PSS

一个易于开发基于插件的跨平台网络服务器框架

----PSS V0.95

前言

PSS 是一个服务器框架。支持开发者使用插件(so 或者 dll)的方式，开发相关 TCP 和 UDP 的逻辑服务。 本服务框架基于 ACE，可以在 windows 和 linux 下自适应编译运行。

本服务框架的宗旨是尽量剥离网络 IO 和逻辑开发者之间的关系，让逻辑开发者更专注于自己的业务，而网络 IO 部分完全由配置文件去实现。 运维管理者可以通过辅助的管理工具，获得框架运行状态，工作线程，数据流状态，连接状态等信息。辅助运维管理，问题排查。 另外，开发者还可以在完全脱离框架的基础上，利用框架周边工具，压力测试自己的逻辑模块，这样尽量减少上线前所可能出现的问题。 在使用框架前，你可以使用框架提供的小工具，压力测试当前框架的性能指标，作为是否采用的依据。

example 下有专门的测试用例，可以提供开发者参考。 设计这个框架的目标不仅仅是一个网络 IO 的插件接口，而是一整套开发流程，尽量做到减少开发者的付出，规范开发过程（~~目前 SVN 上提供整套的框架测试工具，具体功能请参考先读我.txt~~） 希望能和大家一步步完善这个体系，真正做到有价值的框架。我的目标是，你用的爽，就是成功。也希望大家越来越喜欢它。 另外最新代码会发布在 SVN 上，如果你有 SVN，最好从 SVN 上直接下载，我会定时打版本包放在 download 里面提供下载。

一。为什么要设计 PurenessScopeServer？

在我的到目前为止开发生涯中，其实一直挺讨厌网络通讯模块的，为什么？首先还是自己做的并不是很精，而我也是一个懒人，不喜欢成天纠结着IOCP,EPOLL,SELECT 之类的模式词汇。

纵观这几年自己做过的一些项目，几乎每个项目都离不开网络，自己经历开发的几款网络游戏，所用的模型都不同，而且写多了自己也快乱掉了，而且不具备通用性和移植性。

几年前，忽然想，是否可以设计出一款跨平台，与逻辑无关的网络框架？我可不想每次都一遍遍的重写网络模块。而且要做到和逻辑无关。其实这个想法从我工作的第二年就开始有了。

自己也尝试的写了几个模块程序，那时候封装了一个 IOCP 的代码，不过这些代码只能运行在 windows 上，达不到我的要求。

后来工作需要，接触了 ACE，我用了一年半的时间，来熟悉 ACE 的各种特性，这一段时间的摸索，花费了不少时间，由于需要大量的精力去开发调试逻辑，走了不少弯路。

于是在那个项目中，我开始尝试自己的梦想，既然有跨平台的库，我为什么不能把它进化成一个消息的传输和分发的框架呢。

不过项目启动初期并不顺利，由于是 linux 下第一次使用 ace，在加上项目比较紧，走了不少弯路。最初的版本并没有达到我的预期。

于是在两年前，决定沉下心来，重做一个真正具备商业使用价值的一个框架，它的目标是解决我的数据传输和消息分发。让我不再关心这部分，而是全部精力放在逻辑开发上。

于是我定义了一个框架的目标。

1.实现逻辑和数据传输的分离，所有的逻辑全部通过模块的方式加载（dll 或者是 so）。

2.逻辑模块可以通过管理工具轻松的插拔，类似于 USB 硬盘，这样做的好处是，我可以方便的调试我的模块功能，并可以提供服务器不停机升级指定模块的服务。

3.支持 TCP 和 UDP 协议，并且，允许程序员自己去定义协议的格式，可以是压缩，加密，二进制协议，文本协议。框架不应该制约数据的格式

4.支持多线程的数据分发模式，在对于每个模块的某个协议而言，它可以清楚的知道这个数据包是来源于谁。

5.为每个模块提供收发模块，模块可以轻松使用这些资源，并且不用去维护这些资源。对于调用者是透明的。

6.必须有足够的数据监控能力，一个链接收发了多少数据，服务器有多少链接，每个协议的实际处理时间，工作线程的状态和数据吞吐量，这些都必须提供给开发者，用于对自己模块服务的监控。

7.强大的日志系统，可以记录框架在服务中产生的异常和正常的数据。

8.支持数据的转发服务，开发者可以轻松的把数据转发给另外的服务器。

这样，对于我而言，我不在关心这些数据底层的问题，而将精力放在不同的逻辑数据处理上。并可以提供不停机的模块卸载和加载。从而减少服务重启进程。

当然，要做到上述的这一切，付出是必不可少的。从最初的 0.50 版本，到现在的版本，弯路也真的走了不少，光 ConnectHander 的代码就至少重写了 3 遍。当然，一切都是为了让它变的更好。

PurenessScopeServer 的名字由来，也是由 Pureness（纯粹）+ Scope（范围），希望它越来越有价值，能够真正的帮助开发者起到实际的作用。

在这里，我期待有兴趣的朋友更多的参与到它的开发中来，并提供出你的宝贵建议，让它更方便的给更多的开发同仁们创造实际的便利，这就是它的价值所在。

感谢 winston，KimilesWood，jackypeng，jamesyang680，7cat，modern，badbrain，Dave，五彩修改花针等朋友的大力支持（排名不分先后），正是有你们的存在，它才会一步步的成长，谢谢你们。

说实话，我并不期待它未来会成为什么，而是希望它真正能给你我，这些开发者，多一分的自信和便利。并能用它展现出你的精彩，越短的时间实现你的想法是它的目标，而不是花精力研究它本身。

越简单就越不容易出错，我一直坚信的信条，尤其是简单到能让初学者都顺利看懂上手的代码是我追求的目标，这才是价值的体现。所以，在框架设计中，我尽量用简单的代码解决问题。方便与阅读者，也方便与自己。

二。为什么要用 PurenessScopeServer，它能给我提供什么？我需要为此付出什么？它适用于什么领域范围？

PurenessScopeServer 实际就是一个数据分发平台。支持自定义的格式的数据包（这部分由开发者自己决定，具体使用请参考《PacketPrase 说明书》）

开发者可以通过配置文件来控制服务器功能（这部分请参考《配置文件说明书》），而不用在开发额外的代码。

其实我的终极目标是，开发者可以不用阅读框架的代码（有兴趣的除外），你只要懂得如何编译它，学会配置它，让它为你服务，就可以了。你可以专注于你的逻辑实现。

它可以负责传输数据，相应的，在逻辑模块，你需要遵循一定的格式和方法，去获得这些数据，并订阅那些你感兴趣的消息。

框架支持多重订阅，就是一个消息，你可以根据消息的 ID，分别在不同的模块定义接受者。

（举个例子，比如我可以在模块 1 中订阅 1 号消息，记录消息到来的时间日志，在模块 2 中订阅 1 号消息，处理消息的本体，这样代码看上去非常清晰。框架会发给这两个模块，互不干扰）

那么，开发者可以不可以脱离框架写自己的逻辑呢？也就是说，我可否在没有框架的时候，编写那些模块文件？比如 dll 和 so？答案当然是可以的。具体方法请参考我的 example-Module 文件夹下的例子。

目前的例子包括基础的数据收发方法，UDP 和 TCP 的，还有一个用户登录引

擎，包括数据多级缓冲机制（这部分请关注 acejoy 上我写的多级缓冲机制的文章）。以后我会不断的持续添加样例模块，帮助大家分享一些开发方法。

PurenessScopeServer 由 4 部分组成,其中两部分来协助完成。他们分别是：

1.框架程序本身（不需要开发者去编写任何代码，当然，配置文件需要开发者去做一些配置）

2.PacketParse 部分（需要开发者编写，用于定义数据协议的格式，目前是包头和包体，还有一个标明协议的标记号）

3.功能模块（需要开发者编写，完成你的逻辑）

4.客户端框架管理工具（这部分不需要开发者去编写，开发者可以用这个工具监控框架服务的各种状态）

目前，PurenessScopeServer 适合于 TCP 和 UDP 协议的开发。你可以通过配置框架支持以上协议。

三。用 PurenessScopeServer 需要注意什么？

需要注意的是在 windows 和 linux 下，由于环境的不同，需要一些基础的网络知识配置。

比如在 linux 和 windows 下，进程 IO 数的配置，在 linux 下注意，如果配置不对可能会导致一些链接建立不正常。（linux 下 ulimit -a 查看）

具体请参考 doc 文件夹下的说明文件。

本程序需要依赖 ACE 才能运行，请到

http://www.cs.wustl.edu/~schmidt/ACE.html 下载 ACE 的运行库。

编译教程请参考 http://www.cs.wustl.edu/~schmidt/ACE-install.html

在编译完 ACE 后，请在对应权限的.bash_profile 文件中添加

这里要注意，如果你要使用 IPv6，必须在 ace 的 config.h 里面添加

#define ACE_HAS_IPV6

如果想用 configure 去编译。

记得先 yum -y install openssl-devel (RedHat Linux)

否则会提示找不到 ssl 的库

或者 ../configure --disable-ssl

如果你用不到 SSL 的话

或者，如果你想手动编译

配置环境变量：

vi /etc/profile

增加如下的内容

ACE_ROOT=/Software/ACE_wrappers ------就是上面存放 ACE 源文件的目录

export ACE_ROOT

LD_LIBARY_PATH=$ACE_ROOT/ace:$LD_LIBRARY_PATH

export LD_LIBRARY_PATH
source /etc/profile

你可以先在 ace 目录下创建一个 config.h
添加如下代码
#include "ace/config-linux.h"
然后进入$ACE_ROOT/include/makeinclude
新建 platform_macros.GNU，在 platform_macros.GNU 中加入一行如 下：
include platform_linux.GNU
然后回到 ace 目录
make
编译完成后，检查上一层 lib 文件夹下的 libace.so 是否存在，存在则编译成功。

LD_LIBRARY_PATH=/usr/local/lib/:/home/freeeyes/PurenessScopeServer/PurenessScopeServer/Linux_Bin/
export LD_LIBRARY_PATH

ACE_ROOT=/usr/src/ACE_wrappers
export ACE_ROOT
解释一下
/home/freeeyes/PurenessScopeServer/PurenessScopeServer/Linux_Bin/是你的工程工作路径，/usr/local/lib/是你的 libace.lib 的路径
ACE_ROOT 是你的 ACE 安装路径。

如果要运行./runlinuxmpc.sh 生成编译配置文件，请先确认安装了 perl。
并请在环境变量里面添加
PATH=$PATH:/root/tools/ACE_wrappers/bin
export PATH 你把这个加到你的文档中吧

另外，如果在 linux 下使用 epoll 模型的话
请配置/etc/security/limits.conf 文件。

在里面添加
```
*       soft    noproc  6000
*       hard    noproc  6000
*       soft    nofile  6000
*       hard    nofile  6000
```

如果你要用默认的并行端口打开数，比如 1024，那么请修改 define.h 里面的 MAX_DEV_POLL_COUNT 为 1000

得到代码后，请先编译 PacketParse 目录下的文件。成功生成

libPacketParse.so 后，拷贝到 PurenessScopeServer 下的 Linux_bin 目录下，然后在 Linux_bin 下运行 make 命令。

首先编译 PurenessScopeServer 工程，然后编译 PacketPrase 工程，把生成的 so 或者 dll 拷贝到 PurenessScopeServer 工程下的相应目录下(Windows 是 Windows_bin，Linux 是 Linux_Bin)

然后编译 example-Module 里面的你想测试的模块。把生成的 so 或者 dll 放在 PurenessScopeServer 工程下的相应目录下(Windows 是 Windows_bin，Linux 是 Linux_Bin)，

并修改 main.conf 文件中的 ModuleString，你也可以把你的所有逻辑 so 放在一个你指定的目录下并设置 ModulePath。(具体请参考，配置文件说明书)

附注：

目前 Pss 支持加载为 windows 服务，如果要想 Pss 以 windows 服务启动

首先确保自己在管理员权限下。

然后打开 main.xml

修改 <ServerType Type="1" name="Pss Service" displayname="PssService">

这里 Type=1 为程序以 windows 服务的形式启动

启动方法，需要在命令行模式下(cmd)，到当前目录，然后运行框架 PurenessScopeServer_D.exe -i

注册 windows 服务。

然后在 windows 服务管理器下（或者在命令行下，使用 PurenessScopeServer_D.exe -s）启动和停止服务。

如果要删除服务，则需要在命令行下 PurenessScopeServer_D.exe -r

注意，如果要以服务形式启动，必须框架依赖的 dll 都在当前路径下，或者在环境变量 path 指定的路径下。否则会启动失败。

版本变化

V0.95

* 重新设计 MessageServer 多工作线程的支持。
* 支持多反应器客户端连接同步。
* 重构框架代码中所有的内存操作边界检查。
* 框架支持插件自检功能。
* 解决了群发消息的一处 BUG。
* 添加服务器心跳超时事件，插件可订阅。
* 更新检查连接超时更新机制。
* 添加新的 BloockSize 相关设置。
* 提交对服务器断开连接的优化。
* 删除无用的信号量注册，Valgrind 检测内存无泄漏和警告达成。
* 修改溢出 PSS_ClientManager 可能存在的 BUG，
* 添加插件可以控制客户端连接断开的两种状态，一种是立即断开，一种是发送完成断开。
* 添加判断客户端连接是否存在的接口。
* 添加插件中获取框架所有插件信息
* 添加服务器间传输获取 IClientMessage 接口，以及通过 ServerID 获得远端 IP 的接口
* 添加监控工具邮件发送功能。
* 添加服务器间通讯第一次连接连接成功通知 Reconnect()。
* 添加 Lua 测试插件，在 Linux 下编译运行测试通过。
* 重写了插件热加载功能。
* 添加对发送数据超时告警的插件通知机制。
* 添加对插件对本地监听端口获取的接口。

V0.94

* 服务器间连接支持指定本地 IP 和端口功能。(有些需求远端需要指定本地 IP 和端口)
* 添加单位时间连接量和断开量统计功能。
* 添加单位时间连接量和断开量统计告警功能。
* 添加告警邮件自动发送功能。
* 添加新的 BuffPacket 支持类型,支持 String 类型的导入导出。
* 修复 Linux PSS 关闭时在某些情况下程序不退出的 BUG。
* 修改 Reactor 主线程为监控线程。
* 重新设计告警配置文件，并支持邮件告警功能。
* 添加对 PSS 的允许最大连接数的远程控制功能。
* 修复 UDP 数据发送的一个 BUG。
* 优化对无头数据包解析用例的代码。
* 添加了数据包头信息代码样例。

* 修改例子代码支持新的 PacketParse 功能以及新接口。
* 添加日志跟踪器工具。
* 添加了自动添加和关闭指定监听端口功能，支持插件内管理以及管理工具管理。
* 开放关于链接别名的函数给逻辑插件使用。
* 添加服务器间连接异常的时候返回远程 IP 功能。
* 解决一个 HTTP 数据包解析造成堆栈崩溃的 BUG。
* 添加定时器插件用例。
* 优化框架代码，减少不必要的代码，优化流程以及代码顺序。

V0.93

* 支持只有包头不含包体的数据。
* 添加了 Websocket 对接协议的 PacketParse 用例。
* 添加了 Http 对接协议的 PacketParse 用例。
* 添加了对 ACE_DEBUG 日志输出文本规格化的支持。
* 修复了一个 ACE_DEBUG 死锁的问题。
* 更新 TcpPost 用例，添加数据回发完整性判定。
* 添加了在 Linux 下对 epollet 模式的支持。
* 修复了一些 Reactor 和 Proactor 下的 BUG。

V0.92

* 优化 ConnectHander 的代码结构
* 添加发送超时配置文件
* 添加了新用例，ftp 用例，实现相关了目录浏览，下载和上传(PSS 插件以及测试客户端)。
* 添加如果二级缓冲用例，实现共享内存和数据库的同步(PSS 插件以及测试客户端)。
* 添加了插件间相互调用用例，实现了插件间的通讯((PSS 插件以及测试客户端)。
* 添加 Linux 下自动设置当前工作目录的功能。
* 添加了 PSS 自测插件功能，实现 PSS 数据包自测用例(PSS 插件以及测试客户端)。
* 更新了 PassTCP 工具，可以支持二进制和文本的数据包发送。
* 重写了 PSS 日志接口，支持输出到屏幕和文件的选择，并支持文本和二进制的记录。
* 添加了 Proxy 代理服务器数据包转发插件，实现了 PSS 网关功能(PSS 插件以及测试客户端)。
* 添加了新的 API，允许插件可以获得工作线程的数量以及当前工作线程的 ID。
* 添加了 uint64 位数据的网络字序和主机字序的转换函数。
* 添加了再 Linux 下自检当前文件并发数的功能，如果文件并发数小于配置文件设置则框架会自动尝试提升当前文件并发数，如果失败则提示框架启动失败。
* 添加了对 core 文件的设置，开发者可以通过配置 core 文件大小来启动当前 PSS。
* 添加了发送缓冲区自检功能，当发送字节和对端收到字节不成正比时，按照配置文件的规则回收当前连接。
* 添加了对 BACKLOG 的设置，可以提升在大并发连接下的连接效率。

V0.91

* 修改了 dev_poll 下设置并行最大连接数配置文件对应关系。
* 再次更新 PacketParse 接口，优化了接口结构，使得开发者更清晰的看到自己要实现代码的地方。
* 更新插件压力测试工具。支持 TCP 和 UDP 压力测试，并会生成测试报告。
* 更新 MakePacket 回应包添加 CommandID 参数，你可以根据不同的连接 ID 决定处理你的发送组包逻辑（比如加解密的随机算法）
* 服务器添加了对 UDP recv 超时的设置
* 更新了 UDP Proactor 模式下的 UDP 设置参数。
* 添加对 IPv4 和 IPv6 的支持。服务器可以使用 IPV6 的地址，但是前提是 OS 必须支持 IPV6
* 添加类视图文档，以 PDF 文档形式提供。
* 更新插件压测工具，提供多线程压测插件功能。
* 修改了 TCP 和 TCP 服务器间测试用例，实现了透传数据的压测用例。
* 添加了 Linux 下结束进程的脚本
* 添加了时间成本宏，你可以用于你的逻辑中，测试函数执行效能
* 添加三个 Try catch 宏用于程序调控。你可以用于你的逻辑中，套在函数里。
* 添加了服务器发送数据水位标，如果服务器发快客户端收慢，那么会有一个阀值保证正常的连接不受影响。
* 修改了若干测试出的 BUG，具体可以浏览 SVN 更新日志。
* 添加了 DEBUG 模式，在 debug 下支持输出所有的数据包文件。这些数据包可以变为压测的回放的依据。