지원하는 기능:

1. Relialbe / UnReliabe send 기능

```
//! 1. UnReliable Send
    CRudpNetManager::GetInstance()->SendPacket(addr1 , &packet , false);

 //! 2. Reliable Send
   CRudpNetManager::GetInstance()->SendPacket(addr1 , &packet , true);
```

Reliable : 패킷 순서가 맞고, 패킷 누실이 없음

2. Udp hole punching 지원 (StartUHP함수만 호출하면, 이벤트로 성공 여부 옴)

```
//! UHP 시작
    if( !CRudpNetManager::GetInstance()->StartUHP(g_vAddr) )
    {
            cout << "UHP Start 실패" << endl;
    }
```

3. 압축 지원 (zlib 사용)

대략 1048byte는 48byte정도로 압축됨.

(1048 문자열을 다 사용한 것은 아니고, 일부 문자열(한 20byte)만 사용해서 압축률이 좋았던 것임)

(실제로 1048byte에 데이타를 다 기록했다면, 압축율은 훨씬 덜어졌을 것입니다.)

4. 암호화 지원 (Seed 사용)

암호화는 엄청 빠르고, 복호화는 대략 0.1ms 걸림

- 대표 클래스 설명

CRUDPNetManager:: 외부에서 사용할 간단한 인터페이스를 가진 RUDP 라이브러리 퍼사드 클래스

RUDPNet: 실제 모든 Reliabel등 udp 네트워크 핵심 기능을 담당함

RUDPResender : 제전송 스레드

RUDPDeliver : recv한 데이타를 처리하는 패킷 처리 스레드

UHPSender: UHP (Udp hole punching)시 임시로 생성되어 UHP을 수행하는 스레드

RUDPSession : IP와 port로 상대방을 구별하며, 그 상대방을 추상화한 클래스

RUDPZlib : Zlib 클래스 인터페이스를 수정하는 adapter 클래스

Seed : Seed 알고리즘 클래스

- 패킷 처리하기

```
#include "RUDPListener.h"

class MyListener : public IRUDPListener
{
public:
 void OnReceive(const RUDPAddr& addr , const RUDPPacket* pPacket)
 {
    ..... 패킷 처리하기
 }
};
```

IRUDPListener을 상속받아서 가상함수를 재정의하면 됨

- 사용되는 대표 알고리즘

무한 재전송 알고리즘 (최적화되어 있어 , cpu 부하를 많이 먹지 않음 , 또한 재전송 개수를 상수 한 개만 전송하기 때문에 재전송 I/O 수는 항상 1)

세션 개념 적용 (많은 다수와 p2p를 할 수 있음)

그외 잡다구리들...기억안남 OTL

- 모듈 test하기

testRUDP 프로젝트로 테스트하기

1. 자기자신한테 쏘기 , 남한테 쏘기 구현되어 있음 (두번 실행해서 주소를 입력해서 서로 패킷을 전송하면 됨)

- UHP test하기

test 폴더 안에

UHPServer , UHPClient 프로젝트 오픈하고 ,

UHPServer는 한개만 필요 , UHPClient 두 개 필요

1. 서로 실행한후 각각 포트 정보와 서버 정보등을 입력한다.

2. 두 개의 UHPClient 주소를 UHPServer에 등록 - 각 UHPClient에서 'r'을 누르면 됨

3. UHPServer에서 주소 교환(Exchange) - 서버에서 'e' 누름

4. 아무 UHPClient에서 UHPStart 시작

5. UHP 성공 메세지가 뜨면, Hello으로 서로 패킷을 보낼 수 있음

참고로 UHP는 실제 공유기 환경에서 테스트 해보지않았습니다. 예전에 UHP로 게임을 만들면서,

정확한 시퀀스가 따르면 UHP가 성공했기 때문에, 그 가정을 하고 테스트 프로젝트를 만든 것입니다.

한번쯤은 꼭 테스트 하시기 바랍니다. ^^* (될 것입니다..^^;;;)

- 폴더 구조

exec : 필요한 lib이나 dll등 모듈이 들어가는 폴더 - RUDP라이브러리 프로젝트를 빌드하면 결과물이 여기에 들어감

include : 외부에서 라이브러리 참조할때 필요한 해더들을 모은 폴더

test : 모듈을 테스트하기 위한 프로젝트들이 있는 폴더

workspace: 실제 라이브러리 프로젝트들이 들어있는 폴더

후기 :

예전에 소켓 두 개 (send을 담당하는 소켓 , recv를 담당하는 소켓)로 p2p 라이브러리를 만든 적이 있습니다.

각 역할을 담당하는 소켓이 있으면 성능이 쥰을 줄 알았습니다. (병렬 처리를 기대)

하지만 성능이나 , 기타 확장성, 유지보수 면에서 사실 그리 쥰지 못했습니다.

ㅜㅜ UHP구현할때도 소켓이 두 개라서 참으로 힘들었죠..ㅜㅜ

그래서 이번에 소켓 하나로 만들었습니다. 확장성 유지보수성은 말할 것 없고, 성능적인 면도 훨씬 쥰았습니다.

또한 CPU 부하와 코드 수행 효율 쥰게 하기 위해 비동기 I/O로 개편했습니다.(전에는 WSAEventSelect모델 , 이 모듈은 WSAOverlapped I/.O모듈)

이론적으로는 빠르게 비동기로 함수가 리턴되고, 더 많은 일을 하기 때문에 효율적이고,

동기 함수보다 비동기함수가 cpu 부하를 덜 먹기 때문에..(동기함수는 내부수행때 유저모드에서 커널모드 변환시 많이 일어나기 때문에, 보안레벨땜시.. cpu가 하는 일이 훨씬 많음) cpu 부하도 클라이언트에서 여유롭게 쓸 수 있을 만큼 된다고 봅니다.

또한 가장 중요한 reliable 구현시 재전송 주기나 재전송 I/O 개수를 상수로 최적화 했기때문에 가장 큰 부하인 I/O에서도 성능이 쥰을 꺼라고 생각합니다..^^

test 프로젝트 이용하면서 디버그 포인터로 잡으시면서 분석하시면 훨씬 빠르게 분석하실 수 있을 것입니다..^^

그럼~~