(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 110659080 B
(45)授权公告日 2022.03.15

(21)申请号 201910809940.8
(22)申请日 2019.08.29
(65)同一申请的已公布的文献号
申请公布号 CN 110659080 A
(43)申请公布日 2020.01.07
(73)专利权人 视联动力信息技术股份有限公司
地址 100000 北京市东城区青龙胡同1号歌华大厦A1103-1113
(72)发明人 孙风荣 潘廷勇 韩杰 王艳辉
(74)专利代理机构 北京润泽恒知识产权代理有限公司 11319
代理人 苏培华
(51)Int.Cl.
G06F 9/445(2018.01)
G06F 16/957(2019.01)

审查员 姜宇萱

权利要求书2页 说明书12页 附图4页

(54)发明名称

页面显示方法、装置、电子设备及存储介质

(57)摘要

本发明实施例提供了一种页面显示方法、装置、电子设备及存储介质。所述方法包括：接收在预置浏览器中生成的页面加载请求；调用内嵌于所述预置浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器；由所述图形处理器根据所述页面数据，渲染显示对应的页面。本发明实施例通过内置于浏览器中的eglfs插件直接将页面数据输入到图形处理器中进行加载，从而可以提高页面的加载速度。

CN 110659080 B

1.一种页面显示方法,应用于视联网,其特征在于,所述方法包括:

接收在预置浏览器中生成的页面加载请求;

调用内嵌于所述预置浏览器中的eglfs插件,将所述页面加载请求对应的页面数据写入图形处理器;

由所述图形处理器根据所述页面数据,渲染显示对应的页面;

其中,在所述接收在预置浏览器中生成的页面加载请求的步骤之前,还包括:

获取预先设置于所述预置浏览器中的linuxfb配置参数;

将所述linuxfb配置参数修改为所述eglfs插件对应的eglfs配置参数;

其中,所述预置浏览器为基于QT的初始浏览器配置Linuxfb配置参数或eglfs配置参数的浏览器。

2.根据权利要求1所述的方法,其特征在于,在所述接收在预置浏览器中生成的页面加载请求的步骤之前,还包括:

编译QT源码;

根据所述QT源码编写生成初始浏览器;

在所述初始浏览器中配置linuxfb配置参数或eglfs配置参数,生成所述预置浏览器。

3.根据权利要求1所述的方法,其特征在于,在所述调用内嵌于所述浏览器中的eglfs插件,将所述页面加载请求对应的页面数据写入图形处理器的步骤之前,还包括:

根据所述页面加载请求,获取待加载的页面的链接地址;

根据所述链接地址,获取所述页面数据。

4.根据权利要求3所述的方法,其特征在于,所述由所述图形处理器根据所述页面数据,渲染显示对应的页面的步骤,包括:

由所述图形处理器根据所述页面数据渲染显示所述待加载的页面。

5.一种页面显示装置,应用于视联网,其特征在于,所述装置包括:

加载请求接收模块,用于接收在预置浏览器中生成的页面加载请求;

页面数据写入模块,用于调用内嵌于所述预置浏览器中的eglfs插件,将所述页面加载请求对应的页面数据写入图形处理器;

页面渲染显示模块,用于由所述图形处理器根据所述页面数据,渲染显示对应的页面;

配置参数获取模块,用于获取预先设置于所述预置浏览器中的linuxfb配置参数;

配置参数修改模块,用于将所述linuxfb配置参数修改为所述eglfs插件对应的eglfs配置参数;

其中,所述预置浏览器为基于QT的初始浏览器中配置Linuxfb配置参数或eglfs配置参数。

6.根据权利要求5所述的装置,其特征在于,还包括:

QT源码编译模块,用于编译QT源码;

初始浏览器生成模块,用于根据所述QT源码编写生成初始浏览器;

预置浏览器生成模块,用于在所述初始浏览器中配置linuxfb配置参数或eglfs配置参数,生成所述预置浏览器。

7.一种电子设备,包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序,其特征在于,所述处理器执行所述计算机程序时实现权利要求1至4任一所述的页

面显示方法。

8.一种计算机可读存储介质,其特征在于,所述计算机可读存储介质存储有执行权利要求1至4任一所述的页面显示方法的计算机程序。

页面显示方法、装置、电子设备及存储介质

技术领域

[0001] 本申请涉及页面显示技术领域，特别是涉及一种页面显示方法、装置、电子设备及存储介质。

背景技术

[0002] 随着视联网的快速发展，基于视联网的视频会议、视频教学等在用户的生活、工作、学习等方面广泛普及。

[0003] 目前在应用视联网的过程中，在渲染显示一个Html页面时，通常是采用基于QT浏览器内置的linuxfb插件，linuxfb是通过linux的fb设备结点，直接向framebuffer内写入数据，而采用linuxfb的加载方式，页面图片渲染较慢，用户所看到的现象是如从A页面切换到B页面时图片加载是一行一行的显示，严重影响视觉效果。

发明内容

[0004] 鉴于上述问题，提出了本申请实施例以便提供一种克服上述问题或者至少部分地解决上述问题的一种页面显示方法、装置、电子设备及存储介质。

[0005] 第一方面，本申请实施例公开了一种页面显示方法，应用于视联网，所述方法包括：

[0006] 接收在预置浏览器中生成的页面加载请求；

[0007] 调用内嵌于所述预置浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器；

[0008] 由所述图形处理器根据所述页面数据，渲染显示对应的页面。

[0009] 优选地，在所述接收在预置浏览器中生成的页面加载请求的步骤之前，还包括：

[0010] 获取预先设置于所述预置浏览器中的linuxfb配置参数；

[0011] 将所述linuxfb配置参数修改为所述eglfs插件对应的eglfs配置参数。

[0012] 优选地，在所述接收在预置浏览器中生成的页面加载请求的步骤之前，还包括：

[0013] 编译QT源码；

[0014] 根据所述QT源码编写生成初始浏览器；

[0015] 在所述初始浏览器中配置linuxfb配置参数或eglfs配置参数，生成所述预置浏览器。

[0016] 优选地，在所述调用内嵌于所述浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器的步骤之前，还包括：

[0017] 根据所述页面加载请求，获取待加载的页面的链接地址；

[0018] 根据所述链接地址，获取所述页面数据。

[0019] 优选地，所述由所述图形处理器根据所述页面数据，渲染显示对应的页面的步骤，包括：

[0020] 由所述图形处理器根据所述页面数据渲染显示所述待加载的页面。

[0021] 第二方面，本申请实施例公开了一种页面显示装置，应用于视联网，所述装置包括：

[0022] 加载请求接收模块，用于接收在预置浏览器中生成的页面加载请求；

[0023] 页面数据写入模块，用于调用内嵌于所述预置浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器；

[0024] 页面渲染显示模块，用于由所述图形处理器根据所述页面数据，渲染显示对应的页面。

[0025] 优选地，还包括：

[0026] 配置参数获取模块，用于获取预先设置于所述预置浏览器中的linuxfb配置参数；

[0027] 配置参数修改模块，用于将所述linuxfb配置参数修改为所述eglfs插件对应的eglfs配置参数。

[0028] 优选地，还包括：

[0029] QT源码编译模块，用于编译QT源码；

[0030] 初始浏览器生成模块，用于根据所述QT源码编写生成初始浏览器；

[0031] 预置浏览器生成模块，用于在所述初始浏览器中配置linuxfb配置参数或eglfs配置参数，生成所述预置浏览器。

[0032] 优选地，还包括：

[0033] 链接地址获取模块，用于根据所述页面加载请求，获取待加载的页面的链接地址；

[0034] 页面数据获取模块，用于根据所述链接地址，获取所述页面数据。

[0035] 优选地，所述页面渲染显示模块包括：

[0036] 页面渲染显示子模块，用于由所述图形处理器根据所述页面数据渲染显示所述待加载的页面。

[0037] 第三方面，本申请实施例还公开了一种计算机设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序，所述处理器执行所述计算机程序时实现上述第一方面任一项所述的页面显示方法。

[0038] 第四方面，本申请实施例还公开了一种计算机可读存储介质，所述计算机可读存储介质存储有执行上述第一方面任一项所述的页面显示方法的计算机程序。

[0039] 本申请实施例提供的页面显示方法、装置、电子设备及存储介质，通过接收在预置浏览器中生成的页面加载请求，调用内嵌于预置浏览器中的eglfs插件，将页面加载请求对应的页面数据写入图形处理器，由图形处理器根据页面数据，渲染显示对应的页面。本发明实施例通过内置于浏览器中的eglfs插件直接将页面数据输入到图形处理器中进行加载，从而可以提高页面的加载速度。

附图说明

[0040] 图1是本申请实施例提供的一种视联网的组网示意图；

[0041] 图2是本申请实施例提供的一种节点服务器的硬件结构示意图；

[0042] 图3是本申请实施例提供的一种接入交换机的硬件结构示意图；

[0043] 图4是本申请实施例提供的一种以太网协转网关的硬件结构示意图；

[0044] 图5是本申请实施例提供的页面显示方法的步骤流程图；

[0045] 图6是本申请实施例提供的页面显示方法的步骤流程图；

[0046] 图7是本申请实施例提供的页面显示装置的结构示意图。

具体实施方式

[0047] 为使本申请的上述目的、特征和优点能够更加明显易懂，下面结合附图和具体实施方式对本申请作进一步详细的说明。

[0048] 视联网是网络发展的重要里程碑，是一个实时网络，能够实现高清视频实时传输，将众多互联网应用推向高清视频化，高清面对面。

[0049] 视联网采用实时高清视频交换技术，可以在一个网络平台上将所需的服务，如高清视频会议、视频监控、智能化监控分析、应急指挥、数字广播电视、延时电视、网络教学、现场直播、VOD点播、电视邮件、个性录制(PVR)、内网(自办)频道、智能化视频播控、信息发布等数十种视频、语音、图片、文字、通讯、数据等服务全部整合在一个系统平台，通过电视或电脑实现高清品质视频播放。

[0050] 为使本领域技术人员更好地理解本申请实施例，以下对视联网进行介绍：

[0051] 视联网所应用的部分技术如下所述：

[0052] 网络技术(Network Technology)

[0053] 视联网的网络技术创新改良了传统以太网(Ethernet)，以面对网络上潜在的巨大视频流量。不同于单纯的网络分组交换(Packet Switching)或网络电路交换(Circuit Switching)，视联网技术采用网络分组交换满足Streaming(译为成流、流、连续播送，是一种数据传送技术，把收到的数据变成一个稳定连续的流，源源不断地送出，使用户听到的声音或看到的图像十分平稳，而且用户在整个数据传送完之前就可以开始在屏幕上进行浏览)需求。视联网技术具备分组交换的灵活、简单和低价，同时具备电路交换的品质和安全保证，实现了全网交换式虚拟电路，以及数据格式的无缝连接。

[0054] 交换技术(Switching Technology)

[0055] 视联网采用以太网的异步和包交换两个优点，在全兼容的前提下消除了以太网缺陷，具备全网端到端无缝连接，直通用户终端，直接承载IP数据包。用户数据在全网范围内不需任何格式转换。视联网是以太网的更高级形态，是一个实时交换平台，能够实现目前互联网无法实现的全网大规模高清视频实时传输，将众多网络视频应用推向高清化、统一化。

[0056] 服务器技术(Server Technology)

[0057] 视联网和统一视频平台上的服务器技术不同于传统意义上的服务器，它的流媒体传输是建立在面向连接的基础上，其数据处理能力与流量、通讯时间无关，单个网络层就能够包含信令及数据传输。对于语音和视频业务来说，视联网和统一视频平台流媒体处理的复杂度比数据处理简单许多，效率比传统服务器大大提高了百倍以上。

[0058] 储存器技术(Storage Technology)

[0059] 统一视频平台的超高速储存器技术为了适应超大容量和超大流量的媒体内容而采用了最先进的实时操作系统，将服务器指令中的节目信息映射到具体的硬盘空间，媒体内容不再经过服务器，瞬间直接送达到用户终端，用户等待一般时间小于0.2秒。最优化的扇区分布大大减少了硬盘磁头寻道的机械运动，资源消耗仅占同等级IP互联网的20％，但产生大于传统硬盘阵列3倍的并发流量，综合效率提升10倍以上。

[0060] 网络安全技术(Network Security Technology)

[0061] 视联网的结构性设计通过每次服务单独许可制、设备与用户数据完全隔离等方式从结构上彻底根除了困扰互联网的网络安全问题，一般不需要杀毒程序、防火墙，杜绝了黑客与病毒的攻击，为用户提供结构性的无忧安全网络。

[0062] 服务创新技术(Service Innovation Technology)

[0063] 统一视频平台将业务与传输融合在一起，不论是单个用户、私网用户还是一个网络的总合，都不过是一次自动连接。用户终端、机顶盒或PC直接连到统一视频平台，获得丰富多彩的各种形态的多媒体视频服务。统一视频平台采用“菜谱式”配表模式来替代传统的复杂应用编程，可以使用非常少的代码即可实现复杂的应用，实现“无限量”的新业务创新。

[0064] 视联网的组网如下所述：

[0065] 视联网是一种集中控制的网络结构，该网络可以是树型网、星型网、环状网等等类型，但在此基础上网络中需要有集中控制节点来控制整个网络。

[0066] 如图1所示，视联网分为接入网和城域网两部分。

[0067] 接入网部分的设备主要可以分为3类：节点服务器，接入交换机，终端(包括各种机顶盒、编码板、存储器等)。节点服务器与接入交换机相连，接入交换机可以与多个终端相连，并可以连接以太网。

[0068] 其中，节点服务器是接入网中起集中控制功能的节点，可控制接入交换机和终端。节点服务器可直接与接入交换机相连，也可以直接与终端相连。

[0069] 类似的，城域网部分的设备也可以分为3类：城域服务器，节点交换机，节点服务器。城域服务器与节点交换机相连，节点交换机可以与多个节点服务器相连。

[0070] 其中，节点服务器即为接入网部分的节点服务器，即节点服务器既属于接入网部分，又属于城域网部分。

[0071] 城域服务器是城域网中起集中控制功能的节点，可控制节点交换机和节点服务器。城域服务器可直接连接节点交换机，也可直接连接节点服务器。

[0072] 由此可见，整个视联网络是一种分层集中控制的网络结构，而节点服务器和城域服务器下控制的网络可以是树型、星型、环状等各种结构。

[0073] 形象地称，接入网部分可以组成统一视频平台(圈中部分)，多个统一视频平台可以组成视联网；每个统一视频平台可以通过城域以及广域视联网互联互通。

[0074] 视联网设备分类

[0075] 1.1本申请实施例的视联网中的设备主要可以分为3类：服务器，交换机(包括以太网网关)，终端(包括各种机顶盒，编码板，存储器等)。视联网整体上可以分为城域网(或者国家网、全球网等)和接入网。

[0076] 1.2其中接入网部分的设备主要可以分为3类：节点服务器，接入交换机(包括以太网网关)，终端(包括各种机顶盒，编码板，存储器等)。

[0077] 各接入网设备的具体硬件结构为：

[0078] 节点服务器：

[0079] 如图2所示，主要包括网络接口模块201、交换引擎模块202、CPU模块203、磁盘阵列模块204。

[0080] 其中，网络接口模块201，CPU模块203、磁盘阵列模块204进来的包均进入交换引擎

模块202;交换引擎模块202对进来的包进行查地址表205的操作,从而获得包的导向信息;并根据包的导向信息把该包存入对应的包缓存器206的队列;如果包缓存器206的队列接近满,则丢弃;交换引擎模202轮询所有包缓存器队列,如果满足以下条件进行转发:1)该端口发送缓存未满;2)该队列包计数器大于零。磁盘阵列模块204主要实现对硬盘的控制,包括对硬盘的初始化、读写等操作;CPU模块203主要负责与接入交换机、终端(图中未示出)之间的协议处理,对地址表205(包括下行协议包地址表、上行协议包地址表、数据包地址表)的配置,以及,对磁盘阵列模块204的配置。

[0081] 接入交换机:

[0082] 如图3所示,主要包括网络接口模块(下行网络接口模块301、上行网络接口模块302)、交换引擎模块303和CPU模块304。

[0083] 其中,下行网络接口模块301进来的包(上行数据)进入包检测模块305;包检测模块305检测包的目的地址(DA)、源地址(SA)、数据包类型及包长度是否符合要求,如果符合,则分配相应的流标识符(stream-id),并进入交换引擎模块303,否则丢弃;上行网络接口模块302进来的包(下行数据)进入交换引擎模块303;CPU模块304进来的数据包进入交换引擎模块303;交换引擎模块303对进来的包进行查地址表306的操作,从而获得包的导向信息;如果进入交换引擎模块303的包是下行网络接口往上行网络接口去的,则结合流标识符(stream-id)把该包存入对应的包缓存器307的队列;如果该包缓存器307的队列接近满,则丢弃;如果进入交换引擎模块303的包不是下行网络接口往上行网络接口去的,则根据包的导向信息,把该数据包存入对应的包缓存器307的队列;如果该包缓存器307的队列接近满,则丢弃。

[0084] 交换引擎模块303轮询所有包缓存器队列,在本申请实施例中分两种情形:

[0085] 如果该队列是下行网络接口往上行网络接口去的,则满足以下条件进行转发:1)该端口发送缓存未满;2)该队列包计数器大于零;3)获得码率控制模块产生的令牌。

[0086] 如果该队列不是下行网络接口往上行网络接口去的,则满足以下条件进行转发:1)该端口发送缓存未满;2)该队列包计数器大于零。

[0087] 码率控制模块208是由CPU模块204来配置的,在可编程的间隔内对所有下行网络接口往上行网络接口去的包缓存器队列产生令牌,用以控制上行转发的码率。

[0088] CPU模块304主要负责与节点服务器之间的协议处理,对地址表306的配置,以及,对码率控制模块308的配置。

[0089] 以太网协转网关:

[0090] 如图4所示,主要包括网络接口模块(下行网络接口模块401、上行网络接口模块402)、交换引擎模块403、CPU模块404、包检测模块405、码率控制模块408、地址表406、包缓存器407和MAC添加模块409、MAC删除模块410。

[0091] 其中,下行网络接口模块401进来的数据包进入包检测模块405;包检测模块405检测数据包的以太网MAC DA、以太网MAC SA、以太网length or frame type、视联网目的地址DA、视联网源地址SA、视联网数据包类型及包长度是否符合要求,如果符合则分配相应的流标识符(stream-id);然后,由MAC删除模块410减去MAC DA、MAC SA、length or frame type(2byte),并进入相应的接收缓存,否则丢弃;

[0092] 下行网络接口模块401检测该端口的发送缓存,如果有包则根据包的视联网目的

地址DA获知对应的终端的以太网MAC DA,添加终端的以太网MAC DA、以太网协转网关的MAC SA、以太网length or frame type,并发送。

[0093] 以太网协转网关中其他模块的功能与接入交换机类似。

[0094] 终端:

[0095] 主要包括网络接口模块、业务处理模块和CPU模块;例如,机顶盒主要包括网络接口模块、视音频编解码引擎模块、CPU模块;编码板主要包括网络接口模块、视音频编码引擎模块、CPU模块;存储器主要包括网络接口模块、CPU模块和磁盘阵列模块。

[0096] 1.3城域网部分的设备主要可以分为3类:节点服务器,节点交换机,城域服务器。其中,节点交换机主要包括网络接口模块、交换引擎模块和CPU模块;城域服务器主要包括网络接口模块、交换引擎模块和CPU模块构成。

[0097] 2、视联网数据包定义

[0098] 2.1接入网数据包定义

[0099] 接入网的数据包主要包括以下几部分:目的地址(DA)、源地址(SA)、保留字节、payload(PDU)、CRC。

[0100] 如下表所示,接入网的数据包主要包括以下几部分:

[0101]

DA	SA	Reserved	Payload	CRC

[0102] 目的地址(DA)由8个字节(byte)组成,第一个字节表示数据包的类型(例如各种协议包、组播数据包、单播数据包等),最多有256种可能,第二字节到第六字节为城域网地址,第七、第八字节为接入网地址。

[0103] 源地址(SA)也是由8个字节(byte)组成,定义与目的地址(DA)相同。

[0104] 保留字节由2个字节组成。

[0105] payload部分根据不同的数据报的类型有不同的长度,如果数据报的类型是各种协议包,则payload部分的长度是64个字节,如果数据报的类型是单组播数据包,则payload部分的长度是32+1024=1056个字节,当然并不仅仅限于以上2种。

[0106] CRC有4个字节组成,其计算方法遵循标准的以太网CRC算法。

[0107] 2.2城域网数据包定义

[0108] 城域网的拓扑是图型,两个设备之间可能有2种、甚至2种以上的连接,即节点交换机和节点服务器、节点交换机和节点交换机、节点交换机和节点服务器之间都可能超过2种连接。但是,城域网设备的城域网地址却是唯一的,为了精确描述城域网设备之间的连接关系,在本申请实施例中引入参数:标签,来唯一描述一个城域网设备。

[0109] 本说明书中标签的定义和多协议标签交换(Multi-Protocol Label Switch,MPLS)的标签的定义类似,假设设备A和设备B之间有两个连接,那么数据包从设备A到设备B就有2个标签,数据包从设备B到设备A也有2个标签。标签分入标签、出标签,假设数据包进入设备A的标签(入标签)是0x0000,这个数据包离开设备A时的标签(出标签)可能就变成了0x0001。城域网的入网流程是集中控制下的入网过程,也就意味着城域网的地址分配、标签分配都是由城域服务器主导的,节点交换机、节点服务器都是被动的执行而已,这一点与MPLS的标签分配是不同的,MPLS的标签分配是交换机、服务器互相协商的结果。

[0110] 如下表所示,城域网的数据包主要包括以下几部分:

[0111]

DA	SA	Reserved	标签	Payload	CRC

[0112] 即目的地址(DA)、源地址(SA)、保留字节(Reserved)、标签、payload(PDU)、CRC。其中,标签的格式可以参考如下定义:标签是32bit,其中高16bit保留,只用低16bit,它的位置是在数据包的保留字节和payload之间。

[0113] 网络接口控制器(英语:network interface controller,NIC),又称网络接口控制器,网络适配器(network adapter),网卡(network interface card),或局域网接收器(LAN adapter),是一块被设计用来允许计算机在计算机网络上进行通讯的计算机硬件。由于其拥有MAC地址,因此属于OSI模型的第1层。它使得用户可以通过电缆或无线相互连接。每一个网卡都有一个被称为MAC地址的独一无二的48位串行号,它被写在卡上的一块ROM中。在网络上的每一个计算机都必须拥有一个独一无二的MAC地址。没有任何两块被生产出来的网卡拥有同样的地址。

[0114] 计算机与外界局域网的连接是通过主机箱内插入一块网络接口板(或者是在笔记本电脑中插入一块PCMCIA卡)。

[0115] 从视联网网卡中抓取的数据包相比于从互联网网卡中抓取的数据包有其自己的特征,因此本申请实施例通过依次对PC当前所有可用网卡抓取一定数量的数据包,分析这些数据包,判定其中具有视联网数据包特征的数据包占总数据的百分比最大的,即为视联网网卡。

[0116] 本申请实施例是在既有视联网网卡,又有互联网网卡及其它网卡的情况下,准确选择出视联网网卡。视联网网卡的选择的目的是找到任意一个能抓到视联网数据的网卡即可。

[0117] 过程特性分析软件包PACP是用来捕获网络数据包的网络编程接口。这个抓包库给抓包系统提供了一个高层次的接口。所有网络上的数据包,包括发送给其他主机的,通过这种机制都可以捕获。它也支持把捕获的数据包保存为本地文件和从本地文件读取信息。

[0118] 实施例一

[0119] 图5示出了本申请实施例提供的一种页面显示方法的步骤流程图,该页面显示方法可以应用于视联网,具体包括以下步骤:

[0120] 步骤501:接收在预置浏览器中生成的页面加载请求。

[0121] 在本发明实施例中,预置浏览器是指内嵌有eglfs插件的基于QT的浏览器。

[0122] QT是一个跨平台C++图形用户界面应用程序开发框架,它既可以开发GUI(Graphical User Interface,图形用户界面)程序,也可以用于开发非GUI程序,比如控制台工具和服务器。

[0123] 页面加载:加载页面对应的资源文件和缓存数据的过程。其中,资源文件为页面加载时的必要组成文件,包括页面的HTML文件、CSS文件、JS文件等。这些资源文件是用于页面显示的框架性基础元素,包括版面结构配置、功能模块配置、系统配置等信息,只有应用提供商对页面进行修改时才会涉及到这些资源文件的修改和更新。缓存数据为需要加载到页面的数据,和资源文件一起组成完整的页面。缓存数据一般指广告或新闻等相关的视频、图片和文字的信息。

[0124] 预置浏览器的生成过程将在下述实施例二中进行详细描述,本发明实施例在此不再加以赘述。

[0125] 页面加载请求可包括对服务网站的访问请求和显示页面中的超链接等页面功能

控件所触发的业务功能启动请求等。上述访问请求和业务功能启动请求包括但不限于用户手动操作客户端设备提供的实体按键或虚拟按钮进行触发。

[0126] 在接收到在预置浏览器中生成的页面加载请求之后，执行步骤502。

[0127] 步骤502：调用内嵌于所述预置浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器。

[0128] 图形处理器(Graphics Processing Unit，GPU)：又称显示核心、视觉处理器、显示芯片，是一种专门在个人电脑、工作站、游戏机和一些移动设备(如平板电脑、智能手机等)上图像运算工作的微处理器。

[0129] 图形处理器可以将计算机系统所需要的显示信息进行转换驱动，并向显示器提供行扫描信号，控制显示器的正确显示，是连接显示器和个人电脑主板的重要元件，也是“人机对话”的重要设备之一。显卡作为电脑主机里的一个重要组成部分，承担输出显示图形的任务，对于从事专业图形设计的人来说显卡非常重要。

[0130] 在接收到在预置浏览器中生成的页面加载请求之后，可以从服务器或终端本地存储的页面加载请求对应的页面数据，在获取到页面加载请求对应的页面数据之后，可以调用内嵌于预置浏览器中的eglfs插件将页面数据直接写入GPU中。

[0131] 可以理解地，本发明实施例中所限定的页面数据即为待显示页面内的图片对应的图片数据。

[0132] 在调用内嵌于预置浏览器中的eglfs插件，将页面加载请求对应的页面数据写入图形处理器之后，执行步骤503。

[0133] 步骤503：由所述图形处理器根据所述页面数据，渲染显示对应的页面。

[0134] GPU可以直接对图片数据进行处理，可以渲染显示对应的页面，即通过显示待显示页面的图片。

[0135] GPU具有处理速度快的特点，本发明实施例通过直接将页面数据写入GPU中进行处理，因此，本发明实施例可以提高页面的加载速度。

[0136] 本申请实施例提供的页面显示方法，通过接收在预置浏览器中生成的页面加载请求，调用内嵌于预置浏览器中的eglfs插件，将页面加载请求对应的页面数据写入图形处理器，由图形处理器根据页面数据，渲染显示对应的页面。本发明实施例通过内置于浏览器中的eglfs插件直接将页面数据输入到图形处理器中进行加载，从而可以提高页面的加载速度。

[0137] 实施例二

[0138] 图6示出了本申请实施例提供的一种页面显示方法的步骤流程图，该页面显示方法可以应用于视联网，具体包括以下步骤：

[0139] 步骤601：编译QT源码。

[0140] 在本发明实施例中，可以预先编译用于生成浏览器的QT源码，QT源码可以是由业务人员根据业务需求编译的，本发明实施例对此不加以限制。

[0141] 在编译QT源码之后，执行步骤602。

[0142] 步骤602：根据所述QT源码编写生成初始浏览器。

[0143] 初始浏览器是直接根据QT源码编写得到的浏览器。

[0144] 在接收到业务人员编译的QT源码之后，可以根据QT源码编写生成初始浏览器。

[0145] 在根据QT源码编写生成初始浏览器之后,执行步骤603。

[0146] 步骤603:在所述初始浏览器中配置linuxfb配置参数或eglfs配置参数,生成所述预置浏览器。

[0147] Linuxfb(即linux Frame Buffer)是指显示设备里的显存。显存里保存是像素的RGB数据。一个显存的大小,屏幕宽度*屏幕高度*每像素字节数。

[0148] eglfs是Qt的一个平台插件,使Qt程序可以利用Opengl ES画图而无需窗口系统。这种方式是在支持GPU的嵌入式设备主要采用的方式;一般需要GPU厂商提供egl和gles驱动模块。

[0149] 在得到初始浏览器之后,可以在初始浏览器中配置linuxfb配置参数,也可以直接配置eglfs配置参数,具体地,可以根据实际情况而定。

[0150] linuxfb配置参数为:export QT_QPA_PLATFORM=linuxfb

[0151] eglfs配置参数为:export QT_QPA_PLATFORM=eglfs

[0152] 在初始浏览器中配置Linuxfb配置参数之后,即可实现初始浏览器中添加Linuxfb插件;而在初始浏览器中配置eglfs配置参数之后,即可实现初始浏览器中添加eglfs插件。

[0153] 预置浏览器是指在初始浏览器中配置Linuxfb配置参数或eglfs配置参数之后,所得到的浏览器。

[0154] 在初始浏览器中配置linuxfb配置参数或eglfs配置参数,生成预置浏览器之后,执行步骤604。

[0155] 步骤604:获取预先设置于所述预置浏览器中的linuxfb配置参数。

[0156] 如果在先步骤中,生成的预置浏览器中配置的为linuxfb配置参数时,需要对linuxfb配置参数进行修改,首先,可以获取预先设置于预置浏览器中的linuxfb配置参数。

[0157] 在获取预先设置于浏览器中的linuxfb配置参数之后,执行步骤604。

[0158] 步骤605:将所述linuxfb配置参数修改为所述eglfs插件对应的eglfs配置参数。

[0159] 在获取linuxfb配置参数之后,可以将linuxfb配置参数修改为eglfs插件对应的eglfs配置参数。即上述步骤中,将linuxfb配置参数:export QT_QPA_PLATFORM=linuxfb,修改为:eglfs配置参数为:export QT_QPA_PLATFORM=eglfs。即实现配置参数的修改。

[0160] 在将linuxfb配置参数修改为eglfs插件对应的eglfs配置参数之后,执行步骤606。

[0161] 步骤606:接收在预置浏览器中生成的页面加载请求。

[0162] QT是一个跨平台C++图形用户界面应用程序开发框架,它既可以开发GUI(Graphical User Interface,图形用户界面)程序,也可以用于开发非GUI程序,比如控制台工具和服务器。

[0163] 页面加载:加载页面对应的资源文件和缓存数据的过程。其中,资源文件为页面加载时的必要组成文件,包括页面的HTML文件、CSS文件、JS文件等。这些资源文件是用于页面显示的框架性基础元素,包括版面结构配置、功能模块配置、系统配置等信息,只有应用提供商对页面进行修改时才会涉及到这些资源文件的修改和更新。缓存数据为需要加载到页面的数据,和资源文件一起组成完整的页面。缓存数据一般指广告或新闻等相关的视频、图片和文字的信息。

[0164] 页面加载请求可包括对服务网站的访问请求和显示页面中的超链接等页面功能

控件所触发的业务功能启动请求等。上述访问请求和业务功能启动请求包括但不限于用户手动操作客户端设备提供的实体按键或虚拟按钮进行触发。

[0165] 在接收到在预置浏览器中生成的页面加载请求之后，执行步骤607。

[0166] 步骤607：根据所述页面加载请求，获取待加载的页面的链接地址。

[0167] 链接地址是指页面加载请求对应的地址，通过链接地址可以获取待加载的页面对应的页面数据。

[0168] 在获取页面加载请求之后，可以根据页面加载请求获取待加载的页面的链接地址，并执行步骤608。

[0169] 步骤608：根据所述链接地址，获取所述页面数据。

[0170] 在获取待加载的页面的链接地址之后，可以根据链接地址从服务器或终端本地获取待加载页面对应的页面数据，具体地获取方式可以根据业务需求而定，本发明实施例对此不加以限制。

[0171] 在根据链接地址获取页面数据之后，执行步骤609。

[0172] 步骤609：调用内嵌于所述预置浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器。

[0173] 图形处理器(Graphics Processing Unit，GPU)：又称显示核心、视觉处理器、显示芯片，是一种专门在个人电脑、工作站、游戏机和一些移动设备(如平板电脑、智能手机等)上图像运算工作的微处理器。

[0174] 图形处理器可以将计算机系统所需要的显示信息进行转换驱动，并向显示器提供行扫描信号，控制显示器的正确显示，是连接显示器和个人电脑主板的重要元件，也是“人机对话”的重要设备之一。显卡作为电脑主机里的一个重要组成部分，承担输出显示图形的任务，对于从事专业图形设计的人来说显卡非常重要。

[0175] 在接收到在预置浏览器中生成的页面加载请求之后，可以从服务器或终端本地存储的页面加载请求对应的页面数据，在获取到页面加载请求对应的页面数据之后，可以调用内嵌于预置浏览器中的eglfs插件将页面数据直接写入GPU中。

[0176] 可以理解地，本发明实施例中所限定的页面数据即为待显示页面内的图片对应的图片数据。

[0177] 在调用内嵌于预置浏览器中的eglfs插件，将页面加载请求对应的页面数据写入图形处理器之后，执行步骤610。

[0178] 步骤610：由所述图形处理器根据所述页面数据渲染显示所述待加载的页面。

[0179] GPU可以直接对图片数据进行处理，可以渲染显示待加载的页面，即通过显示页面内的图片。

[0180] GPU具有处理速度快的特点，本发明实施例通过直接将页面数据写入GPU中进行处理，因此，本发明实施例可以提高页面的加载速度。

[0181] 本申请实施例提供的页面显示方法，通过接收在预置浏览器中生成的页面加载请求，调用内嵌于预置浏览器中的eglfs插件，将页面加载请求对应的页面数据写入图形处理器，由图形处理器根据页面数据，渲染显示对应的页面。本发明实施例通过内置于浏览器中的eglfs插件直接将页面数据输入到图形处理器中进行加载，从而可以提高页面的加载速度。

[0182] 需要说明的是，对于方法实施例，为了简单描述，故将其都表述为一系列的动作组合，但是本领域技术人员应该知悉，本申请实施例并不受所描述的动作顺序的限制，因为依据本申请实施例，某些步骤可以采用其他顺序或者同时进行。其次，本领域技术人员也应该知悉，说明书中所描述的实施例均属于优选实施例，所涉及的动作并不一定是本申请实施例所必须的。

[0183] 实施例三

[0184] 参照图7，示出了本申请实施例提供的页面显示装置的结构示意图，该装置可以应用于视联网中，具体可以包括如下模块：

[0185] 加载请求接收模块710，用于接收在预置浏览器中生成的页面加载请求；

[0186] 页面数据写入模块720，用于调用内嵌于所述预置浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器；

[0187] 页面渲染显示模块730，用于由所述图形处理器根据所述页面数据，渲染显示对应的页面。

[0188] 优选地，所述装置还包括：

[0189] 配置参数获取模块，用于获取预先设置于所述预置浏览器中的linuxfb配置参数；

[0190] 配置参数修改模块，用于将所述linuxfb配置参数修改为所述eglfs插件对应的eglfs配置参数。

[0191] 优选地，所述装置还包括：

[0192] QT源码编译模块，用于编译QT源码；

[0193] 初始浏览器生成模块，用于根据所述QT源码编写生成初始浏览器；

[0194] 预置浏览器生成模块，用于在所述初始浏览器中配置linuxfb配置参数或eglfs配置参数，生成所述预置浏览器。

[0195] 优选地，所述装置还包括：

[0196] 链接地址获取模块，用于根据所述页面加载请求，获取待加载的页面的链接地址；

[0197] 页面数据获取模块，用于根据所述链接地址，获取所述页面数据。

[0198] 优选地，所述页面渲染显示模块730包括：

[0199] 页面渲染显示子模块，用于由所述图形处理器根据所述页面数据渲染显示所述待加载的页面。

[0200] 本申请实施例提供的页面显示装置，通过接收在预置浏览器中生成的页面加载请求，调用内嵌于预置浏览器中的eglfs插件，将页面加载请求对应的页面数据写入图形处理器，由图形处理器根据页面数据，渲染显示对应的页面。本发明实施例通过内置于浏览器中的eglfs插件直接将页面数据输入到图形处理器中进行加载，从而可以提高页面的加载速度。

[0201] 对于装置实施例而言，由于其与方法实施例基本相似，所以描述的比较简单，相关之处参见方法实施例的部分说明即可。

[0202] 本说明书中的各个实施例均采用递进的方式描述，每个实施例重点说明的都是与其他实施例的不同之处，各个实施例之间相同相似的部分互相参见即可。

[0203] 本领域内的技术人员应明白，本申请实施例的实施例可提供为方法、装置、或计算机程序产品。因此，本申请实施例可采用完全硬件实施例、完全软件实施例、或结合软件和

硬件方面的实施例的形式。而且，本申请实施例可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

[0204] 本申请实施例是参照根据本申请实施例的方法、终端设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理终端设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理终端设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0205] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理终端设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0206] 这些计算机程序指令也可装载到计算机或其他可编程数据处理终端设备上，使得在计算机或其他可编程终端设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程终端设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0207] 尽管已描述了本申请实施例的优选实施例，但本领域内的技术人员一旦得知了基本创造性概念，则可对这些实施例做出另外的变更和修改。所以，所附权利要求意欲解释为包括优选实施例以及落入本申请实施例范围的所有变更和修改。

[0208] 最后，还需要说明的是，在本文中，诸如第一和第二等之类的关系术语仅仅用来将一个实体或者操作与另一个实体或操作区分开来，而不一定要求或者暗示这些实体或操作之间存在任何这种实际的关系或者顺序。而且，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、物品或者终端设备不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、物品或者终端设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括所述要素的过程、方法、物品或者终端设备中还存在另外的相同要素。

[0209] 以上对本申请所提供的页面显示方法、页面显示装置、电子设备和存储介质。进行了详细介绍，本文中应用了具体个例对本申请的原理及实施方式进行了阐述，以上实施例的说明只是用于帮助理解本申请的方法及其核心思想；同时，对于本领域的一般技术人员，依据本申请的思想，在具体实施方式及应用范围上均会有改变之处，综上所述，本说明书内容不应理解为对本申请的限制。

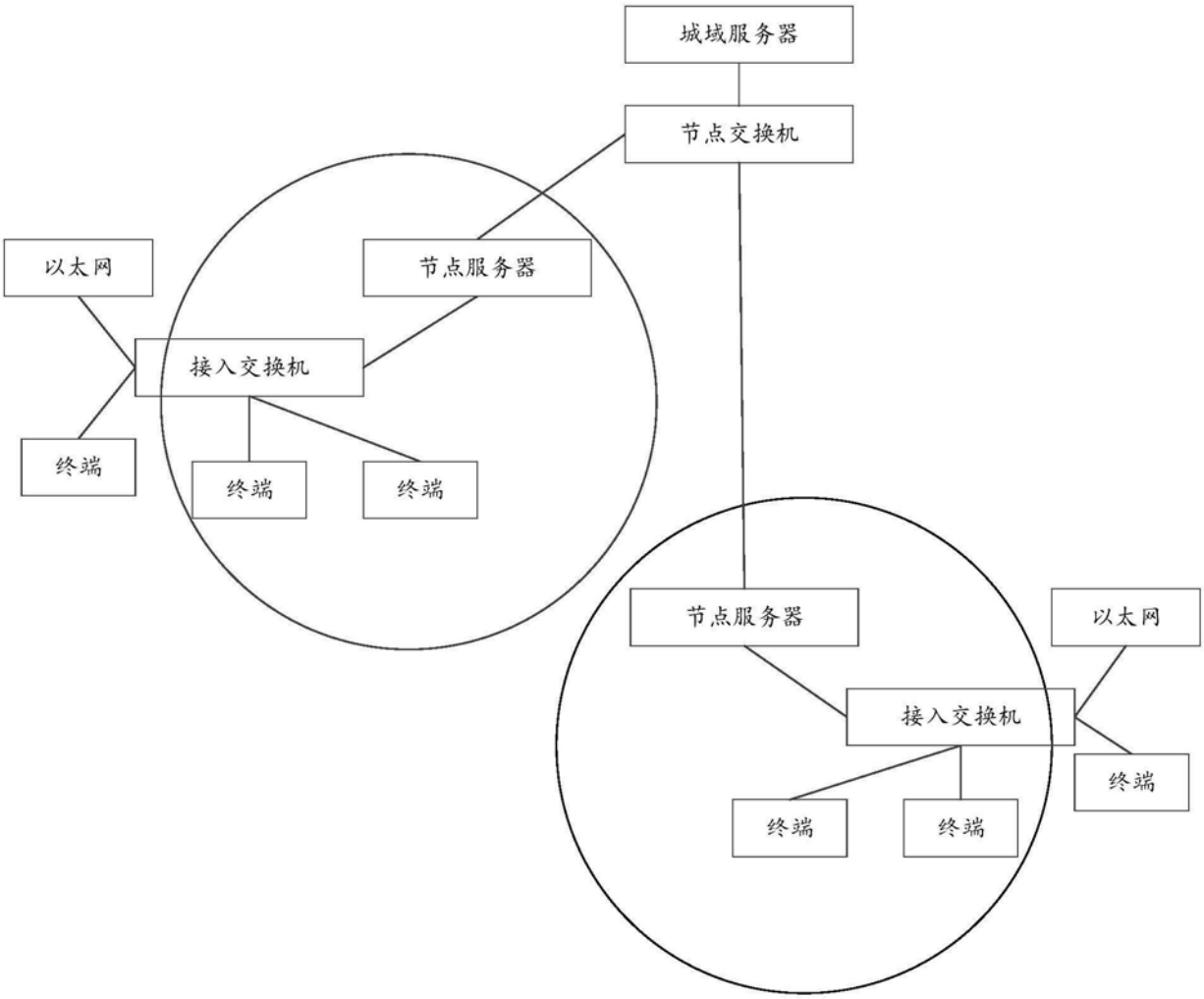

图1

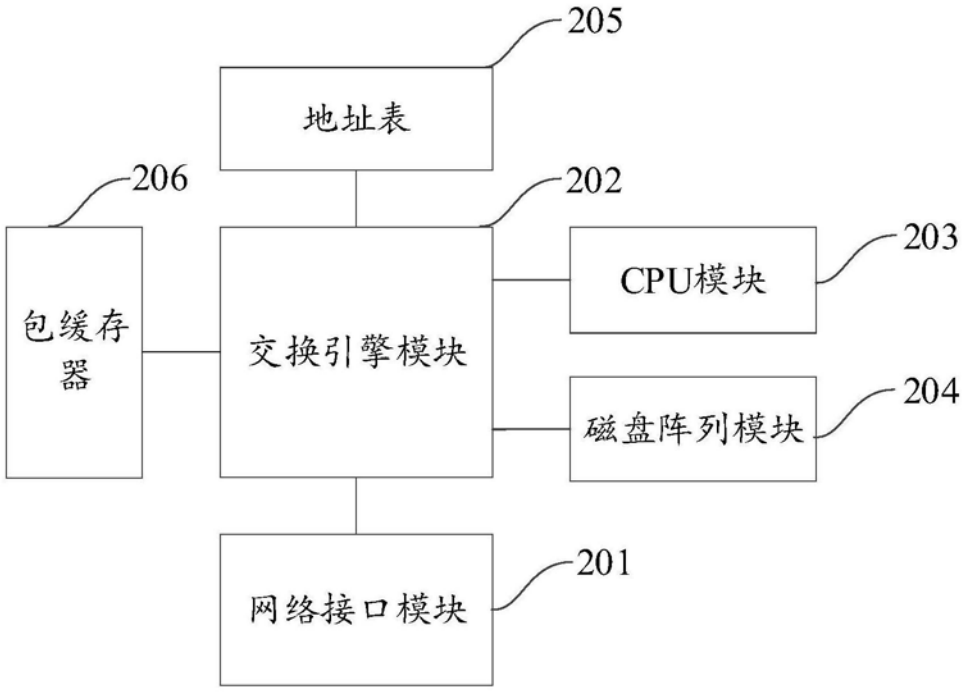
205
地址表
206
202
包缓存器
交换引擎模块
CPU模块
203
磁盘阵列模块
204
网络接口模块
201

图2

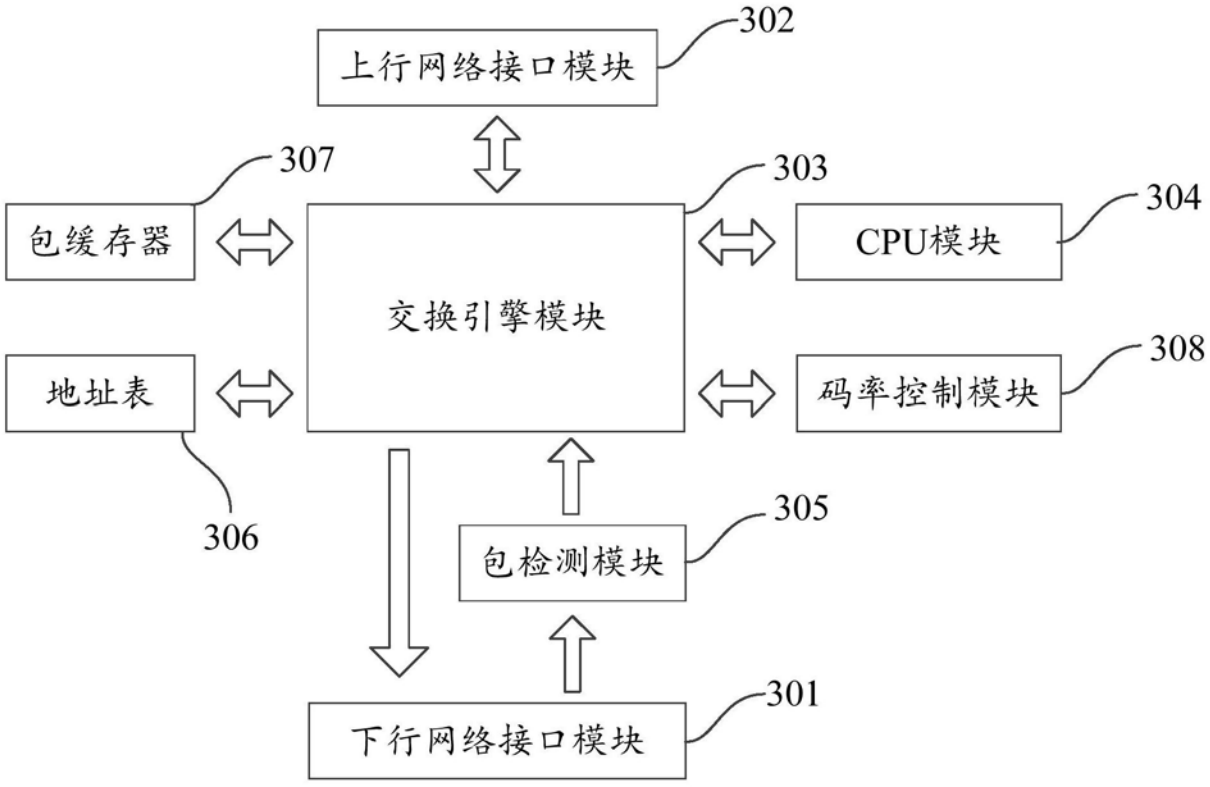
302
上行网络接口模块
307
303
304
包缓存器
交换引擎模块
CPU模块
地址表
308
码率控制模块
306
305
包检测模块
301
下行网络接口模块

图3

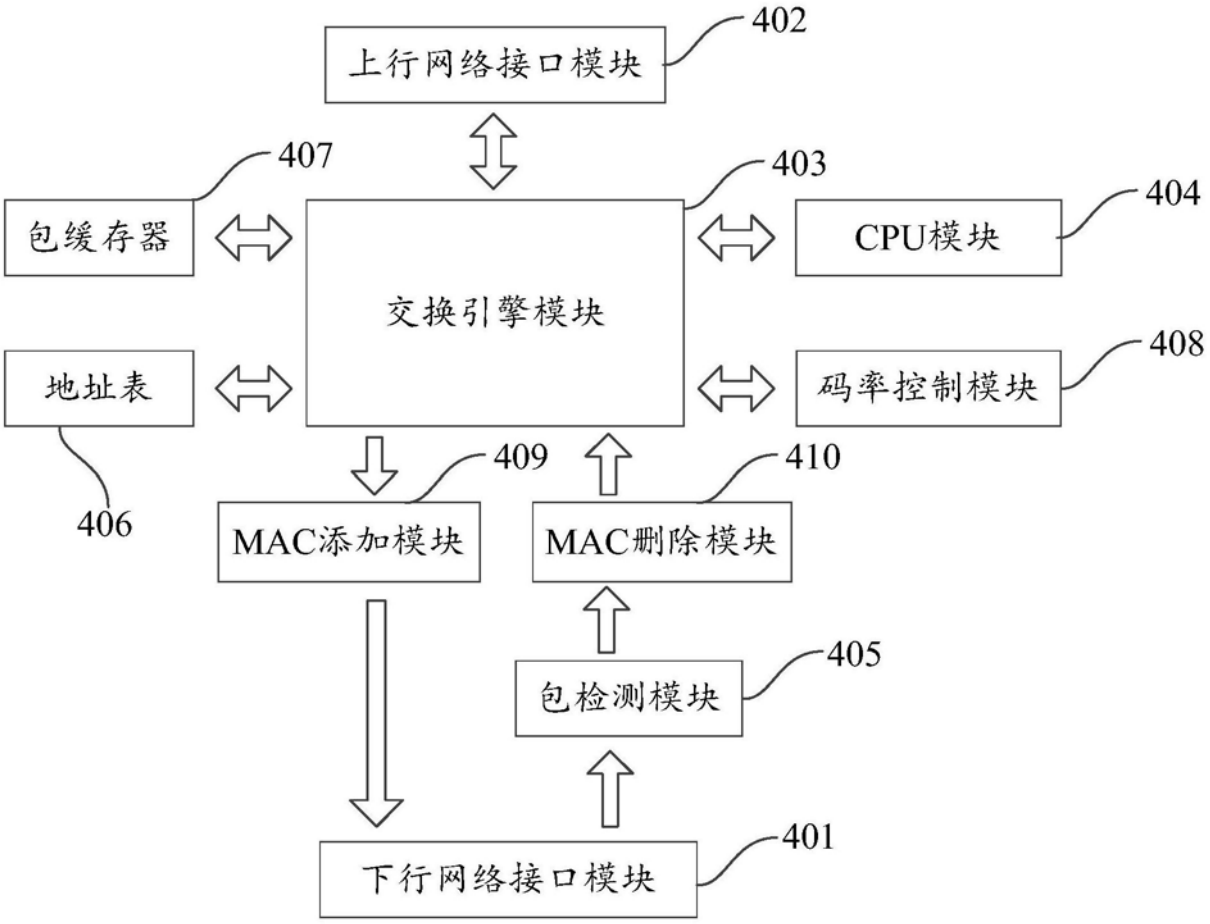
上行网络接口模块
402
包缓存器
407
交换引擎模块
403
CPU模块
404
地址表
406
码率控制模块
408
MAC添加模块
409
MAC删除模块
410
包检测模块
405
下行网络接口模块
401

图4

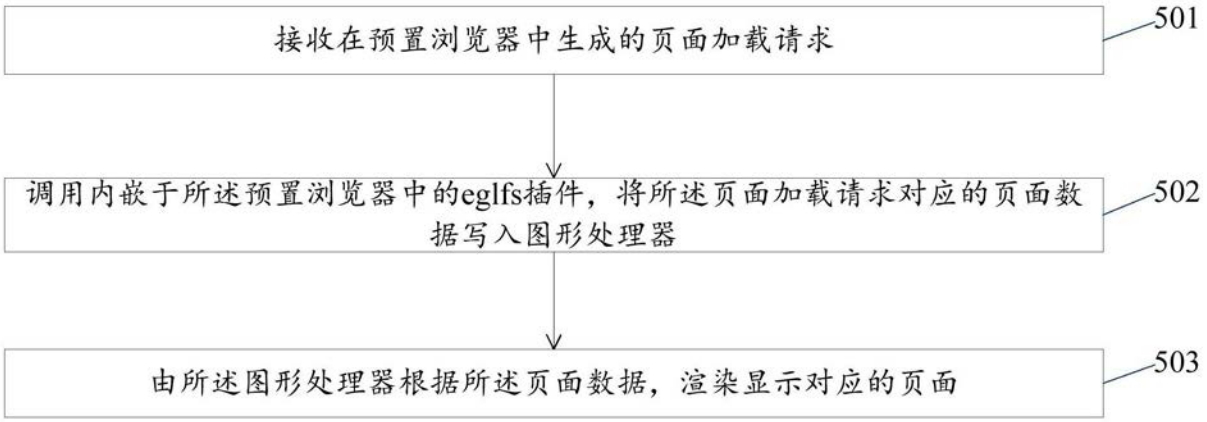
接收在预置浏览器中生成的页面加载请求
501
调用内嵌于所述预置浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器
502
由所述图形处理器根据所述页面数据，渲染显示对应的页面
503

图5

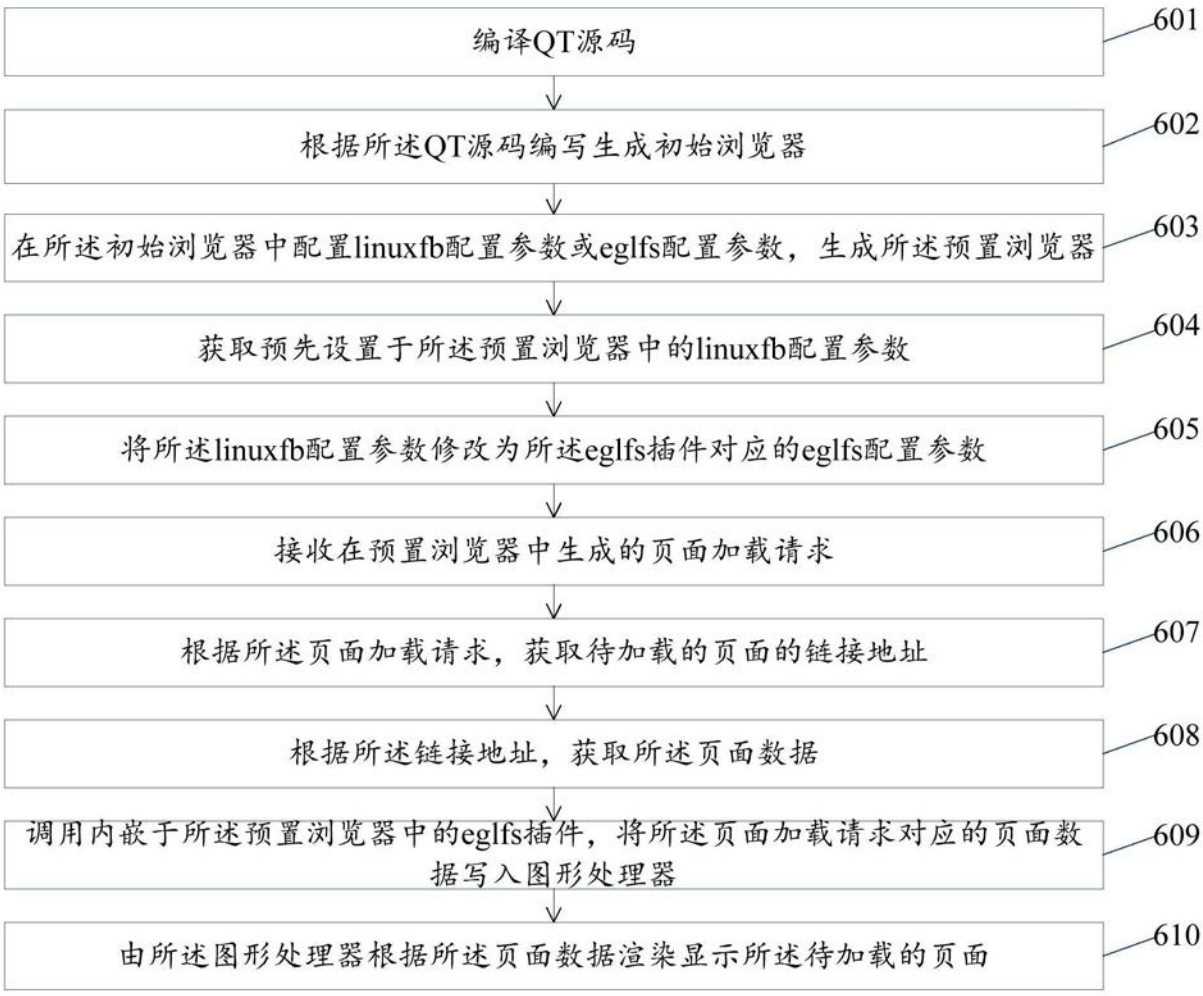
编译QT源码
601
根据所述QT源码编写生成初始浏览器
602
在所述初始浏览器中配置linuxfb配置参数或eglfs配置参数，生成所述预置浏览器
603
获取预先设置于所述预置浏览器中的linuxfb配置参数
604
将所述linuxfb配置参数修改为所述eglfs插件对应的eglfs配置参数
605
接收在预置浏览器中生成的页面加载请求
606
根据所述页面加载请求，获取待加载的页面的链接地址
607
根据所述链接地址，获取所述页面数据
608
调用内嵌于所述预置浏览器中的eglfs插件，将所述页面加载请求对应的页面数据写入图形处理器
609
由所述图形处理器根据所述页面数据渲染显示所述待加载的页面
610

图6

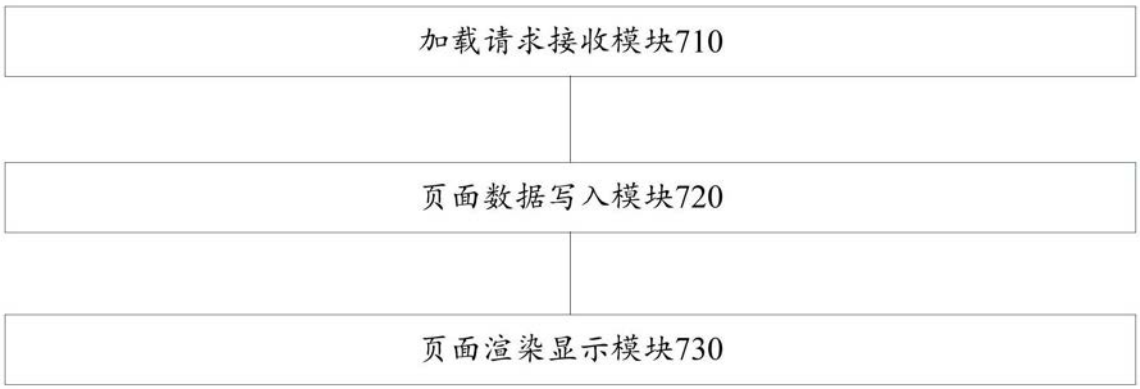
加载请求接收模块710
页面数据写入模块720
页面渲染显示模块730

图7