



(12) 发明专利申请

(10) 申请公布号 CN 115580667 A

(43) 申请公布日 2023. 01. 06

(21) 申请号 202211588437.2

(22) 申请日 2022.12.12

(71) 申请人 腾讯科技(深圳)有限公司

地址 518057 广东省深圳市南山区高新区
科技中一路腾讯大厦35层

(72) 发明人 包增辉 袁勇

(74) 专利代理机构 北京三高永信知识产权代理
有限责任公司 11138

专利代理师 徐耿铭

(51) Int. Cl.

H04L 69/12 (2022.01)

H04L 69/164 (2022.01)

H04L 67/14 (2022.01)

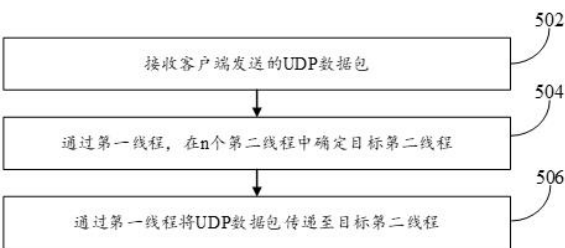
权利要求书3页 说明书22页 附图12页

(54) 发明名称

数据传输方法、装置、设备及存储介质

(57) 摘要

本申请公开了一种数据传输方法、装置、设备及存储介质,属于计算机技术领域。所述方法包括:接收客户端发送的UDP数据包;通过第一线程,在n个第二线程中确定目标第二线程,n个第二线程用于处理UDP数据包,目标第二线程用于管理UDP数据包对应的QUIC连接;通过第一线程将UDP数据包传递至目标第二线程。通过第一线程调度第二线程,不同第二线程可实现管理不同的QUIC连接,实现了通过不同线程来管理QUIC连接,使服务端能够充分利用多个核心的并发处理能力,避免了使用单一线程管理QUIC连接导致对并发量瓶颈限制较大的问题,提升了管理QUIC连接的效率,有助于提升数据传输的性能。



1. 一种数据传输方法,其特征在于,所述方法由服务器执行,所述服务器中运行的服务端与客户端建立有通信连接,所述服务端和所述客户端中集成有快速用户数据报协议互联网连接-软件开发工具包QUIC-SDK,所述QUIC-SDK用于实现基于QUIC协议的传输,所述方法包括:

接收所述客户端发送的用户数据报协议UDP数据包;

通过第一线程,在n个第二线程中确定目标第二线程,所述n个第二线程用于处理所述UDP数据包,所述目标第二线程用于管理所述UDP数据包对应的QUIC连接,所述第一线程和所述第二线程是由所述QUIC-SDK提供的,n为大于1的整数;

通过所述第一线程将所述UDP数据包传递至所述目标第二线程。

2. 根据权利要求1所述的方法,其特征在于,所述通过第一线程,在n个第二线程中确定目标第二线程,包括:

响应于所述第一线程接收到所述UDP数据包,通过所述第一线程解析所述UDP数据包,得到目标连接标识;

通过所述第一线程,根据所述目标连接标识在所述n个第二线程中确定所述目标第二线程。

3. 根据权利要求2所述的方法,其特征在于,所述通过所述第一线程,根据所述目标连接标识在所述n个第二线程中确定所述目标第二线程,包括:

在所述目标连接标识对应的QUIC连接为已存在的连接的情况下,通过所述第一线程,根据缓存将包括所述目标连接标识对应的会话对象的第二线程确定为所述目标第二线程;

其中,所述缓存用于存储不同的所述第二线程中的会话对象和连接标识之间的对应关系,所述会话对象与所述QUIC连接对应。

4. 根据权利要求3所述的方法,其特征在于,所述n个第二线程中的每个线程包括调度组件,所述调度组件用于管理一个或多个所述会话对象;

所述通过所述第一线程将所述UDP数据包传递至所述目标第二线程,包括:

通过所述第一线程将所述UDP数据包传递至所述目标第二线程中的所述调度组件;

通过所述目标第二线程中的所述调度组件确定所述目标连接标识对应的会话对象;

通过所述目标第二线程中的所述调度组件将所述UDP数据包传递至所述目标连接标识对应的会话对象。

5. 根据权利要求2所述的方法,其特征在于,所述通过所述第一线程,根据所述目标连接标识在所述n个第二线程中确定所述目标第二线程,包括:

在所述目标连接标识对应的QUIC连接不为已存在的连接的情况下,通过所述第一线程,在所述n个第二线程中随机确定所述目标第二线程。

6. 根据权利要求5所述的方法,其特征在于,所述n个第二线程中的每个线程包括调度组件,所述调度组件用于管理一个或多个会话对象,所述会话对象与所述QUIC连接对应;

所述通过所述第一线程将所述UDP数据包传递至所述目标第二线程,包括:

通过所述第一线程将所述UDP数据包传递至所述目标第二线程中的所述调度组件;

在所述目标第二线程中的所述调度组件确定所述目标连接标识对应的QUIC连接不为所述已存在的连接的情况下,建立所述UDP数据包的QUIC连接;

在所述目标第二线程中创建所述UDP数据包的QUIC连接对应的会话对象;

通过所述目标第二线程中的所述调度组件将所述UDP数据包传递至所述UDP数据包的QUIC连接对应的会话对象。

7. 根据权利要求6所述的方法, 其特征在于, 所述方法还包括:

在缓存中保存所述目标第二线程中的所述会话对象与所述目标连接标识的对应关系。

8. 根据权利要求6所述的方法, 其特征在于, 所述在所述目标第二线程中创建所述UDP数据包的QUIC连接对应的会话对象, 包括:

基于UDP文件描述符, 在所述目标第二线程中创建所述UDP数据包的QUIC连接对应的会话对象;

其中, 所述UDP文件描述符设置有复用规则, 所述复用规则用于所述第一线程和所述第二线程绑定相同的针对所述UDP数据包的侦听地址。

9. 根据权利要求1至8任一所述的方法, 其特征在于, 所述方法还包括:

在所述服务端的应用层传输发送数据的过程中, 将所述发送数据传递至QUIC线程中的发包处理组件, 所述QUIC线程是由所述QUIC-SDK提供的;

通过所述发包处理组件将所述发送数据保存至本地发送缓存中;

在通过所述发包处理组件进行数据发送时, 发送所述本地发送缓存中的数据。

10. 根据权利要求9所述的方法, 其特征在于, 所述在通过所述发包处理组件进行数据发送时, 发送所述本地发送缓存中的数据, 包括:

在通过所述发包处理组件进行数据发送时, 发送所述本地发送缓存中的全部数据。

11. 根据权利要求9所述的方法, 其特征在于, 所述方法还包括:

通过所述发包处理组件基于发送事件文件描述符对消息环进行唤醒通知, 所述消息环用于管理传输任务;

通过所述消息环唤醒所述发包处理组件以指示所述发包处理组件进行数据发送。

12. 根据权利要求1至8任一所述的方法, 其特征在于, 所述方法还包括:

在接收到所述客户端发送的崩溃信号的情况下, 通过云存储服务器获取崩溃文件;

根据所述崩溃文件确定崩溃的原因和位置;

其中, 所述崩溃文件是所述客户端通过所述QUIC-SDK监测到运行崩溃的情况下生成并上传至所述云存储服务器的。

13. 一种数据传输装置, 其特征在于, 所述装置中运行的服务端与客户端建立有通信连接, 所述服务端和所述客户端中集成有QUIC-SDK, 所述QUIC-SDK用于实现基于QUIC协议的传输, 所述装置包括:

接收模块, 用于接收所述客户端发送的UDP数据包;

确定模块, 用于通过第一线程, 在n个第二线程中确定目标第二线程, 所述n个第二线程用于处理所述UDP数据包, 所述目标第二线程用于管理所述UDP数据包对应的QUIC连接, 所述第一线程和所述第二线程是由所述QUIC-SDK提供的, n为大于1的整数;

传递模块, 用于通过所述第一线程将所述UDP数据包传递至所述目标第二线程。

14. 一种计算机设备, 其特征在于, 所述计算机设备包括处理器和存储器, 所述存储器中存储有至少一条指令、至少一段程序、代码集或指令集, 所述至少一条指令、所述至少一段程序、所述代码集或所述指令集由所述处理器加载并执行以实现如权利要求1至12任一所述的数据传输方法。

15.一种计算机可读存储介质,其特征在于,所述可读存储介质中存储有至少一条指令、至少一段程序、代码集或指令集,所述至少一条指令、所述至少一段程序、所述代码集或指令集由处理器加载并执行以实现如权利要求1至12任一所述的数据传输方法。

数据传输方法、装置、设备及存储介质

技术领域

[0001] 本申请涉及计算机技术领域,特别涉及一种数据传输方法、装置、设备及存储介质。

背景技术

[0002] 云游戏是以云计算为基础的游戏方式。在云游戏的模式下,服务端用于运行游戏并将渲染完毕后的游戏画面压缩后发送至客户端,客户端用于将用户的操作事件发送至服务端以控制游戏的运行。在此过程中,会涉及到频繁的数据传输,因此对传输延迟的要求非常高。

[0003] 用户数据报协议互联网连接(Quick User Datagram Protocol Internet Connection,QUIC)协议是使用用户数据报协议(User Datagram Protocol,UDP)进行多路并发传输的协议,相较于传输控制协议(Transmission Control Protocol,TCP)有诸多优势。相关技术中,通常采用QUIC协议来实现数据传输,例如采用gQUIC协议(一种QUIC协议),以提升数据传输的性能。

[0004] 在采用gQUIC协议对应的QUIC库(包括用于实现gQUIC协议的代码)的情况下,由于无法充分使用中央处理器(Central Processing Unit,CPU)的多核心来管理QUIC连接,存在效率较低的问题。

发明内容

[0005] 本申请提供了一种数据传输方法、装置、设备及存储介质,可以提升管理QUIC连接的效率。所述技术方案如下。

[0006] 根据本申请的一方面,提供了一种数据传输方法,所述方法由服务器执行,所述服务器中运行的服务端与客户端建立有通信连接,所述服务端和所述客户端中集成有QUIC-软件开发工具包(Software Development Kit,SDK),所述QUIC-SDK用于实现基于QUIC协议的传输,所述方法包括以下步骤。

[0007] 接收所述客户端发送的UDP数据包。

[0008] 通过第一线程,在n个第二线程中确定目标第二线程,所述n个第二线程用于处理所述UDP数据包,所述目标第二线程用于管理所述UDP数据包对应的QUIC连接,所述第一线程和所述第二线程是由所述QUIC-SDK提供的,n为大于1的整数。

[0009] 通过所述第一线程将所述UDP数据包传递至所述目标第二线程。

[0010] 根据本申请的一方面,提供了一种数据传输方法,所述方法由客户端执行,所述客户端与服务器中运行的服务端建立有通信连接,所述服务端和所述客户端中集成有QUIC-SDK,所述QUIC-SDK用于实现基于QUIC协议的传输,所述方法包括以下步骤。

[0011] 向所述服务端发送UDP数据包。

[0012] 其中,所述服务端用于通过第一线程,在n个第二线程中确定目标第二线程,并通过所述第一线程将所述UDP数据包传递至所述目标第二线程,所述n个第二线程用于处理所

述UDP数据包,所述目标第二线程用于管理所述UDP数据包对应的QUIC连接,所述第一线程和所述第二线程是由所述QUIC-SDK提供的,n为大于1的整数。

[0013] 根据本申请的另一方面,提供了一种数据传输装置,所述装置中运行的服务端与客户端建立有通信连接,所述服务端和所述客户端中集成有QUIC-SDK,所述QUIC-SDK用于实现基于QUIC协议的传输,所述装置包括以下模块。

[0014] 接收模块,用于接收所述客户端发送的用户数据报协议UDP数据包。

[0015] 确定模块,用于通过第一线程,在n个第二线程中确定目标第二线程,所述n个第二线程用于处理所述UDP数据包,所述目标第二线程用于管理所述UDP数据包对应的QUIC连接,所述第一线程和所述第二线程是由所述QUIC-SDK提供的,n为大于1的整数。

[0016] 传递模块,用于通过所述第一线程将所述UDP数据包传递至所述目标第二线程。

[0017] 在一个可选的设计中,所述确定模块,用于:

响应于所述第一线程接收到所述UDP数据包,通过所述第一线程解析所述UDP数据包,得到目标连接标识。

[0018] 通过所述第一线程,根据所述目标连接标识在所述n个第二线程中确定目标第二线程。

[0019] 在一个可选的设计中,所述确定模块,用于:

在所述目标连接标识对应的QUIC连接为已存在的连接的情况下,通过所述第一线程,根据缓存将包括所述目标连接标识对应的会话对象的第二线程确定为所述目标第二线程。

[0020] 其中,所述缓存用于存储不同的所述第二线程中的会话对象和连接标识之间的对应关系,所述会话对象与所述QUIC连接对应。

[0021] 在一个可选的设计中,所述n个第二线程中的每个线程包括调度组件,所述调度组件用于管理一个或多个所述会话对象。所述传递模块,用于:

通过所述第一线程将所述UDP数据包传递至所述目标第二线程中的所述调度组件。

[0022] 所述确定模块,用于通过所述目标第二线程中的所述调度组件确定所述目标连接标识对应的会话对象。

[0023] 所述传递模块,用于通过所述目标第二线程中的所述调度组件将所述UDP数据包传递至所述目标连接标识对应的会话对象。

[0024] 在一个可选的设计中,所述确定模块,用于:

在所述目标连接标识对应的QUIC连接不为已存在的连接的情况下,通过所述第一线程在所述n个第二线程中随机确定所述目标第二线程。

[0025] 在一个可选的设计中,所述n个第二线程中的每个线程包括调度组件,所述调度组件用于管理一个或多个会话对象,所述会话对象与所述QUIC连接对应。所述装置还包括以下模块。

[0026] 所述传递模块,用于通过所述第一线程将所述UDP数据包传递至所述目标第二线程中的所述调度组件。

[0027] 建立模块,用于在所述目标第二线程中的所述调度组件确定所述目标连接标识对应的QUIC连接不为所述已存在的连接的情况下,建立所述UDP数据包的QUIC连接。

[0028] 所述建立模块,用于在所述目标第二线程中创建所述UDP数据包的QUIC连接对应的会话对象。

[0029] 所述传递模块,用于通过所述目标第二线程中的所述调度组件将所述UDP数据包传递至所述UDP数据包的QUIC连接对应的会话对象。

[0030] 在一个可选的设计中,所述装置还包括以下模块。

[0031] 保存模块,用于在缓存中保存所述目标第二线程中的所述会话对象与所述目标连接标识的对应关系。

[0032] 在一个可选的设计中,所述建立模块,用于:

基于UDP文件描述符在所述目标第二线程中创建所述UDP数据包的QUIC连接对应的会话对象。

[0033] 其中,所述UDP文件描述符设置有复用规则,所述复用规则用于所述第一线程和所述第二线程绑定相同的针对所述UDP数据包的侦听地址。

[0034] 在一个可选的设计中,所述装置还包括以下模块。

[0035] 所述传递模块,用于在所述服务端的应用层传输发送数据的过程中,将所述发送数据传递至QUIC线程中的发包处理组件,所述QUIC线程是由所述QUIC-SDK提供的。

[0036] 保存模块,用于通过所述发包处理组件将所述发送数据保存至本地发送缓存中。

[0037] 发送模块,用于在通过所述发包处理组件进行数据发送时,发送所述本地发送缓存中的数据。

[0038] 在一个可选的设计中,所述发送模块,用于:

在通过所述发包处理组件进行数据发送时,发送所述本地发送缓存中的全部数据。

[0039] 在一个可选的设计中,所述装置还包括以下模块。

[0040] 通知模块,用于通过所述发包处理组件基于发送事件文件描述符对消息环进行唤醒通知,所述消息环用于管理传输任务。

[0041] 唤醒模块,用于通过所述消息环唤醒所述发包处理组件以指示所述发包处理组件进行数据发送。

[0042] 在一个可选的设计中,所述装置还包括以下模块。

[0043] 获取模块,用于在接收到所述客户端发送的崩溃信号的情况下,通过云存储服务器获取崩溃文件。

[0044] 所述确定模块,用于根据所述崩溃文件确定崩溃的原因和位置。

[0045] 其中,所述崩溃文件是所述客户端通过所述QUIC-SDK监测到运行崩溃的情况下生成并上传至所述云存储服务器的。

[0046] 在一个可选的设计中,所述装置还包括以下模块。

[0047] 切换模块,用于在接收到所述客户端发送的所述崩溃信号的情况下,对应用层协议进行切换。

[0048] 在一个可选的设计中,所述装置还包括以下模块。

[0049] 保存模块,用于在所述服务端运行的过程中,将性能侦测数据格式化,并保存至本地文件中。

[0050] 上传模块,用于在所述服务端结束服务的情况下,将所述本地文件上传至云存储

服务器。

[0051] 其中,所述云存储服务器中的所述本地文件用于解析得到性能侦测信息,所述性能侦测信息用于在用户界面中展示。

[0052] 根据本申请的另一方面,提供了一种数据传输装置,所述装置与服务器中运行的服务端建立有通信连接,所述服务端和所述装置中集成有QUIC-SDK,所述QUIC-SDK用于实现基于QUIC协议的传输,所述装置包括以下模块。

[0053] 发送模块,用于向所述服务端发送UDP数据包。

[0054] 其中,所述服务端用于通过第一线程,在n个第二线程中确定目标第二线程,并通过所述第一线程将所述UDP数据包传递至所述目标第二线程,所述n个第二线程用于处理所述UDP数据包,所述目标第二线程用于管理所述UDP数据包对应的QUIC连接,所述第一线程和所述第二线程是由所述QUIC-SDK提供的,n为大于1的整数。

[0055] 在一个可选的设计中,所述装置还包括以下模块。

[0056] 传递模块,用于在所述客户端的应用层传输发送数据的过程中,将所述发送数据传递至QUIC线程中的发包处理组件,所述QUIC线程是由所述QUIC-SDK提供的。

[0057] 保存模块,用于通过所述发包处理组件将所述发送数据保存至本地发送缓存中。

[0058] 所述发送模块,用于在通过所述发包处理组件进行数据发送时,发送所述本地发送缓存中的数据。

[0059] 根据本申请的另一方面,提供了一种计算机设备,所述计算机设备包括处理器和存储器,所述存储器中存储有至少一条指令、至少一段程序、代码集或指令集,所述至少一条指令、所述至少一段程序、所述代码集或指令集由所述处理器加载并执行以实现如上方面所述的数据传输方法。

[0060] 根据本申请的另一方面,提供了一种计算机可读存储介质,所述可读存储介质中存储有至少一条指令、至少一段程序、代码集或指令集,所述至少一条指令、所述至少一段程序、所述代码集或指令集由处理器加载并执行以实现如上方面所述的数据传输方法。

[0061] 根据本申请的另一方面,提供了一种计算机程序产品或计算机程序,该计算机程序产品或计算机程序包括计算机指令,该计算机指令存储在计算机可读存储介质中。计算机设备的处理器从计算机可读存储介质读取该计算机指令,处理器执行该计算机指令,使得该计算机设备执行上述方面的各种可选实现方式中提供的数据传输方法。

[0062] 本申请提供的技术方案带来的有益效果至少包括:

通过第一线程调度第二线程,不同第二线程可实现管理不同的QUIC连接,实现了通过不同线程来管理QUIC连接,使服务端能够充分利用多个核心的并发处理能力,避免了使用单一线程管理QUIC连接导致对并发量瓶颈限制较大的问题,提升了管理QUIC连接的效率,有助于提升数据传输的性能。

附图说明

[0063] 为了更清楚地说明本申请实施例中的技术方案,下面将对实施例描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本申请的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0064] 图1是本申请一个示例性实施例提供的云游戏业务的后台布署的示意图；
图2是本申请一个示例性实施例提供的相关技术中基于QUIC协议的部署方案的示意图；

图3是本申请一个示例性实施例提供的计算机系统的结构框图；

图4是本申请一个示例性实施例提供的基于QUIC协议的云游戏的架构模式的示意图；

图5是本申请一个示例性实施例提供的一种数据传输方法的流程示意图；

图6是本申请一个示例性实施例提供的另一种数据传输方法的流程示意图；

图7是本申请一个示例性实施例提供的又一种数据传输方法的流程示意图；

图8是本申请一个示例性实施例提供的发送数据的过程的示意图；

图9是本申请一个示例性实施例提供的再一种数据传输方法的流程示意图；

图10是本申请一个示例性实施例提供的崩溃分析的过程的示意图；

图11是本申请一个示例性实施例提供的还一种数据传输方法的流程示意图；

图12是本申请一个示例性实施例提供的相关技术中进行性能侦测的过程的示意图；

图13是本申请一个示例性实施例提供的性能侦测的过程的示意图；

图14是本申请一个示例性实施例提供的基于QUIC协议的传输过程的示意图；

图15是本申请一个示例性实施例提供的流量控制的原理的示意图；

图16是本申请一个示例性实施例提供的数据传输方法的流程示意图；

图17是本申请一个示例性实施例提供的第一种数据传输装置的结构示意图；

图18是本申请一个示例性实施例提供的第二种数据传输装置的结构示意图；

图19是本申请一个示例性实施例提供的第三种数据传输装置的结构示意图；

图20是本申请一个示例性实施例提供的第四种数据传输装置的结构示意图；

图21是本申请一个示例性实施例提供的第五种数据传输装置的结构示意图；

图22是本申请一个示例性实施例提供的第六种数据传输装置的结构示意图；

图23是本申请一个示例性实施例提供的第七种数据传输装置的结构示意图；

图24是本申请一个示例性实施例提供的第八种数据传输装置的结构示意图；

图25是本申请一个示例性实施例提供的第九种数据传输装置的结构示意图；

图26是本申请一个示例性实施例提供的第十种数据传输装置的结构示意图；

图27是本申请一个示例性实施例提供的计算机设备的结构示意图。

[0065] 此处的附图被并入说明书中并构成本说明书的一部分，示出了符合本申请的实施例，并与说明书一起用于解释本申请的原理。

具体实施方式

[0066] 为使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请实施方式作进一步地详细描述。

[0067] 首先，对本申请实施例涉及的名词进行介绍。

[0068] 云游戏：云游戏是以云计算为基础的游戏方式。在云游戏的运行模式下，所有游戏都在服务端运行，服务端会将渲染完毕后的游戏画面压缩后通过网络传送给客户端（用

户)。客户端获取用户的操作事件,如触屏事件、键盘鼠标事件、摇杆事件等,通过网络传输到服务端,以达到操作游戏的目的。

[0069] QUIC协议:QUIC和英文quick谐音,简称“快”。QUIC协议是使用UDP进行多路并发传输的协议,相对于TCP有诸多优势。

[0070] IETF QUIC:互联网工程任务组(Internet Engineering Task Force,IETF)标准组织所提供的QUIC协议规范。

[0071] gQUIC:gQUIC协议是一种QUIC协议,本意是设计为通用协议以在浏览器中支持超文本传输安全协议(Hypertext Transfer Protocol Secure,HTTP(S)/HTTPS),同时其也在跟进支持最新的QUIC标准,与IETF QUIC有一些格式上的差异。

[0072] WebRTC:WebRTC是全球广域网(World Wide Web,WEB)实时通信(Real-Time Communication,RTC)的缩写,它既是应用程序编程接口(Application Programming Interface,API)也是协议。它是跨平台的音视频通讯框架,现已成为IETF Web标准。

[0073] Pacing(踱步):即有节奏、步调的进行。在音视频领域描述以固定的频率来执行某个动作。

[0074] 拥塞控制:作用于网络的机制,防止过多的数据注入到网络中,避免出现网络负载过大的情况。

[0075] 流量控制:指如果发送者发送数据过快,接收者来不及接收,那么就会有分组丢失。为了避免分组丢失,需控制发送者的发送速度,使得接收者来得及接收。

[0076] 云对象存储(Cloud Object Storage,COS):指无目录层次结构、无数据格式限制,可容纳海量数据且支持超文本传输协议(Hypertext Transfer Protocol,HTTP)/HTTPS访问的分布式存储服务,可具体实现为云存储服务器。

[0077] Promethues(普罗米修斯):一种开源的侦测告警解决方案,目前已广泛应用于Kubernetes(K8S,一种容器集群管理系统)云原生环境。

[0078] PushGateway(推送网关):一种针对短生命期或无法拉取的目标所提供的组件。

[0079] Grafana(格拉法纳):一款用Go语言(Golang,一种开发语言)开发的开源数据可视化工具,主要用来侦测数据的可视化、告警。

[0080] 在云游戏场景中,对用户的操作延迟非常敏感,延迟的大小主要取决于客户端的网络情况。客户端的网络存在不稳定的特点,比如存在切换网络(无线保真(Wireless Fidelity,Wi-Fi)与第四代(4th generation,4G)移动通信技术/第五代(5th generation,5G)移动通信技术的切换,或,在移动场景(地铁火车上)上频繁切换基站)的情况,以及弱网(网络抖动、丢包)的情况,传统的TCP、基于TCP的全双工通信协议(WebSocket)很难应对这种情况。

[0081] 本申请实施例提供的方法,通过在客户端和服务端之间建立基于QUIC协议的通信连接,相较于传统的TCP和WebSocket,能够减少延迟,从而优化弱网下的用户体验。示例地,本申请实施例提供的方法能够用于云游戏的场景下。图1是本申请一个示例性实施例提供的云游戏业务的后台部署的示意图。如图1所示,客户端101和K8S边缘集群中的X86(一种架构)服务器102建立有基于QUIC协议的通信连接,K8S边缘集群中的X86服务器102和ARM(一种架构)服务器103建立有基于TCP的通信连接。其中,客户端101向X86服务器102的Linux(一种操作系统)容器中的媒体传输模块发送的用户操作指令采用QUIC协议传输,X86服务

器102中的媒体传输模块向客户端101发送的云游戏的音视频流采用实时传输协议(Real-time Transport Protocol, RTP)或实时传输控制协议(Real-time Transport Control Protocol, RTCP)传输,客户端101和X86服务器102之间传输的鉴权、性能侦测数据采用QUIC协议传输。K8S边缘集群中的X86服务器102和ARM服务器103的Andriod(一种操作系统)容器中的媒体编码模块之间的数据传输采用TCP,媒体编码模块用于对云游戏的媒体数据进行编码,ARM服务器103的Andriod容器中的游戏实例即为运行的云游戏。需要说明的是,上述云游戏业务的后台布署方案仅用作实例,不作为对本申请的限制。

[0082] 由于云游戏业务有如下特殊性。

[0083] (1)对延迟要求非常高。

[0084] (2)边缘集群很多,要尽量减少运维成本,并降低单集群的资源成本。

[0085] 在选型实现方案时,需要进行一些权衡。另外,开源领域QUIC库的质量参差不齐,要做合适的选型并改造以适合云游戏的架构,需要做很多工作。本申请实施例提供了一种QUIC协议在云游戏业务的应用方案,能够满足低延迟的需求,且不增加运维成本、不增加单集群的资源成本。

[0086] 示例地,图2是本申请一个示例性实施例提供的相关技术中基于QUIC协议的部署方案的示意图。如图2所示,服务器外部的UDP包通过四层的弹性负载均衡(Elastic Load Balance, ELB)代理201以及七层的ELB代理202后,传输至后端(主机HOST203)中的业务进程,传输至主机HOST203的过程采用TCP,且主机HOST203之后的传输采用TCP。其中,七层ELB代理202中实现了QUIC Server,即基于QUIC协议实现的服务。

[0087] 相关技术中的上述方案存在如下缺点。

[0088] (1)七层的ELB代理需要独占机器,即该部署方案下要求每个边缘集群单独部署2台ELB代理。由于云游戏的边缘集群非常多,这大大增加了机器成本和布署运维成本。

[0089] (2)增加了传输延迟。由于传输链路增加了一跳(七层ELB代理),而七层ELB代理是在应用层实现的QUIC Server,这会增加内核到应用层的上下文切换。实际测试中,这一跳增加了0.5毫秒的延迟。

[0090] (3)增加了ELB代理,HOST主机的路由转发规则的变更。

[0091] 本申请实施例提供的方法,能够通过服务器和客户端中集成用于实现基于QUIC协议的传输的QUIC-SDK,以实现基于QUIC协议的传输。相较于上述方案,无需单独部署用于实现QUIC Server的ELB代理,能够节省机器成本和布署运维成本。并且由于无需部署用于实现QUIC Server的ELB代理,因此能够避免上述传输延期,以及避免了转发规则的变更导致的维护较为复杂的情况。

[0092] 图3是本申请一个示例性实施例提供的计算机系统的结构框图。该计算机系统300包括:终端310、服务器320。

[0093] 终端310安装和运行有支持云游戏的客户端311。当终端310调起客户端311运行时,终端310的屏幕上显示客户端311的用户界面。客户端311中集成有QUIC-SDK,QUIC-SDK用于实现基于QUIC协议的传输。终端310是用户312使用的终端,应用程序311上登录有用户312的用户帐号。终端310可以泛指多个终端中的一个。可选地,终端310的设备类型包括:智能手机、平板电脑、电子书阅读器、MP3播放器、MP4播放器、膝上型便携计算机和台式计算机中的至少一种。

[0094] 终端310通过无线网络或有线网络与服务器320中运行的服务端相连。

[0095] 服务器320包括一台服务器、多台服务器、云计算平台和虚拟化中心中的至少一种。服务器320用于为支持云游戏的客户端311提供后台服务。可选地,服务器320承担主要计算工作,终端310承担次要计算工作;或者,服务器320承担次要计算工作,终端310承担主要计算工作;或者,服务器320和终端310之间采用分布式计算架构进行协同计算。

[0096] 在一个示意性的例子中,服务器320包括处理器321、用户帐号数据库322、QUIC-SDK323、面向用户的输入/输出接口(Input/Output Interface,I/O接口)324。其中,处理器321用于加载服务器320中存储的指令,处理用户帐号数据库322和QUIC-SDK323中的数据;用户帐号数据库322用于存储终端310以及其它终端所使用的用户帐号的数据,比如用户帐号的头像、用户帐号的昵称、用户帐号所在的群组等;QUIC-SDK323用于实现基于QUIC协议的传输;面向用户的I/O接口324用于通过无线网络或有线网络和终端310建立通信交换数据。

[0097] 在选型QUIC库(用于实现QUIC-SDK)时,有如下要求:(1)能跟进QUIC标准,持续更新;(2)成熟、稳定;(3)支持跨平台;(4)支持服务端(server)和客户端(client),支持长连接。目前,开源领域没有成熟的、开箱即用的QUIC库(QUIC程序库)。总体来看,gQUIC库,具有代码质量较高、能够持续跟进标准,也能很好的跨平台的特点,但它存在以下缺点。

[0098] (1)难集成,它依赖了浏览器的很多组件,很难剥离出来。

[0099] (2)只实现了HTTP(基于QUIC)的传输方式,不支持长连接。

[0100] (3)对服务端不友好,不支持高并发,性能和延迟较差。

[0101] 本申请实施例提供了一种QUIC协议在云游戏场景的应用方案,这个方案的优点是传输延迟低、不增加运维成本、不增加单集群的资源成本,很适合边缘计算、要求低延迟的场景。通过提供QUIC-SDK(例如通过GameMatrix(游戏矩阵)SDK提供),以实现各类客户端与服务端建立基于QUIC协议的传输。可用于降低云游戏的操作延迟、提升弱网场景下长连接的稳定性。

[0102] 示例地,图4是本申请一个示例性实施例提供的基于QUIC协议的云游戏的架构模式的示意图。如图4的(a)所示,在该架构下云游戏包括客户端401和服务端(主机HOST)402。可选地,客户端401和服务端402之间部署有弹性负载均衡(Elastic Load Balance,ELB),例如为四层ELB。客户端401中集成有QUIC-SDK,服务端402的各应用层模块中集成有QUIC-SDK,以实现为服务端402的各应用层模块提供QUIC服务(QUIC Server),即实现为各应用层模块提供了基于QUIC协议的传输,QUIC-SDK用于实现基于QUIC协议的传输。在客户端和服务端中集成QUIC-SDK,这个方案的优点是不增加额外的传输延迟、不增加机器成本和运维成本,对于后台部署架构的兼容性较好。

[0103] 集成方式:可选地,本申请实施例中的QUIC-SDK,是基于对gQUIC的gQUIC库进行优化从而得到的。在基于gQUIC的gQUIC库进行QUIC-SDK的集成时,存在三种方案:(1)使用全量的gQUIC库,并将云游戏的业务代码加入,再构建业务模块;(2)将gQUIC库的代码抽离,成为新的库,再放入实际项目中使用;(3)在全量的gQUIC库上进行优化改造,封装成一个新的库,然后在实际项目中使用这个库。本申请的集成方案选择上述方案(3),具有工作量适中、维护成本较低且能够跟随gQUIC库进行升级的优势。需要说明的是,上述方案以集成gQUIC库为例进行说明,本申请提供的方案也能够采用上述方案以对其它QUIC库进行集成,上述

对gQUIC库集成过程的说明不作为对本申请的限制。可选地,具体的集成方案包括如下方案。

[0104] 对于服务端:基于C++语言(一种开发语言),将QUIC SDK编译成一个共享库,服务端进行链接引用。基于Go语言(一种开发语言),将本机QUIC(QUIC NATIVE)通过CGO工具(使Go语言与C语言相互调用)封装Go层的QUIC SDK。

[0105] 对于客户端:对于Android客户端,将QUIC NATIVE代码通过Java(一种开发语言)本机接口(Java Native Interface,JNI)层封装Java层的QUIC SDK。对于超文本标记语言5(Hyper Text Markup Language 5,H5)端和Web端,使用Web程序集(Assembly)技术,将QUIC SDK编译成Wasm(即Web Assembly),然后加载到浏览器中。

[0106] 在上述集成方案中,采用了共享库的方式,而不是静态库的方式。库是一种可执行代码的二进制形式,可以被操作系统载入内存执行,静态库的代码在编译过程中已经被载入可执行程序,而共享库的代码是在可执行程序运行时才载入内存的,在编译过程中仅简单的引用。采用共享库的原因在于gQUIC的符号(用于链接相关代码)较多,静态库的方式很容易造成符号冲突,导致无法编译。采用共享库的好处是不需要将QUIC相关代码集成到程序代码中,因此不会符号冲突。集成的QUIC-SDK还实现了长连接的功能。

[0107] 针对数据接收的优化:由于gQUIC库的服务端无法利用多个CPU核心的扩展能力,这会对服务端的并发量瓶颈产生较大的限制。如图4的(b)所示,本申请实施例中,服务端402在接收客户端401发送的UDP数据包时,会通过第一线程403解析UDP数据包,得到目标连接标识(Connection ID),并通过第一线程403根据缓存(缓存map)确定目标连接标识对应的QUIC连接是否为已存在的连接。在目标连接标识对应的QUIC连接为已存在的连接的情况下,服务端402通过第一线程403根据缓存将包括目标连接标识对应的会话对象(QuicSession)的第二线程404确定为目标第二线程,该会话对象与QUIC连接对应。之后通过第一线程403将UDP数据包传递至目标第二线程中的调度组件(dispatcher),并通过目标第二线程中的调度组件确定连接标识对应的会话对象,之后再通过目标第二线程中的调度组件将UDP数据包传递至连接标识对应的会话对象,以处理该UDP数据包。在连接标识对应的QUIC连接不为已存在的连接的情况下,服务端402通过第一线程403在n个第二线程404中随机确定目标第二线程,并通过第一线程403将UDP数据包传递至目标第二线程中的调度组件。之后服务端会建立UDP数据包的QUIC连接,并在目标第二线程中创建QUIC连接对应的会话对象,之后通过目标第二线程中的调度组件将UDP数据包传递至QUIC连接对应的会话对象,以处理该UDP数据包。通过上述方案,实现完成了对UDP数据包的接收,后续处理可通过会话对象实现。

[0108] 针对数据发送:在服务端402的应用层(应用层模块)传输发送数据的过程中,会将发送数据传递至QUIC线程中的发包处理组件,并通过发包处理组件将发送数据保存至本地发送缓存(local send buffer)中。之后在通过发包处理组件进行数据发送时,服务端402发送本地发送缓存中的数据。在此过程中,服务端402通过发包处理组件基于发送事件(Send Event)文件描述符(File Descriptor,FD)对消息环(Message Loop)进行唤醒通知,该消息环用于管理传输任务。之后通过消息环唤醒发包处理组件指示发包处理组件进行数据发送。需要说明的是,上述数据发送的方案也能够应用在客户端401中,即客户端401也通过QUIC-SDK执行上述数据发送的流程。

[0109] 针对崩溃分析:服务端402在接收到客户端404发送的崩溃信号的情况下,通过云存储服务器(例如COS)获取崩溃文件,从而根据崩溃文件确定崩溃的原因和位置。其中,崩溃文所述客户端401通过QUIC-SDK监测到运行崩溃的情况下生成并上传至云存储服务器的。在接收到客户端401发送的崩溃信号的情况下,客户端401和服务端402会将应用层协议切换为流控制传输协议(Stream Control Transmission Protocol,SCTP)或WebSocket。

[0110] 针对性能侦测:在服务端运行的过程中,服务端402将性能侦测数据格式化,并保存至本地文件中。之后在服务端402结束服务的情况下,将本地文件上传至云存储服务器。云存储服务器中的本地文件用于解析得到性能侦测信息,性能侦测信息用于在用户界面中展示,例如在浏览器中展示。

[0111] 通过在服务端和客户端中集成QUIC-SDK,实现了基于QUIC协议的传输,可提升数据传输的性能。并且,通过第一线程调度第二线程,不同第二线程可实现管理不同的QUIC连接,实现了通过不同线程来管理QUIC连接,使服务端能够充分利用多个核心的并发处理能力,避免了使用单一线程管理QUIC连接导致对并发量瓶颈限制较大的问题,提升了管理QUIC连接的效率,有助于提升数据传输的性能。

[0112] 图5是本申请一个示例性实施例提供的一种数据传输方法的流程示意图。该方法可以用于服务器。如图5所示,该方法包括以下步骤。

[0113] 步骤502:接收客户端发送的UDP数据包。

[0114] 上述服务器中运行的服务端与客户端建立有通信连接,服务端和客户端中集成有QUIC-SDK,QUIC-SDK用于实现基于QUIC协议的传输。服务器接收UDP数据包可指服务器中的应用层模块接收UDP数据包,例如媒体传输模块或业务网关接收UDP数据包,接收UDP数据包的过程是基于QUIC-SDK实现的。

[0115] 示例性地,该UDP数据包包括云游戏的用户对云游戏进行的操作指令,或者包括用于鉴权的数据,或者包括性能侦测数据。

[0116] 步骤504:通过第一线程,在n个第二线程中确定目标第二线程。

[0117] 其中,n为大于1的整数。上述第一线程和第二线程是由QUIC-SDK提供的,QUIC-SDK提供的组件在服务器的不同线程中运行,从而实现提供了第一线程和第二线程。n个第二线程用于处理UDP数据包,不同的第二线程管理了不同的QUIC连接,每个第二线程能够管理一个或多个QUIC连接。该目标第二线程是第一线程确定的用于管理UDP数据包对应的QUIC连接的第二线程。可选地,上述QUIC连接为长连接。

[0118] 第一线程用于接收上述UDP数据包,第一线程在接收到上述UDP数据包,会判断该UDP数据包对应的QUIC连接是否为己存在的连接。在UDP数据包对应的QUIC连接为己存在的连接的情况下,第一线程会确定用于管理该己存在的连接的第二线程作为目标第二线程。在UDP数据包对应的QUIC连接不为己存在的连接的情况下,第一线程会在n个第二线程中按顺序确定目标第二线程,或者按照第二线程管理的QUIC连接的数量,或者随机确定目标第二线程。服务端会为该UDP数据包在目标第二线程中建立相应的QUIC连接,此时目标第二线程会成为用于管理该UDP数据包的QUIC连接的线程。

[0119] 步骤506:通过第一线程将UDP数据包传递至目标第二线程。

[0120] 目标第二线程用于管理该UDP数据包的QUIC连接,通过该QUIC连接能够实现对UDP数据包的解析、处理和发送。可选地,第二线程中的每个线程包括调度组件,该调度组件用

于管理一个或多个会话对象,每个会话对象对应于一个已建立的QUIC连接,以负责管理QUIC连接,对QUIC连接对应的UDP数据包进行处理。第一线程在将UDP数据包传递至目标第二线程,会将UDP数据包传递至目标第二线程的调度组件,目标第二线程的调度组件根据UDP数据包的QUIC连接和会话对象之间的对应关系,将UDP数据包传递至UDP数据包的QUIC连接对应的会话对象,以实现UDP数据包进行处理。

[0121] 综上所述,本实施例提供的方法,通过第一线程调度第二线程,不同第二线程可实现管理不同的QUIC连接,实现了通过不同线程来管理QUIC连接,使服务端能够充分利用多个核心的并发处理能力,避免了使用单一线程管理QUIC连接导致对并发量瓶颈限制较大的问题,提升了管理QUIC连接的效率,有助于提升数据传输的性能。

[0122] 以下实施例对针对数据接收的优化方案进行介绍。

[0123] 图6是本申请一个示例性实施例提供的另一种数据传输方法的流程示意图。该方法可以用于服务器。如图6所示,该方法包括以下步骤。

[0124] 步骤602:接收客户端发送的UDP数据包。

[0125] 上述服务器中运行的服务端与客户端建立有通信连接,服务端和客户端中集成有QUIC-SDK,QUIC-SDK用于实现基于QUIC协议的传输。可选地,服务端通过第一线程进行侦听(UDP listen),以接收客户端发送的UDP数据包。第一线程也可称为主线程,第一线程是由QUIC-SDK提供的。

[0126] 服务端单独使用一个线程进行侦听,并且对于侦听,服务端使用单独的UDP FD。实现第一线程侦听的过程:服务端创建一个UDP FD,对UDP FD设置复用规则,例如包括地址复用(reuse addr)、端口复用(reuse port)属性,然后绑定侦听地址,以使得第一线程调用接收函数(recvfrom())进行UDP数据包的接收。上述复用规则用于使不同线程绑定相同的针对UDP数据包的侦听地址。上述侦听地址指向客户端的发送地址,用于实现侦听客户端发送的数据包,以使得服务端获知其接收的UDP数据包是由客户端发送的,从而进行后续处理,对于非侦听地址发送的数据包不会进行上述处理。

[0127] 步骤604:响应于第一线程接收到UDP数据包,通过第一线程解析UDP数据包得到目标连接标识。

[0128] 目标连接标识(Connection ID)用于标识UDP数据对应的QUIC连接。服务器中的不同第二线程管理了不同的QUIC连接,每个第二线程能够管理一个或多个QUIC连接。通过第一线程根据解析得到的目标连接标识,即可确定用于管理UDP数据包的QUIC连接的第二线程。上述第二线程是由QUIC-SDK提供的。

[0129] 步骤606:通过第一线程,根据目标连接标识在n个第二线程中确定目标第二线程。

[0130] 该目标第二线程是第一线程确定的用于管理UDP数据包对应的QUIC连接的第二线程,n为大于1的整数。第二线程也可称为子线程,可选地,上述QUIC连接为长连接。不同第二线程所管理的QUIC连接与连接标识具有对应关系,通过第一线程,根据目标连接标识即可确定用于管理UDP数据包的QUIC连接的目标第二线程。通过解析连接标识来确定第二线程,提供了一种维护线程和QUIC连接的对应关系的方式,可实现快速确定目标第二线程。

[0131] 第一线程在解析得到UDP数据包的目标连接标识的情况下,会查询该目标连接标识对应的QUIC连接是否为己存在的连接,即查询目标连接标识对应的QUIC连接为新连接(还没连上)还是老连接(已连上了)。

[0132] 针对目标连接标识对应的QUIC连接不为已存在的连接的情况：

在目标连接标识对应的QUIC连接不为已存在的连接的情况下，服务端会通过第一线程在n个第二线程中随机确定目标第二线程。通过根据缓存结合连接标识判断UDP的QUIC连接未存在的情况，随机确定目标第二线程，提供了一种在未建立连接的情况下调度第二线程的方式。

[0133] 针对目标连接标识对应的QUIC连接为已存在的连接的情况：

在目标连接标识对应的QUIC连接为已存在的连接的情况下，服务端通过第一线程根据缓存将用于管理目标连接标识对应的QUIC连接的第二线程确定为目标第二线程。示例地，该缓存是基于开发语言中的map（一种键值对的集合接口）实现的，可称为缓存map。map提供了一个通用的元素存储方法，基于map实现的集合类用于存储元素对（即“键”和“值”），其中每个键映射到一个值。可选地，服务端通过第一线程根据缓存将包括目标连接标识对应的会话对象（QuicSession）的第二线程确定为目标第二线程。其中，该缓存用于存储不同的第二线程中的会话对象和连接标识之间的对应关系，会话对象与QUIC连接对应，用于负责管理其对应的QUIC连接。通过根据缓存结合连接标识来判断UDP的QUIC连接是否已存在，从而根据缓存确定目标第二线程，提供了一种在已建立QUIC连接的情况下，快速选择子线程的方式。

[0134] 步骤608：通过第一线程将UDP数据包传递至目标第二线程。

[0135] 可选地，n个第二线程中的每个线程包括调度组件（dispatcher），该调度组件用于管理一个或多个会话对象，会话对象与QUIC连接对应。服务端通过第一线程将UDP数据包传递至目标第二线程中的调度组件。目标第二线程中的调度组件在接收到UDP数据包后，也会解析出目标连接标识并判断是否为已存在的连接。

[0136] 针对目标连接标识对应的QUIC连接为已存在的连接的情况：

服务端通过目标第二线程中的调度组件根据上述缓存确定目标连接标识对应的会话对象，之后通过目标第二线程中的调度组件将UDP数据包传递至目标连接标识对应的会话对象（会话对象负责目标连接标识对应的QUIC连接），以对UDP数据包进行处理。通过由每个子线程的调度组件来调度QUIC连接会话对象，可实现使每个子线程支持多个QUIC连接，从而提升管理QUIC连接的效率。

[0137] 针对目标连接标识对应的QUIC连接不为已存在的连接的情况：

服务端通过第一线程将UDP数据包传递至目标第二线程中的调度组件，在目标第二线程中的调度组件确定目标连接标识对应的QUIC连接不为已存在的连接的情况下，会建立UDP数据包的QUIC连接，并在目标第二线程中创建UDP数据包的QUIC连接对应的会话对象。之后通过目标第二线程中的调度组件将UDP数据包传递至UDP数据包的QUIC连接对应的会话对象。

[0138] 可选地，在上述过程中，服务端会创建一个新的UDP FD，并对UDP FD设置复用规则，例如包括地址复用（reuse addr）、端口复用（reuse port）属性，然后绑定侦听地址。上述复用规则用于使第一线程和第二线程绑定相同的针对UDP数据包的侦听地址。之后，服务端会对该UDP FD使用连接函数（connect（peer_addr）），连接到对端的地址（peer_addr）上。在创建会话对象时，服务端会基于上述UDP FD在目标第二线程中创建UDP数据包的QUIC连接对应的会话对象（连接对象，例如包括QUIC会话（QuicSession）、QUIC连接

(QuicConnection)、QUIC读取(QuicReader)、QUIC写入(QuicWriter)等),以注册读写事件,之后这个UDP FD的读写操作都在这个QuicSession中处理。其中,UDP FD设置有复用规则,复用规则用于第一线程和第二线程绑定相同的针对UDP数据包的侦听地址。服务端还会在缓存中保存目标第二线程中的会话对象与目标连接标识的对应关系。

[0139] 通过在缓存中存储会话对象与连接标识的对应关系,可实现通过缓存来快速确定QUIC连接和第二线程的对应关系。通过在目标第二线程中建立UDP数据包的QUIC连接,并创建连接相应的会话对象,以实现管理QUIC连接,提供了对未建立的QUIC连接进行建立并对UDP数据包进行调度的具体实现过程。

[0140] 需要说明的是,主线程的UDP FD和子线程的UDP FD都设置了复用规则(reuse addr、reuse port属性),因为若不设置,会导致在Linux上绑定(bind)同一个侦听地址(listen addr)失败。通过对UDP FD设置复用规则,可避免不同UDP FD绑定同一个地址导致地址冲突从而侦听失败。子线程在创建新连接时,要对UDP FD执行connect(peer_addr)。这么做的好处是:会在Linux内核做一个映射(即peer_addr至线程的映射),后续收到peer_addr发来的数据包时,服务端能够优先唤醒对应的线程,然后将UDP数据包分派给这个线程,以提升效率。在服务端支持上述多线程模型后,每个QUIC连接对应一个子线程中的UDP FD,服务端可以使用发送函数(send())和接收函数(recv())来代替发送至函数(sendto(peer_addr))和接收来自函数(recvfrom(src_addr))。这样在收发数据时,无需进行内核寻址,因此减少了一次内核寻址的过程,以降低延迟。

[0141] 综上所述,本实施例提供的方法,通过第一线程调度第二线程,不同第二线程可实现管理不同的QUIC连接,实现了通过不同线程来管理QUIC连接,使服务端能够充分利用多个核心的并发处理能力,避免了使用单一线程管理QUIC连接导致对并发瓶颈限制较大的问题,提升了管理QUIC连接的效率,有助于提升数据传输的性能。

[0142] 以下实施例针对数据发送的优化方案进行介绍。

[0143] 图7是本申请一个示例性实施例提供的又一种数据传输方法的流程示意图。该方法可以用于服务器。如图7所示,该方法包括以下步骤。

[0144] 步骤702:在服务端的应用层传输发送数据的过程中,将发送数据传递至QUIC线程中的发包处理组件。

[0145] 上述服务器中运行的服务端与客户端建立有通信连接,服务端和客户端中集成有QUIC-SDK,QUIC-SDK用于实现基于QUIC协议的传输。上述QUIC线程是由QUIC-SDK提供的线程,QUIC线程具有QUIC-SDK提供的功能,QUIC线程中的组件是由QUIC-SDK提供的。服务端的应用层传输发送数据即服务端的应用层模块进行发送数据的发送。

[0146] 步骤704:通过发包处理组件将发送数据保存至本地发送缓存中。

[0147] 在发包处理组件接收到发送数据时,服务端会将发送数据保存至本地发送缓存(local send buffer)中。本地发送缓存中存储的数据是用于后续进行发送的数据。

[0148] 步骤706:在通过发包处理组件进行数据发送时,发送本地发送缓存中的数据。

[0149] 可选地,在通过发包处理组件进行数据发送时,服务端会将本地发送缓存中全部的待发送数据取出,并进行批量组包,从而发送本地发送缓存中的全部数据。通过一次性发送本地发送缓存中的全部数据,相较于针对数据按照队列逐个发送的方式,能够实现批量发送数据,从而提升数据发送的效率。

[0150] 数据发送的任务是由消息环(MessageLoop)组件进行管理的。消息环也称为消息循环,它是一种等待和分派消息的编程结构,是消息驱动机制的基础。消息环组件能够使所有的任务(例如发送数据的任务)都变成异步驱动。消息环组件用于接收任务,以生成任务列表,并采用循环的方式触发执行任务列表中的任务。可选地,在对发送数据进行传输的过程中,服务端会通过发包处理组件基于消息环组件的发送事件文件描述符(SendEventFD)对消息环组件进行唤醒通知,消息环用于管理传输任务。之后服务端会通过消息环组件来唤醒发包处理组件,以指示发包处理组件进行数据发送,从而实现通过发包处理组件对数据进行发送。

[0151] 在gQUIC库中,数据发送的流程为应用层将数据传递给gQUIC库,由QUIC线程中的消息环组件创建一个异步任务,之后消息环组件按照任务列表的执行唤醒QUIC线程,QUIC线程中的发包处理组件取出异步任务,之后发包处理组件执行发包处理任务,将最终的组包发送到操作系统内核,完成数据发送。

[0152] 本申请实施例提供的方法,通过由发包处理组件获取应用层发送的数据,并通过发送事件文件描述符对消息环组件进行通知,以使消息环唤醒发包处理组件进行数据发送。相较于上述gQUIC的方案,能够无需等待消息环组件的指示,主动对消息环组件进行唤醒以执行发送数据的任务,实现简化从投递发送任务到执行发送任务的流程,提升数据发送的效率。

[0153] 需要说明的是,本申请实施例中内部的事件通知使用EventFD(即SendEventFD),而非使用管道文件描述符(PipeFD)。EventFD和PipeFD均用于进行事件通知,示例地,对于操作系统内核来说,EventFD的开销更低,EventFD只需要创建一个虚拟文件,而PipeFD需要创建两个。并且,EventFD可用于多路复用模型中,从而实现异步的信号通知功能。因此,相较于PipeFD,使用EventFD能够提升通知的效率。

[0154] 并且,本申请实施例还对一次循环(Loop)处理消息的个数进行提升。例如,在gQUIC库中,当存在两个发送数据的任务的情况下,消息环组件通知发包处理组件进行发送时,默认每次只通知执行一个数据的发送任务,对于两个数据的发送需要通知两次,发包处理组件执行一次发送后需要等待下一次通知。本申请实施例提供的方法中,由于不同发送任务的数据均存储在本地发送缓存中,因此能够实现使消息环组件在对发包处理组件进行的一次通知中,指示执行多个发送任务,即对多个任务的数据进行发送,从而使得发包处理组件能够将本地发送缓存中多个发送任务的数据进行批量组包并进行发送,能够提高数据传输的吞吐量。

[0155] 示例地,图8是本申请一个示例性实施例提供的发送数据的过程的示意图。如图8所示,在服务端的应用层模块801发送数据时,会将数据传递给QUIC库(QUIC线程802),QUIC发包处理组件803会将数据保存到本地发送缓存中,并通知(Notify())消息环组件804的SendEventFD对消息环组件进行唤醒。消息环组件804会为该发送事件在任务列表中建立相应的发送任务(PostTask()),并按照本地事件环(Native Event Loop)对任务列表中的任务进行执行,以唤醒(Wakeup())QUIC发包处理组件803,并指示QUIC发包处理组件803进行数据发送。QUIC发包处理组件803会从本地发送缓存中将待发送数据都取出,批量组包并发送到操作系统内核,从而完成数据的发送。其中,操作系统内核用于实现向数据的接收端发送数据。示例地,该操作系统内核指Linux内核。示例地,在操作系统内核获取到需要发送的

数据后,会将数据转换为数据包,并调用服务器的网卡驱动使网卡获取到需要发送的数据并向网卡指示进行数据发送,从而使服务器的网卡将获取的数据发送至数据的接收端。

[0156] 综上所述,本实施例提供的方法,通过在发送数据的过程中先将发送数据保存至本地缓存再进行发送,相较于对发送数据进行拷贝,之后再进行数据发送的方案能够减小内存开销。

[0157] 以下实施例对针对崩溃分析的优化方案进行介绍。

[0158] 图9是本申请一个示例性实施例提供的再一种数据传输方法的流程示意图。该方法可以用于服务器。如图9所示,该方法包括以下步骤。

[0159] 步骤902:在接收到客户端发送的崩溃信号的情况下,通过云存储服务器获取崩溃文件。

[0160] 上述服务器中运行的服务端与客户端建立有通信连接,服务端和客户端中集成有QUIC-SDK,QUIC-SDK用于实现基于QUIC协议的传输。上述崩溃文件是客户端通过QUIC-SDK监测到运行崩溃的情况下生成并上传至云存储服务器的,可选地,该云存储服务器为COS。服务端通过QUIC-SDK实现对崩溃信号的侦听。

[0161] 可选地,上述崩溃文件为小存储器转储文件(minidump),客户端在执行崩溃时,会生成崩溃时的minidump。此时客户端会将minidump上传至COS,并通知服务端。可选地,崩溃信号包括反映QUIC产生了崩溃的信息以及minidump在COS中的文件路径中的至少一种。

[0162] 在接收到客户端发送的崩溃信号的情况下,客户端和服务端会对应用层协议进行切换。例如,将应用层由QUIC-SDK实现的QUIC协议切换到流控制传输协议(Stream Control Transmission Protocol,SCTP)或全双工通信协议(WebSocket)。SCTP是一种在网络连接的两端之间同时传输多个数据流的协议,WebSocket是一种在单个传输控制协议(Transmission Control Protocol,TCP)连接上进行全双工通信的协议。SCTP和WebSocket是基于连接的,而QUIC-SDK实现的QUIC协议基于用户数据报协议(User Datagram Protocol,UDP),UDP是基于非连接的。基于连接来传输数据稳定可靠,适用于对网络通讯质量要求较高的场景,需要准确无误的传输给对方。而UDP的优点是速度快,但是可能产生丢包,所以适用于对实时性要求较高但是对少量丢包并没有太大要求的场景。本实施例提供的方法,通过在系统崩溃的情况下,切换应用层协议,能够实现在崩溃后切换至稳定协议以保证服务质量。

[0163] 示例性地,上述应用层是开放式系统互联(Open System Interconnection,OSI)模型下的第七层,OSI模型将网络通信的工作分为七层,分别是物理层、数据链路层、网络层、运输层、会话层、表示层和应用层。

[0164] 步骤904:根据崩溃文件确定崩溃的原因和位置。

[0165] 该崩溃文件是客户端通过QUIC-SDK监测到运行崩溃的情况下生成并上传至云存储服务器的。可选地,本申请实施例中的QUIC-SDK(QUIC库)集成了崩溃板(Breakpad)库,用以对运行中的崩溃进行检测。在构建QUIC库时,服务端会将代码中的符号信息卸除(dump)下来,并保存成为符号文件。在向外部分发QUIC库时,分发的是去掉了符号信息的QUIC库,从而减少库文件的大小,以及减少符号信息的泄漏。

[0166] 服务端在获取到崩溃文件后,会根据构建QUIC库时保存的符号文件,来定位崩溃的原因和位置,从而进行崩溃的分析。

[0167] 示例地,图10是本申请一个示例性实施例提供的崩溃分析的过程的示意图。如图10所示,客户端1001向操作系统内核注册QUIC库(QUIC-SDK提供)提供的崩溃信号处理函数(CrashHandler O),之后客户端1001通过该崩溃信号处理函数检测操作系统内核的运行崩溃。在检测到运行崩溃的情况下,客户端1001会生成崩溃文件,并上传至COS1003,并且客户端1001还会向云游戏服务端1002发送崩溃信号。在检测到崩溃时,客户端1001还会将应用层降级到备用通道(SCTP或WebSocket)。云游戏服务端1002会通过COS1003下载崩溃文件,并根据构建系统时保存的符号文件结合崩溃文件,分析崩溃的原因和位置。

[0168] 综上所述,本实施例提供的方法,通过服务器获取在崩溃情况下客户端生成的崩溃文件,以对崩溃情况进行分析,提供了一种简单便捷的分析系统崩溃情况的实现方案。并且崩溃文件经由云存储服务器传输,能够保证传输的稳定性,且便于进行崩溃的历史记录。

[0169] 以下实施例针对性能侦测的优化方案进行介绍。

[0170] 图11是本申请一个示例性实施例提供的还一种数据传输方法的流程示意图。该方法可以用于服务器。如图11所示,该方法包括以下步骤。

[0171] 步骤1102:在服务端运行的过程中,将性能侦测数据格式化,并保存至本地文件中。

[0172] 服务端运行指服务端在为客户提供后台服务,例如用户在使用客户端进行云游戏,服务器在此过程中会运行云游戏。通过将性能侦测数据进行格式化处理,能够便于后续的分析。示例地,格式化处理指通过格式操作使任意类型的数据转换成一个字符串,也即是按照便于读取分析的格式对数据进行处理,使得处理得到的数据的格式满足读取分析的要求,从而提升后续读取分析的效率。可选地,上述格式化处理所采用的格式是开发人员设置的。可选地,性能侦测数据包括基于QUIC协议的收发包数、收发速度等反映基于QUIC协议的传输性能的数据。例如,未处理的基于QUIC协议的发送速度数据为每秒“10000000000”kb(二进制),通过格式化处理,得到的数据为每秒“1024”kb(十进制)。

[0173] 步骤1104:在服务端结束服务的情况下,将本地文件上传至云存储服务器。

[0174] 服务端结束服务指服务端停止为客户提供后台服务,例如用户在客户端中退出,服务器在此过程中会停止运行云游戏。上述云存储服务器中的本地文件用于解析得到性能侦测信息,性能侦测信息是能够直观向用户展示的信息,性能侦测信息用于在用户界面中展示。

[0175] 示例地,图12是本申请一个示例性实施例提供的相关技术中进行性能侦测的过程的示意图。如图12所示,边缘集群1201的服务模块通过使用Prometheus-SDK方法将侦测数据推送到后台的侦测服务器1202,从而在后台收集和存储指标数据。之后通过Grafana在浏览器1203中进行侦测数据的查询和展示。上述方案的缺点是对后台服务(存储服务Prometheus、接收网关PushGateway等)的并发量要求高,机器成本大。但是对于侦测来说,日常的查询频率较低,使用大量的机器实现上述方案对资源的利用效率不高。

[0176] 示例地,图13是本申请一个示例性实施例提供的性能侦测的过程的示意图。如图13所示,在本申请实施例提供的方案中,在服务端1302的程序运行过程中,服务端1302将侦测数据格式化,并保存到本地文件中。在用户退出客户端1301时,服务端1302将日志文件上传到COS1303。在事后查询侦测数据时,通过下载日志文件,然后对日志文件进行解析,能够实现将其直观的展示到Web界面上。

[0177] 综上所述,本实施例提供的方法,通过在服务端运行的过程中,保存性能侦测数据并在结束服务时进行上传,实现了轻量化的进行性能侦测,且通过云存储服务器能够便捷的获取性能侦测数据并分析展示,可降低服务端进行性能侦测的开销。

[0178] 对于基于QUIC协议的传输的收发延迟,本申请实施例对此也提供了优化的方案。图14是本申请一个示例性实施例提供的基于QUIC协议的传输过程的示意图。如图14所示,发送端中存在流量控制、拥塞控制、Pacing环节1401来限制发送的速度,以减少针对对端、网络的拥塞程度。接收端接收到数据包后,会执行延迟确认流程1402,确认当前的接收窗口是否足够,如果不够,会向发送端发送延迟确认,并通知对端降低发送速率。本申请实施例提供的优化方案包括如下几个环节。

[0179] 针对流量控制:示例地,图15是本申请一个示例性实施例提供的流量控制的原理的示意图。如图15所示,发送端1501和接收端1502建立有QUIC连接1503,在QUIC协议下,存在两级流量控制。分别为连接(Connection)级和流(Stream)级,Connection级是对一个QUIC连接进行整体的流量控制,Stream级是对QUIC连接中的每个请求进行流量控制,以减少Stream的相互影响。这样做的好处是支持在应用层对每个连接、Stream设置不同的窗口大小。且对于乱序(不同时序)收到的数据包,接收窗口(反映可流量的大小)仍旧能够向之后的时间滑动。如果发送端的发送窗口大小设置过小,会影响发送速率。比如:如果窗口不满足发送的需求,则发送端会缓存数据,等发送窗口恢复,即满足发送需求后再发送。如果接收端的接收窗口大小设置过小,当接收的数据量超过窗口大小时,会向发送端发送延迟确认,表示接收端没有空间接收了。这样发送端就不会持续发送了,它会等到接收端有足够的接收窗口才发送。

[0180] 本申请实施例提供的QUIC-SDK,相较于gQUIC库,调大了发送端的发送窗口大小,以及调大了接收端的接收窗口大小。调整的过程同时涉及上述Connection级和Stream级,即调整包括调大了发送端的和接收端各自的流窗口大小(Stream Windows Size)和连接窗口大小(Connection Windows Size)。调整的数值是根据收发包的实际情况确定的,也能够根据经验值确定。

[0181] 针对拥塞控制:发送端通过拥塞控制中的探测算法,能够实时探测网络链路中的实时带宽大小,探测的实时带宽大小会限制发送端的发送速度,这样能够避免全局的网络拥塞。示例地,发送端通过探测算法,能够周期性地探测网络的带宽大小,并交替测量一段时间内的带宽极大值和时延极小值,之后发送端根据两者乘积可确定拥塞窗口大小。采用交替测量的原因在于带宽和时延是正相关的,即带宽极大时网络拥塞,时延也会极大,时延极小时网络畅通,带宽也会极小,上述探测方式能够使得拥塞窗口的值与网络的实际容量保持一致。拥塞窗口的大小能够反映网络的拥塞程度,从而以此进行拥塞的优化。本申请实施例提供的QUIC-SDK,相较于gQUIC库,提供了修改的拥塞控制中的探测算法,能够使其返回的当前的网络带宽足够大,例如能够支持探测到100MB/s的带宽。

[0182] 针对Pacing:Pacing是指均速发送,即按照相同的时间间隔(Pacing间隔)发送数据。如果当前需要发送的数据量较大,若瞬间将全部数据发出,则需要很大的传输速度,很可能导致网络拥塞。本申请实施例提供的QUIC-SDK,对于Pacing的实现,是发送端通过探测算法获取到当前的带宽,然后按时间单位(例如10ms/50ms/100ms)来分配可发送的数据量,而不是采用瞬间发送的方式。所以当上述探测算法返回的当前带宽足够大时,Pacing间隔

可以忽略不计了。因此在该情况下发送端在需要发送数据时会马上进行数据的发送,而不是间隔发送。

[0183] 上述方案增加的成本(机器成本、运维成本)几乎忽略不计。并且在QUIC上线后,实际场景下业务网关的连接稳定性比WebSocket提升了33%左右,用户操作指令的发送延迟比SCTP降低13%左右。在弱网环境下,QUIC比WebSocket和SCTP有更显著的体验提升。

[0184] 图16是本申请一个示例性实施例提供的数据传输方法的流程示意图。该方法可以用于客户端。如图16所示,该方法包括以下步骤。

[0185] 步骤1602:向服务端发送UDP数据包。

[0186] 上述客户端与服务器中运行的服务端建立有通信连接,服务端和客户端中集成有QUIC-SDK,QUIC-SDK用于实现基于QUIC协议的传输。示例性地,该UDP数据包包括云游戏的用户对云游戏进行的操作指令,或者包括用于鉴权的数据,或者包括性能检测数据。

[0187] 服务端用于通过第一线程,在n个第二线程中确定目标第二线程,并通过第一线程将UDP数据包传递至目标第二线程。n个第二线程用于处理UDP数据包,目标第二线程用于管理UDP数据包对应的QUIC连接,第一线程和第二线程是由QUIC-SDK提供的,n为大于1的整数。

[0188] 可选地,在客户端的应用层传输发送数据的过程中,将发送数据传递至QUIC线程中的发包处理组件,QUIC线程是由QUIC-SDK提供的。通过发包处理组件将发送数据保存至本地发送缓存中。在通过发包处理组件进行数据发送时,发送本地发送缓存中的数据。需要说明的是,客户端进行发送数据传输的具体过程可参见图7对应的实施例,即图7对应的实施例也能够以客户端为执行主体进行执行,本申请实施例在此不作赘述。

[0189] 综上所述,本实施例提供的方法,通过第一线程调度第二线程,不同第二线程可实现管理不同的QUIC连接,实现了通过不同线程来管理QUIC连接,使服务端能够充分利用多个核心的并发处理能力,避免了使用单一线程管理QUIC连接导致对并发量瓶颈限制较大的问题,提升了管理QUIC连接的效率,有助于提升数据传输的性能。

[0190] 需要说明的是,本申请实施例提供的方法步骤的先后顺序可以进行适当调整,步骤也可以根据情况进行相应增减,任何熟悉本技术领域的技术人员在本申请揭露的技术范围内,可轻易想到变化的方法,都应涵盖在本申请的保护范围之内,因此不再赘述。

[0191] 图17是本申请一个示例性实施例提供的第一种数据传输装置的结构示意图。所述装置中运行的服务端与客户端建立有通信连接,所述服务端和所述客户端中集成有QUIC-SDK,所述QUIC-SDK用于实现基于QUIC协议的传输。如图17所示,该装置包括以下模块。

[0192] 接收模块1701,用于接收所述客户端发送的用户数据报协议UDP数据包。

[0193] 确定模块1702,用于通过第一线程,在n个第二线程中确定目标第二线程,所述n个第二线程用于处理所述UDP数据包,所述目标第二线程用于管理所述UDP数据包对应的QUIC连接,所述第一线程和所述第二线程是由所述QUIC-SDK提供的,n为大于1的整数。

[0194] 传递模块1703,用于通过所述第一线程将所述UDP数据包传递至所述目标第二线程。

[0195] 在一个可选的设计中,所述确定模块1702,用于:

响应于所述第一线程接收到所述UDP数据包,通过所述第一线程解析所述UDP数据包,得到目标连接标识。

[0196] 通过所述第一线程,根据所述目标连接标识在所述n个第二线程中确定目标第二线程。

[0197] 在一个可选的设计中,所述确定模块1702,用于:

在所述目标连接标识对应的QUIC连接为已存在的连接的情况下,通过所述第一线程,根据缓存将包括所述目标连接标识对应的会话对象的第二线程确定为所述目标第二线程。

[0198] 其中,所述缓存用于存储不同的所述第二线程中的会话对象和连接标识之间的对应关系,所述会话对象与所述QUIC连接对应。

[0199] 在一个可选的设计中,所述n个第二线程中的每个线程包括调度组件,所述调度组件用于管理一个或多个所述会话对象。所述传递模块1703,用于:

通过所述第一线程将所述UDP数据包传递至所述目标第二线程中的所述调度组件。

[0200] 所述确定模块1702,用于通过所述目标第二线程中的所述调度组件确定所述目标连接标识对应的会话对象。

[0201] 所述传递模块1703,用于通过所述目标第二线程中的所述调度组件将所述UDP数据包传递至所述目标连接标识对应的会话对象。

[0202] 在一个可选的设计中,所述确定模块1702,用于:

在所述目标连接标识对应的QUIC连接不为已存在的连接的情况下,通过所述第一线程在所述n个第二线程中随机确定所述目标第二线程。

[0203] 在一个可选的设计中,所述n个第二线程中的每个线程包括调度组件,所述调度组件用于管理一个或多个会话对象,所述会话对象与所述QUIC连接对应。如图18所示,所述装置还包括以下模块。

[0204] 所述传递模块1703,用于通过所述第一线程将所述UDP数据包传递至所述目标第二线程中的所述调度组件。

[0205] 建立模块1704,用于在所述目标第二线程中的所述调度组件确定所述目标连接标识对应的QUIC连接不为所述已存在的连接的情况下,建立所述UDP数据包的QUIC连接。

[0206] 所述建立模块1704,用于在所述目标第二线程中创建所述UDP数据包的QUIC连接对应的会话对象。

[0207] 所述传递模块1703,用于通过所述目标第二线程中的所述调度组件将所述UDP数据包传递至所述UDP数据包的QUIC连接对应的会话对象。

[0208] 在一个可选的设计中,如图19所示,所述装置还包括以下模块。

[0209] 保存模块1705,用于在缓存中保存所述目标第二线程中的所述会话对象与所述目标连接标识的对应关系。

[0210] 在一个可选的设计中,所述建立模块1704,用于:

基于UDP文件描述符在所述目标第二线程中创建所述UDP数据包的QUIC连接对应的会话对象。

[0211] 其中,所述UDP文件描述符设置有复用规则,所述复用规则用于所述第一线程和所述第二线程绑定相同的针对所述UDP数据包的侦听地址。

[0212] 在一个可选的设计中,如图20所示,所述装置还包括以下模块。

[0213] 所述传递模块1703,用于在所述服务端的应用层传输发送数据的过程中,将所述发送数据传递至QUIC线程中的发包处理组件,所述QUIC线程是由所述QUIC-SDK提供的。

[0214] 保存模块1705,用于通过所述发包处理组件将所述发送数据保存至本地发送缓存中。

[0215] 发送模块1706,用于在通过所述发包处理组件进行数据发送时,发送所述本地发送缓存中的数据。

[0216] 在一个可选的设计中,所述发送模块1706,用于:
在通过所述发包处理组件进行数据发送时,发送所述本地发送缓存中的全部数据。

[0217] 在一个可选的设计中,如图21所示,所述装置还包括以下模块。

[0218] 通知模块1707,用于通过所述发包处理组件基于发送事件文件描述符对消息环进行唤醒通知,所述消息环用于管理传输任务。

[0219] 唤醒模块1708,用于通过所述消息环唤醒所述发包处理组件以指示所述发包处理组件进行数据发送。

[0220] 在一个可选的设计中,如图22所示,所述装置还包括以下模块。

[0221] 获取模块1709,用于在接收到所述客户端发送的崩溃信号的情况下,通过云存储服务器获取崩溃文件。

[0222] 所述确定模块1702,用于根据所述崩溃文件确定崩溃的原因和位置。

[0223] 其中,所述崩溃文件是所述客户端通过所述QUIC-SDK监测到运行崩溃的情况下生成并上传至所述云存储服务器的。

[0224] 在一个可选的设计中,如图23所示,所述装置还包括以下模块。

[0225] 切换模块1710,用于在接收到所述客户端发送的所述崩溃信号的情况下,对应用层协议进行切换。

[0226] 在一个可选的设计中,如图24所示,所述装置还包括以下模块。

[0227] 保存模块1705,用于在所述服务端运行的过程中,将性能侦测数据格式化,并保存至本地文件中。

[0228] 上传模块1711,用于在所述服务端结束服务的情况下,将所述本地文件上传至云存储服务器。

[0229] 其中,所述云存储服务器中的所述本地文件用于解析得到性能侦测信息,所述性能侦测信息用于在用户界面中展示。

[0230] 图25是本申请一个示例性实施例提供的第九种数据传输装置的结构示意图。所述装置与服务器中运行的服务端建立有通信连接,所述服务端和所述装置中集成有QUIC-SDK,所述QUIC-SDK用于实现基于QUIC协议的传输。如图25所示,该装置包括以下模块。

[0231] 发送模块2501,用于向所述服务端发送UDP数据包。

[0232] 其中,所述服务端用于通过第一线程,在n个第二线程中确定目标第二线程,并通过所述第一线程将所述UDP数据包传递至所述目标第二线程,所述n个第二线程用于处理所述UDP数据包,所述目标第二线程用于管理所述UDP数据包对应的QUIC连接,所述第一线程和所述第二线程是由所述QUIC-SDK提供的,n为大于1的整数。

[0233] 在一个可选的设计中,如图26所示,所述装置还包括以下模块。

[0234] 传递模块2502,用于在所述客户端的应用层传输发送数据的过程中,将所述发送数据传递至QUIC线程中的发包处理组件,所述QUIC线程是由所述QUIC-SDK提供的。

[0235] 保存模块2503,用于通过所述发包处理组件将所述发送数据保存至本地发送缓存中。

[0236] 所述发送模块2501,用于在通过所述发包处理组件进行数据发送时,发送所述本地发送缓存中的数据。

[0237] 需要说明的是:上述实施例提供的数据传输装置,仅以上述各功能模块的划分进行举例说明,实际应用中,可以根据需要而将上述功能分配由不同的功能模块完成,即将设备的内部结构划分成不同的功能模块,以完成以上描述的全部或者部分功能。另外,上述实施例提供的数据传输装置与数据传输方法实施例属于同一构思,其具体实现过程详见方法实施例,这里不再赘述。

[0238] 本申请的实施例还提供了一种计算机设备,该计算机设备包括:处理器和存储器,存储器中存储有至少一条指令、至少一段程序、代码集或指令集,至少一条指令、至少一段程序、代码集或指令集由处理器加载并执行以实现上述各方法实施例提供的数据传输方法。

[0239] 示例地,图27是本申请一个示例性实施例提供的计算机设备的结构示意图。

[0240] 所述计算机设备2700包括中央处理单元(Central Processing Unit,CPU)2701、包括随机存取存储器(Random Access Memory,RAM)2702和只读存储器(Read-Only Memory,ROM)2703的系统存储器2704,以及连接系统存储器2704和中央处理单元2701的系统总线2705。所述计算机设备2700还包括帮助计算机设备内的各个器件之间传输信息的基本输入/输出系统(Input/Output系统,I/O系统)2706,和用于存储操作系统2713、应用程序2714和其他程序模块2715的大容量存储设备2707。

[0241] 所述基本输入/输出系统2706包括有用于显示信息的显示器2708和用于用户输入信息的诸如鼠标、键盘之类的输入设备2709。其中所述显示器2708和输入设备2709都通过连接到系统总线2705的输入输出控制器2710连接到中央处理单元2701。所述基本输入/输出系统2706还可以包括输入输出控制器2710以用于接收和处理来自键盘、鼠标、或电子触控笔等多个其他设备的输入。类似地,输入输出控制器2710还提供输出到显示屏、打印机或其他类型的输出设备。

[0242] 所述大容量存储设备2707通过连接到系统总线2705的大容量存储控制器(未示出)连接到中央处理单元2701。所述大容量存储设备2707及其相关联的计算机可读存储介质为计算机设备2700提供非易失性存储。也就是说,所述大容量存储设备2707可以包括诸如硬盘或者只读光盘(Compact Disc Read-Only Memory,CD-ROM)驱动器之类的计算机可读存储介质(未示出)。

[0243] 不失一般性,所述计算机可读存储介质可以包括计算机存储介质和通信介质。计算机存储介质包括以用于存储诸如计算机可读存储指令、数据结构、程序模块或其他数据等信息的任何方法或技术实现的易失性和非易失性、可移动和不可移动介质。计算机存储介质包括RAM、ROM、可擦除可编程只读寄存器(Erasable Programmable Read Only Memory,EPR0M)、电子抹除式可复写只读存储器(Electrically-Erasable Programmable Read-Only Memory,EEPROM)、闪存或其他固态存储设备、CD-ROM、数字多功能光盘(Digital

Versatile Disc,DVD)或其他光学存储、磁带盒、磁带、磁盘存储或其他磁性存储设备。当然,本领域技术人员可知所述计算机存储介质不局限于上述几种。上述的系统存储器2704和大容量存储设备2707可以统称为存储器。

[0244] 存储器存储有一个或多个程序,一个或多个程序被配置成由一个或多个中央处理单元2701执行,一个或多个程序包含用于实现上述方法实施例的指令,中央处理单元2701执行该一个或多个程序实现上述各个方法实施例提供的方法。

[0245] 根据本申请的各种实施例,所述计算机设备2700还可以通过诸如因特网等网络连接到网络上的远程计算机设备运行。也即计算机设备2700可以通过连接在所述系统总线2705上的网络接口单元2711连接到网络2712,或者说,也可以使用网络接口单元2711来连接到其他类型的网络或远程计算机设备系统(未示出)。

[0246] 所述存储器还包括一个或者一个以上的程序,所述一个或者一个以上程序存储于存储器中,所述一个或者一个以上程序包含用于进行本申请实施例提供的方法中由计算机设备所执行的步骤。

[0247] 本申请实施例中还提供了一种计算机可读存储介质,该可读存储介质中存储有至少一条指令、至少一段程序、代码集或指令集,当该至少一条指令、至少一段程序、代码集或指令集由计算机设备的处理器加载并执行时,实现上述各方法实施例提供的数据传输方法。

[0248] 本申请还提供了一种计算机程序产品或计算机程序,该计算机程序产品或计算机程序包括计算机指令,该计算机指令存储在计算机可读存储介质中。计算机设备的处理器从计算机可读存储介质读取该计算机指令,处理器执行该计算机指令,使得该计算机设备执行上述各方法实施例提供的数据传输方法。

[0249] 本领域普通技术人员可以理解实现上述实施例的全部或部分步骤可以通过硬件来完成,也可以通过程序来指令相关的硬件完成,该程序可以存储于一种计算机可读存储介质中,上述提到的可读存储介质可以是只读存储器,磁盘或光盘等。

[0250] 以上所述仅为本申请的可选实施例,并不用以限制本申请,凡在本申请的精神和原则之内,所作的任何修改、等同切换、改进等,均应包含在本申请的保护范围之内。

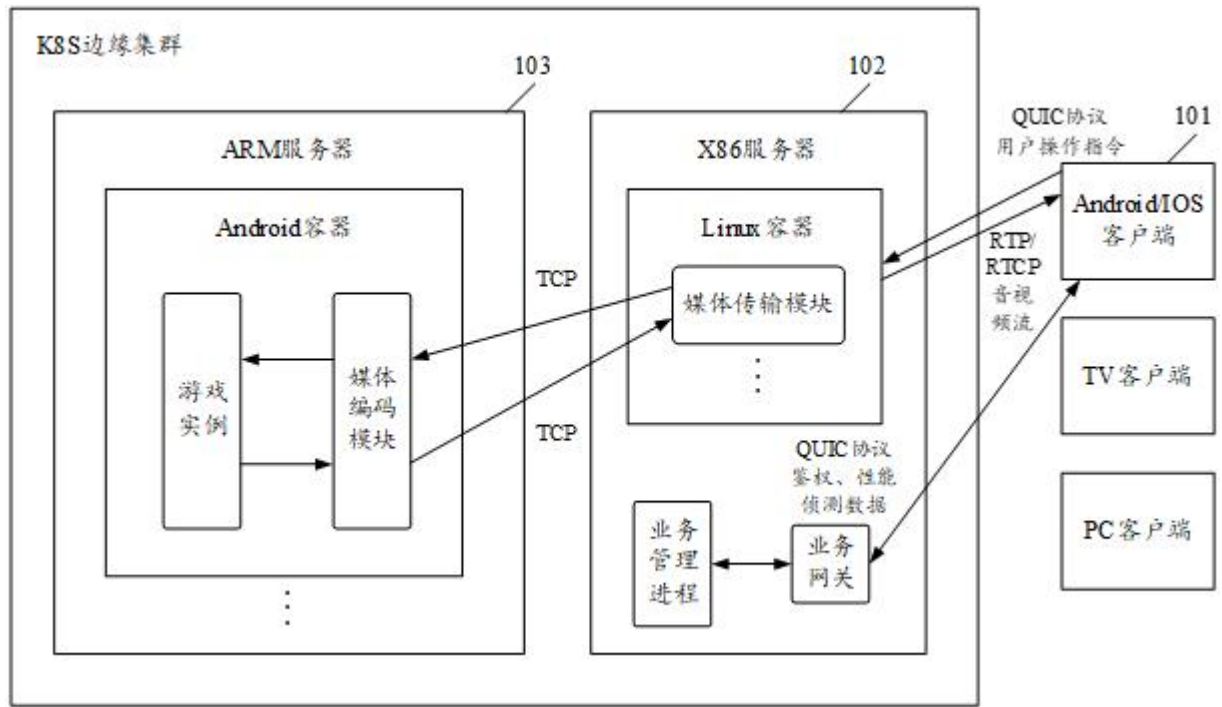


图1

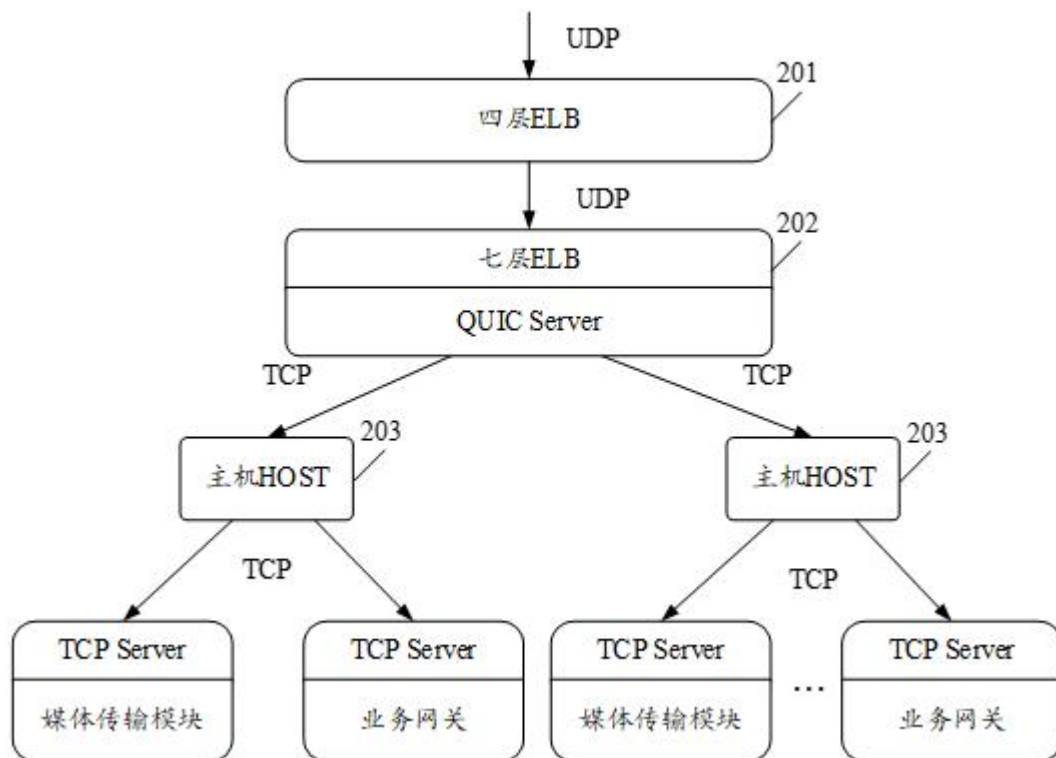


图2

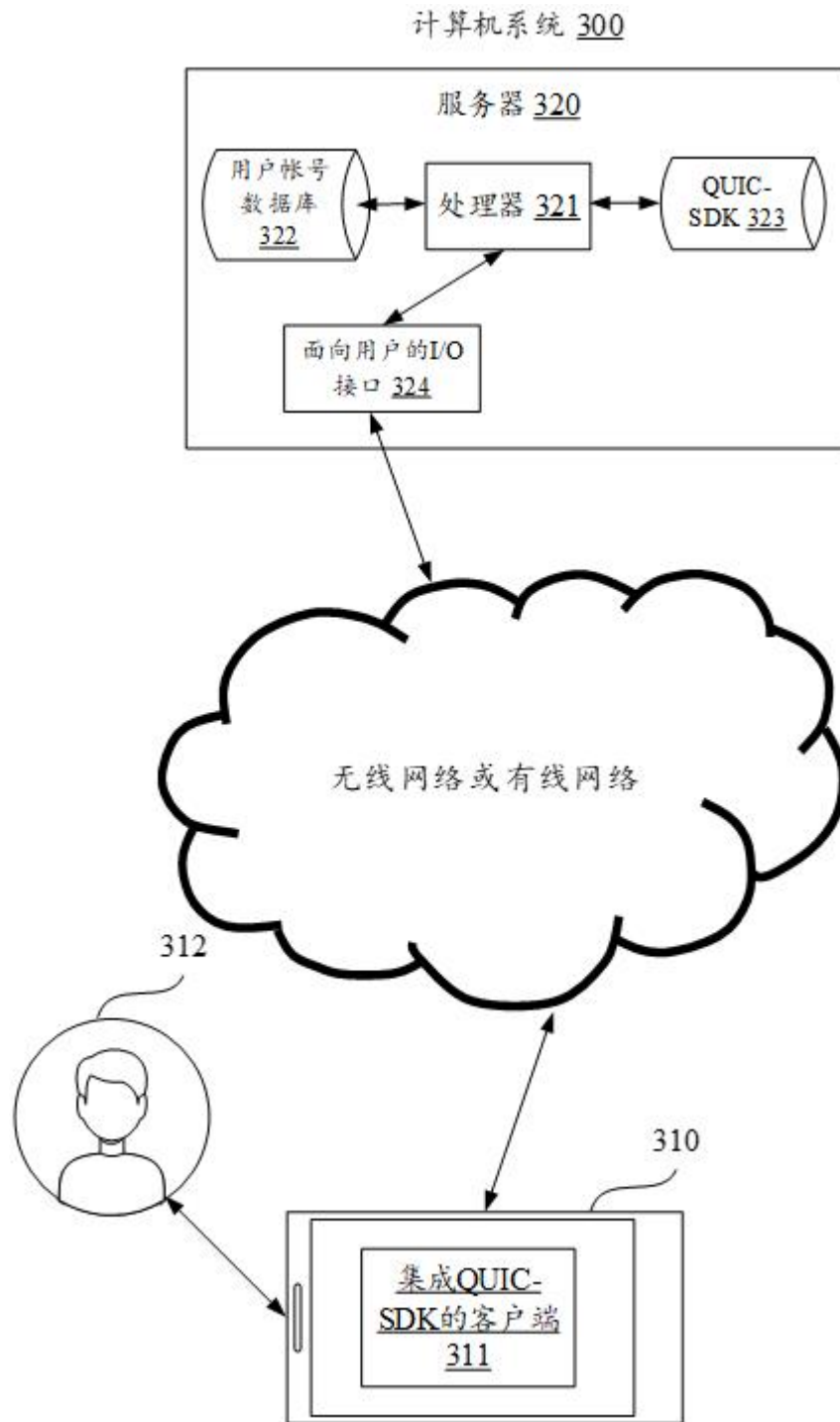


图3

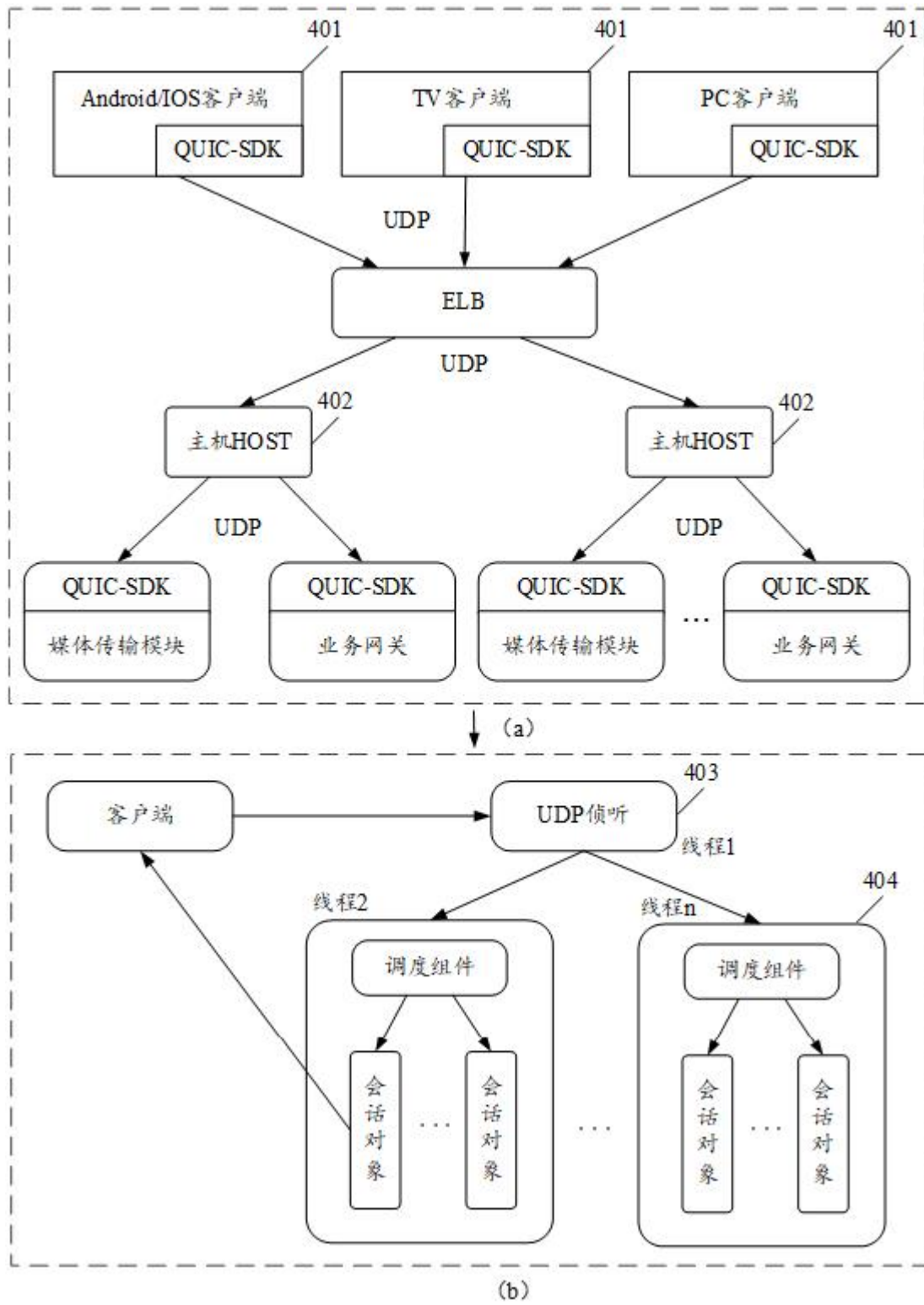


图4

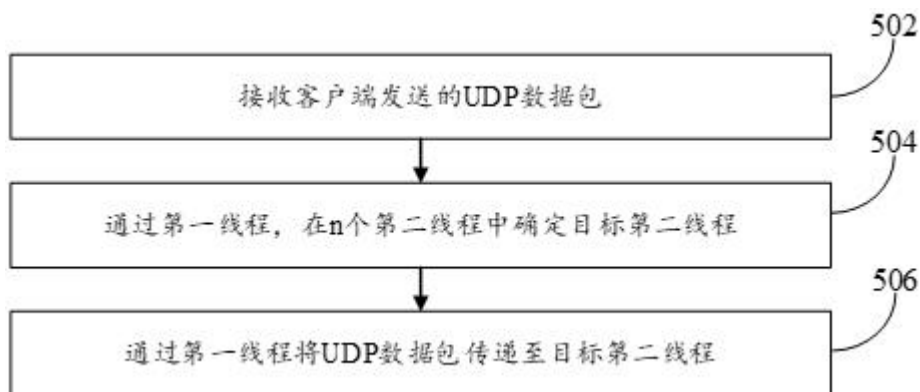


图5

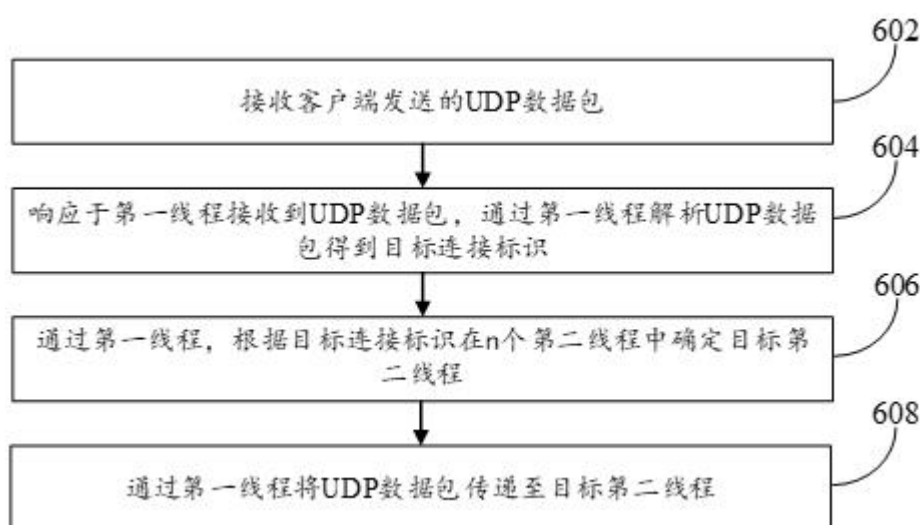


图6

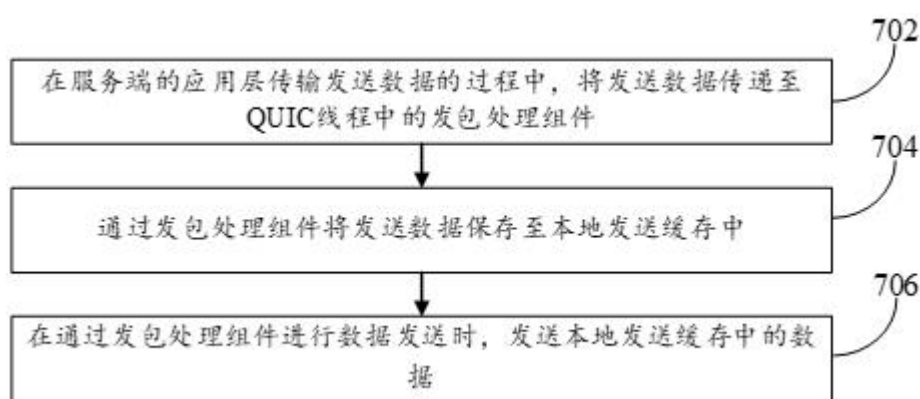


图7

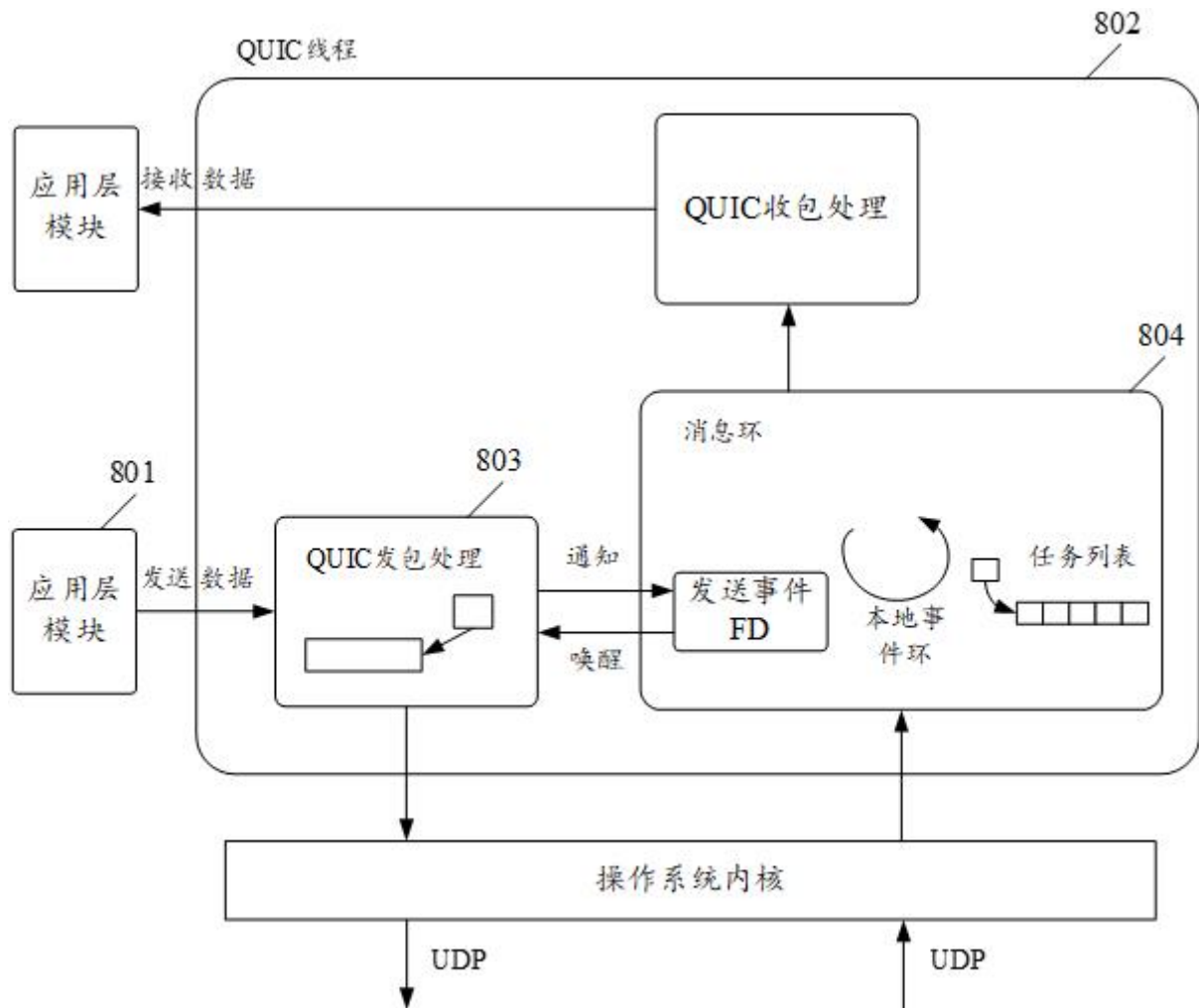


图8

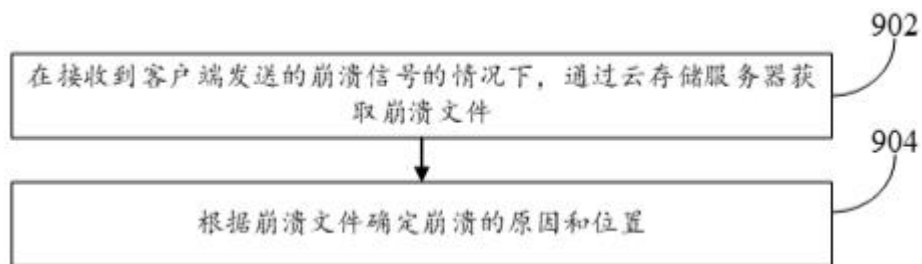


图9

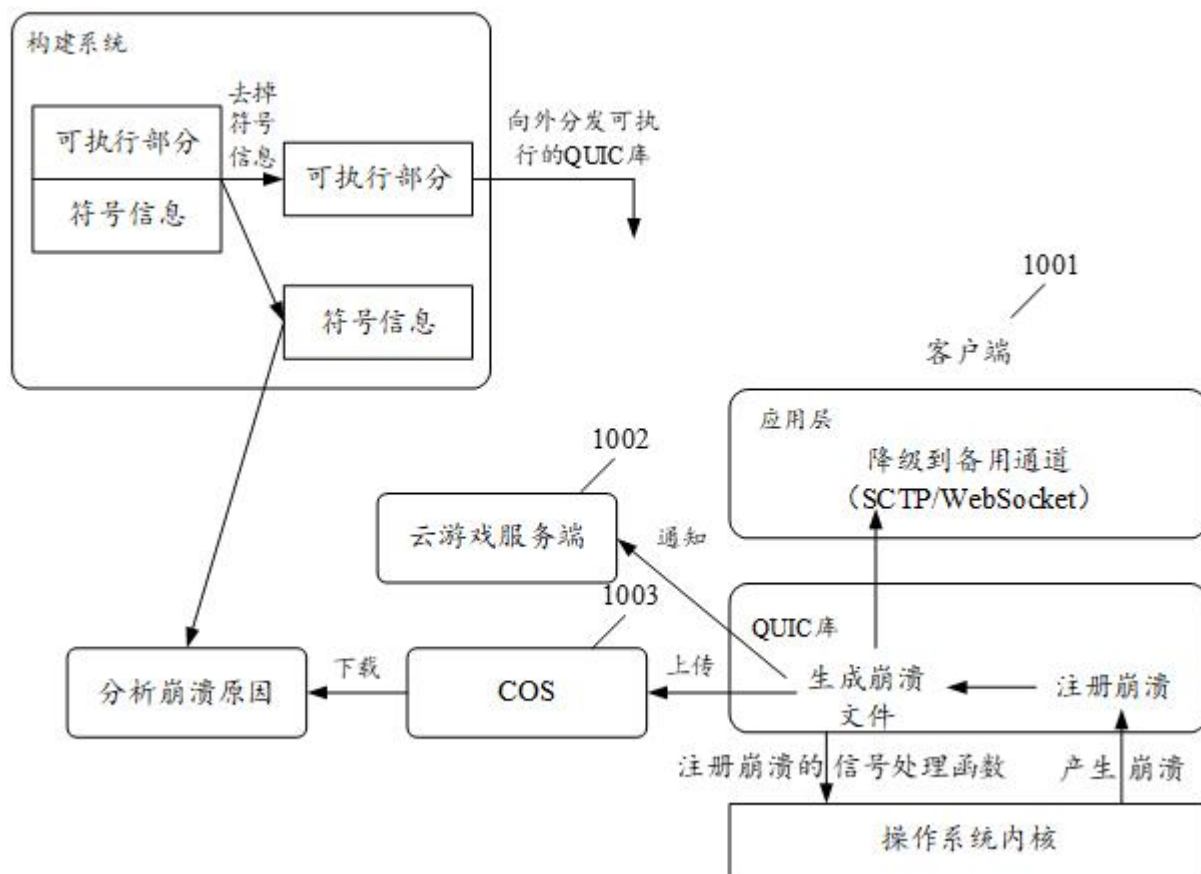


图10



图11

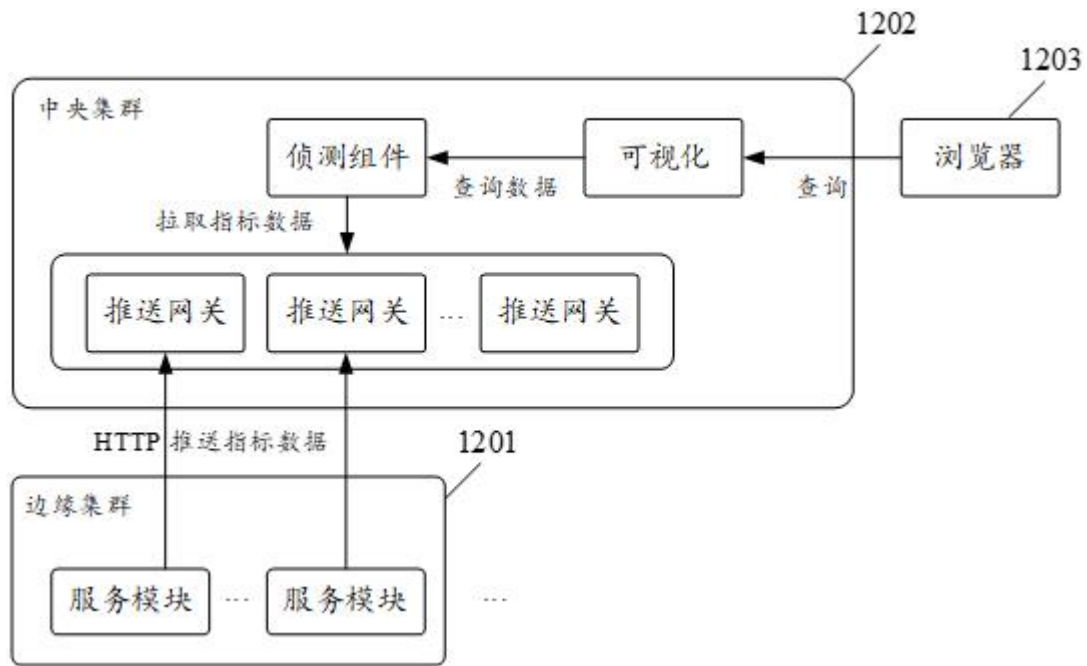


图12

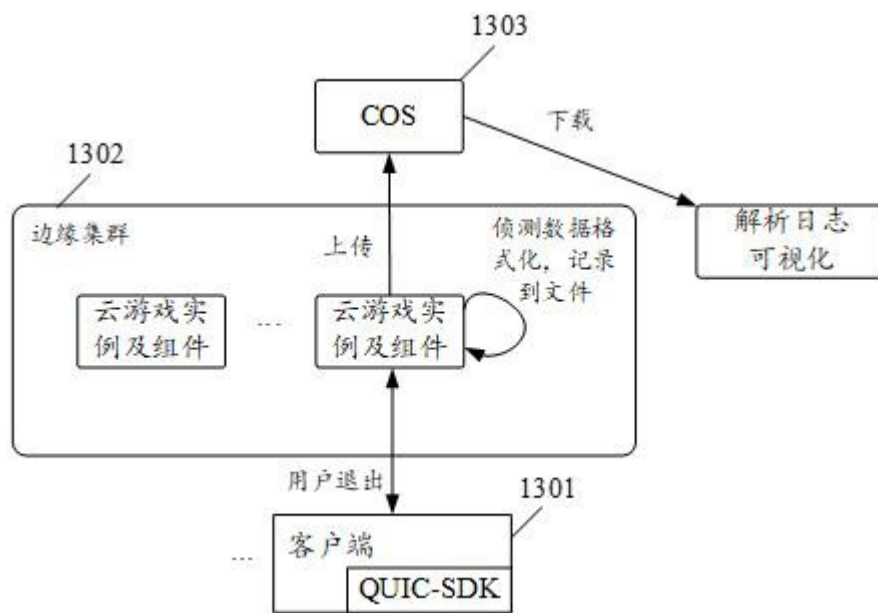


图13

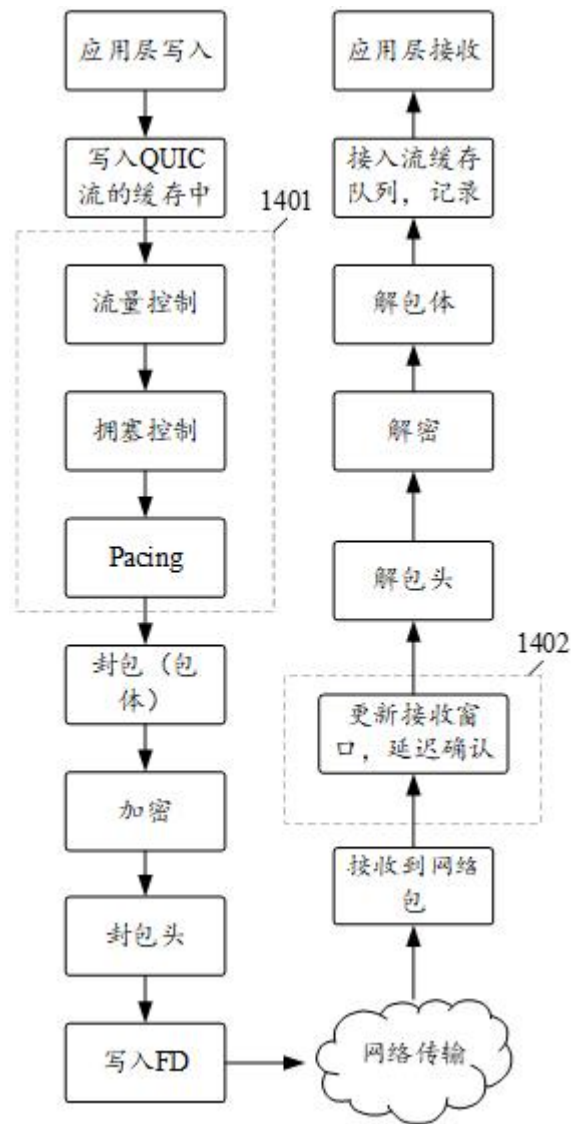


图14

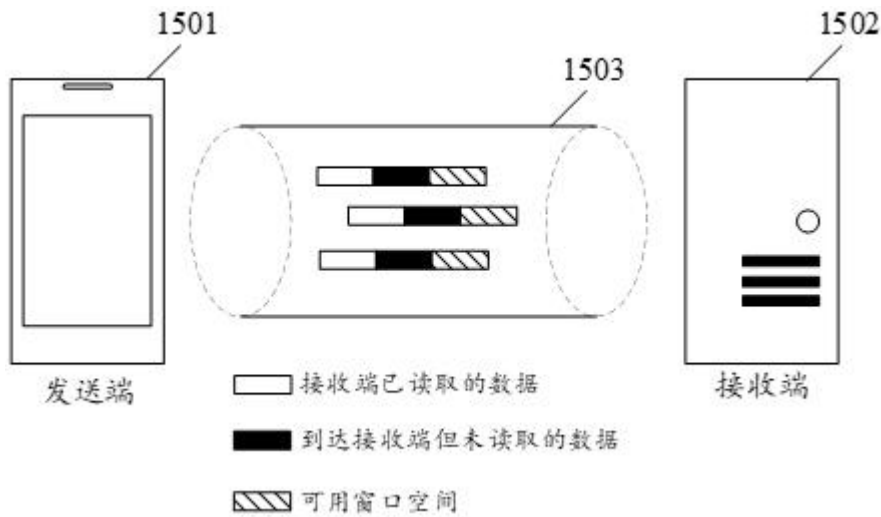


图15

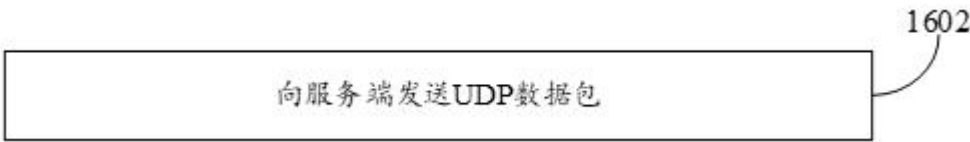


图16

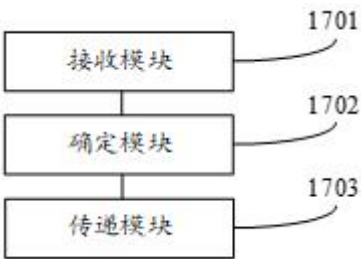


图17

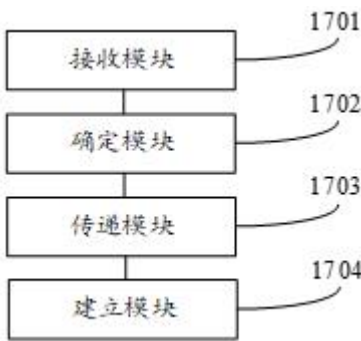


图18



图19



图20



图21

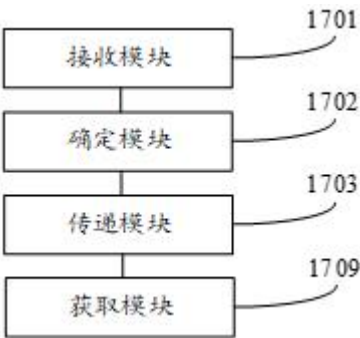


图22

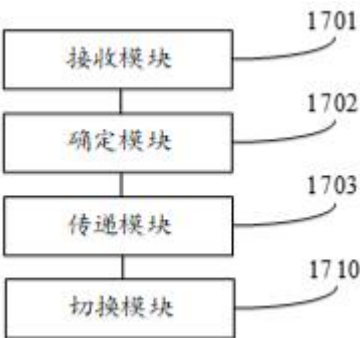


图23



图24

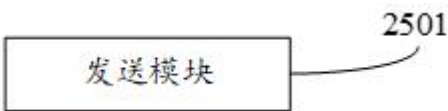


图25



图26

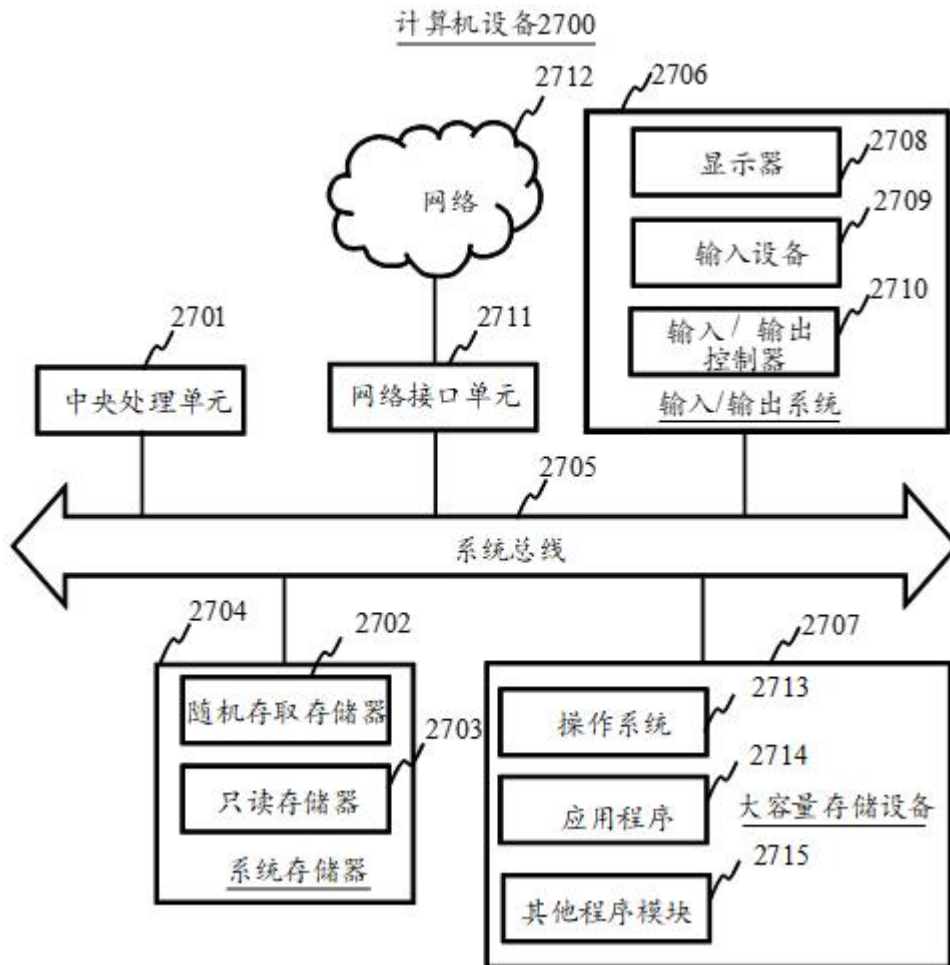


图27