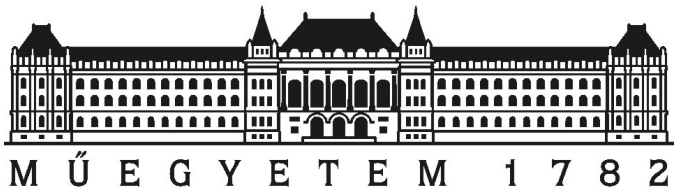


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Terhelés áthelyezés P4 hálózati kapcsolókról NFV-be programozható adatsíkokon

Makara László Árpád
G5YPX8

Konzulens:
Dr. Pekár Adrián

2021. október 28.

Kivonat

A P4 a hálózati eszközök hardveres (nagyobb csomagfeldolgozási teljesítmény a hardveralapú hálózati megvalósítások miatt) és szoftveres (a hálózati műveletek szoftveres programozhatóságának rugalmassága) megvalósításainak előnyeit igyekszik ötvözni. Bár a hálózati P4 funkciók rugalmasan futtathatók dedikált hardveren és nagy teljesítményű ASIC-ken, nem skálázhatók virtualizált módon, mivel rögzített számítási erőforrásokkal rendelkeznek. E tekintetben megvizsgálom a P4 és a hálózati funkciók virtualizálásának (NFV) integrációját. Konkrétabban, analitikus modellt dolgozok ki az NFV-re terhelésáthelyezési képességgel kibővített P4 hálózati kapcsolóhoz, amelynek célja a skálázhatósági problémák és a potenciális teljesítményszűk keresztmetszetek kezelése. Teljesítményelemzésre van szükség az ilyen konszolidáció lehetséges kompromisszumainak azonosításához és megértéséhez. Ennek érdekében a tehermentesítési arányt formálisan optimalizálási problémaként fogalmaztam meg és a Brent-módszert alkalmazásával igyekeztem megtalálni az optimális tehermentesítési hányadot. Megvizsgáltam a P4-ről az NFV-re történő terhelésáthelyezés hatását a fizikai hálózati funkció (PNF), a virtualizált hálózati funkció (VNF) és a hálózati kapcsoló számítási kapacitásai szempontjából, illetve a P4 hálózati kapcsoló és a VNF közötti átlagos késleltetést, valamint a table-miss valószínűségét. Elemzésemből kiderül, hogy a P4-ről az NFV-re történő terhelésáthelyezés előnye jelentős. Konkrétan a dolgozatban bemutatott terhelésáthelyezéses architektúra bizonyítottan minden vizsgált esetben optimálisabban teljesít, mint a tehermentesítés nélküli. Azt is megfigyeltem, hogy a rendszer átlagos késleltetését több mint a felére redukálja a terhelésáthelyezés. Az ebben a munkában tett megfigyelések segíthetnek a P4 és az NFV konszolidáció megfelelő konfigurációjának kiválasztásában bizonyos felhasználási esetekhez.

Abstract

P4 seeks to combine the advantages of hardware (higher packet processing performance due to hardware-based networking implementations) and software (the flexibility of software-based programmability of network operations) instantiations of networking devices. However, although network P4 functions can be flexibly run on dedicated hardware and high-performing ASICs, they cannot be scaled in a virtualized fashion as they have fixed computing resources. In this regard, we examine the integration of P4 and network function virtualization (NFV). Specifically, we develop an analytical model for a P4 switch extended with offloading capability to NFV, aiming to address scalability issues and potential performance bottlenecks. Performance analysis is needed to identify and understand potential trad-offs of such consolidation. To this end, we formally formulated the determination of the offloading ratio as an optimization problem and used Brent's method to find the optimal offloading ratio quickly. We examined the impact of offloading from P4 to NFV from the perspectives of the capacities of the physical network function, virtualized network function (VNF), and switch computing, the delay between the P4 switch and VNF, and the table miss probability. Our analysis reveals that the advantage of offloading from P4 to NFV is significant. Specifically, the consolidated operation has been proven to perform more optimally than without offloading in all the examined cases. We also observed that offloading helps to reduce the average delay of the system by over a factor of two. The observations made in this work can help selecting the proper configuration of P4 and NFV consolidation for a specific use case.

Köszönetnyilvánítás

Ezúton szeretném megköszönni témavezetőmnek, Dr. Pekár Adriánnak a kutatásom során nyújtotta idejét, támogatását, illetve szakmai és konstruktív megjegyzéseit.

Hálás vagyok továbbá a kutatásomhoz fűzött értékes szakmai megjegyzésekért külföldi mentoraimnak, névszerint Yuan-Cheng Lai-nak, a Nemzeti Tajvani Tudományos és Technológiai Egyetem professzorának, Ying-Dar Lin-nek, a Nemzeti Yang Ming Chiao Tung Egyetem kutatóprofesszorának, valamint Winston K.G. Seah-nak, a Wellingtoni Victoria Egyetem professzorának.

Különleges köszönet illeti továbbá Edinát, Tuszit és Danit. A munkámhoz nyújtott segítségetek nélkülözhetetlen volt.

Nem utolsó sorban, köszönet illeti családom többi tagját is az évek során nyújtotta támogatásukért és a kiegyensúlyozott családi háttér biztosításáért. Köszönöm, hogy átsegítettetek az igazán nehéz időszakokon!

Tartalomjegyzék

Táblázatok jegyzéke

Ábrák jegyzéke

Rövidítések jegyzéke

4DMC 4 Dimensional Markov Chain

API Application Programming Interface

ASIC Application-Specific Integrated Circuit

BMv2 Behavioral Model v2

C Controller

CPU Central Processing Unit

ETSI European Telecommunications Standards Institute

FPGA Field-Programmable Gate Array

FTE Flow Table Entries

HW Hardware

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IRTF Internet Research Task Force

ISP Internet Service Provider

ITU International Telecommunication Union

M/H_2/1 Markovian/2-Hyperexponential/1

M/G/1 Markovian/General/1

M/Geo/1 Markovian/Geometric/1

M/M/1 Markovian/Markoivan/1

M/M/n Markovian/Markovian/n

M^x/M/1 Markovian x batch size/Markovian/1

MANO Management, Automation, and Orchestration

MMPP/M/1 Markov Modulated Poisson Process/ Markovian/1

MMPP/M/1/k Markov Modulated Poission Process/Markovian/1/k

NF Network Function

NFV Network Function Virtualization

NFVI Network Functions Virtualization Infrastructure

NPU Network Processing Unit

P4 Programming Protocol-independent Packet Processors

P4-to-VHDL Programming Protocol-independent Packet Processors-to-VHSIC Hardware Description Language

P4FPGA Programming Protocol-independent Packet Processor Field-Programmable Gate Array

PISA Protocol Independent Switching Architecture

PISCES Programmable, Protocol-Independent Software Switch

PNF Physical Network Function

PQ Priority-Queue

RAM Random Access Memory

RX Receiver

SC Switch Communication

SDN Software Defined Networking

SmartNIC Smart Network Interface Card

SP Switch Processing

SW Software

Tbps Terabytes Per Second

TCAM Ternary Content-Addressable Memory

TX Transmit

VNF Virtual Network Function

WAN Wide Area Network

Bevezetés

A Software-Defined Networking (SDN) [1] és Network Functions Virtualization (NFV) [2] technológiáknak köszönhetően számos innováció zajlott le a hagyományos értelemben vett hálózatépítésben, mindamellett számottevő kutatási munka zajlik ezen tematikán belül. Az SDN elválasztja a hálózati vezérlési funkciókat a hálózati továbbítási funkcióktól azáltal, hogy absztrahálja a fizikai hálózati erőforrásokat (pl. kapcsolók és útvonalválasztók), és a döntéshozatalt egy (jellemzően központosított) virtuális hálózati vezérlési síkba [3] helyezi át. A szoftverizált vezérlési sík határozza meg, hogy az eszköz hova küldje az adott forgalmat, míg a hálózati eszközök egyszerű továbbítókká válnak, amelyek irányítják és kezelik a forgalmat. Az NFV ezzel szemben leválasztja a hálózati funkciókat a (szabadalmaztatott) hardvereszközökről (pl. útválasztók és tűzfalak) és egyenértékű hálózati funkciókat biztosít speciális hardverek nélkül. Az SDN és az NFV tehát olyan paradigmáknak tekinthető, amelyekben az SDN a vezérlési síkot szoftverizálja, az NFV pedig az adatsíkot virtualizálja.

A hagyományos értelemben vett hálózatoknak skálázhatósági, üzemeltetési és menedzsment kihívásokkal kell szembenézniük, mivel az általuk szállított forgalom mennyisége és heterogenitása töretlenül növekszik a digitalizáció előrehaladtával. Ezeket a korlátokat a hálózatépítésben a szoftverizáció és a virtualizáció bevezetésével sikerült enyhíteni, amelyek lehetővé tették a hálózatüzemeltetők számára, hogy hatékonyabban tervezzék, valósítsák meg és kezeljék hálózataikat [4], de a hálózati eszközök továbbra is rögzített, jól ismert viselkedéssel rendelkeznek, amelyet a gyártók közvetlenül a chipekbe égettek. Ennek a hardverközpontú hálózatépítésnek az egyik fő jellemzője az volt, hogy a továbbítási logika rugalmas megváltoztatására csak korlátozottan volt lehetőség. Emellett az új protokollok és egyéb hálózati funkciók (például az adatútvonal-vezérlés és analitikai mérések) támogatását sem lehetett hozzáadni.

Az azonos (és nagyobb) teljesítményt nyújtó, programozható chipekkel ellátott eszközök, mint például a protokollfüggetlen kapcsolóarchitektúra (PISA) piacra kerülése azonban új korszakot nyitott a hálózatépítésben, ami újítási és fejlesztési lehetőségeket teremtett. A gyors és megfizethető programozható chipekkel a hálózat programozhatósága megvalósíthatóvá vált. A protokollfüggetlen csomagprocesszorok (P4) programozása a hálózati programozhatóságot elősegítő kulcsfontosságú paradigma. Lehetővé teszi, hogy a továbbítási viselkedést egy tartományspecifikus nyelvvel, a P4-gyel fejezzük ki és ezt követően egy olyan programba fordítsuk, amely betölthető és futtatható egy céleszközben (*i.i.*, hálózati eszközben) [5]. Ennek következtében a P4 a számítógép-hálózatok teljesítményének és rugalmasságának hatékony javítási módjaként jelent meg, és egyre nagyobb figyelmet kap a hálózati közösségben.

A P4 hálózati kapcsolók számos előnnyel rendelkezhetnek NFV szempontjából, hiszen lehetővé teszik a hardverben és szoftverben egyszerre működő hálózati funkciók (NF) kezelését. Mivel a P4 protokollfüggetlen, ezért lehetővé teszi az egyéni kapcsolási pipeline-ok megvalósítását és API-t biztosít az SDN-vezérlő számára az útvonalválasztó táblák feltöltéséhez. Továbbá a P4 kapcsolók futásidejű átkonfigurálhatósága segíti a hálózati funkciók rugalmas működését a hálózat dinamikája [6] miatt. A kizárólag PNF-ként vagy VNF-ként való működtetés azonban költségekkel jár. A PNF alapjául szolgáló fix számítási erőforrás (bare-metal programozható eszközök) nem skálázható virtualizált módon (*i.*, több erőforrás hozzáadása a poolhoz). Másrészt a VNF-et jellemzően hipervizorok alatt telepítik és általános célú, nem kifejezetten számításigényes feladatokra tervezett CPU-kkal rendelkező általános célú hardveren üzemeltetik. Ez a P4 switch szuboptimális működését eredményezi, ami elkerülhetetlenül skálázhatósági és teljesítménybeli kihívásokhoz vezet. Mindezek után úgy tűnik, hogy radikális előrelépést jelenthet a megfelelő tehermentesítési arány meghatározása.

Annak ellenére, hogy a hálózati P4 funkciók rugalmasan futtathatók dedikált hardveren és nagy teljesítményű ASIC-ken, virtualizált módon nem skálázhatók (azaz nem adhatók hozzá további erőforrások a rendszerhez), mivel fix kapacitással rendelkeznek, ezért a P4 és az NFV kombinációja elkerülhetetlen a hardveres erőforráskorlátok leküzdéséhez. E tekintetben megvizsgálom a P4 és az NFV integrációját és kifejlesztek egy M/M/1 sorbanállási modellt, amelyet kibővítek az NFV-re való tehermentesítési képességgel. A kifejlesztett modellt a rendszer teljesítményének elemzésére használom, aminek célja a skálázhatósági problémák és a teljesítményszűk keresztmetszetek kezelése. Különösen arra keresem a választ, hogyan kell a rendszert konfigurálni az optimális teljesítmény elérése érdekében. Ennek érdekében megvizsgálom a különböző paraméterkonfigurációk hatását a hálózati funkciókra, beleértve a hálózati kapcsoló feldolgozást, a hálózati kapcsoló kommunikációt és a késleltetést.

Ez a dolgozat az első, amely a P4 és az NFV konszolidációját modellezi. Míg a közelmúltban számos módszert fejlesztettek ki a szoftveres és hardveres hálózati eszközök egymást kiegészítő működésének modellezésére és elemzésére [7], [8], nem javasoltak modellt olyan forgatókönyvekre, ahol az NFV kiegészíti a P4 hálózati kapcsoló működését. Ebben a tanulmányban tett megállapításaim értékesek lehetnek az internetszolgáltatók (ISP), a hardvergyártók és a chipgyártók számára, ugyanakkor segíthet a kapcsolótervezőknek és a hálózati elemzőknek a teljesítménymérések meghatározásában és az egyéni kapcsolási csővezetékek (pipeline) jobb tervezésében. Továbbá segítséget nyújthat a P4 és az NFV integrált telepítésével kapcsolatos előnyök és kompromisszumok meghatározásában. Az eredmények iránymutatásként is szolgálhatnak a jobb hálózattervezéshez, az architekturális döntésekhez és a telepítési választásokhoz.

A dolgozat újdonsága a következőkben rejlik: (*i*) Analitikus modellt határoz meg a P4 teljesítmény és forgalmi jellemzőinek elemzésére és az NFV-re való terhelésáthelyezési képességekkel rendelkező P4 teljesítményének és forgalmi jellemzőinek vizsgálatára. (*ii*) Paraméteres érzékenységi elemzést nyújt, hogy segítse a hálózattervezőket a hálózati teljesítményt befolyásoló kritikus tényezők azonosításában.

A dolgozat további része a következőképpen szerveződik. Az 1. fejezet rövid elméleti hát-

teret nyújt, beleértve az SDN-t, az NFV-t és a P4-et, valamint tárgyalja a releváns járulékos munkákat. A 2. fejezet leírja az NFV-vel integrált P4 switch működését absztraháló modellt, amelyre késleltetés elemzést is nyújt. A 3. fejezet bemutatja az optimalizálási probléma leírását és a megoldásához használt algoritmusokat. Ezután a 4. fejezetben analitikus eredményeket mutatok be. Dolgozatom végén összegzem az elért eredményeimet és megadom a jövőbeli kutatásaim tervezett irányát.

1. fejezet

Kutatási háttér és kapcsolódó munkák

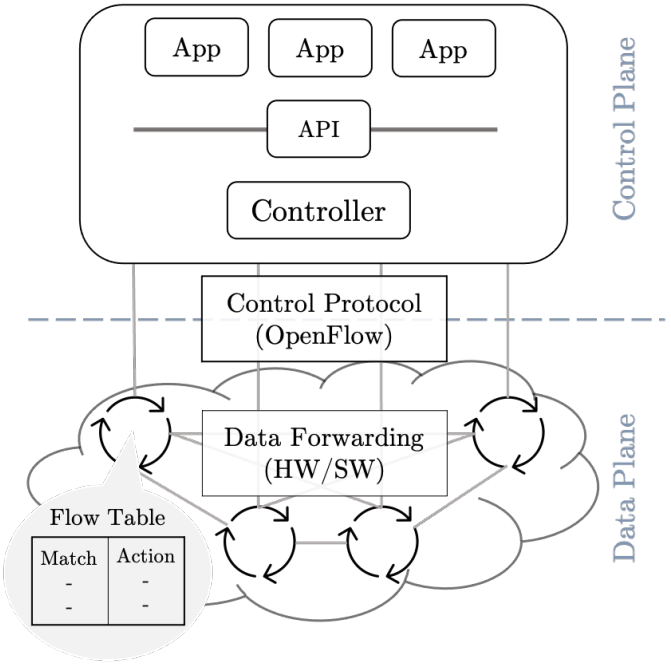
A skálázhatóság és flexibilitás állandó kihívást jelentett a hagyományos hálózatok számára [9], ezeket a korlátokat a szoftverizáció és a virtualizáció bevezetésével enyhítették a hálózatépítésben, ami elősegítette a hagyományos hálózatok átalakulását. Ennek eredményeképpen olyan új paradigmákat fejlesztettek ki, mint az SDN, az NFV és a P4, amik a számítógépes hálózatok teljesítményének, skálázhatóságának és kezelhetőségének javítására szolgáló hatékony módszerekként jelentek meg. A következőkben először röviden bemutatom ezeket a paradigmákat a jelen dolgozat szempontjából releváns kontextusban, amit ezután a kardinálisan kapcsolódó munkák megvitatásával folytatok.

1.1. SDN

A hagyományos hálózatépítés és az SDN közötti legjelentősebb különbség talán az, hogy az SDN szoftveralapú, míg a hagyományos hálózatépítés jellemzően hardveralapú. Történelmileg a hardverkonstrukcióként megvalósított hálózati funkciók gyorsan működtek, így a hagyományos hálózatépítés fix funkciójú hálózati eszközökön alapult. Ennek a konstrukciónak azonban megvannak a maga hátulütői, hiszen az eszközök jól működnek a saját fejlesztésű szoftverekkel, viszont ezek a szoftverek nem módosíthatók tetszőlegesen, csakis kizárólag a kibocsátó gyártó tudja megtenni a szükséges korrigálásokat. Következésképpen a flexibilitás állandó kihívást jelentett a hagyományos hálózatok számára [9].

Az SDN [10], [11] egy olyan hálózati paradigma, amely lehetővé teszi a programozott irányítást és vezérlést azáltal, hogy a hálózati konfigurációt és forgalomtervezést elválasztja az alapvető hardveres infrastruktúrától. Az SDN virtualizálja a hardvert azáltal, hogy a hálózatot két külön síkra bontja, az adatsíkra és a vezérlési síkra. Az adatsík továbbítja a forgalmat, azonban mielőtt a forgalom eléri az adatsíkot, a vezérlő sík határozza meg, hogy milyen útvonal(ak)on haladjanak az adatcsomagok.

Az 1.1 ábra az SDN paradigma általános felépítését mutatja be, ahol minden hálózati eszköz legalább egy útvonalválasztási táblát tart fent, amely útvonalválasztási bejegyzéseket tartalmaz. A csomagok megvizsgálása az útvonalválasztási bejegyzések megfeleltetési prioritása alapján



1.1. ábra. Általános SDN-architektúra.

történik. A bejövő interfészre érkező hálózati adatfolyamok — amelyek megfelelő Flow Table Entries (FTE)-vel rendelkeznek — továbbításra kerülnek a hálózati kapcsoló kimenő interfészére. A nem megfelelő FTE-vel rendelkező csomagok azonban a kontrollerhez kerülnek továbbításra, amely meghatározza a végrehajtandó műveleteket. Ennek eredményeképpen a kontroller frissíti a hálózati kapcsoló útvonalválasztási tábláját, ami ezután az útvonalválasztásokat a szokásos módon kezeli, mivel a megfelelő FTE már telepítve lett az eszközre. Az SDN kontrollerek és a hálózati kapcsolók síkja közötti kommunikáció egy nyílt szabványon keresztül történik, ilyen például az OpenFlow protokoll [12]. Megjegyzendő azonban, hogy az OpenFlow nem irányítja a hálózati kapcsoló viselkedését csak egy módot biztosít az útvonalválasztási táblák tapasztalati úton való feltöltésére (arra a tényre támaszkodik, hogy a hálózati kapcsolók chipjei nem programozhatóak).

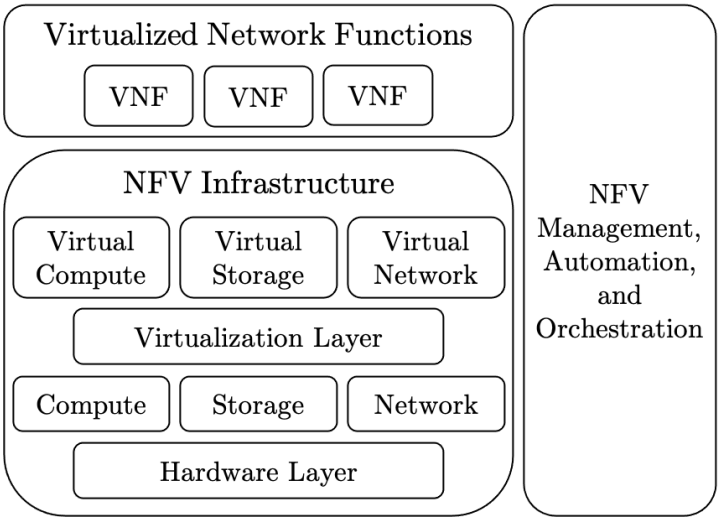
Az SDN lehetővé teszi a hálózati architektúra szükség szerinti újrakonfigurálását, ami a hagyományos hálózatkezeléssel szemben rugalmasságot és dinamizmust visz a hálózatba. Optimalizálta a skálázási folyamat hatékonyságát, segített a skálázható adatközpontok létrehozásában, a hálózati erőforrások karbantartási terhek csökkentésében. Ezek alapján elmondható, hogy az SDN mára a hagyományos hálózatépítés népszerű alternatívájává vált.

1.2. NFV

A virtualizáció elengedhetetlen a modern hálózatépítéshez, az SDN absztrahálja (virtualizálja) a hálózati funkciókat a fizikai hálózati infrastruktúrától és szoftverben definiálja azokat, így a hálózat programozhatóvá válik. Az NFV [13]–[15] ezeket a virtualizált hálózati funkciókat (VNF) szoftverként telepíti virtuális gépekre. Az NFV-t és az SDN-t gyakran együtt

alkalmazzák. Lényegében az SDN a hálózati továbbítási funkciók szétválasztásával segít a hálózat központi kezelésében és programozásában, míg az NFV a hálózati funkciókat a hardverről a szoftverre helyezi át, így az SDN számára olyan infrastruktúrát biztosít, ahol az szoftverét működtetheti.

Az NFV szabványosításában számos munkacsoport vesz részt, többek között az IETF, az IRTF, az ITU, az IEEE és az ETSI [16]. Az 1.2 ábra bemutatja az ETSI által javasolt általános NFV-architektúrát [17], amely két fő komponensből áll [17]. Az egyik komponens a gyakran VNF-ként emlegetett hálózati funkciókat generáló szoftveralkalmazás. A másik a számítási kapacitásból, tárolásból és a hálózatból álló NFV-infrastruktúra (NFVI). Az NFV-infrastruktúra vezérlésért és az új VNF-ek rendelkezésre bocsájtásáért, valamint menedzsmentjért, automatizálásáért és hálózati orkesztrálásárt az NFV Management, Automation and Ochestartion (MANO) a felelős. Manapság a VNF-et leggyakrabban hálózati kapcsolók, tűzfalak, forgalomirányítás és a terheléselosztás esetében alkalmazzák.



1.2. ábra. Az ETSI által javasolt általános NFV-architektúra.

Az NFV növeli a flexibilitást, az agilitást és a skálázhatóságot, miközben csökkenti a költségeket azáltal, hogy a hálózati funkciókat univerzális alaphardvereken futtatja. Emellett jobb szegregálást tesz lehetővé a hálózati biztonság és az éles, valamint tesztkörnyezetek tekintetében. A hálózati funkciókat úgy tervezték, hogy hipervizorok alatt (virtuális gépekben) telepítsék és általános célú CPU-kkal ellátott generikus hardveren üzemeltessék. Alapvetően ezeket nem magas számításigényű feladatokra fejlesztették ki, mint például a hálózati funkciók, emiatt számos skálázhatósági és performancia problémával kell megküzdeniük. [18], [19].

1.3. P4 hálózati kapcsolók

A hagyományos hálózati funkciókat elsősorban dedikált hardveren, például alkalmazásspecifikus integrált áramkörökön (ASIC) valósították meg. Ez a konstrukció elsősorban a chipek teljesítményéből ered, mert a programozható chipek teljesítménye lényegesen alacsonyabb volt,

mint amit az ASIC-ek nyújtani tudtak. Következésképpen a hálózati eszközök rögzített, jól ismert viselkedéssel rendelkeztek, amelyet a gyártók közvetlenül a chipekbe égettek. Ennek a hardverközpontú hálózatépítésnek az egyik fő jellemzője az volt, hogy a továbbítási logika rugalmas megváltoztatására csak korlátozottan volt lehetőség. Emellett új protokollok és egyéb hálózati funkciók támogatását sem lehetett hozzáadni a már meglévő értékkészlethez.

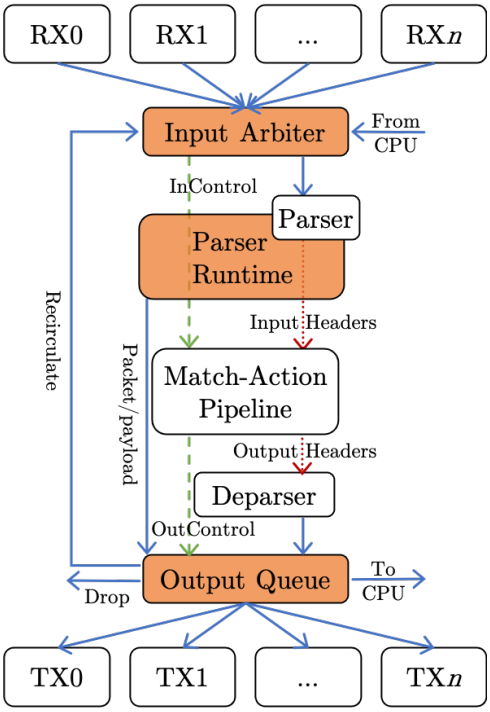
A hagyományos hálózatoknak jelentős skálázhatósági, üzemeltetési és menedzsment kihívásokkal kell szembenézniük, mivel az általuk szállított forgalom mennyisége és heterogenitása töretlenül növekszik a digitalizáció előrehaladtával. A gyors, megfizethető és programozható chipekkel a hálózat dinamikus konfigurálhatósága megvalósíthatóvá vált. A P4 egy kulcsfontosságú paradigma, amely elősegíti a hálózati programozhatóságot, lehetővé teszi, hogy a továbbítási viselkedést egy tartományspecifikus nyelvvel, a P4-gyel fejezzük ki és ezt követően egy olyan programba fordítsuk, amely betölthető és futtatható hálózati eszközökön [5].

Az 1.3 ábra egy P4 hálózati kapcsoló referencia modelljét mutatja be. Az architektúra szorosan követi a Very Simple Switch referenciamodellt [20]. A fehér blokkok olyan programozható komponenseket jelölnek, amelyek viselkedését egy megfelelő P4 programmal kell meghatározni. Architektúrális szinten a programozható csővezeték (azaz a PISA) három fő komponenst tartalmaz [21]:

- egy Parser-t, amely meghatározza, hogy a későbbi szakaszok milyen fejlécmezőket ismerjenek fel és illesszenek össze,
- egy vagy több fejlécmező illesztésére programozott Match-Action Units sorozatot,
- és egy Deparser-t, amely a csomag metaadatait a csomagba vissza szerializálja, mielőtt az a kimeneti portokon továbbításra kerülne.

A narancssárga színnel jelölt bemeneti arbiter, Parser és a kimeneti sorbanállás mind fix funkciójú komponensek. A piros szaggatott nyilak a felhasználó által definiált adatok áramlását jelzik. Végül a programozható és a fix funkciójú komponensek közötti (belső metaadatként megjelenített) információt közveztítő adatsíkinterfészeket szaggatott zöld színű nyíl jelöli. Érdemes megjegyezni, hogy a P4-et úgy tervezték meg, hogy csak a cél adatsík funkcionalitását adja meg [20]. Következtetésképpen nem használható a cél vezérlősík funkcionalitásának leírására, mindamellett a P4 programok részben meghatározzák azt az interfészt, amelyen keresztül a vezérlési sík és az adatsík kommunikál.

Az 1.3 ábrán látható referneciamodell a csomagokat többféle porton keresztül tudja fogadni, ami lehet egy bemeneti Ethernet port, vagy közvetlenül a CPU-hoz csatlakoztatott port, vagy egy recirkulációs csatorna. A rendszer egy Parser-rel rendelkezik, amely egyetlen egy Match-Action pipeline-ba továbbítja a csomagokat, ahol a Deparser [20] feldolgozza és a kimeneti sorba helyezi őket. A csomagelemzés állapotgépként van megvalósítva, ahol math-action táblák definiálják a csomagfeldolgozási pipeline-t, amely alapján előre meghatározott sorrendben történik a műveletvégzés. A match-action táblák feltöltése a vezérlősík interfészén keresztül valósul meg. A csomagokat feldolgozó P4 program viselkedése a bitvektorok bitvektorokra



1.3. ábra. Egy P4 hálózati kapcsoló magas szintű kapcsolási rajza.

való leképezésével írható le. A Deparser-ből kilépve a csomagok a kimeneti Ethernet portokon vagy a három speciális port egyikén keresztül kerülnek kiküldésre: *(i)* CPU port, amely továbbítja a csomagokat a vezérlő sík felé, *(ii)* eldobó port, amely eldobja az adott csomagot és *(iii)* visszacsatoló port, amely a csomagokat egy speciális bemeneti porton keresztül újra a hálózati kapcsolóba küldi.

A fordítás során a feldolgozási csővezeték az adott céleszköz rendelkezésre álló erőforrásaihoz képest kerül leképezésre. A céltípustól függően a megfeleltetési táblázatok jellemzően vagy a RAM-ba (szoftveres kapcsolók esetében), vagy a TCAM-ba (hardveres kapcsolók esetében) kerülnek leképezésre. A P4-et úgy tervezték meg, hogy a csomagfeldolgozó egységek széles skáláján lehessen implementálni, beleértve a hardveres (pl. SmartNIC [22], P4FPGA [23], P4-to-VHDL [24]) és szoftveres (pl.: P4-to-VHDL [24]), PISCES [25], BMv2 [26], Open vSwitch [27]) kapcsolókat is.

1.4. Kapcsolódó munkák

A sorbanállási modellek különböző körülmények között előre jelezhetik a hálózati teljesítményt, ezáltal értékes betekintést nyújthatnak a hálózati architektek és elemzők számára [28]. Továbbá ezek a modellek a szimulációknál lényegesen gyorsabb eredményeket szolgáltathatnak, miközben költséghatékonyabbak is. Ezeknek az előnyöknek köszönhetően a sorbanállás elmélete az elmúlt évtizedben újjáéledt.

Számos munka [28], [31]–[34] foglalkozik az OpenFlow kapcsolók teljesítményével. Jarschel és tsai. [28] elsőként vezettek le egy OpenFlow kontrollerrel kombinált OpenFlow kapcso-

1.1. táblázat. Kapcsolódó kutatások az analitikus modellezés tematikájában.

	Év	SDN hálózati kapcsoló	VNF	PNF	Jellemzők
Jarschel és tsai. [28]	2011	M/M/1	-	-	Fundamentális modellezés
Azodolmolky és tsai. [29]	2013	Hálózati Kalkulus	-	-	Csomagkésleltetés a kontroller esetében és a buffer méretének határértékei
Bozakov és tsai. [30]	2013	Hálózati Kalkulus	-	-	Vezérlőüzenetek feldolgozása
Mahmood és tsai. [31]	2014	M/M/1	-	-	Hálózati kapcsoló érkezési ráta konfigurálása
Jarschel és tsai. [32]	2015	M/M/1	-	-	Több hálózati kapcsoló
Miao és tsai. [33]	2015	HPQ:MMPP/M/1 LPQ:MMPP/M/1/k	-	-	Prioritási sorbanállás
Shang és tsai. [34]	2016	M/H_2/1	-	-	Beérkező üzenetek szegregálása
Sood és tsai. [35]	2016	M/Geo/1	-	-	Geometriai eloszlás a szolgáltatásokhoz
Miao és tsai. [36]	2016	HPQ:MMPP/M/1 LPQ:MMPP/M/1/k	-	-	MMPP használata multimédia forgalomhoz
Xiong és tsai. [37]	2016	M^x/M/1	-	-	Ingadozó érkezési ráta
Goto és tsai. [38]	2016	HPQ:MMPP/M/1 LPQ:MMPP/M/1/k	-	-	Egzakt megoldás Markov-lánccal
Fahmin és tsai. [7]	2018	M/M/1	M/M/1	-	SDN és VNF kombinációja
Singh és tsai. [8]	2018	2D Markov-lánc (HPQ, LPQ)	-	-	Szoftver vs. hardver hálózati kapcsolók
Zhao és tsai. [39]	2020	M/G/1	-	-	Szoftver által definiált WAN
Singh és tsai. [40]	2020	4D MC (belső buffer, HPQ, LPQ, hardver)	-	-	Enkapszuláció vs. belső buffer
Ez a dolgozat		M/M/1	M/M/1	M/M/1	P4 kapcsoló NF-vel

ló továbbítási sebességére és blokkolási valószínűségére vonatkozó alapmodellt, ahol az OpenFlow kontrollert egy M/M/1-S visszacsatolt sorbanállási rendszerrel modellezték. Bár a [28]-ban javasolt modell értékes betekintést nyújt, azonban csak akkor pontos, ha az új flow várakozásának valószínűsége kicsi. Ráadásul a modell kiterjesztése egynél több továbbító elemre az adatsíkon komplex feladat. E korlátozások miatt Mahmood és tsai. [31] olyan modellt javasoltak, amely mindkét kihívást megoldja. Modelljük a Jackson-elméleten [41], [42] alapul, de OpenFlow-alapú SDN-hálózatra szabott korrekciókat tartalmaz. Jarschel és tsai. [32] szintén beszámolnak az OpenFlow-alapú SDN teljesítményelemzéséről. A [32]-ben szereplő munka több-csomópontos környezetet vizsgál, szemben a [28]-rel, amely a hálózat teljesítményét egy-vezérlős beállításban vizsgálja. Ezért a javasolt modell alkalmas lehet a hálózat teljesítményének elemzésére realisztikusabb topológiák mellett.

A kutatási munkák [28], [31], [32] főként a logikailag centralizált kontrollerek kapacitásának feltárására összpontosítottak a vezérlési síkban, miközben nem vizsgálták kvantitatív módon a mögöttes adatsík teljesítményét. Miao és tsai. [33] kísérletet tettek ennek a hiányosságnak a pótlására, és bemutatottak egy preemptív alapú csomagelosztási sémát a globális igazságosság javítása (mérőszám mely alapján eldönthető, hogy az adott alkalmazás méltányos részesedét kap-e a rendelkezésre álló erőforrásokból) és a csomagvesztési arány csökkentése érdekében az SDN adatsíkban. Kvantitatívan értékelték a javasolt ütemezési sémát egy olyan rendszer segítségével, ahol a kontrollert egy M/M/1 sorral modellezték, míg a kapcsolót magas prioritású (MMPP/M/1) és alacsony prioritású (MMPP/M/1/k) sorok kombinációjával valósították meg. Ezt követően pontosan meghatározták az SDN-architektúra teljesítményszűk keresztmetszetét. Később Miao és tsai. [36] egy Priority-Queue (PQ) rendszert adaptáltak az SDN adatsík modellezésére. A javaslatot a multimédiás forgalom reális jellege és a Markov modulált Poisson-folyamat segítségével modellezett csomagérkezési burst-ök valószínű előfordulása motiválta.

Goto és tsai. [38] két magas és alacsony prioritású sorral rendelkező kapcsolókat is figyelembe vesznek. Javasoltak egy OpenFlow-alapú SDN sorbanállási modellt, amely figyelembe veszi a kapcsolóhoz érkező csomagok klaszteres kezelését és pontos elemzést adtak erre a javasolt sorbanállási modellre. A munka az analitikus modellezés két kritikus szempontjára összpontosít: (i) a kontrollerből a kapcsolóhoz érkező csomagok eltérő módon való kezelésére és (ii) az analitikus modellek kísérleti adatokkal történő validálására.

Nem minden létező megoldás alapul M/M/1 sorban állási modelleken. Shang és tsai. [34] azonban az OpenFlow kapcsolók és kontrollerek csomagfeldolgozási idejét M/H_2/1 sorbanállást használva értékelik. Sood és tsai. [35] egy SDN-kapcsolót M/Geo/1 rendszerként modelleznek, ahol a bejövő csomagok Poisson-eloszlásnak engedelmeskednek és a szolgáltatási ráta (az SDN-kapcsoló szabályalapú match-action csomagfeldolgozása) geometriai eloszlás szerint működik. Xiong és tsai. [37] az OpenFlow kapcsolók csomagtovábbítását és az SDN kontroller csomag-beérkező üzenetek feldolgozását M^X/M/1 és M/G/1 sorbanállási rendszerként modellezik. Végül Zhao és tsai. [39] kifejlesztik a csomagtovábbítás teljesítményének sorbanállási rendszerét a szoftveresen definiált WAN-ban, amelynek kontrollerjét M/M/n sorbanállásként modellezik, míg a csomagfeldolgozást M/G/1 sorbanállásként jellemzik.

A hálózati kalkulust [43] (Network Calculus) szintén gyakran használják teljesítménymodellezésre [29], [30]. Azodolmolky és tsai. [29] hálózati kalkulust használnak egy SDN-kapcsoló és kontroller funkcionalitásának leírására, a késleltetés és a sorbanállás hosszának modellezésére, valamint az SDN-kontroller és az SDN-kapcsoló pufferhosszának elemzésére törekedve. Bozakov és tsai. [30] a különböző kapacitású kapcsolók egyidejű működésére összpontosítanak a vezérlőüzenetek feldolgozása közben, ami az SDN-alkalmazásokban kiszámíthatatlan késleltetésekhez vezethet. Ezt a problémát egy sorbanállási modell segítségével kezelik, amellyel a kapcsoló kontroller interfészének szolgáltatását jellemzik. Hálózati kalkulust használnak a megfelelő paraméterek levezetéséhez.

Az SDN és az NFV együttes analitikus modellezésére csak néhány munka irányul. A Fahmin és tsai. úttörő munkája az SDN és NFV architektúrákat egy M/M/1 sor [7] segítségével modellezi és elemzi. A szerzők két architektúrát vesznek figyelembe: az egyikben a kontroller lép kölcsönhatásba az NFV-vel, a másikban pedig a switch lép kölcsönhatásba az NFV-vel. Kapcsolódó munkák továbbá [8], [40], de hardveres és szoftveres kapcsolókra általánosítva Singh és tsai. mutatják be a megoldásokat. Ezek a munkák szintén a sorbanállás elméletét használják a hardveres és szoftveres kapcsolók különböző teljesítményjellemzőinek (pl. késleltetés, csomagvesztés és átviteli teljesítmény) modellezésére az SDN-ben. E munkák fő tanulsága, hogy az SDN és az NFV egymást kiegészítő technológiák. Együttesen hozzájárulhatnak a hálózatok rugalmasságának és egyszerűségének javításához, valamint az azokon keresztül történő szolgáltatásnyújtáshoz.

A dolgozatban analitikus modellt dolgoztam ki a P4 és az NFV együttes működésére. A P4-es felhasználási célok sokfélesége és erőforrásigényeik szerteágazóak a lehetséges konfiguráció variációk végett. A teljesítményanalízis elsődleges célja ezen változatok lehetőségeinek és hát-

rányainak vizsgálata. Az ilyen megértés lehetővé teszi a konszolidációjuk megfelelő konfigurációjának kiválasztását egy adott felhasználási esethez. A P4 meglévő teljesítményértékelései főként az olyan statikus műveletekre összpontosítanak, mint az áteresztőképesség és a csomagfeldolgozási késleltetés [23], [25], [44]. Számos architektúra létezik az NFV erőforrás-kezelési (pl. dinamikus erőforrás ütemezés, hálózati funkciók biztosítása és késleltetés csökkentése) szempontok applikálására [45]–[47]. Végül, de nem utolsósorban, a P4 újrakonfigurációs képességét az adatsík virtualizáció eléréséhez is felhasználták [6], [48]–[50]. Jelen tudomásom szerint a mai napig nem végeztek olyan munkát, amelynek célja a P4 és az NFV együttes analitikus modellezése volt, így jelen dolgozat az első, amely ennek a hiányosságnak a megszüntetésével foglalkozik.

2. fejezet

P4 hálózati kapcsoló modellezése NFV-vel

A dolgozatban egy olyan rendszert modellezek, ahol az NFV kiegészíti a P4 hálózati kapcsoló működését. A jobb működési betekintés érdekében a P4 hálózati kapcsoló belső struktúráját különálló elemeknek tekintem, ahol a Switch Processing (SP) a bejövő csomagok előfeldolgozását végzi, a PNF a fizikai hálózati funkciókat valósít meg, a Switch Computing (SC) pedig a P4 hálózati kapcsoló kimeneti portjain végzett műveletek végrehajtásáért felelős. Továbbá feltételezem, hogy a VNF-re azért van szükség, hogy enyhítse a PNF-ek erőforrás korlátai miatti kihívásokat. Ilyen körülmények között az NF iránti igények olyan mértékben megnövekedhetnek, hogy a megfelelő működés veszélybe kerülhetne. A VNF-eknek továbbított csomagokkal a P4 hálózati kapcsoló erőforrásainak felszabadítása a célom. Bár a VNF-nek szánt csomagok körutazási ideje kétségtelenül megnő, azonban a PNF-re nehezedő terhek enyhülnek, így a rendszer optimális működése érhető el feltételezésem szerint.

2.1. Feltételezések és jelölések

A modellhez több hipotézisem van. Először is, a kapcsolóhoz érkező adatok érkezési folyamata Poisson-folyamat. Másodszor, a bemeneti sorban lévő újonnan észlelt csomagokat ugyanúgy kezelem, mint az NF által már feldolgozott csomagokat (nincs megkülönböztetés közöttük). Harmadszor, az NF feldolgozására csakis kizárólag egyszer van szükség.

Az analitikus modellben a kapcsoló, a kontroller, a VNF, és a PNF számára egymástól függetlenül M/M/1 sorbanállást veszek figyelembe. Ezen sorok érkezési rátája azonban függ a többi sor visszacsatolásától.

Az elemzés során használt jelöléseket a 2.1 táblázat tartalmazza. A különböző paraméterek jelölésére felső indexeket használtam. A csomagok érkezési sebességét a különböző komponenseknél λ-val, míg a számítási kapacitásokat c-vel, a késleltetést pedig D-vel jelölöm. Továbbá az SP és az SC számítási kapacitását külön kezelem, bár egy valódi P4 hálózati kapcsoló esetében általában szimmetrikus kapacitással rendelkeznek ezek. A vizsgált esetben a külön kezelést a PNF modellbe való bevonása tette szükségessé.

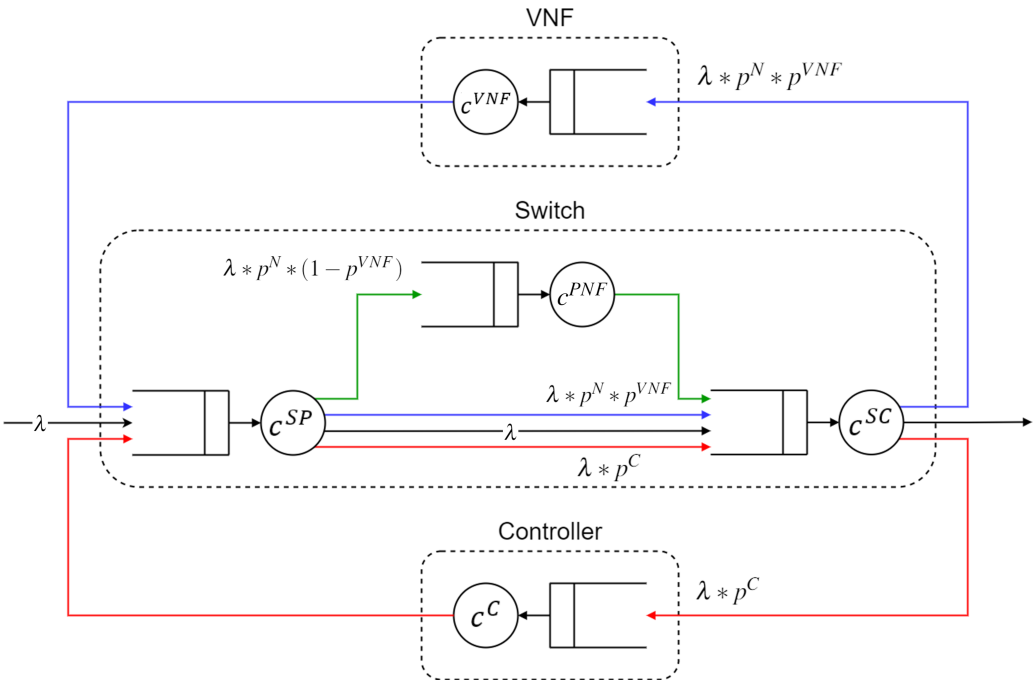
2.1. táblázat. Az elemzés során használt jelölések.

Szimbólum	Leírás
VNF	Virtualized Network Function (NFV-ben fut)
PNF	Physical Network Function (P4-ben fut)
SC	Switch Communication
SP	Switch Processing
p^C	Table miss valószínűsége (a csomag a kontroller számára kerül továbbításra)
λ	Csomagok érkezési rátája
λ^{PNF}	Csomagok érkezési rátája PNF-nél
λ^{VNF}	Csomagok érkezési rátája VNF-nél
λ^{SC}	Csomagok érkezési rátája SC-nél
p^N	NF szükséglet valószínűsége
p^{VNF}	VNF-hez való irányítás valószínűsége
c^{VNF}	VNF számítási kapacitása
c^{PNF}	PNF számítási kapacitása
c^{SP}	SP számítási kapacitása
c^{SC}	SC számítási kapacitása
D^{SV}	Rögzített csomagtovábbítási késleltetés a P4 hálózati kapcsoló és a VNF között
D^{SC}	Rögzített csomagtovábbítási késleltetés a P4 hálózati kapcsoló és a kontroller között
$\overline{D}^{Total}$	Átlagos csomagkésleltetés
$\overline{D}^{SP}$	c^{SP} átlagos csomagkésleltetése
$\overline{D}^{C}$	c^{C} átlagos csomagkésleltetése
$\overline{D}^{VNF}$	c^{VNF} átlagos csomagkésleltetése
$\overline{D}^{PNF}$	c^{PNF} átlagos csomagkésleltetése
$\overline{D}^{SC}$	c^{SC} átlagos csomagkésleltetése

2.2. Rendszermodell

A 2.1 ábrán látható a használt sorbanállási modell struktúrája. A P4 hálózati kapcsoló, a VNF és a kontroller külön szegmensekben láthatóak. A bal oldali bejövő nyíl a Poisson-folyamat szerint érkező csomagokat jelöli, amelyeket a λ szimbólummal jelölök. Az újonnan észlelt csomagot mindig a P4 hálózati kapcsoló dolgozza fel először (ebben az állapottérben az elsőként érkező csomagot tekintettem). Ezután a P4 hálózati kapcsolón belül a ingress művelet végrehajtási sorrendjének részeként a kapcsolófeldolgozó komponens meghatározza, hogy szükséges-e fizikai hálózati funkciót használni a csomag kiszolgálásához. Ha elegendő erőforrással rendelkezik a hálózati kapcsoló, akkor folytatja a csomag feldolgozását a hardverkörnyezeten belül. Optimális p^{VNF} esetén azonban az egress művelet végrehajtása során a VNF felé küldhető ki az adott csomag. Ezért létfontosságú ennek a paraméternek az értékét megfelelően megválasztani, mivel egy nem optimális érték negatívan befolyásolhatja a teljes rendszer átlagos késleltetési idejét.

A hálózati kapcsoló Switch Processing részének van egy további feladata is, abban az eset-



2.1. ábra. Sorbanállási modell.

ben, ha a kapcsoló maga nem tudja eldönteni, hogy az adott csomaghoz milyen műveletre van szükség, akkor a kontrollerhez fordulhat segítségért. A csomagot ezután a rendszer egress lépése során megfelelően megjelöli és elküldi a kontrollernek. Érdemes megjegyezni, hogy ennél a műveletnél a csomagok fizikailag elhagyják a P4 hálózati kapcsolót.

Miután a kontroller elvégezte a szükséges csomagszintű módosításokat a csomag ismét sorba áll a P4 hálózati kapcsoló sorában és később a Switch Processing ismét feldolgozza. A Switch Processing érzékeli a módosított csomagot és továbbítja azt a Switch Communicationnek, amely a megfelelő porton keresztül küldi ki a csomagot. Mind a Switch Processing, mind a Switch Communication M/M/1 sorba állítást alkalmaz, ahol a bejövő csomagok kézbesítési ideje exponenciálisan oszlik meg.

A fentiek tényében a 2.1 ábra színkódolása a következő. A zöld, a PNF-et igénylő csomagok útvonalát jelöli. A λ érkezési rátájú (feketével jelölt) Switch Processing által feldolgozandó csomagokat a c^{PNF} csomópont szolgálja ki. A terheléselosztás céljából a VNF-nek címzett csomagok a kékkel jelölt útvonalon kerülnek továbbításra. Miután a Switch Computing elküldte a csomagot a c^{VNF} csomópontnak és feldolgozta a kérést a csomagot ismét továbbítja a Switch Processing csomópontnak. A Switch Processing ezután további művelet nélkül továbbítja a csomagot a Switch Communication csomópontnak, ahonnan az ezután a switch megfelelő kimeneti portjára kerül. Végül piros szín jelöli azon csomagok útját, amelyek esetében a Switch Processing nem tudja meghatározni a feldolgozáshoz szükséges műveletet, mivel nincs megfelelő bejegyzés a match + action táblájában. Így a Switch Communication az ilyen csomagokat a c^C csomópontba küldi. A szükséges műveletek elvégzése után a csomagok továbbításra kerülnek a Switch Processinghez. Ezután további műveletek nélkül továbbítják őket a Switch Communication-hoz, ahonnan a hálózati kapcsoló megfelelő kimeneti portjára kerülnek.

Azon esetekben amikor nincsen NF igény, de rendelkezik a hálózati kapcsoló megfelelő útvonalválasztási tábla bejegyzéssel a csomagra nézve, a Switch Processing feldolgozza, majd azt közvetlenül Switch Communication számára továbbítja, ahonnan pedig elhagyja a hálózati kapcsolót a megfelelő kimeneti porton. Ez a folyamat a fekete színnel jelölt útvonalon figyelhető meg.

2.3. Késleltetés elemzés

A kapcsoló, a kontroller és a VNF esetében is M/M/1 sorbanállási modellt feltételezek, ahol a csomagok Poisson-eloszlás szerint érkeznek λ rátával. A csomagok átlagos késleltetési idejét a különböző forrásokból érkező ráták és az adott csomópont kapacitásának számításából kaphatjuk meg.

Az átlagos csomagkésleltetést a *(i)* teljes érkezési ráta és *(ii)* a kapcsoló szolgáltatási sebességének, valamint *(iii)* a (kapcsoló) kapacitásának felhasználásával vezetem le. A rendszer átlagos csomagkésleltetésének kiszámításához először ki kell számítanunk a különböző ágak késleltetési értékeit. Mivel a rendszer Switch Processing része egy M/M/1 sor az átlagos csomagkésleltetés a következőképpen származtatható le

$$\overline{D}^{SP} = \frac{1}{c^{SP} - \lambda \times (1 + p^N \times p^{VNF} + p^C)}, \tag{2.1}$$

ahol a szolgáltatási ráta c^{SP} és a csomagok érkezési sebessége ebben a csomópontban a következőképpen határozható meg. A Switch Processing-hez érkező csomagok közé tartoznak *(i)* a másik hálózati kapcsolóból küldött csomagok, *(ii)* a VNF-től érkező csomagok és *(iii)* a kontrollertől érkező csomagok. Az *(i)* esetében az érkezési ráta λ. Az *(ii)* esetében a csomagszám $\lambda \times P^N \times P^{VNF}$, ahol P^N az NF-et igénylő csomagok valószínűsége, P^{VNF} pedig a VNF által kiszolgált csomagok valószínűsége. A *(iii)* esetben az arány $\lambda \times P^C$, ahol P^C a Table-miss valószínűségét jelöli. Így a csomagok érkezési rátája ebben a pontban a következő $\lambda \times (1 + P^N \times P^{VNF} + P^C)$.

Ha a P4 kapcsoló nem tudja, hogy egy adott csomaghoz milyen szolgáltatásra van szükség, akkor azt a kontrollernek küldi. A kontrollerre ugyanazok a Markov-tulajdonságok vonatkoznak, mint a Switch Processing-re. Az átlagos késleltetés ezen a csomóponton a következőképpen számítható ki

$$\overline{D}^{C} = \frac{1}{c^{C} - \lambda \times p^{C}} + 2 \times D^{SC}, \tag{2.2}$$

ahol a kiszolgálási ráta c^C, az érkezési ráta pedig $\lambda \times p^C$, ahogy fentebb említettem. A rendszert ebben a csomópontban egy általános állandó késleltetés D^{SC} jellemzi, amelyet a P4 hálózati kapcsoló és a kontrollert megvalósító hardver közötti kommunikáció miatt kell hozzáadni.

Ha a rendszer azt észleli ,hogy NF-re van szükség, de a PNF nem tudja kiszolgálni az adott kérést (feltéve, hogy van ilyen funkció a kialakított struktúrában) erőforráskorlátok miatt, akkor a csomagot egy külső erőforráshoz küldi. Ebben az esetben a csomópontonkénti átlagos

késleltetés a következőképpen számítandó ki

$$\overline{D}^{VNF} = \frac{1}{c^{VNF} - \lambda \times p^{N} \times p^{VNF}} + 2 \times D^{SV}, \tag{2.3}$$

ahol c^{VNF} a szolgáltatási ráta, az érkezési ráta pedig $\lambda \times p^{N} \times p^{VNF}$ a fentiek szerint. A kontrollerhez hasonlóan a külső kommunikáció miatt a rendszerre egy állandó késleltetési idő hárul, amely a D^{SV}.

A PNF átlagos csomagkésleltetését a következőképpen számíthatjuk ki

$$\overline{D}^{PNF} = \frac{1}{c^{PNF} - \lambda \times p^{N} \times (1 - p^{VNF})}. \tag{2.4}$$

Mivel ez fizikailag a P4 hálózati kapcsoló hardverében van megvalósítva a rendszerbe nem kerül további késleltetési idő. Ekkor a kiszolgálási ráta az adott csomópontban c^{PNF}, az érkezési ráta pedig $\lambda \times p^{N} \times (1 - p^{VNF})$, a korábbiakban bemutatott módon.

A Switch Computing csomópontban az átlagos csomagkésleltetés a következőképpen fejezhető ki

$$\overline{D}^{SC} = \frac{1}{c^{SC} - \lambda \times (1 + p^{N} \times p^{VNF} + p^{C})}, \tag{2.5}$$

ahol a rendszer kiszolgálási rátája c^{SC} és az érkezési ráta $\lambda \times (1 + p^{N} \times p^{VNF} + p^{C})$, a fentiek szerint.

Egy VNF-csomag esetében az átlagos várakozási időt az SP kétszeres, a VNF egyszeres és az SC szintén kétszeres áthaladásához szükséges idők összege adja. Ez formálisan a következőképpen fejezhető ki

$$\overline{D}^{VNF}_{(Total)} = 2 \times \overline{D}^{SP} + 2 \times \overline{D}^{SC} + \overline{D}^{VNF}. \tag{2.6}$$

A kontrollernek címzett csomag esetében az átlagos késleltetési időt az SC és SP kétszeri áthaladásához szükséges idők összege adja, amelyet formálisan a következőképpen fejezhetünk ki

$$\overline{D}^{C}_{(Total)} = 2 \times \overline{D}^{SP} + 2 \times \overline{D}^{SC} + \overline{D}^{C}. \tag{2.7}$$

Végül egy PNF csomag esetében az átlagos késleltetési időt az SP, PNF és SC áthaladásához szükséges idők összege határozza meg, azaz

$$\overline{D}^{PNF}_{(Total)} = \overline{D}^{SP} + \overline{D}^{PNF} + \overline{D}^{SC}. \tag{2.8}$$

A fenti képletek fényében a teljes rendszer átlagos csomagkésleltetése a következőképpen fejezhető ki

$$\begin{aligned} \overline{D}^{Total} = {} & p^{N} \times p^{VNF} \times \overline{D}^{VNF}_{(Total)} + p^{N} \times (1 - p^{VNF}) \\ & \times \overline{D}^{PNF}_{(Total)} + p^{C} \times \overline{D}^{C}_{(Total)} + (1 - p^{N} - p^{C}) \\ & \times (\overline{D}^{SP} + \overline{D}^{SC}). \end{aligned} \tag{2.9}$$

3. fejezet

Optimalizálási probléma

A dolgozatban a P4 hálózati kapcsolókról az NFV-nek való terhelésáthelyezési képességét vizsgálom a programozható adatsíkokban. Ennek érdekében megvizsgálom a VNF-nek küldött csomagok valószínűségét (p^{VNF}) az átlagos csomagkésleltetéshez ($\overline{D}^{Total}$) képest különböző paraméterkonfigurációk esetén. Célom, hogy p^{VNF} vizsgálatával meghatározzam az optimumot úgy, hogy a késleltetés minimális legyen. Formálisan azt szeretném, hogy

$$\begin{aligned} &\text{minimalizálja} \quad \overline{D}^{Total}(p^{VNF}), \quad p^{VNF} = (p_1^{VNF}, \ldots, p_{n_x}^{VNF}) \\ &\text{feltételekkel} \quad p^{VNF} \in [0,1], \quad \overline{D}^{Total}(p^{VNF}) \geq 0, \quad \theta \end{aligned} \tag{3.1}$$

ahol a θ az a paraméter, amelynek hatását vizsgáljuk, míg a többi érték rögzített,

$$\theta = \begin{cases} C^{PNF} \in [190, 250], & \text{if} \quad \theta := C^{PNF}. \\ C^{VNF} \in [200, 1400], & \text{if} \quad \theta := C^{VNF}. \\ D^{SV} \in [0.0, 0.1], & \text{if} \quad \theta := D^{SV}. \\ C^{SC} \in [600, 1600], & \text{if} \quad \theta := C^{SC}. \\ p^{C} \in [0, 1], & \text{if} \quad \theta := p^{C}. \end{cases}$$

A határértékeket egy tipikus P4 hálózati kapcsoló hardveres specifikációja alapján határoztam meg [51]. Ennek érdekében megvizsgáltam, hogy a túlméretezés, vagy alulméretezés hogyan hat ennek a hálózati kapcsolónak az általános működésére és ennek eredményeként kaptam meg a 3.1 egyenletben megadott határértékeket.

3.1. Problémaleírás

Célom egy olyan optimalizálási módszer megtalálása, amely a megengedett tartományból olyan értékeket rendel a döntési változókhoz, hogy a célfüggvény optimalizálva legyen és az összes előírt kényszer teljesüljön. Formálisan az optimalizálási probléma (matematikai programozási modell) a következő formájú

$$p^* := \min_{p^{VNF}} \overline{D}_0^{Total}(p^{VNF}) : \overline{D}_i^{Total}(p^{VNF}) \geq 0, \tag{3.2}$$

ahol $p^{VNF} \in \mathbf{R}^n$ a döntési változó, amely befolyásolja a célfüggvény értékét. A $\overline{D}_0^{Total} : \mathbf{R}^n \to \mathbf{R}$ célfüggvény, amely az optimalizálandó, azaz minimalizálandó mennyiséget jelenti. A $\overline{D}_i^{Total} : \mathbf{R}^n \to \mathbf{R}, i = 1, \ldots, m$ kényszereket jelöli, amelyek korlátozzák a döntési változókhoz rendelhető értékeket, valamint p^* az optimális értéket szimbolizálja.

3.2. Konvex optimalizáció

Az optimalizálási probléma (3.2 egyenlet) konvex optimalizálás formájába öntése azért hatékony, mert korábban szigorú korlátokat vezettek be a megoldások optimumára [52]. Így, figyelembe véve a 3.1 egyenletben felsorolt korlátokat, szigorúan bebizonyítható, hogy létezik egy olyan függvény, amely támogat egy adott hardver célkonfigurációt.

Ez azt jelenti, hogy a függvény $\overline{D}^{Total} : \mathbf{R}^n \to \mathbf{R}$ konvex, ha

$$\begin{aligned} &\forall x_1, x_2 \in \mathbf{R}^n, \\ &\forall \eta \in [0,1], \\ &\overline{D}^{Total}(\eta \times x_1 + (1-\eta) \times x_2) \leq \\ &\qquad \eta \times \overline{D}^{Total}(x_1) + (1-\eta) \times \overline{D}^{Total}(x_2), \end{aligned} \tag{3.3}$$

ahol x_1 és x_2 két tetszőleges pont az x tengely mentén.

Más szóval a függvény mindig a két tetszőleges pontot összekötő lineáris egyenes alatt helyezkedik el. Vagyis a függvény akkor és csak akkor konvex, ha az epigráfja

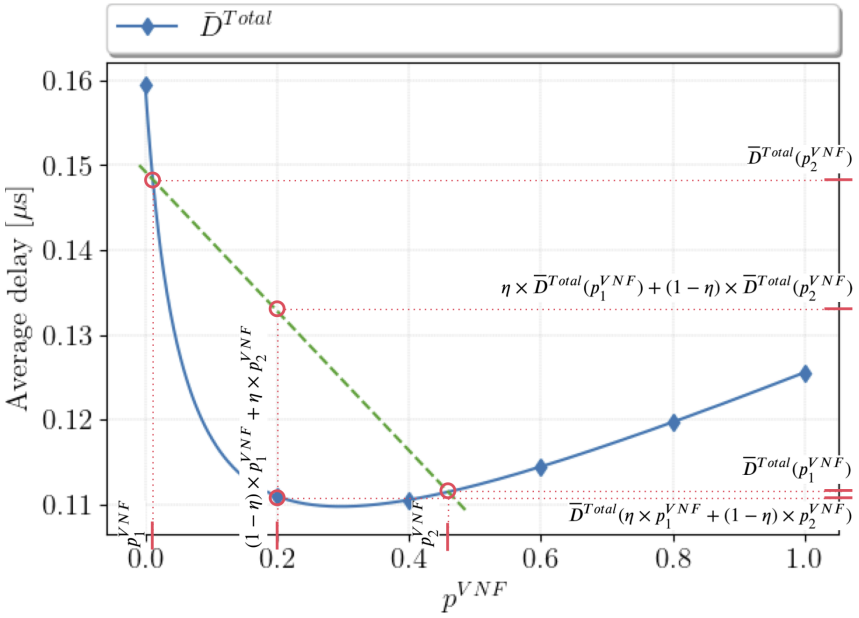
$$\text{epi}\, \overline{D}^{Total} := \left\{ (p^{VNF}, t) \in \mathbf{R}^{n+1} : t \geq \overline{D}^{Total}(p^{VNF}) \right\} \tag{3.4}$$

konvex. Az optimalizálási probléma (3.2 egyenlet) konvex, ha minden benne szereplő függvény $\overline{D}_0^{Total}, \overline{D}_1^{Total}, \ldots, \overline{D}_m^{Total}$ is konvex.

Az optimális értékeket a 1. algoritmus segítségével határoztam meg a fent említett feltételek mellett. A 3.1 ábrán látható a kapott görbe, amelyen az optimalizálási problémám konvexitása jelenik meg. A vizsgálat pontossága határozza meg a folyamat lefutásának sebességét, vagyis azt, hogy mekkora mintavételi gyakoriságot állítok be az algoritmus számára. Azokban az esetekben, amikor a görbület elég nagy egy adott íven, a kis frekvencia beállítása mellett is igazolható az erős konvexitás. Ellenkező esetben a rendszer a kis mintavételi frekvenciája miatt hibás becslést adhat egy kis görbületű ívre.

Algorithm 1 Convexivity check

w = [0, 1, . . . , F]
t = [0, 1, . . . , F]
y = []
x_1 = [0.000, 0.001, . . . , 0.800]
x_2 = [0.200, 0.201, . . . , 1.000]
for i in w **do**
    y.insert(i)
for i in x_1 **do**
    m = $\frac{d^2 i}{dx_1^2}$
    if m $\leq$ 0 **then**
        return m cannot be zero or smaller
    end if
    for j in x_2 **do**
        for z in t **do**
            if i = j **then**
                continue
            end if
            if t = 0.0 **or** 1.0 **then**
                continue
            end if
            left_side = $\overline{D^{total}}(z \times i + (1 - z) \times j)$
            right_side = $z \times \overline{D^{total}}(i) + (1 - z) \times \overline{D^{total}}(j)$
                $-\frac{1}{2} \times m \times z \times (1 - z) \times (i - j)^2$
            if left_side $\leq$ right_side **then**
                return Function is not strongly convex
            end if
return Function is strongly convex



3.1. ábra. A konvex függvény görbéje.

3.3. NFV valószínűség-optimalizálás

Miután megbizonyosodtam arról, hogy egy konvex optimalizálási problémával állok szemben, a minimum helyét és értékét a Brent-módszerrel határoztam meg [53], [54]. A Brent-módszer egy olyan gyökkereső algoritmus, amely a felezés módszerét, a szekánsmódszert és az inverz kvadratikus interpolációt ötvözi. Az algoritmus rendelkezik a bisection megbízhatóságával, de ugyanolyan gyors lehet, mint a kevésbé megbízható módszerek. Garantáltan ésszerű számú lépésszámon belül konvergál bármely függvény esetében.

Adott három pont x_1, x_2 és x_3, az NFV valószínűség-optimalizálás Brent módszerével (NPOB) a y kvadratikus függvényeként illeszkedik x-hez az interpolációs képlet segítségével.

$$
\begin{aligned}
x = & \frac{(y-\bar{D}^{Total}(x_1)) \times (y-\bar{D}^{Total}(x_2)) \times x_3}{(\bar{D}^{Total}(x_3)-\bar{D}^{Total}(x_1)) \times (\bar{D}^{Total}(x_3)-\bar{D}^{Total}(x_2))} \\
& + \frac{(y-\bar{D}^{Total}(x_2)) \times (y-\bar{D}^{Total}(x_3)) \times x_1}{(\bar{D}^{Total}(x_1)-\bar{D}^{Total}(x_2)) \times (\bar{D}^{Total}(x_1)-\bar{D}^{Total}(x_3))} \\
& + \frac{(y-\bar{D}^{Total}(x_3)) \times (y-\bar{D}^{Total}(x_1)) \times x_2}{(\bar{D}^{Total}(x_2)-\bar{D}^{Total}(x_3)) \times (\bar{D}^{Total}(x_2)-\bar{D}^{Total}(x_1)}.
\end{aligned}
\tag{3.5}
$$

Az NPOB segítségével meghatározok egy sor paramétert, amelyek a dolgozat alapigazságaként szolgáltak. Az interpolációt a p^{VNF} paraméter és a vizsgált θ paraméterkonfiguráció mentén végeztem, így kaptam meg a $\bar{D}^{Total}(p^{VNF})$ függvényt. Az optimum helyét az x_2 paraméter jelzi. Az NPOB pszeudo kódja a 2. algoritmusban látható.

Algorithm 2 NFV probability optimization with Brent's method

Tolerance = 0.05
bounds = [constraints in Equation (3.1)]
x_1 = bounds
x_2 = [$\overline{D}^{Total}$(bounds[0]), $\overline{D}^{Total}$(bounds[1])]
if $\overline{D}^{Total}(x_1) < \overline{D}^{Total}(x_2)$ **then**
    i = x_1
    $x_1 = x_2$
    x_2 = i
end if
$x_3 = x_2$
while $|x_2 - x_1|$ **do**
    if $\overline{D}^{Total}(x_2) \neq \overline{D}^{Total}(x_1)$
      and $\overline{D}^{Total}(x_2) \neq \overline{D}^{Total}(x_3)$
      and $(x_2 - x_1) <$ Tolerance $\times\, x_2$ **then**
        Calculate x according to Equation (3.5)
    else
        $x = \frac{x_1 + \overline{D}^{Total}(x_1) \times (x_1 - x_2)}{(\overline{D}^{Total}(x_2) - \overline{D}^{Total}(x_1))}$
    end if
    if $x \notin [x_1; x_2]$ **then**
        $x = \frac{x_1 + x_2}{2}$
        $x_3 = x_2$
    end if
    if $\overline{D}^{Total}(x) \times \overline{D}^{Total}(x_2)$ **then**
        $x_1 = x_2$
    else
        $\overline{D}^{Total}(x_1) = \frac{\overline{D}^{Total}(x_1)}{2}$
        $x_2 = x$
        $\overline{D}^{Total}(x_2) = \overline{D}^{Total}(x)$
    end if
return x_2 # approximate root

4. fejezet

Hatásvizsgálat

Ebben a fejezetben bemutatom a p^{VNF} analitikus eredményeit különböző rendszerparaméterek (3.1 egyenlet) variálásával, beleértve a kapcsolóról a kontrollerre való átirányítás valószínűségét, a PNF-et, a switch processing-et, a switch communication-t és a hálózati funkciót, valamint a különböző komponensek differens szolgáltatási rátáit. Ennek érdekében a 4.1 táblázatban felsorolt paraméterek egy csoportját választottam ki alapparaméterként és egyenként változtattam őket, míg a többit fixen tartottam, hogy felmérjem a modellem hatékonyságát.

Legjobb tudomásom szerint nincs olyan korábbi munka, amely a P4 hálózati kapcsolók alapszintű paramétereit tárgyalná az NFV-re való tehermentesítési képességekkel. Minden bejövő hálózati adatfolyamot új adatfolyamnak feltételezek. Az érkezési ráta paraméterét azonban a Chen és tsai. [55] elmélete szerint állítottam be, hogy reálisabb paraméterbeállításokat kapjak, míg a szolgáltatási sebesség paramétereit elsősorban a Fahmin és tsai. [7] és Zhang és tsai. [56] tanulmányai szerint választottam ki. Továbbá az átirányítási valószínűségi paramétereket úgy határoztam meg, hogy a rendszer tartózkodási ideje egy tipikus P4 hálózati kapcsoló [51], [57] késleltetési ideje alatt legyen.

Ezek függvényében feltételezem, hogy a λ csomagok érkezési rátája 250 pkts/ms. Feltételezem továbbá, hogy az új hálózati adaftolyamok érkezésének valószínűsége 56% körül van. Feltételezem azt is továbbá, hogy a csomagok 75%-a igényel szolgáltatást az NFV-től, míg a csomagok 64%-át a PNF dolgozza fel (31% Switch Processing és 19% Switch Communication). Ez utóbbit szintén a [7] figyelembevételével választottam, amelyben Fahmin és tsai. feltételezik, hogy nem minden csomag igényel szolgáltatást az NFV-től.

A következőkben bemutatom a P4 hálózati kapcsolokról az NFV-re történő tehermentesítés hatását a C^{VNF}, D^{SV}, C^{SC} és p^{C} paraméterek szempontjából. Minden ábrán a baloldali y tengely a *ms*-ban mért átlagos késleltetést, a jobboldali y az optimális p^{VNF}-t, míg a x tengely azt a paramétert jelöli, amelynek hatását vizsgálom. Ezenkívül minden ábra négy görbén keresztül ábrázolja a rendszer viselkedését: (*i*) a kékkel jelölt késleltetés görbéje az az eset, amikor a VNF-hez menő összes csomag valószínűsége 0 ($\overline{D}^{Total}(0)$) (azaz csak PNF-et használunk), (*ii*) a zölddel jelölt késleltetés görbéje azaz eset, ha a VNF-hez menő összes csomag valószínűsége 1 ($\overline{D}^{Total}(1)$) (vagyis, csak VNF-et használunk), (*iii*) az optimális p^{VNF} ($\overline{D}^{Total}(Optimal p^{VNF})$)

4.1. táblázat. A szimuláció paraméterei.

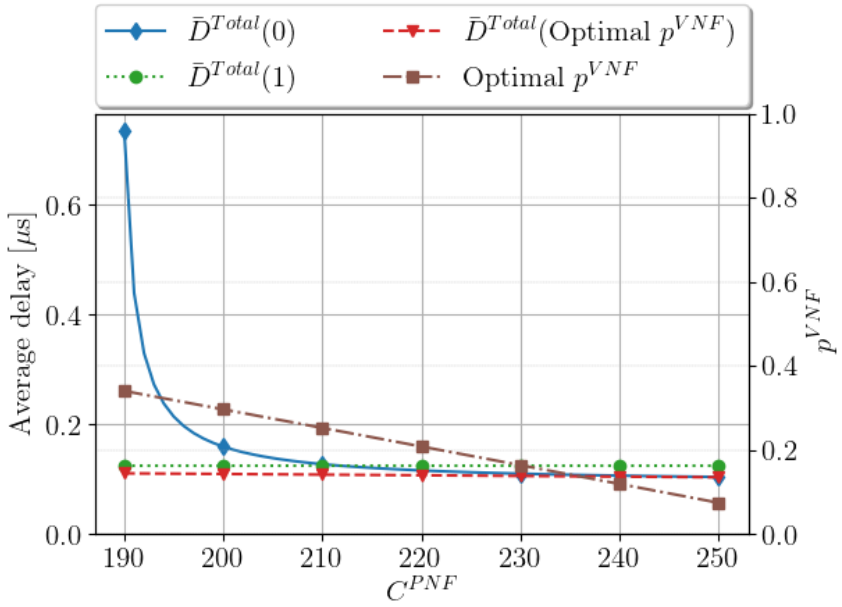
Paraméter	Értéke
Csomagok érkezési rátája C^{SP}-nél, λ	250 pkt/ms
Szolgáltatási ráta Switch Processing-nél, C^{SP}	4000 pkt/ms
Szolgáltatási ráta Switch Communication-nél, C^{SC}	4000 pkt/ms
Szolgáltatási ráta VNF-nél, C^{VNF}	800 pkt/ms
Szolgáltatási ráta PNF-nél, C^{PNF}	200 pkt/ms
Szolgáltatási ráta kontrollernél, C^{C}	100 pkt/ms
VNF-re való átirányítás valószínűsége, p^{PNF}	0.6364
Kontrollerhez való átirányítás valószínűsége, p^{C}	0.5602
Switch Processing-re való átirányítás valószínűsége, p^{SP}	0.3124
Switch Communication-re való átirányítás valószínűsége, p^{SC}	0.1923
NF-re való átirányítás valószínűsége, p^{N}	0.7553
Rögzített csomagtovábbítási késleltetés a VNF felé, D^{SV}	0.0422 ms
Rögzített csomagtovábbítási késleltetés a kontroller felé, D^{SC}	0.1876 ms

alatt kapott késleltetés piros színnel jelölve (azaz a tehermentesítés működik, PNF és az NFV "hibrid üzemmódban" van használatban) és (*iv*) az $Optimal p^{VNF}$ barna színnel van jelölve.

4.1. C^{PNF} hatása

A 4.1 ábra a C^{PNF} hatását mutatja az átlagos késleltetésre nézve *ms*-ban kifejezve. A 4.1 ábra alapján megállapítható, hogy a VNF-re történő terhelésáthelyezés nélkül ($\overline{D}^{Total}(0)$) a PNF kapacitásának (C^{PNF}) meg kell felelnie egy bizonyos fizikai méretnek ahhoz, hogy csökkenjen az átlagos késleltetés. A VNF-re történő terhelésáthelyezés ($\overline{D}^{Total}(1)$) esetén azonban az átlagos késleltetés állandó marad, mivel a feldolgozásra szánt összes csomagot az NFV-hez továbbítja a rendszer és a PNF-ben nem történik feldolgozás. Az átlagos késleltetés ebben az esetben 0,18 *ms*. $\overline{D}^{Total}(0)$ hasonló késleltetéssel rendelkezik, mint $\overline{D}^{Total}(1)$ abban az esetben amikor a C^{PNF} kapacitását 220 pkt/ms-ra skálázzuk. Ez azt jelenti, hogy ha a PNF-t kellően túlméretezzük, akkor ugyanaz a teljesítmény érhető el, mint a tisztán VNF használata esetén.

A rendszer legoptimálisabb működése azonban akkor érhető el, ha a VNF és a PNF egymást kiegészítve működnek. Ez a 4.1 ábrán látható, mivel a $\overline{D}^{Total}(Optimal p^{VNF})$ "hibrid" működés a C^{PNF} minden mért lépésénél eléri a legkisebb késleltetést. Az $Optimal p^{VNF}$ egy lineáris függvényt rajzol ki. A PNF méretének növelésével csökken a VNF-re történő terhelésáthelyezés szükségességének mértéke. Ez azt jelenti, hogy minél kisebb a PNF mérete, annál több VNF-re történő terhelésre van szükség az átlagos késleltetés minimalizálásához. Továbbá egy 190 pkt/ms-re skálázott méretű C^{PNF} esetén a csomagok körülbelül 36%-a kerül továbbításra a VNF-hez. Ha ez a méret 230 pkt/ms-ra skálázódik a VNF-nek továbbított csomagok mennyisége a felére csökken (azaz körülbelül 50%-kal több csomagot kell feldolgoznia a PNF-nek). Összességében kimondható, hogy a tehermentesítés előnye jelentős a paraméter függvényében.



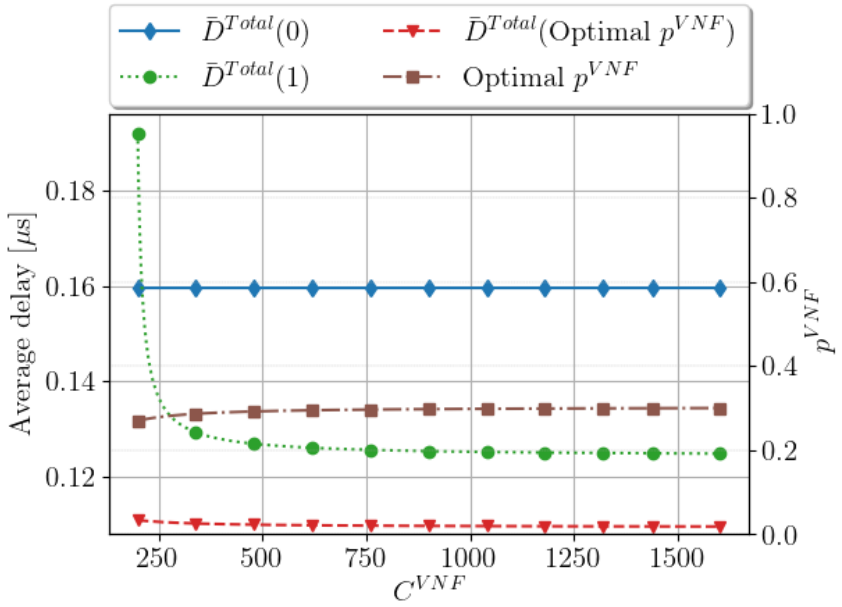
4.1. ábra. Optimális p^{VNF} paraméter különböző C^{PNF} értékekhez képest.

A C^{PNF} 190 pkt/ms-re skálázva az átlagos késleltetés a tehermentesítéssel harmadára csökkenthető (0,78 *ms*-ról 0,12 *ms*-ra), szemben a tisztán P4-es hálózati kapcsolókkal, ahol az összes csomagot csak a PNF dolgozza fel.

4.2. C^{VNF} hatása

A 4.2 ábrán látható a C^{VNF} különböző skálázásának a késleltetésre gyakorolt hatása. A 4.2 ábra alapján elmondható, hogy ha csak PNF-et használunk az átlagos késleltetés rögzítetten 0,16 *ms*. Ennek oka, hogy a C^{VNF} nem befolyásolja a rendszer átlagos késleltetését. Ha azonban minden csomagot továbbítunk a VNF-nek és a VNF kapacitásának növelése elér egy bizonyos pontot a további skálázás már nem befolyásolja az átlagos késleltetést (hiába skálázzuk jelentősen túl a méretét az eszköz átlagos késleltetését nem befolyásolja). Így az $Optimal p^{VNF}$ értéket a 600 pkt/ms-ra beállított C^{VNF} értéktől kezdve érjük el, e értéket meghaladó méretezésnek nincs effektív hozzáadott értéke a teljes rendszer átlagos késleltetésére nézve.

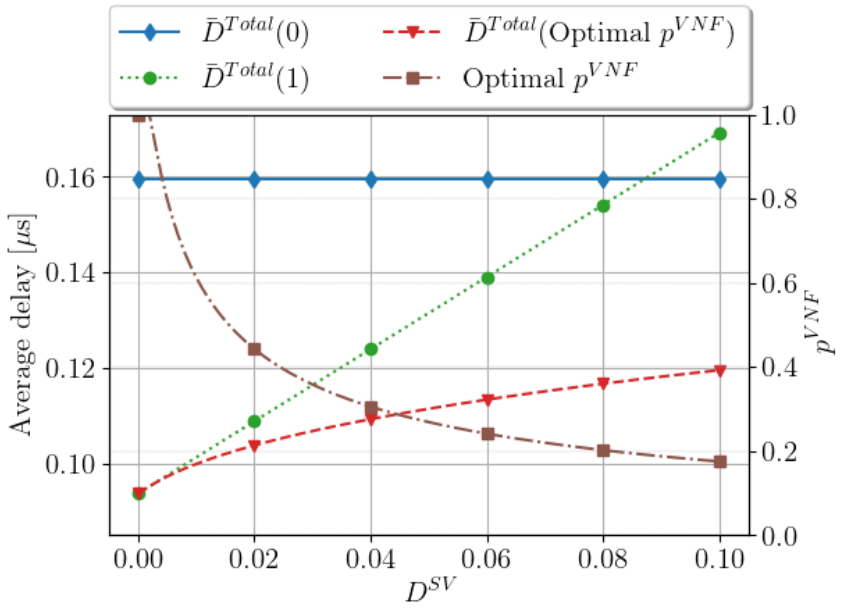
Továbbá, a C^{VNF} értékének növelésével a csomagok küldése a VNF-nek egyre előnyösebbé válik. Ha a C^{VNF} kapacitását 200-ról 600-ra állítjuk, akkor az $Optimal p^{VNF}$ növekedése körülbelül 5% (23%-ról 28%-ra). Összességében a P4 hálózati kapcsolókról az NFV-re történő terhelésáthelyezés előnye a 4.2 ábra esetében is nyilvánvaló, bár úgy tűnik, hogy a C^{VNF} kapacitása nem befolyásolja az átlagos késleltetést. Ennek ellenére az átlagos késleltetés tekintetében a teljesítménynövekedés jelentős. A tehermentesítéssel az átlagos késleltetés majdnem a felére csökkenthető (0,16 *ms*-ról 0,10 *ms*-ra). Ennek oka, hogy a $Optimal p^{VNF}$ 200 pkt/ms C^{VNF} kapacitás mellett 26% körül van. Következésképpen ez olyan kis lokális késleltetést eredményez, hogy nem befolyásolja a teljes rendszerre nézett átlagos késleltetést.



4.2. ábra. Optimális p^{VNF} paraméter különböző C^{VNF} értékekhez képest.

4.3. D^{SV} hatása

A 4.3 ábra alapján megállapítható, hogy ha csak PNF-et használunk ($\bar{D}^{Total}(0)$), akkor az átlagos késleltetés konstans. Ennek oka, hogy D^{SV} a P4 hálózati kapcsoló és a VNF közötti rögzített csomagtovábbítási késleltetést jelöli és mivel a VNF-et nem használjuk ez a késleltetés nem befolyásolja a rendszer működését, ezért állandó az átlagos késleltetés $\bar{D}^{Total}(0)$ esetében. Ha azonban csak VNF-et használunk ($\bar{D}^{Total}(1)$), akkor a D^{SV} növekedése lineárisan arányos a rendszer átlagos késleltetésének növekedésével.



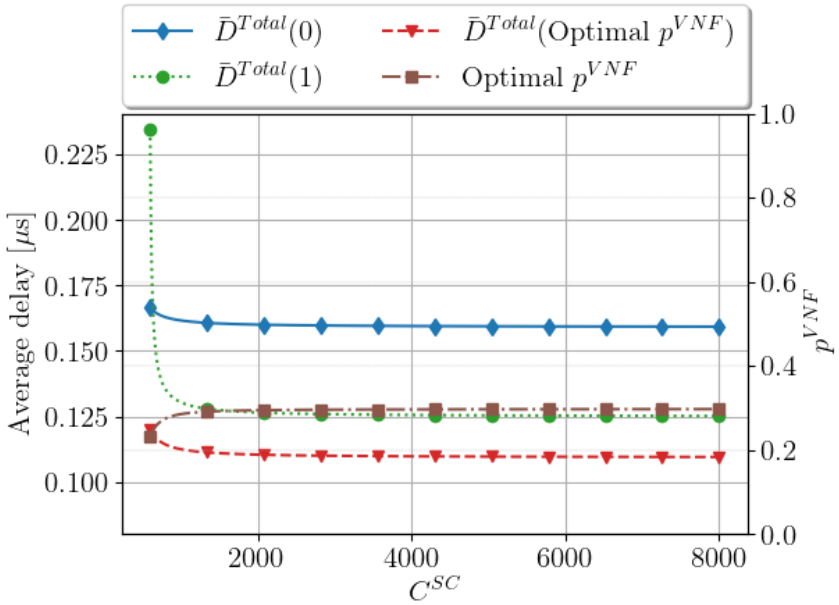
4.3. ábra. Optimális p^{VNF} paraméter különböző D^{SV} értékekhez képest.

A 4.3 ábra alapján nyilvánvaló, hogy a rendszer optimális működése a P4 hálózati kapcsolóról a VNF-re történő terhelésáthelyezéssel érhető el. A legjobb teljesítmény ($Optimal p^{VNF}$)

természeten úgy kapható meg, ha D^{SV} egyenlő 0-val. Ebben az esetben csak a VNF-et érdemes használni, mivel ebben a szituációban alacsonyabb átlagos késleltetéssel lehet operálni. A D^{SV} növekedésével azonban az $Optimal p^{VNF}$ logaritmikus csökkenést mutat, mivel minél nagyobb a D^{SV}, annál kevésbé éri meg a VNF-et használni, hiszen ez a fix késleltetés negatív befolyással van az átlagos késleltetésre. A tehermentesítés esetében, amikor az átlagos késleltetés 0,10 *ms*, a csomagoknak csak 20%-a igényel továbbítást a VNF-hez, viszont ha D^{SV} egyenlő 0,10 *ms*-sel, akkor a tehermentesítés felére csökkentheti az átlagos késleltetést, ami a tehermentesítési képesség nélküli P4 hálózati kapcsolókkal szembeni előnyét jelenti.

4.4. C^{SC} hatása

A 4.4 ábrán látható a C^{SC} paraméter hatása az átlagos késleltetésre nézve *ms*-ban kifejezve. A 4.4 ábra alapján megállapítható, hogy ha csak VNF-et használunk ($\bar{D}^{Total}(1)$), akkor a Switch Computing kapacitásának (C^{SC}) méretét mindenképp méretezni kell az átlagos késleltetés minimalizálásához (0,250 *ms*-ről 0,125 *ms*-ra). Ennek oka, hogy a Switch Computing-nak van egy hozzárendelési funkcionalitása (melyik adott porton kerüljön a csomag kiküldésre a hálózati kapcsolónak), ahol minél kisebb a C^{SC} kiosztott kapacitása, annál lassabban tudja továbbítani a csomagokat a PNF és a VNF felé. Úgy tűnik azonban, hogy a C^{SC} mérete nincs nagy hatással az átlagos késleltetésre, abban az esetben, ha csak PNF-et használunk ($\bar{D}^{Total}(0)$), mivel a csomagok továbbítása döntés nélkül csak egy irányba történhet meg.



4.4. ábra. Optimális p^{VNF} paraméter különböző C^{SC} értékekhez képest.

A C^{SC} kapacitásának növelésével a VNF iránti igény is növekszik. Ennek oka, hogy a VNF használata szükségtelen, ha a Switch Computing kapacitás mérete viszonylag kicsi, ami az átlagos késleltetés növekedését okozza. Ilyen esetben a Switch Computing alulteljesít.

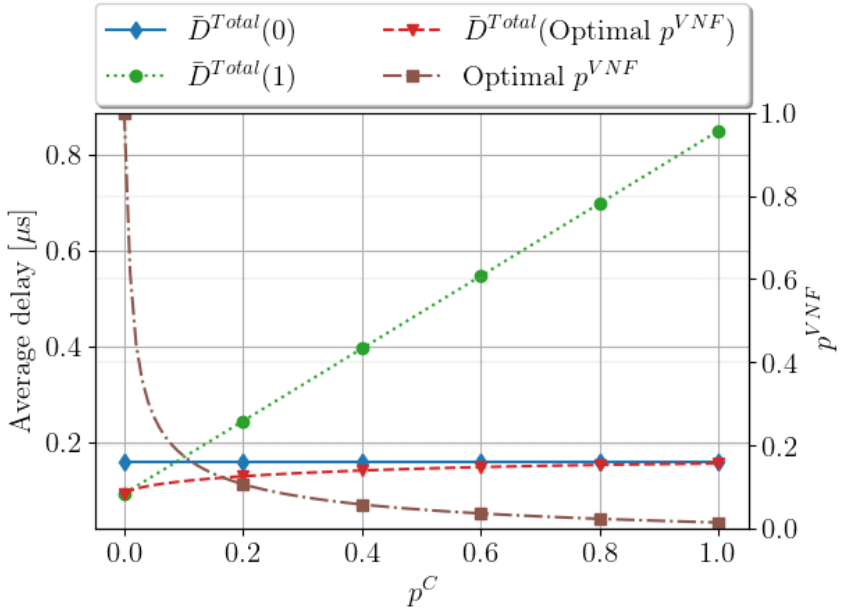
A C^{SC} növelésével (vagyis minél kisebb késleltetést okoz a kimeneten történő döntéshoza-

tal) arányosan egyre nagyobb az $Optimal p^{VNF}$ értéke. Ez azt jelenti, hogy az $Optimal p^{VNF}$ elérése akkor és csak akkor érhető el, ha a Switch Computing kapacitását növeljük.

Ezeknek ellenére a P4 hálózati kapcsolóról az NFV-re történő terhelésáthelyezés során az ideális eset eléréséhez a C^{SC}-t 600 pkt/ms-re kell skálázni. Az átlagos késleltetés majdnem fele akkorára csökkenthető (0,170 *ms*-ról 0,122 *ms*-ra), ha a PNF és az NFV egymást kiegészítve kerül alkalmazásra.

4.5. p^C hatása

A 4.5 ábra alapján az látható, hogy ha csak PNF-et használunk ($\bar{D}^{Total}(0)$) az átlagos késleltetés konstans (azaz 0,18 *ms* körüli). Másrészt, ha minden csomagot továbbítunk a VNF-nek, akkor az ($\bar{D}^{Total}(1)$) átlagos késleltetés lineáris növekedést mutat, amely a table miss valószínűségéhez (p^C) van viszonyítva. A p^C növelése kétségtelenül növeli a kontrollerhez továbbított csomagok számát, ez azonban a rendszer D^{SC} fix késleltetése mellett egy plusz késleltetési időt is jelent. Ettől függetlenül a kontroller valószínűségének lineáris növekedése az átlagos késleltetés nem lineáris növekedését eredményezi.



4.5. ábra. Optimális p^{VNF} paraméter különböző p^C értékekhez képest.

A 4.5–ábra alapján megállapítható, hogy a késleltetés minimalizálása érdekében a kontrollert a lehető legkevesebb csomagra kell felhasználni. Törekedni kell arra, hogy minimalizáljuk a kontroller szükségességét ($Optimal p^{VNF}$) és ha lehetséges magában a fizikai hálózati kapcsolóban kell az implementált logikát használni az ilyen kérések kiszolgálására. Ha ezek a feltételek rögzíthetők, akkor a PNF és a VNF együttes használata ($\bar{D}^{Total}(Optimal p^{VNF})$) előnyös az NFV-nek való tehermentesítési képességgel nem rendelkező P4 hálózati kapcsolókkal szemben ($\bar{D}^{Total}(0)$), bár az átlagos késleltetés csökkenése elhanyagolható. Ez az előny azonban a kontrollerigény növekedésével redukálódik.

Összegzés

A dolgozat keretén belül megvizsgáltam a P4 és az NFV integrációját. A P4 és az NFV kombinálásával a P4 kapcsolók hardveres erőforráskorlátjainak kezelésére törekedtem azáltal, hogy a csomagokat (és a flowokat) szükség esetén egy NFV rétegre tereljük feldolgozásra. Ebben a tekintetben a P4-ről az NFV-re történő terhelésáthelyezés hatásait vizsgáltam egy M/M/1 sorbanállási modell kidolgozásával és elemeztem annak érzékenységét a különböző paraméterek függvényében.

Az elemzésemből kiderül, hogy a P4-ről az NFV-re történő terhelésáthelyezés jelentős előnyökkel jár, ezért érdemes lehet a kettőt együttesen alkalmazni. Segítségével a rendszer átlagos késleltetése akár több mint a felére csökkenhet. Egy ilyen hibrid működés esetén jelentős késleltetés-csökkenést lehet elérni, ezáltal a felhasználói elégedettséget jelentősen lehet növelni, hiszen a lehető leghamarabb tudjuk az információt biztosítani számára, hogy az kielégítse az adat iránti igényt. A dolgozatomban tett megállapítások értékesek lehetnek az internetszolgáltatók (ISP), a hardvergyártók és a chipgyártók számára. Segíthet a kapcsolótervezőknek és a hálózati elemzőknek a teljesítménymérések meghatározásában és egyéni kapcsolási csővezetékek hatékonyabb tervezésében. Segítséget nyújthat továbbá a P4 és az NFV integrált telepítésével kapcsolatos előnyök és hátrányok, valamint kompromisszumok meghatározásában. Az eredmények iránymutatásként is szolgálhatnak a jobb hálózattervezéshez, architekturális döntésekhez és a kitelepítési választásokhoz.

A jövőbeni munkámban tervezem, hogy több, egyetlen NFV-t integráló P4 hálózati kapcsolót hasonlítok össze a jelenlegi munkában bemutatott rendszerrel (egyetlen P4 egyetlen NFV-vel). Továbbá, hogy Markov-lánccal analitikusan megvizsgáljam a match-action pipeline központosított működését, ahol minden egyes match-action csomópont egy puffert valósít meg. Továbbá tervezem a rendszerarchitektúra számítási és kommunikációs korlátjának számszerűsítését szimulációk és prototípus-implementációk segítségével.

Irodalomjegyzék

[1] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka és T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks", *Communications Surveys Tutorials, IEEE*, 16. évf., 3. sz., 1617–1634. old., 2014. márc.

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck és R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges", *IEEE Communications Surveys Tutorials*, 18. évf., 1. sz., 236–262. old., 2016.

[3] E. Tittel, *SDN vs. NFV: What's the difference?*, `https://www.cisco.com/c/en/us/solutions/software-defined-networking/sdn-vs-nfv.html`, Accessed: 2021-07-12, 2021.

[4] M. Condoluci és T. Mahmoodi, "Softwarization and virtualization in 5G mobile networks: Benefits, trends and challenges", *Computer Networks*, 146. évf., 65–84. old., 2018.

[5] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese és D. Walker, "P4: Programming Protocol-Independent Packet Processors", *SIGCOMM Comput. Commun. Rev.*, 44. évf., 3. sz., 87–95. old., 2014. júl.

[6] M. He, A. Basta, A. Blenk, N. Deric és W. Kellerer, "P4NFV: An NFV Architecture with Flexible Data Plane Reconfiguration", *2018 14th International Conference on Network and Service Management (CNSM)*, 2018, 90–98. old.

[7] A. Fahmin, Y.-C. Lai, M. S. Hossain és Y.-D. Lin, "Performance modeling and comparison of NFV integrated with SDN: Under or aside?", *Journal of Network and Computer Applications*, 113. évf., 119–129. old., 2018. júl.

[8] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin és W. K. Seah, "Modelling Software-Defined Networking: Software and hardware switches", *Journal of Network and Computer Applications*, 122. évf., 24–36. old., 2018.

[9] T. Benson, A. Akella és D. Maltz, "Unraveling the Complexity of Network Management", *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09 sor., Boston, Massachusetts: USENIX Association, 2009, 335–348. old.

[10] M. Casado, N. Foster és A. Guha, “Abstractions for Software-Defined Networks”, *Commun. ACM*, 57. évf., 10. sz., 86–95. old., 2014. szept.

[11] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky és S. Uhlig, “Software-Defined Networking: A Comprehensive Survey”, *Proceedings of the IEEE*, 103. évf., 1. sz., 14–76. old., 2015.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker és J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”, *SIGCOMM Comput. Commun. Rev.*, 38. évf., 2. sz., 69–74. old., 2008. márc.

[13] N. M. K. Chowdhury és R. Boutaba, “A survey of network virtualization”, *Computer Networks*, 54. évf., 5. sz., 862–876. old., 2010.

[14] Q. Duan, Y. Yan és A. V. Vasilakos, “A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing”, *IEEE Transactions on Network and Service Management*, 9. évf., 4. sz., 373–392. old., 2012.

[15] Y. Li és M. Chen, “Software-Defined Network Function Virtualization: A Survey”, *IEEE Access*, 3. évf., 2542–2553. old., 2015.

[16] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta és H. A. Chan, “Optimal virtual network function placement in multi-cloud service function chaining architecture”, *Computer Communications*, 102. évf., 1–16. old., 2017.

[17] ETSI DGS/NFV-SW A001, *Network Functions Virtualisation (NFV); Virtual Network Functions Architecture*, `https://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_nfv-swa001v010101p.pdf`, Accessed: 2021-07-13.

[18] Z. Bronstein, E. Roch, J. Xia és A. Molkho, “Uniform handling and abstraction of NFV hardware accelerators”, *IEEE Network*, 29. évf., 3. sz., 22–29. old., 2015.

[19] Z. Niu, H. Xu, L. Liu, Y. Tian, P. Wang és Z. Li, “Unveiling Performance of NFV Software Dataplanes”, *Proceedings of the 2nd Workshop on Cloud-Assisted Networking*, CAN ’17 sor., Incheon, Republic of Korea: Association for Computing Machinery, 2017, 13–18. old.

[20] The P4 Language Consortium, *P4 Language Specification*, `https://github.com/p4lang/p4-spec/releases/download/P4_16-v1.2.2/P4-16-spec.pdf`, Accessed: 2021-07-13, 2021.

[21] L. L. Peterson, C. Cascone, B. O’Connor, T. Vachuska és B. Davie, “Software-Defined Networks: A Systems Approach”, Systems Approach LLC, 2020. dec., 4. fej., 154. old.

[22] Netronome, *SmartNIC*, `https://www.netronome.com/products/smartnic/overview/`, Accessed: 2021-07-13.

[23] H. Wang, R. Soulé, H. T. Dang, K. S. Lee, V. Shrivastav, N. Foster és H. Weatherspoon, “P4FPGA: A Rapid Prototyping Framework for P4”, *Proceedings of the Symposium on SDN Research*, SOSR ’17 sor., Santa Clara, CA, USA: Association for Computing Machinery, 2017, 122–135. old.

[24] P. Benáček, V. Puš, H. Kubátová és T. Čejka, “P4-To-VHDL: Automatic generation of high-speed input and output network blocks”, *Microprocessors and Microsystems*, 56. évf., 22–33. old., 2018.

[25] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown és J. Rexford, “PISCES: A Programmable, Protocol-Independent Software Switch”, *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM ’16 sor., Florianopolis, Brazil: Association for Computing Machinery, 2016, 525–538. old.

[26] The P4 Language Consortium, *Reference P4 software switch (BMv2)*, `https://github.com/p4lang/behavioral-model`, Accessed: 2021-07-13.

[27] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon és M. Casado, “The Design and Implementation of Open VSwitch”, *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI’15 sor., Oakland, CA: USENIX Association, 2015, 117–130. old.

[28] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll és P. Tran-Gia, “Modeling and performance evaluation of an OpenFlow architecture”, *2011 23rd International Teletraffic Congress (ITC)*, 2011, 1–7. old.

[29] S. Azodolmolky, P. Wieder és R. Yahyapour, “Performance Evaluation of a Scalable Software-Defined Networking Deployment”, *2013 Second European Workshop on Software Defined Networks*, 2013, 68–74. old.

[30] Z. Bozakov és A. Rizk, “Taming SDN Controllers in Heterogeneous Hardware Environments”, *2013 Second European Workshop on Software Defined Networks*, 2013, 50–55. old.

[31] K. Mahmood, A. Chilwan, O. N. Østerbø és M. Jarschel, “On The Modeling of OpenFlow-based SDNs: The Single Node Case”, *CoRR*, abs/1411.4733 évf., 2014.

[32] M. Jarschel, O. Østerbø, A. Chilwan és K. Mahmood, “Modelling of OpenFlow-based software-defined networks: The multiple node case”, *IET Networks*, 4. évf., 2015. máj.

[33] W. Miao, G. Min, Y. Wu és H. Wang, “Performance Modelling of Preemption-Based Packet Scheduling for Data Plane in Software Defined Networks”, *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 2015, 60–65. old.

[34] Z. Shang és K. Wolter, “Delay Evaluation of OpenFlow Network Based on Queueing Model”, *CoRR*, abs/1608.06491 évf., 2016.

[35] K. Sood, S. Yu és Y. Xiang, “Performance Analysis of Software-Defined Network Switch Using $M/Geo/1$ Model”, *IEEE Communications Letters*, PP évf., 1–1. old., 2016. szept.

[36] W. Miao, G. Min, Y. Wu, H. Wang és J. Hu, “Performance Modelling and Analysis of Software-Defined Networking under Bursty Multimedia Traffic”, *ACM Trans. Multimedia Comput. Commun. Appl.*, 12. évf., 5s. sz., 2016. szept.

[37] B. Xiong, K. Yang, J. Zhao, W. Li és K. Li, “Performance evaluation of OpenFlow-based software-defined networks based on queueing model”, *Computer Networks*, 102. évf., 2016. ápr.

[38] Y. Goto, H. Masuyama, B. Ng, W. K. G. Seah és Y. Takahashi, “Queueing Analysis of Software Defined Network with Realistic OpenFlow–Based Switch Model”, *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2016, 301–306. old.

[39] J. Zhao, Z. Hu, B. Xiong, L. Yang és K. Li, “Modeling and optimization of packet forwarding performance in software-defined WAN”, *Future Generation Computer Systems*, 106. évf., 412–425. old., 2020.

[40] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin és W. K. Seah, “Full encapsulation or internal buffering in OpenFlow based hardware switches?”, *Computer Networks*, 167. évf., 107033. old., 2020.

[41] J. Walrand és P. Varaiya, “Sojourn times and the overtaking condition in Jacksonian networks”, *Advances in Applied Probability*, 12. évf., 4. sz., 1000–1018. old., 1980.

[42] J. B. Goodman és W. A. Massey, “The non-ergodic Jackson network”, *Journal of Applied Probability*, 21. évf., 4. sz., 860–869. old., 1984.

[43] J.-Y. L. Boudec és P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, Lecture Notes in Computer Science sor. Springer, 2001, 2050. köt.

[44] H. T. Dang, H. Wang, T. Jepsen, G. Brebner, C. Kim, J. Rexford, R. Soulé és H. Weatherspoon, “Whippersnapper: A P4 Language Benchmark Suite”, *Proceedings of the Symposium on SDN Research*, SOSR ’17 sor., Santa Clara, CA, USA: Association for Computing Machinery, 2017, 95–101. old.

[45] W. Shen, M. Yoshida, K. Minato és W. Imajuku, “vConductor: An enabler for achieving virtual network integration as a service”, *IEEE Communications Magazine*, 53. évf., 2. sz., 116–124. old., 2015.

[46] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta és V. Riccobene, “An open framework to enable NetFATE (Network Functions at the edge)”, *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015, 1–6. old.

[47] F. Callegati, W. Cerroni, C. Contoli és G. Santandrea, “Dynamic chaining of Virtual Network Functions in cloud-based edge networks”, *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015, 1–5. old.

[48] D. Hancock és J. van der Merwe, “HyPer4: Using P4 to Virtualize the Programmable Data Plane”, *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT ’16 sor., Irvine, California, USA: Association for Computing Machinery, 2016, 35–49. old.

[49] C. Zhang, J. Bi, Y. Zhou, A. B. Dogar és J. Wu, “HyperV: A High Performance Hypervisor for Virtualization of the Programmable Data Plane”, *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, 1–9. old.

[50] M. Saquetti, G. Bueno, W. Cordeiro és J. R. Azambuja, “P4VBox: Enabling P4-Based Switch Virtualization”, *IEEE Communications Letters*, 24. évf., 1. sz., 146–149. old., 2020.

[51] E. F. Kfoury, J. Crichigno és E. Bou-Harb, “An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends”, *IEEE Access*, 9. évf., 87094–87155. old., 2021.

[52] S. Boyd és L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. márc.

[53] R. P. Brent, “An algorithm with guaranteed convergence for finding a zero of a function”, *The Computer Journal*, 14. évf., 4. sz., 422–425. old., 1971. jan.

[54] W. H. Press, S. A. Teukolsky, W. T. Vetterling és B. P. Flannery, “Van Wijngaarden-Dekker-Brent Method”, *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd, Cambridge, England: Cambridge University Press, 1992, 9. fej., 352–355. old.

[55] X. Chen, D. Zhang, X. Wang, K. Zhu és H. Zhou, “P4SC: Towards High-Performance Service Function Chain Implementation on the P4-Capable Device”, *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, 1–9. old.

[56] C. Zhang, H. Yang, G. F. Riley és D. M. Blough, “Queueing Analysis of Auxiliary-Connection-Enabled Switches for Software-Defined Networks”, *2019 International Conference on Computing, Networking and Communications (ICNC)*, 2019, 497–502. old.

[57] C. Zhang, J. Bi, Y. Zhou és J. Wu, “HyperVDP: High-Performance Virtualization of the Programmable Data Plane”, *IEEE Journal on Selected Areas in Communications*, 37. évf., 3. sz., 556–569. old., 2019.