

[19] 中华人民共和国国家知识产权局



## [12] 发明专利申请公布说明书

[21] 申请号 200480001333.0

[51] Int. Cl.

G06F 15/16 (2006.01)

G06F 13/28 (2006.01)

G06F 3/00 (2006.01)

[43] 公开日 2009 年 2 月 25 日

[11] 公开号 CN 101375263A

[22] 申请日 2004.7.26

[21] 申请号 200480001333.0

[30] 优先权

[32] 2003.12.31 [33] US [31] 10/749,959

[86] 国际申请 PCT/US2004/024026 2004.7.26

[87] 国际公布 WO2005/067430 英 2005.7.28

[85] 进入国家阶段日期 2005.5.20

[71] 申请人 微软公司

地址 美国华盛顿州

[72] 发明人 A·H·莫哈米德 A·F·弗尔梅

[74] 专利代理机构 上海专利商标事务所有限公司

代理人 张政权

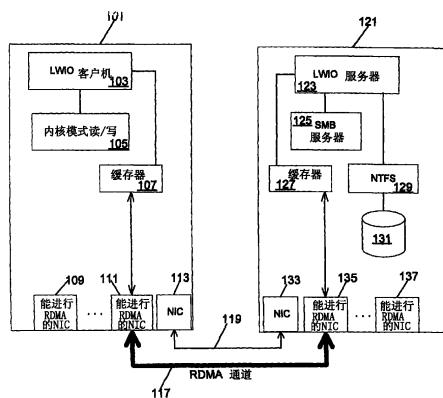
权利要求书 3 页 说明书 23 页 附图 23 页

[54] 发明名称

轻型输入/输出协议

[57] 摘要

揭示了使用能进行 RDMA 的网络互连从第一计算机(图 1, 101)向第二计算机(121)下传 I/O 处理的方法和系统。该方法和系统包括第一计算机(101)上的客户机(103),它借助轻型输入/输出(LWIO)协议通过 RDMA 连接(117)与第二计算机(121)上的服务器(123)通信。该协议通常包括网络发现阶段及随后的 I/O 处理阶段。在发现阶段,客户机(103)和服务器(123)确定共享的能进行 RDMA 的提供者的最小列表。在 I/O 处理阶段,客户机(103)发送 I/O 请求以便通过互相认证的 RDMA 通道(117)下传到第二机器(121)。I/O 模型是非对称的,读操作使用 RDMA 实现而写操作使用常规的发送实现。能以轮询和中断模式完成读和写的请求。通过信用点机制管理缓存器。



1. 一种用于将输入/输出（I/O）任务从第一计算机下传的第二计算机的系统，其特征在于，包括：

在所述第一计算机上运行的客户机；  
在所述第二计算机上运行的服务器；以及

链接所述第一计算机和第二计算机的至少一个 RDMA 通道，其中，所述第一计算机和第二计算机按包括网络发现阶段和 I/O 处理阶段的协议进行通信。

2. 如权利要求 1 所述的系统，其特征在于，在所述 I/O 处理阶段，读操作是使用 RDMA 来实现的，而写操作是使用发送操作来实现的。

3. 如权利要求 1 所述的系统，其特征在于，所述协议是结合第二网络协议来使用的。

4. 如权利要求 3 所述的系统，其特征在于，所述第二协议是 SMB。

5. 如权利要求 3 所述的系统，其特征在于，所述第二协议是 CIFS。

6. 一种计算机可执行指令和计算机可读数据的计算机可读介质，包括在将输入/输出（I/O）任务从第一计算机下传到第二计算机的系统中使用的计算机程序产品，其特征在于，所述系统包括：

链接所述第一计算机和第二计算机的至少一个 RDMA 通道，其中，所述第一计算机和第二计算机按包括网络发现阶段和 I/O 处理阶段的协议进行通信。

7. 一种用于将输入/输出（I/O）任务从第一计算机下传到第二计算机的方法，其特征在于，包括：

由所述第一计算机上的客户机和所述第二计算机上的程序程序发现一个或多个共享的能进行 RDMA 的提供者；以及

由所述客户机发送 I/O 处理请求，以供所述第二计算机上的服务器完成。

8. 如权利要求 7 所述的方法，其特征在于，发现一个或多个共享的能进行 RDMA 的提供者还包括：

由所述客户机从所述服务器获得服务器请求继续键；

由所述客户机打开通向所述服务器的管道；

由所述客户机经过所述管道发送协商请求；以及

由所述服务器经过所述通道发送包括共同提供者的最小列表的协商响应。

9. 如权利要求 7 所述的方法，其特征在于，还包括：

---

由所述客户机创建经过共享的能进行 RDMA 的提供者到所述服务器的 RDMA 连接；以及

由所述客户机及所述服务器认证所述 RDMA 连接。

10. 如权利要求 9 所述的方法，其特征在于，还包括：

由所述客户机通过所述 RDMA 连接向所述服务器注册一个或多个文件。

11. 如权利要求 10 所述的方法，其特征在于，所述注册一个或多个文件包括：

由所述客户机向所述服务器发送注册文件消息；以及

由所述服务器向所述客户机发送注册文件完成消息。

12. 如权利要求 9 所述的方法，其特征在于，所述认证 RDMA 连接还包括：

由所述客户机向所述服务器发送包括密钥的认证请求消息；

若所述密钥匹配以前由所述服务器向所述客户机发送的密钥，则由所述服务器向所述客户机发送认证响应消息。

13. 如权利要求 12 所述的方法，其特征在于，所述之前的密钥是包含在由所述服务器向所述客户机发送的协商响应消息中的密钥。

14. 如权利要求 12 所述的方法，其特征在于，还包括：

由所述服务器向所述客户机发送一状态响应消息，以完成所述认证。

15. 如权利要求 7 所述的方法，其特征在于，所述发送 I/O 处理请求包括由所述客户机发送下列之一：(a) 关闭请求，(b) 取消请求，(c) 读请求，(d) 写请求，(e) 向量化读请求，和(f) 向量化写请求。

16. 如权利要求 15 所述的方法，其特征在于，还包括：

由所述服务器通过使用 RDMA 写操作发送数据来完成所述读请求和所述向量化读请求；以及

由所述服务器通过使用常规发送操作发送数据来完成所述写请求和所述向量化写请求。

17. 如权利要求 15 所述的方法，其特征在于，所述向量化写请求包括所述请求的首部的崩溃标志。

18. 如权利要求 7 所述的方法，其特征在于，发送所述 I/O 请求还包括表明所述服务器是否应以轮询模式完成。

19. 如权利要求 18 所述的方法，其特征在于，表明是否以轮询模式完成包括通过在所述 I/O 处理请求的首部中设置中断标志来表明不应以轮询模式完

成。

20. 如权利要求 18 所述的方法，其特征在于，还包括：

若所述客户机表明不应以轮询模式完成，则由所述服务器借助 RDMA 传输通过向所述第一计算机发送状态块来完成所述 I/O 处理请求。

21. 如权利要求 18 所述的方法，其特征在于，还包括：

若所述客户机表明应以轮询模式完成，且所述客户机已向所述服务器发送中断请求消息，则由所述服务器借助普通发送向所述客户机发送中断响应消息。

22. 如权利要求 7 所述的方法，其特征在于，所述发送 I/O 处理请求还包括在所述请求的首部指定信用点数。

23. 一种存储用于实现将输入/输出 (I/O) 任务从第一计算机下传到第二计算机的方法的计算可执行指令的计算机可读介质，所述方法包括：

由所述第一计算机上的客户机和所述第二计算机上的服务器发现一个或多个共享的能进行 RDMA 的提供者；以及

由所述客户机发送 I/O 处理请求，以供所述第二计算机上的服务器完成。

24. 一种用于管理输入/输出下传协议中的缓存器的方法，其特征在于，所述方法包括：

由服务器向客户机发送包括设成信用点数的信息字段的增量信用点消息，其中，若所述数字是负数-N，则所述服务器请求所述客户机收回 N 个信用点；

若所述信用点数是负数-N，则由所述客户机向所述服务器发送 N 个信用点消息，而否则由所述客户机向所述服务发送一信用点消息；以及

对每个由所述客户机发送的信用点消息，由所述服务器向所述客户机发送状态响应消息。

---

## 轻型输入/输出协议

### 技术领域

本发明一般涉及远程文件访问的系统和方法，尤其涉及使用远程直接存储器存取（RDMA）来下传（offload）输入/输出处理的技术。

### 背景技术

在计算环境中通常希望节省宝贵的 CPU 资源。对如应用程序服务器节点的网络那样的环境，那样的节省是尤其至关重要的。当网络变得更快时，它们对 CPU 提出更高要求来处理包并完成 I/O 操作，这导致更慢的应用程序性能。对于如数据库那样固有地 I/O 密集型应用程序这特别不利。

补救这一问题的一种方法是从 CPU 中下传过多的 I/O 和网络处理。在联网环境中，使用分布式文件系统和注入 NFS 或 SMB/CIFS 等传输协议，可能将 I/O 请求从本地机器发送到远程机器。然而，情况不必是本地机器使用这类方法能够达到显著的处理节省。

在单机情况下，通过将 I/O 任务下传到直接存储器存取（DMA）控制器，I/O 处理的负担减轻了。远程直接存储器存取（RDMA）技术是用于多台连网计算机的最近发展的 DMA 扩展。RDMA 使数据能在装有能够进行 RDMA 网络接口卡（NIC）的两台通信机器上的存储缓存器之间传送，而不必涉及源和目标机器双方的 CPU 和操作系统。能使用 RDMA 将 I/O 处理下传到远程机器，使本地机器能重新要求 CPU 周期用于应用程序。RDMA 已被用于高速、高带宽互连技术，如虚拟接口体系结构（VIA）、InfiniBand 和 iWarp。这些互连特别为在数据中心或其它本地文件共享环境中服务器节点的集群之间的高可靠性网络连接而设计。

为了充分利用与 RDMA 技术关联的能力并有效地得到其好处，必须设计定义在本地下传节点和远程机器之间通信的协议。因此需要本发明的轻型输入/输出（LWI0）协议。

### 发明内容

按本发明的一方面，提供将 I/O 任务从第一计算机下传到第二计算机的系统。该系统包括在第一计算机上运行的客户机和在第二计算机上运行的服务器程序。该系统还包括链接第一计算机和第二计算机的一个或多个 RDMA 通道。客户机和服务器程序按照包括网络发现阶段和 I/O 处理阶段的 LWIO 协议通信。结合如 SMB/CIFS 等另外网络协议使用 LWIO 协议，充分调动了第二协议的安全性和认证的基础结构。为提供更好的安全模型，协议中的 I/O 模型是非对称的：读使用 RDMA 来实现，而写使用发送操作来实现。

按照本发明的另一方面，提供将 I/O 任务从第一计算机下传到第二计算机的方法。该方法利用两台计算上共同的能进行 RDMA 的通信设备，并与轻型输入/输出（LWIO）客户机一服务器协议关联。协议通常包括发现阶段及随后的 I/O 处理阶段。在发现阶段，客户机和服务器确定共享的能进行 RDMA 的提供者的最小列表。在 I/O 处理阶段，客户机发送 I/O 请求以便下传到第二机器。

在发现阶段，客户机最初从服务器器获得服务器请求继续键(resume key)。客户机随后打开通向服务器的管道，客户机通过此管道发送包含第一机器上能进行 RDMA 的提供者列表的协商请求。服务器通过该管道发送包含第二机器上匹配第一机器上的提供者的可用提供者的列表的协商响应。然后客户机创建通过共享的提供者到服务器的 RDMA 连接。客户机及服务器互相认证新的连接。客户机然后注册一个或多个文件为服务器使用。

I/O 处理请求消息包括关闭消息、取消消息、读消息、写消息、向量化读消息、以及向量化写消息。为安全原因，协议表现非对称 I/O 模型。读数据使用 RDMA 写操作被发送到客户机，而写使用常规的发送来完成。由客户机指定读和写请求，让服务器以轮询模式或中断模式完成。若客户机表示，不应以轮询模式完成，则服务器借助通过 RDMA 传输发送状态块到第一计算机来完成 I/O 处理请求。若客户机表示应该以轮询模式完成，则客户机可请求在完成 I/O 之后服务器通过中断请求消息来唤醒它。

按本发明的另外方面，提供以 I/O 下传协议管理缓存器的方法。该方法涉及使用缓存器信用点(credit)机制。服务器一客户机信用点事务包括由服务器发动并完成的三路握手。服务器发送一增量信用点消息给客户机，包括设置成信用点数的信息字段。若该数为负数-N，则客户机必须放弃 N 个信用点。

本发明的另外方面包括作为计算机程序产品及数据结构体现在计算机可读介质的上述特征。

## 附图说明

虽然附后的权利要求以细节列出了本发明的特征，从下面结合附图的详细描述能更好地理解本发明及其目标和优点，附图中：

图 1 是一般示出示例性客户机—服务器计算环境的原理图，它包括两台能通过 RDMA 传输通信的计算机，该环境能加入本发明的各方面；

图 2 是按本发明的实施例一般示出的在 LWI0 协议的发现阶段中采取的初始步骤的流程图；

图 3 是按本发明的实施例一般示出示例性服务器请求继续键的表示的概略图；

图 4A 是按本发明的实施例一般示出示例性客户机协商请求消息的表示的概略图；

图 4B 是按本发明的实施例一般示出示例性服务器协商响应的表示的概略图；

图 5 是按本发明的实施例一般示出在 LWI0 协议的发现阶段采取的附加步骤的流程图；

图 6A 是按本发明的实施例一般示出示例性客户机认证请求消息的表示的概略图；

图 6B 是按本发明的实施例一般示出示例性服务器认证响应的表示的概略图；

图 6C 是按本发明的实施例一般示出完成认证的示例性服务器状态响应的表示的概略图；

图 7A 是按本发明的实施例一般示出示例性客户机注册文件消息的表示的概略图；

图 7B 是按本发明的实施例一般示出完成文件注册的示例性服务器状态响应的表示的概略图；

图 8 是按本发明的实施例一般示出对于以轮询模式和以非轮询模式完成 I/O 请求采取的步骤的流程图；

图 9A 是按本发明的实施例一般示出示例性客户机中断请求消息的表示的概略图；

图 9B 是按本发明的实施例一般示出完成中断请求的示例性服务器状态响

---

应的表示的概略图；

图 10 是按本发明的实施例一般表示关于服务器—客户机信用点事务采取的步骤的流程图；

图 11A 是按本发明的实施例一般表示示例性服务器增量信用点消息的表示的概略图；

图 11B 是按本发明的实施例一般表示示例性客户机—服务器信用点消息的表示的概略图；

图 11C 是按本发明的实施例一般表示完成客户机—服务器信用点事务的示例性服务器状态响应的表示的概图；

图 12A 是按本发明的实施例一般示出示例性客户机关闭请求消息的表示的概略图；

图 12B 是按本发明的实施例一般示出完成关闭请求的示例性服务器状态响应的表示的概略图；

图 13A 是按本发明的实施例一般示出示例性客户机取消请求消息的表示的概略图；

图 13B 是按本发明的实施例一般表示完成取消请求的示例性服务器状态响应的表示的概略图；

图 14A 是按本发明的实施例一般示出在非轮询模式情况下的示例性客户机读请求消息的表示的概略图；

图 14B 是按本发明的实施例一般示出在非轮询模式情况下完成读请求的示例性服务器状态响应的表示的概略图；

图 14C 是按本发明的实施例一般示出在轮询模式情况下示例性客户机读请求消息的表示的概略图；

图 14D 是按本发明的实施例一般示出在轮询模式情况下完成读请求的示例性服务器 I/O 状态块的表示的概略图；

图 15A 是按本发明的实施例一般示出在非轮询模式情况下示例性客户机写请求消息的表示的概略图；

图 15B 是按本发明的实施例一般示出在非轮询模式情况下完成写请求的示例性服务器状态响应的表示的概略图；

图 15C 是按本发明的实施例一般示出在轮询模式情况下示例性客户机写请求消息的表示的概略图；

图 15D 是按本发明的实施例一般示出在轮询模式情况下完成写请求的示例性服务器 I/O 状态块的表示的概略图；

图 16A 是按本发明的实施例一般示出在非轮询模式情况下示例性客户机向量化读请求消息的表示的概略图；

图 16B 是按本发明的实施例一般示出在非轮询模式情况下完成向量化读请求的示例性服务器状态响应的表示的概略图；

图 16C 是按本发明的实施例一般示出在轮询模式情况下示例性客户机向量化读请求消息的表示的概略图；

图 16D 是按本发明的实施例一般示出在轮询模式情况下完成向量化读请求的示例性服务器 I/O 状态块的表示的概略图；

图 17A 是按本发明的实施例一般示出在非轮询模式、非崩溃情况下示例性客户机向量化写请求消息的表示的概略图；

图 17B 是按本发明的实施例一般示出在非轮询模式、崩溃情况下示例性客户机向量化写请求消息的表示的概略图；

图 17C 是按本发明的实施例一般示出在轮询模式、崩溃情况下示例性客户机向量化写请求消息的表示的概略图；

图 17D 是按本发明的实施例一般示例在非轮询模式情况下完成向量化写请求的示例性服务器状态响应的表示的概略图； 和

图 17E 是按本发明的实施例一般示出在轮询模式情况下完成向量化写请求的示例性服务器 I/O 状态块的表示的概略图。

### 具体实施方式

下面参考图 1~17E 来讨论本发明的某些实施例。然而本领域的技术人员容易理解，这里给出的结合附图的详细描述是为说明目的，本发明超越这些实施例而延伸。

图 1 是一般示出在其中可加入本发明的各方面的代表性网络化客户机/服务器环境的某些特征的原理图。在图 1 中画出两台计算机，标记为主机 A 101 和主机 B 121。虽然本发明可在包括许多不同类型和用途的计算机的环境中实施，然而在一个代表性情况下，主机 A 101 起着如数据库服务器等担负 I/O 密集型工作的应用程序服务器的作用。

每个主机 A 101 和主机 B 121 包括若干网络接口卡（NIC）109、111、113、

133、135、137，允许从一个机器到另一机器的网络化数据通信。在这些 NIC 中 NIC 109、111、135、137 允许 RDMA 数据传输。如图示，在两主机 101、121 之间给出非 RDMA 网络链路 119 和 RDMA 通道 117。

在主机 A 101 上执行的是 LWIO 客户机应用程序程序 103，它与负责处理 I/O 任务的应用程序关联，而 I/O 任务与核心模式 I/O 读/写服务器 105 交互。使用 LWIO 客户机 103 程序将 I/O 处理从主机 A 101 下传到主机 B 121。在主机 B 121 上执行 LWIO 服务器 123。按这里描述的 LWIO 协议，LWIO 客户机 103 与 LWIO 服务器 123 通信。LWIO 客户机 103 和 LWIO 服务器 123 使用发送的缓存器 107、127，使与文件关联的数据借助 RDMA 通道连接 117 直接传输。借助 LWIO 协议消息，可将读和写的任务下传到主机 B 121。服务器 123 将 I/O 请求传送到文件系统 129，后者作为到硬盘 131 的接口。

通常，两类消息与 RDMA 连接 117 相关。第一类是普通的网络发送/接收，它在目标机器上生成一中断。第二类是 RDMA 读/写，其中不需远程 CPU 的帮助来访问远程机器中的存储空间，因此不必生成中断。远程 CPU 确定为 RDMA 展现的存储器区域，但通常不知道何时完成 RDMA 操作。

在这里描述的本发明的实施例中，结合如 SMB 或 CIFS 等另一网络协议使用 LWIO 协议，以便利用其它协议的现有安全和认证基础结构。这帮助最小化 LWIO 协议的开销。如图 1 所示，主机 B 121 上的 LWIO 服务器 123 在 SMB 服务器 125 之上操作。SMB 客户机（未示出）类似地在主机 A 101 上运行，并与 LWIO 客户机应用程序程序 103 交互。

LWIO 协议包括两个阶段：发现阶段及随后的 I/O 阶段。与这里描述的实施例相关的数据结构、数据长度如下表示：

BYTE	无符号 8 位整数
CHAR	8 位 ASCII 码字符
UINT16	无符号 16 位整数
NINT32	无符号 32 位整数
UINT64	无符号 64 位整数
INT16	带符号 16 位整数
INT32	带符号 32 位整数
INT64	带符号 64 位整数
WCHAR	16 位统一码（Unicode）字符

---

PVOID32	32 位指针
---------	--------

PVOID64	64 位指针
---------	--------

图 2 示出在本发明的实施例中在 LWIO 协议的发现阶段采取的步骤。对于在其上执行 LWIO 服务器的主机，在步骤 201，LWIO 服务器向在主机上运行的 SMB/CIFS 服务器注册。按此注册，在步骤 203，SMB/CIFS 服务器通知在 LWIS 服务器可用的远程主机上运行的 SMB/CIFS 客户机。在步骤 205，LWIO 客户机请求一服务器请求继续键。继续键是一种认证机制，它已在与本专利具有同一受让人的另一专利中揭示，该美国专利于 2003 年 10 月 24 日提交，序列号：\_\_\_\_\_，名为“Method and System for Accessing a File (Resume KeY)（访问文件的方法和系统（继续键））”，该专利通过引用整体结合于此。

在步骤 207，LWIO 服务器将服务器请求继续键送回客户机。在本发明的一实施例中，服务器请求继续键具有下列结构

```

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;

```

图 3 提供服务器请求继续键 219 的图示。在服务器上生成 ResumeKey(继续键)221、Timestamp(时间标记)223 和 Pid(进程 id)225，它们对客户机是不透明的。Context(上下文)229 是包含由 LWIO 客户机使用来联系服务器的 UNC 名字的数组。ContextLength(上下文长度)227 是在 Context 229 中字节数。

## 网络发现

当客户机应用程序接收服务器请求继续键 219，它从 Context 字段 229 检索服务器 UNC 名字。回到图 2，在步骤 209，客户机打开通向 LWIO 服务器的管道。管道用于以下述方式对在网络上可用的能进行 RDMA 的设备的自动发现。这是本发明重要且有用的特征；通常 VIA 网络和类似网络缺少如 ARP 等地址解析机制。

客户机接着查询服务器以寻找其可用于 LWIO 协议的能将进行 RDMA 的设备（“提供者”）的列表。借助客户机构造并经在步骤 211 新开的管道发送到服务器

的协商请求来完成查询。在本发明的实施例中，协商请求有如下结构：

```

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

```

图 4A 提供在本发明实施例中的协商请求包 231 的图示。协商请求包括控制首部 233、固定长度的统一代码（Unicode）客户机名字段 235、用作密钥的客户机 UUID 237、用于接收响应的本地缓存器长度 239、以及提供者列表 241。在控制首部 233，ProtocolID（协议 ID）‘LWIO’ 243 被储存为首部的前 4 个字节。

RevId 245 保存当前定义的值 0x1001、LWIO\_REV\_ID。Opcode（操作码）247 保存当前定义的值 0xfe, LWIO\_CONTROL\_OPCODE\_NEGOTIATE。Length（长度）249 是包括所有操作码专用数据的拟发送给服务器的整个包的字节数。

服务器使用 ClientName（客户机名）235 来标识客户机。如下所述，Key（密钥）237 用于随后的网络专用认证过程。如下所述，ResponseLength（响应长度）

239 是用于从服务器接收协商响应的缓存器长度。ProviderCount（提供者计数）  
251 是与客户机相关联并客户机正通知的服务器的提供者者的数量。提供者者列表  
241 包含 ProviderCount 提供者的列表。

在提供者列表 241 的元素中，Name（名字）253 是提供者的名。为了检测兼容  
网络，客户机和服务器最好对同一提供者使用同一名。InstanceCount（实例计数）  
255 是特定提供者类型的设备数，实例表 257 是网络/鉴别器对的表，其中一对服  
务器以设备专用方式描述如何形成远程连接。HostAddressLen（主机地址长度）259  
是网络专用主机地址 263 的长度。DiscriminatorLen（鉴别器长度）261 是网络专  
用鉴别器 265 的长度。这些长度字段之后是主机地址 263 的 HostAddressLen 字节  
和鉴别器 265 的 DiscriminatorLen 字节。

回到图 2，在接收到与提供者的客户机列表的协商请求之后，在步骤 213，服  
务器确定它与客户机具有哪些共同的能进行 RDMA 通信的设备。在步骤 215，服务  
器经该管道发送包括共享提供者列表的协商响应到客户机。在本发明的一个实施例  
中，协商响应具有如下结构：

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

图 4B 提供在本发明的实施例中协商响应 267 的图示。控制首部 269 与协商请  
求的一样，不同处是 Length 271 现在反映响应消息 267 的长度。SrvName（服务  
器名）273 保存服务器的名。Key（关键字）275 是服务器生成的 GUID，为客户机使  
用。如下进一步解释，客户机在认证请求中通过使用一个共同通信设备的新连接将  
Key 送回给服务器。ProviderCount 277 是在提供者列表 279 中提供者的数量。提  
供者列表 279 包含对服务器及客户机共同的提供者的列表。不保证客户机实际上能  
连接到这些提供者。

回到图 2，此时服务器和客户机共享通信设备信息，并已确定共同提供者的最  
小列表。在步骤 217，客户机通过一个或多个共享设备创建一个或多个到 LWIO 服  
务器的 RDMA 连接。在本发明的一个实施例中，如此处描述，对客户机-服务器的通  
信定义下列操作码：

---

#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

以下定义的标志用作在客户机-服务器通信中的修改符:

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

在 LWIO 协议的对应客户机-服务器消息表征一共同的头部结构。在本发明的实施例中，共同的头部具有以下格式：

```

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;

    // 数据缓存器块
    struct {

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

---

```

        UINT16                               PageSize;
    } Vec;
};

// io状态块
union {
    struct {
        UINT32                           IosMh;
        PVOID64                          IosVa;
    };
    struct {
        UINT32                           ImmediateCookie;
        UINT64                           Cookie;
    };
} LWIO_COMMON_HEADER;

```

## 连接认证

图 5 示出在本发明的实施例中在 LWIO 协议的初始阶段余下部分由客户机和服务器采取的步骤。在步骤 601，如上所述客户机建立通过共享通信设备到服务器的连接。现在客户机和服务器互相认证新的连接。在步骤 603，客户机发送认证请求消息 (LWIO\_OPCODE\_AUTH) 到服务器。为防止服务器端和客户机端免受欺骗而作认证。若认证不能按时完成，连接终止。

图 6A 提供在本发明的实施例中客户机认证请求消息的图示。认证消息 617 包括共同首部 619 及随后的 LWIO\_AUTH\_PARAMS 结构 621。在首部 619 中，Length 623 被设置成发送到服务器的字节数(共同首部 619 的长度加上 LWIO\_AUTH\_PARAMS 621 的长度)。Opcode 625 被设为 LWIO\_OPCODE\_AUTH(0 × 6)。标志 627 被设成 LWIO\_HDR\_FLAG\_INTERRUPT。在这个和其它客户机协议消息中，Cookie 629 被设成由客户机选择的值，并在服务器应答中被送回。Cookie 值通常被用于将请求与服务器应答匹配。DataVa 631 被设成服务器应当对服务认证参数进行 RDMA 的地址。DataMh 633 保存与 DataVa 631 关联的 RDMA 存储器句柄。

在本发明的实施例中，LWIO\_AUTH\_PARAMS 结构具有如下格式：

---

```

#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;

```

在认证消息 617 中，LWIO\_AUTH\_PARAMS 621 构成包的第二部分。Magic 635 设成 ‘LWIO’。RevId 627 设成 LWIO\_REV\_ID。Endian 639 设成 sizeof(ULONG\_PTR)。PageSize 641 设成 CPU 页面大小（在 32 位机器上为 4K，在 64 位机器上为 8K）。BaseSequence643 设成 0。MaxRdmaWindowSize 645 试图设成客户机在一次 RDMA 传输中可接受的最大字节数；在图示的实施例中设成 64K。MaxSendBufferSize 647 试图设成客户机在单次请求中可向服务器发送的字节数；在图示的实施例中被设成 1K。MaxRecvBufferSize 649 试图设成客户机为了从服务器接收数据而发送的字节数；在图示的实施例中它被设成 16 字节。HeaderSize 651 被设成在 LWIO 控制首部 619 中的字节数。Credits 652 设成客户机希望具有的缓冲器信用点的初始数量。信用点的使用如下解释。服务器可以满足也可以不满足客户机的请求。若客户机不支持 RDMA 读操作，RdmaReadSupport 653 设成 0，若客户机支持 RDMA 读，则设成 1。

LWIO\_AUTH\_PARAMS 结构的部分是一个或多个选项的组。使用选项使认证更灵

活。除了在列表中的最后选项 LWIO\_AUTH\_OPTION\_END 外，每个选择具有选项码、长度和数据，该最后选项只有选项码，作为终止选项列表的空选项。在认证消息中，客户机发送给服务器下列选项：密钥（LWIO\_AUTH\_OPTION\_KEY）和签名（LWIO\_AUTH\_OPTION\_SIGNATURE）。Key 655 设成服务器以前在协商响应中返回的密钥。Signature 657 是除签名以外的 LWIO\_AUTH\_PARAMS 621 的 MD5 签署。

回到图 5，在步骤 605，若在认证消息中发送的 Key 匹配通过管道在协商响应中返回的密钥，则服务器作为认证响应向客户机 RDMA 发送 LWIO\_AUTH\_PARAMS 结构，它包括由客户机在认证消息中向 DataVa 地址及关联的 DataMh 存储器句柄提供的 8 字节 SessionId。在步骤 607，服务器发送 LWIO\_MSG\_STATUS\_RESPONSE 以完成认证。

图 6B 提供在本发明的实施例中由服务器返回的 LWIO\_AUTH\_PARAMS 结构 659 的图示，Magic 611 设成 ‘LWIO’。RevId 663 设成 LWID\_REV\_ID。Endian 655 设成 sizeof(ULONG\_PTR)。PageSize 667 设成 CPU 页大小。BaseSequence 669 试图设成（客户机的 BaseSequewcet1）。MaxRdmaWindowSize 671 试图设成客户机在 RDMA 传输中能接受的最大字节数；在描述的实施例中设成 512K。MaxSendBufferSize 673 试图设成服务器在单个响应中发送给客户机的字节数；在描述的实施例中设成 16 字节。MaxRecvBufferSize 675 试图设成为从客户机接收数据服务器预发送的字节数；在描述的实施例中设成 8K。HeaderSize 667 设成在共同首部的字节数。Credits 679 被设成服务器对客户机可用的初始信用点数。若服务器不支持 RDMA 读，则 RdmaReadSupported 681 被设成 0，若服务器支持 RDMA 读，则设成 1。服务器发送下列选项：Key(LWIO\_AUTH\_OPTION\_Key) 683、SessionId(LWIO\_AUTH\_OPTION\_SESSION\_ID) 685 和 Signature(LWIO\_AUTH\_OPTION\_SIGNATURE) 687。Key 683 被设成客户机以前在协商请求中发送的 Key。如下解释，客户机在向服务器注册客户机文件时使用该 SessionId 685 的值。Signature 687 是 LWIO\_AUTH\_PARAMS 中除 Signature 外的 MDS 签署。

在本发明 MSG\_STATUS\_RESPONSE 的实施例中，LWIO\_MSG\_STATUS\_RESPONSE 结构具有下述格式：

```

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;

```

图 6C 提供在本发明的实施例中为完成认证由服务器返回的 LWIO\_MSG\_STATUS\_RESPONSE 689 的图示。Cookie 691 被设成由客户机在认证消息的首部中设置的 Cookie 值。Information（信息）693 被设成 LWIO\_AUTH\_PARAMS 的字节数加 8 个字节。Status 695 设成 0×0（表示成功）或 0×C0000022（表示“访问被拒绝”）。

## 文件注册

回到图 5，在步骤 609，当新的连接被客户机及服务器互相认证时，客户机开始注册文件以供服务器使用。

图 7A 提供在本发明的实施例中由客户机向服务器发送的注册文件消息的图示。注册消息 701 包括共同首部 703 及随后的 LWIO\_FID\_PARAMS 结构 705。Length（长度）设成发送到服务器的字节数（首部 703 的长度加上 LWIO\_FID\_PARAMS 705 的长度）。Opcode（操作码）709 设成 LWIO\_OPCODE\_REGISTER（0×7）。Flags（标志）711 设成 LWIO\_HDR\_FLAG\_INTERRUPT。在此客户机消息及随后的客户机消息中，Credits（信用点）713 设成在客户机上挂起的 I/O 请求的数量。如下进一步解释，Credits 字段用作对服务器向连接分配更多信用点的暗示，从而允许额外的未完成的 I/O 请求。在任何时刻未完成的客户机请求的数量不能超过“Credits”值。如前所述，Cookie 715 设成客户机指定的值。

在本发明的实施例中，LWIO\_FID\_PARRMS 结构具有下述格式：

```

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

```

在注册文件消息 701 的 LWIO\_FID\_PARRMS 705 中，ResumeKey 717 设成通过初始文件访问通道返回的服务器请求继续键。SessionId 719 设成在连接认证阶段中

由服务器返回的 SessionId。FlagsAndAttributes（标志和属性）721 设成最初用于打开文件的 Win32 创建标志。

回到图 5，在步骤 611，服务器用 LWIO\_MSG\_STATUS\_RESPONSE 作出响出以完成文件注册。图 7B 提供在本发明中由服务器发送的 LWIO\_MSG\_STATUS\_RESPONSE 723 的图示。Information727 设成 Fid(文件 ID)，在发送 I/O 请求时使用。Status (状态) 729 设成 0×0 (成功) 或在失败时设成另外的 NTSTATUS 码。Cookie 725 设成客户机在注册文件消息的首部中的 Cookie 值。

## I/O 处理

在此时，建立了客户机连接，且文件已被注册，并开始 LWIO 协议的 I/O 处理阶段。LWIO 协议的实施例的一个关键特征是对读和写的非对称 I/O 模型。读操作使用 RDMA 实现，而写使用发送操作实现。不使用 RDMA 实现写是为了提供更好的安全模型。若服务器对 RDMA 展现 NIC 上的地址空间，它会引入能由恶意客户机利用的数据破坏的弱点。在此情况，恶意的客户机在循环中发出给定服务器虚拟地址上的 RDMA 写操作。由于服务器地址空间是有限的，且在某一点服务器的虚拟地址必须被重新使用，因此恶意客户机最终捕捉服务器，对不同的连接使用同一虚拟地址，导致数据被写入到可能与不同的客户机关联的服务器缓存器。LWIO 协议中的非对称 I/O 模型避免了这种可能性。此特征是 LWIO 协议和如 DAFS 等其它基于 RDMA 的文件传输协议之间的主要差别。

回到图 5，在步骤 613，客户机开始发送 I/O 处理请求。服务器-客户机的 I/O 请求的完成是非轮询模式或轮询模式。在非轮询模式中，I/O 完成是基于中断的，使用普通的发送/接收消息。在轮询模式中，I/O 的完成是使用 RDMA，且不是基于中断的。

从 LWIO 服务器的一般观点来看，图 8 的流程图一般示出在本发明的实施例中以轮询模式或非轮询模式完成 I/O 请求所采取的步骤。客户机 I/O 请求指定，服务器是否应当发回后发送 (post\_send) (中断 CPU) 或 RDMA 消息。在步骤 801，服务器确定是否在客户机 I/O 请求消息的共同首部中的 LWIO\_HDR\_FLAG\_INTERRUPT 标志是否被置位。若该标志被置位，则在步骤 803，服务器使用普通发送借助 LWIO\_MSG\_STATUS\_RESPONSE 完成客户机请求。若 LWIO\_HDR\_FLAG\_INTERRUPT 标志未被置位（轮询模式），则如步骤 805 所示，服务器通过用 RDMA 发送 LWIO\_ID\_STATUS\_BLOCK 到客户机而完成客户机请求。

## 在轮询模式中唤醒客户机

在轮询模式中，客户机希望在等待从服务器完成 I/O 的同时休眠。在此情况借助 RDMA 将 Completion (完成) 发送到客户机，所以需要一种机制来唤醒客户机，通知它完成已发生。若客户机希望被唤醒，它发送中断请求 (LWIO\_OPCODE\_INTERRUPT) 消息到服务器，由服务器在图 8 的步骤 807 接收。接收中断请求的服务器在服务器完成 I/O 请求之前不发送响应 (步骤 809)。在步骤 811 借助普通发送，将“完成”发送到客户机，中断客户机。对给定客户机连接只有一个中断消息能是未完成的。

图 9A 提供在本发明的实施例中由客户机发给服务器的中断请求消息的图示。消息包括共同的首部 815。Opcode (操作码) 817 被置成 LWIO\_OPCODE\_REGISTER (0 × 9)。标志 819 被置成 (LWIO\_HDR\_FLAG\_INTERRUPT1 LWIO\_HDR\_FLAG\_CONTROL) (0 × C0)。Credits (信用点) 821 被置成在客户机上挂起的 I/O 请求的数量，而 Cookie 823 被设置成客户机指定的值。

在另一 I/O 请求已被处理之后服务器响应中断请求消息。图 9B 提供了在本发明的实施例中由服务器发送的 LWIO\_MSG\_STATUS\_RESPONSE 消息 825 的图示。Information (信息) 827 设成 0。Status (状态) 829 设成 0 × 0 (成功)，或在失败时设成另一 NTSTATUS 码。Cookie 831 设成由客户机发送的中断请求的首部中的 Cookie 值。

## 信用点 (Credits)

已经提到，所有客户机-服务器的 I/O 请求包括在首部的信用点字段。信用点字段是对服务器关于客户机发送给服务器的未完成的 I/O 信用点数的暗示。管理信用点是服务器的责任。信用点提供对转储清除缓存器问题的新颖解决方法。若客户机当前具有 N 个信用点，为了服务器向客户机发送信用点消息，需要发送 N+1 个接收缓存器。在任何一个时刻服务器沿客户机连接只有一个未完成的信用点请求。信用点消息总是以中断模式发送。

信用点事务包括在客户机和服务器之间由服务器启动的三路握手。图 10 一般示出包括在本发明的实施例中信用点事务的步骤。在步骤 1001，服务器沿客户机连接发送增量信用点请求消息。

图 11A 提供在本发明实施例中服务器增量信用点消息的图示。此消息采取

LWI0\_MSG\_STATUS\_RESPONSE 1011 的形式。信用点对应于缓存器。信息 1013 设成客户机应放弃的信用点的数（负数）或服务器新分配给客户机使用的信用点（额外的缓存器）的数量（正数）。Status 1015 设成 LWI0\_NOTIFY\_CREDIT(0×1)。Cookie 1017 设成 0。

回到图 10，客户机从服务器接收信用点消息。需要客户机在同一连接上用 LWI0\_OPCODE\_CREDIT 消息对服务器作出响应。此消息表示释放单个信用点或通知服务器客户机已使用的新分配的信用点数。若在服务器信用点消息中的信息字段包含负数-N（步骤 1003），则如步骤 1005 表明，客户机发送 N 个 LWI0\_OPCODE\_CREDIT 消息（对每个需要放弃的信用点一个）。若信息字段是正，则如步骤 1007 表明，客户机只发送一个 LWI0\_OPCODE\_CREDIT 消息。

图 11B 提供在本发明的实施例中由客户机发送的 LWI0\_OPCODE\_CREDIT 消息的图示。LWI0\_OPCODE\_CREDIT 消息 1019 包括共同的首部 1021。OPCODE 1023 设成 LWI0\_OPCODE\_CREDIT(0×8)。Flags（标志）1025 设成 LWI0\_HDR\_FLAG\_INTERRUPT(0×80)。Credits（信用点）1027 设成在客户机上挂起的 I/O 请求的数量。Cookie 1031 设成客户机指定的值。若客户机接收到正的增量信用点消息，则 Offset（偏置）1029 的较高 32 位设成由服务器分配而客户机未使用的信用点的数量。一旦客户机在此字段返回大于零的值，在发送至少一个负的更新值之前，服务器通常不发送另一正的更新消息。通常客户机返回零。

如上提到，若客户机接收负（-N）增量信用点消息，则需要客户机发送 N 个信用点消息给服务器，对每个放弃的信用点一个。在此情况 Offset 1029 的较高 32 位相应地设成 -N, -(N-1), …, -1。当服务器接收其 Offset 1029 的较高 32 位被设成 -1 的客户机信用点消息时，服务器认为客户机已完成处理服务器信用点消息，并适于接收新的信用点消息。

回到图 10，如步骤 1009 所示，服务器通过发送 LWI0\_MSG\_STATUS\_RESPONSE 消息到客户机完成三路握手。图 11C 提供在本发明的实施例中由服务器发送的 LWI0\_MSG\_STATUS\_RESPONSE 1033 的图示。Information（信息）1037 设成 0。若在由客户机发送的 LWI0\_OPCODE\_CREDIT 消息的首部中 Offset（偏置）的较高 32 位大于或等于 0，则 Status 1039 设成 0×0，表示成功。若 Offset 的较高 32 位设成负数，则为了使客户机能收回信用点，服务器将 Status 1039 设成 LWI0\_CREDIT\_NOTIFY。Cookie 1035 设成由客户机在 LWI0\_OPCODE\_CREDIT 消息的共同首部中设置的 Cookie 值。

## 关闭

关闭消息用于停止对在注册阶段交换的特定 Fid 的 I/O 处理。一旦服务器作出响应，在 Fid 被回收之前任何新的请求将失败。图 12A 提供在本发明的实施例中由客户机发送的关闭消息的图示。关闭消息 1041 包括共同的首部 1043。Opcode(操作码) 1045 设成 LWIO\_OPCODE\_CLOSE(0×4)。Flags(标志) 1047 设成 LWIO\_HDR\_FLAG\_INTERRUPT(0×80)。Credits(信用点) 1049 被设成在客户机上挂起的 I/O 请求的数量。Cookie 1053 设成客户机指定的值。Fid 1051 设成拟关闭的文件的文件 id。

服务器用 LWIO\_MSG\_STATUS\_RESPONSE 作出响应。图 12B 提供在本发明实施例中由服务器返回的关闭完成 LWIO\_MSG\_STATUS\_RESPONSE 1055 的图示。Information(信息) 1059 设成 0。Status(状态) 1061 设成 0，表示成功。Cookie 1057 设成在客户机关闭请求中设置的 Cookie 值。

## 取消

使用取消消息停止对在注册阶段交换的特定 Fid 的 I/O 处理。在发出取消时，服务器程序完成请求。然而不能取消的 I/O 请求仍可在服务器上进行。图 13A 提供在本发明的实施例中由客户机发出的取消消息的图示。取消消息 1063 包括共同的首部 1065。Opcode 1067 设成 LWIO\_OPCODE\_CANCEL(0×5)。Flags(标志) 1069 设成 LWIO\_HDR\_FLAG\_INTERRUPT(0×80)。Credits(信用点) 1071 被设成在客户机上挂起的 I/O 请求的数量。Cookie 1075 设成客户机指定的值。Fid 1073 设成发出取消的文件的文件 id。

服务器用 LWIO\_MSG\_STATUS\_RESPONSE 消息完成取消。图 13B 提供在本发明的实施例中由服务器返回的取消完成 LWIO\_MSG\_STATUS\_RESPONSE 1077 的图示。Information(信息) 1081 设成 0。Status(状态) 1083 设成零，表示成功。Cookie 1079 设成在客户机取消请求中设置的 Cookie 值。

使用读消息从在注册阶段被交换的特定 Fid 中获取数据。对小于一千字节的读请求，若用户的缓存器未向 NIC 注册，则数据被接收到内部的预注册的缓存器中，且一旦从服务器接收数据，将复制完成到用户缓存器中。这样做是因为复制少量数据比注册小的用户缓存器更有效。对大的读，注册用户缓存器，并通过 RDMA 直接接收数据。按照单次读请求的读数据的量受服务器的 MaxRdmaWindowSize 限制。

图 14A 和 14C 提供在本发明的实施例中由客户机发送的读消息的图示，图 14A 给出非轮询情况而图 14C 给出轮询情况。读消息 1401 包括共同的首部 1403。长度 1405 设成拟从相关文件读的字节数。Opcode（操作码）1407 设成 LWIO\_OPCODE\_READ(0×0)。Offset（偏置）1417 设成文件读开始处的字节位置。Marker（标记）1413 设成 0×FF。Flags（标志）1409, 1427 在轮询情况 1427 设成 0 × 0，或在非轮询情况 1409 设成 LWIO\_HDR\_FLAG\_INTERRUPT(0×80)。Credits（信用点）1411 设成在客户机中挂起的 I/O 请求的数量。Fid 1415 设成发出 I/O 的文件的文件 id。DataVa 1419 设成要对读数据进行 RDMA 的位置，而 DataMh 1421 设成关联的存储器句柄。

在非轮询情况，ImmediateCookie 1423 和 Cookie 1425 设成用户程序指定的值。在此 LWIO\_MSG\_STATUS\_RESPONSE 的情况下服务器借助常规发送完成读请求，或若读成功，在 RDMA 的情况下用立即数据完成。因而 RDMA 写的立即数据设成读请求的 ImmediateCookie 值。在轮询情况下，IosVa 1431 设成对服务器响应状态（LWIO\_IO\_STATUS\_BLOCK）进行 RDMA 的位置，而 IosMh 1429 设成关联的存储器句柄。

在非轮询情况下，服务器首先对读数据进行 RDMA。然后，服务器用 LWIO\_MSG\_STATUS\_RESPONSE 作出响应，或服务器能用 RDMA 读数据发送立即数据，在此情况下立即数据设成读请求的 ImmediateCookie 值。图 14B 提供在本发明的实施例中由服务器在非轮询情况下返回的 LWIO\_MSG\_STATUS\_RESPONSE 1433 的图示。

Information（信息）1437 设成读的字节数，Status（状态）1439 设成 0，表示成功，或设成另外的 NTSTATUS，表示失败。Cookie 1435 设成由客户机在读消息的首部中设置的 Cookie 值。

在轮询情况下，服务器首先对读数据进行 RDMA。然后服务器用 RDMA 发送 LWIO\_IO\_STATUS\_BLOCK 到客户机。图 14D 提供在本发明的实施例中由服务器返回的 LWIO\_IO\_STATUS\_BLOCK 1441 的图示。Information（信息）1443 设成读的字节数。Status（状态）1445 设成 0，表示成功，或设成 NTSTATUS，表示失败。

## 写

使用写消息将数据放入在文件注册期间被交换的特定 Fid。使用普通的发送操作发送所有写数据。写的数据量受服务器 MaxRecvBufferSize 的限制。若客户机发送比此更多数据，连接终止。

图 15A 和 15C 提供在本发明的实施例中由客户机发送的写消息的图示，图 15A 给出非轮询情况而图 15C 给出轮询情况。写消息 1501 包括共同的首部 1503。长度 1505 设成拟写数据的字节数。Opcode 1507 设成 LWIO\_OPCODE\_WRITE(0×1)。Offset (偏置) 1517 设成开始写文件数据处的字节位置。Flags (标志) 1509、1529 在轮询情况 1529 被设成 0×0，或在非轮询情况 1509 设成 LWIO\_HDR\_FLAG\_INTERRUPT(0×8)。Marker (标记) 1513 设成 0×FF。Credit 1511 设成在客户机处挂起的 I/O 请求的数量。Fid 1515 设成在其上发出 I/O 的 File (文件) Id。拟写的数据 1527 紧接在写消息的共同首部 1503 之后。

在非轮询情况下，Cookie 1525 设成客户机指定的值，在轮询情况下，IosVa 1533 设成对服务器响应状态 (LWI0\_IO\_STTAUS\_BLOCK) 进行 RDMA 的位置，而 IosMh1531 设成相关联的存储器句柄。

在非轮询情况下，服务器用 LWIO\_MSG\_STATUS\_RESPONSE 响应写消息。图 15B 提供在本发明的实施例中由服务器返回的 LWIO\_MSG\_STATUS\_RESPONSE 1535 的图示。信息 1539 设成被写的字节数。状态 1541 设成 0，表明成功，或设成另一 NTSTATUS，表明失败。Cookie 1537 设成在写消息的首部由客户机设置的 Cookie 值。在轮询情况下服务器对 LWIO\_IO\_STATUS\_BLOCK 进行 RDMA。图 15D 提供在本发明的实施例中由服务器返回的 LWIO\_IO\_STATUS\_BLOCK 1543 的图示。Information (信息) 1545 设成被写的字节数。Status (状态) 1547 设成 0 表示成功，或设成另一 NTSTATUS 表示失败。

## 向量化读

向量化读用于从在注册阶段交换的特定 Fid 获取数据，并在页的基础上将数据散布到请求者的多个段。所有读数据借助 RDMA 写发送到请求者，对每个读的段从服务器进行一次 RDMA 写。从盘读出的数据是连续的。写的数据的量受到在单次请求中描述的目标页的最大数量的限制。此限制是由 sizeof(LWI0\_RDMA\_REGION) 划分的服务器 MaxRecvBufferSize。下面给出 LWIO\_REMA\_REGION 的结构。

图 16A 和 16C 提供在本发明的实施例中由客户机发送的向量化读消息的图示，图 16A 给出非轮询情况而图 16C 给出轮询情况。读消息 1401 包括共同首部 1603，接着是一个或多个 LWIO\_REMA\_REGION 段 1605、1607。在首部 1603，Length (长度) 1609 设成拟从文件读取的数据的字节数。Opcode 1611 设成 LWIO\_OPCODE\_VEC\_READ(0×2)。Offset (偏置) 1621 设成在开始读文件数据处的

字节位置。Flags (标志) 1613, 1631 在轮询情况 1631 设成  $0 \times 0$ , 或在非轮询情况 1613 设成 LWIO\_HER\_FLAG\_INTERRUPT ( $0 \times 80$ )。Marker (标记) 1617 设成  $0 \times FF$ 。Credits 1615 设成在客户机挂起的 I/O 请求的数量。Fid 1619 设成在其上发出 I/O 的文件 Id。NumPages 1623 设成跟在共同首部 1603 后的 LWIO\_RDMA\_REGION 的数量。PageSize 1625 设成本地页的字节数。

在非轮询情况下, ImmediateCookie 1627 和 Cookie 1629 被设成客户机指定的值。在 LWIO\_MSG\_STATUS\_RESPONSE 的情况下服务器借助常规发送能完成向量化读请求, 或若读成功, 在 RDMA 情况下用直接数据完成。因而 RDMA 写的立即数据设成读请求的 ImmediateCookie 1627 值。在轮询情况下, IosVa 1635 设成对服务器响应状态 (LWIO\_IO\_STATUS\_BLOCK) 进行 RDMA 的位置, 而 IosMh 1633 设成关联的存储器句柄。

共同的首部 1603 后面紧接足够数目的 LWIO\_RDMA\_REGION 段 1605、1607 以覆盖请求的长度。所有立即段必须是一页大小。最后段可小于一页, 但它必须是后端 (backend) 盘扇区大小的倍数。在本发明的实施例中, LWIO\_RDMA\_REGION 具有如下格式:

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

第一 LWIO\_RDMA\_REGION 对应于读的第一 PageSize 字节, 第二 LWIO\_RDMA\_REGION 对应于读的第二 PageSize 字节, 等等。DataVa 1637 设成标志放置读数据的页开始处的位置。DataMh 1639 设成 DataVa 1637 的存储器句柄。Length (长度) 1641 设成对除最后区域外所有区域的 PageSize 1625, 对最后区域长度可以小一些, 但必须是后端盘扇区大小的倍数。

在非轮询情况下, 服务器首先对读数据进行 RDMA。然而服务器能用 LWIO\_MSG\_STATUS\_RESPONSE 作出响应, 或用 RDMA 读数据发送立即数据, 在此情况, 立即数被设成读请求的 ImmediateCookie 值。图 16B 提供在本发明的实施例中非轮询情况由服务器返回的 LWIO\_MSG\_STATUS\_RESPONSE 1643 的图示。Information (信息) 1647 设成读字节数。Status (状态) 1649 设成 0, 表示成功, 或另一 NTSTATUS, 表示失败。Cookie 1645 设成在向量化读消息的首部由客户机设置的 Cookie 值。

在轮询情况下, 服务器首先对读数据进行 RDMA, 然后服务器对

LWIO\_IO\_STATUS\_BLOCK 进行 RDMA。图 16D 提供在本发明的实施例中由服务器返回的 LWIO\_IO\_STATUS\_BLOCK 1651 的图示。Information（信息）1653 设成读字节数。Status（状态）1655 设成 0，表示成功，或设成另一 NTSTATUS，表示失败。

## 向量化写

向量化写消息用于完成对在文件注册中交换的特定 Fid 的集成的写。所有写数据使用普通发送操作发送。写的数据量受服务器的 MaxRecvBufferSize 的限制。若客户机发送多于此的数据，连接终止。

图 17A、17B 和 17C 提供在本发明的实施例中由客户机发送的向量化写消息的图示，图 17A 示出非轮询不崩溃情况，图 17B 示出非轮询崩溃情况，图 17C 示出轮询崩溃情况。

写消息 1701 包括共同首部 1703，紧接着是拟写的数据 1705。在共同首部 1703，长度 1707 设成被写的数据的字节数。OpCode 1709 设成 LWIO\_OPCODE\_WRITE(0×3)。Offset（偏置）1719 设成在开始写文件数据处的字节位置。Marker（标记）1715 设成 0×FF。Credit（信用点）1713 设成在客户机上挂起 I/O 信用点。Fid 1717 设成发出 I/O 处的文件 Id。

Flags（标志）1711、1721、1727 设成 0×0，表示轮询 1727，或否则设成 LWIO\_HDR\_FLAG\_INTERRUPT(0 × 80)1711。在后面情况下，标志也能包括 LWIO\_HDR\_FLAG\_INTERRUPT 1721，以表明在写的所有页面包含同样数据，所以只发送单个页数据。这时试图使重复数据的传输最小化的优化。若注册的文件标志包括 FILE\_NO\_INTERMEDIATE\_BUFFERING(0×8)和在认证阶段交换的 PageSize 互相成偶数倍，则只能使用 LWIO\_HDR\_FLAG\_COLLAPSE。在崩溃的 I/O 的情况下，NumPage 1723 设成 I/O 覆盖的数据的页数。由于 Length（长度）参数，最后页面可以是部分的。PageSize 1725 设成本地页面字节数。在轮询情况下，IosVa 1731 设成拟用 RDMA 传输的服务器响应状态（LWIO\_IO\_STATUS\_BLOCK）的位置。IosMh 1729 与存储器句柄相关联。

在非轮询情况下，对非崩溃和崩溃的 I/O 情况，服务器用 LWIO\_MSG\_STATUS\_RESPONSE 对写消息作出响应。

图 17D 提供在本发明的实施例中由服务器返回的 LWIO\_MSG\_STATUS\_RESPONSE 1733 的图示。Information（信息）1737 设成写的字节数。Status（状态）1739 设成 0，表示成功，或设成另一 NTSTATUS，表示失败。Cookie 1735 设成由客户机

---

在写消息头部设置的 Cookie 值。

在轮询情况下，对于非崩溃和崩溃的 I/O，服务器对 LWIO\_IO\_STATS\_BLOCK 进行 RDMA。图 17E 提供在本发明的实施例中由服务器返回的 LWIO\_IO\_STATS\_BLOCK 1741 的图示。Information（信息）1743 设成写的字节数。Status（状态）1745 设成 0，表示成功，或设成另一 NTSTATUS，表示失败。

## 结论

虽然已图示及描述了本发明的图示实施例，然而可以理解，在不偏离本发明的情况可作出各种改变。类似地，这里描述的任何处理步骤可与其它步骤互换以达到同样结果。此外，上述的图示例子不是包括一切或试图将本发明限于描述的精确形式。相反，其目的是覆盖落在本发明的精神及范围内的所有修改、更换、结构和等价技术方案。

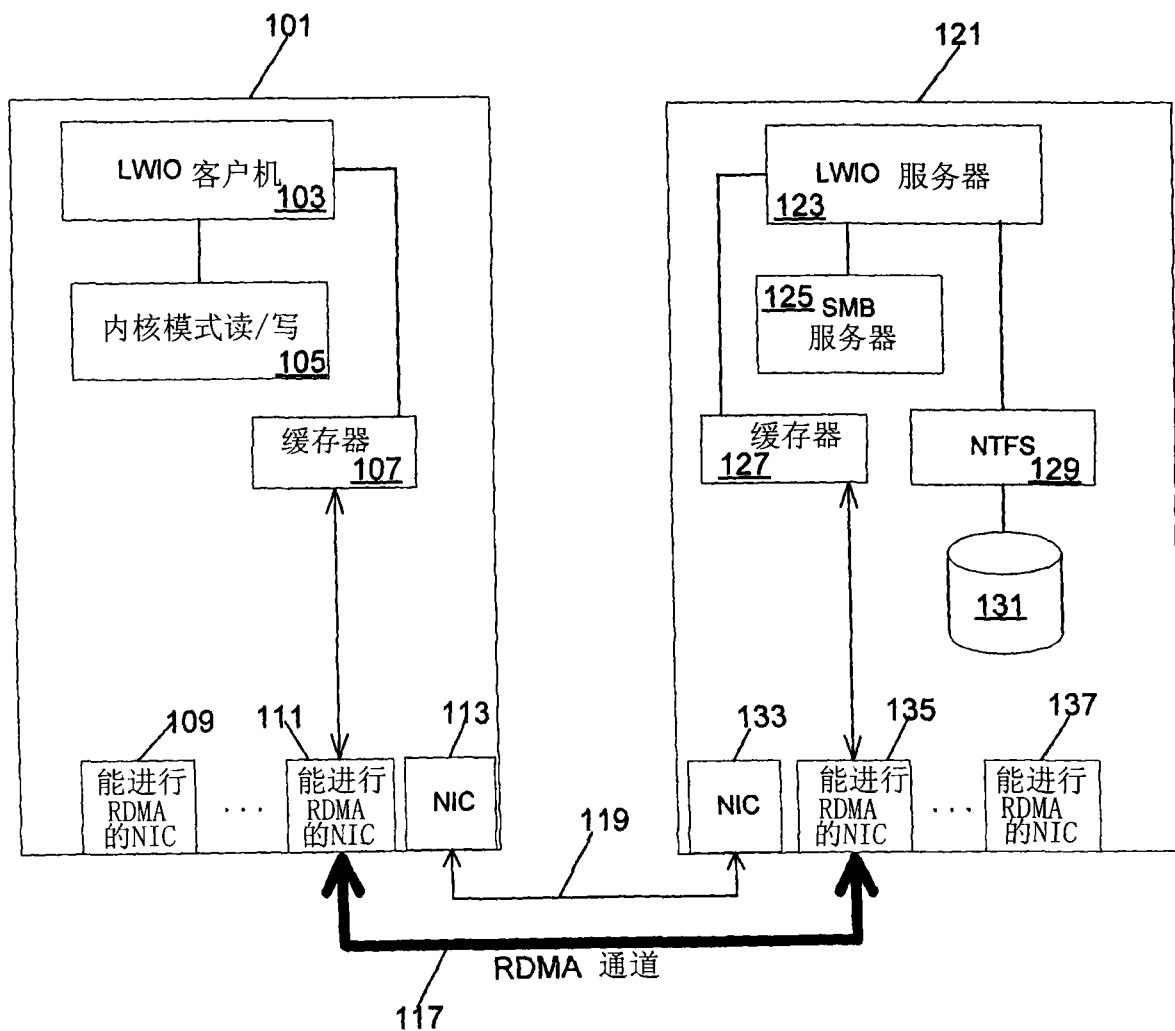


图 1

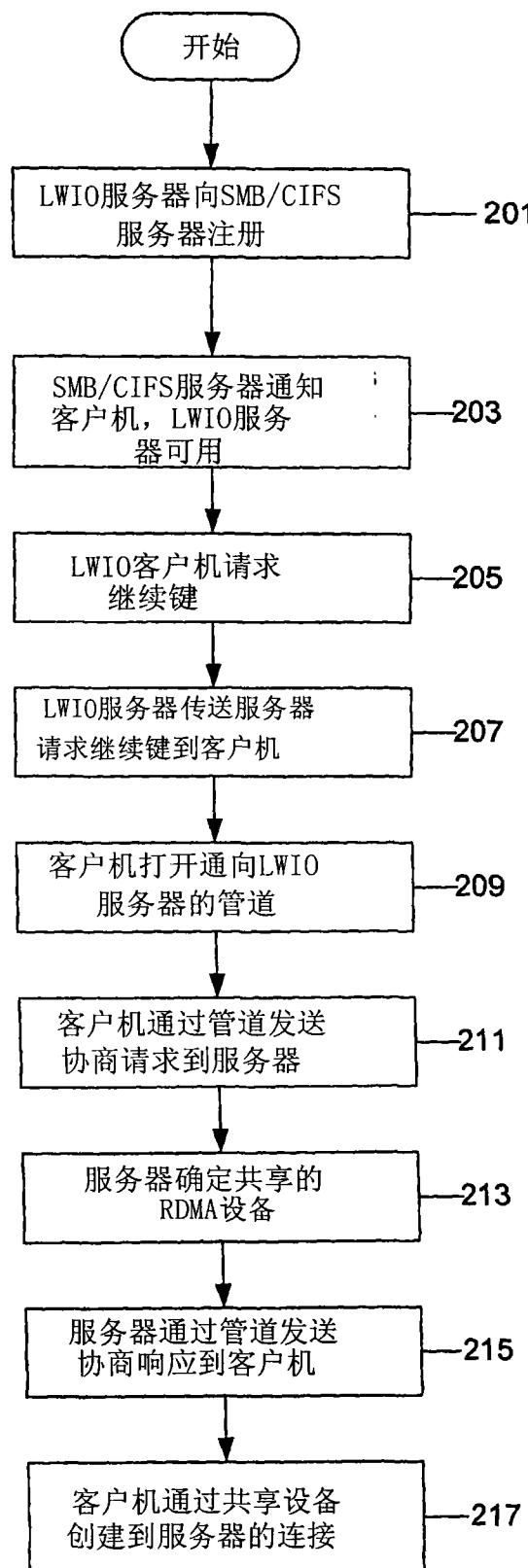


图 2

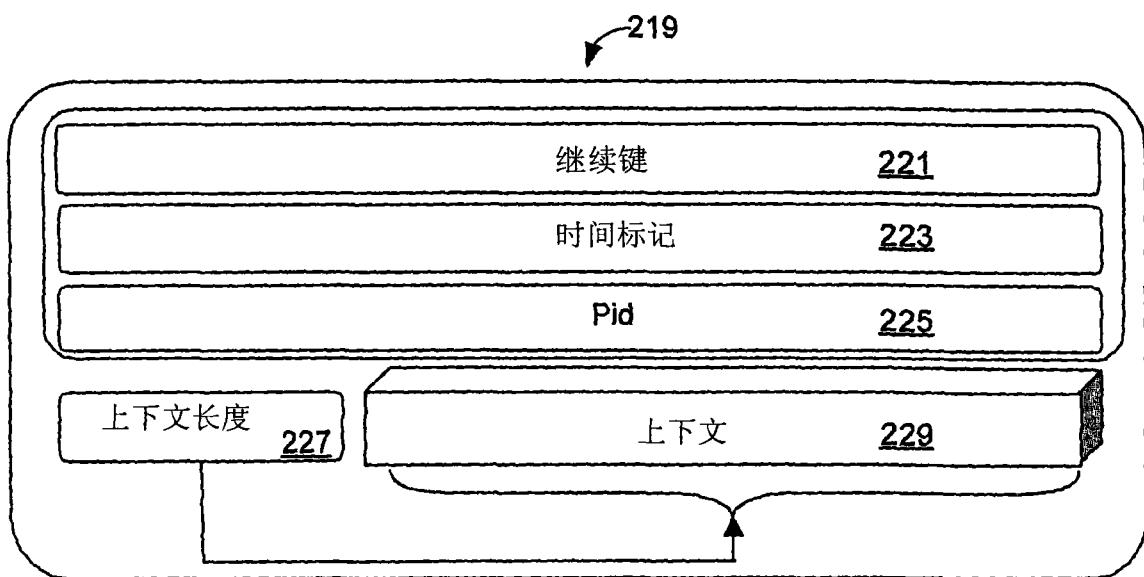


图 4

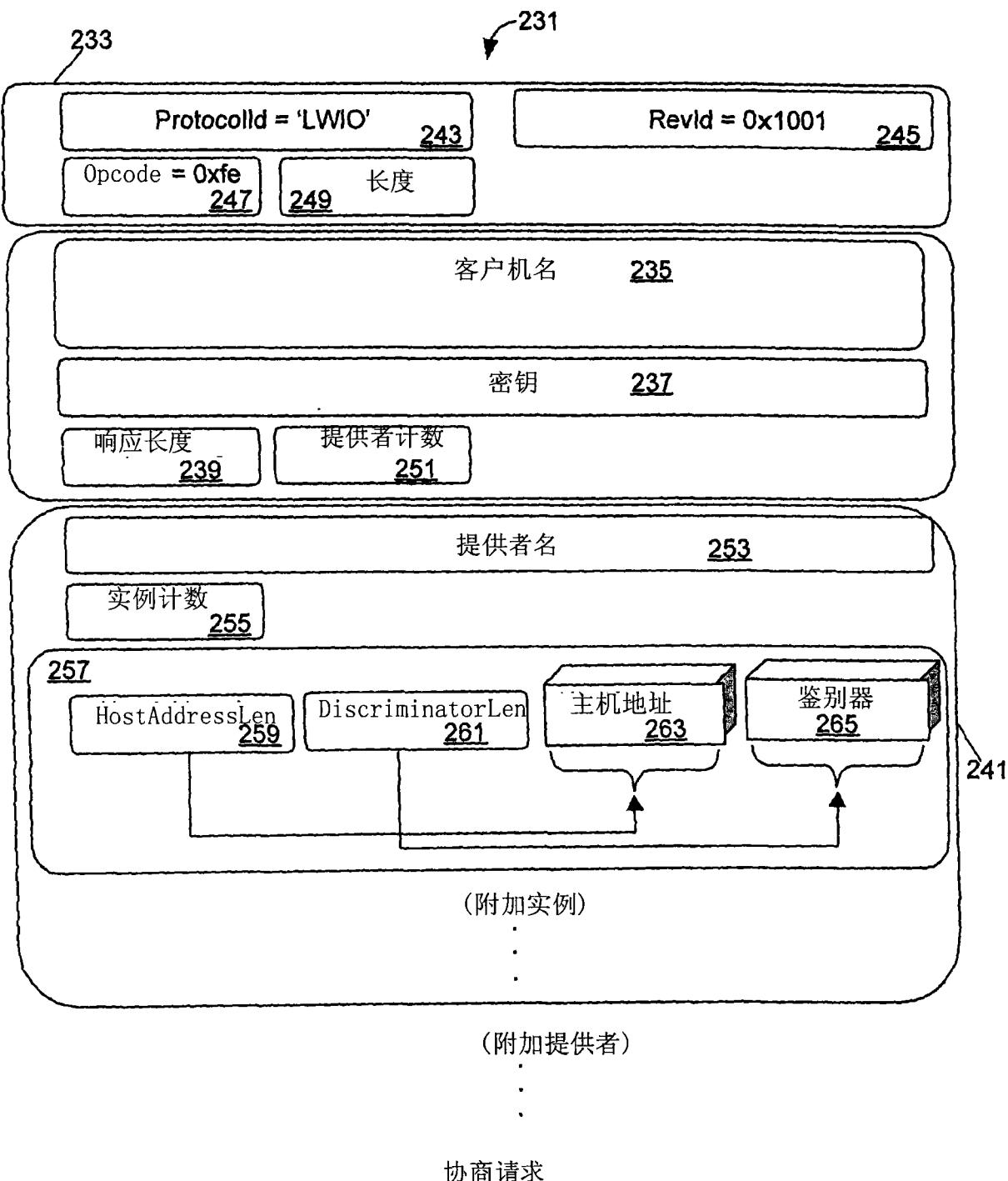


图 4A

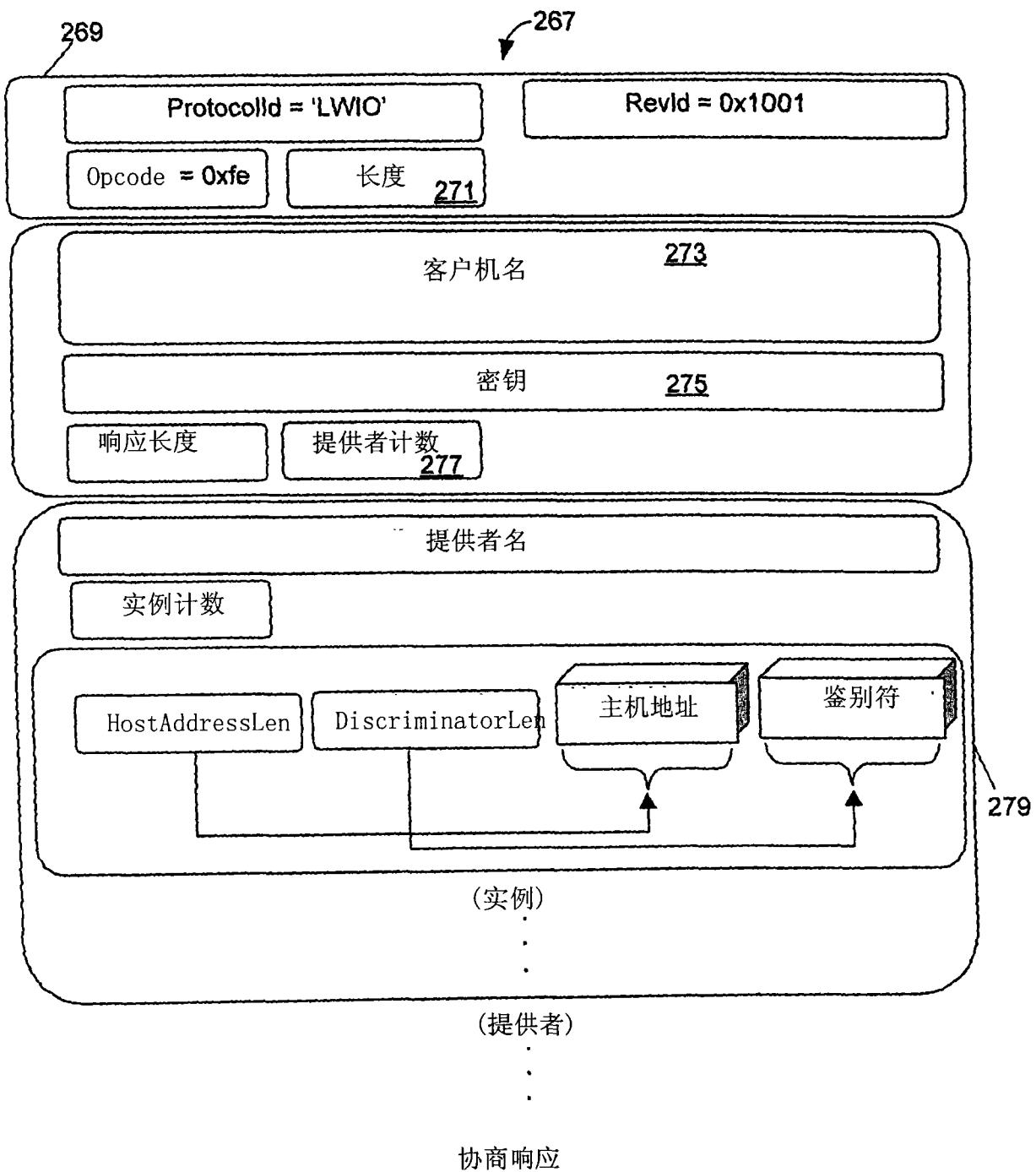


图 4B

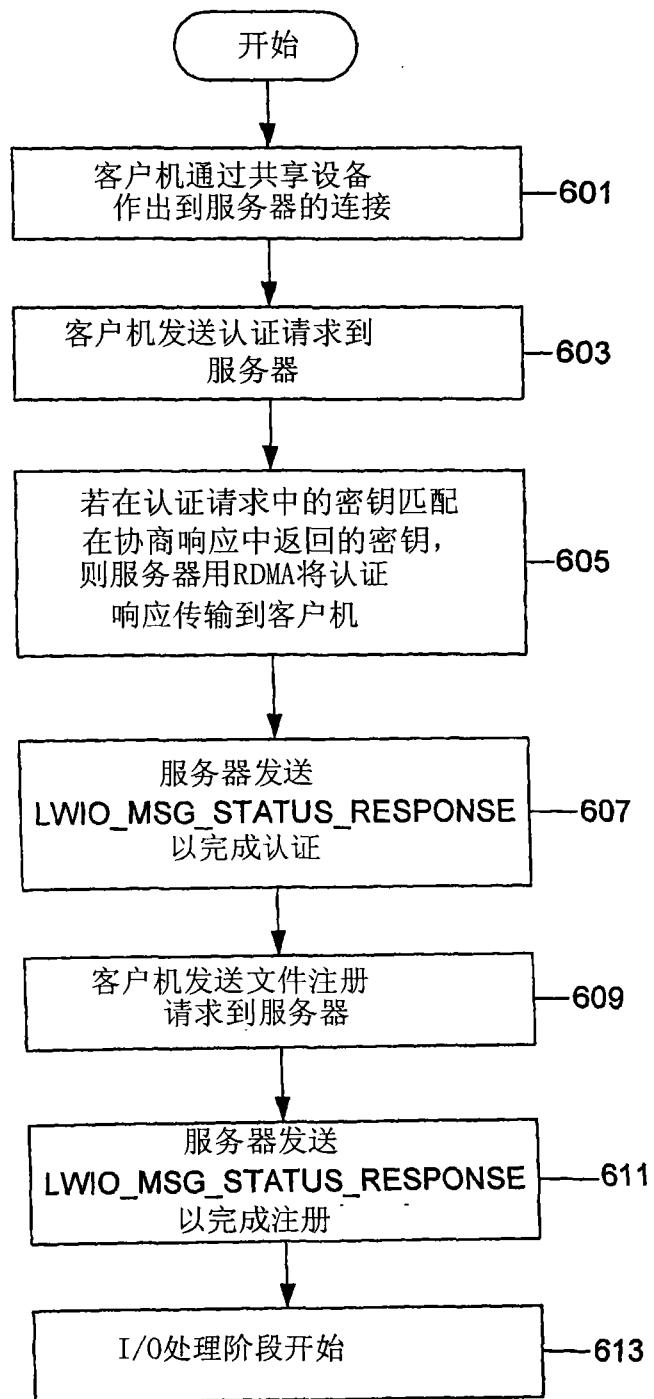
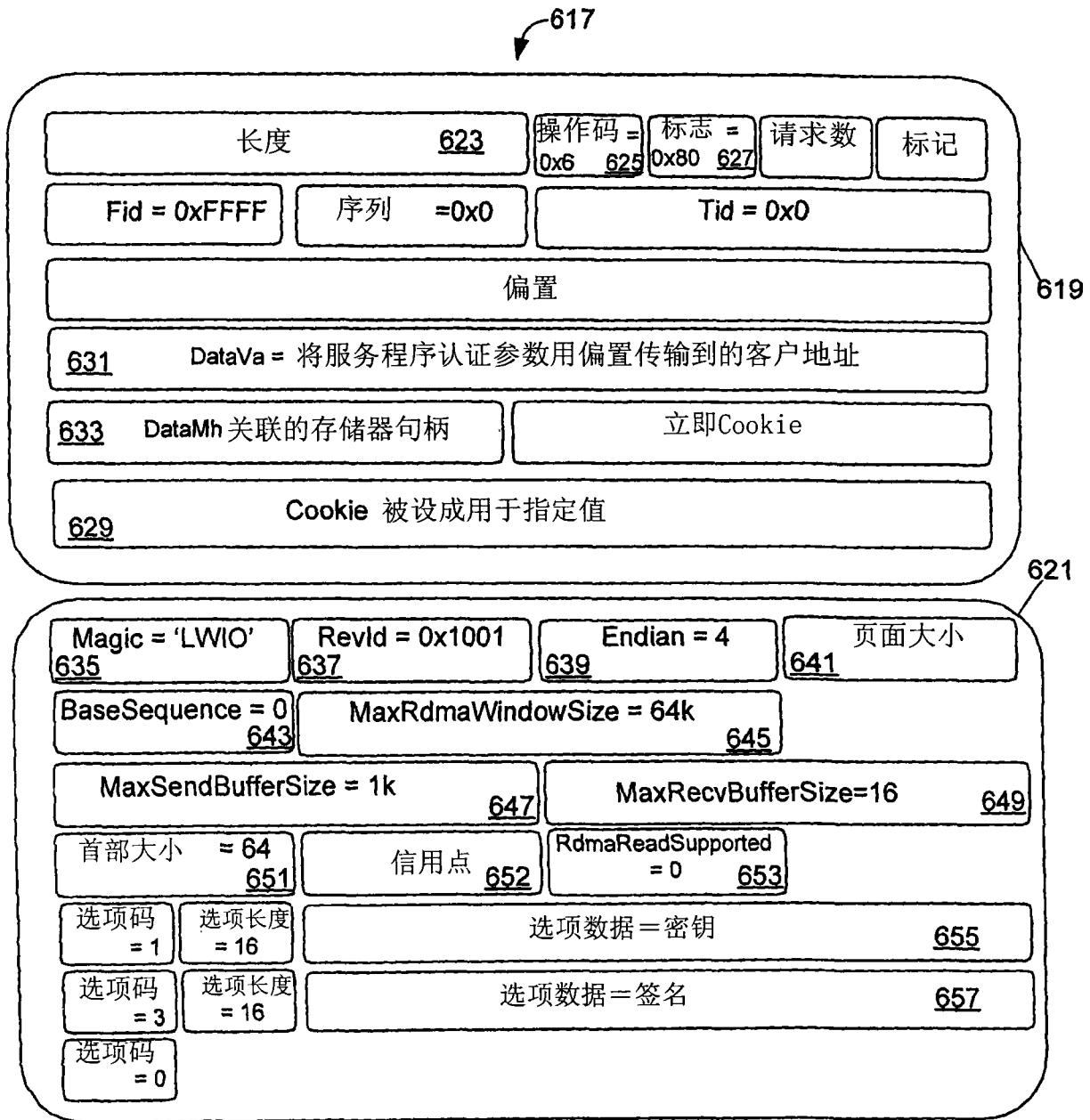
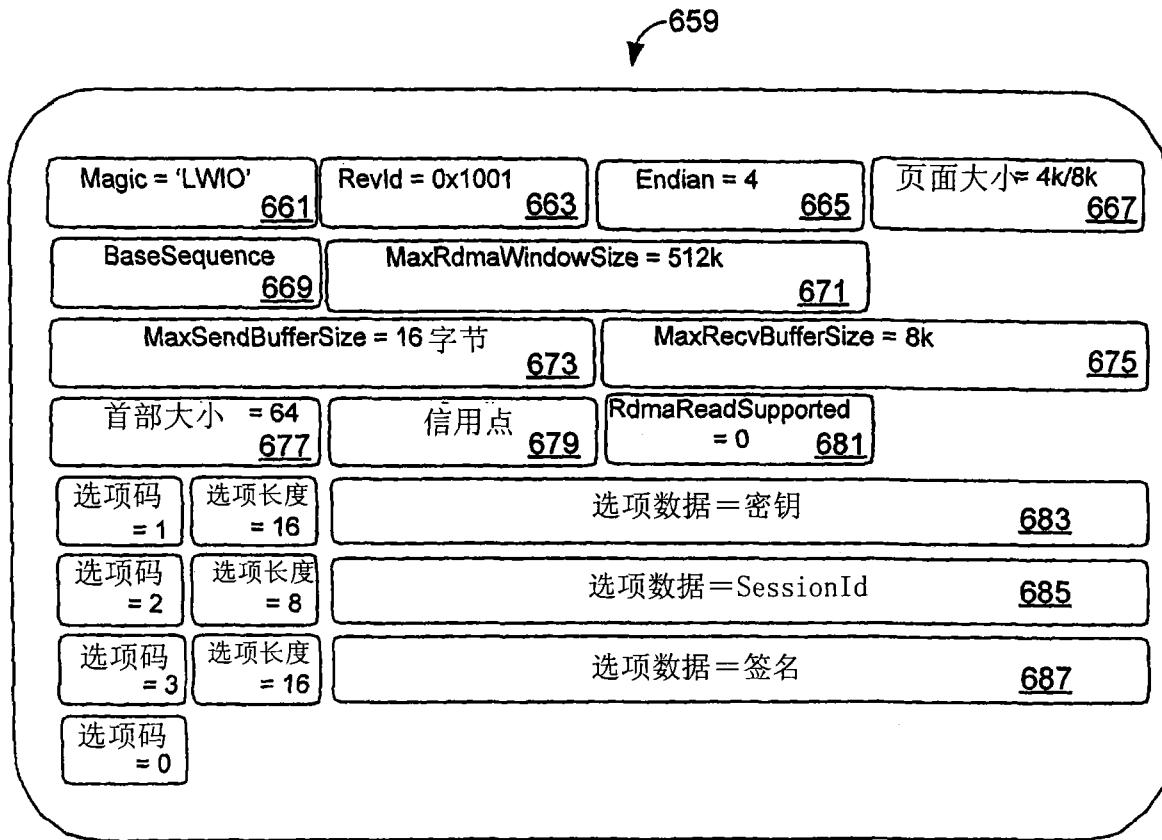


图 5



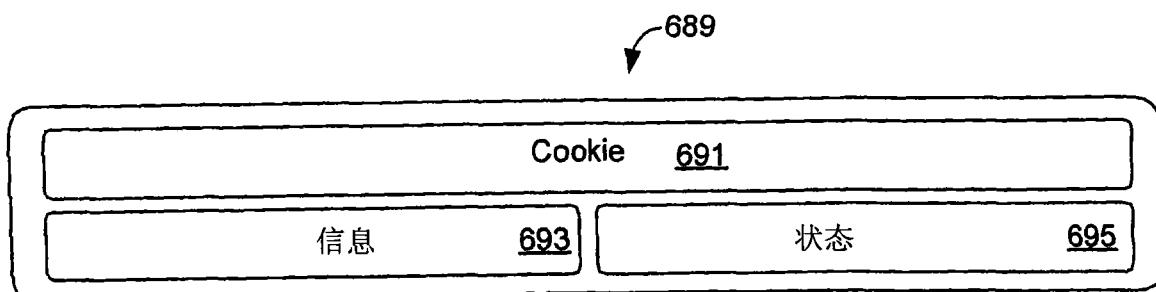
认证

图 6



认证响应

图 6B



服务器认证完成

图 6C

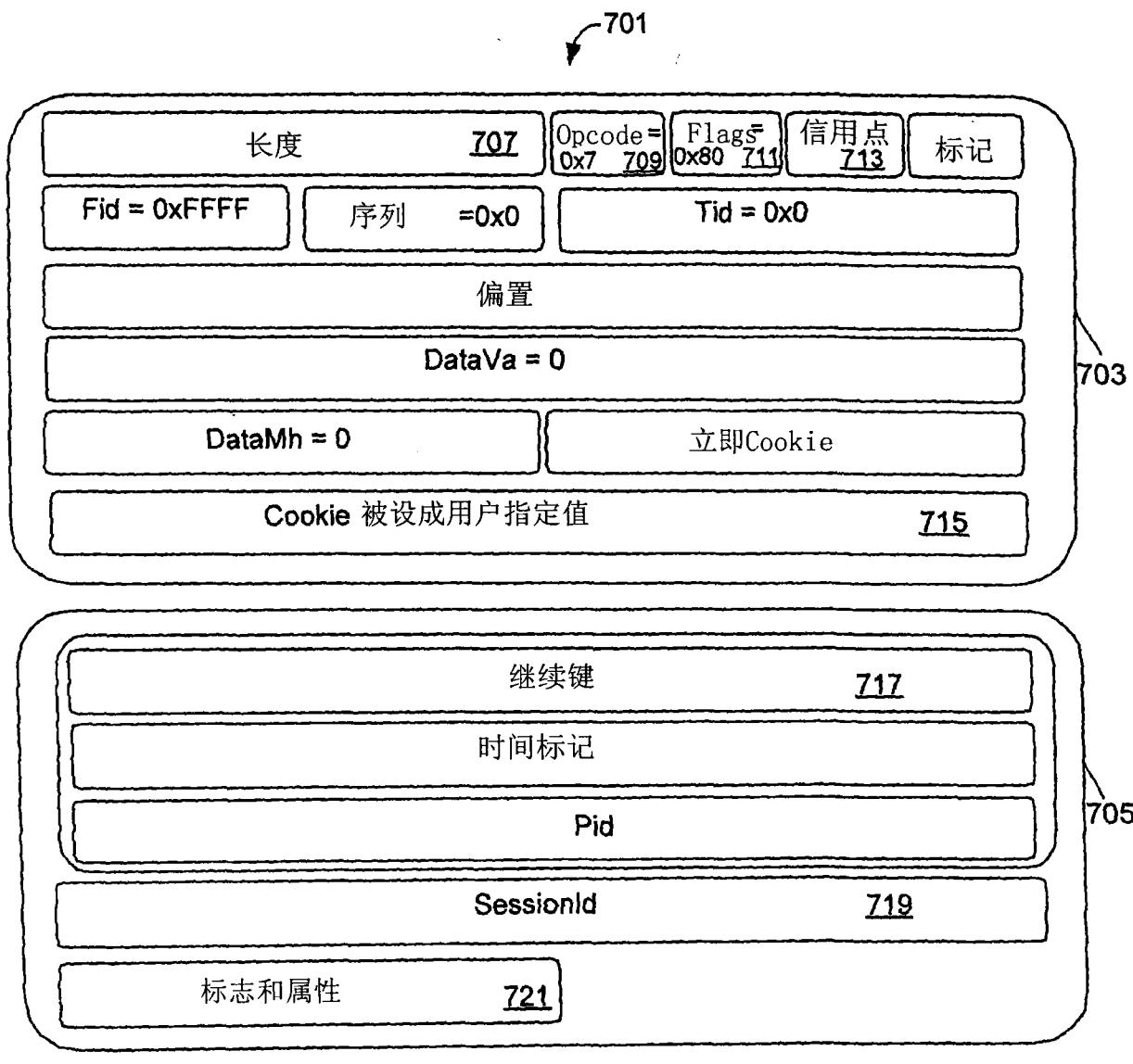
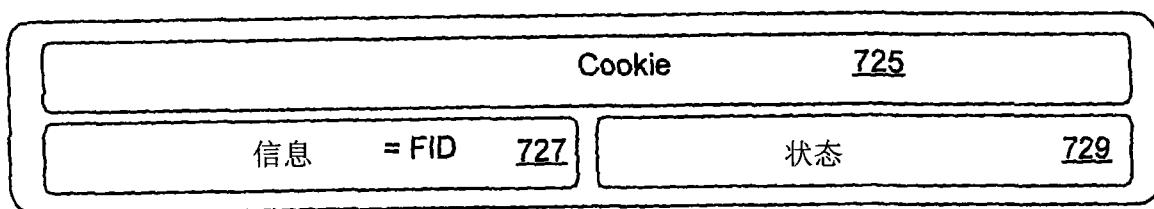


图 7A



服务器注册文件完成

图 7B

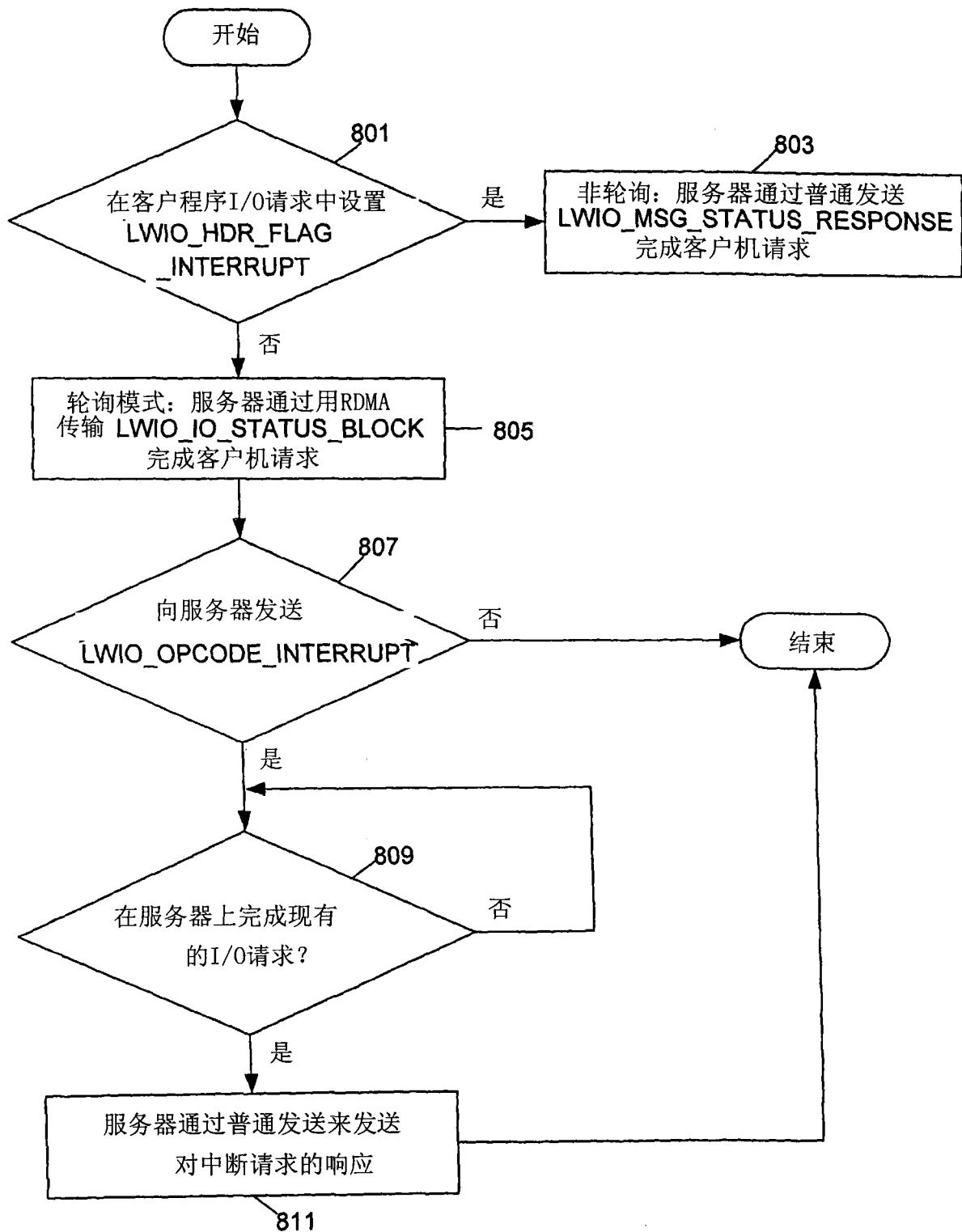
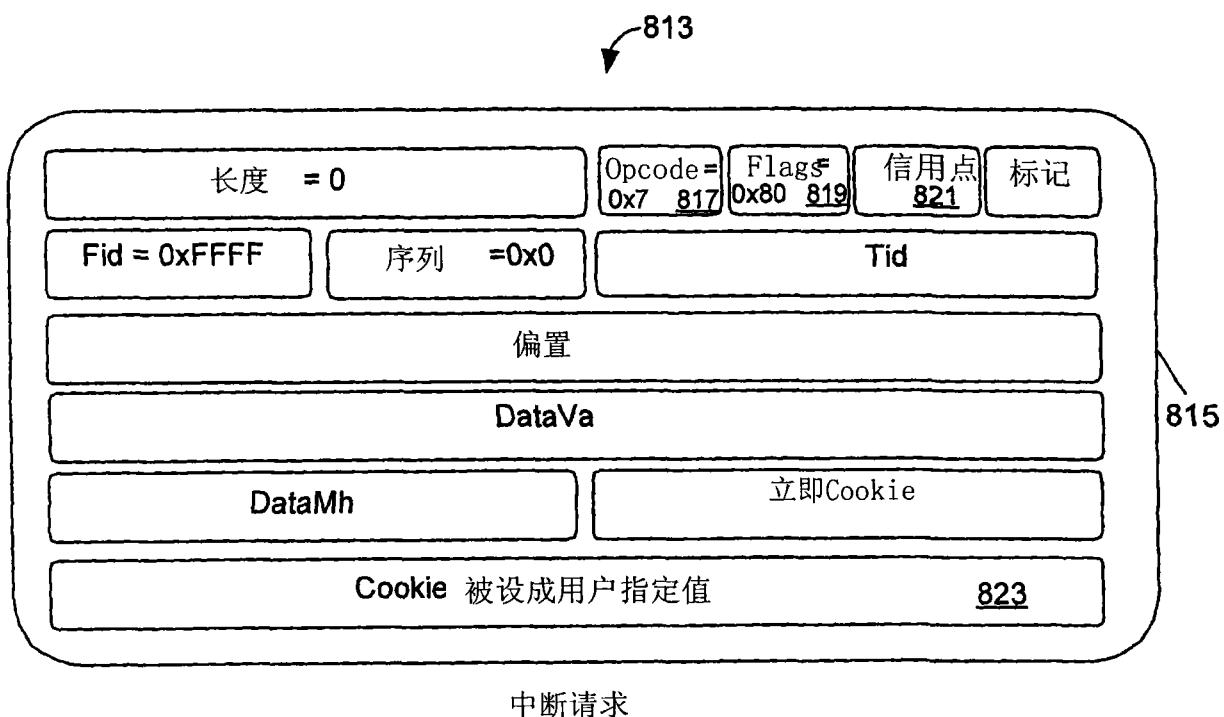
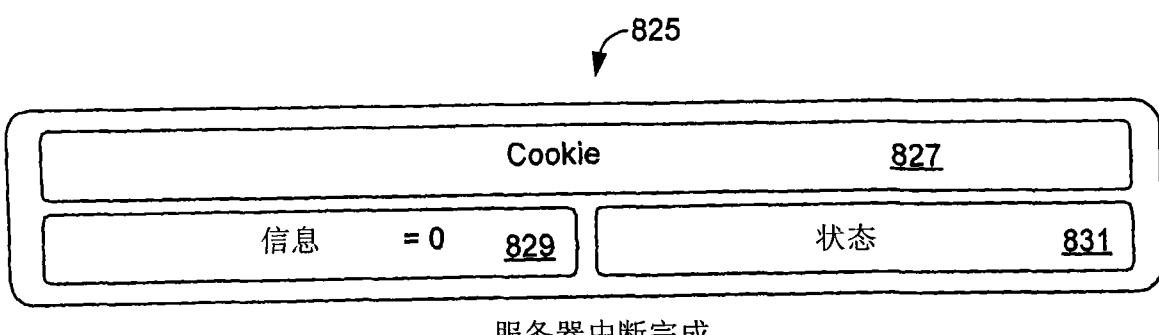


图 8



中断请求

图 9A



服务器中断完成

图 9B

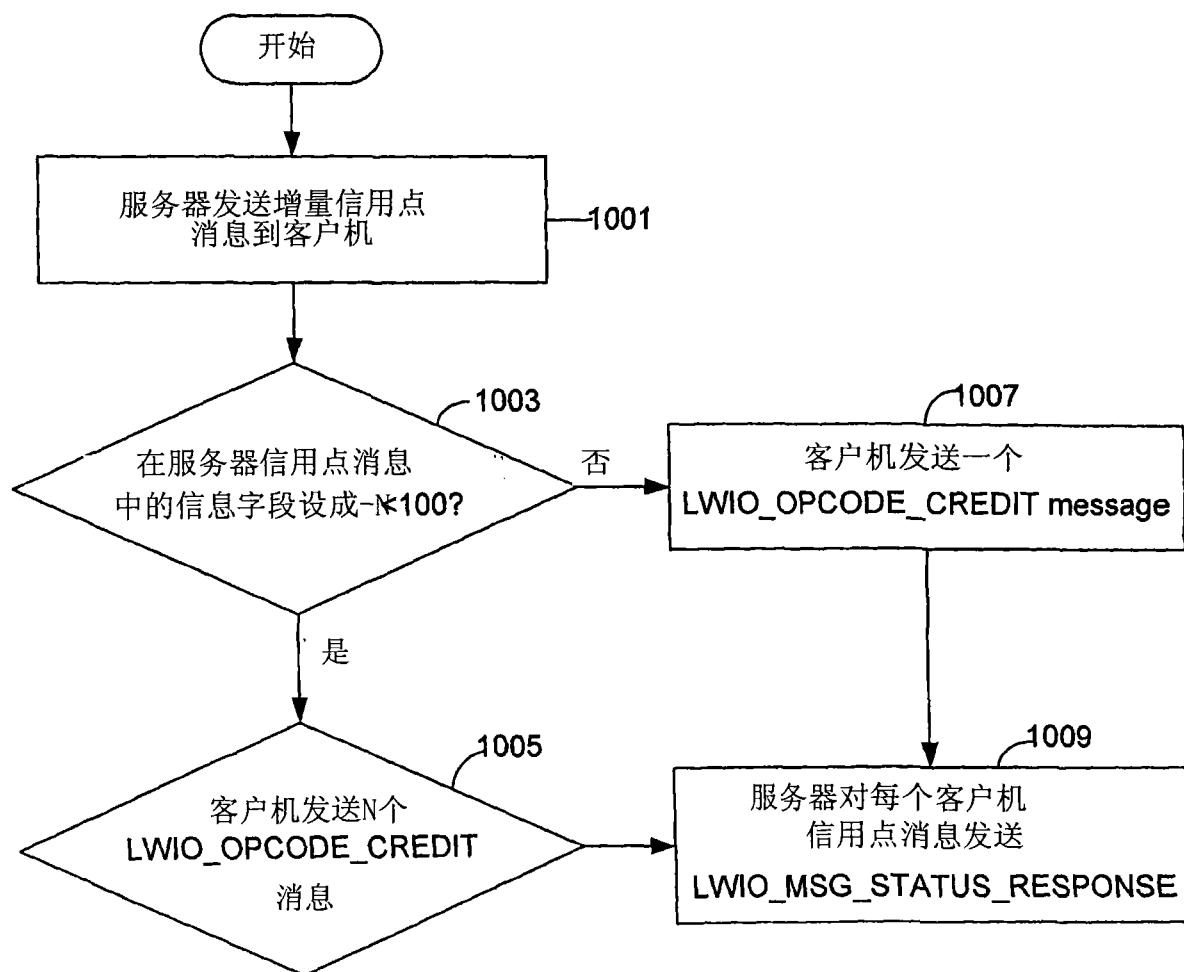
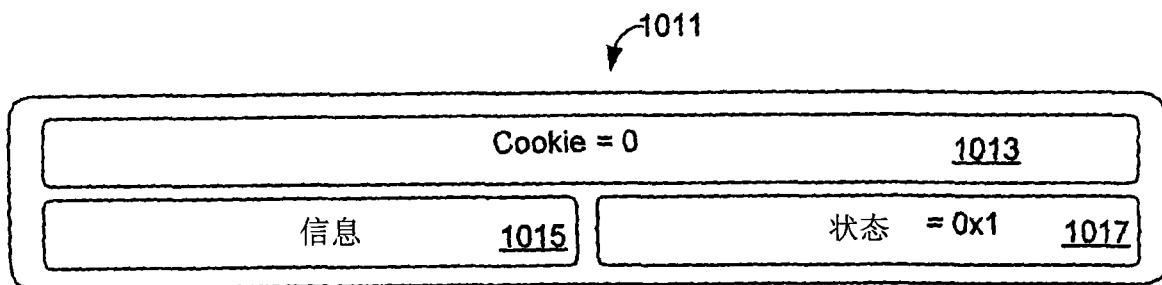
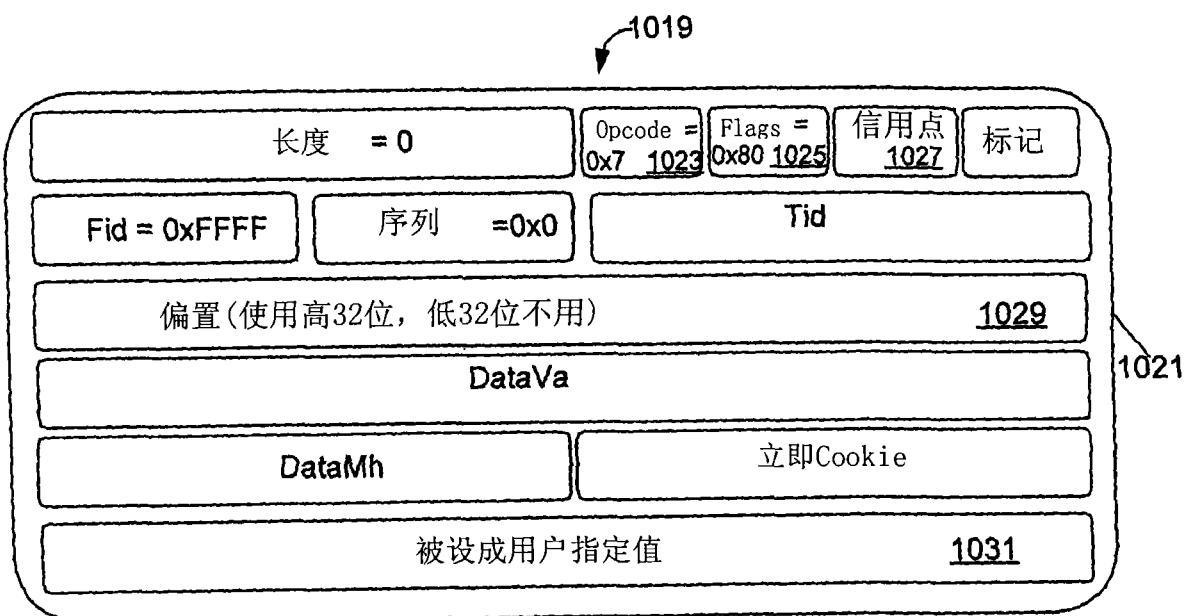


图 10



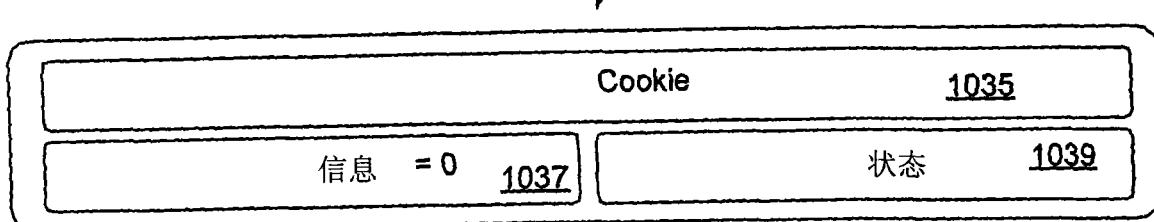
服务程序的客户程序请求数消息

图 11A



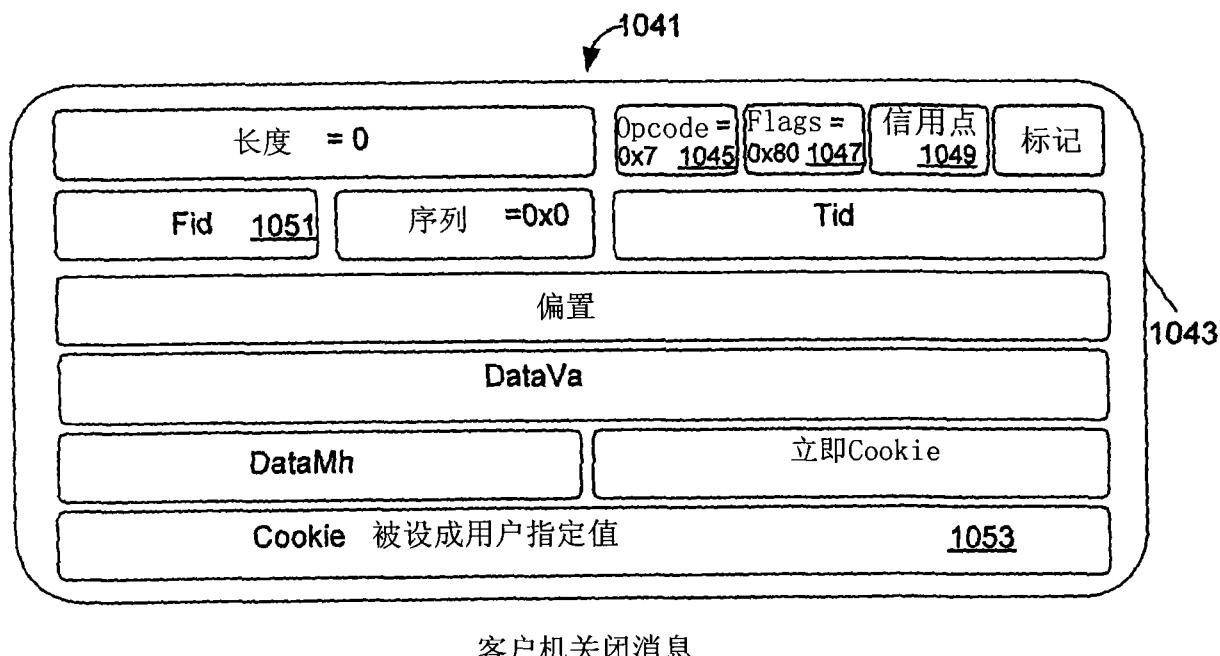
客户机到服务器的信用点消息

图 11B



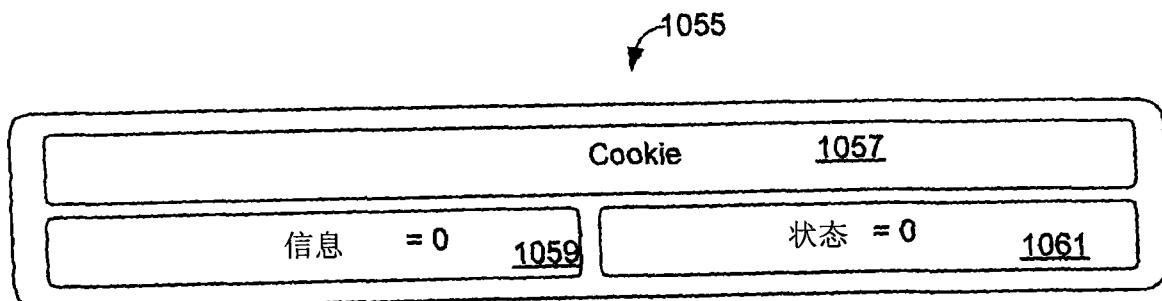
服务器信用点完成

图 11C



客户机关闭消息

图 12A



服务器关闭完成

图 12B

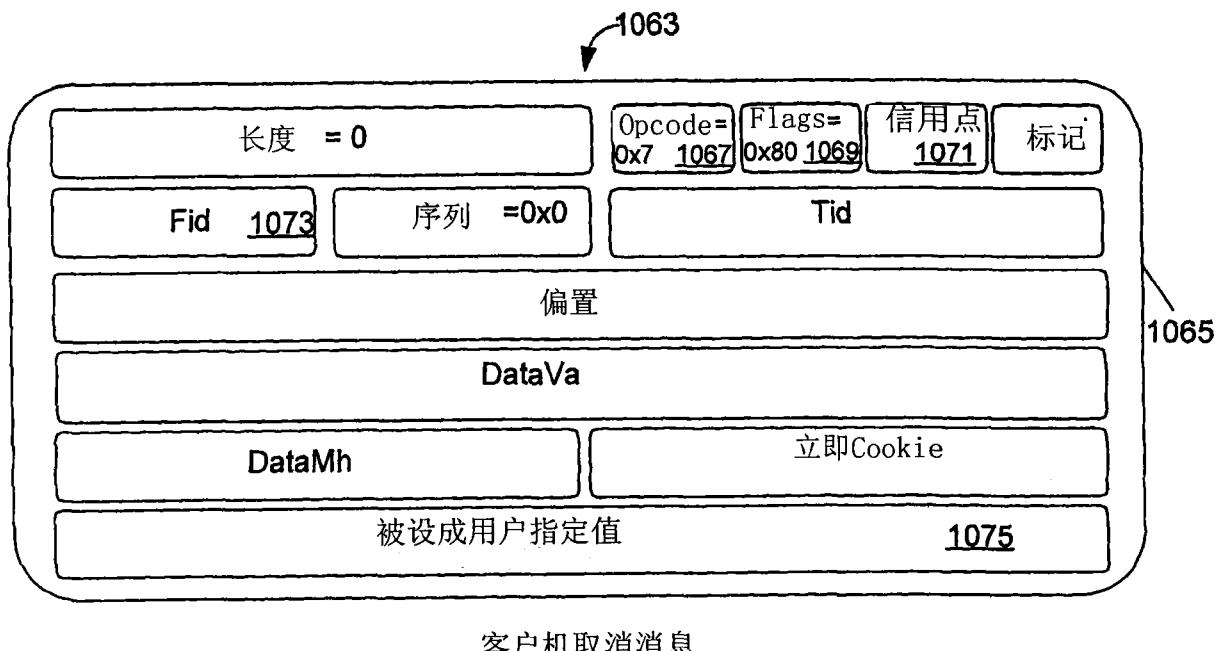


图 13A

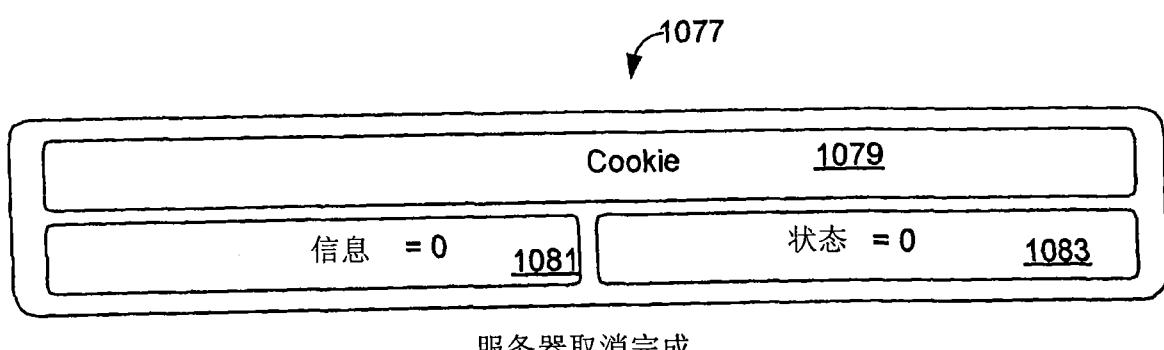
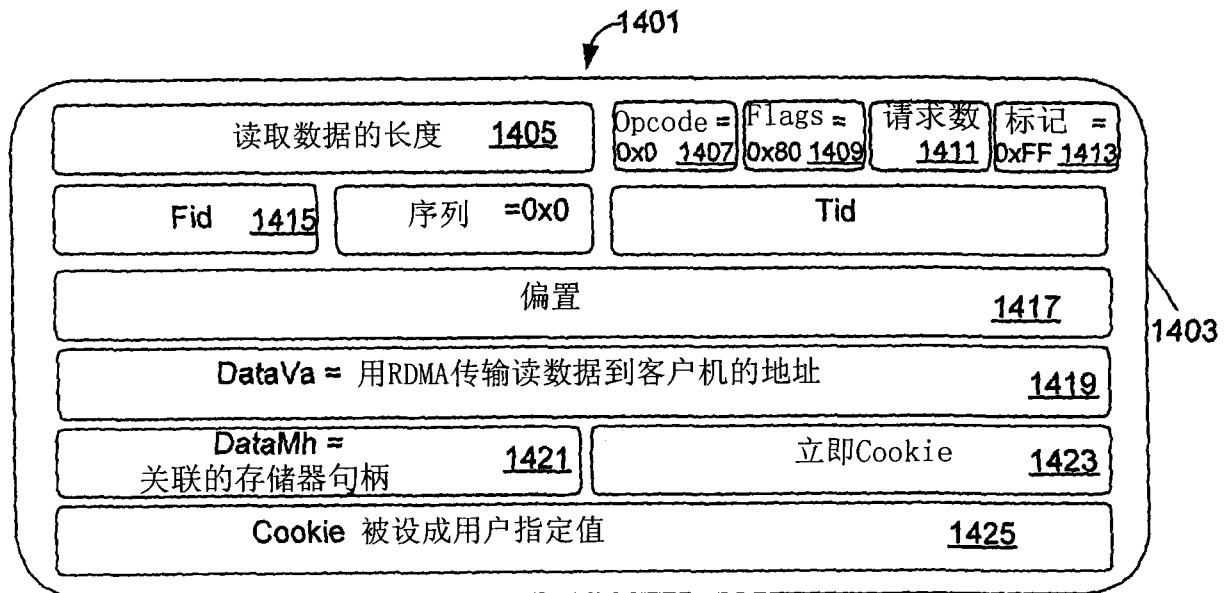
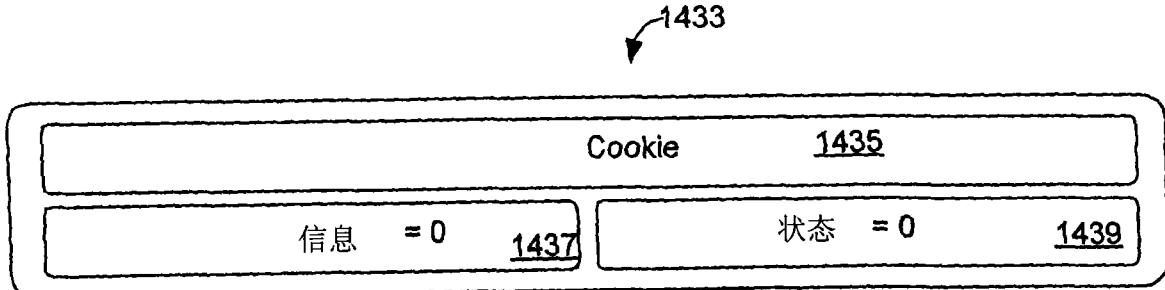


图 13B



客户机读消息(非轮询)

图 14A



读完成(非轮询)

图 14B

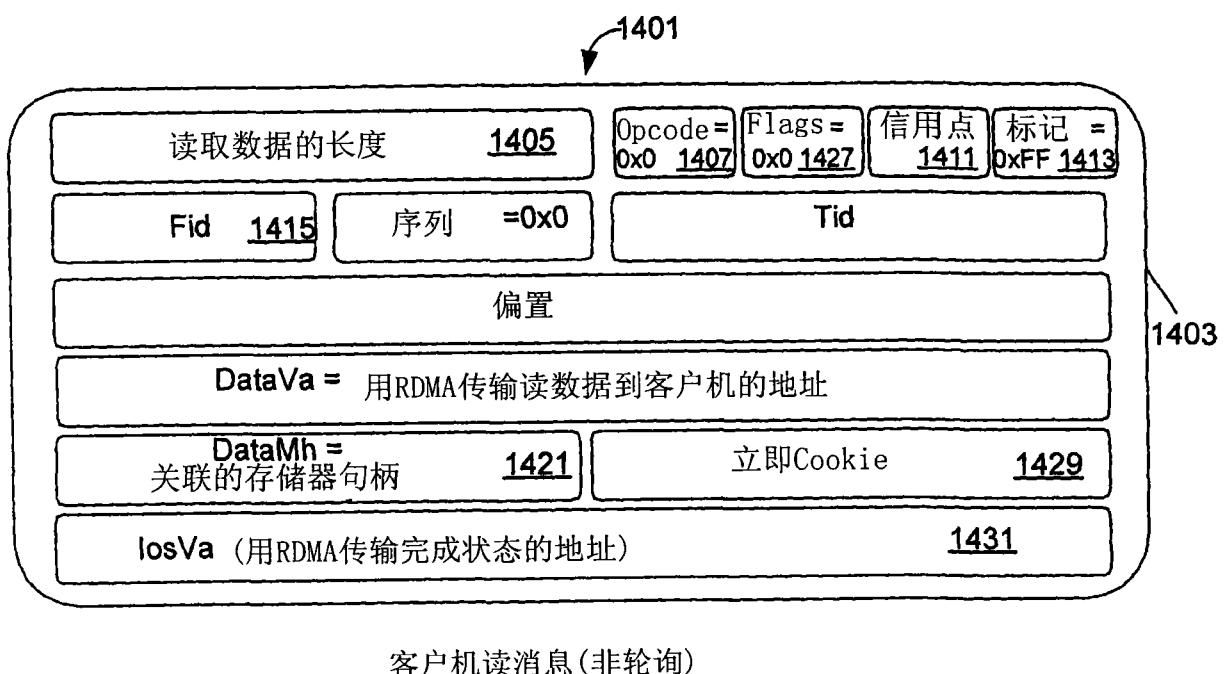


图 14C

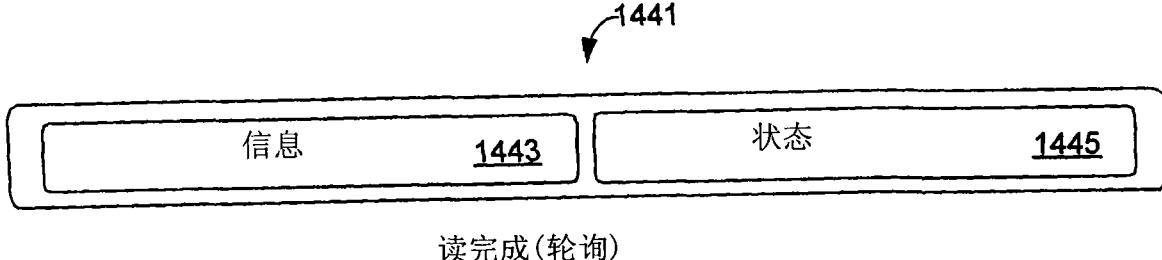


图 14D

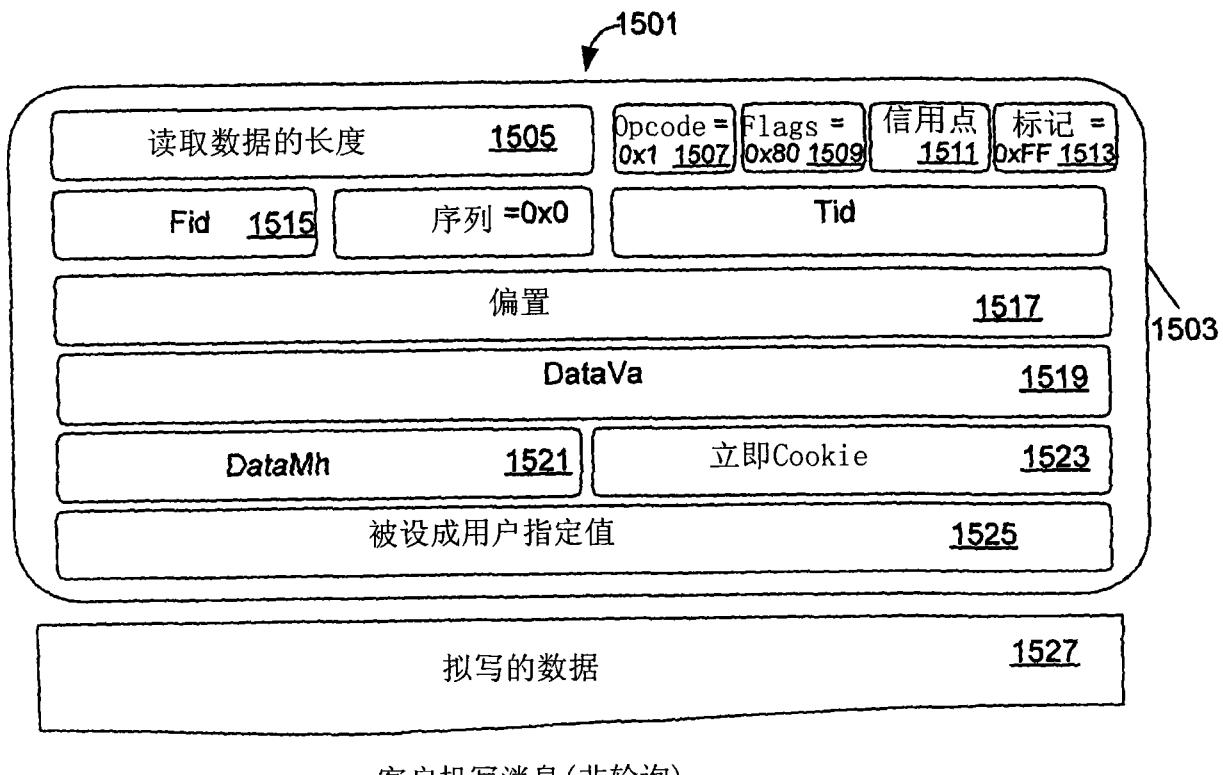
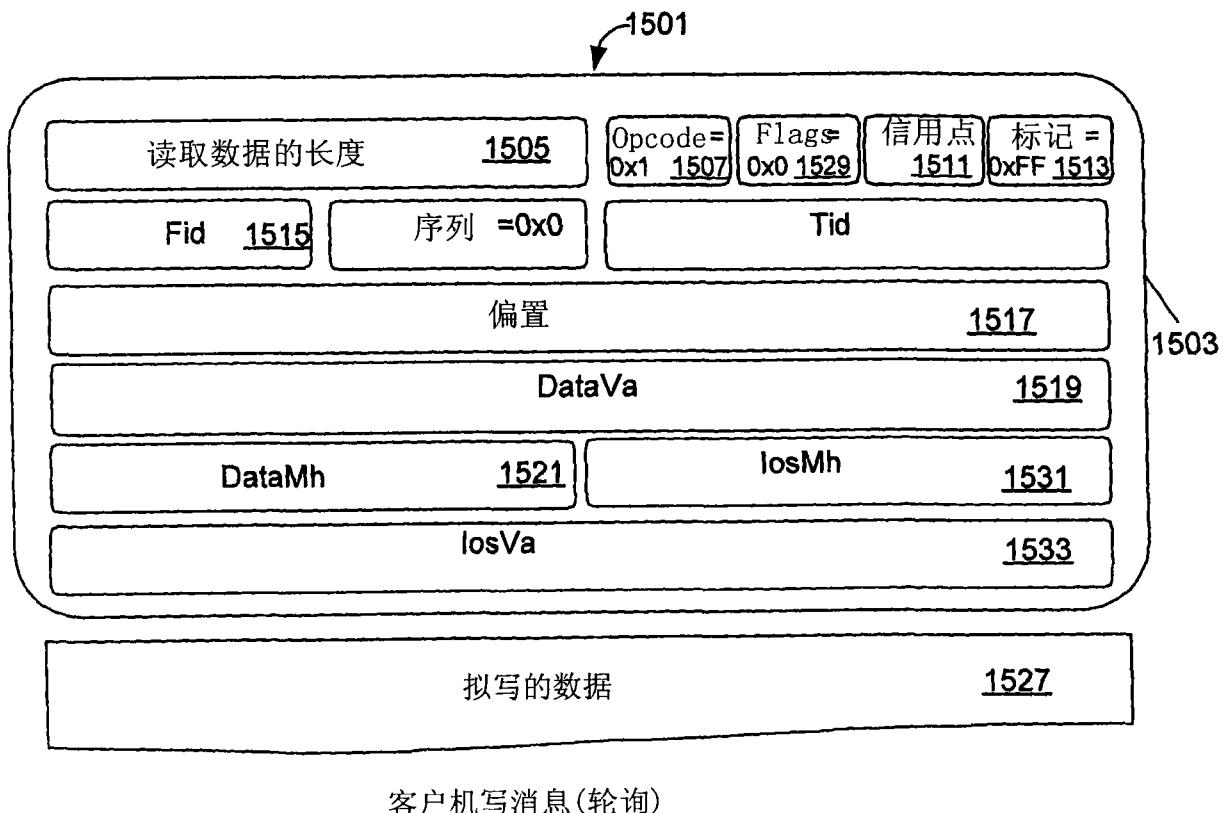


图 15A

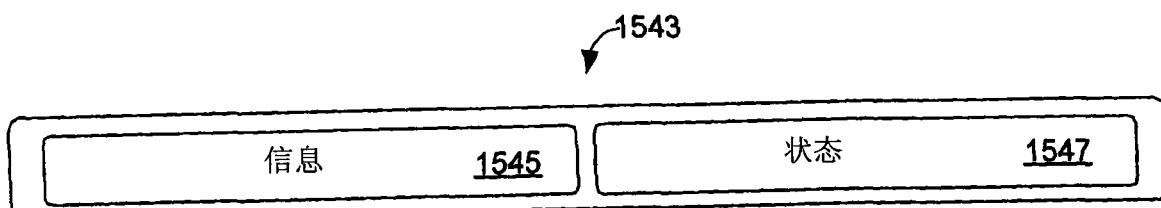


图 15B



客户机写消息(轮询)

图 15C



写完成(轮询)

图 15D

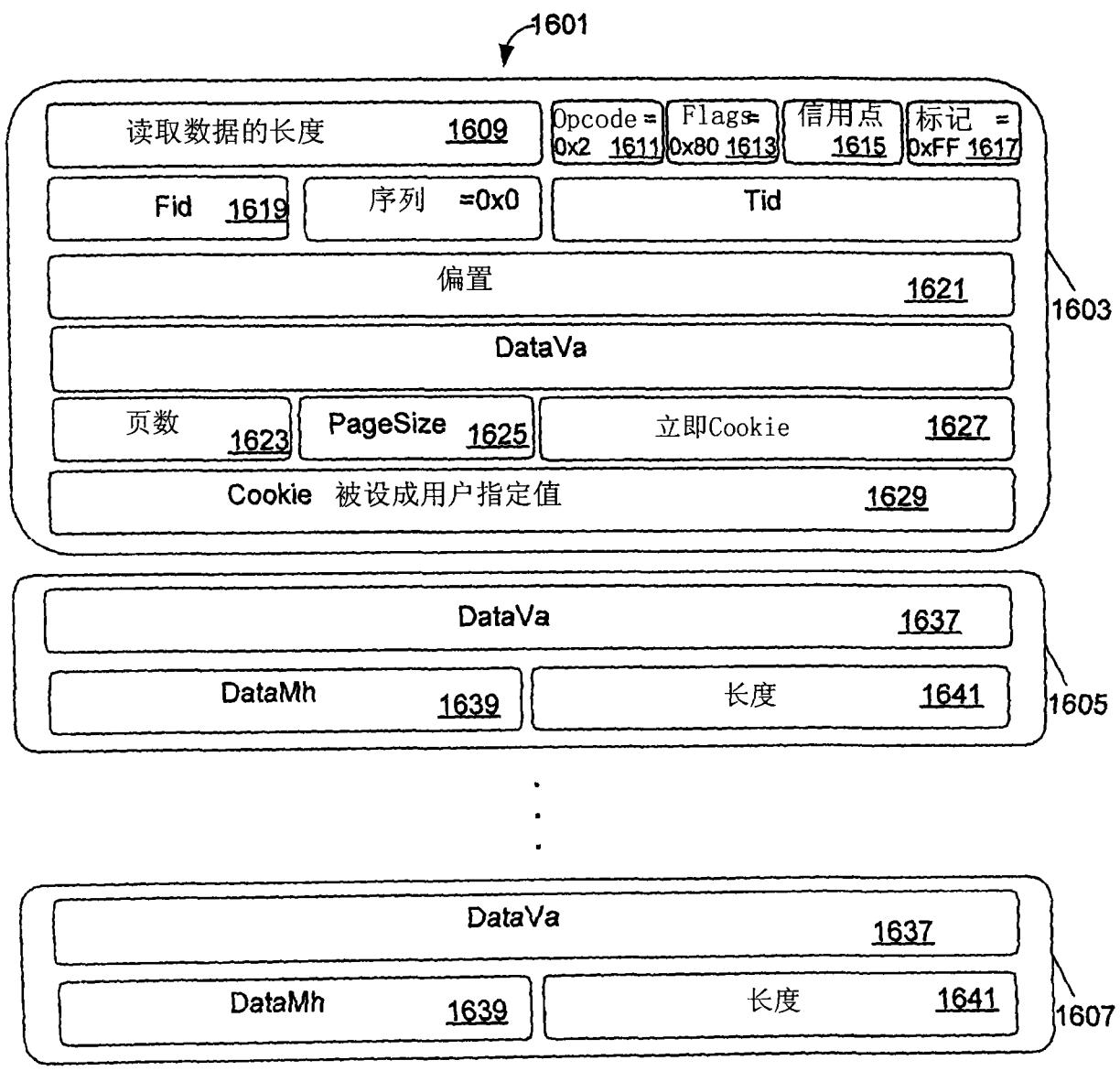
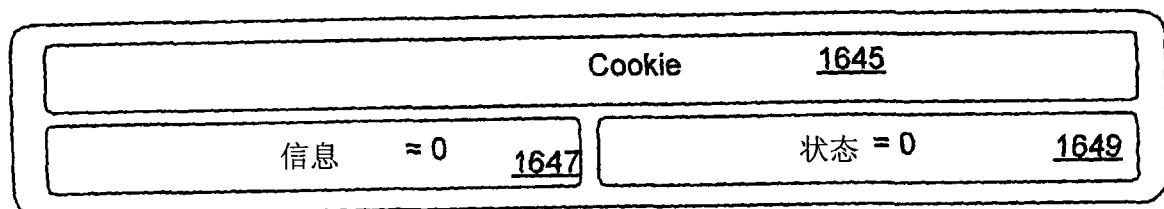
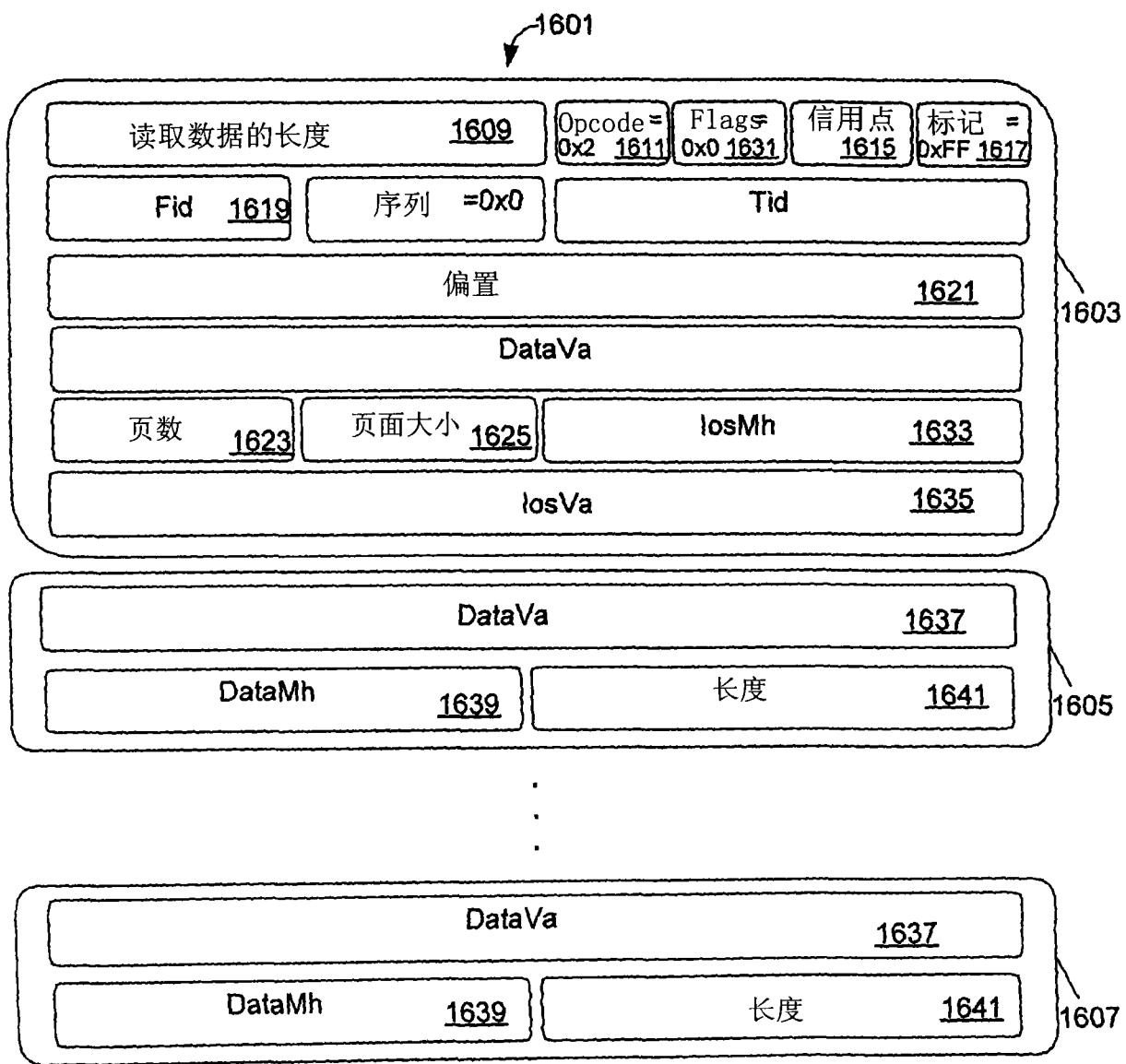


图 16A



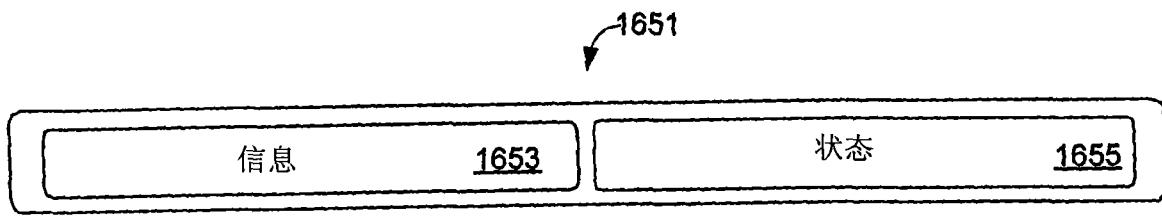
向量化读完成(非轮询)

图 16B



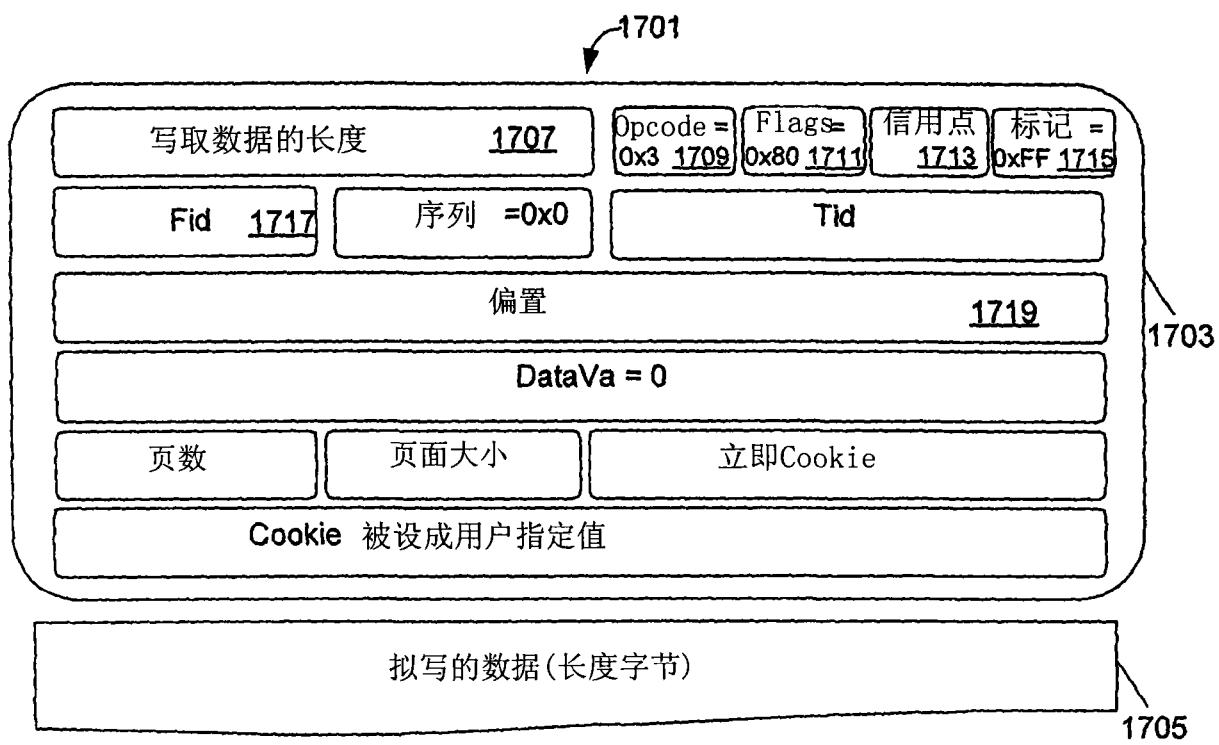
客户机向量化读消息(轮询)

图 16C



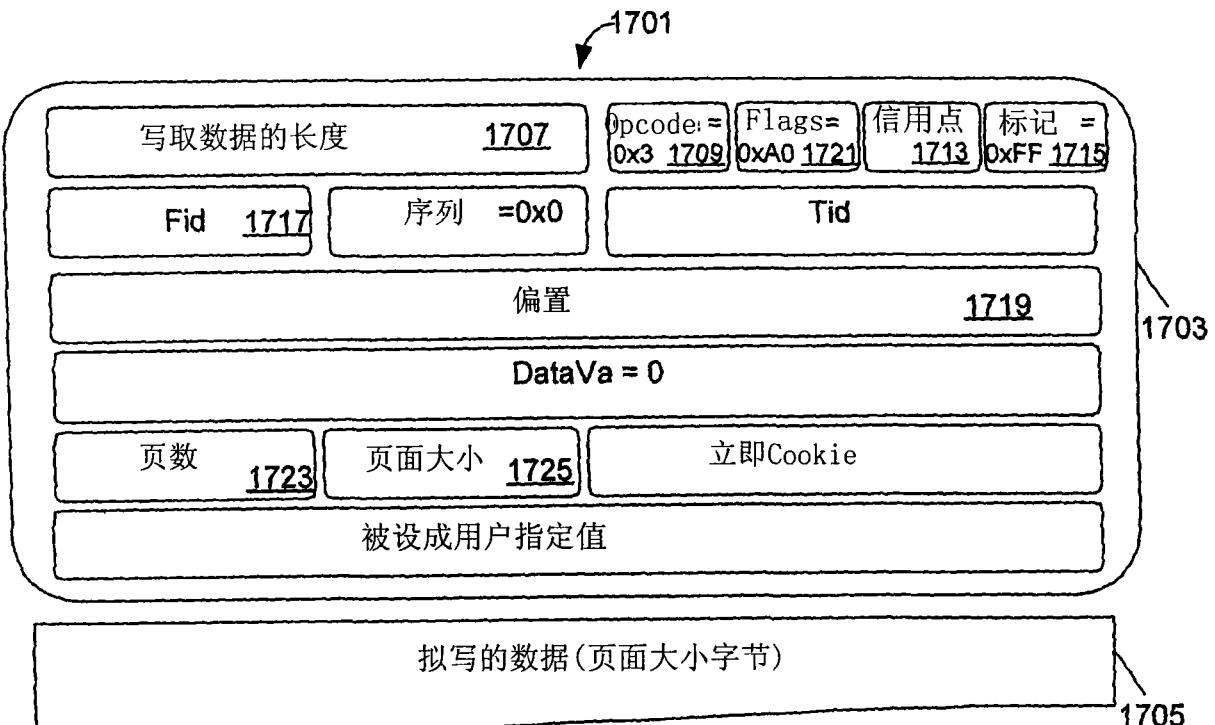
向量化读完成(轮询)

图 16D



客户机向量化写消息，非崩溃(非轮询)

图 17A



客户机向量化写消息，崩溃(非轮询)

图 17B

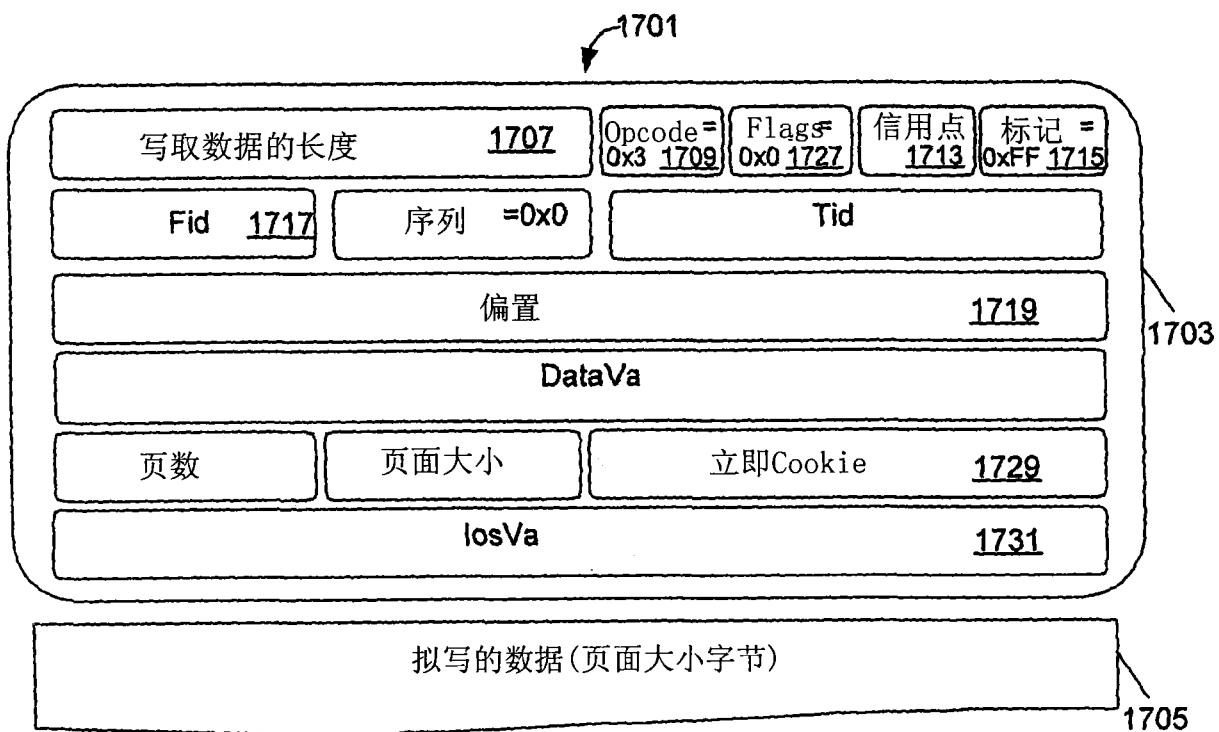


图 17C

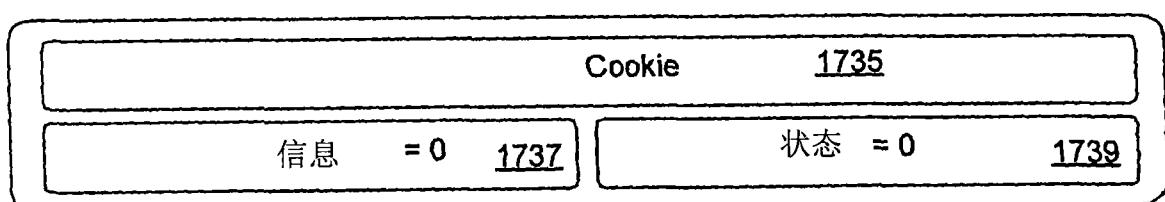


图 17D

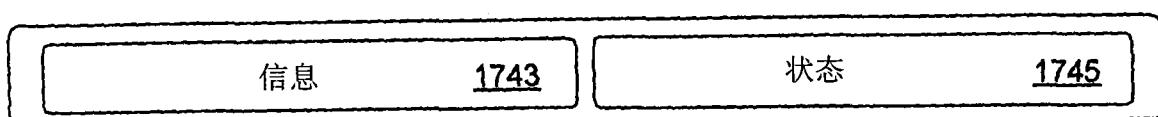


图 17E