Párhuzamos és elosztott rendszerek

Készítette: Dr. Mileff Péter

Interprocessz kommunikáció

Interprocessz kommunikáció: hozzáférést biztosít az erőforrásokhoz egy számítógéphálózaton belül olyan különböző politikák és stratégiák segítségével, amelyek meghatározzák a helyi és távoli kommunikációt a processzek között és a processzek és erőforrások között.

Típusok:

a. helyi kommunikáció
   - a processzek azonos gépen vannak

b. távoli kommunikáció
   - a processzek küéönböző gépen vannak

Az interprocessz kommunikáció részei:

a. interprocessz primitívek
b. transzport protokkolok, amelyek támogatják ezen primitíveket

Interprocessz kommunikációs primitívek

a. Üzenetváltás

   - A kliens-szerver modellen alapuló kommunikációs módszer és a centralizált rendszerek interprocessz kommunikációjának egyfajta kiterjesztése

b. Távoli eljáráshívás

   - A módszer egy típus ellenőrző mechanizmus lehetővé teszi azt, hogy egy adott gépen egy programnyelvi szintű hívás automatikusan a megfelelő programnyelvi hívássá alakul egy másik gépen.

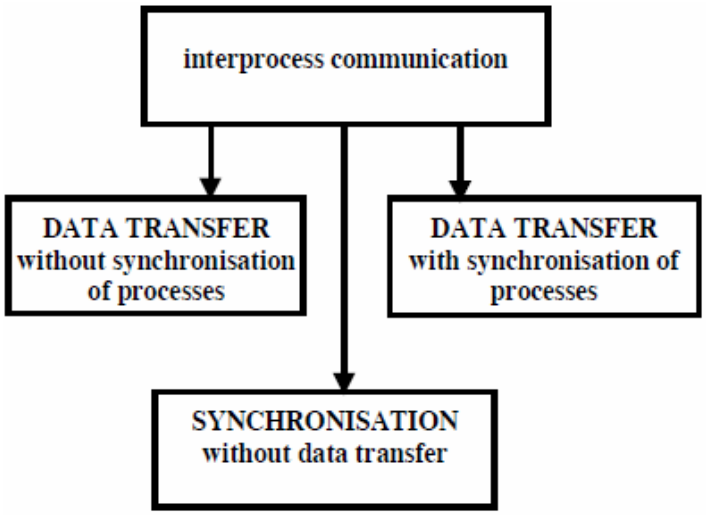
c. Tranzakciók
   - Egy egységes támogatást nyújt az osztott adat-objektumok hívó és szinkronizáló műveleteinek, valamint biztosítja a tranzakciók egymással való szerializációit és atomi viselkedéseit.

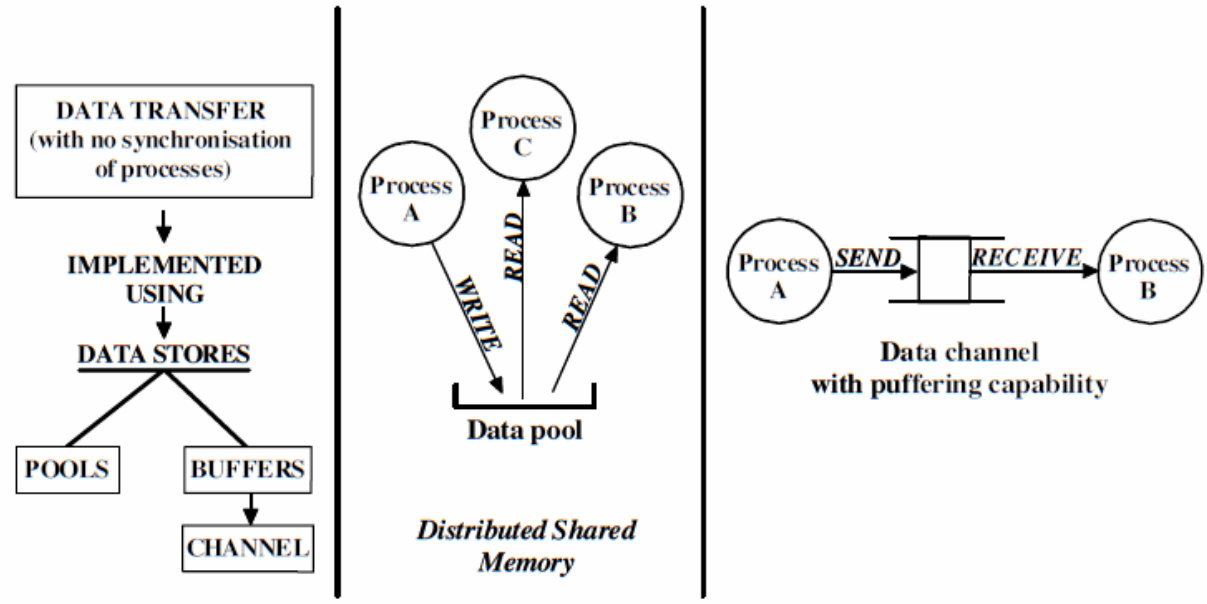
s

Kommunikációs szintek:

interprocess communication	transactions RPC message passing
network operating systems	transport connection
communication network	packet switching

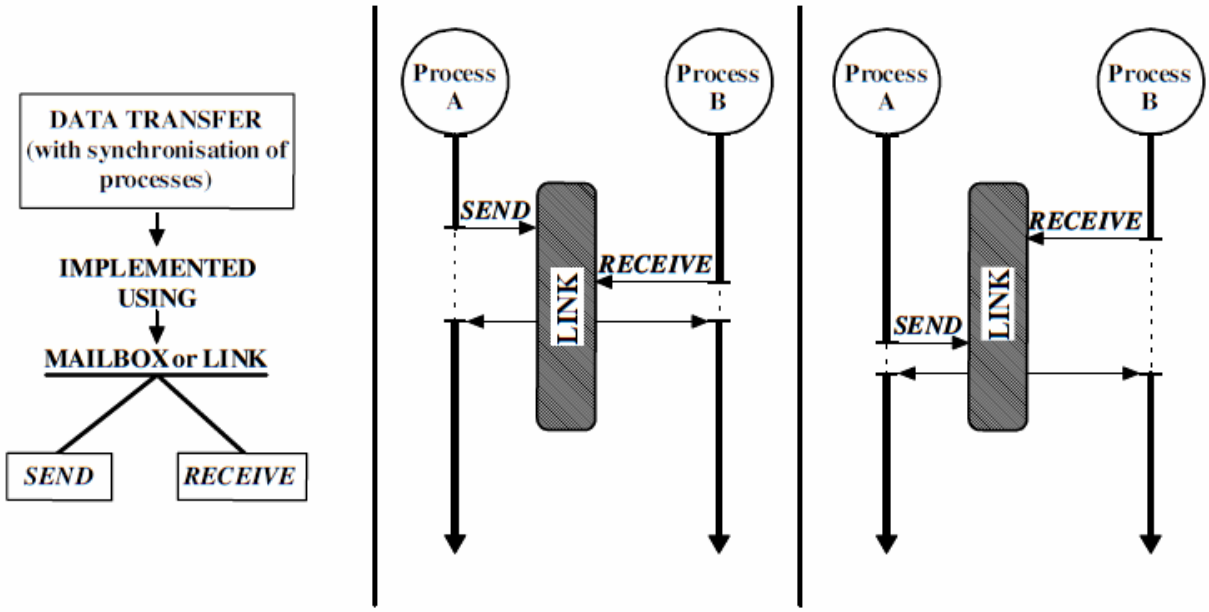
Kommunikációs módok:





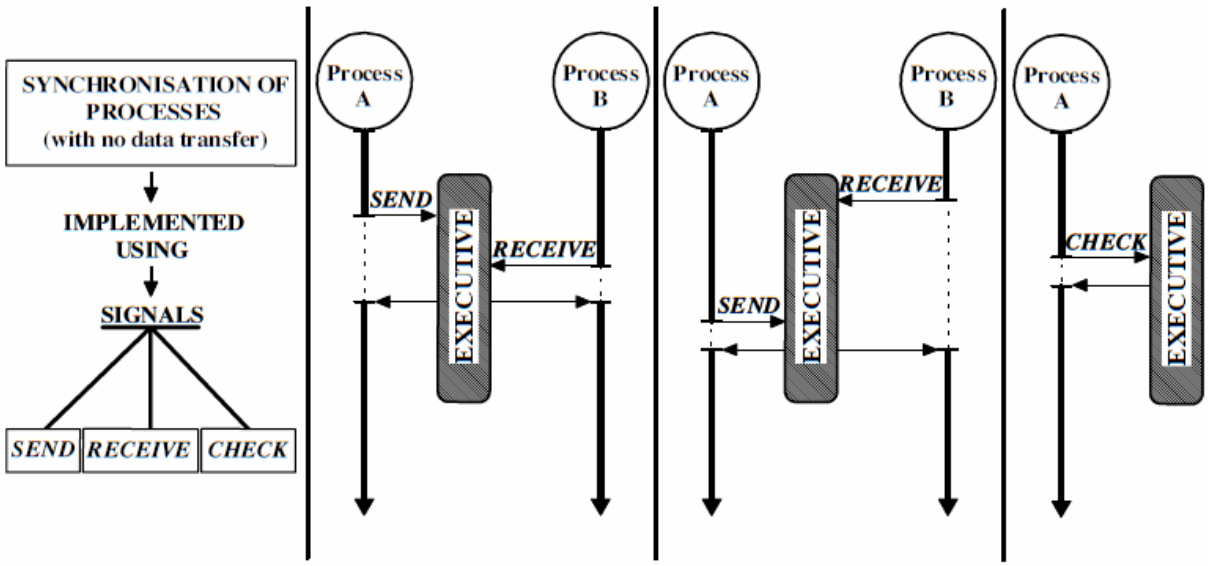
Data transfer without process synchronisation

Processes exchange information either randomly or periodicaly, without any need for synchronization.



Data transfer with process synchronisation

Processes not only wait for events, but also use data associated with those events



Process synchronisation without data transfer, using signals.

Processes are linked by events, not data. They only wait for events.

CSATORNA

CSATORNA: a fizikai kommunikációs hálózat egy olyan absztrakciója, amely kommunikációs „utat” biztosít a processzek között.

- a csatornák a következő két primitív segítségével használhatók
  - küldés
  - fogadás

A kommunikációban biztosított csatornák típusai:

- globális a processzek számára (mint osztott „erőforrás”), vagy lokális, azaz processzenkénti (dedikált).
- egyirányú, vagy több irányú csatornák
- bufferelt vagy nem bufferelt
- megbízható, vagy nem megbízható

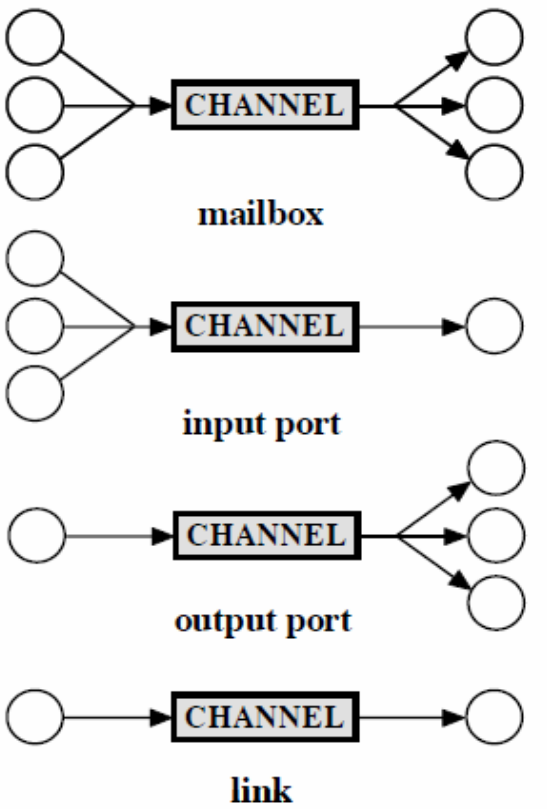
A csatornák elnevezése az alábbi módszerekkel történik:

- direkt vagy indirekt
- statikus vagy dinamikus

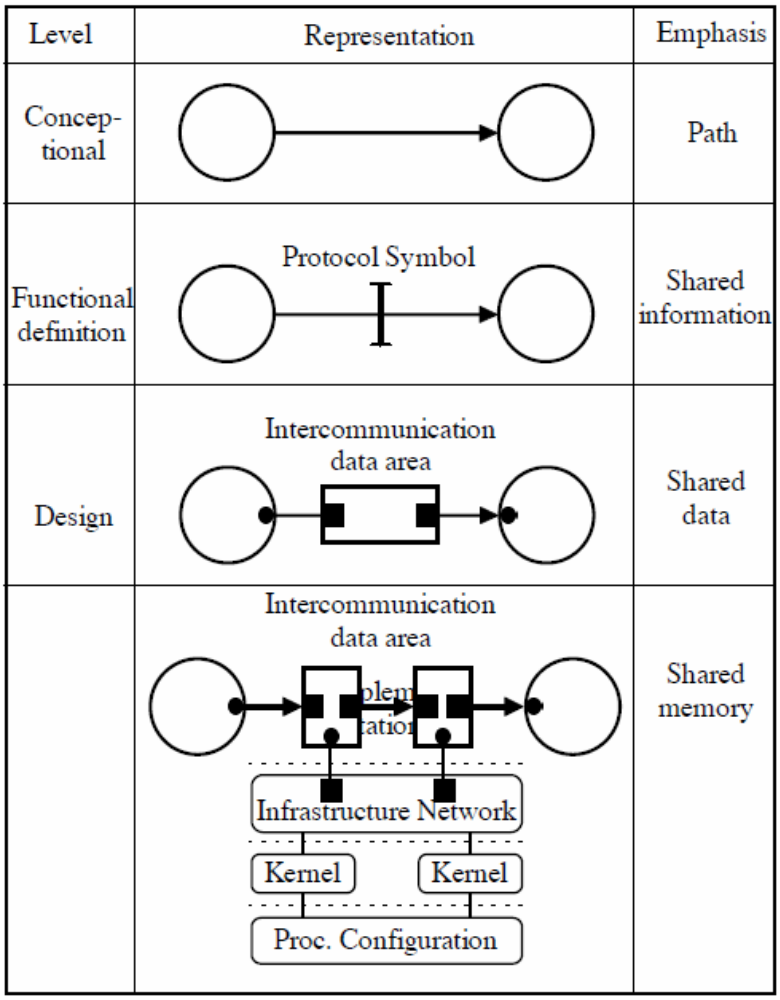
A csatornák megvalósítása a kommunikációban:

- mailbox-ok, vagy portok
- linkek

Az alábbi csatornatípusok bizonyítottan a legnépszerűbb megoldások, mert mindegyik jól alkalmazható valamilyen programozási probléma megoldásában, valamint mindegyik viszonylag hatékonyan megvalósítható:

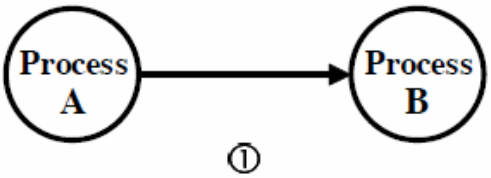

CHANNEL
mailbox
CHANNEL
input port
CHANNEL
output port
CHANNEL
link

Csatorna leírása:



Kommunikációs struktúra

- két, egyazon processzorhoz társított processz közötti kommunikáció direkt iterakció ezen processzek között.

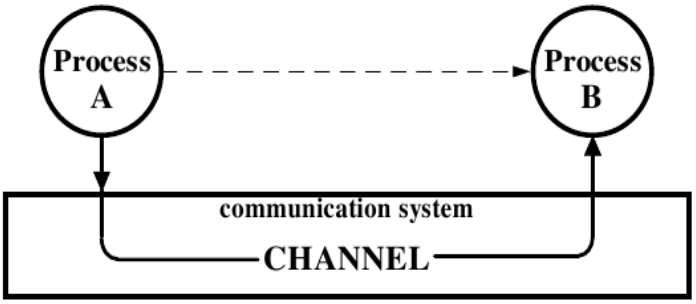


- valamely elosztott rendszerben két processz a kommunikációs rendszer által biztosított valamilyen alrendszeren keresztül kommunikál.

②

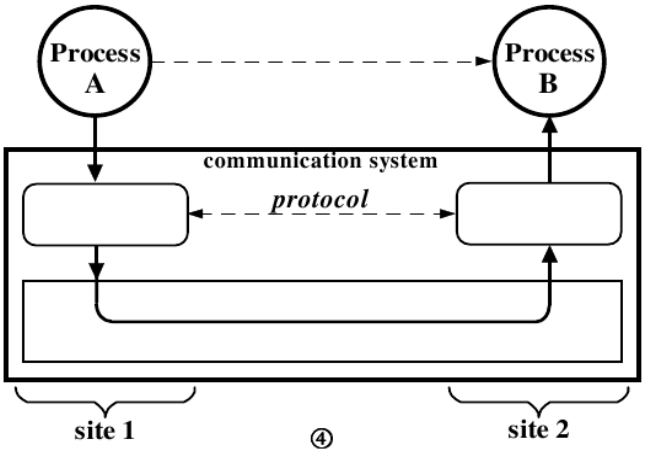
- a kommunikációs rendszer épp annyira transzparens, ahogy a közvetlen a kapcsolat egy processz és a kommunikációs rendszer között.
- Az átlátszóság lehetséges korlátai:
    - átviteli korlátozások
    - várakozás,
    - a rendelkezésre álló kommunikációs primitívek limitációja
    - átviteli hibák
    - elveszett vagy duplikált üzenetek
    - elvesztés vagy üzenetszekvencia
    - komplikál interface-ek a kommunikációs rendszerhez

- A kommunikációs rendszer egy a két processz által használt szolgáltatást biztosít, valamint
- mindkét processz a rendszer architektúrájában a magasabb szinten helyezkedik el.



③

- A kommunikációs rendszer rendszerint hierarchikus, szintekre osztott rendszerként épül fel.



④

Kommunikációs rendszerek az elosztott rendszerek számára

- a számítógépek az elosztott rendszerben egy számítógépes kommunikációs hálózattal vannak összekapcsolva, amely lehet WAN és LAN típusú hálózat is egyben.

WAN: heterogén eszközök között biztosít kapcsolatot egy széles földrajzi területen.
LAN: heterogén eszközök között biztosít kapcsolatot egy kis földrajzi területen.

ISO OSI:

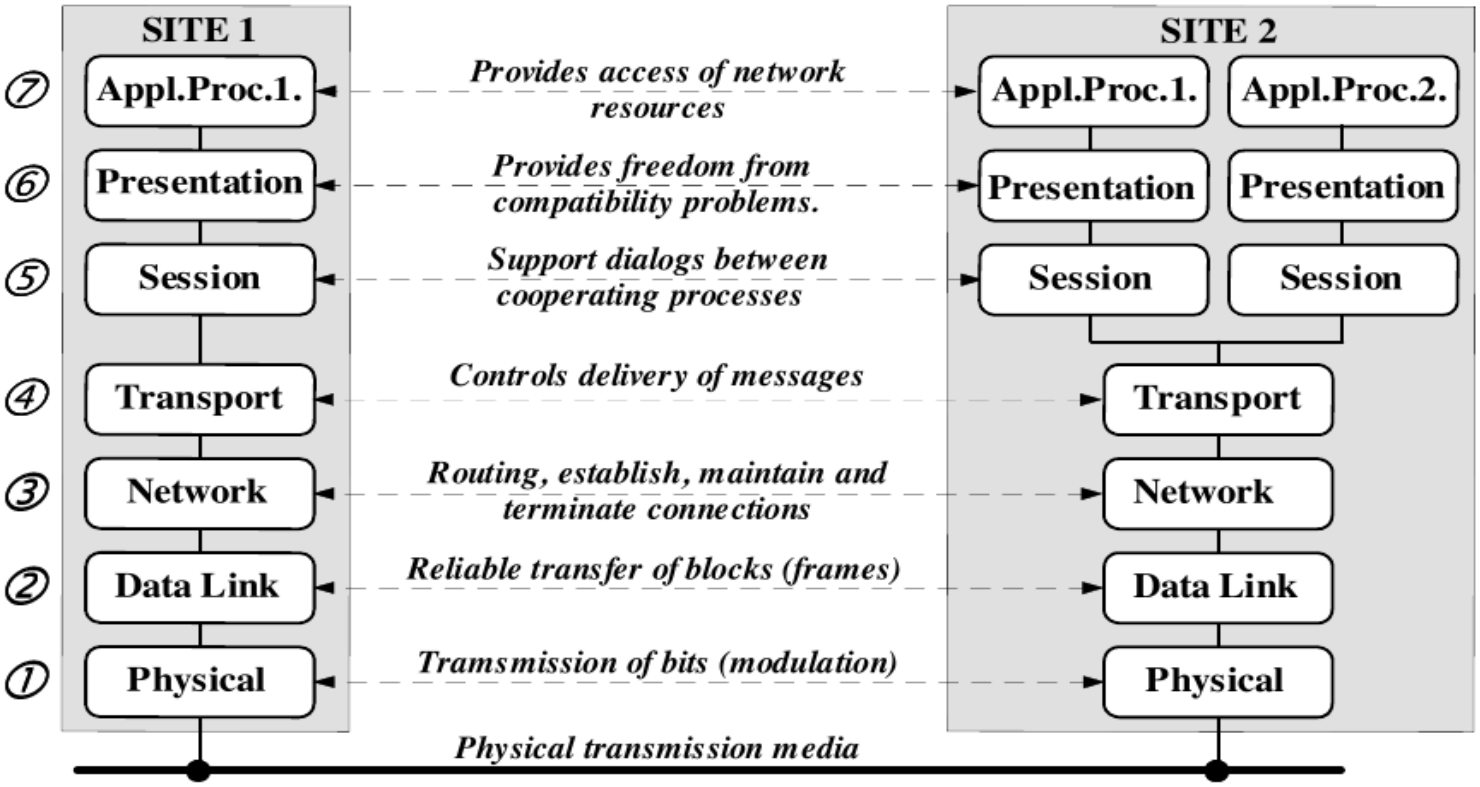
- A kommunikációs protokollok számára egy 7 rétegű keretrendszert biztosít különböző függvényekkel és szolgáltatásokkal.
- A kommunikációs hálózat 3 réteget foglal magába a hét rétegből: a hálózati réteg, az adatkapcsolati réteg és a fizikai réteg.

Szolgáltatások és protokollok

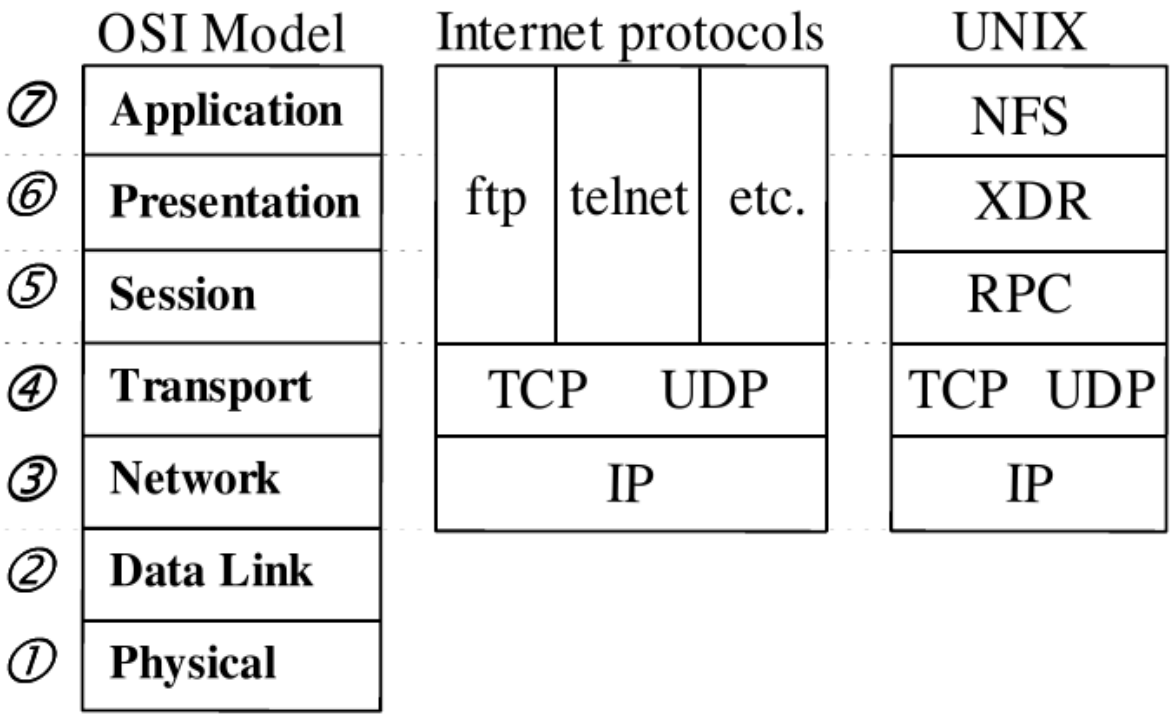
- Az Open System Interface (OSI) számos rétegből áll a hierarchikus struktúrán belül.
- Mindegyik szint két komponensből tevődik össze. Az egyik biztosítja az „n." réteg szolgáltatását az n+1. Rétegnek, a másik pedig a rendszer eggyel alacsonyabb, n-1. szint kommunikációs szolgáltatását használja. Minkét komponens az „n." szint protokolljait követi a kapcsolatban lévő entitások kommunikációinak megvalósításához.

A hierarchikus kommunikációs rendszer szűk keresztmetszetei

- Egy azonos típusú gépekből álló elosztott rendszerben, amelyek nagysebességű helyi hálózattal vannak összekapcsolva, az ISO modell általában nagyon költséges.
- Heterogén számítógépekből álló, lassú kapcsolattal ellátott elosztott rendszerben a hálózati kommunikáció erős limitet jelenthet a szorosan együttműködő számítógépek között. A számítási kapacitás jelen van a komplex protokollok futtatására, azonban ez nem elég a kommunikáció szűk keresztmetszete miatt.



Layering in cooperating open systems



To meet the demand that sites have the mutual ability to transfer files and to invoke remote procedures across diverse networks, DOSes must contain higher level communications protocols, particularly Open System Interconnection (OSI) layers four and five.

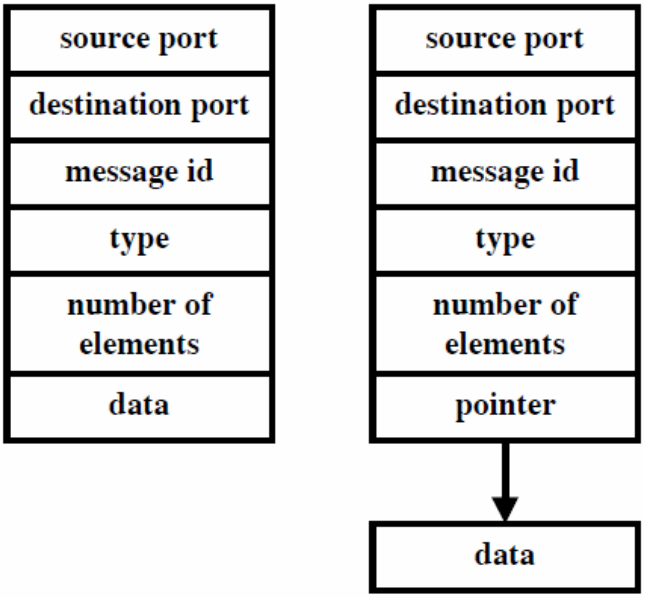
Cross-networking

Interprocessz primitívek: Üzenetküldés (Message Passing)

Az üzenetküldéses kommunikáció során a felhasználó teljes mértékben tisztában van az üzenetekkel és az üzenetek kézbesítésénél és fogadásánál használt módszerekkel.

Üzenetküldés

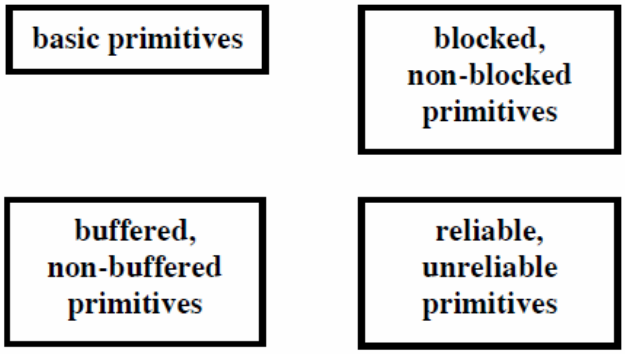
Üzenet: tipizált adatobjektumok együttese, amely egy fix méretű fejléccel és egy változó hosszúságú testtel rendelkezik.



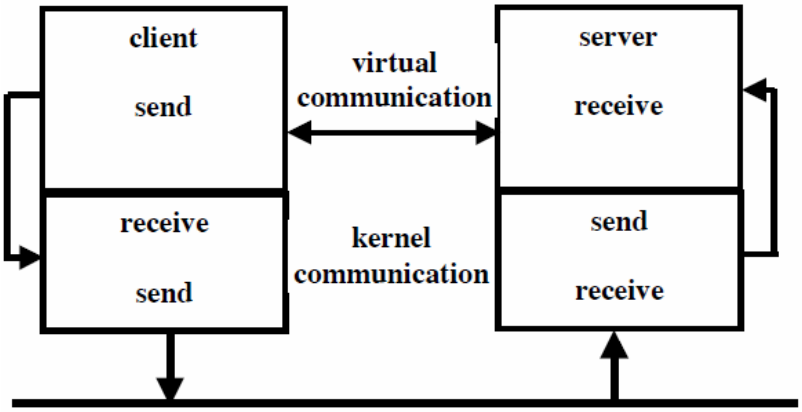
message formats

- a kommunikációban egy processz üzenetet küld a másik processznek és azután várakozik a válaszra.
- Két alapvető problémája van az üzenetküldéses kommunikációnak:
    - Milyen kommunikációs primitívek kerüljenek alkalmazásra?
    - Mi a szemantikája a kommunikációs primitíveknek?

Üzenetküldéses primitívek



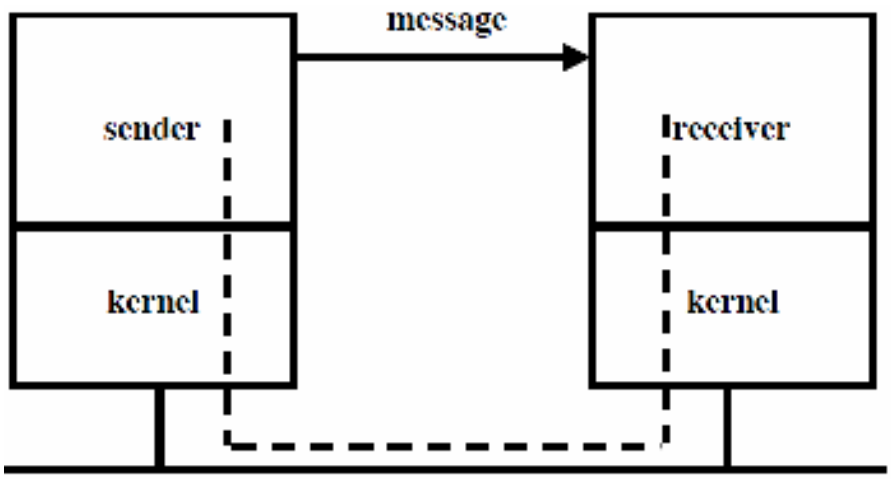
- Az üzenetküldéses kommunikációban az üzenet a feladó processztól a feladó kerneljéhez kerül, ami a kommunikációs hálózathoz továbbítja.
- Ezután a fogadó kernelje fogadja az üzenetet és továbbítja a címzett processzhez.
- Az üzenet eléri a cél kernelt, majd a címzett processzhez továbbítódik.

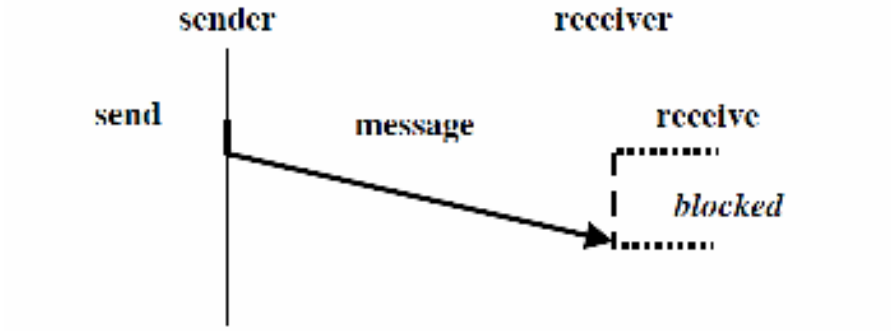


1. Alapvető üzenetküldéses primitívek

küldés(var,did): Az üzenet tartalmazza a változók listáját (var), és a cél címet (did), ahová az üzenetet továbbítani kell. Egy számítógép üzenetet küld egy másik számítógépnek. Pl. a küldő egy üzenetet küld a cél gépnek, amely tartalmaz változókat.

fogadás(var,did): Az üzenet tartalmazza a változók listáját (var), és a forrás gép címét (sid), ahonnan az üzenet jött. Pl. egy számítógép üzenetet fogad egy másik számítógéptől, amelyben változók vannak.





2. **pufferelt és nem pufferelt üzenetküldéses primitívek**

- Némely kommunikációs rendszerekben az üzenetküldés nem pufferelt, némely rendszerekben pedig pufferelt, míg elküldésre és fogadásra kerülnek a kommunikáció gyorsítása érdekében.

A pufferelés három típusa:

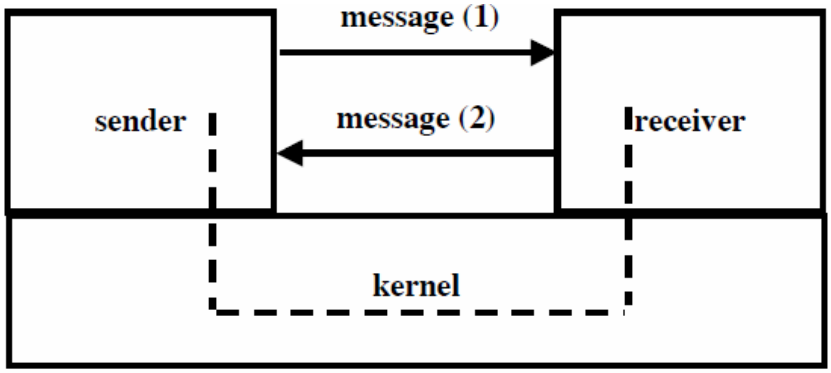
a. **Korlátlan kapacitású puffer:** az üzenetküldő processz sohasem blokkolódik, így a processz messze juthat a futásban az üzenet fogadó processztől. Aszinkron kommunikáció.
b. **Korlátolt kapacitású puffer:** a futó processz végrehajtása a buffer állapotától függ. Pl. tele van a puffer vagy sem.
c. **Zéró kapacitású puffer:** a küldő processz mindig blokkolódik, amíg a fogadó processz meg nem kapja az üzenetet. Szinkron kommunikáció.

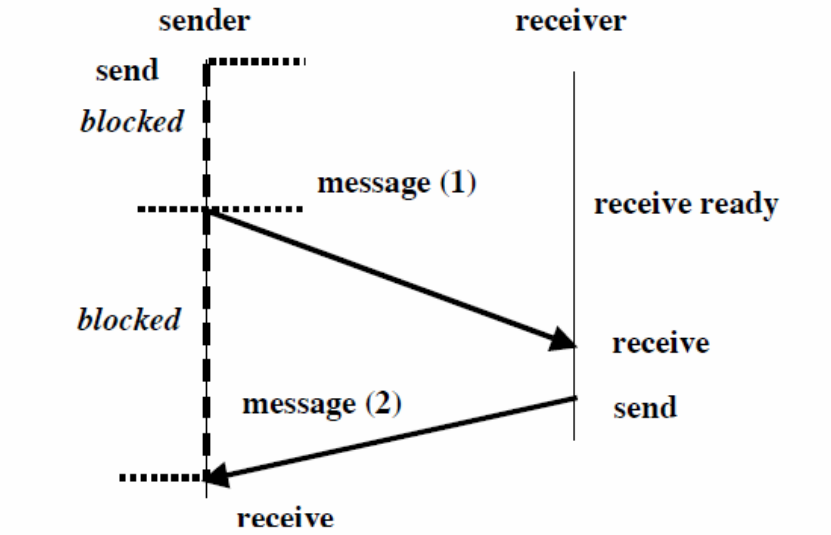
Nem pufferelt primitívek:

- Az üzenet átmásolódik az egyik processz pufferéből közvetlenül a másik processz pufferébe, így a processz egy „küldési primitívet” használ arra, hogy a puffere ne legyen addig újrahasználva, míg a kommunikáció le nem zajlott.
- Egy puffer nélküli rendszerben a processzeket az üzenetküldés miatt szinkronizálni kell.

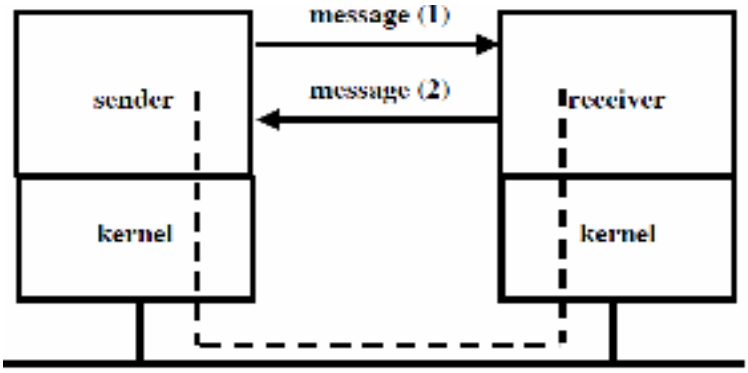
- Emiatt a küldő művelet mindig blokkol, amíg a fogadó művelet nincs kész.
- A nem pufferelt kommunikációnak kétféle típusa van:

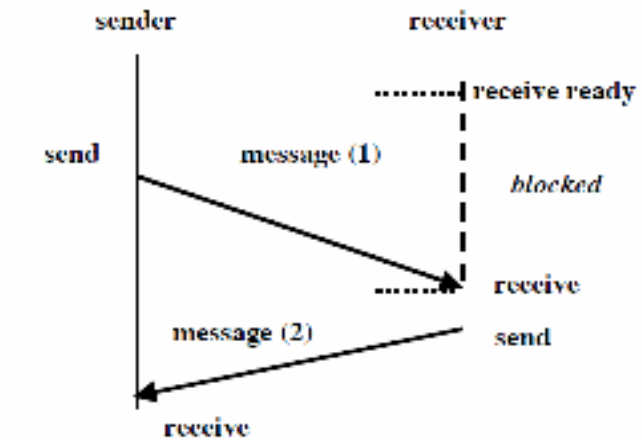
a. Helyi találka
   - Támogatja a kétirányú kommunikációt ugyanazon címtérben elhelyezkedő processz között.
   - Akkor történik, amikor egy processz egy másik processz interface-én keresztül egy belépést kér, és a második processz megadja ezt a belépést a hívásra.
   - A belépésnek lehetnek input és output paraméterei is a küldő vagy fogadó által, valamint a találka kezdetén és végén.





b. Távoli találka
   - A helyi találka kiterjesztése különböző címtérben elhelyezkedő processzek számára.
   - A távoli találka szemantikája megegyezik a helyi találkáéval.
   - A távoli találka megvalósításához a processznek rendelkeznie kell a távoli processz címével és a távoli processz interface-ének másolatával.
   - Ezen információk birtokában a távoli találka végbemenetele azonos a helyi találkáéval.





Nem pufferelt küldés(var,did):

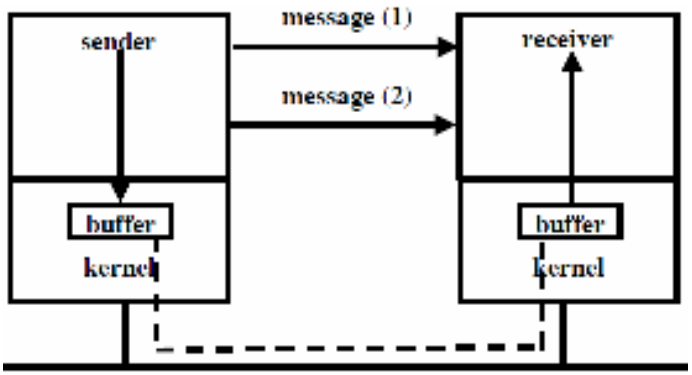
- A küldő processz átmásolja az üzenetet a fogadó processsz belső pufferébe.
- Ha a fogadó processz még nem futtatta a fogadó primitíveket, a küldő processz blokkolódik, amíg a fogadás meg nem történik.
- A fogadó primitívek elvégzése után az üzenet a küldőtől a fogadóhoz másolódik.

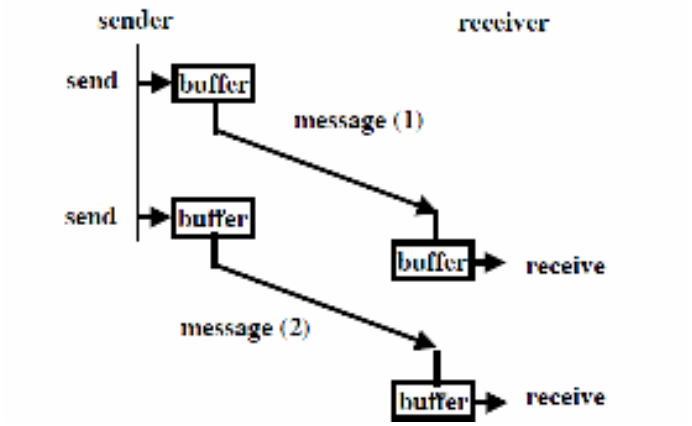
Nem pufferelt fogadás(var,did):

- A nem pufferelt fogadási primitív azonos az alap fogadási primitívvel.

Bufferelt primitívek

- Pufferek vannak, amelyek a vagy a processz vagy pedig a kernel címtérben helyezkednek el.
- Ekkor az üzenet először akár háromszor is másolódhat. A felhasználó puffer éből a kernel pufferbe, másodszor a küldő számítógép kernel puffer éből a fogadó gép kernel pufferébe, végül harmadszor a fogadó gép kernel puffer éből a felhasználó pufferébe.





- A pufferelt üzenetküldés pufferek létrehozását, menedzselését, és megszüntetését igényli.
- A küldő és fogadó aszinkron fut, így az üzeneteket különböző sebességgel generálják és menedzselik. A pufferek kezeli ezt a különbséget.

Bufferelt küldés(var,did):

- a küldési primitív elhelyezi az üzeneteket a pufferbe, ha van hely.
- Ha a puffer tele van, akkor a küldés művelete visszatérhet egy „kóddal" jelezve a puffer telítettségét, vagy várakozik, amíg lesz hely a pufferben.

Bufferelt fogadás(var,did):

- Ha nincs üzenet a pufferben, akkor a fogadó vagy vár, vagy pedig terminálódik egy olyan „kóddal", hogy nincs üzenet a pufferben.
- A primitív kiveszi az üzenetet a pufferből és továbbítja a fogadó processzhez.

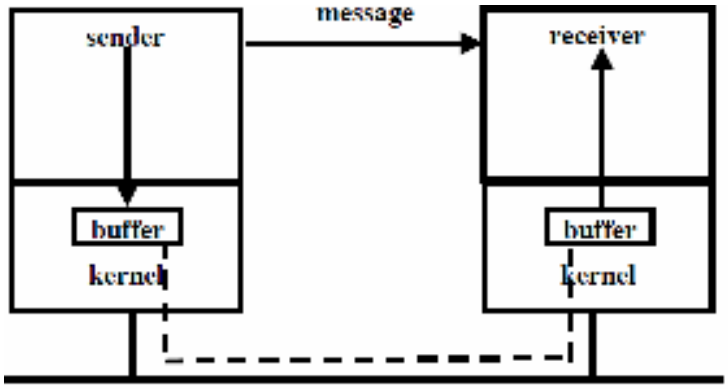
Előnyök: a küldő processz többszörös küldési primitívet valósít meg a puffert felhasználva, így a küldő processz nem várakozik.

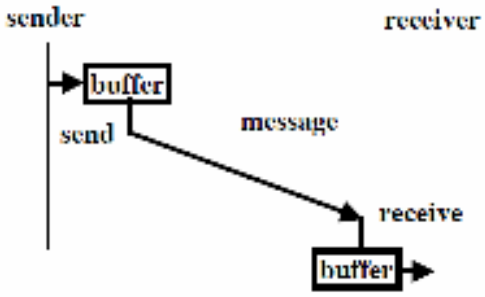
Hátrányok: sokkal komplexebbek, mint a nem puffereltek, mert puffer menedzsmentet igényelnek. Védelmi problémákat generálnak.

3. Blokkoló/nem blokkoló üzenetküldéses primitívek

Nem blokkoló primitívek(aszinkron):

- **c** került, és a küldő folyamat folytathatja a működését az üzenet átvitellel párhuzamosan.
- Miután az üzenet átvitelre került a pufferből, a processz megszakításra kerül, hogy informálódjon arról, hogy a puffer ismét szabad további üzenet fogadására.





Nem blokkoló fogadás(var,sid):

- a küldő primitív jelzi az üzenetfogadás szándékát és biztosít egy puffert az üzenet másolására.
- A fogadó processz vagy folyamatosan nézi, hogy jött-e üzenet, vagy megszakításon keresztül értesül az üzenet érkezéséről.

Előnyök:

- Maximális rugalmasságot biztosítanak a felhasználóknak
- A küldő processz folytathatja a futását az üzenettovábbítással párhuzamosan.

Hátrányok:

- Programozási nehézséget okoznak, főleg valósidejű programozási környezetben.

- Megakadályozzák az üzenethez való hozzáférést, az üzenet megváltoztatását az átvitel előtt és közben.

Blokkoló primitívek(szinkron):

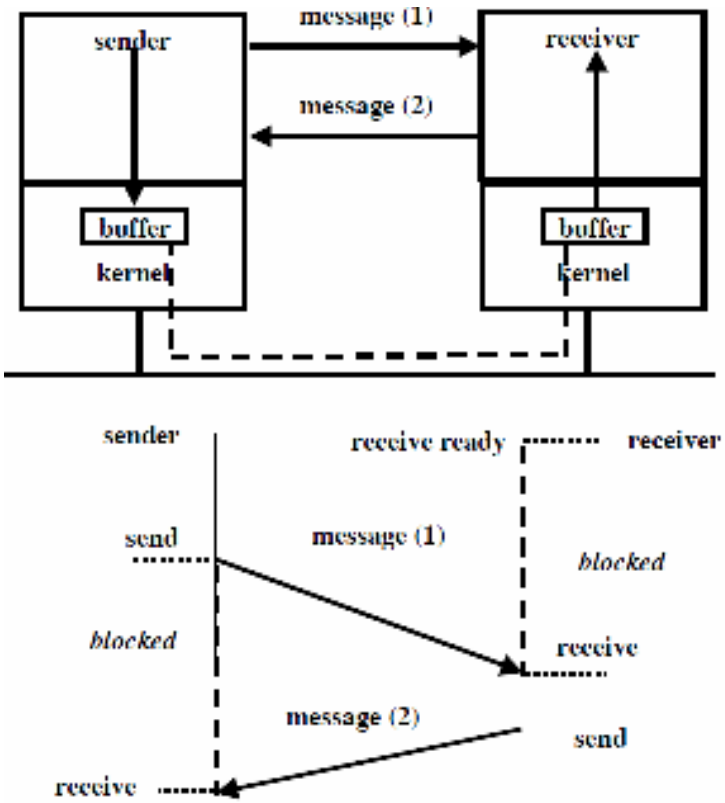
- Kombinálják az adatátvitelt a küldő és fogadó processzek szinkronizálásával.

<u>Blokkoló küldés(var,did):</u>

- A primitív nem adja vissza a vezérlést a felhasználói programnak. Pl. blokkolódik, amíg az üzenet továbbítódik a hálózaton (blokkoló küldés, a processz módosíthatja a puffert), vagy amíg az üzenet kézbesül a célgép kerneljéhez hálózaton (blokkoló küldés, a processz módosíthatja a puffert), vagy amíg egy válasz üzenetet nem kap.

<u>Blokkoló fogadás(var,sid):</u>

- A fogadó primitív nem adja vissza a vezérlést amíg az üzenet be nem kerül a pufferba.
- A fogadó processznem semmi mást nem kell tennie, minthogy várakozik az üzenetre.



Előnyök: a programozást könnyűvé teszik, mert a küldő és fogadó processz közötti kapcsolat jól definiált.

Hátrányok:

- kiiktatják a párhuzamosságot, például a processz blokkolódik a küldésben, és a fogadó művelet nem tud addig hasznos dolgot tenni.
- Nem teszi lehetővé azt, hogy egy processz több eseményre várva blokkolódjon.

- Sok esetben a nem blokkoló küldést és a blokkoló fogadást kombinálják, mert a küldő processz folytathatja a működését miután az üzenet elküldésre került és a küldő kerneljéhez másolódott. A fogadó processznek várakoznia kell az üzenet megérkeztéig.

4. Megbízható és nem megbízható üzenetküldéses primitívek

- A kommunikációs rendszer meghibásodása számos következménnyel és sérüléssel járhat, amelyek menedzselésére megbízhatósági primitíveket alkalmaznak.

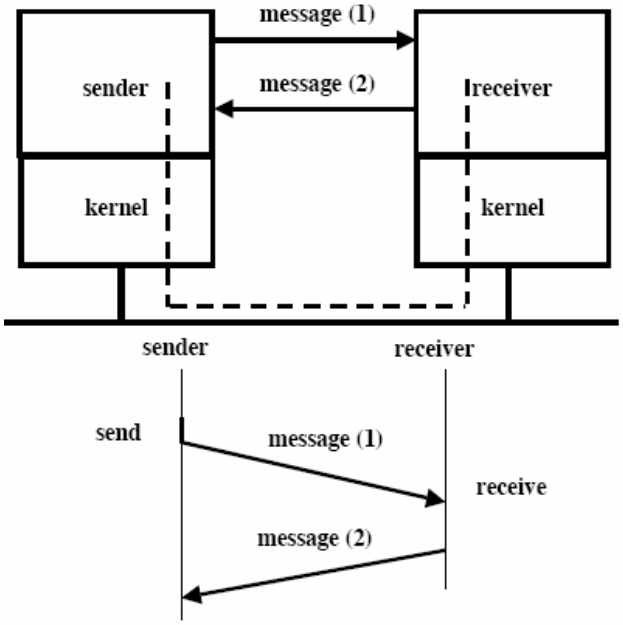
Nem megbízható primitívek:

<u>nem megbítható küldés(var,did)</u>

- a primitív üzenetet küld a hálózatnak, nincs automatikus újraküldés, vagy nem garantált a megérkezés.

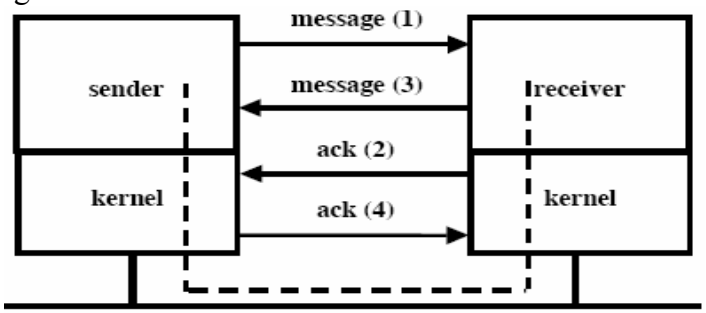
<u>nem megbítható fogadás(var,sid)</u>

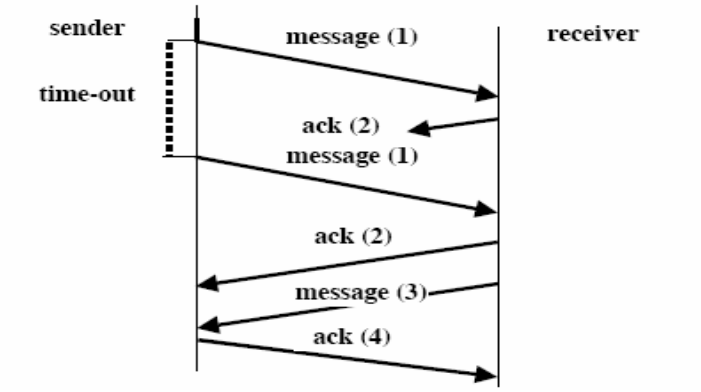
- a fogadó nem küld semmilyen nyugtázást az üzenet megkapásának tényéről.



Megbízható primitívek:

- Amennyiben a küldő igényel valamilyen adatot, a fogadó küld egy választ. A küldőnek pedig fel kell álltani egy fogadó primitívet, hogy tudjon üzenetet fogadni a fogadótól.





<u>megbítható küldés(var,did)</u>

- A primitív menedzseli az elveszett üzeneteket belső újratovábbításokat és nyugtázásokat alkalmazva, amelyek a kernel szinten vannak lekezelve time-out -okat alkalmazva.
- Végeredményképpen, amikor a küldő primitív terminálódik, a küldő bizonyos abban, hogy az üzenet megérkezett és nyugtázva volt.

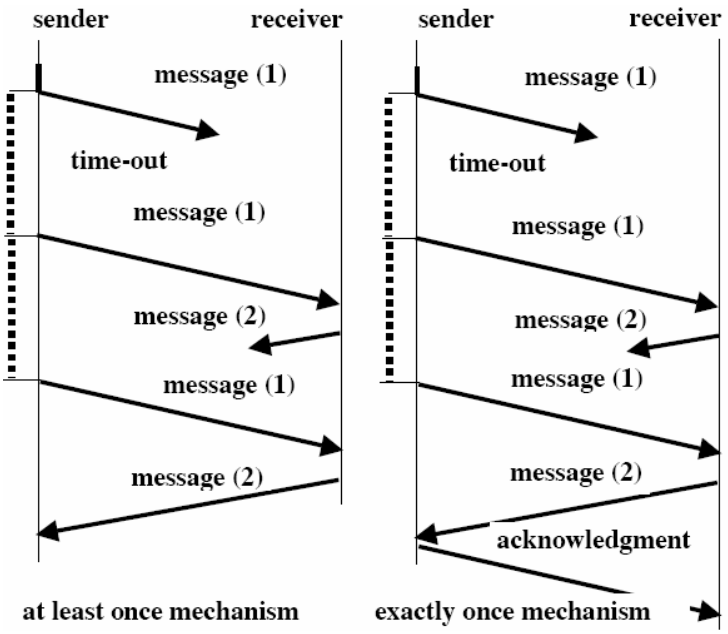
<u>megbítható fogadás(var,sid)</u>

- A primitív automatikusan küld egy nyugtázást az üzenet megérkezéséről.

Előnyök: megvéd a kommunikációs problémák okozta információvesztéstől.

Recovery mechanizmus:

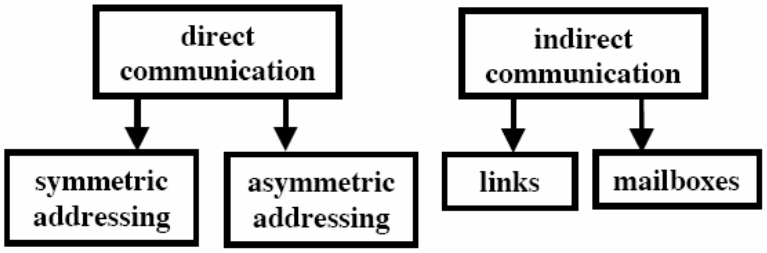
a. *legalább 1 visszaállítási mechanizmus*: megismétli az üzenet feldolgozását még akkor is, ha egy feldolgozás volt megadva.
b. *pontosan 1 visszaállítási mechanizmus*: biztosítja, hogy a fogadó művelete csakis egyszer kerüljön végrehajtásra megvédve az ismételt végrehajtásoktól, amely tönkreteheti az információ konzisztenciáját.



Címzés az üzenetküldésben

Az üzenetküldésben kétféle címzési megközelítést alkalmaznak:

- direkt kommunikáció – a küldő kijelöl egy fix helyű processzt.
- indirekt kommunikáció – a küldő kijelöl egy fix helyet az üzenet kézbesítésére.



1. Direkt kommunikáció

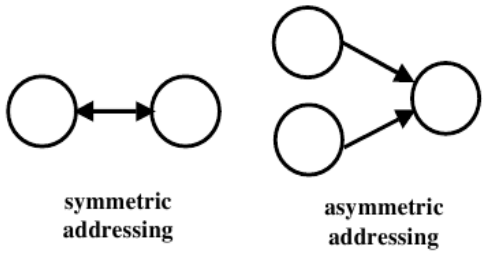
– A kommunikációs útvonal létezik a processzpárok között, amely teljes mértékben transzparens, és az üzenetátvitelben nincs jele a kommunikáció jelzésének.

Szimmetrikus címzés:

- a címzés szimmetrikus, mert mindegyik processz, amely részt vesz a kommunikációban explicite megnevezi a kommunikáció küldőjének és fogadójának nevét.
- A globális processz „ID” tartalmazza a hálózati host címét és a helyi processz „ID”-t.
- Csakis egy kommunikációs útvonal létezik a küldő és fogadó processzek között.

Küldés(proc,mes): a primitív elküldi az üzenetet (mes) egy processznek (proc).

Fogadás(proc,mes): a primitív fogad egy üzenetet (mes) egy processztól (proc).



Aszimmetrikus címzés:

- csak a a küldő processz nevezi meg a fogadó processzt, mert néha szükséges, hogy egy processz ismeretlen processzektől legyen képes üzenetet kapni.
- A címzés aszimmetrikus, mert csak a küldő processznek kell megjelölnie a fogadó processzt.

Küldés(proc,mes): a primitív küld egy üzenetet (mes) egy processznek (proc).

Fogadás(sid,mes): a primitív fogad egy üzenetet (mes) egy „sid”-vel jelzett processsztől.

Előnyök: egyszerű kommunikáció menedzsment.

Hátrányok:

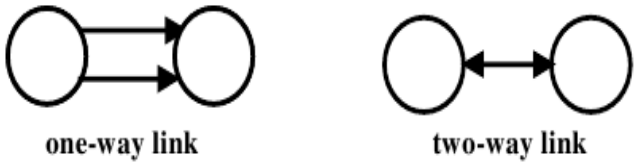
- csak egy kliens és egy szerver közötti kommunikációt tesz lehetővé. Pl. nem teszi lehetővé azt, hogy egy kérést több szervernek lehessen elküldeni.
- Valamely processz nevének megváltoztatása a processz összes régi referenciájának módosítását igényelheti.

2. Indirekt kommunikáció

– A direkt címzés egy kétirányú kommunikációs kapcsolatot biztosít a kommunikáló processzek között. Ezzel ellentétben az indirekt kommunikáció több útvonal azonosítására alkalmas.

Link:

- egyirányú kommunikációs csatorna egy kommunikáció orientált rendszerben, amely védett objektum, a felhasználói, rendszer, vagy kernel processzeknek számára biztosított.
- Egy processzen belül több kommunikációs pont biztosítása több link segítségével.
- Az egyirányú kommunikációs aszimmetrikus, mert először csakis a küldő processznek kell megtartani a linket a fogadó processzhez, másodszor a fogadó processznek kell megtartani a linket a küldő processzhez csakis arra az időperiódusra, ami az üzenet fogadásától a válasz visszaküldéséig eltelik.
- Minden processz egy táblázatban tárolja a hozzá kapcsolódó linkeket. A táblázat minden bejegyzése egy kommunikációs útvonalat tartalmaz valamely processzhez.
- A linken keresztül küldött üzenet egy hálózati útvonalra képződik le, és kézbesítődik a távoli géphez.

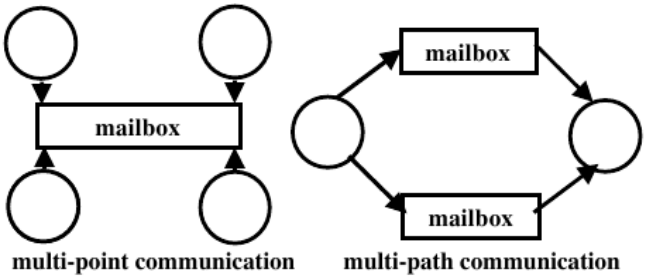


Előnyök: megbízható és sorozatos üzenet kézbesítést biztosít.

Hátrányok: a link menedzsment erőforrásigénye tekintélyes.

Mailbox(port):

- A cél kijelölő (designator) a küldésben és forrás kijelölő a fogadó műveletekben.
- A postafiókok (mailboxes) globális adatstruktúrák, amelyet néhány küldő és fogadó processz oszt meg.
- A postaládák szinkronizációt igényelnek a kommunikációs hálózaton keresztül, amely nehéz feladat.



- A portok speciális postaládák, amelyek védett kernel objektumok. Processzek üzenetek helyezhetnek el bennük, és az üzenetek ki is vehetők belőle.
- Minden portnak véges FIFO tárolója van.
- Üzenetek hozzáfűzhetők és eltávolíthatók a tárolóból küldő és fogadó műveletek segítségével egy kommunikációs útvonalon keresztül.
- A portok a linkekhez hasonlóak, kivéve azt, hogy kétirányúak és bufferelt.

- A processzek különböző portokat használva kommunikálhatnak más processzekkel.

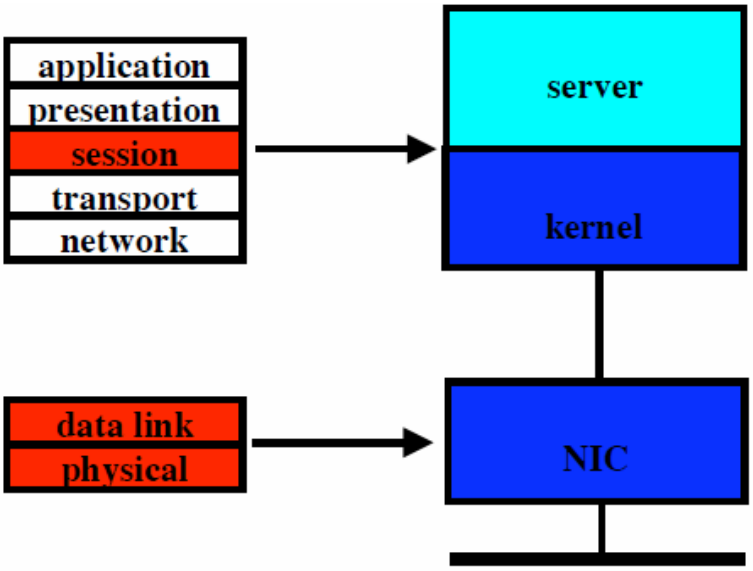
Küldés(port,mes): a primitív küld egy üzenetet (mes) egy portra (port).

Fogadás(port,mes): a primitív fogad egy üzenetet (mes) egy portról (port).

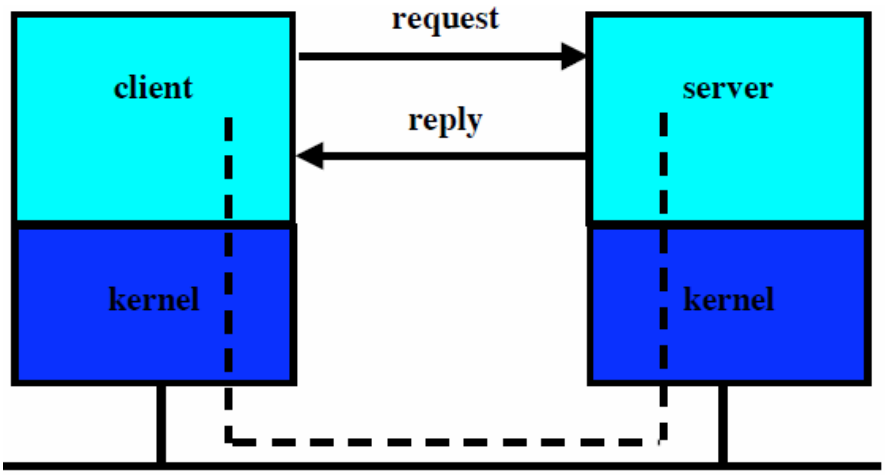
Kliens-szerver kommunikáció

szerverek: együttműködő processzek csoportja, amelyek OS szolgáltatásokat biztosítanak a klienseknek.

kliensek: felhasználói processzek, amelyek szolgáltatásokat igényelnek a szerverektől.

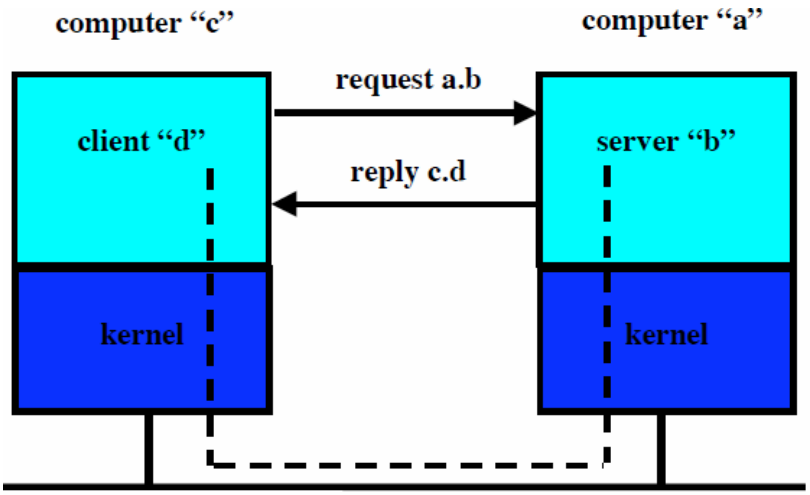


- a kapcsolat orientált protokollok többletráfordítások elkerülése miatt a kliens szerver modell egy egyszerű kapcsolatmentes kérés/válasz protokollon alapul.
- Például a kliens küld egy üzenetet a szervernek valamilyen szolgáltatás kérése miatt, a szerver elvégzi a kért szolgáltatást, és visszaküldi a választ a kliensnek.



Címzések a kliens-szerver kommunikációban

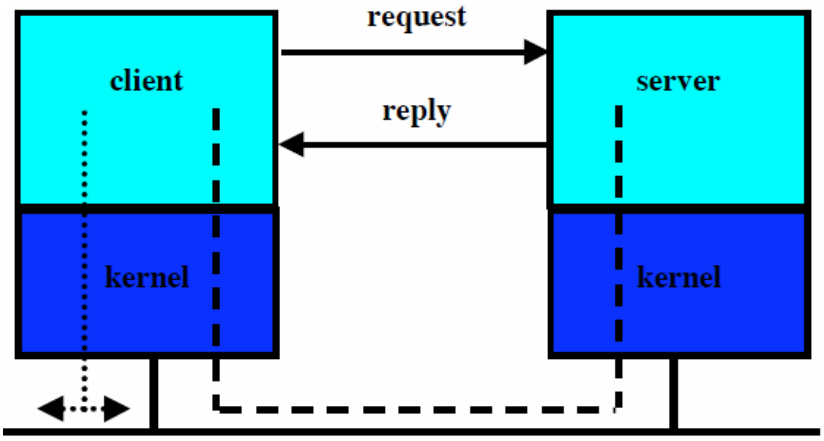
1. **Egyszerű címzés:**
   - A cím két részből áll. Az első rész specifikálja a számítógépet, a második pedig a processzt.
   - A gép száma a kernel számára fontos, hogy kinek kell az üzenetet kézbesíteni. A processz azonosítója pedig azért, hogy azonosítja, hogy melyik processz fogja megkapni az üzenetet.



   - Ez a címzési mód nem transzparens, mert a cél (pl. a gép helye) direktben szerepel a címzésben.
   - Ha a szerver leáll, akkor a direkt címzést használó processzek nem képesek más szerverrel kommunikálni.

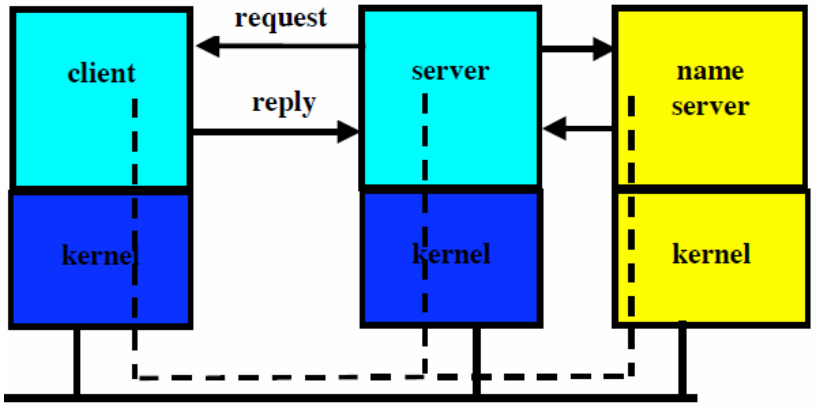
2. **Üzenetszórás alapú címzés:**

- minden processz választ egy azonosítót egy nagy címtartományból, egy 64 bites bináris integer térből.
- A küldő üzenetszórást végez egy speciális „lokalizáló” üzenettel, amely tartalmazza a célprocessz címét.
- Miniden kernel ellenőrzi, hogy a kért cím hozzá tartozik-e, és ha igen, akkor visszaküld egy „itt vagyok” üzenetet a saját hálózati címével együtt.
- A küldő processz kernelje ezt a visszaküldött címet fogja használni a megfelelő szerver eléréséhez.



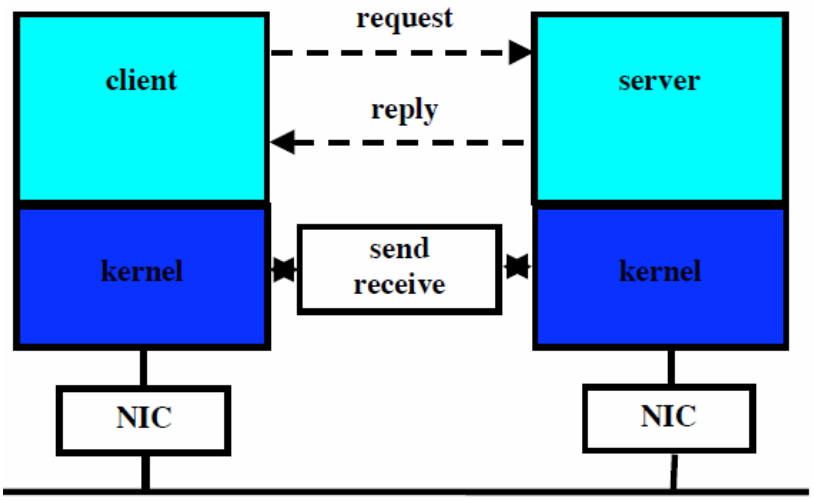
3. **Szervernév alapú címzés**

- A szerver elérésére első próbálkozásra a kliens küld egy speciális lekérdező üzenetet egy speciális, úgynevezett névszervernek, amely során lekéri a csomópont címét.
- A szerver címének birtokában a kliens kérését közvetlenül a megfelelő szerverhez lehet küldeni.



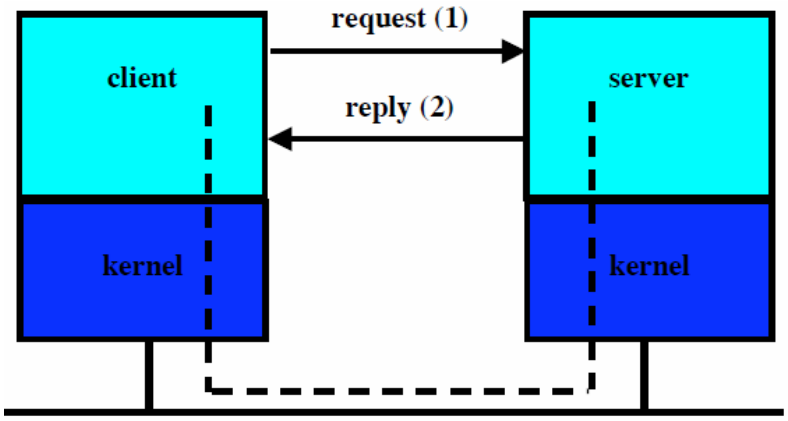
Kommunikáció a kliens-szerver rendszerekben

- a kliens-szerver üzenetváltás egy kérés-válasz párral megvalósított, amely egy küldő és egy fogadó műveletet igényel mind a kliens mind a szerver oldalon.
- A kliens-szerver kommunikációban az alábbi üzenettípusok jelennek meg:
    - Kérés: egy szolgáltatás igénylése a szervertől.
    - Válasz: egy kliensnek küld választ valamely szolgáltatás teljesítése után.
    - Nyugtázás: az előzőleg megérkezett üzenet megerősítése.
    - „Élsz?” (are you alive?): ellenőrzi, hogy egy szerver működik-e.
    - „Élek” (i am alive): megerősít, hogy a szerver műkdik.
    - „Próbáld újra” (try again): a szerver pufferja tele van, később kell újrapróbálni.
    - Ismeretlen cím: az adott címhez nincs processz tárítva.



Nem pufferelt kommunikáció

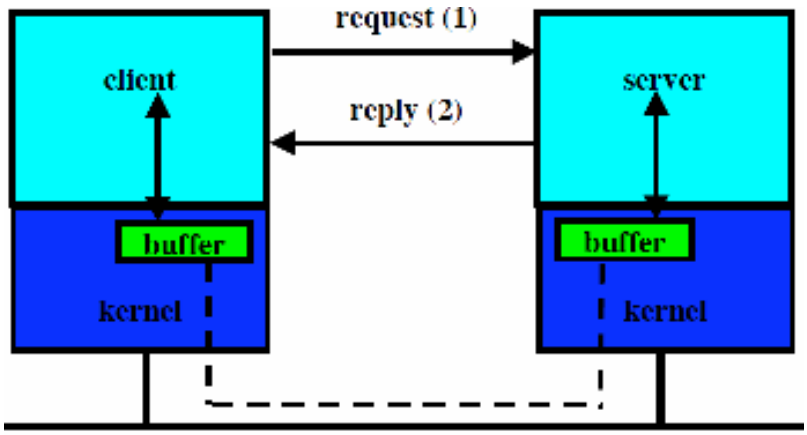
- A szervernek el kell végeznie egy fogadó műveletet mielőtt a kliens végrehajt egy küldő műveletet, mert egyik oldalon sincs puffer.



Pufferelt kommunikáció

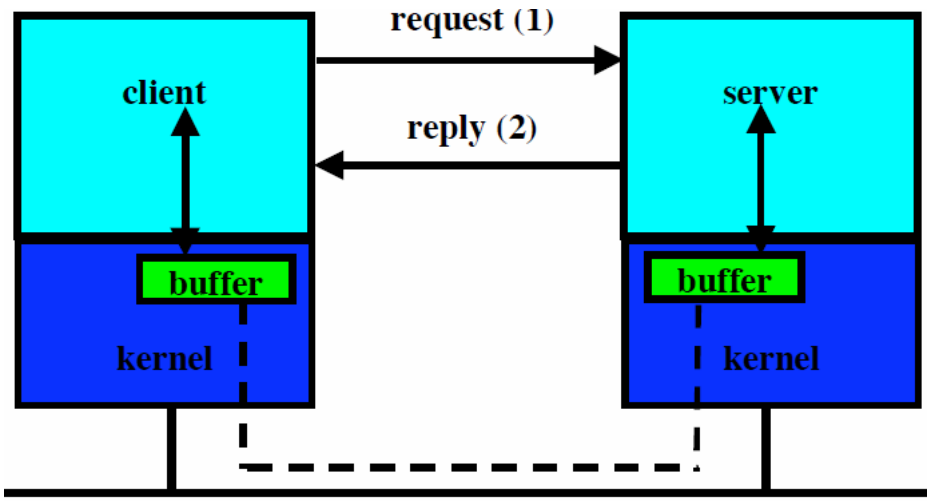
- Az üzeneteket a kliens processzek a kernel pufferba (mailbox) küldik.

- Az üzenet továbbítódik a kliens pufferből a kommunikációs hálózaton keresztül a szerver pufferba.
- A szerver oldali kernel kiveszi a megérkezett üzenetet a kernel pufferből (mailbox), majd és a szerver processnek adja.



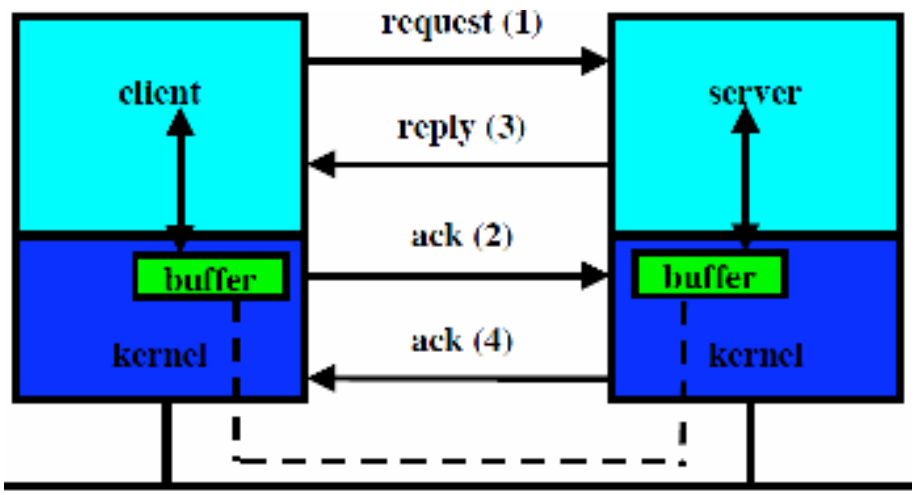
Nem megbízható kommunikáció

- Sem a kérés, sem a válasz üzenet nem visszaigazolt.
- Amennyiben a kérés, vagy válasz üzenet elveszik, visszaállítás (recovery) szükséges a kliens-szerver szinten.

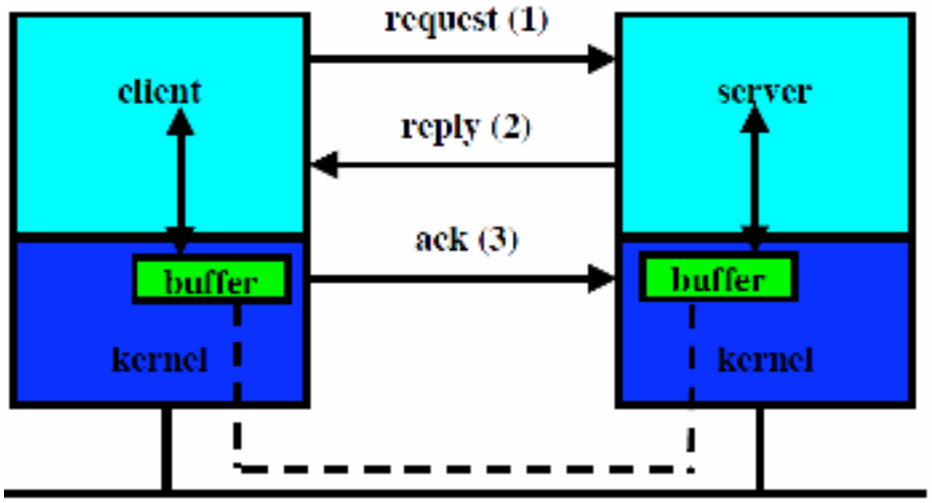


Megbízható kommunikáció

- Mind a kérés, mind pedig a válasz üzenet visszaigazolt a kernel által a kommunikációs hibák kezelése miatt.
- Az időtúllépés (timeout) és az üzenetújraküldés menedzseli a kernel szinten elvesztett üzenetek visszaállítását.



- A kommunikációs hálózat terhelésének csökkentése végett a válaszüzenet egyfajta visszaigazolásként szolgál a kérésre és a kliens processz, és a kliens processz küld egy visszaigazoló üzenetet az adott szolgáltatás teljesítéséről.



Csoport kommunikáció

Csoport

- processzek olyan csoportja, amelyek üzenetküldéssel tevékenykednek együtt az elosztott rendszerekben.
- A csoport lehetővé teszi a processzek számára, hogy a processzek összességét egy egyszerű absztrakciónak értelmezzük.
- Amikor egy üzenet kerül elküldésre a csoportnak, akkor a csoport összes megkapja, mert ez egy egy-több kommunikációs kapcsolat.
- Egy processz küldhet egy üzenetet a szerverek egy csoportjának anélkül, hogy tudomása lenne arról, hogy hol találhatók a szerverek, és hogy hányan vannak a csoportban.
- Egy processz csatlakozhat egy csoporthoz vagy elhagyhat egy csoportot., valamint egyszerre akár több csoport tagja is lehet.

Csoport kommunikáció tulajdonságai

1. **Atomitás**:

- minden szerver megkap minden kérést, így a szerverek mindegyike ugyanazt a szolgáltatást biztosítja és mindig azonos állapotban vannak.
- Amikor egy processz küld egy üzenetet a csoportnak, akkor nem kell amiatt aggódnia, hogy mi történik akkor, ha a csoportból valaki nem kapja meg az üzenetet.
- Az elosztott rendszerek programozását megkönnyítik, mert nem kell amiatt aggódnia, hogy mi történik akkor, ha a csoportból valaki nem kapja meg az üzenetet.

Atomi kommunikációs algoritmusok:

a. a küldő processz küld egy üzenetet a csoport minden tagjának
b. amikor egy processz megkap egy üzenetet, akkor ha még nem kapta meg az üzenetet korábban, akkor elküldi a csoport minden tagjának. Amennyiben már megkapta korábban az üzenetet, akkor eldobja.

2. **Üzenet rendezés:**

- a replikált szerverek mindegyikének azonos sorrendben kell kezelni a kéréseket több kliens esetén is.
- Üzenet sorba rendezés nélkül ha két processz üzenetet küld a csoportnak körülbelül ugyanabban az időben, akkor előfordulhat, hogy az üzenetek nem ugyanabban a sorrendben érkeznének a csoport egyes tagjaihoz.
- Az atomi és megbízható csoport-üzenetküldés megőrzi az üzenetek sorrendjét bármely processzpár között.

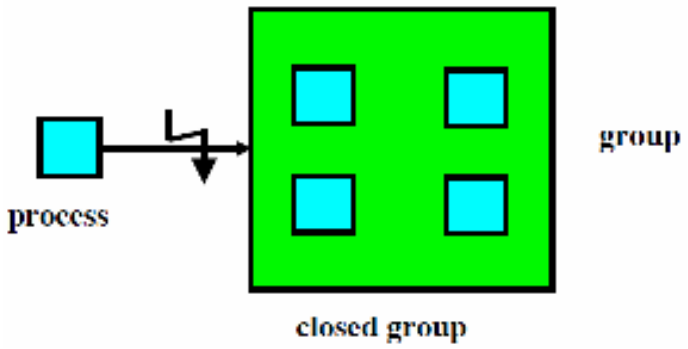
Rendezés típusai:

a. teljesen (globális) rendezett csoport-üzenetküldés: minden üzenet (minden küldőtől) azonos sorrendben kézbesítődik a csoport tagjai között, mint ahogy elküldésre kerültek.
b. Konzisztens idő-rendezett csoport-üzenetküldés: amennyiben két üzenet közel egy időben kerül elküldésre, akkor a rendszer kiválaszt egyet elsőnek megjelölve, és a csoport tagjainak ilyen sorrendben kézbesíti azokat.

Csoport típusok

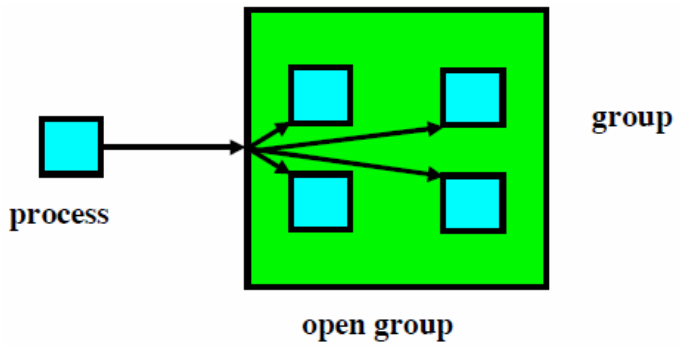
1. Zárt csoportok

- Processzek a csoporton kívül nem küldhetnek üzenetet az egész csoportnak, csakis a csoport egyes tagjainak.
- Csak a csoporthoz tartozó processzek küldhetnek üzenetet a csoportnak
- A zárt csoportok a párhuzamos feldolgozásoknál alkalmazottak. Például a processzek egy csoportja együtt dolgozik egy saját cél érdekében.



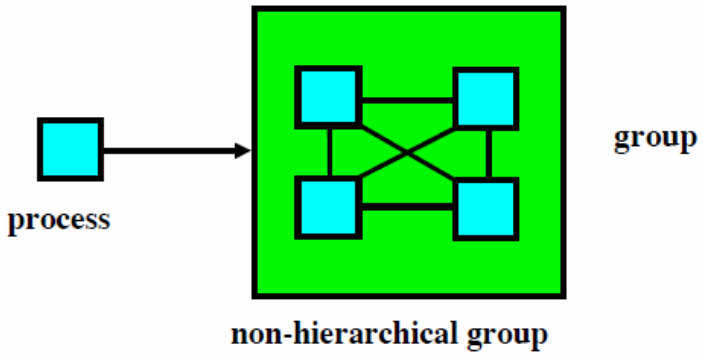
2. Nyitott csoportok

- A rendszerben lévő bármely processz képes üzenetet küldeni az egész csoportnak.



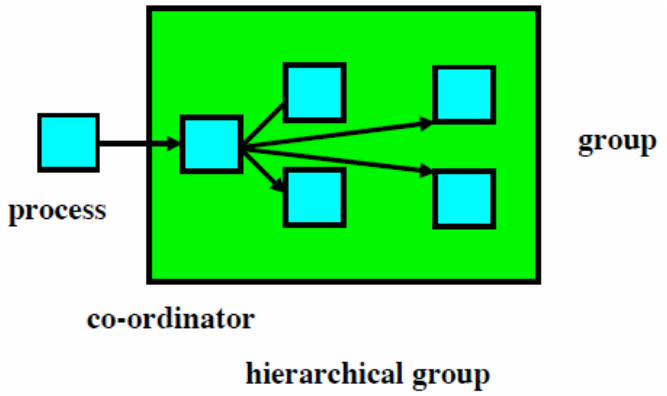
3. **Nem hierarchikus csoportok**

- A csoporton belül minden processz egyenlő és a döntések közösen születnek meg.
- A csoport hibatűrő. Ha valamely processz meghibásodik a csoporton belül, akkor a csoport egyszerűen kisebb lesz, de képes folytatni működését.



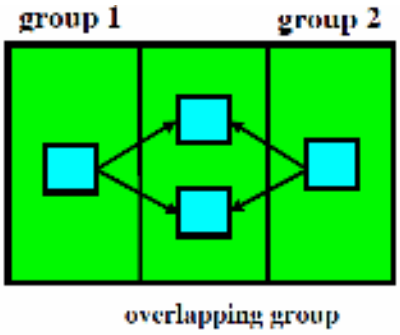
4. **Hierarchikus csoportok**

- A csoporton belül valamilyen hierarchia létezik. Például egy processz a koordinátor és a többi pedig a dolgozó.
- Amikor egy kérés generálódik a csoporthoz, akkor a kérés a koordinátornak kerül elküldésre, akinek a feladata, hogy eldöntse azt, hogy a kérésnek melyik dolgozó felel meg a legjobban, és továbbítja neki.
- A koordinátor meghibásodása az egész csoport leállásához vezet.



5. **Átfedő csoportok**

- Egy processz egyszerre több csoport tagja is lehet egy időben.
- Probléma: inkonzisztencia alakulhat ki, ha egy processz üzenetet akar küldeni több csoportnak.



Csoport tagság

- processz hozzáadását, menedzselését, törlését egy csoporton belül egy csoport szerver végzi, amely mindenen operációkat eltudja végezni.

A csoportszerverek két típusa:

1. **Centralizált csoport szerver**

   - Létrehoz és töröl csoportokat és lehetővé teszi a processzeknek a csoportokhoz való csatlakozást valamint a csoport elhagyását.
   - A csoport szerver egy adatbázist tart fent a rendszerben lévő csoportokrl és azok tagjairól.

2. **Decentralizált csoport szerver**
   - A processz egy „hello” üzenetet a csoport minden tagjának mielőtt belép a csoportba és egy „viszlát” üzenetet mielőtt a csoportot elhagyja.

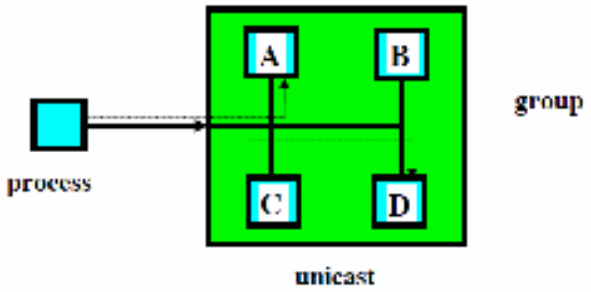
Csoport címzés

- egy processznek hozzá kell férnie a csoporthoz a csoport címét felhasználva, amely egy egyedi azonosító.

1. Minden csoportnak egyedi azonosítója van

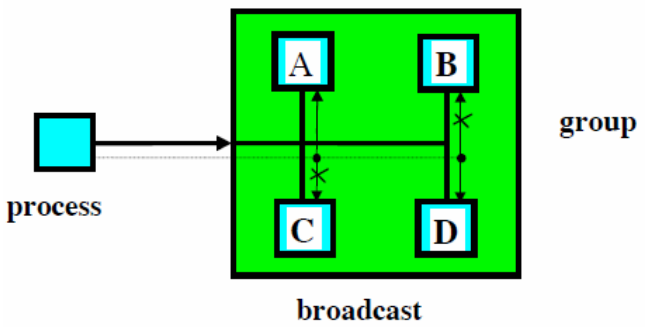
   - A processz az üzenetet a csoport címére küldi és ekkor a csoport minden tagjának kikézbesítódik.

     a. unicast címzés
     - a kernel egy pont-pont üzenete küld minden processznek a csoportban a csoporthoz tartozó processzek listája alapján.



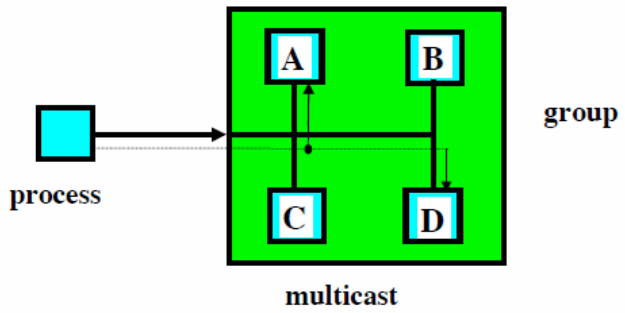
b. broadcast címzés

- minden kernel megkapja az üzenetet és kiveszi a csoport címét. Amennyiben a processz nem tagja egyik csoportnak sem, úgy az üzenet eldobódik.



c. multicast címzés

- létezik egy olyan speciális hálózati cím a rendszerben, amelyen több számítógép is képes hallgatni.
- A kernel elküldi az üzenetet minden olyan csomópontnak, amelyiknek szüksége van rá, és nem többnek.



Multicast típusok:

a. **atomi multicast:** az üzenetet vagy megkapja a csoport minden tagja, vagy egyik sem. (A meghibásodott processzek kizárásra kerülnek a csoportból)

b. **Megbízható multicast:** az üzenetküldés minden tőle telhetőt megtesz, hogy az ünezetet minden tag megkapja a csoportban, azonban nem garantált, hogy így is történik.

Multicast üzenetek előnyei

1. Replikált szervereken alapuló hibatűrési tolerancia: a kliens kérések a csoport minden olyan tagjának kézbesítődnek, amelyek azonos műveletet végeznek.

2. Objektumok helyének meghatározása az elosztott rendszerben: a multicast üzenetek felhasználhatók az elosztott rendszeren belüli objektumok helyének meghatározására is.

3. Replikációk útján nyert jobb teljesítmény: egy szolgáltatás teljesítményének a növelése érdekében az adatok replikálásra kerülnek.

4. Többszörös módosítás: multicast üzenet küldése egy csoportnak alkalmas lehet arra, hogy a processzeket értesítsék arról, hogy valami történt.

2. A hívó fél egy explicit listát biztosít a címzettekről.