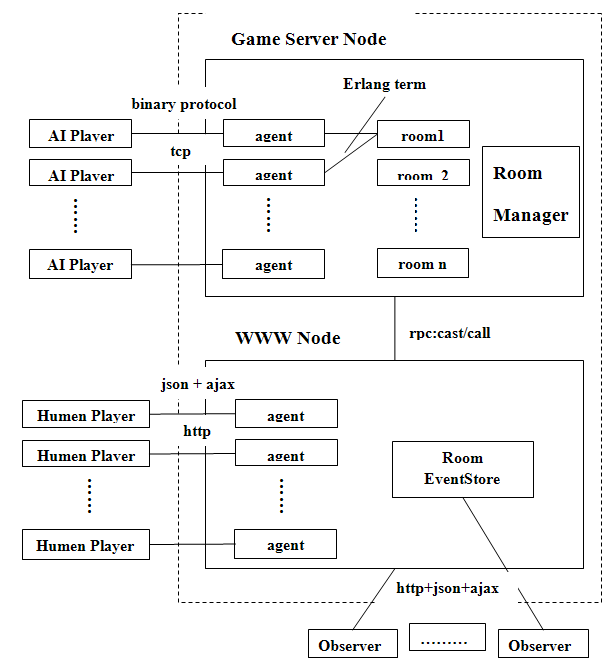
# AlphaTTT多人对弈系统

**孙鸣**

## 总体架构

### 1、组成



**AlphaTTT多人对弈系统架构图**

服务器部分由两个Erlang Node构成，分别为Game Server和WWW Server。Game Server主要负责AI棋手的接入，多room的管理和对弈流程的控制，基本类似于本次培训中的tcp\_server等相关部分。

和AI棋手的接入采用tcp,这样方便各种程序语言棋手的接入。tcp协议部分课堂上采用的是term\_to\_binary,构建该系统时需要根据目前的指令（enter\_room,leave\_room,update,get\_move 等）自定义编解码。  
  
 每个AI棋手接入后，Game Server都为其创建一个agent进程，tcp的编解码就在这里完成，以后，该进程和room的消息交互就采用Erlang数据。  
  
 WWW Server部分主要负责WEB客户端的HTTP接入，同时处理人类棋手的接入、对弈或者是观战。

上图虚线框内为系统开发的内容。  
  
**2、对弈部分设计**  
 人类棋手需要对弈时，可以输入昵称登入（要求昵称不能重复），WWW Server中需要为每个人类棋手创建一个agent进程。

该agent进程负责保存room推送过来的数据，供页面poll查询；数据包括该棋手所在棋局的对弈情况（update）（初期可以直接放在内存中，应该无需持久化）；也包括get\_move这样的请求。

页面就用最简单的poll方式（ajax＋json）定时（20ms或者50ms）从WWW Server获取这些数据，加以处理，对棋局可以展现，对get\_move,可以请求客户输入。  
  
 另外，该agent进程还负责处理人类棋手的enter\_room和leave\_room，该进程可以通过节点间的rpc cast/rpc call和Game Server中的room联系。

同时该agent进程还需负责监控页面，如果一段时间不再收到poll请求就认为页面关闭，该agent进程要死亡，从而释放出room和昵称。  
  
 这样，room面对的棋手无论是AI还是人类，处理逻辑都是一致的。  
  
3、观战部分设计

每个观战者都可以选择room来观战，WWW Server中要有一个RoomEventStore（可以简单用ets实现），room将棋局变化事件（谁下的，当前盘面）推送给RoomEventStore，方便观战页面poll查询。

4、扩展

两个Node调试时可以放在同一台机器上。实际上线时再分开部署。初期，我们的对弈系统可以先只用两个Node，后期如果观战人数多，压力大，可以考虑将对弈和观战分开成两个Node。

## 补充说明

1、选手

棋手的标识除了有名称，还需要有类型（AI或者Humen）

2、Room

## 初期我觉得可以不用做过多的Room管理，可以启动时就为每个地区（深圳，上海，武汉，南京，北京等）各创N个Room，Room的名称上能识别出来即可。 Game Server中的room要能保存历史对弈情况，在这个room中都有哪两个棋手下过，下了多少局，输赢情况如何（需要持久化保存），方便以后查询，不过这个优先级会低一些。 Room课堂的实现中，下完一盘就清空Room，而我们目前构建的系统中，就不能这么做了。我的想法是，比赛的room需要设定盘数，连着比很多盘，决定胜负；而游戏的room则可以一直下，直到某个棋手主动退出，大家可以继续细化或者扩充。

## 基本功能页面

初期的基本功能以能用为主，我觉得只需要“我要下棋”和“我要观战”，再配合当前room列表即可。

1、当前room列表

列出目前所有的room列表，每个room中的选手信息（昵称和类型），方便选择观战或者对弈。

2、我要观战

打开页面，选择观战后，可以显示room列表，然后选择room后，进入观战页面。

3、我要下棋

打开页面，选择下棋后，输入昵称（不能重复），显示room列表，方便选择，进入room后，进入对弈页面。例如：武汉的小伙伴将训练好的AI棋手（可以多个版本，也要起个好听、霸气的昵称名哦）通过tcp接入分配给武汉的room，然后可以昭告天下，求挑战人类。感兴趣的同学可以在room列表中选择相应房间进入，大战一把。如果两地的AI想挑战，也约定一个自己的房间进入即可。

4、扩展

后期的大赛宣传、介绍以及一些管理命令（例如考虑专门的room管理页面以及统计数据查看页面等），优先级可以低一些。初期，可以直接提供room管理的shell命令。

## 迭代开发

可以先开发“当前room列表”功能，配合服务器侧此时可以添加多room的管理；

接着可以开发“我要观战”功能，此时无需人类棋手介入，只需要两个Erlang的AI棋手就可以完成集成和调试，并且也无需修改原有的term\_to\_binary

的协议，可以集中考虑room在不同场景下对盘数的控制；以及WWW Server侧的agent实现和页面的poll机制；

等这个就绪后，可以再去修改协议，联入其他语言的AI棋手。

最后，可以开发“我要下棋”功能。

每个功能，前端和后端可以并行开发，各地同学强强联手，自由组织哦！有任何问题，都欢迎和我沟通交流。