Békési Gábor*

TRANZAKCIÓKEZELÉS A WEBSZOLGÁLTATÁSOKBAN

Tranzakciók

A *tranzakciók* olyan két vagy több önálló műveletből álló tevékenységek, melyeket egy rendszerben egyetlen komplex folyamatnak tekintünk. A tranzakcióban érintett rendszernek konzisztens állapotban kell maradnia, azaz ha minden művelet sikeresen ér véget, akkor a műveletek eredménye a rendszert új állapotba transzformálja, ellenkező esetben rendszerünkben semmilyen változás nem történhet.

Minden tranzakciót az alábbi négy tulajdonság jellemez:

oszthatatlan (atomi)
A tranzakció végrehajtása során a *minden vagy semmi* elve érvényesül. Gondoljuk el, ha egy tranzakció egy adatbázis egyik táblájába beszúr egy rekordot és egy másik táblából egyet töröl, csak akkor lehet sikeres, amennyiben mindkét művelet megtörtént.

konzisztens
A tranzakció nem hagyhat egy rendszert inkonzisztens állapotban. Az előző példánál maradva, megengedhetetlen, hogy a törlés sikertelensége miatt a beszúrt rekord a táblában maradjon.

elszigetelt
Független tranzakciók egymástól függetlenül futnak le. Ez a gyakorlatban a tranzakciók sorba állításával (*szerializáció*) valósul meg. Például érkezési sorrend alkalmazása esetén a később érkezettek várnak, amíg a korábban indult – akár sikeresen, akár nem – befejezi a műveletet.

tartós (perzisztens)
A tranzakciók permanensek, nem veszíthetjük el őket a hálózat meghibásodása esetén sem. Erről a tranzakciókoordinátorok hivatottak gondoskodni. Ezek olyan folyamatok, melyeknek túl kell élniük az ilyen katasztrófákat és minden elkezdett tranzakciót be kell tudniuk fejezni.

* *főiskolai tanár, Általános Vállakozási Főiskola*

Amennyiben egy tranzakció kilép egy folyamatból vagy hálózatba kapcsolt gépek között zajlik, *elosztott tranzakcióról* beszélünk. Az ilyen tranzakciók kezelésének, menedzselésének programozási technikája évek óta ismert és igen kifinomult. A *Microsoft* a WCF[1]-ben már a szolgáltatás szintjén is alkalmazhatóvá teszi az elosztott tranzakciók használatát.

Tranzakciókezelés

A tranzakciók fent felsorolt tulajdonságainak érvényesítésére mind manuális, mind automatizált eljárások ismertek. A manuális megoldások szerepe – különösen nagy projektek irányításában – elvitathatatlan, alkalmazásukhoz azonban két feltételt mindenképpen figyelembe kell venni:

- a tevékenységeknek kapcsolódniuk kell egymáshoz, és minden tevékenységhez le kell írniuk egy visszagörgetési (*rollback*) műveletet is,
- nagyon sok alárendelt komponens és művelet megnehezíti a koordinátor szerepét, hiszen át kell látnia a kapcsolatokat, hogy az egyes lépések sikerességét vagy hibáját megítélhesse.

A gyakorlatban az automatikus megoldások jutnak nagyobb szerephez. A legismertebb szabványosított eljárás az ún. kétfázisú megegyezési protokoll, az angol kifejezés rövidítéséből a 2PC[2]. Ebben a protokollban a döntéshozásért egy önkényesen választott hálózati elem, a koordinátor a felelős, de a tranzakció valamennyi résztvevőjének vétójoga van a végső döntést illetően. A folyamat minden egyes lépését naplófájlban rögzítik.

Az első fázisban a koordinátor egy „készülj fel a befejezésre” üzenetet küld a résztvevőknek és a küldés tényét naplózza. Ha a résztvevő bizonyosan be tudja fejezni a tranzakcióban rá háruló feladatot (azaz az ahhoz szükséges valamennyi erőforrást kizárólagos joggal le tudta foglalni), egy „felkészültem” választ küld vissza a koordinátornak és ezt a naplóban is rögzítik. Amennyiben valamennyi résztvevő üzenete beérkezett, a koordinátor eldönti, hogy a tranzakció sikeresen befejezhető vagy sem.

A második fázis a koordinátor döntésének naplózásával kezdődik. Ezt követően minden résztvevőt értesít a döntéséről. Ha a sikeres befejezés mellett döntött (minden résztvevő válasza a „felkészültem” volt), az egyes résztvevők egy „befejezés megkezdése” (*commit*) üzenetet írnak a naplóba és elvégzik a szükséges műveleteket. A műveletek befejezését az erőforrások felszabadítása követi, és a koordinátorhoz meg a naplónak egy „befejeztem” üzenet megy. A koordinátor bevárja az összes üzenetet (most már csak „befejeztem” üzenetek lehetnek) és lezárja a tranzakciót.

Ha az első fázis végén akár egyetlen résztvevő a „nem sikerült felkészülnöm” üzenetet küldi, a koordinátor lefújja a tranzakciót (*abort*), a résztvevők felszabadítják erőforrásaikat, és ezt is a „befejeztem” üzenettel közlik.

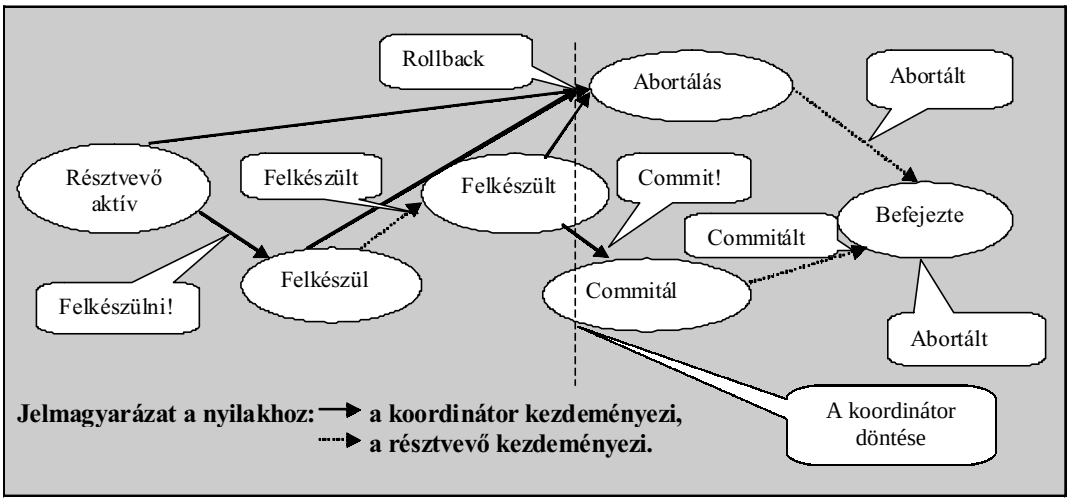
A naplózásos technika teszi lehetővé, hogy a tranzakció mindig befejezhető legyen. A folyamat persze időben nagyon elhúzódhat. Tipikus hiba lehet, hogy egy résztvevő a már elküldött (és naplózott) „felkészültem” üzenete után omlik össze. Ekkor újraindulása után újból le

[1] *Windows Communication Foundation – a Microsoft webszolgáltatásokat támogató platformja.*

[2] *Two-phase commit protocol.*

kell foglalnia az erőforrásait, és ebben az állapotban veszi, majd dolgozza fel a koordinátor válaszát. A többi résztvevő ekkor már réges-régen túl van a „befejeztem” üzenetén. A folyamat vázlata az 1. ábrán látható. Az ovális alakzatok a résztvevő állapotát, a nyilak az üzeneteket jelzik.

1. ábra
A 2PC-protokoll sémája



A *Microsoft* platform tranzakciókezelése

Ebben a környezetben az elfogadott terminológiát használva a 2PC-algoritmust a tranzakciókezelők valósítják meg, míg a tranzakció résztvevőit az erőforrás-kezelők képviselik. A 2PC fent leírt változata a naplófájl központi szerepével a perzisztenciát biztosítja, de a valós igények, az erőforrásokkal való ésszerű gazdálkodás e téren engedményt indokolnak. Amennyiben a tranzakció egyetlen feldolgozó folyamatban zajlik, és nem lép ki a számítógépből, a tranzakció résztvevőinek, az erőforrás kezelőknek nem szükséges perzisztensnek lenniük. Ezek a csak memóriaadatokkal dolgozó, ún. időleges erőforrás-kezelők persze nem élik túl a rendszer összeomlását, viszont a tranzakció nagyon gyorsan újraindítható, visszagörgetni sem kell semmit.

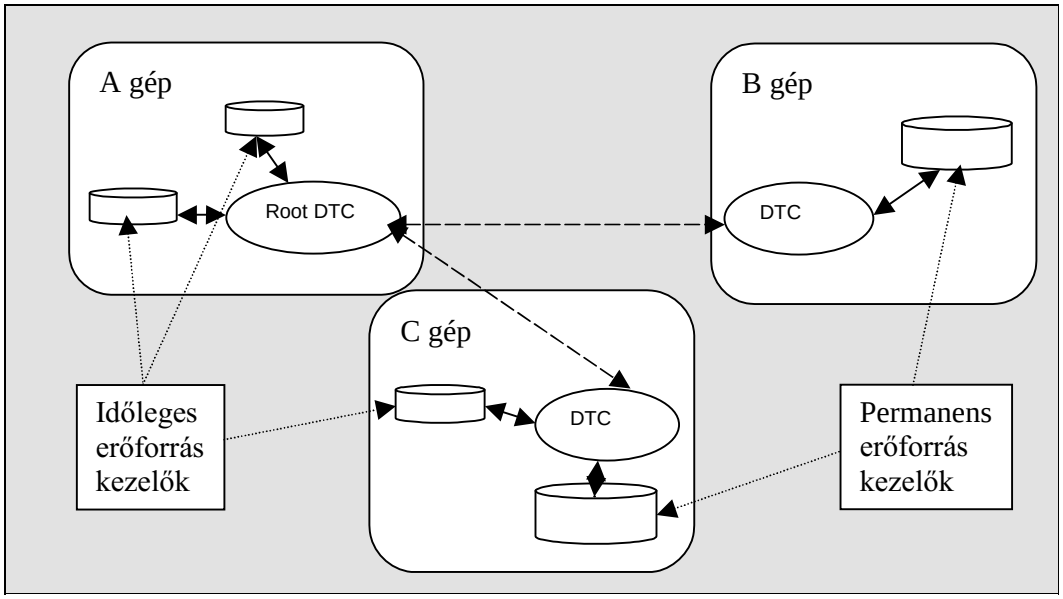
A NET Framework 2.0 már lehetőséget nyújtott a programkódból történő tranzakciókezelésre, elsősorban a nem-perzisztens (időleges) erőforrás-kezelőkre alapozva. Ez a mögöttes tranzakciókezelő igen hatékony egygépes munkafolyamatok esetén.

Alkalmazásaink zöme (köztük a webszolgáltatások) hálózati feldolgozó folyamatok együttesét jelentik. Ilyen esetben nemcsak elvárás, de már erőforrás-kímélő megoldás is a perzisztens erőforrás-kezelők használata és az ezekre építő tranzakciómenedzser. Ez utóbbi a *Microsoft Distributed Transaction Coordinator*, az MSDTC vagy röviden DTC. Jellemzője, hogy:

- alkalmazások, munkafolyamatok és gépek kapcsolatát kezeli,
- több permanens erőforrás-kezelőt használhat.

Minden egyes, a tranzakcióban érintett gépen a *lokális* DTC gyűjti be az (akár időleges, akár permanens) erőforrás-kezelők 2PC-protokoll szerinti válaszait és közülük az lesz a *root*-koordinátor, amelyik a tranzakciót kezdeményező gépen van. Ennek feladata a lokális DTC-k szavazatainak (döntéseinek) az összegyűjtése. Működését az 2. ábra szemlélteti.

2. ábra
A DTC-k működése



A *Microsoft* tranzakciókezelési stratégiájának fontos jellemzője, hogy a tranzakciók menedzselését a DTC *automatikusan* átveszi, mihelyt akár az alkalmazás vagy munkafolyamat hálózati kapcsolatot igényel, akár több permanens erőforrás-kezelő jelenik meg. Programozási feladatot nem jelent, csak a DTC konfigurálását kell elvégezni, ami – többek között – egy nyílt technológiájú protokoll aktiválásából áll.

A szolgáltatások platformfüggetlen tranzakciói

Az OASIS[3] webszolgáltatások tranzakcióival foglalkozó albizottsága a HTTP-re alapozott, platformfüggetlen és üzenetalapú osztott tranzakciók megvalósítására két protokollt is javasolt még 2005 végén: a WS-AT[4]-t ill. a WS-COOR[5]-t. A WS-AT a 2PC-algoritmus megvalósítását

[3] *Organization for the Advancement of Structured Information Standards – egy nemzetközi, nonprofit szervezet webszolgáltatások standardjainak fejlesztésére, koordinálására.*

[4] WS-AtomicTransaction*, utolsó verzója az „1.2" – lásd az irodalomban WS-AT (2009) alatt.*

[5] WS-Coordination*, utolsó verziója az „1.2" – lásd az irodalomban WS-COOR (2009) alatt.*

jelenti, ahol a résztvevők és a koordinátorok szabványos SOAP[6] -üzenetekkel kommunikálnak. Szabályozva van a lehetséges állapotok köre és az üzenetek tartalma, az előforduló hibák is szabványosítottak. A működés biztonsági modelljét a WS-COOR-protokoll specifikálja. A tranzakció valamennyi résztvevője az alkalmas erőforrás-kezelőknél regisztrálva van, az ehhez megfelelő jogokat a koordinátorok kezelik és ellenőrzik. Egy már azonosított (és biztonságosnak ítélt) résztvevő az őt követő résztvevőnek annak regisztrálásához delegálja a saját jogait, amit a koordinátor már felismer. Az egész folyamat a *root*-koordinátortól indul el. Az üzenetek titkosítása az osztott titok módszerén alapszik (ezt *SecureConversation*-nak nevezik), a koordinátorok az aláírásokban egymás hitelesítéséhez a publikus kulcsú technológiát alkalmazzák.

Bár a WS-AT-specifikáció még az utóbbi időben is módosult, a *Microsoft* a WCF-ben implementálta egy korai változatát. Ha ezt a tranzakciótámogatást beállítjuk, a DTC képes ellátni az osztott tranzakció menedzselésében a koordinátori szerepeket, hálózatos kapcsolat esetén is. Ha tehát mind a szerviz, mind a kliens gépén konfiguráltuk és engedélyeztük a WS-AT támogatást, a DTC automatikusan a WS-AT- és WS-COOR-protokollt fogja használni, az üzenetcsatorna szintjén. Ez a viselkedés még vegyes platformú hálózatokban is fennáll, azzal a korlátozással, hogy a HTTP feletti üzenetváltásban az üzenetek egy, csak a *Windows*-os gépeken levő DTC-k számára értelmezhető üzenetfejjel bővülnek, lehetővé téve ezzel közöttük a WS-AT alkalmazását.

A WS-AT működése WCF alatt

A tranzakcióforgalom (*transaction flow*) WCF-terminológia, a szerviz és kliens közötti tranzakciós kapcsolatot jelenti, ami kifejeződik kettőjük összeköttetésében (*binding*), a szolgáltatásra vonatkozó megállapodásban (*contract*) és a szolgáltatás működésének jellemzőiben (*behavior*). Az elvárások az alábbiak:

- az összeköttetésnek alkalmasnak kell lennie a tranzakcióforgalom támogatására és ezt a kezelési módot be is kell kapcsolnunk,
- alkalmas tranzakció kezelési protokoll álljon rendelkezésre,
- a szolgáltatási megállapodásban megadott műveleteknek kezelni kell tudni a tranzakcióforgalmat,
- a műveleteknek képesnek kell lenni felismerni a kliensektől indított tranzakciókat is.

Ami az összeköttetéseket illeti, hálózati forgalomban csak a *NetTcpBinding*, a *WSHttpBinding*, a *WSDualHttpBinding*, valamint a *WSFederationHttpBinding* alkalmas a tranzakcióforgalom kezelésére. Ezekben a kötésekben megtalálható a *transactionFlow* tulajdonság, s ezt igaz értékre kell állítanunk. [Testreszabott kötésekben (*CustomBinding*) ezt az attribútumot a fejlesztőnek kell specifikálni.] Ugyancsak a fenti összeköttetésekben meglévő tulajdonság a *transactionProtocol*. A DTC a vegyes platformú környezetek tranzakciókezeléséhez két protokollt ismer: az *OleTransactions*-t (ezt a sztenderd távoli eljárásos hívással működő üzenetcserék esetére), illetve a *WSAtomicTransactionOctober2004*-t (a WCF csak ezt az egyetlen WS-AT-változatot támogatja, ezt is HTTP felett). Az *OleTransactions* protokoll a gyorsabb, kevésbé erőforrás-igényes, a Tcp kötésű kapcsolatokban ez az alapértelmezett.

A műveletek viselkedésének szintjén a tranzakciókezelést további beállítások szabályozzák, ezekre a tesztfeladatban mutatunk példát.

[6] *Simple Object Access Protocol.*

A WS-AT és az MSDTC konfigurálása

Ahhoz, hogy használhassuk az *AtomicTransaction* protokollt, el kell végeznünk néhány beállítást. A konfigurálást a *WS-AtomicTransaction Configuration Utility* (*WsatConfig.exe*) segítségével oldjuk meg. Ennek a segédprogramnak van parancssoros módja is, mi az MMC[7] beépűlő modult használtuk beállításainkhoz mind az XP, mind a Vista operációs rendszer esetén. Az MMC a WS-AT környezet megjelenítéséhez felhasználja a *WsatUI.dll*-t, amelyet előzőleg regisztrálnunk kell[8]. (Az XP-khez szükséges egy javítás letöltése is a http://www.microsoft.com/downloads/details.aspx?FamilyID=86b93c6d-0174-4e25-9e5d-d949dc92d7e8&DisplayLang=hu címről.)

A WS-AT protokoll az DTC alatt csak akkor működik, ha a gépek közötti kommunikáció HTTPS-en folyik. (Tapasztalataink szerint, ha XP operációs rendszerű gép is részt vesz a tranzakcióban, az SSL[9] használatát nem is lehet megkerülni.) XP-k esetén a „Vezérlőpult/Felügyeleti eszközök/Komponensszolgáltatások”-ra kattintva a konzol ablakban nyissuk meg a „Komponensszolgáltatások” alatt a „Számítógépek” mappát és a „Sajátgép” kiválasztása után jobb gombbal kattintva a „Tulajdonságok” kiválasztásával kapjuk a 3. ábránkon bemutatott, több fület tartalmazó panelt, megnyitva közülük az MSDTC nézetet. Mindezt *Vista* alatt a „*Windows*” gombnál megnyíló kereső sorba gépelt *dcomcnfg.exe* hívással kezdeményezzük. Az MMC ablakban – mint az XP-nél – jussunk el a „Sajátgép”-ig, és válasszuk alatta az „Osztott tranzakciók koordinátora/Helyi DTC” lépéseket, majd ennek a „Tulajdonságok” menüpontjára kattintva jutunk az említett párbeszédpanelhez. Mindkét esetben természetesen rendszergazdai jogosultságra van szükség.

A helyi koordinátor beállításai a 3. ábráról leolvashatók. XP-ken a „Biztonsági beállítások” gomb megnyomásával, a Vista „Helyi DTC” ablakában a „Biztonság” fül kiválasztásával kapjuk a 4. ábrán látott ablakot. Jelöljük be rajta a „Hálózati DTC-hozzáférés” jelölőnégyzetet és a „Tranzakciókezelő kommunikáció” blokkban engedélyezzük mind a bejövő, mind a kimenő forgalmat!

Említettük, hogy a WS-AT biztonsági csatornát használ. Ehhez nemcsak egy HTTPS-portot kell megadnunk, hanem minden gépnek saját SSL-tanúsítványra van szüksége. Az SSL-tanúsítványok publikus kulcsú technológiával készült tanúsítványok azzal a megkötéssel, hogy a tulajdonos nevének minden gépen a tartomány/gép minősítőnévvel azonosnak kell lennie. Tanúsítványaink az *OpenSSL* nyilvános programmal készültek. A saját kulcsot tartalmazó tanúsítványokat (ezek „.pfx” kiterjesztésűek) a számítógépek „Sajátgép/Személyes” tárolójába kell felvennünk, a csak publikus kulcsot hordozók („.cer” kiterjesztésűek) a „Sajátgép/Megbízható személyek” tárolóba kerülnek. (Helyi kibocsátónk tanúsítványa a Megbízható Legfelső Szintű Kibocsátók tanúsítványtárolójában szerepel.) Beállításainkat az XP-s gépen az 5. ábra szemlélteti.

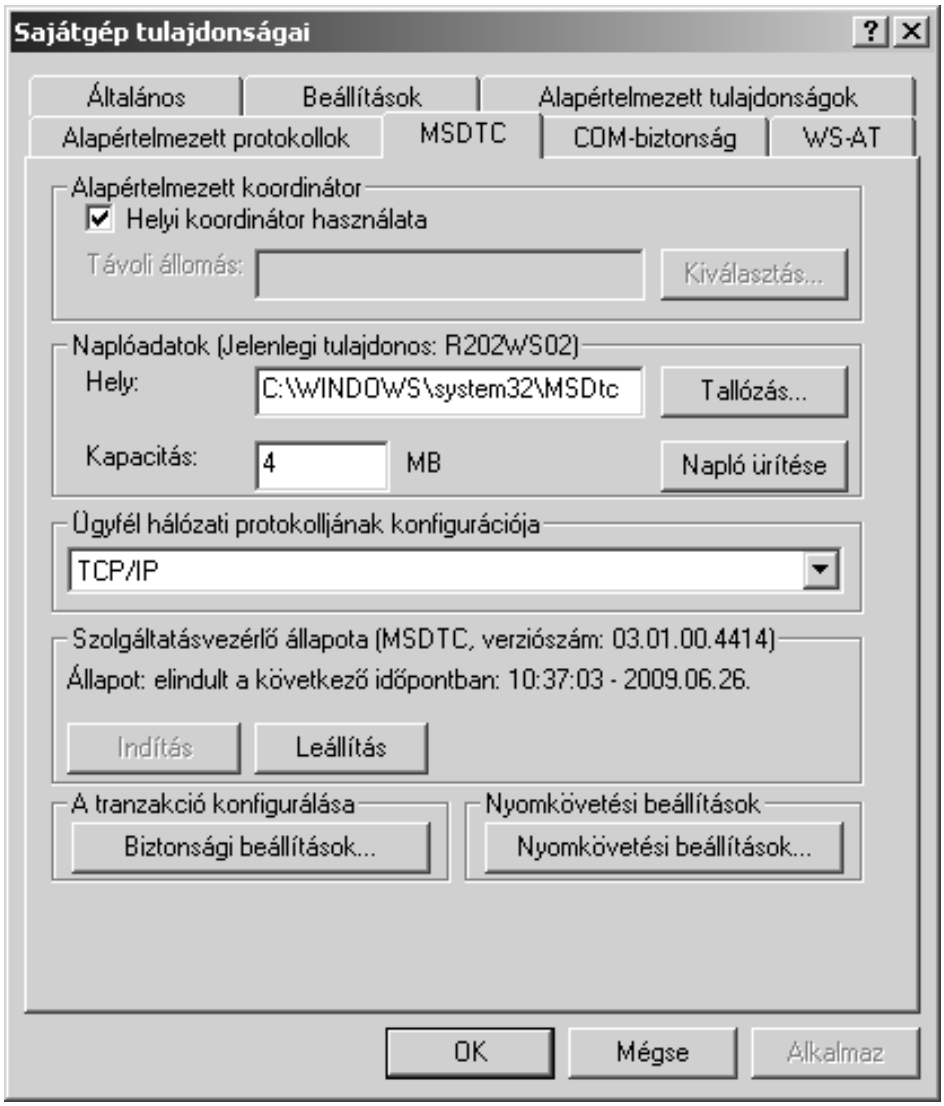
[7] *Microsoft Management Console.*

[8] *Lásd az irodalomban a Configuring WS-Atomic Transaction Support (2009) cikket.*

[9] *Secure Sockets Layer – egy szállítási szintű biztonsági protokoll.*

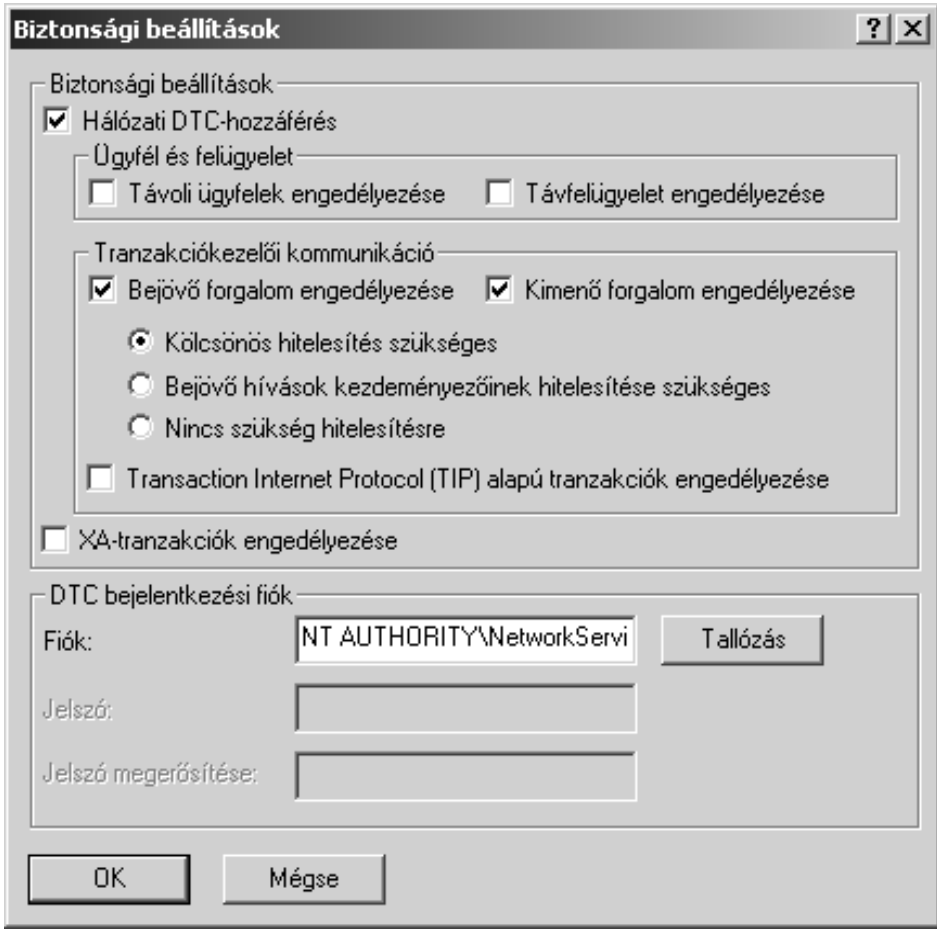
3. ábra

Az MSDTC beállításai



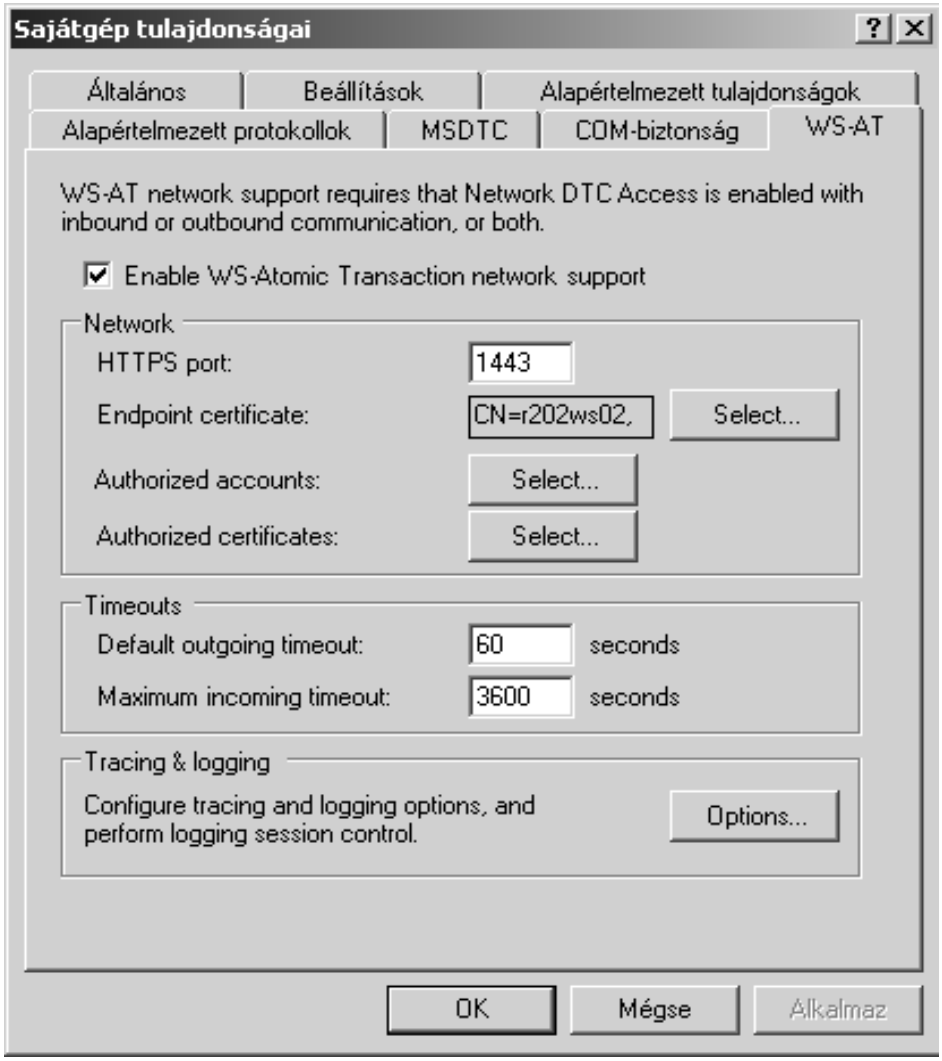
4. ábra

Biztonsági beállítások



5. ábra

A WS-AT beállításai



A WS-AT konfigurációját ellenőrizhetjük a már említett *wsatconfig.exe* parancssoros programmal, ha azt a „*–show*” paraméterrel hívjuk meg.

Az MMC a tűzfal „Kivételei” közé felveszi a megadott HTTPS-portot, az *msdtc.exe* programot azonban manuálisan kell a kivételekhez hozzáadnunk[10].

[10] *Lásd az irodalomjegyzékben Troubleshooting Problems with MSDTC (2009) alatt.*

A tesztprogram[11]

Mintaalkalmazásunk célja a WS-AT beállításainak és működésének tesztelése volt, erre *Bustamante* példája tökéletesen megfelelt. A programban az ügyfél egy MS-Sql-adatbázis *Counters* nevű táblájában módosít értékeket egy-egy rekordban. Az adatbázis-kezelő permanens erőforrás-kezelő a tranzakcióban és – figyelembe véve a hálózati kapcsolatot – a két gép DTC-i a *WSAtomicTransactionOctober2004*-protokoll segítségével kommunikálnak. Az alkalmazás többféle kötést is tesztel, a csővezeték azonban csak lokális résztvevők esetén jöhet szóba.

Itt nem lehet célunk a WCF programozási technológiájának az ismertetése, az érdeklődő olvasó rövid összefoglalót talál a szerző irodalomjegyzékben hivatkozott Békési (2009) munkájában.

A 6. ábra egy szemelvény, ahol a szerver szolgáltatásának és a műveletek megállapodásainak, attribútumainak beállításait láthatjuk (bár itt csak az adatrekordokat kezdeti értékre állító eljárást mutatjuk), amelyek a tranzakció menedzselése szempontjából érdekesek. A 7. ábra egy részletet közöl a szerver konfigurációs állományából a kötéseket és azok viselkedését szemléltetendő.

A Tcp-alapú kötések mindegyikében explicite előírtuk a *WSAtomicTransactionOctober2004*-protokoll használatát.

6. ábra
A szolgáltatás definíciója és egyik művelete

```
[ServiceContract(Namespace =". . .")]
    public interface ICountersService
    {
        [OperationContract]
        [TransactionFlow(TransactionFlowOption.Allowed)]
        void ResetCounters();
        [OperationContract]
        [TransactionFlow(TransactionFlowOption.Allowed)]
        void SetCounter1(int counterValue);
        [OperationContract]
        [TransactionFlow(TransactionFlowOption.Allowed)]
        void SetCounter2(int counterValue);
        [OperationContract]
        [TransactionFlow(TransactionFlowOption.NotAllowed)]
        List<CounterInfo> GetCounters();
    }

    [ServiceBehavior(InstanceContextMode = InstanceContextMode.PerCall)]
    public class CountersService : ICountersService
    {
        [OperationBehavior(TransactionScopeRequired=true,
TransactionAutoComplete=true)]
        public void ResetCounters()
        {
           try
            {
                using (TransactionScope scope = new TransactionScope())
                {
                    using (Counters.Dalc.CountersDataAccess
countersDataAccess = new Counters.Dalc.CountersDataAccess())
                    {
                        countersDataAccess.SetCounter1(0);
                        countersDataAccess.SetCounter2(0);
                    }
                    scope.Complete();
                }
            }
            catch (InvalidOperationException ex)
            {
                throw new FaultException(ex.Message);
            }
        }
```

[11] *A megoldás Bustamante, M. e munkájából származik:* Learning WCF*, O'Reilly, (2007).*

7. ábra

Részlet a szerver konfigurációs fájljából

```
<system.serviceModel>
      <services>
            <service name="Counters.CountersService"
behaviorConfiguration="serviceBehavior">
                  <host>
                        <baseAddresses>
                              <add
baseAddress="net.pipe://localhost/CountersService"/>
                              <add
baseAddress="net.tcp://r202ws02:9000/CountersService"/>
                              <add
baseAddress="http://r202ws02:8000/CountersService"/>
                        </baseAddresses>
                  </host>
                  <endpoint address="wsHttpTx" binding="wsHttpBinding"
contract="Counters.ICountersService" bindingConfiguration="wsHttpTx" />
                  <endpoint address="wsHttpTxRM" binding="wsHttpBinding"
contract="Counters.ICountersService" bindingConfiguration="wsHttpTxRM"
/>
                  <endpoint address="netTcpTx" binding="netTcpBinding"
contract="Counters.ICountersService" bindingConfiguration="netTcpTx" />
                  <endpoint address="netTcpTxRM" binding="netTcpBinding"
contract="Counters.ICountersService" bindingConfiguration="netTcpTxRM"
/>
                  <endpoint address="netPipeTx"
binding="netNamedPipeBinding" contract="Counters.ICountersService"
bindingConfiguration="netPipeTx" />
                  <endpoint address="wsHttpCustomTx"
binding="customBinding" contract="Counters.ICountersService"
bindingConfiguration="wsHttpCustomTx" />
                  <endpoint contract="IMetadataExchange"
binding="mexHttpBinding" address="mex" />
            </service>
      </services>
      <bindings>
            <wsHttpBinding>
                  <binding name="wsHttpTx" transactionFlow="true" />
                  <binding name="wsHttpTxRM" transactionFlow="true">
                        <reliableSession enabled="true" ordered="true"
inactivityTimeout="00:10:00"/>
                  </binding>
            </wsHttpBinding>
            <netTcpBinding>
                  <binding name="netTcpTx" transactionFlow="true"
transactionProtocol="WSAtomicTransactionOctober2004" >
                  </binding>
      . . .
```

Felhasznált irodalom

Békési Gábor (2009): *WCF alkalmazások.* (Kutatást záró dokumentáció). 2009, Budapest, ÁVF, 42 p.

Configuring WS-Atomic Transaction Support. Microsoft, 2009 (http://msdn.microsoft.com/en-us/library/ms733943.aspx)

Troubleshooting Problems with MSDTC. Microsoft, 2009. (http://msdn.microsoft.com/en-us/library/aa561924(BTS.20).aspx)

Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.2, OASIS Standard, 2 February 2009.

Web Services Coordination (WS-Coordination) Version 1.2, OASIS Standard, 2 February 2009.