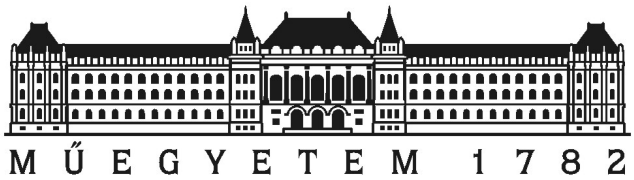


TÁVKÖZLÉSI ÉS MÉDIAINFORMATIKAI TANSZÉK
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

ÚJ MÓDSZEREK AZ IP ALAPÚ GYORS HIBAJAVÍTÁSBAN

Ph.D. tézisfüzet

Írta:
Enyedi Gábor

Tudományos konzulensek:

Dr. Rétvári Gábor
Távközlési és Médiainformatikai Tanszék

Beadva a
DOCTOR OF PHILOSOPHY
fokozat részleges követelményeként a
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM-re
BUDAPEST, MAGYARORSZÁG
2011. FEBRUÁR

1. Bevezető

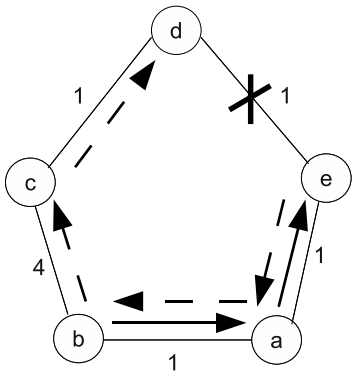
Az elmúlt évtizedekben jelentősen megváltozott életünk; a modern telekommunikációs hálózatok folyamatosan növekvő elterjedtségének köszönhetően ma már szinte bárkit azonnal elérhetünk, vagy megtalálhatunk akármilyen információt. Jelenleg egyre inkább úgy tűnik, hogy a kommunikációs hálózatokban egy közös platformot fognak használni, amely az Internet Protocol (IP) lesz.

Sajnos azonban az IP még ma is számos hiányossággal küzd, ha olyan valós idejű forgalmat kell továbbítania, mint a Voice over IP (például 3G vagy 4G mobil hálózatokban), IPTV, on-line játék, vagy a különösen kritikus tőzsdei műveletek. Ezenkívül számos vállalat bérel virtuális magánhálózatot (Virtual Private Network – VPN) komoly, szerződésben vállalt szolgáltatási minőséggel, melyet aztán IP alapú telefonálásra vagy távoli alkalmazások elérésére használnak. Mivel az IP-t elasztikus forgalom továbbítására tervezték ahol a késleltetés, vagy a meghibásodást követő helyreállás ideje nem volt igazán fontos tényező, még a mai IP hálózatoknak is nehézségeket okoz a szükséges szolgáltatásminőség (Quality of Service – QoS) teljesítése. Ismertetésre kerülő téziseim a meghibásodást követő helyreállással foglalkoznak.

Egy hálózatban kétféle helyreállási módszert alkalmazhatunk: védelmet (protection) és újjáépítést (restoration). A védelmi technikák *proaktívak*, azaz jóelőre, bármiféle erőforrás meghibásodása előtt kiszámítják az elkerülő útvonalakat. Természetesen, mivel lehetetlen felkészülni a meghibásodások minden lehetséges kombinációjára, ezektől a megoldásoktól szükségszerűen csak mérsékelt robosztusságot várhatunk, továbbá alkalmazásukkor tipikusan szuboptimális elkerülő utakkal kell beérjük. Másfelől azonban a védelmi megoldások nagyon gyorsak, hiszen esetükben a javítás csupán egy előre kiszámított alternatívára való átkapcsolást jelent.

A védelmi megoldásokkal szemben az újjáépítő technikák *reaktívak*, ami azt jelenti, hogy meghibásodás áthidalásának módját csak a hiba megjelenése után kezdik keresni. Ennek köszönhetően ezek a megoldások a számítások során minden esetben teljes információval rendelkeznek a kialakult új topológiáról, ami meglehetős robosztussággal ruházza fel őket. A megközelítés hátránya ezzel összhangban természetesen a reaktív viselkedés miatti viszonylagos lassúság.

Sajnos, mivel az IP hálózatokat tisztán elasztikus forgalom továbbítására tervezték, ezen hálózatokban a helyreállítás kizárólag olyan újjáépítő megoldásokon alapul, mint az Open Shortest Path First (OSPF) [Moy98] vagy az Intermediate System to Intermediate System (IS-IS) [fS02], védelmi technikákat pedig egyáltalán nem alkalmaznak. A védelem ezen hiánya egyre komolyabb gonddá kezd válni az IP növekvő elterjedésével, ezért számos javaslat született az IP védelmi képességekkel való felruházására; ezeket a technikákat nevezzük összefoglalóan IP alapú gyors hibajavításnak (IP Fast ReRoute – IPFRR).



1. ábra. Példa a lokális átirányításra; a számok a kapcsolatok mellett azok költségét jelölik

Mivel az IPFRR megoldások védelmi megoldások, alapelveik számottevően eltérnek a tradicionális IP alapú helyreállítás elveitől: ezek a technikák meghibásodás esetén a forgalmat *lokálisan* irányítják át egy *előre kiszámított elkerülő útra*. Ez azt jelenti, hogy mikor egy erőforrás meghibásodik és annak szomszédai detektálják a hibát, a szomszédok képesek a forgalmat átirányítani anélkül, hogy a topológia megváltozásáról más csomópontokat értesítenének; az 1. ábra ezt a megközelítést szemlélteti.

Tegyük fel, hogy b csomópont csomagot akar küldeni d-nek! Ekkor a legrövidebb út b-ből d-be $b \to a \to e \to d$. Azonban ha az összeköttetés e és d között meghibásodik, először csak a kapcsolattal szomszédos csomópontok, e és d szereznek erről tudomást. Mivel nincs idő a meghibásodás tényének terjesztésére, e lokálisan átirányítja a csomagokat. Más lehetősége nem lévén e arra kényszerül, hogy visszaküldje a csomagokat a-nak. Ha a hálózatban IPFRR működik, a nem fogja visszaküldeni őket e-nek, és nem okoz továbbítási hurkot, hanem b-nek továbbítja, amely a helyett most c-t fogja választani, majd a csomagok célba érnek d-ben. Fontos kiemelni, hogy a és b továbbra sem „tud” a meghibásodásról, nem változtatják meg állapotukat; egyszerűen csak olyan bejegyzések vannak ezeknek a csomópontoknak a csomagtovábbítási táblázatukban (Forwarding Information Base – FIB), amelyek alapján az e-től visszatérő speciális csomagokat más úton kell továbbadni. Így aztán amíg csak az IPFRR irányítja át a csomagokat, *minden* csomag a $b \to a \to e \to a \to b \to c \to d$ szuboptimális útvonalat fogja követni. Természetesen ebből következik, hogy ha a meghibásodás állandónak bizonyul (bizonyos idő után), valamilyen újjáépítő megoldásra is szükség lesz a továbbítási útvonalak optimalizálásához.

2. Kutatási célkitűzések

Az előző fejezetben megismerkedtünk az IPFRR módszerek alapelveivel, ebben a szakaszban pedig áttekintjük a fő követelményeket, amelyeknek minden IPFRR módszer eleget kell tegyen. A fejezet zárásaként a jelenlegi megoldások problémáira mutatok rá, majd ismertetem kutatási célkitűzéseimet.

A követelmények, amelyeket egy modern IPFRR módszernek ki kell elégítsen a következők. Először is ezeknek a megoldásoknak egy autonóm rendszeren (Autonomous System – AS) belül kell működniük, azaz az IGP útválasztás kiegészítéseként tekinthetünk rá. Másodszor az IPFRR technikáknak meglehetősen gyorsaknak kell lenniük; ökölszabályként tekinthetjük úgy, hogy legfeljebb 50ms lehet a hibajavítás ideje, ami még hanghívások számára is elfogadható. Továbbá ezt a teljesítményt a hagyományos IP csomagtovábbítás jelentős módosítása nélkül kell elérni. Ezen kívül követelmény, hogy a lehető legtöbb egyszeres hibát képes legyen a módszer kijavítani, bár többszörös meghibásodást nem szükséges védeni. Végezetül pedig elvárt, hogy az elkerülő utak a lehetőségekhez mérten rövidek legyenek.

Jelenleg minden IPFRR módszer küzd valamilyen hiányossággal. Néhány nem tud védelmet nyújtani minden egyszeres meghibásodás ellen, mások hajlamosak továbbítási hurkokat képezni többszörös meghibásodások esetén, a megmaradtak pedig számottevő – többnyire menedzsment – költséget vezetnek be a rendszerbe.

Ezek alapján tehát kutatási célkitűzésem létrehozni egy IPFRR módszert, amely teljesíti az összes az előzőekben említett követelményt, azaz ennek a módszernek képesnek kell lennie minden egyszeres meghibásodás javítására, soha nem okozhat továbbítási hurkot és csak korlátozottan növelheti a rendszer bonyolultságát.

3. Módszertan

A tézisfüzet további részében algoritmusokat mutatok be, amelyek megoldanak bizonyos az IPFRR-hez kapcsolódó problémákat. Módszertanom alapvetően a gráf- és algoritmuselmélet eszközeti használja, ezek segítségével bizonyítom az algoritmusok bizonyos tulajdonságait (helyesség, teljesség és komplexitás).

Számos esetben azonban az általam javasolt algoritmusok nemcsak megoldanak bizonyos problémákat, hanem heurisztikák segítségével optimalizálják is a kapott eredményeket. Habár a helyesség és a teljesség ekkor már elméleti eredményekkel igazolt, ezen heurisztikák teljesítményét kiterjedt szimulációk segítségével vizsgáltam.

Végezetül munkám során prototípusok is elkészültek; így néhány esetben lehetőségem nyílt szimulációk helyett mérések elvégzésére is.

4. Új eredmények

4.1. Több hiba kezelése interface-alapú IPFRR-rel

Az egyszeres hibák 100%-ának javításához a csomagok valamilyen megjelölése elengedhetetlen. Ez a jelölés lehet explicit, mikor a csomag fejléce ténylegesen módosításra kerül, vagy implicit, mikor valamilyen olyan plusz információt veszünk figyelembe, amit a hagyományos csomagtovábbításnál nem használunk. Ilyen információ lehet a bejövő interface, ami közelítőleg jelzi a csomag már megtett útját. Ahogy ebből a fejezetből ki fog derülni, az interface-alapú csomagtovábbítás – amely a célcím mellett a csomag bejövő interface-ét is figyelembe veszi – jól használható IPFRR-re.

Elsőre úgy tűnhet, hogy az interface-alapú csomagtovábbítás viszonylag egyszerűen megvalósítható; mivel a mai routerekben a csomagtovábbítással kapcsolatos döntéseket a saját memóriával rendelkező linecardok végzik, ha különböző csomagtovábbítási táblát (FIB) töltünk a különböző linecardokba, interface-alapú csomagtovábbítást kapunk. Sajnos a helyzet nem ilyen egyszerű, mert egy tipikus linecard több interface-szel is rendelkezhet amelyeket mind különbözően kellene kezelni, ezért egy ilyen megoldás implementálása nem könnyű, ám kétség kívül új eszköz kifejlesztése nélkül is megvalósítható.

Jelenleg számos interface-alapú csomagtovábbítást használó IPFRR megoldás ismert. Ezek közül a módszerek közül az első a U-turn Alternates [Atl06] volt, mely a bejövő interface-ből eredő információt arra használja, hogy detektálja, ha a következő hop visszaküldi a csomagot. A Failure Insensitive Routing (FIR) [NLYZ03, LYN+04, NLY+07] ezt a megoldást fejleszti tovább oly módon, hogy nem csak a következő hop által visszaküldött csomagokat, hanem minden interface-t figyelembe vesz, amelyen hibamentes esetben nem érkezik csomag, ezáltal alkalmassá válik arra, hogy elkerüljön bármilyen egyszeres linkhibát. Később ezt a módszert kiterjesztették csomóponthibákra [ZNY+05] (erre a módszerre a későbbiekben Failure Inferencing based Fast Rerouting (FIFR) néven hivatkozok) valamint tetszőleges egyszeres link vagy csomóponthibára [WN07].

Sajnos minden ilyen technikának van egy súlyos hiányossága: hajlamosak csomagtovábbítási hurkokat generálni többszörös hibák esetén. Mivel az IPFRR technikák használhatóak a tranziens meghibásodások átkonfigurálás nélküli kezelésére, akár számos másodperc, vagy esetenként egy perc is lehet az az időtartam, amíg csupán az IPFRR biztosítja a kapcsolatot,[1] így az IPFRR által generált hurkok akár hosszú életűek is lehetnek. Természetesen azonban ezek a hurkok nem csak késleltetik a helyreállítást egy olyan szintig, ami sokkal rosszabb mint egyszerű újjáépítés használatával lenne, de akár torlódást is okozhatnak. A FIR-t javasló szerzők is felismerték ezt a

[1] [ICM+02] szerint a meghibásodások egy percen belül maguktól is helyreállnak az esetek körülbelül 50%-ában.

problémát, és a Blacklist-based Interface-Specific Forwarding (BIFS) [WZN06] segítségével igyekeztek orvosolni azt. Sajnos azonban a BIFS-nek egyszerű interface-alapú csomagtovábbításnál többre van szüksége, így nem alkalmazható gerinchálózatokban.

Javasoltam egy IPFRR technikát, a Loop-free Failure Insensitive Routingot (LFIR)[C2, C3, P1], amely mindig képes elkerülni tetszőleges egyszeres linkhibát, és amely sosem okozhat hurkot. Az LFIR-rel a 1.2. tézis foglalkozik.

1. TÉZIS CSOPORT: [C2, C3, P1] *Megmutattam, hogy a U-turn alternates, FIR és FIFR módszerek által számított elkerülő utak nem feltétlenül mentesek a továbbítási hurkoktól. Ennek a hiányosságnak az orvoslására javasoltam egy új hurokmentes interface-alapú IPFRR módszert.*

A hurokképzés fő oka az alapértelmezett útvonalak megválasztásából fakad. Ha az ép hálózatban a legrövidebb utakat használjuk alapértelmezettként lehetséges, hogy a hálózat „elfelejti" az első hibát, és hurkot okoz azáltal, hogy megpróbálja kijavítani a másodikat is. A 1.1. tézis ezen az észrevételen alapul.

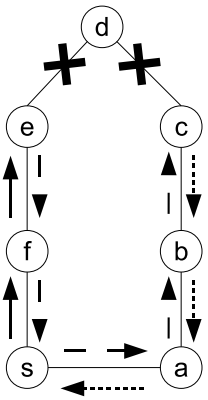
1.1. TÉZIS: [C2, C3] *Konstruktív bizonyítással beláttam, hogy nincs olyan datagram hálózat, amely a csomagokat a célcím és a bejövő interface alapján továbbítja, tetszőleges hálózati topológia és additív élköltség mellett a csomag fejlécének módosítása nélkül védelmet biztosít és képes lenne garantálni, hogy az alábbi két feltétel egyszerre teljesül:*

- *az alapértelmezett utak a legrövidebbek bármely két csúcs közötti és*
- *az elkerülő utak többszörös meghibásodás esetén is hurokmentesek.*

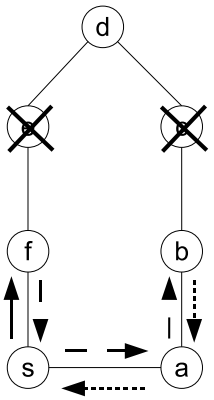
Megjegyzés: *Mivel az U-turn alternates, FIR és FIFR IPFRR technika, amely interface-alapú csomagtovábbítást használ és a legrövidebb utak az alapértelmezettek, így hurkot okozhatnak.*

Az interface-alapú csomagtovábbításon alapuló IPFRR módszerek által generált hurkok azon a jelenségen alapulnak, hogy a csomag állapota elveszhet. Mivel a csomag sikertelen továbbítását csupán a bejövő interface jelzi, a hiba „elfelejthető", ha az elkerülő úton lévő csomag visszatérhet az alapértelmezett útvonalra. Ekkor a második sikertelen csomagtovábbításkor a hálózat a hibát ismét megpróbálja elkerülni, esetleg az eredeti hibán keresztül. Sajnos ha az alapértelmezett utak a legrövidebbek, nem mindig lehetséges biztonságosan elhagyni az elkerülő utat.

Tekintsük a 2. ábrán látható hálózatot, és tételezzük fel, hogy ebben a hálózatban FIR-t használnak és hogy a linkek súlya egységesen 1! Ha s csomagot akar küldeni d-nek, az alapértelmezett útvonal az $s \to f \to e \to d$. Ha azonban $\{e, d\}$ link megsérül, a FIR lokális elkerüléssel azonnal helyreállítja a kapcsolatot. Mivel más lehetőség nincs, e visszaküldi a csomagot f-nek. Mivel interface-alapú csomagtovábbítást használunk, f nem küldi vissza a csomagot e-nek, hanem s-nek adja őket, ahonnan az



2. ábra. Hurok keletkezik, ha d és e, valamint c és d közötti linkek meghibásodnak



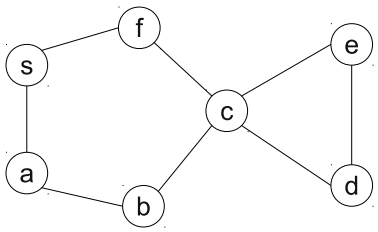
3. ábra. Hurok keletkezik, ha c és e is meghibásodott

$s \to a \to b \to c \to d$ úton eljutnak d-be. Ha azonban van egy másik meghibásodás is, mondjuk ha $\{c, d\}$ link is elérhetetlen lesz, a csomagok épp így eljutnak c-be. Mivel c a csomagokat b-től kapja, és mivel c az alapértelmezett következő hopja b-nek d felé, c-nek nincs lehetősége észlelni hogy ez már a második hiba (emlékezzünk, hogy csak a cél címet és a bejövő interface-t vesszük figyelembe). Így aztán c azt „gondolja", hogy ez az első hiba, megpróbálja elkerülni a $c \to b \to a \to s \to f \to e$ úton, ami hurkot okoz.

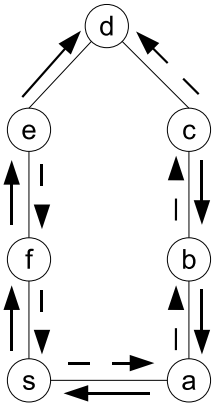
Ha FIFR használatát feltételezzük a helyzet majdnem ugyanez. Tekintsük a 3. ábrán látható hálózatot! Megfigyelhető, hogy most két csomópont meghibásodására van szükségünk, c és e válik elérhetetlenné. Ebben az esetben f megpróbálja javítani a hibát, a csomagok b-hez jutnak, ahol azok ismét javításra kerülnek és így egy hurok keletkezik.

Valójában nem is feltétlenül szükséges többszörös hibákat feltételeznünk a hurkok előállításához; a FIR és a FIFR azon verziója, amely képes kezelni mind link, mind csomóponthibákat [WN07], egyszerű csomóponthiba esetén is hurkot okozhat. Tekintsük a 4. ábrán látható hálózatot és tételezzük fel, hogy a linkek súlyai egységesen 1-ek, valamint hogy s kíván d-nek csomagot küldeni. Ekkor a legrövidebb út $s \to f \to c \to d$. Ha f elveszíti a kapcsolatot c-vel, nem tudja eldönteni, hogy a link vagy a csomópont hibásodott-e meg. Mivel csomóponthibát feltételezve a csomag nem juttataható el d-be, még a FIFR is meg fogja próbálni a hiba elkerülését. Azonban, ha végül mégis csak c a meghibásodott erőforrás, a csomagok ismételten hurokba kerülnek.

Láthattuk, hogy nincs mindig mód a hurkok elkerülésére, ha az alapértelmezett utak a legrövidebbek, ami azonban nem jelenti azt, hogy akkor sincs, ha ezzel a megkötéssel nem élünk.



4. ábra. Példa hálózat a FIFR hurokképzéséhez



5. ábra. Két élidegen irányított feszítőfa d-be, mint célba

1.2. TÉZIS: [C2, C3, P1] *Adtam egy interface alapú IPFRR módszert – a Loop-Free Failure Insensiteive Routingot (LFIR) –, amely a legrövidebb utaknál hosszabb alapértelmezett útvonalak árán képes mindig elkerülni, hogy továbbítási hurkot hozzon létre.*

Megjegyzés: *Ugyan egy IPFRR technikát javasoltam, az IP nem szükségszerű feltétel. Az LFIR minimális módosítással bármely interface-alapú csomagtovábbítást használó datagram hálózatban alkalmazható.*

A korábbiakból kiderült, hogy a legfontosabb ok, amely a hurkok kialakulásához vezet abból a jelenségből fakad, hogy visszajuthatunk az alapértelmezett útra egy hiba után. Ezt lehetséges elkerülni olyan útválasztás használatával, ahol a csomagok ha egyszer elkerülő útra kerültek, többet soha nem hagyják el azt. Ehhez vegyük észre, hogy az IP hálózatokban egy adott célhoz történő továbbítás tipikusan egy irányított feszítőfa mentén történik. Ha lehetséges volna találni egy élidegen feszítőfapárt minden célhoz, az egyiket alapértelmezett, a másikat elkerülő útként használhatnánk.

Az LFIR ezen az ötleten alapul. Mivel bármely kétszeresen élösszefüggő gráfban lehet találni két élidegen irányított feszítőfát, az LFIR képes jól elkülönülő alapértelmezett és elkerülő utat találni a hálózat minden 2-élösszefüggő komponensében, ezáltal kezelni minden olyan egyszeres linkhibát, amely nem szakítja ketté a hálózatot. Továbbá, ha a csomagtovábbítás egy elkerülő úton sikertelen, az LFIR képes ezt detektálni, a csomagokat eldobni, valamint azonnal elindítani az újjáépítő eljárást a hálózat átkonfigurálására. Mivel a csomagok az elkerülő utakat csak akkor hagyhatják el, ha célbaértek vagy ha elhagyták a kétszeresen élösszefüggő komponenst, hurkok nem keletkezhetnek. Az előző hálózatban d-be, mint célba mutató két élidegen feszítőfa látható az 5. ábrán.

Top.	Hurokvalószínűség linkhibával	Hurokvalószínűség csomóponthibával
NSF	0.39 %	5.4 %
Germany	0.82 %	18.68 %
Italy	0.76 %	18.38 %

1. táblázat. Hurokképzés valószínűsége, kétszeres link- vagy egyszeres csomóponthiba esetén

Csomópont-szám	Fokszám	Hurokvalószínűség linkhibával	Hurokvalószínűség csomóponthibával
20	2	0.32 %	11.64 %
20	3	0.02 %	1.35 %
30	2	0.48 %	22.28 %
30	3	0.05 %	4.95 %
40	2	0.61 %	30.37 %
40	3	0.11 %	9.16 %
50	2	0.7 %	36.4 %
50	3	0.14 %	13.35 %

2. táblázat. Hurokképzés valószínűsége BRITE-tal generált véletlen hálózatokban, kétszeres link- vagy egyszeres csomóponthiba esetén

Most ismét tételezzük fel, hogy s csomagot akar küldeni d-nek, és az alapértelmezett csomagtovábbítás az egyszerű nyilak mentén történik. Ha nincs meghibásodás, a csomagok d-hez az $s \to f \to e \to d$ úton jutnak el. Ha a $\{d, e\}$ kapcsolat megszakad, e átirányítja a csomagokat a másodlagos (szaggatott) fára, és a csomagok d-hez az $e \to f \to s \to a \to b \to c \to d$ úton jutnak el. Ha a $\{c, d\}$ kapcsolat is megszakad, c továbbra is a másodlagos fán kap csomagokat. Mivel ekkor c nem tudja a csomagokat ugyanezen a fán továbbítani, a második meghibásodás detektálható, a csomagok eldobásra kerülnek, és azonnal elindítható az újjáépítési eljárás. Figyeljük meg a módszer gyengeségét is: ha történetesen nem s, hanem például a a forrás, a csomagok a hibamentes hálózatban a szuboptimális $a \to s \to f \to e \to d$ utat fogják követni.

Az előzőekből következik tehát, hogy egyfajta kompromisszumra kényszerülünk: a hurkok mindig elkerülhetőek annak fejében, hogy alapértelmezettként hosszabb utakat használunk. A kérdés ezek után, hogy milyen valószínűséggel jönnek létre hurkok az eredeti módszer alkalmazásával, és hogy milyen hosszúak azok a „hosszabb" utak az LFIR alkalmazásával. A következő két tézis ezzel a kérdéssel foglalkozik.

1.3. TÉZIS: [C2, C3] *Valós és mesterségesen generált hálózati topológiákon végzett*

Top.	LFIR heurisztikával	LFIR heurisztika nélkül
NSF	106.27 %	137.37 %
Germany	116.36 %	146.15 %
Italy	112.07 %	150.38 %

3. táblázat. Az utak átlagos hosszának aránya a legrövidebb utakhoz képest heurisztikát használó és azt nem alkalmazó LFIR használata esetén

kiterjedt szimulációs vizsgálatok segítségével megmutattam, hogy egyszeres csomópont- vagy többszörös linkhiba esetén a FIR tényleg hurkot okozhat az LFIR-rel ellentétben, amely mindig garantálja a hurokmentességet.

A szimulációkhoz valódi és véletlenszerűen generált hálózatokat használtam. A véletlenszerűen generált hálózatokat a Boston university Representative Internet Topology gEnerator (BRITE) [MLMB05] alkalmazás segítségével készítettem. Vizsgálataim során azt tapasztaltam, hogy véletlenszerűen választott meghibásodás esetén keletkezhet hurok (1. és 2. táblázat). Észrevehető, hogy ennek valószínűsége nem hanyagolható el, egyszerű csomóponthiba esetén akár 36%-ot is elérheti. Ezek alapján nem férhet kétség ahhoz, hogy az LFIR rendelkezik bizonyos előnyökkel az eredeti FIR-hez képest; az egyetlen kérdés, hogy ez az előny képes-e kompenzálni az alapértelmezett utak hosszának növekedéséből fakadó hátrányt. A következő tézis ezzel a kérdéssel foglalkozik.

1.4. TÉZIS: [C2, C3] *Heurisztikus megoldást javasoltam az LFIR által használt alapértelmezett utak hosszának csökkentésére. Kiterjedt szimulációs vizsgálatokat végeztem és azt találtam, hogy a vizsgált esetekben az LFIR által használt alapértelmezett utak átlagosan legfeljebb 17%-kal hosszabbak, mint a legrövidebb utak, ha a szóbanforgó heurisztikát használjuk az uthosszak csökkentésére.*

A szimulációkat az előzőekben ismertetett módon végeztem azzal a különbséggel, hogy most a hurokképzés esélye helyett az alapértelmezett utak hosszát vizsgáltam. Ahogy az észrevehető (3. és 4. táblázat), az LFIR alapértelmezett útjai alig hosszabbak, mint a legrövidebb utak, legfeljebb 17%-kal, ha az úthosszakat csökkentő heurisztikát is alkalmazzuk. Ezek az eredmények azt is megmutatják, hogy az általam javasolt heurisztika hatékony; a megoldás az utak átlagos hosszát a legrövidebb utak hosszának 15%–30%-ával csökkentette.

Csomópont-szám	Fokszám	LFIR heurisztikával	LFIR heurisztika nélkül
20	2	105.53 %	128.59 %
20	3	101.68 %	127.61 %
30	2	105.26 %	129.27 %
30	3	101.63 %	129.68 %
40	2	105.04 %	129.65 %
40	3	101.56 %	131.04 %
50	2	104.86 %	129.86 %
50	3	101.5 %	132 %

4. táblázat. Az utak átlagos hosszának aránya a legrövidebb utakhoz képest heurisztikát használó és azt nem alkalmazó LFIR használata esetén BRITE-tal generált véletlen hálózatokban

4.2. Csúcsredundáns fák keresése lineáris időben

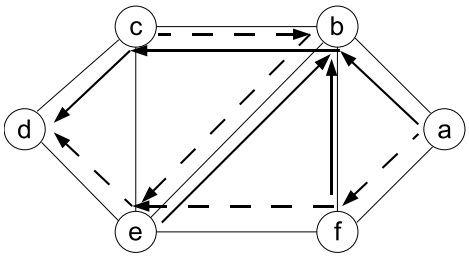
Habár az LFIR az első olyan interface-alapú védelmi mechanizmus, amely nem okozhat hurkokat, ez a megoldás egy ennél sokkal fontosabb tapasztalattal is szolgál: észrevehető, hogy az irányított fák jól alkalmazhatóak IPFRR-re. Sajnos az LFIR élidegen feszítőfáji csak egyszerű linkhibák ellen nyújthatnak védelmet. Mivel a csúcsidegen feszítőfák fogalma értelmetlen, ez a megközelítés nem is terjeszthető ki egykönnyen a csúcshibák esetére. A megoldáshoz tekintsük a 4.1. definíciót!

4.1. Definíció. Adott egy irányítatlan G gráf egy r csúccsal. G egy r-ben gyökerező *él-/csúcsredundáns fapárja* egy irányított fapár egy közös r gyökérrel, oly módon, hogy r minden $s \neq r$ csúcsból mindkét fán elérhető, és a két út a két fán él-/csúcsidegen.

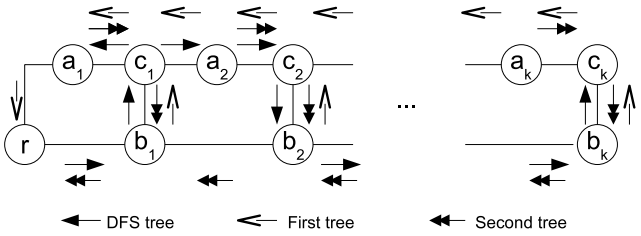
Megjegyzés: Természetesen él-/csúcsredundáns fákat csak 2-él-/2-csúcsösszefüggő gráfokban találhatunk (ez nem csak szükséges, de elégséges feltétel is), amely követelményt viszont a valós hálózatok nem mindig teljesítenek. Ebben a téziscsoportban ezért feltételezem, hogy a gráf mindig 2-csúcsösszefüggő; a következő téziscsoportban erre a megkötésre már nem lesz szükség.

Egyszerűen belátható, hogy az élidegen feszítőfák és az élredundáns fák ekvivalensek. Fontos különbség azonban, hogy az élredundáns fák definíciója már egyszerűen kiterjeszthető csúcsredundáns fákká, amelyek képesek biztosítani az összeköttetést csomóponhibák esetén is.

A redundáns fák [MBFG99, XCT02c, XCT02b, XCT02a, XCT03, ZXTT05, ZXTT08, C5, C6, C7], ismertek még mint független fák [IR84, Han98, ABS96, ZI89, Huc94, MTSIN98, CLY03, CLY06] és színezett fák [Ram04, TRK06, BR06, RHK06, RKK07, JRY09, KRKH09], széleskörűen tanulmányozott gráfelméleti fogalmak. Ezeket az objektumokat először Itai és Rodeh használta [IR84]-ben. Ebben a munkában a szerzők



6. ábra. Egy d-ben gyökerező redundáns fapár



7. ábra. A gráf és a mélységi bejárás, amely esetén a ReducedCostV nem talál redundáns fákat lineáris időben

lineáris számítási idejű algoritmusokat javasoltak él-, illetve csúcsredundáns fák számítására. Optikai hálózatok védelmére először Médard és munkatársai [MBFG99] ajánlották ezeket a feszítőfákat, immár redunáns fa néven.

A redundáns fák nyilvánvaló módon használhatóak hibajavításra: ha egy egyszeres meghibásodás történik, az csak az egyik fán szakíthatja meg a kapcsolatot a gyökér felé, így a hiba könnyedén elkerülhető a forgalom másik fára kapcsolásával; ezen a fán a cél szükség szerűen elérhető marad. A továbbiakban G gráf csúcsainak halmazát $V(G)$, éleinek halmazát pedig $E(G)$ jelöli. A 6. ábra egy csúcsredundáns fapárt ábrázol. Mivel ebben a téziscsoportban csúcsredundáns fákkal foglalkozok, a továbbiakban az egyszerű „redundáns fa" kifejezés ezeket a fákat jelöli.

Az első a talált redundáns fák összköltségét lineáris idejű heurisztikával csökkentő megoldás a ReducedCostV [ZXTT05, ZXTT08] volt. Ennél az algoritmusnál a költség azon élek számát jelentette, amelyek *bármely* irányban szerepeltek valamelyik fában (tehát egy él egy költségűnek számít, ha tartalmazza valamelyik vagy mindkét redundáns fa). A következőkben ki fog derülni, hogy ez az algoritmus sajnos nem lineáris idejű, ezért javasoltam egy *centralizált* megoldást, ami képes kezelni ezt a problémát. Ebben a téziscsoportban bemutatok egy *elosztott* megoldást is nem csupán egyetlen fapár, hanem minden csúcshoz mint gyökérhez egy fapár lineáris idejű számítására.

2. TÉZIS CSOPORT: [C5, C6, C7] *Mutattam egy gráfot és egy lehetséges mélységi bejárását, melyen a ReducedCostV algoritmus nem tud lineáris időben megfelelő redundáns fapárt találni. Javasoltam egy szigorúan lineáris idejű centralizált algoritmust redundáns fák keresésére. Az általam javasolt algoritmus hasonló hatásfokú heurisztikát alkalmaz a redundáns fák költségének csökkentése érdekében, mint az eredeti ReducedCostV. Javasoltam továbbá egy elosztott lineáris idejű algoritmust is, amely minden csúcsba mint gyökérbe talál egy redundáns fapárt.*

2.1. TÉZIS: [C7] *Mutattam egy gráfot és egy lehetséges mélységi bejárását, melyen a ReducedCostV algoritmus nem tud lineáris időben megfelelő redundáns fapárt találni.*

Megjegyzés: *A ReducedCostV módosítható oly módon, hogy biztosítsuk a helyességét, de ekkor szükség szerűen elveszti linearitását.*

A ReducedCostV eljárás a gráf úgynevezett „fül-dekompozícióján" (ear decomposition) alapul. Ez a dekompozíció egy adott csúcsból indul – ez lesz majd a később fák a gyökere. Induláskor a gyökér az egyetlen csúcs a fákban, később más csúcsokkal és élekkel is bővítjük azokat. Minden lépésben egy összefoglalóan fülnek nevezett utat vagy egy kört keresünk az eredeti irányítatlan gráfban, és ennek csúcsait és éleit adjuk hozzá a fákhoz. Fontos azonban, hogy mivel irányított fákat építünk, az élek irányítása is számít; az éleket a fülön az egyik irányban az egyik fához, a másik irányban a másikhoz adjuk hozzá. Minden egyes fül esetén létfontosságú annak eldöntése, hogy melyik irány melyik fához tartozik. Annak érdekében, hogy ezt a kérdést mindig megválaszolhassa a ReducedCostV fenntartja a csúcsok egy rendezését, amit „feszültségnek" nevez.

A ReducedCostV problémái a rendezés fenntartásából fakadnak: a lineáris idejű végrehajtáshoz olyan adatstruktúrára lenne szükség, amely támogatja mind az új elem beszúrását, mind két elem összehasonlítását $O(1)$ időben. Természetesen a hagyományos adatstruktúrák (mint például tömbök, láncolt listák, bináris fák) erre semmiképpen nem alkalmasak.

A szerzőkkel folytatott privát levelezés alapján ismert számomra, hogy ők egyszerűen számokat használtak. Sajnos azonban ha számokat rendelünk minden csúcshoz, azzal egy „címteret" képezünk,[2] ami kimerülhet nem megfelelő hozzárendelés esetén. Használva a 7. ábrán látható gráfot és mélységi bejárást megmutattam, hogy nem létezik megfelelő hozzárendelés, és a címtér mérete exponenciálisan skálázódik a csúcsok számával. Ezzel természetesen fix pontosságú számokkal nem tudhatunk lépést tartani. Megmutattam, hogy 32 bites egészek esetén legfeljebb 103 csúcsra van szüksége az algoritmusnak ahhoz, hogy hibásan működjön, míg 64 bites egészek esetén ez 199 csúcsnál következik be. A lebegőpontos számok sem segíthetnek a problémán, hiszen például dupla pontosság esetén 64 bitet használunk, ami megintcsak kevésnek fog bizonyulni legfeljebb 199 csúccsal rendelkező gráfnál. Tetszőleges pontosságú aritmetikát alkalmazva persze a probléma megoldódik, de ekkor az $O(1)$ időben történő összehasonlítás illetve beszúrás lehetőségét veszítjük el.

Mivel a ReducedCostV problémái abból a tényből fakadnak, hogy feszültséget alkalmaz, azok elkerülhetők a feszültségek mellőzésével. Emlékezzünk arra, hogy a feszültségekre a fülek végpontjainak összehasonlításához volt szükség. Mivel az én algoritmusom mindig „tudja" melyik végpont a kisebb, erre az összehasonításra nincs szüksége.

2.2. TÉZIS: [C7] *Lehetővé téve, hogy az algoritmus a mélységi feszítőfa mentén ne csak lefelé, hanem felfelé is keressen, készítettem egy algoritmust, ami tetszőleges*

[2] Ezek nem igazi címek, csak az összehasonításhoz kellenek.

Topológia	Csúcsok száma	Eredeti költség	Költség a javított algoritmussal	Növekmény
Germany	17	19.82	19.82	0%
NSF	26	31.65	31.8	0.47%
Cost266	37	43.49	44.13	1.47%
Germany50	50	57.06	57.78	1.26%

5. táblázat. A redundáns fák átlagos költségei valós topológiákon

2-csúcsösszefüggő hálózatban képes lineáris – $O(|E(G)|)$ – időben egy csúcsredundáns fapárt találni. Kiterjedt szimulációs vizsgálatokat végeztem és azt találtam, hogy az így meghatározott redundáns fák költsége az általam vizsgált esetekben alig nagyobb, mint azoké, amiket a ReducedCostV határoz meg.

A ReducedCostV mélységi bejárást használ a fülek megtalálására. A módszer lefele halad a mélységi fán addig, amíg el nem jut egy megfelelő őshöz. Ahogy az belátható, ez az út az ősig egy füljelölt. A feszültség nélküli fülkeresés ötlete azon az észrevételen alapul, hogy lehetséges egy fület találni nem csak *lefele*, hanem *felfelé* is haladva a mélységi fán. Így, ha tudjuk, hogy a potenciális további fülek végpontjai közül épp melyiknek a legalacsonyabb a feszültsége, akkor azt csak akkor hagyjuk el, ha már *minden* hozzá kapcsolódó fület megtaláltunk. Természetesen az összes fül megtalálásához néha szükség van a mélységi fán felfele is haladni. Az előbbi elveket az 1. algoritmus foglalja össze.

Algoritmus 1 r gyökérbe lineáris időben két redundáns fát kereső algoritmus pszeudokódjának vázlata

1: Számíts ki egy r-ben gyökerező mélységi feszítőfát! Legyen S egy üres verem! Tedd r-et S tetejére!
2: **while** S nem üres
3: $x \leftarrow$ pop S
4: Keresd meg az összes még nem lefedett fület x-szel mint végponttal (lefele és felfele mozogva)!
5: Add a fület a fákhoz azt figyelembe véve, hogy x a legkisebb feszültségű csúcs!
6: Tedd egy verem tetejére a talált csúcsokat fordított sorrendben!
7: **end while**

Az algoritmusom heurisztikát is használ annak érdekében, hogy minimalizálja a talált fák költségét. Ahogy Zang és munkatársai [ZXTT05, ZXTT08]-ban bebizonyították, minél hosszabb füleket talál egy algoritmus, annál kisebb lesz a fák költsége. Ezért mikor az én algoritmusom lefele halad a mélységi fában, pontosan ugyanazt

Csúcsok száma	Fokszám	Eredeti költség	Költség a javított algoritmussal	Növekmény
20	2	25.74	26.31	2.21%
20	3	23.94	24.38	1.84%
30	2	38.76	39.73	2.5%
30	3	36.26	37.07	2.23%
40	2	51.76	53.04	2.47%
40	3	48.57	49.69	2.31%
50	2	64.68	66.3	2.5%
50	3	60.76	62.16	2.3%

6. táblázat. A redundáns fák átlagos költségei a BRITE-tal generált véletlen topológiákon

teszi, mint amit az eredeti algoritmus csinál. Továbbá, ha felfele mozog, akkor maximalizálja az egy füllel fedett szomszédok számát. Kiterjedt szimulációk segítségével megmutattam, hogy ez a heurisztika hatékony, és alig nagyobb költségű fákat eredményez, mint az eredeti ReducedCostV algoritmus.

A szimulációkhoz ismét valós és véletlenszerűen generált hálózati topológiákat használtam. Ahogy az leolvasható a táblázatokból (5. és 6. táblázat) az általam alkalmazott heurisztika hatékony, a számított fák költségét a vizsgált esetekben lineáris időben lecsökkentette kevesebb mint 2,5%-kal nagyobb értékre, mint amit az eredeti algoritmus adott eredményül.

A csúcsredundáns fák csúcsidegen utakat adnak gyökerük felé, így jól használhatóak arra, hogy a forgalmat egy adott cél felé védjük. Mivel azonban valós hálózatokban általánosságban bármely két csomópont között lehet forgalom, gyaktran szükségessé válik, hogy ne csak egy redundáns fapárt határozzunk meg, hanem egyet minden csúcsba, mint gyökérbe. Ekkor azonban – bár egy redundáns fapár megtalálása $O(|E|)$ idejű – minden csúcsba egy redundáns fapár kiszámítása $O(|V(G)||E(G)|)$ időt vesz igénybe. Továbbá mivel minden feszítőfának $|V(G)|-1$ éle van, $2|V(G)|$ feszítőfának (egy pár minden csúcshoz) már $2|V(G)|(|V(G)|-1)$ éle lesz. Mivel csupán kiírni ezeket az éleket a memóriába $2|V(G)|(|V(G)|-1)$ lépést igényel, az összes fa megtalálása $\Omega(|V(G)|^2)$ idejű, azaz nem létezhet lineáris idejű centralizált algoritmus.

Észrevehetjük azonban, hogy datagram hálózatokban – mint az IP – általában nincs szükség arra, hogy minden csomópont (router) kiszámítsa a teljes fákat, csak arra, hogy a következő hopot, azaz a csomópontot reprezentáló csúcsból kivezető éleket ismerje ezek mentén a fák mentén. Természetesen ezen a módon továbbra is ki lesz számítva az összes redundáns fa, azonban ez az információ most *elosztottan* lesz jelen a hálózatban; habár egyetlen csomópont sem fog ismerni egyetlen teljes fát sem, a hálózat képes lesz a csomagokat a fák mentén továbbítani. Most tehát $|V(G)|$

processzor áll rendelkezésünkre (ezek tipikusan a routerek a hálózatban), amelyek lehetővé teszik, hogy tovább csökkentsük a szükséges számítási komplexitást.

2.3. TÉZIS: [C5, C6] *Adtam egy elosztott algoritmust, mely képes tetszőleges 2-összefüggő gráfban lineáris – $O(|E(G)|)$ – időben minden csúcsba mint gyökérbe találni egy redundáns fapárt, ha rendelkezésre áll legalább $|V(G)|$ processzor – egy minden csúcshoz.*

Következmény: *Egy olyan hálózatban, ahol minden router rendelkezi számítási kapacitással, az összes csomópont ki tudja számítani a következő hopot az összes fa mentén lineáris időben.*

A minden csúcsba mint gyökérbe történő redundáns faszámítás egy speciális köztes feszítőgráfon alapul, amelyet „irányított majdnem körmentes gráfnak" (Almost Directed Acyclic Graph – ADAG) nevezek. Az elnevezés abból a tényből fakad, hogy hogy ezek a gráfok egyetlen csúcs – a gyökér – elhagyásával irányított körmentes gráfokká alakíthatóak. Az általam javasolt módszer feltételezi, hogy mindegyik processzor ugyanazt az ADAG-ot számítja ki, ami könnyen megvalósítható, ha mindegyikük ugyanazt a gráfot kapja bemenetként.

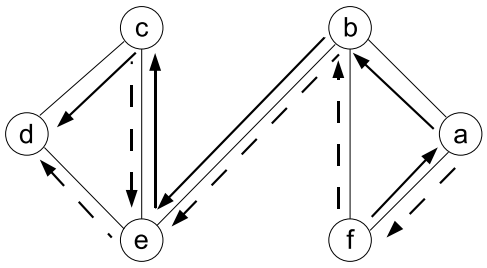
A feszítő ADAG egy féligrendezést definiál amiből minden processzor kiszámíthatja a hozzá tartozó x csúcsnál szigorúan nagyobb és kisebb csúcsokat. Ezen információ alapján pedig lehetséges néhány egyszerű szabályt alkotni, amelyek használatával $O(1)$ időben meghatározhatóak egy adott csúcsban gyökerező redundáns fákhoz tartozó x-ből kilépő élek. Az élek kiszámítása tehát csak $O(|V(G)|)$ időt vesz igénybe, így mivel az ADAG kiszámítása is megvalósítható $O(|E(G)|)$ lépésben, a teljes számítási idő $O(|E(G)|)$-nek adódik, azaz az algoritmus lineáris. Ezeket az elveket foglalja össze a 2. algoritmus.

Algoritmus 2 A redundáns fák egy adott u csúcsból kimenő éleinek kiszámítását végző algoritmus vázlatos pszeudokódja

1: Keress egy feszítő ADAG-ot!
2: Készíts egy féligrendezést a kapott ADAG alapján; legyen V_u^- illetve V_u^+ az u-nál szigorúan nagyobb illetve kisebb csúcsok halmaza!
3: **for** minden d csúcsra
4: Néhány egyszerű szabály és az előző két halmaz alapján számítsd ki az u-ból kimenő, d-ben gyökerező fákhoz tartozó éleket!
5: **end for**

4.3. A redundáns fák továbbfejlesztése

Az előző fejezetben módszereket láttunk redundáns fák kiszámítására. Ebben a fejezetben a kapott fák milyenségével foglalkozok. A fák javítására alapvetően két



8. ábra. Egy d-ben gyökerező maximálisan redundáns fapár

módon van lehetőség.

Egyrészt, ahogy azt korábban már említettem (4.1. definíciónál szereplő megjegyzés), redundáns fákat csak 2-csúcsösszefüggő gráfokban lehet találni. Mivel a valódi hálózatok még hibamentesen sem feltétlenül ilyenek (lásd például [SND]-ből az Abeline és az AT&T, vagy a [GO05]-ben szereplő olasz gerinchálózatot), kiterjesztettem a redundáns fákat 2-csúcsösszefüggő gráfok esetéről tetszőleges gráfokra. Az így kapott általánosított fákat nevezzük maximálisan redundáns fáknak (4.2. definíció). A 8. ábrán egy maximálisan redundáns fapárt láthatunk.

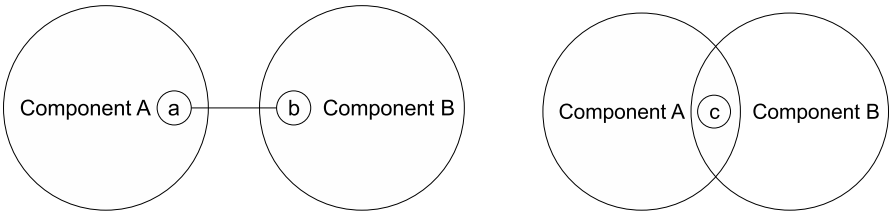
4.2. Definíció. Adott egy irányítatlan G gráf egy r csúccsal. G egy r-ben gyökerező *maximálisan redundáns fapárja* egy irányított fapár egy közös r gyökérrel, oly módon, hogy r minden $s \neq r$ csúcsból mindkét fán elérhető, és a két útnak a két fán minimális számú közös csúcsa van.

Megjegyzés: Abból, hogy az utaknak minimális számú közös csúcsa van (az elvágó csúcsok közösek) következik, hogy a közös élek száma is minimális, hiszen egy él, melynek mindkét végpontja elvágó csúcs, nem más, mint egy elvágó él.

A maximálisan redundáns fák mindig a maximális lehetséges redundanciát biztosítják. Ha van két csúcs között két csúcsidegen út, akkor a maximálisan redundáns fákban a két út csúcsidegen lesz. Egyébként viszont a két út csak minimális számú közös csúcsot fog tartalmazni, azokat, amelyeket minden útnak tartalmaznia kell – azaz ezek épp az elvágó csúcsok. Maximálisan redundáns fákat tetszőleges összefüggő gráfban találhatunk.

Másrészt a legtöb valódi hálózatban az utakat optimalizálják valahogy, azaz általánosságban nem elég egyszerűen redundáns fákat keresni, hanem ezeknek a fáknak teljesíteniük kell néhány további elvárást is. Példaképpen tekinthetünk a korábban már megismert ReducedCostV algoritmusra. Ahogy azt már említettem az előző fejezetben, ez az algoritmus heurisztikát alkalmaz, hogy csökkentse a talált fák öszköltségét.

A ReducedCostV olyan hálózatokban alkalmazható, amelyekben a védelmi erőforrásokat (linkeket) le kell foglalni. Mivel az IP hálózatokban a linkek költségét úgy



(a) nincs közös csúcs (b) egy közös csúcs

9. ábra. 2-csúcsösszefüggő komponensek viszonya

választják meg, hogy a legrövidebb utak épp azok, amelyeket a csomagoknak választani kellene, ezekben a hálózatokban sokkal jobb úgy választani meg a fákat, hogy az általuk tartalmazott utak rövidek legyenek. Ebben a fejezetben tehát heurisztikát javasolok, amely a lineáris futási idő megőrzése mellett jelentősen csökkenti a az utak hosszát.

3. TÉZIS CSOPORT: [C7, J4] *Megmutattam, hogy a redundáns fa meghatározására szolgáló lineáris idejú algoritmusom maximálisan redundáns fákat is képes találni minimális módosítással. Bebizonyítottam továbbá, hogy az általam javasolt elosztott algoritmus kis módosításokkal szintén alkalmazható tetszőleges összefüggő gráfon, és ekkor maximálisan redundáns fákat határoz meg. Végezetül csökkentettem a maximálisan redundáns fák mentén az utak hosszát a számítási komplexitás növelése nélkül.*

3.1. TÉZIS: [C7] *Bebizonyítottam, hogy az általam javasolt lineáris idejű redundánsfakereső algoritmus (lásd a 2.2. tézist) kis módosításokkal használható tetszőleges összefüggő gráfon. Ezeken a gráfokon az algoritmus maximálisan redundáns fákat határoz meg és komplexitása továbbra is lineáris – $O(|E(G)|)$ – marad.*

A megoldás ötlete abból fakad, hogy tetszőleges gráf 2-csúcsösszefüggő komponensekből épül fel (természetesen néhány komponens esetleg csak egy csúcsot tartalmaz), és ezekben a komponensekben a korábbi algoritmusok helyesen működnek. Ez alapján egy algoritmus éppen akkor működik helyesen, ha képes kezelni a 2-csúcsösszefüggő komponensek határait. A komponensek két módon csatlakozhatnak egymáshoz, ezek láthatóak a 9. ábrán.

Két 2-összefüggő komponensnek vagy nincs közös csúcsa, vagy egy közös csúcsa van, hiszen ha legalább két közös csúcsuk lenne, akkor azok igazából egy komponenst alkotnának. Módosítottam tehát az algoritmusaimat, hogy képesek legyenek mindkét helyzetet megfelelően kezelni.

3.2. TÉZIS: [J4] *Továbbfejlesztettem a 2.3. tézisben szereplő elosztott algoritmust oly módon, hogy képes legyen lineáris – $O(|E(G)|)$ – időben minden csúcsba mint gyökérbe egy maximálisan redundáns fapárt meghatározni tetszőleges összefüggő gráfban.*

A korábban tárgyalt észrevételt használva – azaz hogy egy összefüggő gráf 2-összefüggő komponensekből áll – általánosítható a használt köztes gráf; az általánosított köztes gráfot Általánosított ADAG-nak (Generalized ADAG – GADAG) nevezem. Ezt a feszítőgráfot használva ismét a csúcsok féligrendezéséhez juthatunk.

Ez a féligrendezés most is segít megtalálni a kimenő éleket azokhoz a maximálisan redundáns fákhoz, amelyek a csúcs 2-csúcsösszefüggő komponensében gyökereznek, azonban azokhoz a fákhoz, amelyek a fennmaradó csúcsokban gyökereznek, egy megfelelő elvágó csúcsra is szükség van; ezt a csúcsot egy speciális rekurzív algoritmussal kereshetjük meg. A 3. algoritmus ennek a módszernek a vázlatos pszeudokódját szemlélteti.

Algoritmus 3 A maximálisan redundáns fák u-ból kimenő éleinek kiszámítása

1: Számítsd ki az u-val azonos 2-csúcsösszefüggő komponensben lévő csúcsokban gyökerező fákhoz tartozó éleket a 2. algoritmussal!
2: Számítsd ki az éleket a GADAG globális gyökere felé!
3: Használva a „rekurzív keresést” találd meg a fennmaradó fákhoz tartozó éleket!

Ahogy azt korábban már említettem, ebben a fejezetben olyan téziseket mutatok be, amelyek a redundáns fákat javítják. Ugyan az egyik irány a redundáns fák javítására, ha általánosítjuk őket tetszőleges összefüggő gráfokra, de vannak egyéb igények is, amelyek komolyan befolyásolják ezen fák valódi hálózatokban történő alkalmazhatóságát. Egyszerű heurisztikát javasoltam, amely jelentősen csökkenti a talált utak hosszát (a csúcsok számát az úton) az eredményül kapott maximálisan redundáns fákban.

3.3. TÉZIS: [J4] *Az elosztott maximálisan redundánsfa-kereső algoritmushoz olyan a lineáris komplexitást megőrző heurisztikát javasoltam, amely csökkenti a kapott fákban az utakon szereplő csúcsok számát. Valós és mesterségesen generált topológiákon kiterjedt szimulációs vizsgálatokat végeztem és azt találtam, hogy az általam vizsgált esetekben a javasolt heurisztika a maximálisan redundáns fákon szereplő utakban a csúcsok számát 20%–50%-kal csökkenti.*

A maximálisan redundáns fákban az utak hossza a kiszámított GADAG milyenségétől függ. Észrevehető, hogy gyakran van néhány olyan él a gráfban, amely a feszítő GADAG-ban nem fordul elő egyik irányban sem. Ha hozzá tudnánk adni ezeket az éleket egy megfelelő irányban a GADAG-hoz, akkor az csökkenthetné a maximálisan redundáns fákban az utak hosszát. A probléma, hogy ezeket az éleket nem lehet tetszőleges irányítással a köztes gráfhoz adni, mert annak GADAG tulajdonságát fent kell tartani.

Ezt a tulajdonságot általánosságban ugyan nem egyszerű fenntartani, azonban az általam alkalmazott technika speciális GADAG-okat keres, olyanokat, amiket speciális módon is irányított körmentes gráffá lehet alakítani. Bebizonyítottam, hogy mindig

Topológia	Csúcsok száma	Elsődleges út heurisztika nélkül	Másodlagos út heurisztika nélkül	Elsődleges út heurisztikával	Másodlagos út heurisztikával
Abilene	12	210%	212%	168%	171%
Germany	17	231%	230%	191%	190%
AT&T	22	221%	224%	166%	167%
NSF	26	224%	222%	178%	174%
Italy	33	248%	247%	175%	174%
Cost266	37	250%	253%	190%	194%
Germany50	50	304%	309%	212%	214%

7. táblázat. A csúcsok átlagos száma a redundáns fákban lévő utak mentén valódi hálózatokban (100% a legrövidebb utak hossza)

Csúcsok száma	Fokszám	Elsődleges út heurisztika nélkül	Másodlagos út heurisztika nélkül	Elsődleges út heurisztikával	Másodlagos út heurisztikával
20	2	217%	224%	173%	174%
20	3	298%	313%	180%	181%
30	2	235%	243%	182%	182%
30	3	332%	352%	190%	190%
40	2	250%	259%	190%	189%
40	3	361%	385%	198%	197%
50	2	263%	273%	197%	195%
50	3	388%	415%	205%	203%

8. táblázat. A csúcsok átlagos száma a redundáns fákban lévő utak mentén mesterségesen generált hálózatokban (100% a legrövidebb utak hossza)

biztonságos, ha a GADAG-hoz olyan irányban adunk hozzá egy élet, ami nem ad kört ehhez az irányított körmentes gráfhoz. Így nincs másra szükség, mint egy topologikus rendezésre, amit használva minden élre megtalálható a biztonságos irány $O(1)$ időben. Mivel egy topologikus rendezés $O(|E(G)|)$ időben végrehajtható egy összefüggő gráfban, így a maximálisan redundáns fák továbbra is lineáris időben megtalálhatóak.

Sajnos ez a heurisztika egyes esetekben elméletileg akár növelheti is néhány út hosszát, ám átfogó szimulációs vizsgálatokkal megmutattam, hogy ha az elosztott maximálisan redundáns fa számítására használt algoritmussal alkalmazzuk, az utak hossza összességében csökken. Az eredményeket a 7. és a 8. táblázatban foglaltam össze.

4.4. Lightweight Not-Via

Számos IP alapú gyors hibajavítást lehetővé tevő módszer született már, azonban a legnagyobb ipari támogatás azok közül, amelyek minden meghibásodást képesek áthidalni, kétség kívül a „Not-via addresses” [BSP10] mögött van. Habár ennek a

módszernek vannak hátulütői, a Not-via egy meglehetősen hatékony megoldás. Ez a téziscsoport a Not-via hátulütőivel és ezek felszámolásával illetve csökkentésével foglalkozik.

A Not-via legfontosabb problémája, hogy sok IP címet használ a meghibásodott erőforrás azonosítására, ezzel jelentős címkezelési feladatot hárít az operátorokra. Továbbá a Not-vianak számos legrövidebb út számításra is szüksége van, ami komoly számítási komplexiátssal jár. Végezetül a Not-via néhány helyzetet speciálisan kénytelen kezelni, ezzel nehezítve a hibakeresést és fejlesztési munkálatokat.

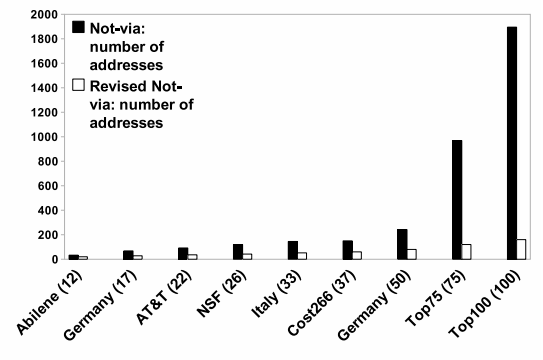
4. TÉZIS CSOPORT: [C5, C6, J4, P2] *Maximálisan redundáns fák alkalmazásával megmutattam, hogy a Not-via algoritmus számítási és menedzsment költsége a hagyományos legrövidebb úton alapuló útválasztáshoz hasonló szintre csökkenthető.*

4.1. TÉZIS: [C5, C6, J4, P2] *Újradefiniáltam a Not-via által számított elkerülő utakat oly módon, hogy azok maximálisan redundáns fák mentén legyenek meghatározva, és ezt az új IPFRR módszert Lightweight Not-vianak neveztem. Megmutattam, hogy a Lightweight Not-via csomópontonként csak három IP címet használ és alkalmazásával az elkerülő utak meghatározásából adódó számítási teher aszimptotikusan eltűnik.*

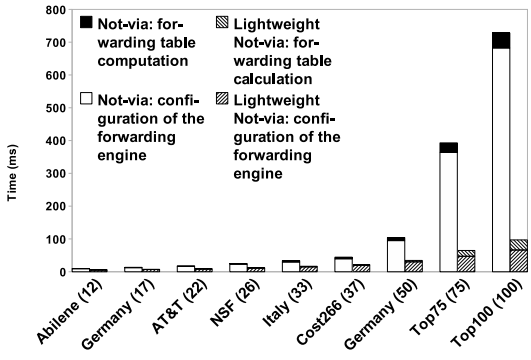
Megjegyzés: *A Lightweight Not-vianak egyáltalán nincs szüksége plusz IP címekre számos valós hálózatban.*

A Lightweight Not-via azon az ötleten alapul, hogy maximálisan redundáns fákat is használhatunk ahelyett, hogy számos legrövidebb utat keresnénk elkerülő útként. A hibamentes hálózatban a csomagok továbbítsa továbbra is a legrövidebb úton történik, és a Lightweight Not-via viselkedése az eredeti Not-viaéhoz nagyon hasonló marad akkor is, ha meghibásodás lép fel. A csomagokat ekkor ez a módszer is egy IP-in-IP alagútba teszi, és így próbálja eljuttatni őket a következő utáni hopra, ahol elhagyhatják az alagutat, és ahonnan már a legrövidebb úton továbbíthatóak. A különbség, hogy az elkerülő utakat teljesen másképpen számítjuk. A Lightweight Not-via egy maximálisan redundáns fapárt számít ki minden csomópontba a számos legrövidebb út számítás helyett. Vegyük észre, hogy hasonló megoldást javasolnak [CHA07]-ben is, habár ott – egyéb különbségek mellett – redundáns fákat keresnek, amiket csak 2-összefüggő hálózatokban lehet találni.

Mivel egy maximálisan redundáns fapár minden olyan hiba esetén biztosítja az összeköttetést, amely nem szakítja ketté a hálózatot, a következő utáni hop legalább az egyik benne gyökerező fán elérhető marad. Azaz a csomagokat egy olyan alagútba kell tenni, mely IP címének jelentése, hogy az elsődleges fán kell a csomagot továbbítani. Ha a csomag továbbítása ismét sikertelen valahol, ezt a célcímet megváltoztatjuk, és a csomagok a másodlagos fán kerülnek továbbításra. Ha a továbbítás ezen a fán is sikertelen, a csomagokat el kell dobni, és az újjáépítési eljárást azonnal el kell kezdeni, hiszen a hálózatban többszörös meghibásodás lépett fel.



10. ábra. Az eredeti és a Lightweight Not-via által igényel extra címek száma egyes közismert ISP topológiákon (a csúcsok száma zárójelben van megadva), minden ötödik csomópontot LAN-nak feltételezve



11. ábra. A továbbítási táblázat kiszámításának és a továbbítási egységbe történő letöltése az eredeti és a Lightweight Not-via esetén, minden ötödik csomópontot LAN-nak feltételezve

Vegyük észre, hogy ezt a módszert használva minden csomópontnak három IP címe van: egy IP cím leírja az alapértelmezett legrövidebb utat és két másik a maximálisan redundáns fákat. Számos IP hálózatban azonban, ahol minden interface-nek saját címe van, akár teljesen el is lehet kerülni az extra címek használatát: mivel minden több úton is elérhető csomópontnak legalább két interface-e van, ezeknek az interface-eknek a címei leírhatják a redundáns fákat, és a loopback cím jelentheti, hogy a csomagot a legrövidebb úton kell továbbítani.

Mivel az elkerülő utak maximálisan redundáns fákon alapulnak és mivel egy maximálisan redundáns fa keresése minden csomópontba lineáris időben is elvégezhető (lásd 3.2. tézis), a Lightweight Not-via útszámítását az alapértelmezett utakat számító Dijkstra-algoritmus dominálja.

4.2. TÉZIS: [C5, C6] *Valós és mesterségesen generált hálózatokat használva egy IPFRR tesztrendszeren kiterjedt méréseket folytattam és azt találtam, hogy a vizsgált esetekben a Not-via által igényelt extra menedzsment és számítási kapacitás legalább egy nagyságrenddel nagyobb, mint a Lightweight Not-via esetében.*

Irányításommal MSc hallgatók tejes értékű Not-via és Lightweight Not-via prototípusokat fejlesztettek. Ezen prototípusokat átalakítottam, hogy képesek legyenek szimulált topológiákat figyelembe venni, majd vizsgáltam a számítások által igényelt időt és a szükséges címek számát.

Habár a Not-via alkalmazása akár pont-pont hálózatokban is problémákba ütközhet, teljesen menedzselhetetlenné LAN-ok jelenlétében válik. Ezért valós és véletlen

hálózati topológiákat használtam a tesztrendszerhez, és mértem Not-via illetve Lightweight Not-via esetén az IP címek számát és a számítási időt. Eredményeim a 10. és a 11. ábrán láthatóak.

Megfigyelhető, hogy önmagában már a címek puszta száma megakadályozhatja a Not-via használatát. Természetesen több ezer IP címet kézzel beállítani lehetetlen, de még központi menedzsment eszközökkel is komoly gondot jelent. Továbbá az IP címek magas száma nem csak a menedzsmentkomplexitás növekedésében jelenik meg. Ahogy az a 11. ábrán látható, az idő, ami a kiszámított információ továbbítási táblába történő letöltéséhez kell, szintén rosszul skálázódik, ha Not-viat használunk.

Ezzel szemben vegyük észre, hogy a Lightweight Not-via képes volt az IP címek számát alacsonyan tartani. Továbbá a kevesebb IP cím kisebb számítási időt is maga után vont, ami így már összemérhető a jelenlegi routerek újrakonfigurálásra fordított idejével.

5. Az új eredmények alkalmazhatósága

Napjainkban egyre fokozódó érdeklődés irányul az IPFRR módszerekre. Köszönhetően a telekommunikációs hálózatokban megfigyelhető konvergenciának, az IP-t egyre gyakrabban használják nem csak elasztikus, de valós idejű forgalom továbbítására is. Mivel ez a forgalom különleges követelményeket támaszt, számos probléma áll elő.

Egyike a megkerülhetetlen problémáknak a gyors helyreállítás igénye. Habár az IP tradicionálisan ellenálló a hibákkal szemben, a helyreállítás időigényét sohasem tekintették lényeges problémának. Sajnos a hosszú helyreállítási idő manapság már nem elfogadható, és az IPFRR technikák pár éven belül elengedhetetlenné fognak válni.

Habár az igény folyamatosan növekszik, a gyors hibajavítás IP hálózatokban komoly gondokat vet fel, kielégítő megoldásokkal nem rendelkeztünk. Az általam javasolt LFIR algoritmus – 1. tézis csoport – leküzdi az interface alapú csomagtovábbításon alapuló IPFRR módszerek egyik legnagyobb hibáját nevezetesen, hogy ezek módszerek hajlamosak továbbítási hurkokat okozni. Az általam javasolt Lightweight Not-via – 4. tézis csoport – pedig a Not-via legfontosabb hátulütőjére, az extrém menedzsment erőforrásigény kontrolálására ad lehetőséget. Ezek az IPFRR módszerek akár azonnal alkalmazhatóak lennének valós routerekben, ahogy az teszthálózat szintjén meg is történt a Lightweight Not-via esetén.

A 2. és 3. tézis csoportban ismertetett eredményeim a redundáns fák keresésének módszereit terjesztik ki számos irányban. Ezek az eredmények elengedhetetlenek, hisz ezek teremtik meg az elméleti alapot a Lightweight Not-via számára. Természetesen ezen eredmények alkalmazhatósága nem korlátozódik az IPFRR-re; használhatóak a

telekommunikáció számos területén, ahol a redundáns fákat alkalmazták. Ilyen területek például a nagyméretű, sokprocesszoros rendszerek [IR84], az áramkör kapcsolt hálózatok [MBFG99, XCT02c, XCT02b, XCT02a, XCT03, ZXTT05, ZXTT08] vagy a szenzorhálózatok [Ram04, TRK06, BR06, RHK06, RKK07, JRY09].

6. A tézisekhez kapcsolódó publikációk

6.1. [C2] cikk: A Novel Loop-Free IP Fast Reroute Algorithm

Enyedi G., Rétvári G., Cinkler T.
13th EUNICE Open European Summer School; BEST PAPER AWARD
A cikk az 1.1., 1.2., 1.3. és 1.4. tézishez kapcsolódik.

Összefoglalás: Ebben a cikkben leírásra kerül az LFIR. Megmutatjuk, hogy az eredeti FIR hurkokat okozhat, valamint hogy az LFIR nem igényel számottevően több erőforrást.

Hozzájárulás: A cikk első szerzője formalizálta a problémát, kifejlesztette a matematikai módszereket annak kezelésére, elvégezte a szimulációs vizsgálatokat és megírta a cikket a második és harmadik szerzők irányításával.

6.2. [C3] cikk: A Loop-Free Interface-Based Fast Reroute Technique

Enyedi G., Rétvári G.
4th EURO-NGI Conf. on Next Generation Internet Networks (EuroNGI)
A cikk a 1.1., 1.2., 1.3. és 1.4. tézishez kapcsolódik.

Összefoglalás: Ez a cikk az előző egy kibővített, javított változata. Az LFIR leírása további részleteket is tartalmaz, valamint új szimulációs eredményekkel támogatjuk az állításokat.

Hozzájárulás: A cikk első szerzője formalizálta a problémát, kifejlesztette a matematikai módszereket annak kezelésére, elvégezte a szimulációs vizsgálatokat és megírta a cikket a második szerző irányításával.

6.3. [C5] cikk: IP Fast ReRoute: Lightweight Not-Via without Additional Addresses

Enyedi G., Szilágyi P., Rétvári G., Császár A.
IEEE INFOCOM-MiniConference
A cikk a 2.3., 4.1. és 4.2. tézishez kapcsolódik.

Összefoglalás: Ebben a mini-konferencia cikkben röviden bemutatjuk a Not-via hibáit mérési eredményekkel alátámasztva, a Lightweight Not-via 2-összefüggő hálózatokban használható verziója (ez a módszer tehát csak redundáns fákat használ), valamint ismertetjük a több redundáns fát lineáris időben kereső algoritmust.

Hozzájárulás: A cikk első szerzője formalizálta a problémát, kifejlesztette a matematikai módszereket annak kezelésére, elvégezte a szimulációs vizsgálatokat és a cikk számottevő részét írta a harmadik és a negyedik szerzők irányításával. A tesztrendszert a második szerző fejlesztette. A cikk fennmaradó részét a harmadik szerző írta.

6.4. [C6] cikk: IP Fast ReRoute: Lightweight Not-Via

Enyedi G., Rétvári G., Szilágyi P., Császár A.
IFIP Networking
A cikk a 2.3., 4.1. és 4.2. tézishez kapcsolódik.

Összefoglalás: Ez a cikk az előző rövidített cikk teljes értékű konferenciacikk változata. A Not-via hibái és a Lightweight Not-via részletes leírásra kerül.

Hozzájárulás: A cikk első szerzője formalizálta a problémát, kifejlesztette a matematikai módszereket annak kezelésére, elvégezte a szimulációs vizsgálatokat és a cikk számottevő részét írta a második és a negyedik szerzők irányításával. A tesztrendszert a harmadik szerző fejlesztette. A cikk fennmaradó részét a második szerző írta.

6.5. [C7] cikk: On Finding Maximally Redundant Trees in Strictly Linear Time

Enyedi G., Rétvári G., Császár A.
IEEE Symposium on Computers and Communications, ISCC
A cikk a 2.1., 2.2. és 3.1. tézishez kapcsolódik.

Összefoglalás: Ebben a cikkben mutatjuk meg, hogy Zhang algoritmusa nem feltétlenül képes lineáris időben megfelelő redundáns fákat találni. Megadjuk a javított, centralizált algoritmust, ami most már szigorúan lineáris idejű. Végezetül ismertetésre kerül ezen algoritmus maximálisan redundáns fákat kereső kiterjesztése.

Hozzájárulás: A cikk első szerzője formalizálta a problémát, kifejlesztette a matematikai módszereket annak kezelésére, elvégezte a szimulációs vizsgálatokat és megírta a cikket a második és harmadik szerzők irányításával.

6.6. [J4] cikk: Finding Multiple Maximally Redundant Trees in Linear Time

Enyedi G., Rétvári G.
Periodica Polytechnica Electrical Engineering
A cikk a 3.2., 3.3. és 4.1. tézishez kapcsolódik.

Összefoglalás: Ebben a cikkben az elosztott maximálisan redundáns fákat kereső algoritmus, valamint az ezek mentén meghatározott utak hosszát csökkentő heurisztika kerül ismertetésre. A cikk végén megmutatjuk, hogy hogyan használhatjuk a maximálisan redundáns fákat a Lightweight Not-via kiterjesztésére, és ezzel a módszert tetszőleges hálózatban alkalmazhatóvá tesszük.

Hozzájárulás: A cikk első szerzője formalizálta a problémát, kifejlesztette a matematikai módszereket annak kezelésére, elvégezte a szimulációs vizsgálatokat és megírta a cikket a második szerző irányításával.

6.7. [P1] szabadalmi bejelentés: Link failure recovery method and apparatus

Császár A., Enyedi G.
Patent application
Ez a publikáció az 1.2. tézishez kapcsolódik.

Összefoglalás: A szabadalmi bejelentés az LFIR módszerét tárgyalja. Az LFIR publikálásra került [C2, C3]-ban.

Hozzájárulás: A publikáció második szerzője formalizálta a problémát, kifejlesztette a matematikai módszereket annak kezelésére valamint az első szerzővel közösen elkészítette a szabadalmi bejelentést.

6.8. [P2] szabadalmi bejelentés: Lightweight Not-Via IP Fast Reroute

Császár A., Enyedi G.
Patent application
Ez a publikáció a 4.1. tézishez kapcsolódik.

Összefoglalás: A szabadalmi bejelentés a Lightweight Not-via módszerét tárgyalja. A Lightweight Not-via publikálásra került [C5, C6, J4]-ben.

Hozzájárulás: A publikáció második szerzője formalizálta a problémát, kifejlesztette a matematikai módszereket annak kezelésére valamint aze első szerzővel közösen elkészítette a szabadalmi bejelentést.

Referenciák

[ABS96] F. Annexstein, K. Berman, and R. Swaminathan. Independent spanning trees with small stretch factors. Technical report, 1996.

[Atl06] A. Atlas. U-turn alternates for ip/ldp fast-reroute. Internet Draft, available online: `http://tools.ietf.org/html/draft-atlas-ip-local-protect--uturn-03`, February 2006.

[BR06] R. Balasubramanian and S. Ramasubramanian. Minimizing average path cost in colored trees for disjoint multipath routing. In *15th International Conference on Computer Communications and Networks, ICCCN 2006*, pages 185–190, October 2006.

[BSP10] S. Bryant, M. Shand, and S. Previdi. IP fast reroute using Not-via addresses. Internet Draft, available online: `http://tools.ietf.org/html/draft-ietf--rtgwg-ipfrr-notvia-addresses-06`, 2010.

[CHA07] T. Cicic, A. F. Hansen, and O. K. Apeland. Redundant trees for fast IP recovery. In *Broadnets*, pages 152–159, 2007.

[CLY03] S. Curran, O. Lee, and X. Yu. Chain decompositions and independent trees in 4-connected graphs. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 186–191, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[CLY06] S. Curran, O. Lee, and X. Yu. Finding four independent trees. *SIAM Journal on Computing*, 35(5):1023–1058, 2006.

[fS02] International Organization for Standardization. OSI IS-IS intra-domain routing protocol. ISO/IEC 10589:2002, 2002.

[GO05] M. L. Garcia-Osma. TID scenarios for advanced resilience. Tech. Rep., The NOBEL Project, Work Package 2, Activity A.2.1, Advanced Resilience Study Group, Sep 2005.

[Han98] Dagmar Handke. Independent tree spanners. In *Graph-Theoretic Concepts in Computer Science*, pages 203–214, 1998.

[Huc94] A. Huck. Independent trees in graphs. *Graphs and Combinatorics*, 10(1):29–45, 1994.

[ICM+02] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 237–242, New York, NY, USA, 2002. ACM.

[IR84] A. Itai and M. Rodeh. The multi-tree approach to reliability in distributed networks. In *SFCS '84: Proceedings of the 25th Annual Symposium onFoundations of Computer Science, 1984*, pages 137–147, Washington, DC, USA, 1984. IEEE Computer Society.

[JRY09] G. Jayavelu, S. Ramasubramanian, and O. Younis. Maintaining colored trees for disjoint multipath routing under node failures. *IEEE/ACM Transactions on Networking*, 17(1):346–359, 2009.

[KRKH09] S. Kini, S Ramasubramanian, A. Kvalbein, and A. F. Hansen. Fast recovery from dual link failures in IP networks. INFOCOM 2009, April 2009.

[LYN+04] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah. Proactive vs reactive approaches to failure resilient routing. In *IEEE Infocom'04*, 2004.

[MBFG99] M. Médard, R. A. Barry, S. G. Finn, and R. G. Galler. Redundant trees for preplanned recovery in arbitary vertex-redundant or edge-redundant graphs. *IEEE/ACM Transactions on Networking*, 7(5):641–652, Oct 1999.

[MLMB05] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Boston university Representative Internet Topology gEnerator. `http://www.cs.bu.edu/brite`, 2005.

[Moy98] J. Moy. OSPF version 2. Internet Engineering Task Force: RFC 2328, April 1998.

[MTSIN98] K. Miura, D. Takahashi, S.-I.Nakano, and T. Nishizeki. A linear-time algorithm to find four independent spanning trees in four-connected planar graphs. In *WG '98: Proceedings of the 24th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 310–323, London, UK, 1998. Springer-Verlag.

[NLY+07] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C-N. Chuah. Fast local rerouting for handling transient link failures. *IEEE/ACM Transaction on Networking*, 15(2):359–372, 2007.

[NLYZ03] S. Nelakuditi, S. Lee, Y. Yu, and Z.-L. Zhang. Failure insensitive routing for ensuring service availability. In *Proceedings International Workshopon Quality of Service (IWQoS)*, 2003.

[Ram04] S. Ramasubramanian. Supporting multiple protection strategies in optical networks. Technical report, Department of Electrical and Computer Engineering, University of Arizona, November 2004.

[RHK06] S. Ramasubramanian, M. Harkara, and M. Krunz. Distributed linear time construction of colored trees for disjoint multipath routing. In *IFIP Networking*, pages 1026–1038, May 2006.

[RKK07] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz. Disjoint multipath routing using colored trees. *Computer Networks*, 51(8):2163–2180, 2007.

[SND] Survivable fixed telecommunication Network Design library (SNDlib). `http://sndlib.zib.de`.

[TRK06] P. Thulasiraman, S. Ramasubramanian, and M. Krunz. Disjoint multipath routing in dual homing networks using colored trees. In *IEEE Globecom*, pages 1–5, November/December 2006.

[WN07] J. Wand and S. Nelakuditi. IP fast reroute with failure inferencing. In *Proc. of ACM SIGCOMM Workshop on Internet Network Management – The Five-Nines Workshop*, 2007.

[WZN06] J. Wang, Z. Zhong, and S. Nelakuditi. Handling multiple network failures through interface specific forwarding. In *IEEE Globecom*, November 2006.

[XCT02a] G. Xue, L. Chen, and K. Thulasiraman. Cost minimization in redundant trees for protection in vertex-redundant or edge-redundant graphs. In *PCC '02: Proceedings of the Performance, Computing, and Communications Conference, 2002. on 21st IEEE International*, pages 187–194, Washington, DC, USA, 2002. IEEE Computer Society.

[XCT02b] G. Xue, L. Chen, and K. Thulasiraman. Delay reduction in redundant trees for preplanned protection against single link/node failure in 2-connected graphs. In *IEEE Globecom*, November 2002.

[XCT02c] G. Xue, L. Chen, and K. Thulasiraman. QoS issues in redundant trees for protection in vertex-redundant or edge-redundant graphs. In *IEEE International Conference on Communications (ICC)*, volume 5, pages 2766–2770, 2002.

[XCT03] G. Xue, L. Chen, and K. Thulasiraman. Quality-of-service and quality-of-protection issues in preplanned recovery schemes using redundant trees. *IEEE Journal on Selected Areas in Communications*, 21(8):1332–1345, October 2003.

[ZI89] A. Zehavi and A. Itai. Three tree-paths. *Journal of Graph Theory*, 13(2):175–188, 1989.

[ZNY+05] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C-N Chuah. Failure inferencing based fast rerouting for handling transient link and node failures. In *IEEE Infocom'05*, 2005.

[ZXTT05] W. Zhang, G. Xue, J. Tang, and K. Thulasiraman. Linear time construction of redundant trees for recovery schemes enhancing QoP and QoS. INFOCOM 2005, March 2005.

[ZXTT08] W. Zhang, G. Xue, J. Tang, and K. Thulasiraman. Faster algorithms for construction of recovery trees enhancing QoP and QoS. *IEEE/ACM Trans on Networking*, 16(3):642–655, 2008.

Az új eredmények publikálása

[J] **Folyóiratok**

[J1] András Császár, **Gábor Enyedi**, Gábor Rétvári, Marcus Hidell, Peter Sjödin, „Converging the Evolution of Router Architectures and IP Networks”, *IEEE Network Magazine*, Special Issue on Advances in Network Systems Architecture, 21:4(8–14) 2007.

[J2] Péter Fodor, **Gábor Enyedi**, Gábor Rétvári, Tibor Cinkler, „Layer-Preference Policies in Multi-layer GMPLS Networks”, *Photonic Network Communications* 2009.

[J3] **Gábor Enyedi**, Gábor Rétvári, „Gyors hibajavítás IP hálózatokban (Hungarian)”, *Híradástechnika*, 64:3–4(20–24) 2009.

[J4] **Gábor Enyedi**, Gábor Rétvári, „Finding Multiple Maximally Redundant Trees in Linear Time”, *Elfogadva: Periodica Polytechnica Electrical Engineering*, online elérhető: `http://opti.tmit.bme.hu/~enyedi/ipfrr/`, 2010.

[C] **Konferenciák**

[C1] Péter Fodor, **Gábor Enyedi**, Tibor Cinkler, „A Fast and Efficient Traffic Engineering Method for Transport Networks”, V Workshop in G/MPLS Networks (WGN5), pages 129–141, 2006.

[C2] **Gábor Enyedi**, Gábor Rétvári, Tibor Cinkler, „A Novel Loop-free IP Fast Reroute Algorithm”, 13th EUNICE Open European Summer School and IFIP TC6.6 Workshop on Dependable and Adaptable Networks and Services (EUNICE), Twente, The Netherlands, 2007. BEST PAPER AWARD

[C3] **Gábor Enyedi**, Gábor Rétvári „A Loop-Free Interface-Based Fast Reroute Technique”, 4th EURO-NGI Conf. on Next Generation Internet Networks (EuroNGI), Kraków, Poland, pages 39–44, 2008.

[C4] Péter Fodor, **Gábor Enyedi**, Gábor Rétvári, Tibor Cinkler, „An Efficient and Practical Layer-preference Policy for Routing in GMPLS Networks”, 13th Int. Telecommunications Network Strategy and Planning Symposium (NETWORKS), Budapest, Hungary, 2008.

[C5] **Gábor Enyedi**, Péter Szilágyi, Gábor Rétvári, András Császár, „IP Fast ReRoute: Lightweight Not-Via without Additional Addresses”, IEEE INFOCOM-MiniConference, Rio de Janeiro, Brazil, April 2009.

[C6] **Gábor Enyedi**, Gábor Rétvári, Péter Szilágyi, András Császár, „IP Fast ReRoute: Lightweight Not-Via”, IFIP Networking, Aachen, Germany, May 2009.

[C7] **Gábor Enyedi**, Gábor Rétvári, András Császár, „On Finding Maximally Redundant Trees in Strictly Linear Time”, IEEE Symposium on Computers and Communications, ISCC, Sousse, Tunisia, July 2009.

[C8] Gábor Rétvári, János Tapolcai, **Gábor Enyedi**, András Császár „IP Fast ReRoute: Loop-free Alternates Revisited”, Elfogadva: IEEE INFOCOM, Shanghai, online elérhető: `http://opti.tmit.bme.hu/~enyedi/ipfrr/`, China, April 2011.

[P] **Szabadalmi bejelentések**

[P1] András Császár, **Gábor Enyedi**, „Link failure recovery method and apparatus”, Szabadalmi bejelentés WO/2009/010090, 2009.

[P2] András Császár, **Gábor Enyedi**, „A System and Method of Implementing Lightweight Not-via IP Fast ReRoutes in a Telecommunications Network”, Szabadalmi bejelentés WO/2010/055408, 2010.

[P3] András Császár, **Gábor Enyedi**, „Fast Path Notification”, Szabadalmi bejelentés PCT/EP2010/059391, 2010.

[P4] András Császár, **Gábor Enyedi**, „IP Fast ReRoute Relying on Fast Path Notification”, Szabadalmi bejelentés PCT/EP2010/065040, 2010.