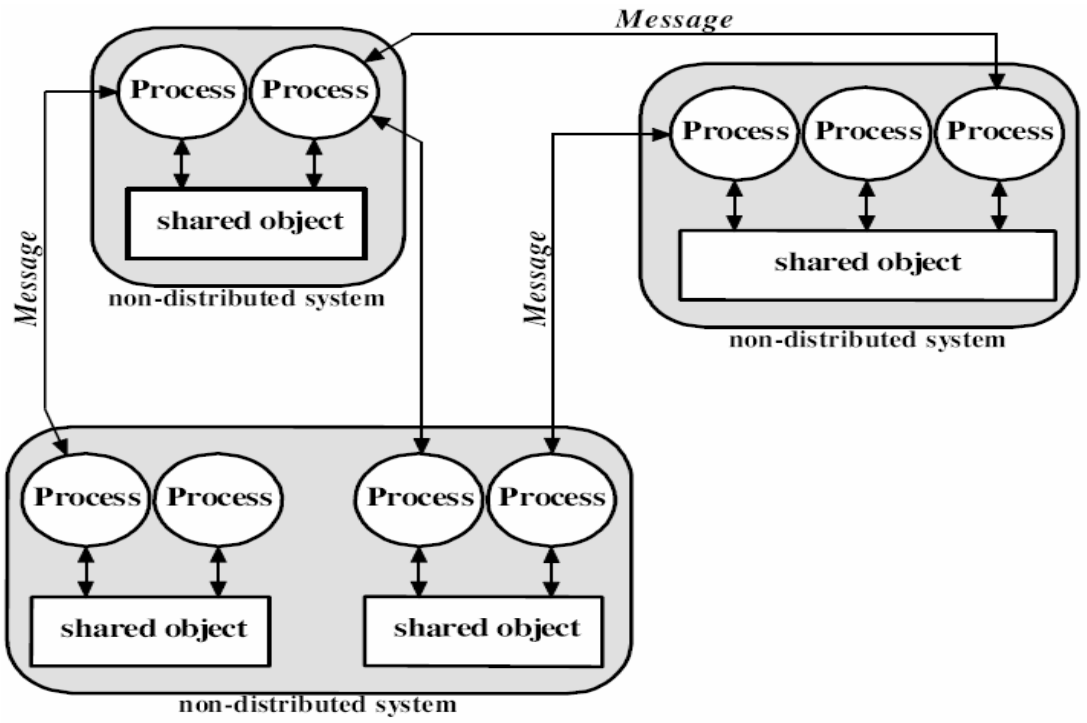
Bevezetés az elosztott rendszerekbe

Elosztott rendszerek
    elosztott rendszerek definíciója
Elosztott rendszerek struktúrája
    Hardver architektúrák
        Szorosan kapcsolt rendszerek
        Lazán kapcsolt rendszerek
Szoftver architetúrák
    Hálózati operációs rendszer (NTOS)
    Multiprocesszor operációs rendszr (MTOS)
    Elosztott operációs rendszer (DTOS)
    Kooperatív autonóm rendszerek
Eloszott rendszerek modelljei
    Mini computer modell
    Workstation modell
    Workstation -server modell
    Processzor – pool modell
    Hibrid modell
Az eloszott rendszerek tulajdonságai
    Fő karakterisztikák
    A rendszer tervezési szempontjai
    Előnyök és hátrányok

Elosztott rendszerek definíciója

- az elosztott rendszer olyan számítógépek halmaza, amely a felhasználók számára egy számítógépként jelenik meg, valójában azonban több különböző gép összessége.
- a felhasználó a rendszert egy virtuális „uniprocesszornak" látja és nem gépek halmazának.
- A felhasználó nem tudja, hogy egy feladat melyik gépen kerül végrehajtásra, hogyan kerül tárolásra és azt, hogy hogyan valósul meg a kommunikáció és szinkronizáció a számítógépek között.
- A különböző adatok elérése és a számítógépek közötti kommunikáció tehát láthatatlan marad a felhasználók előtt.
- A láthatatlanság úgy valósul meg, hogy az elosztott rendszer számítógépeinek mindegyike ugyanazt az operációs rendszert, az elosztott operációs rendszert futtatja
- az elosztott operációs rendszer elrejti a megvalósítást a felhasználók előtt.



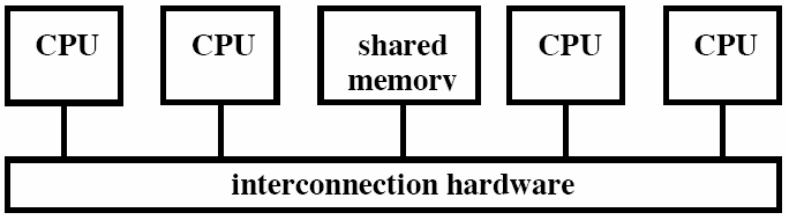
Distributed system which comprises several non-distributed systems.

Elosztott rendszerek struktúrája

Hardver architekturák

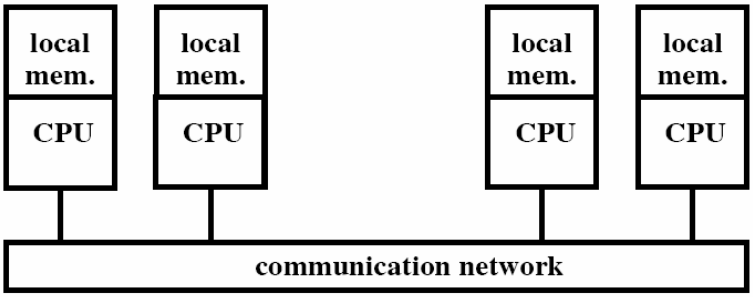
<u>Szorosan kapcsolt rendszerek</u>

- Tipikus párhuzamos rendszerek, ahol a memória címzési területe közös minden processzornak.



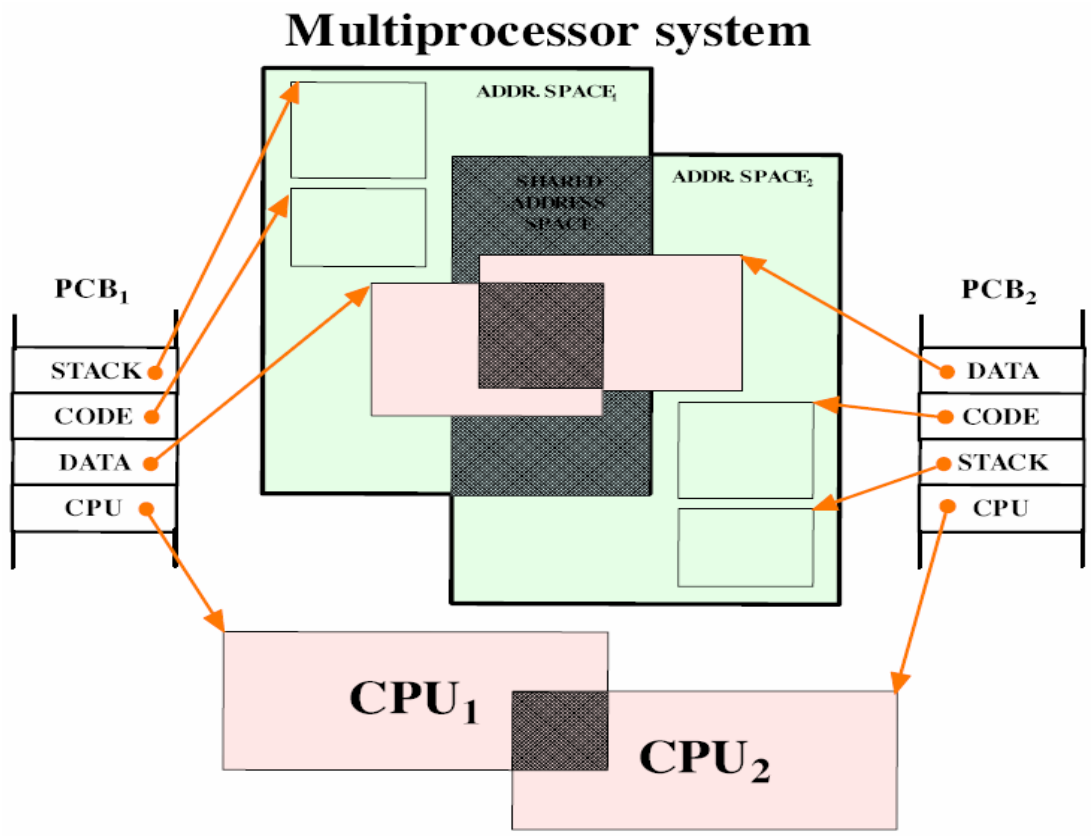
tightly-coupled systems

<u>Lazán kapcsolt rendszerek:</u> Olyan elosztott rendszerek, ahol minden processzornak saját memóriája van.

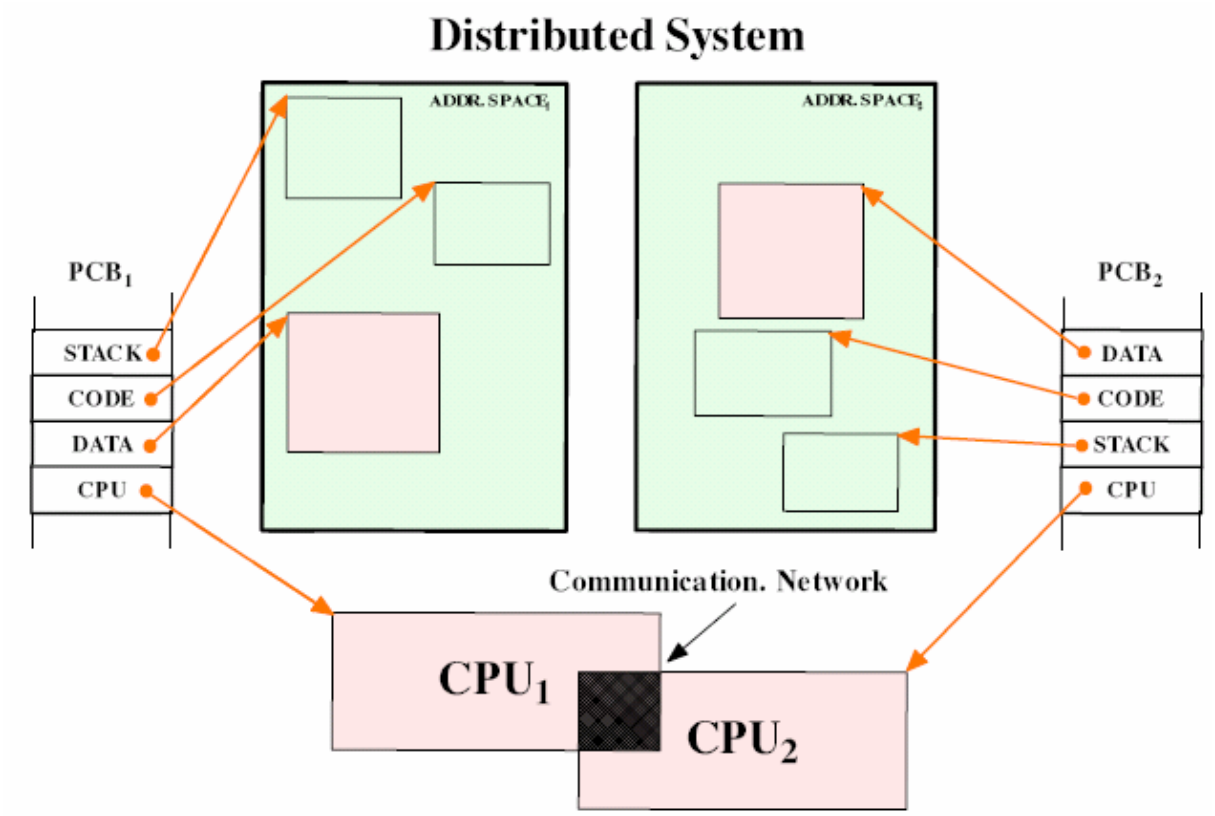


loosely-coupled systems

Multiprocesszor rendszerek:



Elosztott rendszerek:



Szoftver architektúrák

Hálózati operációs rendszer (Network operating System(NTOS))

- kommunikációs hálózaton keresztül lazán kapcsolt számítógépek halmaza, ahol az egyes számítógépek operációs rendszeri is lazán kapcsoltak. Pl. mindegyik gép saját operációs rendszert futtat, amelyek talán különböznek is egymástól.
- az operációs rendszer menedzseli az egyes számítógépeket és a kapcsolatot a felhasználók és a szerverek között.
- mindegyik számítógép „azt tehet amit akar" mindaddig, amíg ez a tevékenység megfelel az OS szabványos protokolának.
- nincs közvetlen hardver vagy szoftver felügyelet a gépek között a hálózatban és a gépek között a kommunikáció sokkal nagyobb, mint a rendszeren belüli adatátvitel.
- a rendszer tartalmazhat különféle szervereket is, mint például egy file vagy nyomtatószerver.

Multiprocesszor operációs rendszer (MTOS)

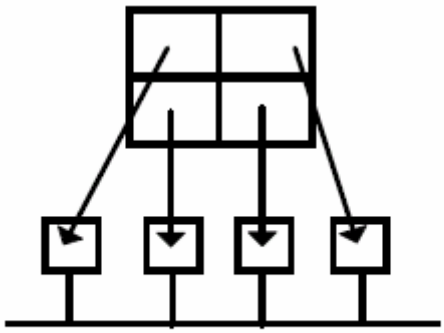
Legfontosabb jellemzői:

- a számítógépek szorosan kapcsoltak egy buszrendszer segítségével, és szorosan kapcsolt operációs rendszer jellemző.
- Jellemzően osztott memóriát használ a megközelítés, ahol megtalálható az összes futásra kész process listája
- a processzoroknak nincs saját lokális memóriája, minden program a globális, osztott memóriában tárolódik

Elosztott operációs rendszer (DTOS)

Legfontosabb jellemzői:

- a számítógépek lazán kapcsoltak egy kommunikációs rendszeren keresztül, és szorosan kapcsolt operációs rendszerrel rendelkeznek. Pl. minden gép ugyanolyan típusú kernellel rendelkezik.
- a processz menedzsment azonos minden gép esetében, ezért léteznie kell egy globális file rendszernek az egész elosztott rendszerre nézve.
- támogatja a globális interprocess kommunikációs mechanizmust. Pl. lokális és távoli kommunikáció azonos módon történik.



decomposition in distributed systems

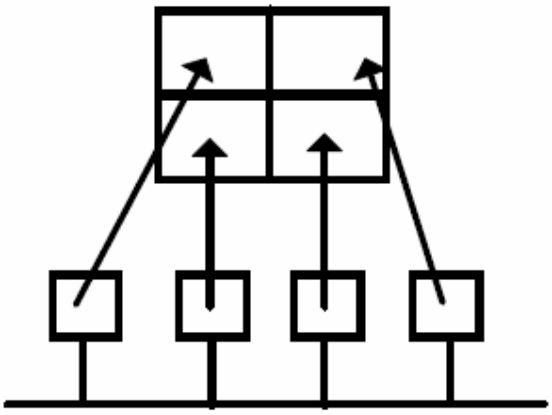
- az elosztott rendszerek jellemzője a szolgáltatás alapú elosztottság

Együttműködő autonóm rendszer

Legfontosabb jellemzői:

- magas szintű szolgáltatás-orientált szoftver rendszer, amely kommunikációt igényel.

- a kulcskoncepciója ezen rendszereknek az, hogy a különböző szolgáltatásokat integrálja valamely magasabb szintű együttműködő aktivitás létrehozása érdekében.
- az együttműködő autonóm rendszerek intelligens „trader" és „broker" szolgáltatásokat alkalmaznak más együttműködő rendszerek összekapcsolására
- az ilyen rendszerekben az egyes szoftver és hardver elemek teljes mértékben különállók és decentralizáltak.
- a rendszer minden felhasználója autonóm tevékenykedik a rendszerben szolgáltatások hívásával
- csoportos aktivitások szintén megvalósíthatók alacsonyabb szintű szolgáltatásokból magasabb szolgáltatásokat szervezve.
- az autonóm elemekből tetszőleges szoftverrendszer alakítható ki minden korlát nélkül különféle szolgáltatások integrálásával.



composition in co-operative systems

- számos fejlesztés kíséri az autonóm együttműködő rendszereket: Pl.

ODP (Open Distributed Processing): támogatja az elosztást, az együttműködést, a hordozhatóságot heterogén folyamatok számára mind autonóm rendszereken belül, mind pedig autonóm rendszerek között.

CORBA (Common Object Request Broker Architecture): Objektum orientált modellt alkalmaz annak megvalósítására, hogy egységes szolgáltatásokon keresztül valósuljon meg a heterogén rendszerek összekapcsolása.

Elosztott rendszerek modelljei

Minicomputer modell

- számos interaktív terminál van kapcsolva a valamely miniszámítógéphez, amelyek pedig egy kommunikációs hálózaton keresztül kapcsolódnak egymáshoz.
- mindegyik számítógép több felhasználót támogat, valamint lehetőséget biztosít távoli erőforrásokhoz.
- a számítógépek és a felhasználók számának aránya kisebb, mint 1.

minicomputer model

Workstation modell

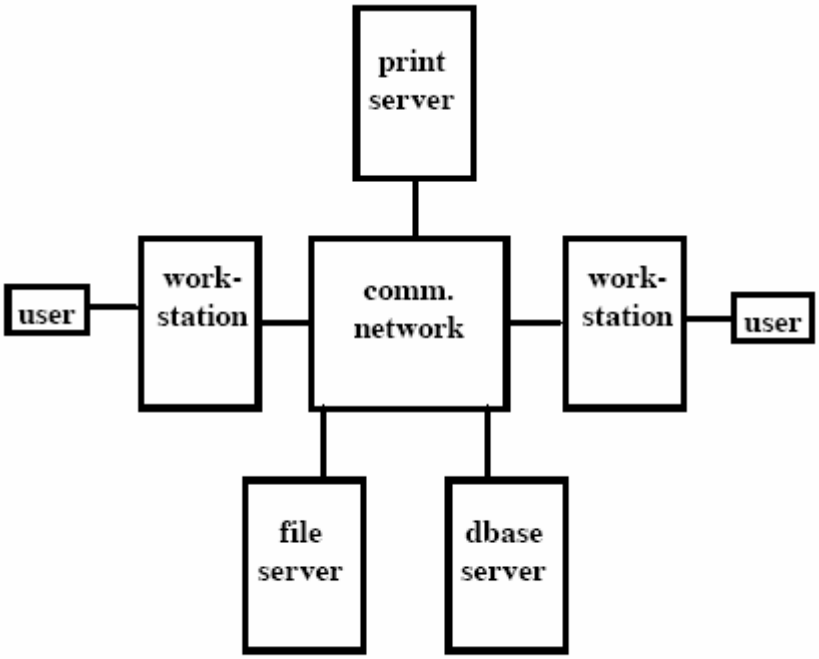
- számos munkaállomás kapcsolódik össze egy kommunikációs hálózaton keresztül
- minden felhasználónak saját munkaállomása van, amelyen futtatja feladatait
- jellemző az elosztott file rendszer, amely lehetőséget biztosít a felhasználók adatainak hozzáféréséhez
- a számítógépek és a felhasználók számának aránya pontosan 1.

workstation model

Workstation szerver modell

- a modell több munkaállomáson alapszik, amelyeket a felhasználók használnak, valamint több szerveren, amelyek mindegyik valamilyen szolgáltatást biztosít a felhasználóknak.

- egy munkaállomás olyan önálló számítógép, amely saját helyi feldolgozási lehetőséggel rendelkezik, és a felhasználók számára hozzáférést biztosít a kommunikációs hálózathoz.
- a különböző munkaállomások a szerverekkel kommunikációs hálózat segítségével kapcsolódnak, és érik el a szerver szolgáltatásokat.
- a szerverek olyan számítógépek, amelyek gyors és/vagy nagy teljesítményű hardver és szoftver erőforrásokkal vannak felszerelve, és szolgáltatásokat biztosítanak a munkaállomások számára
    - kliens: olyan felhasználó, aki szolgáltatást igényel a szervertől
    - szerver: szolgáltatásokat biztosít az „ismereten" kliensek számára
    - szolgáltatás: valamilyen szoftver entitás, amely egy vagy több szerveren fut



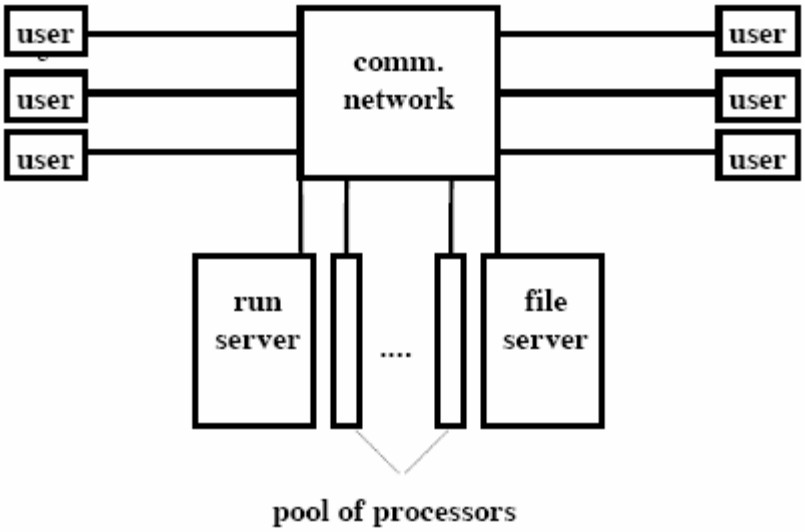
workstations-server model

A workstation szerver modell előnyei:

- a felhasználók által használt gyors és nagy teljesítményű erőforrások megvalósítása sokkal „olcsóbb" a modell segítségével, mint ha külön dedikált erőforrásokat rendelnénk minden felhasználóhoz.
- a modell nem igényel processz szintű migrációt, mert kérés-válasz (request-response) alapú protokollt használ.
- a felhasználóknak garantált a válaszidő nagysága, mert a munkahelyek nem futtatnak távoli feladatokat.

Processor-pool modell

- az egyes processzorokat összegyűjtik, „egyesítik" és megosztják őket a felhasználók között igény szerint. Minden processzornak saját memóriával rendelkezik azért, hogy rendszer, vagy más alkalmazásokat tudjon futtatni függetlenül a többitől.



processor-pool model

- egy speciális processzor (run server) menedzseli a processzorok felhasználókhoz való rendelését igény szerint
- a modell egy vagy több processzort allokálhat egy felhasználóhoz figyelembe véve a felhasználó igényeit. A processzorok a feladatuk elvégzése után visszatérnek a csoporthoz, és várják a további igényeket.
- a processzorok és a felhasználók számának hányadosa tipikusan több mint 1.

Előnyök:

- a rendszerben megtalálható erőforrások jobb kihasználását teszi lehetővé
- a rendszer szolgáltatásai könnyen bővíthetők több számítógép telepítése nélkül is

Hátrányok:

- nagy sebességű grafikus vagy ablakos rendszereket használó interaktív alkalmazások nem támogatottak.

Hibrid modell

A hibrid modell a *workstation* és a *processzor-pool* modellen alapszik.

Elosztott rendszerek fő tulajdonságai

<u>Erőforrás megosztás:</u> Az erőforrás megosztás az elosztott rendszerek legfontosabb jellemzője.

- valamely erőforrás fizikailag egy számítógéphez tartozik, amely más számítógépekről is elérhető kommunikáció útján.
- a DTOS rendszerek felhasználóinak képesnek kell lennie a távoli erőforrások olyan könnyen elérni, mintha a saját helyi erőforrását használná.

Az erőforrás megosztás szükségei:

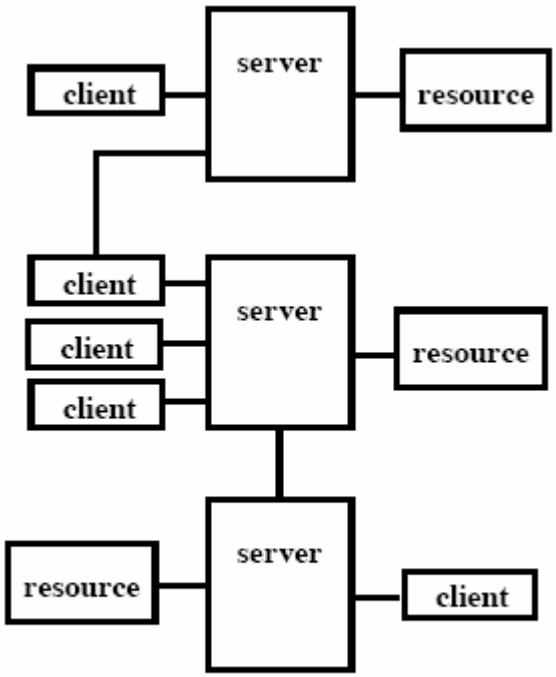
- elnevezési konvenció az erőforrások minden csoportjának, melyeket távolról lehet elérni
- az erőforrásnevek hálózati címekre való leképzése
- a konkurens hozzáférések menedzselése

Az erőforrások a következő formában álnak rendelkezésre a felhasználók számára:

- **adat migráció**: az adat mindig a felhasználás helyére szállítódik. Az adat lehet file, vagy egy lokális vagy távoli gép fizikai memória tartalma egyaránt.
- **számítási migráció:** az egyes számítások különböző helyeken hajtódnak végre.
- **elosztott ütemezés:** az egyes processzek egyik gépről átkerülhetnek más gépekre végrehajtás céljából. A feladatok áthelyezése szükséges, hiszen az a s
- **s**zámítógép, ahonnan a processz származik, túlterhelődhet.

Az erőforrás megosztás modelljei

<u>Kliens-szerver modell:</u>



client-server model

- A kliens szerver modell egy olyan folyamat modell, amely az üzenetküldés (message passsing) módszerével biztosítja az interakciót és kooperációt a folyamatok között.
- szerverek halmaza, amelyek mindegyike erőforrás menedzserként viselkedik az erőforrások kollekciója felett.
- kliens processzek halmaza, ahol mindegyik folyamat valamely osztott hardver vagy szoftver erőforrás hozzáférést igényel.
- a rendszerben a kliens folyamatok kéréseket intéznek a szerverek felé, amely során valamely erőforrást igényelnek. Ezután a szerver biztosítja a kérést és visszaküldi a választ a kliens folyamatnak.

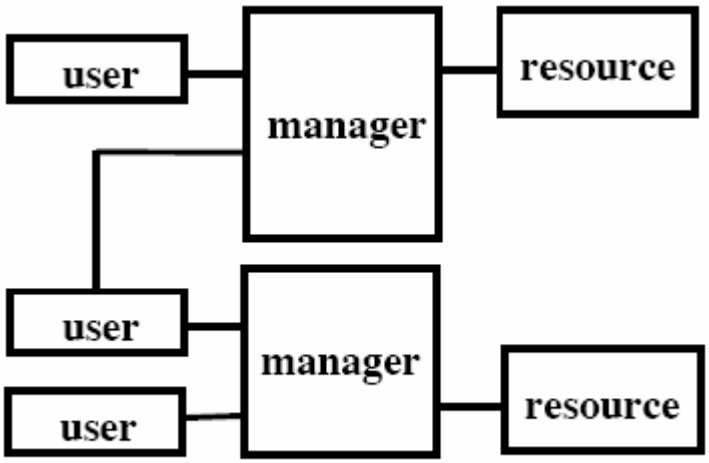
A kliens szerver modell problémái:

- mivel az erőforrások vezérlése centralizált egy adott szerverre, így szűk keresztmetszetet jelenthet, különösen meghibásodás esetén.
- a teljesítmény növelése megvalósítható több szerverrel, ez azonban növeli az elosztott rendszer költségét.

Objektum alapú modell:

- minden entitás egy futó programban objektumként értelmezett, amelynek működéséhez üzenetkezelő interace-eken keresztül lehet hozzáférni.

- minden osztott erőforrás szintén objektumként értelmezett. Pl. összetartozó adatstruktúrák együttese, az ehhez definiált operációk halmaza, és az operációk, szolgáltatások objektumok által megvalósított.



object-based model

- Az objektumok üzenetekben megvalósított kommunikációs primitívekkel kommunikálnak.
- a szinkronizáció és a vezérlés menedzsment során valósul meg.
- A kommunikáció menete: ha egy program hozzáférést akar valamely erőforráshoz, üzenetet küld, amely tartalmazza a kérést a megfelelő objektumhoz. Ezután az üzenet kézbesítésre kerül a megfelelő folyamathoz, amely elvégzi a megfelelő operációkat, majd visszaküldi a választ a kérő folyamatnak, ha szükséges.

<u>Pipe (cső) modell:</u>

- a cső egy a file rendszert használó kommunikációs lehetőség, amely
    - az adattranszfert a processzer között FIFO stratégiával végzi
    - lehetővé teszi a processzek szinkronizálást
- a cső alapú kommunikáció lehetővé teszi a folyamatok számára az üzenetek cseréjét úgy, hogy nem tudják, milyen folyamat van a cső végén.
- a módszert távoli eljárások kommunikációjánál alkalmazzák, azonban a cső nem blokkolja a hívót, és nincs visszatérési értéke.
- típusok: nevezett (*Named pipe*) cső – *open* rendszerhívás UNIX, nem nevezett cső (*Unnamed pipe*) – *pipe* rendszerhívás UNIX.

Autonómia és egymásrautaltság

A DTOS autonómiát nyújt a következő okok miatt:

- szabad politikát biztosít azért, hogy egy adott felhasználó hol, hogyan és mikor végezzen számításokat.

- robosztus működés. Pl. a rendszer működőképes marad akkor is ha egy rendszerkomponens kiesik.
- különböző felhasználók közötti együttműködés kontrollálása, és a helyes működés biztosítása különböző problémák fellépése esetén is.

A DTOS minimalizálja az egymásrautaltságot a következő módszerek felhasználásával:

- az olyan szolgáltatások, amikre nincs szükség, nem kell használni.
- a kommunikáció túlterheltségének kivédése érdekében olyan munkaállomások alkalmazása szükséges, ahol a kért szolgáltatások lokálisan elérhetők.
- az erőforrások replikálása szükséges, amennyiben valamilyen kritikus szolgáltatást kell használni.

Heterogenitás és nyitottság

Az elosztott rendszerek heterogenitási szintjei:
- számítógép heterogenitás
- hálózat heterogenitás
- operációs rendszer heterogenitás

A nyitott elosztott rendszerek több formában is bővíthetők. Hardver szinten gépek hozzáadásával, szoftver szinten pedig új szolgáltatások hozzáadásával lehetővé téve a programoknak az erőforrások megosztását, mert

- a kulcs interface-ek publikáltak,
- és egységes interprocess kommunikáción alapulnak.

Konkurencia és párhuzamosság

Az elosztott rendszerekben a konkurencia és a párhuzamosság a felhasználók szeparált aktivitásaiból, az erőforrások függetlenségéből, és a szerver folyamatok különböző gépeken való elhelyezkedéséből származik. Ennek oka:

- több felhasználó egyidejűleg hív parancsokat vagy kommunikál más programokkal.
- több szerver folyamat konkurensen fut, mindegyik a kliens processzek különböző kéréseire válaszol.

Skálázhatóság

- a számítógéprendszerek idővel folyamatosan nőnek, amelynek eredménye nem szabad, hogy a sebesség, vagy a rendelkezésre állás rovására menjen.
- a rendszer skálájának növelése nem járhat a szoftver illetve hardver elemeinek megváltozásával.
- egy elosztott rendszerben nem egyetlen fajta erőforrást tételezünk fel, igény esetén a rendszernek lehetővé kell tennie a rugalmas bővítést is.

Kompatibilitás

A kompatibilitás a rendszerben lévő különböző erőforrások együttműködését jelenti. Három alapvető szintje:

- **bináris szintű kompatibilitás**: minden processzor azonos bináris utasításkészlettel dolgozik még akkor is, ha a processzorok teljesítménye és input-output tevékenysége különbözik.
- **futás szintű kompatibilitás:** ugyanaz a forráskódnak kell lefordulnia és helyesen végrehajtódnia minden számítógépen a rendszerben.
- **protokoll szintű kompatibilitás:** az interoperabilitást úgy valósítja meg, hogy minden rendszerkomponens azonos protokoll készletet használ. A számítógépek különböző operációs rendszereket futtathatnak megtartva az interoperabilitást.

Átlátszóság

- Az átlátszóság célja az, hogy elrejtve legyenek a felhasználók elől az egyes hardver és szoftver komponensek és a különböző rendszer specifikus komponensek
- a DTOS célja az, hogy a felhasználó az egész elosztott rendszer egyetlen virtuális „uniprocesszor"nak lássa.
- alkalmazások fejlesztése egy elosztott rendszer számára sokkal nehezebb, mint egy centralizált rendszer számára, mivel
    - hozzáférés egy távoli erőforráshoz más mint a lokális erőforrások elérése.
    - a lehetséges hibák száma sokkal több, mint egy centralizált rendszer esetén.
    - az átvitel késleltetése befolyásolja a működést a rendszeren belül.
- az átlátszóság a hálózatot láthatatlanná teszi a felhasználók és alkalmazásaik előtt.

<u>Átlátszóság típusai:</u>

- hozzáférési átlátszóság
    - lehetővé teszi a távoli és helyi információk elérését azonos operációk segítségével.

    - a rendszerobjektumok fizikai szeparációja rejtett a felhasználó előtt

- konkurencia átlátszóság
    - lehetővé teszi hogy a processzek konkurensen működjenek osztott információkat használva minden beavatkozás nélkül. Ennek eredményeképpen egy felhasználó nem érzékeli más felhasználók jelenlétét.

- párhuzamosság átlátszóság
    - párhuzamos aktivitások megengedettek anélkül, hogy a felhasználó tudná, hogy hogyan, mikor, és hol kerülnek végrehajtása ezek az aktivitások a rendszerben.

- replikáció átlátszóság
    - lehetővé teszi az információk több példányban való felhasználását anélkül, hogy a felhasználó vagy az alkalmazás tudna ezekről.
    - szorosan kapcsolódik a konkurencia átlátszósághoz, csak a megvalósítás más, hiszen a file-ok és adatok speciális objektumok.

- skálázhatósági átlátszóság
    - lehetővé teszi a rendszer és az alkalmazások számára a skála kiterjesztését anélkül, hogy a rendszer vagy az alkalmazás algoritmusainak struktúráját meg kellene változtatni.

- elhelyezkedési átlátszóság
    - az információk az elhelyezkedésük ismerete nélkül elérhetők.
    - az objektumokat a rendszer leképezi és logikai nevet rendel hozzá, és ezen néven érhetők el. Így nevezhetjük a tulajdonságot *név átlátszóság*nak is.

- migrációs átlátszóság
    - lehetővé teszi az erőforrások és információk mozgását a rendszeren belül anélkül, hogy a felhasználó alkalmazásának működését befolyásolná.
    - az objektumok nem csupán a logikai nevekkel vannak leképezve, de különböző fizikai helyre is „vándorolhatnak" névváltozás nélkül.

- futtatási átlátszóság
    - lehetővé teszi az adatok, programok mozgását, a terheléselosztást (load balancing), és a különböző processzorok processzei közötti kommunikációt.

- teljesítmény átlátszóság

    - lehetővé teszi a rendszer újrakonfigurálását a teljesítménynövelés érdekében figyelembe véve a terhelés változását.
- hiba átlátszóság
    - lehetővé teszi az esetleges szoftver illetve hardver hibák elrejtését a felhasználók előtt így biztosítva a felhasználói feladatok végrehajtását.
    - a rendszerben jelentkező hibákat teljesítmény degradációvá transzformálja, megpróbálja a felhasználó veszteségét minimalizálni

Megbízhatóság

- az erőforrásoknak bármikor rendelkezésre kell állnia a felhasználók számára. A rendelkezésre állást kritikus komponensek alkalmazásával vagy redundanciával valósítható meg.
- a megbízhatóság magába foglalja azt is, hogy az erőforrás védett a nem engedélyezett hozzáférésektől.
- a meghibásodási módok függetlenek egymástól, mivel a rendszer komponensei egymástól függetlenül hibásodhatnak meg.
- a hibatűrési tolerancia két megközelítéssel növeli a megbízhatóságot:
    - **hw redundancia:** redundáns komponensek alkalmazása
    - **szoftver visszaállítás (software recovery):** lehetőség a meghibásodásból való visszaállításra.

Állapot konzisztencia

- a konkurens hozzáférések, a kommunikációs várakozások, valamint a meghibásodások az elosztott rendszert egy inkonzisztens állapotba viszi.

Időzítés

- az elosztott rendszeren belül minden gépnek saját órája van, de azonos időt nem tudnak garantálni. (Nehéz szinkronizáció).

Tervezési célok és problémák

Tervezési célok

- nagy teljesítmény
- hatékonyság:

    - az alkalmazásokat elosztott folyamatokként kell struktúrálni, így a számítás és a kommunikáció egyensúlyozható, és átlapolható ha szükséges.
    - a számítási sebesség, és a rendszer teljesítménye növelhető elosztott feldolgozással és terhelés elosztással, amennyiben a kommunikációs rendszer megfelelően van tervezve.
- skálázhatóság
- rugalmasság
    - magába foglalja a rendszer interface-ek egyszerű használatát, és a számítási folyamathoz való kapcsolódást.
    - a rendszer képesség a folyamatok kiértékelése és migrációja, amely magába foglalja a modularitást, a skálázhatóságot, hordozhatóságot, és az interoperabilitást.
- konzisztencia
    - a globális információ hiánya, az adatok lehetséges replikálásai és partícionálásai, a komponensek meghibásodása, a modulok közötti interakció a rendszer inkonzisztens állapotához vezethet.
- megbízhatóság
- biztonság
    - az elosztott rendszer támadható a nyitottsága miatt
    - a biztonság megőrzése hitelesítéssel történik, amely magában foglalja a kliensek, üzenetek, szerverek hitelesítését, valamint azonosítással, amely a kommunikációs hálózathoz való hozzáférést védi.
- robosztusság
    - a rendszer azon képessége, amikor képes önmagát újrakonfigurálni egy olyan állapotba, ahol a rendszer integritása stabil, némi teljesítmény degradációval.
- hibatűrő képesség
    - a hibák kezelését a redundanciák növelésével érhetjük el.
    - a redundancia a DS rendszerek velejárója, adat és erőforrás replikálható
    - a hibatűrés megköveteli a rendszer állapotának folyamatos figyelését

Tervezési problémák az elosztott rendszerben

1. elnevezés

- Az objektumok az elosztott rendszeren belül lehetnek processzek, fájlok, memória területek, I/O eszközök, hálózatok, és processzorok is.
- minden objektumhoz hozzárendelhető néhány jól definiált hozzáférési operáció,

- az objektumokat az objektum kiszolgálók menedzselik, amely a szervereken helyezkednek el.
- az erőforrásokhoz, objektumokhoz társított névnek globálisnak kell lennie, és a névértelmező rendszernek támogatnia kell. Ez az értelmező fordítja le a neveket, és lehetővé teszi a különböző programok számára az objektumokhoz való hozzáférést.
- **Tervezés célja:** olyan elnevezési séma kialakítása, amely könnyen skálázható, és ahol a nevek hatékonyan fordíthatók, és amely megfelelő teljesítményt nyújt.
- **Tervezés problémája:** a megfelelő névtér kiválasztása minden erőforrás számára
    - az erőforrás neveknek a kommunikáció és a szerver azonosítás során feloldása szükséges.

2. kommunikáció

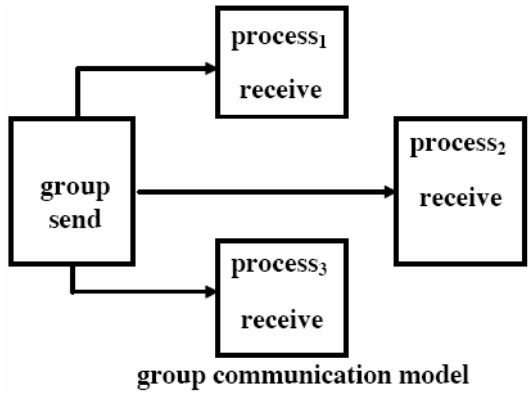
A kommunikáció a legfontosabb „dolog" egy elosztott rendszerben, minden ezen alapszik.

- a legalacsonyabb szinten üzenetküldéses (*message passing*) kommunikációt alkalmaznak a rendszerben, különféle üzenetküldéses primitíveket felhasználva.
- a legmagasabb szinten a logikai kommunikációs módszerek elrejtik az üzenetküldés fizikai megvalósításait.
- **Tervezési célok:**
    - a kommunikáció megvalósításának optimalizálása a magas szintű programozási modell megtartásával.
- **Tervezési problémák:**
    - a folyamatok közötti adatcsere megvalósítása üzenetküldéssel.
    - a küldött és fogadott aktivitások szinkronizációja kommunikációs primitívek segítségével.

Kommunikációs modellek az elosztott rendszerekben

1. **Kliens – szerver kommunikációs modell**:
    - a kliens processz küld egy kérést a szerver processznek és a kliens addig blokkolódik, míg a válasz meg nem érkezik a szerver processztől.
    - a szerver processz végrehajtja a kérést
    - a szerver processz visszaküldi a választ a kliensnek

2. **Csoport kommunikációs modell**:
   - a processz üzenetküldéssel kommunikál, de a cél nem egy külön dedikált processz, hanem a processzek egy csoportja.



3. Szinkronizáció és koordináció

- az egymással interakcióban lévő konkurensen futó processzek között koordináció szükséges a szinkronizáció megvalósítására. A szinkronizáció és koordináció 3 típusa:
  - **Határ szinkronizáció (barrier synchronisation):** a processzek halmazának el kell érnie egy egységes szinkronizációs pontot mielőtt folytatódhatnak.
  - **Feltétel koordináció (condition co-ordination):** a folyamatnak addig kell várnia, amíg egy bizonyos feltétel be nem következik. Ez a feltétel aszinkron úton következik be egy másik beavatkozó processz által, így irányítva a futás folyamatát.
  - **Kölcsönös kizárás (mutual exclusion):** a konkurens processzek kölcsönös kizárást alkalmaznak a kritikus erőforrások hozzáférésekor.
- a szinkronizáció megvalósítása nehéz feladat, mert az elosztott rendszerekben a rendszer teljes információját kell menedzselni.
- a rendszer állapotának meghatározása üzenetküldéssel valósul meg, azonban ez nem hatékony az üzenetküldéses kommunikációs rendszer várakozásai miatt.
  - Tervezési célok:

        - a rendszeren belüli együttműködő folyamatok együttes aktivitásainak specifikálása és vezérlése.
        - az osztott objektumok konkurens hozzáférésének megvalósítása több folyamat segítségével.
    - Tervezési problémák:
        - esemény sorrendiség meghatározása, hogy kizárólag egy folyamat legyen a kritikus szekcióban
        - üzenet alapú feloldási protokollok vagy elosztott koordináció biztosítja a szinkronizációt és koordinációt az elosztott rendszerekben.

4. szoftver architektúra

Tervezési célok:

- a rendszert úgy kell strukturálni, hogy új szolgáltatások könnyen bevezetetők legyenek, amelyek teljes egészében képesek együttműködni a rendszer már létező szolgáltatásaival anélkül, hogy szükség lenne valamely szolgáltatás duplikálására.

Tervezési nehézségek:

- a kernel szerepének lekorlátozása az alapvető erőforrás menedzsmentre.
- minden más szolgáltatást osztott erőforrásokhoz, szolgáltatásokhoz kapcsolódó nyitott szolgáltatásokkal kell megvalósítani.

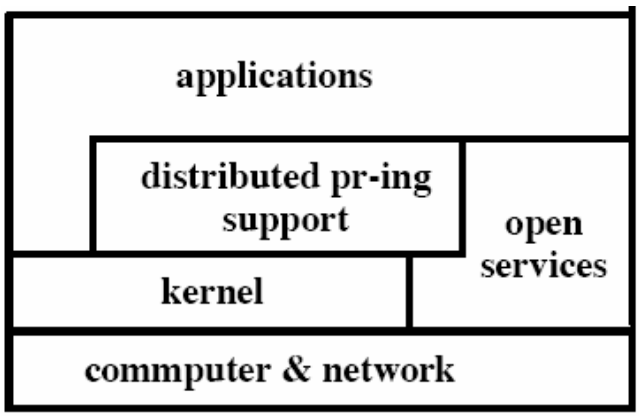
Alapvető kernel típusok:

1. **Monolitikus kernel:** a monolitikus kernel az operációs rendszer minden szolgáltatását magában foglalja. A rendszerhívások közvetlenül a kernelhez továbbítódnak, amely biztosítja a kért szolgáltatást, és az eredményt visszaküldi a kliens processznek.

2. **Microkernel:** az operációs rendszer olyan folyamatok együtteseként strukturált, amelyek nagymértékben függetlenek egymástól.

    - az erőforrások és szolgáltatások legalapvetőbb halmazát biztosítja úgy, hogy közben védi az alapvető hardver és szoftver komponenseket a független folyamatoktól.
    - a mikrokernel támogatja az interakciót a processzek és néhány alapvető szolgáltatás között, amelyek nélkülözhetetlenek minden számítógép számára. Pl.:
        - memória allokáció és védelem

- kisebb fajta processz menedzsment és ütemezés
- interprocessz kommunikáció
- alacsonyszintű input/output kezelés



structure of the microkernel

- a mikrokernel nem kínál szolgáltatásokat a fájlrendszer számára, a rendszerhívások nagy része és a teljes processz menedzsment a felhasználói szinten van megvalósítva.
- a rugalmasság szolgáltatások hozzáadásával illetve módosításával van megvalósítva. Úgynevezett „open services" rendszeren keresztül, ahol nincs szükség a rendszer leállítására és az új kernellel való bootolásra.
  - a nyitott szolgáltatások az elosztott rendszer programozásának lehetőségeit egészen az alkalmazások programozásának szintjéig emeli. Pl.: elektromos levélkézbesítő szolgáltatás.
- futásidejű támogatás arra, hogy lehetőséget biztosítson más programnyelveken készített alkalmazások együttműködésére. Pl.: távoli eljáráshívás, csoportos operáció

5. Terhelés allokáció

Tervezési célok: a rendszerben zajló feldolgozásokat, kommunikációkat, és az erőforrások egy hálózaton belül a lehető legjobban kihasználni. Meg kell próbálni a rendszer erőforrásait optimalizálni változó terhelések mellett.

Tervezési problémák: szabad, vagy alacsonyan terhelt munkaállomások alkalmazása mint extra számítási kapacitás olyan esetekre, amikor a felhasználónak nincs elég számítási kapacitása.

Terhelés csökkentése az elosztott rendszeren belül:

1. **Processzor pool modell:**
   - a processzor pool számítógépek csoportja, ahol mindegyik saját hálózati kapcsolattal rendelkezik.

2. **Processzorok osztott memóriával:**
    - a folyamatok interprocessz kommunikációval kommunkálnak egymással osztot memóri felhasználásával.

6. Átlátszóság

Tervezési célok: annak elérése, hogy a felhasználó az egész elosztott rendszert egyetlen virtuális uniprocesszornak lássák, nem pedig számítógépek halmazának.

Tervezési nehézségek:

- egyszerűbb szoftverfejlesztés biztosítása
- inkrementális változások támogatása
- lehetőség biztosítása növekvő megbízhatóság támogatására
- az elosztott rendszer egyszerűbb felhasználói (user model) modellje

7. Konzisztencia fenntartása

Tervezési célok: az elosztott rendszer teljesítményének optimalizálása

Tervezési nehézségek: a replikált információknak mindenhol egységesnek kell lennie:

- **módosítás konzisztencia**: több processz fér hozzá és módosítja a dátumot konkurensen. Megoldás: együttes hozzáférés kizárása azon adatoknál, amelyeket több processz is módosíthat.
- **replikáció konzisztencia:** egy adott forrásból származó adatokat replikálni kell több számítógépre és utólagosan módosítani többnél. Megoldás: minden módosítást minden többi gépre el kell küldeni.
- **cache konzisztencia:** adatok, amelyeket egy kliens „bekesselt", más kliens módosít. Megoldás: a cache kölcsönös hozzáférésének kizárása.
- **hibatűrés konzisztencia:** ha egy gép meghibásodik, akkor az összes vele együttműködő gép folytatja működését normálisan. Azonban ha ezen gépek működése függ a meghibásodott gép folyamatos működésétől, akkor ezen gépek is leállhatnak. Megoldás: olyan visszaállító eljárások, amelyek a hibás gépet egy ismert állapotba állítja vissza.
- **óra konzisztencia:** a számítógépek óráit csak a kommunikációs hálózaton keresztül lehet szinkronizálni üzenetküldéssel, azonban a hálózaton való üzenetküldés előre meghatározhatatlan késleltetéssel rendelkezik. Megoldás: logikai órát használnak az

események sorrendjének leírására, nem tőrödnek a fizikai óra idejével, amikor az esemény bekövetkezett.