计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(10):263−268 [doi: 10.15888/j.cnki.csa.006005]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

多人在线网络游戏服务器的设计与开发①

吴晶晶, 戴智超

(泉州师范学院 数学与计算机科学学院, 泉州 362000)
(福建省大数据管理新技术与知识工程重点实验室, 泉州 362000)
(智能计算与信息处理福建省高校重点实验室, 泉州 362000)

摘　要: 服务器的效率是网络游戏能否提供高质量的网络服务的重要因素之一. 本文针对这一问题, 提出了模板化回调策略改良现有的网络游戏的会话信息交互. 文中基于 VisualStudio 与 C++, 重点剖析服务器架设过程中的关键技术和策略, 设计开发了经典网络人物扮演游戏服务器, 分别由“数据库”、“账号服务器”、“网络服务”和“副本”组成. 实验结果表明, 采用模板化回调技术, 有效降低服务器信息交互的延迟. 实践证明, 本文所讨论的关键技术可应用于场景漫游、游戏开发等多个领域, 具有一定的实际应用价值.

关键词: 服务器效率; 多人在线网络游戏; 模板化回调设计; 高质量的网络服务

引用格式: 吴晶晶,戴智超.多人在线网络游戏服务器的设计与开发.计算机系统应用,2017,26(10):263–268. http://www.c-s-a.org.cn/1003-3254/6005.html

Development of Multiplaying Online Game Server

WU Jing-Jing, DAI Zhi-Chao

(College of Mathematics and Computer Science, Quanzhou Normal University, Quanzhou 362000, China)
(Fujian Provincial Key Laboratory of Data Intensive Computing, Quanzhou 362000, China)
(Key Laboratory of Intelligent Computing and Information Processing, Fujian Province University, Quanzhou 362000, China)

Abstract: Nowadays, the run-time performance of server becomes a crucial factor for online games. To solve the problem, this paper proposes a new method, which uses the callback strategy based on templates, to improve the run-time performance of game-server. Based on Visual Studio and C++, the key technology and strategy during the erection of server are researched. In the article, the server is divided into 4 parts: database, account server, network and map. The experimental result shows that the delay in server session reduces obviously by using the callback strategy based on templates. Through this experiment, it is proved that the key technology proposed can be used in VR environments, game development and many other fields and has practical value.

Key words: server efficiency; multiplaying-online-game; callback designing based on templates; high-quality network

随着 Android 平台游戏、iPhone 平台游戏以及 Web 网页游戏迅猛发展, 3D 游戏成为当前游戏的发展热点. MMORPG 游戏即多人在线角色扮演类游戏, 是现在最为流行的经典网络游戏形式. 随着高性能智能手机的普及, 市场上出现了如“王者荣耀”、“阴阳师”等这类极为优秀的次时代 MMORPG. 但这依然不足以满足手机游戏玩家的需求, 玩家们开始追求手机平台游戏与 PC 平台游戏的联动.

固然手机性能已经可与 PC 比肩, 但手机网络问题依然高悬. 现今普及的 4G 网络, 其速度已经接近

① 基金项目: 福建省科技厅自然科学基金面上项目(2017J01776); 福建省省属高校科研专项项目(JK2015037); 泉州师范学院青年博士预研基金项目(2015QBKJ02); 泉州师范学院博士科研启动项目 (G17003)
收稿时间: 2017-01-24; 采用时间: 2017-02-20

PC 网络水平, 但网络稳定性依然远远达不到 PC 网络水平, 而且高额的流量费用也成了手机游戏难以与 PC 联动的巨大障碍. 为实现手机网络游戏与 PC 网络游戏的联动, 必须搭建高网络稳定性、低流量消耗、低延迟的高性能多平台通用服务器. 通过这样的服务器来实现高质量网络服务.

高质量的网络服务受限于三大因素: 首先是网络运营商的网络质量, 由第三方提供, 如电信或是联通等运营商; 其次是网络协议层和路由算法的优化程度[1,2], 游戏开发者在这方面则作为用户, 故本文不对其进行讨论; 最后是服务器的效率.

关于如何提高网络服务质量, 国内的研究大多数针对网络协议层, 但国内也有针对提高服务器效率的优秀策略. 如王瑞彪等人基于 IOCP 机制完成服务器通信层的实现[3]. 该方案实现了多线程信息投递并实时监听, 降低了信息重复发送、排序错误概率, 并为用户留下完整的操作接口. 林泊等人基于 J2EE 集成了应用服务器框架[4], 实现在不对读取速度造成影响的情况下降低了维护和操作难度.

本文针对如何提高服务器效率, 提出“模板化回调”策略改良现有的网络游戏的会话信息交互. 实践证明, 本文所讨论的关键技术可应用于场景漫游、游戏开发等多个领域, 具有一定的实际应用价值.

1 服务器设计

1.1 服务器设计

本文中的服务器对应于文中由基于 Unity3D 开发的 MMORPG 游戏客户端. 玩家在游戏客户端注册并登录个人账号, 服务器则将玩家录入的信息新建为 SQL 表并存入数据库[5]中. 服务器核心处理客户端发送来的所有信息请求. 以“组队”请求为例, 由一位“队长”发起请求, 在经过服务器处理后, 服务器再将信息同步到所有账号被请求的“队员”. 通过服务器实现的“组队”如图 1 所示. 基于这种会话形式, 本文实现了多个客户端之间依赖于彼此的账号, 在服务器中进行数据同步.

在服务器的设计上, 传统的高同步帧率普通 RPG 游戏为 30 帧, 高更新率的动作游戏为 60 帧. 相较于传统同步率, 本文则采取 10 帧的同步率, 10 帧的设定足够实现事件流畅同步, 同时避免了过高同步率导致的流量的不必要流失.

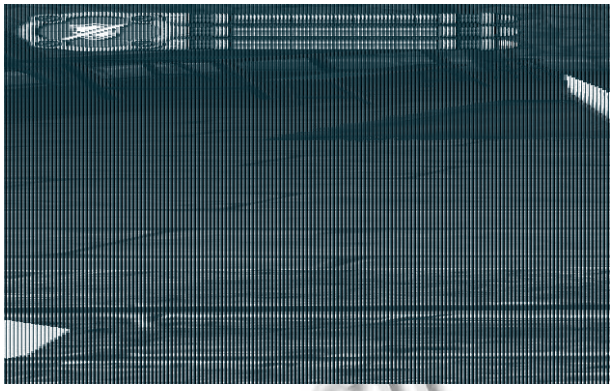

图 1 服务器“组队系统”的设计

1.2 设计策略

为了更好地符合用户的需求, 本文对于服务器的设计提出以下策略:

① 服务器应标识并处理客户端的静态全局型请求, 例如账号登录、登出请求.

② 服务器应实时处理客户端对于账号数据的静态全局型请求, 例如像申请账号、修改账号密码、获得新的道具等需要修改数据库静态表的请求.

③ 服务器应标识并处理客户端的动态非全局型请求, 例如人物受伤后生命值降低、发起的攻击被躲开等. 服务器标识与处理如图 2 所示.

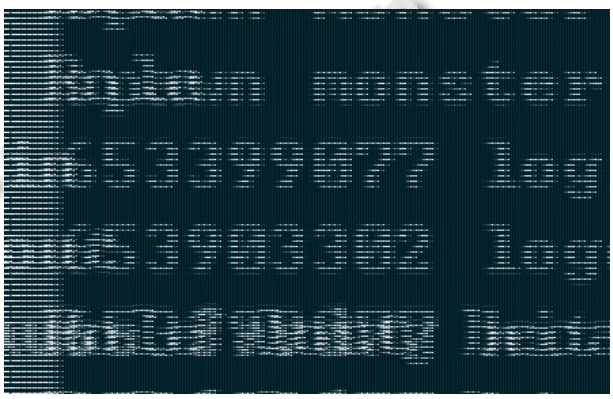

图 2 服务器标识与处理

1.3 服务器结构

本文服务器系统结构如图 3 所示, 服务器基于自主研发的 Core 核心架构, 通过核心连接“数据库”、“账号服务器”、“网络服务”、“副本”四个子程序构成. 服务器这样的设计模式, 使维护、调试难度降低, 并且由于服务器运行基于 Core 核心架构, 提高了运行效率和编译耦合度.

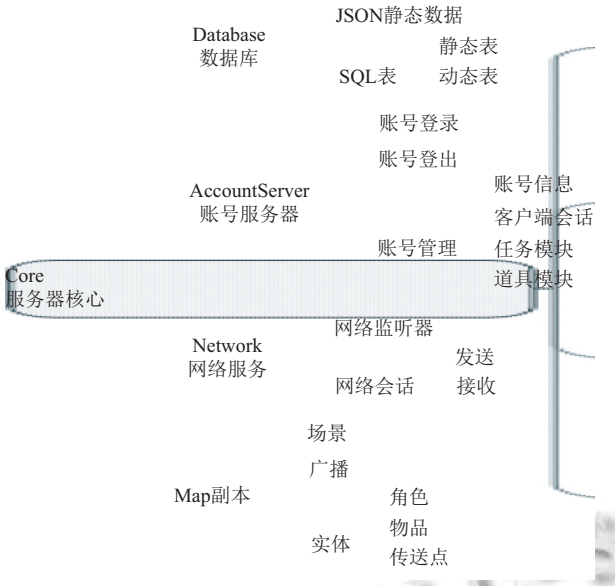

图 3 服务器核心结构

2 关键技术

2.1 账号服务器运行逻辑

账号服务器是服务器中剥离出来的重要子程序之一. 账号服务器只处理客户端发出的账号申请, 例如“登录”、“登出”、“注册”等行为. 在服务器核心中, 由“网络服务”接收账号请求, 然后账号服务器管理这些账号请求, 设定对应账号的访问状态. 然后将请求对应的处理方式经由“网络服务”再次发向对应的账号. 账号服务器运行逻辑如图 4 所示, 以下是对账号服务器运行逻辑的分析:

(1) 预先声明定义好 Account 类型, 构造两个 Account 对象的数组: “登录账号数组”和“登出账号数组”.

(2) Run 每一帧监听客户端请求, 若存在“登录请求”就在“登录账号数组”中创建一个新的 Account 对象, 并把“登录请求”传入的 Account 信息写入新建的 Account 对象中达到复制构造的特性.

(3) Run 每一帧监听客户端请求, 若存在“登出请求”就在“登出账号数组”中保存“登出玩家”, 方法同 (2).

(4) 登出处理是在 Run 函数一开始就执行的. 在每次下一帧的开始, 遍历“登出账号数组”, 然后将“登录账号数组”中所有相同的 Account 删除, 再清除“登出账号数组”.

2.2 服务器信息交互

本文采用常用的会话通信方式, 完成服务器与客户端之间的数据交互. 对于每次客户端行为, 例如来自客户端的一次登陆请求, 客户端都会向服务器发起一次会话. 服务器接收该会话内容, 并通过 RTTI[6,7]技术的设计, 将相应的会话内容进行识别并加以打包. 通过网络监听器监听服务器状态, 网络监听器的设计基于访问者设计模式[8]. 通过这一系列过程, 完成服务器和客户端间的信息交互, 信息交互的实现思路如图 5 所示.

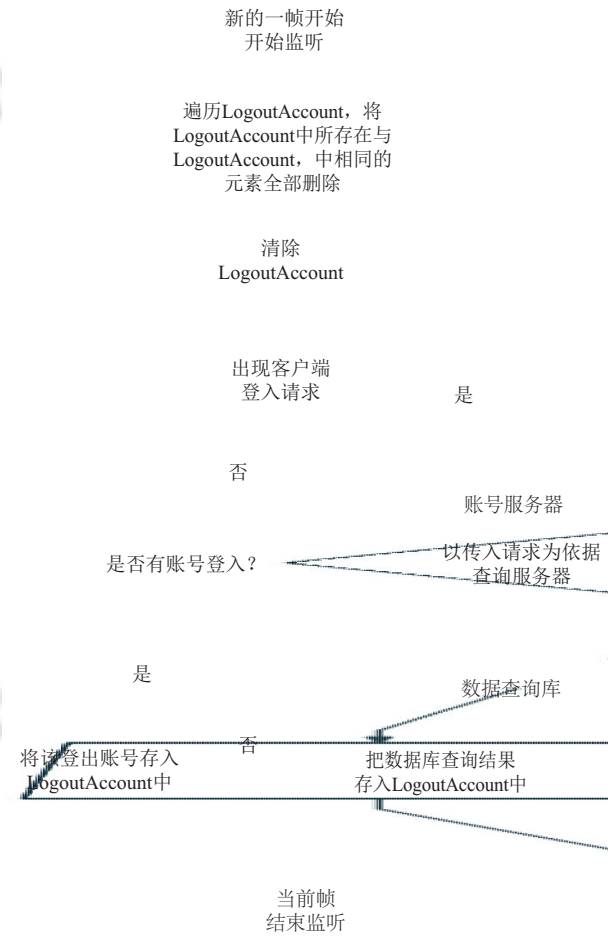

图 4 账号服务器运行逻辑

① 若服务器处于可访问状态时, 服务器可以不断接受信息, 并不断处理信息.

② 若服务器处于不可访问状态时, 则服务器中断接收新信息, 并处理完抛出当前未接受完的数据包.

③ 当信息接收失败时, 服务器直接放弃接收该数据包. 在服务器将该数据包抛出后, 服务器进入下一帧的数据读取.

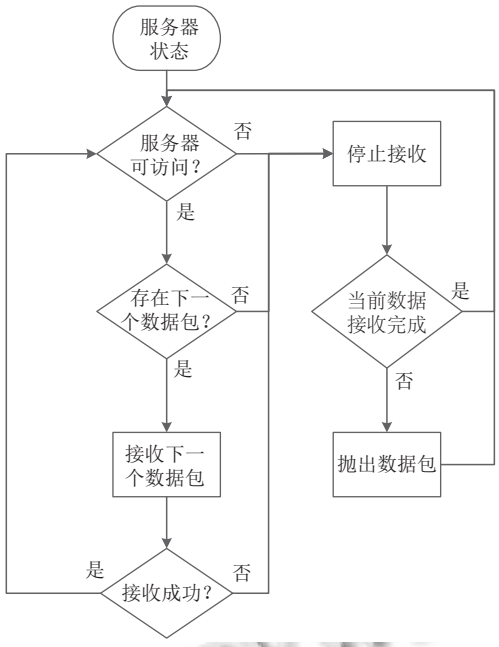

图 5 服务器信息交互

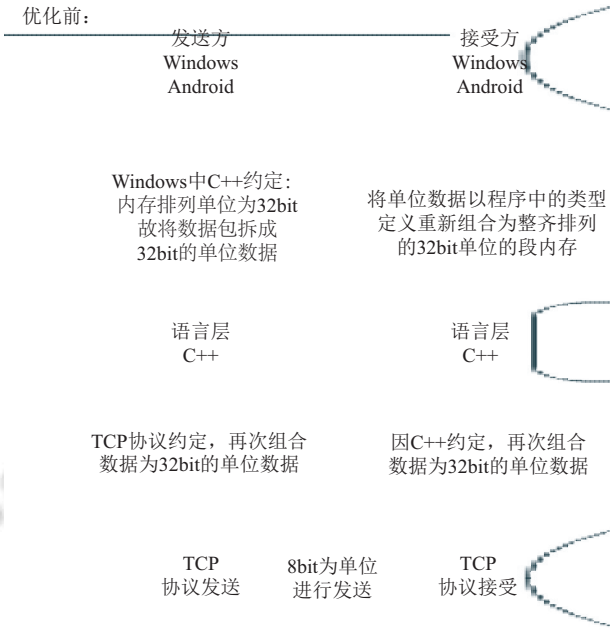

图 6 传统的网络处理

2.3 降低网络延迟的策略

对于 MMORPG 游戏而言, 网络延迟是一个长久以来的难题. 为了提供玩家良好的游戏体验, 服务器的设计不仅要实现“时刻等待并处理”玩家发送的信息, 同时还要保证网络延迟不会对玩家的游戏体验造成可视的影响. 但网络状况往往遭受例如路由协议、网络运营商等因素的干扰, 因而服务器的设计中, 应有应对网络延迟的策略[9].

本次服务器通信协议采用 TCP 协议, 但 TCP 协议与服务器的平台 Windows 的数据标准不同. TCP 协议只允许数据以 8bits 为单位进行发送与接收, 而 Windows 中默认 C++以 32bits 为单位将数据整齐排列到内存中. 此时将数据发送出, 如图 6 所示, 数据在 TCP 协议层还需经过拆解, 并读取接收, 这一过程造成无意义的消耗.

同时, 不同于客户端中的数据接收, 在服务器中, 数据的接收是短暂性的动作, 生命周期较短. 因而数据流始终都是局部变量. 因此, 服务器的数据接收器应该是全局的变量, 并且始终处于可接收新数据的状态. 再者, 因为客户端的数据发送都是成段数据包一起发送, 一旦出现了网络延而中断传输, 数据包丢包将会引起很明显的游戏卡顿现象. 因此若不进行特别的处理, 一次十几毫秒的网络延迟就会带来数个或是数十个数据包丢失.

故而本文采取了提前拆解所有数据包, 以字节为单位进行多次传输. 这样每次就算丢包, 也只丢失几个字节, 而不是一整个包. 并且, 这个过程中, 加快了数据交换的进程. 因为数据在平台中以被拆解为 8bits 单位, 因而在进入 TCP 协议层时, 可以略过繁杂的从 32bits 到 8bits 单位的拆解过程. 通过提前割裂数据来对应高时延网络, 进而提高了网络容断性[10]. 数据拆解传输的基本思路如图 7.

2.4 提高服务器效率的特殊策略——模板化信息回调处理实现低延迟网络

对应上文存在的问题, 需要设计一个合理的算法来完成信息的处理. 前文中已提到服务器接收是一个全局的行为, 接收一次数据是一个极为短暂的行为, 因为每次接收的数据仅仅只有 8bits. 如果每次都声明一个新的接收器对象, 然后把数据包赋值给接收器, 这期间的构造和析构成本就太大了.

因而本文提出以下解决方法: 不予接收器赋值, 而是让接收器内只存放一个函数指针. 每次接收数据时, 数据的缓冲将会在函数指针指向的函数中完成. 然后对该函数再封装一层, 并在封装层中进行函数的回调. 这样就可以避免了不断对一个远大于 8bits 的接收器进行构造和析构. 最后, 再将这一过程进行模板化, 使接收器可以接受其他各种类型的数据.

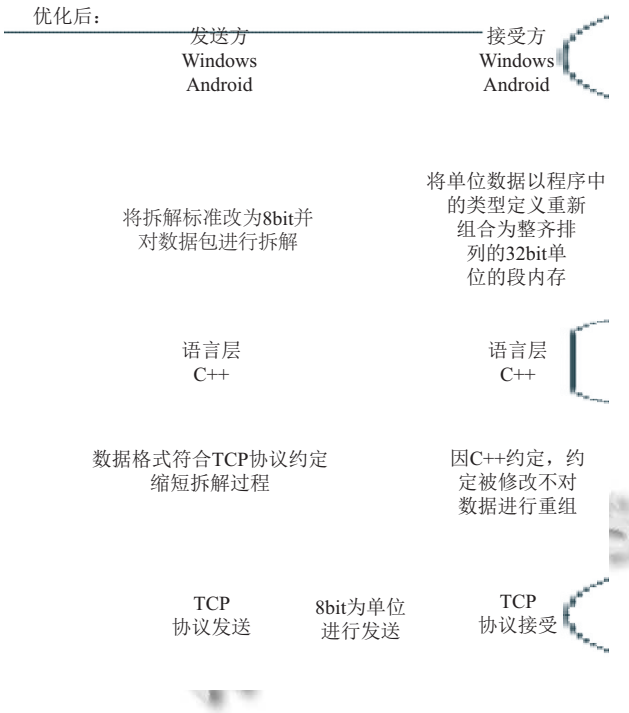

图 7 对网络延迟的特殊处理

出于提高效率的目的, 本文采用了回调而不使用递归. 如果使用递归, 可以有效的利用运行栈[11], 通常认为这是高效的做法, 其实不尽然. 因为这期间的数据接收, 接收次数不可知. 从理论上而言, 只要电源不断、数据包足够大, 服务器将为接受一个数据包, 而不断地接受它的每一个 8bits. 这可能导致二叉树的深度永远无法测量[12], 这个问题很致命, 意味着需求的内存将永远无法被估测. 据此, 本文使用回调代替递归, 回调信息处理关键代码如下:

(1) 定义模板用于导入数据包, 定义与回调相关的函数指针.

(2) 重载所需的运算符, 满足模板对象参与计算时的需求.

(3) 通过模板中传入的对象 Holder, 找到 Holder 中的函数指针 Callback.

(4) 调用异步接收函数对 Callback 进行调用, 然后进入 Callback 的回调接收数据.

```
template<typename Holder>
struct AcceptHandler
{
    typedef void (Holder:: *CallBack)();
    Holder* holder;
    CallBack callback;
    void operator()()
    {
        (holder→*callback)();
    }
};
void Network::Impl::AcceptNext()
{
    newSession = new SessionImpl(ioservice);
    acceptor.async_accept(newSession→socket,
        acceptHandler );
}
```

3 网络游戏服务器实现与效能优化分析

本文采用上述的设计策略和关键技术, 实现了基于 VisualStudio 与 C++的 MMORPG 游戏服务器的开发. 服务器运行状态如图 8 所示, 当玩家登录、登出时, 服务器对其进行标记, 方便服务器管理员实时观测当前服务器运行状态. 以攻击状况为例, 当攻击成功时则产生攻击特效如图 9 所示, 若玩家因为面朝方向错误时进行攻击, 则服务器如图 8 标记该玩家攻击失败的信息.

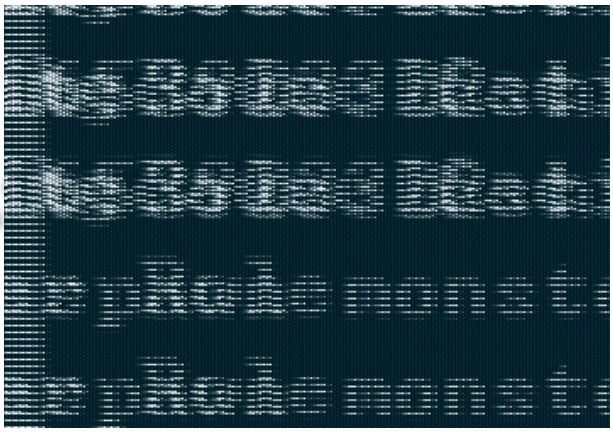

图 8 网络游戏服务器实现

同时, 本设计引用第三方数据库 SQLite3, 建立账号对应的数据库. 服务器则以数据库中 SQL 表为依据, 记录账号对应的角色属性、道具、任务状况等个人信息, 并在服务器运行期, 以同步帧 10 为间隔不断更新数据表内容. 服务器管理员可以通过第三方软件 SQLiteStudio 直接对数据库内容进行操作, 数据库操作如图 10 所示.

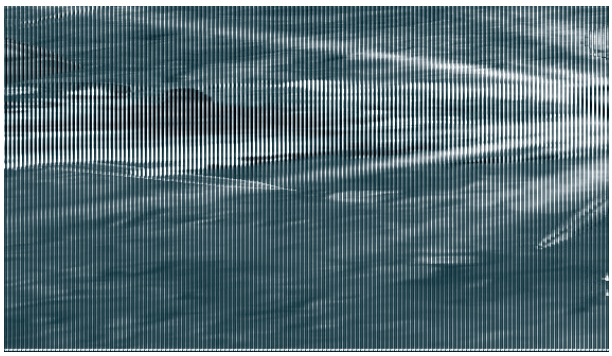

图 9 玩家正确攻击时的特效

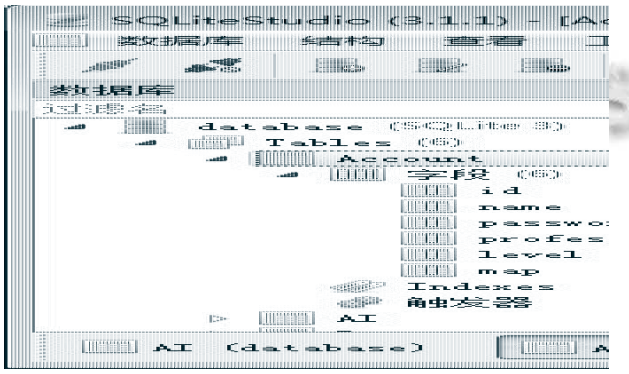

图 10 通过 SQLiteStudio 管理数据库

本设计采取了模板化回调设计策略, 对传统网络游戏服务器进行优化, 服务器优化前后的效能比较如表 1 所示. 从表中可以得出, 使用该优化策略有效的降低了客户端运行期的网络延迟. 同时该策略对于服务器运行期的数据丢包率也具有一定的优化作用. 在降低了延迟和丢包率的基础上, 该策略对于服务器运行时的数据交互速度并未产生影响.

4 结语

本文基于 VisualStudio 与 C++, 研究经典网络人物扮演游戏服务器的设计与开发. 主要研究了通过提高服务器程序工作效率来提升网络服务质量, 从而使玩家获得更好的游戏体验. 在服务器基本架构完成的基础上, 通过“模板化信息回调处理”策略使网络延迟率下降, 提高了网络服务质量.

表 1 服务器优化前后效能

		优化前	优化后
客户端FPS(帧/s)	最高FPS	>60.0	>60.0
	平均FPS	47.9	53.7
平均丢包率(%)		19.3	8.2
客户端延迟(ms)	最高延迟	231	176
	平均延迟	120	44
数据交互速度(Byte/s)	最高速度	>1000	>1000
	平均速度	93	89

参考文献

1 崔勇, 吴建平, 徐恪, 等. 互联网络服务质量路由算法研究综述. 软件学报, 2002, 13(11): 2065–2075. [doi: 10.13328/j.cnki.jos.2002.11.002]

2 罗赟骞, 夏靖波, 智英建, 等. 采用策略的 IP 网络运行质量评估方法实现. 小型微型计算机系统, 2011, 32(1): 112–116.

3 王瑞彪, 李凤岐, 施玉勋, 等. 基于 IOCP 机制的网络游戏服务器通信层的实现. 计算机工程与应用, 2009, 45(7): 75–78, 81.

4 林泊, 周明辉, 刘天成, 等. 一个 J2EE 应用服务器的 Web 容器集成框架. 软件学报, 2006, 17(5): 1195–1203.

5 林培杰, 朱安南, 程树英. Android 数据库 SQLite 性能优化. 计算机系统应用, 2014, 23(4): 193–196.

6 王玉亭, 孙剑. 应用程序框架中对象动态创建和 RTTI 机制的实现. 计算机与现代化, 2007, (8): 50–52.

7 何洪辉, 刘骥宇. RTTI 机制浅析. 计算机与现代化, 2010, (8): 120–123.

8 冯玉琳, 黄涛, 李京. 面向对象的软件构造. 软件学报, 1996, 7(3): 129–136. [doi: 10.13328/j.cnki.jos.1996.03.001]

9 邱航, 何明耘, 陈雷霆. 网络游戏引擎中同步技术研究综述. 计算机应用研究, 2007, 24(1): 14–17.

10 张龙, 周贤伟, 王建萍, 等. 容迟与容断网络中的路由协议. 软件学报, 2010, 21(10): 2554–2572.

11 谷晓铭, 霍玮, 桂剑, 等. 一种检测运行栈与静态数据区重叠的新方法. 计算机工程与应用, 2006, 42(20): 86–88, 112. [doi: 10.3321/j.issn:1002-8331.2006.20.027]

12 王敏, 赵晓雷. 基于遍历搜索二叉树中最长路径的算法研究. 现代电子技术, 2010, 33(8): 54–55, 58.