[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 15/16

[12] 发明专利申请公开说明书

[21] 申请号 03823544.7

[43] 公开日 2005 年 10 月 26 日

[11] 公开号 CN 1688989A

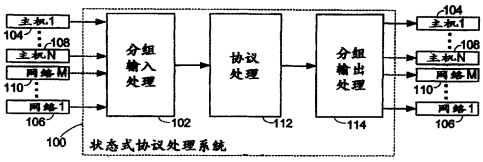
[22] 申请日 2003.7.10 [21] 申请号 03823544.7
[30] 优先权
[32] 2002. 8. 2 [33] US [31] 10/211,434
[86] 国际申请 PCT/US2003/021583 2003.7.10
[87] 国际公布 WO2004/013720 英 2004.2.12
[85] 进入国家阶段日期 2005.4.1
[71] 申请人 机敏网络公司
地址 美国加利福尼亚
[72] 发明人 法西尔·伊斯梅特·奥斯曼
西蒙·约翰·尼
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 董 莘

权利要求书 10 页 说明书 35 页 附图 5 页

[54] 发明名称 高数据率的状态式协议处理

[57] 摘要

本发明公开了在数据通信系统中用于根据一个或多个状态式协议处理消息的方法、系统和装置。属于诸如 TCP 传输层“连接”的各种通信流的消息根据所选择的状态式协议被处理。 属于单一流的消息在不同时间被分配到不同的协议处理核心(PPC)。 流到 PPC 的分配一般在具体的单个流的级别上是可修改的，从而允许 PPC 之间的灵活的负载均衡。 系统的体系结构通过提供管道和并行处理结构的结合来实现所述方法。



ISSN1008-4274

1. 一种在处理多个消息流的状态式协议处理系统中处理数据的方法，其中每个流与唯一对应的流标识相关联，所述流标识由属于这样的流的消息传递，所述方法包括:

a）接收属于特定流的多个消息;

b）从所述多个消息得出与所述特定流相关联的事件;

c）根据所述特定流的状态式协议，具体地分配第一协议处理核心来处理一个或多个所述事件；以及

d）根据所述特定流的状态式协议，具体地分配不同的第二协议处理核心来处理一个或多个其他事件。

2. 根据权利要求 1 所述的方法，其中在步骤 c）和 d）中的分配协议处理核心还包括根据算法来均衡不同协议处理核心之间的负载。

3. 根据权利要求 2 所述的方法，其中所述均衡不同协议处理核心之间的负载的算法包括在一组协议处理核心之间轮循分配。

4. 根据权利要求 2 所述的方法，还包括:

e）在每个协议处理核心的本地队列中存储某些数量的事件；以及

f）识别相关的协议处理核心组中的低负载协议处理核心，所述低负载协议处理核心在其本地队列中存储的事件少于所述相关组中的不同协议处理核心的队列中存储的事件；以及

g）所述均衡不同协议处理核心之间的负载的算法包括分配所述低负载协议处理核心来处理当前没有事件被分配给协议处理核心的流的事件。

5. 根据权利要求 1 所述的方法，还包括:

e）并发地向所述第一协议处理核心分配不同的流的组合的事件；以及

f）向所述第二协议处理核心分配所述不同流的组合之一的随后事件，同时所述第一协议处理核心继续处理所述不同流的组合中的另外事件。

6. 根据权利要求 1 所述的方法，还包括：

e）并发地向所述第一协议处理核心分配不同的流的组合的事件；以及

f）显式地释放所述第一协议处理核心，使其处理所述不同流的组合之一的事件，同时所述第一协议处理核心继续被分配来处理所述不同流的组合中的另外事件。

7. 根据权利要求 1 所述的方法，还包括：

e）接收未分配的流的消息，其中所述未分配的流的事件处理当前没有被分配给任何协议处理核心，所述未分配的流具有对应的流标识；

f）识别多个被配置成处理所述未分配流的事件的通用类型的事件的多个兼容协议处理核心；以及

g）从所述兼容协议处理核心之间选择协议处理核心来处理所述未分配流的一个或多个事件，而不考虑对应于所述未分配的流的流标识。

8. 根据权利要求 1 所述的方法，还包括：

e）接收未分配的流的消息，其中所述未分配的流的事件处理当前没有被分配给任何协议处理核心；以及

f）从与所述事件兼容的协议处理核心之间选择协议处理核心，来处理所述未分配流的一个或多个事件，而不考虑当前排队以等待被所述多个兼容协议处理核心处理的流的流标识。

9. 根据权利要求 1 所述的方法，还包括在选择协议处理核心来处理特定事件之前，显式地确定所述特定事件是否属于当前分配了协议处理核心的流。

10. 根据权利要求 9 所述的方法，还包括如果确定先前选择的协议处理核心当前被分配到对应的流，则通过所述先前选择的协议处理核心来处理所有会影响所述对应的流的状态的进入事件。

11. 根据权利要求 9 所述的方法，还包括释放对先前的协议处理核心的分配，使所述协议处理核心处理特定流的事件，而不考虑任何流的流标识。

12. 根据权利要求 1 所述的方法，还包括基于当前没有相关的影响第一流的状态的事件正在被处理的指示，释放对先前的协议处理核心的分配，使所述协议处理核心处理所述第一流的事件。

13. 根据权利要求 12 所述的方法，其中所述当前没有相关的影响第一流的状态的事件正在被处理的指示基于对所述第一流的相关事件的引入和完成而进行计数的计数器。

14. 根据权利要求 1 所述的方法，还包括在选择协议处理核心来处理所述事件之前，确定事件的类型。

15. 根据权利要求 1 所述的方法，还包括:

e）在为属于各种流的接收消息得出的多个事件选择协议处理核心之前，确定事件类型，所述确定的多个不同的事件类型至少包括:

i）使用第一状态式协议的流的良好事件;

ii）使用不同的第二状态式协议的流的良好事件; 和

iii）被确定有错误的使用所述第一状态式协议的流的事件。

16. 根据权利要求 15 所述的方法，其中所述确定的不同事件类型还包括 iv）分组段。

17. 根据权利要求 1 所述的方法，还包括再循环封装在第一消息中的信息，用作第二状态式协议消息以在所述状态式协议处理系统中进行状态式协议处理。

18. 一种在处理多个数据通信消息流的状态式协议处理系统中处理数据的方法，其中每个流与唯一对应的流标识相关联，所述流标识由属于这样的流的消息传送，所述方法包括：

a）接收属于特定流的消息以及属于其他流的消息；

b）从所述接收的与流相关联的消息中得出事件，所述流由得出所述事件的消息的流标识来指示，所述事件包括与所述特定流相关联的事件以及与所述其他流相关联的事件；

c）将每个事件设置在一组一个或多个预处理队列的其中一个中；

d）分配第一协议处理核心来处理所述特定流的第一事件，而不考虑所述第一事件所位于的预处理队列，随后将所述第一事件传递到所述分配的第一协议处理核心的本地队列；以及

e）分配不同的第二协议处理核心来处理所述特定流的不同的第二事件，而不考虑所述第二事件所位于的预处理队列，随后将所述第二事件传递到所述分配的第二协议处理核心的本地队列。

19. 根据权利要求 18 所述的方法，其中所述第一和第二协议处理核心基本上专用于执行状态式协议消息处理。

20. 根据权利要求 18 所述的方法，其中步骤 b）还包括基本上将所述接收的消息的有效载荷数据从所述得出的事件中排除。

21. 根据权利要求 20 所述的方法，其中步骤 b）还包括在不包括事件类型指示的事件中设置事件类型的指示。

22. 根据权利要求 18 所述的方法，还包括 f）在将事件传递到协议处理核心之前校验事件的完整性。

23. 根据权利要求 18 所述的方法，还包括在物理上与所有的状态式协议处理核心不同的分组处理器中执行步骤 b）。

24. 根据权利要求 23 所述的方法，还包括 f）通过执行所述分组处理器未执行的、且所述协议处理核心也未执行的程序步骤的查阅处理器的动作，为当前未被分配由协议处理核心处理的流的事件确定本地代理标识。

25. 根据权利要求 24 所述的方法，还包括 f）在取决于所述本地代理 ID 的存储器地址访问流状态，对于所述当前未被分配由协议处理核心处理的流，所述流状态已经在先前被存储。

26. 根据权利要求 18 所述的方法，f）通过专用查阅硬件的动作，至少对属于当前未被分配由特定协议处理核心处理的流的事件，确定本地流代理 ID。

27. 根据权利要求 26 所述的方法，其中所述专用查阅硬件包括微处理器。

28. 根据权利要求 18 所述的方法，还包括：

f）在唯一地与所述第一协议处理核心相关联的第一存储器中，保持所述第一事件的数据和相关联的流状态数据；以及

g）在由不同协议处理核心可访问的存储器中，并发地保持不同流的事件的数据和所述不同流的对应流状态的数据，而不考虑由所述第一协议处理核心访问所述第一存储器。

29. 根据权利要求 28 所述的方法，还包括在协议处理核心本地的存储器中保持事件数据和对应的流状态数据，其中所述协议处理核心被分配来处理所述的事件，直到由所述分配的协议处理核心执行的处理结束。

30. 根据权利要求 18 所述的方法，还包括基于用于在不同的协议处理核心之间均衡负载的算法，分配协议处理核心来处理事件。

31. 一种在处理多个消息流的状态式协议处理系统中处理数据的方法，其中每个流与唯一对应的流标识相关联，所述流标识由属于这样的流的消息传送，所述方法包括：

a）实现接收属于特定流的多个消息的步骤；

b）基于所述多个消息定义事件的步骤；

c）实现将特定流的第一事件分配到第一协议处理核心进行状态式协议处理的步骤；和

d）实现将所述特定流的第二事件分配到不同的第二协议处理核心进行状态式协议处理的步骤。

32. 根据权利要求 31 所述的方法，还包括确定是否有任何协议处理核心当前正在处理与新事件的流标识相关联的流的事件的步骤。

33. 根据权利要求 31 所述的方法，还包括确定对于与特定流标识相关联的所有当前事件来说处理何时已经完成。

34. 一种端接数据通信传输层的系统，包括：

a）消息接收器模块，被配置成

i）接收多个数据通信消息，每个这样的消息与由所述消息的流标识唯一地标识的流相关联；

ii）从所述接收的消息得出事件，其中每个事件与流相关联，所述流与所述得出事件的消息相关联；

b）多个协议处理器核心模块，被配置成根据事件的状态式协议来处理所述事件；和

c）事件调度模块，用于在所述多个协议处理器核心模块之间调度事件，所述事件调度模块被配置成：

iv）从所述消息计数器模块接收事件；

v）对于每个接收的事件，确定是否协议处理器核心模块当前被分配来处理所述接收的事件流的事件；以及

vi）从多个兼容的协议处理器核心模块中选择协议处理器核心模块来处理接收的事件，其中所述选择与特定接收的事件流的流标识无关，并且所述接收的事件在步骤 ii）中被确定为当前没有分配任何协议处理器核心以进行处理。

35. 根据权利要求 34 所述的系统，其中所述事件调度模块还包括查阅子模块，所述查阅子模块被配置成确定接收的事件流的本地代理标识，所述接收的事件在步骤 ii）中被确定为当前没有分配任何协议处理器核心以进行处理。

36. 根据权利要求 34 所述的系统，其中所述事件调度模块还包括：

d）事件跟踪子模块，被配置成：

i）存储关于被调度到协议处理器核心模块的事件的流处理状态消息，以确定是否协议处理器核心模块当前被分配来处理特定流的事件。

37. 根据权利要求 36 所述的系统，其中所述事件跟踪子模块（d）还被配置成

ii）根据所述事件的流保持事件的计数，以及

iii）当特定流的事件被调度到被分配给所述流的协议处理器核心模块时，增加所述特定流的计数，以及

iv）当指示所述分配的协议处理器核心模块已经完成所述特定流的事件的处理时，减少所述特定流的计数。

38. 根据权利要求 34 所述的系统，还包括 e）暂存式存储器，被配置成存储与从所述消息得出的事件相独立的消息的有效载荷数据。

39. 根据权利要求 34 所述的系统，还包括 e)消息输出处理模块，被配置成根据来自处理输出消息流的协议处理器核心模块的指令，组装所述输出消息。

40. 根据权利要求 34 所述的系统，其中所述消息接收器模块还包括被配置成在事件中建立事件类型字段的处理器。

41. 一种根据数据通信状态式协议处理消息的设备，所述设备包括：

a）用于从物理层接收消息的装置，所述消息对应于一个或多个通信流；

b）多个用于所述一个或多个通信流中的特定通信流的信息的状态式协议处理的装置；

c）用于在所述多个状态式协议处理装置中选择其中一个来处理所述特定通信流的信息，以均衡处理器负载的装置，其中所述选择与在所述特定流的消息中提供的流标识数据无关。

42. 根据权利要求 41 所述的设备，其中

c）所述用于选择的装置包括：

i）用于分配特定的状态式协议处理装置来处理所述特定通信流的信息的装置，和

ii）用于释放所述分配的状态式协议装置使其不再处理所述特定通信流的信息，而不需释放所述分配的状态式协议装置使其不再处理任何其他通信流的信息的装置。

43. 根据权利要求 42 所述的设备，还包括 d）用于从对应流的接收消息中得出所述对应流的事件的装置。

44. 根据权利要求 43 所述的设备，其中所述用于选择的装置 c）还包括用于确定何时对于已经被传递到分配的协议式处理装置的流的所有事件的处理完成的装置。

45. 一种状态式协议处理设备，包括：

a）多于一个的协议处理核心微处理器，每个协议处理器核心微处理器包括：

i）基本上足够存储流状态的本地存储器，所述本地存储器可由协议处理器核心访问，并且这种访问并不干扰其他的协议处理器核心；和

ii）被配置成在所述设备的操作期间容纳指令代码的程序存储器，以足够使得所述微处理器根据适当的状态式协议来处理接收的数据消息，所述处理包括响应于所述接收的数据消息中的状态相关信息，更新存储在本地存储器中的相关流状态；和

b）调度器，包括用于接收数据消息的输入端以及处理电路，所述处理电路被配置为至少在所述设备的操作期间，

iv）将与特定流相关联的第一接收数据消息的信息转发到第一协议处理核心进行状态式处理，以及

v)将至少与所述特定流相关联的不同第二消息转发到不同的

第二协议处理核心进行状态式处理。

46. 根据权利要求 45 所述的设备，其中每个协议处理核心还包括双端口本地存储器。

47. 一种在数据通信系统中用于在多个兼容协议处理器之间分配属于对应通信流的通信事件的方法，所述方法包括:

a）接收属于用以分配的队列中的第一流的第一事件;

b）在与所述第一事件兼容的多个协议处理器之间选择第一协议处理器来处理所述第一事件;

c）将来自所述队列的第一事件传递到所述第一协议处理器;

d）将来自所述队列的属于其他流的其他事件传递到所述第一协议处理器;

e）接收属于所述队列中的特定流的第二事件;

f）从所述多个兼容协议处理器中选择不同的第二协议处理器来处理所述第二事件；以及

g）将来自所述队列的第二事件传递到所述第二协议处理器，同时所述第一协议处理器继续处理来自所述队列的其他事件。

48. 根据权利要求 47 所述的方法，还包括响应于所述第一协议处理器完成已经被传递到所述第一协议处理器的所述第一流的所有事件，释放所述第一协议处理器使其不再处理所述第一流的事件。

49. 根据权利要求 47 所述的方法，还包括:

h）在步骤 c）之后并在所述第一协议处理器已经完成对所述第一事件的处理之前，接收所述特定流的第三事件；以及

i）随后选择所述第一协议处理器来处理所述特定流的第三事件。

高数据率的状态式协议处理

技术领域

本发明涉及数据传输处理系统。

背景技术

数据传输系统一般通过各种层传送数据，每个层执行不同类型的处理。不同层的数目和性质根据给定通信系统所遵从的概念模型而不同。例子包括由国际标准化组织（ISO）为开放系统互连（OSI）定义的具有七层的模型，以及美国国家标准化学会（ANSI）定义的被称为“光纤信道”模型的五层模型。已经提出了许多其他具有不同数目的层的模型，它们执行某种程度上不同的功能。在大多数数据通信系统中，层包括从物理层到应用层，其中经由物理层，包含数据的信号被传输和接收，经由应用层，高级程序和处理共享信息。在大多数概念层模型中，传输层存在于这些极端之间。在这样的传输层中，执行协调数据传输所需的功能，所述数据可能已经在各种物理链路上发送，以分发到更高层的处理。

在传输层中，通信系统协调分别属于特定“流”或者这样的消息的组的许多消息（例如分组）。每个消息可以由它与特定流标识键（流键）的关联来标识，流标识键一般由关于通信端点的信息来定义。传输层处理一般由被称为传输层终端（TLT）的处理模块来执行处理，TLT 根据由为每个特定流所选择的由传输层协议（TLP）所定义的一组规则来管理从远程 TLT 接收的数据(或者传输到远程 TLT 的数据)。TLT 检查它所处理每个消息的与流状态有关的信息，相应地更新流状态，并基于流状态将消息重构为对于消息目的地适合的形式，所述流状态定义了消息所属的流的状态，所述消息目的地一般为远程 TLT 或者本地主机。流一般为双向通信，所以从远程 TLT 接收属于特定流的

消息的 TLT 一般也向远程 TLT 发送属于同一流的消息。通过保持对应的流状态以根据所选择的 TLP 的对整个流进行管理将传输层处理与链路层处理区别开，链路层处理一般只涉及个体消息。

有许多公知的 TLP，例如光纤信道、SCTP、UDP 和 TCP，以及许多很有可能在未来开发出来的 TLP。TLP 一般用于保证可理解的和准确的信息通信，这是例如通过检测和请求丢失或损坏消息的重传，将流的各种消息重组织为期望的顺序，和/或向目标提供有关通信的相关实事来实现的。传输控制协议（TCP）有可能是最公知的 TLP 的例子，并且广泛地使用于诸如互联网和以太网的网络中。TCP 是面向连接的协议，当连接活动时，必须在连接端点（终端）保持关于连接状态的信息。连接状态信息包括，例如，拥塞控制信息，用于确定分组是否应该重发的定时器，确认信息和包括源和目的地表示和开放/关闭状态的连接标识信息。从而每个活动的 TCP 连接具有唯一的连接 ID 和连接状态。TCP“连接”是这里称为“流”的更一般的状态式协议处理系统（“SPPS”）的例子，而 TCP“连接 ID”和“连接状态”分别是这里称为“流键”和“流状态”的更一般的 SPPS 概念的例子。TLP 中的流键可以唯一地由远程链路（目的地）地址（一般是互联网地址或“IP”地址）、远程（目的地）TCP 端口号、本地链路（源）地址（一般也是 IP 地址）、本地（源）TCP 端口号以及某些情况下的接收机接口 ID 的组合来指定。为了区别具有相同地址但是使用不同 TLP 的流，将协议指示作为通用流键的一部分也是有用的。

数据通信也可以发生在除了典型的传输层以外的许多层。例如，iSCSI 通信发生在传输层以上的层，但是通信包括属于流的状态式消息，从而在某些方面与传输层通信相似。

随着计算机无论在本地（例如，在局域网上）还是在广域（例如互联网上）的日益增加的互连，对于数据通信系统的更高的数据速率有持久的需求。为了达到更高的数据速率，在传输层和其他地方中的状态式协议需要相应的更快的处理。更快的硬件当然可以成比例地增加处理速度。但是，只增加硬件速度将不能有成本效率地按照期望增

加协议处理速度，从而需要对于给定的硬件速度能够通过自身的体系结构和方法进行更快的处理的协议处理系统。

发明内容

这里描述了用于状态式协议处理的方法、系统和设备。状态式协议处理需要保持“状态”以跟踪数据的“流”的状态。特定流的状态经常被更新以反映属于该流的各个数据“消息”，并且流本身由流键来标识，所述流键显式地或者隐含地与属于该流的每个消息相关联。每个消息的协议定义了考虑到流的当前状态，要在每个消息上执行的处理步骤。

在一个方面，本发明涉及一种在被配置成处理多个消息流的状态式协议处理系统（“SPPS”）中处理数据的方法，其中每个流与唯一对应的流键相关联，该流键由属于这样的流的消息传送。该方法包括接收属于特定流的多个消息。接着从接收的消息得出与特定流相关联的各种 SPPS 事件。该方法还包括根据特定流的状态式协议（SP），具体地分配第一协议处理核心(“PPC“)来处理特定流的一个或多个所述事件。另外，根据特定流的状态式协议，具体地分配不同的第二 PPC 来处理特定流的一个或多个其他事件。

这里所描述的 SPPS 可以以许多方式实现，下面将描述一些示例性实施例的细节。一个实施例是包括将特定流的事件（给，从消息得出的信息）的处理（不同于来自更一般的流类的事件）分配到 PPC 的方法。另一个实施例是包括将流的事件的处理分配到 PPC 而与事件所位于的预队列无关，接着将事件传递到所分配的 PPC 的本地队列的方法。另一个实施例是包括接收消息的步骤，基于接收的消息定义事件的步骤，向第一 PPC 分配流的第一事件的步骤和向第二 PPC 分配流的第二事件的步骤的方法。

另一个实施例是用于端接数据通信传输层的系统，其包括消息接收器模块，其被配置成接收各具有流 ID 的消息，并且从消息得出事件（事件可以，例如，简单地是所接收的消息，但一般包括对所接收的消息的内容或者形式的一些修改）。该实施例还包括一些 PPC 模块

和调度器模块，该调度器模块被配置成接收事件，确定是否 PPC 模块已经被分配来处理同一流的事件，如果否的话，就选择已经被配置为与事件兼容的 PPC 来处理事件而不关心与事件相关联的流 ID。

另一个实施例是用于处理消息的装置，其包括用于接收消息的装置，多个用于流信息的状态式协议处理的装置，以及用于与流 ID 无关地选择处理特定流的信息的 PPC。另一个实施例是包括多个 PPC 微处理器的装置，每个 PPC 微处理器被配置成执行对属于所分配的流的消息的 SP 处理，并且具有用于保存这样的分配的流的流状态的本地存储器。该实施例还包括调度器，该调度器接收消息信息，并且将至少一个消息的流状态相关的部分转发到第一 PPC 微处理器，并且将至少另一个消息的流状态相关部分转发到第二 PPC 微处理器。

在 SPPS 的上下文还描述了新的子系统，该新的子系统可以在该 SPPS 中使用。一个例子是数据调度器子系统，另一个例子是事件得出子系统。这些子系统的每个又可以具有进一步的子系统；例如，事件调度器子系统可以包括用于跟踪流的流消息的子系统，所述流尤其是那些当前未被 PPC 处理的流，以及/或者用于跟踪当前正在被 PPC 处理的信息的核心活动管理器子系统。事件调度器子系统可以基于 PPC 负载，引导属于单个流的数据事件分配到多个并行的协议处理器核心，并且可以在没有协议处理器核心正在处理特定流的数据事件时保持该特定流的流状态。

在下面的附图和说明中提出了实施例的细节和一些替换实施例。因为这里不可能描述本发明的所有实施例，所以所描述的实施例必须理解为描述本发明，而不是限制本发明。

附图说明

图 1A 是示出了到状态式协议处理系统的通用接口连接的框图。

图 1B 是典型的计算机系统中的传输层终端系统的框图。

图 2 是诸如图 1 的状态式协议处理系统的一个状态式协议处理系统的更详细的框图。

图3是示出了图2的状态式协议处理系统的一些特征的进一步的细节的框图。

图4是用于改变被选择来处理流的协议核心的动作的流程图。

图5是调度器模块响应于接收了属于流的事件而执行的动作的流程图。

图6是调度器模块响应于接收了来自协议核心的“完成声明”而执行的动作的流程图。

在不同的图中类似的幅图标记表示类似的部件。

具体实施方式

在该说明书中，为了说明本发明的使用和实现，描述了实施例和变化。说明性的描述应理解为表现了本发明的例子，而不是限制本发明的范围。

状态式协议处理需要处理以这里称为“消息”的可识别和可区别的单元到达的数据。多个消息属于“流”，“流”是分别与唯一地标识流的“流键”相关联的一组消息。这里描述的用于状态式协议处理的方法和装置在多个不同的流并发到达时最有用。无论流的消息是否正在被处理，只要还期待另外的消息，流就是“活动的”，当不再期待处理属于该特定流的消息时，流就变成“不活动的”。

“状态式协议”定义了根据被保持以反应流的情况的“状态”来处理属于流的消息的协议。至少一些（一般是许多）属于流的消息会影响流的状态，从而，状态式协议处理包括检查进入的消息对它们所属的流的影响，相应地更新流的状态（或者“流状态”），并根据消息所属的流的当前状态处理由适用的协议所指定的消息。

根据TCP（传输控制协议）处理数据通信是状态式协议处理的一个例子。TCP流一般称为“连接”，而消息是分组。与每个分组相关联的流键主要包括流键端点地址（例如，源和目的地“套接字地址”）。对于每个活动的连接（或者流）保持流状态，流状态被更新以反映被处理的流的每个分组。根据流状态和TCP处理规则来执行对数据的真

实处理。

TCP 是通常用于 TLT（传输层终端）系统的协议。典型的 TLT 接收分组中的消息，并且从分组的首标中所包含的信息识别消息所属的流和要对该消息进行处理的协议。但是，可以以其他方式来提供将消息与其所属的流和将要处理它的协议相关联的信息，例如间接或隐含地从该消息所关联的其他消息获得，或者通过获得该消息的特定源来获得（例如，如果知道特定主机一次只有一个流是活动的，则隐含地，来自该主机的每个消息都属于该主机的活动的流）。

另外，如这里所描述的状态式协议处理可以用于除了 TLT 系统以外的地方，在这种情况下，关于流和协议的信息可以在除了进入分组首标以外的其他地方提供。例如，进入 TCP 分组可以封装了在不同的处理“层”要根据完全不同的协议被处理的数据。因此，在这里所描述的 TLT 系统的上下文中实现的状态式协议处理提供了通用状态式协议处理系统（“SPPS”）的具体例子。属于一个状态式协议流的消息可以，例如被封装在属于不同的状态式协议的消息中。被称为“SCSI”的公知的通信协议提供了在传输层以外的层的数据通信的例子。通常在主机和诸如磁盘驱动器的外围设备之间使用 SCSI。SCSI 通信可以发生在专用于 SCSI 通信的专用连接上，或者它们可以经由不同的层被封装和传输。SCSI 可以被封装在一些传输层协议的消息中，例如，光纤信道和 TCP。“FCP”是 SCSI 消息被封装在光纤信道所用的协议，而“iSCSI”是 SCSI 消息被封装在 TCP 消息中所用的协议。FCP 和 iSCSI 都是状态式协议。

这样的已封装一个例子涉及属于诸如iSCSI流的第一状态式流的信息，该信息在属于诸如 TCP 连接的不同的第二状态式流的本地网络上传递。第一 SPPS 可以跟踪封装 TCP 连接（流）的状态。相同的SPPS或者不同的第二SPPS可以确定由封装流传输的某些消息形成属于封装 iSCSI 流的更高级的消息。封装 iSCSI 流的流键可以被包含在每个已封装消息中，或者它可以由来自传输信息的已封装 TCP/IP 分组的流键的隐含来确定。给定已封装流的流键和用来处理已封装流的

协议（iSCSI）的知识，SPPS 可以保持 iSCSI 的状态，并且可以根据所指定的协议（在该例子中是 iSCSI）来识别和处理相关联的消息。

这样，传输层终端系统可以提供 SPPS（状态式协议处理系统）的一个好例子。实际上，TLT 可能至少包括某个状态式处理，从而被认为是 SPPS。但是，只要处理包括根据为流定义的状态式协议更新多个流所属的流的流状态，SPPS 就可以用于其他数据通信层和其他类型的处理。从而，尽管主要针对 TLT 系统描述了本发明，但是应该注意不要不适当地认为本发明限于 TLT。

图 1A 图示了到 SPPS 100 的接口连接。SPPS 分组输入处理块 102 可以从许多信源接收分组形式的数据。源一般包括诸如“主机 1”104 的主机连接和诸如“网络 1”106 的网络连接，但是单个系统可以使用许多其他主机连接和网络连接，它们用“主机 N”108 和“网络 M”110 来表示。协议处理块 112 根据对于每个数据流的适合的规则（即，诸如由公知的 TCP 定义的状态式协议规则，用于被规定根据这样的状态式协议来处理的状态式消息）处理进入数据。流一般涉及双向通信，所以数据通常被传输到每个主机连接和/或网络连接并从其传输。因此，分组输出处理块 114 将数据传递到分组输入处理块 102 从其接收数据的同一组连接（“主机 1”104 到“主机 N”108 和“网络 1”106 到“网络 M”110）。

图 1B 提供了到实现在计算机系统 152 中的 TLTS 150 的连接的概要图，其中 TLTS 150 提供了简单 SPPS 例子。单个主机系统 154 经由使用公知 SPI-4 协议的连接 156 连接到 TLTS 150。主机 154 表现如图 1A 所示的任何主机 104 到 108，向 TLTS 150 发送消息并从 TLTS 150 接收消息。TLTS 150 经由另一个 SPI-4 协议连接 160 连接到媒体控制层（“MAC”）设备。MAC 158 经由适合的连接 164 连接到网络 162。MAC 在 TLTS 的数据（这里是 SPI-4 格式的数据）和由网络 162 的连接 164 所使用的物理信号之间进行转换。网络 162 可以具有内部连接和分支，向并从远程通信源和/或目标传递数据，例如“源/目标系统 1”170、“源/目标系统 2”180 和“源/目标系统 3”190。许多通信源/

目标可以通过特定网络访问。源/目标系统可以类似于计算机系统 152。更复杂的源/目标系统可以具有多个主机和网络连接，例如图 1A 所示。从而，某些源/目标系统可以有效地将各种不同的网络连接到一起。

图 2 是示出了示例性的 SPPS 200 的模块的框图。在一个实施例中，两个 SPI-4 Rx 单元 202 和 204 在标准 SPI-4 16 位总线上接收数据，该总线符合 2001 年 1 月在加州，Fremont 的光互连工程论坛的“系统分组接口水平 4（SPI-4）阶段 2：用于网络和链路层设备的 OC-192 系统接口。实现约定 OIF-SPI4-02.0”（或者以后的版本）。只有连接的数量影响系统所需的整个处理能力时，连接的数量才是重要的，并且可以从一个接口连接到大量接口。每个单独的接口可以处理到多个网络和/或主机源的通信；分离的物理主机和网络连接不是必须的，但是在概念上和物理上是方便的。另外，虽然在一个实施例中为了方便使用 SPI-4，但在其他实施例中，可以替换地或者附加地(对应于输入框比如 202 和 204 中的处理)使用其他用于到物理层的接口的技术(例如，PCI-X）。

消息分路

仍参考图 2，由接口 202 和 204 接收到的数据被分别传输到消息分路器模块 206 和 208 处理。该传输一般发生在大小“B”的总线上。在该文献中，“B”表示总线大小，可以为了工程方便而选择总线大小以满足速度和布局限制，并且 B 不表示单个值，而是表示从 16 到 128 比特的范围。消息分路器模块 206 和 208 可以执行服务的组合。例如，它们可以重新组织在突发中分段接收的进入的消息，并且可以从分组的源和内容中识别分组的类型，并向消息添加某些数据以为后面的处理阶段简化类型识别。它们还可以将进入消息分路为“有效载荷”数据和“协议事件”（下面简称为“事件”）数据。

随着数据从 SPI-4 接口到达，诸如 206 和 208 的消息分路器模块可以将所有的数据经由适合的宽度 B 的总线移动到暂存式存储器 210 的已知单元中。或者，它可以只向暂存器发送有效载荷数据，或者整

个消息的其他子集。暂存器 210 可以被配置为各种方式；例如，它可以用作公知的先入先出（FIFO）缓冲器。在更复杂的例子中，暂存器 210 可以被组织成有限但有用数目的页面。每个页面可以具有相对短的暂存器引用 ID，通过该 ID 可以定位存储在该暂存器中以这样的页面开始的有效载荷（或者消息）。当有效载荷超过一页时，可以在该页面的末尾提供指示，以识别要连接的下一页面，通过该方式，在可以由第一页面的引用 ID 所标识的一个或多个页面的块中可以容纳任意长度的有效载荷（或消息）。通常有效载荷长度是所接收的消息的首标信息的一部分。暂存器引用 ID 可以提供基址，并且该负载可以以预定方式设置在被基址引用的存储器中。有效载荷隐含地在有效载荷长度的末尾结束，并且这对于独立地跟踪暂存器所接收的字节数目以与在首标指示的有效载荷长度相比较来校验是有益的。如果暂存器还接收消息的首标，可以类似地通过引用暂存器引用 ID 来访问首标。当然，在此情况下，可以容易地在暂存式存储器模块 210 中执行有效载荷校验，但是一般可以在许多其他地方执行这样的校验，例如在源消息分路器（206，208）中，在调度器 212 中，或者在 PPC 216－222 中，从数据处理的角度，这可能是方便的。

事件得出

消息分路器 206，208 的典型的功能是从进入消息得出与消息的状态式处理最相关的信息，以及将该信息格式化并设置“事件”中，该“事件”与得出该消息的流相关。例如，根据许多传输层协议，包括流标识、握手、长度、分组顺序和协议标识的“状态相关”数据被设置在分组首标的已知位置中。每个状态式协议消息具有与它所属的流的相关的信息，并且这样的状态相关的信息将被定位于它可以被识别的位置。（应当注意的是，执行状态式协议处理的系统也可以执行无状态消息。例如，TLP 一般也处理不与建立的流相关联，从而不影响流状态的分组，例如地址请求协议，或者 ARP 分组。这样的“无状态”分组可以由与当前描述的实施例兼容的任何技术来处理。但是，因为重点

是影响消息流的流状态的状态式消息的处理，所以这里不进一步讨论这些技术。）

由诸如206或者208的消息分路器模块从进入消息得出的事件可以采取广泛的形式。在简单的例子中，在一些实施例中，它可以是整个消息。更典型地，事件可以排除对做出 SP 处理不是必须的某些信息。例如，经常可以排除有效载荷并分开处理，从而事件可以简单地是所接收的消息的首标。但是，在一些实施例中，可以从首标添加或者删除信息，并且结果可以被重格式化，以产生在 SPPS 中方便处理的得出的事件。

事件定型

例如，接口（202，204）或者消息分路器（206，208）模块可以在某种程度上检查接收的消息，并且该检查的结果可以用来得出事件的“类型”。例如，如果根据分组所属的流中要求的协议，分组没有差错检验异常，则从这样的包中得出的事件可以被标识上反映协议和消息的视在有效性的事件“类型”字段。从而，被 SPPS 处理的每个不同的协议可以具有特定的“类型”，并且该信息可以被包括在事件中，以简化关于随后处理的判断。可以定义的另一种类型是消息段；这样的段一般可以保持不处理，直到消息的其余部分到达。根据事件的协议，消息段可以具有子类型，但不是必须的。可以定义的另一种事件类型是有差错的消息。因为事件“类型”可以用于指导事件的随后处理，所以应该进行不同的处理的有差错的消息可以标识为一般差错的子类型。作为一个例子，差错类型事件可以被标识有反映事件的 SP 的子类型。

反映随后处理的消息（或得出事件）的任何特征可以是消息定型的候选者。从而，从工程的角度，适合于 SPPS 实施例，消息定型可以是很简单的，或者可以是复杂的。事件定型是在得出事件时可以对接收的消息信息进行的扩充的一个例子。其他的扩充可以包括修改或添加校验和，或者提供基于接收消息的有效性做出的各种检验的成功

或者失败的指示。也可以增加相关位置，例如指示在暂存存储器 210 中何处可以找到消息信息的暂存器位置。注意，如果使用 SPPS 的消息源，例如主机，被设计来在它传输到 SPPS 的消息中提供一些或者所有的这样的“扩充”信息，则消息分路器可以不必真实地添加该信息以获得“扩充的”事件。

除了扩充消息信息，事件得出还可以包括重格式化事件信息以允许 SPPS 对事件的更方便的操作。例如，处理可以对于某些类型的事件（例如，在某些系统中的 TCP 事件）优化，并且得出其他类型的事件可以包括重格式化以适应这样的优化处理。接着，通常，可以通过对接收消息不作任何处理，或者通过扩充和/或重格式化消息的信息（尤其是状态相关的信息）以辅助后面的处理步骤来得出事件。例如，对于 TCP，得到的事件可以主要包括去掉了不必要的信息的主要的首先 256 字节的分组，被添加以反映其被复制到的暂存位置的信息，差错检验的结果以及事件定型。如果主机被配置为以方便的形式准备数据，则从消息分路器发出的得到的主机事件可以是消息的第一字节（例如，首先的 256 个字节），只有很少的变化或者没有变化。

使用运行微代码的嵌入式处理器来实现消息分路器功能可能是方便的，其自身重编程而不需要改变设备设计。但是，消息分路器功能可以可替换地通过在通用处理器中或者在专用集成电路（ASIC）或者以任何其他适合的形式执行的软件来实现。

对于由诸如 206 和 208 的消息分路器模块执行的特定的处理步骤集，可能有许多替换。例如，在消息分路器可以确定流 ID（即，在 SPPS 中足够标识流的表示消息的流 ID 的号码，它对于本地代理更有用）的“本地代理”并将其添加到事件－在所描述的实施例中的后面的处理块中执行的步骤。并且，进入消息不必要完全被分路。相反，进入消息可以保持在一起；例如，它们可以完整地存储在暂存式存储器中以对系统的许多部分可用，或者它们可以被完整地直接转发到事件调度器 212，接着转发到下面将详细描述的协议处理核心（PPC）216－222。如果进入消息未被分路，则这些模块 206，208，可以被重命

名为例如“分组预处理器”以减少混乱。本领域技术人员应该理解，在许多情况下，在复杂的系统中，设计的方便主要地决定了哪个模块执行任何特定动作。

事件调度器

如图 2 所示，有消息分路器 206，208 准备的事件被转发到事件调度器模块 212，在这里它们进入队列。事件调度器模块 212（或者简称调度器）可以通过基于随着消息到达的流标识“键”进行对本地流 ID 代理的搜索来开始处理进入事件。

本地流 ID 代理

流标识键（或者简称“流键”）根据流所使用的 SP（例如，TLP）唯一地标识了消息所属的流。流键可以很大（典型地对于 TCP 为 116 比特），这样，它的格式对于定位有与特定流相关的 SPPS 所保持的信息可能是不方便的。本地流 ID 代理可以替代地用于该目的。本地流 ID 代理（或者简称“本地代理 ID”、“本地流 ID”或者“代理 ID”）通常包括在 SPPS 中唯一地标识特定流的足够的信息，并且对定位与特定流相关的 SPPS 中的信息更有用。例如，可以选择本地流 ID 代理以用作进入流状态存储器 214 的索引以定位在 SPPS 中保持的关于特定流的信息（例如，流状态）。本地流 ID 代理不仅可以是用于 SPPS 的流的更方便的表示，它一般还可以更小。

可以在调度器模块或者其他地方，例如在前面描述的消息分路器模块 206 和 208 中，确定本地流 ID 代理。例如，在大 TLTS（传输层终端系统）中，假如必须保持的很大量的本地流 ID 代理，则确定代理 ID 是件不小的任务。如果这样，从工程角度，通过如下面描述的独立的“查阅”模块来做出这样的确定是方便的。在一些实施例中，这样的查阅模块可以是消息分路器模块 206 和 208 的子模块，或者可以是调度器模块的子模块，或者可以被最好地设计为对各种其他模块独立，并可以由各种其他模块访问。

对于从被配置为在网络上的流消息中包括本地流 ID 代理而不包括（或者还包括）通常的 SP 流键的主机所接收的事件，对于本地流 ID 代理的搜索可以被简化或者消除。这样的主机配置可以降低否则就要确定本地流 ID 代理的任何模块比如调度器的工作量。另一种降低本地流 ID 代理查阅负担的方法是保持最近使用过的流 ID 和它们的相关联的代理的“快速列表“，并且对于每个到达的消息或者事件首先检查该列表。

如果对于到达的消息所属的流，不知道本地流 ID 代理或者流状态，则调度器 212 可以创建新的本地流代理 ID。在许多情况下，调度器（或者查阅子模块）可以接着为这样的新流初始化流状态。选择这样的代理 ID 作为存储器的表的条目的值是有用的，所述存储器可以用于在方便的存储器比如流状态存储器 214 中为这样的新流存储流状态。在大系统中，这样的存储器可以很大，需要特别的管理。

存储器

这里所描述的每个不同的“存储器”比如暂存式存储器 210 和流状态存储器 214，一般不仅包括原始存储器，还包括适合的存储器控制装置。但是，存储器控制器的功能一般不是本说明书的重点，其只要求存储器响应于请求，或者存储或者返回指定的数据块。因为这里所描述的 SPPS 可以能够并发处理几百万个活动流（或者可以限于处理几千甚至更少的活动流），并且因为一般的流状态可以是大约 512 字节，实现图 2 的 SPPS 可能需要若干 GB 的存储器。用于实现这样大的存储器的技术是已知并且持续发展，并且任何已知的或者后来开发的技术都可以与任何类型的存储器一起使用以形成图 2 的 SPPS，只要这样的存储器达到了适合的性能。如果存储器以基本独立的方式工作，则这些存储器彼此区别为不同的存储器。例如，不同的存储器可以是独立可寻址的，以使得寻址存储在一个存储器的数据项目不同时作为寻址存储在不同的存储器中的不相关的项目的前序。不同的存储器还可以是独立可访问的，以使得访问存储在一个存储器的数据项目

不同时作为访问存储在不同的存储器中的不相关的项目的前序。由于这样的独立性，在某些情况下，不同的存储器可以避免困扰公用（或者共享）存储器的数据访问瓶颈。

查阅子模块

图 2 所示的调度器模块 212 可以包括执行调度器任务的特定子集的子模块。例如，结合独立的“查阅”模块以执行基于包括在到达事件中的流键来查阅本地流 ID 代理的功能是有用的。调度器 212 的另一个功能可以是为活动流建立和保持流定时器，这可能是与每个流相关联的特定 SP 所要求的。当在被本地流 ID 代理索引的存储器中保持这样的流定时器是方便的时，查阅模块还可以方便地执行监视流定时器的功能。并且，当分配 PPC 处理流的事件时，调度器 212 可以向 PPC 提供流状态。如果类似地在存储器中被本地流 ID 代理索引的位置保持流状态，则这可以是查阅模块可以方便地执行的另一个功能。这样的查阅模块可以是独立的，或者它可以实际上是调度器的一个子模块。查阅模块还可以主要地与系统的其他部分相关联。例如，如果查阅任务在消息分路器模块 206 和 208 执行，则它可以主要地与消息分路器模块 206 和 208（或者甚至是其子模块）相关联，如果查阅任务主要在 PPC 216－222 执行，则它可以主要与 PPC 216－222 相关联。

查阅处理可以要求诸如散列查阅过程的大规模的处理以基于原始流标识或者“流键”来选择或者确定本地流 ID 代理。同样地，查阅模块（或者子模块）可以利用它自身的微处理器系统和支撑硬件来实现。当通过查阅模块（或者子模块）来执行流 ID 代理确定时，调度器可以向 PPC 分配并传递事件，而不等待确定完成，并且查阅模块可以在以后将（通过使用本地流 ID 代理获得的）流消息传递到分配的 PPC 而不与调度器进一步交互。

一旦建立了“查阅”或者其他子模块作为不同的实体，则为了设计的方便，它可以被配置成执行对调度器（或者其他它所位于的或者它所相关联的模块）有贡献的任何任务，实际上，在本说明书中，在许

多情况下，它可以执行对其他模块比如消息分路器模块有贡献的任务。在不同的功能模块之间移动功能的能力是复杂处理系统的共有特征，本领域技术人员应该理解，在模块之间移动功能一般不会使系统有大的变化。

调度器 212 或者它的子模块可以执行许多其他功能。例如，调度器可以从暂存式存储器 210 请求反映消息有效载荷的校验和，将该校验和与包括在覆盖被转换到事件中的消息的该部分的事件中的校验和合并，并将合并的校验和结合到事件中。在调度器 212 和其他处理块之间示出了足够用于此目的适当大小的总线。如同许多调度器功能，该功能可以在其他地方执行，例如在消息分路器模块 206 和 208 中执行，或者在后面的处理期间执行。

引导器子模块

可以创建另一个模块或者调度器子模块以执行为调度器所做的一些或者全部决定。这样的被称为“引导器”的子模块，可以执行在选择特定 PPC 中所涉及的步骤以处理的流的特定事件，并且对于整个 SPPS（状态协议处理系统），跟踪各种 PPC 中的活动流处理的状态。

由引导器子模块保持的“流处理状态”可以指示，例如，其他流的事件当前正在被 PPC 处理，或者在前一个（流的）事件的 PPC 处理之后生成的新的流状态当前正在被写入流状态存储器。它还可以指示流是否正在被分路，或者对这个流有待处理的定时器。例如，这样的流处理状态信息可以用于使得引导器模块在合适时延迟向 PPC 转发事件以避免在流的流处理状态表明它的流状态正在被从 PPC 写入到流状态存储器时重写流状态。一旦如流处理状态所反映的那样，流状态存储器的更新完成，则新的事件就可以被转发到 PPC。

引导器子模块的流处理状态信息还可以被用于例如在流正在被 PPC 处理时防止定时器期满被不适合地发出。如果处理事件的动作可以引起这样的定时器期满被取消，则这样的定时器事件不应该被发出。引导子模块可以在允许定时器事件被发送到 PPC 之前，引用流处理状

态信息，以使得这样的定时器事件只有在对于该流没有其他事件是活动的时才能发出。利用查阅子模块，作为不同的模块的引导器的组织可以允许调度器简单地将进入事件转交给引导器。

协议处理核心和总线－结构介绍

调度器 212 已经为消息建立了本地流 ID 代理，就根据与该流相关联的 SP，确定消息事件（或者消息分路事件未被分路时的整个消息）应该在何处被处理。在一些实施例中，大量的这样的 SP 处理是由协议处理核心（“PPC”）执行的。具有大量 PPC 的群由 PPC 216 到 218 表示，而 PPC 220 到 222 表示另一个 PPC 集群。虽然示出了两个 PPC 集群，但是可以使用任何数量的这样的 PPC 集群。例如，一个 TLTS 实施例可以只包括单个 PPC 集群，而复杂的 SPPS 实施例可以包括几百个群。图 2 中示出了群中的两个 PPC，但是在任何给定的群中可以有两个或者更多个 PPC，典型的是，每个群有 5 个 PPC。尽管其对于设计对称性是方便的，但是每个群中的 PPC 的数目不必相同。部分地选择特定组织的 PPC 成为群以通过降低总线拥塞来促进数据的转移。每个群可以利用群内核心间总线 224（或者 226）互连每个群的 PPC，并且每个群一般被总线 230 或者 232 连接到总线网络和控制块 228。调度器 212 和 PPC 之间的数据可以被总线网络和控制块组织起来。如下面将更详细地讨论的，总线网络和控制块 228 主要用作“纵横”交换器，其促进了各种模块之间的通信。

PPC（例如 216－222）一般包括处理器核心和使 PPC 处理提交给它的事件的微代码（即，用于处理器核心的某种形式的序列指令）。它们一般还包括足够保存 PPC 正在处理的流的相关流状态数据的本地存储器，PPC 可以访问该本地存储器而不干扰其他 PPC。在处理特定流的消息事件的 PPC 本地存储器中保持该流的许多或者全部流状态一般是方便的。PPC 本地存储器可以被组织成多个块或者“工作空间”，每个都能保存流状态。PPC 一般具有进入事件的队列，以及用于在队列中具有事件的被 PPC 并发处理的若干个不同的流的工作空

间。

这里表示的总线实际上被描述为双向的。但是，如果方便，总线可以实现为两个单向总线，在某些情况下，在两个方向上不是相等的比特宽度。这样，表示为具有宽度 B 比特的总线表示了在特定实现中可以为了方便而选择的总线宽度，并且可以是方向不对称的。对总线大小的考虑一般包括物理布局的空间和驱动器限制，和获得特定性能目标所需的所要求的流量。总线未被穷尽性地示出；例如，消息总线可以有用地（例如通过菊花链）连接在 TPTS 的所有的物理段之间，即使这样的总线未明确地在图 2 中示出。另外，如果 SPPS 实现为运行在通用处理系统上的软件或者固件形式的程序模块，而不是使用具有嵌入式微处理器的 ASIC 的典型实现，则在图 2 中表示的总线可以表示软件模块而不是硬件信号之间的数据传输。

向 PPC 分配事件

在本发明的一些实施例中，调度器 212 选择特定的 PPC 以处理与特定流相关联的事件。这样的分配有许多考虑。首先，PPC 必须是与讨论中该类型的事件兼容或者被配置为处理该类型的事件的 PPC 之一。可以在调度器或者在调度器的流处理状态子系统中，通过指示 PPC 所兼容的事件类型或协议的 PPC 的表来确定这样的兼容性，其中 PPC 所兼容的事件类型或协议又可以与进入事件的协议或者事件类型要求进行比较。在一些实施例中，在处理的另一阶段，例如在消息分路器模块中，事件被标记了它的“类型”的指示。接着，调度器只需要基于预定的事件的“类型”选择兼容的 PPC。一般，事件类型将被如此定义以使所有具有对于特定流的状态相关消息的消息也具有相同的事件类型，并且可以被相同的 PPC 处理。这样，可以从能够处理所指示的事件类型的 PPC 集合中选择 PPC。

根据一种算法从该兼容 PPC 集合中选择 PPC，所述算法可以是例如，比较 PPC 负载以找出最小负载的 PPC，或者以轮循的方式选择 PPC，或者随机选择 PPC。一般，每个流的事件被特定地导向一个

PPC，而不是导向作为一类流的成员的 PPC。这样的每个流的个体处理允许负载均衡，而与一类流的属性无关。当流被分配作为例如共享流 ID（或者流键）的某个特征的类的成员时，可能出现这样的情况，即大量这样的类需要被并发处理，超过了 PPC 的能力，而另一个 PPC 没有负载。当流被分配给具有大量成员的类时，这种效应可能加剧。虽然许多实施例唯一地分配流（类的大小为 1），但是在一些实施例中，将流分配到类，尤其是可以改变小类或者其成员关系以均衡负载的类是有效的。

如果提供了从到特定 PPC 的分配释放特定流的机制，则即使流已经被分配到一个大类，也可以实现类似的负载均衡的结果。在许多实施例中，流到 PPC 的分配和释放都是对于个别或者特定流而完成的。最后，即使向 PPC 分配流，或者从 PPC 释放流是对于流的类执行的，通过使类灵活地重分配以均衡负载也可以实现等效的结果。即，如果被分配给 PPC 的类可以在特定流的水平上被改变，则可以以很大的灵活性均衡负载。在每种情况下，都有可能单个单元地改变分配到 PPC 的流，使得流最终被分配到一个 PPC，而实际上与任何固定的类属性比如将流 ID 散列特定值的特性无关，并且类似地与可以分配到该 PPC（或者另一个 PPC）的其他流无关。

在选择 PPC 转换之后，调度器 212 将事件与关于流状态“工作空间”的指令一起转发到 PPC。如上所述，用于选择 PPC 的决定可以在调度器的引导器子模块中执行。在典型的实施例中，调度器 212 首先确定进入事件是否属于已经具有分配到特定 PPC 的事件的流。在一些实施例中，诸如跟踪 PPC 的活动的核心活动管理器的子模块可以执行该确定，而在其他实施例中，引导器子模块可以执行这些功能。在已经为进入事件的流的事件分配了 PPC 的情况下，进入事件一般被转发到同一 PPC，该 PPC 可能已经在它的本地存储器具有当前的流状态。

但是，如果当前没有 PPC 被分配到该流，则调度器选择诸如 PPC 216 的特定 PPC 来处理进入事件（或者将该流分配到该特定 PPC）。选择可以基于核心活动管理器的信息进行，核心活动管理器保持了可

以用于在各种（兼容）PPC 上均衡负载的活动状态。引导器子模块可以执行真实的分配和均衡决定，并且在一些实施例中，引导器和核心活动管理器基本上是具有专用处理器和程序代码以执行这些任务的单一子模块。分配可以简单地“轮循”最近最少接收事件的兼容 PPC，或者基于 PPC 队列的充满度等。

在 PPC 216 被分配来处理进入事件时，在 PPC 216 的本地存储器中选择工作空间，并且在所选择的工作空间中建立进入事件的流的当前流状态。工作空间的选择可以由调度器模块来完成（例如，通过它的引导器子模块），或者例如由下一可用的 PPC 来完成。流状态可以在所选择的工作空间中以任何方便的方式实现。例如，调度器可以经由调度器向 PPC 发送流状态（例如，作为查阅子模块的动作），或者 PPC 自身可以从存储器（例如，流状态存储器 214）请求流状态。事件一般可以从调度器 212 传递到 PPC 216 的输入队列，在暂存式存储器(如果有)中的事件有效载荷的大小和位置一般被传递到 PPC 216 中。如下面将信息描述的，具有了这些信息，PPC 216 能够在事件到达队列中时处理事件。当 PPC 216 完成对特定事件的处理时，在一些实施例中，它将向调度器 212 传输“完成”消息，使得调度器可以跟踪 PPC 的活动。当然，诸如核心活动模块或者引导器的子模块可以执行这样的跟踪。

跟踪活动流处理的计数事件

调度器 212 已经向所选择的 PPC（216）传输了事件，就在与流相关联（从而与 PPC 216 相关联）的位置递增事件计数器。可以在本地存储器块中或者其他方便的位置与本地流 ID 代理相关联地保持事件计数器，所述本地存储器块是为与当前 PPC 处理有关的消息而保留的(例如，在调度器中的核心活动管理器中)。每次事件被发送到 PPC 时，事件计数器递增，每个 PPC 返回“完成”消息时，事件计数器递减。只要事件计数器非零，PPC 当前就在为相关联的流处理事件。当特定的流事件的计数器到达零时，PPC（216）对于该特定流不再具有事件

要处理，并且它的为处理特定流而分配的资源可以被释放以处理其他流。注意，PPC 216 可能正在处理其他流的事件，它的从该特定流的释放可以与这样的其他流无关。

如果与到达调度器 212 的事件的流相关联的事件计数器非零，则它可以优选地将到达的事件分配并传递给同一 PPC。在一些实施例中，如果 PPC 已经处理事件，流状态的全局（即，流状态存储器 214）版本不再有效。相反，只有 PPC 工作空间中的流状态是有效的。在这样的实施例中，当前 PPC 工作空间中的有效流状态应该对随后选择的 PPC 可用，它也应该在当前 PPC 完成对事件的处理之后完成。因此，至少在这样的实施例中，如下方式是方便的，即分配同一 PPC 来处理属于所选择的流的到达事件，直到该 PPC 完成该所选择的流的所有待完成事件。

到达调度器 212 的已经分配到 PPC 的特定流的事件有时可能需要被传递或者分配到不同的 PPC。在这样的情况下，在调度器 212 中保留该事件直到当前 PPC 完成它已经被分配的所有事件的处理是方便的。在调度器 212 中保存事件避免了需要协调同时更新特定流的流状态的两个 PPC。在这样的移交发生之前，在分配新的 PPC 之前允许 PPC 向流存储器登记它的工作空间（反映当前流状态的存储器）也可能是方便的。或者，在当前 PPC 队列的所有事件已经被处理之后，工作空间可以被从当前 PPC 直接传递到新的 PPC。

如果活动的流的事件到达调度器，但是当事件到达调度器 212 时相关的事件计数器为零（表示当前没有 PPC 被分配到该流），则调度器（或者它的引导器子模块）将选择对处理该事件类型可用的 PPC。该选择一般独立于前面的处理，并且可以基于该流。但是，在一些实施例中，可能考虑由特定 PPC 进行的特定流的处理，例如当 PPC 实际上保留了有用的状态信息的时候。一旦做出了 PPC 选择，处理就如上所述继续，并且事件被传输到新的 PPC，流状态被设置在 PPC 的所选择的本地工作空间中。调度器 212 或者将当前流状态传递到新的 PPC，或者指示在流状态存储器 214 的何处发现当前的流状态。

事件计数器只是可以用于确定特定 PPC 当前是否正在处理同一流的前一个事件的一个手段。或者，例如，当前处理流的事件的 PPC 在发现与活动的工作空间相关联的输入队列中没有事件时，可以标记调度器 212。也可用使用任何其他适合的过程来确定当前 PPC 是否被分配来处理特定流。

更新流状态和释放 PPC

在相关联的事件计数器到达零时，可以从处理特定流的事件的责任中释放 PPC。这样的释放意味着 PPC 可以被分配来处理不同的流的事件，这是因为它因此一般将具有空闲的工作空间。一般，PPC 可以同时处理其他流，并且释放不影响 PPC 对其他流的责任。在典型的事件计数器（或者其他指示）表明特定流的事件可以被重新分配到另一个 PPC 处理的情况下，就使 SPPS 能处理通过在不同的 PPC（或者能够处理流的事件类型的那些）之间独立于可以被 PPC 处理的其他流地移动特定各个流来均衡 PPC 处理负载。与使得 PPC 处理一类流（例如流键具有某种特性的一类流）的事件的技术相比，这样的独立流分配可以降低一个或多个 PPC 空闲而另一个 PPC 连续地处理事件的统计概率。

在 PPC 被释放之前，已经被 PPC（216）更新的流存储器被存储在它将对可能在后面的时间中被选择来处理同一流的不同的 PPC 可用的位置。这可以以合适的方式来完成，例如，通过将相关 PPC 工作空间的内容传递到调度器 212，接着传递到流状态存储器 214。或者 PPC（216）可以与调度器 212 配合地将流状态信息传输到流状态存储器 214 的已知的位置，使得调度器意识到流状态已经被更新，并且为未来的访问准备好了。流状态可以更直接地从 PPC（216）传输到流状态存储器 214，例如从总线物理和控制块 228 经由总线 234 传输。总线 234 可以用于从流状态存储器 214 将流状态“检出”到 PPC，或者从 PPC 将更新的流状态“登记”到流状态存储器 214。当事件计数器到达零并且流状态已经被登记到流状态存储器 214 时，当前 PPC 可用被

释放，并且流将返回到反映当前没有 PPC 被分配到它的条件。在 PPC 中，流状态工作空间可以被指示为空闲。

在一些实施例中可以使用在流状态存储器 214 中存储流状态的另一种替换方法。对于具有足够的 PPC 本地的存储器的 SPPS，可以在处理它的最后的 PPC 的工作空间中保持流状态，直到例如其他地方，例如另一个 PPC 需要它的时候。在这样的实施例中，流状态可以经由诸如 224 或者 226 的群内总线被传递到新的 PPC 中的合适的工作空间。这对于处理有限数量的并发流的小 SPPS 更有可能是可行的替换方法。

套接字存储器和输出处理

在保证信息传递的 TLP 应用中，例如 TCP 中，一个要求是确认发送的消息被正确地接收。在这些 TLP 中，如果消息未被正确地接收，则消息将被重传。因为在重传请求到达前可能有一段时间，所以所传输的消息需要被保持在存储器中（例如，在“发送缓冲器”中）一段时间。即使在第一次传输之前也可能需要发送缓冲，例如当输出目标（例如图 2 中的主机 1 104 或者网络 1 106）未准备好接收数据时。类似地，频繁地要求“接收缓冲器”。例如，消息可能被乱序地接收，或者分段地接收，它们必须被保存一段时间以符合 TCP 规则，即要求完成消息并将它们以正确的顺序设置。对于需要大的发送和接收缓冲器的大系统，当消息可以简单地被存储在暂存式存储器 210 中时，建立分开的“套接字存储器”236 以将大量数据存储稍长的时间可能是更方便的。这样的套接字存储器 236 可以经由图 2 所示的总线 238 与暂存式存储器 210 接口，并且经由另一个总线 240 与总线网络和 PPC 集群控制 228 接口。（在一些实施例中，由于流量大，总线 240 实际上可以包括几个单独的总线结构）。

套接字存储器 236 可以提供用于经由总线 246 和 248 输出到输出处理器和 SPI-4 接口 242 和 244 的数据。但是，当要被输出的数据仍然出现在暂存式存储器 210 时，在一些情况下，直接经由总线 250 和

252 将数据提供到输出处理器 242 和 244 更快。输出处理可以包括诸如主要从事件数据准备消息首标，计算校验和和组装完成的输出消息（“重组”）的任务。事件一般保留每种类型的 SP 或者事件类型标识，并且输出处理器可以使用该信息来确定首标的合适的格式、循环冗余校验（CRC）和其他 SP 簿记信息。在输出处理器重组消息之后，输出单元 242 和 244 的 SPI-4 部分根据 SPI-4（或者其他所选择的）接口协议格式化消息，使得数据可以被输出到相同的连接（例如，“主机 1”104 和“网络 1”106），其中，*在 SPPS 的输入从该连接接收数据。*

协议处理器核心功能

一旦 PPC 接收到了合适类型的事件，并且具有反映任何有效载荷的大小和位置的消息，它就可以根据正在使用的 SP 来指引整个消息的处理。PPC 可以指引关于事件所属的流的动作，例如，请求重传，重发以前传输的消息等等，并且可以适当地更新流的流状态。在一些实施例中，如果 PPC 没有物理地直接向输出处理器（242，244）传递消息，而是简单地引导其他电路传递消息以在输出处理器 242 和 244 重组，则流量拥塞可以降低。

一些输出消息包括非常少的信息（例如，只比首标大很少），例如确认或者请求重传。在这些情况下，处理该事件的 PPC（例如，PPC 216）可以基于事件信息形成首标，并将其传送到套接字存储器 236。套接字存储器 236 在将首标信息传送到输出处理器 242 和 244 之前，几乎对其不作处理。其他输出消息包括实际有效载荷，其例如已经随着进入消息被接收并且存储在暂存式存储器 210 中。PPC 可以引导这样的有效载荷从暂存式存储器 210 移动到套接字存储器 236，并且可以单独地引导例如在输出处理器 242 和 244 之一中的一个这样的有效载荷与由 PPC 形成的合适的首标连接。计算机体系结构领域的技术人员应该意识到，PPC 可以以很多方式控制输出消息和流状态信息。

PPC 可以以任何与它们的功能一致的方式来实现。例如，微可编程处理器在处理各种通信需求时提供一定水平的灵活性。一些或者全

部 PPC 可以可替换地实现为硬件形式的固定状态机，这有可能降低电路尺寸和/或增加处理速度。另外，一些或者全部 PPC 可以包括可在程序存储器之外操作的嵌入式微处理器，所示程序存储器可以“在运行中”被修改，即使在 SPPS 是活动的时候。这样的实现允许调整能够处理特定类型的事件的 PPC 的数量，进一步增加的负载均衡的灵活性。PPC 可以被配置成处理一些状态式协议，而不处理其他的，并且该配置可以是固定的或者可变的。例如，在基于微可编程处理器的 PPC 中，微程序（或者软件）一般确定 PPC 被配置成处理何种事件类型或者协议。当 PPC 被配置为处理这样的事件类型或者根据这样的协议处理消息（事件）时，PPC 可与特定事件类型或者协议“兼容”。

总线网络和 PPC 集群控制

图 3 示出了图 2 的总线网络和 PPC 集群控制 228 的示例性体系结构。在该实施例中，PPC 集群（从 216 到 218）经由集群总线接口 302 被部分控制。通过集群总线接口 302，来自指令存储器 304 的指令对群中所有的 PPC（216－218）可用，所示指令存储器 304 一般使用 RAM 实现。集群总线接口 302 也可以提供到用于群中所有的 PPC 的路由控制表 306 的访问。可以提供群 DMA 控制器 308（“C DMA”），其可以具有入口总线，该入口总线将数据从 DMA 控制器 308 的 FIFO 传递到群接口 302 以及群的 PPC 216－218 的每个的双端口存储器（例如，DPMEM 310 和 312）。*DPMEM 310 和 312 可以由从 DMA 控制器到对应的处理器的另一侧访问*，其中 DPMEM 与对应的处理器相关联成为 PPC 216、218 的一部分。如图 3 所示，群 DMA 控制器 308 可以具有单独的入口总线，通过该入口总线，FIFO 从双端口存储器（例如 DPMEM 310 和 312）以及集群总线接口 302 接收数据。群 DMA 控制器 308 可以用来，例如，在 PPC 本地存储器和流状态存储器 214 之间传递流状态。如图 3 所示，群总线控制器 302 还提供到消息总线 314 的双向总线连接以及到套接字存储器 236 的双向总线连接。PPC 的一些或者基本全部本地存储器可以是诸如 DPMEM 310 的

DPMEM，但是根据设计和制造的方便，替代地可以使用任何合适的本地存储器。

图 3 示出了互连套接字存储器 236 和总线网络和 PPC 集群控制 228 的总线 240，其由三个不同的双向总线实现：互连套接字存储器 236 和消息总线 314 的总线 240a；如上所述的总线 240b；到另一个集群总线接口 316 的总线 240c。集群总线接口 316 类似于集群总线接口 302，针对 PPC 220－222 的群操作，其作为纵横交换器来促进 PPC 和消息总线 314、套接字存储器 236 之间的通信，并且提供到公共指令存储器 318 和路由表 320 的访问。另一个群 DMA 322 类似地管理 PPC 220－222 的双端口存储器和集群总线接口 316 之间的数据流。当然，可以提供并类似地互连另一组类似的模块（路由、指令、集群总线接口和群 DMA）。

计算机体系结构领域的技术人员应该意识可以使用任何适合的总线控制来实现所示的总线网络和 PPC 集群控制 228 的连接。例如，可以在各个 PPC 中保持路由和指令信息。另外，PPC 存储器不需要是双端口的，诸如 308 或者 322 的 DMA 控制器也不是必须的。在复杂性稍低的实施例中，集群总线接口 302、316 可以简单地是消息总线 314 的一部分，或者接口可以被完全省略。相反，可以使用更复杂的总线体系结构来增加一些实施例的速度和能力。

利用交替协议核心的流处理

图 4 是示出了可以由示例性 SPPS 执行的动作，该示例性 SPPS 通常交替 PPC（协议处理核心），即在不同的时间使用不同的 PPC 来执行对属于流的消息的状态式协议处理。如图 4 所示，在步骤 402，接收消息。该步骤可以包括各种子步骤，例如从分组段重构完整的消息，执行有效性检验，和/或建立校验和。接下来，在步骤 404，消息的有效载荷可以被移动到暂存式存储器。在其指示分路消息并将消息的一部分存储在特别对输入和输出处理设备可用的临时存储器位置的情况下，步骤 404 是可选的。或者，例如，消息可以保持在一起，并

且/或者它可以被直接移动到更持久的存储器位置。

转到步骤 406，可以定义消息的事件部分。如上面详细讨论的，事件定义一般包括消息的状态相关部分，并且可以需要重格式化消息的首标以及增加诸如校验和和事件类型指示的信息以促进进一步的事件处理。如果消息未被分路，则“事件”可以包括有效载荷信息，甚至可以是基本与所接收的进入消息相同的进入消息。事件处理转到步骤 408，在此，检查包含在事件中的唯一地标识流的数据（“流键”）以开始确定流状态信息和本地流 ID 代理的位置的处理。决定步骤 410 检查 PPC 是否正活动地处理属于同一流的另一个事件。该检查可以通过搜索本地“活动流”表中的流键来实现。如果在“活动流”表中发现流键，则 PPC 当前正在处理属于同一流的另一个事件，并且测量在“是”分支上离开决定步骤 410。如果流是不活动的（例如，如果流的流键未在“活动流”表中被发现），则处理进行到决定步骤 412。在步骤 410 中可以使用其他技术来确定与流键相关联的事件目前是否正在被任何 PPC 处理，例如搜索为当前正在被 PPC 处理的消息流的状态保留的存储器的区域（例如，在调度器的核心活动管理子模块中）。或者，例如，可以检查单个流状态位置以寻找在一个 PPC 上处理正在进行的指示（例如，标志）。用于确定 PPC 是否正在活动地处理流的其他技术和准则将在下面参考决定步骤 428 来描述。

在决定步骤 412，检查在 SPPS 与流键相关联的流是否是活动的。当没有 PPC 当前正在处理该流的事件时，这可以通过在保持活动流的流状态的流存储器中检查有效流位置来执行。（因为活动流的数目可能非常大，所以流存储器一般是单独可访问的，并且远大于当前正在被 PPC 处理的流所使用的本地流表。）该步骤一般包括确定与流键相关的本地流 ID 代理的“查阅”任务，该任务涉及根据散列算法处理流键信息。一旦确定了本地流 ID 代理，则它一般可以用于定位对应于该流键的流的现存的流状态。仅仅有效流状态的存在就可以引起决定步骤 412 的肯定结果。

如果流是完全不活动的，使得在通用的流状态存储器或者在活动

地处理流的 PPC 中都没有活动的流状态，则处理进行到初始化步骤以在流状态存储器中创建并初始化活动的流状态区域。注意，存在一些不要求流状态的无状态“事件”，例如地址解析协议（ARP）事件，其不属于流，并且不需要为其创建流。ARP 以及其他的这样的“无状态”事件可以被图 4 的主要涉及“状态式”事件的处理步骤独立地处理。

一旦建立了活动流（无论是在决定步骤 412 定位的还是在初始化步骤 414 初始化的），方法就进行到分配步骤 416，分配 PPC 来处理事件。该步骤可以包括若干个子步骤，例如确定并识别哪些 PPC 对于处理该事件是兼容的（即，能够处理此类型的事件）并可用的（例如，在它们的队列中具有空间）。可以以许多方式从符合这两个准则的 PPC 选择一个 PPC，所述方式例如是轮循方式，或者通过选择最空闲的 PPC 本地队列，或者是随机方式，或者通过其他负载均衡方式。因为 PPC 刚被分配来处理该事件，所以在步骤 418 使流状态对 PPC 可用。如上所述，流状态可以被调度器（或者子模块）传递；或者如果与所分配的 PPC 共享全局流状态存储器，则该步骤可以包括向 PPC 标识流状态存储器位置。步骤 418 一般还可以包括识别在处理期间 PPC 可以访问流状态的“工作空间”的位置。这样的工作空间一般在 PPC 本地保持，但是在一些实施例中，可以更全局地保持，或者被分路为本地的和全局的。

一旦 PPC 被分配并且具有活动的流状态，（这是发生在步骤 418 之后，或者肯定步骤 410 之后），处理就转到步骤 420 和 422。步骤 420 跟踪处理流的 PPC 的活动。在本发明的一个实施例中，步骤 420 包括递增与将 PPC 分配来处理流的动作相关联的事件计数器，但是在下面将针对决定步骤 428 描述其他实施例。

在步骤 422，事件的内容被提供到所分配的 PPC。这可以通过物理地将事件内容复制到 PPC 的本地存储器的队列来完成，或者，作为另一个例子，通过向 PPC 标识事件数据的位置来完成。这样的队列可以包含来自不同的流的事件，例如来自对流所对应的流状态的可用的工作空间所对应的许多不同的流。如果在兼容 PPC 中（或者对于兼容

PPC），事件队列或者流状态工作空间都不可用，则调度器可以临时停止将部分或者全部事件/工作空间传递到 PPC。

一旦传递完成，分配的 PPC 就可以访问流的流状态，以及事件数据，事件数据一般包括关于与事件相关联的有效载荷的大小和位置的信息。在步骤 424，PPC 可以为与事件相关联的消息执行许多传输层协议处理。协议定义了这样的处理必须达到的最终效果，但是，当然，该效果可以以目前使用的或者以后开发的用于这样的传输层处理的任何方式来完成。PPC 执行的动作包括，例如，更新流状态，创建要输出的消息的首标，引导先前被传输的消息重传，或者发送对接收出现错误的消息的重传的请求。由 PPC 执行的动作还可以包括将它构造的首标的重组引导到接收的有效载荷，并且传输到在流的另一端连接到网络的不同的 TLTS 或者本地主机。完成该事件之后，在步骤 426 断言一个完成声明。在一个实施例中，完成声明被返回到用于跟踪 PPC 活动的全局调度器。

释放活动 PPC

在 PPC 完成处理当前事件之后，在决定步骤 428 确定 PPC 是否已经完成对事件所属的流的所有处理。在一个实施例中，这样的确定可以由调度器模块做出，该调度器模块响应于“完成”声明递减与 PPC 相关联的事件计数器，并且确定事件计数器是否已经到达零。但是，在不同的实施例中，许多用于建立 PPC 已经完成流的替代方案是适用的。例如，当 PPC 完成在其队列中存在的流的最后事件的处理时，可以认为 PPC“完成”流。作为另一个例子，当 PPC 的本地存储器的流状态被另一个 PPC 中的处理重写或者无效，则可以认为 PPC 已经完成流。这些或者其他“完成”的定义可以在一个（或者多个）不同的位置被跟踪，例如在 PPC 自身中，或者在诸如调度器的更全局的模块（例如，在核心活动管理器子模块中）。

如果在决定步骤 428，确定 PPC 正在活动地处理流，在方法可以进行到结束步骤 430，而没有进一步的处理，这是因为 PPC 本地的流

状态已经更新，且全局流状态不需要被更新。但是，在确定 PPC 完成流的处理后，已经在 PPC 被更新的本地流状态在步骤 432 被传递到更全局的流状态位置，使得 PPC 工作空间变得可用于处理不同的流的事件。接着，当属于该流的其他事件到达时，全局流状态可以随即被访问。基于由调度器、子模块或者其他模块，或者 PPC 自身确定的对流的事件处理完成，可以认为 PPC“完成”。“完成”指示也可以推迟到对来自该流的所有事件的处理完成之后，例如直到 PPC 已经没有其他空间用于新的流和事件。一旦在步骤 434 认为 PPC“完成”，则 PPC 可以被释放而不再“分配”来处理该流，这可以，例如，包括设置指示 PPC 中的流状态存储器不再有效或者可用于不同的流状态的另外的存储的标志。在步骤 434 之后，可以认为 PPC 被从该事件所属的流释放。

决定步骤 436 一般发生在某个点以确定由 PPC 处理的最后事件是否允许流完全关闭。甚至可以在决定步骤 428 发生之前，或者在步骤 432 和/或 434 之前执行决定步骤 436，这是因为结束流的决定可以取消将流状态写到存储器的要求。这样的决定也可以包括 PPC“完成”流的决定。但是，为了处理的方便，可以认为结束决定发生在图 4 所示的顺序。作为 PPC 的 SP 处理责任的一部分，PPC 自身一般确定最后事件是否完成了流（例如，流状态是否进行到了“连接关闭”状态）。但是，可以更全局地做出真实地关闭流的决定，例如，在调度器（或者子模块）做出。如果在决定步骤 436 确定不结束流，则系统一般完成处理该消息，并且进行到完成步骤 430。但是，如果在步骤 436 确定结束流，则本地流 ID 代理和流状态存储器位置可以在之后被释放另作他用。因为 PPC 一般在特定流的级别上被分配来处理属于流的事件或者从其释放，而在很大程度上与其他流被分配到何处（至少在兼容 PPC 的访问内）无关，所以有可能，实际上有很大可能，PPC 被分配来处理属于先前被另一个 PPC 处理的流的事件（或者消息）。这样的流－PPC 重分配可能相当频繁，在某些情况下，可能每个流的事件都发生。

调度器处理

图 5 是示出了示例性 SPPS 中的“调度器”模块可以采取的动作，所述“调度器”模块在不同时间将属于流的事件调度到不同 PPC。图 5 集中于实现进入事件的分配的动作，该动作一般是由调度器模块（和子模块）进行的。这样，图 5 的步骤基本上是诸如图 4 中所示的整个 SPPS 的步骤的子集，尽管图 5 的步骤是从调度器模块的角度，并且也可以包括与图 4 不同的细节。调度器模块在概念上与它向其分配事件的 PPC 分离，并且与它从其结束事件的输入处理分离，并且它可以类似于图 2 中的调度器 212 连接在 SPPS 内，或者以其他方式被连接。调度器模块还可以在概念上甚至物理上被再划分；例如，参考本地流 ID 代理（和/或流状态）“查阅”模块，以及引导器核心活动管理器，它们的每个都可以在概念上或者物理上是调度器模块的子模块，或者是与调度器相关联的附属模块。

如图 5 所示，在步骤 502，从输入源接收事件。事件一般只包含被 SPPS 处理的消息的“状态相关”部分。即，事件一般只包含 PPC（协议处理核心）控制与消息相关联的流的流状态的保持所需的消息，而不包含消息的有效载荷。但是，在一些实施例中，有效载荷或者其部分可以与状态相关数据一起保存。调度器检查包含在事件中唯一地标识事件所属的流的“流键”。在步骤 504，调度器在核心活动管理器（或者“CAM”）中搜索以匹配流键，其指示了 PPC 活动地处理与该流相关的事件。如果在 CAM（其可以物理地或者概念地与调度器分离）中没有发现匹配，则在该示例性实施例中假定没有 PPC 正在活动地处理该流的事件，并且在步骤 506，初始化 CAM 条目以跟踪被分配来处理该事件的 PPC 的活动。

在步骤 508，调度器搜索对应于流键的本地流 ID 代理。对于处理大量流的 SPPS，该搜索可能由不同的查阅模块执行，该模块可以，例如，执行散列查阅以尽可能快地定位本地流 ID 代理。决定步骤 510 取决于是否发现匹配流键的本地流 ID 代理。如果没有，则 SPPS 可能还没有处理来自该流的任何数据，因此在步骤 512，可以选择流状态

ID 以与流相关联，该流由事件的流键唯一地标识。此后，（或者如果发现了本地流 ID 代理，并且在步骤 510 的决定为“是”），处理可以转到步骤 514。

在步骤 514，选择 PPC 来处理该事件。该步骤可以包括确定被处理的事件的类型的子步骤，尽管在一些实施例中，该子步骤是由更早的处理模块（例如，诸如图 2 的 206 或者 208 的消息分路器）执行的。在调度器中保持的对于每个 PPC 的“事件类型掩码”可以与表示事件类型的比特相比较以确定哪些 PPC 与该事件类型兼容。另一个子步骤可以包括检查被配置成处理该事件类型的那些 PPC 的相对活动水平。可以选择最不忙的 PPC，或者可以以轮循的方式选择在其输入队列中具有空间的下一个 PPC。作为另一个例子，可以（例如，在核心活动管理器子模块中）保持包括本地工作空间的关于最近 PPC 活动的分配的数据，并且可以选择还没有为该事件的流重写它的流状态存储器的 PPC，尽管否则就认为它“完成”了该流。引导器子模块，或者与核心活动管理器（CAM）子模块结合或者作为其一部分，可以执行这些动作。一般对每个进入事件的流具体地选择 PPC，而不关心流可能所属（例如根据其流键的特性）的先验类（priori class）。作为这样的个别分配技术的结果，被选择来处理流的特定事件的 PPC 经常不同于处理相同流的先前事件的 PPC（除非如在其他地方解释的，特定流当前在 PPC 是活动的）。

因为在步骤 512 中，或者在步骤 508－510 中初始化流状态，并且在步骤 514 选择 PPC，所以在步骤 516，流状态可以被传递到 PPC。在一些实施例中，这样的传递可以是“虚拟的”，仅提供流状态存在与存储器的何处的指示以使 PPC 可以访问它。接下来，处理可以转到步骤 518。可以直接从决定步骤 504 到达该步骤，这是因为如果决定步骤 504 的决定为“是”，则 PPC 已经处理属于同一流的较早的事件。这样的活动 PPC（在一些实施例中）已经具有该流的最活动的流状态，并且在此情况下一般将被选择来处理当前事件。从而，在步骤 518，事件自身可以被转发到所选择的 PPC 的队列或者输入区域。随着步骤

518 的进行，在步骤 520，事件计数器可以递增。事件计数器是确定 PPC 何时正在活动地处理当前事件的流的另一个事件的一种方法，但是可以使用其他方法，例如等待 PPC 指示它完成了特定流的所有当前事件的处理。这是调度器的接收处理的结尾。

图 6 是图示了调度器（或者它的子模块）可以响应于来自 PPC 的反馈而执行的一些步骤。如图 5 那样，图 6 的步骤很大程度上是整个 SPPS 的步骤的子集，尽管它们是从调度器模块的角度，并且与图 4 所示整个 SPPS 的步骤相比，可以包括更多的或者不同的步骤。

所示的图 6 的响应动作在步骤 602 开始，在步骤 602 期间，调度器接收“完成声明”或者其他指示 PPC 已经完成特定流的处理的指示。调度器接着可以递减该流的事件计数器（参考图 5 的步骤 520 所讨论的）。如果，在决定步骤 606，发现事件计数器已经到达零，或者如果 PPC 所完成的事件的“突发”有其他指示，则调度器可以使由 PPC 更新的流状态存储在更持久的存储器中以释放 PPC 的存储器。（注意，对于这样的实施例该步骤是不需要的：在处理期间 PPC 所使用的存储器与当没有 PPC 正在处理时的存储器时相同，此情况可能发生在，例如，流状态总是保持在相同的全局位置，并且仅被处理流的 PPC 按照需要访问时）。接着，标志或者其他指示可以被包括在 CAM 中，或者被发送到 PPC，以指示存储在 PPC 中的流状态不再有效。接着，在步骤 612，PPC 可以被释放而不再处理它曾处理的特定流。因为现在没有 PPC 正在处理特定流的事件，所以在步骤 614，在其中保持了 PPC 活动的 CAM 块也可以被释放。

应当注意的是，“释放”可能等效于仅仅设置表示 PPC（或者 CAM 存储器块）可用的标志。这样的标志可以指示可用性，但是，只要没有重要数据被重写，在这样的指示之后，PPC 实际上可以被当作它仍然活动地处理事件来对待。在此情况下，决定步骤 606 将返回“否”，直到数据块被真实地重写，从而被破坏。无论如何，如果决定步骤 606 返回“否”，则处理完成，这是因为在该事件中一般不需要步骤 608－614。否则，在步骤 614，CAM 块被释放之后，处理完成。

已封装状态式流处理

例如这里所描述的 SPPS（状态式协议处理系统）可以处理除了传输层以外的层的一种方式是通过抽取已封装消息并将所抽取的消息再循环（recirculate）以进行进一步的处理。这样的进一步的处理可以根据用于已封装消息的合适的协议来执行，其一般不同于用于封装消息的协议（典型地为 TLP）

在获取已封装状态式消息之后，将其重格式化并作为输入（非传输层输入）提供给 SPPS，只要一个或多个 PPC 配置有特定协议（例如，iSCSI）所需的步骤，SPPS 就可以根据该协议处理该消息。这样，简单地使用 SPPS 用于非传输层处理是容易的。

有许多方法可以通知 SPPS：已封装事件请求再循环。例如，如果所有的被处理的事件要求再循环，通知可以是隐含的。或者，封装消息的有效载荷或者首标的一个或多个部分可以包含指示需要这样的再循环的信息。SPPS 可以检查每个有效载荷以查阅再循环已封装信息的指示，或者当在首标提供指示时，它可以只检查有效载荷。这样，SPPS 可以接收指示，该指示是关于有效载荷是否要被检查，它是否请求进一步的处理，以及应该根据什么协议来执行这样的进一步的处理，该指示是封装消息的有效载荷和/或首标中的隐含或者明显信息的任何结合。

可以首先调用“再循环”协议，以使得封装信息的有效载荷（和/或首标的部分）被分段并重组为已封装流的消息。注意，单个封装消息可以包含多个已封装消息的所有或者部分，相反单个已封装消息可以请求传输多个封装消息（例如，当大消息被封装在多个诸如 ATM 分组的小分组中时）。再循环定义合适的已封装消息的重组，还引导它返回 SPPS 的输入端进行进一步处理。这样的再循环协议可以例如通过定义本地流 ID 代理、事件类型和其他已知的有用的信息，将再循环的消息格式化为特别有效率的格式。以此方式，SPPS 再循环协议处理器的作用类似于与 SPPS 紧密配合而操作的主机。这样的具有

对于 SPPS 的理想消息格式的知识的主机可以通过将消息格式化为这样的理想格式而加速处理。

还应该注意，再循环可以通过修改的通信路径来实现，这样，重组或者“输出处理器”242 和/或 244 将重组的已封装消息直接传递回消息分路器 206 或者 208，而不是使其通过再循环可能不需要的 242、244、202 和 204 中的 SPI-4 接口。实际上，再循环的消息可以以由消息分路器 206 或者 208 实现的方式被整个预格式化。所选择的处理封装消息的 PPC（或者相关的处理器）可以执行这样的预格式化，并且引导信息从 242/244 中的重组处理器直接传递到暂存式存储器 210 和调度器 212，从而完全绕过了消息分路器。

一旦已经实现再循环，就可以如上所述，即以与处理 TLP 消息基本相同的方式对已封装信息的进一步处理。在所关心的情况，已封装信息是状态式的，并且属于流，所以可以创建反映消息的状态相关部分的事件，将确定流键的本地代理，将创建或者定位流的状态，并且与协议兼容的 PPC（协议处理核心）将被分配来处理从（先前被封装，现在被再循环的）消息得出的事件。这些步骤不仅可以对再循环的消息执行，还可以对向 SPPS 提供的传输层或者非传输层的任何流。

当然，处理非传输层消息要求消息被发送到另一个子系统。例如，已封装消息内的数据可以要求传递到主机。所分配的 PPC 可以通过引导信息以目标主机可接收的方式被重组，接着引导重组的消息传递到目标主机，来实现这样的发送。在替代方案中，向网络连接发送已封装消息可以要求输出消息被再封装为 TLP 消息（典型但非必须的，用于原始封装消息的同一 TLP，例如 TCP）。这样，在这一点上可以要求另一个再循环以再封装这样的消息。在理论上，至少，消息可以被“套”在一系列的多个封装中，所述封装在可以处理最内部的状态式消息之前必须被剥离。类似地，处理这样的最内部的状态式消息可以要求对消息对称再封装。在实践中，考虑到效率，要避免过多的封装。

虽然已经描述了本发明的多个实施例。然而，应该理解，可以做出各种修改而不偏离本发明的范围。例如，本发明的方法可以以软件

或者硬件，或者软件和硬件相结合的实施例来执行。作为另一个例子，应该理解，所描述的作为一个模块的功能一般可以在另一个模块中等效地执行。作为另一个例子，以特定顺序示出或者描述的步骤一般可以以不同的顺序执行，除非在权利要求中描述的那些实施例包括特定顺序的步骤。

因此，应该理解，本发明不限于具体描述的实施例，而只由权利要求的范围来限定。说明书可以提供类似于权利要求所陈述的特征的例子，但是不应假定，这样的类似的特征等于权利要求中的特征，除非这样的相同性对于理解权利要求的范围是重要的。在一些例子中，通过略有不同的术语来强调权利要求的特征和说明书的特征之间的不同。

图1A

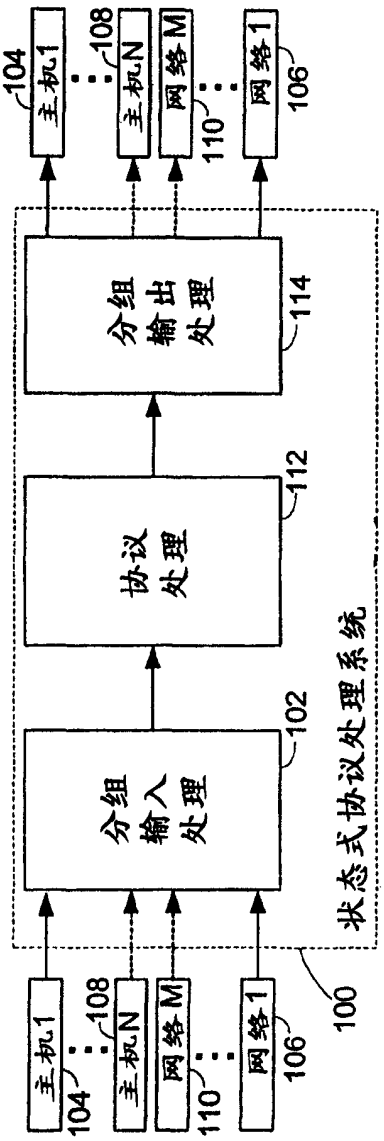
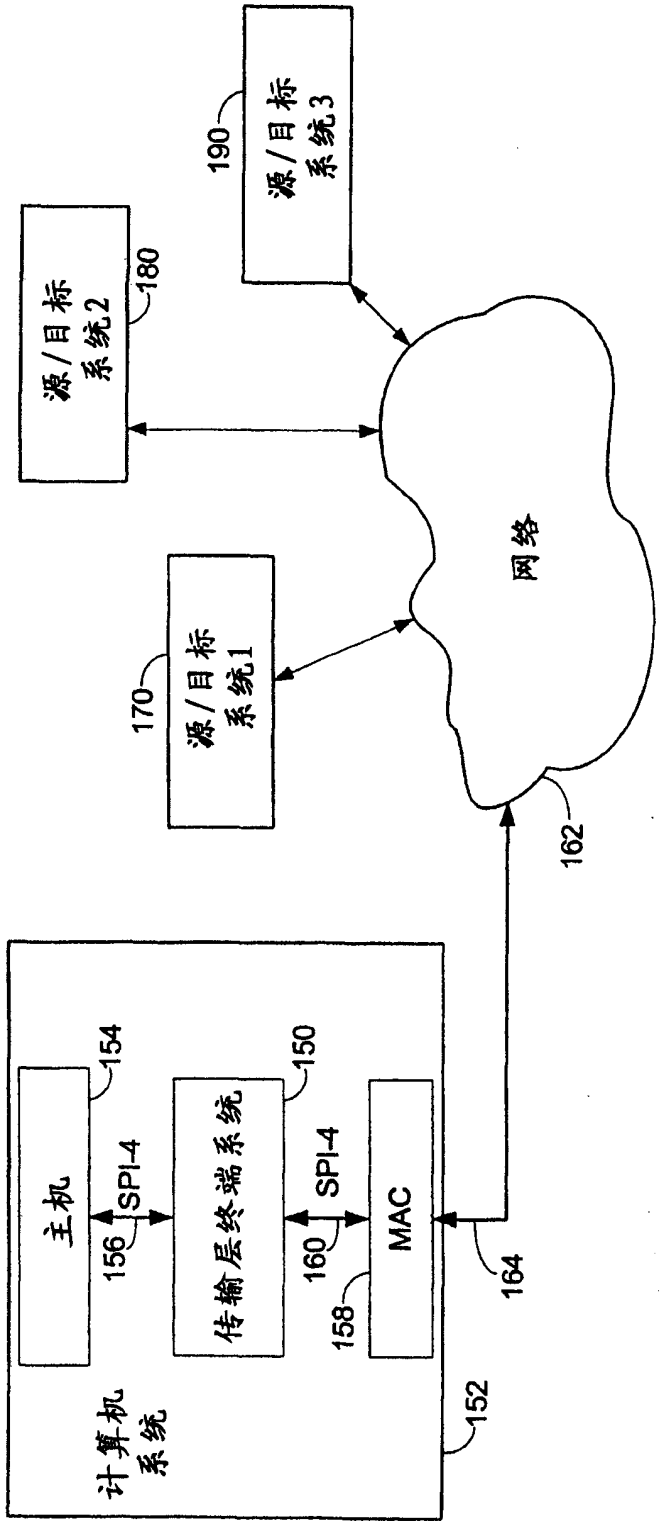

主机1
104
主机N
108
网络M
110
网络1
106
分组输入处理
102
协议处理
112
分组输出处理
114
状态式协议处理系统
100

图1B


源/目标系统2
180
源/目标系统3
190
网络
源/目标系统1
170
162
主机
154
SPI-4
156
传输层终端系统
150
SPI-4
160
MAC
158
164
计算机系统
152

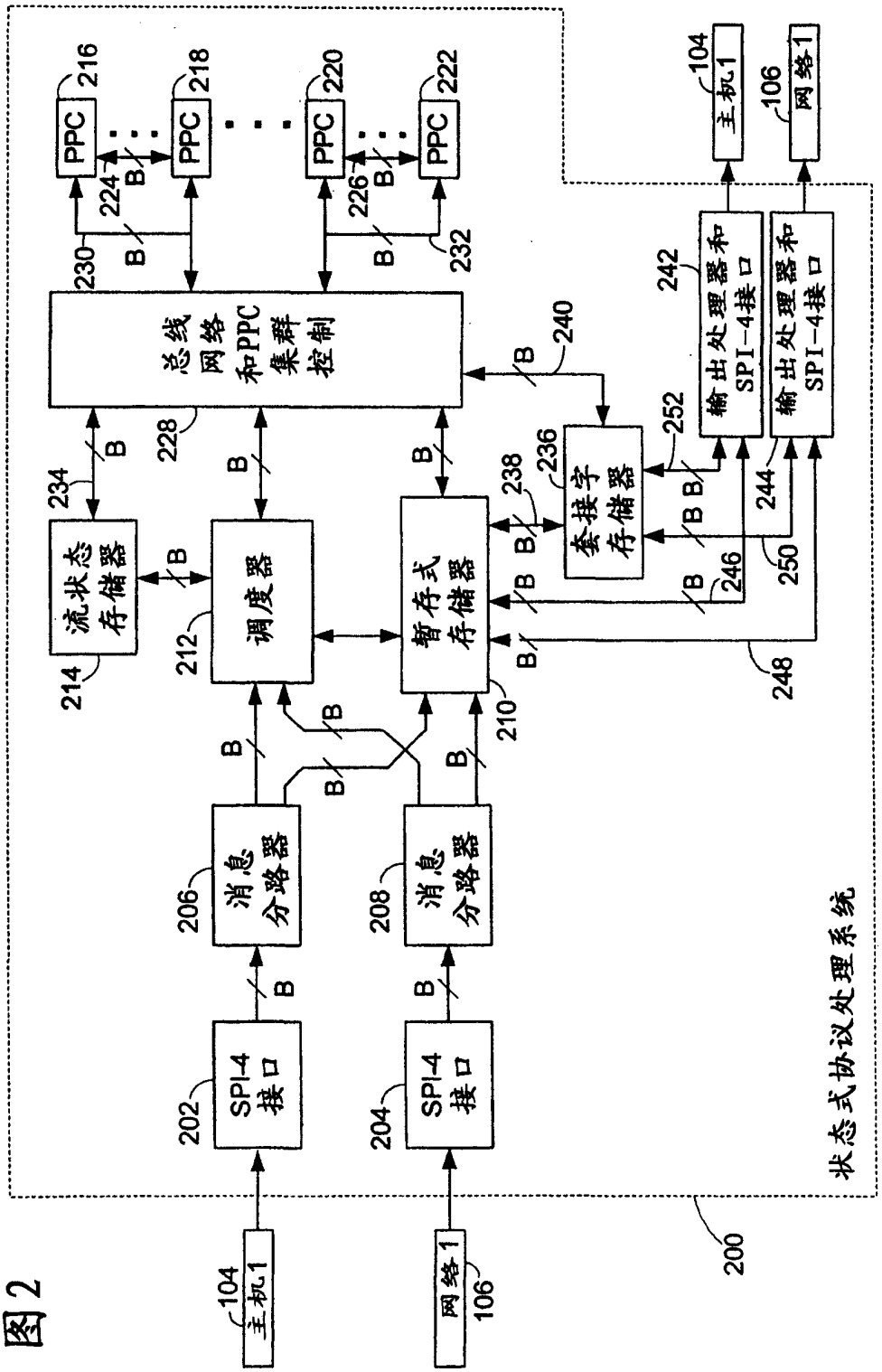

状态式协议处理系统
200
104
主机1
106
网络1
SPI-4
接口
202
204
消息
分路器
206
208
调度器
212
流状态
存储器
214
暂存式
存储器
210
套接字
存储器
236
238
252
输出处理器和
SPI-4接口
242
244
246
250
248
总线
网络和PPC
集群控制
228
240
PPC
216
218
220
222
224
226
230
232
234
B

图2

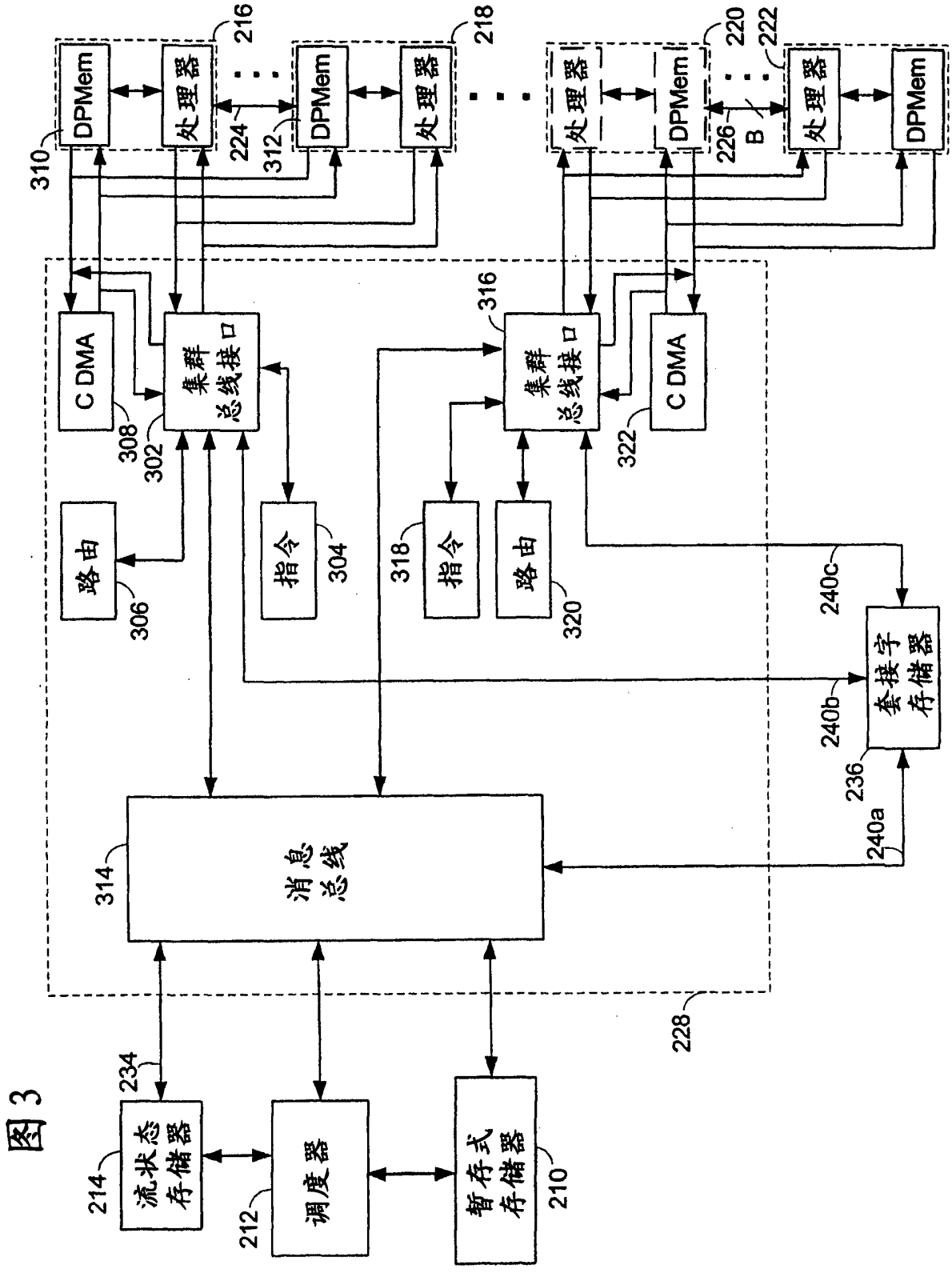

DPMem
处理器
C DMA
集群总线接口
路由
指令
消息总线
流状态存储器
调度器
暂存式存储器
套接字存储器
310
216
218
220
222
224
312
226
B
308
302
316
322
306
304
318
320
240c
240b
236
240a
314
234
214
212
210
228

图 3

图4

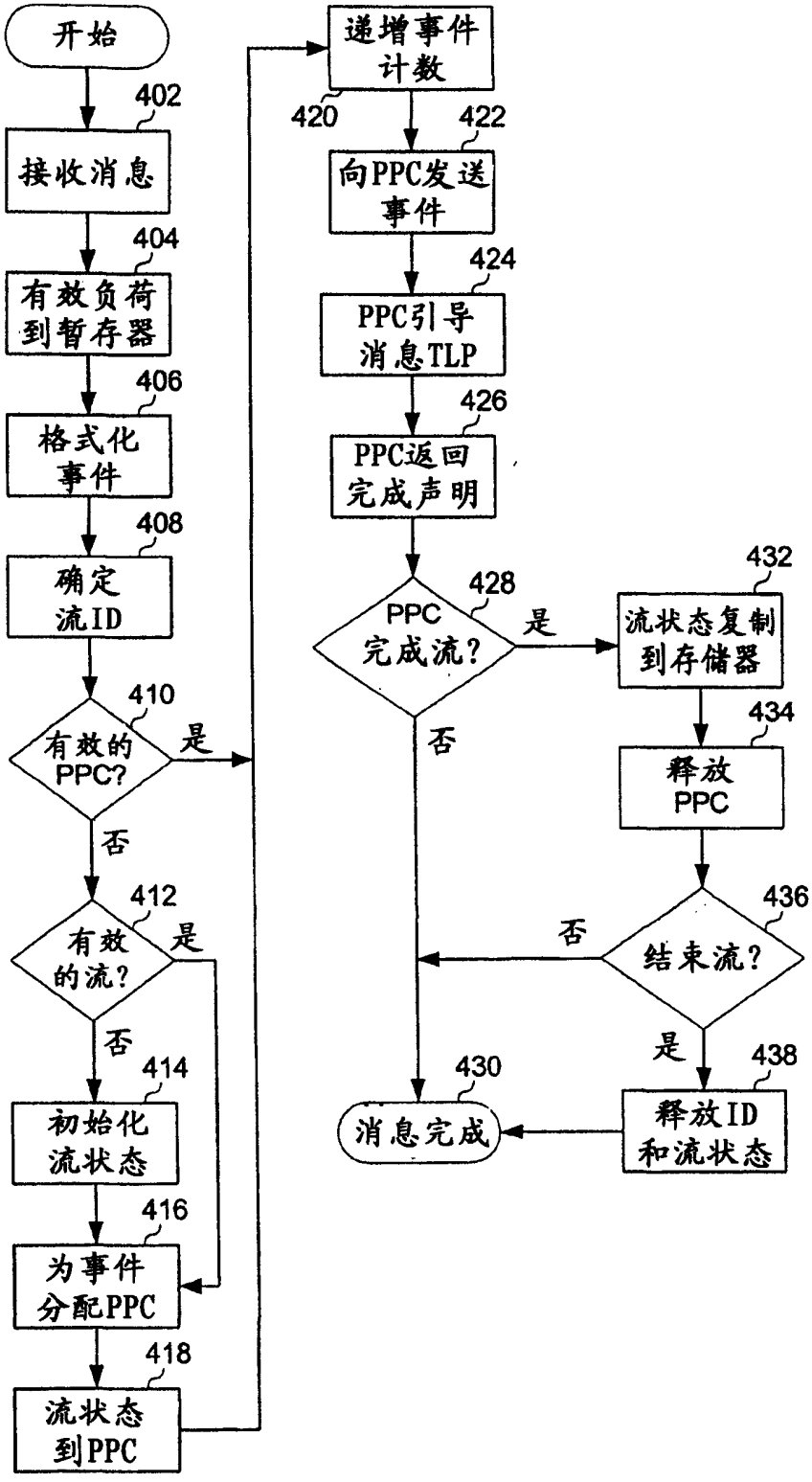

开始
402
接收消息
404
有效负荷到暂存器
406
格式化事件
408
确定流ID
410
有效的PPC?
是
否
412
有效的流?
是
否
414
初始化流状态
416
为事件分配PPC
418
流状态到PPC
递增事件计数
420
422
向PPC发送事件
424
PPC引导消息TLP
426
PPC返回完成声明
428
PPC完成流?
是
否
432
流状态复制到存储器
434
释放PPC
436
结束流?
否
是
438
释放ID和流状态
430
消息完成

图5

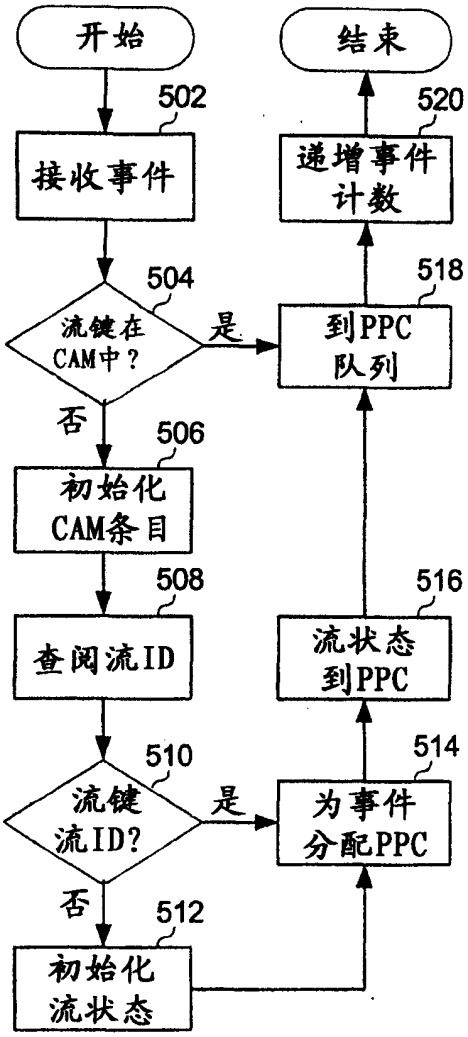
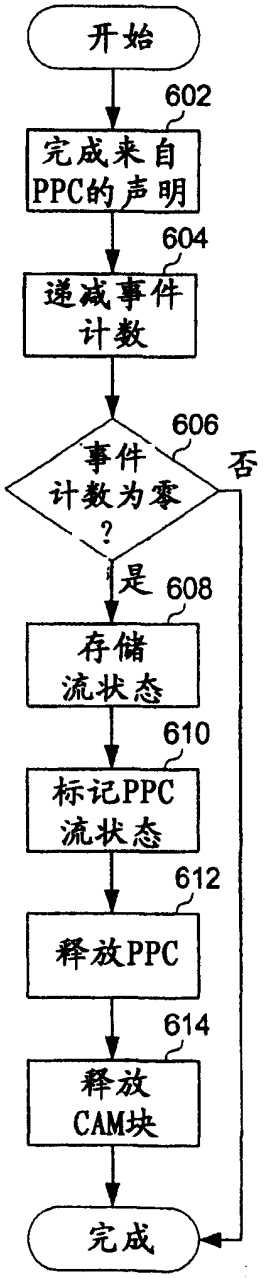

开始
结束
502
接收事件
520
递增事件计数
504
流键在CAM中？
是
518
到PPC队列
否
506
初始化CAM条目
508
查阅流ID
516
流状态到PPC
510
流键流ID?
是
514
为事件分配PPC
否
512
初始化流状态

图6


开始
602
完成来自PPC的声明
604
递减事件计数
606
事件计数为零？
否
是
608
存储流状态
610
标记PPC流状态
612
释放PPC
614
释放CAM块
完成