PÁZMÁNY
Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Számítógépes hálózatok

#07 – Transport layer protocols 2

2024. október 25.

Naszlady Márton Bese

naszlady@itk.ppke.hu

#07/1 – ACK generálás

Stratégiák ACK generálásra

Azonnali ACK

Az ACK a szegmens megérkezése után egyből generálódik. Azonnal visszajelez, de ezzel „fölöslegesen” terheli a hálózatot; várhatott volna, amíg jön egy másik adat, és a kettőt akár egyszerre is nyugtázhatta volna.

Előnyök:

- gyorsaság
- jól használható olyan esetekben, ahol interaktivitás történik (pl. ssh)

Hátrányok:

- nagy overhead (1 bitnyi ACK miatt egy teljes szegmenst küld)

Stratégiák ACK generálásra

Késleltetett ACK (delayed ACK)

A szegmens beérkezésekor elindítunk egy időzítőt. Ha az időzítő lejárta előtt további adatot fogadunk, akkor ezeket mind egyszerre, egy üzenetben fogjuk nyugtázni. Ha lejárt az időzítő, akkor nyugtázzuk azt az egy szegmenst, amit kaptunk.

Előnyök:

- ésszerű mennyiségű ACK-t küld

Hátrányok:

- lehet, hogy túl sokat vár; a küldő hibát feltételez
- nem interaktív; vár, amíg elég adatot gyűjt ahhoz, hogy érdemes legyen adni

ACK generálási események

Esemény	Reakció
A nyugtázott és vett sorszámhoz képest egy rákövetkező szegmens érkezik.	Indítsunk egy késleltetést, ne küldjünk még ACK-t egy kis ideig, hátha jön még valami...
A vett sorszámhoz képest újabb rákövetkező szegmens érkezik, és már ketyeg az előbb elindított késleltetés.	Azonnal küldjünk ACK-t, állítsuk le a késleltetést és léptessük az ablakot.
A vett sorszámmal összevetve egy olyan csomag érkezett, ami kimaradást jelez.	Azonnal küldj ACK-t, ami a legutolsó jó csomagot nyugtázza ismét.
Megérkezett egy olyan csomag, ami pont hiányzott, és ezzel újra folytonosság lett.	Azonnal küldjünk ACK-t, ami a teljes folytonos darabot nyugtázza. Léptessük az ablakot.

Retransmission (újraküldés)

Én vagyok a küldő. Mikor kell újraküldenem valamit?

Két eshetőség van:

1. Tripla ACK-t kapok egy szegmensre, amit korábban küldtem. Azt gondolom, hogy elveszett valami, és attól a szegmenstől kezdve újraküldök mindent.
2. Már jó ideje elküldtem az adatot, de még nem kaptam ACK-t rá. A „jó időt” az RTO (retransmission timeout) érték adja meg; ha ezt túlléptem, akkor ismét elküldöm az adatot, hátha most sikerül.

Retransmission timeout (RTO)

Hogyan válasszam meg az RTO értékét?

Ha túl alacsony, akkor sok felesleges újraküldés történhet.

Ha túl magas, akkor egy csomag elvesztése nagy csuklást okoz; mehetek vissza az ablak elejére, és kezdhetem onnan az egészet.

A TCP szegmensek újraküldésénél is az RTT értékből lesz érdemes számolni.

Feltesszük, hogy már volt néhány sikeres küldésünk, és ebből kaptunk mindenféle RTT értékeket. Ezek elég eltérőek is lehetnek, érdemes az átlagukat számolni.

Retransmission timeout (RTO)

Először is, a most mért RTT értéket vegyük bele súlyozva a becsült RTT értékbe:

$$RTT_{\text{becs, most}} = a \cdot RTT_{\text{becs, eddig}} + (1 - a) \cdot RTT_{\text{mért}} \quad \text{ahol} \quad 0 < a < 1$$

Majd pedig ebből a becsült (ill. sok mérésen alapuló) RTT-ből számoljunk RTO-t:

$$RTO = b \cdot RTT_{\text{becs, most}} \quad \text{ahol} \quad b > 1$$

Vagyis megmérjük, hogy kb. mekkora az RTT, és ennél egy kicsit több időt adunk az ACK-nak, hogy megérkezzen.

Jellemző értékek: $a = 0{,}9$ és $b = 2$

Retransmission timeout (RTO)

Ravaszságok az RTO számításában

- Az α értéknek nem kell konstansnak lenni; pl. ha függ az $RTT_{\text{mért}}$ változásától, akkor ha hirtelen megnő az RTT, akkor azt jobban figyelembe tudjuk venni.
- Ha ismétlünk egy csomagot, akkor az arra jövő ACK-ból ne számoljunk RTT-t, mivel nem tudhatjuk, hogy az ACK az ismételt csomagra jött vagy az eredetire.
- Ha ismétlünk egy csomagot, akkor az RTO-t duplázzuk meg egy max. értékig (exponential backoff). (A max érték jellemzően néhány perc.)
  Ezzel esélyt adunk arra, hogy ha felére romlott a kapcsolat minősége, akkor is megkapjuk az ACK-t.

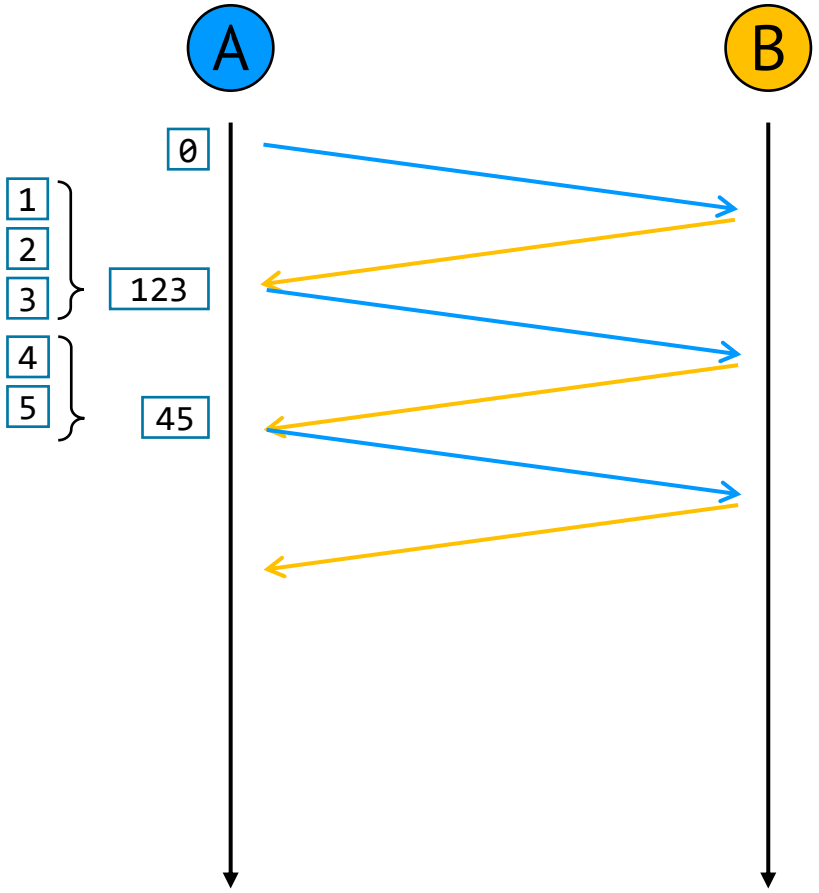
Nagle algoritmus

Ha sok apró adatot akarunk küldeni, azzal eltömíthetjük a hálózatot. A Nagle algoritmus feladata csökkenteni a szegmensek számát úgy, hogy késleltetést visz a rendszerbe. Az algoritmus az alatt az idő alatt, amíg az elküldött csomagra nem jön meg az ACK, egy bufferba gyűjti az elküldendő adatot.

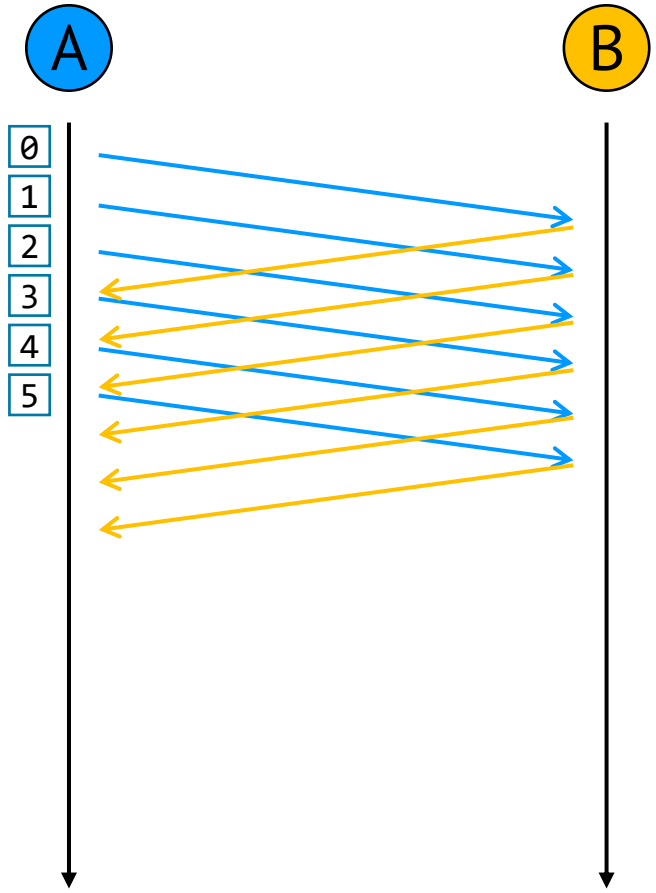
HA van új adat, amit el kellene küldeni, AKKOR
  HA az adattal együtt a buffer mérete > maximum segment size AKKOR
    küldjük el a buffert és az adatot
  EGYÉBKÉNT
    HA a legutolsó küldésünkre még nem kaptunk ACK-t, AKKOR
      tegyük be egy bufferba az adatot
    EGYÉBKÉNT
      küldjük el az adatot

Nagle algoritmus

Nagle bekapcsolva



Nagle kikapcsolva



#07/1 – Összefoglalás

Foglamak Azonnali ACK

Késleltetett ACK

Retransmission timeout

Algoritmus Nagle algoritmus

#07/2 – Torlódáskezelés

Torlódás (congestion)

Számítani kell rá, hogy sok eszközön megy át a csomag, amiket túlterhelhetünk, és amik csomagokat dobhatnak el.

Ha az elveszett csomagokat bambán újraküldjük, akkor tetézzük a bajt, mert további csomagokat viszünk a rendszerbe, fokozzuk a leterheltséget.

A torlódásokra két „kezelés” van:

- **congestion control** – mit tegyünk, ha torlódás van?
- **congestion avoidance** – mit tegyünk, hogy elkerüljük a torlódást?

Slow start

A slow start esetében nem csak a fogadó, hanem a küldő is flow controlt alkalmaz.

Bevezetünk egy **congestion window** ($cwnd$) mennyiséget, ez a hálózat kímélése érdekében létrehozott ablakméret.

A $cwnd$ méretét a küldő határozza meg, kezdetben klasszikusan 1 MSS méretű.

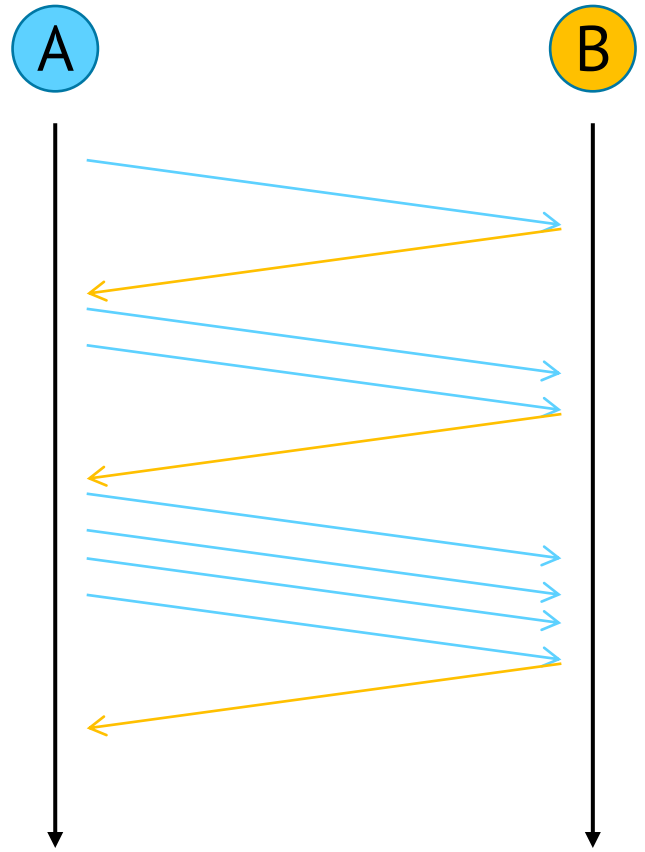
Küldési szabály:
Csak olyan adat küldhető, ami benne van a sliding window-ban ÉS a $cwnd$-ben is.

Slow start

A $cwnd$ mérete minden ACK-zott szegmens után növekszik. Ez praktikusan azt jelenti, hogy először 1, aztán 2, aztán 4, aztán 8... szegmens küldhető, vagyis az átvitel a slow start során exponenciálisan nő.

A slow start révén elkerülhetjük, hogy azonnal bajt okozzunk azzal, hogy egyből ráborítjuk a hálózatra az összes adatunkat.

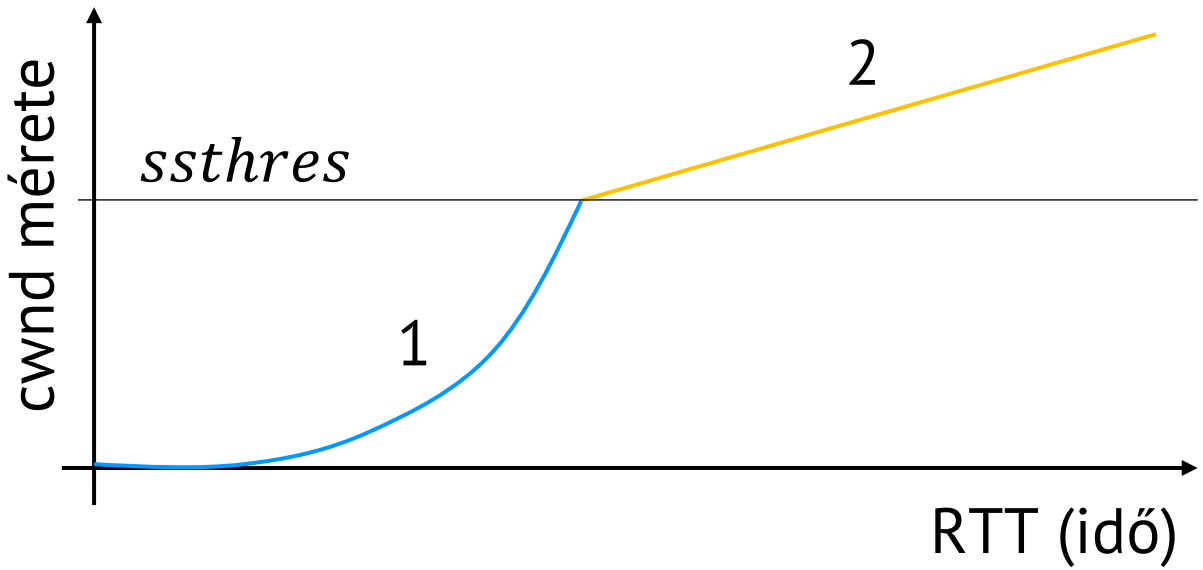
Ha a küldés közben egyszercsak torlódást észlelünk, akkor a collision avoidance algoritmus jut szóhoz.



Congestion aviodance

A congestion avoidance alkalmával a $cwnd$ értékét 1 szegmensnyivel növeljük minden RTT alkalmával. Ez egy lináris növekedést jelent.

Bevezetünk egy új változót, ez a slow start threshold ($ssthres$). Ez adja meg, hogy mikor fogunk exponenciális növekedést (slow startot) és mikor lineáris növekedést (congestion avoidance) alkalmazni.



1-es szakasz: slow start
akkor, ha $cwnd < ssthres$

2-es szakasz: congestion avoidance
egyébként

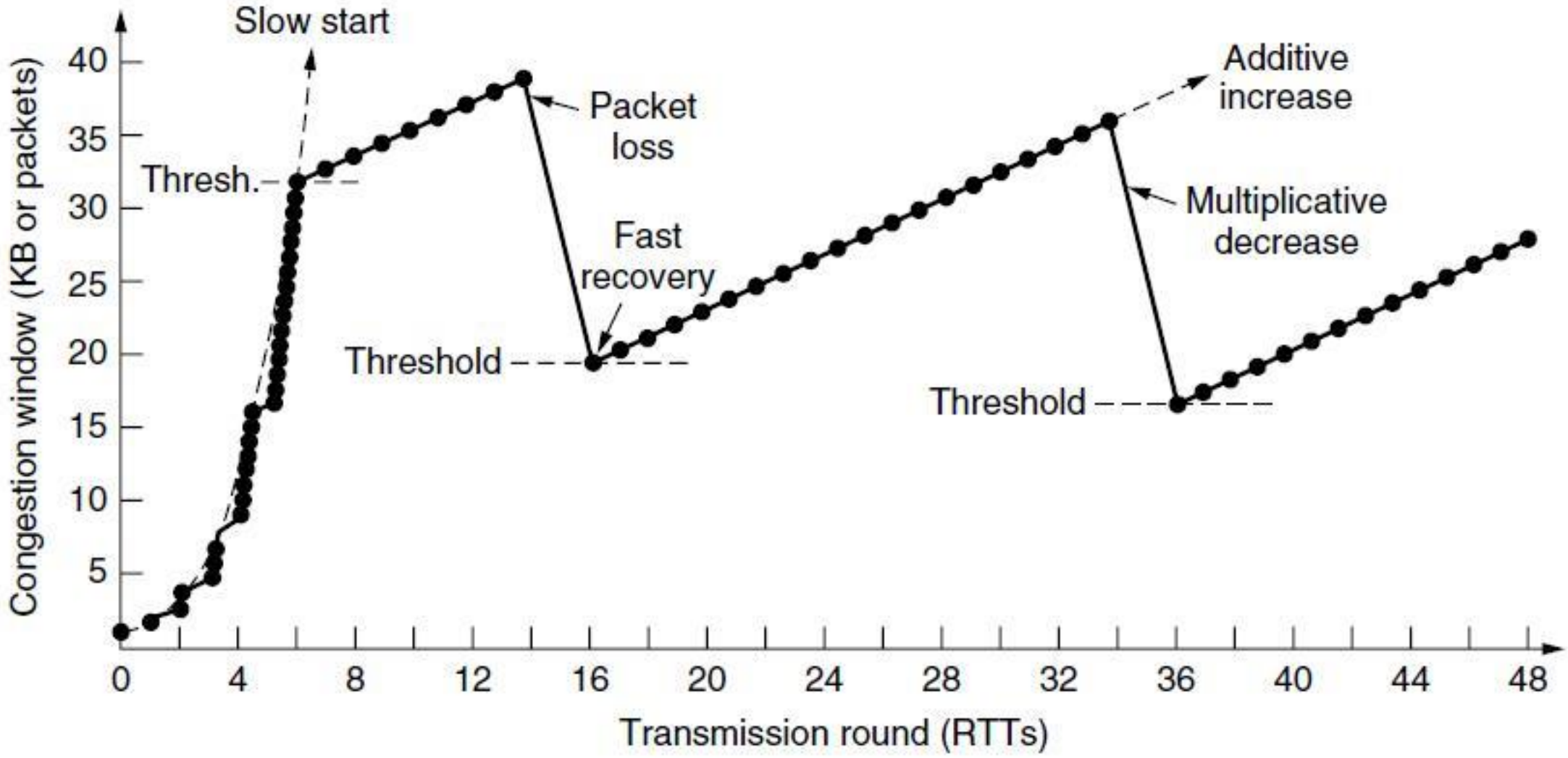
Congestion aviodance

Ha torlódást észlelünk… Torlódásnak számít a tripla ACK vagy az RTO idő túllépése.

Teendők:

1. Újraküldjük a szegmenst ***(fast retransmit)***
2. A nyugtázatlan byte-ok mennyiségének felére, de legfeljebb 2 MSS-re csökkentjük az $ssthresh$ értékét ***(multiplicative decrease)***
3. Hogyan induljunk újra?
   - Ha RTO miatt álltunk meg, akkor slow start-tal indulunk, ekkor $cwnd = 1$
   - Ha tripla ACK miatt álltunk meg, akkor $cwnd = ssthresh + 3$ MSS, és nem slow start, hanem congestion avoidance ***(fast recovery)***

Congestion aviodance



Torlódáskezelés hálózaton belül

A TCP congestion control eljárások élhetőbbé tették az internetet, a TCP kapcsolatok reagálnak a torlódásra.

Viszont a TCP csak két végpont közt van, és a hálózat belsejében is rendet kellene tenni, ott is kezelni kell a torlódásokat.

Torlódáskezelés hálózaton belül

A TCP congestion control eljárások élhetőbbé tették az internetet, a TCP kapcsolatok reagálnak a torlódásra.

Viszont a TCP csak két végpont közt van, és a hálózat belsejében is rendet kellene tenni, ott is kezelni kell a torlódásokat.

Hogyan teszünk rendet a hálózat belsejében?

Az egyik legkorábbi megoldás az, hogy a routerekben az egyes intefészekhez sorokat, várószobákat hozunk létre. Ha torlódás van, akkor betesszük a sorba a csomagot, amíg még fér. Ha lassabban halad a sor, mint ahogy töltődik, akkor azokat a csomagokat, amik nem férnek bele, eldobjuk.

Torlódáskezelés hálózaton belül

A sorok kezelése, viselkedése

Az alapötlet, miszerint ha nem fér bele a sorba, eldobjuk, nem túl szerencsés:

- egyes kapcsolatok monopolizálhatják az erőforrásokat
- ha bedugul a rendszer, nehéz kivergődni belőle
- ha megint érkezik egy csomag-cunami, akkor csak rosszabb lesz a helyzet
- egy-egy csomag eldobása még oké, de sorozatban sok törlése már gondot okoz

Torlódáskezelés hálózaton belül

A sorok kezelése, viselkedése

Mit akarunk a soroktól?

- Arra törekszünk, hogy tartósan kicsik legyenek a sorok
- Úgy is jönnek majd csomag-cunamik, amiket el kell viselni valahogy
- Az interaktív kapcsolatokban kibírhatatlanok a hosszú sorok, hosszú várakozások

Random Early Detection

A Random Early Detection (RED) megpróbálja megoldani, hogy rövidek maradjanak a sorok, és lehetőleg úgy, hogy igazságos legyen a rendszer.

- Az alapelv, hogy minden bejövő csomagot valamilyen valószínűséggel eldobunk.
- A valószínűség annál nagyobb, minél hosszabb volt a sor az elmúlt időben.
- Ha véletlenszerűen dobáljuk el a csomagokat, akkor kisebb az esélye, hogy teljesen megölünk egy összetartozó csomag-sorozatot.

A Weighted RED-ben a csomagoknak fontossága van, ezáltal vannak csomagok, amiket kisebb eséllyel dobunk el.

A fontosság alapja lehet IP cím, protokoll, TOS adat, bármi...

#07/2 – Összefoglalás

Eljárások

slow start

fast retransmit

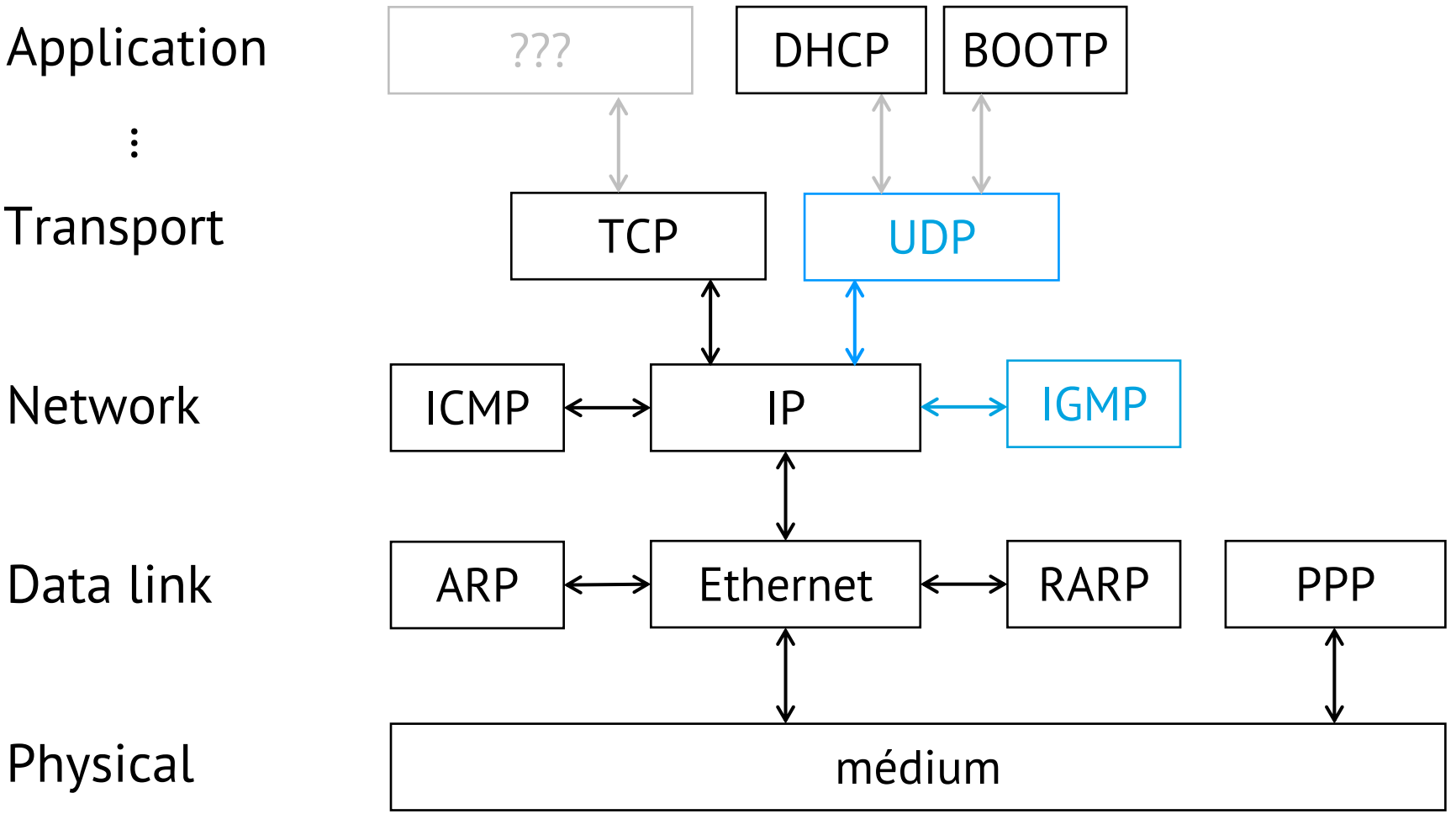
multiplicative decrease

fast recovery

RED

#07/3 – UDP

Itt tartunk most



hálózati réteg beállítása, bootolás

adatszállítás, alkalmazások, portok, handshake

router, útvonalak, irányítás, logikai cím (IP cím)

bridge, switch, fizikai cím (MAC) hibakezelés, ütközés, VLAN

vezeték, repeater, hub, elektromos jelek, zaj

A transport layer implementációi

User Datagram Protocol (UDP)

- Összeköttetés nélküli protokoll
- Nem megbízható, mert nincs benne visszajelzés a kézbesítésről.
- Nincs sorrendtartás.
- Rendkívül gyors és hatékony.

„valakire rákiabálás”

Transmission Control Protocol (TCP)

- Összeköttetés alapú protokoll
- Megbízható, mivel visszajelzést ad a szegmensek megérkezéséről.
- Sorrendtartó átvitel.
- Lassúbb az összeköttetés felépítése, de utána gyors az adatátvitel.

„személyes beszélgetés”

Port és socket

A korábban megismert port és socket fogalom a TCP és UDP esetében is helyes.

port az alkalmazást azonosító, a multiplexálást és demultiplexálást segítő, a transzport réteg szintjén kezelt fogalom.

socket az interfész IP címe, a port száma és a használt protokoll által megadott, az operációs rendszerben kezelt fogalom.

A socket definícióból következik, hogy ugyanahhoz a portszámhoz két socket létezhet, az egyik a TCP, a másik az UDP protokollra; és a két socket akár teljesen más alkalmazásé is lehet.

Port és socket

Nem mindegy, hogy egy adott számú porton csak TCP, csak UDP vagy mindkét protokollt támogatjuk.

Példák:

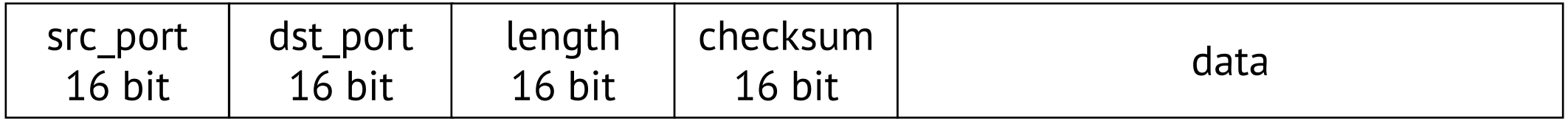
53-as port	Domain Name System	alapból UDP, de néha a válasz TCP
67-es port	BOOTP és DHCP	csak UDP
80-as port	HTTP	csak TCP

User Datagram Protocol

Az UDP egy nagyon gyakori 4. szintű protokoll.

- nem kapcsolatorientált
  (nincs kapcsolat felépítés, állapot vagy bontás)
- nem garantál megbízható átvitelt
  (nem biztos, hogy az információ egyáltalán eljut a címzetthez vagy hogy a sorrend megmarad)
- ellenőrzőösszegeket használ
  (ezzel igyekszik védeni az adat integritását)
- nincs kapcsolat épités és nyugtázás → gyors adatátvitelt tesz lehetővé
  (pl. televíziós adás, telefónia)
- nincs kapcsolatorientáltság → több címzett is lehet
  (multicast, broadcast)

UDP fejrész



- src_port a forrás oldali port száma
- dst_port a céloldali port száma
- length az UDP csomag hossza a fejléccel együtt byte-ban mérve
- checksum az adat vagy legalább a fejléc integritásának ellenőrzésére használható ellenőrző összeg

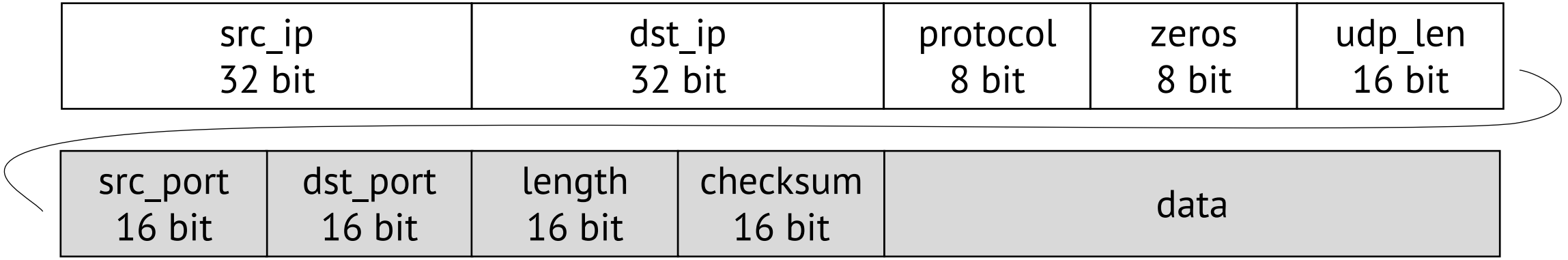
UDP checksum

Az UDP ellenőrzőösszeg többféle lehet:

- **csupa nulla**
  ebben az esetben nem ellenőriz semmit sem
- **„hétköznapi” ellenőrzőösszeg**
  ez az UDP csomag fejlécét és az adatot veszi alapul
- **pszeudo fejléc alapján számolt ellenőrzőösszeg**
  ebben az esetben nem csak az UDP csomag, hanem az IP fejléc egyes paraméterei is számítani fognak a checksum számolásnál
- **vonatkozhat az UDP csomagnak csak egy részére is**
  tehát pl. lehet az, hogy nem az egész fejléc+adat értékre van számolva, hanem csak mondjuk az UDP fejlcére és az adat első 4 byte-jára

UDP checksum pseudo fejléccel

Az UDP ellenőrzőösszeg számolásakor az IP csomag fejrészécenk egyes paramétereit is figyelembe vesszük egy ún. pszeudo fejléc képében:



- src_ip, dst_ip — forrás és cél IP címe
- protocol — az IP fejléc protokoll mezőjének értéke
- zeros — nullák (helykitöltő)
- udp_len — az UDP rész teljes hossza

UDP Lite

UDP Lite esetében az ellenőrzőösszeg csak az UDP csomag egy részére érvényes.

Egyes alkalmazások esetében (pl. videó vagy hang) hasznos lehet megtartani a sérült csomagot is, mert az a sérülés ellenére is a nagyon-nagyon magas réteg (pl. emberi fül) számára még értelmezhető adatot tartalmaz.

Az UDP Lite fejlécben a length mezőben az ún. checksum coverage érték van megadva. Ez azt mondja meg, hogy az UDP fejrész kezdetétől hány byte-ra számoljuk az ellenőrző összeget (min. 8 byte, azaz a fejlécnek benne kell lenni).

UDP csomagméretek

Minimális csomagméret

Az UDP fejléc mérete 8 byte (2 + 2 + 2 + 2). Lehetséges, hogy nem küldünk adatot, ekkor a teljes csomag csak a fejlécből áll.

Maximális elméleti adatméret

A fejlécben lévő length mező maximális értéke 65535, azaz ennyi byte lehet az elméleti leghosszabb UDP-vel küldött adat. Csakhogy! Az egész UDP csomagnak bele kell férnie az alatta lévő IP réteg csomagméretébe.

Maximális gyakorlati csomagméret

Az UDP fejrész 8 byte, az IPv4 fejrész 20 byte, a max. IP csomagméret 65535 byte. Ebből következik, hogy a max UDP adat 65535 – 20 – 8 = 65507 byte lehet.

Az operációs rendszer és egyes alkalmazások azonban még ekkora méretet sem engednek meg. A gyakorlatban általában 2-hatvány értékek vannak (8192, 512).

#07/3 – Összefoglalás

Fogalmak UDP jellemzői

UDP Lite ötlete

Képesség Felismerni és azonosítani az UDP szegmens részeit

#07/4 – Multicast és broadcast

Multicast és broadcast

Multicast
Egyetlen küldő több másik címzett fogadónak küld (rádió)

Boradcast
Egyetlen küldő mindenkinek küld (sziréna)

Fontos, hogy egyik esetben sincs címzett, hanem egy bizonyos *csoport* az, akinek az üzenetet szánjuk.

A multicast és broadcast küldési formák kapcsolatorientált rendszerben, ahol pontosan 1 db címzett van, nem használhatók.

Multicast és broadcast

Motiváció:
Lehet, hogy abszolút fogalmam sincs, hogy mi a helyzet a hálózatban, de a konfigurációs üzeneteket valahogy meg akarom kapni (pl. DHCP) => broadcast

Lehet, hogy a hálózatban egyidejűleg sokan ugyanarra az adatra kíváncsiak. Ekkor megtehetem, hogy a kíváncsi eszközök interfészeinek nem egyesével küldök el mindent külön-külön, hanem egy közös multicast csoportba teszem őket, és annak a csoportnak az IP címe lesz a címzett. A többit már az alsóbb rétegek megoldják. Ezzel meg tudom spórolni, hogy pl. 20 eszköznek 20 példányban küldjek el valamit; multicasttal elég lesz egy darab csomag is, amit majd a switchek megsokszoroznak.

Multicast és broadcast L2 szinten

Az Ethernet szabvényban a MAC cím alapján lesz valami uni-, multi-, vagy broadcast.

Multicast: a MAC cím első byte-ja páratlan értékű (pl. `*1:**:**:**:**:**`)

Broadcast: a MAC cím `FF:FF:FF:FF:FF:FF`

Láttunk már ilyet: ARP request a broadcast címre megy

A MAC címekkel csak data link layer szinten tudunk frame-eket küldeni; ezek nem route-olhatók. A hálózatok közötti átvitelhez IP-re van szükség.

Multicast és broadcast L3 szinten

IPv4 implementációban az IPv4 cím alapján lesz valami uni-, multi-, vagy broadcast.

Multicast: a `224.0.0.0` és a `239.255.255.255` közti címtartomány

Broadcast: a `255.255.255.255` cím

Láttunk már ilyet: a router solicitation és router advertisement a 224.0.0.1 (all hosts) és 224.0.0.2 (all routers) multicast csoportok címét használja.

A multicast tartományon belül vannak lokálisan kioszthető és világállandó címek, utóbbiak természetesen route-olhatók.

Multicast és broadcast L4 szinten

A transport layer szintjén is meg kell (lehet) valósítani
a multicast és broadcast üzenetek küldését, fogadását.

Multicast és broadcast lényege: több címzett van

↓

nem lehet kapcsolatorientált

↓

nem lehet TCP-t használni

Multicast és broadcast L4 szinten

A transport layer szintjén a multicast és boradcast üzenetek átviteléhez UDP-t használunk.

Láttunk már ilyet: DHCP és BOOTP az UDP-t használja a 67-es és 68-as portokon. Erről csak annyit beszéltünk, hogy broadcast üzenetet küldenek. Most már tudjuk, hogy a DHCPDISCOVER üzenet pl. egy UDP broadcast csomag, ahol 0.0.0.0 a kliens IP, 68 a kliens port és 255.255.255.255 a cél IP és 67 a cél port.

Még több multicast

További fontos szempontok a multicast szükségességének indoklására:

- A broadcast olykor fölöslegesen terheli a hálózatot
  (minek ébresztjük föl a hálózati interfészeket csak azért, hogy azok aztán feladják a csomagot, amiről a felettes réteg úgy fog dönteni, hogy kuka).
- Nem ritkán éppen a legnagyobb adatforgalmat bonyolító protokollok küldik egyszerre több címzettnek is ugyanazt az adatot (pl. élő közvetítés)
- Szükség van arra, hogy routereken át haladjon a forgalom.
- Szükség van arra, hogy az egyes interfészek meghatározhassák, hogy mely multicast üzenetek érdeklik őket.

Multicast csoportok

L1 szinten — nem foglalkozunk azzal, hogy mi a csomag tartalma.

L2 szinten — a MAC cím alapján döntünk, hogy mit kezdünk a frame-mel.

A hálózati interface-ek a broadcast címre jövő frame-eket automatikusan feladják, a multicast címekre jövőt csak akkor, ha azt külön kérjük.

L3 szinten — az IP cím alapján döntünk, hogy mit kezdjünk a csomaggal.

A network layer a broadcast IP címre jövő frame-eket automatikusan feladja, a multicast címekre jövőt csak akkor, ha benne vannak az adot multicast cím szerint meghatározott csoportban.

Ahhoz tehát hogy multicast IP címre érkező adatot fogadhassunk, benne kell lennünk a megfelelő csoportban.

Multicast csoportok

A multicast csoportba való tartozás lehet

- automatikus: pl. az all hosts csoportban minden host alapból benne van
- belépés alapján megvalósult: a hálózati réteg felsőbb utasításnak engedelmeskedve elkezdi figyelni a multicast címre jövő dolgokat is.

Az ARP-nak kezelnie kell, hogy a multicast IP-khez multicast MAC címek tartoznak.

A routereknek tudniuk kell, hogy mely interfészeiken mely multicast csoportok vannak, tehát mit hova kell továbbítani.

Multicast és az ARP

Az ARP táblában azt tároljuk, hogy az egyes IP címekhez milyen MAC címek tartoznak. Az unicast MAC és IP közötti megfelelést az ARP kérés-válasz fogja megoldani. Mi a helyzet a multicast címekkel?

A multicast IP címeket az utolsó 23 bit felhasználásával mappeljük a MAC cím-tartomány egy részére:

Multicast és az ARP

Az ARP táblában azt tároljuk, hogy az egyes IP címekhez milyen MAC címek tartoznak. Az unicast MAC és IP közötti megfelelést az ARP kérés-válasz fogja megoldani. Mi a helyzet a multicast címekkel?

A multicast IP címeket az utolsó 23 bit felhasználásával mappeljük a MAC cím-tartomány egy részére:

```
   01    :    00    :    5E    :    00    :    00    :    00
00000001:00000000:01011110:00000000:00000000:00000000
```

Multicast és az ARP

Az ARP táblában azt tároljuk, hogy az egyes IP címekhez milyen MAC címek tartoznak. Az unicast MAC és IP közötti megfelelést az ARP kérés-válasz fogja megoldani. Mi a helyzet a multicast címekkel?

A multicast IP címeket az utolsó 23 bit felhasználásával mappeljük a MAC cím-tartomány egy részére:

```
   01   :   00   :   5E   :   00   :   00   :   00          231   .  205   .   98   .  177
00000001:00000000:01011110:00000000:00000000:00000000    11100111.11001101.01100010.10110001
```

Multicast és az ARP

Az ARP táblában azt tároljuk, hogy az egyes IP címekhez milyen MAC címek tartoznak. Az unicast MAC és IP közötti megfelelést az ARP kérés-válasz fogja megoldani. Mi a helyzet a multicast címekkel?

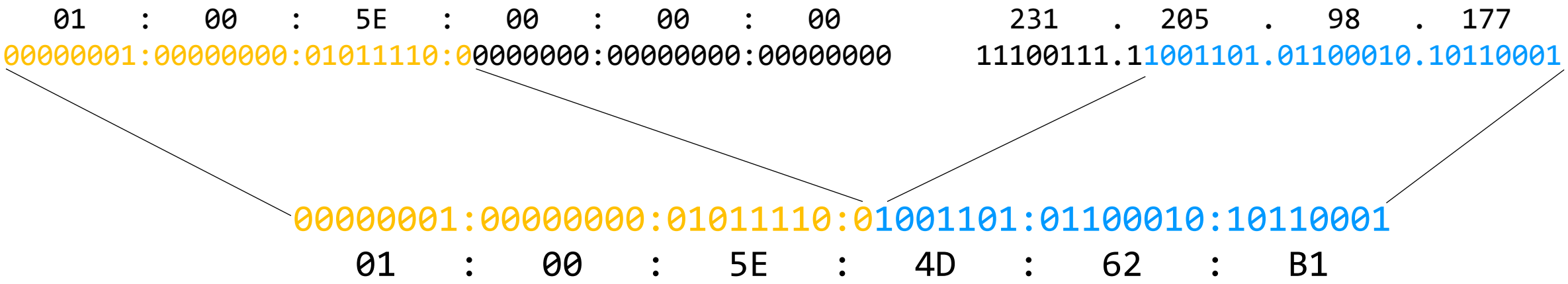
A multicast IP címeket az utolsó 23 bit felhasználásával mappeljük a MAC cím-tartomány egy részére:

```
   01   :   00   :   5E   :   00   :   00   :   00          231   .  205   .   98   .  177
00000001:00000000:01011110:0000000:00000000:00000000     11100111.1001101.01100010.10110001
```

Multicast és az ARP

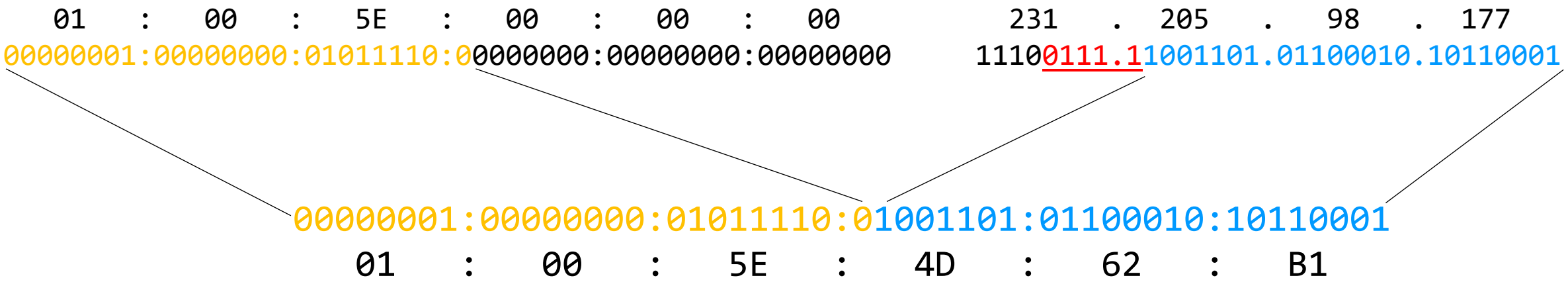
Az ARP táblában azt tároljuk, hogy az egyes IP címekhez milyen MAC címek tartoznak. Az unicast MAC és IP közötti megfelelést az ARP kérés-válasz fogja megoldani. Mi a helyzet a multicast címekkel?

A multicast IP címeket az utolsó 23 bit felhasználásával mappeljük a MAC cím-tartomány egy részére:



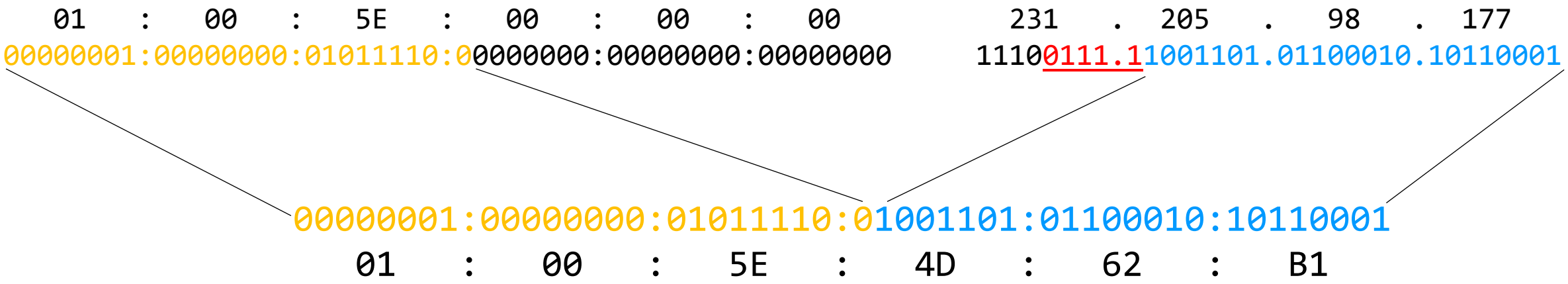
Multicast és az ARP

A mappelős módszer igen jól megoldja az alapvető problémát (IP és MAC összerendelése), viszont nem tökéletes. Sajnos van 5 bit, ami nem kerül bele a MAC címbe, tehát két olyan címnek, ami csak ebben az értékben tér el, azonos lesz a multicast MAC címe.



Multicast és az ARP

A mappelős módszer igen jól megoldja az alapvető problémát (IP és MAC összerendelése), viszont nem tökéletes. Sajnos van 5 bit, ami nem kerül bele a MAC címbe, tehát két olyan címnek, ami csak ebben az értékben tér el, azonos lesz a multicast MAC címe. Ez nem baj, mert az IP réteg majd megoldja az ütközést.



#07/4 – Összefoglalás

Fogalom Broadcast és multicast megvalósítása L2, L3, L4 szinten
Multicast csoportok

Eljárás ARP multicast esetén

#07/5 – IGMP

Internet Group Management Protocol

A routerek esetében már nem olyan egyértelmű, hogy miként kell bánni a multicast üzenetekkel. Irányításra van szükség!

IGMP – Internet Group Management Protocol

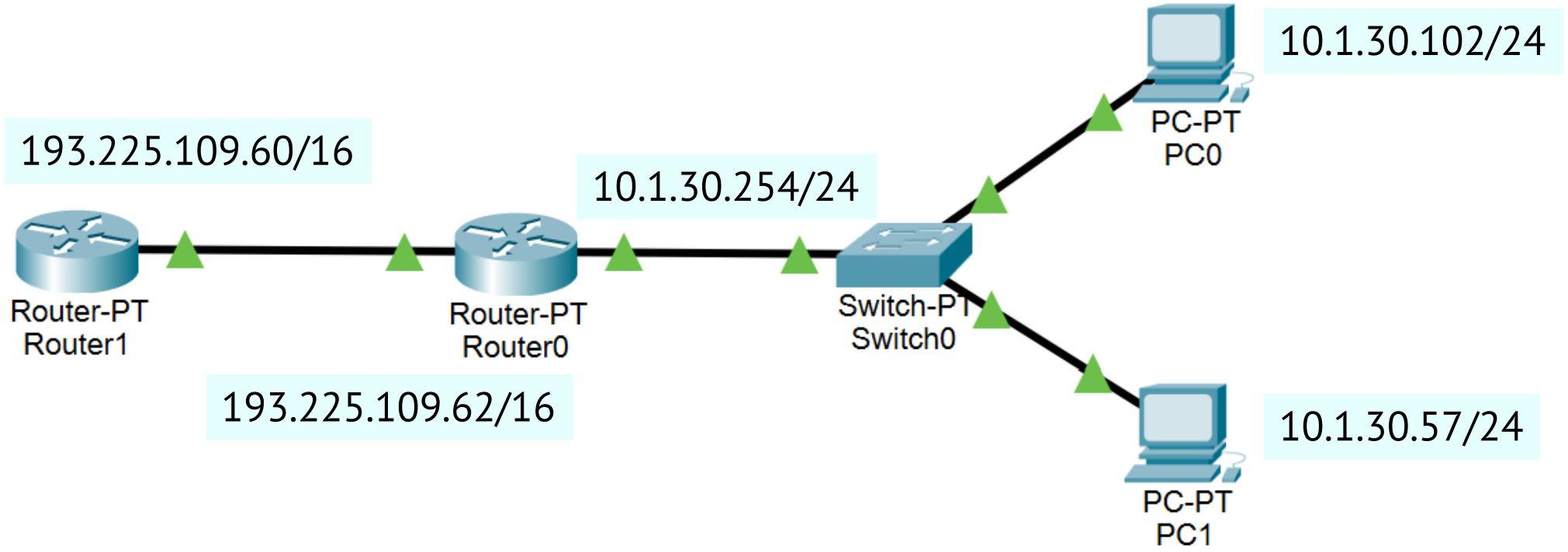
Feladata a multicast csoportok szervezése.

A router és a routerhez közvetlenül kapcsolódó hostok között dolgozik.

Az IGMP a network layer szintjén van implementálva (akárcsak az ICMP).

IGMP feladat 1.

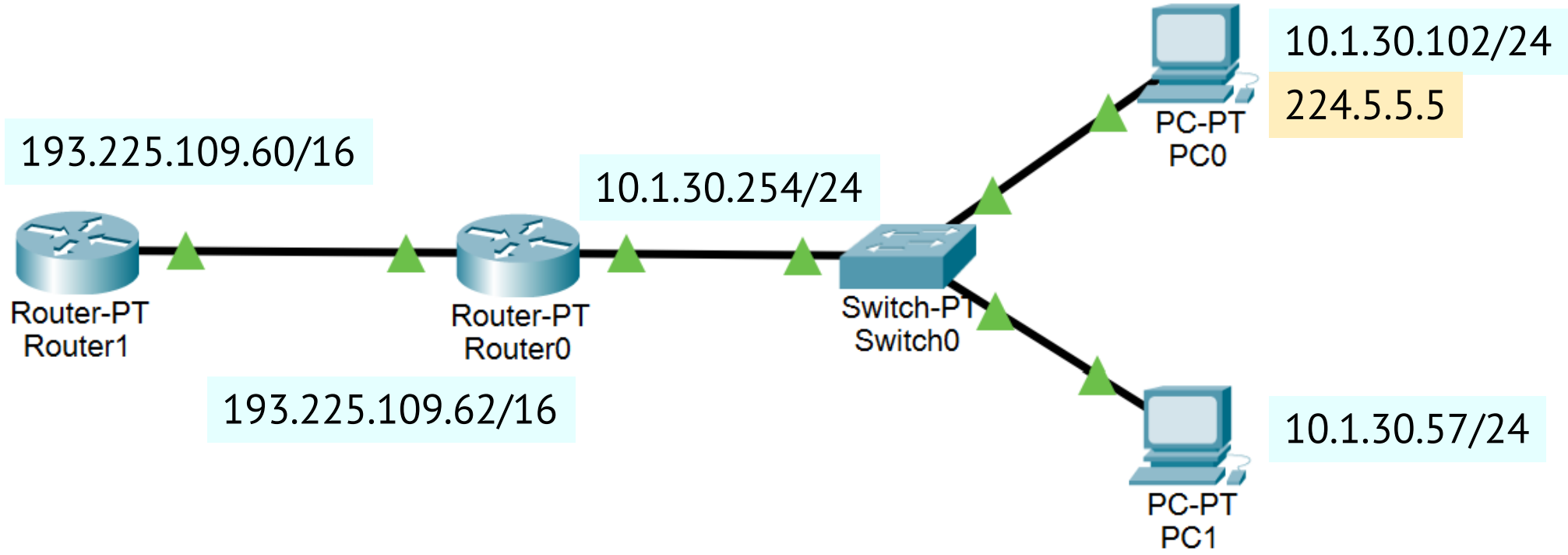
Legyen adott az alábbi hálózat



IGMP feladat 1.

Legyen adott az alábbi hálózat

A PC0 eszköz belép a 224.5.5.5 multicast csoportba. Hogyan kapja meg az üzeneteket?

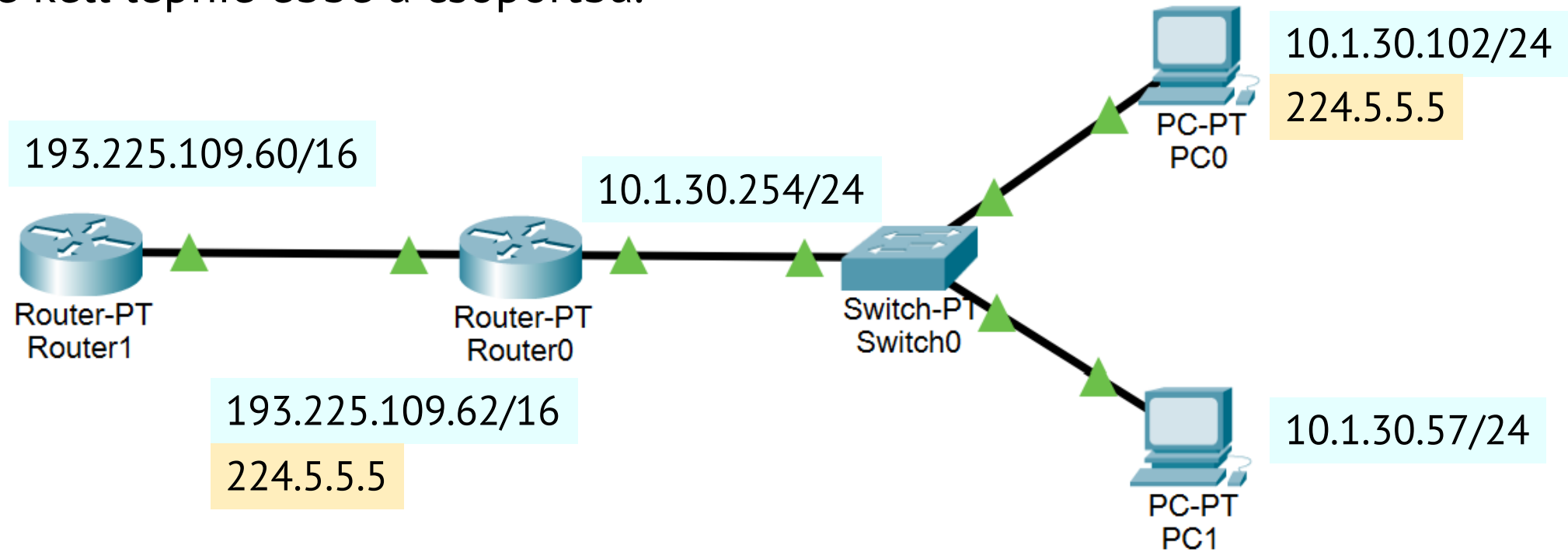


IGMP feladat 1.

Legyen adott az alábbi hálózat

A PC0 eszköz belép a 224.5.5.5 multicast csoportba. Hogyan kapja meg az üzeneteket?

Megoldás: a Router0 eszköznek tudnia kell, hogy van belül egy ilyen eszköz.
Neki is be kell lépnie ebbe a csoportba.



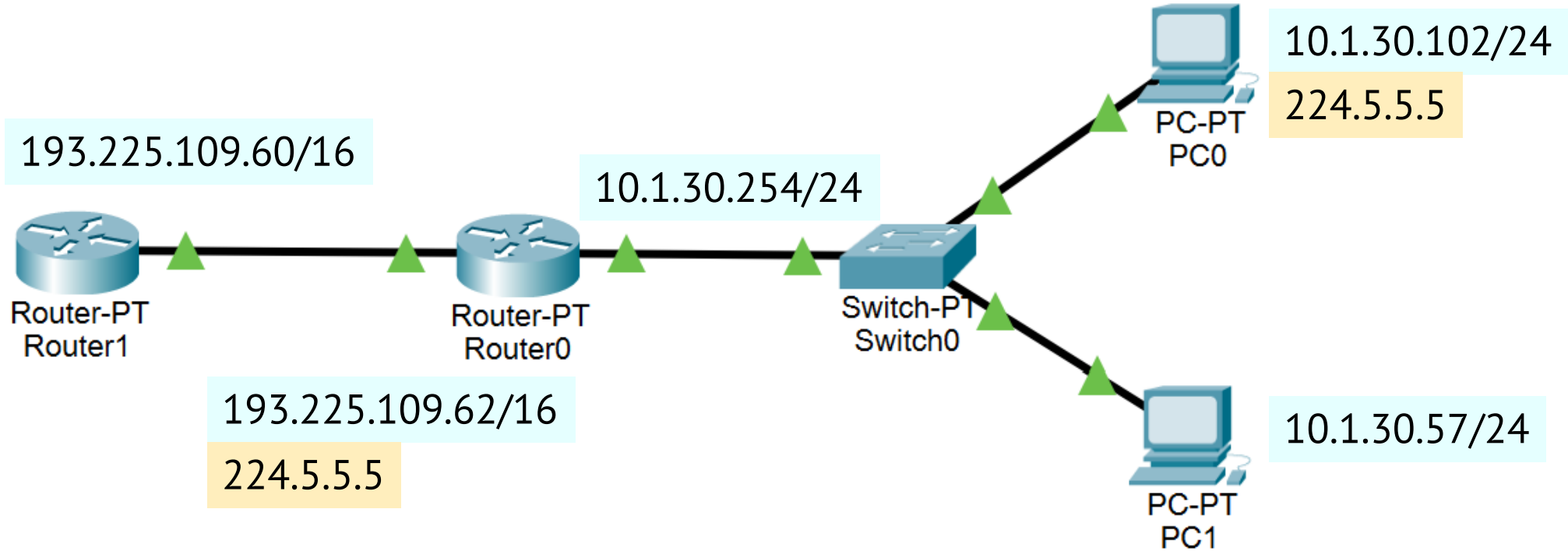
IGMP feladat 1. tanulság

Kell valamilyen együttműködés a végponti hálózati eszköz és a router között.

Ha a végponti eszköz belép egy multicast csoportba, akkor erről a routernek is tudnia kell.

IGMP feladat 2.

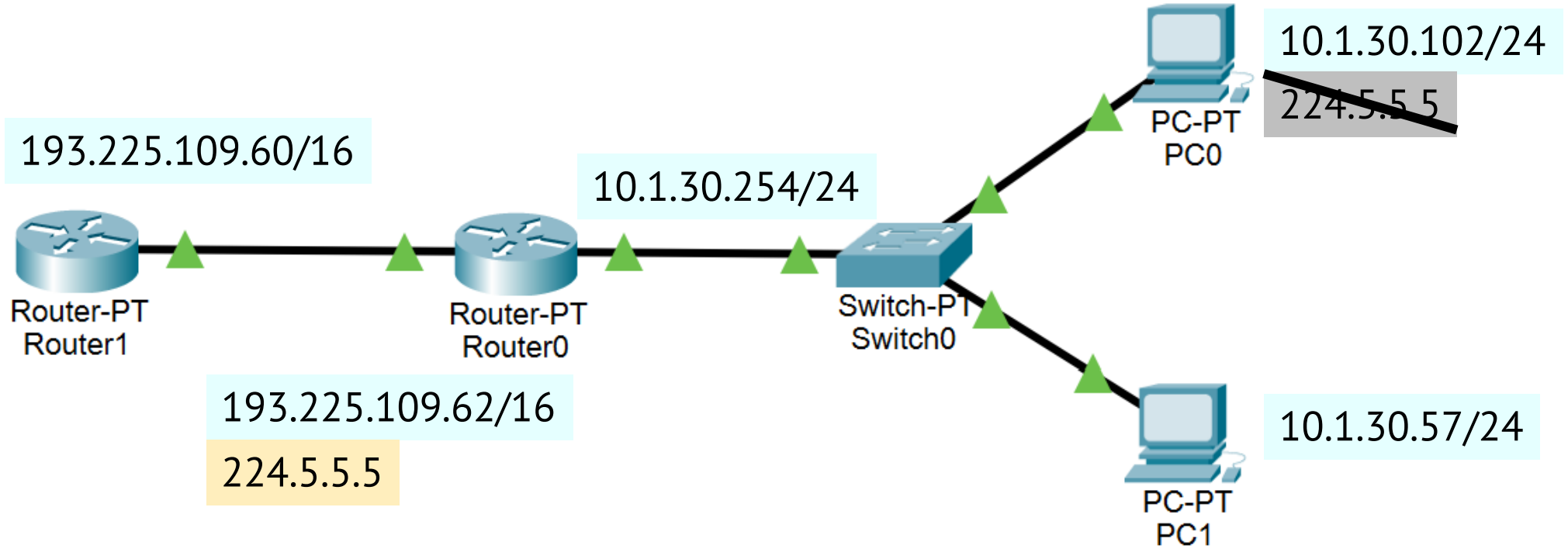
Legyen adott az alábbi hálózat



IGMP feladat 2.

Legyen adott az alábbi hálózat

A PC0 eszköz kilép a 224.5.5.5 multicast csoportból.

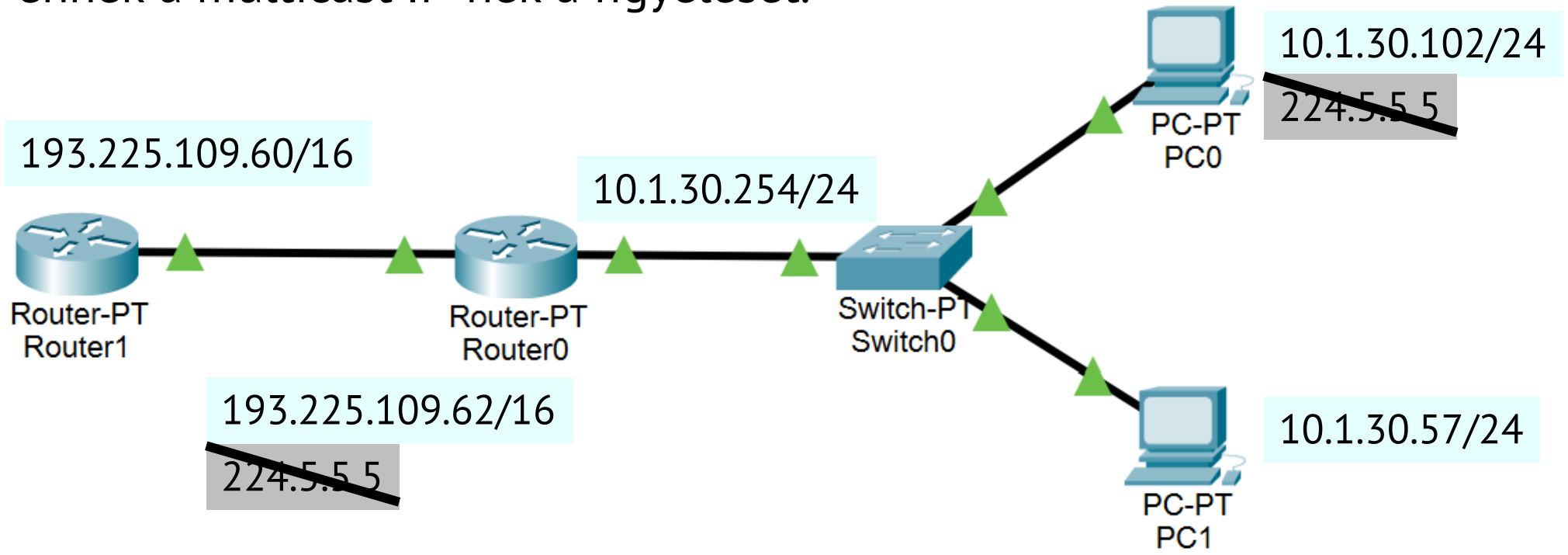


IGMP feladat 2.

Legyen adott az alábbi hálózat

A PC0 eszköz kilép a 224.5.5.5 multicast csoportból.

A Router0 eszközben (ha más, utána lévő eszköz nem igényli), akkor meg lehet szüntetni ennek a multicast IP-nek a figyelését.



IGMP feladat 2. tanulság

Kell valamilyen együttműködés a végponti hálózati eszköz és a router között.

Ha a végponti eszköz kilép egy multicast csoportból, akkor erről a routernek is érdemes tudnia.

IGMP üzenettípusok

IGMP Report – egy host küldi a routernek:
„Kedves router! Szeretném, ha figyelnéd kifelé a 224.5.5.5 címet, és visszaküldenéd nekem az onnan jövő multicast dolgokat! Köszi”

IGMP Query – a router küldi a 224.0.0.1 (all hosts) címre:
„Kedves hostok! Azt mondtátok, hogy figyeljem a 224.5.5.5 címet. Erre szükség van még? Van valaki ebben a csoportban vagy abbahagyhatom?”

IGMP Leave – egy host küldi a routernek:
„Kedves router! Már nem érdekel a 224.5.5.5 cím, ha nincs rajtam kívül más érdeklődő, akkor be lehet szüntetni a kukucskálást. Köszi!”

IGMP verziók

Az IGMP üzenet IP csomagba csomagolva közlekedik; az IP fejrészben a protokoll mező értéke 0x02.

Az IGMP-nek három verziója van:

- IGMPv1
- IGMPv2
- IGMPv3

IGMPv2

Ez az alapértelmezett IGMP üzenetverzió. Az üzenet a következő módon épül föl:

type 8 bit	max_resp_time 8 bit	checksum 16 bit	group_address 32 bit

- type — az IGMP üzenet típusa
  - `0x11` query
  - `0x16` report
  - `0x17` leave
- max_resp_time — a query üzenetekben ennyi ideig várunk válaszra
- checksum — a szokásos ellenőrzőösszeg az üzenetre
- group_address — a csoport, amiről szó van, 0 esetén minden csoport

Milyen címre küldjük az IGMP üzenetet?

IGMP Query üzenetek címzettje

Az IGMP Query üzeneteket az all hosts multicast csopotba küldjük (vagyis minden hoszt számára szól a kérdés). Az all hosts cím 224.0.0.1

IGMP Report és Leave üzenetek címzettje

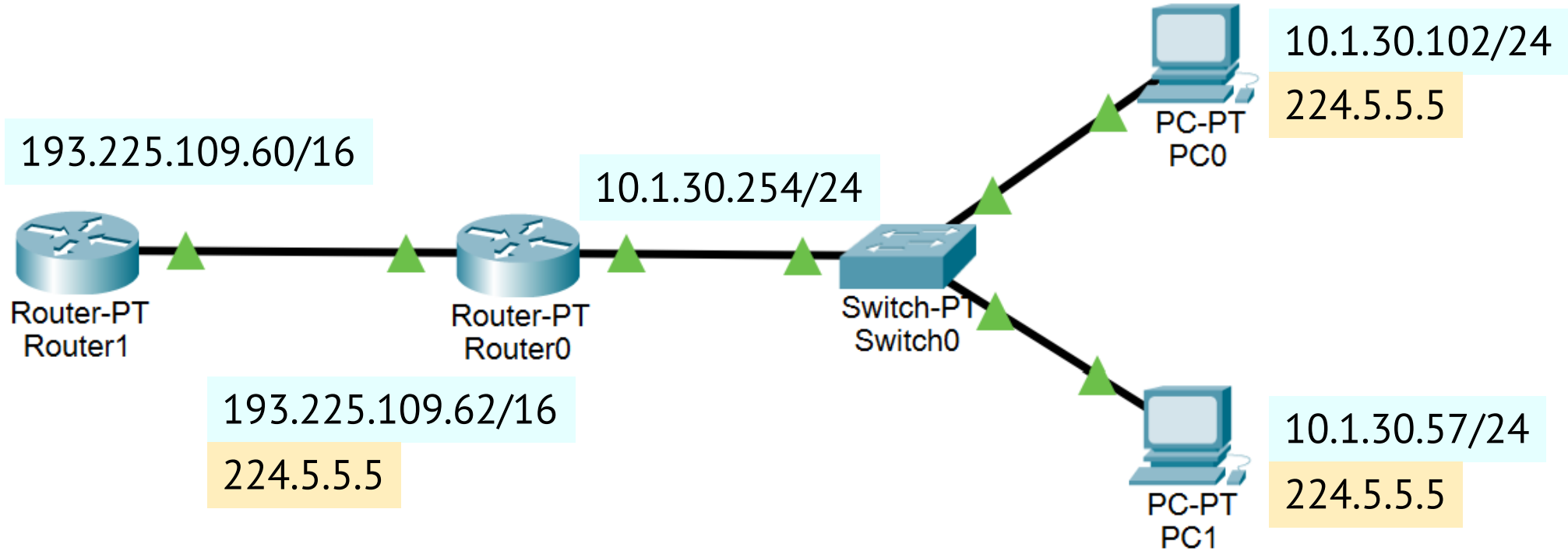
Az IGMP Report üzeneteket a kiszemelt router, vagy jellemzően az all routers multicast csoportba küldjük. Ennek címe 224.0.0.2

Az egyéb üzeneteket a megfelelő multicast csoportnak küldjük meg.

Switchek helyzete

Legyen adott az alábbi hálózat

Mi történik az L3 szintű multicast üzenetekkel a Switch0 esetében?



Switchek helyzete

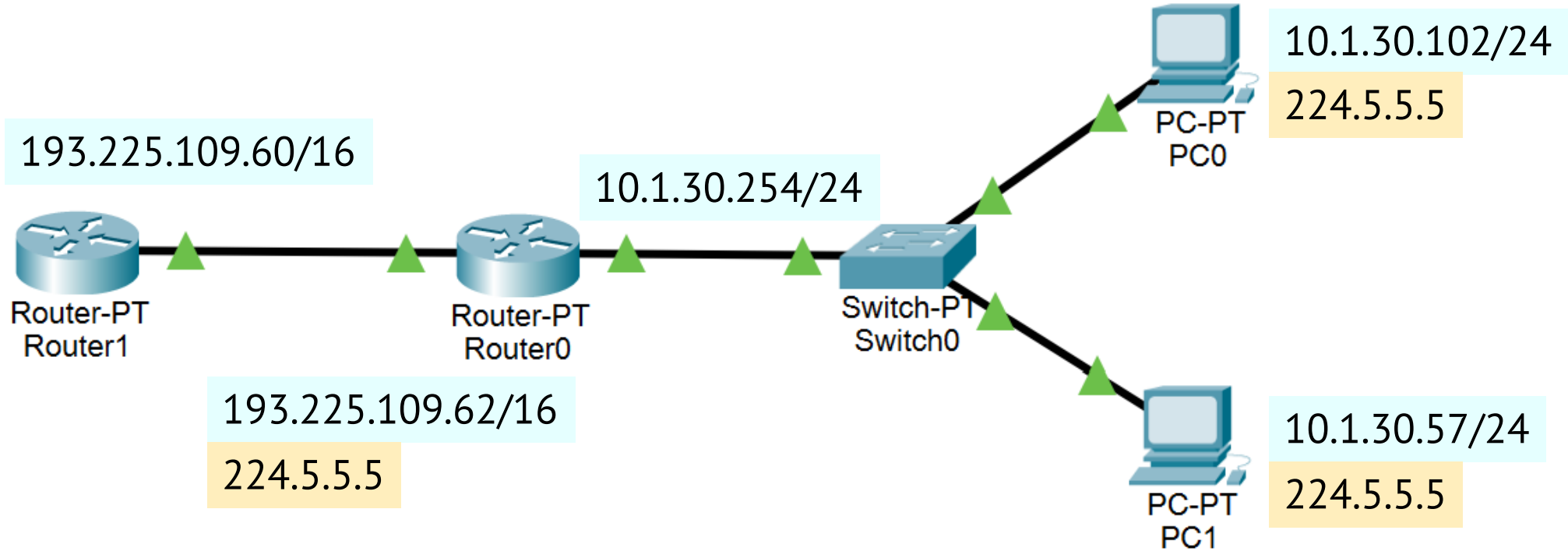
A switchek az IP réteg alatt működnek, ezért az IGMP beszélgetésben nem vesznek részt. Mindezek miatt az ő helyzetük érdekes...

- Lehet, hogy a multicast üzenetet minden interfészen megismétlik.
- Lehet, hogy a rendszergazda erővel beállítja, hogy mit melyik interfészen kell továbbítani.
- Lehet, hogy a switchek „lehallgatják” az IGMP beszélgetést, és ez alapján ők is megtanulják, hogy melyik interfészükön melyik eszköz melyik multicast csoportban van benne.
- És az is lehet, hogy a router és a switch egymással beszéli meg valami okos módon, hogy mi van (pl. a Cisco csinál ilyen furfangokat).

Routerek helyzete

Legyen adott az alábbi hálózat

Ha a multicast csoportba címzett üzenet a Router1-en kersztül jönne, akkor a Router0 és Router1 hogyan egyeznek meg arról, hogy milyen multicast csoportokban vannak?

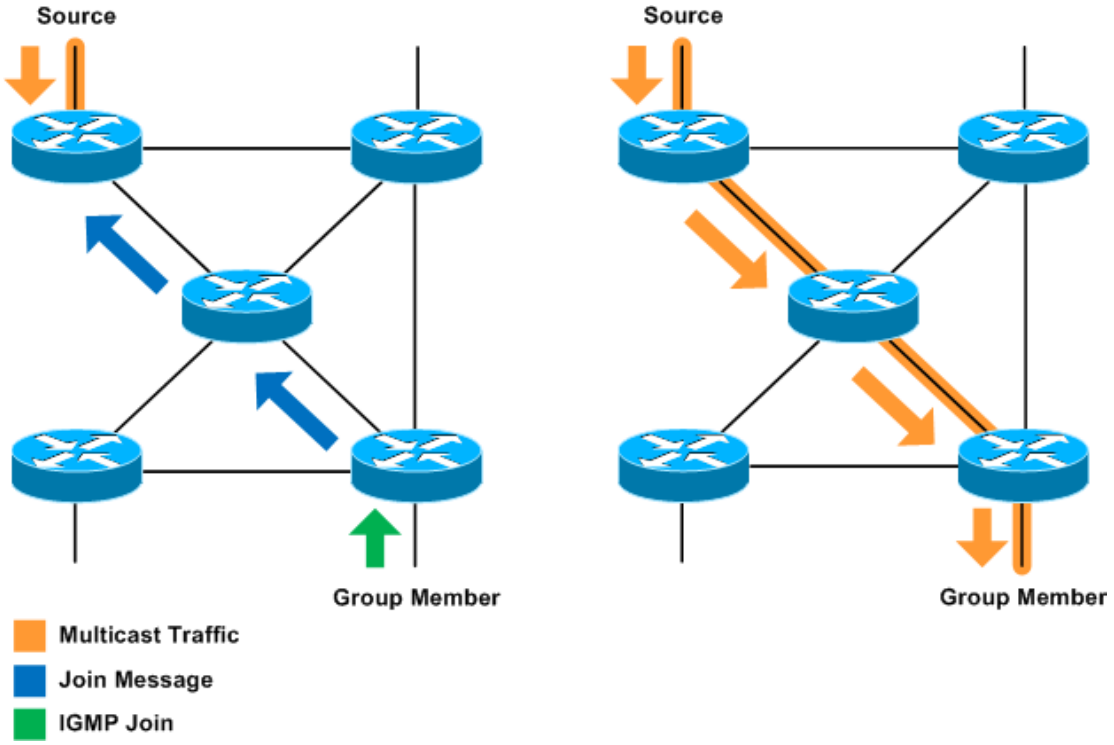


Protocol Independent Multicast (PIM)

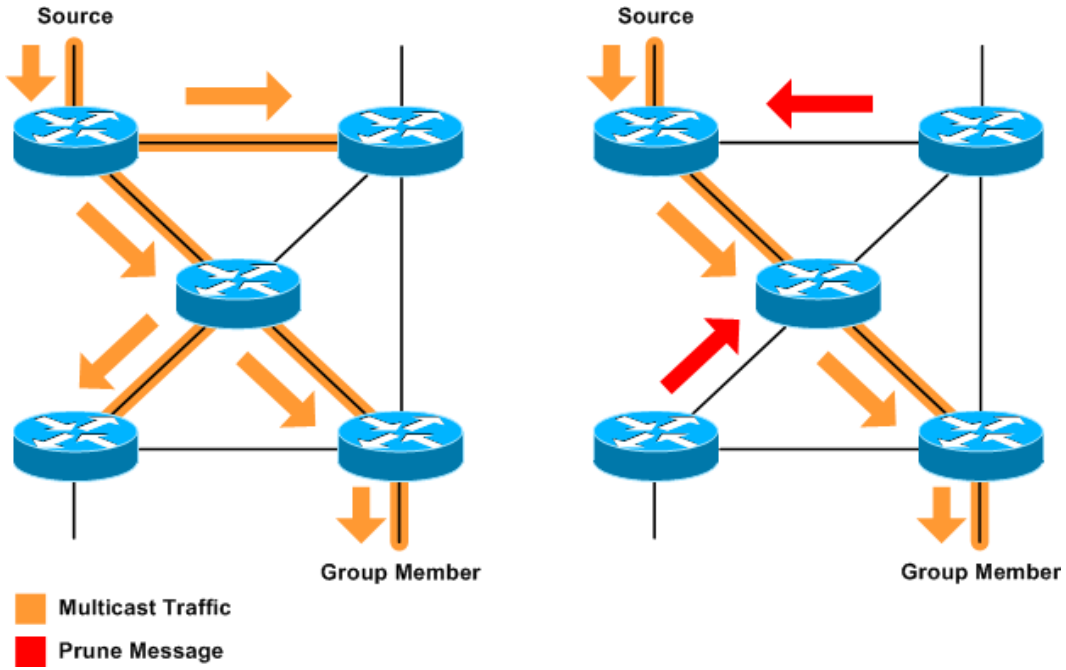
Az IGMP-vel csak a közeli LAN-okon át lehet multicastot szervezni.

A routerek közi multicast szervezés újabb protokollokat igényel. *Léteznek ilyenek :)*

Belépés



Kilépés



#07/5 – Összefoglalás

Foglamak

IGMP rendeltetése

IGMP feladatok

IGMP üzenettípusok

Switchek helyzete

Routerek helyzete

VÉGE

PÁZMÁNY

Pázmány Péter Katolikus Egyetem

Információs Technológiai és Bionikai Kar