBED						
CPU magok (db)	1	2	4	8	16	32	56
Kapcsolatok (db)	4 000 000						
Forrás portszám (db)	40 000						
Cél portszám (db)	100						
Minimum (pps)	74 380	103 438	137 458	189 384	278 440	308 054	322 044
Maximum (pps)	82 868	118 824	151 766	222 934	304 610	342 774	371 102
Medián (pps)	76 183	112 968	143 147	216 160	285 787	328 222	341 258
Diszperzió (%)	11,14	13,62	10,00	15,52	9,16	10,58	14,38

4.14. táblázat: A Jool NAT64 átbecsítőképességé 4 000 000 kapcsolat mellett a CPU magok skálázódása szerint (StarBED)

Az átbecsítőképesség alakulása a CPU magszámok skálázhatóságának tekintetében hasonlóan viselkedtek (ld. 4.13. táblázat és 4.14. táblázat), mint a kapcsolatkiakításnál. Itt is először több nagyobb teljesítmény ugrást tapasztaltam, viszont egy pont után már stagnált a teljesítmény.

A BME szerverein a fenti jelenséget ismét a 4-8 mag környékén tapasztaltam. A 16 mag esetén a medián alapján kicsit gyengébbnek tűnik az implementáció, viszont a minimum és maximum értékek alapján minimális javulás látható.

A StarBED esetében az átbecsátóképességek szépen skálázódtak, de 16 mag környékén lelassult a skálázódás és nem volt olyan nagy a mértéke, mint 1-16 mag esetében. A diszperzióértékek alapján hasonló eredmények születtek, mint a maximális kapcsolat kialakítás méréseknél.

Ebben az esetben is sokkal jobb eredményeket produkáltak a BME szerverei, mint a StarBED rendszerei, még a jelentős maghátrány ellenére is.

4.4 Az nftables eredményei

4.4.1 Maximális kapcsolat kialakítás

Mérési környezet	BME			StarBED		
Kapcsolatok (db)	400 000	4 000 000	40 000 000	400 000	4 000 000	40 000 000
CPU magok (db)	16			56		
Forrás portszám (db)	40 000	40 000	40 000	40 000	40 000	40 000
Cél portszám (db)	10	100	1 000	10	100	1 000
Minimum (cps)	2 411 934	2 142 073	2 037 374	1 847 349	1 752 318	1 675 024
Maximum (cps)	2 729 981	2 417 391	2 297 369	2 276 624	2 065 627	1 952 496
Medián (cps)	2 490 600	2 249 915	2 206 548	1 979 217	1 856 149	1 890 300
Diszperzió (%)	12,77	12,24	11,78	21,69	16,88	14,68

4.15. táblázat: Az nftables NAT44 maximális kapcsolat kialakítási sebessége a maximális magszám mellett a kapcsolatok skálázódása szerint

Az nftables implementációt NAT44 átjáróként alkalmaztam a mérések során. Ismét a DUT minden magját engedélyeztem a méréseknél. A kapcsolat kialakítási sebességek mérésénél a kapcsolatok számának skálázódása tekintetében (ld. 4.15. táblázat) azt fedeztem fel, hogy az nftables diszperziója kissé magas.

A BME szerverein a 400 000, 4 000 000 és a 40 000 000 kapcsolatú mérés között nem történt nagy teljesítménycsökkenés. Tehát ebben a mérési környezetben nem nagyon befolyásolta a kapcsolattábla menedzselése a teljesítményt.

A StarBED rendszereinél mért eredményeknek nagyobb volt a szórása, mint a BME-nél. Itt az figyelhető meg, hogy a kapcsolatok számának növelésével romlott a teljesítmény, de nem jelentős mértékben.

A BME szerverei jobban teljesítettek, de a StarBED rendszereinek a skálázódása kevesebb teljesítménybe került, mint a BME-nél.

Mérési környezet	BME				
CPU magok (db)	1	2	4	8	16
Kapcsolatok (db)	4 000 000				
Forrás portszám (db)	40 000				
Cél portszám (db)	100				
Minimum (cps)	465 677	1 001 748	1 951 170	2 125 837	2 142 073
Maximum (cps)	492 613	1 128 021	2 250 023	2 344 763	2 417 391
Medián (cps)	479 240	1 048 229	2 064 643	2 240 138	2 249 915
Diszperzió (%)	5,62	12,05	14,47	9,77	12,24

4.16. táblázat: Az nftables NAT44 maximális kapcsolatkialakítási sebessége 4 000 000 kapcsolat mellett a CPU magok skálázódása szerint (BME)

Mérési környezet	StarBED						
CPU magok (db)	1	2	4	8	16	32	56
Kapcsolatok (db)	4 000 000						
Forrás portszám (db)	40 000						
Cél portszám (db)	100						
Minimum (cps)	179 141	217 469	247 318	371 091	553 442	1 084 033	1 752 318
Maximum (cps)	188 471	259 869	289 674	424 999	658 749	1 262 434	2 065 627
Medián (cps)	185 616	248 104	277 303	405 101	608 274	1 211 731	1 856 149
Diszperzió (%)	5,03	17,09	15,27	13,31	17,31	14,72	16,88

4.17. táblázat: Az nftables NAT44 maximális kapcsolatkialakítási sebessége 4 000 000 kapcsolat mellett a CPU magok skálázódása szerint (StarBED)

Az nftables kapcsolatfelépítési teljesítménye, a CPU magok skálázódása mellett jó teljesítményt mutatott (ld. 4.16. táblázat és 4.17. táblázat).

A BME esetében szépen skálázódott az implementáció, azonban a 8-ról a 16 magra való növekedésnél csak minimálisan növekedett a teljesítmény. A diszperzió értékek a 4.15. táblázathoz hasonlóan alakultak. Az is megfigyelhető, hogy 1 mag használatánál volt a legkisebb a szórás. Ez logikus, hiszen, nem alakulhat ki versenyhelyzet egy magnál, így sokkal konzisztensebbek a mérések.

A StarBED esetében a CPU magok számának növelésével egyre jobb teljesítmény értékeket tapasztaltam. Az nftables nagyon jól skálázódott a StarBED rendszerein. Érdekes az, hogy 1-4 magig lassú volt a skálázódás, 4-16 magig felgyorsult, végül pedig 16-56 magig nagyon jól skálázódott.

Ebben az esetben is a BME-n mért eredmények voltak erősebbek. Mindkét esetben stabilok voltak a diszperzióértékek, ennek ellenére kicsit magasak voltak a mért adatok.

4.4.2 Átbocsátóképesség

Mérési környezet	BME			StarBED		
Kapcsolatok (db)	400 000	4 000 000	40 000 000	400 000	4 000 000	40 000 000
CPU magok (db)	16			56		
Forrás portszám (db)	40 000	40 000	40 000	40 000	40 000	40 000
Cél portszám (db)	10	100	1 000	10	100	1 000
Minimum (pps)	4 395 100	4 165 574	3 977 534	603 688	467 634	195 584
Maximum (pps)	4 780 074	4 724 294	4 549 870	794 394	574 998	224 382
Medián (pps)	4 555 333	4 396 503	4 263 745	737 329	559 826	219 335
Diszperzió (%)	8,45	12,71	13,42	25,86	19,18	13,13

4.18. táblázat: Az nftables NAT44 átbocsátóképessége a maximális magszám mellett a kapcsolatok skálázódása szerint

Az nftables átbocsátóképességénél, a kapcsolatok szerinti skálázódás méréseknél (ld. 4.18. táblázat) az nftables jó eredményeket produkált. Nyilván a kapcsolatok számának növelésével a teljesítmény is csökkent, viszont a diszperzió értéke stabil maradt a mérések során.

A BME szerverein csak minimális teljesítményromlással járt a kapcsolatok számának két nagyságrenddel való növelése.

A StarBED esetében jelentős romlással járt a kapcsolatok számának növelése, illetve a diszperzió értékek is magasak voltak. Ez alapján az nftables implementációt hátráltatja az ingadozó órajel.

A BME szerverein az nftables átbocsátóképessége CPU magok szerint (ld. 4.19. táblázat) 8 CPU magig, jó skálázódást mutatott. A diszperzió mértéke mérsékelten magas volt, viszont stabil maradt a mérések során.

A StarBED esetében (ld. 4.20. táblázat) a skálázódás lassan indult be, de a 32 és 56 mag esetében felgyorsult, hasonlóan, mint a maximális kapcsolat kialakítás esetében.

A BME szerverei sokkal jobban teljesítettek mindkét esetben, mint a StarBED rendszerei.

Mérési környezet	BME				
CPU magok (db)	1	2	4	8	16
Kapcsolatok (db)	4 000 000				
Forrás portszám (db)	40 000				
Cél portszám (db)	100				
Minimum (pps)	638 110	1 149 370	2 104 682	4 209 394	4 165 574
Maximum (pps)	750 002	1 279 848	2 475 380	4 709 858	4 724 294
Medián (pps)	664 237	1 220 897	2 342 467	4 385 862	4 396 503
Diszperzió (%)	16,85	10,69	15,83	11,41	12,71

4.19. táblázat: Az nftables NAT44 átbocsátóképessége 4 000 000 kapcsolat mellett a CPU magok skálázódása szerint

Mérési környezet	StarBED						
CPU magok (db)	1	2	4	8	16	32	56
Kapcsolatok (db)	4 000 000						
Forrás portszám (db)	40 000						
Cél portszám (db)	100						
Minimum (pps)	186 952	189 376	206 046	205 490	202 976	245 914	467 634
Maximum (pps)	211 822	216 868	258 656	243 752	245 008	328 338	574 998
Medián (pps)	199 387	204 835	214 051	214 258	214 646	261 889	559 826
Diszperzió (%)	12,47	13,42	24,58	17,86	19,58	31,47	19,18

4.20. táblázat: Az nftables NAT44 átbocsátóképessége 4 000 000 kapcsolat mellett a CPU magok skálázódása szerint

4.5 Az eredmények összegzése

Összességében minden technológia működött, viszont egyes technológiák teljesítménye nem bizonyult kellően magasnak. A NAT44 esetében a két vizsgált technológia, az nftables és a VPP NAT44 implementációja volt, ahol azt tapasztaltam, hogy a VPP alacsonyabb szórású technológia volt, mint az nftables, viszont magasabb magszám mellett az nftables jobban skálázódik és nagyobb teljesítményre képes, mint a VPP. A NAT64 esetében az egyik technológia a Jool, a másik pedig a VPP volt. A VPP ismét alacsonyabb szórású technológia volt, mint a Jool, viszont vállalhatatlanul alacsony teljesítményt nyújtott. A Jool teljesítménye messze magasabb volt, még kisebb magszám mellett is.

Kapcsolatkialakítás tekintetében a Jool jól skálázódik a kapcsolattábla méretét illetően, legalábbis csak egy közepes szintű teljesítmény romlás tapasztalható. Az nftables a kapcsolatok számának növelésével kisebb teljesítmény degradációt mutatott. A VPP a NAT44 esetében csak minimális teljesítmény csökkenést mutatott a kapcsolatszám növelésével, viszont a NAT64 esetében hatalmas volt a teljesítménycsökkenés.

Az átbocsátóképesség terén a Jool jól skálázódik a processzor magok tekintetében és a VPP, mind a NAT44 és NAT64 esetben, rossz skálázódást mutatott a „worker” szálak számának növelésével. Az nftables is jó skálázódást mutatott, viszont fontos megjegyezni, hogy magas szórással rendelkezett a nagy teljesítmény ellenére.

Összességében az mondható el, hogy a VPP egy ígéretes technológia, viszont a jelenlegi NAT pluginok állapota alapján még sokat kell fejlődnie ahhoz, hogy éles szolgáltatói hálózatban lehessen használni. A VPP rossz skálázódása részben azzal magyarázható, hogy minden egyes worker saját kapcsolattáblával rendelkezik, ezzel elkerülve a patt¹⁵ helyzetet. Ebből az következik, hogy egy kapcsolatnál, mind az előre és mind a vissza irányú forgalmat, ugyanazon a kapcsolattáblán kell feldolgozni. Tehát, ha egy olyan worker kezdené el feldolgozni a csomagot, aminek a kapcsolattáblájában nem szerepel a csomaghoz tartozó kapcsolat, de egy másik worker táblájában igen, akkor az első workernek át kell adnia a csomagot a másiknak, ami hatalmas rezsiköltséggel jár több csomag esetén [19].

Jelen állás szerint a VPP a legjobb és legstabilabb teljesítményét az egy dedikált worker használata mellett éri el, az nftables és a Jool pedig nagyjából pedig 8-16 mag után már nem skálázódik akkora jelentőséggel, mint előtte, így ezek az implementációk nagyjából akörüli magszámmal a leghatékonyabbak.

4.6 Alkalmazási területek

A 4.5. fejezet mérési összesítése alapján ebben a fejezetben különböző alkalmazási területeket sorolok fel, ahol az egyes implementációk bizonyulnak, az általam vizsgált implementációk közül, a leghatékonyabbnak.

Úgy vélem, hogy a VPP NAT44 variánsa, a rosszabb skálázódása miatt, olyan rendszerekben tud érvényesülni, ahol kevés CPU mag áll rendelkezésre, mivel kevés mag mellett messze nagyobb a teljesítménye, mint az nftablesnek. Ilyen rendszerek lehetnek olyan fogyasztói routerek, amelyekre könnyen lehet a gyártóétól eltérő operációs rendszert (pl.: OpenWrt) telepíteni.

¹⁵ deadlock

A VPP NAT64 variánsát a jelenlegi teljesítménye alapján nem ajánlom semmilyen rendszerben való használatra, mert a Jool teljesítőképessége nagyon hasonló alacsony kapcsolatszám mellett, nagy kapcsolatszám mellett, meg sokkal stabilabb és erősebb implementáció.

A Jool NAT64 implementációja egy megbízható és kellő erővel rendelkező szoftver, így éles szolgáltatói hálózatban jól megállja a helyét, ahol rengeteg CPU mag áll rendelkezésre. Kiemelném azt, hogy a skálázódása egy idő után csak stagnál és így pazarlóan bánhat az értékes magokkal.

Végül pedig az nftables NAT44 implementációként való használatát csak óvatosságra intve ajánlom, mivel nagyon magas a szórása az implementációnak, de cserébe sok CPU magos környezetben kiváló teljesítménye van. Nem ajánlom egy éles szolgáltatói hálózatba, viszont fogyasztói eszközökön kiváló választás.

5 Jövőbeli tervek

A jövőben szeretnék angol nyelvű nemzetközi folyóiratcikket írni a kutatásaimból, ahol érdekes lehet a VPP memória használata más technológiákhoz képest, illetve az implementáció késleltetése is csomagkésleltetési ingadozása is. Különös hangsúlyt szeretnék fektetni a letöltési és feltöltési sebességekre, mert azok nem biztos, hogy szimmetrikusak az egyes implementációknál.

Továbbá fontos még a VPP teljesítményét megvizsgálni, ha nem teljesen telített hálózati forgalomról, hanem más kihasználtságról beszélünk, hiszen ilyenkor problémát okozhat az implementáció nagyobb késleltetése.

Érdekes kutatási terület lenne, ha nem csak szerver, hanem felhasználói minőségű hardveren is lemérhetném a VPP teljesítményét más implementációkhoz képest.

6 Köszönetnyilvánítás

Szeretném megköszönni a konzulensemnek, Dr. Lencse Gábornak, hogy iránymutatásaival és értékes tanácsaival segített a munkafolyamat minden szakaszában. Köszönöm a VPP NAT44 és NAT64 plugin egyik fejlesztőjének, [Ole Trøan](#)nak, hogy a VPP-vel kapcsolatos kérdéseimre hamar és pontosan válaszolt, így közvetlenül segítve a dolgozat elkészítésében.

Irodalomjegyzék

- [1] fd.io, „*VPP/what is VPP?*” (2017. május 26.) https://wiki.fd.io/view/VPP/What_is_VPP%3F (Elérés időpontja: 2024. március 15.)
- [2] Srisuresh, P. and K. Egevang, „*Traditional IP Network Address Translator (Traditional NAT)*”, RFC 3022, DOI 10.17487/RFC3022, 2001. január, <<https://www.rfc-editor.org/info/rfc3022>>.
- [3] DPDK, „*DPDK documentation*” (2024. március 13.) <https://doc.dpdk.org/guides/index.html> (Elérés időpontja: 2024. március 16.)
- [4] G. Lencse, „*Design and Implementation of a Software Tester for Benchmarking Stateless NAT64 Gateways*”, IEICE Transactions on Communications, vol. E104-B, no.2, pp. 128-140., 2021. február 1., DOI: 10.1587/transcom.2019EBN0010
- [5] The kernel development community, „*Scaling in the Linux Networking Stack*” (2023. december 14.) <https://docs.kernel.org/networking/scaling.html> (Elérés időpontja: 2024. március 16.)
- [6] G. Lencse, and Á. Bazsó, „*Benchmarking methodology for IPv4aaS technologies: Comparison of the scalability of the Jool implementation of 464XLAT and MAP-T*”, Computer Communications, vol. 219, April 2024, Pages 243-258, DOI: 10.1016/j.comcom.2024.03.007
- [7] Georgescu, M., Pislariu, L., and G. Lencse, „*Benchmarking Methodology for IPv6 Transition Technologies*”, RFC 8219, DOI 10.17487/RFC8219, 2017. augusztus, <<https://www.rfc-editor.org/info/rfc8219>>.
- [8] G. Lencse and K. Shima, „*Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814 Pseudorandom Port Numbers*”, Internet-Draft, 2024. január, <<https://datatracker.ietf.org/doc/html/draft-ietf-bmwg-benchmarking-stateful>>
- [9] G. Lencse, „*Design and implementation of a software tester for benchmarking stateful NATxy gateways: Theory and practice of extending siitperf for stateful tests.*”, Computer Communications, 2022. május 28., Online ISSN 1745-1345, Print ISSN 0140-3664, DOI 10.1016/j.comcom.2022.05.028
- [10] nftables wiki, „*What is nftables?*”, (2021. április 2.) https://wiki.nftables.org/wiki-nftables/index.php/What_is_nftables%3F (Elérés időpontja: 2024. március 17.)
- [11] G. Lencse and Y. Kadobayashi, „*Comprehensive Survey of IPv6 Transition Technologies: A Subjective Classification for Security Analysis*”, IEICE Transactions on Communications, vol. E102-B, no.10, pp. 2021-2035. DOI: 10.1587/transcom.2018EBR0002
- [12] Lencse, G., Palet Martinez, J., Howard, L., Patterson, R., and I. Farrer, „*Pros and Cons of IPv6 Transition Technologies for IPv4-as-a-Service (IPv4aaS)*”, RFC 9313, DOI 10.17487/RFC9313, 2022 október, <<https://www.rfc-editor.org/info/rfc9313>>.

- [13] NIC México, „*Introduction to Jool*” (2024) <https://nicmx.github.io/Jool/en/intro-jool.html> (Elérés időpontja: 2024. március 21.)
- [14] StarBED5 Project website „*What's StarBED5*” <https://starbed.nict.go.jp/en/aboutus/index.html> (Elérés időpontja: 2024. április 8.)
- [15] Bradner, S. and McQuaid, J., „*Benchmarking Methodology for Network Interconnect Devices*”, RFC 2544, DOI 10.17487/RFC2544, 1999 március, <<https://www.rfc-editor.org/info/rfc2544>>.
- [16] Kosák, M., „*IP csomagtovábbító rendszerek teljesítményének összehasonlító elemzése*”, Tudományos Diákköri Konferencia dolgozat, Széchenyi István Egyetem, Győr, 2024
- [17] Lencse, G., and Shima, K. (2023) „*Optimizing the Performance of the Iptables Stateful NAT44 Solution*”, Infocommunications Journal, <https://www.infocommunications.hu/documents/169298/4861545/>
- [18] Vyacheslav Gapon, „*Tuning nf_conntrack*”, (2019. június 26.) https://ixnfo.com/en/tuning-nf_conntrack.html (Elérés időpontja: 2024. május 8.)
- [19] Ole Troan, „*Why NAT needs handoff?*”, <https://lists.fd.io/g/vpp-dev/message/24831> (Elérés időpontja: 2024. október 23.)

Függelék

A.1 A siitperf szkriptjei és konfigurációja

Mivel a siitperf a DPDK keretrendszert használja, a következő szkripttel engedélyeztem a TESZTER gépen az interfészekhez a megfelelő illesztőprogram használatát:

```
#!/bin/bash
modprobe uio_pci_generic
dpdk-devbind.py --bind=uio_pci_generic 0000:31:00.0
dpdk-devbind.py --bind=uio_pci_generic 0000:31:00.1
```

Ezek a szkriptek a StarBED rendszerre vonatkoznak. A BME rendszerek szkriptjei is hasonlóak voltak, csak eltérő MAC címeket és interfész neveket használtak.

Az egyes interfész IP-címeket és MAC-címeket a siitperf.conf fájlban a következőkre állítottam:

```
IPv4-L-Real 198.18.0.2
IPv4-R-Real 198.19.0.2

MAC-L-Tester B4:96:91:BE:B6:24 # x035 bus 31.0
MAC-R-Tester B4:96:91:BE:B6:25 # x035 bus 31.1
MAC-L-DUT B4:96:91:BA:11:46 # x036 bus 31.0
MAC-R-DUT B4:96:91:BA:11:47 # x036 bus 31.1
```

Látható, hogy a NAT44 esetben ezek a címek az RFC-nek megfelelően lettek megválasztva. A MAC-címek esetében x035 volt a TESZTER és x036 volt a DUT gép, és minden MAC-cím egy-egy hálózati interfész MAC-címeként értendő.

A DUT gépen a következő szkripttel engedélyeztem a csomagtovábbítást és az RSS megfelelő beállítását erre csak az nftables és a Jool esetében volt szükségem, mert a VPP nem kernel szintű interfész illesztőprogramokat használ:

```
#!/bin/bash
#enable ipv4 routing
sysctl -w net.ipv4.ip_forward=1
#enable ipv6 routing
sysctl -w net.ipv6.conf.all.forwarding=1

#rx-flow-hash
ethtool -N enp49s0f0 rx-flow-hash udp4 sdfn
ethtool -N enp49s0f0 rx-flow-hash udp6 sdfn
ethtool -N enp49s0f1 rx-flow-hash udp4 sdfn
ethtool -N enp49s0f1 rx-flow-hash udp6 sdfn
```

A.2 A Jool szkriptjei

Ezek a szkriptek a StarBED rendszerre vonatkoznak. A BME rendszerek szkriptjei is hasonlóak voltak, csak eltérő MAC címeket és interfész neveket használtak.

A következő szkript állítja be az interfészek megfelelő IP-címeit, illetve az ARP és NDP táblák bejegyzéseit is:

```
#!/bin/bash
# ip addr for enp49s0f0
ip addr add 198.18.0.1/24 dev enp49s0f0
ip addr add 2001:2::1/64 dev enp49s0f0

# ip addr for enp49s0f1
ip addr add 198.19.0.1/24 dev enp49s0f1
ip addr add 2001:2:0:8000::1/64 dev enp49s0f1

# arp table fix
ip neighbor add 198.18.0.2 lladdr B4:96:91:BE:B6:24 dev enp49s0f0 nud permanent
ip neighbor add 198.19.0.2 lladdr B4:96:91:BE:B6:25 dev enp49s0f1 nud permanent

# add NDP table entries
ip neighbor add 2001:2::1 lladdr B4:96:91:BE:B6:24 dev enp49s0f0 nud permanent
ip neighbor add 2001:2:0:8000::1 lladdr B4:96:91:BE:B6:25 dev enp49s0f1 nud permanent
```

A set-jool szkript a következőket tartalmazza [13]:

```
modprobe jool
jool instance add --netfilter --pool6 64:ff9b::/96
jool pool4 add 198.19.0.1 --udp 1-65535
```

A del-jool szkript tartalma:

```
modprobe -r jool
```

A.3 A VPP szkriptjei

Ezek a szkriptek a StarBED rendszerre vonatkoznak. A BME rendszerek szkriptjei is hasonlóak voltak, csak eltérő MAC címeket és interfész neveket használtak.

A VPP futtatása előtt a következő szkriptet futtattam, hogy az interfészek a megfelelő DPDK illesztőprogramokat használják:

```
#!/bin/bash
modprobe uio_pci_generic
dpdk-devbind.py --bind=uio_pci_generic 0000:31:00.0
dpdk-devbind.py --bind=uio_pci_generic 0000:31:00.1
```

A NAT44 beállító-szkript egyetlen paramétert várt, a kapcsolattábla méretét. A set-vpp szkript tartalma a következő :

```
#!/bin/bash

systemctl start vpp

sleep 20

vppctl nat44 plugin enable sessions $1
vppctl set interface ip address TwentyFiveGigabitEthernet31/0/0 198.18.0.1/24
vppctl set interface ip address TwentyFiveGigabitEthernet31/0/1 198.19.0.1/24
vppctl set interface state TwentyFiveGigabitEthernet31/0/0 up
vppctl set interface state TwentyFiveGigabitEthernet31/0/1 up
vppctl set interface nat44 out TwentyFiveGigabitEthernet31/0/1 output-feature
vppctl set ip neighbor TwentyFiveGigabitEthernet31/0/0 198.18.0.2 B4:96:91:BE:B6:24
vppctl set ip neighbor TwentyFiveGigabitEthernet31/0/1 198.19.0.2 B4:96:91:BE:B6:25
vppctl nat44 add interface address TwentyFiveGigabitEthernet31/0/1
set nat timeout udp 20000

sleep 10
```

Az „alvásokra” azért van szükség, mert a vpp-nek kell egy kis idő, mire „magához tér” és képes csomagoka továbbítani.

A NAT64 szkript az előzőhöz hasonlóan egyetlen paramétert várt, a kapcsolattábla méretét, így a következőképpen néz ki:

```
systemctl start vpp

sleep 20

vppctl nat64 plugin enable st-buckets $1
vppctl set interface state TenGigabitEthernet1/0/0 up
vppctl set interface state TenGigabitEthernet1/0/1 up
vppctl set interface ip address TenGigabitEthernet1/0/0 2001:2::1/64
vppctl set interface ip address TenGigabitEthernet1/0/1 198.19.0.1/24
vppctl ip route add 2001:2::/64 via 2001:2::1 TwentyFiveGigabitEthernet31/0/0
vppctl ip route add 198.19.0.0/24 via 198.19.0.1 TwentyFiveGigabitEthernet31/0/1
vppctl set ip neighbor static TwentyFiveGigabitEthernet31/0/0 2001:2::2
b8:ca:3a:62:3c:40
vppctl set ip neighbor static TwentyFiveGigabitEthernet31/0/1 198.19.0.2
b8:ca:3a:62:3c:42
```

```
vppctl nat64 add prefix 64:ff9b::/96
vppctl nat64 add pool address 198.19.0.1
vppctl nat64 add interface address TwentyFiveGigabitEthernet31/0/1
vppctl nat64 add interface address TwentyFiveGigabitEthernet31/0/0
vppctl set interface nat64 in TwentyFiveGigabitEthernet31/0/0
vppctl set interface nat64 out TwentyFiveGigabitEthernet31/0/1
vppctl set nat timeout udp 20000
```

A del-vpp szkript tartalma a következő:

```
#!/bin/bash

systemctl stop vpp
```

A VPP konfigurációs fájljának releváns része a következő volt:

```
heapsize 50G
cpu {
    main-core 0

    corelist-workers 4
}
```

Itt a main és a worker szálakat adtam meg. A konfiguráció többi része az alapértelmezett maradt.

A.4 Az nftables szkriptjei

Ezek a szkriptek a StarBED rendszerre vonatkoznak. A BME rendszerek szkriptjei is hasonlóak voltak, csak eltérő MAC címeket és interfész neveket használtak.

A következő szkript állítja be Az interfészek megfelelő IP-címeit, illetve az ARP és NDP táblák bejegyzéseit is:

```
#!/bin/bash
# ip addr for enp49s0f0
ip addr add 198.18.0.1/24 dev enp49s0f0

# ip addr for enp49s0f1
ip addr add 198.19.0.1/24 dev enp49s0f1

# arp table fix
ip neighbor add 198.18.0.2 lladdr B4:96:91:BE:B6:24 dev enp49s0f0 nud permanent
ip neighbor add 198.19.0.2 lladdr B4:96:91:BE:B6:25 dev enp49s0f1 nud permanent
```

A szkript hasonló, mint a Jool esetében, hiszen az nftables is kernel alapú csomagmegszakítással működik.

A set-nftables szkript a következőket tartalmazza:

```
#!/bin/bash
SRCIP='198.18.0.0/24'
IFO='eno12409'

#add nat table
nft add table nat
nft 'add chain nat postrouting {type nat hook postrouting priority 100 ;}'
nft add rule nat postrouting ip saddr $SRCIP oif $IFO masquerade

sysctl -w net.netfilter.nf_conntrack_udp_timeout=20000
```

A del-nftables szkript tartalma:

```
#!/bin/bash
nft flush table nat
```

A.5 A VPP NAT64 plugin hibája

A dolgozat megírása során azt fedeztem fel, hogy a VPP NAT64 plugin egy súlyos hibával rendelkezik. Amikor 40 000 db forrásporttal szerettem volna elvégezni a méréseket, akkor a VPP az állapottáblájának csak a negyedét töltötte fel bejegyzésekkel az előzetes fázisban, így nem tudtam végrehajtani a méréseket.

A kapcsolattábla hosszadalmas tanulmányozása után fedeztem fel a korrelációt a bejegyzések hiánya és a forrás portszámok értékei között. Az hiba megtalálása után úgy döntöttem, hogy csak 4 000 db forrásportot fogok használni és a skálázódást 100, 1 000, illetve 10 000 célportszámmal oldom meg, hogy elérjem a 400 000, 4 000 000 és 40 000 000 kapcsolattal rendelkező táblaméreteket. Azért ezt az utat választottam, mert a VPP dokumentációjában, a NAT64 plugin beállításainál nem találtam a portszám értékekkel kapcsolatos beállítást.