



(12) 发明专利

(10) 授权公告号 CN 1798111 B

(45) 授权公告日 2012.01.11

(21) 申请号 200510128754.6

US 2003/0058878 A1, 2003.03.27, 全文.

(22) 申请日 2005.11.30

US 6058389 B, 2000.05.02, 说明书第5栏64

(30) 优先权数据

11/026,501 2004.12.30 US

行至第7栏第10行, 第9栏3行至第11栏第2
行, 第12栏30行至第13栏第22行, 第17栏
49-63行、摘要.

(73) 专利权人 微软公司

审查员 王伦杰

地址 美国华盛顿州

(72) 发明人 C·-J·黄 J·M·科斯坦恩

P·嘎纳帕斯拉杰 S·科雷普尔

(74) 专利代理机构 上海专利商标事务所有限公

司 31100

代理人 张政权

(51) Int. Cl.

H04L 12/58 (2006.01)

H04L 29/06 (2006.01)

(56) 对比文件

CN 1536845 A, 2004.10.13, 全文.

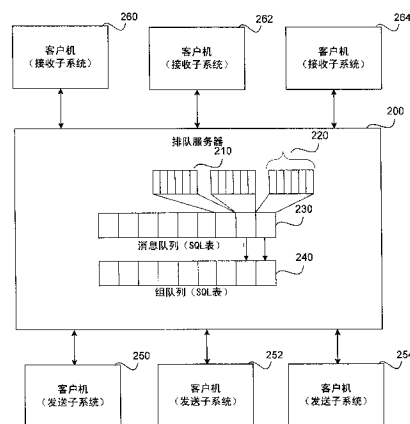
权利要求书 2 页 说明书 9 页 附图 5 页

(54) 发明名称

服务器排队系统和方法

(57) 摘要

一种用于可靠消息传送的排队服务器, 其中一个子系统想要异步地执行一个或多个已排序操作。消息被成组地发送给队列, 一组中可以含有一条或多条消息。在特定组中的消息以预定的顺序处理。可任选地, 消息组可以被标记为相关的, 使得特定相关中的所有组可以按预定的顺序处理。消息可以被存储在 SQL 数据库表中, 直到对该消息的处理完成。消息系统的接收方可以跨多个机器和 / 或跨任何给定机器的可用资源而缩放。系统可以处理发送方的“灾难”情形 (即, 发送机器在发送组过程中崩溃) 和接收方的“灾难”情形 (即, 电源故障引起至少一个接收机器上的重新引导)。



1. 一种用于排队消息的计算机实现的方法,包括:

在排队系统中接收从发送者子系统发送的消息组,其中,每一消息组包含至少一条消息,其中消息队列含有对发送的每一个别消息的记录,并且其中当发送者子系统发送消息组时,新的记录被添加到组队列,使得所述组队列对发送的每一消息组含有一记录;

使用所述排队系统以预定的顺序在所述组队列中存储所接收到的消息组;

在所述排队系统中接收每一消息组的相关标识符,其中,所述相关标识符用于关联消息组;

使用所述排队系统存储所接收的相关标识符;以及

根据所述预定的顺序和所述相关标识符来发送所存储的消息组。

2. 如权利要求 1 所述的计算机实现的方法,其特征在于,所述接收到的消息组被存储在 SQL 表中。

3. 如权利要求 1 所述的计算机实现的方法,其特征在于,所述存储的消息组被发送给多台机器。

4. 如权利要求 1 所述的计算机实现的方法,其特征在于,一优先级值与每一接收到的消息组相关联。

5. 如权利要求 4 所述的计算机实现的方法,其特征在于,所述消息组是根据所述相关联的优先级值来发送的。

6. 如权利要求 1 所述的计算机实现的方法,其特征在于,一 GUID 被用作所属相关标识符。

7. 如权利要求 6 所述的计算机实现的方法,其特征在于,所述 GUID 是客户机提供的。

8. 如权利要求 1 所述的计算机实现的方法,其特征在于,所述预定的顺序是接收所述消息组的顺序。

9. 如权利要求 1 所述的计算机实现的方法,其特征在于,相关联组的最后一个消息组是在发送所述相关联组的任何存储的消息组之前接收的。

10. 如权利要求 1 所述的计算机实现的方法,其特征在于,还包括对由所述相关标识符关联的接收到的消息组锁定发送,以及一旦接收到由所述相关标识符关联的最后一个消息组,就对所接收的消息组解除锁定发送。

11. 一种用于排队消息的系统,包括:

用于在排队系统中发送从发送者子系统发送的消息组的装置,其中,每一消息组包含至少一条消息,其中消息队列含有对发送的每一个别消息的记录,并且其中当发送者子系统发送消息组时,新的记录被添加到组队列,使得所述组队列对发送的每一消息组含有一记录;

用于使用所述排队系统以预定的顺序在所述组队列中存储所接收到的消息组的装置;

用于在所述排队系统中接收每一消息组的相关标识符的装置,其中,所述相关标识符用于关联消息组;

用于使用所述排队系统存储所接收的相关标识符的装置;以及

用于根据所述预定的顺序和所述相关标识符来发送所存储的消息组的装置。

12. 如权利要求 11 所述的系统,其特征在于,所述接收到的消息组被存储在 SQL 表中。

13. 如权利要求 11 所述的系统,其特征在于,所述存储的消息组被发送给多台机器。

14. 如权利要求 11 所述的系统,其特征在于,一优先级值与每一接收到的消息组相关联。

15. 如权利要求 14 所述的系统,其特征在于,所述消息组是根据所述相关联的优先级值来发送的。

16. 一种用于排队消息的计算机实现的系统,包括:

用于在排队系统中接收从发送者子系统发送的消息组的装置,其中,每一消息组包含至少一条消息,其中消息队列含有对发送的每一个别消息的记录,并且其中当发送者子系统发送消息组时,新的记录被添加到组队列,使得所述组队列对发送的每一消息组含有一记录;

用于使用所述排队系统以预定的顺序在所述组队列中存储所接收到的消息组的装置;

用于在所述排队系统中接收每一消息组的相关标识符的装置,其中,所述相关标识符用于关联消息组;

用于使用所述排队系统存储所接收的相关标识符的装置;以及

用于根据所述预定的顺序和所述相关标识符来发送所存储的消息组的装置。

17. 如权利要求 16 所述的系统,其特征在于,一 GUID 被用作所述相关标识符。

18. 如权利要求 16 所述的系统,其特征在于,所述预定的顺序是接收所述消息组的顺序。

19. 如权利要求 16 所述的系统,其特征在于,相关联组的最后一个消息组是在发送所述相关联组的任何存储的消息组之前接收的。

20. 如权利要求 16 所述的系统,其特征在于,还包括用于对由所述相关标识符关联的接收到的消息组锁定发送的装置,以及用于一旦接收到由所述相关标识符关联的最后一个消息组,就对所接收的消息组解除锁定发送的装置。

服务器排队系统和方法

背景技术

[0001] 排队系统允许若干不同的用户（和 / 或进程）发送数据和 / 或处理请求，这些数据和请求将被存储在队列中以供接收子系统稍后使用。例如，项目服务器子系统可以向排队系统发送一条或多条消息。排队系统存储该信息，直到接收子系统能够检索并处理所存储的信息。然而，机器故障和子系统与排队系统之间和之中的通信中断可能会引起处理中的差错和 / 或使得不能够处理所存储的信息。

发明内容

[0002] 本发明针对用于可靠消息传送的服务器排队方法和系统。排队系统可以在例如其中一个子系统想要异步地执行一个或多个已排序操作的情况中使用。消息被成组地发送给队列，一组中可以含有一条或多条消息。在特定组中的消息以预定的顺序处理。消息组可以被标记为相关的，使得特定相关中的所有组可以按预定的顺序处理。消息可以被存储在 SQL 数据库表中，直到对该消息的处理完成。消息系统的接收方可以跨多个机器和 / 或跨任何给定机器的可用资源而缩放。系统可以处理发送方的“灾难”情形（即，发送机器在发送组过程中崩溃）和接收方的“灾难”情形（即，电源故障引起至少一个接收机器上的重新引导）。

[0003] 在本发明的一方面中，一种排队系统接收从发送者子系统发送的消息组，其中，每一消息组包含至少一条消息。所接收的消息组使用排队系统以预定的顺序存储。排队系统接收每一消息组的相关标识符；所接收的相关标识符使用排队系统存储。所存储的消息组和相关标识符根据预定的顺序发送。

附图说明

[0004] 图 1 示出了根据本发明的示例性实施例可以使用的计算设备。

[0005] 图 2 示出了根据本发明的各方面用于可靠消息传送的排队系统的框图。

[0006] 图 3 示出了根据本发明的各方面，示出用于获取下一可用组的进程的操作流程图。

[0007] 图 4 示出了根据本发明的各方面，示出用于使用接收者和工作者线程的进程的操作流程图。

[0008] 图 5 示出了根据本发明的各方面，示出用于使用消息处理工作者线程的进程的操作流程图。

具体实施方式

[0009] 本发明针对一种用于可靠消息传送的服务器排队方法和系统。排队服务器可以例如在其中一个子系统想要异步地执行一个或多个已排序操作的情况下使用。消息被成组地发送给队列，一组中可以含有一条或多条消息。在特定组中的消息以预定的顺序处理。消息组可以被标记为相关的，使得特定相关中的所有组可以按预定的顺序处理。消息可以被

存储在 SQL 数据库表中,直到对该消息的处理完成。消息系统的接收方可以跨多个机器和 / 或跨任何给定机器的可用资源而缩放。系统可以处理发送方的“灾难”情形(即,发送机器在发送组过程中崩溃)和接收方的“灾难”情形(即,电源故障引起至少一个接收机器上的重新引导)。

[0010] 说明性操作环境

[0011] 参考图 1,用于实现本发明的一个示例性系统包括诸如计算设备 100 的计算设备。计算设备 100 可以被配置为客户机、服务器、移动设备或与基于网络的协作系统中的数据交互的任何其它计算设备。在最基本配置中,计算设备 100 一般包括至少一个处理单元 102 和系统存储器 104。取决于计算设备的确切配置和类型,系统存储器 104 可以是易失性的(诸如 RAM),非易失性的(诸如 ROM、闪存等)或者两者的某种组合。系统存储器 104 一般包括操作系统 105、一个或多个应用程序 106,且可以包括程序数据 107。将在下文详细描述的服务排队系统 108 一般在应用程序 106 内实现。

[0012] 计算设备 100 可以具有额外的特征或功能。例如,计算设备 100 也可以包括诸如磁盘、光盘或磁带等附加数据存储设备(可移动和 / 或不可移动)。这样的附加存储在图 1 中由可移动存储 109 和不可移动存储 110 示出。计算机存储介质可以包括以用于诸如计算机可读指令、数据结构、程序模块或其它数据等信息的存储的任何方法或技术实现的易失性和非易失性、可移动和不可移动介质。系统存储器 104、可移动存储 109 和不可移动存储 110 都是计算机存储介质的示例。计算机存储介质包括,但不限于,RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、数字多功能盘(DVD)或其它光盘存储、磁带盒、磁带、磁盘存储或其它磁性存储设备、或能用于存储所需信息且可以由计算设备 100 访问的任何其它介质。任何这样的计算机存储介质可以是设备 100 的一部分。计算设备 100 也可以含有诸如键盘、鼠标、笔、语音输入设备、触摸输入设备等输入设备。也可以包括诸如显示器、扬声器、打印机等输出设备。

[0013] 计算设备 100 也含有允许该设备诸如通过网络与其它计算机设备 118 通信的通信连接 116。网络包括局域网和广域网,以及其它大规模网络,包括但不限于内联网和外联网。通信连接 116 是通信介质的一个示例。通信介质通常具体化为诸如载波或其它传输机制等已调制数据信号中的计算机可读指令、数据结构、程序模块或其它数据,且包含任何信息传递介质。术语“已调制数据信号”指的是这样一种信号,其一个或多个特征以在信号中编码信息的方式被设定或更改。作为示例,而非限制,通信介质包括有线介质,诸如有线网络或直接线连接,以及无线介质,诸如声学、RF、红外线和其它无线介质。在此处使用的术语计算机可读介质包括存储介质和通信介质两者。

[0014] 排队系统服务器

[0015] 图 2 示出了根据本发明的各方面用于可靠消息传送的排队系统的框图。示例性系统包括排队服务器 200、发送子系统客户机 250、252、254 以及接收子系统客户机 260、262、264。(根据本发明的其它系统可以包括一个或多个发送子系统客户机和 / 或一个或多个接收子系统客户机)。服务器 210 包含一条或多条存储的消息,这些消息被组织为消息组 210。消息组 210 被存储在消息队列 230(诸如 SQL 表)中,每一消息组与组队列 240(例如,也可以是 SQL 表)中一个条目相关联。

[0016] 发送消息:

[0017] 发送子系统客户机 250、252、254 可以调用 Sender(发送者)对象,Sender 在下文中使用 API(应用程序编程接口)来定义。例如,Sender 对象可以被定义如下:

[0018] public Sender(Type queueAccessType);

[0019] public Sender(Type queueAccessType, string serverId);

[0020] public Guid BeginMessageGroup(int messageType, int priority, Guid correlationId);

[0021] public void Send(object messageBody);

[0022] public void EndMessageGroup();

[0023] public void CancelMessageGroup();

[0024] 首先,发送子系统客户机创建 Sender 对象。构造函数的参数一般是提供 IQueueAccess 接口(在下文中描述)的类的 System.Type(系统类型)。替换构造函数可以采用 serverId(服务器 ID),serverID 可以是标识发送消息的服务器实例的任意串。如果项目服务器中仅有一个实例在给定的机器上运行,那么 serverID 可以是机器名。发送子系统客户机一般不使用 ServerID 值。

[0025] 一旦构造了 Sender 对象,客户机调用 BeginMessageGroup()(开始消息组)。messageType(消息类型)参数是可以用于区分共享公共队列的不同类型的消息的客户机定义的值。例如,消息类型可以是 Save(保存)、Publish(发布)和 Report(报告)类型的消息。客户机可以为这些消息类型定义某些任意的常量,并且该值可以用来分支到接收方上正确的消息处理代码。

[0026] priority(优先级)参数可以用于为消息组指定优先级值。例如,值“1”可以是最高优先级,而较大的数字可以是较低的优先级。

[0027] correlationId(相关 ID)参数一般是用于使分离的消息组相关的客户机提供的 GUID。例如,根据预定的顺序(诸如以它们被接收的顺序)处理来自特定用户的消息(诸如来自时间表条目程序的消息)。每一时间表的提交可以作为一个消息组发送。排队服务器 200 可以使用相关 Id 来确保即使存在处理时间表消息的多个线程,这些组中的每一个也是根据预定的顺序(诸如按顺序的次序)来处理的。在该时间表示例中,correlationId 值可以是 ResourceID(资源 ID)。

[0028] BeginMessageGroup() 的返回值是组的 UID。UID 可以用于涉及消息组状态的查询。

[0029] 在调用 BeginMessageGroup 之后,发送子系统客户机可以使用 Send() 来发送任何数量的消息。messageBody(消息正文)参数一般是表示正被发送的消息的任意对象。messageBody 参数可以包括消息数据,以及甚至用于在接收方处理数据的方法。然而,该任意对象应该能够根据预定的顺序来组织(诸如能够被串行化)。排队服务器 200 一般维护对象的串行化和反串行化,以存储在队列数据库表中。队列数据库表可以被优化,使得如果消息已经是二进制格式的(例如,字节数组),那么系统可以将数据“按原状”写入数据库。

[0030] 一旦发送了所有消息,客户机可以调用 EndMessageGroup()(结束消息组)。EndMessageGroup() 关闭该组,并标识该组为准备好由接收代码(在下文描述)来处理。

[0031] 如果为某种原因而需要丢弃未完成的组,那么可以调用 CancelMessageGroup()(取消消息组)。CancelMessageGroup() 可以用来从队列中移除在调用

BeginMessageGroup() 之后发送的所有消息。尽管在调用 EndMessageGroup 之后不能向消息组添加另外的消息,但是可以通过仅再次调用 BeginMessageGroup 来重复使用 Sender 对象本身。由 Begin/EndMessageGroup 对之间发送的所有消息定义的消息组一般不被处理,直到成功调用 EndMessageGroup。倘若在发送消息时系统崩溃,将在队列中留下未完成的组。该未完成的组可以由下文描述的清除进程来成功地移除。

[0032] 接收消息:

[0033] 接收者系统被设计成作为从队列中检索出消息并然后调用适当的处理代码的线程来工作。作为结果,处理消息的代码的开发者一般不需要使用由以下 API 定义的任何 Receiver(接收者)对象。该 API 用来在项目服务器被启动或由管理员手动激活时设置接收服务。Receiver 对象可以定义如下:

[0034] public Receiver(Type queueAccessType, Type processMessageType, string serverId);

[0035] public void Start(bool asynchronous);

[0036] public void Stop();

[0037] public bool Done;

[0038] public int ServiceLimit;

[0039] public int ThreadSleepTime;

[0040] Receiver 对象是使用 queueAccessType(队列访问类型)、processMessageType(处理消息类型)和 serverID 参数来构造的。queueAccessType 参数是支持 IQueueAccess 接口的对象的 System.Type, processMessageType 是支持 IprocessMessage 接口的对象的 System.Type,而 serverID 一般是用于标识接收者所属的服务器实例的任意串。如果每个机器仅使用一个实例,那么 serverID 可以是机器名。(这些构造函数参数的操作将在下文更完全地描述。)

[0041] 一旦构造了 Receiver 对象,那么可以通过调用 Start() (启动) 来启动它。一般接收者应该多半运行行为线程,从而“asynchronous(异步)”参数应该为“true(真)”,尽管它可以通过传递参数“false(假)”来同步运行。可以实现替换实施例,使得系统可以总是异步地运行。

[0042] 不论同步还是异步启动,接收服务可以通过调用 Stop() (停止) 来关闭。调用 Stop 指导接收服务关闭任何活动的工作者线程并然后退出。当接收者线程以及与该接收者相关的所有工作者线程实际完成时,Done(完成)属性可以被设为“true”。

[0043] ServiceLimit(服务限制)是可以用来指定接收者可以使用多少工作者线程的读/写属性。ServiceLimit 可以被设为默认为值“1”。ServiceLimit 一般可以在接收者线程运行时被更改。

[0044] ThreadSleepTime(线程睡眠时间)是指定接收者线程在每次向队列轮询新消息之间睡眠多长时间(例如,以毫秒为单位)的读/写属性。ThreadSleepTime 可以被设为默认为值 1000,当使用毫秒为单位时即一秒钟。

[0045] 由于线程的考虑,发送者和接收者一般为 queueAccess 和 processMessageType(而不是对象的实例本身)采用 System.Type 值。如果 Sender 和 Receiver 采纳它们请求的对象的实例,那么这些对象必须是完全线程安全的,这意味

着这些对象引用的任何其它对象也必须是线程安全的。为了避免这个问题, Sender 和 Receiver(使用传递的 System.Type 对象)为每一产生的线程例示指定对象的副本。从而,只有提供 IqueueAccess 或 IprocessMessage 的对象应该必须是部分线程安全的(这意味着它们必须保护对静态类成员的访问,但不保护对任何实例对象的访问)。

[0046] 处理消息:

[0047] 当启动接收服务时,它采用的参数之一是支持 IProcessMessage 接口的对象的 System.Type, IProcessMessage 接口定义如下:

[0048] public interface IProcessMessage{

[0049] bool ProcessMessageCallback(Message msg, Group messageGroup, MessageContext mContext);

[0050] void StartMessageGroup(Group messageGroup, MessageContext mContext);

[0051] void EndMessageGroup(Group messageGroup);}

[0052] 每一 Receiver 对象的工作者线程例示了客户机对象的一个实例。在读取任何消息之前,调用 IProcessMessage.StartMessageGroup() (处理消息的启动消息组)。这向客户机给出在处理消息组之前进行所需的初始化或其它处理的机会。下一步,对消息组中的每一消息,调用 IProcessMessage.ProcessMessageCallback() (处理消息的处理消息返回)。

[0053] ProcessMessageCallback 的返回值指定了该消息是否被成功处理。如果返回了值“true”,那么该消息可以出队。返回值“false”指示该消息失败,且排队系统然后可以使用同一消息再次调用 ProcessMessageCallback。

[0054] 当接收线程完成读消息(或者通过接收最近的消息,或者由于异常而中止)时,可以调用 IProcessMessage.EndMessageGroup() (处理消息的结束消息组)。这向客户机给出进行可能所需的任何清除或结束的机会。

[0055] Message(消息)参数一般是消息本身。该对象含有以下属性:

[0056] public int MessageId;

[0057] public object MessageBody;

[0058] MessageId 给出了消息的 ID,该 ID 在该组中唯一。组中第一条消息一般具有 ID 1,而该组中第二条消息也具有 ID 1,依此类推。MessageBody 是最初通过 Sender.Send() 方法发送的对象。

[0059] Group(组)参数是一般含有关于消息所属的组的元数据的对象。Group 参数提供以下属性:

[0060] public Guid GroupId;

[0061] public Guid CorrelationId;

[0062] public int MessageType;

[0063] public int Priority;

[0064] public int LastMessageId;

[0065] GroupId(组 ID)是该组的唯一标识符。每一消息组具有唯一的 GroupId。CorrelationId 是消息的相关组的唯一标识符。它对应于当调用 Sender.BeginMessageGroup() (发送者的开始消息组)时传递给 Sender.BeginMessageGroup() 的

参数。MessageType 是指定消息类型的客户机定义的枚举类型。MessageType 对应于传递给 Sender.BeginMessageGroup() 的参数。Priority 是组的优先级。Priority 对应于传递给 Sender.BeginMessageGroup() 的参数。LastMessageId(最后消息 ID) 给出了组中最后一条消息的 MessageId。从而当遇到组中最后一条消息时 msg.MessageId 与 messageGroup.LastMessageId 相同。

[0066] MessageContext(消息上下文) 参数可以包含关于消息的当前处理状态的额外信息。MessageContext 参数可以定义如下：

[0067] public int RetryCount;

[0068] public int GroupRetryCount;

[0069] public int SleepBeforeRetry;

[0070] RetryCount(重试计数) 是使用特定消息调用 ProcessMessageCallback 的次数。在第一次调用时,值一般为“1”。GroupRetryCount(组重试计数) 是特定组尝试处理的次数。如果由于某种原因接收者工作者线程异常中止处理组,GroupRetryCount 将重试该组,一般从第一条未处理的消息开始。SleepBeforeRetry(重试之前睡眠) 是客户机设定的参数,它告知工作者线程万一消息的处理失败,它在重试消息之前应该睡眠多长时间。

[0071] 低级队列访问：

[0072] 排队服务器 200 一般是队列存储“不可知的”(即,服务器不依赖于队列存储的内部工作方式)。排队服务器 200 通过支持 IQueueAccess(队列访问) 的对象进行队列读/写/出队访问,IQueueAccess 的 System.Type 被传递给 Sender 和 Receiver 对象的构造函数。IQueueAccess 可以定义如下：

[0073] public interface IQueueAccess{

[0074] // 获取 MSPSQS 应该处理的下一组

[0075] void GetNextGroup(out Group msgGroup);

[0076] // 从队列中移除指定组

[0077] void PopGroup(Group msgGroup);

[0078] // 返回队列中 groupId 与 msgGroup.GroupId 相等的下一条消息

[0079] void ReadMessage(Group msgGroup,out Message msg);

[0080] // 由 msgGroup 和 msg 指定的消息出队)

[0081] void PopMessage(Group msgGroup,Message msg);

[0082] // 给定组元数据,打开消息组

[0083] void openGroup(Group msgGroup);

[0084] // 将由 msgGroup 和 msg 指定的消息入队

[0085] void SendMessage(Group msgGroup,Message msg);

[0086] // 关闭消息组

[0087] void CloseGroup(Group msgGroup);

[0088] // 从消息队列和组队列中删除给定组的所有记录

[0089] void FlushGroup(Group msgGroup);

[0090] // 为由名为 serverId 的服务器实例锁定发送的组解除锁定)

[0091] void UnlockServerGroups(string serverId);}

[0092] IQueueAccess 的默认实现可以使用两个 SQL 表（例如，消息队列 230 和组队列 240）来构建。然而，由于排队服务器 200 的剩余部分与队列访问的分离，整个系统可以被构建在任一合适的存储系统（诸如 MSMQ 或 Yukon SSB）之上。

[0093] 队列存储：

[0094] 消息队列 230 是持有从发送子系统客户机发送的消息的实际消息正文数据的表。对每一 Sender.Send() 调用，一般向该表添加一行。尽管附图将消息组示为群集在一起，但是实际上，来自多个组的消息可以是交错的。消息队列 230 一般用作数据容器。经常在该表上执行的操作是插入 (Insert) 新的消息、删除 (Delete) 处理过的消息和选择 (Select)（读）下一消息。该表中的记录对应于上述 Message 对象。

[0095] 所示实施例示出了消息队列 230 的一个表，但是替换实施例可以按所需的将该表分割成多个表。替换实施例仅需增强 IQueueAccess.ReadMessage、IQueueAccess.PopMessage 及 IQueueAccess.SendMessage 的实现来实现多个表。第二张表，即组队列 (240) 用于选取下一组、处理和管理组及相关锁、以及持有额外的元数据。该组中的记录对应于上述 Group 对象。在消息队列 230 和组队列 (240) 中的记录之间经常存在多对一的关系。消息队列 230 一般含有对发送的每一个别消息的记录。组队列 (240) 一般对发送的每一消息组含有一条记录。

[0096] 向队列添加新组

[0097] 当发送子系统客户机调用 Sender.BeginMessageGroup() 时，新的记录被添加到该组队列。该记录被标记为“锁定发送”，来确保没有接收者在该组完成之前会开始处理该组。当客户机调用了 Sender.Send() 任意次数后，可以调用 Sender.EndMessageGroup()，它使用发送的最后一条消息的 ID 来更新组记录，并将该组标记为未锁定。

[0098] 获取下一可用的组：

[0099] 图 3 示出了根据本发明的各方面，示出用于获取下一可用组的进程的操作流程图。一般而言，该进程寻找这样的最高优先级的组，其相关组（共享相关 ID 的那些组）没有锁定接收，其组没有锁定发送，且其相关不含有拥有更高优先级的、在锁定发送的队列前端的组。该进程的逻辑可以依据排队系统的使用方式而不同。

[0100] 该进程在开始框 300 处开始，并前进至框 310，在那里获取其相关没有锁定接收的所有组的列表。该列表可以按照队列中的位置和特定消息组的优先级来排序。

[0101] 在框 320 中，作出该列表是否非空的判断，并且如果在列表中存在组，那么前进至框 340。如果没有组存在于列表中，该进程前进至框 330 并终止。

[0102] 在框 340 处，获取列表中的第一组，并在框 350 中评估该组来确定该框是否锁定发送。如果该框被锁定发送，那么该进程前进至框 360，在那里从列表中移除共享锁定组的相关 ID 的所有组。如果该框没有被锁定发送，那么该进程前进至框 370，在那里该框被锁定发送，并被返回至调用进程。

[0103] 获取下一可用组：

[0104] 图 4 示出了根据本发明的各方面，示出用于使用接收者和工作者线程的进程的操作流程图。一般而言，接收者线程监视队列的可用组。在接收者线程开始轮询队列之前，它首先对按照被传递给构造函数的 serverId 参数的服务器实例名称锁定发送的任何组解除锁定。组被解除锁定来允许清除 / 恢复进程。如果接收者线程异常停止（例如，由于机器

重引导),当前处理的组将以其它方式保持锁定发送。从而,一旦服务器实例重新启动,这些组就被解除锁定。

[0105] 之后,接收者线程向队列轮询可用组。当有一组可用,且工作者线程的数量小于允许的最大值,那么接收者线程可以创建工作线程来处理该组。如果工作者线程成功地处理该组,那么该组项从组队列中移除。如果没有成功处理,那么工作者线程重新启动该组。

[0106] 该进程在开始框 400 处开始,并前进至框 405,在那里按照接受者所属的且 serverId 相关联的服务器实例对任何锁定的线程解除锁定接收。

[0107] 在框 410 中,作出关于是否有可用的任何工作者线程的判断。如果有任何可用的工作者线程,该进程前进至框 415,在那里获取下一个可用的组。如果没有可用的工作者线程,那么该进程前进至框 430,以检查工作者线程的状态之前进行等待。

[0108] 在框 420 中,作出关于是否获取下一可用组的判断。如果获取了下一可用组,那么该进程前进至框 425,在那里启动工作者线程。当工作者线程被启动之后,该进程返回至框 410。如果没有获取下一可用组,那么该进程前进至框 430,以在检查工作者线程的状态之前进行等待。

[0109] 在框 430 中的等待之后,该进程前进至框 435,在那里检查工作者线程的状态。在框 440 中,作出关于是否已经处理了下一可用组中的所有消息的判断。如果不是下一可用组中的所有消息尚未被处理,那么该进程前进至框 445,在那里重新启动工作者线程。如果下一可用组中的所有消息都已经被处理,那么该进程前进至框 450 处,在那里从组队列中移除该组。

[0110] 在框 445 中重新启动工作者线程或者框 450 中从组队列中移除组之后,该进程前进至框 455,在那里作出是否还存在要检查的工作者线程的判断。如果还存在要检查的工作者线程,那么该进程返回至框 435,在那里检查工作者线程的状态。如果不存在要检查的工作者线程,那么该进程返回至框 410,在那里再次作出关于是否有任何可用的工作者线程的判断。

[0111] 消息处理工作者线程:

[0112] 图 5 示出了根据本发明的各方面,示出用于使用消息处理工作者线程的进程的操作流程图。一般而言,消息处理工作者线程循环操作。当消息位于消息队列上时,消息处理工作者线程读消息并调用 `IProcessMessage.ProcessMessageCallback()`。如果该函数返回“true”(指示消息的成功处理),那么从队列中移除该消息。否则重试。

[0113] 该进程在开始框 500 处开始,并前进至框 505,在那里读消息。在框 520 中,作出关于是否成功获取消息的判断。如果成功地获取了消息,那么该进程前进至框 530。如果没有成功获取消息,那么该进程前进至框 560 并终止。

[0114] 在框 530 中,处理成功获取的消息,且该进程前进至框 540,在那里作出关于是否成功地处理了该消息的判断。如果消息没有被成功地处理,那么该进程返回至框 530,在那里再次处理获取的消息。(可以在进程逻辑中放置检查,以当消息不能被正确地处理时异常中止处理。)如果消息被成功地处理了,那么该进程前进至框 550,在那里该消息从队列中“出队”(即,被移除并调整相关联的指针)。在消息被出队之后,该进程返回至框 510,在那里读取另一条消息。该进程反复,直到不再获取消息,在该情况中,然后处理在结束框 560 处终止。

[0115] 从子系统故障中恢复：

[0116] 解决了子系统故障的两种情况：其中接收机器被重新引导并返回在线的情形，以及其中接收机器崩溃并且在相当的一段时间内没有返回在线的情形。（相当一段时间可以是可以向另一机器分配由崩溃的机器执行的负载的长度的时间。）

[0117] 可通过当组队列 240 中的组条目被锁定接收时，使用接收者的实例名称（传递给构造函数的 serverId 参数）标记它，来使得能够从发送子系统崩溃中恢复。在重新引导的情形中，在接收服务重启之后的第一个动作一般是对由该示例锁定接收的任何组解除锁定。然后可以挑选组并由任何可用的接收服务来处理该组。

[0118] 当接收子系统崩溃并且在相当的时间内没有返回时，那么（如同发送者崩溃的情况）一般存在失去时效的、被标记为用于接收的一个或多个组。这种情况的清除是通过对这些组解除锁定来实现的。一旦这些组被解除锁定，可以挑选这些组来由消息处理线程继续处理。

[0119] 以上说明书、示例和数据提供了本发明的组成部分的制造和使用的完整描述。由于可以实现本发明的多个实施例，而不背离本发明的精神和范围，因此本发明由此后所附的权利要求书定义。

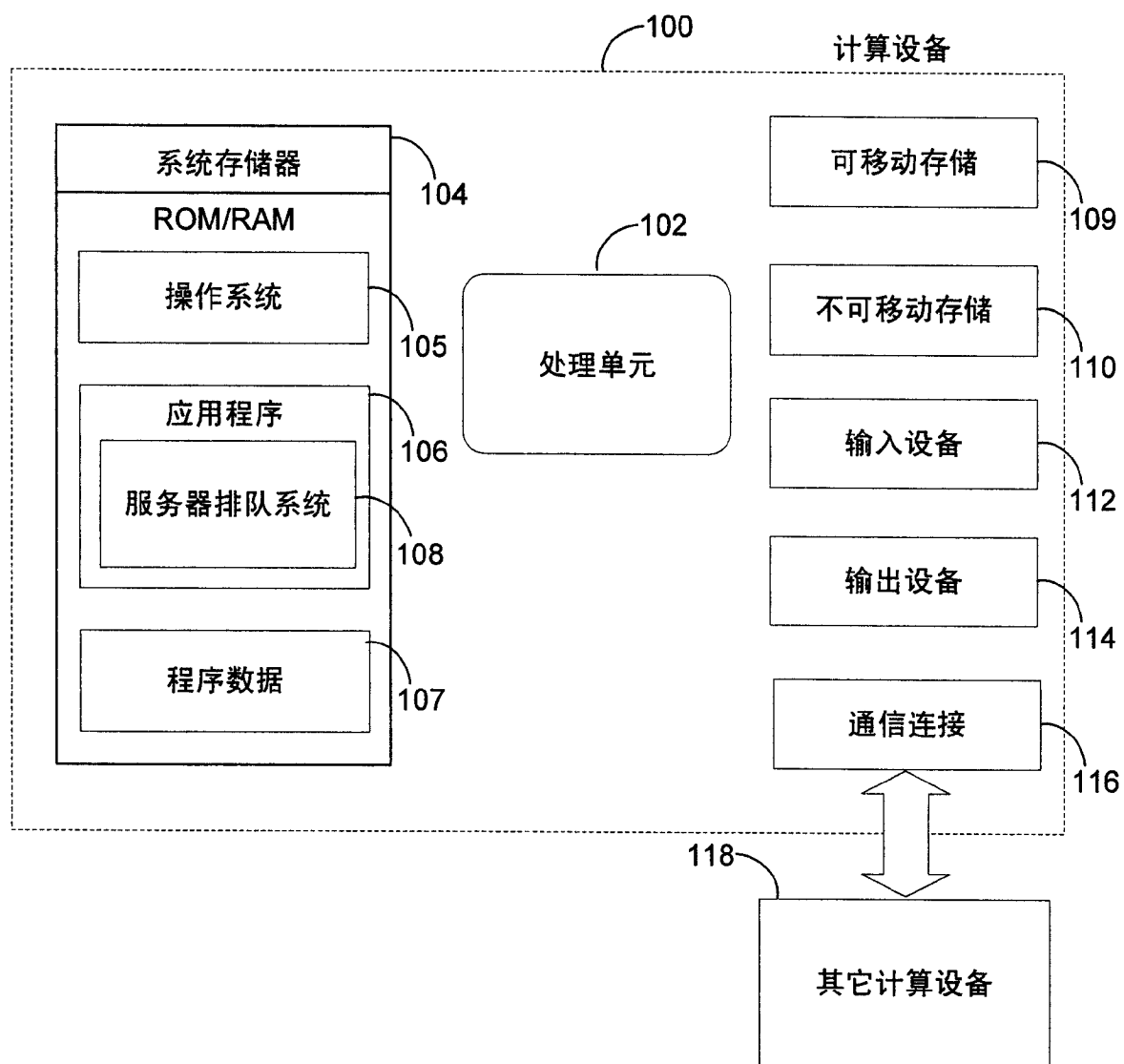


图 1

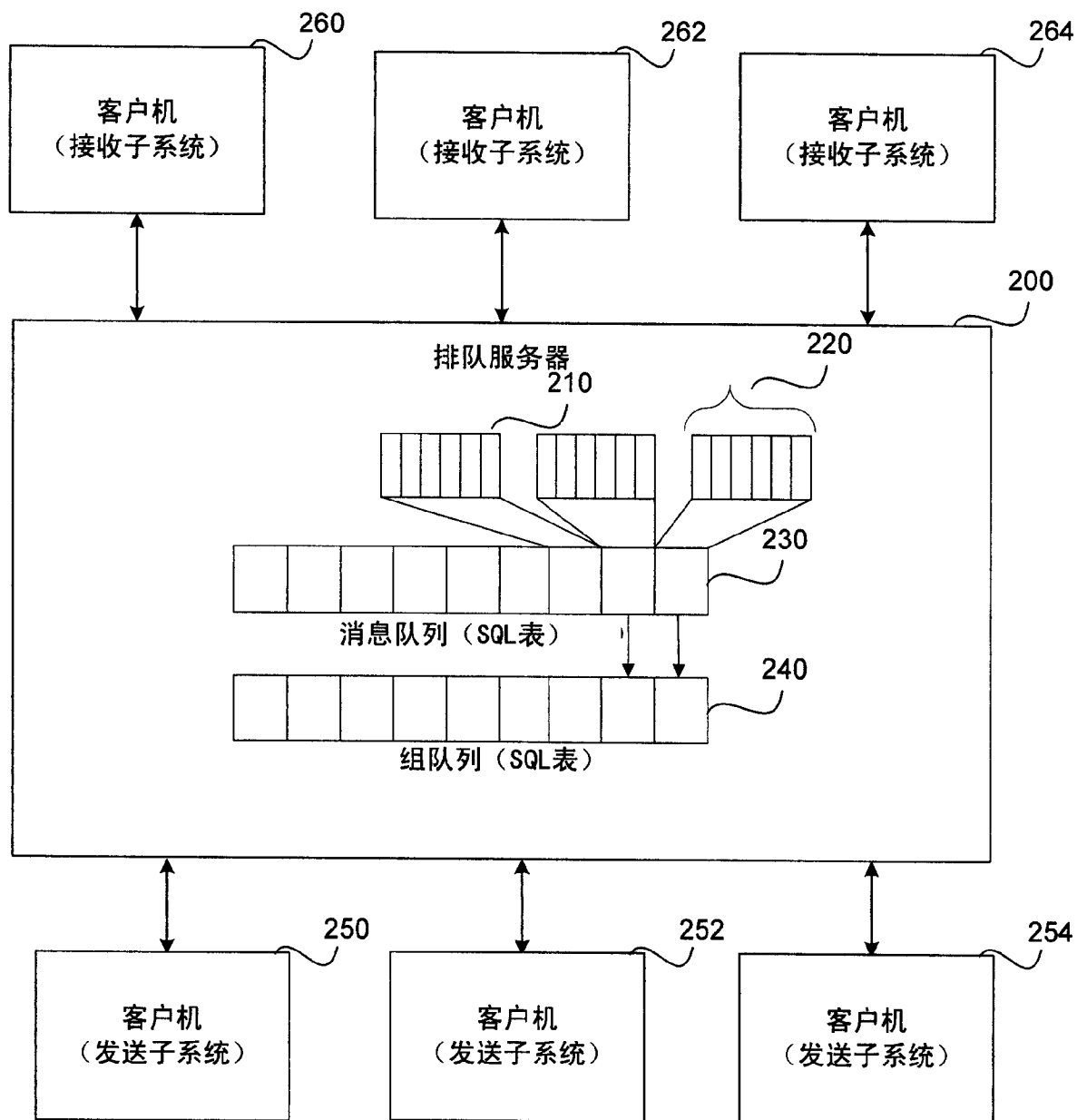


图 2

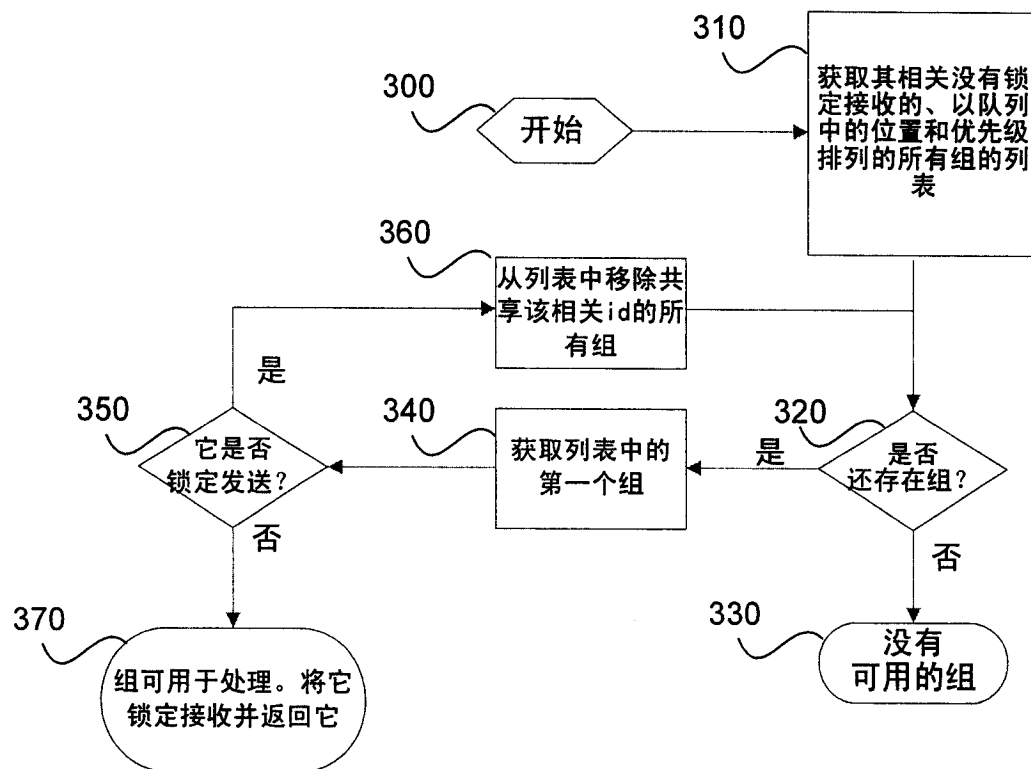


图 3

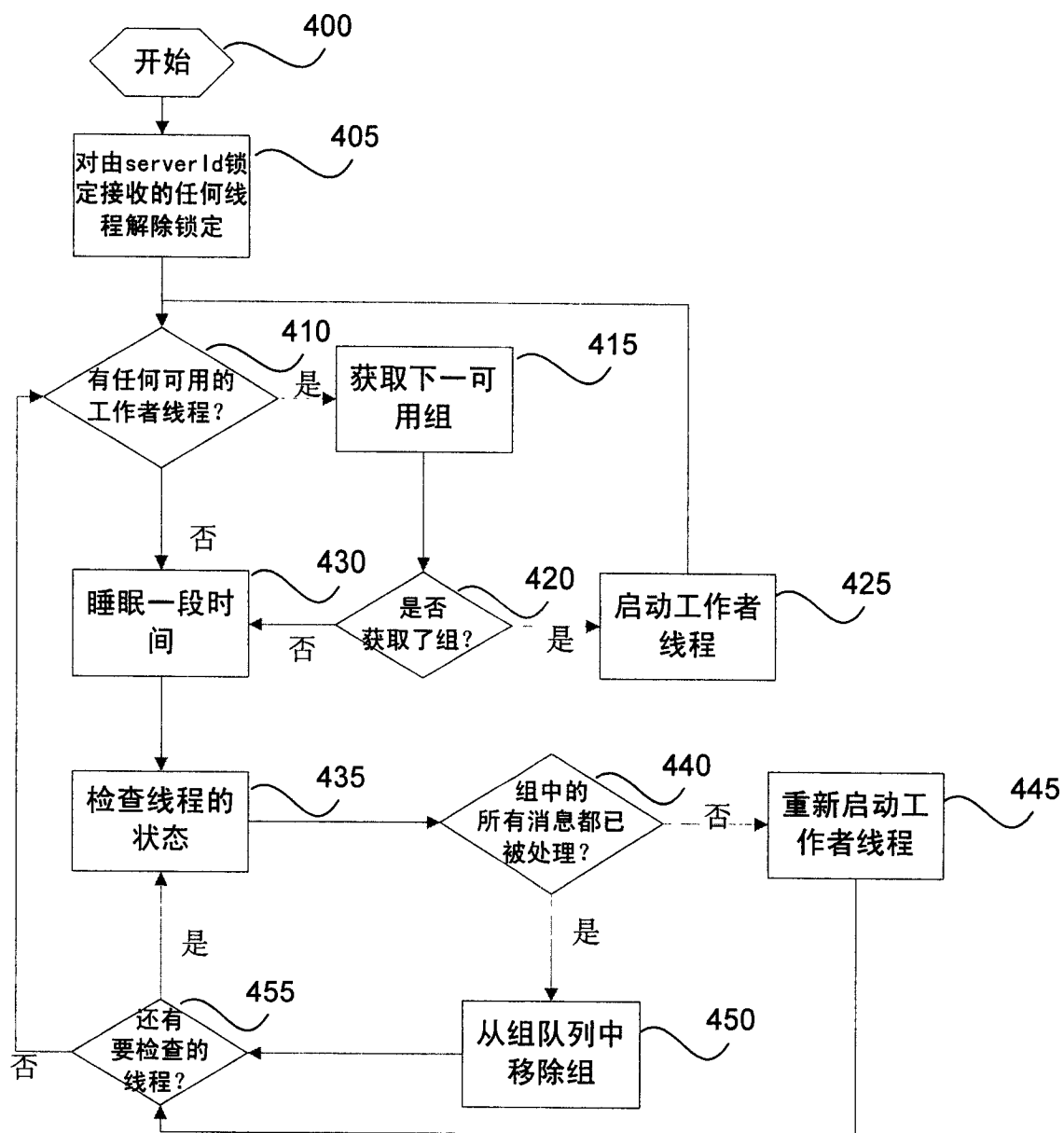


图 4

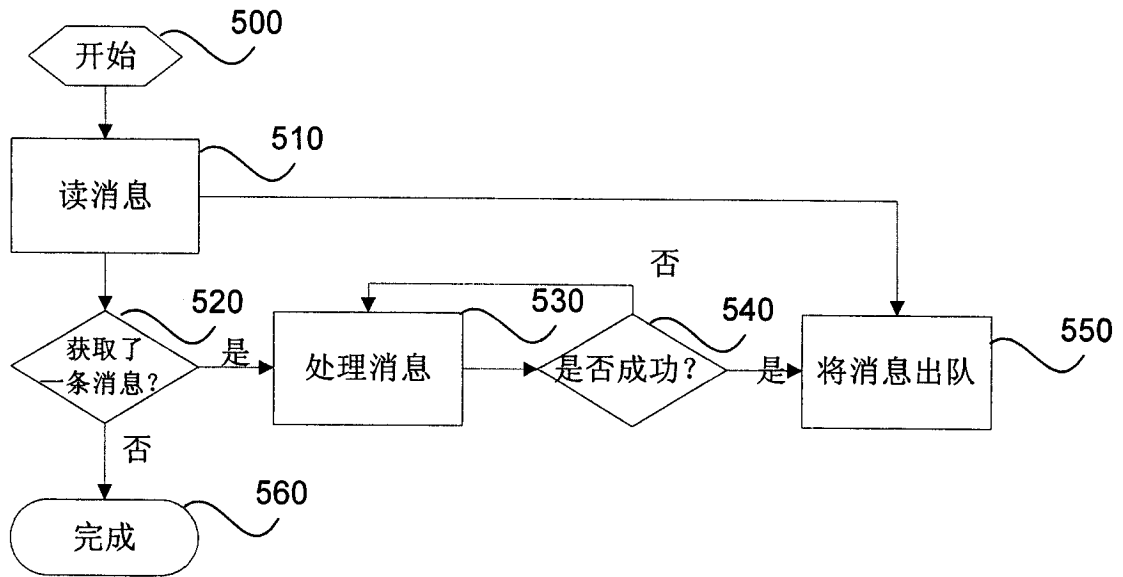


图 5