**SendQ 데이터가 WSASend되지 않는 문제**

서버 - IOCP의 GQCS로 대기하는 WorkerThread가 10개.

더미클라 - Disconnect Delay : 0 Sec, ClientCount : 50, LoopDelay : 0 m/s

**이 네트워크 라이브러리에서는 SendFlag를 두고 Session당 Send를 1회로 제한한다.**

**멀티스레딩 환경이므로 Interlocked계열을 사용합니다.**

**1.**

**SendPacket() -> SendPost() -> WSASend 순으로 호출된다.**

**(SendQ에 데이터저장) (Send하기위한 작업) (WSASend호출)**

**2.**

**GQCS - Send완료통지 -> SendQ에 남은데이터가 있다면 SendPost() 재호출**

이떄 Send를 1회로 제한하기위해, SendPost() 내부로 진입하면 => SendFlag = TRUE.

Send작업이 끝나고 GQCS로 완료통지가 오면 => SendFlag = FALSE.

**1. Empty()를 체크하고, SendFlag를 체크하는 경우**

if (InterlockedExchange8(&pSession->SendFlag, TRUE) == TRUE)

return;

SendPost에는 위와같은 구문이 들어가 Send를 1회로 제한한다.

**그런데 이 체크문이 SendPost 아래나 중간쪽이 있으면, 재차 Interlocked계열 함수가 호출되고 이는 성능저하로 이어지므로 최상단에서 한번확인하고, 이후에는 확인하지 않도록 했다.**

**이 경우 다음과같은 문제가 난다.**

[스레드A] SendFlag를 TRUE로 바꾸고 Send진행 전.

[스레드B] SendPacket()이 호출되어 SendQ.Enqueue() 다음 Empty()구문을 통과. SendPost직전.

[스레드A] SendQ의 모든 데이터를 뽑아 Send가 진행되고, 완료통지까지 받아 SendFlag는 FALSE

[스레드B] SendFlag는 FALSE고, Empty()문도 통과했으니 이제 SendPost하러 들어감. 분명히 SendQ에 데이터를 넣었는데 아무것도 없어서 0을 보내는 상황.

**2. OutOfOrdering로 인해, 로직의 순서가 뒤 바뀌는 경우**

SendFlag는 완료통지가 왔을 때 FALSE로만 바뀌고, SendPost에서는 TRUE로만 바꿈 (단방향)

**방향성이 있으므로 FALSE로 바뀔 때는 Interlocked계열이 아닌 단순대입으로 로직을 진행**

**SendPost() 최상단에서 Empty()로 예외구문을 두었기 때문에, SendQSize가 0인 상태로 데이터를 보내는 것은 있을 수 없다고 생각함.**

**하지만 예상과는 다르게 OutOfOrdering으로 이 두 구문의 순서가 뒤바뀌는 경우가 존재!**

**이 경우 Send 0을 하는 상황이 발생**

[스레드A] SendQSize = 10 일 때, Empty()가 FALSE임. (SendFlag는 나중에 바꿈)

[스레드B] SendPacket()이 호출되어 SendQ.Enqueue()이후 모든 데이터를 Send.

[스레드B] 완료통지까지 와서 SendFlag를 FALSE로 대입

[스레드A] 스레드 A역시 SendFlag를 FALSE로 바꾸고 SendPost를 시도한다. SendQSize는 0이지만, Empty()구문이 미리 실행되었기 때문에 예외처리 되지않고 SendPost구문을 통과. SendQ에 데이터가없는데 WSASend를 하는상황

**또한 이와 같은 상황에서 다른 문제로, Empty()인 것을 확인하고 SendFlag를 FALSE로 바꾸려고 했는데, 다른 스레드에서는 Enqueue()했음에도 SendFlag 때문에 Send하지 못한다면 어느 곳 에서도 Send하지 못하므로 이는 SendQ에 데이터가 적재.**

만약 다시 SendPacket()요청이 온다면 보내지지 않은 데이터가 절대 송신되지않음..

**따라서 Memory-Barrier함수(mfence())를 사용하거나, 일반대입이 아닌 Interlocked()함수를 짝을 맞춰사용함으로 CPU의 OutOfOrdering을 막아야 한다.**

**3. LockFreeQ에 분명 데이터를 넣었는데 Dequeue를 실패하는 경우**

락프리큐에 분명 Enqueue하여 UseSize도 올라갔는데, Send를 하기위해 Dequeue()를 호출했는데 아무것도 없어서 Send 0을 하는 상황

(알파벳 : 노드)

[스레드1] A > (B) A백업, B할당

<자료구조 변화>

[스레드2] C > D 상황에서, A를 Alloc하고 Next를 nullptr세팅

[스레드1] 백업한 A의 Next가 nullptr이므로 B – Enqueue성공

C -> D / A-> B

이때 스레드1 입장에서는 A->B를 연결하는 것으로 Enqueue성공이라고 판단하기 때문에 UseSize++한다. 따라서 UseSize가 n개임에도 n-1개가 있는 상황

따라서 분명히 데이터를 Enqueue(1)하고, UseSize까지 늘어난 것을 확인했음에도 Dequeue를 실패 할 수 있음.

이 경우 데이터가없는데 WSASend호출

따라서 모든 Dequeue에는 예외구문이 들어가며,

**SendPost내부에서도 0인 경우 Send하지않도록 별도의 예외처리를 하여 해결. (사실 이 예외구문으로 위 세가지 문제를 임시적으로 다 막을수 있긴하지만 근본적인 해결방법은 아니다.)**

**Session의 해제-재할당**

넷서버 내부에서는 IOCount를 물고가고, 0이되어야 해제하므로 절대 재할당되지않음.

하지만 이 사정을 모르는 채팅서버는 모름. (해제 후 다른세션으로 재할당되어도 모름)

채팅서버가 넷서버로 요청하는 함수는 두가지다.

**1. SendPacket(송신요청)**

**2. DisconnectSession(송신요청)**

따라서 이 두함수는 다음과같은 예외구문을 거친다.

**1. FindSession -> SessionID로 세션찾음 (없으면 false리턴)**

**2. StartUseSession -> IOCount를 올려놓음으로서 이 세션은 이제 절대 해제되지않음**

**3. 함수호출시 넣었던 SessionID, 그리고 FindSession하여SessionID가 일치하는지 확인.**

**위 구문은 순서까지 맞아야함.**

**문제가 될것같은 부분 사전처리**

1. FindSession 하기도전에 이미 재할당이 되었음 -> false로 리턴되어 걸러짐

2. IOCount를 올리려는 직전에 재할당되었음. 3번에서 ID불일치시 리턴됨.

3. FindID , SessionID를 비교하는데 재할당되버리면? 2번에서 IOCount올려서 절대 해제안됨.

**Disconnect하는 방법**

Session내부에 DisconFlag, ReleaseFlag를 둔다.

DisconFlag - DisconnectSession으로 들어온경우 true.

ReleaseFlag – 완전히 해제된경우 true.

그리고 Session의 소켓을 INVALID\_SOCKET값으로 바꾼다.

**Recv는 아래와같은 안전장치로 모든 경우를 대비**

Recv/Send함수 바깥 -> DisconFlag확인하여 진입 ( true면 I/O를 걸러들어가지않음)

Recv함수 호출 직전 -> 소켓INVALID에서 막힘

Recv함수 호출 직후 -> 캔슬IO

**Send의 경우는 무조건 완료통지가 오기때문에, 완료통지쪽에서 DisconFlag로 막음.**

**외부 공격메시지 대응로직중 기억에 남는 버그**

**자동 형변환**

넷서버에서 메시지가 오면, 헤더를 확인후 길이만큼 메시지가 링버퍼에 들어왔는지 확인.

// 데이터가 덜왔으면 끊지않고 패스.

if (header.Len > pSession->RecvQ.GetUseSize() - HEADER\_SIZE)

break;

그런데 Len이 -1일때, 오른쪽 항이 양수인데도 해당구문은 왼쪽이 더 큰 경우로빠짐.

#define HEADER\_SIZE sizeof(NET\_HEADER) //2(LAN\_HEADER)

HEADER\_SIZE는 위와같아 define되어있고, sizeof의 반환타입은 singed형이 아닌 unsigned형이다!!

결국 -1은 해당 자료형의 가장큰 양수를 표현하게되어 오른쪽이 더 크다고 나오는것.

**CMsg 객체메모리 누수문제**

**문제원인 : 갑자기 CMsg가 해제되지않고 조금씩 누수..**

메모리사용량이 조금씩 증가하는것도 확인.

**1. TLS\_LockFree\_MemoryPool을 LockFreeMemoryPool로 바꿔봄 -> 계속 누수**

**2. Lock을 위아래로 꽉잡아봄 -> 계속 누수**

**3. CMsg를 쓰는곳 중 특정한 곳에서 나는지 확인 -> 쓰는곳 전부에서 누수**

**4. 더미 재연결설정 없이하면 누수안남**

**5. 해제되었을때 송신버퍼에 남아있는 CMsg가 있는지 확인 -> 이상없음**

**6. 해제할떄 재할당을 막기위한 StartUseSession에서... 예외구문을 타고 나가버렸을때 CMsg가 해제되지 못하고 없어지는것을 확인 -> 예외처리 해줘서 해결**

**플레이어 객체 우상향 그래프**

플레이어가 해제되지 못하고 계속하여 할당.

**1. 넷서버 의심. 넷서버의 Session에는 이상이 없음**

**2. ON\_CLIENT\_LEAVE함수에서 플레이어를 찾지 못하는것을 확인**

**3. 더미 재연결설정을 끄면 문제가 나지않음**

**4. 단, 즉시 재연결을 1000m/s이상으로 길게잡으면 문제가 거의나지않음**

**5. ON\_CLIENT\_JOIN으로는 들어왔는데, ON\_CLIENTLEAVE에는 없는 새션을 찾아냄.**

**Index는 같으나(메모리가 같음) SessionID가 다름.**

**3,4,5를 종합해보면 재할당문제라는것을 알수있음.**

\_InvalidIndexStack.push(Index); <<메모리 재사용 가능하게끔 스택에 PUSH

OnClientLeave << 채팅서버에 세션이 해제되었음을 알림

**시나리오**

**스레드1: \_InvalidIndexStack.push(Index); (스레드 멈춤)**

**스레드2: Accept스레드에서 Accept. (재할당). OnClientJoin을 호출하러 가기전에 스레드멈춤**

**스레드1: 스레드 재개. 새로들어온 Session을 OnClientLeave로 핸들링 해버림.**

**새로할당된 Id이므로 플레이어 맵에서 찾을수없음.**

따라서 해제되어야할 Player가 해제되지못하고 계속해서 우상향그래프.

만약 여기서 재할당된 세션을 Disconnect까지 처리했더라면 DownClient까지 발견했을것..

**CMsg 메모리풀의 청크 개수를 늘리면 왜 성능이 떨어지는가?**

TLS는 쓰레드별로 메모리를 가져간다.

청크의 크기커진다? 1개를 할당하더라도 캐시만큼 할당을 해온다. -> 캐시히트 확률 떨어진다.

이렇게 생각한 근거. 캐시메모리보다 더많은 메모리풀을 할당하여 들고있다.

그리고 CMsg의 버퍼크기를 1400 -> 200으로 줄였을때 성능이 획기적으로 늘어났다..