EdenNetwork Documentation

목차

EdenNetwork

EdenNetwork는 TCP 소켓으로 이루어진 게임개발용 네트워킹 라이브러리입니다. 이 절에서는 EdenNetwork의 구조와 간단한 사용법을 소개하겠습니다.

1. EdenNetwork 소개

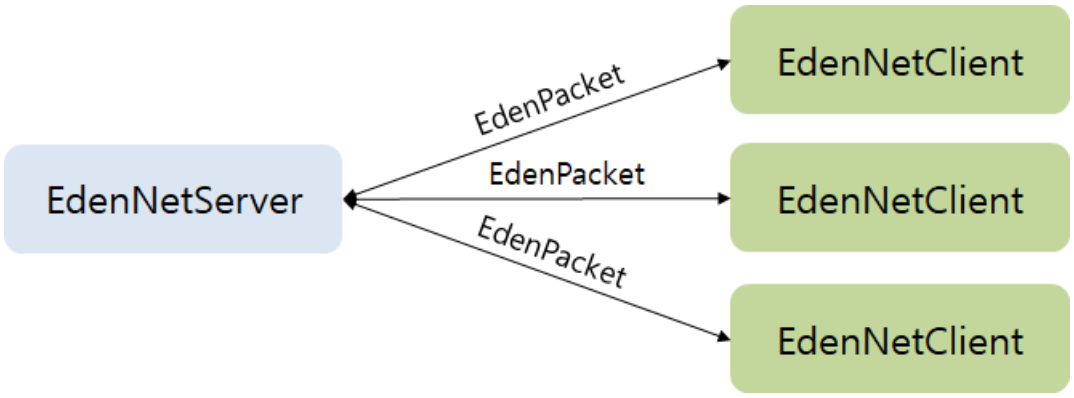


그림 1. EdenNetwork의 구조

EdenNetwork는 EdenNetServer와 EdenNetClient 두 개의 클래스가 서버와 클라이언트의 역할을 하며 1:N으로 통신합니다. 서버와 클라이언트가 모두 송수신을 할 수 있으며 송신은 Send와 Request 두가지 방법을 통해 이루어지고 수신 또한 Receive와 Response 두가지 방법을 통해 이루어집니다. 모든 데이터 교환은 EdenPacket이라는 형식으로 이루어집니다. 데이터가 TCP소켓을 통해 전송될 때는 c#데이터구조를 Json 형식으로 직렬화합니다.

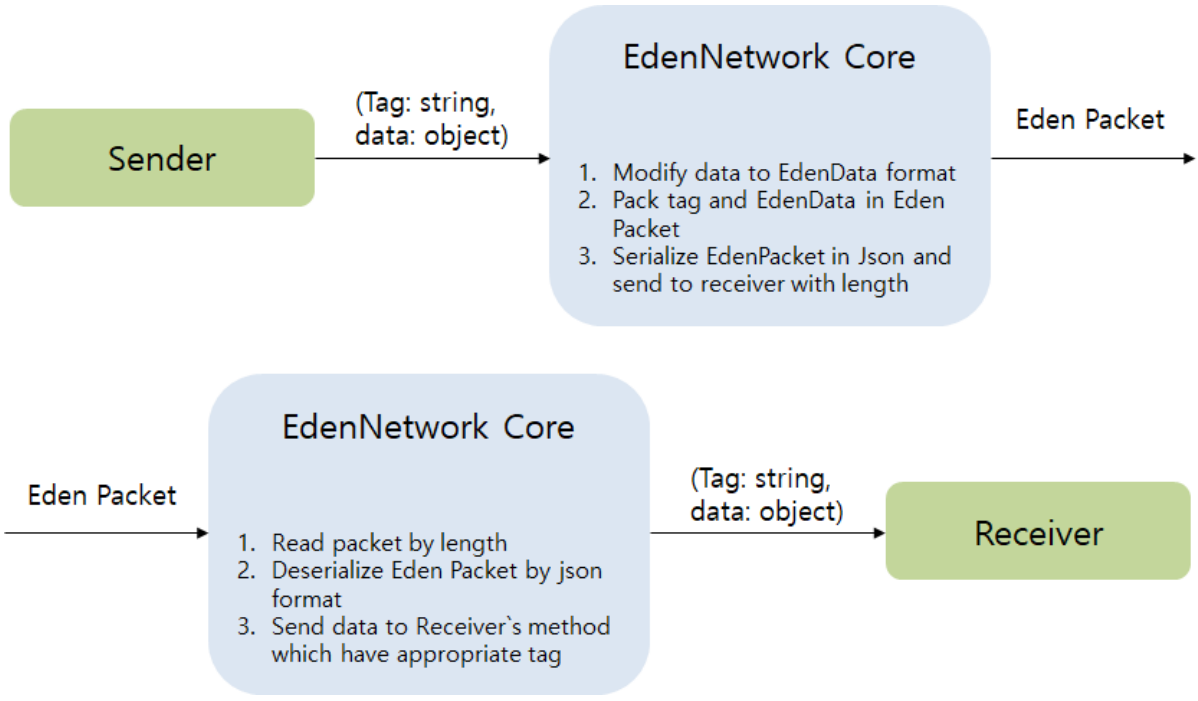


그림2. EdenNetwork의 데이터 송수신 방식

송신자가 태그와 데이터를 보내면 EdenNetwork에서 하나의 패킷으로 만들어 수신자에게 보내게 되고 수신자는 같은 이름의 태그로 메소드를 등록해 해당 메소드를 통해 데이터를 수신하고 처리할 수 있습니다.

2. 통신 시퀀스

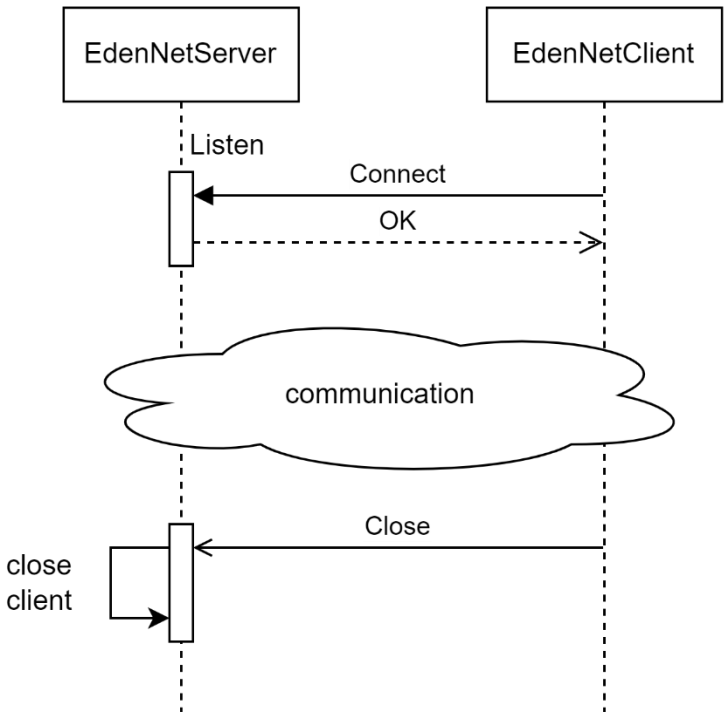


그림 3. EdenNetwork 통신 시퀀스

통신절차는 서버가 Listen을 시작한 후 클라이언트가 Connect로 서버에 접근합니다. 서버가 커넥트에 대해 OK를 반환하면 서버와 클라이언트간 통신을 수행한 후 클라이언트에서 Close를 보내면 서버는 해당 클라이언트를 닫게 됩니다.

2.1. Listen & Connect

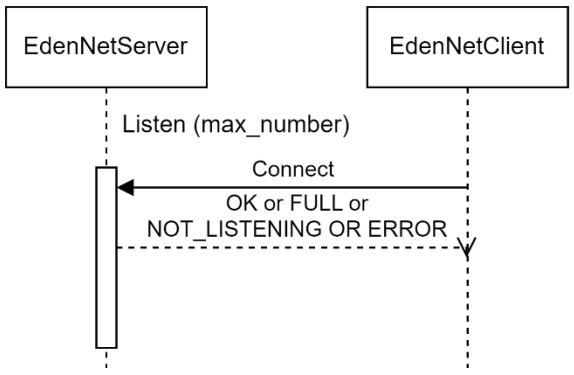


그림4. EdenNetwork Listen & Connect 통신 시퀀스

서버는 Listen을 수행할 때 최대 몇 개까지의 클라이언트의 접속을 허용할 건지 max_number를 결정하며 시작합니다. Listen중인 서버에 클라이언트는 Connect를 통해 접속할 수 있습니다. Connect의 결과로 ConnectState를 반환합니다. ConnectState는 enum형으로 OK, FULL, NOT_LISTENING, ERROR가 존재합니다. OK를 반환받으면 서버접속에 성공한 것입니다. FULL을 반환받으면 서버에 접속된 클라이언트 수가 이미 최대이기에 더 이상 클라이언트가 접속할 수 없다는 의미입니다. NOT_LISTENING은 현재 서

버가 Listen중이 아니라는 의미입니다. 그 외 서버가 Listen중인데 연결에 실패할 경우 ERROR를 반환합니다.

2.2. Send & Receive

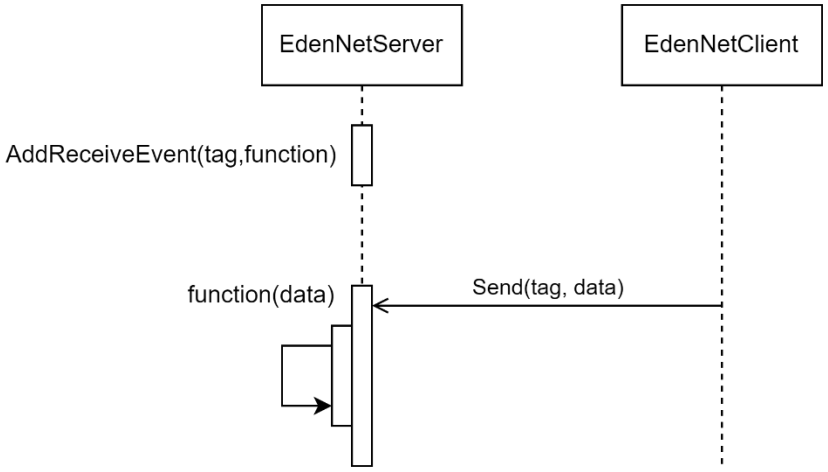


그림5. EdenNetwork Send&Receive 통신 시퀀스

서버에 접속을 성공한 후 서버와 클라이언트는 서로 데이터를 주고 받을 수 있습니다. Send 메소드를 통해 데이터를 보낼 경우 tag와 함께 데이터를 보냅니다. 서버에서는 AddReceiveEvent 메소드를 통해 미리 tag를 처리할 콜백메소드를 등록합니다. 서버에서는 tag가 붙은 데이터가 소켓을 통해 들어오면 해당 tag에 매핑되어 있는 메소드를 실행하여 데이터를 처리합니다. Send는 클라이언트에서 서버로 보낼 수 있으며 서버 또한 클라이언트에게 Send를 통해 데이터를 전송할 수 있습니다.

2.3. Request & Response

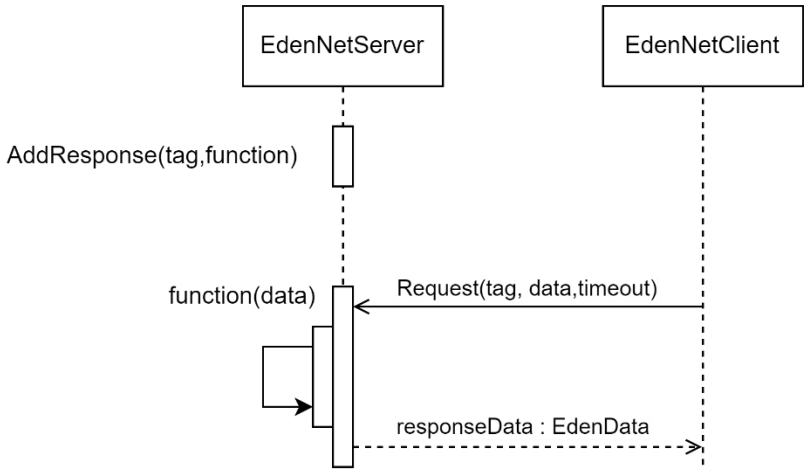
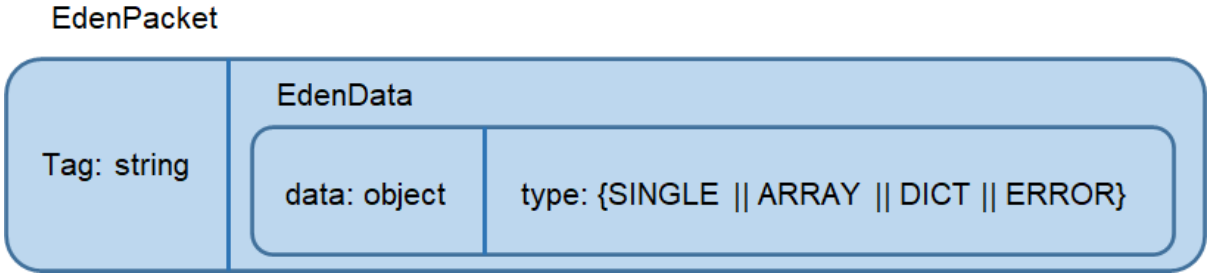


그림6. EdenNetwork Request&Response 통신 시퀀스

Request 메소드를 통해 클라이언트가 서버에 데이터를 보낼 경우도 Send와 마찬가지로 tag와 함께 데이터

를 보냅니다. Request는 서버에서 데이터를 반환하기 때문에 서버의 응답을 기다리는 시간인 timeout을 설정합니다. Request를 요청 후 서버에서 데이터를 반환할 때까지 프로세스를 블락하고 만일 timeout내에 서버가 응답을 돌려주지 않는다면 Error를 반환하며 프로세스 블락을 해제합니다. 서버에서는 AddResponse를 통해 tag를 처리할 콜백메소드를 등록합니다. 이 콜백메소드는 항상 클라이언트에게 응답할 EdenData타입의 responseData를 반환해야합니다.

3. 데이터 접근 방법



EdenNetwork에서는 EdenPacket이라는 클래스를 통해 데이터를 송수신합니다. EdenPacket은 tag부와 data부로 나뉩니다. Tag는 데이터를 구분하는 string형의 고유한 구분자이고 EdenData는 사용자가 통신할 데이터를 패킹한 클래스입니다. 송신자와 수신자는 tag를 공유하여 특정 tag의 데이터를 어떻게 처리해야 할지 판단할 수 있습니다.

EdenData는 SINGLE, ARRAY, DICT 3가지 형식을 가지고 있습니다. 하나의 데이터를 전송할 때는 SINGLE을 이용하고 두 개 이상의 구분된 데이터를 전송할 때는 ARRAY나 DICT를 이용합니다. ARRAY의 경우 인덱스를 통해 데이터에 접근할 수 있고 DICT의 경우 string 형식의 키를 통해 데이터에 접근할 수 있습니다.

사용자가 object형의 data를 전송하면 EdenNetwork에서는 data를 EdenData형식으로 패킹하여 전송합니다. 데이터를 수신할 경우 EdenData에서는는 Get메소드를 통해 데이터를 접근할 수 있습니다. SINGLE, ARRAY, DICTIONARY 타입에 따라 데이터를 접근합니다. 각 타입에 맡게 Get함수를 호출하지 않으면 예외를 반환합니다. 예를 들어 ARRAY타입의 데이터인데 Get()함수를 호출하게 되면 SINGLE 타입이 아니라는 예외를 반환합니다.

SINGLE형식의 데이터를 접근할 시는 인자 없이 접근할 수 있습니다. ARRAY형식의 데이터를 접근할 시 ARRAY의 인덱스를 통해 원하는 데이터를 접근할 수 있습니다. DICT형식의 데이터에 접근할 시 string형의 key값으로 접근합니다.

4. 메소드 소개

4.1. 네트워크 메소드

네트워크 메소드에서 모든 송수신 메소드는 서버와 클라이언트가 기능적으로는 동일합니다. 하지만 서버에서는 어떤 클라이언트에게 데이터를 전송해야 하는지 구분해야 하기 때문에 추가로 clientId값을 인자로 받습니다. clientId는 접속한 클라이언트의 IP주소와 포트번호로 이루어져 있습니다. 서버는 clientId를 저장하고 데이터 전송 시 clientId를 인자로 주어 어떤 클라이언트에게 데이터를 전송할 지 결정합니다.

4.1.1 Listen & Connect

Listen(ServerSide)

void Listen(int maxAcceptNum, Action<string> callback)	
maxAcceptNum	*최대 수신할 클라이언트의 수*
Callback	*클라이언트가 접속 할 시 수행할 콜백메소드*

Callback의 인자인 string은 클라이언트의 식별자(clientId)입니다.

Connect(ClientSide)

Connect를 호출하면 서버에 연결을 시도합니다. 연결이 성공할 시에만 OK를 반환합니다.

```
enum ConnectionState
{
    OK, FULL, NOT_LISTENING, ERROR
}
```

OK	서버에 연결된 상태
FULL	서버에 연결된 인원이 초과하여 연결할 수 없는 상태
NOT_LISTENING	서버가 LISTEN하고 있지 않은 상태
ERROR	그 외 모든 사유로 연결이 실패한 상태

ConnectionState Connect(string IPAddress, int port)	
IPAddress	*연결할 서버의 ip주소*
port	*연결할 서버의 port번호*

Task<ConnectionState> ConnectAsync(string IPAddress, int port)	
IPAddress	*연결할 서버의 ip주소*
port	*연결할 서버의 port번호*

비동기 메소드

ConnectionState BeginConnect(string IPAddress, int port, Action<ConnectionState> callback)	
IPAddress	*연결할 서버의 ip주소*
port	*연결할 서버의 port번호*
Callback	*연결이 끝난 후 수행할 메소드*

콜백 메소드를 통한 비동기 메소드

4.1.2 Send

가장 기본적인 메소드인 Send 메소드입니다. Send가 성공할 경우 true를 반환하고 실패할 경우 false를 반환합니다.

bool Send(string tag, EdenData data)	
tag	*송신할 패킷의 이름*

data	*전송할 데이터*

이 메소드는 송신이 완료될 때까지 프로세스를 블락합니다.

async Task<bool> SendAsync(string tag, EdenData data)	
tag	*송신할 패킷의 이름*
data	*전송할 데이터*

비동기적으로 데이터를 송신합니다.

void BeginSend(string tag, Action<bool> callback, EdenData data)	
tag	*송신할 패킷의 이름*
callback	*송신완료 시 실행할 메소드*
data	*전송할 데이터*

SendAsync와 마찬가지로 비동기적으로 데이터를 송신합니다. 이 메소드는 callback 매개변수로 전달한 메소드를 Send가 끝났을 때 실행합니다.

4.1.3 Request

Request는 송수신이 동시에 일어나는 메소드입니다. Request로 데이터를 요청하면 서버에서 응답을 돌려줄 때까지 Request 메소드에서 대기합니다. Timeout을 설정해 해당 시간 동안 응답이 돌아오지 않는다면 대기를 멈추고 오류를 반환합니다.

EdenData Request(string tag, int timeout, EdenData data)	
tag	*송신할 패킷의 이름*
timeout	*요청을 대기할 시간*
data	*전송할 데이터*

이 메소드는 응답이 돌아올 때 까지 프로세스를 블락합니다.

async Task<EdenData> RequestAsync(string tag, int timeout, EdenData data)	
tag	*송신할 패킷의 이름*
timeout	*요청을 대기할 시간*
data	*전송할 데이터*

비동기적으로 데이터를 송신합니다.

void BeginRequest(string tag, int timeout, Action<EdenData> response, EdenData data)	
tag	*송신할 패킷의 이름*
timeout	*요청을 대기할 시간*
response	*응답이 돌아왔을 때 실행할 메소드*
data	*전송할 데이터*

RequestAsync와 마찬가지로 비동기적으로 데이터를 송신합니다. 이 메소드는 response 매개변수로 전달한 메소드를 응답이 돌아왔을 때 실행합니다.

4.1.4 Receive

Receive는 Send의 수신을 전담하는 메소드입니다. AddReceiveEvent 메소드를 통해 등록한 콜백 메소드가 해당 태그의 패킷을 수신할 시 수행됩니다.

void AddReceiveEvent(string tag, Action〈EdenData〉 receive_event)	
tag	*송신할 패킷의 이름*
receive_event	*데이터 수신 시 수행할 메소드*

receive_event의 매개변수인 EdenData는 송신자가 보낸 데이터입니다.

4.1.5 Response

Response는 Request의 수신을 전담하는 메소드입니다. AddResponse 메소드를 통해 등록한 콜백 메소드가 해당 태크의 패킷을 수신할 시 수행됩니다. Receive와는 다르게 등록한 콜백메소드는 항상 EdenData형의 반환값을 가지고 있어야 합니다.

void AddResponset(string tag, Func〈EdenData, EdemData〉 response)	
tag	*송신할 패킷의 이름*
response	*데이터 수신 시 수행할 메소드*

4.2. 데이터 처리 메소드

EdenData는 Get이라는 함수를 통해 데이터를 접근할 수 있습니다. SINGLE, ARRAY, DICTIONARY 타입에 따라 데이터를 접근합니다. 각 타입에 맡게 Get함수를 호출하지 않으면 예외를 반환합니다. 예를 들어 ARRAY타입의 데이터인데 Get()함수를 호출하게 되면 SINGLE 타입이 아니라는 예외를 반환합니다.

4.2.1 Get-SINGLE

T Get〈T〉()

SINGLE형식의 데이터를 반환합니다.

4.2.2 Get-ARRAY

T Get〈T〉(int idx)	
idx	*접근할 데이터의 ARRAY 인덱스*

ARRAY형식의 데이터를 접근할 시 ARRAY의 인덱스를 통해 원하는 데이터를 접근할 수 있습니다.

4.2.3 Get-DICTIONARY

T Get〈T〉(string key)	
key	*사전에서 접근할 데이터의 key*

DICT형식의 데이터에 접근할 시 key값으로 접근합니다.

4.2.4 Get-ERROR

string GetErrorMessage()

ERROR형식의 데이터일 경우 에러 메시지를 반환합니다.

5. 유니티 클라이언트

유니티에서 사용할 수 있는 EdenNetwork 패키지입니다. 유니티에서는 클라이언트 기능만 사용할 수 있습니다. EdenNetManager가 EdenNetClient와 동등한 기능을 합니다. 몇가지 다른 점은 EdenNetManager는 싱글톤 클래스로 존재하여 EdenNetManager.Instance로 접근하여야 하고 receive

이벤트가 TCP 소켓을 통해 데이터를 수신하는 즉시 수행되는 것이 아닌 유니티 메인스레드의 다음 프레임에 수행된다는 것입니다. 그 이유는 유니티 메소드를 유니티 메인스레드 밖에서 사용할 수 없기 때문에 receive이벤트의 수행을 유니티 메인스레드와 동기화하는 것입니다.