boost::serialization을 사용하여 객체를 serialization 한다.

객체를 직렬화 한 후,
헤더(stream의 길이, 타입) + 데이터(stream)를 보낸다.

기본적으로 서버는 클라이언트 사이를 연결해주는 역할과 게임 중인 사용자가 맞는지 체크만 할 뿐,
그 외는 클라이언트 내에서 다 해결한다. (ex) 놓을 수 없는 돌 위치, 게임 결과 집계 등등.

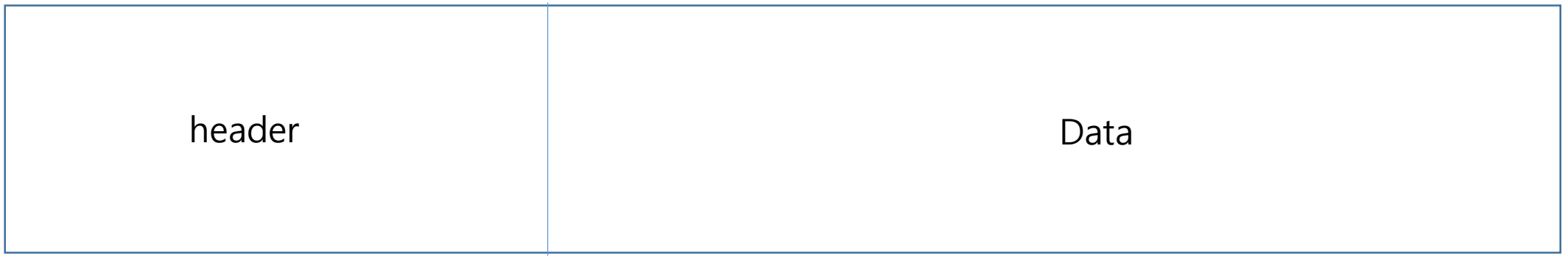
0. header

<사용되는 객체는 이렇다.>

1. 초기 화면에서 Mode 설정 시.
2. 1에 대한 서버의 response.
3. 게임 진행 정보.
4. 3에 대한 서버의 response.

```
class UserAccount
{
private:
string name;
string id;
string pw;
int age;

friend class boost::serialization::access; //직렬화해주는분께서 접근 할 수 있도록 friend선언
template<class Archive>
void serialize(Archive& ar, const unsigned int version)
{
ar& name;
ar& id;
ar& pw;
ar& age;
}
public:
UserAccount(string _name, string _id, string _pw, int _age)
{
name = _name;
id = _id;
pw = _pw;
```



Packet type
모드 선택 인가?
게임 중 인가?
Data length;

Serialization 된 객체

초기

Initial Request (Client)

[Src_IP]
My IP

[MODE]
AI Mode(0), PvP Mode(1)

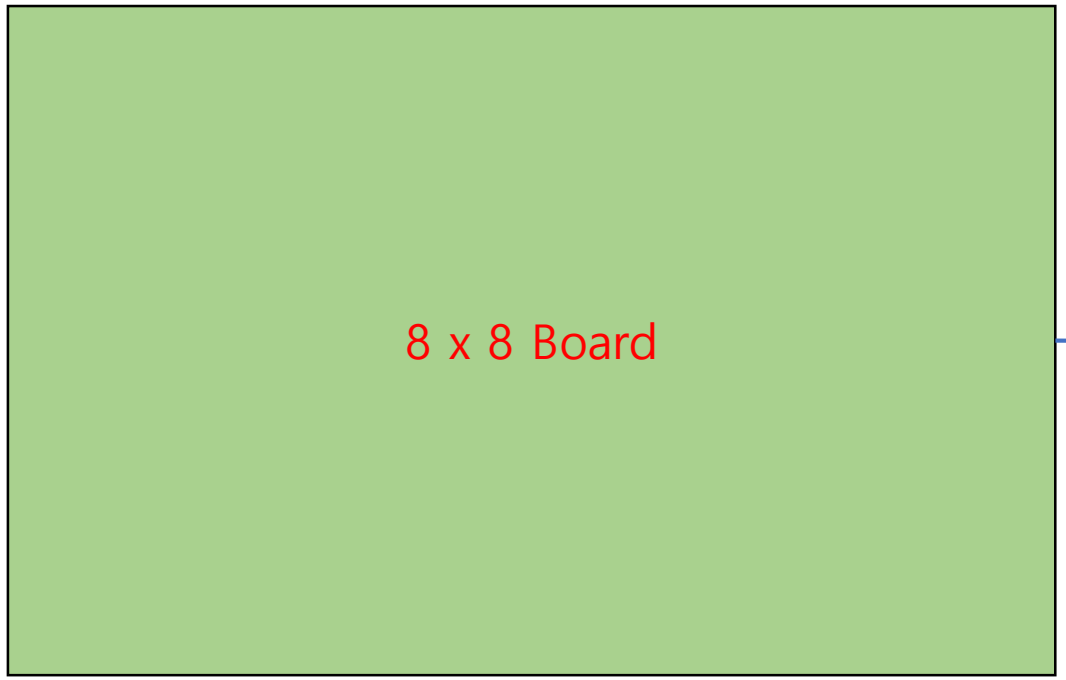
==========

Initial Response (Server)

[STONE]
WHITE(0), BLACK(1)

[STATE]
Stanby(0), Start(1), Full(2)

게임 진행 중



Game Request (Client)

[Src_IP]
My IP

[STONE]
My Stone Color

[MOVE]
돌이 놓아지면? 1
놓을 곳이 더 이상 없다면? 2

[LOCATION]
(x, y)

[RESULT]
Win(0), draw(1)

[SCORE]
집계된 스코어

==========

Game Response (Server)

[MOVE]
상대방의 돌이 놓아졌다면? 1
상대방의 돌이 놓아지지 않았다면? 2

[LOCATION]
(x, y) 상대방이 놓은 위치.

[RESULT]
Win(0), draw(1) 상대방에게 알림.

[SCORE]
집계된 스코어

[DISCONNEC]
상대방의 연결이 끊겼을경우(1)