



(19) **HU**

MAGYAR KÖZTÁRSASÁG
Magyar Szabadalmi Hivatal

(11) Lajstromszám: **225 569**

(13) **B1**

SZABADALMI LEÍRÁS

(21) A bejelentés ügyszáma: **P 98 02083**

(22) A bejelentés napja: **1996. 07. 11.**

(40) A közzététel napja: **1998. 12. 28.**

(45) A megadás meghirdetésének dátuma a Szabadalmi Közlöny és Védjegyértesítőben: **2007. 03. 28.**

(51) Int. Cl.: **H04L 29/08** (2006.01)

(86) A nemzetközi (PCT) bejelentési szám:

PCT/US 96/11553

(87) A nemzetközi közzétételi szám: **WO 9730554**

(30) Elsőbbségi adatok:

08/601,755 1996. 02. 15. US

(72) Feltalálók:

Bittinger, Reed Richard, Raleigh,
North Carolina (US);
Fraenkel, Michael Levi, Raleigh,
North Carolina (US);
Housel, Barron Cornelius III., Chapel Hill,
North Carolina (US);
Lindquist, David Bruce, Raleigh,
North Carolina (US)

(73) Jogosult:

International Business Machines Corp.,
Armonk, New York (US)

(74) Képviselő:

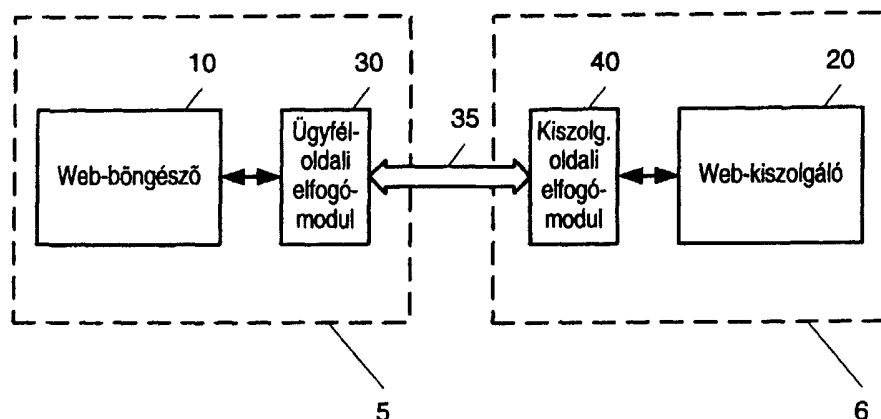
dr. Bogsch Attila, Budapest

(54) **Eljárás, berendezés, valamint számítógépi programtermék számítógépeken lévő alkalmazások között külső kommunikációs kapcsolaton keresztül, TCP kommunikációs protokoll felhasználásával továbbított adatok mennyiségének csökkentésére**

(57) Kivonat

Eljárás, berendezés és számítógépi programtermék egy első számítógépen (5) lévő első alkalmazástól egy TCP protokollt használó kommunikációs kapcsolaton (35) keresztül egy második számítógépen (6) lévő második alkalmazáshoz továbbítandó adatok mennyiségének

csökkentésére. Egy első virtuális szoftvercsatortát hoznak létre az első számítógépen (5), válaszul az első alkalmazás minden egyes csatlakozási kérelmére az első alkalmazástól kiinduló kérelemadatok fogadása céljából. Az első számítógépen (5) egy első valódi szoftvercsa-



2. ábra

HU 225 569 B1

tornát, valamint a második számítógépen (6) egy második valódi szoftvercsatornát hoznak létre az első számítógépnek (5) a második számítógéppel (6) a külső kommunikációs kapcsolaton (35) keresztül való összeköttetése céljából. Ezek fennmaradnak, amíg a kérelemadat el nem jut a második alkalmazáshoz. A második számítógépen (6) egy második virtuális szoftvercsatornát hoznak létre az első alkalmazás minden egyes csatlakozási kérelméhez, ahol a második virtuális szoftvercsatorna az első számítógépen (5) az első alkalmazás csatlakozási kérelmére válaszul létrejött első virtuális szoftvercsatornához tartozik. Az első virtuális szoftvercsatornával kapcsolatos kérelemadatot az első valódi szoftver-

csatornára multiplexelik, és a külső kommunikációs kapcsolaton (35) keresztül továbbítják a TCP protokollt használva. A multiplexelt kérelemadatot a második valódi szoftvercsatornán fogadják a külső kommunikációs kapcsolaton (35) keresztül, majd demultiplexelik azt. A demultiplexelt kérelemadatot a megfelelő második virtuális szoftvercsatornához továbbítják, amelyen keresztül a kérelemadatot átadják a második alkalmazásnak.

A találmány tárgyát alkotja a fenti eljárást megvalósító berendezés, továbbá olyan számítógépi programtermék, amely számítógéppel olvasható tárolóközegén eltároltan tartalmazza az eljárás megvalósítását biztosító programot.

A találmány tárgya egyrészt egy eljárás, másrészt egy berendezés, harmadrészt egy számítógépi programtermék egy első számítógépen lévő első alkalmazástól egy kommunikációs kapcsolaton keresztül egy második számítógépen lévő második alkalmazáshoz egy külső kommunikációs kapcsolaton keresztül, TCP kommunikációs protokoll felhasználásával továbbított adatok mennyiségének csökkentésére. Találmányunk ezen belül elsősorban olyan ügyfél- és kiszolgálóalkalmazások közötti kommunikációval kapcsolatos, mint például egy webböngésző és egy webkiszolgáló. Pontosabban a találmány két számítógép közötti kis sebességű vagy vezeték nélküli kommunikációs kapcsolaton keresztüli kommunikációra vonatkozik, ahol az egyik számítógépen egy ügyfélalkalmazás, a másikon pedig egy kiszolgálóalkalmazás fut, mely alkalmazások a TCP/IP protokollt használják kommunikációs protokollként.

Az utóbbi időben nagy nyilvánosságot kapott „információs szupersztráda” növelte és erősítette az internet mint tömegkommunikációs média megértését és elfogadását. Az internetnek mint hálózatok közötti kommunikációra alkalmas médianak e széles körű elismerése egy olyan felhasználói bázist alakított ki, amely a számítógépes hálózatok közötti kommunikációt megvalósító, szabványos internetprotokollokra épül.

Az internet paradigmája az az ügyfél-kiszolgáló kapcsolat, ahol internetügyfelek (böngészők) kommunikálnak internetkiszolgálókkal. Azért, hogy egyre többen elérhessék az internetet, a kommunikációs protokollokat, valamint az ügyfelek és kiszolgálók által használt nyelveket egységesítették. Ilyen protokoll például az ügyfelek és kiszolgálók közötti kommunikációra alkalmas hiperszövegátviteli protokoll (http, hypertext transfer protocol), valamint az átvitelvezérlő protokoll/internetprotokoll (TCP/IP, transmission control protocol/internet protocol), melynek TCP-része a számítógépek vagy alkalmazások közötti kommunikációnak az adatátvitel-specifikus protokollja. Szintén szabványosított az a hiperszövegnyelvnek (HTML, hypertext markup language) nevezett nyelv, amellyel ügyfelek és kiszolgálók kommunikálnak egymással. Mivel ezek a protokollok és a nyelv géptípustól függetlenek,

és az információtovábbításhoz összeköttetésmentes, a legjobb minőségre törekvő, ám azt nem garantáló (ismert angol kifejezéssel best-efforts) protokollt használnak, minden egyes ügylet teljesen önállóan kezelhető. Így például az ügyféltől kiindul minden egyes üzenet információt tartalmaz a böngésző képességeiről, és független minden egyéb kommunikációtól. Az ügyfél és a kiszolgáló közötti kommunikációnak ezen önálló, független tulajdonsága, amely „állapot nélküli” vagy „állapotmentes” kommunikációnak nevezhető, növeli az adott kommunikáció alkalmával az ügyfél és a kiszolgáló között átvendő adatmennyiséget.

A világhálót (world wide web) használó ügyfél-kiszolgáló alkalmazások környezetében az ügyfél általában egy webböngészőt jelent, amely felhasználói felületként működik. A webböngésző elküldi a felhasználói kérélmeket a megfelelő webkiszolgálóhoz, valamint formázza és megjeleníti a kiszolgáló által visszaküldött HTML-adatokat. A webböngésző emellett meg is vizsgálja a HTML-adatokat, hogy megállapítsa, van-e benne beágyazott hipercsatolás, amely további böngészőkérélmeket tehet szükségessé, amelyeket a böngésző azután kezdeményez is. A webkiszolgáló az ügyfél kiszolgálójaként működik, feldolgozza a webböngésző kérélmeket, és a kért választ egy http-adatfolyam HTML-adatrészeként visszaküldi.

A world wide web kommunikáció egy tipikus példája, amikor a webböngésző egy honlapra irányuló kérélmeket bocsát ki, jól illusztrálja a http, HTML, TCP és a webböngésző, illetve -kiszolgáló közötti kapcsolatot. Amikor a webböngésző felhasználója információt kér egy bizonyos webhelyről, a webböngésző kommunikációt kezdeményez a kiszolgálóval úgy, hogy egy „megkapni” kérélmeket küld a webkiszolgálónak, megadva a keresett webhely leírására szolgáló egységes útvonalat (URL, universal resource locator), amely a példa kedvéért legyen egy honlap. Az URL tulajdonképpen a webhely címének szerepét tölti be, így minden webhelynek egyedi címe van az interneten. A webkiszolgáló ekkor megszerzi és elküldi a webböngészőnek az URL által meghatározott honlapnak megfelelő HTML-adatot. Ez a művelet a webkiszolgálót az interneten további kommunikációra készítheti, vagy az URL azonosíthat-

ja azon helyi hálózatban lévő kiszolgálót is, amelyikhez a böngésző is csatlakozik. A webböngésző ezután kéri a http-adatfolyamként megkapott HTML-adatot, hogy megállapítsa, van-e benne beágyazott hipercsatolás, például egy ikon vagy egy kép. Ha létezik ilyen hipercsatolás, akkor a webböngésző a meghatározott adat beszerzéséhez kérélmeket bocsát ki, meghatározva a hipercsatolás URL-jét. Ezt az adatot aztán a böngésző beágyazza a honlapba, és megjeleníti a felhasználó számára. Mint ebben az egyszerű példában látható, a webböngészőn keresztül megadott egyszeri felhasználói kérelem olyan többszöri kiegészítőkérelmeket eredményezhet, melyeket a webböngésző a felhasználói kérelemre válaszként kapott HTML-adatnak megfelelően automatikusan végrehajt.

Az 1. ábra egy, az internetre épülő ismert rendszer alapvető kommunikációs szerkezetét ábrázolja. Az 1. ábrán egy 10 webböngésző és egy 20 webkiszolgáló kommunikál egymással egy 15 kommunikációs kapcsolaton keresztül. Ez a kommunikációs kapcsolat általában egy helyi hálózati kapcsolat, nagy kiterjedésű hálózati kapcsolat, telefonvonalon keresztüli kapcsolat vagy a különböző kapcsolatfajták kombinációja lehet. A 10 webböngésző a 20 webkiszolgálóval a TCP/IP protokoll felhasználásával folytat kommunikációt. Az internetes kommunikációk többségénél a 10 webböngésző a 20 webkiszolgálóval a http általános kommunikációs protokoll felhasználásával kommunikál a 10 webböngésző és a 20 webkiszolgáló közötti TCP/IP kapcsolaton keresztül. A leírtak szerint a 10 webböngésző és 20 webkiszolgáló között átvitt vagy továbbított tényleges adatok tehát http-adatobjektumok, például HTML-adatok. A 20 webkiszolgáló egy proxy is lehet, amely különböző 10 webböngészőktől fogad kérélmeket, és irányítja azokat a megfelelő 20 webkiszolgálóhoz.

A webböngészők, webkiszolgálók, valamint közös információs és átviteli protokolljaik, a HTML és a http népszerűsége a webtechnológiának mint a hálózati információ-hozzáférés univerzális felületének rendkívül gyors elfogadásához vezetett. Mivel a webböngészők és webkiszolgálók közötti kommunikációs nyelv és protokollok szabványosítottak, függetlenül a felhasználó által az információs hálózat eléréséhez használt webböngésző programtól, a kommunikációs nyelv és protokollok ugyanazok lesznek. Ezért a webböngészők nagy felhasználói bázisa, az internetre csatlakozás lehetőségével és a webalkalmazásoknak a http által definiált, a http-szerver bővítését külső programokkal lehetővé tevő, mai napig legelterjedtebb felületen, a közös átjárófelület (CGI, common gateway interface) segítségével történő egyszerű megírhatóságával együtt nagyon vonzóvá teszik a webtechnológia formátumaira épülő alkalmazások nagy része számára.

Az internet népszerűségének és elfogadottságának növekedésével egyidejűleg a hordozható számítógépek népszerűsége is megnövekedett. A laptopok, a noteszgépek, az elektronikus személyi/kommunikációs titkárok (PDA/PCA) és egyéb hordozható eszközök használata a vezeték nélküli kommunikáció iránti igény növekedéséhez vezetett. A kiterjedt vezeték nélküli há-

lózatok, a mobiltelefonias kommunikáció és a csomagkapcsolt rádiófrekvenciás kommunikáció világhálós környezetben való használata azonban közös korlátokba ütközik. A kommunikáció bájtonkénti magas ára, a hosszú válaszidők, az alacsony sávszélesség és a megbízhatatlanság mind gátolják a vezeték nélküli technológia alkalmazását a world wide web állapotmentes kommunikációs protokolljának felhasználásával. Sőt, mivel a webprotokoll állapotmentes, a kérelenkénti adatmennyiségek és a vezeték nélküli kapcsolaton áthaladó kérelmek száma nagyobb, mint amire akkor lenne szükség, ha a kommunikáció nem lenne önálló. Tehát a vezeték nélküli technika vagy bármilyen más alacsony sebességű kommunikációs technika kombinálása a webtechnológiával célszerűtlennek tűnik, mivel a webtechnológia a maga univerzális természetével súlyosbítja a vezeték nélküli technológia gyengeségeit.

A fenti korlátozásokat szem előtt tartva jelen találmányunk egyik célja egy olyan kommunikációs eljárás létrehozása, amely csökkenti az alkalmazások közötti adatátvitel költségét.

Találmányunk további célja egy olyan kommunikációs eljárás és berendezés létrehozása, amely webböngésző/webkiszolgáló környezetben is használható.

Találmányunk további célja, hogy kompatibilis legyen az alacsony sebességű vagy vezeték nélküli kommunikációs rendszerekben létező kommunikációs protokollokkal és nyelvekkel anélkül, hogy a webböngésző és webkiszolgáló alkalmazásokat módosítani kellene.

Találmányunk további célja, hogy egy olyan kommunikációs megoldást biztosítson, amely csökkenti a szükséges kommunikáció mennyiségét a TCP/IP kommunikációs protokollt használó alkalmazások között, javítva ezzel a kommunikációs rendszer teljesítményét.

A kitűzött feladatot egyrészt egy eljárással oldottuk meg egy első számítógépen lévő első alkalmazástól egy kommunikációs kapcsolaton keresztül egy második számítógépen lévő második alkalmazáshoz egy külső kommunikációs kapcsolaton keresztül, TCP kommunikációs protokoll felhasználásával továbbított adatok csökkentésére, amelynek során az első számítógépen válaszul az első alkalmazás minden egyes csatlakozási kérelmére egy első virtuális szoftvercsatornát hozunk létre az első alkalmazástól kiinduló kérelem-adatok vételére; az első számítógépen egy első valódi szoftvercsatornát és a második számítógépen egy második valódi szoftvercsatornát hozunk létre az első számítógép és a második számítógép külső kommunikációs kapcsolaton keresztül történő összekapcsolására; a második számítógépen létrehozunk egy-egy második virtuális szoftvercsatornát az első alkalmazás által kibocsátott minden egyes csatlakozási kérelemhez úgy, hogy a második virtuális szoftvercsatornát egy, az első számítógépen az első alkalmazás csatlakozási kérelmére válaszképpen létrehozott első virtuális szoftvercsatornához rendeljük hozzá; az első virtuális szoftvercsatornához társított kérelemadatokat az első valódi szoftvercsatornára multiplexeljük; a multiplexelt kérelemadatokat a TCP protokoll felhasználásával a külső

kommunikációs kapcsolaton keresztül a második valódi szoftvercsatornához továbbítjuk; a multiplexelt kérelemadatokat a külső kommunikációs kapcsolaton keresztül fogadjuk; a külső kommunikációs kapcsolaton keresztül a második valódi szoftvercsatorna által fogadott kérelemadatokat demultiplexeljük; a demultiplexelt kérelemadatokat átadjuk annak a második virtuális szoftvercsatornának, amelyet az első alkalmazás által kibocsátott kérelemre válaszképpen létrehozott első virtuális szoftvercsatornához rendeltünk hozzá; a második virtuális szoftvercsatornán keresztül vett kérelemadatokat átadjuk a második alkalmazásnak; és az első valódi szoftvercsatornát és a második valódi szoftvercsatornát mindaddig fenntartjuk, amíg az első virtuális szoftvercsatornát létrehozó alkalmazáshoz tartozó kérelemadatokat át nem adjuk a második számítógépnek.

A találmány szerinti eljárás egy előnyös foganatosítási módja értelmében az első alkalmazás által kibocsátott kérelemhez tartozó második virtuális szoftvercsatornán keresztül az első alkalmazás által kibocsátott kérelemre a második alkalmazás által válaszképpen kibocsátott válaszadatokat fogadjuk; a második virtuális szoftvercsatornán keresztül vett válaszadatokat a második valódi szoftvercsatornára multiplexeljük; a multiplexelt válaszadatokat TCP protokoll felhasználásával a külső kommunikációs kapcsolaton keresztül továbbítjuk az első valódi szoftvercsatornához; a multiplexelt válaszadatokat vesszük a külső kommunikációs kapcsolaton keresztül; az első valódi szoftvercsatornán keresztül fogadott válaszadatokat demultiplexeljük; a demultiplexelt válaszadatokat az első alkalmazás által kibocsátott kérelemre vonatkozó válaszként adjuk az első alkalmazás által kibocsátott kérelemhez tartozó első virtuális szoftvercsatornának; és az első virtuális szoftvercsatornán keresztül vett válaszadatokat átadjuk az első alkalmazásnak.

A találmány szerinti eljárás egy további előnyös foganatosítási módja értelmében a válaszadatokat multiplexelése után a második virtuális szoftvercsatornát lezárjuk; és a válaszadatoknak az első alkalmazás részére történt átadását követően az első virtuális szoftvercsatornát is lezárjuk.

A találmány szerinti eljárás egy további előnyös foganatosítási módja értelmében az első valódi szoftvercsatorna és a második valódi szoftvercsatorna fenntartásához kapcsolódóan az első valódi szoftvercsatornát és a második valódi szoftvercsatornát az összes első virtuális szoftvercsatorna lezárását követően zárjuk le.

A találmány szerinti eljárás egy további előnyös foganatosítási módja értelmében az első valódi szoftvercsatorna és a második valódi szoftvercsatorna fenntartásához kapcsolódóan az első valódi szoftvercsatornát és a második valódi szoftvercsatornát az összes első virtuális szoftvercsatorna lezárását követő előre meghatározott ideig még fenntartjuk.

A találmány szerinti eljárás egy további előnyös foganatosítási módja értelmében az első alkalmazás webböngészőt és a második alkalmazás webkiszolgálót foglal magában.

A találmány szerinti eljárás egy további előnyös foganatosítási módja értelmében külső kommunikációs kapcsolatként vezeték nélküli kommunikációs kapcsolatot használunk.

5 A kitűzött feladatot másrészt egy berendezéssel oldottuk meg egy első számítógépen lévő első alkalmazástól egy kommunikációs kapcsolaton keresztül egy második számítógépen lévő második alkalmazáshoz egy külső kommunikációs kapcsolaton keresztül, TCP
10 kommunikációs protokoll felhasználásával továbbított adatok csökkentésére, amelynek az első alkalmazástól kiinduló kérelemadatok vételéhez az első alkalmazás minden egyes csatlakozási kérelmére válaszképpen az első számítógépen egy első virtuális szoftvercsatornát
15 létrehozó eszköze; az első számítógép és a második számítógép külső kommunikációs kapcsolaton keresztül történő összekapcsolásához az első számítógépen egy első valódi szoftvercsatornát létrehozó és a második
20 számítógépen egy második valódi szoftvercsatornát létrehozó eszköze; az első alkalmazás által kibocsátott minden egyes csatlakozási kérelemhez a második számítógépen egy-egy második virtuális szoftvercsatornát annak az egy, az első számítógépen az
25 első alkalmazás csatlakozási kérelmére válaszképpen létrehozott első virtuális szoftvercsatornához történő hozzárendelésével létrehozó eszköze, az első virtuális szoftvercsatornához társított kérelemadatokat az első
30 valódi szoftvercsatornára multiplexelő eszköze; a multiplexelt kérelemadatokat TCP protokoll felhasználásával a külső kommunikációs kapcsolaton keresztül a
35 második valódi szoftvercsatornához továbbító eszköze; a multiplexelt kérelemadatokat a külső kommunikációs kapcsolaton keresztül fogadó eszköze; a külső kommunikációs kapcsolaton keresztül a második valódi
40 szoftvercsatorna által fogadott kérelemadatokat demultiplexelő eszköze; a demultiplexelt kérelemadatokat az első alkalmazás által kibocsátott kérelemre válaszképpen létrehozott első virtuális szoftvercsatornához
45 hozzárendelt második virtuális szoftvercsatornának átadó eszköze; a második virtuális szoftvercsatornán keresztül vett kérelemadatokat a második alkalmazásnak átadó eszköze; és az első valódi szoftvercsatornát és a
50 második valódi szoftvercsatornát az első virtuális szoftvercsatornát létrehozó alkalmazáshoz tartozó kérelemadatokat második számítógépnek való átadásának befejezéséig fenntartó eszköze van.

A találmány szerinti berendezés egy előnyös kiviteli alakja értelmében a berendezésnek az első alkalmazás által kibocsátott kérelemhez tartozó második virtuális szoftvercsatornán keresztül az első alkalmazás
50 által kibocsátott kérelemre a második alkalmazás által válaszképpen kibocsátott válaszadatokat fogadó eszköze; a második virtuális szoftvercsatornán keresztül vett válaszadatokat a második valódi szoftvercsatornára
55 multiplexelő eszköze; a multiplexelt válaszadatokat TCP protokoll felhasználásával a külső kommunikációs kapcsolaton keresztül az első valódi szoftvercsatornához továbbító eszköze; a multiplexelt válaszadatokat a
60 külső kommunikációs kapcsolaton keresztül vevő eszköze; az első valódi szoftvercsatornán keresztül foga-

dott válaszadatokat demultiplexelő eszköze; a demultiplexelt válaszadatokat az első alkalmazás által kibocsátott kérelemre vonatkozó válaszként az első alkalmazás által kibocsátott kérelemhez tartozó első virtuális szoftvercsatornának átadó eszköze; és az első virtuális szoftvercsatornán keresztül vett válaszadatokat az első alkalmazásnak átadó eszköze van.

A kitűzött feladatot harmadrészt egy számítógépi programtermékkel oldottuk meg egy első számítógépen lévő első alkalmazástól egy kommunikációs kapcsolaton keresztül egy második számítógépen lévő második alkalmazáshoz egy külső kommunikációs kapcsolaton keresztül, TCP kommunikációs protokoll felhasználásával továbbított adatok csökkentésére. A programtermék számítógéppel olvasható tárolóközegén eltárolt tartalmaz egy, az első alkalmazástól kiinduló kérelemadatokat vételéhez az első alkalmazás minden egyes csatlakozási kérelmére válaszképpen az első számítógépen egy első virtuális szoftvercsatornát létrehozó; az első számítógép és a második számítógép külső kommunikációs kapcsolaton keresztül történő összekapcsolásához az első számítógépen egy első valódi szoftvercsatornát létrehozó és a második számítógépen egy második valódi szoftvercsatornát létrehozó; az első alkalmazás által kibocsátott minden egyes csatlakozási kérelemhez a második számítógépen egy-egy második virtuális szoftvercsatornát annak az egy, az első számítógépen az első alkalmazás csatlakozási kérelmére válaszképpen létrehozott első virtuális szoftvercsatornához történő hozzárendelésével létrehozó, az első virtuális szoftvercsatornához társított kérelemadatokat az első valódi szoftvercsatornára multiplexelő; a multiplexelt kérelemadatokat TCP protokoll felhasználásával a külső kommunikációs kapcsolaton keresztül a második valódi szoftvercsatornához továbbító; a multiplexelt kérelemadatokat a külső kommunikációs kapcsolaton keresztül fogadó; a külső kommunikációs kapcsolaton keresztül a második valódi szoftvercsatorna által fogadott kérelemadatokat demultiplexelő; a demultiplexelt kérelemadatokat az első alkalmazás által kibocsátott kérelemre válaszképpen létrehozott első virtuális szoftvercsatornához hozzárendelt második virtuális szoftvercsatornának átadó; a második virtuális szoftvercsatornán keresztül vett kérelemadatokat a második alkalmazásnak átadó; és az első valódi szoftvercsatornát és a második valódi szoftvercsatornát az első virtuális szoftvercsatornát létrehozó alkalmazáshoz tartozó kérelemadatokat második számítógépnek való átadásának befejezéséig fenntartó programot.

Jelen találmányunk csökkenti az egy első számítógépen lévő első alkalmazástól egy kommunikációs kapcsolaton keresztül adatátvitelt egy második számítógépen lévő második alkalmazáshoz, ahol az adatok egy külső kommunikációs kapcsolaton keresztül kerülnek továbbításra az első számítógéptől a második számítógéphez a TCP kommunikációs protokoll felhasználásával.

Amint az a szakember számára nyilvánvaló, a találmány fent vázolt előnyös tulajdonságai különböző be-

rendezésekkel, valamint számítógéppel olvasható programkódokkal is megvalósíthatók.

A találmányt az alábbiakban a csatolt rajz segítségével mutatjuk be részletesebben, amelyen a javasolt eljárás példakénti fogatosítási módját, illetve a javasolt berendezés példakénti kiviteli alakját tüntettük fel. A rajzon az

1. ábra egy ismert, jellemzőnek mondható webböngésző-webkiszolgáló rendszer tömbvázlata, a
2. ábrán a találmány szerinti webböngésző-webkiszolgáló rendszer ügyféloldali elfogást és kiszolgálóoldali elfogást alkalmazó lehetséges kiviteli alakjának tömbvázlatát tüntettük fel, a
3. ábrán az ügyféloldali elfogómodullal végzett műveleteket ábrázoló folyamatábra látható a találmány egy előnyös, koherens gyorsítótárrendszert magában foglaló megvalósításában, a
4. ábrán az ügyféloldali elfogómodullal végzett műveleteket ábrázoló folyamatábra látható a találmány egy további, előnyös, koherens gyorsítótárrendszert magában foglaló megvalósításában, az
5. ábra a kiszolgálóoldali elfogómodullal végzett műveleteket ábrázoló folyamatábrát mutat a találmány egy előnyös, koherens gyorsítótárrendszert magában foglaló megvalósításában, a
6. ábra a kiszolgálóoldali elfogómodullal végzett műveleteket ábrázoló folyamatábrát mutat a találmány egy további, előnyös, koherens gyorsítótárrendszert magában foglaló megvalósításában, a
7. ábra az ügyféloldali elfogómodullal végzett műveleteket ábrázoló folyamatábrát mutat a találmány egy előnyös, különbözeti adatátviteli rendszert magában foglaló megvalósításában, a
8. ábra az ügyféloldali elfogómodullal végzett műveleteket ábrázoló folyamatábrát mutat a találmány egy további, előnyös, különbözeti adatátviteli rendszert magában foglaló megvalósításában, a
9. ábra a kiszolgálóoldali elfogómodullal végzett műveleteket ábrázoló folyamatábrát mutat a találmány egy előnyös, különbözeti adatátviteli rendszert magában foglaló megvalósításában, a
- 10A., 10B. ábrán a kiszolgálóoldali elfogómodullal végzett műveleteket ábrázoló folyamatábra látható a találmány egy további, előnyös, különbözeti adatátviteli rendszert magában foglaló megvalósításában, a
11. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakjának tömbvázlatát mutatja, a
12. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges

kiviteli alakja ügyféloldali elfogómoduljának és kiszolgálóoldali elfogómoduljának tömbvázlátát mutatja, a

- 13A., 13B. ábrán a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakja ügyféloldali elfogómoduljának vagy kiszolgálóoldali elfogómoduljának szoftvercsatorna-menedzsere által végzett műveleteket ábrázoló folyamatábra látható, a
14. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakjában egy ügyféloldali elfogófunkció által végrehajtott műveleteket ábrázoló folyamatábra, a
15. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakjában egy kiszolgálóoldali elfogófunkció által végrehajtott műveleteket ábrázoló folyamatábra, a
- 16–1. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakja által végrehajtott virtuális létrehozás műveletet ábrázoló folyamatábrát mutat, a
- 16–2. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakja által végrehajtott virtuális küldés műveletet ábrázoló folyamatábrát mutat, a
- 16–3. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakja által végrehajtott virtuális fogadás műveletet ábrázoló folyamatábrát mutat, a
- 16–4. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakja által végrehajtott virtuális kiválasztás műveletet ábrázoló folyamatábrát mutat, a
- 17–1. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakja által végrehajtott virtuális kiürítés műveletet ábrázoló folyamatábrát mutat, és a
- 17–2. ábra a találmány szerinti berendezés virtuális szoftvercsatornákat használó lehetséges kiviteli alakja által végrehajtott virtuális lezárás műveletet ábrázoló folyamatábrát mutat.

A továbbiakban találmányunkat részletesebben is bemutatjuk a mellékelt rajz segítségével, amely a találmány néhány lehetséges kiviteli alakját, illetve megvalósítását ábrázolja. Találmányunk számos különböző kiviteli alakban megvalósítható, így ez a leírás nem értelmezhető kizárólagos vagy korlátozó módon. A bemutatásra kerülő példák csak azt a célt szolgálják, hogy a találmány bemutatása lehetőleg teljes és mélyreható legyen, és a találmány oltalmi körét a szakember számára jól érthetővé tegyék. Az azonos hivatko-

zási jelek leírásunkban végig azonos részegységeket, összetevőket jelentenek.

Az 1. ábrát leírásunknak a technika állását bemutató részében már ismertettük.

- 5 A 2. ábra a találmány szerinti berendezés egy lehetséges kiviteli alakját mutatja. A 10 webböngésző egy ügyféloldali 30 elfogómodullal, a 20 webkiszolgáló pedig egy kiszolgálóoldali 40 elfogómodullal áll kapcsolatban. Az ügyféloldali 30 elfogómodul az 1. ábra 10 15 kommunikációs kapcsolatát megvalósító 35 kommunikációs kapcsolaton keresztül kommunikál a kiszolgálóoldali 40 elfogómodullal. A 10 webböngésző és az ügyféloldali 30 elfogómodul egy első 5 számítógépben van kialakítva. A kiszolgálóoldali 40 elfogómodul és a 15 20 webkiszolgálót egy második 6 számítógép tartalmazza. Az első 5 számítógép és a második 6 számítógép a külső 35 kommunikációs kapcsolaton keresztül áll egymással kommunikációs kapcsolatban.

- A 10 webböngésző egy internetböngésző, és hiperszo-
20 perszövegátviteli protokollt (http), valamint hiperszo-
vegyelvet (HTML) használva kommunikál a szintén http-t és HTML-t használó internetes 20 webkiszolgálóval. Működés közben a 10 webböngésző egy http-adatfolyamot bocsát ki, amelyet az ügyféloldali 30 elfogómodul elfog. Az ügyféloldali 30 elfogómodul a 25 http-adatfolyamot például a TCP/IP visszahurkolási tulajdonsága révén foghatja el, ha az ügyféloldali 30 elfogómodul 127-es, például a 127.0.0.1. hálózati számú IP-címen található. Az ügyféloldali 30 elfogómodul 30 a http-adatfolyamot ügyfél-kiszolgáló specifikus protokollra alakítja át, és ezt az ügyfél-kiszolgáló specifikus adatfolyamot továbbítja a külső 35 kommunikációs kapcsolatra. A kiszolgálóoldali 40 elfogómodul fogadja az ügyfél-kiszolgáló specifikus adatfolyamot, és helyreállítja a 10 webböngészőtől kiinduló kommunikációnak megfelelő eredeti http-adatfolyamot. Ezt a visszaállított http-adatfolyamot továbbítja a 20 webkiszolgálónak. A 20 webkiszolgáló a http-adatfolyamra az internetes 20 webkiszolgálók hagyományos módján válaszol. Amint ez a szakember számára nyilvánvaló, a 40 20 webkiszolgáló akár egy proxy is lehet, amely lehetővé teszi több 10 webböngésző csatlakozását az internetre.

- Amikor a 20 webkiszolgáló a 10 webböngészőnek 45 továbbítandó információt fogad, például a 10 webböngésző egy bizonyos URL honlapra vonatkozó kérelmére válaszolva, akkor kibocsát egy, a 10 webböngészőnek küldendő kommunikációnak megfelelő http-adatfolyamot. Ezt a 20 webkiszolgálótól kiinduló kommunikációt a kiszolgálóoldali 40 elfogómodul elfogja, és átalakítja az ügyfél-kiszolgáló specifikus adatfolyammá. A 20 webkiszolgálótól kiinduló kommunikációnak megfelelő ügyfél-kiszolgáló specifikus adatfolyamot a külső 35 kommunikációs kapcsolaton keresztül a második 6 számítógéptől az első 5 számítógéphez küldi. Az ügyfél-kiszolgáló specifikus adatfolyamot az ügyféloldali 30 elfogómodul fogadja, és a webkiszolgálótól kiinduló kommunikációnak megfelelő eredeti http-adatfolyammá állítja vissza, majd továbbítja a 10 webböngészőnek. 60

A találmány szerinti berendezés egy további lehetséges kiviteli alakja értelmében a külső 35 kommunikációs kapcsolat egy vezeték nélküli kommunikációs kapcsolat. Ilyen esetben, hogy a felhasználók számára elfogadható legyen a rendszer teljesítménye, kívánatos a külső 35 kommunikációs kapcsolaton történő átvitelek mértékét csökkenteni, mind az átvitelek gyakoriságát, mind a továbbítandó információk mennyiségét illetően. Ennek megfelelően a jelen találmány gyorsítási, megkülönböztetési és protokollcsökkentési technikákat alkalmaz, a külső 35 kommunikációs kapcsolaton keresztüli kommunikációk mennyiségének minimalizálására. Ezeket a technikákat a http hontalan vagy statisztikai valószínűségeen alapuló (sztochasztikus) protokollnak ügyfél-kiszolgáló specifikus protokollra történő átalakításával valósítja meg. Az ügyfél-kiszolgáló specifikus protokoll az ügyfélre és a kiszolgálóra jellemző információkat használja az átvitel gyakoriságának és mennyiségének csökkentésére.

Bár jelen találmány az eddigiekben és a továbbiakban is egy 10 webböngészőre és egy 20 webkiszolgálóra vonatkoztatva kerül leírásra, de amint az a szakember számára nyilvánvaló, a jelen találmány előnyei és szolgáltatásai több 10 webböngésző egy 20 webkiszolgálóhoz való csatlakoztatása révén is élvezhetők. Így a jelen találmány módszerei, szerkezete és programjai több böngészővel úgy működnek, hogy a böngészők ügyféloldali 30 elfogómodulokkal kommunikálnak, és ezek az ügyféloldali 30 elfogómodulok a 20 webkiszolgáló vagy a webproxy kiszolgálóoldali 40 elfogómoduljával állnak kapcsolatban.

Jelen találmány egyik előnyös megvalósításában mind az ügyféloldali 30 elfogómodul, mind a kiszolgálóoldali 40 elfogómodul rendelkezik gyorsítótárral. Az első 5 számítógépben lévő ügyfélgyorsítótár a 10 webböngészőtől kiinduló kommunikációra válaszként szolgál, a 10 webböngésző által fogadható http-adatfolyamokat tárol. A második 6 számítógépben lévő kiszolgáló-gyorsítótár olyan http-adatfolyamokat tárol, amelyeket a 20 webkiszolgálótól kapott egy 10 webböngészőtől kiinduló kommunikációra adott válaszként.

Amint ez szakember számára nyilvánvaló, az első 5 számítógépben, illetve a második 6 számítógépben található gyorsítótár mérete a számítógépek egyedi hardverkiépítésétől függően bármekkora lehet. Ezek a gyorsítótárak minden egyes átvitelre vonatkozóan tartalmaznak információkat, beleértve a kommunikáció URL-jét, egy, a kommunikáció tartalmán alapuló egyedi azonosítót, mint például kommunikáció adatainak ciklikus redundanciakódját (CRC), a tárolás időpontját (SDT, store date time), amely a gyorsítótár létrejöttének vagy frissítésének időpontját jelzi, valamint az átvitel adatait. Így egy, a gyorsítótárban tárolt minden egyes kommunikációra vonatkozó tárbejegyzéskönyvtárat hozhatunk létre. Továbbá a bármely hardver kiépítésénél meglévő korlátozott erőforrások miatt a szakember által ismert összes olyan gyorsítótár-technika alkalmazható, amely segíti az első és a második 5, 6 számítógépben található gyorsítótárak fenntartását. Így például a gyorsítótár érvénytelenítheti a legkorábbi

könyvtárbejegyzést, ha az újabb bejegyzés hozzáadása meghaladná a felhasználó által meghatározott gyorsítótár méretét, és az új bejegyzést az érvénytelenített bejegyzés helyére teheti. Ezenfelül a gyorsítótár bejegyzések fenntarthatók több párhuzamosan futó 10 webböngésző vagy 20 webkiszolgáló alkalmazás számára, vagy akár az első, illetve második 5, 6 számítógép ki- és bekapcsolásakor is létrehozhatók, hogy általában ismétlődő gyorsítótár jöjjön létre.

A 3–10. ábrákon, valamint a 13–17–2. ábrákon a találmány szerinti eljárás néhány lehetséges fogantatási módjának jellemző folyamatábráit tüntettük fel. A folyamatábrák minden egyes lépése és a lépések kombinációi számítógépi programkódként, utasításként is megvalósíthatók. Ezek a számítógépes programutasítások számítógépbe vagy egyéb programozható készülékbe tölthetők, hogy a számítógépen vagy a programozható készülékben végrehajtott utasítások a folyamatábrák lépéseivel meghatározott feladatok megvalósításának eszközeül szolgáljanak. Ezek a számítógépes programutasítások természetesen számítógéppel olvasható memóriában is tárolhatók, amely úgy képes vezérelni a számítógépet vagy az egyéb programozható készüléket, hogy meghatározott módon üzemelve a számítógéppel olvasható memóriában tárolt utasításokkal például gyártható árucikket állítsunk elő, olyan utasítási módszereket is ideértve, amelyek megvalósítják a folyamatábrák lépéseivel meghatározott feladatokat. A számítógépes programutasítások úgy is betölthetők számítógépbe vagy az egyéb programozható készülékbe, hogy az azáltal végrehajtandó műveleti lépések sorozatát biztosítsák, olyan, számítógép által végrehajtott folyamatot eredményezve, hogy azok az utasítások, amelyek a számítógépen vagy egyéb programozható készülékben végrehajtásra kerülnek, a folyamatábrák lépéseivel meghatározott feladatok megvalósításának egyes mozzanatait adják.

Ennek megfelelően a folyamatábrák meghatározott feladat elvégzésére vonatkozó módszerek, valamint lépések kombinációit is bemutatják. Szintén nyilvánvaló, hogy a folyamatábrák minden egyes lépése és azok kombinációi megvalósíthatók a meghatározott feladatokat vagy lépéseket elvégző különleges célhardver-alapú számítógéprendszerekkel vagy különleges célhardver és számítógépi utasítások kombinációjával.

A továbbiakban az ügyféloldali 30 elfogómodul és a kiszolgálóoldali 40 elfogómodul működését leíró 3–6. ábrák segítségével a találmány egy lehetséges kiviteli alakja szerinti gyorsítótárak szerkezeti felépítését és működését mutatjuk be.

A 3. ábrán a 100 lépésben az ügyféloldali 30 elfogómodullal kérelmet fogadunk a 10 webböngészőtől. Ez a kérelem történhet http-adatfolyam formájában. A 105 lépésben az ügyféloldali 30 elfogómodullal ellenőrizzük a beérkezett kérelem univerzális forrásmeghatározóját (URL), amelynek alapján az ügyféloldali 30 elfogómodullal megállapítjuk, hogy a 10 webböngészőtől kiinduló kérelemnek megfelelő információ már korábban eltárolásra került-e az első 5 számítógépben található ügyfélgyorsítótárban. Ha az URL-nek megfelelő

információt korábban még nem tároltuk el az ügyfélgyorsítótárban, akkor a 106 lépésben az ügyféloldali 30 elfogómodullal a külső 35 kommunikációs kapcsolaton keresztül egy kérelmet továbbítunk a kiszolgálóoldali 40 elfogómodulnak. Ha azonban a 105 lépésben a 10 webböngészőtől kiinduló kommunikáció vizsgálata során kiderül, hogy már létezik a 10 webböngészőtől kiinduló kommunikációnak megfelelő ügyfélgyorsítótár-bejegyzés, akkor a 111 lépésben a legegyszerűbb esetben ezt az információt http-adatfolyamként továbbítjuk a 10 webböngészőnek. Ezen túlmenően a 110 lépésben a bemutatott legegyszerűbb esetben is elvégezzük a 10 webböngészőtől kiinduló kommunikációnak megfelelő gyorsítótár bejegyzésen a leírásunk további részében koherens intervallum-ellenőrzésként említett műveletet.

Az ügyféloldali 30 elfogómodul koherens intervallumának értékét a felhasználó is meghatározhatja. Ez az érték azt az időtartamot jelenti, ameddig a gyorsítótár bejegyzés létezhet, mielőtt elavulnak a benne tárolt adatok. Még ha adott is ez az érték, frissíteniük kell a 20 webkiszolgálótól kért és a 10 webböngészőtől eredő kommunikációra vonatkozó információval. A 110 lépésben látható koherens intervallum-ellenőrzés az aktuális dátum és idő, valamint a 10 webböngészőtől kiinduló kommunikációnak megfelelő gyorsítótár bejegyzés tárolási időpontjának és a felhasználó által meghatározott koherens intervallum értékének összehasonlítása révén lehetséges. Ha az aktuális dátum és idő nagyobb, mint ez az érték, akkor a 10 webböngészőtől kiinduló kommunikációnak megfelelő gyorsítótárban tárolt információk elavulttá váltak és a 110 lépésről a „nem” ágon haladunk tovább. Ha azonban az aktuális dátum és idő kisebb, mint a tárolási időpont és a felhasználó által meghatározott koherens intervallum összege, akkor a 110 lépés „igen” ágát választjuk, és a 111 lépésben a gyorsítótár bejegyzést a böngészőnek http-adatfolyamként továbbítjuk. Ezzel fejezzük be a 3. ábra 100 lépésében az ügyféloldali 30 elfogómodullal fogadott, a 10 webböngészőtől kiinduló kommunikációt.

Ha a 110 lépésben látható koherens intervallum-ellenőrzés során azt állapítjuk meg, hogy az első 5 számítógépben található gyorsítótár bejegyzés elavult, akkor a 112 lépésben kérelmet küldünk a kiszolgálóoldali 40 elfogómodulnak, hogy a második 6 számítógépben található gyorsítótár bejegyzést is ellenőrizze. Ez a 10 webböngészőtől kiinduló http-kérelemnek és a 10 webböngészőtől kiinduló kommunikáció URL-jének megfelelő ügyfélgyorsítótár tartalmáról szóló különleges megkülönböztető jelnek a külső 35 kommunikációs kapcsolaton keresztül a kiszolgálóoldali 40 elfogómodulhoz való elküldésével történhet. Egy előnyös megvalósításban ez a különleges megkülönböztető jel a gyorsítótár bejegyzés ciklikus redundanciakódjának (CRC) az eredménye.

Az 5. ábra a kiszolgálóoldali 40 elfogómodullal a külső 35 kommunikációs kapcsolaton keresztül az ügyféloldali 30 elfogómodultól fogadott információk hatására végzett műveleteket ábrázolja. Amikor a kiszolgálóoldali 40 elfogómodullal az ügyféloldali 30 elfogómodultól ké-

relmet fogadunk, akkor a 120 lépésben a kiszolgálóoldali 40 elfogómodullal fogadjuk az előre meghatározott ügyfél-koherensintervallumot, az ügyfélgyorsítótár bejegyzés ciklikusredundanciakód-értékét és a 10 webböngészőtől kiinduló http-kérelmet.

Az ügyféloldali 30 elfogómodul információinak fogadása után a 125 lépésben a kiszolgálóoldali 40 elfogómodullal a második 6 számítógépben található gyorsítótár ellenőrzük annak megállapítására, hogy létezik-e már a 10 webböngészőtől kiinduló http-kérelem URL-jének megfelelő kiszolgáló-gyorsítótár bejegyzés. Ha a 10 webböngészőtől kiinduló kommunikáció vizsgálata során a kiszolgálóoldali 40 elfogómodullal megállapítjuk, hogy már létezik a 10 webböngészőtől kiinduló kommunikáció által kérelmezett információnak megfelelő gyorsítótár bejegyzés, akkor a 125 lépésről az „igen” ágon haladunk tovább. Ezután a 130 lépésben a kiszolgálóoldali 40 elfogómodullal összehasonlítjuk a saját aktuális dátumot és időt a 10 webböngészőtől kiinduló kommunikáció által kért információnak megfelelő kiszolgáló-gyorsítótár bejegyzés tárolási időpontjának és az ügyféloldali 30 elfogómodultól fogadott, előre meghatározott koherens intervallumnak az értékével.

Ha az aktuális idő és dátum kisebb, mint a kiszolgáló-gyorsítótár bejegyzés tárolási időpontjának és a koherens intervallumnak az összege, akkor az 5. ábra 130 lépéséről az „igen” ágon haladunk tovább. Ezután a 135 lépésben a kiszolgálóoldali 40 elfogómodullal összehasonlítjuk a kiszolgáló-gyorsítótár bejegyzés ciklikus redundanciakódját az ügyfélgyorsítótár bejegyzés ciklikus redundanciakódjával annak meghatározására, hogy a két gyorsítótár bejegyzés azonos-e. Ha azonos, akkor a 136 lépésben egy „összetartozó” választ küldünk az ügyféloldali 30 elfogómodulnak.

Ha a 135 lépésben elvégzett vizsgálat azt mutatja, hogy a két ciklikus redundanciakód nem egyenlő, akkor az ügyfélgyorsítótárban és a kiszolgáló-gyorsítótárban tárolt információk nem azonosak. Ezért a 137 lépésben a kiszolgálóoldali 40 elfogómodullal elküldjük a kiszolgáló-gyorsítótár bejegyzést a külső 35 kommunikációs kapcsolaton keresztül az első 5 számítógépnek. A kiszolgáló-gyorsítótár bejegyzésnek az ügyféloldali 30 elfogómodulnak való elküldése során a kiszolgálóoldali 40 elfogómodullal a bejegyzést átalakítjuk ügyfél-specifikus kommunikációs protokollra, amely tartalmazza a kiszolgáló-gyorsítótár bejegyzés ciklikus redundanciakódját, a kiszolgáló-gyorsítótár bejegyzés adatait és a kiszolgáló-gyorsítótár bejegyzés korát. A kiszolgáló-gyorsítótár bejegyzés kora kiszámítható úgy is, hogy a gyorsítótár bejegyzés tárolási időpontját kivonjuk az aktuális dátumból és időből.

Végül ami az 5. ábrát illeti, ha akár a tárolási időpont és az előre meghatározott ügyfél-koherensintervallum összege kisebb, mint az aktuális dátum és idő, akár nem létezik a 10 webböngészőtől kiinduló kommunikáció URL-jének megfelelő gyorsítótár bejegyzés, akkor a 130 lépésről, illetőleg a 125 lépésről a „nem” ágon haladunk tovább. Ebben az esetben a 126 lépés műveleteit hajtjuk végre, és a kiszolgálóoldali 40 elfo-

gómodullal a 20 webkiszolgálónak a 10 webböngészőtől kiinduló kommunikációt http-adatfolyamként küldjük el. Ha a kiszolgálóoldali 40 elfogómodullal a 10 webböngészőtől kiinduló kommunikációt http-adatfolyamként kell elküldenünk a 20 webkiszolgálónak, akkor a kiszolgálóoldali 40 elfogómodullal a 6. ábrán látható műveleteket hajtjuk végre.

Válaszképpen a 10 webböngészőtől kiinduló kommunikációra, a 140 lépésben a kiszolgálóoldali 40 elfogómodullal a 20 webkiszolgálótól http-adatfolyamot fogadunk. A http-adatfolyam megérkezésekor a kiszolgálóoldali 40 elfogómodullal kiszámítjuk a http-adatfolyam ciklikus redundanciakódját, és átmenetileg tároljuk a http-adatfolyamot. Ezután a 145 lépésben a kiszolgálóoldali 40 elfogómodullal megvizsgáljuk a http-adatfolyamot, hogy létezik-e már a http-adatfolyam URL-jének megfelelő kiszolgáló-gyorsítótár bejegyzés. Ha létezik ilyen bejegyzés, akkor a 145 lépésről az „igen” ágon haladunk tovább. A kiszolgálóoldali 40 elfogómodullal ezután a 150 lépésben összehasonlítjuk a 20 webkiszolgálótól fogadott http-adatfolyam újonnan kiszámolt ciklikus redundanciakódját a 20 webkiszolgálótól kiinduló válaszkommunikáció URL-jének megfelelő kiszolgáló-gyorsítótár bejegyzés ciklikus redundanciakódjával. Ha a ciklikus redundanciakódok egyeznek, akkor a 150 lépésről az „igen” ágon haladunk tovább. A 151 lépésben a kiszolgálóoldali 40 elfogómodullal aktualizáljuk a kiszolgáló-gyorsítótár bejegyzés tárolási időpont bejegyzését, és a 20 webkiszolgáló által fogadott http-adatfolyamot a 152 lépésben kiürítjük az átmeneti tárból.

Ha a ciklikusredundanciakód-összehasonlítás eredménye azt mutatja, hogy a kiszolgáló-gyorsítótár bejegyzés eltér a 20 webkiszolgálótól fogadott http-adatfolyamtól, akkor a 150 lépésről a „nem” ágon haladunk tovább. A 153 lépésben a kiszolgálóoldali 40 elfogómodullal eltávolítjuk a kiszolgáló-gyorsítótárból az adatokat, majd a 154 lépésben az újabb adatokkal aktualizáljuk a kiszolgáló-gyorsítótárát. Ez az aktualizálás magában foglalja a 20 webkiszolgálótól kiinduló kommunikáció ciklikus redundanciakódjának a kiszolgáló-gyorsítótárban való eltárolását, az aktuális dátum és idő gyorsítótár bejegyzés gyorsítótár bejegyzésben való tárolását és a http-adatfolyam tárolását. Mindegyik esetben, akár aktualizáljuk a kiszolgáló-gyorsítótár bejegyzést, akár a 20 webkiszolgálótól fogadott http-adatfolyammal azonosnak bizonyul a kiszolgáló-gyorsítótár bejegyzés, a 155 lépésben a kiszolgálóoldali 40 elfogómodullal megvizsgáljuk, hogy a kiszolgáló-gyorsítótár bejegyzés megegyezik-e a 10 webböngészőtől kiinduló kommunikációnak megfelelő ügyfélgyorsítótár bejegyzéssel.

Ha a kiszolgálóoldali 40 elfogómodullal azt állapítjuk meg, hogy a 20 webkiszolgálótól fogadott válasznak megfelelő gyorsítótár bejegyzés nem létezik, akkor a 145 lépésről a „nem” ágon haladunk tovább. A 146 lépésben egy gyorsítótár bejegyzést hozunk létre a 20 webkiszolgálótól fogadott válasz URL-jének, a 20 webkiszolgálótól fogadott válasz ciklikus redundanciakódjának, a http-adatfolyamnak és az aktuális dátum-

és időtárolás időpontként történő eltárolásával. A 10 webböngészőtől kiinduló kommunikációnak megfelelő gyorsítótár bejegyzés elkészítése után a 155 lépésben a kiszolgálóoldali 40 elfogómodullal ismét összehasonlítjuk ennek a gyorsítótár bejegyzésnek a ciklikus redundanciakódját az ügyfélgyorsítótár bejegyzés ciklikus redundanciakódjával.

Ha a kiszolgáló-gyorsítótár bejegyzés és az ügyfélgyorsítótár bejegyzés összehasonlításának eredménye azt jelzi, hogy a gyorsítótár bejegyzések azonosak, akkor a 155 lépésről az „igen” ágon haladunk tovább. A 156 lépésben a kiszolgálóoldali 40 elfogómodullal egy időben „összetartozó” választ küldünk az ügyféloldali 30 elfogómodulnak. A kiszolgálóoldali 40 elfogómodullal a kiszolgálókérelem gyorsítótár bejegyzést egy ügyfélkiszolgáló specifikus adatfolyammá alakítjuk át az ügyféloldali elfogómodulnak küldött időben „összetartozó”, „nullakorú” válasz elküldésével.

Ha a kiszolgálóoldali 40 elfogómodullal megállapítjuk, hogy az ügyfélgyorsítótár bejegyzés nem azonos a 10 webböngészőtől kiinduló kommunikációnak megfelelő kiszolgáló-gyorsítótár bejegyzéssel, akkor a 155 lépésről a „nem” ágon haladunk tovább. A 157 lépésben a kiszolgálóoldali 40 elfogómodullal a kiszolgáló-gyorsítótár bejegyzést ügyfélkiszolgáló specifikus adatfolyammá alakítjuk át. Az adatfolyam tartalmazza a kiszolgáló-gyorsítótár bejegyzés ciklikus redundanciakódját, a kiszolgáló-gyorsítótár bejegyzés http-adatfolyamot, és a gyorsítótár bejegyzés korát, ami nullára van állítva. Ezt az ügyfélkiszolgáló-specifikus kommunikációt a külső 35 kommunikációs kapcsolaton keresztül az ügyféloldali 30 elfogómodulnak továbbítjuk.

A továbbiakban a 4. ábra segítségével az ügyféloldali 30 elfogómodul működését mutatjuk be egy, a kiszolgálóoldali 40 elfogómodultól fogadott kommunikációt követően. A 160 lépésben az ügyféloldali 30 elfogómodullal fogadjuk vagy beszerezzük a külső 35 kommunikációs kapcsolaton keresztül az ügyfél-kiszolgáló specifikus adatfolyamot. Az ügyféloldali 30 elfogómodullal ezt követően a 165 lépésben megállapítjuk, hogy milyenfajta választ fogadtunk a kiszolgálóoldali 40 elfogómodultól. Ha a kiszolgálóoldali 40 elfogómodul azt jelzi, hogy az ügyfélgyorsítótár bejegyzés időben összetartozó, azaz a kiszolgáló-gyorsítótár bejegyzés és az ügyfélgyorsítótár bejegyzés azonos, akkor a 166 lépésben az ügyféloldali 30 elfogómodullal aktualizáljuk a 10 webböngészőtől kiinduló kommunikációnak megfelelő ügyfélgyorsítótár bejegyzés tárolás időpontját a kiszolgálóoldali 40 elfogómodultól fogadott kor és az aktuális dátum és idő különbsége útján. Így az első 5 számítógép és a második 6 számítógép óráinak összehangolása nélkül találmányunk révén ki tudjuk javítani az első 5 számítógép gyorsítótár bejegyzésének összetartozási idejét úgy, hogy az a második 6 számítógép újabb adatait tükrözze. Miután a 10 webböngészőtől kiinduló kommunikációnak megfelelő ügyfélgyorsítótár bejegyzés tárolás időpontját aktualizáltuk, az ügyféloldali 30 elfogómodullal a gyorsítótár bejegyzést a 174 lépésben http-adatfolyamként továbbítjuk a 10 webböngészőnek.

Ha az ügyféloldali 30 elfogómodullal azt állapítjuk meg, hogy a válasz típusa adat vagy adatfolyamválasz, akkor a 165 lépésről a „folyam” ágon haladunk tovább. A 167 lépésben az ügyféloldali 30 elfogómodullal fogadjuk a http-adatfolyamot, és átmenetileg tároljuk ezt az adatot. Ezután a 170 lépésben az ügyféloldali 30 elfogómodullal megvizsgáljuk, hogy létezik-e a 10 webböngészőtől kiinduló kommunikációnak megfelelő gyorsítótár bejegyzés. Ha létezik ilyen gyorsítótár bejegyzés, akkor a 170 lépésről az „igen” ágon haladunk tovább, és a 171 lépésben a létező gyorsítótár bejegyzést kiürítjük. Az ügyféloldali 30 elfogómodullal ezután a 172 lépésben aktualizáljuk a 10 webböngészőtől kiinduló kommunikációnak megfelelő ügyfélgyorsítótár bejegyzést a kiszolgálóoldali 40 elfogómodultól fogadott http-adatfolyam ciklikus redundanciakódjának tárolásával, a kiszolgálóoldali 40 elfogómodultól fogadott kor és az aktuális dátum és idő különbségének tárolási időpontként való eltárolásával és a http-adatfolyam tárolásával.

Ha a 10 webböngészőtől kiinduló kommunikációnak megfelelő gyorsítótár bejegyzés nem létezik, akkor a 170 lépésről a „nem” ágon haladunk tovább, és a 173 lépésben az ügyféloldali 30 elfogómodullal egy ügyfélgyorsítótár bejegyzést készítünk a kiszolgálóoldali 40 elfogómodultól fogadott http-adatfolyam URL-jének tárolásával, a kiszolgálóoldali 40 elfogómodultól fogadott http-adatfolyam ciklikus redundanciakódjának tárolásával és a http-adatfolyam tárolásával. Az ügyféloldali 30 elfogómodullal szintén aktualizáljuk vagy tároljuk a tárolási időpontot, kivonva az aktuális dátumból és időből a külső 35 kommunikációs kapcsolaton keresztül a kiszolgálóoldali 40 elfogómodultól kapott időt.

Mindazonáltal akár a 166, 172 vagy 173 lépésekben készítünk ügyfélgyorsítótár bejegyzést, az ügyféloldali 30 elfogómodullal a 174 lépésben az ügyfélgyorsítótár bejegyzést http-adatfolyamként továbbítjuk vagy biztosítjuk a 10 webböngészőnek.

Amint a szakember számára nyilvánvaló, az ügyfélgyorsítótárak és a kiszolgáló-gyorsítótárak kivitelezhetők félvezető memóriával vagy nagy kapacitású tárolókkal, például merevlemezzel, olvasható/írható CD-ROM-mal, optikai lemezzel vagy egyéb tárolási technikák segítségével. Mi több, az ügyféloldali 30 elfogómodul és a kiszolgálóoldali 40 elfogómodul kivitelezhetők hardveresen, szoftveresen vagy a kettő kombinációjaként.

Bár a gyorsítótárakra eddig mint egy bizonyos első 5 számítógépben vagy második 6 számítógépben található egységekre hivatkoztunk, a szakember számára nyilvánvaló, hogy jelen találmány előnyei akkor is élvezhetők, ha a gyorsítótár nem az első 5 számítógépben van megvalósítva, hanem egyszerűen csak a külső 35 kommunikációs kapcsolatra ugyanazon az oldalon, mint az 5 számítógép. Így az ügyfélgyorsítótárként szolgáló hardvergyorsítótár létrehozható akár az első 5 számítógépben kívül is, nagy átviteli sebességű vonalakon át az első 5 számítógéphez kapcsolva. Ameddig az ügyfélgyorsítótár a külső 35 kommunikációs kapcsolat ugyanazon oldalán található, mint az

első 5 számítógép, a találmány előnyei ugyanúgy élvezhetők.

A találmány szerinti eljárás egy további előnyös fogantatási módjánál a kiszolgálóoldali 40 elfogómodullal nem tartunk fenn másolatot a 20 webkiszolgálótól fogadott http-adatfolyamról, hanem egyszerűen egy könyvtárbejegyzést tartunk fenn a kommunikáció számára, amely tartalmazza a kommunikáció URL-jét, a http-adatfolyamra kiszámított ciklikus redundanciakódot, azt az időpontot, amikor a http-adatfolyamot fogadtuk a 20 webkiszolgálótól, a kommunikáció tárolási időpontját, amely a ciklikus redundanciakód kiszámításának idejére állítható be. Ilyen esetben, ha az ügyféloldali 30 elfogómodullal a kiszolgálóoldali 40 elfogómodulnak egy kommunikációra vonatkozó kérelmet küldünk, amely egy olyan URL-nek felel meg, amelynek a kiszolgálóoldali 40 elfogómodul fenntartotta a ciklikus redundanciakódját és a tárolás időpontját, akkor a kiszolgálóoldali 40 elfogómodullal ellenőrizzük az ügyféloldali 30 elfogómodultól kapott ciklikus redundanciakódot, hogy megállapítsuk, az megfelel-e a bizonyos URL legutolsó http-adatfolyama ciklikus redundanciakódjának. Ha megfelel, akkor egy időben „összetartozó” választ küldünk az ügyféloldali 30 elfogómodulnak. Ha nem felel meg, akkor a kiszolgálóoldali 40 elfogómodullal elküldjük az ügyféloldali 30 elfogómodultól fogadott http-adatfolyamot a 20 webkiszolgálónak, és a 20 webkiszolgálótól fogadott választ küldjük vissza az ügyféloldali 30 elfogómodulnak.

A 7–10. ábrákon az ügyféloldali 30 elfogómodul és a kiszolgálóoldali 40 elfogómodul által végrehajtott műveleteket mutatjuk be találmányunk egy további lehetséges megvalósítása szerint, ahol különbözőzeti adatküldést használunk a külső 35 kommunikációs kapcsolaton keresztül továbbítandó adatok mennyiségének csökkentésére. A 200 lépésben az ügyféloldali 30 elfogómodullal http-kérelmet fogadunk a 10 webböngészőtől. A 205 lépésben az ügyféloldali 30 elfogómodullal megvizsgáljuk a http-kérelmet, hogy megállapítsuk, a kérelem egy közös átjárófelületre (CGI) vonatkozik-e. Ha a kérelem nem egy közös átjárófelületre vonatkozik, akkor a 206 lépésben az ügyféloldali 30 elfogómodullal a kiszolgálóoldali 40 elfogómodulnak továbbítjuk a kérelmet, ahogy a 3–6. ábrákon bemutattuk.

Amennyiben a 10 webböngészőtől kiinduló kommunikáció egy CGI-nek megfelelő kérelem, akkor a 205 lépésről az „igen” ágon haladunk tovább. A 210 lépésben az ügyféloldali 30 elfogómodullal megvizsgáljuk, létezik-e ügyfélbázis-gyorsítótár bejegyzés, mely egy olyan http-adatfolyamnak felel meg, amelyet egy megfelelő CGI-kérelemre válaszképpen továbbítottunk már korábban a 10 webböngészőnek. A CGI-kérelem vizsgálata az ügyfélbázis-gyorsítótár bejegyzésben tárolt URL-ek és a 10 webböngészőtől kiinduló kommunikáció URL-jének összehasonlításával történhet.

Az ügyfélbázis-gyorsítótár az első ügyféloldali 30 elfogómodullal fogadott, a 10 webböngésző által egy bizonyos URL-re továbbítandó http-adatfolyam eltárolásával inicializálható. Ez a bázisgyorsítótár fenntartható a 10 webböngésző többszöri, akár egyidejű

használata során. Az ügyfélbázis-gyorsítótár bejegyzések a 7–10. ábrákon példaképpen bemutatott módon frissíthetők. Ha a 10 webböngészőtől kiinduló kommunikáció URL-jének megfelelő ügyfélbázis-gyorsítótár bejegyzés már létezik, akkor a 211 lépésben a külső 35 kommunikációs kapcsolaton keresztül a kiszolgálóoldali 40 elfogómodulnak küldendő ciklikus redundanciakódot az ügyfélbázis-gyorsítótár bejegyzés ciklikus redundanciakódjával egyenlővé tesszük. Ha nem létezik ügyfélbázis-gyorsítótár bejegyzés, akkor a 210 lépésről a „nem” ágon haladunk tovább, és a 212 lépésben a külső 35 kommunikációs kapcsolaton keresztül a kiszolgálóoldali 40 elfogómodulnak küldendő kérelem ciklikus redundanciakódját lenullázzuk.

A 213 lépésben a CGI-kérelmet a külső 35 kommunikációs kapcsolaton keresztül a kiszolgálóoldali 40 elfogómodulhoz továbbítjuk. A 213 lépésben az ügyféloldali 30 elfogómodullal továbbítjuk a http-kérelmet és a kérelem ciklikus redundanciakódját, amelyet vagy már lenulláztunk, ha nem létezett a CGI-kérelem URL-jének ügyfélbázis-gyorsítótár bejegyzése, vagy ha létezett, akkor az ügyfélbázis-gyorsítótár bejegyzés ciklikus redundanciakódjának értékére állítottunk be. Így az ügyféloldali 30 elfogómodullal a CGI-kérelmet ügyfél-kiszolgáló specifikus protokollá alakítottuk át, és az ügyfél-kiszolgáló specifikus kommunikációt a külső 35 kommunikációs kapcsolaton keresztül a kiszolgálóoldali 40 elfogómodulhoz továbbítottuk.

A kiszolgálóoldali 40 elfogómodul működését egy CGI-kérelem vételekor a 9. ábrán mutatjuk be. A 220 lépésben a kiszolgálóoldali 40 elfogómodullal CGI-kérelmet fogadunk. Amikor a kiszolgálóoldali 40 elfogómodul megkapja a CGI-kérelmet, a http-kérelem és a ciklikus redundanciakód-érték egy-egy másolatát megtartjuk. A 221 lépésben a kiszolgálóoldali 40 elfogómodullal továbbítjuk a http-kérelmet a 20 webkiszolgálónak.

Amikor a kiszolgálóoldali 40 elfogómodullal a 10 webböngészőtől kiinduló kommunikációnak megfelelő http-kérelemre vagy a CGI-kérelemre választ kapunk, a 230 lépésben a kiszolgálóoldali 40 elfogómodullal http-adatfolyamként fogadjuk ezt a választ (lásd 10. ábra). A kiszolgálóoldali 40 elfogómodullal megőrizzük a http-adatfolyamot és kiszámítjuk a 20 webkiszolgálótól fogadott http-adatfolyam ciklikus redundanciakód-értékét. A kiszolgálóoldali 40 elfogómodullal a különbözeti értéket szintén lenullázzuk a különbözeti érték inicializálása céljából. A kiszolgálóoldali 40 elfogómodullal ezután a 235 lépésben megvizsgáljuk, hogy a 20 webkiszolgálótól kiinduló kommunikációként fogadott válasz egy CGI-kérelemre érkezett-e. Ha nem, akkor a 235 lépésről a „nem” ágon haladunk tovább, és a 236 lépésben a http-adatfolyamot elküldjük az ügyféloldali 30 elfogómodulnak. Ehhez a művelethez a 3–6. ábrákon bemutatott gyorsítótárakat használó technikákat is alkalmazhatjuk. Ha a 230 lépésben fogadott válasz egy CGI-kérelemre jött, akkor a 235 lépésről az „igen” ágon haladunk tovább, és a 240 lépésben a kiszolgálóoldali 40 elfogómodullal megállapítjuk, hogy a CGI-válasznak létezik-e kiszolgálóbázis-gyorsítótár bejegyzése.

A kiszolgálóbázis-gyorsítótár bejegyzés akkor is létrejöhet, amikor a kiszolgálóoldali 40 elfogómodullal először kapunk választ egy CGI-kérelemre. Ebben az esetben a 240 lépésről a „nem” ágon haladunk tovább. A 241 lépésben a kiszolgálóoldali 40 elfogómodullal a CGI URL-jének, a CGI-kérelemre válaszoló http-adatfolyamnak és a http-adatfolyam ciklikus redundanciakódjának eltárolásával elkészítjük a CGI-kérelemnek megfelelő kiszolgálóbázis-gyorsítótár bejegyzést. Hogy a 3–6. ábrákon bemutatott összetartozó gyorsítótárrendszerrel kompatibilis legyen, a kiszolgálóbázis-gyorsítótár bejegyzés szintén tartalmazhatja a tároláspontot. A továbbiakban „kiszolgáló-CGI-bázislapként” említett kifejezés a 10 webböngészőtől kapott CGI-kérelemnek megfelelő kiszolgálóbázis-gyorsítótár bejegyzésre utal.

Ha létezik a CGI-kérelemnek megfelelő kiszolgálóbázis-gyorsítótár bejegyzés, akkor a 240 lépésről az „igen” ágon haladunk tovább. A kiszolgálóoldali 40 elfogómodullal a 245 lépésben összehasonlítjuk a kiszolgálóbázis-gyorsítótár bejegyzés ciklikus redundanciakódját a 20 webkiszolgálótól kapott válasz ciklikus redundanciakódjával. Ha a ciklikus redundanciakódok megegyeznek, akkor a kiszolgálóoldali 40 elfogómodullal megvizsgáljuk, hogy a kiszolgálóbázis-gyorsítótár bejegyzés ciklikus redundanciakódja megegyezik-e az ügyfélbázis-gyorsítótár bejegyzés ciklikus redundanciakódjával. Ha ez a két ciklikus redundanciakód-érték azonos, akkor a kiszolgálóbázis-gyorsítótár bejegyzés, az ügyfélbázis-gyorsítótár bejegyzés és a 20 webkiszolgálótól kapott válasz mind ugyanazt a http-adatfolyamot tartalmazza. A kiszolgálóbázis-gyorsítótár bejegyzést a 250 lépésben hasonlítjuk össze az ügyfélbázis-gyorsítótár bejegyzéssel.

Ha a két bázisgyorsítótár bejegyzés azonos, akkor nem kell a kiszolgálóoldali 40 elfogómodullal a bázisgyorsítótár bejegyzést elküldenünk az ügyféloldali 30 elfogómodulnak, és így a 251 lépésben az ügyféloldali 30 elfogómodulnak küldendő http-adatfolyam adatait is lenullázzuk. A kiszolgálóoldali 40 elfogómodullal ezután a 252 lépésben a CGI-kérelemnek megfelelő kiszolgáló-bázisgyorsítótárban tárolt http-adatfolyam ciklikus redundanciakódjának, a lenullázott http-adatfolyam adatainak és a lenullázott különbözeti adat elküldésével átalakítjuk a 20 webkiszolgálótól fogadott http-adatfolyamot ügyfél-kiszolgáló specifikus kommunikációs protokollá, hogy jelezzük: a CGI-kérelemre érkezett válasz megegyezett az ügyfélbázis-gyorsítótár bejegyzéssel.

Visszatérve a 245 lépésre, ha a CGI-kérelemnek megfelelő kiszolgálóbázis-gyorsítótár bejegyzés ciklikus redundanciakódja különbözik a 10 webböngészőtől kiinduló CGI-kérelemre válaszoló, a webkiszolgálótól kapott válasz ciklikus redundanciakódjától, akkor a 245 lépésről a „nem” ágon haladunk tovább. A kiszolgálóoldali 40 elfogómodullal ezután a 246 lépésben összehasonlítjuk az elfogott CGI-választ és az elfogott CGI-kérelemnek megfelelő kiszolgálóbázis-gyorsítótár bejegyzést, avagy a kiszolgáló-CGI-bázislapot. Az elfogott CGI-válasz és a kiszolgáló-CGI-bázislap összeha-

sonlításával CGI különbözeti adatot kapunk, amely megfelel az elfogott CGI-válasz és a kiszolgáló CGI bázislap különbségének.

A megkülönböztetés szakma által ismert bármilyen módszerrel elvégezhető, amely meghatározza a bázislap és a módosított lap különbségét. A találmányunk esetében jól használható egyik megkülönböztetési módszer leírása megtalálható például Coppieters: „Egy platformok közötti bináris megkülönböztetés” című cikkében („A Cross-Platform Binary Diff”, Dr. Dobb's Journal, 1995. május, 32–36. oldalak), amely dokumentumot jelen leírásunkban referenciaként tekintjük. Egyéb módszerek közül, amelyek a különbözeti adatot meghatározhatják, a DBM Technical Disclosure Bulletin, Vol. 22, No. 8A, 1980. januári számában leírt módszert említjük, amely szintén referenciaként tekintendő.

A kiszolgálóoldali 40 elfogómodullal ezután a 247 lépésben megvizsgáljuk, hogy a kiszolgáló-CGI-bázislap igényel-e aktualizálást. Ezt az dönti el, hogy a kapott CGI-válasz és a kiszolgáló-CGI-bázislap közötti átlag-különbözetiadat nagyobb-e az előre meghatározott küszöbnél. A CGI-kérelemnek megfelelő kiszolgálóbázis-gyorsítótár bejegyzés aktualizálásának szükségességét meghatározó további lehetséges eljárás lehet a 3–6. ábrákon leírt idő-összetartozási módszer vagy más, szakember által ismert módszer, amellyel megállapítható, hogy a különbözeti adat annyira nagyobb lett-e, hogy a bázislap újraalapozásával egy új bázisgyorsítótár bejegyzés készítése javítaná-e a rendszer teljesítményét.

Ha a kiszolgáló-bázislap újraalapozására nincs szükség, akkor a 247 lépésről a „nem” ágon haladunk tovább, és a kiszolgálóoldali 40 elfogómodullal a 250 lépésben meghatározzuk, hogy az ügyfélbázis-gyorsítótár bejegyzés ciklikus redundanciakódja azonos-e a kiszolgálóbázis-gyorsítótár bejegyzés ciklikus redundanciakódjával, avagy a kiszolgáló-bázislap megegyezik-e az ügyfélbázislappal. Ezek a lapok a kiszolgáló- és az ügyfélbázis-gyorsítótár bejegyzései, amelyek a 10 webböngészőtől kiinduló kommunikáció egy bizonyos CGI-kérelemnek felelnek meg. Ha a bázislapok megegyeznek, akkor az ügyfél bázislapját nem kell újraalapozni, és a http-adatfolyamot a 251 lépésben lenullázzuk. Ezután a kiszolgálóoldali 40 elfogómodullal a 252 lépésben elküldjük a különbözeti választ az ügyféloldali 30 elfogómodulnak. Mindez a CGI-kérelemnek megfelelő kiszolgálóbázis-gyorsítótár bejegyzés ciklikus redundanciakódjának (a kiszolgáló-CGI-bázislap CRC-értékének) elküldésével, a bázisadatnak megfelelő lenullázott http-adatfolyam elküldésével és a 246 lépésben meghatározott különbözeti adat elküldésével történik.

Ha a kiszolgálóoldali 40 elfogómodullal megállapítjuk, hogy a kiszolgáló-CGI-bázislap ciklikus redundanciakódja és az ügyfél-CGI-bázislap ciklikus redundanciakódja nem azonos, akkor az ügyfél bázislapját újra kell alapoznunk. Az ügyfél bázislapjának újraalapozása során a kiszolgáló-CGI-bázislapot elküldjük az ügyféloldali 30 elfogómodulnak. A művelet elvégzése érdekében a kiszolgálóoldali 40 elfogómodullal a 253 lépés-

ben az ügyféloldali 30 elfogómodulnak küldendő http-adatfolyam adatait a kiszolgáló-CGI-bázislappal egyenlővé tesszük. Ezután a 252 lépésben a kiszolgálóoldali 40 elfogómodullal a 20 webkiszolgálótól fogadott http-adatfolyamot egy ügyfél-kiszolgáló specifikus protokollá alakítjuk át, a kiszolgáló-CGI-bázislap ciklikus redundanciakódjának elküldésével, a kiszolgáló-CGI-bázislapnak megfelelő http-adatfolyam adatainak elküldésével és a 20 webkiszolgálótól fogadott válasz és a kiszolgáló-CGI-bázislap közötti különbség elküldésével. Ezt az információt a külső 35 kommunikációs kapcsolaton keresztül az ügyféloldali 30 elfogómodulhoz továbbítjuk.

Amennyiben a kiszolgáló bázislapjának újraalapozása szükséges, akkor a 247 lépésről az „igen” ágon haladunk tovább. A 248 lépésben a kiszolgálóoldali 40 elfogómodullal aktualizáljuk a 10 webböngészőtől kiinduló kommunikációnak megfelelő kiszolgálóbázis-gyorsítótár bejegyzést a 20 webkiszolgálótól kapott http-adatfolyammal. A válasz ciklikus redundanciakódja szintén aktualizálódik, és a CGI-különbség lenullázódik. A kiszolgálóoldali 40 elfogómodullal ezután a 250 lépésben összehasonlítjuk az új kiszolgálóoldali gyorsítótár bejegyzést, és végrehajtjuk a fent leírt továbbítást.

Az ügyféloldali 30 elfogómodullal végzett műveleteket a kiszolgálóoldali 40 elfogómodul válaszában fogadásakor a 8. ábrán mutatjuk be. A kiszolgálóoldali 40 elfogómodultól kiinduló választ a 260 lépésben fogadjuk az ügyféloldali 30 elfogómodullal. A 265 lépésben az ügyféloldali 30 elfogómodullal megállapítjuk, hogy a válasz egy CGI-kérelemre jött-e. Ha a válasz nem egy CGI-kérelemre jött, akkor az ügyféloldali 30 elfogómodullal a 267 lépés műveleteit hajtjuk végre, amelyek magukban foglalhatják a 3–6. ábrákon bemutatott gyorsítótár-műveleteket is. Ha azonban a válasz egy CGI-kérelemre érkezett, akkor a 265 lépésről az „igen” ágon haladunk tovább. A 266 lépésben az ügyféloldali 30 elfogómodullal megtartjuk a http-adatfolyam adatait, a különbözeti adatot és a külső 35 kommunikációs kapcsolaton keresztül továbbított ügyfél-kiszolgáló specifikus adatfolyamból szerzett ciklikus redundanciakódot.

Ezután a 270 lépésben az ügyféloldali 30 elfogómodullal megvizsgáljuk, hogy létezik-e egy, a vett CGI-kérelemnek megfelelő, egy ügyfél-CGI-bázislapot tartalmazó ügyfélbázis-gyorsítótár bejegyzés. Ezt a lekérdezést a http-kérelem vagy válasz-URL-jének vizsgálatával vizsgáljuk. Ha létezik egy ügyfélbázislap, akkor a 270 lépésről az „igen” ágon haladunk tovább. Ezután a 275 lépésben az ügyféloldali 30 elfogómodullal a külső 35 kommunikációs kapcsolaton keresztül vett ciklikus redundanciakódot összehasonlítjuk az ügyfél-CGI-bázislapjának ciklikus redundanciakódjával. Ha ezek különböznek, akkor a 275 lépésről a „nem” ágon haladunk tovább, és az ügyfél bázislapját a CGI-bázislap frissítésével újraalapozzuk a 10 webböngészőtől kiinduló kommunikáció CGI-kérelem-URL-jének megfelelő ügyfélbázis-gyorsítótár bejegyzésnek a kiszolgálóoldali 40 elfogómodultól a külső 35 kommunikációs kapcsola-

ton keresztül vett http-adatfolyamra való kicserélésével. Az ügyfélbázis-gyorsítótár bejegyzését a 276 lépésben a http-adatfolyam ciklikus redundanciakódra nézve is felfrissítjük.

Ha a külső 35 kommunikációs kapcsolaton keresztül vett ciklikus redundanciakód ugyanaz, mint a CGI-bázislap ciklikus redundanciakódja, akkor a kiszolgálóoldali 40 elfogómodul kiszolgáló-CGI-bázislapja megegyezik az ügyféloldali 30 elfogómodul ügyfélbázislapjával, és a 275 lépésről az „igen” ágon haladunk tovább.

Attól függetlenül, hogy a bázislapok megegyeznek-e, vagy az ügyfél bázislapját újraalapozzuk, az ügyféloldali 30 elfogómodullal a 277 lépésben rekonstruáljuk a 20 webkiszolgálótól érkezett kommunikációnak megfelelő http-adatfolyamot a külső 35 kommunikációs kapcsolaton keresztül vett ügyfél-kiszolgáló specifikus adatokból az ügyfél-CGI-bázislapjának és a külső 35 kommunikációs kapcsolaton keresztül vett CGI különbözőti adatok kombinálásával, hogy egy, a CGI-válasznak megfelelő http-adatfolyamot hozzunk létre. Ezt a választ a 10 webböngésző a 278 lépésben http-adatfolyamként kapja meg.

Ha az ügyfélnek nincs a CGI-kérelem URL-jéhez megfelelő CGI-bázislapja, akkor a 270 lépésről a „nem” ágon haladunk tovább. A 271 lépésben az ügyféloldali 30 elfogómodullal létrehozunk egy, a CGI-kérelem URL-jének megfelelő ügyfélbázis-gyorsítótár bejegyzést az URL tárolásával, valamint a kiszolgálóoldali 40 elfogómodultól a külső 35 kommunikációs kapcsolaton keresztül vett http-adatfolyam ciklikus redundanciakódjának és magának a http-adatfolyamnak az eltárolásával. Ezen információ eltárolásával létrehozunk egy gyorsítótár bejegyzést a vett CGI-kérelemnek megfelelően, ezáltal létrehozva a CGI-bázislapot. Az ügyféloldali 30 elfogómodullal ezután a 277 lépésben az ügyfél-CGI-bázislap és az esetleg hatálytalanná vált CGI különbözőti adatok egyesítésével vagy összevonásával rekonstruáljuk a http-adatfolyamot.

A jelenlegi különbözőti módszerek a nem-CGI adatokra is alkalmazhatók. Egy ilyen helyzetben a kiszolgálóoldali 40 elfogómodulban a kiszolgálóbázis-gyorsítótár bejegyzések több generációját kellene tárolnunk, hogy felkészüljünk arra az esetre, ha a 10 webböngészőknek a 20 webkiszolgálóhoz kapcsolt ügyféloldali 30 elfogómoduljai különböző bázislapokat tartalmaznak. A kiszolgálóoldali 40 elfogómodullal ekkor összehasonlíthatnánk az ügyféloldali 30 elfogómodultól kapott ciklikus redundanciakódot a kiszolgáló-bázislapok korábbi generációinak ciklikus redundanciakódjaival, amíg egyezést nem találunk. Ezután a kiszolgálóoldali 40 elfogómodullal vagy újraalapozzuk az ügyféloldali 30 elfogómodul bázislapját, vagy egyszerűen elküldjük a különbözőti adatot az ügyféloldali 30 elfogómodulnak. Így az itt leírt, CGI-kérelmekkel kapcsolatos megkülönböztetési módszereket ugyanúgy alkalmazhatjuk http-kérelmek és válaszok esetére is.

Jóllehet a fenti, a bázislapok több generációját fenntartó rendszer lehetővé teszi a különbözőti adatküldést nem-CGI kérelmek esetén is, de nagyobb memóriát és

tárolókapacitást igényel, és nem használja ki teljesen a fent leírt gyorsítási képességeket. A memória- vagy társzükséglet csökkentése és a fent leírt gyorsítási módszerek kihasználása érdekében a nem-CGI kérelmek körében a különbözőtiadat-küldés megvalósítására ezért az alábbi előnyös eljárást alkalmazhatjuk.

A kiszolgálóoldali 40 elfogómodullal kiszámítjuk a különbséget a kérelemnek megfelelő kiszolgáló-bázislap és a 20 webkiszolgálótól válaszként kapott http-adatfolyam között, és ezt a különbözőti adatfolyamot a kiszolgálóoldali 40 elfogómodulban tároljuk. A kiszolgáló-bázislapot ezután a bázislapnak a 20 webkiszolgálótól kapott új válaszra történő kicserélésével frissítjük, beleértve a bázislap ciklikus redundanciakódjának frissítését is. Mindazonáltal a régi ciklikus redundanciakód elhagyása helyett az előző bázislapok ciklikus redundanciakódjait a különbözőti adattal együtt eltároljuk. A különbözőti adatok és ciklikus redundanciakódok megelőző generációit válogatva továbbítjuk az ügyféloldali 30 elfogómodulhoz a nem-CGI kérelemnek megfelelő ügyfélbázislap ciklikus redundanciakódja alapján.

A nem-CGI különbözőti módszerre azt a példát hozhatjuk fel, hogy ha a kiszolgálóoldali 40 elfogómodullal egy nem-CGI kérelmet veszünk, akkor ezt a kérelmet a nem-CGI kérelem URL-jének megfelelő ügyféloldali 30 elfogómodulban található bázislap ciklikus redundanciakódja kíséri. Miután a kiszolgálóoldali 40 elfogómodullal megkaptuk a választ a 20 webkiszolgálótól, akkor kiszámítjuk a válasz ciklikus redundanciakódját. A kiszolgálóoldali 40 elfogómodullal ezután kiszámítjuk a különbséget a válasz és az URL-hez tartozó kiszolgáló-bázislap között, és ezt a különbözőti adatot elmentjük. A kiszolgálóoldali 40 elfogómodullal a kiszolgáló-bázislapot az új válaszadataival frissítjük, és az előző bázislap ciklikus redundanciakódját, valamint a válasz és a régi bázislap közötti különbözőti adatot eltároljuk. A kiszolgálóoldali 40 elfogómodullal ezután összehasonlítjuk az ügyfélbázislap ciklikus redundanciakódját a kiszolgáló-bázislap ciklikus redundanciakódjával és bármilyen más tárolt vagy archivált ciklikus redundanciakóddal, hogy eldöntsük, van-e egyezés. Ha nem találunk egyezést, a választ egyszerűen elküldjük az ügyféloldali 30 elfogómodulnak.

Ha egyezést találunk, a ciklikus redundanciakódnak megfelelő különbözőti adatot és minden későbbi különbözőti adatot a jelenlegi különbözőti adatig bezárólag elküldjük az ügyféloldali 30 elfogómodulnak, amellyel ezután ráillesztjük a különbözőti adatot az ügyfélbázislapra, hogy rekonstruáljuk a választ. Így, ha a ciklikus redundanciakód egy három generációval régebbi bázislap ciklikus redundanciakódjával egyezett, akkor három különbözőti adathalmazt küldünk el az ügyféloldali 30 elfogómodulhoz, és a válasz összeállítása három egymást követő különbözőti adathalmaznak az ügyfélbázislapra való illesztésével valósul meg. Ha a válasz rekonstruálásához szükséges különbözőti adathalmazok száma vagy mérete olyan nagy, hogy a tulajdonképpen válasz küldése kevesebb adat átvitelét igényelné, akkor a kiszolgálóoldali 40 elfogómodullal magát a választ küldjük el. Bármelyik esetben a válasz re-

konstruálása vagy vétele után az ügyféloldali 30 elfogómodullal frissítjük a kérelem URL-jének ügyfélbázislapját a válaszadatokkal, és frissítjük a ciklikus redundanciakódot a válasz ciklikus redundanciakódjával. Mivel az ügyfélbázislap minden egyes alkalommal frissítésre kerül, ha egy meghatározott URL-ről válasz érkezik, a fent leírt ügyfélgyorsítótár az ügyfélbázislap gyorsítótáraként használható, kiiktatva ezzel egy külön gyorsítótár szükségességét az ügyfélbázislapokhoz, ha nem-CGI kérelmek esetén a különbozati módszert használjuk.

Találmányunk egy további lehetséges megvalósítása szerint még nagyobb megtakarítást érhetünk el az átvitelnél egy olyan állapot nélküli kommunikációs protokoll redundanciája alapján, mint a http. Egy ilyen protokollban az ügyfél minden alkalommal, amikor kommunikációt kezdeményez, információt továbbít magáról a kiszolgálónak. Ugyanígy a kiszolgáló minden alkalommal, amikor egy választ elindít, bizonyos információt közöl magáról az ügyféllel.

Találmányunk egy további lehetséges kiviteli alakjánál az első 5 számítógép közli a második 6 számítógéppel az első 5 számítógép előre meghatározott jellemzőinek megfelelő számítógép-specifikus információt. A második 6 számítógép tárolja ezt a számítógép-specifikus információt. Az első 5 számítógép ezután elhagyja a számítógép-specifikus információt az ezt követő, a 10 webböngészőtől kiinduló kommunikációkból a külső 35 kommunikációs kapcsolaton való továbbítást megelőzően. A második 6 számítógép ezután felépíti az eredetileg a 10 webböngészőtől kiinduló kommunikációkat a tárolt számítógép-specifikus információk és az ezt követő, a külső 35 kommunikációs kapcsolaton keresztül vett kommunikációk összevonásával, hogy létrehozzon egy http-adatfolyamot.

A számítógép-specifikus információnak a 10 webböngészőtől kiinduló kommunikációkból való elhagyásán túl ezt a számítógép-specifikus információt a 20 webkiszolgálótól kiinduló kommunikációkból is el lehet hagyni. Egy ilyen esetben például a 2. ábrán látható konfigurációban a második 6 számítógép a külső 35 kommunikációs kapcsolaton keresztül juttatja el az első 5 számítógépnek a második 6 számítógép előre meghatározott jellemzőinek megfelelő számítógép-specifikus információt. Az első 5 számítógép eltávolítja a számítógép-specifikus információt, hogy az a kiszolgáló-fejlécinformációt szolgáltatassa. Ezt követő kommunikációknál a második 6 számítógép elhagyja a számítógép-specifikus információt a 20 webkiszolgálótól kiinduló kommunikációkból, és a külső 35 kommunikációs kapcsolaton keresztül továbbítja a 20 webkiszolgálótól kiinduló kommunikáció megmaradt részét. Az első 5 számítógép veszi a kommunikációt a külső 35 kommunikációs kapcsolaton keresztül, és felépíti az eredeti, a 20 webkiszolgálótól kiinduló kommunikációt a kiszolgáló-fejlécinformáció és a külső 35 kommunikációs kapcsolaton keresztül vett ügyfél-kiszolgáló specifikus adatfolyam összevonásával, hogy létrehozzon egy http-adatfolyamot. Mindkét esetben a számítógép-specifikus információ elhagyásának és az informá-

ció tárolásának – vagy kiszolgáló-fejlécinformáció, vagy ügyfél-fejlécinformáció létrehozása céljából – a műveletét az ügyféloldali 30 elfogómodullal vagy a kiszolgálóoldali 40 elfogómodullal hajtuk végre attól függően, hogy a művelet az első 5 számítógépben vagy a második 6 számítógépben történik.

Találmányunk egy további előnyös megvalósításában a 10 webböngésző az ügyféloldali 30 elfogómodullal TCP/IP protokollon keresztül kommunikál. A TCP protokollt az ügyféloldali 30 elfogómodul és a kiszolgálóoldali 40 elfogómodul közötti kommunikációra is használhatjuk a külső 35 kommunikációs kapcsolaton keresztül. Végül a TCP protokollt a kiszolgálóoldali 40 elfogómodul és a 20 webkiszolgáló közötti kommunikációra is használhatjuk. Bár a találmány szerinti berendezést képező különféle elemek közötti kommunikációra a TCP protokoll használható, a http protokoll nem a leghatékonyabb kommunikációs eszköz egy külső 35 kommunikációs kapcsolaton keresztül. A külső 35 kommunikációs kapcsolat teljesítményének növelésére a találmány szerinti berendezésben létrehozunk azokat a „virtuális szoftvercsatornákat”, amelyek a 10 webböngésző és az ügyféloldali 30 elfogómodul, valamint a kiszolgálóoldali 40 elfogómodul és a 20 webkiszolgáló közötti kommunikációra használhatók. A továbbiakban ezeknek a virtuális szoftvercsatornáknak a kialakítását és működését részletezzük a 11–17. ábrák segítségével.

A 11. ábrán a találmány szerinti berendezés virtuális szoftvercsatornák koncepcióját felhasználó lehetséges kiviteli alakját tüntettük fel tömbvázlatként. Az első 5 számítógép és a második 6 számítógép egy külső 35 kommunikációs kapcsolattal csatlakozik egymáshoz. A 10 webböngésző több valódi szoftvercsatornával rendelkezik, amelyekkel a 10 webböngésző az ügyféloldali 30 elfogómodulhoz csatlakozik. A 11. ábrán láthatóan a 10 webböngésző első valódi 65A szoftvercsatornája az ügyféloldali 30 elfogómodul megfelelő valódi 65B szoftvercsatornájával áll kapcsolatban. Ez az első valódi 65A szoftvercsatorna az a TCP protokollt használó szoftvercsatorna, amelyen keresztül a 10 webböngésző további csatlakozásokat kérelmez az ügyféloldali 30 elfogómodultól.

Amikor a 10 webböngésző egy új TCP-kapcsolatot kér, erre vonatkozó kommunikáció zajlik a valódi 65A szoftvercsatornán keresztül, amelyet a valódi 65B szoftvercsatorna fogad. Az ügyféloldali 30 elfogómodul ezután létrehoz egy további valódi szoftvercsatornát a 10 webböngészővel való kommunikációhoz. Amint a 11. ábrán látható, így több valódi 60A–65A szoftvercsatorna jön létre a 10 webböngészőnél, és mindegyiknek megfelelően egy-egy valódi 60B–65B szoftvercsatorna jön létre az ügyféloldali 30 elfogómodulnál. A valódi 60A–65A szoftvercsatornák a 10 webböngészőhöz, a 60B–65B szoftvercsatornák az ügyféloldali 30 elfogómodulhoz tartoznak. Ezek a valódi 60A–65A, 60B–65B szoftvercsatornák azok az eszközök, amelyeken keresztül a 10 webböngésző és az ügyféloldali 30 elfogómodul egymással kommunikál. A valódi 60A–65A szoftvercsatornák létrehozása után az eze-

ken keresztüli kommunikációk multiplexelve kerülnek egy további valódi 36A szoftvercsatornára, amely elérést biztosít az ügyféloldali 30 elfogómodulnak a külső 35 kommunikációs kapcsolathoz. A 36A és 36B szoftvercsatornák akkor jönnek létre, amikor az első 5 számítógép valódi 37A szoftvercsatornáján keresztül egy csatlakozási kérelmet küld a második 6 számítógép valódi 37B szoftvercsatornájához. Amint a valódi 37B szoftvercsatorna fogadja a csatlakozási kérelmet, létrejönnek a valódi 36A és 36B szoftvercsatornák. A 37A és 37B szoftvercsatornák az ügyféloldali 30 elfogómodul és a kiszolgálóoldali 40 elfogómodul közötti kommunikáció első valódi csatornáiként viselkednek, és csak a 36A és 36B szoftvercsatornák által képviselt két modul összekapcsolására szabad azokat felhasználni. Ezen valódi 36A, 36B, 37A, 37B szoftvercsatornák mindegyike szabványos TCP/IP protokollt használ. Amikor a második 6 számítógép kommunikációt fogad a külső 35 kommunikációs kapcsolaton keresztül, a valódi 36B szoftvercsatornán keresztül veszi azt. A kiszolgálóoldali 40 elfogómodul ezután szétválasztja a 36B szoftvercsatornán vett kommunikációkat, és átadja azokat a megfelelő szoftvercsatornáknak a 20 webkiszolgálóhoz való továbbítás céljából. Tehát például a 60A szoftvercsatornától a 60B szoftvercsatornához érkezett kommunikáció, például egy bizonyos URL-ről származó információ iránti kérelem a 36A szoftvercsatornára multiplexelt, a 36B szoftvercsatorna fogadja, a kiszolgálóoldali 40 elfogómodul szétválasztja, majd a 60C és a 60D szoftvercsatornákon keresztül továbbítja a 20 webkiszolgálóhoz. Ugyanígy a 61A szoftvercsatornán keresztüli kommunikációt a 61B szoftvercsatorna fogadja, az ügyféloldali 30 elfogómodul multiplexeli és a 36A szoftvercsatornától a 36B szoftvercsatornához továbbítja, ahol a kiszolgálóoldali 40 elfogómodul szétválasztja a kommunikációt, és továbbítja a 61C szoftvercsatornán keresztül a 61D szoftvercsatornához. Tehát a 60A és 60B, a 61A és 61B, a 62A és 62B, a 63A és 63B, valamint a 64A és 64B szoftvercsatornákon keresztüli kommunikációk a hozzájuk tartozó szoftvercsatornákon kerülnek továbbításra a kiszolgálóoldali 40 elfogómodul és a 20 webkiszolgáló között a 60C és 60D, a 61C és 61D, a 62C és 62D, a 63C és 63D, valamint a 64C és 64D szoftvercsatornákon.

A 20 webkiszolgálónak a 10 webböngésző kérelmeire adott válaszai hasonló módon a 20 webkiszolgálót a kiszolgálóoldali 40 elfogómodullal összekötő 60C–64C, 60D–64D szoftvercsatornákon keresztül kerülnek továbbításra, majd a külső 35 kommunikációs kapcsolaton át az ügyféloldali 30 elfogómodulhoz és végül a 10 webböngészőkhöz. Tehát például egy, a 20 webkiszolgálótól kiinduló választ a 60D szoftvercsatornán keresztül a 60C szoftvercsatornához lehet küldeni, a kiszolgálóoldali 40 elfogómodul a 36B szoftvercsatornára multiplexeli, ahonnan a külső 35 kommunikációs kapcsolaton keresztül a 36A szoftvercsatornához lehet továbbítani. Az ügyféloldali 30 elfogómodul aztán szétválasztja a kapott kommunikációt, és átadja a 60B szoftvercsatornának a 10 webböngészőn lévő 60A szoftvercsatornához való továbbítás céljából. Ha-

sonló kommunikációs útvonal jön létre a 10 webböngésző és a 20 webkiszolgáló által használt minden szoftvercsatorna esetében. Igaz ugyan, hogy a bemutatott kiviteli alak esetében a 10 webböngésző és a 20 webkiszolgáló között négy szoftvercsatornát tüntettünk fel, de akármennyi szoftvercsatorna megnyitható a 10 webböngésző és a 20 webkiszolgáló közötti kommunikáció biztosítására, amint az szakember számára nyilvánvaló.

Találmányunk egy további előnyös megvalósítását a 12. ábra tömbvázlata és a 13–17. ábrákon látható folyamatábrák segítségével mutatjuk be. A 12. ábra tömbvázlatán a virtuális szoftvercsatorna-rendszert ábrázoltuk az ügyféloldali 30 elfogómodulban és a kiszolgálóoldali 40 elfogómodulban. Az e modulokon kívüli valódi szoftvercsatornák az ügyféloldali 30 elfogómodul és a 10 webböngésző között, valamint a kiszolgálóoldali 40 elfogómodul és a 20 webkiszolgáló között egyszerű TCP/IP szoftvercsatornákként működnek.

A 13. ábrán a 12. ábrán bejelölt 68 valódiszoftvercsatorna-menedzser egy lehetséges működésének folyamatábráját tüntettük fel. A 300 lépésben létrehozuk az ügyféloldali 30 elfogómodul 68 valódiszoftvercsatorna-menedzserét. A 68 valódiszoftvercsatorna-menedzser létrejötte után a 301 lépésben létrehozunk egy első valódi 65B szoftvercsatornát (lásd a 12. ábrán). Az első valódi 65B szoftvercsatorna létrehozása után a 302 lépésben az ügyféloldali 30 elfogómodulban lévő 68 valódiszoftvercsatorna-menedzserrel, amelyet ügyfélszoftvercsatorna-menedzsernek is nevezhetünk, egy, az első valódi 65B szoftvercsatornán bekövetkező eseményre várunk. Amikor egy az első valódi 65B szoftvercsatornán keresztül érkező eseményt észlelünk, a 305 lépésben a 68 valódiszoftvercsatorna-menedzserrel megvizsgáljuk az eseményt, és a vizsgálat alapján öt lehetséges továbblépési ág közül kiválasztunk egyet.

Ha az első valódi 65B szoftvercsatornán fogadott kommunikációs kérelemre válaszként egy valódi szoftvercsatorna jön létre, akkor a 68 valódiszoftvercsatorna-menedzserrel a valódi szoftvercsatornát felvesszük egy valódi esemény-listára. A 68 valódiszoftvercsatorna-menedzserrel ezután a 307 lépésben létrehozunk egy szimplex virtuális szoftvercsatornát. Az ügyféloldali 30 elfogómodul esetében a 68 valódiszoftvercsatorna-menedzserrel a 308 lépésben elindítunk egy alkalmazást, amely elvégzi az ügyféloldali 30 elfogómodul feladatát a virtuális szoftvercsatorna vonatkozásában.

A leírásunkban használt „szimplex valódi szoftvercsatorna” vagy „szimplex virtuális szoftvercsatorna” kifejezések olyan szoftvercsatornára vonatkoznak, amely vagy egyetlen szoftvercsatornához, vagy egyetlen alkalmazáshoz csatlakozik közvetlenül. A leírásunkban használt „multiplex szoftvercsatorna” pedig olyan szoftvercsatornára vonatkozik, amely más szoftvercsatornák sokaságához csatlakozik. Tehát a multiplex szoftvercsatorna multiplexelő- vagy demultiplexelő-feladatot hajt végre, a szimplex szoftvercsatorna pedig közvetlen csatlakozást valósít meg. Így például a 13. ábra 306–308 lépéseinek végrehajtásához a 68 va-

lódisszoftvercsatorna-menedzserrel az első valódi 65B szoftvercsatorna által vett első csatlakozási kérelemre válaszképpen létrehozuk a valódi 60B szoftvercsatornát, a szimplex virtuális 70 szoftvercsatornát, és ügyféloldali elfogási műveletet kezdeményezünk egy 80 elfogási műveletben. Hasonlóan az ezt követő eseményeknél, amikor egy valódi szoftvercsatorna jön létre, a 68 valódiszoftvercsatorna-menedzserrel létrehozuk a valódi 61B, 62B, 63B vagy 64B szoftvercsatornákat és a szimplex virtuális 71, 72, 73 vagy 74 szoftvercsatornákat, valamint a valódi és virtuális szoftvercsatornáknak megfelelően ügyféloldali 81, 82, 83 és 84 elfogási műveletet kezdeményezünk.

A továbbiakban a 12. ábra segítségével az ügyféloldali 81, 82, 83 és 84 elfogási műveletet mutatjuk be a valódi 60B szoftvercsatorna, a szimplex virtuális 70 szoftvercsatorna és az ügyféloldali 80 elfogási művelet vonatkozásában. A 14. ábra 325 lépésében létrehozunk egy ügyféloldali 80 elfogási műveletet. Az ügyféloldali 80 elfogási művelet létrejöttékor a 326 lépésben egy, a szimplex virtuális 70 szoftvercsatormán bekövetkező eseményre várunk, a 16–4. ábrán bemutatásra kerülő virtuális kiválasztási művelet végrehajtásával. Egy esemény észlelésekor a 330 lépésben az eseményt megvizsgáljuk. Ha az esemény egy virtuális 70 szoftvercsatorna-lezárás, akkor az ügyféloldali 80 elfogási művelettel a 349 lépésben töröljük a 70 szimplex virtuális szoftvercsatornát, amely így a 350 lépésben megszűnik.

Ha az esemény valamilyen adat vétele, akkor a 331 lépésben az ügyféloldali 80 elfogási művelettel fogadjuk a 10 webböngészőtől kiinduló kommunikációt a szimplex virtuális 70 szoftvercsatormán keresztül, a 16–3. ábrán bemutatásra kerülő virtuális fogadás művelet végrehajtásával. Az ügyféloldali 80 elfogási művelettel a fentiek szerint 332 lépésben végrehajtjuk az ügyféloldali 30 elfogómodul feladatát (lásd például a 3. és 7. ábrát). Az ügyféloldali 80 elfogási művelettel ezután a 333 lépésben a 16–1. ábrán bemutatásra kerülő virtuális létrehozás művelet végrehajtásával létrehozunk egy multiplex virtuális 90 szoftvercsatornát, amely a valódi 36A szoftvercsatornához csatlakozik az ügyféloldali 30 elfogómodulban. A valódi 36A szoftvercsatorna a kiszolgálóoldali 40 elfogómodul valódi 36B szoftvercsatornájához kapcsolódik. A 334 lépésben elküldjük a 10 webböngészőtől vett információt a 60B szoftvercsatormán és a szimplex virtuális 70 szoftvercsatormán keresztül, miután az ügyféloldali 80 elfogási műveletet befejeztük a 10 webböngészőtől kiinduló kommunikációra nézve. Ezt a kommunikációt a multiplex virtuális 90 szoftvercsatornán várakozási sorba állítjuk a 16–2. ábrán bemutatásra kerülő virtuális küldés művelet végrehajtásával. Az ügyféloldali 80 elfogási művelettel a 335 lépésben a kérelemnek a multiplex virtuális 90 szoftvercsatornához várakozási sorba állítása után ürítjük a várakozási sorba állított adatokat a multiplex virtuális 90 szoftvercsatornán, és ezután a 336 lépésben egy, a multiplex virtuális 90 szoftvercsatormán érkező eseményre várunk. A virtuális ürítés feladatát a 17–1. ábrán bemutatásra kerülő virtuális ürítési művelet

végrehajtásával hajtjuk végre, amely az adatot kiveszi a multiplex virtuális 90 szoftvercsatorna várakozási sorából, és átadja a valódi 36A szoftvercsatormának. A várakozási feladatot a 16–4. ábrán bemutatásra kerülő virtuális kiválasztási művelet végrehajtásával végezzük el. Eddigre az ügyféloldali 30 elfogómodullal vettük a 10 webböngészőtől kiinduló kommunikációt, és a külső 35 kommunikációs kapcsolaton keresztül továbbítottuk a kiszolgálóoldali 40 elfogómodulnak.

A 13. ábrán az akár a kiszolgálóoldali 40 elfogómodulhoz, akár az ügyféloldali 30 elfogómodulhoz tartozó 68, 69 valódiszoftvercsatorna-menedzser lehetséges működésének folyamatábráját tüntettük fel. A 12. ábrán a 69 valódiszoftvercsatorna-menedzser ugyanazt a feladatot tölti be a kiszolgálóoldali 40 elfogómodulban, mint a 68 valódiszoftvercsatorna-menedzser az ügyféloldali 30 elfogómodulban. A 301 lépésben az első valódi szoftvercsatorna létrehozásával a kiszolgálóoldali 40 elfogómodullal létrehozunk egy „jól ismert 37B szoftvercsatornát”, amellyel az ügyféloldali 30 elfogómodultól vesszük a kiszolgálóoldali 40 elfogómodullal kapcsolatos szoftvercsatorna-kérelmeket. Amikor egy valódi esemény történik a kiszolgálóoldali 40 elfogómodul valódi 36B csatornáján, az eseményt a 305 lépésben megvizsgáljuk. Jelen esetben az esemény, adat vétele a valódi 36A szoftvercsatormán zajlik, tehát a 305 lépésről a 320 lépés felé haladunk. A valódi 36B szoftvercsatormán keresztül vett adatot megvizsgáljuk, és mivel jelen példában az adat egy, a 10 webböngészőtől kiinduló és az ügyféloldali 30 elfogómodullal továbbított kommunikáció, egy új virtuális szoftvercsatornát kell létrehoznunk a kiszolgálóoldali 40 elfogómodulban. Tehát a 13. ábrán a 320 lépésről a 321 lépés felé haladunk tovább. A kiszolgálóoldali 69 valódiszoftvercsatorna-menedzserrel ezután végrehajtjuk a 321, 322, 323 és 324 lépéseket: a kiszolgálóoldali 69 valódiszoftvercsatorna-menedzserrel a 321 lépésben létrehozunk egy multiplex virtuális 95 szoftvercsatornát, a 322 lépésben érvénytelenítjük a multiplex 95 szoftvercsatorna tevékenységidőzítójét, és a 323 lépésben elindítunk egy alkalmazást a kiszolgálóoldali 85 elfogási művelet végrehajtására. A valódi 36B szoftvercsatornán vett adatot ezután a multiplex virtuális 95 szoftvercsatorna várakozási sorába továbbítjuk, és egy virtuális eseményt kijelzünk.

A kiszolgálóoldali elfogási művelet végrehajtására szolgáló alkalmazást a 360 lépésben hozzuk létre (lásd 15. ábra). Létrejötté után a 361 lépésben a kiszolgálóoldali elfogási művelettel a multiplex virtuális 95 szoftvercsatornától megkapjuk az ügyféloldali 30 elfogómodultól érkezett adatot, amely megfelel a 10 webböngészőtől kiinduló kommunikációnak. Miután az adatot megkaptuk az ügyféloldali 30 elfogómodultól, a kiszolgálóoldali elfogási művelettel a 362 lépésben feldolgozzuk a korábban a kiszolgálóoldali 40 elfogómodulnál leírt módon. Az információ feldolgozása után a 363 lépésben a kiszolgálóoldali elfogási művelettel létrehozunk egy szimplex virtuális 75 szoftvercsatornát a 16–1. ábrán bemutatásra kerülő virtuális létrehozás művelet végrehajtásával. A kiszolgálóoldali elfogási

művelettel ezután a 364 lépésben a 10 webböngészőtől kiinduló kommunikációt a 364 lépésben elküldjük a szimplex virtuális 75 szoftvercsatornának, a 16–2. ábrán bemutatásra kerülő virtuális küldés művelet végrehajtásával. A kiszolgálóoldali elfogási művelettel ezután a 365 lépésben egy, a 17–1. ábrán bemutatásra kerülő virtuális üritést hajtunk végre, hogy a szimplex virtuális 75 szoftvercsatorna várakozási sorában lévő adatokat a valódi 60C szoftvercsatornára átküldjük, majd egy, a szimplex virtuális 75 szoftvercsatornán bekövetkező eseményre várunk. A várakozási műveletet a 16–4. ábrán bemutatásra kerülő virtuális kiválasztás művelet elvégzésével hajtjuk végre. Amikor a kiszolgálóoldali elfogási művelettel létrehoztuk a szimplex virtuális 75 szoftvercsatornát, akkor ennek megfelelően egy valódi 60C szoftvercsatorna is létrejött. A 10 webböngészőtől kiinduló kommunikáció szimplex virtuális 75 szoftvercsatornára való küldésével a kiszolgálóoldali elfogási művelettel eljuttattuk a 10 webböngészőtől kiinduló információt a 20 webkiszolgálóhoz.

Amikor a kiszolgálóoldali 40 elfogómodullal megkapjuk a választ a 20 webkiszolgálótól a valódi 60C szoftvercsatornán, egy valódi esemény történik, és a kiszolgálóoldali 69 valódiszoftvercsatorna-menedzserrel a 13. ábra 302 lépéséről továbblépünk a 305 lépésre, hogy megvizsgáljuk ezt a valódi 60C szoftvercsatornán történt eseményt. Jelen esetben ez egy létező virtuális szoftvercsatorna számára érkezett adat lesz, így a 13. ábra 324 lépésében a valódi 60C szoftvercsatornán vett adatot a szimplex virtuális 75 szoftvercsatorna várakozási sorába továbbítjuk, és egy virtuális eseményt kijelzünk. Amikor a virtuális esemény jelzése megtörténik, a kiszolgálóoldali elfogási művelettel továbblépünk a 15. ábra 366 lépéséről, és a 370 lépésben megvizsgáljuk az eseményt. Ha az esemény egy szoftvercsatorna lezárása, akkor egy hibafeltétel teljesült, és válaszként a 375 lépésben egy hibaüzenetet állítunk elő. Ha az esemény adat vétele, akkor a 371 lépésben a kiszolgálóoldali elfogási művelettel egy, a 16–3. ábrán bemutatásra kerülő virtuális vétel műveletet hajtunk végre, hogy megkapjuk a 20 webkiszolgáló választát a szimplex virtuális 75 csatornától. A kiszolgálóoldali elfogási művelettel ezután a 372 lépésben végrehajtjuk a szimplex virtuális 75 szoftvercsatorna virtuális lezárását, amelyet a 17–2. ábrán mutatunk be, valamint a 373 lépésben feldolgozzuk a választ a kiszolgálóoldali elfogómodul számára a korábban leírt módon (lásd például a 6. és 10. ábrát).

Akármerre haladunk tovább a 370 lépésről, a 374 lépésben töröljük a szimplex virtuális 75 szoftvercsatornát. A kiszolgálóoldali elfogási művelettel ezután a 376 lépésben egy virtuális küldési műveletet hajtunk végre a multiplex virtuális 95 szoftvercsatornához, hogy azon keresztül továbbítsuk a 20 webkiszolgálótól kiinduló kommunikációt az ügyféloldali 30 elfogómodulnak. A kiszolgálóoldali elfogási művelettel ezután a 377 lépésben végrehajtunk egy virtuális üritési műveletet, amellyel a multiplex virtuális 95 szoftvercsatorna várakozási sorában lévő adatokat töröljük. A kiszolgálóoldali elfogási művelettel ezután a 378 lépésben vég-

rehajtunk egy virtuális lezárási műveletet, amellyel lezárjuk a multiplex virtuális 95 szoftvercsatornát. Végül a kiszolgálóoldali elfogási művelettel a 379 lépésben töröljük a multiplex virtuális 95 szoftvercsatornát, és a 380 lépésben befejezzük a folyamatot.

A kiszolgálóoldali elfogási művelettel végrehajtjuk a virtuális küldési és üritési műveleteket a multiplex virtuális 95 szoftvercsatornán. Ezek eseményeket váltanak ki a valódi 36A szoftvercsatornán, és az ügyféloldali 68 valódiszoftvercsatorna-menedzserrel továbblépve a 302 lépésről a 305 lépésben megvizsgáljuk az eseményt. Mivel az adatok a valódi 36A szoftvercsatornán keresztül érkeztek, a 320 lépésben az adatokat a multiplex virtuális 90 szoftvercsatorna várakozási sorába továbbítjuk. Ezért amikor a valódi 36A szoftvercsatornán vesszük a 20 webkiszolgáló választát a valódi 36B szoftvercsatornától a külső 35 kommunikációs kapcsolaton keresztül, ezt az információt szétválasztjuk és a multiplex virtuális 95 szoftvercsatornához továbbítjuk. Az adat vétele egy virtuális eseményt okoz, amit a 324 lépésben kijelzünk, és a 340 lépésben az ügyféloldali elfogási művelettel megvizsgálunk.

Ha az esemény egy „szoftvercsatorna zárva” válasz, akkor a 345 lépésben az ügyféloldali elfogási művelettel válaszképpen létrehozunk egy hibaüzenetet, és továbblépünk a 14. ábrán látható 344 lépésre. Ha az esemény egy „adat fogadva” válasz, mint jelen példánkban, akkor a 341 lépésben az ügyféloldali elfogási művelettel egy virtuális vétel műveletet hajtunk végre, hogy fogadni tudjuk a multiplex virtuális 90 szoftvercsatorna választát. Az adatnak a multiplex virtuális 90 szoftvercsatornától történő fogadása után az ügyféloldali elfogási művelettel a 342 lépésben végrehajtunk egy virtuális lezárási műveletet, hogy lezárjuk a multiplex virtuális 90 szoftvercsatornát. Az ügyféloldali elfogási művelettel ezután a 343 lépésben feldolgozzuk a választ az ügyféloldali 30 elfogómodul számára a már korábban leírt módon (lásd például a 4. és 8. ábrát).

Ezután a 344 lépésben az ügyféloldali elfogási művelettel töröljük a multiplex virtuális 90 szoftvercsatornát, majd a 346 lépésben egy virtuális küldés műveletet hajtunk végre, hogy a választ elküldjük a 10 webböngészőnek a szimplex virtuális 70 szoftvercsatornán keresztül. Amikor a virtuális küldés megtörténik, az ügyféloldali elfogási művelettel a 347 lépésben végrehajtunk egy virtuális üritési műveletet, hogy a szimplex virtuális 70 szoftvercsatorna várakozási sorában lévő adatokat átküldjük a valódi 60B szoftvercsatornának, majd a 348 lépésben végrehajtunk egy virtuális lezárási műveletet, hogy lezárjuk a szimplex virtuális 70 szoftvercsatornát. A szimplex virtuális 70 szoftvercsatorna lezárása után azt a 349 lépésben töröljük, és a 350 lépésben befejezzük az ügyféloldali elfogási művelet tevékenységét.

Amint az szakember számára nyilvánvaló, a találmány szerinti szimplex és multiplex virtuális szoftvercsatornák, továbbá az ügyféloldali és kiszolgálóoldali elfogási műveletek csupán egyetlen kiviteli alakra vonatkozóan kerültek bemutatásra, de e funkciók sokasága létrehozható egyetlen ügyféloldali 30 elfogómodulon

vagy kiszolgálóoldali 40 elfogómodulon belül. Következésképpen a találmány szerinti ügyféloldali 30 elfogómodul és kiszolgálóoldali 40 elfogómodul létrehozhat kettőjük között egy TCP/IP kapcsolatot, majd 10 webböngészők és 20 webkiszolgálók sokaságától kiinduló kommunikációt multiplexelhet a TCP/IP kapcsolatra, annak fenntartása mellett.

Az ügyféloldali 68 valódiszoftvercsatorna-menedzser és a kiszolgálóoldali 69 valódiszoftvercsatorna-menedzser további funkciói a 16–1–16–4. ábrák, valamint a 17–1. és 17–2. ábra segítségével érthetők meg. Ezek az ábrák az ügyféloldali 30 elfogómodult és a kiszolgálóoldali 40 elfogómodult mutatják be virtuális létrehozás, virtuális küldés, virtuális vétel, virtuális kiválasztás, virtuális ürités vagy virtuális lezárás műveletek végrehajtása során, amelyek lefolyását a 14. és a 15. ábra segítségével már bemutattuk.

Amikor egy, a 14. ábra 333 lépésében és a 15. ábra 363 lépésében feltüntetett virtuális létrehozás művelete zajlik, akkor a 16–1. ábra 400 lépésével kezdődő műveleteket hajtjuk végre. A 400 lépésben megkezdett művelet során a 405 lépésben a 68 valódiszoftvercsatorna-menedzserrel meghatározzuk, hogy szükség van-e egy valódi softvercsatornára. Ha már létezik egy valódi softvercsatorna, például amikor a virtuális létrehozással egy multiplex virtuális softvercsatornát hozunk létre, amelyet egy létező valódi softvercsatornához kell csatlakoztatni, akkor a 405 lépésről a „nem” ágon haladunk tovább, és a virtuális softvercsatornát a 409 lépésben a valódi softvercsatornával összekapcsoljuk. Ha viszont szükség van egy valódi softvercsatornára, akkor a 405 lépésről az „igen” ágon haladunk tovább, és a 406 lépésben létrehozunk egy valódi softvercsatornát. A valódi softvercsatornát ezután a 408 lépésben ellenőrzés céljából hozzáadjuk az eseménylistához. A valódi softvercsatorna létrehozása és egy csatlakozás megteremtése után a virtuális softvercsatornát a 409 lépésben összekapcsoljuk a valódi softvercsatornával, és a létrehozás műveletet a 410 lépésben befejezzük.

Amikor egy, a 14. ábra 334 lépésében és 346 lépésében vagy a 15. ábra 364 és 376 lépéseiben feltüntetett virtuális küldés művelete zajlik, akkor a 16–2. ábra 420 lépésével kezdődő műveleteket hajtjuk végre. Az adatokat a 427 lépésben hozzáadjuk a virtuális softvercsatorna várakozási sorához, és amikor ez készen van, a küldési műveletet a 428 lépésben befejezzük.

A 14. ábra 331 és 341 lépése, valamint a 15. ábra 361 és 371 lépése által tükrözött virtuális vétel művelet a 16–3. ábra 430 lépésével kezdődő műveletek elvégzésével hajtható végre. A 435 lépésben a virtuális softvercsatorna várakozási sorát megvizsgáljuk annak meghatározása céljából, hogy tartalmaz-e adatot. Ha tartalmaz, akkor a 435 lépésről az „igen” ágon továbbhaladva a 436 lépésben az adatokat visszaküldjük a vétel műveletet meghívó művelethez. Ha nincs adat a virtuális softvercsatorna várakozási sorában és a softvercsatorna nincs lezáródóként megjelölve, akkor a 440 lépés „nem” ágon továbbhaladva a 441 lépésben semmit sem küldünk vissza. Ha nincs adat a virtuális

softvercsatorna várakozási sorában és a softvercsatorna lezáródóként van megjelölve, akkor a 440 lépés „igen” ágon továbbhaladva a 442 lépésben a softvercsatornát lezáróként megjelöljük, valamint a 443 lépésben a „softvercsatorna lezárva” választ visszaküldjük a vétel műveletet meghívó művelethez.

Amikor egy, a 14. ábra 326 lépésében és 336 lépésében, valamint a 15. ábra 366 lépésében feltüntetett virtuális kiválasztás művelete zajlik, akkor a 16–4. ábra 445 lépésével kezdődő műveleteket hajtjuk végre. A 446 lépésben először azt határozzuk meg, hogy adat- vagy virtuális lezárás művelet van-e folyamatban a kiválasztott virtuális softvercsatormán. Ha sem adat-, sem virtuális lezárás nincs folyamatban, akkor a 446 lépésről a „nem” ágon haladunk tovább, és a 447 lépésben egy, a virtuális softvercsatormán érkező virtuális eseményre várunk, majd egy ilyen esemény észlelésekor a műveletet a 448 lépésben befejezzük. Ha a kiválasztott virtuális softvercsatormán adat- vagy virtuális lezárás van folyamatban, akkor már történt virtuális esemény, és a 446 lépés „igen” ágon továbbhaladva fejezzük be a műveletet.

Amikor egy, a 14. ábra 335 lépésében és 347 lépésében, valamint a 15. ábra 365 és 377 lépésében feltüntetett virtuális ürités művelete zajlik, akkor a 17–1. ábra 450 lépésével kezdődő műveleteket hajtjuk végre. Meghívásakor a virtuális üritési művelettel a 455 lépésben meghatározzuk, hogy van-e üritendő adat a virtuális softvercsatorna várakozási sorában. Ha nincs adat a virtuális softvercsatorna várakozási sorában, akkor az ürités művelet egyszerűen befejeződik, és az irányítás visszakerül a műveletet meghívó művelethez. Ha van adat a várakozási sorban, akkor a 455 lépésről az „igen” ágon haladunk tovább, és a 460 lépésben meghatározzuk, hogy egy multiplex virtuális softvercsatornaé-e a virtuális softvercsatorna várakozási sora. Ha ez egy multiplex softvercsatorna, akkor a softvercsatorna egyedi azonosítóját és az átvitelben az adat mennyiségét megadó három bájtól álló fejlécet a 461 lépésben hozzáadjuk a valódi softvercsatorna pufferéhez. Akár multiplex softvercsatornáról, akár szimplex softvercsatornáról van szó, a valódi softvercsatornának szánt adatokat a 462 lépésben a valódi softvercsatorna pufferébe töltjük. Ha a valódi softvercsatorna pufferé tele van, akkor a 465 lépésről az „igen” ágon továbbhaladva a 466 lépésben a valódi softvercsatorna pufferében lévő adatokat elküldjük a valódi softvercsatormán keresztül. Ha a valódi puffer nincs tele, akkor a 465 lépésről a „nem” ágon haladunk tovább, és megvizsgáljuk, hogy van-e bármilyen más adat bármelyik másik multiplex virtuális softvercsatorna várakozási sorában, amelyet a valódi softvercsatornára kell küldeni. Ha a válasz igen, akkor a 470 lépésről az „igen” ágon haladunk tovább, és a valódi softvercsatorna pufferében lévő adatokat nem küldjük el, amíg a virtuális üritési művelet nem kerül még egyszer meghívásra, hogy ürítse valamelyik másik virtuális softvercsatorna várakozási sorát. Ha nincs más adat, vagy a többi multiplex virtuális softvercsatorna adatainak hozzáadása után a 466 lépés-

ben a valódi szoftvercsatorna pufferében lévő adatokat elküldjük a valódi szoftvercsatornán. Miután a virtuális üritést meghívó művelethez tartozó összes virtuális szoftvercsatorna várakozási sorában lévő adatot elküldtük a valódi szoftvercsatornán, a virtuális ürités műveletet a 467 lépésben befejezzük.

Amikor egy, a 14. ábra 342 lépésében és 348 lépésében, valamint a 15. ábra 372 és 378 lépésében feltüntetett virtuális lezárás művelete zajlik, akkor a 17–2. ábra 480 lépésével kezdődő műveleteket hajtjuk végre. Amikor egy virtuális lezárás műveletet hívunk meg, a 485 lépésben először megvizsgáljuk, hogy a virtuális lezárás egy multiplex virtuális csatorná-e. Ha ez egy multiplex virtuális szoftvercsatorna, akkor a 485 lépésről az „igen” ágon haladunk tovább, és a 486 lépésben egy „lezárás” műveletjelzőt adunk a virtuális szoftvercsatorna várakozási sorához. Akár egy multiplex virtuális csatorná-e a virtuális lezárás művelet, akár nem, a 487 lépésben a virtuális lezárás művelettel meghívjuk a virtuális ürités műveletet, majd a 488 lépésben megszüntetjük a kapcsolatot a valódi csatornával. Ezt követően a 490 lépésben meghatározzuk, hogy a virtuális lezárás egy szimplex virtuális csatorná-e, és ha nem, a 495 lépésről a „nem” ágon haladunk tovább. Mivel a lezárás egy multiplex csatorná-e, a 495 lépésben megvizsgáljuk, hogy ez az utolsó multiplex virtuális szoftvercsatorna-e, és ha igen, akkor a 496 lépésben beállítjuk a multiplex tevékenység időzítőjét. Ha ez nem az utolsó multiplex virtuális szoftvercsatorna, akkor a 496 lépést kihagyjuk.

Ha a virtuális lezárás egy szimplex virtuális csatorná-e, akkor az ennek megfelelő valódi szoftvercsatornát a 491 lépésben levesszük az eseménylistáról, a valódi szoftvercsatornát pedig a 492 lépésben lezárjuk, majd töröljük. A szoftvercsatorna lehet akár szimplex, akár multiplex virtuális szoftvercsatorna, mindenképpen lezártként jelöljük meg a 497 lépésben, és a lezárás műveletet befejezzük a 498 lépésben.

Röviden visszatérünk a 13. ábrára, mivel az kapcsolódik a 16–1–16–4., valamint a 17–1. és 17–2. ábrához. Amikor egy valódi esemény történik, a 302 lépést követően a 68 valódiszoftvercsatorna-menedzserrel megvizsgáljuk az eseményt annak alapján, ahogyan az esemény előállt. Ha az esemény a multiplex-szoftvercsatorna-tevékenység 17–2. ábra 496 lépésében beállított időzítőjének a lejártá, akkor a 305 lépésről a 312 lépésre térünk át. Amint a 13. ábra mutatja, ezt követően a 68 valódiszoftvercsatorna-menedzserrel a 312 lépésben lezárjuk a multiplex virtuális szoftvercsatornát, és a 313 lépésben töröljük az annak a szoftvercsatornának megfelelő multiplex virtuális szoftvercsatornát, amely az ügyféloldali 30 elfogómodult a kiszolgálóoldali elfogómodullal köti össze. A 68 valódiszoftvercsatorna-menedzserrel ezután megvárjuk a következő valódi eseményt. A 322 lépésben a multiplex eseményidőzítőt a multiplex virtuális szoftvercsatorna létrejöttével nullázzuk.

Ha a valódi szoftvercsatornán történő esemény egy valódi szoftvercsatorna lezárása, például ha a 20 webkiszolgáló végrehajt egy lezárás műveletet a 20 webki-

szolgáló és a kiszolgálóoldali elfogómodul közötti csatlakozáson, akkor a 305 lépésről a 309 lépés felé haladunk. A 309 lépésben 68 valódi szoftvercsatorna menedzserrel eltávolítjuk a valódi szoftvercsatornát a valódi esemény-listáról, és a 310 lépésben lekapcsoljuk a virtuális szoftvercsatornát, illetve multiplex virtuális szoftvercsatornák esetén a szoftvercsatornákat a valódi szoftvercsatornától, illetve szoftvercsatornáktól. A 68 valódiszoftvercsatorna-menedzserrel ezután a 311 lépésben a virtuális szoftvercsatornát lezáródóként megjelöljük, és egy virtuális eseményt kijelzünk, majd amikor minden adatot kiüritettünk a virtuális szoftvercsatorna várakozási sorából, a virtuális szoftvercsatornát lezárjuk. A virtuális szoftvercsatorna lezáródóként való megjelölése után a 68 valódiszoftvercsatorna-menedzserrel a 315 lépésben meghatározzuk, hogy a lezárandó valódi szoftvercsatorna egy szimplex szoftvercsatorna-e, vagy sem. Ha a lezáródó valódi szoftvercsatorna egy szimplex szoftvercsatorna, akkor a 316 lépésben a valódi szoftvercsatornát lezárjuk és töröljük. Ezután a 68 valódiszoftvercsatorna-menedzserrel ismét a 302 lépésben a következő valódi eseményre várunk.

Ha nem egy szimplex szoftvercsatornát zárunk le, akkor a 315 lépésről a „nem” ágon haladunk tovább, és a 68 valódiszoftvercsatorna-menedzserrel a következő valódi eseményre várunk. Tehát a multiplex valódi szoftvercsatornát, azaz az ügyféloldali 30 elfogómodult és a kiszolgálóoldali 40 elfogómodult összekötő szoftvercsatornát csak a multiplex-szoftvercsatorna-tevékenység időzítőjének lejártá után tudjuk lezárni. Ez lehetővé teszi az ügyféloldali 30 elfogómodul és a kiszolgálóoldali 40 elfogómodul közötti kapcsolat fenntartását a modulok közti utolsó kommunikáció után eltelt, a felhasználó által előre meghatározott ideig. A 10 webböngésző egy ezt követő csatlakozási kérélmé esetén a multiplex-szoftvercsatorna-tevékenység időzítőjének lejártá előtt a kommunikáció végbemehet az ügyféloldali 30 elfogómodul és a kiszolgálóoldali 40 elfogómodul közötti újracsatlakozás nélkül, kiküszöbölve ezáltal egy újracsatlakozás költségét.

A 13. ábra utolsó bemutatásra kerülő útvonala azt az esetet tükrözi, amikor egy valódi esemény történik, és ez az esemény adatok vétele, amely a multiplex valódi szoftvercsatornán, azaz a 12. ábra 36A és 36B szoftvercsatornáin keresztül történik. Amikor adat vételére kerül sor a multiplex valódi 36A és 36B szoftvercsatornákon, akkor ezt az adatot megvizsgáljuk, és abban az esetben, ha az adat tartalmazza a lezárás művelet jelzőjét, amelyet például a 17–2. ábra 486 lépésében a virtuális várakozási sorhoz adtunk, akkor egy virtuális lezárási műveletet hajtunk végre a 320 lépésről a 310 lépés felé haladva. A 68 valódiszoftvercsatorna-menedzserrel a 310 lépésben szétkapcsoljuk egymástól a valódi szoftvercsatornát és a valódi szoftvercsatornán keresztül vett adatok által azonosított multiplex virtuális szoftvercsatornát, majd a 311 lépésben a virtuális szoftvercsatornát „lezáródóként” megjelöljük, és egy virtuális eseményt kijelzünk. Mivel a lezárás egy multiplex virtuális szoftvercsatorna lezárása, a 315 lé-

pésről az „igen” ágon haladunk tovább, és a 68 valódi-szoftvercsatorna-menedzserrel a 302 lépésben egy újabb valódi eseményre várunk.

A 13–17. ábrákon bemutatott műveletek végrehajtásával a találmány lehetővé teszi állandó kapcsolat létesítését az első 5 számítógép és a második 6 számítógép között a külső 35 kommunikációs kapcsolaton keresztül. Az állandó kapcsolat addig marad fenn, amíg minden, a 10 webböngészőtől kiinduló kommunikáció befejeződik, és a 10 webböngészőtől kiinduló kommunikációk sokasága elfogásra és külső 35 kommunikációs kapcsolatra multiplexelésre kerül, miközben az állandó kapcsolat fennmarad. Az ügyfél-kiszolgáló specifikus adatfolyam ezután szétválasztható http-adatfolyamok sokaságának létrehozása céljából, majd a http-adatfolyamok sokaságát a 20 webkiszolgálónak átadhatjuk. Az állandó kapcsolat addig fennmarad, amíg minden, a 20 webkiszolgálótól kiinduló kommunikáció befejeződik. A 20 webkiszolgálótól kiinduló adatok sokasága elfogásra és a külső 35 kommunikációs kapcsolatra multiplexelésre kerül, miközben az állandó kapcsolat fennmarad. Továbbá az ügyfél-kiszolgáló specifikus adatfolyamot http-adatfolyamok sokaságának létrehozása céljából demultiplexelhetjük, és a http-adatfolyamok sokaságát a 10 webböngészőnek átadhatjuk.

Eddigi leírásunkban és az ábrákon a találmány jellemző, előnyös kiviteli alakjait mutattuk be, és bár bizonyos szakkifejezések használata elkerülhetetlennek bizonyult, ezeket csupán általános leíró jelleggel, nem pedig korlátozás céljából használtuk. Mindazonáltal találmányunk oltalmi körét a soron következő szabadalmi igénypontok határozzák meg.

SZADALMI IGÉNYPONTOK

1. Eljárás egy első számítógépen (5) lévő első alkalmazástól egy kommunikációs kapcsolaton (35) keresztül egy második számítógépen (6) lévő második alkalmazáshoz továbbított adatok mennyiségének csökkentésére, amelynek során az adatokat egy külső kommunikációs kapcsolaton (35) keresztül továbbítjuk az első számítógéptől (5) a második számítógéphez (6) TCP kommunikációs protokoll felhasználásával, *azzal jellemezve*, hogy az első számítógépen (5) válaszul az első alkalmazás minden egyes csatlakozási kérelmére egy első virtuális szoftvercsatornát (70) hozunk létre az első alkalmazástól kiinduló kérelemadatok vételére; az első számítógépen (5) egy első valódi szoftvercsatornát (36A) hozunk létre és a második számítógépen (6) egy második valódi szoftvercsatornát (36B) hozunk létre az első számítógép (5) és a második számítógép (6) külső kommunikációs kapcsolaton (35) keresztül történő összekapcsolására; a második számítógépen (6) létrehozunk egy-egy második virtuális szoftvercsatornát (75) az első alkalmazás által kibocsátott minden egyes csatlakozási kérelemhez úgy, hogy a második virtuális szoftvercsatornát (75) egy, az első számítógépen (5) az első alkalmazás csatlakozási kérelmére vá-

laszképpen létrehozott első virtuális szoftvercsatornához (70) rendeljük hozzá; az első virtuális szoftvercsatornához (70) társított kérelemadatokat az első valódi szoftvercsatornára (36A) multiplexeljük; a multiplexelt kérelemadatokat a TCP protokoll felhasználásával a külső kommunikációs kapcsolaton (35) keresztül továbbítjuk a második valódi szoftvercsatornához (36B); a multiplexelt kérelemadatokat a külső kommunikációs kapcsolaton (35) keresztül fogadjuk; a külső kommunikációs kapcsolaton (35) keresztül a második valódi szoftvercsatorna (36B) által fogadott kérelemadatokat demultiplexeljük; a demultiplexelt kérelemadatokat átadjuk annak a második virtuális szoftvercsatornának (75), amelyet az első alkalmazás által kibocsátott kérelemre válaszképpen létrehozott első virtuális szoftvercsatornához (70) rendeltünk hozzá; a második virtuális szoftvercsatornán (75) keresztül vett kérelemadatokat átadjuk a második alkalmazásnak; és az első valódi szoftvercsatornát (36A) és a második valódi szoftvercsatornát (36B) mindaddig fenntartjuk, amíg az első virtuális szoftvercsatornát (70) létrehozó alkalmazáshoz tartozó kérelemadatokat át nem adjuk a második számítógépnek (6).

2. Az 1. igénypont szerinti eljárás, *azzal jellemezve*, hogy az első alkalmazás által kibocsátott kérelemhez tartozó második virtuális szoftvercsatornán (75) keresztül az első alkalmazás által kibocsátott kérelemre a második alkalmazás által válaszképpen kibocsátott válaszadatokat fogadjuk; a második virtuális szoftvercsatornán (75) keresztül vett válaszadatokat a második valódi szoftvercsatornára (36B) multiplexeljük; a multiplexelt válaszadatokat TCP protokoll felhasználásával a külső kommunikációs kapcsolaton (35) keresztül továbbítjuk az első valódi szoftvercsatornához (36A); a multiplexelt válaszadatokat vesszük a külső kommunikációs kapcsolaton (35) keresztül; az első valódi szoftvercsatornán (36A) keresztül fogadott válaszadatokat demultiplexeljük; a demultiplexelt válaszadatokat az első alkalmazás által kibocsátott kérelemre vonatkozó válaszként átadjuk az első alkalmazás által kibocsátott kérelemhez tartozó első virtuális szoftvercsatornának (70); és az első virtuális szoftvercsatornán (70) keresztül vett válaszadatokat átadjuk az első alkalmazásnak.

3. A 2. igénypont szerinti eljárás, *azzal jellemezve*, hogy a válaszadatok multiplexelése után a második virtuális szoftvercsatornát (75) lezárjuk; és a válaszadatoknak az első alkalmazás részére történt átadását követően az első virtuális szoftvercsatornát (70) lezárjuk.

4. A 3. igénypont szerinti eljárás, *azzal jellemezve*, hogy az első valódi szoftvercsatorna (36A) és a második valódi szoftvercsatorna (36B) fenntartása során az első valódi szoftvercsatornát (36A) és a második valódi szoftvercsatornát (36B) az összes első virtuális szoftvercsatorna (70) lezárását követően zárjuk le.

5. A 3. igénypont szerinti eljárás, *azzal jellemezve*, hogy az első valódi szoftvercsatorna (36A) és a második valódi szoftvercsatorna (36B) fenntartása során az első valódi szoftvercsatornát (36A) és a második valódi szoftvercsatornát (36B) az összes első virtuális szoft-

vercsatorna (70) lezárását követő előre meghatározott ideig tartjuk fenn.

6. Az 1. igénypont szerinti eljárás, *azzal jellemezve*, hogy az első alkalmazás webböngészőt (10) és a második alkalmazás webkiszolgálót (20) foglal magában.

7. Az 1. igénypont szerinti eljárás, *azzal jellemezve*, hogy külső kommunikációs kapcsolatként (35) vezeték nélküli kommunikációs kapcsolatot (35) használunk.

8. Berendezés egy első számítógépen (5) lévő első alkalmazástól egy kommunikációs kapcsolaton (35) keresztül egy második számítógépen (6) lévő második alkalmazáshoz egy külső kommunikációs kapcsolaton (35) keresztül, TCP kommunikációs protokoll felhasználásával továbbított adatok mennyiségének csökkentésére, *azzal jellemezve*, hogy az első alkalmazástól kiinduló kérelemadatok vételéhez az első alkalmazás minden egyes csatlakozási kérelmére válaszképpen az első számítógépen (5) egy első virtuális szoftvercsatornát (70) létrehozó eszköze; az első számítógép (5) és a második számítógép (6) külső kommunikációs kapcsolaton (35) keresztül történő összekapcsolásához az első számítógépen (5) egy első valódi szoftvercsatornát (36A) létrehozó és a második számítógépen (6) egy második valódi szoftvercsatornát (36B) létrehozó eszköze; az első alkalmazás által kibocsátott minden egyes csatlakozási kérelemhez a második számítógépen (6) egy-egy második virtuális szoftvercsatornát (75) annak az egy, az első számítógépen (5) az első alkalmazás csatlakozási kérelmére válaszképpen létrehozott első virtuális szoftvercsatornához (70) történő hozzárendelésével létrehozó eszköze, az első virtuális szoftvercsatornához (70) társított kérelemadatokat az első valódi szoftvercsatornára (36A) multiplexelő eszköze; a multiplexelt kérelemadatokat TCP protokoll felhasználásával a külső kommunikációs kapcsolaton (35) keresztül a második valódi szoftvercsatornához (36B) továbbító eszköze; a multiplexelt kérelemadatokat a külső kommunikációs kapcsolaton (35) keresztül fogadó eszköze; a külső kommunikációs kapcsolaton (35) keresztül a második valódi szoftvercsatorna (36B) által fogadott kérelemadatokat demultiplexelő eszköze; a demultiplexelt kérelemadatokat az első alkalmazás által kibocsátott kérelemre válaszképpen létrehozott első virtuális szoftvercsatornához (70) hozzárendelt második virtuális szoftvercsatornának (75) átvadó eszköze; a második virtuális szoftvercsatornán (75) keresztül vett kérelemadatokat a második alkalmazásnak átvadó eszköze; és az első valódi szoftvercsatornát (36A) és a második valódi szoftvercsatornát (36B) az első virtuális szoftvercsatornát (70) létrehozó alkalmazáshoz tartozó kérelemadatok második számítógépnek (6) való átvadásának befejezéséig fenntartó eszköze van.

9. A 8. igénypont szerinti berendezés, *azzal jellemezve*, hogy az első alkalmazás által kibocsátott kérelemhez tartozó második virtuális szoftvercsatornán (75) keresztül az első alkalmazás által kibocsátott kérelemre a második alkalmazás által válaszképpen kibocsátott

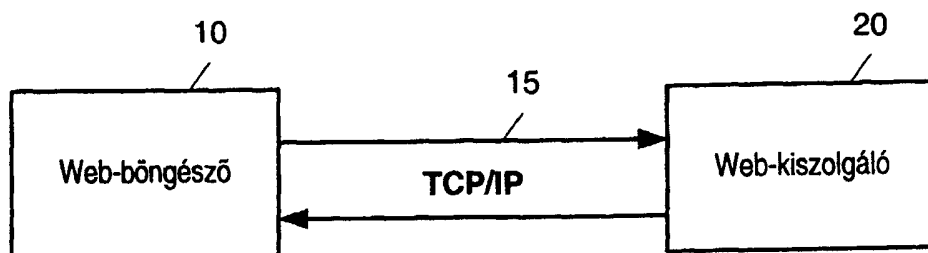
tott válaszadatokat fogadó eszköze; a második virtuális szoftvercsatornán (75) keresztül vett válaszadatokat a második valódi szoftvercsatornára (36B) multiplexelő eszköze; a multiplexelt válaszadatokat TCP protokoll felhasználásával a külső kommunikációs kapcsolaton (35) keresztül az első valódi szoftvercsatornához (36A) továbbító eszköze; a multiplexelt válaszadatokat a külső kommunikációs kapcsolaton (35) keresztül vevő eszköze; az első valódi szoftvercsatornán (36A) keresztül fogadott válaszadatokat demultiplexelő eszköze; a demultiplexelt válaszadatokat az első alkalmazás által kibocsátott kérelemre vonatkozó válaszként az első alkalmazás által kibocsátott kérelemhez tartozó első virtuális szoftvercsatornának (70) átvadó eszköze; és az első virtuális szoftvercsatornán (70) keresztül vett válaszadatokat az első alkalmazásnak átvadó eszköze van.

10. Számítógépi programtermék egy első számítógépen (5) lévő első alkalmazástól egy kommunikációs kapcsolaton keresztül egy második számítógépen (6) lévő második alkalmazáshoz egy külső kommunikációs kapcsolaton (35) keresztül, TCP kommunikációs protokoll felhasználásával továbbított adatok mennyiségének csökkentésére, *azzal jellemezve*, hogy számítógéppel olvasható tárolóközegén eltárolt tartalmaz egy,

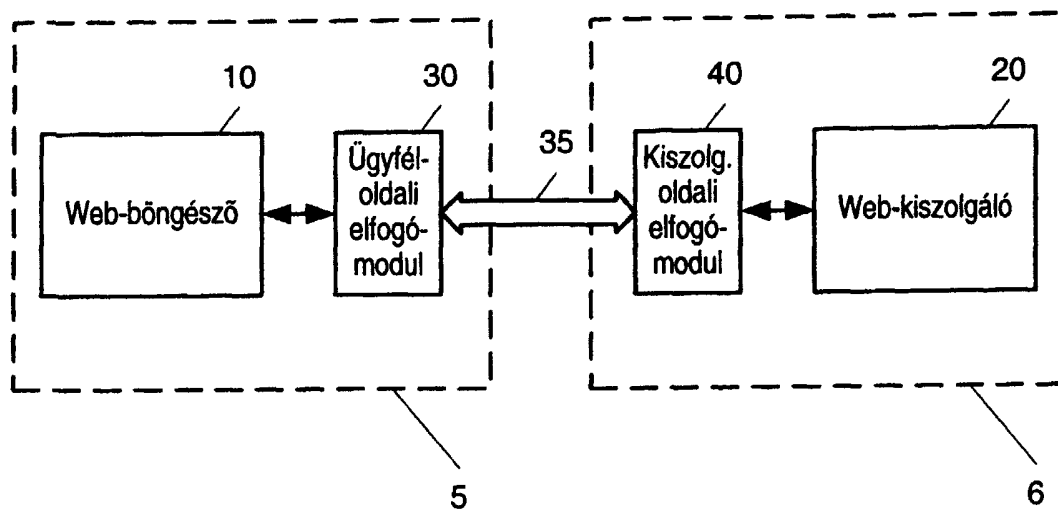
- az első alkalmazástól kiinduló kérelemadatok vételéhez az első alkalmazás minden egyes csatlakozási kérelmére válaszképpen az első számítógépen (5) egy első virtuális szoftvercsatornát (70) létrehozó;
- az első számítógép (5) és a második számítógép (6) külső kommunikációs kapcsolaton (35) keresztül történő összekapcsolásához az első számítógépen (5) egy első valódi szoftvercsatornát (36A) létrehozó és a második számítógépen (6) egy második valódi szoftvercsatornát (36B) létrehozó;
- az első alkalmazás által kibocsátott minden egyes csatlakozási kérelemhez a második számítógépen (6) egy-egy második virtuális szoftvercsatornát (75) annak az egy, az első számítógépen (5) az első alkalmazás csatlakozási kérelmére válaszképpen létrehozott első virtuális szoftvercsatornához (70) történő hozzárendelésével létrehozó;
- az első virtuális szoftvercsatornához (70) társított kérelemadatokat az első valódi szoftvercsatornára (36A) multiplexelő;
- a multiplexelt kérelemadatokat TCP protokoll felhasználásával a külső kommunikációs kapcsolaton (35) keresztül a második valódi szoftvercsatornához (36B) továbbító;
- a multiplexelt kérelemadatokat a külső kommunikációs kapcsolaton (35) keresztül fogadó;
- a külső kommunikációs kapcsolaton keresztül a második valódi szoftvercsatorna (36B) által fogadott kérelemadatokat demultiplexelő;
- a demultiplexelt kérelemadatokat az első alkalmazás által kibocsátott kérelemre válaszképpen

- létrehozott első virtuális szoftvercsatornához (70) hozzárendelt második virtuális szoftvercsatornának (75) átadó;
- a második virtuális szoftvercsatornán (75) keresztül vett kérelemadatokat a második alkalmazásnak átadó; és
- az első valódi szoftvercsatornát (36A) és a második valódi szoftvercsatornát (36B) az első virtuális szoftvercsatornát (70) létrehozó alkalmazáshoz tartozó kérelemadatok második számítógépnek (6) való átadásának befejezéséig fenntartó programot.

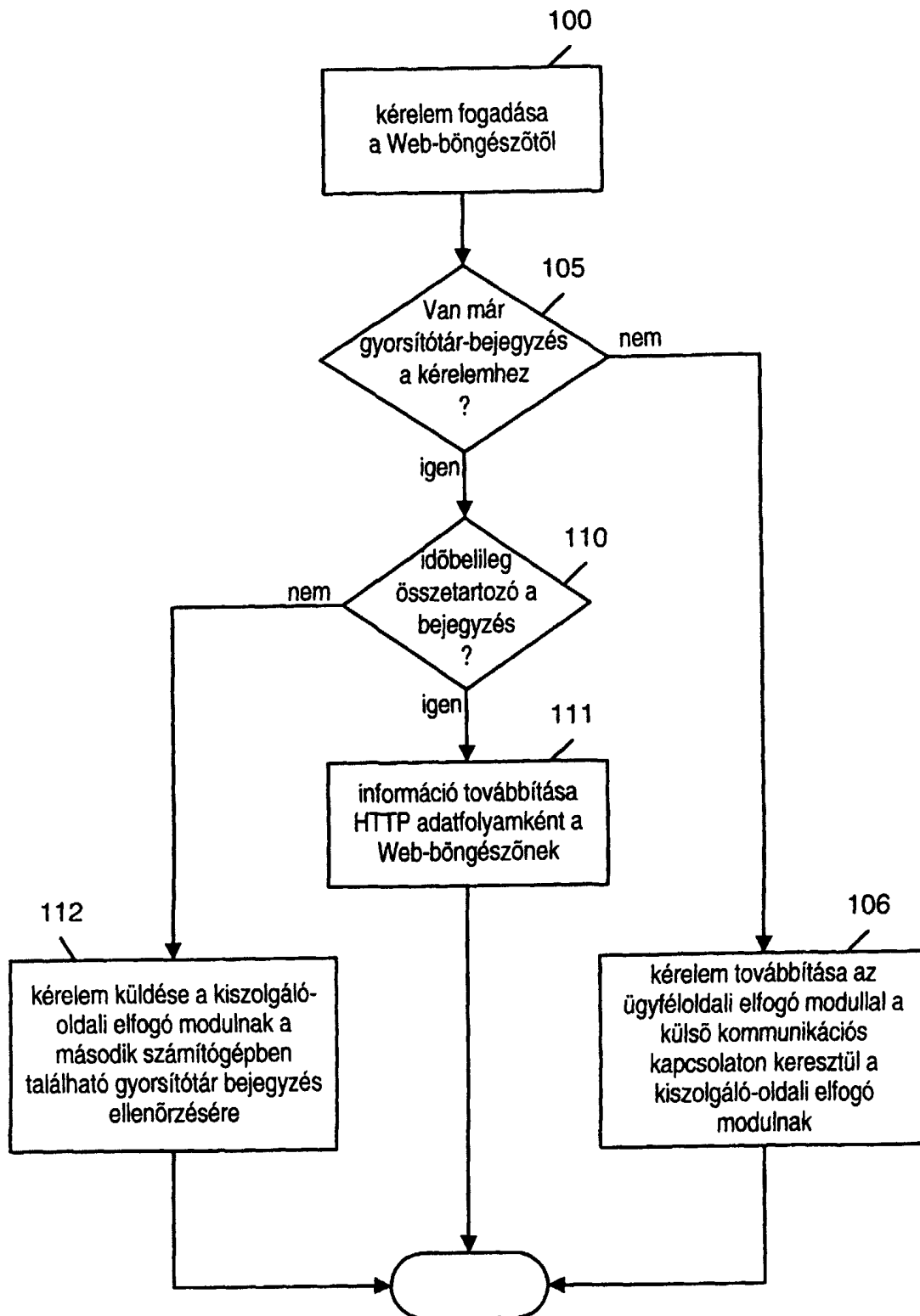
A TECHNIKA ÁLLÁSA



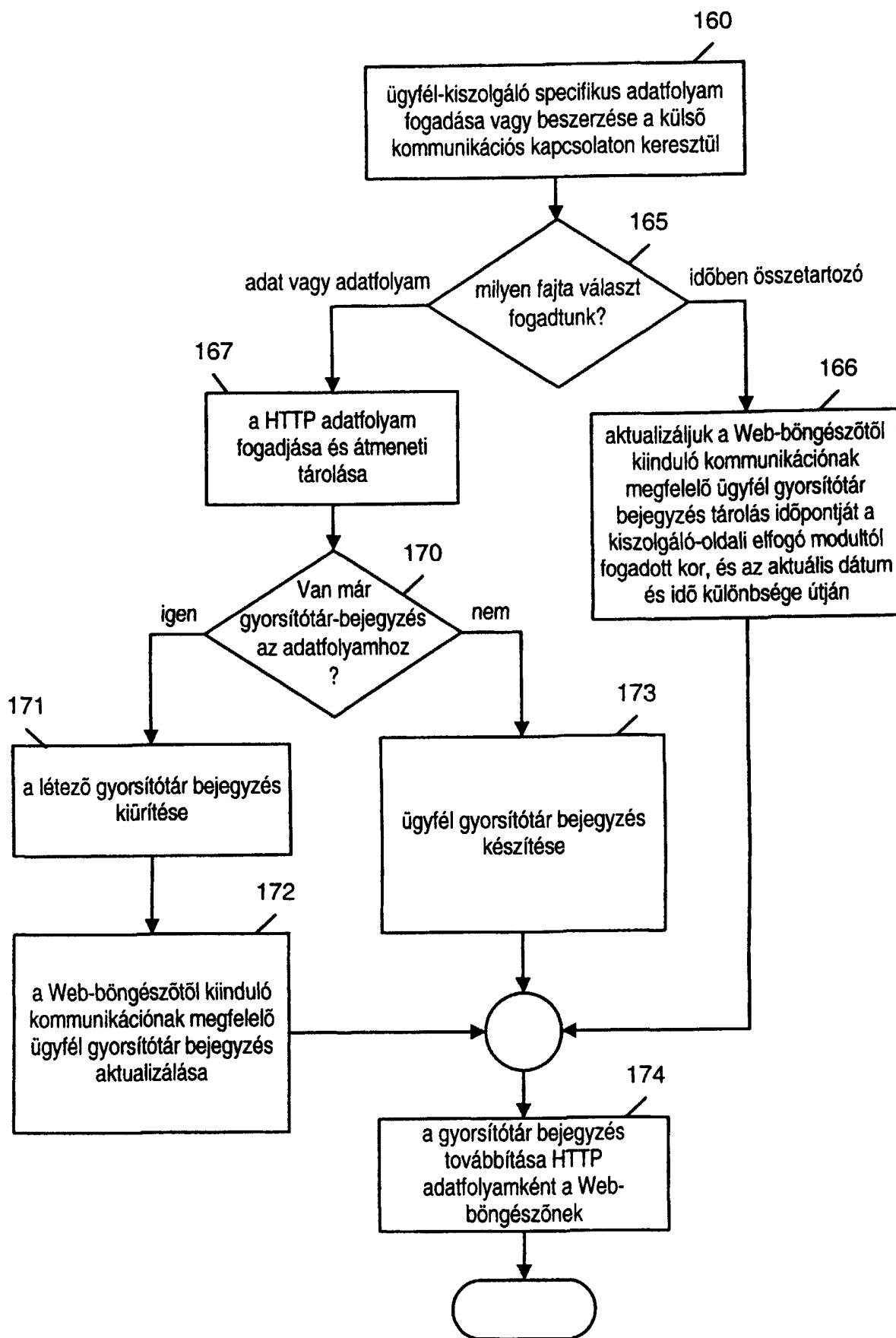
1. ábra



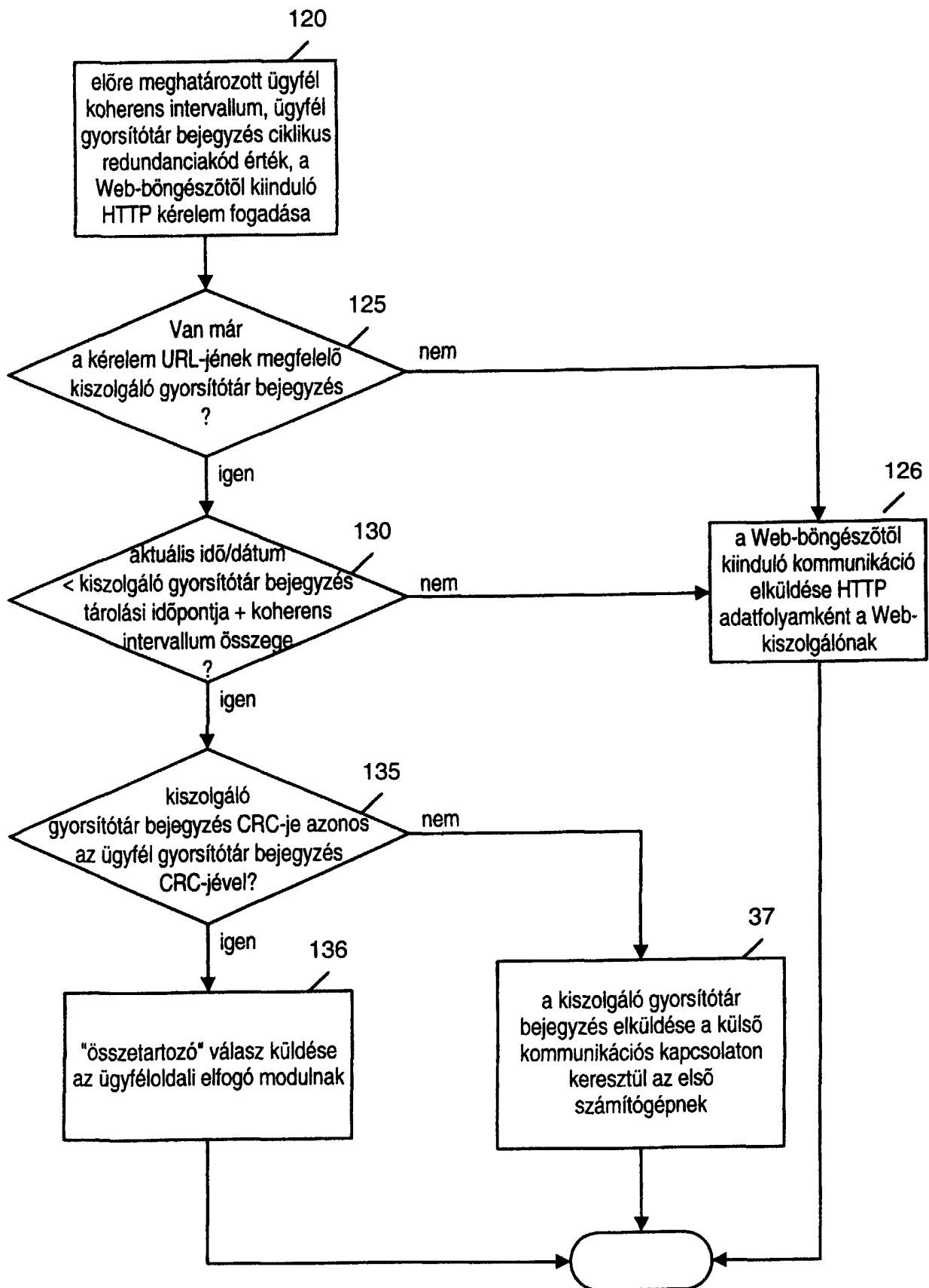
2. ábra



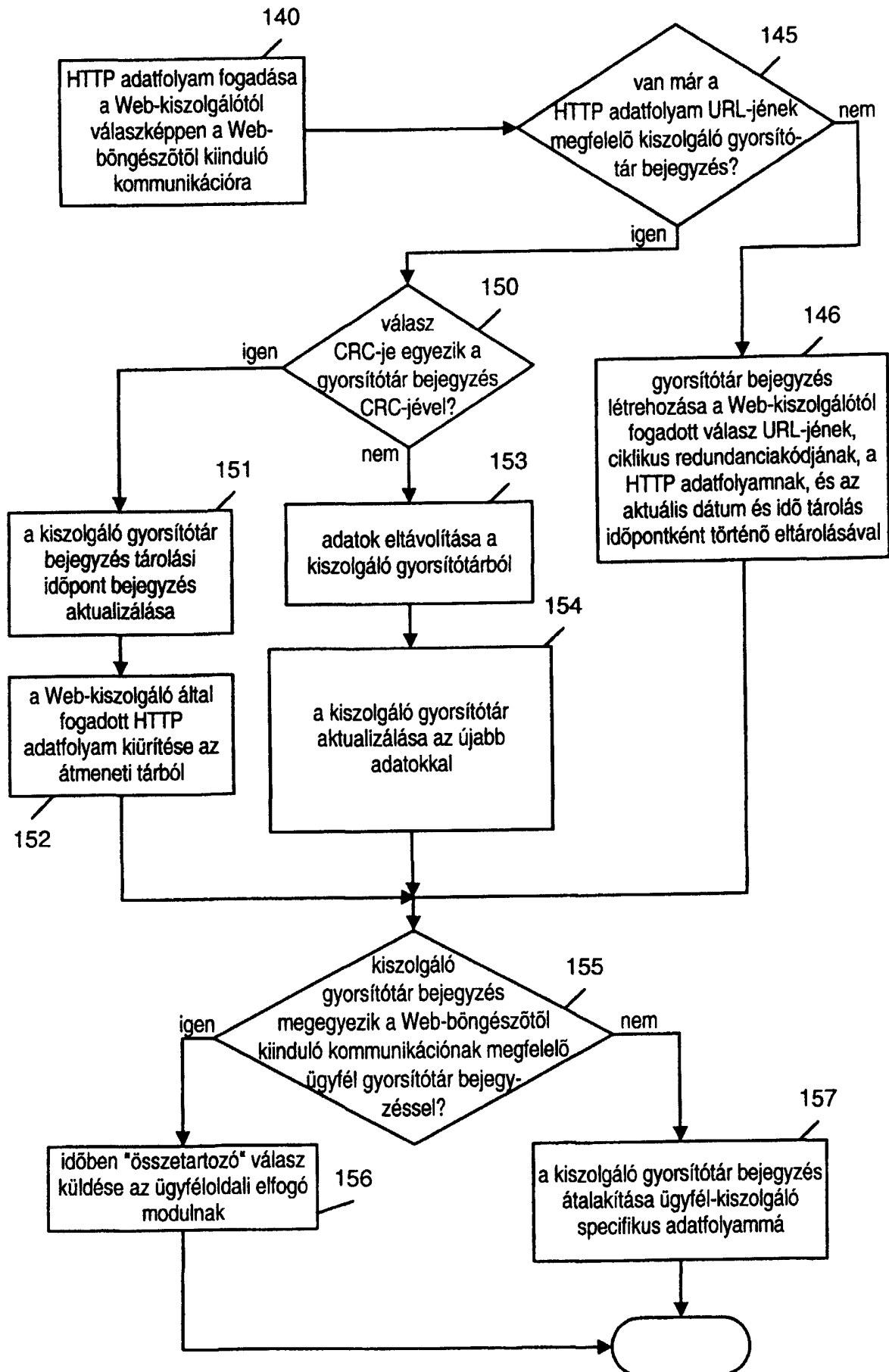
3. ábra



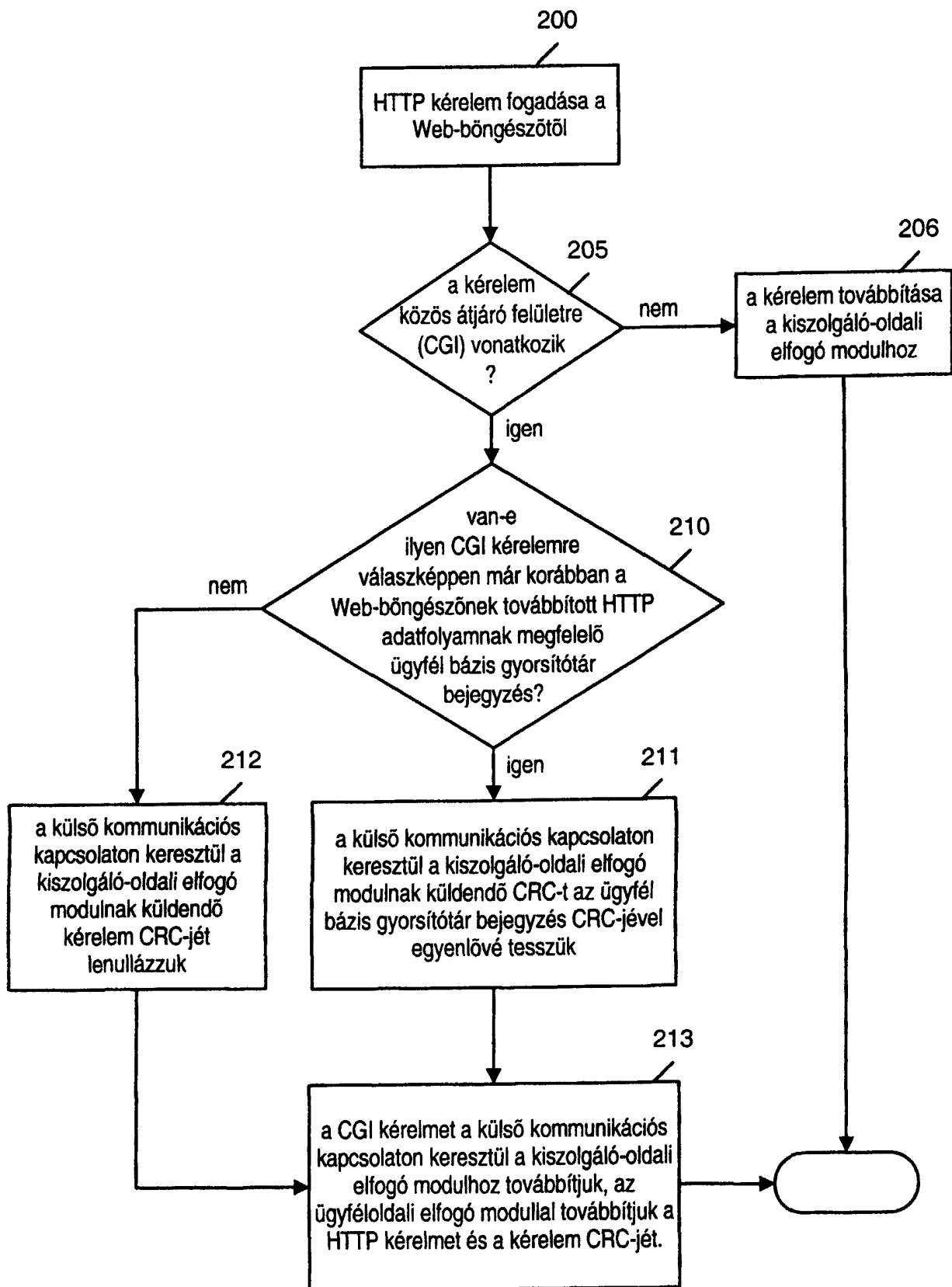
4. ábra



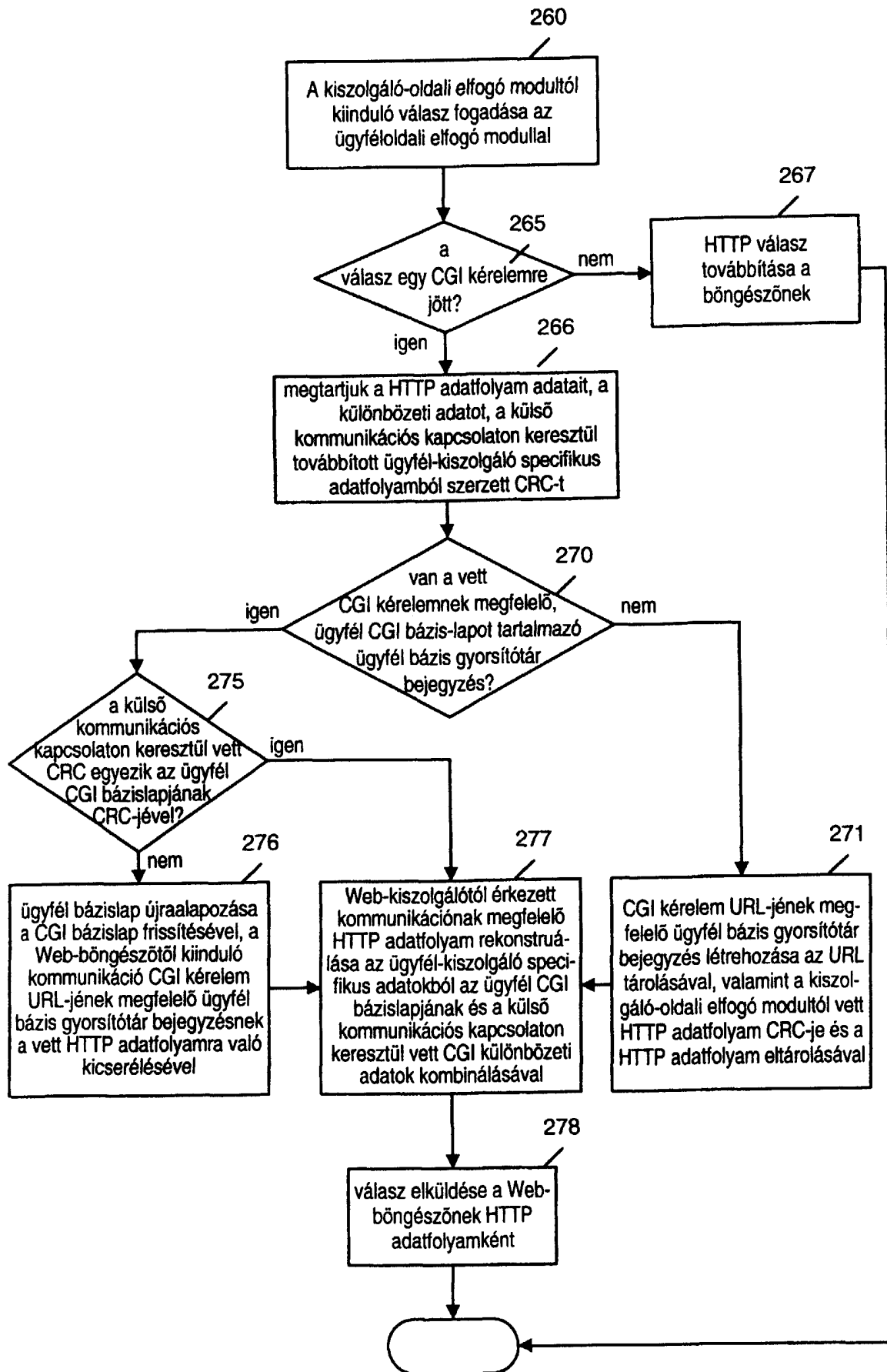
5. ábra



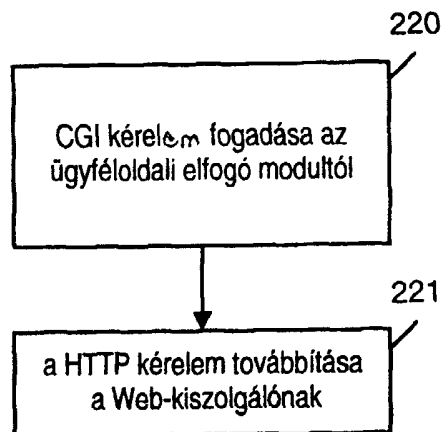
6. ábra



7. ábra



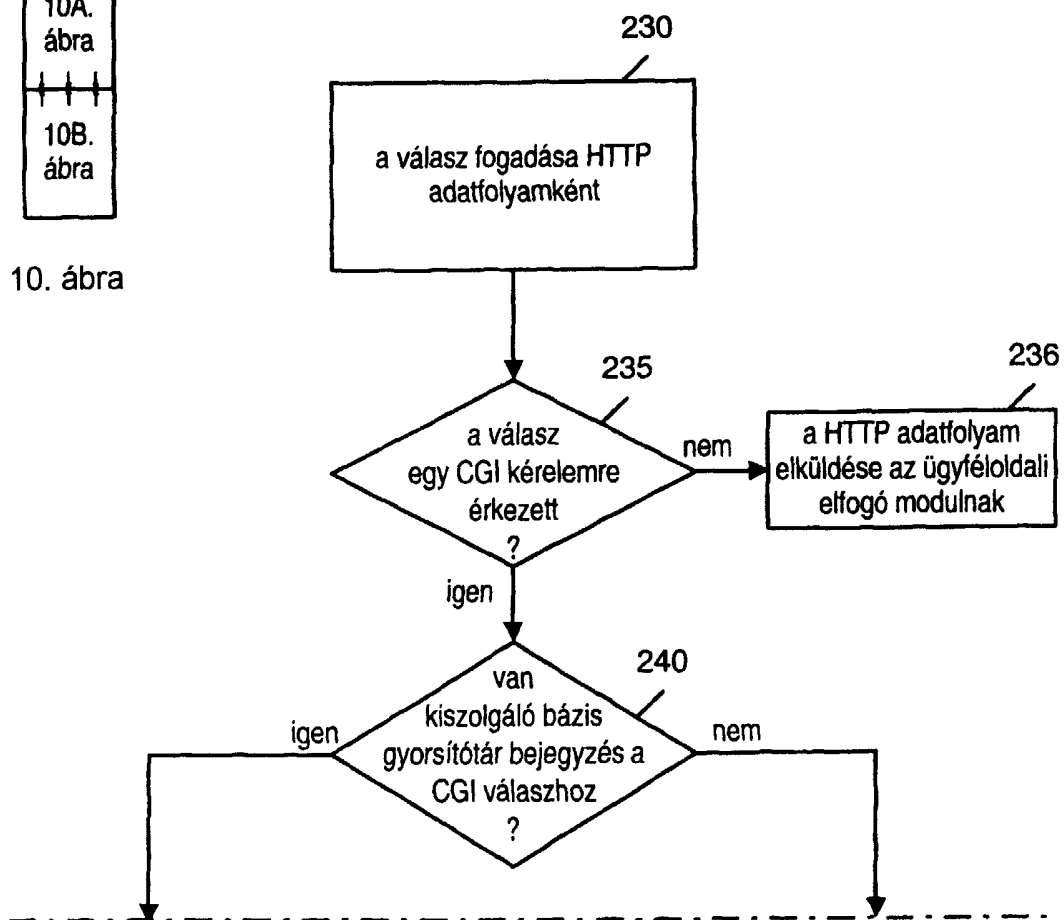
8. ábra



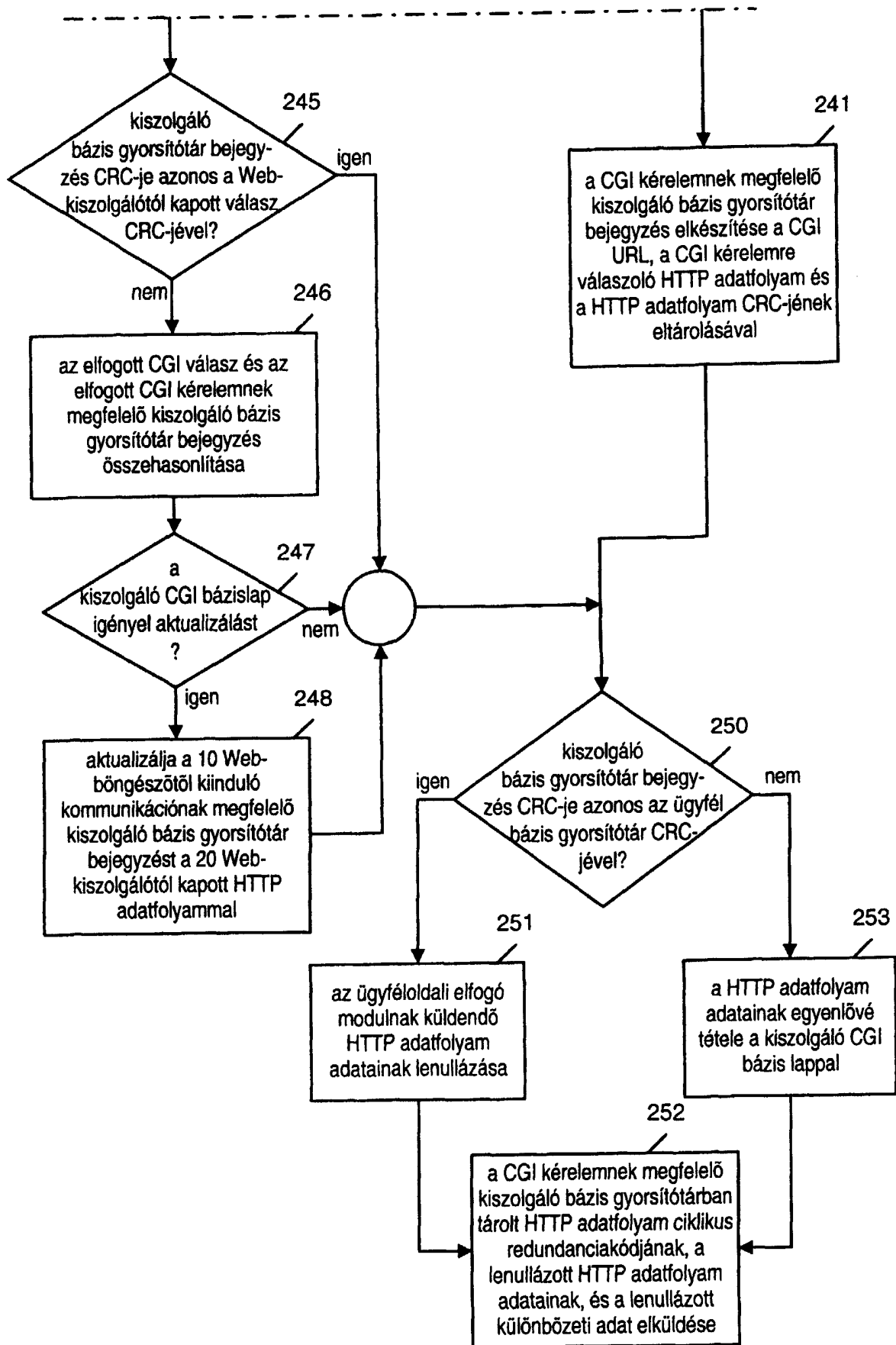
9. ábra



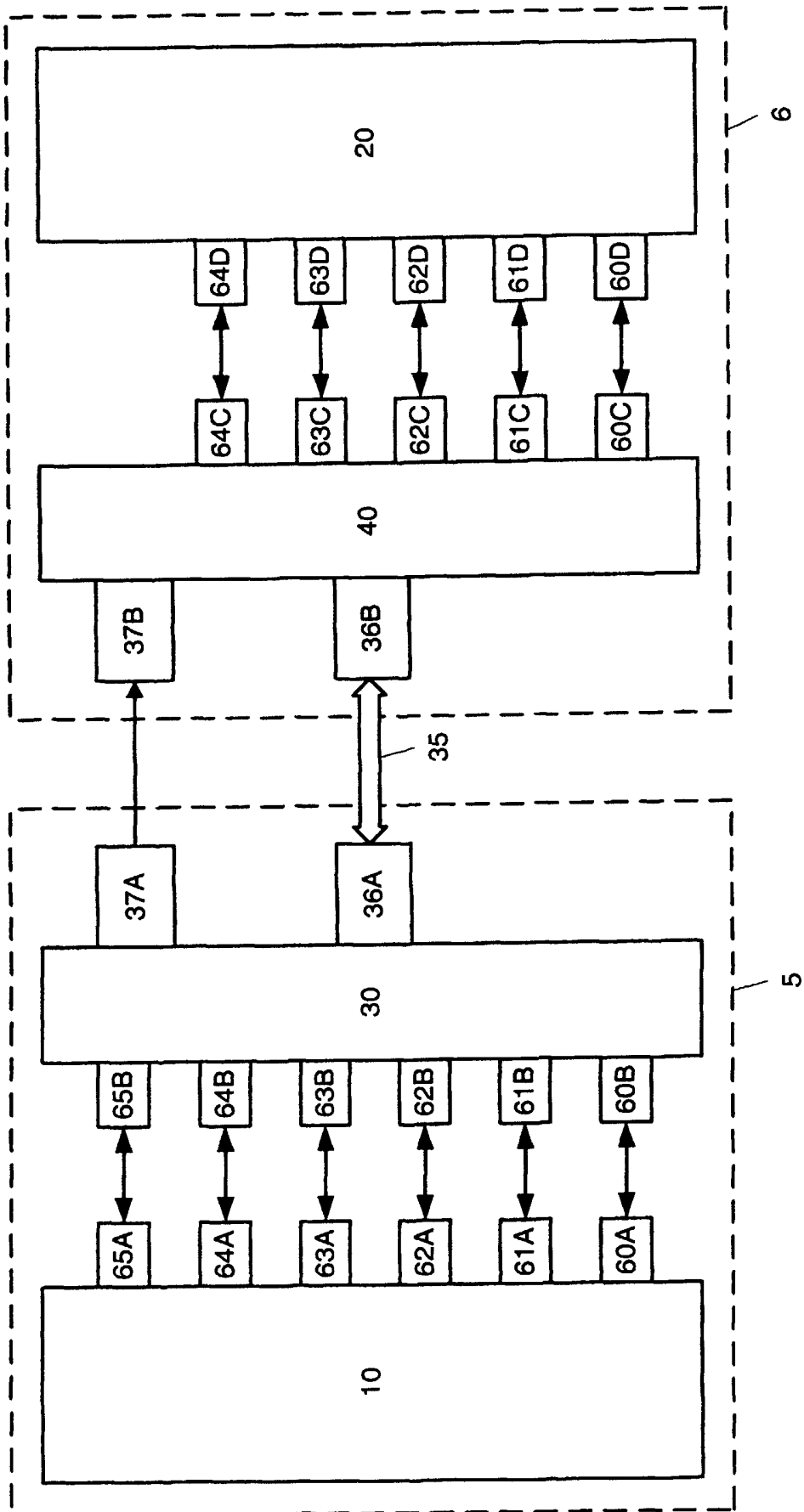
10. ábra



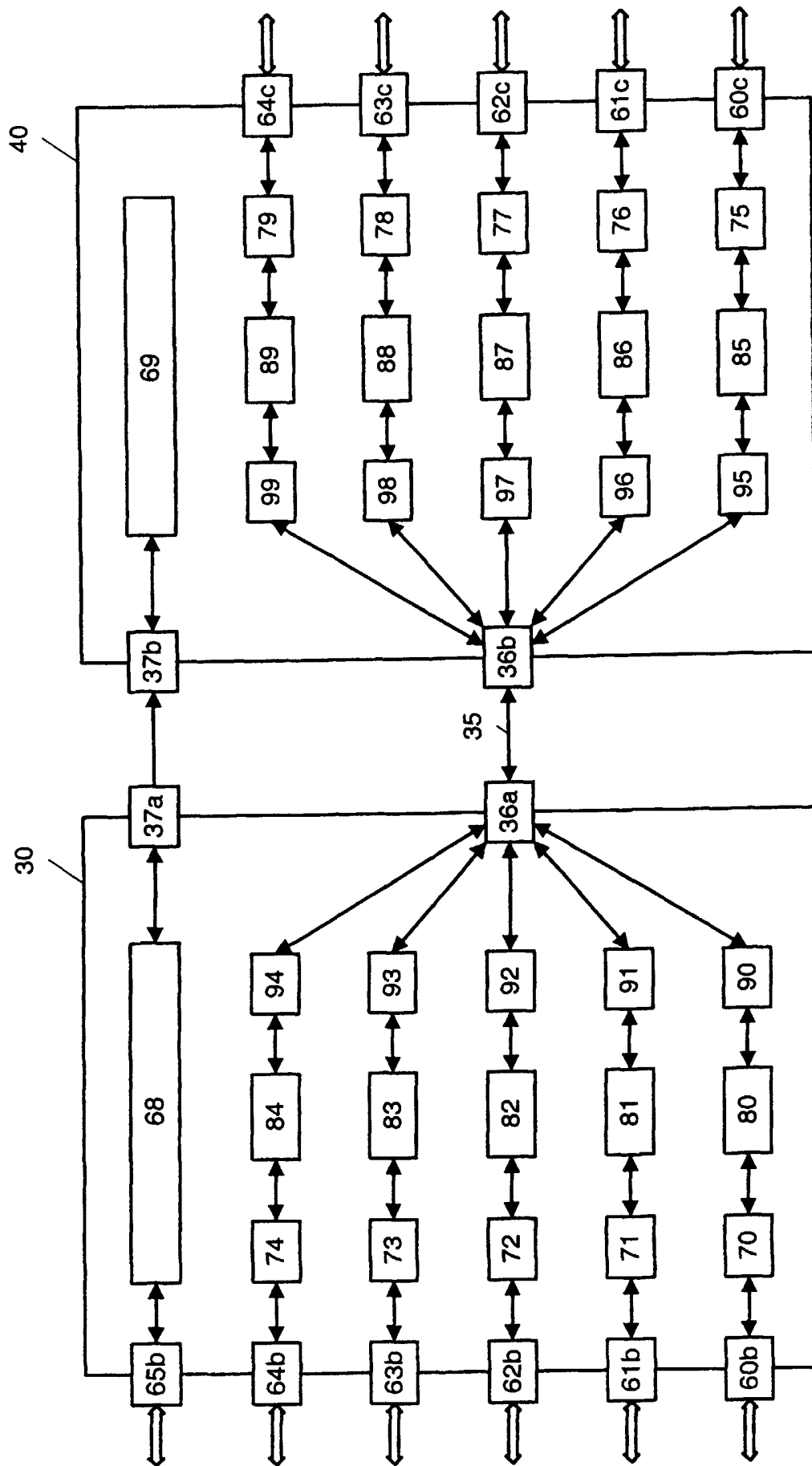
10A. ábra



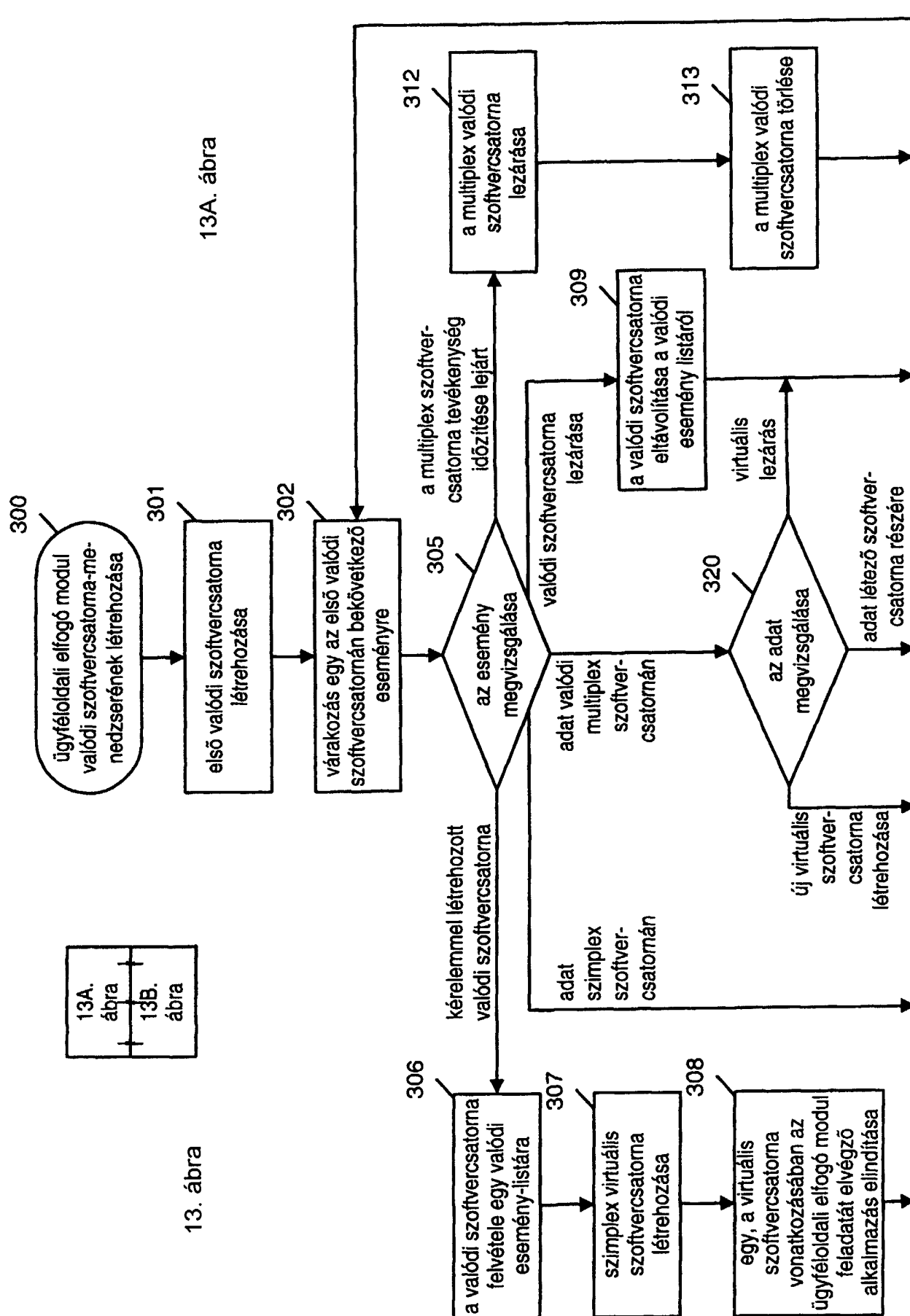
10B. ábra

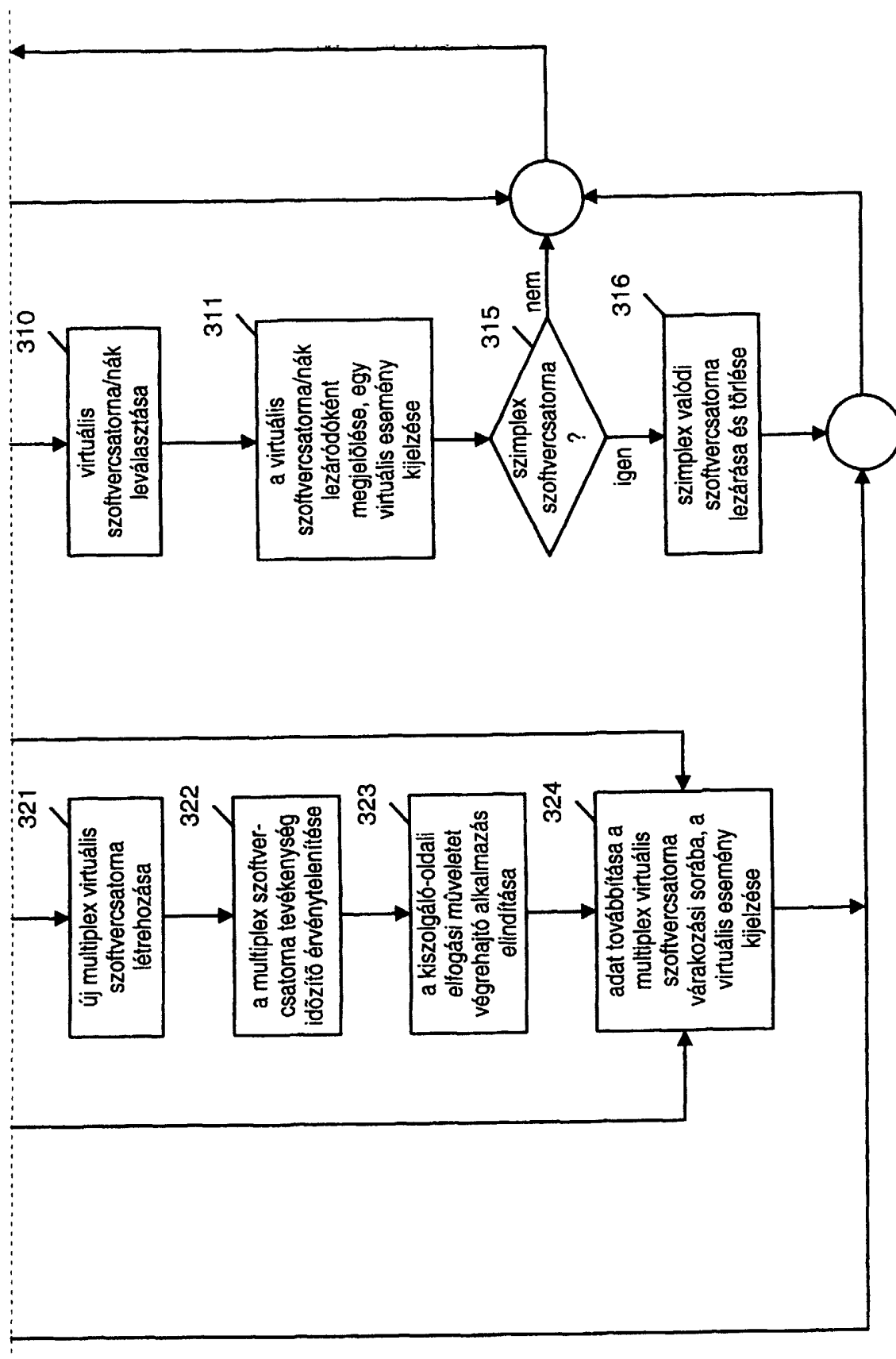


11. ábra

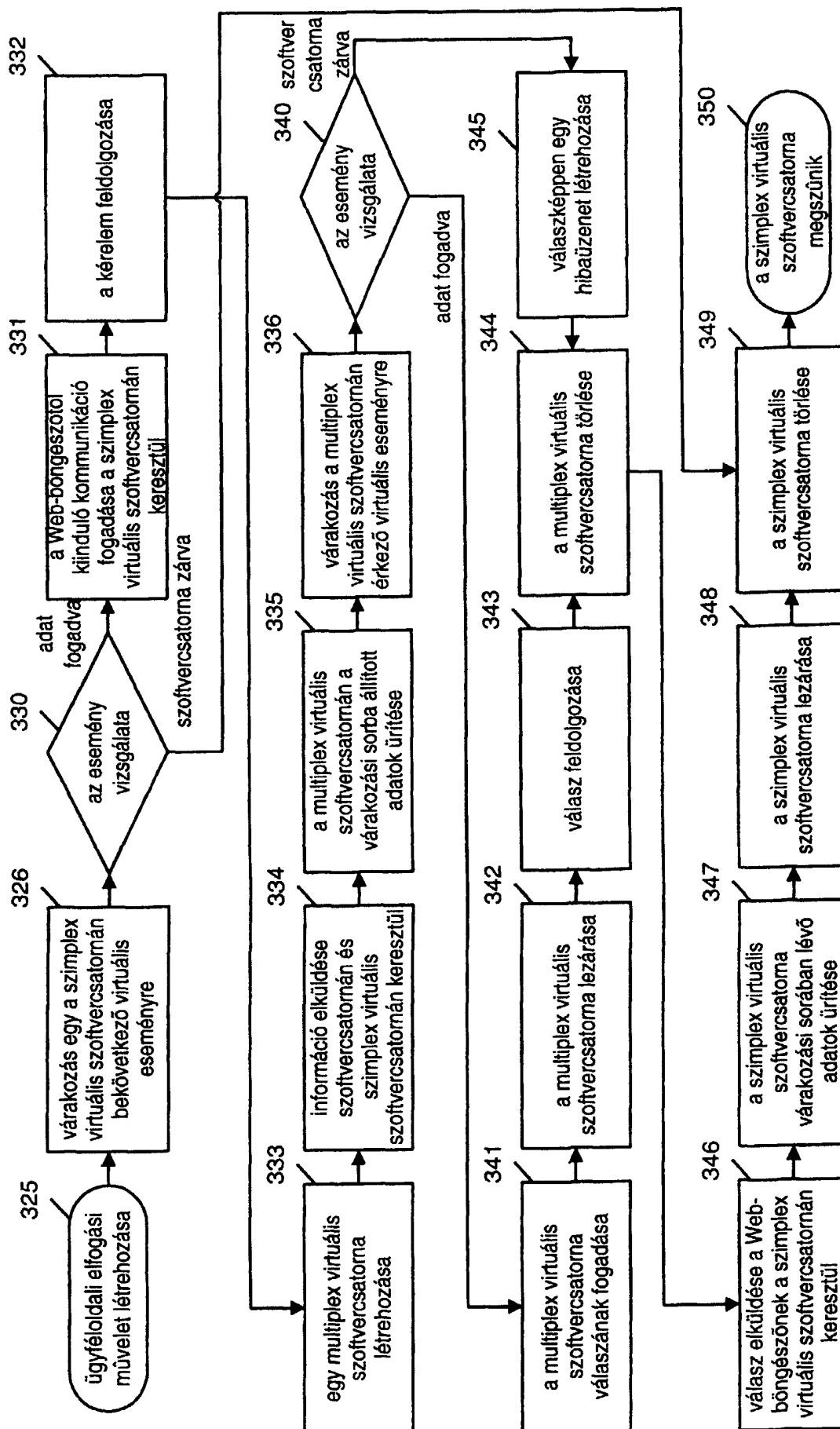


12. ábra

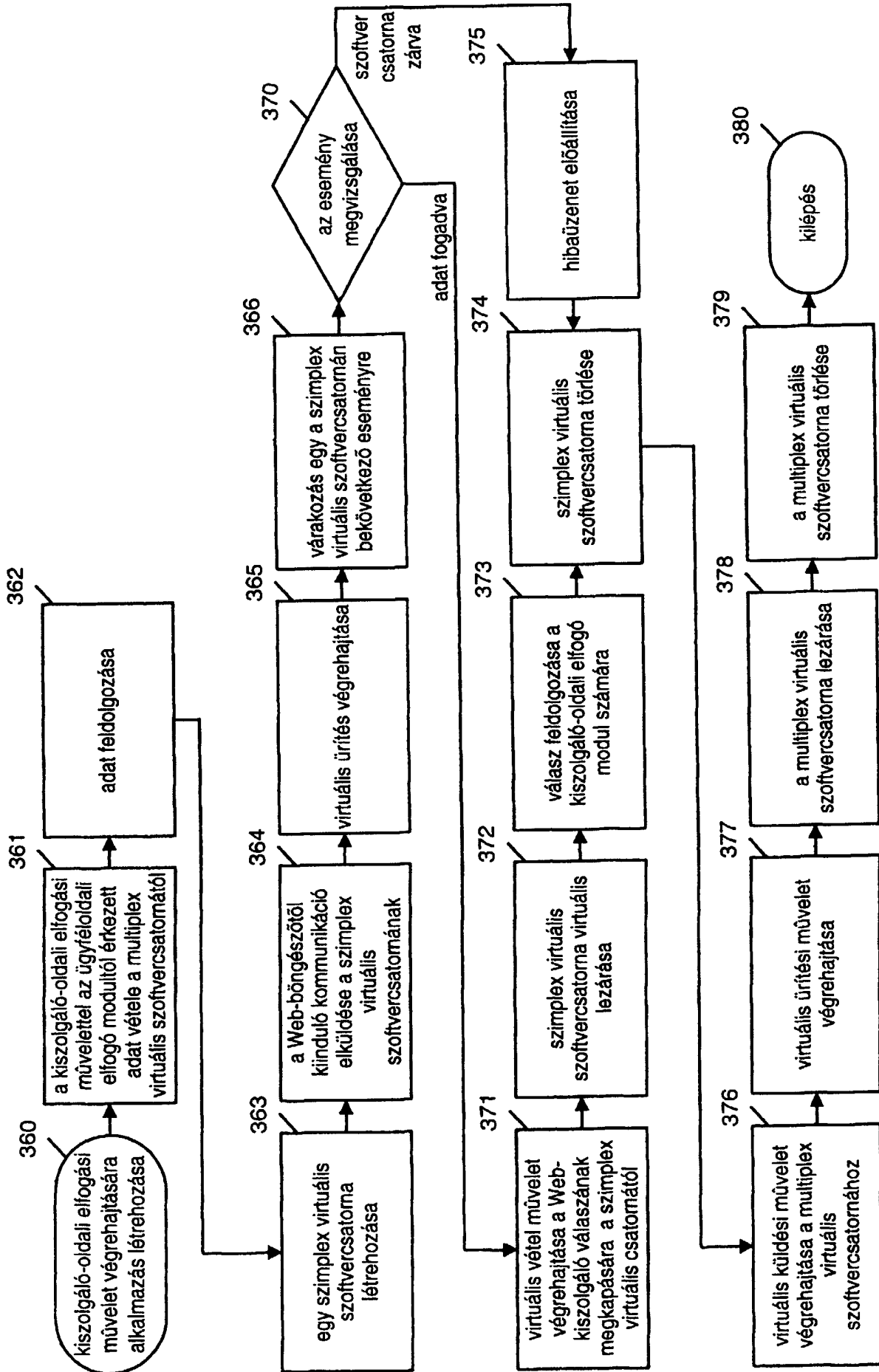




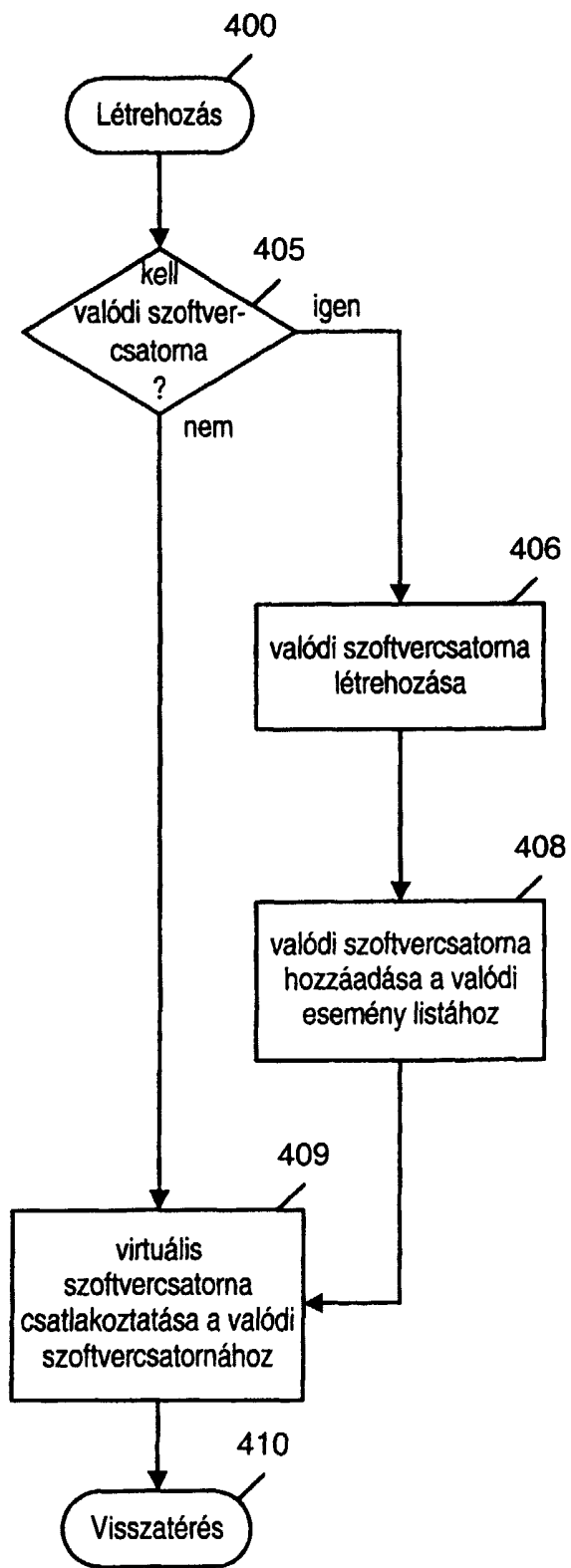
13B. ábra



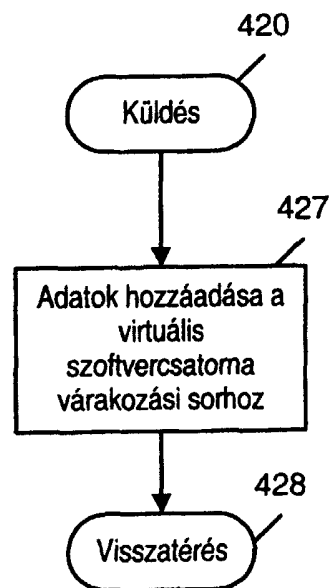
14. ábra



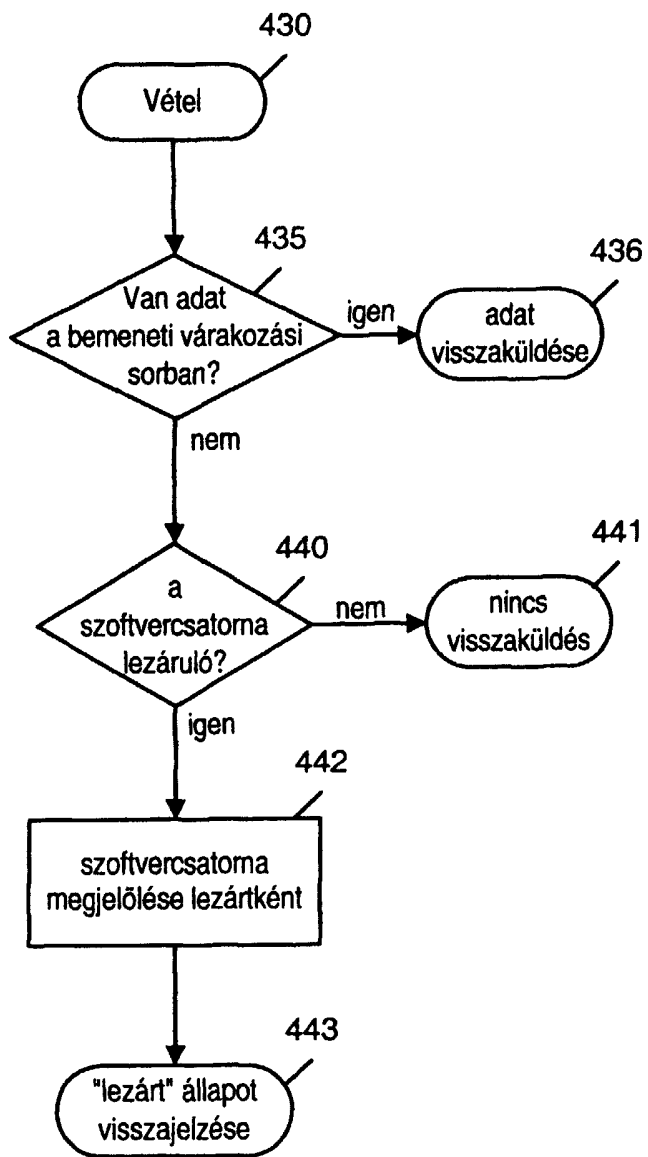
15. ábra



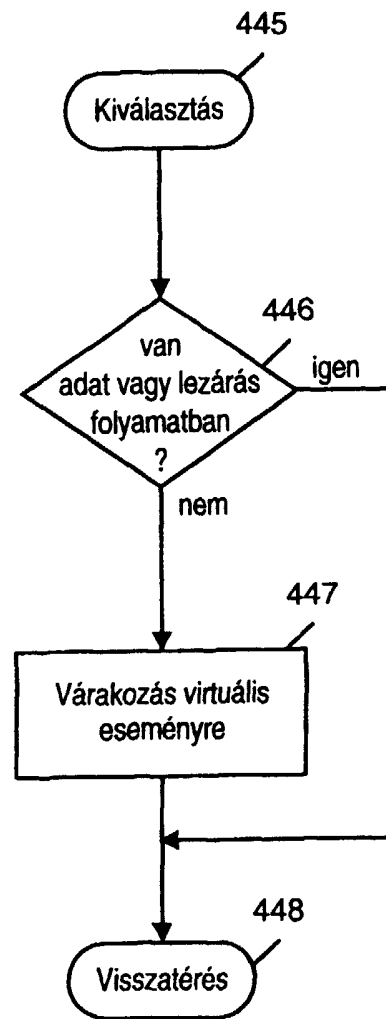
16-1. ábra



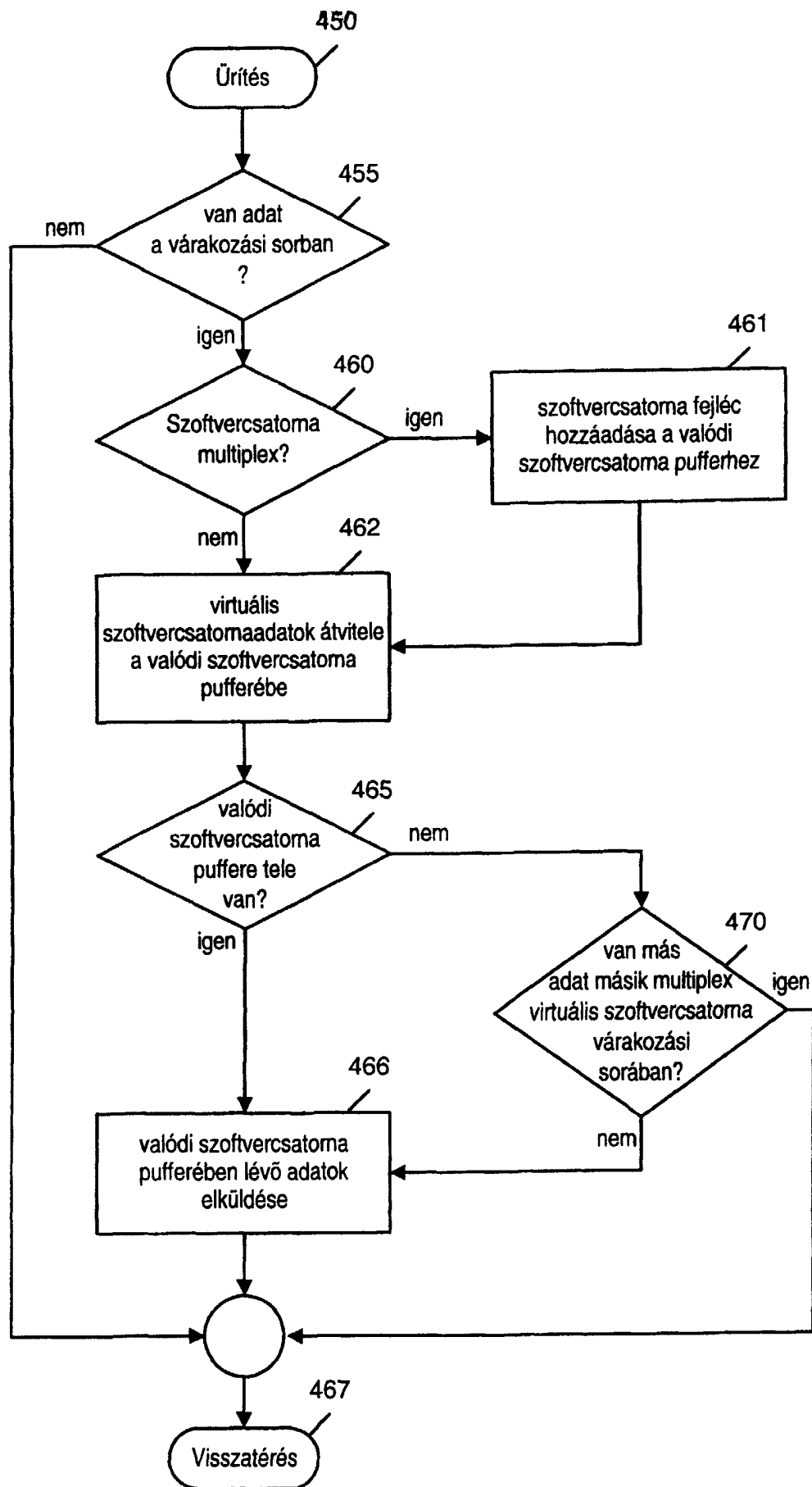
16-2. ábra



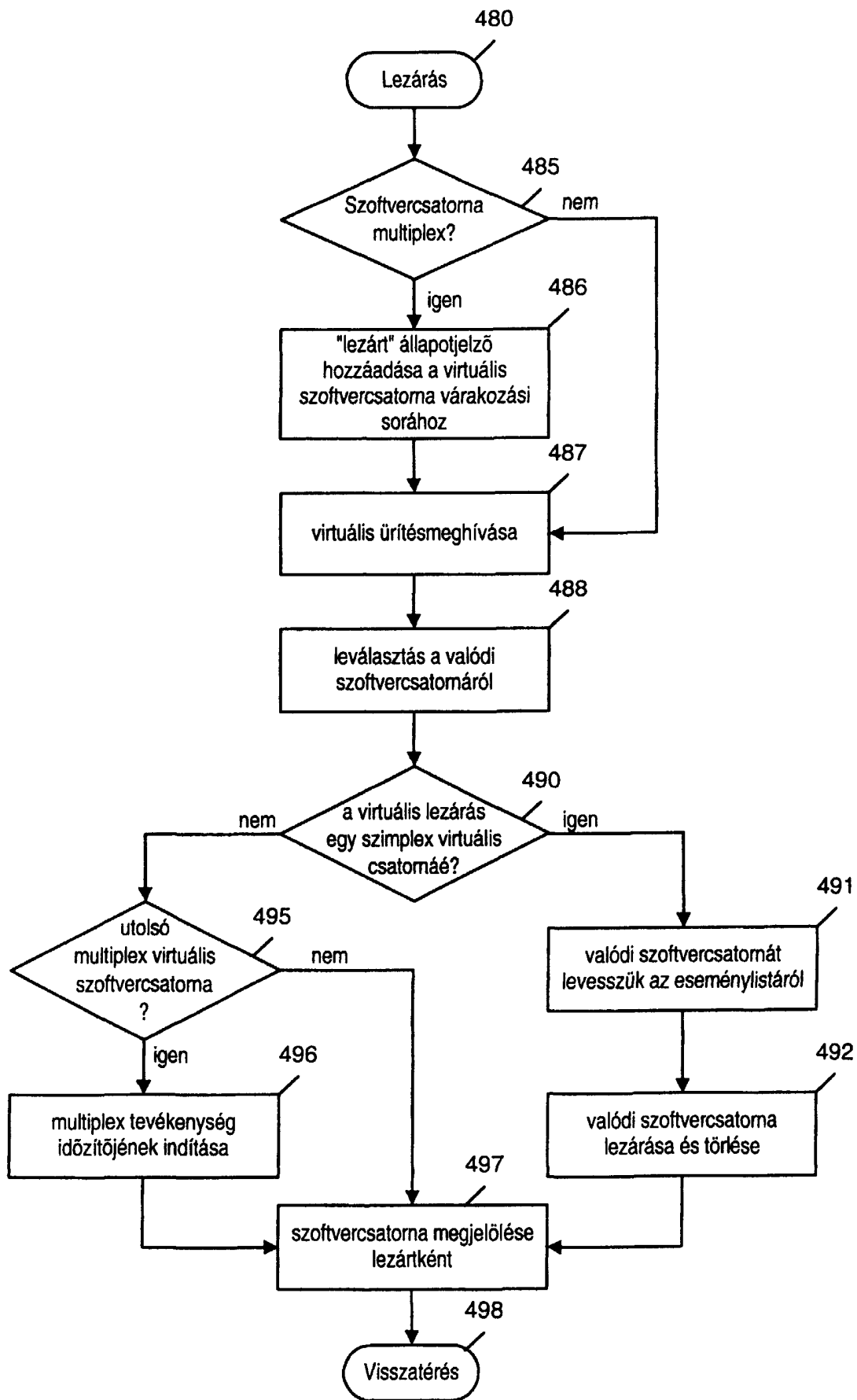
16-3. ábra



16-4. ábra



17-1. ábra



17-2. ábra