

网络数据包的服务质量处理方法、装置、设备和可读介质

本公开要求于 2022 年 04 月 29 日提交的申请号为 202210494011.4、名称为“网络数据包的服务质量处理方法、装置、设备和可读介质”的中国专利申请的优先权，该中国专利申请的5 全部内容通过引用全部并入本文。

技术领域

本公开涉及网络技术领域，具体而言，涉及一种网络数据包的服务质量处理方法、装置、设备和可读介质。

背景技术

目前，如果用户想设置某个连接的服务类型(tos)，除了在代码中调用 setsockopt（套接口设置）函数之外，主要通过 iptables 的 Mangle 表的 tos 参数进行设置，其中，iptables 是与最新的 3.5 版本 Linux 内核集成的 IP 信息包过滤系统。

15 在相关技术中，在代码中调用 setsockopt 固然可以设置服务类型字段值，但是一旦编译成二进制之后，服务类型字段值便无法动态修改，如果要修改服务类型字段值，必须重新改源代码，重新编译才能实现。

Mangle 表对数据包进行重新封装的操作包括修改服务类型(tos)和 TTL(Time To Live, 生存时间)等字段，通过 iptables 设置服务类型字段值是在每个发出的包经过 Mangle 表时，动态地修改 ip 头中的 tos 字段来实现，这就需要符合修改要求的每个网络数据包的服务类型字段值进行修改并重新封装，之后才能进行发送。当网络数据包的流量很大时，20 设置服务类型字段值的开销也会相应地增加，给网络系统带来一定的负担。

需要说明的是，在上述背景技术部分公开的信息仅用于加强对本公开的背景的理解，因此可以包括不构成对本领域普通技术人员已知的现有技术的信息。

发明内容

根据本公开的目的在于提供一种网络数据包的服务质量处理方法、装置、设备和可读介质，用于至少在一定程度上克服由于相关技术的限制和缺陷而导致的服务类型字段值更新的网络开销大的问题。

30 根据本公开实施例的第一方面，提供一种网络数据包的服务质量处理方法，包括：调用数据包过滤模块获取套接字对应的服务类型字段值；将所述服务类型字段值缓存至所述套接字的元数据中；基于缓存的服务类型字段值对路由后的网络数据包进行封包处理。

在本公开的一种示例性实施例中，在调用数据包过滤模块获取套接字对应的服务类型字段值前，还包括：在内核态加入所述数据包过滤模块，所述数据包过滤模块用于对用户态的服务类型字段图表进行更新；在所述内核态的协议栈的网络数据包的服务质量处理进35

程中加入所述数据包过滤模块的调用点,所述套接字在首次路由后在所述调用点调用所述数据包过滤模块,以获取所述服务类型字段值并缓存至所述套接字的元数据中;其中,所述服务类型字段图表中包括所述网络数据包的四元组与所述服务类型字段值之间的对应关系。

5 在本公开的一种示例性实施例中,还包括:触发所述用户态下所述网络数据包对应的配置文件或命令行参数确定所述对应关系;将所述对应关系写入所述服务类型字段图表。

在本公开的一种示例性实施例中,还包括:确定封包后的所述网络数据包的服务类型字段值;根据所述服务类型字段值对封包后的所述网络数据包进行分类控制处理。

在本公开的一种示例性实施例中,所述数据包过滤模块包括 eBPF 模块。

10 在本公开的一种示例性实施例中,所述服务类型字段值包括 tos 值。

在本公开的一种示例性实施例中,所述四元组为所述网络数据包对应的源 IP 地址、源端口、目标 IP 地址和目标端口。

根据本公开实施例的第二方面,提供一种网络数据包的服务质量处理装置,包括:调用模块,设置为调用数据包过滤模块获取套接字对应的服务类型字段值;缓存模块,设置为将所述服务类型字段值缓存至所述套接字的元数据中;封包模块,设置为基于缓存的服务类型字段值对路由后的网络数据包进行封包处理。

根据本公开的第三方面,提供一种电子设备,包括:存储器;以及耦合到所述存储器的处理器,所述处理器被配置为基于存储在所述存储器中的指令,执行如上述任意一项所述的方法。

20 根据本公开的第四方面,提供一种计算机可读存储介质,其上存储有程序,该程序被处理器执行时实现如上述任意一项所述的网络数据包的服务质量处理方法。

本公开实施例,通过调用数据包过滤模块获取套接字对应的服务类型字段值,并将所述服务类型字段值缓存至所述套接字的元数据中,进而基于缓存的服务类型字段值对路由后的网络数据包进行封包处理,提高了服务类型字段值更新的效率和灵活性,减少了服务类型字段值更新所需的开销,降低了网络数据包的交互压力。

应当理解的是,以上的一般描述和后文的细节描述仅是示例性和解释性的,并不能限制本公开。

附图说明

30 图 1 示出了可以应用本公开实施例的网络数据包的服务质量处理方案的示例性系统架构的示意图;

图 2 是本公开示例性实施例中一种网络数据包的服务质量处理方法的流程图;

图 3 是本公开示例性实施例中另一种网络数据包的服务质量处理方法的流程图;

图 4 是本公开示例性实施例中另一种网络数据包的服务质量处理方法的流程图;

35 图 5 是本公开示例性实施例中另一种网络数据包的服务质量处理方法的流程图;

图 6 是本公开示例性实施例中另一种网络数据包的服务质量处理方案的交互示意图；

图 7 是本公开示例性实施例中一种网络数据包的服务质量处理装置的方框图；

图 8 是本公开示例性实施例中一种电子设备的方框图。

5 具体实施方式

现在将参考附图更全面地描述示例实施方式。然而，示例实施方式能够以多种形式实施，且不应被理解为限于在此阐述的范例；相反，提供这些实施方式使得本公开将更加全面和完整，并将示例实施方式的构思全面地传达给本领域的技术人员。所描述的特征、结构或特性可以以任何合适的方式结合在一个或更多实施方式中。在下面的描述中，提供许多具体细节从而给出对本公开的实施方式的充分理解。然而，本领域技术人员将意识到，可以实践本公开的技术方案而省略所述特定细节中的一个或多个，或者可以采用其它的方法、组元、装置、步骤等。在其它情况下，不详细示出或描述公知技术方案以避免喧宾夺主而使得本公开的各方面变得模糊。

此外，附图仅为本公开的示意性图解，图中相同的附图标记表示相同或类似的部分，因而将省略对它们的重复描述。附图中所示的一些方框图是功能实体，不一定必须与物理或逻辑上独立的实体相对应。可以采用软件形式来实现这些功能实体，或在一个或多个硬件模块或集成电路中实现这些功能实体，或在不同网络和/或处理器装置和/或微控制器装置中实现这些功能实体。

图 1 示出了可以应用本发明实施例的网络数据包的服务质量处理方案的示例性系统架构的示意图。

如图 1 所示，系统架构 100 可以包括终端设备 101、102、103 中的一种或多种，网络 104 和服务 105。网络 104 用以在终端设备 101、102、103 和服务 105 之间提供通信链路的介质。网络 104 可以包括各种连接类型，例如有线、无线通信链路或者光纤电缆等等。

应该理解，图 1 中的终端设备、网络和服务器的数目仅仅是示意性的。根据实现需要，可以具有任意数目的终端设备、网络和服务。比如服务器 105 可以是多个服务器组成的服务器集群等。

用户可以使用终端设备 101、102、103 通过网络 104 与服务 105 交互，以接收或发送消息等。终端设备 101、102、103 可以是具有显示屏的各种电子设备，包括但不限于智能手机、平板电脑、便携式计算机和台式计算机等等。

在一些实施例中，本发明实施例所提供的网络数据包的服务质量处理方法一般由终端 105 执行，相应地，网络数据包的服务质量处理装置一般设置于终端设备 103（也可以是终端设备 101 或 102）中。在另一些实施例中，某些服务器可以具有与终端设备相似的功能从而执行本方法。因此，本发明实施例所提供的网络数据包的服务质量处理方法不限在终端设备执行。

下面结合附图对本公开示例实施方式进行详细说明。

图 2 是本公开示例性实施例中网络数据包的服务质量处理方法的流程图。

参考图 2，网络数据包的服务质量处理方法可以包括：

步骤 S202，调用数据包过滤模块获取套接字对应的服务类型字段值。

5 步骤 S204，将服务类型字段值缓存至套接字的元数据中。

步骤 S206，基于缓存的服务类型字段值对路由后的网络数据包进行封包处理。

本公开实施例，通过调用数据包过滤模块获取套接字对应的服务类型字段值，并将服务类型字段值缓存至套接字的元数据中，进而基于缓存的服务类型字段值对路由后的网络数据包进行封包处理，相较于相关技术中的按包设置而言，本公开的技术方案提高了实现了服务类型字段的按流设置，服务类型字段值更新的效率和灵活性，减少了服务类型字段值更新所需的开销，降低了网络数据包的交互压力。

下面，对网络数据包的服务质量处理方法的各步骤进行详细说明。

在本公开的一种示例性实施例中，如图 3 所示，在调用数据包过滤模块获取套接字对应的服务类型字段值前，还包括：

15 步骤 S302，在内核态加入数据包过滤模块，该数据包过滤模块用于对用户态的服务类型字段图表进行更新。

步骤 S304，在内核态的协议栈的网络数据包的服务质量处理进程中加入所述数据包过滤模块的调用点，套接字在首次路由后在该调用点调用数据包过滤模块，以获取服务类型字段值并缓存至套接字的元数据中；其中，服务类型字段图表中包括网络数据包的四元组与服务类型字段值之间的对应关系。

在本公开的一个实施例中，数据包过滤模块基于 eBPF 实现，服务类型字段基于 tos 实现，本公开的实施例只需要进行 1 次服务类型字段值设置，内核中在封包之前加入 ebpf 调用点，通过调用数据包过滤模块来根据 map 中的数据设置 ip 头中的 tos 字段。对于同一个 socket，该 tos 值会缓存到 socket 元数据中，除第一次设置之外，以后的服务类型字段值会自动使用该缓存值，从而实现服务类型字段值的按流设置。基于此，tos 值的获取逻辑可以随时更改，因为 eBPF 程序可以在不需要修改内核的情况下，进行单独更新，具有更大的灵活性。

在本公开的一种示例性实施例中，如图 4 所示，网络数据包的服务质量处理方法还包括：

30 步骤 S402，触发用户态下网络数据包对应的配置文件或命令行参数确定网络数据包的四元组与服务类型字段值之间的对应关系。

步骤 S404，将该对应关系写入服务类型字段图表。

在上述实施例中，用户态通过读取配置文件或者通过命令行参数，将四元组和服务类型字段值的对应关系通过系统调用写入服务类型字段图表中，需要支持增删查改等常见操作，也即用户态可以随时修改某个 socket 连接的服务类型字段值，并且即时生效，eBPFmap

中保存了四元组信息和相应的服务类型字段值，初次建立连接时，内核代码会调用 eBPF 程序查询 map，获取服务类型字段值后将该服务类型字段值写入 IP 头中。

在本公开的一种示例性实施例中，如图 5 所示，网络数据包的服务质量处理方法还包括：

5 步骤 S502，确定封包后的网络数据包的服务类型字段值。

步骤 S504，根据服务类型字段值对封包后的网络数据包进行分类控制处理。

在本公开的一种示例性实施例中，数据包过滤模块包括 eBPF 模块。

在上述实施例中，基于 eBPF 的 QoS (Quality of Service) 服务是用来对 IP 流量进行分类控制的主要手段，IP 通过在头部加入 TOS(type of service) 字段，来对不同服务需求
10 进行区分和监控。

在本公开的一种示例性实施例中，服务类型字段值包括 tos 值。

在本公开的一种示例性实施例中，四元组为网络数据包对应的源 IP 地址、源端口、目标 IP 地址和目标端口。

对应于上述方法实施例，本公开还提供一种网络数据包的服务质量处理装置，可以用于
15 于执行上述方法实施例。

如图 6 所示，Linux 内核一直是实现监视/可观察性、网络和安全性的理想场所。但这通常是不切实际的，因为它需要更改内核态 602 源代码或加载内核态 602 模块，并导致彼此堆叠的抽象层。eBPF (extended Berkeley Packet Filter，增强型伯克利包过滤器) 是一项革命性的技术，可以在 Linux 内核中运行沙盒程序，而无需更改内核态 602 源代码或加
20 载内核态 602 模块。通过使 Linux 内核可编程，基础架构软件可以利用现有的层，从而使它们更加智能和功能丰富，而无需继续为系统增加额外的复杂性层。

eBPF 技术导致了网络、安全性、应用程序配置/跟踪和性能故障排除等领域的新一代工具的开发，这些工具不再依赖现有的内核态 602 态的功能，而是在不影响执行效率或安全性的情况下主动重新编程运行时行为，数据包过滤模块包括 “BPF Bytecode”、
25 “Verifier+JIT” 和 “Native Code” 等。

(1) map(映射)是 BPF 最核心的方面之一，内核态 602 上运行的代码和加载了该代码的程序可以在运行时使用消息传递相互通信。

BPF 映射是驻留在内核态 602 中的键/值存储，任何 BPF 程序都可以访问它们。在用户态 604 中运行的程序也可以使用文件描述符访问这些映射。只要事先正确指定数据大小，就可以在映射中存储任何类型的数据。内核态 602 将键和值视为二进制代码，它并不关心在映射中保留的内容。BPF 验证程序包括多种保护措施，以确保创建和访问映射的方式是安全的。当然 BPF 映射类型有很多，比如哈希表映射、数组映射、Cgroup 数组映射等，分别满足不同的场景。

(2) BPF 验证程序也是在客户端系统上运行的程序，因此，对其进行严格审查是确
35 保其正确执行工作的目标。

验证程序执行的第一项检查是对 VM (Virtual Machine, 虚拟机) 即将加载的代码的静态分析。第一次检查的目的是确保程序有预期的结果。为此, 验证程序将使用代码创建有向循环图 (DAG)。验证程序分析的每个指令将成为图中的一个节点, 并且每个节点都链接到下一条指令。验证程序生成此图后, 它将执行深度优先搜索 (DFS), 以确保程序完成并且代码不包含危险路径。这意味着它将遍历图的每个分支, 一直到分支的底部, 以确保没有递归循环。

这些是验证器在第一次检查期间可能出现拒绝代码的情形, 要求有以下几个方面:

该程序不包含控制循环, 即为确保程序不会陷入无限循环, 验证程序会拒绝任何类型的控制循环。

10 该程序不会尝试执行超过内核态 602 允许的最大指令数的指令。此时, 可执行的最大指令数为 4096。此限制是为了防止 BPF 永远运行。该程序不包含任何无法访问的指令, 例如从未执行过的条件或功能。这样可以防止在 VM 中加载无效代码, 这也会延迟 BPF 程序的终止, 该程序不会尝试越界。

15 验证者执行的第二项检查是 BPF 程序的空运行。这意味着验证者将尝试分析程序将要执行的每条指令, 以确保它不会执行任何无效的指令。此执行还将检查所有内存指针是否均已正确访问和取消引用。最后, 空运行向验证程序通知程序中的控制流, 以确保无论程序采用哪个控制路径, 它都会到达 BPF_EXIT 指令。为此, 验证程序会跟踪堆栈中所有访问过的分支路径, 并在采用新路径之前对其进行评估, 以确保它不会多次访问特定路径。经过这两项检查后, 验证者认为程序可以安全执行。

20 (3) hook: 由于 eBPF 是事件驱动的, 所以 eBPF 是作用于具体的 hook 的, 根据不同的作用, 常用的有 XDP、trace 和套接字等。

(4) 帮助函数: eBPF 程序无法调用任意内核态 602 功能 (即图 6 所示的 Kernel Function), 允许这样做会将 eBPF 程序绑定到特定的内核态 602 版本, 并使程序的兼容性复杂化。取而代之的是, eBPF 程序可以调用帮助函数, 该函数是内核态 602 提供的众所周知且稳定的 API。

图 6 所示的网络数据包的服务质量处理架构应用于本方案时, 应用程序的数据报文 skb 从协议栈从上往下到达 IP 层时, 调用 eBPF 代码, 获取该 socket 对应的 tos 值。协议栈代码将该值写入 IP 数据包的头的 tos 字段, 从而实现了对流量的 Qos 的控制, 具体包括以下几个关键阶段。

30 (1) 内核代码修改, 具体包括:

(1.1) 扩展内核代码, 加入新的 ebpf 系统调用, 实现对 tos map 进行更新。

(1.2) 加入 eBPF 代码调用点, 确保 socket 在首次路由之后, 调用 eBPF 代码获取 tos 值并缓存到 socket 元数据中, 随后的封包直接使用该缓存值。

(2) 运行在内核态的 ebpf 程序, 该程序负责读取 tos map 数据, 根据映射逻辑返回正确的 tos 值。

(3) 用户态配置程序开发, 即用户态 604 通过读取配置文件或者通过命令行参数, 将四元组和 tos 值的对应关系通过系统调用写入 tos map 中, 需要支持增删查改等常见操作。

图 7 是本公开示例性实施例中一种网络数据包的服务质量处理装置的方框图。

5 参考图 7, 网络数据包的服务质量处理装置 700 可以包括:

调用模块 702, 设置为调用数据包过滤模块获取套接字对应的服务类型字段值。

缓存模块 704, 设置为将服务类型字段值缓存至套接字的元数据中。

封包模块 706, 设置为基于缓存的服务类型字段值对路由后的网络数据包进行封包处理。

10 在本公开的一种示例性实施例中, 在调用数据包过滤模块获取套接字对应的服务类型字段值前, 还包括: 在内核态加入数据包过滤模块, 该数据包过滤模块用于对用户态的服务类型字段图表进行更新; 在内核态的协议栈的网络数据包的服务质量处理进程中加入数据包过滤模块的调用点, 套接字在首次路由后在调用点调用数据包过滤模块, 以获取服务类型字段值并缓存至套接字的元数据中; 其中, 服务类型字段图表中包括网络数据包的四元组与服务类型字段值之间的对应关系。

在本公开的一种示例性实施例中, 网络数据包的服务质量处理装置 700 还用于: 触发用户态下网络数据包对应的配置文件或命令行参数确定网络数据包的四元组与服务类型字段值之间的对应关系; 将该对应关系写入服务类型字段图表。

20 在本公开的一种示例性实施例中, 网络数据包的服务质量处理装置 700 还用于: 确定封包后的网络数据包的服务类型字段值; 根据服务类型字段值对封包后的网络数据包进行分类控制处理。

在本公开的一种示例性实施例中, 数据包过滤模块包括 eBPF 模块。

在本公开的一种示例性实施例中, 服务类型字段值包括 tos 值。

25 在本公开的一种示例性实施例中, 四元组为网络数据包对应的源 IP 地址、源端口、目标 IP 地址和目标端口。

由于网络数据包的服务质量处理装置 700 的各功能已在其对应的方法实施例中予以详细说明, 本公开于此不再赘述。

30 应当注意, 尽管在上文详细描述中提及了用于动作执行的设备的若干模块或者单元, 但是这种划分并非强制性的。实际上, 根据本公开的实施方式, 上文描述的两个或更多模块或者单元的特征和功能可以在一个模块或者单元中具体化。反之, 上文描述的一个模块或者单元的特征和功能可以进一步划分为由多个模块或者单元来具体化。

在本公开的示例性实施例中, 还提供了一种能够实现上述方法的电子设备。

35 所属技术领域的技术人员能够理解, 本发明实施例的各个方面可以实现为系统、方法或程序产品。因此, 本发明实施例的各个方面可以具体实现为以下形式, 即: 完全的硬件实施方式、完全的软件实施方式 (包括固件、微代码等), 或硬件和软件方面结合的实施

方式，这里可以统称为“电路”、“模块”或“系统”。

下面参照图 8 来描述根据本发明的这种实施方式的电子设备 800。图 8 显示的电子设备 800 仅仅是一个示例，不应对本发明实施例的功能和使用范围带来任何限制。

如图 8 所示，电子设备 800 以通用计算设备的形式表现。电子设备 800 的组件可以包括但不限于：上述至少一个处理单元 810、上述至少一个存储单元 820、连接不同系统组件（包括存储单元 820 和处理单元 810）的总线 830。

其中，存储单元存储有程序代码，程序代码可以被处理单元 810 执行，使得处理单元 810 执行本说明书上述“示例性方法”部分中描述的根据本发明各种示例性实施方式的步骤。例如，处理单元 810 可以执行如本公开实施例所示的方法。

存储单元 820 可以包括易失性存储单元形式的可读介质，例如随机存取存储单元（RAM）8201 和/或高速缓存存储单元 8202，还可以进一步包括只读存储单元（ROM）8203。

存储单元 820 还可以包括具有一组（至少一个）程序模块 8205 的程序/实用工具 8204，这样的程序模块 8205 包括但不限于：操作系统、一个或者多个应用程序、其它程序模块以及程序数据，这些示例中的每一个或某种组合中可能包括网络环境的实现。

总线 830 可以为表示几类总线结构中的一种或多种，包括存储单元总线或者存储单元控制器、外围总线、图形加速端口、处理单元或者使用多种总线结构中的任意总线结构的局域总线。

电子设备 800 也可以与一个或多个外部设备 840（例如键盘、指向设备、蓝牙设备等）通信，还可与一个或者多个使得用户能与该电子设备 800 交互的设备通信，和/或与使得该电子设备 800 能与一个或多个其它计算设备进行通信的任何设备（例如路由器、调制解调器等等）通信。这种通信可以通过输入/输出（I/O）接口 850 进行。并且，电子设备 800 还可以通过网络适配器 860 与一个或者多个网络（例如局域网（LAN），广域网（WAN）和/或公共网络，例如因特网）通信。如图所示，网络适配器 860 通过总线 830 与电子设备 800 的其它模块通信。应当明白，尽管图中未示出，可以结合电子设备 800 使用其它硬件和/或软件模块，包括但不限于：微代码、设备驱动器、冗余处理单元、外部磁盘驱动阵列、RAID 系统、磁带驱动器以及数据备份存储系统等。

通过以上的实施方式的描述，本领域的技术人员易于理解，这里描述的示例实施方式可以通过软件实现，也可以通过软件结合必要的硬件的方式来实现。因此，根据本公开实施方式的技术方案可以以软件产品的形式体现出来，该软件产品可以存储在一个非易失性存储介质（可以是 CD-ROM，U 盘，移动硬盘等）中或网络上，包括若干指令以使得一台计算设备（可以是个人计算机、服务器、终端装置、或者网络设备等）执行根据本公开实施方式的方法。

在本公开的示例性实施例中，还提供了一种计算机可读存储介质，其上存储有能够实现本说明书上述方法的程序产品。在一些可能的实施方式中，本发明实施例的各个方面还

可以实现为一种程序产品的形式，其包括程序代码，当程序产品在终端设备上运行时，程序代码用于使终端设备执行本说明书上述“示例性方法”部分中描述的根据本发明各种示例性实施方式的步骤。

根据本发明的实施方式的用于实现上述方法的程序产品可以采用便携式紧凑盘只读存储器(CD-ROM)并包括程序代码，并可以在终端设备，例如个人电脑上运行。然而，本
5 发明实施例的程序产品不限于此，在本文件中，可读存储介质可以是任何包含或存储程序的有形介质，该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。

程序产品可以采用一个或多个可读介质的任意组合。可读介质可以是可读信号介质或者可读存储介质。可读存储介质例如可以为但不限于电、磁、光、电磁、红外线、或半导体的系统、装置或器件，或者任意以上的组合。可读存储介质的更具体的例子（非穷举的
10 列表）包括：具有一个或多个导线的电连接、便携式盘、硬盘、随机存取存储器（RAM）、只读存储器（ROM）、可擦式可编程只读存储器（EPROM 或闪存）、光纤、便携式紧凑盘只读存储器(CD-ROM)、光存储器件、磁存储器件、或者上述的任意合适的组合。

计算机可读信号介质可以包括在基带中或者作为载波一部分传播的数据信号，其中承载了可读程序代码。这种传播的数据信号可以采用多种形式，包括但不限于电磁信号、光信号或上述的任意合适的组合。可读信号介质还可以是可读存储介质以外的任何可读介
15 质，该可读介质可以发送、传播或者传输用于由指令执行系统、装置或者器件使用或者与其结合使用的程序。

可读介质上包含的程序代码可以用任何适当的介质传输，包括但不限于无线、有线、
20 光缆、RF 等等，或者上述的任意合适的组合。

可以以一种或多种程序设计语言的任意组合来编写用于执行本发明实施例操作的程序代码，程序设计语言包括面向对象的程序设计语言—诸如 Java、C++等，还包括常规的过程式程序设计语言—诸如“C”语言或类似的程序设计语言。程序代码可以完全地在用户计算设备上执行、部分地在用户设备上执行、作为一个独立的软件包执行、部分在用户
25 计算设备上部分在远程计算设备上执行、或者完全在远程计算设备或服务器上执行。在涉及远程计算设备的情形中，远程计算设备可以通过任意种类的网络，包括局域网（LAN）或广域网（WAN），连接到用户计算设备，或者，可以连接到外部计算设备（例如利用因特网服务提供商来通过因特网连接）。

此外，上述附图仅是根据本发明示例性实施例的方法所包括的处理的示意性说明，而
30 不是限制目的。易于理解，上述附图所示的处理并不表明或限制这些处理的时间顺序。另外，也易于理解，这些处理可以是例如在多个模块中同步或异步执行的。

本领域技术人员在考虑说明书及实践这里公开的发明后，将容易想到本公开的其它实施方案。本申请实施例旨在涵盖本公开的任何变型、用途或者适应性变化，这些变型、用途或者适应性变化遵循本公开的一般性原理并包括本公开未公开的本技术领域中的公知
35 常识或惯用技术手段。说明书和实施例仅被视为示例性的，本公开的真正范围和构思由权

利要求指出。

工业实用性

- 5 本公开通过调用数据包过滤模块获取套接字对应的服务类型字段值,并将服务类型字段值缓存至套接字的元数据中,进而基于缓存的服务类型字段值对路由后的网络数据包进行封包处理,相较于相关技术中的按包设置而言,本公开的技术方案提高了实现了服务类型字段的按流设置,服务类型字段值更新的效率和灵活性,减少了服务类型字段值更新所需的开销,降低了网络数据包的交互压力。

权利要求

1. 一种网络数据包的服务质量处理方法，包括：
调用数据包过滤模块获取套接字对应的服务类型字段值；
将所述服务类型字段值缓存至所述套接字的元数据中；
5 基于缓存的服务类型字段值对路由后的网络数据包进行封包处理。
2. 如权利要求 1 所述的网络数据包的服务质量处理方法，其中，在调用数据包过滤模块获取套接字对应的服务类型字段值前，还包括：
在内核态加入所述数据包过滤模块，所述数据包过滤模块用于对用户态的服务类型字段图表进行更新；
10 在所述内核态的协议栈的网络数据包的服务质量处理进程中加入所述数据包过滤模块的调用点，所述套接字在首次路由后在所述调用点调用所述数据包过滤模块，以获取所述服务类型字段值并缓存至所述套接字的元数据中，
其中，所述服务类型字段图表中包括所述网络数据包的四元组与所述服务类型字段值之间的对应关系。
- 15 3. 如权利要求 2 所述的网络数据包的服务质量处理方法，其中，还包括：
触发所述用户态下所述网络数据包对应的配置文件或命令行参数确定所述对应关系；
将所述对应关系写入所述服务类型字段图表。
4. 如权利要求 1 所述的网络数据包的服务质量处理方法，其中，还包括：
确定封包后的所述网络数据包的服务类型字段值；
20 根据所述服务类型字段值对封包后的所述网络数据包进行分类控制处理。
5. 如权利要求 1-4 中任一项所述的网络数据包的服务质量处理方法，其中，所述数据包过滤模块包括 eBPF 模块。
6. 如权利要求 1-4 中任一项所述的网络数据包的服务质量处理方法，其中，所述服务类型字段值包括 tos 值。
- 25 7. 如权利要求 1-4 中任一项所述的网络数据包的服务质量处理方法，其中，所述四元组为所述网络数据包对应的源 IP 地址、源端口、目标 IP 地址和目标端口。
8. 一种网络数据包的服务质量处理装置，包括：
调用模块，设置为调用数据包过滤模块获取套接字对应的服务类型字段值；
缓存模块，设置为将所述服务类型字段值缓存至所述套接字的元数据中；
30 封包模块，设置为基于缓存的服务类型字段值对路由后的网络数据包进行封包处理。
9. 一种电子设备，包括：
存储器；以及
耦合到所述存储器的处理器，所述处理器被配置为基于存储在所述存储器中的指令，
执行如权利要求 1-7 中任一项所述的网络数据包的服务质量处理方法。
- 35 10. 一种计算机可读存储介质，其上存储有程序，该程序被处理器执行时实现如权利

要求 1-7 中任一项所述的网络数据包的服务质量处理方法。