Médiaminőség-vizsgálat
hálózati forgalom veszteségmentes analízisével

Doktori (PhD) értekezés

Skopkó Tamás

Témavezetők:

Dr. Orosz Péter

Dr. Almási Béla †

Debreceni Egyetem

Természettudományi Doktori Tanács

Informatikai Tudományok Doktori Iskola

Debrecen, 2016.

Ezen értekezést a Debreceni Egyetem Természettudományi Doktori Tanács Informatikai Tudományok Doktori Iskola Informatikai Rendszerek és Hálózatok doktori programja keretében készítettem a Debreceni Egyetem természettudományi doktori (PhD) fokozatának elnyerése céljából.
Debrecen, 2016. június 1.

……
Skopkó Tamás
jelölt

Tanúsítom, hogy Skopkó Tamás doktorjelölt 2009-2016 között a fent megnevezett Doktori Iskola Informatikai Rendszerek és Hálózatok programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javasolom.
Debrecen, 2016. június 1.

……
Dr. Orosz Péter
témavezető

Médiaminőség-vizsgálat
hálózati forgalom veszteségmentes analízisével

Monitoring media quality
based on lossless traffic evaluation

Értekezés a doktori (Ph.D.) fokozat megszerzése érdekében
informatikai tudományok tudományágban

Írta: Skopkó Tamás okleveles programtervező matematikus

Készült a Debreceni Egyetem
Informatikai Tudományok Doktori Iskolája
Informatikai Rendszerek és Hálózatok programja keretében

Témavezetők:
Dr. Orosz Péter
Dr. Almási Béla †

A doktori szigorlati bizottság:

elnök:	Dr. Végh János	……
tagok:	Dr. Fehér Gábor	……
	Dr. Fazekas Gábor	……

A doktori szigorlat időpontja: 2014. július 9.

Az értekezés bírálói:

……	……
……	……

A bírálóbizottság:

elnök:	……	……
tagok:	……	……
	……	……
	……	……
	……	……

Az értekezés védésének időpontja:. 2016. …… … .

Tartalomjegyzék

Köszönetnyilvánítás

Köszönöm témavezetőmnek, kutatótársamnak és egyúttal jó barátomnak, **Orosz Péter**nek az elmúlt évek közös munkáját, a tanulságos beszélgetéseket. Emellett a rengeteg türelmet, amellyel a szakmai fejlődésemet, kutatói utamat támogatta. A közös projektek sikerélménye az önbiztalmamat fejlesztették egy-egy jól teljesített feladat után.

Köszönöm **kutatótársaim**nak a közös munkát. Úgy gondolom, az **ATMA** kutatócsoportban egy igazán inspiráló közösségben alkothattam.

Köszönöm másik témavezetőm, **Almási Béla** szakmai tanácsait, útmutatásait.

Köszönöm **szüleim és testvérem** sokrétű támogatását és bíztatását. Köszönöm párom, **Szilvia** fáradhatatlan lektorálási kedvét a dolgozat sokadszori átolvasásában és azt a leírhatatlanul sok szeretetet, türelmet és figyelmet, ami újra és újra energiát adott.

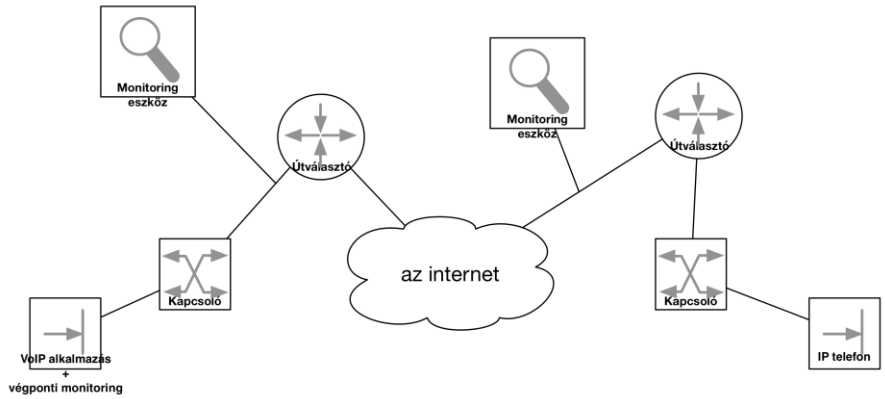
...és mindazon kollegáim, barátaim, ismerőseim segítségét, akik közvetve mindannyian segítettek biztosítani munkámhoz a megfelelő körülményeket.

1 Bevezetés

Az internetet, mint általános célú adattovábbító infrastruktúrát, egyre több és több alkalmazás használja. Az alkalmazások felhasználói ma nemcsak az elégséges működést, hanem a minél jobb felhasználói élményt is elvárják. Ez az igény még inkább jelen van a fizetős szolgáltatások esetében. Ezért az ipari szereplők, mind a hálózati infrastruktúra üzemeltetői, mind az alkalmazások szolgáltatói a magas szintű felhasználói élmény biztosítására törekszenek. Az egyre összetettebb hálózatok aktuális és jövőbeli igényeknek is megfelelő folyamatos hangolásához és fejlesztéséhez a szolgáltatóknak olyan eszközökre van szükségük, amelyek mérhető paraméterek segítségével naprakész, áttekintő képet adnak az infrastruktúra és a szolgáltatás állapotáról. E mutatók folyamatos követésével pedig hosszútávú trendek vázolhatók fel, amelyek a fejlesztési stratégiát is meghatározhatják.

A Quality of Service (QoS, szolgáltatási minőség) a publikus internet elterjedésével párhuzamosan fejlődő alapvető eszközrendszer, amely a hálózati teljesítménymutatók mérésére és meghatározására szolgál. Olyan metrikák gyűjteménye, amelyet a korszerű hálózati infrastrukturális aktív eszközök mintavételezéssel tudnak mérni. A legfontosabb QoS mutatók: az útvonali késleltetés (latency), érkezési ingadozás (jitter), csomagvesztés (packet loss), átviteli ráta (throughput), csomagátrendeződés (reordering) és duplikáció (duplication). Az internetszolgáltató (Internet Service Provider, ISP) saját eszközeit monitorozva a begyűjtött QoS metrikák értékeit mint adatsorokat tárolja. A passzív megfigyelésen túl a metrikákhoz határértékeket állapíthat meg, amelyeket meghaladva lépéseket tehet a hálózat erőforrásainak átcsoportosítására. Egy alkalmazás szolgáltatója szintén képes QoS mutatókat mérni, de jellemzően csak azokon a végpontokon, ahol a saját alkalmazásai is futnak (1.1. ábra). Továbbá az is jellemző, hogy az egymással kapcsolatban álló szolgáltatók nem osztják meg mért adataikat egymással, és nem létezik széles körben elfogadott interfész és eljárás sem, amely biztosítaná egy heterogén, több szolgáltatót is igénybe vevő szolgáltatás esetén az egységes, globális tájékozódást.

A QoS eszközrendszer már a kezdetek óta a halózatokkal kapcsolatos kutatások egyik fő fókuszpontja. A témakörben született publikációk rámutattak, hogy a szolgáltató hálózatában pusztán a metrikák értékeinek javulása nem feltétlenül eredményezi, hogy az alkalmazások minősége is érezhetően javulni fog [**1**]. Sokáig a hálózati infrastruktúra üzemeltetők az akkut csomagvesztésre a hálózati sávszélesség növelésével reagáltak, ám a beruházástól remélt minőségi javulás sok esetben elmaradt.



1.1. ábra QoS monitorozás az infrastruktúra különböző pontjain

A végfelhasználó rendszerint nem tudja, és nem is kell ismernie, hogy az általa használt szolgáltatások milyen bonyolult infrastruktúrán működnek. Ő leginkább abban érdekelt, hogy az általa igénybevett szolgáltatással elégedett legyen. Fizetős szolgáltatás esetén az ár/érték arány is fontos szempont. A felhasználó nézőpontját feltárni próbáló vizsgálati módszereket Quality of Experience (QoE, érzeti minőség) fogalom alatt tartjuk számon. Ez a szintén régóta aktív kutatási terület kezdetben elsősorban szubjektív vizsgálati módszereket nyújtott. Ezek a módszerek a felhasználó megkérdezésével és a leadott értékelések statisztikai összesítésével adtak képet, rendszerint egy adott szolgáltatás érzeti minőségéről. Mivel a szubjektív értékelés rendszeres végrehajtása meglehetőségen körülményes és költséges, ráadásul az azonnali visszacsatolás nagyon nehezen oldható meg, a QoS–QoE összekapcsolását megkísérlő kutatási területek létező és fontos szolgáltatói igényeket is kielégíthetnek [**2**]. Ezek a módszerek a QoS mérhető paramétereit próbálják megfeleltetni a QoE érzeti minőségi értékeinek.

Kutatásaim célkitűzése szolgáltatások és médiaátvitel érzeti minőségét becslő módszerek, ill. ezek alapjául szolgáló metrikák létrehozása. A módszerek kidolgozásához adatgyűjtés és elemzés szükséges. Az adatgyűjtést a hálózati forgalom monitorozásával és a forgalom elkapásával végzem. A veszteségmentes forgalom-megfigyelés és -gyűjtés munkám során mindvégig fontos szempontként szerepelt. A begyűjtött forgalmi mintázatok hatékony analíziséhez ugyanis lényeges a megfelelő felbontású és részletességű adatsorok megléte. Eleinte az elérhető céleszközök híján olyan módszerekre volt szükség, amelyekkel a hozzáférhető, jellemzően általános célú architektúrákon is végezhető, lehetőleg veszteségmentes hálózati forgalmi mérés. A mérések egyik fontos eleme az időbélyegzés, amely a forgalmi mintázat rekonstrukciója során fontos szerepet játszik. Az adott környezet és sávszélesség mellett kielégítő pontosságú időbélyegzés megvalósítása kihívást jelentett. Mindemellett a végponti időbélyegzés többlet erőforrásigénye negatívan befolyásolta az elemezni kívánt rendszer működését is.

Ezért olyan módszereket dolgoztam ki, amelyek javították a rendelkezésünkre álló mérőeszközökön elérhető időbélyegzési pontosságot, valamint a rendszerek csomagfeldolgozási hatékonyságát. A kutatások ezen szakaszának végére a kutatócsoportunk *Rnetprobe* néven olyan mérőeszközt készített, amely képes 1 Gbit/s linksebességű kapcsolaton a forgalom veszteségmentes monitorozására, az e sávszélességen megkövetelt időbélyegzési pontosság mellett [**C6**].

Később egy nagyobb projekt keretében olyan mérőeszköz fejlesztésében vettem részt, amely szolgáltatói maghálózatokban is képes összetett monitoring funkciók elvégzésére. Ebben a projektben számomra a kihívást az összegyűjtött és előfeldolgozott adatok elosztott tároló és utófeldolgozó végpontokra történő eljuttatása jelentette. A cél megvalósításához egy, az adott környezet támasztotta igényekhez szabott transzport protokollt terveztem és implementáltam.

Az említett két projekt során sok tapasztalatot gyűjtöttem a QoS forgalommérési területen. Az akkoriban még nagyon friss Opus hangkódoló adta az ötletet, hogy megfelelő mérési technikák segítségével az alkalmazások szemszögéből is vizsgálhassuk az érzeti minőséget. Így kutatásaim fókuszába a

hálózati forgalom analízise került. Az Opus hangkódolóval kapcsolatos vizsgálati módszerek eredményeként a hálózati forgalommérés (QoS) és forgalomanalízis (QoE) területeket sikerült összekapcsolni.

A dolgozat 2. fejezetében egy olyan, hálózati forgalom mérését segítő időbélyegzési módszert ismertetek, amely a szoftveres időbélyegzés pontosságát és az időbélyegzés alkalmazása mellett történő csomagfeldolgozás hatékonyságát javítja. A disszertáció 3. fejezetében bemutatott protokoll szintén hálózati forgalommérő rendszerek számára készült, de akár egyéb területeken is kiaknázható. Az ismertetett protokoll kis erőforrásigénye mellett képes más transzport protokollokkal összemérhető átviteli teljesítmény elérésére, a csomagvesztés elkerülése mellett. A 4-5. fejezetben egy, a forgalom analízise során alkalmazható érzeti minőségbecslő eljárást ismertetek. Az eljárás a forrásanyag rendelkezésre állása nélkül képes az Opus hangkódolót alkalmazó internet feletti interaktív hangtovábbító alkalmazás érzeti minőségét megbecsülni. A függvény kidolgozásakor alkalmazott módszer további VoIP hangkódolók érzetiminőség-becslő módszerének elkészítéséhez is segítséget nyújthat.

2 Optimalizált szoftveres időbélyegzés

Az internet univerzális adattovábbító közművé formálódása már jó ideje tart. Ezt az információs csatornát mind az általános célú alkalmazások (web böngészés, email, stb.), mind a speciális, üzleti célú felhasználás közösen kívánják kiaknázni. Az egyre szélesebb körben és nagyobb sávszélességgel hozzáférhető interneten folyamatosan bővülnek az alkalmazási területek is. Mivel az internet a szolgáltatók heterogén csomagkapcsolt hálózataiból formálódott, így az áramkör kapcsolt hálózatok (pl. kapcsolt közcélú telefonhálózat (Public Switched Telephone Network, PSTN), Global System for Mobile Communications, (GSM), Universal Mobile Telecommunications Systems, UMTS) világában megszokott minőségi garanciák infrastrukturális szinten nem adottak. Így folyamatos kihívást jelent e garanciák biztosítása az egyre nagyobb arányban megjelenő, speciális igényeket támasztó multimédiás és egyéb, időzítésre érzékeny (jellemzően interaktív és valós idejű) alkalmazások számára. A QoS-paletta olyan metrikákból és módszerekből áll, amelyeket az infrastruktúra üzemeltetők alkalmazhatnak a teljesítményparaméterek mérésére, illetve azok garantálására. Szolgáltatói oldalon ezekkel könnyen megoldható a monitoring, és mind szolgáltatói, mind fejlesztői szempontból a végponton is hasznosíthatók. A végponti QoS méréséhez bár elegendő a minden rendszerben elérhető 10^{-3} s pontosságú időbélyegzés, ugyanakkor nagy kihívást jelent annak hatékonysága, hogy minél kevesebb erőforrást (számítási teljesítményt) vonjon el az eszközön futó egyéb folyamatoktól, illetve végrehajtása minél kevésbé interferáljon azokkal. Az időbélyegzés során a beérkező csomagokhoz metaadat rendelődik, amely egy vagy több időpontot jelöl meg (pl. a rendszerben történő megjelenést) [**C8**]. Segítségével rekonstrukálhatók a csomagok érkezési időközei, vagy a rétegelt hálózati architektúra egyes rétegeiben végbemenő folyamatok.

A végponti eszközök mára nemcsak személyi számítástechnikát (personal computing architectures) kielégítő hardvereket jelentenek, amelyek teljesítménye bőven elegendő az időbélyegzéshez, hanem olyan, teljesítményben jóval korlátosabb mobil- és beágyazott eszközöket is lefednek, amelyek általános célú operációs rendszerre (pl. a Linux-bázisú Androidra) alapozva szintén összetett szoftverkörnyezetet futtatnak. Ezekben a

rendszerekben az időbélyegzés teljesítményigénye már számottevő is lehet, ugyanakkor legtöbbjük szintén több processzormaggal rendelkezik, viszont ez nem jelenti a számítási kapacitás lineáris skálázódását [**3**] [**4**] [**5**]. Ugyanakkor, csomagfeldolgozás tekintetében a mai általános célú operációs rendszerek hálózati interfészenként (esetleg várakozási soronként) egyetlen processzormag használatára vannak korlátozva. Ezért az olyan módszerek, amelyek a szoftveres időbélyegzés pontosságát és terheléselosztását javítják, hasznosak lehetnek a végponti QoS biztosításában. Léteznek hardver segítségével időbélyegző eszközök, kártyák is. Ezek viszont meglehetősen költségesek és nem nyújtanak kellő szabadságot azt illetően, hogy milyen eszközzel és a csomagfeldolgozás mely műveletéhez kötve készítsünk időbélyeget. A hardver alapú időbélyegző eszközök az időbélyeget jellemzően a csomag első bitjének fizikai rétegben (Physical Layer, PHY) történő megjelenéséhez társítják, de implementációtól függően más eseményhez is kapcsolható. Szoftveres időbélyeg esetén a tárolt időpont jellemzően a csomagnak a csomagkezelő rendszerben történő megjelenéséhez, esetleg hozzáférhetőségéhez kötődik. Egy alkalmazás vizsgálatakor a szoftveres időbélyeg szolgáltathat több információt, míg aggregált hálózati forgalom esetén a fizikai rétegben történő megjelenés időpontja adhat részletesebb képet.

2.1 Kapcsolódó kutatások

2.1.1 Csomagelkapás szoftveres eszközökkel

A *TcpDump* az első forgalomelemzést segítő eszközök egyike [**6**]. Alapja a *LibPcap* könyvtár, amely programozási felületet (Application Programming Interface, API) biztosít az operációs rendszer kapcsolódó csomagfeldolgozó rutinjainak használatához [**7**]. Ezek a könyvtárak az egyik legsokoldalúbb hálózati forgalom analizáló eszköz, a *Wireshark* alapját is képezik [**8**]. Ez az alkalmazás számos protokoll mélyebb szintű vizsgálatát is lehetővé teszi, ezért mind az alkalmazásfejlesztők, mind a protokollokat készítő szakemberek számára hatékony eszközként tartják számon.

A csak Linux platformon elérhető *PF_RING* könyvtár a LibPcap-hez viszonyítva nagyobb hatásfokú csomagelkapást tesz lehetővé, a csomag metaadat másolásainak csökkentése segítségével [**9**] [**10**].

N. Bonelli és társai *PFQ* néven több processzormagos rendszereket kiaknázó, a PF_RING hiányosságait kompenzáló csomagelkapó motort készítettek [**11**].

J. L. García-Dorado és társai átfogó elemzést készítettek az általános célú rendszerekhez rendelkezésre álló hálózati forgalmat monitorozó megoldásokról [**12**]. Rávilágítottak az egyes módszerek előnyeire, illetve azok szűk keresztmetszeteire, továbbá a több processzormagos rendszerekben levő kihasználatlan lehetőségekre is.

L. Deri és társai *Ntopng* néven egy nyílt forráskódú, nagy sávszélességű aggregált hálózatokban is használható monitorozó rendszert fejlesztettek ki, melynek segítségével egyszerűen elemezhetők a fontos teljesítménymutatók és könnyebb a szűk keresztmetszetek felderítése [**13**].

Az általános célú architektúrák csomagtovábbítási megoldásainak szűk keresztmetszeteit kutatták P. Emmerich és társai [**14**]. Munkájukban a szoftver architektúrából fakadó korlátok mellett a hardver jelentette nehézségekre is rávilágítottak.

2.1.2 Csomagelkapás hardver gyorsítással

Az *Endace DAG* (Data Acquisition and Generation) kártyák általános célú architektúrákon használható, nagy teljesítményű csomaggenerálásra és elkapásra tervezett, emellett hálózati csatolóként (Network Interface Card, NIC) is működőképes eszközök [**15**]. A hardveres gyorsításnak köszönhetően képesek a csomagok hardverrel történő időbélyegzésére is. Így akár aggregált hálózati kapcsolatokon történő forgalomgyűjtésre is használhatók, ahogy ezt A. A. Heyde a jogszerű hálózati forgalom megfigyelése kapcsán is vizsgálta [**16**].

A *NetFPGA* platform rugalmassága és kedvező ár/érték aránya megnyitotta az utat a hálózati funkciók hardverben történő implementálásához. G. A. Covington és társai NetFPGA alapú csomaggenerátort hoztak létre, amely a beérkező csomagok időbélyegzését is hardver segítségével végzi [**17**].

Részt vettem a NetFPGA alapú *Rnetprobe* forgalom monitorozó rendszer implementálásában [**C6**]. Ebben a rendszerben szoftveresen előállított

időbélyeg használata mellett lehetőség van hardverrel előállított időbélyegek alkalmazására is. A hardver alapú időbélyegeket a NetFPGA belső órajele segítségével állítja elő. A saját órajelforrásból származtatott időt a rendszer a rendszerórához szinkronizálja.

Mivel a dedikált hardverkártyás megoldások a rendszerétől eltérő időforrást alkalmaznak, ezért az órák rövid- és hosszútávú együtt futásának, szinkronizálásának problémáját is felvetik. NetFPGA platformon Z. Zhou és társai készítettek egy lehetséges, a PCI buszon keresztül történő szinkronizálást segítő módszert [**18**].

2.1.3 A szoftveres időbélyegzés kihívásai

V. Moreno és társai a csomagfeldolgozás időbélyegzési pontosságra gyakorolt hatását vizsgálták [**19**]. Kimutatták, hogy 10 Gbit/s és nagyobb kapacitású összeköttetés esetén a kötegelt csomagfeldolgozás következtében az időbélyegek egyre pontatlanabbá válnak. A pontosság javításához egy olyan összetett módszert mutattak be, amely egyfelől a tömegesen beérkező csomagok között „szétteríti" az időbélyegeket, másfelől a hálózati meghajtóprogram áttervezésével egy olyan rendszermag privilégiumokkal futó szálat működtet, amely a hálózati csatoló puffereit folyamatosan lekérdezi és azonnal időbélyegzi a beérkező csomagokat.

B. Villain és társai a csomagfeldolgozás különböző pontjain mért szoftveres időbélyegzéskor tapasztalt késleltetéseket vizsgálták és a hálózati csatoló saját időbélyegeit használták referenciaként [**20**]. Megkeresték azt a pontot, ahol a legkisebb késleltetéssel készíthetők szoftveres időbélyegek. Erre alapozva javítható egyes szoftveres óraforrásokra támaszkodó alkalmazások (pl. időszinkronizációs protokollok) hatékonysága.

I. Fedotova és társai az általános architektúrákon elérhető időforrások pontosságát kutatták és egy pontos időkezelést segítő rutinkönyvtárat készítettek [**21**].

Célom egy hatékonyabb időbélyegzési módszer készítése volt, amely már a meglevő szoftveres csomagelkapási könyvtárakra épül, és nem igényli a csomagfeldolgozó alrendszer radikális átalakítását. A fenti kutatások bár

érintették a csomagelkapás és időbélyegzés problémakörét, közvetlen megoldást nem nyújtottak erre a kihívásra.

2.2 Az abszolút idő mérése és ábrázolása

Az aktuális idő követése alapvető fontosságú az informatikai rendszerekben, hiszen nagyon sok művelet végrehajtása időhöz kötött vagy időzített módon történik, esetleg adott időn belül kell végrehajtódnia. A rendszerekben naplózott eseményeket is a hozzájuk kapcsolt idő metaadat segítségével tudjuk visszakövetni. Ezért minden rendszernek alkalmaznia kell egy helyi időforrást, amely gondoskodik a rendszeridő folyamatos nyilvántartásáról. Az egymással összekapcsolt rendszereknek közös referenciaidőhöz kell igazodniuk, különben a több rendszerben tárolt és egymáshoz viszonyítandó információk időbeniségét nem lehet egyértelműen rekonstruálni.

A gyakorlatban a Greenwich Mean Time (GMT) illetve Coordinated Universal Time (UTC) abszolút idő használata a jellemző. A GMT legfontosabb jellemzője, hogy egy jól definiált, a Föld forgásához és meghatározott pontjához kapcsolt idő a referencia, ezzel szemben az UTC alapja rendszerint valamilyen nagy pontosságú időforrás (pl. atomóra), éppen ezért minden, nem csillagászati vonatkozású idő méréséhez használható referenciaként. A GMT-t leginkább emberi vonatkozású időmérésnél alkalmazzák, az UTC rendszerint digitális órák szinkronizálásához.

Legtöbb esetben szoftver architektúrális opció, hogy az GMT-t, az UTC-t, vagy a valamelyikből geolokáció alapján származtatott helyi időt tartják-e nyilván. Ezért a rendszerek közötti időinformáció összevetésekor fontos, hogy helyi vagy abszolút időről van-e szó.

A rendszeridő számítása valamilyen óraforrás alapján történik. Mivel az óraforrás frekvenciája számos körülmény hatására (pl. hőmérséklet, tápellátás stabilitása stb.) ingadozást mutathat, az ebből származtatott idő sem tekinthető önmagában megbízhatónak. Az előbbi körülmények miatt fellépő, a referencia időhöz viszonyított rendszeres eltérést ingadozásnak (jitter) hívjuk. Emellett az óraforrás effektív órajele sem mindig egyezik a névlegessel, ezért lassabban és gyorsabban is járhat, mint a referencia idő, amit csúszásnak (skew) nevezünk.

Az előbbi problémák a rendszeridőnek egy referencia időhöz történő szinkronizálásával oldhatók meg. Ez egy nagyon megbízhatónak tekintett időmérő eszköz (pl. atomóra) lekérdezésével történik. Történhet egyszeri, a rendszer indításának időpontjában, de teljeskörű megoldást a folyamatos szinkronizálás jelenthet, mely rendszeres lekérdezéssel és visszacsatolás alkalmazásával valósulhat meg. Ethernet környezetben a Network Time Protocol (NTP) elsősorban alkalmazás szintű óraszinkronizációt valósít meg. Az elérhető pontosság nagy mértékben függ a rendszermag ütemezőjétől, ezért nem ad minden alkalmazási területen valódi garanciát [**22**]. Ugyanakkor a protokollt privilegizált végrehajtási módban futtatva az elérhető pontosság növelhető [**23**]. Magasabb szintű pontosságot biztosít a Precision Time Protocol (PTP), amelyet kifejezetten időzítésre érzékeny környezetbe ajánlanak, mert olyan megoldásokat tartalmaz, amelyek az infrastruktúrán és az eszköz helyi időforrásában fellépő ingadozás kezelését is megcélozzák. E protokollt jellemzően hardverben vagy rendszermag folyamat szintjén implementálják [**24**].

2.3 Az időbélyeg előállításának folyamata

Szoftveres időbélyegzés során az időbélyeg a rendszeridő lekérdezése után kerül a csomagokra. Egy rendszerben az időt szolgáltató alrendszerek általában egyszerű órajel számlálók, amelyek önmagukban nem szolgáltatnak pontos időt. Ezeket az órajel forrásokat a frekvenciájuk ismeretében időméréshez is tudjuk használni. A pontos idő követéséhez szükség van egy referencia időforráshoz történő (egyszeri, rendszeres vagy folyamatos) szinkronizálásra is. Így a tényleges időbélyegzés két fő részműveletre bontható: i.) az órajelforrás lekérdezése (clock read) és ii.) a kapott nyers delta ciklusszám UTC időre történő alakítása (clock conversion). Az óraforrás jellemzően valamilyen hardver komponens, amelyhez a hozzáférés a megfelelő illesztőfelületen, meghajtóprogramon, esetleg processzor regiszteren keresztül történik. Minél rövidebb az óraforrás hozzáférési ideje és ez a hozzáférési idő minél kisebb szórást mutat, annál alkalmasabb a pontos időbélyegzéshez való felhasználásra. A ciklusszámláló frekvenciája pedig meghatározza az időmérési pontosságot, ezáltal az időbélyegét is.

Az abszolút idővé alakítás erőforrásigénye az ábrázolási pontosság függvénye. Befolyásolhatja az is, ha a rendszermagból hiányzik a lebegőpontos műveletek támogatása.

Definíció: Egy művelet költsége (cost of operation, jelölje C) az a processzoridő (órajelciklusok száma), amelyet a végrehajtásával tölt a processzor. Komplex szoftverkörnyezetben, amilyenek az általános célú architektúrák is, ez a költség valamilyen tartományban szóródik, mivel akár az ütemező, akár egy hardver megszakítás az adott művelet végrehajtását bizonyos időre felfüggesztheti. Az időbélyegzés költsége (Timestamping Time) az időbélyegzés fázisainak összes végrehajtási költsége, jelölje C_t.

$$C_t \geq C_r + C_c \tag{1}$$

ahol C_r az óraforrás kiolvasási ideje, C_c az óraforrás nyers delta értékének abszolút idővé konvertálásához szükséges idő. Mivel a konverzió elemi műveletei az ütemező által megszakíthatók, a tényleges időbélyeg-előállítási idő a két fázis költségének összegétől nem lehet kisebb.

Az időbélyegzés tekintetében fontos metrika annak felbontása (Timestamping Resolution), jelölje R_t. Minden helyi órajelforrást jellemez annak felbontása, vagyis az időforrásban nyilvántartott számláló információ (pl. regiszter) ábrázolási pontossága, finomsága (granularity). A helyi óraforrás és a kiolvasott érték tárolására szolgáló adatszerkezet ábrázolási felbontásának együttese határozza meg az előállított időbélyeg felbontását. A felbontás jellemzi, milyen finomságban tudjuk előállítani az időbélyeget. Az időbélyeg felbontása függ az óraforrás felbontásától, az ahhoz való hozzáférési időtől, a kapott értéket tároló adatszerkezet kapacitásától és az átalakítás pontosságától:

$$R_t = min(R_r, C_r, S, R_c) \tag{2}$$

ahol R_r a nyers órajelforrás felbontását, C_r az órajel kiolvasásához szükséges időt (órajelciklusokban mérve), S az időbélyeg tárolásához szükséges tárterület méretét és R_c az abszolút idő felbontását jelöli.

Amennyiben egy 1 Gbit/s kapacitású Ethernet kapcsolat forgalmát szeretnénk elemezni, úgy a legkisebb bruttó 72 byte méretű csomagokból akár 1 488 096 is érkezhet másodpercenként. Ez azt jelenti, hogy 672 ns-onként érkezhet egy új csomag. A forgalmi mérést szerver oldalon végezve még kisebb

érkezési időközökkel számolhatunk a jelenleg elterjedt 10 Gbit/s sávszélességű kapcsolatokon. Vagyis a teljeskörű vizsgálathoz nem elegendő a 10^{-6} másodperces skála, 10^{-9} másodperces felbontásra volna szükség, továbbá olyan adatszerkezetre, amely képes a 10^{-9} felbontású idő ábrázolására (pl. 32+32 bit) [**C1**].

Az időbélyegzési precizitás (Timestamping Precision) metrikája a mérési eredmény szórásának feleltethető meg. Ezt az óraforrásból származtatott idő egy referenciának tekintett időforrással való együttfutása határozza meg (2.1. ábra).

$$P_r = \sigma_C \tag{3}$$

ahol P_r az órajelforrás kiolvasási pontosságát, σ_C az óraforrás szórását jelöli.

Az óraforrás kiolvasásának költsége is mutathat szórást, mivel az ütemező és a megszakításkezelés is hatással van rá. Az időbélyegzés precizitása így az óraforrás precizitása és a hozzáférés költségének szórása:

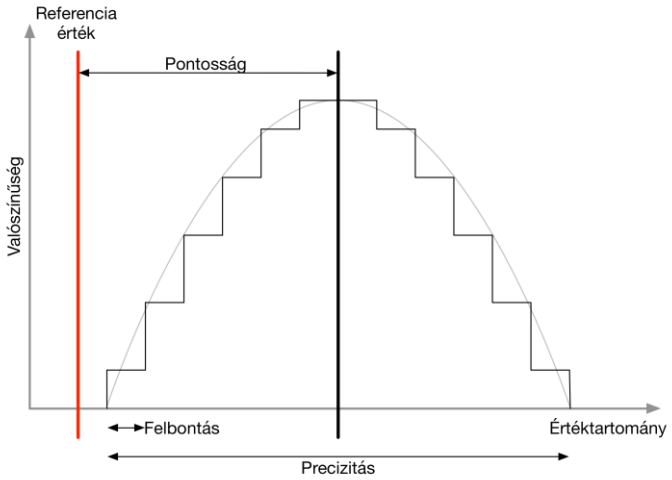
$$P_t = P_r + \sigma_r \tag{4}$$

ahol P_t az időbélyegzési precizitást, és σ_r az óraforrás kiolvasásához szükséges idő szórását jelöli.

A harmadik időbélyegzési metrika az időbélyeg pontossága (Timestamping Accuracy, offset), amely az előállított abszolút időbélyeg helyi referencia időforráshoz képesti eltérését jellemzi. Minél kisebb ez az eltérés, annál inkább a megjelölni kívánt esemény bekövetkezésének valós idejéhez közeli időbélyegeket állítunk elő. A referencia óraforrás jellemzően egy, a rendszeren kívüli óra. Felbontása praktikusan legalább a helyi óraforrásával megegyező.

$$A_t \geq max(C_t) \tag{5}$$

ahol A_t az időbélyegzés pontosságát és C_t az időbélyegzési folyamat végrehajtásához szükséges időt jelöli.



2.1. ábra Precizitás és pontosság: a mért értékek viszonya a referenciához képest

A metrikák ismeretében az időbélyegek akkor állnak legközelebb az ábrázolni kívánt tényleges időpontokhoz, ha minél kisebb az időbélyegzési folyamathoz szükséges idő (2.1. ábra). A hatékony időbélyegzés kellően nagy felbontás mellett az órajelforrást alacsony elérési idővel és az elérési idő minimális szórásával kérdezi le, valamint alakítja a lekérdezett nyers időértéket rendszeridővé.

2.3.1 Végrehajtási kontextusok

Az operációs rendszerek rendszerfolyamatai megkülönböztetett környezetben (kernel kontextusban), privilegizálva futnak a hatékony működés érdekében. Csomag érkezésekor megszakítás generálódik, a végrehajtás egy kitüntetett módba, megszakítási kontextusba vált. A rendszer ebben az állapotban van addig, amíg a kapcsolódó műveleteket (pl. a csomagok regisztrálása) el nem végezte. Megszakítási kontextusban az adott processzormagon minden más folyamat végrehajtása szünetel. Hatékonysági okokból célszerű, ha minél rövidebb ideig tartózkodik a rendszer ebben a kizárólagos állapotban. Felhasználói végrehajtási módban a kernelen kívüli, felhasználói folyamatok futnak, amelyek ütemezéséért a rendszermag ütemezője a felelős. Fontos tényező, hogy a kernel kontextusai és a felhasználói végrehajtási mód közötti adatmozgatás extra időbeli költséggel jár, ezért (1)-ben a kontextusváltások

járulékos költsége miatt legalább $C_r + C_c$ költséggel kell számolnunk az időbélyegzés mellett.

2.3.2 Csomagfeldolgozási kapacitás

Minden rendszer rendelkezik fizikai (pl. számítási kapacitás) korlátokkal. Az egyre fejlődő hálózati technológiákkal párhuzamosan egyre nagyobb linksebességek állnak rendelkezésre. A megnövekedett sávszélesség természetesen csak a rendelkezésre álló számítási teljesítmény mellett szolgálható és aknázható ki. Általános célú architektúrákban ez a számítási teljesítmény megoszlik a rendszermag és az alkalmazások között.

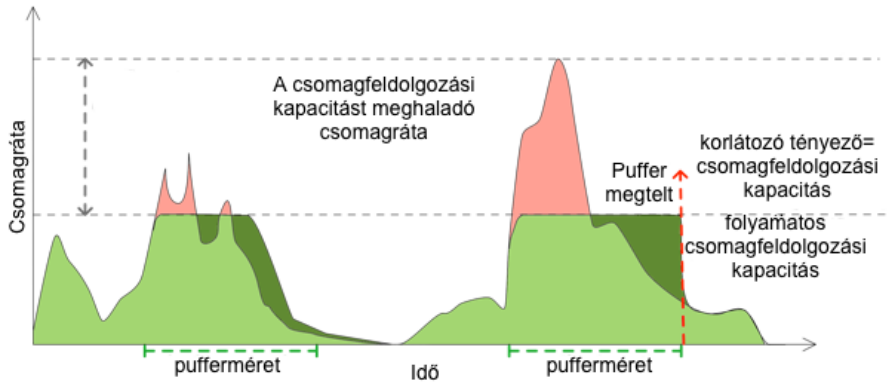
$$P = p_P / p_T \times 100 \tag{6}$$

ahol P a csomagfeldolgozási teljesítményt (%), p_P a feldolgozott csomagok számát (pps) és p_T az adott linkkapacitás mellett elérhető legmagasabb csomagrátát (pps) jelöli. Az egy csomagra jutó csomagfeldolgozási költséget C_p-vel jelölve, az érintett processzormag legfeljebb $p_T \times C_p$ processzoridőt tölthet csomagfeldolgozással.

Mivel P nem nőhet 100% fölé, továbbá az időbélyegzésnek is van saját költsége (C_t), ezért az időbélyegzés bekapcsolásával az egy csomagra jutó csomagfeldolgozási költség $C_p + C_t$-re nő. Ezáltal a csomagfeldolgozási kapacitás (p_T')-re csökken.

$$P' = p_P / p_T' \times 100 < P \tag{7}$$

ahol P' az időbélyegzés melletti csomagfeldolgozási teljesítményt jelöli.



2.2. ábra A csomagfeldolgozási kapacitás viselkedése a forgalomalakítóhoz hasonlóan

A rendszert számítási kapacitásának határán használva az óraforrás elérési idejének szórása megnövekszik, ezért az időbélyegzés pontossága romolhat, valamint az általa felemésztett erőforrás ronthatja az alkalmazás, illetve az igénybe vett szolgáltatás minőségét, a QoE-t. A 2.2. ábra a csomagfeldolgozó rendszer működését szemlélteti. Amint a csomagok beérkezési intenzitása meghaladja a feldolgozó kapacitás határát, átmenetileg a pufferek segítenek áthidalni a szűk keresztmetszetet a feldolgozás késleltetésével. A pufferek betelése után érkező csomagok viszont elvesznek. Ily módon a csomagfeldolgozás egy forgalomalakítóhoz (traffic shaper) hasonlóan működik, amelyben a limitáló tényező az adott rendszerre jellemző folyamatos csomagfeldolgozó kapacitás. Az ezt meghaladó beérkezési intenzitás mellett a forgalom időben késleltetést szenved, a pufferkapacitást meghaladva a működés csomagvesztésbe torkollik [**J2**].

Bár általános célú operációs rendszerekben a csomagfeldolgozásban egy hálózati csatolóhoz több processzormag nem rendelhető, csomagfeldolgozás közben a további processzormagok egyéb feladatokat viszont futtathatnak. A hatékonyság növelése érdekében ezek a processzormagok bevonhatók bizonyos csomagfeldolgozási részműveletek elvégzésébe is.

2.4 Motiváció

A mobil eszközök feldolgozási kapacitása erősen limitált. A munkaállomásokénál jóval kisebb teljesítményű központi feldolgozó egységnek (Central Processing Unit, CPU, processzor) számos folyamatot kell futtatnia. Ugyanakkor a több processzormagos technológia már ezekben a készülékekben is elérhetővé vált. Az időbélyegzés folyamatát érdemes a több processzormagos környezet adta lehetőségekkel kihasználni oly módon, hogy az időbélyegzés fázisai közül azt, amely nem szerves része a helyi óraforrás kiolvasásának, áthelyezhetjük a végrehajtás egy későbbi, jól párhuzamosítható szakaszába.

Az általam bemutatott módszer a helyi óraforrásból származó érték abszolút idejű ábrázolásra történő konverzióját áthelyezi egy utófeldolgozási szálba, amelyet több processzormagos rendszerben alacsony hozzáférési idejű (pl. regiszter alapú) helyi óraforrással kombinálva hatékonyan csökkenthető a megszakítási kontextusban töltött idő [**C4**]. Emellett a konverziós művelet

leginkább tétlen processzormagokon történő végrehajtásával növekszik a csomagfeldolgozó kapacitás és csökken a csomagvesztés lehetősége.

2.5 Szoftveres időbélyegzés tehermentesítéssel

Fontos tényező, hogy az időbélyeg tényleges felhasználása a csomagban hordozott hasznos teher feldolgozásánál később történik meg. Ez idő alatt a már lekérdezett óraforrásból származó érték konverziója a felhasználásig késleltetve is elvégezhető. A késleltetett vagy nem kritikusan leterhelt környezetben történő végrehajtást a szakirodalom tehermentesítő (offloading) technikának hívja. A csomagfeldolgozás kritikus szakaszában, a megszakítási kontextusban csak az órajelforrás lekérdezése és a kapott érték eltárolása történik meg. Ebben a kontextusban dolgozva a precizitás (P_t) és pontosság (A_t) jobban kézben tartható. A konverzió tehermentesített esetben felhasználói végrehajtási módban zajlik, amely a csomagfeldolgozási teljesítmény (P') javulását eredményezi az időbélyegző műveletek használata mellett. Több csomag esetében a ciklus-abszolút idő párhuzamos konverziója nem kivitelezhető, mivel a csomagok időbélyegeit szekvenciálisan kell átalakítani, a megelőző ciklusszámláló értékek alapján.

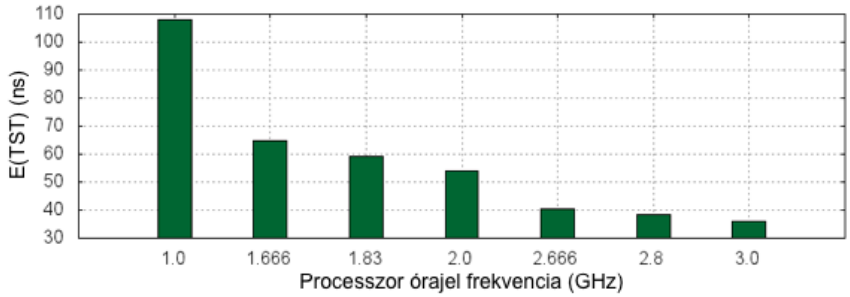
2.6 I.1. Tézis

Új szoftveres időbélyegzési módszert dolgoztam ki, amely az óraforrást valós idejű késleltetéssel (megszakítási kontextusban) olvassa ki, míg a konverziós lépéseket áthelyezi késleltetéstűrő felhasználói végrehajtási módba [**J1**].

Igazolás: A módszer működésének alátámasztása laboratóriumi, kontrollált körülmények között végzett mérésekkel valósult meg. A méréshez olyan prototípus rendszert készítettem, amely több processzormagos Linux környezetben és programozható logikai kapuk lapkáját (Field Programmable Gate Array, FPGA) tartalmazó kártyákon működik. A mérőrendszerben hardverrel támogatott csomaggenerátor segítségével állandó küldési időközzel azonos méretű csomagok sorozatát generáltam 1 Gbit/s közvetlen kapcsolaton. A fogadó oldalon a csomagokat időbélyegezzük. A mérés célja a hagyományos és a tehermentesített időbélyegzés teljesítményének összevetése volt.

Az x64 processzormagban található a Timestamp Counter (TSC) regiszter. Ez a regiszter egyben az általános x64 architektúrák leghatékonyabb óraforrása is [**25**]. Minden processzormagnak külön számlálója van, amelyek egymástól függetlenül órajelenként eggyel növekednek. Az állandó TSC (invariant TSC) tulajdonsággal rendelkező processzorokban a számláló egy másodperc alatt mindig a processzor névleges frekvenciájával nő, így az energiatakarékos üzemmódok használata mellett is alkalmas a megbízható óraforrás szerepének betöltésére [**26**]. A számláló egy 2 GHz órajel frekvenciájú processzoron elméletben 0,5 ns-os felbontású időinformációt szolgáltat (R_t=0,5 ns). Ez a felbontás korszerű processzorokban a frekvenciával arányosan növekszik, és arányosan kisebb időbeli költséggel olvasható ki. Ezzel az órajelforrással a ≥1 GHz frekvencián üzemelő processzormagokon elviekben a nanoszekundum felbontású P_t elérhetővé válik. A TSC óraforrás rövid elérési idejű és elegendően stabil az időbélyegzéshez való felhasználáshoz.

Az időforrás kiválasztása az általános célú architektúrákon elérhető források vizsgálata alapján történt. Egy módosított Linux kernel segítségével a csomagfeldolgozó rutinokban az időbélyegzés belépő és kilépő pontjain elhelyezett TSC regiszter-lekérdezés segítségével meghatároztam az időbélyeg előállításához szükséges órajel ciklusok számát [**J1**]. Az első cél a mérés saját költségének meghatározása volt, mivel a C_t vizsgálatakor kapott értékeket a mérés költségével kompenzálni kell majd. Ezért speciális mérési eset volt egy „pszeudo időbélyegzési" mód alkalmazása, amikor csak az időbélyegzési ág végrehajtása történt meg, az óraforrás kiolvasása valójában nem. Ismételt mérések segítségével kimutathatóvá vált az ebben a mérési technikában alkalmazott mintavételi eljárás átlagos költsége és annak szórása.



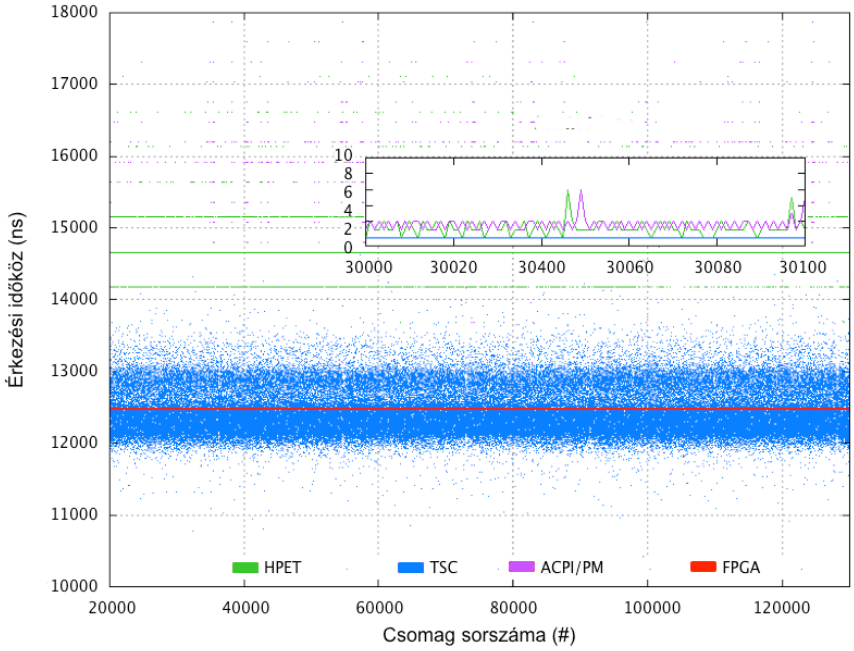
2.3. ábra Időbélyegzési költség a különböző órajelen működő processzorok esetében

Kimutattam, hogy a TSC-vel történő mérés a vizsgált rendszerekben[1] alacsony (±4 órajel) szórást mutat. A 2.3. ábra a várható TSC mérési költségeket szemlélteti, néhány különböző, eltérő órajelű processzoron. Mivel a mérések költsége állandónak tekinthetők, a magasabb órajelen működő rendszerekben maga a mérési mintavétel kevesebb abszolút időt vesz igénybe.

Az alacsony szórás oka, hogy a „Read TSC" (RDTSC) egy elemi utasítás, amelyet nem szakít meg az ütemező. Továbbá az RDTSC-vel történő órakiolvasás mellett a csomagfeldolgozás további elemi műveletei jó eséllyel maradhatnak bent belsőbb szintű processzor gyorsítótárban (CPU cache).

Az adott rendszeren kapott várható értéket (TSC kiolvasási költségeket) egyfajta „táraként" alkalmazva kimutathatóvá válik a különféle időforrások várható C_t értéke és szórása. A mérések során FPGA csomaggenerátor segítségével előállított 72 byte-os csomagok egyenközű sorozatát időbélyegeztem. A csomagokat 1472 byte-os keretközi réssel (inter-frame gap, IFG) küldtem ki. A 2.4. ábra az egyes időforrások használatakor mért, az FPGA hardver időbélyegtől való eltérést mutatja, az egyes forrásokkal kapott eredmények eltérő színnel vannak jelölve. Az ábrán vörös szín jelöli a referencia időbélyegeket. A legnagyobb szórást az Advanced Configuration and Power Interface/Power Management (ACPI/PM) időforrás (magenta színnel ábrázolva) használata eredményezi, míg lényegesen kisebb szórás mellett működik a High Precision Event Timer (HPET) forrás (zöld színnel jelölve). A legkisebb szórást, vagyis a hardver időbélyegekhez leginkább közeli értékeket a TSC forrás (kék színű pontokkal jelölve) használata eredményezte, így a többi időforrás használatát a későbbi mérések során elvetettem.

[1] A vizsgált architektúrákon 24 órajel várható értékkel történt a végrehajtás.



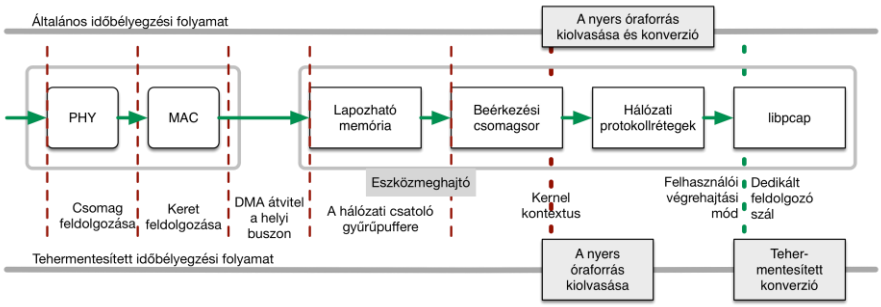
2.4. ábra Az időbélyegzés pontossága a referencia időbélyegekhez viszonyítva a különböző időforrások használatával

Lényeges részlet, hogy a processzormagok közötti TSC számlálók nem szinkronizálódnak ugyan, de ez a tényező az időbélyegzést végző csomagfeldolgozási szál (Software Interrupt Request, SoftIRQ) dedikált magon történő futtatásával áthidalható.

A Linux rendszerben a megszakításkezelést két részre bontották. A felsőszintű megszakításkezelő (*top half handler)* végzi a csomagok átvételét a hálózati csatolótól és időzíti az alsószintű megszakításkezelő (*bottom half handler*, SoftIRQ) hívását. A hatékonyságot alapvetően meghatározza, hogy az erőforrások minél rövidebb ideig legyenek lekötve, a végrehajtás pedig a lehető legrövidebb ideig legyen felfüggesztve. Az ellenőrzés, feldolgozás, időbélyegzés az alsószintű megszakításkezelő feladata. Egy hálózati csatoló fogadó pufferét (Receive Queue, RX) jellemzően egyetlen alsószintű megszakításkezelő tudja kiszolgálni, amely egy processzormagon futhat egy időben. Emiatt a gyors működés ezen a ponton is fontos szempont a csomagfeldolgozási teljesítmény tekintetében.

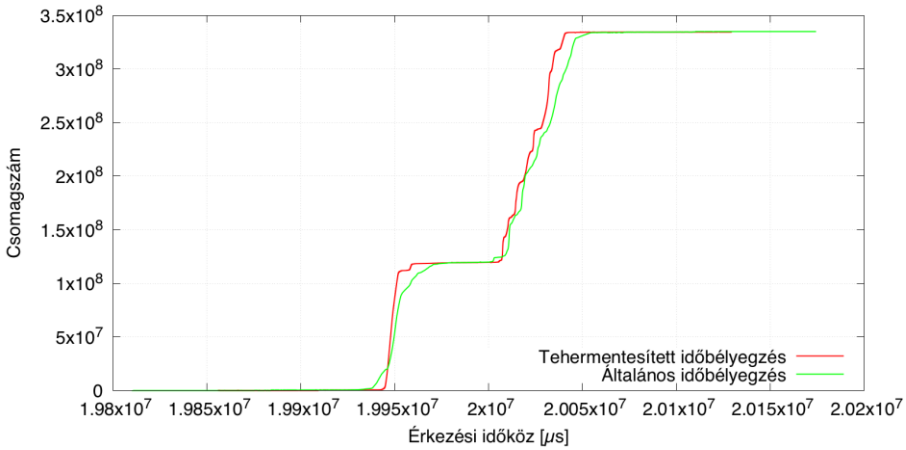
A tehermentesített időbélyegzési módszer az általános időbélyegzési módszerhez viszonyított különbségeit a 2.5. ábra szemlélteti. Míg az általános

időbélyegzés során minden időbélyegzési lépés a megszakításkezelőn belül történik, addig a tehermentesített módszerben csak az órajelforrás kiolvasása történik meg. A mérési környezetben használt Linux implementációban a konverzió a LibPcap-ben, külön szálként futó folyamatként zajlik.



2.5. ábra Az új időbélyegzési módszer a nyers idő információt alacsony költséggel rendszermag futtatási módban végrehajtva állítja elő. A konverziót felhasználói módban futva, dedikált szálon, késleltetve végzi el.

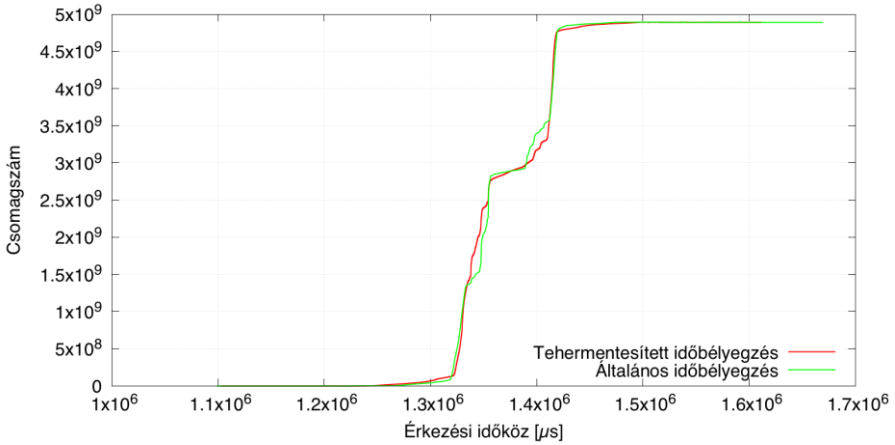
A módszer hatékonyságát VoIP forgalom emulálásával vizsgáltam. Ez a forgalom 140 byte-os csomagok 20 ms-onként kiküldött sorozatát jelenti, amely 2 449 860 byte-os keretközi résnek felel meg, azaz a csomagfeldolgozó rendszer nem szaturált körülmények mellett működött. A 2.6. ábra a csomagok érkezési időközeinek kumulatív eloszlását ábrázolja a két különböző módszer esetében. Mindkét módszernél TSC óraforrást alkalmaztam. Az általános időbélyegzési módszer mérési eredményei zöld, míg a tehermentesített módszeré vörös színnel lettek jelölve.



2.6. ábra A csomagérkezési időközök hisztogramja a VoIP mérésekben

Az érkezési időközök nem mutatnak jelentős eltérést, ami azt jelenti, hogy az új módszer használata nem torzítja az előállított időbélyegek értékeit. Az érkezési időközök természetesen bizonyos mértékű szórást mutatnak, amit az általános célú architektúra ütemezője okoz. Az emulált forgalom esetén ekkora ingadozást könnyen áthidal az alkalmazás saját puffere.

Egy további méréssorozatban HD videófolyamot emuláló forgalmi mintázattal végeztem el a méréseket, amelyek érkezési időközeinek kumulatív eloszlását a 2.7. ábra szemlélteti, az előző ábrával megegyező jelölésekkel. A csomagok 1368 byte méretűek voltak, 20 ms küldési időközökkel, amely 169 631 byte-os keretközi résnek felel meg.



2.7. ábra A csomagérkezési időközök hisztogramja a HD-videó mérésekben

Bár ezekben a mérésekben nagyobb intenzitás mellett nagyobb méretű csomagok érkeztek, a két módszer által előállított időbélyegek itt sem mutattak szignifikáns, az alkalmazási rétegben gondot okozó eltérést.

2.7 I.2. Tézis

Kimutattam, hogy az időbélyegzés konverziós fázisának késleltetéstűrő végrehajtási módba történő áthelyezésével jelentős csomagfeldolgozási teljesítménynövekedés érhető el, amely a csomagfeldolgozást érintő erőforrások szaturációjakor a csomagvesztés arányát csökkenti [**J2**].

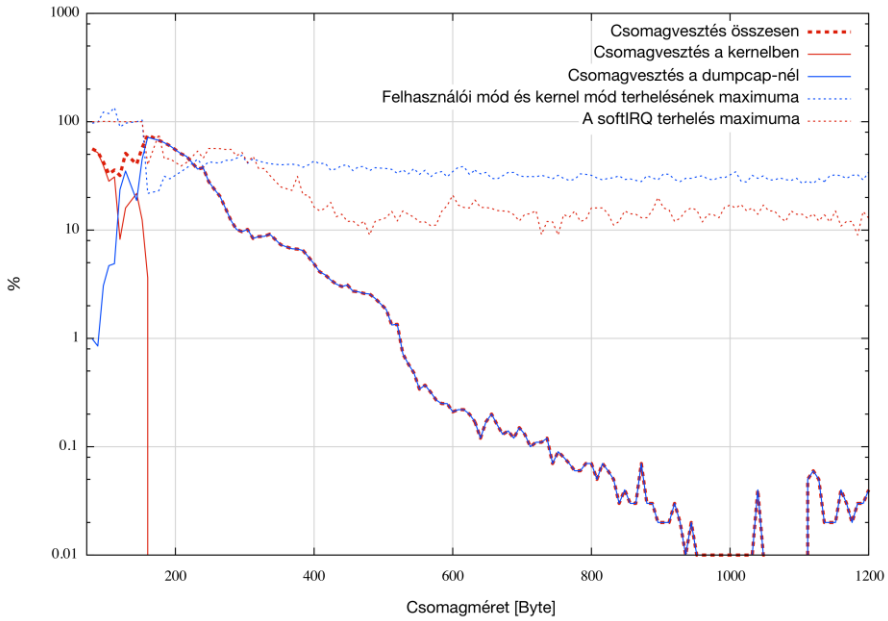
Amennyiben P' a hagyományos időbélyegzés, P'' pedig az új módszer melletti csomagfeldolgozási hatékonyságot jelöli, úgy

$$P'' > P' \tag{8}$$

Általános időbélyegzés esetében a csomagfeldolgozásban érintett processzormag p_T'×(C_p+C_r+C_c) időt tölt a csomagfeldolgozással (C'_{max}). A tehermentesitett módszer esetében ez a költég p_T''×(C_p+C_r), ezt jelölje (C''_{max}).

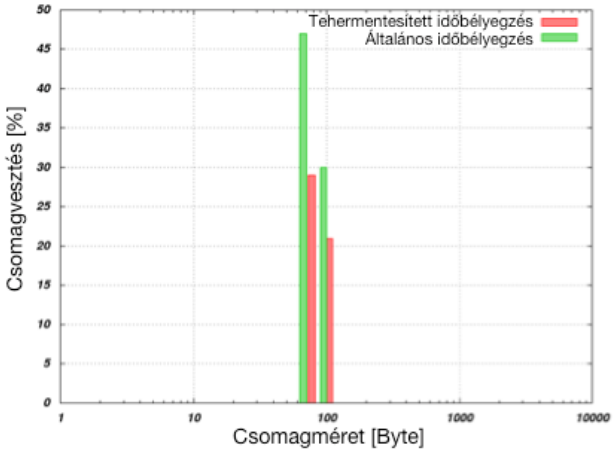
$$\begin{aligned} C'_{max} &= C''_{max} \\ p'_T \times (C_p + C_r + C_c) &= p''_T \times (C_p + C_r) \\ p''_T &= p'_T \times (1 + \frac{C_C}{(C_p + C_r)}) \end{aligned} \tag{9}$$

Igazolás: A hatékonyságvizsgálatot a csomagfeldolgozás szűk keresztmetszeteinek feltárásával kezdtem a hatékonyságvizsgálatot. Olyan méréssorozatot terveztem, amelyben a csomagvesztés arányát a csomagméret függvényében láthatjuk. Az egyenletes időközzel kiküldött csomagokat FPGA csomaggenerátor segítségével állítottam elő. A csomaggenerátoron egy Gigabit Ethernet interfész található és a rajta futó kód adott méretű és küldési időközű csomagok sorozatos továbbítását szolgálja. A generátor célhardverként működik, így mindig egyenlő időközökkel, a kívánt rátával küldi ki a csomagokat, akár a legkisebb méretben, vonali rátán is. Minden Ethernet keretben elhelyez egy sorozatszámot is. Minden keret kiküldése után a számláló értéke eggyel növekszik. Ezáltal a vételi oldalon eldönthető, hogy történt-e csomagvesztés illetve átrendeződés. Közvetlen összeköttetés esetén átrendezéssel nem kell számolni. Az eszköz vezérlése a felé küldött speciális Ethernet keret segítségével történik, amelyben megadható az összeállítandó csomagok mérete és a küldési időköz. Csomagelkapás közben a szoftver megszakításkezelő (softIRQ), valamint az egyéb kernel és felhasználói módú folyamatok jelentette terhelést is monitoroztam. Ahogy azt a 2.8. ábra is mutatja, a mérési környezetben 160 byte-nál kisebb csomagok nagy intenzitású beérkezése esetén csomagvesztés lép föl [**J1**][**J2**][**C2**][**C3**][**C5**].



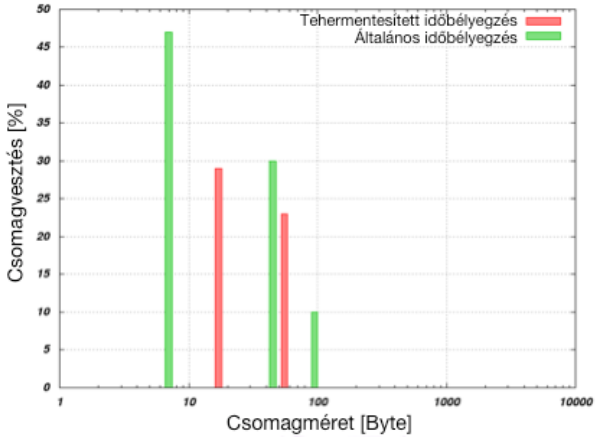
2.8. ábra Csomagméret és csomagvesztés aránya, valamint a mérési rendszer terheltsége

Az elveszített csomagok aránya a mérettel fordított arányosságot mutat. A terhelési görbék jól mutatják, hogy a csomagvesztést az okozza, hogy az adott csomagméret és küldési intenzitás mellett a rendszer elérte a számítási kapacitásának maximumát, így a csomagfeldolgozás során szűk keresztmetszet lépett fel. A csomagfeldolgozás számításigénye független a csomagmérettől, ezért a kisebb méretű csomagokkal kitöltött linkkapacitás esetén jóval több csomagfeldolgozási műveletet kell a rendszernek egységnyi idő alatt végrehajtania.



2.9. ábra A csomagvesztés aránya csomagméret függvényében

Az új időbélyegzési módszer csomagvesztésre gyakorolt hatását újabb méréssorozat segítségével vizsgáltam. A mérések során a célrendszernek 20 000 db 64 byte méretű csomagokból álló, FPGA alapú csomaggenerátorral előállított, nagy intenzitású forgalmat kellett elkapnia és időbélyegeznie. A 2.9. ábra az adott csomagméret mellett mért csomagvesztési arányt mutatja. A két vizsgált csomagméret 96 és 100 byte volt. A keretközi szünet függvényében tapasztalt csomagvesztést a 2.10. ábra mutatja. Az ábrákon zöld szín jelöli a hagyományos időbélyegzési módszer mellett mért adatokat, vörös szín pedig az új módszer segítségével mért értékeket.



2.10. ábra Csomagvesztés a keretközi szünet függvényében

A mérések kimutatták, hogy azonos érkezési intenzitás és csomagméretek esetén az új módszerrel csökkent a csomagvesztés aránya[2]. Ennek oka, hogy a csomagfeldolgozást végző processzormag időbélyegzési feladata az időforrás kiolvasására korlátozódott, ezáltal több számítási kapacitás maradt a csomagfeldolgozásra. Az időbélyegek további feldolgozása ezalatt egy másik processzormagon még a felhasználásuk előtt, késleltetéstűrő feltételek mellett történt.

2.8 I.3. Tézis

Az időbélyegzés óraforrás kiolvasó elemi műveletének megszakítási kontextusban tartása és a konverzió célirányos késleltetése az időbélyegzési precizitást (P_t) növeli.

Igazolás:

A vizsgálathoz az I.1. tézis alátámasztásánál alkalmazott VoIP jellegű forgalom méréssorozatát használtam fel. A 2.1. táblázatban a Linux általános időbélyegzési módszerének és a szintén Linuxban implementált tehermentesített módszer időforrás hozzáférési költségeinek átlaga és szórása látható. Mindkét módszer TSC időforrást használt.

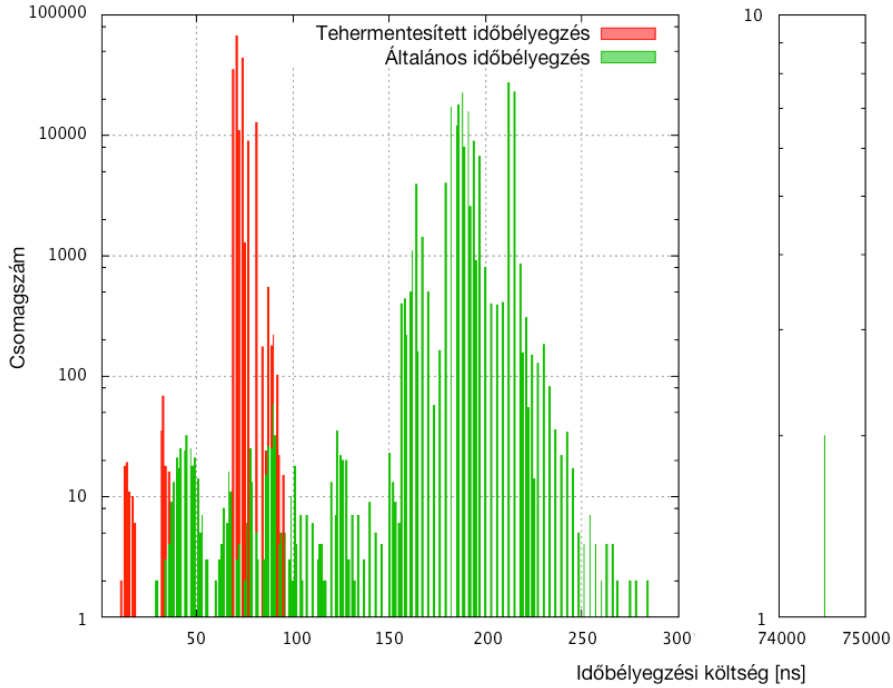
2.1. táblázat Az időbélyegzés hatékonysága VoIP-jellegű forgalom esetén

Módszer	P_r [ns]	σ_r [ns]
Általános időbélyegzés	195	176
Tehermentesített időbélyegzés	73	4

Látható, hogy az új módszerrel lerövidül a csomagonkénti feldolgozási idő (P_r), továbbá annak szórása (σ_r). A 2.11. ábra az időbélyegzési módszerek rendszermag futtatási módban felhasznált költségét ábrázolja. Látható, hogy az

[2] Egy olyan rendszerben, ahol nem szaturált a csomagfeldolgozó alrendszer, a csomagvesztésért nem a csomagfeldolgozási kapacitás szűkössége a felelős. Ilyen esetben csak akkor számíthatunk a csomagvesztés csökkenésére, ha a hatékonyabb időbélyegzés által felszabaduló erőforrás átcsoportosítható azon pufferhez kapcsolódó feldolgozó folyamat számára, amelyben a csomagvesztés bekövetkezett.

új módszerrel lényegesen kisebb az időbélyegzésre fordított költség a csomagfeldolgozás szempontjából kritikus kernel kontextusban.



2.11. ábra Az időbélyegzési módszerek rendszermagbeli időköltsége a VoIP jellegű forgalom esetében

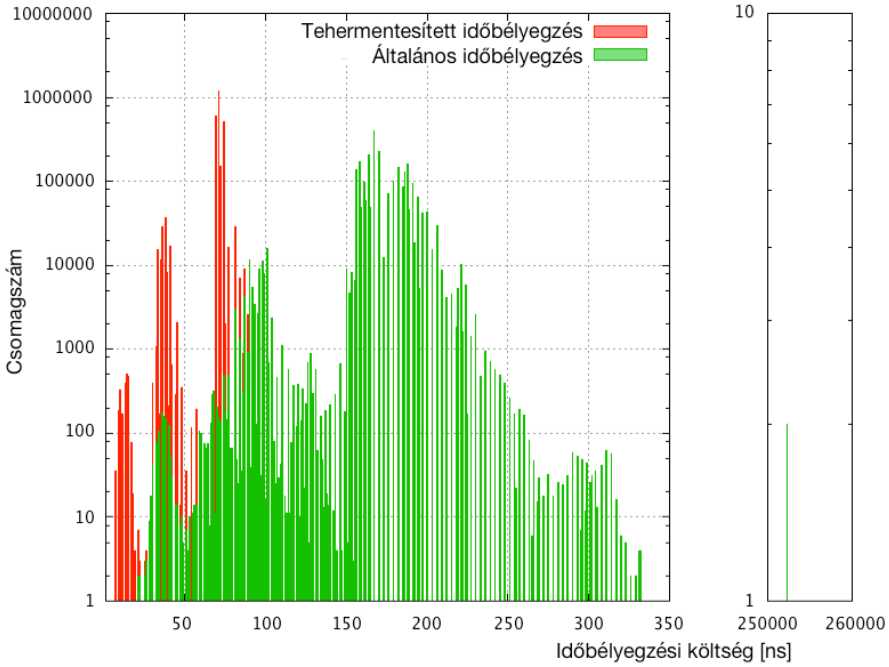
Az ábrán bizonyos esetekben kiugróan alacsony (≤50 ns-os) költség is látható. Ez azzal magyarázható, hogy a konverziós lépések áthelyezésének eredményeként az új módszerrel lerövidült az egy csomag feldolgozásakor lefuttatandó utasítások sorozata.

A méréssorozatot a HD videófolyam jellegű forgalom esetében is megismételtem. Ebben a méréssorozatban nagyobb csomagméretű és érkezési intenzitású forgalom időbélyegzése történt. A 2.2. táblázatban szintén látható az új módszer alacsonyabb költsége és kisebb szórása.

2.2. táblázat Az időbélyegzés hatékonysága HD videó jellegű forgalom esetén

Módszer	P_r [ns]	σ_r [ns]
Általános időbélyegzés	173	177
Tehermentesített időbélyegzés	71	8

A költséghisztogramot a 2.12. ábra szemlélteti. Az eloszlás a VoIP jellegű forgaloméhoz hasonló, vagyis jóval alacsonyabb a kritikus kontextusba eső időbélyegzési költség és annak szórása.



2.12. ábra Az időbélyegzési módszerek rendszermagra rótt időköltsége a HD-videó jellegű forgalom esetében

Mivel az új módszer alkalmazásakor a csomagfeldolgozó utasítássor mérete lerövidült, így megnőtt az esélye, hogy az utasítások a processzormag alacsonyabb szintű gyorsítótárában maradjanak. Ez a hatás nagy csomagérkezési intenzitás esetén valószínűbb, hiszen ilyenkor az adott processzormag szinte kizárólag csomagfeldolgozással foglalkozik. Ilyenkor a tehermentesített módszer előnyei fokozottan jelentkeznek.

A kisebb költségek további vonzata, hogy az egymást követő csomagok feldolgozása időben kevésbé tolódik el, vagyis a beérkezéshez képest hamarabb elkezdődhet az újabb csomagok feldolgozása. Ennek következtében az időbélyegek eltolása (offset), vagyis A_t értéke is kisebb lesz, azaz a valós érkezési időközökhöz közelebb eső (pontosabb) időbélyegeket kaphatunk.

2.8.1 Többrétegű időbélyegzés

A tehermentesített időbélyegzési technika alkalmazásával elérhetővé váló erőforrások lehetővé teszik, hogy a csomagfeldolgozási útvonalon több ponton is helyezhessünk el időbélyeget, amely az operációs rendszer és az alkalmazások fejlesztőinek munkáját segítheti a protokollrétegek átfogóbb monitorozásával.

2.9 Konklúzió

Az I. téziscsoport kapcsán bemutatott tehermentesített időbélyegzési módszer segítségével az általános célú architektúrákon a szoftveres időbélyegzési pontosság növelhető. Továbbá a több processzormaggal rendelkező környezetben az időbélyegzési művelet konverziós lépéseinek a csomagfeldolgozás kritikus időszakából későbbi végrehajtási fázisba áthelyezése növeli az adott csomópont csomagfeldolgozási kapacitását. Ezáltal az erőforrásokban szűkösebb, jellemzően mobil végponti eszközökön hatékonyabb QoS mérés valósítható meg, segítve ezzel a hálózati infrastruktúra vagy az alkalmazások fejlesztőinek, üzemeltetőinek munkáját.

3 Rate Control Transport Protocol

3.1 Bevezetés

K+F projekt keretében 100 Gbit/s sávszélességű szolgáltatói (mag-)hálózatok aggregált forgalmának monitorozására képes eszköz megalkotásában vettem részt. Az eszköz egyik fő funkciója a forgalom hardverrel gyorsított veszteségmentes monitorozása. A megcsapolt forgalmat szűrve és folyamokra bontva, azokat 1 vagy 10 Gbit/s áteresztőképességű kapcsolatokon adatgyűjtő- és feldolgozó szerverek felé továbbítja. Mivel a szerverek a forgalom további, mélyebb szintű elemzésével, statisztika készítéssel is foglalkoznak, feldolgozási oldalon a hardveres gyorsítás nem választható. Az általános célú operációs rendszerrel működő gépek, illetve az azokon futó adatgyűjtő, feldolgozó alkalmazások felé az előszűrt byte-folyamokat veszteségmentesen kellett továbbítani.

A monitoring eszközt FPGA alapú hardver segítségével valósítottuk meg. Számos korláttal kellett számolni a tervezés és fejlesztés során. Ezek egyike volt, hogy minél több funkciót kellett az FPGA chipben elhelyezni, annál több logikai kaput kellett „feláldozni". Bár egyre nagyobb kapacitású és teljesítményű FPGA IC-k érhetők el, a rendelkezésre álló kapacitás mindig véges és az áruk az integrált logikai elemek számával hozzávetőleg exponenciálisan emelkedik. Ráadásul a komplexitás növekedésével egyre nehezebb a megtervezett áramkör implementálásakor az optimális órajelfrekvencia megtartása. Ezért fontos, hogy az alkalmazott transzport protokoll erőforrástakarékos és minél egyszerűbb legyen. Az említett projekt kezdetén felvetődött a Transmission Control Protocol (TCP) alkalmazása. Mivel hálózati interfészenként példányosítani kellett a transzport protokollt, annak sokszoros erőforrásigénye miatt alternatív protokoll alkalmazásának és kifejlesztésének igénye vetődött fel.

3.2 Kapcsolódó kutatások

A TCP protokoll az internet alapvető transzport protokollja, amely két IP csomópont között hatékony adatátvitelt képes megvalósítani. Algoritmuskészlete biztosítja, hogy a legmostohább hálózati feltételek (pl. változó késleltés, csomagvesztés, átrendeződés stb.) esetén is nagy

teljesítményű, veszteségmentes átvitelt valósítson meg. Az elvesztett byte-folyam részek nyugtázását és újraküldését pufferek segítik. A TCP pufferméretek hangolásával számos kutatás foglalkozott már. A protokoll komplexitása és pufferelési szükséglete viszont korlátozza az FPGA környezetben több példányban történő alkalmazását [**27**].

Az UDP protokoll egyszerűségénél fogva alkalmas ugyan kevés erőforrással rendelkező hardverben történő implementálásra, de nem tartalmaz olyan szabályzó mechanizmust, amely egy esetlegesen fellépő torlódás kezelésére alkalmassá teszi [**28**]. Az erre a protokollra építkező alkalmazások rendszerint önálló pufferelés segítségével oldják meg az adatok veszteség esetén történő újraküldését. Az UDP Lite egy UDP variáns, amely lehetővé teszi a nem ép csomagok továbbítását is, de a fő célja az olyan multimédiás alkalmazások kiszolgálása, ahol az esetleges csomagvesztés rendszerint nem jelent problémát, ráadásul az időkritikus működés miatt az újraküldés igénye sem vetődik fel [**29**].

Az első megbízható üzenetváltás alapú protokoll-kísérlet a Bell Labsnál fejlesztett Reliable UDP (rUDP) volt [**30**]. Ez egy olyan könnyűsúlyú protokoll, amelyben a TCP-féle újraküldési mechanizmushoz hasonló pufferelést alkalmaznak, ezáltal egy extra puffer alkalmazása válik szükségessé. Bár a Cisco és Microsoft is dolgoztak a protokollon, mégis IETF draft státuszban maradt.

UDT néven készült egy UDP alapokon nyugvó protokoll, kifejezetten nagy kiterjedésű hálózatok (Wide Area Networks, WAN) számára [**31**]. Ez a protokoll rendszeresen gyűjt információt a lassú és megbízhatatlan kapcsolatokról, de alkalmazásának a küldőoldali puffer szükségessége az említett protokollokhoz hasonlóan korlátot támaszt.

A Datagram Conguestion Control Protocol (DCCP) szintén TCP-szerű megbízhatósági tulajdonságokkal rendelkezik, amelyet az Explicit Notification Control (ECN) segítségével biztosít [**32**]. A sorrendkövető (in-order) csomagtovábbítást nem valósítja meg, de az adatcsomagok időbélyegzésével és a vételi oldalon történő rendezésével szükség esetén ez áthidalható. Alkalmazásának legnagyobb problémája a sorozatos csomagvesztés keltette „felépülési folyamat” (recovery process), amely szintén pufferelés szükségét vetíti előre.

Az átvitt adatok darabolásával a TCP-t és UDP-t hatásosan kombinálja a Stream Control Transmission Protocol (SCTP) [**33**]. A TCP bizonyos gyengeségeit annak egyes funkcióinak kikapcsolásával kerüli meg, de mivel ez a protokoll sem foglalkozik a rendszer szűk keresztmetszeteinek monitorozásával, az esetlegesen elvesző adatot újra kell küldenie, amely a pufferelés szükségét továbbra is fenntartja.

Az Oracle-nél kifejlesztett Reliable Datagram Sockets-t (RDS) elsősorban InfiniBand fölötti folyamatközi kommunikációhoz (Inter-process Communication, IPC) fejlesztették ki [**34**]. A fogadó csomagpuffer (socket buffer) telítettségének figyelésével, az annak telítődéséből fakadó vesztést el tudja kerülni. Mivel azonban vételi oldalon nem csak e puffer túlcsordulása okozhat csomagvesztést, ezért általános célú architektúrákon teljes veszteségmentességet önmagában ez a protokoll sem garantál.

Kifejezetten mérőrendszerek számára lehet előnyös a Scalable and Secure Transport Protocol (SSTP), amely rövid adatfolyamok megbízható átvitelét biztosítja, emellett titkosítást is támogat [**35**]. Bár a TCP-hez viszonyítva kevésbé összetett, a méltányosság (fairness) és a biztonságos réteg funkciók miatt továbbra is túlságosan bonyolult.

Rendszeresen előállított és mérési adatok továbbítására szolgál a Smart Grid Transport Protocol (SGTP), amely rövid ideig tartó, kis méretű csomagokból álló folyamokat feltételez [**36**]. A szóban forgó mérőrendszerben állandó, hosszú, interfészenként önálló folyamok a jellemzőek, így számunkra nem nyújtott megoldást.

Az Ethernet Flow Control mechanizmusát R. Takano és társai vizsgálták [**37**]. A módszerrel impulzusszélesség moduláció (Pulse-width modulation, PWM) szabályzással a forgalom csomósodását (burst) lehet csökkenteni. Az úgynevezett szüneteltető keretek (pause frame) segítségével a küldő utasítható, hogy azonnal függessze fel a csomagok kiküldését. Megvizsgáltam, hogy a technika alkalmas-e a csomagvesztés elkerülésére az adott környezetben. A szüneteltető keretekkel történő felfüggesztés egy adott időtartamra szól, amely 512 bitidőnyi egységekben határozható meg 0 és 65 535 között. Ez 1 Gbit/s linksebesség esetén 512 és 33 553 920 ns=33,5 ms közötti impulzusszélességű PWM szabályzást tesz lehetővé. Ezek a korlátok

csak durva visszacsatolást tesznek lehetővé, amely a feldolgozást hektikussá teszi és oszcillációt, vagyis a szélsőséges állapotok közötti periodikus ingadozást is okozhat. A monitoring környezetben ettől jóval finomabb szabályzásra van szükség. Takanóék módszere elsősorban a csomósodó forgalom kisimítását célozza meg. Ez ellentmond annak a célunknak, hogy mihelyt feldolgozási kapacitás áll rendelkezésre, az adatot minél nagyobb átviteli teljesítménnyel továbbítsuk a fogadó irányába. Ehhez visszacsatoló mechanizmusra is szükség lenne, de az említett módszer ilyet nem tartalmaz.

3.3 Motiváció

A mérőrendszerhez olyan transzport protokoll létrehozása volt a célom, amely minimális erőforrás felhasználás mellett képes a veszteségmentes továbbításra. A mérőeszköz több (akár 10 darab) fizikai hálózati csatolón keresztül továbbítja a mérési adatokat. Az architektúra adottságaiból fakadóan ezeket a fizikai kapcsolatokat számos egyéb funkció mellett független példányokként kell implementálni. Ezért az erőforrás-takarékosság kiemelkedő szempont volt a tervezéskor. Ez a protokoll akár általánosabb célokra is alkalmazható lehet, pl. beágyazott és egyéb alacsony erőforrásokkal bíró rendszerekben.

A Machine To Machine (M2M), valamint az intelligens épületgépészet olyan mérőrendszerek telepítését teszik szükségessé, amelyeknek számos pontról, akár nagy mennyiségű adat veszteségmentes begyűjtését kell megvalósítaniuk. Ezekre az alkalmazásokra jellemző, hogy a generált adatforgalom egyik irányban, a szenzortól (probe) az adatfeldolgozó irányban domináns. Továbbá, mivel nagyszámú mérőeszközt kell telepíteni, fontos szempont azok energiahatékonysága is, ami elsősorban alacsony összetettségű hardver alkalmazásával valósítható meg.

A lehetséges torlódás és csomagvesztés egyaránt bekövetkezhet a továbbítást végző infrastruktúrán és a végponton is. Vezeték nélküli kapcsolatoknál a továbbító közeg kiszámíthatatlan: zaj, interferencia bármikor felléphet, az emiatt előforduló csomagvesztés a fizikai rétegben bekövetkezett átviteli hiba miatt történik. Vezetékes hálózatokban fellépő vesztés a hálózati forgalom torlódásából, valamely eszköz pufferének telítődésétől következik be. Dedikált célra létesített hálózatok (pl. adatközpontokban, mérőrendszerekben,

blokkolásmentes adatátviteli kapcsolókban (non-blocking switch) megfelelő tervezés mellett ezektől mentesek lehetnek. Ilyen környezetben csak a végponton következhet be csomagvesztés.

Egy általános célú operációs rendszerekkel rendelkező architektúrákban szintén puffereken keresztül vezet a csomagok útja a fogadó alkalmazásig. Az első puffer a hálózati csatoló saját puffere, amelybe a fizikai rétegen (Physical Layer, PHY) keresztül érkezik a csomag. Ez a puffer rendszerint minimális méretű, de jól tervezett meghajtóprogrammal alkalmas arra, hogy a következő csomag érkezéséig áthidaljon, amíg az aktuális csomag bemásolódik a rendszermemóriába (Random Access Memory, RAM). A további pufferek már az operációs rendszer és a csomagfeldolgozó alrendszer részeit képezik. A legtöbb esetben ezen pufferek méretei hangolhatók, ami biztosítja, hogy impulzusszerűen érkező forgalom esetén is elég kapacitást lehessen allokálni az átmeneti tároláshoz. A beérkező csomagokat az operációs rendszernek fel kell dolgoznia: a csomag az őt érintő protokollok fejléceinek ellenőrzésén megy át, amely egyfelől segít kiszűrni a hibás vagy a rendszer és alkalmazásai számára érdektelen csomagokat, másfelől a megfelelő alrendszer felé irányítható tovább az információ. Ezek a feladatok a csomagok beérkezési intenzitásának növekedésével egyre nagyobb terhet rónak a rendszerre. Amennyiben a beérkező csomagok feldolgozására nem áll rendelkezésre elegendő számítási kapacitás, a pufferek telítődése után érkező csomagok elvesznek. A végpontokon a pufferkapacitások mellett tehát fontos mutató a csomagfeldolgozó alrendszer teljesítménye is.

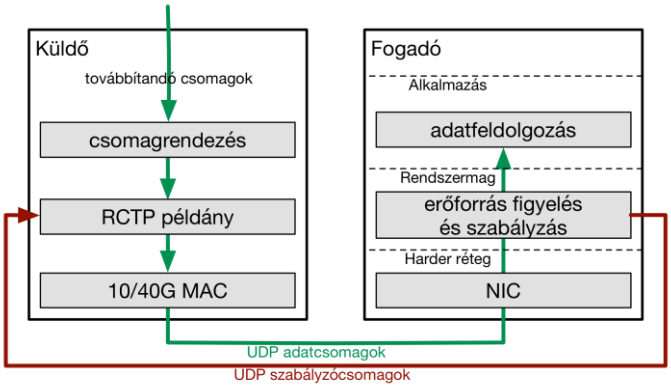
3.4 II.1. Tézis

Kidolgoztam az RCTP torlódás megelőző transzport protokollt, amely a TCP-hez viszonyítva alacsonyabb erőforrásigény mellett képes – akár vonali rátán is – elkerülni a csomagvesztést [**J4**].

3.5 A protokoll algoritmikus működése

Az RCTP protokoll egy zárt hurkú szabályzóra épül, amelynek feladata a küldési intenzitás hangolása a fogadóoldal teljesítménymutatói alapján (3.1. ábra). A szabályzófüggvény bemenő paraméterei időről időre történő mintavételezéssel

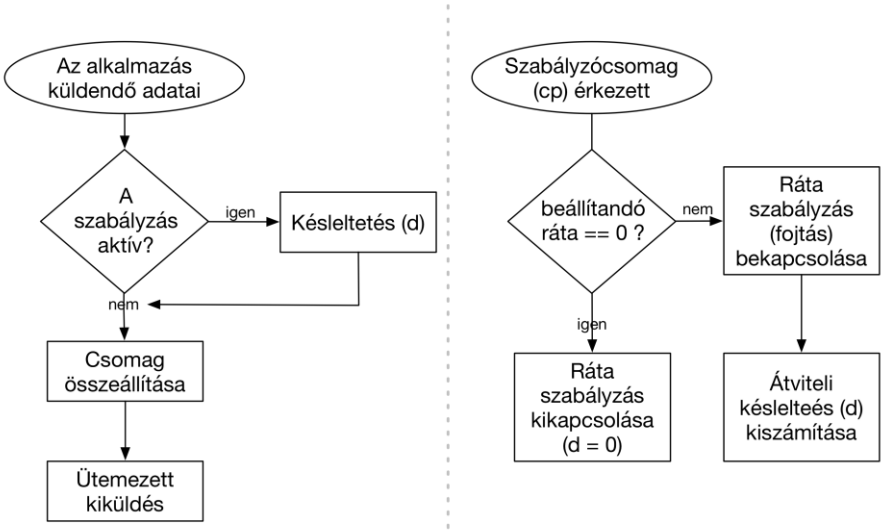
állnak elő: jelen esetben az érintett pufferek telítettsége és a csomagfeldolgozást végző processzormag kihasználtsága. A protokoll ezek alapján egy, az adott paraméterekre jellemző maximális küldési rátát számol. Előnye az alacsony összetettség, hiszen küldő oldali pufferre nincs szükség (így akár közvetlen hardveres implementáció is lehetséges), valamint a gyors visszacsatolás. Ugyanakkor elkerüli a folyamatos nyugtázást, amit eseményvezérléssel helyettesít.



3.1. ábra Az RCTP protokoll szabályzási mechanizmusa

Az esetleges csomagvesztés elkerülhető a kritikus erőforrások folyamatos monitorozásával, valamint a csomagfolyam útjába eső pufferek megfelelő hangolásával, a mintavételezési periódusok és a visszacsatolási késleltetés áthidalásával. A mintavételezés állandó periódusidővel történik. Mintavételkor az algoritmus a pufferek telítettségét és a processzormagok kihasználtságát kérdezi le. Ezek a paraméterek a vezérlőfüggvény bemenetét fogják képezni.

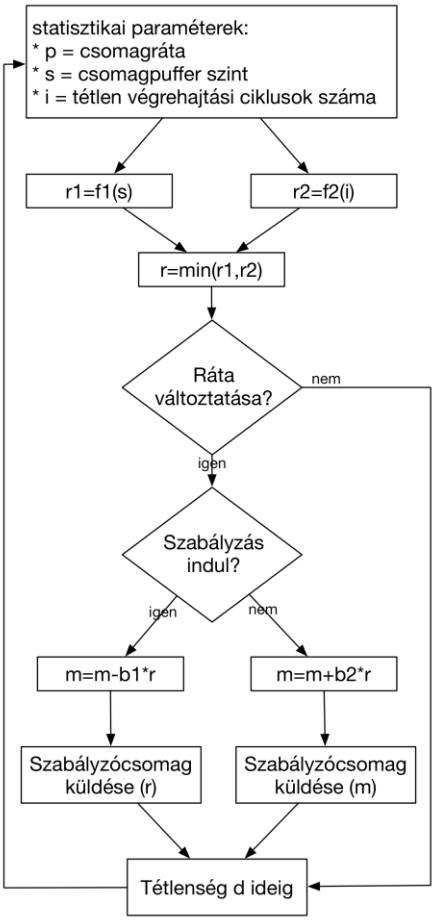
A protokoll küldő oldala kezdő lépcsőfokként a küldendő byte-folyam csomagokba szervezését végzi. A folyamot a hatékonyság érdekében maximális átviteli egység (Maxmimum Transmission Unit, MTU) méretű csomagokba célszerű szervezni. Ha az alacsony késleltetés is szempont, egy várakozási küszöbérték segítségével biztosítható, hogy minden csomag véges határidővel feldolgozásra kerüljön. A 3.2. ábra a csomagtovábbító ciklust ábrázolja, amelyben a csomagok kiküldése az aktuálisan érvényben lévő küldési ráta szerint történik. Ez a mindenkori ráta egy rátaszabályzó (rate control) csomag érkezésekor bírálódik felül.



3.2. ábra A küldő oldali csomagtovábbítás, amely az aktuális késleltesen alapul (balra), és a szabályzó csomag visszacsatolási mechanizmusa (jobbra)

A csomagok felépítésére és tárolására, illetve a byte-folyam kívánt küldési rátához szabására egy kisebb méretű egyszerű várakozási sor (First In First Out, FIFO) is elegendő.

A vételi oldal szerepe az erőforrás monitorozás és visszacsatolás. Ez egy időkritikus folyamat, ezért biztosítani kell, hogy a végrehajtás pontosan kerüljön a folyamatra. A 3.3. ábra a küldési ráta számítását és a visszacsatolás folyamatát ábrázolja.



3.3. ábra A szabályzás fogadó oldali mechanizmusa

A mintavételezés során a kiolvasott metrikák (pufferállapot és csomagfeldolgozó processzormag terheltsége) a vezérlőfüggvényük segítségével (f_1–gyel és f_2–vel jelölve) elállítja az aktuális körülmények között teljesíthető maximális csomagrátát. Az effektív kívánt csomagráta ezek minimuma lesz:

$$f_r = min(f_1, f_2) \tag{8}$$

ahol f_r a kívánt rátát, f_n (n=1,2,..) a metrikák mért értékeit jelöli.

Amennyiben a számított érték különbözik az előző ciklusban meghatározottól, azt egy szabályzó csomagban küldjük át. Ha a kiszámolt ráta a korlátozás feloldását követeli meg, akkor a vezérlőcsomagban a maximális csomagrátát küldjük ki. E rátát induláskor a linksebességből származtatjuk. A

korlátozás életbelépésekor egy b_1 konstanssal súlyozva csökkentjük a rátát. Feloldáskor b_2-vel súlyozva növeljük azt. A konstansok megfelelő megválasztásával stabilabb átvitel érhető el és megelőzhető az oszcilláció.

3.5.1 Az operációs rendszer paramétereinek hangolása az RCTP számára

Az algoritmus működését FGPA küldő oldali és Linux fogadó oldali implementációjával validáltam, 10 Gbit/s átviteli ráta mellett.

Az operációs rendszert fel kell készíteni a protokoll használatához, ebben a lépésben a linksebesség egy fontos paraméter. Linux környezetben a beérkező csomagok – hálózati meghajtóprogramtól függően – két vagy három pufferen keresztül haladnak. A hálózati csatolón található kis méretű puffert a meghajtóprogramon keresztül éri el a rendszermag. Ennek monitorozása a korszerű kártyákon nem szükséges, mert a jól implementált hálózati csatoló hardvere és meghajtópogramja képes maximális csomagráta esetén is a rendszermemóriába továbbítani a beérkező csomagokat. Régi, nem-NAPI jellegű meghajtók esetén a feldolgozásra váró csomagok pufferébe (backlog) kerülnek át a csomagok. A korszerű meghajtóprogramok ezt megkerülik és saját privát várakozási sorukban (private queue) gyűjtik a hálózati csatolótól átemelt csomagokat, amelyek innen azonnal a csomagfeldolgozó alrendszerhez kerülnek. A vizsgált környezetekben e várakozási sorok kezelése nem képezett szűk keresztmetszetet, így a monitorozásuk nem szükséges. A csomagok a feldolgozás során átkerülnek a fogadó alkalmazás csomagpufferébe. A csomagfeldolgozó rendszer működését a 2.3.2. fejezetben szemléltettem. A csomagpufferből már a felhasználói végrehajtási módban futó alkalmazás emeli ki a csomagokat. A puffer aszerint ürül, ahogy az alkalmazás kezelni képes a hozzá érkező adatokat. Ha a számítási kapacitás szűkös, akkor itt szűk keresztmetszet képződhet, ezért ezt a puffert mindenképpen monitorozni kell.

A modern Linux rendszerekben megszakításelosztó (IRQ balancer) gondoskodik az aktív folyamatoknak a rendelkezésre álló processzormagok közötti elosztásáról. Ez a folyamat statikus (indításkor meghatározott) vagy dinamikus módon próbálja egyenletesen elosztani a rendelkezésre álló processzormagok között a megszakításkezelési feladatokat. Használatának

hátránya, hogy nem jelezhető előre, hogy egy adott folyamat mely magon és milyen más folyamatokkal fog osztozni. Ha egy folyamat számára exklúzív hozzáférést szeretnénk biztosítani, a feladat az *smp_affinity*, *taskset*, *cgroup*, *cset* rendszereszközök segítségével oldható meg.

Az általános és UDP fogadó pufferek mérete a linksebesség függvényében határozható meg:

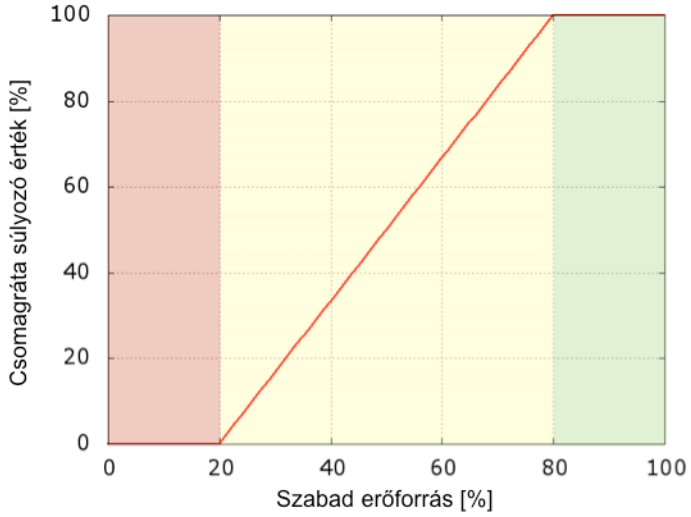
$$S = 2 \times B \times \frac{1}{F} \times D / 8 \tag{9}$$

ahol S a puffer szükséges mérete (byte), B a linksebesség (bit/s), F a mintavételezési frekvencia (másodperc) és D a késleltetési idő (másodperc).

A hálózati csatoló tehermentesítő technikáit ebben a konfigurációban csak korlátozottan aknázhatjuk ki. A vételi ellenőrzőösszeg-számítás (RX checksumming) az UDP csomagok ellenőrzőösszegei kiszámításának terhét veszi át a rendszermagtól. Az implementáció ezt nem használta ki, de az éles környezetben célszerű kiaknázni. Mivel MTU méretű csomagokba szerveztük a byte-folyamot, a hálózati csatoló UDP összeillesztést segítő (UDP Fragmentation Offload, UFO) funkciójának használata nem jelent további előnyt. Ahogy az általános fogadást tehermentesítő (Generic Receive Offload, GRO) és nagyméretű csomagok fogadását tehermentesítő (Large Receive Offload, LRO) NIC funkciók használata sem, mivel ezek az MTU méretűnél kisebb csomagokat segítenének kevesebb, de nagyobb méretűekké összefűzni, ezáltal csökkentve a megszakítások (és a kontextusváltások) gyakoriságát. Ezek a funkciók viszont torzítják a protokoll számára fontos statisztikát, amely a hatékony működését akadályozza.

3.5.2 Szabályzófüggvények

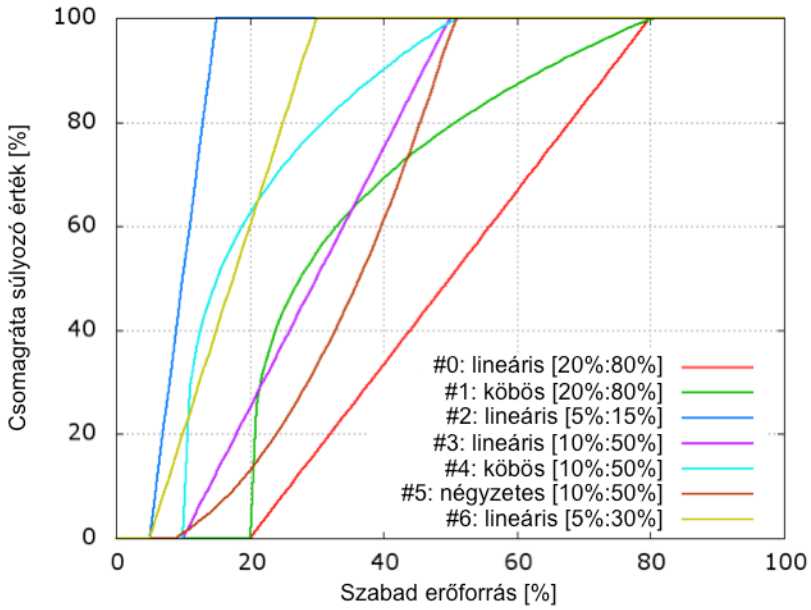
A protokoll működésekor a mért metrikák alapján történik az új, küldő számára javasolt küldési ráta számítása. A 3.4. ábra egy lineáris függvényt mutat be, amely a szabályzás tartományát három részre bontja. Az erőforrás felső korlátja fölött nem alkalmaz szabályzást (zöld terület), abben tartózkodva a küldő a maximális rátával küldheti az adatot. A metrika szerint mért érték felső korláta alá csökkenve a protokoll a felső és alsó korlát közötti lineáris mértékű ráta visszafogást javasol (fojtás, sárga terület). Az alsó korlát alá csökkenve a küldés teljesen felfüggesztésre kerül (vörös terület).



3.4. ábra Lineáris szabályzófüggvény 20%-os és 80%-os küszöbértékekkel

3.1. táblázat Nem lineáris vezérlőfüggvények a szabályzó tartományban

Szimbolikus név	t_0	t_{max}	F(x)
Lineáris [5%:30%]	5	30	$100 - (t_{max} - t_0) \times (x - t_0)$
Köbös [10%:50%]	10	50	$29 \times \sqrt[3]{x - t_0}$
Köbös [20%:80%]	20	80	$25{,}5 \times \sqrt[3]{x - t_0}$
Négyzetes [10%:50%]	10	50	$\sqrt{x/5} - 3$



3.5. ábra A különböző szabályzófüggvények fojtógörbéi

Több különböző, magasabb fokú függvényekkel dolgozó szabályzófüggvényt is megvizsgáltam azzal a céllal, hogy kiderítsem, megvalósítható-e velük a veszteségmentes átvitel, továbbá hogy megkeressem, mely alkalmazásával érhető el nagyobb átviteli teljesítmény. Ezek közül néhányat a 3.1. táblázat mutat be. A köbös jellegűek kevésbé agresszíven szabályoznak le, míg a négyzetesek drámaibban avatkoznak közbe, amint azt a 3.5. ábra is szemlélteti. A szabályzás hatékonysága a bemenetét képező metrika felbontásától is függ. A pufferméret jellemzően legalább 10^5 nagyságrendű felbontást jelent, míg a processzormag kihasználtság általában jóval kisebb értéktartományban mérhető. Kernel modulként működve további lényeges tényező, hogy nem alkalmazhatunk lebegőpontos számítást. Linux rendszerben a legkézenfekvőbb megoldás a tétlen processzormag ciklusok mérése. E metrika felbontása a következőképp számítható ki:

$$R = f \,/\, (1000 \,/\, F) \tag{10}$$

ahol F a kernel időperiódusát (jiffy) és f a mintavételezési (ms-ban kifejezve) jelöli.

A processzormag-tétlenség metrika felbontása rendszerint kisebb, mint a csomag puffer méretéé, ráadásul az utóbbi hangolható rendszerparaméter. Ezért a fojtást rendszerint az első metrika fogja aktiválni. A mérések során a szabályzófüggvények skáláját is e metrika tekintetében vizsgáltam behatóbban.

A szabályzófüggvények használata önmagában még nem garantálja a stabil átvitelt. Előfordulhat oszcilláció, vagyis a ráta bizonyos szintek közötti folyamatos ingadozása. Ennek másik véglete a „lusta" állapot, amikor egy stabilizált, de alacsonyabb átviteli teljesítményű rátán tartja a szabályzó az átvitelt. Ezért ún. „jutalom" és „büntető" konstansokkal (b_1, b_2) súlyozott rátával határozzuk meg a szabályzó új értékét. Ezek alkalmazásával egyszerre stabilizálható a működés és stabil állapotban tovább kísérlehető a ráta növelése.

3.6 II.2. Tézis

A szabályzófüggvények vizsgálatával kimutattam, hogy az 5 és 30% között lineáris karakterisztikájú függvény biztosítja — 0,2-es büntető, 0,1-es jutalmazó konstansokkal alkalmazva — a leghatékonyabb szabályzást az RCTP protokoll számára. Ezen túl rámutattam, hogy a rendszerparaméterek megfelelő megválasztásával a vonali ráta megközelítéséhez nem szükségszerű a mintavételezési periódusidő csökkentése, bár a maximális átviteli teljesítmény gyorsabban elérhető a rövidebb mintavételezési periódusidővel.

3.7 Hatékonyságvizsgálat az implementáció segítségével

A protokoll működőképességének és teljesítményének vizsgálatát laborkörnyezetben végzett mérésekkel igazoltam. Erre a célra 10 Gbit/s közvetlen kapcsolattal összekötött végpontokból építettem tesztkörnyezetet. A küldő oldalt FPGA-n implementáltuk, mivel előzetes tapasztalatok alapján általános operációs rendszerrel ez a linkkapacitás nem tölthető ki teljesen egyetlen UDP csomagfolyammal, még vállalati kategóriájú hardverrel sem. A küldő oldal egy egyszerű csomaggenerátort is magába foglalt, amely megadott rátával képes csomagsorozatot generálni.

A vételi oldal továbbra is általános célú architektúra, többmagos x64 processzorral és 82599ES 10GbE hálózati csatolóval. A vételi oldalt Linux környezetben implementáltam C nyelven. Az operációs rendszert és a

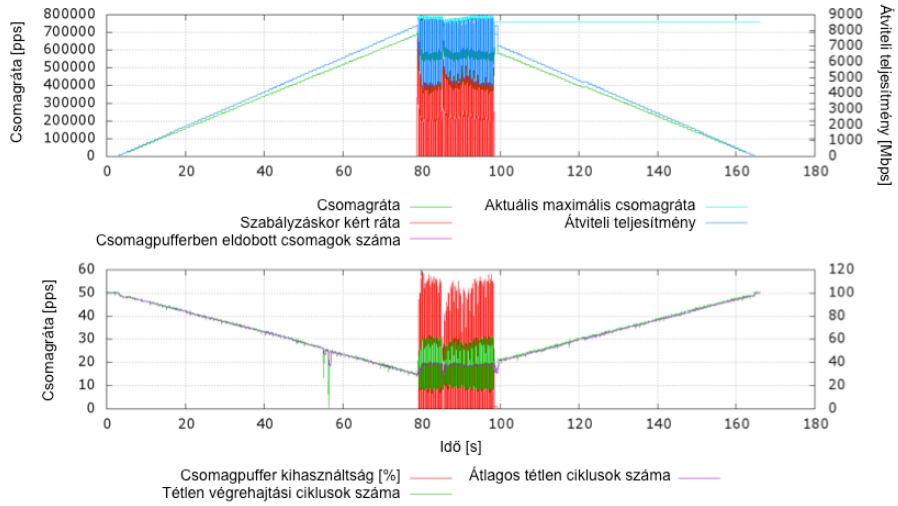
puffereket a protokoll elvárásainak megfelelően hangoltam. Azért, hogy egyéb folyamatok ne zavarják a protokoll működését, 1000 Hz-es, beelőzést tiltó (non-preemptive) kernelt alkalmaztam.

Az implementáció egy megfigyelő interfészt is tartalmaz, amely segítségével a szabályzás alapját képező metrikák folyamatosan követhetők, továbbá maga a fojtás folyamata és az átviteli teljesítmény is monitorozható.

A mérés során a küldő oldal csomaggenerátor modulja folyamatosan állította elő a csomagfolyamot, amelyet a vételi oldalnak veszteségmentesen kellett tudnia átvenni és szükség esetén a küldési rátát is megváltoztatni.

3.7.1 A szabályzófüggvények hatása

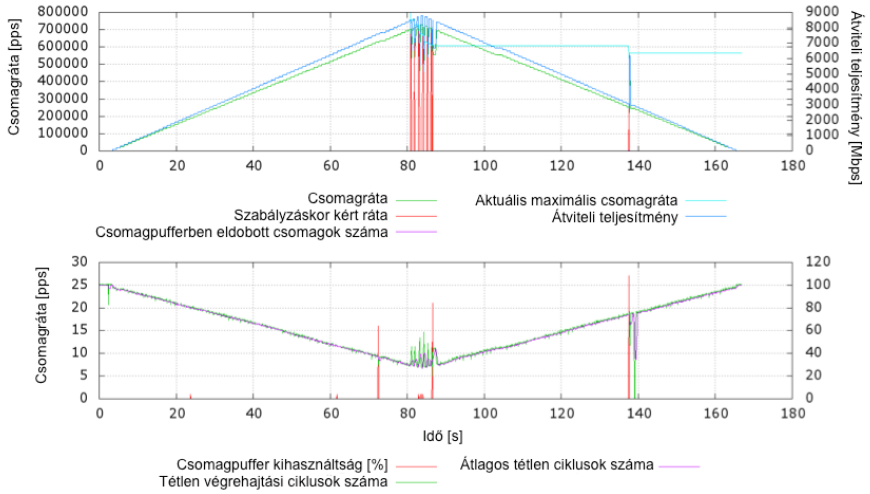
A megfigyelt metrikák bemenő paraméterként szolgálnak a vezérlés számára. A számítási igény csökkentése végett a szabályzófüggvények előre kalkulált indextáblákon alapulnak. Hatékonyság szempontjából több különböző, lineáris és nemlineáris karakterisztikájú szabályzófüggvényt is megvizsgáltam.



3.6. ábra Billegés effektus, amit a b_1=0,1 és b_2=0,1 értékek okoztak

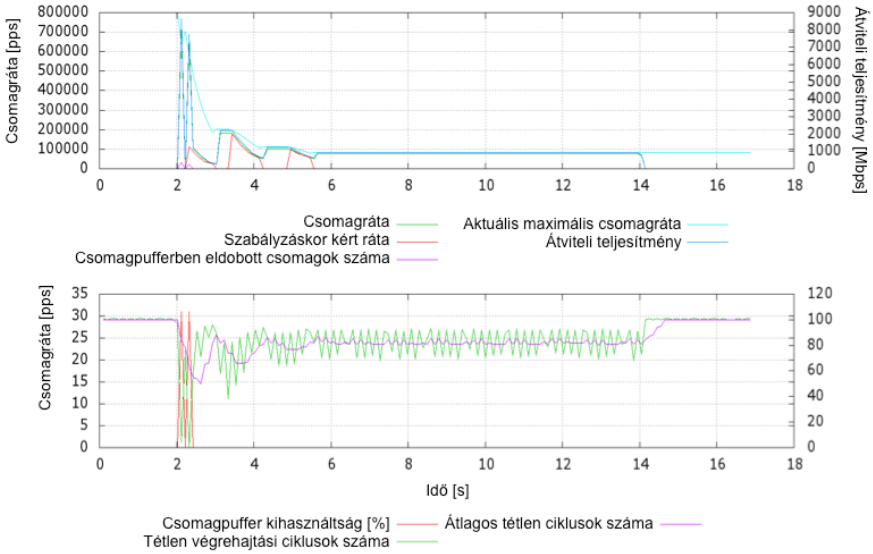
A *köbös [10%:50%]* függvény elég széles tartományban engedi a linksebességű működést, majd egy viszonylag enyhe, de növekvő mértékű leszabályzással reagál az erőforrás elfogyására (3.6. ábra). Itt a b_1 és b_2 konstansok azonos értékei esetén oszcilláció lépett fel. A konstansok értékét a 0 és 1 intervallumban, 0,1-es lépésközzel végzett iteratív mérések segítségével határoztam meg. A konstanspárok eltérő értékei biztosítják, hogy az erőforrás

fogyására illetve gyarapodására eltérően reagáljon a szabályzó (3.7. ábra). A *lineáris [5%:30%]* függvény számára a b_1=0,2 és b_2=0,1 konstansok biztosították a legstabilabb átvitelt a legjobb átviteli teljesítmény mellett.

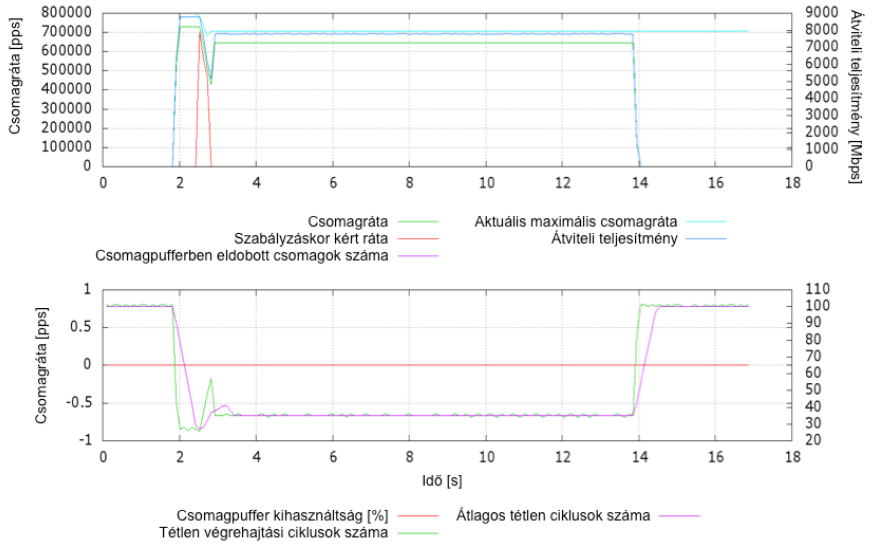


3.7. ábra Stabilizált átvitel, amit a b_1=0,2 és b_2=0,1 értékekkel értem el

A 3.8. ábra a *köbös [20%:80%]* függvény fojtógörbéjét és bemeneti metrikáit mutatja. Bár az átvitel karakterisztikája stabil, ez a függvény „túl óvatos", vagyis túl nagy súllyal törekszik a megelőzésre. Jóval megengedőbb a szabályzási tartományban a négyzetes jellegű függvény, de ez a tesztek során oszcillációt idézett elő, ami a stabil átviteli teljesítményt nem tette elérhetővé. A 3.9. ábra a *lineáris [5%:30%]* függvény átviteli és szabályzási karakterisztikáját szemlélteti. E függvény viszonylag tág tartományban engedi a teljes rátán működést és egészen kis erőforrásmennyiségig engedi a lineáris leszabályzást. A fogadó oldali pufferek elegendő ideig segítenek áthidalni az ideiglenes processzormag-túlterhelés okozta feldolgozási kapacitás hiányát.



3.8. ábra A szabályzás vezérlésének működése a köbös [20%:80%] függvény esetében



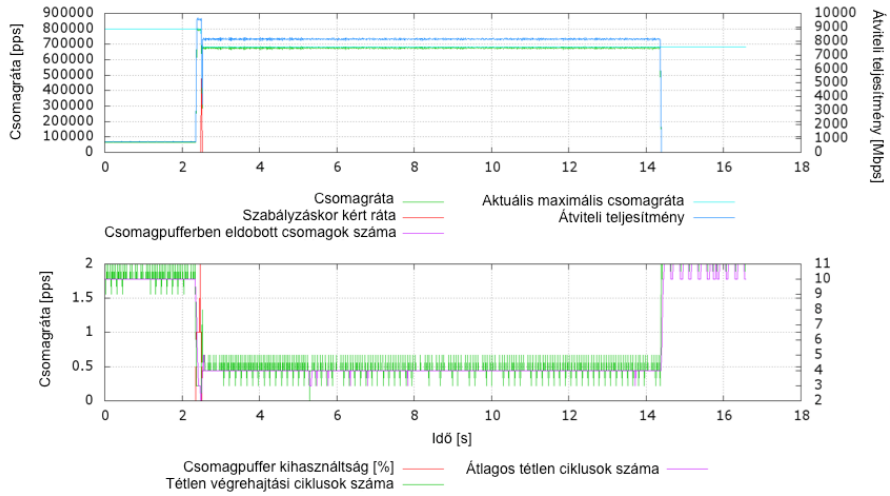
3.9. ábra A szabályzás vezérlésének működése a lineáris [5%:30%] függvény esetében

A *lineáris [5%:30%]* szabályzófüggvénnyel oszcilláció nélkül gyorsan elérhető a stabil átvitel, amely hasonló körülmények között a TCP teljesítményével is összemérhető.

3.7.2 Mintavételezési frekvencia

A korábban említettek szerint a különböző metrikák nem feltétlenül azonos értéktartományúak és felbontásúak. A processzormag-terhelés metrikája függ a mintavételezési frekvenciától: vagyis minél ritkábban mintavételezünk, annál több ciklus (Linux terminológiával jiffy) „fér be" két mintavétel közé. Ugyanakkor a ritkább mintavételezés lassabb szabályzási visszacsatolást is jelent, amely analóg a hosszabb (hálózati) útvonallal. Ezt a fogadó oldali pufferek méretével lehet kompenzálni. A mintavételezési frekvenciához kapcsolódó mérések során arra kerestem a választ, hogy jelent-e előnyt a jobb felbontású metrika alkalmazása, vagy célravezetőbb-e a kisebb felbontású metrikával történő sűrűbb mintavételezés a gyorsabb konvergencia lehetőségével.

A 3.9. ábra a 100 ms-os mintavételezés mellett készült mérések eredményeit mutatja be 8×10^6 elérni kívánt csomagráta mellett. A méréseket 10, 25, 50 és 75 ms mintavételezés mellett is elvégeztem. A 10 ms-os mintavételezéssel kapott átvitelnél jóval gyorsabb a konvergencia, hiszen a szabályzó jóval sűrűbben tudja „jutalmazni" a stabil átvitelt (3.10. ábra).



3.10. ábra Gyorsabb konvergencia a maximális vonali rátához a nagyobb mintavételezési frekvencia esetén

3.7.3 A hálózati csatoló tehermentesítő szolgáltatásai

Az UDP ellenőrző összegek implementálására abban az esetben van szükség, ha a protokoll fölött átvivendő adat konzisztenciája egyéb módon nem ellenőrizhető. Ilyen környezetben a hálózati csatoló RX tehermentesítő (RX offloading) funkciója segíthet a hatékony feldolgozásban, mert a vételi oldali rendszermagnak már nem kell az ellenőrzőösszegeket számolnia, ami nagy forgalmi intenzitásnál további előnyt jelenthet.

3.7.4 Az RCTP teljesítménye a TCP-hez viszonyítva

Bár a TCP kimagaslóan nagy átvitel elérésére képes megbízható csomagtovábbítás mellett, az általam bemutatott RCTP protokollal elérhető teljesítmény összemérhető vele. A 3.2. táblázat a két protokoll funkcionális összehasonlítását tartalmazza. Bár az RCTP nem rendelkezik bizonyos megoldásokkal, amelyek többek között a csatorna hibájából bekövetkező adatvesztés kiküszöbölésére irányulnak, az RCTP működési környezetében infrastrukturális csomagvesztéssel gyakorlatilag nem kell számolni. Így a TCP hibák esetén alkalmazott újraküldési funkciói kiválthatók egy kifinomultabb, csomagvesztést elkerülő, küldési intenzitást szabályzó mechanizmussal.

3.2. táblázat A TCP és az RCTP funkcionális összehasonlítása

Erőforrás	TCP	RCTP
Folyamat felépítés	van	nincs
Megbízható átvitel biztosítása	újraküldéssel, ütközéskezeléssel, folyamvezérléssel	küldési ráta szabályzása
Csomagvesztés elleni védelem	van	van *
Átrendeződés elleni védelem	van	nincs
Csomagsérülés elleni védelem	van	nincs
Méltányosság	van	nincs **
Kétirányú működés	van	nincs

* Az infrastruktúrán bekövetkező csomagvesztést nem kezeli
** A kapcsolatonként egy csomagfolyam továbbítása miatt nem szükséges

A 3.3. táblázatban a TCP és az általam készített RCTP implementáció által egy adott FPGA platformon, a célrendszerhez hasonló környezetben felhasznált erőforrásai láthatók. A táblázat utolsó oszlopában a platformon rendelkezésre álló erőforrások teljes száma olvasható. Az RCTP protokoll az erőforrások terén

a TCP még eme egyszerűsített implementációjával szemben is jelentősen takarékosabb. Ezek az előnyök hatványozottan jelentkeznek, ha a protokollt az adott eszközön több példányban is el kell helyezni, ami a több hálózati interfésszel rendelkező eszközökön eredendően szükséges.

3.3. táblázat A TCP és az RCTP szimulációs összehasonlítása adott Xilinx XC5VLX110T FPGA platformon

Erőforrás	TCP	RCTP	Felhasználható mennyiség
'Slice' regiszterek száma	1 350	449	69 120
'Slice' igazságtáblák száma	2 322	647	
Blokk memóriák/FIFO vermek száma	9	0	148

Az implementációval végzett mérések alátámasztották a protokoll működőképességét. A vizsgált szabályzófüggvények közül az 5%:30%-os tartományban lineáris szabályzást végző kellően hatékonynak bizonyult, így nincs szükség bonyolultabb karakterisztikájú szabályzófüggvények alkalmazására. A vizsgált mérési környezetben rövidebb, akár 10 ms-onként történő mintavételezéssel is stabil működést tapasztaltam, amely a lassabb mintavételezéshez képest gyorsabb konvergenciát eredményez. Az FGPA küldőoldali implementáció az azonos architektúrán történő TCP implementációhoz képest jóval kevesebb erőforrást igényelt, ezáltal használata a beágyazott rendszerekben előnyt jelenthet, hiszen több erőforrás marad egyéb, nem csomagtovábbító funkciók számára.

3.8 Konklúzió

A fejezetben ismertetett transzport protokoll zárt hurkú szabályzást megvalósítva az alternatív protokollokkal összemérhető átviteli teljesítmény elérését teszi lehetővé, miközben a protokoll, különösen a küldő oldalon nagyon egyszerűen implementálható. Ezáltal alkalmazása olyan beágyazott, kis erőforrású környezetekben jelenthet alternatívát, ahol egy komplex protokoll megvalósítása akadályokba ütközik.

4 Az Opus hangkódoló hatékonyságvizsgálata

A digitális beszédtovábbítás térnyerése már az internet terjedése előtt elkezdődött. A globális internet forgalmi mixben az interaktív, illetve folyamatosan küldött (streaming) jellegű szolgáltatások arányának további növekedésére lehet számítani [**38**] [**39**]. Az áramkörkapcsolásra épülő szolgáltatások (mint pl. az LTE előtti mobil hangtovábbítás) a kezdetektől fogva a beszédhang továbbítására optimalizált, egyszerű kódolók (coder/decoder, codec) segítségével oldják meg az adott, rendszerint fix sávszélességű csatorna nyújtotta kapacitás optimális kihasználását. A kódolók többségét (pl. G.711, GSM) már évtizedek óta alkalmazzák, hiszen a végponti eszközök egyre szélesedő skálájával való kompatibilitást a szolgáltatóknak fenn kell tartania.

Az internet elterjedésével párhuzamosan megjelentek azok a szolgáltatások, amelyek publikus, illetve heterogén hálózatokon biztosítanak IP-feletti (Voice over IP, VoIP) hangtovábbítást. A nagyobb elérhető számítási kapacitás és sávszélesség nyitotta új lehetőségek az általános architektúrák szoftverkörnyezetéből fakadó rugalmasságának köszönhetően gyorsan fejlődésnek indultak a hangkódolók is, amelyek adott sávszélesség mellett sokkal jobb minőségre képesek, mint a távközlési szolgáltatók által támogatott hagyományos hangkódolók.

A Skype, mint a publikus hálózaton nyújtott VoIP szolgáltatás úttörője a SILK kódolójában folyamatosan javította az elérhető hangminőséget. Ezt a folyamatot a felhasználók visszajelzésére támaszkodó szubjektív értékelésekkel segítette. A Xiph által Speex néven fejlesztett nyílt forráskódú kódoló fontos előfutára a későbbi CELT–nek, amely az alacsony késleltésű hangtovábbításra fókuszált. Ezek a fejlesztések később összeérve az Opus kódoló formájában egyesítették előnyeiket. Ez a kódoló, köszönhetően a fontos ipari szereplők támogatásának, rövid időn belül Internet Engineering Task Force (IETF) szabvánnyá vált. A lépéssel fontos mérföldkőként jött létre egy olyan, nyílt forráskódú és szabadon felhasználható hangkódoló, amely kiemelkedő minőségű beszédhang továbbítást tesz lehetővé, járulékos licenc díj nélkül.

4.1 Motiváció

Kutatásomat motiválta, hogy az Opus kódolót VoIP környezetben nem vizsgálták korábbi publikációkban. Léteznek olyan minőségbecslő módszerek, amelyek a forrásanyag ismeretében nagy biztonsággal képesek a dekódolt anyag érzeti minőségének becslésére. Ezeket referencia alapú (Full Reference, FR) módszereknek hívjuk. A gyakorlatban rendszerint az eredeti hanganyag nem hozzáférhető. A forrásanyag rögzítése számos kérdést felvet. Hogyan oldható meg a tárolás, a rögzített és a dekódolt anyag szolgáltatóhoz történő megbízható és transzparens eljuttatása? A rögzített anyag felhasználása továbbá jogi aggályokat is felvet. Ezért a referencia alapú módszerek a szolgáltatók számára csak ritkán minősülnek gyakorlatban alkalmazható eszköznek. Ha az eredeti anyag teljes egészében nem, de annak bizonyos jellemzői rendelkezésre állnak, csökkentett referenciájú (Reduced Reference, RR) módszereket alkalmazhatunk a becsléshez. A gyakorlatban ez a VoIP forgalom egy részének, pl. a csomag fejlécek elkapásával történhet. Ez az út még mindig elég távol esik a szolgáltatói felhasználás lehetőségeitől. Kutatásaim ezért a referencia nélküli (No Reference, NR) módszerek felé irányultak, amelyek mérhető metrikák alapján segítik az érzeti minőség becslését.

A Quality of Service (QoS) olyan módszerek, metrikák gyűjteménye, amely a hálózati infrastruktúra üzemeltetőinek segítenek az infrastruktúra bizonyos pontjain a teljesítmény mérésére (pl. sávszélesség, csomagvesztés, késleltetés). A Quality of Experience (QoE) az objektív metrikák mérése helyett a szubjektív minőséget keresi, vagyis a szolgáltatást igénybevevő elégedettségét. Ennek számszerű ábrázolását rendszerint valamilyen egyszerű numerikus skálán oldják meg. Számos kutatás irányult a QoS metrikák és a QoE közötti összefüggések keresésére. Önállóan egy-egy metrika (csomagvesztés, késleltetési ingadozás) jellemzően nem mutatott kellő korrelációt az érzeti minőséggel, ezért az újabb kutatások már inkább a metrikák kombinációival foglalkoznak.

Kutatásaim során az Opus hangkódoló azon fontos tulajdonságára derítettem fényt, hogy más alternatív hangkódolókhoz viszonyítva a csomagvesztés és az érkezési ingadozás QoS paraméterekre lineárishoz közeli összefüggést mutat.

4.2 Beszédhang-kódolók minőségvizsgálata

A VoIP szolgáltatások minőségének vizsgálatára általános gyakorlat a szubjektív értékelés. Egy ilyen értékelés során a felhasználót megkérik a szolgáltatás használata után annak értékelésére. A legjellemzőbb az 5-pontos átlagos minőségérzeti (Mean Opinion Score, MOS) skála használata. Ezen a skálán az elégtelentől (1) a kiválóig (5) jellemezhető a szolgáltatás. Minél nagyobb mennyiségű mintát sikerül begyűjteni, annál jobb képet kaphat a kérdező az adott szolgáltatás átlagos érzeti minőségéről. A szubjektív értékelések módszere ugyanakkor nagyon körülményes, csak kampányszerűen vethető be, esetleg kellően nagy felhasználói tábor esetén szúrópróbaszerűen is alkalmazható. Ezért régóta igény van olyan objektív módszerekre is, amelyek automatizáltan képesek a szolgáltatáshoz kapcsolódó érzeti minőség becslésére.

4.3 Kapcsolódó kutatások

S. Karapantazis és társai átfogó tanulmányt készítettek a VoIP alkalmazások terén [**40**]. Munkájukban áttekintik az érintett QoS metrikákat, a VoIP kódolokat, transzport mechanizmusokat, minőségvizsgálati módszereket, továbbá rávilágítanak, mely aspektusokban várható fejlődés, mely problémák terén (pl. biztonság) van szükség további új módszerek kutatására.

4.3.1 QoE vizsgálati módszerek, QoS–QoE megfeleltetés

A. Raake a VoIP beszédminőség vizsgálat és értékelés általános sarokpontjait foglalta össze munkájában [**41**].

A. Rix és társai átfogó elemzést készítettek az objektív beszédhangminőség vizsgálati technikákkal és alkalmazásokkal kapcsolatban [**5**]. A tanulmány mind „betolakodó", mind a felhasználó számára láthatatlan módszerek használatát is áttekinti.

S. Cardeal és társai bemutatták az ArQoS-t, amely egy szolgáltatói hálózatba helyezhető szonda [**42**]. Elsősorban hálózati teljesítmény paraméterek figyelésére szolgál, de QoE értékelési módszereket is képes bevonni az eredmények elemzéséhez.

W. Cherif és társai A_PSQA néven egy nem betolakodó QoE becslő módszert mutattak be, amely a Véletlen Neurális Hálózatok (Random Neural Network, RNN) megközelítésen alapszik [**43**].

L. Fei és társai a csomagtovábbítás késleltetését és a sávszélességet feltételezik a QoE-t meghatározó tényezőnek [**44**]. Munkájukban egy LTE hálózatokban használható vivőidőzítő modellt (carrier scheduling scheme) dolgoztak ki. Szimulációs eredményeik a QoE komoly javulását mutatták ki.

S. Jelassi és társai átfogó tanulmányt készítettek az objektív és szubjektív QoE értékelési módszerekkel kapcsolatban [**45**].

V. Aggarwal és társai módszerével 80%-os megbízhatósággal jósolható a VoIP szolgáltatás minőségérzete [**46**]. Technikájuk a gépi tanulás segítségével elemzi a passzív megfigyeléssel összegyűjtött QoS mérési adatokat

C. C. Wu és társai egy olyan keretrendszert alkottak, amelyben a kutatók a szubjektív QoE értékeléseket kiszervezhetik, hogy nagy mennyiségű statisztikai minta álljon rendelkezésükre [**47**]. Az eljárás törekszik a nem megbízható értékelések kiszűrésére, a MOS skálán túl összehasonlítható QoE értékelésekkel is dolgozik, a konzisztencia magasan tartása miatt.

4.3.2 Referenciaalapú minőségvizsgálati módszerek

Az ITU számos FR módszert bocsájtott az ipari szereplők rendelkezésére. A Perceptual Objective Listening Quality Assessment (POLQA, ITU-T P.863) módszer a korábbi PEAQ, PSQM PESQ és PESQ-WB módszereket váltja le [**48**]. A módszer a digitális beszédhang matematikai módszerekkel elvégzett analízise segítségével ad becslést az érzeti minőségre, egészen 14 kHz mintavételezési frekvenciáig. A módszer által szolgáltatott eredmények nagymértékben korrelálnak a szubjektív eredményekkel, ezért széles körben alkalmazzák.

Y. Hu és P. C. Loizou kombinált módszerével a korábbi (pl. PESQ) módszerekkel szolgáltatott becslések korrelációja 0,2-vel javul [**49**].

A. Hines és N. Harte módszere, az NSIM (Neurogram Similarity Index Measure) az idegrendszer viselkedésének modellezésével ad becslést az érzeti minőségre [**50**].

A. Hines és társai bemutatták a Virtual Speech Quality Objective Listener (VisQOL) módszert [**51**]. Az új eszköz célja az NSIM időelcsúszás és érkezési ingadozás okozta pontatlanság csökkentése [**52**] [**53**].

M. Alreshoodi és J. Woods áttekintő tanulmányt írtak a különböző QoS–QoE megfeleltetési módszerekről [**54**]. Számos referencia nélküli módszert áttekintenek, bár ezek zömében videótovábbítás jellegű szolgáltatások voltak.

4.3.3 Referencia nélküli minőségvizsgálati módszerek

Neves és társai egy E-modellen alapuló referencia nélküli VoIP QoE megfigyelő módszert dolgoztak ki [**55**]. A módszer C2 megbízhatósági osztályú becslést képes adni a MOS skála segítségével végzett szubjektív értékelésekre.

4.3.4 Beszédhang kódolók

4.3.4.1 Keskenysávú kódolók

G.711

A PCM sémán alapuló kódoló, amely 8 bites értékekre képzi le a 125 μs hosszúságú hangmintákat, 64 kbit/s rátájú folyamot eredményez [**56**]. Az μ-Law (14 bites) vagy A-Law (13 bites) szabványok előírása alapján logaritmikus skálázás után 8 bites tömörített PCM hangmintafolyam keletkezik, a dekódolás is ugyanezen séma szerint történik. A G.711 az integrált digitális hálózati szolgáltatások (Integrated Services Digital Network, ISDN) világának és a H.323 átvitel jellemző kódolója.

G.723.1

Az ITU által szabadalmaztatott kettős rátájú kódoló, amelyet a PSTN telefonvonalak, valamint a H.323 és H.324 hang- és videókonferencia szolgáltatások számára fejlesztettek ki [**57**]. Két üzemmódja a 6,3 kbit/s sávszélességű Multi Pulse Coding-Maximum Likelihood Quantization (MPC-MLQ) és az 5,3 kbit/s rátájú Algebraic Code Excited Linear Prediction (ACELP) algoritmus szerint működik. Fontos újítása a hangaktiválás érzékelés (Voice Activity Detection, VAD), amely a beszédközi szünetek magasfokú tömörítésével igyekszik javítani a sávszélesség kihasználáson.

G.726

A kódolót a G.723 kiváltására hozta létre a Nemzetközi Távközlési Egyesület (International Telecommuncation Union, ITU) [**58**]. Négy különböző bitrátájú üzemmódja van (16, 24, 32 és 40 kbit/s). Leginkább a 64 kbit/s szélességű μ-Law vagy A-Law kódolású, 8 kHz-es PCM csatornák adaptív különbségi PCM (Adaptive-Differential PCM, ADPCM) tömörítésére szánták. Ez 3,5 illetve 3,7 bites felbontású tárolást tesz lehetővé.

G.728

A 16 kbit/s sávszélességű kódoló, amely a beszéd alacsony késleltetésű kódolására szolgál az Excited Linear Prediction (LD-CELP) algoritmus segítségével [**59**]. G jelű melléklete a kódoló fixpontos specifikációját tartalmazza [**60**]. Az I jelű melléklet a csomagvesztést kompenzáló (Packet Loss Concealment, PLC) algoritmust tartalmazza [**61**]. Ez egy olyan hatékony és robosztus kódoló, amely a 32 kbit/s-en dolgozó ADPCM-mel összevethető minőségű tömörítést tesz lehetővé.

G.729

A kódoló fő újítása, hogy képes több párhuzamos beszédfolyam közös adatfolyamba csatornázására, ezáltal javítva a sávszélesség kihasználását [**62**]. Az alapvető algoritmus 8 kbit/s-re kódol. A Conjugerated Structures Algebraic Code Excited Linear Prediction (CS-ACELP) segítségével 10 ms-os kereteket dolgoz fel, az előző kódolókhoz képest meglehetősen komplex módon. Az A jelű mellékletben egy kevésbé összetett, ezáltal alacsonyabb minőségre képes változatot is bemutattak, amely még mindig képes a 32 kbit/s-en üzemelő ADPCM-hez hasonló reprodukcióra [**63**].

A B jelű kiegészítés további funkciókat, mint VAD és nem-folyamatos átvitel (Discontinuous Transmission, DTX), valamint háttérzaj előállítása a szünetek alatt (Comfort Noise Generator, (CNG) is tartalmaz [**64**]. A C és C+ jelű melléklet a lebegőpontos változatot írja le.

A D jelű melléklet egy 6,4 kbit/s-en is üzemelni képes változatot mutat be, így a tovább csökkenő sávszélesség mellett is képes viszonylag kis

minőségromlással is működni [**65**]. A felszabaduló sávszélesség hibajavító információ tárolásához is felhasználható.

Az E jelű mellékletben ismertetett algoritmus a fennmaradó sávszélesség további jobb kihasználását segíti, pl. háttérzaj vagy háttérzene esetén [**66**]. A kódoló 11,8 kbit/s bitrátán működik, amely duplán kódol minden keretet, és a jobb minőségűt tárolja majd el. Az egyik kódoló algoritmus alapja a CS-ACELP, a másik egyes részei pedig az LPC-vel visszafelé kompatibilisek.

GSM

Az Európai Távközlési Szabványok Intézete (European Telecommunications Standard Institute, ETSI) GSM 06.10 teljes rátájú (Full Rate, FR) kódolója volt az első GSM-ben használatos digitális beszédkódoló [**67**]. 13 kbit/s átlagos bitrátán dolgozó algoritmus Regular Pulse Excitation-Long Term Prediction-Linear Predictive (RPE-LTP) kódolást alkalmaz. A 13 bites PCM kódolású mintákat vár a bemenetére, a PSTN-ről érkező 8 bites A-Law kódolású anyagot is átkódolja erre. Mai füllel hallgatva nem nyújt túl jó hangminőséget, de kifejlesztésekor az alacsony számításigény fontos szempont volt.

A GSM 06.20 félrátás (Half Rate, HR) már a számításigényesebb Vector-Sum Excited Linear Prediction (VSELP) algoritmuson alapul [**68**]. A teljes rátás GSM-hez képest némileg alacsonyabb hangminőségért cserébe mindössze 5,6 kbit/s sávszélességet igényel.

A GSM 06.60 továbbfejlesztett teljes rátás (Enhanced Full Rate, EFR) változata már az ACELP-en alapul és 12,2 kbit/s sávszélességet igényel [**69**]. Robosztusságának köszönhetően viszonylag kis csatornakapacitás mellett toleránsabb a hálózati átvitel hibáival szemben.

MELPe

Az Amerikai Védelmi Minisztérium (United State Department of Defence, DoD) Digitális Hangfeldolgozó Konzorciuma (Digital Voice Processing Consortium, DDVPC) ezt a kódolót választotta kereskedelmi és hadászati célú alkalmazásokra egyaránt, mivel a zajos átvitellel és háttérzajjal szemben kiemelkedően toleráns [**70**]. Alapja a Mixed-Excitation Linear Prediction (MELP) algoritmus, amely 1,2 és 2,4 kbit/s bitrátát használ, amelybe 180 darab

8 kHz-es mintavételezésú hangmintát kódol közelítőleg 22,5 ms-onként. Mivel alacsony számítási kapacitás igényű, ezért bevezetésekor hordozható rendszerekben is hatékonyan fel tudták használni.

AMR

Az adaptív többrátás (Adaptive Multi-Rate, AMR) kódolót a Harmadik Generációs Partneri Projekt (3rd Generation Partnership Project, 3GPP) közreműködésével jött létre [**71**]. Kifejezetten 3G távközlési felhasználásra szánták, ahol a vezetéknélküli átvitel megbízhatatlanságát is jól kellett kezelnie. Nyolc különböző alaprátán tud dolgozni és ezek között menet közben is tud váltani. Minden ráta esetén az ACELP kódolást, valamint VAD és CNG algoritmusokat alkalmaz. Az egyik legszélesebb körben alkalmazott kódoló, amely a legmostohább átviteli körülmények között is jó érthetőséggel állítja vissza a digitális beszédfolyamot.

iLBC

A Global IP Sound (GIPS) által kifejlesztett Internet Low Bitrate Codec (iLBC) egy jogdíj mentesen felhasználható kódoló [**72**]. A szabad felhasználás lehetősége miatt számos alkalmazás kezdte el használni, köztül a Skype és a Google Talk is. Két rátát és kerethosszat, 20 ms és 15,2 kbit/s-t, valamint 30 ms és 13,33 kbit/s-t támogat. 304, illetve 400 bites blokkokat állít elő, amelyet blokkfüggetlen LPC-vel dolgoz fel tovább. A blokkok függetlenségének köszönhetően 10%-os csomagvesztésig viszonylag toleráns, efölött viszont jelentősen romlik a minőség.

4.3.4.2 Szélessávú kódolók

G.722

A kódolót az ITU hozta létre, mint az első szélessávú ISDN átvitelre szánt megoldást [**73**]. 7 kHz mintavételezésig képes dolgozni, szemben a keskenysávú kódolók 3,6 kHz-es korlátjával. A hanginformációt két alsávra bontja, és a Sub-Band ADPCM (SB-ADPCM) segítségével kódolja. 48, 56 és 64 kbit/s sávszélekre tud dolgozni, a legmagasabb rátát az alsó alsáv információi tárolása esetén használja ki.

A G.722.1 24 és 32 kbit/s rátákra szánt megoldás [**74**]. 20 ms-os keretekkel dolgozik, amelyeket Modulated Lapped Transform (MLP) algoritmussal kódol. A 640 bites blokkok 50%-ban átlapolódnak.

A G.722.2-t a Nokiával együttműködve fejlesztette ki a VoiceAge [**75**]. A kódolót mind vezetékes, mind vezetéknélküli alkalmazások terén használták. Később Adaptive Multi-Rate Wideband (AMR-WB) néven szabványosították. Kilenc különböző rátát támogat, amelyeket ACELP algoritmus szerint kódol.

AMR-WB+

Ez a változat magasabb mintavételezést és sztereó hangtárolást is biztosít az AMR-WB-hez képest [**76**]. A Transform Coding alkalmazásával a hangminőség is tovább javult.

G.729.1

A korábbi G.729 kódolót továbbfejlesztve megnőtt a mintavételezési frekvencia [**77**]. A kódoló keskenysávú üzemmódot is támogat, amely 4 kHz-ig működik, 8 és 12 kbit/s ráták mellett, ezáltal biztosítva a G.729-alapú rendszerekkel való kompatibilitást.

iSAC

Akárcsak az iLBC, az Internet Speech Audio Codec (iSAC) szintén a GIPS fejlesztése, de ez egy licencdíj-köteles kódoló. (Később megvásárolta a Google WebRTC-s fejlesztéseihez.) Kifejlesztésekor cél volt a PSTN-t meghaladó hangminőség. 10-től 32 kbit/s sávszélességig adaptívan változtatja a bitrátát, de a csomagvesztésre meglehetősen kényes.

4.3.4.3 Vegyes üzemmódú kódolók

Speex

Kifejlesztésekor egy nem pusztán VoIP, hanem széles körben kihasználható, ingyenes kódoló kifejlesztése volt a cél [**78**]. A keskenysávú 8 kHz, a szélessávú 16 kHz és az ultra-szélessávú üzemmód 32 kHz sávszélességű. A kódolás alapja a CELP és 2,2-től 44 kbit/s-ig dinamikusan képes a bitrátát változtatni. Olyan pszichoakusztikai jelenségeket aknáz ki, mint bizonyos hangok alacsonyabb

rátán, mások magasabb rátán történő kódolása. A csomagvesztési toleranciája vezetéknélküli alkalmazásoknál is előnyt jelent. Hátránya, hogy a VBR átvitel miatt nehéz adott csatornakapacitás melletti működés garantálása, ráadásul a maximális bitrátája magasabb lehet az adott pillanatban rendelkezésre álló sávszélességtől.

BroadVoice

A kódolócsaládot a Broadcom fejlesztette ki kábeltévés szolgáltatói hálózatban, illetve digitális előfizetői vonali (Digital Subscriber Line, DSL) hozzáférés fölötti történő VoIP és IP telefóniás felhasználásra [**79**] [**80**]. Keskenysávú változata (Broadvoice16, BV16) 8 kHz mintavételezéssel 16 kbit/s-re kódol. Szélessávú verziója (Broadvoice32, BV32) 16 kHz mintavételezésű mintákat 32 kbit/s szélességű csatornára kódolja.

4.3.5 Az Opus hangkódoló

A. Ramö és H. Toukomaa az Opus MDCT (Modified Discrete Cosine Transformation) és LP (Linear Prediction) üzemmódjait vizsgálta. Szubjektív értékelések segítségével számos klasszikus hangódolóval hasonlították össze őket [**81**]. A vizsgálat a kódolót mindkét üzemmódban a klasszikus konkurens megoldások jó alternatívájának találta.

C. Hoene, J. M. Valin és társai számos cikkben foglalkoznak az Opus további kódolókkal (többek között a Speex normál- és szélessávú változataival) történő összevetésével [**82**]. Kimutatták, hogy az Opus-szal alacsony bitrátán egyértelműen jobb minőséget lehet elérni, bár az AMR változatok szélessávon még jobban teljesítenek.

Az Opus kódolóhoz J. M. Valin és társai további változtatásokat végeztek a kódolás hallható jegyeinek (artifact) minimalizálása végett [**83**].

4.4 III.1. Tézis

Az Opus hangkódolóra vonatkozóan VoIP környezetben végzett mérések analitikus kiértékelésekor QoS-QoE korreláció tekintetében a következőket állapítottam meg. A csomagérkezési ingadozás esetében a QoS-QoE kapcsolat lineáris, a csomagvesztés esetében pedig másodfokú összefüggés van. A

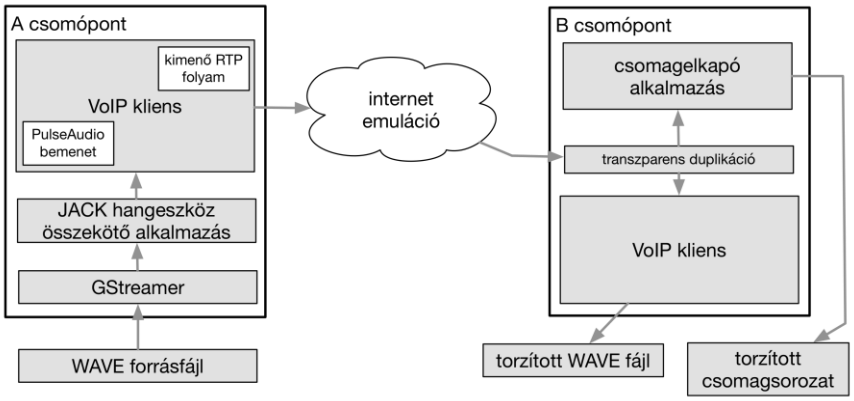
kombinált QoS paraméterek esetén a másodfokú összefüggés szintén jól leírja a korrelációt [**C7**].

4.5 Az Opus hangkódoló vizsgálata

A vizsgálatok célja az Opus kódoló hálózati anomáliák esetén tapasztalható viselkedése, a felhasználói minőségérzetre gyakorolt hatása volt. A Speex, amely az Opus előtt egy széles körben alkalmazott hangkódoló volt, képezte az összehasonlítások ellenpontját.

4.5.1 Mérési környezet

A kódolók összehasonlításához két kommunikációs végpont között nagy távolású hálózati útvonalat emuláló mérési környezetet alakítottunk ki (4.1. ábra). A végpontok többprocesszoros x64-alapú általános architektúrák voltak.



4.1. ábra Mérési környezet: a beszédfolyamot az A csomóponton futó VoIP kliens felé továbbítjuk, amely RTP felett a B csomópont felé továbbítja azt csomagfolyamként.

A végpontokat Intel PRO/1000 hálózati csatolókkal láttuk el és 2 méter ipari minőségű CAT6 kábelezéssel kötöttük össze. Operációs rendszerük Fedora Core 18 volt gyári 3.8.1 Linux kernellel, 1000 Hz-es jiffy-vel. Az sflPhone alkalmazás 1.2.2-es verzióját használtuk a VoIP szolgáltatás emulálásához, mivel az mind a Speex, mind az Opus kódolót támogatja. Az átvitel során tapasztalt forgalmi

mintázat (csomagráta, a beérkezési időközök és csomagméretek egyenletes eloszlása) a várt QoS mérésekkel megegyező volt [**84**].

A forrásanyagot körültekintően választottuk ki. A minta szöveg jól érthető rövid emberi beszéd volt, egy hangcsatornán kódolva. A forrást az A csomóponton futó VoIP alkalmazás hangbemenetére továbbítottuk. Ehhez a JACK Audio Connection Kit szoftvert használtuk, amely egy általános célú hangösszekötő segédeszköz és képes a különböző alkalmazások hang be- és kimeneteinek összekapcsolására [**85**]. Az sflPhone bemenetére ALSA eszköz és PulseAudio forrás köthető. Mivel a PulseAudio összeköthető a JACK-kel, ezért ezt a típusú bemenetet alkalmaztuk.

Mivel ez a VoIP alkalmazás rendelkezik natív JACK kimeneti kiegészítővel (sink), a tömörítetlen impulzus kód modulált (Pulse Code Modulation, PCM) hangfolyamot WAVE (Waveform Audio File Format) fájlformátumban a GStreamer alkalmazással is feldolgozhattuk [**86**]. Az alkalmazásokat körültekintően állítottuk be, hogy ne végezzenek felesleges mintavételezési frekvencia és bitmélység konverziót a digitális hangfolyam teljes útja során. A B csomóponton futó VoIP alkalmazást úgy konfiguráltuk, hogy az érkezett beszédfolyamot tömörítetlen PCM WAVE formátumban mentse el, ezáltal lehetővé téve a későbbi minőségértékelést. A mérések során a NetEm Linux kernel modult használtuk, amelyet mindkét végponton és irányban azonos beállításokkal paramétereztünk azért, hogy szimmetrikus nagy távolságú útvonalat, illetve az azon bekövetkező külöböző hálózati anomáliákat emuláljunk [**87**]. Ezek az anomáliák a QoS-sel mérhető csomagvesztési és érkezési ingadozási paramétereket foglalják magukba. A mérések során a vételi oldalon kapott RTP folyamot is eltároltuk. A forrásanyag első 35 másodpercnyi részét használtuk fel a mérések bemeneteként. Az átvitel során mindkét irányban 100 ms-os késleltetést alkalmaztunk. A hangkódolókat egymástól függetlenül vizsgáltuk, minden mérést ugyanazon paraméterkészlettel végrehajtva. A NetEm paramétereket a 4.1. táblázatban látható séma szerint iteráltuk.

4.1. táblázat A mérések paraméterei

Méréssorozat	Opus	Speex
Ingadozás (ms)	0, 1, 2, 3, ..., 20	
Csomagvesztés (%)	1, 2, 3, ..., 40	
Kombinált: ingadozás (ms) és csomagvesztés (%)	ingadozás: 1, 2, 3, ..., 10 csomagvesztés: 1, 2, 3, ..., 10	

A mérések során kódolónként 160 beszédrészlet keletkezett. A vizsgálatok referenciájának a nulla érkezési ingadozású és csomagvesztésű mérésekkor keletkezett anyagokat használtuk fel.

4.5.2 A mérések kiértékelése

Mindkét kódoló esetében 16 kHz mintavételezési frekvenciát alkalmaztunk, mivel a szélessávú (Wide Band, WB) működés ma már egy természetes felhasználói elvárás. A kódolók szélessávú változó bitrátájú (Variable Bitrate, VBR) üzemmódban dolgoztak a teljes méréssorozat alatt. Az Opus kódoló esetében az sflPhone alkalmazás másodpercenként 100 darab RTP csomagot generált, amelyek 40 és 159 byte közötti méretű csomagokban továbbítódtak 8 ms-os követési időközökkel (500 µs szórással). Az Opus-t korlátozott VBR üzemmódba állítottuk, amely módban a kódoló az átviteli csatorna átlagos kapacitását igyekszik kihasználni a névleges bitráta mellett. A mérések során az Opus névleges bitrátáját 64 kbit/s-ban állapítottuk meg. Az Opus Forward Error Correction (FEC) beállítása a mérések során aktív volt[3].

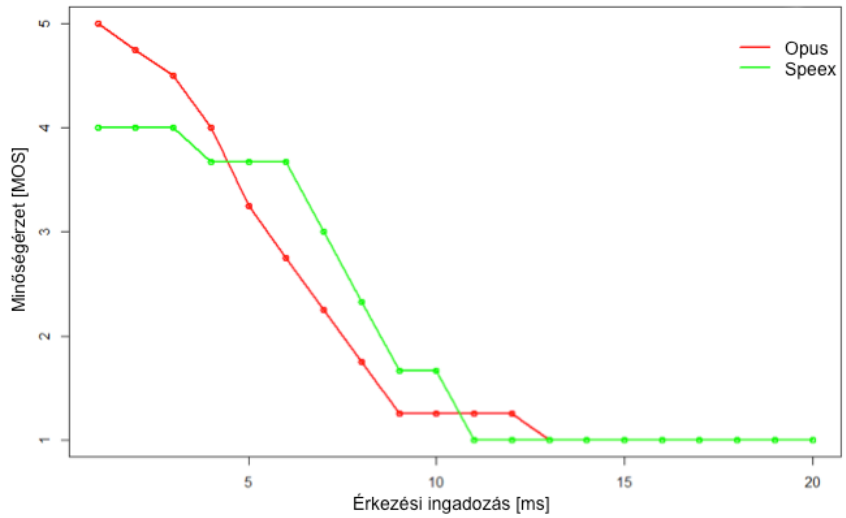
Speex esetében másodpercenként 50 darab RTP csomag haladt át 18 ms-os követési időközzel (1 ms-os szórással). A csomagok mérete mindvégig 124 byte, az átlagos sávszélesség 42 kbit/s volt. A Speex ezen beállítását szélessávúnak tekinti. A korszerű végfelhasználói internetkapcsolatnak ezt a beállítást már ki kell tudnia szolgálni.

A kutatásban részt vevő önkéntesek minden elkapott hangfájlt szekvenciálisan (a romló QoS paraméterek mentén) haladva értékeltek ki. A meghallgatások között a felhasználók szünetet tarthattak és az értékeléseket az 5-pontos MOS skála mentén kellett leadniuk.

[3] Az opció használata a mérési környezet paramétereinek megfelelően irányonként további 10 ms késleltetést eredményez.

4.5.2.1 *Érkezési ingadozással szembeni tolerancia*

Normál körülmények között az RTP csomagoknak korlátozott időablakon belül kell megérkezniük ahhoz, hogy a dekódoló a valós idejű szolgáltatást fenntarthassa és folyamatos beszédhangot állíthasson vissza. A csomagérkezési ingadozást az infrastruktúra (mivel az útválasztók különböző prioritásokat rendelhetnek az egyes folyamokhoz), vagy a fogadó végpont átmeneti túlterhelése okozza, pl. egy hosszabb csomósodás (burst) által, mint amit a pufferek kompenzálni tudnának. Egy valós idejű hangátvitelt végző alkalmazás ingadozáskompenzáló puffere fogadja a beérkező csomagokat. Az alkalmazás eldobja azokat a csomagokat, amelyek a várt időablakon kívül érkeznek. Lejátszáskor ebből a pufferből igény szerint történik a kiolvasás. Ha a lejátszáskor következő csomag nem áll rendelkezésre a pufferben, akkor a lejátszás megszakad. A puffer mérete kompromisszumot kíván: a túl kis méret kevésbé véd az érkezési ingadozással szemben, a túl nagy puffer a megnövekedő késleltés miatt nehezíti az interaktív kommunikációt.



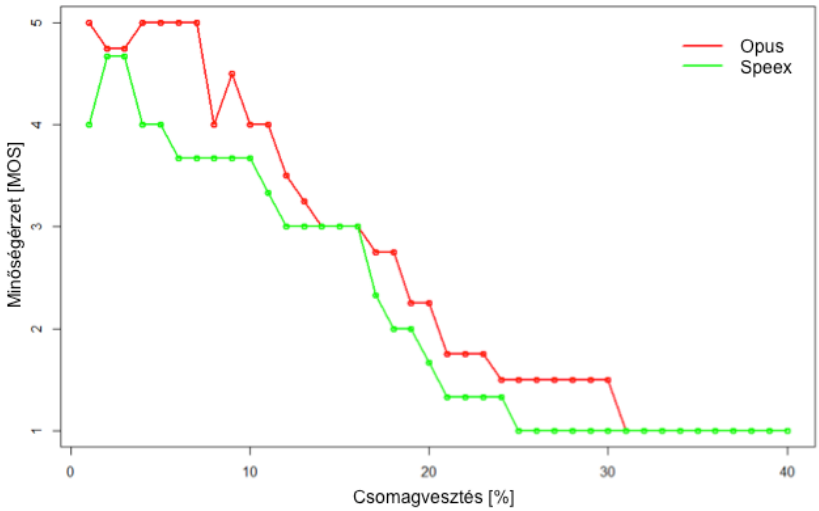
4.2. ábra A késleltetési ingadozás és az érzeti minőség közötti kapcsolat MOS skálán kifejezve

Az Opus kódoló az érkezési ingadozással szemben érzékenyebbnek bizonyult, miközben szélsőséges körülmények között a Speex-hez képes toleránsabban viselkedett (4.2. ábra). Az Opus kódoló használata alacsony késleltetési ingadozás esetén jobb érzeti minőséghez vezetett, valamint a Speex-szel

összehasonlítva akár még 11 ms-os ingadozás esetén is érthetőbb maradt a beszéd. Még a legalacsonyabb, 1 ms-os ingadozás esetén sem értékelte senki 5-ösre a Speex minőségét, mivel az nem zavaró, de mégis jól hallható pattogást okozott. Késleltetési ingadozás szempontjából a Speex szélesebb tartományban nyújt átlagos teljesítményt. Ezzel szemben az Opus 4 ms ingadozás alatt jobb hangminőséget produkál, továbbá fellépő hálózati anomália esetén jobb érthetőséget eredményez a Speex-szel szemben.

4.5.2.2 Csomagvesztéssel szembeni tolerancia

A csomagok elveszhetnek a hálózati útvonalon (rendszerint az útválasztóban), vagy magán a kommunikációs végponton. Nem triviális annak meghatározása, hogy az egyes kódolók milyen hatékonyan tudják az adatvesztést áthidalni.



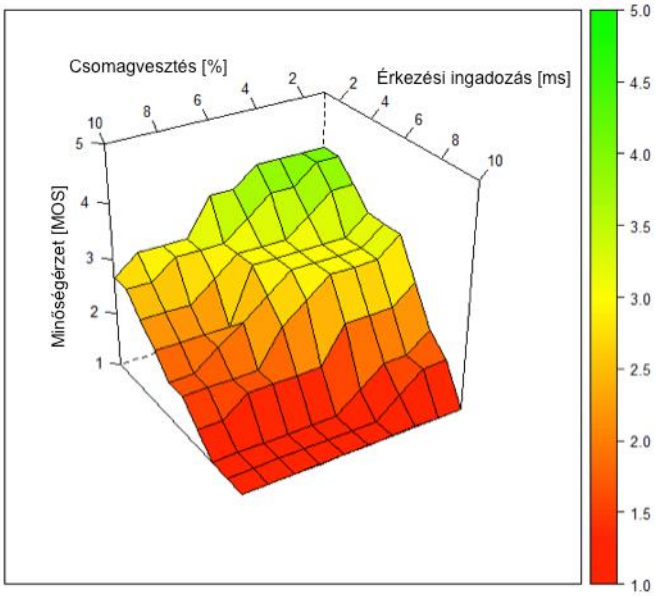
4.3. ábra A csomagvesztés és az érzeti minőség közötti kapcsolat MOS skálán kifejezve

Amint azt a 4.3. ábra is szemlélteti, az Opus kódoló hatékonyabban kezeli a csomagvesztés okozta kiesést. Míg a Speex esetében már 25%-os veszteség mellett is (ami egy 802.11-es vezetéknélküli kapcsolat esetén teljesen életszerű szituáció) értelmezhetetlenné válhat a továbbított beszéd, az Opus még 30%-os veszteség esetén is képes elfogadható reprodukcióra. A felhasználók véleménye megoszlik azt illetően, hogy az Opus kódoló alacsony csomagvesztés esetén mennyire reprodukálja tökéletesen a beszédet, mivel az értékelések alapján 2%-os vesztés esetén jobb eredmények jöttek, mint 1%-nál. Ezt

alátámaszthatja, hogy a kódolóban hatékonyabb pszichoakusztikus modell dolgozik, mint a Speex-ben. Adott veszteség mellett az Opus minősége kevésbé romlott, mint a Speex-é.

4.5.2.3 Többdimenziós toleranciavizsgálat

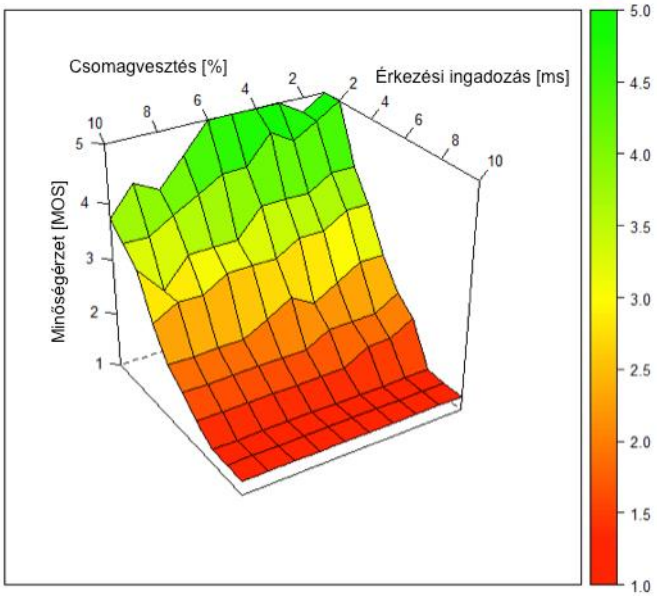
Az előző fejezetekben ismertetett anomáliák rendszerint együttesen fordulnak elő. Valós körülmények között az érkezési ingadozás és csomagvesztés valamilyen kombinációjára lehet számítani. Ennek megfelelően egy olyan, mindkét kódolóra kiterjedő, összetett méréssorozatot állítottunk össze, amelyben a két paraméter együttes hatását vizsgáljuk a szubjektív értékelések függvényében.



4.4. ábra A Speex kódoló értékelései összetett hálózati körülmények között

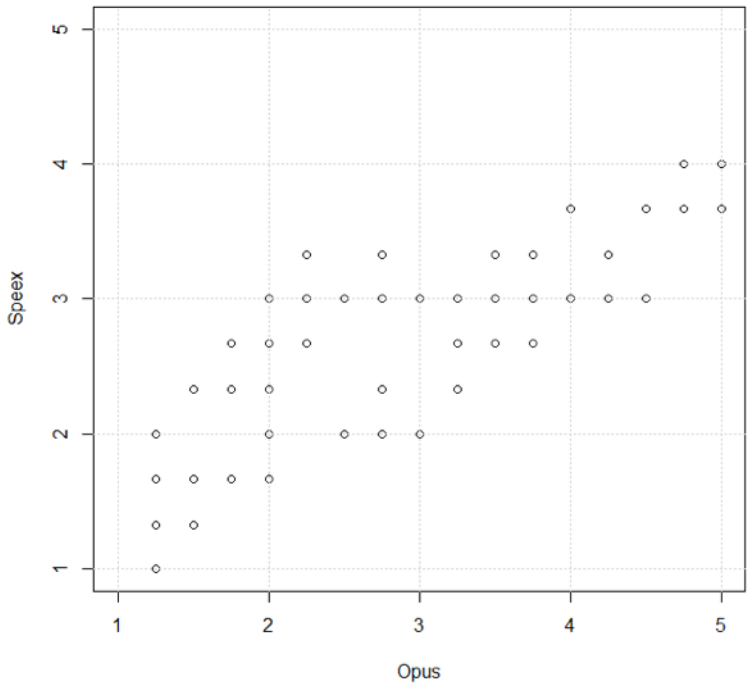
A 4.4. ábra szerint a Speex 3-as MOS értékelést kapott a vizsgált QoS paraméterek egy elég széles tartományában. Ez az érték a MOS skálán az elfogadható érték alsó küszöbét reprezentálja.

Eközben a minőségérzet még kis mennyiségű anomália esetén sem érte el az 5-ös értéket. Ezzel szemben az Opus a szélsőséges viszonyok esetén is egyenletesebb minőséget biztosít (4.5. ábra).



4.5. ábra Az Opus kódoló értékelései összetett hálózati körülmények között

Bár az Opus esetében az érkezési ingadozás növekedése erősebb minőségromlást okoz, a veszteséggel szemben toleránsabb, mint a Speex kódoló. A vizsgálatok alapján megállapítható, hogy az Opus kódolónál érkezési ingadozás tekintetében a QoS–QoE kapcsolatban várható érzeti minőség közelebb áll a lineárishoz, mint a Speex esetében.



4.6. ábra A Speex és Opus MOS értékeinek kapcsolata az összes kombinált mérés esetén

Ahogy az a korábbi mérések során is történt, az értékeléseket az önkéntesek a MOS skála szerint adták le. A kombinált méréseknek (amelyek mind az érkezési ingadozást, mind a csomagvesztést magukba foglalták) kódolónként 100 darab MOS értéket eredményeztek minden egy-egy főre jutó értékeléssorozatnál. A két MOS sorozat közötti korrelációt a 4.6. ábra prezentálja. A korrelációt az (11) alapján számítottuk, értéke 0,8859.

$$r_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_x \sigma_y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_x \sigma_y} \tag{11}$$

ahol μ a valószínűségi változó várható értékét, σ pedig annak szórását jelöli.

4.5.3 Konklúzió

Elmondható, hogy az Opus kódoló vizsgált hálózati anomáliák esetén legalább olyan jó minőségű hangtovábbítást valósít meg, mint az alternatívaként vizsgált Speex. Emellett feltehető, hogy az érzeti minőséggel mutatott összefüggés karakterisztikája a QoS-QoE megfeleltetést is lehetővé teszi.

5 Referencia nélküli minőségbecslés az Opus hangkódolóhoz

A korábbi fejezetben ismertetett vizsgálatok kimutatták, hogy az Opus hangkódoló a korábbi Speex kódoló hatékony alternatívája. Arra a megállapításra jutottam, hogy a kódoló konzisztenesen viselkedik adott csomagvesztés és érkezési intenzitás QoS paraméterekre adott QoE értékelések tekintetében. Ezáltal érdemes olyan becslő eljárást kidolgozni, amely az említett QoS paraméterek ismeretében nagy biztonsággal tud QoE becslést adni az Opus-t alkalmazó VoIP alkalmazás felhasználói minőségérzetét illetően.

A III. tézist alátámasztó vizsgálatok során kétváltozós polinomiális regresszió és négyzetes hibák összege (Sum of Squares due to error, SSE) illeszkedésvizsgálat segítségével megállapítottuk, hogy az érkezési ingadozás és a szubjektív MOS értékek lineáris, míg a csomagvesztés és a MOS értékek négyzetes összefüggést mutatnak. Ezt felhasználva lehetséges egy olyan NR módszer kidolgozása, amely alacsony fokszámú polinomokra építve képes az Opus VoIP kódoló QoS–QoE becslésére.

5.1 III.2. Tézis

A III.1. tézis összefüggései alapján olyan VoIP QoE-becslő módszert hoztam létre, amely megbízhatóan jelzi előre az Opus-alapú VoIP alkalmazás minőségét a csomagvesztés és érkezési ingadozás QoS metrikákból származtatva. A módszer kis számításigényű, mivel alacsony fokszámú polinomokon alapul [**J3**].

5.1.1 Háromfázisú NR módszer

Az eljárás első fázisában elvégeztük a különböző forrásanyagok hálózati hibák emulálása melletti minőségelemzését. A szubjektív értékelések begyűjtése után polinomiális regresszió segítségével megállapítottuk az első fázisban alkalmazott QoS paraméterek és a kapott QoE értékelések közötti kapcsolat fokszámát. Korreláció analízis segítségével állítottuk elő az adott fokszám melletti polinomokban alkalmazandó együtthatókat.

5.1.1.1 A vizsgálatok első fázisa: referencia értékelések

A kutatás során egyre nagyobb figyelmet fordítottunk a mérési környezet kialakítására, a reprodukálhatóságra és kézben tarthatóságra. Ez mind a mérés tárgyául szolgáló anyagok előállítására, mind a szubjektív értékelés minél több aspektusú megszervezésére is kiterjedt. A hálózati anomáliák, mint a korábbiakban említett csomagvesztés és érkezési ingadozás véletlenszerűen következik be, így olyan hálózati emulátort választottunk, ami hasonló módon és statisztikai valószínűségek alapján működik.

A forrásanyag

Egy jó minőségű, jól érthető, férfi beszélő anyanyelvi hanganyagát választottunk ki. A hangoskönyvből származó forrás tömörítetlen CDDA (Compact Disc Digital Audio, 44.1 kHz, 16 bit, sztereó) formátumban volt tárolva. Ezt monó PCM WAVE-ba kódoltuk át a VoIP átvitel kedvéért. Az anyagból csak az első 60 másodpercnyi részt használtuk a mérésekhez[4]. Az ettől hosszabb anyag viszont jelentősen megnyújtotta volna a kiértékelésekhez szükséges időt, valamint bizonytalanná tette volna az önkéntesek által leadott értékeket is[5].

Mérési környezet

Ebben az esetben is nagy távolságú hálózati kapcsolatot emuláltunk két kommunikációs végpont között. Ezek a végpontok általános célú x64-alapú architektúrák voltak, Intel PRO/1000 hálózati csatolókkal és Fedora Core 18 operációs rendszerrel, valamint 1000 Hz-es jiffy-vel. A szoftveres környezetben módosításokat nem végeztünk. A VoIP alkalmazás továbbra is az sflPhone volt [**88**].

A forrásanyagot az A csomóponton futó VoIP alkalmazás bemenetére kapcsoltuk. Ebben az esetben is a JACK alkalmazás szolgált a különböző

[4] Mivel az internetes átvitel emulációját statisztikai alapon működő eszköz végezte, az adott mérés során előírt QoS paraméter értékek csak megfelelő mennyiségű csomag átvitele után érhetők el. A VoIP átvitel forgalmi intenzitása mellett az egy percnyi csomagsorozat ehhez már elegendő hosszúságúnak bizonyult.

[5] A méréssorozatokban részt vevő önkéntesek halmaza nem volt átfedésben sem a jelen, sem a 4. fejezetben ismertett mérésekével. Ennek megfelelően a mérések függetlennek tekinthetők.

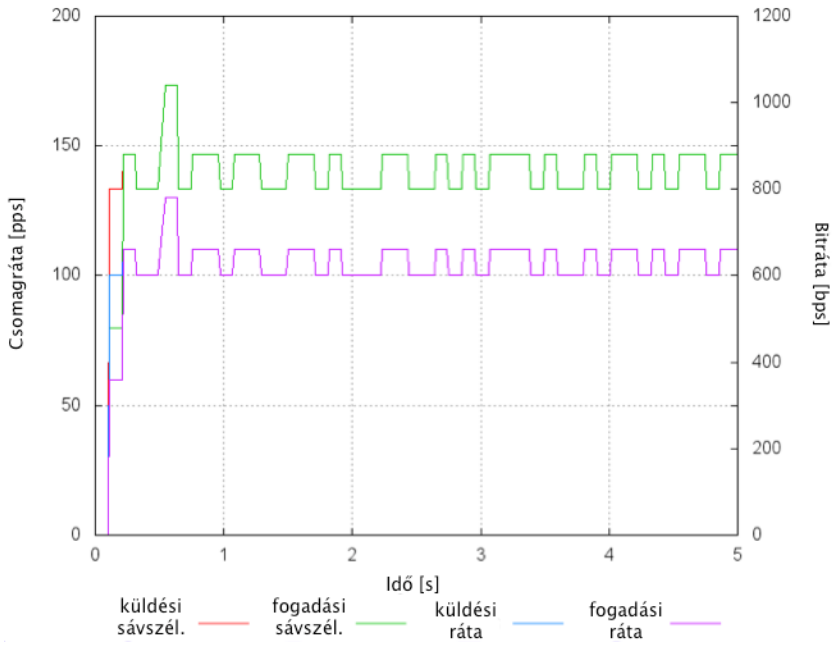
szoftverek hangösszeköttetésére. Az sflPhone-ban a PulseAudio bemenet fogadta a forrást, amelyet a GStreamer alkalmazásból PCM WAVE formátumban kapott meg. Körültekintő beállításokkal kerültük a felesleges újramintavételezést. A B csomóponton futó VoIP alkalmazást úgy állítottuk be, hogy a kapott hanganyagot tömörítetlen PCM WAVE formátuman mentse el a későbbi vizsgálatokhoz. A mérések során a zajcsökkentő és visszhang kioltó beállításokat letiltottuk. A mérések során most is a NetEm kernel modult alkalmaztuk, amely szimmetrikus beállítások mentén torzította a közvetlen kapcsolatot, így emulálva a nagy távolságú összeköttetés során fellépő anomáliákat.

Ellentétben a videótovábbítással, VoIP forgalom esetén nem kell különbséget tenni a különböző csomagtípusok között (pl. nincsenek kulcs keretek), így a NetEm adatkapcsolati rétegbeli (L2) emulációja megfelelő a mérések lebonyolításához. A méréssorozat kielemzéséhez a hanganyagon túl a végponton kapott RTP folyamot is elmentettük [**89**]. Az ITU ajánlások szerint a csomagtovábbítási késleltetés nem haladhatja meg a 150 ms-ot, ha a cél az elfogadható minőségű kommunikáció. Ezért irányonként 100 ms-os késleltetést választottunk a nagy távolságú kapcsolat emulációjához. A kódolót az 5.1. táblázat szerinti NetEm QoS paraméterkészlet segítségével vizsgáltuk.

5.1. táblázat A hálózati átvitel emulációjakor alkalmazott előre meghatározott QoS paraméterek értékei

NetEm paraméterek	Értéktartomány
Érkezési ingadozás	0-20 ms, 1 ms lépésekkel
Csomagvesztés	0-40%, 1% lépésekkel

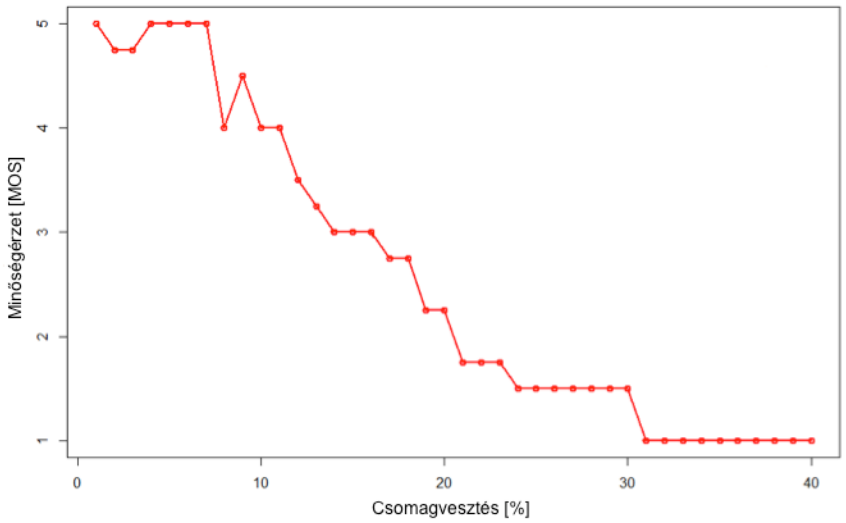
A VoIP alkalmazás másodpercenként 100 darab RTP csomagot állított elő. A csomagméret 40 és 159 byte között változott (5.1. ábra). A kódoló 8 ms-onként állított elő egy hangkeretet. Korlátozott VBR (Constraint VBR, CVBR) üzemmódban működve 64 kbit/s-os névleges bitrátát alkalmazott. Az érkezési időközök ±500 μs között változtak.



5.1. ábra Csomagráta és sávszélesség a hangátvitel során

Szubjektív értékelések

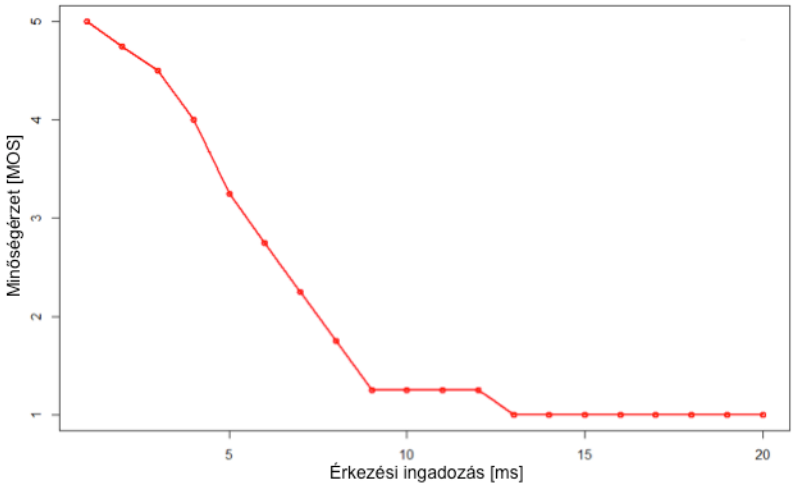
Mivel széles paramétertartományt vizsgáltunk, a méréssorozat során több, mint 100 darab hanganyag keletkezett. Finomabb tartomány alkalmazásával a szám tovább emelkedett volna, így a kiértékelést a nagy mintaszám már túlságosan monotonná tett volna. Az értékelés során referenciának a csomagvesztést és érkezési ingadozást el nem szenvedett átvitelekor keletkezett anyagokat tekintettük. Az egyperces hanganyagokat tizenöt önkéntes segítségével értékeltük. Ők a saját életritmusukhoz igazodva, nyugodt körülmények között végezhették az értékeléseket. Egyikük sem volt VoIP szakember, az alkalmazásokkal és a hangkódolókkal kapcsolatban nem rendelkeztek mély ismerettel, átlagfelhasználóként működtek közre. A monotóniát megtörendő az értékelésekkor a referenciákhoz közeli sorozatok elején szünetet is tarthattak. Minden értékelés elején a referenciát hallgatták meg először. A QoE értékelés során az egyes hangmintákhoz MOS értékeket rendeltek.



5.2. ábra A csomagvesztés és a MOS skálán leadott szubjektív értékelések közötti korreláció

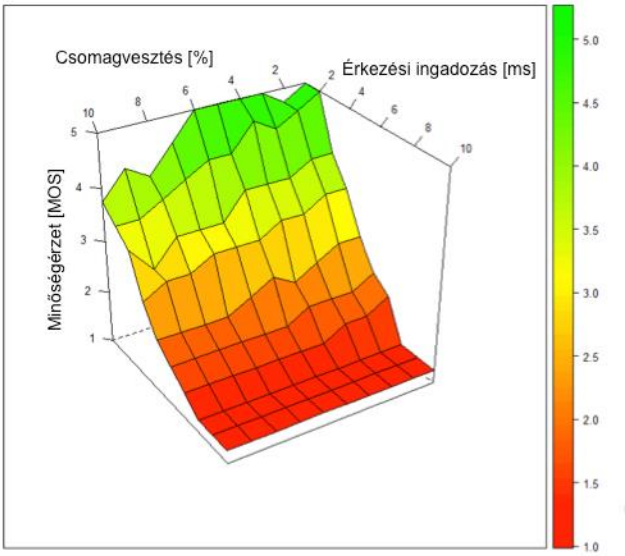
A csomagvesztés növekedése és a minőségromlás között közel lineáris összefüggés mutatkozik (5.2. ábra).

A VoIP alkalmazás az RTP fölött küldött Opus csomagok egészen 30%-os csomagvesztéséig viseli az információkiesést. Ez vezetéknélküli összeköttetés esetén komoly előnyt nyújthat. Az alacsonyabb QoS paraméterek esetén adott alacsonyabb MOS értékek a szubjektív értékelések mellékhatásaként jelentkeztek: az értékelők gyorsan megszokják a jó minőségű anyagot (referencia), amely után a legkisebb minőségromlást is viszonylag nagy mértékű lepontozással büntetnek.



5.3. ábra Az érkezési ingadozás és a MOS skálán leadott szubjektív értékelések közötti korreláció

Az érkezési ingadozással szembeni tolerancia is konzisztens volt, bár távolabb esett a lineáris összefüggéstől, mint a csomagvesztés esetében (5.3. ábra). Ennek oka, hogy a késleltetési ingadozás bizonyos határon túl csomagvesztést is eredményez, mivel – rendszerint a csomósodó (bursty) csomagtovábbítás miatt – a túl korán vagy túl későn érkező csomagokat a transzport protokoll eldobja. Az Opus jól viseli a kismértékű, leginkább 4 ms alatti érkezési ingadozást. Ugyanakkor az ingadozás kompenzálásának hatékonysága a fogadó puffer méretétől is függ. Az attól nagyobb késleltetési ingadozás jelentős minőségromlást okozhat. Vizsgálataink során 9 ms fölött kezdett el nagy mértékben romlani a minőség. Ezzel szemben 10 és 12 ms között még megfigyelhető egy stabil állapot, amikor még a minőség elégséges szinten maradt.



5.4. ábra A többdimenziós hálózati anomáliák melletti esetekben leadott MOS értékelések

Az említett kombinált hálózati anomáliák esetében mutatott tolerancia az egyszerű esetek tapasztalatait erősítette meg. Az Opus csomagvesztés tekintetében egyenletesen teljesített. Ezzel szemben késleltetési ingadozás felléptekor a viselkedés összetettebb (5.4. ábra). Bár a kombinált mérések MOS értékeléseire illesztett felület nem eredményezett túl összetett mintázatot, a QoS–QoE kapcsolat további vizsgálatot tett szükségessé.

5.1.1.2 A vizsgálatok második fázisa: statisztikai analízis

Polinomiális regresszió

Az első fázis eredményeit, vagyis a csomagvesztéshez, érkezési ingadozáshoz és MOS vizsgálathoz kapcsolódó értékelések egy 3-tengelyú grafikonon ábrázolhatók. Polinomiális regresszió segítségével meghatározható az illeszkedő felületet leíró kétváltozós polinom. Ez az (12)-nél látható függvényosztály segítségével állítható elő.

$$F = \{p_n(x) = a_0 + a_1 x + \cdots + a_n x^n\} \tag{12}$$

Az illesztett polinomiális regresszió a (13) polinom osztályai alapján adódik.

$$M(\eta - f^*(\xi))^2 = \min_{\forall f \in F} M(\eta - f(\xi))^2 \tag{13}$$

A (14)-ben található egyenletrendszer megoldásaként határozhatók meg az együtthatók.

$$\begin{pmatrix} 1 & M\xi & \cdots & M\xi^n \\ M\xi & M\xi^2 & \cdots & M\xi^{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ M\xi^i & M\xi^{i+1} & \cdots & M\xi^{i+n} \\ \vdots & \vdots & \ddots & \vdots \\ M\xi^n & M\xi^{n+1} & \cdots & M\xi^{2n} \end{pmatrix} * \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_i \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} M\eta \\ M\eta\xi \\ \vdots \\ M\eta\xi^i \\ \vdots \\ M\eta\xi^n \end{pmatrix} \tag{14}$$

A polinomiális regresszió kiszámítása a *Matlab* alkalmazás *poly*{ij} függvénye segítségével történt. A bementi paraméterek fokszámának változtatásával változtatható az illesztett felület. A másodfokú első változót és első fokú második változót tartalmazó polinomot a (15) szerinti képlettel írhatjuk le. Ezt a *poly21*() függvény hívásával számíthatjuk ki.

$$Z = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy \tag{15}$$

ahol *x* a csomagvesztés arányát, *y* pedig az érkezési ingadozás mértékét jelöli.

5.1.1.2.1 Illeszkedésvizsgálat SSE segítségével

A négyzetes hibák összegének (Sum of Squares due to Error, SSE) módszere meghatározza adott ponthalmaz és a rá illesztett felület távolságát (szórás) a (16) segítségével.

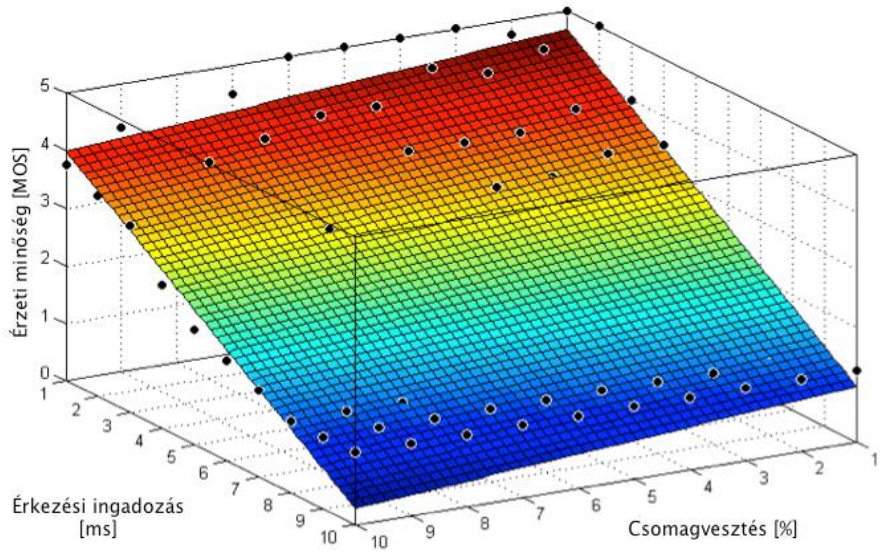
$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{16}$$

Minél nagyobb értéket ad az SSE, annál kevésbé illeszkedik a felület a megadott ponthalmazra.

Felületillesztés – poly11

Az 5.5. ábra a tisztán lineáris modell számára illesztett felületet ábrázolja, ahol a felületet (17) segítségével határozzuk meg, az együtthatókat pedig az 5.2. táblázat tartalmazza.

$$f(x,y) = p_{00} + p_{10}x + p_{01}y \tag{17}$$



5.5. ábra A (17) és az 1. fázisból származó MOS értékek által meghatározott felület

Ez esetben az SSE segítségével kapott illeszkedési jóság (goodness of fit) értéke 16,24, vagyis a kapott felület meglehetősen távol esik a MOS értékelések pontjaitól.

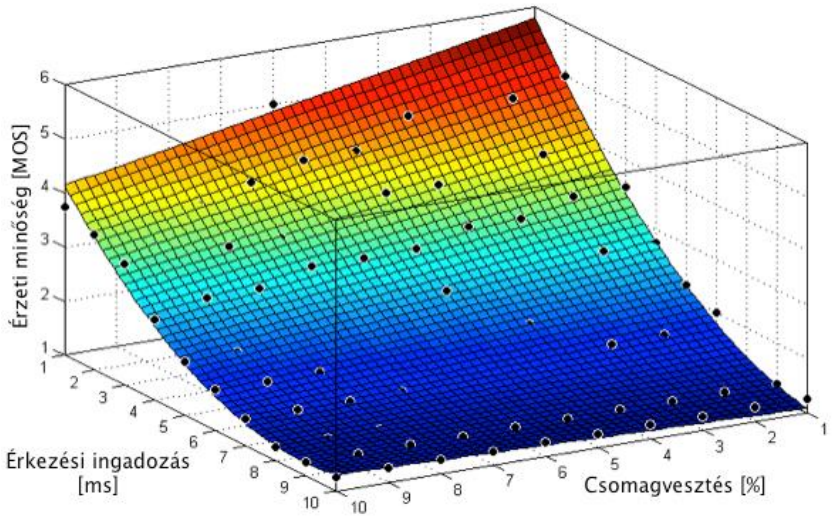
5.2. táblázat A (17) együtthatói

p_{00}	5,151
p_{10}	-0,07513
p_{01}	-0,4111

Felületillesztés – poly12

Az érkezési ingadozás (y) esetében másodfokú változót alkalmazva, a függvény a (18) segítségével írható le. Az együtthatók az 5.3. táblázat tartalmazza.

$$f(x,y) = p_{00} + p_{10}x + p_{01}y + p_{02}y^2 + p_{11}xy \tag{18}$$



5.6. ábra A (18) és az 1. fázisból származó MOS értékek által meghatározott felület

Az illeszkedés jósága 2,336. A másodfokú érkezési ingadozási változó bevezetésével jelentősen javult az illeszkedés mértéke, amint azt az 5.6. ábra is mutatja.

5.3. táblázat A (18) együtthatói

p_{00}	6,985
p_{10}	-0,2052
p_{01}	-1,063
p_{11}	0,02292
p_{02}	0,04696

Felületillesztés – poly22

Ez esetben a csomagvesztés számára is másodfokú változót vezetünk be. A kapcsolódó polinom (19) segítségével áll elő. Együtthatóit az 5.4. táblázat tartalmazza.

$$f(x,y) = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 \tag{19}$$

Az illeszkedés mértéke ez esetben 2,289, amely csak kismértékű javulást jelent a *poly21* esethez képest, ugyanakkor a számításigény az összetettebb polinom miatt tovább nőtt. Ennek oka, hogy a csomagvesztés mentén eleve lineárishoz

közeli összefüggést mutattak ki az értékelések, e változó fokszámának növelése már nem eredményezi az SSE érték javulását.

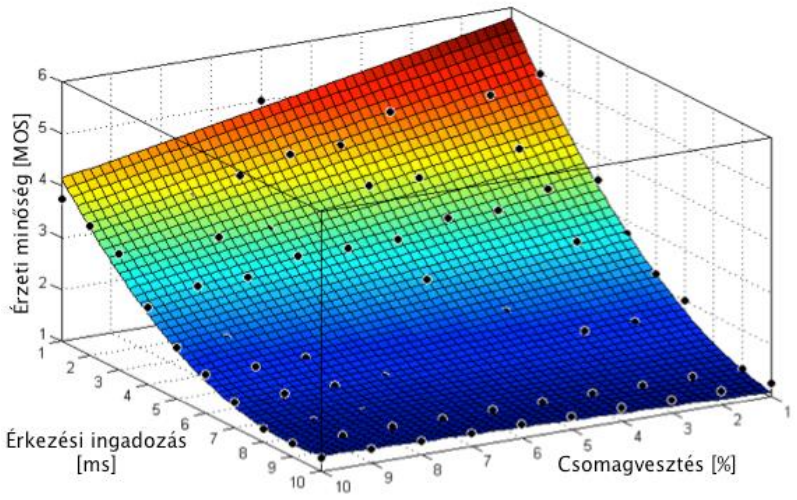
5.4. táblázat A (19) együtthatói

p_{00}	7,067
p_{10}	-0,2382
p_{01}	-1,07
p_{20}	0,003474
p_{11}	0,02217
p_{02}	0,04787

Felületillesztés – poly13

Érkezési ingadozás tekintetében magasabb fokú változó alkalmazását is vizsgáltuk. A (20) által leírt polinom együtthatóit az 5.5. táblázat tartalmazza.

$$f(x,y) = p_{00} + p_{10}x + p_{01}y + p_{11}xy + p_{02}y^2 + p_{12}xy^2 + p_{13}y^3 \qquad (20)$$



5.7. ábra A (20) és az 1. fázisból származó MOS értékek által meghatározott felület

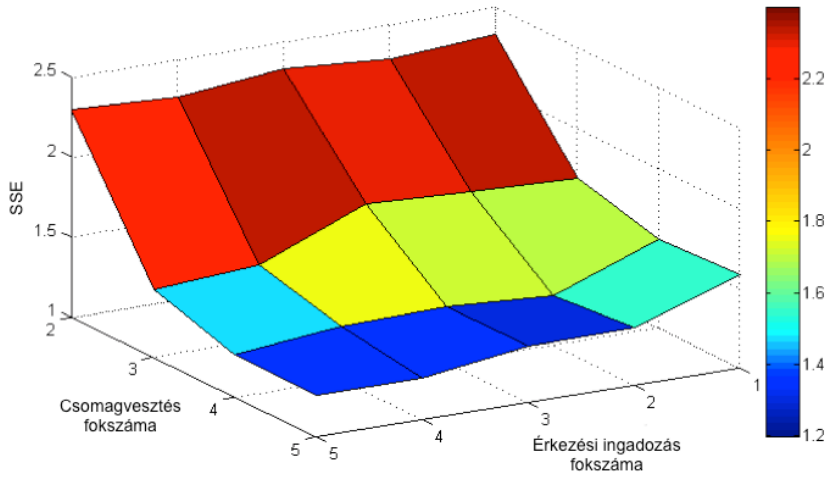
A felület illeszkedési jósága 1,69. Ez lényegesen jobb közelítést jelent ugyan, de a harmadfokú változó használata miatt a számítási igénye is nagyobb (5.7. ábra).

5.5. táblázat A (20) együtthatói

p_{00}	6,728
p_{10}	-0,241
p_{01}	-0,7999
p_{11}	0,04121
p_{02}	-0,01849
p_{12}	-0,001633
p_{03}	0,004354

További felületillesztés

Magasabb fokú polinomokkal leírt felületek illeszkedését is vizsgáltuk. A konkrét együtthatók számát az 5.6. táblázat tartalmazza. Az 5.8. ábra az alkalmazott fokszámok mellett elért SSE értékeket személteti, melyeket számszerűen az 5.7. táblázat tartalmaz. A kalkulációk alapján két kérdés is adódik. Jobb korrelációt eredményez-e a magasabb fokú polinomok alkalmazása? Egy valós idejű NR módszerben történő alkalmazás esetén milyen fokszám jelenti a megfelelő kompromisszumot?



5.8. ábra A polinomiális regresszió során az egyes paraméterek esetében adott fokszám melletti SSE illeszkedési jóság

5.6. táblázat A polinomok együtthatóinak száma adott fokszámú változók alkalmazása mellett

Fokszám	Érkezési ingadozás				
Csomagvesztés	1	2	3	4	5
1	3	5	7	9	11
2	5	6	9	12	15
3	7	9	10	14	18
4	9	12	14	14	20
5	11	15	18	20	21

5.7. táblázat SSE jósági értékek a csomagvesztés és késleltetési ingadozás különböző fokszámú változói mellett

Fokszám	Érkezési ingadozás				
Csomagvesztés	1	2	3	4	5
1	16,2389	2,3363	1,6905	1,5559	1,5829
2	14,1052	2,2895	1,7138	1,3088	1,362
3	13,8641	2,3346	1,7445	1,353	1,3542
4	13,9947	2,2665	1,4665	1,3375	1,2664
5	13,7948	2,3008	1,4275	1,2668	1,2674

5.1.1.3 A vizsgálatok harmadik fázisa: a módszer igazolása

Az utolsó mozzanat fő célja annak vizsgálata, hogy mennyire általános érvényű a második fázisban a polinomok és meghatározott együtthatók alkalmazásakor várható korreláció. Az első fázisban ismertetett mérési környezet változatlan maradt. A vizsgálat végső célja egy lehetőleg alacsony fokszámú becslőfüggvény definíciója, amely bemenő paraméterként objektív, mért értékeket kapva, a MOS skálán határozza meg a QoE becsült értékét.

Forrásanyagok

Négy különböző beszédhangrészletet választottunk ki. Az *A* és *B* anyagokban női, míg a *C* és *D* klipekben férfi beszélő anyanyelvi beszéde volt eltárolva. Minden anyag hangoskönyvből származik, szabványos CDDA (44.1 kHz

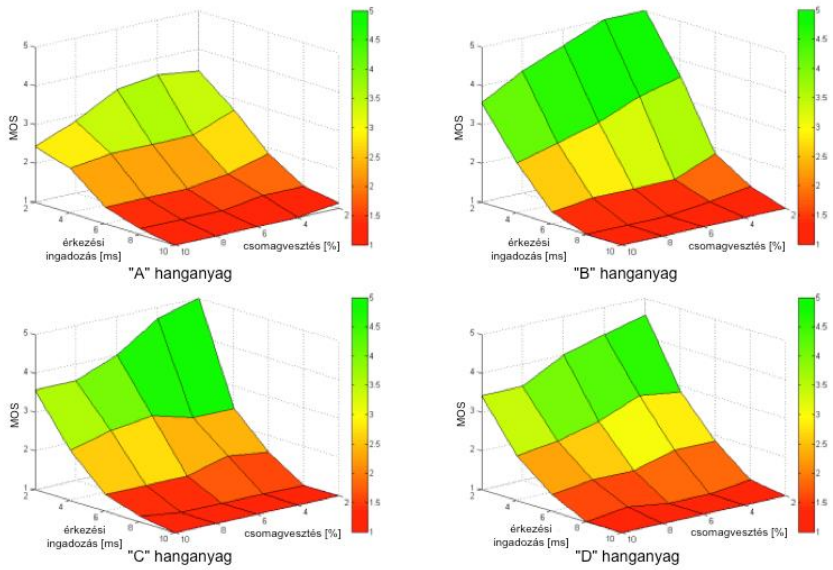
mintavételezési frekvenciájú, 16 bites felbontásban, sztereóban) tárolva. Minden kiválasztott részletet 60 másodperc hosszúra vágtunk.

A QoS-QoE megfeleltetés hatékonyságvizsgálata

Az értékeléseket az első fázisban alkalmazott módszer szerint hajtottuk végre. Jelen esetben tíz önkéntes vett részt a szubjektív értékelésekben. A női hangú A klipben a vártnál alacsonyabb MOS értékek érkeztek. Ez a beszélő intonációjának következménye lehet. Összehasonlítottuk az eredeti forrásanyagot a 64 kbit/s sávszélességű Opus-kódolt mintával, hálózati hibák nélkül. Bár semmilyen zavaró jelenség nem volt megfigyelhető a forrásanyagban, az Opus kódoló időnként megemelte a beszélő hangerejét. A kódoló ilyen viselkedése állhat a klip és beszélő esetén kapott alacsonyabb MOS értékek hátterében.

Elkészítettük a vizsgált QoS paramétertartomány esetében kapott MOS értékekre illesztett térbeli felszíneket. Ezeket az 5.9. ábra mutatja be.

Az *A* minta esetében a MOS skálán 2-től 4-ig terjedő tartományban megközelítőleg lineáris korreláció tapasztalható. 4 ms-nál nagyobb érkezési ingadozás esetén minőségromlás kezdődik, ugyanakkor a csomagvesztés növekedése nem okozott drasztikus leértékelést.



5.9. ábra A különböző csomagvesztési és érkezési ingadozási paraméterek esetén kapott MOS értékelések az összes hanganyag esetében

A *B* klip, amely szintén női beszédhangot tartalmazott, nem mutatott az A-hoz hasonló kódolási anomáliát, ezáltal magasabb átlagos MOS értékeket is kapott. Ez esetben is megfigyelhető a csomagvesztéskor tapasztalt lineárist közelítő kapcsolat. 2-es MOS érték fölött ez az összefüggés az érkezési ingadozás tekintetében is fenn áll.

A MOS értékek és a késleltetési ingadozás korrelációja 2-es MOS érték fölött elsőfokú a férfihangot tartalmazó *C* klip esetében, míg a csomagvesztéssel a másodfokú függvényhez közelít.

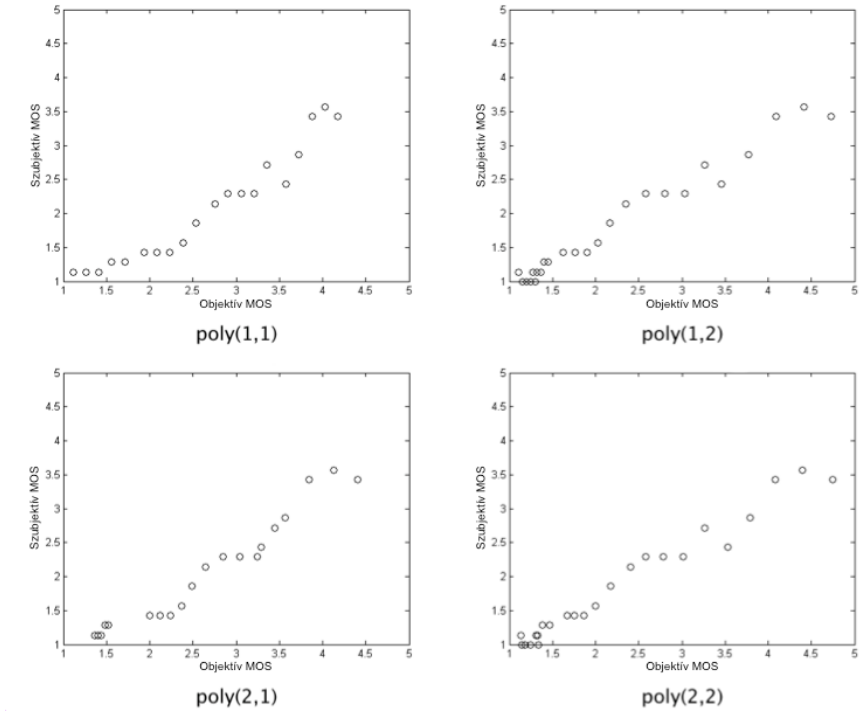
A *D* minta a *C*-hez hasonló karakterisztikájú, de kissé alacsonyabb pontszámokat kapott. Ennek oka szintén a forrásanyagban keresendő.

A felületillesztés eredményei alapján megállapítható, hogy csomagvesztés tekintetében lineáris az összefüggés a MOS értékekkel, annak 2 és 4 közötti tartományában. Ez kellően jól közelíti az érzeti minőséget. Feltételezzük, hogy a leginkább fontos MOS tartomány a 3 és 4 közötti. Ez az a szint, amikor a felhasználó gondolkodóba esik a szolgáltatás használhatóságát illetően [**80**]. A 4-es érték már egy hosszú távon is elfogadható szolgáltatási színvonalat reprezentál, míg 3 alatt hosszútávon zavaró lesz a szolgáltatás használata a rendszeresen hallható anomáliák miatt.

Megfigyelhető továbbá, hogy bizonyos esetekben az Opus kódoló érzékeny a beszélő intonációjára. Minden beszédminta a VoIP szolgáltatás elérhető szintjétől jobb minőségű forrásból származott, de a mérések során alkalmazott bitráta hallható kódolási jelenségeket is produkált, amelyek a csomagvesztés, illetve érkezési ingadozás emelkedésekor drámaian rombolják a minőségérzetet is.

A polinomiális regresszió eredménye

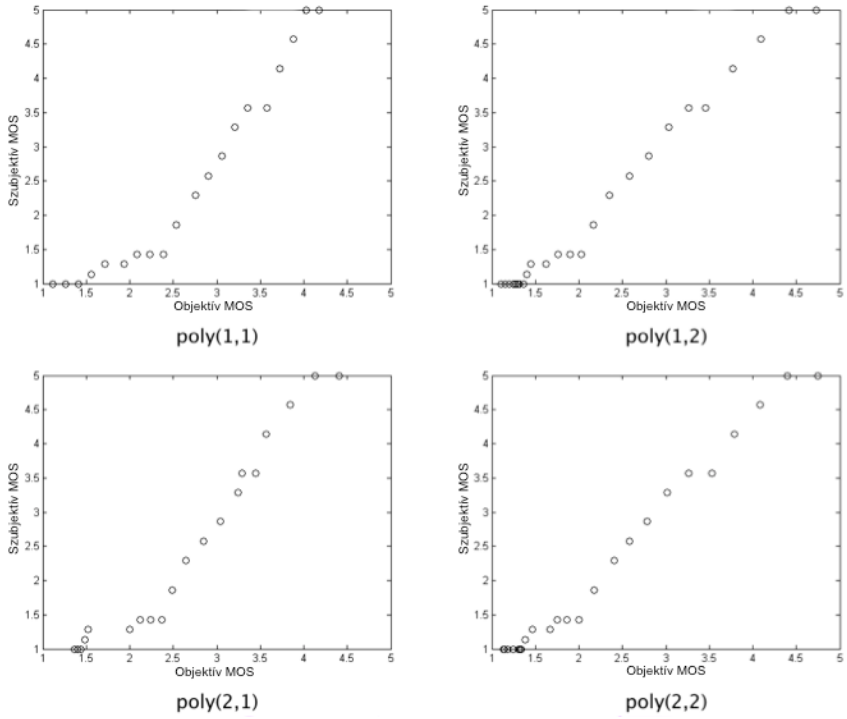
A második fázis tanulságai alapján egy olyan becslőfüggvényt definiáltam, amely legfeljebb másodfokú változók használata mellett is kielégítő hatékonyságot mutat. Hardverben történő implementáláskor az alacsony számítási igény miatt az alacsonyabb fokszám különösen előnyös lehet.



5.10. ábra Korreláció különböző fokszámú polinomok használatával, az A hanganyag esetében

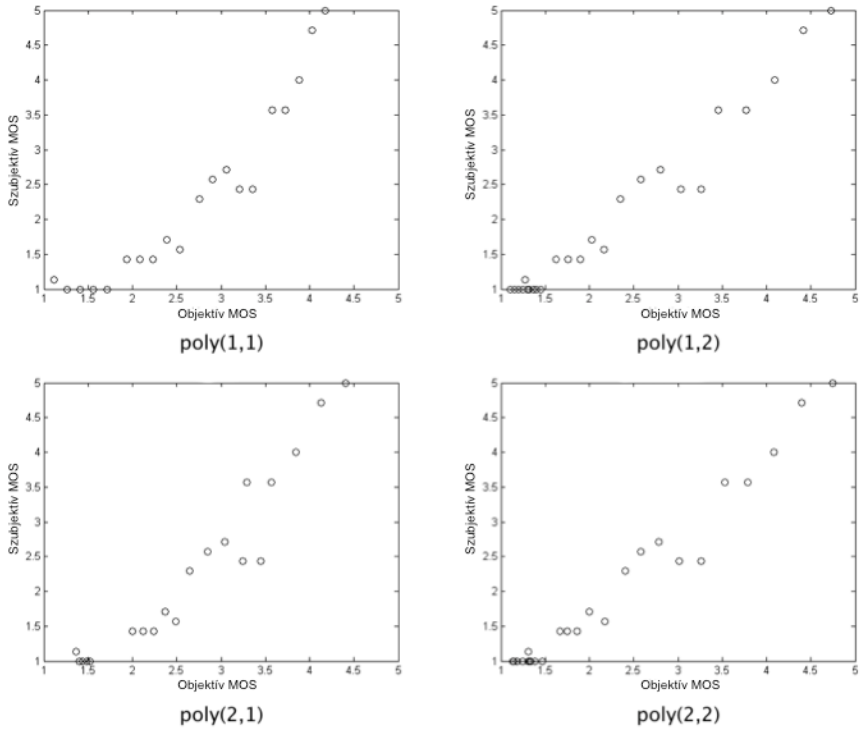
Az egyes hanganyagok esetében leadott MOS értékelések és az objektív paraméterek közötti korrelációt a következő négy ábracsoport mutatja be. *Poly*(x,y) a regressziós függvény, amelyben az *x* a csomagvesztést, *y* az érkezési ingadozást jelentő paraméter.

Az *A* minta esetében jó korrelációt a MOS skála mindkét QoS paraméterrel csak a 2 és 2,5 közötti tartományában mutat (5.10. ábra). A csomagvesztés esetében alkalmazott másodfokú változót fogadó függvénnyel csak kevéssé emelkedik a korreláció értéke, leginkább a MOS skála alján mutat erősödést. Az érkezési ingadozás esetében másodfokú változót használva a korreláció nemcsak az alacsonyabb MOS értékek esetében, hanem a teljes skálán erősödik. Mindkét változó másodfokúra emelése már nem eredményezi a korreláció további jelentős erősödését.



5.11. ábra Korreláció különböző fokszámú polinomok használatával, a B hanganyag esetében

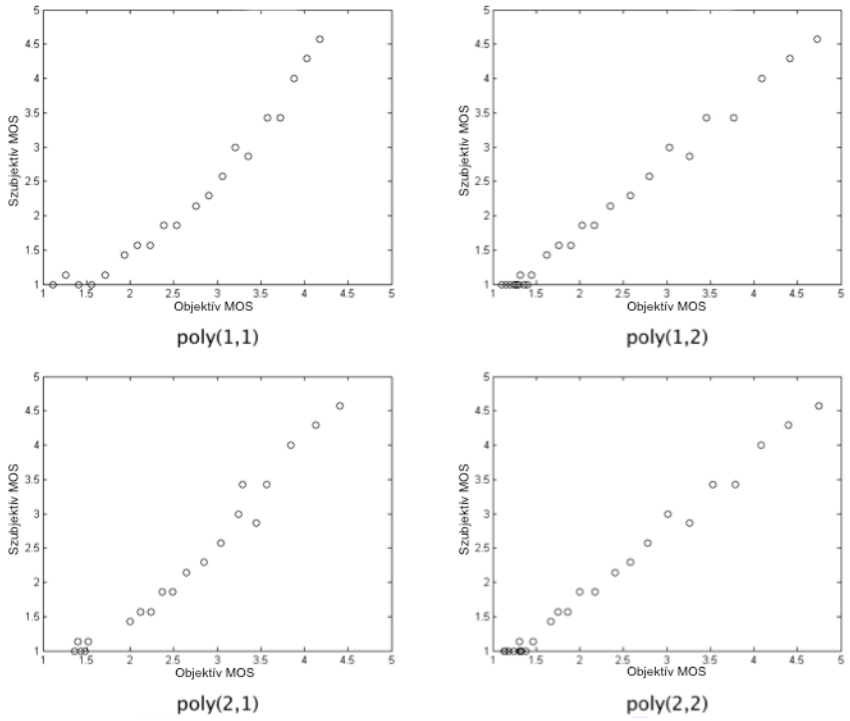
A *B* minta esetében a lineáris függvények csak 92,21%-os korrelációt mutatnak, és ez másodfokú csomagvesztési paraméterrel sem javul érezhetően (5.11. ábra). Másodfokú érkezési ingadozás paraméterrel viszont jelentős az erősödés, egészen 99%-ot ér el a korreláció. A 2 és 4 közötti MOS tartományban minden esetben lineáris összefüggés tapasztalható.



5.12. ábra Korreláció különböző fokszámú polinomok használatával, a C hanganyag esetében

A *C* férfi beszédhang anyag hasonló eredményeket hozott, de az előbb említett tartományban nem olyan egyenletes az eredmény, mint az előbbi esetekben (5.12. ábra). A másodfokú érkezési ingadozás paraméter jelentette korreláció erősödést ebben az esetben is tapasztalhattuk.

A *D* minta esetében gyakorlatilag a teljes MOS skálán egyenletes eredményeket tapasztaltunk (5.13. ábra).



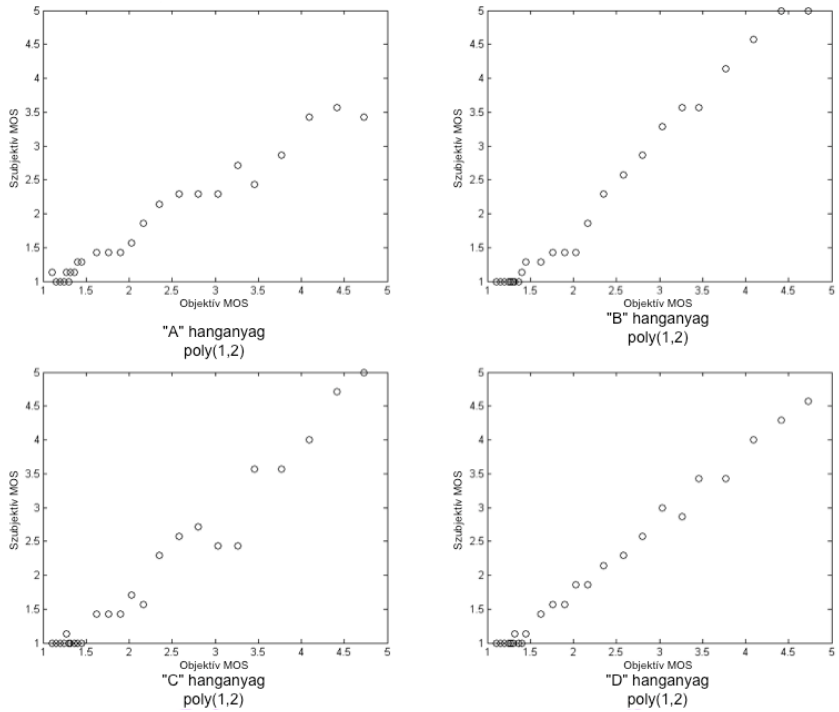
5.13. ábra Korreláció különböző fokszámú polinomok használatával, a D hanganyag esetében

Minden hanganyag esetében mind az első, mind a másodfokú paraméterekkel rendelkező polinomok esetében tapasztalt korrelációs értékeket az 5.8. táblázat foglalja össze. A *poly*(1,2) függvény jó kompromisszum lehet számítási kapacitásban szűkös alkalmazások esetén is.

5.8. táblázat A regresszió analízis korrelációs értékei első- és másodfokú polinomok esetén. Az első paraméter a csomagvesztést, a második az érkezési ingadozást jelöli.

	hanganyag			
	A	B	C	D
poly(1,1)	0,9454	0,9221	0,9041	0,9361
poly(1,2)	0,9861	0,9928	0,9817	0,9962
poly(2,1)	0,9539	0,9324	0,9159	0,9469
poly(2,2)	0,9856	0,9927	0,9828	0,9962

A számítások alapján a *poly*(1,2) által generált polinomot javaslom gyakorlati alkalmazásra, mivel az alacsony fokszám miatt nem túl nagy a számításigénye, de mégis jól közelíti a szubjektív értékelések eredményét (5.14. ábra). A változók fokszámának további növelése már nem növeli jelentősen a becslés pontosságát.



5.14. ábra A szubjektív értékelések és a MOS skálán adott becslések közötti korreláció, érkezési ingadozás esetén elsőfokú, csomagvesztés esetén másodfokú változókat alkalmazó becslőfüggvény használata esetén

Az eredmények alapján megállapítható, hogy Opus kódoló esetében az első fokú polinomokra épülő becslőfüggvény a csomagvesztés és érkezési ingadozás QoS paraméterek mért értéke alapján 90%-os megbízhatósággal képes megbecsülni az érzeti minőséget. Nagy számú VoIP kommunikáció valós időben történő becsléséhez hardverben megvalósított implementáció jelenthet hatékony megoldást. Ilyen esetben a magasabb fokszámú polinomok használata akadályba ütközhet. Másodfokú polinomok alkalmazásával a megbízhatóság 98%-os szintig emelkedik. Ez akár olyan rendszerekben is megvalósítható, amelyekben a lebegőpontos számításokhoz nincs natív támogatás. A módszer használható további kódolók minőségbecslő függvényeinek definiálásához is.

5.2 Konklúzió

A fejezetben alkalmazott háromfázisú módszerrel tetszőleges kódolót és azonos hangkerettípusokat alkalmazó VoIP alkalmazás számára definálható olyan becslőfüggvény, amely mérhető QoS értékek alapján referencia nélkül képes az érzeti minőség (QoE) becslésére a MOS skála szerint.

6 Összegzés

A dolgozatban ismertetett tézisek a hálózati forgalom mérés és analízis területét átfogó módszereket és egy protokollt mutatnak be.

Az I. téziscsoport a hálózati forgalmi mérés terén alkalmazható új, hatékonyabb időbélyegzési módszert mutat be, amely a pontosabb időbélyegzés mellett a csomagfeldolgozás hatékonyságát is javítja. Az I.1. tézisben egy új időbélyegzési módszer ismertettem, amely az időbélyeg előállítását két fontos fázisra bontja. A kiolvasási lépést meghagyja az időbélyegzés eredeti pontján, míg a konverziós lépést áthelyezi egy későbbi, a csomagfeldolgozást közvetlenül nem érintő környezetbe. Az így előállított időbélyegek nem torzulnak. Ezt az állítást laborkörnyezetben végzett mérésekkel igazoltam.

Az I.2. tézisben a tehermentesített szoftveres időbélyegzés csomagvesztést csökkentő hatását mutattam be. Különösen nagy sebességű kapcsolatokon, nagy érkezési intenzitás esetén a szoftveres időbélyegzés jelentős erőforrásigényt von maga után, amit az új módszer hatékonyan csökkent. Ezt az állítást szintén laborkörnyezetben történt mérésekkel támasztottam alá.

Az I.3. tézis a tehermentesített időbélyegzési módszer pontosságot növelő hatását mutatta be. Mivel a módszer hatására csökkent a csomagfeldolgozás kritikus szakaszában az időbélyegzés erőforrásigénye, ezért az egy csomagra jutó feldolgozási idő lerövidül, ezáltal az előállított időbélyegek a referencia időhöz közelebb kerültek. Ezt ismét laborkörnyezetben tervezett és végrehajtott mérésekkel igazoltam.

A II. téziscsoport egy új — a forgalmi mérések során begyűjtött adatok továbbítását segítő — alacsony erőforrásigény mellett nagy átviteli teljesítményt nyújtó transzport protokollal kapcsolatos eredményeket öleli át. A II.1. tézisben az RCTP protokollt mutattam be, amelyet egy konkrét mérőeszköz prototípus rendszerében implementáltam. Az ebben a laborkörnyezetben történt mérésekkel elemeztem a protokoll hatékonyságát. A mérések során sikerült csomagvesztés nélkül, a TCP-t megközelítő átviteli teljesítményt megvalósítani, miközben az implementáció erőforrásigénye a TCP-ének töredéke.

Több eltérő karakterisztikájú függvényt is megvizsgáltam, keresve a leghatékonyabb szabályzást végzőt. A II.2. tézis ennek eredményét összegzi.

A III. téziscsoport a hálózati forgalmi analízis terén az RTP feletti Opus kódolóval működő VoIP alkalmazás minőségvizsgálatához kapcsolódik. A III.1. tézis annak a kutatásomnak eredményét összegzi, amely az Opus kódoló viselkedését annak népszerű elődjével, a Speex-szel hasonlította össze. Ehhez laboratóriumi méréseket terveztünk, amelyben módszeresen emulált hálózati anomáliák mellett torzítottuk az átvitelt. Az eredményeket önkéntesek bevonásával értékeltük ki. A vizsgálatok kimutatták, hogy az Opus kevésbé reagál hektikusan a kombináltan jelentkező anomáliákra, ezért további mérésekkel és értékelésekkel részletesebben vizsgáltuk meg a kódolót.

Statisztikai módszerek bevonásával sikerült feltárni az összefüggést a mérhető QoS értékek (csomagvesztés, érkezési ingadozás) és a szubjektív QoE között. Polinomiális regresszió segítségével előálltak a különböző fokszámú polinomok és együtthatók, amelyek segítségével hatékonyan becsülhető az érzeti minőség. Egy háromfázisú módszer formájában mutattam be, hogyan lehet – akár egyéb VoIP kódolókra alapozva – feltérképezni az adott kódoló viselkedését, a mérési környezet kialakítását és a szubjektív értékelések végrehajtása után statisztikai módszerek segítségével referencia nélküli becslő módszert készíteni. Az eljárást III.2. tézisben foglaltam össze.

6.1 Tézisek listája

I. téziscsoport: Tehermentesített szoftveres időbélyegzési módszer

I.1. tézis: Új szoftveres időbélyegzési módszert dolgoztam ki, amely az óraforrást valós idejű késleltetéssel (megszakítási kontextusban) olvassa ki, míg a konverziós lépéseket áthelyezi késleltetéstűrő felhasználói végrehajtási módba.

I.2. tézis: Kimutattam, hogy az időbélyegzés konverziós fázisának késleltetéstűrő végrehajtási módba történő áthelyezésével jelentős csomagfeldolgozási teljesítménynövekedés érhető el, amely a

csomagfeldolgozást érintő erőforrások szaturációjakor a csomagvesztés arányát csökkenti.

I.3. tézis: Az időbélyegzés óraforrás kiolvasó elemi műveletének megszakítási kontextusban tartása és a konverzió célirányos késleltetése az időbélyegzési precizitást növeli.

II. téziscsoport: Az RCTP torlódás megelőző transzport protokoll és az alkalmazott szabályzófüggvények hatékonysága

II.1. tézis: Kidolgoztam az RCTP torlódás megelőző transzport protokollt, amely az TCP alternatívákhoz képest alacsonyabb erőforrásigény mellett képes – akár vonali rátán is – elkerülni a csomagvesztést.

II.2. tézis: Az RCTP protokoll szabályzófüggvények vizsgálata után megállapítottam, hogy az 5 és 30% között lineáris karakterisztikájú függvény — 0,2-es büntető, 0,1-es jutalmazó konstansokkal alkalmazva — biztosítja a leghatékonyabb szabályzást a protokoll számára. Ezen túl a rendszerparaméterek megfelelő megválasztásával nem szükségszerű a mintavételezési periódusidő csökkentése, bár a maximális átviteli teljesítmény gyorsabban elérhető a rövidebb mintavételezési periódusidővel.

III. téziscsoport: Az Opus hangkódoló VoIP környezetben végzett vizsgálatai során a QoS paraméterek és QoE értékelések viszonya és a tapasztalt viselkedés alapján létrehozott becslési eljárás és az annak megfelelően definiált becslő függvény

III.1. tézis: Az Opus hangkódolóra vonatkozóan VoIP környezetben végzett végzett mérések analitikus kiértékelésekor QoS-QoE korreláció tekintetében a következőket állapítottam meg. A csomagérkezési ingadozás esetében a QoS-QoE kapcsolat lineáris, a csomagvesztés esetében pedig másodfokú összefüggés van. A kombinált QoS

paraméterek esetén a másodfokú összefüggés szintén jól leírja a korrelációt.

III.2. tézis: A III.1 tézis összefüggései alapján olyan VoIP QoE-becslést lehetővé tevő módszert hoztam létre, amely megbízhatóan jelzi előre az Opus-alapú VoIP alkalmazás minőségét a csomagvesztés és érkezési ingadozás QoS metrikákból származtatva. A módszer kis számításigényű, mivel alacsony fokszámú polinomokon alapul.

7 Summary

Theses introduced in the dissertation aim creating methods for traffic measurement and analysis, as well as a new transport protocol.

Thesis group I introduces a new, more efficient method for timestamping. The new method helps to increase packet processing performance, reduce packet loss ratio on saturated systems, and also improves timestamping precision. In Thesis I.1. I separate the two phases of timestamp generation from each other. The first phase (clock reading function) is kept at the original entry point. While the second phase (conversion) is relocated to a latency-tolerant execution context. This approach is called an offloading technique. On non-saturated systems timestamps generated this way are very similar to the ones generated by the generic method. The method was evaluted in a controlled laboratory environment.

Timestamping involves additional resources during packet processing. In Thesis I.2. I evalutate reduction effect of the new method on packet loss in saturated systems.

In Thesis I.3. I point out how the offloaded timestamping method improves timestamp precision. This effect is due to the shorter execution path in the critical packet processing phase.

I introduce a new transport protocol in thesis group II. The new protocol described in Thesis II.1. was created to fulfil the requirements of a network traffic measurement system. The main factors were low implementation complexity and resource demand, as well as line rate packet forwarding and lossless transmission. The protocol was evaluated in controlled laboratory environment.

In Thesis II.2. the impact on throughput and lossless transmission of different control functions were evaluated. Performance comparable to TCP was achieved while the protocol's resource demand is kept low.

Thesis group III is related to the Opus audio codec in a VoIP environment. Thesis III.1. summarizes the observations of voice transmission using Opus compared to different audio codecs. The relation of QoE and two QoS metrics is described. The measurements were executed in a controlled laboratory

enviroment and a number of volunteers were involved into the subjective evaluations.

In Thesis III.2. a method built up from three phases is deployed to create a no-reference QoE prediction method for the Opus audio codec. The first phase is based on subjective evaluations. In the second phase statistical analysis is used to correlate QoS and QoE. Using polynomial regression polynomials of several degrees and their coefficients are calculated. These polynomials and coefficients are used as a base of the prediction. The distance between the real QoE values and the predicted values is evaluated using surface matching. This technique helps us to decide what degree of polynomials to use in a real environment. In the third phase the polinomals and coefficients are evaluated against four different voice clips. The analysis prooved the second degree polnomials being 98% reliable for predicting QoE based on two QoS metrics.

7.1 List of Theses

Thesis group I**.:** An offloaded software timestamping method.

Thesis I.1.: I worked out a new software-based timestamping method, that queries clock source with real-time latency (in the interrupt context), while conversion operations are relocated into the delay-tolerant user space context.

Thesis I.2.: I pointed out that relocating the conversion phase of timestamping to delay-tolerant execution context improves packet processing performance significantly. This processing power gain helps to decrease packet loss ratio when system resources are saturated at a high system load.

Thesis I.3.: Keeping the clock source read operation in interrupt context and targeted delaying of the conversion improves timestamping precision.

Thesis group II.: Rate Control Transport Protocol (RTCP) for a high performance distributed traffic monitoring system.

Thesis II.1.: I worked out a new transport protocol called Rate Control Transport Protocol (RCTP) aiming at congestion avoidance. This protocol puts significantly lower demand on resources compared to TCP, while it is able to transmit packets to the destination even at line rate without loss.

Thesis II.2.: During the inspection of control functions I pointed out that control function doing a linear control between 5% and 30% using 0.2 as penalty constant and 0.1 as rewarding constant ensures the best performance for RTCP. Moreover I pointed out that by specifying the appropriate system parameters using higher sampling frequency is not necessary. However convergence to the line rate can be achieved faster when using a higher sampling rate.

Thesis group III.: Opus audio codec voice quality evaluation and QoE prediction for VoIP application based on the Opus audio codec.

Thesis III.1.: During the analysis of the measurement of Opus audio codec in VoIP environment I determined the following facts on QoS-QoE correlation. There is a linear relationship between QoS parameter jitter and QoE, there is a quadratic relationship between QoS parameter packet loss and QoE. The correlation of combined QoS parameters and QoE can be described using a quadratic relation.

Thesis III.2.: I constructed a No Reference estimation method based on low degree polynomials for VoIP applications using the Opus voice codec to reliably predict QoE. The input parameters of the prediction function are the jitter and packet loss QoS metrics.

8 Publikációk listája

8.1 Folyóirat cikkek

[J1] Orosz, P., **Skopkó, T.**: Performance Evaluation of a High Precision Software-based Timestamping Solution for Network Monitoring. Int. J. Adv. Softw. 4 (1/2), 181-188, 2011. EISSN: 1942-2628.

[J2] **Skopkó, T.**: Loss Analysis of the Software-based Packet Capturing. CJECE. 5 (1), 107-111, 2012. ISSN: 1844-9689.

[J3] Orosz, P., **Skopkó, T.**, Nagy, Z., Lukovics, T.: A No-reference Voice Quality Estimation Method for Opus-based VoIP Services. 1942-2601 7 (1-2), 12-21, 2014

[J4] Orosz, P., **Skopkó, T.**, Varga, M.: RCTP: A Low-complexity Transport Protocol for Collecting Measurement Data. Infocommun. J. 6 (3), 28-36, 2014. ISSN: 2061-2079.

8.2 Konferencia cikkek

[C1] Orosz, P., **Skopkó, T.**: Timestamp-resolution Problem of Traffic Analysis on High Speed Networks. In: International Conference on Applied Informatics (8)(2010.01.27-2010.01.30)(Eger)Proceedings of the 8th International Conference on Applied Informatics : January 27-30, 2010, Eger. Ed.: by Attila Egri-Nagy et al., Eszterházy K. College, Eger, 237-245, [2011]. ISBN: 9789639894723

[C2] Orosz, P., **Skopkó, T.**: Software-Based Packet Capturing with High Precision Timestamping for Linux.
In: The Fifth International Conference on Systems and Networks Communications : ICSNC 2010. [IEEE Computer Society], [Los Alamitos, CA], 381-386, 2010. ISBN: 9780769541457

[C3] Orosz P., **Skopkó T.**: Adatcsomagok nagyfelbontású időbélyegzése osztott erőforrású környezetben.
In: XI. ENELKO - XX SzámOkt Nemzetközi Energetikai-elektrotechnikai és Számítástechnika és Oktatási Konferencia : Szatmárnémeti, Románia, 2010.10.07-2010.10.10. [Erdélyi Magyar Műszaki Tudományos Társaság], Kolozsvár, 192-197, 2010.

[C4] **Skopkó, T.**, Orosz, P.: Investigating of the Precision of the TSC-based Packet timestamping. CJECE. 4 (1), 117-122, 2011. ISSN: 1844-9689.

[C5] Orosz, P., **Skopkó, T.**, Imrek, J.: Performance Evaluation of the Nanosecond Resolution Timestamping Feature of the Enhanced Libpcap. In: The Sixth International Conference on Systems and Networks Communications, ICSNC 2011, October 23-29, 2011 - Barcelona, Spain. [s.n.], [s.l.], 1-6, 2011.

[C6] Orosz, P., **Skopkó, T.**, Imrek, J.: A NetFPGA-based Network Monitoring System with Multi-layer Timestamping: Rnetprobe. In: Proceedings of the 15th International Telecommunications Network Strategy and Planning Symposium. [s.n.], [s.l.], 319-324, 2012.

[C7] Orosz, P., **Skopkó, T.**, Nagy, Z., Lukovics, T.: Performance Analysis of the Opus Codec in VoIP Environment Using QoE Evaluation. In: 8th International Conference on Systems and Networks Communications, ICSNC 2013, October 27-31, 2013, Venice, Italy : Proceeding. Ed.: Renzo Davoli, Josef Noll, [s.n.], [s.l.], 89-93, 2013. ISBN: 9781612083056

[C8] Orosz, P., **Skopkó, T.**: Multi-threaded Packet Timestamping for End-to-End QoS Evaluation.
In: 8th International Conference on Systems and Networks Communications, ICSNC 2013, October 27-31, 2013, Venice, Italy : Proceeding. Ed.: Renzo Davoli, Josef Noll, [s.n.], [s.l.], 94-99, 2013. ISBN: 9781612083056

[C9] Orosz, P., **Skopkó, T.**, Nagy, Z., Varga, P., Gyimóthi, L.: A Case-Study on Correlating Video QoS and QoE.
In: Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS). Ed.: Hanan Lutfiyya, Piotr Cholda, Institute of Electrical and Electronic Engineers, Red Hook, NY, 1-5, 2014. ISBN: 9781479909124 DOI: http://dx.doi.org/10.1109/NOMS.2014.6838399

9 Táblázatok listája

10 Ábrák listája

11 Irodalomjegyzék

[1] Marcio Nieblas Zapater and Graca Bressan, "A proposed approach for quality of experience assurance of IPTV," in *Digital Society, 2007. ICDS'07. First International Conference on the*, 2007, pp. 25-25.

[2] Markus Fiedler, Tobias Hossfeld, and Phuoc Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *Network, IEEE*, vol. 24, no. 2, pp. 36-41, 2010.

[3] Shekhar Borkar and Andrew A Chien, "The future of microprocessors," *Communications of the ACM*, vol. 54, no. 5, pp. 67-77, 2011.

[4] Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger, "Dark silicon and the end of multicore scaling," *IEEE Micro*, no. 3, pp. 122-134, 2012.

[5] Robert Schone, Daniel Hackenberg, and Daniel Molka, "Memory performance at reduced CPU clock speeds: an analysis of current x86_64 processors," in *Presented as part of the 2012 Workshop on Power-Aware Computing and Systems*, 2012.

[6] TCPDUMP/LIBPCAP public repository, [Online; accessed February-2016].

[7] WinPcap: The industry standard windows packet capture library, [Online; accessed February-2016].

[8] Wireshark: a free and open-source packet analyzer, [Online; accessed February-2016].

[9] PF_RING™ – High-speed packet capture, filtering and analysis, [Online; accessed February-2016].

[10] Luis Zabala, Alberto Pineda, Armando Ferro, and Daniel Fern, "Comparing Network Traffic Probes based on Commodity Hardware," *ICN 2014*, p. 272, 2014.

[11] Nicola Bonelli, Andrea Di Pietro, Stefano Giordano, and Gregorio Procissi, "On multi--gigabit packet capturing with multi--core commodity hardware," in *Passive and Active Measurement*, 2012, pp. 64-73.

[12] Jos et al., "High-performance network traffic processing systems using commodity hardware," *Data Traffic Monitoring and Analysis*, pp. 3-27, 2013.

[13] Luca Deri, Maurizio Martinelli, and Alfredo Cardigliano, "Realtime high-speed network traffic monitoring using ntopng," in *28th Large Installation System Administration Conference (LISA14)*, 2014, pp. 78-88.

[14] Paul Emmerich, Daniel Raumer, Florian Wohlfart, and Georg Carle, "Assessing soft-and hardware bottlenecks in PC-based packet forwarding systems," *ICN 2015*, p. 90, 2015.

[15] Endace: The Dag project, [Online; accessed February-2016].

[16] Amiel A Heyde and others, "Investigating the performance of Endace DAG monitoring hardware and Intel NICs in the context of Lawful Interception," *Centre for Advanced Internet Architectures (CAIA), Swinburne University of Technology, Tech. Rep. A*, vol. 80222, 2008.

[17] G Adam Covington, Glenn Gibb, John W Lockwood, and Nick Mckeown, "A packet generator on the NetFPGA platform," in *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines*, 2009, pp. 235-238.

[18] Zhiqiang Zhou, Lin Cong, Guohan Lu, Beixing Deng, and Xing Li, "Hats: high accuracy timestamping system based on netfpga," in *Advances in Computer Science and Information Technology*.: Springer, 2010, pp. 183-195.

[19] Victor Moreno, Pedro M Santiago Del R, Javier Ramos, Jaime J Garnica, and Jos, "Batch to the future: Analyzing timestamp accuracy of high-performance packet I/O engines," *Communications Letters, IEEE*, vol. 16, no. 11, pp. 1888-1891, 2012.

[20] Benjamin Villain, Michael H Davis, Julien Ridoux, Darryl Veitch, and Nicolas Normand, "Probing the latencies of software timestamping," in *Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2012 International IEEE Symposium on*, 2012, pp. 1-6.

[21] Irina Fedotova, Eduard Siemens, and Hao Hu, "A high-precision time handling library," *Journal of Communication and Computer*, vol. 10, pp. 1076-1086, 2013.

[22] DL Mills, "RFC 958: Network time protocol (NTP)," *Network Working Group*, 1985.

[23] Carsten Rieck, "An approach to primary NTP by using the linux kernel," in *Frequency Control Symposium, 2007 Joint with the 21st European Frequency and Time Forum. IEEE International*, 2007, pp. 873-876.

[24] Teodor Neagoe, Valentin Cristea, and Logica Banica, "NTP versus PTP in com puter networks clock synchronization," in *Industrial Electronics, 2006 IEEE International Symposium on*, vol. 1, 2006, pp. 317-362.

[25] Performance monitoring with the RDTSC instruction, [Online; accessed February-2016].

[26] Attila P and Darryl Veitch, "PC based precision timing without GPS," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, 2002, pp. 1-10.

[27] Tomohisa Uchida, "Hardware-based TCP processor for gigabit ethernet," *Nuclear Science, IEEE Transactions on*, vol. 55, no. 3, pp. 1631-1637, 2008.

[28] J Postel, "RFC 768: User Datagram Protocol (UDP)," *Request for Comments, IETF*, 1980.

[29] Lars-Ake Larzon, Mikael Degermark, Stephen Pink, Lars-Erik Jonsson, and Godred Fairhurst, "The lightweight user datagram protocol (UDP-Lite)," *Network Working Group, RFC*, vol. 3828, 2004.

[30] Tom Bova and Ted Krivoruchka, "Reliable UDP protocol," *draft-ietf-sigtran-reliable-udp-00. txt*, 1999.

[31] Yunhong Gu and Robert L Grossman, "UDT: UDP-based data transfer for high-speed wide area networks," *Computer Networks*, vol. 51, no. 7, pp. 1777-1799, 2007.

[32] Eddie Kohler, Mark Handley, and Sally Floyd, "Designing DCCP: Congestion control without reliability," in *ACM SIGCOMM Computer Communication Review*, vol. 36, 2006, pp. 27-38.

[33] Randall Stewart and Chris Metz, "SCTP: new transport protocol for TCP/IP," *Internet Computing, IEEE*, vol. 5, no. 6, pp. 64-69, 2001.

[34] R Pandit, "Reliable Datagram Sockets (RDS)," in *OpenIB Developers Workshop, Feb*, 2006.

[35] Young-Jin Kim, Vladimir Kolesnikov, Hongseok Kim, and Marina Thottan, "SSTP: a scalable and secure transport protocol for smart grid data collection," in *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, 2011, pp. 161-166.

[36] Young-Jin Kim and Marina Thottan, "SGTP: smart grid transport protocol for secure reliable delivery of periodic real time data," *Bell Labs Technical Journal*, vol. 16, no. 3, pp. 83-99, 2011.

[37] Ryousei Takano, Tomohiro Kudoh, Yuetsu Kodama, and Fumihiro Okazaki, "High-resolution timer-based packet pacing mechanism on the linux operating system," *IEICE transactions on communications*, vol. 94, no. 8, pp. 2199-2207, 2011.

[38] Cisco Visual Networking Index Cisco, "Global mobile data traffic forecast update, 2013--2018," *white paper*, 2014.

[39] Ericsson Mobility Report June 2015, [Online; accessed February-2016].

[40] Stylianos Karapantazis and Fotini-Niovi Pavlidou, "VoIP: A comprehensive survey on a promising technology," *Computer Networks*, vol. 53, no. 12, pp. 2050-2090, 2009.

[41] Alexander Raake, *Speech quality of VoIP: assessment and prediction*.: John Wiley \& Sons, 2007.

[42] Sim, Filipe Neves, Salviano Soares, Filipe Tavares, and Pedro Assun, "ArQoS\textregistered: System to monitor QoS/QoE in VoIP," in *EUROCON-International Conference on Computer as a Tool (EUROCON), 2011 IEEE*, 2011, pp. 1-2.

[43] Wael Cherif, Adlen Ksentini, Daniel N, and Mamadou Sidibe, "A_PSQA: PESQ-like non-intrusive tool for QoE prediction in VoIP services," in *Communications (ICC), 2012 IEEE International Conference on*, 2012, pp. 2124-2128.

[44] Fei Liu, Wei Xiang, Yueying Zhang, Kan Zheng, and Hui Zhao, "A novel QoE-based carrier scheduling scheme in LTE-Advanced networks with multi-service," in *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, 2012, pp. 1-5.

[45] Sofiene Jelassi, Gerardo Rubino, Hugh Melvin, Habib Youssef, and Guy Pujolle, "Quality of experience of VoIP service: a survey of assessment approaches and open issues," *Communications Surveys \& Tutorials, IEEE*, vol. 14, no. 2, pp. 491-513, 2012.

[46] Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Shobha Venkataraman, and He Yan, "Prometheus: toward quality-of-experience estimation for mobile apps from passive network measurements," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, 2014, p. 18.

[47] Chen-Chi Wu, Kuan-Ta Chen, Yu-Chun Chang, and Chin-Laung Lei, "Crowdsourcing multimedia QoE evaluation: A trusted framework," *Multimedia, IEEE Transactions on*, vol. 15, no. 5, pp. 1121-1137, 2013.

[48] John G Beerends et al., "Perceptual objective listening quality assessment (POLQA), the third generation ITU-T standard for end-to-end speech quality measurement part I—Temporal alignment," *Journal of the Audio Engineering Society*, vol. 61, no. 6, pp. 366-384, 2013.

[49] Yi Hu and Philipos C Loizou, "Evaluation of objective quality measures for speech enhancement," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 229-238, 2008.

[50] Andrew Hines and Naomi Harte, "Speech intelligibility prediction using a neurogram similarity index measure," *Speech Communication*, vol. 54, no. 2, pp. 306-320, 2012.

[51] Andrew Hines, Jan Skoglund, Anil Kokaram, and Naomi Harte, "ViSQOL: The virtual speech quality objective listener," in *Acoustic Signal Enhancement; Proceedings of IWAENC 2012; International Workshop on*, 2012, pp. 1-4.

[52] Andrew Hines, Jan Skoglund, Anil Kokaram, and Naomi Harte, "Robustness of speech quality metrics to background noise and network

degradations: Comparing ViSQOL, PESQ and POLQA," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 3697-3701.

[53] Andrew Hines et al., "ViSQOLAudio: An objective audio quality metric for low bitrate codecs," *The Journal of the Acoustical Society of America*, vol. 137, no. 6, pp. EL449--EL455, 2015.

[54] Mohammed Alreshoodi and John Woods, "Survey on QoE$\backslash$ QoS correlation models for multimedia services," *arXiv preprint arXiv:1306.0221*, 2013.

[55] Filipe Neves, Simao Cardeal, Salviano Soares, Pedro Assuncao, and Filipe Tavares, "Quality model for monitoring QoE in VoIP services," in *EUROCON-International Conference on Computer as a Tool (EUROCON), 2011 IEEE*, 2011, pp. 1-4.

[56] ITUT Rec, "G. 711: Pulse code modulation (PCM) of voice frequencies," *International Telecommunication Union, Geneva*, vol. 18, 1988.

[57] ITUT Rec, "G. 723.1,"," *Dual rate speech coder for multimedia communications transmitting at*, vol. 5, 1996.

[58] ITUT Rec, "G. 726, 40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)," *International Telecommunication Union, Geneva*, vol. 18, 1990.

[59] ITUT Rec, G. 728: Coding of speech at 16 kbit/s using low-delay code excited linear prediction, 1994.

[60] Juin-Hwey Chen and Richard V Cox, "The creation and evolution of 16 kbit/s LD-CELP: From concept to standard," *Speech Communication*, vol. 12, no. 2, pp. 103-111, 1993.

[61] ITU-T, G.728 Annex I: Frame or packet loss concealment for the LD-CELP decoder, 1999.

[62] ITUT Rec, "G. 729: Coding of speech at 8 kbit/s using conjugate structure algebraic-code-excited linear-prediction (CS-ACELP)," *Reduced complexity*, vol. 8, 1996.

[63] Redwan Salami, Claude Laflamme, Bruno Bessette, and J-P Adoul, "Description of ITU-T Recommendation G. 729 Annex A: reduced complexity 8 kbit/s CS-ACELP codec," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 2, 1997, pp. 775-778.

[64] Adit Benyassine et al., "ITU-T Recommendation G. 729 Annex B: a silence compression scheme for use with G. 729 optimized for V. 70 digital simultaneous voice and data applications," *Communications Magazine, IEEE*, vol. 35, no. 9, pp. 64-73, 1997.

[65] Erik Ekudden et al., "ITU-t g. 729 extension at 6.4 kbps.," in *ICSLP*, 1998.

[66] St, Redwan Salami, and Roch Lefebvre, "Design of test sequences for G. 729 Annex E," in *Speech Coding Proceedings, 1999 IEEE Workshop on*, 1999, pp. 120-122.

[67] ETS ETSI, "300 961 (GSM 06.10)," *Full rate speech transcoding*.

[68] ETS ETSI, "300 969 (GSM 06.20)," *Half rate speech transcoding*.

[69] ETS ETSI, "300 726 (GSM 06.60)," *Enhanced Full Rate (EFR) speech transcoding*.

[70] Tian Wang, Kazuhito Koishida, Vladimir Cuperman, Allen Gersho, and John S Collura, "A 1200/2400 bps coding suite based on MELP," in *Speech Coding, 2002, IEEE Workshop Proceedings.*, 2002, pp. 90-92.

[71] AMR speech Codec, General Description (Release 9), 3GPP Technical Specification 3GPP TS 26.071 V9. 0.0, 2009.

[72] S Andersen, A Duric, H Astrom, R Hagen, and W Kleijn, "J. Linden," Internet Low Bit Rate Codec (iLBC)," RFC 3951, December, Tech. rep. 2004.

[73] Paul Mermelstein, "G. 722: a new CCITT coding standard for digital transmission of wideband audio signals," *Communications Magazine, IEEE*, vol. 26, no. 1, pp. 8-15, 1988.

[74] ITUT Rec, "G. 722.1, Low-Complexity Coding at 24 and 32 kbit/s for Hands-Free Operation in Systems With Low Frame Loss," *International Telecommunication Union, Geneva, Switzerland*, vol. 18, 2005.

[75] Bruno Bessette et al., "The adaptive multirate wideband speech codec (AMR-WB)," *Speech and Audio Processing, IEEE Transactions on*, vol. 10, no. 8, pp. 620-636, 2002.

[76] Jari M et al., "AMR-WB+: a new audio coding standard for 3rd generation mobile audio services," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, vol. 2, 2005, pp. ii--1109.

[77] Stephane Ragot et al., "ITU-T G. 729.1: An 8-32 kbit/s scalable coder interoperable with G. 729 for wideband telephony and Voice over IP," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, 2007, pp. IV--529.

[78] Jean-Marc Valin, "Speex: a free codec for free speech," in *Proceedings of the Australian National Linux Conference*, 2006.

[79] Juin-Hwey Chen and Jes Thyssen, "The broadvoice speech coding algorithm," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, 2007, pp. IV--537.

[80] Raimund Schatz, Sebastian Egger, and Alexander Platzer, "Poor, good enough or even better? Bridging the gap between acceptability and QoE of mobile broadband data services," in *Communications (ICC), 2011 IEEE International Conference on*, 2011, pp. 1-6.

[81] Anssi R and Henri Toukomaa, "Voice Quality Characterization of IETF Opus Codec.," in *INTERSPEECH*, 2011, pp. 2541-2544.

[82] Christian Hoene, Jean-Marc Valin, Koen Vos, and Jan Skoglund, "Summary of Opus listening test results," 2013.

[83] Jean-Marc Valin, Gregory Maxwell, Timothy B Terriberry, and Koen Vos, "High-quality, low-delay music coding in the opus codec," in *Audio Engineering Society Convention 135*, 2013.

[84] Van Jacobson, Ron Frederick, Steve Casner, and H Schulzrinne, "RTP: A transport protocol for real-time applications," 2003.

[85] JACK Audio Connaction Kit, [Online; accessed February-2016].

[86] GStreamer: a library for constructing graphs of media-handling components, [Online; accessed February-2016].

[87] Stephen Hemminger and others, "Network emulation with NetEm," in *Linux conf au*, 2005, pp. 18-23.

[88] Ring (formerly sflPhone): a secure and distributed voice, video and chat communication platform, [Online; accessed February-2016].

[89] Julian Spittka and Koen Vos, "RTP Payload Format for the Opus Speech and Audio Codec," *RFC 7587*, 2015.

[90] Antony W Rix, John G Beerends, Doh-Suk Kim, Peter Kroon, and Oded Ghitza, "Objective Assessment of Speech and Audio Quality\&\# 8212; Technology and Applications," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 6, pp. 1890-1901, 2006.

[91] J-H Chen, W Lee, and J Thyssen, "RTP payload format for BroadVoice speech codecs," Tech. rep. 2005.

[92] Jes Thyssen, Robert Zopf, Juin-Hwey Chen, and Nikhil Shetty, "A Candidate for the ITU-T G. 722 packet loss concealment standard," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, 2007, pp. IV--549.