게임서버 포트폴리오

이창하 (Com2us)

개요

소개: C++서버, 유니티 클라로 개발된 Zone 기반의 MMORPG

개발환경

1. OS: WINDOWS 11
2. TOOL: VISUAL STUDIO 2022
3. CPU: AMD RYZEN 9 7900
4. RAM: 32GB

개발스택

SERVER: C++, IOCP, PROTOBUF, MSSQL

CLIENT: UNITY, PROTOBUF

URL

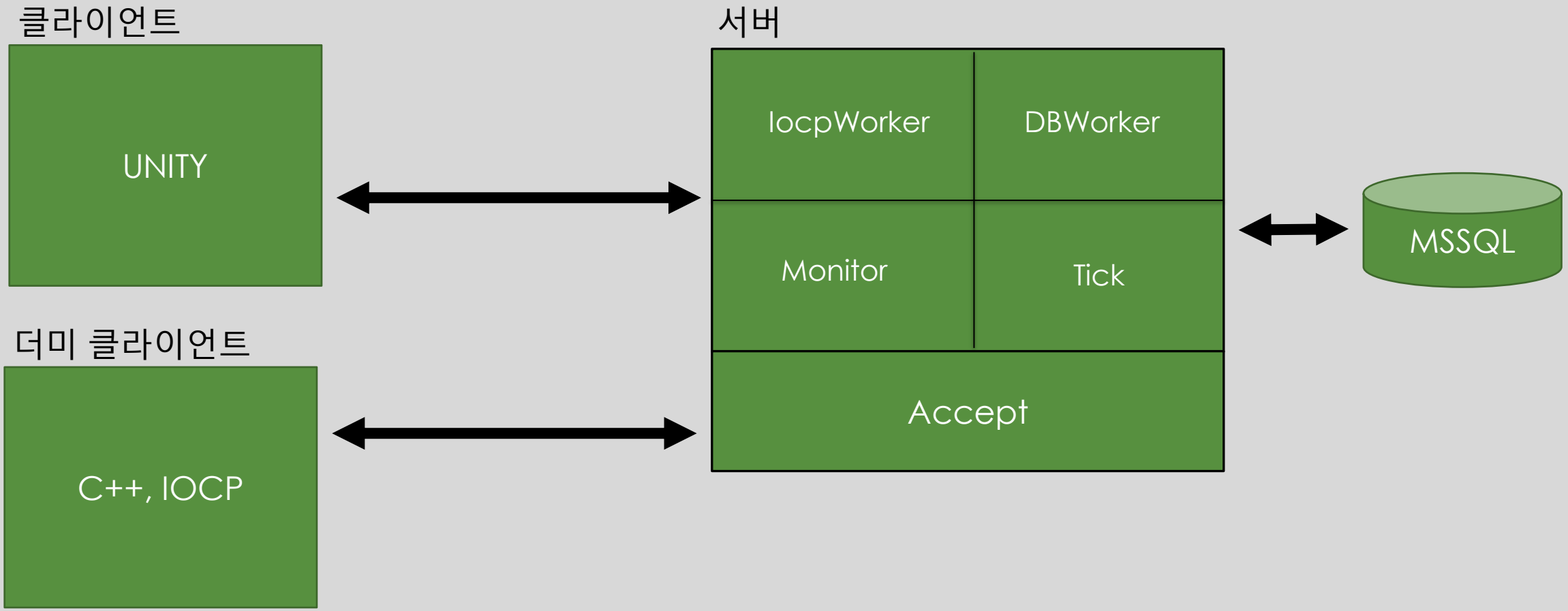
유튜브:

HTTPS://WWW.YOUTUBE.COM/WATCH?V=GUQN8-QRQLS

깃허브:

HTTPS://GITHUB.COM/CKDGKWKD27/LCHMMOPROJECT

프로젝트 구성 도식화


클라이언트
UNITY
더미 클라이언트
C++, IOCP
서버
IocpWorker
DBWorker
Monitor
Tick
Accept
MSSQL

서버 쓰레드 구성

- Accept: 새로운 접속자를 받아들이는 쓰레드입니다.
- IocpWorker: 서버의 네트워크 작업 및 DB 결과를 전달받는 클래스입니다.
- DBWorker: DB요청을 받아서 쿼리 작업을 수행하는 쓰레드입니다. 결과를 IocpWorker 쓰레드에 전달합니다.
- Monitor: 서버의 성능을 보고하는 쓰레드입니다. 주기적으로 send/recv 처리량을 출력합니다
- Tick: 주기적인 작업들을 처리하는 쓰레드입니다. 주로 Zone에서의 주기적인 작업들을 처리합니다.

Protobuf

- 서버/클라에서는 protobuf를 사용해 패킷 직렬화 사용

```
if (packet.ParseFromArray(buffer + sizeof(PacketHeader), len - sizeof(PacketHeader)) == false)
```

```
auto _player = GPlayerManager.NewPlayer();
_player->ActorInfo.set_objecttype((uint32)ObjectType::PLAYER);
_player->ActorInfo.set_actorid(GZoneManager.IssueActorID());
_player->ActorInfo.set_name("Player" + std::to_string(_player->playerId));
_player->ActorInfo.mutable_posinfo()->set_state((uint32)MoveState::IDLE);
_player->ActorInfo.mutable_posinfo()->set_movedir((uint32)MoveDir::UP);
_player->ActorInfo.mutable_posinfo()->set_posx(0);
_player->ActorInfo.mutable_posinfo()->set_posy(0);
_player->ActorInfo.mutable_statinfo()->set_level(1);
_player->ActorInfo.mutable_statinfo()->set_hp(100);
_player->ActorInfo.mutable_statinfo()->set_maxhp(100);
_player->ActorInfo.mutable_statinfo()->set_attack(5);
_player->ActorInfo.mutable_statinfo()->set_speed(5);
_player->ActorInfo.mutable_statinfo()->set_totalexp(0);
_player->zoneID = 0;
_player->ownerSession = playerSession;
_player->ownerSession->currentPlayer = _player;
```

```
message ObjectInfo {
  uint32 ObjectType = 1;
  uint32 actorId = 2;
  string name = 3;
  PositionInfo posInfo = 4;
  StatInfo statInfo = 5;
}

message PositionInfo {
  uint32 state = 1;
  uint32 moveDir = 2;
  int32 posX = 3;
  int32 posY = 4;
}

message StatInfo {
  uint32 level = 1;
  uint32 hp = 2;
  uint32 maxHp = 3;
  int32 attack = 4;
  float speed = 5;
  uint32 totalExp = 6;
}

message RequestJoin
{
    string id = 1;
    string password = 2;
}

message ReturnJoin
{
    uint32 result = 1;
}

message RequestLogin
{
    string id = 1;
    string password = 2;
}

message ReturnLogin
{
    uint32 result = 1;
    uint32 playerId = 2;
}
```

.proto 파일

패킷 처리

- 패킷을 처리하는 함수는 요청에 맞는 작업(db요청, Zone 작업) 을 알맞은 대상에게 위탁.
- Zone 기반 작업은 싱글쓰레드인 MessageQueue에 Push 하는 방식. Tick 쓰레드에서 주기적으로 Flush 하며 Zone 작업을 수행
- Multiple Producer Single Consumer

```
_zone->messageQueue.Push([_zone, _player]() {_zone->EnterGame(_player, _zone->zoneID); });
```

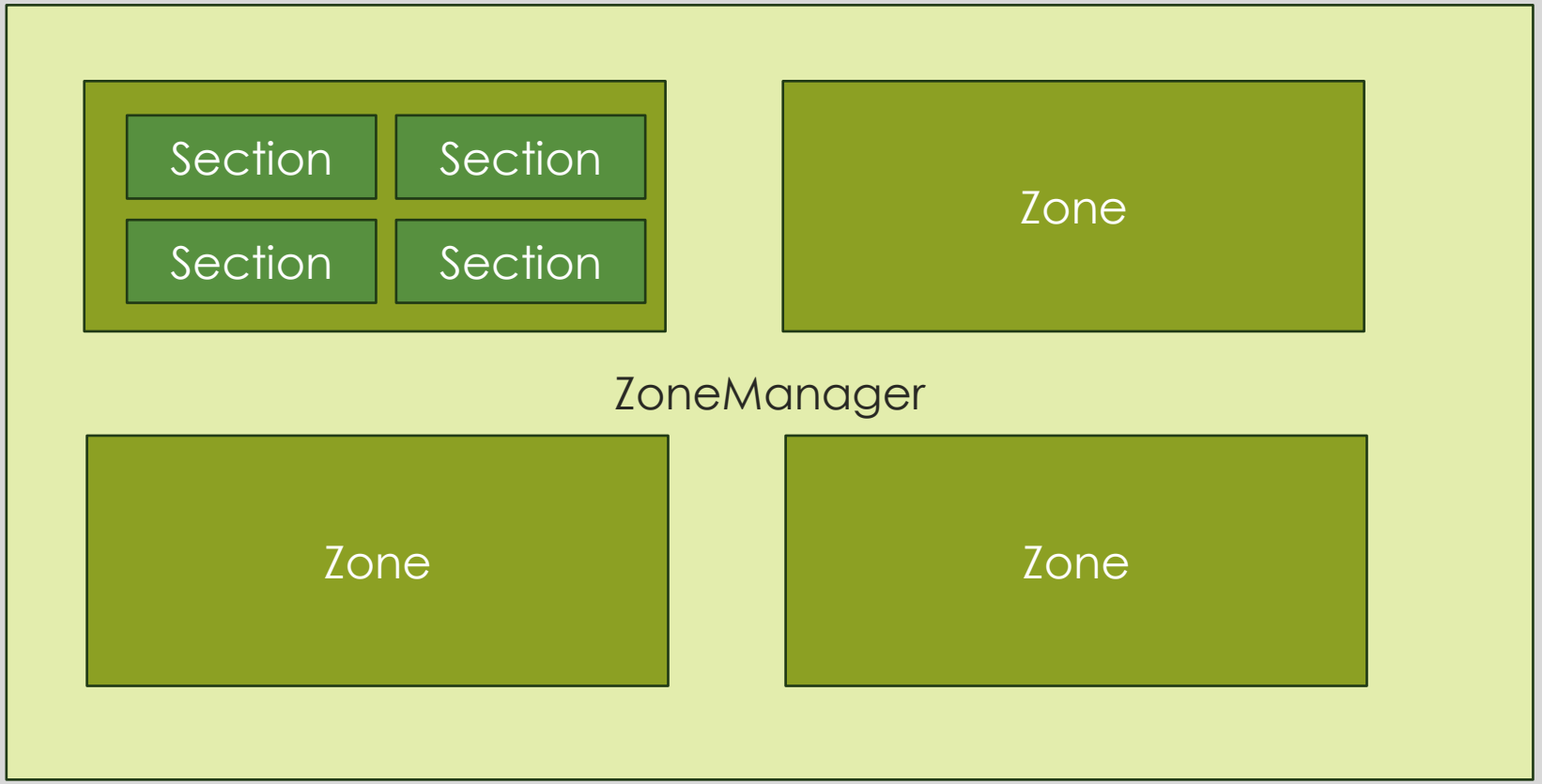
```
void MessageQueue::FlushAll(std::vector<MessageFuncType>& messages)
{
    LockGuard guard(messageLock);

    while (_queue.empty() == false)
    {
        auto _message = _queue.front();
        messages.push_back(_message);
        _queue.pop();
    }
}
```

2024-05-20

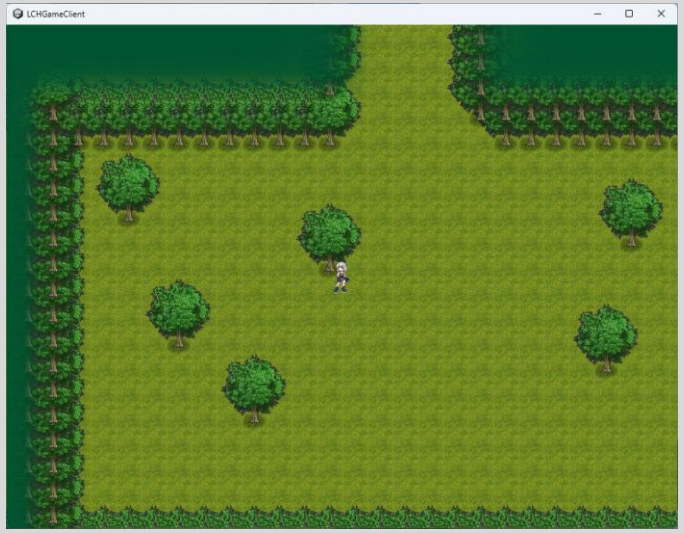
월드 구성

- 월드는 Zone 단위로 구성되며 Zone은 Section (10x10 사이즈) 으로 나뉘어짐
- Broadcast는 Section 단위로 전송

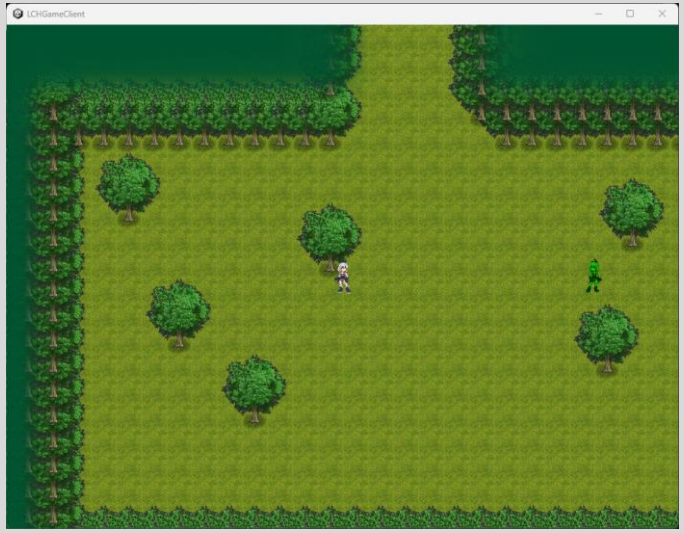


플레이어 시야(Viewport)

- 패킷전송을 최소화 시키기 위해 플레이어는 시야를 보유
- 시야 내부에 들어오거나 밖으로 나간 액터들을 주기적으로 계산하여(0.2초) 클라에게 화면에 렌더링 할 대상들을 전송
- 시야를 업데이트 할때 주변의 Section들을 검사해서 판별



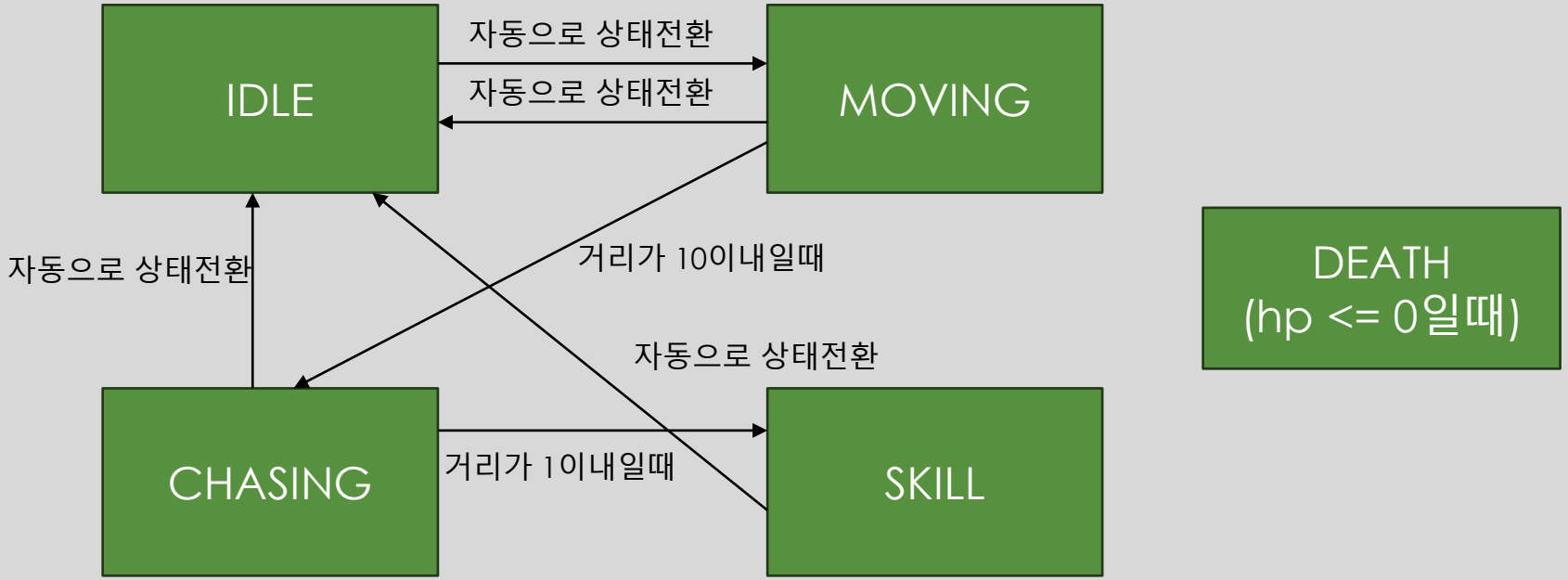
Viewport Size = 1



Viewport Size = 10

몬스터 AI

◦ 몬스터는 IDLE, MOVING, CHASING, SKILL, DEATH 상태를 갖고 있습니다



데드 레커닝 (클라이언트)

- 클라이언트 측에서는 서버 부하감소와 게임 반응성 향상을 위해 데드레커닝을 사용

```
protected override void Update()
{
    base.Update();

    bool ForceSendPacket = false;
    if(LastDesiredInput ≠ DesiredInput)
    {
        ForceSendPacket = true;
        LastDesiredInput = DesiredInput;
    }

    if(DesiredInput == Vector2.zero)
    {
        if(_moveUpdateReserved == false)
            PosInfo.State = (uint)CreatureState.IDLE;
    }
    else
    {
        PosInfo.State = (uint)CreatureState.MOVING;
    }

    MovePacketSendTimer -= Time.deltaTime;

    if (MovePacketSendTimer ≤ 0 || ForceSendPacket)
    {
        MovePacketSendTimer = MOVE_PACKET_SEND_DELAY;
        Protocol.RequestMove movePacket = new Protocol.RequestMove();
        movePacket.PosInfo = PosInfo;
        Managers.Network.Send(movePacket);
    }
}
```

MyPlayer

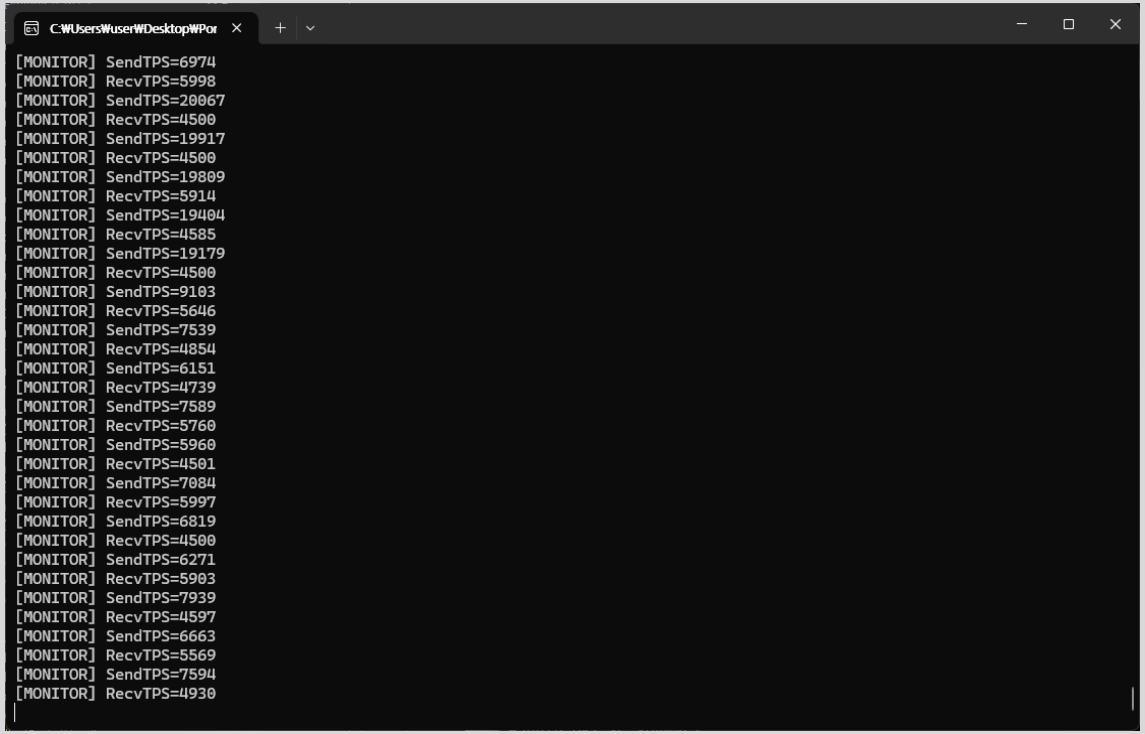
```
Unity Message | 4 references
protected override void Update()
{
    base.Update();

    if (State == CreatureState.MOVING && IsMyPlayer() == false)
    {
        Vector3 direction = Util.GetVecFromDir((MoveDirType)destination.MoveDir);
        transform.position += direction * Stat.Speed * Time.deltaTime;
    }
}
```

OtherPlayer

Dummy Client

- 서버의 성능을 평가하기 위한 쓰레드. 매 초마다 Send/Recv 한 패킷수를 보고
- 더미 클라이언트와 함께 사용되어 서버의 처리량을 측정



1500명 접속, 0.2초당 이동패킷을 보냈을때 패킷처리량

감사합니다