Budapesti Műszaki és Gazdaságtudományi Egyetem
Távközlési és Telematikai Tanszék

Útvonalválasztási Algoritmusok Tervezése és Teljesítményvizsgálata Adathálózatokban

Szviatovszki Balázs

Ph.D. Tézisek

Tudományos vezetők:

Dr. Faragó András
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas

Dr. Csopaki Gyula
Budapesti Műszaki és Gazdaságtudományi Egyetem
Távközlési és Telematikai Tanszék

Dr. Sallai Gyula
Budapesti Műszaki és Gazdaságtudományi Egyetem
Távközlési és Telematikai Tanszék

Budapest,
2002

1. Bevezetés

Húsz évvel ezelőtt a különböző alkalmazások által generált forgalom elvezetése dedikált hálózatokon valósult meg. Ebben az időben az áramkörkapcsolás elvét használó távbeszélő szolgáltatás jelentette a domináns forgalmat. Az egységes, több szolgáltatás együttes nyújtását levetővé tevő (u.n. multi-service) hálózatok kiépítése csak mostanában kezdődött el, amikor az internet és a mobiltelefónia által generált forgalommennyiség megközelítette a telefónia által generált forgalommennyiséget. A szigorú szolgáltatásminőség igények miatt, a multi-service hálózatok az aszinkron átviteli mód (ATM) technológiát veszik alapul. Manapság azonban a legforróbb téma a távközlési és az internet társadalomban egyaránt a csomagkapcsolás elvén alapuló, szolgáltatásminőséget (QoS-t) is biztosító hálozatokkal kapcsolatos kutatás és szabványosítás. Egyáltalán nem könnyű feladat azonban, egy hálózaton több szolgáltatás együttes, minőséget is garantáló elvezetésének biztosítása, valamint a fejlesztés során az egyszerűség elvének betartása.

Az Internet Szabványosítási szervezete, az IETF által definiált forgalomfolyam elvezetéssel kapcsolatos (u.n. traffic engineering) keretrendszer általános keretet ad az internet protokoll (IP) alapú hálózatok útvonalválasztására. Azonban az alap IP szolgáltatáshoz képest az új komponensek számos új kérdést vetnek fel, mind a berendezéstervezés, mind a hálózat optimális üzemeltetésének területén. A legfontosabb új komponensek:

- útválasztó protokollok forgalomfolyam elvezetéssel kapcsolatos él-állapot terjesztési kibővítései;
- sávszélességigényt figyelembe vevő útvonalszámítási algoritmusok;
- útfelépítést támogató jelzési protokollok;
- prioritás hozzárendelése forgalomfolyamokhoz, és forgalom kiszorítási metódusok;
- tartalék útvonal konfigurálási és számítási módszerek;

A MultiProtocol Label Switching (MPLS) azaz a címkekapcsoláson alapuló átviteli mód lehetőséget ad aggregált forgalomfolyamok útvonalának egyértelmű (explicit) meghatározására, úgynevezett címkekapcsolt utakon (label switched paths — LSP). Ez hasonló az ATM-ben használt kvázi-állandó virtuális kapcsolat (soft permanent virtual connection — SPVC) koncepcióhoz, amit PNNI (private network-to-network interface, azaz magán hálózat-hálózat interfész közti) útválasztó protokoll esetében használnak. Egy LSP sávszélességet foglal az útvonalán. Amikor a topológiailag legrövidebb úton nincs elég kapacitás a forrás és a nyelő pont között, lehetőség van hosszabb explicit út választására, amin a kért sávszélességigény teljesíthető. Az MPLS és a PNNI esetében is, az él-állapot alapú útválasztó protokoll terjeszti a hálózat összeköttetéseiről a topológiára és a sávszélesség foglalásra vonatkozó informaciókat. Az MPLS-ben ez erőforrás foglaltságot is tartalmazó él-állapot hirdetések (link-state advertisement —

LSA) terjesztésével valósul meg, amit az OSPF (open shortest path first) protokoll kiterjesztéseként valósítanak meg. A PNNI esetében az LSA-nak megfelelő entitást PNNI topológia állapot elemnek (PNNI topology state element — PTSE) nevezik.

A hálózatban terjesztett topológia és foglaltság információk lehetővé teszi, hogy a berendezések figyelembe vegyék a sávszélességigényt is az útvonalválasztás során. Az IETF szolgáltatásminőség garancia nélküli (u.n., best-effort) IP útválasztás továbbfejlesztéséül, szolgáltatásminőség igényeket figyelembe vevő (constrained shortest path first — CSPF) algoritmusokat [8] javasol. A CSPF algoritmusok a terjesztett erőforrás foglalási információkat egy kapacitás értékeket tartalmazó gráf felépítésére használják. Ezen a gráfon az útválasztás során figyelmen kívül hagyhatók azok az összeköttetések, amin nem áll rendelkezésre a megfelelő sávszélesség az LSP számára. Az így leszűkített gráfon, az igényektől függően, útvonal választó algoritmusként megvalósítható az egyszerű Dijkstra algoritmustól, bonyolultabb több metrikát használó algoritmusokig tetszés szerint bármi. Annak érdekében, hogy a hálózat üzemeltetési elvárások teljesüljenek, gyakran olyan útvonalat keresnek, amin nemcsak a kívánt sávszélesség áll rendelkezésre, hanem ami ráadásul globálisan nézve is optimális vagy közel optimális hálózati erőforrás kihasználtságot eredményez. Elosztott algoritmusok persze nem tudják garantálni a globálisan optimális erőforrás kihasználtság elerését, ezért valamiféle heurisztikán alapuló megoldást keresnek. Ezen heurszitikán alapuló módszerekben legtöbbször valamilyen intelligens terhelésmegosztásra irányuló technikát használunk [8]. Amikor a hálózati folyamok útvonalait központi helyen határozzák meg, a globális optimalizálás lehetősége is adott. Az irodalomban léteznek javaslatok ennek úgynevezett Traffic Engineering eszközökben való megvalósítására [11], [J6]. A LSPket a kiszámolt explicit utakon az RSVP és LDP kiterjesztéseként kapott protokollokkal építik fel, amikben ugynevezett Explicit Route Objects (EROs) mezők írják le az útvonal kötelezően érintendő csomópontjait.

Forgalomfolyamok prioritás alapú kezelésének fontos szerepe van az útvonalszámítás és a felépítés során egyaránt. A késleltetés érzékeny forgalomhoz magas prioritást rendelve elérhető, hogy az útvonalszámításkor minimális élszámú utakat vegyünk csak figyelembe. Ezt a sávszélesség foglalások megkülönböztetésével érjük el: egy adott prioritású forgalom útvonalválasztó algoritmusa, csak a hasonlóan fontos forgalom foglalásairól szerez információkat. Alacsonyabb prioritású sávszélesség foglalások figyelmen kívül hagyhatók, így, amikor az útvonalfelépítés történik, ezek kiszorítására külön eljárások szükségeltetnek. Ezáltal elérhető, hogy például a best-effort forgalom olyan utakra helyeződjön át, amit nem használnak fontosabb alkalmazások [10].

Bizonyos alkalmazások számára a kívánt sávszélességet még a hálózati összeköttetések hibája esetén is garantálni kell. Az MPLS helyreállítási (restoration) módszerek általában egyszeres hibát feltételeznek. Olyan élfüggetlen útpár számítása, ahol az élsúlyok összege minimális, megoldható minimális költségü folyamalgoritmusok segítségével [15]. Az így kapott útvonalak közül az egyik, tipikusan a rövidebb, kiépül, mint üzemi út, míg a másik tartalék útként tárolódik el. Amikor egy összeköttetés az üzemi út mentén meghibásodik, a tartalék út kerül felépítésre, hogy elvezesse a védett forgalmat.

Miután a meghibásodott összeköttetés újra működésbe lép, a forgalom visszaterelhető az eredeti útra. Elképzelhető az is, hogy az üzemi és a védelmi útvonal szerepet cserél a hiba elhárítás után, azaz, a forgalom nem terelődik vissza az üzemi útra. Ez, abban az esetben fordulhat elő, ha a védelmi út úgynevezett nemvisszahelyezendő (non-revertive) módon van konfigurálva.

2. Kutatási célkitűzések

Disszertációm célkitűzése gerinc adathálózatok teljesítmény–optimalizálási problémáinak vizsgálata, valamint meglévő útválasztó algoritmusok kiterjesztése vagy alternatív algoritmusok fejlesztése volt.

A disszertációmban, egy szolgáltató hálózatán belül jelentkező útvonalválasztási problémákra koncentrálok, és elsősorban él-állapot terjesztésen alapul útválasztó protokoll használatát feltételezem. IP hálózatok esetében ez azt jelenti, hogy a hálózat vagy az ISIS (intermediate system to intermediate system) vagy az OSPF (open shortest path first) protokollt használja, míg ATM esetében vizsgálatomat a PNNI protokoll vizsgálatára szűkítem. A fő érdeklődési terület ezeken a protokollokon belül, az útvonal számítási komponensben megvalósítot útkereső algoritmusok vizsgálata.

Útvonalszámításra alapvetően kétféle stratégia létezik: lokális vagy globális. Az első nagyobbfokú automatizálást tesz lehetővé, a CSPF algoritmus útválasztókban való implementálásával, míg a második az egész hálózatra kiterjedő erőforrás optimalizálást tesz lehetővé, többtermékes folyamprobléma megoldásán alapuló algoritmusokkal.

A disszertáció fő célkitűzései:

- olyan útvonalválasztó algoritmusok kifejlesztése, amelyek növelik a sávszélesség igénnyel rendelkező folyamok felépítési sikerességét;
- tartalék útvonal számítási algoritmusok kifejlesztése és analízise, figyelembe véve a gyors helyreállításra és optimális hálózati összeköttetés kihasználtságra vonatkozó szolgáltatói elvárásokat;
- MPLS hálózatokban a kiszorítási (preemption) folyamat, és az útválasztó módszer hatásának vizsgálata a kiszorítás miatti folyam áthelyezés gyakoriságára. Olyan útvonalválasztó algoritmusok kifejlesztése és analízise, amelyek elkerülik a szükségtelen kiszorítást;
- útvonalszámító algoritmus megvalósítások teljesítmény tesztelő módszereinek vizsgálata.

3. Kutatási módszerek

A három legfontosabb módszer hálózati architektúrák teljesítmény vizsgálatára az:

- analízis,
- a szimuláció, valamint
- a mérés, illetve prototípus készítés.

A 80-as években a PSTN hálózatok útvonalválasztási algoritmusainak vizsgálatához széles körben használtak analítikus módszereket. Egy adott útvonalválasztó algorimus viselkedéséről ez a megközelítés nyújtja a legmélyebb betekintést és a legáltalánosabb eredményt. Amint azt Ash is összefoglalta könyvében [13], számos kutató (többek között Gibbens, Mitra és Kelly [12]) által végzett tamulmány és ezek eredményei alapozták meg a dinamikus útvonalválasztó algoritmusokat használó telefonhálózatok méretezésének elméleti hátterét (például dinamikus nemhierarchikus útválasztás (DNHR) vagy a dinamikus alternatív útválasztás (DAR) esetében). A PSTN hálózatok fontos tulajdonsága, hogy ezek jelentősen összekötött topológiával rendelkeznek, minden csomópont közvetlen összeköttetéssel kapcsolódik egymáshoz, valamint a hívások maximum két élet tartalmazó utat használhatnak. Általában elmondható, hogy a telefonhálózatok, egyes tulajdonságaiban különböznek a ma elterjedt adathálózatoktól. A hálózattopológia a mai adathálózatokban ritkábbá vált, a hálózat átmérője megnőtt, és a forgalom is nehezen jósolható (a PSTNtől eltérően), ezért az analítikus módszerek a legtöbb esetben nehezen használhatók.

Útvonalválasztó algoritmusok teljesítmény vizsgálatára a *szimuláció* módszerét használják leggyakrabban. Én is ezt a teljesítmény analízis módszert használtam disszertációmban. A szimulációban, a probléma tipusától függően, az abszrakciós szint szabadon választható. A disszertációban tanulmányozott legtőbb útvonalválasztási problémához, elegendő volt egy egyszerűsített útválasztási modell használata, ami szükségtelenné tette az él-állapot terjesztésen alapuló útválasztó algoritmusok minden komponensének részletes modellezését. A hálózati folyamproblémák gyakran igénylik *lineáris programozás* használatát. Az első tézis során ezt a módszert is használom.

A legnagyobb fejlesztési munkát a prototípuskészítés és mérések végzése jelenti, de egyben ezek szolgáltatják a legpontosabb eredményeket is. A negyedik tézisben egy emulátor architektúrát javaslok, valós útválasztó protokoll megvalósítások vizsgálatára. A javaslat megvalósításra is került, amivel elvégeztem egy valós berendezés teljesítményvizsgálatát *mérések* segítségével.

4. Új eredmények

Tézis 1 LSP útválasztás MPLS hálózatokban egyetlen már felépített LSP áthelyezésével

A sávszélességigényt figyelembe vevő, MPLS címkekapcsolt utak (LSP) kihúzásánál használt útvonalszámítási algoritmus, némely esetben nem talál megfelelő utat az LSP számára, mert az LSP forrás és nyelő csomópontja között nincs olyan út, amin rendelkezésre állna a kívánt sávszélesség. Ebben az esetben kézenfekvő megoldásként a hálózat üzemeltető vagy blokkolja új LSPt, vagy az egész hálózat útvonalainak az újra optimalizálását kezdeményezi, erőforrások szabaddá tétele érdekében. Az első eset az alacsonyabb teljesítmény miatt alacsonyabb bevételt jelent, a második pedig nagymértékű LSP útvonal áthelyezést eredményez a hálózatban.

Tézis 1.1 Hatékony algoritmus az LSP áthelyezés feladatára

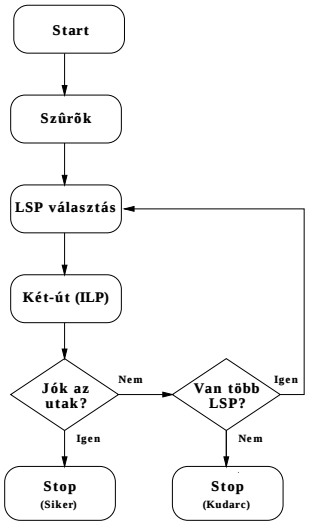
Javasoltam egy új koncepciót, ami szükségtelenné teszi a hálózat nagymértékü átkonfigurálását, mégis megnöveli a címkekapcsolt utak felépítésének valószínűségét, abban az esetben, amikor a szolgáltatásminőség igényeket figyelembe vevő u.n., constrained shortest path first (CSPF) algoritmus nem képes megfelelő utat találni. Törekedve a hálózatban eszközlendő beavatkozások minimalizálására, javasoltam, egy új algoritmust, amely az MPLS hálózatban *egy LSPt helyez át* az új LSP felépítésének érdekében [C5]. Vizsgálatomat azért szükítettem az egy LSP áthelyezhetőségének eldöntésére, hogy az algoritmus futási idejét minimalizáljam. Habár több LSP áthelyezése növelhetné az algoritmus sikerességi valószínűségét, a nagyobb komplexitás miatt nem feltétlenül éri meg ezt vizsgálni.

A javasolt algoritmus két fő részből áll. Az *első "szűrő" rész* célja áthelyezésre esélyes LSPk kijelölése, míg a *második "két-út" rész* egy egészértékű lineáris programozáson (ILP) alapuló megoldást használ annak eldöntésére, hogy a kiválasztott LSP és az új LSP egyidőben felépíthető-e. Az ILP feladat megoldása időigényes lehet, ezért a szűrő rész fontos szerepet játszik az *algoritmus futásidejének csökkentésében*. Egyszerű szűrők segítségével, ez a rész az LSPk jelentős részéről eldönti, hogy azokra nem érdemes végrehajtani a "két-út" függvényt.

Az 1-es ábrán az optimalizáló algoritmus folyamatábrája látható. Ha a "két-út" függvény talál megoldást (azaz létezik útvonal a már bentlévő és az új LSP részére is), az optimalizálás készen van, tehát az algoritmus véget ér. Abban az esetben, ha nem volt áthelyezhető régi LSP, az algoritmus sikertelenül felyeződik be.

Szűrők

El kell kerülni azokat az LSP áthelyezési próbálkozásokat, amelyek nem segítik az új LSP felépítését. Következésképpen az algoritmus *első része* csökkenti azon LSPk számát,



1. ábra: Az optimalizáló algoritmus folyamatábrája

amelyeket a második ILP alapú funkcióval meg kell vizsgálni. Erre a célra három szűrőt javasoltam:

1. Olyan *szűk keresztmetszetű* élek vizsgálata, amelyeken nincs elegendő szabad sávszélesség az új LSP számára. A szűrő először a Dijkstra legrövidebb út algoritmussal meghatározzal az új LSP számára a szűk keresztmetszetet jelentő élek minimális számát, az alábbi költségfüggvényt haszálva:

$$w(e) = \begin{cases} 1 & \text{ha az } e \text{ élen nincs elég szabad sávszélesség} \\ & \text{az új igény számára,} \\ 0 & \text{egyébként.} \end{cases}$$

   Ezek után, kiszűrésre kerülnek azok az LSPk, amelyek kevesebb szűk keresztmetszetű élet tartalmaznak, mint a legrövidebb út súlya a fönti súlyfüggvényt használva. Ezen LSPk felszabadított útján, minimum egy élen nem lesz elég sávszélesség az új LSP számára, tehát áthelyezésüknek vizsgálata szükségtelen.

2. Azok a szűk keresztmetszetű élek, amelyek közvetlenül (azaz egy másik szűk keresztmetszetű él érintése nélkül) elérhetők az új LSP kezdő csomópontjából, úgynevezett *első szűk élek*. Az áthelyezett LSPnek tartalmaznia kell egy első szűk élet. Az első szűk élek halmazának meghatározásához az alábbi algoritmust javasoltam:

   (a) Keressük a legrövidebb utat az LSP kezdő és végpontja között ugyanolyan súlyfüggvényt használva mint fönt.

   (b) A megtalált úton keressük az első szűk élet, és átmenetileg töröljük azt a gráfból, valamint eltároljuk azt egy listában.

(c) Ismételjük az első két lépést, amíg adódik út az LSP igény kezdő és végpontja között.

Amikor minden lehetséges első szűk élet megtaláltunk, azokat a felépített LSPket amelyek nem használnak első szűk élet kiszűrjük a többi közül.

3. Ellenőrizzük, hogy egy már felépített LSP-t lebontva, annak szűk keresztmetszetét jelentő élén, felszabadul-e elég kapacitás az új LSP kiépítéséhez. Ez a szűrő kizárja annak lehetőségét, hogy olyan LSP-t helyezzünk át, ami még akkor sem segíti az új LSP felépítését, ha teljesen kivesszük a hálózatból.

Mivel minden szűrő csökkenti azon LSP-k számát, amiket a "két-út" algoritmus majd vizsgálni fog, a szűrők végrehajtási sorrendje tetszőlegesen választható.

Két-út

Az LSP-áthelyezés feladatának megoldásához meghatároztam a `két-út` alfeladatot. Először a gráf élével feltöltünk három halmazt, a következő szabályokat használva: az E_1 halmaz élein van elég kapacitás az első út elvezetéséhez, az E_2 halmaz élein pedig a második út elvezetéséhez (a két útnak lehet más más sávszélesség igénye). Az E_b halmaz élei nem képesek a két út együttes elvezetésére. A `két-út` algoritmus keres két P_1 és P_2 utat, s_1 és t_1 valamint s_2 és t_2 csomopontok között, úgy, hogy $P_1 \subseteq E_1$, $P_2 \subseteq E_2$ valamint $P_1 \cap P_2 \cap E_b = 0$, ami annyit tesz, hogy a P_i út csak az E_i halmaz éleit használhatja, továbbá, azok az élek amelyeket mindkét út használ, nem szerepelhetnek az E_b halmazban.

Megmutatható, hogy már ez az alapfeladat is NP-nehéz. Javasoltam egy módszert ami a következő *egészértékű lineáris program* felíráson alapszik:

Keressük x_1 és x_2 bináris vektorokat, amik teljesítik az alábbi feltételeket:

$$x_1^e \in \{0,1\} \qquad \forall e \in E_1 \tag{1a}$$

$$x_2^e \in \{0,1\} \qquad \forall e \in E_2 \tag{1b}$$

$$\sum_{e \in \rho(v) \cap E_1} x_1^e - \sum_{e \in \delta(v) \cap E_1} x_1^e = \delta_{v,t_1} - \delta_{v,s_1} \qquad \forall v \in V \tag{1c}$$

$$\sum_{e \in \rho(v) \cap E_2} x_2^e - \sum_{e \in \delta(v) \cap E_2} x_2^e = \delta_{v,t_2} - \delta_{v,s_2} \qquad \forall v \in V \tag{1d}$$

$$x_1^e + x_2^e \leq 1 \qquad \forall e \in E_b, \tag{1e}$$

közelítve

$$\min \sum_{e \in E_1} c_1(e) x_1^e + \sum_{e \in E_2} c_2(e) x_2^e \tag{2}$$

ahol $\rho(v)$ és $\delta(v)$ a v csomópont bejövő és kimenő éleit tartalmazó halmaz, valamint $\delta_{i,j}$ 1 vagy 0 értéket vesz fel attól függően, hogy i egyenő-e j-vel vagy nem. A két költségfüggvény, $c_1, c_2 : E \longrightarrow \mathbf{R}_+$ az élekhez van rendelve. Az x_i^e változó mutatja, hogy az i út tartalmazza-e e élet vagy nem.

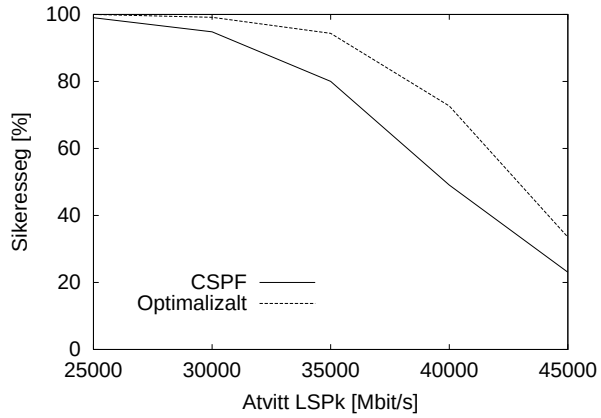
A felírt ILP feladat hatékony megoldásához használható bármilyen ILP szoftver csomag. Tesztjeimben az `lp_solve` szoftvert használtam [17], ami jó eredményt adott, annak ellenére, hogy a feladat NP-nehéz. Az `lp_solve` csomag a *simplex módszert* használja az LP feladat megoldására.

Tézis 1.2 A javasolt LSP áthelyező algoritmus teljesítmény vizsgálata

A javasolt LSP útvonalválasztó algoritmus teljesítményét szimulációval vizsgáltam [C5]. A szimulációkhoz az AT&T IP gerinchálózati topológiáját használtam [18], ami 25 csomópontból és 88 élből áll, ami 3.52-es átlagos csomóponti fokszámot jelent. A hálózati forgalmat random generált forgalmi szituációkkal állítottam elő és egy adott terheltségi szintnél (amit az *össz átvitt forgalommal* jellemeztem, azaz az összes felépített LSP sávszélesség összegével) összehasonlítottam a javasolt algoritmust az alap CSPF algoritmussal, ami blokkolja az LSP felépítést, abban az esetben, amikor nem áll rendelkezésre elég erőforrás a hálózatban.

Megmutattam, hogy a vizsgált hálózaton a javasolt algoritmus:

- nagy mértékben javítja az új LSP felépítési valószínűségét (Lsd. 2. ábra);
- gyakorlatilag használható hálózatméretekre nagyon jól skálázható, bár az alap feladat NP-nehéz. Az algoritmus futásideje 4.6 másodperc volt 150 csomóponttal rendelkező hálózat esetében, és 8.2 másodperc 200 csomópont esetében, egy Sun UltraSparc 5-ös munkaállomást és az `lp_solve` csomagot használva.



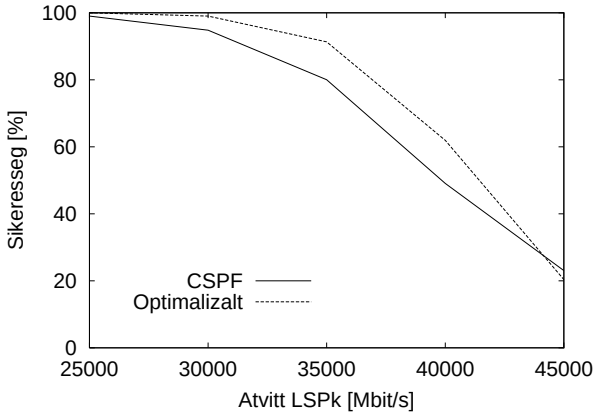
2. ábra: LSP-felépítés sikerességének valószínűsége az új és az alap CSPF algoritmus esetében, azonos kezdeti állapot esetén

Megállapítottam, hogy a javasolt szűrők nagyban csökkentik az algoritmus futási idejét. Megmutattam, hogy az algoritmus futási ideje tovább csökkenthető, ha az ILP feladat megoldásánál a *oszlop generálásos eljárást* használjuk [16] az `lp_solve` csomag által használt a szimplex módszer helyett.

Az előző vizsgálatokban a kívánt hálózati terheltségszint eléréséhez az LSPk felépítésénel a CSPF algoritmust használtam. Az algoritmus használhatóságát ellenőriztem úgy is, hogy megvizsgáltam, jelentős marad-e a teljesítményjavulás, ha minden LSP felépítéséhez már a szimulációs vizsgálatok legelejétől az új LSP áthelyező algorimust használjuk. Ebben az esetben, minden sikertelen LSP-felépítésnél megkíséreltünk áthelyezni egy már felépített LSPt. Ez hosszabb átlagos úthosszakat eredményez, vagyis, egy adott átvitt forgalmi szintnél magasabb lesz az átlagos hálózati élterheltségszint. Ebből azt várjuk, hogy különböző kezdeti állapot esetében a sikeres felépítés valószínűsége kicsit alacsonyabb lesz, az azonos kezdeti feltételek esetéhez képest.

Arra a következtetésre jutottam, hogy a vizsgált hálózaton az algoritmus:

- lényegesen magasabb sikerességi valószínűséget eredményez az alap CSPF módszerhez képest, mérsékelt hálózati terheltségszintek esetében (Lsd. 3. ábra);
- a teljesítménykülönbség csökken magasabb átlagos élterheltségek esetében (pl, 37000Mbit/másodperc össz átvitt forgalom esetében, ami 60% átlagos élterheltséget jelent), amikor az áthelyezés magasabb átlagos úthosszakat eredményez;



3. ábra: LSP-felépítés sikerességének valószínűsége az új és az alap CSPF algoritmus esetében, különböző kezdeti állapot esetén

Ezenfelül végeztem még szimulációs vizsgálatokat annak eldöntésére, hogy mennyi teljesítményjavulást eredményezne, ha az algoritmus elosztott módon az útválasztók szoftverében lenne megvalósítva. Ezen alkalmazás esetében azon LSPk áthelyezése válik lehetővé, amelyeknek egyezik a kezdő csomópontja az új felépítendő LSP-vel. A szimulációs vizsgálatok eredményei azt mutatták, hogy még ebben az esetben is körülbelül

2-3%-al növelhető az új LSP felépítésének valószínűsége. Ez viszont csak marginális hasznot jelent, így az algoritmus elosztott megvalósításának kisebb gyakorlati jelentősége van.

Tézis 2 Lergövidebb út alapú helyreállítási módszer

Egy adott helyreállítási módszer tervezésénél fontos figyelembe venni a üzemeltetési elvárásokat. Gerinchálózati szolgáltatók esetében az egyik legfontosabb elvárás, a szolgáltatás kimaradási idejének minimalizálása, és például 99.99%-os rendelkezésreállás elérése. Ez gyors helyreállítási módszert igényel, ami jól skálázható akkor is, amikor egy egyszeres élszakadás következtében több száz, vagy ezer összeköttetés helyreállítása szükséges. Mivel a gerinchálózatok állapotának rendkívül stabilnak kell lennie, a szolgáltatók alapvető érdeke, hogy minimalizálják a hálózat átkonfigurálását, valamint, hogy miután a hibás él újra üzemi állapotba kerül, egyszerűsítsék a visszaállítási ciklust.

Más szerzők [19] megmutatták, hogy azok a helyreállítási módszerek, amik előzetes útszámítást használnak, de nem építik fel előzetesen a védelmi utakat, gyors helyreállítást tesznek lehetővé, ugyanakkor jó erőforrás-kihasználtságot is garantálnak. Az optimális erőforrás kihasználtság eléréséhez fontos, hogy a lehető legrövidebb utakat használjuk egy adott igény elvezetéséhez. Habár a legrövidebb üzemi út és hozzá a legrövidebb védelmi út mehatározásának feladata NP-nehéz, ezt egészértékű lineáris program formájában felírva jó gyakorlati eredményeket kaphatunk, ahogy mások ezt [20] megmutatták. Viszont, ha eltekintünk attól a feltételtől, hogy csak egy védelmi utat használhatunk az üzemi út hibájának védelmére, egy új még megoldatlan feladatosztályt kapunk.

Tézis 2.1 Legrövidebb védelmi út algoritmus

Működési elvek egy halmazát javasoltam egy új MPLS védelmi út számolási módszer számára [C8]:

1. A védelmi utak előre kiszámítottak, de nincsenek előre felépítve;
2. Az üzemi út és a nemvisszahelyezendő védelmi út egyaránt legrövidebb utak;
3. Az üzemi út és a nemvisszahelyezendő védelmi út közös éleinek száma minimális számú legyen;
4. Ha az üzemi út és a nemvisszahelyezendő védelmi út él diszjunkt, nincs szükség több védelmi útra;
5. Ha az üzemi út és a nemvisszahelyezendő védelmi út rendelkezik közös élekkel, még egy út kerül kiszámításra, ami a közös éleken bekövetkező hibák ellen nyújt védelmet. Ezt az utat visszahelyezendő védelmi útként használjuk.

Ezen elvárások között a második jelenti az alapvető újdonságot, az ötödik következmény, míg a többi azert szükséges, hogy a környezet tökéletesen meg legyen határozva.

A fönti működési elveken alapuló algoritmus új, mivel minimalizálja a védelmi utak erőforrás használatát, ráadásul, növeli annak a valószínűségét, hogy nemvisszahelyezendő utak konfigurálhatók.

Kereskedelmi forgalomban kapható útválasztókban (pl. [21]) lehetőség van egy üzemi úthoz rendelten több védelmi út tárolására, mi több, ezek visszahelyezendő és nemvisszahelyezendő módon is konfigurálhatók. Az utak előre kiszámításának az az értelme, hogy így gyorsabb helyreállítási idő érhető el, mint hibaészlelés utáni útszámítás esetében. A második nemvisszahelyezendő védelmi út számolását tovább lehetne optimalizálni, de meggyőződésem szerint, ennek csak korlátozott hatása lenne a legfontosabb mérőszámokra, például az átlagos védelmi úthosszra, valamint annak a valószínűségére, hogy a védelmi út valóban legrövidebb út-e. Ezért csupán azt az elvárást fogalmaztam meg a harmadik úttal szemben, hogy az legyen élfüggetlen az üzemi és a nemvisszahelyezendő védelmi úttól.

A fönti elvárásokat kielégítő algoritmus tervezése során két alfeladatot határoztam meg. Az egyik lényege *két legrövidebb út meghatározása, minimális számú közös éllel* (1-es alfeladat, a 2. és a 3. feltétel teljesítésére), míg a másik lényege *harmadik út meghatározása, ami teljes védelmet biztosít az üzemi út számára* (2-es alfeladat, az 5. feltétel teljesítésére). Az egyes alfeladatokra javasolt algoritmusok leírása alább található.

A következő algoritmust javasoltam két, minimális közös élt tartalmazó, legrövidebb út meghatározására, (1-es alfeladat) [J1], [C8]:

1. A Dijkstra legrövidebb út algoritmust futtatva egy rögzített s csomópontból, határozzuk meg a minimális $\pi_{min}(v)$ úthosszat a gráf minden egyes v csomópontjára;

2. Töröljük ki az összes olyan uv élet $\in E$, amire $\pi_{min}(v) - \pi_{min}(u) \neq w(u,v)$;

3. A megmaradó éleket duplázzuk meg, '0' költséget rendelve az egyikhez, és '1'-et a másikhoz;

4. Az így módosított gráfon, a két-értékű egységkapacitású többtermékes minimális folyamprobléma algoritmussal (amit néha 'Suurballe módszer' elnevezéssel hivatkoznak) határozzunk meg két minimális összköltséggű P_1, P_2 utat.

A minimális folyamprobléma megoldásaként adódó két út az eredeti gráfban legrövidebb út lesz, minimális számú közös éllel. Abban az esetben, ha a két út teljesen élfüggetlen, az egyik lesz az üzemi út, a másik pedíg a nem-visszaállítandó védelmi út. Ebben az esetben nincs szükség újabb védelmi útra. Azonos utak esetén az egyiket megtartjuk, mint üzemi út, a másikat kitöröljük.

A harmadik utat, ami a garantált védelmet jelenti az üzemi út bármely élének meghibásodása ellen, az alábbi algoritmussal számítjuk (2-es alfeladat) [C8]:

1. Induljunk ki az eredeti gráfból;

2. Rendeljünk az előző algoritmus által talált két legrövidebb út közös éleihez $\mid P_1 \cap P_2 \mid$, olyan $w(e)$ súlyt, ami nagyobb, mint a gráf élsúlyösszege. Ezáltal, amennyiben az előző algoritmus által adott két út teljesen egyforma volt, az üzemi út minden egyes élén emelt súly fog szerepelni;

3. Határozzuk meg a minimális s,t utat a Dijkstra legrövidebb út algoritmus segítségével;

Tézis 2.2 A javasolt helyreállítási algoritmus teljesítményvizsgálata

A javasolt helyreállítási útvonalszámító algoritmus teljesítményének vizsgálatához szimulációkat végeztem [C8]. A vizsgálatokban referenciaként, két, az irodalomban [14] használt algoritmust használtam, névszerint a 'Suurballe' valamint a 'Penalty' módszert.

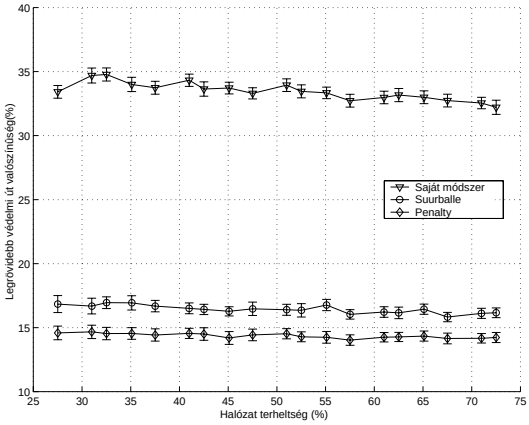
Az első vizsgálatban, melynek célja a hálózat a karakterizálása volt, alap mérőszámokat határoztam meg a Cable & Wireless és az AT&T IP hálózati topológiákra [18]. Megállapítottam, hogy:

- az algoritmus a hálózat csomópont párjainak 35-40%-hoz talált két legrövidebb utat, míg ez hagyományos algoritmusokkal csupán 15-20% volt; ezenfelül,
- a talált utak több mint a felénél csupán egy közös él adódott, ami azt jelenti, hogy elég nagy annak valószínűsége, hogy legrövidebb út használható védelmi útként. Ez nemvisszahelyezendő védelmi útként érhető el.

Egy másik vizsgálatban, a különböző módszerek teljesítmény tényezőinek mérésére az AT&T hálózati topológiát használtam, random forgalommal. Megállapítottam, hogy:

- a javasolt védelmi út számító algoritmus átlagos védelmi úthossz mérőszáma megközelítőleg 8-9%-al alacsonyabb a referenciaként használt algoritmusok átlagos úthosszánál (konzisztensen minden vizsgált hálózat terhelési szintnél).
- a javasolt algoritmusokat használva, megduplázódik annak a valószínűsége, hogy a helyreállításra legrövidebb út használható (Lsd. 4. ábra). Ez azt jelenti, hogy meghibásodásnál, az LSP áthelyezés során a hálózati erőforrások jóval hatékonyabban használhatók ki, ráadásul, miután a hibát kijavították, ezeket az LSPket nem szükséges visszahelyezni az eredeti útjukra.

Utolsó vizsgálatomban különböző csomópontszámú és élsűrűségű random hálózatokkal ellenőriztem előző kísérleteim eredményeinek helyességét. Az találtam, hogy a hálózat csomópontszámának növelésével a javasolt algoritmus teljesítménye javul. Hasonlóan, amikor azonos csomópontszám mellett az élszámot növeltem, az algoritmusok közti különbség egyre markánsabban jelentkezett.



4. ábra: Legrövidebb védelmi út használatának valószínűsége

Tézis 3 Folyamkiszorítást és sávszélességigényt figyelembe vevő útvonalválasztás MPLS hálózatokban

MPLS hálózatokban az egyértelmű (explicit) útvonalak meghatározása érdekében a szolgáltatásminőség igényeket figyelembe vevő u.n., constrained shortest path first (CSPF) algoritmusnak szüksége van az aktuális élterheltségszintek ismeretére. Az útválasztó protokollok forgalomfolyam elvezetéssel kapcsolatos (u.n. traffic engineering) kiterjesztései szétküldik a hálózat összes éléről a B_{max} maximális foglalható sávszélesség információt és a $\mathbf{B}_u = (B_{u0}, B_{u1}, B_{u2}, \cdots, B_{u7})$ még le nem foglalt (azaz szabad) sávszélesség értékeket. Az utóbbiban az MPLS-ben definiált, szolgáltatás diferenciálás támogatására használt prioritás szintek miatt jelentkezik nyolc B_u érték. Annak érdekében, hogy az útvonalválasztás figyelembe vegye a sávszélességigényt, a hálózatszéli útválasztók CSPF algoritmusa kiszűri a $B_{us} < B_{LSP}$, képlet alapján nem megfelelőnek bizonyuló éleket, ahol B_{LSP} az LSP sávszélesség igénye és s a prioritás szintje. Az így kapott gráfon végzett útvonalválasztás során, minden ismert CSPF algoritmus csakis az újonnan behúzandó LSP szintjén veszi figyelembe a szabad sávszélesség információt, és nem törődik az alsóbb szintek szabad sávszélesség információjával. Habár az útfelépítés során az alacsonyabb prioritású LSPk kiszorításra kerülhetnek, az útvonalszámítás során az ismert CSPF algoritmusok ezt nem veszik figyelembe.

Tézis 3.1 Kiszorítási mértékek

Javasoltam, az alacsonyabb prioritás szintek szabad sávszélesség információjának figyelembe vételét az LSP útvonalának kiszámítása során [J4], [C7]. Ennek érdekében meghatároztam két, a terjesztett él-állapot információkból származtatható kiszorítási mértéket, melyek a következők:

1. *szabad sávszélesség*, azaz, a legalacsonyabb (7-ik) prioritásszint le nem foglalt sávszélesség:

$$B_{free} = B_{u7} \tag{3}$$

2. *sávszélesség kiszorítási vektor*:

$$\mathbf{B}_p = (B_{p0}, B_{p1}, B_{p2}, \cdots, B_{p7}) \tag{4}$$

   ez egy olyan összetett mérték, ami tartalmazza az él egyes prioritás szintjein kiszorítandó, becsült sávszélességet. E mérték segítségével meghatározható például, hogy melyik az élen a *legmagasabb érintett prioritásszint*, azaz a legkisebb olyan i, amire $B_{pi} \neq 0$. Az 5. ábrán javaslok egy eljárást a sávszéleség kiszorítási vektor kiszámítására.

```
procedure CalcBwPreemptionVector(B_u, B_LSP)
   B_p = 0
   B_u(-1) = B_max
   for (i = 7, i ≥ 0, i--)
       if B_LSP ≤ B_ui return B_p
       B_pi = min((B_LSP − B_ui), (B_u(i−1) − B_ui))
   end for
   return B_p
end procedure
```

5. ábra: A sávszéleség kiszorítási vektor számítása

Tézis 3.2 A kiszorítási mértékek használata útvonalválasztó algoritmusokban

A 3.1 tézisben kidolgozott kiszorítási mértékeket felhasználva javasoltam egy új folyamkiszorítást és sávszélességigényt figyelembe vevő CSPF algoritmust [J4], [C7].

A *szabad sávszélességmértékhez* javasoltam, hogy a Dijkstra algoritmus során, egy út metrikájaként, az út szűk keresztmetszetét jelentő él szabad sávszélesség mértékét használjuk. Két út metrikájának összevetésénel javasoltam, hogy a nagyobb szabad sávszélességgel rendelkező utat tekintsük jobb alternatívának.

A *sávszélesség kiszorítási vektor* esetében, két él mértékének akkumulálásánál, javasoltam, hogy a nagyobb $\mathbf{B}_p$ vektort használjuk az út metrikájaként. Két út sávszélesség kiszorítási vektorának összevetésénél, lexikografikus összehasonlítást javasoltam.

Javasoltam, hogy minden kiszorítást figyelembe vevő algoritmus először végezze el az élek redukálását a $B_{us} < B_{LSP}$ képlet alapján. Az eredményül kapott gráfban javasoltam, az útvonalálasztás leszűkítését az OSPF metrika alapján adódó legrövidebb utakra. Javaslatom szerint a folyamkiszorítási információk használata az egyébként legrövidebb utak közötti legjobb út kiválasztása során történne. Így, a kiszorítást minimalizáló metrikák nem érintenék hátrányosan a CSPF algoritmus LSP felépítési sikerességét, valamint a magas prioritású LSPk úthosszát. A kiszorítást figyelembe vevő CSPF algoritmusok különböző lehetőségei az 1 táblázatban kerültek összefoglalásra.

1. táblázat: A javasolt algoritmusok metrikái és azok sorrendjei

	első metrika	második metrika	harmadik metrika
1	legrövidebb	szabad sávszélesség	—
2	legrövidebb	sávszélesség kiszorítási vektor	—
3	legrövidebb	sávszélesség kiszorítási vektor	szabad sávszélesség

Tézis 3.3 A javasolt kiszorítást és sávszélességigényt figyelembe vevő útvonalválasztó algoritmus teljesítményvizsgálata

A javasolt kiszorítást és sávszélességigényt figyelembe vevő útvonalválasztó algoritmus teljesítményének vizsgálatához szimulációkat végeztem [J4], [C7]. A vizsgálathoz a 31 csomópontot és 102 élet tartalmazó, 3.29-as átlagos csomóponti fokszámú *Cable & Wireless* gerinchálózati topológiát használtam. A forgalmat random generáltam. Ezt úgy tettem, hogy a hálózatot egy bizonyos mértékig feltöltöttem random LSP igényekkel, melyeknek random sávszélesség igényük volt. Az LSP sávszélességek az $(0, B_{MaxLSP}]$ intervallumban egyenletesen lettek generálva. A kísérletben olyan esetekre koncentráltam, amikben a B_{MaxLSP} a legjellemzőbb OC-12-es élkapacitás 8-10%-a volt, ami összhangban van Rexford feltételezésével [7]. Az LSP prioritás szinteket is random módon állítottam be, a [0-7] tartományban egyenletes eloszlás szerint.

Referencia algoritmusként különböző, az irodalomban fellelhető terhelésmegosztó algoritmusokat választottam (névszerint a 'widest-shortest path' módszert [6], the 'discrete link cost' módszert [7] és a 'residual bandwidth ratio' módszert [8]).

Megállapítottam, hogy:

- a javasolt kiszorítást és sávszélességigényt figyelembe vevő útvonalválasztó algoritmus, sikerességét terkintve nagyon közel teljesített (1-2%-os eltérésen belül) a legjobb CSPF algoritmushoz, az átlagos hálózati terheltségtől függetlenül;

- a javasolt kiszorítást minimalizáló módszerekkel a kiszorítás valószínűsége jóval alacsonyabb. Az érintett prioritás szintek számát minimalizáló stratégia hatékonyabb, mint az egyszerűbb, ami az út szűk keresztmetszetű élének szabad kapacitását minimalizálja. Magas terheltségi szinteknél az előbbi 10%, míg a második 5% javulást ért el a különböző terhelésmegosztó algoritmusokhoz képest;

- a kiszorítási láncban érintett LSPk száma nem különbözött jelentősen más-más módszerek esetében.

Általánosabb eredmények levonása végett, random hálózatokon is végeztem szimulációkat, változtatva a hálózat csomópontszámát és az átlagos csomoponti fokszámot. A vizsgálat hasonló eredményeket mutatott a valós hálózati szimulációkhoz. A javulás mértéke a 3-11% sávban mozgott. Azt találtam, hogy a csomópontszám és az átlagos fokszám növelésével, a kiszorítást figyelembe vevő CSPF technikákkal elérhető teljesítményjavulás egyre növekszik.

Tézis 4 Architektúra valós útvonalválasztó algoritmus megvalósítások tesztelésére

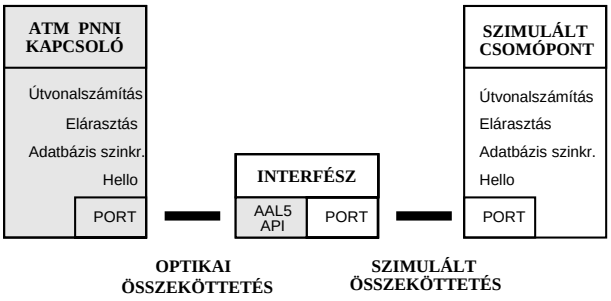
Az útvonalválasztó algoritmusok szimulációs vizsgálata fontos szerepet játszik adat gerinchálózatok teljesítmény analízisében. Amikor kiválasztanak egy konkrét algoritmust és azt megvalósítják útvonalválasztók szoftverében, a gyártók alapvető érdeke, hogy az új modul alapos tesztelésen menjen keresztül. Ezért a szimulációs vizsgálatok mellett konformancia és skálázhatósági teszteket is el kell végezni. A kód funkcionális működésének specifikáció szerinti ellenőrzéséhez, azaz a konformancia teszteléshez, egyszerű teszthálózatok is elegendők. A skálázhatósági tesztekhez ellenben, nagy teszthálózatok szükségeltetnek. Lehetnek azonban olyan szituációk, amikor nem léteznek már megépített, elegendően nagy hálózatok, vagy amik léteznek, kereskedelmi szolgáltatás nyújtására már igénybe vannak véve.

Javasoltam egy emulátor architeckúrát [C3] PNNI (private network-to-network interface) protokoll [5] megvalósítások teljesítményvizsgálatára. Ennek segítségével a tesztelt berendezés (implementation under test – IUT) egy szimulált hálózat részeként vizsgálható, minek következtében annak úgy kell működnie, mintha egy valós hálózat részét képezné. Íly módon a teszt konfiguráció létrehozható a szimulátor konfigurálásán (azaz bemeneti file-ok szerkesztésén) keresztül, ahelyett, hogy valós berendezésekből álló, több tucat összekapcsolással rendelkező környezetet kéne felállítani.

Meghatároztam az architektúra legfontosabb komponenseit:

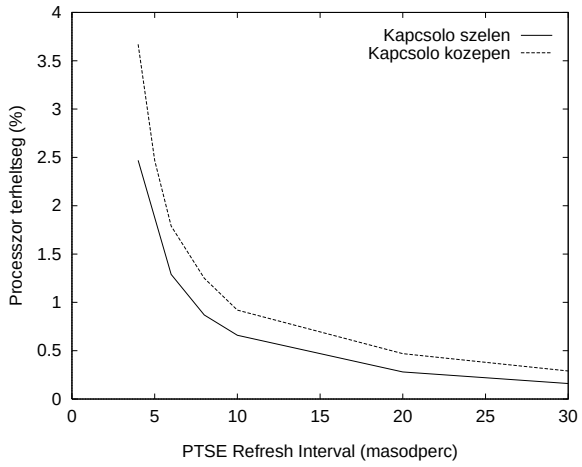
- a szimulátornak egy munkaállomáson kell futnia;
- a szimulátornak részletesen kell modelleznie a PNNI útválasztó és jelzés protokollt: a Hello, az adatbázis szinkronizáció és az elárasztás funkciókat a PNNI specifikáció szerint kell megvalósítani [5];
- egy hatékony topológia adatbázis megvalósítás és a hálózat állapotát leíró absztrakt modell segítségével, a szimulátornak lehetővé kell tennie egyrészt a memória hatékony kihasználását, másrészt a protokoll üzenetkezelés egyszerűsített megvalósítását;
- a szimulátort futtató munkaállomás, ATM AAL5 üzenetek küldésére és fogadására alkalmas ATM kártyával kell felszerelni;
- a szimulátorban meg kell valósítani egy interfész funkciót, ami átkonvertálja a szimulált üzeneteket bit-helyes PNNI jelzésekké és viszont;
- a szimulátornak valós időben kell futnia, az eseményvezérelt mód helyett;
- az architektúrának lehetővé kell tennie az IUT és az emulátor több éllel (routing control channel-el) való összekapcsolását.

A 6. ábrán egy kétcsomópontos hálózat funkcionális ábrája látható, úgy, hogy egy valós ATM kapcsoló van összekötve, egy másik kapcsolót modellező emulátorral.



6. ábra: Egy kétcsomópontos hálózatot modellező emulátor funkcionális ábrája

A javasolt emulátor architektúra alapján megvalósítottam egy emulátor prototipust. A megvalósítás alapját egy rendelkezésre álló szimulátor szoftver jelentette [22]. Az emulátor architektúra prototipus megvalósításának segítségével elvégeztem egy kereskedelmi forgalomban kapható ATM kapcsoló teljesítmény vizsgálatát. A cél a javasolt architektúra használhatóságának ellenőrzése volt. A kísérletben a processzorterhelést vizsgáltam különböző mennyiségű üzenettel való elárasztás függvényében. Az él-terheltség információt tartalmazó protokoll csomagok gyakori terjesztését úgy oldottam meg, hogy az emulált csomópontok 'PTSE Refresh Interval' beállításait változtattam. A 7. ábra eredményei kétféle hálózati konfigurációval adódtak; az egyikben az IUT egy éllel kapcsolódott a hálózathoz, míg a másikban két élet használtam, így az IUT-t úgy tudtam tesztelni, mintha egy valóságos hálózat közepén működne. A kapott eredmények magukban nem jelentenek tudományos újdonságot, hiszen nem is ez volt a cél, viszont segítségükkel megmutattam, hogy az emulátor architektúra alkalmas kereskedelmi forgalomban kapható ATM kapcsolók, PNNI protokoll megvalósításának tesztelésére.



Beállítások

Kapcsoló és Emulátor:
MinPTSEInterval: 0.1s,
PTSELifetimeFactor: 101%,
PeerDelayedAckInterval: 1s.
Kapcsoló:
PTSERefreshInterval: 30s,
Emulator:
PTSERefreshInterval:
30 másodpercről csökkent 4 másodpercre.

7. ábra: Egy, az emulátorral összekötött, valós kapcsoló processzor terheltsége

5. Az eredmények alkalmazása

Az útvonalválasztó algoritmus optimalizálásával és optimális folyamelvezetéssel kapcsolatos kutatásomat részben az Ericsson vállalat üzleti egységei, részben az Ericsson kutatási szervezetének megbízásából felállított projektek részeként végeztem. A disszertációban javasolt és vizsgált új útvonalválasztó algoritmusok támogathatják Ericsson termékek jövőbeli fejlesztését. Az új és meglévő algorimusok teljesítményvizsgálati eredményei támágathatják az Ericsson üzleti egységeit az útválasztók szoftvereiben megvalósítandó legmegfelelőbb algoritmusok kiválasztásában.

A harmadik tézisben ismertetett, folyamkiszorítást és sávszélességigényt figyelembe vevő útvonalválasztási algoritmus szabadalmaztatásra került.

A negyedik tézis emulátor architektúrája megvalósításra került egy prototípus alkalmazásban, és ennek tesztkörnyezetbeli vizsgálata is megtörtént egy Cisco ATM kapcsoló segítségével.

Irodalom

[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja and X. Xiao, *A Framework for Internet Traffic Engineering*, Internet Engineering Task Force, Internet Draft, April 2001, Work in Progress.

[2] D. Awduche, L. Berger, D.H. Gan, T. Li, G. Swallow, V. Srinivasan, *RSVP-TE: Extensions to RSVP for LSP Tunnels*, Internet Engineering Task Force, Request For Comments (Proposed Standard) 3209, December 2001.

[3] B. Jamoussi (editor), *Constraint-Based LSP Setup Using LDP*, Internet Engineering Task Force, Request For Comments (Proposed Standard) 3212, January 2002.

[4] D. Katz, D. Yeung and K. Kompella, *Traffic Engineering Extensions to OSPF*, Internet Engineering Task Force, Internet Draft, March 2001, Work in Progress.

[5] ATM Forum Technical Committee: *Private Network-Network Interface Specification Version 1.0*, ATM Forum af-pnni-0055.000, March 1996.

[6] G. Apostolopoulos, R. Williams, S. Kamat, R. Guerin, A. Orda, A. Przygienda, *QoS Routing Mechanisms and OSPF Extensions*, Internet Engineering Task Force, Request For Comments (Proposed Standard) 2676, August 1999.

[7] A. Shaikh, J. Rexford and K.G. Shin, *Evaluating the impact of stale link state on quality-of-service routing*, IEEE/ACM Transactions on Networking, vol.9, no.2, pages 162–176, April 2001.

[8] K. Kompella and D. Awduche, *Notes on Path Computation in Constraint-Based Routing*, Internet Engineering Task Force, Internet Draft, September 2000, Work in Progress.

[9] V. Sharma et al, *Framework for MPLS-based Recovery*, Internet Engineering Task Force, Internet Draft, March 2001, Work in Progress.

[10] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, *Requirements for Traffic Engineering Over MPLS*, Internet Engineering Task Force, Request For Comments (Proposed Standard) 2702, September 1999.

[11] P. Aukia, M. Kodialam, P.V. Koppol, T.V. Lakshman, H. Sarin, B. Suter, *RATES: A server for MPLS traffic engineering*, IEEE Network, vol.14, no.2, pages 34–41, 2000.

[12] F. P. Kelly, *Bounds on the performance of dynamic routing schemes for highly connected networks*, Mathematics of Operations Research, Vol.19, No.1, February 1994.

[13] G. R. Ash, *Dynamic Routing in Telecommunications Networks*, chapter 8.2 "Real-Time Dynamic Routing Methods", pp 299–308, ISBN 0-07-006414-8, McGraw-Hill, 1997.

[14] A. Iwata, R. Izmailov, B. Sengupta, *Alternative Routing Methods for PNNI Networks with Partially Disjoint Paths*, IEEE Globecom 98, November 1998.

[15] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, New Jersey 07458. 1993.

[16] A. Schrijver, *Theory of Linear and Integer Programming*, Jowh Wiley&Sons, Wiley-Interscience series in discrete mathematics, 1986.

[17] M. Berkelaar, J. Dirks, *lp_solve 2.2*,
ftp://ftp.es.ele.tue.nl/pub/lp_solve

[18] Russ Haynal's ISP Page,
http://navigators.com/isp.html

[19] R. Doverspike, J. Yates, *Challenges for MPLS in Optical Network Restoration*, IEEE Communications Magazine, Vol. 39, No. 2, pages 89–96, February 2001.

[20] P. Laborczi, J. Tapolcai, P.-H. Ho, T. Cinkler, A. Recski and H.T. Mouftah, *Algorithms for Asymmetrically Weighted Pair of Disjoint Paths in Survivable Networks*, Third International Workshop on Design of Reliable Communications Networks, (DRCN) October 2001.

[21] A. Statti, P, Muraca, C. Morris, *JUNOS 5.0 Internet Software Configuration Guide:MPLS Applications*, Juniper Networks, Inc. 1194 North Mathila Avenue Sunnyvale, CA 94089 USA, August 2001
http://www.juniper.net/techpubs

[22] Á. Szentesi, *Routing and Topology Optimization in ATM Networks*, Ph.D thesis, Department of Telecommunication and Telematics, Budapest University of Technology and Economics, 2002.

Publications

Journal papers

[J1] Á. Magi, Á. Szentesi, B. Szviatovszki, A. Faragó: *Dinamikus útvonalválasztás ATM hálózatokban*, Híradástechnika, Vol. 50, No. 11, pages 2–11, November 1999.

[J2] Á. Magi, Á. Szentesi, B. Szviatovszki: *Analysis of link cost functions for PNNI routing*, Computer Networks, Vol. 34, No. 1, pages 181–197, July 2000.

[J3] A. Faragó, Á. Szentesi, B. Szviatovszki: *Inverse Optimization in High Speed Networks*, To appear in the Spec. Issue on Combinatorial and Algorithmic Aspects of Telecom., Discrete Applied Mathematics Journal (submitted: January 1998, reviewed: April 1999)

[J4] B. Szviatovszki, Á. Szentesi, A. Jüttner: *Minimizing Re-Routing in MPLS Networks with Preemption-Aware Constraint-Based Routing*, Computer Communications Journal, Spec. Issue on Advances in Performance Evaluation of Computer and Telecommunications Networking, Vol. 25, No. 11–12, pages 1076–1084, May 2002.

[J5] D. Orincsay, B. Szviatovszki, G. Böhm: *Prompt Partial Path Optimization in MPLS networks*, Submitted to The International Journal of Computer and Telecommunications Networking, (submitted: November 2002)

[J6] J. Forslow, I. Jarrett, P. Moran and B. Szviatovszki: *Management solutions for IP networks*, Ericsson Review. vol.77, no.1, pages 42–52. 2000

Conference and workshop papers

[C1] A. Faragó, Á. Szentesi, B. Szviatovszki: *Allocation of Administrative Weights in PNNI*, In proc. of the 8th International Telecommunication Network Planning Symposium, Networks'98, pages 621–626, Sorrento, Italy, October 18–23, 1998

[C2] Á. Magi, G. Nagy, Á. Szentesi, B. Szviatovszki: *Analysis of link cost functions for PNNI routing*, In proc. of the 6th IFIP Workshop on Performance Modeling and Evaluation of ATM Networks, pages 50/1–50/10, Ilkley, UK, July 20–22, 1998

[C3] Á. Magi, B. Szviatovszki, G. Rózsa, L. Németh: *Novel Performance Evaluation of PNNI Equipment*, In proc. of the XVII World Telecommunications Congress incorporating ISS'2000, Birmingham, UK, May 7–12, 2000.

[C4] B. G. Józsa, D. Orincsay, Á. Magi, B. Szviatovszki: *On the Use of Trunk Reservation in PNNI Routing*, In proc. of the IEEE Conference on High Performance

Switching and Routing (ATM2000), pages 135–139, Heidelberg, Germany, June 26–29, 2000.

[C5] A. Jüttner, B. Szviatovszki, Á. Szentesi, D. Orincsay, J. Harmatos: *On-demand optimization of label switched paths in MPLS networks*, In proc. of the 9th International Conference on Computer Communications and Networks, IEEE ICCCN 2000, pages 107–113, Las Vegas, USA, October 16–18, 2000.

[C6] A. Jüttner, B. Szviatovszki, I. Mécs, Zs. Rajkó: *Lagrange Relaxation Based Method for the QoS Routing Problem*, In proc. of the IEEE Infocom 2001, pages 100–109 Anchorage, Alaska, USA, April 22–26, 2001.

[C7] B. Szviatovszki, Á. Szentesi, A. Jüttner: *Minimizing Re-Routing in MPLS Networks with Preemption-Aware Constraint-Based Routing*, In proc. of the 2001 International Symposium of Performance Evaluation of Computer and Telecommunication Systems, Spects 2001, pages 249–261, Orlando Florida, July 15–19, 2001

[C8] B. Szviatovszki, Á. Szentesi, A. Jüttner: *On the effectiveness of Restoration Path Computation Methods*, In proc. of the IEEE International Conference on Communications, New York, April 28–May 2, 2002.

Patents

[P1] B. Szviatovszki, Á. Szentesi, A. Jüttner: *Priority-Based Path Selection in a Data Network Control*, (US Patent Application No. P14200US, 2001)