

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 15/16 (2006.01)



[12] 发明专利申请公布说明书

[21] 申请号 200680050079.2

[43] 公开日 2009 年 1 月 21 日

[11] 公开号 CN 101351784A

[22] 申请日 2006.12.26

[21] 申请号 200680050079.2

[30] 优先权

[32] 2005.12.30 [33] US [31] 11/323,165

[86] 国际申请 PCT/US2006/049317 2006.12.26

[87] 国际公布 WO2007/079095 英 2007.7.12

[85] 进入国家阶段日期 2008.6.30

[71] 申请人 阿西式·A·潘迪亚

地址 美国加利福尼亚州

[72] 发明人 阿西式·A·潘迪亚

[74] 专利代理机构 北京市中咨律师事务所
代理人 杨晓光 于 静

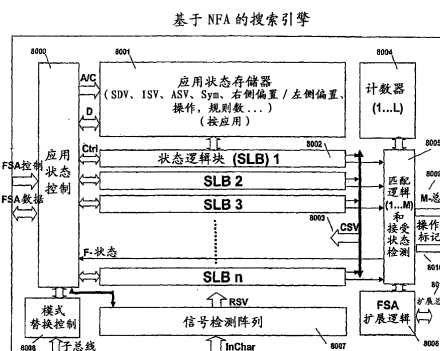
权利要求书 7 页 说明书 141 页 附图 98 页

[54] 发明名称

运行时自适应搜索处理器

[57] 摘要

披露了一种运行时自适应搜索处理器。所述搜索处理器提供了高速内容搜索能力以满足增长到 1Gbps、10Gbps 及更高的线路速度的性能需求。所述搜索处理器提供了独特的基于 NFA 和 DFA 组合的搜索引擎，所述引擎可以并行处理数据以根据搜索引擎中编程的特定规则执行搜索。处理器体系结构还提供了通过传输协议层传输和处理来自第 2 层的网际协议 (IP) 分组的能力，并且还提供了通过第 7 层执行分组检查。还披露了一种安全性系统，该系统实现了一种使用具有适当安全特性的协议处理硬件以分布方式实施企业网络内部安全能力的新方法。



1. 一种处理信息内容并具有多个硬件配置的运行时自适应搜索处理器，所述处理器包括：

控制处理器和调度器，用于控制和调度所述信息内容以便处理；

自适应控制器，用于在运行时配置所述运行时自适应搜索处理器的多个配置；

至少一个配置存储器，用于存储至少所述多个硬件配置；

至少一个运行时自适应 NFA 搜索引擎，用于使用至少一个 NFA 对所述信息内容执行搜索；

至少一个运行时自适应 DFA 搜索引擎，用于使用至少一个 DFA 对所述信息内容执行搜索；以及

以上各项中的任何项的组合。

2. 根据权利要求 1 的运行时自适应搜索处理器，还包括以下项中的至少一项：

至少一个搜索群集，其包括至少一个用于搜索信息的搜索引擎；

全局配置控制器，用于进行全局配置控制；

配置存储器控制器，用于控制对至少所述配置存储器的访问；

群集间路由交换机，用于控制多个所述至少一个搜索群集之间的路由；

全局存储器，用于提供存储器存储；

全局存储器控制器，用于控制对至少所述全局存储器的访问；以及

以上各项中的任何项的组合。

3. 根据权利要求 2 的运行时自适应搜索处理器，其中所述至少一个搜索群集包括：

群集配置控制器，用于配置所述至少一个搜索群集；

至少一个运行时自适应搜索引擎，用于搜索信息；以及

可配置群集互连网络，用于连接多个所述至少一个运行时自适应搜索引擎。

4. 根据权利要求2的运行时自适应搜索处理器,其中所述至少一个搜索群集进一步包括以下项中的至少一项:

群集存储器,用于提供存储器存储;

群集存储器控制器,用于控制至少所述群集存储器;

群集路由交换机,用于将所述至少一个运行时自适应搜索引擎与所述至少一个搜索群集或所述群集间路由交换机互连;

群集流控制器,用于至少存储和检索流上下文;以及

以上各项中的任何项的组合。

5. 根据权利要求3的运行时自适应搜索处理器,其中所述至少一个运行时自适应搜索引擎包括以下项中的至少一项:

运行时自适应 NFA 搜索引擎,用于使用至少一个 NFA 执行搜索;

运行时自适应 DFA 搜索引擎,用于使用至少一个 DFA 执行搜索;或
以上各项的组合。

6. 根据权利要求1的运行时自适应搜索处理器,其中所述至少一个运行时自适应 NFA 搜索引擎计算至少一个所述 NFA 并且包括:

至少一个状态逻辑块,用于执行状态的状态逻辑操作;

至少一个应用状态存储器,用于存储应用状态信息;

至少一个符号检测逻辑,用于检测符号;

至少一个应用状态控制器,用于控制所述至少一个运行时自适应 NFA 搜索引擎的操作;以及

以上各项中的任何项的组合。

7. 根据权利要求1的运行时自适应搜索处理器,其中所述至少一个运行时自适应 NFA 搜索引擎进一步包括以下项中的至少一项:

模式替换控制,用于执行模式替换;

计数器,用于为各种事件提供计数;

匹配逻辑,用于检测规则匹配;

FSA 扩展逻辑,用于使得所述运行时自适应 NFA 搜索引擎能够支持更大的 FSA;以及

以上各项中的任何项的组合。

8. 根据权利要求 1 的运行时自适应搜索处理器, 其中所述至少一个运行时自适应 DFA 搜索引擎包括:

至少一个 DFA 操作引擎, 用于执行状态逻辑操作;

至少一个 DFA 指令和状态流存储器, 用于存储 DFA 操作指令和指令的状态流;

至少一个 DFA 状态控制器, 用于控制所述至少一个运行时自适应 DFA 搜索引擎的操作; 以及

以上各项中的任何项的组合。

9. 根据权利要求 1 的运行时自适应搜索处理器, 其中所述至少一个运行时自适应 DFA 搜索引擎进一步包括以下项中的至少一项:

模式替换控制, 用于执行模式替换;

计数器, 用于为各种事件提供计数;

DFA 上下文控制器, 用于控制所述 DFA 搜索引擎的应用上下文;

DFA 扩展逻辑, 用于使得所述运行时自适应 DFA 搜索引擎能够支持更大的 FSA; 以及

以上各项中的任何项的组合。

10. 根据权利要求 6 的运行时自适应搜索处理器, 其中所述至少一个状态逻辑块包括:

状态相关向量 (SDV), 用于启用或禁用所述状态与所述 NFA 的一个或多个状态之间的互连;

当前状态向量 (CSV), 其代表所述 NFA 的所述一个或多个状态中的部分状态的当前状态, 所述状态依赖该当前状态得到其下一状态;

所接收符号向量 (RVS), 其代表作为到所述符号检测逻辑的输入接收的所述符号的值;

左侧偏置或右侧偏置信号, 其指示 NFA 类型;

状态转移逻辑, 其使用至少所述 SDV、所述 CSV、所述 RSV 以及所述左侧偏置或右侧偏置信号来生成所述状态的下一状态值;

init 逻辑块, 用于控制所述状态的下一状态的值;

状态存储器, 用于存储所述状态的值;

接受检测逻辑, 用于检测和发信号通知所述状态何时为接受状态并且何时被激活; 以及

以上各项中的任何项的组合。

11. 根据权利要求 6 的运行时自适应搜索处理器, 其中所述至少一个状态逻辑块进一步包括以下项中的至少一项:

用于使所述 SDV 启用所述 SDV 的运行时配置的可编程存储装置;

用于使所述左侧偏置 (LB) 或右侧偏置 (RB) 信号启用实现的 NFA 类型的运行时配置的可编程存储装置;

用于存储起始标志以指示所述状态是否为所述 NFA 的起始状态的可编程存储装置;

用于存储接受标志以指示所述状态是否为所述 NFA 的接受状态的可编程存储装置;

用于存储操作字段以指示当在所述 NFA 计算期间激活或达到所述状态时应采取的操作的可编程存储装置;

用于存储标记字段以指示当在所述 NFA 计算期间激活或达到所述状态时发出的标记的可编程存储装置; 以及

以上各项中的任何项的组合。

12. 一种具有多个状态的运行时自适应有限状态自动机体系结构, 所述体系结构包括:

在所述多个状态中的任意两个状态之间的至少一个互连;

所述多个状态中的每个状态的状态相关向量 (SDV), 其用于启用或禁用所述两个状态之间和所述多个状态中的其他状态之间的所述至少一个互连;

所述多个状态中的每个状态的当前状态向量 (CSV), 其代表所述多个状态中的部分状态的当前状态, 所述多个状态中的状态依赖该当前状态得到其下一状态值;

至少一个与所述多个状态中的每个状态关联的符号，所述符号用于控制转换为所述多个状态中的所述每个状态或从所述多个状态中的所述每个状态进行的转换；

至少一个符号检测逻辑，用于接收符号、检测所述符号的值并生成所接收符号向量（RSV）；

至少一个应用状态存储器，用于保存使用所述有限状态自动机的应用上下文；

所述多个状态中的每个状态的状态转移逻辑，其使用至少所述状态相关向量、所述当前状态向量以及所述所接收符号向量来生成所述多个状态中的每个状态的下一状态；以及

以上各项中的任何项的组合。

13. 根据权利要求 12 的运行时自适应有限状态自动机（FSA），还包括以下项中的至少一项：

用于使 SDV 启用 SDV 的运行时配置的可编程存储装置；

用于使左侧偏置（LB）或右侧偏置（RB）信号启用实现的 NFA 类型的运行时配置的可编程存储装置；

用于存储起始标志以指示所述状态是否为所述 NFA 的起始状态的可编程存储装置；

用于存储接受标志以指示所述状态是否为所述 NFA 的接受状态的可编程存储装置；

用于存储操作字段以指示当在所述 NFA 计算期间激活或达到所述状态时应采取的操作的可编程存储装置；

用于存储标记字段以指示当在所述 NFA 计算期间激活或达到所述状态时发出的标记的可编程存储装置；以及

以上各项中的任何项的组合。

14. 一种用于处理网络分组或本地内容的系统，所述系统能够与网络相连，所述系统包括运行时自适应搜索处理器以便处理嵌入所述网络分组或所述本地内容中的内容。

15. 根据权利要求 14 的系统, 其中所述运行时自适应搜索处理器进一步提供了运行时自适应搜索处理器阵列硬件, 以动态地选择所述运行时自适应搜索处理器中的硬件配置以便支持所述网络分组或所述本地内容的应用或服务处理需求。

16. 根据权利要求 14 的系统, 包括内容搜索 API, 其使得应用能够利用所述运行时自适应搜索处理器阵列硬件来加速内容搜索; 其中所述内容搜索 API 包括: 用于初始化所述运行时自适应搜索处理器的装置; 用于将内容搜索规则传送给所述运行时自适应搜索处理器的装置; 用于传送要由所述运行时自适应搜索处理器搜索的内容的装置; 用于将所述运行时自适应搜索处理器进行的所述内容搜索的结果传送给所述应用的装置; 用于启动和停止所述运行时自适应搜索处理器以启动和停止内容搜索的装置; 以及以上各项中的任何项的组合。

17. 根据权利要求 14 的系统, 其中所述运行时自适应搜索处理器启用电子邮件安全应用或 Web 安全应用, 所述电子邮件安全应用包括: 反垃圾邮件; 反病毒; 反间谍软件; 反网络钓鱼; 防止机密泄漏或合规性; 或以上各项的组合, 所述 Web 安全应用包括: 防火墙; 入侵检测和防范; 应用层安全; 反垃圾邮件; 反病毒; 反间谍软件; 反网络钓鱼; 防止机密泄漏或合规性; 或以上各项的组合; 其中可以以网络线路速度执行所述应用; 其中所述网络线路速度在 100 Mbps 以下到 10 Gbps 以及更高的线路速度范围之内。

18. 根据权利要求 6 的运行时自适应搜索处理器, 其中所述至少一个应用状态存储器包括至少一个应用状态上下文, 当选择所述至少一个应用状态上下文以便在所述 NFA 搜索引擎中进行配置时, 该上下文代表要在所述 NFA 搜索引擎中编程的上下文信息; 其中所述应用状态上下文包括可应用于所述 NFA 的所有状态的 NFA 参数, 并且至少一组状态参数可应用于所述 NFA 的所述至少一个状态中的每个状态或每个状态的组合。

19. 根据权利要求 14 的系统, 其中所述运行时自适应搜索处理器从网络接收数据分组并向网络发送数据分组, 并且进一步提供了根据从所述

网络接收的数据分组、发送到所述网络的数据分组或它们的组合来选择下列各项的能力：

- 基于策略的服务、应用或策略或它们的组合；
- 用户选择的服务、应用或策略或它们的组合；
- 用户定义的服务、应用或策略或它们的组合；
- 用户配置的服务、应用或策略或它们的组合；以及
- 以上各项中的任何项的组合。

20. 一种根据权利要求 1 并嵌入处理网络分组或本地内容的系统的运行时自适应搜索处理器，所述系统能够与网络相连，其中所述运行时自适应搜索处理器处理嵌入所述网络分组或所述本地内容中的内容，所述运行时自适应搜索处理器还包括运行时自适应搜索处理器阵列硬件，以动态地选择所述运行时自适应搜索处理器中的硬件配置以便支持所述网络分组或所述本地内容的应用或服务处理需求。

运行时自适应搜索处理器

相关申请

本申请要求在 2005 年 12 月 30 日提交的标题为 “Runtime Adaptable Search Processor” 的美国专利申请 No. 11/323,165 的优先权。

背景技术

本发明一般地涉及内容搜索、存储和联网半导体，具体地说，涉及可以在联网、存储、安全性、生物信息、芯片组、服务器、搜索引擎等中使用的高性能内容搜索、网络存储和安全性处理器。

许多现代应用都依赖于快速信息搜索和检索。随着万维网的出现及其使用的显著增多，内容搜索已成为关键能力。由于现有水平的微处理器在正则表达式驱动搜索方面的性能限制，在 Web 搜索应用中部署了大量服务器。

针对在字符串中搜索词汇信息或模式这一主题，已投入了大量的研究和开发资源。从 20 世纪 50 年代中期起，已经广泛使用正则表达式来描述字符串中用于内容搜索、词汇分析、信息检索系统等模式。最初由 S.C. Kleene 在 20 世纪 50 年代中期对正则表达式进行研究以描述神经活动现象。业界公知的是，正则表达式 (RE) 也可以使用有限状态自动机 (FSA) 来表示。非确定性有限状态自动机 (NFA) 和确定性有限状态自动机 (DFA) 是两种已在计算历史中广泛使用的 FSA。Rabin 和 Scott 在 1959 年首先证明了 DFA 和 NFA 在识别语言能力方面的等价性。通常，研究的主要部分集中于正则表达式。在 Hopcroft 和 Ullman 所著的 “Introduction to Automata Theory, Languages and Computation” 中可以找到正则表达式理论，并且在 Aho、Sethi 和 Ullman 所著的 “Compilers: Principles, Techniques

and Tools”一书中也可以找到该主题的重要讨论。

网际协议(IP)是在如局域网(LAN)、城域网(MAN)和广域网(WAN)等各种网络之间部署的最普遍的联网协议。存储区域网络(SAN)主要基于光纤通道(FC)技术。存在创建基于IP的存储网络的需求。

当根据旨在传输数据流的IP来传输块存储业务时,使用被分层以在IP顶部运行的传输控制协议(TCP)来传输数据流。TCP/IP是在操作系统内的软件中实现的可靠连接/面向会话的协议。TCP/IP软件栈处理将在未来部署的较高线路速度时会非常缓慢。目前,运行TCP/IP栈并且具有1Gbps网络连接的基于1GHz处理器的服务器将使用50-70%或更多的处理器周期,剩余最低限度的周期以供处理器分配给在服务器上运行的应用。当在TCP/IP上传输存储数据时以及对于高性能IP网络,这种开销是不可容忍的。因此,新的硬件解决方案将加速TCP/IP栈以承载存储和网络数据业务并与基于FC的解决方案相媲美。除了TCP协议之外,可以使用诸如SCTP和UDP协议之类的其他协议,以及其他适于传输数据流的协议。

在企业 and 全球范围内,计算机越来越多地进行联网。这些联网的计算机正在改变信息管理和安全性的范例。现在在网络上生成、访问和存储大量信息(包括高度机密、个人和敏感的信息),需要保护这些信息以防止未授权访问。此外,通过电子邮件、Web访问、即时消息传送、Web下载和其他方式,对用户进行连续的垃圾邮件、病毒和其他不适当内容的攻击,导致生产力和资源的严重损失。

企业和服务提供商网络的线路速度快速地从10/100Mbps发展到1Gbps、10Gbps和更高的线路速度。由于组织的外围边界变得模糊,用于保护信息系统的传统外围安全模型存在许多问题。如今,由于雇员、承包商、远程用户、合作者和客户都需要从外部访问企业网络,因此一个外围安全模型是不够的。对于这些组织的重要信息和计算资源,此使用模型将产生严重的安全漏洞。因此,必须通过网络核心的安全性来支持传统的外围安全模型。此外,新的威胁源和高线路速度网络的组合使得基于软件的外围安全性不足以停止外部和内部攻击。显然,需要在核心内部硬件或外

围防火墙外的终端系统中启用安全性处理，作为主要安全措施之一来防止日益增加的安全漏洞和攻击。

近年来，FBI 和其他主要研究机构已报告组织中 70% 以上的入侵是内部的。因此，如上所述，借以保护组织免受外部攻击的外围防御是不够的。还需要组织监视出站业务以防止意外或恶意泄露私有和机密信息，以及防止其网络资源被用于传播垃圾邮件、病毒、蠕虫和其他恶意软件。显然，需要检查网络业务的数据有效负载，以保护组织网络的入站和出站安全性。

使用 TCP/IP 或其他协议传输的数据在网络中的源、目的地或中间系统或它们的组合中处理，以便提供数据安全性或其他服务，如用于套接字层安全性的安全套接字层（SSL）、传输层安全性、加密/解密、RDMA、RDMA 安全性、应用层安全性、虚拟化或更高级的应用层处理，所述处理可进一步包括应用级协议处理（例如，用于 HTTP、HTTPS、XML、SGML、安全 XML、其他 XML 派生物、Telnet、FTP、IP 存储、NFS、CIFS、DAFS 等的协议处理）。许多这些处理任务都会给主机处理器带来可能直接影响应用和硬件系统性能的沉重负担。因此，其中某些任务需要使用专用硬件（例如 SSL 或 TLS 加速）进行加速。随着 Web 应用对 XML 使用的增加，预期给主机处理器带来沉重的性能负担并且也将从硬件加速中获益。检测垃圾邮件、病毒和其他不适当的内容需要深度分组检查和分析。此类任务可能给主机处理器带来极大的处理负担并且可能会显著降低网络线路速度。因此，还需要深度分组内容搜索和分析硬件。

互联网已成为小型到大型组织开展业务的重要工具。近几年来，随着基于 XML 的服务的部署，基于 HTML 的静态 Web 正在转换为动态环境。XML 正在成为混合语 Web，并且预期其使用将显著增加。XML 是一种描述性语言，通过使文档成为自描述以进行自动处理而提供许多优点，但还公知的是导致最佳服务器处理器极大的性能开销。可以通过处理 XML 文档中嵌入的智能做出决策，以实现企业对企业交易以及其他信息交换。但是，由于分析 XML 文档对最佳服务器处理器产生的过多性能负担，因此它们不能用于需要网络线路速度 XML 处理以提供智能联网的系统中。显

然，需要用于以接近 1Gbps 和 10Gbps 的网络线路速度进行 XML 文档分析和内容检查的加速解决方案，以实现基于 XML 服务的动态 Web 的优点。

正则表达式可用于表示用于如上面讨论的各种应用的内容搜索字符串。然后，一组正则表达式可以形成用于搜索特定应用的规则集，并可应用于任何文档或数据流以对其进行检查。正则表达式用于描述反垃圾邮件规则、反病毒规则、XML 文档搜索构造等。这些表达式被转换为 NFA 或 DFA 以在通用处理器上进行计算 (evaluation)。但是，每一类表示将产生显著的性能和存储限制。例如，对于 DFA 的各状态，N 字符正则表达式将占用约 2^N 个存储单元，而对于 NFA 的各状态，N 字符正则表达式将占用约 N 个存储单元。另一方面，在现代微处理器上，对于 M 字节输入数据流，DFA 计算性能为约 M 次存储器访问，对于 NFA 表示，为约 $N*M$ 个处理器周期。

当正则表达式的数量增加时，对性能的影响也会增大。例如，在诸如反垃圾邮件之类的应用中，可能具有数百个正则表达式规则。可以使用单独的 NFA 或 DFA 在服务器处理器上计算这些正则表达式。还可以创建复合 DFA 以表示这些规则。假设某个应用存在 X 个 RE，则每个单独 RE 的基于 DFA 的表示结果将达到约 $X*2^N$ 种状态，但是计算时间将增至约 $X*N$ 个存储周期。通常，由于 DFA 状态数的潜在扩大，需要将它们存储在片外存储器中。使用主存储系统的 100ns 的典型访问时间滞后，在 M 字节数据流上，将需要约 $X*100ns*N*M$ 的时间来处理具有 N 种状态的拥有 X 个 RE 的 DFA。对于适度大小的 X、N 和 M，这可以导致数十 Mbps 的性能。显然，此性能远远低于当今 1Gbps 到 10Gbps 的网络线路速度的需要。另一方面，如果创建了复合 DFA，它可以导致约 2^{N*X} 的存储上限，即使对于几百个 RE 而言，此上限也可能不在典型商用计算系统的存储器大小的物理限制内。因此，DFA 的存储器扩展的上限可能是一个重要的问题。而另一方面，NFA 本质上是非确定性的并可以导致能够同时发生的多个状态转移。NFA 只能以标量方式在最新的微处理器中处理，导致对于每个已启用的路径多次执行 NFA。X 个具有 N 个字符的 RE 在平均上可以以上限 ($X*N$)

种状态表示为 NFA。但是，每个 NFA 对于 M 字节的流将需要 M 次迭代，从而导致上限 ($X*N*M$ 个处理器周期/循环)。假设处理周期数约为 10 个周期，则对于 4 GHz 的最佳处理器，处理时间可能约为 $X*N*M*2.5\text{ns}$ ，这对于 N 为 8 且 X 为数十的额定情况，可能导致低于 100 Mbps 的性能。显然，需要创建基于高性能正则表达式的内容搜索处理器，所述处理器可以提供与达到 1Gbps 和 10Gbps 的网络速度一致的性能。

将正则表达式转换为 NFA 和 DFA 的方法是公知的。生成的自动机能够辨别字符串是否属于正则表达式所定义的语言，但是它不能高效地确定正则表达式的特定子表达式是否在匹配字符串中或确定字符串的范围。带标记的 NFA 使此类查询能够高效地进行而不必再次扫描匹配字符串。有关带标记的 NFA 的讨论，请参阅由芬兰赫尔辛基理工大学的 Ville Laurikari 所著的“NFAs with Tagged Transitions, their Conversion to Deterministic Automata and Application to Regular Expressions”。

美国专利申请 20040059443 和 20050012521 描述了一种用于有效实施和计算状态机和可编程有限状态自动机的方法和装置。这些申请说明了用于使用一组 NFA 计算正则表达式以创建高性能正则表达式处理的装置。所述申请认识到 DFA 的存储上限问题，作为使用 NFA 实现正则表达式的原因。但是，所述申请未能认识到即使 DFA 最差的存储要求比 NFA 高出很多，但是许多 DFA 比 NFA 具有更少的存储需求。许多正则表达式的 DFA 可能导致比 NFA 更少的状态数。例如在反垃圾邮件应用中，根据开源工具 SpamAssassin，大量正则表达式规则导致小于 NFA 的 DFA。因此，重要的是不要忽略仅基于最差情况的 DFA 实现。这些专利申请还创建了按照 NFA 块处理单个 RE 的 NFA 引擎。因此，如果 RE 使用的状态少于 NFA 块的最少状态，则没有设备能够在同一块中同时使用多个 RE。在本发明中，描述了使用一组运行时自适应搜索引擎的内容搜索处理器，其中所述搜索引擎可以是用于计算正则表达式的运行时自适应 DFA 搜索引擎或运行时自适应 NFA 搜索引擎或它们的组合。所述搜索处理器的内容搜索引擎还提供了按照 NFA 或 DFA 引擎使用多个 RE 的灵活性。本发明还提供了

支持带标记 NFA 实现（在这些申请中不支持该实现或未讨论该实现）的能力。此外，这些申请没有解决根据被发送到网络或从网络接收的传输数据来动态配置要应用的硬件或规则的需要。本发明的处理器可以动态地适应应用与传输数据相关的基于硬件的规则集，这在上述申请中并未描述。此外，本发明表明某些 DFA 可以是比 NFA 更高效实现的硬件资源，并且可使当今最新 FPGA 能够实现大量正则表达式，而不必花费大量投资来使用高级工艺技术创建专用集成电路。这还将专门讨论，因为这在上述申请中是不可行的。本发明还显示内容搜索加速可用于通过本发明的内容搜索应用编程接口（API）和搜索处理器来改进应用加速。

当为每个单独类型的网络数据部署专用加速器时，用于每种网络数据有效负载的硬件加速会很昂贵。显然，需要一种处理器体系结构，它可以使自身适应网络数据的需要，提供必要的加速并由此减小对主机性能的影响。本申请描述了此类使自身适应网络数据需要的新颖体系结构。本申请的处理器可以被重新使用并适合不同类型的有效负载的不同需要，并且仍提供硬件加速的优点。与专用的特定于应用的加速器相比，这可以显著降低加速解决方案部署的成本。

动态可重新配置计算是已得到大量研发关注的领域，以解决重新配置硬件资源的需求以便满足应用需要。研究的主要焦点集中于创建可以适应新指令执行资源以满足应用需要的通用微处理器替代物。

现场可编程门阵列（FPGA）已从简单的“与或”逻辑块发展为更为复杂的元件，所述元件提供大量可编程逻辑块和可编程路由资源以将这些逻辑块连接在一起或连接到输入/输出块。美国专利 No. 5,600,845 描述了包括动态可配置 FPGA 的集成电路计算设备。门阵列被配置为创建具有可配置指令执行单元的 RISC 处理器。此动态重新配置能力允许动态可重新配置指令执行单元改为在硬件中实现在软件中运行可能很费时的操作。此类布置需要预配置的指令集以执行进站指令，并且如果指令不存在，则必须将其视为具有显著处理开销的异常。美国专利 No. 5,600,845 中的发明解决了通用微处理器的限制，但是没有解决根据被发送到网络或从网络接收的

传输数据动态配置硬件的需要。

美国专利申请 No. 20030097546 描述了一种可重新配置的处理器，所述处理器接收指令流，指令测试模块检查该指令流以判定现有的不可重新配置硬件或通过软件例程配置的可重新配置硬件是否支持该指令，并且所述处理器根据测试结果来执行指令流。如果不支持该指令，则处理器确定要采取的操作，包括在软件中执行该指令流。专利申请 No. 20030097546 同样没有解决根据被发送到网络或从网络接收的传输数据动态配置硬件的需要。

美国专利申请 No. 20040019765 描述了一种流水线可重新配置动态指令集处理器。在此申请中，描述了在微控制器控制下的动态可重新配置流水线级。这是另一种动态可重新配置处理器，所述处理器可以根据被处理的指令调整其流水线级及各级的互连，以作为通用微处理器的替代物。

可重新配置计算领域已通过在创建动态可重新配置逻辑设备作为 FPGA 或可重新配置处理器（如上所述，主要通过添加可重新配置执行单元或可重新配置协处理器来解决通用处理器的限制）方面的研究日趋成熟。例如，瑞士联邦理工学院的 Andreas Romer 所著的毕业论文“Reconfigurable FPGA processor”专注于通过在可重新配置 FPGA（如 Xilinx 公司的 Virtex 和 XC6200 设备）的可重新配置部分中动态创建执行功能单元，来创建类似 ASIC 的性能和面积（但具有通用处理器级灵活性）的需要。同样，J.R. Hauser 和 J Wawrzynek 的题为“*Garp: A MIPS Processor With a Reconfigurable Coprocessor*”的论文（发表在关于定制计算机器的 FPGA 的 IEEE 讨论会学报（FCCM’97）中）专注于为 MIPS 处理器创建解决 MIPS 处理器的通用处理能力限制的定制协处理支持的需要。

已发表的研究或专利申请没有解决根据传输数据以及针对被发送到网络或从网络接收的此特定数据而采取的操作和部署的应用/服务来动态配置硬件的需要。此申请描述了一种新颖的体系结构，此体系结构使自身适应网络数据的需要并且是运行时自适应的，以执行结合本发明的系统的传

输数据和定义策略的耗时安全策略操作或应用/服务或其他数据处理需要。此体系结构还包括深度分组检查引擎，此引擎可用于检测垃圾邮件、病毒、数字版权管理信息、即时消息检查、URL 匹配、应用检测、恶意内容，以及其他内容并应用可实现反垃圾邮件、反病毒等能力的特定规则。

发明内容

本发明描述了针对正则表达式使用硬件加速的高性能运行时自适应搜索处理器。将正则表达式转换为等价的 DFA 或 NFA，然后选择最经济有效的解决方案。内容搜索处理器包括一组 DFA 处理引擎和 NFA 处理引擎。然后将所转换的 RE 映射到适当的 NFA 或 DFA 引擎。所述处理器还可以包括一组复合 DFA 处理引擎，它们可用于承担超出 FSA 引擎组支持的规则数的数量增长。因此，在复合 DFA 导致存储器使用量明显超出与内容搜索处理器关联的可用存储器之前，无需新的系统硬件。

在许多内容搜索应用（如安全性）中，需要不断地更新规则或用于检测恶意业务的签名。在此类应用中，重要的是解决方案可适应跟上安全威胁不断发展的特性。在始终连接类型的使用模型中，极为重要的是经常安全系统中更新最新的安全威胁缓解规则。当使用复合 DFA 类型体系结构时，编辑和发布任何新的安全规则或策略会消耗大量时间，其中更新可能无法及时避免安全威胁的影响。在此类环境中，发布新的规则库可能占用 8 到 24 小时，这严重延迟了对不断发展的威胁的响应。在本发明的处理器中，此问题得到解决，因为新规则的发布就是将这些规则转换为 NFA 和 DFA 并且仅将这些非常小的规则更新到内容搜索处理器中。因此，对新威胁的响应可以非常及时，并且将新规则集成到复合规则库中并将它们转换为复合 DFA 不需要极大的延迟。

存在若干仅包括少数状态的 RE 实例。例如，如果内容搜索包括查找 *.exe 或 *.com 或 *.html 等，则用于这些 RE 的 NFA 或 DFA 包括少量状态。因此，如果所有 DFA 或 NFA 引擎支持例如 16 种状态，则每个引擎可以包括多个规则。本发明通过允许每个 FSA 引擎包括多个规则，实现了最大

程度地利用 FSA 引擎。所述引擎还提供 FSA 扩展逻辑以将基础引擎链接在一起，从而创建可以处理更大 FSA 的超级块。

Berry 和 Sethi 在其发表于 1986 年的《Theoretical Computer Science》上的论文“From Regular Expressions to Deterministic Automata”中，说明了正则表达式可以由 NFA 表示，以便通过一个符号来输入状态机中的给定状态，这一点与 Thompson NFA 不同。此外，Berry-Sethi NFA 没有 ε (ε -free)。可以使用“V+1”种状态 NFA（使用类似 Berry-Sethi 的 NFA 实现方法）表示‘V’项 RE。如果所有导致机器脱离状态的转移依赖于同一符号，则还存在 Berry-Sethi 方法的二元性。这在 Bruce Watson 发表于 1994 年的论文“A Taxonomy of finite automata construction algorithms”的 4.3 节中进行了说明。本发明说明了一种用于利用 Berry-Sethi NFA 实现的原理及其构造的二元性来创建 NFA 搜索引擎体系结构的方法。NFA 搜索引擎是可编程的，以便实现任意正则表达式。

在本发明中，还说明了如何使用本发明的内容搜索处理器在计算设备（如服务器、个人计算机、工作站等）中创建总体应用加速。本发明说明了可以用作常规工具的实例内容搜索 API，此 API 可以由这些设备的操作系统提供给在其上运行的应用（所述应用可以利用该内容搜索处理器），与使应用在这些设备的通用处理器上运行相比，这显著提高了这些应用的性能。

在本申请中说明了反垃圾邮件的实例应用，此应用可以被加速以成为高线路速度应用，其不同于在运行这些应用的计算服务器的通用处理器上运行的当前解决方案。

本发明还说明了使用与协议处理器相连的内容搜索处理器，以便可以根据业务进行内容搜索处理以及根据业务内容采取适当操作的实例。因此，本发明的处理器可以用于针对业务流应用内容特定的搜索规则。例如，如果特定分组或分组流承载 SMTP 业务，则协议处理器可以使内容搜索处理器知道此情况并提供适当的流信息。然后，内容搜索处理器从存储器中检索当前流的流上下文，并从与搜索引擎关联的应用存储器中检索 SMTP 应

用规则上下文。搜索引擎被配置为处理特定流的内容及其关联应用上下文。因此，即使当流中的各分组相隔很长时间到达，内容搜索也可以继续跨同一流的多个分组，并且还可以应用多个不同的应用规则上下文。与无法迅速更改规则上下文并且可能需要将上下文从全局存储器放入每个 FSA 引擎的体系结构相比，通过使用此类体系结构可以获得显著的性能优点。

本发明还描述了显著降低主机处理器中 TCP/IP 协议栈开销并实现基于 IP 的高线路速度存储和数据传输解决方案的高性能硬件处理器。

本申请还描述了显著降低主机处理器中 TCP/IP 协议栈开销并能够进行高线路速度安全性处理（包括防火墙、加密、解密、入侵检测等）的新颖高性能处理器。本申请还描述了可用于检测垃圾邮件、病毒、数字版权管理信息、即时消息检查、URL 匹配、应用检测、恶意内容和其他内容以及应用可启用反垃圾邮件、反病毒等功能的特定规则的内容检查体系结构。所述内容检查引擎可用于检测和实施内容的数字版权管理规则。所述内容检查引擎还可用于 URL 匹配、字符串搜索、基于内容的负载平衡、敏感信息搜索，如信用卡号或社会保险号或健康信息等。所述内容检查引擎结果也可以用于指导运行时自适应处理器的操作。

本申请还描述了可在运行时适应于发送到网络或从网络接收的数据的需求的新颖处理器体系结构。此处理器的运行时自适应特性可用于部署在用户可定义的策略的控制下处理网络数据的服务。当未对网络数据执行处理时或在有足够的自适应硬件资源可用的情况下用于网络数据处理时，所述自适应处理器还可以用于动态地从主机处理器卸载计算密集型操作。所述处理器使用所披露的高性能协议处理器执行诸如 TCP/IP 或 SCTP 或 UDP 之类的协议处理，然后使用自适应处理硬件提供其他功能或服务，如套接字层安全性、传输层安全性、加密/解密、RDMA、RDMA 安全性、应用层安全性、内容检查、深度分组检查、病毒扫描或检测、策略处理、基于内容的转换、负载平衡、基于内容的负载平衡、虚拟化或更高的应用层处理或上述内容的组合。更高层的处理可以进一步包括还可以通过动态地改变或重新配置本申请的处理器来加速的应用级协议处理（例如 HTTP、

HTTPS、XML、SGML、安全 XML、其他 XML 派生物、Telnet、FTP、IP 存储、NFS、CIFS、DAFS 等的协议处理)。这可以在不增加因为添加专用加速器硬件而产生的主要系统成本的情况下显著降低目标系统上主机处理器的处理开销。

部署了本申请的运行时自适应处理器的系统的处理能力可以在无需持续升级具有主机处理器的系统的情况下继续扩展和改善以实现性能优点。所述处理器的硬件可以包括组织为计算群集的计算元件。计算元件可以提供逻辑运算和算术运算以及其他功能。计算元件可以处理 1 位、2 位、4 位、8 位或 n 位数据大小，这可以根据实施方式来选择。因此多个计算元件可以共同提供期望大小的操作数。例如，如果每个计算元件提供最大 8 位数据大小的处理，为了处理 32 位操作数，则四个计算元件中的每个计算元件可以处理 32 位操作数的一个字节。所述计算群集内的各计算元件可以使用可动态更改或自适应的互连网络以编程的方式彼此互连。所述计算群集也可以彼此以编程的方式动态地互连以形成自适应网络。因此，可以在计算群集内的计算元件阵列之间以及计算群集以外创建任意互连。所述计算元件中的每个元件可以通过以编程的方式选择和连接必要的逻辑块（通过多路实现的逻辑块或其他逻辑块或其他方式）来被动态地更改，以便提供必要的一个或多个功能或操作。所述计算元件还可以是具有 ALU 和其他功能块的简单处理器。在这种情况下，为了更改计算元件 (CE) 的硬件功能，需要以编程的方式执行一个或多个特定功能或操作。为每个计算元件选择的操作（多个）可以是不同的。因此，本申请的处理器硬件可以在操作或运行时被动态地更改或改变以提供不同的功能。为了说明，以具有 8 个计算元件（每个元件提供 8 位运算）的计算群集为例。如果要执行两个 32 位运算（例如，一个 32 位加法后跟一个 n 位移位运算），则可以将各计算元件分为两个组，每个组包含四个元件。第一组将被编程为提供加法运算，其中每个元件一次可以处理 8 位。可以通过能够以编程方式选择的 CE 之间的互连提供适当的进位和标志以及其他输出。第二组 CE 将被编程为提供对进站数据的移位运算。一种此类 CE 设置可以称为具体化

(Avatar)或虚拟配置/设置。然后在维持此具体化的时间内,CE可以继续提供对输入操作数的这些运算。然后可以动态地将该具体化更改为新的具体化。例如在上述实例中,假定在特定时间段之后(其可以为一个时钟周期或多个时钟周期或其他周期),处理器需要从提供对32位加法后跟移位运算的加速支持切换到两个16位减法后跟两个16位逻辑‘与’运算。在这种情况下,将硬件设置为形成四组(每组2个)CE,其中每组处理16位操作数。这种情况下,前四个CE现在可以被动态地从提供加法功能切换或更改为提供减法功能。此外,它们现在可以动态地切换为以两组进行处理来提供16位运算,而不是以一组进行处理以提供先前具体化中的32位运算。同样,先前具体化中的后四个CE现在可以动态地切换或更改为提供逻辑‘与’运算,并且还可以设置为提供16位运算的两个组。这形成硬件的新具体化,此具体化已经根据所需功能在当时的需要被动态地更改。因此本申请的运行时自适应协议处理器可以细致的粒度更改其功能以及由CE提供的这些运算符的互连以形成运行时可更改或自适应硬件平台。所支持的运算可以比上述实例中使用的运算更为复杂。所提供的实例主要是为了更好地理解所述能力并且有意使这些实例简单化。虽然所述实例建议了单向流,但是不应将这解释为唯一的工作模式。上述实例中的运算输出可以重复循环到第一组CE,这将允许硬件流水线循环。借助CE的动态自适应特性和互连网络,可以实行更复杂的方案,其中可随时间切换CE的不同级以提供硬件中启用的算法或应用或服务所需的不同功能。因此运算的流水线级可以根据需要具有任意环回。从而,在硬件中加速或支持的应用或服务可以随时间而增加,其中用户可以决定根据需要以及由于成本、性能、资源、应用发现、应用开发或任何其他原因(可能导致他们发明或开发新应用或服务)而可行时,通过将应用适当地映射到运行时自适应协议处理器来加速所选应用。因此硬件系统可以适应于使用此类应用而没有购买新系统或加速器等带来的成本。随着开发和/或部署使用本发明处理器的自适应组件的新服务,系统功能可随时间增加。可以在用户控制下,通过网络将新服务或策略或其组合部署到相应的系统。

传统上, TCP/IP 联网栈在操作系统内核内部作为软件栈实现。如上所述, 当提供 1Gbps 网络时, 软件 TCP/IP 栈实现消耗了 1GHz 处理器中 50% 以上的可用处理周期。该开销来自软件 TCP/IP 栈的各个方面, 包括校验和计算、存储器缓冲区复制、分组到达时的处理器中断、会话建立、会话关闭以及其他可靠传输服务。在较高线路速度时, 软件栈开销将变得高得惊人。在线路速度较低的网络(如使用较低性能的主机处理器的无线网络)中也会出现类似问题。硬件实施方式可以消除主机处理器中的开销。

操作系统提供的软件 TCP/IP 联网栈将用尽大部分主机处理器周期。TCP/IP 是可以在不可靠数据链路上运行的可靠传输。因此, 当网络分组丢失或具有错误时, TCP 将重新传输分组。使用分组内承载的校验和来检测分组中的错误。TCP 分组的接收方计算所接收分组的校验和并将其与所接收的校验和相比较。这是对每个分组(包括分组中每个接收的字节)执行的昂贵的计算密集的操作。源和目的地之间的分组可能不按顺序到达, 并且 TCP 层在将分组提供给上层之前执行数据流排序。IP 分组也可能根据链路层的最大传输单元(MTU)进行分段, 因此接收方需要取消分组分段。这些功能导致将不按顺序的分组、分段分组或未确认的分组临时存储在例如网卡上的存储器中。当线路速度增加到高于 1Gbps 时, 存储器大小开销以及这些开销导致的存储器速度瓶颈显著增加了网卡成本并且还导致极大的性能开销。另一种消耗大量处理器资源的功能是将数据复制到网卡缓冲区、内核缓冲区和应用缓冲区以及从其中复制数据。

微处理器使用深度流水线和超标量体系结构不断实现其高性能和速度。当较小分组到达时中断这些处理器将由于上下文切换开销、流水线刷新和流水线填充而导致严重的性能降低。因此应将中断处理器尽量减少到只有最重要的中断。当通过 TCP/IP 网络传输块存储业务时, 这些性能问题变得至关重要, 严重影响了存储业务的吞吐量和等待时间。因此, 对于基于 IP 的存储解决方案, 需要尽量减少传输存储业务的整个过程中的处理器介入, 以便具有与其他指定为硬件实现的专用网络体系结构(如光纤通道)可比的性能和等待时间。新出现的如 iSCSI、FCIP、iFCP 和其他协议

(如 NFS、CIFS、DAFS、HTTP、XML、XML 派生物(如语音 XML、EBXML、微软 SOAP 等)、SGML 和 HTML 格式)之类的基于 IP 的存储标准将存储和数据业务封装在 TCP/IP 段中。但是,由 TCP 分组封装的 TCP 段和协议数据单元之间通常没有对齐关系。当分组不按顺序到达时这便成为问题,这是当今网络中相当常见的情况。在流中的中间分组到达之前,无法从不按顺序的分组中提取存储和数据块以供使用,这将导致网络适配器将这些分组存储在存储器中,检索它们并在中间分组到达时对其进行排序。由于所需存储器存储的大小以及期望存储子系统支持的性能,尤其是在线路速度超过 1Gbps 时,此操作的代价可能非常高。如果每个 TCP 段可以唯一地标识协议数据单元及其顺序,则可以消除此开销。这可以允许将分组直接传输到它们在主机系统中的最终存储单元。在可以传输到存储子系统或与群集环境或其他客户机服务器环境中的其他处理器共享的大型数据块的传输中,也应该尽量减少主机处理器介入。应仅在存储命令边界上中断处理器以尽量减小影响。

在此说明的 IP 处理器通过创新的体系结构特性和设计消除或显著减小了上述各种问题的影响。所述处理器体系结构提供了用于终止承载存储和数据有效负载的 TCP 业务的功能,从而消除或显著降低了主机处理器上的 TCP/IP 联网栈开销,导致允许分组从输入到输出以最小等待时间通过的分组流体系结构。为了能够通过 IP 承载高线路速度存储或数据业务,需要维护传统上由主机内核或驱动软件所维护的各种连接(会话)的传输控制块信息。如本申请中所使用的,术语“IP 会话”表示在 IP 上运行的面向会话的协议的会话。实例为 TCP/IP、SCTP/IP 等。访问每个分组的会话信息显著增加了处理开销。所述体系结构创建了显著降低此开销的高性能存储子系统。所述处理器体系结构主要在命令或数据传输完成边界为主机处理器提供了尽量减少中断的智能流控制功能。

当今, TCP/IP 处理器都不具有安全性。

当今部署的常规网络安全性模型包括形式为外围防火墙和入侵检测系统的外围安全性。但是,随着越来越多的业务在线进行,需要为“受信任

内部人员” - 雇员、合作者、客户和承包者提供从外部对企业网络的访问。这将对企业网络内部的信息资产产生潜在的威胁。主要公司和 FBI 的最近研究发现 70%以上的对信息系统未授权访问是由雇员或受信任内部人员进行的，并且因此 95%以上的入侵导致巨大的财务损失。在远程访问服务器、与合作者的对等网络、VPN 和无线访问点使网络边界模糊的环境中，外围安全性并不足够。在此类环境中，组织需要采取解决所有层（包括外围、网关、服务器、交换机、路由器和客户机）的网络安全性的集成策略，而不是在外围使用点安全性产品。

传统的防火墙通过将入侵 IP 地址阻止在内部网络之外而在网络层提供外围安全性。但是，由于许多新的攻击作为利用众所周知的软件和较高级协议的公知漏洞的病毒或垃圾邮件到达，因此需要开发和部署应用层防火墙。这些防火墙还应跨网络分布而不是主要分布在外围。

当前，由于 TCP/IP 处理作为软件栈存在于客户机、服务器以及其他核心和终端系统中，因此还以软件（尤其是诸如防火墙、入侵检测和防护之类的功能）执行安全性处理。当这些网络的线路速度达到 1Gbps 和 10Gbps 时，必须以硬件实现 TCP/IP 协议栈，因为软件栈将消耗大部分的可用主机处理器周期。同样，如果在核心或终端系统处部署安全性处理功能而不是仅在外围部署，则执行这些操作所需的处理能力会在这些系统的主机处理器上产生极大的开销。因此，基于软件的分布式安全性处理将增加系统的所需处理能力并增加部署此类解决方案的成本。基于软件的实施方式将有损服务器的性能并显著增加服务器对客户机的响应延迟或等待时间，而且可能限制所服务的客户机数。此外，如果主机系统软件栈在网络攻击期间受到损坏，可能无法隔离安全性功能，从而危及网络安全性。此外，由于 TCP/IP 协议处理逐渐以硬件执行，因此软件网络层防火墙可能没有对执行安全性功能所需的所有状态信息的访问权限。因此，可能需要协议处理硬件提供对它处理的协议层信息的访问权限，并且主机可能必须重新执行某些功能以满足网络防火墙需求。

本申请的基于硬件的 TCP/IP 和安全性规则处理处理器解决了分布式

核心安全性处理瓶颈，同时还解决了 TCP/IP 协议栈的性能瓶颈。本申请的硬件处理器显著降低了主机 CPU 中的 TCP/IP 协议栈处理开销，并且在诸如链路、网络和传输层之类的各种协议层处实现了安全性处理功能（如防火墙），从而极大提高了预期应用的主机 CPU 性能。此外，此处理器提供了可以用于使用所披露的可编程处理和分类/策略引擎执行深度分组检查以执行更高层安全性功能的能力。本申请还描述了一种内容检查体系结构，其可以用于检测垃圾邮件、病毒、数字版权管理信息、即时消息检查、URL 匹配、应用检测、恶意内容以及其他内容，并应用可实现反垃圾邮件、反病毒等安全性和内容检查和处理功能的特定规则。因此，本申请的处理器能够在 OSI 栈的所有层处进行硬件 TCP/IP 和安全性处理，以便在所有层（包括网络层和应用层）实施诸如防火墙之类的功能。

本申请的处理器体系结构还提供了集成高级安全性功能。此处理器允许以分组为基础对分组上的网络业务进行流中加密和解密，从而允许高线路速度并同时提供数据业务的机密性。同样，当通过网络将存储业务从服务器运载到 SAN 或其他存储系统中的存储阵列时，它将暴露于直接连接存储系统不必处理的各种安全性漏洞。此处理器允许对存储业务进行流中加密和解密，从而允许高线路速度并同时提供存储数据业务的机密性。

网络业务的分类是另一项消耗分组处理器中一半可用处理周期的任务，剩下很少的周期以进行分组检查和处理。按照协议的特性，基于 IP 的存储业务需要高速度低等待时间的深度分组处理。所述 IP 处理器通过提供可编程分类引擎显著降低了分类开销。本申请的可编程分类引擎允许部署在分组、事务和流基础上实施的高级安全性策略。这将显著改善以高性能且经济有效的方式部署分布式企业安全性解决方案，以解决组织内出现的安全性威胁。

为了能够创建分布式安全性解决方案，必须解决信息技术管理者的需要以便经济有效地管理整个网络。如果添加分布式安全性而没有易于管理它的方法，则会显著增加网络的管理成本。本申请所披露的内容还提供了安全性规则/策略管理功能，IT 人员可使用该安全性规则/策略管理功能从

中央位置向使用本申请的处理器的各种内部网络系统分发安全性规则。所述处理器包括可以与中央规则管理系统（多个）交互的硬件和软件功能。因此，IT 管理者可以从一个或多个中央位置处理安全性规则的分发以及规则符合性或违反性信息或其他诸如入侵系统、用户之类的相关信息的收集。因此可以从一个或多个中央位置分别控制多个分布式安全性部署。

本申请还提供了用于创建协议栈处理的安全操作环境的方式，即使主机系统受到病毒或恶意攻击的破坏，其也允许维护网络安全性和完整性。本申请显著增加了下一代计算系统的受信计算环境需求。

存储容量和存储网络的飞速增长使得存储区域管理成为 IT 部门的主要成本项。需要基于策略的存储管理以包含管理成本。所述可编程分类引擎允许部署可在分组、事务、流和命令边界基础上实施的存储策略。这将对存储区域管理成本具有显著改进。

可编程 IP 处理器体系结构还提供了足够的空间以允许部署客户特定的应用。这些应用可以属于多个类别，例如网络管理、存储防火墙或其他安全性功能、带宽管理、服务质量、虚拟化、性能监视、分区、LUN 屏蔽等。

所述自适应处理器硬件可用于根据可用的可重新编程资源、已部署的应用、服务、策略或它们组合，来加速上面列出的许多应用或服务。

附图说明

图 1 示出了分层 SCSI 体系结构以及位于发起系统和目标系统之间的各层间的交互；

图 2 示出了在发起系统和目标系统之间具有基于 iSCSI 和 TCP/IP 的传输的分层 SCSI 体系结构；

图 3 示出了基于软件的 TCP/IP 栈与诸如光纤通道之类的面向硬件的协议的 OSI 栈比较；

图 4 示出了具有基于硬件的 TCP/IP 实现以提供与其他面向非 IP 硬件的协议同等的性能的 OSI 栈；

图 5 示出了具有实施联网栈和存储栈的操作系统层的主机软件栈;

图 6 示出了软件 TCP 栈数据传输;

图 7 示出了使用如本申请所述的主处理器中的 TCP/IP 卸载的远程直接存储器存取数据传输;

图 8 示出了用于通过 IP 网络传输块存储数据的主机软件 SCSI 存储栈层;

图 9 示出了本发明的实施例的特定 iSCSI 网络层栈的详细信息;

图 10 示出了本发明的实施例的 TCP/IP 网络栈的功能详细信息;

图 11 示出了通过本发明的实施例的各种元件的 iSCSI 存储数据流;

图 12 示出了可在本发明中使用的 iSCSI 存储数据结构;

图 13 示出了可在本发明的实施例中使用的会话数据库表项的 TCP/IP 传输控制块数据结构;

图 14 示出了可在本发明的实施例中使用的 iSCSI 会话数据库结构;

图 15 示出了可在本发明的实施例中使用的 iSCSI 会话存储器结构;

图 16 示出了可在本发明的实施例中使用的 IP 网络应用处理器的高级体系结构块图;

图 17 示出了图 16 中的 IP 网络应用处理器的体系结构块图的详细视图;

图 18 示出了 IP 处理器的一个实施例的输入队列和控制器。

图 19 示出了可在 IP 处理器的一个实施例中使用的分组调度器、定序器和负载均衡器;

图 20 示出了分组分类引擎, 包括 IP 存储处理器的一个实施例的策略引擎块;

图 21 在更高的级别上概括示出了 IP 处理器的一个实施例的 SAN 分组处理器块的实施例;

图 22 进一步示出了所述的 IP 处理器的 SAN 分组处理器块的实施例;

图 23 示出了可用作所述的 SAN 分组处理器的一部分的可编程 TCP/IP 处理器引擎的实施例;

图 24 示出了可用作所述的 SAN 分组处理器的一部分的可编程 IP 存储处理器引擎的实施例;

图 25 示出了图 17 中的可编程 IP 处理器的输出队列块的实施例;

图 26 示出了存储流控制器和 RDMA 控制器的实施例;

图 27 示出了可在本发明的实施例中使用的 IP 处理器的主机接口控制器块的实施例;

图 28 示出了安全引擎的实施例;

图 29 示出了可在所述处理器中使用的存储器和控制器的实施例;

图 30 示出了可在所述的分类引擎的实施例中使用的数据结构;

图 31 示出了发起方和目标之间的存储读取流;

图 32 示出了通过所述处理器的各流水线级的读取数据分组流;

图 33 示出了发起方和目标之间的存储写入操作流;

图 34 示出了通过所述处理器的各流水线级的写入数据分组流;

图 35 示出了位于发起方和目标之间并使用发起方和目标间的远程 DMA (RDMA) 功能的存储读取流;

图 36 示出了位于发起方和目标之间并使用 RDMA 的通过所述处理器的各流水线级的读取数据分组流;

图 37 示出了位于发起方和目标之间并使用 RDMA 功能的存储写入流;

图 38 示出了使用 RDMA 的通过所述处理器的各流水线级的写入数据分组流;

图 39 更详细地示出了通过所述处理器的各个流水线级的发起方命令流;

图 40 更详细地示出了通过所述处理器的各个流水线级的读取分组数据流;

图 41 更详细地示出了通过所述处理器的各个流水线级的写入数据流;

图 42 示出了当分组采用密文形式或者是通过所述处理器的各个流水线级的安全分组时的读取数据分组流;

图 43 示出了当分组采用密文形式或者是通过本发明的一个实施例的所述处理器的各个流水线级的安全分组时的写入数据分组流;

图 44 示出了通过所述处理器的各个流水线级的 RDMA 缓冲广告流;

图 45 更详细地示出了通过所述处理器的各个流水线级的 RDMA 写入流;

图 46 更详细地示出了通过所述处理器的各个流水线级的 RDMA 读取数据流;

图 47 示出了通过所述处理器的各个流水线级的会话创建流的步骤;

图 48 示出了通过所述处理器的各个流水线级的会话关闭流的步骤;

图 49 示出了从目标的角度通过所述处理器的各个流水线级的会话创建和会话关闭步骤;

图 50 示出了通过所述处理器的各个流水线级的目标子系统 R2T 命令流;

图 51 示出了通过所述处理器的各个流水线级的目标子系统写入数据流;

图 52 示出了通过所述处理器的各个流水线级的目标读取数据流;

图 53 示出了具有外围安全性的典型企业网络;

图 54 示出了具有分布式安全性并使用本申请的各种元素的企业网络;

图 55 示出了具有包括存储区域网络安全性的分布式安全性并使用本申请的各种元素的企业网络;

图 56 示出了中央管理器/策略服务器和监视站;

图 57 示出了所披露的安全功能的中央管理器流;

图 58 示出了中央管理器的规则分布流;

图 59 示出了本申请的处理器控制平面处理器/策略驱动器流;

图 60 示出了可部署在分布式安全系统中的分组过滤规则的样例;

图 61 示出了图 16 和图 17 中的 IP 处理器的 TCP/IP 处理器版本;

图 62 示出了自适应 TCP/IP 处理器;

图 63 示出了可用作图 62 中的处理器的替代处理器的自适应 TCP/IP

处理器;

图 64 示出了运行时自适应处理器;

图 65 示出了计算群集;

图 66 示出了用于提供网络安全性的安全解决方案;

图 67 示出了安全解决方案编辑器流;

图 68 示出了通过安全网卡体系结构的流;

图 69 示出了具有后援 (look-aside) 安全体系结构的网线卡;

图 70 示出了安全和内容搜索加速器适配器体系结构;

图 71 示出了安全性处理器体系结构;

图 72 示出了备选安全性处理器体系结构;

图 73 示出了第三安全性处理器体系结构;

图 74 示出了内容搜索和规则处理引擎体系结构;

图 75 示出了正则表达式规则的实例 (现有技术);

图 76a 示出了 Thompson NFA (现有技术);

图 76b 示出了 Berry-Sethi NFA (现有技术);

图 76c 示出了 DFA (现有技术);

图 77a 示出了左侧偏置 NFA 和状态转移表 (现有技术);

图 77b 示出了右侧偏置 NFA 和状态转移表 (现有技术);

图 78a 示出了状态转移控制;

图 78b 示出了每个状态的可配置下一状态表;

图 79a 示出了状态的状态转移逻辑 (STL);

图 79b 示出了状态逻辑块;

图 80 示出了基于 NFA 的搜索引擎;

图 81 示出了应用状态存储器实例配置;

图 82 示出了基于 DFA 的搜索引擎;

图 83 示出了实例 DFA 操作;

图 84 示出了运行时自适应搜索处理器;

图 85 示出了运行时自适应搜索处理器阵列;

- 图 86 示出了实例搜索群集;
- 图 87 示出了内部搜索群集规则组和切换实例;
- 图 88 示出了实例规则配置和上下文切换;
- 图 89 示出了具有内容搜索加速器的计算设备;
- 图 90 示出了实例反垃圾邮件性能瓶颈和解决方案;
- 图 91 示出了具有反病毒性能瓶颈的反垃圾邮件功能;
- 图 92 示出了应用内容搜索性能瓶颈和解决方案;
- 图 93 示出了实例内容搜索 API 使用模型;
- 图 94 示出了具有实例功能的实例内容搜索 API;
- 图 95 示出了使用搜索处理器的实例应用流 (静态设置);
- 图 96 示出了 FSA 编辑器流;
- 图 97 示出了搜索编辑器流 (完全 + 递增规则分布); 以及
- 图 98 示出了 FPGA 的 FSA 合成与编辑器流。

具体实施方式

本发明提供了可以解除主机处理器中的内容搜索性能瓶颈的新型高性能内容搜索处理器。本申请的搜索处理器用于使用基于 NFA 和 DFA 的搜索引擎并行执行大量正则表达式搜索。所述搜索处理器可以以从 1Gbps 到 10Gbps 及更高的线路速度执行高速内容搜索,而最佳服务器微处理器只能以远低于 100Mbps 的速度执行相同的任务。所述内容搜索处理器不仅可以用于执行可用于分类和安全性应用的第 2 层到第 4 层搜索,而且它还可以用于执行深度分组检查以及第 4 层到第 7 层内容分析和安全性应用。

本发明提供了以硬件实现 TCP/IP 栈的新的高性能且低等待时间方式,以消除软件 TCP/IP 栈对主机处理器的严重性能影响。然后,此硬件 TCP/IP 栈与其他处理元件连接以实现高性能且低等待时间的基于 IP 的存储应用。

此系统还实现了一种使用硬件 TCP/IP 实现以分布方式在企业网络内部实施安全性功能 (如防火墙) 的新方法,且在具有处理元件的硬件中包含适当安全性功能,以实现高性能且低等待时间的基于 IP 的网络安全性应

用。所述硬件处理器可以用于服务器、工作站、客户机 PC、笔记本电脑、手持设备、交换机、路由器以及其他联网设备的网络接口卡的内部。所述服务器可以是 Web 服务器、远程访问服务器、文件服务器、部门服务器、存储服务器、网络连接存储服务器、数据库服务器、刀片服务器、群集服务器、应用服务器、内容/介质服务器、网格计算机/服务器等。所述硬件处理器还可以用于诸如交换机或路由器或电器之类的终端系统或网络核心系统之一的 I/O 芯片组的内部。

此系统在企业网络的内部系统中实现了分布式安全性功能，例如防火墙、入侵检测、病毒扫描、虚拟专用网络、机密性服务等。可以使用本申请的硬件处理器在每个系统或它的某些关键系统中实现分布式安全性功能，而其他系统可以以软件部署这些服务。因此，在不同的系统中，整体网络将包括作为硬件实现或软件实现或其组合的分布式安全性，具体取决于 IT 管理者所确定的性能、成本和安全性需求。将从 IT 管理者使用的一个或多个中央系统管理所述分布式安全性系统，以便使用所述原理管理网络。这允许使用本申请的各种元件在网络中高效且一致地部署安全性。

这可以以各种形式实现以提供 TCP/IP 终止、高性能且低等待时间 IP 存储功能、远程 DMA (RDMA) 功能、安全性功能、可编程分类、策略处理功能、运行时自适应处理、高速内容搜索等的优点。以下是可以实现上述内容的某些实施例：

服务器

所述体系结构可以体现在高性能服务器环境中，所述环境提供基于硬件的 TCP/IP 功能或硬件 TCP/IP 和安全性或搜索功能或上述各项的组合，它们可消除一个或多个主机服务器处理器的 TCP/IP 和/或安全性和/或搜索软件开销及其性能开销。本发明的所有处理器可以是服务器芯片组的伴随处理器，提供与硬件 TCP/IP 和/或安全性和/或内容搜索的高性能联网接口。服务器可以为各种形状因数，如刀片服务器、专用 (appliance) 服务器、文件服务器、瘦服务器、群集服务器、数据库服务器、内容搜索服务器、

游戏服务器、网格计算服务器、VOIP 服务器、无线网关服务器、安全性服务器、网络连接存储服务器或传统服务器。当前实施例将允许在服务器母板上创建高性能网络接口。

此外，除了协议处理之外，所述运行时自适应协议处理器或安全性处理器或搜索处理器体系结构或它们的组合还可以用于提供其他功能或服务，如套接字层安全性、传输层安全性、加密/解密、RDMA、RDMA 安全性、应用层安全性、虚拟化或更高应用层处理，后者可以进一步包括应用级协议处理（例如，用于 HTTP、HTTPS、XML、SGML、安全 XML、其他 XML 派生物、Telnet、FTP、IP 存储、NFS、CIFS、DAFS 等的协议处理）。一个实施例可以包括使用专用协议处理器的 TCP/IP 协议处理以及映射到如本申请中所披露的运行时自适应处理器的 XML 加速。所述协议处理器可以提供也可以不提供 RDMA 功能，这取决于系统需求和支持的线路速度。本发明的安全性处理功能也可以可选地集成在此实施例中。还可以使用相同的体系结构为本申请的运行时自适应处理器上的 XML 数据处理提供安全性加速支持。

服务器芯片组的伴随处理器

除了高性能 TCP/IP 和/或 RDMA 能力之外，所述服务器环境还可以利用所述处理器的高性能 IP 存储处理能力。在此类实施例中，处理器可以是服务器芯片组的伴随服务器，除了从服务器处理器卸载 TCP/IP 之外还提供高性能网络存储 I/O 能力。此实施例将允许在母板上产生高性能的基于 IP 的网络存储 I/O。换言之，它将在母板上启用 IP SAN。

类似于上述的服务器实施例，此实施例也可以利用本申请的运行时自适应处理器以便在服务器芯片组中提供自适应硬件加速以及协议处理支持。所述运行时自适应处理器可以被配置为提供存储服务，例如虚拟化、安全性服务、多路径、协议转换等。协议转换可以包括例如光纤通道协议到 IP 存储协议的来回转换、串行 ATA 协议到 IP 存储或光纤通道协议的来回转换、串行连接 SCSI 协议到 IP 存储或光纤通道协议的来回转换等。

此外，类似于上述的服务器实施例，本申请的运行时自适应搜索处理器可以为服务器芯片组提供内容搜索加速能力。所述内容搜索处理器可以用于针对搜索规则使用正则表达式的大量应用。所述应用涵盖一系列领域，例如生物信息学、应用和网络层安全性、蛋白质组学、遗传学、Web 搜索、协议分析、XML 加速、数据仓储等。

存储系统芯片组

所述处理器还可以用作存储系统中的芯片组的伴随处理器，所述存储系统可以是在存储联网环境中执行存储数据服务器功能的存储阵列（或某些其他适合的存储系统或子系统）控制器。所述处理器将为基于 IP 的 SAN 中的网络提供存储阵列控制器的 IP 网络存储能力。配置可以类似于服务器环境中的配置，但在系统中具有其他能力以访问存储阵列以及提供其他以存储为中心的功能。

此实施例也可以利用本申请的运行时自适应处理器，以便在存储系统芯片组中提供自适应硬件加速以及协议处理支持。所述运行时自适应处理器可以被配置为提供存储服务，例如虚拟化、安全性服务、多路径、协议转换等。协议转换可以包括例如光纤通道协议到 IP 存储协议的来回转换、串行 ATA 协议到 IP 存储或光纤通道协议的来回转换、串行连接 SCSI 协议到 IP 存储或光纤通道协议的来回转换等。所述运行时自适应处理器还可以用于提供存储系统元数据处理加速以提高系统性能。

此实施例还可以利用本申请的运行时自适应搜索处理器，以便为诸如机密信息策略实施、病毒/恶意软件检测、为方便搜索的高速索引、虚拟化、基于内容的切换、安全性服务之类的各种应用提供存储业务的高速搜索。

服务器/存储主机适配器卡

所述 IP 处理器还可以嵌入提供高速 TCP/IP 联网的服务器主机适配器卡中。相同的适配器卡还能够为 IP 网络提供高速网络安全能力。同样，所述适配器卡还能够为基于 IP 的存储网络提供高速网络存储能力。所述适

配器卡可以用于传统服务器中,并且还可以用作刀片服务器配置中的刀片。所述处理器还可以用于提供基于 IP 的存储联网能力的存储阵列(或其他存储系统或子系统)前端中的适配器。所述适配器卡也可以以类似于上述的方式利用本申请的运行时自适应处理器。

处理器芯片组组件

所述 TCP/IP 处理器可以包含在处理器芯片组的内部,提供 TCP/IP 卸载能力。此类配置可以用于与高速网络连接的高端服务器、工作站或高性能个人计算机中。此类实施例还可以包括本发明的 IP 存储或 RDMA 能力或其组合,以便为基于 IP 的存储联网和/或 TCP/IP 提供嵌入在芯片组中的 RDMA 能力。对所述体系结构的多项能力的使用可以与使用此实施例或其他实施例中的其他能力无关,作为特性要求、开发时线和成本、硅片成本等的折衷。所述处理器芯片组还可以结合本申请的运行时自适应处理器,以便通过为期望应用配置处理器来按需提供可变功能集。

存储或 SAN 系统或子系统交换线卡

所述 IP 处理器还可以用于创建高性能、低等待时间 IP SAN 交换系统(或其他存储系统或子系统)线卡。所述处理器可以用作终止和发起到达/来自线卡的基于 IP 的存储业务的主处理器。此处理器将与可以充当主机的交换系统结构控制器配合使用,以便将终止的存储业务根据其 IP 目的地传输到交换系统中存在的转发信息库所确定的相应交换线卡。此类交换系统可以支持纯基于 IP 的联网或可以支持多协议支持,允许与基于 IP 的 SAN 以及其他数据中心 SAN 结构(如光纤通道)连接。非常类似的配置可以存在于网关控制器系统的内部,所述系统终止来自 LAN 或 WAN 的 IP 存储业务并发起新的会话以将存储业务运载到 SAN(其可以是基于 IP 的 SAN 或更可能是从数据中心内部如光纤通道之类的其他结构构建的 SAN)。所述处理器还可以包含在 SAN 网关控制器中。这些系统也将使用此处理器的安全性能力以在企业存储区域网络内部创建分布式安全性网络。

本申请的运行时自适应处理器在单独提供硬件加速能力或如上所述同时还提供诸如协议转换、虚拟化、安全性、带宽管理、速度限制、清理、基于策略的管理之类的功能时会非常有效。

本申请的运行时自适应处理器可以用于在此实施例中提供深度内容检查能力，此实施例可以用于创建诸如基于内容的切换、面向应用的联网之类的智能联网能力。

网络交换机、路由器、无线访问点

所述处理器还可以嵌入为交换机、路由器、网关、无线访问点等提供高速 TCP/IP 联网的网络接口线卡中。相同的适配器卡还能够为 IP 网络提供高速网络安全能力。此处理器将提供随后可以用于分布式安全性网络中的安全性能力。

本申请的运行时自适应处理器还可以用于此类实施例中，所述实施例提供上述服务和能力以及环境所需的其他有线等效保密安全性能力、RADIUS 以及类似的安全性能力。除了 TCP/IP 处理之外，所述运行时自适应处理器还可以用于提供动态可更改协议处理能力以支持无线协议，例如蓝牙、HomeRF、各种线路速度的无线以太网 LAN 协议、3GPP、GPRS、GSM 或其他无线 LAN 或 RF 或蜂窝技术协议，或它们的任意组合。

本申请的运行时自适应搜索处理器可以用于在此实施例中提供深度内容检查能力，此实施例可以用于产生诸如基于内容的切换、面向应用的联网之类的智能联网能力。

存储设备

存储网络管理成本正在迅速增长。为了能够管理网络和存储容量的显著增长，将需要创建提供存储区域管理功能的特殊设备。用于高性能基于 IP 的 SAN 的所述管理设备将实现本发明的高性能 IP 处理器，以便能够对 TCP/IP 分组内部传输的存储业务执行其功能。这些系统将需要高性能处理器以执行深度分组检查并在 IP 业务中提取存储负载以提供基于策略的管

理和实施功能。所述的安全性、可编程分类和策略引擎以及高速 TCP/IP 和 IP 存储引擎将使这些设备以及本申请中所述的其他实施例能够以高线路速度低等待时间执行深度分组检查和分类并按分组对分组应用必要的策略。此外, 这些能力可以创建能够在带内对存储区域网络业务执行其功能(例如, 虚拟化、基于策略的管理、安全性实施、访问控制、入侵检测、带宽管理、业务成形、服务质量、反垃圾邮件、病毒检测、加密、解密、LUN 屏蔽、分区、链路聚合等)的存储管理设备。在本申请中所述的其他实施例内部还可以支持类似的基于策略的管理以及安全性操作或功能。本申请的运行时自适应处理器可以用于动态支持或加速一个或多个应用/服务。可以在用户或管理员的影响或控制下通过现有策略选择所支持的应用/服务。本发明的运行时自适应搜索处理器可以用于在此实施例中执行高速网络线路速度搜索, 此实施例可以用于上述几种深度分组检查、内容搜索和安全性应用。

群集环境

在群集环境中使用服务器系统以便为群集数据库之类的应用提高系统性能和可伸缩性。在高性能群集服务器上运行的应用需要能够高速共享数据以进行进程间通信。通过传统软件 TCP/IP 网络在群集处理器之间传输此进程间通信业务将引起严重的性能开销。因此, 已在此类配置中使用诸如光纤通道之类的专用结构。但是, 可以允许在通信进程存储器之间进行直接存储器存取的基于 TCP/IP 的结构能够在不更改为诸如光纤通道之类的专用结构的情况下, 由运行于任何 TCP/IP 网络的应用使用。所述具有其高性能 TCP/IP 处理能力和 RDMA 特性的 IP 处理器可以包含在群集服务器环境中, 以便提供高性能且低等待时间的直接存储器到存储器数据传输的优点。此实施例还可以用于创建全局群集, 并且还可以用于使数据能够在网格计算机和网格网络中传输。本申请的处理器还可以用于加速使用轻量协议而不是 TCP/IP 的本地群集数据传输, 以避免等待时间和协议处理开销。可以利用所述运行时自适应处理器体系结构以支持此类轻量协议。

因此同一处理器体系结构可以用于本地以及全局群集，并且使数据能够在网格计算机和网格网络中传输。本申请的可编程处理器还可以用于类似用途而不会增加运行时自适应处理器的负担。因此本申请的处理器体系结构可以用于实现效用计算。本申请的运行时自适应处理器还可以用于为群集环境提供上面的其他实施例所述的能力。

XML 加速器

本申请的运行时自适应 TCP/IP 处理器还可以用作系统或适配器卡内部的组件或用作提供 TCP/IP 协议终止或 XML 加速或其组合的芯片组的一部分。随着 Web 服务使用的增加，越来越多的 Web 文档可能开始使用 XML 或 XML 派生协议。针对每个网页访问处理 XML 可能会为主机处理器带来非常沉重的负担，从而需要额外的硬件支持。本申请的运行时自适应处理器可以用于此类环境中以提供 XML 处理加速，而传输协议处理由本申请的专用协议处理器处理。XML 文档也可能需要安全性支持，在此情况下，所述处理器可以被动态配置为提供安全 XML 文档的安全性加速。同样，本发明的运行时自适应搜索处理器还可以用于此类实施例中，并且提供高速 XML 处理能力以满足 1Gbps、10Gbps 及更高的线路速度。本发明的搜索处理器允许针对此应用同时计算数千个正则表达式规则，并且因此还可以用于联网系统中以提供智能联网和面向应用的联网能力。

基于 IP 的话音 (VOIP) 设备

本申请的处理器还可以嵌入诸如 VOIP 电话、服务器、网关、手持设备之类的基于 IP 的话音设备内。所述协议处理器可以用于在 VOIP 环境中根据需要提供 IP 协议处理以及传输层协议处理。此外，所述运行时自适应处理器可以动态地改变，以提供 VOIP 设备和运行于此设备上的应用可能需要的信号处理和 DSP 硬件加速能力。除了类似于上述的 VOIP 应用之外，本申请的运行时自适应搜索处理器可以用于语音识别应用。

手持设备

本申请的处理器还可以用于为手持设备、电话、个人数字助理等的处理器或芯片组提供协议处理硬件能力。所述协议处理器以及运行时自适应处理器可以为许多实施例提供上述的多种能力。本申请的处理器可以用于在这些设备内部创建安全协议处理栈以及提供其他使用硬件加速的服务。所述运行时自适应处理器可以用于以有线或无线方式使所述手持设备联网。然后所述设备通过使用本申请的协议处理器和运行时自适应处理器，可以动态地改变以与多种协议共同工作，所述协议如蓝牙、无线以太网 LAN、RF、GPRS、GSM、CDMA、CMDA 变型或其他 3G 蜂窝技术或其他无线或蜂窝或 RF 技术。本申请的运行时自适应搜索处理器还可以包含在手持设备中以加速已经低性能的处理器搜索性能。本申请的搜索处理器可以结合在所述手持设备的芯片组或处理器中，以加速运行于这些平台上的应用的搜索性能。

所述搜索处理器还可以用作提供密码安全性、网络层安全性功能到应用层安全性的多用途安全性处理器，此处理器可以包含在芯片组或主机处理器中或作为所述手持设备中的伴随处理器，以使各种手持设备免于出现的安全性威胁。

其他实施例

所述处理器体系结构可以部分以软件实现而部分以硬件实现。性能需求和所涉成本可以驱使对本发明的总体系统体系结构的硬件和软件划分进行折衷。还可以将此体系结构作为芯片组以及硬件和软件划分的组合实现或独立于划分而实现。例如，所述安全性处理器和分类引擎可以位于不同的芯片上并提供类似的功能。这会导致较低的 IP 处理器硅成本（包括开发和制造成本），但是在某些情况下可能增加系统中的元件数并可能增加占用面积或总的解决方案成本。安全性和分类引擎也可以是不同的芯片。如在此所使用的，芯片组可以表示多芯片的芯片组或仅包括单个芯片的芯片组，具体取决于应用。

存储流控制器和队列可以在主机上的软件中维护或可以成为芯片组中其他芯片的一部分。因此，此体系结构可以存在多种划分方法，以实现未来高性能处理器所需的基于 IP 的高性能存储和 TCP/IP 卸载应用。根据 iSCSI 给出了存储引擎说明，但是对于 TCP/IP 和存储引擎可编程性、分类器可编程性和存储流控制器以及控制处理器，可以使用适合的固件实现诸如 iFCP、FCIP 之类的其他 IP 存储协议。iSCSI 操作还可以表示 IP 存储操作。高性能 IP 处理器核心可以与多个较低线路速度的输入输出端口相连，从而匹配总吞吐量以同时创建多端口 IP 处理器实施例。

可以将此体系结构用于主处理器中的高性能 TCP/IP 卸载而不使用存储引擎。这为用于数据和电信应用的下一代高性能网络产生了硅和系统解决方案。所述 TCP/IP 引擎可以使用应用特定的分组加速器来加强并利用核心体系结构得到此处理器的新特征。可以使用诸如防火墙引擎或路由查找引擎或电信/网络加速引擎之类的其他应用特定的加速器以及本发明的其他能力来更改存储引擎，并将此处理器体系结构的目标定为电信/联网和其他应用。

本发明的运行时自适应搜索处理器还可以包含在目标为安全性或内容搜索应用等的各种应用特定的系统的内部。例如，所述搜索处理器可以包含在电子邮件安全性设备的内部，此设备提供诸如反垃圾邮件、反病毒、反间谍软件、反蠕虫病毒、其他恶意软件防御，以及机密信息安全性和其他用于萨班斯·奥克斯利法案（Sarbanes-Oxley）、格雷姆-里奇-比利雷法案（Gramm-Leach-Bliley）、健康保险便利及责任法案（HIPAA）等规范的规范符合性之类的能力。所述搜索处理器还可以包含在可用于外泄检测的系统和防护系统中，以在企业边缘防止机密信息泄漏以及提供其他如上面列出的规范符合性。

详细说明

能够执行内容搜索已经成为网络化世界中的一项关键能力。随着网络线路速度达到 1Gbps、10Gbps 以及更高，必须能够为许多应用以线路速度

执行深度分组检查。如果能够以高线路速度检查网络业务，则可以检测若干安全性问题（如病毒、蠕虫、机密信息泄漏等）并防止其引起破坏。通常，可以使用正则表达式来表示内容搜索规则。可以使用 FSA 表示和计算正则表达式规则。NFA 和 DFA 是两种类型的用于计算正则表达式的 FSA。对于高线路速度应用可以使用复合 DFA，其中可以按存储器访问周期处理输入流的每个字符。但是，这在存储器访问速度所规定的可以执行搜索的最大速度方面存在限制。此类方法的另一个限制是搜索甚至适度数量的正则表达式规则所需的存储器量。如上所述，NFA 在通用处理器上实现高性能方面也具有其限制。通常，当今的最佳微处理器针对少量正则表达式使用 NFA 或 DFA 只能实现低于 100Mbps 的性能。因此，显然需要创建目标内容搜索处理器硬件以将搜索性能提高到 1Gbps 和 10Gbps 的线路速度。本发明示出了目标可定为高线路速度的此类高性能内容搜索处理器。

在过去几年中，存储成本和需求已经迅速增长。预期在可预见的未来还将以相同的速度增长。随着电子商务的出现，无论服务器或系统何时出现故障数据都能随时随地可用是至关重要的。这强烈需要将服务器连接的存储装置移动到网络中以提供存储联合、数据可用性和数据易管理性。存储区域网络（SAN）现在主要基于光纤通道技术，与 TCP/IP 技术相比，此技术通过其面向硬件的栈提供如低等待时间和高性能之类的各种优点。

某些系统在设计为传输数据流的 IP 上传输块存储业务。使用被分层以在 IP 顶部运行的传输控制协议（TCP）传输数据流。TCP/IP 是在操作系统内以软件实现的面向连接的可靠协议。TCP/IP 软件栈处理将在未来部署的高线路速度时非常缓慢。新的硬件解决方案将加速 TCP/IP 栈以承载存储和网络业务并与基于 FC 的解决方案相媲美。

高性能服务器、工作站以及存储控制器和阵列中最常用的存储协议是已经出现约 20 年的 SCSI 协议。SCSI 体系结构被构建为分层的协议体系结构。图 1 示出了发起方（块 101）和目标子系统（块 102）内的各种 SCSI 体系结构层。如本申请中所使用的，术语“发起方”和“目标”表示数据处理装置或包括它们的子系统或系统。术语“发起方”和“目标”还可以

表示客户机或服务器或对等设备。同样，术语“对等设备”可以表示对等数据处理装置或其子系统或系统。“远程对等设备”可以是位于世界各地或房间各处的对等设备。

图 1 中的发起方和目标子系统使用 SCSI 应用协议层（块 103）来彼此交互，块 103 用于提供客户机-服务器请求和响应事务。它还在发起方和目标大容量存储设备（其可以采取多种形式，如盘阵列、磁带驱动器等）之间提供设备服务请求和响应。传统上，目标和发起方使用承载 SCSI 协议（块 104）的 SCSI 总线体系结构互连。SCSI 协议层是允许客户机和服务器使用 SCSI 应用协议彼此交互的传输层。传输层必须向上层呈现相同的语义，以便上层协议和应用可以保持与传输协议无关。

图 2 示出了基于 IP 的传输层顶部的 SCSI 应用层。IETF 标准跟踪协议 iSCSI（IP 上的 SCSI）尝试提供基于 IP 的存储传输协议。存在其他类似尝试，包括 FCIP（封装在 IP 中的 FC）、iFCP（IP 上的 FC）等。这些协议中的多个协议以类似图 2 中所示的方式位于作为传输机制的 TCP/IP 的顶部。如图 2 所示，iSCSI 协议服务层（块 204）提供到 SCSI 应用层（块 203）的分层接口。如标准所定义的，iSCSI 作为 iSCSI 协议数据单元（PDU）承载 SCSI 命令和数据。然后可以使用 TCP/IP（块 205）等通过网络传输这些协议数据单元。所述标准没有规定实现承载 iSCSI PDU 的基础传输的方式。图 2 示出了位于提供 iSCSI PDU 传输的 TCP/IP 上的 iSCSI。

基于 IP 的存储协议（如 iSCSI）可以以软件形式位于基于软件的 TCP/IP 栈的顶部。但是，此类实施方式将产生由软件 TCP/IP 以及位于其顶部的存储协议引起的严重性能恶化。此类实施方式将严重影响主机处理器的性能并且可能使处理器在线路速度超过 1Gbps 时无法用于任何其他任务。因此，将以硬件实现 TCP/IP 栈，减轻可在其上建立存储协议的主机处理器的负担。存储协议（如 iSCSI）可以以运行于主机处理器上的软件建立，或者可以如本申请中所述那样使用硬件实现来加速。软件 iSCSI 栈将多次中断主机处理器以从接收的 TCP 段中提取能够对其执行操作的

PDU。此类实施方式将由于类似于基于软件的 TCP 栈的原因而产生严重的性能恶化。所述处理器提供了高性能且低等待时间的体系结构，以在基于 TCP/IP 的网络上传输存储协议，此体系结构消除或显著降低了主机处理器中的性能恶化以及产生的等待时间影响。

图 3 示出了参考 OSI 联网栈的 TCP/IP 栈与光纤通道的比较。如先前在本申请的“发明内容”部分中所述，TCP/IP 栈（块 303）具有由主机上的软件实现产生的性能问题。与其相比，诸如光纤通道（块 304）之类的专用联网协议被设计为以硬件实现。硬件实现允许联网解决方案比基于 IP 的解决方案具有更高的性能。但是，IP 的普遍存在性以及 IT 用户和开发人员对 IP 的熟悉性使得 IP 更适合于广泛部署。如果由 TCP/IP 产生的性能恶化被降低到等同于其他竞争专用协议产生的性能恶化，便可以实现这一点。图 4 示出了在用于 TCP/IP（块 403）的硬件和软件中分层以便与其他所示专用协议相媲美的协议级别。

图 5 示出了使用本申请的基于硬件的 TCP/IP 和存储协议实现的主机操作系统栈。实现协议以便可以将其引入主机操作系统栈（块 513），以使它上面的操作系统层不变。这允许 SCSI 应用协议在不发生任何更改的情况下工作。驱动器层（块 515）以及基于 IP 的存储接口下面的栈（块 501）将表示与非联网 SCSI 接口（块 506 和 503）或光纤通道接口（块 502）类似的接口。

图 6 示出了软件 TCP/IP 栈中涉及的数据传输。此类 TCP/IP 栈实施方式由于数据传输的存储器复制而产生极大的性能恶化。此图示出了客户机和服务器联网栈之间的数据传输。需要在客户机和服务器之间来回传输的用户级应用缓冲区（块 601）经过所示的数据传输的各个级别。源处的用户应用缓冲区被复制到 OS 内核空间缓冲区（块 602）。然后此数据被复制到网络驱动器缓冲区（块 603），数据从块 603 中被 DMA 传输到网络接口卡（NIC）或主机总线适配器（HBA）缓冲区（块 604）。缓冲区复制操作涉及主机处理器并耗尽宝贵的处理器周期。此外，要传输的数据在主机上经过校验和计算，这将用尽主机的其他计算周期。在主机上数据多次

进出系统存储器还将产生存储器带宽瓶颈。传输到 NIC/HBA 的数据然后继续被发送到网络（块 609）并到达目的地系统。在目的地系统处，数据分组通过后面类似的缓冲区复制和校验和运算以与主机相反的方向经过软件联网栈。对于可能在源和目的地之间传输大量数据的块存储数据传输和群集应用而言，此类 TCP/IP 栈实施方式的效率非常低。

图 7 示出了发起方和目标中允许本申请中所述体系结构的远程直接存储器存取（RDMA）功能的联网栈。以下内容可以称为 RDMA 能力或 RDMA 机制或 RDMA 功能。在此类系统中，运行于发起方或目标上的应用注册存储器区域（块 702），该区域可供其对等方（多个）使用以便可直接从 NIC/HBA 存取而无需大量主机干预。这些应用还使其对等方（多个）知道可用于 RDMA（块 708）的存储器区域。当通信的对等双方准备好使用 RDMA 机制后，如果对等方中的 NIC/HBA 硬件实现 RDMA 能力，则 RDMA 区域中的数据可以在实际上没有复制开销的情况下从源传输到目的地，而无需大量主机干预。源（或发起方）将通知其对等方它需要读取或写入已启用 RDMA 的特定缓冲区，然后使目的地（或目标）将数据推入其 RDMA 缓冲区或从其中拉出数据。发起方和目标 NIC/HBA 然后使用本申请中所述的 TCP/IP 硬件实现 RDMA 703、TCP/IP 卸载 704、RDMA 708 和 TCP/IP 卸载 709 相互传输数据而无需主机处理器大量干预，从而显著降低处理器开销。此机制将显著降低主机处理器上的 TCP/IP 处理开销并且不需要针对图 6 中所示的数据传输进行多次缓冲区复制。因此，已启用 RDMA 的系统将允许系统（无论较快还是较慢）执行数据传输而不会对其对等方产生性能瓶颈。在基于 IP 的存储解决方案中，在此处理器中实现的 RDMA 能力消除了主机干预（通常除了数据传输开始和终止以外）。这减轻了目标和发起方系统中的主机处理器的负担，以便执行有用任务而无需在每个分组到达或传输时被中断。RDMA 实现还使系统变得安全并防止未授权访问。这通过使用区域的存取控制密钥以及区域 ID 向 HBA/NIC 注册输出存储器区域来实现。HBA/NIC 执行从远程主机到 RDMA 缓冲区的存储器区域请求的地址转换、执行诸如安全性密钥验证之类的安全性操

作，然后允许数据传输。例如，此处理在位于 HBA/NIC 上或作为母板上主机处理器的伴随处理器的本发明处理器中的主机处理器之外执行。此能力还可以用于服务器群集应用以及客户机服务器应用的大型数据传输。在源或发起方和目的地或目标之间传输大量数据的实时媒体应用可从中受益。

图 8 示出了以软件实现的主机文件系统和 SCSI 栈。如先前所指出的，基于 IP 的存储栈（块 805、806、807、808 和 809）应对 SCSI 层（块 803 和 804）表示一致的接口，如 SCSI 传输层（块 811）或光纤通道传输（块 810）所提供的接口。除了由 IP（其未被设计为传输性能敏感的块数据）的各种问题施加的要求，此图还示出了从系统级别施加于基于 IP 的存储实现的高级要求。

图 9 根据图 8 中所示更详细示出了 iSCSI 栈。iSCSI 栈（块 805 到 809）应向 SCSI 命令联合层（块 803 和 804）提供 OS 定义的驱动器接口级功能，以便此层及其顶部的其他层的行为不变。图 9 示出了为提供 IP 存储能力而实现的一组功能。提供 iSCSI 功能的各功能被分组成相关的功能组，但如本领域的技术人员所理解的，可以存在许多变型。需要一组功能以满足标准（例如目标和发起方登录和注销）功能（块 916）、连接建立和关闭功能（块 905）。此图示出了允许 OS SCSI 软件栈发现 iSCSI 设备（块 916），设置和获取选项/参数（块 903 和 909）以启动设备（块 913）和释放设备（块 911）的功能。除了先前所述的控制功能之外，iSCSI 实现通过队列 912 和 917 提供大容量数据传输功能，以传输 iSCSI 标准所指定的 PDU。iSCSI 栈还可以包括直接数据传输/放置（DDT）或 RDMA 功能或它们的组合（块 918），发起方和目标系统使用块 918 执行实际上不进行缓冲区复制且主机不干预的数据传输，包括存储和其他大容量块数据传输。SCSI 命令以及与这些命令相关的块数据传输作为在所述处理器上执行的命令队列（块 912 和 917）实现。主机主要在命令完成时被中断。已完成的命令进行排队以便主机在其方便时对这些命令执行操作。此图示出了位于 TCP/IP 栈（块 907 和 908）上的 iSCSI 协议层和驱动器层，TCP/IP 栈也

在此处所述的 IP 处理器系统上的主机处理器之外实现。

图 10 示出了在所述 IP 处理器系统中实现的 TCP/IP 栈功能。这些功能提供到上层协议功能的接口以承载 IP 存储业务以及其他可受益于直接 OS TCP/IP 旁路、RDMA 或网络套接字直接能力或其组合的应用，以利用此处理器的高性能 TCP/IP 实现。TCP/IP 栈提供能力以发送和接收上层数据(块 1017 和 1031)以及命令 PDU, 建立传输连接和关闭功能(块 1021), 发送和接收数据传输功能、校验和功能(块 1019)、错误处理功能(块 1022), 以及分段和排序和窗口化操作(块 1023)。诸如校验和验证/创建之类的特定功能接触数据传输的每个字节, 而对于数据传输的每个分组, 将调用传输数据分组和更新传输控制块或会话数据库的某些功能。会话数据库(块 1025)用于维护与活动会话/连接相关的各种信息以及 TCP/IP 状态信息。在提供如标准所需的 IP 功能的 IP 层顶部建立 TCP 层。此层提供功能以按路径 MTU 将分组分段/取消分段(块 1033)、提供路由和转发信息(块 1032), 以及提供到传送错误(例如 ICMP, 块 1029)等所需的其他功能的接口。IP 层与以太网层或其他介质访问层技术连接以在网络上传输 TCP/IP 分组。低层在本说明书的各个图中被示为以太网, 但可以是例如 SONET 之类的其他技术, 以通过 MAN/WAN 上的 SONET 传输分组。以太网还可以用于类似的应用中, 但可能在例如 LAN 和专用本地 SAN 环境内用的更多。

图 11 示出了 iSCSI 数据流。此图示出了数据流的接收和发送路径。与 iSCSI 驱动器配合使用的主机 SCSI 命令层(同时在块 1101 中示出)在图 26 中更详细示出的存储流控制器中, 将要处理的命令调度到命令调度器(块 1108)。命令调度器 1108 调度新命令以便在图 17 中更详细描述的处理器的运行。用于具有现有连接的目标设备的新命令被入队到此现有连接(块 1111)中。当与目标设备的连接不存在时, 新命令被入队到未分配的命令队列(块 1102)。然后调用如图 47 以及块 905 和 1006 中所示的会话/连接建立进程以连接到目标。如图所示, 建立连接后, 队列 1102 中的相应命令由命令调度器 1108 入队到新创建的连接命令队列 1111。命令到达

执行级后，将激活接收 1107 或发送 1109 路径，具体取决于命令是读取还是写入事务。传输命令的连接/会话的状态用于在会话数据库中记录命令执行进度，如随后所述那样。与数据传输关联的缓冲区可以被锁定直到传输完成。如果使用 RDMA 机制在发起方和目标之间传输数据，则相应的区域缓冲区标识符、存取控制密钥和相关 RDMA 状态数据在处理器板上的存储器中维护并且也可以在片外存储器中维护，这具体取决于所选择的实施方式。当与命令关联的数据传输（可能在多个 TCP 段上）完成时，命令执行的状态被传送到主机 SCSI 层，此层然后进行相应的处理。这可能包括释放用于将数据传输到应用的缓冲区、统计信息更新等。在传输期间，iSCSI PDU 由与传输命令引擎（块 1110）配合使用的传输引擎（块 1109）传输，该传输引擎解释 PDU 并执行相应的操作，如使用 DMA 将应用缓冲区从主机存储器检索到存储处理器并保持 iSCSI 连接数据库中的存储命令流信息随进度更新。如本申请中所使用的，术语“引擎”可以是适用于引擎功能或使用的数据处理器或数据处理器的一部分。同样，接收引擎（块 1107）将接收的命令解释成需要相应执行操作的新请求、响应、错误或其他命令或数据 PDU。与命令引擎（块 1106）配合使用的这些接收引擎通过直接数据传输/放置或为 iSCSI 会话表中的会话维护的 RDMA 控制信息，将所读取的数据或所接收的数据路由到相应的已分配应用缓冲区。当命令完成时，将释放对各个缓冲区（块 1103 和 1112）的控制以供应用使用。接收和发送引擎可以是与会话数据库表项 1704 中记录的会话信息配合使用的此 IP 处理器的图 17 的 SAN 分组处理器 1706 (a) 到 1706 (n)，从图 23 的 TCP/IP 处理器或图 24 的 IP 处理器的角度，该表项可以被视为全局存储器。可以为不同的分组和命令重复使用同一引擎，所述分组和命令具有由下面针对图 17 的块 1704 和 1708 中的会话数据库部分更详细讨论的会话数据库所提供的适当存储流上下文。为了清晰，术语 IP 网络应用处理器、IP 存储处理器、IP 存储网络应用处理器和 IP 处理器可以是同一实体，这具体取决于应用。IP 网络应用处理器核心或 IP 存储网络应用处理器核心可以是同一实体，这具体取决于应用。

同样，控制命令可以使用发送路径而所接收的响应将使用接收路径。发起方以及目标上可以存在类似的引擎。取决于发起方还是目标，数据流方向将有所不同。但是，发起方和目标上存在基本类似的数据流，且在目标处具有其他步骤。目标需要执行其他操作以保留例如获取写入命令的数据所需的缓冲区，或者可能需要在将数据提供给发起方之前准备读取数据。在中间设备中将存在类似的情况，但在可以是交换机或设备的此类装置中，可以执行某种级别的虚拟化或帧过滤或此类其他操作，这些操作可能需要在一端终止会话而在另一端发起会话。此体系结构支持此功能，但未在图中显式示出，因为它为本领域中的技术人员所公知。

图 12 到图 15 示出了有关传输会话的特定信息以及可如何在存储器的数据库中存储此信息。

图 12 示出了为 iSCSI 协议和关联的 TCP/IP 连接维护的数据结构。通过相应的连接（块 1202）承载属于每个 iSCSI 会话（块 1201，实际上是发起方和目标连接的联系）的数据。在同一连接的队列中调度相关命令以保持命令（块 1203）的顺序。但是，可以将无关命令分配给不同的传输连接。如果所述实施方式针对每个会话仅支持一个连接，则可以将所有命令排队到同一连接。但是，每个会话的多个连接可以支持发起方和目标之间的线路中继。例如，在某些应用中，发起方和目标将相互通信并通过协商确定接受多个连接。在其他应用中，发起方和目标将仅通过一个会话或连接通信。图 13 和图 14 示出了每个会话和连接的 TCP/IP 和 iSCSI 会话数据库或传输控制块。这些表项可以作为单独的表承载或可以一起作为复合的表承载，如随后针对图 23、24、26 和 29 所示的那样，具体取决于所选择的实施方式以及所实现的功能（例如仅 TCP/IP、TCP/IP 及 RDMA、仅 IP 存储、IP 存储及 TCP/IP、IP 存储及 RDMA 等）。执行 TCP/IP 和存储流控制的各种引擎使用这些字段的全部或部分或更多未示出的字段，以指导通过 TCP/IP 进行块数据传输。在数据传输过程中，在连接经历多种状态时更新相应的字段。图 15 示出了一种在包括片上会话高速缓存（块 1501 和 1502）以及片外会话存储器（块 1503、1504、1505、1506 和 1507）的

存储子系统中存储传输控制表项的方法，此子系统保留数据传输的一系列进度所需的状态信息。

图 16 更抽象地示出了 IP 处理器体系结构。所述处理器包括模块化且可伸缩的 IP 网络应用处理器核心（块 1603）。其功能块提供了用于通过 IP 网络启用高速存储和数据传输的功能。所述处理器核心可以包括智能流控制器、可编程分类引擎和存储/网络策略引擎。每个元件都可以被视为单独的处理器或者它们的任何组合都可以实现为单个处理器。所披露的处理器还包括安全性处理块以便为网络分组提供高线路速度加密和解密功能。同样，这可以是单个处理器，或与上面提及的其他元件组合。所披露的处理器包括存储器子系统（包括管理片上会话高速缓存/存储器的存储器控制器接口），以及管理对片外存储器的访问的存储器控制器（块 1602），所述片外存储器可以是 SRAM、DRAM、FLASH、ROM、EEPROM、DDR SDRAM、RDRAM、FCRAM、QDR SRAM 或静态或动态随机存取存储器的其他派生物或其组合。所述 IP 处理器包括相应的系统接口以允许将其用于目标市场细分，从而为 LAN、SAN、WAN 和 MAN 网络及类似网络提供正确的介质接口（块 1601），以及适合的主机接口（块 1606）。所述介质接口块和主机接口块可以采用多端口形式，其中某些端口可以在使用所披露的处理器和网络和系统中提供冗余和故障转移功能。所述处理器还可以包含协处理器接口（块 1605）以便扩展主处理器的能力，例如，创建多处理器系统。系统控制器接口（块 1604）允许此处理器与现成微控制器（该控制器可用作其中可以使用所披露处理器的系统的系统控制器）连接。所述处理器体系结构还支持可以用作系统控制器或会话管理器的板上控制平面处理器。还提供所述系统控制器接口以便能够使用外部处理器。此版本的处理器可能由于芯片成本原因而不包括控制处理器。可以针对特定系统要求（例如，服务器适配器或存储控制器或交换机线卡或其他联网系统）创建各种类型的核心体系结构。主要区别已在本申请的先前部分进行了讨论。这些处理器块提供了能力和性能以使用诸如 iSCSI、FCIP、iFCP 之类的标准协议实现基于 IP 的高性能存储。这些块的详细体系结构将在以下说

明中进行讨论。

图 17 更详细地示出了 IP 处理器体系结构。所述体系结构提供能力以便通过完全 TCP/IP 终止和深度分组检查来处理来自介质访问控制(MAC)层或其他相应层的入站 IP 分组。此块图没有示出 MAC 层(块 1601)或图 16 的块 1602、1604 或 1605。MAC 层将块连接到图 16 中所示介质接口(块 1601)中的处理器的输入队列(块 1701)以及输出队列(块 1712)。MAC 功能可以基于与网络相关的特定类型的标准。以太网和 SONET 上的分组是当今最常用的接口实例,这些接口可以包括在同一硅片或针对每个接口创建的不同处理器版本中。

图 17 中的块图将输入队列和输出队列(块 1701 和 1712)示为两个单独的块。可以使用组合的块提供所述功能。输入队列(块 1701)包括逻辑、控制和存储以便从 MAC 接口块中检索入站分组。块 1701 与分组调度器 1702 和分类引擎 1703 配合使用,在分组从接口到达时对分组进行排队并创建相应标记以标识分组开始、分组结束以及其他如分段分组或安全分组之类的属性。分组调度器 1702 可以从输入队列控制器中检索分组并将它们传送到分类引擎以便分类。分类(块 1703)示为在调度器之后,但从逻辑观点看,分类引擎从输入队列接收分组,对分组进行分类并为分组提供分类标记,然后调度器将分组调度到处理器阵列 1706(a) ... 1706(n)。因此,分类引擎可以用作通透(pass-through)分类引擎,以全线路速度维持通过其结构的分组流。分类引擎是可编程引擎,对从网络接收的各种类别的分组进行分类并使用分类结果标记分组以供调度器和其他分组处理器使用。网络业务分类是计算非常密集的活动,可以占用分组处理器中的一半可用处理器周期。此集成分类引擎是可编程的,以执行第 2 层到第 7 层检查。要分类的字段使用与它们关联的比较和操作(如果存在匹配)的预期值进行编程。分类器收集分类遍历(walk)结果并可以将这些结果作为标记提供给分组,如随后针对图 30 所示的那样标识分类结果。这很像树结构并被理解为“遍历”。然后将已分类的分组提供给调度器 1702 以作为处理流水线的下一阶段。

分组调度器（块 1702）包括将分组分配给所披露的处理器上的相应执行引擎的状态控制器和定序器。执行引擎是 SAN 分组处理器（块 1706（a）到 1706（n）），包括 TCP/IP 和/或存储引擎以及存储流/RDMA 控制器（块 1708）或主机旁路和/或其他相应处理器，具体取决于所需的实施方式。为了简洁，术语“/”在本申请中用于指定硬件组件时，可以在适用情况下表示“和/或”。例如，组件“存储流/RDMA 控制器”根据实施方式可以是存储流和 RDMA 控制器、存储流控制器，或 RDMA 控制器。调度器 1702 还维护通过处理器的分组顺序，其中同一连接/会话中的分组间的状态相关性对于正确处理进站分组很重要。调度器维护各种表以跟踪调度的分组通过处理器的进度直到分组退役。调度器还从主机处理器或交换机结构控制器或接口的出站命令和分组中接收需要被调度到分组处理器的命令。

在图 17 中，TCP/IP 和存储引擎连同可编程分组处理器一起被标记为 SAN 分组处理器 1706（a）到 1706（n）。这些分组处理器是作为独立可编程实体并充当特定角色的引擎。备选地，可以将它们中的两个或更多个实现为单个处理器，具体取决于所需的实施方式。在此实例中，将图 23 的 TCP/IP 引擎和图 24 的存储引擎配置为图 21 的可编程分组处理器引擎（块 2101）的协处理器。因此，由于成本、可制造性、市场细分等原因，此体系结构可以通过代替/移除存储引擎而相对轻松地应用于存储之外的应用。在纯联网环境中，可以移除存储引擎，剩下具有专用 TCP/IP 引擎并应用于联网业务的分组处理器，其将面对与 TCP/IP 软件栈相同的处理开销。备选地，可以为所需实施方式移除一个或多个引擎，例如对于仅支持 IP 存储功能的处理器，可以移除可能位于单独芯片中的 TCP/IP 引擎和/或分组引擎。因此，核心可伸缩且模块化的体系结构可以存在多种变型。因此，可以在除基于 IP 的存储应用之外的应用中通过以其他专用引擎（例如，高性能网络安全性和策略引擎、高性能路由引擎、高性能网络管理引擎、提供字符串搜索的深度分组检查引擎、用于 XML 的引擎、用于虚拟化的引擎等）代替存储引擎来利用所述核心体系结构，以提供应用特定的加速支持。可以通过调整芯片中的 SAN 分组处理器（块 1706（a）到 1706

(n)) 数来调整此 IP 处理器的处理能力, 以满足网络接口的线路速度要求。可伸缩性的主要限制将来自所需硅片面积以及由硅工艺技术施加的限制。基本上, 此体系结构通过增加多个 SAN 分组处理器块可扩展到非常高的线路速度, 从而提高处理能力。获得类似结果的其他方式是将处理器的工作时钟频率增加到处理器技术限制内的可行频率。

图 17 还示出了 IP 会话高速缓存/存储器和存储器控制器 (块 1704)。该高速缓存可以被视为内部存储器或本地会话数据库高速缓存。该块用于在特定数目的活动会话期间缓存和存储 TCP/IP 会话数据库以及存储会话数据库。可缓存的会话数直接取决于所选择的硅片面积以及在经济上可以制造的产品。不在片上的会话通过可以是块 1704 的一部分或其他元件的高性能存储器控制器块存储或检索到被视为外部存储器的片外存储器上, 并从所述片外存储器存储或检索到片上。该处理器的各个处理元件均使用高速内部总线共享该控制器以存储和检索会话信息。在分组是分段的、主机接口或出站队列已满时, 所述存储器控制器还可用于临时存储分组。所述控制器还可用于存储由所披露的处理器或在所披露的处理器或主机处理器上运行的应用收集的统计信息或任何其他信息。

图 17 的处理器块图还示出了主机接口 (块 1710), 主机输入队列 (块 1707)、主机输出队列 (块 1709) 以及存储流/RDMA 控制器 (块 1708)。这些块提供了向/从主机 (也称为“对等设备”) 存储器或交换结构传输数据所需的功能。这些块还提供了允许基于主机的驱动器调度命令、检索入站状态、检索会话数据库表项、编程所披露的处理器以及执行类似操作的功能, 从而启用如套接字直连架构、完全 TCP/IP 终止、IP 存储卸载之类的能力以及类似使用或不使用 RDMA 的能力。在图 27 中详细示出的主机接口控制器 1710 提供了配置寄存器、用于直接存储器到存储器数据传输的 DMA 引擎、执行上述某些任务的主机命令块、以及主机接口事务控制器和主机中断控制器。主机输入和输出队列 1707 和 1709 提供入站和出站分组的排队。存储流和 RDMA 控制器 (块 1708) 提供主机将命令排队到所披露的处理器所需的功能, 所述处理器然后接受并执行这些命令, 在命令

结束时中断主机处理器。块 1708 的 RDMA 控制器部分提供了启用远程直接存储器存取所需的各种能力。它具有包括如 RDMA 区域、存取密钥之类的信息以及虚拟地址转换功能的表。该块中的 RDMA 引擎执行数据传输并解释收到的 RDMA 命令以便执行允许的事务。块 1708 的存储流控制器还跟踪当目标和发起方之间发生数据传输时已调度的各种命令的进度状态。存储流控制器调度要执行的命令并且还向主机驱动器提供命令完成信息。上述内容可以被视为 RDMA 能力并且可以按照所述实现或通过实施为单个处理器来实现，具体取决于设计者的选择。同样，可以在不偏离本申请的精神或范围的情况下将其他功能添加到所述功能中或从所述功能中删除其他功能。

该处理器的控制平面处理器（块 1711）用于为 TCP/IP 和/或存储协议提供相对较慢的路径功能，所述存储协议可以包括带有 ICMP 协议的错误处理、名称解析、地址解析协议，并且其还可以编程为用作会话控制器/连接管理器以执行会话发起/关闭，执行登录和参数交换等。该控制平面处理器可以在片外以为系统开发人员提供对控制平面处理器的选择，也可以在片上以提供集成解决方案。如果所述控制平面处理器位于片外，则可以在此处创建或集成一个接口块以允许该处理器与控制平面处理器进行交互并执行数据和命令传输。出于性能、芯片成本要求之类的因素，内部总线结构和功能块互连可以与所示的所有详图有所不同并且不偏离本申请的精神和范围。

上述图 17 中的各个块的功能将在下面详细地介绍，这些功能实现了允许分组借助所披露处理器的各种处理资源的流内处理，以最小的等待时间从输入流向输出的分组流体系结构。

图 18 更详细地示出了在图 17 的 1701 处概括示出的输入队列和控制器块。该块的核心功能是在块 1801 以及 1802 (i) 到 1802 (n) 中接受来自多个输入端口（端口 1 到 N）的入站分组，并使用输入分组队列（块 1810）的固定或可编程优先级对这些分组进行排队，在所述输入分组队列处，针对分类器、调度器使所述分组出队并通过调度器 1/F（块 1807-1814）进行

进一步分组处理。输入队列控制器与每个输入端口（在多端口实施方式中为端口 1 到 N）进行交互，并将分组排队到输入分组队列 1810。分组入队控制器和标记（块 1804）可以提供固定优先级功能或可以是可编程的，以允许根据诸如端口速度、端口的网络接口、端口优先级以及其他适当特性之类的各种特性将不同的策略应用于不同的接口。可编程各种优先级模式，如循环法、加权循环法等。输入分组出队控制器 1812 使分组出队并通过调度器 1/F 1814 将它们提供给分组调度器（图 17 中的块 1702）。一旦图 17 中的分类引擎 1703 将分组进行了分类，所述调度器就将分组调度到 SAN 分组处理器 1706 (a) -1706 (n)。如果实施方式包括安全处理，则加密分组可以首先被分类为加密类型，然后由图 18 的安全分组接口（块 1813）传送到图 17 的安全引擎 1705 以进行验证和/或解密，否则可以不存在安全接口并使用外部安全性处理器执行类似的功能。来自明文分组接口（块 1811）的解密分组然后通过块 1812 被提供给输入队列，从块 1812，分组沿着与明文分组相同的路线传送。分段 IP 分组可以存储在片上的分段分组存储和控制器缓冲区（块 1806）中，也可以存储在内部或外部存储器中。当最后一个分段到达时，与图 17 的分类引擎和调度器协作的分段控制器（块 1806）将这些分段进行合并以组装成完整分组。一旦将分段分组组合成完整的分组，就通过块 1804 将分组调度到输入分组队列，然后由分组出队控制器（块 1812）对所述分组进行处理以使其传递到此处理器的其他各个处理级。图 18 的输入队列控制器将由属性管理器（块 1809）管理的分组标记/描述符分配给每个入站分组，所述属性管理器使用分组描述符字段（如分组开始、大小、缓冲区地址等）以及任何其他来自分类引擎的安全信息，并将所述标记/描述符存储在分组属性和标记阵列（块 1808）中。所述分组标记和属性用于通过调度器和处理器的其他元件，以有效的方式经由接口 1807、1811、1813、1814 控制通过处理器的分组流。

图 19 更详细地示出了图 17 中的分组调度器和定序器 1702。该块负责将分组和任务调度到该处理器的执行资源并同时充当负载平衡器。调度器通过输入队列控制器 1901 检索标头队列（块 1902）中的分组标头并将它

们传送到图 17 中的分类引擎 1703, 分类引擎 1703 然后将分类结果返回到由其他处理器引擎使用的分类器队列 (块 1909)。分类引擎可以主要包括标头, 但是如果还编程了深度分组检查, 则分类引擎可以接收其在分类之后路由到调度器的完整分组。调度器包括分类控制器/调度器 (块 1908), 后者管理通过分类引擎的分组的执行。在分段分组或安全分组的情况下, 图 19 中的块 1908 将命令提供给输入队列控制器 (块 1901) 以为此类分组执行适当的操作, 例如将加密分组调度到图 17 的安全引擎。调度器状态控制和定序器 (块 1916) 接收处理器内各种活动的事务/操作的状态信息并为下一组操作提供指令。例如, 调度器从输入分组队列 (块 1903) 检索分组, 然后根据从分类器接收的分类结果将这些分组调度到适当的资源队列中, 或者通过 1906 将分组导向分组存储器 (块 1913 或 1704), 从而当调度时或调度后, 适当资源执行其操作需要可用于检索分组的分组描述符/标记时, 创建这些描述符/标记。状态控制和定序器 (块 1916) 使用分类结果 (块 1914) 指示/指导将分组存储在分组存储器 (块 1913) 中, 可以在调度分组进行操作时从该分组存储器检索分组。状态控制器和定序器标识应接收分组以进行操作的执行资源、创建命令并且将此带有分组标记的命令分配给图 19 的资源队列 (块 1917, 控制平面)、1918 (端口 i 到端口 n)、1919 (旁路) 和 1920 (主机)。优先级选择器 1921 是根据分配的优先级从各个队列检索命令和分组标记并将其传递到分组取回和命令控制器 (块 1922) 的可编程块。该块从分组存储库 1913 检索分组连同分类结果, 并在诸如 1926 之类的高性能处理器命令和分组总线上的适当资源可以执行操作时将分组传输调度到该资源。各接收方的总线接口块 (如命令总线接口控制器 1905) 解释命令并接受分组和分类标记以便操作。这些执行引擎在分组操作完成时以及将分组调度到最终目的地 (主机总线接口或输出接口或控制平面接口等) 时通知调度器。这允许调度器在退役引擎 (块 1904) 的帮助下使分组退出其状态并释放资源分配表 (块 1923) 中该会话的资源表项。定序器使用资源分配表将收到的分组分配给特定的资源, 具体取决于这些资源的内部状态的当前状态, 例如在 SAN 分组处理器引擎中缓冲的

会话数据库高速缓存表项、正在资源中执行的当前分组的连接 ID 等。因此，依赖于顺序执行的分组主要分配给同一资源，这通过使用处理器的会话存储器中的当前数据库状态并且不必检索新的会话表项而改进了存储器业务和性能。定序器还具有到存储器控制器（块 1906）的接口，以便对分段分组进行排队和/或处理由于下游的分组处理瓶颈而导致调度器队列已满的情况（所述瓶颈由特定应用引起，该应用执行分组的时间长于为了维持全部线路速度性能而分配的时间），或者处理任何其他下游系统已满而无法维持线路速度的情况。

如果按照上面参考图 17 所述在调度器之前实现分类器，其中分类引擎接收来自输入队列的分组，则 1901、1902、1908、1909 和 1910 等各项将位于分类器中，或者不需要这些项，具体取决于特定的设计。可以在此情况下创建分类器与调度器（块 1903、1907、1914 和 1915）之间的适当双向连接并且分类器直接连接到图 18 的输入队列块。

图 20 示出了在图 17 的图 1703 处概括示出的分组分类引擎。将分组分类成它们的各种属性是一项计算密集型操作。分类器可以是可编程处理器，其检查收到的分组的各字段以标识分组类型、协议类型（如 IP、ICMP、TCP、UDP 等）、端口地址、源和目的地字段等。分类器可用于测试标头或有效负载中的某个特定字段或一组字段。块图示出了基于可按内容寻址的存储器的分类器。但是，如前所述，它也可以是可编程处理器。主要区别在于引擎的性能和实现复杂性。分类器通过调度器从输入队列（图 20 中的块 2005 和 2004）获取输入分组。输入缓冲区 2004 将需要分类的分组/描述符和/或分组标头进行排队。然后分类定序器 2003 取回队列中的下一可用分组并根据可以或能够被编程的全局字段描述符集（块 2007）提取相应的分组字段。然后，分类器将这些字段传递到可按内容寻址的存储器（CAM）阵列（块 2009）以执行分类。当字段通过 CAM 阵列时，这些字段的匹配将标识下一组要比较的字段并潜在地标识它们的位字段单元。CAM 阵列中的匹配将导致操作/事件标记，后者由结果编辑器（块 2014）收集（其中“编辑”的意思为“收集”）并且作为需要更新与特定 CAM

条件或规则匹配关联的存储器阵列（块 2013）中的数据的操作而执行。这可以包括在该字段上执行可以视为一个执行资源实例的算术逻辑单元（ALU）运算（块 2017），例如递增或递减条件匹配等。CAM 阵列使用字段、字段期望值以及匹配操作进行编程，其中包括下一要比较的字段，所述字段通过数据库初始化（块 2011），并且可以访问以便通过主机或控制平面处理器接口 1710、1711 进行编程。一旦分类达到叶节点，所述分类便完成并且生成分类标记，所述分类标记标识已遍历并在随后由 IP 处理器的其他引擎使用的路径以避免执行相同的分类任务。例如，分类标记可以包括流或会话 ID、协议类型指示（如 TCP/UDP/ICMP）、值（指示是处理、忽略、丢弃分组还是丢弃会话等），也可以包括特定固件代码例程指针以便执行资源开始分组处理，或者可以包括所遍历的分类路径的签名等。根据处理器实施方式和功能来选择分类标记字段。分类器退役队列（块 2015）保存已分类的分组/分组描述符和分类标记并等待调度器进行检索。可以使用数据库扩展接口和流水线控制逻辑（块 2006）扩展分类数据库。这允许构建针对更大的分类数据库进行扩展的系统。具有操作解释器的分类引擎、ALU 和范围匹配（块 2012）还提供编程存储/网络策略/操作（在满足特定策略时需要采取该操作）的能力。可以以规则和操作表的形式实施策略。在分类引擎中通过主机接口和分类表对策略进行编辑和编程。数据库接口和流水线控制 2006 可以实现为与伴随处理器相连以扩展分类/策略引擎的大小。

图 21 示出了在图 17 的 1706 (a) 到 1706 (n) 处概括示出的 SAN 分组处理器。分组处理器可以是专门设计的分组处理器，也可以是诸如 ARM、ARC、Tensilica、MIPS、StrongARM、X86、PowerPC、Pentium 处理器、iA64 之类的任何适合的处理器或任何其他提供此处所述功能的处理器。在本申请的各部分中，所述分组处理器也称为分组处理器复合体。该分组处理器包括分组引擎（块 2101）（通常为带有分组处理目标指令的 RISC OR VLIW 机器）或 TCP/IP 引擎（块 2102）或 IP 存储引擎（块 2103）或上述引擎的组合。这些引擎可以配置为分组引擎的协处理器或者可以是独立

的引擎。图 22 更详细地示出了分组引擎。该分组引擎通常为如上所述的 RISC OR VLIW 机器，其具有都可以是 RAM 并用于保存分组处理微例程和分组及提供中间存储的指令存储器（块 2202）和数据存储器（块 2206）。指令存储器 2202，如本申请中的所有此类存储器那样，可以是 RAM 或其他适合的存储装置，并使用在分组处理期间执行的代码进行初始化。分组处理代码被组织为恰好放入所分配存储器中的微例程。指令解码器和定序器（块 2204）从指令存储器 2202 中取回指令，将这些指令解码并通过 ALU（块 2208）中包含的执行块对指令进行排序。该机器可以是简单的流水线引擎，或是也可以设计为提供面向分组的指令集的更复杂的深度流水线机器。DMA 引擎（块 2205）和总线控制器（块 2201）允许分组引擎将数据分组从图19的调度器和主机接口移动到数据存储器 2206 中以便进行操作。DMA 引擎可以保存多个存储器描述符以便将分组/数据存储/检索到主机存储器/分组存储器或从主机存储器/分组存储器存储/检索分组/数据。这允许存储器存取与分组处理器引擎操作同时发生。DMA 引擎 2205 还用于将数据分组移动到 TCP 引擎 2210 和存储引擎 2211 以及从所述引擎中移动数据分组。一旦分组执行完成，就将所提取的数据或新生成的分组传输到输出接口以便向介质接口或主机接口发送。

图 23 更详细地示出了在图 22 的 2210 处概括示出的可编程 TCP/IP 分组处理器引擎。该引擎通常是一个带有常见 RISC OR VLIW 指令与各种面向 TCP/IP 的指令以及执行引擎的可编程处理器，但也可以是带有本申请所述的相应执行引擎的微代码或状态机驱动型处理器。TCP 处理器包括校验和（块 2311），用于通过在处理器上执行指令来执行 TCP 校验和验证以及生成新的校验和。校验和块从分组缓冲存储器（数据 RAM 是此类存储器的一个实例）2309 提取数据分组并执行校验和生成或验证。分组查找接口（块 2310）协助执行引擎和指令定序器 2305，提供对各个数据分组字段或完整数据分组的访问。如果选择了此类实施方式，则指令解码器 2304 使用分类标记解释器 2313 来根据分类结果引导程序流。处理器提供包括分段在内的特定定序和窗口操作（块 2315）以用于例如 TCP/IP 数据

定序计算,以便查找下一个预期的序列号并查看接收的内容是否位于约定的滑动窗口内,所述滑动窗口是 TCP 协议的公知部分并用于分组所属的连接。该元件 2315 还可以包括如图 24 的 2413 处显示的分段控制器。备选地,本领域的技术人员借助本申请的教导可以在图 23 的 TCP/IP 处理器上的其他位置容易地实现分段控制器。处理器提供散列引擎(块 2317),所述散列引擎用于针对分组的特定字段执行散列运算,以执行获取正确的分组会话表项所需的散列表遍历。处理器还包括寄存器文件(块 2316),其提取各种常用的标头字段进行 TCP 处理,包括数据源和目的地的指针寄存器、上下文寄存器集以及保存 TCP 状态和通用寄存器文件的寄存器。TCP/IP 处理器可具有多个用于分组执行的上下文,以便当给定分组执行因诸如存储器存取之类的任何原因停止时,可唤醒其他上下文并且处理器在损失较少的效率的情况下继续执行另一分组流。TCP/IP 处理器引擎还维护本地会话高速缓存(块 2320),该高速缓存保存最近使用或最常使用的表项,所述表项无需从全局会话存储器中进行检索即可在本地使用。本地会话高速缓存可以视为 TCP/IP 处理器(其可以是分组处理器)的内部存储器。当然,在不从会话或全局或存储器检索其他表项的情况下,存储在内部存储器本地并要使用的表项越多,处理的效率就越高。将通知图 19 的分组调度器根据每个 TCP/IP 处理器资源所缓存的连接 ID,以便将属于同一会话的分组调度到同一分组处理器复合体。当分组处理器未保存特定连接的会话表项时,与会话管理器(块 2321)和散列引擎协作的 TCP 会话数据库查找引擎(块 2319)将通过存储器控制器接口(块 2323)从全局会话存储器检索相应的表项。诸如会话管理器内允许访问会话表项或会话表项字段的逻辑电路之类的装置与散列引擎协作,以生成用于将相应的会话表项或其字段存储/检索到会话数据库高速缓存的会话标识符。作为分组处理的结果,这可用于更新那些字段或表项。当取回新表项时,正在替换的表项被存储到全局会话存储器。本地会话高速缓存可以遵循独占缓存原则,以便多个处理器复合体不会引起任何竞争条件,从而破坏会话状态。还可以使用诸如 MESI 之类的其他缓存协议获得类似的结果。当会话表项缓存在某

个处理器复合体中时,但是另一处理器复合体需要该表项,则会根据算法,使用独占访问或适当的缓存状态将该表项传送到新的处理器。在特定的情况下,该会话表项还可以被写入全局会话存储器。TCP/IP 处理器还包括 TCP 状态机(块 2322),该状态机用于遍历正在处理的连接的 TCP 状态。该状态机接收在会话表项中存储的状态信息以及影响新收到的分组的状态的相应字段。这允许状态机在存在状态转移时生成下一状态,并且在会话表表项中更新信息。TCP/IP 处理器还包括用于提取帧信息并执行乱序分组执行操作的帧控制器/乱序管理器(块 2318)。该块还可以包括如图 24 的 2417 处所示的 RDMA 机制,但是用于非存储数据传输。本领域的技术人员,借助本申请的教导,也可以在 TCP/IP 处理器上的其他位置实现 RDMA 机制。该体系结构创建可以使用分组 CRC 作为帧密钥或其他密钥的上层成帧机制,所述密钥由可编程帧控制器使用,以当分组甚至以乱序到达时提取嵌入的 PDU 并允许引导这些分组到达最终缓冲目的地。该单元与会话数据库交互来处理乱序到达信息,所述信息将被记录以便在中间分段到达时避免重新传输。一旦通过 TCP/IP 处理器处理了分组,便将所述分组传递到存储引擎进行处理,如果分组属于存储数据传输并且特定实施方式包括存储引擎的话,否则所述分组将被继续传递到主机处理器接口或存储流/RDMA 控制器(块 1708)以便进行处理,并通过 DMA 传送到最终缓冲目的地。分组还可以传输到分组处理器块以执行对分组的任何其他处理。这可以包括在由 TCP/IP 处理器和存储处理器处理之前或之后,可以对分组执行的应用与客户特定的应用代码。从主机到输出介质接口的数据传输还将通过 TCP/IP 处理器以形成要在数据周围创建的适当标头,同时执行适当的数据分段,所述分段操作还与帧控制器和/或存储处理器协作以更新会话状态。可以作为主机命令或调度器调度到分组处理器进行处理的已接收网络分组的结果来检索此数据。出于性能、芯片成本要求等原因,内部总线结构和功能块互连可以与所示的有所不同。例如,主机控制器接口 2301、调度器接口 2307 和存储器控制器接口 2323 可以是总线控制器的一部分,所述总线控制器允许向/从调度器或存储流/RDMA 控制器或主机或

会话控制器或其他资源传输数据分组或状态信息或命令或它们的组合，所述资源包括但不限于安全性处理器或介质接口单元、主机接口、调度器、分类处理器、分组缓冲区或控制器处理器或它们的任意组合。

图 24 更详细地示出了图 22 的 IP 存储处理器引擎。该存储引擎是可编程引擎，带有与基于 IP 的存储配合的指令集以及常见的类似 RISC OR VLIW 的分组处理指令集。IP 存储处理器引擎包含块 2411 以执行 CRC 运算。该块允许执行 CRC 生成和验证。带有 IP 存储的进站分组将从 TCP/IP 引擎通过 DMA（块 2402 和 2408）进站数据存储（数据 RAM 是此类存储器的一个实例）（块 2409）。当实施方式不包括 TCP/IP 引擎或分组处理器引擎或它们的组合时，可以例如直接从调度器接收分组。可以根据需要从本地会话高速缓存中检索与连接相关的 TCP 会话数据库信息，也可以从 TCP/IP 引擎与分组一起接收该信息。将存储 PDU 提供给 PDU 分类器引擎（块 2418），该引擎将 PDU 分为相应的命令，然后使用所述命令调用相应的存储命令执行引擎（块 2412）。可以使用 RISC OR VLIM 或等价指令集或使用专用硬件引擎完成命令执行。命令执行引擎执行在 PDU 中收到的命令。收到的 PDU 可以包含读取命令数据或未决写入命令的 R2T 或 IP 存储协议所需的其他命令。这些引擎从主机接口检索写入数据或将读取数据导向目的地缓冲区。将存储会话数据库表项缓存在可以视为本地存储器的数据库高速缓存（块 2420），其对处理器所处理的最新或常用连接而言是本地的。命令执行引擎执行命令并与存储状态机（块 2422）及会话管理器（块 2421）协作更新存储数据库表项。连接 ID 用于标识会话，如果会话不在高速缓存中，则由存储会话查找引擎（块 2419）从图 17 的全局会话存储器 1704 进行检索。对于从发起方到目标的数据传输，处理器按照诸如路径 MTU 之类的各种网络约束使用分段控制器（块 2413）将数据单元分为多个段。分段处理器尝试确保出站 PDU 的大小最适合于连接。如果所请求的数据传输大于最大有效分段大小，则分段控制器将数据打包为多个分组并与序列管理器（块 2415）协作以正确地分配序列号。分段控制器 2413 也可以在图 23 的 TCP/IP 处理器内实现。也就是说，当该处理器

用于 TCP/IP 操作而非存储操作时,分段控制器可以是图 23 的排序/窗口化操作管理器 2315 的一部分。使用本申请的教导,本领域的技术人员可以容易地提出在 TCP/IP 处理器中包括分段控制器的备选实施例。图 24 的存储处理器(或图 23 的 TCP/IP 处理器)还可以包括 RDMA 引擎,该引擎为使用此 RDMA 机制实现的存储或网络数据传输解释在 PDU 中收到的远程直接存储器存取指令。例如,在图 24 中此引擎是 RDMA 引擎 2417。在图 23 的 TCP/IP 处理器中, RDMA 引擎可以是帧控制器和乱序管理器 2318 或其他适合组件的一部分。如果连接的两端都认可数据传输的 RDMA 模式,则无需主机的大量介入,便可使用 RDMA 引擎调度目标和发起方之间的数据传输。在会话数据库表项中维护 RDMA 传输状态。该块创建要置于数据周围的 RDMA 标头,并且还用于从在启用 RDMA 的连接上收到的分组中提取这些标头。RDMA 引擎通过传递消息/指令与存储流/RDMA 控制器 1708 以及主机接口控制器 1710 协作并执行大块数据传输而无需主机的大量介入。IP 处理器的存储流/RDMA 控制器(块 1708)的 RDMA 引擎对所请求的操作执行保护检查并同时提供从 RDMA 区域标识符到主机空间中物理地址或虚拟地址的转换。此功能也可以由 SAN 分组处理器的存储引擎的 RDMA 引擎(块 2417)根据所选择的实施方式来提供。2417 和 1708 以及其他类似引擎之间的 RDMA 能力分配是一种实施方式选择,本领域的技术人员能够使用本申请的教导实现此选择。出站数据由 PDU 创建器(块 2425)打包为基于标准的 PDU。还可以使用分组处理指令来完成 PDU 格式设置。图 24 的存储引擎与图 23 的 TCP/IP 处理器以及图 17 的分组处理器引擎协作来执行 IP 存储操作,所述操作包括双向数据和命令传输,即发起方到目标和目标到主机以及相反的方向。也就是说,主机控制器接口 2401、2407 将命令或数据或命令与数据的组合存储到主机处理器,或从主机处理器检索命令或数据或命令与数据的组合。这些接口可以直接连接到主机,也可以通过中间连接与主机相连。尽管示为两个装置,但是接口 2401 和 2407 可以实现为单个装置。通过这些块的数据流根据传输方向的不同可以有所不同。例如,当命令或数据被从主机发送到目标时,将

首先调用存储处理引擎以设置 PDU 的格式，然后将该 PDU 继续传送到 TCP 处理器以便将 PDU 打包到有效的 TCP/IP 段中。但是，收到的分组在调度到存储处理器引擎之前将通过 TCP/IP 引擎。出于性能、芯片成本要求等原因，内部总线结构和功能块互连可以与所示的有所不同。例如，与图 23 类似，主机控制器接口 2401、2407 和存储器控制器接口 2423 可以是总线控制器的一部分，所述控制器允许向/从调度器或主机或存储流/RDMA 控制器或会话控制器或其他资源传输数据分组或状态信息或命令或它们的组合，所述资源包括但不限于安全性处理器或介质接口单元、主机接口、调度器、分类处理器、分组缓冲区或控制器处理器或它们的任意组合。

在通过（作为一个实例）诸如图 24 的 iSCSI 处理器之类的 IP 存储处理器执行片上存储的应用中（所述芯片不包括图 23 的 TCP/IP 处理器），TCP/IP 接口 2406 将充当调度器接口来通过 IP 存储处理器调度 IP 存储分组处理。在了解本申请的披露后，本领域的技术人员可做出类似的变型。

图 25 更详细地示出了图 17 的输出队列控制器（块 1712）。该块接收需要发送到图 16 的网络介质无关接口 1601 的分组。分组可以附有标记以指示它们是否需要在发送之前进行加密。控制器通过队列 2511 和安全引擎接口 2510 将要保护的分组排队到安全引擎。从安全引擎接收加密后的分组并将它们排队到块 2509 中以便发送到它们的目的地。输出队列控制器可以将分组分配到它们各自的服务质量（QOS）队列，如果支持此类机制的话。可编程分组优先级选择器（块 2504）选择下一个要发送的分组并针对相应的端口（端口 1 到端口 N）调度所述分组。与端口关联的介质控制器（块 1601）接受分组并将它们发送到其目的地。

图 26 更详细地示出了在图 17 的 1708 处概括示出的存储流控制器/RDMA 控制器块。存储流和 RDMA 控制器块提供主机将命令（存储或 RDMA 或套接字直连或它们的组合）排队到处理器所需的功能，所述处理器接受这些指令并执行它们，主要在命令终止时中断主机处理器。新的和活动的命令队列（块 2611 和 2610）以及完成队列（块 2612）可以部分位

于芯片上,部分位于主机存储区域或与IP处理器关联的存储器中,可以从这些位置取回命令以及存储完成状态。RDMA引擎(块2602)提供启用远程直接存储器存取所需的各种能力。它具有类似RDMA查找表2608的表,这些表包括诸如RDMA区域和存取密钥之类的信息以及虚拟地址转换功能。该块2602中的RDMA引擎执行数据传输并解释收到的RDMA命令以执行允许的事务。存储流控制器还在目标和发起方之间发生数据传输时跟踪已调度的各种命令的进度状态。存储流控制器调度要执行的命令,同时还向主机驱动器提供命令完成信息。存储流控制器提供存储来自主机的新请求的命令队列,并将活动命令保存在活动命令队列中。命令调度器(块2601)将收到的针对没有连接存在的目标的新命令分配给调度器来发起新的连接。调度器1702使用在图17的1711处概括示出的控制平面处理器建立连接,此时连接表项将移动到图15和图17的1704处概括示出的会话高速缓存中,并且存储流控制器(块2601)中的状态控制器将新命令移到活动命令中并将所述命令与相应的连接进行关联。块2610中的活动命令被检索并发送到调度器(块1702)以便由分组处理器进行操作。将对命令状态的更新提供回流控制器,所述流控制器然后将更新存储在命令状态表(块2607)中并通过块2603进行访问。定序器2601应用可编程优先级进行命令调度并从活动命令和新命令中选择下一要调度的命令。流控制器还包括用于入站命令的新请求队列(块2613)。一旦主机驱动器在主机上完成相应的处理和缓冲区保留,就将新请求传输到活动命令队列。在调度要执行的命令时,状态控制器2601发起由主机数据预取管理器(块2617)使用主机接口(块2707)的DMA引擎从主机存储器进行的数据预取,从而在执行命令时使数据准备好提供给分组处理器复合体。输出队列控制器(块2616)与主机控制器接口(块2614)协作以启用数据传输。存储流/RDMA控制器维护目标-发起方表(块2609),所述表将已解析的目标/发起方和已建立的连接相关联,以便快速查找以及将命令关联到活动连接。如果被执行的命令是RDMA命令,或者如果协商在发起连接时通过RDMA机制完成存储传输,则命令定序器还与RDMA引擎2602协作。上面所述

的 RDMA 引擎 2602 提供接受多个 RDMA 区域、存取控制密钥以及虚拟地址转换指针的功能。主机应用（可以是用户应用或 OS 内核功能、诸如下载网页、视频文件之类的存储或非存储应用）使用所披露的处理器通过关联主机驱动器所提供的服务注册要在 RDMA 事务中使用的存储器区域。完成此操作之后，主机应用将此信息传送给其远端的对等设备。现在，远程机器或主机可以执行由两端的 RDMA 块提供的 RDMA 命令而无需主机的大量介入。RDMA 传输可以包括从区域读取带有特定偏移的特定字节数或具有类似属性的写入之类的操作。RDMA 机制还包括发送功能，当在两个端节点之间创建通信管道时此功能非常有用。这些特性在群集应用中非常有用，在所述群集应用中，在群集内的服务器上运行的两个应用的缓冲区之间，或更有可能在两个不同的服务器群集内的各服务器之间，或此类其他群集系统之间需要进行大量数据传输。存储数据传输还可以使用 RDMA 机制完成，因为它允许在不需要主机大量介入的情况下进行大块数据传输。两端的主机初始地参与约定进行 RDMA 传输，并通过共享的存取控制密钥分配存储区域和权限。然后两个节点之间的数据传输就可以继续而无需主机处理器介入，只要两端的节点维护可用的缓冲区空间以及缓冲区传输信任即可。存储数据传输协议将通过同意使用 RDMA 协议并在两端启用该协议而在 RDMA 之上运行。图 26 的存储流控制器和 RDMA 控制器然后可以使用 RDMA 命令执行存储命令和数据传输。当预期的数据传输完成后，使用完成队列 2612 将存储命令完成状态传送给主机。由图 17 的分组处理器复合体处理来自网络的进站数据分组，然后在存储/RDMA 数据分组的情况下，将提取 PDU 并将其提供给图 26 的流控制器。然后将它们分配给进站队列（块 2604），并通过查找接收缓冲区的存储器描述符并使用主机接口（块 2707）内的 DMA 引擎执行 DMA 来传输到目的地缓冲区。RDMA 命令还可以根据 RDMA 初始化经历保护密钥查找和地址转换。

上述内容还可以视为 RDMA 能力或 RDMA 机制或 RDMA 功能的一部分。

图 27 更详细地示出了图 17 的主机接口控制器 1710。主机接口块包括主机总线接口控制器（块 2709），该控制器提供到主机总线的物理接口。当在交换机或网关或类似的配置中实施时，主机接口块可以实现为结构接口或介质无关接口，具体取决于系统体系结构，并且还可以提供虚拟输出排队和/或其他服务质量特性。事务控制器部分（块 2708）执行各种总线事务并维护它们的状态，接收所请求的事务以便完成。主机命令单元（块 2710）包括主机总线配置寄存器和一个或多个命令解释器以执行主机所传递的命令。主机驱动器通过主机输出队列接口 2703 将这些命令提供给该处理器。这些命令提供诸如设置配置寄存器、调度 DMA 传输、设置 DMA 区域和权限（如果需要），建立会话表项，检索会话数据库，配置 RDMA 引擎之类的各种功能。还可以使用该接口传输存储命令和其他命令以供 IP 处理器执行。

图 28 更详细地示出了图 17 的安全引擎 1705。所示的安全引擎提供了 IPSEC 等标准所需的验证、加密和解密服务。安全引擎所提供的服务可以包括多种验证和安全算法。安全引擎可以位于处理器之上，也可以如此前所述是独立硅芯片的一部分。提供 IP 安全服务的外部安全引擎将位于数据流中的类似位置，作为入站分组的分组处理的最初级之一以及出站分组的最后级之一。所示的安全引擎提供基于高级加密标准（AES）的加密和解密服务，这些服务是作为安全标准采用的非常能发挥硬件性能的算法。该块还可以提供诸如 DES、3DES 之类的其他安全能力。所支持的安全与验证算法和功能由芯片成本和开发成本决定。所选的算法还将是 IP 存储标准所需的算法。所示的验证引擎（块 2803）包括 SHA-1 算法作为一个可用的算法实例。该块提供 IP 安全标准中规定的消息摘要和验证能力。当需要安全和消息验证服务时，数据将流过这些块。去往目标的明文分组在途中被加密，然后使用适当的引擎对其进行所需的验证。所接收的安全分组以相反的顺序经历相同的步骤。将对安全分组进行验证，然后使用该块的引擎 2803、2804 对其进行解密。安全引擎还在安全上下文存储器（块 2809）中维护针对连接建立的安全关联。安全关联（可以包括安全会话索引、安

全密钥、所用的算法、会话的当前状态等)用于执行消息验证和加密/解密服务。可以彼此独立地使用消息验证服务和加密/解密服务。图 28 的安全引擎或图 20 的分类/策略引擎或其组合连同本申请的其他协议处理硬件能力将使用本申请的硬件处理器创建安全的 TCP/IP 栈。

图 29 更详细地示出了在图 17 的 1704 处概括示出的会话高速缓存和存储器控制器复合体。存储器复合体包括在本申请中称为会话/全局会话存储器或会话高速缓存的 TCP/IP 会话数据库的高速缓存/存储器体系结构,所述体系结构实现为高速缓存或存储器或两者的组合。会话高速缓存查找引擎(块 2904)提供查找特定会话高速缓存表项的功能。该查找块根据所提供的字段创建散列索引或可以接受散列密钥并查找会话高速缓存表项。如果在高速缓存阵列中没有与散列索引的标记匹配,查找块将使用该密钥从外部存储器中查找会话表项,并使用该会话表项替换当前的会话高速缓存表项。它向请求分组处理器复合体提供会话表项字段。位于本地处理器复合体高速缓存中的高速缓存表项在全局高速缓存中被标记为共享。因此当任何处理器请求该高速缓存表项时,它被转移到全局高速缓存和请求处理器中并在全局高速缓存中进行相应的标记。会话存储器控制器还负责将驱逐的本地会话高速缓存表项移到该块内的全局高速缓存中。这样对于会话表项,在任何时刻,只有最新的会话状态可用于请求方。如果会话高速缓存已满,则新的表项可以导致将最近最少使用的表项驱逐到外部存储器。会话存储器可以是单路或多路高速缓存,也可以是散列索引存储器或它们组合,具体取决于给定工艺技术上可用的硅片面积。使用高速缓存存储会话数据库表项的独特之处在于,在用于网络交换机或路由器的联网应用中,分组之间通常不存在太多的局部引用特性,因此由于高速缓存未命中,使用高速缓存可能无法提供太大的性能改进。但是,存储事务是两个末端系统之间的长时间事务并且可以交换大量数据。在两个节点之间发生大量数据传输的情况下,例如在群集或媒体服务器之类的环境中,基于高速缓存的会话存储器体系结构将通过减少来自片外存储器的大量数据传输而显著提高性能。会话高速缓存的大小是可用硅片面积的函数并且根据折衷对性

能具有影响。存储器控制器块还其他需要在存储器中存储分组、分组段或任何其他操作数据的块提供服务。存储器接口提供单个或多个外部存储器控制器（块 2901），具体取决于需要支持的预期数据带宽。它可以是双倍数据速率控制器或 DRAM 或 SRAM 或 RDRAM 或其他动态或静态 RAM 或它们组合的控制器。图中示出了多个控制器，但是控制器数量将根据所需的带宽和成本而变化。存储器复合体还提供计时器功能以用于会话的重传超时，所述会话将在会话数据库存储器块维护的重传队列中进行排队。

图 30 示出了分类引擎的数据结构详细信息。这是组织分类引擎的数据结构的一种方法。分类数据库示为树结构（块 3001），并且在树中具有节点（块 3003），同时还示出了与这些节点关联的操作（块 3008），所述操作允许分类引擎遍历树结构来比较特定的节点值。节点值和节点所表示的字段是可编程的。当操作字段与特定的节点值匹配时，便会提取该字段。操作项定义下一步骤，所述步骤可以包括提取和比较新字段，执行其他操作，例如对与该节点-值对关联的特定数据字段执行 ALU 操作，或者可以指示终端节点，特定分组的分类将在该点处完成。分类引擎使用此数据结构对从分组调度器接收的分组进行分类。在重复分组的不同字段时，使用值匹配检索到的操作项由结果编辑器用于创建分类标记，所述标记附加到分组，通常位于分组标头之前。处理器的其他部分然后使用分组标记作为参考来根据分类结果决定需要采取的操作。具有自己的可编程特征的分类器允许分类树结构进行系统内更改并允许处理器在具有不同分类需求的系统中使用。分类引擎还允许创建存储/网络策略，后者可以编程为分类树-节点-值-操作结构的一部分并在基于 IP 的存储系统中提供非常强大的能力。所述策略将增强对使用该处理器的系统的管理并允许在满足或违反特定策略或规则时执行强制操作的能力。当特定的系统约束需要时，分组引擎允许通过外部组件扩展分类数据库。根据硅片面积和性能折衷来决定树和节点的数量。数据结构元素在分类引擎的各个块中维护并且由分类定序器用于引导分组分类通过结构。分类数据结构所需的字段可能多于或少于

所示的字段，具体取决于目标解决方案。因此可以在不偏离基本体系结构的情况下使用较少的组件和结构实现分类的核心功能。分类过程将如编程的那样遍历树和节点。特定节点操作可以使新的树用于对其余字段进行分类。因此，分类过程从树的根开始并逐步遍历各个节点，直至到达叶节点。

图 31 示出了发起方和目标之间的读取操作。发起方将 READ 命令请求（块 3101）发送到目标以启动事务。这是一个应用层请求，该请求被映射到特定的 SCSI 协议命令，所述协议命令然后作为 READ 协议数据单元（块 3102）在基于 IP 的存储网络中传输。目标准备所请求的数据（块 3103）并提供读取响应 PDU（块 3105），所述 PDU 将进行分段以满足最大传输单元限制。发起方然后从 IP 分组检索数据（块 3106），然后将数据存储在为该操作分配的读取缓冲区中。所有数据传输之后，目标使用命令完成和识别（sense）状态（块 3107）进行响应。一旦完成全部传输（块 3109），发起方则使命令退出。如果目标处存在任何错误且命令因任何原因终止，发起方将单独启动恢复程序。该事务是标准的 SCSI READ 事务，其中数据通过诸如 iSCSI 之类的基于 IP 的存储协议作为该协议的 PDU 进行传输。

图 32 针对图 31 所示的事务的已接收 READ PDU 之一示出了本发明中的 IP 处理器内的数据流。其中示出了由发起端上 IP 处理器所接收的读取数据 PDU 的内部数据流。该图示出了分组所经历各个操作阶段。可以将这些阶段视为分组所遍历的各个流水线级。所遍历的管道级数取决于接收的分组的类型。图中示出了通过所建立的连接接收的分组所通过的管道级。分组将经过以下主要的管道级：

1. 接收管道级（块 3201），其中主要步骤在块 3207 中示出：分组由介质访问控制器接收。然后对分组进行检测，删除报头和尾部并提取具有第二层标头和有效负载的分组。将在这一阶段针对目标接收方以及任何错误检测执行第二层验证。可以根据所建立的策略应用服务质量检查。一旦分组验证完成，将分组排队到输入队列。

2. 安全管道级（块 3202），其中主要步骤在块 3208 中示出。分组被从输入队列移到分类引擎，将在该引擎中快速确定安全性处理，并且如

果分组需要进行安全性处理，则进入安全管道级。如果接收的分组为明文分组并且无需验证，则可以跳过安全管道级。如果安全引擎未与 IP 处理器集成，则也可以忽略安全管道级。分组将通过安全引擎的各个级，其中首先从存储器中检索该连接的安全性关联，然后使用所选的消息验证算法对分组进行验证。接着使用为会话建立的安全密钥对分组进行解密。一旦分组为明文形式，便可重新排队到输入队列控制器。

3. 分类管道级（块 3203），其中主要步骤在块 3209 中示出。调度器从输入队列检索明文分组并调度要分类的分组。分类引擎执行各种任务，如针对第三层及更高层分类从分组提取相关字段，识别 TCP/IP/存储协议等，并创建这些分类标记，还可以根据在分类引擎中编程的策略执行诸如拒绝分组或标记要忽略的分组之类的操作。分类引擎还可以使用会话或其所属的流对分组进行标记，同时标记分组标头和分组有效负载以便于提取。所列的某些任务可以执行，也可以不执行，并且可以执行其他任务，具体取决于分类引擎的编程。完成分类之后，将分类标记添加到分组并对分组进行排队以供调度器处理。

4. 调度管道级（块 3204），其中主要步骤在块 3210 中示出。将从分类引擎队列检索分类后的分组并将其存储在调度器中进行处理。如果分类器未对分组标头中的源字段和目的地字段执行散列运算以标识分组所属的流，则调度器将执行此操作。一旦完成流识别，将根据流相关性将分组分配给执行资源队列。当资源可用于接受新的分组时，将分配队列中的下一分组以便由该资源进行处理。

5. 执行管道级（块 3205），其中主要步骤在块 3211 中示出。当执行分组的资源可用时，该分组将进入执行管道级。将分组传输到预定执行分组的分组处理器复合体。处理器将查看附加到分组的分类标记以确定该分组所需的处理步骤。如果这个一个基于 IP 的存储分组，则检索该会话的会话数据库表项。如果本地会话高速缓存中已经保存了该会话表项，则不需要访问数据库。如果根据流执行了分组分配，则不需要从全局会话存储器中检索会话表项。分组处理器然后启动 TCP 引擎/存储引擎来执行其操

作。TCP 引擎执行各种 TCP 检查，包括校验和、序列号检查，借助必要的 CRC 运算的帧检查以及 TCP 状态更新。然后提取存储 PDU 并将其分配给存储引擎来处理。存储引擎解释 PDU 中的命令，并在此特定情况下将其识别为活动会话的读取响应。然后验证有效负载完整性和序列完整性，接着在会话数据库表项中更新存储流状态。还从会话数据库表项中检索目的地缓冲区的存储器描述符并将提取的 PDU 有效负载排队到存储流/RDMA 控制器和主机接口块，以便它们通过 DMA 将数据传输到最终的缓冲区目的地。可以将数据与存储器描述符和要执行的命令/操作一起传递到流控制器。在这种情况下，存储用于该活动读取命令的数据。存储流控制器将更新其活动命令数据库。执行引擎将指示调度器分组已退役并且分组处理器复合体准备接收下一命令。

6. DMA 管道级（块 3206），其中主要步骤在块 3212 中示出。一旦存储流控制器对存储器描述符、命令和流状态进行了相应的验证，则将数据块传递到主机 DMA 引擎以便传输到主机存储器。如果编程或实现了此类 QOS 机制，则 DMA 引擎可以执行基于优先级的排队。数据将通过 DMA 传输到主机存储单元。如果这是命令的最后一个操作，则通知主机驱动器命令执行完成。如果这是命令的最后一个操作并且所述命令已排队到完成队列，则释放为该命令分配的资源以便接受新命令。可以收集并与完成状态一起传输执行性能分析、策略管理或其他网络管理或统计任务所需的命令统计信息。

图 33 示出了发起方和目标之间的写入命令操作。发起方将 WRITE 命令（块 3301）发送到目标以启动事务。该命令作为 WRITE PDU（块 3302）在 IP 存储网络上传输。接收方将所接收的命令排队到新的请求队列中。一旦操作中的旧命令完成（块 3304），接收方便分配资源以接受与命令对应的 WRITE 数据（块 3305）。在此阶段，接收方将待传输（R2T）PDU（块 3306）发送到发起方，并指出它愿意接收的数据量以及从哪个位置接收。发起方解释 R2T 请求的字段并根据收到的 R2T 将数据分组发送（块 3307）到接收方。发起方和目标之间的此系列交换将继续，直到命令终止为止。

目标将命令成功完成或错误情况传送到发起方作为响应 PDU, 该响应 PDU 然后终止命令。出现错误时, 可能需要发起方启动恢复进程。这未在图 33 的交换中示出。

图 34 针对图 33 中示出的 R2T PDU 和写入事务的后续写入数据之一示出了 IP 处理器内的数据流。发起方通过其网络介质接口接收 R2T 分组。分组通过所有的级 (块 3401、3402、3403 和 3404), 并且主要步骤在块 3415、3416、3409 和 3410 中示出, 与图 32 中的 READ PDU 类似, 包括接收、安全、分类、调度和执行。该图中未示出安全性处理。在这些阶段之后, R2T 使用图 34 中示出的 DMA 阶段 (块 3405 和 3411) 触发写入数据取回。然后通过执行阶段 (块 3406 和 3412) 对写入数据进行分段并将其置于 TCP/IP 分组内。将针对 WRITE 命令更新 TCP 和存储会话数据库表项并传输数据以响应 R2T。然后将分组排队到输出队列控制器。根据连接的安全性协议, 分组可以进入安全管道级 (块 3407 和 3413)。一旦对分组进行了加密并生成消息验证代码, 则将分组排队到网络介质接口以便传输到目的地。在此级 (块 3408 和 3414), 如果分组处理器没有将分组封装在第二层标头中, 则执行此操作并传输分组。每个流水线级中接下来的步骤与上述 READ PUD 管道级中的步骤类似, 只是多出了写入数据分组阶段, 图中示出了该阶段。在每个级中执行的特定操作取决于命令类型、会话状态、命令状态以及可设置的策略的各种其他配置。

图 35 示出了在发起方和目标之间使用 RDMA 机制实现的 READ 数据传输。发起方和目标在发起 RDMA 数据传输之前注册 RDMA 缓冲区 (块 3501、3502 和 3503)。发起方发出 READ 命令 (块 3510), 将 RDMA 缓冲区作为预期接收方。将该命令传输到目标 (块 3511)。目标准备要读取的数据 (块 3504), 然后执行 RDMA 写入操作 (块 3505) 以将读取数据直接存储到发起方的 RDMA 缓冲区而无需主机介入。然后使用命令完成响应指示操作完成。

图 36 示出了实现 READ 命令流的 RDMA 写入分组的内部体系结构数据流。RDMA 写入分组与网络接口上收到的任何其他有效数据分组一样通

过相同的管道级。该分组通过接收管道级的第二层处理(块 3601 和 3607),在该级对分组进行排队以便调度器检测安全性处理需求。如果需要对分组进行解密或验证,分组将进入安全管道级(块 3602 和 3608)。然后将解密后的分组调度到分类引擎以便该引擎执行已编程的分类任务(块 3603 和 3609)。一旦完成分类,标记的分组便进入调度管道级(块 3604 和 3610),其中调度器根据基于流的调度将该分组分配给资源特定的队列。当预定资源准备执行该分组时,分组被传输到分组处理器复合体(块 3605 和 3611),将在此处执行所有的 TCP/IP 验证、检查和状态更新并提取 PDU。然后存储引擎将 PDU 标识为属于使用 RDMA 实现的存储 PDU 的存储流并解释 RDMA。在这种情况下是到特定 RDMA 缓冲区的 RDMA 写入。提取该数据并将其传递给执行 RDMA 区域转换和保护检查的存储流/RDMA 控制器块,并且对分组进行排队以便通过主机接口执行 DMA 操作(块 3606 和 3612)。一旦分组通过分组处理器复合体完成操作,将通知调度器并使分组从在调度器中包含的状态退役。一旦进入 DMA 阶段, RDMA 数据传输将完成,并且如果这是完成存储命令执行的最后一次数据传输,将使该命令退役并将其分配给命令完成队列。

图 37 示出了使用 RDMA 读取操作的存储写入命令执行。发起方和目标首先向其 RDMA 控制器注册其 RDMA 缓冲区,然后还将缓冲区通知其对等方。然后发起方向目标发出写入命令(块 3701),其中使用 IP 存储 PDU 传输命令。接收方通过首先分配 RDMA 缓冲区以接收写入,然后请求到发起方的 RDMA 读取来执行写入命令(块 3705 和 3706)。然后作为 RDMA 读取响应分组提供要从发起方写入的数据(块 3707 和 3708)。接收方将分组直接存储到 RDMA 缓冲区而无需任何主机交互。如果读取请求针对大于段大小的数据,则发起方将发送多个 READ 响应 PDU 以响应该 READ 请求。数据传输完成后,向发起方传输完成状态并向主机指示命令完成。

图 38 示出了图 37 中所示流事务的一部分的 RDMA 读取请求和产生的写入数据传输的数据流。此数据流非常类似于图 34 中所示的写入数据

流。RDMA 读取请求分组流经各个处理管道级，包括：接收、分类、调度和执行（块 3801、3802、3803、3804、3815、3816、3809 和 3810）。此请求执行后，它将生成 RDMA 读取响应分组。通过首先从系统存储器执行所请求数据的 DMA（块 3805 和 3811），然后通过执行级（块 3806 和 3812）创建段和分组来生成 RDMA 响应。将更新相应的会话数据库表项，并且数据分组在必要时将转到安全性级（块 3807 和 3813）。然后将安全或明文分组排队到传输级（块 3808 和 3814），此级执行相应的第 2 层更新并将分组传输到目标。

图 39 更详细地示出了从发起方发起的存储命令的发起方命令流。如所示的那样，以下是命令所遵循的一些主要步骤：

1. 主机驱动器将命令排队到存储流/RDMA 控制器中的处理器命令队列；
2. 如果命令被成功调度以便操作，则通知主机并保留资源；
3. 如果已建立到目标的连接，则存储流/RDMA 控制器将要操作的命令调度到分组调度器。否则控制器将启动目标会话启动，并在建立会话后将命令调度到分组调度器；
4. 调度器将命令分配给准备好接受此命令的 SAN 分组处理器之一；
5. 处理器复合体向会话控制器发出会话表项请求；
6. 向分组处理器复合体提供会话表项；
7. 分组处理器形成分组以便作为 PDU 承载命令并将分组调度到输出队列；以及
8. 向网络介质接口提供命令 PDU，此接口将命令 PDU 发送到目标。

这是当发起方和目标之间已建立连接后，从发起方向目标发出的大多数命令主要遵循的高级流。

图 40 更详细地示出了读取分组数据流。在此最初使用类似于图 39 中所示的流将读取命令从发起方发送到目标。目标向发起方发送读取响应 PDU，其遵循图 40 中所示的流。如图所示，读取数据分组将通过以下主要步骤：

1. 从网络介质接口块接收输入分组;
2. 分组调度器从输入队列检索分组;
3. 调度分组以进行分类;
4. 从分类器返回带有分类标记的已分类分组;
5. 根据分类和基于流的资源分配, 将分组分配给对分组进行处理的分组处理器复合体;
6. 分组处理器复合体在会话高速缓存中查找会话表项(如果在本地不存在);
7. 将会话高速缓存表项返回到分组处理器复合体;
8. 分组处理器复合体执行 TCP/IP 操作/IP 存储操作并在有效负载中提取读取数据。将带有诸如 MDL (存储器描述符列表) 之类的相应目的地标记的读取数据提供给主机接口输出控制器; 以及
9. 主机 DMA 引擎将读取数据传输到系统缓冲存储器。

在示出安全分组流的图 32 中更详细地提供了这些步骤中的一些步骤, 而图 40 示出了明文读取分组流。通过使用所披露的处理器相应资源(本领域的技术人员能够通过本申请的教导执行此操作), 此流程和本申请中所示的其他流程可应用于存储和非存储数据传输。

图 41 更详细地示出了写入数据流。写入命令遵循的流类似于图 39 中所示的流。发起方向目标发送写入命令。目标使用待传输(R2T) PDU 对发起方进行响应, 此 PDU 向发起方指示目标已准备好接收指定量的数据。发起方然后将所请求数据发送到目标。图 41 示出了所请求写入数据分组所遵循的从发起方到目标的 R2T。此流中遵循的主要步骤如下所示:

1. 从网络介质接口块接收输入分组;
2. 分组调度器从输入队列检索分组;
3. 调度分组以进行分类;
4. 从分类器返回带有分类标记的已分类分组;
 - a. 根据分类和基于流的资源分配, 将分组分配给对分组进行处理的分组处理器复合体;

5. 分组处理器复合体在会话高速缓存中查找会话表项（如果在本地不存在）；
6. 将会话高速缓存表项返回分组处理器复合体；
7. 分组处理器确定 R2T 并通过向存储流/RDMA 控制器发出请求来请求写入数据；
8. 流控制器启动到主机接口的 DMA；
9. 主机接口执行 DMA 并将数据返回到主机输入队列；
10. 分组处理器复合体从主机输入队列接收数据；
11. 分组处理器复合体将数据形成有效的 PDU 和分组，更新相应的会话表项并将分组传输到输出队列；以及
12. 将分组传输到输出网络介质接口块，此接口块将数据分组传输到目的地。

图 41 中的流示出了明文数据传输。如果数据传输需要是安全的，则流类似于图 43 中所示的流，其中输出数据分组通过安全分组进行路由，如标为 11a 和 11b 的箭头所示。如果安全，则输入 R2T 分组还通过安全性引擎进行路由（这未在图中示出）。

图 42 示出了当分组为密文或安全时的读取分组流。先前在图 32 中通过相关说明更详细地示出了此流。安全读取流与明文读取流之间的主要区别是分组最初由分类器分类为安全分组，并且因此被路由到安全性引擎。这些步骤由标为 2a、2b 和 2c 的箭头示出。安全性引擎对分组进行解密并执行消息验证，并且将明文分组传输到输入队列以进一步处理，如标为 2d 的箭头所示。然后调度器检索明文分组并将其提供给分类引擎，如图 42 中标为 2e 和 3 的箭头所示。其余步骤和操作与上述图 40 中的步骤和操作相同。

图 44 示出了 RDMA 缓冲区通知流。此流被示为与图 39 中所示的任何其他存储命令流非常类似。在主要步骤中执行的详细操作根据命令而有所不同。对于 RDMA 缓冲区通知和注册，将创建和记录 RDMA 区域 ID 以及记录此区域的地址转换机制。RDMA 注册还包括访问控制的保护密钥

并且可以包括 RDMA 传输所需的其他字段。为命令创建分组的步骤类似于图 39 的步骤。

图 45 更详细地示出了 RDMA 写入流。RDMA 写入类似于到接收 RDMA 写入的发起方的正常读取 PDU。RDMA 写入分组与图 40 中所示的读取 PDU 遵循相同的主要流程步骤。RDMA 传输包括 RDMA 地址转换和区域访问控制密钥检查，更新 RDMA 数据库表项以及其他会话表项。主要流程步骤与常规读取响应 PDU 相同。

图 46 更详细地示出了 RDMA 读取数据流。此图示出了由发起方从目标接收的 RDMA 读取请求以及从发起方写入目标的 RDMA 读取数据。此流程非常类似于存储写入命令遵循的 R2T 响应。在此流程中使用 RDMA 读取完成存储写入命令。分组遵循的主要步骤与图 41 中所示的 R2T/写入数据流基本相同。

图 47 示出了会话创建流的主要步骤。此图示出了针对在发起方和目标之间启动会话时所需的此缓慢路径操作使用控制平面处理器。此功能可以通过分组处理器复合体实现。但是，在此它被示为使用控制平面处理器实现。两种方法都可接受。以下是会话创建过程中的主要步骤：

1. 主机驱动器调度命令；
2. 通知主机驱动器已调度命令并且已传送主机所需的任何控制信息；
3. 存储流/RDMA 控制器检测到将命令发送到不存在会话的目标的请求，并且因此它将请求传送到控制平面处理器以建立传输会话；
4. 控制平面处理器向输出队列发送 TCP SYN 分组；
5. 将 SYN 分组传输到网络介质接口，从网络介质接口将该分组传输到目的地；
6. 目的地在接收 SYN 分组之后使用 SYN-ACK 响应进行响应，在从网络介质接口接收分组后将该分组排队在输入队列中；
7. 分组调度器检索分组；
8. 将分组传送到分类引擎；
9. 向调度器返回带有标记的已分类分组；

10. 调度器根据分类将此分组转发到控制平面处理器;
11. 处理器然后使用 ACK 分组对输出队列进行响应;
12. 然后将分组传输到最终目的地, 因此完成会话建立握手; 以及
13. 建立会话后, 将此状态提供给存储流控制器。因此创建会话表项, 该表项随后被传送到会话存储器控制器 (此部分未在图中示出)。

如在步骤 13 中那样获取已建立状态的会话之前, 可能需要控制平面处理器执行存储协议的完整登录阶段, 从而针对特定连接 (如果这是存储数据传输连接) 交换参数并记录参数。验证登录并且参数交换完成后, 使会话进入上面步骤 13 中所示的会话建立状态。

图 48 示出了会话关闭流中的主要步骤。此流程中的步骤非常类似于图 47 中的步骤。两个流程之间的主要区别是在发起方和目标之间传输 FIN、FIN-ACK 和 ACK 分组, 而不是用于会话创建的 SYN、SYN-ACK 和 ACK 分组。主要步骤在其他方面非常类似。在此的另一个主要区别是没有创建相应的会话表项, 而是从会话高速缓存和会话存储器中删除相应的会话表项。记录连接的运行统计信息并可以将其提供给主机驱动器, 但这未在图中示出。

图 49 示出了从目标角度的会话创建和会话关闭步骤。以下是创建会话所遵循的步骤:

1. 通过网络介质接口接收来自发起方的 SYN 请求;
2. 调度器从输入队列检索 SYN 分组;
3. 调度器将此分组发送到分类引擎以进行分类;
4. 分类引擎返回带有相应标记的已分类分组;
5. 调度器根据分类为 SYN 分组, 将此分组传输到控制平面处理器;
6. 控制平面处理器使用 SYN-ACK 确认分组进行响应。它还请求主机分配相应的缓冲区空间以用于来自发起方的未经请求的数据传输 (此部分未示出);
7. 将 SYN-ACK 分组发送到发起方;
8. 发起方然后使用 ACK 分组确认 SYN-ACK 分组, 完成三向握手。

在第 2 层处理之后,通过网络介质接口接收此分组并将其排队到输入队列;

9. 调度器检索此分组;

10. 将分组发送到分类器;

11. 将已分类的分组返回调度器并将其调度为提供给控制处理器来完成三向握手;

12. 控制器获取 ACK 分组;

13. 控制平面处理器现在具有处于建立状态的连接,并将其传送到在会话高速缓存中创建表项的存储流控制器; 以及

14. 通知主机驱动器已完成会话创建。

会话建立还可以包括登录阶段,此阶段未在图 49 中示出。但是,登录阶段和参数交换在会话进入已完全配置和已建立状态之前发生。这些数据传输和握手可以主要由控制处理器完成。执行这些步骤后,可以执行上面流程中的其余步骤。

图 50 和 51 示出了目标子系统中的写入数据流。图 50 示出了 R2T 命令流,目标使用此流程通知发起方它已准备好接受来自发起方的数据写入。然后发起方发送在目标处接收的写入,内部数据流在图 51 中示出。这两个图一起示出了一个 R2T 和数据写入对。以下是如图 50 和 51 一起所示的所遵循的主要步骤:

1. 目标主机系统响应于接收如图 33 中所示的写入请求,准备相应的缓冲区以接受写入数据,并在准备好时通知存储流控制器可以向发起方发送准备传输请求;

2. 流控制器向主机驱动器确认请求的接收以及用于 DMA 的缓冲区指针;

3. 流控制器然后将要执行的 R2T 命令调度到调度器;

4. 调度器向准备好执行此命令的分组处理器复合体之一发出命令;

5. 分组处理器从会话高速缓存控制器请求会话表项;

6. 将会话表项返回分组处理器;

7. 分组处理器形成 TCP 分组,封装 R2T 命令并将其发送到输出队列;

8. 然后将分组发出到网络介质接口，此接口然后将分组发送到发起方。如果传输需要是安全传输，则可以包括安全性引擎；

9. 然后如图 51 中所示，发起方通过向目标发送写入数据对 R2T 进行响应。网络介质接口接收分组并将其排队到输入阵列；

10. 分组调度器从输入队列检索分组；

11. 将分组调度到分类引擎；

12. 分类引擎将带有分类标记的已分类分组提供给调度器。所示流程用于未加密分组并且因此不执行安全性引擎；

13. 调度器根据基于流的资源分配队列将分组分配给分组处理器队列。然后在分组处理器准备好执行分组时将此分组传输到分组处理器复合体；

14. 分组处理器请求会话高速缓存表项（如果它在其本地高速缓存中尚未具有此表项）；

15. 将会话表项返回发出请求的分组处理器；

16. 分组处理器执行所有 TCP/IP 功能，更新会话表项，并且存储引擎作为写入命令提取 PDU 以响应先前的 R2T。它更新存储会话表项并将分组路由到主机输出队列以将其传输到主机缓冲区。分组可以标记有存储器描述符或存储器描述符列表，所述描述符或描述符列表可用于执行此分组到主机已分配目的地缓冲区的 DMA；以及

17. 主机接口块执行 DMA 以完成此写入数据命令部分。

图 52 示出了目标读取数据流。此流程非常类似于图 41 中所示的发起方 R2T 和写入数据流。此流程中所执行的主要步骤如下所示：

1. 从网络介质接口块接收输入分组；

2. 分组调度器从输入队列检索分组；

3. 调度分组以进行分类；

4. 从分类器返回带有分类标记的已分类分组；

a. 根据分类和基于流的资源分配，将分组分配给对分组进行处理的分组处理器复合体

5. 分组处理器复合体在会话高速缓存中查找会话表项（如果在本地不存在）；
6. 将会话高速缓存表项返回分组处理器复合体；
7. 分组处理器确定读取命令 PDU 并通过向流控制器发出请求来请求读取数据；
8. 流控制器启动到主机接口的 DMA；
9. 主机接口执行 DMA 并将数据返回主机输入队列；
10. 分组处理器复合体从主机输入队列接收数据；
11. 分组处理器复合体将数据形成有效的 PDU 和分组，更新相应的会话表项并将分组传输到输出队列；以及
12. 将分组传输到输出网络介质接口块，此接口块将数据分组传输到目的地。

上面对流程的讨论是高带宽数据传输中某些主要流程的举例说明。诸如分段数据流、具有多种不同类型错误的错误流、名称解析服务流、地址解析流、登录和注销流之类的几种流程未被示出，但本发明的 IP 处理器支持这些流程。

如上面说明中所讨论的，由于企业网络的边界模糊，外围安全性模型并不足以保护企业网络免受安全性威胁。此外，在内部发生大量未经授权的信息访问。外围安全性方法无法阻止此类安全性攻击。因此，跨网络部署安全性并从内部以及外围保护网络至关重要。企业网络内部的 LAN 和 WAN 中的网络线路速度达到 1Gbps、几 Gbps 和 10Gbps。如前所述，如果以软件实现，则分布式防火墙和安全性方法在每个系统主机 CPU 上需要大量处理开销。此开销可以导致增加服务器响应等待时间，降低其总体吞吐量并为应用剩下很少的处理周期。为了解决上述问题，需要能够部署软件驱动的安全性服务的高效硬件实现。本申请的处理器解决了这些关键问题中的一些问题。此外，在高线路速度时将基于软件的 TCP/IP 协议处理从主机 CPU 卸载到协议处理硬件以减小对主机 CPU 的影响至关重要。因此，协议处理硬件应提供措施以执行诸如防火墙、加密、解密、VPN 之类

的安全性功能。所述处理器提供了可以解决企业网络内不断增长的分布式安全性和高线路速度需求的此类硬件体系结构。

图 53 示出了具有外围防火墙的传统企业网络。此图示出了企业网络内部的局域网和存储区域网络。此图示出了一组使用无线 LAN 连接到企业网络的客户机 5301 (1) 到 5301 (n)。可以具有多个不同类型的客户机, 例如手持计算机、PC、瘦客户机、便携式计算机、笔记本电脑、平板电脑等。此外, 它们可以使用无线 LAN 访问点 (WAP) 5303 连接到企业 LAN。可以具有一个或多个连接到 LAN 的 WAP。同样, 此图还示出了多个通过有线网络连接到企业 LAN 的客户机。这些客户机可以位于不同子网段或同一子网段或以点对点连接直接链接到交换机, 具体取决于网络规模、线路速度等。网络可以具有多个为设备网络提供内部连接性的交换机和路由器。此图还示出了网络连接存储设备 5311, 其为客户机提供网络文件服务和存储服务。此图还示出了一个或多个连接到网络的服务器 5307 (1) 到 5307 (n) 以及 5308 (1) 到 5308 (n), 它们为网络内部的客户机提供这些服务器上托管的各种应用服务, 以及通过外部作为网页访问或其他网络访问进行访问的服务。服务器农场中的服务器可以以传统的三层或 n 层网络进行连接, 提供诸如 Web 服务器、应用服务器、数据库服务器之类的不同服务。这些服务器可以包含用于所需存储的直接连接存储设备和/或使用 SAN 连接性和交换机 5309 (1) 到 5309 (n) 连接到存储区域网络 (SAN), 以便连接到存储系统 5310 (1) 到 5310 (n) 以满足其存储需求。存储区域网络还可以使用网关设备 5313 连接到 LAN 以便为 LAN 客户机提供对存储系统的访问。存储系统还可以类似于 NAS 5311 直接连接到 LAN 以提供使用诸如 iSCSI 之类的协议的块存储服务。这未在图中示出。此图中所示的网络通过外围防火墙 5306 与外部网络安全隔离。如此图中所示, 此类环境中的内部网络没有实现安全性, 这会对内部人员攻击产生严重的安全性漏洞。

图 54 示出了具有分布式防火墙和安全性能的企业网络。所示的网络配置类似于图 53 中的配置。此类网络中所示的分布式安全特性可以由中央

IT 管理者(多个)5412 通过一组中央网络管理系统进行配置、监视、管理、启用和更新。管理者(多个)可以从管理站(多个)设置分布式安全性策略,将相应的策略规则分发到每个已启用的节点以实现分布式安全性策略,并使用本申请的处理器监视分布式安全性处理器中的任何违反或报告。所述网络可以是包括一个或多个节点、一个或多个管理站或其组合的网络。此图示出了 SAN 设备没有在分布式安全性网络内。此图中的 SAN 设备可以在单独安全性域内或可以被信任以便通过 SAN 边缘处的安全性免受内部人员和外部人员的攻击。

图 55 示出了具有其中 SAN 设备也在分布式安全性域内的分布式防火墙和安全性能的企业网络。其余的网络配置可以类似于图 54 中的配置。在此方案中, SAN 设备可以与其余网络设备实现类似的安全性策略并且可以在同一 IT 管理系统的控制之下。SAN 安全性的实现可以不同于网络的其余部分,具体取决于安全性需求、信息的敏感度和潜在的安全性风险。例如, SAN 设备可以实现完整的加密/解密服务以及防火墙安全性能,以确保不会发生未授权访问以及放置在 SAN 外的数据始终处于保密模式。可以从同一网络管理系统分发这些策略和规则,或者可以具有可用于创建此类分布式安全 SAN 的特殊 SAN 管理系统(未示出)。图 54 和图 55 中的系统使用本申请的处理器和分布式安全性系统。

图 56 示出了中央管理器/策略服务器和监视站(也称为中央管理器)。中央管理器包括安全策略开发人员接口(块 5609), IT 管理者使用此接口输入组织的安全策略。安全策略开发人员接口可以是使 IT 管理者能够以安全策略描述语言输入安全策略的命令行接口、脚本工具、图形接口或它们的组合。它还可以通过安全通信连接提供从远程对 IT 管理者的访问。安全策略开发人员接口与一组使 IT 管理者能够高效输入组织策略的规则模块协作。规则模块可以提供规则模板,所述模板可以由 IT 管理者填充或可以是轻松输入规则的交互工具。这些模块根据分布式安全系统所支持的能力来提供规则。块 5601 为安全性开发人员接口提供联网第 2 到第 4 层(L2、L3、L4)规则、规则类型、模板等。这些规则可以包括源和目的地的 IP

地址、源和目的地的 L2 地址、L2 有效负载类型、缓冲区溢出条件、服务类型、连接优先级、链路使用统计信息等或它们的组合。协议/端口级规则（块 5602）为安全性开发人员接口提供规则、规则类型、模板等。这些规则可以包括诸如 IP、TCP、UDP、ICMP、IPSEC、ARP、RARP 之类的协议类型或源端口或目的地端口（包括用于公知上层应用/协议的公知端口）或它们的组合。块 5603 为安全性开发人员接口提供应用级或上层（L5 到 L7）规则、规则类型、模板等。这些规则可以包括依赖于上层应用或协议（例如 HTTP、XML、NFS、CIFS、iSCSI、iFCP、FCIP、SSL、RDMA 等）的类型、其使用模型、其漏洞或它们的组合的规则。基于内容的规则（块 5604）为安全性开发人员接口提供规则、规则类型、模板等以便输入内容相关的规则。这些规则可以像其他规则那样随时间而变化，以涵盖公知的威胁或潜在的新威胁并包括各种情况，如社会保险号、保密/专用文档、雇员记录、患者记录、信用卡号、入侵 URL、公知病毒特征、缓冲区溢出条件、长 Web 地址、入侵语言、淫秽内容、垃圾邮件等或它们的组合。可以提供这些规则、模板或规则类型以便分布式安全系统管理者以所选策略描述语言（多种）轻松创建规则。安全策略开发人员接口可以在没有规则模块的情况下存在，并且继续为 IT 管理者提供装置以便在系统中输入安全策略。通过所述接口输入的以安全策略语言表示的规则然后由安全规则编辑器（块 5611）编辑以便分发到网络节点。安全规则编辑器利用网络连接性数据库 5605 以及节点能力和特征数据库 5606 为网络中是监视/实施安全策略一部分的每个节点生成特定规则。网络连接性数据库包括物理邻接信息，或物理层连接性，或链路层连接性，或网络层连接性，或 OSI 层两个地址，或 OSI 层三个地址，或路由信息或它们的组合。节点能力和特征数据库包括：硬件安全特性或软件安全特性或规则引擎大小或安全引擎（多个）性能或服务特性或主机操作系统或托管应用（多个）或网络连接的线路速度或主机性能或它们的组合。这些数据库中的信息将使安全规则编辑器能够正确地将安全策略映射到节点特定的规则。将节点特定规则和通用全局规则存储到规则数据库 5607 并从其中检索所述规则。安全规则编

辑器然后与规则分发引擎 5608 协作以将编辑后的规则分发到每个节点。规则分发引擎与分布式安全系统的每个安全节点交互以便发送要在此特定节点处使用的规则集。规则分发引擎可以直接从规则数据库检索规则集或与安全规则编辑器协作或采用它们的组合以检索规则。将规则传播到各个节点后，中央管理器便开始监视和管理网络。

中央管理器与安全网络中的每个节点协作，以便使用事件和报告收集/管理引擎 5616 收集实施事件和报告、统计信息、违反等。事件/报告收集引擎与安全监视引擎 5613 协作以创建事件和信息报告数据库 5614 和 5615，这些数据库保存所收集信息的永久性记录。安全监视引擎分析报告和事件以检查任何违反并且转而可以向 IT 管理者通知这些违反。根据发生违反时要执行的操作，安全监视引擎可以创建可被重新分发到节点的策略或规则更新。安全监视引擎与安全策略管理器接口 5612 和策略更新引擎 5610 协作以创建和重新分发更新。安全策略管理器接口为 IT 管理者提供工具以执行事件和信息记录搜索。IT 管理者可以根据所监视事件或组织策略中的其他搜索或更改来开发新规则或安全策略更新并创建策略更新。这些更新由安全策略编辑器编辑并重新分发到网络。安全策略开发人员接口 5609 可以根据实施方式选择来提供安全策略管理器接口 5612 和策略更新引擎 5610 的功能。可以进行此类功能和功能块重组而不偏离本申请的教导。当存在需要解决的违反或任何其他需要采取的操作（例如，为安全性启用节点、禁用节点、更改节点的角色、更改节点的配置、在节点上启动/停止/部署应用，或按需提供其他能力或其他管理功能或其组合以便中央管理器高效管理节点的网络）时，安全监视引擎、安全策略管理器接口和事件/报告收集/管理接口还可以用于管理特定的节点。

图 57 示出了本申请的中央管理器流。中央管理器可以包括流程块所示的各种过程步骤。IT 管理者（多个）在块 5701 所示的中央管理系统（多个）中创建和输入组织的安全策略。然后安全策略编辑器使用网络连接性数据库以及节点能力和特征数据库将策略编辑为规则，如块 5702 所示。中央管理器然后在块 5703 中通过已启用安全能力的网络从节点特征数据库

中标识节点以将规则分发到这些节点。管理器然后可以从这些节点中选择一个节点（如块 5704 所示），并从规则数据库检索相应的安全规则（如块 5705 所示），然后将规则传送到节点（如块 5706 所示以及如图 58 所示）。中央管理器继续检索规则和传送规则的过程直到所有节点均已被处理，如块 5707 中所有节点已完成的比较所示。将规则分发到所有节点后，中央管理器便开始管理和监视网络以检查策略实施、违反或其他管理任务，如块 5708 所示。如果监视到任何策略更新，则中央管理器退出监视以通过检查创建和更新新的策略，如块 5709 和 5710 所示。如果存在新的策略更新，则中央管理器遍历图 57 的流程以编辑规则并将它们重新分发到受影响的节点，然后继续监视网络。当其他模块在处理安全策略和规则的更新时，中央管理器的事件收集引擎继续监视和记录事件和信息报告。因此当规则进行更新和分发时，网络将继续被监视。规则更新完成后，安全监视引擎和其他引擎将处理所收集的报告。可以并行执行将规则传送到节点以及监视/管理节点以提高安全系统的性能和效率。中央管理器可以将新的规则或更新并行（而不是使用串行流）传送到多个节点，并将已接收规则的节点指定为中央管理器的监视/管理状态。同样，策略创建或更新也可以与规则编辑、分发和监视并行执行。

图 58 示出了本申请的规则分发流。与安全策略编辑器协作的规则分发引擎检索要传送到特定节点的规则或规则集，如 5801 所示。然后它启动与选定节点的通信，如 5802 所示。中央管理器和节点可以使用约定的方法或协议相互验证，如 5803 所示。验证可以包括完整的登录过程，或安全加密会话或明文模式会话或它们的组合。节点和中央管理器相互验证后，便在中央管理器和节点的控制平面处理器或基于主机的策略驱动器之间建立通信，如 5804 所示。建立通信后，规则分发引擎向节点发送规则或规则集或更新后的规则或它们的组合，如 5805 所示。此规则交换可以通过安全/加密会话或明文链路，具体取决于组织的策略。为传送规则而部署的协议可以使用公知的协议或专用协议。将规则集发送到节点后，中央管理器可以等待从节点接收已成功将新规则插入节点的确认，如 5806 所示。接收成功

确认后,便结束用于一个节点的规则分发流,如 5807 所示。用于节点的相应规则数据库表项将被标记有分发完成状态。为所有需要从中央管理器的规则分发引擎接收规则的节点重复图 58 的流程。规则分发引擎还可以将规则并行分发到多个节点以提高规则分发过程的效率。在此情况下,规则分发引擎可以为多个节点并行执行流程的各个步骤,如验证节点,建立与节点的通信,向节点发送一个或多个规则等。

图 59 示出了本申请的控制平面处理器或基于主机的策略驱动器流。针对每个遵循本申请的分布式安全性、包括硬件处理器的节点执行此流程。当中央管理器启动策略规则分发后,或当重置或加电或其他管理事件或它们的组合后,策略驱动器将建立与中央管理器/策略服务器的通信,如 5901 所示。策略驱动器从中央管理器接收规则集或对现有规则的更新,如 5902 所示。如果将规则的格式设置为插入特定策略引擎实施、大小等,规则将被接受以便在策略引擎中进行配置。如果中央管理器始终正确设置规则的格式,则可以避免执行块 5903 中所示的检查。否则,如果规则并非始终被设置格式或以其他方式准备好直接插入策略引擎,如块 5903 中确定的那样,则驱动器为节点配置规则,如块 5904 所示。驱动器然后与处理器的策略引擎的数据库初始化和接口(图 20 的块 2011)通信。这由块 5905 示出。然后驱动器向策略引擎发送规则,该引擎将更新如图 30 中所示的引擎数据结构中的此规则,图 30 中的数据结构包括三进制或二进制 CAM、关联存储器、ALU、数据库描述以及图 20 的分类/策略引擎中的其他元件。这由块 5906 示出。此过程将继续直到通过决策过程在策略引擎中输入所有规则,如 5907、5908 和 5906 所示。输入所有规则后,策略引擎将激活与驱动器协作的新规则,如块 5909 所示。驱动器然后将规则更新/发送到永久性存储装置以供将来参考和/或检索,如块 5910 所示。驱动器然后将节点中的更新完成和新规则激活传送到中央管理器/策略服务器,如块 5911 所示。策略驱动器然后可以进入将管理信息、事件、报告传送到中央管理器的模式。驱动器的此部分未在图中示出。可以由主机上的安全过程或节点的控制平面处理器执行管理功能。上述机制允许针对协议栈处理创建安

全操作环境，其中即使主机系统受到病毒或恶意攻击的损坏，它也允许维护网络安全性和完整性，因为基于控制平面处理器的策略驱动器不允许主机系统影响策略或规则。策略引擎中的活动规则将阻止病毒或入侵者使用此系统或节点以用于进一步传播病毒或攻击网络中的其他系统。所述规则还可以阻止攻击者从系统提取任何重要信息，如信用卡号、社会保险号、医疗记录等。此机制显著增加了下一代计算系统的受信计算环境需求。

此流程的某些或所有部分可以并行执行，并且某些部分可以组合在一起执行。例如，策略驱动器可以将一个或多个规则一起传送到数据库初始化/管理接口（块 2011），此接口然后可以以原子方式更新策略引擎中的规则，而不是每次更新一个规则。此外，当策略驱动器或策略引擎或其组合接收到新规则时，硬件处理器可以对入站或出站网络业务继续并行执行规则实施和活动规则集分析。

图 60 示出了可以在使用本申请的分布式安全系统中部署的规则。IT 管理者（多个）可以确定需要针对不同访问类型部署的策略。这些策略在中央管理系统 5512 或 5412 处被转换为规则，以便分发到网络中实现一种或多种安全能力的每个节点。然后将规则提供给相关节点上的处理器。与分类和策略引擎 1703 和处理器的数据库初始化/管理控制接口（图 20 的 2011）协作的控制平面处理器（图 17 的 1711）在处理器中配置规则。每个实现分布式安全系统的节点可以具有需要应用于在此节点处通过、发起或终止的网络业务的独特规则。中央管理系统与所有相应的节点交互并为每个节点提供其相关规则。中央管理系统还与控制平面处理器（其与节点的分类/策略引擎协作）交互以便从分布式安全系统的节点中检索规则实施信息和其他管理信息。

图 60 示出了可应用于网络中的一个或多个节点的规则。规则可以包含比图中所示更多或更少的字段。在此图中，规则包括应用规则的网络业务的方向（输入或输出）；源和目的地地址，其可以属于内部网络节点地址或属于网络外部节点的地址；分组的协议类型，例如 TCP、UDP、ICMP 等以及源端口和目的地端口和任何其他深度分组字段，包括 URL 信息、

敏感信息（如信用卡号或社会保险号）或任何其他受保护信息（如用户名、密码等）。然后规则包含操作字段，此字段指示当匹配特定规则时需要执行的操作。所述操作可以包括各种类型，例如允许访问，拒绝访问，丢弃分组，关闭连接，记录请求，发送警报或上述操作的组合或可适用于匹配规则的更多操作。可以以优先级方式从上到下或以任何其他可在系统中实现的顺序应用规则。最后一个规则指示当其他规则都不匹配时的情况，如此实例中所示，将拒绝访问。

图 61 示出了图 16 和图 17 中所示的 IP 处理器的 TCP/IP 处理器版本。此处理器包括网络接口（块 6101），此接口用于将此处理器连接到网络。网络接口可以是有线或无线以太网接口、Sonet 上的分组接口、介质无关接口（MII）、GMII、XGMII、XAUI、系统分组接口、SPI 4 或 SPI 5 或其他 SPI 派生物或其他网络协议接口或它们的组合。这是用于向与此处理器连接的网络发送分组或从其中接收分组的接口。智能流控制器和分组缓冲区（块 6103）提供图 17 的块 1702 的分组调度器功能以及块 1701 和 1712 的输入和输出队列控制器功能。当特定实施方式支持安全能力时，可编程分类/规则引擎/安全处理（块 6102）提供块 1703 的分类和策略/规则处理功能以及块 1705 的安全处理功能。

块 6104 的 TCP/IP 分组处理器引擎类似于块 1706（a）到 1706（n）的 SAN 分组处理器的 TCP/IP 处理器引擎。连接（会话）存储器（块 6105）提供块 1704 的 IP 会话高速缓存/存储器的功能，而块 6106 的连接管理器和控制平面处理器提供类似于块 1704 和 1711 的会话控制器和控制平面处理器功能。RDMA 控制器（块 6107）提供类似于块 1708 的 RDMA 功能。存储器控制器（块 6109）提供类似于块 1704 的存储器控制器所提供的存储器接口。TCP/IP 处理器可以具有外部存储器，此存储器可以是 SRAM、DRAM、FLASH、ROM、EEPROM、DDR SDRAM、RDRAM、FCRAM、QDR SRAM、磁 RAM 或磁存储器或其他静态或动态随机存取存储器派生物或它们的组合。主机/结构/网络接口（块 6108）提供主机总线接口或交换机结构接口或网络接口，具体取决于其中结合此处理器的系统。例如在

服务器或服务器适配器环境中，块 6108 将提供类似于块 1710 的主机总线接口功能，其中主机总线可以是 PCI 总线、PCI-X、PCI-Express 或其他 PCI 派生物或其他主机总线（如 AMBA 总线或 RapidIO 总线或 HyperTransport 或其他派生物）。交换机或路由器或网关或具有用于连接多个线卡的交换机结构的设备将针对块 6108 具有相应的结构接口功能。这可以包括具有优先级机制以避免线路头部阻塞的队列、交换机结构所需的分段和取消分段电路，以及相应的流控制机制以确保对交换机结构资源的平衡使用。在诸如网关或在入口和出口连接到网络的设备之类的环境中，块 6108 将提供类似于块 6101 的网络接口功能。

图 61 中所示的 TCP/IP 处理器是图 16 和图 17 中所示的体系结构的版本，如通过上面说明所了解的那样。块 6104 的 TCP/IP 处理器引擎可以使用块 1706 (a) 到 1706 (n) 的 SAN 分组处理器代替，并且这两种体系结构将提供相同的功能。因此图 61 可以被看作图 16 和图 17 中所示体系结构的不同视图和/或分组。可以通过 SAN 分组处理器的分组引擎块加强所述 TCP/IP 处理器引擎以提供可编程处理，其中除了根据分组对分组进行协议处理以外，还可以部署其他服务。在本申请中，图 61 的块 6110（如一组块周围的点线所示）被称为“TCP/IP 处理器核心”。RDMA（块 6107）被示为 TCP/IP 处理器核心的一部分，尽管它在诸如低线路速度应用或不支持 RDMA 的应用之类的特定 TCP/IP 处理器核心实施例是可选项。同样还可以存在安全引擎，具体取决于所选实施方式和系统实施例。

图 62 示出了本申请的自适应 TCP/IP 处理器。此处理器包括网络接口（块 6201）、主机/结构/网络接口（块 6207）、TCP/IP 处理器核心（块 6202）、运行时自适应处理器（RAP）（块 6206）或其组合。自适应 TCP/IP 处理器还可以包括自适应控制器（块 6203）、配置存储器（块 6204）、存储器接口（块 6205）、数据缓冲区（块 6209）、存储控制器（块 6208）、RAP 接口（块 6210）、RAP 扩展接口（块 6211）或其组合。TCP/IP 处理器核心（块 6202）是图 61 中所示的 TCP/IP 处理器核心（块 6110）。如先前所讨论的，TCP/IP 处理器核心的安全性和 RDMA 块可以存在也可

以不存在，具体取决于应用和系统环境。TCP/IP 处理器核心提供完整的 TCP/IP 协议处理、协议终止和协议启动功能。TCP/IP 处理器核心可以提供 TCP/IP 协议栈，其中包括以下硬件实现的功能中的至少一项：

- a. 发送和接收数据，包括上层数据；
- b. 建立传输会话和会话关闭功能；
- c. 执行错误处理功能；
- d. 执行超时；
- e. 执行重新传输；
- f. 执行分段和排序操作；
- g. 维护有关活动传输会话的协议信息；
- h. 维护一个或多个会话连接中的每个连接的 TCP/IP 状态信息；
- i. 对数据分组进行分段和取消分段；
- j. 路由和转发数据和控制信息；
- k. 向对方发送或从其中接收为 RDMA 保留的存储器区域；
- l. 在 RDMA 数据库中记录为 RDMA 保留的所述存储器区域并维护所述数据库；
- m. 执行 RDMA 能力所提供的操作；
- n. 执行安全管理功能；
- o. 执行策略管理和实施功能；
- p. 执行虚拟化功能；
- q. 传送错误；
- r. 处理第 2 层介质访问功能以接收和传输数据分组，验证分组，处理错误，传送错误以及其他第 2 层功能；
- s. 处理物理层接口功能；
- t. 执行 TCP/IP 校验和生成和验证功能；
- u. 处理乱序分组；
- v. CRC 计算功能；
- w. 处理直接数据放置/传输；

- x. 上层成帧功能;
- y. 处理套接字 API 的功能和接口;
- z. 为所传输数据形成 TCP/IP 的分组标头以及从所接收分组提取有效负载; 以及
 - aa. 分别针对要传输的数据和已接收的数据分组的第 2 层协议处理标头信息和有效负载提取。

如前所述, TCP/IP 处理器核心可以提供传输层 RDMA 能力。TCP/IP 处理器核心还可以提供安全性功能, 如网络层安全性、传输层安全性、套接字层安全性、应用层安全性或其组合以及线路速度加密和解密能力。因此, TCP/IP 处理器核心还可以提供硬件形式的安全 TCP/IP 栈, 此栈具有以硬件实现的上述几种功能。尽管已经使用本申请中所示的 TCP/IP 处理器核心说明了自适应 TCP/IP 处理器, 但是 TCP/IP 处理器核心可以具有各种其他体系结构。除了本申请中披露的体系结构之外, TCP/IP 处理器核心还可以是固定功能实施方式, 或可以作为硬件状态机实现或可以支持部分协议卸载能力 (例如支持以硬件进行的快速路径处理), 其中控制平面处理以及会话管理和控制可以位于单独的控制平面处理器或主机处理器或上述各种备选体系结构的组合中。所选的 TCP/IP 处理器核心体系结构还可以包括用于安全性或 RDMA 或其组合的功能。此外, 通过使用适合协议的处理器核心代替所述 TCP/IP 处理器核心, 自适应 TCP/IP 处理器体系结构可以用于 TCP/IP 之外的协议, 如 SCTP、UDP 或其他传输层协议。这将能够创建用于相关特定协议的自适应协议处理器。此类处理器的运行时自适应处理器的运行可以类似于本申请中的说明, 并且通过使用其动态自适应能力为类似应用/服务提供硬件加速。

运行时自适应处理器 (块 6206) 提供动态可更改硬件, 其中逻辑和互连资源可以通过编程方式实时适用于根据应用/服务的需要创建虚拟硬件实施方式。自适应控制器 (块 6203) 可以用于动态更新 RAP 块。自适应控制器可以与主机处理器或控制平面处理器或 TCP/IP 处理器或其组合连接, 以便确定何时切换 RAP 块的配置从而创建新的具体化以支持所需的硬

件功能,此 RAP 块的具体化应执行的功能,从何处取回新的具体化,具体化的有效时间,何时更改具体化,以及在 RAP 块中提供多种同时进行的功能支持。RAP 块可以从一种具体化动态切换到另一种具体化,具体取决于在 TCP/IP 处理器核心中进行的分析。例如,TCP/IP 处理器核心可以具有编程的策略,此策略将要求处理器核心标记任何可能包含 XML 数据的已接收数据有效负载,并使所提取的数据有效负载通过 RAP 以进行处理而不是将数据直接发送到主机处理器。在这种情况下,当接收到包含 XML 数据的分组时,TCP/IP 处理器核心可以相应地标记数据,并将分组排队到外部存储器以便由 RAP 进一步处理,或将数据传递到块 6209 的数据缓冲区中以便由 RAP 进一步处理。TCP/IP 处理器核心可以连接到 RAP 接口(块 6210),此接口可以提供 TCP/IP 处理器核心与 RAP 和自适应控制块连接所需的功能。此功能可以直接是 RAP 或自适应控制器或 TCP/IP 处理器核心的一部分。在这种情况下,RAP 接口将向自适应控制器通知 XML 业务到达,以便自适应控制器可以从配置存储器(块 6204)取回相应的配置,配置存储器可以是内部或外部存储器或其组合。自适应控制器然后可以向 RAP(块 6206)提供配置,并且在 XML 数据准备好进行处理并在数据缓冲区或外部存储器中准备好以供 RAP 取回时启用此配置。同样,根据可以在 TCP/IP 处理器核心中编程的策略和所接收的网络业务,RAP 块可以被配置为新的硬件具体化,以便根据所接收业务和策略的特性来动态支持特定应用、服务或功能或其组合。

根据策略和/或所接收数据的特性,TCP/IP 处理器核心还可以选择将所接收数据传送到主机处理器而不使其通过 RAP 以进一步处理。因此,如果尚未针对特定操作或功能或策略或应用实现硬件配置,因为操作或策略或功能或应用可能不经常使用并且因此不会导致性能问题,或者由于成本原因或任何其他业务或其他原因而尚未分配资源以开发加速支持,则可以在主机上执行这些操作。当这些操作实现为运行时自适应配置时,可以将此配置提供给自适应控制器以便它可以根据需要针对此操作动态配置 RAP 块。TCP/IP 处理器然后还被通知标识此类操作并使它们通过 RAP 块。

使用此类技术，可以在一段时间内加速更多的应用而无需更改或添加任何硬件加速器。可以在用户或管理员的控制下，使用非常类似于针对安全策略部署和管理所示的机制，在运行时自适应处理器上部署新的策略或应用或服务或操作。因此，中央管理员可以根据需要以及在需要时有效地将新的配置部署到使用本申请的运行时自适应协议处理器的系统中。同样，用户或管理员可以根据系统更改的需要或使用来删除或更改 RAP 支持的功能或策略或应用。例如，使用本申请的运行时自适应协议处理器的系统最初可以用于 XML 业务，但是其使用可以更改为支持基于 IP 的语音应用并且可能不需要 XML 加速，转而需要某些其他基于 IP 的语音加速。在此情况下，用户或管理员可以从一个或多个特定系统中更改、添加或删除可选硬件支持的配置。图 56 到图 59 中所示的中央管理器/策略服务器流、中央管理器流、规则分发流、控制平面处理器/策略驱动器流等适用于对运行时自适应配置进行管理、部署、更改、监视等，并且可以进行类似于如下所述的相应更改。例如，可以将运行时自适应配置创建流添加到安全策略创建流。可以从其他供应者提供新的配置并且用户只需选择要部署的相关配置。配置分发流可以类似于规则分发流，其中用于支持配置（多个）的策略可以分发到 TCP/IP 处理器核心快，而配置可以分发到相关系统的自适应控制器或主机系统上的驱动器或配置控制过程或它们的组合。因此当在此环境中使用时，运行时自适应协议处理器系统可以完美集成到其他企业管理系统中。应用或策略或服务或操作配置可以通过其他方式来分发，例如作为软件更新通过网络或通过大容量存储设备或其他方式来分发。上面的描述是在某个使用环境中提供更新的一种方法，但是如本领域的技术人员可理解的，可以针对每个实施例和使用环境具有多种其他方法来执行相同操作，并且因此不应将其视为限于上面的描述。

还可能需要自适应控制器配置 RAP 块以便对从网络发出的数据进行处理。在此情况下，可能需要 RAP 块在将数据传送到 TCP/IP 处理器核心以通过网络将其发送到预定接收方之前对数据进行处理。例如，在 TCP/IP 处理器核心将数据封装在传输和网络标头之前，可能需要对数据执行安全

套接字层 (SSL) 操作。发送此数据的主机驱动器或应用将向自适应控制器通知在将数据传送到 TCP/IP 处理器核心之前对数据执行的操作。这可以通过从主机/结构接口 6207 到自适应控制器 6203 的直接路径进行。自适应控制器然后可以配置 RAP (块 6206) 或其一部分以动态执行所请求的操作并使 RAP 对数据进行处理。RAP 操作完成后, 它可以向自适应控制器通知操作完成, 然后自适应控制器可以与 TCP/IP 处理器核心协作, 以便 TCP/IP 协议处理器进行相应的协议处理、标头封装等之后将此在途中的数据发送到其目的地。RAP 6206 可以通过数据缓冲区 (块 6209) 或通过使用存储器接口 (块 6205) 将数据排队到存储器中来将已处理的数据传送到 TCP/IP 处理器核心。因此本申请的运行时自适应 TCP/IP 处理器可以被配置为在由 TCP/IP 处理器核心处理之前或之后, 对进站以及出站数据进行处理。

可以通过物理、成本、性能、功率或其他约束限制运行时自适应处理器 6206 的大小或能力。还可以在自适应 TCP/IP 处理器上提供 RAP 扩展接口 (块 6211) 以将 RAP (块 6206) 连接到一个或多个提供运行时自适应处理器功能的外部组件。因此所述解决方案可以使用 RAP 扩展接口 6211 扩展到更大的大小或功能或能力。RAP 扩展接口包括将 RAP 无缝扩展到一个或多个外部组件所需的所有必要控制、路由、数据、存储器接口总线和连接。

图 63 示出了本申请的替代上述处理器的自适应 TCP/IP 处理器。如先前所指出的, 此处理器的 TCP/IP 处理器部分不仅可以是本申请中所披露的体系结构, 而且还可以是固定功能实施方式, 或可以作为硬件状态机实现或可以支持部分协议卸载能力 (例如支持以硬件进行的快速路径处理), 其中控制平面处理以及会话管理和控制可以位于单独的控制平面处理器或主机处理器或上述各种备选体系结构的组合中。所选的 TCP/IP 处理器核心体系结构还可以包括安全性或 RDMA 或其组合的功能。此外, 通过使用适合协议的处理器核心代替所述 TCP/IP 处理器核心, 自适应 TCP/IP 处理器体系结构可以用于 TCP/IP 之外的协议, 如 SCTP、UDP 或其他传

输层协议。图 63 的备选自适应 TCP/IP 处理器示出了运行时自适应处理器（块 6311）、自适应控制器（块 6310）和配置存储器（块 6312），它们更加密切地集成在 TCP/IP 处理器体系结构中以创建运行时自适应 TCP/IP 处理器。RAP（块 6311）、自适应控制器（块 6310）和配置存储器（块 6312）提供的功能非常类似于图 62 中相应块的功能。RAP 接口（块 6210）或存储器接口（块 6205）或数据缓冲区（块 6209）的功能可以相应地由块 6310、6311 或 6312 或其组合提供。此功能还可以分布在 TCP/IP 处理器元件内。此体系结构还可以提供类似于块 6211 的 RAP 扩展接口以提供 RAP 可伸缩性，尽管未在图 63 中示出此类块。此版本的自适应 TCP/IP 处理器还可以像图 62 中的处理器那样运行，并且还可以被配置为在由 TCP/IP 处理器核心块处理之前或之后，对入站以及出站数据进行处理。

图 64 示出了本申请的运行时自适应处理器。运行时自适应处理器包括可以被动态更改以映射各种需要加速的硬件功能的计算逻辑和互连资源。计算逻辑块可以使用诸如用于精细控制的组合块之类的 FPGA 实现或可以包括一个或多个简单可编程处理器、ALU 和存储器，它们可以被配置为在给定时刻提供特定硬件功能，然后可以被动态更改以支持新功能。可以通过配置控制器根据使用 RAP 的需要进行功能的动态更改。例如，计算块（多个）可以被设置为在选定时段内为到计算快的入站数据提供加法运算，但是随后当创建新的具体化时，所提供的运算可以被选择在新具体化期间为减法运算。可以由相应的配置控制器进行用于新具体化的新运算的选择。因此，由计算块提供的功能可以动态更改为其他功能，而某些其他计算块可以继续提供其选定运算。计算块功能更改可能花费一段时间，可能如一个时钟周期那样短或为多个时钟周期或其他周期。可以根据需要同时或以其他方式执行一个或多个计算块的动态自适应。互连资源的实现方式还可以类似于 FPGA 的可重新配置路由资源的实现方式。图 64 将本申请的运行时自适应处理器体系结构示为分层计算逻辑，所述计算逻辑称为计算群集（块 6401（1）到 6401（Z）），使用包括路由资源 6407（a）到 6407（n）的路由网络进行互连。这些路由资源使用群集间路由交换机 6403 互

连。群集间路由交换机可以被动态配置为在各个计算群集之间提供高度可编程互连，从而创建硬件的不断变化的具体化。计算群集可以由与图 62 的自适应控制器（块 6203）协作的全局配置控制器（块 6405）单独配置以动态适应 RAP。全局配置控制器与配置存储控制器（块 6406）以及图 62 的配置存储器（块 6204）和自适应控制器 6203 协作，以便检索用于动态更改单独计算群集配置和群集间路由交换机的配置信息以进行互连配置。输入/输出接口和控制器（块 6404）用于连接运行时自适应处理器与图 62 的自适应处理器（块 6203）、数据缓冲区（块 6209）、RAP 扩展接口（块 6211）或主机/结构/网络接口（块 6207）。全局存储器和控制器（块 6402）为计算群集提供全局存储器，并且还提供控制器以便与外部存储器接口（块 6205）连接。计算群集（块 6401（1）-6401（Z））内部的计算逻辑可能需要除每个群集内部之外的存储器。全局存储器块和控制器可以满足此需要。此图示出了多个充当不同角色的互连元件。互连通道 6407（a）到 6407（n）是路由通道，用于将每个群集连接到群集间路由交换机以便为每个群集启用多向连接能力，以从 RAP 的其他群集获取信息。互连通道 6408（a）到 6408（n）为计算群集提供存储器互连资源，以便这些群集获取对全局存储器的访问。这些存储器通道可以在列中的群集之间共享或存在多个到存储器的路径，列中的多个计算群集可以同时使用这些路径以便向存储器写入数据或从中读取数据。互连通道 6409（a）到 6409（m）是配置通道，允许全局配置控制器向计算群集发送配置信息以及从计算群集接收事件信息或其他信息，这些信息可用于更改给定群集或某些群集的配置或所有 RAP 群集配置。上述互连通道体系结构和实施方式可以使用宽总线、高速串行互连或其他实施选择实现。此图并未指示或暗示特定选择或拓扑。

可以配置图 64 的运行时时自适应处理器以便可以将计算阵列分成部分区域，其中每个区域可以被配置为执行特定硬件操作。例如群集 6401（1）、6401（2）可以形成一个区域，而群集 6401（3）到 6401（M）可以形成另一个区域，而某些其他群集还可以形成另一个区域。某些区域可以被互连为流水线级，如映射到运行时自适应处理器的硬件功能所要求的那样。映

射区域可以彼此互连或可以对独立数据或流进行操作，如映射操作所需的那样。所述区域均可以动态适应处理要求的变化需求。所述区域可以非常精细，也可以仅包含部分计算群集。因此本申请的运行时自适应处理器动态适应于非常精细的级别以满足所需处理的需求。

图 65 示出了本申请的计算群集。计算群集包括提供计算逻辑的计算元件 (CE) (块 6501 (1) 到 6501 (Z))。CE 可以包括类似 FPGA 的组合逻辑块和互连资源，或可以是根据编程指令提供给定硬件功能的具有 ALU 和存储器的简单可编程处理器。可以通过更改对输入数据或存储数据或其组合执行的指令来动态配置 CE 以便执行新的硬件功能。所述处理器可以通过组合逻辑支持硬连线指令的简单处理器，其可选择组合逻辑中配置的所需硬件操作。所述处理器可以是更复杂的处理器，其中硬件配置可以选择通过 ALU 执行的指令以及其他提供虚拟硬件具体化/配置的功能资源。具体化还可以包括多个通过形成更复杂配置的资源执行的指令。可以在 CE 中编程多个具体化，并且可以动态选择特定具体化，提供非常灵活的硬件体系结构。CE 可以根据配置选项粒度的需要提供逐位操作，以及针对位组（如 4 位、8 位、16 位分组）的操作。此位分组可以是实施方式选择，其中可以选择较大或较小的分组而不偏离本申请教导的原理。群集配置控制器（块 6507）与图 64 的全局配置控制器（块 6405）交互以为每个 CE 选择特定具体化。互连通道 6502 (1) 到 6502 (N) 将来自配置控制器的配置信息提供给 CE，并且将来自 CE 的任何执行信息或事件或具体化更改请求或其组合提供给群集配置控制器。此信息可以用于引导配置控制器与全局配置控制器 6405 和/或图 62 的自适应控制器 6203 一起映射不同具体化的流程。此信息可以用于创建操作的流水线级，其中某个计算群集的不同部分或多个计算群集提供多个操作级。互连通道 6503 (1) 到 6503 (M) 提供群集存储器和控制器（块 6506）与 CE 的连接性。可以具有多个到群集存储器的并行路径，从而允许对所示存储器的不同区域同时进行多个访问。给定通道上的 CE 全部可以共享此通道或每个通道可以同时具有多个到存储器的路径。作为实施方式选择，群集存储器可以是单个

存储器阵列或可以是多个存储器阵列。群集存储器还可以通过诸如 6508 以及图 64 的 6408 (a) 到 6408 (n) 之类的通道连接到图 64 的全局存储器和控制器 (块 6402)。全局存储器和群集存储器都可以通过主机/结构/网络接口 6207 或自适应控制器 6203 或存储器接口 6205 或其组合进行访问, 以便单独或作为一组单元读取或写入存储单元以执行初始化或其他用途 (如 DMA 访问、测试等)。CE 还可以提供与其下一个邻居的连接性, 如图 65 中的箭头所示。并非所有指示的相邻连接都必须存在。这可以是实施方式选择。这些连接允许 CE 向其邻居发送或接收输出或输入数据、标志、异常条件等信息或其组合。用于 CE 的选定具体化将确定使用哪些输入检索所需信息以进行处理。来自 CE 的输出还可以被选择为连接到群集路由交换机 6506, 此交换机然后在群集内的各 CE 之间提供选定连接性, 以及通过连接到群集间路由交换机 6403 和诸如 6407 (a) 到 6407 (n) 之类的互连通道来提供与其他计算群集的连接性。可以配置群集路由交换机以便借助与互连通道 6506 的连接, 通过群集配置控制器 6507 进行相应的互连。

图 66 示出了使用本申请的教导的安全解决方案。该安全解决方案包括中央管理器、网络、一个或多个线卡和安全芯片。中央管理器是一组位于中央管理系统中的由 IT 管理者用于创建、部署和监视安全规则的功能模块。中央管理器模块类似于图 56 中所示的中央管理器模块, 在以下说明中可互换地使用它们。这些模块可以位于用于管理整个网络的同一组系统上或可以位于独立系统上。块 6601 被示为表示可以针对网络部署的网络管理应用和安全性应用。IT 管理者使用这些应用创建其安全性规则或策略。中央管理器提供应用编程接口 (API) (块 6602), 此接口提供到 6601 的安全和管理应用的统一接口以使用本申请的分布式安全系统。API 与网络层规则引擎 (块 6603)、应用层规则引擎 (块 6604)、存储区域网络规则引擎 (块 6605) 或其他应用特定的规则引擎 (块 6619) 或其组合连接。这些规则引擎为它们表示的一种或多种特定规则类别提供 API 支持功能。它们还可以提供预先配置的规则模板, IT 管理者可以针对其特定需要通过填充规则的相关字段使用这些模板。例如, 可以具有这样一组规则, 所述规则

拒绝向网络地址不是本地地址的所有用户发出的连接请求，或拒绝所有外部连接向特定端口（如端口 80）发出的请求等或它们的组合。这些规则引擎组装规则并将它们提供给规则编辑器（块 6606），以编辑规则以便分发到安全节点。编辑器使用节点能力和连接性数据库（块 6617）以编辑适合节点的规则和操作。将编辑后的规则存储在已编辑规则数据库（块 6618）中。规则分发引擎（块 6607）使用类似于图 57 和图 58 中所示的中央管理器流和规则分发流将规则分发到相应的节点。可以将安全规则分发到如图 58 中所示的主机处理器或控制平面处理器或分发到下面所述的控制处理器和调度器（块 7103）或其组合，具体取决于节点能力。可以根据特定节点的能力，通过网络使用采用专用或标准协议的安全链路或不安全链路来分发规则。网络可以是局域网（LAN）、广域网（WAN）、城域网（MAN）、无线 LAN、存储区域网络（SAN）或系统区域网络或部署的其他网络类型或其组合。网络可以基于以太网、基于网际协议或基于 SONET 或基于其他协议或它们的组合。利用监视接口（块 6609）以及事件记录引擎和数据库（块 6608）从各个被监视节点收集各种安全性和/或管理事件，以查看如规则所定义的安全性违反或其他情况。这些块表示上述的中央管理器块 5613、5614、5615 和 5616 并且提供类似的功能。监视引擎可以提供如针对块 5613 所述的分析能力或可以与块 6610 所示的分析和报告应用（多个）协作，以便根据问题性质及其严重程度，在请求时或如 IT 管理者编程的那样，自动向 IT 管理者提供安全违反或违背或符合或其他问题的智能报告。中央管理器模块还可以部署到网络节点系统（例如交换机或路由器）的本地，并且在系统的控制和管理软件内运行。它可以用于部署和监视各种线卡或加速器卡或其他提供系统安全能力的卡的本地规则。在此情况下，用于传送规则的网络可以是特定系统的本地总线或系统区域网络或其组合。

安全解决方案包括线卡，所述线卡可以结合本申请中所披露的安全性处理器、SAN 协议处理器、TCP/IP 处理器或运行时自适应协议处理器或各种其他处理器。线卡配置和体系结构可以随特定系统和应用而不同。本申请中示出了三种类型的线卡体系结构，a）直通（flow-through）b）后

援, 以及 c) 加速器卡, 以说明本申请的处理器的使用模型。图 68、图 69 和图 70 使用基于安全性处理器的系统示出了这些配置, 但它还可以基于本申请的其他处理器。块 6612 和块 6613 示出了其中两种类型的卡配置。这些卡中示出的安全性处理器是本申请中所披露的安全性处理器。可以根据处理器中结合的功能而创建安全性处理器的各种不同变型。块 6614 和块 6615 示出了两个版本的此类安全性处理器。块 6614 示出了包括至少一个与运行时自适应处理器连接的内容搜索和规则处理引擎的安全性处理器核心。此处理器类似于图 71 中所示的处理器并在下面详细说明。块 6615 示出了与 TCP/IP 处理器或协议处理器连接的块 6614 的安全性处理器, 以便提供更多可在作为安全性处理器的安全节点中使用的功能。还可以通过从块 6614 中删除运行时自适应处理器和关联逻辑来创建功能减少的安全性处理器(未示出), 以提供基于内容搜索和规则处理引擎的安全性处理器。安全性处理器的选择可以取决于部署安全性处理器的系统、系统支持的功能、解决方案成本、性能要求或其他原因或它们的组合。安全性处理器可以使用一个或多个端口以连接到外部存储器(块 6616), 外部存储器可以用于存储所需的规则信息或其他中间数据或分组或其他信息以执行安全性处理所需的各种功能。所述存储器可以是各种类型, 例如 DRAM、SDRAM、DDR DRAM、SRAM、RDRAM、FCRAM、QDR SRAM、DDR SRAM、磁存储器、闪存或其组合或此类存储器技术的将来派生物。本申请中所披露的发明实现所示体系结构的许多变型, 并且本领域的技术人员可以理解, 可在实施例中进行更改而不偏离本发明的原理和精神。

图 67 示出了安全性解决方案编辑器流。如上所述, 安全性规则可以具有各种类型, 例如应用层规则(块 6701)、网络层规则(块 6702)、存储区域网络规则(块 6703)或应用特定规则(块 6619)或其组合。如图 67 中所示, 应用层规则包括可以以特定语言或诸如正则表达式之类的标准表示或其组合表达的基本字符串搜索规则。可以使用正则表达式表示通常需要在分组内部深入搜索字符串的应用层规则。应用层规则或网络层规则或 SAN 规则或应用特定规则的类型可以随可部署规则的特定节点、使用规则

的组织或实体、要防御的安全性威胁或其他用途或其组合而不同。此图示出了可以根据使用模型而创建的各种类别的规则。可以由反垃圾邮件软件供应商或使用安全系统的实体或提供安全解决方案的供应商或其他第三方或它们的组合来创建这些规则。可以具有用于防御垃圾邮件的应用层规则。这可以包括已经使用垃圾邮件或有害消息的知识创建的规则。例如，规则可以是在包括标头和消息的电子邮件内任何位置，在任何入站电子邮件内部搜索诸如“receive million USD”之类的消息。此规则可以使用诸如“.*receive million [U][u][s][S][D][d]”之类的正则表达式表示，此表达式将检测感兴趣的消息，即“receive million USD”，但是还可以检测此消息的变型，其中 USD 并非全部大写（例如 usd 或 UsD 或 usD 等）。此规则中的前导字符“.*”表示在所接收数据分组内的任何位置搜索消息。可以定义类似上述的一组规则以形成反垃圾邮件规则集。可以在发现新的垃圾邮件类型或方法时更新这些规则并且可以使将这些规则与不断发展的威胁保持同步。同样，可以开发一组用于执行病毒扫描功能的规则以检测各种已知病毒。反病毒规则通常是类似于上述的特征匹配或模式匹配规则。可以在消息或文件中的特定位置查找病毒特征并且可以使用类似方法描述病毒特征。根据已知病毒特征定义一组反病毒规则以检测已知病毒。当新的病毒或蠕虫病毒变为已知时，可以更新反病毒规则以防御这些病毒。然后将如前所述的那样通过安全编辑器流编辑这些规则并将其分发到所有感兴趣的节点。部署这些规则后，可以对安全节点编程以执行与反垃圾邮件或反病毒规则匹配对应的操作，以便拒绝访问发出消息的特定节点或断开连接或标记与 IT 管理者的会话或规则定义的其他相应操作。中央管理器模块为 IT 管理者提供了当满足类似于上述的特定条件时定义此类操作的能力。所述操作可以包括以下操作中的一个或多个：断开连接，拒绝访问，报告违反，通知网络管理者，允许访问但记录违反以供未来分析，或将源节点隔离到特定虚拟 LAN 或将连接转移到某些其他节点或其他适用的类似操作。

同样，可以具有许多其他应用层规则类别。例如，可以定义用于可根

据需要管理电子文档或媒体或内容的所有者的数字版权的规则。此类规则的定义方式还可以类似于上述特征或模式匹配或字符串匹配规则的定义方式。这些规则可以标记到内容内部的特定数字版权签名的匹配，其然后可用于参考数字版权数据库，该数据库可指示是否允许所有者对数字内容进行此类访问或使用。数字版权所有权数据库可以位于与安全性处理器和控制处理器（如下面所述的块 6809 或块 7103）关联的存储器中，可以参考此数据库以确定是否存在有效的所有权，以及如果不存在，应根据所定义的规则采取何种特定操作。可以由执行数字版权签名匹配的特定节点中的某些其他设备或处理器执行数字版权确认。确定在何处执行此类分析功能可以取决于特定系统使用模型和系统设计选择。还可以创建一组用于数字版权管理的规则，作为安全性处理器的应用层规则的一部分。

即时消息传送（IM）在其个人以及公司使用方面已经获得巨大成功。即时消息传送可以针对各种行业（如金融行业）进行管理以保留供将来参考，并像其他模式通信（如电子邮件）那样，也受到垃圾信息的攻击。因此某些组织可以创建专门针对即时消息传送的规则，以避免持续发生错误使用的情况或防止用户受到有害垃圾信息的攻击或用于组织认为适当的其他原因。即时消息传送存在的一个问题是任何级别的控制都必须以流的形式执行而不会在通信中产生延迟。因此可能需要本申请的基于硬件的安全实施以监视 IM。这些规则类似于上述的其他应用层规则，并且可以使用类似方式（如使用正则表达式定义消息搜索字符串）来创建。

FBI 和其他部门的最近调查已经发现 70% 以上对信息技术的攻击来自组织内部。因此需要开发一类安全设备和规则以防止受到此类攻击的破坏性影响。这些规则被定义为外泄检测规则。可以创建外泄检测规则以使用企业外围网络检测组织的保密或专有或敏感信息的有意或无意泄漏。例如，软件公司可能需要保护其核心软件源代码以防止被意外或恶意泄漏给未经授权获取此代码的个人或实体。因此组织可以创建一组规则，这些规则可以在所有出站消息内搜索特定字符串或段落或代码模块或其他相应信息，并对它们进行标记或防止发送这些信息。还可以使用安全性编辑器流编辑此

类规则并将其分发到相应的一个或多个节点。例如，可以定义规则以便在从组织出站的任何发送消息中搜索“Top Secret”词组并标记此类消息以便 IT 管理者进一步检查，或断开此类连接并通知用户或其他负责人员。可以定义正则表达式规则“.*Top Secret”以便在消息中的任何位置搜索此词组。还可以将此类规则创建为应用层规则，然后可以编辑所述规则并将其分发到相应的节点以检测和实施外泄检测安全功能。

IT 管理者可以通过应用层规则或网络层规则或 SAN 规则或应用专用规则或其他规则来创建多类规则，并将一类规则部署到一类安全节点，而将不同的一类规则部署到另一组安全节点。例如管理者可以创建部署到通过本申请的教导启用安全性的网络的交换机和路由器的特定应用层规则（如反垃圾邮件或反病毒规则和网络层规则），以及另一组用于保存绝对机密的关键信息的敏感服务器的规则（如外泄检测规则和网络层规则）。可以创建可根据组织内的职责部署的不同规则集。例如，在制造部门内部署的安全节点可以获取一组规则，而工程部门中的安全节点可以获取不同的一组规则。可以将创建用于不同类型设备或不同设备用户的不同规则组或节点特定规则或其组合用作在组织内创建普遍且分层的安全性的过程。

同样可以具有检测或标记对特定 Web 地址或 URL 或其他保密信息（如包括客户信用卡号或健康信息或财务报告等的客户信息）的访问的应用层规则，可以使用这些规则创建不同的一组应用规则，如块 6701 中所示。随着在组织内以及通过因特网使用基于 IP 的话音的解决方案的增加，安全性威胁也在增加。因而可能有必要创建专用于 VOIP 的规则，例如可能需要检测和标记恶意连接，或可能不允许 VOIP 业务流出组织边界，或检测通过 VOIP 连接进入组织的病毒，或通过加密 VOIP 业务创建其保密性等。VOIP 规则也可以使用相同的应用层规则引擎来创建并在网络中的相应节点处检测规则匹配。当通过 VOIP 规则匹配检测到 VOIP 业务时，可以使用下面所述的运行时自适应处理器（块 7102）为此类业务提供加密或解密服务。同样，还可以开发其他应用专用规则并在中央管理器模块中提供这些规则，使用图 67 中所示的编辑器流对这些规则进行编程、编辑并分发到

网络中的安全节点。

网络层规则（块 6702）可以包括各种针对网络和网络传输层的规则。这些规则类似于图 60 中所示的规则。这些规则可以包括 IP 级地址规则、协议端口规则、协议专用规则、面向连接方向的规则等。这些规则可以以专用语言或使用正则表达式描述。在基于 TCP/IP 的网络中，这些规则主要是基于 TCP 和 IP 标头字段的规则，其中可以针对源地址或目的地地址或地址范围或端口号或协议类型或其组合来定义匹配。同样可以具有专门针对可传输组织的关键信息资产的存储区域网络的规则。这被示为不同类别的规则，但可以包括存储网络的网络层规则、应用层规则等。可以具有针对特定逻辑单元号（LUN）或区域（源/目的地地址组）或逻辑或物理块地址等的规则。也可以以特定语言或使用正则表达式作为字符串或数据模式来表示这些规则。

图 67 的安全性解决方案编辑器允许 IT 管理者如上述那样创建不同类型的安全规则并使他们能够创建分层和/或普遍的安全性模型。将特定节点的特征提供给编辑器流，这些特征包括安全性能力存在、规则传送方法、支持的规则库的大小、节点的性能量度、部署位置（例如 LAN 或 SAN 或其他位置）等。编辑器流然后使用此知识来编辑 IT 管理者创建的规则集（多个）中的节点专用规则。编辑器包括用于分析要提供给词法分析生成器（块 6705）的规则的分析器（块 6704），词法分析生成器分析规则并创建用于分析内容的规则数据库。规则分析器可以从规则文件或直接从命令行或其组合中读取规则，具体取决于规则引擎的输出。将分析特定节点的规则以标识用于描述规则的语言特定标记或正则表达式标记。分析器然后将标记提供给词法分析生成器。词法分析器处理入站标记并生成代表用于分析内容的规则的非确定性有限状态自动机（NFA）。然后词法分析生成器将 NFA 转换为确定性有限状态自动机（DFA）以启用确定性规则状态处理。现代编辑器开发人员非常熟悉创建 NFA 和 DFA 的过程。但是，词法分析生成器针对硬件词法分析器使用的规则创建表示 DFA 状态的各种表和状态转移表，而不是如针对编辑器那样生成词法分析软件。一种查看规

则的方法是它们定义了用于识别内容的语言。词法分析器硬件或下面所述的内容搜索和规则处理引擎（块 7106）使用这些表来分析提供给本申请的安全性处理器的数据流。正则表达式规则可以被视为定义状态转移表。例如，如果使用正则表达式“help”搜索字符串“help”，则正则表达式的每个字符可被视为代表一个状态。可以具有开始状态 s_0 以及字符特定状态 $s_1(h)$ 、 $s_2(e)$ 、 $s_3(l)$ 和 $s_4(p)$ ，其中 $s(x)$ 代表字符 x 的状态。还可以具有诸如 s_{err} 之类的错误状态，当不满足相应的转移条件而终止搜索时可能进入此状态。当硬件词法分析器分析输入流时，将激活此状态机，当遇到第一个“h”时，状态机将达到 s_1 。现在如果流中的下一个字符是“e”，则状态机将转移到 s_2 。因此如果遇到字符串“help”，则状态机将达到状态 s_4 。状态 s_1 到 s_3 是接受状态，表示它们将搜索继续到下一状态。对于此字符串，词法分析生成器将状态 s_4 标记为终端状态。这些状态在接受表中被标记为接受或终端状态。当比较达到终端状态时，可以指示与特定规则的匹配。任何根据规则匹配而需要采取的操作将在匹配/操作表中创建为操作标记或指令，然后内容搜索和规则处理引擎（块 7106）使用此标记或指令执行特定操作，或将匹配和操作信息转发到控制处理器（块 7103）以采取相应的规则特定操作。但是，如果只存在部分规则匹配（例如，如果输入内容包括字符串“her”），则规则处理硬件将在遇到“he”时进入状态 s_2 ，但是分析完“r”之后，将指示错误以表示没有规则匹配并且输入流处理将从该点开始回到初始状态 s_0 。虽然针对每种状态使用单个字符匹配给出了上述说明，但是可以同时分析多个字符以加速硬件分析。例如，词法分析生成器可以创建能在每个状态转移 4 个字符的表，从而使内容搜索速度提高三倍。词法分析生成器将创建字符类表（块 6706）、下一状态查找表（块 6709）、状态转移表（块 6707）、接受状态（块 6708）和匹配/操作表（块 6710），然后将它们存储在已编辑规则数据库存储装置（块 6711）中。通过将创建一组类似状态转移的字符压缩到一组状态来创建字符类表以简化表示。状态转移表包括 DFA 表中的状态行，并将压缩后的字符类作为列以查找下一状态转移。下一状态表用于从 DFA 表示的状态

机中的当前状态索引到下一状态。这些表存储在与本申请的安全性处理器关联的片上和片外存储器中。本申请的编辑器使用节点特性和连接性数据库逐节点地创建规则。如果特定规则或规则大小与特定节点的能力不匹配,编辑器将向 IT 管理者指示错误以便管理者更正规则或规则大小。将从如块 6713 所示的节点特性和连接性数据库检索此信息。

规则分发引擎(块 6712)遵循图 57 和图 58 中所示的中央管理器和规则分发流。可以将安全规则分发到如图 58 中所示的主机处理器或控制平面处理器或分发到下面所述的控制处理器和调度器(块 7103)或其组合,具体取决于节点能力。可以根据特定节点的能力,通过网络使用采用专用或标准协议的安全链路或不安全链路来分发规则。

图 71 示出了本申请的安全性处理器。安全性处理器包括协处理器或主机总线接口(块 7101)、控制处理器和调度器(块 7103)、至少一个内容搜索和规则处理引擎(块 7106)、下一状态存储器(块 7110)、匹配/操作表存储器(块 7111)、字符类表存储器(块 7107),以及接受和状态转移存储器(块 7108)。安全性处理器还可以包括分组缓冲区(块 7104)、存储器控制器(块 7112)、运行时自适应处理器(块 7102)、自适应控制器(块 7105)和配置存储器(块 7109)。可以通过使用协处理器或用作数据接口的主机接口控制器、控制处理器和调度器、至少一个内容搜索和规则处理引擎、下一状态存储器、匹配/操作表存储器、字符类表存储器、接受和状态转移存储器,以及存储控制器来创建安全性处理器版本。在规则数少到足以置于片上存储器内的系统应用中,可能不需要存储器控制器。此类处理器可以执行所有的内容搜索任务;但是它可能无法提供针对性的应用加速,这可以使用包括运行时自适应处理器的安全性处理器实现。

控制处理器和调度器(块 7103)与规则分发引擎(块 6712)通信以便在启动内容检查之前接收相应的数据表。它将所接收的状态信息存储到其相应专用存储器中。块 6706 中的字符类表存储在存储器(块 7107)中。状态转移和接受表(块 6707 和 6708)存储在其由块 7108 表示的相应存储器中。块 7108 由于性能原因也可以是两个或更多个单独存储器,但在图中

由一个块表示。块 6709 中的下一状态查找表存储在下一状态存储器（块 7110）中。块 6710 中的匹配/操作表存储在其存储器（块 7111）中。这些表可能大于安全性处理器片上可用的存储器，并且可以存储在由存储器控制器（块 7112）访问的一个或多个外部存储器中。可以存在多个到存储器的端口以加速访问外部存储器中存储的数据表。这些存储器可以具有各种类型，如 DRAM、SDRAM、DDR DRAM、SRAM、RDRAM、FCRAM、QDR SRAM、DDR SRAM、磁存储器、闪存或其组合或此类存储器技术的将来派生物。对于大多数应用，下一状态表和操作表可能需要在片外，而其他表可以在片上维护，具体取决于规则的大小和数量。规则分发引擎将表提供给控制处理器和调度器（块 7103），并且在其相应存储器中设置这些表之后，安全性处理器可以开始处理数据流以执行内容检查并标识潜在的安全规则匹配或违反。通过协处理器/主机接口控制器接收安全性处理器状态配置信息。本申请的安全性处理器可以部署在各种配置中，如图 69 中所示的后援配置或图 68 中所示的直通配置或图 70 中所示的加速器适配器配置，以及本领域的技术人员可以理解的其他未示出的配置。在后援或加速器适配器配置中，本申请的安全性处理器在主处理器的控制下，主处理器可以是网络处理器或交换机处理器或 TCP/IP 处理器或分类处理器或转发处理器或主机处理器等，具体取决于此类卡所在的系统。在此类主处理器的控制下，与规则引擎通信以接收包含配置信息的分组的控制处理器和调度器接收配置信息并将其继续传送到安全性处理器。配置完成后，主处理器将分组提供给需要使用协处理器或主机接口执行内容检查的安全性处理器。协处理器或主机接口可以是诸如 PCI、PCI-X、PCI-Express、RapidIO、HyperTransport 或 LA-1 或 SRAM 存储器接口之类的标准总线或专用总线。总线带宽应足以保持内容搜索引擎以其峰值线路速度运行。在后援或加速器配置中，安全性处理器可以是主处理器空间中的存储器映射或 IO 映射设备以便其接收分组和其他配置信息。安全性处理器可以由主处理器进行轮询，或可以为主处理器提供门铃或中断机制以指示它何时完成处理给定分组或何时发现对编程规则的匹配。控制处理器和调度器（块

7103) 以及块 7101 与主处理器协作以提供上述功能。控制处理器和调度器将入站分组存储到分组缓冲区 (块 7104), 并调度分组以便在内容搜索和规则处理引擎变得可用时对其进行处理以分析内容。调度器维护由特定引擎处理的分组的记录, 并在处理分组后, 它将通知主处理器。块 7106 的内容搜索和规则处理引擎在发现规则匹配和与此规则关联的如在匹配/操作表中编程的操作时, 将通知控制处理器和调度器。控制处理器转而可以将此信息发送到主处理器, 其中主处理器可以针对规则指示的分组执行特定操作。所述操作可以是多种操作中的一种, 如先前所述的丢弃分组或断开连接或通知 IT 管理者等。当安全性处理器包括诸如块 7102 之类的运行时自适应处理器时, 控制处理器和调度器可以调度针对通过块 7102 的分组的操作。控制处理器将与自适应控制器 (块 7105) 协作以针对所需操作选择特定的处理器具体化。例如, 需要在分析之前解密的分组可以在内容搜索引擎分析之前被调度到自适应处理器。自适应处理器对分组进行解密后, 块 7103 便将分组调度到块 7106。但是, 当内容搜索引擎发现匹配或搜索引擎处理分组而没有出现任何问题之后, 运行时自适应处理器便可以对分组进行处理。例如, 没有发现问题之后可能需要对分组数据进行加密。控制处理器和调度器以根据操作需要定义的适当顺序将分组调度到运行时自适应处理器。运行时自适应处理器 (块 7102)、自适应控制器 (块 7105) 和配置存储器 (块 7109) 类似于图 62、63、64 和 65 中的那些块。运行时自适应处理器及相关块提供与连接到安全性处理器的控制处理器和调度器的相应逻辑增强类似的功能。如果部署了相应的自适应配置, 运行时自适应处理器可用于为分组提供压缩和解压缩服务。运行时自适应处理器还可用于 VOIP 分组, 为这些分组提供相关硬件加速服务, 如 DSP 处理或加密或解密等。

安全性处理器还可能需要在源和目的地之间的连接中提供跨多个分组的检查能力。控制处理器和调度器 (块 7103) 也提供此类功能。控制处理器可以将内容搜索和安全处理引擎的内部处理状态存储在连接数据库中, 可以在控制处理器中的片上存储器或片外存储器中维护此数据库。在下面

所述的直通配置中，当对应于连接的分组由主处理器提供给控制处理器和调度器或位于入站业务中时，控制处理器和调度器将查找给定连接的执行或分析状态。连接 ID 可以由主处理器创建并与要检查的分组一起提供给安全性处理器，或安全设备可以从分组标头中导出连接关联。在 IP 协议情况下，可以通过使用从源地址、目的地地址、源端口、目的地端口和协议类型导出的 5 元组散列来创建连接 ID。控制处理器和调度器创建并在散列冲突情况下解析连接 ID 之后，它便检索与此连接关联的状态并将状态提供给搜索引擎（块 7106）以开始从此状态进行搜索。此机制用于针对每个连接创建多分组搜索并检测任何跨分组边界的安全违反或威胁。例如，如果具有定义为搜索“Million US Dollars”的规则并且如果此字符串在连接数据传输时出现在两个单独分组中，其中“Million U”出现在一个分组中而“S Dollars”出现在另一个分组中，则如果没有提供本申请的基于连接的多分组搜索机制，就可能不会检测到安全违反，因为每个分组单独都不与规则匹配。但是，当执行多分组搜索时，无论两个分组到达安全节点的间隔时间多长，从连接的一个分组到另一个分组都将保持搜索状态，并且将检测两个分组的字符串并将其标记为连续字符串“Million US Dollars”。

如前所述，本申请的安全性处理器还可以部署在直通配置中。对于此类配置，安全性处理器可以包括两组介质接口控制器端口，如块 7201 和 7213 所示。图 72 中所示的安全性处理器非常类似于图 71 中的安全性处理器；但是，与诸如块 7101 之类的主机或协处理器接口块相比，它具有多个介质接口控制器端口。端口数可以取决于每个端口的线路速度和安全性处理器的性能。入站端口线路速度的总和应与安全性处理器的处理性能匹配以便充分为整体入站业务提供安全检查。如果已知不需要为安全目的检查所有业务，则可以进行有意选择以使用高于处理器能力的线路速度总和。可以根据中央管理器的处理器中编程的连接或会话确定必须检查的业务。因此，图 72 的安全性处理器可以用于为业务提供直通安全检查并可以用于如图 68 所示的直通配置中。可以为诸如交换机或路由器线卡或主机服务器适配器或存储联网线卡或适配器之类的各种类型的系统创建直通配置。在

直通配置中，安全性处理器直接暴露于网络中的业务。因此，中央管理器和规则分发引擎可以直接与安全性处理器的控制处理器和调度器（块 7203 或块 6809）通信。块 6802 的安全性处理器类似于图 72 中所示的安全性处理器，其中没有结合运行时自适应处理器。直通配置中需要解决的一个问题是安全性处理器在业务中引起的等待时间。例如网络交换机或路由器对系统的等待时间性能非常敏感。因此在此类配置中，深度分组检查会显著增加等待时间从而损害系统性能。因此，本发明用于直通配置的安全性处理器提供了块 6807 所示的直通逻辑，此逻辑用于将来自安全性处理器的输入的数据业务以最低等待时间传送到其输出以便支持整体系统性能需要。图 72 的控制处理器和调度器（块 7203）提供直通逻辑并且未单独示出。在直通配置中，一旦发现匹配，安全性处理器就可以在系统内部创建特殊控制分组，其中系统的交换机处理器或网络处理器或其他处理器可以解释这些消息，并在允许使用直通模式的分组退出系统之前对这些分组执行相应的操作。此类协议可以是系统内的专用协议或可以使用可适用于结合了直通安全配置的系统的标准协议。

图 73 示出了另一个版本的安全性处理器，其非常类似于图 71 和 72 中的处理器但具有某些附加功能。由分类/规则引擎（块 7313）、分类/规则数据库（块 7314）和数据库扩展控制器（块 7315）提供附加功能。这些块类似于上述图 20 和 30 中的块。这些块可用于使用基于 CAM 的体系结构提供高性能网络层规则处理。基于三进制 CAM 的数据库还可以用于提供网络标头中特定字段的快速匹配，以便创建散列密钥以进行连接标识和连接状态检索或更新。控制处理器和调度器根据其中编程的规则相对内容搜索和规则处理引擎中编程的规则来确定将分组的哪些部分提供给分类/规则引擎。基于 CAM 的体系结构通常消耗大量功率并因此在其应用中受到限制，除了在极低等待时间情况下需要极高速度以外。内容搜索和规则处理可以以较低功率提供此功能，同时还针对较大规则集（与基于 CAM 的体系结构中的规则集相比）执行搜索。数据库扩展端口（块 7315）可用于使用外部分类/规则引擎扩展 CAM 数据库大小。

图 74 示出了块 7106、7206、7306 的内容搜索和规则处理引擎。内容搜索和规则处理引擎都包括到各种存储器的接口块,所述存储器保存了由图 67 的规则分发引擎分发的规则状态表。此引擎包括从分组缓冲区(块 7104)或图 72 和 73 中的等效块取回待分析分组数据的内容取回(块 7405)。字符类查找(块 7406)、接受/状态转移查找(块 7407)、下一状态查找(块 7408)、匹配/操作查找(块 7409)都从图 71、72 和 73 的相应状态存储器中执行状态数据取回。内容搜索状态机(块 7404)包括用于使用各种表中的数据分析所取回字符的状态机。状态机使用所取回的字符对字符类表进行索引以检索 DFA 状态机的列地址。并行地,状态机使用当前状态作为索引取回状态转移表数据以检索 DFA 状态机的行地址。当开始新搜索时,可以将当前状态初始化为开始状态,否则接下来检索的下一状态将成为状态机的下一次重复的当前状态。然后使用行地址和列地址检索状态机的下一状态。所检索的下一状态索引还用于取回操作标记(如果这是终端状态)。并行执行的接受状态查找用于标识所检索的状态是终端状态还是错误状态还是继续或接受状态。内容搜索状态机(块 7404)高效地遍历上述步骤,直到发现错误或发现匹配或耗尽分组或其组合。对于基于连接的查找功能,将当前内部状态(如地址指针)、当前状态和下一状态等提供给控制处理器和调度器(块 7103 或 7203 或 7303),以便其在连接数据库中维护状态。当调度给定连接的新分组时,将检索内容搜索状态机的所存储内部状态并将其提供给块 7404 以开始处理连接的新分组,好像此分组与连接的先前分组是连续流那样。包括运行时自适应处理器的安全性处理器还可以包括 RAP 命令控制器(块 7403),此控制器连接到自适应控制器(块 7205),以使运行时自适应处理器适应于提供随之发现的匹配和操作标记所需的服务。还可以将操作标记提供给控制处理器和调度器以使其将所分析的分组调度到运行时自适应处理器。自适应控制器可以使用块 7403 提供的命令作为提示或命令以使处理器准备所需的具体化配置信息,如果此信息尚未作为一种具体化存在于运行时自适应处理器(块 7102)中。

如前所述,本发明的安全性处理器可以嵌入具有许多不同配置的系统

中，具体取决于系统环境、系统功能、系统设计或其他考虑事项。本申请在图 68、69 和 70 中示出了三种此类配置。如上所述，图 68 示出了提供直通安全性的网络线卡或适配器中的安全性处理器。在此配置中，安全性处理器可以位于所示介质接口的旁边或位于靠近主机或背板接口（块 6803）的块 6803 之后。此类决策是系统设计决策并且不会从本申请的安全性处理器的使用中排除。如果安全性处理器结合了 TCP/IP 或协议处理能力，则在某些系统中可能不需要块 6803。图 69 示出了网络线卡或适配器的后援安全配置。在此类配置中，存在可以是交换机处理器、网络处理器、转发引擎或分类引擎或其他处理器的主处理器，如块 6903 所示。主处理器与如前所述的图 66 的中央处理器通信，以接收规则并向与安全性处理器协作的中央管理器提供事件。主处理器还可以结合块 6902 和 6904 所示的功能。主处理器同时还可以是 TCP/IP 处理器或可从本申请的处理器实行的其他 IP 处理器变型。

图 70 示出了安全和内容搜索加速适配器。此类适配器可以作为加速器卡插入上述多个联网系统（如服务器、路由器、交换机等）中。此加速器卡上的安全性处理器可以直接或通过如块 7003 所示的桥接器设备连接到主机总线或背板。安全性处理器与系统的主机处理器或主处理器通信以接收要检查的分组或内容并提供结果。主机或主处理器上的驱动器可以执行此与安全性处理器的通信。此类驱动器或运行于主机或主处理器上的其他软件可以与中央管理器通信，以便根据编程的操作接收规则数据库或更新此数据库以向中央管理器提供匹配结果。加速器卡可以包括诸如基于三进制 CAM 的搜索引擎之类的其他设备，这些设备可以用于执行网络层安全功能或连接 ID 检测或散列密钥生成或其他功能或其组合，这些功能可以有助于执行上述的网络层和应用层安全加速功能。

图 71、72 或 73 的安全性处理器还可以用于对文档或数字信息执行内容搜索，并用于创建可用于加速搜索（如 Google 或其竞争对手提供的 Web 搜索能力）的索引。

与使用基于通用处理器的软件相比，使用本发明的安全性处理器执行

此类任务可以显著提高编制索引和搜索的性能。对于此类应用，安全性处理器的控制处理器和调度器可以利用内容搜索引擎和规则处理引擎对提供给其的数据执行关键短语搜索并获取匹配索引。在控制处理器和调度器或处理最终用户的内容搜索请求的系统的其他处理器中运行的进程然后可以使用这些结果创建主搜索索引。然后可以参考所述主索引提供快速和全面的搜索结果。

上述图 71、72、73 中的安全性处理器可以与图 16、17、61 或 62 中的处理器元件相连，以便为本申请的不同版本的协议处理体系结构提供安全能力。例如，块 6615 示出了一个此类变型，其中 TCP/IP 协议处理器与图 71、72 或 73 的处理器相连，以创建另一具有 TCP/IP 处理功能的安全性处理器。可以通过将 IP 存储协议处理能力集成到安全性处理器或将具有 RDMA 能力的 TCP/IP 处理器与图 71、72 或 73 的安全性处理器相连或上述操作的组合，来创建类似的版本。在将安全性处理器编程为搜索块 1703 中使用的分类字段时，还可以使用图 71、72 或 73 的安全性处理器替换分类引擎（块 1703，在上述图 20 和 30 详细示出）。

图 75 示出了正则表达式规则的一个实例。如上所述，RE 从上世纪 50 年代便开始使用并且在许多流行的应用中采用。图中示出了一个小型正则表达式规则集（在本文中也称为规则树），可以使用此规则集分析和解析在 HTTP 中使用的某些构造。可以将 RE 规则（如使用完整规则集为语言或协议等示出的规则）提供给诸如 lex、flex 之类的工具，将使用这些工具创建词法分析器，然后使用所述词法分析器针对特定的语言或协议分析内容。为了便于说明，此图使用 R1 到 R9 来标记规则，它们不代表特定工具的语法。所述规则分为四个规则组：<COMMENT>、<FNT>、<S_TAG> 和 <INITIAL>。如图所示，规则 R1 到 R3 属于 <COMMENT> 规则组，其他类似的规则属于其他规则组。每个规则都包括一个正则表达式以及一个或多个操作/标记，在触发或匹配正则表达式规则时将处理所述操作/标记。例如，R9 是在内容中搜索尖括号（'<'）的 <INITIAL> 组的一部分。如果在内容中找到尖括号，则处理操作/标记。在此规则中，操作包括启动

S_TAG 规则组。

此图假设当内容搜索开始时，首先激活标记为<INITIAL>的规则组，然后操作/标记驱使流移动以便根据内容激活其他规则组。在此图中，当检测到第一尖括号时，将激活是 S_TAG 组的一部分的规则并且 INITIAL 组留待处理。当处理其他内容时，规则将遍历规则树，其中<INITIAL>规则组为规则树的根，从这里开始内容搜索的处理。这是现有技术中创建内容搜索规则的实例。

如上所述，正则表达式可以使用诸如 NFA 或 DFA 之类的 FSA 进行表示。图 76a 示出了正则表达式 $(xy+y)^*yx$ 的 Thompson 构造。Thompson 构造以逐步的方式执行，其中每个步骤都引入两个新状态，因此结果 NFA 所具有的状态数最多为正则表达式中符号和运算符的两倍。FSA 由状态、状态转移以及使 FSA 从一种状态转移到另一状态的符号组成。FSA 包括至少一个开始状态、至少一个接受状态，其中开始状态是 FSA 计算开始的状态，接受状态是当 FSA 发现字符串时所达到的状态。块 7601 表示 FSA 的开始状态，而块 7605 为接受状态。块 7602 表示状态 2，块 7604 表示状态 3。从状态 2 到状态 3 的转换在符号 x (7603) 上触发并表示为两个状态之间的有向边。Thompson 的 NFA 包括' ϵ '转移 7616，它是可以在没有任何输入符号的情况下发生的状态之间的转移。

图 76b 示出了正则表达式 $(xy+y)^*yx$ 的 Berry-Sethi NFA。Berry 和 Sethi 描述了使用称为“标记”正则表达式的技术将正则表达式转换为 FSA 的算法。这种算法产生的 NFA 具有这样的特性：到任意状态的所有转移都来自同一符号。例如，所有到状态 1 (7607) 的转移都来自符号 ' x '。Berry-Sethi 构造的其他特性在于：NFA 状态数与正则表达式中的符号数相同并具有一个开始状态。在此类构造中，将每个符号的出现都视为一个新的符号。这种构造将正则表达式 $(xy+y)^*yx$ 转换为标记表达式 $(x_1y_2+y_3)^*y_4x_5$ ，其中每个 x_1 导致同一状态 7607。图中未示出标记。一旦构造了 FSA 之后，就立即删除标记。图 76b 示出了删除标记之后的 NFA。可以从图中看出，在 Berry-Sethi 构造中，所有到达一个状态的入站转移都依赖于同一符号。同

样,如上所述的文献还研究并记载了 Berry-Sethi 构造的二元性,其中除了所有入站转移依赖于同一符号外,所有来自某一状态的出站转移也依赖于同一符号。Berry-Sethi 构造还称为左侧偏置构造,与其对偶的构造称为右侧偏置构造。

图 76c 示出了同一正则表达式 $(xy+y)^*yx$ 的 DFA。DFA 是确定的,因为在给定的时间仅激活了其状态之一,并且根据输入符号仅进行一次转移。而在 NFA 中,多个状态可以同时被激活,并且可以根据一个输入符号执行从一种状态到多种状态的转移。所述文献中存在许多诸如子集构造之类的公知算法以便将 RE 或 NFA 转换为 DFA。对于针对正则表达式示出的 DFA 要指出的一点是, DFA 中的状态数要少于 Thompson NFA 以及 Berry-SethiNFA 中的状态数。N 字符 DFA 的状态数上限为 2^N ,而在 Aho、Sethi 和 Ullman 所著的书中有关上述编辑器的章节 3.7 内提及的词法分析应用中,不常出现导致 DFA 状态数上限的表达式。

图 77a 示出了左侧偏置 NFA 及其状态转移表(现有技术)。图中为一般四状态的类似 Berry-Sethi 的 NFA,其中示出了通过转移所依赖的相应符号从一个节点到其他节点的所有转移。例如,状态 A (7701) 具有依赖于符号'a'的所有入站转移,如标记为 7702 和 7703 的转移实例所示。当 FSA 处于状态 A (7701) 时,输入符号'd'会将 FSA 转移到状态 D,即通过转移 7704 从状态 A 转移到状态 D。图中的表示出了使用状态转移表的同一个 FSA。列'PS' (7711) 是 FSA 的当前状态,而行'sym' (7712) 为状态转移所依赖的所有符号的列表。表 7713 示出了当接收到来自 sym 标题行的输入符号时,FSA 将从当前状态(PS)转移到的下一状态(NS)。在此 FSA 中,状态'A'为开始状态,状态 C 为接受状态。因此,如果 FSA 处于当前状态'A',并且收到输入符号'b',则 FSA 将转移到下一状态'B'。因此,当收到下一输入符号时,FSA 处于当前状态'B'中并使用与当前状态'B'对应的行进行计算以进行状态转移。

图 77b 示出了右侧偏置 NFA 及其状态转移表(现有技术)。图中为一般的四状态二元 Berry-Sethi NFA,其中示出了通过转移所依赖的相应符

号从一个节点到其他节点的所有转移。例如，状态 A (7705) 具有依赖于符号'a'的所有出站转移，如标记为 7708 和 7709 的转移实例所示，这与上述左侧偏置 NFA 不同，每个入站转移不在同一符号上，例如标记为 7706 和 7707 的转移分别依赖于符号'b'和符号'd'。此图中的状态转移表与左侧偏置图中的转移表类似，只是 FSA 根据同一输入符号转移到多种状态。例如，如果 FSA 处于当前状态'B'并且收到输入符号'b'，则 FSA 将转移到所有状态'A'、'B'、'C'和'D'。当收到将 FSA 指向空格（如 7716）的输入符号时，FSA 收到其无法识别的字符串。然后可以初始化 FSA 以便再次从开始状态开始计算下一字符串并可能指示字符串无法识别。

图 77a 和 77b 示出了一般的四状态 NFA，其中根据左侧偏置或右侧偏置构造特性示出了从每种状态到其他状态的所有转移。但是并非所有的四状态 NFA 都需要进行所有转移。因此，如果收到要求 FSA 从当前状态转移到下一状态的符号，并且不能在收到的输入符号上执行此类转移，则称 NFA 无法识别输入字符串。此时将以开始状态重新启动 NFA 来识别下一字符串。一般而言，如果具有根据 RE 的结果四状态 NFA 来启用或禁用给定转移的机制，用户便可使用这些实例四状态 NFA 以左侧偏置 (LB) 或右侧偏置 (RB) 的形式表示任何四状态 RE。

图 78a 示出了左侧偏置和右侧偏置 NFA 的状态转移控制。图中示出了具有状态'A' (7800) 的左侧偏置 NFA，所述状态'A'具有依赖于来自状态'B' (7801)、'C' (7802) 和'D' (7803) 的接收输入符号'S1'的入站转移。但是，除了接收输入符号'S1'以外，仅当设置了相应的状态相关控制时，才会从状态'B'、'C'和'D'中的每个状态转移到状态'A'。用于从状态'B'转移到状态'A'的状态相关控制为 V_2 ，而用于从状态'C'和'D'转移到状态'A'的状态相关控制分别为 V_3 和 V_4 。到下一状态'A'的转移依赖于当前状态'A'并通过状态相关控制 V_1 实现。因此，如果收到的输入符号为'S1'并且设置了用于相应转移的状态相关控制，才会执行到状态'A'的转移。因此，用户可以通过设置或清除用于特定转移的状态相关控制来表示任意的四状态 NFA。因此，如果四状态左侧偏置 NFA 包括从状态'B'和'C'到状态'A'的转移，而

不包括从状态'A'或'D'的转移,则可以将状态相关控制设置为 $V_1=0$ 、 $V_2=1$ 、 $V_3=1$ 和 $V_4=0$ 。因此,如果 NFA 处于状态'D'并接收符号'S1',则 NFA 不会转移到状态'A',但是如果 NFA 处于状态'B'并接收符号'S1',NFA 将转移到状态'A'。

同样地,图 78b 还示出了右侧偏置 NFA 的状态和转移。图中示出了具有状态'A' (7806) 的右侧偏置 NFA,所述状态'A'具有在分别收到输入符号'S2'、'S3'和'S4'时进行的来自状态'B' (7807)、'C' (7808) 和'D' (7809) 的入站转移。但是,除了接收相应的输入符号以外,仅当设置了相应的状态相关控制时,才会从状态'B'、'C'和'D'中的每个状态转移到状态'A'。用于从状态'B'转移到状态'A'的状态相关控制为 V_2 ,而用于从状态'C'和'D'转移到状态'A'的状态相关控制分别为 V_3 和 V_4 。到下一状态'A'的转移依赖于当前状态'A'并通过状态相关控制 V_1 实现。因此,如果收到了相应的输入符号并且设置了用于相应转移的状态相关控制,将执行到状态'A'的转移。因此,用户可以通过设置或清除用于特定转移的状态相关控制来表示任意的四状态右侧偏置 NFA。用于给定状态的所有状态转移控制将形成状态相关向量 (SDV),对于图 78a 中的左侧偏置和右侧偏置 NFA 实例,所述向量包括 V_1 、 V_2 、 V 和 V_4 。

图 78b 示出了按状态列出的可配置下一状态表。表 7811 示出了'NS=A'的左侧偏置状态表,而表 7812 示出了'NS=A'的右侧偏置状态表。左侧偏置和右侧偏置 NFA 状态的状态相关向量都相同,而对于左侧偏置和右侧偏置 NFA 状态而言,收到的驱动转移的输入符号是不同的。因此,状态可以使用诸如左侧偏置 (LB)、右侧偏置 (RB)、开始状态、接受状态、SDV 以及在计算包括该状态的 NFA 的输入字符串时已达到该状态的情况下采取的操作之类的特性来表示。

图 79a 示出了状态的状态转移逻辑 (STL)。STL 用于计算某状态的下一状态。使用状态的 STL 计算的下一状态依赖于 NFA 的当前状态、SDV 以及分别针对左侧偏置和右侧偏置 NFA 收到的一个或多个符号。将使用符号检测逻辑 (块 7900) 根据符号'S1'到'Sn'计算 InChar 输入,其中'n'为表

示 NFA 的 RE 中符号数的整数。'n'的选择取决于通常针对使用搜索处理器的应用的 NFA 而预期的状态数。因此，可以将'n'选择为 8、16、32 或其他任意整数。最简单的符号检测操作是将输入符号与'S1'到'Sn'进行比较。符号检测逻辑的输出称为所接收符号向量(RSV)，它由各个检测符号'RS1'到'RSn'组成。LB/RB#是指示是否定义了左侧偏置 NFA 或右侧偏置 NFA 的符号（即，左侧偏置或右侧偏置符号）。LB/RB#还在计算状态转移时用作输入。状态的 STL 支持创建左侧偏置和右侧偏置 NFA 构造。LB/RB#符号控制 STL 将实现左侧偏置构造还是右侧偏置构造。采用'V1'到'Vn'形式的状态相关向量还作为输入应用于 STL。

SDV 允许使用 STL 作为图 79b 中示出的状态逻辑块的基础来创建任意的'n'状态 NFA。当前状态被馈入 STL 作为包括'Q1'到'Qn'的当前状态向量(CSV)。STL 在下一输入时钟信号上生成在状态存储器(块 7902)中更新的符号'N1'。'N1'在逻辑上表示为 $N1 = ((V1 \text{ and } Q1 \text{ and } (LB/RB\# \text{ OR } RS1)) \text{ OR } (V2 \text{ and } Q2 \text{ and } (LB/RB\# \text{ OR } RS2)) \text{ OR } \dots (Vn \text{ and } Qn \text{ and } (LB/RB\# \text{ OR } RSn)) \text{ AND } ((\text{NOT } LB/RB\# \text{ OR } RS1))$ 。将使用类似的逻辑生成另一状态'n'的类似符号，只是馈入‘或’门 7915 的信号 7901 将为'RSn'，它是第'n'个符号检测逻辑的输出，将节点'N1'逻辑的最后一项从((NOT LB/RB# OR RS1)更改为((NOT LB/RB# OR RSn)。可以将状态存储器 7902 实现为将在下面讨论的状态逻辑块中的单位触发器。

图 79b 示出了状态逻辑块(SLB)。SLB 包括 STL 7906、初始化逻辑 7908、状态存储器 7910、接受状态检测逻辑 7911、该状态的 SDV 7907、开始标志 7909、接受标志 7912、与该状态关联的标记 7919 或与该状态关联的操作 7913 或上述内容的组合。SLB 为下列每项提供可编程存储:SDV、开始标志、接受标志、标记字段、操作字段、左侧偏置或右侧偏置信号以及状态存储器或上述内容的组合，可以使用与下面介绍的群集配置控制器 8607、全局配置控制器 8505 协作的应用状态控制器(块 8000)编程所述存储，以便在运行时配置 SLB 中已实现的 FSA 的状态。SLB 接收馈入 STL 的当前状态向量和所接收符号向量来确定下一状态。然后可以通过更新状

态的 SDV 并选择 NFA 检测和处理的符号来实现任意 NFA 的状态。进而，可以通过开始标志和接受标志将每个状态标记为开始状态或接受状态或它们的组合，或标记为既不是开始也不是接受状态。初始化逻辑块 7908 接收控制符号，所述控制符号指示是需要从开始状态初始化状态，从更新清除或禁用状态，使用另一状态值直接加载状态，还是检测下面介绍的计数器值并确定是否接受转移等。初始化块可用于忽略 STL 计算并设置状态存储器以激活或取消激活状态。STL 7906 提供如图 79a 中示出的功能，除了状态存储器包括在 SLB 中作为独立的功能块 7910。状态存储器 7910 可以实现为单位触发器。设置了状态存储器时，它将指示状态是激活的，否则状态未激活。接受检测逻辑 7911 检测是否已经激活了该状态，以及状态是否为已实现的 NFA 的接受状态。如果状态为接受状态，并且在 NFA 计算期间达到该状态，则提供关联的操作或标记以作为 A1 信号 7916 上的 SLB 的输出以及 M1 (7917) 上指示的接受状态激活。所述操作或标记可以被设置为在状态激活时以及当状态不是 NFA 实现所需的接受状态时作为 SLB 的输出。这使 SLB 能够用于标记的 NFA 实施方式，其中操作或标记可以与给定的状态转移关联。

如果在下文所介绍的搜索引擎中支持' n '个状态，则每个 SLB 需要' n '位 SDV，所述 SDV 可以作为包括触发器的寄存器来存储，其中 2 位分配给开始标志和接受标志，1 位用于 LB/RB#， m 位用于操作存储并且 t 位用于标记存储。因此，如果 $n=16$ 并且 $m=6$ 以及 $t=4$ ，则每个 SLB 使用的总存储将为 29 位寄存器，其中每个状态略少于 4 字节。

图 80 示出了基于 NFA 的搜索引擎。所述搜索引擎包括' n '个 SLB，即 SLB1 到 SLB n (块 8002)、应用状态控制 (块 8000)、应用状态存储器 (块 8001)、计数器 (块 8004)、匹配逻辑和接受状态检测 (块 8005)、FSA 扩展逻辑 (块 8006)、符号检测阵列 (块 8007) 或模式替换控制 (块 8008) 或上述内容的组合，以生成运行时自适应 NFA 搜索引擎。每个搜索引擎中包含的状态数可以是任意整数' n '。例如，' n '可以是 8、16 或 32 或其他任意整数。搜索引擎完全支持具有近' n '个符号的正则表达式。如果从

RE 得到的状态数多于搜索引擎所支持的数目, 则可以使用 FSA 扩展逻辑 (块 8006) 将两个或更多搜索引擎连接到一起以处理更大的 RE。如果 RE 的状态数小于 'n', 则搜索引擎允许每个搜索引擎具有多个 NFA。这可以有效利用搜索引擎资源, 这与不允许每个搜索引擎具有多个 RE 的解决方案不同。每个搜索引擎可以支持 'M' 个规则, 其中 M 可以是小于 'n' 的任意整数。例如, 如果 'M' 为 4, 而 'n' 为 16, 则可以在搜索引擎中配置四个 RE 规则以便同时进行计算。虽然 'M' 是可以同时在 SLB 中配置来实现计算的数字, 但是可以存在更多能存储在应用状态存储器中的规则。存储在应用状态存储器中的规则和应用上下文总数是按搜索引擎分配的存储器量的函数并且不限于 'n' 和 'M'。SLB 是在图 79b 中示出的状态逻辑块。搜索引擎中的 SLB 数与搜索引擎所支持的状态数相同。

搜索引擎分别接收 FSA 控制和数据总线 ('FSA 控制' 和 'FSA 数据')、总线 'InChar' 和替换控制上的输入符号以及 '子总线' 上的数据作为输入。为了简化视图, 其他必要的如电源、时钟、重置之类的标准输入未在此图或本披露中的其他图中示出。在基本硬件设计课本中大量使用了这些符号并且本领域的任何技术人员都将理解它们。尽管文中的描述参考了所示的输入, 但是可以在多路复用总线或其他并行总线上接收信息, 因此不能将此描述视为对本发明的限制。这些符号和总线的示出是为了便于介绍。搜索引擎还包括保存诸如 HTTP、SMTP、XML 之类的多种应用的应用专用 NFA 规则上下文的应用状态存储器。应用状态存储器保存诸如每个应用 NFA 的 SDV、初始状态向量 (ISV)、接受状态向量 (ASV)、LB/RB#、操作/标记、规则号之类的信息。应用状态控制 (块 8000) 由搜索处理器的全局配置控制器 (将在下文介绍) 使用, 以便为每个所支持的应用在应用状态存储器中配置相应的 NFA 参数。然后, 配置控制器可以指示应用状态控制使用特定应用 NFA 对 SLB 编程以便搜索该应用。如果 RE 的状态数加起来小于搜索引擎所支持的数目, 则每个应用可以在搜索引擎中具有多个 RE 或规则。应用状态控制可以从应用状态存储器中检索相应的应用上下文, 并使用相应的信息对每个 SLB 编程以实现针对应用而编程的

NFA。在每个 SLB 中配置的参数包括 SDV、开始状态、接受状态、LB/RB#、操作或标记。应用状态控制器还使用从应用状态存储器中检索的 NFA 的符号来配置符号检测阵列（块 8007）。使搜索引擎开始处理在 'InChar' 上收到的输入符号之前，还可以由应用状态控制针对 NFA 完成相应的计数器初始化。将 SLB 的状态输出连接在一起以形成 CSV。RSV 是符号检测阵列的输出。将 RSV 和 CSV 馈入每个 SLB。然后，SLB 根据其配置执行状态转移计算，直到达到接受状态，或者 NFA 达到 Null 状态，所述 Null 状态指示输入字符串被 NFA 识别或者被 NFA 拒绝。符号检测阵列 8007 包括计算引擎阵列（最简单的形式可以是比较器）以及用于所搜索符号的存储器。将在 'InChar' 上收到的输入符号同时与所有的符号相比较，如果所搜索的符号与所收到的符号匹配，则设置 RSV 相应信号。符号检测阵列还与计数器（块 8004）交互，以启用在执行特定转移之前检测符号出现的规则。匹配和接受状态检测逻辑（块 8005）使用每个 SLB 中的接受状态和操作字段来检测 NFA 是否达到接受状态，然后注册 NFA 操作以将其提供给下文介绍的全局配置控制器。操作或标记可以提供已匹配 RE 规则的指示，但是还可以提供可以在输入流上执行的操作。例如，所述操作可以指示输入流中包含被搜索的计算机病毒，因此需要隔离输入流，或者所述操作检测到被搜索的保密信息是流的一部分，因此，应停止该流的传输或执行类似的操作。操作或标记还可以包括指向全局控制器或控制处理器（将在下文介绍）执行的代码例程的指针。所述代码例程可以使搜索引擎的配置更改为搜索其他规则集。例如，如果 NFA 是上文中的规则集<INITIAL>的一部分并且被触发，则要执行的操作可以例如是激活<S_TAG>规则集，并将<INITIAL>集留待处理。作为匹配 RE 规则的结果，可以使用调用的代码例程来完成无数种任务，这对本领域的技术人员是显而易见的。一旦 NFA 达到接受状态，可以在输入流上执行复杂操作集，可以激活其他控制功能，或者可以激活管理代码等，如果编程了关联的操作或标记以使此类操作发生的话。匹配逻辑 8005 可以在 M 总线 8009 上输出匹配指示以及匹配的规则 ID，以便群集控制器（将在下文介绍）或全局配置控制器采取

相应的操作。操作/标记 8010 将关联的操作或标记传输给可能执行上述操作的群集控制器或全局配置控制器。匹配逻辑被设置为支持由搜索引擎执行的'M'个并行规则计算。

存在一些需要在状态转移发生之前对特定事件数进行计数的正则表达式实例。例如可以创建[3,5]abc之类的 RE 规则，其指示 RE 识别具有 3 到 5 个前导'a'，后跟'b'和'c'的字符串。如果输入字符串为'aabc'或'abc'或'aaaaaabc'，则该 RE 将无法识别这些字符串，而该 RE 将识别输入字符串'aaaabc'或'aaabc'或'aaaaabc'。为了启用此类 RE，在搜索引擎中提供了若干计数器作为计数器（块 8004）的一部分。可能存在'L'组计数器来支持同时执行多个规则。如果某些规则不需要使用计数器，则可以将这些计数器分配给其他规则使用。'L'的选择可以是搜索引擎资源可用性以及使用搜索引擎支持的应用对计数器的需求的函数。如果搜索引擎中配置的 NFA 需要计数器，则应用状态控制器可以在配置应用状态时将计数器值读取或写入到 SLB。将计数器块的输出提供给 SLB 以使其决定所需的计数器值。所述计数器也可以由符号检测阵列来读取和写入，以便该阵列支持依赖于出现了特定次数的计数符号的 RE 规则，如上所述的情况那样。计数器块还保持计数器以跟踪'M'个规则中每个规则的字符串匹配的开始和结尾。开始计数和结束计数分别向群集控制器或控制处理器指示了输入流中字符串匹配的开始标记和输入流中的结束标记。

FSA 扩展逻辑（块 8006）用于扩展可以为 RE 支持的 NFA 状态数。因此，如果 RE 所需的状态数多于单个搜索引擎所支持的'n'，则可以将多个搜索引擎一起连接成组来支持所述 RE。FSA 扩展逻辑块提供使搜索引擎能够知道组中其他搜索引擎的状态的输入和输出信号。尽管未在图 79a 或 79b 或 80 中显式示出，但是所述信号可以被馈入 SLB 来作为控制符号。

搜索引擎还包括模式替换控制 8008，它保存要在搜索引擎经过 NFA 时用于替换输入字符串的模式（多个）。这可以支持在文献中记载且公知的有限状态变换器功能。该块通过跟踪两个连续的 CSV 并确定启用的转移以及要为替换收到的输入字符串而生成的关联变换器符号来计算通过

NFA 进行的转移。

具有' $n=16$ '个状态的搜索引擎将需要用于 SLB 的 $16 \times 4=64$ 个字节的存储, 加上用于 16 个符号的 16 字节的存储以及 M 个计数器, 其中如果 $M=4$, 每个计数器为 16 位, 则将 8 个字节用于计数器。因此, 对于 $n=16$, 应用状态上下文 (将在下文中介绍) 所需的总存储大约为 90 字节。

图 81 示出了应用状态存储器实例配置。应用状态存储器 8100 与图 80 中的块 8001 相同。图 81 示出了如何将该存储器组织为存储各种应用的应用状态上下文。应用状态上下文包括诸如 SDV、开始状态、接受状态、NFA 计算符号、操作及状态标记之类的信息。这是应用状态控制可以从应用状态存储器检索并配置搜索引擎的 SLB、计数器、符号检测阵列、匹配检测和接受检测逻辑块的信息。它还可以包括指示该特定 RE 是否需要与其他搜索引擎块合成一组, 以支持比搜索处理器支持的状态数所支持的 RE 更大的 RE。因此, 上下文中承载了计算 NFA 所必须的全部状态信息。可存储在应用状态存储器中的应用状态上下文数 ' N ' 完全是为每个搜索引擎分配的应用状态存储器量以及每个上下文所需的存储量的函数。当 ' $n=16$ ' 时, 如果为每个搜索引擎分配的存储器量为 512 字节, 则可以将 5 个应用上下文存储在应用状态存储器中。1KB 的存储器允许存储 10 个应用上下文。但是, 如果每个搜索引擎的状态数为 8, 则上下文数最多可以是状态数为 16 的情况下的上下文数的两倍。

有 ' N ' 个应用状态上下文 (ASC) 8101、8102、8103 存储在应用状态存储器 8100 中。ASC1 8101 示为用于反垃圾邮件应用的 RE 规则之一, 而 ASCn 8103 示为用于 XML 应用的 RE 规则之一。每个 ASC 都可以被进一步组织为多个 NFA 参数 8104 和 SLB 参数 8105。SLB 参数是状态专用参数, 例如开始标志、接受标志、SDV、操作以及应用于特定 SLB 的参数。这些可能只是一个 4 字节存储单元。同样地, NFA 参数是可应用于整体 NFA 的参数, 如符号、每个 NFA 的规则数、替换表等。右侧偏置/左侧偏置信息可以通用于整个 NFA 或可以按状态进行存储。

ASC 的优点在于当控制处理器或全局配置处理器或群集处理器 (将在

下面介绍)需要针对与当前在搜索引擎中配置的应用不同的应用来应用规则时,它可以通过通知每个搜索引擎的应用状态控制来迅速转移到新的应用的规则。因此,在可用于并行执行大量 RE 计算的搜索引擎阵列中,可以很快地切换到某个应用上下文,与必须从全局存储器或外部系统存储器检索配置信息并将此信息提供给每个搜索引擎的情况不同。在 $n = 16$ 个状态的实例中,可以在 20 到 25 个时钟内为每个搜索引擎切换应用上下文。如果并行使用 1000 个搜索引擎执行 1000 个 RE 规则搜索,并且如果基于应用的上下文必须从全局存储器或外部存储器加载到每个搜索引擎,则可能潜在需要 20,000 到 25,000 个时钟周期,与本发明中的按搜索引擎使用 ASC 相比,这会显著降低搜索处理器的性能。

尽管上述内容针对不同的应用,但是可以为不同 ASC 中的同一应用使用不同的 RE。对于某一应用,这可允许搜索处理器支持更多 RE,尽管在所有 RE 必须应用于输入流的情况下,这会对性能造成影响。在这种情况下,输入流会被多次提供给搜索引擎,每次都切换 ASC,实际上将用于该应用的新 RE 应用到所述流。另外,ASC 的使用模型可以用于同一应用的嵌套 RE 规则。例如,如图 75 所示,RE 规则是嵌套规则,其中仅有有限集合被并行应用到输入流。在此情况下,可以在各个 ASC 中设置来自每个嵌套级的一个规则,并且编程的操作将指示在当前规则达到接受状态时,需要将上下文切换到另一 ASC。例如,R9、R7 和 R4 可以被分别存储在三个独立的 ASC 中,其中规则 R9 被用作初始上下文。现在,如果输入字符串与 R9 匹配,则可以切换上下文以引入随后查找'FONT'字符串的规则 R7。如果 R7 达到接受状态,则可以将上下文切换到 R4,依此类推。因此,本发明中的搜索引擎可以轻松支持用于任何给定应用的嵌套 RE 规则集。从而,用户可以使用上述基于 NFA 的搜索引擎的各种运行时可配置和可编程功能来创建运行时自适应有限状态自动机体系结构。

如上所述,可以使用公知的方法将正则表达式转换为 NFA 和 DFA。进而,还公知的是同一 RE 的 NFA 和 DFA 是等价的。但是如上所述,NFA 和 DFA 的性能度量方面存在差异。与 NFA 相比,DFA 在最坏的情况下可

能导致用于状态的大量存储要求。但实际上不存在这种情况，并且在很多情况下 DFA 所具有的状态数要少于 NFA。因此，仅使用基于 NFA 的搜索引擎的内容搜索处理器可能无法实现最高效的实施方式。用户可以使用 NFA 和 DFA 的组合计算一组正则表达式。进而，如果 DFA 实施方式可以比 NFA 更高效，则与单纯使用 NFA 的搜索处理器相比，单个集成搜索处理器可支持更多的规则。DFA 中的一种状态可以转移到多种状态，每种转移都在不同的输入符号上进行。因此为了处理状态，用户需要检测多个符号并根据输入符号选择适当的转移。在 DFA 中，由于 DFA 的确定性特性，一次只有一个状态是活动的。因此，可以选择与图 76c 中所示的 DFA 类似的 DFA，其中 DFA 中的状态数小于 NFA 中的状态数或可以与之相比并且从一种状态进行的转移数也是有限的（在这种情况下为二者中的上限），并对它们编程以便使用如图 82 中示出的基于 DFA 的搜索引擎进行计算。只要 DFA 的状态数不超过下文中描述的 DFA 指令和状态流存储器中可保存的状态数，便可以选择 DFA 而非 NFA 进行计算。

图 82 示出了基于 DFA 的搜索引擎。DFA 搜索引擎包括 DFA 操作引擎 8204、DFA 状态控制器 8203、DFA 上下文控制 8200、DFA 指令和状态流存储器 8201、计数器 8202、DFA 扩展逻辑 8208、发生计数器 8206，最小和最大计数器 8207 或模式替换控制 8205 或上述内容的组合，以构成运行时自适应 DFA 搜索引擎。DFA 搜索引擎还具有使用 DFA 扩展逻辑 8208（其提供与图 80 中的 FSA 扩展逻辑 8006 类似的功能）创建 FSA 扩展的工具。因此，可以将多个 DFA 块分组到一起以支持产生比一个 DFA 搜索引擎支持更大 DFA 的 RE。DFA 操作引擎是支持执行 DFA 特定的目标操作的小型处理器。图 83 中示出了所述类型的 DFA 操作一个实例。DFA 状态控制器 8203 接收来自群集控制器或全局配置控制器的命令，以利用形成 DFA 流的 DFA 操作，使用指令序列对指令存储器和状态流存储器 8201 编程。DFA 上下文控制器 8200 控制当前执行的应用和 DFA 上下文。DFA 搜索引擎还支持与图 80 的 NFA 搜索引擎类似的应用状态上下文并且所述上下文存储在指令和状态流存储器 8201 中。DFA 的应用上下文主要包括

用于每个应用的 RE 规则的指令序列。与上述 NFA 搜索引擎类似，DFA 搜索引擎也可以保存用于同一应用的多个 RE，从而支持上文针对 NFA 所述的 RE 嵌套。DFA 操作引擎 8204 本质上是支持与下文中描述的实例类似的所有 DFA 操作的标量处理引擎。DFA 操作引擎可以是多路超标量体系结构，其中可以同时处理多个指令。多路指令本质上是状态转移计算指令，该转移计算指令将根据收到的输入符号使 DFA 达到多个下一状态中的一个状态。如果 DFA 操作引擎支持两个并行计算来支持每个状态最多两次转移的 DFA，则可以将每个 RE 转换为 DFA 以满足此类需求，或者用户可以比较将 RE 实现为 NFA 与实现为 DFA 的成本，并通过将该 RE 映射到内容搜索处理器上的适当搜索引擎来选择成本最低的实现方式。通常，DFA 操作引擎不保留状态信息，因为 DFA 指令流保存状态信息。进而，DFA 操作引擎通过同一引擎计算每个入站符号，因此与 NFA 搜索引擎不同，它没有多符号检测阵列的硬件开销。这使得 DFA 搜索引擎与 NFA 搜索引擎相比，实现成本非常低，其中设备利用差别数可以达到 5 倍。DFA 上下文控制器 8200 将上下文和上下文内指令的指针提供给指令存储器，所述指令存储器检索要由 DFA 操作引擎 8204 计算的指令向量（多个）。指令向量包括指令或操作、预期符号（多个）、下一状态指针（多个）、默认状态或控制信息或上述内容的组合。如果为 DFA 搜索引擎选择了多路体系结构，则指令向量可以包含多个转移的信息。DFA 操作引擎根据收到的/输入符号执行或计算指令并决定接下来转移到那个状态并开始计算。如果不存在与收到的符号和预期 DFA 操作的匹配，流将转移到可能是开始状态或错误状态的默认状态。如果 DFA 操作引擎达到接受状态，DFA 操作引擎在 M 总线上生成相应的匹配符号连同与该接受状态关联的标记或操作。此信息由群集控制器或全局配置控制器处理，它们决定针对输入执行的下一操作过程。DFA 操作引擎可以访问用于支持各种 DFA 操作的发生计数器 8206、最小和最大计数器 8207 以及字符计数器 8202。字符计数器用于标记与 DFA 规则匹配的字符串的开始和结尾。模式替换控制（块 8205）和 DFA 扩展逻辑（块 8208）提供与分别针对图 80 中示出的块 8008 和 8006

描述的功能类似的功能。

图 83 示出了实例 DFA 操作。所述操作已经被定义为模拟通用正则表达式运算符。在不偏离本申请的教导的情况下，DFA 操作的数量和类型可以与所示的不同。例如，所示的运算“EQ X”可用于计算收到的符号是否等于'X'。如果此计算的结果是肯定的，则可以执行需要此计算的转移。所示的 DFA 操作形成由 DFA 操作引擎 8204 处理或计算的指令。DFA 指令包括的字段多于图 83 中示出的字段。DFA 指令可以包括图 83 中示出的 DFA 操作、预期符号（多个）、指令计算成功时所分支到的目标指令地址（多个）、计算不成功时的备选目标指令地址或诸如计数器设置、计数器检索、输出接受状态标志、操作或标记之类的其他控制字段或上述内容的组合。所计算的符号数以及目标指令地址数取决于支持的并行转移数。当 DFA 操作引擎为双路超标量时，将有两个符号并且除了默认地址外，还有两个与它们关联的目标地址。DFA 操作的每个指令都表示从一种 DFA 状态进行转移的计算。因此，当分支到目标指令地址时，目标处的指令通常为用于 DFA 所转移到的状态的指令，除非该状态是错误状态，在此情况下，可以在初始或开始状态时重新启动 DFA。例如，用户可以使用图 83 中示出的某些 DFA 操作按照下面的方式编写图 76c 中示出的 DFA 的代码。以下实例并非限制 DFA 操作的格式，仅作为示例来更好地解释 DFA 搜索引擎操作。

状态/IP	op1	S1	T1	op2	S2	T2	其他
A:0	EQ	'x'	1	EQ	'y'	2	0, 开始计数
B:1	EQ	'y'	0	EQ	-	-	0
C:2	EQ	'y'	2	EQ	'x'	3	0
D:3	GEN_FLAG	-	-	EQ	'y'	0	0、操作、标记、标志、 结束计数

上述指令的字段为 op1（第一操作）、S1（符号 1）、T1（IP 目标 1）、op2（第二操作）、S2（符号 2）、T2（IP 目标 2）以及其他字段（例如默

认目标、初始开始计数、初始结束计数、操作、标记等)。对于图 76c 中示出的 DFA 计算,此类代码可以存储在可从中检索和处理每个指令的 DFA 指令和状态流存储器中。标题为“状态/IP”的列表示 DFA 的相应状态以及相应指令的指令指针(IP)值。所示内容之前可具有需要在 DFA 代码之前执行的前导代码,以便设置相应的计数器、错误状态处理等。但是,在上面示出的实例中,如果输入符号不是 DFA 所预期的符号,则将开始状态'A'及其 IP 用作默认目标。DFA 计算从开始状态且 IP=0 时开始,首先根据输入符号取回和计算 IP=0 处的指令。如果输入符号为'y',则预期 DFA 从状态'A' 7612 转移到状态'C' 7614,但是,如果收到的符号为'x',则会转移到状态'B' 7613。假设输入符号为'y',则第一指令的指令计算将导致符号 2 (S2)与'y'的肯定比较,因此将关联的目标 IP 2 选为要根据下一符号取回和计算的下一指令。IP=2 对应于状态'C',所述状态被预期进行图 76c 中的 DFA 所定义的转移。DFA 搜索引擎现在处理 IP=2 处的指令,然后执行指令所定义的相应的流。当执行到达与接受状态'D' 7615 对应的 IP=3 时,DFA 搜索引擎将输出匹配标志和关联的标记或操作,以指示找到了与当前 DFA 匹配的字符串。DFA 引擎还记录结束计数来指示字符串匹配在输入流中结束的位置。具有上面示出的多路执行的 DFA 引擎还为同时执行的多个操作提供优先级,其中如果第一操作的结果是肯定的,则会获得比其他操作更高的优先级。通常,由于 DFA 是确定性的,因此仅有一个列出的指令操作产生肯定的结果,尽管可以创建此类机制来避免执行不定性。

尽管上述实例仅使用了两类操作,但是 DFA 操作引擎可以执行如图 83 中示出的许多其他操作。可以定义 RANGE 操作,此操作允许用户搜索落入两个符号的 ASCII 代码范围内的符号'S',其中'X'表示范围的下限,'Y'表示范围的上限。在这种情况下,'X'和'Y'可以是任何 8 位或 16 位扩展 ASCII 代码。同样,定义了一组操作来允许 DFA 代码在给定的状态下循环,直到满足特定的条件或者出现特定次数的特定符号。还存在其他方法来实现这些操作,它们在本发明的范围之内并且对本领域的技术人员是显而易见的。

图 82 的基于 DFA 的搜索引擎本质上是一个处理如图 83 中所示的目标指令的应用专用处理器。DFA 搜索引擎通过使 DFA 上下文控制 8200 从 DFA 指令和状态流存储器 8201 取回指令来进行操作,然后 DFA 操作引擎 8204 通过执行指令所指示的操作来计算指令,所述计算通常包括使用指令所预期的操作计算在'inChar'上收到的输入符号,然后根据计算结果确定下一要取回的指令。DFA 操作引擎 8204 将有关下一指令的信息提供给 DFA 状态和上下文控制(块 8203 和 8200),所述控制又将下一指令地址提供给 DFA 指令和状态流控制器 8201。该处理器的性能可以达到或高于使用高级工艺技术的最新微处理器的性能,从而达到 1Gbps 到 10Gbps 甚至更高的线路速度性能。因此,可以使用上述基于 DFA 的搜索引擎的各种运行时可配置和可编程功能来创建运行时自适应有限状态自动机体系结构。

图 82 中示出的 DFA 搜索引擎可以在当今 FPGA 的 50 到 200 个逻辑元件中实现,具体取决于支持的 DFA 操作和多路执行支持。指令和流状态存储器可以是每个搜索引擎 512 字节或更少,具体取决于支持的应用上下文数和需要支持的状态数。这可以允许单个 FPGA 中包含上千个 DFA,所述单个 FPGA 可能具有近两万个逻辑元件。因此,基于 DFA 和 NFA 的内容搜索处理器可以在当今最高级的 FPGA 中实现,这种 FPGA 支持在以 FPGA 中可获得的时钟速率运行的一个 FPGA 中具有一千到两千个 RE。如果考虑了多个应用规则,则 FPGA 格式中支持更多数量的 RE,这将使数量达到数千个规则。因此,本发明支持除了 ASIC、ASSP 以及其他形式的集成电路之外,使用 FPGA 创建内容搜索解决方案,当上市时间很重要并且容量不足以抵消创建 ASIC、ASSP 或其他 IC 的成本时,这种方法很有用。使用在当今的最新微处理器中应用的最高级工艺技术(如 90nm 或 65nm)实现本发明的 NFA 和 DFA 搜索引擎时,这些引擎的性能可以达到相同或更高的频率,因此可以使用非常高的线路速度处理正则表达式,这一点与当今的微处理器是不同的。与 4GHz 处理器用于计算先前所述的几百个 NFA 和 DFA 时低于 100Mbps 的性能不同,本发明的具有 NFA 和 DFA

搜索引擎的一千 RE 内容搜索处理器可以达到与最新微处理器相同的时钟速率,并且在每个时钟内计算大约一个输入符号,所述输入可以是单字符或多字符符号,从而达到远在 10Gbps 之上的性能。进而,随着诸如 90nm、65nm 和更低的小尺寸工艺技术的发展,器件密度在不断增大,可以结合到一个 IC 中的 RE 数可能达到数千个,具体取决于所选的 NFA 和 DFA 的混合。通过在内容搜索处理器上混合使用 NFA 和 DFA,可以创建一系列在芯片上提供不同数量的 NFA 和 DFA 的产品,例如,一种产品可以提供 25% 的 DFA 和 75% 的 NFA,而另一种产品可以提供 75% 的 DFA 和 25% 的 NFA。因此可以满足不断变化的市场需求。如果应用所需的 RE 数多于在单个内容搜索处理器上提供的 RE 数,则可以并行使用多个处理器来增加 RE 数。因此,可使用本发明支持需要数千个 RE 的搜索应用,而无需将存储器使用量增加到复合 DFA 中所需的量(其对于一般用途而言是不经济的)。

图 84 示出了运行时自适应搜索处理器。所述处理器包括控制处理器和调度器 8402、协处理器/主机接口控制器 8401、数据/分组缓冲区 8403、自适应控制器 8404、运行时自适应搜索处理器阵列 8405、配置和全局存储器 8406 或存储器控制器 8407 或它们的组合。在本发明中,运行时自适应搜索处理器也称为搜索处理器或自适应搜索处理器或内容搜索处理器或运行时自适应内容搜索处理器。同样,运行时自适应搜索处理器阵列在本发明中也称为搜索处理器阵列。搜索处理器可以嵌入多类具有各种类似于图 68、69 和 70 中的配置的系统,具体是将这些图中的安全性处理器替换为搜索处理器。从本发明的处理器得到的搜索解决方案包括可以结合搜索处理器、安全性处理器、SAN 协议处理器、TCP/IP 处理器或运行时自适应协议处理器或本发明披露的其他各种处理器或上述内容组合的线卡。线卡配置和体系结构可以随特定的系统和应用而不同。本申请示出了三类线卡体系结构: a)直通线卡、b)后援线卡以及 c)加速线卡,以说明本申请的处理器器的使用模型。图 68、69 和 70 示出了使用基于安全性处理器的系统的这些配置,尽管还可以基于本申请的上述搜索处理器或其他处理器。块

6612 和 6613 示出了其中两类线卡配置。这些卡中示出的安全性处理器可以由本申请披露的搜索处理器替代。根据处理器中结合的功能，可以创建搜索处理器的各种不同变型。块 6614 和 6615 示出了两种版本的安全性处理器，所述块还可用于通过将安全性处理器替换为搜索处理器来示出搜索处理器。块 6614 示出了包括至少一个与运行时自适应处理器相连的内容搜索和规则处理引擎的安全性处理器。可以通过将运行时自适应处理器替换为本应用中所披露的运行时自适应搜索处理器阵列 8405 来创建类似的搜索处理器。该处理器类似于图 71 中所示的具有相同替换并在上面参考安全性处理器上下文进行了详细描述的处理器的处理器。块 6615 示出了块 6614 的安全性处理器，所述安全性处理器与 TCP/IP 处理器或协议处理器相连以提供更多可以在作为安全性处理器的安全节点中使用的功能。同样，搜索处理器也可以与 TCP/IP 处理器或协议处理器相连以提供更多可以在内容搜索处理器节点中可用的功能，所述内容搜索处理器节点可以在网络中用作网络元素或客户机节点或服务器节点。此类处理器可用于启用应用感知联网能力来实现智能联网。搜索处理器的选择依赖于部署该处理器的系统、该系统所支持的功能、解决方案成本、性能要求或其他因素，或者上述内容的组合。搜索处理器可以使用一个或多个端口来连接到外部存储器（块 6616），所述存储器可用于存储搜索处理器中存储的规则以外的其他规则，或其他中间数据或分组或执行搜索处理所需的各种功能所需的其他信息。所述存储器可以具有各种类型，如 DRAM、SDRAM、DDR DRAM、DDR II DRAM、RLDRAM、SRAM、RDRAM、FCRAM、QDR SRAM、DDR SRAM、磁存储器、闪存或上述内容的组合或此类存储技术的未来派生物。本申请所披露的发明允许所示体系结构的多种变型，本领域的技术人员将理解，可以在不偏离本发明的原理和精神的情况下在各实施例中做出更改。

图 84 的搜索处理器可以在需要对本本地内容、远程内容或网络分组中收到的内容等的内容搜索进行加速的系统中用作协处理器。可使用该处理器的配置与图 69 和 70 中所示的配置类似，其中块 6906 和 7005 的安全性处理器可以由图 84 的搜索处理器或上述搜索处理器的其他变型进行替换。搜

搜索处理器可以结合在用作图 89 中示出的系统内的加速卡的附加卡中,其中搜索处理器 8904 可以是图 84 的搜索处理器或是上述搜索处理器的其他变型。搜索处理器还可以按照图 71 中所示的形式来配置,在图 71 中,运行时自适应处理器(块 7102)可以替换为运行时自适应搜索处理器阵列(块 8405)。此类搜索处理器的优点是允许处理器所支持的规则数超过运行时自适应搜索处理器阵列所支持的数目。因此,如果该搜索处理器阵列能够支持两千个 RE,则当用户需要将 RE 数目增加到超过两千时,可以将它们作为在内容搜索/规则处理引擎 7106 上实现的复合 DFA。因此,用户可以将 RE 的数目增加到外部存储器能够支持的数目,而无需添加其他搜索处理器。本申请中的内容搜索处理器还可以用作应用中的安全性处理器,所述应用需要深度分组检查或需要扫描诸如反垃圾邮件、反病毒、XML 安全性、Web 过滤防火墙、入侵检测/防护之类的应用内容。因此,本申请中所有涉及安全性处理器的图形还可以结合搜索处理器。

搜索处理器从如图 69、70、71、73 或 89 中的协处理器配置内的主处理器接收需要搜索的内容,或者当它与诸如图 68 或 72 中的网络接口之类的网络接口连接时,还可以从网络接收所述内容。控制处理器和调度器 8402 将要搜索数据或信息内容存储在数据/分组缓冲区 8403 中,以便在处理器阵列准备接收它们时将其调度到搜索处理器阵列中。通常,搜索处理器可以满足从 100Mbps 以下、1Gbps 到 10Gbps 及更高的线路速度,因此分组不需要进行缓冲。但是,如果分组或内容进入搜索处理器的速度高于该处理器可以处理的速度,则需要对分组进行缓冲。另外,可能需要调度属于搜索处理器按照正确的顺序处理的同一个流(例如位于同一传输层连接中)的分组。此类流中的分组也将存储在数据/分组缓冲区中,直到在流中位于它们前面的分组被处理完毕。搜索处理器还需要提供跨源和目的地之间的连接中的多个分组的检查能力。控制处理器和调度器(块 8402)也提供此类功能。控制处理器可以将运行时自适应搜索处理器阵列的内部处理状态存储在可以在配置和全局存储器 8406 或片外存储器中维护的连接数据库中。控制处理器和调度器将在收到主处理器所提供的与给定连接对

应的分组时查找该给定连接的执行或分析状态，同时还会查找直通配置中的入站业务。连接 ID 可以由主处理器创建并提供给带有要检查的分组的搜索处理器，或者搜索处理器可以根据分组的标头得出连接关联。在 IP 协议情况下，可以使用从源地址、目的地地址、源端口、目的地端口以及协议类型导出的 5 元组散列来创建连接 ID。一旦创建了连接 ID 并在散列冲突时由控制处理器和调度器解析了该 ID 之后，它便检索与该连接关联的状态并将此状态提供给搜索处理器阵列（块 8405）以便从连接状态开始搜索。此机制用于创建每个连接上的多分组搜索并检测所有跨分组边界的搜索字符串或安全违规或威胁。例如，如果定义了一个搜索“Million US Dollars”的规则并且该字符串出现在连接数据传输中的两个独立分组内，其中“Million U”出现在一个分组中，而“S Dollars”出现在另一分组中，则当没有本申请中的基于连接的多分组搜索机制时，可能无法检测到安全违规，因为单独每个分组并不与所述规则匹配。但是，当执行多分组搜索时，无论这两个分组到达搜索处理器的时间间隔多长，都会从连接的一个分组到另一分组保持搜索状态，并检测到两个分组的所述字符串且将它们标记为一个连续字符串“Million US Dollars”。

控制处理器和调度器按照适当的顺序将分组或数据调度到运行时自适应搜索处理器阵列。运行时自适应搜索处理器阵列（块 8405）、自适应控制器（块 8404）以及配置和全局存储器（8406）与图 62、63、64 和 65 中示出的相应元件类似。运行时自适应搜索处理器阵列及其关联的块提供类似的功能，其中进行了适当的逻辑增强以连接到搜索处理器的控制处理器和调度器。下面将介绍运行时自适应搜索处理器阵列及其组件。将为搜索处理器阵列提供入站分组或数据的每个字符，所述阵列随后检查这些字符与其中编程的 RE 规则的字符串匹配。当字符串匹配时，搜索处理器阵列将该指示连同诸如 RE 规则标识、要采取的操作、与此规则关联的标记、匹配的开始计数、匹配的结束计数之类的其他信息提供给控制处理器和调度器，后者然后根据所匹配的一个或多个规则采取相应的操作。搜索处理器阵列内的控制器还将采取下文描述的适当的必要操作。

图 97 示出了用于完全和增量规则分发的搜索编辑器流。图 97 的搜索编辑器允许 IT 管理人员创建上文图 67 中所述的不同类型的搜索和安全规则，并且还允许他们创建分层的和/或普适的安全模型。将向编辑器流提供特定节点的特征，例如安全能力提供、规则传送方法、支持的规则库大小、节点的度量性能、诸如 LAN 或 SAN 或 WAN 或其他之类的部署位置等。然后，编辑器流使用这些信息来根据 IT 管理人员所创建的规则集（多个）编辑节点专用规则。编辑器包括用于分析要提供给 FSA 编辑器流（块 9706）的规则的分析器（块 9704），所述 FSA 编辑器流分析规则并创建用于分析内容的规则数据库。规则分析器可以从规则文件读取规则，也可以直接从命令行或读取规则，或是它们的组合，具体取决于规则引擎的输出。将分析特定节点的规则以识别用于描述规则的语言特定的标记或正则表达式标记以及输出正则表达式规则 9705。然后，分析器将 RE 提供给将在下文参考图 96 详细描述 FSA 编辑器流。FSA 编辑器流处理入站 RE 并生成用于 RE 的 NFA 和 DFA。然后根据使用 DFA 和 NFA 成本以及节点的搜索处理器能力来确定是将 RE 作为 DFA 还是 NFA 进行处理。接着使用可以加载到搜索处理器中的格式生成 NFA 或 DFA 并将其存储在已编辑规则数据库存储装置 9708 中。

规则分发引擎（块 9709）遵循图 57 和 58 中示出的中央管理器流和规则分发流，只是进行了适当更改以达成正确的搜索处理器配置。可以根据节点能力将搜索规则相应地分发到图 58 中示出的主机处理器或控制平面处理器或控制处理器和调度器（块 8402）或它们的组合。可以按照特定节点的能力在网络上相应地使用专用协议或标准协议通过安全链路或不安全链路来分发规则。

控制处理器和调度器（块 8402）与规则分发引擎（块 9709）通信以便在启动内容检查之前接收相应的已编辑规则表。它将所接收的规则编程到上述与自适应控制器 8404 协作的相应 NFA 或 DFA 搜索引擎中。根据搜索处理器支持的应用上下文数，每个搜索引擎可以存储多个规则。当规则分发引擎将已编辑规则提供给控制处理器和调度器并且它们在各自的引擎

中设置完成后，搜索处理器便准备好开始处理数据流以执行内容检查。通过协处理器/主机接口控制器或介质接口控制器（未示出）来接收搜索处理器状态配置信息。本申请的搜索处理器可以部署在各种配置中，如图 69 中示出的后援配置或图 68 中示出的直通配置或图 70 中示出的加速器适配器配置，以及本领域的技术人员可以理解的其他未示出的配置。在后援配置或加速器适配器配置中，本申请的搜索处理器由主处理器控制，所述主处理器可以是网络处理器或交换处理器或 TCP/IP 处理器或分类处理器或转发处理器或主机处理器等，具体取决于此类适配卡所驻留的系统。控制处理器和调度器在此类主处理器的控制下接收配置信息，所述主处理器与规则引擎通信以接收包含配置信息的分组并将其继续传输到搜索处理器。一旦配置完成，主处理器便将分组或数据文件或内容提供给需要针对其使用协处理器或主机接口执行内容检查的搜索处理器。协处理器或主机接口可以是诸如 PCI、PCI-X、PCI express、RapidIO、HyperTransport 或 LA-1 或 SRAM 存储器接口之类的标准总线，也可以是专用总线。总线上的带宽应足以使内容搜索引擎以其峰值线路速度运行。搜索处理器可以是位于主处理器空间中的存储器映射或 IO 映射设备，以使其在后援配置或加速器配置中接收内容和其他配置信息。搜索处理器可以由主处理器进行轮询或者为主处理器提供门铃机制或中断机制，以指示它何时完成处理给定分组或内容或何时发现与编程规则的匹配。控制处理器和调度器（块 8402）及接口控制器（块 8401）与主处理器协作提供上述功能。

控制处理器和调度器将进站分组存储在分组缓冲区（块 8403）中，并调度分组以便在搜索处理器阵列（块 6405）可用于分析内容时由该阵列进行处理。调度器维护由特定引擎处理的分组的记录并在处理完分组之后立即通知主处理器。搜索处理器阵列在发现规则匹配以及与此规则关联的操作时通知控制处理器和调度器。控制处理器又将此信息发送给主处理器，其中主处理器将针对分组采取规则所指示的特定操作。所述操作可以是诸如丢弃分组或断开连接或通知 IT 管理人员或之前所述的操作之类的多个操作中的一种操作。

图 85 示出了运行时自适应搜索处理器阵列。运行时自适应搜索处理器阵列类似于图 64 中示出的运行时自适应处理器以及上述的运行时自适应处理器。RAP 和搜索处理器阵列的各个块的功能是相似的。搜索处理器阵列示出图 64 中从 6401 (1) 到 6401 (Z) 的每个计算群集都可以是计算群集或搜索群集或它们的组合, 如块 8501 (1) 到 8501 (Z) 所示。如果所述群集为计算群集, 则它的功能与图 64 和 65 中介绍的功能相同, 但如果群集是搜索群集, 则它的功能是不同的, 如图 85 中所示并在下面进行描述。运行时自适应搜索处理器阵列可以替代图 62 和 63 中的运行时自适应处理器使用, 并向自适应 TCP/IP 处理器提供运行时自适应搜索处理和运行时自适应计算处理能力, 从而实现上述功能以及可同时利用大量规则搜索的深度分组搜索、应用层搜索、上层 (第四层到第七层) 安全性等功能。全局存储器和全局存储器控制器 8502 提供的功能与针对上面的块 6402 描述的功能类似, 它还可以与图 84 中示出的运行时自适应搜索处理器的全局和配置存储器 8406 进行交互。

图 86 示出了实例搜索群集。搜索群集类似于图 65 中示出的计算群集, 但是, 搜索群集包括搜索引擎而不是包括计算元件。搜索引擎可以是图 82 中示出的 DFA 搜索引擎或图 80 中示出的 NFA 搜索引擎。搜索群集可以包括基于 DFA 的搜索引擎和基于 NFA 的搜索引擎的组合, 也可以仅包括基于 DFA 的或基于 NFA 的一类搜索引擎。可以在实现搜索处理器之时做出选择来实现各种设计优化, 例如在搜索群集内保持规则结构, 这可以实现高效的面积利用、电力网格、时钟网络等。对于本领域的技术人员而言, 通过本申请的教导, 搜索群集的各种配置是显而易见的并且不可能介绍所有的配置。搜索群集可以包括搜索引擎、群集路由交换机 8605、群集存储器和流控制器 8606 或群集配置控制器 8607 或它们的组合。群集存储器和流控制器 (块 8606) 包括群集存储器、群集存储器控制器和群集流控制器。搜索处理器所包括的搜索群集可以有些基于 DFA, 而另一些基于 NFA 或基于 DFA 和 NFA 的组合。因此, 可以创建一系列提供各种数量的上述基于 DFA 和基于 NFA 的群集的产品。搜索引擎 8601 (1) 到 8601 (Z) 可

以根据需要彼此互连来创建大于单个搜索引擎 (SE) 所支持的规则数的 RE 规则。还可以创建全部作为图 75 中示出的 RE 规则组的一部分的 SE 组。属于一组的 SE 可以由控制器 8707 和 8709 启用或禁用, 具体取决于当前启用的 SE 组所搜索的结果。针对搜索引擎中配置的规则需要被搜索的内容由搜索处理器阵列 (图 85) 通过群集配置控制器 8607、群集路由交换机 8605 以及群集间路由交换机 8503 提供给搜索引擎。搜索结果还可以通过类似的路由被从搜索群集传输到控制处理器和调度器 8402。搜索结果还被传输到群集存储器和流控制器 8606 以及群集配置控制器 8607, 所述控制器可以被配置为按照匹配规则的要求而采取操作。这些控制器可以将需要采取的操作通知给其他群集, 以便所有搜索群集在活动的规则组以及对于每个群集需要为活动的当前应用上下文方面相互保持同步。

搜索群集中的搜索引擎由与全局配置控制器 8505 交互的群集配置控制器 8607 进行配置。全局配置控制器用于全局配置控制并与自适应控制器 8404、控制处理器和调度器 5402、配置和全局存储器 5406 以及存储器控制器 8407 进行交互, 以接收、存储和检索用户或 IT 管理人员使用图 97 中示出的搜索编辑器流配置的规则所需的各种硬件配置信息。搜索编辑器按照搜索设备的特定搜索处理器的能力编辑规则, 然后通过主机处理器或主处理器或上面描述的其他元件与控制处理器和调度器 8402 进行交互, 以使用这些规则配置搜索处理器阵列。这些规则由与上面列出的各个块交互的控制处理器传送给群集配置控制器。群集配置控制器然后针对基于 NFA 的搜索引擎配置应用状态存储器 8001, 针对基于 DFA 的搜索引擎配置 DFA 指令和状态流存储器 8201。一旦使用相应的规则和应用上下文配置了所有群集中的所有搜索引擎之后, 便可启动内容搜索。如上所述, 可能需要搜索处理器维护流或会话信息, 以继续搜索同一流中在不同时间到达搜索处理器的分组中的内容。同样, 在需要搜索可能由主机处理器发送给搜索处理器的消息或文件或其他内容的解决方案中, 如果内容以字节块、页、段或类似方式发送时, 则可能需要维护上下文。流上下文可以在搜索处理器的全局存储器 8406 中维护, 也可以存储在与存储器控制器 8407 相连的

存储器中。当搜索引擎的数目很大时,需要针对给定上下文存储的信息量可能会造成性能问题。本申请的 DFA 和 NFA 搜索引擎对于每个上下文可能需要保存最少一个 32 位字。因此,如果搜索引擎的数目为 1024,分布在 64 个群集中,一个群集中有 16 个搜索引擎,则用户需要在全局存储器中存储并从中检索最多 1024 个 32 位字。对于需要支持 1Gbps 到 10Gbps 线路速度的搜索处理器而言,这将是很大的性能障碍。因此,对于需要高线路速度性能的应用而言,本发明允许群集在本地将流上下文信息存储在群集存储器 8606 中。对于当前实例,如果将搜索引擎组织成 4×4 阵列,并且有四个与本地存储器连接的端口,则在每个群集的基础上,需要存储和检索四个 32 位字的流上下文来进行流上下文切换。由于所有群集都能并行访问它们的本地存储器,因此可以在 8 个时钟周期内完成流上下文切换,如果必须在全局存储器中存储和检索流信息时,则需要 2048 个时钟周期才能完成切换。加载新的上下文将需要 4 个时钟周期,这是切换上下文所需的最短时间,其中当开始搜索新的上下文时,可以在后台完成被换出的上下文的存储。因此,本发明的申请可以解决需要将上下文存储在全局存储器中的体系结构中存在的主要性能瓶颈。

图 87 示出了内部搜索群集规则组和交换实例。如上所述,正则表达式规则可以如图 75 所示进行分组。本发明的搜索处理器允许对搜索引擎进行分组以实现正则表达式的规则分组。图 87 示出了两个此类规则组,所述规则组可以按照应用需要和应用上下文来创建。所示的规则组 1 (8701) 包括三个搜索引擎,而规则组 2 (8702) 示为包括至少 8 个搜索引擎。规则组中的搜索引擎数可以根据应用上下文而变化,并且规则组可以跨一个或多个搜索群集。因此,本发明可以创建非常灵活的搜索处理器。图中主要示出了限于搜索群集的规则组。图中还示出了规则组切换的实例。假设首先启用搜索引擎组 1 来搜索内容,并且规则组 2 被禁用。一旦内容搜索开始,如果规则组 1 中的某个规则找到匹配,它便通过将消息 S1 (8703) 发送给存储器控制器,然后由所述存储器控制器将类似的消息 Sn (8711) 发送给配置控制器 8709 的方式,通知群集存储器和流控制器 8707 以及群集

配置控制器 8709 发现匹配后需要执行的操作。消息 Sn 用于示出控制器之间的多个交互以简化消息流中每个步骤的视图。控制器随后将消息 S2 (8704) 发送给规则组 1 来停用组中的搜索引擎, 将消息 S3 (8705 和 8706) 发送给规则组 2 来激活规则组 2 中的搜索引擎。一旦发生规则组切换, 新激活的规则组 2 就开始针对其规则检查内容, 而规则组 1 则停止检查内容。因此可使用本发明创建嵌套的规则组。群集配置控制器 8709 可以与全局配置控制器 8505 交互, 以根据搜索结果的需要切换到驻留在其他搜索引擎 (多个) 中的另一规则组, 或者切换到跨搜索处理器的其他应用上下文。

图 88 示出了实例规则配置和上下文切换。图中示出了图 85 中的运行时自适应搜索处理器阵列, 只是在每个搜索群集中配置了不同类型的规则。块 8801 (1) 到 8801 (Z) 为与块 8501 (1) 到 8501 (Z) 类似的搜索群集。图中为每个群集示出了两个上下文。搜索群集 8801 (1) 中的搜索引擎使用 HTTP 和 SMTP 以及其他所有未示出的协议进行配置, 而 8801 (L) 使用 HTML 和 SMTP 进行配置。标签 “I:<AppName1>” 用于指示群集将 <AppName1> 应用配置为初始配置。标签 “S:<AppName2>” 用于指示群集在全局配置控制器的控制下切换到 AppName2。某些群集标记有标签 “I:WAIT”, 这表示这些群集在搜索的初始阶段未处于活动状态。但是, 这些群集可以切换到示为具有 S:<appName>” 标签的应用上下文。当按照图 88 中所示配置搜索处理器, 且初始群集组处于活动状态时, 这些群集便开始检查入站内容。在此实例中, 每个群集都搜索收到的特殊类型的应用内容, 此操作可以例如在联网系统中完成。假设入站分组属于 SMTP 应用, 则搜索群集 8801 (3) 中的搜索引擎检测 SMPT 分组并通知群集控制器, 后者通过发送消息 8817 与全局配置控制器交互, 所述消息指示找到 SMPT 内容并且其他群集应当应用可以是反垃圾邮件规则的 SMPT 规则, 或将 SMPT 内容解析成其组件等。然后, 全局配置控制器将消息 8818、8819、8820 发送给所有群集, 以使它们将其已配置的上下文切换到 SMPT 并开始将这些规则应用于收到的内容。因此, 搜索处理器阵列可以在运行时改变以执行复杂搜索操作。尽管为了简化说明, 图 87 和 88 中使用的实例非常

简单,但是如本领域中的技术人员将理解的,可以使用运行时自适应搜索处理器创建复杂得多的情况。

图 89 示出了具有内容搜索加速器的计算设备。所述计算设备可以是服务器、工作站、个人计算机、诸如交换机或路由器之类的联网设备或其他类型的计算设备。这是一种类型的配置,在此配置中,可以使用利用本发明的一个内容搜索处理器版本的内容搜索加速器。图中示出的计算设备包括一个或多个 CPU (8900 (1) 到 8900 (n))、至少一个芯片组 8902、至少一个存储器组件 8901、至少一个内容搜索加速器 8903 以及零个或多个提供其他功能的适配器。内容搜索加速器可以包括搜索处理器 8904。它还包括至少一个与所述搜索处理器相连的存储器组件 8905。可使用本发明的内容搜索处理器创建许多不同的系统配置。因此,本申请中的实例不应用作限制搜索处理器的范围,而应主要作为解释某些示例使用情况中的搜索处理器的方式。本申请的搜索处理器可用于线卡、网络适配器或网络接口控制器、存储联网卡、IO 卡、主板、控制处理卡、交换卡或其他例如联网设备的系统元件,如路由器、交换机、管理设备、安全设备、网关、虚拟设备、存储联网设备、服务器、存储阵列等。搜索处理器或其组件还可以与微处理器、网络处理器、交换芯片、协议处理器、TCP/IP 处理器、控制平面处理器、芯片组、控制处理器或作为功能块结合在这些处理器或芯片上的其他设备相连。内容搜索处理器可用于在其所结合的系统中以高线路速度执行内容检查,在所述系统中,内容搜索处理器卸载此类系统中主处理器的内容处理或帮助进行此类内容处理。可以存在这样的配置:其中还可以将多个搜索处理器结合到系统中以针对内容搜索提供扩展性能或规则数或二者的组合。搜索处理器可以根据业务结合在网线卡上,并提供上述的线路速度深度分组检查。

图 89 中示出的配置可用于电子邮件安全或即时消息安全或出站安全或外泄检测或 HIPAA 法案遵循或萨班斯-奥克雷法案遵循或格拉姆-里奇-布里雷法案遵循或 Web 安全或上述内容的组合。所列的安全能力包括反垃圾邮件、反病毒、反网络钓鱼、反间谍软件、检测/防止目录搜集攻击、检

测/防止蠕虫病毒、入侵检测/防护、防火墙等，或检测/防止泄漏保密信息、医疗信息、客户信息、信用卡号、社会保险号等或上述内容的组合。此类设备中的内容搜索处理器可使用用于上面列出的一种或多种应用的安全规则集进行配置并针对出入设备的信息提供内容搜索加速。因此，内容搜索处理器可根据业务来使用，可以在网络中的任意位置部署设备，例如接近组织网络的路由器或交换机或网关，或者在部门级或数据中心或它们的组合中部署，并针对网络的入站或出站业务流提供高速内容检查。

图 90 示出了实例反垃圾邮件性能瓶颈和解决方案。如上所述，在微处理器上使用基于 DFA 或 NFA 的搜索的内容搜索性能将导致性能低于 100Mbps。图 90 示出了反垃圾邮件应用作为实例应用来显示基于硬件的内容搜索的价值。性能数未按比例示出。图中示出了四类设备（appliance）中的四个垂直操作栈。所示的第一栈 9000 表示电子邮件设备栈。电子邮件设备通常包括驱动设备上的硬件设备的设备驱动器、带有操作系统（OS）的其他功能的联网协议栈以及邮件传输代理（MTA），它们与其他应用软件一起通常为软件组件。当今通常用于电子邮件设备的服务器能够跟上近 1Gbps 的线路速度，并凭借其高性能的处理器执行应用功能。

通常需要 1GHz 的处理器来处理 1Gbps 的线路速度业务以实现网络协议栈处理。由于如今的最新处理器约为 4GHz，因此服务器可以处理网络业务并具有可供执行服务器上运行的操作系统和应用的其他需求的处理能力。因此，在高端服务器上运行的电子邮件设备栈 9000 应能跟上高线路速度。由网络世界杂志在 2004 年 12 月完成的研究“Spam in the Wild: Sequel”显示了大量反垃圾邮件软件和设备供应商的性能比较。在它们的配置中，所列出的供应商产品的邮件处理性能的范围从每秒 5 封邮件到每秒 21 封邮件。当使用每个消息 10,000 个字符的网络世界所用的最坏邮件大小将此性能数转换为线路速度性能时，线路速度性能将低于 2Mbps 持续性能。所有的供应商（无论是软件还是设备解决方案）都基于双 Xeon 处理器服务器。因此，可以处理 1Gbps 的线路速度业务的服务器，当执行反垃圾邮件应用时，它的性能将降到 10Mbps 以下。出现这种情况的原因是多数反垃圾邮

件供应商所广泛使用的一种功能是根据一组规则来搜索邮件，所述规则通常表示为正则表达式。反垃圾邮件设备栈 9001 示出了其上加载有反垃圾邮件能力的电子邮件设备。反垃圾邮件应用通常执行许多基于复杂正则表达式规则的过滤以及统计过滤、基于信誉的过滤等。反垃圾邮件规则通常按顺序逐一应用于每封入站电子邮件以查找与邮件内容匹配的规则。然后，反垃圾邮件应用对匹配规则进行评分，然后根据它收到的总分判定该邮件是否为垃圾邮件。此类操作使得栈性能需求基本上增长为高于一般的电子邮件设备栈，其中反垃圾邮件过滤器 9005，设备性能的开销基本上将反垃圾邮件服务器设备的整体性能降为低于 10Mbps。本发明的内容搜索处理器可以在此类反垃圾邮件设备中使用以实现重大性能改进。硬件加速的反垃圾邮件设备栈 9002 示出了使用本发明的搜索处理器对系统整体性能的影响。在这种情况下，可以在搜索处理器 9006 上配置所有反垃圾邮件过滤器（9011 到 9013），然后使用所述搜索处理器检查每封入站邮件。因为将同时搜索所有规则，所以基于搜索处理器的设备将达到 1Gbps 的线路速度性能或更高，这是因为主机 CPU 将从执行密集正则表达式搜索中解放出来。图 89 中示出的计算设备可以是此类配置，其可用作反垃圾邮件设备以获取比基于标准服务器的反垃圾邮件设备高出多个数量级的性能。栈 9003 示出了增强的消息传送设备栈，该栈可使用从主机 CPU 卸载协议处理的 TCP/IP 卸载处理器以及本发明的内容搜索处理器。因此，可以将大量 CPU 带宽用于其他应用，所述应用否则不可能在不对性能产生重大影响的情况下在计算设备上执行。可分别或同时使用 TCP/IP 卸载处理器和内容搜索处理器，并且使用其中一个时并不要求同时使用另一个。TCP 卸载处理器和内容搜索处理器可以位于提供网络连接性和加速的同一设备上。

图 91 示出了具有反病毒性能瓶颈的反垃圾邮件。此图与图 90 非常相似，只是示出其栈的反垃圾邮件设备还支持反病毒能力。反病毒搜索不同于反垃圾邮件搜索，但它们同样增加了显著的性能开销，如栈 9104 所示。尽管在使用本发明的内容搜索处理器（多个）时，也可以如 9105 所示显著降低反病毒开销，但是反病毒的过滤器数远多于反垃圾邮件的过滤器数。

图 92 示出了应用内容搜索性能瓶颈和解决方案。对于需要内容搜索，但在主机处理器上执行搜索的大量应用而言，本发明的内容搜索处理器可用作搜索加速器。由于用于内容搜索的主机处理器的性能如上所述不是很高，因此，基于内容搜索处理器的加速器可以显著增加这些应用的性能。需要内容搜索的应用非常多，例如数据仓库应用、数据库应用、与生物信息相关的应用、与遗传学、蛋白质组学、药物开发相关的应用等。图中示出了三个块 9200、9201 和 9202，它们表示就主机 CPU 负载而言基于内容搜索的应用性能。传统的应用在服务器或工作站或个人计算机上运行，并执行内容搜索，夹杂以应用需要执行的其他任务。如果应用执行大量搜索，则应用的搜索部分的性能需求可能会显著高于其他部分。通过将应用 app1 和 appN 的内容搜索部分 9203 和 9205 分别与这些应用的其他代码 9204 和 9206 相比可以看出这一点。9200 中的栈是当前或现有技术用于内容搜索应用的解决方案。尽管所述栈针对内容搜索部分和其他代码部分示出了连续的栈，但是实际执行通常包括搜索同时间以其他代码功能。在执行此功能的计算设备上使用本发明的内容搜索处理器和加速器时，可以使应用利用处理器的搜索能力并显著加速应用性能（与没有搜索加速支持的计算设备相比）。9201 中的栈示出了当转换为利用内容搜索加速器时，对 CPU 负载以及应用所花费的结果时间的影响。栈 9203 和 9205 可以使用明显较少的负载和时间，分别如栈 9207 和 9208 所示。类似地，通过如 9209 所示卸载 TCP/IP 协议处理，可以进一步增加系统的性能。如上所述，TCP/IP 卸载和内容搜索卸载相互独立，无需对方可在系统中执行。但是，用户也可以将内容搜索处理器和 TCP/IP 处理器一起用作单独组件或在同一器件上使用来获得性能优势。

图 93 示出了实例内容搜索 API 使用模型。如上所述，内容搜索处理器可用于加速一般应用的内容搜索部分。为了轻松创建新的应用并迁移现有应用来利用此类搜索处理器加速能力，用户可以针对内容搜索创建应用编程接口（API）。图 94 中示出了实例内容搜索 API 并在下文中进行介绍。内容搜索 API 可以位于用户级或具有用户级调用的内核级，或者同时位于

两者中。图 93 示出了位于用户层的内容搜索 API 9307。内容搜索 API 提供 API 功能，任何应用都可以调用此功能来获得内容加速优势。应用使用内容搜索 API 存在使用约定。例如，应用在运行之前需要设置可以使用 API 调用在搜索处理器上配置的搜索规则，或者需要动态地创建规则并以相应的格式设置这些规则，以便可以使用 API 在内容搜索处理器上配置它们或需要同时执行这两种操作。可以存在根据硬件系统、操作系统或搜索处理器或它们的组合而建立的 API 调用约定。然后，应用按照 API 约定进行编码并获得搜索处理器的加速优势。图中示出了与内容搜索 API 9307 协作来使用 9312、9313 和 9314 所示的逻辑接口路径访问内容搜索加速硬件 9317 的应用 1 (9300) 到应用 N (9303)。内容搜索 API 通过在内核中运行的端口驱动器访问内容搜索加速器所提供的服务和资源。应用可以直接通过此接口传送要搜索的内容，或者将要搜索的内容以及根据需要设置的表放入应用的缓冲区 9304、9305，然后通过内容搜索 API 指示内容搜索处理器从这些缓冲区中检索信息。所述 API 可以将这些缓冲区映射到内核空间，以使搜索 API 的端口驱动器可以将其提供给内容搜索处理器，或者可以使搜索处理器硬件对缓冲区进行直接存储器访问。搜索处理器可以将内容存储在片上或片外存储器缓冲区 9318 中，然后对内容执行所请求的搜索操作。一旦完成搜索，就使用门铃机制或回调机制将搜索结果提供回应用或数据缓冲区或操作系统模型所允许的存储位置。内容搜索 API 还提供轮询机制，应用可使用此机制检查和/或检索搜索结果。

图 94 示出了带有示例函数的实例内容搜索 API。此图示出了一组可以是实例内容搜索 API 一部分的函数。虽然函数列表可以多于或少于示出的函数，但是函数提供了使应用能够利用 API 使用内容搜索硬件的基本集合。示例函数没有示出 API 函数调用的输入、输出或返回参数，这些参数可以取决于操作系统、调用约定等。应用可以使用 API，方法是首先查询硬件引擎的能力，然后使用内容搜索处理器通过内核或用户模式或其组合与应用及其资源通信可能需要的相应规则、指针、权限等初始化硬件引擎。应用可以将特定规则设置为可在搜索处理器中配置的 DFA 规则或 NFA 规

则。可以给予应用对多个上下文的访问权限，应用可利用这些上下文执行基于上下文的搜索。使用所有必要规则适当设置内容搜索处理器之后，应用便可以开始对其内容执行搜索。应用可以使用 API 并通过此接口发送字节流，将要搜索的内容直接传送到搜索处理器。可以存在诸如 `sendData()` 之类的多个版本的 API 函数（未示出），应用可以使用此函数开始将数据发送到搜索处理器，启动搜索以及指示搜索处理器应在何时停止搜索。执行搜索的一种更高效的方法可以是应用填充要搜索的一个缓冲区或一组缓冲区，然后为搜索处理器提供到缓冲区（多个）的指针（多个），当搜索处理器接收调用以使用诸如 `startHWsearch()` 之类的 API 调用启动搜索之后，它便可以使用已配置的规则开始搜索缓冲区。搜索处理器可能已被初始化为将搜索结果通过多种机制之一传送到应用，所述机制例如将结果复制到结果缓冲区，或将结果存储在与搜索处理器关联的存储器中，或调用应用注册到操作系统的回调函数等。搜索处理器还可以使用门铃机制与应用通信以通知它搜索已完成。存在许多不同的传送信息的方法（如前所述）并且可以取决于操作系统和系统硬件体系结构。还可以具有可通过诸如 `isSearchDone()` 之类的 API 函数（未示出）提供的轮询机制，此机制可以将特定搜索是否已完成的查询答案提供给搜索硬件。如果硬件对应用的回答是搜索已完成，则应用可以使用诸如 `getRes()` 之类的 API 调用请求特定结果，或应用可以使用诸如图 94 中所示的 `getResPtr()` 之类的调用请求指向可以保存结果的缓冲区的指针。应用完成特定搜索或不再使用搜索处理器之后，应用可以调用 API 函数 `stopHWsearch()` 以使硬件处理器停止针对此应用执行搜索。还可以具有诸如 `removeAppContext()` 之类的 API 调用（未示出），应用可以调用它向 OS 和搜索处理器硬件指示应用已使用搜索处理器完成搜索，并且因此搜索处理器硬件可以释放其所有关联上下文以供其他可能需要搜索处理器资源的应用使用。还可以具有诸如 `setRuleGroup()`、`selectRuleGroup()`、`setInitGroup()` 之类的其他硬件功能专用 API 调用，它们可以允许应用使用先前所述的规则分组机制来创建规则组及其执行顺序。如前所述，对本领域的技术人员显而易见的是，根据

本发明的教导，可以创建更多函数以及 API 函数的变型，以便使用硬件搜索处理器实现常规内容搜索应用加速。因此可以创建内容搜索 API 以便为常规应用提供内容搜索能力。虽然通过实例（其中应用在启动搜索之前设置要使用的规则）给出了如上描述，但是可以在搜索正在进行时动态更新在搜索处理器中配置的规则集（方法是通过添加、删除和/或修改已配置的规则），以便应用针对任何将来搜索开始使用更新后的规则集。

图 95 示出了使用搜索处理器的实例应用流（静态设置）。此流程示出了用于设置规则和硬件处理器的静态过程，但是如上所述，如对本领域的技术人员显而易见的，也可以进行动态设置。此流程可以允许应用在运行时执行时在处理器中添加/删除/修改规则以实现动态流。此示例提供了一种机制，其中可以使用内容搜索规则以及使应用能够使用内容搜索处理器的 API 调用来更新现有应用或新的应用。应用源（9500）可以被更新（9501）以创建包含针对内容搜索进行的修改的应用源，其中内容搜索规则如编辑器编码约定预期那样，可以在独特代码段中设置或可以被清楚地标记（9502），然后内容搜索型编辑器（9503）编辑这些规则。编辑器生成目标代码（9504），并且在段中编辑内容搜索规则，加载器可以使用这些规则以便在搜索处理器中配置它们。然后将应用目标代码分发到内容搜索处理器的客户或用户以加速应用的搜索性能。可以使用互联网、万维网、企业网络或其他网络，或使用其他装置（如 CD、DVD），或其他可用于加载应用的计算机存储装置以电子方式分发应用代码。应用更新（9501）可以根据需要手动或使用工具或采用这两种方式执行。分发的目标代码（9506）由加载器（9507）或操作系统提供的使用应用的类似函数读取，并进行设置以便在系统上执行。加载器或其他函数可以使用一组 API 调用或端口驱动器或其他 OS 函数或其组合，以便通过应用需要的相应规则（如块 9508 所示那样在目标代码中编码）配置内容搜索处理器。设置搜索应用硬件并且保留或设置应用需要的其他资源之后，OS 便启动应用（9509）。如果在内容搜索之前需要，应用可以执行任务（9510）。应用然后可以设置需要搜索处理器搜索的内容（9511）。然后它便启动搜索处理器以执行

搜索(9513)。搜索完成之后,它可以检索结果(9514)。当搜索处理器进行搜索时,应用可以继续在此系统的主 CPU 或其他元件上执行其他任务(9512)。如果应用执行完毕,则应用可以退出,否则应用可以根据需要继续执行,其中可以执行包括新搜索的更多任务。此流程图将任务执行示为从 9515 到 9510 的循环,尽管不同时间内通过此循环执行的任务可以截然不同。此循环并不表示重复相同的代码序列。它旨在表明可以重复任务类型。此外,应用流中无需存在从 9510 到 9515 的所有任务。应用执行完毕之后,它可以释放其使用的所有资源以及用于内容搜索处理器的资源。

图 96 示出了 FSA 编辑器流。正则表达式形式的内容搜索规则被用作 FSA 编辑器流程的输入。FSA 编辑器在所有规则或规则组内进行搜索,并分析它们(9602)。然后编辑器生成 NFA 和 DFA 以满足内容搜索处理器的所实施体系结构,如步骤 9603 和 9604 所示。编辑器然后比较将 RE 实现为 NFA 与 DFA 的成本,并根据处理器能力选择 NFA 或 DFA,如步骤 9606 和 9611 所示。编辑器生成可能带有 RE 或 RE 组所需的操作和/或标记的 NFA 或 DFA 代码。编辑器从步骤 9602 继续到 9607,直到所有 RE 规则均被处理。它然后生成整个 RE 规则集所需的所有规则 FSA 代码和标记(如 9608 所示),并将规则存储在存储装置内的规则数据库(9609)中,所述存储装置可以是 RAM、ROM、硬盘、软盘、CD、DVD 或任何其他计算机存储装置。然后可以将已编辑的规则分发到内容搜索处理器,如步骤 9610 所示。可以通过上述用于安全编辑器流的分发引擎来分发规则。

图 97 示出了搜索编辑器流(完全和增量规则分发)。当定义完全规则集,或对规则集进行任何更新或修改并且需要在搜索处理器中传送和配置对规则集进行的增量更改时,可以使用此流程分发搜索规则或安全规则。图 97 中示出的流程非常类似于图 67 中所示的流程,但上述 FSA 编辑器流是图 97 中的流的一部分。诸如应用层规则、网络层规则或存储网络层规则或任何其他搜索规则之类的规则可以使用手动或自动方式创建,并以预先定义的格式作为输入提供给搜索编辑器流。搜索编辑器的规则分析器(9704)分析规则并将它们转换为正则表达式格式(如果规则尚未采用此

格式)。然后正则表达式规则通过图 96 中所示的 FSA 编辑器流。FSA 编辑器的输出是编辑到具有内容搜索处理器的节点的节点能力并存储在规则数据库中的规则。规则分发引擎检索规则数据库中的规则并将其分发到具有搜索处理器的一个或多个相应节点。规则分发引擎(块 9709)使用类似于图 57 和图 58 中所示的中央管理器流和规则分发流将规则分发到相应的节点。可以将搜索或安全规则分发到如图 58 中所示的主机处理器或控制平面处理器或分发到上面所述的控制处理器和调度器(块 8402)或它们的组合,具体取决于节点能力。可以根据特定节点的能力,通过网络使用采用专用或标准协议的安全链路或不安全链路分发规则。所述网络可以是局域网(LAN)、广域网(WAN)、互联网、城域网(MAN)、无线 LAN、存储区域网络(SAN)或系统区域网络或部署的其他网络类型或它们的组合。所述网络可以基于以太网、基于网际协议或基于 SONET 或基于其他协议或它们的组合。

图 98 示出了用于 FPGA 的 FSA 合成与编辑器流。此图中示出的流类似于图 96 中示出的 FSA 编辑器流。步骤 9801 到 9811 基本上与步骤 9601 到 9611 相同。用于 FPGA 的 FSA 合成与编辑器流示出了几个附加步骤,它们用于转换 FSA 规则以便在 FPGA 中合成用于它们的内容搜索处理器。转换为 NFA 或 DFA 的 RE 规则用于以自动方式生成相应的 RTL,如 9812 所示。NFA 或 DFA RTL 生成可以使用 9817 所示的库中的 NFA 和 DFA 构建块 RTL。当通过所有 RE 规则针对搜索处理器阵列的所有元件生成 RTL 之后,可以使用用于已编辑规则的 NFA 或 DFA 专用搜索引擎生成搜索处理器的顶级 RTL。然后将输出 RTL 与搜索处理器(例如存储器控制器)的任何其他 RTL 代码合并,并且可以将输出存储在计算机存储装置中。然后将此合并后的 RTL 作为输入提供给 FPGA RTL 合成工具,所述合成工具生成基于 FPGA 的搜索处理器以专门满足输入到 FSA 合成与编辑器流的正则表达式规则的需要。

在各种可能实施例的所选实施例中,可以使用制造工艺将本发明的处理器制造成硬件产品而没有任何限制,如下具体所述。可以针对用于特定

目标制造工艺技术的功能、计时以及其他设计和制造约束，在各种芯片设计抽象级别（如 RTL 级别、电路/示意图/门电路级别、布局级别等）设计和验证所述处理器。相应物理/布局级别处的处理器设计可以用于创建掩膜集合，以便用于以目标工艺技术制造芯片。然后使用掩膜集合通过用于所选工艺技术的步骤构建处理器芯片。所述处理器芯片然后可以根据需要经历测试/封装过程以确保所制造的处理器产品的质量。

虽然参考本发明的特定实施例进行了上述说明，但是本领域的技术人员将理解，可以在这些实施例中作出更改而不偏离本发明的原理和精神。

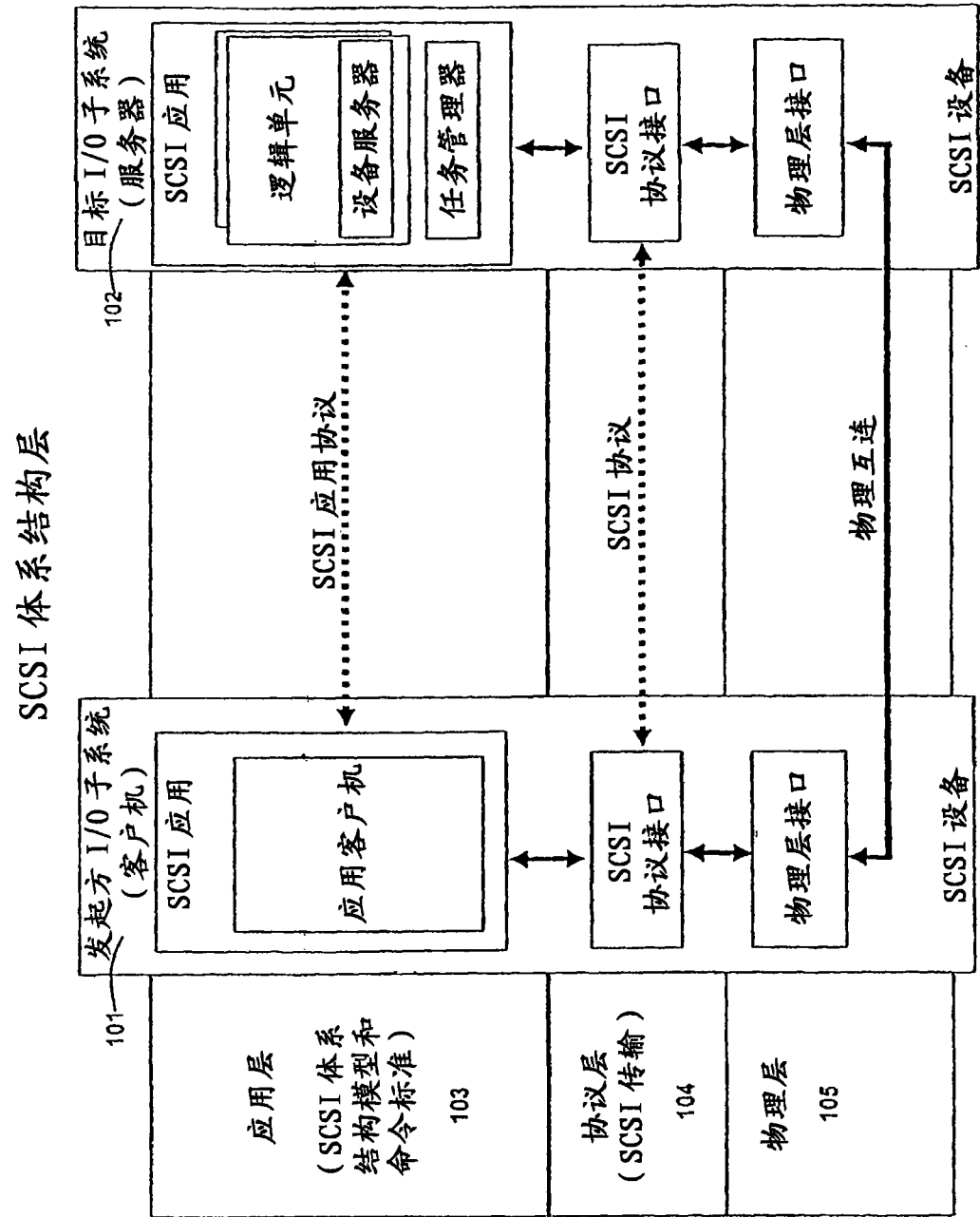


图 1

iSCSI 上的 SCSI 体系结构层

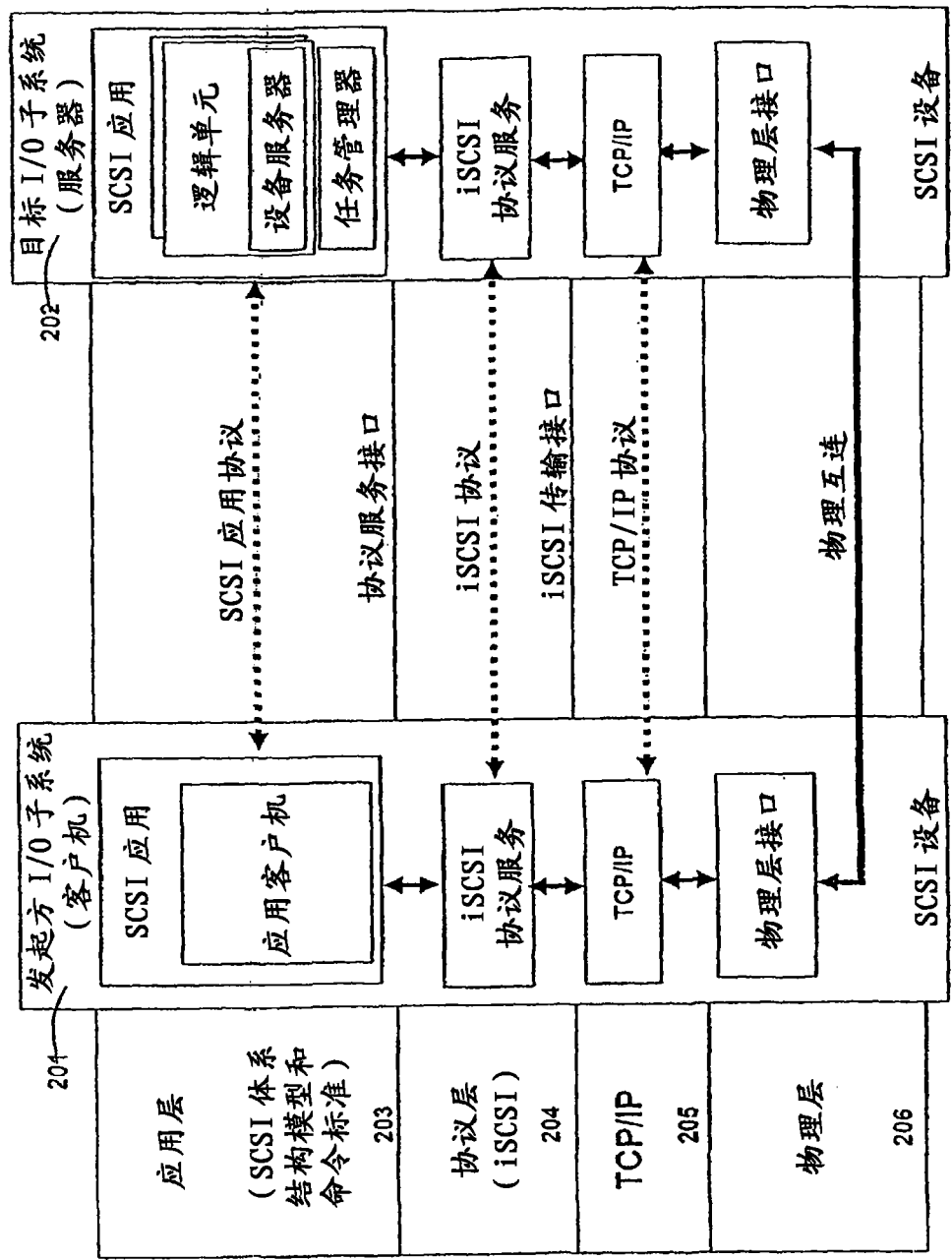


图 2

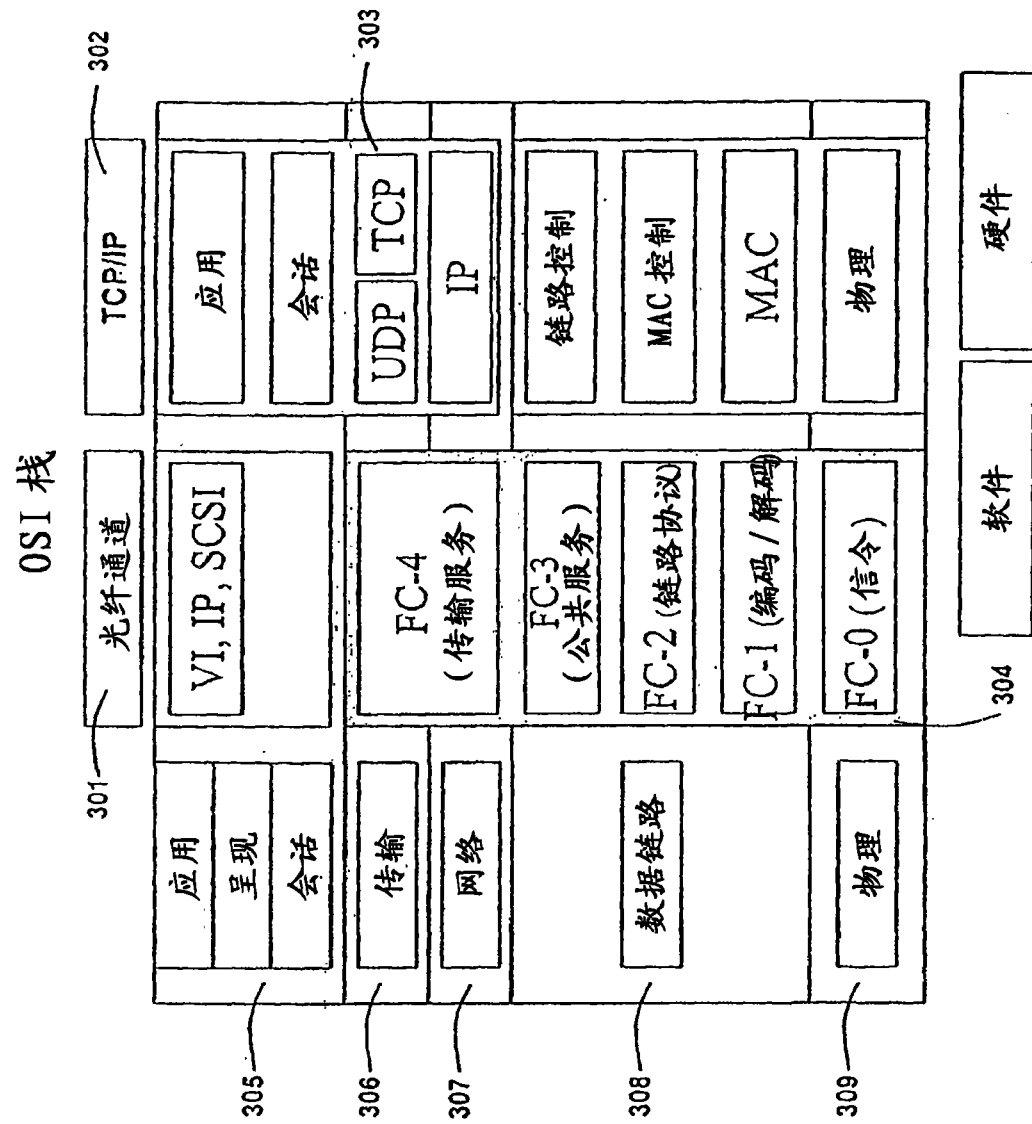


图 3

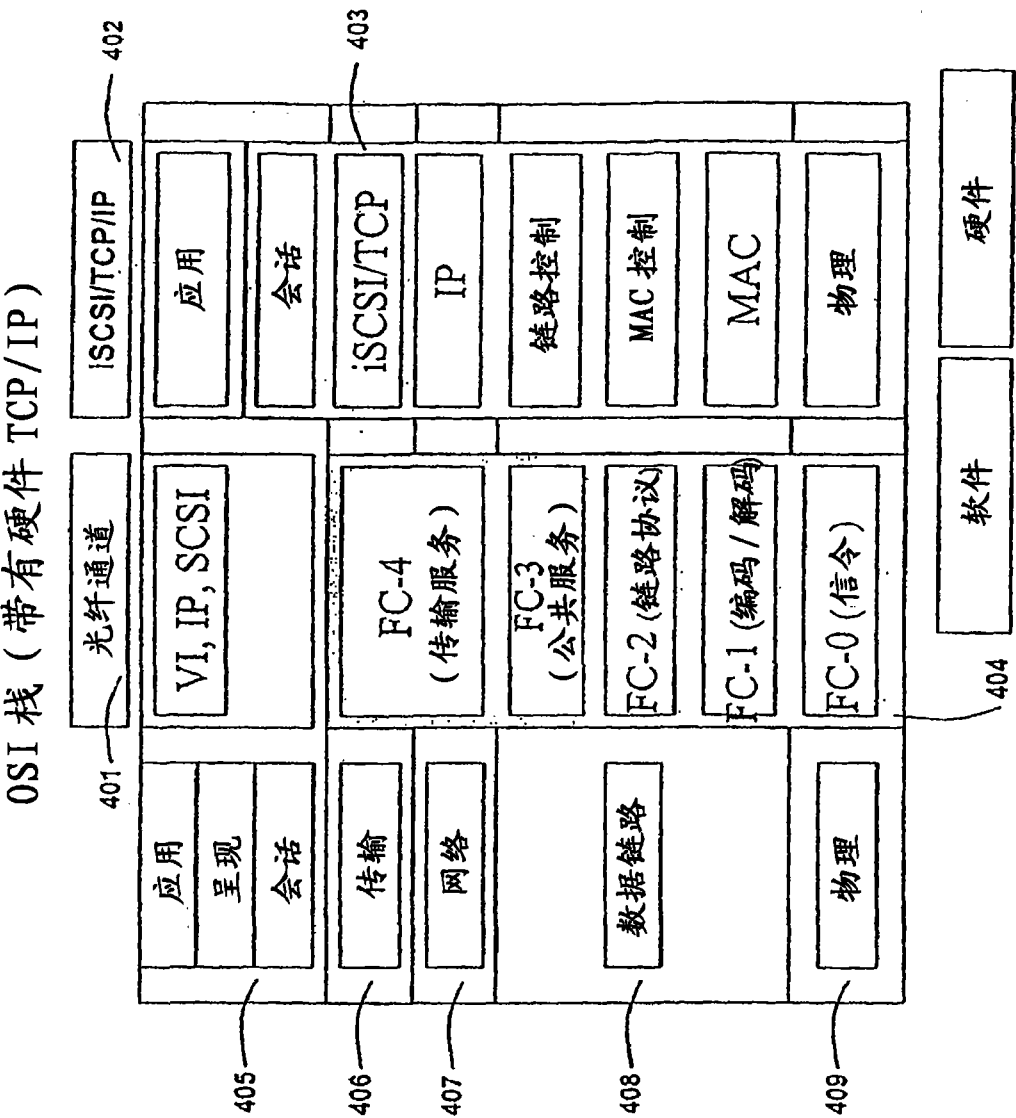


图 4

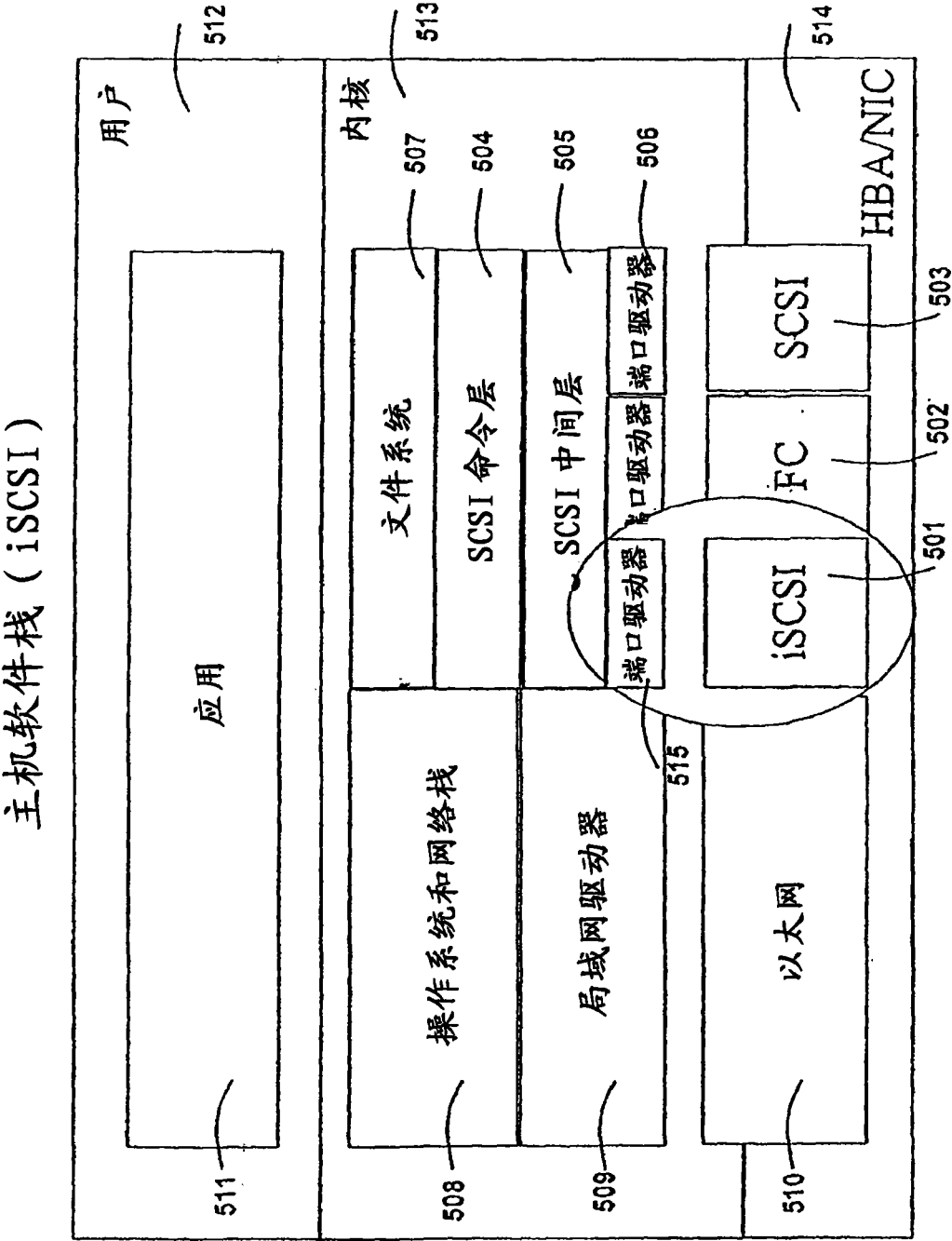
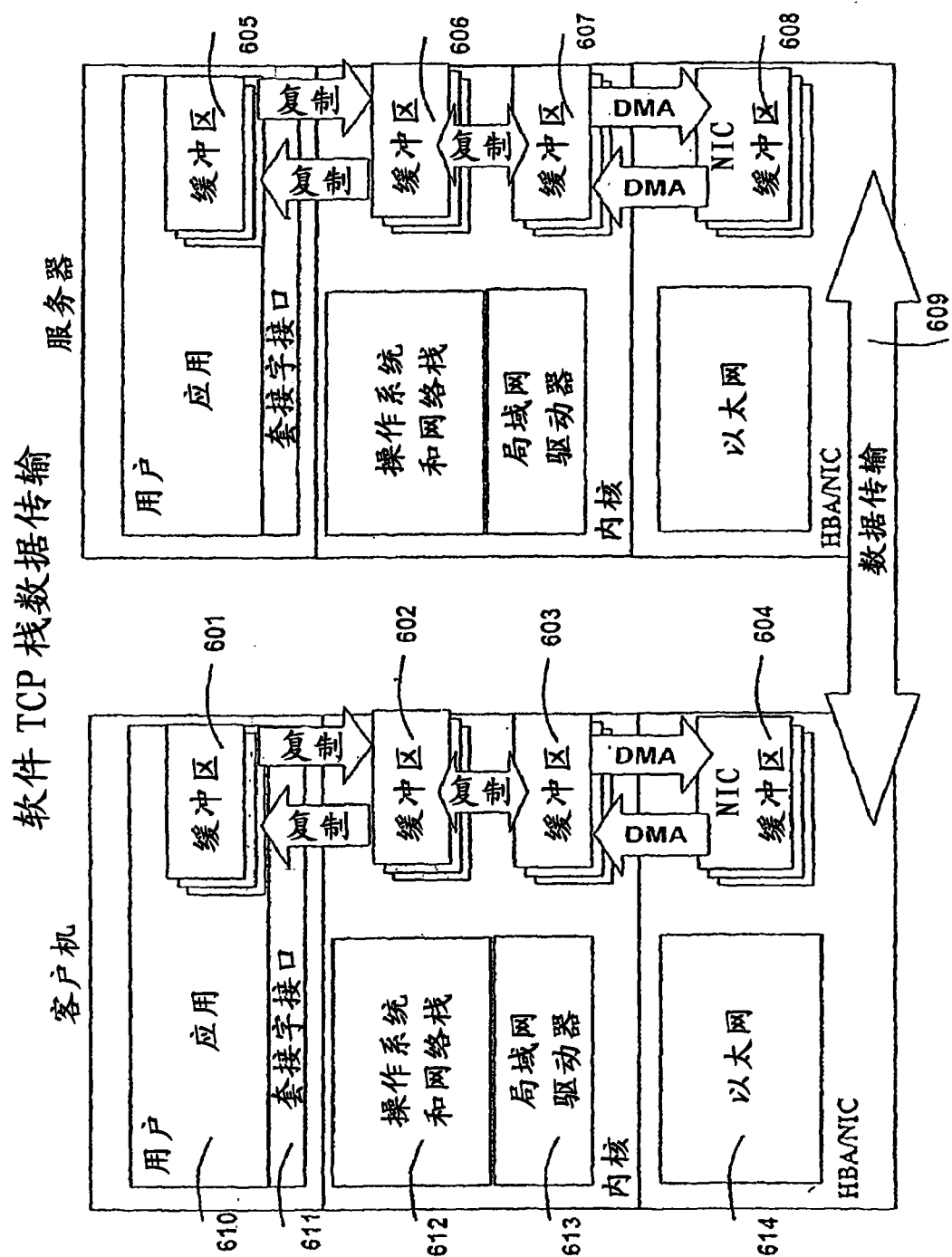


图 5



远程直接存储器存取

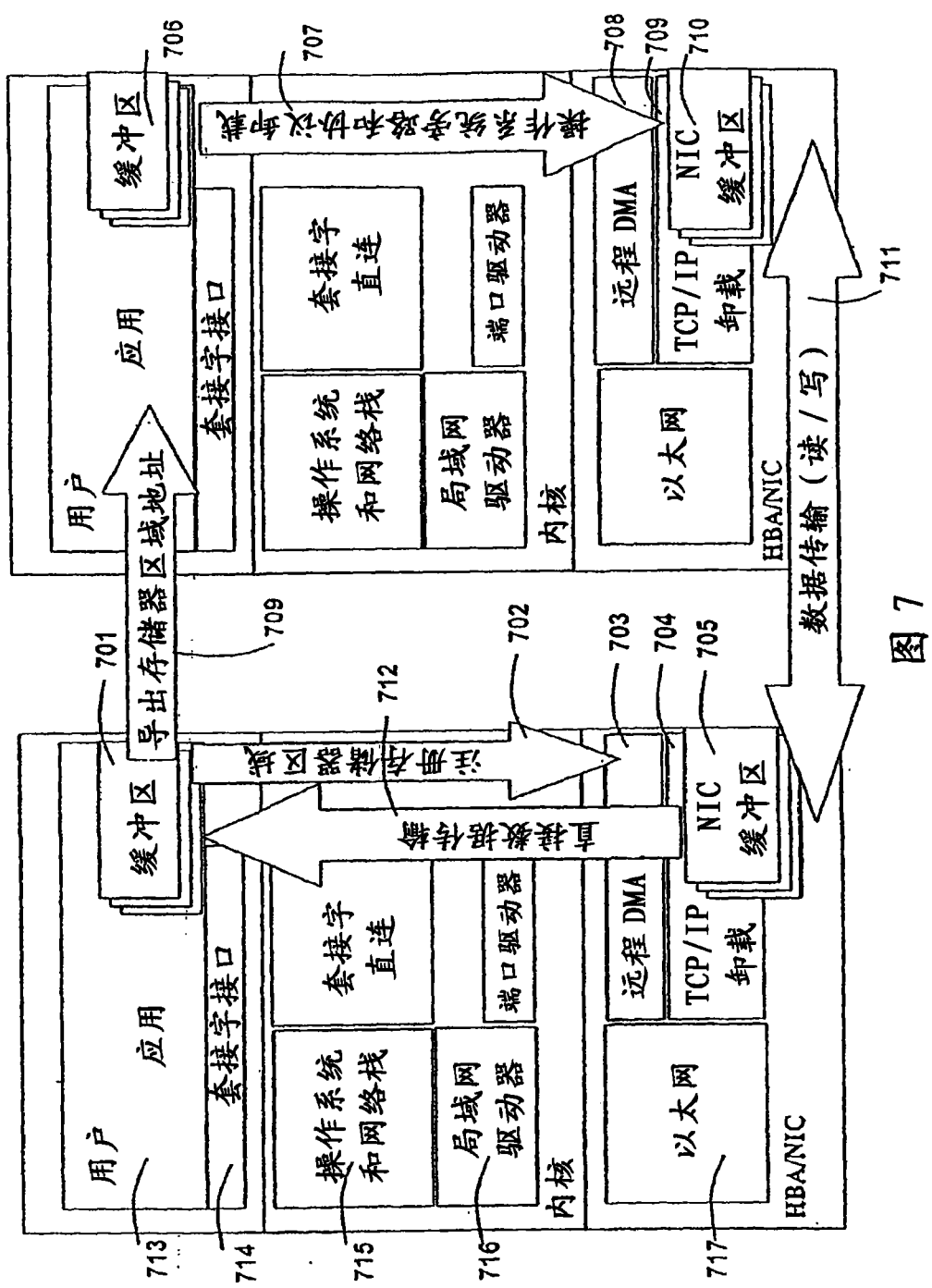


图 7

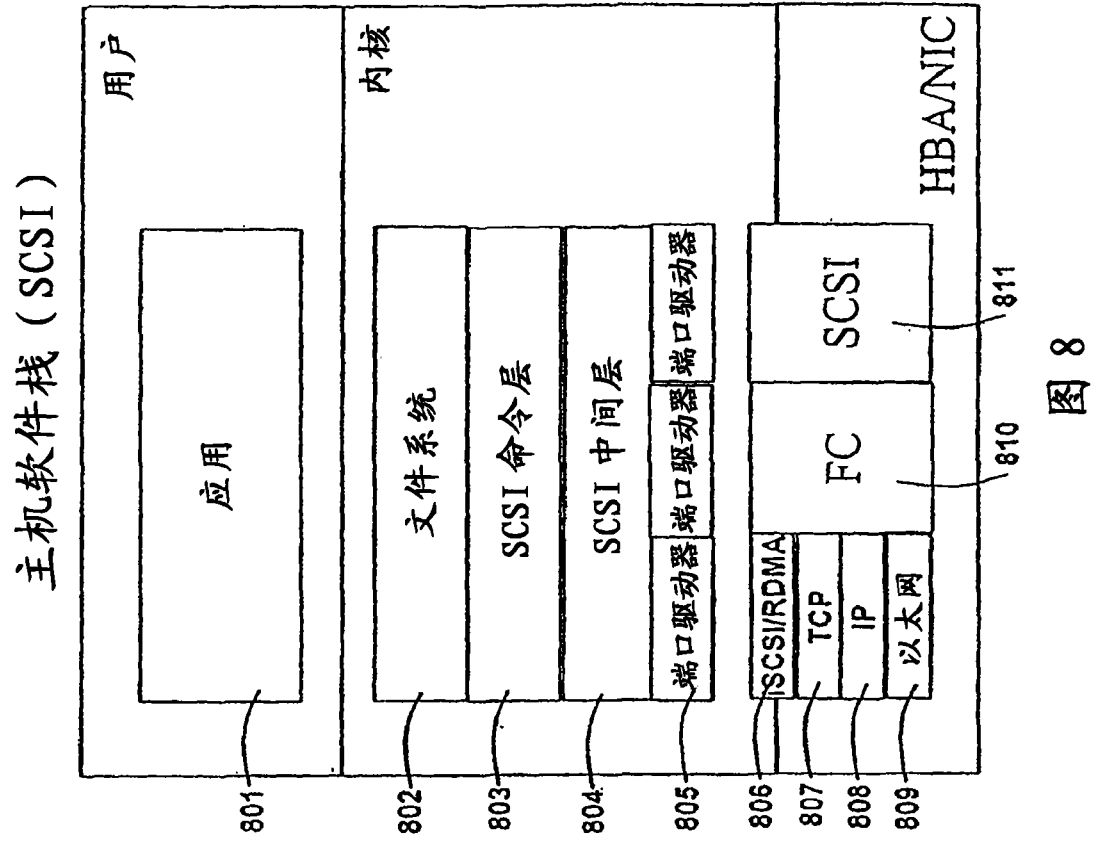


图 8

iSCSI 栈

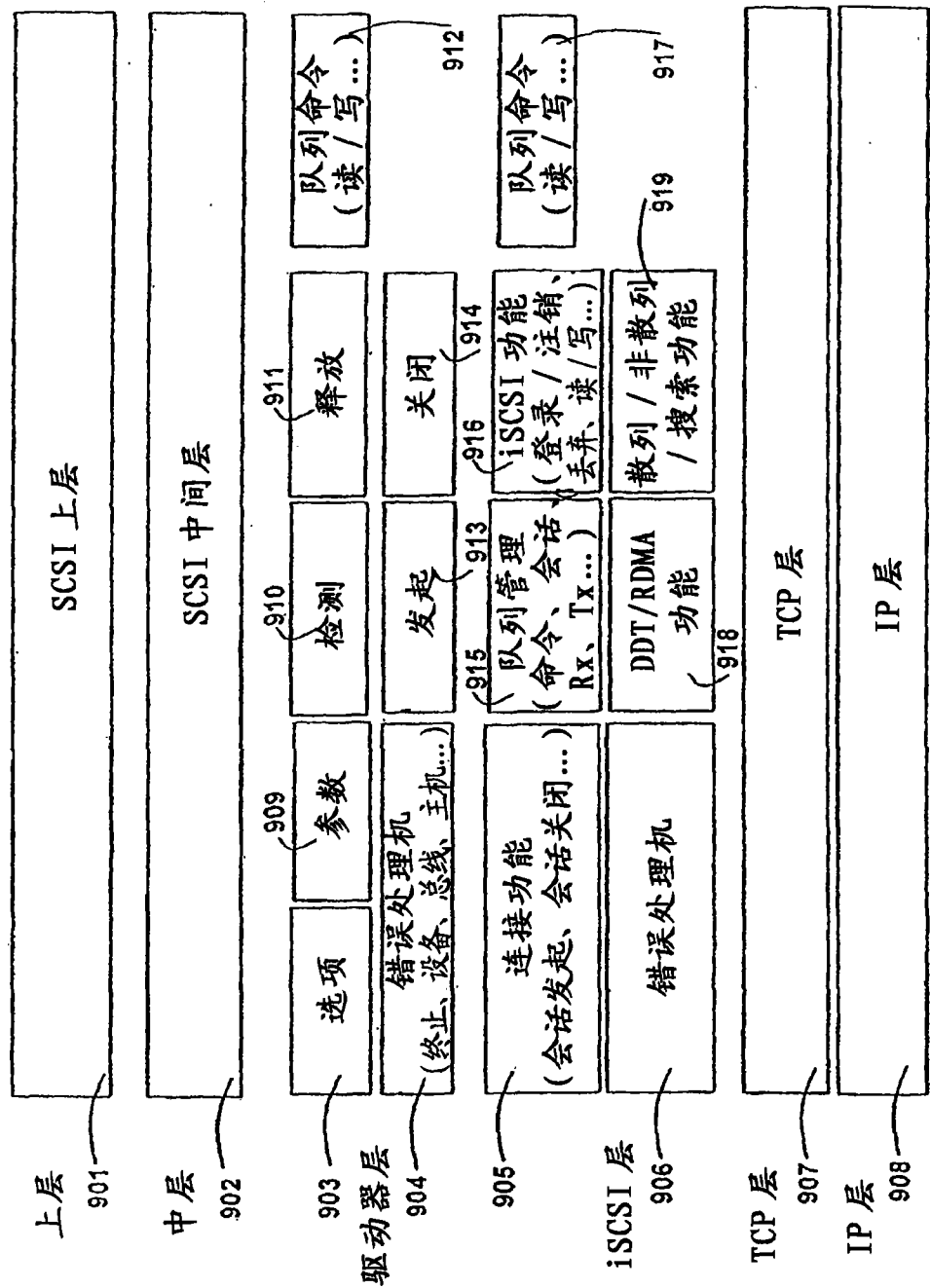


图 9

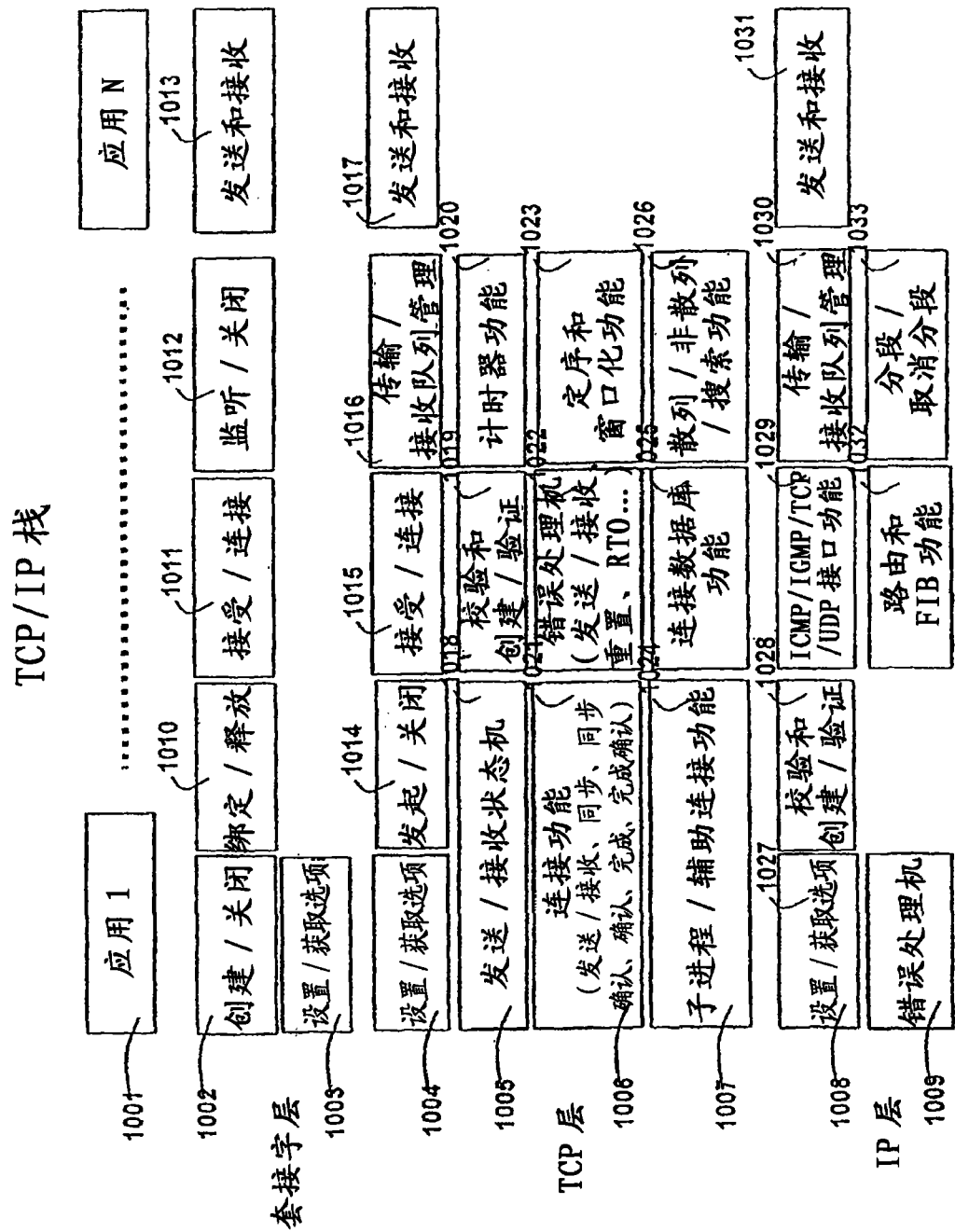


图 10

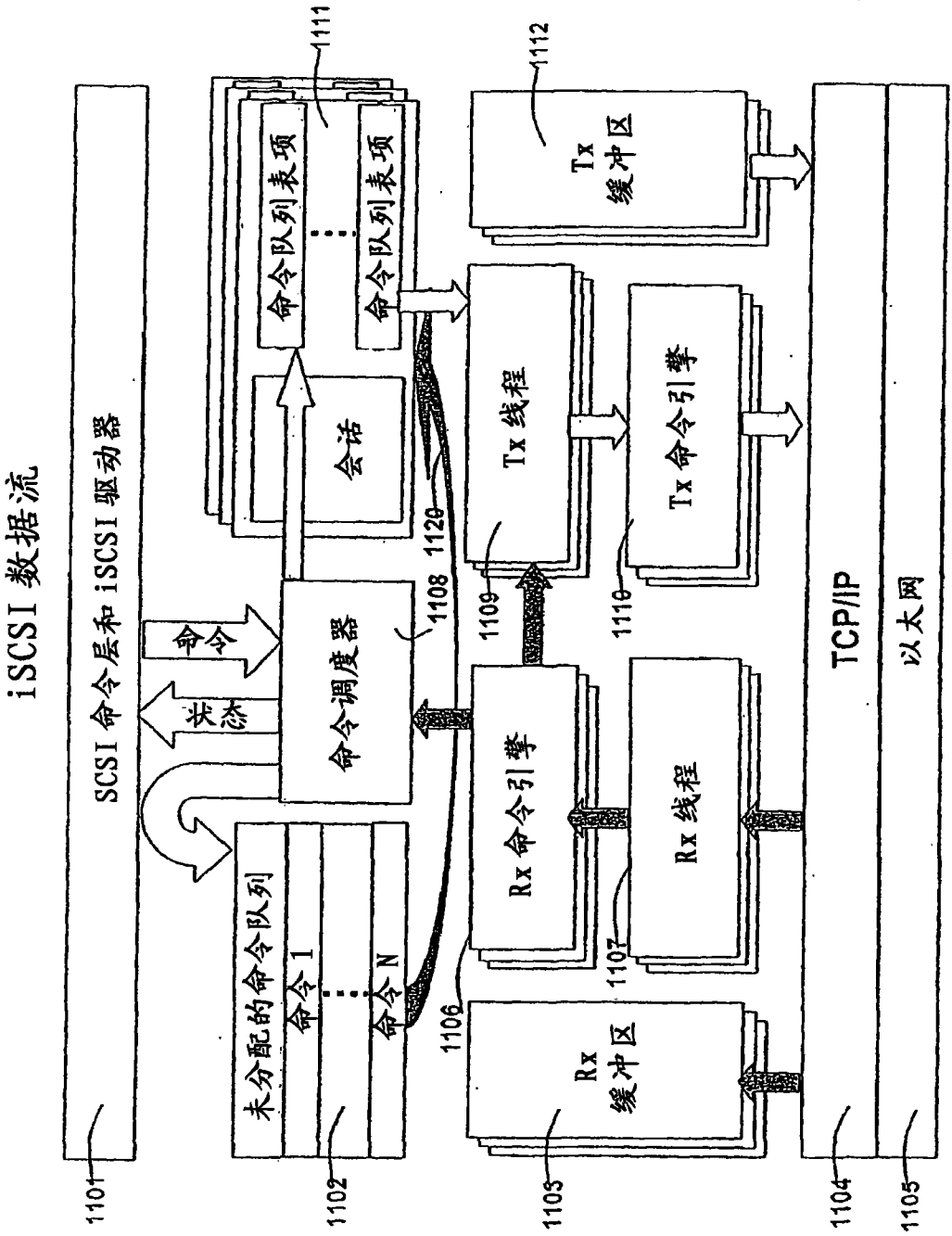


图 11

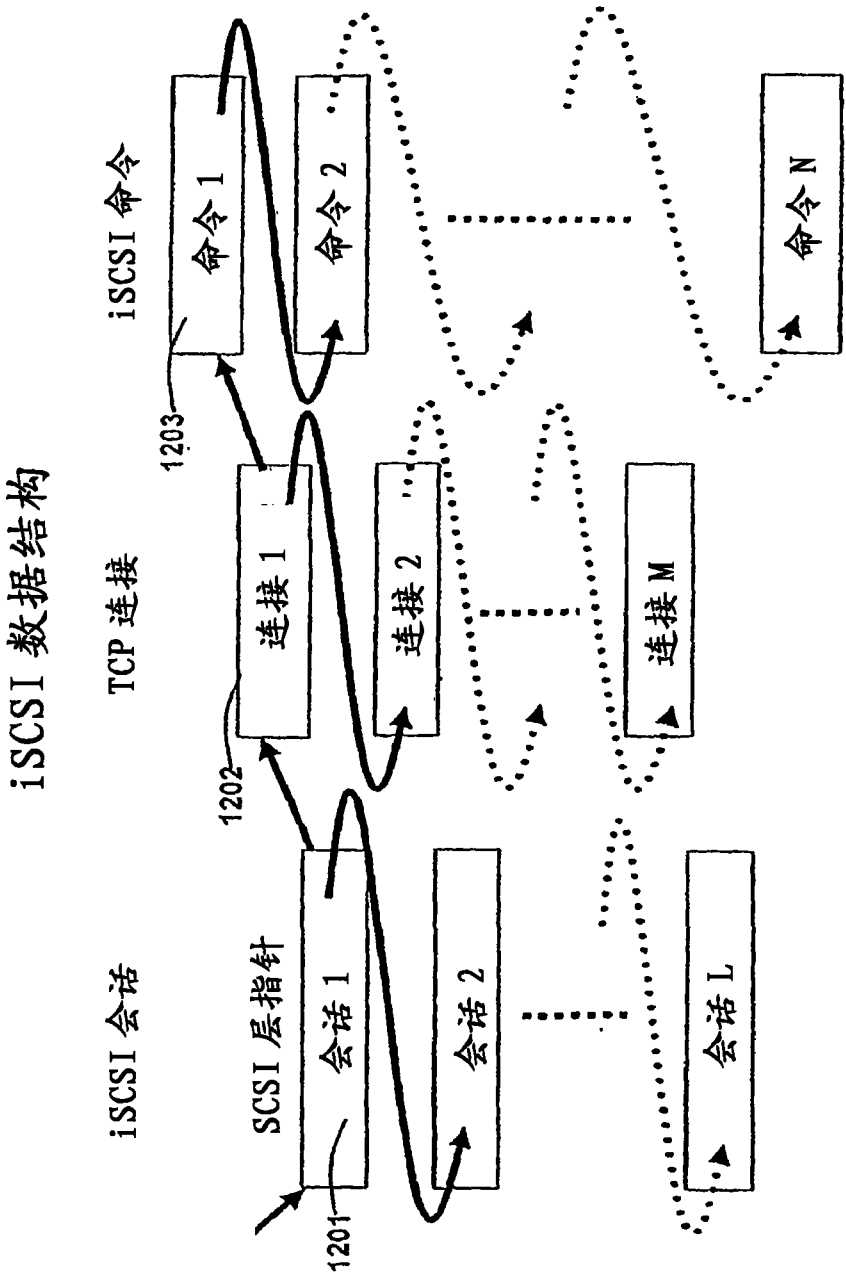


图 12

TCP/IP 会话数据库表项

源-IP	目的地-IP	源-端口	目的地-端口	协议
连接-ID	TCP-窗口-指针	窗口-大小	发送方MSS/MTU	预期-SQN
TCP-状态	TCP-发送-序号	TCP-序号	TCP-确认号	TCP-接收-序号
分段-指针	分段-发送-指针 (字节粒度)	分段-开始-时间	最后-分组-时间	TCP-参数 (慢启动、重新传输...)
分组-计数	拥塞窗口	慢启动阈值	接收方MSS	其他字段
帧-密钥	乱序指针	其他帧字段		
散列-密钥	下一匹配-指针	有效/无效		

图 13

iSCSI 会话数据库表项

iSCSI - 状态	数据 - 序号	命令 - 序号	任务 - 标记	R2T - 序号
状态 - 序号	EXP - 状态 - 序号	EXP - 命令 - 序号	目标 - 模式 (请求式或非请求式)(初次, 续发, 最大, PDU...)	目标 - 参数
ISID	TSID	门户组标记	传输方向	其他字段
连接 - ID	命令 ID	启用 RDMA	RDMA 密钥	其他 RDMA 参数
MDL - 列表 - 指针	存储器描述符	块大小	块数 (发送的或收到的)	ULP - 指针
散列 - 密钥	下一且匹配 - 指针	有效 / 无效		

图 14

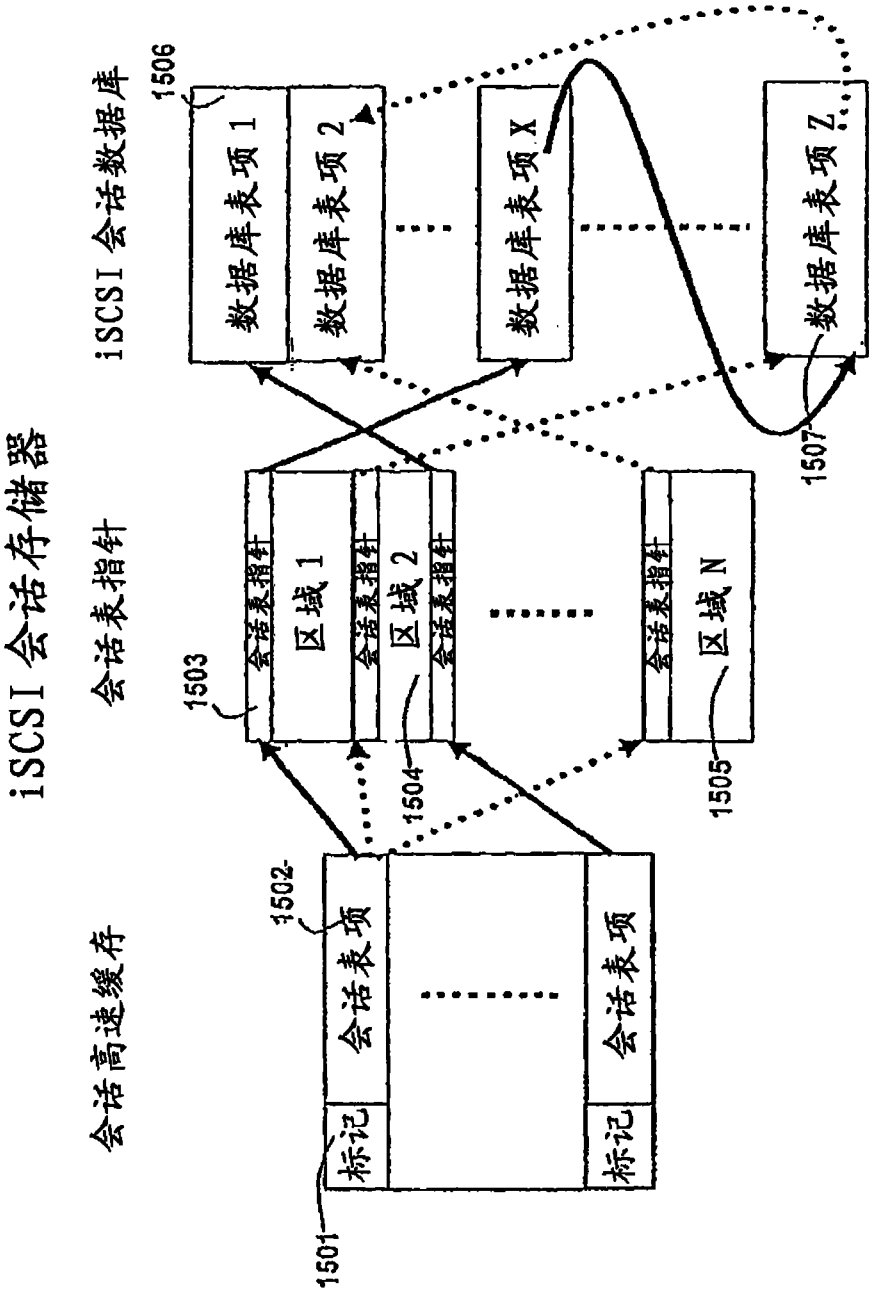


图 15

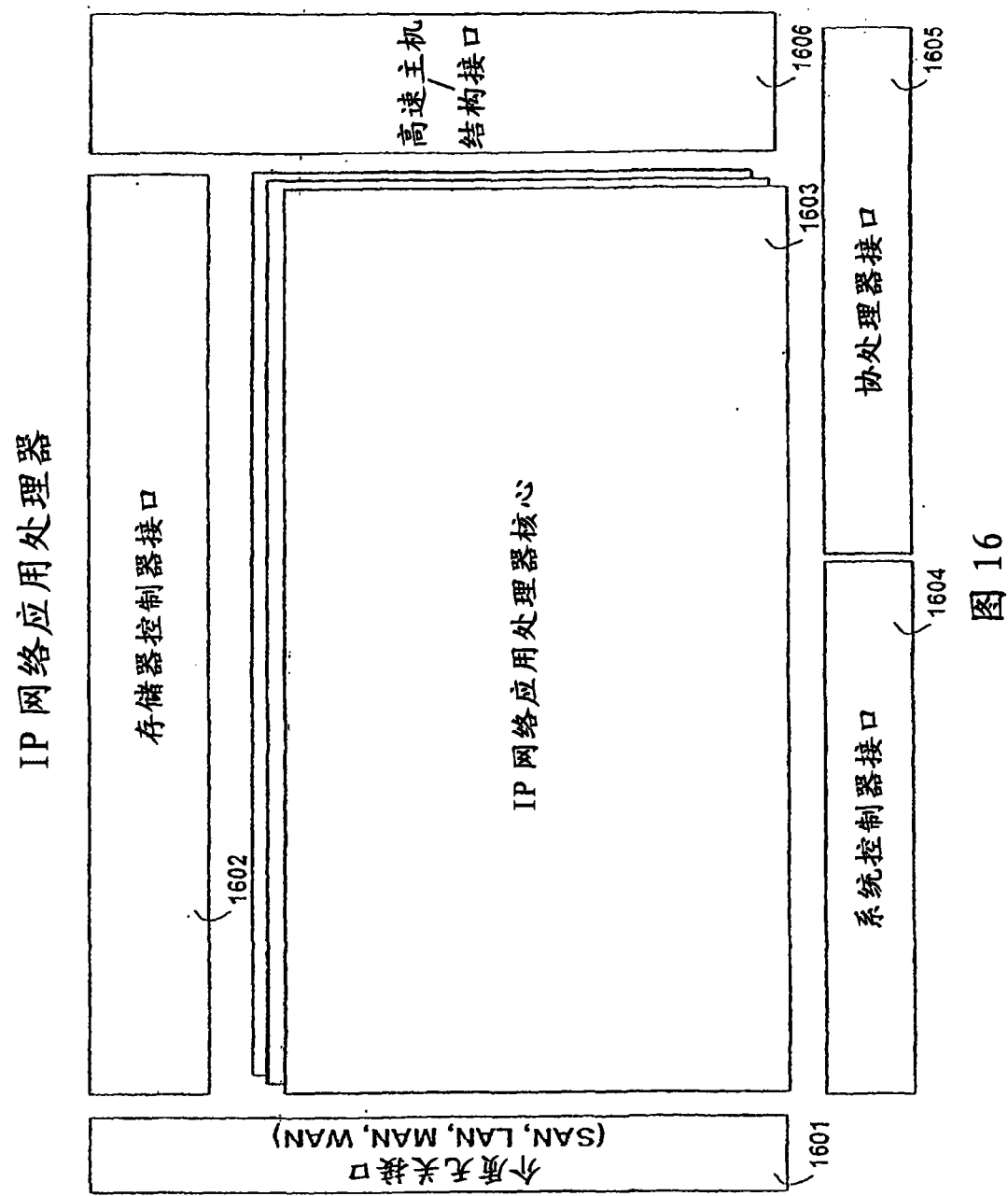


图 16

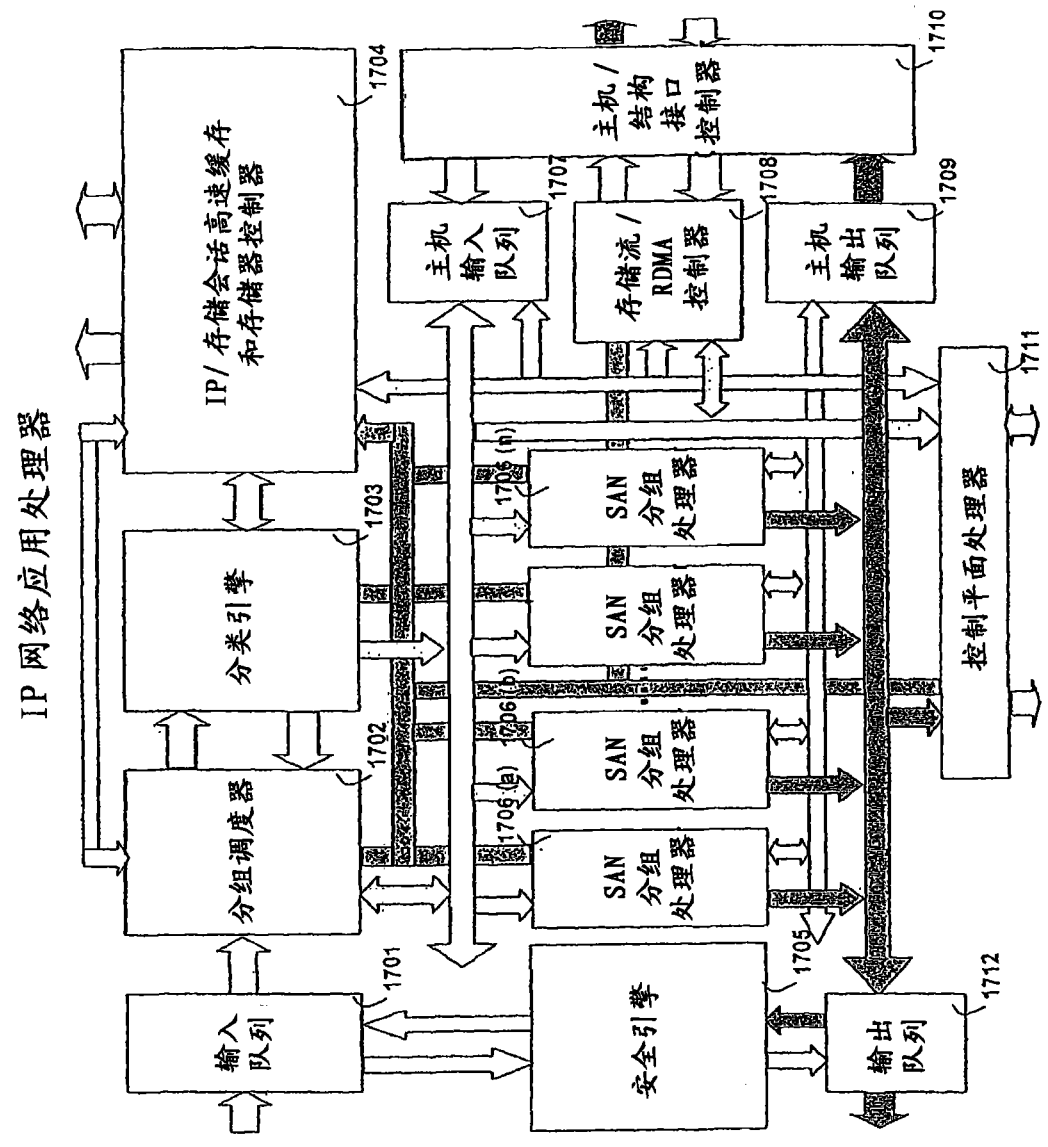
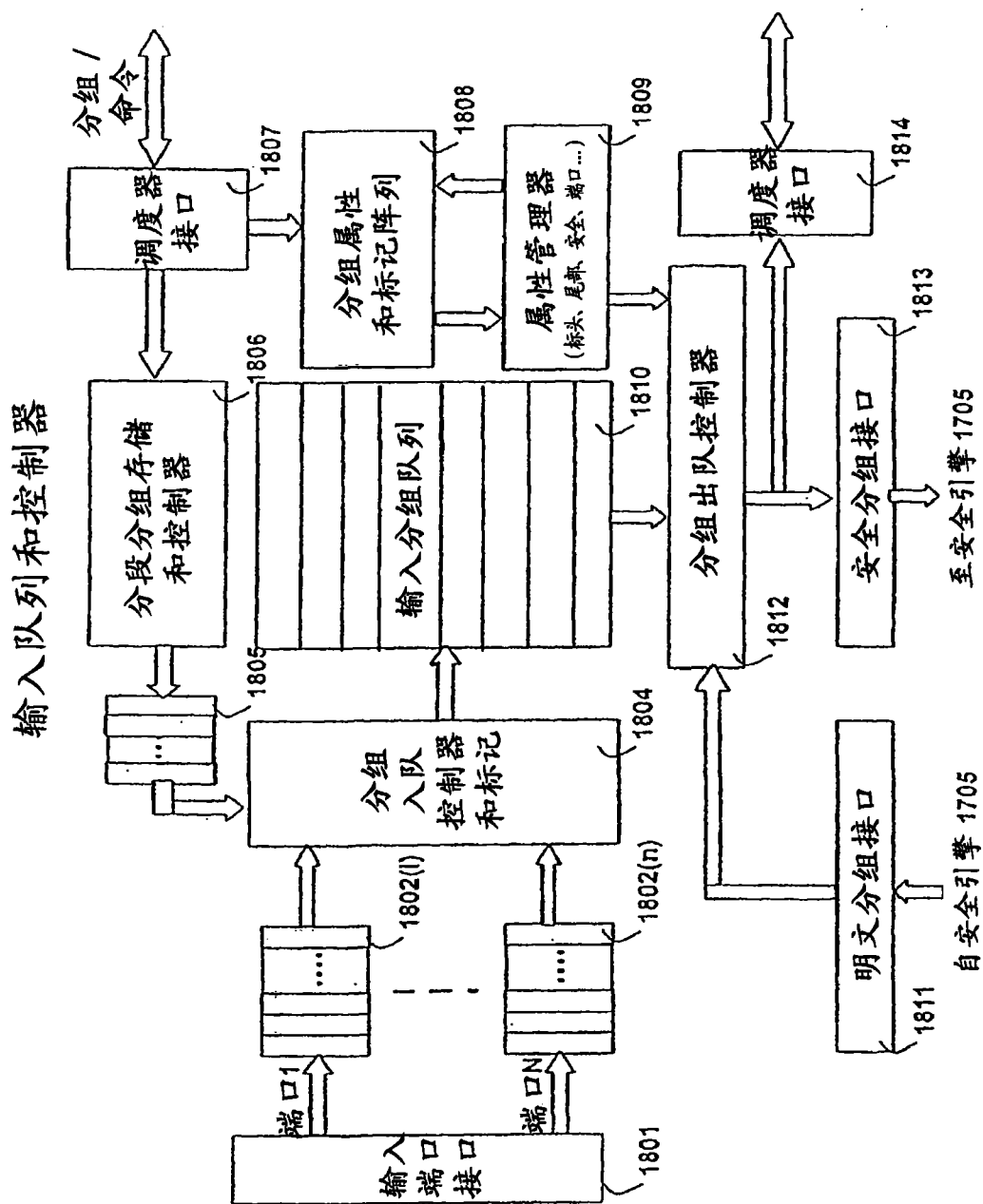
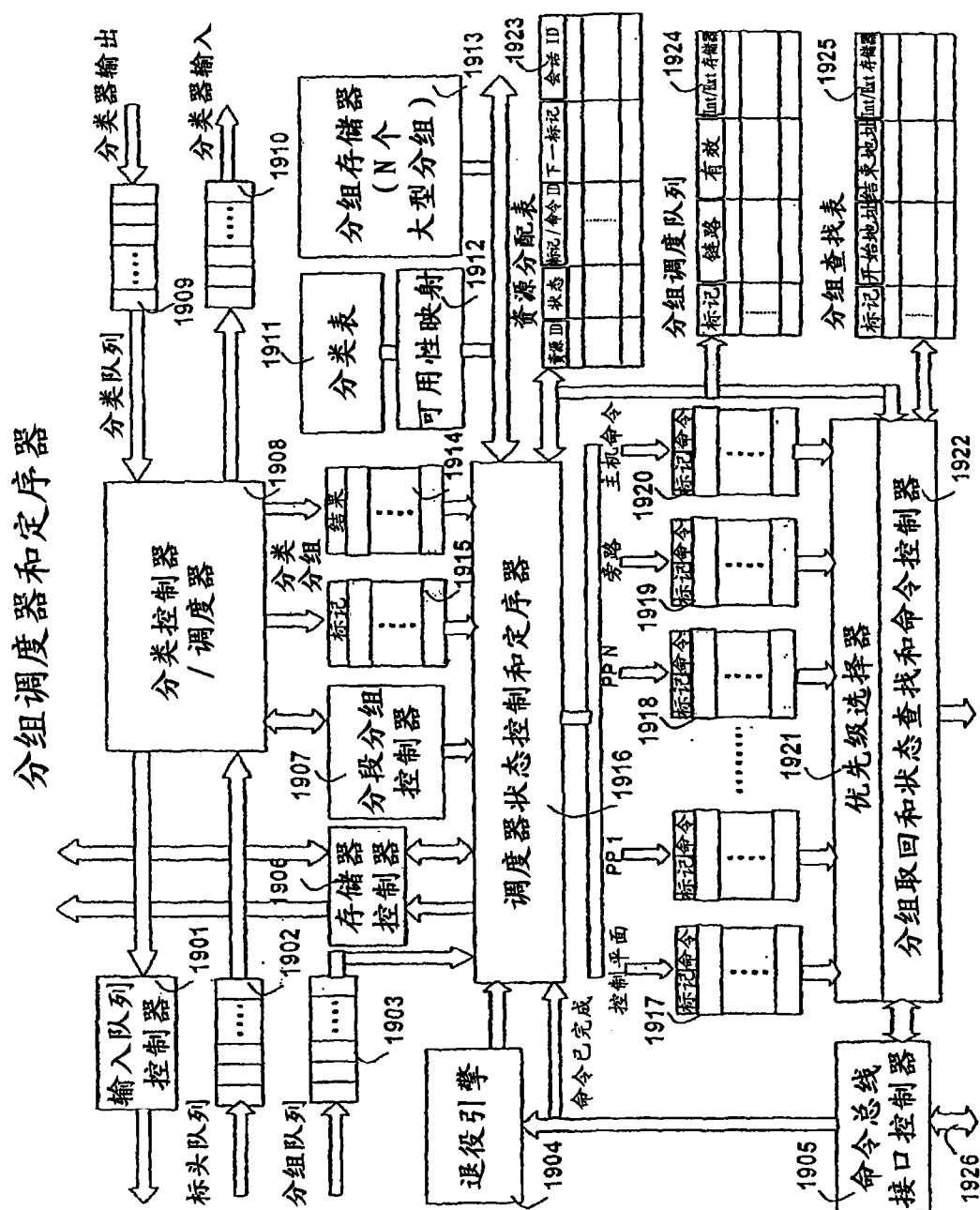


图 17



18



分组分类/策略引擎

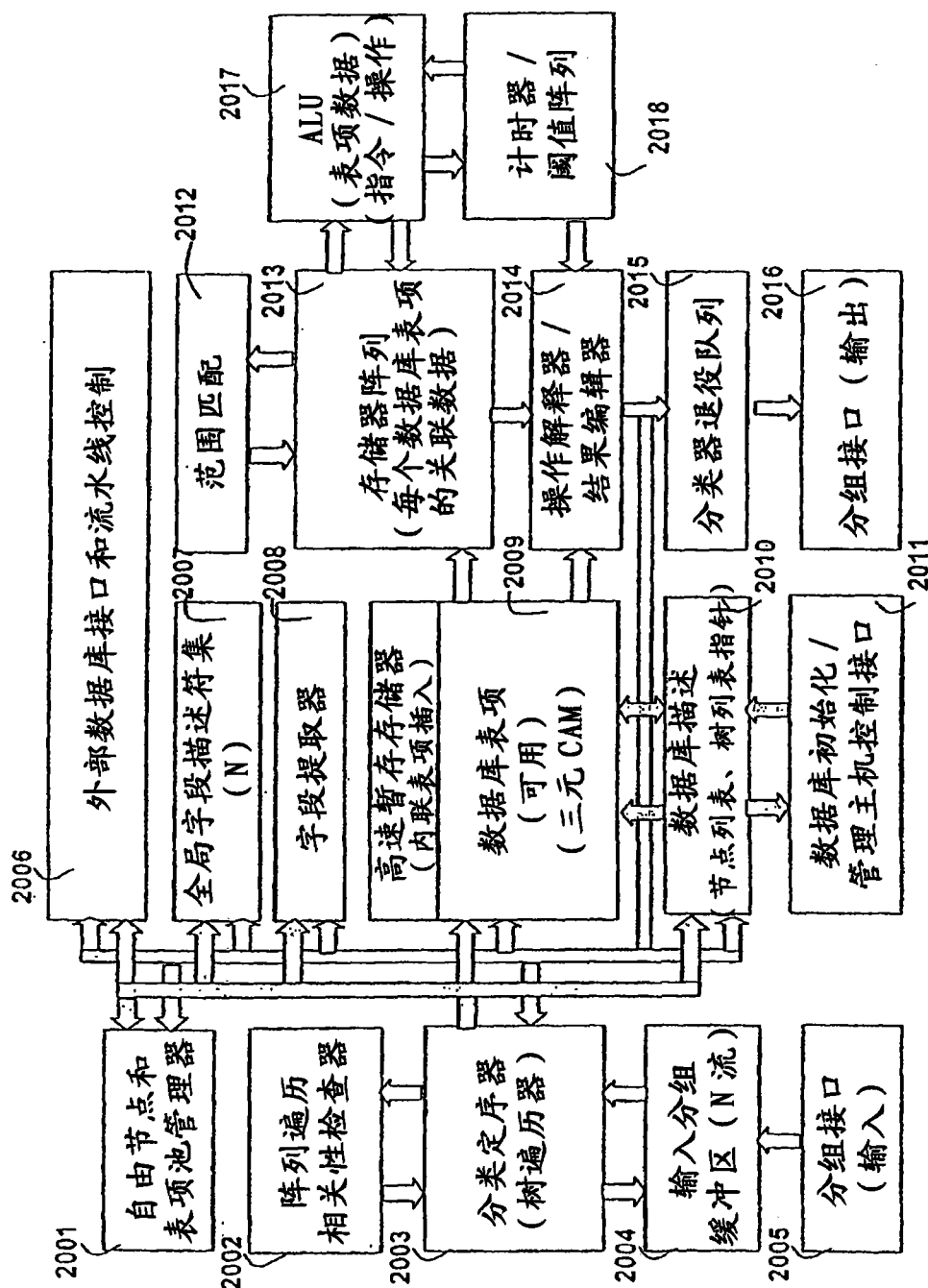


图 20

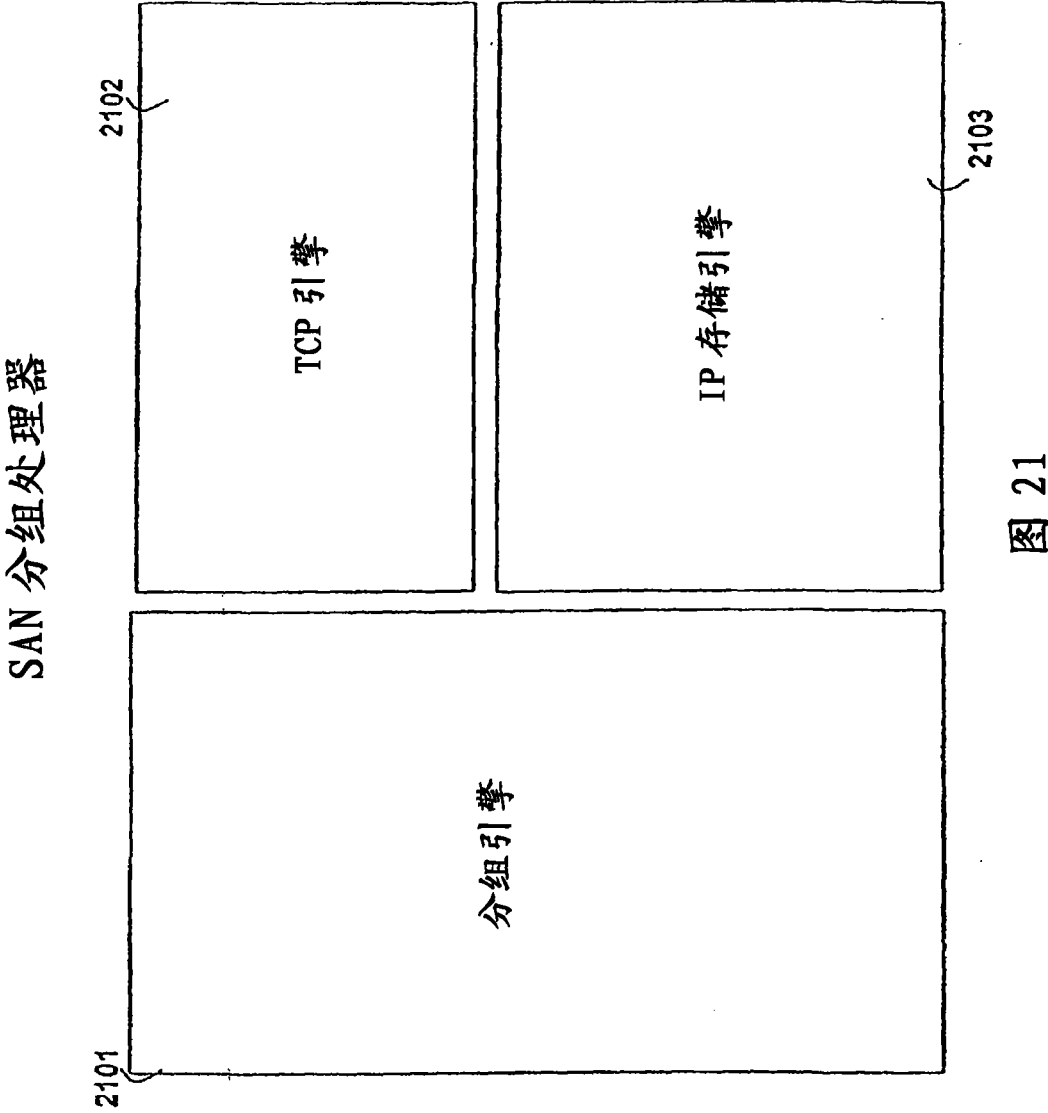


图 21

SAN 分组处理器

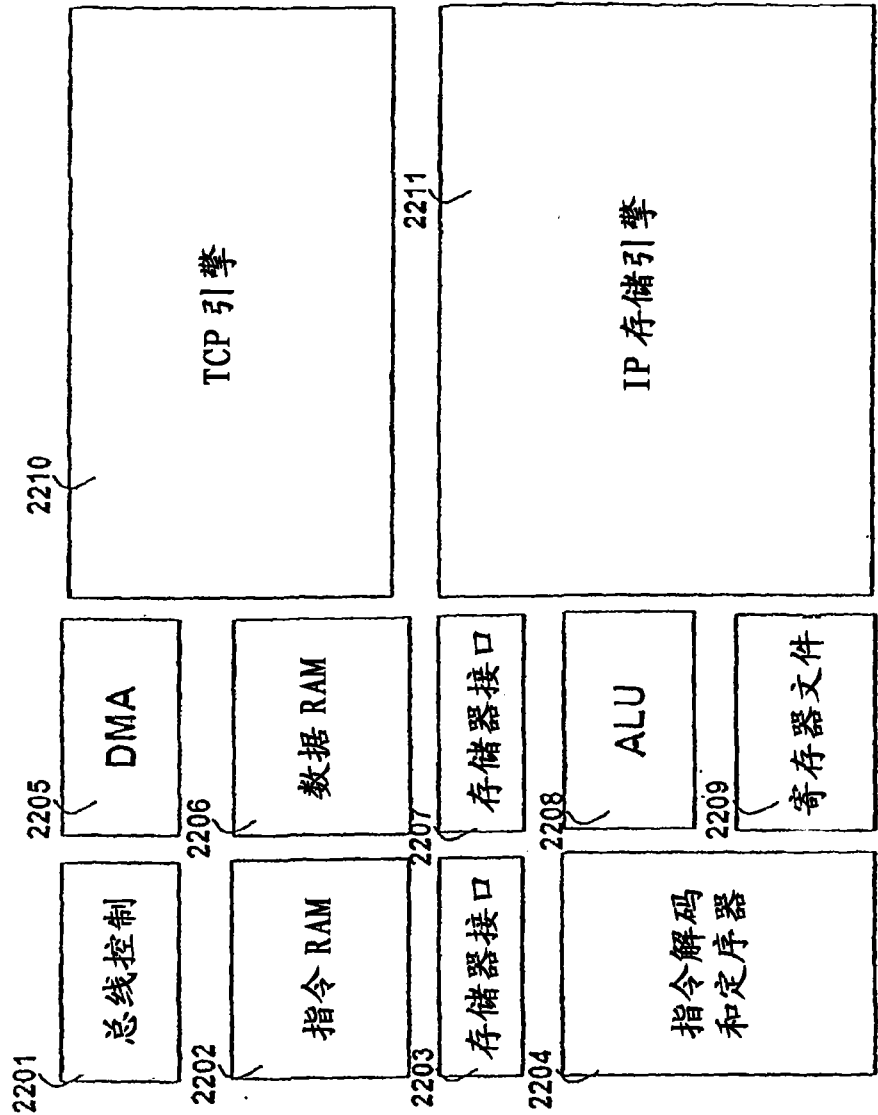


图 22

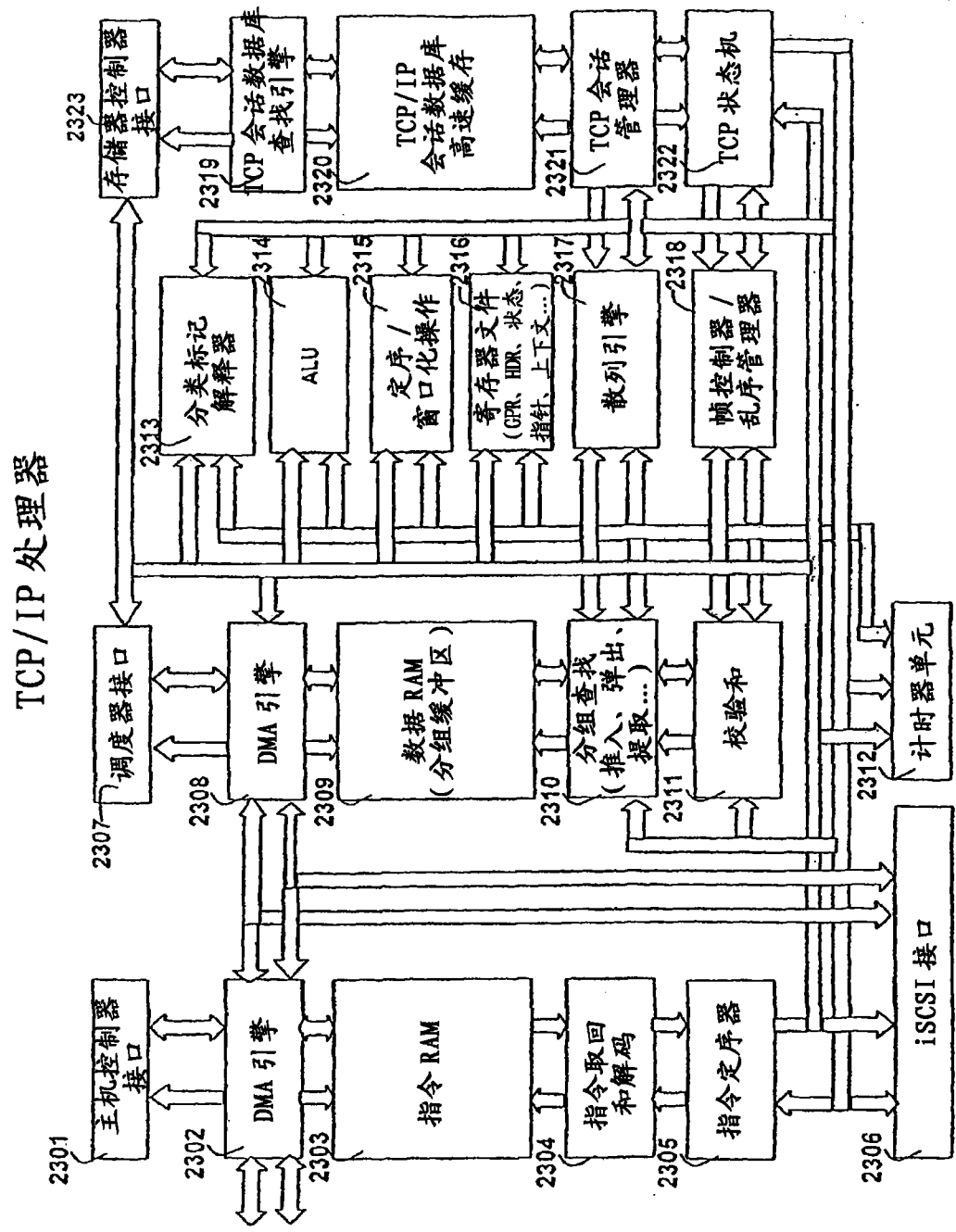


图 23

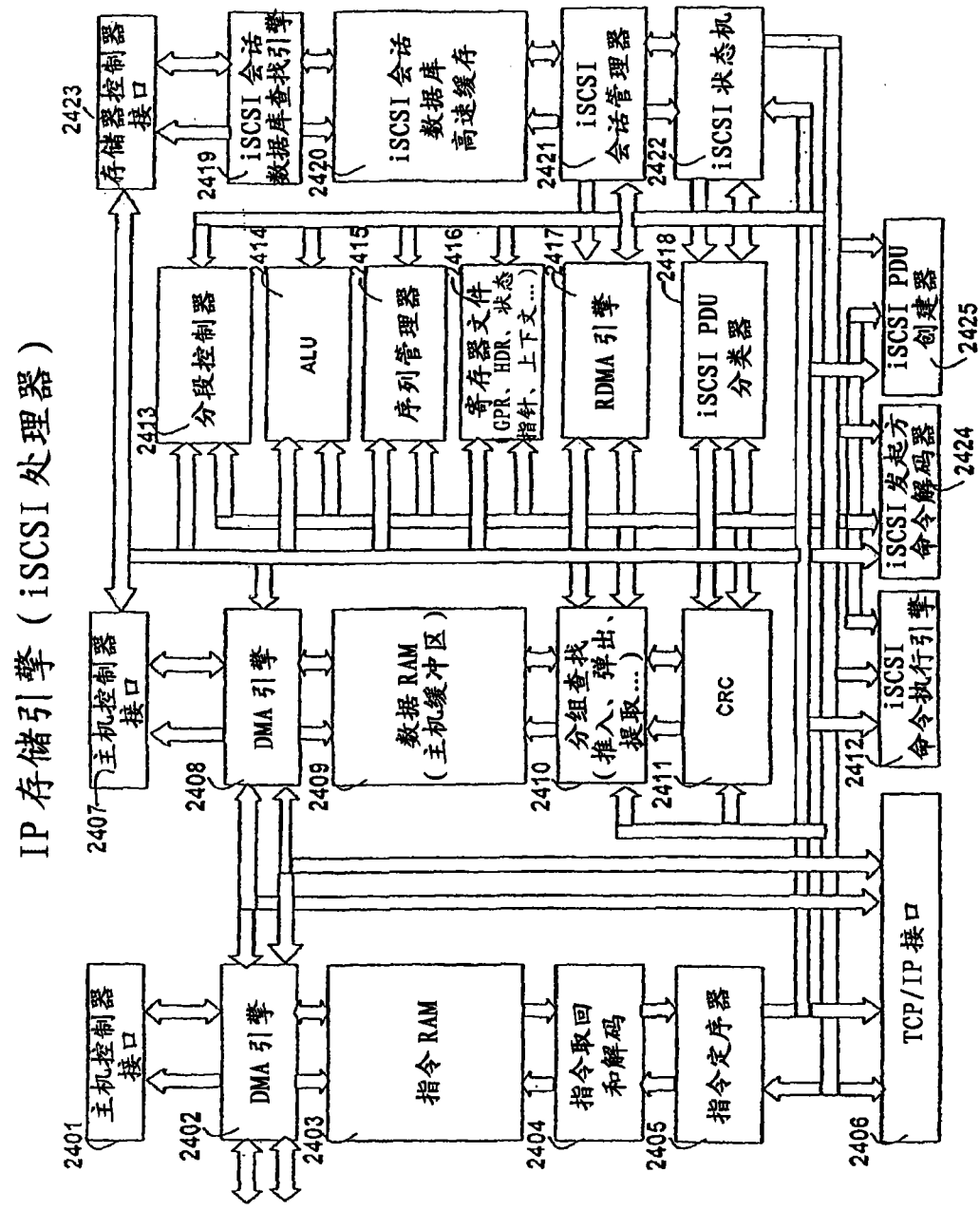


图 24

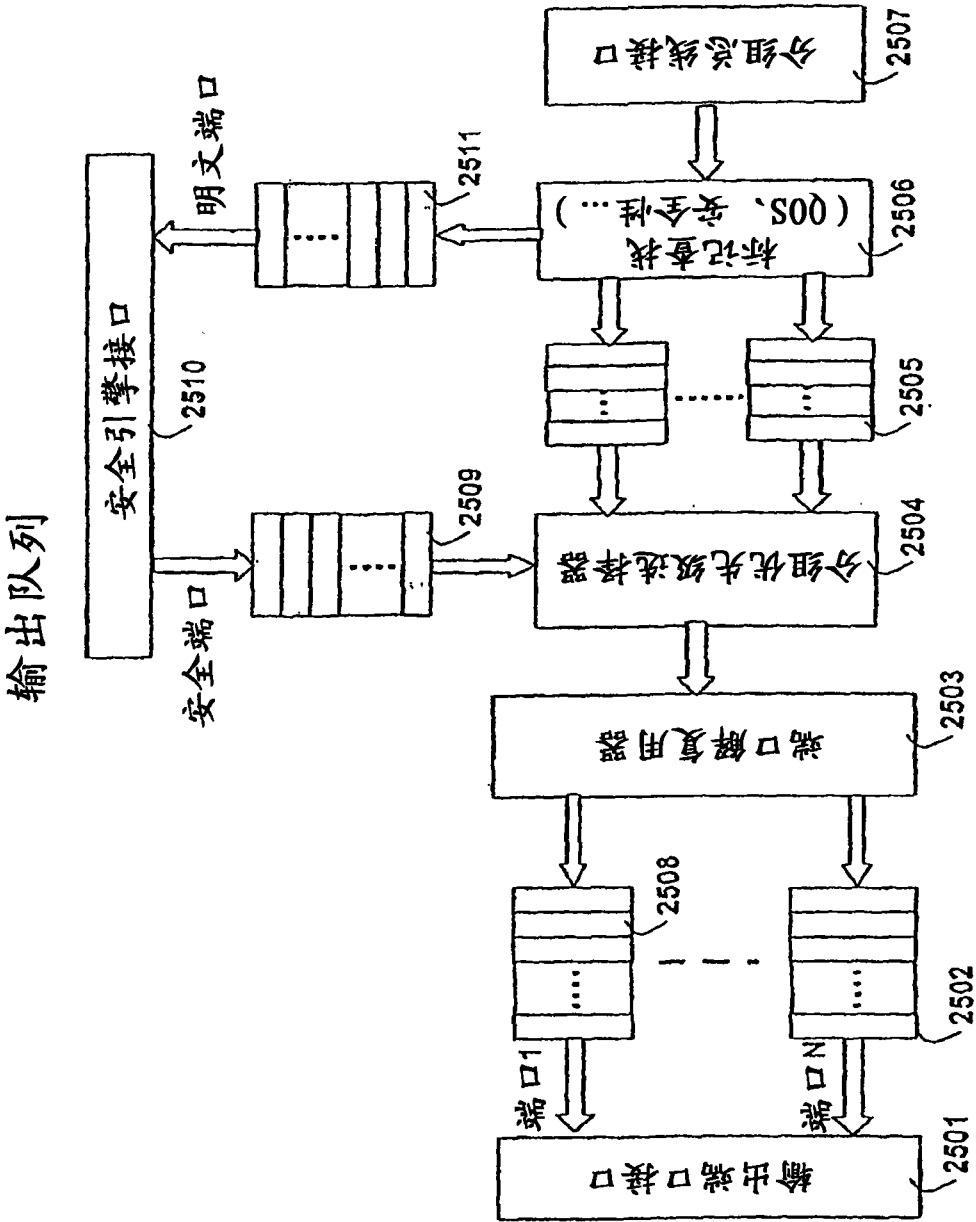


图 25

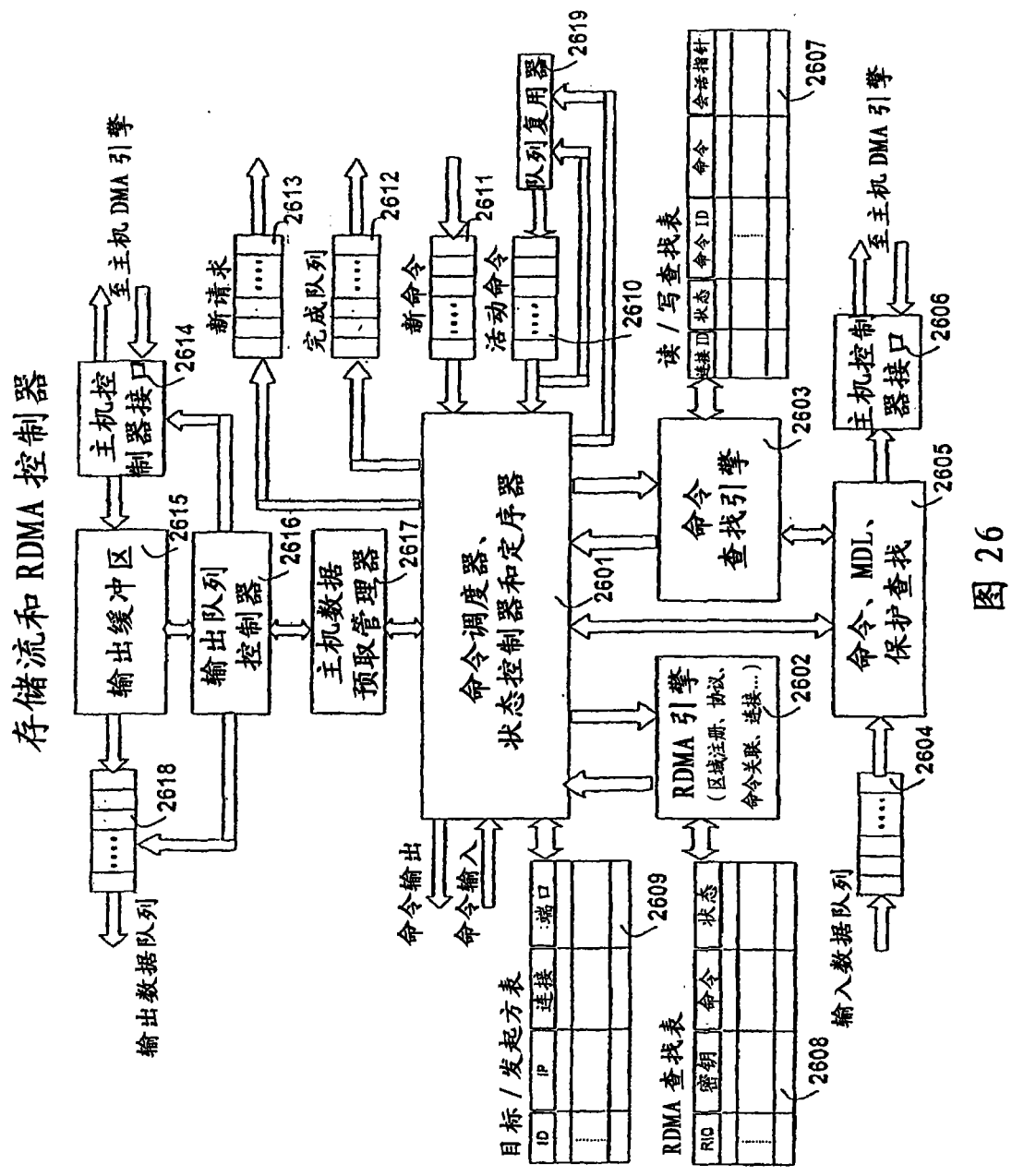


图 26

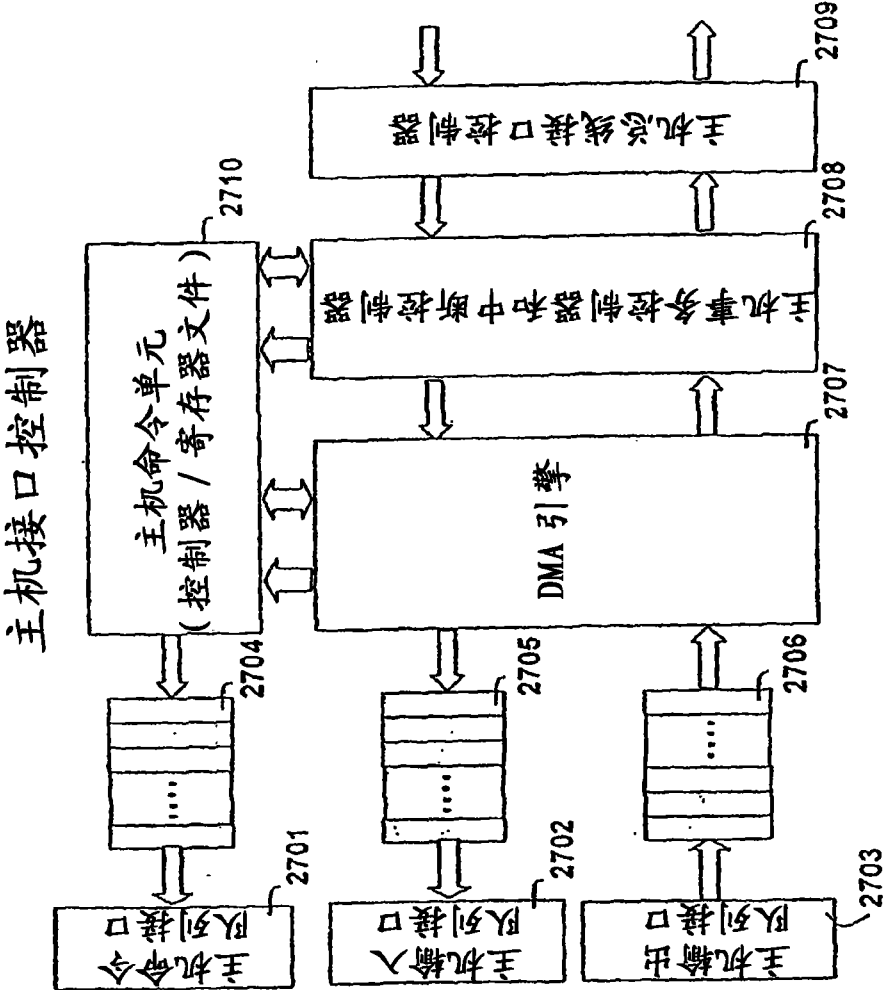


图 27

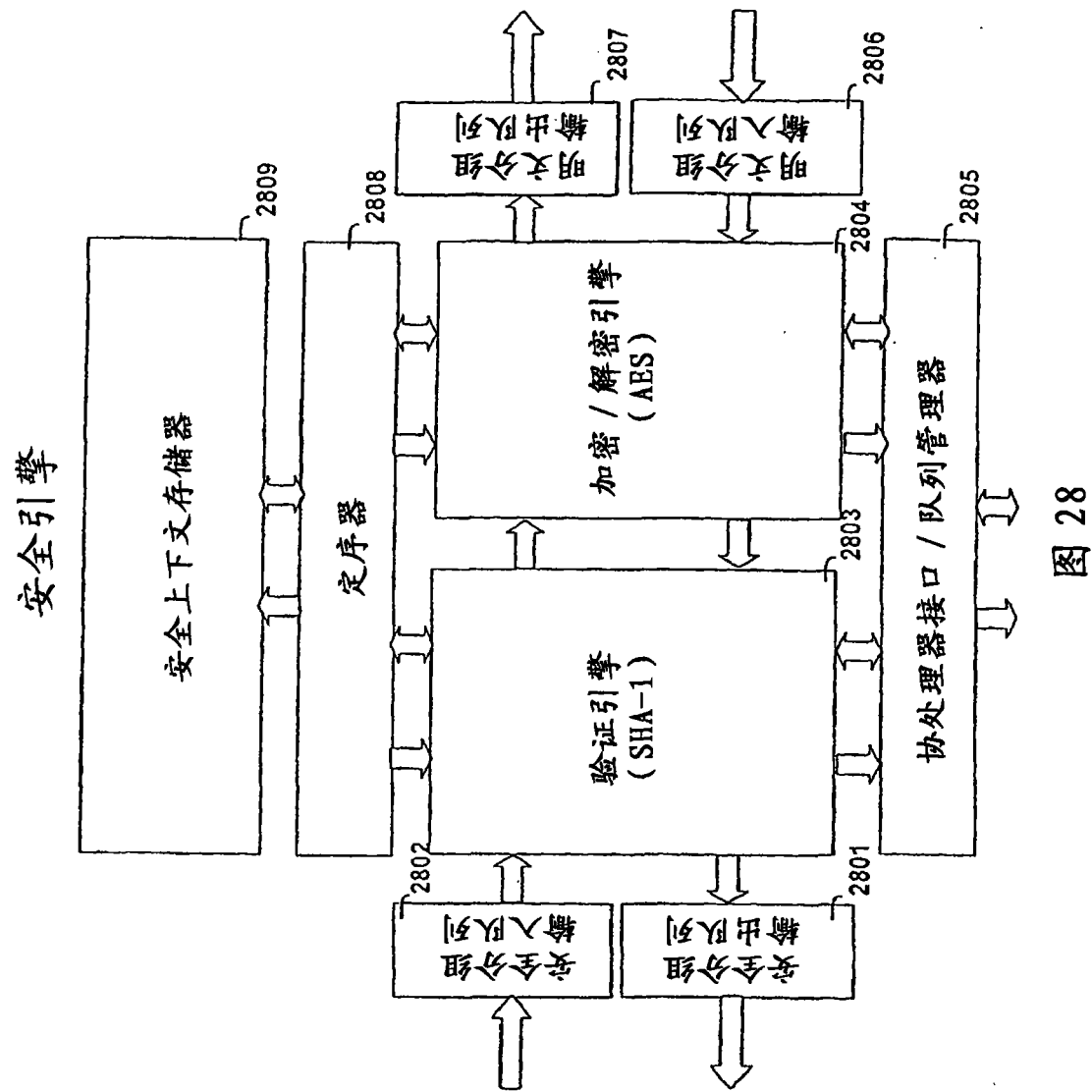


图 28

IP 存储会话高速缓存和存储器控制器复合体

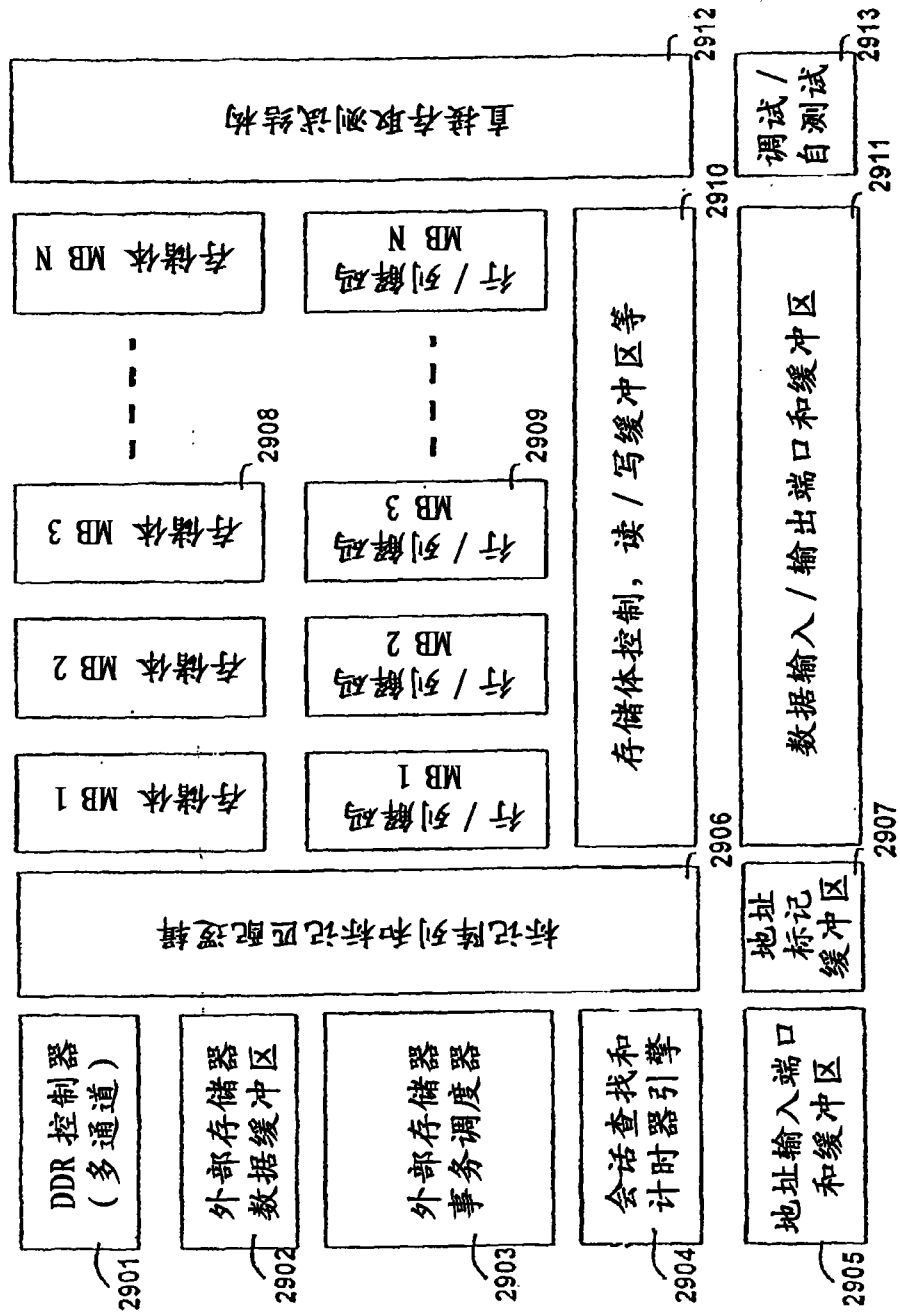


图 29

分类引擎数据结构

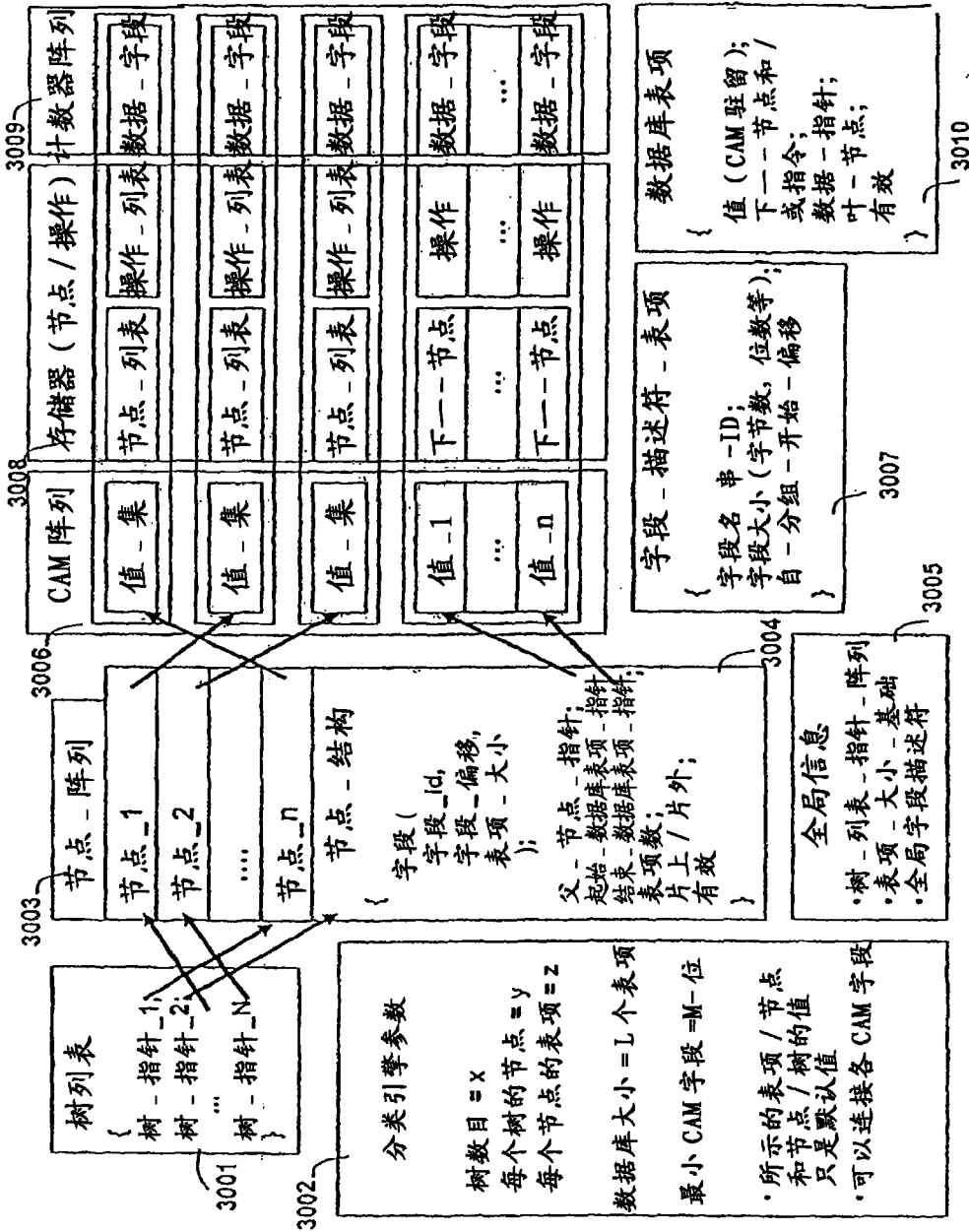


图 30

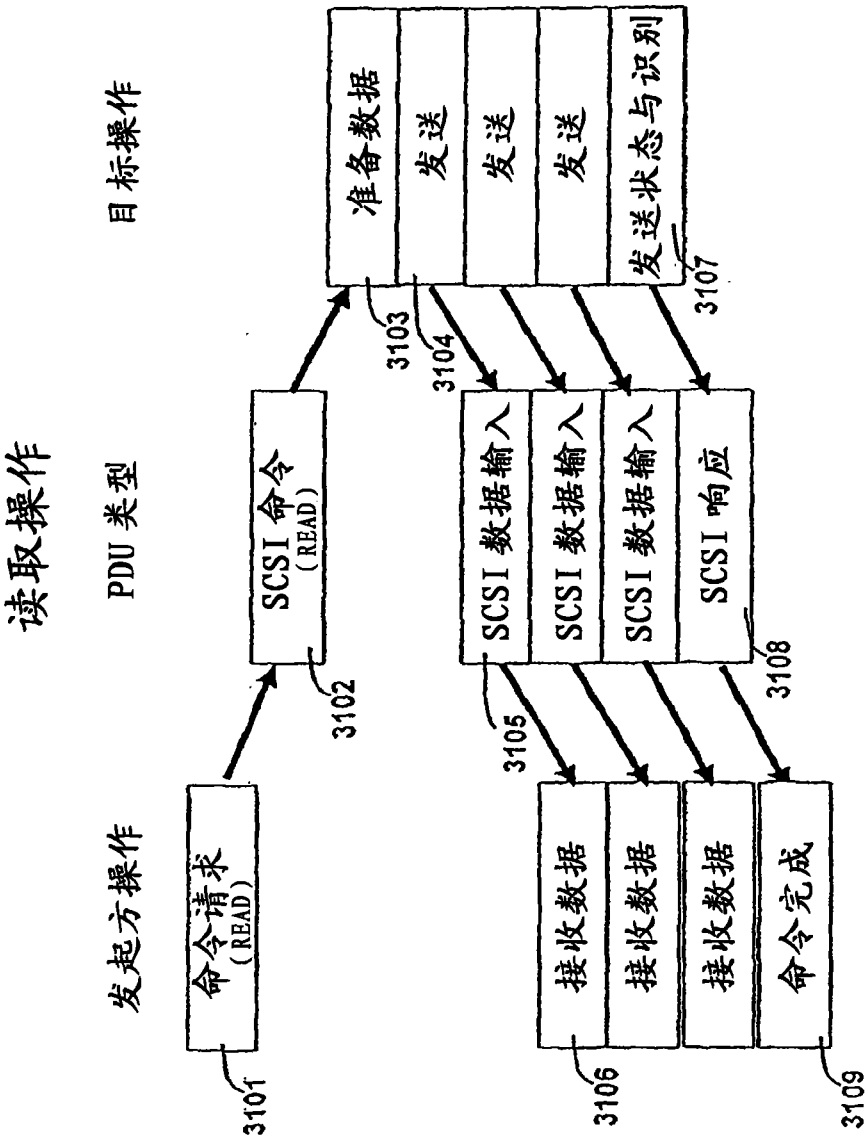


图 31

读取数据分组流

3201	接收	3202	安全	3203	分类	3204	调度	3205	执行	3206	DMA
检测	安全性关联	L3 字段	列	分组传输	基于优先级	提取分组	验证	iSCSI/TCP	将数据传输		
L2 验证	解密	L4 字段	相关性	数据项	传输			TCP 校验和	到系统存储器		
排队	排队	标记 TCP	基于流的	TCP 状态更新	资源分配			iSCSI 状态更新	SCSI 操作		
		标记 iSCSI	排队	iSCSI 操作标识				提取 MDL	完成信号		
		操作 (拒绝、忽略等)		排队 PDU				使分组退役			
		流标记									
		标记包头									
		标记有效负载									
3207		3208		3209		3210		3211		3212	

图 32

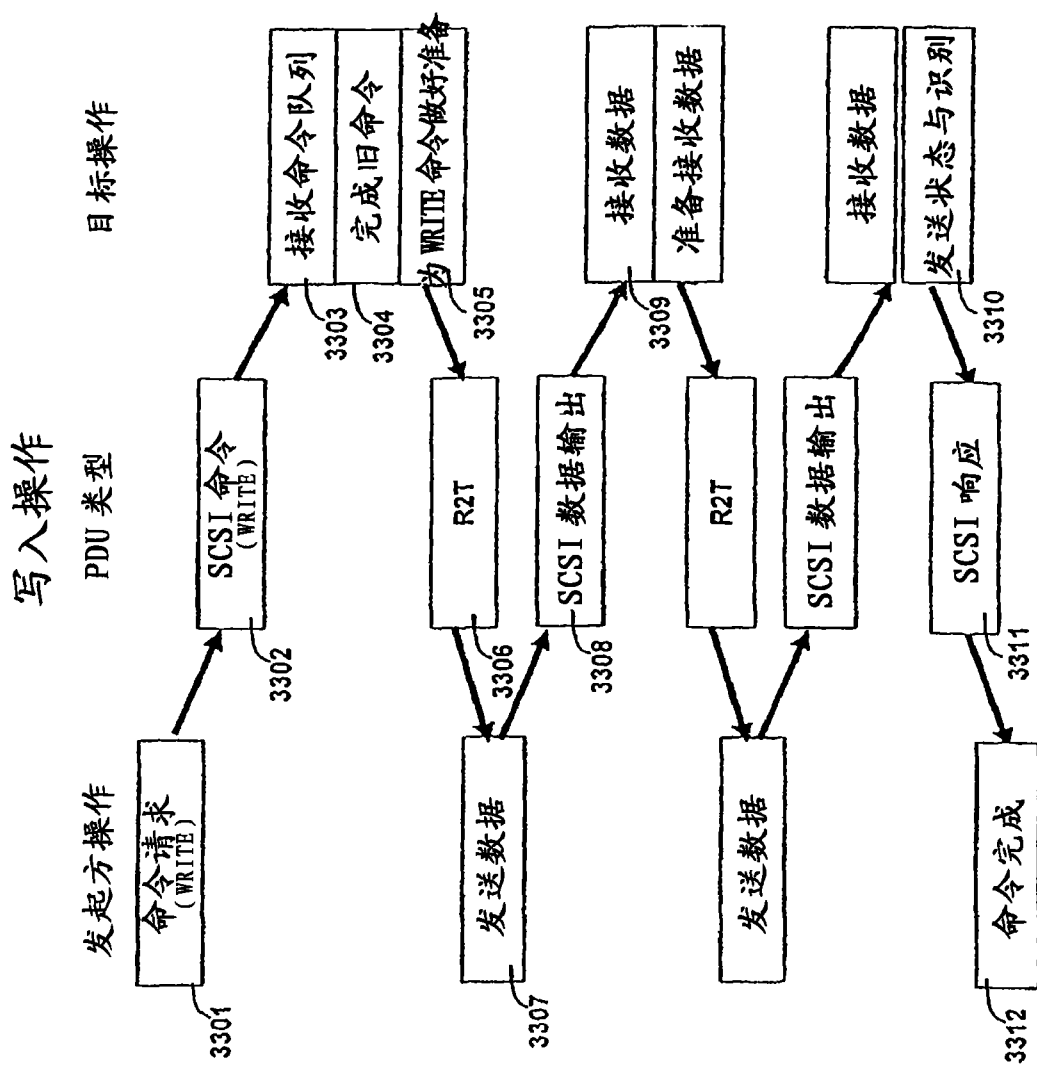


图 33

写入数据分组流

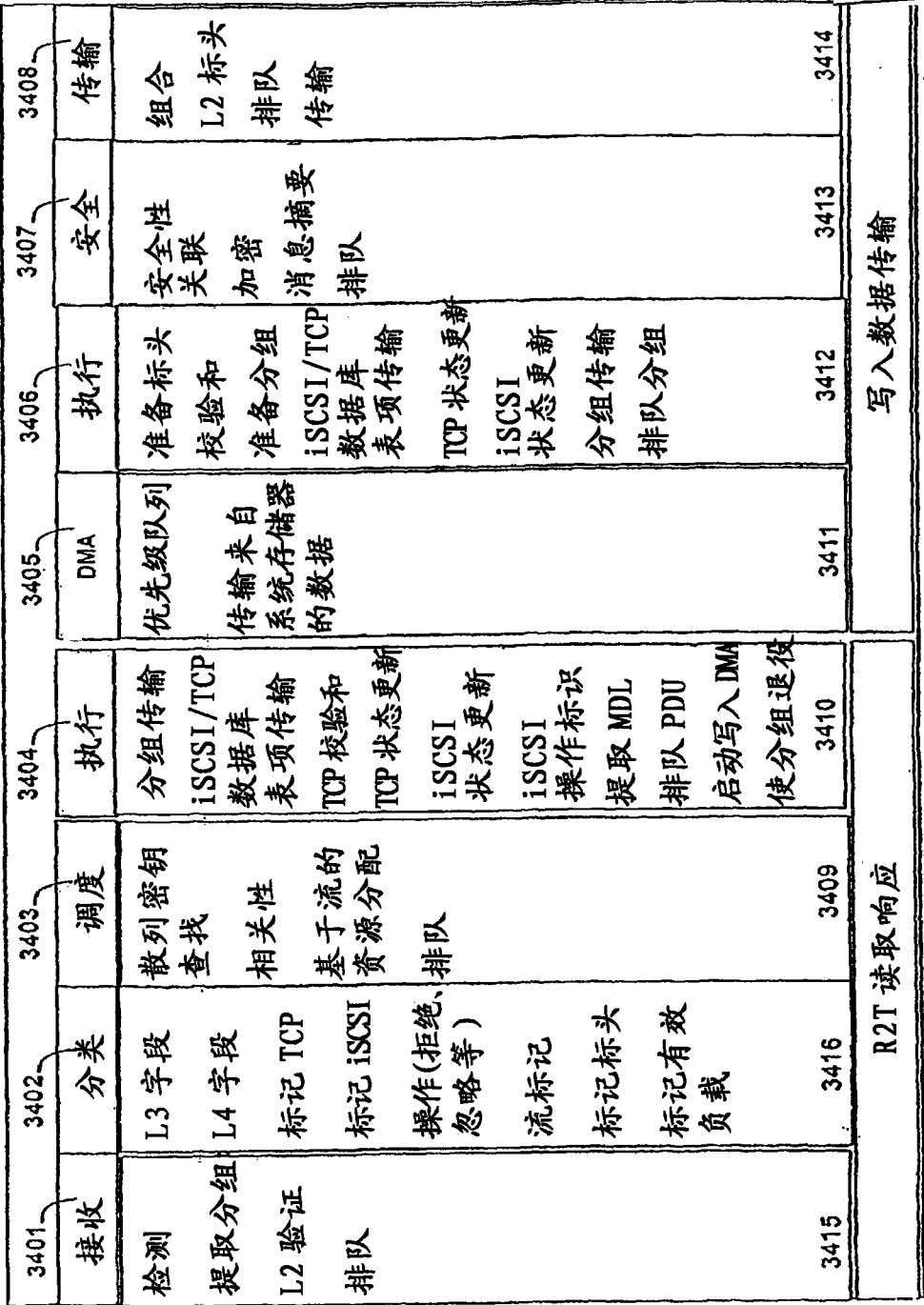


图 34

使用 RDMA 写入进行 iSCSI 读取

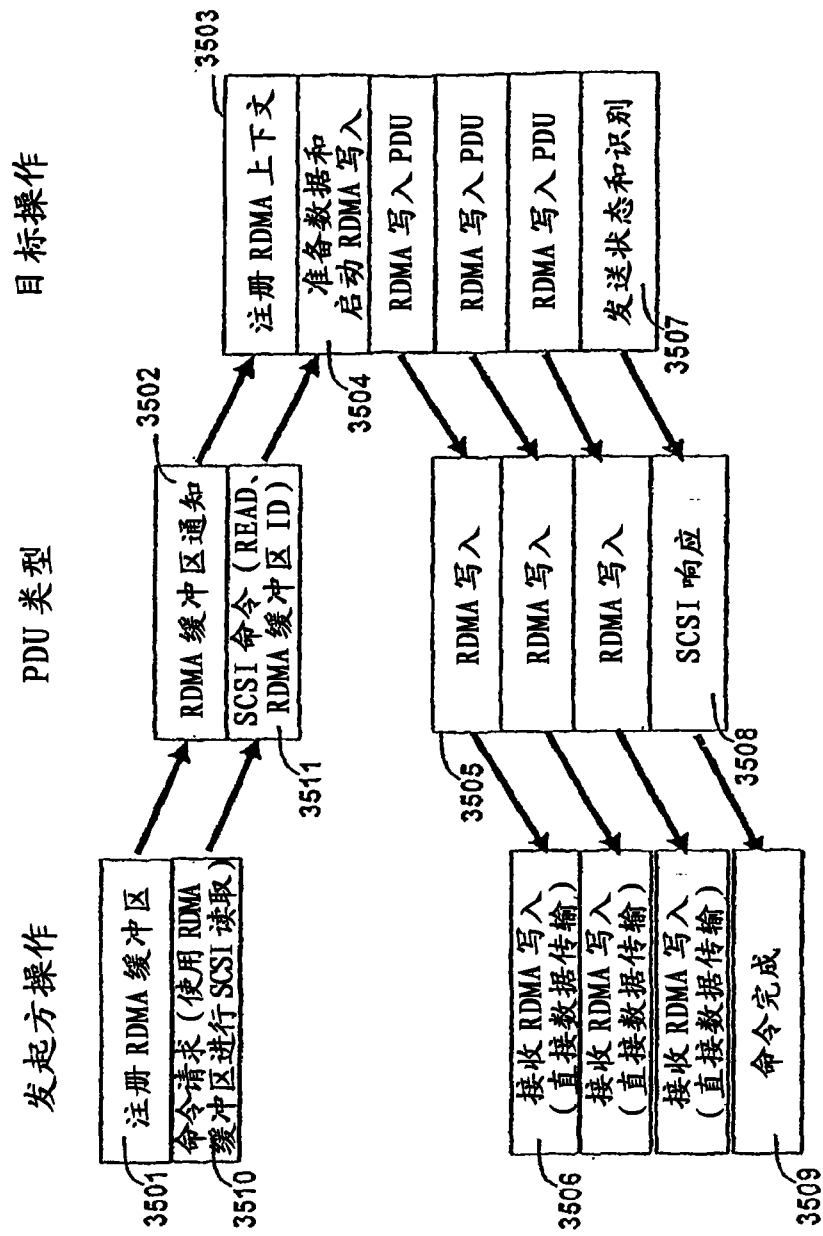


图 35

读取数据分组流 (RDMA)

3601	接收	3602	安全	3603	分类	3604	调度	3605	执行	3606	DMA
检测	安全性关联	L3 字段	散列	分组传输	基于优先级的队列						
提取分组	验证	L4 字段	相关性	iSCSI/TCP 数据库	将数据传输到系统存储器						
L2 验证	解密	标记 TCP	基于流的资源分配	TCP 状态更新	RDMA 操作						
排队	排队	标记 iSCSI 操作 (拒绝、忽略等)	排队	TCP 校验和	完成信号						
		流标记		iSCSI /RDMA 状态更新	iSCSI 操作完成信号						
		标记包头		RDMA 操作识别	(如果一个 RDMA)						
		标记有效负载		提取 MDL							
				排队 PDU							
				使分组退役							
3607		3608		3609		3610		3611		3612	

图 36

使用 RDMA 读取进行 iSCSI 写入

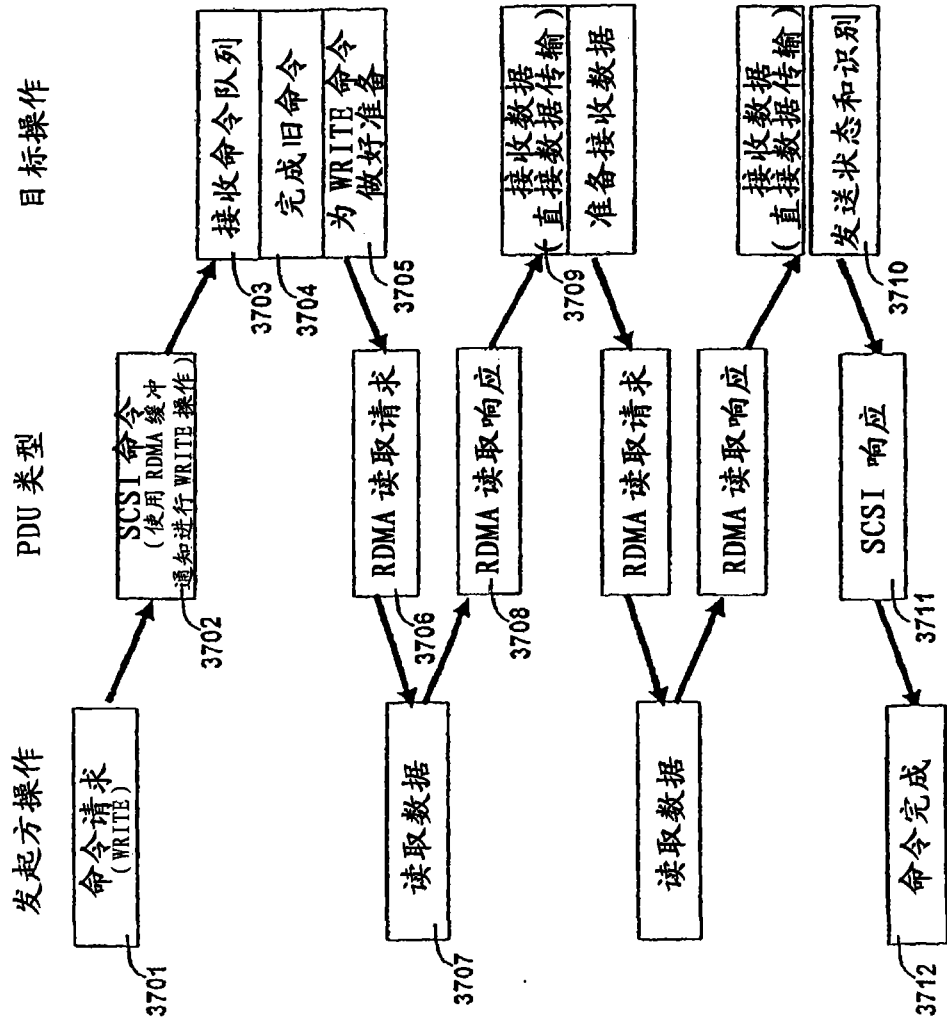


图 37

写入数据分组流 (RDMA)

3801		3802		3803		3804		3805		3806		3807		3808	
接收		分类		调度		执行		DMA		执行		安全		传输	
检测 提取分组 L2 验证 排队	L3 字段		散列密钥 查找 相关性 基于流的 资源分配 排队	分组传输 iSCSI/TCP 数据库 表项传输 TCP 校验和 TCP 状态更新 iSCSI /RDMA 状态更新 RDMA 操作识别 提取 RID 存取检查 启动写入 RDMA 使分组退役	优先级队列 从系统 存储器 传输数据	准备标头 校验和 准备分组 iSCSI/TCP 数据库 表项传输 TCP 状态更新 iSCSI/RDMA 状态更新 分组传输 排队分组	安全性关联 加密 消息摘要 排队	组合 L2 标头 排队 传输							
	L4 字段														
	标记 TCP														
	标记 iSCSI														
	操作 (拒绝、 忽略等)														
流标记															
标记标头															
标记有效负载															
3815	3816	3809	3810	3811	3812	3813	3814								
RDMA 读取请求				RDMA 读取响应分组 (写入)											

图 38

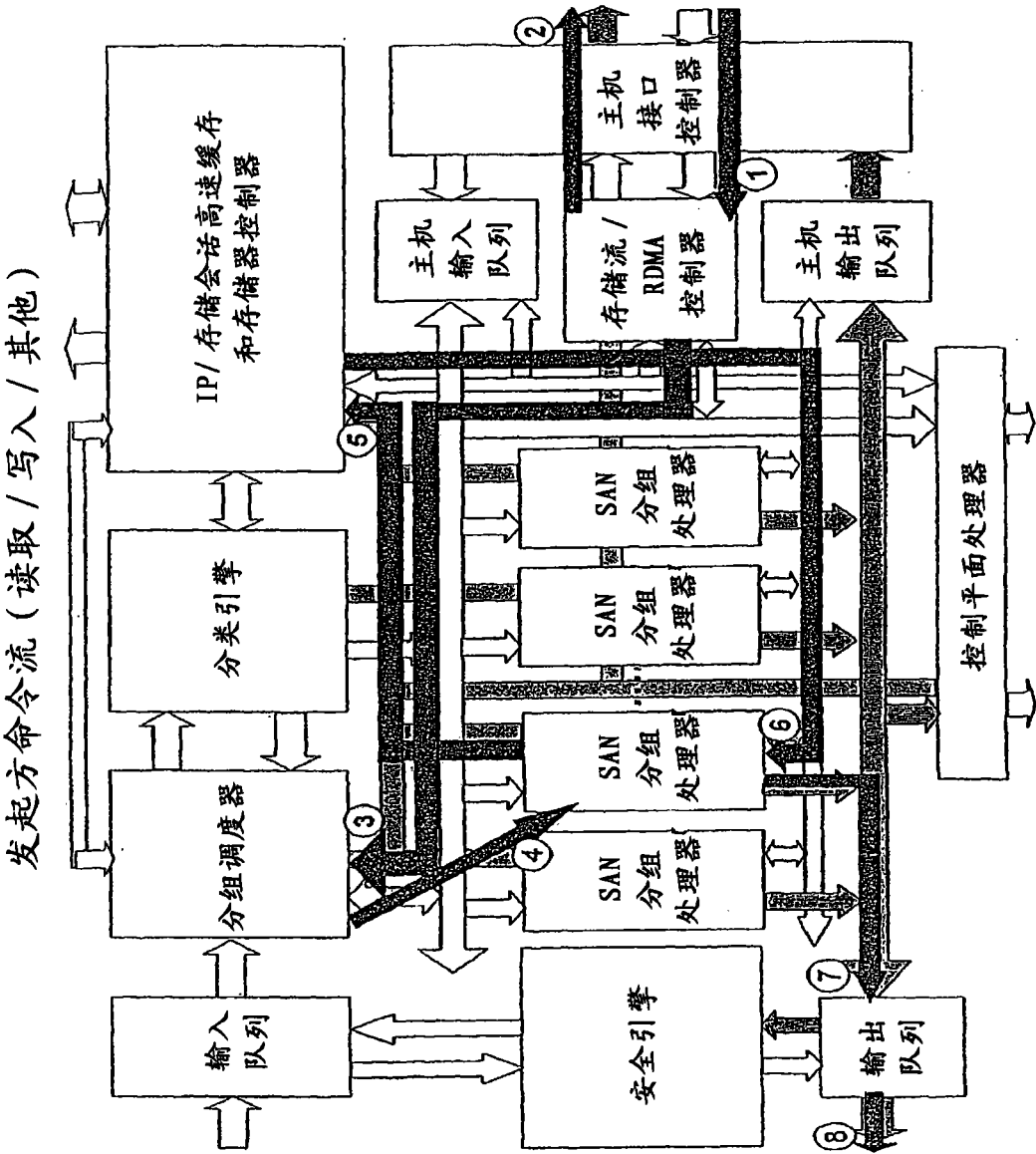


图 39

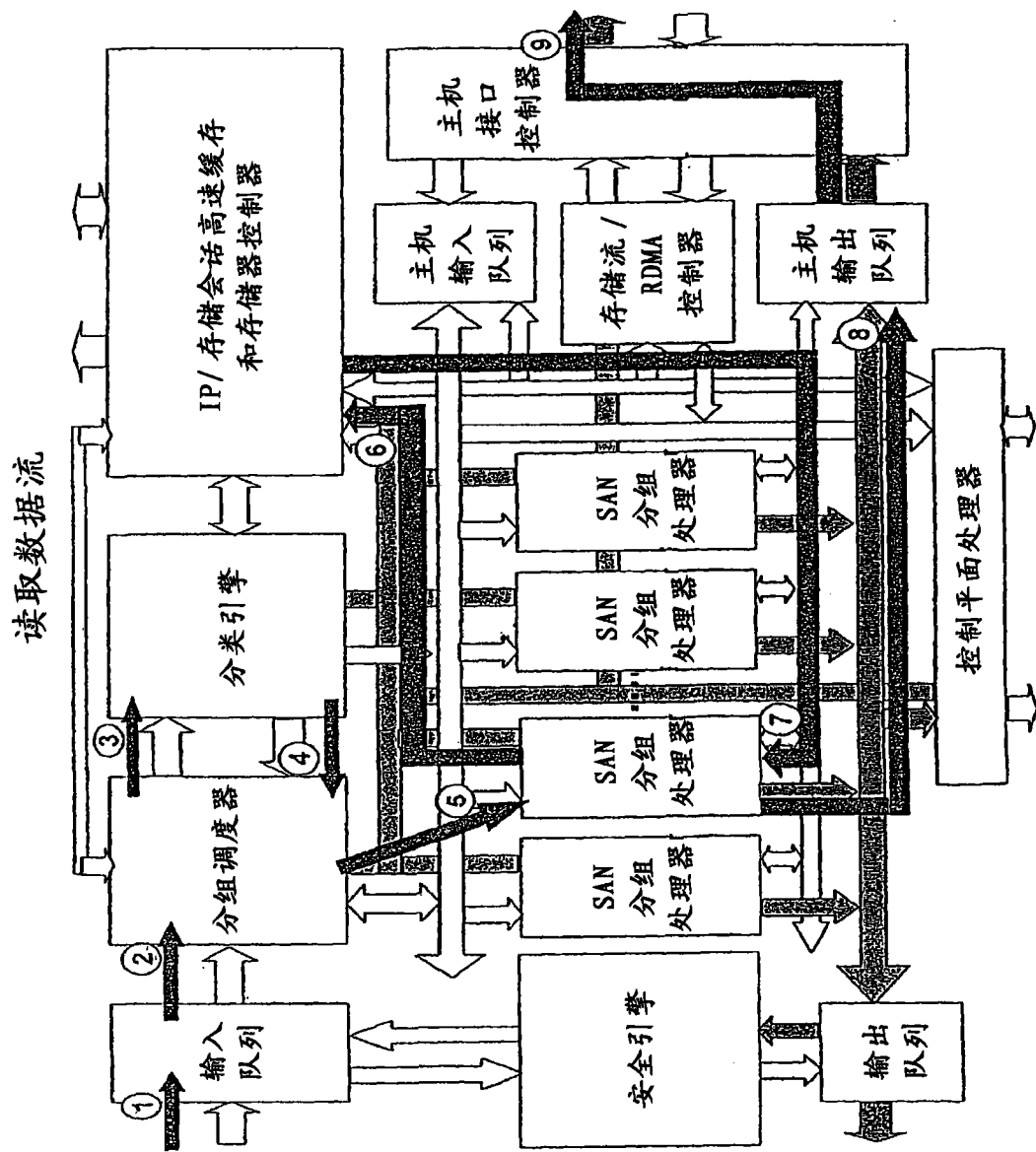
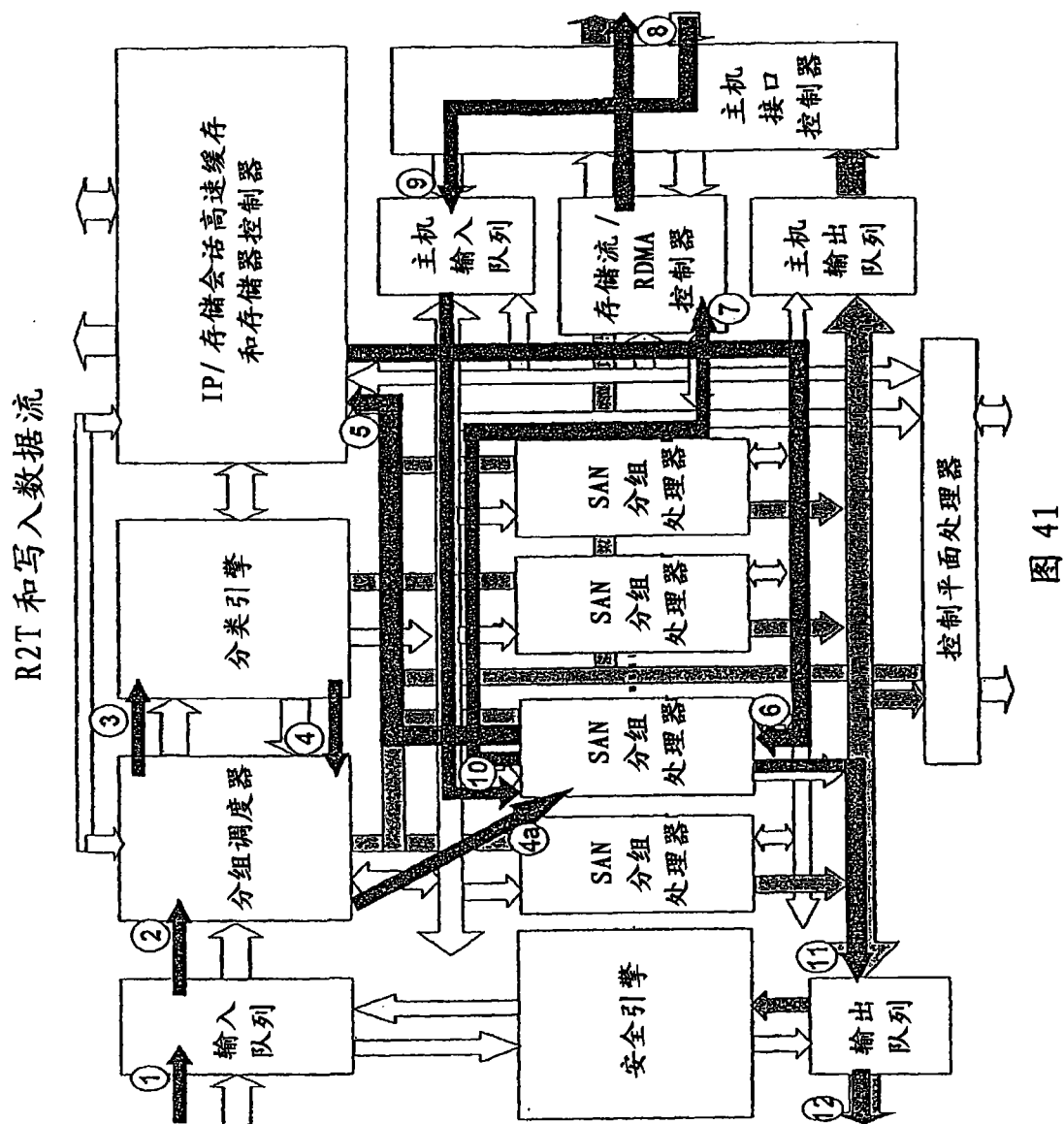


图 40



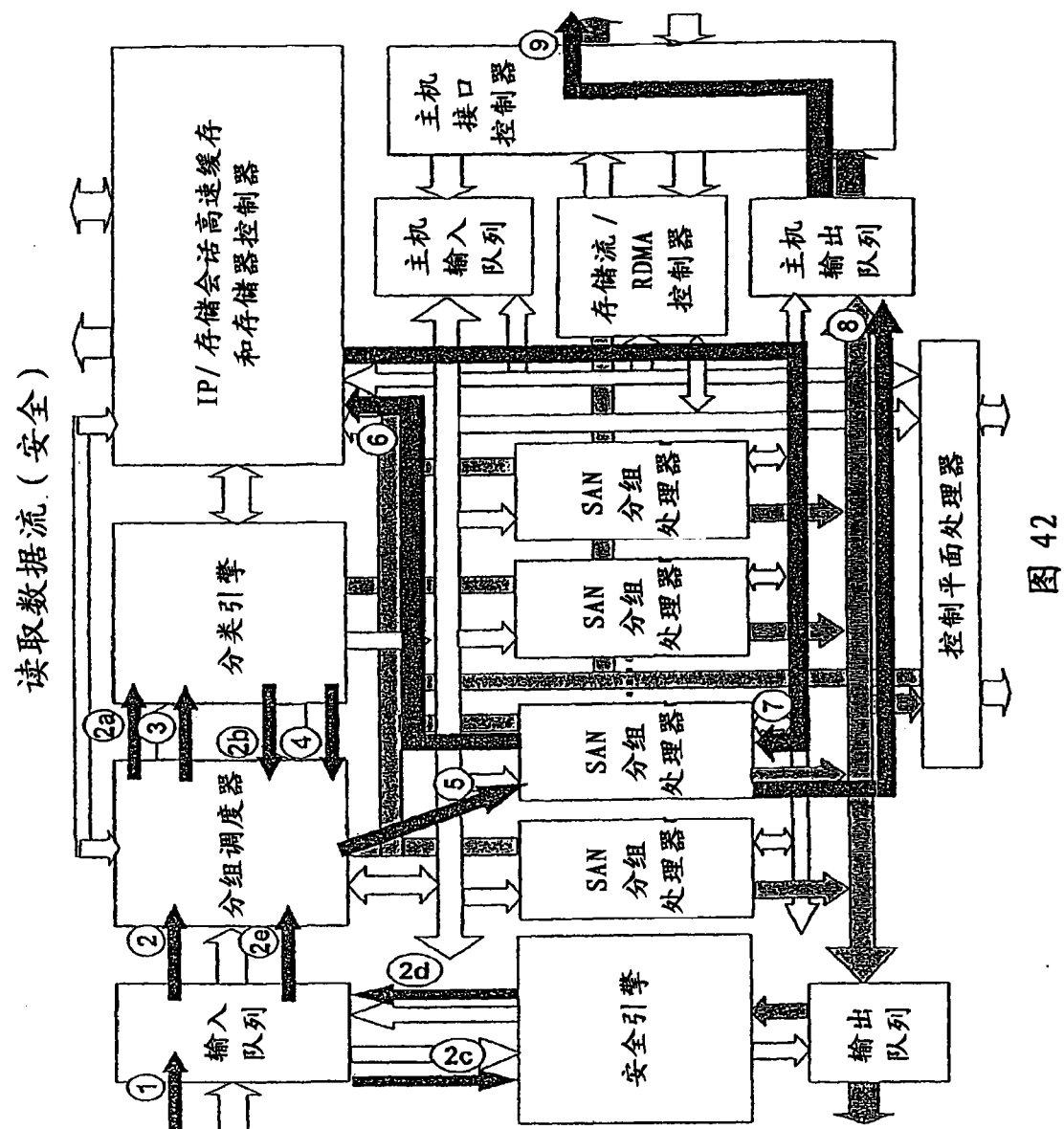
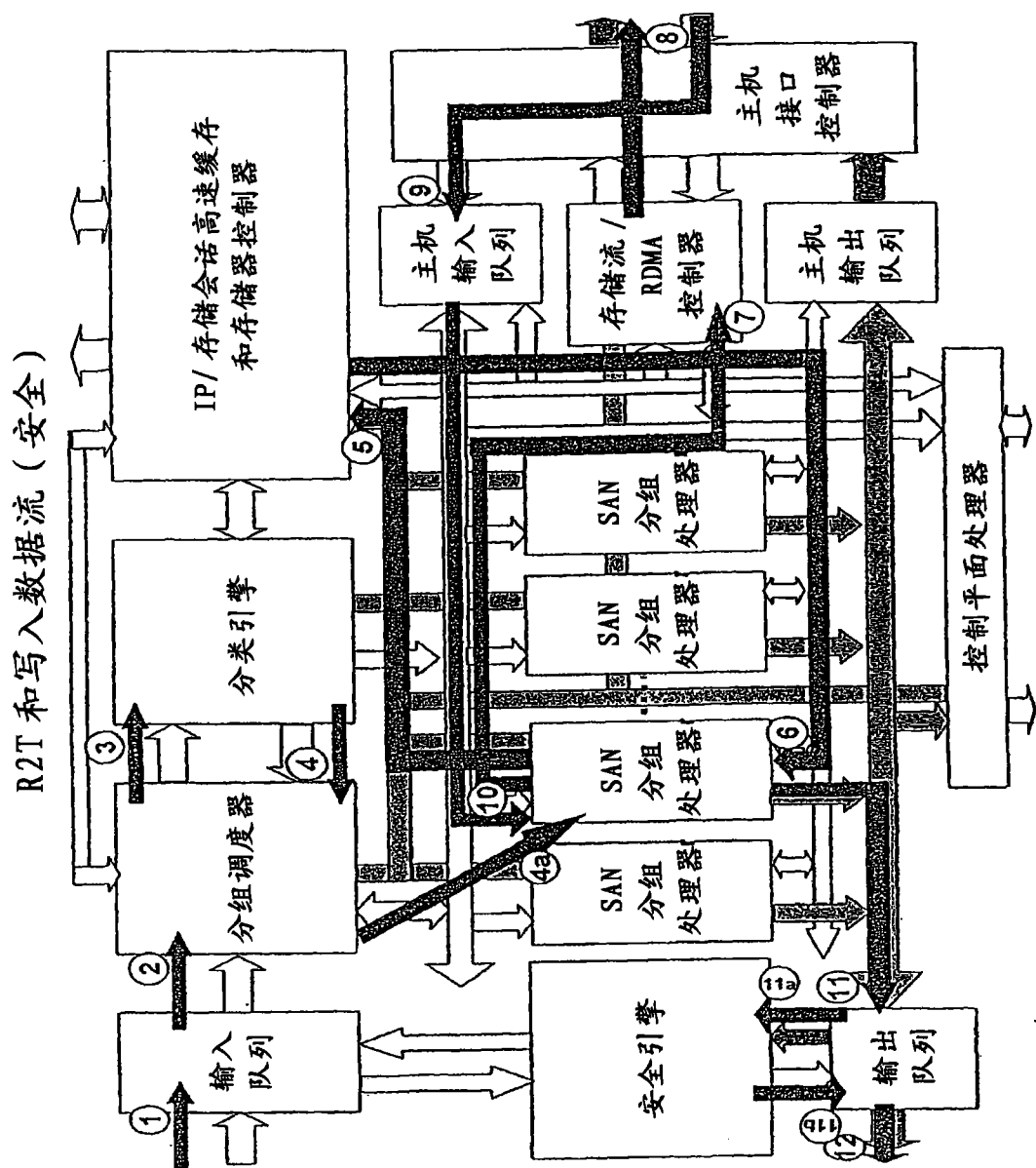


图 42



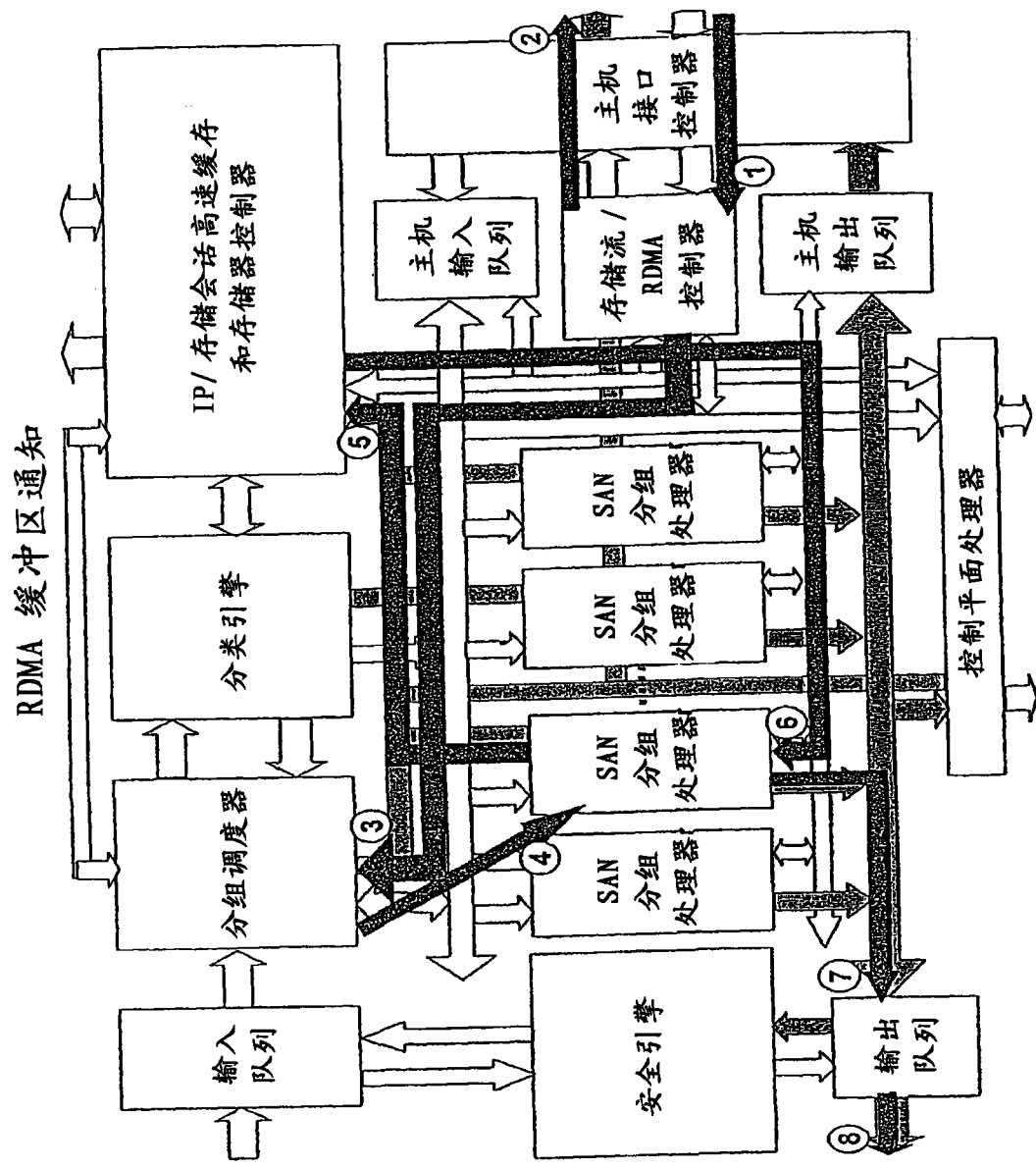
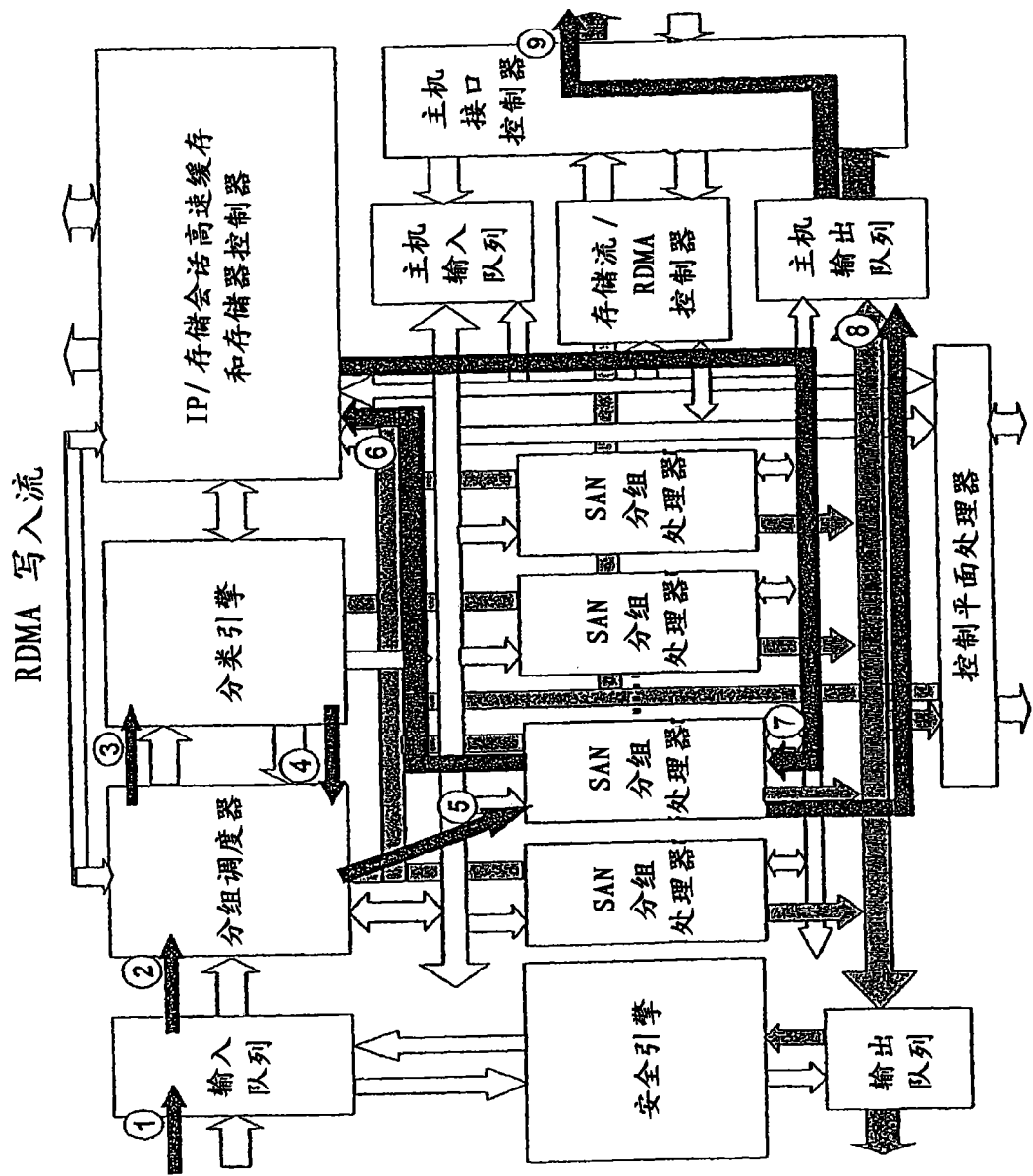
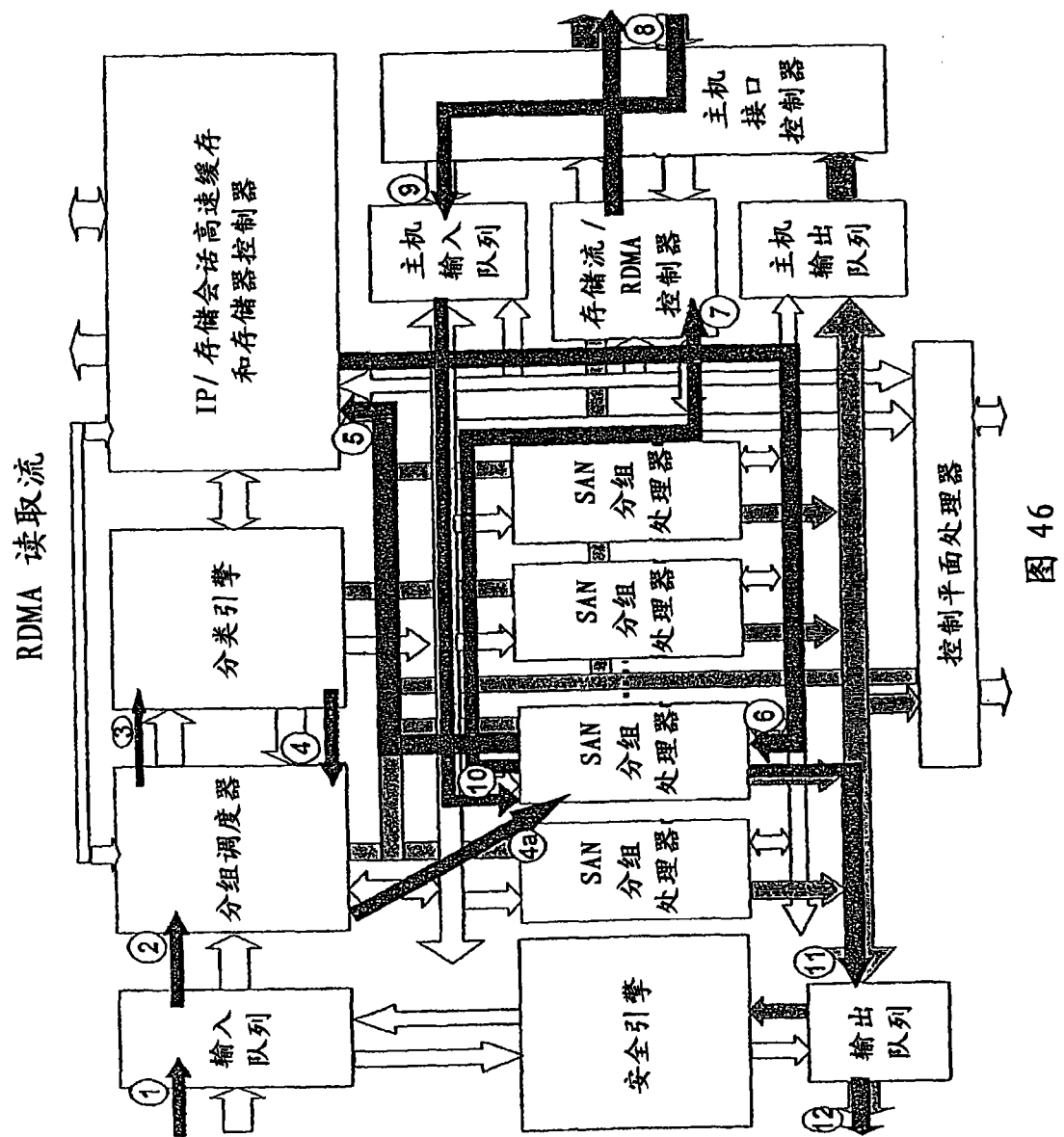
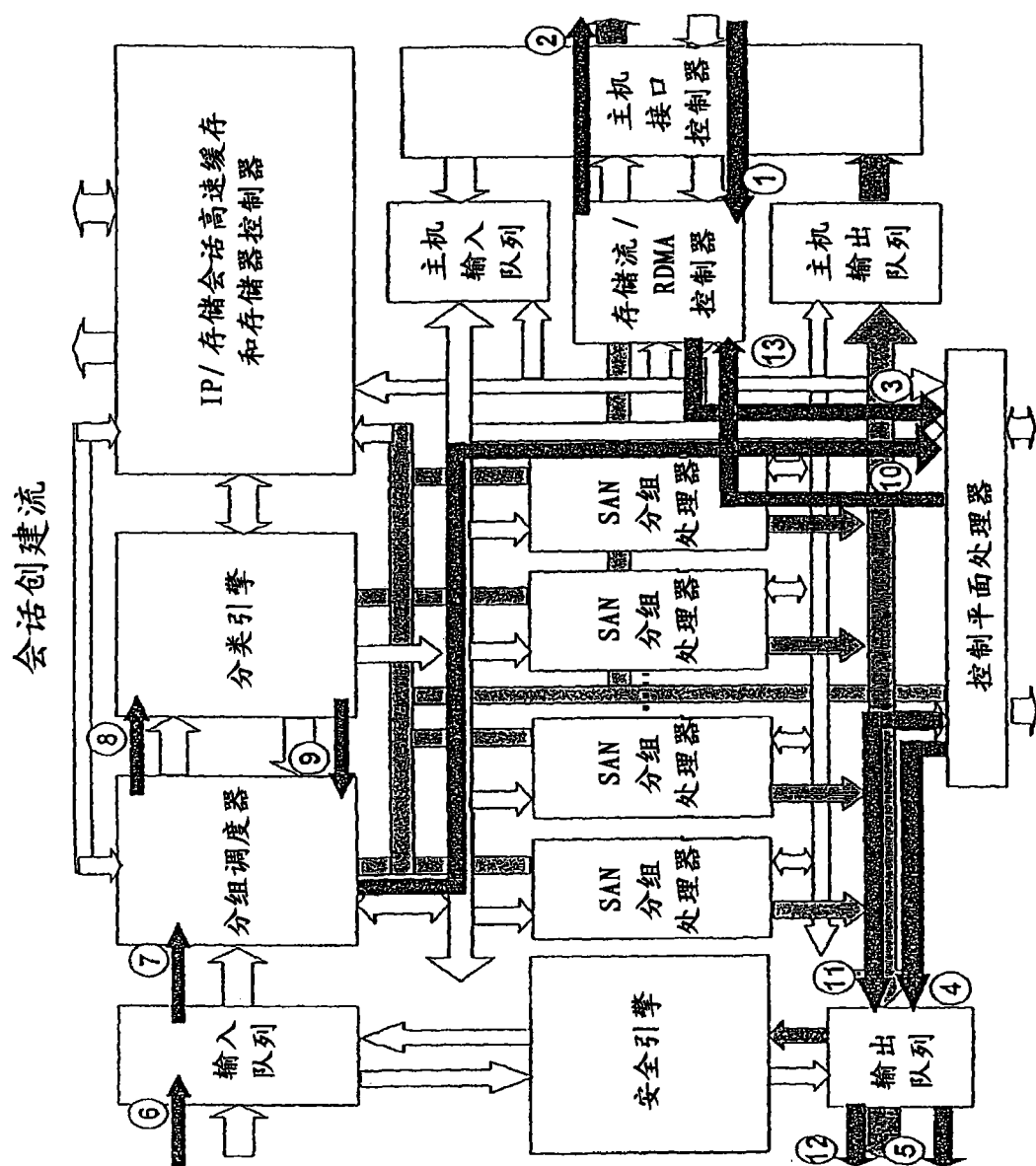


图 44







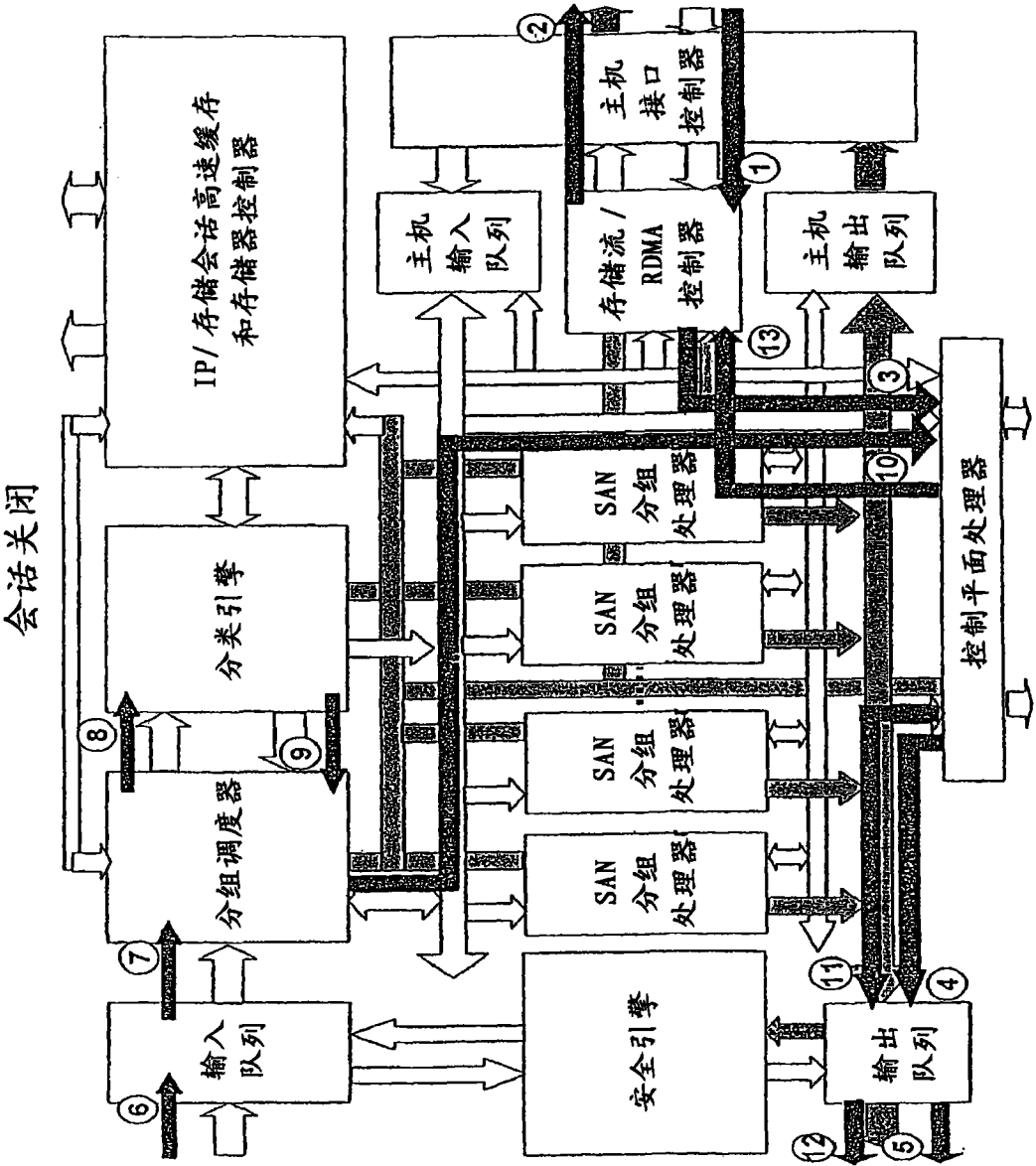


图 48

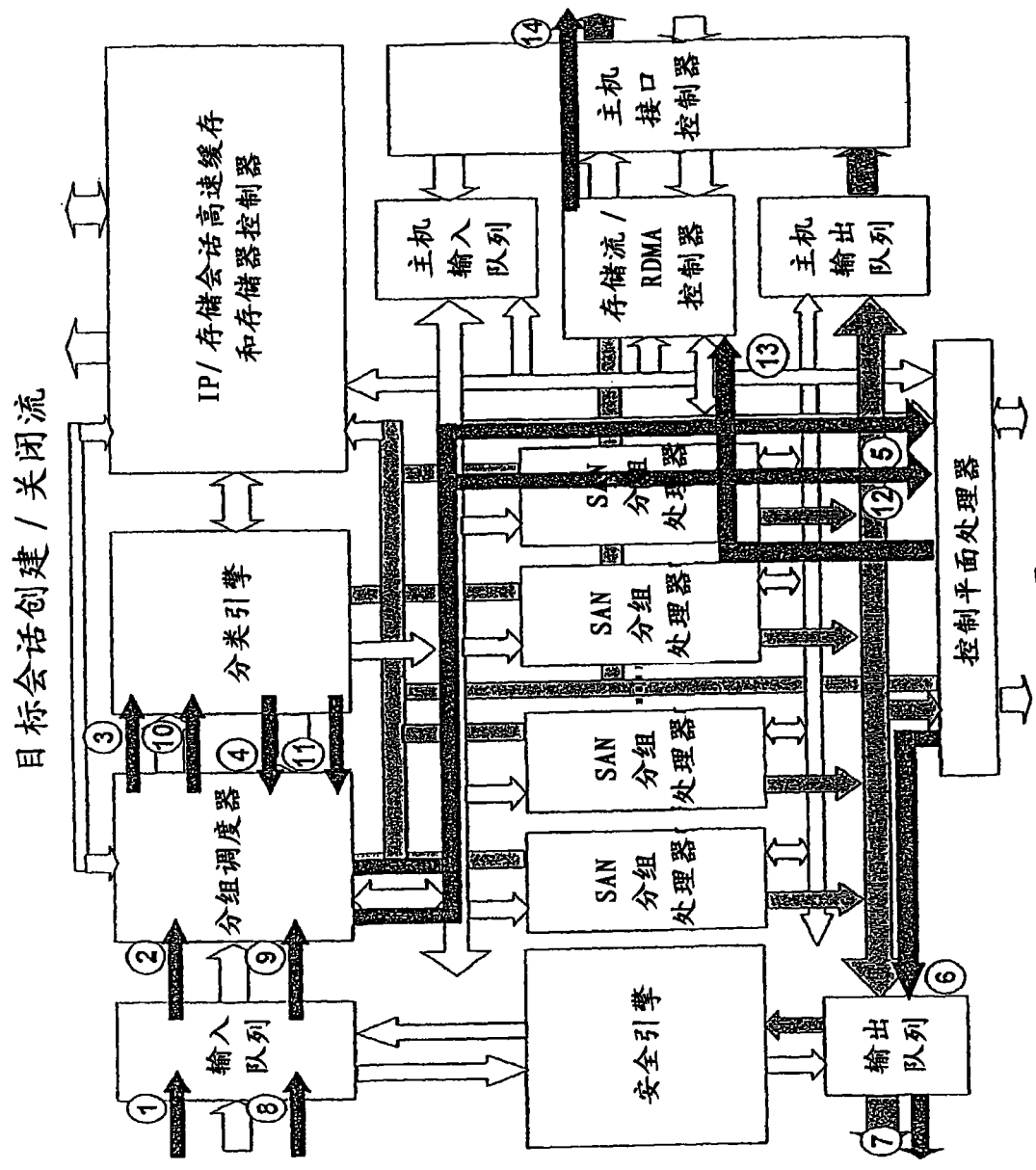


图 49

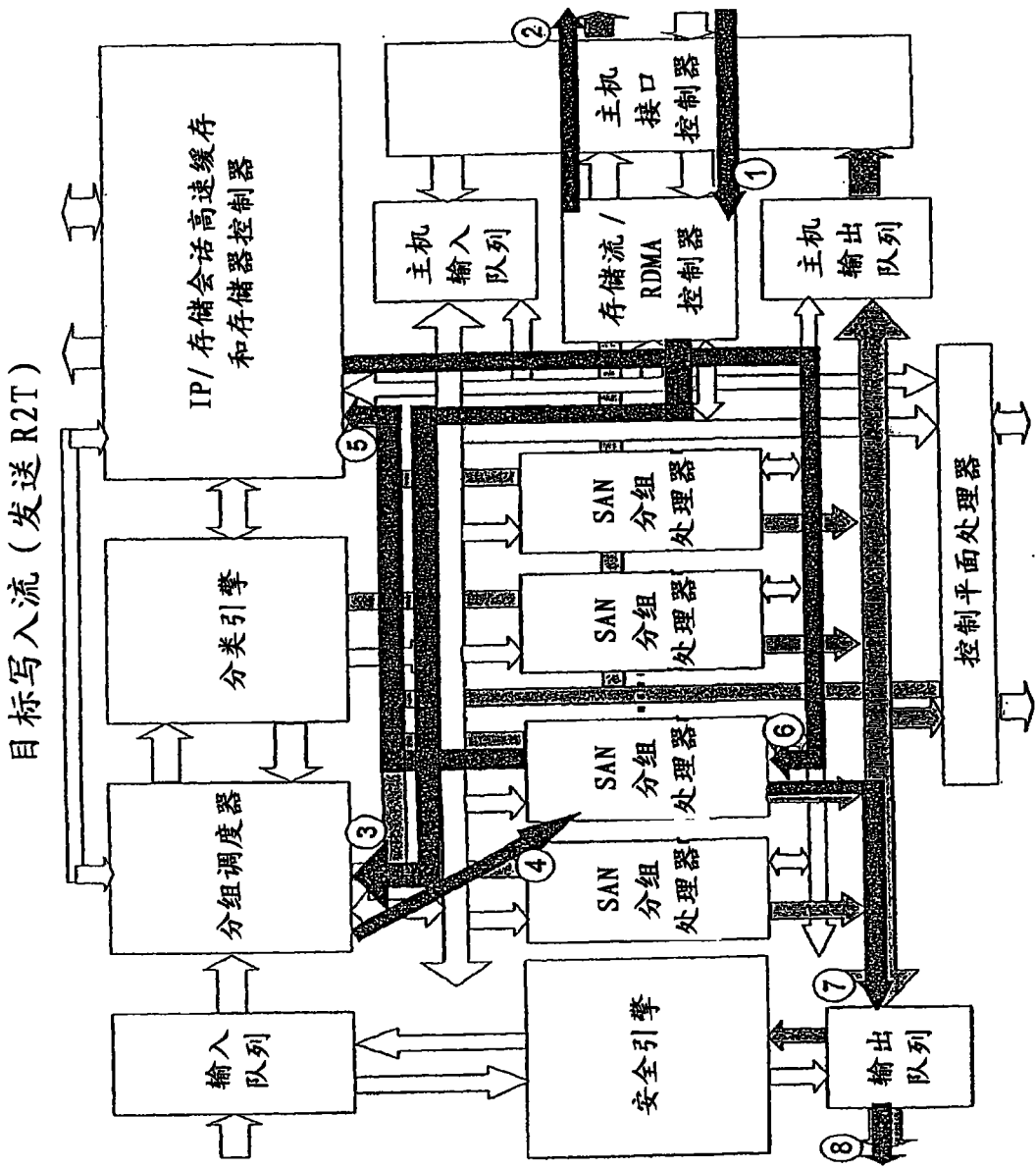


图 50

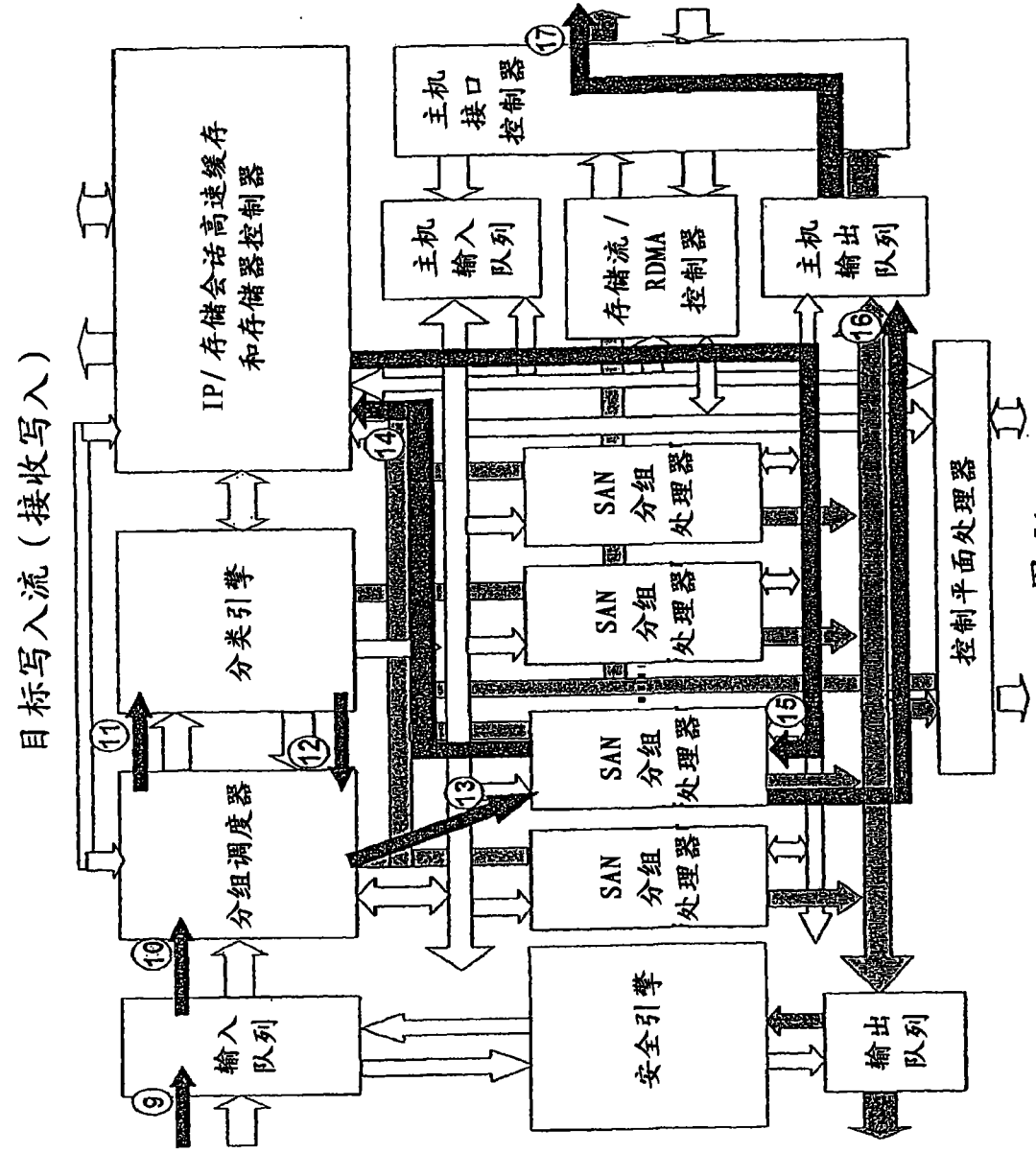


图 51

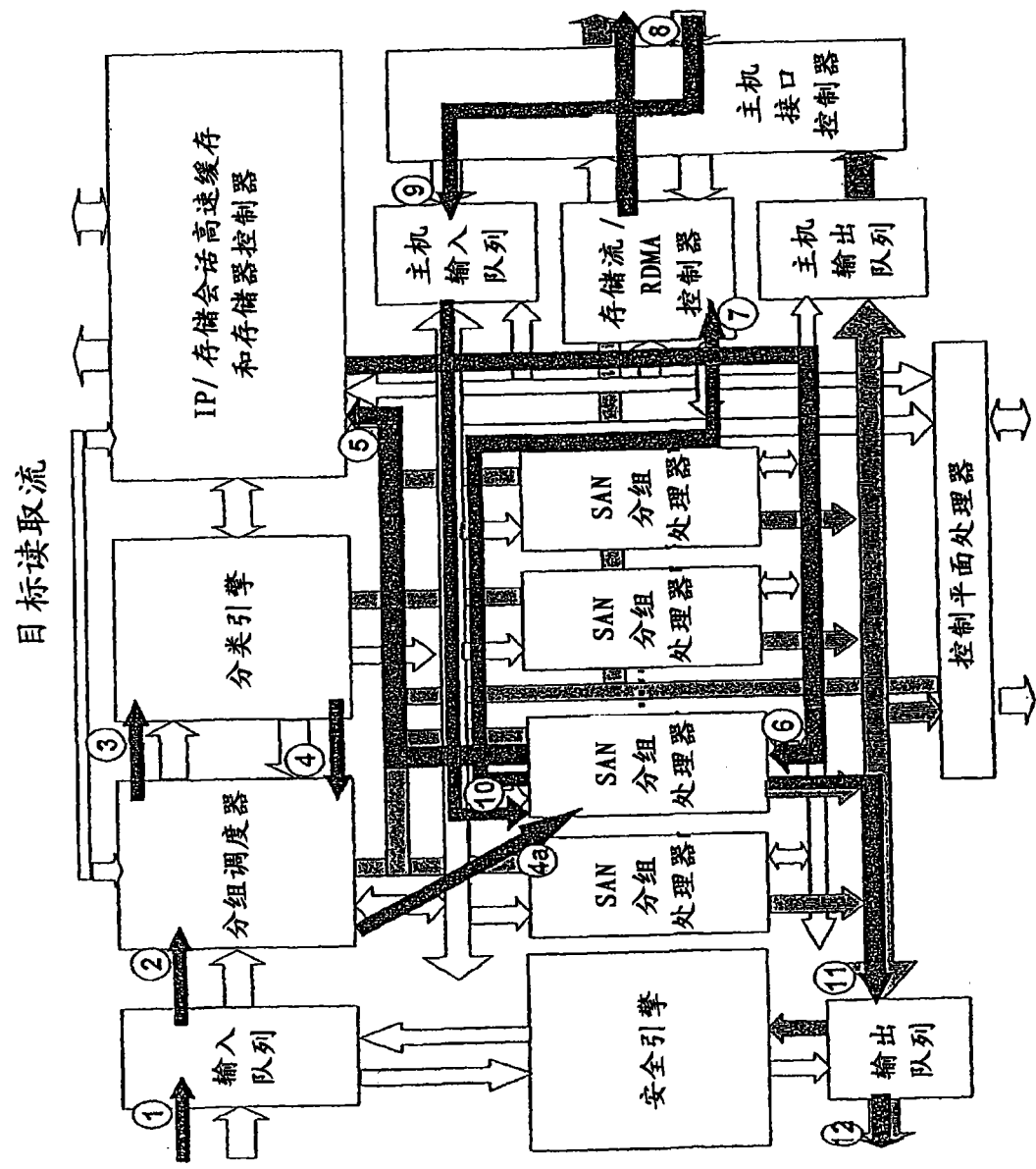


图 52

带有外围安全性的企业网络

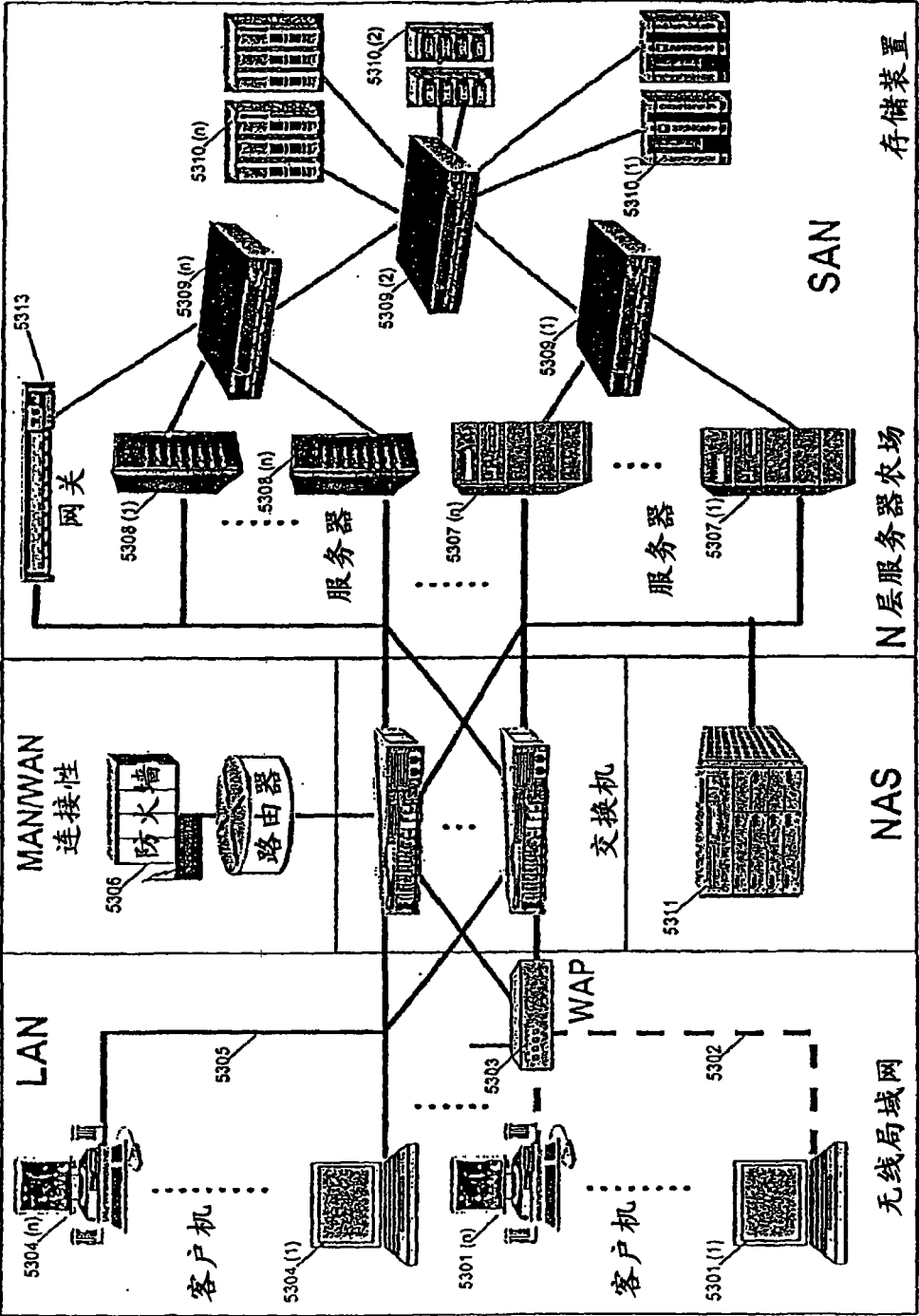


图 53

带有分布式安全性的企业网络

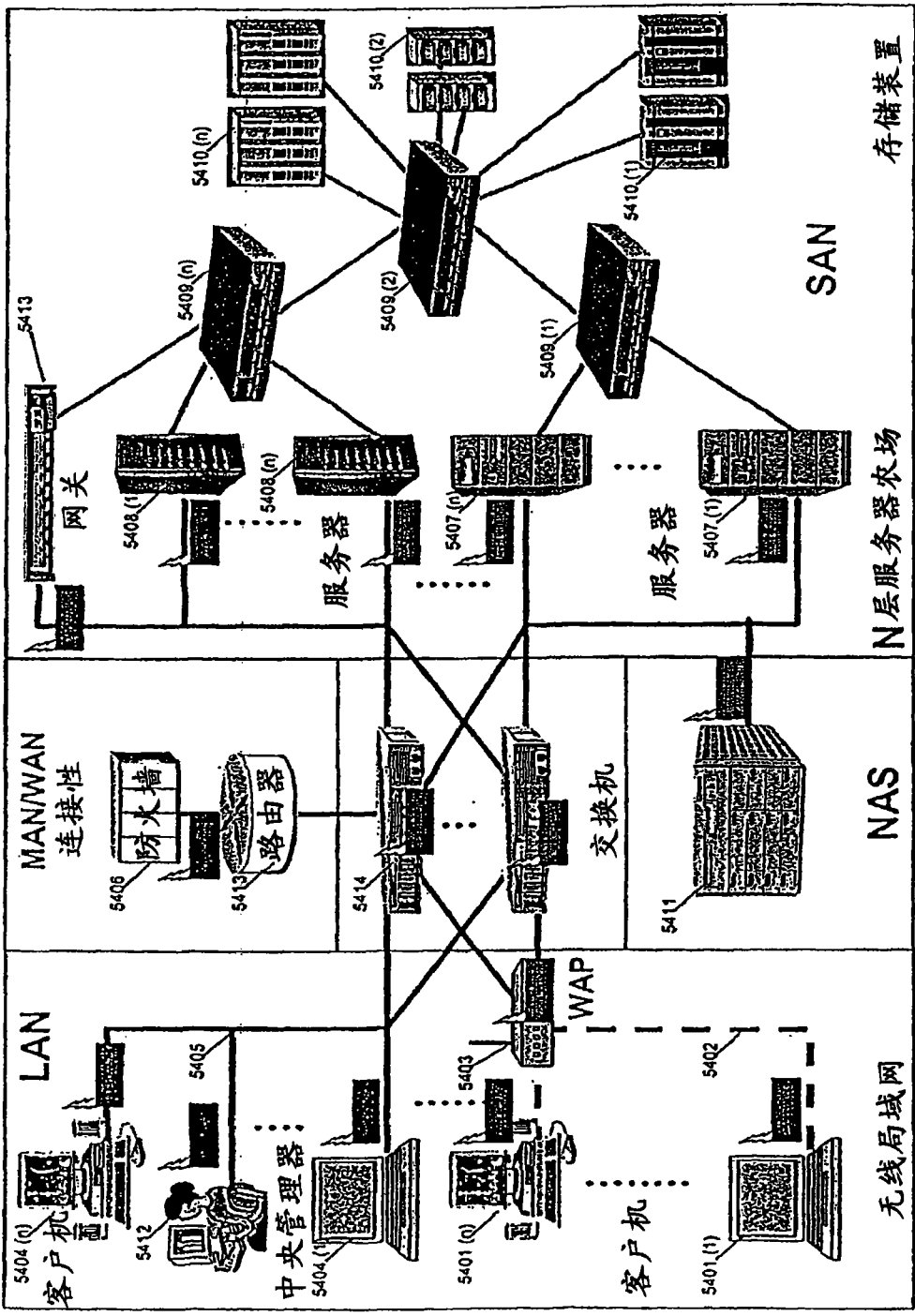
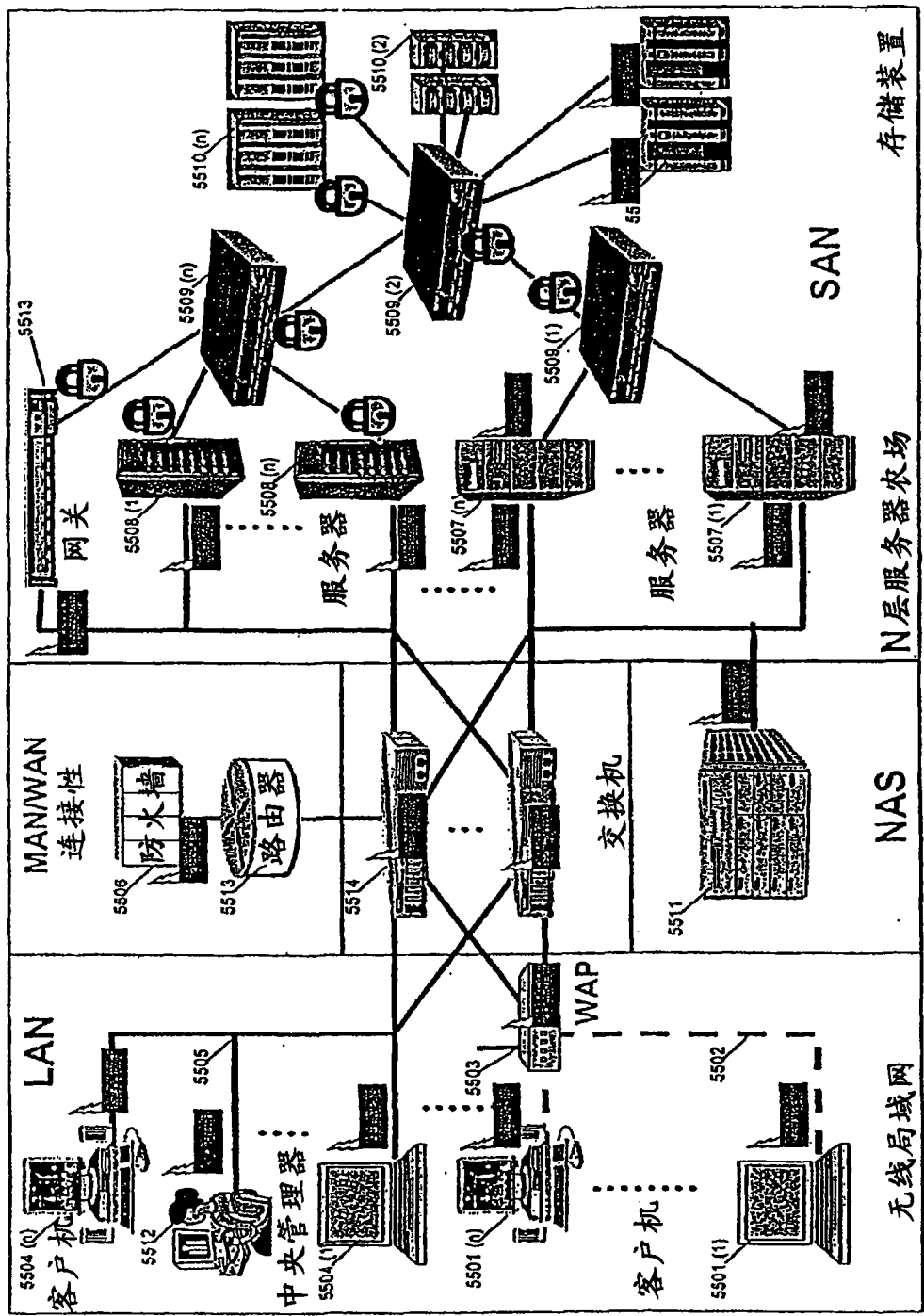


图 54

带有分布式安全性的企业网络



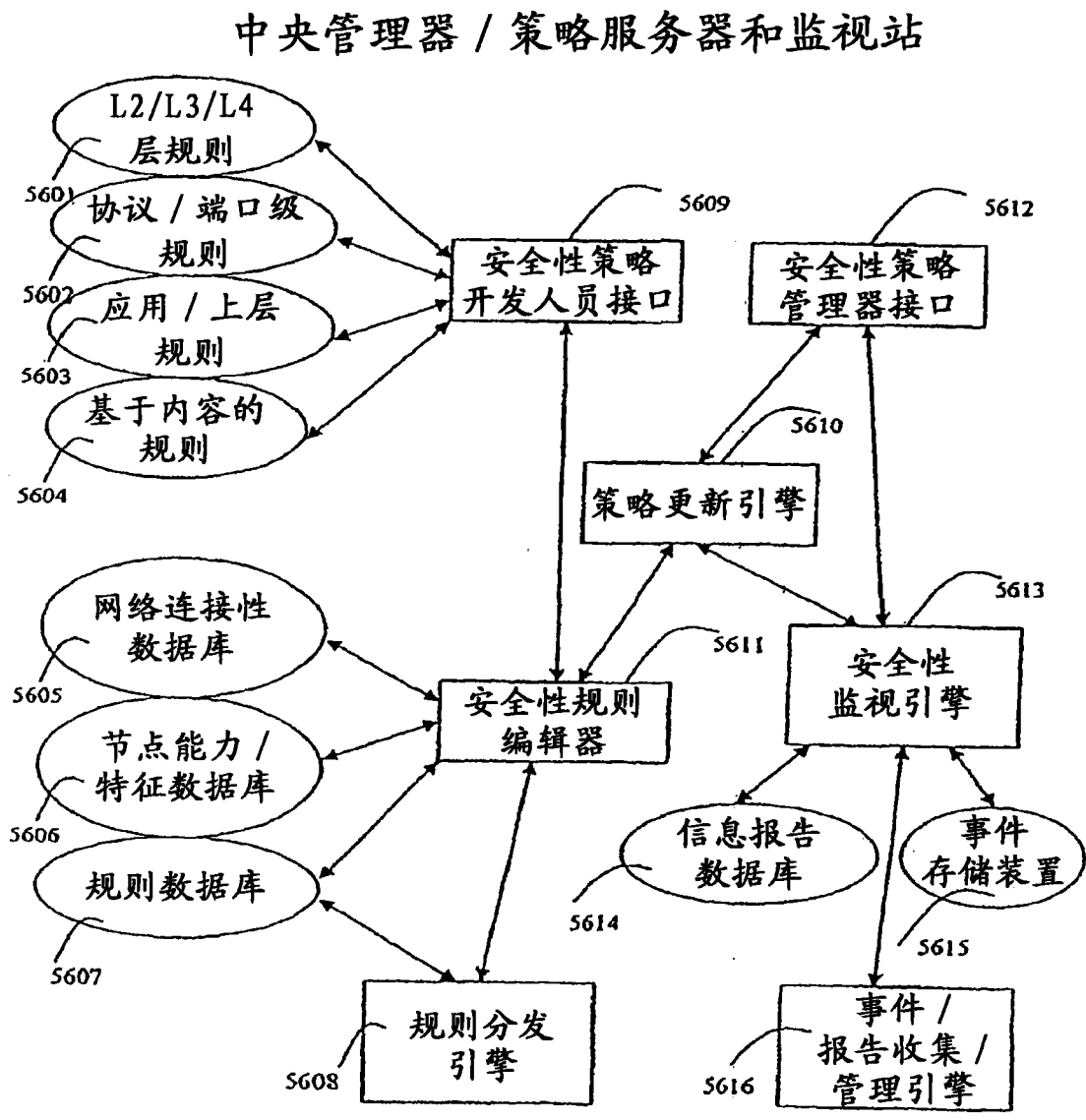


图 56

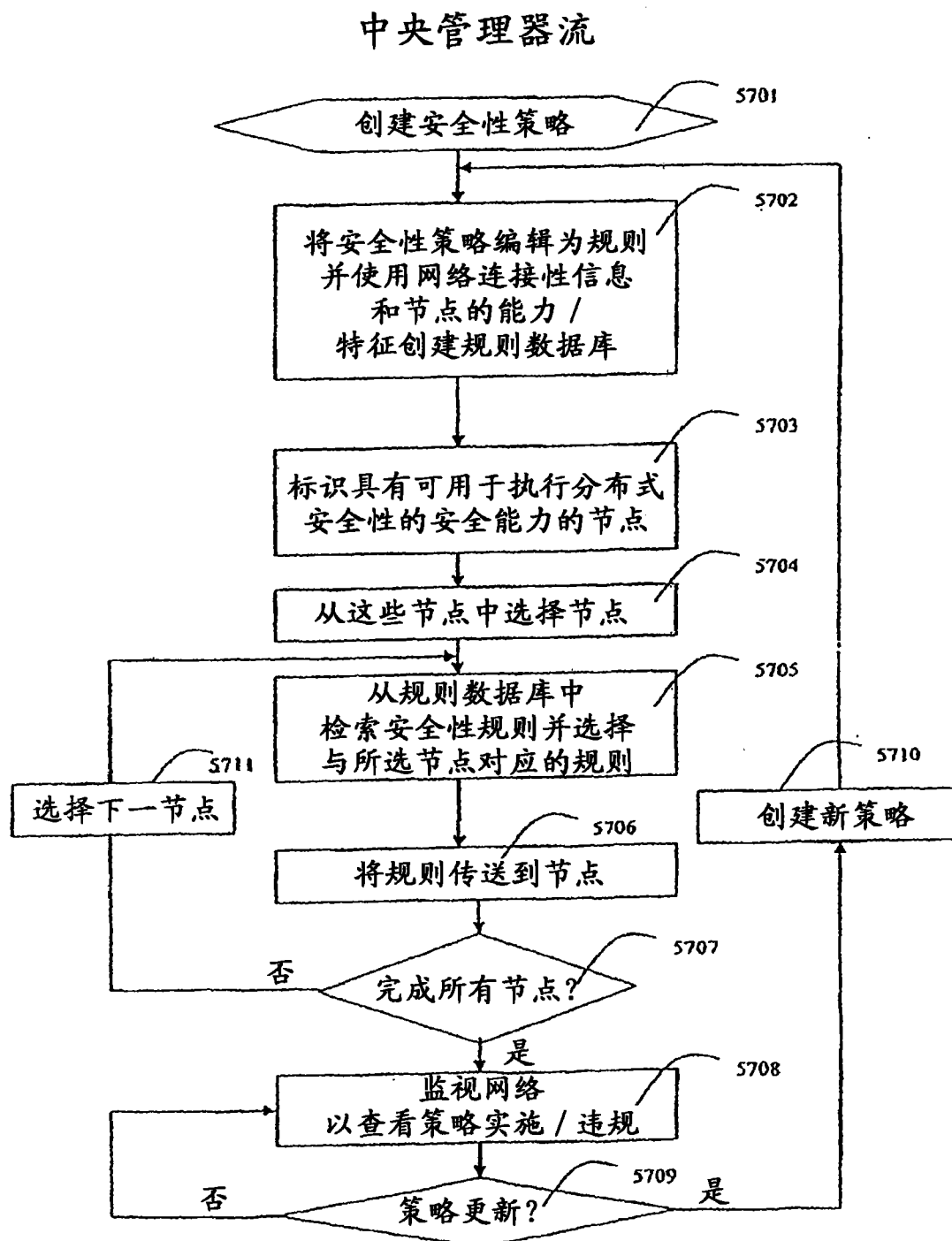


图 57

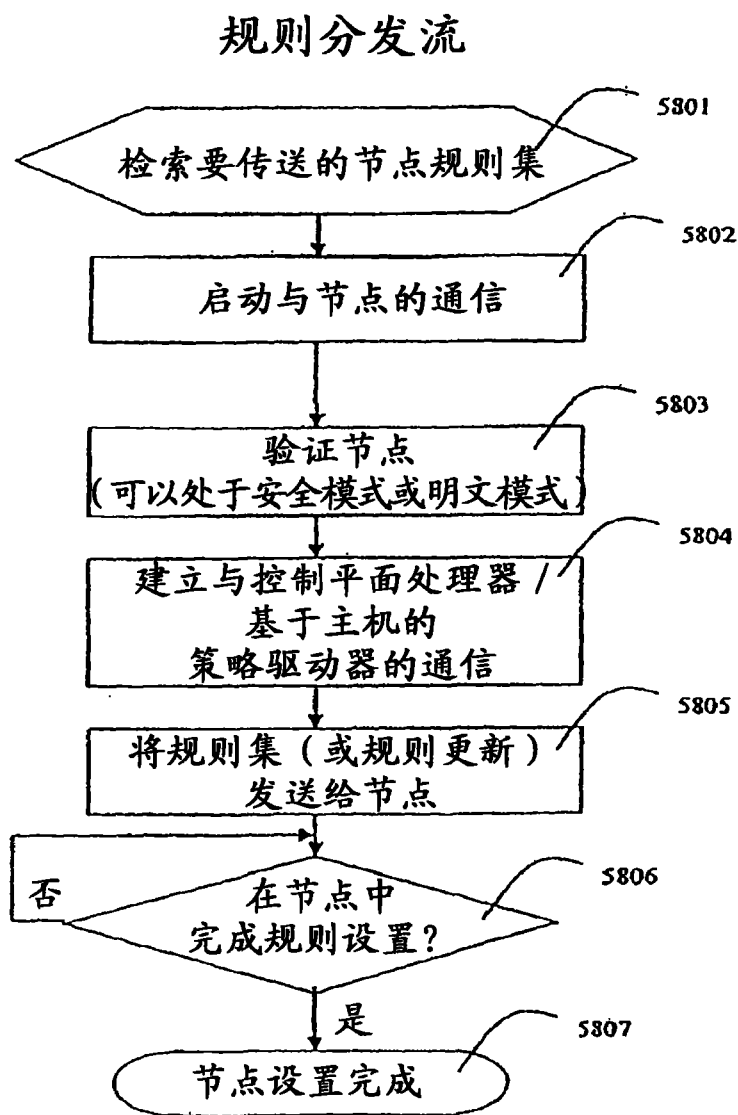


图 58

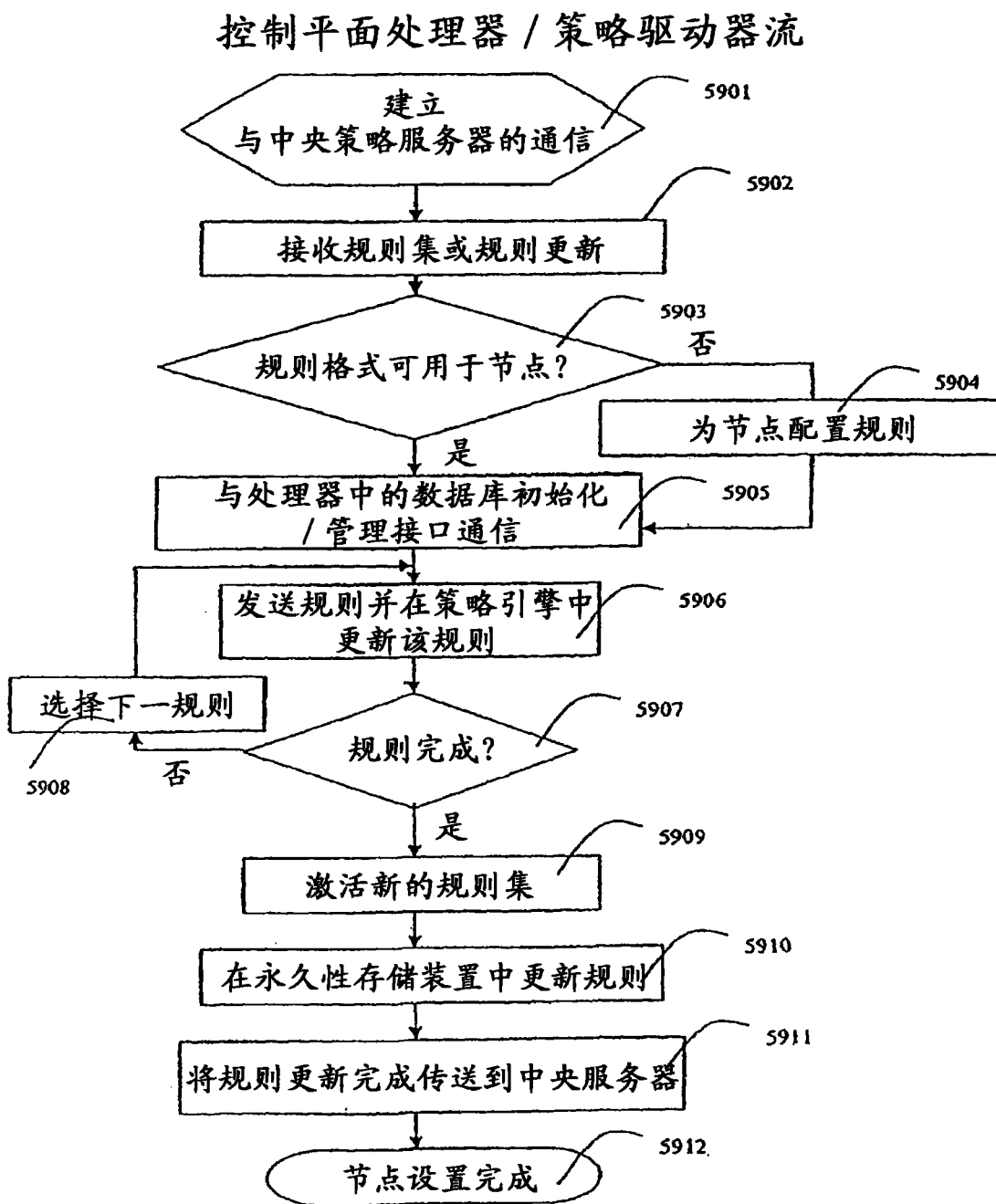


图 59

分组过滤规则实例

方向	源	目的地	协议	源端口	目的地端口	其他（例如，应用层、用户名、URL 协议等）	操作
入	外部	内部	TCP	> 1023	25	任意	允许
出	内部	外部	TCP	25	> 1023	任意	允许
入	内部	内部	TCP	任意	80	从 8 点到 17 点	允许
出	内部	内部	TCP	80	任意	任意	允许
入	外部	内部	ICMP	任意	任意	任意	拒绝
入	外部	内部	TCP	任意	80	有效负载中的长 URL	拒绝
出	内部	内部	UDP	5004	> 1023	任意	允许
出	内部	外部	UDP	5004	> 1023	任意	拒绝
出	内部	外部	UDP	80	任意	社会保险号	拒绝
任一	任意	任意	任意	任意	任意	任意	拒绝

图 60

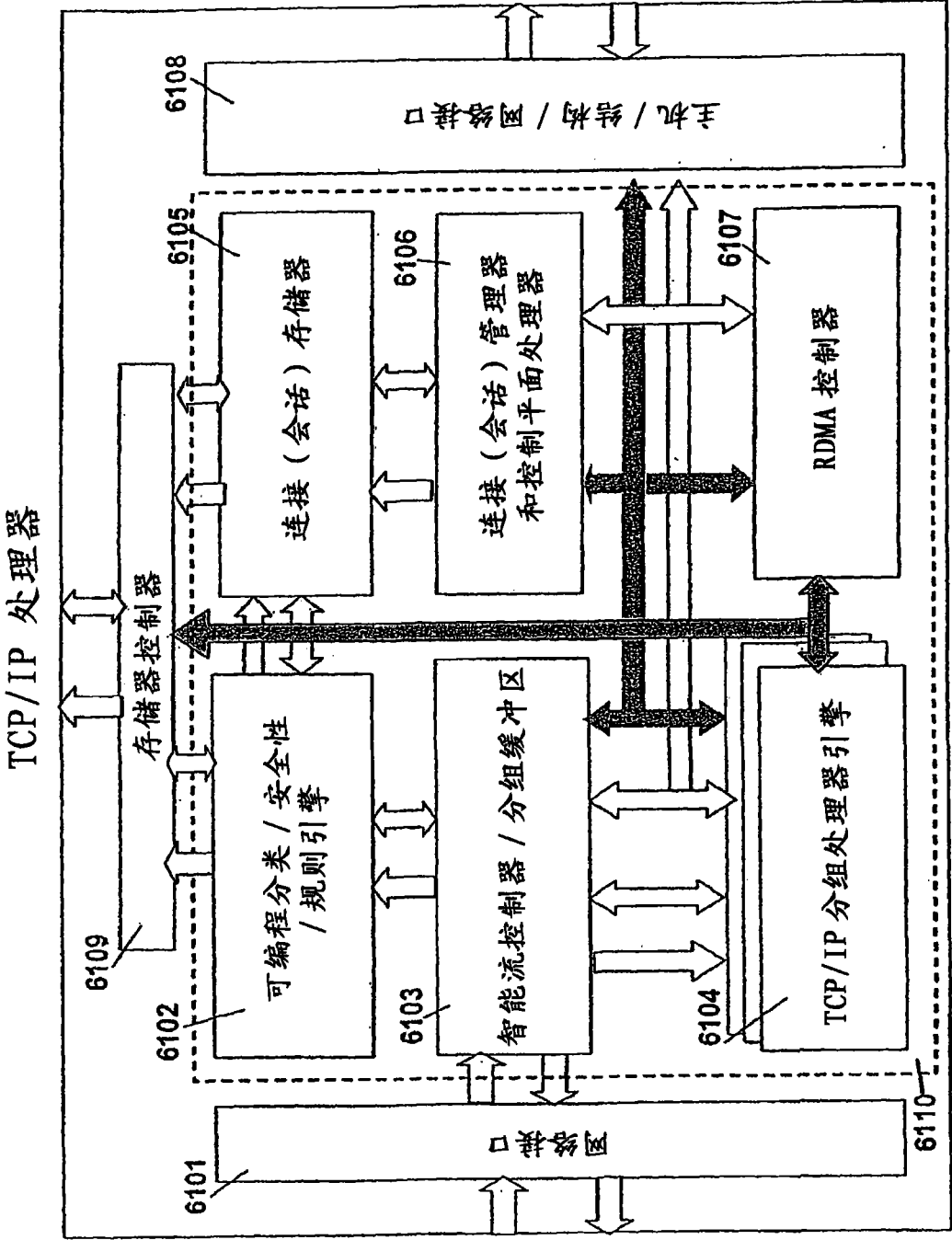


图 61

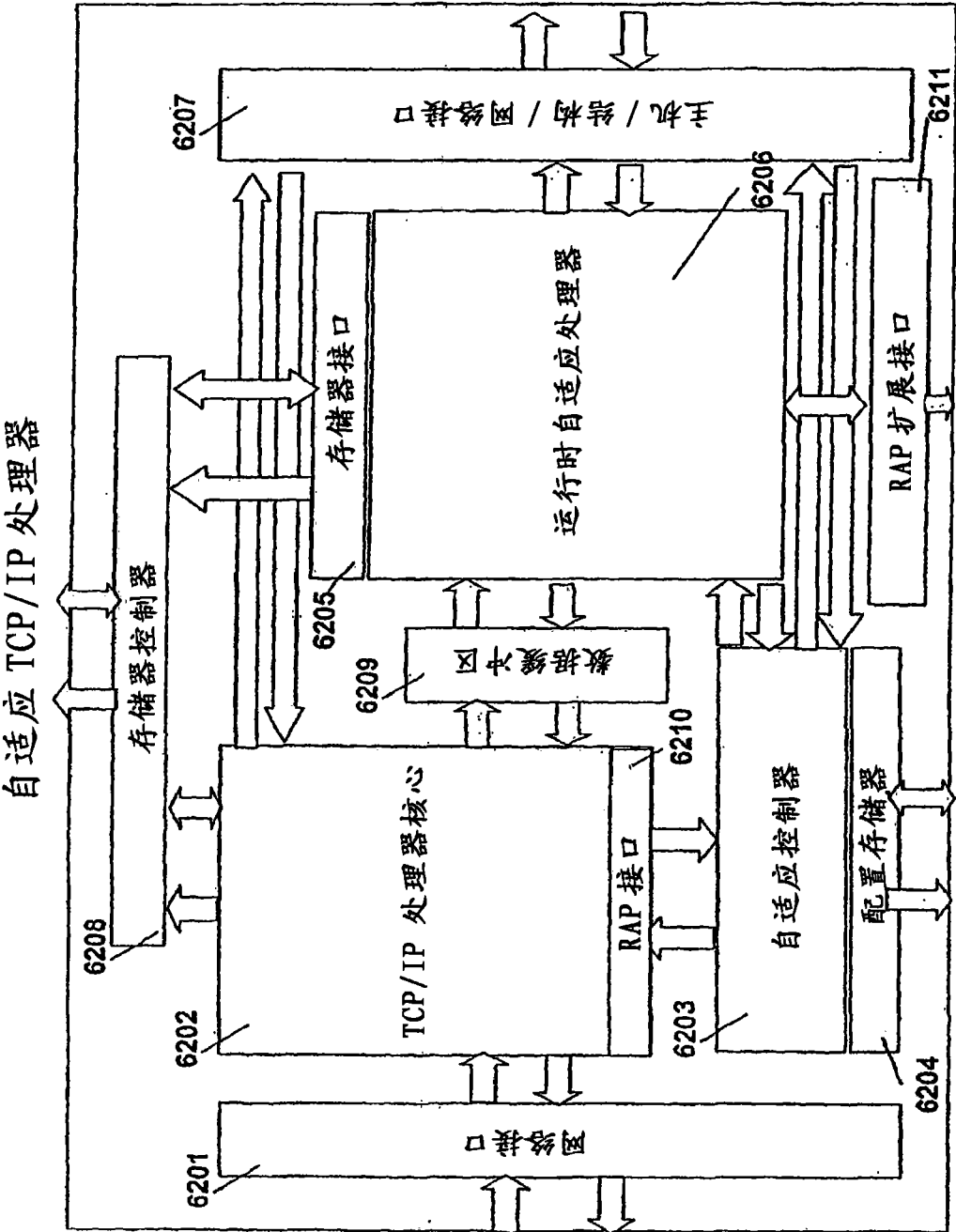


图 62

备选自适应 TCP/IP 处理器

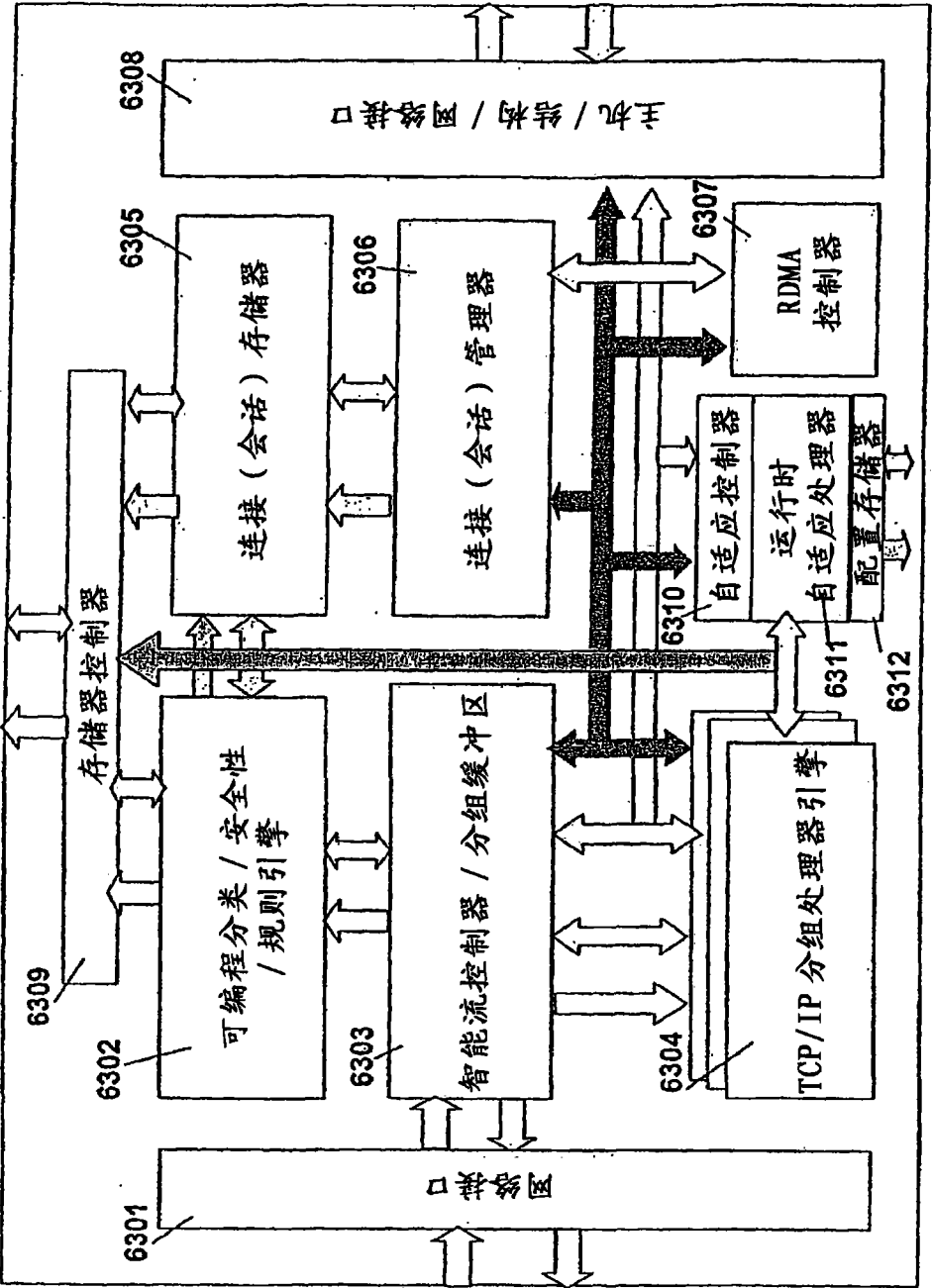
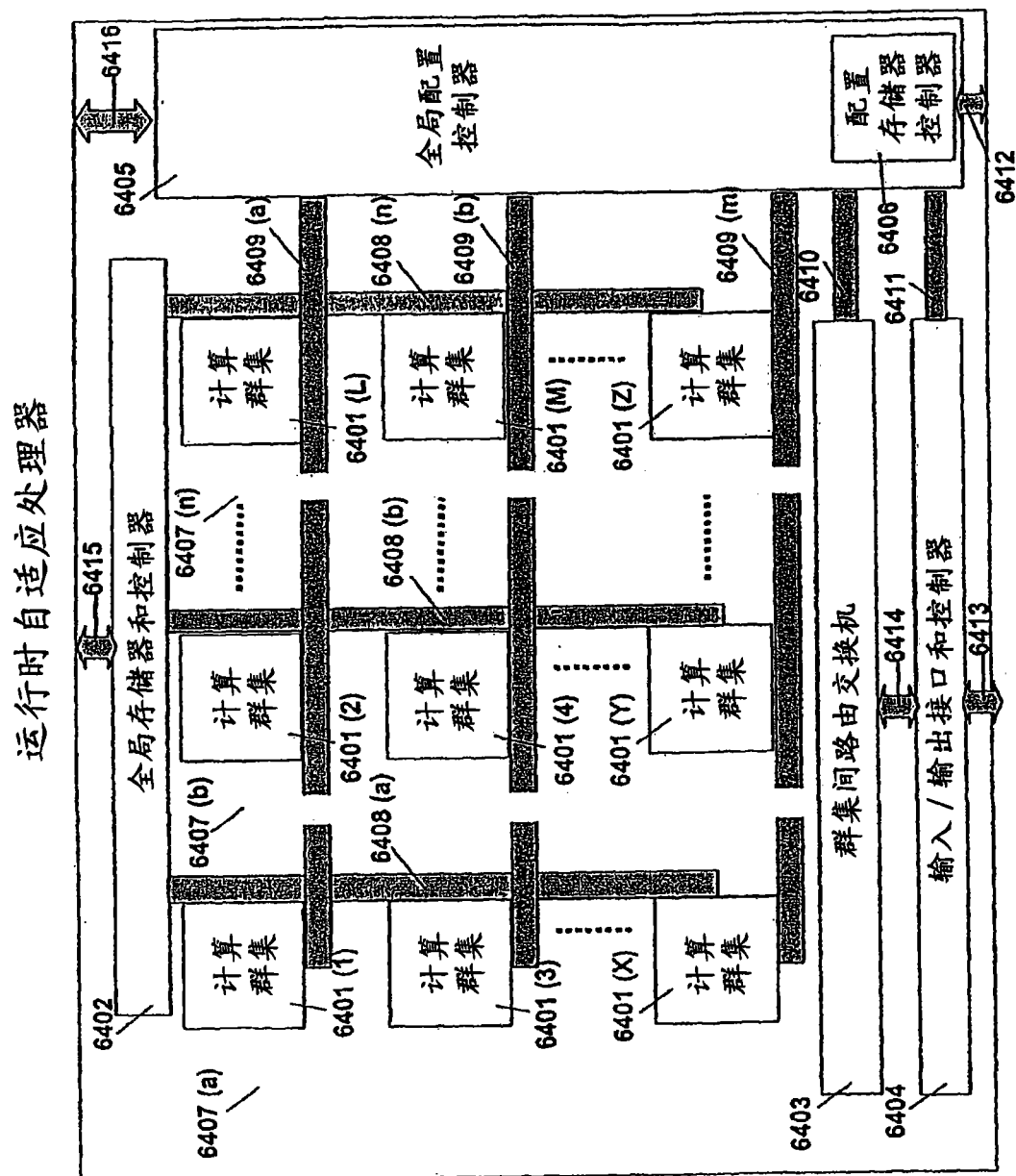


图 63



计算群集

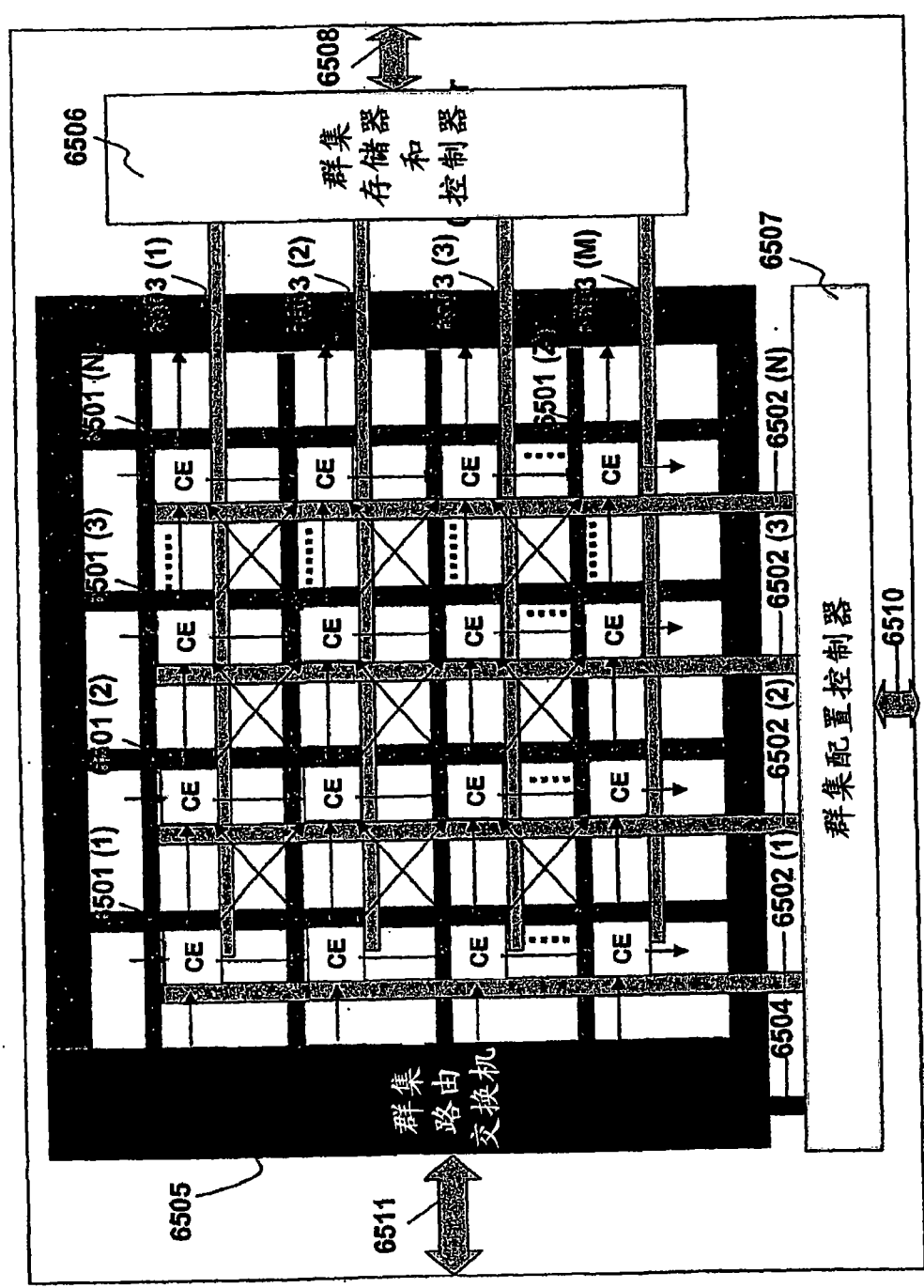
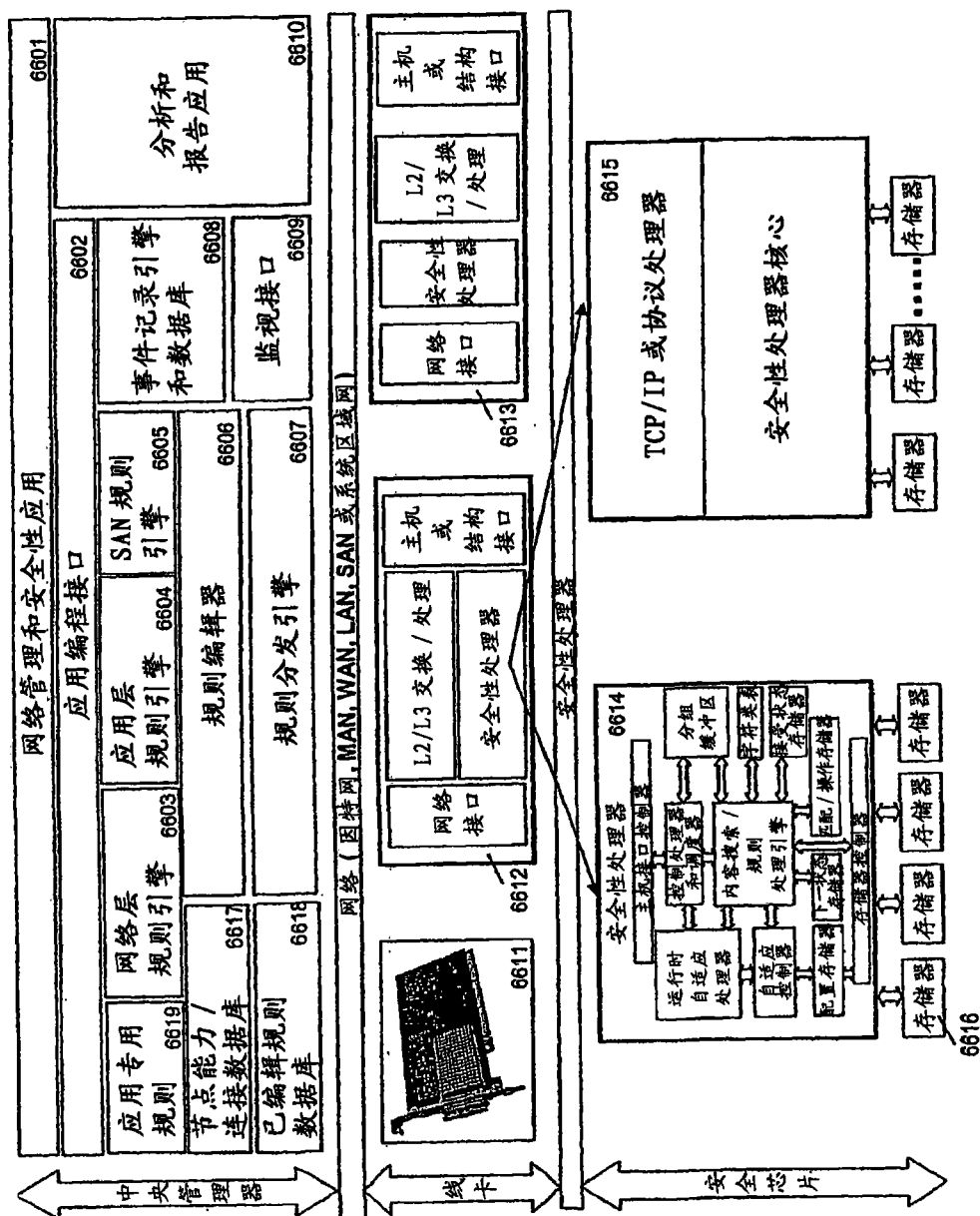


图 65

安全性解决方案



四 66

安全解决方案编辑器流

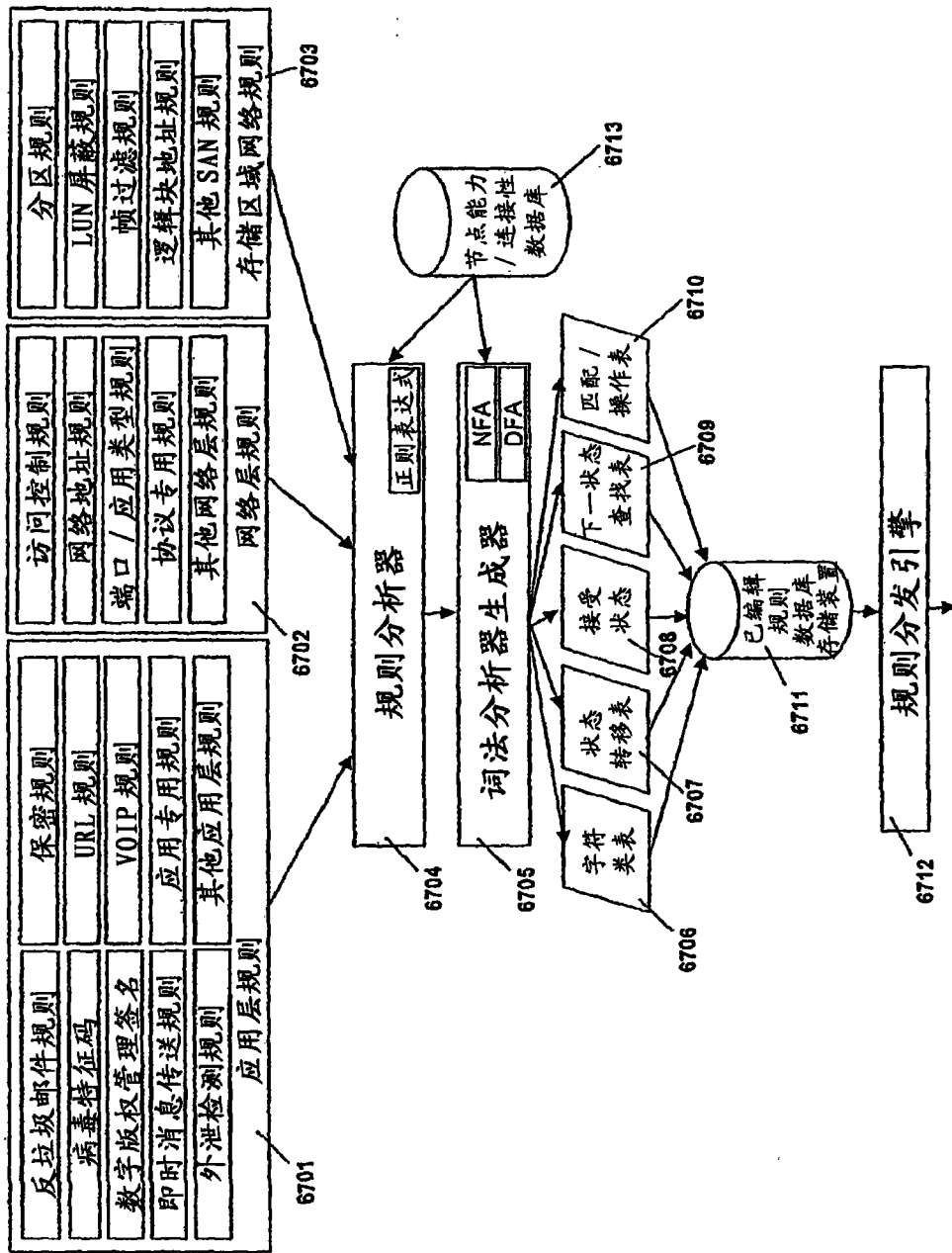


图 67

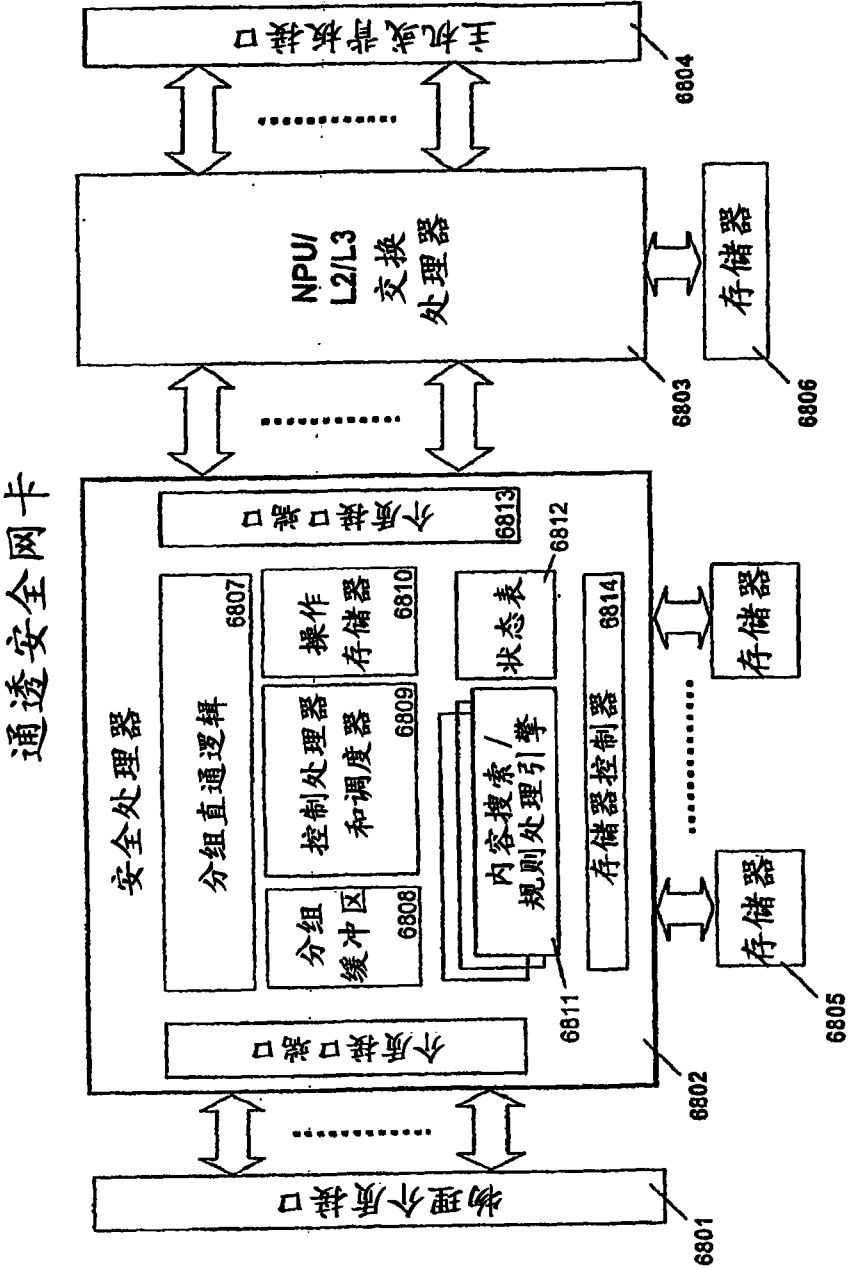


图 68

具有后援安全性的网络线卡

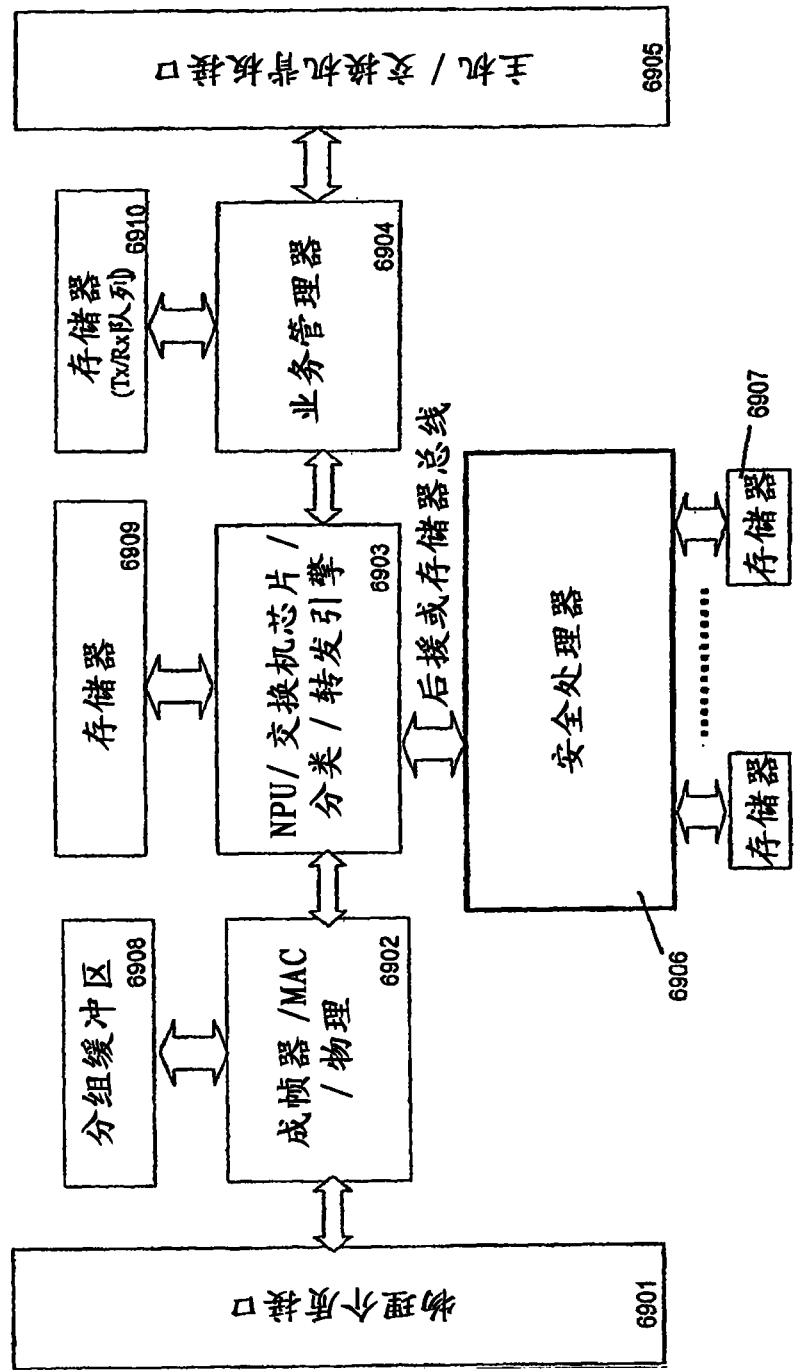


图 69

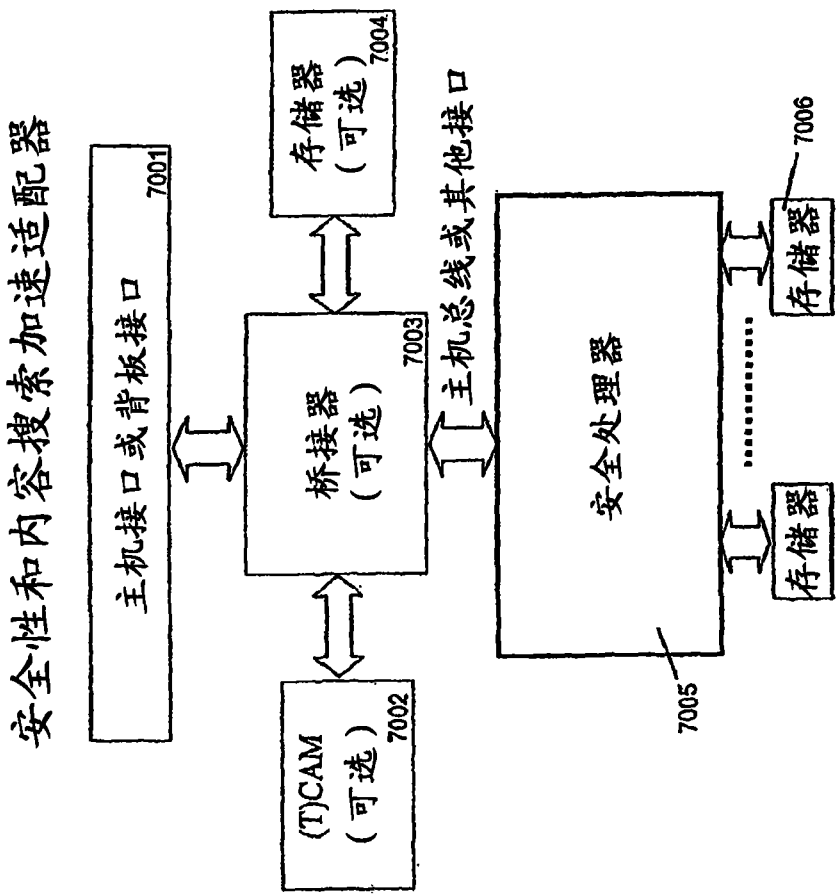
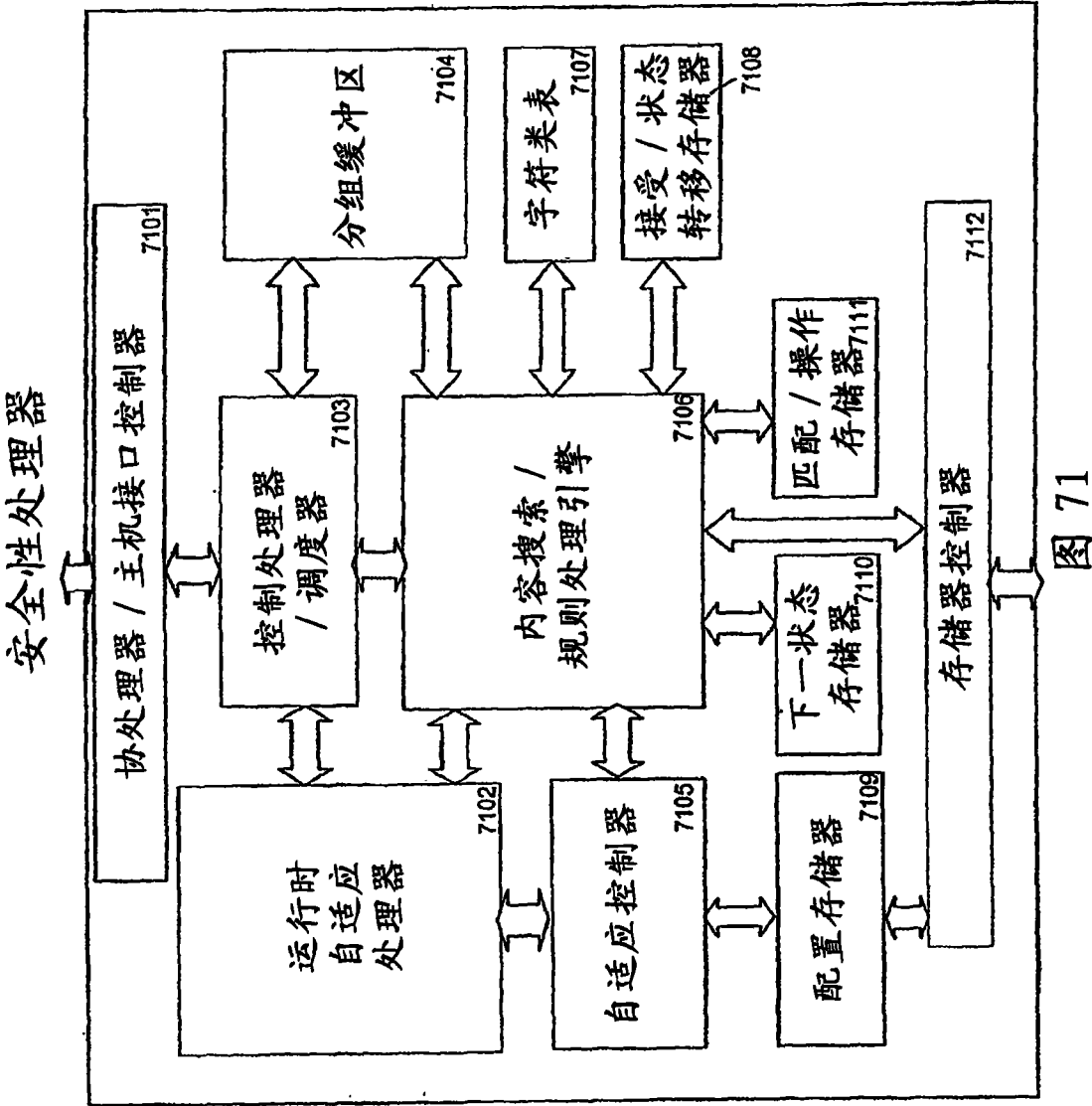
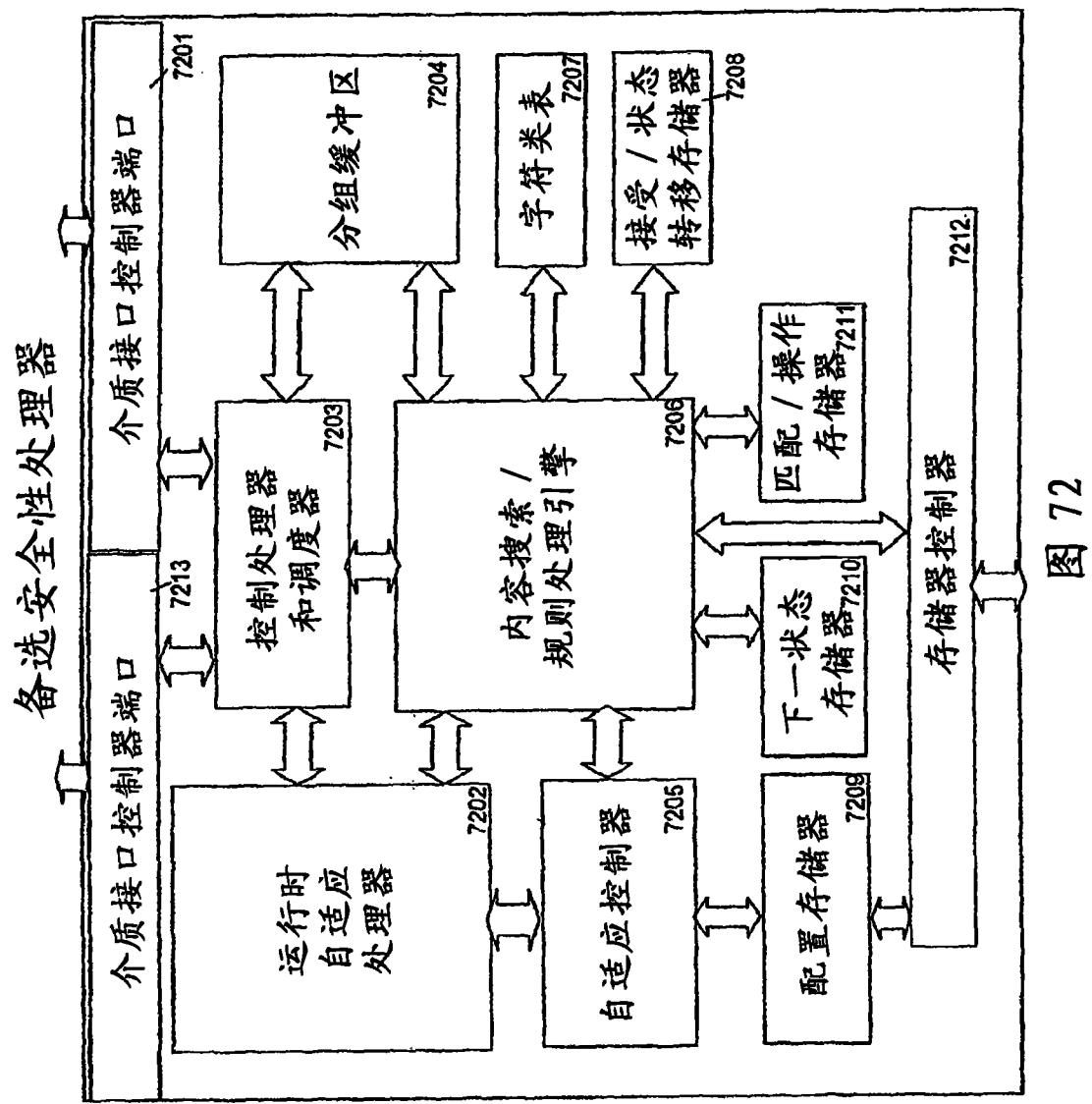


图 70





第三安全性处理器

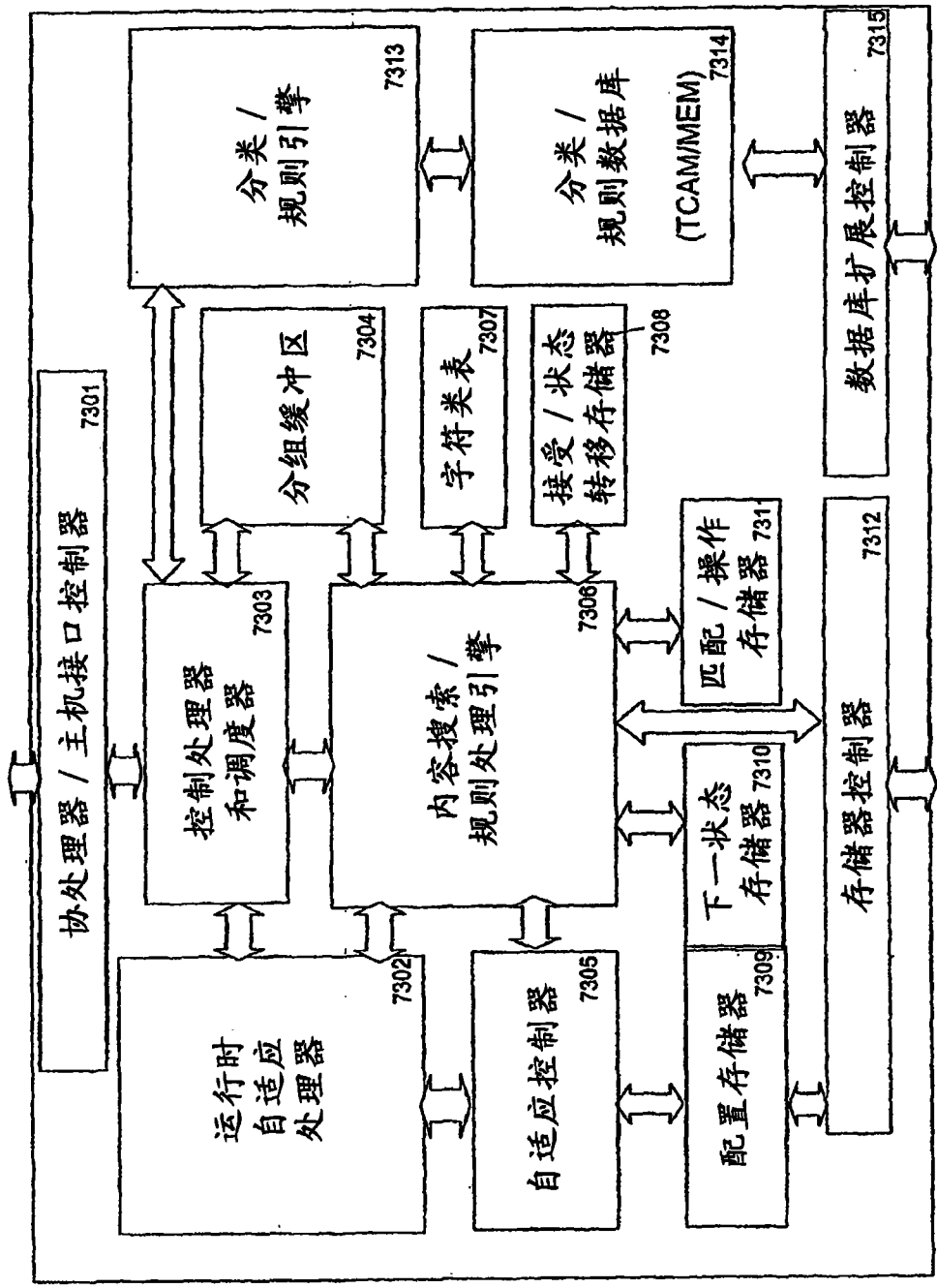


图 73

内容搜索和规则处理引擎

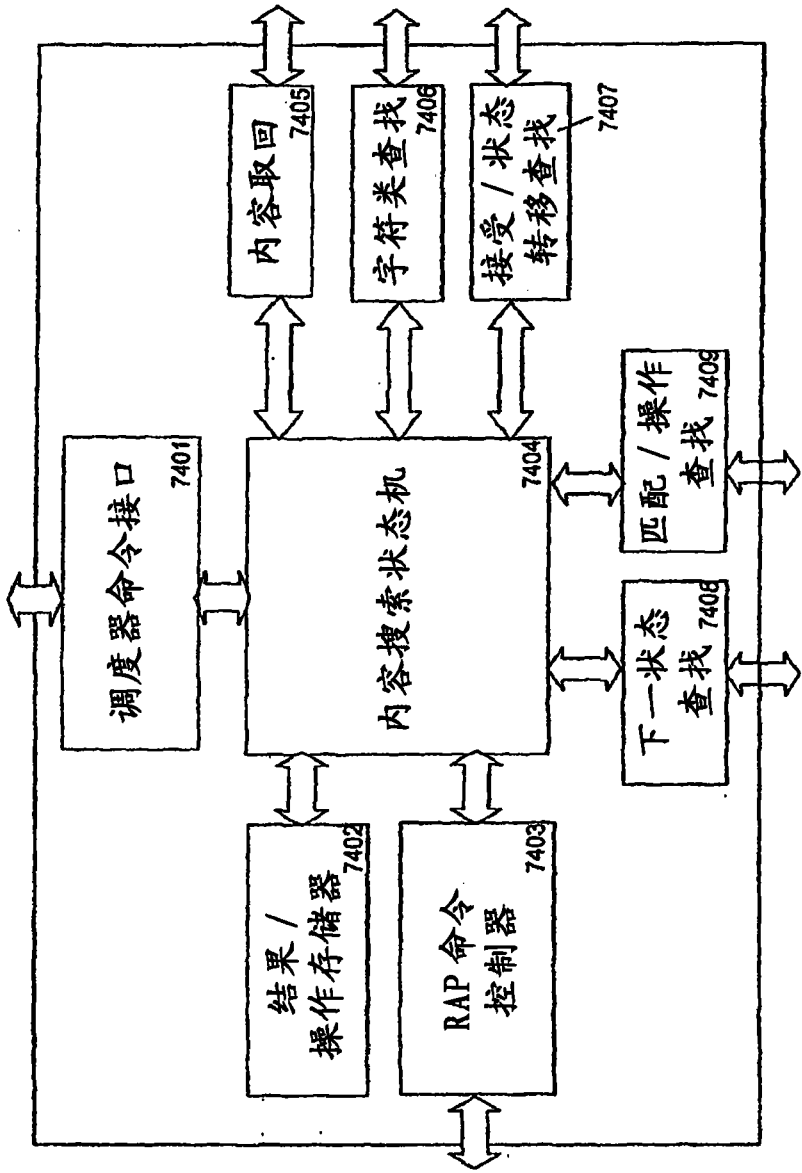


图 74

正则表达式规则实例（现有技术）

规则 #	规则组	正则表达式	操作 / 标记
R1:	<COMMENT>	[^\\ \"'\\n\\r]+	/* 跳过注释 */
R2:	<COMMENT>	->	/* 跳过注释 */
R3:	<COMMENT>	{2,} >	START(INITIAL);
R4:	<FNT>	[^\\ \"'> \\n]+	/* 跳过 FONT 标记 */
R5:	<FNT>	>	START(INITIAL);
R6:	<S_TAG>	!--	START(COMMENT);
R7:	<S_TAG>	FONT	START(FNT);
R8:	<S_TAG>	>	START(INITIAL);
R9:	<INITIAL>	<	START(S_TAG);

图 75

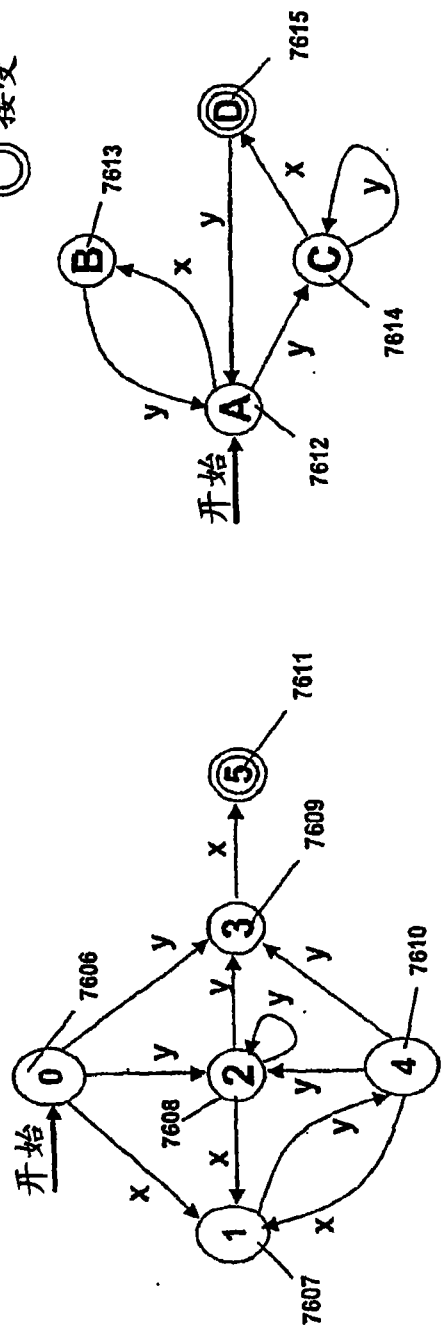
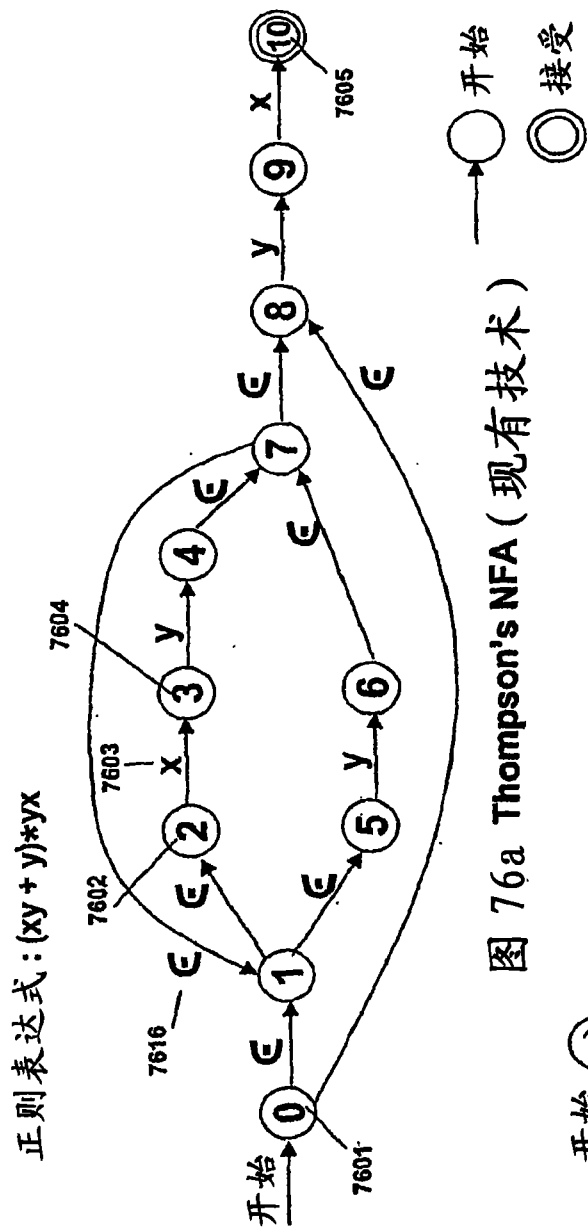


图 76c DFA (现有技术)

右侧偏置 NFA 和状态转移表
(现有技术)

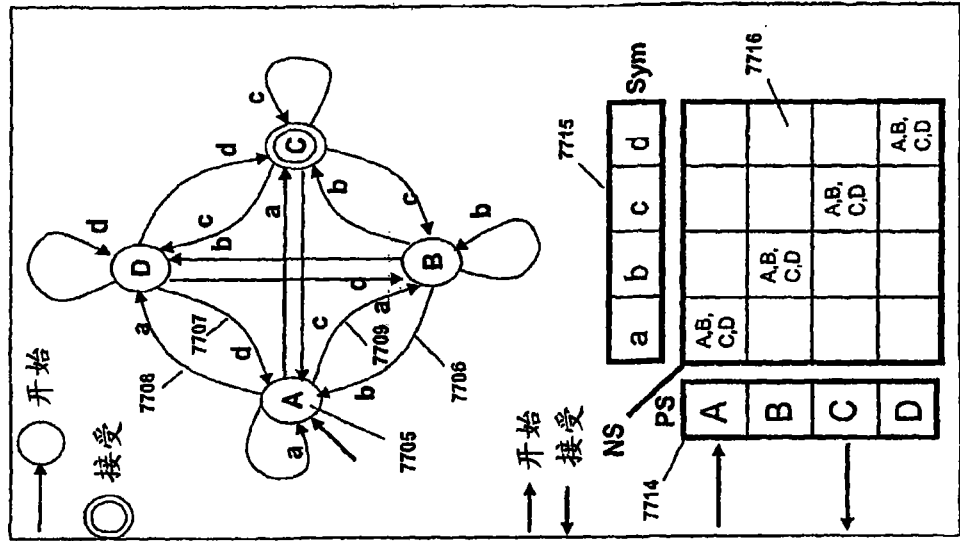


图 77b

左侧偏置 NFA 和状态转移表
(现有技术)

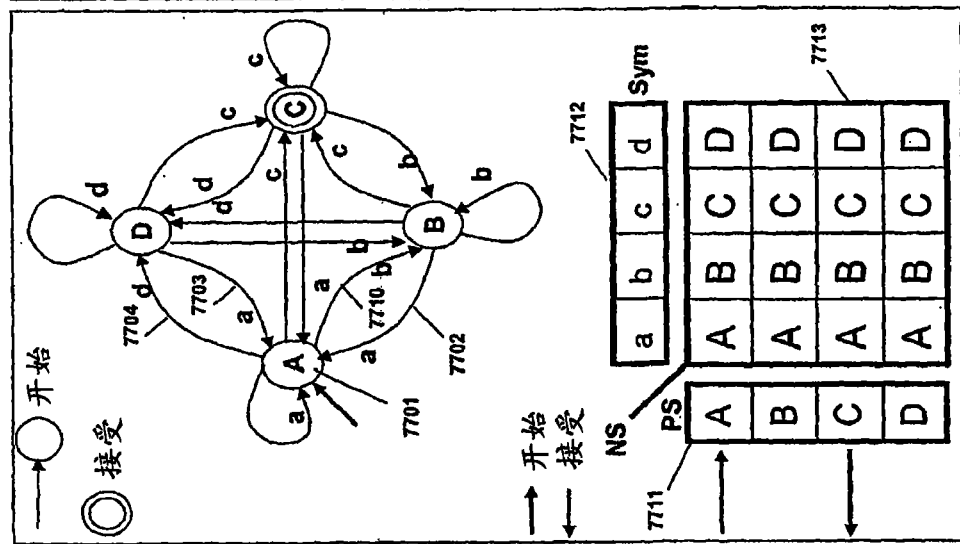
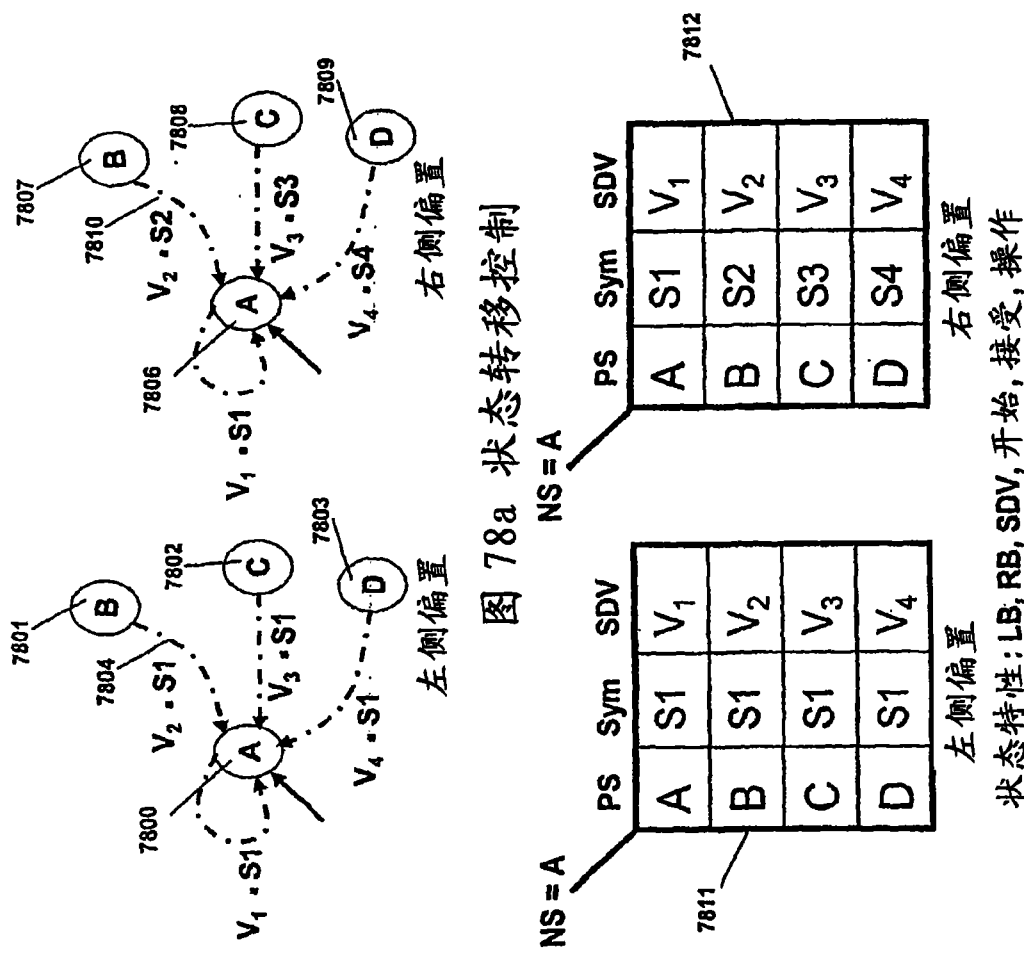


图 77a



NS = A

PS	Sym	SDV
A	S1	V ₁
B	S2	V ₂
C	S3	V ₃
D	S4	V ₄

7811

左侧偏置

NS = A

PS	Sym	SDV
A	S1	V ₁
B	S2	V ₂
C	S3	V ₃
D	S4	V ₄

7812

右侧偏置

状态特性: LB, RB, SDV, 开始, 接受, 操作

图 78b 每个状态的可配置下一状态表

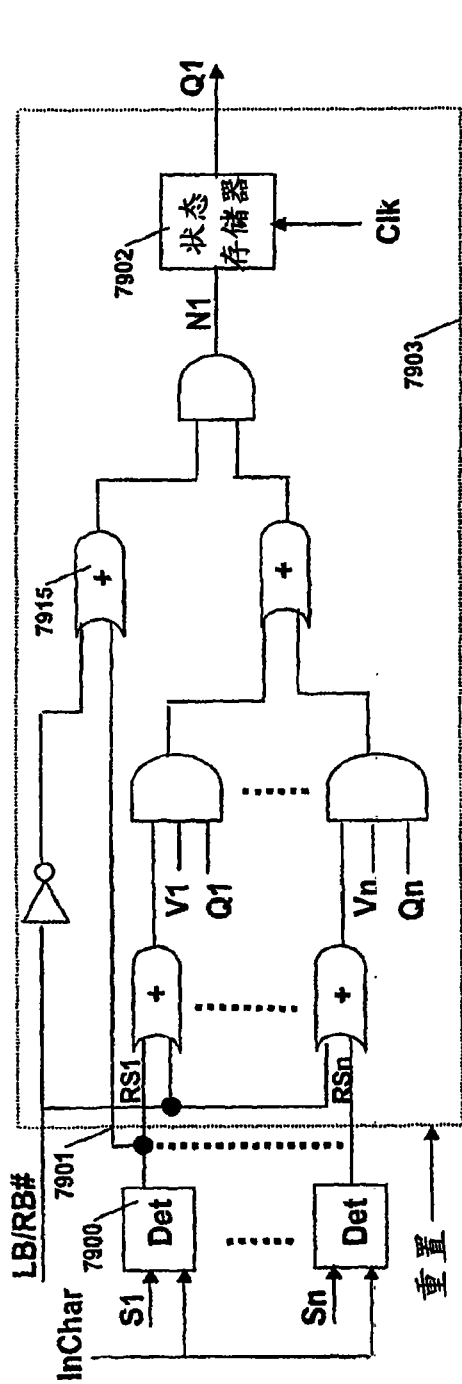


图 79a 状态的状态转移逻辑 (STL)

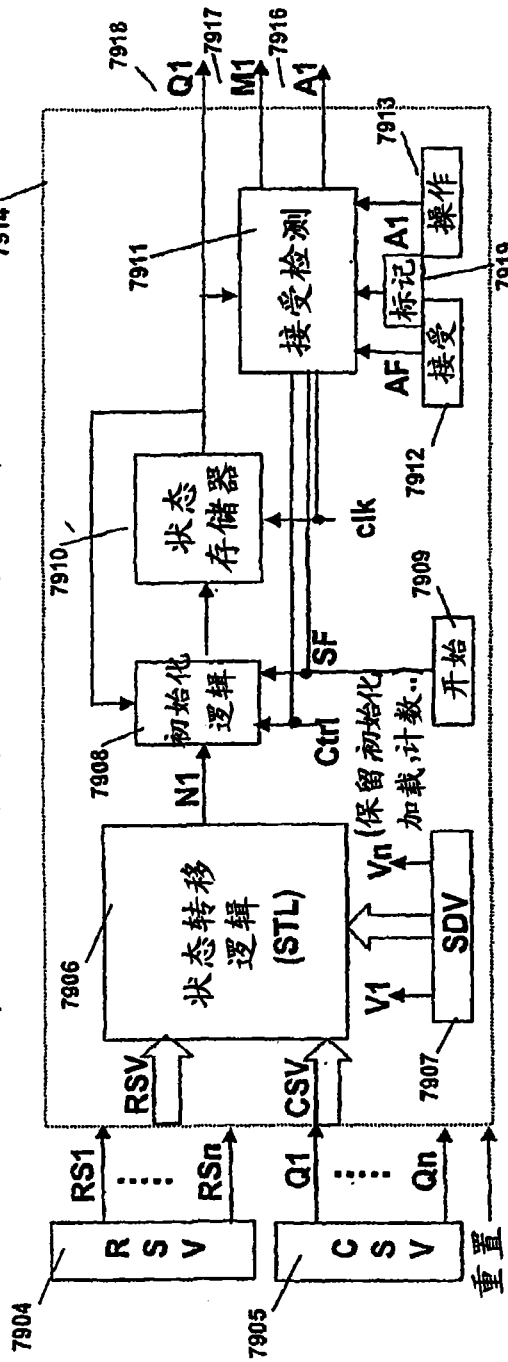
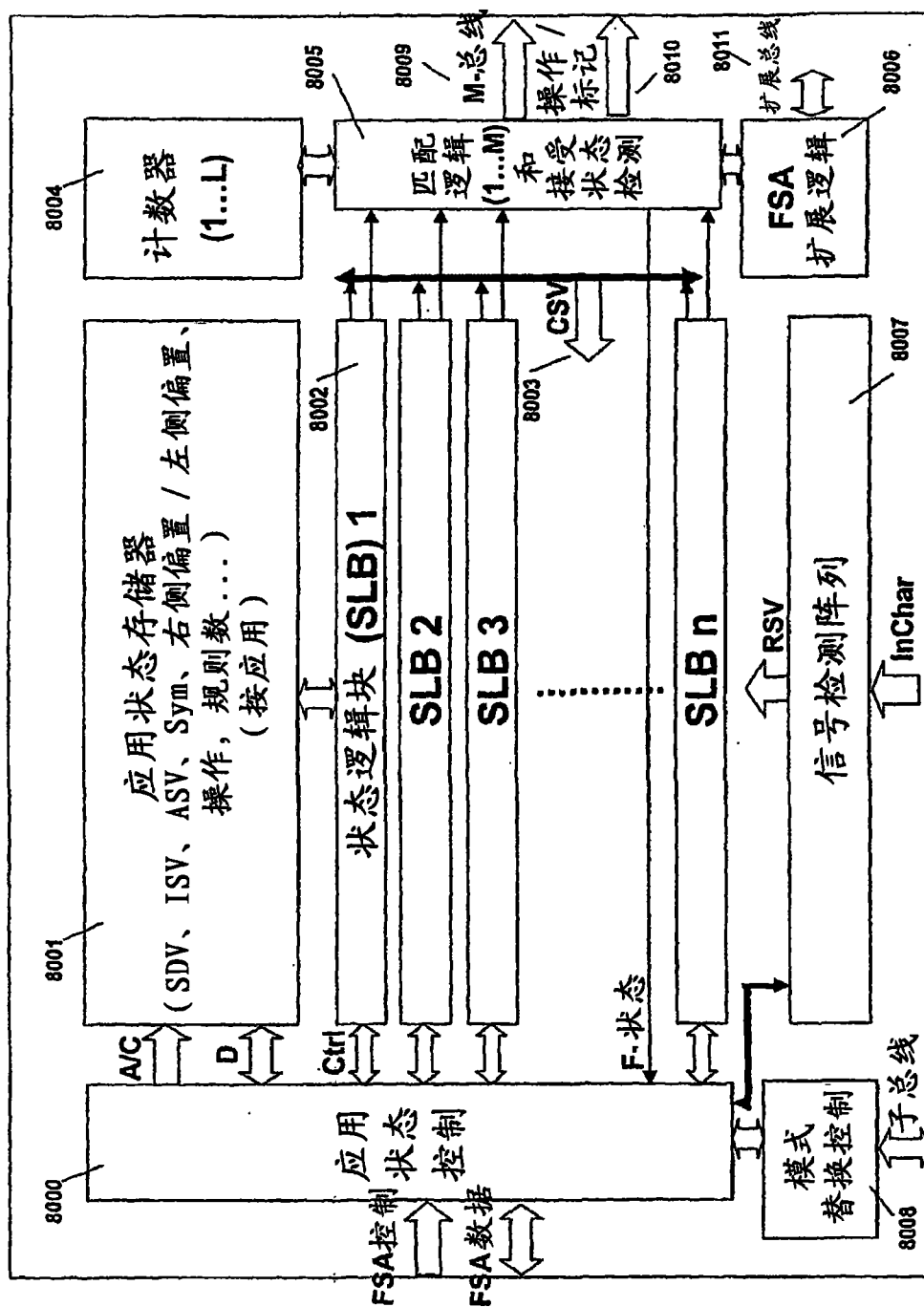


图 79b 状态逻辑块

基于NFA的搜索引擎



80

应用状态存储器实例配置

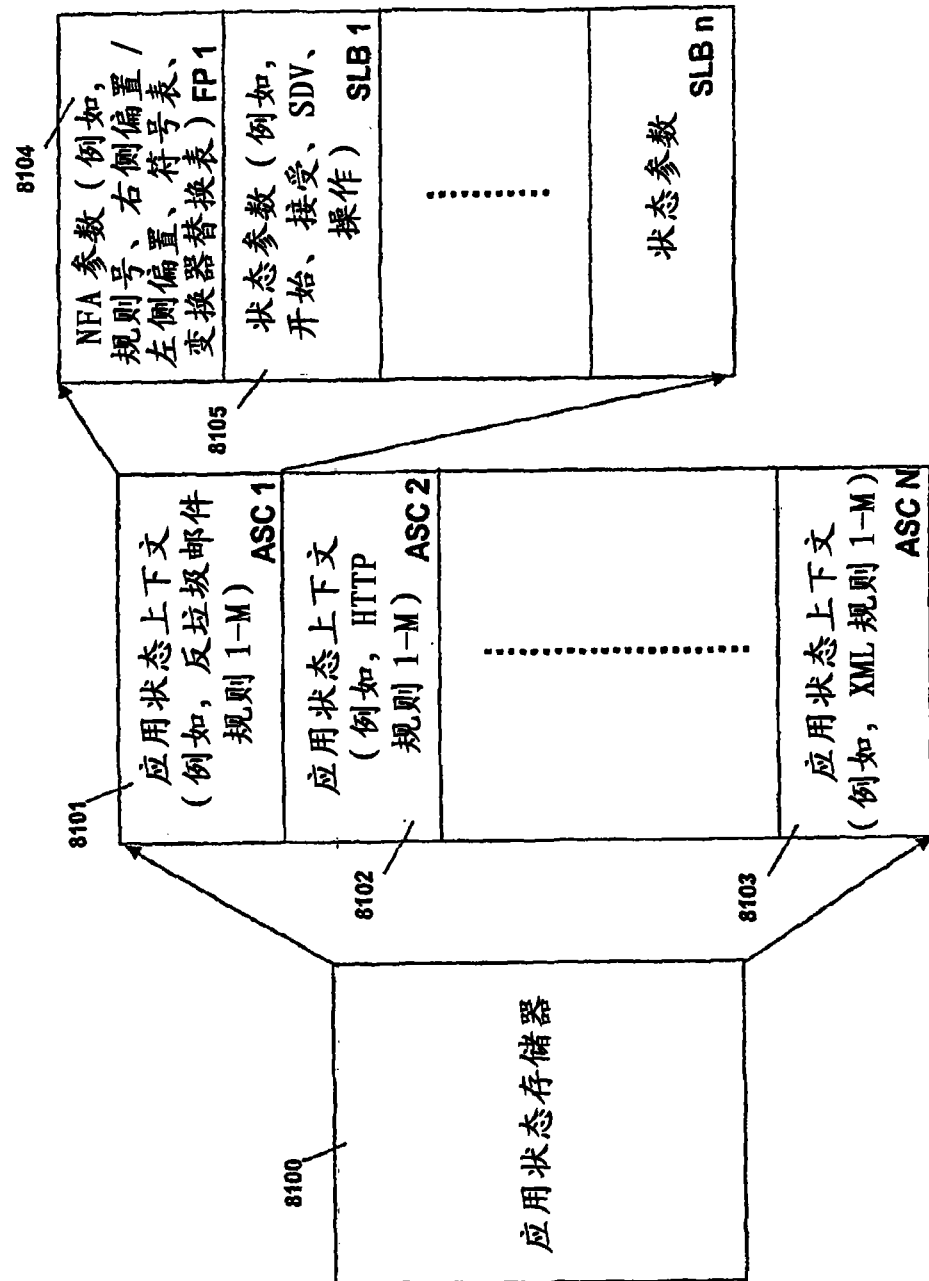


图 81

基于 DFA 的搜索引擎

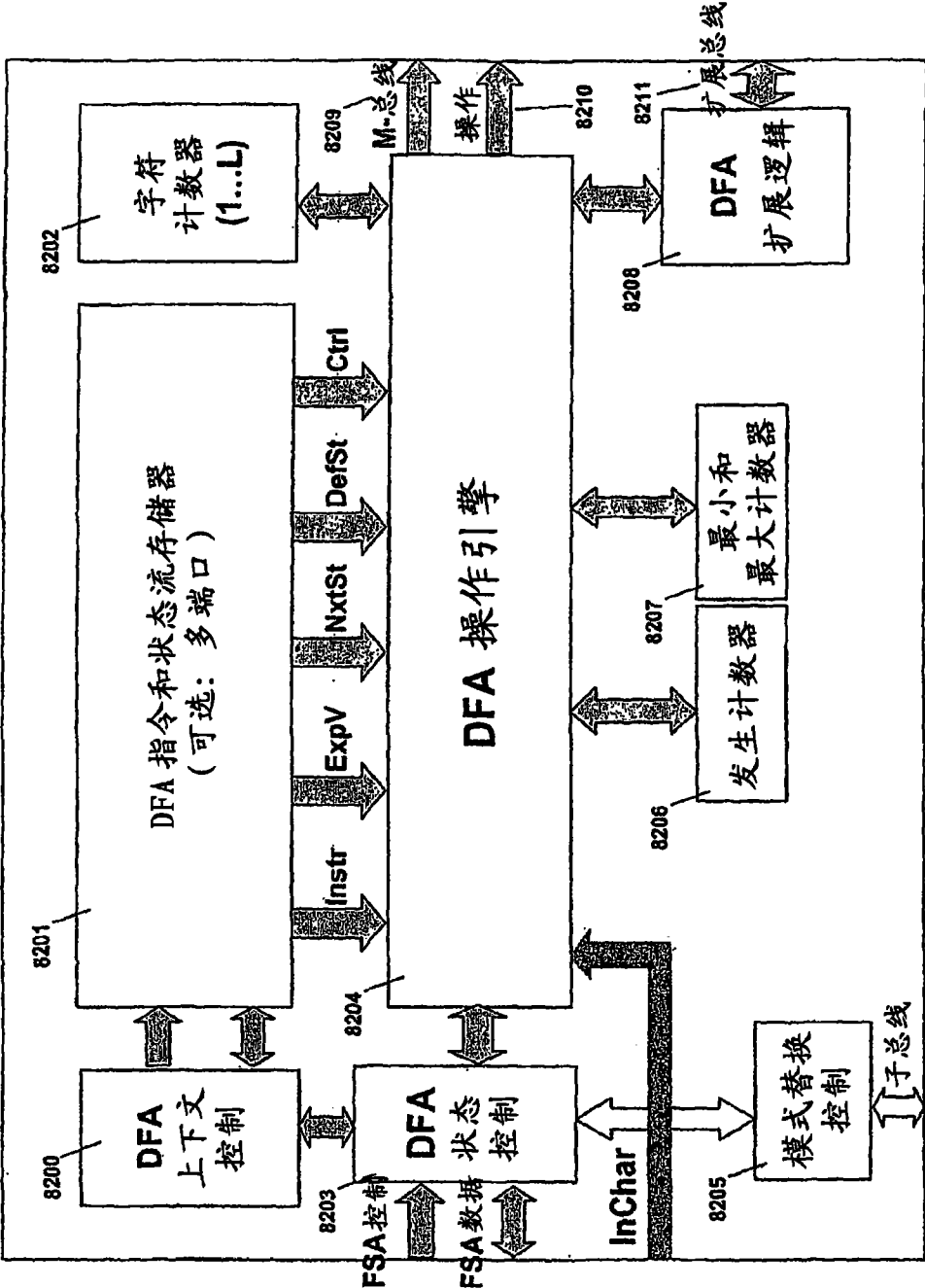
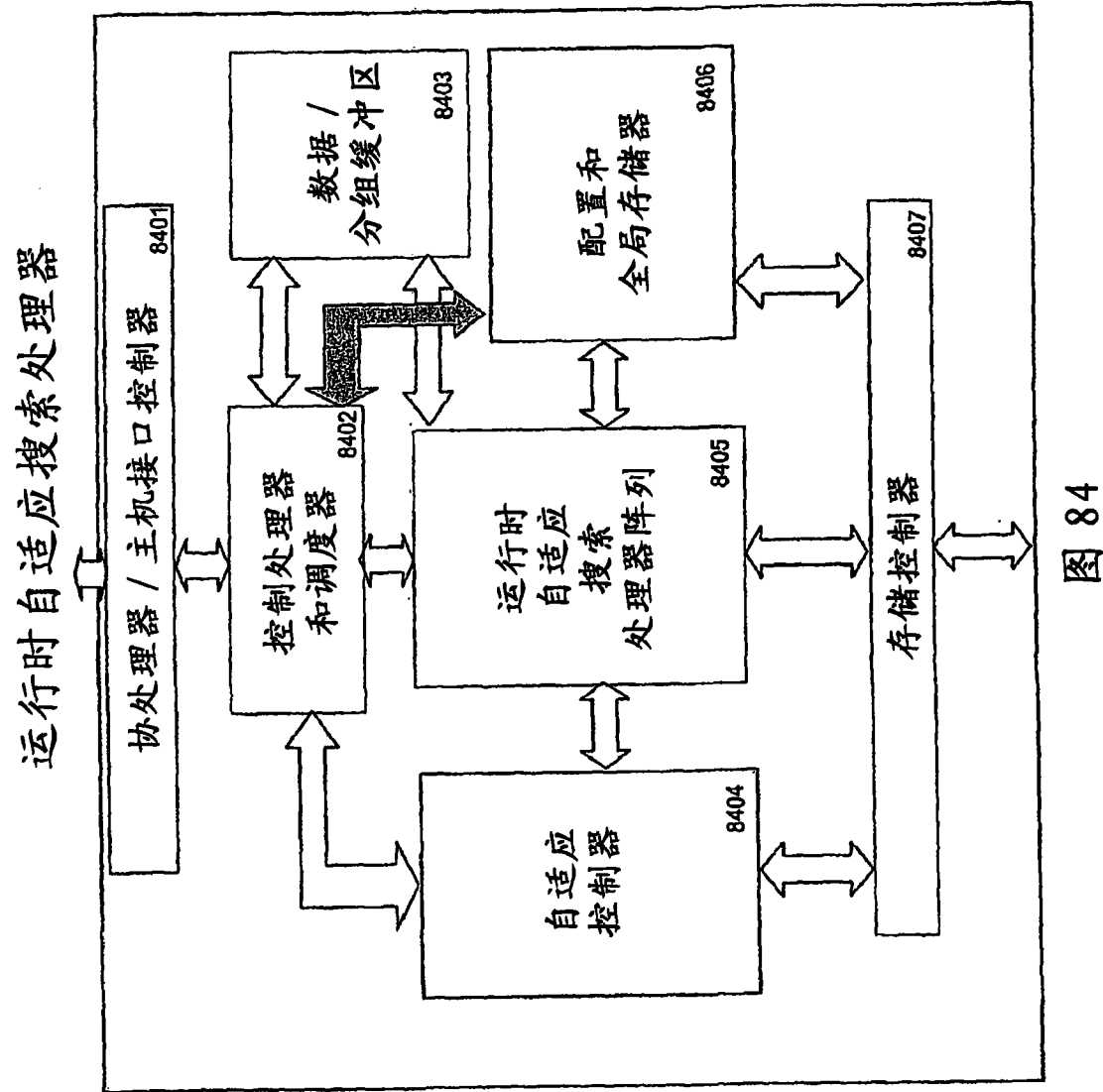


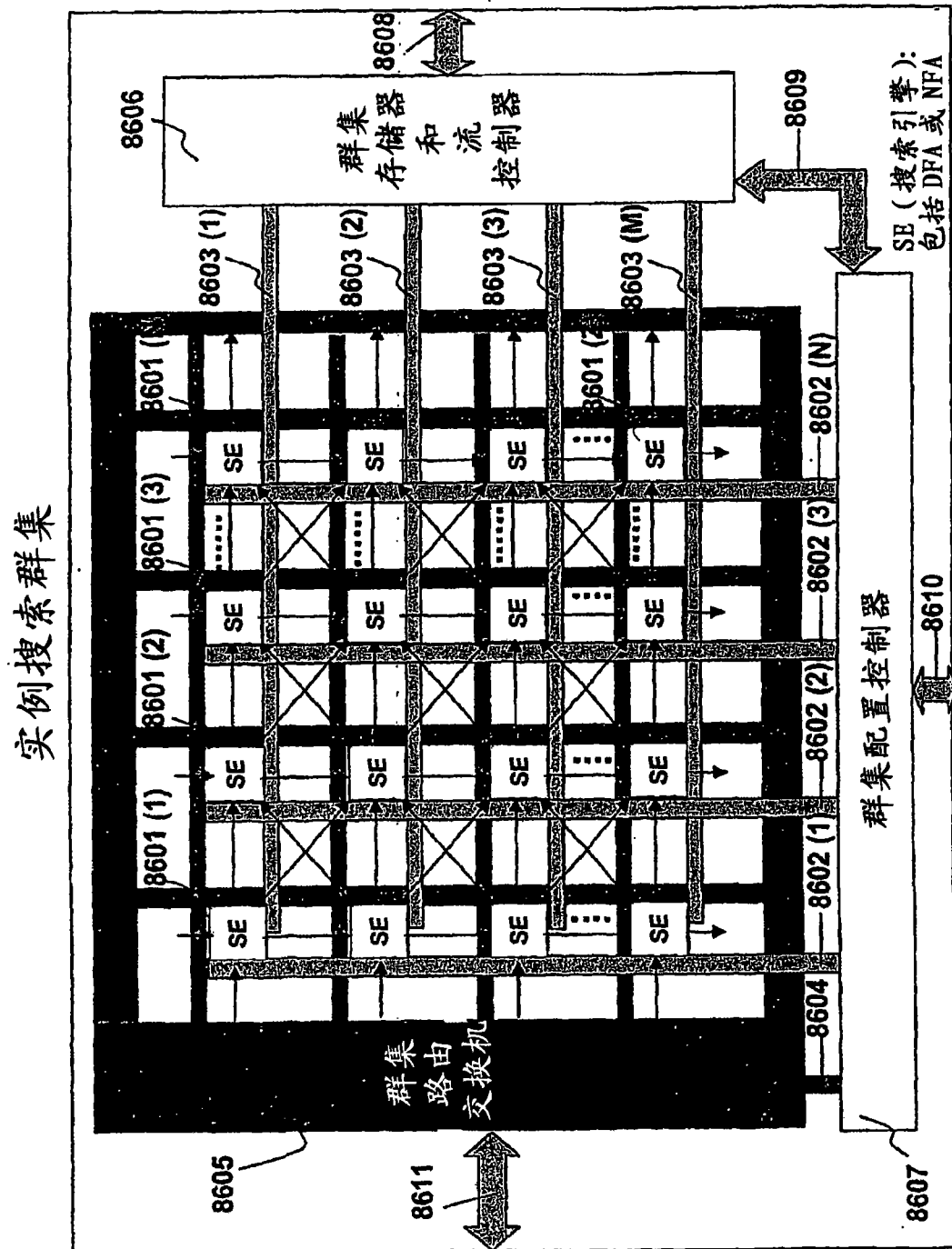
图 82

实例 DFA 操作

操作	等效的正则表达式	描述
EQ	X	在输入为 X 的情况下转到下一状态，否则转到备选状态
NEQ	~X	在输入不是 X 的情况下转到下一状态，否则转到备选状态
RANGE	X,Y [X-Y]	在输入处于 X 到 Y 的范围之内的情情况下转到下一状态，否则转到备选状态
LU_EQ	X *.X	在当前状态等待，直到首次出现 X，然后转到下一状态
LU_NEQ	X X+	一直等到序列中最后一次出现 X，然后转到下一状态
LU_MIN_EQ X	[n,] X	在当前状态中等待并检测 X 的最少 n 次发生，如果小于 n，则转到备选状态
LU_MIN_MAX_EQ X	[n,m] X	在当前状态中等待并检测 X 的 n 到 m 次发生，如果不在此范围内，则转到备选状态
EQ_CHAR	[a-zA-Z]	在输入处于字母范围的情况下转到下一状态，否则转到备选状态
EQ_DIGIT	[0-9]	在输入为十进制数字的情况下转到下一状态，否则转到备选状态
GEN_FLAG		生成操作或标记以及达到接受状态的标志

图 83





86 圖

内部搜索群集规则组和交换实例

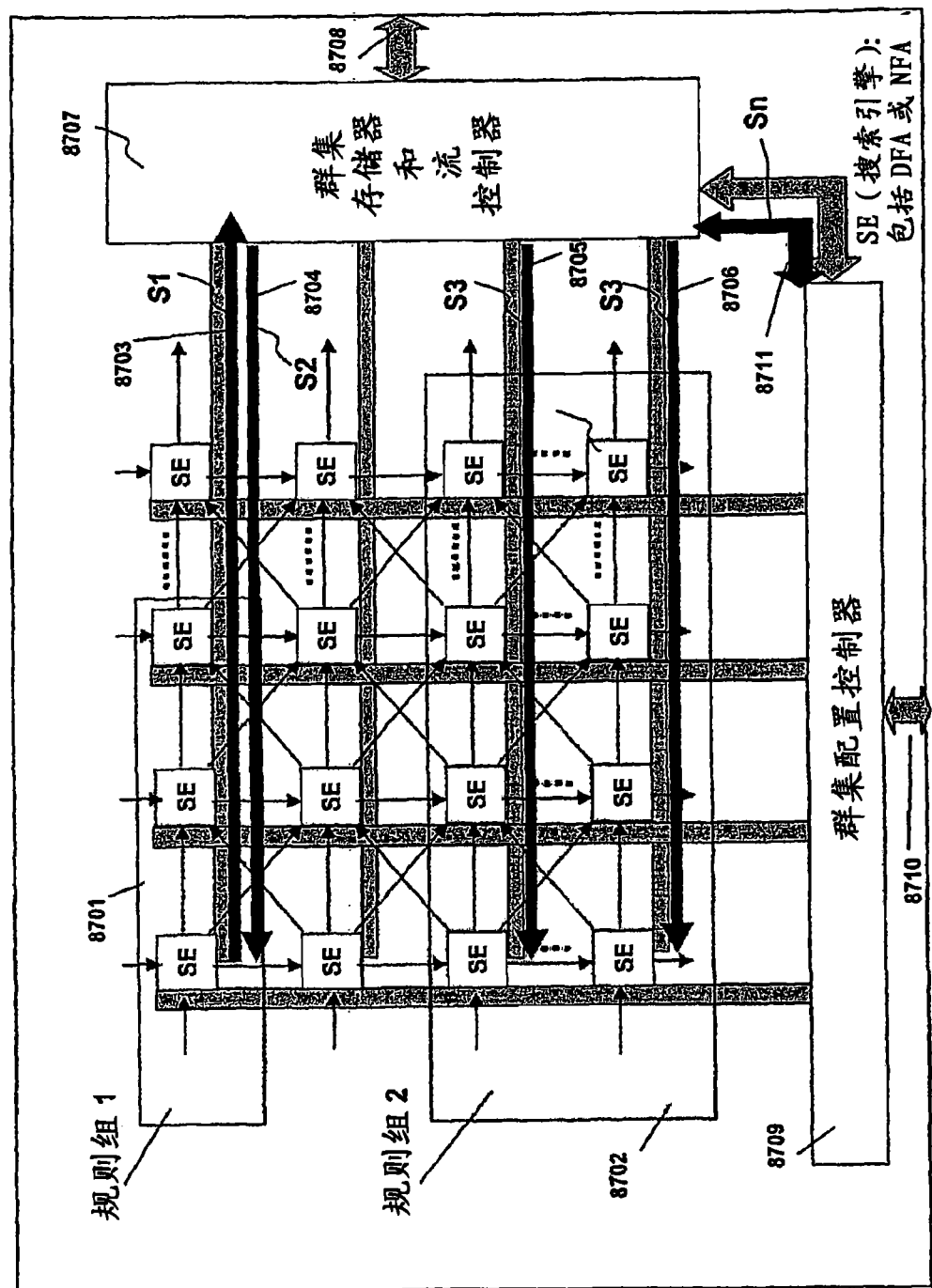
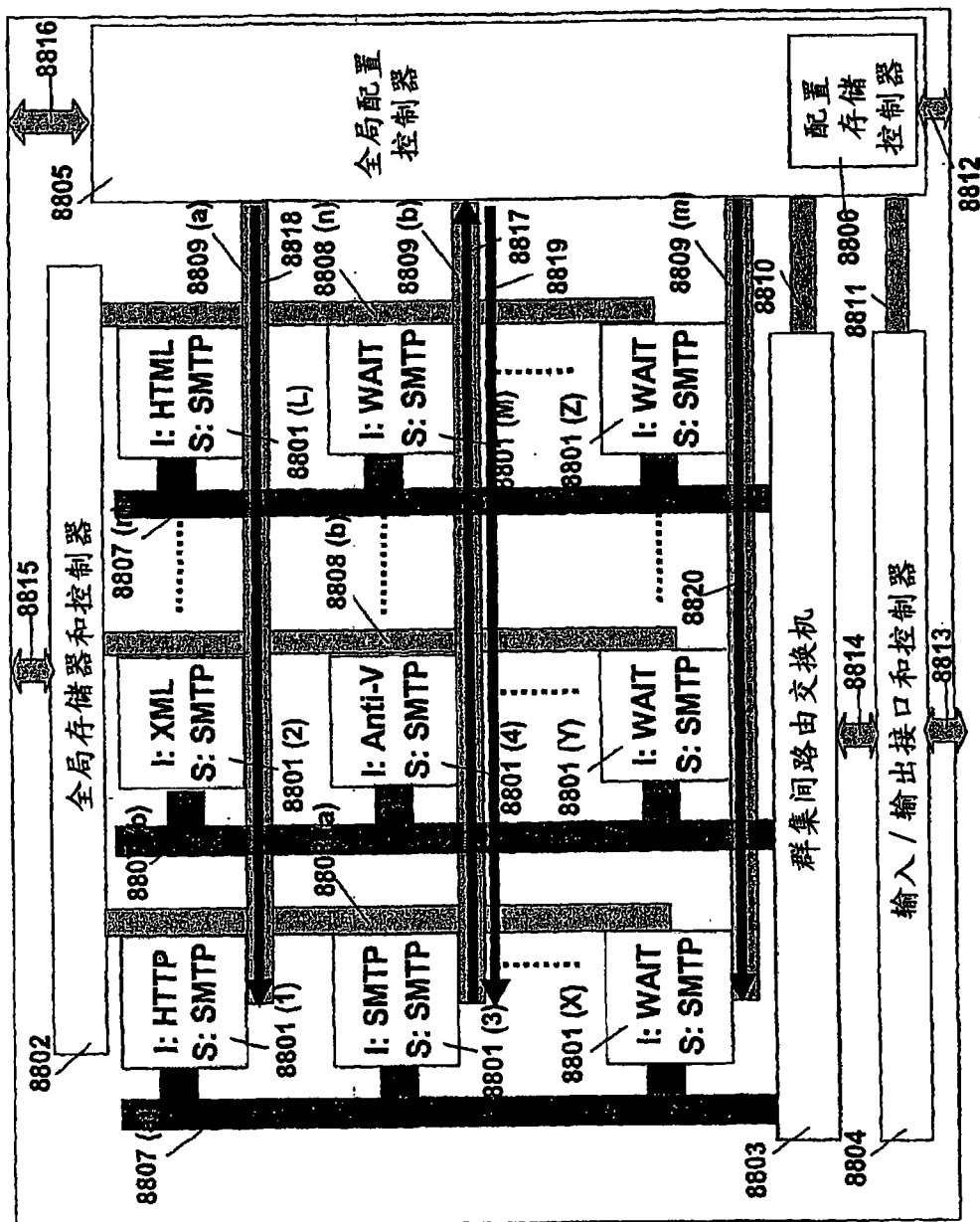


图 87

实例规则配置和上下文切换



88

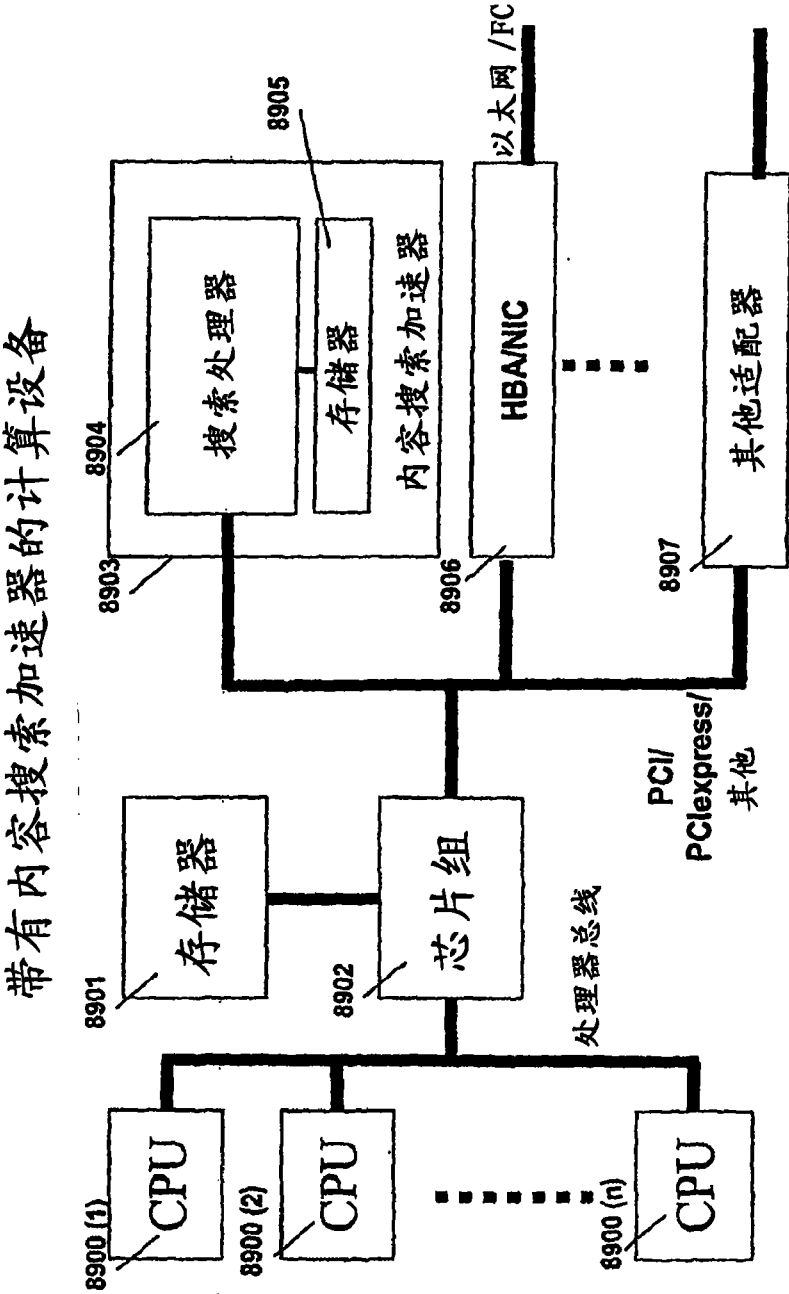


图 89

实例反垃圾邮件性能 * 瓶颈和解决方案

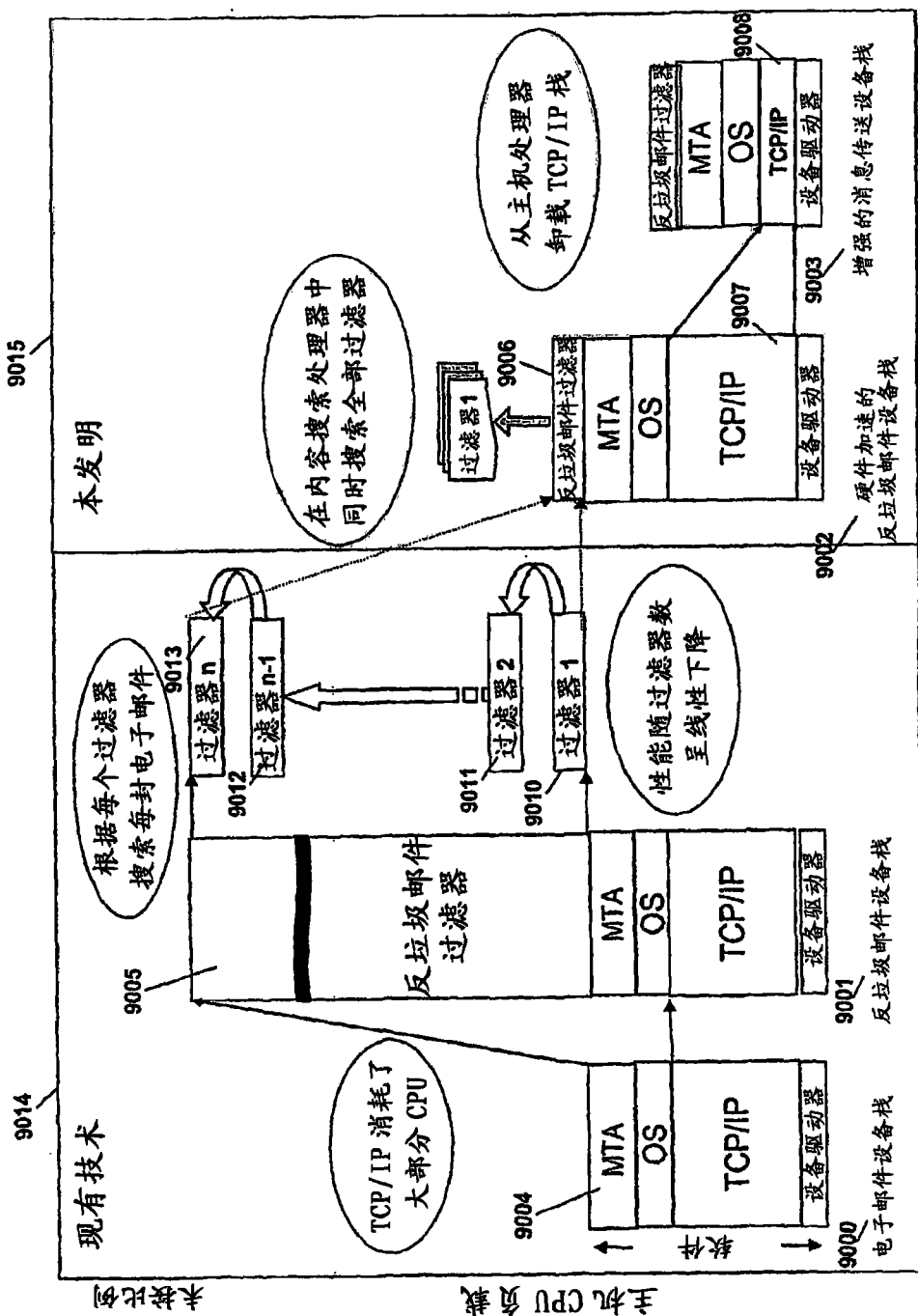


图 90

具有反病毒性能 * 瓶颈的反垃圾邮件

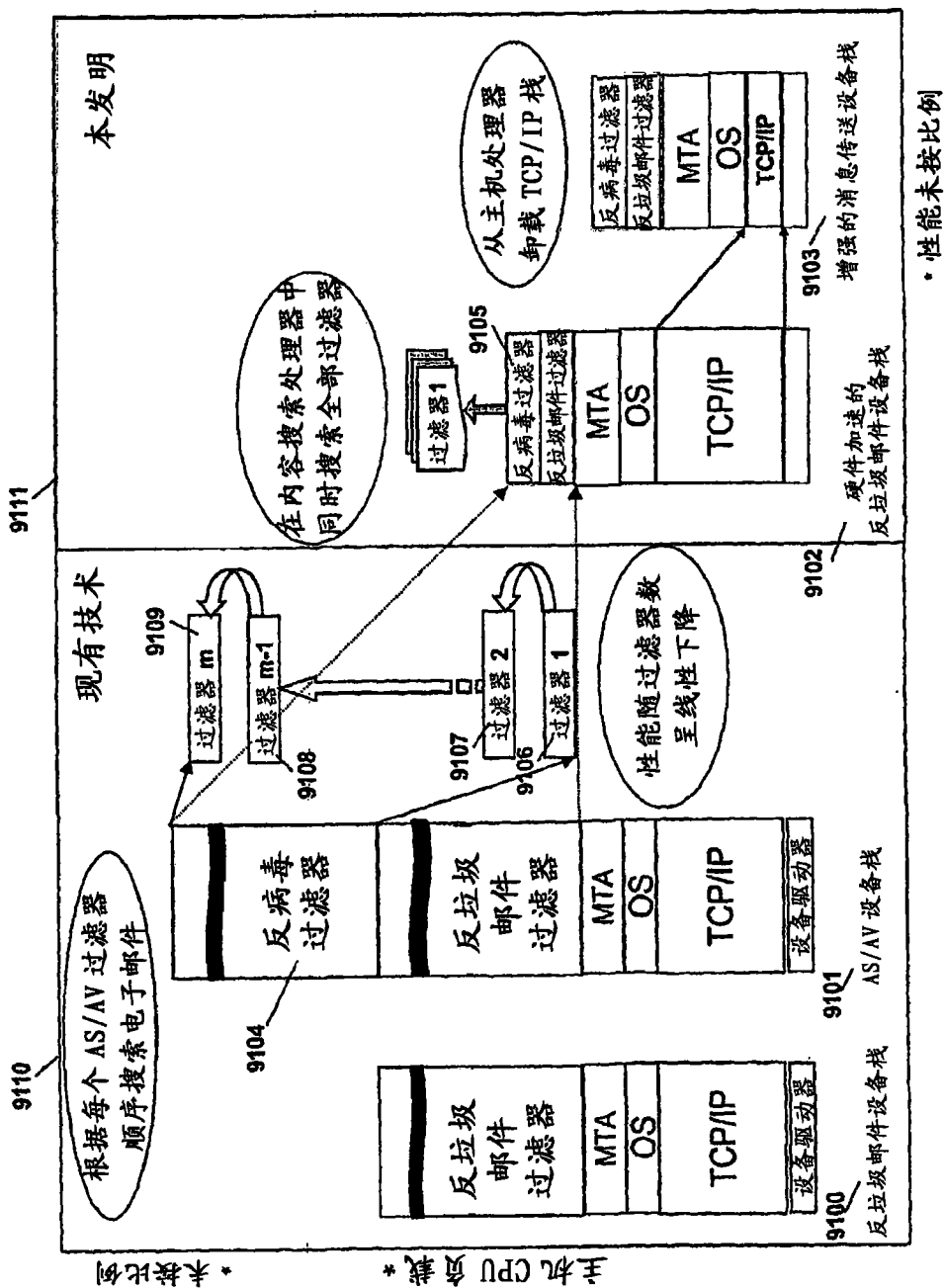


图 91

应用内容搜索性能 * 瓶颈和解决方案

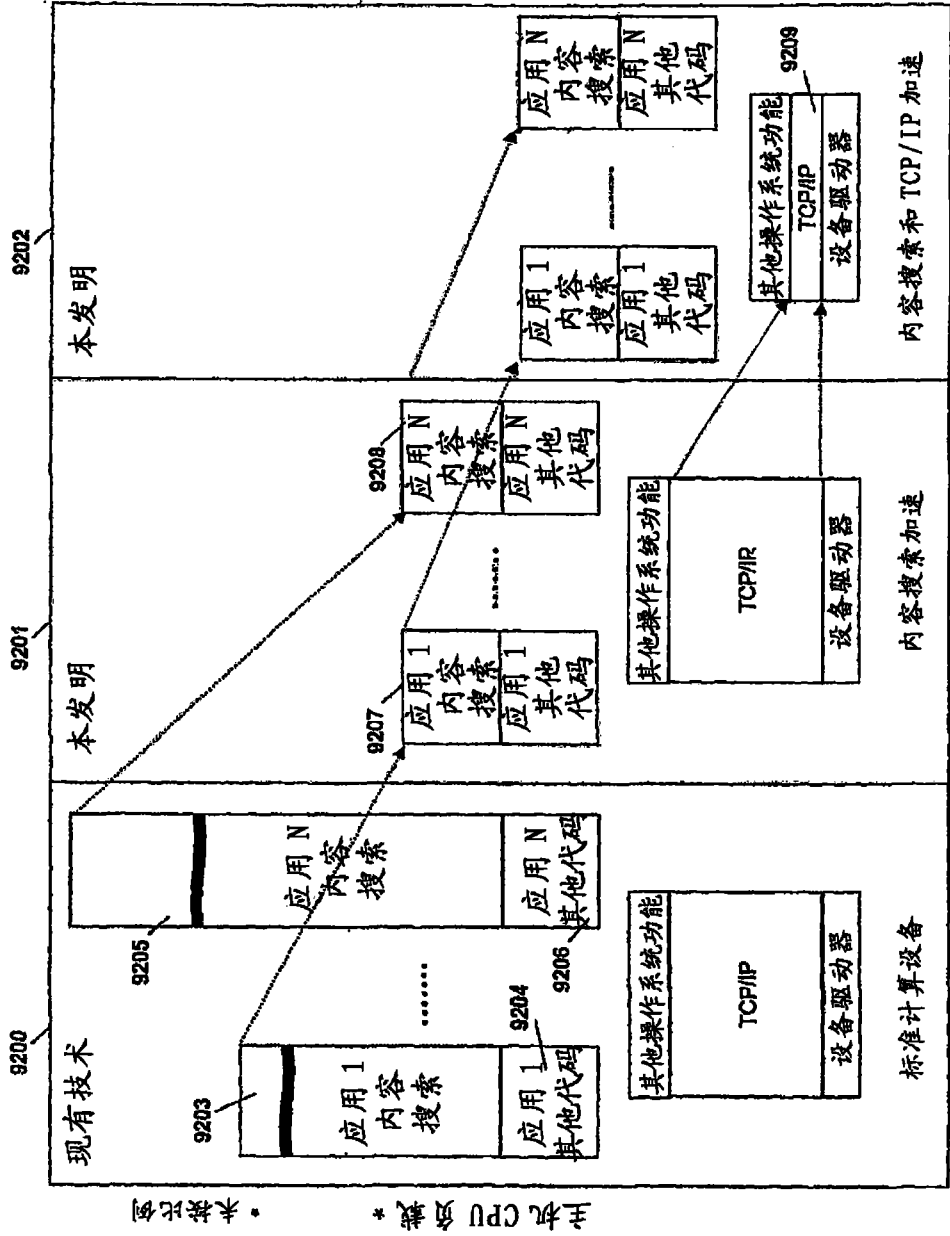


图 92

实例内容搜索 API 使用模型

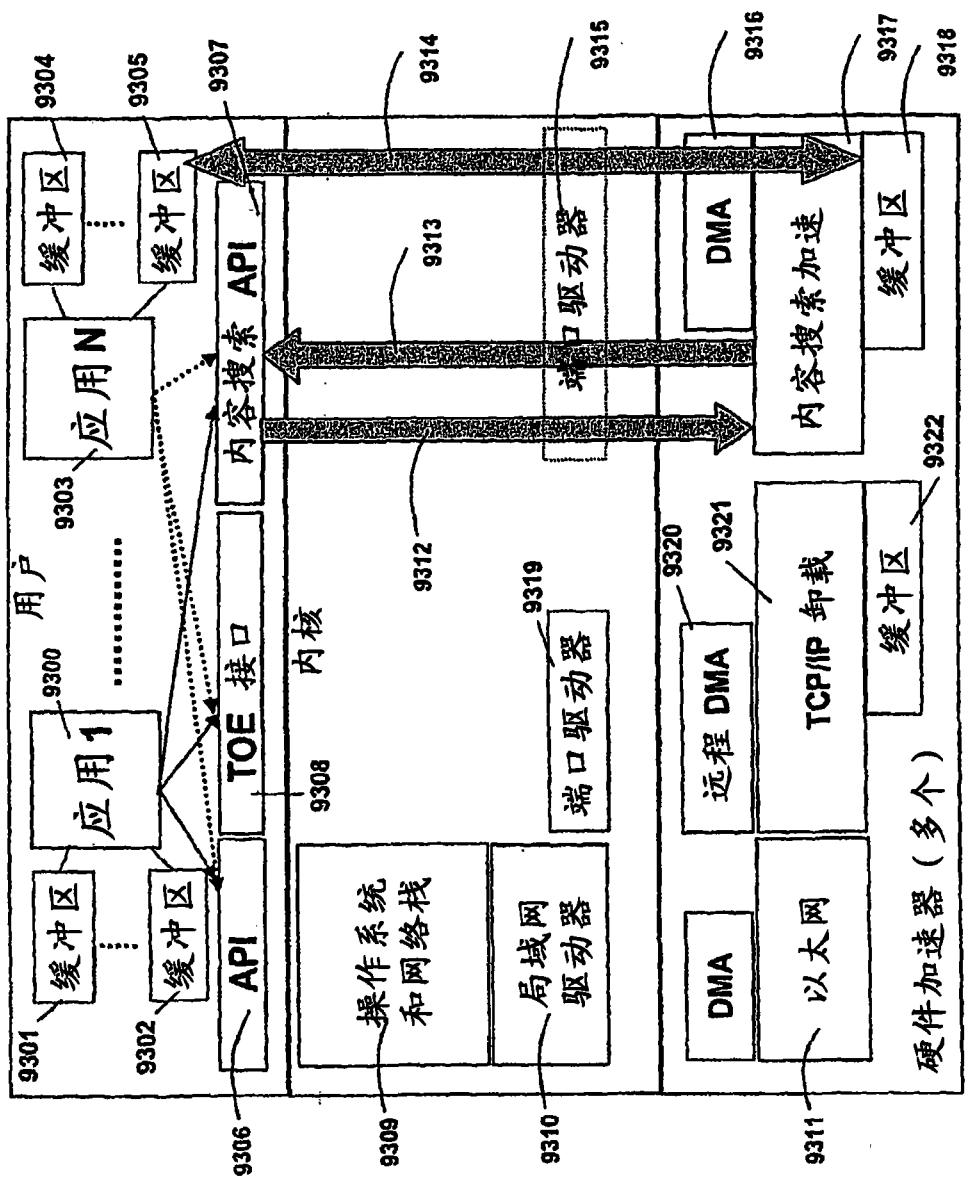


图 93

带有示例功能的实例内容搜索 API

API *	描述
("参数未示出")	
Initialize()	初始化硬件
getCapabilities()	查询现有硬件的能力
setDFARule()	在硬件处理器中设置 DFA 规则
setNFARule()	在硬件处理器中设置 NFA 规则
setContext()	设置搜索引擎 (多个) 的上下文
sndDataPtr()	发送需要搜索的数据缓冲区的指针
getRes()	从硬件处理器中检索结果
getResPtr()	检索指向保存结果的存储器缓冲区的指针
switchContext()	切换 DFA/NFA 引擎的当前活动上下文
setRuleGroup()	设置特定规则所属的规则组
selectRuleGroup()	选择特定的规则组进行访问
setInitGroup()	设置对于给定流活动的初始组
startHWSearch()	启动硬件搜索引擎
stopHWSearch()	停止硬件搜索引擎

图 94

使用搜索处理器的实例应用流 (静态设置)

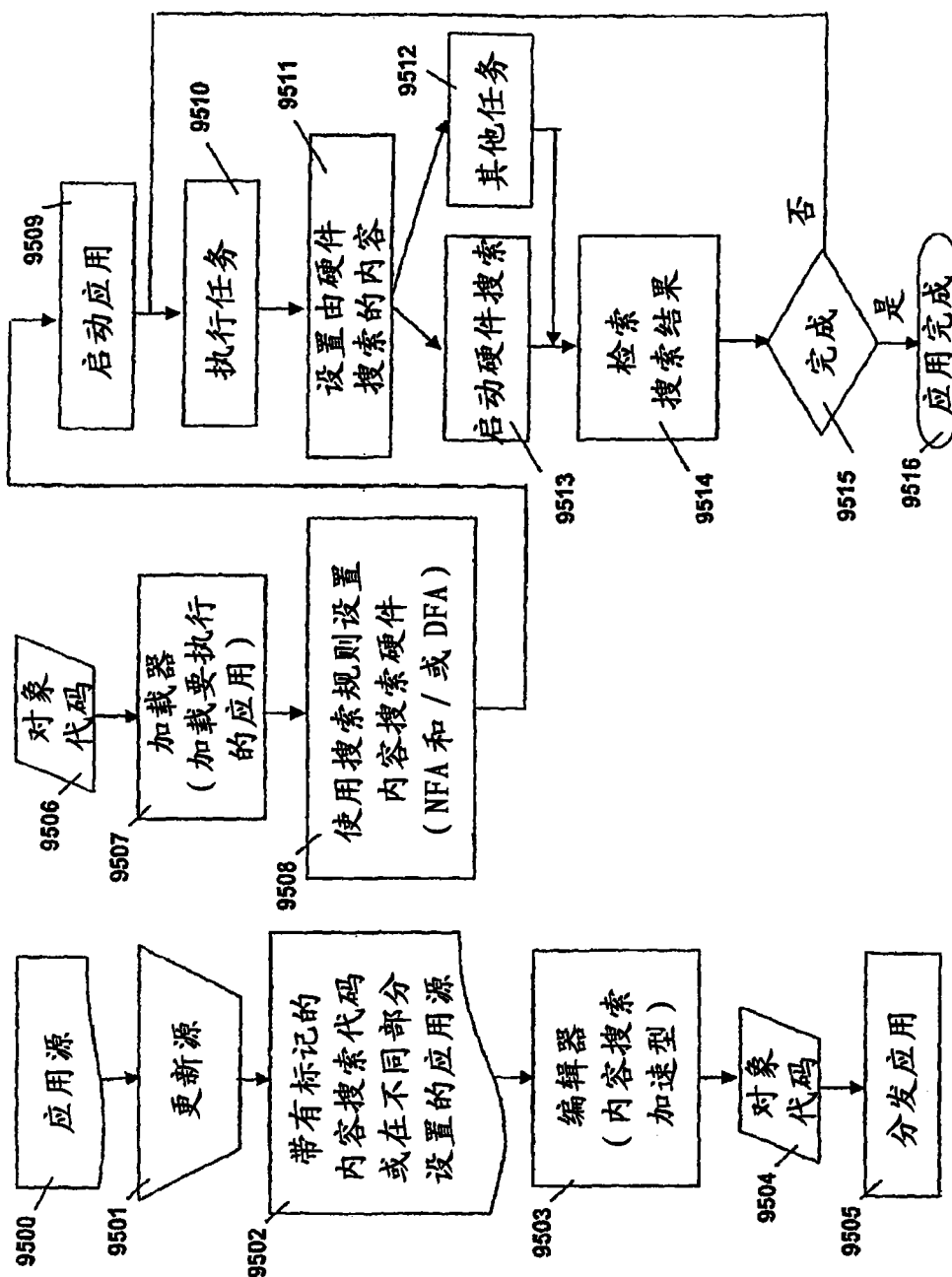


图 95

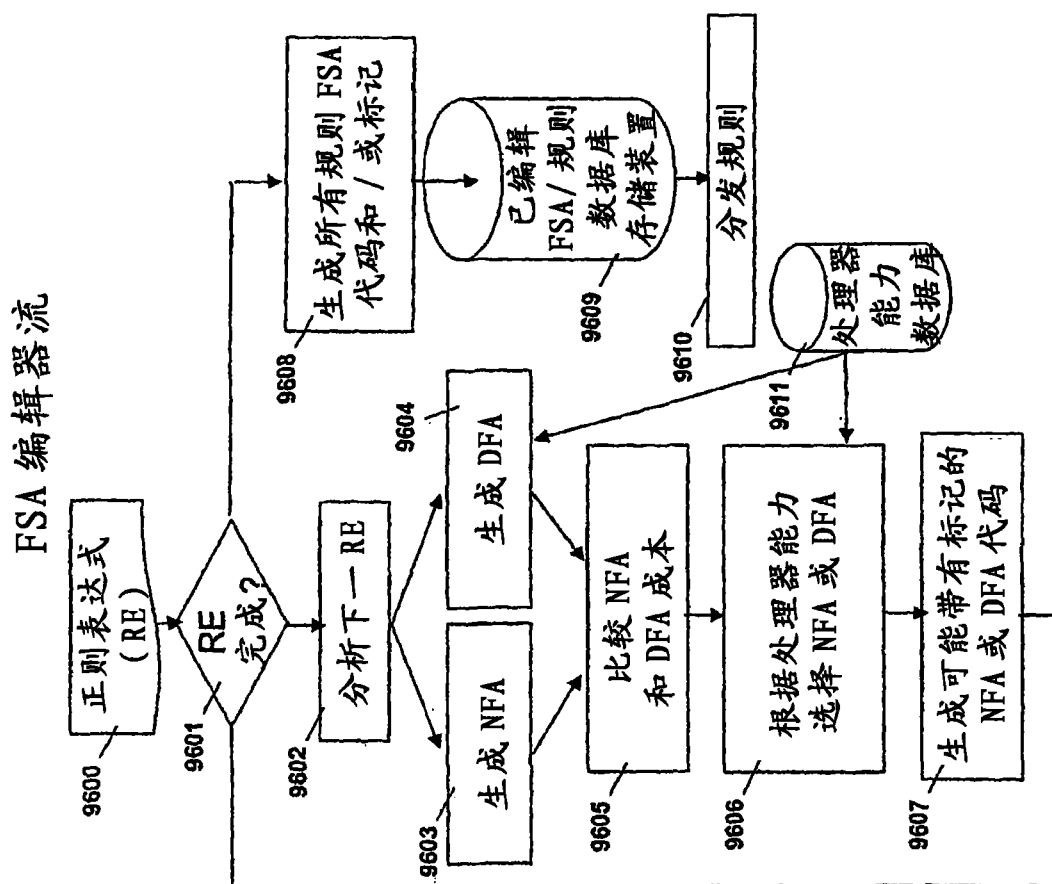


图 96

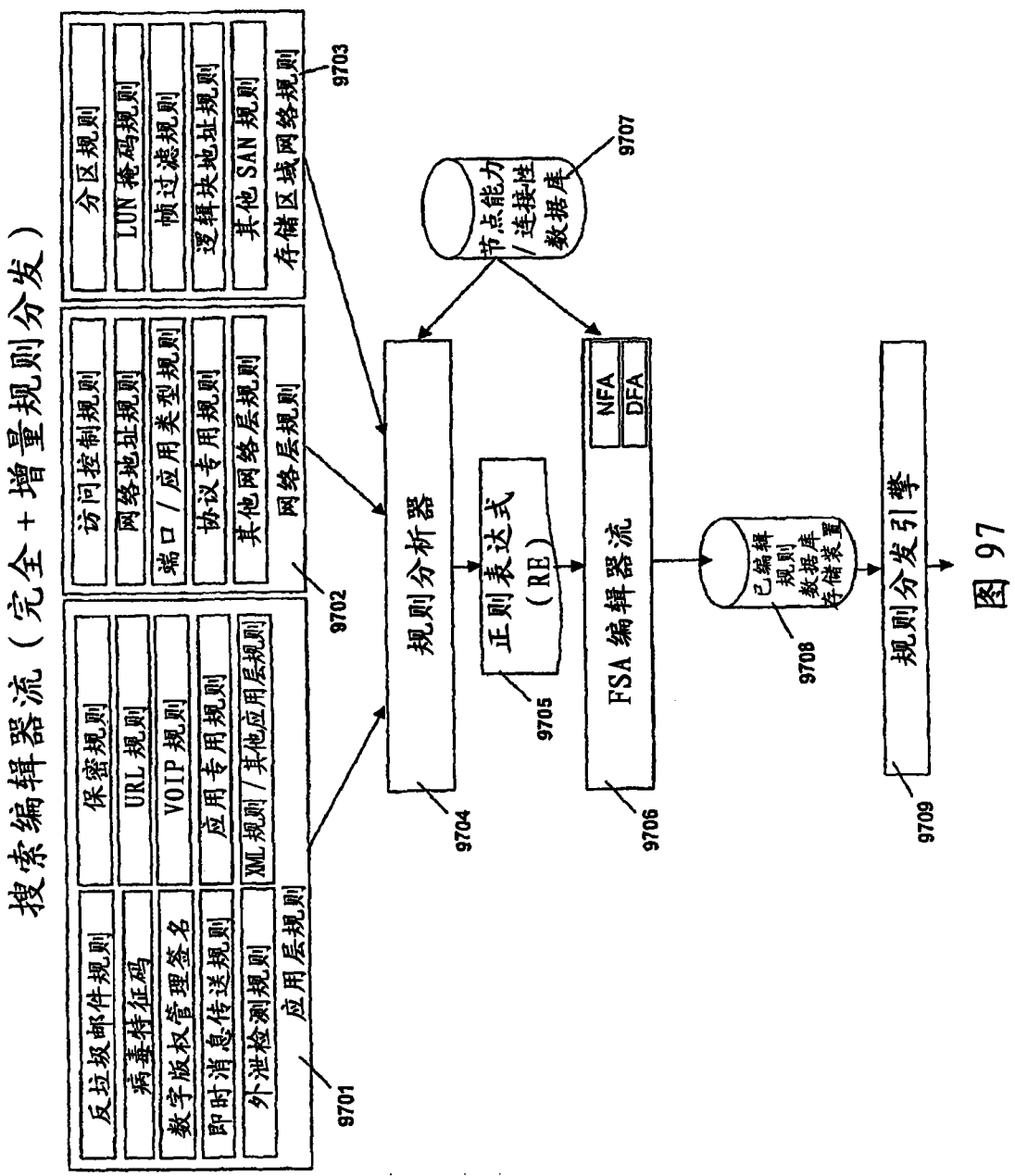


图 97

用于FPGA的FSA合成与编辑器流

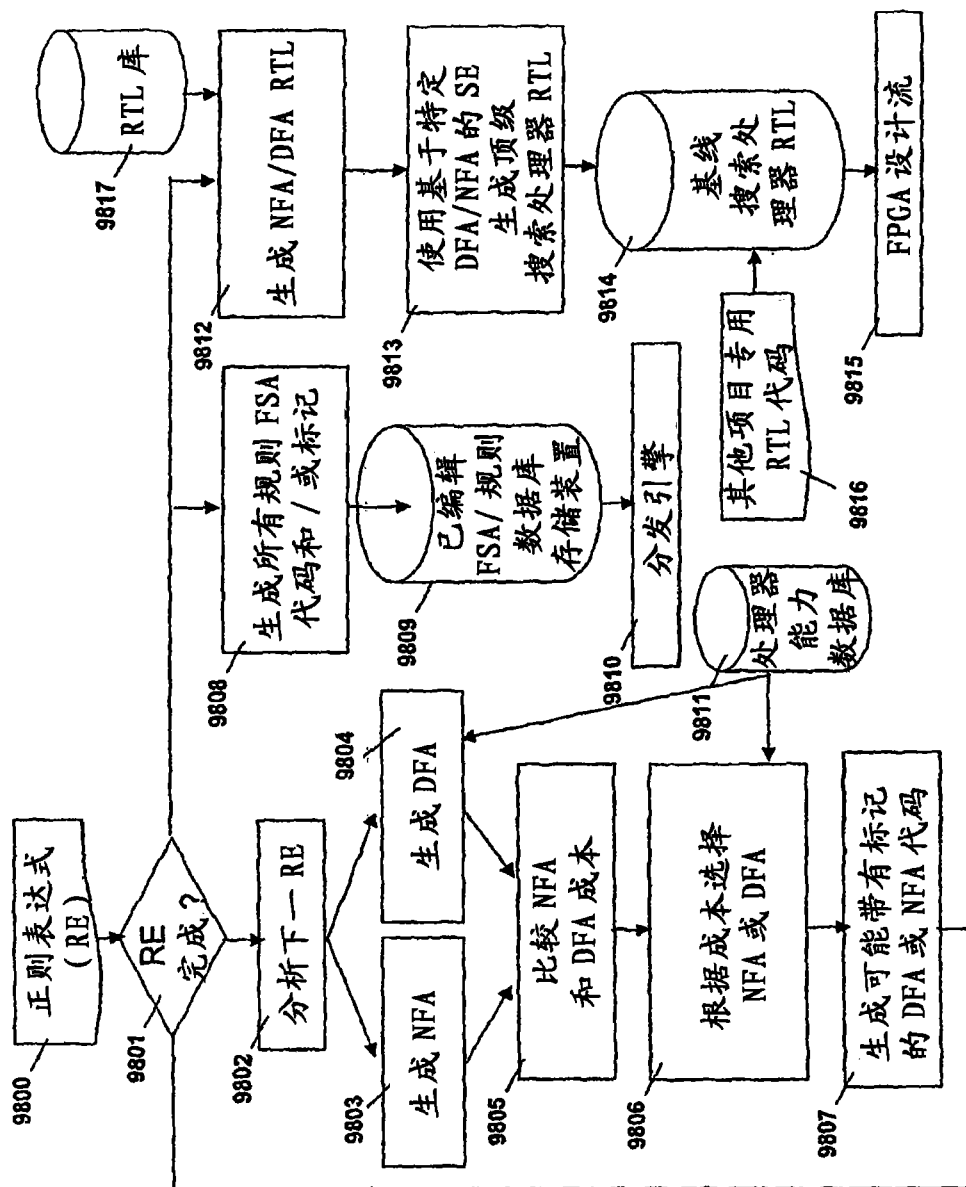


图 98