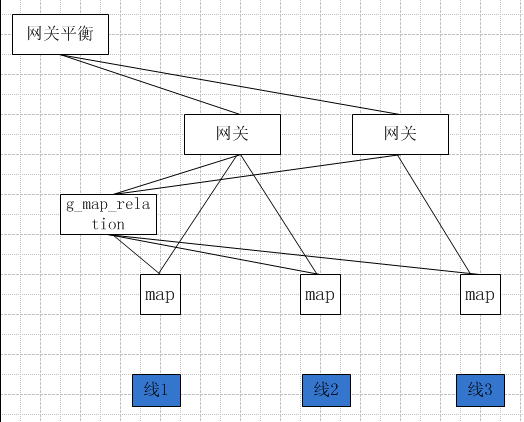
模型图:



这个设计的意图，是以最大的通讯单元视成一个完整的游戏世界为出发点。即，不管线与线之间通讯还是跨服等，都将其视为普通地图的一部分。只是在转发的时候加以区别是哪个线，哪张图。

**各部件功能**

网关平衡:

顾名思义，负责控制最外层网关的连接平衡。

当客户端请求连接时，返回当前负载量最小的网关IP地址信息、各条线的人数信息。

网关:

负责整个最大通讯单元(如果线与线可以通讯，那就是以服为单位)客户端的并发连接。可以视服中线状况而调整。比如一个服最开始有8条线，那么可以配置3台网关，随后随着线的减少网关也可逐渐减少。

维护的路由信息不涉及游戏内的具体场景，只包括外部信息。比如加解密包，包的校验，用户连接有效的检测，包的转发

g\_map\_relation:

当人物不在同一个场景进行交互的时候，通过它来查询和转发.因为这种交互往往带有延迟的性质，前期不作为多台机器考虑。但是做为多核的考虑，处理能力单核有把握到2-3万/s的计算量

维护所有map间的人物关系列表，关系网

map:场景服务器

**各部件维护的具体资源**

网关平衡:

接受网关的连接。维护一张网关连接的资源表，以及网关定时更新各个线的人数负载信息。

网关:接受g\_map\_relation、map服务器的连接。维护的资源路由结构如下:

每个玩家连接

{

线编号

map机器IP数组索引

}

场景服务器

{

线编号

map机器IP数组

}

g\_map\_relation:

接收所有map服务器连接。维护的资源路由结构如:

每个玩家

{

线编号

地图编号

玩家:{连接的网关}

}

关系网

{

线编号

公会

家族

}

map:场景服务器

**工作流程模拟**

G=g\_map\_relation

当一个玩家登陆:

1. 从网关平衡器处获取连接的信息,连接至网关,网关将玩家路由设为指向g
2. 玩家发送登陆信息(用户名密码),网关转发给g
3. G进行验证等工作，如果失败则关闭，成功则取得相应的玩家所在地图信息等，在G本地建立玩家路由表，同时返回给网关。

当一个玩家跨线组队:

假设A和B在不同的场景(不同的线也一样)

1. A发出组队信息，场景服务器发现B不在当前场景，逐向G发出请求
2. G通过建立的人物关系表查到该B所在的网关，转发给网关
3. 网关转发给B所在的场景服务器

当一个玩家工会内聊天:

1. 玩家发出聊天信息
2. 场景服务器收到后，不做查询，直接转发给G
3. G通过关系网查到各个玩家所在的网关信息，转发给网关
4. 网关统一发送