Elosztott vezérlorendszerek eroforrásainak dinamikus kezelése

Somodi Zoltán
Kolozsvári Muszaki Egyetem

The classical measurement, control and actuator devices were based on simple physical principles (mechanical, hydraulic, pneumatic, electrical). Often they were used as stand-alone devices for relatively closed automation solutions. With the introduction of microprocessor technology and its fast spreading, the focus shifted from stand-alone devices to much more complex device systems connected to networks. Applications developed for these systems should be based on distributed services like: time service, event service, name service, real-time task distribution, error-handling service, group communication service etc. The name service is one of the most important services in distributed systems. This paper presents the design and implementation of a distributed dynamic resource management service, which makes possible a transparent data access.

1. Szolgáltatásokon alapuló elosztott vezérlorendszerek

A számítógépes hálózatok rohamos fejlodése teret hódít az ipari vezérlorendszerek területén is. Napjainkban több piacon levo termék is lehetoséget ad elosztott vezérlorendszerek kezelésére. Ezek a rendszerek, általában, egyes szakosodott cégek speciális termékei, amelyek olyan hálózatokon muködnek, amelyeken a többi csomópont is az illeto cégtol származik. Vagyis két különbözo cég által gyártott berendezés nem tud egymással kommunikálni, ezért nagyon nehéz vagy sokszor lehetetlen egy nem homogén hálózat kiépítése.

A Kolozsvári Muszaki Egyetem egyik kutatócsoportja egy olyan összefüggo szolgáltatásrendszer megtervezését és elkészítését tuzte ki célul, amely támogatja a vezérlo-alkalmazások fejlesztését. A rendszer tervezésénél figyelembe vettük egy jól ismert szabvány, az MMS (Manufacturing Message Specification) ajánlásait. Ezen rendszer segítségével fejlesztett alkalmazásoknak különbözo típusú eszközökkel kell tudniuk kommunikálni ahhoz, hogy a létrejött hálózat homogén legyen.

A cél eléréséhez a tervezésnél a következo elvekre támaszkodtunk:

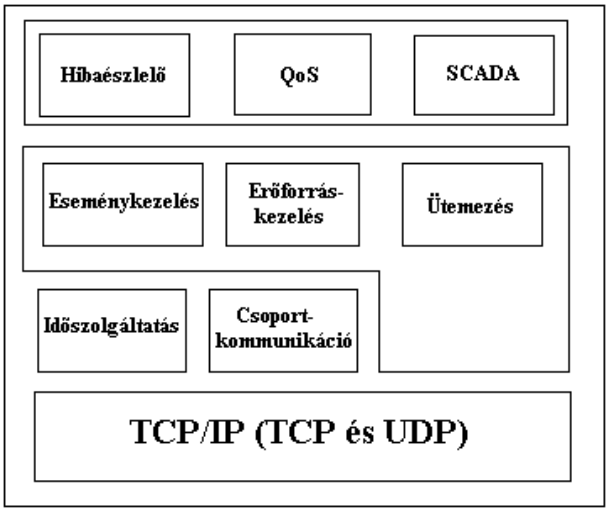
- **Autonóm és elosztott szolgáltatások:** a szolgáltatások ne függjenek egymástól, vagyis egy szolgáltatásnak akkor is kell muködnie, ha egyedül van a hálózaton. Ugyanakkor, nem lehet egy központi csomópont, amely irányítja a szolgáltatások muködését.
- **Egy szolgáltatás a hálózaton lévo hasonló helyi szolgáltatások összessége:** az autonóm és elosztott szolgáltatásoknak együtt kell muködniük ahhoz, hogy minél kielégítobb választ tudjanak adni a kliensek kéréseire.
- A szolgáltatásoknak rendelkezniük kell a következo mechanizmusokkal:
  - ***Idokezelés***: egy globális ido szükséges a rendszerben, amelynek segítségével megállapítható az események idobeli sorrendje.
  - ***Minoségkezelés***: a szolgáltató megegyezhet a felhasználóval a szolgáltatás minoségi szintjében, amelyet a szolgáltató köteles betartani.
- **A rendszer dinamikus újrakonfigurálásának lehetosége:** egy csomópont meghibásodása, vagy egy új csomópont megjelenése esetén a rendszer automatikusan újra kell magát konfigurálja.
- **Egységes hozzáférés a szolgáltatásokhoz:** egy egységes interfész létezését feltételezi, amelyen keresztül elérhetjük a szolgáltatásokat.
- Korlátozott eroforrással rendelkezo csomópontok kiszolgálása

Ezeket az elveket figyelembe véve megterveztünk egy elméleti modellt, amely a következo szolgáltatásokból épül fel:

1. **Idoszolgáltatás:** fo feladata a globális ido kezelése.
2. **Eseménykezelés:** az események észlelése, bejegyzése, figyelmeztetések és tevékenységek aktiválása.
3. **Valós ideju ütemezo:** feladatok (task-ok) ütemezése
4. **Hibaészlelo/kezelo:** hibák észlelése és kezelése, megfelelo események létrehozása
5. **Eroforrás-kezelés:** adatokhoz való transzparens hozzáférés biztosítása
6. **Csoportkommunikáció**
7. **SCADA (**Supervisory Control and Data Acquisition**):** adatok gyujtése és megjelenítése

Az 1. ábra a felsorolt szolgáltatások egymáshoz való viszonyát ábrázolja.

A kommunikáció a TCP és UDP protokollokra épül, amelyekhez egyes szolgáltatásoknak közvetlen hozzáférésük van, mások csak más szolgáltatásokon keresztül érhetik el.



1. ábra
A rendszer szolgáltatásai

2. Eroforrás-kezelo szolgáltatás

Ebben a cikkben az eroforrás-kezelo szolgáltatást ismertetjük részletesebben. Ez a szolgáltatás az egyik legfontosabb a rendszer muködése szempontjából, mivel az elosztott rendszerek egyik alapkövetelményét, az eroforrásokhoz való transzparens hozzáférést, biztosítja.

A szolgáltatás fo célja, hogy az eroforrásokat olyan szimbolikus nevekkel lehessen azonosítani, amelyek nem függenek az eroforrás típusától és a hálózaton való elhelyezkedésétol. A továbbiakban eroforrás alatt olyan fizikai vagy logikai eszközt értünk, amelyhez a felhasználónak közvetlen hozzáférése van. Eszerint eroforrás lehet egy változó, egy program, egy fizikai eszköz vagy berendezés, egy regiszter, egy esemény, vagy akár egy szolgáltatás. Különbözo típusú eroforrásokhoz különbözo hozzáférési módszer társul. Például, egy változót lehet olvasni és írni, egy programot viszont futtatni és megállítani lehet.

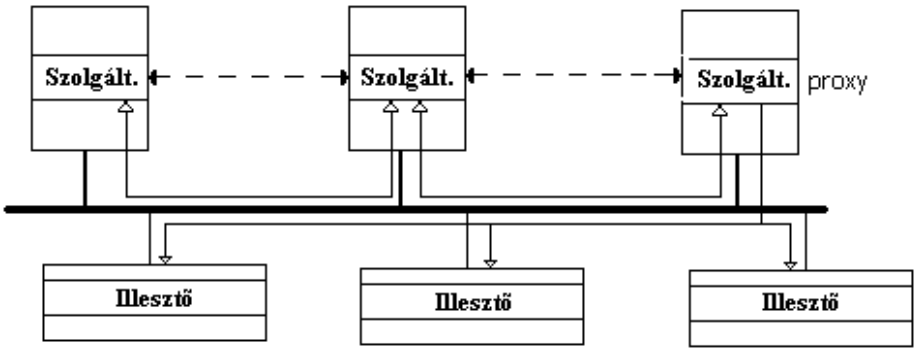
Az eroforrás-kezelo szolgáltatásnak a következo fobb feladatai vannak:

- **Helyi eroforrások azonosítása:** helyi eroforrásoknak nevezzük azokat az eroforrásokat, amelyek közvetlen kapcsolatban vannak egy csomóponttal. Például, helyi eroforrás lehet a soros csatlakozóhoz kötött eszköz vagy a csomóponton futó program.
- **Szimbolikus nevek és fizikai címek táblázatának frissítése:** minden csomópont egy táblázatban tárolja a különbözo fizikai címekhez rendelt szimbolikus neveket. Ezek a címek a rendszer muködése során változhatnak, ezért a táblázatot idonként frissíteni kell.
- **Szolgáltatások együttmuködése távoli eroforrások azonosítása érdekében:** a felhasználó mindig a helyi csomóponton futó szolgáltatáshoz intézi a kéréseit. Ha az eroforrás nem ezen a csomóponton van, tehát nem helyi eroforrás, a szolgáltatás továbbítja a kérést a megfelelo csomóponthoz, majd miután megkapta a választ, közvetíti azt a felhasználónak. Ez a mechanizmus a felhasználó számára transzparens.
- **Fizikai cím lekérdezése:** egyes esetekben (például biztonsági okokból) a felhasználónak szüksége lehet arra, hogy közvetlenül hozzáférjen egy bizonyos eroforráshoz, a eroforrás-kezelo szolgáltatás igénybevétele nélkül. Ilyenkor lehetoség nyílik a fizikai cím lekérdezésére.
- **Eroforrásokhoz való hozzáférés biztosítása:** változók írása/olvasása, programok elindítása/megállítása stb. A hozzáférés egyforma kell legyen helyi és távoli eroforrások esetén is.
- **Korlátozott eroforrásokkal rendelkezo eszközökhöz való hozzáférés:** egy olyan eszközzel, amelynek nincs például operációs rendszere, pont úgy kell tudnunk kommunikálni, mint egy másik, operációs rendszerrel rendelkezo eszközzel. Ezt a feladatot látja el a késobbiekben ismertetett proxy csomópont.

3. A szolgáltatás muködési elve

A kommunikáció kliens-szerver modellre épül és a TCP/IP protokoll-családot használja. A felhasználó mindig a kliens szerepét tölti be. Egy csomóponton futó szolgáltatás viszont – amint azt az alábbiakban látni fogjuk – lehet kliens is és szerver is.

A hálózaton minden csomóponton, amelyhez csatlakozik valamilyen eroforrás, futnia kell az eroforrás-kezelo szolgáltatásnak. Ezeken kívül más csomóponton is muködhet a szolgáltatás. A hálózaton muködo összes helyi szolgáltatás alkotja a valódi globális szolgáltatást. Ezt az elosztott rendszert szemlélteti a 2. ábra. Bizonyos eszközök illesztok segítségével kapcsolódnak a hálózatra. Ha egy új csomópont kerül be a hálózatba, jelt adva magáról, rögtön kapcsolatba kerül a többi szolgáltatással. Így a szolgáltatások automatikusan felfedezik egymás. Ugyanez az automatizmus történik akkor is, amikor egy csomópont eltunik a hálózatról.



2. ábra
Az elosztott szolgáltatás felépítése

Mint már említettük, minden csomópont rendelkezik egy táblázattal, amelyben a hozzákapcsolt eroforrások fizikai címét és szimbolikus nevét tárolja. Ugyanebben a táblázatban tárolja a többi, nem helyi eroforrás nevét és annak a csomópontnak a címét, amelyhez tartozik. Ha egy kliens egy kérést intéz a szolgáltatáshoz, e táblázat alapján történik meg a kliens által ismert szimbolikus név leképzése fizikai címmé. A kliens mindig csak a helyi szerverrel kommunikál. Abban az esetben, mikor a helyi szerver nem tud választ adni a kliens kérésére, mert az eroforrás nem helyi, a táblázat alapján továbbítja a kérést annak a szervernek, amely rendelkezik a megfelelo információkkal. Ilyenkor az elso szerver kliens szerepet tölt be addig, amíg meg nem érkezik a válasz. Ezután újra szerverként fog muködni, és a kapott választ elküldi a kliensnek.

A szerverek rövid ideig tárolják a más szerverektol kapott információt, tehát egy ún. *cache* szerepet is betöltenek. Ez azt jelenti, hogy ha egy bizonyos idon belül megismétlodik a kérdés, a helyi szerver nem kéri le újra az adatokat a távoli csomóponttól, hanem rögtön válaszol a kliensnek a tárolt adatok alapján. Hogy egy adott eroforrás adatai mennyi ideig maradhatnak „idegen” szerverek memóriájában, a *ttl* értékük határozza meg. Ez az érték, ami egy 0-nál nagyobb egész szám, az eroforrás egy változtatható tulajdonsága. Amikor az eroforrás adatai átkerülnek egy másik csomópont memóriájába, ez a szám csökkeni kezd az ido függvényében. Ha eléri a 0-t, az adatok érvényüket vesztik, és törlodnek a memóriából.

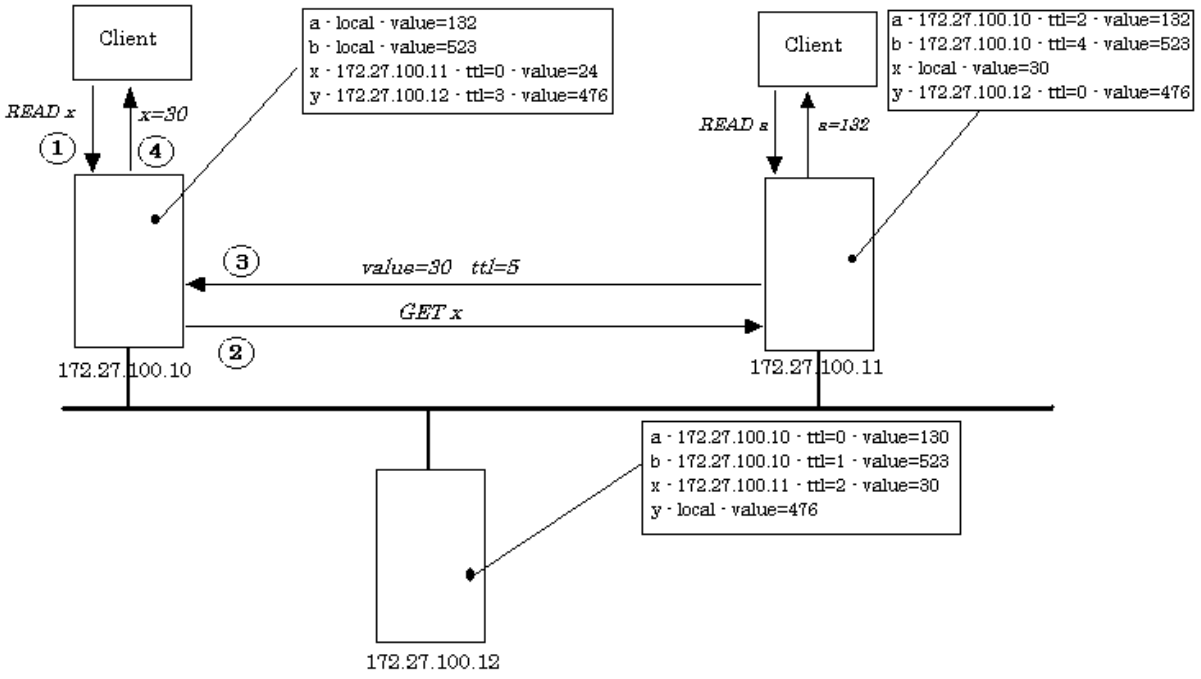
A rendszer muködését jobban megérthetjük, ha megvizsgáljuk a 3. ábrát. A kliens az *x* változó értékét akarja lekérdezni, kérését elküldi a helyi szolgáltatásnak (1). A szolgáltató táblázatából kiderül, hogy az *x* nem helyi eroforrás. Értéke ugyan ott van még a cache-ben, de már nem érvényes, mivel a *ttl* egyenlo 0-val. Ezért továbbítja a kérést a megfelelo szervernek (2), ahol az *x* helyi változóként szerepel. A szerver megkapja a változó értékét (3) és ennek alapján már válaszolni tud a kliensnek (4). Ugyanakkor újra beállítódik a *ttl* az eredeti értékre. Egy másik kliens az *a* változó értékét szeretné leolvasni. Az adott csomóponton a *a* nem helyi változó, de a cache-ben levo érték *ttl*-je nagyobb 0-nál. Így a szerver rögtön tud válaszolni a kérésre.

4. A szolgáltatás gyakorlati megvalósítása

Kutatócsoportunk az elméleti modell megtervezése után rátért a gyakorlati megvalósításra. Ehhez a Windows operációs rendszer alatt futó Visual C++ programozási környezetet használtuk. A kommunikációt a hagyományos socket-ekkel oldottuk meg.

A megvalósított szolgáltatás a következo kérés-típusokat tudja értelmezni:

? **Egyszeru azonosítás** : ahhoz, hogy a felhasználó igénybe tudja venni a szolgáltatást, jelszóval azonosítania kell magát.
? **Eroforrás létezése:** megtudhatjuk, hogy egy bizonyos nevu eroforrás létezik-e.
? **Eroforrás-lista lekérdezése:** az összes, a hálózaton lévo, eroforrás nevét kapjuk eredményül.
? **Eroforrás értékének írása és olvasása:** foleg változók esetében használjuk.
? **Eroforrás aktiválása/deaktiválása:** az eroforrás állapotának megváltoztatása.
? **Eroforrás állapotának, típusának és címének lekérdezése:** információt kapunk az eroforrás tulajdonságairól, illetve megtudhatjuk fizikai címét.
? **Szerverek lekérdezése:** minden szerver le tudja kérdezni, hogy ki van még a hálózaton, aki hasonló szolgáltatást nyújt.



3. ábra
Muködési példa

Minden függvénynek paraméterként az eroforrás nevét adjuk meg. A kliensek csak név szerint tudják azonosítani az eroforrásokat. Ezek a függvények könyvtárakba vannak csoportosítva. Ha egy olyan alkalmazást akarunk írni, amely a szolgáltatásunkat használja, csak ezt a könyvárat kell bekössük programunkba. Ezzel elrejtettük a programozó elol a kommunikáció részleteit.

5. Összefoglalás

Kutatómunkánk eredményeként létrejött egy elosztott szolgáltatásokon alapuló elméleti modell, amely felhasználható vezérlorendszeres alkalmazások fejlesztésére. A megtervezett szolgáltatások szorosan együttmuködnek egymással az alapelvek teljesítése érdekében. Részletesen kidolgoztuk az eroforrás-kezelo szolgáltatás elméleti modelljét is, valamint e szolgáltatás kapcsolatait a rendszer többi részével.

Elméleti modellünket ki is próbáltuk ,megvalósítva néhány szolgáltatás alapveto függvényét. Az eroforrás-kezelés az egyik legjobban kidolgozott szolgáltatás, amely rendelkezik egyszerubb és bonyolultabb függvényekkel. Ezenkívül létrehoztuk a fobb adatstruktúrákat megvalósítva a szolgáltatás muködési mechanizmusát. A tesztek során bebizonyosodott az elméleti modell helyessége, ugyanakkor fény derült a megvalósítási hiányosságokra. Itt foleg a Windows operációs rendszer és a valós ideju rendszerek közötti inkompatibilitást említhetjük.

Az elkészített program jelenlegi változatában az eroforrás-kezelo magába foglalja a periodikus adatgyujtést is. Ezt a következo lépésben szét fogjuk választani, és létrehozunk egy különálló adatgyujto szolgáltatást, amelynek az lesz a fo feladata, hogy a perifériikus eszközökrol idonként leolvassa az adatokat, és azt egy

adatbázisban tárolja. Az eroforrás-kezelo pedig ebbol az adatbázisból fogja felépíteni a táblázatokat. Mivel két szolgáltatás is használni fogja az adatbázist, létrehozunk egy adatbázis-kezelo szolgáltatást is. Így csak ennek a szolgáltatásnak lesz közvetlen hozzáférése az adatbázishoz, a többi szolgáltatás csak rajta keresztül érheti el az adatokat. Ez megkönnyíti és transzparenssé teszi a hozzáférést.

A kutatás folytatásaként a fent említett módosításokon kívül, ki szeretnénk bovíteni az igénybe veheto kérések számát, és bevezetni egy többszintu felhasználó-azonosítást. Ugyanakkor meg szeretnénk vizsgálni más kommunikációs módszerek alkalmazását, amivel jobban tudnánk alkalmazkodni a valós-ideju rendszerek követelményeihez.

Könyvészet

[1] Fredriksson L.B. [1994], "Controller Area Networks and the Protocol CAN for Machine Control Systems", *Mechatronics* , Vol. 4 No.2 pp 159-192

[2] Cheriton D. R., Mann T. P., *Decentralizing a Global Naming Service for Improved Performance and Fault Tolerance*, Standford University, 1997

[3] Sebestyen G. [1999], " Industrial Communication Networks – Modeling and Simulation", Technical Report, UTCN, pp. 1-55

[4] Sebestyen G., *Planificarea in sistemele distribuite de timp-real* - referat de doctorat

[5] Pusztai K., Sebestyen G., „New Communication Technologies for Distributed Control", A&QT-R 2002 (THETA 13), 2002 IEEE-TTTC International Conference on Automation, Quality and Testing, Robotics, May 23-25, 2002, Cluj-Napoca

[6] Pusztai K., Sebestyen G., Kiss Cs., „Distributed Control through Web Technologies", First RoEduNet International conference, Cluj-Napoca, 2002

[7] Sebestyen G., Pusztai K., Somodi Z., „Dynamic Resource Management For Distributed Control Systems", The 14th International Conference on Control Systems and Computer Science, 2003, Bucuresti

[8] Mackiewicz R. *Overview to the Manufacturing Message Specification*, http://litwww.epfl.ch/~mms/mms_IntroSISCO, 1994

[9] O. Redell, *Modelling of Distributed Real-Time Control Systems - An Approach for Design and Early Analysis*, Royal Institute of Technology, Stockholm 1998