Többutas megoldások és stratégiák többinterfészes mobil környezetben

Fejes Ferenc, Katona Róbert, Püsök Levente
Debreceni Egyetem Informatikai Kar
4028, Debrecen, Kassai út 26.
{fejes, robert.k, pusok.levente}@opmbx.org

***Kivonat*—Napjaink hálózatain jelen van több elérhető hálózati útvonal a végberendezések között. Ez nem csak a számítógépes környezetre igaz, hanem a mobil eszközökre is. Egy belépő kategóriás mobil eszköz is rendelkezik több, általában heterogén technológiát használó rádiós hálózati interfésszel (Wi-Fi, WiMAX, LTE, 3G, Bluetooth LE, stb.). Az ezekhez tartozó hálózatok késleltetése, sávszélessége, hibaszázaléka és egyéb karakterisztikái egymástól eltérőek lehetnek. A munkánkban röviden összefoglaljuk napjaink megoldásait több hálózati interfész együttes használatára. Röviden bemutatunk néhány már megvalósított többinterfészes mobilos környezetet, ezek működési elvét, előnyeit és hátrányait. Áttekintjük a többinterfészes mobil működés főbb nehézségeit, ezek megoldásait az egyes rendszerekben. Bemutatjuk a Debreceni Egyetem Informatikai Karán kifejlesztett MPT Library-val korábban elért, a témához kapcsolódó eredményeket és ismertetjük a szoftver Androidos implementációjának a részleteit és koncepcióit. Megmutatjuk a megoldás különbségeit más megoldásokhoz képest és vizsgáljuk a továbbfejlesztési lehetőségeket is.**

I. Bevezető

Az egyre növekvő felhasználói igényekhez igazodva manapság sok helyen érhető el valamilyen publikus Wi-Fi internet hozzáférési lehetősége. Egyre több internetszolgáltató indítja el valamilyen előfizetés ellenében hozzáférhető városi Wi-Fi hálózatát, de több nagyváros forgalmasabb részén is elérhetőek szabadon hozzáférhető hálózatai. Hasonló mértékben növekszik világszerte a mobilnetes lefedettség is. A nagy sebességű LTE mobilnetes hozzáférés mára a világ mintegy 148 országában elérhető [1], ami egy nagyon dinamikus fejlődést jelent az LTE 2010-ben történő bevezetése óta. Ugyan ezen felmérés rámutatott, hogy bár nagyok a különbségek az egyes eszközök hozzáféréseire nézve, az átlagos Wi-Fi hozzáférési sávszélesség 6 Mbit/s, míg az LTE 13.5 Mbit/s (ezek mögött lemaradva a 3G átlagos 3.5 Mbit/s sebessége). Manapság a legtöbb mobil eszköz több rádiós hálózati interfésszel is rendelkezik, mint például a Wi-Fi vezeték nélküli interfész és az LTE modem. Több kutatás is elkezdődött annak vizsgálata érdekében, hogy miképp lehetne együtt használni több interfészt mobil eszközökön a felhasználói élmény növelése. Különböző megoldások és stratégiák születtek a témában, ezek mindegyikének fő célja, hogy megoldjanak valami olyan problémát, mely az egy interfészes működéssel hozható összefüggésbe. Mobil eszközöknél főbb problémák a kapcsolat megszakadása valamilyen mozgás következtében (például egy Wi-Fi bázisállomás hatókörét elhagyva az aktív kommunikációs viszonyok megszakadhatnak), magas késleltetés és ebből adódó magas körülfordulási idő (round-trip time, RTT) főleg 3G hálózatok esetében, Wi-Fi kapcsolat jellegéből adódó véletlen csomagveszteség és magas késleltetés ingadozás (jitter), stb. Amennyiben több interfész áll rendelkezésre, abban az esetben azokat konkurens módon, egyszerre használva lehetőségünk van azok sávszélességének összegzésére gyorsabb letöltéseket eredményezve. Növelhető a kapcsolatunk megbízhatósága "make-before-break" módon váltva interfészt még a tényleges, fizikai kapcsolat megszűnése előtt. Egyes esetekben üzemidőt is nyerhetünk, ha egy adott alkalmazás igényeit kielégíti egy esetlegesen lassabb, de energiatakarékosabb vezeték nélküli kapcsolódási lehetőség (pl. Bluetooth LE) [2]. Nem elhanyagolható szempont az egyes hálózatok elérésének financiális költségei sem, hiszen valamilyen 3G vagy LTE áll előfizetés által felhasználható adatforgalmazási lehetőség nem korlátlan mennyiségben rendelkezésre. A továbbiakban mobil többutas megoldásokat mutatunk be és ismertetjük a saját megoldásunkat is. Az első egy intelligens, QoS (Quality of Service) megfontolásokat is végző flow-bindingon alapuló rendszer, amely elsősorban mobil egészségügyi (mHealth) alkalmazáshoz lett kifejlesztve, de egy általános Mobile IPv6 megoldás Linux és Android rendszerekhez. Az utóbbi évek fontos többutas technológiája az Multipath TCP [3], amely a hagyományos TCP többutas kiterjesztése. A protokoll mobil környezetben való kutatásáról több munka is született, melyek különböző esettanulmányokon és teszteken keresztül vizsgálják a hatékonyságát, mi ezekből ismertetünk néhányat röviden. Egy Androidos Open vSwitch-en alapuló többinterfészes megoldást is bemutatunk, ami szintén képes több link sebességének aggregációjára és intelligensebb váltásra hálózatok között. Végül a saját munkánkat ismertetjük, melyben egy már korábban kifejlesztett és tesztelt rendszernek, az MPT Library-nek [4] az Androidos kísérleti implementációját mutatjuk be, mely szemben a többi megoldással, nem igényel speciálisan (szoftveresen) módosított mobil készüléket a többutas hálózati működéshez. Kitérünk a megoldás előnyeire és nehézségeire egyaránt.

II. Intelligens folyam-menedzsment Android környezetben

Flow alatt a jelenlegi terminológiában csomagok egy olyan sorozatát értjük, melyek speciális kezelést igényelnek nem csak a küldő oldalán, hanem a köztes berendezésektől is. Flow binding alatt azt értjük, hogy adott forgalom szelektor mező (traffic selector, pl. a csomagok fejlécében lévő feladó IP, cél IP, transzport protokoll, portszám, magasabb réteg információ, stb.) által meghatározott csomagokat valamilyen adott módon dolgozunk fel (például magasabb/alacsonyabb prioritás, eltérő routing, stb.). A témával több RFC is foglalkozik, ezek közül a mobilos környezethez kapcsolódóak [5] [6], illetve néhány főbb use-case [7] [8]. Vegyünk egy példa eset-tanulmányt. A

legtöbb applikáció valamilyen ismert, rá jellemző transzport réteg (TCP/UDP) portszámot használ. Egy flow-binding intelligencia képes ebből felismerni az applikációt és ennek megfelelően választani például kimeneti hálózati interfészt (például HTTP menjen LTE-n ha elérhető, SSH vagy más szöveges chat alkalmazás 3G-n és FTP, RTP vagy más sávszélesség igényes alkalmazások Wi-Fi-n).

Egy komplex, flow-bindingot is támogató Linuxos és Androidos többinterfészes környezetet implementált Varga N. és társai [9]. A MIP6D-NG (Mobile IPv6 Daemon - Next Generation) egy komplex megoldás, amely több réteg információit (MAC, IP, transzport vagy ezeknél is magasabb) monitorozza valós időben és ezek alapján hoz különböző döntéseket a folyamok kezelésénél. Adaptív módon, az aktuális hálózat QoS paramétereit (késleltetés, csomagvesztés, jitter, stb.) figyelembe véve dinamikusan helyezi át más hálózati interfészre a már élő folyamokat (a rendszer terminológiájában a folyam egy 5 tuple, cél és feladó IP, cél és feladó portszám valamint transzport protokoll típusa). Léteznek megoldások és javaslatok ilyen, ún. vertikális handoverre, amikor az applikációs réteg alatt váltunk szolgáltatót vagy hálózati interfészt (ez történhet Layer 2-ben, Layer 3-ban és transzport rétegben is). Egy ilyen megoldás alapja lehet például a jelerősség: amennyiben valamilyen kritikus szint alá csökken, abban az esetben - még a tényleges kapcsolat megszakadás előtt - átvált egy másik átviteli technológiára. Az ilyen megoldások hátránya, hogy minden aktív folyam át lesz helyezve egy új interfészre, valamint a döntést csak a jelerősség alapján hozza meg a váltásról. Jelen megoldás egy komplexebb döntési motorral (decision engine) rendelkezik, amely amellett, hogy figyelembe veszi a hálózat más paramétereit is, képes arra, hogy egyes folyamokat szeparáltan helyezzen át egy másik interfészre. A testbed architektúra egy módosított Android verzión alapszik. A módosítások több okból is szükségesek voltak, hiszen bár a MIP6D-NG az Android natív részében fut, speciális kernel-space módosításokra is szükség van a működéséhez, például kibővített IPv6 támogatás, alap Android kernelből hiányzó modulok hozzáadása és betöltése (virtuális bridge interfész támogatáshoz). Az Android OS részében is szükség volt módosításokra annak érdekében, hogy ne kapcsolja ki a 3G interfészt aktív Wi-Fi interfész esetén (ez ugyanis az alap működési szabály), ez által lehetővé téve az interfészek közötti a lehető leggyorsabb átváltást. Az alap működés során minden flow a 3G interfészre kerül regisztrálásra. Ez után az elérhető Wi-Fi hálózatokon több rétegben méréseket végez. Link layerben jelerősség információk alapján sorba rendezi az elérhető hálózatokat, felcsatlakozik a legerősebbre, majd hálózati rétegbeli méréseket végez (válaszidő, csomagvesztés és jitter) melyek alapján dönt, hogy megfelelő-e a QoS profilnak. Amennyiben nem, megvizsgálja a következő elérhető Wi-Fi hálózatot. Ha ez megfelelőnek bizonyul, akkor áthelyezi az adott QoS profilt használó applikációk flowjait és frissíti a flow adatbázisát.

A működés hatékonyságát több teszttel is vizsgálták. A mobilos testbed egy MIP6D-NG környezettel ellátott eszközből, két Wi-Fi routerből, egy 3G hálózatból (OpenVPN segítségével kapott IPv6-os címet a 3G interfész) és egy végberendezésből (TCP letöltő és UDP listener funkciót ellátó) állt oly módon, hogy a végberendezés és a mobil node között egy WAN emulációt ellátó gép (Wanulator WAN emulációs szoftverrel) volt elhelyezve. Ezen haladt át mindhárom lehetséges útvonal a mobil eszköztől a végpontig. Így szabadon, egymástól függetlenül lehetett módosítani az útvonalak QoS paramétereit az útvonalakon. A kísérlet a 90 másodperc alatt átvitt bájtok számát mérte miközben új Wi-Fi hálózatok váltak elérhetővé. Amennyiben egy új Wi-Fi hálózat megjelent és a QoS paraméterek megfelelőek voltak, az átvitel közben arra került át a flow. Normális esetben ez gyorsabb átvitelt eredményezett. Viszont ha az elérhetővé vált Wi-Fi kapcsolatra csatlakozott mobil és a végpont között a WAN emulátorral bizonyos mértékű csomagvesztés lett beállítva, a nagyobb jelerősség ellenére sem került át rá a folyam. Ez a megoldás jelentős előnyt jelent a leginkább elterjedt jelerősségen alapuló statikus flow binding algoritmusokkal szemben. Mivel a handover decision engine teljesen absztrakt módon van megvalósítva, a megoldásba tetszőlegesen bonyolult mechanizmus is megvalósítható, ami fontos érzékeny alkalmazások esetén. A hátránya a bonyolultabb hálózat, vizsgálatok esetén történő lassabb működés: az algoritmus lefutása bizonyos időt vesz igénybe, mely a jelenlegi esetben 20 másodpercig is eltarthat.

III. Kimenő forgalom felosztása több interfész között

Egy másik ötlet, stratégia, amikor egy folyam vagy kommunikációs viszony forgalmát osztjuk fel több hálózati interfész között. Amennyiben több kijárati ponttal rendelkezünk a távoli hoszt felé, abban az esetben lehetőség nyílik megfelelő stratégia mellett ezeknek az útvonalaknak a kapacitását összegezni (link aggregáció). Yap K. K. és társai egy olyan környezetet javasoltak és implementáltak, melyben a kimeneti interfészek között kerül felosztásra a forgalom. A megoldás alapját egy általuk kernelbe implementált Open vSwitch [10] képezi, amely load balancer funkciót lát el (az Open vSwitch alapértelmezetten megtalálható a Linux kernelben 3.3-as verzió óta). Ahhoz, hogy ez működjön további kernel modulokat is be kellett fordítani a kernelbe, mint például a virtuális ethernet interfészek támogatásáért felelőt. A megoldásban egy virtuális ethernet interfész (ezen egy virtuális IP címmel) forgalmát osztja fel a fizikai interfészek között. Ez a módosított kernel verzió tesztelésre került mind notebookon, mind Andoridos készülékeken. Az alap Android operációs rendszerben is kellett módosításokat végezni. Akárcsak a fenti, MIP6D-NG esetében, itt is engedélyezni kellett több hálózati interfész együttes használatát. Az Open vSwitch-el a kommunikáció OpenFlow [11] protokollon keresztül valósult meg. A data plane a kernel spaceben volt, a control plane pedig jelen esetben az eszközön, tehát a felhasználó vezérelhette a forgalom felosztását. A control plane technikailag lehetne bárhol. A megoldásban a control plane-ek egymással is kommunikálhatnak, JSON üzenetek formájában. Erre szükség is van, hogy a forgalom interfészek közötti felosztása és a vevő oldali összeállítása összehangoltan működjenek. A megoldás nem igényli a köztes aktív eszközök módosítását, viszont feltételezi hogy a fogadó fél is rendelkezik ezzel a módosított hálózati stackkel.

A mérések igazolták, hogy a módosított kernel hatékonyan működik mobil eszközökön (három mobil készüléken is lett tesztelve). A késleltetésre nem volt észrevehető hatással az Open vSwitch match-action alapú flow táblája, viszont a kapcsolások az átvitel sebességét 2%-al csökkentették, a CPU használatot 1.8%-al növelték, de ezek az értékek elhanyagolhatóak. A mobil készülék Wi-Fi és WiMAX interfészekkel

rendelkezett és ezeket egyszerre használta az adatok fogadáshoz. Az eredmény, több mérés átlaga alapján, a teszt mobil készüléken 77%-os linkaggregációt tett lehetővé (iperf-es TCP sebesség mérések alapján). Egy másik esettanulmányban vertikális handovert hajtottak végre a megoldással. Egy HTTP fölött futó videó folyam lejátszása közben történt váltás az interfészek között. Megvalósításban ez úgy történt, hogy a fizikai interfész váltás előtt még a régi fizikai interfészről de már az új interfész feladó IP-vel kerültek kiküldve a csomagok és ment egy kontroll üzenet a túlsó félnek a váltás szándékával (helyi control plane küldte a távoli control plane-nek). Ez után már fizikailag is az új interfészről kerültek kiküldésre a csomagok. Ennek eredményeként a kommunikációs viszony nem szakadt meg a két fél között és a videó folyam rendben folytatódott tovább. A váltás manuálisan lett kezdeményezve, de a kontroll interfész tetszőleges OpenFlow kompatibilis megoldás lehet, a fejlesztők javasoltak egy megoldást, amely gyorsulásérzékelő adatai alapján választ: mozgás esetén WiMAX-ot, ha nem mozog az eszköz Wi-Fi-t.

Néhány megoldási nehézsége is van a rendszernek. Az egyik, hogy ha azonos privát címtartományt használ mindkét elérhető alhálózat, akkor nincs mi alapján eldönteni közvetlenül elérhető fél esetén, hogy melyik interfészen is kell kiküldeni a csomagot. A felderítési protokollok (pl. DNS, DHCP) általában egy interfészhez tartoznak, ezért a fizikai interfészek állapotához és az elérhető hálózatokhoz igazodva frissíteni kell ezek állapotát. A middleboxokkal (NAT, proxy, tűzfalak) való együttműködés sem triviális, hiszen egyes készülékek érzékenyek arra, hogy volt-e háromutas kézfogás az adott TCP folyam esetén, ami nem feltétlenül igaz, hiszen a forgalom felosztásra kerül az interfészek között, amik a külvilágot nem biztos, hogy ugyan a mögül a NAT mögül érik el. Az utolsó problémás terület a fejléc formátum. Létezik, hogy készülékenként eltérnek a fizikai interfészek kereteinek formátumai, valamint az MTU méretei. Ezek a problémák azonban nem zárják ki a megoldás használhatóságát, további munkával és módosításokkal megoldhatóak.

IV. Többutas TCP (MPTCP) mobil környezetben

Az MPTCP [3] egy szabványos többutas technológia. Funkcionalitását tekintve a transzport rétegben foglal helyet, lehetővé teszi több interfész kapacitásának összegzését és képes backup útvonalak létrehozására, illetve az elsődleges útvonal meghibásodása esetén ezekre történő gyors váltásra. Működését tekintve az adatfolyamot felosztja az elérhető hálózati interfészek között úgy, hogy hagyományos TCP alfolyamokat hoz létre rajtuk, melyek kapacitását intelligens módon használja fel. A fő tervezési szempontok között szerepelt a TCP-vel való kompatibilitás, együttműködés a legkülönfélébb NAT megoldásokkal és fairness torlódás szabályozás hagyományos TCP folyamok mellett is. Az applikációk a szabványos socket API-t használják, így nem kell rajtuk semmilyen speciális módosítást végezni a kompatibilitás érdekében. A megoldás kernel-spaceben került implementálásra, ami azt jelenti, hogy lehetőség van mobil eszközön való tesztelésre is. Ebben a témában több eredmény is született.

Az MPTCP működéséből adódóan alkalmas vertikális handover lebonyolítására is: amikor valamelyik alfolyamon hibát érzékel (pl. Wi-Fi kapcsolat megszűnése miatt), akkor a forgalmat áttereli a működő (pl. 3G vagy LTE) útra, így az MPTCP kommunikációs viszony nem szakad meg. Ez által növelhető a megbízhatósága a kapcsolatnak, a felmerülő nehézségeket és a váltás hatékonyságát vizsgálja [12]. Az MPTCP első ilyen irányú széles körben elterjedt implementációja az Apple iOS 7 mobil operációs rendszerében volt 2013-ban, annak érdekében, hogy a hangfelismerő applikáció hatékonyságát ne rontsa Wi-Fi és 3G/LTE közötti váltás. Sajnos az implementáció részletei nem ismertek, sem a felmerülő nehézségek és azok megoldásai.

A második kereskedelmileg elérhetően bevezetett MPTCP megoldás 2015-ben a Korean Telecomé [13]. A megoldás neve GIGA Path, a célja pedig a jelenleg elérhető leggyorsabb mobilos technológiák sávszélességének aggregációja. Ez az MPTCP alfolyamokat egyszerre használja adatátvitelre mindkét interfészen a mobilkészülék és a szolgáltató MPTCP képes SOCKS proxy szervere között. Ez után a proxy szerver hagyományos TCP-t használva, a szolgáltató szélessávú gerinchálózatán kommunikál az internettel. Az így elérhető elméleti sebesség: 300 Mbit/s LTE-A és 867 Mbit/s Wi-Fi összege, nagyjából 1.17 Gbit/s. Ezt jelenleg csak néhány csúcskészülék támogatja, az ezeken mért sebesség is elmarad az elméleti maximumtól, nagyjából 800 Mbit/s, viszont ez újabb, gyorsabb készülékek megjelenésével változni fog.

Az MPTCP mobilos környezetben való alkalmazásának másik lényeges kérdése az energiatakarékos működés. Ez irányú mérések igazolták [12], hogy a megoldás több interfész együttes használatával kevesebb energiát fogyaszt ugyan akkora fájl letöltése esetén, mint ha magán a 3G interfészen menne az adatfogadás. Egy másik megoldás [14] még tovább finomítja az energiahatékonyságot, az eMPTCP variáns valós környezetben mérve kevesebb energiát fogyaszt mint az MPTCP.

Egyik hátránya a socket API-val való kompatibilitásnak az MPTCP esetében az, hogy a felhasználó számára nem létezett olyan eszközkészlet, mellyel explicit módon vezérelhette volna a dataplane-t, vagyis az alfolyamok létrehozását, törlését, stb. Ez irányú törekvések történtek nemrég. Implementációban eltérő megoldások, de a szándék mindkét esetben valamilyen, az applikáció és a felhasználó felőli kontroll interfész biztosítása (control plane). A Socket Intents [15] ötlete azon alapszik, hogy egy socket létrehozásánál az applikáció ismer olyan plusz információkat a kommunikációról, mint annak a várható hossza (adat folyam esetén), az átvitel ideje és a bitarány egy videó vagy hang folyam esetén vagy akár az átvitel típusa (egy nagyobb fájl letöltése, mikor a sávszélesség kritikus, vagy sok kis fájl letöltése egy HTTP kérésnél, amikor az alacsonyabb RTT a lényeges). A kiterjesztés lehetővé teszi, hogy az applikáció jelezhesse, milyen igényei vannak az átvitelt tekintve, ennek hatására születik döntés a hálózati erőforrások használatáról (pl. melyik elérhető hálózat lesz használva, vagy hány alfolyam lesz létrehozva). A másik javaslat, a SMAPP [16] (Smart Multipath TCP-enabled APPlications), amely egy interfészt biztosít a kernel-spaceben futó MPTCP dataplane bizonyos eseményeire történő feliratkozásra (pl. egy kapcsolat felépülésére, egy alfolyam létrejöttére, a kapcsolat befejezésére). Az események bekövetkezésekor az applikáció a megoldás által definiált interfészen át módosíthatja a dataplane működését. Ilyen módosítás lehet például egy alfolyam törlése, vagy új alfolyam létrehozása, de bármikor lekérdezhetjük a kapcsolat aktuális állapot információit és ezekre reagálva is megvalósíthatjuk a saját alfolyam vezérlő intelligenciánkat.

V. MPT Android

Az MPT Multipath Library [4] egy többutas tunnelezésen alapuló megoldás. Az alapötlet a két végpont között egy tunnel kapcsolat létrehozása és a virtuális tunnel interfész forgalmának felosztása a fizikai hálózati interfészek között. A megoldás ilyen módon a hálózati rétegben működik a felsőbb rétegbeli entitások számára, így az applikációk tetszőleges transzport rétegbeli protokollt használhatnak a működéshez. A tunnel forgalmának az enkapszulációja a GRE in UDP internet drafton alapszik [17]. A technológia alkalmazható a tunnel alatti fizikai interfészek sebességének aggregációjára, ezt több munkában is vizsgálják [18], [19], [20]. Más munkáinkban mobilos környezetben is alkalmazható eseteket vizsgálunk. Az MPT-vel megvalósított vertikális handovert vizsgálja karakteres távoli terminál környezetben [21]. A vizsgálat során a két végpont között kiépült tunnel interfészen egy ssh távoli kapcsolat lett létrehozva. Ezen a kapcsolaton keresztül a távoli fél gépről lett megpingelve a kezdeményező gép tunnel interfésze, eközben pedig a kontroll interfésszel történt út le- majd visszakapcsolás. A kísérlet során nem történt sem csomagveszteség, sem ssh kapcsolat megszakadás. A másik tesztsorozatban [22] [23] az ilyen jellegű vertikális handover hatását vizsgáltuk HTTP és RTP alapú videó folyamokra. Megnéztük továbbá, hogy milyen hatást gyakorol rá a tervezett út leállítás (a kontroll interfész segítségével) és a nem tervezett leállítás (a Wi-Fi router WAN portjából kihúztuk a kábelt). A szoftver rendelkezik egy kontroll interfésszel. Ezzel többek között megvalósítható egy csomagvesztés mentes útvonal vagy interfész lekapcsolása, automatikus tunnel kapcsolat felépítése a felek között, új útvonal hozzáadása, konfiguráció futás közbeni megváltoztatása. A korábbi kutatások által elért eredmények arra bátorítottak minket, hogy elkészítsük a szoftver mobil eszközökre szánt verzióját. Szerettük volna, ha szoftver kielégíti az alábbi kritériumokat:

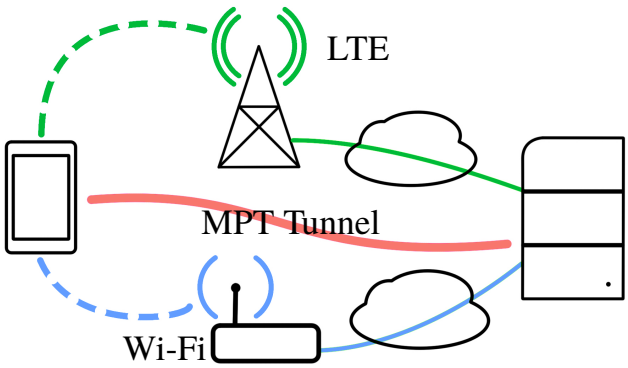
• Ne igényeljen speciális, módosított kernelt. A fentebb bemutatott valamennyi megoldás azon alapszik, hogy a mobil eszköz által használt kernel a megoldás igényeinek megfelelően lett módosítva, speciális kiegészítések lettek rajta eszközölve. Ennek megvan az az előnye, hogy a felhasználó nem tud hozzáférni, nem tudja rosszul konfigurálni a működését. További nagy előnye, hogy gyors: ha a dataplane kernelspaceben van elhelyezve, úgy kevesebb másolással tud működni, nincs felesleges adatkommunikáció kernelspace és userspace között. Bár folynak kutatások nagy sebességű és userspaceben működő network stack megoldásokkal [24], ezek egyelőre mobil eszközökön nem elérhetőek. Az egyre gyorsabb mobil és SBC (Single Board Computer) processzoroknak [25] és a függetlenül ez irányban folyó kutatásoknak köszönhetően elképzelhető, hogy használhatóak lesznek ezek a megoldások mobilos környezetben is, még abban az esetben is, ha nem az a cél platform. Lényeges, hogy ne a processzor limitálja az elérhető maximális (akár aggregált) sávszélességet.

• Ne igényeljen speciálisan módosított Android OS-t. Több fenti megoldás is támaszkodik az Android operációs rendszer valamilyen szintű módosítására. Az Android hálózati interfészek kezelésének alapfilozófiája, hogy amennyiben elérhető Wi-Fi-s hálózat, abban az esetben a mobilos interfészeket (WiMAX, 3G, LTE, stb.) nem használja adatkommunikációra és alvó állapotba helyezi őket. A felhasználó dönthet a mobil adatkommunikációról manuálisan felülbírálva az operációs rendszer döntését, de ez csak arra vonatkozik, hogy ha a Wi-Fi hálózat nem elérhető, használható legyen-e a mobilos interfész adatkommunikációra. Annak érdekében, hogy Wi-Fi hálózat elérhetősége esetén a mobilos interfész is használható legyen, módosítani kell a beépített Connectivity Service-t, felülbírálva annak az alapértelmezett viselkedését. Ez nem egy egyszerűen megoldható probléma, speciális, gyártó általi módosítást igényelne, vagy valamilyen ilyen irányban módosított egyedi ROM-ot. Egy ilyen ROM csere viszont egyes gyártók készülékeinél a garancia elvesztését okozza. Szerencsére az elmúlt időszakban az Android API-ba bekerültek több interfészes működést támogató kiegészítések [26]. Ezeknek az új API hívásoknak a segítségével jelezhetjük a rendszer felé, hogy valamilyen típusú hálózathoz tartozó interfészt szeretnénk igénybe venni (Wi-Fi, Cellular, Ethernet, stb.). Abban az esetben ha ez megoldható, applikációs szinten használhatjuk ennek a hálózatnak az erőforrásait. Ez az új API támogatás az MPT androidos implementációjának is az alapja.

• Ne igényeljen rootolt eszközt. Az Android operációs rendszer alapjait a hagyományos GNU/Linux rendszer képezi. Ennek megfelelően jelen vannak az ott megismert felhasználói jogosultságok és azok kezelése (persze ez a felhasználó számára transzparens módon van jelen). Rootolt androidos eszköz alatt pedig azt értjük, hogy a felhasználó rendelkezhet kiterjesztett, mindenre kiterjedő jogosultságokkal, szabadon módosíthatja a rendszer működését (routing tábla bejegyzések, hálózati interfészek címei, állapotuk, stb.). Ez viszont biztonsági okokból a legtöbb készülék rendszerén ki van kapcsolva, hiszen rosszindulatú alkalmazások ez által átvehetik a kontrollt a rendszer olyan részei felett, amelyeket nem szabályoz megfelelően az API-ban megszabott és menedzselt, engedélyeken alapuló jogosultság kezelése. Léteznek eljárások root jogosultságok elnyerésére Android operációs rendszer alatt, sok alkalmazás igényli is ezt a működéséhez. Arra viszont nem szabadna kényszeríteni a felhasználót, hogy rootolja az eszközét a megoldás használatához, hiszen egyes gyártók esetében ez a garancia elvesztését okozhatja.

A fenti kritériumoknak megfelelő kísérleti implementációt elkészítettük és a teszteléséhez egy hétköznapi életben is előforduló szituációt használtunk. A mobil terminál forgalmat generál Wi-Fi-n (jelen tesztesetben egy TCP letöltés) közben mozog és elhagyja a hozzáférési pont hatósugarát. Ahogy távolodik, egyre csökken a jelerősség, mígnem annyira lecsökken, hogy a letöltési sebessége drasztikusan leesik, a letöltés többször is megakad kisebb ideig. A tesztekhez a 1. ábrán látható elrendezést használtuk. Az MPT Android szoftverben két útvonalat definiáltunk, az egyik a Wi-Fi-t, a másik az LTE-t használta. A szerveren az MPT szoftver Linuxos környezethez kifejlesztett C-ben írt verziója futott. Alapértelmezetten a letöltéshez a Wi-Fi-s útvonalat használtuk, az LTE úton nem ment forgalom. Ez az elrendezés hasonló a [27]-ben használthoz (melyben új LTE-t használó MPTCP alfolyamot indítanak a gyenge Wi-Fi esetén, valamint terminálják azt ha a jelerősség ismét jobb lesz. A váltáshoz egy módosított MPTCP stackkel ellátott kernelverziót használnak, ami lényeges különbség a mi megoldásunkhoz képest, ami mindent standard Android API-n keresztül valósít meg, beleértve a jelerősség figyelését is). A Wi-Fi hozzáférési pont a mobil terminálttól néhány méterre volt, egy téglafal mögött. Itt indítottuk el a letöltést, majd folyamatosan távolodtunk a hozzáférési ponttól egy egyenes folyosón, aminek a végén visszafordulva ismét

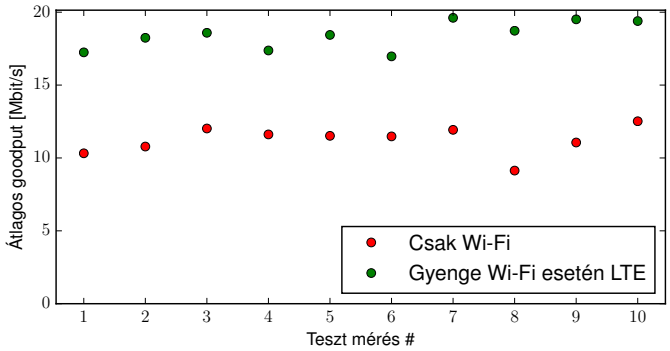
közeledni kezdtünk és megálltunk a kiindulópontnál, itt leállítottuk a letöltést. A letöltéshez egy néhány soros programot használtunk, ami minden másodpercben logolta, hogy hány bájtot töltött le eddig, valamint ezzel párhuzamosan az aktuális Wi-Fi jelerősséget is. Így nem szimpla átbocsátóképességet mértünk (melyben benne vannak az esetleges TCP újrakülldések is) hanem goodputot (az applikáció számára hasznos átbocsátóképesség).



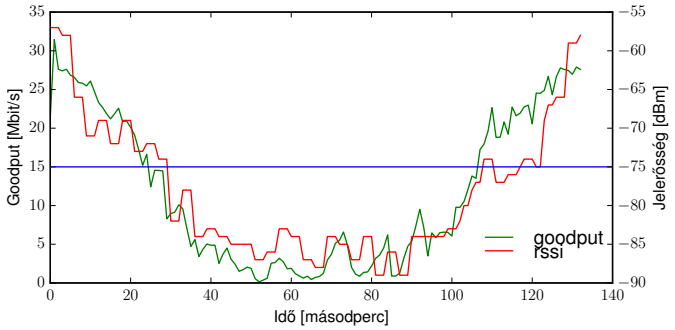
1. ábra. A tesztekhez használt topológia

A tesztekhez egy Samsung Galaxy Tab S2-t használtunk, gyári szoftverrel ami egy Android 6.0.1 verzió. Alapvetően két tesztesetet vizsgáltunk: egyik esetben a rossz jelerősség ellenére is maradtunk Wi-Fi-n végig, a másik esetben pedig ha a jelerősség egy bizonyos szint alá süllyedt (ez -75 dBm esetünkben, próbálkozások után választottuk, ugyanis ez alatt kezdett egyre lassulni a letöltési sebesség) akkor váltottunk az LTE-s útra, de amint ismét javult a jelerősség, visszaváltottunk Wi-Fi-re, minimalizálva az LTE interfészen az adatforgalmat.

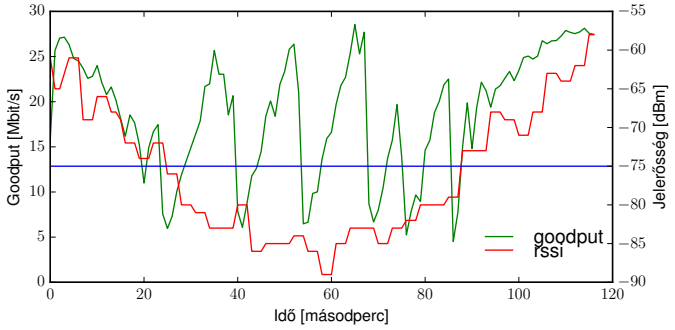
Mivel mind a Wi-Fi, mind az LTE letöltési sebesség fluktuált, ezért a fenti tesztet tízszer végeztük el (felváltva a csak Wi-Fi-s és a Wi-Fi-ről LTE-re váltásos eseteket) és a végén az átlagos goodput értékét vettük. Bár ez az érték nem a legjobb mérőszám az architektúránk hatékonyságának a méréséhez, hiszen az LTE sebessége eltérhet a Wi-Fi-től, esetünkben hosszabb iperf szoftverrel végzett mérés alapján közel azonos (25-27 Mbit/s) átlagos letöltési sebességet mértünk mindkét úton, jó jelerősség mellett.



2. ábra. Egyes mérések átlagos applikációs rétegbeli átbocsátóképességei (goodput).



3. ábra. A Wi-Fi jelerősség és aktuális letöltési sebesség végig Wi-Fi-t használva (egy kiragadott mérés a tízből).



4. ábra. A Wi-Fi jelerősség és aktuális letöltési sebesség végig abban az esetben, ha -75 dBm alá süllyedő jelerősségnél (vízszintes vonallal jelölve) LTE-t használtunk letöltéshez (egy kiragadott mérés a tízből).

Mint az a 2. ábrán látható, azokban az esetekben, amikor rossz jelerősség esetén is végig maradtunk Wi-Fi-n, a átlag letöltési sebesség a tíz mérést átlagolva 11,23 Mbit/s környékén alakult. Azokban az esetekben mikor váltottunk LTE-re a gyenge Wi-Fi jelerősség esetén, 18.41 Mbit/s lett a tíz mérés átlaga.

A 3. ábrán látszik, hogy korreláció figyelhető meg az aktuális letöltési sebesség és a jelerősség között. A csökkentő jelerősség eléri a -90 dBm-t, aminél már alig használható, akadozóvá válik a letöltés is. Ezzel szemben azokban az esetekben, mikor LTE-re váltottunk a beállított küszöb jelerősség alatt (mint az egy kiragadott mérés esetében a 4. ábrán is látszik) a letöltési sebesség nem esik le, hanem folytatódik az LTE interfészen (a zöld goodput görbén látszik is, ahogyan a gyorsan változó Wi-Fi helyett a TCP által LTE-n tapasztalt nagy körülfordulási idő és buffer méret dominál, az egyes torlódási csomagvesztések drasztikusan visszaveszik a küldő sebességét, ami aztán lassan újra megnövekszik a következő torlódásig). Korreláció a jelerősség és a sebesség között csak a küszöbérték fölött van, hiszen csak akkor használjuk a Wi-Fi-t.

Az általunk vizsgált eset úgy is módosítható, hogy figyeljük az LTE jelerősségét, vagy azt, hogy nem-e váltottunk mozgás közben egy másik cella bázisállomásra, amely csak 3G-t támogat, és ennek függvényében visszaváltunk Wi-Fi-re akkor is, ha annak a jelerőssége a küszöb alatt van.

VI. Összefoglalás és jövőbeli tervek

A fentebb ismertetett valamennyi technológia létrejöttét a felhasználói élmény növelése inspirálta. Mint az látszik, több elérhető hálózat megfelelő stratégia szerint történő intelligens alkalmazásával lehetőség van lényeges előnyökre szert tenni a kapcsolat megbízhatósága vagy épp sebessége szempontjából. Látható, hogy a stratégiák érvényesek és alkalmazhatók több hálózati rétegben is (adatkapcsolati, hálózati, transzport). Megmutattuk, hogy a többutas tunnelezés működik androidos mobil környezetben is, bármilyen kernel vagy szoftver módosítás nélkül, rootolatlan eszközön. Bemutattunk egy egyszerű esetet, melyben a mobil terminálnál robusztusabb letöltési sebességet tudtunk biztosítani egy egyszerű, hálózati interfész váltási stratégiával. Mivel a megoldás hálózati rétegben működik, lehetőség van TCP-n túli, akár kísérleti transzportprotokollok (vagy viszonyrétegbeli protokollok) többutas működésének a vizsgálatára. Érdekes kutatási téma még és a jövőbeli terveink között szerepel a TCP viselkedésének részletes elemzése gyakori, különböző sebességű és késleltetésű útvonalak közötti váltás esetén, valamint erre való modell felállítása. Cél egy TCP barát váltási stratégia kidolgozása, mely egyszerűen implementálható akár más, a miénkhez hasonló rendszerben.

Köszönetnyilvánítás

Szeretnénk megköszönni a munkában nyújtott segítséget Dr. Szilágyi Szabolcsnak a Debreceni Egyetem Informatikai Karáról valamint az Ericsson Traffic Laboratory munkatársainak, Dr. Rácz Sándornak és Dr. Szabó Gézának a segítségét.

Hivatkozások

[1] OpenSignal. The State of LTE (February 2016). Hozzáférve: 2016. 05. 08. [Online]. Available: https://opensignal.com/reports/2016/02/state-of-lte-q4-2015/

[2] G. Kalic, I. Bojic, and M. Kusek, „Energy consumption in android phones when using wireless communication technologies," in *MIPRO, 2012 Proceedings of the 35th International Convention*, May 2012, pp. 754–759.

[3] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, „TCP Extensions for Multipath Operation with Multiple Addresses," Internet Requests for Comments, RFC Editor, RFC 6824, January 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6824.txt

[4] MPT - Multipath Communication Library. Hozzáférve: 2016. 05. 08. [Online]. Available: http://irh.inf.unideb.hu/user/szilagyi/mpt/

[5] S. Gundavelli, K. Leung, G. Tsirtsis, and A. Petrescu, „Flow-Binding Support for Mobile IP," Internet Requests for Comments, RFC Editor, RFC 7629, August 2015.

[6] H. Yokota, D. Kim, B. Sarikaya, and F. Xia, „Flow Bindings Initiated by Home Agents for Mobile IPv6," Internet Requests for Comments, RFC Editor, RFC 7109, February 2014.

[7] C. Perkins, D. Johnson, and J. Arkko, „Mobility Support in IPv6," Internet Requests for Comments, RFC Editor, RFC 6275, July 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6275.txt

[8] G. Tsirtsis, H. Soliman, N. Montavont, G. Giaretta, and K. Kuladinithi, „Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support," Internet Requests for Comments, RFC Editor, RFC 6089, January 2011.

[9] MIP6D-NG. Next Generation Mobile IPv6 for Linux and Android. Hozzáférve: 2016. 05. 08. [Online]. Available: http://www.mip6d-ng.net/

[10] Open source multilayer virtual switch. Open vSwitch. Hozzáférve: 2016. 05. 08. [Online]. Available: http://openvswitch.org/

[11] OpenFlow. Hozzáférve: 2016. 05. 08. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow/

[12] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, „Exploring Mobile/WiFi Handover with Multipath TCP," in *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, ser. CellNet '12. New York, NY, USA: ACM, 2012, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/2342468.2342476

[13] IETF 93 – MPTCP WG. Korean Telecom Gigapath. Hozzáférve: 2016. 05. 08. [Online]. Available: https://www.ietf.org/proceedings/93/slides/slides-93-mptcp-3.pdf

[14] Lim, Yeon-sup and Chen, Yung-Chih and Nahum, Erich M and Towsley, Don and Gibbens, Richard J and Cecchet, Emmanuel, „Design, Implementation, and Evaluation of Energy-Aware Multi-Path TCP."

[15] Schmidt, Philipp S and Enghardt, Theresa and Khalili, Ramin and Feldmann, Anja, „Socket intents: Leveraging application awareness for multi-access connectivity," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 295–300.

[16] Hesmans, Benjamin and Detal, Gregory and Bauduin, Raphaël and Bonaventure, Olivier and others, „SMAPP: Towards Smart Multipath TCP-enabled APPlications," in *CoNEXT'15*, 2015.

[17] Lucy Yong and Edward Crabbe and Xiaohu Xu and Tom Herbert, „GRE-in-UDP Encapsulation," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-tsvwg-gre-in-udp-encap, Hozzáférve: 2016. 05. 08. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tsvwg-gre-in-udp-encap/

[18] Béla, Almási and Szabolcs, Szilágyi, „Multipath FTP and stream transmission analysis using the MPT software environment," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 11, pp. 4267–4272, 2013.

[19] Béla, Almási and Szabolcs, Szilágyi, „Throughput performance analysis of the multipath communication library MPT," in *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*. IEEE, 2013, pp. 86–90.

[20] Lencse, Gábor and Kovács, Ákos, „Advanced Measurements of the Aggregation Capability of the MPT Network Layer Multipath Communication Library," *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, vol. 4, no. 2, pp. 41–48, 2015.

[21] Béla, Almási, „A solution for changing the communication interfaces between WiFi and 3G without packet loss," in *Telecommunications and Signal Processing (TSP), 2015 38th International Conference*, July 2015, pp. 1–5.

[22] B. Almási and M. Kósa and F. Fejes and R. Katona and L. Püsök, „MPT: A solution for eliminating the effect of network breakdowns in case of HD video stream transmission," in *Cognitive Infocommunications (CogInfoCom), 2015 6th IEEE International Conference*, Oct 2015, pp. 121–126.

[23] Fejes Ferenc, Katona Róbert, Püsök Levente, „Eredmények a többutas hálózati kommunikációs technológiák területén," Hiradástechnika MediaNet 2015 különszám.

[24] Tazaki, Hajime and Nakamura, Ryo and Sekiya, Yuji, „Library Operating System with Mainline Linux Network Stack," in *Proceedings of netdev 0.1*, 2015, p. 6.

[25] Lencse, Gábor and Répás, Sándor, „Benchmarking Further Single Board Computers for Building a Mini Supercomputer for Simulation of Telecommunication Systems," *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, vol. 5, no. 1, pp. 29–36, 2016.

[26] IETF 91 - MIF WG. MIF in Android 5.0. Hozzáférve: 2016. 05. 08. [Online]. Available: https://www.ietf.org/proceedings/91/slides/slides-91-mif-3.pdf

[27] Y. s. Lim, Y. C. Chen, E. M. Nahum, D. Towsley, and K. W. Lee, „Cross-layer path management in multi-path transport protocol for mobile devices," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 1815–1823.