

(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国际局

(43) 国际公布日
2018 年 12 月 13 日 (13.12.2018)



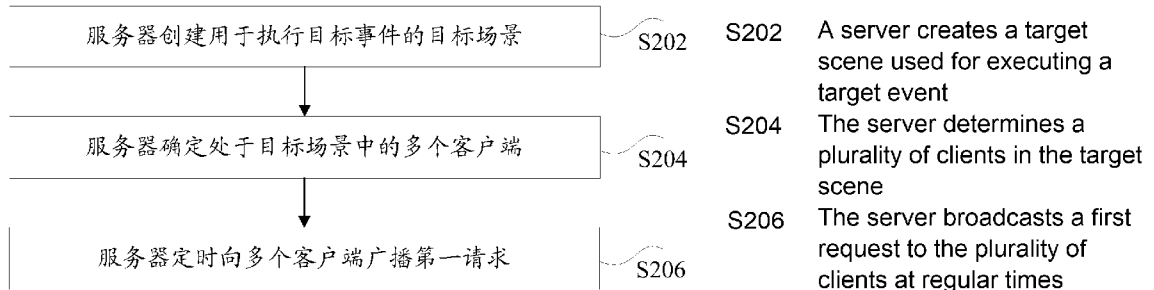
(10) 国际公布号
WO 2018/223800 A1

- (51) 国际专利分类号:
H04L 29/08 (2006.01)
- (21) 国际申请号: PCT/CN2018/086327
- (22) 国际申请日: 2018 年 5 月 10 日 (10.05.2018)
- (25) 申请语言: 中文
- (26) 公布语言: 中文
- (30) 优先权:
201710430708.4 2017年6月8日 (08.06.2017) CN
- (71) 申请人: 腾讯科技(深圳)有限公司 (TENCENT TECHNOLOGY (SHENZHEN) COMPANY LIMITED) [CN/CN]; 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong 518057 (CN)。
- (72) 发明人: 韩伟(HAN, Wei); 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong

518057 (CN)。 艾风(AI, Feng); 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong 518000 (CN)。 吴佳伟(WU, Jiawei); 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong 518000 (CN)。 蔡锦彬(CAI, Jinbin); 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong 518000 (CN)。 潘兵辉(PAN, Binghui); 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong 518000 (CN)。 许路平(XU, Luping); 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong 518000 (CN)。 杨凯(YANG, Kai); 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong 518000 (CN)。 王伟光(WANG, Weiguang); 中国广东省深圳市南山区高新区科技中一路腾讯大厦 35 层, Guangdong 518000 (CN)。

(54) Title: DATA PROCESSING METHOD AND DEVICE, STORAGE MEDIUM, AND ELECTRONIC DEVICE

(54) 发明名称: 数据处理方法和装置、存储介质及电子装置



(57) Abstract: Disclosed in the present application are a data processing method and device, a storage medium, and an electronic device. The method comprises: a server creates a target scene used for executing a target event, the target event being a real-time battle event; the server determines a plurality of clients in the target scene, the plurality of clients being used for executing the target event; the server broadcasts a first request to the plurality of clients at regular times, wherein the first request is used for requesting the plurality of clients to execute a first event synchronously, and carries a first data packet used for executing the first event and sent by at least one of the plurality of clients, and the target event comprises the first event. The present application resolves the technical problem in the related art of high development costs of data processing.

(57) 摘要: 本申请公开了一种数据处理方法和装置、存储介质及电子装置。其中, 该方法包括: 服务器创建用于执行目标事件的目标场景, 其中, 目标事件为实时对战类事件; 服务器确定处于目标场景中的多个客户端, 其中, 多个客户端用于执行目标事件; 服务器定时向多个客户端广播第一请求, 其中, 第一请求用于请求多个客户端同步执行第一事件, 且携带有由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包, 目标事件包括第一事件。本申请解决了相关技术中数据处理的开发成本高的技术问题。

(74) 代理人: 北京康信知识产权代理有限公司 (KANGXIN PARTNERS, P. C.); 中国北京市海淀区知春路甲 48 号盈都大厦 A 座 16 层, Beijing 100098 (CN)。

(81) 指定国 (除另有指明, 要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。

(84) 指定国 (除另有指明, 要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告 (条约第 21 条 (3))。

说明书

数据处理方法和装置、存储介质及电子装置

本申请要求于 2017 年 6 月 8 日提交中国专利局、优先权号为 2017104307084、发明名称为“数据处理方法和装置”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本申请涉及数据处理领域，具体而言，涉及一种数据处理方法和装置、存储介质及电子装置。

背景技术

目前，在数据处理中，比如，在现代网络游戏市场中，越来越多的实时对战玩法已经出现。实时对战玩法因其带来的紧张、刺激的体验，受到了大量游戏玩家的欢迎。而实时对战玩法的具体分类非常多，常见的有多人在线战术竞技游戏类（Multiplayer Online Battle Arena Games，简称为 MOBA）、即时战略游戏（Real-Time Strategy Game，简称为 RTS）、射击类游戏类（Shooting Game，简称为 STG）、飞行射击类、横版格斗类、赛车竞速类、各种球类比赛等。如此种类繁多的游戏，为了实现实时对战玩法，都需要开发一套利用网络来实时传送玩家操作、同步游戏状态的服务器系统，数据处理的开发成本高。

在实时对战的数据处理方法中，通常采用比较通用的广播服务器方案，以及具体某个游戏专用的方案。

在通用的广播服务器方案中，利用服务器来广播客户端的一些玩家操作，或者某些游戏状态的变更事件，以便使连接到这个服务器上的客户端，都能通过这些广播数据，同步游戏的状态。在通用的广播服务器方案中，比较著名的有客户端游戏引擎所自带的广播服务，比如，Unity3D 就提供

了对游戏对象的特定数值的“同步”功能；再比如，Unreal 引擎也有自己的游戏服务器端方案，具备广播功能。除了客户端游戏引擎外，还有一些更加通用的广播服务器，如 Smart Fox Server 这款利用 JAVA 开发的商业软件，通过设置一个“服务器端变量”的概念，让多个客户端都可以读、写这个变量，同时也可以收到这个变量变更通知，从而实现游戏状态同步的目的。另外，还采用 Adobe Media Server 这种用来广播视频流的服务器，以作为游戏状态广播的服务器。

但是，Unity/Unreal 这类专业游戏引擎所带的广播服务器，一般都和其客户端引擎死死绑定，如果想用其他客户端技术，几乎是无法使用他们的服务器功能，因而通用性是一个问题。而 Smart Fox Server/Adobe Media Server 这一类的广播服务器功能，往往只有实时广播的功能，而缺乏实时对战游戏所需要的帧同步广播的功能。性能问题也是通用广播服务器的另一大缺点。上述游戏服务器基本上都不具备分布式能力，也即，只设计了让玩家集中在一台服务器上玩的功能。如果用大量服务器组成一个服务器集群，则这些复杂的服务器端协调和通讯工作，就只能让开发者自己来解决。另外，这些服务器大部分都是商业软件，是不能修改的，而且其中一些是没有源代码的，因而要添加集群功能几乎是不可能的。

在具体某个游戏专用的方案中，采用专门的编写广播服务器。这些服务器的功能除了用于对多个客户端实现广播之外，通常还具备大量针对具体游戏业务逻辑的控制代码，比如，控制战场上怪物的生成、BOSS 怪物的 AI、物品奖励的掉落等等。

专用游戏服务器存在的问题，第一，通用性差，每款游戏都要重新开发一套，几乎每个专业的游戏都有一套独立的“战斗服务器”，由于每个游戏的玩法不一样，所以战斗服务器中运行的逻辑代码千差万别。这导致了每开发一款游戏，都要从头去编写这个“战斗服务器”，包括底层性能攸关的广播、通信代码，这就导致了很高的开发成本；第二，游戏逻辑和网络功能耦合，除错和实现负载均衡（运维）都比较困难，“战斗服务器”

由于是游戏中最重要的服务器之一，集中了大量复杂的游戏逻辑代码。因此战斗服务器也是故障最频繁，BUG 出现概率最高的地方，由于战斗是一个持续在线的过程，所以大量的状态数据和网络包在战斗服务器中处理，这本身是一个多用户并发的复杂业务场景，不管是游戏逻辑，还是底层的网络状况出问题，都会导致这个复杂的实时系统出现故障，并且这种故障由于大部分状态都是在内存，而不是磁盘上，因而很难保留现场，所以战斗服务器的调试和除错，一直是一个难题。

针对上述的数据处理的开发成本高的问题，目前尚未提出有效的解决方案。

发明内容

本申请实施例提供了一种数据处理方法和装置、存储介质及电子装置，以至少解决相关技术中数据处理的开发成本高的技术问题。

根据本申请实施例的一个方面，提供了一种数据处理方法。该方法包括：服务器创建用于执行目标事件的目标场景，其中，目标事件为实时对战类事件；服务器确定处于目标场景中的多个客户端，其中，多个客户端用于执行目标事件；服务器定时向多个客户端广播第一请求，其中，第一请求用于请求多个客户端同步执行第一事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包，目标事件包括第一事件。

根据本申请实施例的另一方面，还提供了一种数据处理装置，应用于服务器。该装置包括：创建单元，设置为创建用于执行目标事件的目标场景，其中，目标事件为实时对战类事件；确定单元，设置为确定处于目标场景中的多个客户端，其中，多个客户端用于执行目标事件；广播单元，设置为定时向多个客户端广播第一请求，其中，第一请求用于请求多个客户端同步执行第一事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包，目标事件包括第一事件。

根据本发明的实施例的又一方面，还提供了一种存储介质，该存储介质中存储有计算机程序，其中，该计算机程序被设置为运行时执行上述方法实施例中的步骤。

根据本发明实施例的又一方面，还提供了一种电子装置，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，其中，上述处理器通过计算机程序执行上述方法实施例中的步骤。

在本申请实施例中，服务器创建用于执行目标事件的目标场景，其中，目标事件为实时对战类事件；服务器确定处于目标场景中的多个客户端，其中，多个客户端用于执行目标事件；服务器定时向多个客户端广播第一请求，其中，第一请求用于请求多个客户端同步执行第一事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包，目标事件包括第一事件，由于定时向多个客户端广播第一请求，以使多个客户端同步执行第一事件，达到了通用广播服务器的帧同步广播的目的，从而实现了降低数据处理的开发成本的技术效果，进而解决了相关技术中数据处理的开发成本高的技术问题。

附图说明

此处所说明的附图用来提供对本申请的进一步理解，构成本申请的一部分，本申请的示意性实施例及其说明用于解释本申请，并不构成对本申请的不当限定。在附图中：

图 1 是根据本申请实施例的数据处理方法的硬件环境的示意图；

图 2 是根据本申请实施例的一种数据处理方法的流程图；

图 3 是根据本申请实施例的一种数据处理的结构示意图；

图 4 是根据本申请实施例的一种同时支持帧同步和即时广播功能的方法的流程示意图；

图 5 是根据本申请实施例的一种帧同步网络广播的方法的流程示意图；

图 6 是根据本申请实施例的一种数据冗余多发的方法的流程示意图；

图 7 是根据本申请实施例的一种断线重连的方法的流程示意图；

图 8 是根据本申请实施例的一致缓存发包的方法的流程示意图；

图 9 是根据本申请实施例的一种服务器端的架构图；

图 10 是根据本申请实施例的一种游戏房间操作流程的示意图；

图 11 是根据本申请实施例的一种消息分发流程的示意图；

图 12 是根据本申请实施例的客户端的登录流程的示意图；

图 13 是根据本申请实施例的一种开始流程的示意图；

图 14 是根据本申请实施例的一种帧同步流程的示意图；

图 15 是根据本申请实施例的一种广播流程的示意图；

图 16 是根据本申请实施例的一种 P2P 方式的同步游戏的示意图；

图 17 是根据本申请实施例的一种参数同步的接口方法的流程图；

图 18 是根据本申请实施例的一种数据处理装置的示意图；以及

图 19 是根据本申请实施例的一种终端的结构框图。

具体实施方式

为了使本技术领域的人员更好地理解本申请方案，下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本申请一部分的实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都应当属于本申请保护的范围。

需要说明的是，本申请的说明书和权利要求书及上述附图中的术语“第一”、“第二”等是用于区别类似的对象，而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换，以便这里描

述的本申请的实施例能够以除了在这里图示或描述的那些以外的顺序实施。此外，术语“包括”和“具有”以及他们的任何变形，意图在于覆盖不排他的包含，例如，包含了一系列步骤或单元的过程、方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元，而是可包括没有清楚地列出的或对于这些过程、方法、产品或设备固有的其它步骤或单元。

根据本申请实施例的一个方面，提供了一种数据处理方法的实施例。

可选地，在本实施例中，上述数据处理方法可以应用于如图 1 所示的由服务器 102 和终端 104 所构成的硬件环境中。图 1 是根据本申请实施例的数据处理方法的硬件环境的示意图。如图 1 所示，服务器 102 通过网络与终端 104 进行连接，上述网络包括但不限于：广域网、城域网或局域网，终端 104 并不限于 PC、手机、平板电脑等。本申请实施例的数据处理方法可以由服务器 102 来执行，也可以由终端 104 来执行，还可以是由服务器 102 和终端 104 共同执行。其中，终端 104 执行本申请实施例的数据处理方法也可以是由安装在其上的客户端来执行。

可选地，上述服务器可以为用于广播第一请求的服务器。

图 2 是根据本申请实施例的一种数据处理方法的流程图。如图 2 所示，该方法可以包括以下步骤：

步骤 S202，服务器创建用于执行目标事件的目标场景。

在本申请实施例上述步骤 S202 提供的技术方案中，目标事件为实时对战类事件，为在数据处理过程中执行的实时对战类事件，可以为在执行某一局游戏时所产生的与游戏相关的实时同步的战斗游戏事件。目标场景为参与执行目标事件的客户端所处的场景，参与执行目标事件的客户端在目标场景中可以接收到服务器广播的数据，以保证同步处理数据。

可选地，目标场景为虚拟的房间。在这个房间内，所有的客户端都可以发送消息，服务器系统会向房间中的客户端广播这条消息，并且目标事件在执行时可以在运行时创建、销毁、进入、退出这些目标场景。

在目标事件的执行逻辑中，一旦开启一场需要实时同步的目标事件，就可以创建用于执行目标事件的目标场景。

该实施例通过创建用于执行目标事件的目标场景，从而实现了对目标场景的管理。

步骤 S204，服务器确定处于目标场景中的多个客户端。

在本申请实施例上述步骤 S204 提供的技术方案中，多个客户端用于执行目标事件，针对任何目标事件的处理，参与的客户端都需要确定下来，也即，需要明确哪些客户端需要在目标场景中广播同一局的数据。在创建用于执行目标事件的目标场景之后，服务器确定处于目标场景中的多个客户端，多个客户端都可以在目标场景中发送消息。

该实施例的多个客户端并非与服务器绑定，该服务器的功能可以由多个客户端通用，其它客户端也可以在进入目标场景之后也可以与服务器进行通信，提高了事件处理功能的通用性。

在服务器确定处于目标场景中的多个客户端之后，就可以开始目标事件的同步数据广播了。

步骤 S206，服务器定时向多个客户端广播第一请求。

在本申请实施例上述步骤 S206 提供的技术方案中，第一请求用于请求多个客户端同步执行第一事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包，目标事件包括第一事件。

普通的即时广播，可以用于实现游戏中的频道聊天、准备状态通知等同步性要求不高的功能，该实施例不只是在收到第一请求时，就立刻广播，而且还支持以帧同步的方式进行广播。该实施例的第一请求为需要服务器广播的请求，可以由客户端中的至少一个客户端发送，并且携带有用于执行目标事件中的第一事件的第一数据包，该第一数据包可以包括用于控制游戏角色执行的动作的帧数据。可选地，帧同步的广播功能可以通过用户数据报协议（User Datagram Protocol，简称为 UDP）进行实现。

在服务器确定处于目标场景中的多个客户端之后，服务器定时向多个客户端广播第一请求以实现广播数据的目的，可以由定时器在固定的时间广播数据。帧同步广播是一种服务器自动搜集客户端输入组包，然后定时发送广播的游戏开发功能，由服务器定时广播数据包，以驱动客户端渲染帧的实时网络画面同步，几乎所有的实时动作网游游戏，都可以使用这个功能。

可选地，帧同步只有满足以下条件之一才会开始：所有客户端都准备好，服务器（LockStep Server）超时，服务器（LockStep Server）的其它机制满足条件。帧数据由服务器（LockStep Server）按照自己的定时器在固定的时间产生，即使没有任何用户输入，也会生成空帧。客户端会自动检查每个帧同步包的序号，如果出现丢失，则启动 TCP 的下载通道，向中继服务器（RelayServer）去下载缺失的帧，而这个时候 PopFrame()接口将不会返回帧数据，直到所有的缺失帧补齐，才按顺序返回完整的帧数据。

上述两种广播功能，可以通过两个不同的接口提供。其中，帧同步的广播方式是不管服务器有没有收到客户端发送的请求，都会定时对同一目标场景中的其它客户端进行广播，其中，广播的频率可以配置，比如，广播的频率为 0.1 秒到 0.05 秒；如果任何数量的客户端都向服务器发送请求，服务器会保存这些需要广播的请求，在下一次定时广播的数据包内集中起来，再向所有客户端发送。

该实施例的广播通道不依赖于目标事件是否开始执行，只要客户端成功登录之后就可以进行收发数据。广播数据一旦到达客户端的软件开发工具包（Software Development Kit，简称为 SDK），立刻回调用户定义的收包函数进行处理。

该实施例的以上功能，用户在使用的时候，需要通过客户端接口和服务端接口来调用，该实施例可以抽象实时对战类事件的共性，统一地完成上述在事件处理中的功能，提供了多个客户端同步处理所需的广播能力，是一种接入即可用的数据处理方法，只需要开发具体的客户端代码，即可

以快速地实现网络实时对战事件的处理功能，提高了数据处理效率，降低了数据处理的开发成本。

通过上述步骤 S202 至步骤 S206，由于服务器定时向多个客户端广播第一请求，以使多个客户端同步执行第一事件，达到了通用广播服务器的帧同步广播的目的，从而实现了降低数据处理的开发成本的技术效果，进而解决了相关技术中数据处理的开发成本高的技术问题。

作为一种可选的实施方式，步骤 S206，服务器定时向多个客户端广播第一请求包括：服务器接收第一数量的客户端发送的第一请求；在接收到第一请求之后，服务器在第一时间向多个客户端广播第一请求。

多个客户端包括第一数量的客户端。在服务器确定处于目标场景中的多个客户端之后，服务器可以接收多个客户端中的任何数量的客户端发送过来的第一请求，服务器都会保存这些需要广播的请求。

在接收到第一请求之后，服务器在第一时间向多个客户端广播第一请求，第一时间与上一次向多个客户端广播第二请求的时间相隔第一预设时间，第二请求用于请求多个客户端同步执行第二事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第二事件的第二数据包，目标事件包括第二事件。

服务器会定时对同一目标场景中的客户端进行广播。在接收到第一数量的客户端发送的第一请求之后，服务器会保存这些第一请求，在第一时间向多个客户端广播第一请求。可选地，服务器将第一数量的客户端发送的第一请求在定时广播的数据包内集中起来，在第一时间向多个客户端广播第一请求。该第一时间为定时到达的时间，与上一次服务器向多个客户端广播第二请求的时间相隔第一预设时间，该第一预设时间可以设置，比如，第一预设时间为 0.1 秒到 0.05 秒，第二请求与第一请求类似，用于请求多个客户端同步执行第二事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第二事件的第二数据包，上述目标事件包括第二事件。

作为一种可选的实施方式，步骤 S206，服务器定时向多个客户端广播第一请求包括：当第一时间到达时，且未接收到任何多个客户端发送的请求的情况下，向多个客户端广播携带空帧的第一请求。

在服务器定时向多个客户端广播第一请求时，服务器无论有没有收到客户端发送的请求，都会相隔第一预设时间对同一目标场景内的客户端广播接收到的请求。当第一时间到达时，且未接收到任何多个客户端发送的请求的情况下，服务器仍然向多个客户端广播携带空帧的第一请求，也即，即使客户端没有任何输入，也会生成空帧，从而实现通用服务器的帧同步的广播功能。

作为一种可选的实施方式，步骤 S206，服务器定时向多个客户端广播第一请求包括：服务器分别获取前 N 次向多个客户端广播的请求中所携带的 N 个数据包；服务器定时向多个客户端广播携带有 N 个数据包和第一数据包的第一请求。

服务器分别获取前 N 次向多个客户端广播的请求中所携带的 N 个数据包，N 次与 N 个数据包一一对应，N 为大于等于 1 的自然数。

在对战类事件的数据处理的过程中，比如，在实时同步动作游戏开发的过程中，网络会出现延迟波动、偶发丢包、短暂断线等问题，因而需要对弱网络环境进行优化。

在帧同步广播的时候，每个数据包，实际上都会重发多次，也即，每次广播的数据包都会带上之前几次应该发的数据包中的内容。在确定处于目标场景中的多个客户端之后，服务器分别获取前 N 次向多个客户端广播的请求中所携带的 N 个数据包，N 次为数据包的重发次数，与 N 个数据包一一对应，为大于等于 1 的自然数，是可以进行设置的。

在分别获取前 N 次向多个客户端广播的请求中所携带的 N 个数据包之后，服务器定时向多个客户端广播携带有 N 个数据包和第一数据包的第一请求，从而实现了一种使用冗余多发的策略来降低丢包带来的影响。

该实施例通过上述方法实现了定时向多个客户端广播第一请求的目的，针对网络出现的延迟波动、偶发丢包等问题做了特别优化，而不需要开发者过多关注，从而实现了一种使用冗余多发的策略来降低丢包带来的影响。

作为一种可选的实施方式，在步骤 S206，服务器定时向多个客户端广播第一请求之后，记录第一数据包；服务器定时向多个客户端广播第三请求；记录第三数据包。

在服务器端记录下目标事件在目标场景中执行过程中所广播过的所有数据包。在定时向多个客户端广播第一请求之后，记录第一数据包，从而实现对第一数据包保存，以在客户端出现断线重连的情况下，可以再次向客户端发送未接收到的第一数据包。

服务器定时向多个客户端广播第三请求，该第三请求用于请求多个客户端同步执行第三事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第三事件的第三数据包，目标事件包括第三事件。

在记录第一数据包之后，服务器接收到多个客户端中的至少一个客户端发送的第三请求，当定时时间到达时，向多个客户端广播第三请求，该第三请求携带有由多个客户端中的至少一个客户端发送的用于执行第三事件的第三数据包，目标事件包括第三事件，

在服务器定时向多个客户端广播第三请求之后，记录第三数据包，从而实现对第三数据包保存，以在客户端出现断线重连的情况下，可以再次向客户端发送未接收到的第三数据包。

该实施例通过上述方法达到了在服务器端记录下目标场景下广播过的所有数据包的目的，以在客户端断线重连期间未接收到的数据包。

作为一种可选的实施方式，在步骤 S206，服务器定时向多个客户端广播第一请求之后，该方法还包括：在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，服务器向第一客户

端发送第一数据包和第三数据包，多个客户端中的第一客户端失去连接包括：多个客户端中的第一客户端所处的网络断线；或者多个客户端中的第一客户端退出事件执行进程。

在弱网络情况下，网络出现延迟波动、偶尔丢包、短暂断线的情形。由于在断线之后，会导致数据包丢失，也即，在定时向多个客户端广播第一请求之后，在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，服务器第一客户端在失去连接的过程中未接收到第一数据包和第三数据包。由于在服务器端已经记录下第一数据包和第二数据包，向第一客户端发送第一数据包和第三数据包，实现客户端快速获取未接收到的数据包，从而避免客户端丢包的情况出现，提高数据处理效率。

可选地，数据包通过序列号进行标识，每个数据包的序列号具有预设顺序。客户端在断线重新连接之后，会自动从上次断线之后最后广播的数据包的序列号开始，重新下载未接收到的，且服务器记录过的数据包。

可选地，多个客户端中的第一客户端失去连接包括两种情况，第一种情况因为网络断线，在这种情况下，第一客户端可以调用接口重新登录；第二种情况为第一客户端退出事件执行的进程，比如，游戏玩家退出游戏进程，此时目标场景中的地址信息、键值信息、当前帧的帧号等信息丢失，客户端需要重新登录，并从第一帧（PopFrame）开始同步数据。需要说明的是，客户端在收到第一帧时，即可以认为目标事件开始执行。

作为一种可选的实施方式，在步骤 S206，服务器定时向多个客户端广播第一请求之后，在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，服务器接收客户端发送的自定义的第四请求；服务器响应第四请求，并向第一客户端发送第一数据包或第三数据包。

在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，服务器接收客户端发送的第四请求。第四请求用于

请求获取第一数据包或第三数据包，第一客户端在失去连接的过程中未接收到第一数据包和第三数据包。

在服务器定时向多个客户端广播第一请求之后，在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，客户端可以自定义选择未接收到的，且服务器已经记录过的数据包进行接收。接收客户端发送的自定义的第四请求，该第四请求用于请求获取客户端自定义选择的第一数据包或第三数据包。

可选地，客户端也可以自定义从某一个数据包对应的序列号开始下载，这适用于游戏自己有实现状态快照的情况，也即，适用于定时在客户端记录下游戏中的状态的情况，这样可以通过恢复最近的状态快照，从此状态快照处开始下载丢失的数据，其中，状态快照处根据广播的数据包的序列号确定。这个功能也可以用来做游戏的录像重播，由于只记录游戏的操作数据，因而这种游戏录像比视屏录像保存的文件小很多。

服务器响应第四请求，并向第一客户端发送第一数据包或第三数据包，在服务器接收到客户端发送的第四请求之后，响应第四请求，向第一客户端发送第一数据包或第三数据包。

需要说明的是，上述第一数据包和第三数据包并不限定服务器只广播第一数据包和第二数据包，只要是服务器定时向客户端广播的请求中携带的数据包都在本申请的保护范围之内，此处不再一一举例说明。

该实施例通过上述方法实现了客户端快速获取未接收到的数据包，从而避免客户端丢包的情况出现，提高数据处理效率。

作为一种可选的实施方式，服务器记录第一数据包包括：在第一数据包不为空帧的情况下，服务器记录第一数据包；服务器记录第三数据包包括：在第三数据包不为空帧的情况下，服务器记录第三数据包。

在服务器定时向多个客户端广播第一请求之后，记录第一数据包，在第一数据包不为空帧的情况下，记录第一数据包，也即，当第一数据包为

空帧的时候，则不会在服务器中保留。在定时向多个客户端广播第三请求之后，在第三数据包不为空帧的情况下，记录第三数据包，也即，当第三数据包为空帧的情况下，则不会在服务器中保留。因而，断线重连的客户端不会再次获取空帧的数据包。

作为一种可选的实施方式，第一数据包通过第一序列号进行标识，第三数据包通过第二序列号进行标识，其中，第一序列号和第二序列号连续。

服务器每次广播的数据包通过序列号进行标识，也即，每个数据包具有唯一的序列号，每个数据包的序列号可以随着发送时间的递增而连续递增，这样当存在丢包的情况下，客户端可以根据多个连续序列号中缺失的序列化确定丢失的数据包。第一数据包通过第一序列号进行标识，第三数据包为在第一数据包发送之后发送的数据包，通过第二序列号进行标识，第一序列号和第二序列号连续。

作为一种可选的实施方式，步骤 S206，服务器定时向多个客户端广播第一请求包括：服务器定时向每个客户端的缓冲区发送第一数据包，其中，第一数据包定时由每个客户端进行处理。

在弱网络下，延迟波动会影响画面的忽快忽慢，从而难以准确把握输入和响应之间的提前量。在客户端处建立一定大小的缓冲区，服务器定时向每个客户端的缓存区发送第一数据包，客户端在收到第一数据包之后不立刻处理，而是先放入缓冲区，然后定时将缓冲区中的第一数据包由客户端代码进行处理，这样客户端得到的数据包就会在一定时间段内的处理速度都是比较稳定的。可选地，客户端在收到网络广播包之后，不立刻通知游戏逻辑代码，而是先放入缓冲区，然后定时把缓冲区的数据包通知给游戏客户端代码。这样客户端代码就会在一定时间段内稳定地处理得到的数据包。虽然这会增加整体网络数据包的延迟，但是能让所有的网络包的到达速度保持一致，从而为用户形成一个稳定的操作手感，提升用户体验。

作为一种可选的实施方式，在步骤 S206，服务器定时向多个客户端广播第一请求时，该方法还包括：在验证出第一数据包不合法的情况下，

服务器禁止发送第一数据包的客户端执行目标事件,和/或服务器记录用于指示第一数据包不合法的信息。

验证第一数据包是否合法,在对战类事件处理的过程中,可能出现作弊的情况。需要服务器对数据包进行合法性验证。在定时向多个客户端广播第一请求时,验证第一数据包是否合法。

可选地,网络游戏一般都要面对外挂和破解的威胁,在游戏的实时战斗过程中,往往也需要对广播的数据包进行合法性验证。

在验证出第一数据包不合法的情况下,服务器禁止发送第一数据包的客户端执行目标事件,和/或服务器记录用于指示第一数据包不合法的信息。在验证第一数据包是否合法之后,在验证出第一数据包不合法的情况下,服务器禁止发送第一数据包的客户端执行目标事件,比如,在游戏处理事件中,当发现某个客户端作弊时,服务器可以禁止作弊客户端在目标场景中执行游戏事件,也即,将作弊客户端“剔除”本局游戏。

可以获取作弊客户端的完整作弊记录,以用于后期对客户端进一步的处理,比如,根据作弊客户端的完整作弊记录对作弊客户端进一步惩罚和预防。

该实施例服务器端接口,拥有客户端接口的全部功能,再额外增加了对目标场景的管理能力,降低了数据处理的开发成本。

作为一种可选的实施方式,服务器验证第一数据包是否合法包括:服务器通过第一服务器验证第一数据包是否合法。

服务器向第一服务器广播第一数据包,将所有向客户端广播的数据,都同样的向第一服务器广播,该第一服务器可以为开发者所设置的游戏服务器(GameServer),这样游戏服务器就能实时地掌握所有客户端收到的数据包,从而进行任何需要获取的验证鉴权,其中,游戏服务器用于在一般网络游戏中,运算主要的游戏逻辑的服务器进程。

在服务器向第一服务器广播第一数据包之后,通过第一服务器验证第

一数据包是否合法。当游戏服务器发现有某个客户端作弊，开发者不但可以立刻把有问题的客户端“踢出”本局游戏，而且可以获得完整的作弊记录，用作日后的进一步惩罚和预防。

该实施例通过上述方法实现了验证第一数据包是否合法的目的。

作为一种可选的实施方式，在通过第一服务器验证第一数据包是否合法之后，该方法还包括：在通过第一服务器验证第一数据包不合法的情况下，通过第一服务器设置预设操作信息；通过第一服务器向多个客户端中除发送第一数据包的客户端之外的客户端广播预设操作信息。

在通过第一服务器验证第一数据包不合法的情况下，通过第一服务器设置预设操作信息，预设操作信息用于指示多个客户端中除发送第一数据包的客户端之外的客户端执行预设操作。

在通过第一服务器验证第一数据包不合法的情况下，确定与第一数据包的客户端为作弊客户端，踢出第一数据包的客户端。预设操作信息用于指示多个客户端中除发送第一数据包的客户端之外的客户端执行预设操作，比如，在游戏中掉落战利品。

在通过第一服务器设置预设操作信息之后，通过第一服务器向多个客户端中除发送第一数据包的客户端之外的客户端广播预设操作信息，比如，通过服务器的广播接口将掉落的战利品的数据信息向存在于目标场景中的客户端发送。

该实施例通过上述方法实现了目标场景下的信息的广播。

作为一种可选的实施方式，步骤 S202，服务器创建用于执行目标事件的目标场景包括：服务器通过调用预设接口创建目标场景；和/或服务器通过预设接口向多个客户端发送目标场景的信息，其中，目标场景的信息包括以下至少之一：目标场景的地址信息；目标场景的键值信息；目标场景的统一资源定位符。

服务器在建用于执行目标事件的目标场景时，调用预设接口，该预设

接口可以为 LockStep Service 的接口,通过该接口创建目标场景。可选地,在通过调用预设接口创建目标场景之后,通过预设接口向多个客户端发送目标场景的信息,该目标场景的信息包括目标场景的地址信息、目标场景的键值信息、目标场景的统一资源定位符等。

可选地,在游戏玩家连上 GameSvr 大厅后,发起 5V5 对战,GameSvr 通过调用 LockStep Service 的接口创建 PVP 房间,LockStep Service 会返回各个成员的 RoomID, RoomKey 以及房间 Url,通过 GameSvr 将此信息返回给客户端。

作为一种可选的实施方式,步骤 S206,服务器定时向多个客户端广播第一请求包括:服务器获取每个客户端的操作信息;将每个客户端的操作信息组合为数据帧;服务器定时向多个客户端广播携带有数据帧的第一请求。

在确定目标场景下的每个客户端之后,服务器获取每个客户端的操作信息,也即,收集每个客户端的动作信息。

在获取每个客户端的操作信息之后,服务器将每个客户端的操作信息组合为数据帧,也即,组成一个完整的帧内容。

在将每个客户端的操作信息组合为数据帧之后,服务器定时向多个客户端广播携带有数据帧的第一请求,以将数据帧向目标场景中的其他游戏玩家对应的客户端广播。

该实施例通过上述方法实现了定时向多个客户端广播第一请求的目的,由于定时向多个客户端广播第一请求,以使多个客户端同步执行第一件事件,达到了通用广播服务器的帧同步广播的目的。

作为一种可选的实施方式,步骤 S202,服务器创建用于执行目标事件的目标场景包括:服务器获取多个第二服务器的场景信息;接收场景创建请求,并根据多个第二服务器的场景信息确定用于创建目标场景的目标服务器;服务器向目标服务器发送创建指令;在目标服务器接收到创建指

令之后，通过目标服务器创建目标场景。

服务器获取多个第二服务器的场景信息，每个第二服务器的场景信息包括用于指示每个第二服务器允许创建的场景和已经创建的场景的信息。

在创建用于执行目标事件的目标场景时，服务器获取多个第二服务器的场景信息，该第二服务器可以为房间服务器（RoomSvr），每个第二服务器会定期向所有的场景管理服务器（RoomMng）上报每个第二服务器的场景信息，该场景信息用于指示每个第二服务器允许创建的场景和已经创建的场景的信息。比如，RoomSvr 会定期向所有 RoomMng 上报自身的信息，包括总共支持多少房间，当前已创建了多少房间等，其中，RoomSvr 上存有房间的具体信息，以及该房间所有产生的帧数据，进程崩溃（Crash）之后，对应房间的信息将丢失，让业务侧重新创建房间即可，可平行扩展，根据业务需要可动态扩容；RoomMng 主要功能是收集各个 RoomSvr 的状态，管理房间的创建，具体房间信息是存在 Tcaplus 中，而不是在 RoomMng 的内存中，进程崩溃后可以立即重启并向 RoomSvr 拉取信息，迅速恢复服务，可平行扩展，内存和 CPU 消耗不会太大，在一个集群里部署两个节点即可。

服务器接收场景创建请求，并根据多个第二服务器的场景信息确定用于创建目标场景的目标服务器，场景创建请求用于请求创建目标场景。

在服务器获取多个第二服务器的场景信息之后，接收用于请求创建目标场景的场景创建请求，该场景创建请求会随机落在一个 RoomMng 上，RoomMng 收到请求后，会综合当前所有的第二服务器的负载情况，选择一个目标服务器以发送创建房间的命令。

服务器向目标服务器发送创建指令，该创建指令用于指示目标服务器创建目标场景。

在根据多个第二服务器的场景信息确定用于创建目标场景的目标服务器之后，服务器向目标服务器发送创建指令，以指示目标服务器创建目

标场景。

在目标服务器接收到创建指令之后，服务器向目标服务器发送创建指令，通过目标服务器创建目标场景，并给目标场景中的每个客户端分配一个随机键，该随机键可以用于后续客户端到服务器之间的信息传输时为信息加密或者解密。

该实施例通过上述方法实现了创建用于执行目标事件的目标场景的目的，进而确定处于目标场景中的多个客户端；定时向多个客户端广播第一请求，以使多个客户端同步执行第一事件，达到了通用广播服务器的帧同步广播的目的，从而实现了降低数据处理的开发成本的技术效果。

作为一种可选的实施方式，在步骤 S202，服务器创建用于执行目标事件的目标场景之后，该方法还包括以下之一：在多个客户端执行目标事件之后，服务器关闭目标场景；在多个客户端退出目标场景之后，服务器关闭目标场景；在第二预设时间内未与多个客户端通信时，服务器关闭目标场景；在创建目标场景的时间超过第三预设时间时，服务器关闭目标场景。

在创建用于执行目标事件的目标场景之后，服务器可以对创建的目标场景进行关闭，包括：在多个客户端执行目标事件之后，服务器关闭目标场景，可选地，游戏服务器（GameSvr）主动销毁房间，包括对战正常结束，或者游戏发现客户端作弊而异常关闭的，消息路径可以为 GameSvr->RoomMng->Tcaplus；在多个客户端退出目标场景之后，关闭目标场景，可选地，RoomSvr 收到了所有客户端退出房间的消息，而关闭房间，这个是对应正常的 PVP 战斗结束流程，消息路径可以为 RoomSvr->RoomMng->Tcaplus；在第二预设时间内未与多个客户端通信时，关闭目标场景，可选地，RoomSvr 上因为房间若干时间没消息通信而关闭房间，这种主要为防止 PVP 已经正常结束，但 GameSvr 和客户端没有主动调用关闭房间的情况，消息路径可以为 RoomSvr->RoomMng->Tcaplus；在创建目标场景的时间超过第三预设时间时，关闭目标场景，可选地，自

动扫描工具扫描 Tcaplus 表中超过 2 天前建立的房间，将其关闭，这种是为防止 RoomSvr 重启导致房间信息丢失的情况，消息路径为 CleanTool->Tcaplus，其中，Tcaplus 为一款 IEG 研发部自主研发的存储系统，CleanTool 为清除数据的工具。

作为一种可选的实施方式，在步骤 S204，服务器确定处于目标场景中的多个客户端之后，该方法还包括：服务器对多个客户端的登录信息进行校验；在对至少一个客户端的登录信息校验失败的情况下，服务器发送校验失败信息；在对多个客户端的登录信息校验成功的情况下，服务器将多个客户端发送的数据设置为空帧。

在确定处于目标场景中的多个客户端之后，服务器对多个客户端的登录信息进行校验，对每个客户端的键值（MemberKey）进行校验。

在对多个客户端的登录信息进行校验之后，在对至少一个客户端的登录信息校验失败的情况下，服务器发送校验失败信息，也即，如果客户端的登录信息校验失败，则返回错误信息。

在对多个客户端的登录信息进行校验之后，在对多个客户端的登录信息校验成功的情况下，服务器将多个客户端发送的数据设置为空帧，可以将多个客户端发送的数据的当前帧号（FrameId）标记为 0，并且 FrameId 为 0 的数据不会保留，也即，在客户端断线重连之后，不会收到 FrameId 为 0 的数据。

可选地，对局登录会对 MemberKey 进行校验，如果失败返回错误信息。在登录阶段，将服务器对局状态标记为 Login，这个时候服务器会对客户端的输入进行转发，但 FrameId 标记为 0，并且 FrameId 为 0 的数据不会在 LockStepSvr 保留，也即，断线重连上来的客户端不会收到 FrameId 为 0 的数据。

该实施例通过上述方法实现了目标场景下的客户端的登录。

作为一种可选的实施方式，步骤 S206，服务器定时向多个客户端广

播第一请求包括: 服务器接收多个客户端发送的第五请求; 响应第五请求, 并允许定时向多个客户端广播第一请求。

服务器接收多个客户端发送的第五请求, 第五请求用于请求开始执行目标事件。

在定时向多个客户端广播第一请求时, 服务器接收多个客户端发送的第五请求, 该第五请求用于请求开始执行目标事件。可选地, 对局开始的时机为服务器接收到所有客户端的发送的第五请求才开始正式进行对局转发, 该第五请求可以为对战开始请求 (BattleStart)。

在接收多个客户端发送的第五请求之后, 服务器响应第五请求, 此时将当前帧号 FrameId 标记为 1, 表示帧同步正式开始, 并允许定时向多个客户端广播第一请求。

该实施例通过上述方法实现了开始定时向多个客户端广播第一请求的目的。

作为一种可选的实施方式, 服务器允许定时向多个客户端广播第一请求包括: 在未全部接收到多个客户端发送的第五请求的情况下, 服务器在第三预设时间允许定时向多个客户端广播第一请求。

服务器允许定时向多个客户端广播第一请求时, 在未全部接收到多个客户端发送的第五请求的情况下, 也即, 只收到了部分客户端发送的第五请求, 服务器会触发一个超时机制, 在第三预设时间允许定时向多个客户端广播第一请求, 比如, 在收到第一客户端发送的第五请求之后 10 秒, 也会启动帧同步流程, 上述第三预设时间可以由服务器进行设定。

作为一种可选的实施方式, 在步骤 S206, 服务器定时向多个客户端广播第一请求之后, 该方法还包括: 服务器接收多个客户端发送的第六请求, 其中, 第六请求用于请求结束执行目标事件; 服务器响应第六请求, 并关闭目标场景以使多个客户端结束执行目标事件。

服务器接收多个客户端发送的第六请求, 第六请求用于请求结束执行

目标事件。

在定时向多个客户端广播第一请求之后，可以结束多个客户端对目标事件的执行。服务器接收多个客户端发送的第六请求，该第六请求用于请求结束预设时间。可选地，客户端调用结束接口（Stop 接口），此时丢弃客户端的输入。

在接收到多个客户端发送的第六请求之后，服务器响应第六请求，并关闭目标场景以使多个客户端结束执行目标事件，目标场景进入销毁状态。

该实施例通过上述方法实现了多个客户端结束执行目标事件的目的。

作为一种可选的实施方式，步骤 S206，服务器定时向多个客户端广播第一请求包括：服务器获取帧同步参数；按照帧同步参数获取多个客户端发送的数据包；服务器对多个客户端发送的数据包进行打包处理，得到第一数据包。

服务器获取帧同步参数，帧同步参数用于指示每秒传输的帧数。

帧同步参数在目标场景创建的时候就传入，用于指示每秒传输的帧数，比如，帧同步参数为 15 帧/s，用于指示每秒传输 15 帧。

在获取帧同步参数之后，服务器按照帧同步参数获取多个客户端发送的数据包，可以自发出去的第 1 帧开始，对当前 FrameId 进行自增策略，获取多个客户端发送的数据包。

服务器对多个客户端发送的数据包进行打包处理，得到第一数据包，第一数据包至少包括：每个客户端的标识信息、当前帧号、每个客户端发送的数据包。

在按照帧同步参数获取多个客户端发送的数据包之后，服务器对多个客户端发送的数据包进行打包处理，得到第一数据包，可以在 n 帧和 n+1 帧之间，对所有客户端的输入数据包进行打包处理，得到第一数据包。

在对多个客户端发送的数据包进行打包处理，得到第一数据包之后，

服务器向多个客户端广播第一数据包，也即，将第一数据包广播至目标场景中的所有客户端。

可选地，该实施例的服务器向客户端下发的第一数据包包括：每个客户端的标识信息、当前帧号、每个客户端发送的数据包，其中，每个客户端的标识信息用于唯一标识每个客户端，每个客户端发送的数据包为客户端的输入信息。

该实施例通过上述方法实现了定时向多个客户端广播第一请求的目的，达到了帧同步的技术效果。

作为一种可选的实施方式，服务器向多个客户端广播第一数据包包括：在第一数据包超过预设传输量的情况下，服务器在当前帧中向多个客户端广播在预设传输量之内的第一数据包，其中，超出预设传输量的第一数据包在当前帧的下一帧中进行打包处理。

该实施例的预设传输量可以为最大传输单元（Maximum Transmission Unit，简称为 MTU），为通信协议的某一层上面所能通过的最大数据报大小，该预设传输量可以以字节为单位。在向多个客户端广播第一数据包时，在第一数据包超过预设传输量的情况下，在当前帧中向多个客户端广播在预设传输量之内的第一数据包。可选地，在在 n 帧和 $n+1$ 帧之间，如果 m 个客户端输入（input）了 k 个消息，则服务器打包得到 $m*k$ 个消息，如果超过了 MTU，则将超出的消息放到下一帧去进行打包。在 n 帧和 $n+1$ 帧之间，如果 m 个客户端没有输入，则自增当前 FrameId，同时广播所有客户端，但不打包任何消息。

作为一种可选的实施方式，步骤 S206，服务器定时向多个客户端广播第一请求包括：在多个客户端中的至少一个客户端接收到的第一数据包的第一序列号与至少一个客户端之前接收的数据包的第三序列号不连续的情况下，服务器接收至少一个客户端发送的第七请求；响应第七请求，并向至少一个客户端发送第四序列号标识的数据包。

在多个客户端中的至少一个客户端接收到的第一数据包的第一序列号与至少一个客户端之前接收的数据包的第三序列号不连续的情况下，服务器接收至少一个客户端发送的第七请求，其中，第七请求用于请求获取第四序列号标识的数据包，第一序列号用于标识第一数据包，第三序列号用于标识至少一个客户端之前接收的数据包，第四序列号位于第一序列号与第三序列号之间，且与第一序列号和第三序列号连续。

在该实施例中，服务器和客户端的接口进行配合，以保证数据包的序列号是连续的。如果数据包的序列号不连续，则会主动发送请求以获取丢失的帧（LostFrame）。

在接收至少一个客户端发送的第七请求之后，服务器响应第七请求，并向至少一个客户端发送第四序列号标识的数据包，从而实现丢包补偿策略。

可选地，帧同步默认冗余 2 帧信息，最大冗余 3 帧信息，可以根据用户网络环境自适应切换。

该实施例通过上述方法实现了定时向多个客户端广播第一请求的目的。

作为一种可选的实施方式，通过服务器接口和客户端接口调用数据处理方法。

可选地，该实施例的客户端通过 Tconnd 访问用于创建目标场景的服务器 RoomSvr，客户端到 Tconnd 之间采用用户数据报协议 UDP 透传模式，Tconnd 上不进行数据加解密，加解密的工作由 RommSvr 完成。其中，Tconnd 为一款 IEG 研发部自主研发的网络接入组件进程，可以把并发的网络连接（TCP/UDP）数据，转换成对内网进程的消息队列请求，用于解耦网络连接处理和业务逻辑，为接入层，无状态，进程重启不影响服务功能。

RoomMng 主要功能是收集各个 RommSvr 的状态，管理房间的创建，

具体房间信息是存在 Tcaplus 中,而不是在 RoomMng 的内存中,进程 Crash 后可以立即重启并向 RoomSvr 拉取信息,迅速恢复服务。

RoomSvr 上存有房间的具体信息,以及该房间所有产生的帧数据,进程 Crash 后,对应房间的信息将丢失,由于 PVP 一场对局持续时间不会太久,所以影响不大,让业务侧重新创建房间即可。Tconnd 为接入层,无状态,进程重启不影响服务功能。

该实施例可以通用于需要帧同步机制的实时同步游戏,也可以用于最简单的网络聊天室应用。在游戏应用中,除了支持帧同步,也支持部分状态同步的同步实现方法,使用起来非常灵活。该实施例可以使没有能力,或者不想自己编写帧同步机制的游戏开发团队,也能很简单的使用帧同步这种技术,来开发自己的游戏。并且使用这种技术,并不需要源代码上的授权,仅仅接入本系统的 SDK 即可。

下面结合一种可选的实施例对本申请的技术方案进行说明,具体以目标事件为游戏事件,目标场景为房间进行举例说明。

图 3 是根据本申请实施例的一种数据处理的结构示意图。如图 3 所示,数据处理包括中继管理进程 (RelayMgr)、中继服务器 (RelaySvr)、接入组件进程 (Tconnd)、客户端 (Client)、游戏服务器 (GameSvr)、服务接口 (LockStep Service Api),中继管理进程与服务接口之间进行交互,实现创建房间和关闭房间,中继服务器向服务器接口传输数据帧,接入组件与客户端之间传出数据帧,客户端与游戏服务器之间采用游戏逻辑。

一般游戏开发者,需要在客户端和服务端两侧都接入该实施例描述的通用实时同步服务器系统,简称为服务器系统。在该实施例中,服务器系统主要提供游戏同步所需的广播能力。下面对其进行详细介绍:

下面对房间管理进行介绍。

针对任何一局游戏,参与的玩家都需要确定下来。也即,需要明确哪几个客户端需要广播同一局的数据。将需要一起工作的客户端称为进入了

一个虚拟的房间。在这个房间内，所有的客户端都可以发送消息，服务器系统会将这条消息向所有其他在这个房间里的客户端进行广播。由于游戏需要在运行时创建、销毁、进入、退出这些虚拟房间，所以服务器系统需要提供这些能力。在游戏逻辑中，一旦开启一场需要实时同步的战斗，就可以调用服务器系统的上述功能去创建一个房间，然后让客户端进入这个房间，以开始游戏的同步数据广播。

下面对广播数据的功能进行介绍。

广播数据是本系统最主要的功能，但该实施例不只是收到一个请求包就立刻广播，而且还支持以帧同步的方式进行广播。这两种广播功能，通过两个不同的接口提供。其中，帧同步的广播方式为，服务器不管有没有接收到客户端的请求，都会定时（频率可配置，一般为 0.1 秒到 0.05 秒）对同一房间内的其它客户端进行广播；如果任何数量客户端发送请求来，服务器都会保存这些需要广播的请求，再下一次定时广播的包内集中起来，最后发到所有客户端上。可选地，普通的即时广播，可以用来做游戏中的频道聊天、准备状态通知等同步性要求不高的功能；帧同步则可以同时作为游戏中动作同步的实现。

图 4 是根据本申请实施例的一种同时支持帧同步和即时广播功能的方法的流程示意图。如图 4 所示，客户端 1 (1P Client) 向中继服务器 (Relay Server) 广播输入数据 (Input)，中继服务器将客户端 1 的输入数据广播至客户端 2 (2P Client)。

图 5 是根据本申请实施例的一种帧同步网络广播的方法的流程示意图。如图 5 所示，客户端 1P 向服务器端发送数据 Opr1 和 Opr4，客户端 2P 向服务器端发送数据 Opr2 和 Opr3，服务器端将客户端 1P 发送的数据 Opr1 和 Opr4 和客户端 2P 发送的数据 Opr1 和 Opr4 以及空帧 (Empty) 向客户端 1P 和客户端 2P 广播，实现了服务器不管有没有收到客户端的请求，都定时对同一房间内的其它客户端进行广播。

下面对弱网络优化的方法进行介绍。

在实时同步动作游戏开发中，网络会出现延迟波动、偶发丢包、短暂断线这一类问题。该实施例针对上述问题做了优化，而不需要开发者过多关注。

图 6 是根据本申请实施例的一种数据冗余多发的方法的流程示意图。如图 6 所示，该实施例包括客户端 1 (Client1)、客户端 2 (Client2)、客户端 3 (ClientN)，客户端 1 向云服务器 (Lock Step Cloud) 输入数据 Input1、Input2、Input3，客户端 2 输入数据 Input1、Input2，客户端 3 输入数据 Input1，拂去其返回第一帧数据 Frame1、Frame2、Frame3、其中，Frame1 包括数据 Client1.Input1、Client2.Input1、Client3.empty；Frame2 包括数据 Client1.Input2、Client2.empty、Client3.empty；Frame3 包括数据 Client1.Input3、Client2.Input2、Client3.Input1。使用冗余多发的策略来降低丢包带来的影响，在帧同步广播的时候，每个数据包，实际上都会重发 2 至 3 遍，也即，每个数据包都会带上之前两次应该发的数据包的内容。这个重发的次数是可以设置的，从而解决丢包数据传输过程中的丢包问题。

对于由于弱网络环境而引起的断线重连的问题，由于在断线之后，大量的数据包丢失，此系统会在服务器端记录下整局游戏所广播过的所有包，客户端在断线重新连接之后，会自动从上次断线的最后一个广播包的序列号开始，重新下载所有记录过的数据包，然后重新加速运算整局游戏。客户端也可以自定义从某一个序列号开始下载，这适用于游戏自己有实现状态快照的情况，也即，定时在客户端记录下游戏中的状态情况，这样可以通过恢复最近的状态快照，然后从此状态快照处 (根据广播包序列号确定) 开始下载丢失的广播数据。这个功能也可以用来做游戏的录像重播，由于只记录的游戏的操作数据，所以这种游戏录像比视屏录像保存的文件小很多。

图 7 是根据本申请实施例的一种断线重连的方法的流程示意图。如图 7 所示，由于弱网络环境，导致中继服务器 (Relay Server) 的多个包未被客户端 1 (1P Client) 处理，因而需要客户端 1 快进处理，客户端 1 在断

线重新连接之后，会自动从上次断线的最后一个广播包的序列号开始，重新下载所有记录过的数据包，然后重新加速运算整局游戏，最终与客户端 2（2P Client）的显示结果一致。

最后要解决的是弱网络环境下的延迟波动，这个问题直接影响游戏画面的忽快忽慢，让玩家难以准确把握输入和响应之间的提前量。可以在客户端处建立一定大小的缓冲区，在收到网络广播包后不立刻通知游戏逻辑代码，而是先放入缓冲区，然后定时将缓冲区的数据包通知给游戏客户端代码。

图 8 是根据本申请实施例的一致缓存发包的方法的流程示意图。如图 8 所示，客户端（Client）中的游戏逻辑（Game Logic）向中继服务器（Realy Server）发送操作，中继服务器接收到操作之后，广播收到的操作，客户端在收到操作后不立刻通知游戏逻辑代码，而是先放入缓冲区（Cache），等累积到固定个数之后，定时将缓冲区的数据包通知给游戏客户端代码。这样客户端代码得到的数据包，就会在一定时间段内速度是比较稳定的。虽然这会增加整体网络数据包的延迟，但是能让所有的网络包的到达速度保持一致，让玩家形成一个稳定的操作手感。

下面对逻辑校验进行介绍。

由于网络游戏一般都要面对外挂和破解的威胁，所以游戏的实时战斗过程，往往也需要由服务器进行合法性验证。该实施例将所有广播给客户端的数据，都同样的广播一份给开发者所设置的游戏服务器（GameServer），这样游戏服务器就能实时地掌握所有客户端接收到的数据包，从而进行任何自己想要的验证鉴权。当游戏服务器发现有某个客户端作弊，开发者不但可以立刻将有问题的客户端“踢出”本局游戏，而且还可以获得完整的作弊记录，用作日后对作弊客户端的进一步惩罚和预防。同样，游戏服务器也可以利用这个功能，来进行敏感操作的服务器端运算，比如，需要在游戏中掉落战利品，这个操作可以由游戏服务器来完成，然后同样利用广播接口发送给所有客户端，这些掉落的战利品的数据信息。

该实施例的游戏服务器端接口，实际上拥有客户端接口的全部功能，再额外增加了房间管理的能力。

以上功能，用户在使用的时候，需要通过客户端接口和服务器端接口来调用

下面结合另外一种优选的实施例对本申请的技术方案进行说明。

图 9 是根据本申请实施例的一种服务器端的架构图。如图 9 所示，该服务器段架构图包括存储系统（Tcaplus）、房间管理服务（RoomMng）、房间服务器（RoomSvr）、接入组件进程（Tconnd）、客户端（Client）、接入开发工具包（LockStepFrame）、游戏服务器（GameSvr）、服务接口（Apollo Service Api）。房间管理服务与服务器接口之间的数据传输协议为传输控制协议，还可以与游戏服务器的数据传输协议为超文本传输协议。

RoomMng：房间管理服务，为 GameSvr 游戏服务器提供创建房间，销毁房间，查询房间的功能。

RoomSvr：为帧同步房间服务，为同一 Room（一次 PVP 即玩家互相利用游戏资源攻击而形成的互动竞技对局）的玩家提供收集帧数据、组帧、广播帧数据的功能。一个 RoomSvr 可以同时为多个房间服务。

Tconnd：作为 RoomSvr 的接入层，为客户端玩家提供 UDP 接入功能。

LockStepFrame：面向客户端的接入 SDK，提供建立连接，发送帧数据，关闭连接等接口。

Apollo Service Api：面向 GameSvr 的接入 SDK，提供创建房间，销毁房间，查询房间接口。

Tcaplus：RoomMng 会把创建的房间信息存到 Tcaplus 中。

在客户端连上游戏服务器 GameSvr 大厅后，发起例如 5V5 对战，GameSvr 通过调用 LockStep Service 的接口创建 PVP 房间，LockStep Service 会返回各个成员的 RoomID，RoomKey 以及房间 Url，房间管理服务

务 GameSvr 将此信息返回给客户端,各个游戏客户端都会连到同一个 PVP RoomSvr, 当对局内玩家全部连上后, 开始进入帧同步转发逻辑, 也即, 收集每个客户端的动作, 组成一个完整的帧内容广播给对局内其他玩家。

图 10 是根据本申请实施例的一种游戏房间操作流程的示意图。如图 10 所示, 客户端登录游戏, 发起对战请求, 游戏服务器 (GameSvr) 通过 CreatRoom () 函数创建房间, 房间管理服务 (RoomMng) 将创建房间的信息向房间服务器 (RoomSvr) 传输, 房间服务器向房间管理服务返回房间信息, 房间管理服务将房间信息向游戏服务器传输, 客户端从游戏服务器获取房间信息。客户端登录 PVP 房间, 客户端正式对战前进行消息同步, 例如, 加载进度的同步, 也即, 加载速度同步, 房间服务器广播其他客户端的数据, 客户端通知房间服务器准备完毕, 房间服务器等待客户端全部准备完成, 通知所有客户端战斗开始。客户端向房间服务器发送帧数据, 房间服务器等待玩家的帧数据, 组帧, 向所有客户端广播帧数据。在客户端发生断线的情况下, 客户端向房间服务器请求补帧, 房间服务器向客户端发送补帧数据, 在游戏结束时, 客户端退出房间。

表 1 存储数据结构表

字段名	类型	描述
room_id	uint64	房间 ID, 主键
business_id	uint32	业务 ID
process_id	uint32	房间所属 roomsvr 进程 ID
access_url	string(64)	客户端接入房间的地址
member_count	uint32	房间人数

member_info	blob(2564)	一个 member 的信息为 4 字节 objectid+16 字节 memberkey=20 字节 2564 可以存下 128 个 member 信息
create_time	uint64	房间创建时间戳
last_update_time	uint64	最后更新时间戳

表 1 为本申请实施例的一种存储数据结构表。房间 ID，主键的字段名为 room_id，类型为 uint64；业务 ID 的字段名为 business_id，类型为 uint32；房间所属 roomsvr 进程 ID 的字段名为 process_id，类型为 uint32；客户端接入房间的地址的字段名为 access_url，类型为 string(64)；房间人数的字段名为 member_count，类型为 uint32；一个 member 的信息为 4 字节 objectid+16 字节 memberkey=20 字节 2564 可以存下 128 个 member 信息的字段名为 member_info，类型为 blob(2564)；房间创建时间戳的字段名为 create_time，类型为 uint64；最后更新时间戳的字段名为 last_update_time，类型为 uint64。

下面对房间创建流程进行介绍。

一个房间对应于一次 PVP 对局，房间存在于 RoomSvr 中，一个 RoomSvr 可以支撑多个房间。RoomSvr 会定期向所有 RoomMng 上报自身的信息，包括总共支持多少房间、当前已创建了多少房间等。当 GameSvr 发起创建房间的命令时，该请求会随机落在一个 RoomMng 上，RoomMng 收到请求后，会综合当前所有 RoomSvr 的负载情况，选择一个 RoomSvr 发送创建房间的命令。RoomSvr 收到创建房间的命令后，会创建房间，并给房间的每个 member 分配一个随机 key，该 Key 将用于后续客户端到 RoomSvr 的消息加解密。

RoomID 为 int64 类型，前 32 位为 RoomSvr 进程 ID，后 32 位为进程内自增序列，RoomKey 为随机生成的字符串，长度为 32。

RoomMng 在收到 RoomSvr 成功创建房间的应答后，会将这个房间的信息写入 TCAPLUS，信息包括 RoomID，业务 ID，成员数，RoomSvr 的进程 ID，接入地址，各成员的 key。

下面对房间销毁进行介绍。

房间销毁的途径包括以下 4 种：

GameSvr 主动销毁房间，这个包括对战正常结束，或者游戏发现有玩家作弊而异常关闭的，消息路径为 GameSvr->RoomMng->Tcaplus；

RoomSvr 接收到了所有客户端退出房间的消息，而关闭房间，这个是对应正常的 PVP 战斗结束流程，消息路径为 RoomSvr->RoomMng->Tcaplus；

RoomSvr 上因为房间若干时间没消息通信而关闭房间，这种主要为防止 PVP 已经正常结束，但是 GameSvr 和客户端没有主动调用关闭房间的情况，消息路径为 RoomSvr->RoomMng->Tcaplus；

自动扫描工具扫描 Tcaplus 表中超过 2 天前建立的房间，将其关闭，这种是为防止 RoomSvr 重启导致房间信息丢失的情况，消息路径为 CleanTool->Tcaplus。

图 11 是根据本申请实施例的一种消息分发流程的示意图。如图 11 所示，在对局登录（Login）时，客户端 1 登录房间服务器（RoomSvr），房间服务器对局登录会对 MemberKey 进行校验（Verify），在校验失败的情况下，回复错误信息。客户端 2 登录房间服务器。在登录阶段，LockStepSvr 对局的状态标记为 Login，这个时候 LockStepSvr 会对客户端 Input 进行转发，但 FrameId 标记为 0，并且 FrameId 为 0 的数据不会在 LockStepSvr 保留，也就是断线重连上来的客户端不会收到 FrameId 为 0 的数据。

客户端 1 在正式对战前的消息同步，比如加载进度等。

在对局开始时，对局开始的时机为 LockStepSvr 收到所有客户端的 BattleStart 请求才开始正式的对局转发。在其他玩家都登录完毕后，开始向服务器（SVR）发送战斗开始请求（BattleStart），此时 FrameId 标记为 1，表示帧同步正式开始。可能会存在这样的情况，3v3 对局，只收到了 5 个 BattleStart 请求，出现这种情况，LockStepSvr 会触发一个超时机制，如收到第一 BattleStart 请求之后 10 秒（具体时间 GameSvr 指定），也会启动帧同步流程，此时 LockStepSvr 状态为 LockStep。在向服务器发送战斗开始请求之后，等待其他客户端准备。

房间服务器在等到收到所有客户端的开始（START）命令后，开始向客户端通知（BattleStart Notify）。

客户端 1 和客户端 2 向房间服务器输入帧（Frame Input），房间服务器组帧，得到缓冲（cache）帧数据。房间服务器向客户端 1 和客户端 2 发送完整的帧数据（Complete Frame Data）。客户端 1 和客户端 2 在接收到房间服务器发送的房间服务器之后，退出登录（LogOut）房间服务器。

可选地，在对局重连时，对局重连分为两种情况，第一种情况是玩家因为网络断线，客户端调用 Login 接口重新登录，同时调用 RequestFrame(LastFrameId)来获取丢失的帧数据。第二种情况是玩家进程退出，房间 URL、MemberKey、FrameId 等信息丢失，这种情况需要 GameSvr 去 RoomSvr 查询获得相关信息然后重新 Login，并从第一帧开始同步数据。

在对局结束时，客户端调用 Stop 接口，LockStep 标示该链接为 Stop 状态，此时将丢弃该客户端的 Input 输入，并且收到多数客户端的 Stop 请求之后，房间进入销毁状态。

GameSvr 在创建房间的时候会传入帧同步参数，比如，15 帧/s，LockStepSvr 从发出去第 1 帧开始对 FrameId 进行自增策略。在 n 帧和 n+1 帧之间，对所有客户端输入进行打包，然后广播给所有在局用户。服务器下发给客户端信息主要包括 3 部分，ObjectId（客户端的唯一表示）、FrameId（当前帧号）以及客户端的 Input 信息。

在 n 帧和 $n+1$ 帧之间, 如果 m 个客户端 input 了 k 个消息, 那么 LockStepSvr 将打包 $m*k$ 个消息, 如果超过了 MTU, 将部分消息放到下一帧去进行打包。

在 n 帧和 $n+1$ 帧之间, 如果 m 个客户端没有 input 输入, 那么 LockStepSvr 会自增 FrameId, 同时广播所有客户端, 但不打包任何消息。

LockStepSvr 和客户端 API 配合保证使用者接受到的消息一定是帧序号连续的, 客户端 API 如果发现帧序号不连续, 会主动请求丢失的帧 (LostFrame), 使用者无需关心丢帧问题。

帧同步默认冗余 2 帧信息, 最大冗余 3 帧信息, 根据用户网络环境自适应切换。

客户端通过 Tconnd 访问 RoomSvr, 客户端到 Tconnd 之间采用 Udp 透传模式, Tconnd 上不进行数据加解密, 加解密的工作由 RommSvr 完成。

RoomMng 主要功能是收集各个 RommSvr 的状态, 管理房间的创建, 具体房间信息是存在 Tcaplus 中, 而不是在 RoomMng 的内存中, 进程 Crash 后可以立即重启并向 RoomSvr 拉取信息, 迅速恢复服务。

RoomSvr 上存有房间的具体信息, 以及该房间所有产生的帧数据, 进程 Crash 后, 对应房间的信息将丢失, 由于 PVP 一场对局持续时间不会太久, 所以影响不大, 让业务侧重新创建房间即可。

Tconnd 为接入层, 无状态, 进程重启不影响服务功能。

RoomMng 可平行扩展, 内存和 CPU 消耗不会太大, 在一个集群里部署两个节点即可。

RoomSvr 可平行扩展, 根据业务需要可动态扩容。

图 12 是根据本申请实施例的客户端的登录流程的示意图。如图 12 所示, 游戏客户端 (Game Client) 需要为战斗游戏创建房间, 游戏服务器 (Game Server) 向服务器系统 (LockStep Server) 发送请求创建房间的请

求，服务器系统向游戏服务器返回响应请求创建房间的信息列表，游戏客户端向游戏服务器发送进入房间的请求，游戏服务器向游戏客户端发送响应进入房间的请求的结果，游戏客户端登录客户端系统，与服务器系统相连接，如果服务器系统需要客户端系统重新登录，客户端系统向服务器系统发送重新登录请求，服务器系统返回响应重新登录请求的结果（需要说明的是，图 12 中的序号 0 至 9 用于表示各个步骤执行的先后顺序）。

图 13 是根据本申请实施例的一种开始流程的示意图。如图 13 所示，游戏客户端 1（Game Client1）向客户端系统（LockStep Client）发送广播（SendBroadcast）消息，客户端系统向服务器系统（LockStep Server）发送广播消息，服务器系统向客户端系统发送对广播消息的应答信息，客户端系统向客户端 1 发送广播消息的应答信息。

游戏客户端 N 向服务器系统发送广播消息，服务器系统响应广播消息，并向游戏客户端 N 发送广播消息的应答消息，游戏客户端 N 在收到第一帧（PopFrame），即可以认为游戏已经开始，游戏也可以访问 HasStarted 属性判断对局是否开始（需要说明的是，图 13 中的序号 1 至 3、1' 和 2' 用于表示各个步骤执行的先后顺序）。

图 14 是根据本申请实施例的一种帧同步流程的示意图。如图 14 所示，游戏客户端 1 向客户端系统发送输入数据，客户端系统向服务器系统发送输入数据，服务器系统将输入数据组帧，并将组帧后的数据包向客户端系统发送，游戏客户端收到第一帧（PopFrame）数据。游戏客户端 N 向服务器系统发送输入数据，服务器系统向客户端 N 返回第一帧数据（需要说明的是，图 14 中的序号 1 至 5、1' 和 5' 用于表示各个步骤执行的先后顺序）。

该实施例的帧同步只有满足以下条件之一才会开始：所有玩家都 Ready；LockStep Server 超时；满足 LockStep Server 的其它机制。

帧数据由 LockStep Server 按照自己的定时器在固定的时间产生，即使没有任何用户输入，也会生成空帧。

客户端会自动检查每个帧同步包的序号，如果出现数据包丢失，则启动 TCP 的下载通道，向 RelayServer 去下载缺失的帧，而这个时候 PopFrame()接口将不会返回帧数据，直到所有的缺失帧补齐，才按顺序返回完整的帧数据。

图 15 是根据本申请实施例的一种广播流程的示意图。如图 15 所示，游戏客户端 1 发送广播信息 (SendBroadcast)，客户端系统向服务器系统发送广播信息，服务器系统向客户端系统返回广播数据，客户端系统向游戏客户端 1 发送广播数据。

游戏客户端 2 向服务器系统发送广播消息，服务器系统向游戏客户端 2 发送广播数据（需要说明的是，图 15 中的序号 1 至 4、1' 和 4' 用于表示各个步骤执行的先后顺序）。

该实施例的广播通道不依赖于游戏是否开始，只要成功登录之后就可以进行收发数据。广播数据一旦到达客户端的软件开发工具包，就会立刻回调用户定义的收包函数进行处理。

该实施例的数据处理方法的通用性好，可以用于需要帧同步机制的实时同步游戏，也可以用于最简单的网络聊天室应用。在游戏应用中，除了支持帧同步，也支持部分状态同步的同步实现方法，使用起来非常灵活。该实施例可以让没有能力，或者不想自己编写帧同步机制的游戏开发团队，也能很简单的使用帧同步这种技术，来开发自己的游戏。并且使用这种技术，并不需要源代码上的授权，仅仅接入本系统的 SDK 即可。

该实施例在网络优化方面，集成了很多实时同步玩法游戏所需要的功能，比如，可变的冗余 UDP 包，用以降低丢包带来的体验下降；自动下载缺失帧和 UDP 冗余这两项设计，承担起丢包、断线的网络包恢复工作，使得开发者不需要再自行开发相关的网络操作代码，大大降低了开发成本；

该实施例的另外一个突出优点是，可以把网络同步的服务，从一般的游戏服务器集群中分离开来，形成一个可以单独部署和运维的集群。与一

般的游戏引擎提供的广播服务有本质区别的是，这种服务器系统是完全集群化的，也即，可以无限扩充其承载能力，并且也拥有运行时扩容、动态容灾、负载平衡这些集群服务固有的好处。最重要的是，这些好处都不需要游戏开发者做任何开发工作。这让游戏项目可以仅仅是通过接入本系统的云服务，就可以拥有强大的承载能力，这对于游戏项目的运营有非常大的便利性。同时以云的方式，多个游戏项目可以共享同一套网络同步集群，能最大限度减少硬件资源的浪费。

在一些网络游戏中，有一些使用了 p2p 的数据广播方式，代替从服务器端定时广播。这种方案同样还是需要房间管理、游戏服务器验证等功能，只不过是广播数据的通道变化了而已，这依然是在该实施例的主体设计之内。

图 16 是根据本申请实施例的一种 P2P 方式的同步游戏的示意图。如图 16 所示，多个手机客户端进入房间，向房间管理服务器发送游戏数据，房间管理服务器向游戏服务器发送同步游戏数据，游戏服务器验证同步游戏数据的合法性，其中，多个手机客户端之间的传输为 P2P 传输同步游戏数据。

另外还有一些方案，采用的接口不是“收取”“发送”形式的函数，而是通过“状态参数”同步的形式来实现。

图 17 是根据本申请实施例的一种参数同步的接口方法的流程图。如图 17 所示，游戏状态同步系统、多个手机客户端，其中，多个手机客户端中包括参数 A。手机客户端通过参数 A 向游戏状态同步系统发送参数修改事件，游戏状态同步系统通过参数 A 接收参数修改事件，多个手机客户端中的其它客户端的参数和发送修改参数事件的手动客户端中的参数看起来是同步的。这种方案实际上，只是把状态参数的修改事件，进行了广播，然后对此事件做了一层包装。这种方案底层还是存在和本系统相同的“收取”、“发送”形式的接口的。

该实施例的服务器系统，需要完成数据广播功能、实时同步游戏状态、

优化网络延迟丢包体验、提供玩家断线重连能力、对局游戏的重播等一系列任务。通过抽象实时对战玩法的共性，统一地完成了上述常见的功能，提供了一套接入即可用的服务器系统，令游戏开发团队，仅需要开发具体游戏的客户端代码，即可快速实现网络实时对战功能，降低了数据处理的开发成本。

需要说明的是，对于前述的各方法实施例，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，本申请并不受所描述的动作顺序的限制，因为依据本申请，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，所涉及的动作和模块并不一定是本申请所必须的。

通过以上的实施方式的描述，本领域的技术人员可以清楚地了解到根据上述实施例的方法可借助软件加必需的通用硬件平台的方式来实现，当然也可以通过硬件，但很多情况下前者是更佳的实施方式。基于这样的理解，本申请的技术方案本质上或者说对相关技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质（如ROM/RAM、磁碟、光盘）中，包括若干指令用以使得一台终端设备（可以是手机，计算机，服务器，或者网络设备等）执行本申请各个实施例所述的方法。

根据本申请实施例的又一方面，还提供了一种用于实施上述数据处理方法的数据处理装置，应用于服务器。图 18 是根据本申请实施例的一种数据处理装置的示意图。如图 18 所示，该装置可以包括：创建单元 10、确定单元 20 和广播单元 30。

创建单元 10，设置为创建用于执行目标事件的目标场景，其中，目标事件为实时对战类事件。

确定单元 20，设置为确定处于目标场景中的多个客户端，其中，多个

客户端用于执行目标事件。

广播单元 30，设置为定时向多个客户端广播第一请求，其中，第一请求用于请求多个客户端同步执行第一事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包，目标事件包括第一事件。

需要说明的是，该实施例中的创建单元 10 可以设置为执行本申请实施例中的步骤 S202，该实施例中的确定单元 20 可以设置为执行本申请实施例中的步骤 S204，该实施例中的广播单元 30 可以设置为执行本申请实施例中的步骤 S206。

该实施例由于定时向多个客户端广播第一请求，以使多个客户端同步执行第一事件，达到了通用广播服务器的帧同步广播的目的，从而实现了降低数据处理的开发成本的技术效果，进而解决了相关技术中数据处理的开发成本高的技术问题。

此处需要说明的是，上述单元与对应的步骤所实现的示例和应用场景相同，但不限于上述实施例 1 所公开的内容。需要说明的是，上述模块作为装置的一部分可以运行在如图 1 所示的硬件环境中，可以通过软件实现，也可以通过硬件实现。其中，硬件环境包括网络环境。

根据本申请实施例的又一方面，还提供了一种用于实施上述数据处理方法的电子装置，应用于服务器或终端。

图 19 是根据本申请实施例的一种终端的结构框图。如图 19 所示，该终端可以包括：一个或多个（图中仅示出一个）处理器 191、存储器 193、以及传输装置 195，如图 19 所示，该终端还可以包括输入输出设备 197。

其中，存储器 193 可用于存储软件程序以及模块，如本申请实施例中的数据处理方法和装置对应的程序指令/模块，处理器 191 通过运行存储在存储器 193 内的软件程序以及模块，从而执行各种功能应用以及数据处理，即实现上述的数据处理方法。存储器 193 可包括高速随机存储器，还可以

包括非易失性存储器，如一个或者多个磁性存储装置、闪存、或者其他非易失性固态存储器。在一些实例中，存储器 193 可进一步包括相对于处理器 191 远程设置的存储器，这些远程存储器可以通过网络连接至终端。上述网络的实例包括但不限于互联网、企业内部网、局域网、移动通信网及其组合。

上述的传输装置 195 用于经由一个网络接收或者发送数据，还可以用于处理器 191 与存储器 193 之间的数据传输。上述的网络具体实例可包括有线网络及无线网络。在一个实例中，传输装置 195 包括一个网络适配器（Network Interface Controller，NIC），其可通过网线与其他网络设备与路由器相连从而可与互联网或局域网进行通讯。在一个实例中，传输装置 195 为射频（Radio Frequency，RF）模块，其用于通过无线方式与互联网进行通讯。

其中，具体地，存储器 193 用于存储应用程序。

处理器 191 可以通过传输装置 195 调用存储器 193 存储的应用程序，以执行下述步骤：

创建用于执行目标事件的目标场景，其中，目标事件为实时对战类事件；

确定处于目标场景中的多个客户端，其中，多个客户端用于执行目标事件；

定时向多个客户端广播第一请求，其中，第一请求用于请求多个客户端同步执行第一事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包，目标事件包括第一事件。

处理器 191 还用于执行下述步骤：接收第一数量的客户端发送的第一请求，其中，多个客户端包括第一数量的客户端；在接收到第一请求之后，在第一时间向多个客户端广播第一请求，其中，第一时间与上一次向多个客户端广播第二请求的时间相隔第一预设时间，第二请求用于请求多个客

户端同步执行第二事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第二事件的第二数据包，目标事件包括第二事件。

处理器 191 还用于执行下述步骤：当第一时间到达时，且未接收到任何多个客户端发送的请求的情况下，向多个客户端广播携带空帧的第一请求。

处理器 191 还用于执行下述步骤：分别获取前 N 次向多个客户端广播的请求中所携带的 N 个数据包，其中，N 次与 N 个数据包一一对应，N 为大于等于 1 的自然数；定时向多个客户端广播携带有 N 个数据包和第一数据包的第一请求。

处理器 191 还用于执行下述步骤：在定时向多个客户端广播第一请求之后，记录第一数据包；定时向多个客户端广播第三请求，其中，第三请求用于请求多个客户端同步执行第三事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第三事件的第三数据包，目标事件包括第三事件；记录第三数据包。

处理器 191 还用于执行下述步骤：在定时向多个客户端广播第一请求之后，在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，向第一客户端发送第一数据包和第三数据包，其中，第一客户端在失去连接的过程中未接收到第一数据包和第三数据包，多个客户端中的第一客户端失去连接包括：多个客户端中的第一客户端所处的网络断线；或者多个客户端中的第一客户端退出事件执行进程。

处理器 191 还用于执行下述步骤：在定时向多个客户端广播第一请求之后，在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，接收客户端发送的第四请求，其中，第四请求用于请求获取第一数据包或第三数据包，第一客户端在失去连接的过程中未接收到第一数据包和第三数据包；响应第四请求，并向第一客户端发送第一数据包或第三数据包。

处理器 191 还用于执行下述步骤：在定时向多个客户端广播第一请求时，验证第一数据包是否合法；在验证出第一数据包不合法的情况下，禁止发送第一数据包的客户端执行目标事件，和/或记录用于指示第一数据包不合法的信息。

处理器 191 还用于执行下述步骤：向第一服务器广播第一数据包；通过第一服务器验证第一数据包是否合法。

处理器 191 还用于执行下述步骤：获取每个客户端的操作信息；将每个客户端的操作信息组合为数据帧；定时向多个客户端广播携带有数据帧的第一请求。

处理器 191 还用于执行下述步骤：获取多个第二服务器的场景信息，其中，每个第二服务器的场景信息包括用于指示每个第二服务器允许创建的场景和已经创建的场景的信息；接收场景创建请求，并根据多个第二服务器的场景信息确定用于创建目标场景的目标服务器，其中，场景创建请求用于请求创建目标场景；向目标服务器发送创建指令，其中，创建指令用于指示目标服务器创建目标场景；在目标服务器接收到创建指令之后，通过目标服务器创建目标场景。

处理器 191 还用于执行下述步骤：在确定处于目标场景中的多个客户端之后，对多个客户端的登录信息进行校验；在对至少一个客户端的登录信息校验失败的情况下，发送校验失败信息；在对多个客户端的登录信息校验成功的情况下，将多个客户端发送的数据设置为空帧。

处理器 191 还用于执行下述步骤：接收多个客户端发送的第五请求，其中，第五请求用于请求开始执行目标事件；响应第五请求，并允许定时向多个客户端广播第一请求。

处理器 191 还用于执行下述步骤：在未全部接收到多个客户端发送的第五请求的情况下，在第三预设时间允许定时向多个客户端广播第一请求。

采用本申请实施例，提供了一种数据处理的方案。由于定时向多个客

户端广播第一请求,以使多个客户端同步执行第一事件,达到了通用广播服务器的帧同步广播的目的,从而实现了降低数据处理的开发成本的技术效果,进而解决了相关技术中数据处理的开发成本高的技术问题。

可选地,本实施例中的具体示例可以参考上述实施例中所描述的示例,本实施例在此不再赘述。

本领域普通技术人员可以理解,图 19 所示的结构仅为示意,终端可以是智能手机(如 Android 手机、iOS 手机等)、平板电脑、掌上电脑以及移动互联网设备(Mobile Internet Devices, MID)、PAD 等终端设备。图 19 其并不对上述电子装置的结构造成限定。例如,终端还可包括比图 19 中所示更多或者更少的组件(如网络接口、显示装置等),或者具有与图 19 所示不同的配置。

本领域普通技术人员可以理解上述实施例的各种方法中的全部或部分步骤是可以通过程序来指令终端设备相关的硬件来完成,该程序可以存储于一计算机可读存储介质中,存储介质可以包括:闪存盘、只读存储器(Read-Only Memory, ROM)、随机存取器(Random Access Memory, RAM)、磁盘或光盘等。

根据本申请的实施例的又一方面,还提供了一种存储介质。可选地,在本实施例中,上述存储介质可以用于执行数据处理方法的计算机程序。

可选地,在本实施例中,上述存储介质可以位于上述实施例所示的网络中的多个网络设备中的至少一个网络设备上。

可选地,在本实施例中,存储介质被设置为存储用于执行以下步骤的计算机程序:

创建用于执行目标事件的目标场景,其中,目标事件为实时对战类事件;

确定处于目标场景中的多个客户端,其中,多个客户端用于执行目标事件;

定时向多个客户端广播第一请求，其中，第一请求用于请求多个客户端同步执行第一事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包，目标事件包括第一事件。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：接收第一数量的客户端发送的第一请求，其中，多个客户端包括第一数量的客户端；在接收到第一请求之后，在第一时间向多个客户端广播第一请求，其中，第一时间与上一次向多个客户端广播第二请求的时间相隔第一预设时间，第二请求用于请求多个客户端同步执行第二事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第二事件的第二数据包，目标事件包括第二事件。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：当第一时间到达时，且未接收到任何多个客户端发送的请求的情况下，向多个客户端广播携带空帧的第一请求。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：分别获取前N次向多个客户端广播的请求中所携带的N个数据包，其中，N次与N个数据包一一对应，N为大于等于1的自然数；定时向多个客户端广播携带有N个数据包和第一数据包的第一请求。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：在定时向多个客户端广播第一请求之后，记录第一数据包；定时向多个客户端广播第三请求，其中，第三请求用于请求多个客户端同步执行第三事件，且携带有由多个客户端中的至少一个客户端发送的用于执行第三事件的第三数据包，目标事件包括第三事件；记录第三数据包。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：在定时向多个客户端广播第一请求之后，在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，向第一客户端发送第一数据包和第三数据包，其中，第一客户端在失去连接的过程中未接收到第一数据包和第三数据包，多个客户端中的第一客户端失去连接包

括：多个客户端中的第一客户端所处的网络断线；或者多个客户端中的第一客户端退出事件执行进程。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：在定时向多个客户端广播第一请求之后，在多个客户端中的第一客户端失去连接，且在记录第三数据包之后重新建立连接的情况下，接收客户端发送的第四请求，其中，第四请求用于请求获取第一数据包或第三数据包，第一客户端在失去连接的过程中未接收到第一数据包和第三数据包；响应第四请求，并向第一客户端发送第一数据包或第三数据包。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：在定时向多个客户端广播第一请求时，验证第一数据包是否合法；在验证出第一数据包不合法的情况下，禁止发送第一数据包的客户端执行目标事件，和/或记录用于指示第一数据包不合法的信息。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：向第一服务器广播第一数据包；通过第一服务器验证第一数据包是否合法。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：通过调用预设接口创建目标场景；和/或通过预设接口向多个客户端发送目标场景的信息，其中，目标场景的信息包括以下至少之一：目标场景的地址信息；目标场景的键值信息；目标场景的统一资源定位符。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：获取每个客户端的操作信息；将每个客户端的操作信息组合为数据帧；定时向多个客户端广播携带有数据帧的第一请求。

可选地，存储介质还被设置为存储用于执行以下步骤的计算机程序：获取多个第二服务器的场景信息，其中，每个第二服务器的场景信息包括用于指示每个第二服务器允许创建的场景和已经创建的场景的信息；接收场景创建请求，并根据多个第二服务器的场景信息确定用于创建目标场景的目标服务器，其中，场景创建请求用于请求创建目标场景；向目标服务

器发送创建指令，其中，创建指令用于指示目标服务器创建目标场景；在目标服务器接收到创建指令之后，通过目标服务器创建目标场景。

可选地，存储介质还被设置为存储用于执行以下步骤之一的计算机程序：在确定处于目标场景中的多个客户端之后，对多个客户端的登录信息进行校验；在对至少一个客户端的登录信息校验失败的情况下，发送校验失败信息；在对多个客户端的登录信息校验成功的情况下，将多个客户端发送的数据设置为空帧。

可选地，存储介质还被设置为存储用于执行以下步骤之一的计算机程序：接收多个客户端发送的第五请求，其中，第五请求用于请求开始执行目标事件；响应第五请求，并允许定时向多个客户端广播第一请求。

可选地，存储介质还被设置为存储用于执行以下步骤之一的计算机程序：在未全部接收到多个客户端发送的第五请求的情况下，在第三预设时间允许定时向多个客户端广播第一请求。

可选地，本实施例中的具体示例可以参考上述实施例中所描述的示例，本实施例在此不再赘述。

可选地，在本实施例中，上述存储介质可以包括但不限于：U盘、只读存储器（ROM，Read-Only Memory）、随机存取存储器（RAM，Random Access Memory）、移动硬盘、磁碟或者光盘等各种可以存储计算机程序的介质。

上述本申请实施例序号仅仅为了描述，不代表实施例的优劣。

上述实施例中的集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用，可以存储在上述计算机可读的存储介质中。基于这样的理解，本申请的技术方案本质上或者说对相关技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来，该计算机软件产品存储在存储介质中，包括若干指令用以使得一台或多台计算机设备（可为个人计算机、服务器或者网络设备）执行本申请各个实施

例所述方法的全部或部分步骤。

在本申请的上述实施例中，对各个实施例的描述都各有侧重，某个实施例中未详述的部分，可以参见其他实施例的相关描述。

在本申请所提供的几个实施例中，应该理解到，所揭露的客户端，可通过其它的方式实现。其中，以上所描述的装置实施例仅仅是示意性的，例如所述单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个系统，或一些特征可以忽略，或不执行。另一点，所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口，单元或模块的间接耦合或通信连接，可以是电性或其它的形式。

所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

另外，在本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。

以上所述仅是本申请的优选实施方式，应当指出，对于本技术领域的普通技术人员来说，在不脱离本申请原理的前提下，还可以做出若干改进和润饰，这些改进和润饰也应视为本申请的保护范围。

工业实用性

在本申请实施例中，服务器创建用于执行目标事件的目标场景，其中，目标事件为实时对战类事件；服务器确定处于目标场景中的多个客户端，其中，多个客户端用于执行目标事件；服务器定时向多个客户端广播第一请求，其中，第一请求用于请求多个客户端同步执行第一事件，且携带有

由多个客户端中的至少一个客户端发送的用于执行第一事件的第一数据包，目标事件包括第一事件，由于定时向多个客户端广播第一请求，以使多个客户端同步执行第一事件，达到了通用广播服务器的帧同步广播的目的，从而实现了降低数据处理的开发成本的技术效果。

权 利 要 求 书

1. 一种数据处理方法，包括：

服务器创建用于执行目标事件的目标场景，其中，所述目标事件为实时对战类事件；

所述服务器确定处于所述目标场景中的多个客户端，其中，所述多个客户端用于执行所述目标事件；

所述服务器定时向所述多个客户端广播第一请求，其中，所述第一请求用于请求所述多个客户端同步执行第一事件，且携带有由所述多个客户端中的至少一个客户端发送的用于执行所述第一事件的第一数据包，所述目标事件包括所述第一事件。

2. 根据权利要求 1 所述的方法，其中，所述服务器定时向所述多个客户端广播所述第一请求包括：

所述服务器接收第一数量的客户端发送的所述第一请求，其中，所述多个客户端包括所述第一数量的客户端；

所述服务器在接收到所述第一请求之后，在第一时间向所述多个客户端广播所述第一请求，其中，所述第一时间与上一次向所述多个客户端广播第二请求的时间相隔第一预设时间，所述第二请求用于请求所述多个客户端同步执行第二事件，且携带有由所述多个客户端中的至少一个客户端发送的用于执行所述第二事件的第二数据包，所述目标事件包括所述第二事件。

3. 根据权利要求 2 所述的方法，其中，所述服务器定时向所述多个客户端广播所述第一请求包括：

当所述第一时间到达时，且未接收到任何所述多个客户端发送的请求的情况下，所述服务器向所述多个客户端广播携带空帧的所述第一请求。

4. 根据权利要求 1 所述的方法，其中，所述服务器定时向所述多个客户端

端广播所述第一请求包括:

所述服务器分别获取前 N 次向所述多个客户端广播的请求中所携带的 N 个数据包, 其中, 所述 N 次与所述 N 个数据包一一对应, 所述 N 为大于等于 1 的自然数;

所述服务器定时向所述多个客户端广播携带有所述 N 个数据包和所述第一数据包的所述第一请求。

5. 根据权利要求 1 所述的方法, 其中, 在所述服务器定时向所述多个客户端广播所述第一请求之后, 所述方法还包括:

所述服务器记录所述第一数据包;

所述服务器定时向所述多个客户端广播第三请求, 其中, 所述第三请求用于请求所述多个客户端同步执行第三事件, 且携带有由所述多个客户端中的至少一个客户端发送的用于执行所述第三事件的第三数据包, 所述目标事件包括所述第三事件;

所述服务器记录所述第三数据包。

6. 根据权利要求 5 所述的方法, 其中, 在所述服务器定时向所述多个客户端广播所述第一请求之后, 所述方法还包括:

在所述多个客户端中的第一客户端失去连接, 且在记录所述第三数据包之后重新建立连接的情况下, 所述服务器向所述第一客户端发送所述第一数据包和所述第三数据包, 其中, 所述第一客户端在失去连接的过程中未接收到所述第一数据包和所述第三数据包, 所述多个客户端中的第一客户端失去连接包括: 所述多个客户端中的第一客户端所处的网络断线; 或者所述多个客户端中的第一客户端退出事件执行进程。

7. 根据权利要求 5 所述的方法, 其中, 在所述服务器定时向所述多个客户端广播所述第一请求之后, 所述方法还包括:

在所述多个客户端中的第一客户端失去连接, 且在记录所述第三数据包之后重新建立连接的情况下, 所述服务器接收所述客户端发送

的第四请求,其中,所述第四请求用于请求获取所述第一数据包或所述第三数据包,所述第一客户端在失去连接的过程中未接收到所述第一数据包和所述第三数据包;

所述服务器响应所述第四请求,并向所述第一客户端发送所述第一数据包或所述第三数据包。

8. 根据权利要求1所述的方法,其中,在所述服务器定时向所述多个客户端广播所述第一请求时,所述方法还包括:

所述服务器验证所述第一数据包是否合法;

在验证出所述第一数据包不合法的情况下,所述服务器禁止发送所述第一数据包的客户端执行所述目标事件,和/或所述服务器记录用于指示所述第一数据包不合法的信息。

9. 根据权利要求8所述的方法,其中,所述服务器验证所述第一数据包是否合法包括:

所述服务器向第一服务器广播所述第一数据包;

所述服务器通过所述第一服务器验证所述第一数据包是否合法。

10. 根据权利要求1所述的方法,其中,所述服务器定时向所述多个客户端广播所述第一请求包括:

所述服务器获取每个客户端的操作信息;

所述服务器将所述每个客户端的操作信息组合为数据帧;

所述服务器定时向所述多个客户端广播携带有所述数据帧的所述第一请求。

11. 根据权利要求1所述的方法,其中,所述服务器创建用于执行所述目标事件的所述目标场景包括:

所述服务器获取多个第二服务器的场景信息,其中,每个第二服务器的场景信息包括用于指示所述每个第二服务器允许创建的场景和已经创建的场景的信息;

所述服务器接收场景创建请求，并根据所述多个第二服务器的场景信息确定用于创建所述目标场景的目标服务器，其中，所述场景创建请求用于请求创建所述目标场景；

所述服务器向所述目标服务器发送创建指令，其中，所述创建指令用于指示所述目标服务器创建所述目标场景；

在所述目标服务器接收到所述创建指令之后，所述服务器通过所述目标服务器创建所述目标场景。

12. 根据权利要求 1 所述的方法，其中，在所述服务器确定处于所述目标场景中的所述多个客户端之后，所述方法还包括：

所述服务器对所述多个客户端的登录信息进行校验；

在对至少一个客户端的登录信息校验失败的情况下，所述服务器发送校验失败信息；

在对所述多个客户端的登录信息校验成功的情况下，所述服务器将所述多个客户端发送的数据设置为空帧。

13. 根据权利要求 1 所述的方法，其中，所述服务器定时向所述多个客户端广播所述第一请求包括：

所述服务器接收所述多个客户端发送的第五请求，其中，所述第五请求用于请求开始执行所述目标事件；

所述服务器响应所述第五请求，并允许定时向所述多个客户端广播所述第一请求。

14. 根据权利要求 13 所述的方法，其中，所述服务器允许定时向所述多个客户端广播所述第一请求包括：

在未全部接收到所述多个客户端发送的所述第五请求的情况下，所述服务器在第三预设时间允许定时向所述多个客户端广播所述第一请求。

15. 一种数据处理装置，应用于服务器，包括：

创建单元，设置为创建用于执行目标事件的目标场景，其中，所述目标事件为实时对战类事件；

确定单元，设置为确定处于所述目标场景中的多个客户端，其中，所述多个客户端用于执行所述目标事件；

广播单元，设置为定时向所述多个客户端广播第一请求，其中，所述第一请求用于请求所述多个客户端同步执行第一事件，且携带有由所述多个客户端中的至少一个客户端发送的用于执行所述第一事件的第一数据包，所述目标事件包括所述第一事件。

16. 一种存储介质，其特征在于，所述存储介质中存储有计算机程序，其中，所述计算机程序被设置为运行时执行所述权利要求 1 至 14 任一项中所述的方法。
17. 一种电子装置，包括存储器和处理器，其特征在于，所述存储器中存储有计算机程序，所述处理器被设置为通过所述计算机程序执行所述权利要求 1 至 14 任一项中所述的方法。

说明书附图

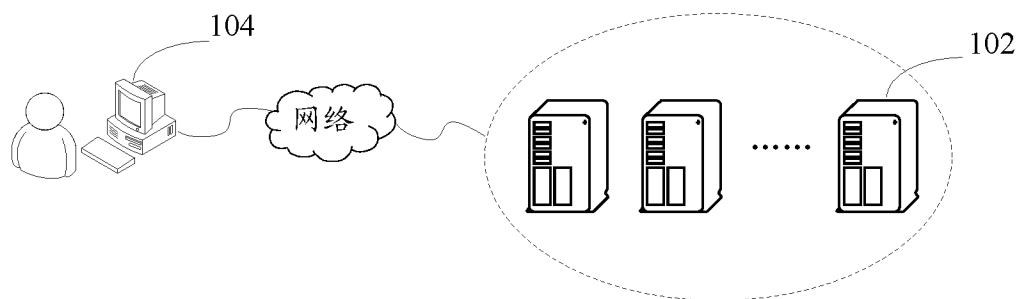


图 1

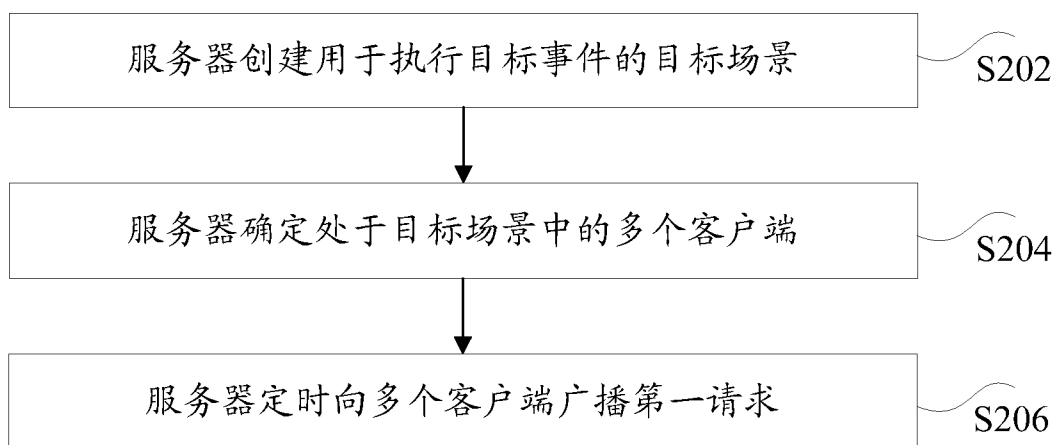


图 2

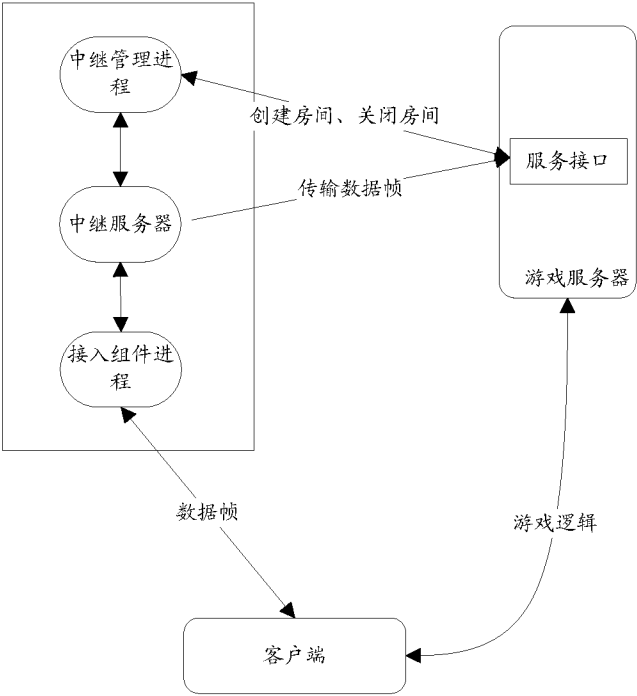


图 3

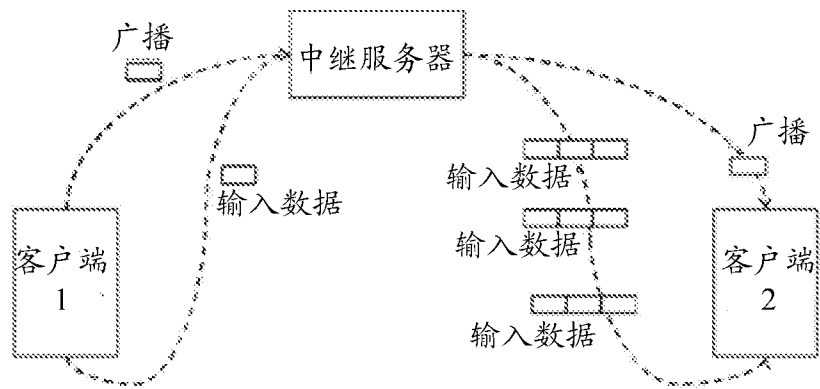


图 4

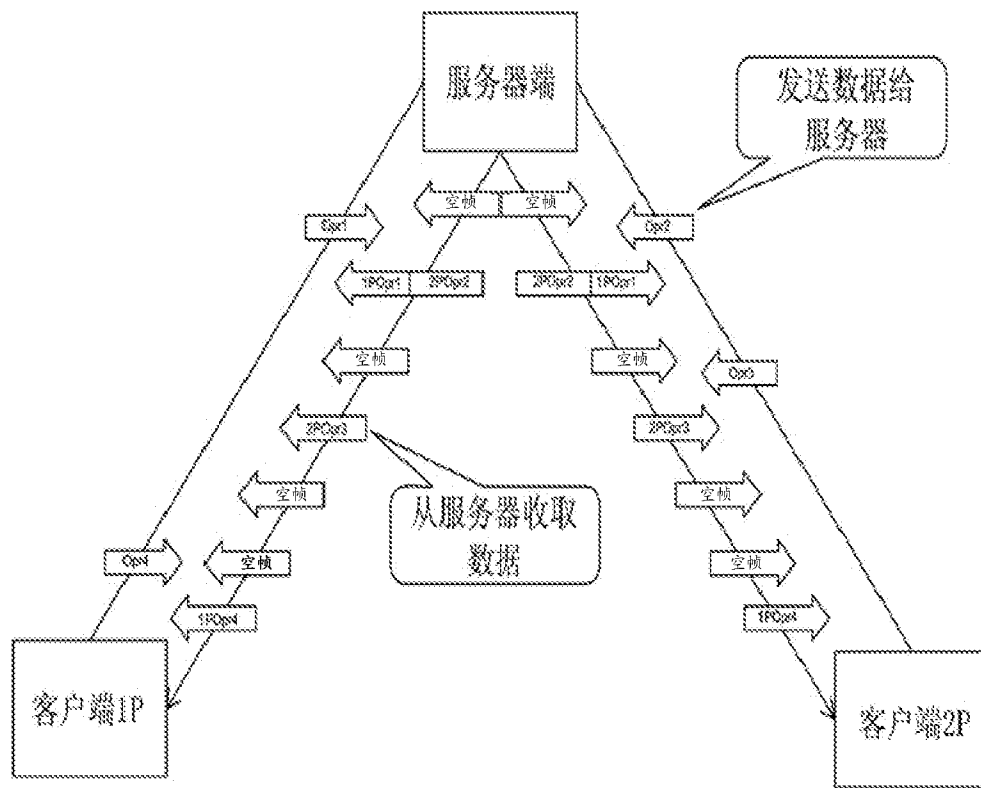


图 5

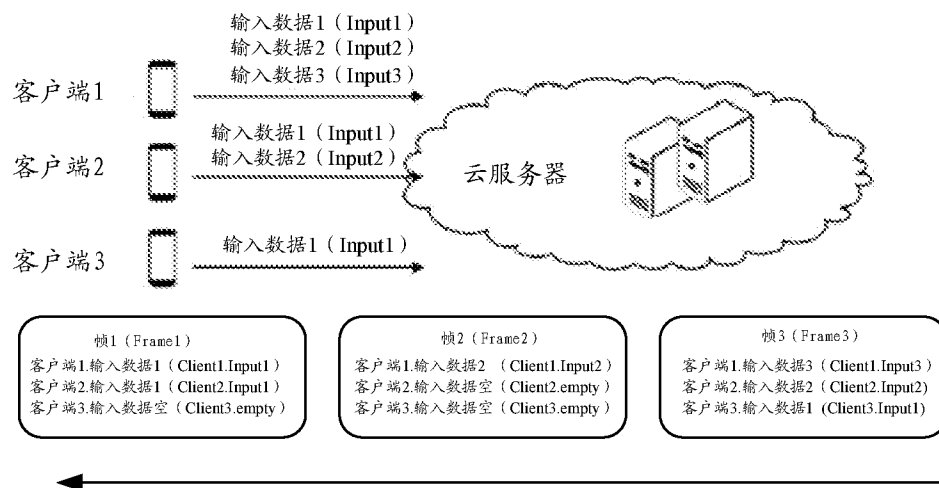


图 6

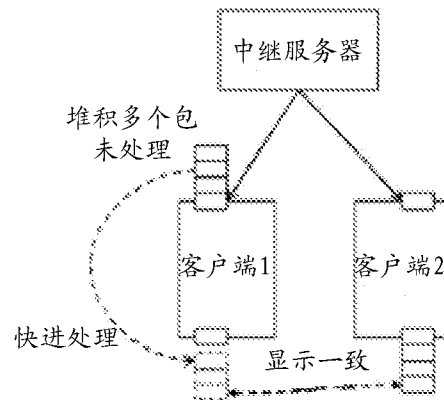


图 7

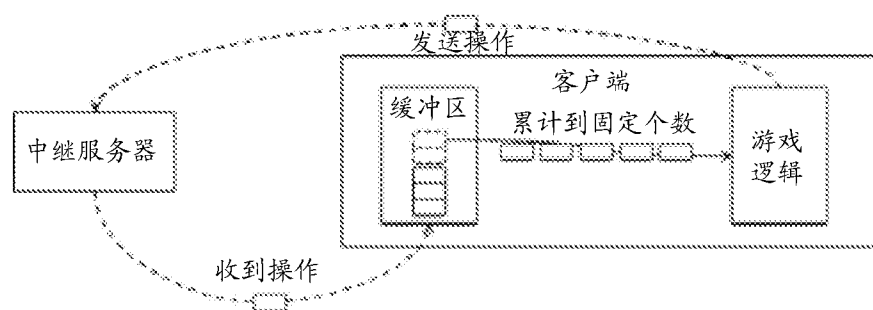


图 8

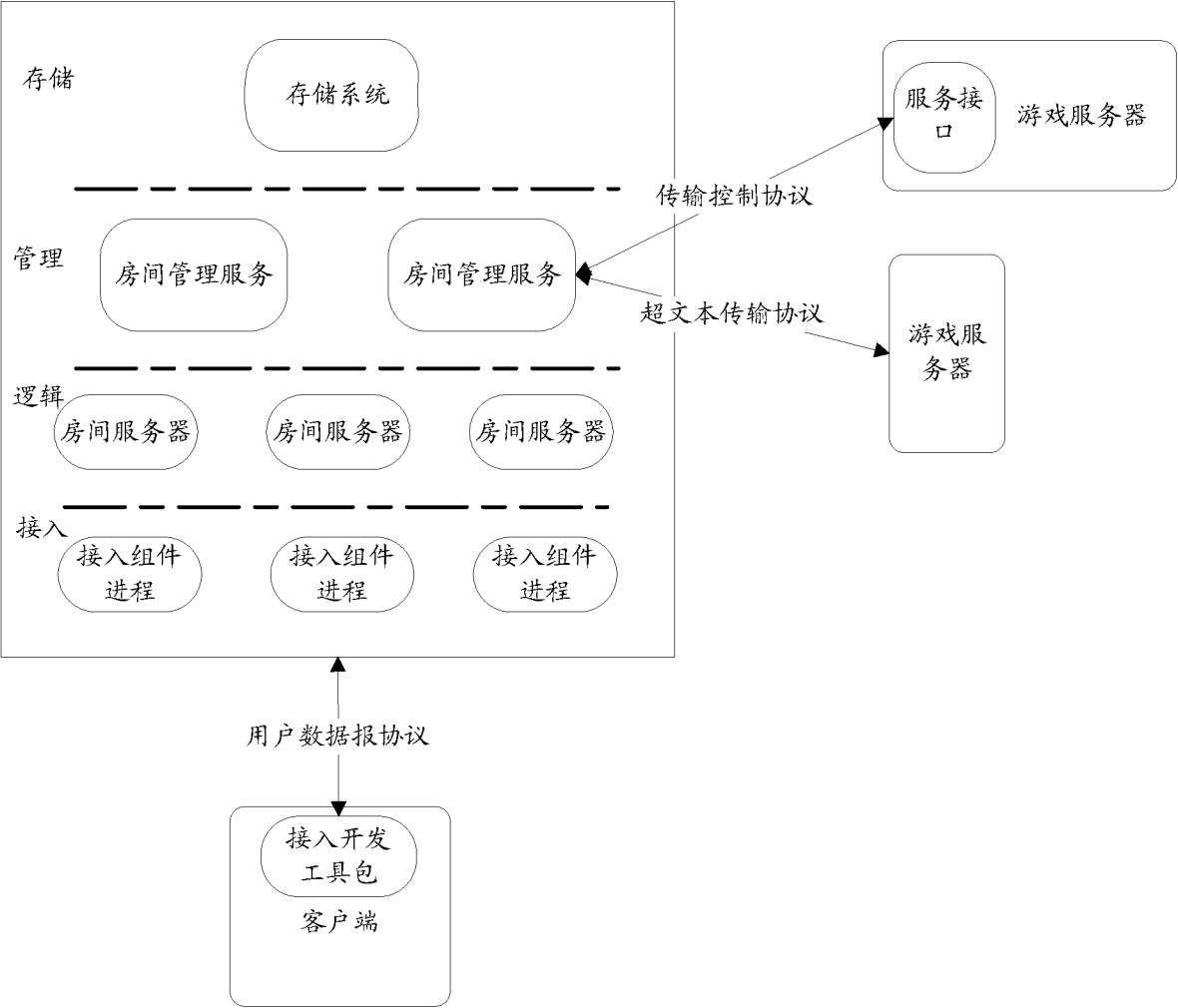


图 9

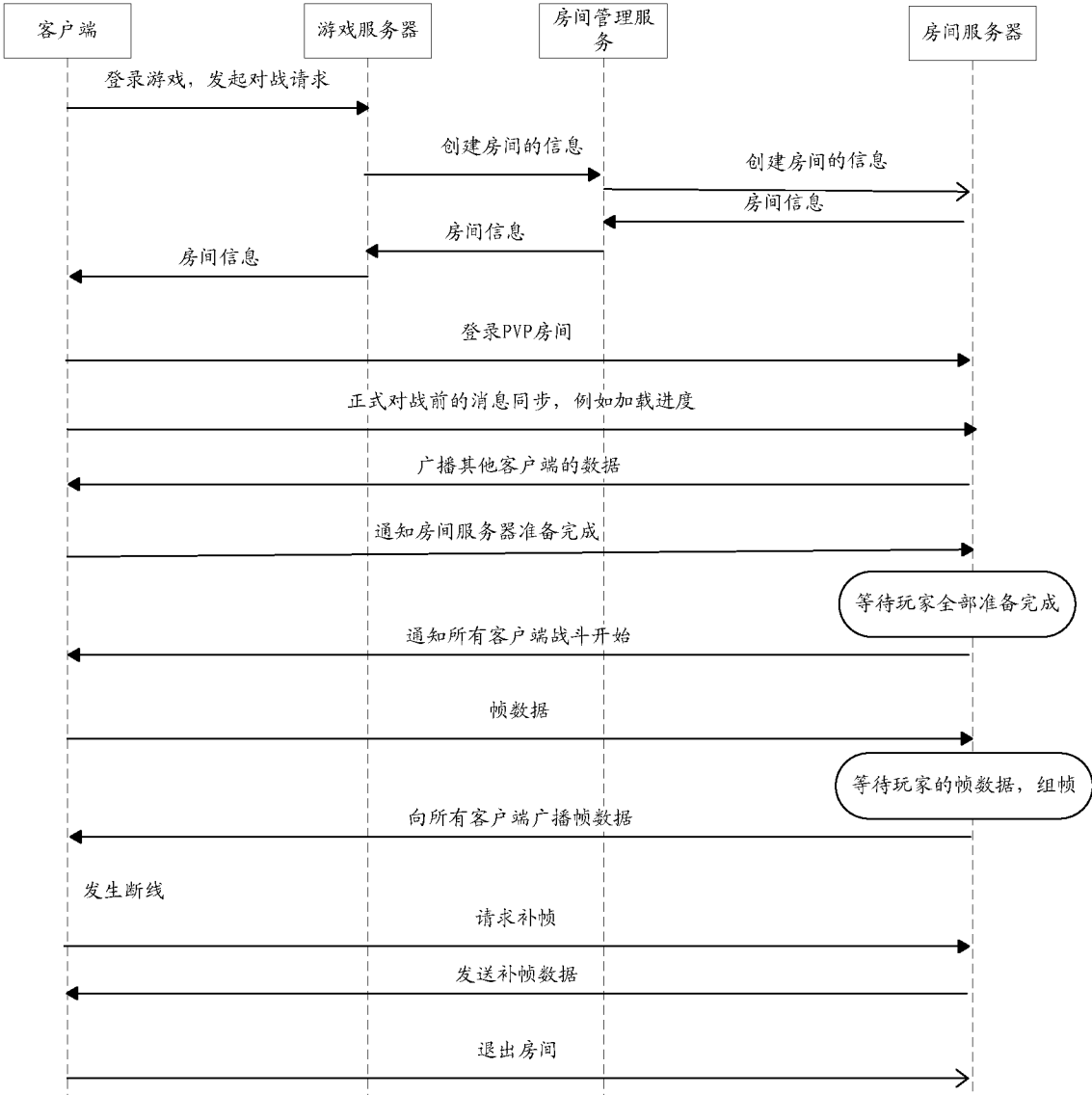


图 10

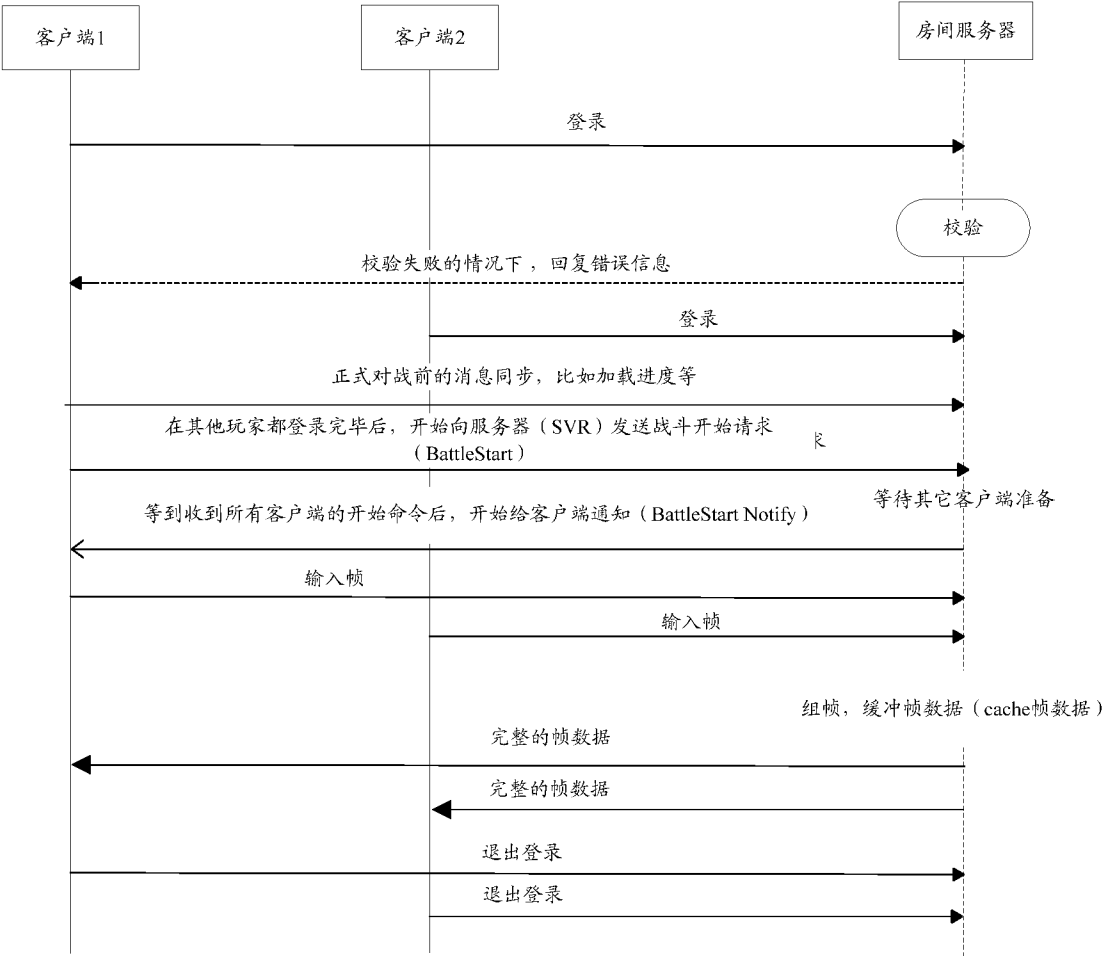


图 11

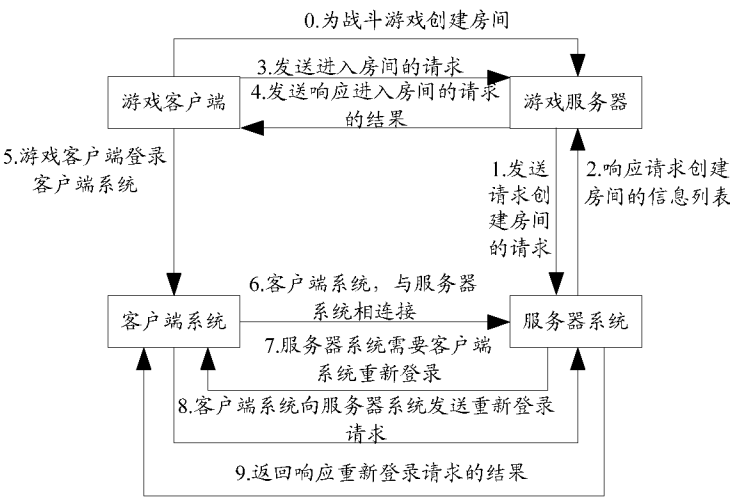


图 12

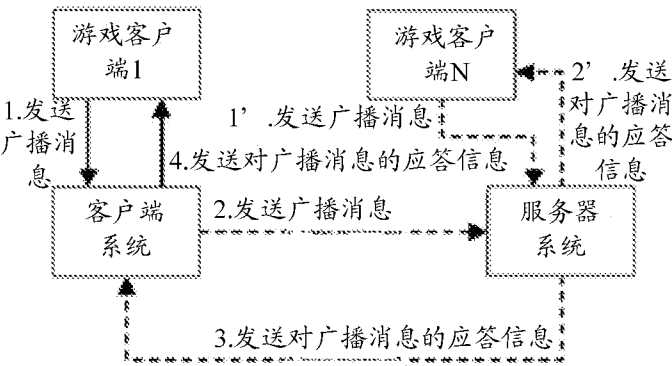


图 13

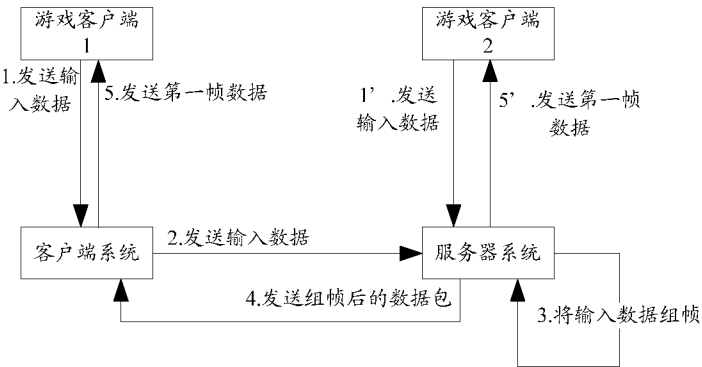


图 14

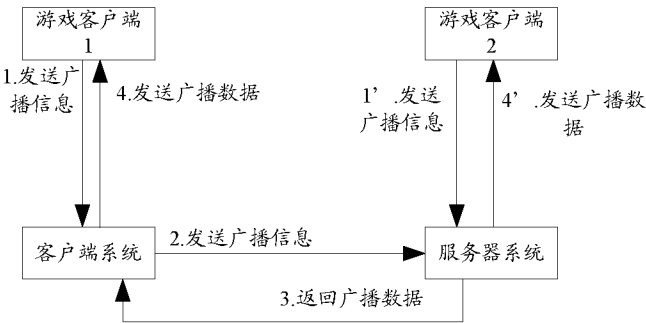


图 15

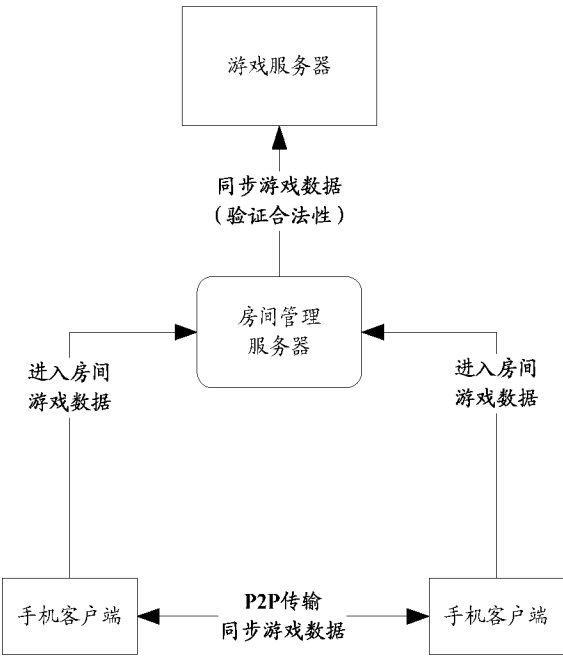


图 16

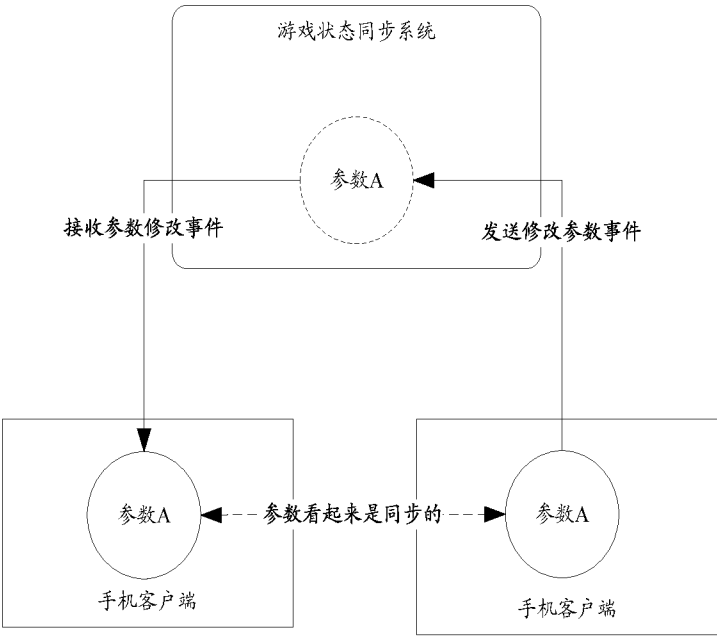


图 17

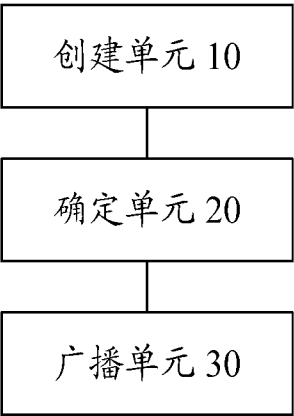


图 18

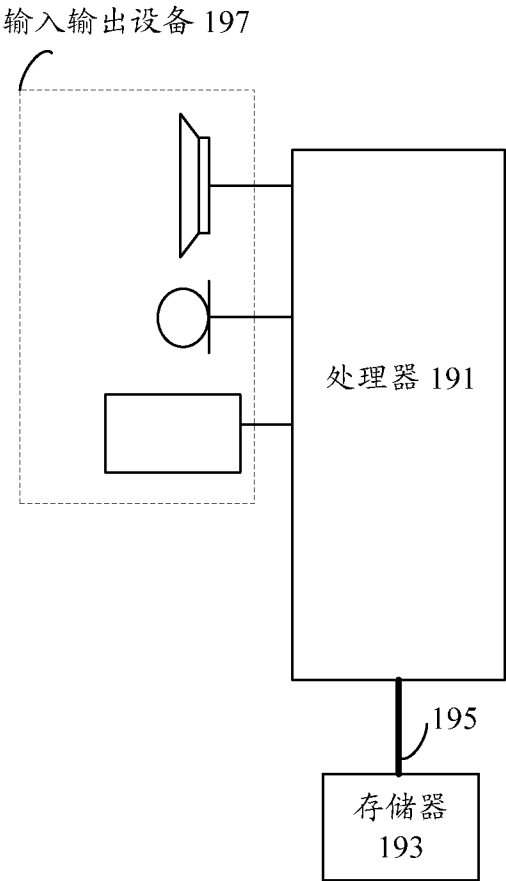


图 19

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2018/086327

A. CLASSIFICATION OF SUBJECT MATTER

H04L 29/08 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L, H04W

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT; WPI; EPODOC; BAIDU: 游戏, 帧同步, 定时, 间隔, 时间段, 频率, 帧, 周期, 广播, 空帧, 空指令, 场景, 合法, 校验, 冗余, 多发, 作弊, 外挂, lockstep, fsp, interval, timing, frequency, frame, synchroni+, broadcast, scene, event

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	CN 107404514 A (TENCENT TECHNOLOGY SHENZHEN CO., LTD.), 28 November 2017 (28.11.2017), entire document	1-17
X	WADEHAN, “帧同步游戏开发基础”, 腾讯 GAD, 08 September 2016 (08.09.2016), entire document, non-official translation (“Development Foundation of Frame Synchronization Game”, Tencent GAD)	1-7, 11-17
Y	WADEHAN, “帧同步游戏开发基础”, 腾讯 GAD, 08 September 2016 (08.09.2016), entire document, non-official translation (“Development Foundation of Frame Synchronization Game”, Tencent GAD)	8-10
Y	CN 103701918 A (BEIJING PIXEL SOFTWARE TECHNOLOGY JOINT-STOCK CO., LTD.), 02 April 2014 (02.04.2014), description, paragraphs 4-5	8-9
Y	CN 106375314 A (TENCENT TECHNOLOGY (SHENZHEN) CO., LTD.), 01 February 2017 (01.02.2017), description, paragraphs 7-9	10
A	KR 20140094423 A (NEXON CORP.), 30 July 2014 (30.07.2014), entire document	1-17

☒ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:	“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
“A” document defining the general state of the art which is not considered to be of particular relevance	“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
“E” earlier application or patent but published on or after the international filing date	“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	“&” document member of the same patent family
“O” document referring to an oral disclosure, use, exhibition or other means	
“P” document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 20 July 2018	Date of mailing of the international search report 08 August 2018
Name and mailing address of the ISA State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No. (86-10) 62019451	Authorized officer SUN, Jie Telephone No. 86-(10)-53961615

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2018/086327

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ENUMA, "Realization of Unity3D RTS 游戏中帧同步实现", 腾讯 GAD, 17 April 2017 (17.04.2017), entire document, non-official translation ("Frame Synchronization in Unity3D RTS Game", Tencent GAD)	1-17
A	游戏陀螺, "实时对战网络游戏基于帧同步的最佳实践", 搜狐, 08 December 2016 (08.12.2016), entire document, non-official translation (Youxituoluo, "Best Practice of Real-Time Battle Network Game Based on Frame Synchronization", Sohu)	1-17

International application No.
PCT/CN2018/086327

Form PCT/ISA/210 (patent family annex) (January 2015)

国际检索报告

国际申请号

PCT/CN2018/086327

A. 主题的分类 H04L 29/08 (2006.01) i 按照国际专利分类 (IPC) 或者同时按照国家分类和 IPC 两种分类																							
B. 检索领域 检索的最低限度文献 (标明分类系统和分类号) H04L; H04W 包含在检索领域中的除最低限度文献以外的检索文献 在国际检索时查阅的电子数据库 (数据库的名称, 和使用的检索词 (如使用)) CNPAT; WPI; EPODOC; BAIDU: 游戏, 帧同步, 定时, 间隔, 时间段, 频率, 帧, 周期, 广播, 空帧, 空指令, 场景, 合法, 校验, 冗余, 多发, 作弊, 外挂, lockstep, fsp, interval, timing, frequency, frame, synchroni+, broadcast, scene, event																							
C. 相关文件 <table border="1"> <thead> <tr> <th>类 型*</th> <th>引用文件, 必要时, 指明相关段落</th> <th>相关的权利要求</th> </tr> </thead> <tbody> <tr> <td>PX</td> <td>CN 107404514 A (讯科技深圳有限公) 2017年 11月 28日 (2017 - 11 - 28) 全文</td> <td>1-17</td> </tr> <tr> <td>X</td> <td>wadehan. "帧同步游戏开发基础" 腾讯GAD, 2016年 9月 8日 (2016 - 09 - 08), 全文</td> <td>1-7, 11-17</td> </tr> <tr> <td>Y</td> <td>wadehan. "帧同步游戏开发基础" 腾讯GAD, 2016年 9月 8日 (2016 - 09 - 08), 全文</td> <td>8-10</td> </tr> <tr> <td>Y</td> <td>CN 103701918 A (北京像素软件科技股份有限公司) 2014年 4月 2日 (2014 - 04 - 02) 说明书第4-5段</td> <td>8-9</td> </tr> <tr> <td>Y</td> <td>CN 106375314 A (腾讯科技深圳有限公司) 2017年 2月 1日 (2017 - 02 - 01) 说明书第7-9段</td> <td>10</td> </tr> <tr> <td>A</td> <td>KR 20140094423 A (NEXON CORP) 2014年 7月 30日 (2014 - 07 - 30) 全文</td> <td>1-17</td> </tr> </tbody> </table>			类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求	PX	CN 107404514 A (讯科技深圳有限公) 2017年 11月 28日 (2017 - 11 - 28) 全文	1-17	X	wadehan. "帧同步游戏开发基础" 腾讯GAD, 2016年 9月 8日 (2016 - 09 - 08), 全文	1-7, 11-17	Y	wadehan. "帧同步游戏开发基础" 腾讯GAD, 2016年 9月 8日 (2016 - 09 - 08), 全文	8-10	Y	CN 103701918 A (北京像素软件科技股份有限公司) 2014年 4月 2日 (2014 - 04 - 02) 说明书第4-5段	8-9	Y	CN 106375314 A (腾讯科技深圳有限公司) 2017年 2月 1日 (2017 - 02 - 01) 说明书第7-9段	10	A	KR 20140094423 A (NEXON CORP) 2014年 7月 30日 (2014 - 07 - 30) 全文	1-17
类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求																					
PX	CN 107404514 A (讯科技深圳有限公) 2017年 11月 28日 (2017 - 11 - 28) 全文	1-17																					
X	wadehan. "帧同步游戏开发基础" 腾讯GAD, 2016年 9月 8日 (2016 - 09 - 08), 全文	1-7, 11-17																					
Y	wadehan. "帧同步游戏开发基础" 腾讯GAD, 2016年 9月 8日 (2016 - 09 - 08), 全文	8-10																					
Y	CN 103701918 A (北京像素软件科技股份有限公司) 2014年 4月 2日 (2014 - 04 - 02) 说明书第4-5段	8-9																					
Y	CN 106375314 A (腾讯科技深圳有限公司) 2017年 2月 1日 (2017 - 02 - 01) 说明书第7-9段	10																					
A	KR 20140094423 A (NEXON CORP) 2014年 7月 30日 (2014 - 07 - 30) 全文	1-17																					
☒ 其余文件在C栏的续页中列出。 ☒ 见同族专利附件。																							
<table border="0"> <tr> <td style="vertical-align: top;"> * 引用文件的具体类型: "A" 认为不特别相关的表示了现有技术一般状态的文件 "E" 在国际申请日的当天或之后公布的在先申请或专利 "L" 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件 (如具体说明的) "O" 涉及口头公开、使用、展览或其他方式公开的文件 "P" 公布日先于国际申请日但迟于所要求的优先权日的文件 </td> <td style="vertical-align: top;"> "T" 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件 "X" 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性 "Y" 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性 "&" 同族专利的文件 </td> </tr> </table>			* 引用文件的具体类型: "A" 认为不特别相关的表示了现有技术一般状态的文件 "E" 在国际申请日的当天或之后公布的在先申请或专利 "L" 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件 (如具体说明的) "O" 涉及口头公开、使用、展览或其他方式公开的文件 "P" 公布日先于国际申请日但迟于所要求的优先权日的文件	"T" 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件 "X" 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性 "Y" 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性 "&" 同族专利的文件																			
* 引用文件的具体类型: "A" 认为不特别相关的表示了现有技术一般状态的文件 "E" 在国际申请日的当天或之后公布的在先申请或专利 "L" 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件 (如具体说明的) "O" 涉及口头公开、使用、展览或其他方式公开的文件 "P" 公布日先于国际申请日但迟于所要求的优先权日的文件	"T" 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件 "X" 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性 "Y" 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性 "&" 同族专利的文件																						
国际检索实际完成的日期		国际检索报告邮寄日期																					
2018年 7月 20日		2018年 8月 8日																					
ISA/CN的名称和邮寄地址		受权官员																					
中华人民共和国国家知识产权局 (ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10) 62019451		孙洁 电话号码 86-(10)-53961615																					

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
A	Enuma. "Unity3D RTS游戏中帧同步实现" 腾讯GAD, 2017年 4月 17日 (2017 - 04 - 17), 全文	1-17
A	游戏陀螺. "实时对战网络游戏 基于帧同步的最佳实践" 搜狐, 2016年 12月 8日 (2016 - 12 - 08), 全文	1-17

国际检索报告
关于同族专利的信息

国际申请号

PCT/CN2018/086327

检索报告引用的专利文件			公布日 (年/月/日)	同族专利	公布日 (年/月/日)
CN	107404514	A	2017年 11月 28日	无	
CN	103701918	A	2014年 4月 2日	无	
CN	106375314	A	2017年 2月 1日	无	
KR	20140094423	A	2014年 7月 30日	无	