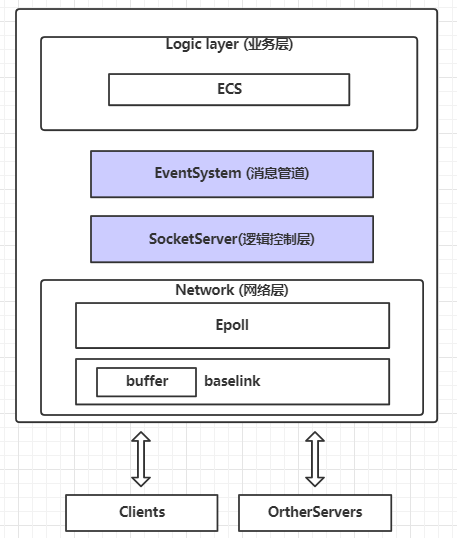
**服务器精英训练营**

##### 第五周作业（葛华盛）：

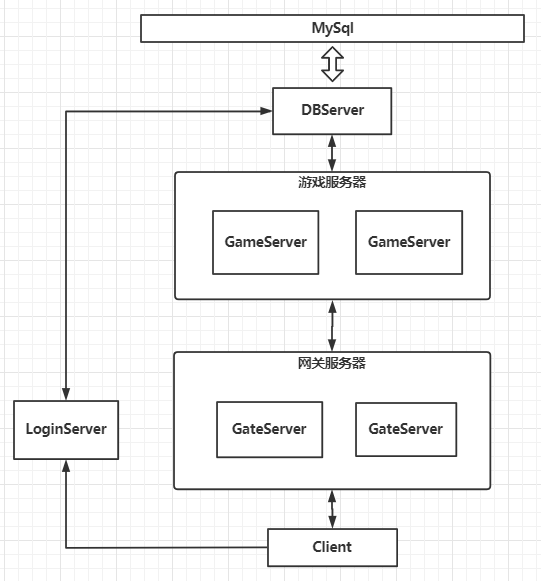
**1、服务器架构设计**

**(**[**GameServerDev · master · 葛华盛 / Server\_Training\_Camp · GitLab (123u.com)**](http://git.intra.123u.com/huashengge/Server_Training_Camp/tree/master/GameServerDev)**)**

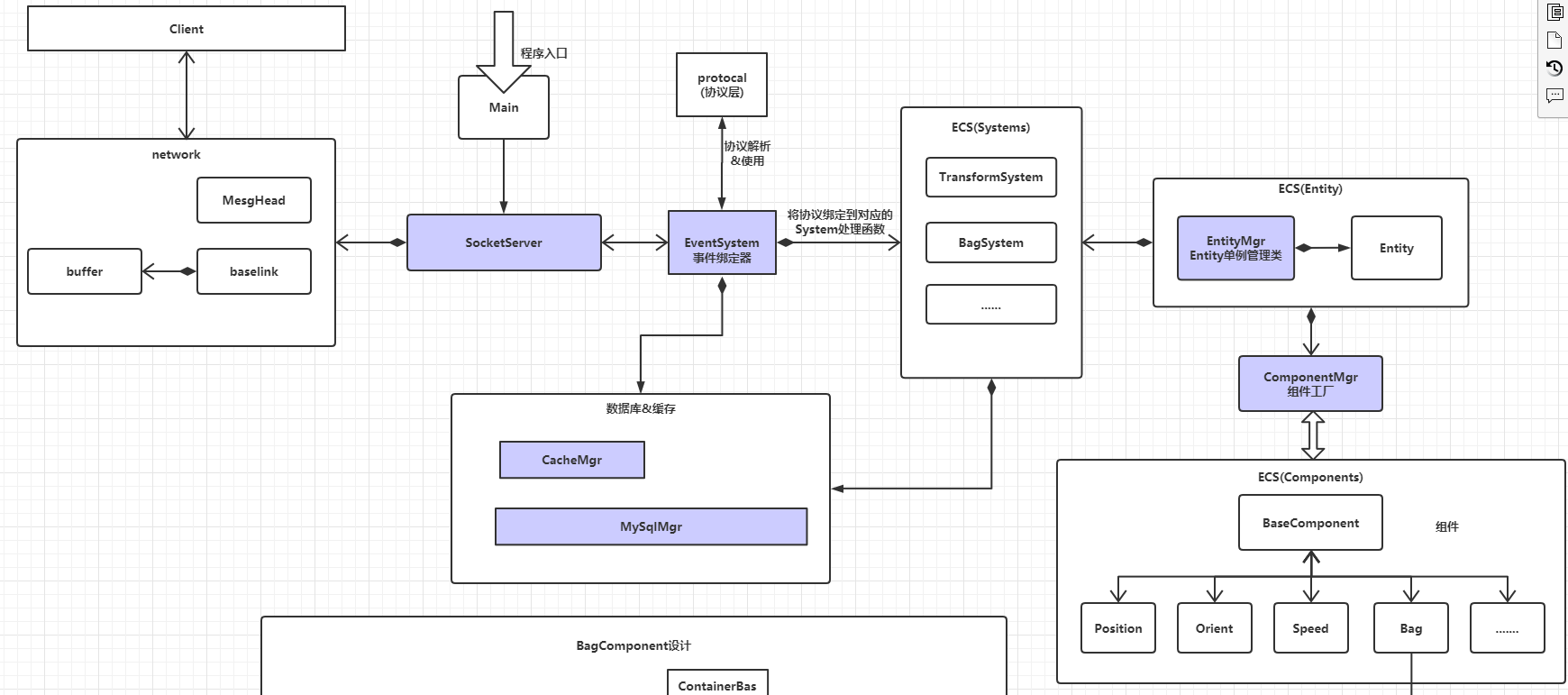
**(git@git.intra.123u.com:huashengge/Server\_Training\_Camp.git)**

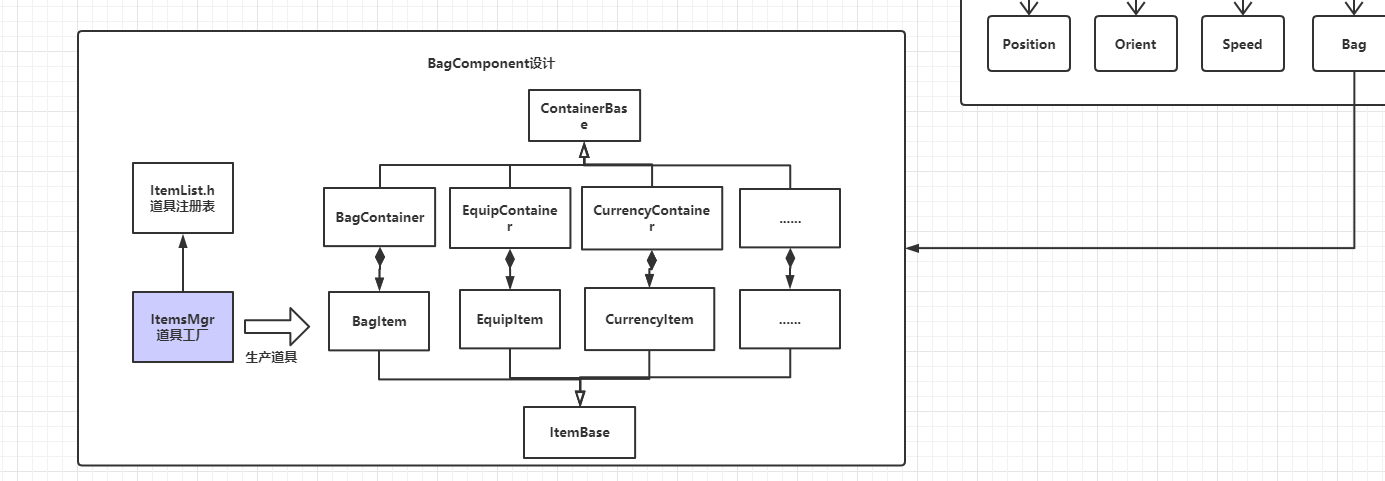


*图1：单个服务器架构图*



*图2：总体服务器架构图*



*图3：GameServer详细设计（紫色为单例类）*

**（框架图1-2可以看：**[**服务器架构图 - ProcessOn**](https://www.processon.com/diagraming/5fcd2159e401fd1e15cb57b5)**）**

**（框架图3可以看：**[**https://www.processon.com/view/link/5fb90cf87d9c0857dda95da5**](https://www.processon.com/view/link/5fb90cf87d9c0857dda95da5)**）**

**框架开发总结：**

这周内容主要是在上周内容的基础上进行对上周的游戏服务器的拆分，主要的思路和PPT上基本一样，分为Login、Gate、Game、DB四种类型的服务器。单个服务器的框架主要继承于上周的内容，基本上分为几个模块，一是网络模块和客户端或是其他服务器交互，二是流程处理模块（SocketServer&EventSystem）三是ECS的玩家数据&逻辑处理模块，四是数据落地模块。基本上实现了多服务器，切每种服务器（Gate、Game、DB）多个并存。

对于上述的四个服务器而言，前两个模块在框架基本上没有变化，即网络模块与流程处理模块，服务器或是客户端之间都通过protobuf定义的协议来进行操作。其中为了实现保存连接类型的功能需要在网络模块（Epoll）中增加对每个socket连接属性的标注，同时为了存储玩家消息转发的目标服务器，需要在SocketServer中增加类似跳表的数据来记录某个玩家消息下个要转发服务器的地址或是套接字的句柄。其次就是在消息管道类EventSystem中，对于绑定的消息都有很大的改变，除了GameServer的可以继承上周的内容，其他服务器需要自己消息所对应的函数。

对于第三模块逻辑的设计，GameServer基本上保留的从前的内容，采用了ECS的架构来处理玩家的操作，而其他几个模块由于逻辑都挺简单的，基本上就设计了单例类或是有的直接在EventSystem处理转发了。

还有一个比较重要的方面就是，服务器注册到其他服务器的方式，以及如何保存这个服务器信息的处理方式。我的处理思路是由于两个服务器建立Socket连接时并不知道对方是什么服务器，所以在建立连接后需要发一条服务器注册的消息，另一个服务器在收到这条消息后将这个连接标注为特点的服务器，然后发回包并填上错误码和自己的信息。其次各个服务器连接是有主次的，我的设计主要是Gate连Login，Game连Gate，Game连DB。

**框架设计细节：**

LoginServer：Login的逻辑比较简单，在打开服务器后，接受gate的注册，然后接受玩家的注册消息并选一个gate发回给玩家，并将这个玩家的注册消息发送给gate，并在login上保留一下玩家信息，未来可以做断线重连。

GateServer：打开时发送注册的消息到Login，并接收game的注册，然后每收到一条消息就判断一下消息的类型，是玩家的消息就判断一下在不在gate上，然后转发。其他服务器消息如login的玩家注册消息就通过EventSystem找对应的函数处理

GameServer：打开时注册到Gate、DB，对于收到的消息做逻辑处理

DBserver：接收消息，处理数据，其中由于目前我没有做delay RPC，GameServer一般是存的消息。后来需要做delay RPC才能等待DB回数据后处理。

数据落地使用protobuf，pb通过分块来减少IO的数据量，这次我是把player信息和BagInfo分开了。

**框架代码各个的功能与设计意义（GameServer）（其他的几个server基本上是网络模块加一些简单的逻辑）：**

**（**[GameServerDev · master · 葛华盛 / Server\_Training\_Camp · GitLab (123u.com)](http://git.intra.123u.com/huashengge/Server_Training_Camp/tree/master/GameServerDev)**）**

**.**

**Buildincr (增量编译)**

**Buildnew（重新编译）**

**Client （客户端）**

**GateServer**

**DBServer**

**LoginServer**

**GameServer**

**├── bin (执行文件)**

**├── build（cmake&make的文件夹）**

**├── CMakeLists.txt**

**├── components （组件）**

**│   ├── ItemComponent.h (背包组件，相当于BagComponent，刚开始名字没想好)**

**│   ├── PlayerInfoComponent.h （用户信息组件）**

**│   └── TransformComponent.h （位置信息组件，有三个component）**

**├── include （基本上所有的头文件）**

**│   ├── BagContainer.h**

**│   ├── BaseComponent.h**

**│   ├── baselink.h**

**│   ├── bloom\_filter.h**

**│   ├── buffer.h**

**│   ├── CacheEnum.h**

**│   ├── CacheMgr.h**

**│   ├── ComponentsMgr.h**

**│   ├── CurrencyContainer.h**

**│   ├── Entity.h**

**│   ├── EntityMgr.h**

**│   ├── EquipContainer.h**

**│   ├── EventSystem.h**

**│   ├── ItemBase.h**

**│   ├── ItemContainer.h**

**│   ├── ItemList.h**

**│   ├── ItemsMgr.h**

**│   ├── MesgHead.h**

**│   ├── MsgHandler.h**

**│   ├── MySqlMgr.h**

**│   ├── Player.h**

**│   ├── PlayerMgr.h**

**│   ├── Singleton.h**

**│   ├── SocketClient.h**

**│   ├── SocketServer.h**

**│   └── util.h**

**├── Network （网络库）**

**│   ├── baselink.cpp （封装的linux Sokcet接口）**

**│   ├── buffer.cpp （封装的buffer，放在baselink中）**

**│   ├── Epoll.cpp （封装的Epoll，管理所有的baselink）**

**│   └── MesgHead.cpp （协议头的储存和解析都在这里）**

**├── protocal （协议）**

**├── Server （服务器的大多数类）**

**│   ├── CacheMgr.cpp （redis的单例类）**

**│   ├── CMakeLists.txt**

**│   ├── ComponentsMgr.cpp （组件工厂，用注册表方便的生成组件）**

**│   ├── EntityMgr.cpp （实体管理类）**

**│   ├── EventSystem.cpp （事件绑定的类，相当于一个消息管道）**

**│   ├── ItemsMgr.cpp （道具的类）**

**│   ├── main.cpp （程序的入口）**

**│   ├── MySqlMgr.cpp （数据库的）**

**│   └── SocketServer.cpp （相当于一个epoll，管理了消息的收发的流程）**

**└── systems （ECS的system部分）**

**├── BagSystem.cpp （Bag的system处理，道具之间的处理都在这里）**

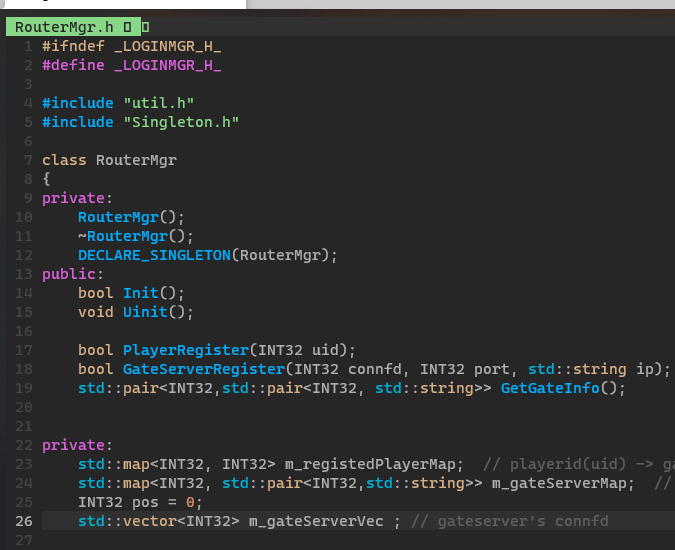
**├── BagSystem.h**

**├── TransformSystem.cpp （移动system，上周的内容）**

**└── TransformSystem.h**

**协议定义详情（GameSpce.proto & MsgID.proto）：**

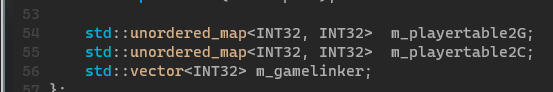
Login存储玩家数据（RouterMgr）：



保存socket连接类型（Epoll）：

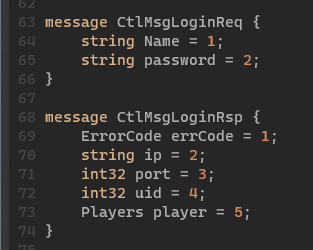


玩家的跳表（SocketServer）：

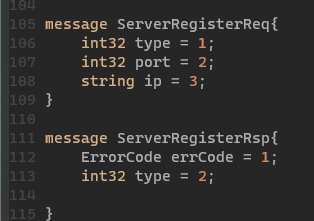


协议的定义（GameSpec.proto）：

玩家登录：



服务器注册到其他服务器：



Login将玩家注册到Gate：

