GatewayServer 设计

版权说明

对比特项目感兴趣，可以联系这个微信。

代码 & 板书链接

https://gitee.com/bitedu-tech/cpp-chatsystem

功能设计

网关服务器在设计中，最重要的两个功能：

- 作为入口服务器接收客户端的所有请求，进行请求的子服务分发，得到响应后进行响应
- 对客户端进行事件通知（好友申请和处理及删除，单聊/群聊会话创建，新消息）

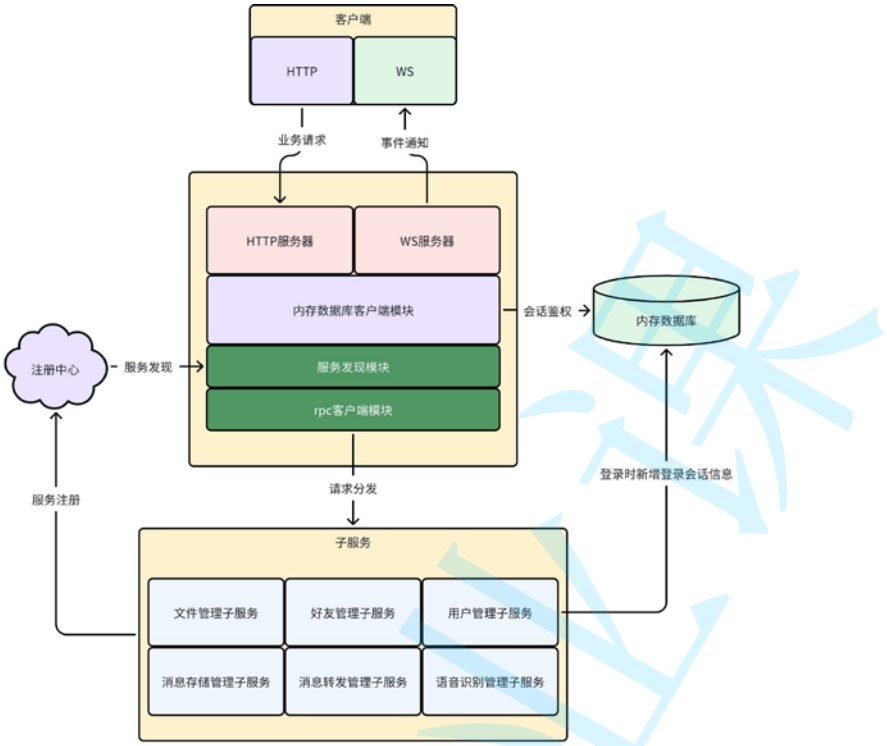
基于以上的两个功能，因此网关服务器包含两项通信：

- HTTP 通信：进行业务处理
- WEBSOCKET 通信：进行事件通知

模块划分：

1. 参数/配置文件解析模块：基于 gflags 框架直接使用进行参数/配置文件解析。
2. 日志模块：基于 spdlog 框架封装的模块直接使用进行日志输出。
3. rpc 服务发现与调用模块：基于 etcd 框架与 brpc 框架封装的服务发现与调用模块
    a. 因为要分发处理所有请求，因此所有的子服务都需要进行服务发现。
4. redis 客户端模块：基于 redis++封装的客户端进行内存数据库数据操作
    a. 根据用户子服务添加的会话信息进行用户连接身份识别与鉴权
5. HTTP 通信服务器模块：基于 cpp-httplib 库搭建 HTTP 服务器，接收 HTTP 请求进行业务处理。
6. WEBSOCKET 服务器模块：基于 Websocketpp 库，搭建 websocket 服务器，进行事件通知。
7. 客户端长连接管理模块：建议用户 ID 与长连接句柄映射关系，便于后续根据用户 ID 找到连接进行事件通知

模块功能示意图：



接口实现流程：

用户名注册

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 查找用户子服务
3. 调用子服务对应接口进行业务处理
4. 将处理结果响应给客户端。

用户名登录

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 查找用户子服务
3. 调用子服务对应接口进行业务处理
4. 将处理结果响应给客户端。

短信验证码获取

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 查找用户子服务

3. 调用子服务对应接口进行业务处理
4. 将处理结果响应给客户端。

手机号码注册

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 查找用户子服务
3. 调用子服务对应接口进行业务处理
4. 将处理结果响应给客户端。

手机号码登录

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 查找用户子服务
3. 调用子服务对应接口进行业务处理
4. 将处理结果响应给客户端。

用户信息获取

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找用户子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

修改用户头像

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找用户子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

修改用户签名

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找用户子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

修改用户昵称

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找用户子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

修改用户绑定手机号

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找用户子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

获取好友列表

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找好友子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

发送好友申请

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID

3. 查找用户子服务
4. 根据请求中的用户 ID，调用用户子服务，获取用户的详细信息
5. 查找好友子服务
6. 调用子服务对应接口进行业务处理
7. 若处理成功，则通过被申请人 ID，查找对方长连接
    a. 若长连接存在（对方在线），则组织好友申请通知进行事件通知
8. 将处理结果响应给客户端。

获取待处理好友申请

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找好友子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

好友申请处理

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找用户子服务
4. 根据请求中的用户 ID，调用用户子服务，获取申请人与被申请人的详细信息
5. 查找好友子服务
6. 调用子服务对应接口进行业务处理
7. 若处理成功，则通过申请人 ID，查找申请人长连接，进行申请处理结果的通知
    a. 若处理结果是同意，则意味着新聊天会话的创建，则对申请人继续进行聊天会话创建通知
        i. 从处理结果中取出会话 ID，使用对方的昵称作为会话名称，对方的头像作为会话头像组织会话信息
    b. 若处理结果是同意，则对当前处理者用户 ID 查找长连接，进行聊天会话创建的通知
        i. 从处理结果中取出会话 ID，使用对方的昵称作为会话名称，对方的头像作为会话头像组织会话信息

    c. 清理响应中的会话 ID 信息，
8. 将处理结果响应给客户端

删除好友

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找好友子服务
4. 调用子服务对应接口进行业务处理
5. 若处理成功，则通过被删除者用户 ID，查找对方长连接
    a. 若长连接存在（对方在线），则组织好友删除通知进行事件通知
6. 将处理结果响应给客户端。

搜索用户

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找好友子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

获取用户聊天会话列表

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找好友子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

创建多人聊天会话

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找好友子服务

4. 调用子服务对应接口进行业务处理
5. 若处理成功，循环根据会话成员的 ID 找到他们的长连接，
    a. 根据响应中的会话信息，逐个进行会话创建的通知
    b. 清理响应中的会话信息
6. 将处理结果响应给客户端。

获取消息会话成员列表

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找好友子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

发送新消息

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找消息转发子服务
4. 调用子服务对应接口进行业务处理
5. 若处理成功，则根据处理结果中的用户 ID 列表，循环找到目标长连接，根据处理结果中的消息字段组织新消息通知，逐个对目标进行新消息通知。
6. 若处理失败，则根据处理结果中的错误提示信息，设置响应内容
7. 将处理结果响应给客户端。

获取指定时间段消息列表

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找消息存储子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

获取最近 N 条消息列表

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找消息存储子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

搜索关键字历史消息

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找消息存储子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

单个文件数据获取

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找文件子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

多个文件数据获取

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找文件子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

单个文件数据上传

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化

2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找文件子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

多个文件数据上传

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找文件子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。

语音转文字

1. 取出 HTTP 请求正文，进行 ProtoBuf 反序列化
2. 根据请求中的会话 ID 进行鉴权，并获取用户 ID，向请求中设置用户 ID
3. 查找语音子服务
4. 调用子服务对应接口进行业务处理
5. 将处理结果响应给客户端。