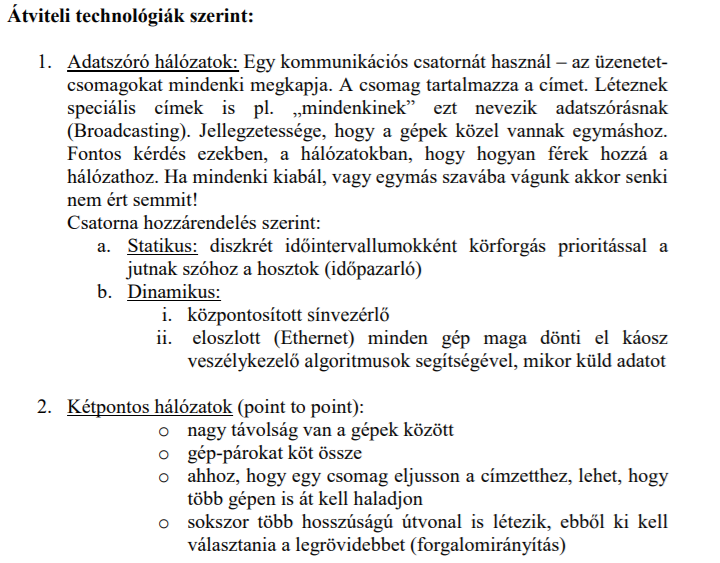
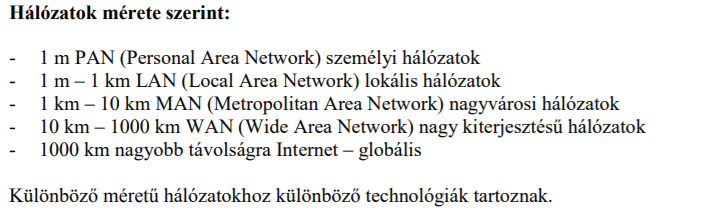
**1. Az OSI és a TCP/IP referencia modell rétegei és a rétegek feladatai.**

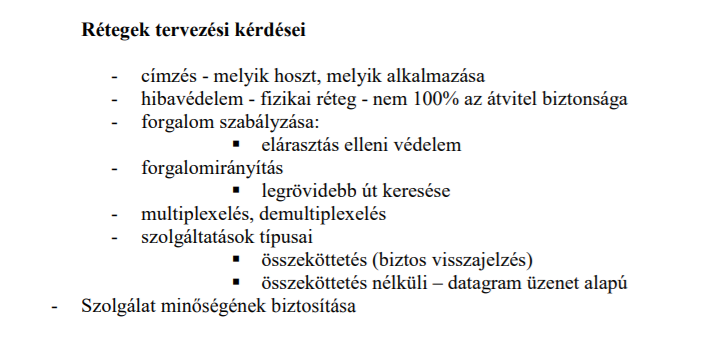
- Kommunikáció – információ (adat) csere

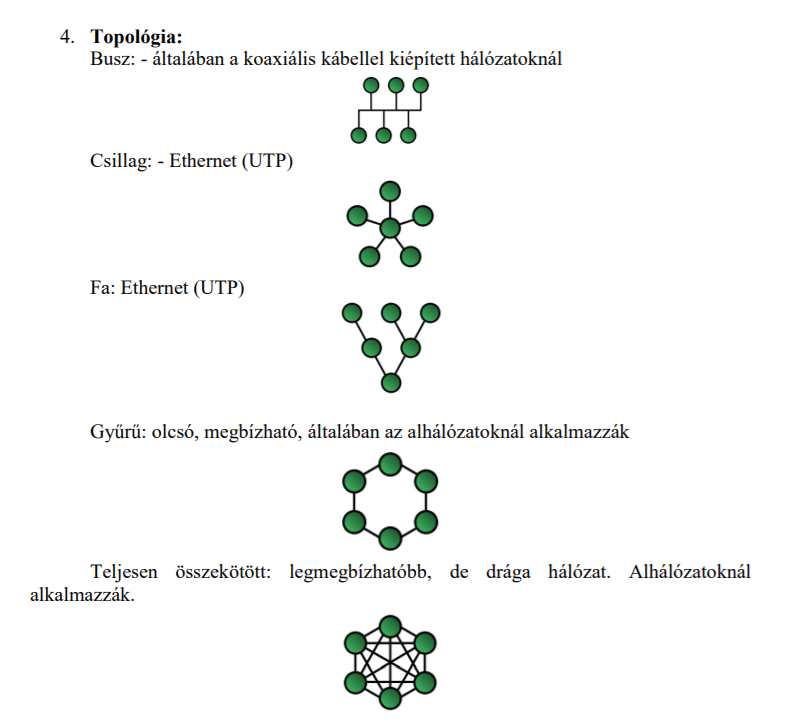
- Kommunikáció szempontjából egyenrangú – kezdeményezheti és befejezheti bármelyik fél a kommunikációt

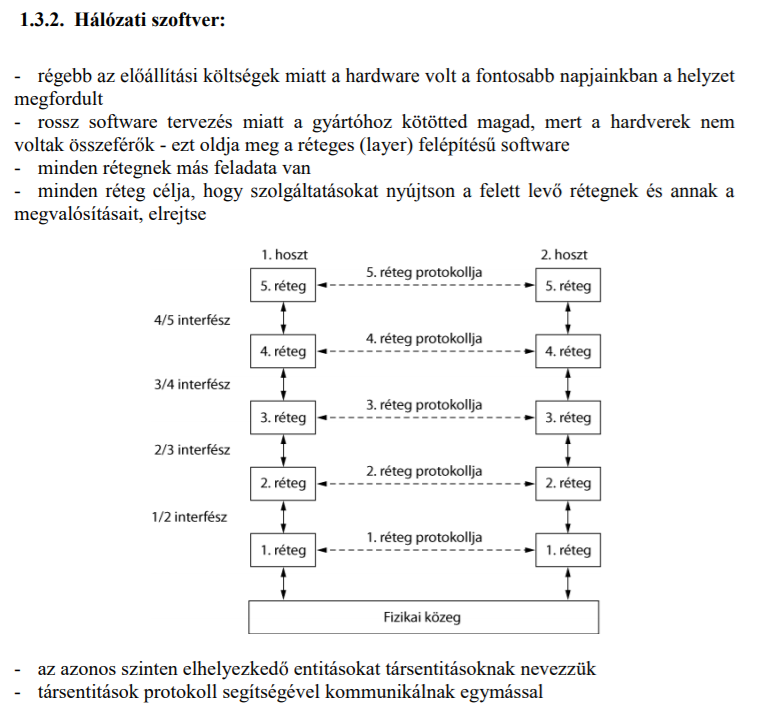
- Autonóm entitás – önálló működésre alkalmas



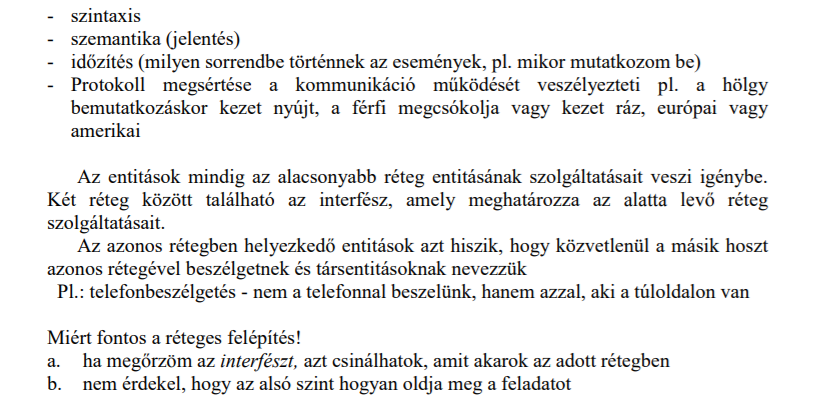




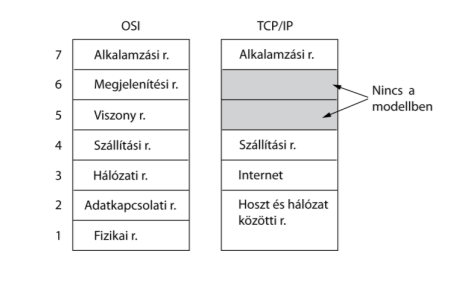




Protokoll:



**Modell rétegei:**



**OSI (Open System Interconection) modell**

1. **Fizikai réteg:**

* Feladata, hogy továbbítsa a biteket a kommunikációs csatornán
* Hozzárendeli a logikai értékekhez a megfelelő feszültség szintet
* Meghatározza a kábel, csatlakozók paramétereit
* Az interfész, mechanikai és elektronikai kérdésekre összpontosít

Összehasonlítási kritériumok: sávszélesség, késleltetés, költségek(telepítés, karbantartás)

Vezeték típusok:

* sodort pár (analóg, digitál jelre is alkalmas), azért sodorják, hogy csökkentsék az elektromágneses hatásokat, teljesítmény-sáv arány jó
* koaxiális kábel (analóg, digitál jelre is alkalmas), jobb árnyékolás->nagyobb távolság
* optikai szálak hatalmas sávszélesség, fény logikai 1, fényhiány logikai 0.

Vezeték nélküli átvitelre elektromágneses hullámot alkalmaznak:

* rádióhullám
  + alacsony frekvencia(egyszerű az előállításuk - áthalad az épület falain - minden irányba terjednek)
  + magas frekvencia(egyenes vonalban terjed, tárgyakról visszaverődik - az eső elnyeli - az elektromos készülékek zavarják - interferencia probléma (nagy távolságokra eljut) )
* mikrohullám(parabola antennával egy pontba fókuszáljuk jel/zaj arány megnő - vevő, adó antennákat jól kell fókuszálni - egymás mellett elhelyezett adó, vevő interferencia nélkül működhet - az optikai szálak előtt ez alkotta a nagy távolságú telefon átvitel alapját - egyenes vonalban terjed → földgörbületi probléma a tornyok 80-100 km-re kell legyenek egymástól. - Nem hatolnak át a falakon)
* infravörös(kis hatótáv)
* látható fény(lézeradó, légmozgás eltéríti)
* műholdak(hatalmas mikrohullámú ismétlő (transzponderek: figyelnek egy spektrumot felerősítik és egy másik frekvencián visszaküldik, hajlított cső üzemmód) - lefelé a nyalábok szélesen terítik a földet). Híresebb rendszerek: Iridium, Globalstar, Teledesic
* Nyílvánosan kapcsolt telefon hálózatok
* Mobil telefon rendszer

1. **Adatkapcsolati réteg**

* Az átvitel a fizikai rétegben nem tökéletes ezért megpróbálja kijavítani
* Hibaérzékelő vagy hibajavító kódokat használ
* Keretezés, nyugtázás a csomag vesztések kezelésére
* A használt egyszerű címzési séma fizikai szintű, azaz a használt címek fizikai címek ([MAC címek](https://hu.wikipedia.org/wiki/MAC_c%C3%ADm)) amelyeket a gyártó fixen állított be hálózati kártya szinten.

**Adatok szállítása:**

1. Nyugtázatlan összeköttetés nélküli szolgálat (ha egy zaj miatt a keret elvesztődik, vagy meghibásodik, nem próbálja meg helyreállítani, vagy újraküldeni a keretet) - megbízható csatornák esetében a hibajavítás a felsőbb rétegekben van megvalósítva - valósidejű adatok pl. hang, átviteléhez használjuk

2. Nyugtázott összeköttetés nélküli szolgálat (optimalizálás) - minden egyes keret megérkezését nyugtázza a célállomás - létezik az újraküldés lehetősége - megbízhatatlan csatornák esetében hasznos (pl. vezeték nélküli hálózatokban) - ha egy üzenet 20 keretből áll, akkor hatékony a keretenkénti nyugtázás (mindig csak egy keretet kell újraküldeni, nem mind a 20-at)

3. Összeköttetés alapú szolgálat - Első lépés az összeköttetést felépítése – inicializálódnak: számlálók, pufferek, változók - Második lépés adatkeretek küldése – sorszámozottak a keretek ezzel biztosítjuk, hogy csak egyszer érkeznek meg és sorrendben - Utolsó lépés az összekötetés bontása – memória felszabadítása - célja, hogy egy zajos csatornát látszólag tökéletessé tegye a hálózati réteg számára

**Keretezés:**

Az átvitt bitsorozat hibamentességét a fizikai réteg nem szavatolja! Lehetséges hibák: - bitek száma - értéke Hibák kezelése: - jelezni - javítani, ha lehet Megoldás - keretekre való tördelés + ellenőrző összeg hozzáfűzése.

**Hibakezelés:**

- A keretek helyes sorrendbe érkeztek-e - A keretek megérkeznek-e - Biztonságos átvitelhez szükséges a visszacsatolás - Ha a csomag valamilyen formában megérkezik pozitív nyugta minden rendben negatív nyugta – újraküldés - időzítő – ha a csomag nem érkezik meg akkor - újraküldés - sorszámozás – ha a csomag késve érkezik meg – akkor sorrendbe állítás

**Forgalomszabályozás:**

- Elárasztás – az adó gyorsabban próbálja küldeni az adatokat, mint ahogy a vevő venni tudja - Gyors adó - lassú vevő esetében még a tökéletes átviteli réteg mellett is elvesztődik a csomag Visszacsatolás alapú forgalomszabályozás - vevő tájékoztatásokat küld saját állapotáról, vagy engedélyt ad a további küldésre (szabvány keretet küldhet) Sebesség alapú forgalomszabályzás - a protokollba be van építve a sebességkorlát

**Hibajavítás:**

1. Annyi redundáns bitet csatolunk a kerethez, hogy belőle meg tudjuk mondani a helyes keretet - hibajavító kódok (vezeték nélküli közegeknél) gyenge minőségű átvitel esetén fontos, amikor az újraküldött keret meghibásodásának is nagy az esélye vagy a csatorna egyirányú

2. Csak hibajelzésre elegendő redundáns bitet fűzünk a kerethez – hiba esetén újraküldjük a keretet – hibajelző kódok (optikai) megbízható csatornák

Módszerek:

Hamming kód-csak 1 bittes hibák javítására alkalmas

Sokkal hatékonyabb az újraküldés (kevesebb adatforgalom egy keret újraküldése, mintha minden egyes kerettel redundáns biteket is küldenénk)

Bővebben: <https://ms.sapientia.ro/~vajdat/education/computernetworks/halozatjegyzet.pdf> 37. oldal

Protokollok:

* Korlátozás nélküli szimplex protokoll(- nincs semmi megkötés - nincsenek hibák - vevő és az adó is megfelelően gyors)
* Szimplex megáll és vár(- Megakadályozzuk, hogy az adó gyorsabban adjon, mint ahogy a vevő venni tudja - szükséges a maximális idő, amely a keret feldolgozásához kell – küldő oldalán beiktatódik egy várakozás – a visszajelzésre, amit a vevő küld)
* Szimplex zajos csatornára(- a csatorna hibázhat – keret megsérül vagy teljesen, elvesztődik - időzítő az elveszet keretek kezelésére vagy a hibás keretek újraküldésére - csak a hibátlanul megérkezett keretekre küldjünk nyugtát. - Nyugta elvesztődése!- egy keret többször érkezik meg- nem megengedhető - Számlálót a fejlécben- hossza 1 bit- nem küldök új csomagot, amíg 1- meg nem győződtem, hogy megkapta- modulo 2-vel való osztással tudom, hogy új csomag vagy a régi és el kell dobjam)
* Csúszóablakos protokoll(- adatkeret csak egy irányba halad - 2 áramkör mind a kettő szimplex - ugyanazon a csatornán 2 irányban (duplex) - egy mező a fejlécben megmondja, hogy adat-e vagy nyugta - nyugtát hozzácsatoljuk a kimenő kerethez, - nyugta ingyen utazik (ráültetés))
* Egybites csúszóablakos protokoll(már 1 nagyságú ablaka lehet - áll és vár technikát alkalmazza - Gond mikor egyszerre küldik a csomagokat – vagy az időzítő időtartama túl rövid)
* n visszalépés(Ha a küldő ablak nagyobb, mint 1 és az n.-ik csomag meghibásodik akkor az összes csomagot, amely a hibás csomag után érkezett a vevő újraküldi)
* szelektív ismétlés(Hasonlít az előző protokollra, de ebben az esetben csak a hibás csomagokat küldi újra az adó)
  1. **Közegelérési alréteg**:
* Az osztott csatornához való hozzáféréssel foglalkozik

A számítógép hálózatokat két nagy csoportra oszthatjuk:

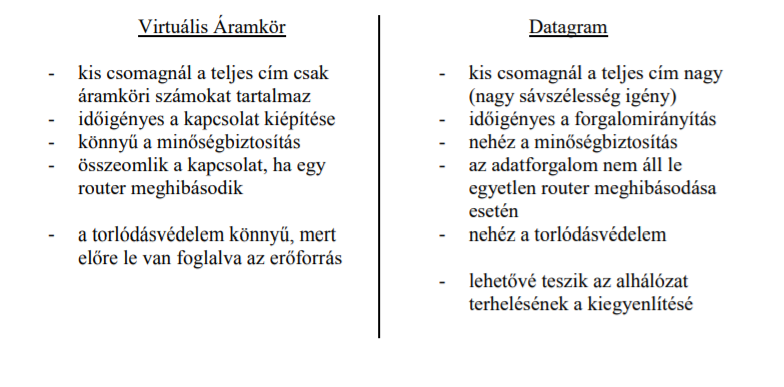
1. Pont-pont hálózatok ebben az esetben a közeget a gépek bármelyik időpillanatban elérhetik anélkül, hogy versengenie kellene ezért. A nagy kiterjedésű hálózatok ilyen típusú felépítést használnak. Pl. a telefonbeszélgetés.
2. Adatszóró hálózatok esetében a gépek egy csatornát használnak közösen. Ahhoz, hogy információt továbbíthassanak a csatornán, versengeniük kell érte. Pl.: egy gyűlés. Az adatszóró hálózatokat csoportosíthatjuk a csatornához való hozzáférés szerint: - többszörös elérésű / hozzáférésű csatorna (multiacces chanel) - véletlen elérésű / hozzáférésű csatorna (random acces chanel)
3. **Hálózati réteg**

* Alhálózat működését irányítja
* Útkeresés:
  + Statikus: táblázatokban
  + Dinamikus: minden csomag új útvonalat keres
* Torlódás (túl sok csomag van a hálózatban) kezelése
* Szolgáltatás minőségének a biztosítása
  + sebesség ingadozás
  + átviteli idő
  + késleltetés
* Különböző hálózatok közti átmenet megoldása
* Adatszóró hálózatokban csökken a szerepe, vagy nem létezik

A hálózati réteg tervezési kérdései:

Szállítási rétegnek nyújtott szolgáltatás(- a szolgálatok függetlenek, kell legyenek az alhálózattól - takarnia kell az alhálózatok számát, típusát és topológiáját - a szállítási réteg rendelkezésére bocsátott hálózati címeknek egységes rendszert kell alkotniuk)

Összeköttetés nélküli szolgálat (internet) (- hibajavítás a felsőbb rétegekben - mindegyik csomag tartalmazza az IP címet - sorrendi kezelésre nincs szükség - forgalom-szabályzás sem kell (Elárasztás))



**Forgalomirányító algoritmusok:** egy hálózati software azon részét, amely azért a döntésért felel, hogy egy bejövő csomag melyik kimeneten távozzon, forgalomirányító algoritmusnak nevezzük - a hálózati réteg feladata: a célig eljuttatni a csomagokat - az összeköttetés nélküli szolgáltatásnál minden egyes csomag esetében alkalmaznak ilyen algoritmusokat.Az összeköttetés alapú szolgáltatásnál pedig az új áramkör kiépítésekor viszonyforgalomirányítás történik (session routing)

Tipusok:

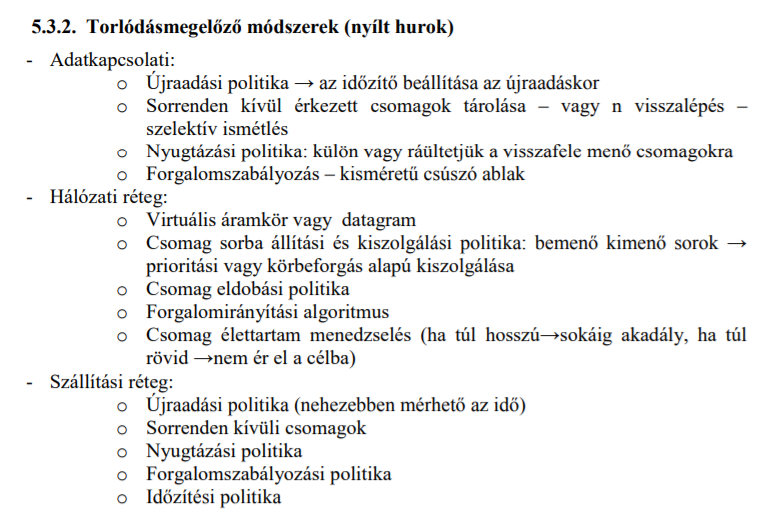
* Az optimalitási elv(Ha A-tól C be vezető legrövidebb út átvisz a B-n akkor a B-től C-be vezető út is optimális)
* Legrövidebb útvonal alapú forgalomirányítás:súlyozott élű gráf → statikus forgalomirányítás (távolság, ugrások száma, sávszélesség, késleltetés) - óránként próbafutással lehet mérni az élek költségét (Dijkstra) - egy csomópontot akkor teszek véglegessé, amikor az összes környezetét ismerem - mindig a legrövidebb utat teszem véglegessé - mindegy, hogy a célból indulok vagy a forrásból
* Elárasztás (floading) - statikus forgalomirányítás – nem veszi figyelembe a hálózat terhelését - minden kimenő vonalon kiküldöm a csomagot, kivétel képez a bejövő vonal - csomagok számának csökkentése – ugrásszámláló - a router sorszámozza a csomagokat, és ha egy visszaérkezik, nem küldi ki még egyszer (forrásonként külön sorszámozza)
* Távolság (vektor) alapú forgalomirányítás – dinamikus minden routernek egy táblázatot kell karbantartania (vektort) amelyben minden célhoz szerepel a legrövidebb út - a vektort a körülötte levő routerektől kapott vektorokból állítja össze - táv – hoppok száma, késleltetés, sorba álló csomagok száma lehet a mérték
* Kapcsolatállapotú forgalomirányítás: 1. Felkutatni a szomszédjait és megtudni a hálózati címeket 2. Megmérni a késleltetéseket minden szomszédjáig 3. Összeállítani egy csomagot, amely tartalmazza az aktuális információkat 4. Elküldeni az összes routernek 5. Kiszámítani az összes többi routerhez vezető legrövidebb utat Dijkstra algoritmussal
* Hierarchikus forgalomirányítás hálózat méretével arányosan nőnek a routerek forgalomirányító táblázata → nő a szükséges sávszélesség, az állapotjelentésekhez → routereket tartományokra osztjuk és szintekre - minden router ismeri a saját tartományát + más tartományok fele, de nem ismeri a többi tartomány belső szerkezetét
* Adatszóró forgalomirányítás: egy csomag mindenhova történő egyidejű elküldését adatszórásnak nevezzük - mindenkinek külön csomag (sávpazarló)+ forrás a célcímek listájával - elárasztás (túl sok csomagot generál) - több célú forgalomirányítás → bit térkép → a cél címekkel → egy csomópont megnézi, hogy a cél címek között van-e olyan, amelyiknek az optimális útja rajta 67 keresztül vezet → megfelelő kimenetekre megfelelő címek társaságában továbbküldi → 1 csomag viszi a többit (1 változat) - feszítőfa (spanning tree) → csomagot minden feszítőfához tartozó vonalra kimásolja kivéve amin érkezett (kapcsolatállapot esetén jó (van fa) távolság vektor esetén nincs fa) - vissza irányú továbbítás - ha azon a vonalon jön amelyiken szokott az adatszórás forráshoz küldeni → minden kimenő vonalra - másképp eldobja
* Többes küldés forgalomirányítás: csoportok kialakítása (nem a router feladata) - routerek kell tudjanak hosztok hova tartozásáról 1) minden csoportnak egy fát kell kiépíteni → nagyon sok fa információját kell a routereken tárolni 2) mag. alapú. fa → mindig 1 routernek kell küldeni a csomagot aki ismeri a fát
* Forgalomirányítás mozgó hosztok esetében: Stabil gépek - Vándorló hosztok → egyik helyről a másikra - fizikailag kapcsolódik hozzá mozgó hoszt → állandó kapcsolat. bejelentkezik az idegen ügynökhöz → lakcím + biztonsági információkat küld - idegen ügynök megkeresi a hazait. hazai ügynök ellenőrzi az információkat - idegen ügynök megkapja a nyugtát – regisztrálja a felhasználót - csomagot a hazai ügynök átirányítja az idegen ügynökhöz → hazai ügynök utasítja az adót, hogy az aktuális címre küldje

**Torlódásvédelem:** Minden hálózat rendelkezik egy maximális kapacitással, amelyet a routerek közötti kapcsolatok sávszélessége, a routerek puffer mérete és a számítási kapacitása határoz meg. Abban az esetben, ha a hálózat egy csomópontjában akár rövidebb időintervallumra is akkora mennyiségű adat érkezik, hogy az a szolgálat minőségének a romlásához vezet akkor torlódásról beszélünk. Ebben az esetben nagyszámú csomag kerül eldobásra.

- Torlódás okai: - több bejövő vonalról →1 kimenő vonalra → - sor képződik → memória hiány (több memória nem oldja meg, lejár az időzítő+1 csomag kerül a hálózatba) - lassú processzor - kicsi a vonal sávszélessége - torlódás öngerjesztő → torlódás → csomag eldobódik → előző router újraküldi → még nagyobb torlódás

**Nyílthurok** (nincs visszajelzés) - Olyan szabályok, amelyek megelőzik a torlódást - nem veszik figyelembe, hogy van-e vagy nincs torlódás - üzembe helyezéskor eldöntik, milyen algoritmust használnak (tervezési kérdés)

**Zárthurok** (van visszacsatolás) 1. Figyelem a rendszert, hogy észrevegyem hol és mikor fog torlódás bekövetkezni - torlódás mérése: eldobott csomagok aránya általános csomagok késleltetés általános sávhosszak - ha mind a három növekedik súlyosbodó torlódásnak nevezik 2. Továbbadni az információt, azokra a helyekre ahol be lehet avatkozni: - Csomag küldése a forráshoz (amúgy is terhelt a hálózat) - Minden csomagban 1 bitet fenntartani ilyen célra - Próba csomagok kiküldése a routerek által 3. Módosítani a rendszer működését, hogy helyrehozzuk a problémát - időléptéket gondosan beállítani túl gyors reagálás → ingadozás, és az algoritmus nem stabil túl lassú → nincs haszna



**Dzsitterszabályozás** Hang és kép átvitelekor fontos, hogy a képek vagy hang csomagok egyenletesen érkezzenek - A csomagok megérkezési idejének a szórását (ingadózása) dzsitternek nevezzük - Dzsitter szabályozásának egyik módja, hogy az adatot a routerek tárolják a pufferben és úgy küldik tovább hogy a megfelelő időpontba érkezzenek - A szabályzás másik módja, ha a vevő oldalon pufferelik az adatot – videó anyag vizualizálása - Videokonferencia vagy telekonferencia esetében ez nem alkalmazható.

A **szolgálat minőségét** folyamok igényeinek paraméterei együttesen határozzák meg: - Megbízhatóság - Késleltetés - Dzsitter - Sávszélesség.

Megoldások: túlméretezés, pufferelés, forgalomformálás.

**A lyukas vödör algoritmus :**

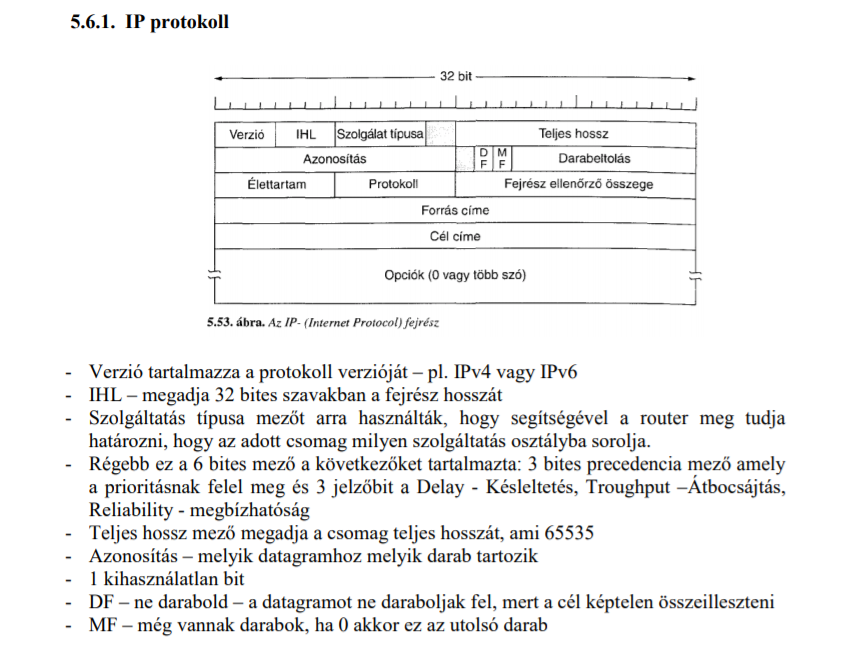
Célja, hogy a hosztok a hálózatba egyenletesen küldjék a csomagokat (bájtokat) és ne löketszerűen ezzel is csökkentve a torlódások kialakulását. - Addig teszem a csomagokat a vödörbe (véges sorba) ameddig az megtelik, ha túlcsordul akkor a többit eldobom. - A vederből egy lyukon keresztül folyamatosan és egyenletesen folyik ki a csomag (ameddig van benne) – a véges sorból a hálózati interfész vagy az operációs rendszer időintervallumonként vesz ki egy csomagot. - Csomagot használunk, ha a csomag mérete konstans pl. ATM, és bájtokat, ha változó a csomagméret. Annyi csomagot veszek ki, ameddig belefér az előre meghatározott bájtmennyiségbe.

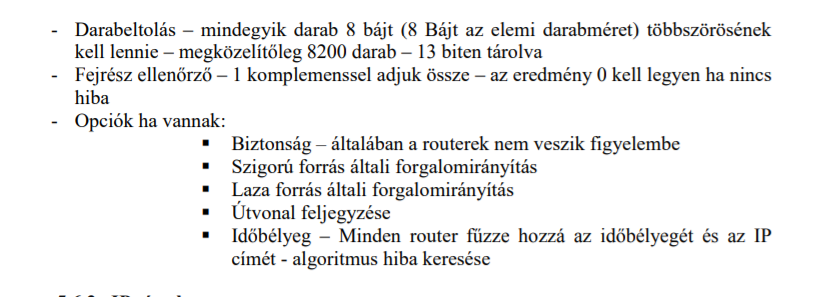
**Vezérjeles vödör algoritmus:** A vezérjeles vödör algoritmus esetében nincsen adatvesztés és a kimeneten megengedett kisebb gyorsulások, amikor adatlöket érkezik. - A lyukas vödör vezérjeleket tartalmaz. Meghatározott időközönként vezérjelet teszünk a vederbe. Ha a veder túlcsordul, akkor a vezérjelek eldobódnak. - Csomagot csak akkor küldhetek, amikor van vezérjel a vödörbe és annyit amennyi vezérjel van a vödörben. - Lehetséges a két algoritmus kombinálása is

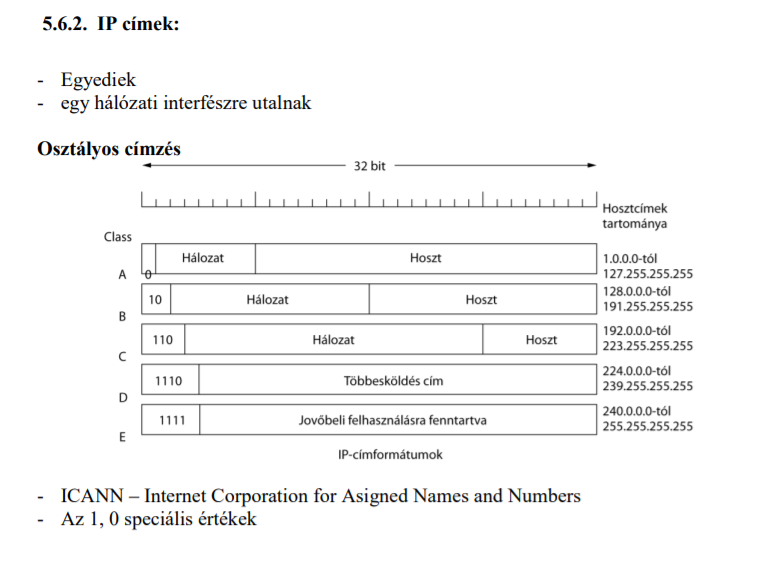
**Erőforrás lefoglalás :** Nehéz a forgalomszabályozást megvalósítani, ha mindegyik csomag más-más úton jut el a célig. A forgalom szabályozásához szükséges hogy minden csomag úgyanazt az utat kövesse –virtuális áramkör féle kapcsolat létrehozása. - Az útvonal mentén erőforrásokat foglalunk le: - Sávszélesség – nem foglaljuk túl a kimeneti vonalat - Puffer terület – lefoglalunk egy bizonyos területet néhány forgalom számára - Processzor idő – csomagok feldolgozás – ennek a lefoglalása a legbonyolultabb

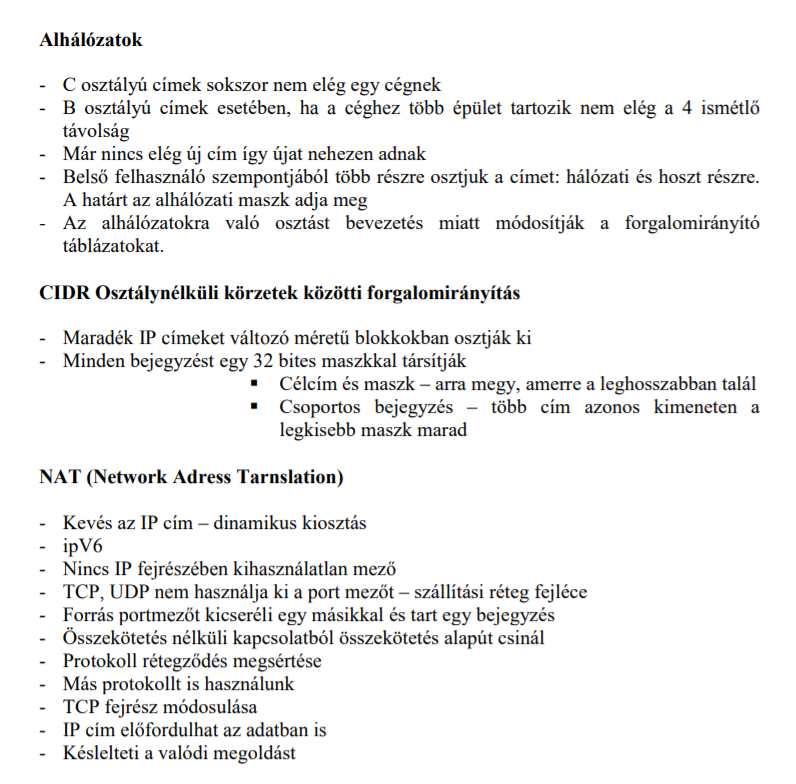
**Belépés engedélyezése** Szükséges az erőforrás lefoglaláshoz, hogy az érintett routerekkel megegyezzünk az igényekről és ebből mit tudnak biztosítani - Általában nem tudják pontos meghatározni a szükséges erőforrásigényeket (puffer, processzoridő) az alkalmazások - Léteznek változtatható igények (30fps helyett 25fps) - Engedélyezési paraméterek halmozását folyam meghatározásnak nevezzük - A forrás állítja elő a folyam meghatározást majd végighalad az forrás cél úton és egyezkedik a routerekkel az igények csak csökkenhetnek az egyezkedés folyamán.

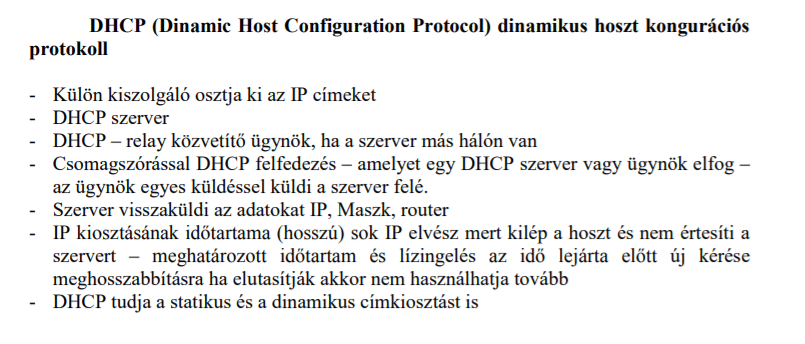
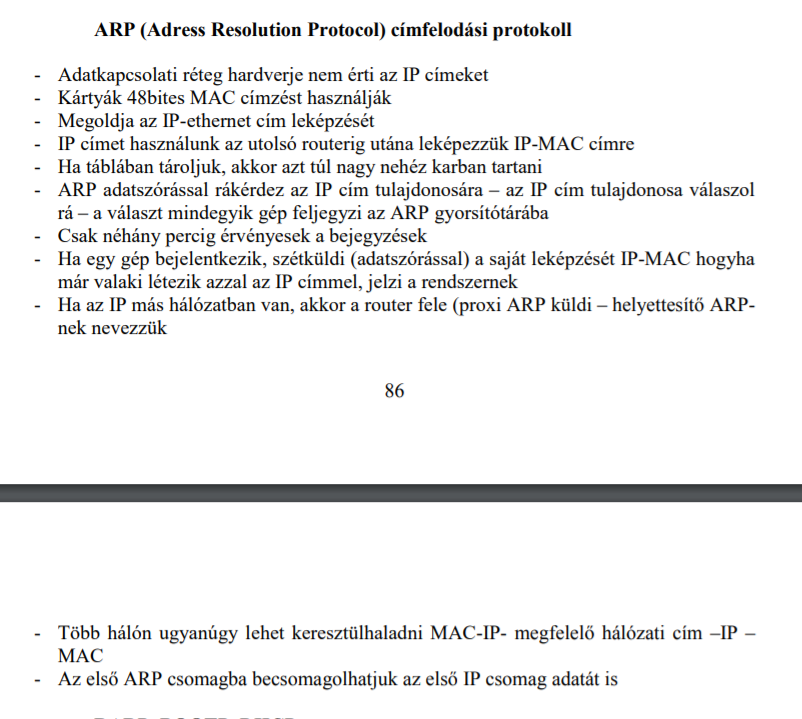
**Csomagütemezés** A routerek több folyamatot kezelnek egyidejűleg egy jól meghatározott ütemezési politika nélkül egy folyamat kisajátíthatja nagy részét amíg a többiek nem vagy csak alig fognak működni. - Egy lehetőség, ha beérkezés sorrendjében küldjük tovább a csomagokat, de ebben az esetben az agresszív adó kedvezményezett lesz. - Egyenlő esélyű sorbaállás ebben az esetben külön várakozó sort rendelünk minden egyes forrásnak és körforgásos alapon minden egyes sorból egyszerre csak egy csomagot veszek ki – a nagyobb csomagok előnyben vannak - Bájtonkénti körforgást szimulálja és a csomag véget érés pillanatában küldi el.

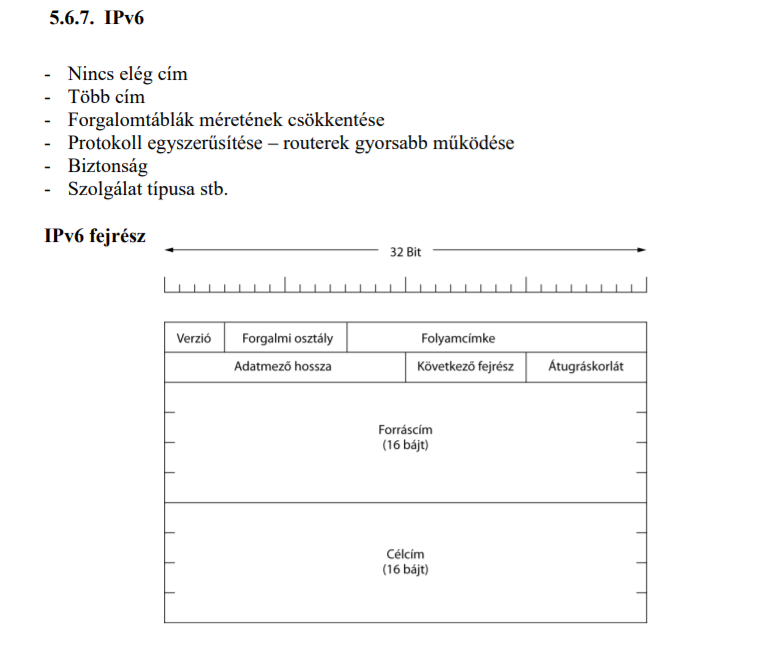












1. **Szállítási réteg**

* Adatokat fogad a viszony rétegből feldarabolja, továbbítja a hálózati rétegnek
* Feladata, hogy az adat hibátlanul megérkezzen a túlsó oldalra vagy jelezve a hibákat
* Itt dől el a szolgáltatások típusa:
  + kapcsolat orientált
  + üzenetalapú -
* Míg az alacsonyabb szintű protokollokat a közvetlen szomszédokkal való kommunikációra használjak a szállítási protokollt a célhoszt és a forráshoszt közötti adatátvitelt szabályozza függetlenül attól, hogy hány alhálózaton van közöttük.

Kéttípusú szolgáltatás van: 1. összekötetés alapú szállítási szolgálat 2. összekötetés nélküli szállítási szolgálat. -

Ezek a szolgáltatások nagyon hasonlítanak a hálózati réteg szolgáltatásaira. Az összekötetés alapú szállítási szolgálat épülhet egy összekötetés nélküli szállítási szolgálatra

Szállítási réteg kódja a felhasználó gépein fut csak, míg a hálózati réteg a routereken is (ebbe nincs beleszólása a felhasználóknak, elromolhatnak) ebből kifolyólag a nagy hasonlóság ellenére sem lehet megvalósítani ugyanazt a megbízhatóságot csak a hálózati réteg segítségével. - Második jelentős szolgáltatása az egységes felületet az alkalmazási réteg fele (függetlenül attól, hogy hogyan van megvalósítva) – ezeket nevezzük szolgáltatási primitíveknek ilyen például a LISTEN, CONNECT, RECV, SEND stb. - A programozók egy szabványos primitív készletre írhatják az alkalmazásaikat függetlenül attól, hogy milyen a hálózati réteg típusa, vagy milyen típusú kapcsolatot használunk. - Az 1-4. rétegeket szállítási szolgáltatónak, míg a 4.-nél magasabb rétegeket szállítási szolgálat felhasználójának nevezzük.

**Szállítási protokollok elemei :**

* címzés(előre ismert porton, folyamat szerver)
* összeköttetés létesítése(Legegyszerűbb a kapcsolat létesítésére egy üzenetváltás lenne - Connect request – connection accept - Viszont a következő két lehetséges hiba következhet be: a csomag elvesztés és kettőzése (Összekötetés létesítése – végrehajtja a feladatot befejezi a kapcsolatot – később érkező kettőzött csomagokat – megismétlik az egészet. pl. Banki tranzakciók esetében katasztrofális eredmény)
* összekötés bontása: (asszimetrikus-váratlanul történik, adatvesztéssel jár. szimemtrikus- 2 független egyenírányú összekötetés
* forgalomszabálozás, pufferelés(Problémái azonosak az adatkapcsolat rétegével - Csúszóablakos módszer alkalmazása – elárasztás ellen lassú vevő gyors adó - Router kevés vonal – hoszt sok kapcsolat – pufferelési módszer hatékonysága nem egyforma – xx több puffer – szállítási rétegben - Küldő puffer minden kimenő TPDU megbízhatatlan háló esetén – háló visszajelzés nem biztos a kézbesítés - Szállítási bejövő puffer tele - Vevő puffer mérete? o Egyforma láncolt puffernek o Különböző méretű láncolt puffer bonyolult o 1 körpuffer – jó kihasználtság, ha terheltek az összekötetések)
* Nyalábolás(- Feltöltési multiplexelés – több száll egy IP - Letöltési multiplexelés – több vonalat használ a sávszélesség növelésére)
* Összeomlás utáni helyreállítás

1. **Viszony réteg**

* két hoszt közötti viszony felépítése a feladata
* párbeszéd kialakítása
* vezérlőjel keresés
* kritikus műveletek végrehajtásához
* szinkronizáció

A réteg lehetővé teszi, hogy két számítógép felhasználói kapcsolatot létesítenek egymással. A viszonyréteg segítségével egy felhasználó állományokat mozgathat számítógépek között. Jellegzetes feladata a logikai kapcsolat felépítése és bontása, párbeszéd szervezése. Szinkronizációs feladatokat is ellát, ellenőrzési pontok beépítésével. Gyakran az együttműködési réteg elnevezéssel is illetik.

1. **Megjelenítési réteg**

* adattípusok közötti konverziót hajtja végre

A megjelenítési réteg felelős az információ formázásáért és eljuttatásáért az alkalmazási rétegnek, ahol a további feldolgozás illetve megjelenítés történik. Gondoskodik róla, hogy az alkalmazási rétegnek már ne kelljen foglalkoznia a végfelhasználói rendszerek esetleg különböző adatértelmezési módszereiből származó szintaktikai eltérésekkel. Megjegyzés: A megjelenítési réteg szolgáltatására jó példa egy EBCDIC kódolású szövegfájl konvertálása ASCII kódolásúvá. A megjelenítési réteg az első, ahol az adatra már nem csak úgy tekintünk, mint egy köteg 0 vagy 1-esre. A titkosítás is tipikusan a megjelenítési réteg feladatai közé tartozik. Igaz, ez elvégezhető az alkalmazási, a viszony, a szállítási vagy a hálózati rétegekben is, azonban előnyeik mellett mindegyiknek megvan a maga hátránya is. Egy másik példa a megjelenítési réteg feladataira a struktúra értelmezése, ami ezen a szinten már standardizált. Leggyakrabban az XML-t használják e célra.

Több széles körben használt alkalmazásban illetve protokollban nem különböztetik meg a megjelenítési és az alkalmazási réteget. Például az alkalmazási réteg protokolljának tekintett HTTP rendelkezik a megjelenítés réteg néhány tulajdonságával. Így képes például felismerni a karakterkódolásokat további konvertálás céljából, amely konvertálás ekkor már az alkalmazási rétegben történik meg.

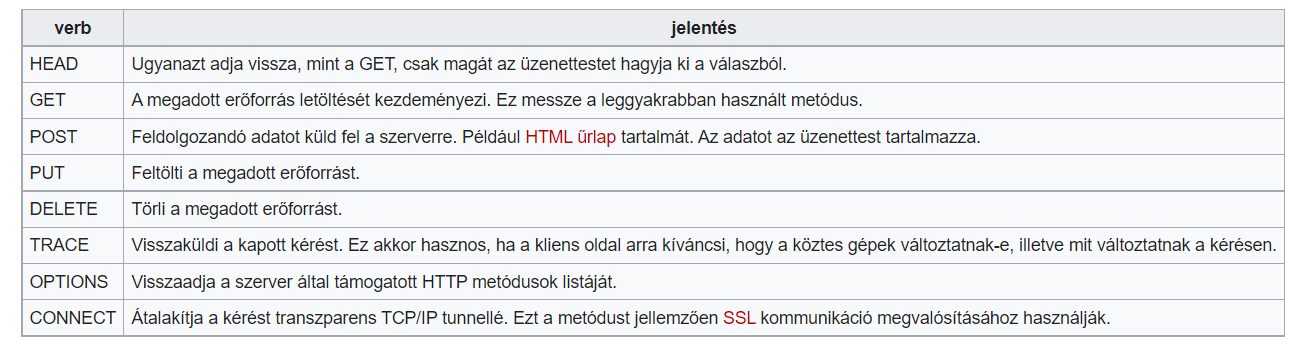
1. **Alkalmazási réteg**

* gyakran használt protokoll sokasága: HTTP, SAP

A felső szintű protokollok kezelését, a megjelenítést, a kódolást és a párbeszédek vezérlését végzi el.

**Néhány protokoll:**

* **HTTP**: A HTTP (HyperText Transfer Protocol) egy információátviteli protokoll elosztott, kollaboratív, hipermédiás, információs rendszerekhez. A HTTP egy kérés-válasz alapú protokoll kliens és szerver között. A HTTP-klienseket a „user agent” gyűjtőnévvel is szokták illetni. A user agent jellemzően, de nem feltétlenül webböngésző. A HTTP a TCP/IP réteg felett helyezkedik el. A HTTP implementálható más megbízható szállítási réteg felett is, akár az interneten, akár más hálózaton. Kizárólagosan TCP protokollt használ, mivel az adatveszteség nem megengedhető.



* **DNS:** A Domain Name System (DNS), azaz a tartománynévrendszer egy hierarchikus, nagymértékben elosztott elnevezési rendszer számítógépek, szolgáltatások, illetve az internetre vagy egy magánhálózatra kötött bármilyen erőforrás számára. A részt vevő entitások számára kiosztott tartománynevekhez (doménekhez) különböző információkat társít. Legfontosabb funkciójaként az emberek számára értelmes tartományneveket a hálózati eszközök számára érthető numerikus azonosítókká „fordítja le”, „oldja fel”, melyek segítségével ezeket az eszközöket meg lehet találni, meg lehet címezni a hálózaton.
* **DHCP**: Ez a protokoll azt oldja meg, hogy a TCP/IP hálózatra csatlakozó hálózati végpontok (például számítógépek) automatikusan megkapják a hálózat használatához szükséges beállításokat. Ilyen szokott lenni például az IP-cím, hálózati maszk, alapértelmezett átjáró stb.

3 féle IP-kiosztás lehetséges DHCP-vel:

* kézi ([MAC-cím](https://hu.wikipedia.org/wiki/MAC-c%C3%ADm) alapján)
* automatikus (DHCP-vel kiadható IP-tartomány megadásával)
* dinamikus (IP-tartomány megadásával, de az IP-címek „újrahasznosításával”)
* **SMTP** Az SMTP a Simple Mail Transfer Protocol rövidítése, ami egy de facto szabvány kommunikációs protokoll az e-mailek Interneten történő továbbítására. Az SMTP egy viszonylag egyszerű, szövegalapú protokoll, ahol egy üzenetnek egy vagy több címzettje is lehet.
* **SSH**: A Secure Shell (röviden: SSH) egy szabványcsalád, és egyben egy protokoll is, amit egy helyi és egy távoli számítógép közötti biztonságos csatorna kiépítésére fejlesztettek ki. Nyilvános kulcsú titkosítást használ a távoli számítógép hitelesítésére, és opcionálisan a távoli számítógép is hitelesítheti a felhasználót. Az SSH-t leggyakrabban arra használják, hogy egy távoli gépre belépjenek vele és parancsokat adjanak ki, de támogatja a tunnelinget, azaz tetszőleges TCP portok és X11 kapcsolatok továbbítását; fájlok biztonságos átvitelére is használható a kapcsolódó SFTP (Secure FTP) és SCP (Secure Copy) protokollok segítségével. Az SSH szerverek alapértelmezésben a 22-es TCP portot figyelik.
* **Telnet**: A Telnet lényege, hogy a saját számítógépéről be tud jelentkezni egy másik (mindegy, hogy a világ melyik részén lévő) számítógépre. Az FTP-vel és a Gopher-rel csak az ott lévő adatokat érte el, Telnet esetében programokat is futtathat a távoli (remote) gépen.
* **BitTorrent**: A BitTorrent, vagy sokszor csak Torrent informatikai protokoll, valamint e protokollt használó p2p alapú fájlcserélő rendszer. A szoftvert Bram Cohen írt kliensek a fájlokat szeletekben (darabokban) töltik le. Minden csomópont megkeresi a hiányzó részhez a lehető leggyorsabb kapcsolatot, miközben ő is letöltésre kínálja fel a már letöltött fájldarabokat. A módszer nagyon jól beválik nagyméretű fájloknál, például videók és nagyobb szoftverek esetében. Ennek az az oka, hogy az ilyen letöltéseknél a szűk keresztmetszetet általában a szerver sávszélessége jelenti. A BitTorrent esetében minél keresettebb egy fájl, annál többen vesznek részt az elosztásában, ezáltal az elosztása gyorsabban megtörténik, mintha mindenki egy központi helyről (szerverről) töltötte volna le. A fájlok darabolásából adódik, hogy a megszakadt letöltések könnyen folytathatóak.
* **FTP**: A File Transfer Protocol, vagy rövid nevén FTP TCP/IP hálózatokon – mint amilyen az internet is – történő állományátvitelre szolgáló szabvány.

Gyakran van szükség arra, hogy valamilyen állományt hálózaton keresztül töltsünk le saját gépünkre, vagy egy állományt mások számára hozzáférhetővé tegyünk. Erre alkalmas az FTP, ami lehetővé teszi a különböző operációs rendszerű gépek között is az információcserét. A világon nagy mennyiségű információforrás áll rendelkezésre, melyek letöltése ilyen módon megvalósítható. A hozzáférési jog alapján kétféle kapcsolattípus létezik: letöltés, vagy feltöltés nyilvánosan hozzáférhető állományokból vagy állományokba, letöltés, vagy feltöltés olyan gépről, ahol azonosítóval rendelkezünk. Azt a folyamatot, amikor egy távoli számítógépről fájlt mentünk a saját számítógépünk háttértárára, letöltésnek nevezzük; feltöltésnek nevezzük, ha a folyamat fordított irányban zajlik, és mi töltünk fájlt mások gépére.

**TCP/IP modell**

**Meg egyeznek a leírások az OSIVAL**

1. Hoszt és hálózat közötti réteg
2. Internet

* az egész architektúrát összefogja - bármilyen hálózatba csomagokat tudjon küldeni - és az célállomásra megérkezzen függetlenül milyen típusú hálózaton kell áthaladjon, és hogy milyen sorrendben érkezik meg IP (INTERNET PROTOKOLL) = hálózati réteg

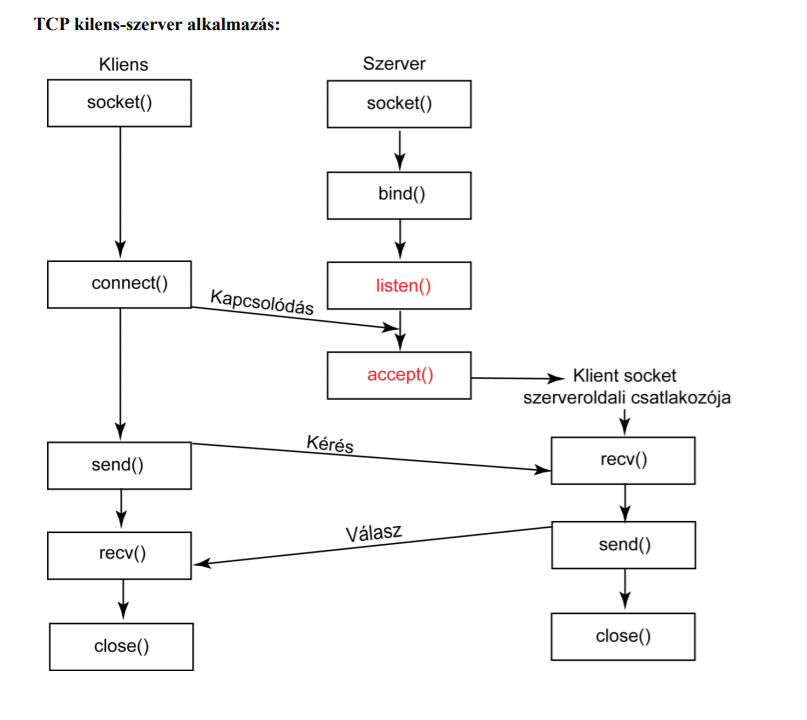
1. Szállítási réteg
2. Alkalmazási réteg

* Az alkalmazási rétegben található az összes magasabb szintű protokoll FTP, HTTP, SNTP, DAS

**2. A TCP protokoll szolgáltatásai és tulajdonságai.**

Összeköttetés alapú kétpontos, duplex protokoll, amely nem képes adatszórásra vagy többesküldésre. Az összekötetés a hosztokon egy-egy csatlakozóban (SOCKET) végződik. A csatlakozók címzési mechanizmusa a következő: az IP címek segítségével megcímezzük a hálózati interfészt (hálózati kártyát), az IP csomag fejrésze tartalmaz egy mezőt, amely megadja a protokoll típusát, es ezután következik egy 16 bites port szám a csatlakozó megcímzésére. Ez a címzési mechanizmus egy hálózaton belül minden egyes sockethez egyedi címet rendel. A TCP bájtfolyamot biztosít a két végpont között. A bájtfolyam a következő tulajdonságokkal rendelkezik:

* rendezett: az adatokat ugyan olyan sorrendben kézbesíti a fogadó hoszt a felhasználó folyamatnak, mint amilyen sorrendbe elküldték
* hibamentes: a protokoll redundáns bitekkel ellenőrzi az elküldött adatok helyességét, valamint a helyesen megérkezet adatokat, nyugtázza. Abban az esetben, ha a kapcsolat megszakad, értesíti a felhasználói folyamatot.
* nincsen adat kettőzés: minden egyes bájthoz egy sorszámot rendel, aminek a segítségével nyomon tudja követni a kettőzött csomagokat.
* A protokoll nem őrzi meg az üzenet határokat. A küldendő adatot küldheti azonnal vagy pufferelheti.



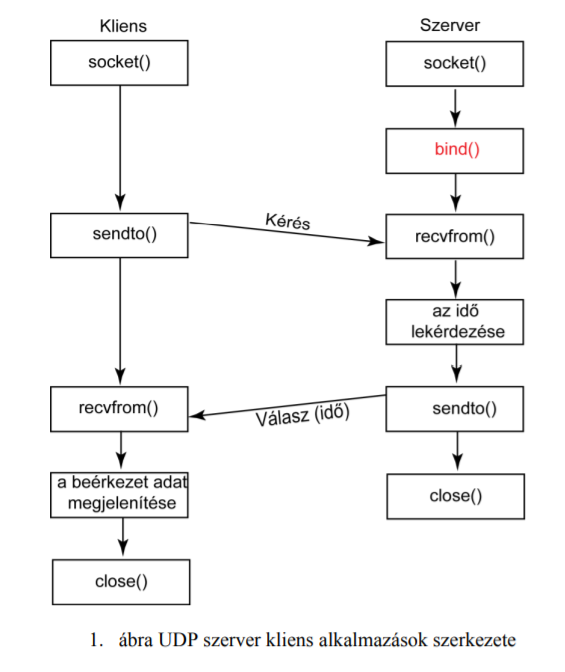
példa:https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/

**3. Az UDP protokoll szolgáltatásai és tulajdonságai.**

A User Datagram Protocol (UDP) az internet egyik alapprotokollja. Feladata datagram alapú szolgáltatás biztosítása, azaz rövid, gyors üzenetek küldése. Jellemzően akkor használják, amikor a gyorsaság fontosabb a megbízhatóságnál, mert az UDP nem garantálja a csomag megérkezését. Ilyen szolgáltatások például a DNS, a valós idejű multimédia átvitelek, vagy a hálózati játékok.

Összekötetés nélküli kommunikációs protokoll azt jelenti, hogy üzenet váltás elött nincs szükség kommunikációs csatorna kiépítésére. Legjobban a levelezésre hasonlít, ahol elégséges ismerni a cél pontos címét. Az üzenetet betehetjük egy “borítékba”, amelyen megadjuk a címzett elérhetőségét. Nem megbízható. Alapértelmezés szerint az UDP protokoll nem rendelkezik arról, hogy a fogadó fél értesítést küldjön az küldemény sikeres megérkezéséről. Ezért nem tudjuk biztosan hogy a csomag megérkezett vagy sem. Vannak viszont olyan megvalósítások, ahol nyugtát küldenek minden egyes megkapott UDP csomagról. Van maximális mérete. Egyszeri küldéssel nem tudunk bármekkora méretű adatot elküldeni. Ezt egyrészt meghatározza az UDP mező hossza (16 bit) valamint az IP csomagok hossza. Az egy csomagban elküldött üzenet egy csomagban érkezik meg. Abban az esetben ha a csomag nem haladja meg az UDP csomag maximális méretét ha megérkezik akkor egyszerre kerül kézbesítésre

Egy UDP vagy TCP alkalmazás megcímzése négy összetevővel rendelkezik:

* MAC cím: fizikai szinten címzi meg a hálózati intefészt. Programozási szemponból nem érint.
* IP cím: hálózati szinten címzi meg a hélózati interfészt (kártyát).
* Protokoll típusa. A hálózati csomag jelen esetben az IP csomag fejrészében egy mezőben van meghatározzva hogy milyen tipusú csomagot szállít az adott csomag
* Port: meghatározza, az alkalmazás kommunikációs csatornáját SOCKET végponttal. Szállítási réteg protokolljainak a fejrészében találjuk meg a portra vonatkozó mezőt.
* 
* példa: https://www.geeksforgeeks.org/udp-server-client-implementation-c/