Industria y Comercio

SUPERINTENDENCIA

Delegatura de propiedad Industrial

División de Nuevas Creaciones

PCT SOLICITUD

PATENTE DE INVENCIÓN

21. **EXPEDIENTE No.** 05-50648

54. **TÍTULO** PROTOCOLO LIGERO DE ENTRADA/SALIDA

51. **CLASIFICACIÓN INTERNACIONAL** _____

71. **SOLICITANTE** MICROSOFT CORPORATION

DOMICILIO REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA

74. **APODERADO** DARIO CARDENAS MAYAS

22. **BOGOTÁ, D. C.,** _____

(FORMA P-10)

PETITORIO

Industria y Comercio
SUPERINTENDENCIA

SUPERINDUSTRIA Y COMERCIO
Radicacion : 05050648 00060870 Folios: 136
Fecha (AMD): 2005-05-25 11:03:31
Tramite : 911 PCT-PATENT 370 FASE NACI 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

FORMULARIO UNICO DE SOLICITUD DE PATENTE
PCT
ENTRADA EN FASE NACIONAL

① SOLICITUD DE:

[X] Patente de Invención [ ] Patente de Modelo de Utilidad

[X] Capítulo I [ ] Capítulo II

② SOLICITANTE (71)

Nombre: **MICROSOFT CORPORATION**

Dirección: One Microsoft Way Redmond, Washington 98052, Estados Unidos de America

Nacionalidad o Domicilio: REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA.

Lugar de Constitución: Estado de Washington, Estados Unidos de América

Teléfono: 3123600 Fax: 3122410

E-mail: general@cardenasycardenas.com

IDENTIFICACION

C.C. [ ] NIT [ ]
C.E. [ ] Otro [X]
Cual _____
Número _____

③ REPRESENTANTE APODERADO

Nombre: **DARIO CARDENAS NAVAS**

Dirección: Carrera 7 No 71-52 Torre B Piso 9

Teléfono: 3123600 Fax: 3122410

E-Mail: general@cardenasycardenas.com

IDENTIFICACION

C.C. [X] NIT [ ]
C.E. [ ] Otro [ ]
Cual
No 17.088.629 BTA TP 1.250 C.S.J.

④ INVENTOR(ES) (72)

1.Nombre:
1. Ahmed H. Mohamed
2. Anthony F. Voellm

2. Dirección:
1. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
2. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America

3.Nacionalidad o Domicilio:
1. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
2. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA

⑤ PCT (86)

Solicitud Internacional No. PCT/US2004/024026 Fecha Julio 26, 2004
Publicación Internacional No. Fecha

⑥ Título(54): PROTOCOLO LIGERO DE ENTRADA / SALIDA

AMR/amg
Fol No P-326

PROTOCOLO LIGERO DE ENTRADA / SALIDA

⑦ **Clasificación Internacional (51):**

⑧ **Prioridad**

SI [X] NO [ ]

(33)País de Origen	(32)Fecha	(31)N° de Solicitud
U.S.A.	Diciembre 31, 2003	10/749.959

Comprobante de Pago No.: 05-37.298 **Fecha** Mayo 19, 2005

ANEXOS

- [X] Comprobante de pago de la tasa de presentación de la solicitud.
- [X] Documento que acredita la existencia y representación legal cuando el solicitante sea persona jurídica. (Protocolo 16.069)
- [X] Poderes, si fuere el caso. (Protocolo 16.069)
- [X] Documento de cesión del inventor al solicitante o a su causante.
- [ ] Declaraciones
- [X] Copia de la solicitud internacional. (Resumen, descripción, reivindicaciones, dibujos)
- [X] Traducción de la solicitud internacional al castellano. (Resumen, descripción, reivindicaciones, dibujos)
- [ ] Copia del examen preliminar internacional efectuado por la Autoridad Internacional de Examen Preliminar (IPEA).
- [ ] Traducción del examen preliminar internacional efectuado por la IPEA.
- [ ] De ser el caso, copia del contrato de acceso.
- [ ] De ser el caso, documento que acredite la licencia o autorización de uso de conocimientos tradicionales de las comunidades indígenas.
- [ ] De ser el caso, certificado de depósito del material biológico.
- [X] Arte final 12x12.
- [ ] De ser el caso, información sobre otras solicitudes de patente o títulos obtenidos en el extranjero por el mismo titular o su causante, relacionadas parcial o totalmente con la invención de esta solicitud.
- [X] De ser el caso, modificaciones efectuadas a la solicitud internacional.
- [ ] Otros.

11 FIGURA CARACTERISTICA

12 Solicito la Concesión de la patente,

NOMBRE: DARIO CARDENAS NAVAS

FIRMA:

C.C. 17.066.629 BTA TP. 1.250 C.S.J.

3

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

NIT : 800176089-2

SUPERINDUSTRIA Y COMERCIO
Radicacion : 05458648 00000000
Fecha (AMD): 2005-05-25 11:03:31 Folios: 136
Tramite : 011 PCT-PATENT 378 FASE NAC: 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

RECIBO OFICIAL DE CAJA : 05 - 37,298
FECHA : MAYO 19 DE 2005

C O N S I G N A C I O N

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PGO	Vr. PAGO
CARDENAS Y CARDENAS P.I. CONSIGNACION	BANCO POPULAR	050-00110-6	38039997	19/05/2005	3,028,000.00	

C O N C E P T O

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	50003-01-01 SOLICITUDES	1 TRAMITES DE SOL. DE PATENTE DE IN	443,000.00
		TOTAL : $	443,000.00

SON: CUATROCIENTOS CUARENTA Y TRES MIL PESOS

RESPONSABLE : _____

RECIBO DE CAJA APLICADO AL EXPEDIENTE No. Protocolo Ligero de Entrada/Salida

CARDENAS & CARDENAS
ABOGADOS

3

4

TRATADO DE COOPERACIÓN
PCT

PCT/US2004/024026

NOTIFICACIÓN DE RECIBO DE COPIA DEL RÉCORD
(Regla PCT 24.2(a))

De LA AGENCIA INTERNACIONAL

☐

CONKLIN, John, B.
Leydig, Voit & Mayer, LTD.
Two Prudential Plaza, Suite 4900
180 N Stetson Av Chicago, IL 60601-6780
Estados Unidos de America ¶

Fecha de envió (día/mes/año)	NOTIFICACIÓN IMPORTANTE
Septiembre 21 2004 (21.09.2004)	
Archivo de Referencia del Agente o Solicitante	**No.de solicitud internacional**
229789	PCT/US2004/024026

Por medio de la presente se notifica al solicitante que la agencia Internacional ha recibido copia del récord de la solicitud internacional como se detalla a continuación.

Nombre(s) del solicitante(s) y los Estado(s) en los que se solicita

MICROSOFT CORPORATION (todos los estados designados excepto los EU)
MOHAMED Ahmed, H et al (para los EU)

Fecha de archivo internacional: Julio 26 2004 (26.072004)

Fecha(s) de reivindicación de prioridad : Diciembre 31 2003(31.12.2003)

Fecha en la que la Agencia Internacional recibe copia del récord: Septiembre 13 2004 (13.09.2004)

Lista de oficinas designadas

AP: BW,GH,GM,KE,LS,MW,MZ,NA,SD,SL,SZ,TZ,UG,ZM,ZW
EA: AM,AZ,BY,KG,KZ,MD,RU,TJ,TM
EP: AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PL,PT,RO,SE,SI,SK, TR
OA: BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG
Nacional: AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM, DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS, LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NA,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK, SL,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,YU,ZA,ZM,ZW

La Agencia Internacional WIPO 34, chemin des Colombettes 1211 Ginebra 20, Suiza Fax No. (41-22)338.70.80	Oficial Autorizado: **Norbert RIGHETTO** Teléfono No. (41-22) 338-9889

Formato PCT/IB/301 (Anexo)(Enero 2004)

4

229789 ¼

SOLICITUD PCT Original (para sumisión)

0 0-1	Para uso exclusivo de la oficina receptora Solicitud Internacional No.	
0-2	Fecha de Presentación Internacional	
0-3	Nombre de la Oficina Receptora y "Solicitud Internacional PCT"	
0-4 0-4-1	Formato PCT/RO/101 SOLICITUD PCT Preparar utilizando	PCT-SEGURO (Modo FÁCIL) Versión 3.50 (Construida 0002.162)
0-5	Solicitud Los abajo firmantes solicitan que la presente solicitud internacional sea procesada de acuerdo con el Tratado de Cooperación de Patentes.	
0-6	Oficina receptora (especificada por el solicitante)	Oficina de Patentes y Marcas registradas de los Estados Unidos de America (USPTO)(RO/EU)
0-7	Archivo de Referencia del Agente o Solicitante	229789
1	Título del invento	PROTOCOLO LIGERO DE ENTRADA/SALIDA
II II-1 II-2 II-3 II-4 II-5 II-6 II-7 II-8 II-9	Solicitante Esta persona es: Solicitante para Nombre Dirección Estado o nacionalidad Estado donde reside Teléfono Facsimil	 Solicitante únicamente Todos los estados designados excepto EU MICROSOFT CORPORATION One Microsoft Way Redmond, Washington 98052 Estados Unidos de América EU EU 425-882-8080 425-936-7329
III-1 III-1-1 III-1-2 III-1-4 III-1-5 III-1-6 III-1-7	Solicitante y/o inventor Esta persona es Solicitante para Nombre (Apellido, Nombre) Dirección Estado o nacionalidad Estado donde reside	 Solicitante y inventor EU únicamente MOHAMNED Ahmed H. 29 210 Place SE Sammamish, Washington 98074 Estados Unidos de America EU EU

III-2	Solicitante y/o Inventor	Solicitante y Inventor
III-2-1	Esta persona es:	EU únicamente
III-2-2	Solicitante para	
III-2-4	Nombre (apellido seguido por nombre)	VOBLLM, Anthony, F.
III-2-5	Dirección	9800 229th Lane NE Redmond, Washington 98053 Estados Unidos de América
III-2-6	Estado o nacionalidad	EU
III-2-7	Estado donde reside	EU
IV-1	AGENTE O REPRESENTANTE COMÚN; O DIRECCIÓN DE CORRESPONDENCIA	Agente
IV-1-1	Por medio de la presente la persona identificada adelante ha sido señalada para actuar	CONKLIN, John, B.
IV-1-2	en nombre del (los) solicitante (s) ante las Autoridades Internacionales competentes como: Nombre y dirección: (apellido seguido por nombre) Dirección	LEYDIG, VOIT & MAYER, LTD. Two Prudential Plaza, Suite 4900 180 N Stetson Av Chicago, Illinois 60601-6780 Estados Unidos de América
IV-1-3	Teléfono	312-616-5600
IV-1-4	Facsimil	312-616-5700
IV-1-5	e-mail	mail@leydig.com
IV-1-6	No. de registro del agente	30369
V	Designaciones	
V-1	El archivo de la presente solicitud constituye ante la Norma 4.9(a), de la designación de todos los Estados Contratantes que se rigen por el PCT en la fecha internacional de archivo, para que se otorgue toda la protección disponible y, donde sea pertinente, para otorgar tanto la patente regional como la patente nacional.	
VI-1 VI-1-1 VI-1-2 VI-1-3	Reivindicación de la prioridad de la siguiente solicitud nacional temprana Fecha de presentación Numero país	 Diciembre 31 2003 (31.12.2003) 10/749.959 EU
	Se solicita a la Oficina receptora preparar y transmitir a la Oficina Internacional una copia certificada de la solicitud temprana identificada arriba como: Items:	VI-1

Solicitud para utilizar los resultados de la búsqueda tempranas; Investigación de tal búsqueda

VII-1	Elección de la Autoridad de Búsqueda Internacional	Oficina de Patentes y Marcas registradas de los Estados Unidos de America (USPTO)(ISA/EU)	
VII-2	Solicitud para utilizar los resultados de la búsqueda tempranas; Investigación de tal búsqueda Fecha: Numero País (oficina regional)	Diciembre 31 2003 (31.12.2003) 10/749.959 EU	
	Declaraciones		
	Declaración de la identidad del inventor:		
	Declaración del derecho del solicitante, como la fecha de presentación internacional, para solicitar y otorgarse una patente		
	Declaración del derecho del solicitante, como la fecha de presentación internacional, para reivindicar la prioridad de una solicitud temprana		
	Declaración de inventor (solo para los propósitos de designación de los Estados Unidos de América)		
	Declaración de descripción no perjudicial o excepciones para falta de novedad		
	Lista De Verificación	Numero de hojas	Archivos electrónicos anexos
	Solicitud(incluyendo hojas de declaración)	4	x
	Descripción	25	
	Reivindicaciones	4	
	Resumen	1	x
	Dibujos	23	
	TOTAL	57	
	Items que acompañan		
	Hoja de calculo de las tasas	x	
	PCT-SEGURO medios físicos		
	Dibujos que acompañan al resumen	1	
	Idioma en que se presenta la solicitud internacional	Ingles	
	Firma Del Solicitante, agente o representante común Nombre Nombre del firmante capacidad	CONKLIN, John, B.	

PARA USO DE LA OFICINA RECEPTORA UNICAMENTE

10-1	Fecha de recibo actual de la solicitud internacional propuesta:	
10-2 10-2-1 10-2-2	Dibujos Recibidos No recibidos	
10-3	Fecha corregida de recibo actual debido a posterior pero oportuno recibo de papeles o dibujos que completan la solicitud internacional propuesta	
10-4	Fecha de recepción oportuna de las correcciones requeridas de acuerdo con el Artículo 11(2) PCT:	
10-5	Autoridad de Búsqueda Internacional	ISA/EU
10-6	Tramite de la copia de búsqueda retrasada hasta que la tasa de búsqueda se pague	
PARA USO DE LA AGENCIA INTERNACIONAL UNICAMENTE		
11-1	Fecha de recibo de la copia de registro por la Oficina Internacional:	

TRATADO DE COOPERACIÓN DE PATENTES

De la OFICINA RECEPTORA

Para:

JOHN, B CONKLIN,.
LEYDIG, VOIT & MAYER, LTD.
TWO PRUDENTIAL PLAZA, SUITE 4900
180 N STETSON AV
CHICAGO, IL 60601-6780

PCT

NOTIFICACIÓN DEL NUMERO DE SOLICITUD INTERNACIONAL Y FECHA DE ENTREGA

Fecha de envió (*dia/mes/año*)	09 Septiembre 2004

Archivo de referencia del solicitante	NOTIFICACIÓN IMPORTANTE	
229789		
No. de solicitud internacional PCT/US2004/024026	Fecha de solicitud internacional (*dia/mes/año*) Julio 26 2004	Fecha prioritaria(*dia/mes/año*) Diciembre 31 2003
Solicitante: MICROSOFT CORPORATION		
Título del invento: PROTOCOLO LIGERO DE ENTRADA/SALIDA		

1. Por medio de la presente se notifica al solicitante que a la solicitud internacional le ha sido asignado el numero de solicitud Internacional y la fecha de archivo indicados arriba.
2. se notifica al solicitante que la copia del récord de la solicitud internacional

☒ Fue transmitida a la Agencia Internacional el 09 de Septiembre de 2004

☐ no ha sido transmitida a la Agencia Internacional por las razones que se indican abajo y una copia de esta notificación ha sido enviada a la Agencia Internacional*.

☐ porque el visto bueno de la seguridad nacional no se recibido

☐ porque (especificar razón)

* La Agencia Internacional monitoreo el tramite de la copia del récord por parte de la oficina Receptora y notificara al solicitante (con el formato PCT/IB/301) de su recepción. Si la copia del récord no se ha recibido el día de vencimiento 14 meses de la fecha de prioridad, la Agencia Internacional notificara al solicitante (Regla 22.1(e)).

3. INFORMACIÓN SOBRE EL TRAMITE DE LA LICENCIA INTERNACIONAL Completada por K.D..

☐ Licencia adicional para el tramite Internacional no se requiere. Este asunto ya esta cubierto por una licencia expedida o por una solicitud nacional de EU equivalente. Refiérase a esta licencia para información sobre su alcance.

☐ La licencia para tramite internacional no se requiere. 37 CFR.5.11(e)(i) O 37 CFR 5.11(e)(2).sin embargo una licencia puede ser necesaria para tramites adicionales. Ver CFR5.15(b).

☒ La licencia de tramite internacional ha sido otorgada. 35 U.S.C. 184; 37 CFR5.11 en: Septiembre 03 de 2004.

☐ 37 CFR 5.15(a) ☒ 37CFR 5.15(b)

Nombre y dirección de correo de IB/	Oficial Autorizado
Mail Stop PCT, Comisionado de Patentes P.O. Box 1450 Alexandria, VA 22313-1450 Facsímil No. 703-305-3230	Kareem Duncan Teléfono: 703-305-3685
Formato PCT/RO/105 (julio 1992)	

10

PROTOCOLO LIGERO DE ENTRADA/SALIDA

CAMPO TECNICO

[0001] El presente invento se relaciona en general con sistemas y métodos de acceso a archivos remotos, y en particular con técnicas para descargar procesos utilizando el Acceso Directo a la Memoria Remota (RDMA).

ANTECEDENTES

[0002] en ambientes de computación es deseable conservar los escasos recursos de la CPU. Esto es especialmente critico en algunos ambientes, tales como redes de aplicaciones de nodos de servidores. A medida que las redes se hacen más rápidas, también se hacen más exigentes con las CPUs para procesar paquetes y ejecutar operaciones de I/O, con el resultado de una ejecución mas lenta. Esto es especialmente perjudicial para aplicaciones intensivas de I/O como son las bases de datos.

[0003] una solución a este problema es descargar de la CPU trabajos excesivos de red y de I/O en ambientes de red; cuando se utilizan protocolos de sistemas de archivos distribuidos y de transporte como el NFS o el SMB/CIFS es posible enviar una petición de I/O de una maquina local a una maquina remota. Sin embargo, no necesariamente es el caso que la maquina local logre un ahorro significativo utilizando estos métodos.

[0004] en un contexto de una sola maquina, la carga del procesamiento I/O se puede aliviar redirigiendo las tareas de I/O a un controlador de acceso directo de memoria (DMA). La tecnología de acceso directo de memoria remota (RDMA) es una extensión, desarrollada recientemente, del DMA para computadores múltiples en red. RDMA permite mover los datos entre los búfers de memoria en dos máquinas comunicadas y equipadas con una tarjeta de interfaz de red RDMA (NICs) sin tener que involucrar a la CPU o al sistema operativo de la máquina que origina o de la maquina de destino. El RDMA puede usarse para descargar el procesamiento I/O en una máquina remota, mientras permite que la maquina local reclame ciclos de CPU para las aplicaciones. El RDMA se ha utilizado para aprovechar las tecnologías de interconexión de alta velocidad y banda ancha, como la Interfase de Arquitectura Virtual (VIA), la Infiniband, y la iWarp. Estas interconexiones están diseñadas particularmente para ser de alta confiabilidad en redes conectadas entre grupos de nodos de servidores con un centro de datos u otros ambientes locales donde se comparten archivos.

[0005] Los protocolos que definen la comunicación entre un nodo de descarga local y una máquina remota debe diseñarse para las capacidades asociadas con la tecnología de RDMA para poder ser utilizado

10'

en forma total y lograr sus beneficios eficazmente. En consecuencia, existe la necesidad del protocolo ligero de entrada/salida (LWIO) del presente invento.

RESUMEN DEL INVENTO

[0006] De acuerdo a una implementación del presente invento, se provee un sistema para descargar las tareas de I/O de un primer computador a un segundo computador. El sistema incluye a un cliente que opera un primer computador y a un servidor que funciona en un segundo computador. El sistema incluye además uno o mas canales RDMA para unir al primer computador con el segundo. El cliente y el servidor se comunican de acuerdo con un protocolo LWIO que comprende una fase de descubrimiento de red y una fase de procesamiento I/O. El protocolo LWIO se usa en asociación con otro protocolo de red, como el SMB/CIFS, normalizando la seguridad e infraestructura de la autenticación del segundo protocolo. Para proporcionar un modelo mas seguro, el modelo I/O en el protocolo es asimétrico: se llevan a cabo las lecturas usando RDMA, mientras la escritura se lleva a cabo usando operaciones de envió.

[0007] de acuerdo con otro aspecto del presente invento, se proporciona un método para descargar una tarea de I/O de una primera computadora a una segunda computadora. El método aprovecha los dispositivos capaces de comunicación RDMA comunes en las dos computadoras y es asociado con un protocolo ligero cliente-servidor (LWIO). El protocolo generalmente comprende una etapa de descubrimiento seguida por una fase de procesamiento I/O. Durante la fase de procesamiento I/O, el cliente hace una petición de descarga de I/O a la segunda maquina.

[0008] En la fase de descubrimiento, el cliente inicialmente obtiene una petición de una llave de continuación del servidor El cliente luego abre un canal con el servidor, a través del cual el cliente envia una petición de negociación que contiene una lista de proveedores de RDMA en la primera maquina. El servidor luego envía una respuesta de negociación a través del canal que contiene una lista de proveedores capaces en la segunda maquina y que hace juego con los proveedores en la primera maquina. El cliente luego crea una conexión RDMA con el servidor a través del proveedor que se comparte. El cliente y el servidor autentican mutuamente la nueva conexión. El cliente luego registra uno o más archivos para utilizar con el servidor.

[0009]Los mensajes de petición de procesamiento I/O incluyen un mensaje de cerrar, un mensaje de cancelación, un mensaje de lectura, un mensaje de escritura, un mensaje de lectura vectorizada y un mensaje de escritura vectorizada. El protocolo tiene un modelo asimétrico I/O por razones de seguridad. Los datos de lectura son enviados al cliente utilizando operaciones de escritura RDMA, mientras las escrituras se completan usando envios comunes. El cliente puede especificar en las peticiones de lectura y escritura que el servidor las complete en modalidad de consulta o en modalidad con interrupciones. Si el cliente especifica que no se debe completar en modalidad conjunta, el servidor completa la petición de procesamiento I/O enviando un bloque de estatus al primer computador a través de una transferencia RDMA. Si el cliente

12

especifica que se debe completar en modalidad conjunta, el cliente puede pedir que el servidor lo despierte tan pronto se termine el I/O a través de un mensaje de petición con interrupciones.

[0010] de acuerdo con otro aspecto del presente invento, se provee un método para manejar los búfers en un protocolo de descarga I/O. El método involucra el uso de un mecanismo de búfer de crédito. Una transacción cliente/servidor de crédito comprende un saludo triple iniciado y terminado por el servidor. El servidor le envía al cliente un mensaje de crédito delta que incluye un campo de información asociado a un numero de créditos. Si el numero es un numero negativo N, el cliente tiene que entregar N créditos.

[0011] otros aspectos del invento incluyen los rasgos antedichos implementados en los medios de comunicación legibles por computador como son los productos de programas de computador y estructuras de datos.

DESCRIPCIÓN BREVE DE LOS DIBUJOS

[0012] mientras las reivindicaciones anexas presentan, con particularidad, los rasgos del presente invento, el invento, junto con sus objetos y ventajas, puede entenderse mejor con la siguiente descripción detallada que tomada junto con los dibujos la acompañan:

[0013] La FIG.1. es un diagrama que generalmente muestra un ambiente de computación cliente-servidor ejemplar que involucra dos computadores capaces de comunicarse por transferencias RDMA, dentro de este aspecto es que el presente invento se puede incorporar;

[0014] La FIG 2. es un diagrama de flujo que muestra, de acuerdo con una implementación del presente invento, las etapas iniciales en la fase de descubrimiento de dos protocolos LWIO;

[0015] La FIG 3. es un diagrama que generalmente muestra una representación ejemplar de una petición de llave de continuación del servidor de acuerdo con una implementación del presente invento;

[0016] La FIG 4A. es un diagrama que generalmente muestra una representación de un mensaje de petición del cliente de negociación de acuerdo con una implementación del presente invento;

[0017] La FIG. 4B es un diagrama que generalmente muestra una representación ejemplar de una respuesta de negociación del servidor de acuerdo con una implementación del presente invento;

[0018] La FIG. 5 es un diagrama de flujo que generalmente muestra etapas adicionales que se efectúan en la fase de descubrimiento de los protocolos LWIO de acuerdo con una implementación del presente invento;

[0019] La FIG. 6A es un diagrama que generalmente muestra una representación ejemplar de un mensaje de petición del cliente de autenticación de acuerdo con una implementación del presente invento;

[0020] La FIG. 6B es un diagrama que generalmente muestra una representación ejemplar de una respuesta del servidor de autenticación de acuerdo con una implementación del presente invento;

12

[0021] La FIG. 6C es un diagrama que generalmente muestra una representación ejemplar de una respuesta de estatus del servidor completando la autenticación de acuerdo con una implementación del presente invento;

[0022] La FIG. 7A. es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de registro de archivo de acuerdo con una implementación del presente invento;

[0023] La FIG. 7B es un diagrama que generalmente muestra una representación ejemplar de una respuesta del estatus del servidor que completa el registro del archivo de acuerdo con una implementación del presente invento;

[0024] La FIG. 8. es un diagrama de flujo que generalmente muestra los pasos que se toman con respecto a completar la petición de I/O en modalidad consulta y en modalidad no-consulta, de acuerdo con una implementación del presente invento;

[0025] La FIG. 9A. es un diagrama que generalmente muestra una representación ejemplar de una petición de interrupción del cliente de acuerdo con una implementación del presente invento;

[0026] La FIG. 9B. es un diagrama que generalmente muestra una representación ejemplar de una respuesta de estatus del servidor que completa la petición de interrupción de acuerdo con una implementación del presente invento;

[0027] La FIG. 10. es un diagrama de flujo que generalmente muestra los pasos a seguir con respecto a una transacción de crédito cliente-servidor de acuerdo con una implementación del presente invento;

[0028] La FIG. 11A es un diagrama que generalmente muestra una representación ejemplar de un mensaje de crédito delta de acuerdo con una implementación del presente invento;

[0029] La FIG. 11B es un diagrama que generalmente muestra una representación ejemplar de un mensaje de crédito cliente-a-servidor de acuerdo con una implementación del presente invento;

[0030] La FIG. 11C es un diagrama que generalmente muestra una representación ejemplar de una respuesta del estatus del servidor que completa una transacción de crédito cliente-servidor de acuerdo con una implementación del presente invento;

[0031] La FIG.12A es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de solicitud de cerrar de acuerdo con una implementación del presente invento;

[0032] La FIG.12B es un diagrama que generalmente muestra una representación ejemplar de una respuesta de estatus del servidor que completa la petición de cerrar de acuerdo con una implementación del presente invento;

[0033] La FIG.13A es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de cancelar de acuerdo con una implementación del presente invento;

B

[0034] La FIG.13B es un diagrama que generalmente muestra una representación ejemplar de una respuesta de estatus del servidor que completa la petición de cancelar de acuerdo con una implementación del presente invento;

[0035] La FIG. 14A es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de lectura en el caso de estar en modalidad de no-consulta de acuerdo con una implementación del presente invento;

[0036] La FIG.14B es un diagrama que generalmente muestra una representación ejemplar de una respuesta de estatus del servidor que completa la petición de lectura en modalidad de no-consulta de acuerdo con una implementación del presente invento;

[0037] La FIG.14C8 es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de lectura en modalidad de consulta de acuerdo con una implementación del presente invento;

[0038] La FIG.14D es un diagrama que generalmente muestra una representación ejemplar de un bloque de estatus de servidor I/O que completa la petición de lectura en el caso de estar en modalidad de consulta, de acuerdo con una implementación del presente invento;

[0039] La FIG. 15A es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de lectura en modalidad de no-consulta de acuerdo con una implementación del presente invento;

[0040] La FIG.15B es un diagrama que generalmente muestra una representación ejemplar de una respuesta de estatus de servidor que completa la petición de escritura en el caso de estar en modalidad de no-consulta, de acuerdo con una implementación del presente invento;

[0041] La FIG.15C es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de escritura en el caso de estar en modalidad de consulta, de acuerdo con una implementación del presente invento;

[0042] La FIG.15D es un diagrama que generalmente muestra una representación ejemplar de un bloque de estatus de servidor I/O que completa la petición de escritura en el caso de estar en modalidad de consulta, de acuerdo con una implementación del presente invento;

[0043] La FIG.16A es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de lectura vectorizada en el caso de estar en modalidad de no-consulta, de acuerdo con una implementación del presente invento;

[0044] La FIG.16Bis es un diagrama que generalmente muestra una representación ejemplar de respuesta de estatus de servidor que completa la petición de lectura vectorizada en el caso de estar en modalidad de no-consulta, de acuerdo con una implementación del presente invento;

[0045] La FIG.16C es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de lectura vectorizada en el caso de estar en modalidad de consulta, de acuerdo con una implementación del presente invento;

[0046] La FIG.16D es un diagrama que generalmente muestra una representación ejemplar de un bloque de estatus de servidor I/O que completa la petición de lectura vectorizada en el caso de estar en modalidad de consulta, de acuerdo con una implementación del presente invento;

[0047] La FIG.17A es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de escritura vectorizada en el caso de estar en modalidad de no-consulta, de acuerdo con una implementación del presente invento;

[0048] La FIG.17B es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de escritura vectorizada en el caso de estar en modalidad de no-consulta, caso colapsado, de acuerdo con una implementación del presente invento;

[0049] La FIG. 17C es un diagrama que generalmente muestra una representación ejemplar de un mensaje de cliente de petición de escritura vectorizada en el caso de estar en modalidad de consulta, caso colapsado, de acuerdo con una implementación del presente invento;

[0050] La FIG. 17D es un diagrama que generalmente muestra una representación ejemplar de respuesta de estatus de servidor que completa la petición de escritura vectorizada en el caso de estar en modalidad de no-consulta, de acuerdo con una implementación del presente invento; y

[0051] La FIG. 17E es un diagrama que generalmente muestra una representación ejemplar de un bloque de estatus de servidor I/O que completa la petición de escritura vectorizada en el caso de estar en modalidad de consulta, de acuerdo con una implementación del presente invento;

DESCRIPCIÓN DETALLADA

[0052] ciertas implementaciónes del la invento presente se discuten a continuación en referencia a las FIGS. 1-17e. Sin embargo, aquellos diestros en el arte apreciarán rápidamente quo la descripción detallada dada aquí con respecto a estas figuras tiene propósitos ilustrativos, y que la invención se extiende más allá de estas implementaciones.

[0053] La FIG. 1 es un diagrama esquemático que generalmente ilustra ciertos rasgos de un ambiente representativo de una red cliente/servidor dentro de la cual se pueden incorporar aspectos del presente invento. En la FIG.1 se muestran dos computadores, llamados Anfitrión A101 y Anfitrión B 121. Mientras la invención se puede aplicar en un ambiente que involucra computadoras de muchos tipos y usos, en un escenario representativo el Anfitrión A 101 funciona como una maquina servidora de aplicaciones cargada con trabajo intensivo de I/O, tal como un servidor de base de datos.

16

[0054] El Anfitrión A 101 y Anfitrión B 121,cada uno, incluye varias tarjetas de interfaz de red (NICs) 109, 111, 113, 133, 135, 137 que permite la comunicación de datos de una máquina al la otra. Entre estos NICs esta NICs 109, 111, 135, 137 permitiendo la trasferencia de datos RDMA. Como se ilustro, un vinculo de red no-RDMA 119 y un canal RDMA 117 se presentan entre los dos anfitriones 101,121.

[0055] Ejecutando en el Anfitrión A esta una aplicación cliente LWIO 103, asociada con una aplicación responsable de procesar tareas de I/O que interactúan con los servicios en modo kernel de lectura/escritura I/O 105. El cliente LWIO 103 se usa para descargar el procesamiento I/O del Anfitrión A 101 al Anfitrión B 121. En el Anfitrión B 121 un servidor LWIO 123 está ejecutando. De acuerdo con el protocolo LWIO descrito acá, el cliente LWIO 103 se comunica con el servidor LWIO 123. El cliente LWIO 103 y el servidor LWIO 123 utilizan los búfers determinados 107, 127, permitiendo transferir directamente datos asociados a archivos a través de una conexión RDMA 117. Las tareas de lectura y escritura se descargan en el Anfitrión B 121a través de mensajes protocolares LWIO. El servidor 123 le pasa una potición de I/O al sistema de archivos 129 que sirve de interfase con el disco duro 131.

[0056] típicamente, dos tipos de mensajes se asocian con una conexión RDMA 117. El primer tipo es una red envia/recibe generando una interrupción en la máquina de destino. El segundo tipo es un RDMA lectura/escritura en donde el espacio de memoria en la máquina remota se accede sin la ayuda del CPU remoto y de esta manera no tiene que generar una interrupción. La CPU remota determina las regiones de memoria que se exponen para RDMA pero típicamente no se da cuenta de cuando ha realizado una operación RDMA.

[0057] en una implementación del invento descrita aqui, el protocolo de LWIO se usa en asociación con otro protocolo de red, como SMB o CIFS para aprovechar la seguridad existente e infraestructura de autenticación del otro protocolo. Esto ayuda a minimizar el overhead del protocolo LWIO. Como se muestra en la FIG. 1, el servidor LWIO 123 en el Anfitrión B 121 opera sobre un servidor SMB 125. Un cliente SMB (no se muestra) en forma similar corre en el Anfitrión B 101 e interactúa con la aplicación cliente LWIO 103.

[0058] el protocolo LWIO comprende dos fases: una fase de descubrimiento seguida por una fase de I/O. En estructuras de datos asociadas con una implementación descrita aqui, el tamaño de los datos es el siguiente:

BYTE	unsigned 8-bit integer
CHAR	8-bit ASCII character
UINT16	unsigned 16-bit integer
UINT32	unsigned 32-bit integer
UINT64	unsigned 64-bit integer

16

```
INT16       signed 16-bit integer
INT32       signed 32-bit integer
INT64       signed 64-bit integer
WCHAR 16-bit Unicode character
PVOID32     32-bit pointer
PVOID64     64-bit pointer
```

[0059] La FIG. 2 ilustra los pasos a seguir en la fase de descubrimiento del protocolo LWIO en una implementación del invento. Con respecto al anfitrión en donde el servidor LWIO ejecuta, en el paso 201 el servidor LWIO se registra con el servidor SMB/CIFS que corre en la máquina anfitrión. De acuerdo con este registro, en el paso 203 el servidor SMB/CIFS notifica a un cliente SMB/CIFS que corre en el anfitrión remoto que el servidor LWIO está disponible. En el paso 205 el cliente LWIO hace una petición de servidor para una llave de continuación. La llave de continuación es un mecanismo de autenticación que se ha expuesto en otra aplicación que tiene la misma asignación que la presente aplicación, "Método y Sistema para acceder a un Archivo(llave de continuación)" Solicitud de Patente, No, serial_____ archivada el 24 de Octubre de 2003, que se incorpora acá en su totalidad por referencia.

[0060] En el paso 207 el servidor LWIO le pasa la petición de una llave de continuación al cliente. En una implementación del invento la petición del servidor para una llave de continuación tiene la siguiente estructura:

```
typedef struct _SRV_RESUME_KEY {
UINT64 ResumeKey;
UINT64 Timestamp;
UINT64 Pid;
} SRV_RESUME_KEY, *PSRV_RESUME_KEY;
typedef struct _SRV_REQUEST_RESUME_KEY {
SRV_RESUME_KEY Key;
UINT16 ContextLength;
BYTE Context[1];
} SRV_REQUEST_RESUME_KEY, *PSRV_REQUEST_RESUME_KEY;
```

La FIG. 3. muestra una representación de la petición del servidor para una llave de continuación 219. ResumeKey 221, Timestamp 223, y el Pid 225 que se generan en el servidor son opacos al cliente. Contexto 229 es una matriz que tiene un nombre UNC que el cliente LWIO utiliza para hacer contacto con el servidor. ContextLength 227 es el numero de bites en Context 229.

Descubrimiento de Red

18

[0061] Cuando la aplicación cliente recibe la petición del servidor para la llave de continuación 219, recupera el nombre UNC del servidor del campo de Contexto 219. Regresando a la FIG. 2, en el paso 209, el cliente abre un canal al servidor LWIO. El canal se utiliza para descubrir automáticamente dispositivos capaces de RDMA que estén disponibles en la red, como se describe mas abajo. Este es un rasgo importante y útil del presente invento; mecanismos de resolución de dirección como el ARP generalmente están ausentes en redes VIA y otras similares.

[0062] El cliente luego consulta al servidor para una lista de dispositivos capaces de RDMA ("provcedores") que están disponibles para utilizar con el protocolo LWIO. La consulta se efectúa a través de una petición negociada, que el cliente construye y envía al servidor a través del canal recientemente abierto. En una implementación del presente invento, la petición negociada tiene la siguiente estructura:

```
typedef struct {
LWIO_CONTROL HEADER;
WCHAR      ClientName[LWIO_MAX_HOST_NAME];
UUID       Key;
UINT16     ResponseLength;
UINT6      ProviderCount;
LwioAddressBlk_tproviderlist (1);
} LwioNegotiateRequest_t;
typedef struct {
    CHAR        ProtocolId[4];
    UINT32      RevId;
    UINT16      Opcode;
    UINT16      Length;
} LWIO_CONTROL_HEADER;
typedef struct _GUID {
    UINT32      Data1;
    IJINT16     Data2;
    UINT16      Data3;
    BYTE        Data4[8];
} GUID, UUID;
typedef struct {
    WCHAR Name[LWIO_MAX_PROVIDER_NAME];
    UINT16      InstanceCount;
```

15

```
    LWIO_NET_ADDRESS InstanceTable[1];
} LwioAddressSlk_t;
typedef struct _LWIO_NET_ADDRESS {
UINT16 HostAddressLen;
UINT;6 DiscriminatorLen;
BYTE HostAddressFollowedByDiscriminator[1];
} LWIO_NET_ADDRESS;
```

[0063] La FIG. 4A nos muestra, en una implementación del invento, una representación ilustrativa del paquete de petición 231. La petición de negociación incluye un encabezado de control 233, un campo de nombre de cliente Unicode de largo fijo 235, un UUID 237 que se utiliza como llave, un tamaño de búfer local 239 para recibir la respuesta, y la lista de proveedores 241. En el encabezado de control 233, el protocolo "LWIO" se guarda como los cuatro primeros bites del encabezado.

[0064] RevId 245 tiene un valor definido actual 0x[001, LWIO_REV_ID. Opcode 247 tiene un valor definido actual Oxfe, LWIO_CONTROL_OPCODE_NEGOTIATE. Length 249 es el tamaño en bites del paquete completo para ser enviado al servidor, incluyendo todos datos específicos opcode.

[0065] El servidor utiliza el ClientName 235 para identificar al cliente. La llave 237 se utiliza en procedimientos específicos de autenticación en red, tal como se describe más abajo. ResponseLength 239 es el tamaño del búfer para recibir una respuesta de negociación del servidor, como se describe más abajo. ProviderCount 251 es el numero de proveedores asociados con la maquina del cliente y sobre la cual el cliente esta informando al servidor. La lista de proveedores 241 contiene la lista de proveedores ProviderCount.

[0066] en un elemento de la lista de proveedores 241, Name 253 nos da el nombre del proveedor. Para detectar redes que sean compatibles, el cliente y el servidor preferiblemente utilizan el mismo nombre de proveedor. InstanceCount 255 es el numero de dispositivos del mismo tipo del proveedor. La tabla de instancias 257 es una tabla de pares red/discriminadores, en donde un par sirve para describir, en forma especifica a los dispositivos, de como se puede hacer una conexión remota. HostAddressLen 259 es el largo de la dirección del anfitrión especifica a la red 263. DiscriminatorLen 261 es el largo de discriminador especifico a la red 265. Siguiendo a estos campos de largo esta HostAddressLen bites, la dirección del anfitrión 263 y el DiscriminatorLen bites del discriminador 265.

[0067] Volviendo a la FIG. 2, una vez se ha recibido la respuesta negociada con la lista de los proveedores del cliente, en el paso 213 el servidor determina que dispositivos capaces de comunicación RDMA tienen en común con el cliente. En el paso 215 el servidor envía una respuesta negociada al cliente a

través del canal, que incluye una lista de proveedores que se comparten. En una implementación del presente invento, la respuesta negociada tiene la siguiente estructura:

```
typedef struct {
LWIO_CONTROL HEADER;
  WCHAR SrvName[LWIO_MAX_HOST_NAME];
  UUID  Key;
  UINT16      ProviderCount;
  LwioAddressBlk_t  ProviderList[1];
} LwioNegotiateResponse_t;
```

[0068] La FIG. 4B nos muestra una representación ilustrativa de la respuesta negociada 267 en una implementación del presente invento. El encabezado de control 269 es una petición negociada, con excepción de Length 271, ahora refleja el tamaño del mensaje de respuesta 267. SrvName 273 tiene el nombre del servidor. Key 275 es un GUID generado por el servidor para uso del cliente. Como se explica más abajó, el cliente envía la llave de vuelta al servidor en una petición autenticada a través de una conexión nueva utilizando uno de los dispositivos de comunicación que tienen en común. ProviderCount 277 es el numero de proveedores en la lista de proveedores 279. La lista de proveedores 279 tiene una lista de proveedores comunes al servidor y al cliente. No hay garantía de que el cliente se pueda conectar con estos proveedores.

[0069] Regresando a la FIG. 2, en este punto el servidor y el cliente tienen información sobre dispositivos de comunicación que comparten, y la lista mínima de proveedores comunes se ha determinado. En el paso 217 el cliente crea una o más conexiones RDMA con el servidor LWIO a través de uno o más dispositivos que se comparten. en una implementación del presente invento, tal como se describe en este documento los siguientes opcodes se definen para la comunicación cliente-a-servidor:

```
#define LWIO_OPCODE_READ            0x0
#define LWIO_OPCODE_WRITE           0x1
#define LWIO_OPCODE_VEC_READ        0x2
#define LWIO_OPCODE_VEC_WRITE       0x3
#define LWIO_OPCODE_CLOSE           0x4
#define LWIO_OPCODE_CANCEL          0x5
#define LWIO_OPCODE_AUTH            0x6
#define LWIO_OPCODE_REGISTER        0x7
#define LWIO_OPCODE_CREDIT          0x8
#define LWIO_OPCODE_INTERRUPT       0x9
```

Las siguientes banderas definidas se utilizan como modificadores en una comunicación cliente-a-servidor:

```
#define LWIO_HDR_FLAG_INTERRUPT     0x80
#define LWIO_HDR FLAG CONTROL       0x40
#define LWIO HDR_FLAG_COLLAPSE_IO   0x20
```

El mensaje correspondiente cliente a servidor en el protocolo LWIO tienen la misma estructura de encabezado. En una implementación del presente invento el encabezado común tiene el siguiente formato :

```
typedef struct {
UINT32          Length;
union {
UINT32          Status;
struct {
    BYTE        Opcode;
    BYTE        Flags;
    BYTE        Credits;
    BYTE        Marker;
} i
    } ;
struct i
    UINT16          Fid;
    UINT16          Sequence;
    UINT32          Tid;
} i
UINT64          Offset;
// data búfer block
PVOID64   DataVa;
union {
UINT32    DataMh;
Strict {
    UINT16      NumPages;
    UINT16      PageSize
} Ve?c;
```

```
    } i
    } i
    // is status block
    union {
    struct {
        UINT32      IosMh;
        PVOID64     Iosvai
}i
    struct {
        UINT32      ImmediateCookie
        UINT64      Cookies
    } i } i } LWIO_COMMON_HEADER;
```

Autenticación de la conexión

[0070] La FIG. 5 muestra los pasos que el cliente y el servidor toman durante el resto de la fase inicial del protocolo LWOI, en una implementación del presente invento. En el paso 601 como se explico anteriormente, el cliente establece una conexión con el servidor a través de un dispositivo de comunicación que se comparte. El cliente y el servidor autentican mutuamente la nueva conexión. En el paso 603 el cliente envía un mensaje de petición de autenticación (LWIO_OPCODE_AUTH) al servidor. La autenticación se hace para prevenir engaños del lado del servidor o del cliente. Si la autenticación no se completa a tiempo, la conexión se termina.

[0071] La FIG. 6A nos da una representación ilustrativa de un mensaje de petición de autenticación en una implementación del presente invento. El mensaje de autenticación 617 comprende un encabezado común 619 seguido de una estructura LWIO_AUTH_PARAMS 621. En el encabezado 619, Length 623 es el numero de bites enviados al servidor (el tamaño del encabezado común 619 mas el tamaño del LWIO_AUTH_PARAMS 621). El Opcode 625 se fija en LWIO_OPCODE_AUTH (0x6). Las banderas 627 se fijan en LWIO_HDR_FLAG_INTERRUPT. Cookie 629, en este y otros mensajes de protocolo, se fija con un valor que el cliente determina y se envía de vuelta en la respuesta del servidor. El valor de Cookie se usa tipicamente para que corresponda entre una peticion y una respuesta del servidor. La DataVa 631 se fija con la dirección a la que el servidor debe RDMA los parámetros de autenticación del servidor. DataMh 633 es la manija de la memoria RDMA asociada con DataVa 631.

[0072] en una implantación del presente invento, la estructura LWIO_AUTH_PARAMS tiene el siguiente formato:

```
#define LWIO_AUTH_OPTION_END      0
#define LWIO_AUTH OPTION KEY      1
#define LWIO_AUTH_OPTION_SESSION_ID     2
#define LWIO_AUTH_OPTION SIGNATURE      3
#define LWIO_AUTH_OPTION_KEY_LENGTH     16
#define LWIO_AUTH_OPTION_SESSION_ID_LENGTH 8
#define LWIO_AUTH OPTION SIGNATURE LENGTH 16_ _
typedef struct { UCHAR OptionCode; UCHAR OptionLen; BYTE OptlonData[1];
) LWIO_AUTH OPTIONS, *LPLWIO AUTH OPTIONS;_
typedef struct {
    CHAR  Magic[4]; // 'LWIO'
    UINT16      RevId;
    UINT16      Endian;
    UINT16      PageSize;
    UINT16      BaseSequence;
    UINT32      MaxRdmaWindowSize;
    UINT32      MaxSendBúfersize;
    UINT32      MaxRecvBúfersize;
    UINT16      HeaderSize;
    UINT16      Credits;
    UINT16      RdmaReadSupported;
LWIO_AUTH_OPTIONS Options[1];
} LWIO_AUTH_PARAMS, *LPLWIO_AUTH_PARAMS;
```

[0073] en un mensaje de autenticación 617 un LWIO_AUTH_PARAMS 621 forma parte del segundo paquete. Magic 635 se fija en 'LWIO'. RevId 637 se fija en LWIO_REV_ID. Endian 639 se fija en sizeof(ULONG_PTR). PageSize 641 se fija en el tamaño de pagina de la CPU (4k en maquinas de 32-bits y 8k en maquinas de 64-bits). BaseSequence 643 se fija en 0. MaxRdmaWindowSize 645 se fija en el numero máximo de bites que el cliente puede aceptar en una transferencia RDMA; en la implementación que se ilustra se fija en 64K. MaxSendBúfersize 647 se fija en el numero de bites que el cliente puede enviar al servidor en una sola petición; en la implementación que se ilustra se fija en 1K . MaxRecvBúfersize 649 se fija en los números de bites que el cliente ha anunciado para recibir datos del servidor; en la implementación que se ilustra se fija en 16K. HeaderSize 651 se fija en el numero de bites del encabezado de control LWIO 619 .

Credits 652 se fija en el numero de créditos búfer que el cliente desea tener. El uso de créditos se explica mas abajo. El servidor puede o no satisfacer la petición del cliente. RdmaReadSupported 653 se fija en 0 si el cliente no sustenta lecturas RDMA y se fija en 1 si el cliente si sustenta lecturas RDMA.

[0074] Parte de la estructura LWIO_AUTH_PARAMS es un conjunto de una o más opciones. Las opciones se utilizan para hacer la autenticación mas flexible. Cada opción tiene un código de opción, largo y datos, a excepción de la última opción en cada lista, LWIO_AUTH_OPTION_END, que únicamente tiene el código de opción, que sirve como una opción nula que termina la lista de opciones. En el mensaje de autenticación, el cliente le envía al servidor las siguientes opciones: La Llave (LWIO_AUTH_OPTION_KEY) y una firma (LWIO_AUTH_OPTION_SIGNATURE). La llave 655 se fija en la llave previamente devuelta por el servidor en la respuesta negociada. La Firma 657 es una firma MD5 del LWIO_AUTH_PARAMS 621 que excluye la firma.

[0075] Regresando a la FIG. 5, en el paso 605, si la llave enviada en el mensaje de autenticación corresponde a la llave devuelta en la respuesta negociada a través del canal, el servidor le hace un RDMA al cliente, como respuesta de autenticación, una estructura LWIO_AUTH_PARAMS, que incluye un SessionId de ocho bites, la dirección DataVa asociada a la manija de memoria DataMh provista por el cliente en el mensaje de autenticación. En el paso 607 el servidor envía una LWIO_MSG_STATUS_RESPONSE para completar la autenticación.

[0076] La FIG. 6B nos muestra una representación ilustrativa de la estructura LWIO_AUTH_PARAMS 659 devuelta por el servidor en una implementación del presente invento. Magic 661 se fija en 'LWIO'. RevId 663 se fija en LWIO_REV_ID. Endian 665 se fija en sizeof(ULONG_PTR). PageSize 667 se fija en el tamaño de pagina de la CPU. Se pretende fijar la BaseSequence 669 en(client BaseSequence + 1). Se pretende fijar MaxRdmaWindowSize 671en el numero máximo de bites que el cliente puede aceptar en una transferencia RMA; en el presente invento se fija en 512K. Se pretende fijar MaxSendBúfersize 673 en el numero de bites que el servidor envía al cliente en una sola respuesta; en la presente implementación se fija en 16 bites. Se pretende fijar MaxRecvBúfersize 675en el numero de bites que el servidor ha prefijado para recibir datos del cliente; en la presente implementación se fija 8k. HeaderSize 677 se fija en el numero de bites que tiene el encabezado común. Credits 679 se fija en el numero inicial de créditos que el servidor tiene disponibles para el cliente. RdmaReadSupported 681 se fija en 0 si el servidor no es compatible con lecturas RDMA y se fija en 1 si el servidor es compatible con lecturas RDMA. El servidor envía las siguientes opciones: Key (LWIO_AUTH_OPTION_KEY) 683, SessionId (LWIO_AUTH_OPTION_SESSION_ID) 685, y una firma (LWIO_AUTH_OPTION_SIGNATURE) 687. La Llave 683 se fija en Key que el cliente ha enviado previamente en la petición negociada. El cliente utiliza el valor de SessionId 685 para registrar los archivos

del cliente con el servidor, tal como se explica más abajo. La Firma 687 es un acto de firmar MD5de los LWIO_AUTH_PARAMS que excluyen la firma.

[0077] en una implementación del presente invento, la estructura LWIO_MSG_STATUS_RESPONSE tiene el siguiente formato:

```
typedef struct LWIO IO STATUS BLOCK { _
VINT32 Information;
UINT32 Status;
} LWIO_IO_STATUS_BLOCK, *LPLWIO_IO_STATUS_BLOCK;
typedef struct _LWIO_MSG_STATUS_RESPONSE {
UINT64 Cookie;
LWIO_IO_STATUS_BLOCK Ios;
} LWIO_MSG_STATUS_RESPONSEt *LPLWIO_MSG_STATUS_RESPONSE;
```

La FIG. 6C nos muestra una representación ilustrativa del LWIO_MSG_STATUS_RESPONSE 689 devuelto por el servidor para completar la autenticación en una implementación del presente invento. El valor de la Cookie 691 lo fija el cliente en el encabezado del mensaje de autenticación. La Información 693 se fija en el numero de bites de LWIO_AUTH_PARAMS mas ocho bites mas. El Status 695 se fija en 0x0 (que significa éxito) o 0xC0000022 (que significa que el acceso a sido denegado).

El registro de archivos

[0078] regresando a la FIG. 5, en el paso 609, cuando la nueva conexión ha sido mutuamente autenticada por el cliente y el servidor, el cliente comienza a registrar los archivos que se van a utilizar con el servidor. Las operaciones con un archivo no se procesan a través de un vinculo hasta tanto el cliente no haya registrado los archivos que se van a utilizar con el servidor.

[0079] La FIG. 7A nos muestra una representación ilustrativa del mensaje de registro por parte del cliente con el servidor en una implementación del presente invento. El mensaje de registro 701 consta del encabezado común 703 seguido de una estructura LWIO_FID_PARAMS 705. El Length 707 se fija en el numero de bites enviados al servidor (el tamaño del encabezado 703 más el tamaño de LW1O_FID_PARAMS 705). El Opcode 709 se fija en LWIO_OPCODE_REGISTER (0x7). Flags 711 se fija en LWIO_HDR _FLAG_INTERRUPT. En este mensaje de cliente y los subsiguientes, Credits 713 es el numero de peticiones I/O pendientes con el cliente. El campo Crédito sirve como una sugerencia al servidor para asignar más créditos a la conexión, de tal manera que se permitan más peticiones I/O excepcionales, como se explica más abajo. El numero de peticiones excepcionales no puede superar en ningún momento el valor de "Credits". Como antes, el valor de Cookie 715 se fija en un valor determinado por el cliente.

[0080] en una implementación del invento, la estructura LWIO_FID_PARAMS tiene el siguiente formato:

```
typedef struct {
  SRV_RESUME_KEY    ResumeKey;
  INT64 SessionId;
  UINT32       FlagsAndAttributes;} LWIO_FID_PARAMS, *LPLWIO_FID_PARAMS;
```

El LWIO_FID_PARAMS 705 del mensaje de registro de archivo 701, ResumeKey 717 se fija en el resume key que el servidor regreso a través del canal inicial de acceso a los archivos. SessionId 719 se fija en el SessionId que el servidor regreso durante la etapa de autenticación de la conexión. FlagsAndAttributes 721 se fija en Win32 Create Flags utilizado primero para abrir el archivo.

[0081] Regresando a la FIG. 5, en el paso 611 el servidor responde con un LWIO_MSG_STATUS_RESPONSE para completar el registro de archivos. La FIG. 7B nos muestra una representación ilustrativa de la LWIO_MSG_STATUS_RESPONSE 723 enviada por el servidor en una implementación del presente invento. Information 725 se fija en el Fid (File ID) que se utilizara para enviar la peticiones I/O. El Status 727 se fija en 0x0 (éxito) o en otro código de denegación NTSTATUS. Cookie 729 se fija en el valor de cookie que el cliente fija en el encabezado del mensaje de registro de archivos.

Procesamiento I/O

[0082] en este punto, la conexión del cliente se ha establecido, los archivos han sido registrados y la etapa de procesamiento I/O del protocolo LWIO comienza. Uno de los rasgos sobresalientes de la implementación del protocolo LWIO es el modelo asimétrico I/O para lecturas y escrituras. Las operaciones de lectura se implementan utilizando RDMA, mientras que las de escritura se implementan utilizando operaciones de envió. Las operaciones de escritura no se implementan utilizando RDMA para garantizar un modelo más seguro. Si el servidor expone su espacio de dirección en el NIC para RDMA, introduce una vulnerabilidad a la corrupción de datos que podría ser aprovechada por el cliente malicioso. En ese escenario, el cliente malicioso emite, en forma circular, operaciones de escritura RDMA en la dirección virtual del servidor. Porque el espacio serverIddress es finito y en algún momento la dirección virtual del servidor se tiene que volver a utilizar, el cliente malicioso puede eventualmente encontrar al servidor utilizando la misma dirección virtual para una conexión diferente, haciendo que los datos se escriban en un búfer del servidor que se puede asociar con un cliente diferente. El modelo asimétrico de I/O en el protocolo LWIO previene esta posibilidad. Este rasgo es una de las diferencias principales entre un protocolo LWIO y otros utilizando transferencias de archivos basadas en protocolos RDMA, tal como el DAFS.

[0083] Regresando a la FIG. 5. en el paso 613, el cliente a anuncia el procesamiento I/O en modalidad de consulta. En modalidad de no consulta, las terminaciones I/O están basadas en interrupciones, utilizando

mensajes comunes de envia/recibe. En modalidad de consulta, las terminaciones I/O utilizan RMDA y no están basadas en interrupciones.

[0084] El diagrama de flujo de la FIG. 8 generalmente ilustra, desde la perspectiva general del servidor LWIO, los pasos a seguir, en una implementación del presente invento, con respecto a completar una demanda de I/O en modalidad de consulta o de no-consulta. Una petición I/O de un cliente especifica si el servidor debe enviar de vuelta un anuncio de envió (interrumpiendo la CPU) o un mensaje RDMA. En el paso 801 el servidor determina si una bandera LWIO_HDR_FLAG INTERRUPT se ha fijado en el encabezado común del mensaje de peticion I/O del cliente. Si la bandera se ha fijado, en el paso 803 el servidor completa la petición del cliente por medio de un LWIO_MSG_STATUS_RESPONSE utilizando un envió ordinario. Si la bandera LWIO_HDR_FLAG_INTERRUPT no esta fija (en modalidad de consulta), entonces el servidor completa la peticion del cliente enviando un RDMA LWIO_IO_STATUS_BLOCK al cliente, como se indica en el paso 805.

Despertado del cliente en modalidad de consulta

[0085] en modalidad de consulta, el cliente puede dormir mientras espera que se termine un I/O del servidor. En este caso la terminación se envía a través de un RDMA al cliente, por lo que se necesita un mecanismo que despierte al cliente para notificarlo de que ha terminado. Si el cliente quiere que lo despierten, envía una solicitud de interrupción (LWIO_OPCODE_INTERRUPT) al servidor, que el servidor recibe en el paso 807 de la FIG.8. Un servidor que recibe una solicitud de interrupción no enviara una respuesta hasta que la solicitud de I/O se haya terminado en el servidor (paso 809). La terminación se le envía al cliente en el paso 811 por medio de un envió corriente, interrumpiendo al cliente. Solo un mensaje de interrupción puede ser sobresaliente para una conexión del cliente.

[0086] La FIG. 9A nos muestra una representación ilustrativa del mensaje de solicitud de una interrupción enviada por el cliente al servidor en una implementación del presente invento. El mensaje consta de un encabezado común X 15. El Opcode 817 se fina en LWIO_OPCODE_REGISTER (0x9). Las Flags 819 se fijan en (LWIO_HDR_FLAG_INTERRUPT | LWIO_HDR_FLAG_CONTROL) (0xC0). Credits X21 en el numero de peticiones I/O solicitadas al cliente y Cookie X23 se fija en un valor especificado por el cliente.

[0087] el servidor responde con un mensaje de solicitud de interrupción luego de que otros proceso I/O han sido solicitados. La FIG. 9B nos muestra una representación ilustrativa del mensaje LWIO_MSG_STATUS_RESPONSE 825 enviado por el servidor en una implementación del presente invento. La Informacion X27 se fija en 0. El Status 829 se fija en 0x0 (éxito) o en otro código NTSTATUS code 011 (denegado). Cookie 831se fija en el valor Cookie en el encabezado de la solicitud de interrupción enviada por el cliente.

Creditos

[0088] como se ha anotado, todas las solicitudes cliente-servidor de I/O incluyen un campo de crédito en el encabezado. El campo de crédito es una sugerencia al servidor en cuanto al numero de solicitudes I/O sobresalientes que el cliente quisiera enviar al servidor. Es la responsabilidad del servidor manejar los créditos. Los créditos son una solución novedosa al problema de vaciar los búfers. Si el cliente actualmente tiene N créditos, es necesario anunciar N+1 búfers de recepción para que el servidor le envié al cliente un mensaje de crédito. El servidor solo tiene una solicitud de crédito sobresaliente a través de una conexión con un cliente en un determinado momento. Los mensajes de crédito siempre se envían en modalidad de interrupción.

[0089] Una transacción de crédito comprende un saludo de tres vias iniciado por el servidor entre un cliente y un servidor. La FIG. 10 generalmente ilustra los pasos que comprenden una transacción de crédito en una implementación del invento. En el paso 1001 el servidor envía un mensaje de solicitud de crédito delta a través de una conexión con un cliente.

[0090] La FIG. 11A nos muestra una representación ilustrativa del mensaje de crédito delta en una implementación del invento. Este mensaje toma la forma de un LWIO MSG_STATUS_RESPONSE 1011. Los créditos corresponden a los búfers. La información 1013 se fija en el numero de créditos que el cliente debe entregar (un numero negativo) o el numero de créditos (búfers extras) que el servidor ha asignado para uso del cliente (un numero positivo). Status 1015 se fija en LWIO_NOTIFY_CREDIT (0xl). Cookie 1017 se fija en 0.

[0091] Regresando a la FIG. 10, el cliente recibe el mensaje de crédito del servidor. El cliente debe responder con un mensje LWIO_OPCODE_CREDIT al servidor a través de la misma conexión. Este mensaje significa la liberación de un sólo crédito o la notificación al servidor del numero de créditos asignados recientemente para uso del cliente. Si el campo de Información en el mensaje de crédito del servidor contiene un numero negativo, -N (paso 1003), el cliente envía un mensaje N LWIO_OPCODE_CREDIT (uno por cada crédito que tenga que entregar) indicado como paso 1005. Si el campo de información es positivo, entonces el cliente envia un solo mensaje LWIO_OPCODE_CREDIT que se indica con el paso 1007.

[0092] la FIG. 11 B nos muestra una representación ilustrativa, en una implementecion del invento, de un mensaje LWIO_OPCODE_CREDIT enviado por un cliente. El mensaje LWIO_OPCODE_CREDIT 1019 comprende un encabezado común 1021. Opcode 1023 se fija en LWIO OPCODE_CREDIT (0x8). Flags 1025 se fija enLWIO_HDR FLAt,_INTERRUPT (0x80). Credits 1027 en el numero de solicitudes de I/O pendientes con el cliente. Cookie 1031 se fija en un valor especificado por el cliente. Si el cliente recibe un mensaje de crédito delta positivo, los 32 bits superiores del FOCET 1029 se fijan en el numero de

créditos asignados por el servidor que el cliente no utilizo. Una vez que el cliente devuelve un valor superior a cero en este campo, el servidor normalmente no enviara otro mensaje de actualización positivo hasta que, por lo menos una actualización negativa haya sido enviada. En forma típica el cliente regresa un cero.

[0093] Como se anoto anteriormente, si el cliente recibió un mensaje de crédito delta negativo (-N), el cliente debe enviar N mensajes de crédito al servidor, uno por cada crédito que entrega. Los 32 bits superiores del Offset 1029 en este caso se fijan de acuerdo a –N, -(N-1), . . ., -1. Cuando el servidor recibe el mensaje de crédito del cliente con los 32 bits superiores del FOCET 1029 fijados en –1, el servidor asume que el cliente ha terminado de procesar el mensaje de crédito del servidor y esta dispuesto a recibir nuevos mensajes de crédito.

[0094] Regresando a la FIG. 10, el servidor termina el saludo a tres vías enviando un mensaje LWIO MSG_STATUS_RESPONSE como se indica en el paso 1009. La FIG. 11C muestra en forma ilustrativa, en una implementación del presente invento, una representación del LWIO MSG_STATUS_RESPONSE 1033 enviado por un servidor. Information 1037 se fija en 0. Si los 32 bits superiores del Offset en el encabezado del mensaje LWIO_OPCODE_CREDIT enviado por el cliente era mayor o igual a cero, Status 1039 se fija en 0x0, que significa éxito. Si los 32 bits superiores del Offset se fijan en un numero negativo, el servidor fija el Status 1039 en LWIO_CREDIT_NOTIFY para permitir al cliente retirar el crédito. Cookie 1035 se fija en el valor de Cookie fijado por el cliente en el encabezado común del mensaje LWIO_OPCODE_CREDIT.

Cerrar

[0095] El mensaje de cerrar se usa para detener el proceso I/O para un FID particular que se intercambio durante la etapa de registro. Una vez que el servidor responde, cualquier solicitud nueva fracasara hasta que el Fid se recicla. La FIG. 12A nos muestra, en una implementación del invento, una representación ilustrativa del mensaje de cerrado enviado por el cliente. El mensaje de cerrar 1041 consta de un encabezado común 1043. Opcode 1045 se fija en LWIO_OPCODE_CLOSE (0x4). Flags 1047se fija en LWIO_HDR_FLAG_INTERRUPT (0x80). Credits 1049 en el numero de solicitudes I/O pendientes con el cliente. Cookie 1053 se fija en un valor determinado por el cliente. Fid 1051 se fija en File id del archivo que se va a cerrar.

[0096] el servidor responde con un LWIO_MSG_STATUS_RESPONSE. La FIG. 12B muestra una representación ilustrativa, en una implementación del invento, la finalización del cerrado LWIO_MSG_STATUS_RESPONSE 1055 devuelta por el servidor. Information 1059 se fija en 0. Status 1061 se fija en 0, que indica éxito. Cookie 1057 se fija en el valor de Cookie que se fijo en la solicitud del cliente para cerrar.

Cancelar

[0097] El mensaje de cancelar se usa para detener el procesamiento I/O para un Fid particular que se intercambio en la etapa de registro. Cuando cancelar se emite, el servidor completa la solicitud. Sin embargo solicitudes I/O que no se pueden cancelar pueden continuar en el servidor. La FIG. 13A nos muestra, en una implementación del invento, una representación ilustrativa del mensaje de cancelación enviado por el cliente. El mensaje de cancelar 1063 contiene un encabezado común 1065. Opcode 1067 se fija en LWIO_OPCODE_CANCEL (0x5). Flags 1069 se fija en LWIO_HDR FLAG_INTERRUPT (0x80). Credits 1071 se fija en el numero de solicitudes I/O pendientes en el cliente. Cookie 1075 se fija en un valor determinado por el cliente. Fid 1073 se fija en el File Id en el cual cancelar es emitido.

[0098] el servidor termina la cancelación con un mensaje LWIO_MSG_STATUS_RESPONSE. La FIG. 13B nos muestra una representación ilustrativa de la terminación de la cancelacion LWIO_MSG_STATUS_RESPONSE 1077. Information 1081 se fija en 0. Status 1083 se fija en 0, que indica exito. Cookie 1079 se fija en el valor de Cookie que se fijo en la solicitud del cliente de cancelar.

[0099] el mensaje de lectura se usa para obtener los datos de un Fid particular que se intercambió durante la fase de registro. Para una solicitud de lectura de menos de un kilobyte, si el búfer del usuario no se ha registrado con el NIC, los datos se reciben en un búfer pre-registrado interior, y se hace una copia en el búfer del usuario una vez se reciben los datos del servidor. Esto se hace porque es más eficiente copiar pequeñas cantidades de datos que registrar búfers pequeños de usuario. Para grandes lecturas se registra el búfer del usuario y los datos se reciben directamente por medio de una escritura RDMA. La cantidad de datos que se pueden leer en una sola petición de lectura esta limitada por el MaxRdmaWindowSize del servidor.

[0100] La FIGS. 14A y 14C nos muestra, en una implementación del invento, una representación ilustrativa de un mensaje de lectura enviado por un cliente, con la FIG 14A mostrando el caso de consulta y la FIG.14C mostrando el caso de no-consulta. El mensaje de lectura 1401 comprende un encabezado común 1403. Length 1405 se fija en el numero de bites que se van a leer del archivo asociado. Opcode 1407 se fija en LWIO_OPCODE READ (0x0). Offset 1417 Se fija en la localidad del bit donde la lectura comenzo. El Marker 1413 se fija en 0xFF. Flags 1409, 1427 se fijan en 0x0 en el caso de consulta 1427 o LWIO_HDR_FLAG_INTERRUPT (0x80) en el caso de no consulta 1409. Los Credits 1411 se fijan en el numero de solicitudes I/O al cliente pendientes. Fid 1415 se fija en el File Id en que se emite el I/O. DataVa 1419 se fija en la dirección a donde los datos leidos se van a RDMA, y DataMh 1421 se fija en la manija de la memoria asociada.

[0101] en el caso de no consulta, ImmediateCookie 1423 y Cookie 1425 se fijan en valores determinados por el cliente. El servidor en este caso puede completar una solicitud de lectura con un

LWIO_MSG_STATUS_RESPONSE a través de un envió normal, o con un RDMA con datos inmediatos si la lectura es exitosa. Los datos inmediatos de la escritura RDMA se fijan de acuerdo al valor de ImmediateCookie de la solicitud de lectura. En la modalidad de consulta losVa 1431 se fijan en el lugar al cual la respuesta del estatus del servidor(LWIO_IO_STATUS_BLOCK) se RDMA, y losMh 1429 se fijan en la manija de la memoria asociada.

[0102] en el caso de no consulta, el servidor primero RDMA los datos de lectura. El servidor luego puede responder con un mensaje LWIO_MSG_STATUS_RESPONSE, o el servidor puede enviar datos inmediatos con los datos de la lectura RDMA, caso en el cual los datos inmediatos se fijan en el valor ImmediatCookie de la solicitud de lectura. La FIG. 14B nos muestra, de acuerdo a una implementación del invento, una representación ilustrativa de laLWIO_MSG_STATUS_RESPONSE 1433 devuelto por el servidor en el caso no-consulta.

[0103] Information 1437 se fija en el numero de bites que se leyeron. Status 1439 se fija en 0, indicando éxito o en NTSTATUS, indicando fracaso. Cookie 1435 se fija en el valor Cookie determinado por el cliente en el encabezado del mensaje de lectura.

[0104] En el caso de consulta, el servidor primero RDMA los datos de lectura. El servidor luego RDMA un LWIO_IO_STATUS_BLOCK al cliente. La FIG. 14D nos muestra una representación ilustrativa del LWIO_IO_STATUS_BLOCK 1441devuelto por el cliente en una implementación del invento. Information 1443 se fija en el numero de bites que se han leído. Status 1445 se fija en 0, lo que indica éxito, o en NTSTATUS, que indica fracaso.

Escritura

[0105] El mensaje de escritura se utiliza para colocar datos en un Fid particular que se intercambio durante el registro de archivos. Todos los datos de escritura se envían utilizando operaciones normales de envió. La cantidad de datos escritos esta limitada por el MaxReovBúfersize del servidor. Si el cliente envía mas datos, la conexión se termina.

[0106] La FIGS. 15A y 15C nos muestra, en una implementación del invento, una representación ilustrativa de un mensaje de escritura, con la FIG 15A ilustrando el caso de no-consulta y la FIG 15C mostrando el caso de consulta. El mensaje de escritura incluye un encabezado común 1503. Length 1505 se fija en el numero de bites de datos que se van a escribir. Opcode 1507se fija en LWIO_OPCODE_WRITE (0x 1). Offset 1517se fija en la localidad del bit donde se va a comenzar la escritura de los datos. Flags 1509, 1529 se fijan en 0x0 en el caso consulta 1529 y en LWIO_HDR FLAG_INTERRUPT (0x80) en el caso no-consulta 1509. Marker 1513 se fija en 0xFF. Credits 1511 se fija en el numero pendiente de solicitudes I/O al cliente. Fid 1515 se fija en el File id en el cual se emite el I/O. Los datos que se van a escribir 1527 siguen inmediatamente al encabezado común 1503 del mensaje de escritura.

[0107] en el caso no-consulta, Cookie 1525 se fija en un valor determinado por el cliente. En el caso de consulta, losVa 1533se fija en la localidad en que la respuesta del estatus del servidor (LWIO_IO_STATUS_BLOCK) se RDMA, y losMh 1531 se fija en la manija asociada de la memoria.

[0108] En el caso de no-consulta, el servidor responde al mensaje de escritura LWIO MSG_STATUS_RESPONSE. La FIG. 15B nos muestra, en una implementación del invento, una representación ilustrativa del LWIO_MSG_STATUS_RESPONSE 1535 devuelto por el servidor. Information 1539 se fija en el numero de bites escritos. Status 1541 se fija en 0, que indica éxito , o en otro NTSTATUS, que indica fracaso. Cookie 1537 se fija en el valor Cookie fijado por el cliente en el encabezado del mensaje de escritura. En el case de consulta el servidor RDMA un LWIO_IO_STATUS_BLOCK. La FIG. 15D nos muestra una representación ilustrativa del LWIO_IO_STATUS_BLOCK 1543 devuelto por el servidor en una implementación del invento. Information 1545 se fija en el numero de bites escritos. Status 1547 se fija en 0, que indica éxito, o en otro NTSTATUS, que indica fracaso.

Lectura Vectorizada

[0109] la lectura vectorizada se usa para obtener los datos de un Fid particular que se intercambió durante la fase de registro y para esparcir los datos en base a páginas en segmentos múltiples en el solicitante. Todos los datos leídos son enviados al solicitante a través de una escritura RDMA del servidor para cada segmento leído. Los datos leídos del disco son contiguos. La cantidad de datos leidos esta limitada por el numero máximo de paginas de destino que se pueden describir en una sola solicitud. Este limite es MaxRecvBúfersize del servidor dividido por sizeof(LWIO_RDMA_REGION). La estructura de LWIO_RDMA_REGION se muestra mas abajo.

[0110] La FIGS. 16A y 16C nos muestra, en una implementación del invento, una representación ilustrativa de el mensaje de lectura vectorizada enviado por un cliente, con la FIG 16A en el caso de no-consulta y con la FIG 16C en el caso de consulta. El mensaje de lectura 1401 tiene un encabezado de columna 1603 seguido por uno o más segmentos LWIO_RDMA_REGION 1605, 1607. En el encabezado 1603, Length 1609 se fija en el numero de bites de datos que se van a leer del archivo. Opcode 1611se fija en LWIO_OPCODE_VEC_READ (0x2). Offset 1621se fija en la localidad del bit donde se va a comenzar la lectura de los datos del archivo. Flags 1613, 1631 se fijan en 0x0 en el caso de consulta 1631, o en LWIO_HDR_FLAG_INTERRUPT (0x80) en el caso de no-consulta 1613. Marker 161 se fija en 0xFF. Credits 1615se fija en el numero de solicitudes I/O pendientes al cliente. Fid 1619se fija en File Id en el que se emite el I/O. NumPages 1623 se fija en el numero de LWIO_RDMA_REGIONs que sigue al ecabezado comun1603. PageSize 1625 se fija en tamaño de la pagina local en bites.

[0111] en el caso de no-consulta, ImmodiateCookie 1627 y Cookie 1629 se fijan con valores especificados por el cliente. el servidor puede en este caso completar la solicitud de lectura vectorizada con un LWIO_MSG_STATUS_RESPONSE a través de un envió normal, o con un RDMA con datos inmediatos si la lectura es exitosa. Los datos inmediatos de la escritura RDMA se fijan de acuerdo al valor de ImmediateCookie 1627 de la solicitud de lectura. En el caso de consulta, losVa 1635 se fijan en la localidad en que la respuesta del estatus del servidor (LWIO_IO_STATUS_BLOCK) se RDMA, y losMh 1633 se fijan en la manija asociada de la memoria.

[0112] Al encabezado común 1603 le sigue inmediatamente un numero suficiente de segmentos LWIO_RDMA_REGION 1605, 1607 para cubrir el largo de la solicitud. Todos los segmentos intermedios deben tener el tamaño de una pagina. El segmento final, en una implementación del invento, puede ser mas pequeño que una pagina, pero debe ser un múltiplo del sector final del disco. El LWIO_RDMA_REGION tiene el siguiente formato:

```
typedef volatile struct {
PVOID64    DataVa;
UINT32     DataMh;
UINT32     Length;
} LWIO_RDMA_REGION;
```

La primera LWIO_RDMA_REGION corresponde al primer PageSize bites leidos, la segunda LWIO_RDMA_REGION corresponde a los segundos PageSize leidos, y asi. DataVa 1637 se fija en la localidad que marca el comienzo de la pagina en donde los datos leídos se van a colocar. DataMh 1639 se fija en la manija de la memoria de DataVa 1637. Length 1641 se fija en PageSize 1625 para todas las regiones excepto la región final, para la cual Length puede ser mas pequeño pero tiene que ser un múltiplo de el sector final del disco.

[0113] En el caso de no-consulta, el servidor primero RDMA los datos que se van a leer. El servidor luego puede responder con un LWIO MSG_STATUS_RESPONSE, o puede enviar datos inmediatos de lectura RDMA, en tal caso los datos inmediatos se fijan en el valor de ImmediateCookiede de la solicitud de lectura. La FIG. 16B nos muestra, en una implementación del invento, una representación ilustrativa de la LWIO_MSG STATUS_RESPONSE 1643 devuelta por el servidor en el caso de no-consulta. Information 1647 se fija en el numero de bites leidos. Status 1649 se fija en 0, que indica éxito, o en otro NTSTATUS, que indica fracaso. Cookie 1645 se fija en el valor de Cookie fijado por el cliente en el encabezado del mensaje de lectura vectorizada.

[0114] En el caso de consulta, primero el servidor RDMA los datos que se van a leer, y luego el servidor RDMA un LWIO_IO_STATUS_BLOCK. La FIG. 16D nos muestra una representación ilustrativa

de un LWIO_IO_STATUS_BLOCK 1651 devuelto por el servidor en una implementación del presente invento. Information 1653se fija en el numero de bites leídos. Status 1655sc fija en 0, que indica éxito, y otro NTSTATUS, que indica fracaso.

Escritura vectorizada

[0115] el mensaje de escritura vectorizada se usa para reunir la escritura en una sola operación de envió. La cantidad de datos escritos esta limitada por el MaxReovBúfersize del servidor. si el cliente envía mas datos que estos la conexión termina.

[0116] La FIGS. 17A, 17B y 17C nos muestra, en una implementación del invento, una representación ilustrativa de un mensaje de escritura vectorizada enviado por un cliente, con la FIG 17Amostrando el caso no consulta y no-colapsado, FIG 17B mostrando el caso no consulta, caso colapsado y la FIG17C muestra el caso consulta y colapsado.

[0117] el mensaje de escritura 1701 incluye un encabezado común 1703 seguido inmediatamente por los datos que se van a escribir 1705. En el encabezado común 1703, Length 1707 se fija en el numero de datos que se van a escribir. Opcode 1709sc fija en LWIO_OPCODE_WRITE (0x3). Offset 1719sc fija en la localidad del bit donde comienza la escritura de los datos de archivo. Marker 1715 se fija en 0xFF. Credits 1713 se fija en el numero pendiente de solicitudes I/O al cliente. Fid 1717 se fija en el File Id en el cual se emite el I/O.

[0118] Flags 1711, 1721, 1727se fija en 0x0 que significa que esta en consulta 1727,de lo contrario LWIO_HDR FLAG_INTERRUPT (0x80) 1711. En el ultimo caso, las banderas pueden incluir un LWIO_HDR FLAG_COLLAPSE 1721 para indicar que todas las paginas en la escritura contienen los mismos datos, por lo que solo una pagina de datos ha sido enviada. Esta es una optimización que tiene como propósito minimizar la transferencia de datos redundantes. LWIO_HDR_FLAG_COLLAPSE solo se puede utilizar si la bandera del archivo registrado incluye un FILE NO_INTERMEDIATE_BÚFERING (0x8) y el PageSizes intercambiado durante la etapa de autenticación son múltiplos pares de cada uno. En el caso de un I/O, colapsado NumPages 1723 se fija en el numero de paginas de datos abarcados por este I/O. La última pagina puede ser parcial debido al parámetro Length. PageSize 1725se fija en el tamaño de la pagina local en bites. En el caso consulta, IosVa 1731se fijan en la localidad en donde la respuesta del estatus del servidor (LWIO_IO_STATUS_BLOCK)se ha RDMA. RDMAcd. IosMh 1729 se fija en la manija de la memoria asociada.

[0119] en el caso no-consulta, para ambos casos, colapsado y no colapsado I/O, el servidor responde al mensaje de escritura con un LWIO_MSG_STATUS_RESPONSE.

[0120] La FIG . 17D nos muestra, en una implementación del invento, una representación ilustrativa del LWIO_MSG_STATUS_RESPONSE 1733 devuelto por el servidor. Information 1737 s fija en el

numero de bites escritos. Status 1739 se fija en 0, indicando éxito, o en otro NTSTATUS, indicando fracaso. Cookie 1735 se fija en el valor de Cookie value determinado por el cliente en el encabezado del mensaje de escritura.

[0121] en el caso de consulta, para ambos casos, I/O colapsado y no colapsado, el servidor RDMAs un LWIO_IO_STATUS_BLOCK. La *FIG. 17E* nos muestra una representación ilustrativa de un LWIO_IO_STATUS_BLOCK 1741devuelto por el servidor en una implementación del invento. Information 1 743 se fija en el numero de bites escritos. Status 1745 se fija en 0, indicando éxito, o en otro NTSTATUS, indicando fracaso.

Conclusión

[0122] Aunque se han descrito y mostrado implementaciones ilustrativas del presente invento, se puede apreciar que varios cambios se pueden hacer sin apartarse del invento. En forma similar, se pueden variar en forma intercambiable los pasos en el proceso que se ha descrito logrando el mismo resultado. Adicionalmente, los ejemplos descritos anteriormente no tienen la intención de limitar el alcance del invento. Por el contrario, la intención es la de cubrir todas las modificaciones, construcciones alternativas, y equivalentes, que estén dentro del espíritu y alcance del invento.

LO QUE SE REIVINDICA ES:

1. Un sistema para descargar las tareas de entrada/salida I/O de un primer computador a un segundo computador, lo que comprende: un cliente que opera en el primer computador; un servidor que opera en el segundo computador; y por lo menos un canal RDMA que vincula al primer computador con el segundo, donde el primer computador y el segundo computador se comunican de acuerdo a un protocolo que consta de una fase de descubrimiento en red y una fase de procesamiento I/O.

2. El sistema que se reivindica en 1 donde la fase de procesamiento I/O, y las operaciones de lectura se implementan utilizando RDMA y las operaciones de escritura se implementan utilizando operaciones de envió.

3. El sistema que se reivindica en 1 donde el protocolo se utiliza en asociación con un segundo protocolo de red.

4. El sistema que se reivindica en 3 donde el segundo protocolo es SMB.

5. El sistema que se reivindica en 3 donde el segundo protocolo es CIFS.

6. Un medio legible por computador que guarda instrucciones ejecutables por computador y datos legibles por computador que comprenden un producto de programación de computador para utilizar en un sistema para descargar las tareas de entrada/salida I/O de un primer computador a un segundo computador, sistema que consta de: por lo menos un canal RDMA que vincula al primer computador con el segundo

36

computador, donde el primer computador y el segundo computador se comunican de acuerdo a un protocolo que consta de una fase de descubrimiento en red y una fase de procesamiento I/O.

7. Un método para descargar una tarea de entrada/salida I/O de un primer computador a un segundo computador, que consta de: descubrir, por parte de un cliente en un primer computador y un servidor en un segundo computador, uno o mas proveedores capaces de RDMA que se comparten, y un anuncio, por parte del cliente, de una solicitud de procesamiento I/O para que el servidor en el segundo computador complete.

8. El método de la reivindicación 7 donde descubrir uno o mas proveedores que se comparten y son capaces de RDMA incluye además:

que un cliente obtenga una solicitud del servidor para una llave de continuación;

abrir, por parte del cliente, un canal al servidor;

enviar, por parte del cliente y a través del canal una solicitud de negociación; y

enviar, a través del canal, una respuesta de negociación que incluye una lista minima de proveedores comunes.

9. El método de la reivindicación 7, que incluye además:

la creación, por parte del cliente, de una conexión RDMA con el servidor a través de un proveedor capas de RDMA, y

la autenticación por parte del cliente y del servidor, de la conexión RDMA.

10. El método de la reivindicación 9, que incluye además:

el registro por parte del cliente de uno o mas archivos para usar con el servidor a través de la conexión RDMA.

11. El método de la reivindicación 10, donde registrar uno o mas archivos consta de:

el envió, por parte del cliente al servidor, de un mensaje de registro de archivo; y

el envió, por parte del servidor al cliente, de un mensaje de terminación de registro de archivo.

12. El método de la reivindicación 9, donde la autenticación de la conexión RDMA incluye además:

el envió, por parte del cliente, de un mensaje de solicitud autenticado que incluye una llave;

si la llave corresponde a una previamente enviada por el servidor al cliente, el envió, por parte del servidor, de un mensaje autenticado de respuesta al cliente.

13. El método de la reivindicación 12, donde la llave anterior es una llave contenida en el mensaje de respuesta negociado y enviado por el servidor al cliente.

14. El método de la reivindicación 12, que incluye además:

36

El envió por parte del servidor al cliente, de un mensaje de respuesta del estatus para completar la autenticación.

15. El método de la reivindicación 7, donde el anuncio de la solicitud de procesamiento I/O incluye el envió, por parte del cliente de: (a) una solicitud de cerrar, (b) una solicitud de cancelación, (c) una solicitud de lectura, (d) una solicitud de escritura, (e) una solicitud de lectura vectorizada, y (f) una solicitud de escritura vectorizada

16. El método de la reivindicación 15, que incluye además:

completar, por parte del servidor, la solicitud de lectura y de lectura vectorizada enviando datos utilizando operaciones de escritura RDMA, y

completando por parte del servidor, la solicitud de escritura y escritura vectorizada enviando datos utilizando operaciones de envió normales.

17. El método de la reivindicación 15, donde la solicitud de escritura vectorizada incluye una bandera de colapso en el encabezado de la solicitud.

18. El método de la reivindicación 7, donde el anuncio de la solicitud de procesamiento I/O incluye además indicar si el completar por parte del servidor debe ser en modalidad de consulta.

19. El método de la reivindicación 18, donde la indicación sobre si el completar debe ser en modalidad de consulta comprende el indicar que el completar no debe ser en modalidad de consulta fijando una bandera de interrupción en el encabezado de la solicitud de procesamiento I/O.

20. El método de la reivindicación 18, que incluye además:

Si el cliente indica que el completar no se debe hacer en modalidad de consulta, completando, por parte del servidor, la solicitud de procesamiento I/O a través del envió de un bloque de estatus al primer computador a través de una transferencia RDMA.

21. El método de la reivindicación 18, que incluye además:

si el cliente indica que el completar debe ser en modalidad de consulta, y el cliente ha enviado un mensaje de solicitud de interrupción al servidor, enviar, del servidor al cliente, un mensaje de interrupción de respuesta a través de un envió ordinario.

22. El método de la reivindicación 7, donde anunciar la solicitud de procesamiento I/O incluye además especificar el numero de créditos en el encabezado de la solicitud.

23. Medios legibles por computador almacenando instrucciones ejecutables por computador para implementar un método para descargar una tarea de entrada/salida I/O de un primer computador a un segundo computador, método que consta de:

el descubrimiento, por parte de un cliente en el primer computador y un servidor en el segundo computador, de uno o mas proveedores compartidos y capaces de RDMA; y

el anuncio, por parte del cliente, de una solicitud para procesamiento I/O para ser completada por el servidor en el segundo computador.

24. un método para manejar búfers en un protocolo de descarga de entrada/salida que incluye: el envió, por parte de un servidor a un cliente, de un mensaje de crédito delta que incluye un campo de información fijado con el numero de créditos, donde, si el numero es negativo –N, el servidor requiere que el cliente retire N créditos; si el numero de créditos es negativo –N, enviar, por parte del cliente al servidor, N mensajes de crédito, o por el contrario el envió, por parte del cliente al servidor, de un mensaje de crédito; y por cada mensaje de crédito enviado por el cliente, enviar, por parte del servidor al cliente un mensaje de estatus de respuesta.

COMPENDIO

Un método y sistema para descargar procesamientos de I/O de un primer computador a un segundo computador, utilizando interconexiones en red capaces de RDMA, son expuestas. El método y sistema incluye a un cliente en el primer computador comunicándose a través de una conexión RDMA con un servidor en el segundo computador a través de un protocolo ligero de entrada/salida I/O (LWIO). El protocolo generalmente consta de una fase de descubrimiento seguida de un fase de procesamiento I/O. Durante la fase de descubrimiento, el cliente y el servidor determinan una lista minima de proveedores capaces de RDMA que comparten. Durante la fase de procesamiento I/O, el cliente anuncia una petición de I/O para descargar del segundo computador a través de un canal RDMA mutuamente autenticado. El modelo I/O es asimétrico con operaciones de lectura implementadas usando RDMA y las operaciones de escritura implementadas utilizando envios corrientes. La solicitudes de lectura y escritura se pueden completar en modalidad de consulta y modalidad de interrupciones. Los búfers se manejan a través de un mecanismo de crédito.

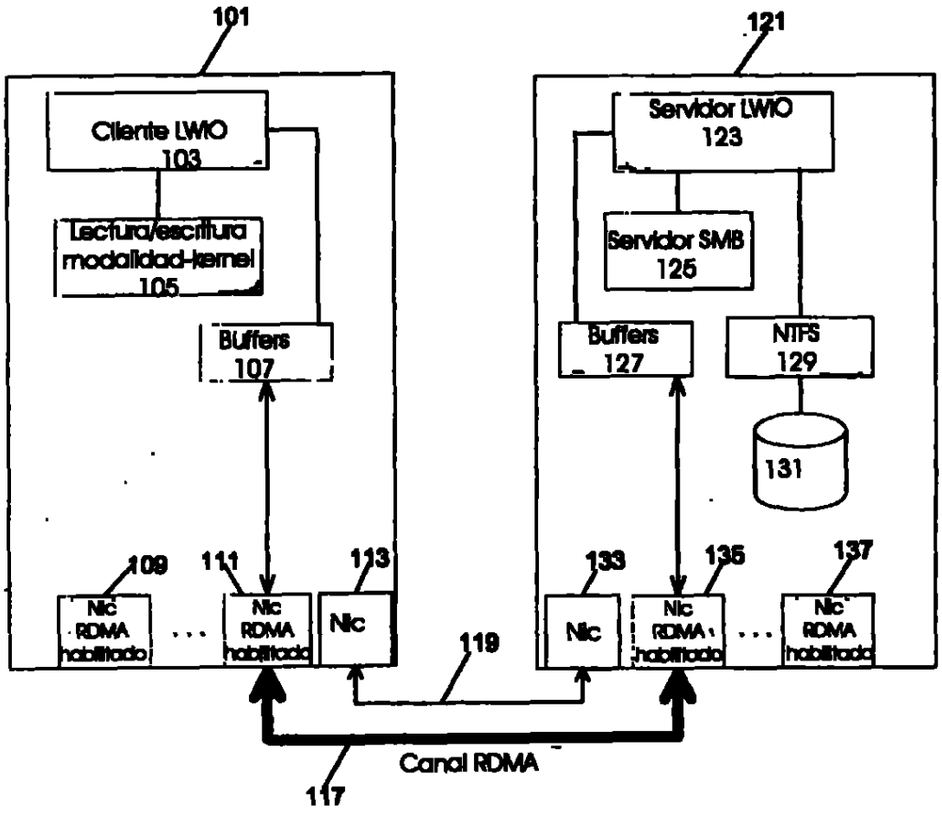

101
121
Cliente LWIO
103
Lectura/escritura
modalidad-kernel
105
Buffers
107
Servidor LWIO
123
Servidor SMB
125
Buffers
127
NTFS
129
131
109
111
113
133
135
137
Nic
RDMA
habilitada
Nic
Nic
Nic
RDMA
habilitada
Nic
RDMA
habilitada
119
Canal RDMA
117

FIG. 1

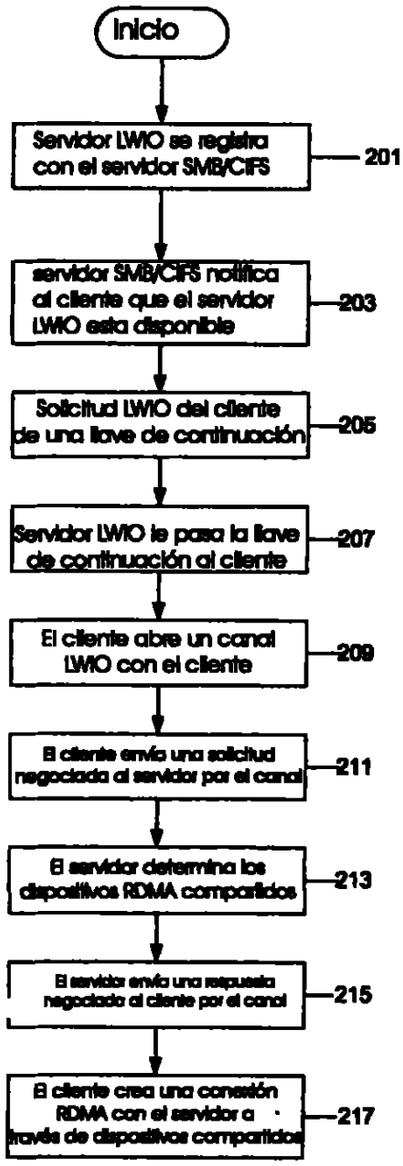
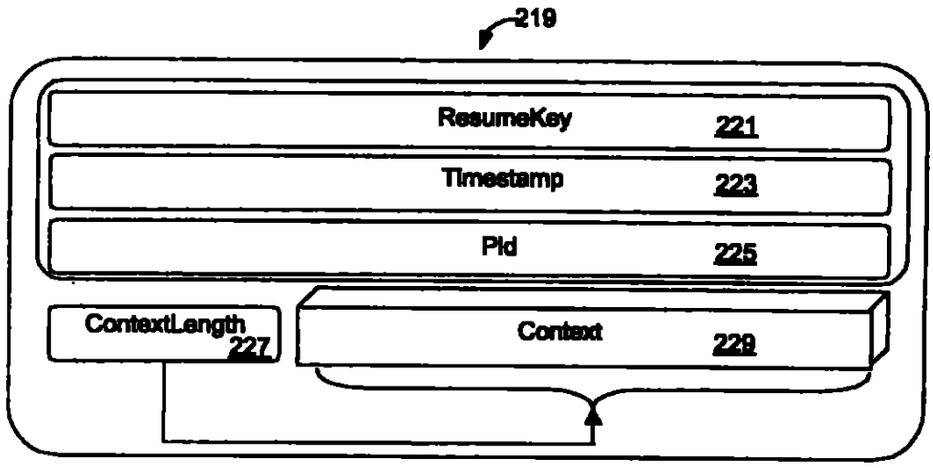

Inicio
Servidor LWIO se registra con el servidor SMB/CIFS
201
servidor SMB/CIFS notifica al cliente que el servidor LWIO esta disponible
203
Solicitud LWIO del cliente de una llave de continuación
205
Servidor LWIO le pasa la llave de continuación al cliente
207
El cliente abre un canal LWIO con el cliente
209
El cliente envía una solicitud negociada al servidor por el canal
211
El servidor determina los dispositivos RDMA compartidos
213
El servidor envía una respuesta negociada al cliente por el canal
215
El cliente crea una conexión RDMA con el servidor a través de dispositivos compartidos
217

FIG. 2



Solicitud del servidor para una llave de continuación

FIG. 3

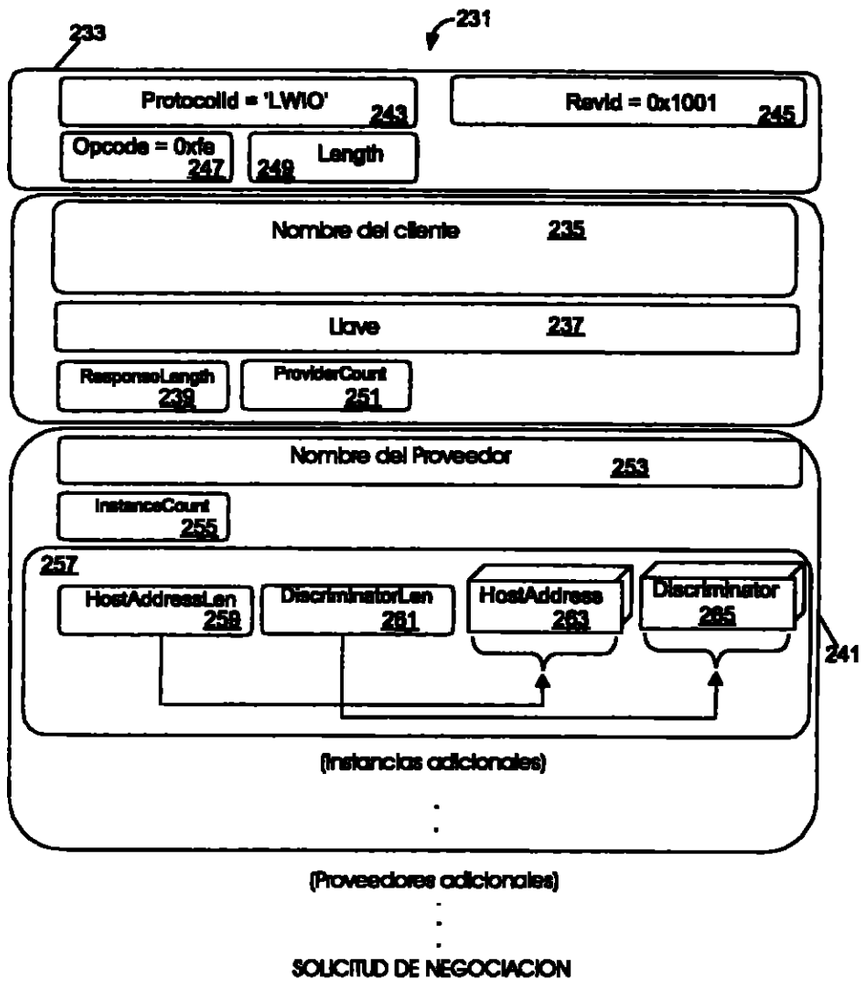

233
231
ProtocolId = 'LWIO' 243
RevId = 0x1001 245
Opcode = 0xfe 247
Length 249
Nombre del cliente 235
Llave 237
ResponseLength 239
ProviderCount 251
Nombre del Proveedor 253
InstanceCount 255
257
HostAddressLen 259
DiscriminatorLen 261
HostAddress 263
Discriminator 265
241
(Instancias adicionales)
(Proveedores adicionales)
SOLICITUD DE NEGOCIACION

FIG. 4A

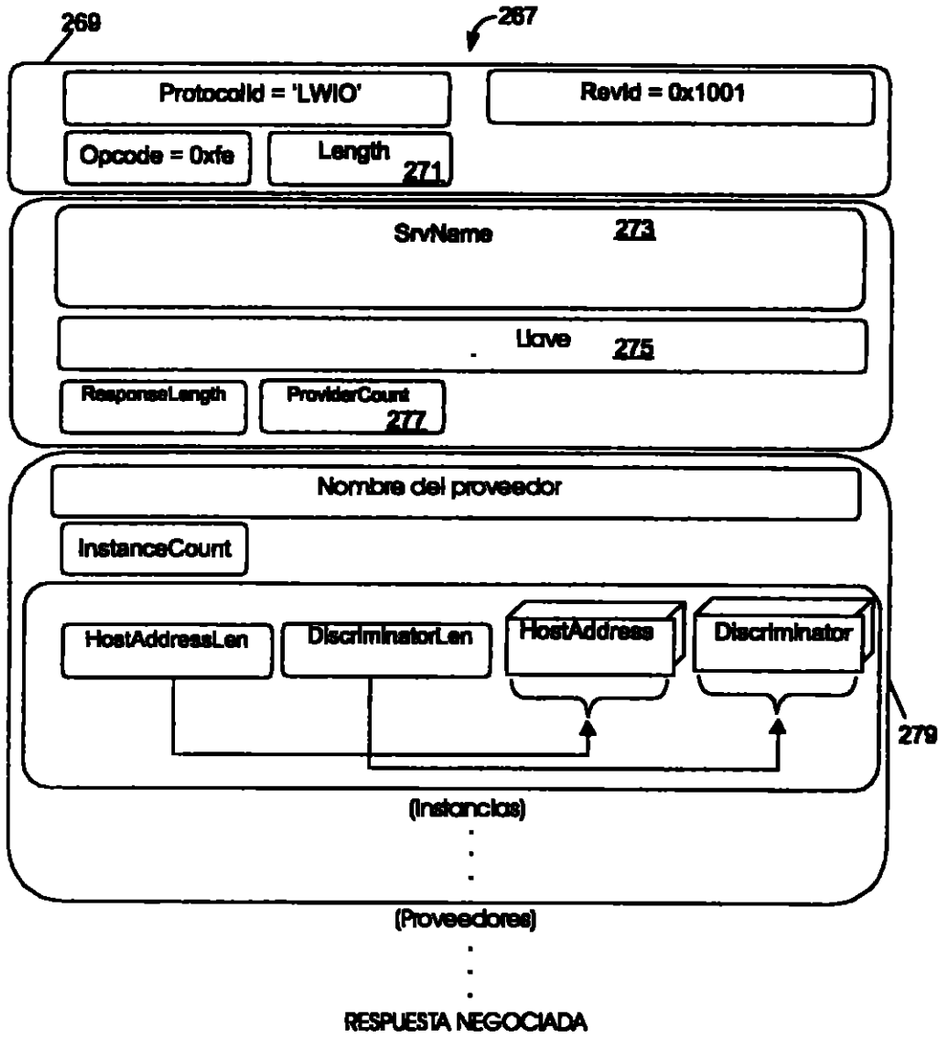

269
267
ProtocolId = 'LWIO'
RevId = 0x1001
Opcode = 0xfe
Length
271
SrvName
273
Llave
275
ResponseLength
ProviderCount
277
Nombre del proveedor
InstanceCount
HostAddressLen
DiscriminatorLen
HostAddress
Discriminator
279
(Instancias)
(Proveedores)
RESPUESTA NEGOCIADA

FIG. 4B

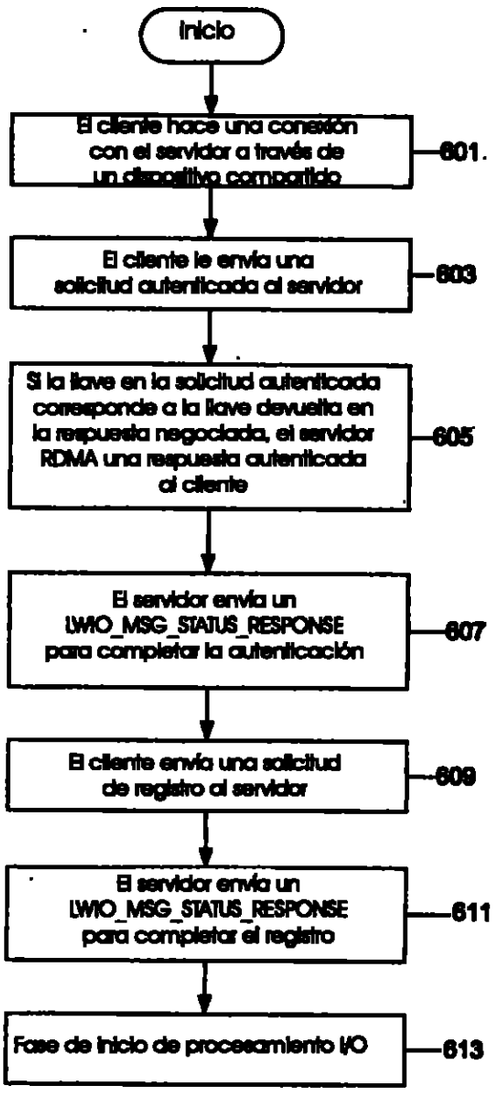

Inicio
El cliente hace una conexión con el servidor a través de un dispositivo compartido
601
El cliente le envía una solicitud autenticada al servidor
603
Si la llave en la solicitud autenticada corresponde a la llave devuelta en la respuesta negociada, el servidor RDMA una respuesta autenticada al cliente
605
El servidor envía un LWIO_MSG_STATUS_RESPONSE para completar la autenticación
607
El cliente envía una solicitud de registro al servidor
609
El servidor envía un LWIO_MSG_STATUS_RESPONSE para completar el registro
611
Fase de inicio de procesamiento I/O
613

FIG. 5

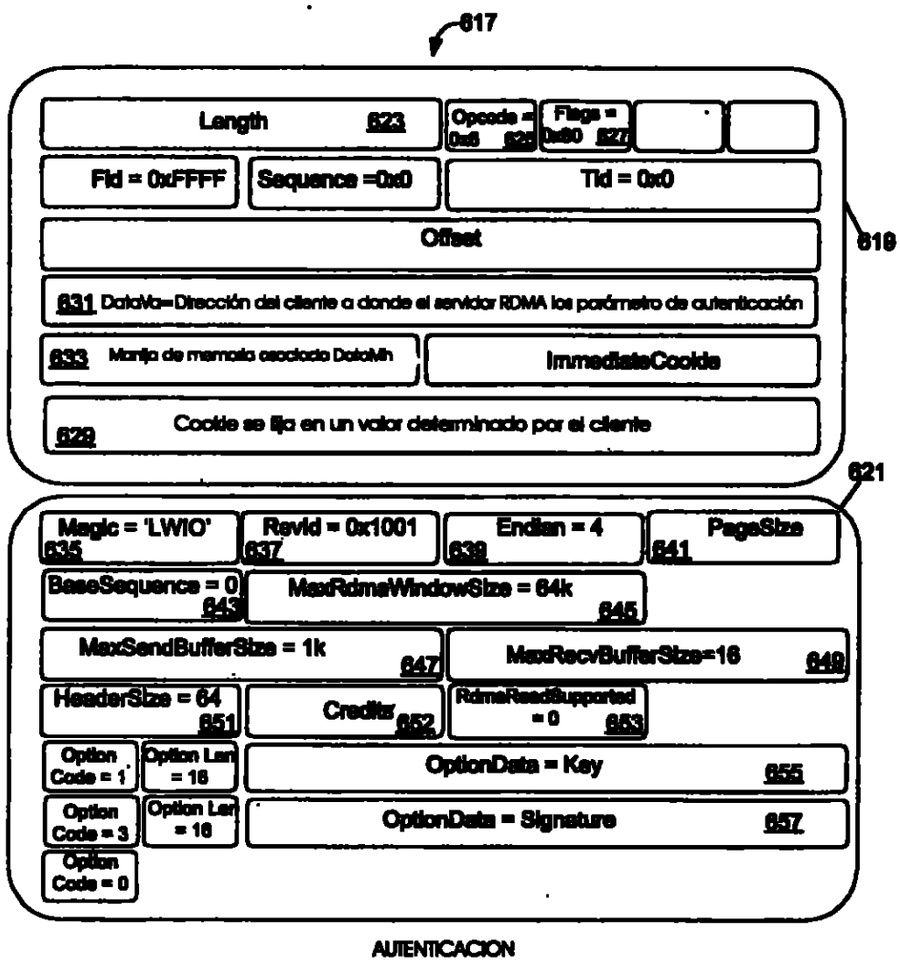
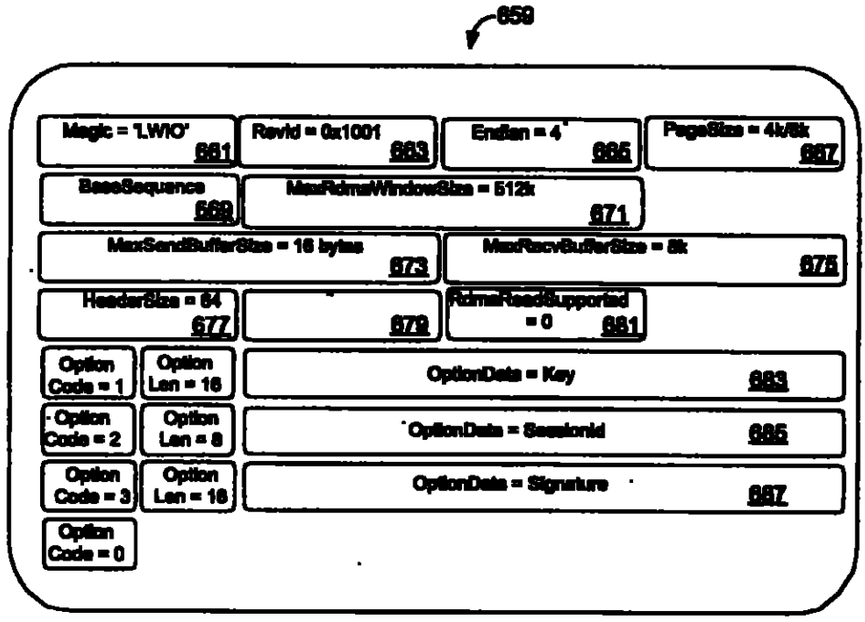
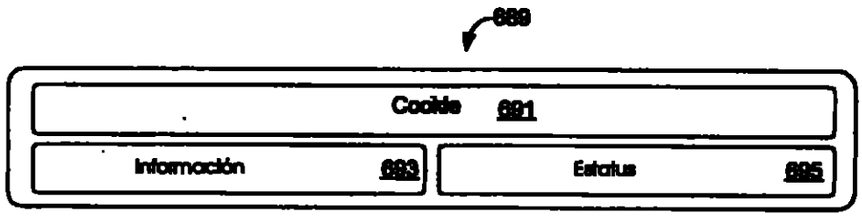

617
Length 623
Opcode = 0x6 625
Flags = 0x80 627
Fld = 0xFFFF
Sequence =0x0
Tid = 0x0
Offset
619
631 DataVa=Dirección del cliente a donde el servidor RDMA los parámetro de autenticación
633 Manija de memoria asociada DataMh
ImmediateCookie
629 Cookie se fija en un valor determinado por el cliente
621
Magic = 'LWIO' 635
RevId = 0x1001 637
Endian = 4 639
PageSize 641
BaseSequence = 0 643
MaxRdmaWindowSize = 64k 645
MaxSendBufferSize = 1k 647
MaxRecvBufferSize=16 649
HeaderSize = 64 651
Credits 652
RdmaReadSupported = 0 653
Option Code = 1
Option Len = 16
OptionData = Key 655
Option Code = 3
Option Len = 16
OptionData = Signature 657
Option Code = 0
AUTENTICACION

FIG. 6A



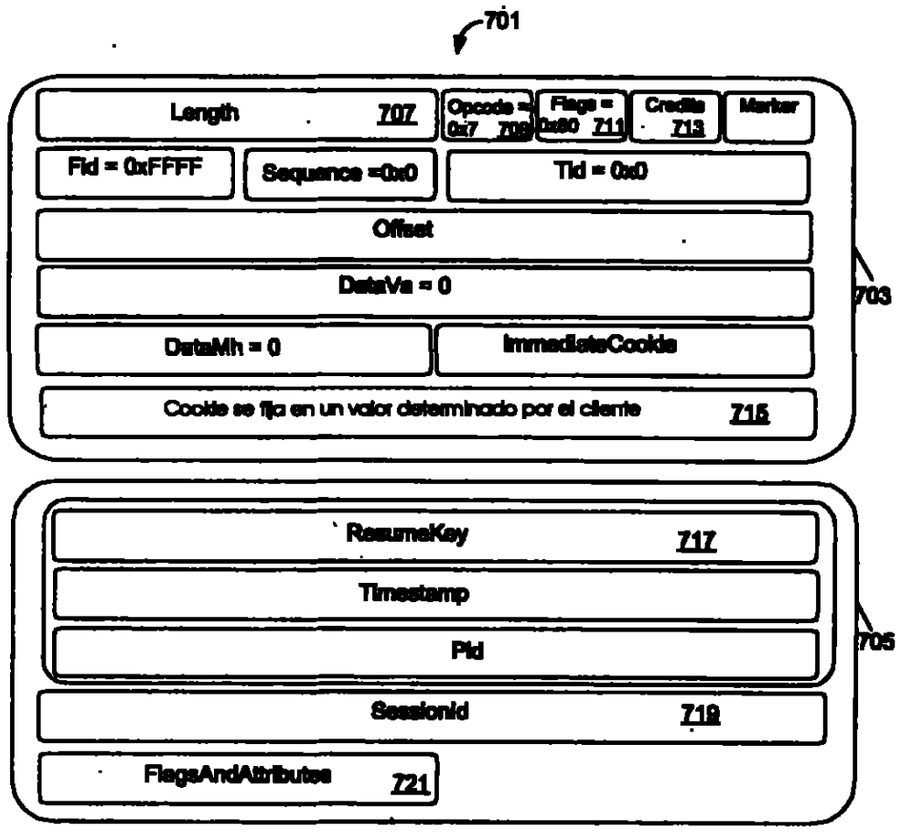
RESPUESTA AUTENTICADA

FIG. 6B



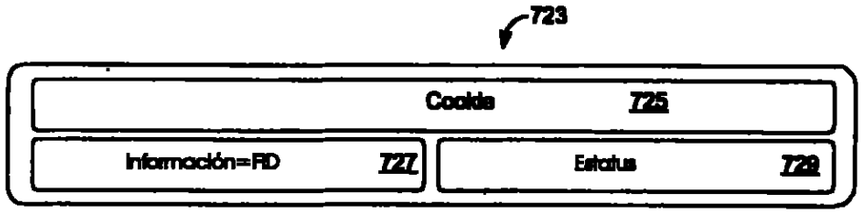
TERMINACIÓN DE LA AUTENTICACIÓN POR PARTE DEL SERVIDOR

FIG. 6C



REGISTRO DE ARCHIVOS

FIG. 7A



TERMINACIÓN DE REGISTRO DE ARCHIVOS POR PARTE DEL SERVIDOR

FIG. 7B

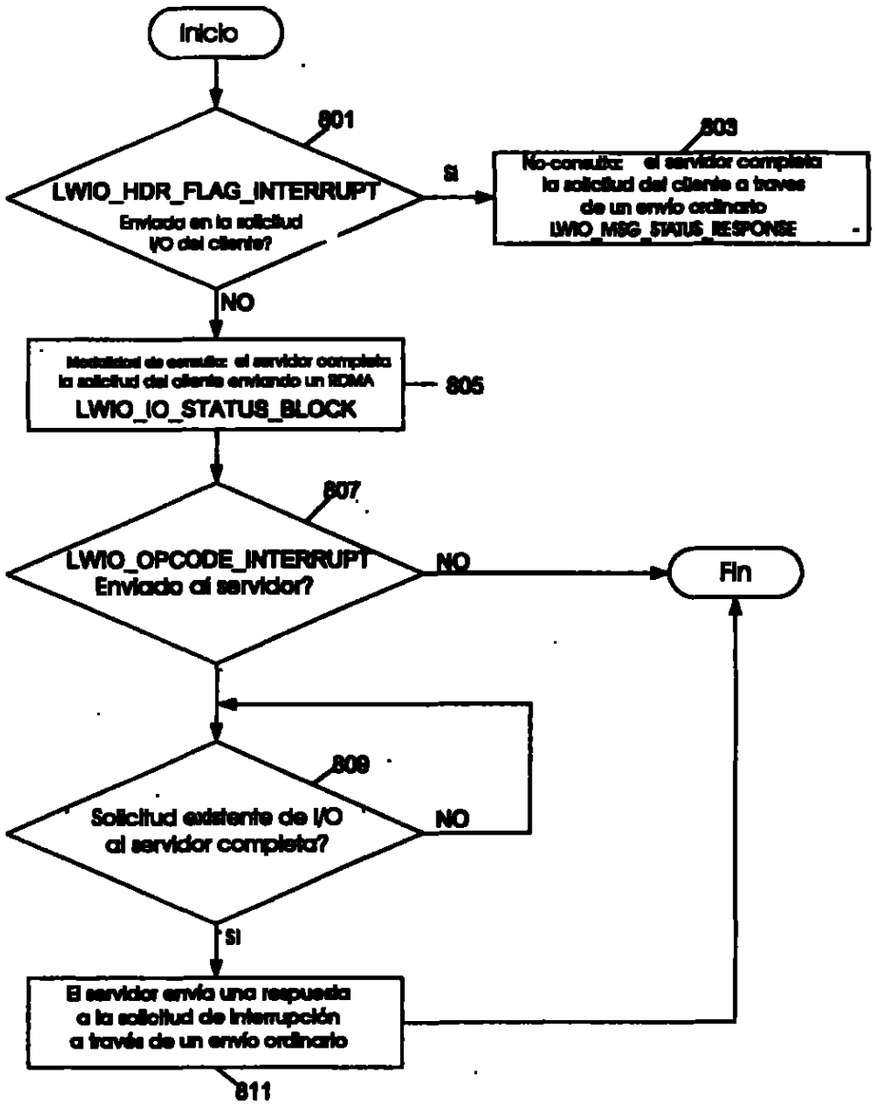

Inicio
801
LWIO_HDR_FLAG_INTERRUPT
Enviado en la solicitud
I/O del cliente?
SI
803
No-consulta: el servidor completa
la solicitud del cliente a través
de un envío ordinario
LWIO_MSG_STATUS_RESPONSE
NO
Modalidad de consulta: el servidor completa
la solicitud del cliente enviando un RDMA
LWIO_IO_STATUS_BLOCK
805
807
LWIO_OPCODE_INTERRUPT
Enviado al servidor?
NO
Fin
809
Solicitud existente de I/O
al servidor completa?
NO
SI
El servidor envía una respuesta
a la solicitud de interrupción
a través de un envío ordinario
811

FIG. 8

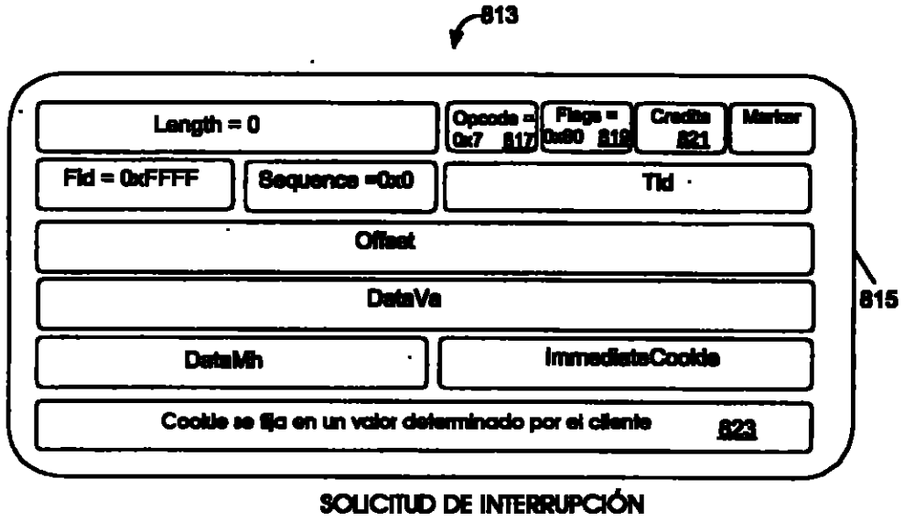

813
Length = 0
Opcode = 0x7 817
Flags = 0x80 819
Credits 821
Marker
Fid = 0xFFFF
Sequence =0x0
Tid
Offset
DataVa
DataMh
ImmediateCookie
Cookie se fija en un valor determinado por el cliente 823
815
SOLICITUD DE INTERRUPCIÓN

FIG. 9A

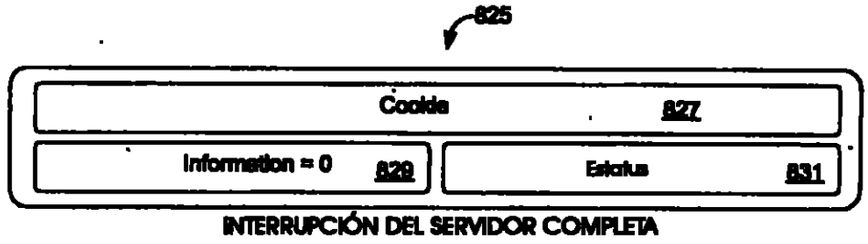

825
Cookie 827
Information = 0 829
Estatus 831
INTERRUPCIÓN DEL SERVIDOR COMPLETA

FIG. 9B

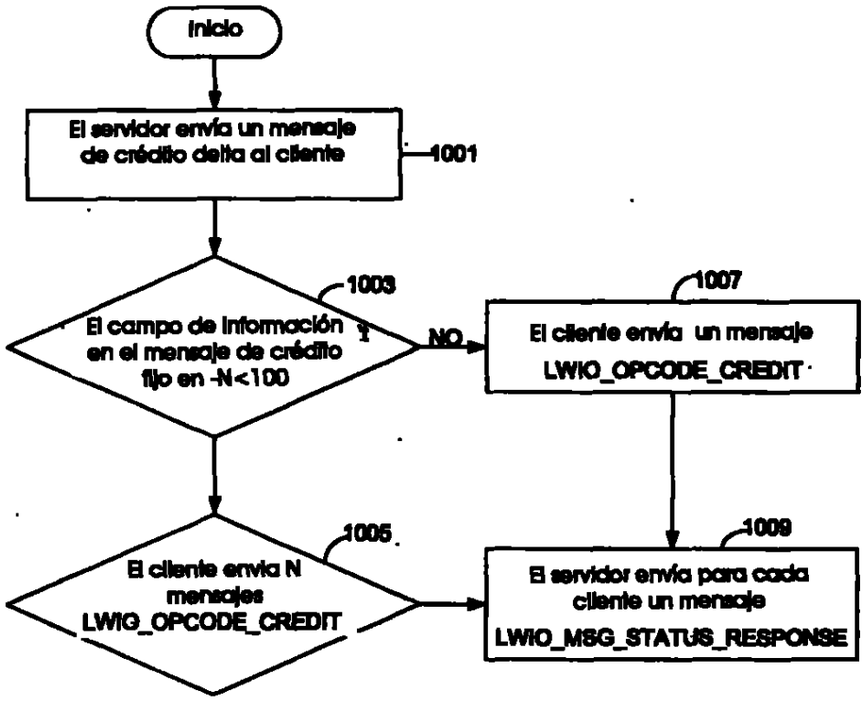
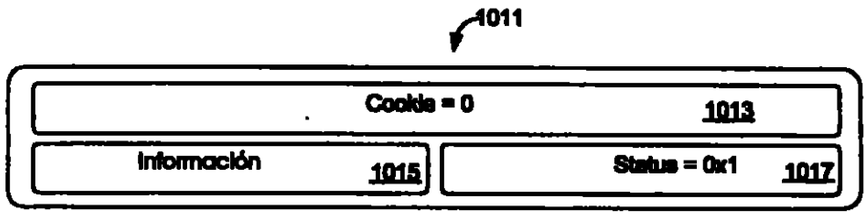
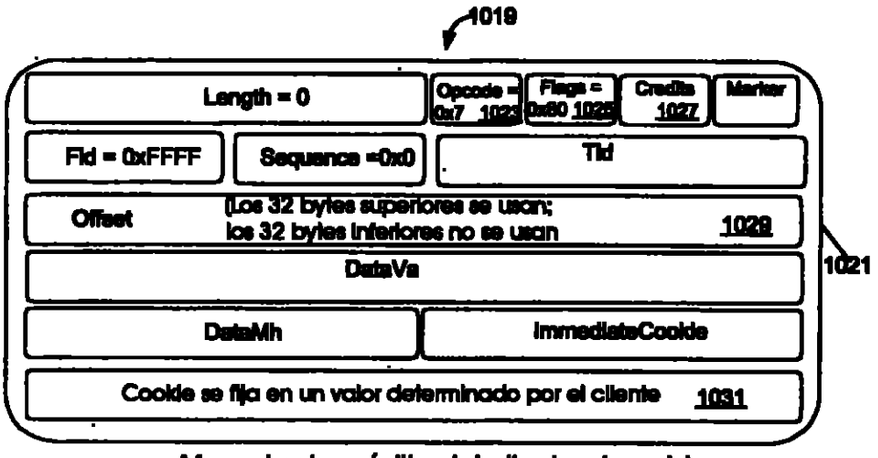

Inicio
El servidor envía un mensaje de crédito delta al cliente
1001
1003
El campo de información en el mensaje de crédito fijo en -N<100
NO
1007
El cliente envía un mensaje LWIO_OPCODE_CREDIT
1005
El cliente envía N mensajes LWIO_OPCODE_CREDIT
1009
El servidor envía para cada cliente un mensaje LWIO_MSG_STATUS_RESPONSE

FIG. 10



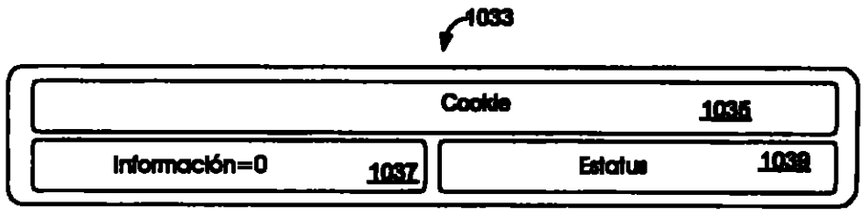
Mensaje de crédito del servidor al cliente

FIG. 11A



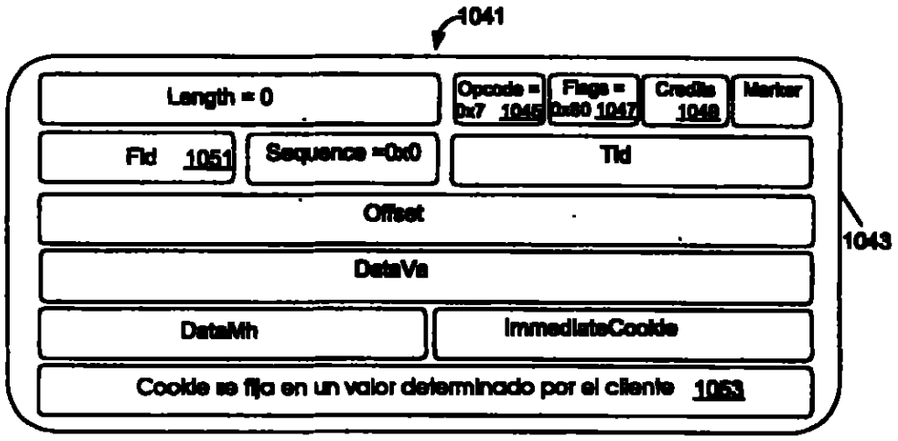
Mensaje de crédito del cliente al servidor

FIG. 11B



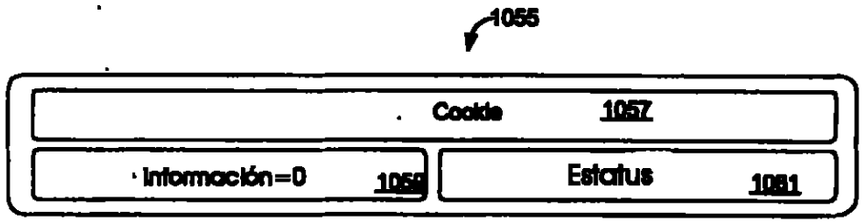
Se completa el credito del servidor

FIG. 11C



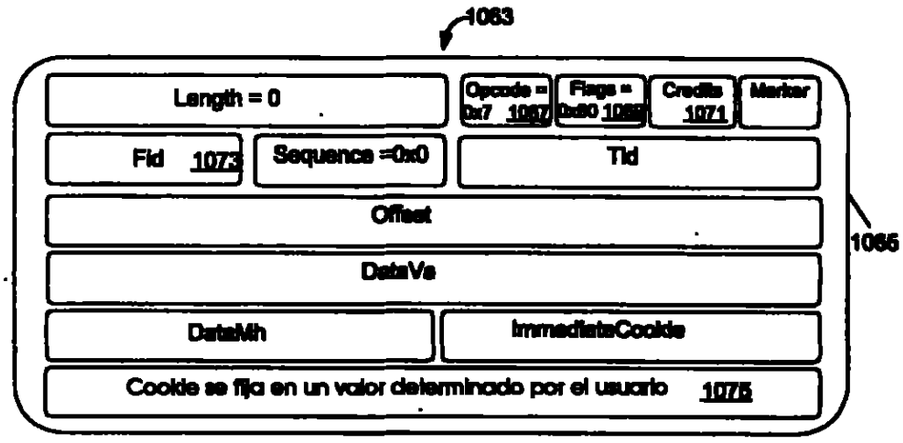
Mensaje del cliente para cerrar

FIG. 12A



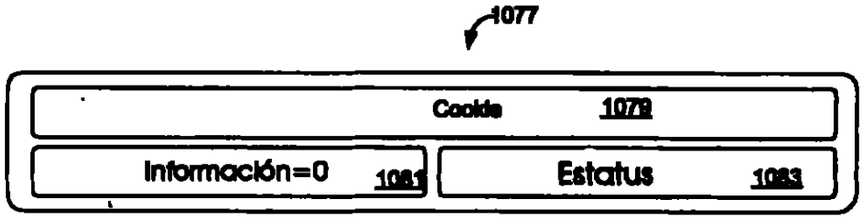
El servidor completa el cierre

FIG. 12B



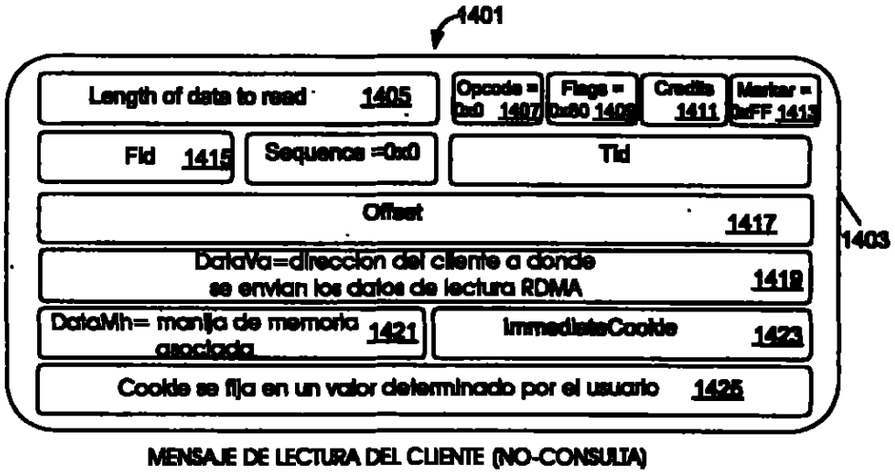
Mensaje de cancelación del cliente

FIG. 13A



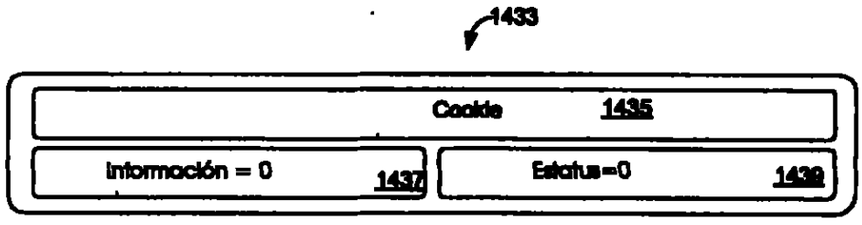
El servidor completa la cancelación

FIG. 13B



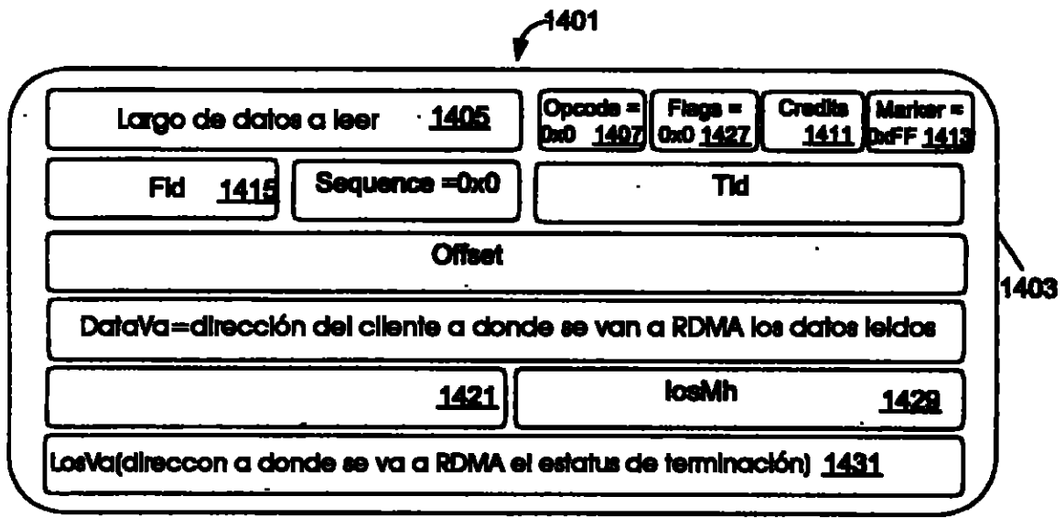
MENSAJE DE LECTURA DEL CLIENTE (NO-CONSULTA)

FIG. 14A



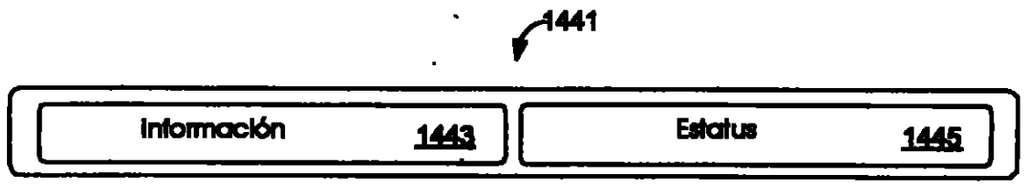
SE COMPLETA LA LECTURA (NO-CONSULTA)

FIG. 14B



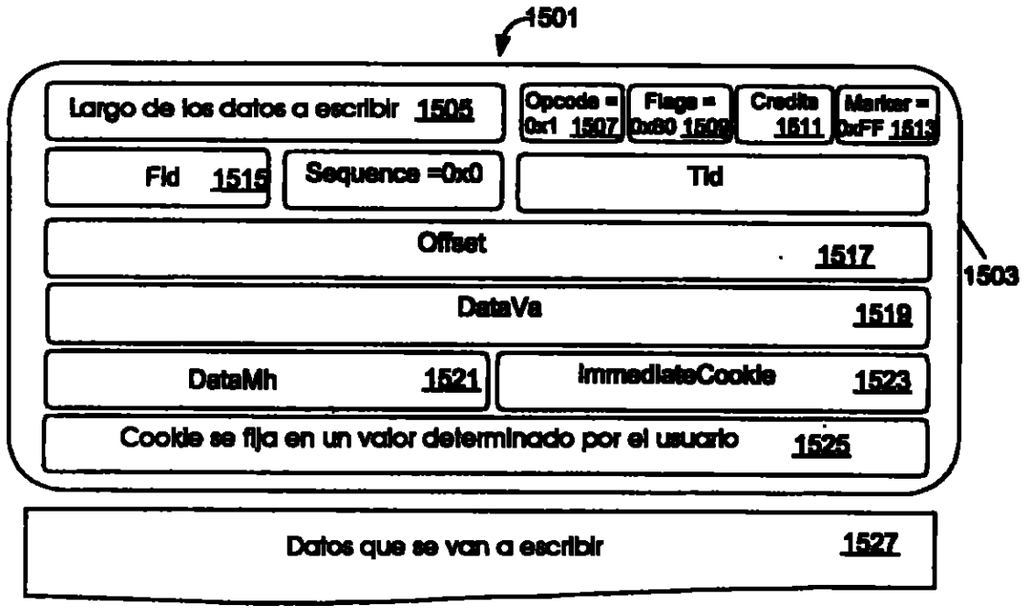
MENSAJE DE LECTURA DEL CLIENTE (CONSULTA)

FIG. 14C



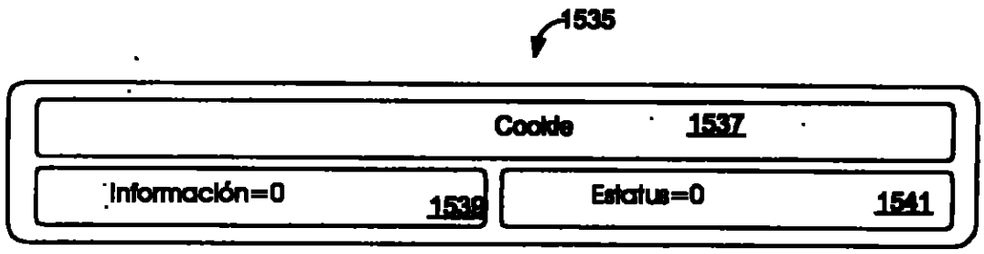
LECTURA COMPLETA (CONSULTA)

FIG. 14D



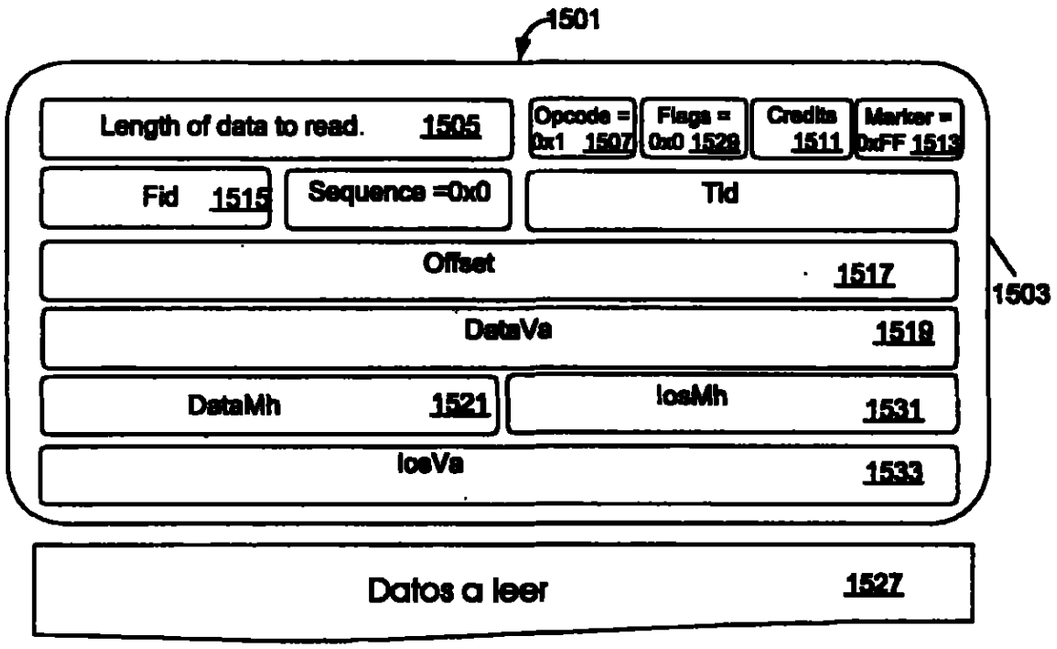
MENSAJE DE CLIENTE DE ESCRITURA (NO-CONSULTA)

FIG. 15A



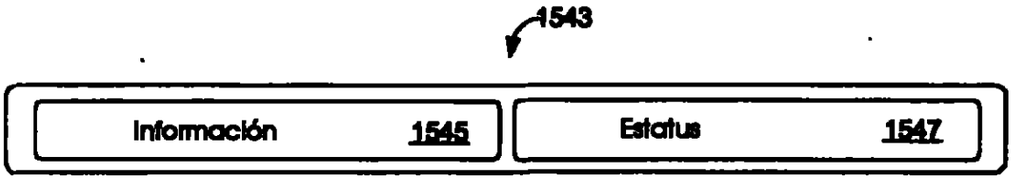
TERMINACION DE LA ESCRITURA (NO-CONSULTA)

FIG. 15B



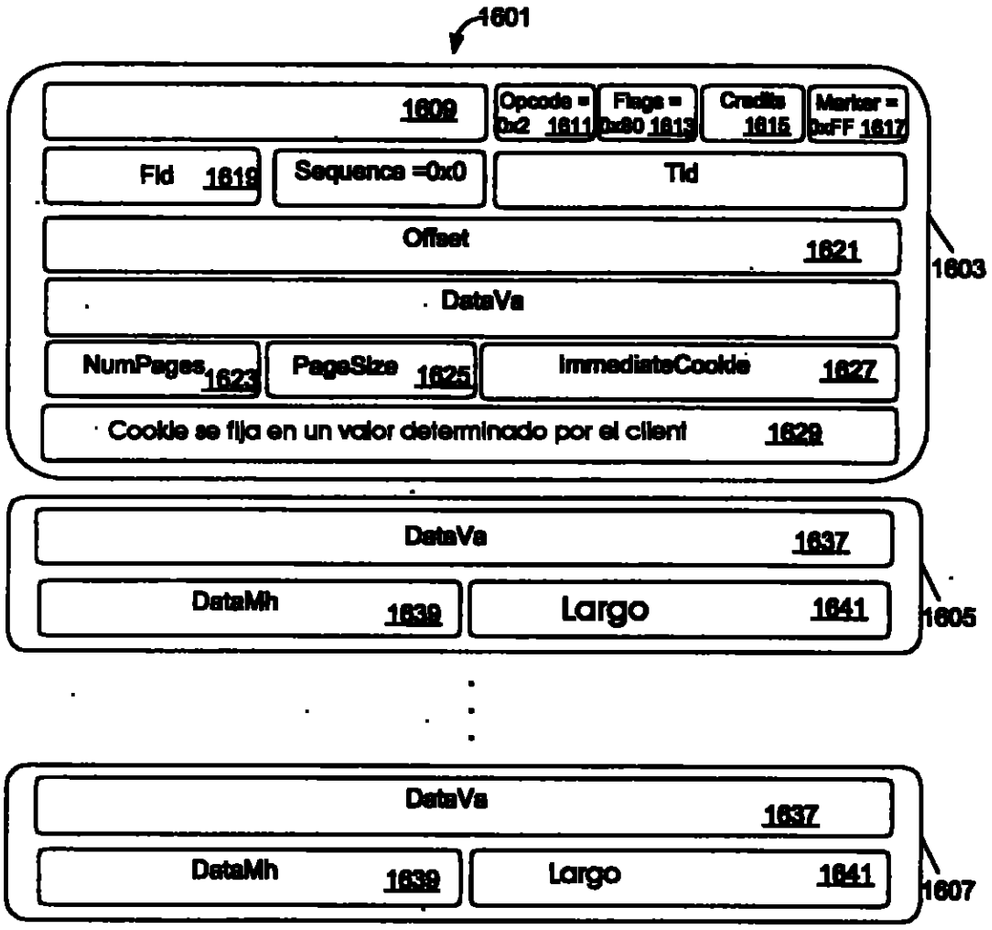
MENSAJE DE CLIENTE DE ESCRITURA (CONSULTA)

FIG. 15C



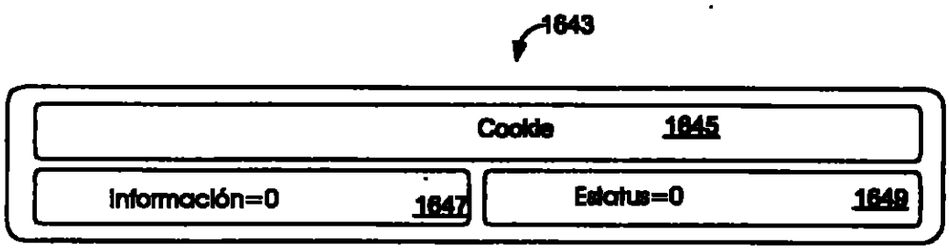
TERMINACION DE LA ESCRITURA (CONSULTA)

FIG. 15D



LECTURA VECTORIAL COMPLETA (NO-CONSULTA)

FIG. 16A



LECTURA VECTORIZADA COMPLETA (NO-CONSULTA)

FIG. 16B

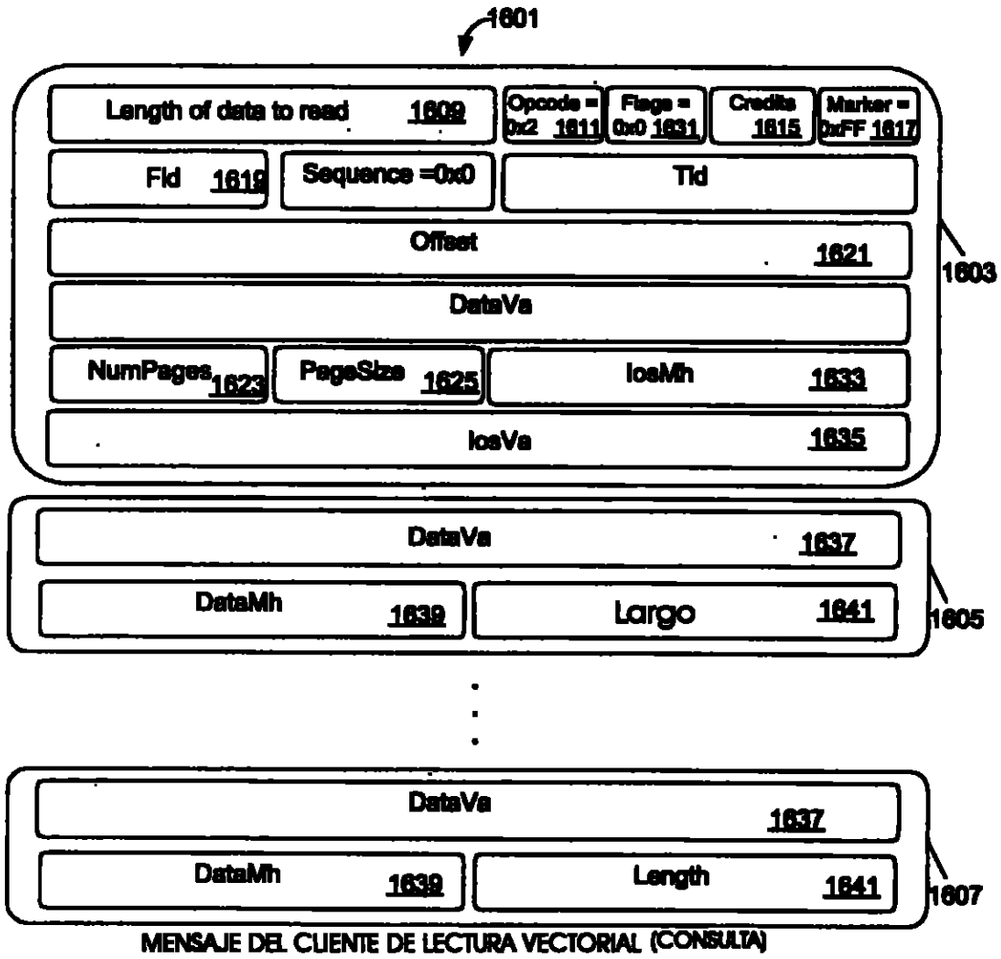


FIG. 16C

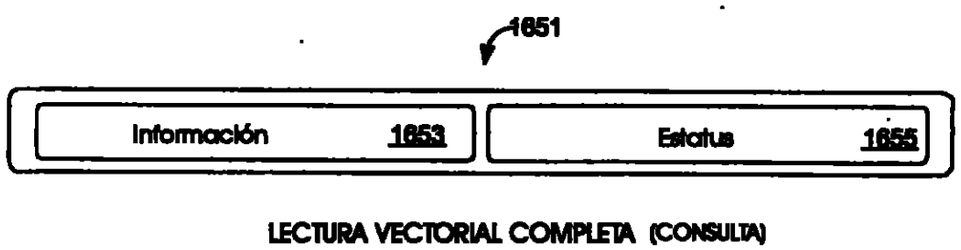
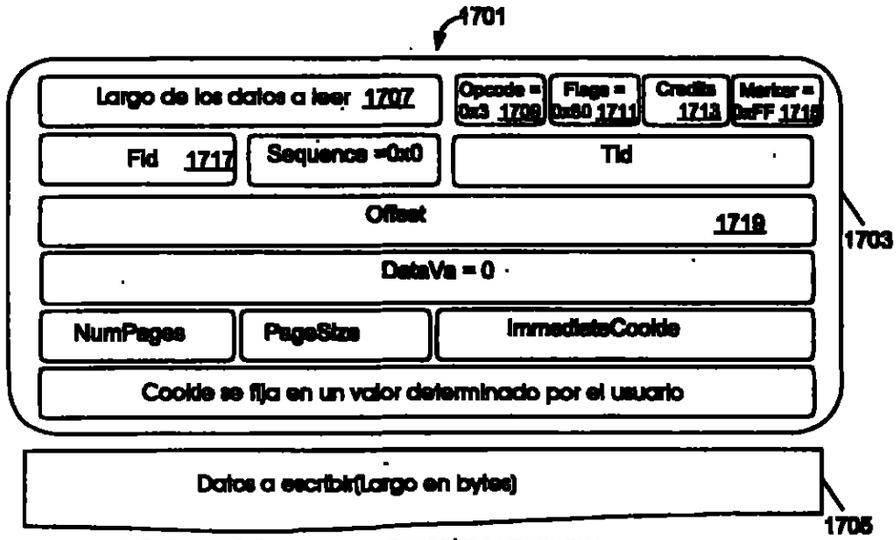
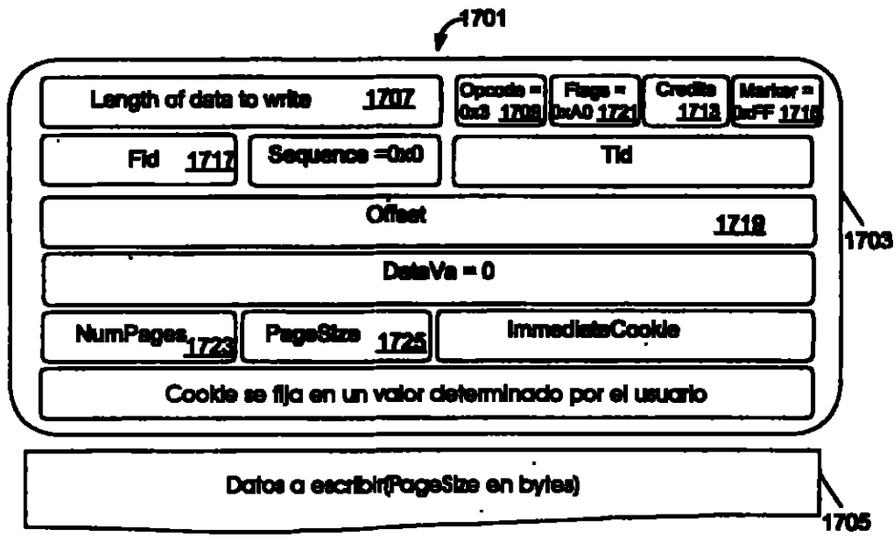


FIG. 16D



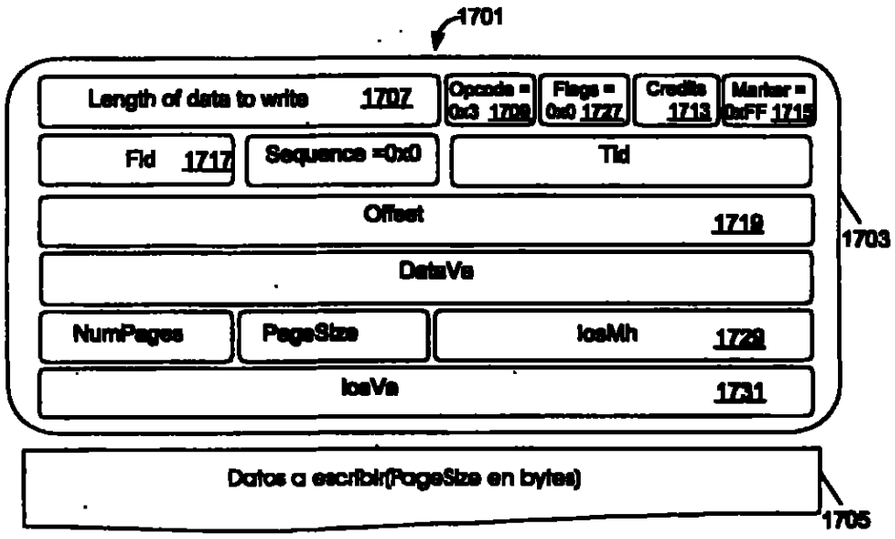
MENSAJE DE CLIENTE PARA ESCRITURA VECTORIAL, NO COLAPSADO (NO CONSULTA)

FIG. 17A



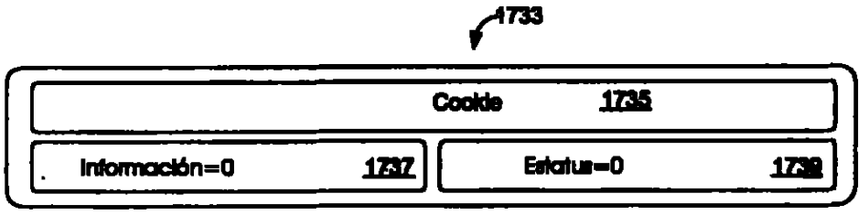
MENSAJE DE CLIENTE PARA ESCRITURA VECTORIAL, COLAPSADO (NO CONSULTA)

FIG. 17B



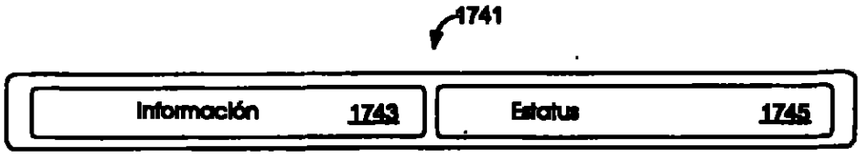
MENSAJE DE CLIENTE PARA ESCRITURA VECTORIAL, COLAPSADO (CONSULTA)

FIG. 17C



ESCRITURA VECTORIAL COMPLETA, (NO CONSULTA)

FIG. 17D



ESCRITURA VECTORIAL COMPLETA, (CONSULTA)

FIG. 17E

PCT/US2004/024026

PATENT COOPERATION TREATY

From the INTERNATIONAL BUREAU

PCT

NOTIFICATION OF RECEIPT OF RECORD COPY

(PCT Rule 24.2(a))

To:

CONKLIN, John, B.
Leydig, Voit & Mayer, LTD.
Two Prudential Plaza, Suite 4900
180 N Stetson Av
Chicago, IL 60601-6780
United States of America

Date of mailing (day/month/year)	IMPORTANT NOTIFICATION
21 September 2004 (21.09.2004)	
Applicant's or agent's file reference	**International application No.**
229789	PCT/US2004/024026

The applicant is hereby notified that the International Bureau has received the record copy of the international application as detailed below.

Name(s) of the applicant(s) and State(s) for which they are applicants:

MICROSOFT CORPORATION (for all designated States except US)
MOHAMED, Ahmed, H. et al (for US)

International filing date	:	26 July 2004 (26.07.2004)
Priority date(s) claimed	:	31 December 2003 (31.12.2003)
Date of receipt of the record copy by the International Bureau	:	13 September 2004 (13.09.2004)
List of designated Offices	:	

AP :BW,GH,GM,KE,LS,MW,MZ,NA,SD,SL,SZ,TZ,UG,ZM,ZW
EA :AM,AZ,BY,KG,KZ,MD,RU,TJ,TM
EP :AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PL,PT,RO,SE,SI,SK,
TR
OA :BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG
National :AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,
DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,
LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NA,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,
SL,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,YU,ZA,ZM,ZW

JMG LAH

The International Bureau of WIPO 34, chemin des Colombettes 1211 Geneva 20, Switzerland	Authorized officer: Norbert RIGHETTO
Facsimile No. (41-22) 338.70.80	Telephone No. (41-22) 338 9889

Form PCT/IB/301 (January 2004) 008438430

PCT REQUEST

Original (for SUBMISSION)

0	For receiving Office use only	
0-1	International Application No.	
0-2	International Filing Date	
0-3	Name of receiving Office and "PCT International Application"	

0-4	Form PCT/RO/101 PCT Request	
0-4-1	Prepared Using	PCT-SAFE [EASY mode] Version 3.50 (Build 0002.162)
0-5	Petition The undersigned requests that the present international application be processed according to the Patent Cooperation Treaty	
0-6	Receiving Office (specified by the applicant)	United States Patent and Trademark Office (USPTO) (RO/US)
0-7	Applicant's or agent's file reference	229789
I	Title of invention	LIGHTWEIGHT INPUT/OUTPUT PROTOCOL
II	Applicant	
II-1	This person is	applicant only
II-2	Applicant for	all designated States except US
II-4	Name	MICROSOFT CORPORATION
II-5	Address	One Microsoft Way Redmond, Washington 98052 United States of America
II-6	State of nationality	US
II-7	State of residence	US
II-8	Telephone No.	425-882-8080
II-9	Facsimile No.	425-936-7329
III-1	Applicant and/or inventor	
III-1-1	This person is	applicant and inventor
III-1-2	Applicant for	US only
III-1-4	Name (LAST, First)	MOHAMED, Ahmad, H.
III-1-5	Address	29 210th Place SE Sammamish, Washington 98074 United States of America
III-1-6	State of nationality	US
III-1-7	State of residence	US

III-2	Applicant and/or inventor	
III-2-1	This person is	applicant and inventor
III-2-2	Applicant for	US only
III-2-4	Name (LAST, First)	VOELLM, Anthony, F.
III-2-5	Address	9800 229th Lane NE Redmond, Washington 98053 United States of America
III-2-6	State of nationality	US
III-2-7	State of residence	US
IV-1	Agent or common representative; or address for correspondence	
	The person identified below is hereby/ has been appointed to act on behalf of the applicant(s) before the competent International Authorities as:	agent
IV-1-1	Name (LAST, First)	CONKLIN, John, B.
IV-1-2	Address	LEYDIG, VOIT & MAYER, LTD. Two Prudential Plaza, Suite 4900 180 N Stetson Av Chicago, Illinois 60601-6780 United States of America
IV-1-3	Telephone No.	312-616-5600
IV-1-4	Facsimile No.	312-616-5700
IV-1-5	e-mail	mail@leydig.com
IV-1-6	Agent's registration No.	30369
V	DESIGNATIONS	
V-1	The filing of this request constitutes under Rule 4.9(a), the designation of all Contracting States bound by the PCT on the International filing date, for the grant of every kind of protection available and, where applicable, for the grant of both regional and national patents.	
VI-1	Priority claim of earlier national application	
VI-1-1	Filing date	31 December 2003 (31.12.2003)
VI-1-2	Number	10/749,959
VI-1-3	Country	US
VI-2	Priority document request	
	The receiving Office is requested to prepare and transmit to the International Bureau a certified copy of the earlier application(s) identified above as item(s):	VI-1

66

229789

3/4

PCT REQUEST

Original (for SUBMISSION)

VII-1	International Searching Authority Chosen	United States Patent and Trademark Office (USPTO) (ISA/US)	
VII-2	Request to use results of earlier search; reference to that search		
VII-2-1	Date	31 December 2003 (31.12.2003)	
VII-2-2	Number	10/749,959	
VII-2-3	Country (or regional Office)	US	
VIII	Declarations	Number of declarations	
VIII-1	Declaration as to the identity of the inventor	-	
VIII-2	Declaration as to the applicant's entitlement, as at the international filing date, to apply for and be granted a patent	-	
VIII-3	Declaration as to the applicant's entitlement, as at the international filing date, to claim the priority of the earlier application	-	
VIII-4	Declaration of inventorship (only for the purposes of the designation of the United States of America)	-	
VIII-5	Declaration as to non-prejudicial disclosures or exceptions to lack of novelty	-	
IX	Check list	number of sheets	electronic file(s) attached
IX-1	Request (including declaration sheets)	4	✓
IX-2	Description	25	-
IX-3	Claims	4	-
IX-4	Abstract	1	✓
IX-5	Drawings	23	-
IX-7	TOTAL	57	
	Accompanying Items	paper document(s) attached	electronic file(s) attached
IX-8	Fee calculation sheet	✓	-
IX-17	PCT-SAFE physical media	-	✓
IX-19	Figure of the drawings which should accompany the abstract	1	
IX-20	Language of filing of the international application	English	
X-1	Signature of applicant, agent or common representative		
X-1-1	Name (LAST, First)	COSKUN, John B.	
X-1-2	Name of signatory		
X-1-3	Capacity		

67

FOR RECEIVING OFFICE USE ONLY

10-1	Date of actual receipt of the purported international application	
10-2	Drawings:	
10-2-1	Received	
10-2-2	Not received	
10-3	Corrected date of actual receipt due to later but timely received papers or drawings completing the purported international application	
10-4	Date of timely receipt of the required corrections under PCT Article 11(2)	
10-5	International Searching Authority	ISA/US
10-6	Transmittal of search copy delayed until search fee is paid	

FOR INTERNATIONAL BUREAU USE ONLY

11-1	Date of receipt of the record copy by the International Bureau	

67

68

PATENT COOPERATION TREATY

From the RECEIVING OFFICE

To:
JOHN B. CONKLIN
LEYDIG, VOIT & MAYER, LTD.
TWO PRUDENTIAL PLAZA, SUITE 4900
180 N STETSON AV
CHICAGO, ILLINOIS 60601-6780

PCT

NOTIFICATION OF THE INTERNATIONAL APPLICATION NUMBER AND OF THE INTERNATIONAL FILING DATE

(PCT Rule 20.5(c))

Date of mailing *(day/month/year)*	09 Sep 2004
Applicant's or agent's file reference: 229789	IMPORTANT NOTIFICATION

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/US2004/024028	26 Jul 2004	31 Dec 2003

Applicant: MICROSOFT CORPORATION

Title of the invention: LIGHTWEIGHT INPUT/OUTPUT PROTOCOL

1. The applicant is hereby notified that the international application has been accorded the international application number and the international filing date indicated above.

2. The applicant is further notified that the record copy of the international application:

☒ was transmitted to the International Bureau on 09 Sep 2004.

☐ has not yet been transmitted to the International Bureau for the reason indicated below and a copy of this notification has been sent to the International Bureau*:

☐ because the necessary national security clearance has not yet been obtained.

☐ because (reason to be specified):

LEYDIG, VOIT & MAYER RECEIVED SEP 14 2004

* The International Bureau monitors the transmittal of the record copy by the receiving Office and will notify the applicant (with Form PCT/IB/301) of its receipt. Should the record copy not have been received by the expiration of 14 months from the priority date, the International Bureau will notify the applicant (Rule 22.1(c)).

3. FOREIGN TRANSMITTAL LICENSE INFORMATION — Completed by: KD

☐ Additional license for foreign transmittal not required. This subject matter is covered by a license already granted or the equivalent U.S. national application. Refer to that license for information concerning its scope.

☐ License for foreign transmittal not required. 37 CFR 5.11(e)(1) or 37 CFR 5.11(e)(2). However, a license may be required for additional subject matter. See 37 CFR 5.15(b).

☒ Foreign transmittal license granted. 35 U.S.C. 184; 37 CFR 5.11 on 03 Sep 2004 (date):

☐ 37 CFR 5.15(a) ☒ 37 CFR 5.15(b)

Name and mailing address of the receiving Office	Authorized officer
Mail Stop PCT, Commissioner for Patents P.O. Box 1450, Alexandria, VA 22313-1450	Kareem Duncan
Facsimile No. 703-305-3230	Telephone No. (703) 305-3685

Form PCT/RO/105 (July 1992)

JMC LAH

68

LIGHTWEIGHT INPUT/OUTPUT PROTOCOL

TECHNICAL FIELD

[0001] The present invention relates generally to systems and methods of remote file access, and more particularly to techniques for offloading input/output processing using Remote Direct Memory Access (RDMA).

BACKGROUND

[0002] In computing environments it is generally desirable to conserve scarce CPU resources. For some such environments, such as networks of application server nodes, such conservation is especially critical. As networks become faster, they make greater demands on CPUs to process packets and perform I/O operations, resulting in slower application performance. This is particularly detrimental for inherently I/O-intensive applications like databases.

[0003] One approach to remedying this problem is to offload excessive I/O and network processing from the CPU. In a networked environment, using distributed file systems and transport protocols like NFS or SMB/CIFS, it is possible to send I/O requests from a local machine to a remote machine. However, it is not necessarily the case that the local machine will achieve significant processing economies using such approaches.

[0004] In the single machine context, I/O processing burdens can be alleviated by offloading I/O tasks to a direct memory access (DMA) controller. Remote Direct Memory Access (RDMA) technology is a more recently-developed extension of DMA for multiple networked computers. RDMA allows data to be moved between memory buffers on two communicating machines equipped with RDMA-capable network interface cards (NICs) without having to involve the CPU and operating system of either the source or the destination machine. RDMA can be used to offload I/O processing to a remote machine, thereby enabling the local machine to reclaim CPU cycles for applications. RDMA has been exploited in high-speed, high-bandwidth interconnect technologies, such as the Virtual Interface Architecture (VIA), InfiniBand, and iWarp. These interconnects are particularly designed for high-reliability network connections between clusters of server nodes within a data center or other local file-sharing environment.

[0005] Protocols defining the communication between a local offloading node and a remote machine must be designed in order for the capabilities associated with RDMA technology to be fully utilized and their benefits effectively achieved. Therefore, there is a need for the lightweight input/output (LWIO) protocol of the present invention.

SUMMARY OF THE INVENTION

[0006] In accordance with one aspect of the present invention, a system for offloading an I/O task from a first computer to a second computer is provided. The system includes a client running on the first computer and a server running on the second computer. The system further includes one or more RDMA channels linking the first computer and the second computer. The client and server communicate in accordance with an LWIO protocol comprising a network discovery phase and an I/O processing phase. The LWIO protocol is used in association with another network protocol, such as SMB/CIFS, leveraging the security and authentication infrastructure of the second protocol. In order to provide a better security model, the I/O model in the protocol is asymmetric: reads are implemented using RDMA, while writes are implemented using send operations.

[0007] In accordance with another aspect of the present invention, a method for offloading an I/O task from a first computer to a second computer is provided. The method takes advantage of common RDMA-capable communication devices on the two computers and is associated with a lightweight input/output (LWIO) client-server protocol. The protocol generally comprises a discovery phase followed by an I/O processing phase. During the discovery phase, the client and server determine a minimal list of shared RDMA-capable providers. During the I/O processing phase, the client posts I/O requests for offloading to the second machine.

[0008] During the discovery phase, the client initially obtains a server request resume key from the server. The client then opens a pipe to the server, over which the client sends a negotiate request containing a list of RDMA-capable providers on the first machine. The server sends a negotiate response over the pipe containing a list of available providers on the second machine that match providers on the first machine. The client then creates an RDMA connection to the server over a shared provider. The client and the server mutually authenticate the new connection. The client then registers one or more files for use with the server.

[0009] I/O processing request messages include a close message, a cancel message, a read message, a write message, a vectored read message, and a vectored write message. The protocol features an asymmetric I/O model for security reasons. Read data is sent to the client using RDMA write operations, while writes are completed using ordinary sends. Read and write requests can be specified by the client to be completed by the server in polling mode or in interrupt mode. If the client indicates that the completion should not be in polling mode, the server completes the I/O processing request by sending a status block to the first computer by way of RDMA transfer. If the client indicates that the completion should be in polling mode, the client may request that it be woken up by the server upon completion of the I/O by way of an interrupt request message.

[0010] In accordance with another aspect of the present invention, a method for managing buffers in an I/O offload protocol is provided. The method involves the use of a buffer credit mechanism. A server-client credit transaction comprises a three-way handshake initiated and completed by the server. The server sends a delta credit message to the client, including an information field set to a number of credits. If the number is a negative number -N, the client must give up N credits.

[0011] Other aspects of the invention include the above-mentioned features embodied on computer-readable media as computer program products and data structures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] While the appended claims set forth the features of the present invention with particularity, the invention, together with its objects and advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings, of which:

[0013] FIG. 1 is a diagram generally illustrating an exemplary client-server computing environment involving two computers capable of communicating by way of RDMA transfer, within which aspects of the present invention can be incorporated;

[0014] FIG. 2 is a flow diagram generally illustrating initial steps taken in the discovery phase of the LWIO protocol in accordance with an embodiment of the invention;

[0015] FIG. 3 is a diagram generally illustrating a representation of an exemplary server request resume key in accordance with an embodiment of the invention;

[0016] FIG. 4A is a diagram generally illustrating a representation of an exemplary client negotiate request message in accordance with an embodiment of the invention;

[0017] FIG. 4B is a diagram generally illustrating a representation of an exemplary server negotiate response in accordance with an embodiment of the invention;

[0018] FIG. 5 is a flow diagram generally illustrating additional steps taken in the discovery phase of the LWIO protocol in accordance with an embodiment of the invention;

[0019] FIG. 6A is a diagram generally illustrating a representation of an exemplary client authenticate request message in accordance with an embodiment of the invention;

[0020] FIG. 6B is a diagram generally illustrating a representation of an exemplary server authenticate response in accordance with an embodiment of the invention;

[0021] FIG. 6C is a diagram generally illustrating a representation of an exemplary server status response completing authentication in accordance with an embodiment of the invention;

[0022] FIG. 7A is a diagram generally illustrating a representation of an exemplary client register file message in accordance with an embodiment of the invention;

[0023] FIG. 7B is a diagram generally illustrating a representation of an exemplary server status response completing file registration in accordance with an embodiment of the invention;

[0024] FIG. 8 is a flow diagram generally illustrating steps taken with respect to completion of an I/O request in polling mode and in non-polling mode, in accordance with an embodiment of the invention;

[0025] FIG. 9A is a diagram generally illustrating a representation of an exemplary client interrupt request message in accordance with an embodiment of the invention;

[0026] FIG. 9B is a diagram generally illustrating a representation of an exemplary server status response completing an interrupt request in accordance with an embodiment of the invention;

[0027] FIG. 10 is a flow diagram generally illustrating steps taken with respect to a server-client credit transaction in accordance with an embodiment of the invention;

[0028] FIG. 11A is a diagram generally illustrating a representation of an exemplary server delta credit message in accordance with an embodiment of the invention;

[0029] FIG. 11B is a diagram generally illustrating a representation of an exemplary client-to-server credit message in accordance with an embodiment of the invention;

[0030] FIG. 11C is a diagram generally illustrating a representation of an exemplary server status response completing a client-server credit transaction in accordance with an embodiment of the invention;

[0031] FIG. 12A is a diagram generally illustrating a representation of an exemplary client close request message in accordance with an embodiment of the invention;

[0032] FIG. 12B is a diagram generally illustrating a representation of an exemplary server status response completing a close request in accordance with an embodiment of the invention;

[0033] FIG. 13A is a diagram generally illustrating a representation of an exemplary client cancel request message in accordance with an embodiment of the invention;

[0034] FIG. 13B is a diagram generally illustrating a representation of an exemplary server status response completing a cancel request in accordance with an embodiment of the invention;

[0035] FIG. 14A is a diagram generally illustrating a representation of an exemplary client read request message in the non-polling mode case, in accordance with an embodiment of the invention;

[0036] FIG. 14B is a diagram generally illustrating a representation of an exemplary server status response completing a read request in the non-polling mode case, in accordance with an embodiment of the invention;

[0037] FIG. 14C is a diagram generally illustrating a representation of an exemplary client read request message in the polling mode case, in accordance with an embodiment of the invention;

[0038] FIG. 14D is a diagram generally illustrating a representation of an exemplary server I/O status block completing a read request in the polling mode case, in accordance with an embodiment of the invention;

[0039] FIG. 15A is a diagram generally illustrating a representation of an exemplary client write request message in the non-polling mode case, in accordance with an embodiment of the invention;

[0040] FIG. 15B is a diagram generally illustrating a representation of an exemplary server status response completing a write request in the non-polling mode case, in accordance with an embodiment of the invention;

[0041] FIG. 15C is a diagram generally illustrating a representation of an exemplary client write request message in the polling mode case, in accordance with an embodiment of the invention;

[0042] FIG. 15D is a diagram generally illustrating a representation of an exemplary server I/O status block completing a write request in the polling mode case, in accordance with an embodiment of the invention;

[0043] FIG. 16A is a diagram generally illustrating a representation of an exemplary client vectored read request message in the non-polling mode case, in accordance with an embodiment of the invention;

[0044] FIG. 16B is a diagram generally illustrating a representation of an exemplary server status response completing a vectored read request in the non-polling mode case, in accordance with an embodiment of the invention;

[0045] FIG. 16C is a diagram generally illustrating a representation of an exemplary client vectored read request message in the polling mode case, in accordance with an embodiment of the invention;

[0046] FIG. 16D is a diagram generally illustrating a representation of an exemplary server I/O status block completing a vectored read request in the polling mode case, in accordance with an embodiment of the invention;

[0047] FIG. 17A is a diagram generally illustrating a representation of an exemplary client vectored write request message in the non-polling mode, non-collapsed case, in accordance with an embodiment of the invention;

[0048] FIG. 17B is a diagram generally illustrating a representation of an exemplary client vectored write request message in the non-polling mode, collapsed case, in accordance with an embodiment of the invention;

[0049] FIG. 17C is a diagram generally illustrating a representation of an exemplary client vectored write request message in the polling mode, collapsed case, in accordance with an embodiment of the invention;

[0050] FIG. 17D is a diagram generally illustrating a representation of an exemplary server status response completing a vectored write request in the non-polling mode case, in accordance with an embodiment of the invention; and

[0051] FIG. 17E is a diagram generally illustrating a representation of an exemplary server I/O status block completing a vectored write request in the polling mode case, in accordance with an embodiment of the invention.

DETAILED DESCRIPTION

[0052] Certain embodiments of the present invention are discussed below with reference to FIGS. 1-17E. However, those skilled in the art will readily appreciate that the detailed description given herein with respect to these figures is for illustrative purposes, and that the invention extends beyond these embodiments.

[0053] FIG. 1 is a schematic diagram generally illustrating certain features of a representative networked client/server environment within which aspects of the present invention may be incorporated. Depicted in FIG. 1 are two computer machines, labeled Host A 101 and Host B 121. While the invention may be practiced in an environment involving computers of many different types and uses, in one representative scenario Host A 101 functions as an application server machine charged with I/O-intensive work, such as a database server.

[0054] Each of Host A 101 and Host B 121 include a number of network interface cards (NICs) 109, 111, 113, 133, 135, 137 allowing for networked data communication from one machine to the other. Among these NICs are NICs 109, 111, 135, 137 permitting RDMA data transfer. As illustrated, a non-RDMA network link 119 and an RDMA channel 117 are present between the two hosts 101, 121.

[0055] Executing on Host A 101 is an LWIO client application 103, associated with an application responsible for processing I/O tasks which interacts with kernel-mode I/O read/write services 105. The LWIO client 103 is used to offload I/O processing from Host A 101 to Host B 121. On Host B 121 an LWIO server 123 is executing. In accordance with the LWIO protocol described herein, the LWIO client 103 communicates with the LWIO server 123. The LWIO client 103 and the LWIO server 123 make use of posted buffers 107, 127, enabling file-associated data to be transferred directly by way of the RDMA channel connection 117. By way of LWIO protocol messages, read and write tasks are offloaded to Host B 121. The server 123 passes on I/O requests to the file system 129, which serves as the interface to the hard disk 131.

74

[0056] Typically, two kinds of messages are associated with an RDMA connection 117. The first type is an ordinary network send/receive, generating an interrupt at the destination machine. The second type is an RDMA read/write, in which memory space on the remote machine is accessed without the aid of the remote CPU and thus without having to generate an interrupt. The remote CPU determines the memory regions that are exposed for RDMA but typically is unaware of when an RDMA operation is performed.

[0057] In an embodiment of the invention described herein, the LWIO protocol is used in association with another network protocol, such as SMB or CIFS, in order to take advantage of the existing security and authentication infrastructure of the other protocol. This helps to minimize the overhead of the LWIO protocol. As illustrated in FIG. 1, the LWIO server 123 on Host B 121 operates above an SMB server 125. An SMB client (not shown) similarly runs on Host A 101 and interacts with the LWIO client application 103.

[0058] The LWIO protocol comprises two phases: a discovery phase followed by an I/O phase. In data structures associated with an embodiment described herein, data sizes are as follows:

BYTE	unsigned 8-bit integer
CHAR	8-bit ASCII character
UINT16	unsigned 16-bit integer
UINT32	unsigned 32-bit integer
UINT64	unsigned 64-bit integer
INT16	signed 16-bit integer
INT32	signed 32-bit integer
INT64	signed 64-bit integer
WCHAR	16-bit Unicode character
PVOID32	32-bit pointer
PVOID64	64-bit pointer

[0059] FIG. 2 illustrates steps taken in the discovery phase of the LWIO protocol in an embodiment of the invention. With respect to the host on which the LWIO server is executing, at step 201 the LWIO server registers with the SMB/CIFS server running on that host machine. In accordance with this registration, at step 203 the SMB/CIFS server notifies a SMB/CIFS client running on a remote host that the LWIO server is available. At step 205 the LWIO client requests a server request resume key. The resume key is an authentication mechanism that has been disclosed in another application having the same assignee as the present application. "Method and System for Accessing a File (Resume

Key)." U.S. Patent Application Serial No. _____, filed on October 24, 2003, which is hereby incorporated herein in its entirety by reference.

[0060] At step 207 the LWIO server passes the server request resume key back to the client. In an embodiment of the invention the server request resume key has the following structure:

```
typedef struct _SRV_RESUME_KEY {
     UINT64              ResumeKey;
     UINT64              Timestamp;
     UINT64              Pid;
} SRV_RESUME_KEY, *PSRV_RESUME_KEY;

typedef struct _SRV_REQUEST_RESUME_KEY {
     SRV_RESUME_KEY      Key;
     UINT16              ContextLength;
     BYTE                Context[1];
} SRV_REQUEST_RESUME_KEY, *PSRV_REQUEST_RESUME_KEY;
```

FIG. 3 provides an illustrative representation of the server request resume key 219. ResumeKey 221, Timestamp 223, and Pid 225 are generated on the server and are opaque to the client. Context 229 is an array containing a UNC name that is used by the LWIO client to contact the server. ContextLength 227 is the number of bytes in Context 229.

Network Discovery

[0061] When the client application receives the server request resume key 219, it retrieves the server UNC name from the Context field 229. Returning to FIG. 2, at step 209 the client opens a pipe to the LWIO server. The pipe is used for automatic discovery of RDMA-capable devices that are available in the network, in a manner described further below. This is an important and useful feature of the present invention; address resolution mechanisms like ARP are generally absent from VIA networks and similar networks.

[0062] The client next queries the server for a list of its RDMA-capable devices ("providers") that are available for use with the LWIO protocol. The querying is accomplished by way of a negotiate request, which the client constructs and sends to the server over the newly-opened pipe at step 211. In an embodiment of the invention, the negotiate request has the following structure:

```
typedef struct {
     LWIO_CONTROL_HEADER;
     WCHAR               ClientName[LWIO_MAX_HOST_NAME];
     UUID                Key;
     UINT16              ResponseLength;
     UINT16              ProviderCount;
     LwioAddressBlk_t    ProviderList[1];
```

76

```
} LwioNegotiateRequest_t;

typedef struct {
     CHAR           ProtocolId[4];
     UINT32         RevId;
     UINT16         Opcode;
     UINT16         Length;
} LWIO_CONTROL_HEADER;

typedef struct _GUID {
     UINT32         Data1;
     UINT16         Data2;
     UINT16         Data3;
     BYTE           Data4[8];
} GUID, UUID;

typedef struct {
     WCHAR          Name[LWIO_MAX_PROVIDER_NAME];
     UINT16         InstanceCount;
     LWIO_NET_ADDRESS InstanceTable[1];
} LwioAddressBlk_t;

typedef struct _LWIO_NET_ADDRESS {
     UINT16         HostAddressLen;
     UINT16         DiscriminatorLen;
     BYTE           HostAddressFollowedByDiscriminator[1];
} LWIO_NET_ADDRESS;
```

[0063] FIG. 4A provides an illustrative representation of the negotiate request packet 231 in an embodiment of the invention. The negotiate request includes a control header 233, a fixed-length Unicode client name field 235, a client UUID 237 used as a key, a local buffer size 239 for receiving a response, and the list of providers 241. In the control header 233, the ProtocolId 'LWIO' 243 is stored as the first four bytes of the header.

[0064] RevId 245 holds a currently defined value 0x1001, LWIO_REV_ID. Opcode 247 holds a currently defined value 0xfe, LWIO_CONTROL_OPCODE_NEGOTIATE. Length 249 is the size in bytes of the complete packet to be sent to the server, including all opcode-specific data.

[0065] ClientName 235 is used by the server to identify the client. Key 237 is used in a subsequent network-specific authentication procedure, as described below. ResponseLength 239 is the size of the buffer for receiving a negotiate response from the server, as described below. ProviderCount 251 is the number of providers associated with the client machine and about which the client is informing the server. The provider list 241 contains the list of ProviderCount providers.

[0066] In an element of the provider list 241, Name 253 is the name of the provider. In order for compatible networks to be detected, the client and the server should preferably use the same name for the same provider. InstanceCount 255 is the number of devices of a particular provider type. The instance table 257 is a table of network/discriminator pairs, in which a pair serves to describe, in a device-specific way, how to form a remote connection. HostAddressLen 259 is the length of the network-specific host address 263. DiscriminatorLen 261 is the length of the network-specific discriminator 265. Following these length fields are the HostAddressLen bytes of the host address 263 and the DiscriminatorLen bytes of the discriminator 265.

[0067] Returning to FIG. 2, having received the negotiate request with the client's list of providers, at step 213 the server determines which RDMA-capable communication devices it has in common with the client. At step 215 the server sends a negotiate response to the client over the pipe, including a list of shared providers. In an embodiment of the invention, the negotiate response has the following structure:

```
typedef struct {
     LWIO_CONTROL_HEADER;
     WCHAR                    SrvName[LWIO_MAX_HOST_NAME];
     UUID                     Key;
     UINT16                   ProviderCount;
     LwioAddressBlk_t         ProviderList[1];
} LwioNegotiateResponse_t;
```

[0068] FIG. 4B provides an illustrative representation of the negotiate response 267 in an embodiment of the invention. The control header 269 is as in the negotiate request, except that Length 271 now reflects the size of the response message 267. SrvName 273 holds the name of the server. Key 275 is a server-generated GUID for use by the client. As explained further below, the client sends the Key back to the server in an authenticate request over a new connection using one of the common communication devices. ProviderCount 277 is the number of providers in the provider list 279. The provider list 279 contains a list of providers common to the server and the client. There is no guarantee that the client can actually connect to these providers.

[0069] Returning to FIG. 2, at this point the server and the client have shared communication device information, and the minimal list of common providers has been determined. At step 217 the client creates one or more RDMA connections to the LWIO server over one or more of the shared devices. In an embodiment of the invention, as described herein, the following opcodes are defined for client-to-server communication:

```
#define    LWIO_OPCODE_READ              0x0
#define    LWIO_OPCODE_WRITE             0x1
#define    LWIO_OPCODE_VEC_READ          0x2
#define    LWIO_OPCODE_VEC_WRITE         0x3
#define    LWIO_OPCODE_CLOSE             0x4
#define    LWIO_OPCODE_CANCEL            0x5
#define    LWIO_OPCODE_AUTH              0x6
#define    LWIO_OPCODE_REGISTER          0x7
#define    LWIO_OPCODE_CREDIT            0x8
#define    LWIO_OPCODE_INTERRUPT         0x9
```

The following defined flags are used as modifiers in client-to-server communication:

```
#define    LWIO_HDR_FLAG_INTERRUPT       0x80
#define    LWIO_HDR_FLAG_CONTROL         0x40
#define    LWIO_HDR_FLAG_COLLAPSE_IO     0x20
```

The corresponding client-to-server messages in the LWIO protocol feature a common header structure. The common header has the following format in an embodiment of the invention:

```
typedef struct {
     UINT32                          Length;
     union {
     UINT32                          Status;
     struct {
          BYTE                       Opcode;
          BYTE                       Flags;
          BYTE                       Credits;
          BYTE                       Marker;
     };
     };

     struct {
     UINT16                          Fid;
     UINT16                          Sequence;
     UINT32                          Tid;
     };

     UINT64                          Offset;

     // data buffer block
     struct {
```

21

```
    PVOID64                         DataVa;
    union {
        UINT32                      DataMh;
        struct {
        UINT16                      NumPages;
        UINT16                      PageSize;
        } Vec;
    };
    };

    // io status block
    union {
     struct {
        UINT32                      IosMh;
        PVOID64                     IosVa;
     };
     struct {
        UINT32                      ImmediateCookie;
        UINT64                      Cookie;
     };
    };
} LWIO_COMMON_HEADER;
```

Connection Authentication

[0070] FIG. 5 illustrates steps taken by the client and the server in an embodiment of the invention, during the remainder of the initial phase of the LWIO protocol. At step 601 the client establishes a connection to the server over a shared communication device, as explained above. The client and the server now mutually authenticate the new connection. At step 603 the client sends an authentication request message (LWIO_OPCODE_AUTH) to the server. Authentication is done in order to prevent server-side and client-side spoofing. If the authentication is not timely completed, the connection is terminated.

[0071] FIG. 6A provides an illustrative representation of the client authenticate request message in an embodiment of the invention. The authenticate message 617 comprises the common header 619 followed by an LWIO_AUTH_PARAMS structure 621. In the header 619, Length 623 is set to the number of bytes sent to the server (the size of the common header 619 plus the size of the LWIO_AUTH_PARAMS 621). Opcode 625 is set to LWIO_OPCODE_AUTH (0x6). Flags 627 is set to LWIO_HDR_FLAG_INTERRUPT. Cookie 629, in this and the other client protocol messages, is set to a value chosen by the client and is sent back in the server reply. The Cookie value is typically used to match a request with a server reply. DataVa 631 is set to the address to which the server should RDMA the server authentication parameters. DataMh 633 holds the RDMA memory handle associated with DataVa 631.

80

[0072] In an embodiment of the invention, the LWIO_AUTH_PARAMS structure has the following format:

```
#define LWIO_AUTH_OPTION_END                    0
#define LWIO_AUTH_OPTION_KEY                    1
#define LWIO_AUTH_OPTION_SESSION_ID             2
#define LWIO_AUTH_OPTION_SIGNATURE              3

#define LWIO_AUTH_OPTION_KEY_LENGTH             16
#define LWIO_AUTH_OPTION_SESSION_ID_LENGTH      8
#define LWIO_AUTH_OPTION_SIGNATURE_LENGTH       16

typedef struct {
     UCHAR                  OptionCode;
     UCHAR                  OptionLen;
     BYTE                   OptionData[1];
} LWIO_AUTH_OPTIONS, *LPLWIO_AUTH_OPTIONS;

typedef struct {
     CHAR                   Magic[4]; // 'LWIO'
     UINT16                 RevId;
     UINT16                 Endian;
     UINT16                 PageSize;
     UINT16                 BaseSequence;
     UINT32                 MaxRdmaWindowSize;
     UINT32                 MaxSendBufferSize;
     UINT32                 MaxRecvBufferSize;
     UINT16                 HeaderSize;
     UINT16                 Credits;
     UINT16                 RdmaReadSupported;
     LWIO_AUTH_OPTIONS      Options[1];
} LWIO_AUTH_PARAMS, *LPLWIO_AUTH_PARAMS;
```

[0073] In the authenticate message 617, an LWIO_AUTH_PARAMS 621 forms the second part of the packet. Magic 635 is set to 'LWIO'. RevId 637 is set to LWIO_REV_ID. Endian 639 is set to sizeof(ULONG_PTR). PageSize 641 is set to the CPU page size (4k on 32-bit machines and 8k on 64-bit machines). BaseSequence 643 is set to 0. MaxRdmaWindowSize 645 is intended to be set to the maximum number of bytes that the client can accept in an RDMA transfer; in the depicted embodiment it is set to 64k. MaxSendBufferSize 647 is intended to be set to the number of bytes that the client can send to the server in a single request; in the depicted embodiment it is set to 1k. MaxRecvBufferSize 649 is intended to be set to the number of bytes that the client has posted to receive data from the server; in the depicted embodiment it is set to 16 bytes. HeaderSize 651 is set to the number of bytes in the LWIO control header 619. Credits 652

81

is set to the initial number of buffer credits that the client wishes to have. The use of credits is explained further below. The server may or may not satisfy the client's request. RdmaReadSupported 653 is set to 0 if the client does not support RDMA read operations and is set to 1 if the client does support RDMA read.

[0074] Part of the LWIO_AUTH_PARAMS structure is a set of one or more options. The options are used to make authentication more flexible. Each option has an option code, length and data, except for the last option in the list, LWIO_AUTH_OPTION_END, which has the option code only, serving as a null option terminating the list of options. In the authenticate message, the client sends the server the following options: Key (LWIO_AUTH_OPTION_KEY) and a signature (LWIO_AUTH_OPTION_SIGNATURE). Key 655 is set to the key previously returned by the server in the negotiate response. Signature 657 is an MD5 signing of the LWIO_AUTH_PARAMS 621 excluding the signature.

[0075] Returning to FIG. 5, at step 605, if the Key sent in the authenticate message matches the key that was returned in the negotiate response over the pipe, the server RDMAs to the client as an authenticate response an LWIO_AUTH_PARAMS structure, including an eight-byte SessionId, to the DataVa address and associated DataMh memory handle provided by the client in the authenticate message. At step 607 the server sends an LWIO_MSG_STATUS_RESPONSE to complete the authentication.

[0076] FIG. 6B provides an illustrative representation of the LWIO_AUTH_PARAMS structure 659 returned by the server in an embodiment of the invention. Magic 661 is set to 'LWIO'. RevId 663 is set to LWIO_REV_ID. Endian 665 is set to sizeof(ULONG_PTR). PageSize 667 is set to the CPU page size. BaseSequence 669 is intended to be set to (client BaseSequence + 1). MaxRdmaWindowSize 671 is intended to be set to the maximum number of bytes that the client can accept in an RDMA transfer; in the depicted embodiment it is set to 512k. MaxSendBufferSize 673 is intended to be set to the number of bytes that the server sends to the client in a single response; in the depicted embodiment it is set to 16 bytes. MaxRecvBufferSize 675 is intended to be set to the number of bytes that the server has pre-posted to receive data from the client; in the depicted embodiment it is set to 8k. HeaderSize 677 is set to the number of bytes in the common header. Credits 679 is set to the initial number of credits that the server has available for the client. RdmaReadSupported 681 is set to 0 if the server does not support RDMA read and is set to 1 if the server does support RDMA read. The server sends the following options: Key (LWIO_AUTH_OPTION_KEY) 683, SessionId (LWIO_AUTH_OPTION_SESSION_ID) 685, and a Signature (LWIO_AUTH_OPTION_SIGNATURE) 687. Key 683 is set to the Key that the client had sent previously in the Negotiate Request. The SessionId 685 value is

82

used by the client in registering client files with the server, as explained below. Signature 687 is an MD5 signing of the LWIO_AUTH_PARAMS excluding the Signature.

|0077| In an embodiment of the invention, the LWIO_MSG_STATUS_RESPONSE structure has the following format:

```
typedef struct _LWIO_IO_STATUS_BLOCK {
     UINT32                    Information;
     UINT32                    Status;
} LWIO_IO_STATUS_BLOCK, *LPLWIO_IO_STATUS_BLOCK;

typedef struct _LWIO_MSG_STATUS_RESPONSE {
     UINT64                    Cookie;
     LWIO_IO_STATUS_BLOCK      Ios;
} LWIO_MSG_STATUS_RESPONSE, *LPLWIO_MSG_STATUS_RESPONSE;
```

FIG. 6C provides an illustrative representation of the LWIO_MSG_STATUS_RESPONSE 689 returned by the server to complete the authentication in an embodiment of the invention. Cookie 691 is set to the cookie value set by the client in the header of the authenticate message. Information 693 is set to the number of bytes of LWIO_AUTH_PARAMS plus eight bytes. Status 695 is set to 0x0 (signifying success) or 0xC0000022 (signifying "access denied").

File Registration

|0078| Returning to FIG. 5, at step 609, when the new connection has been mutually authenticated by the client and the server, the client begins registering files for use with the server. File operations for a file are not processed over a link until the client has registered the file for use with the server.

|0079| FIG. 7A provides an illustrative representation of the register file message sent by the client to the server in an embodiment of the invention. The registration message 701 comprises the common header 703 followed by an LWIO_FID_PARAMS structure 705. Length 707 is set to the number of bytes sent to the server (the size of the header 703 plus the size of the LWIO_FID_PARAMS 705). Opcode 709 is set to LWIO_OPCODE_REGISTER (0x7). Flags 711 is set to LWIO_HDR_FLAG_INTERRUPT. In this client message and subsequent client messages, Credits 713 is set to the number of pending I/O requests on the client. The Credits field serves as a hint to the server to allocate more credits to the connection, thus allowing additional outstanding I/O requests, as explained further below. The number of outstanding client requests at any one time cannot exceed the "Credits" value. As before, Cookie 715 is set to a client-specified value.

95

[0080] In an embodiment of the invention, the LWIO_FID_PARAMS structure has the following format:

```
typedef struct {
     SRV_RESUME_KEY       ResumeKey;
     INT64                SessionId;
     UINT32               FlagsAndAttributes;
} LWIO_FID_PARAMS, *LPLWIO_FID_PARAMS;
```

In the LWIO_FID_PARAMS 705 of the register file message 701, ResumeKey 717 is set to the server request resume key that was returned over the initial file access channel. SessionId 719 is set to the SessionId that was returned by the server during the connection authentication stage. FlagsAndAttributes 721 is set to the Win32 Create Flags used initially to open the file.

[0081] Returning to FIG. 5, at step 611 the server responds with an LWIO_MSG_STATUS_RESPONSE to complete the file registration. FIG. 7B provides an illustrative representation of the LWIO_MSG_STATUS_RESPONSE 723 sent by the server in an embodiment of the invention. Information 725 is set to the Fid (File ID) to be used when sending I/O requests. Status 727 is set to 0x0 (success) or another NTSTATUS code on failure. Cookie 729 is set to the cookie value that the client set in the header of the register file message.

I/O Processing

[0082] At this point client connections are established and files have been registered, and the I/O processing phase of the LWIO protocol begins. One key feature of embodiments of the LWIO protocol is an asymmetric I/O model for reads and writes. Read operations are implemented using RDMA, while writes are implemented using send operations. Writes are not implemented using RDMA in order to provide a better security model. If the server exposes its address space over the NIC for RDMA it introduces a data corruption vulnerability that can be exploited by a malicious client. In this scenario, the malicious client issues, in a loop, RDMA write operations on a given server virtual address. Because the server address space is finite and at some point server virtual addresses must be reused, the malicious client eventually catches the server using the same virtual address for a different connection, causing the data to be written into a server buffer that might be associated with a different client. The asymmetric I/O model in the LWIO protocol guards against this possibility. This feature is a principal difference between the LWIO protocol and other RDMA-based file transfer protocols, such as DAFS.

[0083] Returning to FIG. 5, at step 613, the client begins posting I/O processing requests. Server-to-client completions of I/O requests are either in non-polling mode or

81

polling mode. In non-polling mode, I/O completions are interrupt-based, using ordinary send/receive messages. In polling mode, I/O completions use RDMA and are not interrupt-based.

[0084] The flow diagram of FIG. 8 generally illustrates, from the general perspective of the LWIO server, steps taken in an embodiment of the invention with respect to completing an I/O request in polling mode or non-polling mode. A client I/O request specifies whether the server should send back a post-send (interrupting the CPU) or an RDMA message. At step 801, the server determines whether an LWIO_HDR_FLAG_INTERRUPT flag is set in the common header of the client I/O request message. If this flag is set, at step 803 the server completes the client request by way of an LWIO_MSG_STATUS_RESPONSE using an ordinary send. If the LWIO_HDR_FLAG_INTERRUPT flag is not set (polling mode), then the server completes the client request by RDMAing an LWIO_IO_STATUS_BLOCK to the client, as indicated at step 805.

Wakeup of Client in Polling Mode

[0085] In polling mode, the client may wish to sleep while waiting for an I/O completion from the server. Completions in this case are sent by way of RDMA to the client, so a mechanism is needed to wake up the client to notify it that a completion has occurred. If the client wishes to be woken up, it sends an interrupt request (LWIO_OPCODE_INTERRUPT) message to the server, received by the server at step 807 of FIG. 8. A server that receives an interrupt request will not send a response until an I/O request has completed on the server (step 809). The completion is sent to the client at step 811 by way of an ordinary send, interrupting the client. Only one interrupt message can be outstanding for a given client connection.

[0086] FIG. 9A provides an illustrative representation of the interrupt request message sent by the client to the server in an embodiment of the invention. The message comprises the common header 815. Opcode 817 is set to LWIO_OPCODE_REGISTER (0x9). Flags 819 is set to (LWIO_HDR_FLAG_INTERRUPT | LWIO_HDR_FLAG_CONTROL) (0xC0). Credits 821 is set to the number of pending I/O requests on the client, and Cookie 823 is set to a client-specified value.

[0087] The server responds to the interrupt request message after another I/O request has been processed. FIG. 9B provides an illustrative representation of the LWIO_MSG_STATUS_RESPONSE message 825 sent by server in an embodiment of the invention. Information 827 is set to 0. Status 829 is set to 0x0 (success) or another NTSTATUS code on failure. Cookie 831 is set to the Cookie value in the header of the interrupt request sent by the client.

Credits

[0088] As has been noted, all client-to-server I/O requests include a credits field in the header. The credits field is a hint to the server regarding the number of outstanding I/O requests that the client would like to send to the server. It is the responsibility of the server to manage credits. Credits provide a novel solution to the problem of flushing buffers. If the client currently has N credits, it is required to post N+1 receive buffers in order for the server to send the client a credit message. The server has only one outstanding credit request along a client connection at any one time. Credit messages are always sent in interrupt mode.

[0089] A credit transaction comprises a server-initiated three-way handshake between client and server. FIG. 10 generally illustrates the steps comprising the credit transaction in an embodiment of the invention. At step 1001 the server sends a delta credit request message along a client connection.

[0090] FIG. 11A provides an illustrative representation of the server delta credit message in an embodiment of the invention. This message takes the form of an LWIO_MSG_STATUS_RESPONSE 1011. Credits correspond to buffers. Information 1013 is set to the number of credits that the client should give up (a negative number) or the number of credits (extra buffers) that the server has newly allocated for the client's use (a positive number). Status 1015 is set to LWIO_NOTIFY_CREDIT (0x1). Cookie 1017 is set to 0.

[0091] Returning to FIG. 10, the client receives the credit message from the server. The client is required to respond with an LWIO_OPCODE_CREDIT message to the server on the same connection. This message signifies either the releasing of a single credit or notifying the server of the number of the newly-allocated credits that the client has used. If the Information field in the server credit message contains a negative number, -N (step 1003), the client sends N LWIO_OPCODE_CREDIT messages (one for each credit that it is required to give up), indicated as step 1005. If the Information field is positive, then the client sends only one LWIO_OPCODE_CREDIT message, indicated as step 1007.

[0092] FIG. 11B provides an illustrative representation of the LWIO_OPCODE_CREDIT message sent by the client in an embodiment of the invention. The LWIO_OPCODE_CREDIT message 1019 comprises a common header 1021. Opcode 1023 is set to LWIO_OPCODE_CREDIT (0x8). Flags 1025 is set to LWIO_HDR_FLAG_INTERRUPT (0x80). Credits 1027 is set to the number of pending I/O requests on the client. Cookie 1031 is set to a client-specified value. If the client received a positive delta credit message, the upper 32 bits of Offset 1029 are set to the number of credits allocated by the server that the client did not use. Once the client returns

96

a value greater than zero in this field, the server normally does not send another positive update message until at least one negative update is sent. Typically, the client returns zero.

[0093] As noted above, if the client received a negative (-N) delta credit message, the client is required to send N credit messages to the server, one for each credit that it is giving up. The upper 32 bits of Offset 1029 in this case are accordingly set to -N, -(N-1), ... -1. When the server receives the client credit message with the upper 32 bits of Offset 1029 set to -1, the server assumes that the client has finished processing the server credit message and is eligible to receive new credit messages.

[0094] Returning to FIG. 10, the server completes the three-way handshake by sending an LWIO_MSG_STATUS_RESPONSE message to the client, indicated as step 1009. FIG. 11C provides an illustrative representation of the LWIO_MSG_STATUS_RESPONSE 1033 sent by the server in an embodiment of the invention. Information 1037 is set to 0. If the upper 32 bits of Offset in the header of the LWIO_OPCODE_CREDIT message sent by the client was greater than or equal to zero, Status 1039 is set to 0x0, signifying success. If the upper 32 bits of Offset were set to a negative number, the server sets Status 1039 to LWIO_CREDIT_NOTIFY in order to allow the client to retire the credit. Cookie 1035 is set to the Cookie value set by the client in the common header of the LWIO_OPCODE_CREDIT message.

Close

[0095] The close message is used to stop I/O processing for a particular Fid that was exchanged during the registration stage. Once the server responds, any new requests will fail until the Fid is recycled. FIG. 12A provides an illustrative representation of the close message sent by the client in an embodiment of the invention. The close message 1041 comprises a common header 1043. Opcode 1045 is set to LWIO_OPCODE_CLOSE (0x4). Flags 1047 is set to LWIO_HDR_FLAG_INTERRUPT (0x80). Credits 1049 is set to the number of pending I/O requests on the client. Cookie 1053 is set to a client-specified value. Fid 1051 is set to the File Id of the file that is to be closed.

[0096] The server responds with an LWIO_MSG_STATUS_RESPONSE. FIG. 12B provides an illustrative representation of the close completion LWIO_MSG_STATUS_RESPONSE 1055 returned by the server in an embodiment of the invention. Information 1059 is set to 0. Status 1061 is set to 0, indicating success. Cookie 1057 is set to the Cookie value that was set in the client close request.

Cancel

[0097] The cancel message is used to stop I/O processing for a particular Fid that was exchanged during the registration stage. When the cancel is issued, the server completes the

87

request. However, I/O requests that cannot be canceled may still proceed on the server. FIG. 13A provides an illustrative representation of the cancel message sent by the client in an embodiment of the invention. The cancel message 1063 comprises a common header 1065. Opcode 1067 is set to LWIO_OPCODE_CANCEL (0x5). Flags 1069 is set to LWIO_HDR_FLAG_INTERRUPT (0x80). Credits 1071 is set to the number of pending I/O requests on the client. Cookie 1075 is set to a client-specified value. Fid 1073 is set to the File Id on which the cancel is being issued.

[0098] The server completes the cancel with an LWIO_MSG_STATUS_RESPONSE message. FIG. 13B provides an illustrative representation of the cancel completion LWIO_MSG_STATUS_RESPONSE 1077 returned by the server in an embodiment of the invention. Information 1081 is set to 0. Status 1083 is set to 0, indicating success. Cookie 1079 is set to the Cookie value that was set in the client cancel request

[0099] The read message is used to obtain data from a particular Fid that was exchanged during the registration stage. For a read request smaller than one kilobyte, if the user buffer is not registered with the NIC, the data is received into an internal pre-registered buffer, and a copy is performed into the user buffer once the data is received from the server. This is done because it is more efficient to copy small amounts of data rather than to register small user buffers. For large reads the user buffer is registered and the data is received directly by way of RDMA write. The amount of data read pursuant to a single read request is limited by the server MaxRdmaWindowSize.

[0100] FIGS. 14A and 14C provide illustrative representations of the read message sent by the client in an embodiment of the invention, with FIG. 14A giving the non-polling case and FIG. 14C giving the polling case. The read message 1401 comprises a common header 1403. Length 1405 is set to the number of bytes to be read from the associated file. Opcode 1407 is set to LWIO_OPCODE_READ (0x0). Offset 1417 is set to the byte location at which the file read is to begin. Marker 1413 is set to 0xFF. Flags 1409, 1427 is set to 0x0 in the polling case 1427 or LWIO_HDR_FLAG_INTERRUPT (0x80) in the non-polling case 1409. Credits 1411 is set to the number of pending I/O requests on the client. Fid 1415 is set to the File Id on which to issue the I/O. DataVa 1419 is set to the address to which the read data is to be RDMAed, and DataMh 1421 is set to the associated memory handle.

[0101] In the non-polling case, ImmediateCookie 1423 and Cookie 1425 are set to client-specified values. The server can complete the read request in this case with an LWIO_MSG_STATUS_RESPONSE by way of a normal send, or with an RDMA with immediate data if the read is successful. The immediate data of the RDMA write is accordingly set to the ImmediateCookie value of the read request. In the polling case, IosVa 1431 is set to the location to which the server response status

88

(LWIO_IO_STATUS_BLOCK) is RDMAed, and IosMh 1429 is set to the associated memory handle.

[0102] In the non-polling case, the server first RDMAs the read data. The server then can respond with an LWIO_MSG_STATUS_RESPONSE, or the server can send immediate data with the RDMA read data, in which case the immediate data is set to the ImmediateCookie value of the read request. FIG. 14B provides an illustrative representation of the LWIO_MSG_STATUS_RESPONSE 1433 returned by the server in the non-polling case in an embodiment of the invention.

[0103] Information 1437 is set to the number of bytes read. Status 1439 is set to 0, indicating success, or to another NTSTATUS, indicating failure. Cookie 1435 is set to the Cookie value set by the client in the header of the read message.

[0104] In the polling case, the server first RDMAs the read data. The server then RDMAs an LWIO_IO_STATUS_BLOCK to the client. FIG. 14D provides an illustrative representation of the LWIO_IO_STATUS_BLOCK 1441 returned by the server in an embodiment of the invention. Information 1443 is set to the number of bytes read. Status 1445 is set to 0. indicating success, or another NTSTATUS, indicating failure.

Write

[0105] The write message is used to place data into a particular Fid that was exchanged during the file registration. All write data is sent using ordinary send operations. The amount of data written is limited by the server MaxRecvBufferSize. If the client sends more data than this, the connection is terminated.

[0106] FIGS. 15A and 15C provide illustrative representations of the write message sent by the client in an embodiment of the invention, with FIG. 15A giving the non-polling case and FIG. 15C giving the polling case. The write message 1501 includes a common header 1503. Length 1505 is set to the number of bytes of data to be written. Opcode 1507 is set to LWIO_OPCODE_WRITE (0x1). Offset 1517 is set to the byte location at which to begin writing the file data. Flags 1509, 1529 is set to 0x0 in the polling case 1529 or LWIO_HDR_FLAG_INTERRUPT (0x80) in the non-polling case 1509. Marker 1513 is set to 0xFF. Credits 1511 is set to the number of pending I/O requests on the client. Fid 1515 is set to the File Id on which to issue the I/O. The data to be written 1527 immediately follows the common header 1503 of the write message.

[0107] In the non-polling case, Cookie 1525 is set to a client-specified value. In the polling case, IosVa 1533 is set to the location at which the server response status (LWIO_IO_STATUS_BLOCK) is RDMAed, and IosMh 1531 is set to the associated memory handle.

89

[0108] In the non-polling case, the server responds to the write message with an LWIO_MSG_STATUS_RESPONSE. FIG. 15B provides an illustrative representation of the LWIO_MSG_STATUS_RESPONSE 1535 returned by the server in an embodiment of the invention. Information 1539 is set to the number of bytes written. Status 1541 is set to 0, indicating success, or to another NTSTATUS, indicating failure. Cookie 1537 is set to the Cookie value set by the client in the header of the write message. In the polling case, the server RDMAs an LWIO_IO_STATUS_BLOCK. FIG. 15D provides an illustrative representation of the LWIO_IO_STATUS_BLOCK 1543 returned by the server in an embodiment of the invention. Information 1545 is set to the number of bytes written. Status 1547 is set to 0, indicating success, or to another NTSTATUS, indicating failure.

Vectored Read

[0109] The vectored read is used to obtain data from a particular Fid that was exchanged during the registration stage and to scatter the data on a page basis to multiple segments on the requester. All data read is sent to the requester by way of RDMA writes, with one RDMA write from the server for each read segment. The data read from disk is contiguous. The amount of data read is limited by the maximum number of destination pages that can be described in a single request. This limit is the server MaxRecvBufferSize divided by sizeof(LWIO_RDMA_REGION). The structure of LWIO_RDMA_REGION is given below.

[0110] FIGS. 16A and 16C provide illustrative representations of the vectored read message sent by the client in an embodiment of the invention, with FIG. 16A giving the non-polling case and FIG. 16C giving the polling case. The read message 1401 comprises a common header 1603 followed by one or more LWIO_RDMA_REGION segments 1605, 1607. In the header 1603, Length 1609 is set to the number of bytes of data to be read from the file. Opcode 1611 is set to LWIO_OPCODE_VEC_READ (0x2). Offset 1621 is set to the byte location at which to begin reading the file data. Flags 1613, 1631 is set to 0x0 in the polling case 1631, or LWIO_HDR_FLAG_INTERRUPT (0x80) in the non-polling case 1613. Marker 1617 is set to 0xFF. Credits 1615 is set to the number of pending I/O requests on the client. Fid 1619 is set to the File Id on which to issue the I/O. NumPages 1623 is set to the number of LWIO_RDMA_REGIONs that follow the common header 1603. PageSize 1625 is set to the local page size in bytes.

[0111] In the non-polling case, ImmediateCookie 1627 and Cookie 1629 are set to client-specified values. The server can complete the vectored read request in this case with an LWIO_MSG_STATUS_RESPONSE by way of a normal send, or with an RDMA with immediate data if the read is successful. The immediate data of the RDMA write is accordingly set to the ImmediateCookie 1627 value of the read request. In the polling case,

90

IosVa 1635 is set to the location at which the server response status (LWIO_IO_STATUS_BLOCK) is RDMAed, and IosMh 1633 is set to the associated memory handle.

[0112] The common header 1603 is immediately followed by a sufficient number of LWIO_RDMA_REGION segments 1605, 1607 to cover the length of the request. All intermediate segments must be one page in size. The final segment may be smaller than a page, but it must be a multiple of the backend disk sector size. In an embodiment of the invention, the LWIO_RDMA_REGION has the following format:

```
typedef volatile struct {
     PVOID64         DataVa;
     UINT32          DataMh;
     UINT32          Length;
} LWIO_RDMA_REGION;
```

The first LWIO_RDMA_REGION corresponds to the first PageSize bytes read, the second LWIO_RDMA_REGION corresponds to the second PageSize bytes read, and so on. DataVa 1637 is set to the location marking the beginning of the page in which the read data is to be placed. DataMh 1639 is set to the memory handle of the DataVa 1637. Length 1641 is set to the PageSize 1625 for all regions except for the final region, for which Length may be smaller but must be a multiple of the backend disk sector size.

[0113] In the non-polling case, the server first RDMAs the read data. The server then can respond with an LWIO_MSG_STATUS_RESPONSE, or the server can send immediate data with the RDMA read data, in which case the immediate data is set to the ImmediateCookie value of the read request. FIG. 16B provides an illustrative representation of the LWIO_MSG_STATUS_RESPONSE 1643 returned by the server in the non-polling case in an embodiment of the invention. Information 1647 is set to the number of bytes read. Status 1649 is set to 0, indicating success, or to another NTSTATUS, indicating failure. Cookie 1645 is set to the Cookie value set by the client in the header of the vectored read message.

[0114] In the polling case, first the server RDMAs the read data, and then the server RDMAs an LWIO_IO_STATUS_BLOCK. FIG. 16D provides an illustrative representation of the LWIO_IO_STATUS_BLOCK 1651 returned by the server in an embodiment of the invention. Information 1653 is set to the number of bytes read. Status 1655 is set to 0. indicating success, or another NTSTATUS. indicating failure.

Vectored Write

[0115] The vectored write message is used to perform a gather write into a particular Fid that was exchanged during the file registration. All write data is sent using ordinary

91

send operations. The amount of data written is limited by the server MaxRecvBufferSize. If the client sends more data than this, the connection is terminated.

[0116] FIGS. 17A, 17B and 17C provide illustrative representations of the vectored write message sent by the client in an embodiment of the invention, with FIG. 17A illustrating the non-polling, non-collapse case, FIG. 17B illustrating the non-polling, collapse case, and FIG. 17C illustrating the polling, collapse case.

[0117] The write message 1701 includes a common header 1703, immediately followed by the data to be written 1705. In the common header 1703, Length 1707 is set to the number of bytes of data being written. Opcode 1709 is set to LWIO_OPCODE_WRITE (0x3). Offset 1719 is set to the byte location at which to begin writing the file data. Marker 1715 is set to 0xFF. Credits 1713 is set to the number of pending I/O requests on the client. Fid 1717 is set to the File Id on which to issue the I/O.

[0118] Flags 1711, 1721, 1727 is set to 0x0, signifying polling 1727, or else to LWIO_HDR_FLAG_INTERRUPT (0x80) 1711. In the latter case, flags can also include LWIO_HDR_FLAG_COLLAPSE 1721 to indicate that all pages in the write contain the same data, so that only a single page of data has been sent. This is an optimization intended to minimize the transfer of redundant data. LWIO_HDR_FLAG_COLLAPSE can only be used if the registered file flags include FILE_NO_INTERMEDIATE_BUFFERING (0x8) and the PageSizes exchanged during the authentication stage are even multiples of each other. In the case of a collapsed I/O, NumPages 1723 is set to the number of pages of data spanned by the I/O. The last page may be partial due to the Length parameter. PageSize 1725 is set to the local page size in bytes. In the polling case, IosVa 1731 is set to the location at which the server response status (LWIO_IO_STATUS_BLOCK) is to be RDMAed. IosMh 1729 is the associated memory handle.

[0119] In the non-polling case, for both non-collapsed and collapsed I/O, the server responds to the write message with an LWIO_MSG_STATUS_RESPONSE.

[0120] FIG. 17D provides an illustrative representation of the LWIO_MSG_STATUS_RESPONSE 1733 returned by the server in an embodiment of the invention. Information 1737 is set to the number of bytes written. Status 1739 is set to 0, indicating success, or to another NTSTATUS, indicating failure. Cookie 1735 is set to the Cookie value set by the client in the header of the write message.

[0121] In the polling case, for both non-collapsed and collapsed I/O, the server RDMAs an LWIO_IO_STATUS_BLOCK. FIG. 17E provides an illustrative representation of the LWIO_IO_STATUS_BLOCK 1741 returned by the server in an embodiment of the invention. Information 1743 is set to the number of bytes written. Status 1745 is set to 0, indicating success, or to another NTSTATUS, indicating failure.

92

93

Conclusion

[0122] While illustrative embodiments of the invention have been illustrated and described, it will be appreciated that various changes can be made without departing from the invention. Similarly, any process steps described herein may be interchangeable with other steps in order to achieve the same result. In addition, the illustrative examples described above are not intended to be exhaustive or to limit the invention to the precise forms disclosed. On the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

98

WHAT IS CLAIMED IS:

1. A system for offloading an input/output (I/O) task from a first computer to a second computer, comprising:

a client running on the first computer;

a server running on the second computer; and

at least one RDMA channel linking the first computer and the second computer, wherein the first computer and the second computer communicate in accordance with a protocol comprising a network discovery phase and an I/O processing phase.

2. The system of claim 1 wherein, in the I/O processing phase, read operations are implemented using RDMA and write operations are implemented using send operations.

3. The system of claim 1 wherein the protocol is used in association with a second network protocol.

4. The system of claim 3 wherein the second protocol is SMB.

5. The system of claim 3 wherein the second protocol is CIFS.

6. A computer-readable medium storing computer-executable instructions and computer-readable data comprising a computer program product for use in a system for offloading an input/output (I/O) task from a first computer to a second computer, the system comprising:

at least one RDMA channel linking the first computer and the second computer, wherein the first computer and the second computer communicate in accordance with a protocol comprising a network discovery phase and an I/O processing phase.

7. A method for offloading an input/output (I/O) task from a first computer to a second computer, comprising:

discovering, by a client on the first computer and a server on the second computer, one or more shared RDMA-capable providers; and

posting, by the client, an I/O processing request for completion by the server on the second computer.

8. The method of claim 7 wherein the discovering one or more shared RDMA-capable providers further comprises:

94

obtaining, by the client, a server request resume key from the server;

opening, by the client, a pipe to the server;

sending, by the client over the pipe, a negotiate request; and

sending, by the server over the pipe, a negotiate response including a minimal list of common providers.

9. The method of claim 7, further comprising:

creating, by the client, an RDMA connection to the server over a shared RDMA-capable provider; and

authenticating, by the client and the server, the RDMA connection.

10. The method of claim 9, further comprising:

registering, by the client, one or more files for use with the server over the RDMA connection.

11. The method of claim 10 wherein the registering one or more files comprises:

sending, by the client to the server, a register file message; and

sending, by the server to the client, a register file completion message.

12. The method of claim 9 wherein the authenticating the RDMA connection further comprises:

sending, by the client, an authenticate request message to the server, the authenticate request message including a key;

if the key matches a previous key sent by the server to the client, sending, by the server, an authenticate response message to the client.

13. The method of claim 12 wherein the previous key is a key contained in a negotiate response message sent by the server to the client.

14. The method of claim 12, further comprising:

sending, by the server to the client, a status response message to complete the authenticating.

15. The method of claim 7 wherein the posting the I/O processing request comprises sending, by the client, one of (a) a close request, (b) a cancel request, (c) a read request, (d) a write request, (e) a vectored read request, and (f) a vectored write request.

98

16. The method of claim 15, further comprising:

completing, by the server, the read request and the vectored read request by sending data using RDMA write operations; and

completing, by the server, the write request and the vectored write request by sending data using normal send operations.

17. The method of claim 15 wherein the vectored write request includes a collapse flag in a header of the request.

18. The method of claim 7 wherein posting the I/O processing request further includes indicating whether the completion by the server should be in polling mode.

19. The method of claim 18 wherein the indicating whether the completion should be in polling mode comprises indicating that the completion should not be in polling mode by setting an interrupt flag in a header of the I/O processing request.

20. The method of claim 18, further comprising:

if the client indicates that the completion should not be in polling mode, completing, by the server, the I/O processing request by sending a status block to the first computer by way of RDMA transfer.

21. The method of claim 18; further comprising:

if the client indicates that the completion should be in polling mode, and the client has sent an interrupt request message to the server, sending, by the server to the client, an interrupt response message by way of an ordinary send.

22. The method of claim 7 wherein posting the I/O processing request further includes specifying a number of credits in a header of the request.

23. Computer-readable media storing computer-executable instructions for implementing a method for offloading an input/output (I/O) task from a first computer to a second computer, the method comprising:

discovering, by a client on the first computer and a server on the second computer, one or more shared RDMA-capable providers; and

posting, by the client, an I/O processing request for completion by the server on the second computer.

96

24. A method for managing buffers in an input/output offload protocol, comprising:

sending, by a server to a client, a delta credit message including an information field set to a number of credits, wherein, if the number is a negative number -N, the server requires the client to retire N credits;

if the number of credits is a negative number -N, sending, by the client to the server, N credit messages, and otherwise sending, by the client to the server, one credit message; and

for each credit message sent by the client, sending, by the server to the client, a status response message.

97

ABSTRACT

A method and system for offloading I/O processing from a first computer to a second computer, using RDMA-capable network interconnects, are disclosed. The method and system include a client on the first computer communicating over an RDMA connection to a server on the second computer by way of a lightweight input/output (LWIO) protocol. The protocol generally comprises a network discovery phase followed by an I/O processing phase. During the discovery phase, the client and server determine a minimal list of shared RDMA-capable providers. During the I/O processing phase, the client posts I/O requests for offloading to the second machine over a mutually-authenticated RDMA channel. The I/O model is asymmetric, with read operations being implemented using RDMA and write operations being implemented using normal sends. Read and write requests may be completed in polling mode and in interrupt mode. Buffers are managed by way of a credit mechanism.

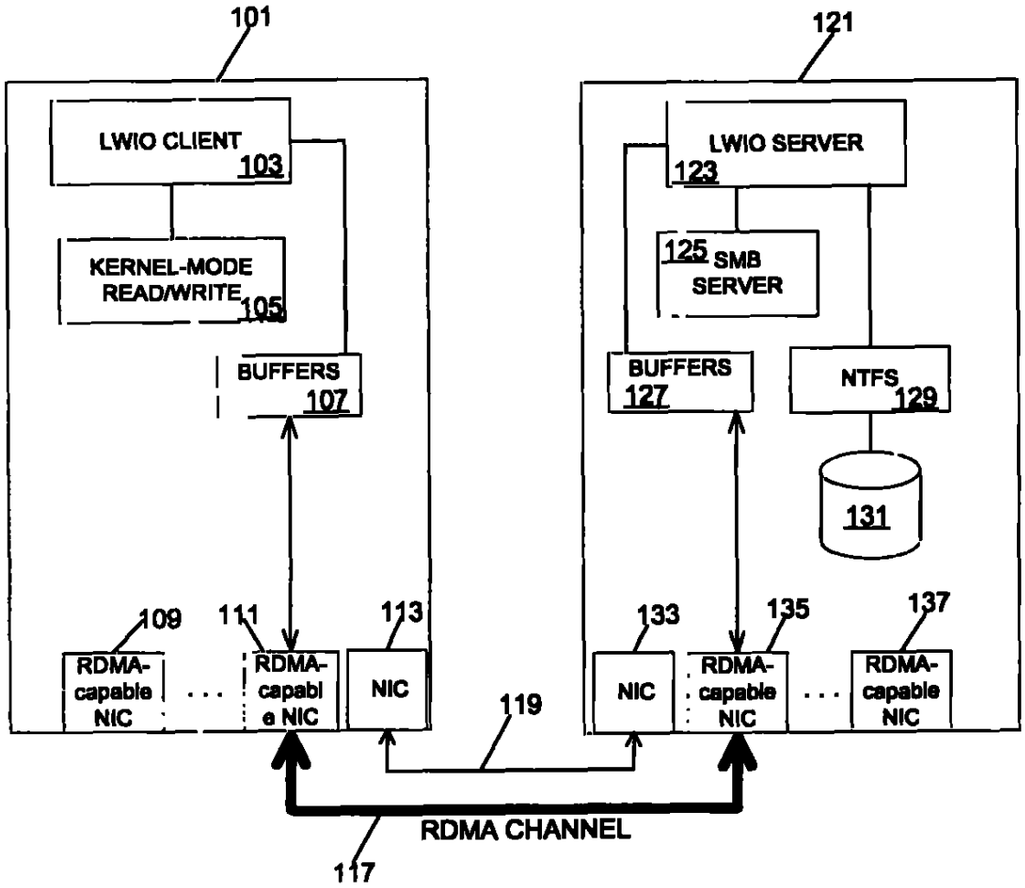

101
121
LWIO CLIENT
103
KERNEL-MODE
READ/WRITE
105
BUFFERS
107
LWIO SERVER
123
125 SMB
SERVER
BUFFERS
127
NTFS
129
131
109
111
113
133
135
137
RDMA-
capable
NIC
RDMA-
capabl
e NIC
NIC
NIC
RDMA-
capable
NIC
RDMA-
capable
NIC
119
RDMA CHANNEL
117

FIG. 1

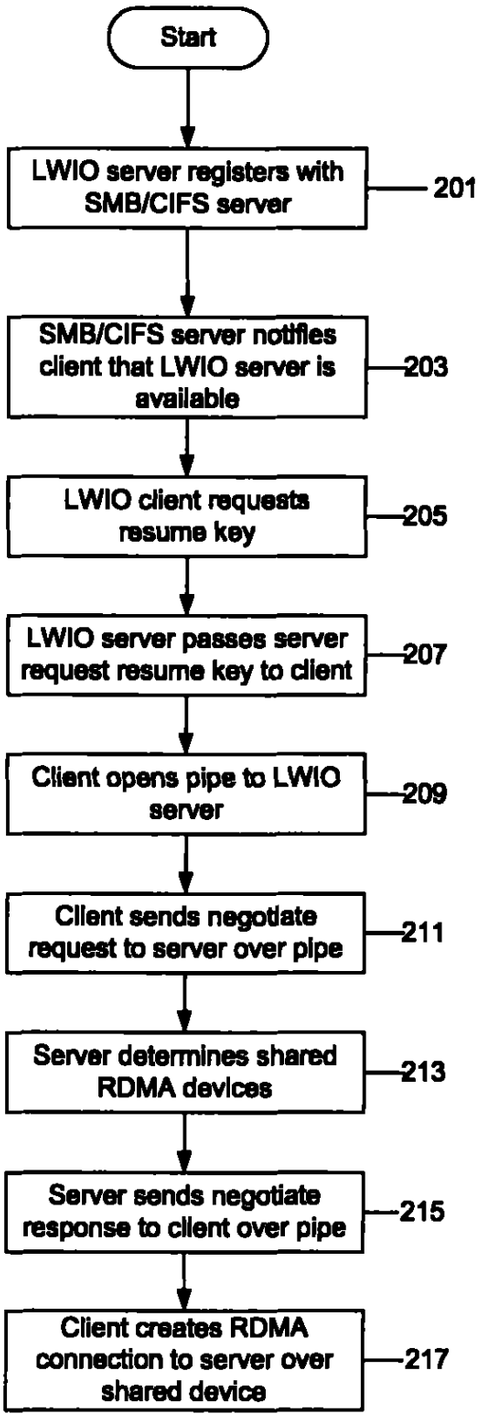
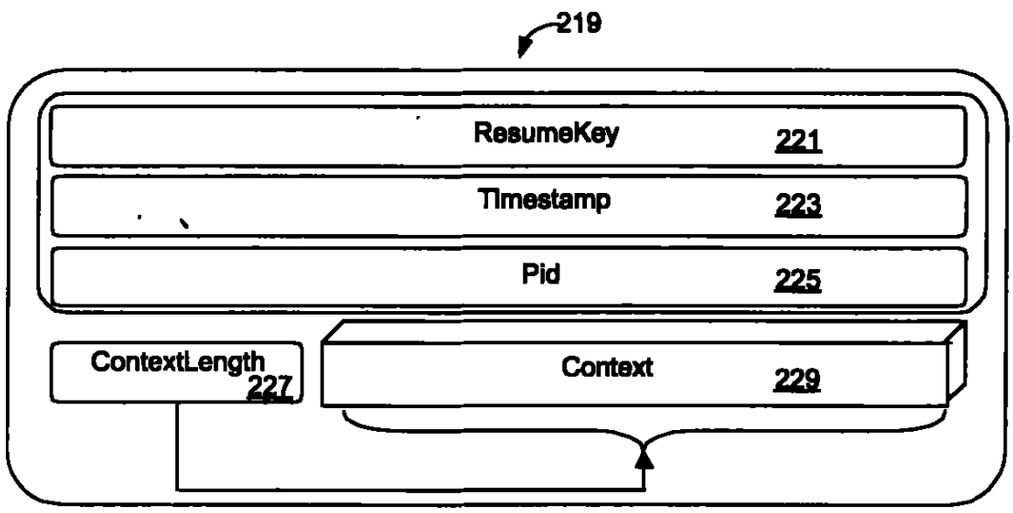
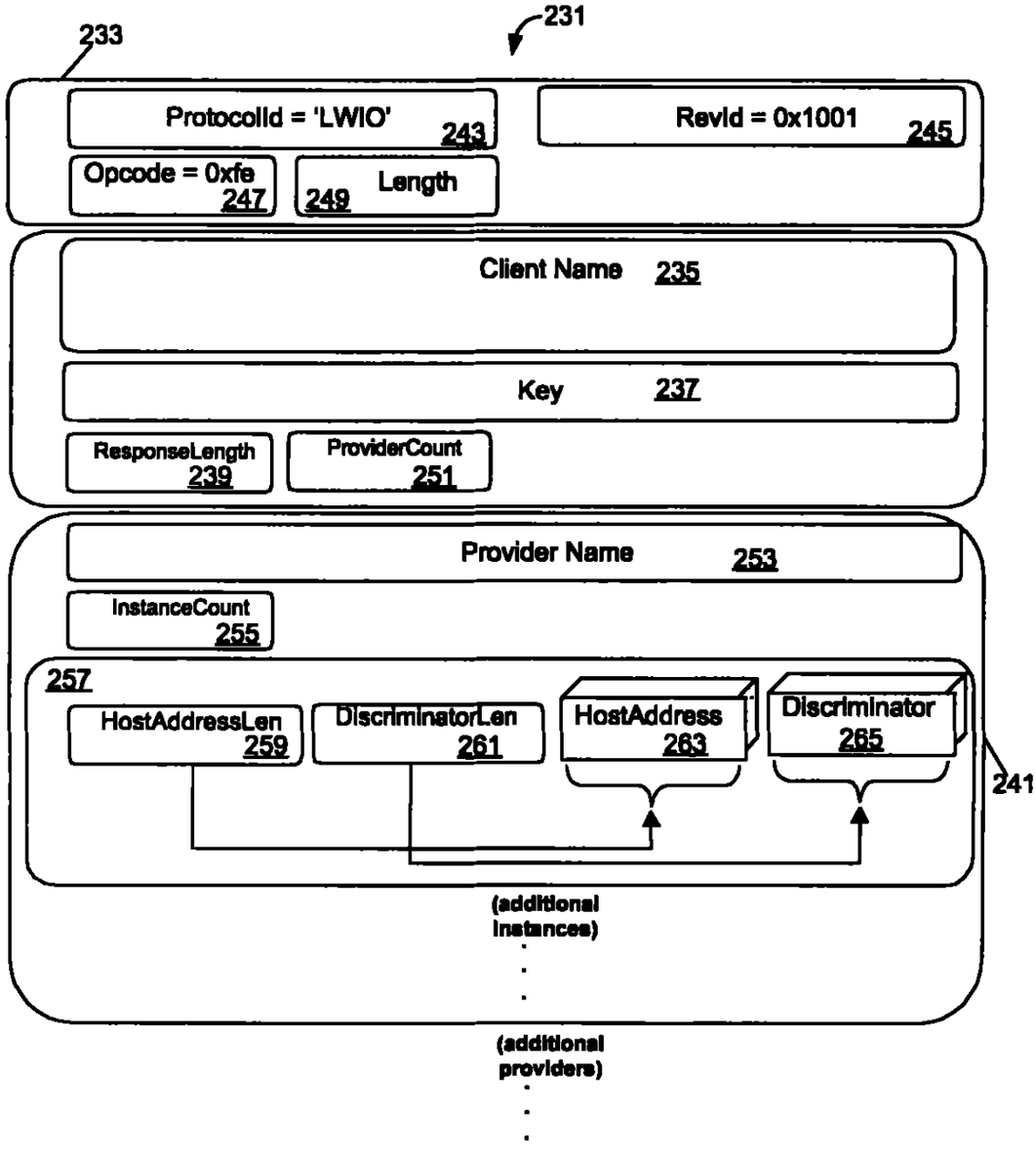

Start
LWIO server registers with SMB/CIFS server
201
SMB/CIFS server notifies client that LWIO server is available
203
LWIO client requests resume key
205
LWIO server passes server request resume key to client
207
Client opens pipe to LWIO server
209
Client sends negotiate request to server over pipe
211
Server determines shared RDMA devices
213
Server sends negotiate response to client over pipe
215
Client creates RDMA connection to server over shared device
217

FIG. 2



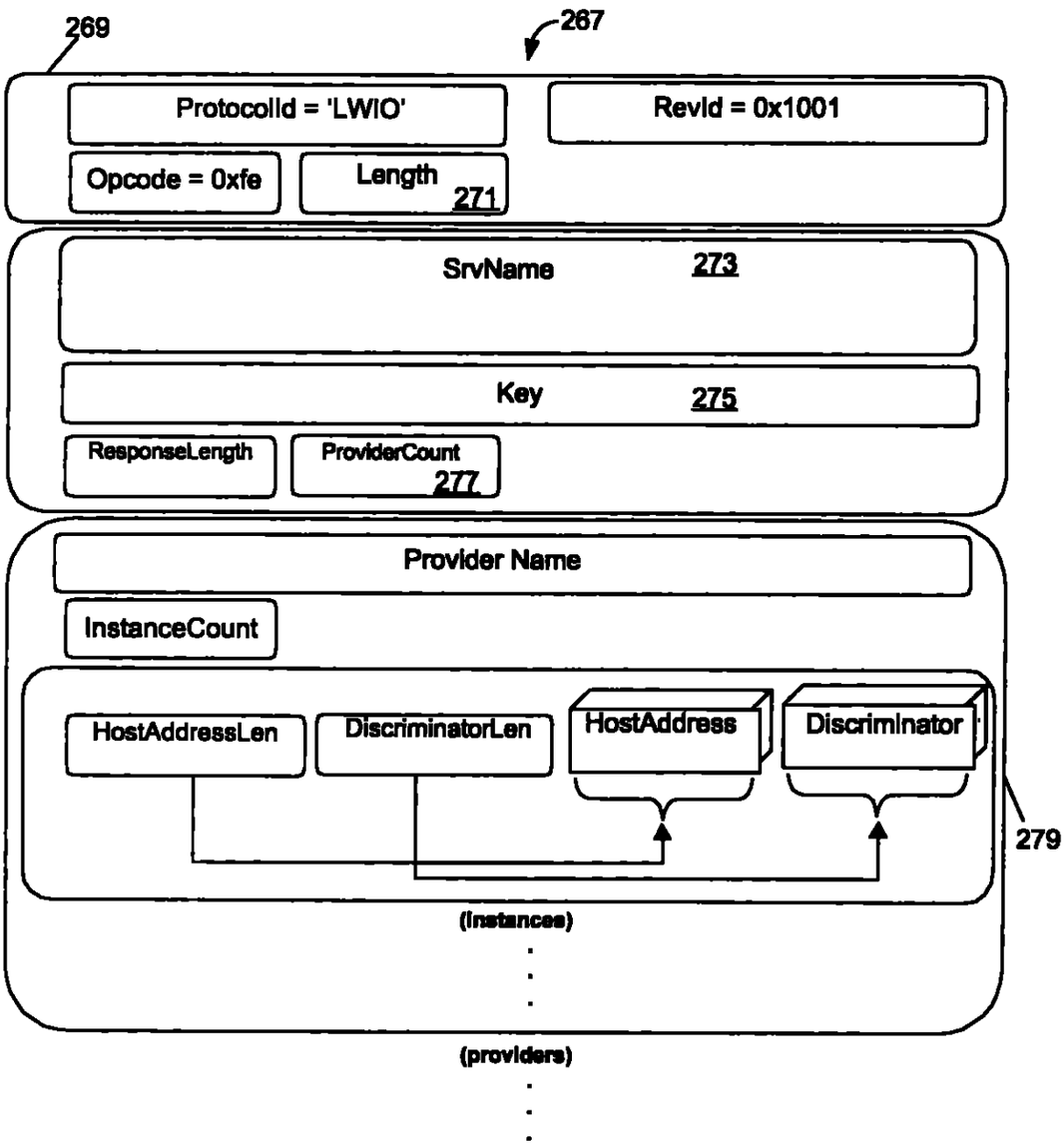
SERVER REQUEST RESUME KEY

FIG. 3



NEGOTIATE REQUEST

FIG. 4A



NEGOTIATE RESPONSE

FIG. 4B

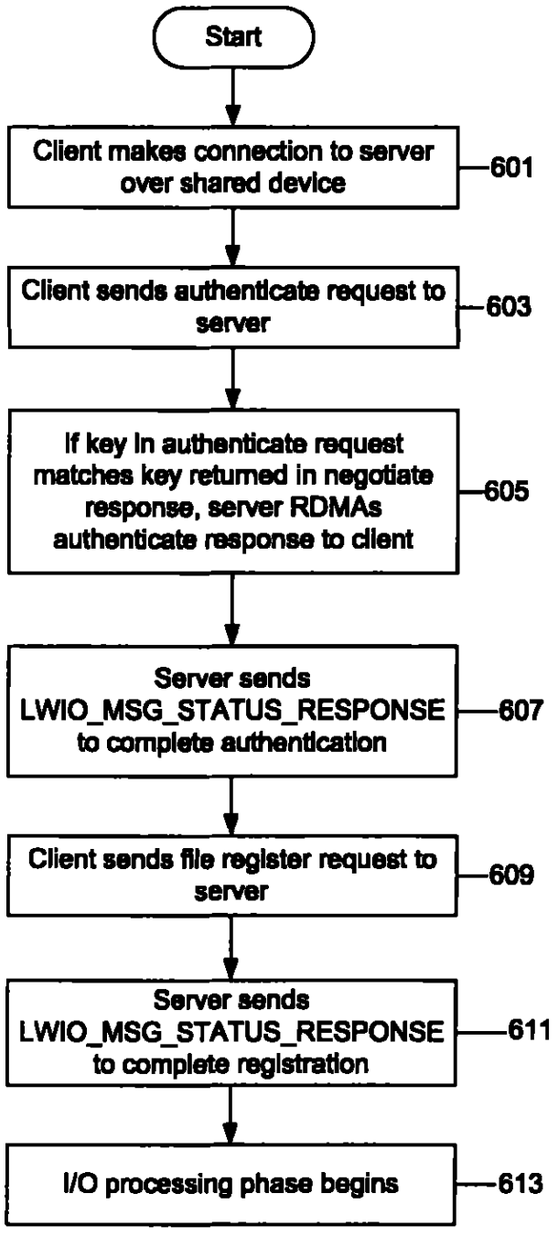
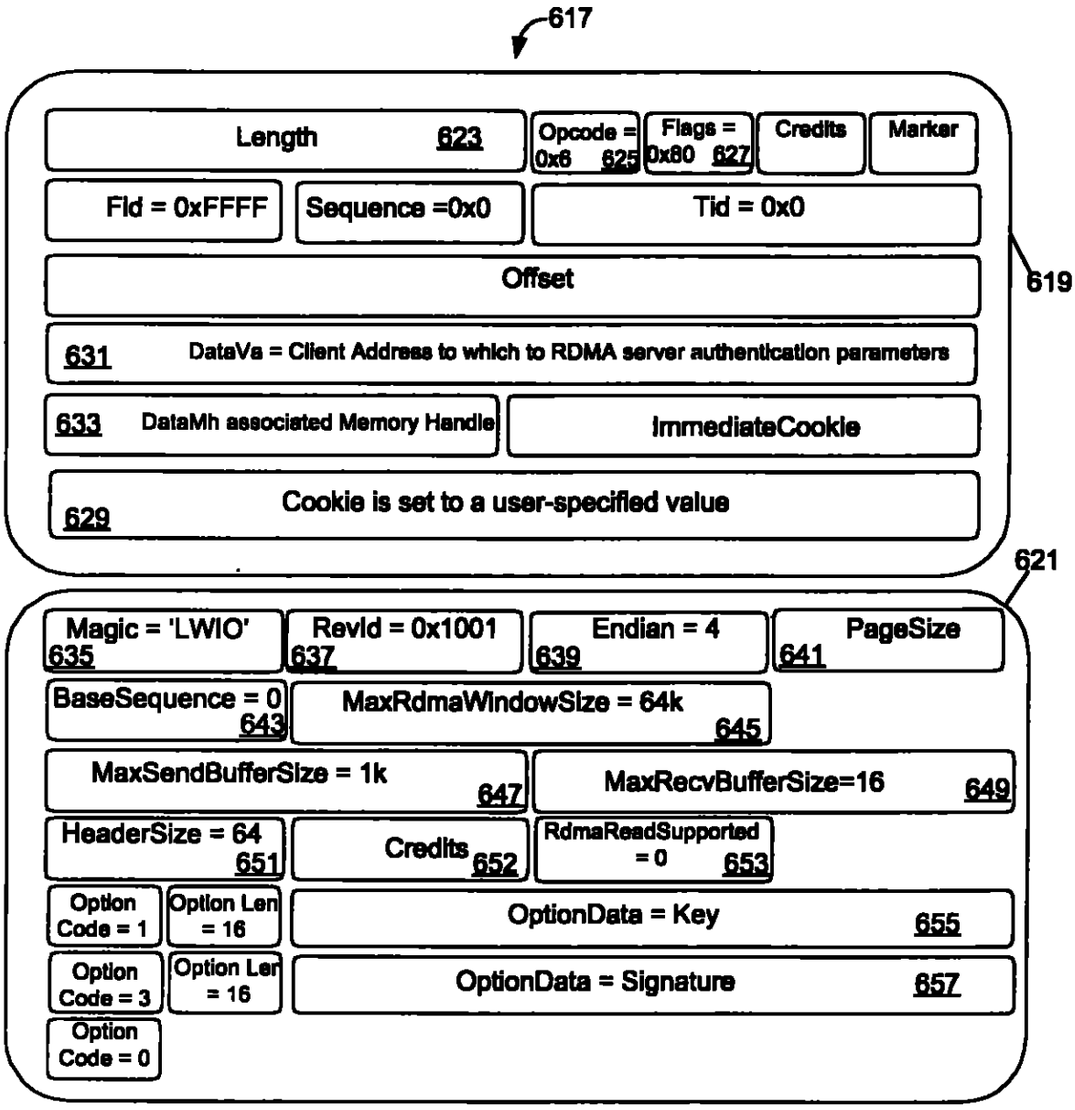

Start
Client makes connection to server over shared device
601
Client sends authenticate request to server
603
If key in authenticate request matches key returned in negotiate response, server RDMAs authenticate response to client
605
Server sends LWIO_MSG_STATUS_RESPONSE to complete authentication
607
Client sends file register request to server
609
Server sends LWIO_MSG_STATUS_RESPONSE to complete registration
611
I/O processing phase begins
613

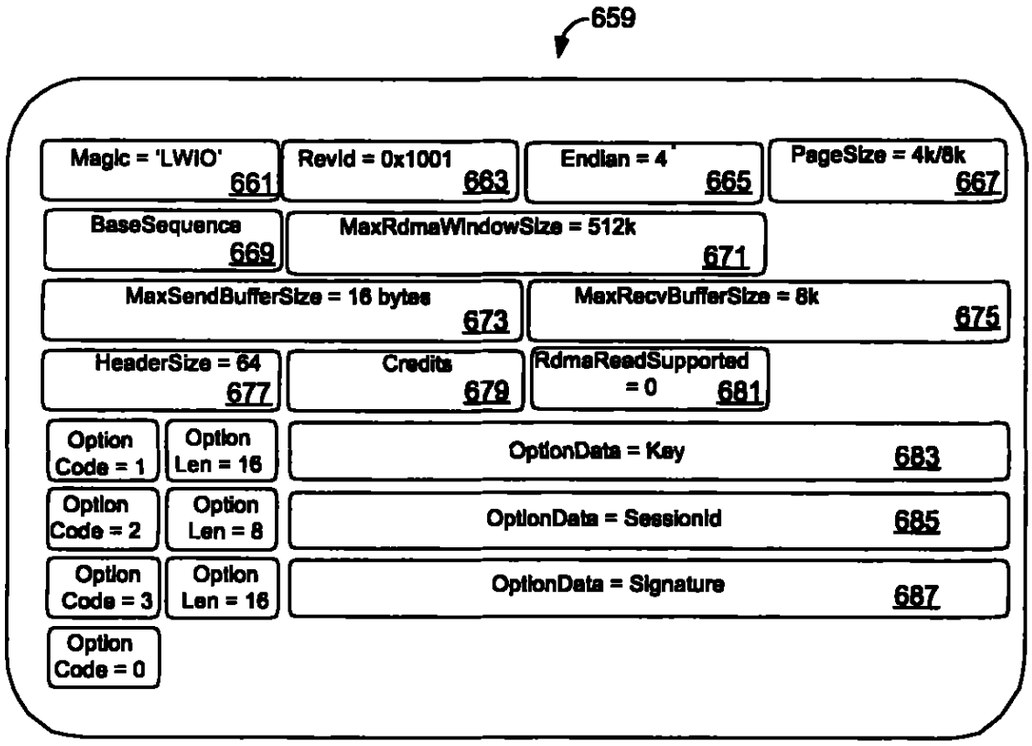
FIG. 5



AUTHENTICATE

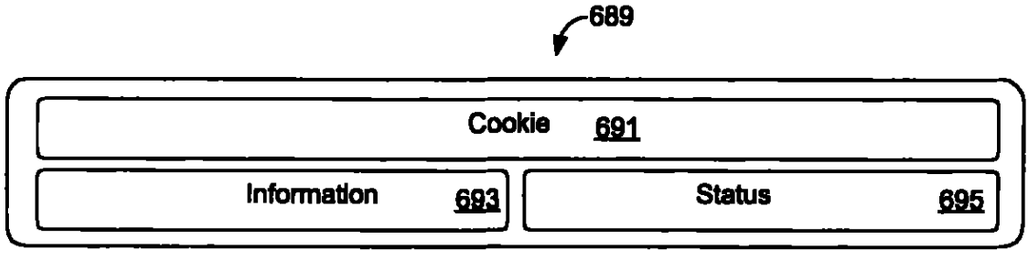
FIG. 6A

105



AUTHENTICATE RESPONSE

FIG. 6B



SERVER AUTHENTICATE COMPLETION

FIG. 6C

106

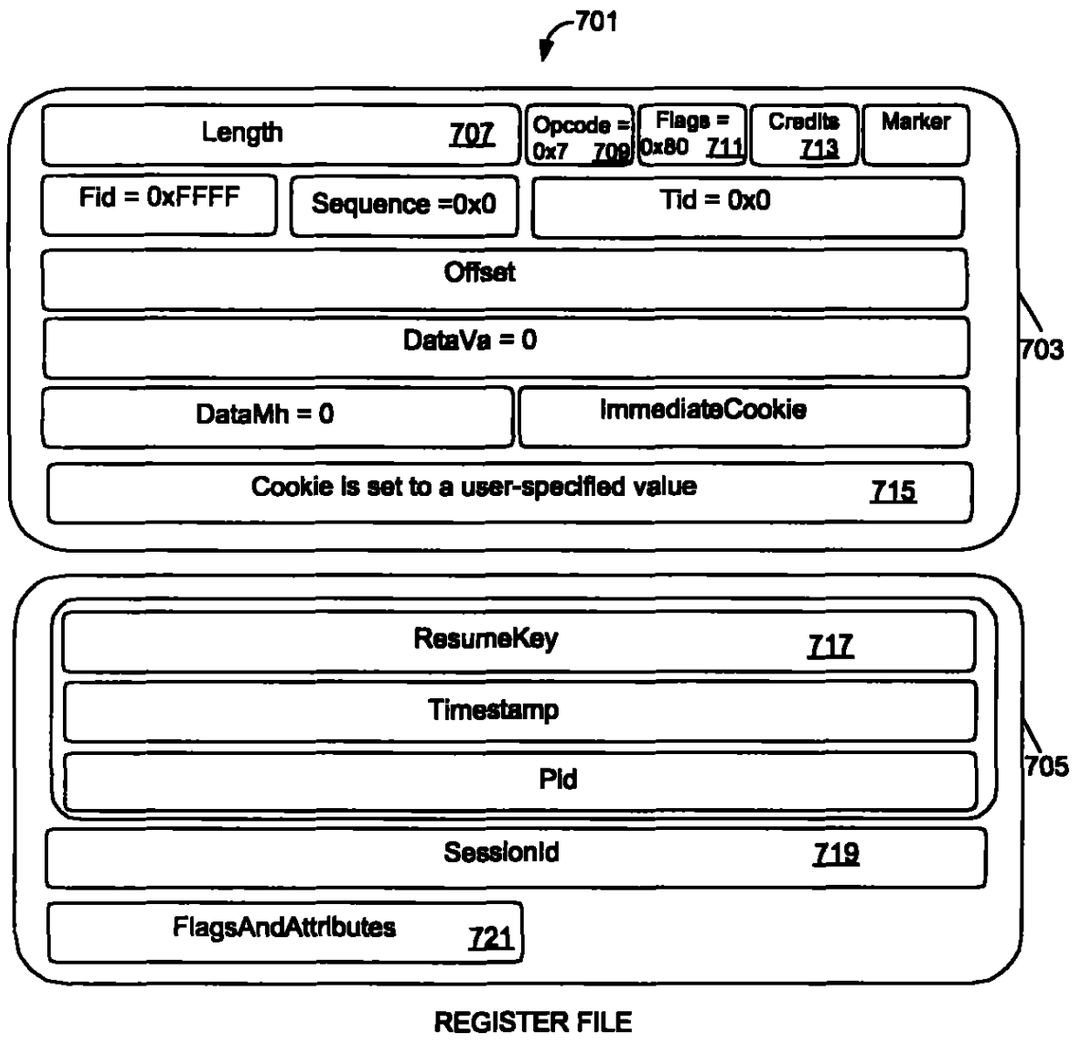

701
Length 707
Opcode = 0x7 709
Flags = 0x80 711
Credits 713
Marker
Fid = 0xFFFF
Sequence =0x0
Tid = 0x0
Offset
DataVa = 0
DataMh = 0
ImmediateCookie
Cookie is set to a user-specified value 715
703
ResumeKey 717
Timestamp
Pid
705
SessionId 719
FlagsAndAttributes 721
REGISTER FILE

FIG. 7A

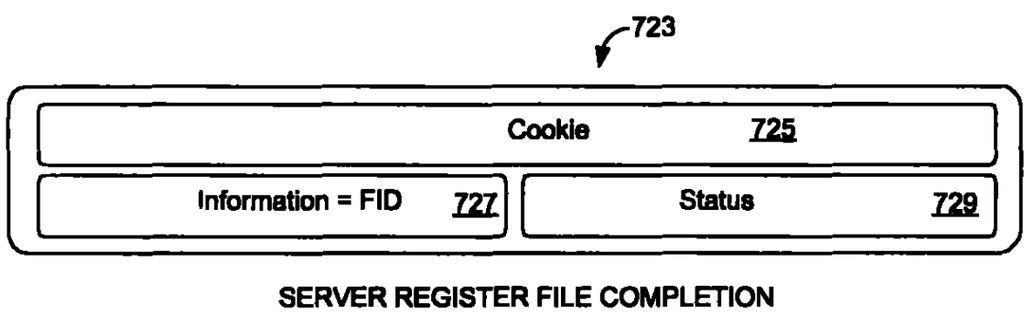

723
Cookie 725
Information = FID 727
Status 729
SERVER REGISTER FILE COMPLETION

FIG. 7B

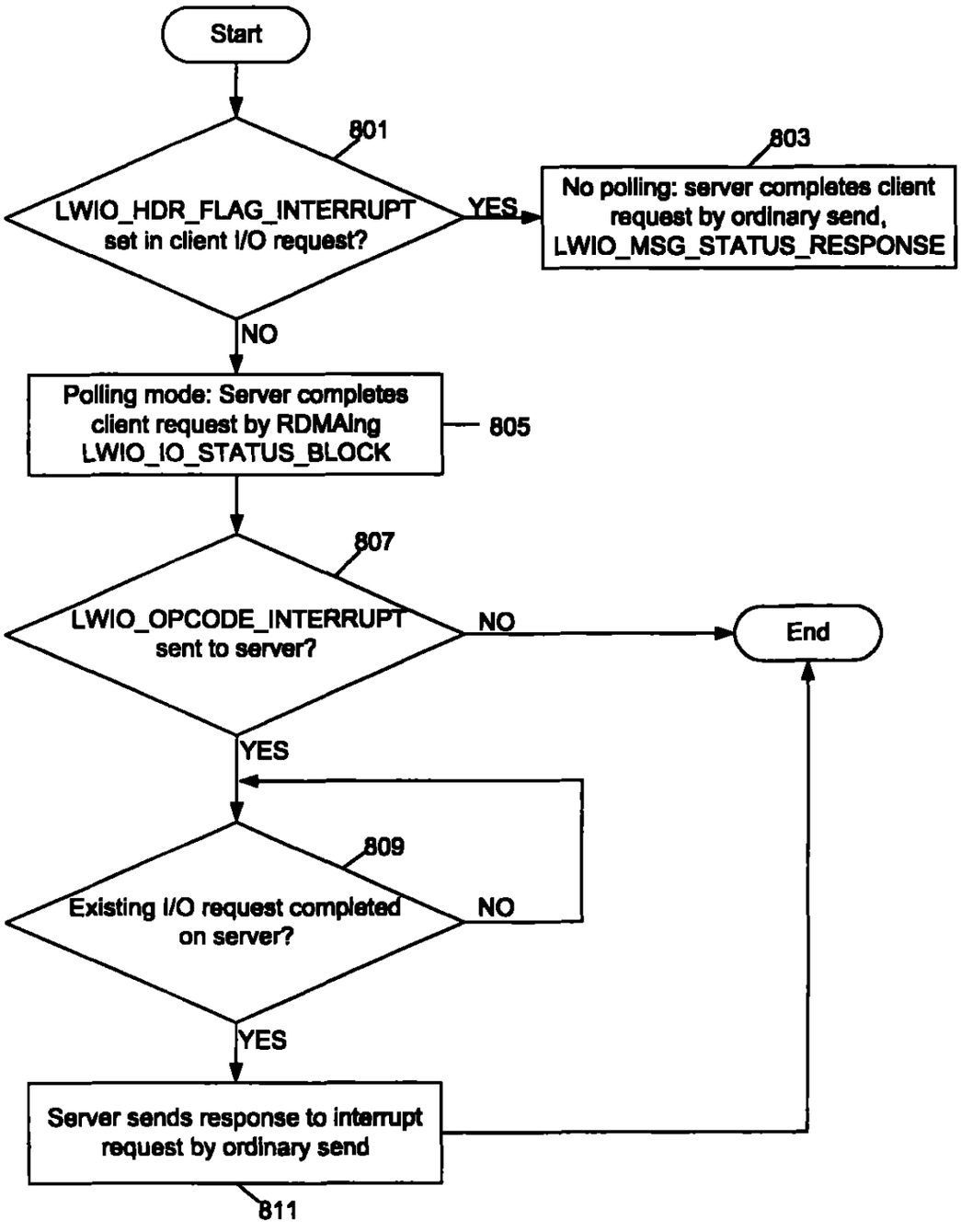

Start
801
LWIO_HDR_FLAG_INTERRUPT set in client I/O request?
YES
803
No polling: server completes client request by ordinary send, LWIO_MSG_STATUS_RESPONSE
NO
Polling mode: Server completes client request by RDMAing LWIO_IO_STATUS_BLOCK
805
807
LWIO_OPCODE_INTERRUPT sent to server?
NO
End
YES
809
Existing I/O request completed on server?
NO
YES
Server sends response to interrupt request by ordinary send
811

FIG. 8

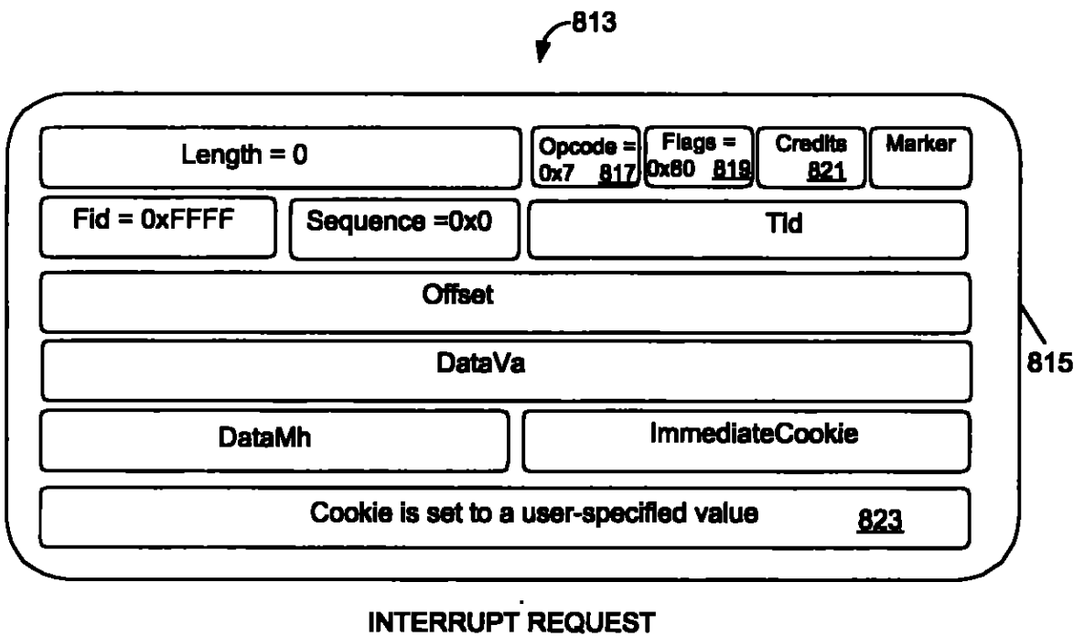

813
Length = 0
Opcode = 0x7 817
Flags = 0x80 819
Credits 821
Marker
Fid = 0xFFFF
Sequence =0x0
Tid
Offset
DataVa
DataMh
ImmediateCookie
Cookie is set to a user-specified value 823
815
INTERRUPT REQUEST

FIG. 9A

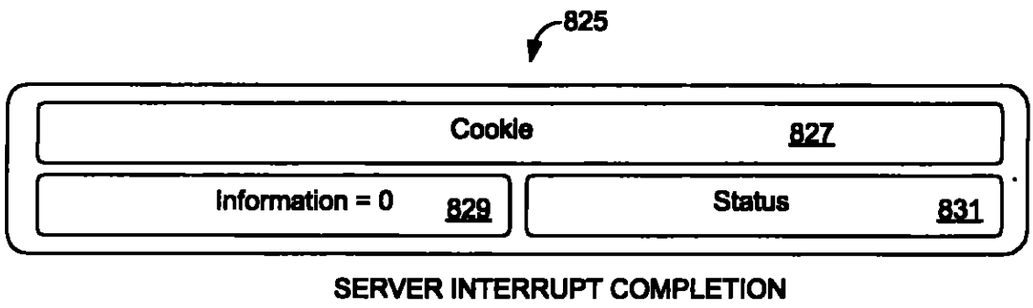

825
Cookie 827
Information = 0 829
Status 831
SERVER INTERRUPT COMPLETION

FIG. 9B

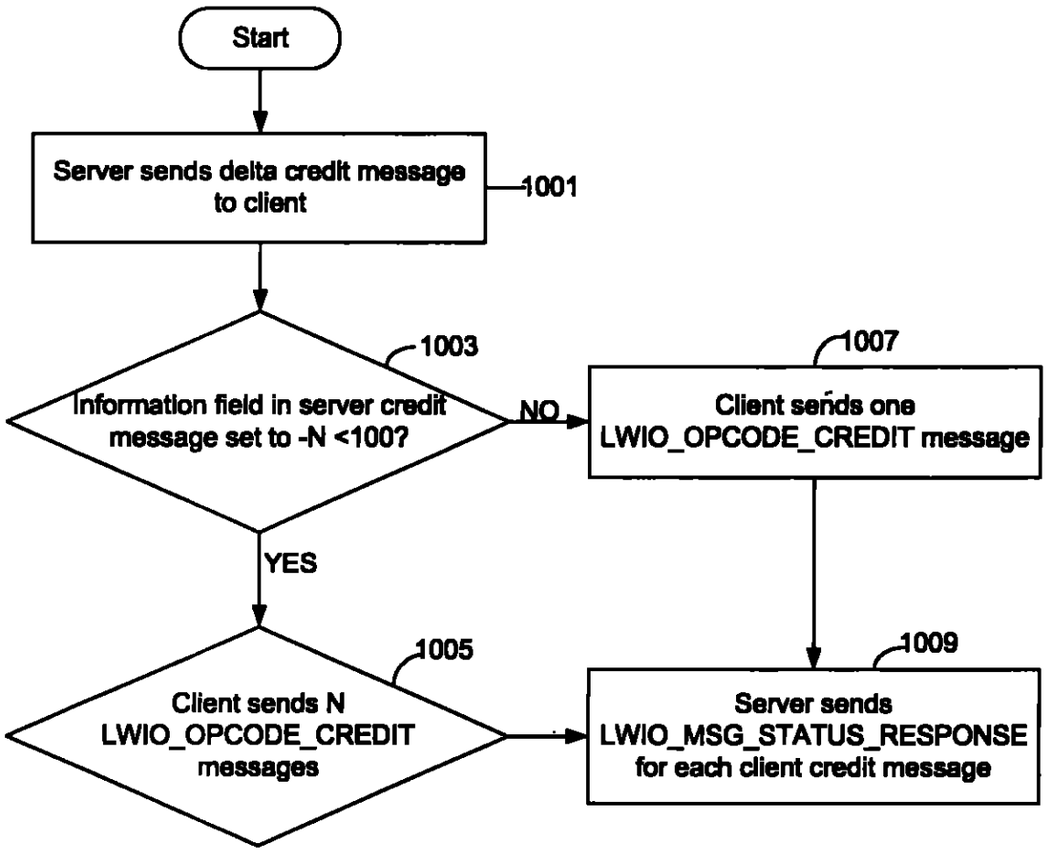
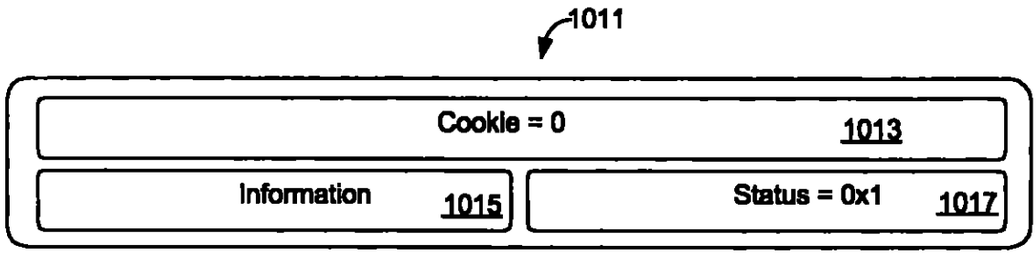
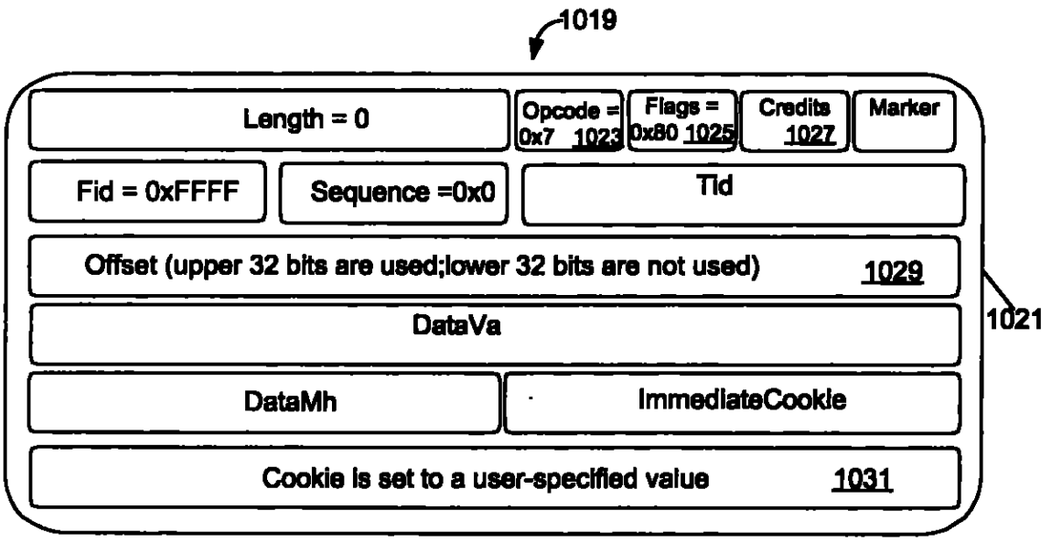

Start
Server sends delta credit message to client
1001
1003
Information field in server credit message set to -N <100?
NO
1007
Client sends one LWIO_OPCODE_CREDIT message
YES
1005
Client sends N LWIO_OPCODE_CREDIT messages
1009
Server sends LWIO_MSG_STATUS_RESPONSE for each client credit message

FIG. 10



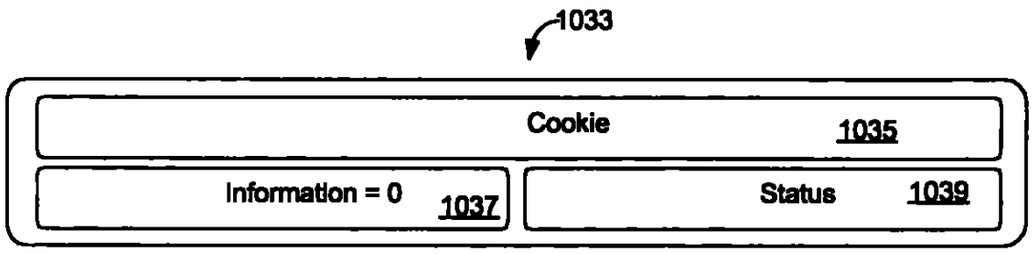
SERVER-TO-CLIENT CREDIT MESSAGE

FIG. 11A



CLIENT-TO-SERVER CREDIT MESSAGE

FIG. 11B



SERVER CREDIT COMPLETION

FIG. 11C

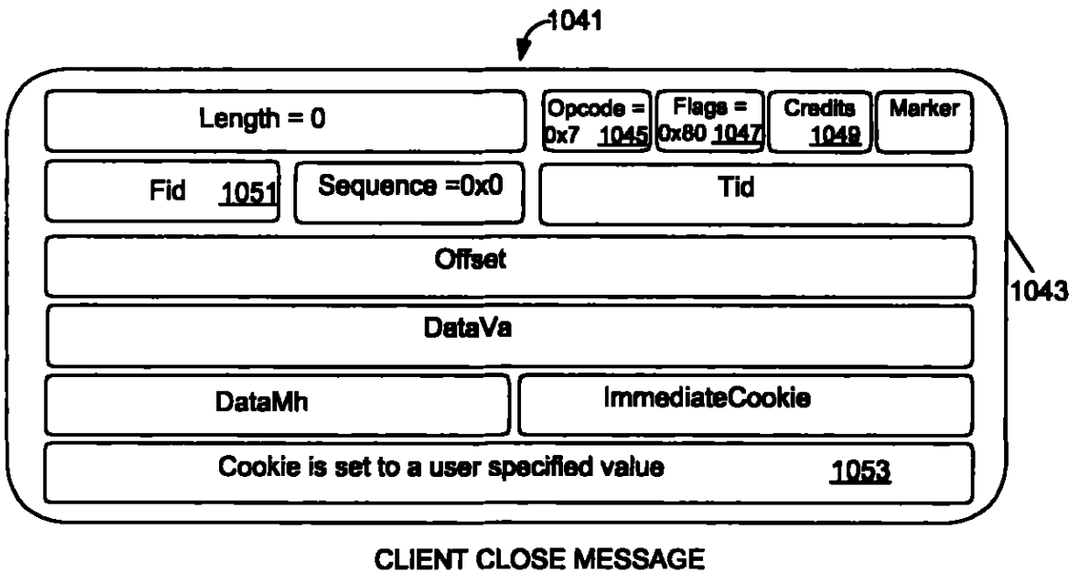


FIG. 12A

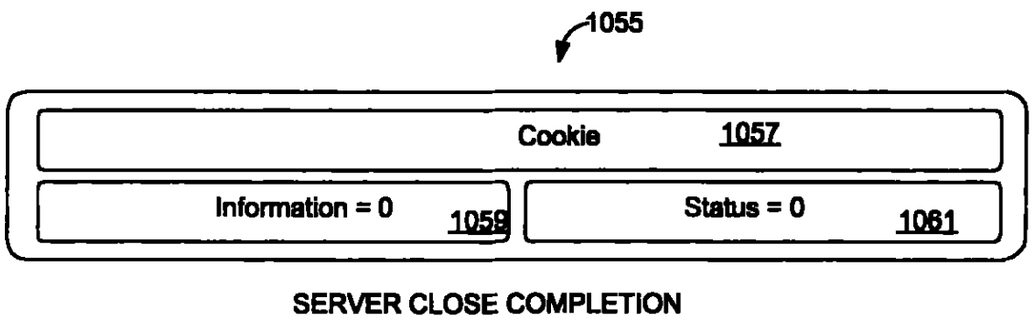
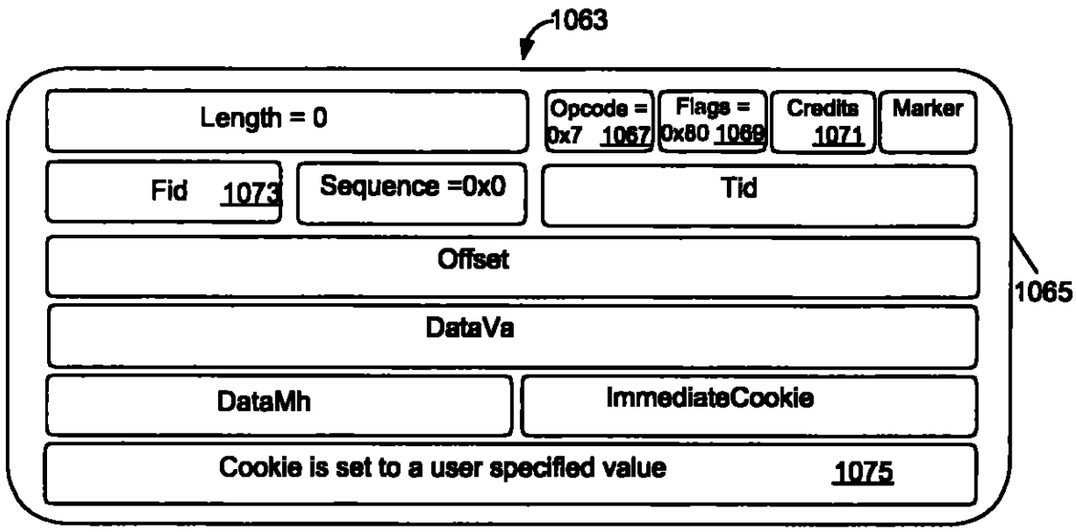
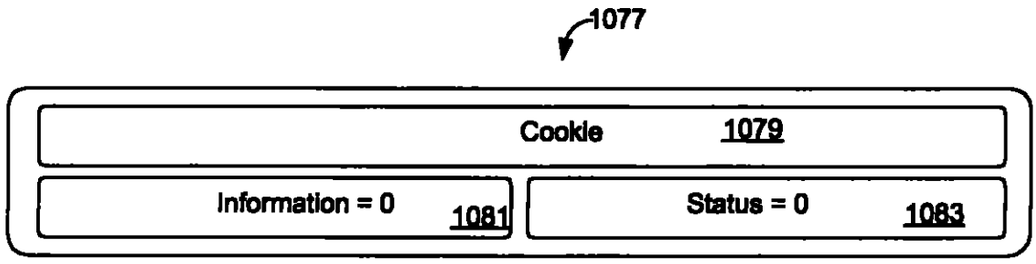


FIG. 12B



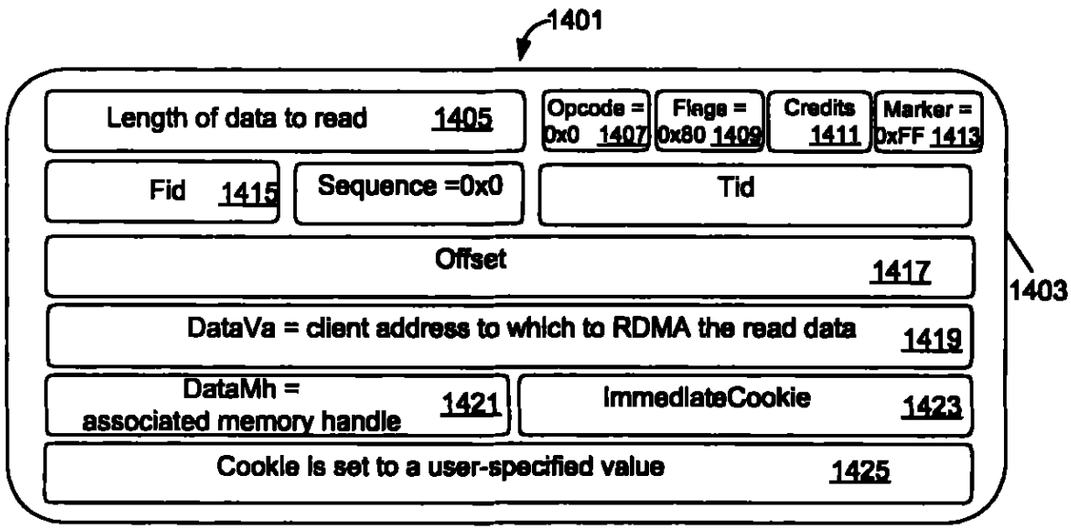
CLIENT CANCEL MESSAGE

FIG. 13A



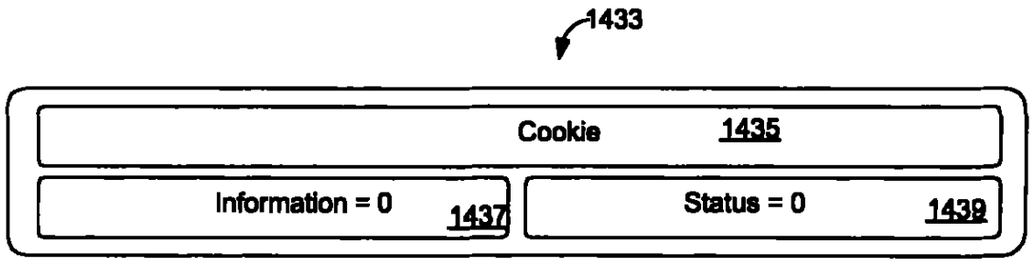
SERVER CANCEL COMPLETION

FIG. 13B



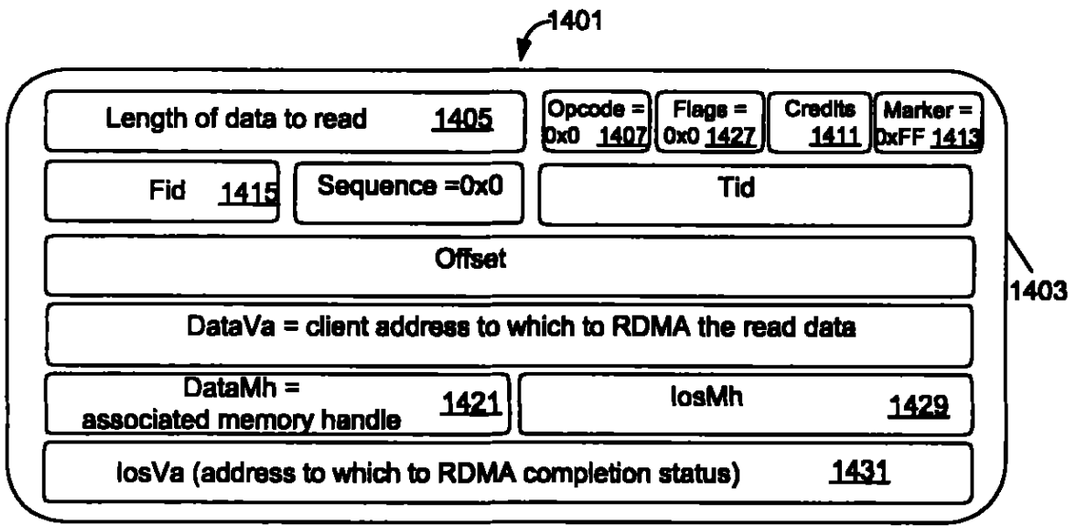
CLIENT READ MESSAGE (NO POLLING)

FIG. 14A



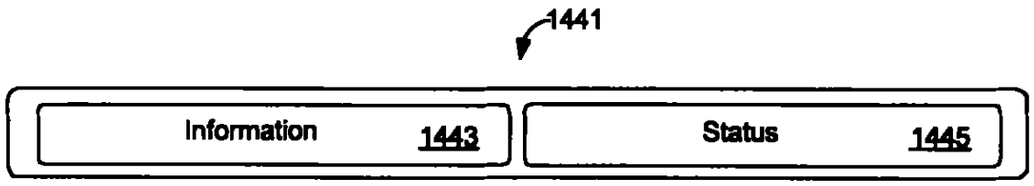
READ COMPLETION (NO POLLING)

FIG. 14B



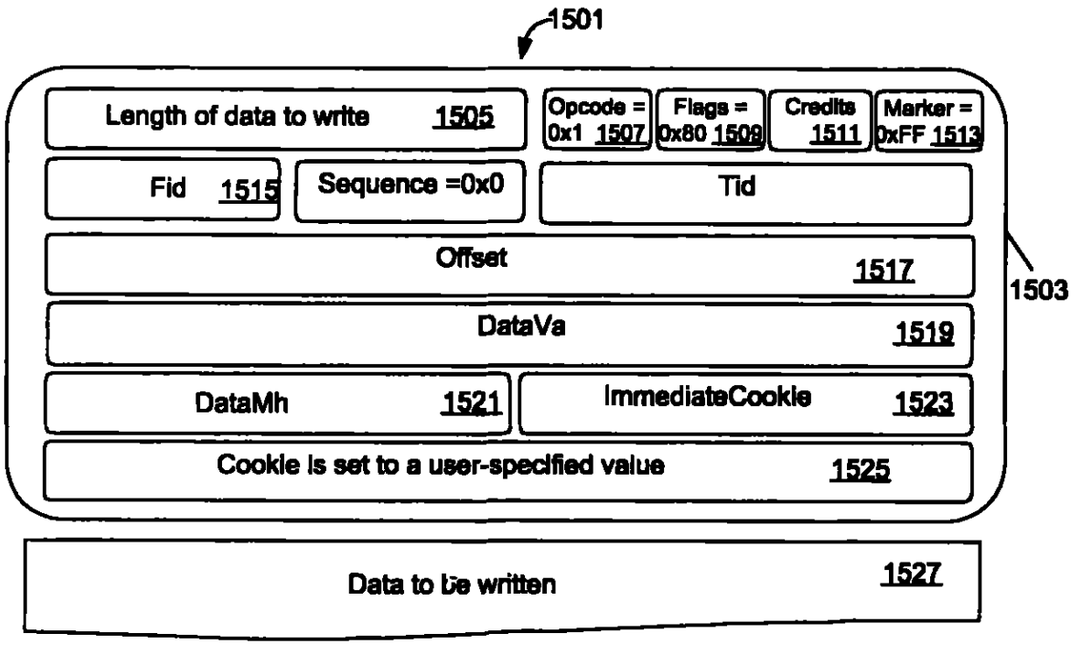
CLIENT READ MESSAGE (POLLING)

FIG. 14C



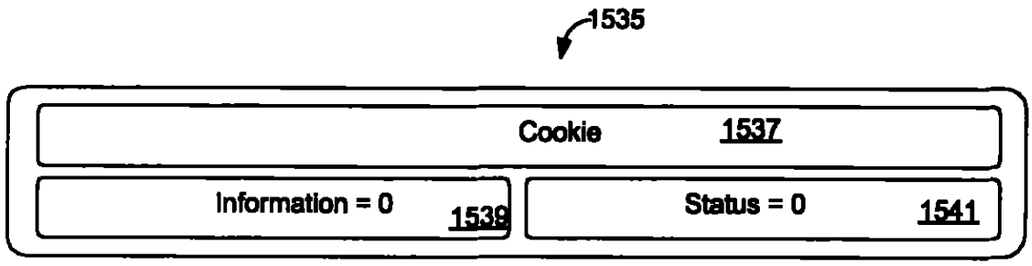
READ COMPLETION (POLLING)

FIG. 14D



CLIENT WRITE MESSAGE (NO POLLING)

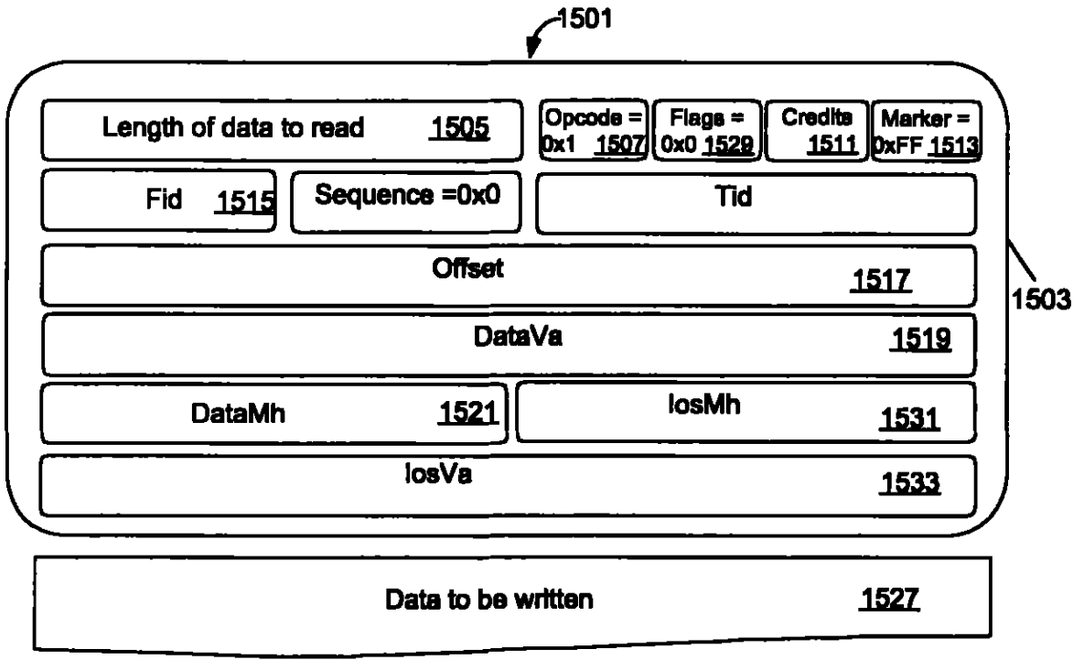
FIG. 15A



WRITE COMPLETION (NO POLLING)

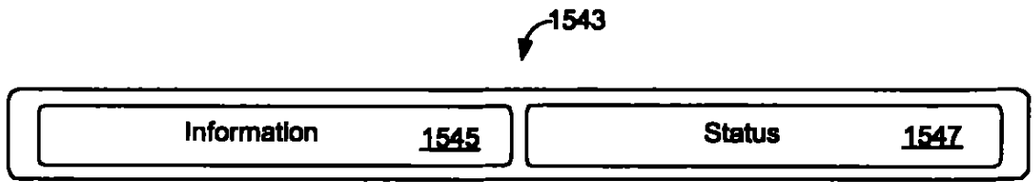
FIG. 15B

116



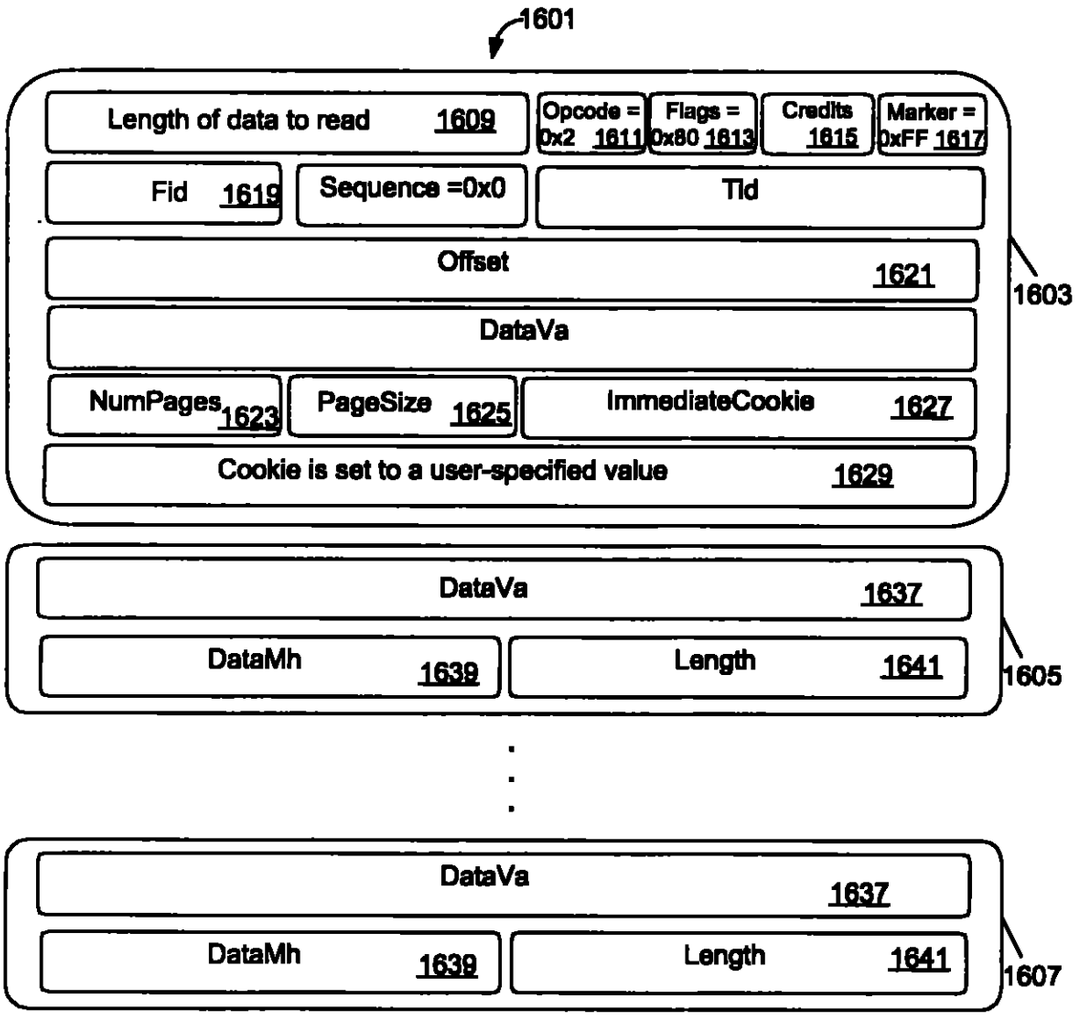
CLIENT WRITE MESSAGE (POLLING)

FIG. 15C



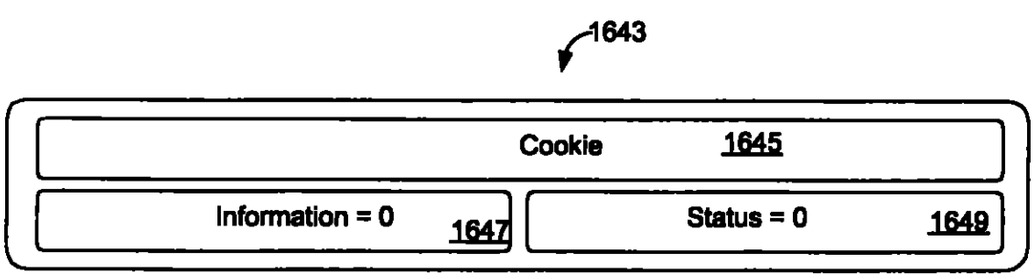
WRITE COMPLETION (POLLING)

FIG. 15D



CLIENT VECREAD MESSAGE (NO POLLING)

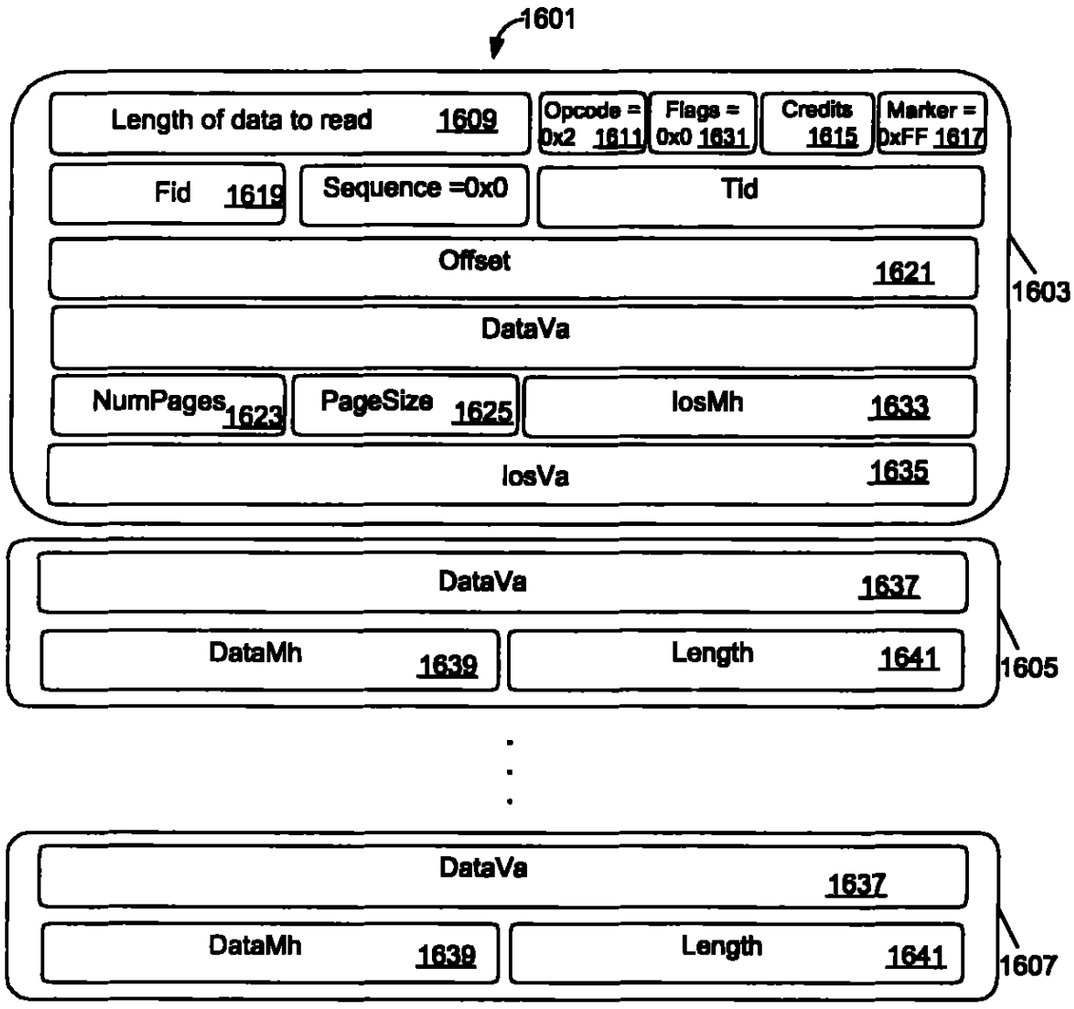
FIG. 16A



VECREAD COMPLETION (NO POLLING)

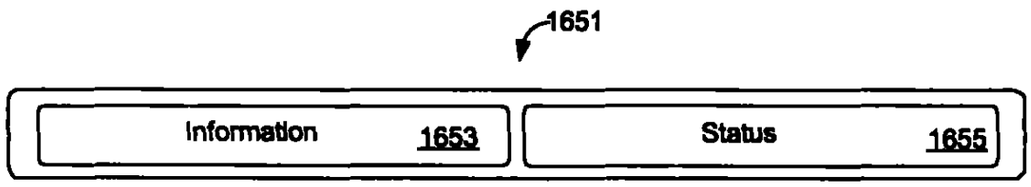
FIG. 16B

148



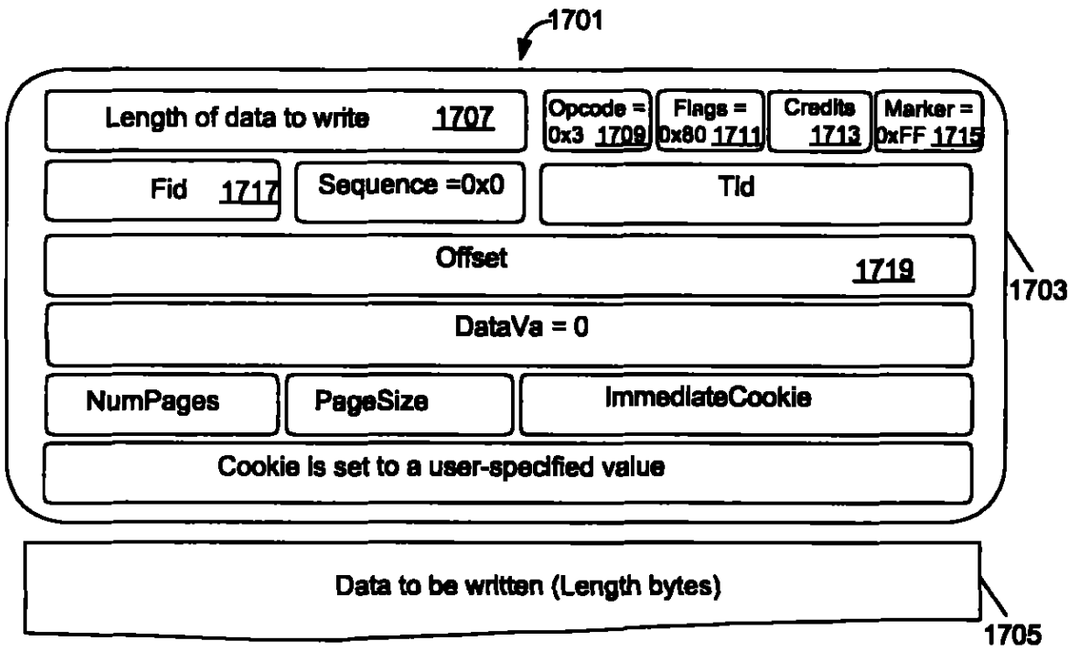
CLIENT VECREAD MESSAGE (POLLING)

FIG. 16C



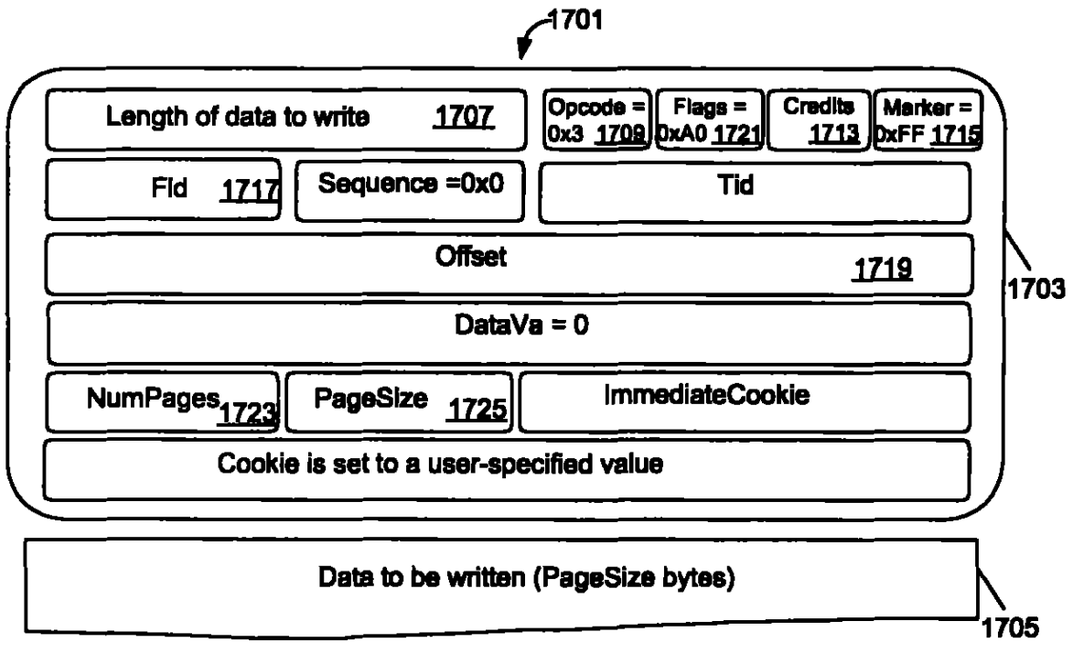
VECREAD COMPLETION (POLLING)

FIG. 16D



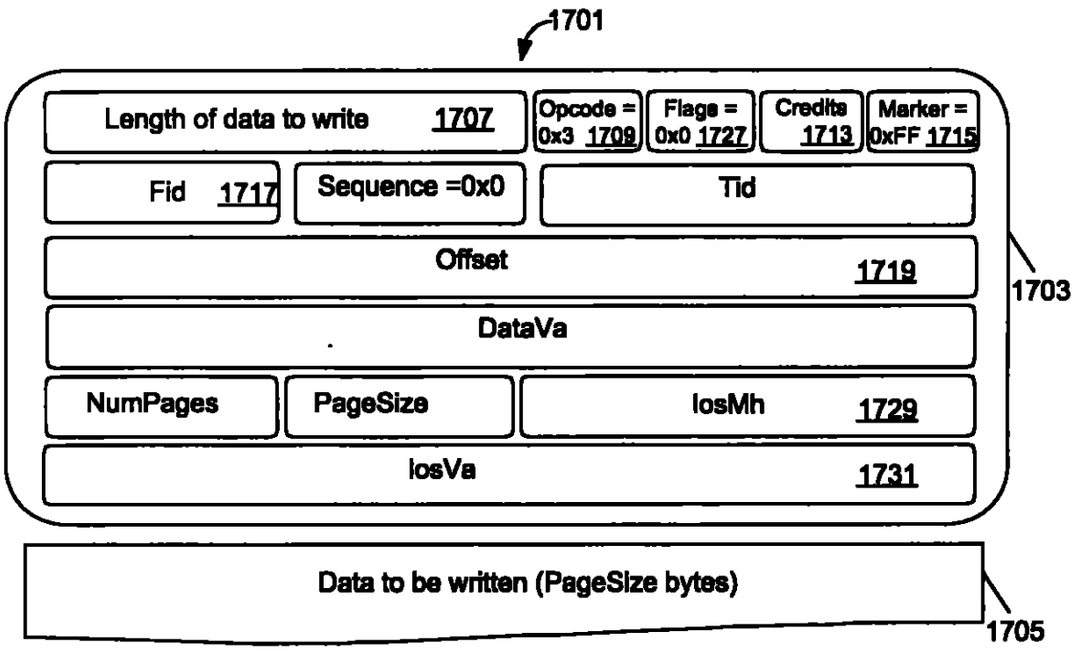
CLIENT VECWRITE MESSAGE, NOT COLLAPSED (NO POLLING)

FIG. 17A



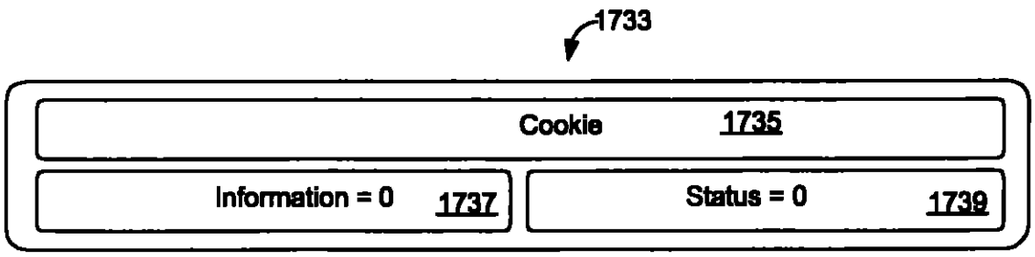
CLIENT VECWRITE MESSAGE, COLLAPSED (NO POLLING)

FIG. 17B



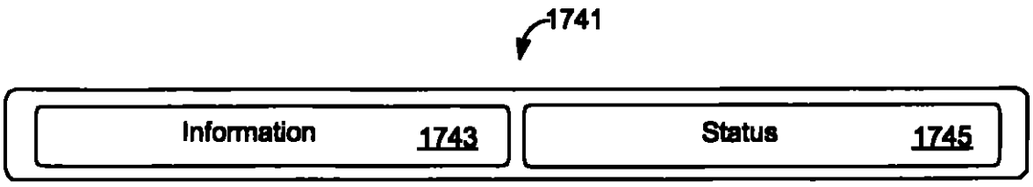
CLIENT VECWRITE MESSAGE, COLLAPSED (POLLING)

FIG. 17C



CLIENT VECWRITE COMPLETION (NO POLLING)

FIG. 17D



CLIENT VECWRITE COMPLETION (POLLING)

FIG. 17E

121

122

APOSTILLE

(Convention de La Haye du 5 octobre 1961)

1. Country: *United States of America*

This public document

2. has been signed by N. Williams

3. acting in the capacity of Certifying Officer

4. bears the seal/stamp of Patent and Trademark Office

Certified

5. at Washington, D.C.

6. the eighth of December, 2004

7. by *Assistant Authentication Officer, United States Department of State*

8. No. 05005596-1

9. Seal/Stamp:

10. Signature:

Sonya N. Johnson

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
8 FEB 2005
ESTA FOTOCOPIA CORRESPONDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

122

123

1253881

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

December 06, 2004

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF A DOCUMENT RECORDED ON JUNE 15, 2004

NOTARÍA 16 DE BOGOTÁ, D.C.
LIZARETH RAMIREZ R.
NOTARIA
08 FEB 2005

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

N. WILLIAMS
Certifying Officer

123

RECORDATION FORM COVER SHEET
PATENTS ONLY

U.S. Department of Commerce
Patent and Trademark Office
PATENT

TO: The Commissioner of Patents and Trademarks: Please record the attached original document(s) or copy(ies).

Submission Type
- ☒ New
- ☐ Resubmission (Non-Recordation)
  Document ID #
- ☐ Correction of PTO Error
  Reel # Frame #
- ☐ Corrective Document
  Reel # Frame #

Conveyance Type
- ☒ Assignment
- ☐ License
- ☐ Merger
- ☐ Security Agreement
- ☐ Change of Name
- ☐ Other:

Conveying Party(ies) — **Execution Date(s)**
1. MOHAMED, Ahmed H. — 06/10/2004
2. VOELLM, Anthony F. — 06/10/2004
3.
4.
5.
6.

☐ Mark if Additional Names of Conveying Parties Attached

Receiving Party
Name: Microsoft Corporation
Name:
Address: One Microsoft Way
Address:
Address: Redmond (City) | Washington (State/Country) | 98052 (Zip Code)

☐ Mark if Additional Names of Receiving Parties Attached

Correspondent Name and Address

Richard E. Fontana, Reg. No. 59,592
Lloyd& Volt & Mayer, Ltd.
Two Prudential Plaza, Suite 4900
Chicago, Illinois 60601-6780

Telephone: (312) 616-5600
Facsimile: (312) 616-5700
Attorney Docket No. 223814

Pages Enter the total number of pages of the attached conveyance document including any attachments: 2

Application Number(s) or Patent Number(s) ☐ Mark if additional numbers attached
Enter either the Patent Application Number or the Patent Number; DO NOT ENTER BOTH numbers for the same property.

Patent Application Numbers			Patent Numbers		
10/748,959					

If this document is being filed together with a *new* Patent Application, enter the date the patent application was signed by the first named executing inventor. — Month Day Year

Patent Cooperation Treaty (PCT)
Enter PCT application number *only if* a U.S. Application Number has not been assigned.

PCT	PCT	PCT
PCT	PCT	PCT

Number of Properties Enter the total number of properties involved: 1

Fee Amount
Fee Amount for Properties Listed (37 CFR 3.41): $40.00

Method of Payment:
- ☐ Enclosed is a check in the amount of
- ☒ Charge Deposit Account No. 12-1216

Authorization to Charge Additional Fees to Deposit Account No. 12-1216: ☐ Yes ☒ No

Statement and Signature
To the best of my knowledge and belief, the foregoing information is true and correct and any attached copy is a true copy of the original document. Charges to deposit account are authorized, as indicated herein.

Richard E. Fontana	Richard E. Fontana	June 18, 2004
Name of Person Signing	Signature	Date

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial Number 10/749,959
Filing Date ..... December 31, 2003
Inventorship ..... Ahmed H. Mohamed et al.
Applicant ..... Microsoft Corporation
Attorney Docket No. 229814
MS Docket No. 305420.01
Title: LIGHTWEIGHT INPUT/OUTPUT PROTOCOL

PATENT ASSIGNMENT

PARTIES TO THE ASSIGNMENT

Assignor(s):

Ahmed H. Mohamed
29 210th PL SE
Sammamish, Washington 98074

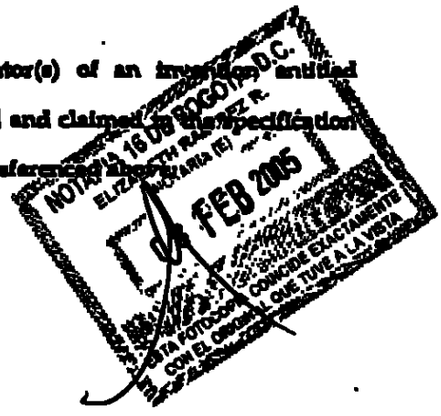
Anthony F. Voellm
9800 229th LN NE
Redmond, Washington 98053

Assignee:

Microsoft Corporation
Corporation in the State of Washington
One Microsoft Way
Redmond, WA 98052

AGREEMENT

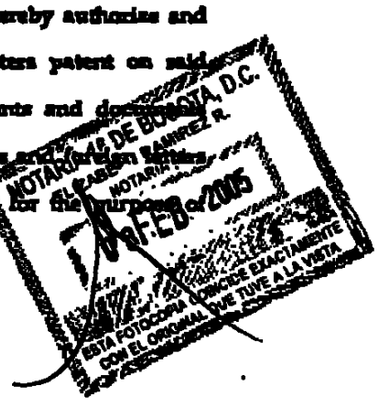
WHEREAS, ASSIGNOR(S) (listed above) are inventor(s) of an invention, entitled "LIGHTWEIGHT INPUT/OUTPUT PROTOCOL," as described and claimed in the specification forming part of an application for United States letters patent referenced above



Attorney Docket No. 223814
MS Docket No. 305420.01
Serial No. 10/748,959

WHEREAS, Microsoft Corporation (hereinafter referred to as ASSIGNEE), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, WA 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNOR(S) hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, reissues and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNOR(S) had this assignment, transfer and sale not been made. ASSIGNOR(S) hereby authorize and request the Commissioner of Patents and Trademarks to issue all letters patent on said invention to ASSIGNEE. ASSIGNOR(S) agree to execute all instruments and documents required for the making and prosecution of applications for United States and foreign letters patent on said invention, for litigation regarding said letters patent, or for the purpose of protecting title to said invention or letters patent therefor.



Attorney Docket No. 223814
MS Docket No. 305420.01
Serial No. 10/749,959

6-10-2004
Date

Ahmed H. Mekkawi

6/10/2004
Date

Anthony F. Voelkl

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 FEB 2005
COPIA FIEL DE LA ORIGINAL
COINCIDE EXACTAMENTE
QUE TUVE A LA VISTA

128

TRADUCCIÓN DEL INGLÉS AL ESPAÑOL DE UNA AP STILLA ANEXA A UN DOCUMENTO ORIGINAL QUE CONSTA DE CINC PÁGINAS ENTREGADO EN UN FOLDER DE LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA POR NORA JIMÉNEZ DE ADAM, TRADUCTORA E INTÉRPRETE OFICIAL. TEL: 571-2572804; 1° DE FEBRERO DE 2005

PRIMERA PÁGINA:

"APOSTILLA

(Convención De La Haya del 5 de octubre de 1961)

1. País: Estados Unidos de América

Este documento público

2. Ha sido firmado por N. Williams

3. Actuando como Funcionario de Certificaciones

4. Lleva el sello/estampilla de La Oficina de Patentes y Marcas Registradas

Certificado

5. En Washington, D.C.

6. El Ocho de diciembre de 2004

7. Por *Asistente del Funcionario de Autenticaciones del Departamento de Estado de los Estados Unidos*

8. N° 05005596-1

9. Sello/Estampilla: Sello seco: "Departamento de Estado – Estados Unidos de América" 10. Firma: Sonya N. Johnson

CERTIFICO QUE ESTA ES UNA TRADUCCION FIEL DEL INGLES AL ESPAÑOL DE UN DOCUMENTO QUE HE TENIDO A LA VISTA; QUE SOY TRADUCTORA E INTERPRETE OFICIAL ACREDITADA POR RESOLUCIONES 0035 Y 530 DE LA UNIVERSIDAD NACIONAL DE COLOMBIA Y DEL MINISTERIO DE LA JUSTICIA RESPECTIVAMENTE, Y QUE ESTOY REGISTRADA ANTE EL MINISTERIO DE RELACIONES EXTERIORES DE COLOMBIA

Bogotá, 1° de febrero 2005

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA
08 FEB 2005

NORA JIMÉNEZ DE ADAM
C.C. 20.335.247

105

129

MINISTERIO DE RELACIONES EXTERIORES

086633

CUYA FIRMA APARECE EN EL DOCUMENTO DESEMPEÑA LAS FUNCIONES ANOTADAS

NO SE ASUME RESPONSABILIDAD DEL TEXTO

2005 FEB -7 A 11:53

JEFE DE LEGALIZACIONES SANTAFE DE BOGOTA D.C.

NOTARIA 16 DE BOGOTA D.C.

129

130

TRADUCCIÓN DEL INGLÉS AL ESPAÑ L DE UN DOCUMENTO ORIGINAL QUE CONSTA DE CINCO PÁGINAS ENTREGADO EN UN FOLDER DE LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA POR NORA JIMÉNEZ DE ADAM, TRADUCTORA E INTÉRPRETE OFICIAL. TEL: 571-2572804; 1° DE FEBRERO DE 2005

SEGUNDA PÁGINA:

Emitida en papelería especial con reborde negro-gris con el número A1253881 en la parte superior del documento: **"ESTADOS UNIDOS DE AMERICA – PARA TODOS AQUELLOS A QUIENES PUEDA INTERESAR: - DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS – Oficina de Patentes y Marcas Registradas de los Estados Unidos – 6 de diciembre de 2004**

ESTE DOCUMENTO CERTIFICA QUE LOS ANEXOS SON COPIA FIEL QUE FUE REGISTRADO EN ESTA OFICINA EL 15 DE JUNIO DE 2004.

Bajo la Potestad del COMISIONADO DE PATENTES Y MARCAS REGISTRADAS

(Firma) N. WILLIAMS – Funcionario de Certificaciones

Estampilla dorada con emblema en el centro y a su alrededor: "Oficina de Patentes y Marcas Registradas de los Estados Unidos– Departamento de Comercio"

TERCERA PÁGINA:

¿.15.2004 11:19AM LVM 312 616 5700 N° 7174 P.2

FORMATO DE PORTADA DE REGISTRO LEGAL

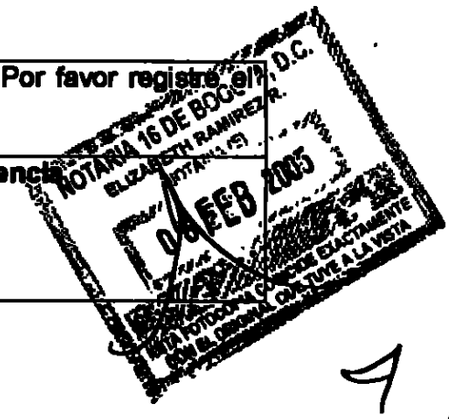
SÓLO PARA PATENTES

Departamento de Comercio de los Estados Unidos

Oficina de Patentes y Marcas Registradas

PATENTE

PARA: Comisionado de Patentes y Marcas Registradas: Por favor registre el (los) documento (s) originales anexos o copia (s).	
Tipo de Solicitud	**Tipo de Transferencia**
X Nueva	X Cesión
_ Re-emisión (no de registro legal)	_ Licencia



7

137

131

No. de Identificación del documento	_ Fusión
_ Corrección de Error PTO.	_ Acuerdo de Garantía
No. de Carrete No. de Pantalla	_ Cambio de Razón Social
_ Documento Correctivo	_ Otros:
No. de Carrete No. de Pantalla	

Parte (s) que Transfieren

Nombre	Fecha (s) de Ejecución
1. MOHAMMED, Ahmed	05/10/2004
2. VOELLM, Anthony F.dru	05/10/2004
3.	
4.	
5.	
6.	
_ Llenar en caso de que se anexen los nombres de otras Partes Cedentes	

Parte (s) Receptora (s)

Nombre:	MICROSOFT CORPORATION
Nombre	
Dirección:	One Microsoft Way
Dirección	
Dirección	
Ciudad:	Redmond
Estado/País:	WASHINGTON
Código Postal:	98052
_ Llenar en caso de que se anexen los nombres de otras Partes Receptoras	

Nombre y Dirección del Correspondiente	
Richard E. Fontana, Reg. No. 59,902	Teléfono: (312) 616-5600
Leydig, Voit & Mayer, Ltd.	Telefax: (312) 616-5700
Two Prudential Plaza, Suite 4900	No. de Expediente del Abogado
Chicago, Illinois 60601-6780	223814

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
FEB 2005
HASTA FOTOCOPIA CONCUERDA EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

134

clam	
Páginas: Ingrese el número total de páginas del documento de transferencia incluyendo anexos: 2	
Número (s) de Solicitud o de Patente *Ingrese ya sea el número de Solicitud o el número de patente (NO INGRESE AMBOS números de la misma propiedad)*	_ Llenar en caso de que se anexen números adicionales
Números de Solicitud de Patente	Números de Patente
10/749,959	
En el caso de que este documento sea registrado junto con una *nueva* Solicitud de Patente, ingrese la fecha en que la solicitud de Patente fue firmada por el primer inventor ejecutor nombrado	Mes-Día-Año

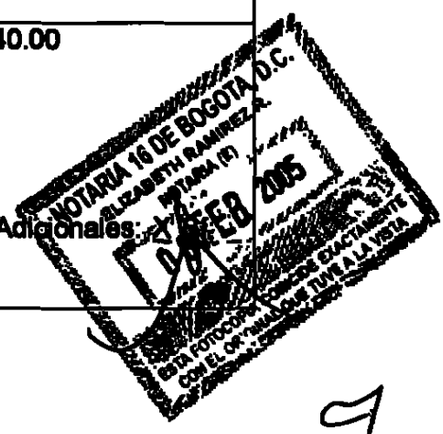
Acuerdo de Cooperación de Patente (Patent Cooperation Treaty – PCT)
Ingrese el número de solicitud de PCT *únicamente* en el caso de que no haya sido asignado un número de Solicitud de Estados Unidos

PCT	PCT	PCT
PCT	PCT	PCT

Número de Propiedades - Ingrese el número total de propiedades involucradas: 1

Derechos
Derechos para las Propiedades Indicadas (37 CFR 3.41): $40.00 Método de Pago: _ Encuentre cheque adjunto por un valor de X Debite la Cuenta no. 12-1216 Se autoriza debitar la Cuenta No. 12-1216 por Derechos Adicionales: No

Declaración y Firma



133

Según mi conocimiento, la información suministrada es verdadera y correcta y cualquiera copia anexa es fiel al documento original. Quedan autorizados a debitar la cuenta de acuerdo con lo indicado.

Richard E. Fontana (Firma) — junio 15 de 2004

Nombre de la Persona que Firma — Fecha

700091548 **PATENTE**

Carrete: 014731; Pantalla: 0475

Borde derecho a lo largo: ch $40,00 121216 10749969

CUARTA PÁGINA:

¿ 2004 11:20 AM LVM 312 616 5700 N° 7174 P. 4

EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS ESTADOS UNIDOS

N° de Serie.....	10/749,959
Fecha de Presentación....	31 de diciembre de 2003
Autoría del Invento.....	Ahmed H. Mohammed y otros
Solicitante.....	Microsoft Corporation
Expediente del Abogado N°	223814
Expediente de MS N°	305420.01

Título: PROTOCOLO LIGERO DE ENTRADA/SALIDA

CESIÓN DE PATENTE

PARTES DE LA CESIÓN

Cedente(s):

Ahmed H. Mohammed
29 210th. PL SE
Sammamish, Washington 98074

Anthony F. Voelim
9800 229th. LN NE
Redmond, Washington 98053

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
0 6 FEB 2005
COPIA FOTOSTATICA CONCUERDA EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

9

133

134

Cesionario:

Microsoft Corporation

Corporation in the State of Washington (Corporación en el Estado de Washington)

One Microsoft Way

Redmond, WA 98052

CONVENIO

CONSIDERANDO QUE EL(LOS) CEDENTE(S) (relacionados arriba) es (son) creador(es) de un invento llamado "PROTOCOLO LIGERO DE ENTRADA/SALIDA", de acuerdo con lo descrito y afirmado en la especificación que forma parte de una solicitud de patentes para los Estados Unidos referenciados anteriormente;

PATENTE

CARRETE: 014731 PANTALLA: 0476

QUINTA PÁGINA:

¿ 1.2004 12:20 AM LVM 312 616 5700 N° 7174 P. 5

Expediente del Abogado N° 223814

Expediente de MS N°: 305420.01

CONSIDERANDO QUE Microsoft Corporation (en adelante el CESIONARIO), una corporación del Estado de Washington, cuya sede es One Microsoft Way, Redmond, Washington 98052, desea adquirir la totalidad de los derechos, títulos e intereses en los inventos y por ellos, y en o por cualquiera de las patentes que puedan ser otorgadas en los Estados Unidos y en cualquiera o en todos los países extranjeros;

AHORA, POR LO TANTO, a cambio de contraprestaciones onerosas, el recibo de las cuales se reconocen en este documento, los CEDENTES por el presente, venden ceden y transfieren al CESIONARO, los derechos, títulos e intereses totales en y por dicho invento, dichas solicitudes, cualquiera y todas las patentes que puedan ser otorgadas sobre el mismo en los Estados Unidos de América y sus posesiones territoriales y en cualquier o en todos los países extranjeros, y en cualquiera y en todas las divisiones, reexpisiones y

134

135

extensiones de las mismas, incluyendo el derecho a registrar solicitudes en el extranjero directamente a nombre del CESIONARIO y a reclamar los derechos prioritarios derivados de dicha solicitud de los Estados Unidos a los cuales tienen derecho las solicitudes extranjeras, en virtud de convenios internacionales, tratados u otras maneras; dicho invento, solicitud y todas las patentes sobre el mismo serán conservadas y en posesión del CESIONARIO y sus sucesores o apoderados, para su uso y beneficio y el de sus sucesores y apoderados, tan completa y totalmente como si fuesen conservados y en posesión de los CEDENTES, si esta cesión, transferencia o venta no se hubiere realizado. Los CEDENTES por el presente autorizan y solicitan al Comisionado de Patentes y Marcas Registradas que emita todas las patentes sobre dicho invento al CESIONARIO. Los CEDENTES acuerdan formalizar todos los instrumentos y documentos requeridos para llevar a cabo y proseguir la solicitud de las patentes en los Estados Unidos, y en el extranjero sobre dicho invento, para fines de litigios relacionados con dichas patentes o para proteger dicho título sobre el invento o patente.

PATENTE
CARRETE: 014731 PANTALLA: 0477

SEXTA PÁGINA:

¿¿ 2004 11:20 AM LVM 312 616 5700 N° 7174 P. 6

Expediente del Abogado N° 223814
Expediente de MS N° 305420.01

Fecha: 6/10/2004 (Firma) AHMED H. MOHAMMED

Fecha: 6/10/2004 (Firma) ANTHONY F. VOELLM

PATENTE

REGISTRADO: 06/15/2004 **CARRETE: 014731 PANTALLA: 0478**

CERTIFICO QUE ESTA ES UNA TRADUCCION FIEL DEL INGLES AL ESPAÑOL DE UN DOCUMENTO QUE HE TENIDO A LA VISTA; QUE SOY TRADUCTORA E INTERPRETE OFICIAL ACREDITADA POR RESOLUCIONES 0035 Y 530 DE LA UNIVERSIDAD NACIONAL DE COLOMBIA Y DEL

4

135

136

MINISTERIO DE LA JUSTICIA RESPECTIVAMENTE, Y QUE ESTOY REGISTRADA ANTE EL MINISTERIO DE RELACIONES EXTERIORES DE COLOMBIA

Bogotá, 1° de febrero 2005

Traducción e Interpretación Oficial
Inglés-Español / Español-Inglés,
Según Resolución 0530 del 11 de Junio de 2002
Emitida por el Ministerio de Justicia y del Derecho
Nora Jiménez de Adam
C.C. 20.335.247

NORA JIMENEZ DE ADAM
C.C. 20.335.247

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 FEB 2005
ESTA FOTOCOPIA CONCUERDA EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

136

137
137

SUPERINDUSTRIA Y COMERCIO
Radicacion : 05[illegible] 00000000 Folios: 136
Fecha (AMD): 2005-05-25 11:03:31
Tramite : 011 PCT-PATENT 370 FASE NACI 411 PRESENTAR
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

REQUISITOS NECESARIOS PARA PRESENTACION Y AGILIZACION DEL TRAMITE

		USUARIO	RADICADOR
PATENTE DE INVENCIÓN	MODELO DE UTILIDAD	( )	( )
- Indicación de que se solicita la concesión de una patente.		( )	(✓)
- Datos de identificación del solicitante o de la persona que se presenta la solicitud.		( )	(✓)
- Descripción de la invención.		( )	( )
Dibujos de ser estos pertinentes.		( )	(✓)
Comprobante de Pago de las tasas establecidas.			
COMPLETA ( )	INCOMPLETA	( )	( )
DISEÑO INDUSTRIAL ( )			
- Indicación de que se solicita el registro de Diseño Industrial		( )	( )
- Datos de identificación del solicitante o de la persona que presenta		( )	( )
- la solicitud.			
- Representación gráfica o fotográfica del Diseño Industrial o muestra del Material que incorpora el diseño.		( )	( )
Comprobante de pago de las tasas establecidas		( )	( )
COMPLETA ( )	INCOMPLETA	( )	( )
ESQUEMA DE TRAZADO ( )			
- Indicación de que solicita el registro de un Esquema de trazado.		( )	( )
- Datos de identificación del solicitante o de la persona que presenta La solicitud.		( )	( )
Representación gráfica del esquema de trazado		( )	( )
Comprobante de pago de las tasas establecidas.		( )	( )
COMPLETA ( )	INCOMPLETA	( )	( )

Firma Responsable:

Fecha: mayo 25 de 2005

133

PETITORIO

Industria y Comercio
SUPERINTENDENCIA

SUPERINDUSTRIA Y COMERCIO
Radicación : 05804646 00000020 Folios: 136
Fecha (AMJ): 2005-09-21 1:03:31
Trámite : 911 PCT-PATENT 372 FASE NACI ALL PRESENTA
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

FORMULARIO UNICO DE SOLICITUD DE PATENTE
PCT
ENTRADA EN FASE NACIONAL

① SOLICITUD DE:

[X] Patente de Invención [ ] Patente de Modelo de Utilidad

[X] Capítulo I [ ] Capítulo II

② SOLICITANTE (71)

Nombre: **MICROSOFT CORPORATION**

Dirección: One Microsoft Way Redmond, Washington 98052, Estados Unidos de America

Nacionalidad o Domicilio: REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA.

Lugar de Constitución: Estado de Washington, Estados Unidos de América

Teléfono: 3123600 Fax: 3122410

E-mail: general@cardenasycardenas.com

IDENTIFICACION

C.C. [ ] NIT [ ]
C.E. [ ] Otro [X]
Cual _____
Número _____

③ REPRESENTANTE O APODERADO

Nombre: **DARIO CARDENAS NAVAS**

Dirección: Carrera 7 No 71-52 Torre B Piso 9

Teléfono: 3123600 Fax: 3122410

E-Mail: general@cardenasycardenas.com

IDENTIFICACION

C.C. [X] NIT [ ]
C.E. [ ] Otro [ ]
Cual
No 17.066.629 BTA TP 1.250 C.S.J.

④ INVENTOR(ES) (72)

1.Nombre:
1. Ahmed H. Mohamed
2. Anthony F. Voellm

2. Dirección:
1. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
2. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America

3.Nacionalidad o Domicilio:
1. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
2. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA

⑤ PCT (86)

Solicitud Internacional No. PCT/US2004/024026 Fecha Julio 26, 2004
Publicación Internacional No. Fecha

⑥ Título(54): PROTOCOLO LIGERO DE ENTRADA / SALIDA

AMR/amg
Fol No P-326

139

PROTOCOLO LIGERO DE ENTRADA / SALIDA

⑦ Clasificación Internacional (51):

⑧ Prioridad

SI [X] NO [ ]

(33)País de Origen	(32)Fecha	(31)N° de Solicitud
U.S.A.	Diciembre 31, 2003	10/749.959

Comprobante de Pago No.: 05-37.298 **Fecha** Mayo 19, 2005

ANEXOS

- [X] Comprobante de pago de la tasa de presentación de la solicitud.
- [X] Documento que acredita la existencia y representación legal cuando el solicitante sea persona jurídica. (Protocolo 16.069)
- [X] Poderes, si fuere el caso. (Protocolo 16.069)
- [X] Documento de cesión del inventor al solicitante o a su causante.
- [ ] Declaraciones
- [X] Copia de la solicitud internacional. (Resumen, descripción, reivindicaciones, dibujos)
- [X] Traducción de la solicitud internacional al castellano. (Resumen, descripción, reivindicaciones, dibujos)
- [ ] Copia del examen preliminar internacional efectuado por la Autoridad Internacional de Examen Preliminar (IPEA).
- [ ] Traducción del examen preliminar internacional efectuado por la IPEA.
- [ ] De ser el caso, copia del contrato de acceso.
- [ ] De ser el caso, documento que acredite la licencia o autorización de uso de conocimientos tradicionales de las comunidades indígenas.
- [ ] De ser el caso, certificado de depósito del material biológico.
- [X] Arte final 12x12.
- [ ] De ser el caso, información sobre otras solicitudes de patente o títulos obtenidos en el extranjero por el mismo titular o su causante, relacionadas parcial o totalmente con la invención de esta solicitud.
- [X] De ser el caso, modificaciones efectuadas a la solicitud internacional.
- [ ] Otros.

11 FIGURA CARACTERISTICA

12 Solicito la Concesión de la patente,

NOMBRE: DARIO CARDENAS NAVAS

FIRMA:

C.C. 17.068.629 BTA TP 1.250

140

numero de bites escritos. Status 1739 se fija en 0, indicando éxito, o en otro NTSTATUS, indicando fracaso. Cookie 1735 se fija en el valor de Cookie value determinado por el cliente en el encabezado del mensaje de escritura.

[0121] en el caso de consulta, para ambos casos, I/O colapsado y no colapsado, el servidor RDMAs un LWIO_IO_STATUS_BLOCK. La *FIG. 17E* nos muestra una representación ilustrativa de un LWIO_IO_STATUS_BLOCK 1741devuelto por el servidor en una implementación del invento. Information 1 743 se fija en el numero de bites escritos. Status 1745 se fija en 0, indicando éxito, o en otro NTSTATUS, indicando fracaso.

Conclusión

[0122] Aunque se han descrito y mostrado implementaciones ilustrativas del presente invento, se puede apreciar que varios cambios se pueden hacer sin apartarse del invento. En forma similar, se pueden variar en forma intercambiable los pasos en el proceso que se ha descrito logrando el mismo resultado. Adicionalmente, los ejemplos descritos anteriormente no tienen la intención de limitar el alcance del invento. Por el contrario, la intención es la de cubrir todas las modificaciones, construcciones alternativas, y equivalentes, que estén dentro del espíritu y alcance del invento.

LO QUE SE REIVINDICA ES:

1. Un sistema para descargar las tareas de entrada/salida I/O de un primer computador a un segundo computador, lo que comprende: un cliente que opera en el primer computador; un servidor que opera en el segundo computador; y por lo menos un canal RDMA que vincula al primer computador con el segundo, donde el primer computador y el segundo computador se comunican de acuerdo a un protocolo que consta de una fase de descubrimiento en red y una fase de procesamiento I/O.

2. El sistema que se reivindica en 1 donde la fase de procesamiento I/O, y las operaciones de lectura se implementan utilizando RDMA y las operaciones de escritura se implementan utilizando operaciones de envió.

3. El sistema que se reivindica en 1 donde el protocolo se utiliza en asociación con un segundo protocolo de red.

4. El sistema que se reivindica en 3 donde el segundo protocolo es SMB.

5. El sistema que se reivindica en 3 donde el segundo protocolo es CIFS.

6. Un medio legible por computador que guarda instrucciones ejecutables por computador y datos legibles por computador que comprenden un producto de programación de computador para utilizar en un sistema para descargar las tareas de entrada/salida I/O de un primer computador a un segundo computador, sistema que consta de: por lo menos un canal RDMA que vincula al primer computador con el segundo

35

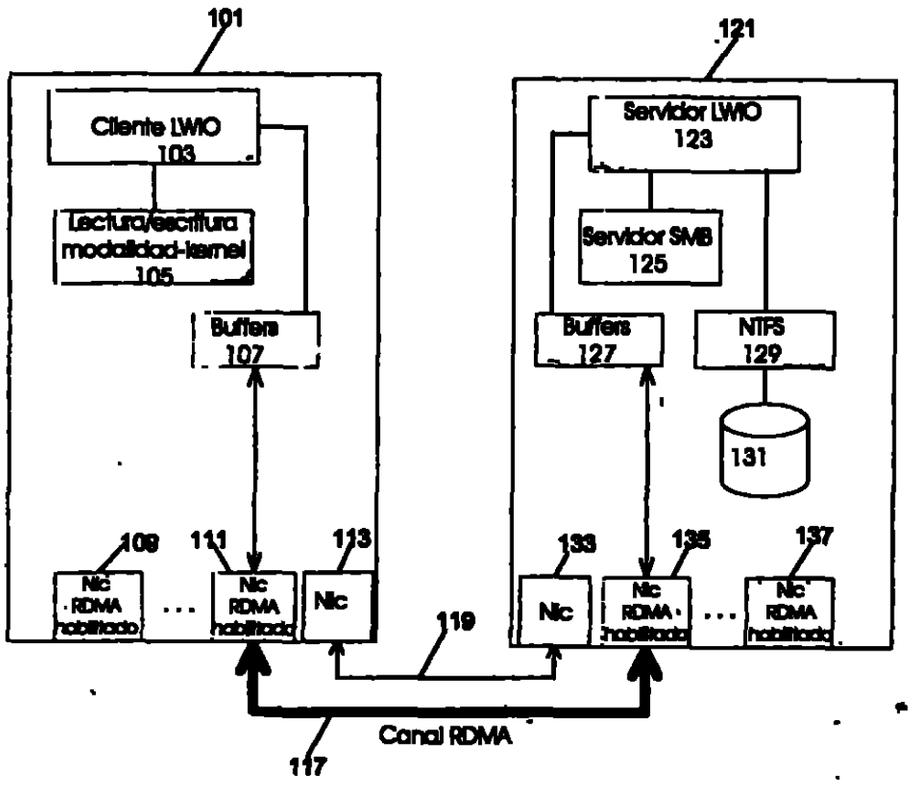

101
Cliente LWIO
103
Lectura/escritura
modalidad-kernel
105
Buffers
107
109
111
113
Nic
RDMA
habilitada
Nic
RDMA
habilitada
Nic
119
121
Servidor LWIO
123
Servidor SMB
125
Buffers
127
NTFS
129
131
133
135
137
Nic
Nic
RDMA
habilitada
Nic
RDMA
habilitada
Canal RDMA
117

FIG. 1

número de bites escritos. Status 1739 se fija en 0, indicando éxito, o en otro NTSTATUS, indicando fracaso. Cookie 1735 se fija en el valor de Cookie value determinado por el cliente en el encabezado del mensaje de escritura.

[0121] en el caso de consulta, para ambos casos, I/O colapsado y no colapsado, el servidor RDMAs un LWIO_IO_STATUS_BLOCK. La FIG. 17E nos muestra una representación ilustrativa de un LWIO_IO_STATUS_BLOCK 1741devuelto por el servidor en una implementación del invento. Information 1 743 se fija en el número de bites escritos. Status 1745 se fija en 0, indicando éxito, o en otro NTSTATUS, indicando fracaso.

Conclusión

[0122] Aunque se han descrito y mostrado implementaciones ilustrativas del presente invento, se puede apreciar que varios cambios se pueden hacer sin apartarse del invento. En forma similar, se pueden variar en forma intercambiable los pasos en el proceso que se ha descrito logrando el mismo resultado. Adicionalmente, los ejemplos descritos anteriormente no tienen la intención de limitar el alcance del invento. Por el contrario, la intención es la de cubrir todas las modificaciones, construcciones alternativas, y equivalentes, que estén dentro del espíritu y alcance del invento.

LO QUE SE REIVINDICA ES:

1. Un sistema para descargar las tareas de entrada/salida I/O de un primer computador a un segundo computador, lo que comprende: un cliente que opera en el primer computador; un servidor que opera en el segundo computador; y por lo menos un canal RDMA que vincula al primer computador con el segundo, donde el primer computador y el segundo computador se comunican de acuerdo a un protocolo que consta de una fase de descubrimiento en red y una fase de procesamiento I/O.

2. El sistema que se reivindica en 1 donde la fase de procesamiento I/O, y las operaciones de lectura se implementan utilizando RDMA y las operaciones de escritura se implementan utilizando operaciones de envió.

3. El sistema que se reivindica en 1 donde el protocolo se utiliza en asociación con un segundo protocolo de red.

4. El sistema que se reivindica en 3 donde el segundo protocolo es SMB.

5. El sistema que se reivindica en 3 donde el segundo protocolo es CIFS.

6. Un medio legible por computador que guarda instrucciones ejecutables por computador y datos legibles por computador que comprenden un producto de programación de computador para utilizar en un sistema para descargar las tareas de entrada/salida I/O de un primer computador a un segundo computador, sistema que consta de: por lo menos un canal RDMA que vincula al primer computador con el segundo

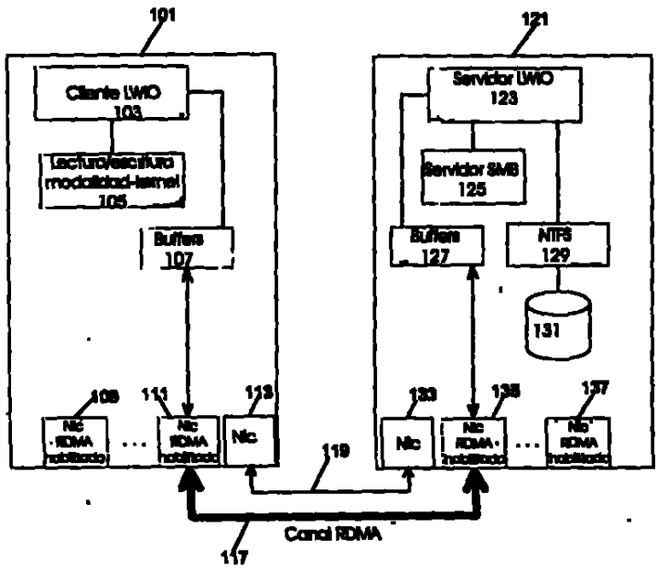

101
Cliente LWIO
103
Lectura/escritura
modalidad-kernel
105
Buffers
107
109
111
113
Nic
RDMA
Nic
RDMA
Nic
121
Servidor LWIO
123
Servidor SMB
125
Buffers
127
NTFS
129
131
133
135
137
Nic
Nic
RDMA
Nic
RDMA
119
Canal RDMA
117

FIG. 1