## P2P聊天室项目要求

之前我们已经完成了P2P聊天室的初步版本,实现了用户登录,删除和遍历的功能。但是在实现的时候,我们采用的是一种试验性质方法,主要在于实现功能,而没有考虑程序的性能以及后续扩展的需求。

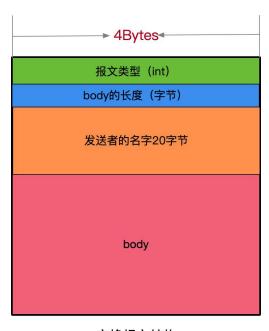
在很多软件系统中,包括 HTTP , FTP , DNS 等常见的应用层协议,以及 TCP , UDP 等传输层协议,甚至更底层的协议中,通讯双方每次发送的都是用一定格式的报文。而不是意义不是很明确的字符 串。

在我们初步设计的这个系统中, 定义了一个监听端口8888,此端口我们用作以下三种用途:

- 1. 用户上线后第一次告知其他对等方自己的登录信息; (已实现)
- 2. 用户在上线后,与其他对等方实现心跳机制; (已实现)
- 3. 用户在上线后,与其他对等方交换自己的在线用户信息,用以保证某一个用户上线后只要有一个对等方知道它在线,势必会有更多的用户知道他上线,即使他自己和其他对等方都没有在自己的配置中包含对方所在的 IP 范围(试想一下,在当前的系统中,如果某个客户L把初始IP范围设置为空或者自己的 IP,而他又是最后一个启动的,那么他将永远不会被其他对等方知晓,只有当另外一个其他用户 x 启动时,才有可能扫描到这个客户 L ,对于 L 来说,他也将只能知道有 x 在线,其他对等方他无法察觉,其他人也不会察觉到他)。未实现

在这三个用途中,**8888**这个端口负责监听,而它收到的数据就会有三种不同的格式,前两种用途我们已经在同一个端口上实现,那么第三个功能如何实现?

如果我们使用繁琐的判断,一定也可以从收到的数据中判断是何种信息,但这样系统实现起来会很麻烦,并且可读性与扩展性也会极差。所以我们需要设计一种报文结构,**不管是哪一个功能,发送的都是同一个结构的报文数据**,对报文中的数据我们可以做简单的判断与分析。



交换报文结构

以上是我为大家设计的一个报文结构,这是一个大小可扩展的报文结构,我们暂定于对其大小不做限制,因为按照我们的设计,这是一个局域网内部使用的P2P聊天系统,并且这个报文结构中,数据量最大的是上面讲到的第三个功能,因为我们的应用场景限制了其长度不会很大,我们在此不做限制,在很多协议中,一个报文的最大长度是有限制的,对于一个很大的**Message**,会切分成多个报文进行发送。

- 1. 报文类型是一个32位的整形:
  - 1. 当其值为**0**时:这是一个登录报文,**body**为空,**body**的长度为**0**收到该报文的一方需回复一个该报文;
  - 2. 当其值为1时:这是一个心跳报文,body中为空,body的长度0;
  - 3. 当其值为**2**时:这是一个在线用户交换报文,body为一个接一个的sockaddr\_in结构 体。
- 2. 需要注意的是,如果**body**的实际大小与报文中**body**长度的值不一致,则丢弃该报文;
- 3. 为了能完整的收到整个报文,接收方应该分多次接受数据,不能先入为主的认为所有的报文可以一次收完,然后进一步去判断。
- 4. 为了让大家实现的客户端都能够互相识别与通信,请大家遵循以上报文规则,在8888端口上进行通信的所有数据,都按照该报文进行封装,然后发送。

注意: 名字"suyelu"如果通过socket发送时,长度为6个字节,**而不是20字节**。

之前为了简单,咱们把主要用于工作的 INS 个线程写成了心跳线程,这样做实际上是不合理的,因为我们用多线程是为了能让多核的CPU更可能的多并发的去处理核心业务,而不是做心跳这种辅助工作,所以对于这一块我们需要进行代码架构上的重构,当然即使不重构,只要你遵循的也是咱们定义好的报文结构,依然可以去其他人进行数据通信。

1. 使用单独一个线程进行真正意义上的心跳,暂定为每隔**20s**心跳一次,心跳时,此线程对所有的链表进行循环遍历,判断当未成功心跳次数超过**3**次的为对方已下线,进行删除,在整个系统中,只在这里进行删除操作;

但是需要注意的是,如果在未心跳时,给某个用户发送信息不成功,也是一种隐含的心 跳失败,我们应该将其失败次数加一,只是在这里不做删除;

当某次发送信息成功时,应该将其失败次数重置为0;

2. 使用单独一个线程,用来在结点之间交换用户列表信息,该线程每隔**1min**将自己知道的所有在线用户信息,转发给所有其他用户,当其他用户收到这些用户信息后,在自己的链表中进行查找,如果不存在某个用户,直接将该用户加到在线用户列表中,不做登录及心跳测试。

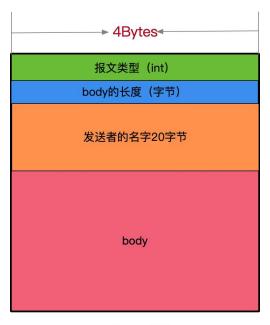
对于上面两个业务模块,如果不对线程之间做同步处理操作的话,有可能在同一时刻,一个线程在 对链表进行插入操作,另一个线程在对这个链表进行删除操作,这就是我们通常所说的线程不安全,解 决线程不安全问题,就需要引入线程互斥锁,在链表删除或者插入前,加上锁,这样能保证在此时刻, 只有自己可以对链表进行修改操作、保证数据安全;

同样,在你发送信息时,也不应该在此时刻对链表进行删除操作,因为遍历链表的同时进行删除也 是有风险的。

INS 个工作线程的作用应该是将你输入的信息转发给目标用户,如果是公聊信息,应该转发给所有人,也就是所有链表中的所有元素,而当是私聊信息的时候,只发给某个目标用户,*私聊我们暂时约定只能发给一个特定用户,而不能发给多个特定用户,当用户名有重复时,只发给第一个找到的用户*。

在信息发送过程中,我们应该使用 epoll 去处理,但是,如果如果你暂时不太会用 epoll ,或者只是在前期想先把功能跑通,也可以使用循环的方式给每一个用户去发送,只是这样效率会比较差,但是功能上没有问题。

接受信息的数据连接端口,我们使用9999,在此端口上发送的,都是用户与用户之间的交流信息。



交换报文结构

信息报文我们依然使用上面的交换报文:

- 1. 当报文类型为4时:为某一用户发给你的私聊信息;
- 2. 当报文类型为5时:为公聊信息;
- 3. 为了让大家实现的客户端都能够互相识别与通信,请大家遵循以上报文规则,在 9999 端口上进行通信的所有数据,都按照该报文进行封装,然后发送。