MO Game 'Battle Snake' 서버 전체 구조

BackEnd

Matchmaking DB (Replication)

Matchmaking_status (master)
serverstatusTBL

Replication

Matchmaking_status x n (slave)
serverstatusTBL

Sharding Info DB (Replication)

shDB_Index (master)
AllocateTBL

Replication

shDB_Index x n (slave)
AllocateTBL

shDB_Info (master)
AvailableTBL
dbconnectTBL

Replication

shDB_Info x n (slave)
AvailableTBL
dbconnectTBL

Content DB (Sharding)

shDB_data_1
accountTBL
contentsTBL

shDB_data_2
accountTBL
contentsTBL

• • •

PHP API

- 계정 생성 API
- 유저 정보 갱신 / 얻기 API
- 컨텐츠 정보 갱신 / 얻기 API
- 매치메이킹 서버 정보 얻기 API

매치메이킹 서버 정보 갱신

Front

특수한 경우를 제외한 DB 접속은 모두 PHP로 제작한 API를 거친다.
- Front에서 BackEnd로 접근하기 위한 Gate 역할.
- 서버 구조 내에서, Front와 BackEnd를 명확히 구분
- Front에서는, BackEnd의 DB 구조를 전혀 몰라도, API만 호출하면 원하는 결과를 얻을 수 있다.

Server

Matchmaking Server x n
- 매치메이킹 Net 서버
- 마스터 Lan 클라이언트

Master Server
- 배틀 Lan 서버
- 매치메이킹 Lan 서버

Apache (웹 서버)

HTTP (POST) JSON

Apache를 향한 요청은 모두 POST 형태.
- **Body :** JSON

2

Battle 서버 군 x n

Battle Server
- 배틀 MMO 서버
- 채팅 Lan 서버
- 마스터 Lan 클라이언트

Chat Server
- 채팅 Net 서버
- 배틀 Lan 클라이언트

3

HTTP (POST) JSON

클라이언트 접속 절차

1. Lobby 서버에서, 접속할 매치메이킹 서버를 알아온다.
2. 매치메이킹 서버에서, 접속할 배틀서버,채팅서버와 입장할 방을 알아온다.
3. 배틀서버, 채팅서버 입장 후, 방 입장

클라이언트 Connect

HTTP (POST) JSON

Lobby Server

1

DB Sharding 분산

Content DB(Sharding)에 접근하기 위해서는, Sharding Info DB를 거쳐야 한다.

1. shDB_Index에서 ,해당 유저가 저장된 샤딩DB의 번호를 알아온다.
2. 알아온 샤딩DB의 번호를 이용해, shDB_Info에서 해당 DB로 접속하기 위한 정보를 알아온다.
3. 알아온 정보를 이용해 DB 접속.

샤딩 DB 접근 요청

Sharding Info DB (Replication)

shDB_Index (master)

AllocateTBL

유저 식별자, 샤딩 DB 번호

1

shDB_Info (master)

AvailableTBL

dbconnectTBL

샤딩 DB 별 저장 가능한 유저 수

분산 지점

새로운 계정 생성 시, AvailableTBL의 값이 가장 큰 샤딩 DB에 유저정보 저장

2

샤딩 DB 번호, 샤딩 DB 접속 정보

Content DB (Sharding)

3

shDB_data_1

accountTBL

contentsTBL

shDB_data_2

accountTBL

contentsTBL

• • •

유저의 닉네임, 전적정보 등 컨텐츠 정보

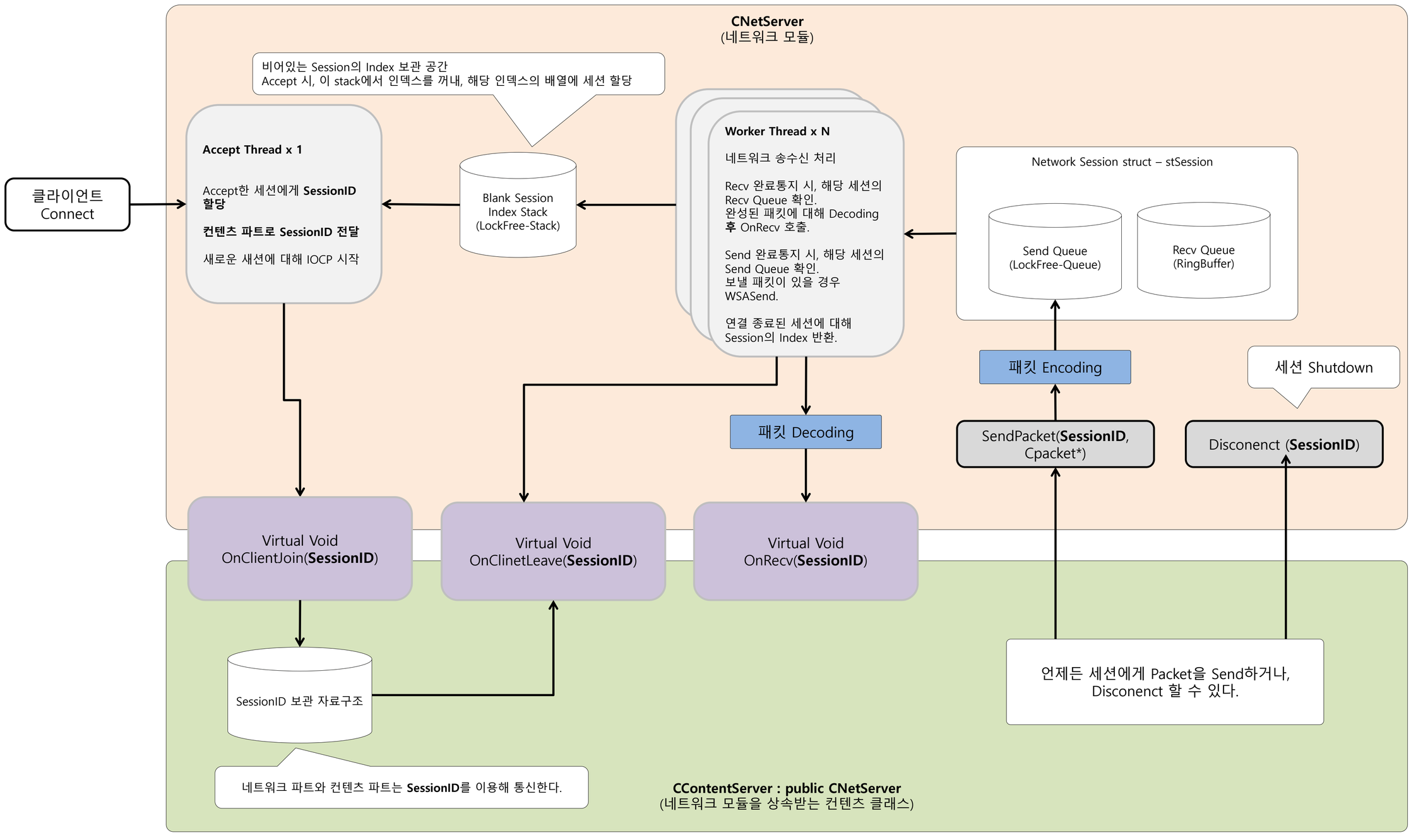
네트워크 모듈 – CNetServer / CLanServer

특징

- LockFree 구조의 네트워크 모듈
- SessionID를 이용해 네트워크와 컨텐츠간 통신

CNetServer, CLanServer의 차이점

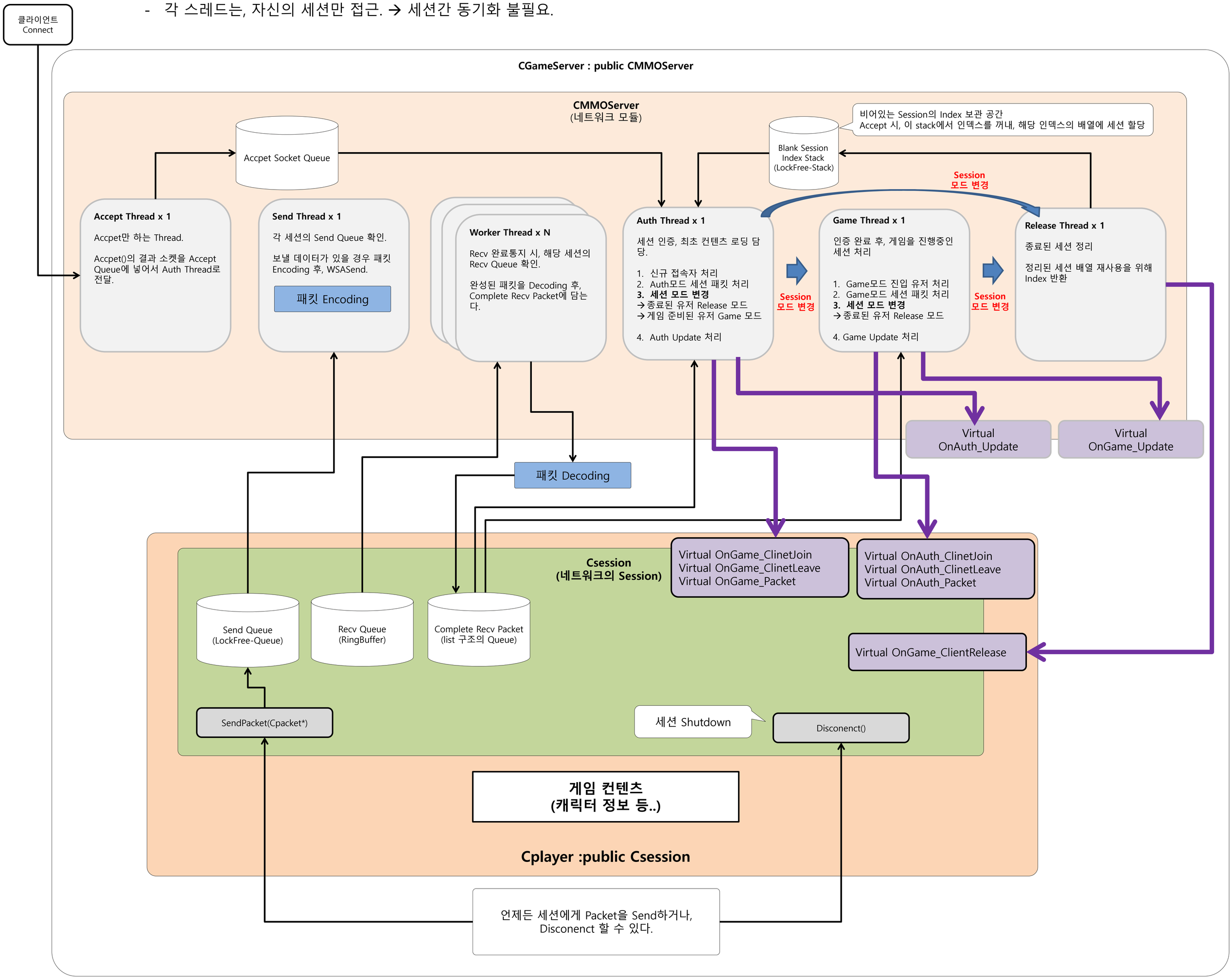
- CLanServer는 패킷 Encoding / Decoding 절차가 없다.
- 그 외에는 완전히 동일한 구조



네트워크 모듈 - MMOServer

특징

- LockFree구조의 네트워크 모듈
- 네트워크 파트의 Session과 컨텐츠의 Player가 1개로 구성 (상속 구조). 때문에, 네트워크와 컨텐츠 간의 통신 프로토콜 불필요
- 각 세션은 현재 자신의 상황에 따라 모드가 결정됨. (Auth 모드, Game 모드, Release 모드)
- 각 스레드는, 자신의 세션만 접근. → 세션간 동기화 불필요.



MO Game 'Battle Snake' 서버 전체 구조

Matchmaking DB (Replication)

Matchmaking_status (master)
- serverstatusTBL

Replication

Matchmaking_status x n (slave)
- serverstatusTBL

Sharding Info DB (Replication)

shDB_Index (master)
- AllocateTBL

Replication

shDB_Index x n (slave)
- AllocateTBL

shDB_Info (master)
- AvailableTBL
- dbconnectTBL

Replication

shDB_Info x n (slave)
- AvailableTBL
- dbconnectTBL

Content DB (Sharding)

shDB_data_1
- accountTBL
- contentsTBL

shDB_data_2
- accountTBL
- contentsTBL

● ● ●

PHP API

- 계정 생성 API
- 유저 정보 갱신 / 얻기 API
- 컨텐츠 정보 갱신 / 얻기 API
- 매치메이킹 서버 정보 얻기 API

매치메이킹 서버 정보 갱신

특수한 경우를 제외한 DB 접속은 모두 PHP로 제작한 API를 거친다.
- 서버 구조 내에서도 Front와 BackEnd를 구분
- Front에서는, BackEnd의 DB 구조를 전혀 몰라도, API만 호출하면 원하는 결과를 얻을 수 있다.

Server

Matchmaking Server x n
- 매치메이킹 Net 서버
- 마스터 Lan 클라이언트

Master Server
- 배틀 Lan 서버
- 매치메이킹 Lan 서버

Apache (웹 서버)

HTTP (POST) JSON

Apache를 향한 요청은 모두 POST 형태.
- **Body :** JSON

2

Battle 서버 군 x n

Battle Server
- 배틀 MMO 서버
- 채팅 Lan 서버
- 마스터 Lan 클라이언트

Chat Server
- 채팅 Net 서버
- 배틀 Lan 클라이언트

3

HTTP (POST) JSON

HTTP (POST) JSON

Lobby Server

1

클라이언트 접속 절차

1. Lobby 서버에서, 접속할 매치메이킹 서버를 알아온다.
2. 매치메이킹 서버에서, 접속할 배틀서버,채팅서버와 입장할 방을 알아온다.
3. 배틀서버, 채팅서버 입장 후, 방 입장

클라이언트 Connect