# *SZAKDOLGOZAT*

*Varga Norbert Lajos*

*Debrecen,*

*2010*

**Debreceni Egyetem**

**Informatikai Kar**

**Informatikai Rendszerek és Hálózatok Tanszék**

# Aktuális problémák vizsgálata kommunikációs és hálózati rendszerekben – Megbízható IP multicasting

Témavezető:

Dr. Almási Béla

egyetemi docens

Készítette:

Varga Norbert Lajos

Mérnök-informatikus BsC

*Debrecen,*

*2010*

# Tartalomjegyzék

# 1. Bevezetés

A számítógépes hálózatokban a technológia változásoknak köszönhetően egyre inkább növekszik a küldendő adat mennyisége, az egyre több, egymással kommunikáló alkalmazás egyre nagyobb átviteli sebességet követel meg. Ezzel egy időben a csomópontok száma is növekszik LAN és WAN környezetekben egyaránt. Naponta milliók használják az Internet adta legmodernebb szolgáltatásokat, melyek már szinte mind a számítógépes hálózatokba integrálódtak. Legyen szó multimédiás szolgáltatásokról vagy állományküldő rendszerekről, a késleltetésbeli vagy épp sebességbeli elvárások hatalmasak. Egyre inkább ki kell használni a már meglévő, de eddig annyira nem elterjedt technikákat, hogy enyhíteni lehessen a hálózatokban küldés során fellépő torlódáson és a kiszolgálók túlterheltségén.

Egyik ilyen lehetséges kihasználandó technológia a multicast üzenetküldés a számítógépes hálózatokban. „Már az 1980-as évek végén a Stanford Egyetemen Steven Deering azon dolgozott, hogy hogyan tudna létrehozni egy elosztott operációs rendszert („Vsystem”), mely az alhálózatokon keresztül többes küldéssel képesek lennének kommunikálni, és a multicast forgalom képes legyen forgalomirányítókon keresztül eljutni a célállomásokhoz.”[3] Ezek után kezdték kidolgozni a multicast kommunikáció alapvető szabványait, a működéséhez szükséges irányító- és egyéb protokollokat. A multicast lényege, hogy egy adott információt, üzenetet nem csak egy gépnek, felhasználónak küldhetünk el, hanem egy ún. csoportnak, mely több állomást foglal magába, és az üzenetet a csoport összes tagja megkapja. Ha küldésnél nem multicast-ot használnánk, akkor minden címzettnek külön-külön kellene elküldenünk ugyanazt az üzenetet, ennek azonban ára van: már néhány címzett esetén hatalmas forgalmat generál ez a művelet a hálózatban, nem beszélve arról, hogy a küldő felet is leterheli.

Annak ellenére, hogy a multicast rendszerek alapjait már viszonylag régen lefektették, ez a küldési mód a következő években fog csak igazán elterjedni, mert egyrészt a támogatottsága olyan eszközöket (kapcsolók és forgalomirányítók) igényel a hálózatban lehetőség szerint a teljes útvonalon, melyek képesek a multicast-hoz szükséges speciális irányító- és csoportmenedzsment protokollok futtatására. Másrészt pedig a technikai fejlődés most jutott el arra a

szintre, hogy már sokszor nem elég egy infrastrukturális fejlesztés a hálózatban bővítés céljából, hanem egyéb más módszereket is szükséges használni. Ugyanakkor a többes küldés egy másfajta szemléletet kíván, más módon történik a forgalomirányítás, mint egy unicast üzenet küldése esetén. Különböző problémákat vet fel ez a fajta átviteli mód, például a címzési rendszer adatkapcsolati és hálózati szinten, a multicast forgalomirányítás, vagy az átvitel megbízhatóságának garantálása.

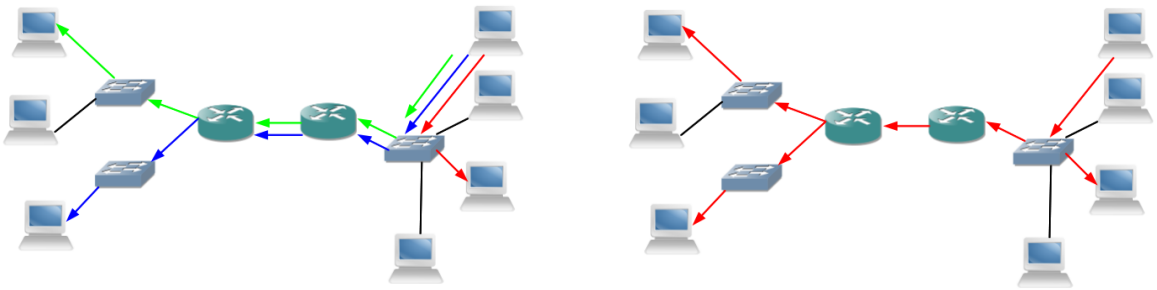
Dolgozatom célja, hogy általános képet adjak az IP multicast átvitel működéséről, az alhálózatokon belüli csoportmenedzsment és az alhálózatok közötti forgalomirányítás folyamatáról. Be kívánom mutatni elsősorban gyakorlati szemszögből a programozás szintjén, hogy hogyan is épül fel egy multicast-ot használó alkalmazás, hogyan zajlik az üzenetküldés, melyek azok a fontos tényezők, melyeket egy általános hálózati programban is figyelembe kell vennünk, mindezt kiegészítve a programozás során szerzett tapasztalatokkal. Különböző módszereket kívánok felvázolni arra, hogy hogyan lehet egy alapvetően nem kapcsolatorientált többes küldést megbízhatóvá tenni alkalmazási réteg szinten, amire leginkább fájlátvitel vagy adatok továbbítása esetén lehet nagy szükség. A dolgozat keretében egy ilyen fájltovábbító program implementációját végzem el, mely kimutathatóan hatékonyabban fogja az átvitelt megvalósítani, mint hasonló, unicast-ban működő alkalmazások.

A dolgozat megértéséhez feltételezett egy alapvető jártasság a számítógépes hálózatokban és a programozásban. A dolgozat megírásához leginkább angol nyelvű szakirodalmat használtam fel, ezért találhatók benne angol szakkifejezések, ezek legtöbbjének a magyar informatikai terminológiában sincs magyar megfelelője.

# 2. Az IP multicasting működése

## 2.1. A többes küldés fogalma

Ha valamit egy hálózatban hagyományos formában, unicast címzéssel küldünk el, akkor azt az információt egyetlen címzett fogja megkapni egy pont-pont típusú küldés során. A többes küldés az adatátvitelnek egy olyan formája, melyben egy pont-multipont jellegű küldéssel a forrás több (akár több száz) címzetthez is eljuttathatja ugyanazt az üzenetet. Erre nincs szükség, ha kevés információt (csomagot) küldünk legfeljebb egy-két címzettnek, mert ez a hálózatot nem terheli le jelentősen. Azonban kézenfekvővé válik a multicast technológia használata, ha sok a címzett, illetve ha a küldendő információt is sok csomagban továbbítjuk. Ugyanezekhez az állomásokhoz a nagy mennyiségű adatot unicast küldéssel továbbítva jelentős hatással lenne a hálózat és a forrás teljesítményére is. Minél több a célállomás és a küldendő adat, annál jobban látszik, hogy a többes küldésnek létjogosultsága van bizonyos esetekben.



1. ábra. Unicast és multicast átvitel.

A többes küldés egyfajta korlátozott szórásnak is tekinthető, melyben mi adhatjuk meg a szórási tartomány kiterjedését egy úgynevezett csoportcím segítségével. Amelyik állomás egy csoport tagja, megkapja a csoportcímre küldött üzeneteket. Egy csoporthoz az egyes állomások speciális protokoll üzenetekkel csatlakozhatnak és léphetnek ki belőle. Ha a továbbítás alhálózatokon keresztül történik, a fogadó állomásoknak jelezniük kell az alhálózatban lévő forgalomirányító felé, hogy meg szeretnék kapni a multicast csoportnak címzett csomagokat. Ezek után a forgalomirányítók minden olyan alhálózat felé kézbesítik a csoporthoz tartozó csomagokat,

ahol van csatlakozott állomás.

Abban az esetben, ha a hálózati eszközök (kapcsolók és forgalomirányítók) nem képesek a speciális multicast irányító- és csoportkezelő protokoll futtatására, a többes küldés lényege elvész, nem lesz hatékony, vagy egyáltalán nem is fog működni. A kapcsolók alapértelmezetten a multicast forgalmat szórásként továbbítják, így minden állomásnak, amely a kapcsolóra van kötve, fel kell dolgoznia az elküldött keretet. Azonban ha működik az eszközön a megfelelő csoportkezelés, adott célcímű multicast csomagot csak azokra a portokra továbbítja, amelyeken található az adott csoporthoz tartozó állomás.

Azért nevezik a többes küldést IP[1] multicasting-nak, mert legmagasabb szinten a hálózati rétegmodell harmadik (hálózati) rétegének szintjén valósul meg a multicast forgalom továbbítása. Egy teljesen hagyományos IP csomagba ágyazott UDP[2] szegmensben küldhető el a multicast információ. Egyedül adatkapcsolati címzés és az IP csomag célcíme speciális, utóbbinál az IP címek erre a célra fenntartott D címosztályába tartoznak (lásd 2.2. fejezet).

Az UDP előnyei miatt gyakoriak az olyan multimédiás szolgáltatások, melyek multicast átvitelt alkalmaznak. Itt nem okoz kijavíthatatlan problémát néhány csomag elvesztése, és vesztés esetén sincs feltétlenül idő arra, hogy a forrás újraadjon, mert ezek leggyakrabban valóidejű szolgáltatások. Ilyenek például VoIP[3] hangátvitel és videokonferencia szolgáltatások, VoD[4] manapság a legelterjedtem multimédia multicast szolgáltatás, az IP-TV digitális műsorátvitel. Konkrét alkalmazások közül kiemelendő a Cisco Systems IPTV streaming szoftvere, vagy a VideoLAN és az mpeg4ip nevű nyílt forrású video streaming szoftverek. Léteznek az MBone-on[5] keresztül is működő alkalmazások, melyek az interneten keresztül is elérhető szolgáltatásokat nyújtanak többes küldést használva.

[1] Internet Protocol
[2] User Datagram Protocol
[3] Voice over Internet Protocol
[4] Video-on-demand
[5] multicast gerinchálózat

## 2.2. Címzési rendszer

### 2.2.1. Hálózati címek

A D osztályú IPv4 címek első 4 bitje 1110 értékű, így a multicast csoportcímek a 224.0.0.0 és 239.255.255.255 közötti tartományból kerülnek ki. A multicast IP címblokkokat nem az IANA[6] szervezete osztja ki az unicast címekkel ellentétben, hanem ún. állandó címeket definiál bizonyos tartományokban, mely csoportok előre definiált funkcióval rendelkeznek. Ezzel ellentétben a tranziens címmel rendelkező csoportokat külön kérést küldve, dinamikusan lehet létrehozni, és addig léteznek, amíg van csatlakozott tag a csoportban. Az IETF[7] szervezete határozza meg egy dokumentumban [7], hogy hogyan osztja fel az egész multicast címteret. A leggyakrabban használt címblokkokat IPv4 esetén a táblázat szemlélteti.

| **Címtér** | **Aggregált cím** | **Csoport leírása** |
|---|---|---|
| 224.0.0.0 - 224.0.0.255 | 224.0.0.0/24 | Helyi vezérlő címblokk |
| 224.0.1.0 - 224.0.1.255 | 224.0.1.0/24 | Internetes vezérlő címblokk |
| 224.2.0.0 - 224.2.255.255 | 224.2.0.0/16 | SDP/SAP címblokk |
| 232.0.0.0 - 232.255.255.255 | 232.0.0.0/8 | Forrás alapú multicast blokk |
| 233.0.0.0 - 233.255.255.255 | 233.0.0.0/8 | GLOP címblokk |
| 239.0.0.0 - 239.255.255.255 | 239.0.0.0/8 | Adminisztratív hatókörű címek |

1. táblázat. IPv4 multicast címtartományok.

A helyi vezérlő multicast tartomány címei helyi hatókörrel rendelkeznek, azaz az alhálózat forgalomirányítói nem továbbítják a külvilág felé az ilyen célcímű csomagokat. Ebben a blokkban többek között a 224.0.0.1 minden hosztot, a 224.0.0.2 minden forgalomirányítót, a 224.0.0.5 az összes OSPF[8] protokollt futtató forgalomirányítót azonosítja. A második tarto-

[6] Internet Assigned Numbers Authority
[7] Internet Engineering Task Force
[8] Open Shortest Path First

mányba, a nem helyi vezérlő címek közé tartozik például a Cisco multicast forgalomirányítók randevúpont felfedezés (224.0.1.40) és randevúpont bejelentés (224.0.1.39) csoportcíme, az ilyen üzenetek már továbbíthatók a forgalomirányítók által is. Az MBone-on is működő katalógusnyilvántartó (SDR) program használja a 224.2.0.0/16 címblokkot, ahol a jól ismert címekre bejelentéseket küldenek a kliensprogramok, így például publikus videostreamingekre lehet csatlakozni, majd a tartományból dinamikusan választott címeken folyik a streaming. Forrás alapú multicast forgalom továbbítható a 232.0.0.0/8 tartományban (lásd 2.6. fejezet). Az ún. GLOP címblokk minden autonóm rendszerhez azonos számú multicast címet rendel hozzá, ezek használatáról az autonóm rendszerek maguk rendelkeznek. Képzése a 16 bites autonóm rendszer azonosítóval történik. [6] Az IP cím első bájtja 233, a második és harmadik bájt lesz a 16 bites azonosító, az utolsó pedig a csoportcímek megkülönböztetésére szolgál az autonóm rendszeren belül. Az adminisztratív hatókörű blokkot további tartományokra osztották, így olyan multicast csoportcímekhez juthatunk, amelyeknél megállapítható, hogy adott célcímű IP csomag milyen hatókörben terjedhet a hálózaton (lásd 2.5. fejezet).

Multicast tartományoknak az IPv6-os világban is megvan az IPv4-es megfelelőjük. IPv6-ban azonban nem létezik a szórás fogalma, amely eddig csökkentette a hálózat hatékonyságát, erre a funkcióra a többes küldés használható. A 16 bájtos cím első bájtja csupa 1-es, ez jelzi, hogy multicast címről van szó. A következő bájt első négy bitje 0000, ha az IANA által kiosztott állandó a csoportcím, 0001, ha tranziens. A bájt utolsó 4 bitje határozza meg a hatókört (lásd 2.5. fejezet). A fennmaradó 14 bájt magát a csoportot azonosítja. [8] [9]

### 2.2.2. Fizikai címek

Az Ethernet közvetlenül támogatja multicast keretek küldését a hálózatra. Ehhez az szükséges, hogy egy szabványos módszerrel leképezzük a D osztályú IP címet az Ethernet keret cél fizikai címére. [10] Egy Ethernet hálózati kártya alapértelmezetten csak a saját fizikai címére érkező keret tartalmát adja tovább a felsőbb rétegek felé, azonban multicast esetén a csatlakozott állomások meg kell, hogy kapják azoknak a csoportoknak küldött a kereteket, amelyekhez

| **IPv6 cím** | **Csoport leírása** |
|---|---|
| FF01:0:0:0:0:0:0:1 | Minden csomópont egy gépen belül |
| FF02:0:0:0:0:0:0:1 | Minden csomópont az alhálózatban |
| FF02:0:0:0:0:0:0:2 | Minden forgalomirányító az alhálózatban |
| FF02:0:0:0:0:0:0:5 | Minden OSPF forgalomirányító az alhálózatban |
| FF05:0:0:0:0:0:0:127 | Cisco randevúpont bejelentések a telephelyen belül |
| FF05:0:0:0:0:0:0:128 | Cisco randevúpont felfedezések a telephelyen belül |
| FF02::1:FFxx:xxxx | Csomópont-megszólítási csoportcím (xx:xxxx a csomópont eredeti IPv6-os címének utolsó 24 bitje) |
| FF3x::0/96 | Forrás alapú multicast blokk |
| ... | ... |

2. táblázat. IPv6 multicast címtartományok.

csatlakoztak.

A multicast IPv4-es cím utolsó 23 bitjét használjuk fel a fizikai címben, mégpedig úgy, hogy a fizikai cím első három bájtja mindig 01-00-5E, a következő bit 0, a maradék 23 bit pedig a csoportcím utolsó 23 bitje. Így képezhető le például 224.0.0.1 csoportcím a 01-00-5E-00-00-01 fizikai címmé. Mivel egy D osztályú címből az formátumjelző 4 bit után csak az utolsó 28 bit változik, és ebből csak 23-at képezünk fizikai címmé, ezért előfordulhat, hogy két különböző csoportcím konvertálódik ugyanazzá a MAC címmé, mert 5 bit elvész a konverzió során. Ez az eset áll fenn például 239.255.10.10 és a 233.127.10.10 címek esetén is, mindkettőnek a fizikai multicast címe 01-00-5E-7F-0A-0A. A hálózat adminisztrátorának figyelembe kell vennie a tervezésnél az ilyen címduplikációkat.

IPv6 esetén is hasonlóan működik a megfeleltetés. [11] Itt a multicast fizikai cím hexa 33-33-mal kezdődik, majd a hálózati cím utolsó 4 bájtja kerül a fizikai cím végére. Így lesz például az FF02:0:0:0:0:0:0:1 IPv6-os címből 33-33-00-00-00-01 MAC cím. Hasonlóan az előzőekhez, itt is lehetnek átfedések a konvertált címek között.

Azok az átviteltechnikák, melyek az IEEE 802.2-es szabványt használják (logikai kapcsolatvezérlés), mint például Ethernet, FDDI, Token ring, a fent említett módon képesek megfeleltetni a hálózati címet a fizikai címnek.

## 2.3. Alhálózaton belüli kommunikáció

Egy alhálózatban történő hatékony multicast kommunikációhoz egy (vagy több), multicast segédprotokollokat futtatni képes kapcsoló szükséges megfelelően konfigurálva. Ha az eszköz nem képes ezeket a multicast kereteket szelektíven a megfelelő portjaira másolni, a multicast forgalom az összes porton szórásként továbbítódik, mert a kapcsolótáblában nincs olyan bejegyzés, amely a cél fizikai multicast címével megegyezne, és nem is tudja ezeket megtanulni, mint az unicast címek esetében. Mivel a csoportmenedzsment (IGMP[9] protokoll) üzenetek alapvetően a forgalomirányítóknak szólnak (a küldés is a 224.0.0.2 címre történik az első verziókban), a kapcsolóknak külön figyelniük kell valamilyen módon ezekre az értesítésekre, hogy tudják, mely portjaikon kell továbbítani egy multicast csomagot. Erre a problémára háromféle módszer létezik:

– Statikus hozzárendelés: Egyenként megadjuk a kapcsolóeszköznek, hogy milyen cél csoportcímmel érkező csomagot mely portokra továbbítson. A módszer hátránya, hogy egyáltalán nem skálázható és nem dinamikus. Csak akkor használható, ha a csoport összetétele és a portokra kötött állomások mindig ugyanazok maradnak.

– CGMP[10]: A Cisco eszközök saját csoportkezelő protokollja. Ha egy forgalomirányító valamilyen IGMP üzenetet érzékel, feldolgozza azt, és ez alapján tájékoztatja az általa elérhető kapcsolókat CGMP protokollon keresztül a csoportban bekövetkezett változásokról. A CGMP hátránya, hogy nem mindig működik együtt az összes IGMP verzióval, így az IGMP bizonyos funkcionalitásai elveszhetnek.

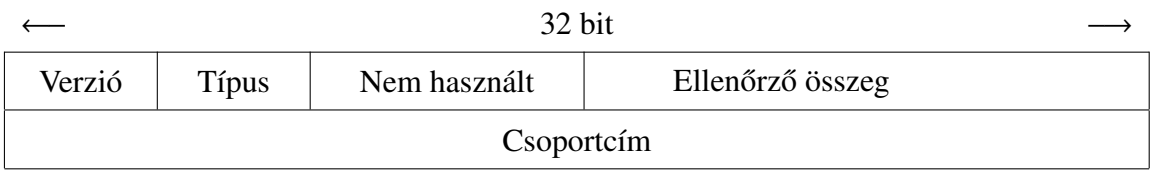
[9] Internet Group Management Protocol
[10] Cisco Group Management Protocol

– IGMP Snooping: A legjobb megoldás az, ha a kapcsoló figyeli a rajta keresztülhaladó IGMP üzeneteket, és a változásoknak megfelelően alakítja a kapcsolótáblát. Ha csatlakozást érzékel egy porton, akkor a kapcsolótáblában a csoport fizikai címét a porthoz rendeli, ha távozást, eltávolítja a bejegyzést kapcsolótáblából. Hátránya, hogy elég nagy terhelést ró az eszközre, mert egy kapcsoló tipikusan csak második réteg szintű kapcsolást végez, nem feladata, hogy egy IGMP üzenet tartalmát elemezze. Továbbá megfelelő IGMP verzió is szükséges hozzá, amely specifikusan csatlakozási üzenetet küld minden hoszttól. (lásd 2.3.3. fejezet)

A forgalomirányítók és IGMP figyelés esetén a kapcsolók értelmezik az IGMP protokoll üzeneteket, és ezen információk alapján tudják, hogy melyik csoporthoz tartozó kereteket vagy csomagokat melyik portra vagy interfészre szükséges továbbítani. A protokoll az IP része épp úgy, mint az ICMP[11], az IP csomag fejlécében 2-es típussal kerül megjelölésre. IPv6 esetében az IGMP már az ICMPv6 részét képezi. Jelenleg 3 verziója létezik az IGMP-nek, melyek különböző többlet funkcionalitásokkal rendelkeznek az újabb verziók felé haladva.

### 2.3.1. IGMP v1

Az első IGMP verziójának leírása az 1112-es RFC dokumentumban található. [10] Az IP csomagon belül az összesen 8 bájtos üzenet felépítése az ábrán látható.




2. ábra. Az IGMP v1 csomagszerkezete.

A verzió mezőben az IGMP verziószáma található 4 biten tárolva. A következő 4 bit az üzenet típusát jelöli, mely lehet

[11] Internet Control Message Protocol

– Host Membership Query: a helyi multicast forgalomirányító tudja lekérdezni a hosztokat, hogy mely csoportoknak tagjai

– Host Membership Report: a kérés üzenetre válaszként küldik az állomások, hogy tagjai egy csoportnak

– DVMRP forgalomirányító üzenet

A nem használt 8 bites mezőt a küldő fél kinulláza, amikor küld, fogadáskor pedig figyelmen kívül hagyja. Az ellenőrző összeg 2 bájtos, melyet a küldő oldal kiszámol és a mezőbe helyez. Az utolsó 4 bájton a csoportcím található, válaszadáskor kerül bele a mezőbe az állomás csoportcíme.

IGMP üzenet fogadásához az állomásnak először csatlakoznia kell a 224.0.0.1 csoporthoz, mert csak így tudnak az erre a címre küldött kérések eljutni hozzá. Ha egy állomás kérést kapott, akkor sorban elküldi a csoporttagságára vonatkozó választ arra a multicast címre, amelyiket éppen jelent, így a csoport összes tagja érzékeli, hogy kik vannak a csoportban. Kérés érkezésekor válaszadás előtt elindít egy-egy válasz késleltetési időzítőt minden egyes tagságára vonatkozóan (kivéve a 224.0.0.1 tagságot), amely véletlen ideig késlelteti a válasz elküldését, így megakadályozható, hogy egyszerre túl sok választ kapjanak a csoport tagjai és a forgalomirányító. Egy kérés vagy válasz 1-es TTL értékkel rendelkezik az IP fejrészben, tehát nem tud kijutni az alhálózatból.

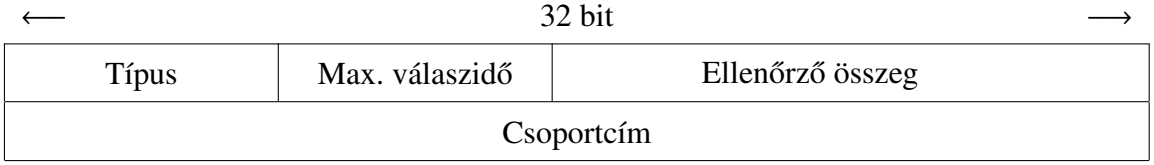
A forgalomirányító egy listát vezet arra vonatkozóan, hogy mely csoportoknak van tagja az ő alhálózatában. Mivel a forgalomirányítónak nem kell tudnia pontosan, hogy mely állomások melyik csoporthoz tartoznak, ezért ha egy csoporttag állomás érzékel egy választ arra a csoportra, amelyiknek ő is tagja, akkor az időzítőjét leállítja, és nem küld választ. Tehát általában a forgalomirányító egyetlen válaszban értesül arról, hogy van az alhálózaton csoporttag.

A forgalomirányítók periodikusan küldenek kéréseket. Ha nem érkezik válasz egy állomástól sem a csoportot illetően, többször is próbálkozik, és egy idő után úgy tekinti, hogy nincs a csoportnak tagja az érintett alhálózaton, tehát ide nem kell eljuttatni a csoportnak címzett multicast üzeneteket a továbbiakban. Az első verzió még nem volt alkalmas a korábban említett

IGMP Snooping vagy CGMP technikákra, mert nem jelenti az összes hoszt, hogy tagja lett egy csoportnak. Az első és második verziót arra lehetett használni, hogy a routerek értesüljenek arról, hogy van-e egy csoportnak tagja az alhálózatában.

### 2.3.2. IGMP v2

A második verziós IGMP [12] üzenet felépítése nagyon hasonló az előzőhöz.




3. ábra. Az IGMP v2 csomagszerkezete.

A típus mezőt összevonták a verziószámmal, így a típus a következők valamelyike lehet:

- General Membership Query: általános lekérdezés, mely csoportoknak vannak tagjai az alhálózatban
- Group-specific Membership Query: csoportra vonatkozó lekérdezés, bizonyos csoporthoz tartoznak-e tagok az alhálózatban
- Membership Report (IGMP v2): válasz a lekérdezésre illetve csatlakozás csoporthoz
- Membership Report (IGMP v1): első verziós válasz visszafelé kompatibilitás miatt
- Leave Group: csoport elhagyása

Új mező a maximális válaszolási idő, mely kérés esetén van kitöltve. Az érték $\frac{1}{10}$ másodperces egységekben mutatja, hogy a hoszt legfeljebb mekkora értékig generálhat véletlen számot, amennyi ideig aztán várakoztatni fogja a válasz elküldését. A General Membership Query-re pontosan olyan algoritmussal ad választ (Membership Report), mint az előző verzióban, figyelembe véve a várakozási időt.

Ahogy egy állomás csatlakozott egy csoporthoz, azonnal Membership Report üzenetet küld a csoport címére, ha ő az első tag a csoportban, az előző verzióban ezt nem tette meg automatikusan. Csoport elhagyásakor Leave Group csomagot küld a 224.0.0.2 címre (összes forgalomirányító). Bár ez utóbbi nem kötelező, de így mindig pontos képe lehet egy multicast forgalomirányítónak arról, hogy egy csoportnak van-e tagja egy hálózati szegmensen, és nem kell megvárni a periodikus lekérdezési időzítőt. Megváltozott az a módszer is, hogy minden forgalomirányító lekérdezéseket indít az állomások felé. Ha több eszköz van, megsokszorozódhat a lekérdezések száma, és ez felesleges forgalmat generál.

A multicast forgalomirányítók IGMP v2 esetén kétféle szerepben képesek működni: lekérdező és nem lekérdező módban, kéréseket csak lekérdező módban lehet küldeni. Egy lekérdező lesz egy hálózati szegmensen, és ezt a forgalomirányítók az alacsonyabb interfész IP cím alapján döntik el. A lekérdező figyeli a beérkező csatlakozási és csoport elhagyási üzeneteket egy-egy csoportra vonatkozóan, és nyilvántartja a csoporthoz tartozó hosztok számát. Ha a hosztok száma nulla lesz egy csoportra vonatkozóan, akkor csoport specifikus lekérdezést küld a csoport címére, hogy maradt-e még állomás, mely a csoporthoz tartozik. Mint az IGMP v1 esetén, itt is csoportonként egy-egy időzítővel, periodikusan általános lekérdezéseket küld, de csak az egyetlen lekérdező multicast router.

### 2.3.3. IGMP v3

A harmadik verziónál [13] a Membership Report csomagszerkezet új mezőkkel egészült ki. A protokoll a forrás alapú multicast (SSM, lásd 2.6. fejezet) támogatására képes, így egy multicast csoporton belül a forrásokat tudja szűrni.

Egy csoportnál csatlakozáskor lehetőség van megjelölni azokat az állomásokat, melyektől a multicast csoportban szeretnénk üzeneteket fogadni. Kilépéskor az IGMP csomagban az új mezők segítségével meg lehet adni, hogy az egész csoportból ki szeretnénk-e lépni, vagy csak forrásokat adunk meg, melyektől a továbbiakban nem szeretnénk multicast csomagokat fogadni. A forrás alapú multicast segítségével az állapotváltó üzenetekben vagy elhagyunk, vagy behívunk bizonyos forrás IP címeket.

Az IGMP v3 már nem nyomja el a csatlakozási és kilépési üzeneteket, mint az előző verziók azért, hogy minél kevesebb üzenettel lehessen tudatni a forgalomirányítóval a csoport tagjainak állapotát. Ez azért is nagyon fontos, mert így már lehetséges az IGMP Snooping technikát használni a kapcsolókon, mert pontosan nyomon lehet követni, hogy melyik állomás csatlakozik vagy hagyja el a csoportot. Ezt segíti az is, hogy az IGMP válaszok a protokoll saját címére, a 224.0.0.22 csoportcímre továbbítódnak.

A skálázhatósági szempontokat figyelembe véve növelték a maximális várakozási időt, és újraadást is építettek a protokollba. A különböző verziók nem mindig működnek teljesen együtt egymással, ezért a hálózat tervezésekor az eszközök képességét számításba kell venni.

## 2.4. Alhálózatok közötti kommunikáció

Ha az IGMP protokoll megfelelően működik a helyi szegmensen, akkor mind a kapcsolók, mind a forgalomirányítók megfelelő információval rendelkeznek ahhoz, hogy tudják, melyik portjukra illetve interfészükre melyik multicast csoport (vagy SSM csatorna) tagjai csatlakoznak.

Így nem okoz gondot a multicast forgalom eljuttatása az alhálózaton belül bárhová, viszont alhálózatok között a routerek még nem tudják, hogy mit kezdjenek a multicast csomagokkal. Szükséges egy irányító protokoll, mely az IGMP-vel összegyűjtött információkat feldolgozza, felépít egy továbbítási fát, és saját formátumában tovább tudja terjeszteni a hálózaton (lásd 3. fejezet). Ha egy ilyen multicast irányítási protokoll jól működik, az összes ilyet futtató forgalomirányító tudni fogja, hogy melyik multicast csoport csomagjait hová továbbítsa. Unicast esetben a célcím alapján lehet dönteni, de multicast forgalomnál azt is kell nézni, hogy a csomag honnan érkezik. Ehhez általában valamilyen továbbítási fát építenek fel magukban egy-egy csoportra vonatkozóan, amely mentén továbbítani lehet a multicast csomagokat az egész hálózatban.

Ha a csoport multicast csomagja eljutott egy cél helyi forgalomirányítójához, az képes továbbítani az alhálózatba, mivel van nála feliratkozott vevő. Az alhálózatban pedig legjobb esetben a kapcsolók a megfelelő portokra továbbítják.

## 2.5. Multicast üzenetek hatóköre

Unicast csomagok továbbításánál ha egy forgalomirányító nem ismeri a célhoz vezető utat, és nem tudja, melyik interfészre továbbítsa a csomagot, akkor az eldobásra kerül. Multicast esetben ez nem mindig van így, mert van olyan régebbi irányító protokoll, tipikusan a sűrű módban működő protokollok (PIM-DM), amelyek először szórásként küldik szét az adatcsomagokat, aztán később levágják a fában azokat a részeket, ahol nincs vevő fél. Így a csomagok a hálózat nem kívánatos részeire is továbbítódhatnak, teljesítménybeli és biztonsági gondokat is okozva ezzel. Ma már fejlődtek ezek a protokollok is, így a lehető legkevesebb felesleges csomagot küldik tovább a forgalomirányítók, de olykor még mindig hasznos lehet, ha a multicast üzenetek terjedését szabályozzuk.

Két módszer létezik arra, hogy egy multicast IP csomag terjedésének hatókörét megszabjuk.

### 2.5.1. TTL hatókör

Az IP csomag 1 bájtos TTL (Time-to-Live) mezőjének értékét használjuk fel (IPv6 esetén ezt a mezőt Hop limitnek hívják). Minden forgalomirányító, mielőtt egy csomagot továbbít, az élettartam értéket eggyel csökkenti. Ha az érték eléri a nullát, a csomag eldobásra kerül. A kezdetleges irányító protokolloknál ez nagyon hasznos volt, mert irányítási hiba esetén nem tudott ugyanazon az útvonalon a végtelenségig körbe-körbe keringeni egy csomag a hálózaton.

Amikor a multicast technológia megszületett, kezdetben a forgalomirányítókban TTL határértékeket definiáltak. Ez azt jelentette, hogyha a TTL nem ért el egy bizonyos értéket, akkor a multicast csomagokat akkor is eldobhatta, ha az érték nagyobb volt nullánál. Így tartották kívül egy hálózatból az olyan multicast csomagokat, melyek nem rendelkeztek elég nagy TTL értékkel. Kezdetben ezek az értékek a helyi alhálózatot (1), szervezetet (16), régiót (63) és a globális hatókört (127) jelentették. A programozóknak kellett azt beállítani az élettartam értékkel, ha mondjuk azt szerették volna, hogy a szervezeten túl ne továbbítódjanak a csomagok. A határroutereken pedig beállították, hogy a 16-nál kisebb TTL értékű csomagokat a router dobja el.

Ma is előnyös tesztelésnél és fejlesztésnél olyan élettartammal küldeni csomagot, amely éppen elegendő, így konfigurációs hibák miatt sem továbbítódhat a hálózat nem kívánt részeire a multicast forgalom.

Míg unicast esetben ha egy csomagot egy forgalomirányító eldobott az élettartam érték nullára futása miatt, és a feladónak egy ennek megfelelő ICMP üzenetet küldött, addig a multicastnál meg kell akadályozni, hogy egy csomag eldobásakor a forgalomirányítók ICMP csomagot küldjenek vissza. Ma már nem jellemző, hogy ez a beállítás problémát okozna a forgalomirányítókban.

### 2.5.2. Multicast cím adminisztratív hatóköre

A cím hatókör a TTL hatókör mellett vagy helyett is használható. Azt használja fel, hogy különböző tartományba eső célcímű multicast csomagok különböző távolságra terjednek a hálózat gerince felé haladva.

IPv4 esetén a hatókörök a következőképpen alakulnak:

- 239.255.0.0/16: helyi hatókör tartomány, a hálózat határrouterei eldobják az ilyen célcímű csomagokat. Unicast címzésnél a privát címtartományokra hasonlít. (10.0.0.0/8, 192.168.0.0/24 stb.) A tartomány lefelé kiterjeszthető.
- 239.192.0.0/16: szervezeti helyi hatókör tartomány, a szervezeten belüli egyediséget biztosít, szabadon feloszthatják és használhatják. A tartomány lefelé kiterjeszthető.
- 224.0.0.0/24: helyi kapcsolati hatókör tartomány. Helyi címek, melyeket a forgalomirányítók nem továbbítanak. A helyi kapcsolati tartomány címei jól ismert címek.
- Az összes többi, ezeken kívül eső, a 2.2. fejezetben is megismert címtartomány globális hatókörű.

IPv6-ban a multicast hatókört a 16 bájtos cím második bájtjának utolsó 4 bitje írja le.

| | |
|---|---|
| 0000 | Fenntartott érték |
| 0001 | Node local – egy gépen belüli |
| 0010 | Link local – egy linken belüli |
| 0101 | Site local – egy telephelyen belüli |
| 1000 | Organization local – egy szervezeten belüli |
| 1110 | Globális hatókörű cím |
| 1111 | Fenntartott érték |

## 2.6. Forrás alapú multicast

Alaphelyzetben a multicast küldés az ún. bármely forrású multicast átviteli módban működik (ASM[12]). Ha egy hoszt egy csoporthoz csatlakozik, akkor utána az összes forrástól megkapja a csoportnak címzett csomagokat. Nem kell megadni csatlakozás előtt, hogy ki lesz a forrás, kitől fogunk multicast csomagokat kapni, a hálózat forgalomirányítói és kapcsolói az összes forrástól érkező csomagot továbbítják a csoporttagoknak.

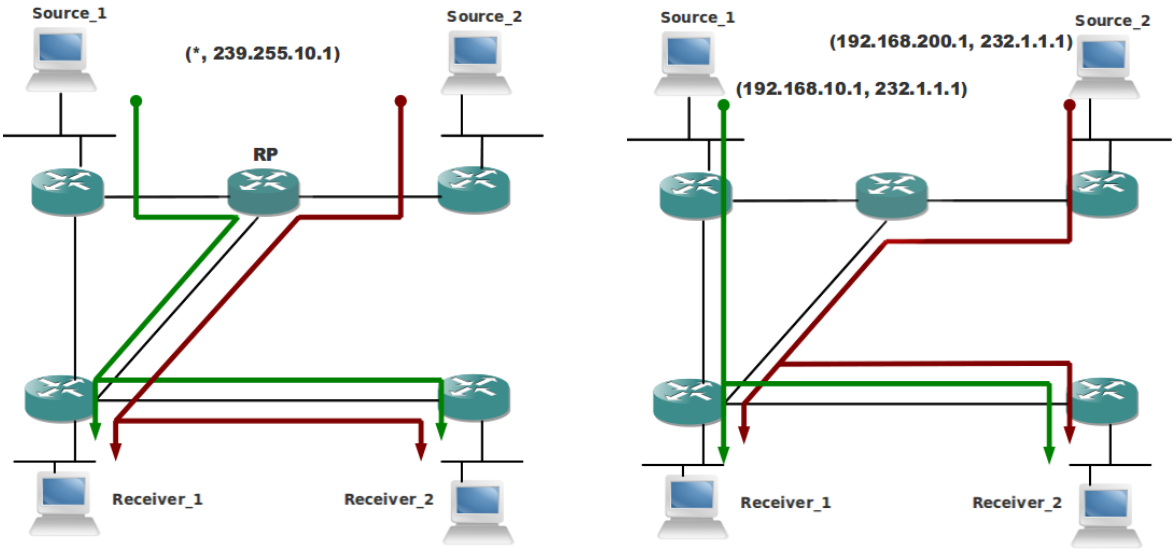
A forrás alapú multicast (SSM[13]) esetén egy multicast csoportot nem csak a csoportcím azonosítja, hanem a multicast forrás (vagy források) unicast címe is. A módszer előnye, hogy az eszközök forráscím alapján is ki tudják szűrni azt a forgalmat, mely bizonyos forrásoktól érkezik. Ezt felhasználva egy multicast csoport forgalma nem jut el a hálózat olyan részeire és olyan fogadókhoz, akik az adott forrásoktól nem is kértek továbbítást. Nagy hátrány, hogy a csoport tagjainak fel kell térképezniük, hogy kitől akarnak fogadni, illetve a csatlakozási üzenetnek nem csak a csoport címét kell tartalmaznia, hanem a forrásokat is meg kell adni. Abban az esetben, ha több forrástól kíván fogadni ugyanazon a címen, több IGMP üzenet szükséges.

Míg ASM esetén egy multicast csoport megnevezésére a (*, G) jelölés szolgál, addig SSM használatakor az (S, G) jelölés vezethető be, mely (S, G) párost csatornának nevezünk. Például a (*, 239.255.10.10) egy ASM csoportot jelöl, azaz minden forrástól fogadni fog a hoszt a 239.255.10.10 multicast címen. A (192.100.100.5, 232.1.1.1) csatorna pedig a 192.100.100.5

[12] Any Source Multicast
[13] Source Specific Multicast

címről érkező, 232.1.1.1 csoport forgalmát fogja továbbítani. Fontos megjegyezni, hogy egy globálisan használt SSM csatorna csoportcíme az erre kijelölt IP cím tartományból kell, hogy kikerüljön (lásd 2.2. fejezet).

Az SSM használatának legnagyobb előnye, hogy képes csak bizonyos forrásoktól származó csomagokat terjeszteni a hálózaton. Így egy nem kívánt csoporttag hiába kezd el nagy intenzitással csomagokat küldeni, ha a fogadó nem arra az SSM csatornára csatlakozott, amelyik tartalmazza a küldő címét, nem fogja megkapni a datagramokat. Az IGMP harmadik verziója teszi azt lehetővé, hogy a csoport címe mellett specifikáltan elküldjük az állomáscímeket, ahonnan fogadni szeretnénk.




4. ábra. ASM és SSM továbbítási fa.

Ha SSM-et használunk, a multicast forgalomirányítás is egyszerűsödhet. ASM-nél a felépített továbbítási fák (lásd 3. fejezet) gyökerének helyén egy kitüntetett szerepű, középponti forgalomirányító állt, például ritka módban működő protokollok esetén. Itt az adott csoport összes forgalma ezen a ponton haladt keresztül, a forrásoknak és a vevőknek a középponti router felé kellett a csatlakozási üzeneteiket küldeni. Ezen keresztül pedig mindegyik forrástól a cél felé egy fát kellett felépíteni. SSM használatakor mivel előre meg kell határozni a forrásokat, egyik félnek sem kell a középpont felé hirdetni vagy csatlakozni, hanem egyszerűen csatornára

lehet feliratkozni, így nincs is szükség a középpontra. A csoport- és forráscím alapján a céltól a (specifikált) forrás felé egy legrövidebb út algoritmussal a forgalomirányítók felépítik a fordított továbbítási fát egy csatornára vonatkozóan. A fenti ábrán látható, hogyha nincs randevú pont, az egyes célt eggyel kevesebb ugráson belül érheti el az egyes forrásból érkező multicast adatforgalom.

Belátható az is, hogy forrás specifikus multicast továbbításnál címeket spórolhatunk meg. Míg ASM-nél a 232.1.1.1 cím egy csoportot jelöl, addig ugyanezen a címen akár több csatorna is azonosítható a küldő címével az SSM használatával, például a (192.168.100.5, 232.1.1.1) és a (192.100.111.10, 232.1.1.1) két különböző csatornát jelöl, bár a csoportcímük ugyanaz. Gyakorlatilag minden hosztnak az egész SSM címtér a rendelkezésére áll, nem fognak a használt (S, G) csatornák más csatornákkal ütközni, hiába ugyanaz a G csoportcím.

IGMP v3 támogatás szükséges az eszközökben és a rendszerek szintjén, hogy a forrás alapú multicast működjön, de ma már ez megvalósul a modern operációs rendszerekben. Applikációs szinten, programozáskor is másként kell kezelni egy SSM továbbítást, mint egy ASM-et. (lásd 5.2. fejezet)

# 3. Multicast irányító protokollok

Az előzőekben leírtak szerint egy alhálózatban működne a multicast küldés, azonban külön alhálózatban, egymástól több ugrásra lévő állomások nem tudnak többes küldéssel kommunikálni egymással. Valahogyan a forgalomirányítóknak továbbítaniuk kell a multicast csoportok csomagjait a forrásoktól a célok alhálózatai felé. Erre a célra multicast irányító protokollokat futtatnak, melyek különböző algoritmusokat használva, továbbítási fákat építve megállapítják, hogy hová kell egy csoport forgalmát eljuttatni.

Alapvetően kétfajta továbbítási fát építhetnek a forgalomirányítók:

– Forrás alapú fák: Hatékony útvonalak létrehozását biztosítja, de erőforrás igényes, mert általában minden csoport minden küldőjéhez külön továbbítási fa épül.

– Osztott fák: egy speciális szerepű router (középpont) különböztethető meg a többitől, amelyen egy csoport forgalma áthalad. Kevesebb erőforrást követel meg, ugyanakkor nem mindig optimális az általa létrehozott útvonal. (lásd 4. ábra)

## 3.1. Sűrű módú protokollok

Az egyszerű elárasztásos módszer nagyon alacsony hatékonysággal rendelkezik, mert habár az összes forgalomirányító megkap egy elküldött multicast üzenetet, de a küldés csomagduplikációt, továbbítási hurkokat eredményezne, végül az egész hálózat működésképtelenné válna a rengeteg csomagtól. Ezért ez önmagában nem is használható módszer, de speciális mechanizmusokkal növelni lehet a továbbítás hatékonyságát.

Amikor egy csoportban egy adó adni kezd, az alhálózat routerei speciális elárasztásos módszerrel küldik szét a csomagokat a hálózat többi része felé. Ha valahol egy router ugyanazt a csomagot több interfészén is megkapja, ekkor csak azt az egyet veszi figyelembe, amelyiken a legjobb unicast útvonallal rendelkezik vissza a forrás felé. Ez lesz a visszafelé továbbító (RPF[14]) link. Ha ezen a linken érkezett a csomag, azt az összes interfészén elküldi, kivéve, amelyiken érkezett. Ha ettől különbözőn, akkor eldobja a csomagot. Ezt az RPB[15] algoritmust felhasználva a hálózat legtávolabbi pontját is eléri a multicast csoport forgalma hurok kialakulása nélkül. Azért nevezik az algoritmust fordított útvonalúnak, mert a fa nem a gyökerétől, hanem a levélelemektől, a cél routerektől épül fel a forrás felé, de a forgalom ennek ellentétesen továbbítódik. Minden egyes forrásra vonatkozóan felépül egy fordított továbbítási feszítő fa a csoportban (forrás alapú fa), melynek gyökere a forráshoz tartozó router, levelei a hálózat összes többi routere. Ennek a módszernek a belátható hátránya, hogy alaphelyzetben azok a routerek is megkapják a csomagokat, amelyeknek már nem szükséges továbbítaniuk, mert a továbbítási irányokban nincs a csoportra feliratkozott vevő. Az RPB nem veszi figyelembe a multicast címinformációkat.

Ennek megoldására ha egy csomópontokhoz kapcsolódó forgalomirányító úgy érzékeli,

[14] Reverse Path Forwarding
[15] Reverse Path Broadcasting

hogy nincs a csoportban vevő egyik interfészén sem (tehát nem kapott IGMP join üzenetet), akkor a forrás felé levágási üzenetet küld. Ezeket a közbülső routerek is megkapják, így már nem fogják olyan irányokba továbbítani a csomagokat, ahol nincs is vevő. A továbbítási fát ezek a levágási üzenetek olyan minimális méretű fává vágják vissza (fordított legrövidebb útvonalú fa – RSPT[16] vagy SPT), amelynek levélelemei kizárólag azok a routerek, melyeknek vannak feliratkozott vevői. Ez az RPB módosítása, az RPM[17], más néven az "elárasztás és tisztít" algoritmus. A legtöbb sűrű módú protokoll az RPM-et használja.

A sűrű módban működő multicast irányító protokollok hátránya egyrészt az, hogy időnként elárasztással továbbítják a csoport forgalmát a routerek között. Másrészt az autonóm hálózat összes multicast routere számon kell, hogy tartsa a csoportokat (az (S, G) párost), nem elég csak azoknak, amelyeken a forgalom keresztülhalad.

### 3.1.1. Distance Vector Multicast Protocol (DVMRP)

A DVMRP az unicast IP forgalomirányításnál használt RIPv2[18] protokollhoz hasonlóan egy távolság vektort használ. A router ezeknél a protokolloknál nem rendelkezik teljes képpel a hálózat topológiájáról, csak a szomszéd routerek által küldött információkat használja fel. Ugyanúgy az ugrások (hop-ok) számát használja távolságként és egyetlen metrika adatként, mint a RIP. A második verzióhoz hasonlóan a multicast protokoll is támogatja az osztály nélküli IP címeket és így a változó hosszúságú alhálózati maszkokat is. A DVMRP saját irányító táblát épít fel arra is, hogy meghatározza az útvonalat vissza a forráshoz.

Elárasztással, vágással és ún. graft-ok küldésével végzi a feszítő fa kezelését és ezzel a multicast forgalomirányítást.

– Első lépésben az RPB algoritmussal elárasztja a hálózat forgalomirányítóit a multicast adatforgalom első csomagjával. Ez feleslegesnek tűnhet, hiszen előfordulhat, hogy az egész hálózatban csak 1-2 routernél jelentkeznek vevők. Az elárasztás a hálózatban kor-

[16] Reverse Shortest Path Tree
[17] Reverse Path Multicasting
[18] Routing Information Protocol version 2

látozható az IP fejrész TTL értékével úgy, hogy a DVMRP interfészeken egy határértéket adunk meg, és az adott interfész csak ettől nagyobb élettartam értékkel rendelkező csomagokat küld tovább.

– Ha egy levél szintű kijelölt forgalomirányítónak (amely IGMP-t futtat az állomásokat tartalmazó interfészén) a csoporthoz tartozó IGMP táblája üres, levágási ("prune") üzenetet küld a forrás felé az ("upstream") SPF interfészen, hogy nem érdekeltek a csoport üzeneteinek fogadásában. Néhány perc múlva a levágott ág "újranő", azaz ismét továbbítani fogja a forgalomirányító az üzeneteket az interfészen, hiába kapott előtte levágási üzenetet, így adva lehetőséget az újonnan csatlakozottaknak, hogy ők is fogadják a csoport üzeneteit. Ha a forgalomirányító mégsem érdekelt, ismét levágási üzenetet küld.

– Amennyiben egy forgalomirányító nem akarja kivárni, amíg ez az automatikus továbbítás megtörténik, graft üzeneteket küldhet, mely által az ágon azonnal továbbítják a routerek az adatfolyamot.

A DVMRP megvalósításánál valójában nem kerül törlésre egy ág az irányítási adatbázisból, csupán egy mezőt tartanak fenn arra, hogy jelezze, hogy egy ág milyen állapotú. Ezzel a plusz információval processzoridő takarítható meg, mert nem kell a teljes procedúra alatt a bejegyzéseket ki-be mozgatni az adatbázisból az összes routeren. Az így kialakított, dinamikus fa egy legrövidebb út fa. (SPT)

Ahhoz, hogy egy DVMRP forgalomirányító felfedezze a szomszédos DVMRP forgalomirányítókat, "Hello" üzeneteket küld a 224.0.0.4 címre (összes DVMRP router), és a DVMRP szomszédokkal periodikus időközönként kicseréli az irányítási tábláját. Hátránya, ahogy a RIP-nek is, hogy időbe telik, amíg egy link vagy topológia változásnak elterjed a híre a routerek között.

Az irányító tábla mellett egy továbbítási táblát is fenntartanak, melyben az (S, G) forrás és csoport párost, a csoporthoz tartozó bejövő és kimenő interfészeket és a korábban említett, az ág státusz mezőt tárolja. A továbbítási táblából kiolvasva a megfelelő bejegyzést képes továbbítani a router vonali kártyája egy multicast csomagot. Ha egy multicast csomag nem azon az inter-

fészen érkezett be, mely a továbbítási táblában bejövőként szerepel, a csomag eldobásra kerül, ellenkező esetben az összes megjelölt kimeneti interfészen elküldi. A DVMRP szomszédsági viszonyok létrejötte után a forgalomirányítók tudni fogják, hogy melyek azok a kimenő interfészek, melyeken más routerek optimálisan érik el a forrást rajta keresztül, így nem terheli feleslegesen a szomszédos, de nem "downstream" routereket.

Ezt a protokollt elég könnyű implementálni, viszont a használata sok memóriát igényel, hogy karbantartsa az irányító- és továbbítási táblákat nagy hálózat esetén. Kisebb infrastruktúrában azonban jól használható.

### 3.1.2. Multicast Open Shortest Path First (MOSPF)

Kapcsolatállapot alapú, az OSPF irányítási protokoll multicast változata. Bár a forgalmat nem továbbítja szórásként, de azért került a sűrű protokollok közé, mert az útvonal- és csoport hirdetéseket az autonóm rendszer összes routere megkapja. Az OSPF routerek kezdetben elárasztással elküldik egy üzenetben, hogy mely szomszédokkal rendelkeznek. Később csak akkor küldenek hirdetést, ha valami változást történt a hálózat topológiájában. A hirdetések segítségével az összes router felépíti a hálózat topológiáját, és ez alapján Dijkstra algoritmussal kiszámolja az egyes célokhoz vezető optimális útvonalat, ami alapján utána gyorsan dönthet, ha irányítania kell egy csomagot.

A protokoll multicast változatában az ilyen üzenetekben azt is jelzik, hogy mely csoportnak vannak csatlakozott vevői, mely az IGMP tábla alapján megállapítható. Ha már a router tudja, hogy hogyan épül fel a hálózat, és hogy hol találhatók bizonyos csoportok vevői, akkor fel tud építeni egy multicast továbbítási táblát az unicast irányítási tábla mellett minden csoportra.

Az MOSPF hátránya, hogy nem skálázható. Számára a multicast adó nem jelzi implicit módon, hogy hol van a forrás, mint a DVMRP esetében, hanem az explicit módon küldött irányító protokoll üzenetek mondják meg, hogy hol vannak a vevők. Az MOSPF forgalomirányítóknak azokat az állapot üzeneteket kell tárolni és küldeni, hogy hol nincs vevő, és ez az információ halmaz egy hálózatban nyilvánvalóan a routerek számával növekszik.

### 3.1.3. Protocol Independent Multicast – Dense Mode (PIM-DM)

A protokollfüggetlen multicast irányító protokollokat azért fejlesztették ki, hogy lecserélje a sűrű módú protokollokat. Azért van a PIM-nek mégis sűrű módú protokollja is, hogy ha mégis előnyösebb sűrű módot alkalmazni, akkor alternatívát nyújtson a DVMRP vagy az MOSPF helyett.

A PIM protokoll hasonló DVMRP-hez, RPM-et alkalmaz, elárasztás után levágja a szükségtelen ágakat a továbbítási fából a levágási üzenetek segítéségével. Alapértelmezetten 3 percenként kezdi újra az elárasztást a nem aktív ágakon.

Van két fő különbség a DVMRP-hez képest. Az egyik, hogy mindkét módban a forráshoz vezető legrövidebb út interfész ellenőrzését a meglévő unicast irányítási tábla alapján végzi, és nem épít külön irányítási táblát azért, hogy megtalálja a legjobb útvonalat vissza a forrás felé. Azért hívják protokollfüggetlennek, mert bármilyen unicast irányító protokoll használható, hogy az unicast irányító táblát felépítse. A másik különbség, hogy a DVMRP megpróbálja elkerülni a szükségtelen csomagok küldését azoknak a szomszédoknak, akik azután úgyis levágási üzenetet küldenének, mert nem rajtuk keresztül vezet a legjobb útvonal a forráshoz. Egy DVMRP router által megállapított kimeneti interfészek csak azok a "downstream" routerek lesznek, amelyek az adott routert használják arra, hogy elérjék a forrást. A PIM-DM nem válogatja ki, hogy a fában lefelé vezető úton kinek lehet potenciális tranzit routere a forrás felé, és az összes PIM-DM routernek küldeni fogja a csomagokat a kimeneti interfészeken.

## 3.2. Ritka módú protokollok

Az osztott fák felhasználhatók a ritka módú protokollokhoz, mert hatékonyan tudnak működni, amikor a csoporttagok viszonylag kis területen koncentrálódnak, azaz kevés alhálózatnak van tagja egy-egy csoportnak. Ahelyett, hogy elárasztanánk a hálózatot, majd pedig visszavágnánk a fa nem aktív ágait, a vevőállomások explicit csatlakozási üzeneteket küldhetnek egy kitüntetett szerepű routernek, amely a továbbítási fák közös pontját fogja képezni. A különböző csoportoknak különböző középpontjuk lehet. Amikor az adatforgalom megindul, az adóállomá-

sok is ennek a routernek küldik a multicast csomagjaikat, a vevők pedig innen kapják. Ritka módnál nem kell az összes forgalomirányítónak tárolnia az (S, G) aktív párosokat az autonóm hálózatban, elég, ha csak éppen az útvonalon továbbító routerek ismerik ezeket.

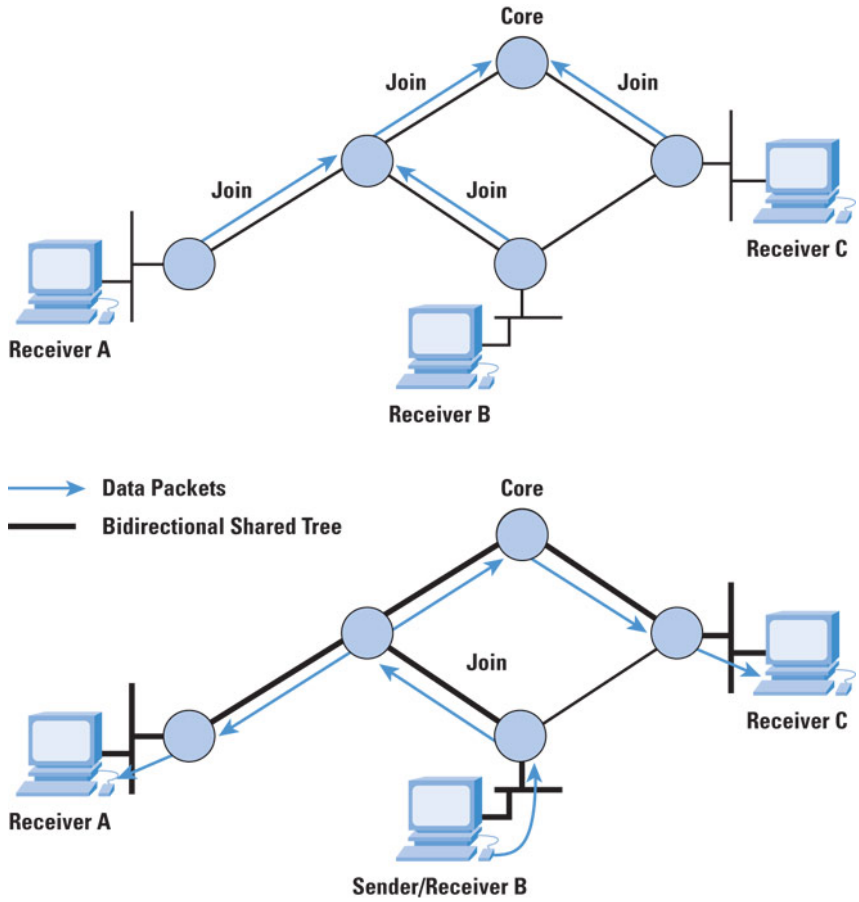
Több probléma is felmerül a középpontú fák használatakor. Egyrészt a fa középpontját egy unicast címmel lehet azonosítani, és ezt valamilyen módon meg kell feleltetni egy-egy csoportnak. Másrészt úgy kell meghatározni ezt a középpontot, hogy a lehető legoptimálisabbak legyenek azok az útvonalak, melyek a forrásoktól a célállomások felé továbbítják az adatfolyamot. Ha adott a középpont címe, valamilyen módon köré kell szervezni az állomásokat, hogy az egy fát alkosson.

### 3.2.1. Core Based Tree (CBT)

A mag alapú fa protokoll a legkorábbi és legegyszerűbb osztott fa implementáció. Egy állomás csatlakozik egy csoporthoz, melynek hatására a helyi CBT router lekérdezi, hogy mi az adott csoport mag routerének unicast címe, és ennek egy csatlakozási üzenetet küld a csoport címével. Ez ahogy halad a mag felé, áthalad a közbülső CBT routereken is, melyeknél aktiválódik a továbbítási állapot, és visszafelé megerősítést küldenek a megelőző routereknek. Miután az állomások csatlakoztak, kialakul az ábrán látható továbbítási fa, mely két irányban képes a csoport forgalmát továbbítani attól függően, hogy a forrás éppen hol található.

Mivel multicast küldéskor nem feltétlenül szükséges a csoporthoz csatlakozás, ezért ha egy állomás csak adóként működik és nem csatlakozik, nem fogja továbbítani a csoport forgalmát a helyi router, ha nem része a továbbítási fának. Ekkor a forgalom el kell, hogy jusson egy olyan next hop CBT routerhez, amely része a fának, és így az átvitel működőképes lesz.

Fontos megjegyezni, hogy CBT esetén nem megoldott a mag router címének és a csoportcímnek a megfeleltetése, illetve ha nem megfelelően van megválasztva a középpont, kevésbé lehet hatékony. Éppen ezért is kevésbé elterjedt a CBT protokoll, nagyobb méretű hálózatokban nem használják.




5. ábra. Mag alapú továbbítási fa.

### 3.2.2. Protocol Independent Multicast – Sparse Mode (PIM-SM)

PIM ritka módot használó továbbítási középpontját itt randevú pontnak (RP) hívják. A protokoll osztott fákat használ, de forrás alapján is képes legrövidebb útvonalú továbbítási fát (SPT) építeni, ezáltal támogatja az SSM használatát is. A CBT-vel ellentétben itt nem lényeges, hogy egy adóállomás tagja-e a csoportnak vagy sem (illetve, hogy van-e tag az alhálózatban), és a továbbítás alapértelmezetten egyirányú (de a PIM-nek létezik kétirányú változata).

Amikor egy multicast állomás adni kezd, a PIM-SM routernek ismernie kell, hogy melyik RP címhez tartozik a csomag cél csoportcíme. Erre többféle megoldás létezik.

– Legegyszerűbb, ha statikusan beállítjuk a levél routereken a csoportokhoz tartozó RP

címeket. Így azonban később egyenként kellene módosítani a bejegyzéseket, ha a csoport RP címe megváltozik. Ennek a módszernek rossz skálázhatósága.

– A Cisco routerek képesek az Auto-RP protokoll futtatására, melyek jól ismert címekre RP felfedezést (224.0.1.40) és RP hirdetést (224.0.1.39) küldenek. Az RP routereket be kell konfigurálni a csoportok számára. Ezután az RP-k hirdetéseket, a levél routerek pedig felfedezéseket küldenek az ismert csoportcímekre.

– A PIM saját módszere is használható, amely egy ún. bootstrap folyamat. Az összes PIM routernek címezve (224.0.0.13) a beállított RP elterjeszti a hálózatban, hogy ő milyen csoportoknak a randevú pontja.

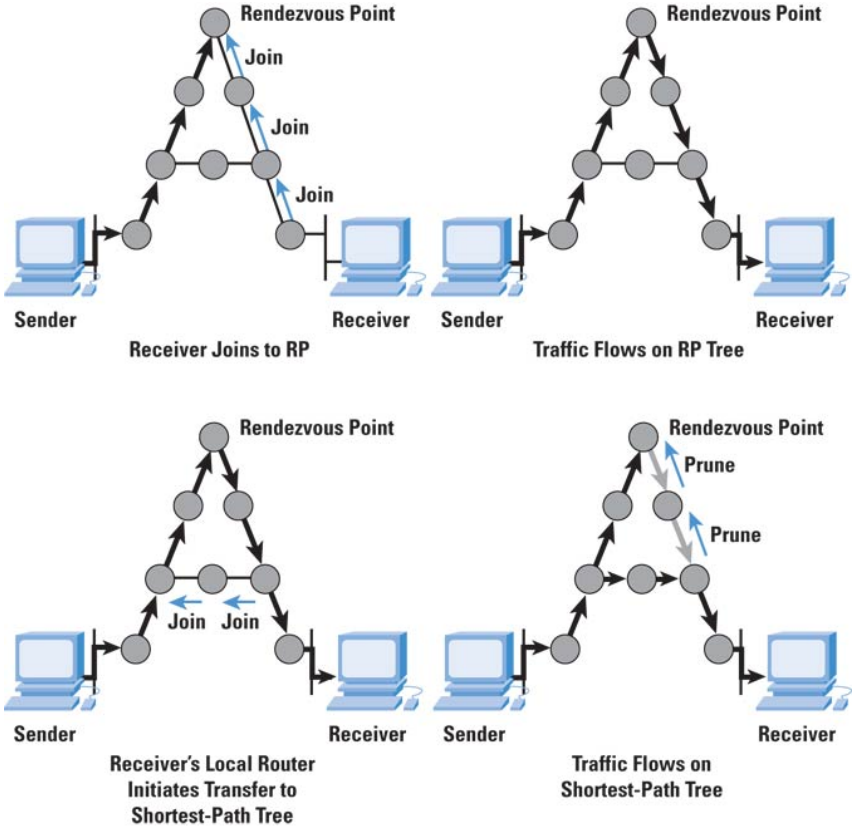
Ha a PIM levél router ismeri az RP címét, az állomástól fogadott multicast IP csomagot egy másik IP csomagba ágyazva egyenesen az RP-nek küldi unicast címzéssel. Így az RP-ig vezető útvonalon nem okoz többletterhelést és adminisztrációs feladatokat a routereken az RP-nek küldött forgalom.

Felmerülhet a kérdés, hogy ha a továbbítási fa egyirányú, akkor hogyan továbbítódik a csoport forgalma vissza a forrás felé. A megoldás az, hogy a randevú pont már az első forrástól kapott csomagnál tudja, hogy a forrás létezik, mert a levél router az első multicast csomagnál forrás-regisztrációs unicast üzenetet küldött a randevú pontnak. Ő pedig innentől meg tudja találni a legrövidebb útvonalat vissza a forrás felé.

A fogadó fél ha csatlakozik egy csoporthoz, a helyi PIM-SM router csatlakozási üzenetet továbbít az RP felé, ha szükséges, és az útvonalon mindegyik router továbbítási állapotba kerül az adott csoportra. Az RP kibontja a beágyazott multicast csomagot, és a továbbítási fa mentén elküldi, majd a legvégén a fogadó állomások megkapják. (lásd az ábra felső részén)

Itt is küldhető olyan üzenet, amely az osztott fa egy ágát levágja, és utána már nem jut el az ágba a csoport forgalma. Erre akkor lehet szükség, ha egy levél router úgy érzékeli, hogy az összes vevőállomás távozott (IGMP leave üzenetekkel vagy időtúllépéssel) a csoportból, vagy ha van egy optimálisabb útvonal, melyen az adatfolyam továbbítható.

A PIM-SM egyik legnagyobb előnye például a CBT-hez képest, hogyha van jobb útvonal




6. ábra. PIM irányítási protokoll továbbítási fa kezelése.

az eredeti osztott fánál, a fogadó levél routere kérheti, hogy egy legrövidebb útvonalú fa mentén közvetlenül kapja a forrástól az adatfolyam csomagjait, és ne az RP-n keresztül a hosszabb útvonalon. Ekkor első lépésként csatlakozási üzeneteket küld a forrás felé, és a routerek továbbító módba kerülnek az új fa mentén (lásd az ábra bal alsó részén). Végül, hogy ne kapja meg kétszer ugyanazokat a csomagokat a cél levél routere, az osztott fán az RP felé levágási üzenetet továbbít (lásd az ábra jobb alsó részén). Így létrejön egy legrövidebb útvonalú fa (SPT), melyen keresztül optimálisan továbbítódik a multicast forgalom, mint ahogyan azt a sűrű módú protokollok esetében láttuk.

### 3.2.3. Bidirection Protocol Independent Multicast (BIDIR-PIM)

A PIM protokoll harmadik változata, mely ritka módban képes a fa mentén kétirányú multicast forgalom továbbításra. A PIM-SM-től abban különbözik, hogy milyen módon juttatja el a forgalmat a forrástól a randevú pontig.

Nem IP csomagba ágyazva küldi a forrás csomagjait a levél router az RP felé, hanem az osztott kétirányú fa mentén. Ebben a típusban nincs lehetőség átállni legrövidebb útvonalú fára, így nem támogatott a forrás alapú multicast küldés sem. Minden RP interfészenként választ egy kijelölt továbbító routert (DF[19]) az RP felfedezésekor, így lehet elkerülni a továbbítási hurkok kialakulását.

A kétirányú fát használó PIM akkor előnyös, ha sok a forrás egy csoportban, viszont a hátránya, hogy a forgalomnak mindenképpen az RP-n keresztül kell haladnia, ami mint láttuk, nem feltétlenül eredményezi a legjobb útvonalat. [16]

## 3.3. Kevert módú PIM protokollok

Legtöbbször a három protokollfüggetlen multicast irányító protokoll valamelyikét (PIM-DM, PIM-SM, BIDIR-PIM) használják egy autonóm hálózatban attól függően, hogy milyen jellegű a forgalom, hogyan helyezkednek az adók és a vevők. Olykor azonban felmerülhet az igény, hogy ezek közül több protokollt használjunk egyszerre egy hálózatban. Ez lehetséges, de egy csoport akkor is csak egy módban továbbíthatja az adatát.

Például a Cisco routerek képesek PIM ritka-sűrű üzemmódban működni, ekkor az összes PIM forgalomirányító ugyanazon "Hello" csomaggal kommunikál, és a különböző módú routerek együtt kell, hogy működjenek egymással, továbbítják egymás információit. Ez azonban már implementáció kérdése, és nem a PIM protokoll szabja meg ezt az együttműködést.

[19] Designated Forwarder

## 3.4. Tartományok közötti multicast forgalomirányítás

A fent ismertetett irányítási protokollok mindegyike csak egy bizonyos hálózat átmérőig használható, mert a sűrű módú protokollok az egész hálózatot elárasztják vagy adatcsomagokkal vagy irányítási hirdetésekkel, a ritka módúak pedig terjesztik a randevú pontra vonatkozó információkat. Egy randevú pont egyébként sem tudja túl sok csoport forgalmát kezelni egyszerre. Ezért ezek a belső irányító (IGP[20]) protokollok csoportjába tartoznak, melyek egy jól meghatározott tartományon belül (autonóm rendszerben) működnek. Kérdés, hogy a tartományok között hogyan irányítsák a routerek a multicast adatforgalmat.

### 3.4.1. Multiprotocol Border Gateway Protocol (MGBP)

A legnagyobb problémát az okozza, hogy a szolgáltatók autonóm rendszereinek határ routerei nem mindig támogatják a multicast protokollokat, és nem biztos, hogy a PIM protokoll például megfelelő útvonalat talál vissza a forrás felé az unicast irányító táblában a tartomány határán.

Mivel ma már legnagyobb részt a BGP-t használják, mint tartományok közötti unicast irányító protokollt, ezért kézenfekvőnek tűnik, hogy ezzel más jellegű forgalom irányító információit is ki lehessen cserélni.

Megszületett a Multiprotocol BGP, amely külön továbbította a multicast irányító táblát. A PIM protokoll pedig már használhatta ezeket az MBGP-vel átvitt speciális útvonalakat, a csatlakozási üzenetek kikerülték a hálózat azon részét, ahol nem működik a multicast funkcionalitás.

### 3.4.2. Multicast Source Discovery Protocol (MSDP)

Még mindig nem megoldott az a kérdés, hogy osztott továbbítási fák esetén a randevú pontokat hogy találják meg a levél routerek, ha azok éppen egy másik tartományban vannak.

A multicast forrás felfedezési protokoll nem jelent ugyan hosszútávú és jól skálázható megoldást, de biztosítja, hogy másik autonóm rendszerben lévő randevú pontok könnyen megtalál-

[20] Interior Gateway Protocol

hatók legyenek, és így PIM-SM protokollal megvalósulhasson a tartományok közötti multicast továbbítás.

Első lépés, hogy a különböző tartományokban lévő, adott csoport randevú pontjai MSDP protokollon keresztül felveszik egymással a kapcsolatot. Normál PIM-SM működésként zajlik a csatlakozás tartományon belül az RP-k felé. Ha az egyik tartományban egy forrás kezd küldeni, a helyi RP normál módon továbbít a tartományában, és a másik tartomány RP-jének "Forrás aktív" üzenetet küld. Erre a másik tartományban lévő RP a jelzett forráshoz csatlakozik, így becsomagolt üzenetben, unicastként továbbítódik a forrástól a másik tartomány randevú pontjáig az adatcsomag, amely a saját tartományában a PIM-SM szerint elterjeszti a kibontott csomagot.

A protokoll hátránya, hogy az összes tartomány összes randevú pontjának tájékoztatnia kell a többit minden egyes küldő esetén. Nagy hatással lehet az egész mechanizmus működésére, ha világszerte sok a forrás. Másrészt nem szabad, hogy a küldő első csomagjai elvesszenek, így azokat is el kell juttatni a "Forrás aktív" üzenettel együtt az összes RP-nek a többi tartományba, mert potenciális fogadók lehetnek, de csak a jelzés után csatlakoznak, ha szükséges. Ez is nagyban lerontja a működés hatékonyságát.

# 4. Multicast alkalmazás vizsgálata

Sokféle hálózati alkalmazás létezik, ezek közül manapság legelterjedtebbek a webes alkalmazások, melyek leginkább Java nyelven íródtak, és egy internetező nap mint nap találkozik velük, még ha nem is tud róla. A natív (nem webes, asztali) hálózati programokat is többféle programnyelven meg lehet írni, mert minden modern programnyelvhez létezik funkciójában egységes hálózati kommunikációt biztosító programkönyvtár. A dolgozat létrejöttéhez és a programozáshoz a C nyelvet választottam jó teljesítménye és a multiplatformos környezet miatt.

Egyik fontos jellemzője azonban a mai hálózatos alkalmazásoknak, hogy képesek legyenek valamilyen módon több csatornán hatékonyan komunikálni. Ha egy szál van a programban, és két fogadás között több, időigényes művelet is történik, akkor csak a socket-ek folyamatos

lekérdezésével tudjuk azt elérni, hogy ne veszítsünk csomagot. Erre C környezetben a select() utasítás használható, melyben socket-ek egy halmaza adható meg egy meghatározott időértékkel együtt. Datagram átvitelt használva az utasítás eredményeképp megkapjuk, hogy van-e a megadott halmazban olyan socket, melyhez tartozó várakozási sorban beérkezett és feldolgozható UDP csomag várakozik. Ha van új csomag, további függvények használatával meghatározható, hogy a socket halmaz mely eleme vagy elemei érintettek. Intervallum megadásakor a select() utasítás legfeljebb ennyi ideig várakozik, ha üresek a várakozási sorok, azután továbblép. Ha várakozási időként nullát adunk meg, akkor a sorok állapotától függetlenül nincs várakozás. Bár az intervallum egészen mikroszekundumos határig megadható, de az az operációs rendszer ütemezőjétől függ, hogy mekkora felbontásban lehet késleltetni a gyakorlatban. Így általában van egy minimális idő, mely alá nem lehet menni. (Windows rendszerekben ez alapértelmezetten néhány milliszekundum)

Az UDP csomagot fogadó függvény a C nyelvben ún. blokkoló függvény, mely nem engedi addig tovább futni a programot, amíg be nem fejezte működését, azaz nem fogadott egy csomagot. Előzetesen socket lekérdezést használva biztosak lehetünk benne, hogyha van csomag a várakozási sorban, akkor a függvény nem fog blokkolódni. Ez azért fontos, mert így nem vesszük el az időt a program más, kritikus részeitől.

Jó megoldás lehet több szál használata, amelyek dedikáltan egy-egy feladatot futtatnak (például csomagot fogadó szál), ekkor azonban meg kell oldani a konzisztens adatcserét a szálak között. Másik megoldás, hogy nem blokkoló típusú socket függvényeket használunk, ekkor nincs szükség a szálak közvetlen kezelésére. Bizonyos nyelvekben lehetőség van arra, hogy a program aszinkron jelzéseket rendeljen bizonyos eseményekhez. Így ha történt valamilyen esemény (csomag érkezett), akkor azt egy megfelelő függvény tudja szinte azonnal lekezelni. Mivel nem állnak rendelkezésre nem blokkoló típusú socket függvények a C nyelvben, ezért több szálat vagy select() függvényt kell alkalmaznunk.

Linuxos környezetben a C nyelv már önmagában képes több szálat kezelni, ezek az ún. POSIX szálak. Ha több szálon programozunk egy bonyolultabb alkalmazást, a program strukturáltságát is meg tudjuk őrizni, mert egyébként nagyban növekedne a komplexitás a programozás

során.

Mivel egy asztali alkalmazás nehezen képzelhető el grafikus felület nélkül, ezért egy többszálú programozást és több platformot (Windows, Linux) támogató, grafikus felületet nyújtó programkönyvtárat használtam a fejlesztés során, ez a GTK+ keretrendszer (`http://www.gtk.org/`, GNU LGPL 2.1 [17]).

## 4.1. Megbízható alkalmazások háttere

A TCP/IP hálózati rétegmodell negyedik rétege, a szállítási réteg képes az adatátvitelt megbízhatóvá tenni az alkalmazási réteg felé. Észleli, ha csomagok vesztek el az adás során, mivel az alsóbb rétegek nem foglalkoznak azzal, hogy hiba esetén újraküldjenek, nincs is sok eszközük a hibák felismerésére. A szállítási entitás érzékelheti a csomagvesztést, és újraküldi a meg nem érkezett csomagokat.

„A szállítási rétegnek köszönhetően a hálózati programozók olyan szabványos eljárásokból álló kódot írhatnak, amely nem függ a fizikai hálózatok megvalósításától, és nem kell amiatt aggódniuk, hogy különböző állomásokon különbözőek az interfészek vagy az eltérések miatt nem megbízható a kapcsolat. Ha az összes hálózat egyforma és hibamentes lenne, ugyanazokat az eljárásokat használhatná a programozó, és azok sosem változnának, akkor valószínűleg nem is lenne szükség a szállítási rétegre. A valós életben azonban a legfontosabb feladat, hogy a felsőbb rétegeket el lehessen szigetelni az alsóbb rétegektől, a hálózat hibáitól és annak fizikai megvalósításától.” [2, 6.1 fejezet] Sajnos szinte egy hálózat sem tökéletes, a csomagok elveszhetnek az átvitel közben, amely a hálózati réteg szolgáltatásában még fel sem tűnik. A szállítási réteg alatt nem foglalkoznak a rétegek azzal, hogy a csomagot újraadják. A szállítási réteg kapcsolatorientált és kapcsolatmentes szolgáltatást képes nyújtani.

### 4.1.1. Kapcsolatorientált szolgáltatás

A kapcsolatorientált megbízható, végponttól végpontig tartó bájtfolyam szolgáltatás, a hálózat tökéletlenségeit, az átvitel hibáit képes elfedni a felhasználó elől, ilyen a TCP[21]. A TCP entitás legelőször felveszi a kapcsolatot a fogadó állomással (ezért kapcsolatorientált), ezután megkapja az elküldendő adatot az alkalmazástól, és széttördeli legfeljebb 64 kilobájtos részekre (a gyakorlatban akkora méretre tördelnek, hogy beleférjen az adatrész az IP csomaggal együtt egy Ethernet keretbe), ezeket az adatrészeket TCP szegmensbe ágyazva IP datagramként küldi el a fogadó félnek. Amikor megérkezik a csomag, az abban lévő TCP szegmenst a fogadó átadja a megfelelő entitásnak, ami ezekből összeállítja a bájtfolyamot, ezzel az alkalmazást megkapja a teljes elküldött adatot. A hálózati rétegben az IP nem garantálja, hogy minden csomag megérkezik, illetve azt sem, hogy sorrendben érkezik meg, ezért a TCP a beérkezett szegmenseket sorrendbe állítja, vagy ha elveszett csomagot érzékel (a küldő fél nem kap nyugtát a fogadó féltől), az időzítő lejártával újraküldi az érintett csomagokat. A sorrendiséget a TCP fejrészében a négy bájtos sorszám segítségével állapítja meg. A TCP adatfolyam-vezérlést is végez (ablakozási technikával), hogy megszüntesse a „gyors adó, lassú vevő” problémáját. A küldés befejeztével a kapcsolatot le kell zárni, ezzel megszűnik a virtuális áramkör, amely a kapcsolatfelvételkor létrejött.

### 4.1.2. Kapcsolatmentes szolgáltatás

Nem megbízható, kapcsolatmentes datagram szolgáltatás az UDP[22]. Egyik nagy előnye a hatékonysága, fejrésze is rendkívül egyszerű, ellentétben a TCP-vel, csak forrás- és célportot, a hosszt és egy ellenőrző összeget tartalmaz. Mivel mindegyik protokoll adategység az alkalmazási réteg alatt rendelkezik ellenőrző összeg mezővel, ezért ha az ellenőrző összeg nem egyezik a kiszámított értékkel az adott réteg szintjén, akkor az nem továbbítja felfelé az beágyazott adatot, így az alkalmazás nem kapja meg a várt csomagot, és nem is értesül arról, hogy valamelyik réteg eldobta. Az alsóbb rétegek a TCP esetén a szállítási rétegre, UDP esetén az alkalmazási ré-

[21] Transmisson Control Protocol

[22] User Datagram Protocol

tegre bízzák csomagvesztés érzékelését és a csomagok újraadását (ha az követelmény). Itt nem épül ki virtuális áramkör a két fél között, a protokoll legjobb szándékú (best effort) kézbesítést hajt végre, azaz csak megcímzi majd elküldi az UDP szegmenst IP csomagba ágyazva a címzett felé, de semmilyen nyugtázást, torlódáskezelést nem végez.

Ha megbízhatóság az elsődleges szempont, akkor a legnagyobb probléma az IP multicasttal, hogy nem áll rendelkezésre két fél, akik között a kapcsolatot ki lehetne építeni, vagy küldés közben az átvitelt vezérelni, mert egy multicast cím címzettek egy csoportját jelenti, és a küldő nem is tud róla, hogy kik azok az állomások, amelyek a csoportba feliratkoztak. Ráadásul rengeteg időbe telne, ha mondjuk egy 1000 tagú multicast küldés esetén az összes állomással három utas kézfogást hajtana végre a kezdeményező fél. A periodikus időközönként elküldött nyugták is problémát okozhatnak, mert túlságosan leterhelné a sok nyugta az adó állomást. Többes küldésnél csak kapcsolatmentes protokoll használható, az UDP, ez viszont önmagában nem biztosítja a datagramok (sorrendhelyes) megérkezését, adatfolyam-vezérlést, torlódáskezelést, nyugtázást, ezért ezeket a funkciókat egy megbízható multicast alkalmazásnál az alkalmazási rétegbe kell beépíteni.

## 4.2. Multicast socket-ek használata

Ahhoz, hogy küldeni és fogadni tudjunk a hálózaton keresztül egy programban, szükséges egy erre alkalmas alkalmazásprogramozási felület (API[23]). Egyik legelterjedtebb a Berkeley UNIX socket-programozási API, mely a BSD operációs rendszer születésével jött létre, és ma már szinte mindegyik operációs rendszerben és programozási nyelvben megtalálhatók az API eljárásai. A socket (foglalat) egy kétirányú kommunikációra képes végpont, melyet egyértelműen azonosít egy hálózati cím (cím típussal, például IPv4 vagy IPv6), egy portszám, illetve a használt szállítási protokoll a kapcsolat típusával (például datagram vagy bájtfolyam).

Mindenfajta kommunikáció első lépése a socket létrehozása. Ezen túl ha fogadni szeretnénk, értesíteni kell az operációs rendszert, hogy az adott címre és (esetünkben UDP) port-

[23] Application Programming Interface

számra érkező csomagokat a programunknak továbbítsa, ez az ún. kötés (bind). Semmilyen kapcsolatfelvételi eljárást nem kell meghívnunk, csak a küldési vagy fogadási eljárást, mert az UDP kapcsolatmentes protokoll. Küldésnél viszont az elküldött csomagok további sorsáról semmilyen információt nem fogunk kapni. Ha a fogadó fél éppen nem áll készen arra, hogy a csomagokat fogadja, vagy továbbítás közben vesznek el, akkor azok nem érik el a célt. Erről a küldő nem értesül, így magától újraküldeni sem tud. Ezért kell ezeket a fellépő problémákat a későbbiekben alkalmazási réteg szintjén kezelni.

Sok esetben, ha a megbízhatóság nem követelmény, például hang- vagy videofolyam továbbítása esetén, az UDP sokkal megfelelőbb szállítási protokoll lehet. Egy-egy elveszett csomag újraadásakor elképzelhető, hogy mire az megérkezik, már nem használható a fogadó oldalon. Az UDP-nek megvan az az előnye, hogy másfajta, akár részleges megbízhatóságot adjunk hozzá az átvitelhez az alkalmazási rétegben, ráadásul a kapcsolatfelépítésre sem kell időt fordítani, mint a TCP-nél.

A példáknál és a programozás során a használt címek IPv4-es címek. Mivel a legtöbb mai operációs rendszerben implementálva van az IPv6-os protokollcsomag (stack), így az átállás a programozásban csak a címrendszert érintené, a multicast működése ugyanaz maradna.

### 4.2.1. UDP üzenetek küldése

A küldés folyamata a következő részekből áll:

1. Socket létrehozása
2. A csomag hatókörének beállítása TTL értékkel (opcionális)
3. Az adat elküldése a socket-re
4. Socket lezárása

Socket-et úgy hozunk létre, mint unicast esetben. Linuxban a szükséges header fájlok a <sys/types.h>, <sys/socket.h> és <netinet/in.h>, amelyek tartalmazzák a szükséges függvények deklarációit és a bennük használható paraméter konstansokat. Windowsban a Winsock API használható socket programozásra (winsock.h). A socket létrehozó függvény visszatérési

értékként egy socket fájlleíró (file descriptor) egész értéket (int) ad vissza, melyet a továbbiakban küldésre, fogadásra és beállításra használunk.

```
socket(socket_file_descriptor, PF_INET, SOCK_DGRAM);
close(socket_file_descriptor);
```

Ha nem sikerül létrehozni a socket-et, akkor a visszatérési érték -1. A hívás paraméterei mind a header-ökben definiált konstans értékek. A protokoll család IPv4 esetén PF_INET, a szállítási protokoll típus UDP esetén SOCK_DGRAM, és mivel multicast-nál csak UDP használható szállítási protokollként, ezért IPPROTO_UDP értéket állítunk be a socket-re. Ha már nincs szükség a socket-re, lezárjuk, mely leállítja a kommunikációt és felszabadítja a tárterületet. Ez egyébként a program befejeztével automatikusan megtörténik.

Küldés előtt meg kell adnunk, hogy hova címezzük a datagramot. A cél socket címe az IP címből és a portszámból áll, melyet a sockaddr_in struktúrával tudunk megadni. A címváltozóhoz allokált területet mindig érdemes 0-kal feltölteni használat előtt. Deklaráció után a következő struktúra-tagokra hivatkozva tudjuk beállítani a cím paramétereit:

```
struct sockaddr_in to_socket_address;
to_socket_address.sin_family = AF_INET;
to_socket_address.sin_addr.s_addr = inet_addr("239.255.100.1");
to_socket_address.sin_port = htons(25000);
```

Deklarálás után beállítjuk, hogy a cím egy IPv4-es cím, majd egy függvény a multicast IP cím sztringjét az IP cím bináris reprezentációjává (struct in_addr) alakítja. Ugyanilyen funkcióval rendelkezik az inet_aton() (adress to network). Ezeknek ellentéte, az inet_ntoa() (network to adress) a struct in_addr belső struktúrából visszaadja a pontozott decimálisokból álló sztringet. Ezek a függvények már elavultak, mert egyrészt nem támogatják az IPv6-ot, másrészt például az inet_ntoa() egy statikus bufferrel tér vissza, így egy többszálú programban, ahol a szálak között megosztott a memória, nem biztonságos a használata. Helyettük konzekvensen az inet_ntop() és inet_pton() függvények használandók. A port beállítását a htons() függvénnyel (host-to-network short) kell végezni a platform-kompatibilitás miatt (big-endian/little-endian konverziós problé-

ma). Ezeket és további konverziós függvényeket az <arpa/inet.h> header tartalmaz. Eddig azon kívül, hogy a multicast tartományból állítottunk be címet a küldéshez, semmilyen multicast-specifikus műveletet nem végeztünk.

Küldéskor a TTL alapértelmezett értéke 1, azaz a socket-en küldött IP csomag nem juthat ki az adott alhálózatból, mert a routerek nem továbbítják. Ha 0, az állomás interfészkártyáját sem hagyhatja el. Mivel minden router a továbbításkor csökkenti eggyel a TTL értéket, ezen csomagok esetén az első forgalomirányítónál eléri a nullát, így eldobja őket. A TTL érték egy socket opció, melyet küldés előtt tudunk beállítani a socket fájlleíróját megadva, bár átállítása nem kötelező. Ezt, és több más beállítást a következő függvény segítségével tehetjük meg.

```
setsockopt(socket_fd, IPPROTO_IP, IP_MULTICAST_TTL, &option,
    sizeof(option));
```

Első argumentum a socket, amelyikre a beállítás vonatkozik. A második megmondja, hogy milyen szintre vonatkozik a beállítás, jelen esetben (és a legtöbb multicast-ra vonatkozó beállításnál) a hálózati réteg szintjére. Az következő az opció neve, hogy mit szeretnénk beállítani, majd az opcióhoz tartozó változó címe és mérete, jelen esetben ezeket nem használjuk. Ha több interfésze van egy állomásnak, akkor lehetőség van beállítani, hogy a csomag melyik interfészen továbbítódjon, az opció neve IP_MULTICAST_IF, értékként struct in_addr változó címét kell átadni a setsockopt()-nak. Az IP_MULTICAST_LOOP opcióval visszacsatolás állítható be. Ha ki van kapcsolva, akkor a socket, amelyik tagja egy multicast csoportnak, nem kapja meg a saját csomagjait, amelyet a csoportcímre küld. Ha engedélyezzük (ez az alapértelmezett), a socket látja a saját maga által küldött csomagokat, ha feliratkozik arra a csoportra, amelyikre küld. Az opció értéke 0 vagy 1 egész lehet. A setsockopt() függvény párja a getsockopt(), amely pont az ellentétét végzi, lekérdezi az adott socket opcióra vonatkozó információkat.

Ha elvégeztük a szükséges beállításokat, elküldjük a csomagot.

```
sendto(socket_fd, &packet, 100, 0, (struct sockaddr *) &
    to_socket_address, sizeof(to_socket_address);
```

Második paraméterként az elküldendő üzenetet egy mutatón keresztül olvassa ki, figyelembe véve a hosszát. Ez azért szükséges, mert így a gyakorlatban alkalmazható az a módszer, hogy egy-egy változót (például egy struktúrát) szerializálunk, azaz a változóhoz tartozó memória tartalmat továbbítjuk a hálózaton, fogadáskor pedig a változó típusára kényszerítve visszakapjuk az eredeti struktúrát. A szerializálással vigyázni kell, mert különböző platformok esetlegesen másképp allokálhatják a memóriaterületeket ugyanazon típusú változókhoz a struktúrákban is, így mást kaphatunk fogadáskor, ha az alkalmazás másik platformra lett lefordítva.

Negyedik paraméter a küldés jelzőbitjeit tartalmazza, majd következő kettő a korábban beállított cím mutatója és mérete. Egy UDP szegmensben fragmentáció nélkül legfeljebb 1472 bájt adat küldhető, mert az IP és UDP fejrész 28 bájt, az IP MTU-ja (Max Transfer Unit) pedig 1500 bájt.

### 4.2.2. UDP üzenetek fogadása

Multicast üzenetek fogadása a következő lépésekből áll:

1. Socket létrehozása
2. Socket opciók megadása, ha szükséges
3. Bind-olás (kötés) a socket-hez
4. Csatlakozás a multicast csoporthoz
5. Multicast üzenetek fogadása
6. Kilépés a multicast csoportból
7. Socket lezárása

A socket létrehozása ugyanúgy történik, mint az előző esetben. Ha egyet már létrehoztunk küldés céljából, nem feltétlenül muszáj újat létrehozni, mert egy socket kétirányú kommunikációra képes, használható fogadásra is.

Következő lépésben beállíthatjuk a socket opciókat fogadásra vonatkozóan hasonlóképpen, mint küldésnél a setsockopt()-tal. Fontos, hogy ezeket még a kötés előtt be kell állítani, mert különben nem fognak működni.

Az egyik használható opció a cím újrafelhasználás. Ez azt jelenti, hogy alapértelmezetten egy socket-re csak egy alkalmazás (processz) tudna kötni, de ezt beállítva akár több is. Két fél esetén ennek nem lenne értelme, mert úgysincs több résztvevő a kommunikációban, viszont multicast-nál semmi akadálya nem lenne annak, hogy egy állomáson akár több alkalmazás fogadja ugyanazt a csomagot ugyanazon a porton. Kétféle opció megadás lehetséges, bármelyik használható a SO_REUSEPORT vagy a SO_REUSEADDR közül (nem biztos, hogy mindkettő rendelkezésre áll), és socket szintre állítjuk be (SOL_SOCKET), az értéke 0 vagy 1 egész lehet.

Még fogadás előtt értesítenünk kell az operációs rendszert, hogy a használt socket-hez tartozó címre és portra érkező datagramokat az alkalmazásunkhoz továbbítsa, ez a kötés (bind).

```
bind(socket_fd,(struct sockaddr *) &from_socket_address, sizeof
    (from_socket_address));
```

A socket cím egy ugyanolyan struktúra, mint amelyet a küldésnél a cél IP cím és port megadásakor használtunk, de most azt adjuk meg, hogy mely portra és mely célcímre érkező datagramokat szeretnénk megkapni. Megadjuk, hogy bármilyen IPv4-es címre (0.0.0.0) és a megadott portra bejövő csomagok legyenek továbbítva az alkalmazásnak. A bármilyen IP címet az INADDR_ANY konstans jelöli.

Fel kell iratkoznunk arra a címre, amely csoport üzeneteit meg szeretnénk kapni, ehhez ugyancsak a setsockopt() használata szükséges. Az opció az IP_ADD_MEMBERSHIP, és mivel ez multicast-specifikus (hálózati rétegbeni) beállítás, ezért a szintje IPPROTO_IP. Értékként meg kell adni egy speciális struktúra címét, amely majd azokat az információkat tartalmazza, hogy hova csatlakozunk. Megadjuk a csoport multicast címét és a helyi interfészt, ahol majd fogadni fogjuk a csoportcímre érkező csomagokat. Az interfész megadásánál itt is használhatunk „any” címet, vagy több interfész esetén pontosan megadhatjuk a fogadó interfész IP címét.

```
struct ip_mreq mc_request;
mc_request.imr_multiaddr.s_addr = inet_addr("239.255.100.1");
mc_request.imr_interface.s_addr = htonl(INADDR_ANY);
```

```
setsockopt(socket_fd, IPPROTO_IP, IP_ADD_MEMBERSHIP, &
    mc_request, sizeof(mc_request));
```

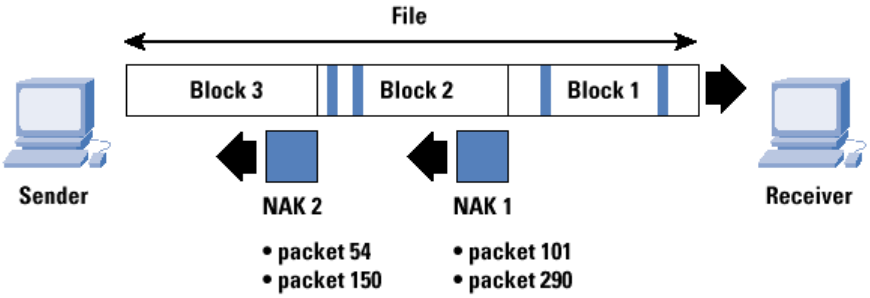
Ennél socket beállításnál két dolog történik. Egyrészt az operációs rendszer innentől kezdve adja át egyáltalán szállítási réteg felé azokat az UDP szegmenseket, melyek a megadott csoportcímre érkeztek. Másrészt a hoszt egy IGMP üzenetet küld a routerekhez a helyi alhálózatban, hogy az adott állomás meg kívánja kapni a join üzenetben szereplő csoport üzeneteit. A legtöbb jelenlegi operációs rendszer támogatja az IGMP harmadik verzióját, így ilyen üzenet továbbítódik. Egy időben ugyanazon a socket-en akár több csoportra vonatkozóan is lehet csatlakozni, így több csoportból is kaphat üzeneteket, ennek számát az operációs rendszer korlátozza.

Mielőtt a socket-et lezárnánk, küldenünk kell a helyi router felé egy IGMP távozási üzenetet, illetve jelezni kell a rendszer felé, hogy már nem kívánjuk megkapni a csoport címére küldött csomagokat. Ezt a feliratkozáshoz használt setsockopt() függvénnyel tehetjük meg, a leiratkozási opció neve IP_DROP_MEMBERSHIP, és a korábban használt multicast request változót kell megadnunk hozzá. Ez a művelet nem feltétlenül szükséges, mert a program befejeződésekor általában automatikusan megtörténik, akárcsak a socket lezárása.

# 5. Multicast megbízható állománykuldés

## 5.1. Állománykuldésre használható megbízható protokollok

A programban az adat küldésének folyamata a Multicast File Transfer Protocol ötletét használja fel. Az adatszegmenseket blokkokba szervezi, a csomagokat tartalmazó blokkokat pedig egymás után sorban elküldi, a hibáktól függően egy vagy több menetben. A kliensek minden fogadott blokk után megállapítják, hogy melyek azok a csomagok a blokkban, melyeket nem kaptak meg. Ahelyett, hogy valamennyi sikeresen fogadott csomag után unicast címzésű pozitív nyugtákat küldenének a kliensek (mint például a TCP-nél), negatív nyugtákkal tájékoztatják a szervert, ha hiányos blokkot fogadtak. Ha a blokk teljes, akkor nem küldenek semmit sem a szervernek, hogy feleslegesen ne terheljék. Ha nem teljes a blokk, akkor egy blokktérképet ké-




7. ábra. Az MFTP jelzési mechanizmusa.

szítenek, mely a blokk csomagjait egy-egy bittel reprezentálja, és 1-gyel jelzi, mely csomagok hiányoznak. Ezeket elküldik a szervernek, majd a szerver a kapott negatív nyugtákat összegzi. Így a szerver biztos lehet benne, hogy mindazon csomagokat újraküldi a következő menetben, melyre valamelyik kliensnek szüksége van. Természetesen nincs arra garancia, hogy a blokktérképek megérkeznek a szerverhez, viszont a küldés addig és annyi menetben történik, amíg nem jelzi az összes kliens, hogy a fájl teljes. Mivel nem valós időben kell újraküldeni, ezért elegendő, ha az újraküldéssel megvárja a menet végét. A nem sorrendi érkezéssel nem kell törődnie a fogadó feleknek, mert ha egy csomag sorrenden kívül érkezik, a fájlban egyszerűen a megfelelő helyre ugrik a sorszám alapján, és oda írja az adatrészt. A módszer szelektív elutasítást használ, azaz ha valamelyik kliens olyan csomagot kap, amelyet már korábban megkapott, azt figyelmen kívül hagyja. [14]

Jó megoldásnak mondható a negatív nyugták alkalmazása, ha nem egy majdnem valósidejű, de megbízható átvitelről van szó. Valós idejű küldésnél az egyik megközelítés, amelyet a megbízható multicast átviteli protokoll (RMTP) is alkalmaz, hogy a küldő felen kívül több fogadó állomást is kijelölhetünk újraadásra a továbbítási fában, és egy meghatározott tartományban ő gyűjti össze a nyugtákat és küldi újra a szükséges adatrészeket. A másik lehetséges módszer, hogy bármely fogadó küldhet újra egy tartományban, célszerűen ez a legközelebbi ahhoz, amelyiknek szüksége van az elvesztett csomagokra. Így működik a skálázható megbízható multicast (SRM).

## 5.2. Az állomány szegmentálása

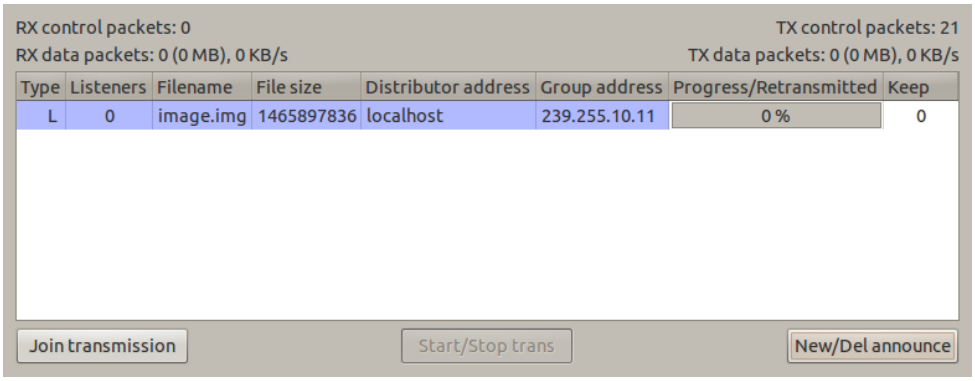
Mivel az UDP nem tudja sorrendbe állítani az egymás után érkező szegmenseket (a fejrész nem tartalmaz sorszámot, mint a TCP esetén), ezért az adatot tartalmazó szegmens 4 bájtos sorszámból és 1468 bájt adatrészből áll. Az elküldött IP csomag mérete így Ethernetre optimalizált, 1472 bájt egy szegmensben a maximális méret, amely még fragmentáció nélkül elküldhető. A küldendő fájlt összesen annyi részre osztjuk, amennyi csomagban az elküldhető (teljes méret bájtokban osztva 1468-cal, felfelé kerekítve), ezt az összes résztvevő kiszámolja a fájl méretéből, így tudni fogja, összesen hány csomag várható. A küldendő csomagokat pedig további egységekben, blokkokban csoportosítva küldjük, egy blokk jelen esetben 4096 csomagot tartalmaz.

## 5.3. Az átvitel működése

A program indításakor mind a szerver, mind a kliensek egy közös multicast címre (vezérlő üzenetek címére) csatlakoznak és figyelik erre a címre és bizonyos UDP portra (vezérlő üzenetek portjára) érkező csomagokat. Ez a multicast cím a helyi tartományból került kiválasztásra (239.255.10.10). Az alkalmazás szerver és kliens módban is képes működni, és akár egyszerre több átvitel is folyhat egyszerre a hálózaton. A küldő fél megnyitja az adott fájlt, és arra vonatkozóan bizonyos időközönként (2 másodperc) a program hirdetéseket küld a vezérlő üzenetek címre, ami az összes klienshez eljut.

Egy vezérlő üzenet tartalma:

- Üzenet típusa (1 bájt): hirdetés esetén ANNOUNCEMENT
- Fájlnév (80 bájt): a küldendő fájl neve
- Fájl mérete (15 bájt): a küldendő fájl mérete bájtokban, szövegesen tárolva a kompatibilitási problémák miatt. A fájl méretéből a kliensek később tudni fogják, hogy mekkora az adat, melyre várniuk kell.
- Csoportcím (4 bájt): az a választott csoportcím, melyen az adatküldés fog zajlani. Különbözőnek kell lennie a vezérlő címtől.




8. ábra. Az alkalmazás szerver oldalon működtetve, új hirdetés létrehozása.

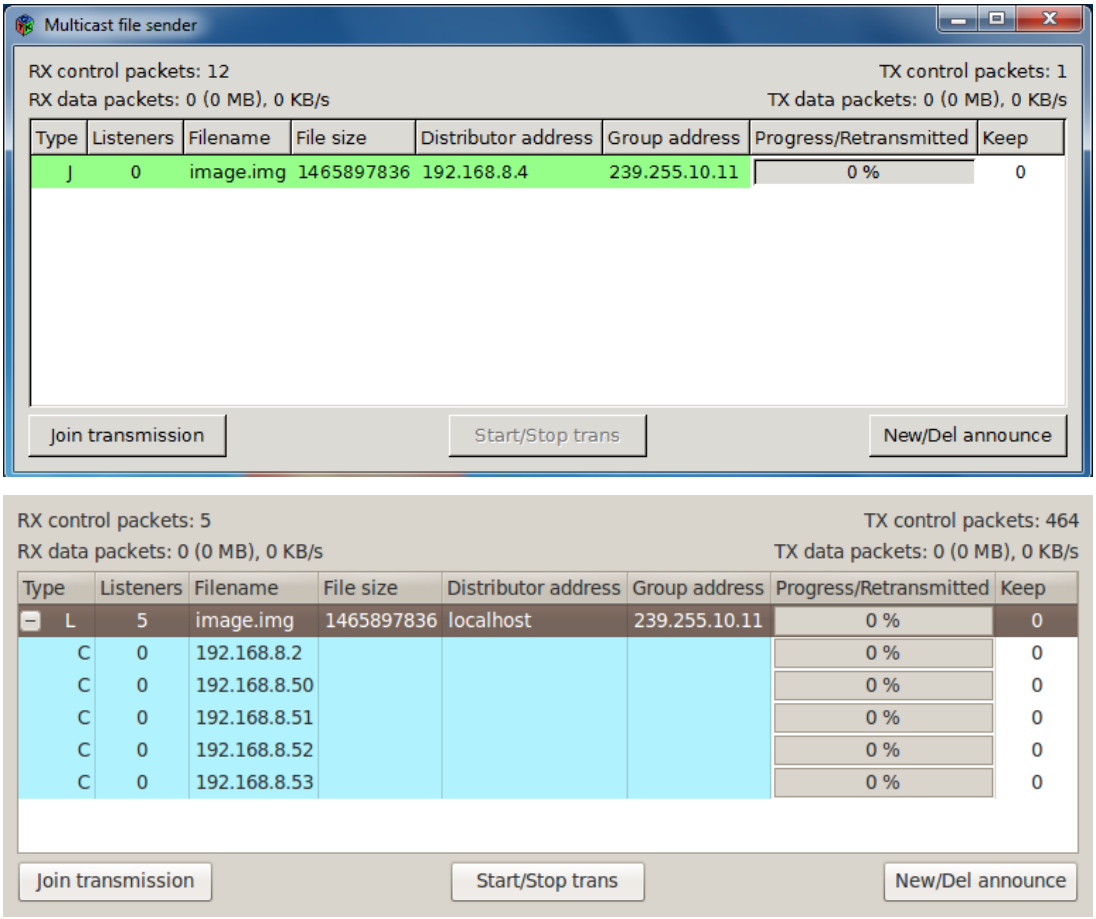
Az alkalmazás alapja egy GtkTreeStore objektum, melyet egy komponens táblázat formájában tud dinamikusan megjeleníteni, azaz a tároló objektum elemeinek változtatásával a táblázat komponens is ennek megfelelően változik. A táblázat tartalmazza, hogy milyen nevű és méretű az állomány, melyik IP című gép terjeszti, hány feliratkozott állomás van és melyik multicast címen fogja forgalmazni a fájl adatcsomagjait. Egyéb oszlopokat is tartalmaz a táblázat, melynek egy része rejtett a komponensben.

Ha a kliens egy hirdetést fogad, eltárolja a hozzá tartozó adatokat a listában, amennyiben még nem tartalmazta korábban a hirdetést. Ellenkező esetben egy számlálót nulláz a bejegyzésre vonatkozóan. Ez azért szükséges, mert a lista egy öregedési számlálót alkalmaz, hogy egy-egy bekerült hirdetés ne maradjon a listában, ha a küldés véget ért, vagy a fogadó felek nem értesültek arról, hogy a küldő visszavonta a küldési szándékát. A programban egy függvény („szemétgyűjtő”) bizonyos időközönként lefut, és azokat a bejegyzéseket törli, amelyek felől régóta nem kapott hirdetést.

A küldés három fázisra bontható, a legelsőben a szerver értesül arról, hogy mely kliensek fogják a fájlt fogadni.

### 5.3.1. Regisztrációs fázis

A kliensek a fogadott hirdetésekre regisztrálnak, és ezzel jelzik a küldőnek, hogy meg kívánják kapni a küldött fájlt. A regisztráláskor a fogadó fél egy JOIN típusú vezérlő üzenetet




9. ábra. Regisztrált kliensek a kliens- és szerver oldalon.

küld a szerver unicast címére (a legtöbb vezérlő üzenet csak a fájlnév mezőt használja), melyre ő megerősítő ACK típusú üzenetet küld vissza. Nyugtára azért van szükség, mert az UDP üzenetek esetén a fogadó nem lehet biztos abban, hogy a csatlakozási szándékot jelző vezérlő üzenetet a küldő fél fogadta. Addig próbálkozik a JOIN-ok küldésével, amíg vagy megerősítést nem kap, vagy a próbálkozás el nem ért egy meghatározott számot. Ha sikerült csatlakozni, a kliens előkészíti a fájl fogadásához a kapcsolatot. Egy új szálban csatlakozik a multicast csoporthoz, bind-ol a hirdetésben szereplő csoport címre és előre definiált portra, melyet a program az adatrészek küldésére fog használni (adatfolyam port), ezt követően arra várakozik, hogy az első adatcsomagok megérkezzenek. A szerver számon tartja, hogy mely kliensek csatlakoztak hozzá. JOIN érkezésekor felveszi a listájába a kliens IP címét, így tudni fogja, hogy mely

állomásokra kell várnia küldéskor, hogy a fájl összes részét megkapják. JOIN_CANCEL és ANNOUNCEMENT_DELETE típussal küldött vezérlő üzenet csatlakozás és küldés visszavonására szolgál.

### 5.3.2. Küldési fázis

A forrás alapú multicast modellje jól használható az adatfolyam továbbítására, mert egy adó ad, a többi fél pedig fogad. Ez megfelel egy SSM csatorna leírásának.

Az SSM csatornára való feliratkozás valamelyest különbözik az ASM-nél megismerttől, mert itt meg kell adni a forrás címét is. Erre egy speciális beépített struktúra, az ip_mreq_source használható. Amikor csatlakozunk a setsockopt() függvénnyel a csoporthoz, jelezni kell az opcióban, hogy ez egy forrás alapú multicast csatorna.

```
struct ip_mreq_source req;
req.imr_multiaddr.s_addr = inet_addr(group_addr);
req.imr_sourceaddr.s_addr = inet_addr(source_addr);
req.imr_interface.s_addr = htonl(INADDR_ANY);
setsockopt(socket_fd, IPPROTO_IP, IP_ADD_SOURCE_MEMBERSHIP, (
    void *)&req, sizeof(req)));
```

Addig nincs címütközés, amíg egy szerver csak egy fájlt hirdet. A kliensek hirdetésnél megtudják a csatorna adatait, a csoportcímet a hirdetés tartalmazza, a forrás pedig a feladó unicast címe lesz.

Miután az összes kliens csatlakozott, a szerveren el kell indítani a küldést, ezzel lezárul a csatlakozási fázis. Innentől a szerver ugyanúgy 2 másodpercenként küld hirdetéseket, de AT_TRANSMISSION típussal jelzi a nem csatlakozottaknak, hogy a megadott multicast csatornán küldés folyik. A második fázisban történik az adatszegmensek elküldése.

Mindenki, aki résztvevője az átvitelnek, a programban nyilvántart egy ún. fájltérképet. Ez egy lefoglalt memória terület, mely annyi bitből áll, amennyi csomagban az állomány elküldésre kerül. Ha az adat maximális küldési egységet vesszük Ethernet esetén (1468 bájt), ez a fájltérkép

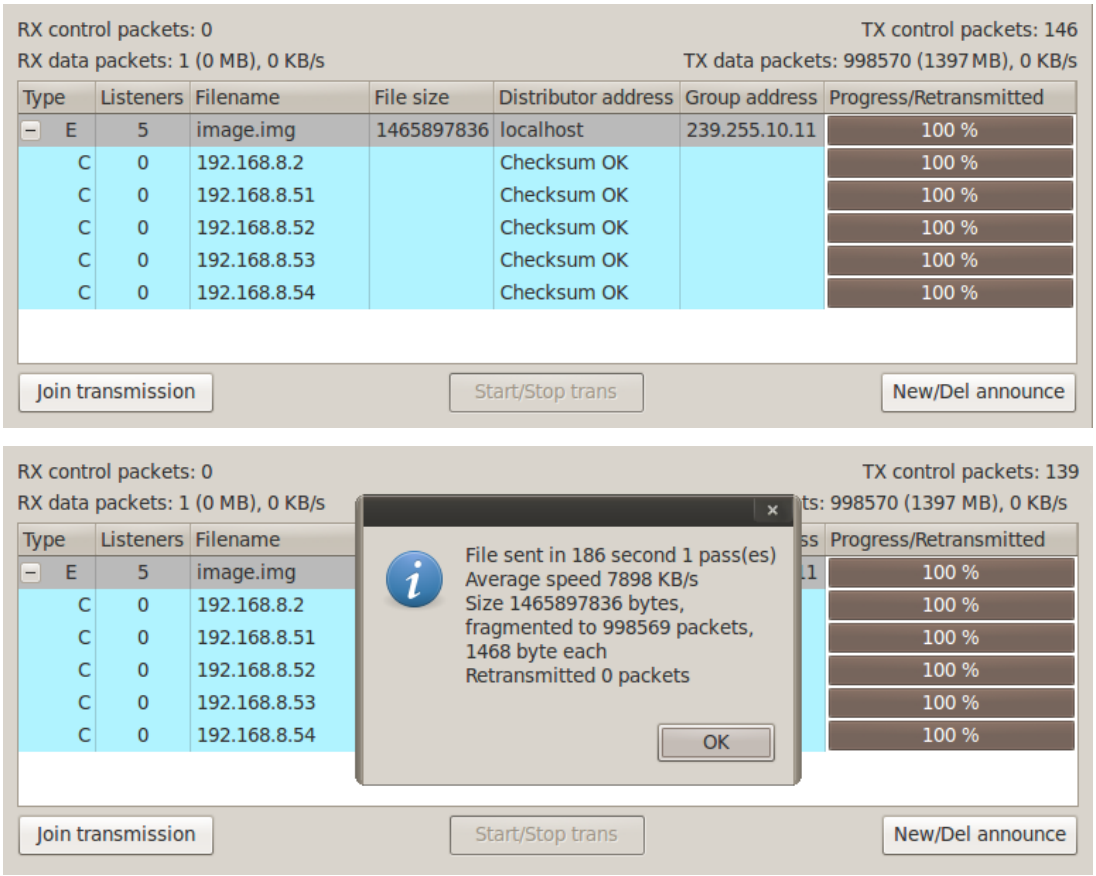
még egy 100 gigabájtos fájl esetén is kevesebb, mint 9 megabájt memóriaterületet foglal el, így nem jelentős a memóriaigénye. Ezt használva a küldés sikerességét az egész átvitel során az összes résztvevő figyelemmel kíséri.

A küldő folyamatosan adni fogja a hiányzó csomagokat egy-egy menetben, nem áll meg időtúllépéssel összegyűjteni az adott blokkra vonatkozóan a blokktérképeket. A fogadók viszont érzékelik, hogy a küldés mikor érkezik blokkhatárhoz, és ekkor a sorszám alapján hiányzó adatcsomag esetén a megelőző blokk blokktérképét (a fájltérkép meghatározott szeletét) elküldi a szervernek unicast üzenetként. Az egyszerűség kedvéért egy blokktérkép üzenet felépítése megegyezik az adatszegmens felépítésével, itt a 4 bájtos sorszám jelzi, hogy a blokktérkép hanyadik sorszámú csomaggal kezdődik. Az adatrészben 512 bájt lesz hasznos, amely a blokktérkép bitjeit tartalmazza (4096-os blokkméret esetén). Az adó és a vevők is csupa 0-val feltöltött fájltérképpel indulnak, és a szerverhez beérkező blokktérképen és a meglévő fájltérkép szeleten bináris ÉS műveletet hajt végre.

### 5.3.3. Ellenőrző fázis

Az adó ha egy csomagot leadott, a vevő pedig ha egy csomagot vett, a sorszámhoz tartozó bitet a fájltérképen 1-re állítja. A kliens figyeli, hogy mikor válik teljessé a fogadott a fájl. Ha ez megtörténik, a harmadik fázisban a vezérlő csatornán PEER_ALL_RECEIVED típusú üzenetet küld, a szerver pedig tárolja, hogy a kliens a fájl összes részét megkapta. A küldő bizonyos időnként ellenőrzi a meglévő bejegyzéseket, és ha mindegyik kliens megkapta az összes részt, akkor leállítja a küldő ciklust, következhet a fájl integritásának ellenőrzése. A küldő az első menetben minden csomagot elküld, így lehetősége van arra, hogy egymás után a fájl részeiből egy MD5 ellenőrző összeget számoljon. Ha egy menet véget ért, azaz elküldte az összes, fájltérképen 0-val jelzett csomagot, elküldi az első menetben kiszámolt ellenőrző összeget. A kliensek egyrészt innen tudják, hogy új menet kezdődött, másrészt ha megkapják, össze tudják később hasonlítani a saját maguk által számolt MD5 hash-sel. A kliensek ezt nem tudják fogadási időben számolni, mert nem garantált, hogy a csomagok sorrendben érkeznek, ezért utólag kell a fájlt újra megnyitni, beolvasni és kiszámolni az ellenőrző összegét. Ha egyezik a fogadott és

a kiszámolt hash érték, a szerver felé PEER_FCS_OK típusú vezérlő üzenetet küld, ellenkező esetben PEER_FCS_MISMATCH típusút. Mindkét fél jelzi az egyezést vagy az eltérést, így ha egyezést mutat a két hash érték, akkor biztosnak tekinthető, hogy az elküldött és a fogadott fájl megegyezik.




10. ábra. Kliensek visszajelzései a szervernek a küldés végén.

Mivel hibás UDP szegmens elvileg nem kerülhet feldolgozásra (nem kapja meg az alkalmazási réteg), ezért az ellenőrző összeg akár opcionális is lehet, küldése eredetileg tesztelési célokat szolgált. A harmadik fázissal a küldés befejeződött, a szerver utoljára AT_TRANSMISSION típusú hirdetés helyett TRANSMISSION_END típusút küld, ez jelzi az összes félnek, hogy a küldés véget ért az adott csatornán.

## 5.4. Adatfolyam-vezérlés

Az átvitel ezzel az újraküldési mechanizmussal már biztonságosnak mondható, normális esetben a kliens minden csomagot megkap véges számú menetben. Mivel azonban nagy teljesítményű továbbításról van szó, azaz akár az átviteli csatorna teljes sávszélességét kihasználhatjuk, ezért az állomásokat fel kell készíteni ilyen intenzitású adatforgalomra.

Megfelelő értékre kell állítani a bejövő UDP puffert az adatot fogadó socket-re, mert előfordulhat, hogy az alapbeállítás értéke túl alacsony, és az UDP csomagok túl gyorsan érkeznek ahhoz, hogy azokat fel lehessen dolgozni. Például Windows rendszerekben ez a puffer 32 kilobájt, míg Linux-ban 122 kilobájt nagyságú. Windows-ban alapértelmezett méretű fogadó puffer esetén mintegy fél százaléka az állomásnak küldött csomagoknak elvész 100 megabit per szekundumos hálózaton. Ha azonban megnöveljük 120 kilobájt körüli értékre, akkor a fogadásnál gyakorlatilag már nem történik csomagvesztés. A puffer nagysága tovább növelhető szükség esetén, Windows-ban 1 megabájt méretig. A programozás során egy socket opcióként lehet megadni a puffer méretet. A példában egy köztes értékre lesz állítva a socket-hez tartozó puffer.

```
int rcvbuflen = sizeof(rcvbuf);
int rcvbuf = 512000;
setsockopt(sock_fd, SOL_SOCKET, SO_RCVBUF, &rcvbuf, rcvbuflen);
```

Kizárólag az adó felelőssége, hogy olyan sebességgel adjon, hogy a kliensek fogadni tudják, és a lehető legkevesebb csomag vesszen el az átvitel során. Ha az alkalmazási rétegben a program nem tudja elég gyorsan feldolgozni az érkező csomagokat nagy vételi puffer ellenére sem, akkor külön szálon kell futtatni a fogadó socket függvényt, és alkalmazási rétegben kell egy puffert alkalmazni.

A program szerver oldali részén a sebességvezérlés úgy történik, hogy bizonyos időegységenként (például 10 ms) valamennyi csomagot lead. Az első menetben korlátozás nélkül, maximális sebességgel ("drótsebességgel") forgalmaz, így ugyan a lassabb vevőknél torlódás alakul ki, de a gyors vevők már első menetben megkaphatják a teljes fájlt. A többi menetben az adási sebesség egyre kisebb lesz, míg végül a lassabb vevők is megkapják a hiányzó csomagjaikat.

Egy másik használható megoldás az lenne, hogy figyelembe vesszük a lassú vevőket is. Beérkező negatív nyugták esetén az adási sebességet a szerver fokozatosan csökkenti, így azonban a gyorsabb vevők hiába tudnának nagyobb intenzitással fogadni, a szerver a leglassabb vevőhöz alkalmazkodik. Előnyként jelentkezik, hogy viszonylag kicsi lesz a veszteség, és kevés menetben lehet továbbítani, viszont lassabban.

## 5.5. Visszacsatolási problémák multicast esetén

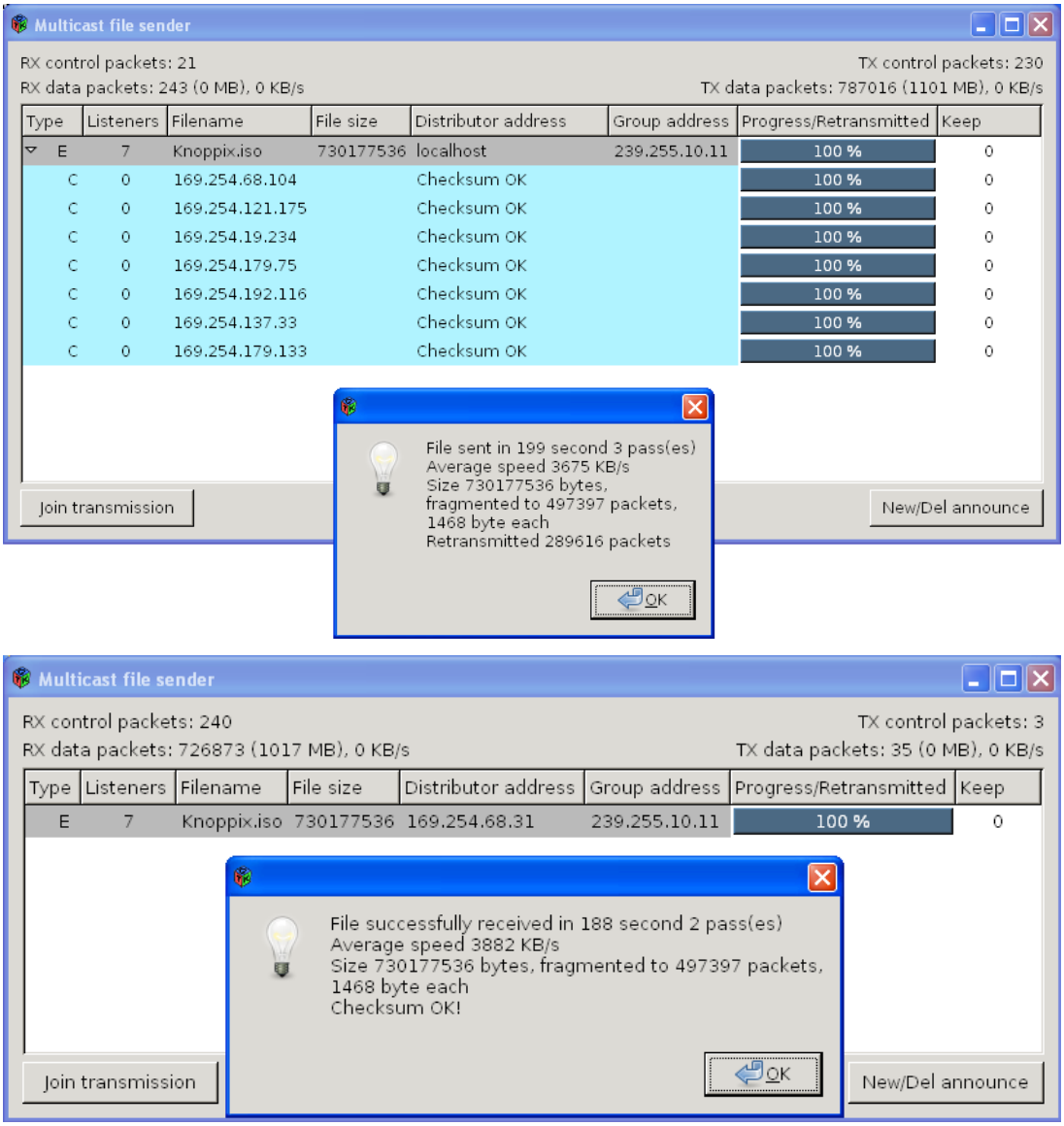
A programozás során nem kell foglalkozni azzal, hogy hogyan jut el a csomag egyik alhálózatból a másikba, akárcsak unicast küldés esetén, ezt a hálózati eszközökre bízzuk. Természetesen ehhez multicast támogatás szükséges az eszközök szintjén. Azonban azt figyelembe kell venni, hogy mennyi csomag érkezik egyszerre egy adott állomáshoz.

Ha megbízhatóságot akarunk garantálni, időnként szükséges, hogy a fogadó felek unicast nyugtákat küldjenek. Nyugtát nyilván valamilyen esemény hatására lehet elküldeni (a programban egy blokknyi fogadott adatcsomag után, ha szükséges), így előfordulhat, hogy sok állomás ugyanabban az időpillanatban küldi el nyugtáját a szerver felé. Amikor ezek megérkeznek hozzá, torlódást okozhatnak, és időbe telik, mire az feldolgozza. Így akár el is veszhetnek nyugták, ami ronthatja a hatékonyságot.

Meg kell akadályozni, hogy ugyanabban az időpillanatban többen küldjenek nyugtát a szervernek. Erre a legegyszerűbb módszer, ha meghatározunk egy intervallumot, amin belül véletlen ideig várakoztatjuk a nyugtát, mielőtt elküldenénk a szervernek.

## 5.6. Tesztelés

Az alkalmazás tesztelése egyetemi tantermi számítógépeken történt. Erősen befolyásolta a teszt eredményeit, hogy a legtöbb gép viszonylag lassú volt a sok telepített alkalmazás miatt. Az átvitel jellegéből adódan néhány lassabb vevő megnövelheti az egész átvitel idejét. A csomagok több, mint felét ebben az esetben újra kellett adni, de még így is tapasztalható sebességbeli javulás unicast átvitelhez képest.




11. ábra. A szerver és egy kliens a tesztkörnyezetben.

Mindezeket figyelembe véve az ábrán látható, hogy a szerver által küldött adatfolyam átlagos sebessége kb. 3675 kilobájt volt másodpercenként, és három menet kellett a küldés befejezéséhez (a fenti kliens két menetben megkapta az adatrészeket). Ez azt jelenti, hogy mivel 7 kliens vett részt az átvitelben, az átvitel aggregált átlagos sebessége a kliensek felé kb. 25725 KB/s. Ha unicast küldéssel küldenénk (például FTP-vel), a 7 kliens összesen maximum 10000-

12000 KB/s átviteli rátán kellene, hogy osztozzon egy 100 megabit per szekundumos hálózaton. Az előzőekből következik, hogy már 7 kliens esetén is az össz-átviteli sebesség több, mint két-szerese a multicast használatakor unicasthoz képest, így felére rövidül az átviteli idő. A kliensek számának növelésével ez a hatékonysági mutató tovább növekedhet.

# 6. Összefoglalás

A multicast technológia használata a már meglévő területek mellett egyre inkább általánossá válik. Erre szükség is van, hiszen a multimédiás és egyéb szolgáltatások száma folyamatosan növekszik, és általuk nagy teher hárul a feleket összekötő hálózatra.

Általánosságban ismertettem, mi is az a többes küldés, hol érdemes használni, és mi a működésének az alapja, az egyes csoportok és állomások hogyan feleltethetők meg egymásnak, és milyen címzési rendszer használható. Bemutattam a működéshez szükséges csoportmenedzsment- és irányító protokoll típusokat, és, hogy azok milyen irányelvek mentén működnek. A végponttól végpontig tartó kommunikációt figyelembe véve elsősorban gyakorlati szempontból ismertettem a multicast üzenetküldéskor fellépő problémákat és különbségeket az unicast küldéssel szemben, illetve ezek kezelésére módszereket adtam meg. Különböző jellegű forgalmak esetén különböző lehetőségeket mutattam be a multicast csomagok továbbítására.

Felvázoltam, hogy milyen kritériumokat kell teljesítenie egy multicast küldéssel működő programnak ahhoz, hogy az átvitel megbízhatónak legyen tekinthető. A kritériumok alapján már meglévő mechanizmusok ötleteit felhasználva megvalósítottam egy multicast fájlátviteli alkalmazást, amely demonstrációs célokat szolgál, és amely alkalmazással összemérhető más hasonló, de unicast átvitelt használó alkalmazás hatékonysága.

Az elméleti működés bemutatása során elsősorban angol nyelvű RFC dokumentumokat dolgoztam fel remélve, hogy később hasznos lehet azon érdeklődők számára is, akik ebben a témában magyar nyelvű irodalmat keresnek. A megírt programmal igyekeztem ezt a témát a gyakorlati oldaláról is bemutatni, bizonyítani a multicast működőképességét és létjogosultságát. A programozás során magam is sok hasznos tapasztalattal lettem gazdagabb.

# Ábrák jegyzéke

# Hivatkozások


[1] **David Makofske, Kevin Almeroth** (2003) *Multicast Sockets* (ISBN: 1-55860-846-X)

[2] **Andrew S. Tanenbaum** (2003) *Computer Networks*, 4th edition (ISBN: 0-13-066102-3)

[3] **IP-Multicasting Technology**, Intelligraphics Inc.
http://www.intelligraphics.com/articles/ipmulticasting1_article.html
(Letöltve: 2010. 09. 15.)

[4] **Internet Multicast Today**
http://www.cisco.com/web/about/ac123/ac147/ac174/ac198/about_cisco_ipj_archive_article09186a00800c851e.html (Letöltve: 2010. 09. 15.)

[5] **Multicast in a Campus Network: CGMP and IGMP Snooping**
http://www.cisco.com/en/US/products/hw/switches/ps708/products_tech_note09186a00800b0871.shtml (Letöltve: 2010. 10. 20.)

[6] **RFC 3180, GLOP Addressing in 233/8 (September 2001)**
http://www.rfc-editor.org/rfc/rfc3180.txt (Letöltve: 2010. 10. 01.)

[7] **RFC 3171, IANA Guidelines for IPv4 Multicast Address Assignments (August 2001)**
http://www.rfc-editor.org/rfc/rfc3171.txt (Letöltve: 2010. 10. 03.)

[8] **RFC 2375, IPv6 Multicast Address Assignments (July 1998)**
http://www.rfc-editor.org/rfc/rfc2375.txt (Letöltve: 2010. 10. 01.)

[9] **RFC 3513, nternet Protocol Version 6 (IPv6) Addressing Architecture (April 2003)**
http://www.rfc-editor.org/rfc/rfc3513.txt (Letöltve: 2010. 10. 01.)

[10] **RFC 1112, Host Extensions for IP Multicasting (August 1989)**
http://www.rfc-editor.org/rfc/rfc1112.txt (Letöltve: 2010. 10. 02.)

[11] **RFC 2464, Transmission of IPv6 Packets over Ethernet Networks (December 1998)** http://www.rfc-editor.org/rfc/rfc2464.txt (Letöltve: 2010. 10. 01.)

[12] **RFC 2236, Internet Group Management Protocol, Version 2 (November 1997)** http://www.rfc-editor.org/rfc/rfc2236.txt (Letöltve: 2010. 10. 01.)

[13] **RFC 3376, Internet Group Management Protocol, Version 3 (October 2002)** http://www.rfc-editor.org/rfc/rfc3376.txt (Letöltve: 2010. 10. 01.)

[14] **Reliable Multicast Protocols and Applications** http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-2/reliable_multicast.html (Letöltve: 2010. 07. 05.)

[15] **NIIF, multicast szolgáltatás** http://www.niif.hu/adathalozat/hbone/multicast (Letöltve: 2010. 06. 27.)

[16] **PIM Overview** http://www.metaswitch.com/multicast/what-is-pim.aspx (Letöltve: 2010. 10. 07.)

[17] **GTK+ Reference Manual** http://library.gnome.org/devel/gtk/stable/ (Letöltve: 2010. 08. 03.)

[18] **Winsock reference** http://msdn.microsoft.com/en-us/library/ms741416%28v=VS.85%29.aspx (Letöltve: 2010. 08. 07.)

# 7. Köszönetnyilvánítás

Köszönöm témavezetőmnek, Dr. Almási Béla egyetemi docensnek, hogy szakmai tanácsaival, illetve az alkalmazás fejlesztése és tesztelése során szükséges eszközök biztosításával hozzájárult e dolgozat létrejöttéhez.