实验二 网络中继服务器设计与实现

一、 实验目的

1. 加深理解 epoll 网络 IO 模型
2. 掌握基于 Agent 的 c++编程方法
3. 掌握 Agent 复杂业务状态的设计方法
4. 掌握用 valgrind 检测内存出错的方法
5. 掌握用 gprof 分析程序代码 CPU 开销的方法

二、 实验环境

1. Linux 2.6 及以上内核的操作系统
2. gcc/g++，编译工具
3. gdb ，调试工具
4. nmon,系统性能分析工具
5. vi,代码编辑工具
6. tcpdump，网络协议分析工具
7. netstat，网络状态分析工具
8. man, linux 的一般系统调用使用手册查看工具
9. **valgrind,内存检查工具**
10. **gprof，程序代码性能分析工具**

三、 实验内容

1. 在实验一的基础之上，重构代码，使用面向对象的 Agent 通信对实验一的代码进行重新设计和编码调试，深入理解 epoll 的 EVENT 回调指针与 Agent 的关系。
2. 实现功能:基于 TCP 长连接模式,实现两个通信客户端之间的报文双向转发;

在报文转发过程中，任何一个链接断掉后，服务器自动关闭另外一个链接，并回收 Agent 及相关数据结构的内存及套接字；开发模拟多客户端的 EPOLL 架构的负载发生器，通过传入会话数量运行参数，模拟多个并发会话(每个会话包括两个客户端)，每个客户端在内存中生成要转发的字符串数据，通过服务器中继到同一个会话的另外一个客户端（与发送客户端都在同一个发生器进程中），由此产生对服务器不中断的数据转发压力。

3. 性能和可靠性：支持 2000 个以上的并发通信，形成 1000 个通信转发双向通道，在没有终止客户端的情况下，消息转发一直进行；关闭任何一个链接，服务器回收资源后，没有内存泄漏，没有指针越界及段错误；

4. 在功能稳定后，用 valgrind 对代码动态代码质量进行检查，任意时刻启动或终止负载发生器时，valgrind 对服务器程序分析输出均不会包括内存泄漏、越界访问、未初始化读和段错误等出错信息。如果出现错误信息，能根据 valgrind 的检测报告修改代码。

5. 关闭 valgrind 之后，使用 gprof 工具对程序性能进行综合分析。会根据 gprof 手册要求修改 g++编译参数。能根据 gprof 的输出找出程序代码的瓶颈语句或低效算法问题，并优化代码。

思考：

a) 如何设计高效的数据结构，方便多 Agent 之间的数据转发。

b) 总结内存泄露的常见原因，规范自己的编码习惯，特别是指针使用和堆内存的回收方法。

c) 思考自己低效代码产生的原因，在算法上能否改进，思考 C++ STL 容器的高效算法实现机制。

d) 从架构上能否还存在优化的余地，基本思路是怎样的？

四、 实验结果

1. 完成时间：**两周内完成**

2. 输出形式：

a) 源码：**基于 EPOLL 的中继服务器程序**；模拟高并发中继客户端的 EPOLL **负载发生器程序**。

b） **服务器程序详细设计方案**

包括文字描述、程序框架图，核心数据结构设计，类图，时序图，

c） **综合实验报告：**

实验报告应包括：程序运行的截图，遇到的问题及其解决方法。Valgrind 的代码检查报告；gprof 进行性能分析报告；用 nmon 的数据生成 CPU 的 excle 图表，并配合图表进行文字说明。

业务数据分析：在 500，1000，2000 个不同会话数的情况下，由发生器采集和计算的单个会话延时、系统平均会话延时。

d） **工作报告 PPT 及公开答辩：**

通信中继服务的理解，性能分析的结果；结合内核原理分析性能差异原因，性能进一步提升的编程思路；此阶段编程中的问题回顾和心得体会。

五、 参考教材

1. 《UNIX 环境高级编程》
2. 《UNIX 网络编程 第 1 卷：套接口 API（第 3 版）》
3. Google 资源：gdb 手册、nmon 手册
4. Linux man 手册： netstat，epoll，tcpdump、valgrind、gprof