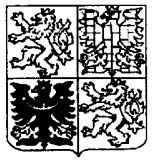
PATENTOVÝ SPIS

(11) Číslo dokumentu:

287 679

(19) ČESKÁ REPUBLIKA



ÚŘAD PRŮMYSLOVÉHO VLASTNICTVÍ

(21) Číslo přihlášky: **1997 - 3541**

(22) Přihlášeno: **11.07.1996**

(30) Právo přednosti:
15.02.1996 US 1996/601753

(40) Zveřejněno: **13.09.2000**
(Věstník č. 9/2000)

(47) Uděleno: **14.11.2000**

(24) Oznámeno udělení ve Věstníku: **17.01.2001**
(Věstník č. 1/2001)

(86) PCT číslo: **PCT/US96/11552**

(87) PCT číslo zveřejnění: **WO 97/30403**

(13) Druh dokumentu: **B6**

(51) Int. Cl.[7]:
G 06 F 17/30

(73) Majitel patentu:
INTERNATIONAL BUSINESS MACHINES CORPORATION, Armonk, NY, US;

(72) Původce vynálezu:
Bittinger Reed Richard, Raleigh, NC, US;
Fraenkel Michael Levi, Raleigh, NC, US;
Housel Barron Cornelius, Chapel Hill, NC, US;
Lindquist David Bruce, Raleigh, NC, US;

(74) Zástupce:
Kalenský Petr JUDr., Hálkova 2, Praha 2, 12000;

(54) Název vynálezu:

Způsob zachycování dat přijatých od druhé aplikace, zařízení a počítačový programový produkt k jeho provádění

(57) Anotace:

Způsob, zařízení a počítačový programový produkt pro zachycení dat, přijatých od první aplikace, které mají být poskytnuty druhé aplikaci, v odpovědi na požadavek od této druhé aplikace. Způsob, zařízení a počítačové programové produkty zahrnují uložení datového toku, který má být přijat od první aplikace a který má být poskytnut druhé aplikaci, do cache paměti s cílem vytvořit klientův cache zápis, odpovídající požadavku od druhé aplikace. Čas vytvoření klientova cache zápisu je rovněž uložen tak, aby se vytvořil časový záznam klientova cache zápisu. Požadavky od druhé aplikace jsou zachycovány za účelem stanovit, zda existuje klientův cache zápis, odpovídající danému požadavku. Časový záznam klientova cache zápisu pro klientův cache zápis, odpovídající požadavku od druhé aplikace, je vyhodnocen s cílem stanovit, zda byl vytvořen klientův cache zápis, odpovídající požadavku od druhé aplikace, během předem definovaného koherentního časového intervalu před vyžádáním informací druhou aplikací. Klientův cache zápis je předán druhé aplikaci v odpovědi na požadavek v případě, že klientův cache zápis pro daný požadavek od druhé aplikace byl vytvořen během předem definovaného klientova koherentního časového intervalu před vyžádáním informací druhou aplikací.

CZ 287679 B6

Způsob zachycování dat přijatých od druhé aplikace, zařízení a počítačový programový produkt k jeho provádění

Oblast techniky

Předložený vynález se týká způsobu zachycování dat, který je vhodný zejména u systémů s komunikacemi po pomalém nebo bezdrátovém komunikačním spoji mezi dvěma počítači, kde jeden pracuje s aplikací typu klient a druhý s aplikací typu server. Vynález se dále týká zařízení a počítačového programového produktu k jeho provádění.

Dosavadní stav techniky

Nedávná publikace a důraz na „informační superdálnici" měly za následek zvýšení povědomí o internetu a jeho přijetí jako masového komunikačního média. Toto široce založené poznání možností internetu jako životaschopného média pro komunikaci a interakci mezi mnoha různými sítěmi rovněž vytvořilo rozsáhlou uživatelskou bázi vybudovanou na internetových standardizačních protokolech pro interakci mezi počítačovými sítěmi.

Typické pro internet je to, že je zde vztah typu klient–sever, kde internetový klient (browser) komunikuje s internetovým serverem. V zájmu dosažení lepšího přístupu na internet jsou postupně komunikační protokoly a jazyky používané klienty a servery standardizovány. Tyto protokoly zahrnují i Hyper–Text Transfer Protocol (HTTP), který je komunikačním protokolem používaným pro komunikaci mezi klienty a servery, a dále Transfer Control Protocol/Internet Protocol (TCP/IP), kde část protokolu TCP představuje přenosově specifický protokol pro komunikaci mezi počítači nebo aplikacemi. Standardizovaný je rovněž jazyk, v kterém klienti a servery vzájemně komunikují a který se nazývá Hyper–Text Markup Language (HTML). Protože jsou tyto protokoly a jazyk strojově nezávislé a protože používají k odesílání informací protokol s minimalizací propojení a nejlepším výkonem, je každá transakce plně samostatná. To znamená, například, že každé hlášení od klienta obsahuje informace o schopnostech browseru (prohlížeče) a je nezávisle z pohledu komunikace, která má být dokončena, na jakýchkoliv dalších komunikacích. Tato samostatnost komunikace mezi klientem a serverem může být označována za „nestavovou" komunikaci, která zvyšuje u dané komunikace množství dat, jež musí být přenášeno mezi klientem a serverem.

V kontextu World Wide Web aplikací typu klient/server může být klientem web browser, který vytváří uživatelská rozhraní. Tento browser odesílá požadavky uživatele na příslušný web server a formátuje a zobrazuje data HTML vracená z web serveru. Web browser rovněž vyhodnocuje tato data HTML s cílem určit, zda se zde nachází mnoho vložených hyper–link příkazů, které by vyžadovaly následné požadavky browseru, jež by pak byly browserem iniciovány. Web server pracuje jako server pro klienta a zpracovává požadavky web browserů a odpovídá jim na požadavky ve formě části dat HTML v datovém toku HTTP.

Příkladem typické komunikace world wide web je příklad, kdy web browser iniciuje požadavek na „home page" (domovskou stránku) z web serveru, neboť to ilustruje základní vztah mezi HTTP, HTML, TCP a web browserem a serverem. Když uživatel web browseru požaduje informace ze specifického místa web, iniciuje web browser komunikaci s web serverem odesláním požadavku typu „získat" na web server se specifikací Universal Resource Lacator (URL) požadovaného místa web, např. na již zmíněnou domovskou stránku. URL přestavuje adresu místa web a je jedinečný v celé síti internet. Web server by potom získal a poslal web browseru data HTML odpovídající domovské stránce specifikované URL. Tato operace může zahrnout další komunikace na internetu ze strany web serveru internetu, anebo URL může specifikovat server v lokální síti, k němuž je browser připojen. Web browser by potom vyhodnotil přijatá data HTML jako datový tok HTTP od web serveru, aby zjistil, zda se zde

nevyskytují vložené spoje typu hyper–link, jako jsou ikony a obrázky, a pokud se vyskytují, iniciuje požadavky specifikující pro URL hyper–link, jak získat potřebná data. Takto obdržená data jsou pak začleněna do domovské stránky a zobrazena uživateli. Jak je z tohoto příkladu zřejmé, jednoduchý uživatelský vstupní požadavek od web browseru může mít za následek mnohonásobné dodatečné požadavky, které provádí web browser automaticky v reakci na přijatá data HTML odpovídající vstupnímu požadavku uživatele.

Popularita web browseru/web serveru a jejich společných informačních a přenosových protokolů, HTML a HTTP, vedla k rychlému přijetí technologie web jako univerzálního stykového rozhraní pro přístup k informacím na síti. Dále v důsledku standardizace protokolů a jazyka pro komunikaci mezi web browsery a web servery budou komunikační protokoly a jazyk stejné nezávisle na tom, zda uživatel pracuje s programem Netscape Navigator, NCSA Mosaic, WebExplorer nebo jakýmkoli jiným web browserem, který mu umožňuje přístup k síťovým informacím. To znamená, že velká instalovaná uživatelská báze pro web browsery zkombinovaná s připojením na Internet a snadnost zapisování na aplikační web servery pomocí definovaného HTTP rozhraní Common Gateway Interface (CGI) vytváří z technologie web velmi atraktivní prostředek pro velkou třídu aplikací založených na formulářích.

V době, kdy rostla popularita internetu a rozšiřovala se jeho dostupnost, rostla rovněž i popularita mobilního výpočetnictví. Používání laptopů, notebooků, prostředků Personal Digital/ Communication Assistants (PDA/PCA) a dalších přenosných zařízení vedlo ke zvýšení požadavků na bezdrátovou komunikaci.

Dálkové bezdrátové komunikační sítě, celulární komunikace a paketové radiospojení však trpí společnými omezeními v situacích, kdy se používají v kontextu web. Vysoké náklady na komunikaci při přepočtení na jeden byte, pomalá časová odezva, malá šířka pásma a nespolehlivost – to vše brání použití bezdrátové technologie pro nestavový komunikační protokol World Wide Web. Protože je protokol web nestavový, je i množství dat na jeden požadavek a také počet požadavků přenášených po bezdrátových spojeních větší, než by bylo nezbytné, kdyby komunikace nebyla samostatná. Takže kombinace bezdrátové technologie nebo jiné pomalé komunikační technologie s technologií web se zdá být nepraktická, protože síla technologie web je v její univerzální povaze, a ta obnažuje slabosti bezdrátové technologie.

Kromě nevýhod při používání bezdrátové komunikace v prostředí web klient/server nabízejí tradiční techniky zachycování pouze omezené použití. Obnova zachycených dat, přijatých z web serveru, znamená spolehlivost a znalost obtíží při použití ve web prostředí. Jsou–li například zachycená data obnovována pokaždé při iniciaci nové instance web browseru, potom může být generováno více zbytečných požadavků browseru v případě, že tato informace není využita během konkrétní instance web browseru. Jestliže zachycená data nejsou obnovena, potom nejsou tato data v cache paměti spolehlivá.

Podstata vynálezu

Ve světle výše uvedených omezení je jedním z cílů tohoto vynálezu poskytnout systém zachycování, který zajistí spolehlivá data, bez nutnosti nadbytečných aktualizací zachycovaných dat.

Dalším cílem stávajícího vynálezu je poskytnout strukturu zachycování, která může být použita v prostředí web browser/server.

Cílem tohoto vynálezu je také zachovat kompatibilitu s existujícími komunikačními protokoly a jazyky u pomalých nebo bezdrátových komunikačních systémů, bez nutnosti úpravy aplikací web browseru nebo web serveru.

K dalším cílům stávajícího vynálezu patří poskytnutí způsobu zachycování, který redukuje objem komunikace, požadované mezi web browserem a web serverem, a tedy zvýšení výkonnosti komunikačního systému.

Z pohledu těchto a dalších cílů poskytuje stávající vynález způsob zachycování dat, přijatých od druhé aplikace, a také poskytnutých na první aplikaci, v odpovědi na požadavek od první aplikace. Způsob zahrnuje uložení datového toku, který má být přijat z druhé aplikace a který má být poskytnut první aplikaci, do první cache paměti a cílem vytvořit klientův cache zápis, odpovídající požadavku z první aplikace. Čas vytvoření klientova cache zápisu je rovněž uložen tak, aby se vytvořil záznam času klientova cache zápisu. Požadavky z první aplikace jsou zachycovány s cílem stanovit, zda existuje klientův cache zápis, odpovídající požadavku. Pokud existuje klientův cache zápis, který odpovídá požadavku, potom je vyhodnocen záznam času klientova cache zápisu pro klientův cache zápis odpovídající požadavku první aplikace tak, aby se určilo, zda byl vytvořen klientův cache zápis, odpovídající požadavku první aplikace, během předem definovaného klientova koherentního časového intervalu před tím, než první aplikace požadovala informace. Klientův cache zápis je předán na první aplikaci v odpovědi na dotaz, zda byl vytvořen klientův cache zápis během předem definovaného klientova koherentního časového intervalu před tím, než první aplikace vyžádala informace. V dalším začlenění stávajícího vynálezu jsou udržovány klientovy cache zápisy pro více instancí první aplikace.

V alternativním začlenění stávajícího vynálezu je první aplikace rezidentní na prvním počítači a druhá aplikace rezidentní na druhém počítači, přičemž první a druhá aplikace spolu navzájem komunikují přes externí komunikační spoj. V tomto alternativním začlenění stávající vynález dále zahrnuje uložení datového toku od druhé aplikace v odpovědi na požadavek od první aplikace do druhé cache paměti rezidentní na druhém počítači tak, aby se vytvořil cache zápis požadavku serveru. Požadavek vytvořený první aplikací je dále vyhodnocen s cílem stanovit, zda bal cache zápis požadavku serveru, odpovídající tomuto požadavku, dříve uložen v druhé cache paměti. Cache zápis požadavku serveru, odpovídající požadavku od první aplikace, je potom odeslán na první aplikaci rezidentní na prvním počítači přes externí komunikační spoj.

V jiném alternativním začlenění stávajícího vynálezu určí druhý počítač, zda cache zápis požadavku serveru, odpovídající požadavku od první aplikace, byl vytvořen během předem definovaného klientova časového intervalu před tím, než druhý počítač přijal požadavek. Cache zápis požadavku serveru, který odpovídá požadavku od první aplikace, je odeslán na první počítač v případě, že tento cache zápis požadavku serveru byl vytvořen během předem definovaného klientova koherentního časového intervalu.

Alternativní začlenění stávajícího vynálezu může rovněž zahrnovat určení, zda existuje klientův cache zápis, odpovídající požadavku od první aplikace, který je identický se serverovým cache zápisem požadavku. Časový interval mezi tím, kdy druhý počítač přijal požadavek od první aplikace,a tím, kdy byl vytvořen cache zápis požadavku serveru, odpovídající požadavku od první aplikace, je vypočten s cílem poskytnout data o stáří zápisu. Data o stáří zápisu pro serverový cache zápis, odpovídající požadavku od první aplikace, jsou přenesena n první aplikaci rezidentní na prvním počítači, a záznam času klientova cache zápisu, odpovídajícího požadavku od první aplikace, je aktualizován odečtením dat o stáří zápisu, přijatých z druhého počítače, od aktuálního času na prvním počítači

V jiných začleněních stávajícího vynálezu tvoří první aplikaci web browser a druhou aplikaci web server, a v dalším začlenění komunikují první a druhý počítač přes bezdrátový komunikační spoj.

Výše popsané aspekty stávajícího vynálezu mohou být podle potřeb poskytnuty rovněž i v jiné formě, např. jako počítačový program (tj. soubor, který lze číst na počítači).

Přehled obrázků na výkresech

Vynález je dále blíže objasněn na příkladech provedení pomocí výkresů.

Obr. 1 znázorňuje blokový diagram typického systému web browser/web server.

Obr. 2 znázorňuje blokový diagram systému web browser/web server podle jednoho provedení předloženého vynálezu, využívajícího zachycení klienta a zachycení serveru.

Obr. 2a znázorňuje blokový diagram systému web browser/web server podle jednoho provedení předloženého vynálezu, využívajícího klientovou první cache pamět a serverovou druhou cache paměť.

Obr. 3 znázorňuje vývojový diagram operací prováděných zachycovacím modulem na straně klienta v přednostním provedení předloženého vynálezu implementujícího koherentní vyrovnávací systém cache.

Obr. 4 názorňuje vývojový diagram operací prováděných zachycovacím modulem na straně klienta v přednostním provedení předloženého vynálezu implementujícího kohorentního vyrovnávací systém cache.

Obr. 5 znázorňuje vývojový diagram operací prováděných zachycovacím modulem na straně serveru v přednostním provedení předloženého vynálezu implementujícího koherentní vyrovnávací systém cache.

Obr. 6 znázorňuje vývojový diagram operací prováděných zachycovacím modulem na straně serveru v přednostním provedení předloženého vynálezu implementujícího koherentní vyrovnávací systém cache.

Obr. 7 znázorňuje vývojový diagram operací prováděných zachycovacím modulem na straně klienta v přednostním provedení předloženého vynálezu implementujícího přenosový systém rozdílových dat.

Obr. 8 znázorňuje vývojový diagram operací prováděných zachycovacím modulem na straně klienta v přednostním provedení předloženého vynálezu implementujícího přenosový systém rozdílových dat.

Obr. 9 znázorňuje vývojový diagram operací prováděných zachycovacím modulem na straně serveru v přednostním provedení předloženého vynálezu implementujícího přenosový systém rozdílových dat.

Obr. 10 znázorňuje vývojový diagram operací prováděných zachycovacím modulem na straně serveru v přednostním provedení předloženého vynálezu implementujícího přenosový systém rozdílových dat.

Obr. 11 znázorňuje blokový diagram jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 12 znázorňuje blokový diagram zachycovacího modulu na straně klienta a zachycovacího modulu na straně serveru podle jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 13 znázorňuje vývojový diagram operace prováděné manažerem soketů u zachycovacího modulu na straně klienta, anebo zachycovacího modulu na straně serveru, podle jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 14 znázorňuje vývojový diagram popisující operace prováděné zachycovací funkcí na straně klienta v jednom provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 15 znázorňuje vývojový diagram popisující operace prováděné zachycení funkcí na straně serveru v jednom provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 16–1 znázorňuje vývojový diagram popisující virtuální operaci vytváření podle jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 16–2 znázorňuje vývojový diagram popisující virtuální operaci odesílání podle jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 16–3 znázorňuje vývojový diagram popisující virtuální operaci příjmu podle jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 16–4 znázorňuje vývojový diagram popisující virtuální operaci výběru podle jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 17–1 znázorňuje vývojový diagram popisující virtuální operaci vyrovnání podle jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Obr. 17–2 znázorňuje vývojový diagram popisující virtuální operaci zavření podle jednoho provedení předloženého vynálezu, využívajícího virtuální sokety.

Příklady provedení vynálezu

Stávající vynález bude nyní popsán podrobněji, a to s pomocí příslušných obrázků, v nichž jsou znázorněny preferované popisy vynálezu. tento vynález může být však začleněn v mnoha různých formách a neměl by být konstruován jako limitovaný na zde uváděná začlenění. Toto začlenění jsou uváděna tak, aby jejich popis byl úplný a kompletní a aby byl plně předkládaný vynález prezentován těm, kdo jej popisují. V celém dokumentu jednotlivé elementy označují čísla.

Základní komunikační struktura pro systém založený na síti internet je znázorněna na obr. 1. Na obrázku 1 komunikuje web browser 10 s web serverem 20 po komunikačním spoji 15. Tento komunikační spoj je obvykle spoj LAN, WAN s využitím telefonních linek nebo kombinace propojovacích metod. Web browser 10 komunikuje s web serverem 20 pomocí TPC/IP. Většina komunikací po Internetu je taková, že web web browser komunikuje s web serverem prostřednictvím generického komunikačního protokolu HTTP, který je přenášen mezi web browserem a web serverem po spoji TCP/IP vytvořeném mezi web browserem a web serverem. Skutečná data přenášena mezi web browserem 10 a web serverem 20 jsou tvořena objekty HTTP (např. data HTML), jak bylo výše popsáno. Web server 20 může být i zprostředkovatelem, který přijímá web browserové komunikace od více web browserů a routerů (směrovačů) a předává je na příslušný server.

Obr. 3 až 10 a 13 až 17–2 jsou vývojové diagramy metod a systémů podle vynálezu. Mělo by být jasné, že každý blok vývojových diagramů a také kombinace těchto bloků mohou být implementovány pomocí instrukcí počítačového programu. Tyto instrukce počítačového

programu mohou být zaváděny na počítač nebo mohou být tvořeny jinými programovatelnými zařízeními s cílem zhotovit stroj tak, aby instrukce, které bude počítač nebo jiné programovatelné zařízení provádět, vytvořily prostředky pro implementování funkcí specifikovaných v bloku nebo v blocích vývojového diagramu. Tyto počítačové programové instrukce mohou být uloženy v počítačem čitelné paměti, která může řídit počítač nebo jiné programovatelné zařízení, aby pracovalo určitým způsobem, tak jako instrukce uložené v počítačem čitelné paměti vytvářejí výrobní článek včetně instrukčních prostředků, které implementují funkci specifikovanou v bloku nebo v blocích vývojového diagramu. Počítačové programové instrukce mohou být rovněž zaváděny do počítače nebo jiného programovatelného zařízení tak, že způsobí řadu funkčních kroků prováděných na počítači nebo jiném programovatelném zařízení s cílem vytvořit počítačem implementovaný proces, tak jako instrukce, které provádějí na počítači nebo jiném programovatelném zařízení kroky pro implementaci funkcí specifikovaných v bloku nebo v blocích vývojového diagramu.

Podobně bloky vývojových diagramů podporují kombinace prostředků i kroků pro provádění specifických funkcí. Každý blok na ilustraci vývojového diagramu, popřípadě kombinace bloků, může být implementován speciálním jednoúčelovým hardwarově založeným počítačovým systémem, který provádí specifikované funkce nebo kroky, anebo kombinací speciálního účelového hardware a počítačových instrukcí.

Obrázek 2 ilustruje jedno začlenění stávajícího vynálezu. Jak je zřejmé z obrázku 2, web browser 10 komunikuje se zachycovacím modulem na straně klienta 30. Web server 20 komunikuje se zachycovacím modulem na serverové straně 40. Zachycovací modul na klientově straně 30 pak komunikuje se zachycovacím modulem na serverové straně 40 přes komunikační spoj 35. Web browser 10 a zachycovací modul na klientově straně 30 se mohou nacházet v prvním počítači 5. Zachycovací modul na serverové straně 40 a web server 20 se mohou nacházet ve druhém počítači 6. První počítač 5 a druhý počítač 6 spolu komunikují přes externí komunikační spoj 35.

Přednostně je web browser 10 internetový web browser využívající hypertext transfer protocol (HTTP) a hypertext markup language (HTML) pro komunikaci s internetovým web serverem 20, který rovněž používá HTTP a HTML. V provozu by web browser 10 dával na výstup datový tok HTTP, který je zachycovám modulem na straně klienta 30. Zachycení datového toku HTTP zachycovacím modulem na straně klienta 30 může být prováděno použitím funkce zpětnovazební smyčky TCP/IP, kde zachycovací modul na straně klienta 30 je rezidentní na adrese IP a má síťové číslo 127, např. 127.0.0.1. Zachycovací modul na straně klienta 30 potom konvertuje nebo transformuje datový tok HTTP do specifického protokolu klient/server a odešle specifický datový tok klient/server na externí komunikační spoj 35. Zachycovací modul na straně serveru 40 přijme specifický datový tok a rekonstruuje původní datový tok HTTP odpovídající komunikaci vytvářené web browserem. Tento rekonstruovaný datový tok HTTP je potom odeslán na web server 20. Web server 20 odpoví na datový tok HTTP obvyklým způsobem pro internetový web server. Z pohledu hodnocení vynálezu je nutné říci, že web server 20 může být rovněž nahrazen tak, aby bylo dosaženo připojení na Internet pro více browserů.

Jakmile je informace přijata web serverem 20 pro odeslání na web browser 10, například jako odpověď na požadavek browserů pro specifickou domovskou stránku URL, odešle web server 20 výstupní datový tok HTTP odpovídající komunikaci, která má být odeslána na web browser 10. Tato komunikace vytvářená web serverem je zachycena zachycovacím modulem na straně serveru 40 a transformována na specifický datový tok klient/server. Tento specifický datový tok klient/server odpovídající komunikaci vytvářené web servrem je potom odeslán na externí komunikační spoj 35 druhým počítačem do prvního počítače. Specifický datový tok klient/server je přijat prvním zachycovacím modulem na straně klienta 30, který rekonstruuje původní datový tok HTTP odpovídající komunikaci vytvářené web serverem a poskytuje jej web browseru 10.

V konkrétním začlenění stávajícího vynálezu je externí komunikační spoj 35 bezdrátovým komunikačním spojem. V takovém případě, aby bylo dosaženo systémové výkonnosti přijatelné

pro uživatele, je zapotřebí snížit objem komunikace přes externí komunikační spoj 35, a to jak co do četnosti komunikace, tak do objemu informací, které musí být přenášeny přes komunikační spoj 35. Podobně současný vynález využívá techniky vyrovnávání, vytváření rozdílů a redukce protokolů s cílem minimalizovat potřebný objem komunikace přes externí komunikační spoj 35. Tyto techniky jsou prováděny konvertováním vestavových nebo stochastických protokolů HTTP na specifický protokol klient/server, který využívá informace specifické pro klienta a pro server tak, aby minimalizoval objem a četnost komunikace.

Zatímco stávající vynález je a bude popisován s ohledem na jednu aplikaci web browser a jednu aplikaci web server je nutné předeslat pro hodnocení tohoto vynálezu, že přednosti a výhody stávajícího vynálezu mohou být rovněž dosahovány násobnými web browsery přiřazenými jednomu web serveru. Stávající vynález se tedy vztahuje i na metody, zařízení a programové produkty ve spojení s násobnými browsery, kde každý z nich komunikuje pomocí zachycovacího modulu na straně klienta a kde tyto zachycovací moduly na straně klienta komunikují se zachycovacím modulem na straně serveru u web serveru nebo jiného obvodového web zařízení.

V jednom začlenění stávajícího vynálezu jak u prvního zachycovacího modulu na straně klienta, tak u zachycovacího modulu na straně serveru 40 se využívá možnosti ukládání do vyrovnávací paměti. Klientova vyrovnávací cache paměť se nachází na prvním počítači a ukládá datové toky HTTP, které mají být přijaty web browserem v odpovědi na komunikaci vytvářenou web browserem. Serverová vyrovnávací cache paměť je rezidentní na druhém počítači a ukládá datové toky HTTP, které jsou přijaty z web serveru v odpovědi na komunikaci vytvářenou browserem.

Obr. 2a znázorňuje uložení cache paměti odpovídající zachycovacímu modulu podle provedení vynálezu. Klientova první cache paměť 31 na prvním zachycovacím modulu na straně klienta 30 je rezidentní na prvním počítači 5 a druhá cache paměti na straně serveru 40 je rezidentní na druhém počítači 6.

Z pohledu hodnocení tohoto vynálezu cache paměť rezidentní na prvním počítači nebo na druhém počítači může být libovolné velikosti a libovolného specifického hardware či konfigurace počítače. Tyto vyrovnávací cache paměti uchovávají informace pro jednotlivé komunikace včetně URL komunikace, jednoznačného identifikátoru založeného na komunikacích obsahujícího například cyklickou kontrolu redundance (CRC), dat komunikace, data a času uložení (SDT) indikujících dobu, kdy byl vytvořen nebo obnoven cache zápis, a komunikačních dat. Pro každou komunikaci ukládanou do paměti cache může být tedy vytvořen adresář cache zápisů. Tím jsou klientovy cache zápisy 32 uloženy v první cache paměti 31 na straně klienta 30 a cache zápis 42 na straně serveru 40 je uložen v druhé cache paměti 41 na straně serveru 42, jak je patrné na obr. 2a. Dále může být libovolný, v důsledku omezených zdrojů dostupných pro danou hardwarovou konfiguraci, počet technik ukládání do vyrovnávací paměti a rovněž udržování rezidentních cache pamětí na prvním i na druhém počítači. Na to vše vztahuje tento vynález. To znamená, že například cache paměť může zrušit platnost nejstaršího adresáře zápisů v případě, že uživatelem definovaná cache velikost je překročená v důsledku přidání nového zápisu. Cache zápisy mohou být dále udržovány prostřednictvím násobných entit aplikací web browseru nebo web serveru nebo dokonce cyklů spouštěných po zapnutí na prvním nebo na druhém počítači pro vytvoření stálé cache paměti.

Činnost struktury vyrovnávání je podle jednoho provedení předloženého vynálezu popsáno pomocí obr. 3 až 6, což jsou vývojové diagramy popisující činnost zachycovacího modulu na straně klienta 30 a zachycovacího modulu na straně serveru 40. Konkrétně u obr. 3 označuje blok 100, že zachycovací modul na straně klienta přijal požadavek od web browseru 10. Tento požadavek může být ve formě datového toku HTTP. Zachycovací modul na straně klienta 30 kontroluje Uniform Resource Locator (URL) příchozího požadavku, jak je znázorněno v bloku 105. Zachycovací modul na straně klienta 30 určí z URL, jestli byla informace, dříve uložená

v klientově cache rezidentní v prvním počítači 5, odpovídající požadavku vytvořeného web browserem.

Jestliže informace odpovídají URL nebyla předtím uložena v klientově cache paměti, potom se uskuteční zachycovacím modulem na straně klienta činnosti znázorněné v bloku 106. Zachycovací modul na straně klienta 30 odešle požadavek přes externí komunikační spoj 35 na zachycovací modul na straně serveru 40.

Jestliže však po prozkoumání komunikace vytvořené web browserem, jak je znázorněno v bloku 105, se zjistí, že existuje zápis klientovy cache, který odpovídá komunikaci vytvořené web browserem, potom v nejjednodušším začlenění by byla tato informace poskytnuta web browseru jako datový tok HTTP. Avšak, jak je znázorněno na obrázku 3, provádí preferované začlenění stávajícího vynálezu to, co je zde označováno jako koherentní intervalová kontrola na cache zápisu odpovídající komunikaci vytvořené web browserem. Tato operace je znázorněna v bloku 110 na obrázku 3.

Koherentní interval pro zachycovací modul na straně klienta může být uživatelem definovaný a je dán časovou délkou, po kterou může existovat cache zápis předtím, než se stane zastaralým, a musí být obnoven položením dotazu na informaci odpovídající komunikaci vytvořené web browserem, a to web serveru. Koherentní intervalová kontrola, znázorněná v bloku 110, může být prováděna porovnáním aktuálního data a času se sumou SDT cache zápisu, odpovídající komunikaci vytvořené web browserem a koherentnímu intervalu specifikovanému uživatelem. Jestliže aktuální datum a čas je větší než tato suma, potom bude informace uložená v cache paměti odpovídající komunikaci vytvářené web browserem zastaralá a uplatní se větev „No“ bloku 110. Jestliže však aktuální datum a čas je menší než suma SDT plus uživatelem definovaný koherentní interval, potom platí větev „Yes“ bloku 110 a jak je dále znázorněno v bloku 111, je cache zápis předán browseru jako datový tok HTTP. Komunikace vytvářená browserem bude potom přijata zachycovacím modulem na straně klienta 30 v bloku 100, viz obrázek 3.

Jestliže kontrola koherentního intervalu znázorněná v bloku 110 určí, že cache zápis, rezidentní na prvním počítači, je zastaralý, vygeneruje požadavek na zachycovací modul na straně serveru 40, aby zkontroloval koherentnost cache zápisu rezidentního na druhém počítači. Tato operace je znázorněna blokem 112 na obrázku 3. Je prováděna pomocí externího komunikačního spoje 35 použitého k předání koherentního intervalu na zachycovací modul na straně serveru 40 pro konkrétní zachycovací modul na straně klienta, který odpovídá web browseru 10, jenž vytvořil dotaz, a dále je předána jednoznačná indicie o obsahu klientovy cache, odpovídající URL komunikaci vytvářené web browserem. V preferovaném začlenění je tato jednoznačná indicie výsledkem cyklické redundační kontroly nebo CRC pro daný cache zápis.

Věnujme nyní pozornost obrázku 5, který znázorňuje činnost zachycovacího modulu na straně serveru v odpovědi na informaci přijatou přes externí komunikační spoj 35 od zachycovacího modulu na straně klienta 30. Když zachycovací modul na straně serveru 40 přijme požadavek od zachycovacího modulu na straně klienta, zachycovací modul na straně serveru 40 přijme předem definovaný koherentní časový interval, CRC hodnotu pro klientův cache zápis a HTTP dotaz, vytvořený web browserem. Potvrzení této informace je znázorněno v bloku 120 na obrázku 5.

Po přijetí informace od zachycovacího modulu na straně klienta 30 zkontroluje zachycovací modul na straně serveru 40 svou serverovou cache rezidentní na druhém počítači, aby určil, zda existuje serverový cache zápis odpovídající URL požadavku HTTP vytvořeného web browserem. Jestliže po prozkoumání komunikace vytvořené web browserem, jak je naznačeno v bloku 125, určí zachycovací modul na straně serveru 40, že existuje cache zápis odpovídající informaci, která byla vyžádána komunikací vytvořenou web browserem, uplatní se větev „Yes“ bloku 125. Zachycovací modul na straně serveru 40 potom porovná datum a čas modulu SSI 40 se sumou SDT serverového cache zápisu odpovídajícího informaci, která byla vyžádána komunikací

vytvořenou web browserem, a s předem definovaným klientovým koherentním časovým intervalem přijatým od zachycovacího modulu na straně klienta.

Jestliže je aktuální datum a čas menší než suma SDT pro serverový cache zápis a koherentní interval, potom se uplatní větev „Yes" bloku 130 na obrázku 5. Zachycovací modul na straně serveru 40 potom porovná CRC serverového cache zápisu s CRC klientova cache zápisu, aby určil, zda jsou tyto cache zápisy identické. Pokud jsou tyto dva cache zápisy identické, uplatní se větev „Yes" v bloku 135 a jak je znázorněno v bloku 136, odešle se „koherentní" odezva na zachycovací modul na straně klienta 30.

V případě, že podmínkový blok 135 určí, že CRC nejsou identické, potom nejsou identické ani informace obsažené v klientově cache paměti a v serverové cache paměti a jak je znázorněno v bloku 137, zachycovací modul na straně serveru odešle serverový cache zápis na první počítač přes externí komunikační spoj. Při odesílání serverového cache zápisu na zachycovací modul na klientově straně 30 konvertuje zachycovací modul na serverové straně zápis na klientův specifický komunikační protokol, který obsahuje CRC, data a stáří serverového cache zápisu. Stáří serverového cache zápisu je vypočítáváno odečtením SDT cache zápisu od aktuálního data a času.

A pokud, jak je konečně znázorněno na obrázku 5, je buď suma SDT plus předem stanovený klientův koherentní časový interval menší než aktuální datum a čas, nebo neexistuje žádný cache zápis odpovídající URL komunikaci, vytvořené web browserem, potom se uplatní větev „No" bloku 130, popřípadě větev „No" bloku 125. Tak se provedou operace bloku 126 a zachycovací modul na straně serveru 40 odešle na server komunikaci vytvořenou web browserem jako datový tok HTTP. Jestliže zachycovací modul na straně serveru 40 musí odeslat komunikaci vytvořenou web browserem na server jako datový tok HTTP, potom zachycovací modul na straně serveru 40 bude provádět operace popsané obrázkem 6.

Jak je patrno z bloku 140 na obrázku 6, v odpovědi na komunikaci vytvořenou web browserem bude zachycovací modul na straně serveru přijímat datový tok HTTP z web serveru 20. Po přijetí datového toku HTTP vypočte zachycovací modul na straně serveru 40 CRC pro datový tok HTTP a dočasně uloží tento datový tok HTTP. Potom, jak je znázorněno v bloku 145, zachycovací modul na straně serveru prouzkoumá datový tok HTTP a určí, zda existuje serverový cache zápis odpovídající URL datového toku HTTP. Jestliže takovýto zápis existuje, uplatní se cesta „Yes" bloku 145. Zachycovací modul na straně serveru 40 potom porovná nedávno vypočtený CRC datového toku HTTP přijatého od web serveru 20 s RC serverového cache zápisu odpovídajícího URL komunikaci v odpovědi vytvářené web serverem, jak je znázorněno v bloku 150. Jestliže jsou CRC identické, uplatní se větev „Yes" v bloku 150. Zachycovací modul na straně serveru 40 aktualizuje zápis SDT pro serverový cache zápis, jak je znázorněno v bloku 151, a smaže dočasně uložený datový tok HTTP přijatý web serverem 20, jak je znázorněno blokem 152.

Jestliže výsledky CRC indikují, že serverový cache zápis je jiný než datový tok HTTP přijatý od web serveru 20, potom se uplatní větev „No" bloku 150. Zachycovací modul na serverové straně 40 odstraní ze serverové cache paměti stávající data, jak je znázorněno v bloku 153, a potom, jak je patrné z bloku 154, aktualizuje serverovou cache novějšími informacemi. Z bloku 154 je dále patrné, že tato aktualizace zahrnuje uložení CRC komunikace web serveru do serverové cache, uložení aktuálního data a času ve formě SDT jako součásti cache zápisu a uložení datového toku HTTP. Vždy při aktualizaci serverového cache žápisu nebo při zjištění, že serverový cache zápis je identický s přijatým tokem HTTP od serveru 20, zachycovací modul na serverové straně určí, zda serverový cache zápis se shoduje s klientovým cache zápisem odpovídajícím komunikaci, která je vytvářena web browserem. Tato operace je znázorněna blokem 155.

Jestliže zachycovací modul na serverové straně 40 určí, že neexistuje cache zápis odpovídající odezvě od web serveru 20, uplatní se větev „No" bloku 145. Serverový cache zápis je vytvářen,

jak je znázorněno blokem 146, uložením URL odpovědi od web serveru, uložením CRC odpovědi od web serveru, uložením datového toku HTTP a uložením aktuálního data a času jako SDT. Po vytvoření cache zápisu odpovídajícího komunikaci vytvořené web serverem zachycující modul na straně serveru 40 znovu porovná CRC tohoto serverového cache zápisu s CRC odpovídajícím klientově cache zápisu, jak je znázorněno v bloku 155.

Jestliže výsledky porovnání serverového cache zápisu a klientova cache zápisu indikují, že cache zápisy jsou identické, uplatní se větev „Yes" bloku 155 a provedou se operace bloku 156. Z bloku 156 je zřejmé, že zachycovací modul na serverové straně 40 odesílá koherentní odezvu na zachycovací modul na straně klienta 30. Zachycovací modul na serverové straně 40 transformuje serverový požadavek a cache zápisu na specifický datový tok klient/server tak, že odešle koherentní odezvu a také nulové stáří na zachycovací modul na straně klienta.

Pokud zachycovací modul na straně serveru 40 určí, že klientův cache zápis není identický se serverovým cache zápisem odpovídajícím komunikaci vytvářené web browserem, uplatní se větev „No" bloku 155 a provedou se operace bloku 157. Jak je zřejmé z bloku 157, zachycovací modul na serverové straně 40 konvertuje nebo transformuje serverový cache zápis do specifického datového toku klient/server. Tento datový tok zahrnuje CRC serverového cache zápisu, datový tok HTTP serverového cache zápisu a stáří cache zápisu, které je nastaveno na nulu. Tato specifická komunikace klient/server je potom odeslána přes externí komunikační spoj 35 na zachycovací modul na straně klienta 30.

Funkce zachycovacího modulu na straně klienta 30 po přijetí komunikace od zachycovacího modulu na straně serveru bude popsána s ohledem na obrázek 4. Jak je zřejmé z bloku 160, zachycovací modul na straně klienta 30 přijímá nebo vyžaduje specifický datový tok klient/server, který byl přenášen přes externí komunikační spoj 35. Zachycovací modul na straně klienta potom určí, jaký typ odezvy byl přijat od zachycovacího modulu na straně serveru 40, jak je znázorněno v bloku 165. Jestliže zachycovací modul na straně serveru 40 indikuje, že klientův cache zápis je koherentní, tj. že jsou identické cache zápisy serveru a klienta, potom se provedou operace znázorněné blokem 166. Jak je zřejmé z bloku 166, aktualizuje zachycovací modul na straně klienta 30 SDT klienta cache zápisu odpovídajícího komunikaci vytvořené web browserem o rozdíly aktuálního data a času a stáří, které byly přijaty od zachycovacího modulu na straně serveru 40. Takže bez synchronizace hodin prvního počítače 5 a druhého počítače 6 stávající vynález zrevidoval koherentní čas cache zápisu prvního počítače tak, aby v něm zohlednil novější data z druhého počítače. po aktualizaci SDT klientova cache zápisu odpovídajícího komunikaci vytvořené web browserem odešle zachycovací modul na straně klienta 30 klientův cache zápis do web browseru 10 jako datový tok HTTP. Tato činnost je znázorněna blokem 174.

Jestliže však zachycovací modul na straně klienta 30 určí, že typem odpovědi jsou daty nebo datový tok, uplatní se větev „stream" bloku 165 a provedou se operace bloku 167. Zachycovací modul na straně klienta 30 přijme datový tok HTTP a dočasně tato data uloží. Potom, jak je znázorněno v bloku 170 na obrázku 4, zachycovací modul na straně klienta 30 určí, zda existuje cache zápis odpovídající komunikaci vytvořené web browserem. Jestliže tento cache zápis existuje, uplatní se cesta „Yes" bloku 170 a jak je naznačeno v bloku 171 regeneruje se stávající cache zápis. Zachycovací modul na straně klienta potom aktualizuje cache zápis odpovídající komunikaci vytvořené web browserem, a to uložením CRC datového toku HTTP přijatého od zachycovacího modulu na straně serveru 40, uložením jako SDT rozdílu mezi aktuálním datem, časem a stářím přijatými od zachycovacího modulu na straně serveru 40 a uložením datového toku HTTP. Tato operace je znázorněna blokem 172.

Jestliže neexistuje cache zápis odpovídající komunikaci vytvořené web browserem, uplatní se cesta „No" bloku 170. Je vytvořen klientův cache zápis, a to provedením operací znázorněných blokem 173. Jak je zřejmé z bloku 173, vytvoří zachycovací modul na straně klienta 30 klientův cache zápis uložením URL datového toku HTTP přijatého od zachycovacího modulu na straně serveru 40, uložením CRC datového toku HTTP přijatého od zachycovacího modulu na straně

serveru 40 a uložením datového toku HTTP. Zachycovací modul na straně klienta 30 rovněž aktualizuje SDT nebo uloží SDT odečtením stáří od aktuálního data a času, které bylo přijato přes externí komunikační spoj 35 od zachycovacího modulu na straně serveru 40.

Avšak klientův cache zápis se vytváří při každém průchodu operacemi bloků 166, 172 nebo 173. V tom případě zachycovací modul na straně klienta předá nebo poskytne klientův cache zápis web browseru 10 jako datový tok HTTP. Tyto operace jsou znázorněny v bloku 174 na obrázku 4.

Z pohledu hodnocení tohoto vynálezu je nutné podotknout, že klientova a serverová cache paměť mohou být implementovány s jinou pamětí, například s pevnými disky, CD–ROM, optickými disky nebo jinými technologiemi ukládání. Dále je nutné v souvislosti s tímto vynálezem zmínit, že zachycovací modul na straně klienta a zachycovací modul na straně serveru mohou být implementovány prostřednictvím softwaru, hardwaru nebo jejich kombinací.

Zatímco reference o vytvoření cache pamětí jsou rezidentní na konkrétním prvním nebo druhém počítači, je nutné z pohledu posuzování tohoto vynálezu uvést, že přednosti z něj vyplývající mohou být získány i v případě, že cache paměti nebudou rezidentní na prvním počítači, ale jsou pouze na stejné straně externího komunikačního spoje jako počítač. Hardwarová cache paměť může být implementována externě na první počítač, kde slouží jako klientova cache paměť, a může být připojena k prvnímu počítači vysokorychlostní komunikaci a i přesto, pokud je cache na stejné straně komunikačního spoje jako první počítač, bude dosaženo výhod stávajícího vynálezu.

V alternativním začlenění stávajícího vynálezu zachycující modul na straně serveru 40 neuchová kopii datového toku HTTP přijatého od web serveru 20, ale pouze zápis adresáře pro komunikaci. Zápis adresáře pak obsahuje URL komunikace, CRC vypočtený pro datový tok HTTP a čas, kdy byl datový tok HTTP přijat od web serveru, a SDT pro komunikaci, který může být nastaven na čas, kdy byl CRC vypočten. V takovém případě, když zachycovací modul na straně klienta 30 odešle požadavek zachycovacímu modulu na straně serveru 40 na komunikaci odpovídající URL, pro kterou udržuje zachycovací modul na straně serveru CRC a SDT, tak zachycovací modul na straně serveru zkontroluje CRC přijatý od zachycovacího modulu na straně klienta 30, aby určil, zda odpovídá CRC nejnovějšímu datového toku HTTP pro specifikovaný URL. Pokud odpovídá, potom je odeslána koherentní odpověď na zachycovací modul na straně klienta. Neodpovídá–li, odešle zachycovací modul na straně serveru datový tok HTTP přijatý od zachycovacího modulu na straně klienta na web server 20 a vrátí na zachycovací modul na straně klienta 30 odpověď přijatou od web serveru 20.

Obrázky 7, 8, 9 a 10 znázorňují operace prováděné zachycovacím modulem na straně klienta 30 a zachycovacím modulem na straně serveru 40 v jiném aspektu stávajícího vynálezu, který využívá stanovení rozdílu (diferencování) k tomu, aby snížil množství dat přenášených přes externí komunikační spoj 35. Jak je zřejmé z obrázku 7, blok 200 znázorňuje potvrzení zachycovacího modulu na straně klienta 30 na požadavek HTTP od web browseru 10. Jak je uvedeno v bloku 205, zachycovací modul na straně klienta 30 prozkoumá zachycený požadavek HTTP od web browseru 10, aby určil, zda je tento požadavek Common Gateway Interface (CGI). V případě, že tento požadavek je Common Gateway Interface, postoupí ho zachycovací modul na straně klienta 30 zachycovacímu modulu na straně serveru, jak je znázorněno na obrázcích 3 až 6 a je ilustrováno blokem 206 na obrázku 7.

Pokud však komunikace vytvořená web browserem odpovídá požadavku CGI, uplatní se větev „Yes“ bloku 205. A jak je reflektováno v bloku 210, zachycovací modul klient/server 30 stanoví, zda existuje klientův bázový cache zápis odpovídající datovému toku HTTP, který byl předtím poskytnut na web browser v odpovědi na jemu odpovídající požadavek CGI. Toto zachycení požadavku CGI může být prováděno porovnáním URL komunikace vytvořené web browserem s URL uloženými v klientově bázové cache paměti.

Klientova bázová cache může být inicializována uložením prvního datového toku HTTP přijatého zachycovacím modulem na straně klienta 30, který by měl být poskytnut na web browser 10 pro daný URL. Tento bázový cache zápis může být udržován pomocí množství nebo relací web browseru 10. Klientovy bázové cache zápisy mohou být aktualizované způsobem znázorněným na obrázcích 7, 8, 9 a 10. Jestliže existuje klientův bázový cache zápis odpovídající URL pro komunikaci vytvořenou web browserem, potom CRC, který má být odeslán na zachycovací modul na straně serveru 40 přes externí komunikační spoj 35, je nastaven stejně jako CRC pro klientův bázový cache zápis, jak je znázorněno v bloku 211 na obrázku 7. Jestliže takový klientův bázový cache zápis existuje, potom se uplatní větev „No" bloku 210 na obrázku 7 a CRC pro požadavek, který má být odeslán přes externí komunikační spoj 35 na zachycovací modul na straně serveru 40, je nulován. Tato operace je znázorněna v bloku 212 na obrázku 7.

Blok 213 ilustruje operace odesílání požadavku CGI na zachycovací modul na straně serveru 40 přes externí komunikační spoj. Jak je v bloku 213 znázorněno, zachycovací modul na straně klienta 30 odešle požadavek HTTP a požadavek CRC, který má být buď nastaven na nulu (pokud neexistuje klientův bázový cache zápis pro URL požadavku CGI), nebo má být nastaven na CRC klientova bázového cache zápisu (pokud takový zápis existuje). Zachycovací modul na straně klienta tedy konvertoval požadavek CGI na specifický protokol klient/server přenášený specifickou komunikací klient/server přes externí komunikační spoj, který má být přijat zachycovacím modulem na straně serveru 40.

Činnosti prováděné zachycovacím modulem na straně serveru v době, kdy je přijat požadavek CGI, jsou znázorněny na obrázku 9. Potvrzení požadavku CGI zachycovacím modulem na straně serveru 40 je znázorněno v bloku 220. Když zachycovací modul na straně serveru 40 přijme požadavek CGI, uloží kopii hodnoty CRC a požadavek HTTP. Jak je zřejmé z bloku 221, zachycovací modul na straně serveru 40 předá požadavek HTTP na web server 20.

Když zachycovací modul na straně serveru 40 přijme odpověď na požadavek HTTP odpovídající komunikaci vytvořené web browserem nebo požadavek CGI, bude tato odpověď zachycovacím modulem na straně serveru 40 přijata jako datový tok HTTP, jak je znázorněno v bloku 230 na obrázku 10. Jak je zřejmé z bloku 230, zachycovací modul na straně serveru 40 uloží datový tok HTTP a vypočte hodnotu CRC pro datový tok HTTP přijatý od web serveru 20. Zachycovací modul na straně serveru 40 rovněž vynuluje rozdílovou (diferenční) hodnotu, a to na inicializační rozdílová data. Zachycovací modul na straně serveru potom stanoví, zda odpověď přijatá jako komunikace vytvořená web serverem je odpověď na požadavek CGI, jak je znázorněno blokem 235. Jestliže je odpověď ne, uplatní se cesta „No" bloku 235 na obrázku 10 a provedou se operace bloku 236 pro odeslání datového toku HTTP na zachycovací modul na straně klienta. Jak je zřejmé z bloku 236, může tato operace zahrnovat operace zachycení popsané obrázky 3 až 6. Jestliže odpověď přijatá blokem 230 je odpověď na požadavek CGI, potom se uplatní cesta „Yes" bloku 235 a zachycovací modul na straně serveru dále stanoví, jestli existuje serverový bázový cache zápis pro odpověď CGI, jak je znázorněno v bloku 240.

Serverový bázový cache zápis může být vytvořen, když zachycovací modul na straně serveru 40 poprvé přijme odpověď na požadavek CGI. V tomto případě bude výsledkem podmínky znázorněné blokem 240 provedení cesty „No" bloku 240. Zachycovací modul na straně serveru 40 potom vytvoří serverový bázový cache zápis odpovídající požadavku CGI, a to uložením URL pro CGI, datového toku HTTP odpovědi na požadavek CGI a CRC pro datový tok HTTP. Tato operace je znázorněna v bloku 241. Má-li být dosaženo kompatibilnosti s koherentním cache systémem popsaným obrázky 3 až 6, může serverový bázový cache zápis také obsahovat SDT. Podobně, jak je zde použito, označuje termín forma serverové CGI báze serverový bázový cache zápis odpovídající požadavku CGI přijatému od web browseru 10.

Jestliže existuje serverový bázový cache zápis odpovídající požadavku CGI, uplatní se cesta „Yes" bloku 240. Zachycovací modul na straně serveru porovná CRC serverového bázového

cache zápisu s CRC odpovědi přijaté od web serveru 20. Tyto operace jsou znázorněny v bloku 245 na obrázku 10. V případě, že jsou CRC shodné, stanoví zachycovací modul na straně serveru, zda CRC pro serverový bázový cache zápis odpovídá CRC pro klientův bázový cache zápis. Jestliže jsou tyto dvě hodnoty CRC stejné, potom klientův bázový cache zápis, serverový bázový cache zápis a odezva přijatá z web serveru 20 budou obsahovat stejný datový tok HTTP. Porovnání serverového bázového cache zápisu s klientovým bázovým cache zápisem je znázorněno v bloku 250.

Jestliže jsou tyto dva zápisy shodné, potom zachycovací modul na straně serveru nemusí odesílat bázový cache zápis na zachycovací modul na straně klienta 30 a tedy, jak je znázorněno v bloku 251, tok HTTP dat, která mají být přenesena na zachycovací modul na straně klienta 30, je vynulován. Zachycovací modul na straně serveru 40 potom konvertuje datový tok HTTP přijatý od web serveru 20 na specifický komunikační protokol klient/server odesláním CRC datového toku HTTP uloženého v serverové bázové cache paměti odpovídajícího požadavku CGI, vynulovaný tok HTTP dat a vynulovaná rozdílová (diferenční) data s cílem indikovat, že odpověď CGI byla identická klientovu bázovému cache zápisu, jak je znázorněno blokem 252.

Zpět k bloku 245. Jestliže je CRC pro serverový bázový cache zápis odpovídající požadavku CGI odlišný od CRC pro odpověď přijatou od web server v odpovědi na požadavek CGI vytvořený web browserem, potom se uplatní větev „No“ bloku 245. Zachycovací modul na straně serveru 40 potom provede operace znázorněné blokem 246. Zachycovací modul na straně serveru 40 porovná zachycenou odpověď CGI se serverovým bázovým cache zápisem odpovídajícím zachycenému požadavku CGI nebo s bázovou formou CGI. Toto porovnání zachycené odpovědi CGI se serverovou bázovou formou CGI poskytne rozdílová (diferenční) data CGI, která odpovídají rozdílu mezi zachycenou odpovědí CGI a serverovou bázovou formou CGI.

Stanovení rozdílu (diferencování) může být, z pohledu hodnocení stávajícího vynálezu, provedeno libovolnou známou metodou pro stanovení rozdílu mezi bázovou formou a upravenou (modifikovanou) formou. Jedna metoda vhodná pro použití ve stávajícím vynálezu je popsána v „Cross–Platform Binary Diff“ publikované Dr. Dobb's Journal, květen 1995, str. 32–36. Její popis je zde zapracován, a rovněž je zde uvedena reference pro plnou citaci. Dalšími metodami, které mohou být použity pro stanovení rozdílových dat, jsou metody popsané v IBM Technical Disclosure Bulletin, sv. 22, č. 8A, leden 1980, které jsou zde rovněž zapracovány a pro plnou citaci uvedeny referenční odkazy. Zachycovací modul na straně serveru 40 potom stanoví, zda serverová bázová forma CGI vyžaduje aktualizaci, jak je znázorněno v bloku 247. Toto stanovení může být provedeno určením, zda průměrná rozdílová data mezi zachycenou odpovědí CGI a serverovou bázovou formou CGI překračují stanovený práh. Další metody pro stanovení, zda serverový bázový cache zápis odpovídající požadavku CGI by měl být aktualizován, mohou zahrnout časovou koherenci, tak jak je popsáno obrázky 3 až 6, nebo to mohou být i jiné známé metody vhodné ke stanovení, zda se rozdílová data změnila natolik, že je zapotřebí vytvořit nový bázový cache zápis ke zlepšení systémové výkonnosti.

Jestliže vytvoření nového bázového cache zápisu není zapotřebí, uplatní se větev „No“ bloku 247 a zachycovací modul na straně serveru 40 provede operace popsané blokem 250 ke stanovení, zda CRC klientova bázového cache zápisu je shodný se serverovým bázovým cache zápisem nebo zda je serverová bázová forma CGI identická s klientovou bázovou formou CGI, což jsou v podstatě bázové cache zápisy serveru a klienta, jež odpovídaly konkrétnímu požadavku CGI komunikace vytvořené web browserem. Jestliže jsou tyto bázové formy shodné, není nutné obnovovat a informace o datovém toku HTTP mohou být vynulovány, jak je znázorněno v bloku 251. Zachycovací modul na straně serveru 40 potom odešle rozdílovou odpověď na zachycovací modul na straně klienta 30, a to odesláním CRC serverového bázového cache zápisu odpovídajícího požadavku CGI (tj. CRC serverové bázové formy CGI), odesláním vynulovaného datového toku HTTP, který by korespondoval s bázovými daty, a odesláním rozdílových (diferenčních) dat stanovených v bloku 246. Tyto operace jsou opět reflektovány jako blok 252 na obrázku 10.

Jestliže zachycovací modul na straně serveru 40 stanoví, že porovnávané CRC nejsou shodné (pro klientovou bázovou formu CGI a serverovou bázovou formu CGI), potom je nutné obnovit klienta. Operace obnovení klienta zahrnuje odeslání serverové bázové formy CGI na zachycovací modul na straně klienta 30. Při provedení této operace zachycovací modul na straně serveru nastaví data datového toku HTTP pro odeslání na zachycovací modul na straně klienta 30 shodně se serverovou bázovou formou CGI. Tatooperace je znázorněna v bloku 253. Zachycovací modul na straně serveru 40 potom konvertuje datový tok HTTP přijatý od web serveru na specifický protokol klient/server odesláním CRC serverové bázové formy CGI, dat datového toku HTTP odpovídajícímu serverové bázové formě CGI a odesláním rozdílových (diferenčních) dat bázové formy CGI a odezvy přijaté od web serveru, jak je zřejmé z bloku 252. Tato informace je pak předána přes externí komunikační spoj 35 na zachycovací modul na straně klienta 30.

Zpět k bloku 247. Jestliže je zapotřebí obnova, uplatní se cache „Yes" bloku 247. Jak je znázorněno blokem 248, zachycovací modul na serverové straně aktualizuje serverový bázový cache zápis odpovídající komunikaci vytvořené browserem a datový tok HTTP přijatý od web serveru. Aktualizován je rovněž CRC odpovědi a rozdílová (diferenční) data CGI jsou vynulována. Zachycovací modul na straně serveru pak porovná CRC nového cache zápisu serverové strany, jak je znázorněno v bloku 250, a dokončí přenos výše popsaným způsobem.

Operace zachycovacího modulu na straně klienta po přijetí odpovědi od zachycovacího modulu na straně serveru 40 jsou znázorněny na obrázku 8. Přijetí odpovědi od zachycovacího modulu na straně serveru 40 zachycovacím modulem na straně klienta 30 je znázorněno v bloku 260. Jak je zřejmé z bloku 265, zachycovací modul na straně klienta 30 stanoví, zda odpověď je odpovědí na požadavek CGI. Pokud to není odpověď na požadavek CGI, potom zachycovací modul na straně klienta provede operace popsané blokem 267, které mohou zahrnovat operace cache znázorněné obrázky 3 až 6. Jestliže však odpověď na požadavek CGI, uplatní se větev „Yes" bloku 265. Zachycovací modul na straně klienta 30 uloží data datového toku HTTP, rozdílová data a CRC vyžádaný od specifického datového toku klient/server přenášeného přes externí komunikační spoj. Tyto operace jsou reflektovány v bloku 266 na obrázku 8.

Zachycovací modul na straně klienta 30 potom stanoví, zda existuje klientův bázový cache zápis odpovídající zachycenému požadavku CGI, který by obsahoval klientovu bázovou formu CGI. Toto zkoumání je zaznamenáno v bloku 270 a může být prováděno prověřením URL požadavku HTTP nebo odpovědi HTTP. Jestliže klientova bázová forma CGI existuje, uplatní se větev „Yes" bloku 270. Zachycovací modul na straně klienta 30 potom porovná CRC přijatý přes externí komunikační spoj s CRC klientovy bázové formy CGI, jak je znázorněno v bloku 275. Pokud jsou odlišné, uplatní se cesta „No" bloku 275 a provede se obnova klienta aktualizací bázové formy CGI, a to výměnou klientova bázového cache zápisu odpovídajícího URL požadavku CGI komunikace vytvořené web browserem za data datového toku HTTP přijatá přes externí komunikační spoj 35 od zachycovacího modulu na straně serveru 40. Klientův bázový cache zápis je rovněž aktualizován s ohledem na CRC pro datový tok HTTP. Tyto operace jsou reflektovány v bloku 276 na obrázku 8.

Jestliže CRC přijatý přes externí komunikační spoj 35 je stejný jako CRC bázové formy CGI, potom serverová bázová forma CGI na zachycovacím modulu na straně serveru je shodná s klientovou bázovou formou CGI na zachycovacím modulu na straně klienta, a uplatní se větev „Yes" bloku 275.

Jsou–li bázové formy stejné nebo klient byl obnoven, budou provedeny operace uvedené v bloku 277, a to zachycovacím modulem na straně klienta 30. Blok 277 odráží, jak zachycovací modul na straně klienta 30 rekonstruuje datový tok HTTP odpovídající komunikaci od web serveru 20 ze specifického datového toku klient/server přijatého přes externí komunikační spoj 35, a to sloučením (zkombinováním) klientové bázové formy CGI s rozdílovými (diferenčními) daty CGI přijatými přes externí komunikační spoj 35 tak, aby se vytvořil datový tok HTTP odpovídající

zachycené odpovědi CGI. Jak je zřejmé z bloku 278, bude tato odpověď poskytnuta web browseru 10 jako datový tok HTTP.

Pokud u klienta neexistuje žádná bázová forma CGI odpovídající URL požadavku CGI, potom, se uplatní větev "„o"“bloku 270 na obrázku 8. Jak je zřejmé z loku 271, zachycovací modul na straně klienta 30 vytvoří klientův bázový cache zápis odpovídající URL požadavku CGI, a to uložením URL, CRC datového toku HTTP přijatého přes externí komunikační spoj od zachycovacího modulu na straně serveru 40 a skutečných dat datového toku HTTP. Uložení těchto informací vytvoří klientův bázový cache zápis odpovídající zachycenému požadavku CGI, a tedy vytvoří klientovu bázovou formu CGI. Zachycovací modul na straně klienta může pak provést operace bloku 277, a to rekonstruováním datového toku HTTP pomocí sloučení (zkombinování) nebo vnoření klientovy bázové formy CGI a rozdílových (diferenčních) dat CGI, které mohly být vynulovány.

Stávající techniky vytváření rozdílu (diferencování) mohou být uplatněny rovněž u dat jiných než CGI. V takovém případě by zachycovací modul na straně serveru 40 potřeboval uchovávat násobné generace serverových bázových cache zápisů, které by mu umožnily, aby zachycovací moduly na straně klienta od připojených web browserů na web server mohly mít různé bázové formy. Zachycovací modul na straně serveru by potom mohl porovnávat CRC přijatý od zachycovacího modulu na straně klienta s CRC jednotlivých předchozích generací serverových bázových forem až do doby, kdy by byla nalezena (získána) shoda. Zachycovací modul na straně serveru 40 může potom volitelně obnovit zachycovací modul na straně klienta 30 nebo prostě předat rozdílová (diferenční) data na zachycovací modul na straně klienta 30. Rozdílové (diferenční) metodiky zde popisované by se tedy pak uplatnily s ohledem na požadavek CGI stejným způsobem, jako u každého jiného požadavku HTTP a odpovědi.

Zatímco výše uvedený systém udržování násobných generací bázových forem může dovolovat použití diferencování i u jiných požadavků než CGI, je tato metodika paměťově neboli kapacitně náročnější a nevyužívá plně výše popsaných zachycovacích schopností. V zájmu snížení nároků na paměť či kapacitu a využití výše popisovaných zachycovacích metod je upřednostněna možnost použití následující metody diferencování u jiných než CGI požadavků. U této preferované implementace vypočítává zachycovací modul na straně serveru rozdíl (diferenci) mezi serverovou bázovou formou odpovídající požadavku a datovým tokem HTTP odpovědi od web serveru. tato rozdílová (diferenční) data jsou pak uložena zachycovacím modulem na straně serveru. Serverová bázová forma je potom aktualizována výměnou bázové formy za novou odpověď od web serveru, a to včetně aktualizace CRC této bázové formy. Avšak spíše než zlikvidováním starého CRC, jsou jednotlivé CRC pro dřívější bázové formy uloženy jako rozdílová data. Předchozí generace rozdílových (diferenčních) dat a jednotlivých CRC jsou pak selektivně přenášeny na zachycovací modul na straně klienta, a to na základě CRC klientovy bázové formy odpovídající požadavku jinému než CGI.

Příklad metody diferencování u jiných požadavků než CGI – jestliže zachycovací modul na straně serveru přijme jiný požadavek než CGI, byl by tento požadavek rovněž doprovázen CRC bázové formy rezidentní na zachycovacím modulu na straně klienta odpovídající URL non–CGI požadavku. Když by zachycovací modul na straně serveru přijal odpověď od web serveru, vypočítal by CRC odpovědi. Zachycovací modul na straně serveru by potom vypočetl rozdíl mezi odpovědí a serverovou bázovou formou pro URL a tato rozdílová data by uložil. Zachycovací modul na straně serveru by dále aktualizoval serverovou bázovou formu a data odpovědi a archivoval CRC dřívější bázové formy i rozdílová (diferenční) data mezi odpovědí a starou bázovou formou. Zachycovací modul na straně serveru by potom porovnal CRC klientovy bázové formy s CRC serverové bázové formy a se všemi uloženými nebo archivovanými CRC tak, aby určil, zda je nalezena shoda, pokud shoda není nalezena, bude odpověď jednoduše odeslána na zachycovací modul na straně klienta.

Jestliže je nalezena shoda, jsou odeslána rozdílová data odpovídající shodnému CRC a všechna následná rozdílová (diferenční) data včetně aktuálních rozdílových dat na zachycovací modul na straně klienta. Zachycovací modul na straně klienta pak rozdílová data použije v klientově bázové formě k rekonstrukci odpovědi. Potom, vyskytne-li se např. shoda CRC a CRC pro bázovou formu, která byla tři generace stará, budou odeslány tři sady rozdílových dat na zachycovací modul na straně klienta a konstrukce odpovědi bude provedena uplatněním těchto tří následných (diferenčních) datových sad v klientově bázové formě. Jestliže však jsou počet sad rozdílových dat nebo velikost sad rozdílových dat potřebných pro rekonstrukci odpovědi tak značné, že odeslání skutečné odpovědi by vyžadovalo přenášení méně dat, potom může být odeslána zachycovacím modulem na straně serveru vlastní odpověď. V každém případě by po rekonstrukci nebo po přijetí odpovědi zachycovací modul na straně klienta aktualizoval klientovu bázovou formu pro URL požadavku o data odpovědi a také aktualizoval CRC o CRC pro odpověď. Protože je klientova bázová forma aktualizována při každém přijetí pro konkrétní URL, může být využito výše popsané klientovy cache paměti jako cache pro klientovu formu, a tak dosaženo eliminace potřeby mít samostatnou cache paměť klientových bázových forem za situace, kdy je diferencování využíváno pro jiné požadavky než CGI.

Uplatněním dalšího aspektu stávajícího vynálezu může být další úspora komunikace, a to na základě omezení redundance nestavového komunikačního protokolu, jako např. HTTP. U tohoto protokolu klient odesílá informace o sobě na server pokaždé, když je iniciována komunikace. Podobně server poskytuje specifické informace o sobě vždy, když iniciuje svoji odpověď na klienta.

V jednom z alternativních začlenění stávajícího vynálezu první počítač 5 komunikuje s druhým počítačem 6, kde mu předává počítačově specifické informace související s předem definovanými charakteristikami prvního počítače. Druhý počítač tuto počítačově specifickou informaci uloží. První počítač potom odstraní počítačově specifickou informaci z následných komunikací vytvářených web browserem před odesláním přes externí komunikační spoj 35. Druhý počítač 6 potom rekonstruuje originální komunikaci vytvořenou web browserem, a to sloučením (zkombinováním) uložené počítačově specifické informace a následné komunikace přijaté přes externí komunikační spoj 35 tak, aby vytvořila datový tok HTTP.

Kromě odstranění počítačově specifických informací z komunikací vytvořených web browserem je možné tyto počítačově specifické informace odstranit i z komunikací vytvářených web serverem. V tomto případě druhý počítač 6, viz obrázek 2, poskytne prvnímu počítači 5 přes externí komunikační spoj 35 počítačově specifické informace v souladu s předem definovanými charakteristikami druhého počítače 6. První počítač 5 uloží tyto počítačově specifické informace tak, aby dokázal vytvořit serverové záhlaví. U následné komunikace potom druhý počítač 6 odstraní počítačově specifické informace komunikace vytvořené web serverem a odešle zbývající část komunikace vytvořená web serverem na externí komunikační spoj 35. První počítač tuto komunikaci 5 přijme přes externí komunikační spoj a rekonstrukce původní komunikaci vytvořenou web serverem, a to sloučením (zkombinováním) informace serverového záhlaví a specifického datového toku klient/server přijatého přes externí komunikační spoj tak, aby vytvořil datový tok HTTP. V obou případech jsou operace odstranění počítačově specifických informací a uložení těchto informací pro vytvoření buď informace serverového záhlaví, nebo informace klientova záhlaví prováděny zachycovacím modulem na straně klienta 30 či zachycovacím modulem na straně serveru 40, v závislosti na tom, zda se operace provádějí na prvním počítači 5 nebo na druhém počítači 6.

V jednom začlenění stávajícího vynálezu komunikuje web browser 10 se zachycovacím modulem na straně klienta 30 s využitím protokolu nazvaného Transmission Control Protocol/Internet Protocol (TCP/IP). TCP může být rovněž použit pro komunikaci mezi zachycovacím modulem na straně klienta 30 a zachycovacím modulem na straně serveru 40 přes externí komunikační spoj 35. Konečně, TCP může být použit pro komunikaci mezi zachycovacím modulem na straně serveru 40 a web serverem 20. Zatímco TCP může být použit

pro komunikace mezi různými komponenty, které vytvářejí systém stávajícího vynálezu, protokol HTTP neposkytuje nejefektivnější prostředky pro komunikaci přes externí komunikační spoj. Ke zvýšení výkonnosti externího komunikačního spoje 35 vytváří jedno ze začlenění stávajícího vynálezu to, co se zde označuje pod pojmem „virtuální sokety", které se využívají při spojení mezi web browserem a zachycovacím modulem na straně klienta 30 a mezi zachycovacím modulem na straně serveru 40 a web serverem 20. Činnost těchto virtuálních soketů bude nyní popsána s odvoláním na obrázky 11 až 17.

Obrázek 11 je blokové schéma jedné z možných implementací stávajícího vynálezu využívající koncepci virtuálních soketů. Jak je zřejmé z obrázku 11, první počítač 5 a druhý počítač 6 jsou vzájemně propojeny přes externí komunikační spoj 35. Web browser 10 má pluralitu reálných soketů, které propojují web browser 10 se zachycovacím modulem na straně klienta 30. Na obrázku 11 je první reálný soket znázorněn na web browseru 10 jako 65a a jemu odpovídající soket na zachycovacím modulu na straně klienta 30 jako 65b. Tento první reálný soket je soket TCP, přes který web browser 10 vyžaduje další spojení od zachycovacího modulu na straně klienta 30.

Když web browser 10 vyžaduje nové spojení TCP, objeví se komunikace přes reálný soket 65a, která je přijata na reálný soket 65b. Zachycovací modul na straně klienta 30 vytvoří potom další reálný soket pro komunikaci s web browserem 10. Jak je zřejmé z obrázku 11, pluralita reálných soketů je vytvořena na web browseru 10 společně s odpovídajícím reálným soketem vytvářeným na zachycovacím modulu na straně klienta 30. Tyto reálné sokety 60a až 64a jsou znázorněny na web browseru 10 a 60b až 64b na zachycovacím modulu na straně klienta 30u. Tyto reálné sokety tvoří prostředky, přes které web browser 10 komunikuje se zachycovacím modulem na straně klienta 30. Po vytvoření reálných soketů 60a až 64a a 60b až 64b jsou komunikace přes tyto sokety multiplexovány na reálný soket 36a, který poskytuje přístup zachycovacímu modulu na straně klienta 30 na externí komunikační spoj 35. Reálné sokety 36a a 36b jsou vytvořeny v době, kdy je odeslán požadavek přes reálný soket 37a počítače 5 na reálný soket 37b počítače 6. Po potvrzení požadavku o spojení reálným soketem 37b se vytvoří reálné sokety 36a a 36b. Sokety 37a a 37b pracují jako první reálné sokety pro potřeby komunikace mezi zachycovacím modulem na straně klienta a zachycovacím modulem na straně serveru a mohou být používány pouze pro ustanovení spojení mezi dvěma moduly znázorněnými sokety 36a a 36b. Každý z těchto reálných soketů pracuje pod standardními protokoly TCP/IP. Když jsou druhým počítačem 6 přijímány komunikace přes externí komunikační spoj 35, jsou přijmuty na reálný soket 36b. Zachycovací modul na straně serveru 40 potom demultiplexuje tyto komunikace na soketu 36b a poskytuje je příslušnému soketu za účelem přenosu na web server 20. Tak může být například uskutečněna komunikace přes soket 60a na 60b za účelem požadavku informace od specifického URL, která by byla multiplexována na soketu 36a, přijata soketem 36b, demultiplexována zachycovacím modulem na straně serveru 40 a přenesena ze soketu 60c na soket 60d na web serveru 20. Podobně jsou i komunikace vyskytující se přes soket 61a přijaty soketem 61b, multiplexovány zachycovacím modulem na straně klienta 30 a odeslány ze soketu 36a na soket 36b, kde zachycovací modul na straně serveru 40 tyto komunikace demultiplexuje a odešle přes soket 61c na soket 61d. Takže komunikace přes soket 60a a 60b, 61a a 61b, 62a a 62b, 63a a 63b a 64a a 64b jsou přenášeny přes příslušné odpovídající sokety mezi zachycovacím modulem na straně serveru 40 a web serverem 20, tj. soket 60c a soket 60d, soket 61c a 61d, soket 62c a soket 62d, soket 63c a soket 63d a soket 64c a 64d.

Podobně pak odpovědi na otázky od web browseru 10 produkované web serverem 20 jsou rovněž přenášeny přes soketové spojení web serveru 20 na zachycovací modul na straně serveru 40 a přes externí komunikační spoj 35 na zachycovací modul na straně klienta 30, a potom dále na web browser 10. Potom například by odpověď vytvořená web serverem 20 byla odeslána přes soket 60d na soket 60c a mutiplexována zachycovacím modulem na straně serveru 40 na soket 36b, která by byla odeslána přes externí komunikační spoj 35 na soket 36a. Zachycovací modul na straně klienta 30 potom demultiplexuje komunikaci a poskytne ji na soket 60b pro přenos na 60a na web browseru 10. Obdobná komunikační cesta je ustanovena pro každý soket, jenž je

využíván web browserem 10 nebo web serverem 20. Z pohledu hodnocení tohoto vynálezu je nutné poznamenat, že zatímco byl stávající vynález popisován s ohledem na 4 soketová spojení mezi web browserem 10 a web serverem 20, může být otevřen libovolný počet soketů pro zajištění komunikačního přístupu mezi web browserem 10 a web serverem 20.

Obrázek 12 představuje blokové schéma implementace virtuálního soketového systému v zachycovacím modulu na straně klienta 30 a zachycovacím modulu na straně serveru 40. Externě k těmto modulům jsou reálné sokety mezi zachycovacím modulem na straně klienta 30 a web browserem 10 a zachycovacím modulem na straně serveru 40 a web serverem 20, které fungují jako normální TCP/IP sokety. To znamená, že použití virtuálních soketů je transparentní vůči web browseru 10 i web serveru 20.

Konkrétní začlenění stávajícího vynálezu bude popsáno s ohledem na blokové schéma uvedené na obrázku 12 a vývojové diagramy na obrázcích 13 až 17. Obrázek 13 je vývojový diagram soketového manažera znázorněného jako blok 68 na obrázku 12. S odvoláním na obrázek 13, blok 300 odráží vytváření manažera reálných soketů 68 zachycovacího modulu na straně klienta 30. Poté, co byl manažer reálných soketů 68 vytvořen, vytvoří se první reálný soket znázorněný na obrázku 12 jako soket 65b. Vytvoření tohoto prvního reálného soketu je reflektováno blokem 301 na obrázku 13. Po vytvoření prvního reálného soketu 65b počká manažer soketů 68 rezidentní na zachycovacím modulu na straně klienta 30, v tomto materiálu rovněž označovaný jako klientův manažer soketů, na událost na prvním reálném soketu 65b, jak je znázorněno v bloku 302 na obrázku 13. Jakmile je přijata událost na prvním reálném soketu 65b, manažer reálných soketů 68 ji prozkoumá a s ohledem na výsledek tohoto prozkoumání zvolí jednu z pěti možných cest, jak je znázorněno v bloku 305 na obrázku 13.

V případě vytvoření reálného soketu v odpovědi na požadavek o komunikaci přijatý na první reálný soket 65b, přidá, jak je naznačeno v cestě od bloku 305 do bloku 306 na obrázku 13, manažer reálných soketů 68 reálný soket do seznamu reálných událostí. Manažer reálných soketů potom vytvoří simplexní virtuální soket, jak je naznačeno v bloku 307. V případě zachycovacího modulu na straně klienta bude manažer reálných soketů indikovat aplikační funkci, která zajistí provedení funkcí zachycovacího modulu na straně klienta pro vytvořený virtuální soket, jak je znázorněno v bloku 308 na obrázku 13.

Jak je uvedeno, termín „simplexní soket“ nebo „simplexní virtuální soket“ odkazuje na soket, který spojuje přímo buď jednoduchý soket, nebo jednoduchou aplikaci. Zde je použit také „multiplexní soket“, který označuje soket pro spojení na pluralitu dalších soketů. Multiplexní soket tedy vykonává multiplexní nebo demultiplexní funkci a simplexní soket vykonává individuální spojení. To znamená například při provádění funkcí bloků 306 až 308 na obrázku 13, že manažer soketu klienta 68 v odpovědi na požadavek o první spojení přijatý prvním reálným soketem 65b vytvoří soket 60b, simplexní virtuální soket 70 a označí zachycovací funkci na straně klienta v aplikaci 80. Podobně pro následné události, kde je vytvářen reálný soket, manažer reálných soketů vytvoří reálné sokety 61b, 62b, 63b nebo 64b a simplexní virtuální sokety 71, 72, 73 nebo 74 a určí funkci CSI odpovídající vytvořeným reálným a virtuálním soketům, znázorněným jako bloky 81, 82, 83 nebo 84 na obrázku 12.

Činnost zachycovací funkce na straně klienta bude nyní popsána s odvoláním na reálný soket 60b, simplexní virtuální soket 70 a zachycovací funkci na straně klienta 80, znázorněnou na obrázku 12. Blok 325 na obrázku 14 odráží vytvoření zachycovací funkce na straně klienta 80. Po vytvoření čeká zachycovací funkce na straně klienta 80 na událost na simplexní virtuální soketu 70, jak je naznačeno v bloku 326. Tato operace čekání se vykonává prováděním virtuální funkce výběru, která je popsána obrázkem 16–4. Po potvrzení události je tato událost prozkoumána, jak je naznačeno v bloku 330. Jestliže událost je zavření virtuálního soketu, potom zachycovací funkce na straně klienta 80 smaže simplexní virtuální soket 70, jak je naznačeno v bloku 349, a ukončí se, jak je naznačeno v bloku 350 na obrázku 14.

Jestliže událost je příjem dat, potom se uskuteční cesta od bloku 330 k bloku 331 a zachycovací funkce na straně klienta 80 přijme komunikaci vytvořenou browserem od simplexního virtuálního soketu 70, a to uskutečněním virtuální operace příjmu popsanou v tomto materiálu s odvoláním na obrázek 16–3. Zachycovací funkce na straně klienta potom provede výše popsanou funkci zachycovacího modulu na straně klienta (viz např. obrázky 3 a 7), která je znázorněna v bloku 332. Zachycovací funkce na straně klienta 80 potom vytvoří multiplexní virtuální soket 90, který je spojen s reálným soketem 36a zachycovacího modulu na straně klienta 30. Reálný soket 36a je spojen s reálným soketem 36b zachycovacího modulu na straně sereveru 40. Vytvořením multiplexního virtuálního soketu je znázorněno v bloku 333 na obrázku 14 a je uskutečňováno prováděním virtuální operace vytvoření popsané v tomto materiálu s odvoláním na obrázek 16–1. Blok 334 znázorňuje operaci odeslání informace přijaté od web browseru přes reálný soket 60b a simplexní virtuální soket 70 po provedení zachycovací funkce na straně klienta 80 pro komunikaci vytvořenou web browserem. Tato komunikace je řazena do formy na multiplexní virtuální soket 90, a to provedením virtuální operace odeslání popisované v tomto materiálu s odvoláním na obrázek 16–2. Zachycovací funkce na straně klienta 80 po zařazení do fronty požadavku na multiplexní virtuální soket 90 zarovná data ve frontě na multiplexní virtuálním soketu 90, jak je znázorněno v bloku 335 na obrázku 14, a potom počká na událost na multiplexním virtuálním soketu, jak je naznačeno blokem 336. Funkce virtuálního zarovnání se uskutečňuje provedením virtuální funkce zarovnání popsané v tomto materiálu vzhledem k obrázku 17–1, která převezme data z multiplexované virtuální soketové fronty a předá je na reálný soket 36a. Operace čekáním může být uskutečněna prováděním virtuální funkce výběru popsané obrázkem 16–4. V tomto okamžiku zachycovací modul na straně klienta komunikaci vytvořenou web browserem a tuto komunikaci přenesl na zachycovací modul na straně serveru přes externí komunikační spoj 35.

Zpět k obrázku 13, který znázorňuje diagram funkce manažeru soketu buď na zachycovacím modulu na straně serveru 40, nebo na zachycovacím modulu na straně klienta 30. Manažer reálných soketů v zachycovacím modulu na straně serveru nebo manažer serverových soketů, znázorněný jako blok 69 na obrázku 12, provádí stejnou funkci jako manažer klientových soketů, znázorněný jako blok 68. Při vytváření prvního reálného soketu, znázorněného v bloku 301, zachycovací modul na straně serveru 40 „dobře známý port“ 37b pro příjem požadavků na sokety od zachycovacího modulu na straně klienta 30 spojeného se zachycovacím modulem na straně serveru 40. Vyskytne-li se reálná událost na reálném soketu 36b u zachycovacího modulu na straně serveru 40, je tato událost prozkoumána, jak je naznačeno v bloku 305. V tomto případě je událostí příjem dat od reálného soketu 36a, proto se uplatní cesta od bloku 305 k bloku 320, znázorněná na obrázku 13. Data přijatá na reálný soket 36b jsou prozkoumána a v tomto konkrétním případě, protože se jedná o data vytvořená web browserem odeslána zachycovacím modulem na straně klienta, musí být vytvořen nový virtuální soket v zachycovacím modulu na straně serveru 40. To znamená, že se uplatní cesta od bloku 320 k bloku 321 na obrázku 13. manažer serverových soketů 69 potom provede operace naznačené v blocích 321, 322, 323 a 324 na obrázku 13. Manažer serverových soketů 69 vytvoří multiplexní virtuální soket 95, jak je naznačeno v bloku 321, zruší časovač aktivity multiplexního soketu, jak je naznačeno v bloku 322, a iniciuje aplikaci zachycovací funkce na straně serveru, jak je naznačeno v bloku 322 na obrázku 13 a znázorněno jako blok 85 na obrázku 12. Data přijatá na reálný soket 36 jsou potom zařazena do formy na multiplexní virtuální soket 95 a je signalizována virtuální událost.

Vytvoření zachycovací funkce na straně serveru, jak je znázorněno v bloku 323, je označeno jako blok 360 na obrázku 15. Po vytvoření zachycovací funkce na straně serveru 85 funkce přijme data z multiplexního virtuálního soketu 95, která byla odeslána od zachycovacího modulu na straně klienta 30 a odpovídají komunikaci vytvořené web browserem. Tato operace je označena jako blok 361 na obrázku 15. Po přijetí dat od zachycovacího modulu na straně klienta zpracuje zachycovací funkce na straně serveru 85 data výše popsaným způsobem pro zachycovací modul na straně serveru. provádění funkcí na straně serveru je označeno v bloku 362 (viz příklad na obrázcích 5 a 9). Po zpracování informací vytvoří zachycovací funkce na straně serveru 85 simplexní virtuální soket 75, a to prováděním virtuálního vytvoření, tj. operace, která je v tomto

materiálu popisována vzhledem k obrázku 16–1. Tato operace je naznačena blokem 363 na obrázku 15. Zachycovací funkce na straně serveru 85 potom odešle komunikaci vytvořenou web browserem na simplexní virtuální soket 75, jak je naznačeno v bloku 364, uskutečněním virtuálního odeslání, tj. operace, která je v tomto materiálu popsána s odvoláním na obrázek 16-2. Zachycovací funkce na straně serveru 85 potom provede virtuální zarovnání dat ve formě na simplexním virtuálním soketu 75 pro reálný soket 60c a počká na událost na simplexním virtuálním soketu 75. Operace virtuálního zarovnání je v tomto materiálu popsána vzhledem k obrázku 17–1. Operace odeslání a zarovnání jsou zobrazeny v blocích 364 a 365 na obrázku 15. Operace čekání může být uskutečněna prováděním funkce virtuálního výběru popsané na obrázku 16–4. Jestliže zachycovací funkce na straně serveru 85 vytvořila simplexní soket 75, byl vytvořen rovněž odpovídající reálný soket 60c. Odesláním komunikace vytvořené web browserem na simplexní virtuální soket 75 přenese zachycovací funkce na straně serveru 85 komunikaci vytvořenou web browserem a web server.

Když zachycovací modul na straně serveru 40 přijme odpověď od web serveru na reálný soket 60c, dojde k reálné události a manažer serverových soketů 69 ukončí blok 302 na obrázku 13 a prozkoumá událost, která nastala na reálném soketu 60c, jak je naznačeno blokem 305. V konkrétním případě se jedná o data pro existující virtuální soket a cestu od bloku 320 na blok 324 na obrázku 13. Data přijatá na reálný soket 60c jsou zařazena do formy na virtuálním soketu 75 a je signalizována virtuální událost. Je–li signalizována virtuální událost, ukončí zachycovací funkce na straně serveru 85 blok 366 na obrázku 15 a prozkoumá událost, jak je naznačeno blokem 370. Jestliže událostí je zavření soketu, nastane chybová podmínka a je jako odpověď zkonstruováno chybové hlášení, jak je naznačeno v bloku 375 na obrázku 15. Jestliže však je událostí příjem dat, uplatní se cesta od bloku 370 k bloku 371 a zachycovací funkce na straně serveru 85 provede virtuální příjem, jak je v tomto materiálu popisováno s odvoláním na obrázek 16–3, aby tak získala odpověď serveru ze simplexního virtuálního soketu 75, jak je zřejmé z bloku 371. Zachycovací funkce na straně serveru 85 potom provede virtuální zavření simplexního virtuálního soketu 75, jak je naznačeno blokem 372 a v tomto materiálu popisováno s odvoláním na obrázek 17–2, a zpracuje odpověď, jak je výše popisováno, pro zachycovací modul na straně serveru, a jak je znázorněno v bloku 373 (viz například obrázky 6 a 10).

Je–li ukončovací cesta bloku 370 na obrázku 15 chybovou cestou k bloku 375 nebo datovou cestou k bloku 371, je v bloku 374 smazán simplexní virtuální soket 75. Zachycovací funkce na straně severu potom provede operaci virtuálního odeslání na multiplexní virtuální soket 95 s cílem odeslat komunikaci vytvořenou web serverem na zachycovací modul na straně klienta 30, jak je znázorněno v bloku 376. Zachycovací funkce na straně serveru 85 potom uskuteční operaci virtuálního zarovnání dat ve frontě v multiplexním virtuálním soketu 95. Tyto operace jsou znázorněny v bloku 377. Zachycovací funkce na straně serveru 85 potom provede operaci virtuálního zavření, při které zavře multiplexní virtuální soket 95, jak je znázorněno v bloku 378 na obrázku 15. Zachycovací funkce na straně serveru 85 nakonec smaže multiplexní virtuální soket a ukončí se, jak je znázorněno bloky 379 a 380.

Zachycovací funkce na straně serveru provede operace virtuálního odeslání a zarovnání u multiplexního virtuálního soketu 95. Tímto se spustí události na reálném soketu 36a, přičemž manažer klientových soketů 68 ukončí blok 302 a prozkoumá událost, jak je znázorněno v bloku 305, a protože data byla přijata na reálný soket 36a, bude použita cesta od bloku 305 k bloku 320 na obrázku 13 a data budou zařazena do fronty na multiplexní virtuální soket 90. Když tedy reálný soket 36a přijme odpověď web serveru od reálného soketu 36b přes externí komunikační spoj 35, jsou tato data demultiplexována a poskytnuta na příslušný multiplexní virtuální soket. Příjem dat způsobí vznik virtuální události, která by měla být, jak je ukázáno blokem 324 na obrázku 13 a blokem 336 na obrázku 14, ukončena a zachycovací funkce na straně klienta 80 by měla tuto událost prozkoumat, jak je znázorněno v bloku 340 na obrázku 14.

Jestliže událost je odezva na uzavřený soket, uplatní se cesta od bloku 340 k bloku 345 na obrázku 14 a zachycovací funkce na straně klienta 80 vytvoří odpověď chybového hlášení

a předá ji bloku 344 na obrázku 14. Pokud událost jsou přijatá data, což je právě uváděný příklad, uplatní se cesta od bloku 340 k bloku 341 na obrázku 14 a zachycovací funkce na straně klienta 80 provede operaci virtuálního příjmu k přijetí odpovědi z multiplexního virtuálního soketu 90. Tato operace příjmu je znázorněna blokem 341 na obrázku 14. Po přijetí dat z multiplexního virtuálního soketu 90 provede zachycovací funkce na straně klienta 80 operaci virtuálního uzavření a zavře multiplexní virtuální soket 90, jak je naznačeno blokem 342. Zachycovací funkce na straně klienta 80 potom zpracuje odpověď výše popsaným způsobem pro zachycovací modul na straně klienta, jak je znázorněno v bloku 343 (viz např. obrázky 4 a 8).

Operace bloku 344 jsou potom prováděny při použití kterékoli cesty k ukončení bloku 340. Zachycovací funkce na straně klienta 80 smaže multiplexní virtuální soket, jak je naznačeno blokem 344, a potom provede operaci virtuálního odeslání s cílem odeslat odpověď na browser přes simplexní virtuální soket 70, jak je naznačeno blokem 346. Po dokončení operace virtuálního odeslání provede zachycovací funkce na straně klienta 80 operaci virtuálního zarovnání tak, aby zarovnala data ve frontě simplexního virtuálního soketu, jak je znázorněno blokem 347 pro reálný soket 60b, a potom provede operaci virtuálního zavření s cílem zavřít simplexní virtuální soket, jak je znázorněno v bloku 348. Po zavření simplexního virtuálního soketu pro zachycovací funkci na straně klienta je smazán simplexní virtuální soket a zachycovací funkce na straně klienta se ukončí, jak je zřejmé z bloku 349 a 350 na obrázku 14.

Z pohledu hodnocení stávajícího vynálezu, je tento popisován s ohledem na jeden konkrétní příklad vytvoření simplexního a multiplexního virtuálního soketu a zachycovací funkcí na straně klienta a na straně serveru, avšak pluralita funkcí může být vytvořena v rámci jednoho zachycovacího modulu na straně klienta, respektive zachycovacího modulu na straně serveru. Obdobně může zachycovací modul na straně klienta a zachycovací modul na straně serveru, podle stávajícího vynálezu, vytvořit TCP/IP spojení mezi zachycovacím modulem na straně klienta 30 a zachycovacím modulem na straně serveru 40 a potom provádět multiplex na pluralitě spojení TCP/IP komunikací vytvořených web browserem nebo web serverem, a to při udržení spojení TCP/IP.

Zbývající funkce manažera klientových soketů a manažera serverových soketů lze nejlépe pochopit s použitím obrázků 16–1 až 16–4 a obrázku 17–1 a 17–2, které popisují operace prováděné zachycovacím modulem na straně klienta a zachycovacím modulem na straně serveru při uskutečnění operací virtuálního vytváření, virtuálního odeslání, virtuálního příjmu, virtuálního výběru, virtuáolního zarovnání nebo virtuálního zavření, jak je znázorněno vývojovými diagramy na obrázcích 14 a 15. Když se uskutečňuje operace virtuálního vytváření, jak je známo z bloku 333 na obrázku 14 a bloku 363 na obrázku 15, začíná se od bloku 400 na obrázku 16–1. Manažer soketů potom určí, zda je vyžadován reálný soket, jak je znázorněno v bloku 405. Jestliže již reálný soket existuje, anebo při vytváření multiplexního virtuálního soketu, který má být připojen na stávající reálný soket, uplatní se cesta „No“ bloku 405 a virtuální soket se spojí s reálným soketem, jak je zřejmé z bloku 409. Jestliže je však reálný soket zapotřebí, použije se potom cesta „Yes“ bloku 405. Jak je zřejmé z bloku 406, vytvoří se reálný soket. Reálný soket je potom přidán do seznamu událostí, jak je naznačeno v bloku 408, pro potřeby monitorování, jak je reflektováno v bloku 302 na obrázku 13. Po vytvoření reálného soketu a ustanovení spojení je virtuální soket připojen k reálnému soketu, jak je uvedeno v bloku 409, a dokončí se operace vytvoření, jak je zřejmé z bloku 410.

Při provádění operace virtuálního odeslání naznačené v blocích 334 a 346 na obrázku 14 nebo blocích 364 a 376 na obrázku 15 začínají operace od bloku 420 na obrázku 16–2. Do fronty virtuálního soketu jsou přidána data, jak je zřejmé z bloku 427, a potom se ukončí operace odeslání, jak je vidět z bloku 428.

Operace virtuálního příjmu znázorněné v blocích 331 a 341 na obrázku 14 a blocích 361 a 371 na obrázku 15 jsou uskutečňovány prováděním operací začínajících od bloku 430 na obrázku 16–3. Jak je vidět v bloku 435, je vyhodnocena virtuální soketová fronta, aby se stanovilo, zda jsou ve

virtuální soketové frontě nějaká data. Jestliže se ve virtuální soketové frontě data vyskytují, uplatní se cesta „Yes“ bloku 435 a tato data jsou vrácena na funkci zvanou příjmová operace, jak je patrno z bloku 436. Jestliže ve virtuální soketové frontě nejsou žádná data, avšak soket není označen pro zavření, uplatní se cesta „No“ rozhodovacího bloku 440 a nic se nevrátí, jak je zřejmé z bloku 441. Jestliže však nejsou ve frontě žádná data a soket je označen pro zavření, platí cesta „Yes“ bloku 440 a soket je označen jako zavřený, jak je zřejmé z bloku 442, a na operaci vyžadující příjem je vrácena informace, že je soket zavřený, jak je zřejmé z bloku 443.

Operace virtuálního výběru vykonává v blocích 326 a 336 na obrázku 14 a v bloku 366 na obrázku 15 se uskutečňuje provedením operací, které začínají blokem 445 na obrázku 16–4. Jak je zřejmé z bloku 446, nejdříve je stanoveno, zda data nebo operace virtuálního zavření nejsou dosud na zvoleném virtuálním soketu vyřízená. V případě, že nejsou vyřízená žádná data nebo virtuální zavření, uplatní se cesta „No“ bloku 446 a proces čeká na virtuální událost na zvoleném virtuálním soketu, jak je znázorněno v bloku 447, a po přijetí takové události se ukončí, jak je znázorněno v bloku 448. Jestliže však data a virtuální zavření jsou dosud nevyřízená a čekají na vyřízení, čekají na zvoleném virtuálním soketu, virtuální událost se již vyskytla, a proto se uplatní cesta „Yes“ bloku 446, proces se zakončí, jak je znázorněno v bloku 448.

Operace virtuálního zarovnání, jak je zřejmé z bloků 335 a 347 na obrázku 14 a bloků 365 a 377 na obrázku 15 se uskutečňuje provedením operací, které začínají blokem 450 na obrázku 17–1. Je–li operace virtuálního zarovnání spouštěna, stanoví, zda jsou ve frontě virtuálního soketu nějaká data, která mají být zarovnána, jak je naznačeno v rozhodovacím bloku 455. Jestliže ve virtuální soketové frontě nejsou žádná data, potom se operace zarovnání prostě ukončí a vrátí se do funkce, jež ji spustila, jak je zřejmé z větve „No“ bloku 455. Jestliže však jsou ve frontě nějaká data, potom se uplatní cesta „Yes“ bloku 455 a je určeno, zda virtuální soketová fronta je pro multiplexní soket, jak je patrno z bloku 460. Jedná–li se o multiplexní soket, potom se soketové záhlaví, které se skládá ze tří byte dat odrážejících jednoznačný identifikátor pro soket a z objemu dat pro přenos, přidá do vyrovnávací paměti (bufferu) reálného soketu, jak je znázorněno v bloku 461. V obou případech, jestliže se jedná o multiplexní soket nebo simplexní soket, se data pro reálný soket potom přesunou do vyrovnávací paměti pro reálný soket, jak je naznačeno v bloku 462. V případě, že vyrovnávací paměť reálného soketu je plná, uplatní se větev „Yes“ bloku 465 a data se z vyrovnávací paměti pro reálný soket odešlou na tento reálný soket, jak je naznačeno v bloku 466. Pokud však vyrovnávací paměť pro reálný soket není plná, platí cesta „No“ bloku 465. Virtuální zarovnávací funkce potom provede test s cílem určit, zda jsou ještě nějaká další data na případné další frontě multiplexního virtuálního soketu, která by měla být odeslána na reálný soket. Pokud odpověď zní ano, uplatní se cesta „Yes“ bloku 470 a data ve vyrovnávací paměti reálného soketu nejsou odeslána, dokud není operace virtuálního zarovnání znovu vyvolána, aby provedla zarovnání jedné z ostatních virtuálních soketových front. Pokud však nejsou již žádná další data nebo po přidání z ostatních multiplexních virtuálních soketů, provede se operace bloku 466 a data ve vyrovnávací paměti reálného soketu jsou odeslána na reálný soket. Po tom všem se data ve virtuální soketové frontě odpovídající funkci, která volala operaci virtuálního zarovnání, odešlou na reálný soket a operace virtuálního zarovnání se ukončím, jak je znázorněno v bloku 467.

Virtuální operace zavření znázorněná v blocích 342 a 348 na obrázku 14 a blocích 372 a 378 na obrázku 15 se uskutečňuje provedením operací, které začínají blokem 480 na obrázku 17–2. Když je volána operace virtuálního zavření, nejdříve provede testy s cílem určit, zda se virtuální zavření týká multiplexního virtuálního soketu, jak je znázorněno v bloku 485. Je–li to multiplexní virtuální soket, potom se uplatní cesta „Yes“ bloku 485 a do virtuální soketové fronty se přidá indikátor operace „zavření“. Ať již jde o virtuální zavření multiplexního virtuálního soketu či nikoli, operace virtuálního zavření zavolá operaci virtuálního zarovnávání, jak je znázorněno v bloku 487 a potom provede odpojení od reálného soketu, jak je znázorněno v bloku 488. Operace potom provede testy s cílem poznat, zda virtuální zavření se týká simplexního virtuálního soketu, jak je zřejmé z bloku 490; a pokud tomu tak není, uplatní se cesta „No“ bloku 495 s cílem stanovit, zda se jedná o poslední multiplexní virtuální soket, a je–li tomu tak, nastaví

se časovač multiplexní aktivity, jak je zřejmé z bloku 496. Pokud se však nejedná o poslední multiplexní virtuální soket, potom se blok 496 přeskočí.

Zpět k bloku 490. Jestliže se virtuální zavření týká simplexního virtuálního soketu, potom je příslušný reálný soket odstraněn ze seznamu událostí, jak je znázorněno v bloku 491 a reálný soket je zavřen a smazán, jak je zřejmé z bloku 492. Ať již je soketem simplexní nebo multiplexní virtuální soket, je označen jako zavřený, viz blok 497, a operace zavření se ukončí, viz blok 498. Obrázek 13 bude nyní popsán ve vztahu k obrázkům 16–1 až 16–4 a obrázkům 17–1 a 17–2. Když se vyskytne reálná událost, blok 302 na obrázku 13 se ukončí a manažer soketů událost prozkoumá na základě toho, jak byla tato událost generována. Jestliže byl u události překročen časový interval (timing) pro časovač aktivity multiplexního soketu, který byl stanoven v bloku 496 na obrázku 17–2, potom se uplatní cesta od bloku 305 k bloku 312 na obrázku 13. Jak je zřejmé z obrázku 13, provede manažer soketů operace bloku 312 a bloku 313 pro zavření multiplexního reálného soketu a smaže multiplexní reálný soket, který odpovídá soketu, jenž spojuje zachycovací modul na straně klienta se zachycovacím modulem na straně serveru. Manažer soketů potom vyčká na další reálnou událost. tento časovač multiplexních událostí je resetován (nulován) vytvořením multiplexního virtuálního soketu, jak je vidět v bloku 322.

Jestliže událost vyskytující se na reálném soketu je zavření reálného soketu, např. web server provádějící operaci zavření na soketových spojeních mezi web serverem a zachycovacím modulem na straně serveru, potom se uplatní cesta od bloku 305 k bloku 309 na obrázku 13. Manažer soketů odstraní reálný soket ze seznamu reálných událostí, jak je zřejmé z bloku 309 a odpojí virtuální soket nebo sokety, v případě několika multiplexních soketů, od reálného soketu nebo soketů, jak je vidět z bloku 310. Manažer soketů potom označí virtuální soket pro uzavření a oznámí (signalizuje) virtuální událost. Tato operace je reflektována v bloku 311 a jakmile jsou všechna data z virtuální soketové fronty odstraněna, bude virtuální soket zavřen. Po označení virtuálního soketu pro zavření manažer soketů následně stanoví, zda je nebo není reálný soket, který má být zavřen, simplexním soketem, jak je naznačeno v rozhodovacím bloku 315. Jestliže zavření reálného soketu se týká simplexního soketu, je tento reálný soket zavřen a smazán, jak je znázorněno v bloku 316. Manažer soketů potom čeká na další reálnou událost, jak je patrné z bloku 302.

Není–li to simplexní reálný soket, jenž se má zavřít, potom se uplatní cesta „No“ bloku 315 a manažer soketů vyčká na následnou reálnou událost. Multiplexní reálný soket nebo soket spojující zachycovací modul na straně klienta a zachycovací modul na straně serveru může být pouze zavřen s použitím časového hlídání (timeout) ze strany časovače aktivity multiplexního soketu. To umožňuje údržbu spojení mezi zachycovacím modulem na straně klienta a zachycovacím modulem na straně serveru i poté, co proběhla poslední komunikace mezi moduly v rámci pro uživatele specifikovaného předem stanoveného času. V situaci následného požadavku spojení od browseru před vypršením časového intervalu určeného časovačem aktivity multiplexního soketu by mohla být komunikace uskutečněna bez znovuustanovení spojení mezi zachycovacím modulem na straně klienta a zachycovacím modulem na straně serveru, a tak eliminována potřeba dalších nároků v důsledku znovuustanovení takového spojení.

Poslední větev, kterou je nutné popsat u obrázku 13, je větev, kdy nastane reálná událost a touto událostí je příjem dat na multiplexní reálný soket nebo reálné sokety 36a nebo 36b na obrázku 12. Jsou–li data přijata na multiplexní reálné sokety, jsou tato data prozkoumána a v případě, že zahrnují indikátor operace zavření, podobně jako tomu bylo u virtuální fronty v bloku 486 na obrázku 17–2, potom se provede operace virtuálního zavření, a tedy použije cesta od bloku 320 k bloku 310. Manažer soketů odpojí od reálného soketu multiplexní virtuální soket označený v přijatých datech na reálný soket, viz blok 310, a potom označí virtuální soket pro „zavření“ a signalizuje virtuální událost tak, jak je znázorněno v bloku 311. Protože zavření je vlastně zavření multiplexního virtuálního soketu, uplatní se cesta „No“ bloku 315 a potom manažer soketů počká na reálnou událost, jak je vidět v bloku 302.

Prostřednictvím provádění operací popsaných na obrázcích 13 až 17 konkrétní aspekt stávajícího vynálezu zustanovuje stálé spojení mezi prvním počítačem a druhým počítačem přes externí komunikační spoj. Toto stálé spojení je udržováno do doby, než všechny komunikace vytvořené web browserem jsou dokončeny a pluralita komunikací vytvořených web browserem je zachycena a následně multiplexována na externí komunikační spoj v obě, kdy je stálé spojení udržováno. Specifický datový tok klient/server může být pak demultiplexován tak, aby se vytvořila pluralita datových toků HTTP a tato pluralita datových toků HTTP je předána na web server. Stálé spojení je rovněž udržováno do dokončení všech komunikací vytvořených web serverem. Pluralita komunikací vytvořených web serverem je zachycena a multiplexována na externí komunikační spoj v době, kdy je stálé spojení udržováno. Specifický datový tok klient/server může být potom demultiplexován tak, aby se vytvořila pluralita datových toků HTTP a tato pluralita datových toků HTTP se předala na web server.

Ve výkresové dokumentaci i specifikacích jsou zapracována typická preferovaná začlenění vynálezu, a přestože jsou použity specifické termíny, jsou použity pouze v generickém a popisovém smyslu slova a nikoli pro účely omezení, rozsah vynálezu je vyložen v dále uvedených nárocích.

PATENTOVÉ NÁROKY

1. Způsob zachycování dat přijatých od druhé aplikace, která mají být poskytnuta první aplikaci v odpovědi na požadavek od první aplikace, **vyznačující se tím**, že zahrnuje následující kroky:

uloží se datový tok, který má být přijat od druhé aplikace a který má být poskytnut první aplikaci, do první cache paměti (31) tak, aby se vytvořil klientův cache zápis (32), odpovídající požadavku první aplikace,

uloží se čas vytvoření klientova cache zápisu (32) tak, aby se vytvořil záznam času klientova cache zápisu (32),

uloží se čas vytvoření klientova cache zápisu (32) tak, aby se vytvořil záznam času klientova cache zápisu (32),

zachytí se požadavek od první aplikace, přičemž se stanoví, zda existuje klientův cache zápis (32), odpovídající požadavku,

vyhodnotí se záznam času klientova cache zápisu (32) pro klientův cache zápis (32), odpovídající požadavku od první aplikace, přičemž se stanoví, zda se vytvořil klientův cache zápis (32), odpovídající požadavku od první aplikace během předem definovaného klientova koherentního časového intervalu před vyžádáním informací první aplikace, dodá se klientův cache zápis (32) první aplikaci v odpovědi na požadavek, pokud ve zmíněném kroku vyhodnocení bylo určeno, že byl vytvořen klientův cache zápis (32) pro daný požadavek od první aplikace během předem definovaného klientova koherentního časového intervalu před vyžádáním informací první aplikace.

2. Způsob podle nároku 1, **vyznačující se tím**, že dále zahrnuje krok, při němž se udržují klientovy cache zápisy (32) u více instancí první aplikace.

3. Způsob podle nároku 1, **vyznačující se tím**, že první aplikace je rezidentní na prvním počítači (5), druhá aplikace je rezidentní na druhém počítači (6), přičemž první a druhá aplikace spolu navzájem komunikují přes externí komunikační spoj (35), přičemž se provádí následující kroky:

uloží se datový tok od druhé aplikace v odpovědi na požadavek první aplikace do druhé cache paměti (41) rezidentní na druhém počítači (6) tak, aby se vytvořil cache zápis (42) požadavku serveru (40),

zachytí se požadavek, vytvoření první aplikací a přijatý druhou aplikací, přičemž se stanoví, zda byl již dříve uložen v druhé cache paměti (41) cache zápis (42) požadavku serveru (40), odpovídající danému požadavku,

odešle se cache zápis (42) požadavku serveru (40), odpovídající danému požadavku od první aplikace, na první aplikaci rezidentní na prvním počítači (5) přes externí komunikační spoj (35).

4. Způsob podle nároku 3, **vyznačující se tím**, že dále zahrnuje kroky:

stanoví se, jestli se vytvořil cache zápis (42) požadavku serveru (40), odpovídající požadavku od první aplikace, během předem definovaného klientova koherentního časového intervalu před přijetím požadavku druhým počítačem (6),

přičemž, pokud, se při stanovení určí, že se cache zápis (42) požadavku serveru (40) vytvořil v rámci předem definovaného klientova koherentního časového intervalu, při odeslání se odešle cache zápis (42) požadavku serveru (40), odpovídající požadavku od první aplikace, první aplikaci.

5. Způsob podle nároku 3, **vyznačující se tím**, že dále zahrnuje kroky:

stanoví se, zda existuje klientův cache zápis (32), odpovídající požadavku od první aplikace, který je identický s cache zápisem (42) požadavku serveru (40), odpovídajícím danému požadavku,

vypočte se časový interval mezi okamžikem, kdy druhý počítač (6) přijal požadavek od první aplikace, a okamžikem kdy se vytvoří cache zápis (42) požadavku serveru (40), odpovídající požadavku od druhé aplikace, čímž se poskytnou data o stáří zápisu,

přičemž při odeslání se přenáší data o stáří zápisu pro cache zápis (42) požadavku serveru (40), odpovídající požadavku od první aplikace, na první aplikaci rezidentní na prvním počítači (5), a

aktualizuje se časový záznam klientova cache zápisu (32), odpovídajícího požadavku od první aplikace tak, že se od aktuálního času na prvním počítači (5) odečtou data o stáří zápisu přijatá od druhého počítače (6).

6. Způsob podle nároku 1, **vyznačující se tím**, že první aplikací je web browser (10) a druhou aplikací je web server (20).

7. Způsob podle nároku 3, **vyznačující se tím**, že druhá aplikace je web server (20), první aplikace je web browser (10), přičemž první a druhý počítač (5, 6) spolu komunikují přes bezdrátový komunikační spoj (35).

8. Zařízení pro zachycení dat přijatých od druhé aplikace pro poskytnutí první aplikaci v odpovědi na požadavek od první aplikace, **vyznačující se tím**, že sestává z prvního

zachycovacího modulu zapojeného mezi první aplikaci a druhou aplikaci, přičemž první zachycovací modul je upraven pro uložení datového toku, přijímaného od druhé aplikace a poskytovaného první aplikaci, do první cache paměti (31), připojené k prvnímu zachycovacímu modulu, rezidentnímu na prvním počítači (5) k vytvoření klientova cache zápisu (32), odpovídajícímu požadavku první aplikace, první zachycovací modul je dále upraven pro uložení času vytvoření klientova cache zápisu (32) pro vytvoření záznamu času klientova cache zápisu (32), první zachycovací modul je dále upraven pro zachycení požadavku od první aplikace pro stanovení existence klientova cache zápisu (32), odpovídajícímu požadavku, první zachycovací modul je dále upraven pro vyhodnocení záznamu času klientova cache zápisu (32) pro klientův cache zápis (32), odpovídající požadavku od první aplikace, pro stanovení, zda je vytvořen klientův cache zápis (32), odpovídající požadavku od první aplikace, během předem definovaného klientova koherentního časového intervalu před vyžádáním informací první aplikací, první zachycovací modul je dále upraven pro dodání klientova cache zápisu (32) první aplikaci v odpovědi na požadavek, pokud je určeno, že je vytvořen klientův cache zápis (32) pro požadavek od první aplikace během předem definovaného klientova koherentního časového intervalu před vyžádáním informací první aplikací.

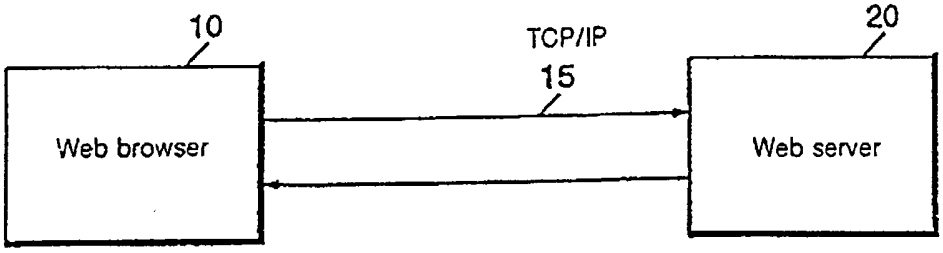
9. Zařízení podle nároku 8, **vyznačující se tím**, že druhá aplikace je rezidentní na druhém počítači (6)a první aplikace je rezidentní na prvním počítači (5), přičemž první a druhá aplikace jsou spojeny přes externí komunikační spoj (35), druhý zachycovací modul je připojen mezi první zachycovací modul a druhou aplikaci, přičemž druhý zachycovací modul je upraven pro uložení datového toku od druhé aplikace v odpovědi na požadavek první aplikace do druhé cache paměti (41), připojené k druhému zachycovacímu modulu, rezidentnímu na druhém počítači (6) pro vytvoření cache zápisu (42) požadavku serveru, druhý zachycovací modul je dále upraven pro zachycení požadavku od první aplikace, zda byl již dříve uložen v druhé cache paměti (41) cache zápis (42) požadavku serveru, odpovídajícímu danému požadavku a druhý zachycovací modul je dále upraven pro odesílání cache zápisu (42) požadavku serveru, odpovídajícího danému požadavku od první aplikace, na první aplikaci rezidentní na prvním počítači (5) přes externí komunikační spoj (35).

10. Počítačový programový produkt k provádění způsobu podle nároku 1, který je uložen na počítačem čitelné paměťové médium a zahrnuje programové kódové prostředky pro provedení v počítačovém systému a který je čitelný počítačem, **vyznačující se tím**, že zahrnuje:

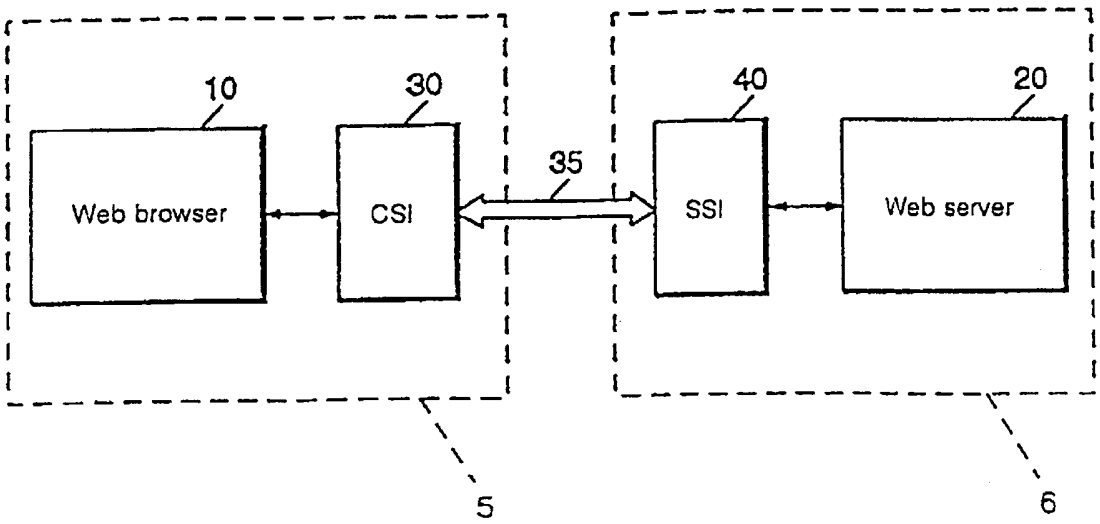
počítačem čitelné programové kódové prostředky pro vyvolání prvním počítačem (5) uložení datového toku, který má být přijat od druhé aplikace a který má být poskytnut první aplikaci, do první cache paměti (31) pro vytvoření klientova cache zápisu (32), odpovídajícího požadavku první aplikace, počítačem čitelné programové kódové prostředky pro vyvolání prvním počítačem (5) uložení času vytvoření klientova cache zápisu (32) tak, že se vytvoří záznam času klientova cache zápisu (32), počítačem čitelné programové kódové prostředky pro vyvolání prvním počítačem (5) zachycení požadavku od první aplikace pro stanovení, zda existuje klientův cache zápis (32) odpovídající požadavku, počítačem čitelné programové kódové prostředky pro vyvolání prvním počítačem (5) vyhodnocení záznamu času klientova cache zápisu (32) pro klientův cache zápis (32), odpovídající požadavku od první aplikace pro stanovení, zda byl vytvořen klientův cache zápis (32), odpovídající požadavku od první aplikace, během předem definovaného klientova koherentního časového intervalu před vyžádáním informací první aplikací, počítačem čitelné programové kódové prostředky pro vyvolání prvním počítačem (5) dodání klientova cache zápisu (32) první aplikaci v odpovědi na požadavek, pokud je v kroku vyhodnocení určeno, že byl vytvořen klientův cache zápis (32) pro daný požadavek od první aplikace během předem definovaného klientova koherentního časového intervalu před vyžádáním informací první aplikací.

19 výkresů

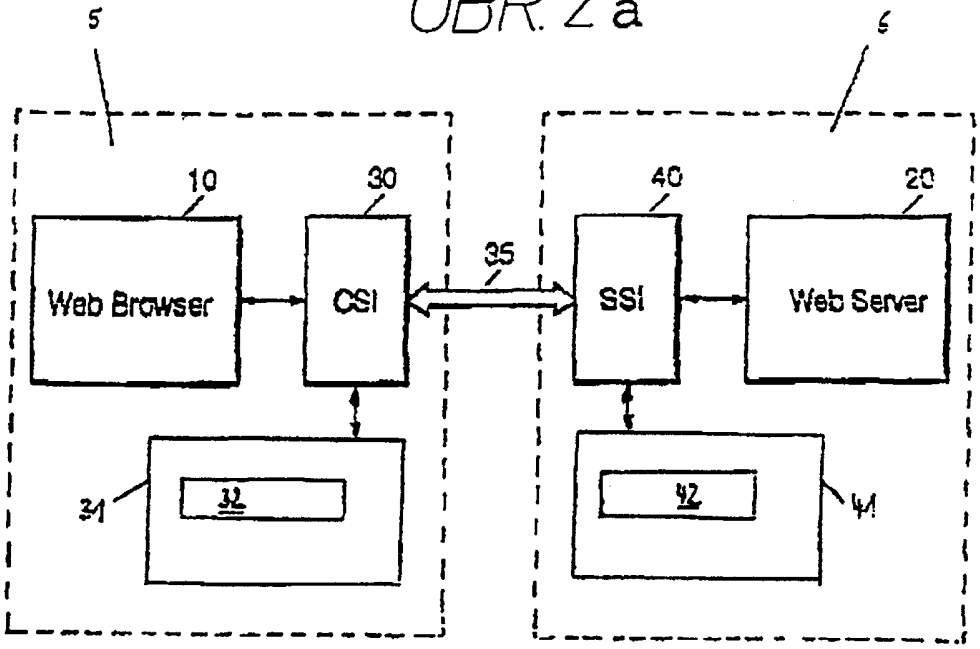
Obr. 1



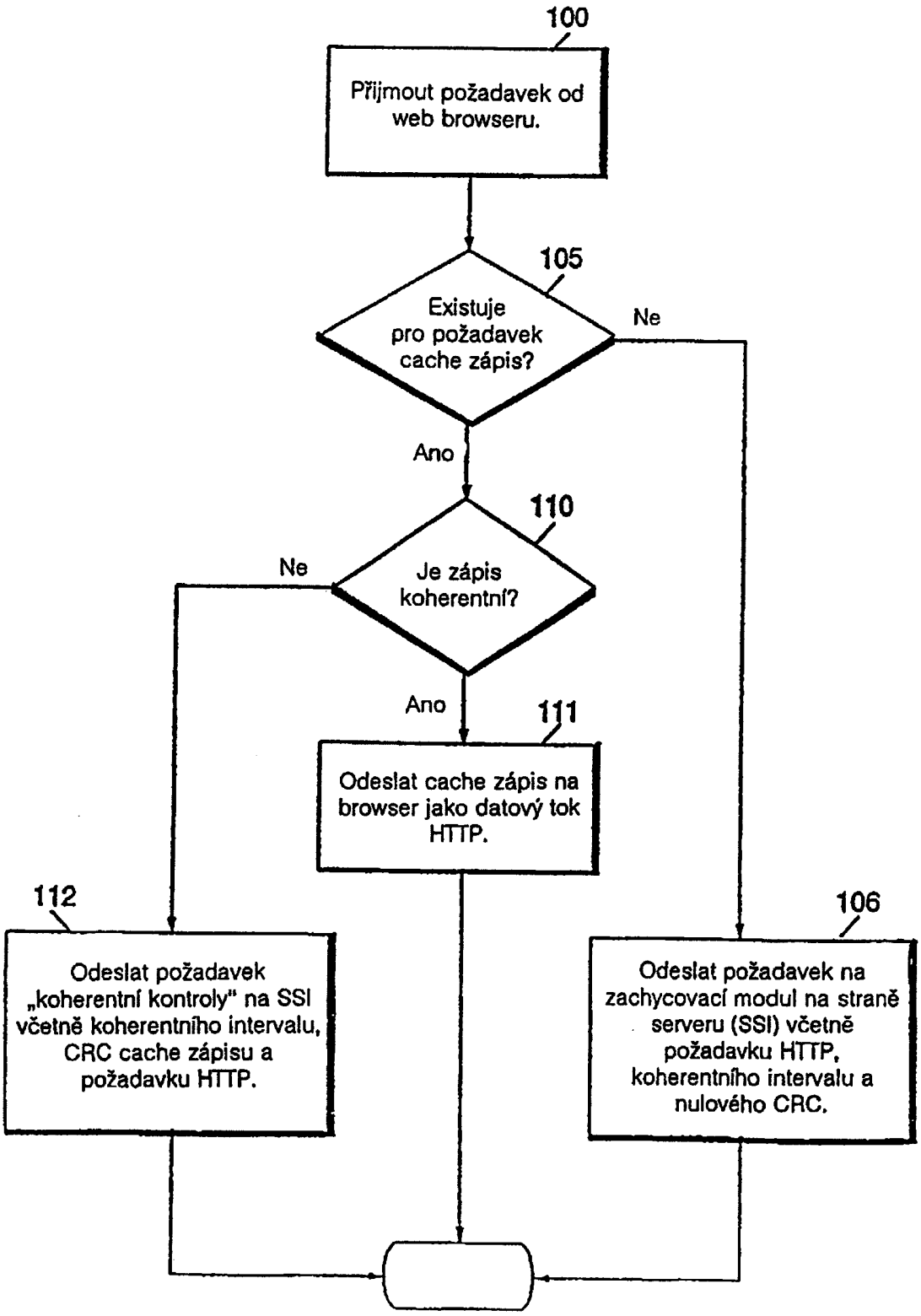
Obr. 2



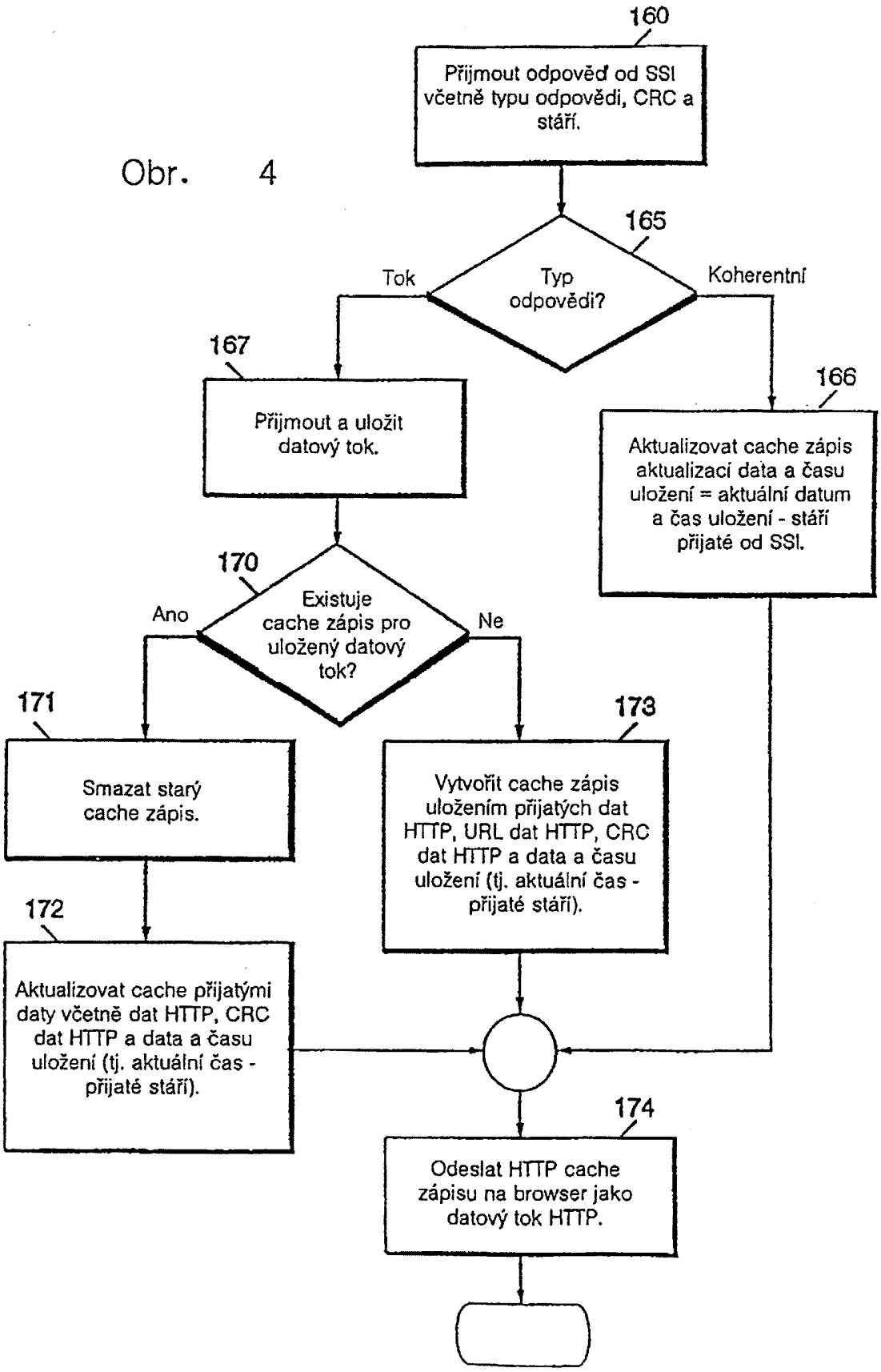
OBR. 2a

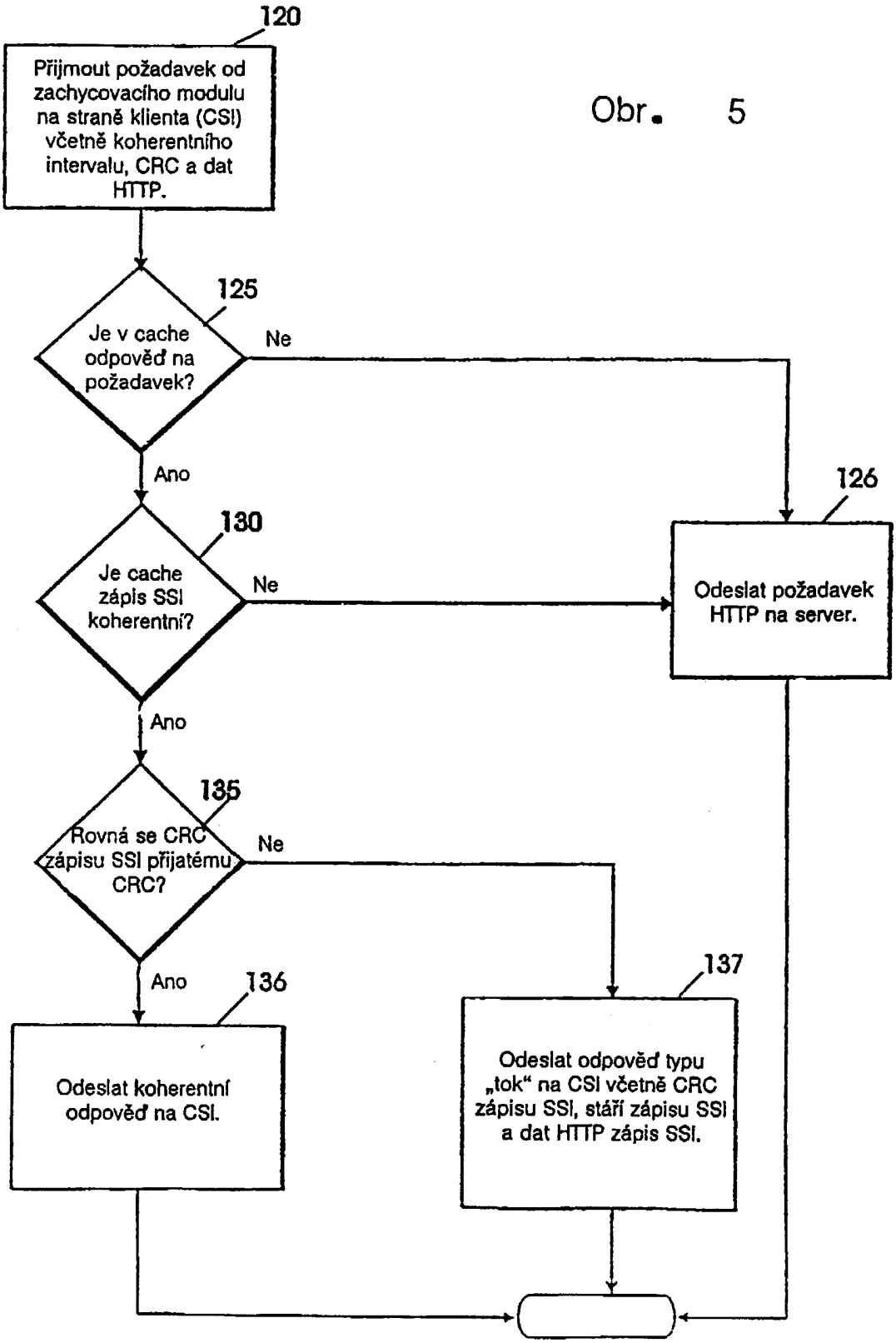


Obr. 3

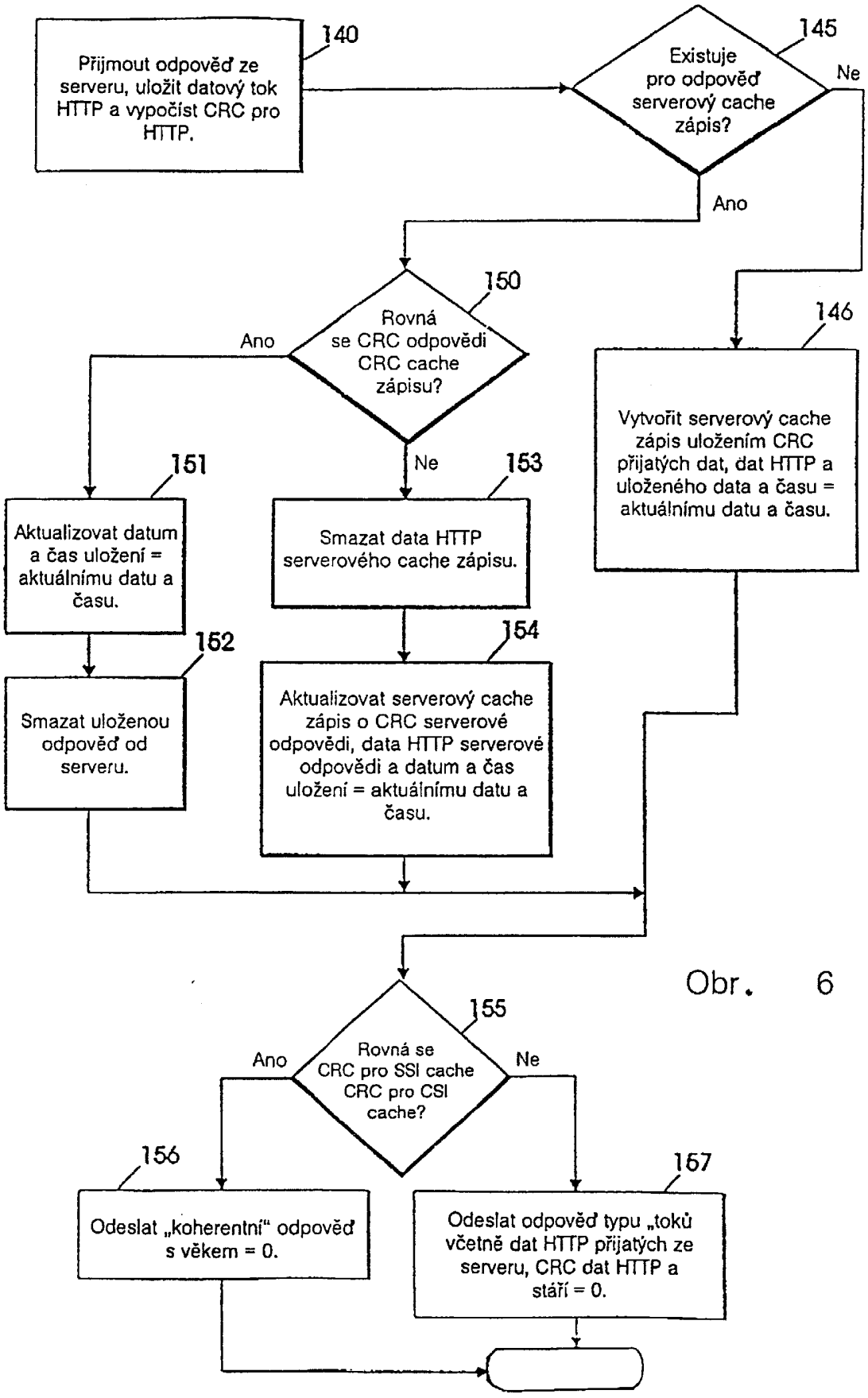

100
Přijmout požadavek od web browseru.
105
Existuje pro požadavek cache zápis?
Ne
Ano
110
Je zápis koherentní?
Ne
Ano
111
Odeslat cache zápis na browser jako datový tok HTTP.
112
Odeslat požadavek „koherentní kontroly" na SSI včetně koherentního intervalu, CRC cache zápisu a požadavku HTTP.
106
Odeslat požadavek na zachycovací modul na straně serveru (SSI) včetně požadavku HTTP, koherentního intervalu a nulového CRC.

Obr. 4


160
Přijmout odpověď od SSI včetně typu odpovědi, CRC a stáří.
165
Typ odpovědi?
Tok
Koherentní
167
Přijmout a uložit datový tok.
166
Aktualizovat cache zápis aktualizací data a času uložení = aktuální datum a čas uložení - stáří přijaté od SSI.
170
Existuje cache zápis pro uložený datový tok?
Ano
Ne
171
Smazat starý cache zápis.
173
Vytvořit cache zápis uložením přijatých dat HTTP, URL dat HTTP, CRC dat HTTP a data a času uložení (tj. aktuální čas - přijaté stáří).
172
Aktualizovat cache přijatými daty včetně dat HTTP, CRC dat HTTP a data a času uložení (tj. aktuální čas - přijaté stáří).
174
Odeslat HTTP cache zápisu na browser jako datový tok HTTP.

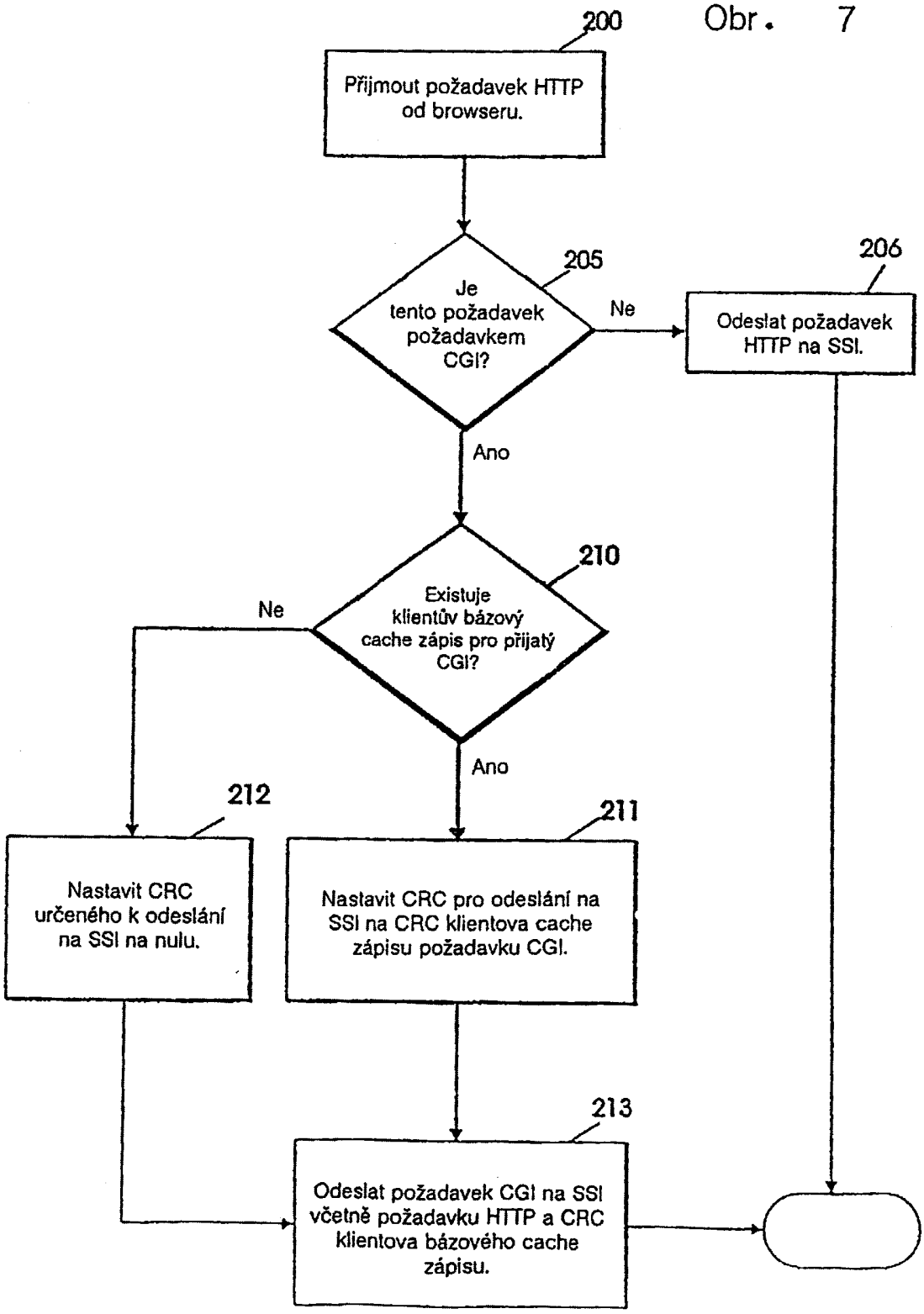

120
Přijmout požadavek od zachycovacího modulu na straně klienta (CSI) včetně koherentního intervalu, CRC a dat HTTP.
125
Je v cache odpověď na požadavek?
Ne
Ano
126
130
Je cache zápis SSI koherentní?
Ne
Odeslat požadavek HTTP na server.
Ano
135
Rovná se CRC zápisu SSI přijatému CRC?
Ne
Ano
136
137
Odeslat koherentní odpověď na CSI.
Odeslat odpověď typu „tok“ na CSI včetně CRC zápisu SSI, stáří zápisu SSI a dat HTTP zápis SSI.

Obr. 5

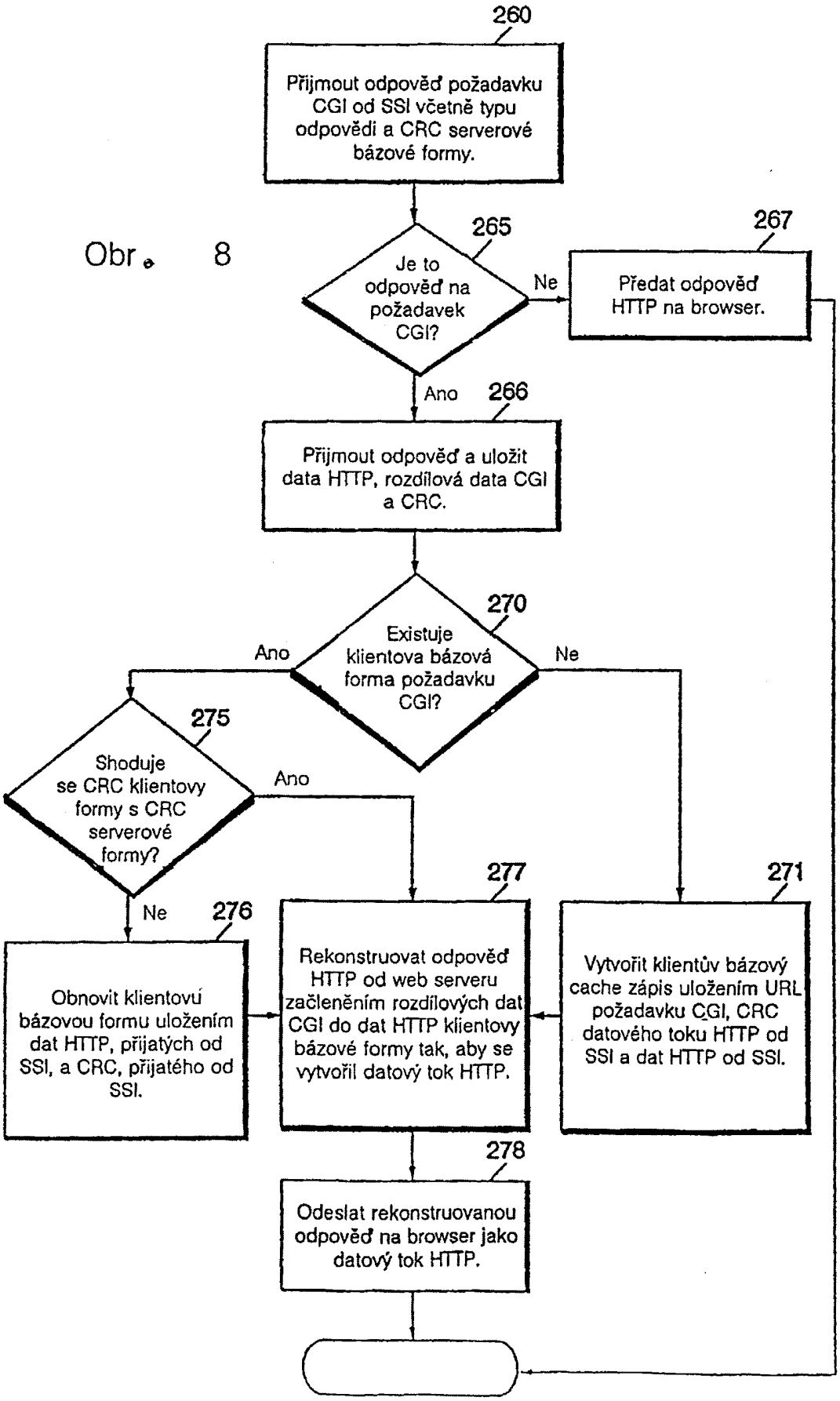

Přijmout odpověď ze serveru, uložit datový tok HTTP a vypočíst CRC pro HTTP.
140
Existuje pro odpověď serverový cache zápis?
145
Ne
Ano
Rovná se CRC odpovědi CRC cache zápisu?
150
Ano
Ne
146
Vytvořit serverový cache zápis uložením CRC přijatých dat, dat HTTP a uloženého data a času = aktuálnímu datu a času.
151
Aktualizovat datum a čas uložení = aktuálnímu datu a času.
153
Smazat data HTTP serverového cache zápisu.
152
Smazat uloženou odpověď od serveru.
154
Aktualizovat serverový cache zápis o CRC serverové odpovědi, data HTTP serverové odpovědi a datum a čas uložení = aktuálnímu datu a času.
155
Rovná se CRC pro SSI cache CRC pro CSI cache?
Ano
Ne
156
Odeslat „koherentní“ odpověď s věkem = 0.
157
Odeslat odpověď typu „toků včetně dat HTTP přijatých ze serveru, CRC dat HTTP a stáří = 0.

Obr. 6

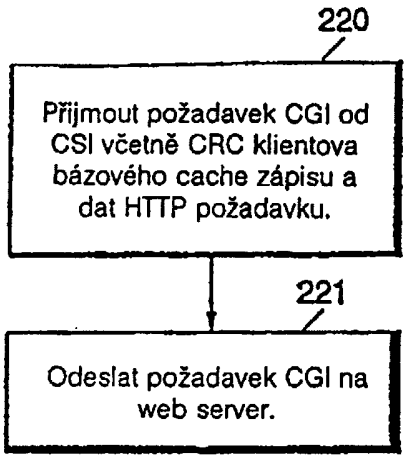
Obr. 7


200
Přijmout požadavek HTTP od browseru.
205
Je tento požadavek požadavkem CGI?
Ne
206
Odeslat požadavek HTTP na SSI.
Ano
210
Existuje klientův bázový cache zápis pro přijatý CGI?
Ne
Ano
212
Nastavit CRC určeného k odeslání na SSI na nulu.
211
Nastavit CRC pro odeslání na SSI na CRC klientova cache zápisu požadavku CGI.
213
Odeslat požadavek CGI na SSI včetně požadavku HTTP a CRC klientova bázového cache zápisu.

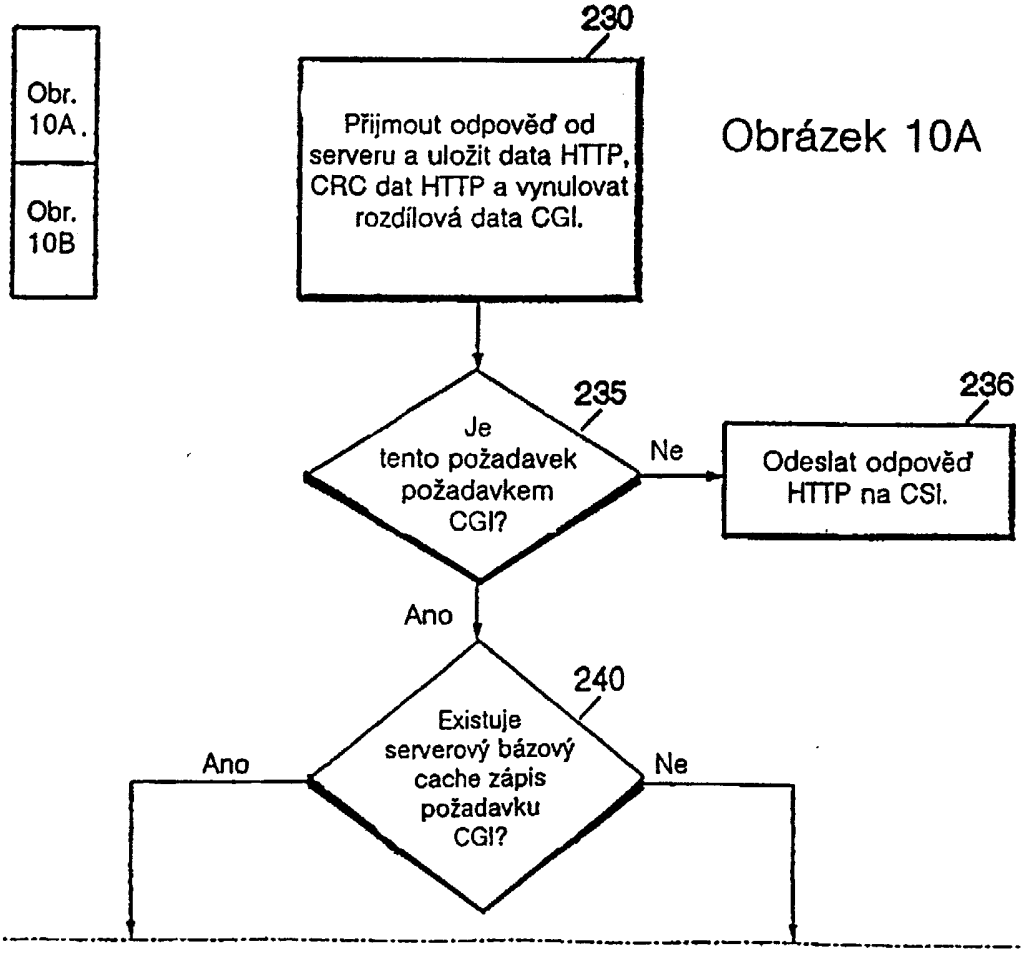
Obr. 8


260
Přijmout odpověď požadavku CGI od SSI včetně typu odpovědi a CRC serverové bázové formy.
265
Je to odpověď na požadavek CGI?
Ne
267
Předat odpověď HTTP na browser.
Ano
266
Přijmout odpověď a uložit data HTTP, rozdílová data CGI a CRC.
270
Existuje klientova bázová forma požadavku CGI?
Ano
Ne
275
Shoduje se CRC klientovy formy s CRC serverové formy?
Ano
Ne
276
Obnovit klientovú bázovou formu uložením dat HTTP, přijatých od SSI, a CRC, přijatého od SSI.
277
Rekonstruovat odpověď HTTP od web serveru začleněním rozdílových dat CGI do dat HTTP klientovy bázové formy tak, aby se vytvořil datový tok HTTP.
271
Vytvořit klientův bázový cache zápis uložením URL požadavku CGI, CRC datového toku HTTP od SSI a dat HTTP od SSI.
278
Odeslat rekonstruovanou odpověď na browser jako datový tok HTTP.

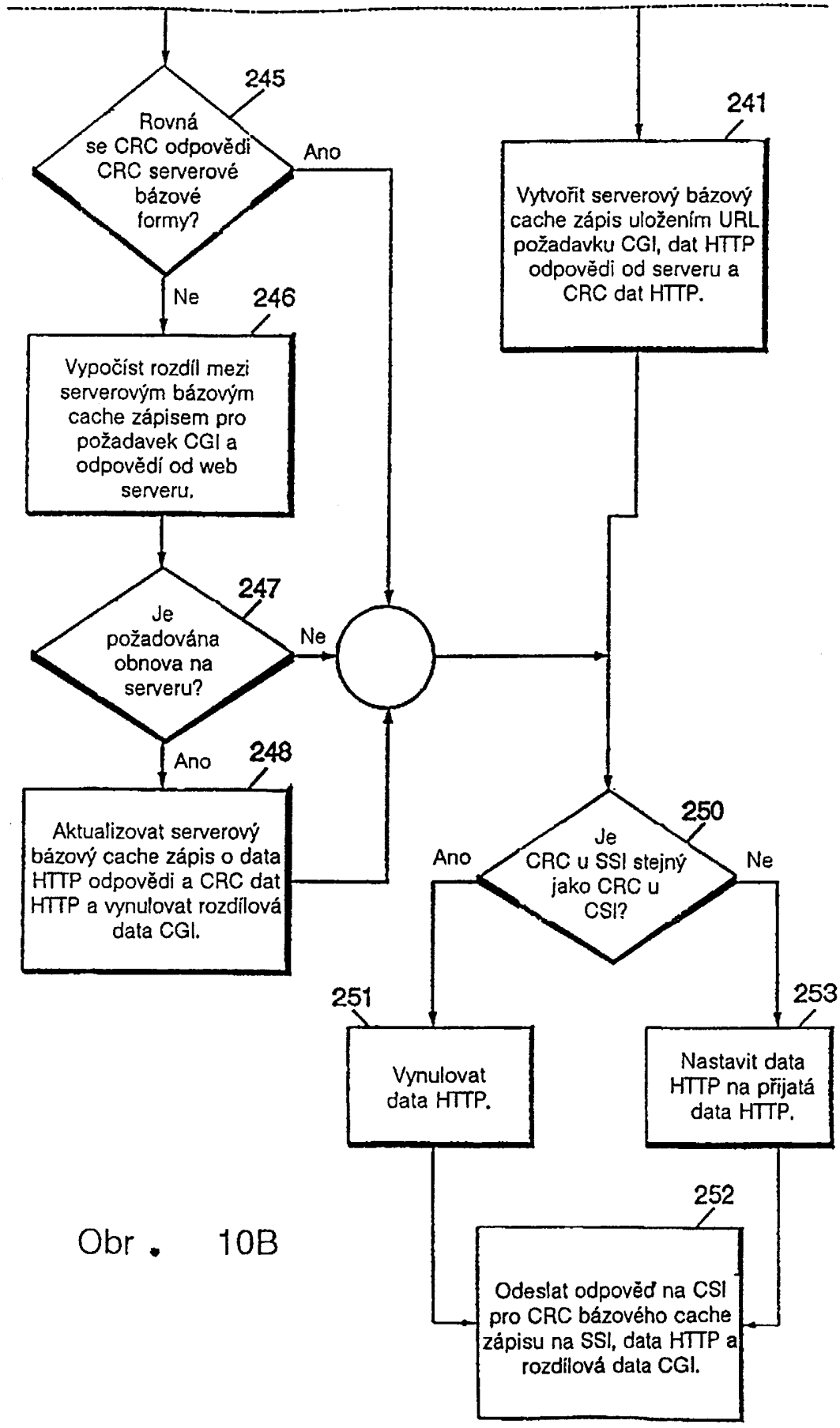
Obr. 9



Obr. 10

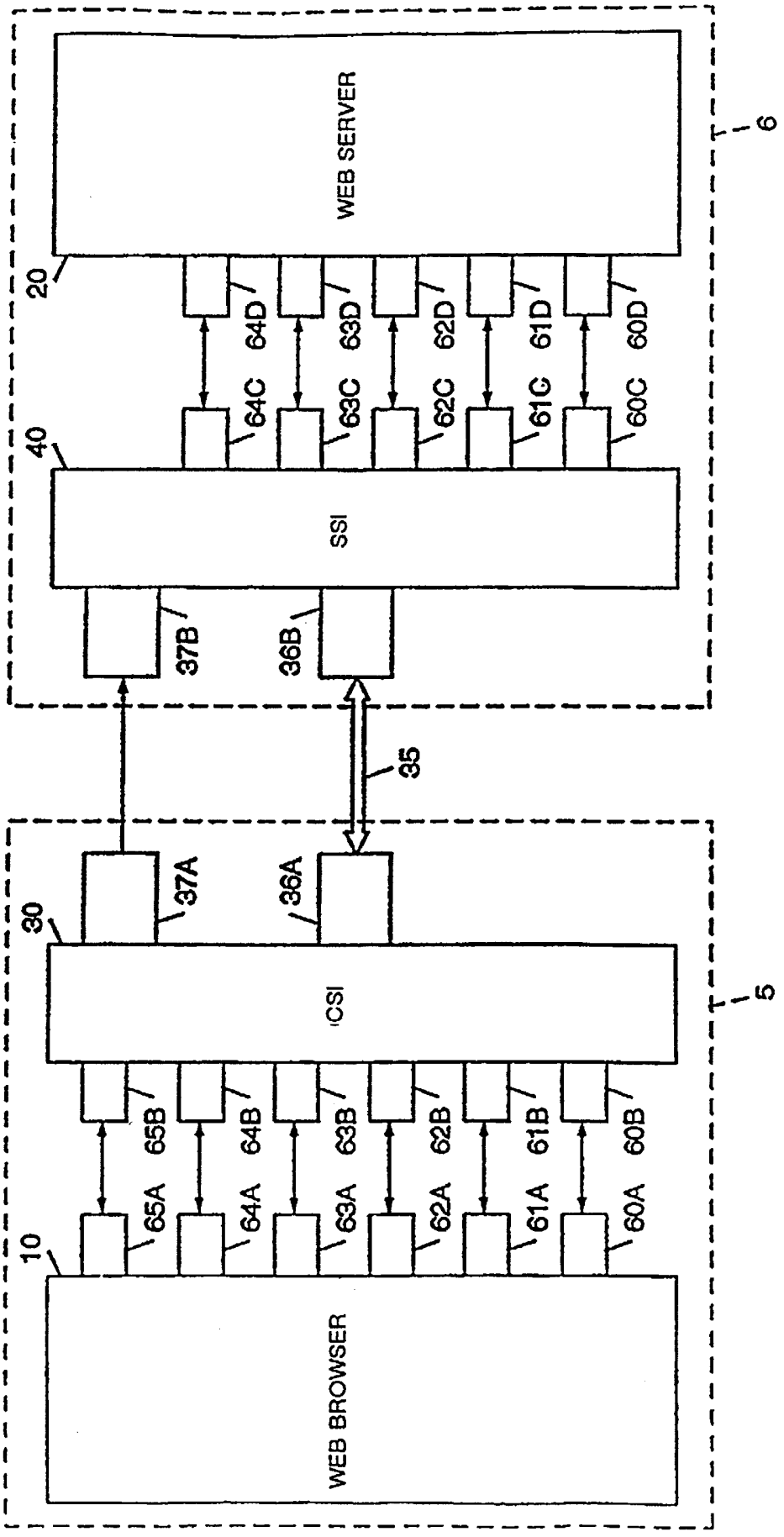


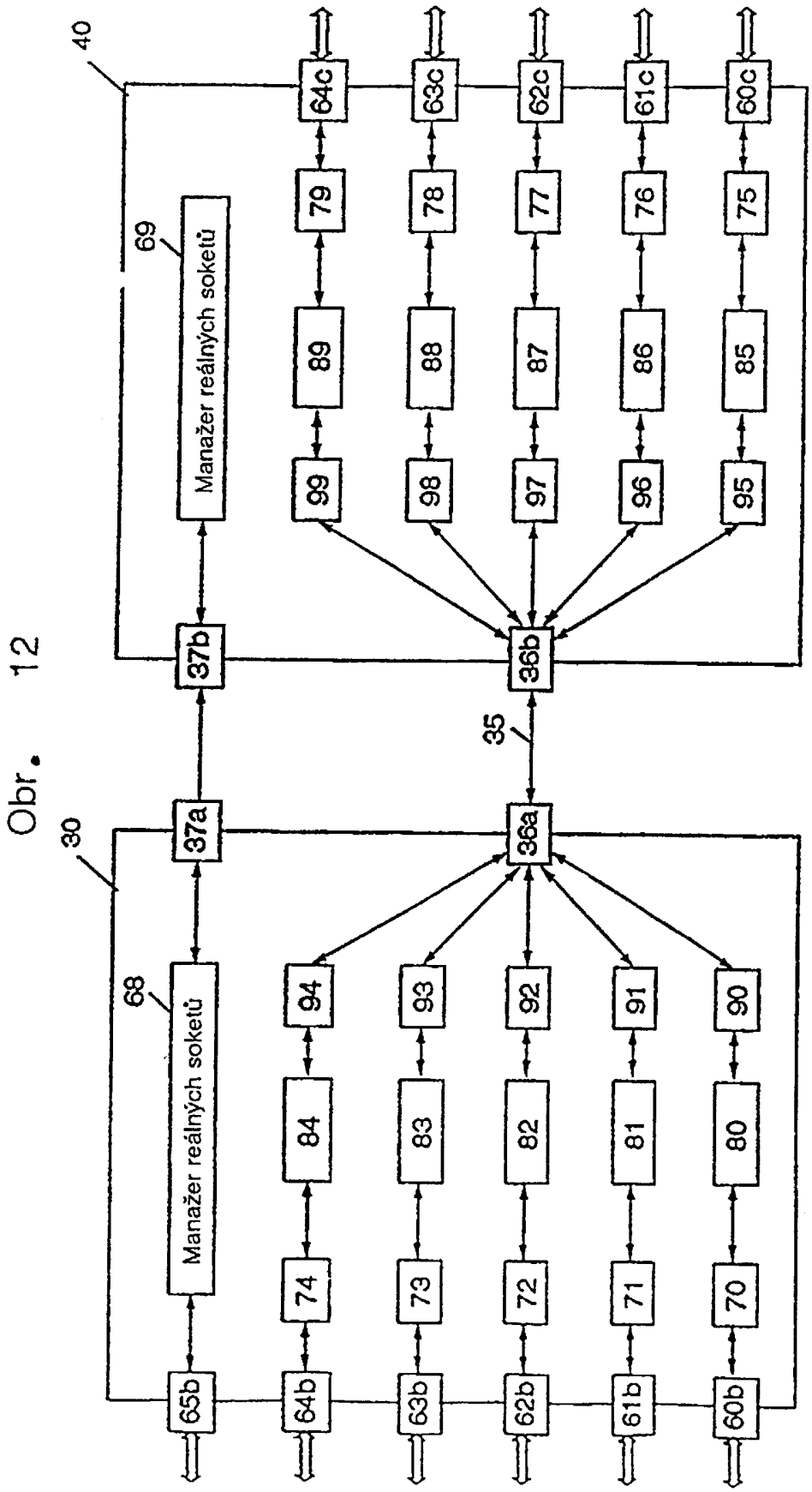
Obrázek 10A


245
Rovná se CRC odpovědi CRC serverové bázové formy?
Ano
Ne
246
Vypočíst rozdíl mezi serverovým bázovým cache zápisem pro požadavek CGI a odpovědí od web serveru.
247
Je požadována obnova na serveru?
Ne
Ano
248
Aktualizovat serverový bázový cache zápis o data HTTP odpovědi a CRC dat HTTP a vynulovat rozdílová data CGI.
241
Vytvořit serverový bázový cache zápis uložením URL požadavku CGI, dat HTTP odpovědi od serveru a CRC dat HTTP.
250
Je CRC u SSI stejný jako CRC u CSI?
Ano
Ne
251
Vynulovat data HTTP.
253
Nastavit data HTTP na přijatá data HTTP.
252
Odeslat odpověď na CSI pro CRC bázového cache zápisu na SSI, data HTTP a rozdílová data CGI.

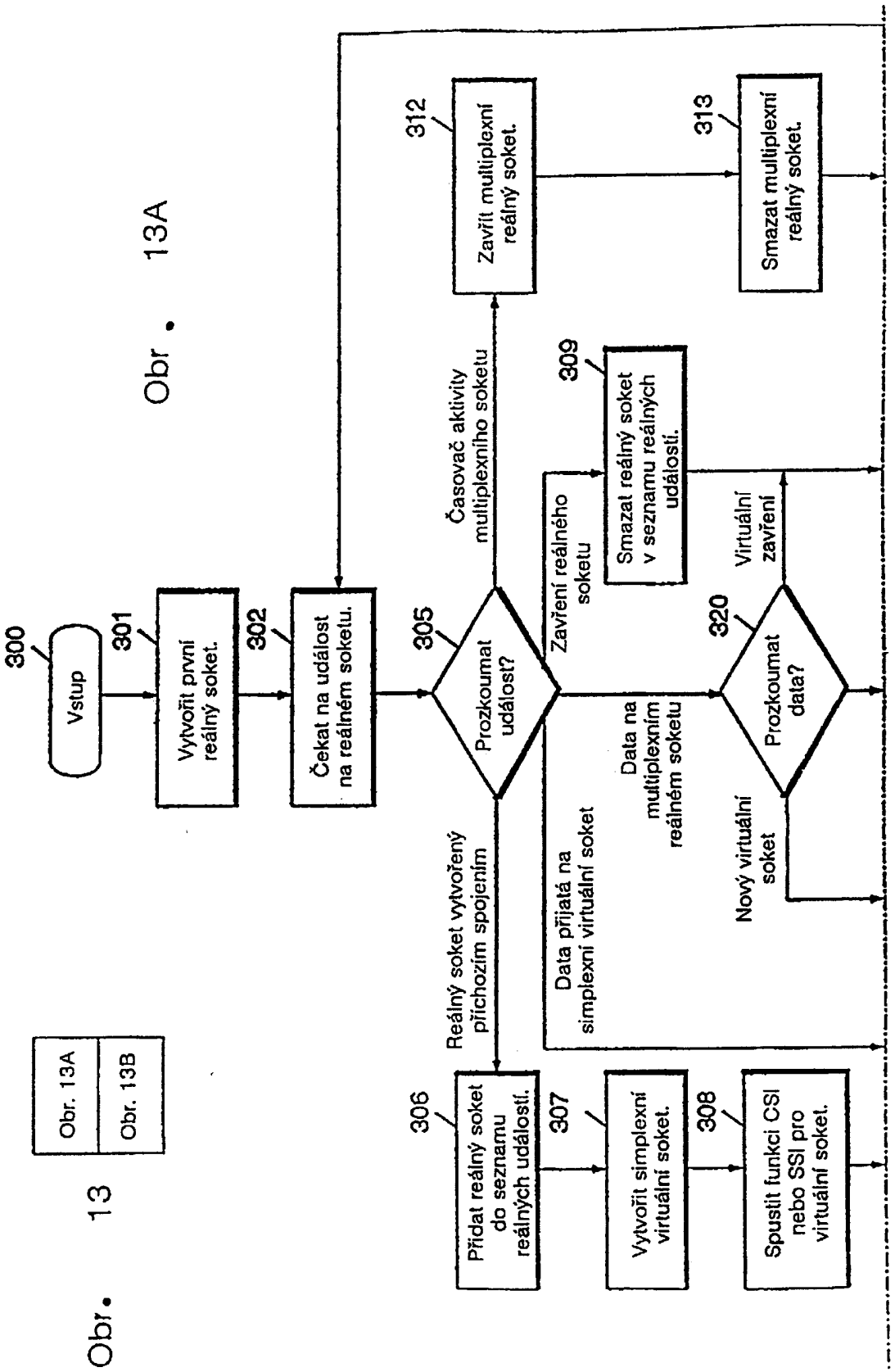
Obr. 10B

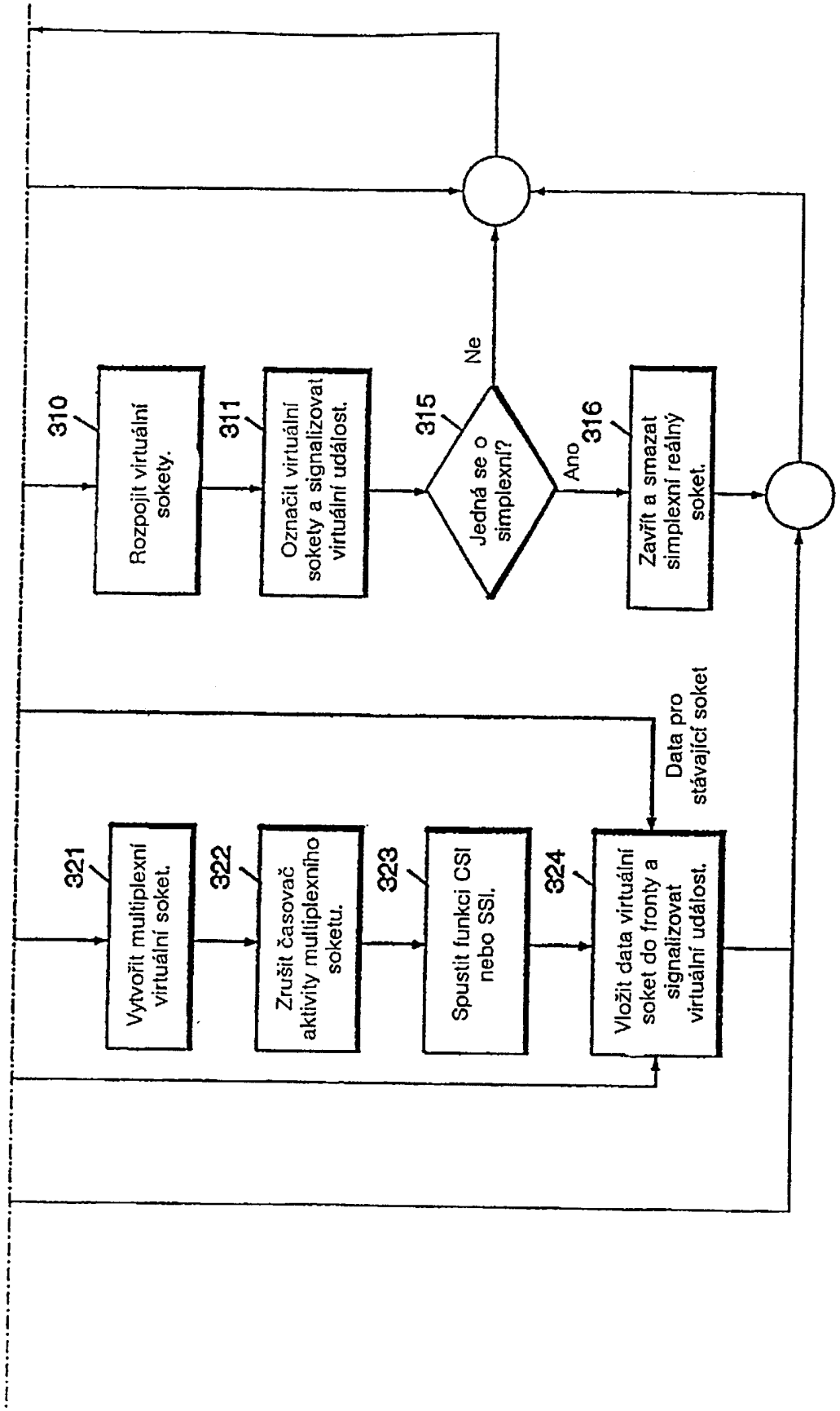
Obr. 11


WEB SERVER
6
20
64D
63D
62D
61D
60D
64C
63C
62C
61C
60C
40
SSI
37B
36B
35
37A
36A
30
CSI
5
65B
64B
63B
62B
61B
60B
65A
64A
63A
62A
61A
60A
10
WEB BROWSER


40
64c
63c
62c
61c
60c
79
78
77
76
75
89
88
87
86
85
99
98
97
96
95
69
Manažer reálných soketů
37b
36b
35
37a
36a
30
68
Manažer reálných soketů
94
93
92
91
90
84
83
82
81
80
74
73
72
71
70
65b
64b
63b
62b
61b
60b

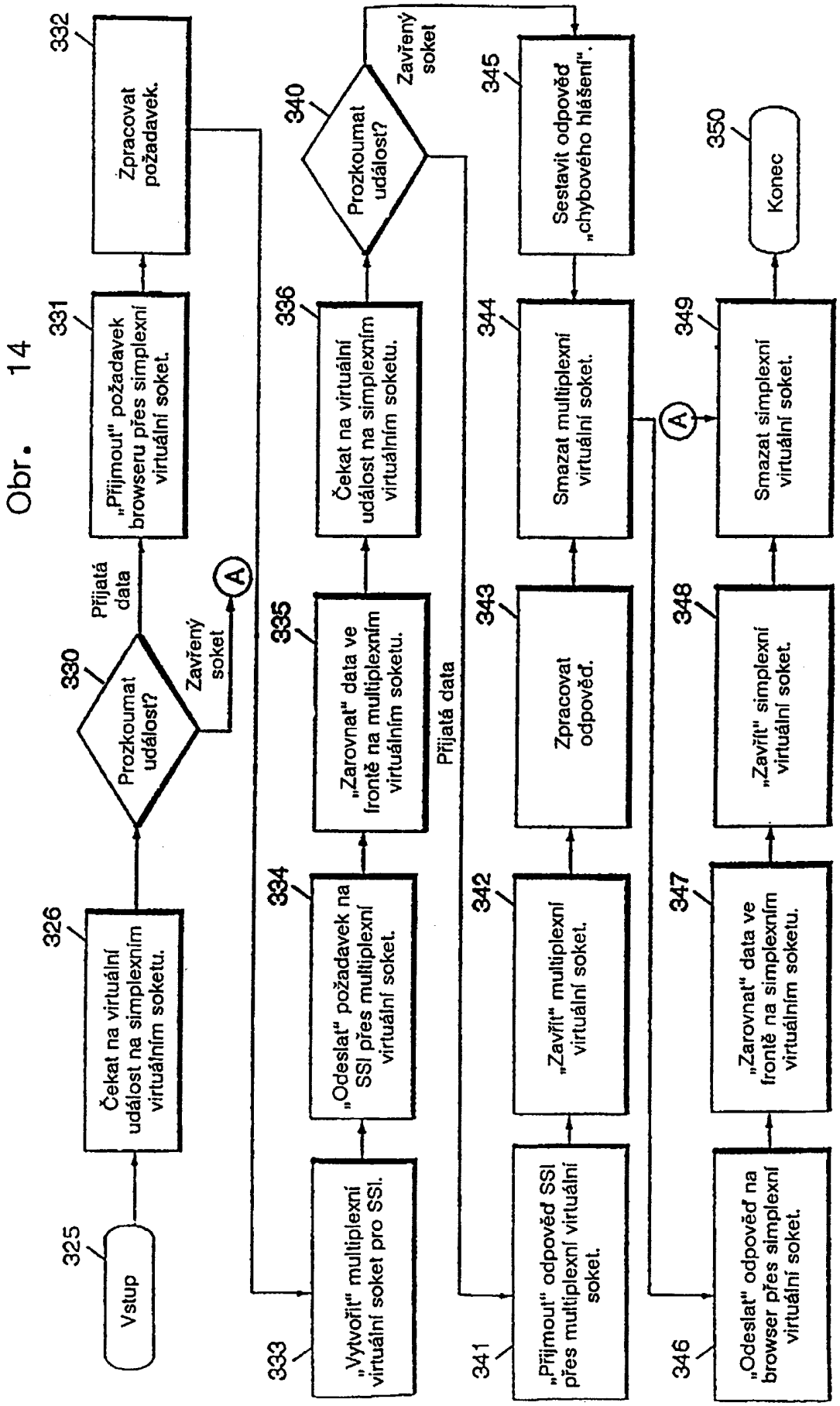
Obr. 12

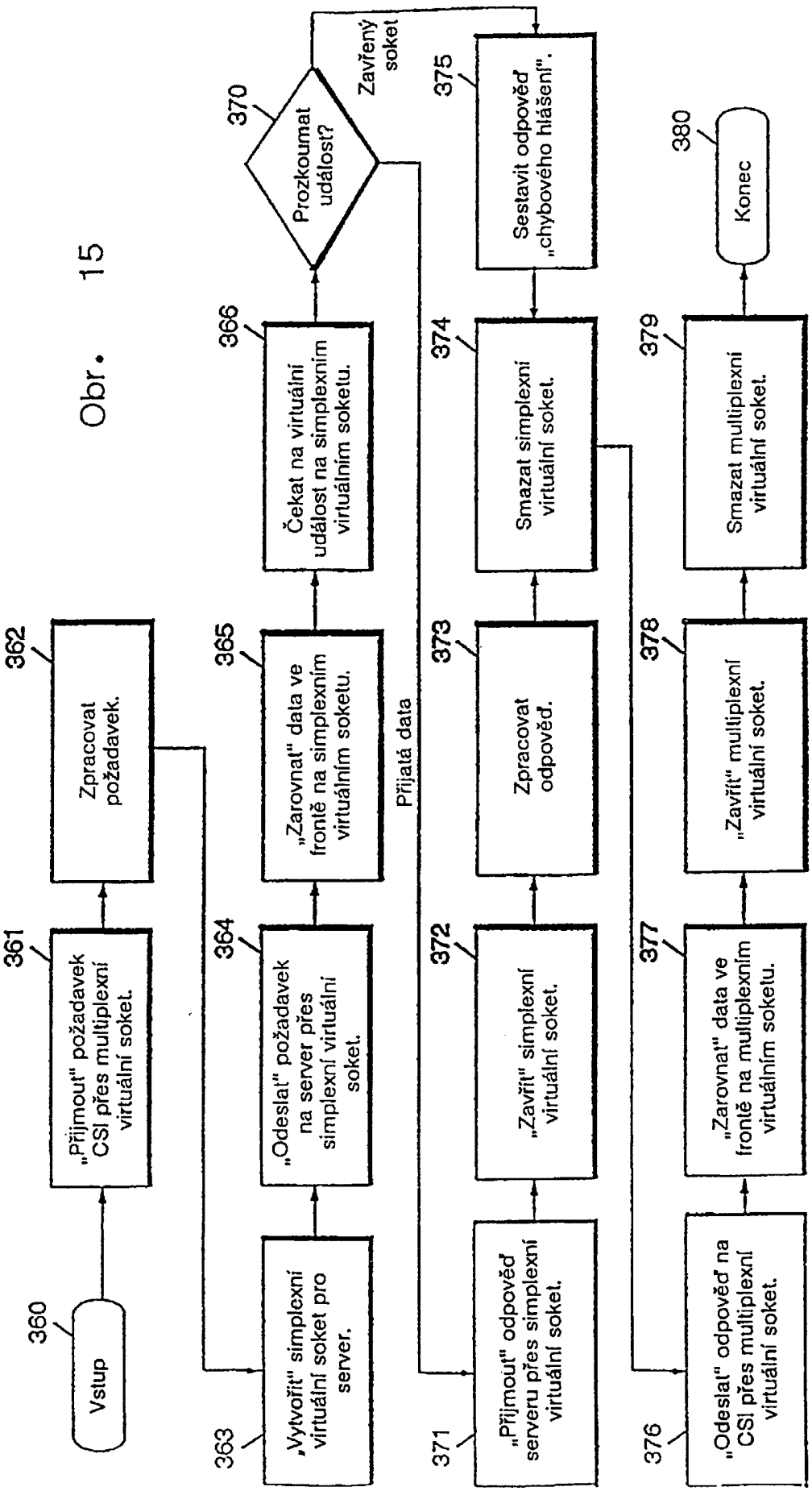

Obr. 13A
Obr. 13
Obr. 13A
Obr. 13B
300
Vstup
301
Vytvořit první reálný soket.
302
Čekat na událost na reálném soketu.
305
Prozkoumat událost?
Časovač aktivity multiplexního soketu
312
Zavřít multiplexní reálný soket.
313
Smazat multiplexní reálný soket.
Zavření reálného soketu
309
Smazat reálný soket v seznamu reálných událostí.
Virtuální zavření
320
Prozkoumat data?
Data na multiplexním reálném soketu
Nový virtuální soket
Reálný soket vytvořený příchozím spojením
Data přijatá na simplexní virtuální soket
306
Přidat reálný soket do seznamu reálných událostí.
307
Vytvořit simplexní virtuální soket.
308
Spustit funkci CSI nebo SSI pro virtuální soket.


310
Rozpojit virtuální sokety.
311
Označit virtuální sokety a signalizovat virtuální událost.
315
Jedná se o simplexní?
Ne
Ano
316
Zavřít a smazat simplexní reálný soket.
Data pro stávající soket
321
Vytvořit multiplexní virtuální soket.
322
Zrušit časovač aktivity multiplexního soketu.
323
Spustit funkci CSI nebo SSI.
324
Vložit data virtuální soket do fronty a signalizovat virtuální událost.

Obr. 13B

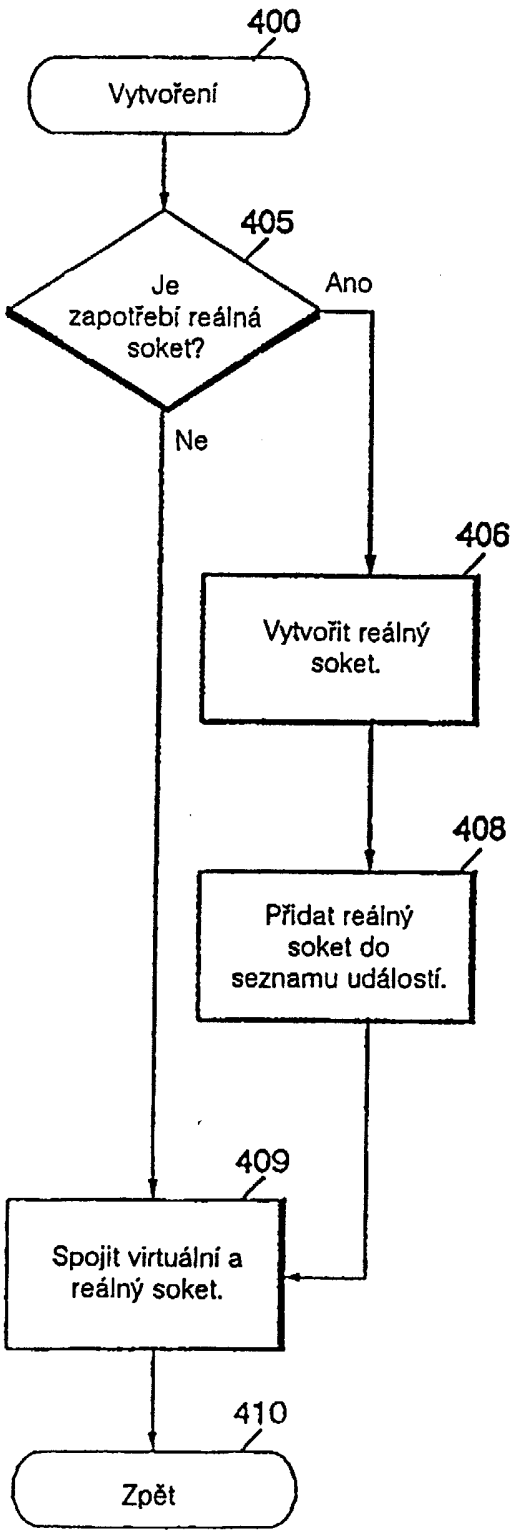
Obr. 14


325
Vstup
326
Čekat na virtuální událost na simplexním virtuálním soketu.
330
Prozkoumat událost?
Přijatá data
Zavřený soket
A
331
„Přijmout" požadavek browseru přes simplexní virtuální soket.
332
Zpracovat požadavek.
333
„Vytvořit" multiplexní virtuální soket pro SSl.
334
„Odeslat" požadavek na SSl přes multiplexní virtuální soket.
335
„Zarovnat" data ve frontě na multiplexním virtuálním soketu.
336
Čekat na virtuální událost na simplexním virtuálním soketu.
340
Prozkoumat událost?
Zavřený soket
Přijatá data
341
„Přijmout" odpověď SSl přes multiplexní virtuální soket.
342
„Zavřít" multiplexní virtuální soket.
343
Zpracovat odpověď.
344
Smazat multiplexní virtuální soket.
345
Sestavit odpověď „chybového hlášení".
346
„Odeslat" odpověď na browser přes simplexní virtuální soket.
347
„Zarovnat" data ve frontě na simplexním virtuálním soketu.
348
„Zavřít" simplexní virtuální soket.
349
Smazat simplexní virtuální soket.
350
Konec

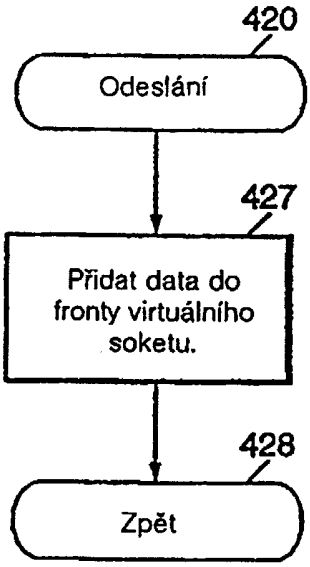

360
Vstup
361
„Přijmout" požadavek CSl přes multiplexní virtuální soket.
362
Zpracovat požadavek.
363
„Vytvořit" simplexní virtuální soket pro server.
364
„Odeslat" požadavek na server přes simplexní virtuální soket.
365
„Zarovnat" data ve frontě na simplexním virtuálním soketu.
366
Čekat na virtuální událost na simplexním virtuálním soketu.
370
Prozkoumat událost?
Zavřený soket
Přijatá data
371
„Přijmout" odpověď serveru přes simplexní virtuální soket.
372
„Zavřít" simplexní virtuální soket.
373
Zpracovat odpověď.
374
Smazat simplexní virtuální soket.
375
Sestavit odpověď „chybového hlášení".
376
„Odeslat" odpověď na CSl přes multiplexní virtuální soket.
377
„Zarovnat" data ve frontě na multiplexním virtuálním soketu.
378
„Zavřít" multiplexní virtuální soket.
379
Smazat multiplexní virtuální soket.
380
Konec

Obr. 15

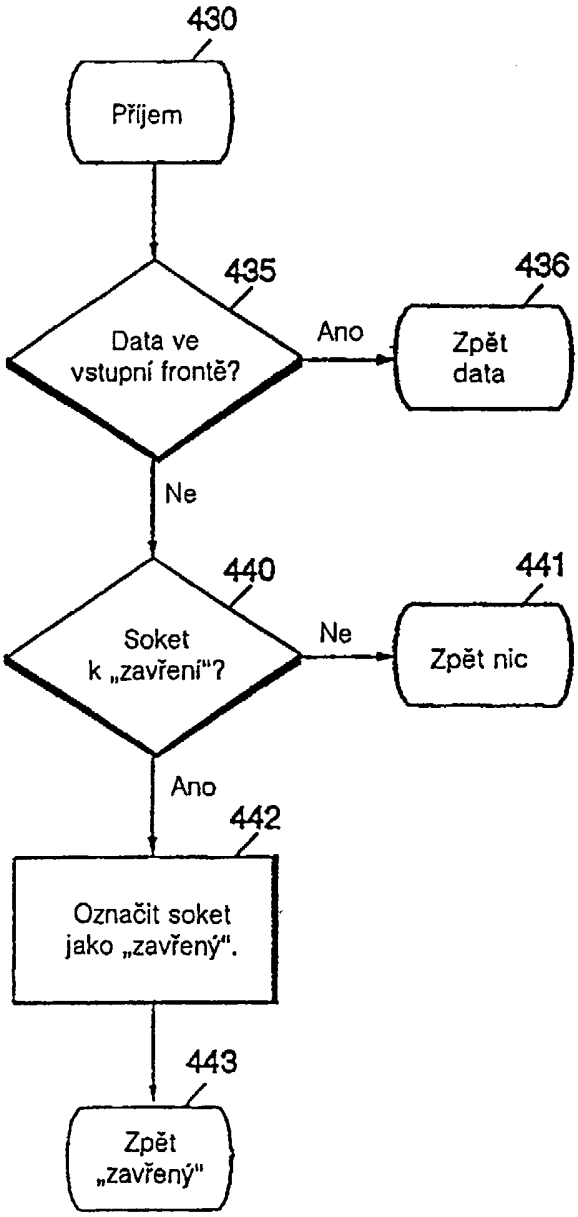
Obr. 16-1



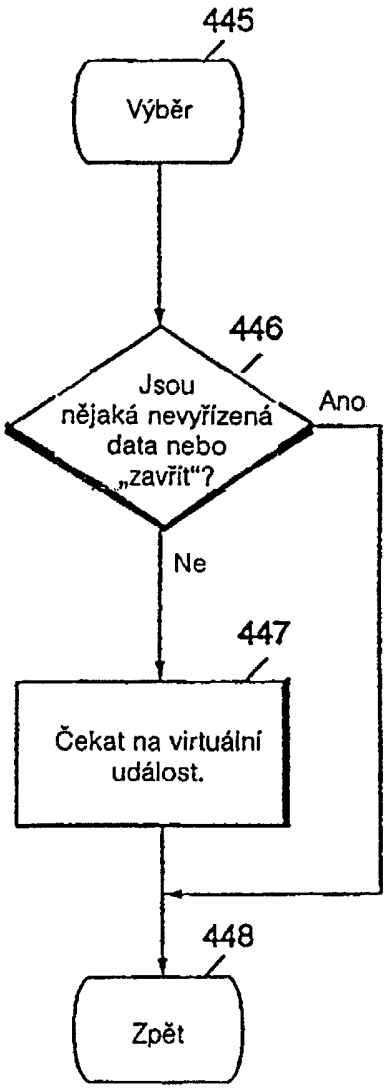
Obr. 16-2

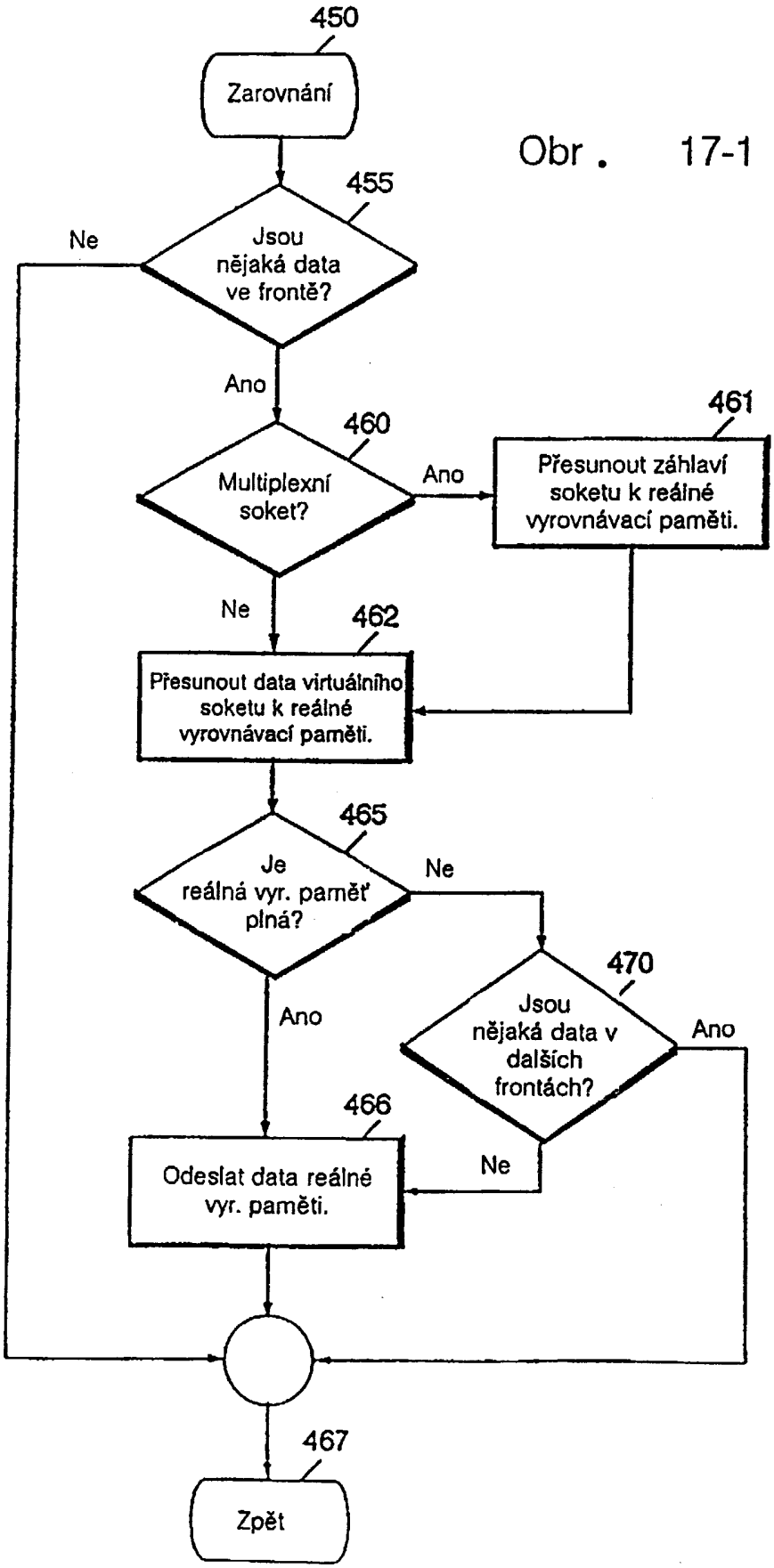


Obr. 16-3

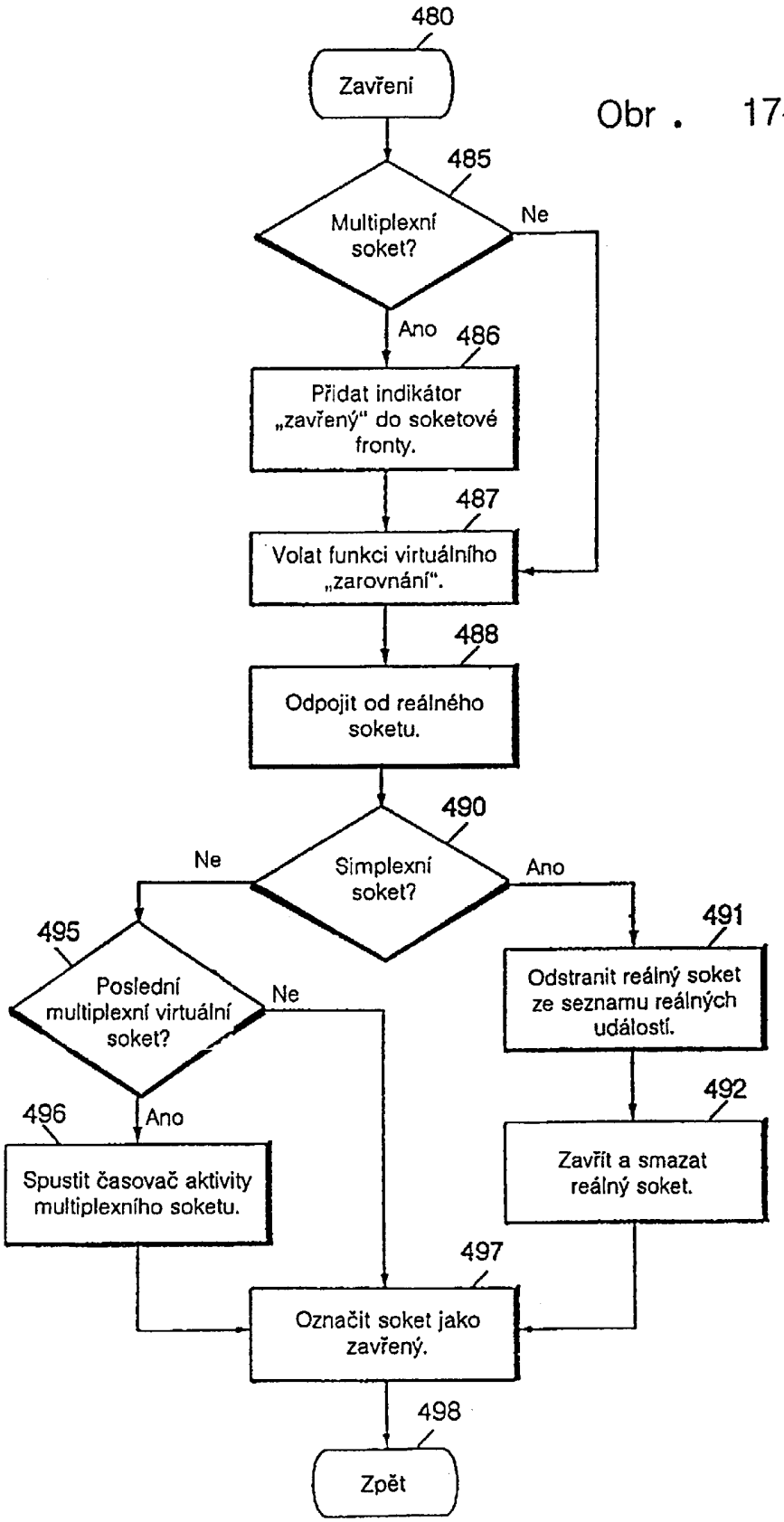


Obr. 16-4




450
Zarovnání
455
Jsou nějaká data ve frontě?
Ne
Ano
460
Multiplexní soket?
Ano
461
Přesunout záhlaví soketu k reálné vyrovnávací paměti.
Ne
462
Přesunout data virtuálního soketu k reálné vyrovnávací paměti.
465
Je reálná vyr. paměť plná?
Ne
Ano
470
Jsou nějaká data v dalších frontách?
Ano
Ne
466
Odeslat data reálné vyr. paměti.
467
Zpět

Obr. 17-1



Obr. 17-2

Konec dokumentu