Önálló laboratórium beszámoló

Távközlési és Médiainformatikai Tanszék

készítette: **Schweitzer András Attila**
schweitzeraa16@gmail.com
neptun-kód: **TLEIB5**
ágazat: **Intelligens hálózatok**
konzulens: **Németh Felicián**
nemethf@tmit.bme.hu
konzulens: **Lévai Tamás**
levait@tmit.bme.hu

Többutas adatátvitel Media over QUIC rendszerben (Media over Multipath QUIC)

Feladat:
A Media over QUIC (MoQ) egy új, még fejlesztés alatt álló protokollcsalád, amelyet internetes médiaadatok hatékony továbbítására terveztek. Ennek egyik meglévő megvalósítása a LibQuicR nevű könyvtár, amely jelen feladat alapját képezi.

A feladat célja a LibQuicR módosítása úgy, hogy képes legyen kihasználni a több útvonalon történő párhuzamos adatátvitelt (multipath), amit a PicoQUIC nevű, QUIC protokollt megvalósító könyvtár biztosít. Ennek érdekében a LibQuicR transzport rétegét úgy kell átalakítani, hogy egyetlen kapcsolat keretében több hálózati útvonalat is tudjon létrehozni és kezelni a PicoQUIC multipath képességein keresztül.

A módosítások működését egy demonstrációval igazoljuk, amelyet egy virtuális tesztkörnyezetben hajtunk végre. Ebben a környezetben több hálózati interfésszel rendelkező kliensek vesznek részt, és az adatátvitel során aktívan kihasználjuk a multipath lehetőségeket. A teszt során szándékosan megszakítunk egy éppen használt útvonalat, hogy bemutassuk: a LibQuicR – az új multipath támogatással – képes automatikusan fenntartani az adatfolyam továbbítását, minimális késleltetéssel és megszakítás nélkül.

Tanév: 2024/25. tanév, II. félév

1. A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

1.1. Bevezető

Napjainkban az internetes adatforgalom túlnyomó részét különféle médiatartalmak – elsősorban videók és élő közvetítések – teszik ki. Egyes kutatások szerint az internetes forgalom 65%-a ilyen típusú tartalom lehet [1]. Ezek hatékony és megbízható továbbítása azonban komoly technológiai kihívást jelent, különösen akkor, ha a felhasználói élményt is figyelembe vesszük: az alacsony késleltetés, a megszakításmentes lejátszás és a biztonságos adatátvitel alapvető elvárások.

E kihívásokra jött létre válaszként a QUIC protokoll[2], amely UDP-alapú, alacsony késleltetésű, titkosított és megbízható adatátvitelt tesz lehetővé. Erre épülve fejlődik jelenleg a Media over QUIC (MoQ) szabvány[5], amely kifejezetten médiatartalmak hatékony továbbítását célozza a QUIC képességeit kihasználva.

Ezzel párhuzamosan a felhasználói eszközök – különösen a mobiltelefonok és laptopok – egyre gyakrabban rendelkeznek több hálózati interfésszel (például Wi-Fi és 5G). Mégis gyakran tapasztalható kapcsolat megszakadás, hosszú töltési idők vagy éppenséggel elérhetetlenné váló szolgáltatások, ha az egyik kapcsolat megszakad vagy instabillá válik. A többutas adatátvitel lehetőséget kínál ezen problémák áthidalására azáltal, hogy párhuzamosan több hálózati útvonalat használ az adatok továbbítására, növelve ezzel a robusztusságot és az elérhető sávszélességet.

A jelen munka célja ezen két technológia – egy adott MoQ implementáció és a multipath kommunikáció – egyesítése. A cél egy olyan rendszer bemutatása, amely képes több útvonalat kihasználva, élő adatfolyamot hatékonyan és megbízhatóan továbbítani. Ez nemcsak a hálózati erőforrások jobb kihasználását segíti elő, hanem hozzájárul a végfelhasználói élmény javításához is.

1.2. Elméleti összefoglaló

1.2.1. A QUIC protokoll

Az internetes multimédia átviteli technológiai fejlődése során egyre nagyobb igény mutatkozik olyan protokollokra, amelyek képesek rugalmasan és hatékonyan kezelni a valós idejű adatfolyamokat, még változékony és megbízhatatlan hálózati körülmények között is. A QUIC protokoll[2] – mint UDP-alapú, titkosított és kapcsolatorientált transzport protokoll – alapjaiban újraértelmezi a hálózati kommunikáció lehetőségeit, különösen olyan kiterjesztésekkel, mint a *multipath* adatátvitel [3]. A multipath képesség lehetővé teszi több párhuzamos hálózati útvonal (**path**) egyidejű vagy váltott használatát egyetlen logikai kapcsolat keretén belül, ami különösen hasznos mobil eszközök, redundáns hálózatok vagy peremhálózati környezetek esetén.

A QUIC protokoll és a HTTP/3 működésének vizsgálatára fejlesztették ki hozzá a qlog [8] formátumot, amelynek célja, hogy strukturált és gépileg olvasható formátumban rögzítse a QUIC kapcsolatok eseményeit és metrikáit. Ez egy JSON-t generál a kapcsolat bármely végpontjában ami tartalmaz minden olyan adatot, amely a kapcsolat során generálódik a QUIC rétegben beleértve a csomagok küldését és fogadását, a kapcsolat állapotát, a késleltetéseket és kommunikáció során felmerülő, protokoll számára felismerhető hibákat is.

1.2.2. QUIC multipath alapjai

A többutas adatátvitel technikai alapja, hogy a QUIC kapcsolat során nemcsak egy, hanem több **path** hozható létre, amelyeket a transzport réteg párhuzamosan vagy dinamikusan képes használni. Egy *path* definíció szerint egy adott forrás-cél IP-cím és port kombináció, azaz különböző fizikai vagy logikai hálózati kapcsolatok reprezentálása. A projektben használt PicoQUIC könyvtár egy QUIC megvalósítás, amelynek fejlesztés alatt álló multipath támogatása lehetővé teszi, hogy egy kapcsolat során több ilyen útvonal aktív legyen, és az alkalmazás (QUIC-et megvalósító rétegében egy belső algoritmus) meghatározza, hogyan ossza meg az adatforgalmat ezek között [4].

PicoQUIC esetében a multipath kezelés alapvetően decentralizált és eseményvezérelt. A kapcsolat felépítése során egy adott interfészen elindított kapcsolat kiegészíthető további **PATH CHALLENGE / RES-**

PONSE üzeneteken keresztül feltérképezett útvonalakkal. Ha egy új útvonal válik elérhetővé (például egy új IP-cím vagy interfész aktiválódik), a rendszer felismeri azt, és lehetőséget biztosít az adatküldés ezen az útvonalon történő elindítására. Az útvonalak állapotát folyamatosan figyeli a protokoll (RTT, veszteség, állapotváltozás), így lehetővé válik az útvonalak közötti dinamikus váltás vagy forgalommegosztás a kapcsolatok megszakítása nélkül. Ez különösen fontos valós-idejű alkalmazásokban, mivel lehetővé teszi a megszakítás nélküli adattovábbítást, hálózati hiba vagy mobilitás esetén is.

1.2.3. A Media over QUIC (MoQ) protokoll

A Media over QUIC (MoQ) egy új, még fejlesztés alatt álló protokoll [5], amely a QUIC nyújtotta lehetőségekre építve biztosít alacsony késleltetésű, valós idejű médiatovábbítást. A MoQ rendszerében a szerepkörök három alapvető típus köré szerveződnek: a **Publisher** (szolgáltató) az, aki a médiatartalmat (például videó vagy hangfolyam) létrehozza és továbbításra bocsátja, a **Subscriber** (fogyasztó) a végponti fogyasztó, aki ezt a tartalmat fogadja és feldolgozza/lejátsza, míg a **Relay** (továbbító) köztes szereplőként funkcionál, amely a számára elérhető tartalmat hirdeti és továbbítja más résztvevők felé, jellemzően a késleltetés, terheléselosztás és elérhetőség optimalizálása érdekében. Ezek a szereplők struktúrált MoQ adatfolyamokon keresztül kommunikálnak egymással, amelyek sávokból (*track*) azon belül pedig objektumokból (*object*) épülnek fel, lehetővé téve a médiatartalom rugalmas azonosítását és replikációját.

A kidolgozás alatt álló szabvány szerint egy MoQ-alapú rendszerben a relay képes egyszerre több publisher és subscriber felé is kapcsolatot fenntartani, és akár multicast-szerű módon továbbítani a médiatartalmat. A hálózati topológia e szerepkörök között rengetegféle lehet, mivel a relayek közötti kapcsolat is támogatott, ami lehetővé teszi a tartalom replikációját és elosztását a különböző relayek között. Ezáltal a gazdag struktúrabeli opciók lehetőséget biztosítanak a tartalom dinamikus és optimalizált terjesztésére, azonban egyben igényli azt is, hogy a transzport réteg rugalmasan tudjon alkalmazkodni a változó hálózati viszonyokhoz – például egy útvonal meghibásodásához, vagy új alternatív útvonal megjelenéséhez.

1.2.4. Multipath célja a MoQ-ban

A multipath működés bevezetése ebbe az architektúrába jelentős előnyt nyújthat, különösen olyan eszközök esetén, amelyek egyszerre több hálózati interfésszel rendelkeznek (például Wi-Fi és mobil adatkapcsolat szimultán használata). A relay-ek és végpontok (subscriber, publisher) több interfésszel rendelkező környezetben történő működése során a multipath támogatás nemcsak a redundanciát növeli, hanem lehetőséget biztosít egyfajta **hálózati adaptivitásra is**, amely révén például a relay automatikusan kiválaszthatja a legjobb elérhető útvonalat egy adott irányba. Ez az architektúra ideális alapot teremt olyan rendszerek számára, ahol fontos a magas rendelkezésre állás, az alacsony késleltetés és az automatikus hibatűrés.

A MoQ protokoll jelenlegi állapota még fejlesztés alatt áll, és jelenleg egyik implementáció sem támogat transzportprotokoll szintű multipath képességeket.

Bár más MoQ megvalósítások is léteznek, azért a LibQuicR[7] könyvtárra esett a választás, mert a QUIC protokollt a PicoQUIC könyvtár segítségével implementálja, amely jelenleg is folyamatosan bővül, és multipath támogatása is követi a legújabb szabványosítási irányokat.

1.3. A munka állapota, készültségi foka a félév elején

A félév kezdetén sem a QUIC protokoll, sem a LibQuicR könyvtár kapcsán nem rendelkeztem korábbi gyakorlati tapasztalattal, és mivel a választott tématerület is viszonylag újnak számít, a munka elindításához nem állt rendelkezésre közvetlenül felhasználható előzetes alap.

2. Az elvégzett munka és az eredmények ismertetése

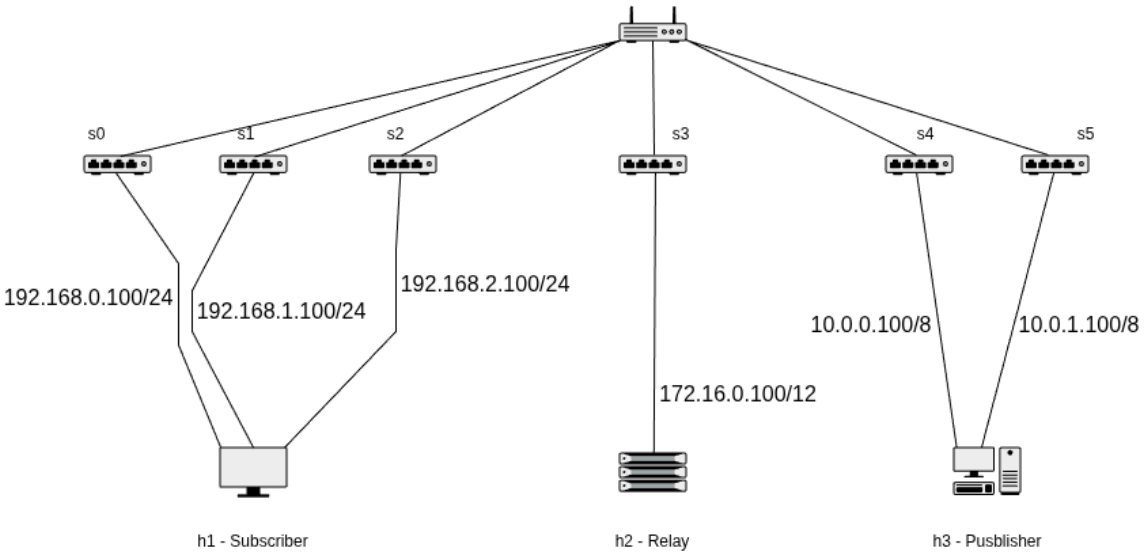
2.1. A fejlesztés és elemzés lépései egy multipath demonstrációhoz

2.1.1. Mininet emulációs környezet

Első sorban szükséges volt egy hálózat virtualizálására alkalmas környezet, amely lehetővé teszi a különböző hálózati topológiák és viszonyok emulálását, hiszen csak egy kontrollált környezetben lehet érdemben tesztelni a multipath működést. E célból a Mininet nevű eszközre esett a választás, mivel a használatához szükséges tudás elsajátítása nem igényelt sok időt.

A Mininet segítségével könnyen létrehozhatók virtuális gépek (továbbiakban hostok vagy h1/h2/h..), amelyek a futtató gépen létrehozott hálózati névterek, amelyeken virtuális hálózati interfészeket és azokhoz hálózati konfigurációt rendelhetünk. Ezekben a névterekben lehet futtatni a tesztelni és megfigyelni kívánt programokat. A Mininet emellett lehetővé teszi a hálózati topológia emuláció közbeni megváltoztatását is, így például kábelszakadás is könnyen emulálható vele. [6]

Az 1. ábrán látható topológiát hoztam létre a Mininet segítségével, amellyel a multipath működését teszteltem:



1. ábra. A Mininetben létrehozott topológia logikai ábrázolása

A h1, h2, h3 jelölik a hostokat, s0-s5 jelölik a switch-eket, és r0 jelöli a routert. h1 három interfésszel rendelkezik, egyenként csatlakozik a s0, s1 és s2 switch-ekhez, azok pedig a r0 routerhez. Hasonlóan h2 is, csak egy interfésszel, a h3 pedig kettővel csatlakozik.

A hostok és a router között azért van szükség switch-ekre, mert a kapcsolatok megszakítása emuláció közben problémát okoz, ha a megváltoztatott kapcsolat közvetlenül a host-okhoz csatlakozik, mivel az alapértelmezett átjárók alhálózatonként eltérnek, így ha egy közvetlen linket megszakítunk (amely külön alhálózaton van a többi interfészhez képest), akkor az alapértelmezett átjáró információja is elveszik, és a link újraépítése után nem tartozik hozzá alapértelmezett átjáró, tehát nem tesztelhető az az eset, amikor a link megszakítása után azt visszaállítjuk. Így az alábbi topológián egy switch és a router közötti kapcsolat megszakításával könnyen lehet emulálni a kapcsolat megszakadását az alapértelmezett átjáró eltűnési problémája nélkül.

IP-alapú címzés szempontjából a hostok eltérő alhálózatokban vannak, (a h1-nek még emellett mindhárom interfésze eltérő alhálózatban van, hogy az alapértelmezett átjáró eltérő lehessen, ezzel pontosabban emulálva a valós környezetet). Mindez azért is fontos, mert bár a MoQ egy alkalmazásrétegbeli (Layer 7) protokoll, a működéséhez elengedhetetlen a hálózati réteg (Layer 3) megfelelő működése is. A multipath viselkedés ugyanis IP-szinten – azaz hálózati interfészekhez és címekhez rendelt útvonalak mentén – valósul meg, így a tesztelés során szükséges az alsóbb rétegek kontrollált kezelése is.

A topológia kialakítása során cél volt, hogy a multipath továbbítás minden releváns esete tesztelhető

legyen. Például a h1 három interfésszel rendelkezik, hogy biztosítva legyen az, hogy nem csak két interfészen működik a multipath, azért van h3-nak is két interfésze, hogy meg lehessen figyelni hogy viselkedik h2 egynél több multipath kapcsolat esetén, vagy éppen a router és az alhálózatok is elhagyhatóak a működés szempontjából, azonban valós környezetben ezek kikerülhetetlenek.

A munkát és a tesztelést felgyorsítva mindezt egy Python szkript segítségével automatizáltam, amely a Mininet API-ját használja a virtuális gépek és kapcsolatok létrehozására, konfigurálására, kezelésére, beleértve a hostokon futtatott demonstrációs programokat és a Wireshark csomagelkapást is. Tehát egy script futtatásával pillanatok alatt megfigyelhető a kívánt program működése.

2.1.2. A PicoQUIC programkönyvtár

A fejlesztéshez elengedhetetlen a PicoQUIC alapvető ismerete, amihez megfelelő alapot képez a picoquicdemo program. A picoquicdemo egy egyszerű példa a PicoQUIC használatára, amely bemutatja a PicoQUIC könyvtár alapvető funkcióit és lehetőségeit, beleértve a kapcsolat felépítését, adatátvitelt és a hibakezelést, illetve multipath funkcionalitást is kínál.

Maga a pcioquicdemo felépítése elég egyszerű, olyan értelemben hogy ez egy darab C fájl, amely a PicoQUIC függvényeit használja, és a szerver, valamint a kliensek funkcióit is ellátja egyaránt.

Sajnos pont emiatt az átláthatóságot nem feltétlen egyszerűsíti meg, de összességében kiderült, hogy két dolog szükséges a multipath működéshez.

Egyrészt a kezdeti multipath negotiation (egyeztetés) – ami arra szolgál, hogy a két fél közötti első kapcsolat felépítésekor megegyezzenek az összes többi transzport paraméter mellett abban is, hogy mind a két fél támogatja-e a többutas kapcsolatot; másrészt az "extra" útvonalak tényleges kiépítése.

Ezek közül az utóbbit megvalósító kódrészletek könnyen elkülöníthetőek a kód többi részétől, mivel ezen a téren a program sok fajta opciót biztosít, amelyek közül a legegyszerűbb eshetőség az alapszintű multipath, amikor több interfész használatával több útvonalat alakítunk ki, amelyeket a PicoQUIC könyvtár automatikusan kezel.

Jelenlegi munka szempontjából ez az egyszerű multipath a leginkább releváns, mivel amíg az nem működik megfelelően, addig a komplexebb multipath forgatókönyvek nem is igazán tesztelhetők/használhatók. Az ezt ellátó kódrészlet a picoquicdemo-ban jól elkülönül, így jó alapot biztosított a LibQuicR könyvtárbeli implementációhoz.

2.1.3. Egyéb következtetések a PicoQUIC kapcsán

A Wireshark használata során érdekes kérdés merült fel a multipath PicoQUIC implementálásával kapcsolatban, mégpedig, hogy nem feltétlenül kezeli a kapcsolat útvonalait az IETF QUIC Multipath draftja szerint, mivel adott útvonal kapcsán útvonal szakadás esetén nem láttunk PATH ABANDON üzenetet, ami arra szolgálna, hogy egy már nem működő útvonalat inaktívvá nyilvánítson mind a két fél számára. Azonban olyan érdekes jelenség is előfordult, hogy A és B útvonal esetében A útvonal megszakítása után B útvonalon ment tovább az adatátvitel, de ha A útvonal újra aktívvá vált, akkor nem használta újra azt, és ezután B útvonal megszakítása után sem váltott vissza A útvonalra. Ami ezt még inkább érdekessé teszi, az az, hogy B útvonalat újra aktiválva az adatátvitel folytatódott. Tehát a PicoQUIC multipath implementációja nem feltétlenül kezel minden esetet a draft szerint, de ez nem feltétlenül jelent problémát a jelenlegi fejlesztési szakaszban, mivel a protokoll még nem végleges, és ez a viselkedés egyben kijelölhet egy lehetséges irányt a projekt további bővítésére vagy finomhangolására.

2.1.4. Működés elemzésének módjai

A PicoQUIC rengeteg funkciót és metrikát biztosít a működés elemzésére, első sorban a qlog fájlok generálása lenne a bevett módszer, és a protokoll alapszintű működésének megértése szempontjából valóban hasznos a qlog, de nem minden esetben elegendő, mivel a QUIC protokoll sokféle eseményt és metrikát generál, amelyek közül sok nem feltétlenül releváns a multipath működés szempontjából és maga a multipath is megnehezíti mivel a qlog események elemézésére használatos qvis program[9] – ami egy webes alkalmazás, a qlog fájlok vizualizálására – nem képes könnyen és átláthatóan megjeleníteni a multipath eseményeket, mivel azok könnyen eltorzítják a két oldal kapcsolatának szinkronizációját, így ezzel nem lehet megfelelően elemezni a kapcsolatokat.

A qlog fájlok elemzésénél sokkal hasznosabb a Wireshark program, mivel a QUIC protokollt is támogatja. Azonban ez olyan kihívást állít a qlog-gal szemben, hogy míg a qlog fájlokat a program maga generálja, tehát a titkosítást ki tudja kerülni, addig a Wireshark magukat a hálózati interfészen megjelenő csomagokat rögzíti, így a QUIC csomagok titkosítása megnehezítette a folyamatot, főleg a multipath miatt, mivel a másodlagos útvonalon küldött és fogadott csomagokat a Wireshark nem tudja hozzákötni automatikusan az első útvonal interfészén létrejött kapcsolat titkosított csomagjaihoz.

A PicoQUIC beépítetten támogatja a SSL keylog fájlok generálását a "SSLKEYLOGFILE" környezeti változó beállításával, ami segítségével fel lehet oldani a QUIC által használt TLS titkosítást, így a Wireshark képes dekódolni a csomagokat, és megjeleníteni azok tartalmát (legalábbis a kapcsolat kezeléséért felelős QUIC réteg szempontjából).

Ez a megoldás lehetővé teszi, hogy a Wireshark segítségével generált pcapng fájlokat átkonvertáljuk dekódolt formátumra az alábbi paranccsal:

```
editcap --inject-secrets tls,<keylog_file> <input_pcap_file>
  <output_pcap_file>
```

Ez a parancs a Wireshark által generált TLS titkosítást használó QUIC csomagokat tartalmazó pcapng fájlokat dekódolja, a keylog fájl segítségével, így azok bármilyen kontextusban, a keylog fájl nélkül megtekinthetőek és elemezhetőek.

2.1.5. LibQuicR felépítése és átalakítása

A LibQuicR könyvtár egy összetettebb kódbázisú program mint a picoquicdemo, mivel amíg az utóbbi egy darab C fájl, ami direkt használja a PicoQUIC könyvtárat demonstrációs célokra, addig a LibQuicR sokkal komplexebb struktúrával rendelkezik, amelyben a PicoQUIC könyvtár sokkal kevésbé közvetlenül van jelen. Ennek ellenére pont azért, mert egy jelentősen nagyobb struktúráról van szó, szükségszerűen sokkal jobban elkülönülnek a különböző funkciók, így a PicoQUIC API-ját használó kódrészletek is.

A multipath működés támogatásához három fő funkció került implementálásra a LibQuicR könyvtárban. Egyrészt bővítésre került a parancssori felhasználói felület (CLI), amely új, multipath-specifikus argumentumokat fogad. Másrészt megvalósításra került a multipath negotiation folyamata, valamint a másodlagos hálózati útvonalak kiépítésének logikája is.

Ebből a második és harmadik megegyezik a korábban PicoQUIC-nél említettekkel, mivel ezek elengedhetetlenek a multipath működéshez, de ahhoz, hogy ezt érdemben lehessen használni is, szükséges volt a CLI argumentumok kiegészítése is. Ezt a picoquicdemo-ban használatos argumentumokhoz hasonlóan valósítottam meg, mivel ha már PicoQUIC-re alapoz a LibQuicR, és maga a fejlesztésem pedig annak a multipath demonstrálására szolgáló demó programjára alapoz, akkor a CLI argumentumoknak hasonló megválasztása egy logikus lépés.

Relay esetén ez a CLI argumentumok kiegészítése csak egy extra argumentumot jelentett:

```
-m, --multipath
```

Ez egy szimpla kapcsoló (nem fogad extra paramétert), amely oly módon engedélyezi a multipath támogatást, hogy amikor létrejön a relay objektum QUIC transzportért felelős része, akkor azt úgy inicializálja, hogy egy bejövő kapcsolat esetén a transzport paraméterek egyeztetésénél a saját paraméterei között szerepeljen a multipath támogatás is.

Végponti kapcsolat esetén ez két kiegészítést jelentett:

- `-m, -multipath`
- `-a „...", -alt_ifaces „..."`

Itt természetesen a multipath kapcsoló ugyanúgy működik mint a relay esetén, de emellett szükséges volt egy új argumentum is, amely lehetővé teszi a felhasználó számára, hogy megadja a másodlagos interfészeket, amelyeken a multipath harmadik szükséges elemét, a másodlagos útvonalakat kiépíti. Ehhez szüksége van paraméterekre, amelyek a másodlagos interfészeket tartalmazzák, itt is a picoquicdemo-val azonos formátumot használtam, tehát a következő formátumot várja:

```
-a "<IP-cím>/<interfész száma>,<IP-cím>/<interfész száma>,..."
```

Például a h1 host esetén ez az alábbi módon néz ki: "192.168.1.100/3,192.168.2.100/4". Látható, hogy az alapértelmezett IP-címre nincs szüksége, mert azon már elkezdődött a kapcsolat, és csak azokat az interfészeket kell megadni, amelyeken a másodlagos útvonalakat kiépíti.

Az eltérés a relay és a végponti kapcsolat között abban rejlik, hogy míg a relay fogadja a bejövő kapcsolatokat, addig a végponti kapcsolat esetén önállóan, a kapcsolat létrejötte után küld PATH CHALLENGE üzeneteket a másodlagos interfészeken, hogy azok is elérhetővé váljanak a kapcsolat számára (amennyiben a multipath negotiation sikeresen megtörtént). Relay esetén több interfész használatát nem implementálja jelenleg a munkám.

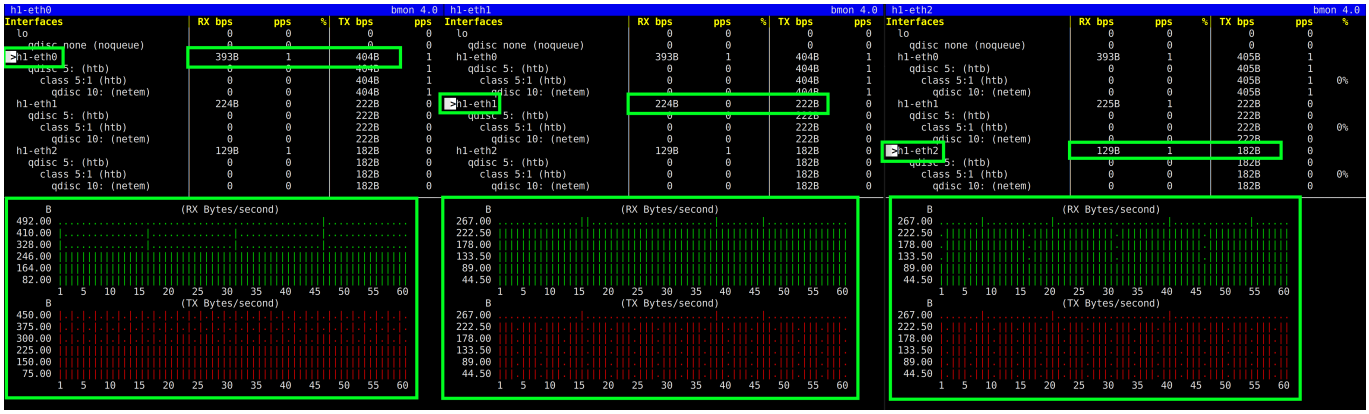
A multipath negotiation megvalósulásához ki kellett egészíteni a LibQuicR-ben használt adatstruktúrákat, hogy eljussanak a multipath és az interfész információk arra a szintre, ahol a PicoQUIC könyvtár API-ját használja a program, valamint a PicoQUIC transzport objektum konstruktorát kellett kiegészíteni, és a transzport paraméterek inicializálásakor a multipath paraméter mellett az `initial_max_path_id` változót kellett beállítani, hogy kapcsolat létrehozása után tudjon kezelni több útvonalat is.

A többi útvonal kiépítése gyakorlatilag megegyező módon működik, mint a picoquicdemo esetén. A PicoQUIC `packet_loop_v3()` függvénye minden egyes csomag feldolgozása után meghív egy callback függvényt, amely a LibQuicR könyvtárban van implementálva, és ebben történik meg – ha minden körülmény adott, tehát kliensről van szó, multipath engedélyezett, és addig a pontig még nem volt új útvonal próbálva – hogy a másodlagos interfészeken PATH CHALLENGE üzeneteket küldjön. Ebben a callback függvényben történik egy segédfüggvény segítségével a másodlagos interfészeket tartalmazó sztring feldolgozása is, amely logikailag ugyancsak azonos módon működik mint a picoquicdemo esetén.

2.1.6. Funkcionális tesztelés

Módosításaim eredményeként a bemutatott környezetben működik a többutas adatávitel, amely a másodlagos interfészeken keresztül adatokat küldeni és fogadni, továbbá valamely útvonal megszakadása esetén minimális késleltetéssel képes a másodlagos interfészen folytatni az adatátvitelt. Maga az adatátvitel időbélyegek másodpercenkénti küldését jelenti, ami egy beépített módja a LibQuicR demó programjának.

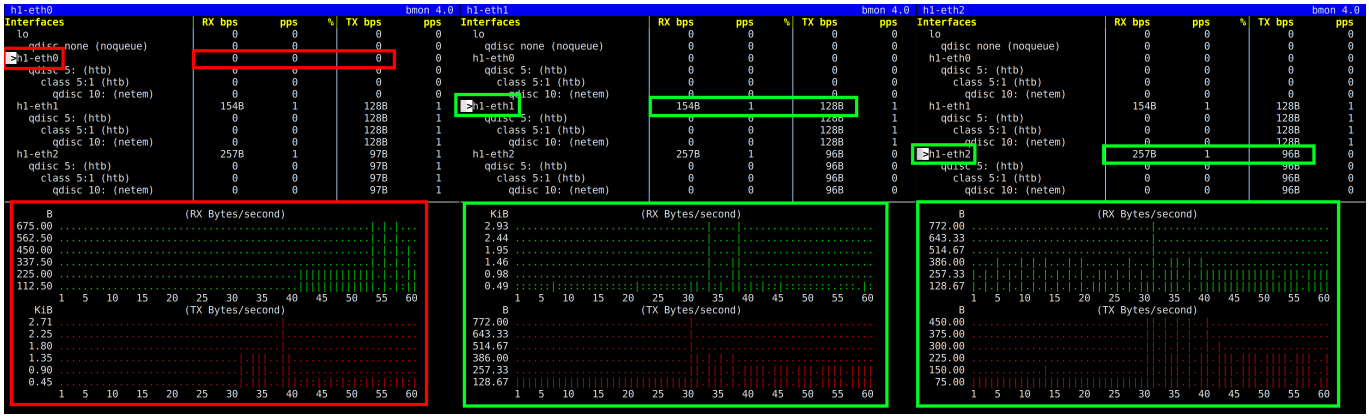
A következő ábrák a bmon program kimenetét mutatják a h1 hoston, amely subscriber üzemmódban fut és három interfészen, interfészenként külön alhálózaton kommunikál a h2 hosttal egy rotueren keresztül, amely relay-ként üzemel.



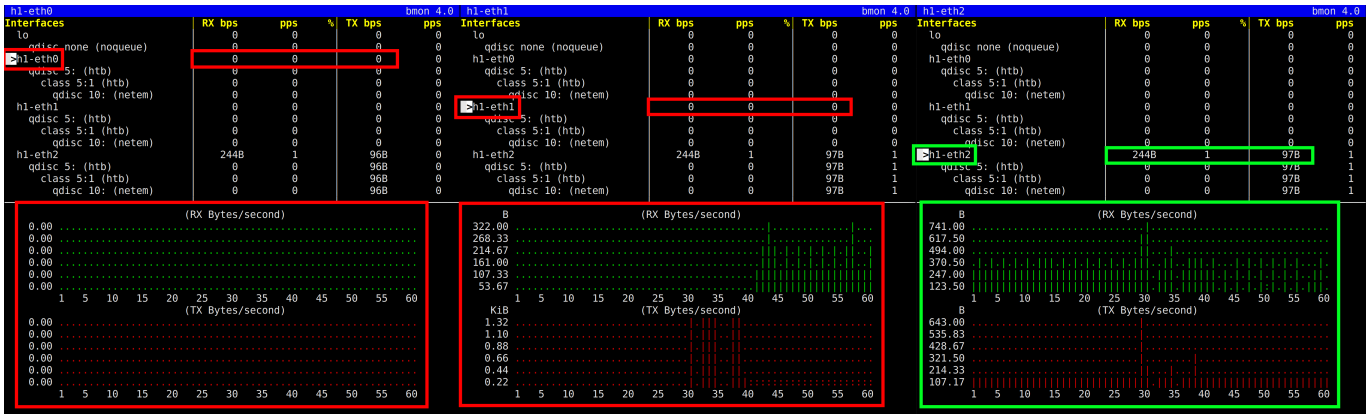
2. ábra. Interfész aktivitás 3 aktív útvonallal, sorrendben 0, 1, és 2-es indexű interfészek

2. ábrán látható, hogy 3 aktív interfészen keresztül van folyamatos adatátvitel, majd a 3. ábrán már látható, hogy a nullás indexű interfészen nem folyik adat, mivel itt az ahhoz kapcsolódó switch és a router közötti kapcsolatot megszakítottam. Ezért a másodlagos interfészeken keresztül folytatódik az adatátvitel. Ezek nem csak azt bizonyítják, hogy a multipath implementáció kezeli a kapcsolatszakadást, de azt is, hogy akár 3 interfészt is tud hasznosítani.

Majd pedig a 4. ábrán látható, hogy az egyes indexű interfészen is leállt az adatátvitel (ugyancsak kapcsolatszakadás miatt), de a kettes indexű interfészen továbbra is fennmarad az adatátvitelt.



3. ábra. Interfész aktivitás 2 aktív útvonallal, sorrendben 0, 1, és 2-es indexű interfészek



4. ábra. Interfész aktivitás 1 aktív útvonallal, sorrendben 0, 1, és 2-es indexű interfészek

Fontos megjegyezni, hogy a relay minimális kódmódosítással képes kezelni valamennyi releváns kapcsolódási szituációt. Ennek megfelelően működőképes akkor is, ha a subscriber és a publisher egyaránt egyetlen hálózati interfészen keresztül csatlakozik, ha csak az egyik fél – például a subscriber vagy a publisher – rendelkezik több interfésszel, illetve abban az esetben is, ha mindkét végpont több interfészen keresztül kapcsolódik a hálózathoz.

Illetve érdekes azt is megjegyezni, hogy a korábban említett `initial_max_path_id` változó a kódban 2-re van állítva, ha multipath környezetben indul el a program, de ez nem korlátozza le kapcsolatonként két interfész használatára sem a relayt, sem pedig a subscribert vagy a publishert, mint látható a 2. ábrán, ahol 3 interfész van aktívan használva.

2.2. Összefoglalás

2.2.1. Féléves feladat összefoglalása

Féléves feladatom a QUIC feletti médiaátvitelt megvalósító LibQuicR könyvtárban egy kezdetleges multipath támogatás implementálása volt, valamint ehhez kapcsolódóan egy demonstrációs környezet kialakítása és a megvalósítás működésének értékelése. A munka során az alábbi eredményeket értem el:

- A LibQuicR könyvtárban a multipath működést kezdetlegesen implementáltam, amely megalapozhatja a jövőbeni, fejlettebb multipath támogatást a projektben. Implementációm GitHub repositoryja: github.com/Schweitzee/libquicr (multipath branch)
- A fejlesztést dokumentáltam, a megvalósítást letisztáztam és előkészítettem GitHub-on történő közzétételre pull request formájában az eredeti projekthez.

- SSL keylog fájl generálásának lehetőségét beépítettem a LibQuicR könyvtár általam módosított verziójába, ez segíti a protokoll működésének nyomon követését és az elemzést például Wireshark eszközzel. Elvégzett módosítások: github.com/Schweitzee/libquicr (ssl-keylog branch)
- A Mininet program segítségével egy demonstrációs környezetet alakítottam ki, amelyben a megvalósítás különböző hálózati szcenáriókban is kipróbálható.

2.2.2. Továbbfejlesztési lehetőségek

Ez az implementáció az első ilyen jellegű próbálkozás a Media over QUIC (MoQ) architektúrában, és bár jelenleg csupán kezdetleges multipath funkcionalitást nyújt, már most jól látható, hogy számos továbbfejlesztési lehetőséget kínál, amelyek megvalósításához a félév során elvégzett munka szilárd kiindulási alapot jelent.

A projekt lehetséges folytatási irányai közé tartozik például a relay oldali fejlesztések kibővítése oly módon, hogy az ne csupán egyetlen, hanem több interfészt is képes legyen aktívan használni a továbbítás során.

Emellett további multipath képességek is implementálhatók, amelyek a PicoQUIC picoquicdemo programjában már elérhetők, de ebben az implementációban még nem kerültek alkalmazásra. Ilyen lehetőség például a kapcsolati azonosítók dinamikus megújítása, vagy a NAT rebinding kezelése. Utóbbi alatt azt értjük, hogy a hálózati címfordító (NAT) eszköz egy új külső IP-cím és port párost rendel a klienshez – például újracsatlakozás esetén –, amit a protokollnak megfelelően le kell tudni követni. Szintén megvalósítható az azonos IP-címről, de eltérő porttal indított új útvonalak kezelése, illetve annak a lehetősége is, hogy a kapcsolat teljes egészében átmigráljon egy másodlagos interfészre, ha például az elsődleges hálózati kapcsolat megszakad vagy instabillá válik.

Ezen felül érdemes megvizsgálni a multipath továbbítás alkalmazhatóságát relay-ek közötti kapcsolatok esetén is, hiszen ez további redundanciát és rugalmasságot biztosíthat a rendszer egészében.

Végül egy hosszabb távú fejlesztési irány lehet egy olyan demonstrációs program készítése, amely valós médiatartalmakat – például élő videó streamet – továbbít multipath használatával, ezzel lehetőséget nyújtva a gyakorlati előnyök részletesebb értékelésére és bemutatására.

3. Irodalom, és csatlakozó dokumentumok jegyzéke

3.1. A tanulmányozott irodalom jegyzéke

[1] Douglas Karr, *Live Streaming Trends and Statistics (2024)*, (Accessed: 2024-05-12) https://martech.zone/live-streaming-trends-statistics/

[2] J. Iyengar and M. Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, IETF, May 2021. https://www.rfc-editor.org/rfc/rfc9000

[3] QUIC Working Group, *Multipath Extension for QUIC, draft-ietf-quic-multipath-14, 2025. apr. 24.* https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/14/

[4] Christian Huitema / Private Octopus Inc., *PicoQUIC*, [GitHub repository], (Accessed: 2024-05-12) https://github.com/private-octopus/picoquic

[5] IETF, *Media over QUIC Transport, 2025. apr. 28.*, https://datatracker.ietf.org/doc/draft-ietf-moq-transport/11/

[6] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*, pages 19–24, Monterey, California, 2010. Association for Computing Machinery. https://doi.org/10.1145/1868447.1868466.

[7] Quicr, *LibQuicR*, [GitHub repository], (Accessed: 2024-05-12) https://github.com/Quicr/libquicr

[8] Robin Marx. qlog: A standardized schema for logging and visualizing QUIC events. (Accessed: 2024-05-12) https://github.com/quiclog/qlog

[9] Robin Marx. qvis: A web-based tool for visualizing qlog files. (Accessed: 2024-05-12) https://qvis.quictools.info

3.2. A csatlakozó dokumentumok jegyzéke

Saját multipath implementációm GitHub repository-ja: https://github.com/Schweitzee/libquicr/tree/multipath

SSL keylog fájl generálásához szükséges módosítások: https://github.com/Schweitzee/libquicr/tree/ssl-keylog