Nemzeti Információs Infrastruktúra Fejlesztési Intézet


# Elosztott skálázható, kreatív erőforrás ütemező

**(követelmény-specifikáció és rendszerterv)**

Készítette
**Stefán Péter**
**Szalai Ferenc**
**Vitéz Gábor**

Magyar Informatikai Erőforráshálózat (Grid közmű) alapjai (MEGA) projekt
(NKFP OM-00262,00263,00264,00266,00268/2004)

NIIF Intézet, Budapest
2006. június 30.

# Tartalomjegyzék

# 1. Bevezetés

E dokumentumban szeretnénk összefoglalót adni arról, hogy a grid rendszerek egyik legfontosabb feladatával, az erőforrás-allokációval kapcsolatban milyen követelmények támaszthatók, másrészt hogyan építhető fel egy jól skálázható, viszonylag nagyszámú feladat és erőforrás összerendelésére is alkalmas, moduláris bróker rendszer. E dokumentum elsősorban grid fejlesztők számára készült.

A 2. Fejezet bemutatja az erőforrás bróker rendszerek fontosabb feladatát, illetve azt, hogy a jelenleg használatos grid rendszerek hogyan oldják meg ezt a feladatot. A 3. Fejezet bemutatja a legfőbb követelményeket, a 4. Fejezet egy általános keretrendszer tervét ismerteti, amely moduláris felépítése miatt alkalmas tetszőleges interfészek, illetve tetszőleges döntéshozó modulok befogadására. Az IP-alapú forgalomirányítás mintájára az 5. Fejezetben bemutatunk egy lehetséges elosztott döntési, és hálós információs rendszeren alapuló hierarchikus modellt. A 6. Fejezetben bemutatjuk azt, hogy miként lehet a grid rendszereket hálózati algebrával (a hálózati struktúrák vizsgálatával) elemezni, illetve, hogy hogyan lehet struktúrált módon felépíteni.

# 2. Erőforrás bróker rendszerek

## *2.1. Feladata*

Az erőforrás bróker legfontosabb feladata, hogy a grid rendszerekben feladott feladatokat a számukra legmegfelelőbb erőforráshoz rendelje. A hozzárendelés „attribútum=érték" attribútumok alapján történik, azaz mind az erőforrások, mind a feladatok attribútumokat hirdetnek magukról: az erőforrás attribútumai a „kínálatot", a feladat attribútumai a „keresletet" reprezentálják. A bróker a rendelkezésére álló attribútum-információ alapján megkeresi a feladat számára legmegfelelőbb erőforrást.

Az attribútumok alapvetően kétféle csoportba oszthatók:

- szigorú attribútumok,
- optimalitást jelölő attribútumok.

A szigorú attribútumok azok, amelyeket a brókernek mindenképp be kell tartania, azaz olyan erőforrást kell keresnie, amelyen minden szigorú attribútum teljesül. Az optimalitást jelölő attribútumoknak nem kell feltétlenül teljesülni, ugyanakkor, ha egy erőforrás betartja ezen attribútumok többségét, az a feladat futása szempontjából előnyös lehet. Ilyenek például a geográfiai-, illetve adatközelséget kérő, illetve kínáló attribútumok, vagy ilyenek lehetnek a gyorsabb processzorok, illetve egy speciális optimált matematikai szoftver-könyvtárat biztosító erőforrás jelenléte.

Szintén az erőforrás bróker feladata az adott feladat ütemezése (scheduling) a kiválasztott erőforráson. Az ütemezés annak klasszikus definíciója szerint nem más, mint egy adott feladat, adott erőforráshoz rendelése egy adott időintervallumban. Grid rendszerek esetén az időszelet vége nem mindig ismert, mivel a feladat egy várakozási listára (Pending Queue, PQ) kerül, amelyről a processzorra kerülés, illetve a befejeződés nem a brókertől, hanem helyi erőforrás menedzsertől (Local Resource Management System, LRMS) függ.

Általában, különösen a „push modellt" használó brókerek külön alrendszere foglalkozik a feladat erőforrásra történő juttatásával (dispatching).

## *2.2. Ütemezési szempontok*

E fejezetben összefoglaljuk azokat a legfontosabb szempontokat, kérdéseket amelyekkel egy erőforrás bróker implementációja során szembesülünk.

### 2.2.1. Megszakíthatóság

A Grid feladatok általában nem megszakíthatók, vagyis ha egyszer egy feladat processzor erőforráshoz jutott, akkor azt onnan általában nem tudjuk úgy eltávolítani, hogy az az eltávolítás, és újraütemezés során a megszakítás pillanatbeli állapotától kezdve folytassa tovább a futását. A feladat állapotának lementését, illetve onnan való újraindítást nevezik checkpointing/restart (CKPT) mechanizmusnak [1]. Vannak olyan feladatok, amelyek képesek környezetük állapotát elmenteni, akár alkalmazás szinten, akár szoftverkönyvtár-szinten, a felhasználó számára transzparens módon, ez azonban nem általánosan feltételezhető tulajdonság. (Az, hogy egy feladat újraindítható-e lehet annak egy „attribútum=érték“ formában kifejezett tulajdonsága.)

Annak, hogy egy feladat nem általánosságban megszakítható, az a következménye, hogy az erőforrás brókernek nem kell szigorúan támogatnia a preemptív ütemezést, mivel helyi erőforrás menedzser kezeli le az ilyen jellegű megszakításokat, ugyanakkor, ha egy feladat futása nem fejezhető be azon az erőforráson, amelyre az eredetileg került, akkor szükség lehet annak grid-szintű újraütemezésére, új erőforráshoz történő allokálására. A grid ütemezőnek tehát lehetővé kell tennie azt, hogy egy már megszakított, és grid szinten újraütemezett feladat képes legyen egy másik erőforráson újra elindulni.

A feladatok megszakíthatósága, illetve a feladat újraütemezése függhet attól is, hogy az milyen architektúrán elérhető, milyen architektúrán futott a megszakítás pillanatáig, illetve az esetleges bináris checkpoint állomány (általában valamilyen memória-lenyomat) milyen architektúráról, illetve milyen operációs rendszerről származik.

### 2.2.2. „Pull“ és „Push“ modellek

Az erőforrás bróker rendszereket megkülönböztethetjük az alapján, hogy ki az aki kezdeményezi a feladat ütemezését. Ha az erőforrás „kéri“ a feladatot az erőforrás brókertől, azt nevezzük „pull“ modellnek. Ilyen modellt használ például a BOINC [2] szoftverrendszer, amely web protokollon keresztük kér végrehajtható feladatot az „ütemezőtől“. E megoldás nagy előnye az, ha az erőforrás egy védett hálózaton található, amelyről kifelé el lehet érni a hálózatot, de a befelé irányuló forgalom szűrve van (tipikusan NAT hálózat, állapottartó tűzfalak), akkor is tudunk feladatokat erőforráshoz rendelni.

Ha az erőforrás bróker átküldi a feladatot az erőforrásnak, azaz a bróker kezdeményezi a feladat végrehajtását, azt nevezzük „push“ modellnek. Ilyen megoldást használ az EGEE [3], illetve a ClusterGrid infrastruktúra [4] rendszer. E megoldás előnye, hogy a feladat attribútumokat, az azokban megfogalmazott követelményeket jobban figyelembe lehet venni a feladat kiosztásakor.

### 2.2.3. Ütemezési stratégiák

Illeszkedve az előbb említett kétféle modellre a bróker alkalmazhat mohó, vagy lusta ütemezési stratégiát. A mohó stratégia igyekszik egy feladatot minél előbb erőforráshoz juttatni, így egy feladat futtathatóságát több erőforrás viszonylatában vizsgálja, vagyis versenyezteti az erőforrásokat. Lusta stratégia alkalmazása esetén a

bróker megvárja amíg egy olyan erőforrás nem jelentkezik, amely a feladat futtatása szempontjából (még az optimalitást jelölő attribútumok szempontjából is) tökéletes, és ahhoz rendeli a feladatot. E stratégia egy erőforrást vizsgál több feladat viszonyában, azaz a feladatokat versenyezteti meg az erőforráson.

### 2.2.4. Ütemezési architektúrák

Napjaink grid rendszereiben alapvetően kétféle ütemezési architektúrát alkalmaznak:

- bróker alapú megoldást, és
- ágens alapú megoldást.

A kétféle architektúra között alapvetően a döntéshez szükséges, rendelkezésre álló információ „frissességében", illetve „érvényességében", valamint a döntéshozók számában van fő különbség.

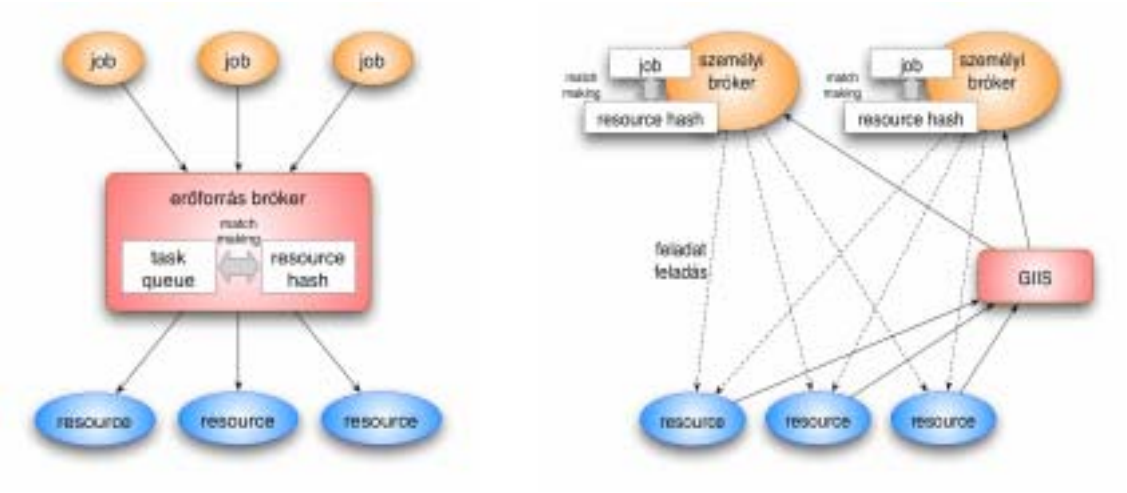
Bróker rendszerek esetén egy központi elem van, az *erőforrás bróker*, amely begyűjti az erőforrásokról származó erőforrás-információt, majd ezeket vagy helyben tárolja, vagy szintetizálja számára kényelmesen feldolgozható (kereshető) adathalmazzá (például láncolt listákra helyezi, vagy hash-eli). Mivel az információ a döntés bemeneti adata, az akár csak olvasható üzemmódban is tárolható. A bróker szintén tárol feladat-információt az összes feladatról, amelyet rajta keresztül ütemeztek[1]. A feladat-információt szintén valamilyen belső adatreprezentációban tárolják, például hash listákat alkalmaznak. A bróker döntéshozó modulja e két listát vizsgálja meg periodikusan, és az egyező attribútumok alapján összerendeli a feladatokat az erőforrásokkal.

A bróker által tárolt erőforrás-információnak van egy frissességi jellemzője, amely megmutatja, hogy mennyire fedi az adott információ a valóságot. Előfordulhat ugyanis, hogy a brókernél tárolt információ már nem érvényes, ellenben az ezek alapján hoz meg egy döntést. Ekkor az erőforrás rendszerint elutasítja a feladatot. (Bizonyos grid megoldások esetén létezik tárgyalási protokoll, amely segítségével a bróker, illetve az erőforrás „megbeszéli" egymással a feladat átadását-átvételét.) Brókert használnak az EGEE [3], illetve a ClusterGrid [4] grid rendszerek.

E problémát, valamint azt, hogy éles környezetben az egyetlen bróker komoly szűk keresztmetszetté tud válni, próbálja orvosolni az ágens alapú megközelítés. Ennek az a lényege, hogy maga a felhasználói interfész egyben egy könnyű (lightweight) bróker, amely mindig friss erőforrás információból dolgozik, azaz csak referenciákat használ, illetve tárol, és közvetlenül az erőforrástól kér le on-line információt. Az is jelentősen egyszerűsíti a feladatát, hogy nagyságrendileg csökken az erőforrás-feladat összehasonlítások száma, mivel egy kliensen, a grid rendszer összes feladott feladatához képest, viszonylag alacsony számú feladatot adnak fel. Cserébe persze több ilyen bróker van, praktikusan annyi, ahány felhasználó. Bár az összehasonlítási feladat dimenziószámának redukálása csökkenti az egy brókerre jutó terhelést a kereslet-kínálat összehasonlításban, a több „személyes bróker" (Personal Broker, PB) megjelenése miatt nagyobb arányú lehet a kölcsönös kizárások, valamint a hamis koallokációs ütemezések száma. (Ez olyan ütemezés, amikor az adott erőforrásra, egyidőben valamelyik másik PB is már ütemezett feladatot, kizárva ezzel épp az mi feladatunkat az erőforrásról.)

Ilyen, egyszerű algoritmusokkal megvalósítható megoldást használ a Nordugrid ARC szoftvere, amelyről bővebb információ található a [5] portálon. A bróker és az ágens-alapú megközelítéseket mutatja be a 2.1. ábra.

[1] Klasszikus grid rendszereknél lehetséges az, hogy a felhasználó a feladatát GRAM protokoll segítségével közvetlenül egy erőforrásra küldi, amiről az erőforrás bróker nem tud.




2.1. ábra: A bróker alapú és az ágens alapú ütemezési architektúrák.

### 2.2.5. Erőforrás információ frissítési stratégiák

Az erőforrás-információ frissítési algoritmusok, a feladat kiosztási megoldásokhoz hasonlóan szintén „push" és „pull" jellegűek lehetnek. „Push" modellről beszélünk akkor, ha az erőforrás vagy periodikusan, vagy valamilyen belső állapotváltozás után küld információt a brókereknek.

A „pull" (vagy „poll") modell esetén az erőforrás bróker próbálja lekérdezni az erőforrást annak pillanatnyi állapotáról.

## 2.3. Skálázhatóság fogalma

Az erőforrás bróker skálázhatóságán azt értjük, hogy mennyire változik meg annak teljesítménye, ha a feladatok és/vagy az erőforrások száma növekszik. Ez leginkább egy háromdimenziós grafikonon szemléltethető, amely két független változója az erőforrás brókerbe feladott feladatok száma, illetve a bróker által látható erőforrások száma, függő változója pedig valamilyen teljesítmény-paraméter, például időegység alatt összerendelt feladat-erőforrás párok száma.

A feladott feladatok száma adottság, nagyobbrészt attól függ mennyien és pontosan mire használják a grid rendszert. A feladatokat osztályokba sorolhatjuk a futtatási jellemzőik alapján:

- paraméter feltérképezési alkalmazások, amelyek ugyanazt a binárist futtatják, egymástól függetlenül, más-más input állományokkal;
- párhuzamos, elosztott memóriát (distributed memory) használó feladatok, amelyek különálló, egymással hálózati protokollon keresztül kommunikáló komponensekből állnak;
- párhuzamos, megosztott memóriát (shared memory) használó feladatok, amelyek különálló, párhuzamosan futó folyamatokból állnak, amelyek egy közös memóriaterületet használnak;
- szekvenciális alkalmazások.

E feladatosztályok mindegyikére más-más tipikus feladatszám paraméter jellemző. Általánosságban ezek eloszlása, illetve a felhasználók lehetséges száma alapján meg lehet becsülni a feladatok számát.

Az erőforrások mennyiségének becslése bonyolultabb: egyrészt függ attól, hogy mekkorának definiálunk egy cluster erőforrást, másrészt attól, hogy milyen jellegű erőforrásokat vonunk be a grid rendszerbe. Ez alapján beszélhetünk:

- nagy granularitású erőforrásokról (szuperszámítógépek, nagy clusterek, storage alrendszerek, speciális eszközök),
- közepes granularitású erőforrásokról (kis, és közepes méretű clusterek), illetve
- kis granularitású erőforrásokról (desktop-ok, PC-k, stb.)

Itt a granularitás azt jelenti, hogy az adott erőforrás száma hogyan viszonyul a grid rendszerben található összes erőforrás számához. Nagy granularitású erőforrás viszonylag kevés van a rendszerben, de egyenként nagyobb processzorszámot, nagyobb tárolókapacitást jelentenek, míg a kicsiből relatíve sok van, és egyenként viszonylag kis kapacitást nyújtanak. Kapacitásuktól függetlenül azonban ezek közel egyforma méretű bejegyzéseket jelentenek az információs rendszerben.

# 3. Követelmények az erőforrás brókerrel szemben

Az erőforrás brókerrel szemben támasztott követelményeket az alábbiakban foglaljuk össze:

- Skálázhatóság: Az erőforrások számának növekedésével nem romlik az időegység alatt meghozott döntések száma.
- Portabilitás: Az erőforrás bróker tetszőleges operációs rendszer és hardver környezetben képes működni.
- Szabványosság: Egyrészt a belső kommunikáció, másrészt, és ez a fontosabb, a külső kommunikáció szabványos megoldásokon, grid ajánlásokon alapul. Szabványos, vagy széles körben elfogadott adatreprezentációt, illetve kommunikációs protokollokat kell használni, mint például az OGSA-BES [6], illetve a JSDL [7] ajánlásokat.
- Konvertibilitás: Nem csak a JSDL-ben leírt feladatokat tudunk értelmezni, hanem más Condor-CLASSAD [12], RSL [13], illetve xRSL [14] leírásokat is tudunk konvertálni, akár egy speciális szolgáltatás segítségével JSDL reprezentációra. Képes legyen más rendszerekkel való együttműködésre is.
- Modularitás: Az erőforrás bróker fel van készítve funkcionális bővíthetőségre, azaz arra, hogy egy új kommunikációs protokoll használata, vagy speciális ütemezési algoritmus bevezetése esetén ne kelljen a keretrendszeren változtatni, az új funkció egy új modul megírásával elérhetővé válik.
- Egyszerűség, áttekinthetőség: Az erőforrás bróker szerkezete egyszerű, világos, áttekinthető. Ne tartalmazzon felesleges alrendszereket, illetve ki-nem-használt függvénykönyvtárakat.
- Biztonság: A bróker, mint szolgáltatás biztonságos módon üzemeltethető, védhető külső támadások ellen.
- Robusztusság: Maga a szolgáltatás funkciója rendelkezik valamilyen szintű redundanciával. E redundancia nem feltétlenül grid rendszeren belüli, lehet alacsonyabb szintű megoldás is, például nagy rendelkezésreállású fürtök.

- Rendelkezésre állás: A bróker szolgáltatás rendelkezésre állási ideje minél magasabb, a bróker bármikor képes külső erőforrás, vagy feladat információt fogadni.
- Hatékonyság: A cserélt információt minél tömörebben fogadja, illetve adja át.
- Alacsony fenntarthatósági költségek: A bróker minél kevesebb humán erőforrással üzemeltethető, hiba esetén minimális beavatkozással újra üzemszerű állapotba hozható, esetleg magától javul meg.
- Funkcionális bővíthetőség: Ez részben összefügg a modularitással, ugyanakkor azt is jelöli, hogy a nem-moduláris komponensek is viszonylag alacsony költséggel modulárissá tehetők.

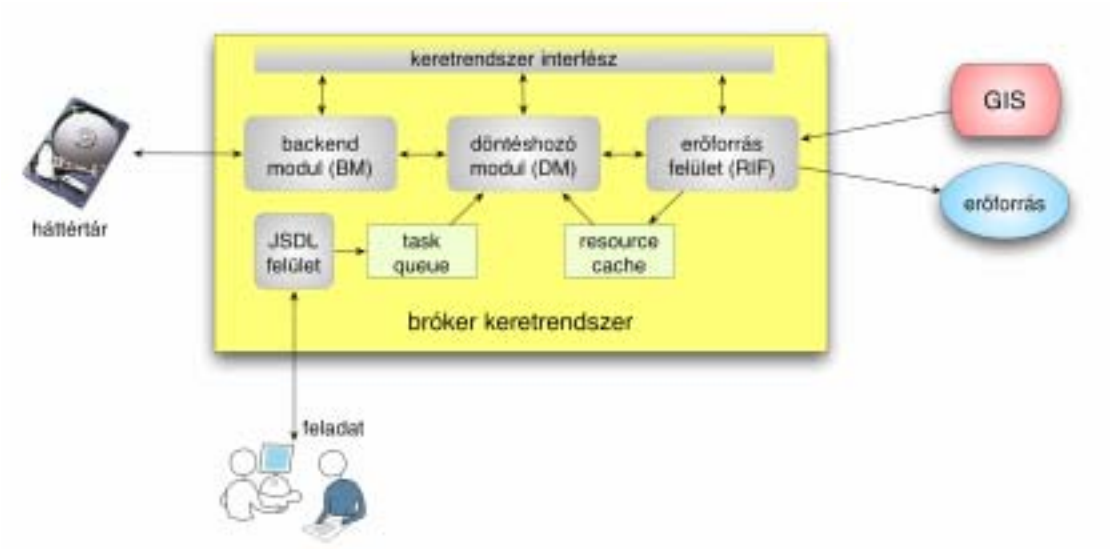
# 4. Az erőforrás-bróker ágens alapú modellje, rendszer-felépítés

Az előbb megfogalmazott követelményeknek leginkább egy olyan rendszer felel meg, amely elsősorban egy keretszerkezet biztosításával kellően rugalmas, lehetővé teszi különböző modulok betöltését, és maga a keretrendszer rendkívül egyszerű, több platformra adaptálható, és kevés hibalehetőséget tartalmaz. A brókernek rendelkeznie kell minimális intelligenciával, ami a konkrét esetben önálló döntéshozási képességet jelent. Grid feladat-erőforrás allokáció esetén az önálló döntés azt jelenti, hogy a bróker képes az összerendelés döntést a külső feltételek megváltozásának megfelelően önállóan módosítani. Például a rendelkezésre álló erőforrások számától, jellegétől függően tudja önmagát hangolni a mohó és a lusta ütemezési stratégiák között.

## *4.1. Bróker keretrendszer*

A bróker az általa kinyert információt tekinti belső állapottérnek, és a döntéseit pedig akcióknak. Minden egyes döntésről visszacsatolás érkezhet a brókerhez a döntés „jóságáról", azaz arról, hogy az adott feladat-erőforrás összerendeléssel mennyire volt „megelégedve" a felhasználó, illetve az erőforrás. Ezeket az információkat a bróker feldolgozhatja, és felhasználhatja egy későbbi döntése során, mint tapasztalatot.

Az erőforrás bróker nagymértékben függ a grid információs rendszer felépítésétől, illetve az abban tárolt információ kinyerésének és továbbításának módjától. Ezért javasoljuk a 4.1. ábrán látható általános keretrendszer kialakítását.




4.1. ábra: Erőforrás-bróker rendszervázlat

Maga a rendszer több komponensből, illetve több modulból áll.

A keretrendszer (Broker Frame, BF) egy olyan általános rendszer, amely a bróker alacsony-szintű kommunikációs folyamatait kezeli. Alacsony szintű kommunikáción értjuk az adathálózati, a web, illetve a web kérésekbe becsomagolt SOAP csomagok továbbítását. E rendszer gondoskodik arról is, hogy a grid erőforrás-bróker, mint web-szolgáltatás egyáltalán megjelenhessen az adathálózaton.

Az erőforrás-interfész modul (vagy modulok, Resource Interface, RIF) gondoskodnak az erőforrás-információ kinyeréséről, illetve annak belső tárolásáról. Az erőforrás információ egy belső táblába kerül, amelyet a bróker döntéshozó modulja kényelmesen el tud érni. Az erőforrás listán cache-elt, csak olvasható erőforrás-információ található. A modul másik funkciója az, hogy a felhasználó nevében eljárva feladatot tudjon átadni erőforrásoknak, például számítási csomópontoknak (dispatching). Magát a feladatot a célszerű nem itt tárolni, hanem a grid rendszer egy másik pontján, például egy tárolási csomóponton, itt csak egy hivatkozást tárolunk. Egy információs rendszerrel kommunikáló, illetve feladat-átadó alrendszer lehet egy RIF modul, külön modul foglalkozhat Globus 4.x, ARC, illetve gLite rendszerekkel való kapcsolattartással.

A felhasználó a feladatának leírását egy külön alrendszeren, a feladat-feladó (Job Submission, JS) alrendszeren keresztül adhatja át a brókernek. E leírást a bróker feldolgozza, majd egy belső feladat-soron (Task Queue, TQ) helyezi el. A TQ minden az adott brókernél megfordult, még le nem zárt feladatot tartalmazza. Ha a felhasználó brókeren keresztül futtat feladatot, akkor ez az a feladat-nyilvántartás, amelyből legutoljára kell a feladat-információnak a feladat végeztével, az eredmények kinyerése után törlődni. Célszerű, ha ez az alrendszer valamilyen szabványos feladat-leírást használ, például a JSDL [7] nyelvet.

A döntéshozó modul (vagy modulok, Decision Maker, DM) gondoskodik a két fő listán tárolt igények és lehetőségek kombinálásáról. Mivel az ütemezési döntések tetszőlegesen bonyolultak is lehetnek, akár probléma-specifikus modulokat is használhatunk, amelyek speciális attribútumok alapján speciális döntésekez hoznak. A döntés eredményeképp a feladat bejegyzésébe bekerül a hozzá tartozó, javasolt erőforrás, amelyet a bróker a RIF alrendszerén keresztül megszólít, és megbeszéli vele azt, hogy a feladat végrehajtható-e. Ha igen, akkor a feladat-leírást átadja az erőforrásnak, ha nem, akkor új erőforrást keres.

Az összerendelésekhez kapcsolódó adatok permanens tárolására való a háttértár modul (Backend Modul, BM), amely az adatokat, adat-elemeket valamilyen perzisztens

formában tárolja. Ez legegyszerűbb esetben egy állományrendszer megfelelő jegyzékében történő struktúrált tárolás, de használhatunk akár SQL adatbázist is.

### *4.2. Tapasztalatok: Hogyan működjön a „jó“ bróker?*

A 4.1. részben egy általános keretrendszer tervét ismertettük, amelyet a moduljai tesznek működőképessé. Hogyan működjenek ezen modulok?

Célszerű a RIF modult úgy kialakítani, hogy mind a „pull“, mind a „push“ típusú információ és feladat-továbbítási módszert támogassa. Képes legyen fogadni külső erőforrások állapot-változását leíró információt, de képes legyen más erőforrásokat valamilyen formában lekérdezni. Ugyanez igaz a feladatokra: azaz képes legyen egy erőforrás tőle feladatot kérni, és a bróker képes legyen feladatot átadni.

Másik fontos RIF modulbeli elem, a tárgyalási protokoll. A bróker nem csak egy szuper-ütemező, aki a feladatokat visszacsatolás nélkül átnyújtja az erőforrásnak, hanem egy olyan intelligens döntéshozó, aki megbeszéli az erőforrással a feladat végrehajtását, és amennyiben az az általa korábban propagált információ ellenére nem tudja a feladatot fogani (mert például azóta megtelt feladatokkal), akkor az erőforrásnak módja van visszautasítani azt.

A döntéshozó modulnak támogatnia kell a korábban hivatkozott bróker, illetve ágens architektúrát egyaránt. Ezt lehet akár különböző modulokkal is biztosítani. Az is elképzelhető, hogy a döntéshozó egy teljesen új architektúrával működik, például kialakítható egy olyan bróker-láncolat, amelyben az egyes brókerek feladatot és erőforrás-közeli információt adnak/vesznek át más brókerektől.

A rendszert úgy kell elkészíteni, hogy abban több bróker is működhessen. Ha egy grid rendszerben feladott összes feladat, és a gridben megtalálható erőforrások Descartes-szorzatát vesszük, akkor a brókerekben található erőforrás-feladat diszjunkt összerendelések konjukciója adja meg a teljes szorzatot. Ezen a szinten nincs szükség bróker-szintű tartalékolásra, ezt célszerű egy szinttel alacsonyabban, valamilyen nagy rendelkezésre-állású megoldással biztosítani (redundáns adatbázis, forgalomelosztás).

## 5. Hierarchikus döntéshozó rendszerek

A jelenlegi megoldások elsősorban egyszintű döntéshozási mechanizmust alkalmaznak. Ez azt jelenti, hogy egyetlen entitás (csomópont) összegyűjti az összes feladat, és erőforrás információt, majd ezeket összevetve egy keresési algoritmussal találja meg az egymáshoz illeszkedő párokat. A gyakorlatban azonban mindkét információhalmaz túl nagyra tud növekedni ahhoz, hogy egyetlen csomóponton ezeket hatékonyan lehessen tárolni, illetve feldolgozni. Így a jelenlegi, egyszintű döntési mechanizmuson alapuló bróker megoldások komoly szűk keresztmetszetekké válhatnak.

A hierarchikus döntéshozás abban különbözik a jelenlegi döntési mechanizmusoktól, hogy a döntést nem egy lépésben, hanem egymás után következő egyre finomabb, és egyre aktuálisabb információk alapján hozzuk meg. A hierarchikus döntés történhet úgy, hogy előbb egy aggregált információ-halmaz alapján döntünk, majd a kezdeti „betájolás“ után, az információs rendszer számunkra érdekes részét részletesebb információ után kutatva lekérdezzük, így egyre közelebb jutunk a számunkra érdekes erőforráshoz.

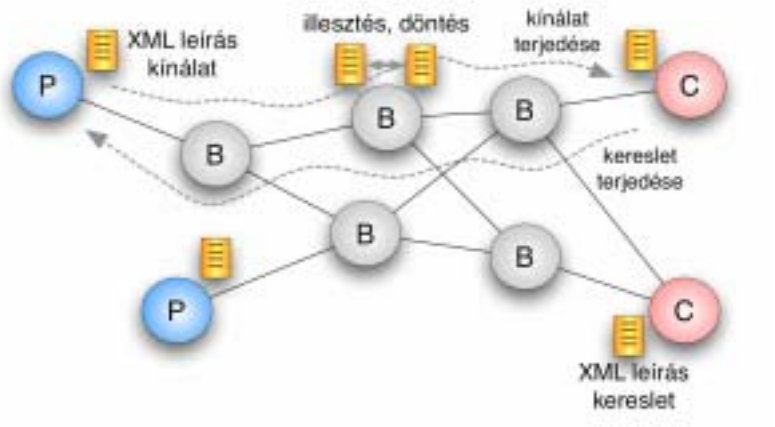
### *5.1. Tartalom-alapú forgalomirányítás*

A tartalom-alapú forgalomirányítás (Content Based Routing, CBR) [8][9] egy napjainkban igen népszerű kutatási terület, amelyet arra találtak ki, hogy az információ-források, illetve az információ-fogadók kölcsönösen megtalálják egymást egy route-olt adathálózaton. A tartalom-alapú forgalomirányítás, amely feltételezi azt, hogy az információ források, és információ fogadók között közbülső, kizárólag információ-továbbításra használt csomópontok is vannak, alapvetően kétféle implementációt támogat:

- az IP-alapú megoldásokat, amelyek elsősorban az IP-multicast legfontosabb korlátait próbálják orvosolni, ellátva az adatcsomagokat bitekre-képzett szemantikai információval, illetve
- az alkalmazási szintű megoldásokat, amelyek esetén egy, az IP-fölötti, attól független, peer-to-peer hálózat épül ki, és az információ termelők, illetve fogyasztók közbülső csomópontokon keresztül cserélnek információt.

Utóbbi esetben az alkalmazás nagyobb rugalmassággal ruházható fel, mint egy kernel-szintű, IP-csomagot feldolgozó modul, ugyanakkor kérdés annak hatékonysága.

A CBR alapötlete az, hogy az információ továbbítása nem a rendeltetési hely alapján történik, hanem a tartalom figyelembevételével. Az információ-források leírják valamilyen formában azt, hogy milyen tartalmú adatokat biztosítanak, például XML nyelven, majd az így reprezentált tartalmat hirdetik meg a hálózaton. (Normál IP-routing esetén az erőforrások azt hirdetik magukról, hogy hozzájuk milyen IP-című alhálózatok tartoznak.) Az információ fogadók, szintén kifejezik érdeklődési profiljukat egy, az előbbihez hasonló reprezentációs formában. Ezeket a fogadók szintén meghirdetik. A közbülső forgalomirányító elemek összevetik a keresett és a kínált információ-profilt, és azt csak annak a szomszédnak továbbítják, amely felől „érdeklődés érkezett" hasonló jellegű tartalom iránt (5.1. ábra). Így nincs egy központi adattároló elem, információs rendszer az igények, lehetőségek fellelhetőségére, hiszen egy fogyasztóhoz, elméletileg csak olyan információ juthat el, amely iránt ő érdeklődését fejezte ki, azaz feliratkozott rá.




5.1. ábra: Forgalomirányítás tartalom alapján. A közbülső csomópontok végzik a kereslet és a kínálat összevezését és szűrését.

A CBR megoldások lehetnek kétfázisúak, vagy egyfázisúak. Előbbi esetben elkülöníthető a források által végzett információ-telítési fázis, illetve a kliensek által válaszul adott feliratkozási fázis, utóbbi esetben minden egyszerre történik, azaz a kliens jelzi a

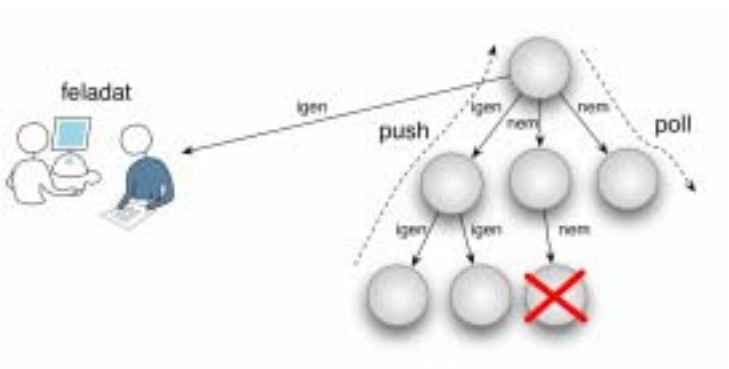
szomszédjának, hogy mire van szüksége, aki ezt az információt továbbíthatja, illetve a forrás jelzi a szomszédjának, hogy mit kínál, aki ezt az információt továbbítja.

### *5.2. Hálós szerkezetű információs rendszer, hierarchikus döntések*

A grid erőforrás brókerek is használhatnak hasonló alapelveket az információ szűrésére és továbbítására. A 4. fejezetben javasolt megoldás, moduláris szerkezete miatt, képes arra, hogy tetszőleges információtovábbítási módot megvalósítson a keretrendszer megváltoztatása nélkül. A hálós szerkezetű információs rendszer a tartalom-alapú forgalomirányítás alapelveinek grid környezetbe való adaptálását jelenti:

- Az erőforrás, illetve a feladat attribútumokat XML nyelven, „attribútum=érték" formában írjuk le, ahol az attribútumoknak jól definiált értékkészlete lehet.
- Ezeket az információkat SOAP felületen keresztül cserélik ki egymással olyan köztes csomópontok (erőforrások), ahol az információs rendszer helyi információt, és távolról begyűjtött információt is tárol. Azt, hogy egy bejegyzés helyi eredetű-e, vagy távoli, minden erőforrás megcímkézi.
- A helyi vagy távoli erőforrás információk, valamint a feladat által specifikált erőforrás-követelmények alapján egy erőforrás eldöntheti, hogy a hozzá érkező feladat helyben futtatható-e, vagy nem. Ha nem, akkor átadja a feladatot annak az erőforrás-szomszédjának, amelyikről úgy gondolja, hogy a legközelebb áll ahhoz, amely annak az igényeit kielégíti. Ezzel egyidőben finomítja saját belső állapot-reprezentációját, úgy hogy megerősíti a kereslet, illetve a kínálat összerendelést.
- Amennyiben a döntéshozó bizonytalan saját döntésében, azaz nincs olyan lehetőség, amely egyértelműen az adott specifikációnak megfelel, felderítő-útra küldheti a feladatot valamelyik szomszédja felé. Hogy konkrétan melyik felé, esetleg melyek felé, arról akár determinisztikus, akár szochaszikus úton is hozhat döntést, utóbbi esetben véletlenül választva a szomszédok közül egy meghatározott valószínűség-eloszlás alapján.
- Felderítő úton a feladat rendelkezhet egy maximális távolság paraméterrel, amely minden egyes átadásnál csökken, és ha elfogy, akkor a rendszer azt jelzi vissza, hogy nincs számára a belátható körben megfelelő erőforrás. (Erre a felhasználó, vagy a helyette eljáró kiinduló bróker megpróbálhatja megnövelt maximális távolság paraméterrel újraütemezni a feladatot.)
- Sikeres ütemezés esetén az elfogadó erőforrás egy, az ütemezés sikerességét reprezentáló jelet küldhet vissza a feladónak, a feladat-továbbítási láncon keresztül, amelyet a továbbításban résztvevő közbülső brókerek feldolgozhatnak, saját belső összerendelésüket, és forgalomirányítási táblázatukat megerősítve.

Mindez a feladatot feladó kliens számára az 5.2. ábrán látható fastruktúrában reprezentálható. A hierarchia a klienstől távolodva azokat az erőforrásokat jelöli, amelyekre a feladat egyáltalán elküldhető. A hierarchiában lefelé haladva, a feladótól távolodva, az erőforrások jelzik azt, hogy képesek-e a megadott paraméterekkel rendelkező feladatokat futtatni, a továbbításban részt vevő brókerek pedig aggregálják ezt az információt, legegyszerűbb esetben veszik a válaszok unióját.




5.2. ábra: Az elosztott döntéshozatal a kliens (feladatot feladó gép) szemszögéből.

Az így kialakított rendszer figyelembe tudja venni a topológia megváltozását, illetve az egyes erőforrások paramétereinek, attribútumainak változását is. A működési alapelv kiegészíthető még a korábban említett tárgyalási protokollal is.

### *5.3. Az információ-továbbító bróker működési elve*

A javasolt adatmodell esetén nem foglalkozunk hatékonysági kérdésekkel. A jelenleg javasolt adatstruktúráknak léteznek algoritmikusan feljavított reprezentációi, amelyekkel azok egy konkrét esetben akár kicserélhetők.

- A feladat illetve erőforrás listák esetén javasolt az indexelt lista alkalmazása. A lista egy-egy eleme egy-egy XML dokumentum, amely leírja a feladat, illetve az erőforrás attribútumokat.
- Az erőforrás-feladat összerendelést célszerű a mindkét oldalon nyilvántartani, azaz minden egyes beérkező feladathoz hozzá kell rendelni egy vagy több erőforrást, amelyre a feladatot továbbküldtük, és minden erőforráshoz be kell regisztrálni, hogy azon milyen jellegű feladatok futnak, futottak. Ezáltal az erőforrás-igény, és erőforrás-kínálat paraméterek mindkét irányban gyorsan továbbíthatók. A hozzárendelésnek lehet súlya is, amely preferenciákat is kijelölhet. Az erőforrásokat és a feladatokat URL-el hivatkozzuk meg. A kettős nyilvántartást konzisztens állapotban kell tartani, azaz a műveleteknek valamennyire atominak kell lenni.
- A listák, illetve a rajtuk hivatkozott összerendelések a „routing táblák".
- Ha egy kínálat bejegyzésre vagy kereslet típusra még nincs bejegyzés, akkor a továbbító az összes szomszédjának továbbküldi a feladatot, kivéve azt, amelyről a feladat érkezett. Ha a feladat helyben már nyilván van tartva az azt, jelenti, hogy itt már járt, azaz innen nem is küldjük tovább.
- Ha egy feladat túl messze távolodott a kiindulási ponttól, és lejárt a maximális átadási szám paramétere (amely a hálózat default paramétere is lehet), akkor a feladatot eldobjuk, és annak a csomópontnak, aki átadta visszaadunk egy alacsony preferenciát kifejező számot (büntetés), illetve egy hibakódot, aki ezt a kódot tovább propagálja visszafelé.
- A preferencia javasolt intervalluma: [0,255], ahol az intervallum alja alacsony, a teteje magas preferenciát jelöl.
- A visszafelé történő negatív propagáció során a csomópontok gyengítik az adott profil, illetve erőforrás-leírás összerendelésüket.

- Ha egy erőforrás elfogadja a feladatot, akkor magas preferenciával regisztrálja magát a feladat lista elemén, a feladat profilja pedig az erőforrás hasonló adatstruktúrájában. Az erőforrás minden csatornáján hirdeti ezt a feladat-típust (elárasztás), vagy csak korlátozott mértékben terjeszti (korlátozott elárasztás).
- Az feladat-lista (Task Queue, TQ) feloszthető aktív és inaktív feladatokra. Az inaktív feladatok egy feladat profilt jelölnek, ezzel jelképezve azt, hogy egy adott erőforrás ilyen típusú feladatokat képes kezelni.
- Az erőforrásokat, illetve a feladatokat globálisan egyedi azonosítókkal lehet reprezentálni.
- Mintaillesztésnél egy adott feladat profilját összevetjük a TQ-ban található inaktív feladatok profiljával, ha egyezés van, akkor az ilyen típusú bejegyzéseket futtatni képes erőforrás adott. Ha nincs illeszkedés, akkor az erőforrásokkal vetjük össze. Az erőforrás visszacsatolást indít el a kliens felé, hogy a feladat ütemezhető, aki közvetlenül is megtárgyalja az erőforrással azt, hogy a feladat valóban futtatható-e.
- Ha nem, akkor visszafelé történő negatív propagáció következik, ha a feladat futtatható, akkor az erőforrás afelé a szomszédja felé, akitől a feladatot kapta elindít egy összerendelés-erősítési láncolatot a kliensig.

A listákon az XML dokumentum hozzáadás, elvétel, illetve a súlybejegyzések létrehozása, módosítása, törlése operációkat kell értelmezni. A RIF modul az erőforrás-, feladat-, forgalomszabályozó-információkat SOAP felületen keresztül, web fölött továbbítja.
Az így kialakított erőforrás bróker rendszer támogatja mind a poll, mind a push típusú adat- és feladat-továbbítást, és kellően rugalmas ahhoz, hogy a peer-to-peer alapon felépített hálózatok topológiáját úgy térképezze fel, hogy a feladat a számára minél megfelelőbb erőforrást találja meg.

# 6. A grid rendszerek felépítésének tanulmányozása gráfelmélet segítségével

## *6.1. Hálózati struktúrák*

A 2002-es év fordulópont volt a grid rendszerek kifejlesztésében, hiszen a világ ettől kezdve kezdett el foglalkozni grid infrastruktúrák létrehozásával. Korábban leginkább teszt-környezetek, tanulmányok, pilot rendszerek léteztek, ettől kezdve azonban megjelentek a professzionálisan felépített, a grid szervezeti és szerkezeti struktúrájával is foglalkozó, geográfiailag nagy kiterjedésű grid rendszerek. Az infrastruktúra-építés egyik fő ismérve az volt, hogy különböző rendszer elemeket hálózati struktúrális megfontolások nélkül kapcsolták össze, így azokban számos túlterhelt csomópont keletkezett, lerontva ezzel a grid teljesítményét. A tapasztalati úton felépített hálózatokat nevezzük mérnöki modelleknek.

A hálózat struktúrális felépítésével, illetve az ebből származó előnyökkel, hátrányokkal ma egy speciális, hazai vonatkozásokat is bőven tartalmazó terület, a skálázható hálózatok tudományterülete foglalkozik [17]. E terület alapvetően a hálózatot, mint gráfot modellezi, és megpróbál a kapcsolatok természetéből, a gráf csomópontjainak fokszámából, azok teljes gráfon belüli eloszlásából következtetni a hálózat főbb jellemzőire. E terület eddig rávilágított arra, hogy több, egymástól független hálózat pusztán e struktúráltságát véve alapul egymáshoz hasonló módon viselkedik.

A hálózatok felépítésének tudománya nemcsak elemzi korunk főbb hálózatait, hanem útmutatást is ad, hogy bizonyos feltételeknek milyen felépítésű hálózatok felelnek meg leginkább, illetve hogyan hozzunk létre ilyen hálózatokat. Ezek alapján modelleket építhetünk fel arra, hogy egy megadott követelményeket támasztó hálózat milyen struktúrában bővüljön.

## *6.2. Grid hálózatok*

Grid rendszerek esetén több hálózat is létezik: külön hálózatot alkotnak a számítási erőforrások, az erőforrás brókerek, az adattároló csomópontok, az információs rendszerek, stb. Ezek mindegyike akár egy független hálózatnak is tekinthető, saját csomópontokkal (azonosított csomópontokkal), illetve élekkel, kapcsolatokkal e csomópontok között. A kapcsolatok szintén többfélék lehetnek, például a bróker többféle protokollt használva többféle célú kapcsolatot tud kezdeményezni brókertársaival: képes információt cserélni, de képes akár egy feladatot, annak leírását is átadni. Ezek alapján megkülönböztethetjük az egyes grid szolgáltatások egymáshoz kapcsolódásának hálózatát.

## *6.3. Milyen az ideális hálózat?*

Az ideális hálózat olyan, hogy a hálózat elemei által végzett feladatokra valamilyen teljesítmény-kritériumot fogalmazunk meg, és e kritériumnak keressük a szélső értékeit. Sajnos itt az egzakt, zárt-alakú megoldások szóba sem jönnek, így elsősorban arra koncentrálunk, hogy a hálózat felépítésére jellemző paramétereket fogalmazunk meg, és azt vizsgáljuk, hogy miként változik az általunk megfogalmazott teljesítmény-jellemző, a hálózati paraméterek hangolásának függvényében. A legegyszerűbb esetet feltételezve, a hálózatnak csak egy paramétere van, a scale-free hálózat exponenciális kitevője, amely egy új csomópont egyik régi csomóponthoz való kapcsolódásának sztochasztikus valószínűségeloszlását jelöli ki. A teljesítmény-kritériumunknak megfelelően keressük az optimális hálózati paramétereket.

## *6.4. Grid hálózatok*

### 6.4.1. GIS hálózat felépítése

Az információs rendszer (GIS) hálózat csomópontjai információ források (erőforrások), információ fogadók (kliensek), és információ továbbítók. Ezek között XML adatreprezentációjú, általában kölcsönös információcsere történik (kapcsolatok). Az információcsere nem minden esetben kölcsönös, gondoljunk például a NAT tűzfal mögötti erőforrásokra! E hálózat legfontosabb feladata az információ frissen tartása, így fontos optimum-kritérium, hogy a csomópontok lehetőleg alacsony áttétellel cseréljenek információt.

E kritériumnak leginkább egy olyan peer-to-peer hálózat felel meg, amely viszonylag alacsony számú, de mégis redundáns információ-továbbító csomópontot tartalmaz, melyek egyben gyűjtő-csomópontoknak is számítanak. Az erőforrások és kliensek átlagos távolságára megfogalmazható egy maximális átmérő paraméter, amely garantálja azt, hogy mindig friss információhoz jutunk. E paraméter ideális esetben mégis nagyobb, mint 3, hiszen maga az erőforrás is szűk keresztmetszetté válhat az információ továbbításában. Ez az előbbi, az információ-frissességgel ellentétesen ható kritérium, hiszen egy nagy központi továbbító csomópont sem nem terhelhető a végtelenségig, sem nem biztosít kellő mértékű redundanciát.

Magára a kliensre nem érdemes kritériumokat megfogalmazni, hiszen arról feltételezzuk, hogy kellően nagy számú kapcsolatot képes kezdeményezni, kellően nagy számú erőforrással, vagy információ-továbbítóval.

Az erőforrások azonban korlátosak abból a szempontból, hogy maximum hány kliensnek (vagy információ-továbbítónak) adnak tovább erőforrás-leírást. Itt megfogalmazhatunk egy maximális kapcsolódási szám paramétert.

### 6.4.2. A feladatkezelő hálózat felépítése

A feladatkezelő (Job Submission, JS) hálózat elsősorban erőforrásokból, erőforrás brókerekből, illetve tárolási csomópontokból áll. Ez utóbbiakra azért van szükség, hogy a feladathoz tartozó állományokat (bináris, szoftverkönyvtárak, licensz állományok, paraméter-állományok, forrás, stb.) a felhasználó egy jól meghatározott helyre tegye, és az erőforrás bróker innen másolja föl azt a számítási erőforrásra.

Itt a csomópontok az előbb említett entitások, a kapcsolódások alapvetően feladat, vagy feladat-leírás átadások, amelyek protokollja lehet web, vagy GridFTP alapú. E kapcsolódások alapvetően kétszintű pont-pont rendszerek, amelyekben az egyik pontot a kliensek (feladat feladók), a másik pontot az erőforrások jelentik. Mindezekhez kapcsolódnak a storage csomópontok, amelyek kapcsolódva az erőforrásokhoz, kliensekhez, egyben hub csomópontként is szolgálnak a hálózaton belül.

A legfontosabb teljesítmény-kritérium egyrészt a storage csomóponthoz kapcsolódó kliensek és erőforrások száma, másrészt az egy erőforráshoz kapcsolódó kliensek száma. A storage csomópontnak van egy maximális kiszolgáló-kapacitása, amely egyben limitációt is jelent a rendszerben.

#### 6.4.3. Adattároló rendszerek hálózat felépítése

A JS hálózatban található tárolóelem valójában nem egyetlen elem, hanem egy komplex rendszer belépési pontja. Az adattároló hálózat adattároló felületekből, adatkezelő, és meta-adat kezelő csomópontokból áll. E hálózat szerves részét képzi még az autozirációs rendszer elérési pontja.

E hálózat szerkezetét tekintve egy hierarchikus felépítésű rendszer, amelynek tetején az adattároló interfészek találhatók (Storage Manager, StM). Ehhez kapcsolódnak a viszonylag alacsony számú metaadat-tároló elem (Storage Catalog, SC), illetve a tényleges adattároló elem (Storage Controller, StC).

A rendszer a UNIX állományrendszer adattárolási szerkezetét másolja le, úgy, hogy egy hierarchikus felépítésű állományrendszert képzünk le adatokká és metaadatokká. A rendszer csomópontjai közötti kapcsolat XML-alapú web-es kommunikáció, illetve, kompatibilitási okok miatt, GridFTP. A legfontosabb struktúrális kritériumok:

- a metaadatbázis terhelhetőségének vannak felső korlátai;
- az adattároló csomópontok kapacitása véges;
- egy StC elemnek vannak sávszélességi korlátai.

Maga a metaadatbázis is alkothat egy komplex hálózatot, a jelenlegi implementációs javaslatban egy gyűrű topológiára vannak a metaadat-bejegyzések felfűzve. Más topológia alkalmazásával ezen adatbázis teljesítménye, redundanciája is vizsgálható.

### 6.4.4. Autentikációs, autorizációs, accounting (AAA) hálózat felépítése

### 6.4.5. Monitorozó, logoló, és számlázó hálózat felépítése

Monitorozó rendszeren itt erőforrás-állapot, illetve feladat-állapot megfigyelését értjük, és nem foglalkozunk a feladatok futásának nyomonkövetésével (feladat-monitorozás). A logoló rendszert a monitorozó rendszer részeként képzeljük el. E hálózat csomópontjai az erőforrások információs rendszere, valamint log csatornái, az ezeket az információkat agregáló gyűjtő csomópontok, illetve a megfigyelésre használt monitorozó csomópontok. Mindegyik típusú csomópontból tetszőlegesen nagy számú lehet a hálózatban. Az összeköttetések alapvetően web kommunikációk, általában egyirányú kapcsolatok.

E hálózat gyakorlatilag egy olyan rendszer, amelyen „átfolyik" az információ, a forrástól, a monitorozóig. A legfontosabb kritérium, hogy ez maximális átfolyási kapacitással történjen, azaz az információ minél gyorsabban, minél nagyobb kapacitásban jusson a feladótól a címzettig. Ehhez kell illeszkedni a hálózati struktúrának is. A hálózatnak illeszkednie kell geográfiai viszonyokhoz is, mivel az erőforrások általában szervezeti csoportokba is tartozhatnak.

### 6.4.6. Virtuális szervezetek hálózatának felépítése

A virtuális szervezetek jelenléte partícionálást jelent az eddigi hálózatokat reprezentáló gráfokban, azok diszjunkt részgráfokra történő szétdarabolása során keletkeznek. Itt a legfontosabb kritérium a gráf összefüggőségének kérdése, illetve az izolált erőforrások száma. Azt kell tehát vizsgálnunk, hogy a virtuális szervezetek létrehozásával történő feldarabolás nem növeli-e meg az izolált csomópontok számát, illetve nem szakítja-e el a virtuális szervezetet egy fontos erőforrástól (központi bróker, storage element).

### 6.4.7. Feladat futtató, lokális hálózatok

Maguk a párhuzamosan futó feladatok is alkothatnak hálózatot, mivel azok egyes folyamat-komponensei több csomóponton futhatnak [15]. Itt a csomópontok az MPI, vagy PVM azonosítóval jelölt programkomponensek, az összeköttetések pedig TCP/IP socket kapcsolatok, amelyen valamilyen, a párhuzamosításra használt könyvtárra jellemző protokollon cserélnek a csomópontok információt.

A gyakorlati tapasztalat azt mutatja, hogy e hálózatok két főbb topológiát használnak, a mesh, illetve a kétszintű hierarchikus (master-worker) topológiát, és a feladatok csak elhanyagolhatóan kis százaléka használ ettől eltérő megoldást. Így e hálózatokkal részletesebben nem foglalkozunk.

## 7. Összefoglalás

E dokumentumban röviden összefoglaltuk azokat a legfontosabb alapelveket, amelyekre egy grid erőforrás-bróker kialakítása során tekintettel kell lenni. Bemutattuk az általános szempontokat, illetve felvázoltuk a brókerrel szemben támasztott legfontosabb követelményeket. Bemutattunk egy keretrendszer modellt amely alapul szolgálhat a projektben vállalt erőforrás brókerének kialakításához. A modell kellően általános ahhoz, hogy tetszőleges döntéshozó modul, erőforrás-felület modul, illetve backend részévé válhasson.

Végül néhány gondolatban összefoglaltuk a grid felépítéséhez szükséges legfontosabb hálózati szempontokat, amelyben a grid rendszer egymáshoz kapcsolódó szolgáltatásait, komponenseit, mint struktúrált hálózatokat vizsgáltuk.

## Hivatkozások

[1] http://en.wikipedia.org/wiki/Checkpointing
[2] http://boinc.berkeley.edu/
[3] http://public.eu-egee.org/
[4] http://www.clustergrid.niif.hu/
[5] http://www.nordugrid.org/
[6] https://forge.gridforum.org/projects/ogsa-bes-wg/
[7] https://forge.gridforum.org/projects/jsdl-wg/
[8] http://www.semandex.net/whitepaper/Semandex_Content_Based_Routing_101.pdf
[9] http://en.wikipedia.org/wiki/Publish/subscribe
[10] http://en.wikipedia.org/wiki/XML
[11] http://en.wikipedia.org/wiki/Peer-to-peer
[12] http://www.cs.wisc.edu/condor/
[13] http://hpc.doc.ic.ac.uk/PPS/globus4/tsld007.htm
[14] http://www.nordugrid.org/documents/xrsl.pdf
[15] http://www.citebase.org/abstract?id=oai%3AarXiv.org%3Acond-mat%2F0312603
[16] http://www.stanford.edu/~ashishg/papers/scale-free.pdf
[17] http://en.wikipedia.org/wiki/Scale-free_network