《计算机网络协议开发》实验报告

第4次实验

文字对战游戏编程实验

姓名：陈越琦

学号：121160005

计算机系13级

邮箱：Yueqichen.0x0@gmail.com

时间: 04/05/2017

实验目的

熟悉网络应用层协议的设计与实现,掌握服务器套接字编程。

实验内容

游戏功能

根据实验讲义，我们实现了以下的必需功能：

1. 用户注册，登陆，登出功能
2. 好友查看功能
3. 好友状态变动推送
4. 对战邀请功能
5. 对战功能

此外，我还实现了以下的额外功能

1. 积分榜查看功能
2. 实时聊天功能。

客户端设计

客户端包含了两个线程，分别是主线程和监听线程。主线程监视用户输入，并对用户输入做出响应，包括给出操作提示以及向服务器发送报文等功能。监听线程监听来自服务器发送的报文，根据报文信息做出响应，包括改变客户端状态，向用户打印好友状态变动信息，以及打印对战中每一回合的对战结果等功能。

客户端中的主线程和监听线程通过**信号量**机制实现同步。更具体的，主线程在接受用户的每一次输入之后都会通过抢占一个初始化为零的全局信号量阻塞自己，在等待监听线程获得服务器端报文，根据报文应用层内容修改客户端状态或者打印必要信息之后释放信号量之后，继续执行。

事实上，客户端中的主线程和监听线程在绝大部分时间内，都是交叉轮流执行，即发包-收包。从并行角度说，在这种情况下采用多线程的方式实现客户端是没有必要的，甚至是低效的（考虑到线程切换和系统调用需要时间）。但是，小组成员在讨论之后，仍然认为这样的实现方式是必要的。因为好友状态变动推送与客户端所处状态以及用户输入并没有直接关系，换句话说，服务器向客户端发送好友状态变动推送不是因为客户端发送了查询报文，而是一个“突发”的情况。基于这方面的考虑，为了能够将好友状态变动的消息及时得告知用户，需要使用另外一个线程，也就是监听线程实时处理服务器发送的报文。这样的设计尽管从某种程度上说，使得客户端的实现更为复杂，但是是值得的。

服务器设计

服务器包含了四个线程，分别是主线程，广播线程，客户线程和对战线程。此外，服务器还在内存中记录了所有在线用户的相关信息，以及每个对战房间内的对战信息。

广播线程在服务器启动之初建立。用于向所有的在线非对战用户广播用户登陆登出或者加入对战信息。每当这些事件发生的时候，客户线程或者对战线程都会通过一个与广播线程相关的**信号量**“解锁”广播线程，并将用户的状态变化写入一个与广播线程相关的队列。被“解锁”广播线程从队列头部获得用户状态变化信息，广播出去。这是一个典型的生产者-消费者模型，但是正如读者所见，我们只是用了一个信号量对队列加锁解锁，这样势必导致互斥访问问题。我们在实验中尚存问题小节中进一步讨论这个问题。

每当有客户端连接到服务器都会产生一个客户线程。客户线程监听并处理客户端发送的报文，每当客户端有用户登陆，客户线程都会将当前用户信息记录下来。当用户从客户端退出（无论是正常退出还是突然掉线），客户线程会将用户信息持久化存储到磁盘中，以便下次使用。除此之外，客户线程还会建立对战线程，并将接收到的所有与对战相关的报文存储到对战线程相应的房间中，以供对战线程使用。

对战线程由客户线程建立，在每一个回合的对战中，对战线程处理房间内玩家的客户线程提供的对战报文，判断胜负并通知各个玩家。此外，对战线程还负责处理对战邀请的报文。正如上文所言，对战过程中，客户线程不会暂停执行，而是会继续接受客户端报文，并将报文提供给对战线程处理。

客户端与服务器协同

从简化客户端设计原则角度出发，我们将很多功能集成到服务器中处理。比如，每次对战的输赢判断和积分榜排序都是在服务器中完成。客户端只是将服务器的判断结果或者是积分榜信息显示给用户。

这样的设计方法使得客户端异常简单，同时也缩小了数据传输过程中遭受攻击的可能性。关于安全方面的考虑，在实验中尚存问题一节中会进一步讨论。

应用层协议

我们将应用层数据划分成不同的字段，赋予不同字段不同的含义，如报文功能，报文接受方用户名等。客户端和服务器在收到对方发送的报文之后，能够及时得处理。

应用层协议在应用层协议文件中有更为详细的描述。

实验中尚存问题

因为缺少多线程编程经验，在我们的实验设计中存在着诸多的问题和漏洞。我们将这些问题归纳整理如下：

- 线程同步与数据互斥访问问题

  - 在服务器中，所有的客户线程或者对战线程都会向与广播线程相关的队列写入用户状态变化信息。但是我们仅仅通过一个信号量对这个队列加锁解锁。这种情况下就会发生，线程A将用户状态变化信息写入队列之后，处理器调度广播线程进行处理，不幸的是，广播线程还没有来得及处理这个消息就被处理器挂起并调度另外一个线程B，线程B也把自己的用户状态信息写入队列，从而覆盖了线程A的用户状态信息，导致信息丢失。所以，仅仅使用一个信号量是远远不够的。作为一个典型的生产者-消费者模型，需要三个信号量才能满足要求。
  - 服务器中，所有的线程都可以访问全局的用户信息表和对战信息表，尽管彼此之间没有共享的表格项，但是表格本身是共享的，因此对于这些全局的表格，应该使用信号量保证互斥访问，尤其当多处理器操作系统将线程分配到不同的处理器，实现真正并行的时候，删除和添加表项是令人头痛的问题。

- 安全问题。应用层数据都是通过明文传递信息，给攻击者提供了很多的攻击机会。比如攻击者可以修改对战中，每个回合对方客户端发送的数据报文，使得对方玩家每次对战都是失败。

- 可进一步优化的问题。在服务器中，对战线程接受对战双方对应的客户线程提供的对战数据报文，并判断输赢。事实上，完全可以在对战线程启动之初就杀死双方的客户线程，在对战结束之后重新启动，因为所有用户的实时信息都保存在全局可访问的用户信息表中。这样的方案可以进一步降低服务器的负载。

实验合作和任务分工

在本次实验中，小组成员首先讨论了应用层协议，尤其是数据格式（具体参见协议文件），接着进一步细化了实验完成方案，并对实验任务分工如下：

- 陈越琦：完成服务器端部分；搭建客户端框架。

- 周子博：填充客户端框架，实现客户端各个功能；撰写应用层协议文件。

在实验过程中，小组成员积极讨论，对实验中出现的问题提出了各自的解决方案，并对最终的实现方法达成一致。通过小组成员之间的合作，我们成功完成了实验。

实验感悟

相比前三次实验，实验四给了我更多的感受。最大的感受是：实现一个可靠的成熟的游戏对战平台是一件很不容易的事情，除了要实现每个功能，共享数据的访问和线程间的同步，都需要花很多的精力去设计，而且即使暂时满足了要求，一旦平台规模变大，数据的存储和一致性等问题都会逐渐显现。当然，随着编程经验的增长，这些问题可能会变成“1+1=2”。但是他们都是实在存在的困难，值得我们关注。

实验时间统计

- 小组内部讨论时间总计：3 hours
- 服务器功能实现：6 hours
- 客户端功能实现：5 hours
- 服务器，客户端联合调试：5 hours
- 实验报告撰写：2 hours

参考文献

[1] 计算机网络协议开发实验讲义(2017版)，南京大学计算机科学与技术系，陈健，附录D，E，F