Kwak Kyung In rhkrruddls19999@gmail.com 010-7496-6307

welcome to my **POPTFOLIO**

IOCPCHATSERVER

https://github.com/KwakKyungIn/portfolio-kwak-

Est. 2025

IOCPCHATSERVER

프로젝트 목적

고성능 비동기 서버 구조 설계 및 대규모
채팅 트래픽 처리·지연 최적화 검증

주요 기능

실시간 채팅 서비스 필수 기능(로그인/세션
바인드, 방 생성·참여·채팅, 퇴장) 구현

사용 기술 스택

C++17, Win32 IOCP, Protobuf, ODBC(MSSQL), Custom Network Library(ThreadManager, JobQueue, Buffer, Session 등 자체 구현)

차별점 & 강점

IOCP 기반 JobQueue·ThreadPool 설계, 직렬화/버퍼 관리 최적화,
대규모 부하 테스트(500 clients, 10 rooms, 10.0 RPS) 안정성 검증

향후 개선점

분산 서버 구조 확장, 모니터링/로깅 고도화, 게임 서버 기능 확장
(매치메이킹, 인벤토리, DB 연동 등)

CONTENTS

IocpChatServer

Performance Test Summary

Workload

- ☑ Client : 500 | Rooms : 10
- ☑ Per - Client RPS : 10.0
- ☑ Payload mix : 64B 80 % / 256B 18% / 1~2KB 2%

Throughput (app in/s)

4,948 req/s

Broadcast (bc - deliv/s)

248,153

fan-out x50.2

대역폭 (out)

11.66 MB/s

in 0.107 MB/s

Latency(지연)(avg | p50 | p90 | p00)

0.456 ms | 0.142 ms | 1.163 ms | 5.569 ms

<=100µs:205707건 | <=500µs:277936건 | <=1ms:126558건 | <=5ms:61255건 | >50ms:0 (250s) (250s 기준)

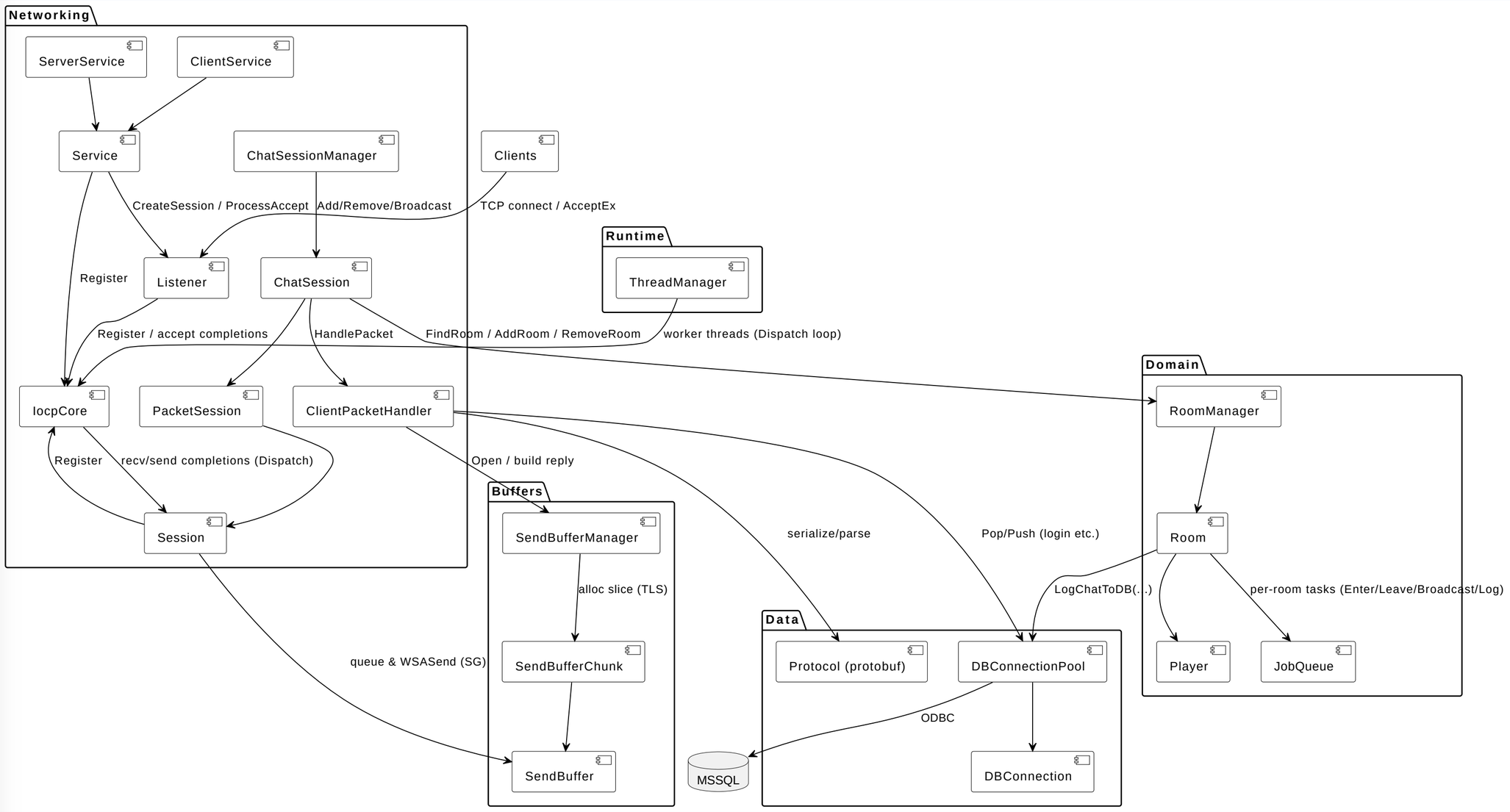
Queue & Buffer

- ✓ job e/x 9,889/9,889
- ✓ jobs q-peak : 110
- ✓ sendbuf inuse : 15.82(peak 18.28)
- ✓ mpool inuse : ~ 69,632
- ✓ mpool free/s ~ 101,642

IocpChatServer

System Architecture (Component)

- Performance Test Summarry
- **System Architecture (Component Diagram)**
- Core Operations
  - Login & Session Bind
  - Create Room
  - Join Room
  - Room Chat
  - Leave Room
- System Architecture (Class Diagram)
- File Structure Overview
- Performance Test
  - Workload&PerformanceKPIs
  - Queues & Buffers KPIs
  - Performance Analysis



Login & Session Bind

Feature

- ☑ **연결 직후 계정 확인:** 클라이언트 요청을 받아 DB에서 사용자 정보를 조회해 신원을 확인한다.
- ☑ **세션-플레이어 바인딩:** 성공 시 Player를 생성해 ChatSession에 붙이고, 이후 기능(방/채팅) 사용 기준으로 삼는다.
- ☑ **간결한 응답 경로:** 응답 패킷은 MakeSendBuffer<>로 직렬화 후 세션 송신 큐→WSASend(SG) 배치 전송한다.
- ☑ **부하 대비:** DBConnectionPool 사용으로 로그인 스파이크 시에도 Pop/Push + acquire wait(ms) 추적 가능하다.

Execution Snapshot

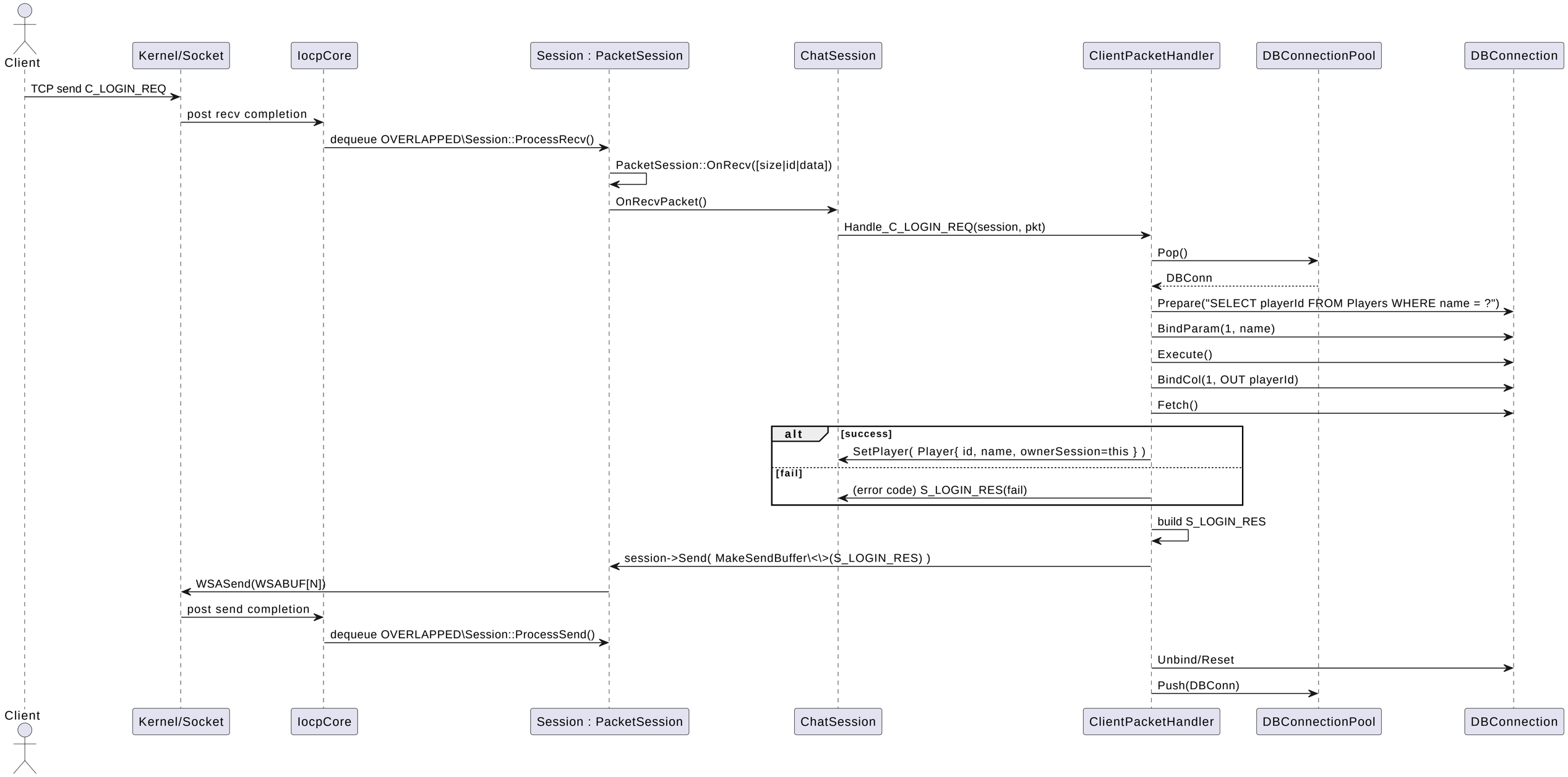
```
CoreGlobal Initialized
DB Connection established.
[00:31:16.102] CoreGlobal Initialized | IOCP workers=16
[00:31:16.897][thr:13920][sess:00000193653A2270]
 OnConnected()
[00:31:31.041][thr:13920][sess:00000193653A2270]
 C_LOGIN_REQ name="User1"
[00:31:31.042][thr:13920][sess:00000193653A2270]
[player:5] S_LOGIN_RES success=true
```

<Server>

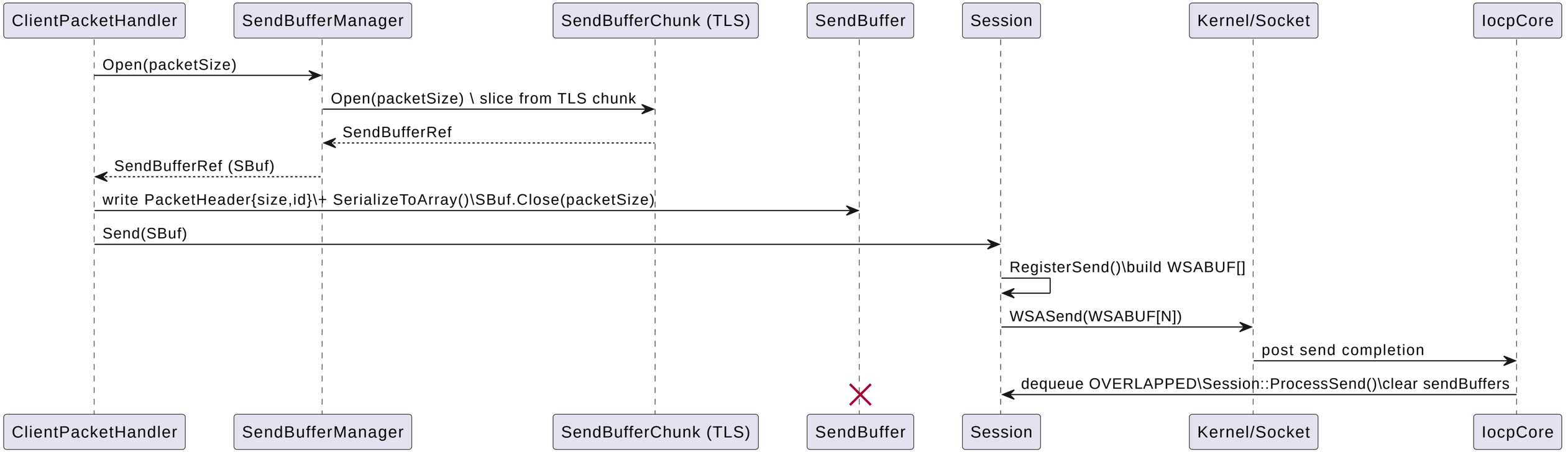
```
CoreGlobal Initialized
로그인할 플레이어 이름을 입력하세요: User1
서버에 'User1' 플레이어의 로그인 요청을 보냈습니다.

로그인 성공! Player ID: 5
서버 메시지: Login successful.

[방 생성: 1] [방 입장: 2] ->
```

<Client>

IocpChatServer Login & Session Bind — Packet Sequence



IocpChatServer Login & Session Bind — Memory Sequence



Create Room

Feature

☑ **방 생성 & 즉시 입장:** 요청을 받아 Room을 만들고, 생성자를 바로 입장시켜 이후 채팅/멤버 관리의 기준 상태를 마련.

☑ **빠른 응답 경로:** RoomManager.AddRoom으로 등록 후 SetId/SetName/SetType/Enter 처리, 응답은 MakeSendBuffer<> → session Send() → WSASend(SG) 로 반환

Execution Snapshot

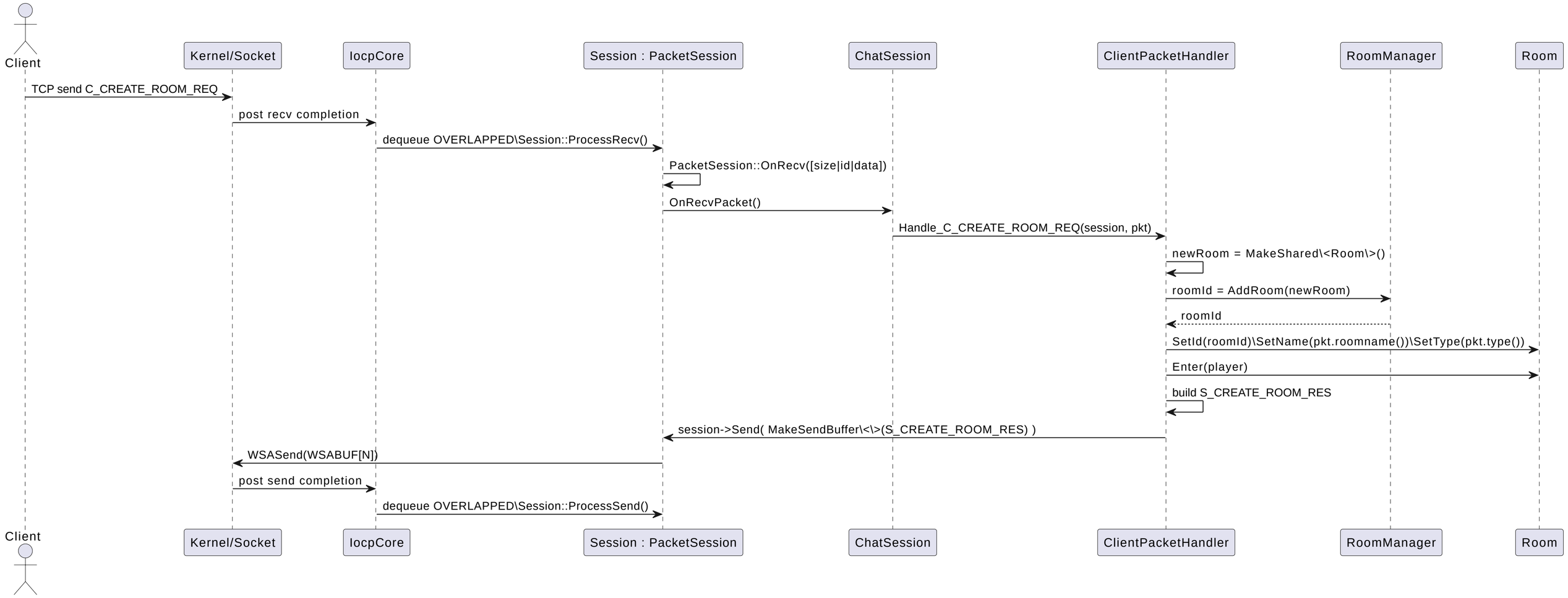
```
[player:5] C_CREATE_ROOM_REQ name="TestRoom"
[00:32:24.850][thr:12280]
[room:1] RoomManager.AddRoom WRITE → roomId=1
[00:32:24.851][thr:12280][sess:00000193653A2270]
[player:5][room:1] RoomManager.AddRoom → roomId=1
[00:32:24.852][thr:12280][sess:00000193653A2270]
[player:5][room:1] Enter OK members=1
[00:32:24.852][thr:12280][sess:00000193653A2270]
[player:5][room:1] Room.Init SetId/Name/Type; Enter(player=5)
[00:32:24.852][thr:12280][sess:00000193653A2270]
[player:5][room:1] S_CREATE_ROOM_RES success=true
Player User1 created a room. Room ID: 1
```

<Server>

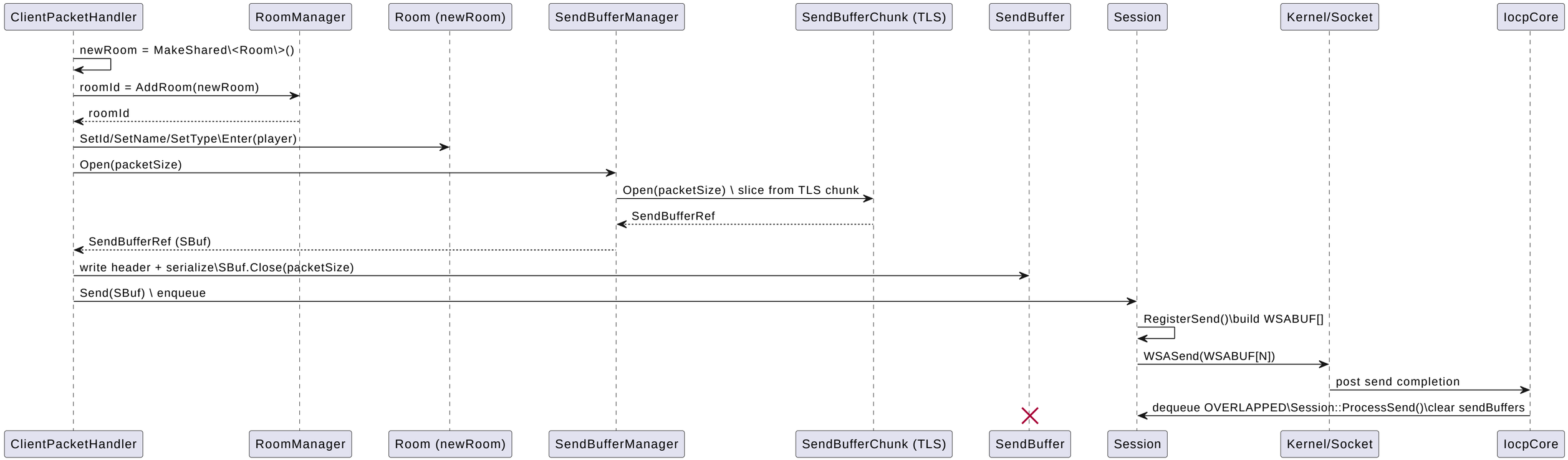
```
[방 생성: 1] [방 입장: 2] -> 1
생성할 방 이름을 입력하세요: TestRoom
'TestRoom' 방 생성 요청을 보냈습니다.

==========================================
Room created successfully!
Room ID: 1, Room Name: TestRoom
채팅을 시작하세요! (나가려면 /exit 입력)
--- 현재 접속자 ---
- User1 (ID: 5)
==========================================
```

<Client>

IocpChatServer Create Room — Packet Sequence



IocpChatServer Create Room — Memory Sequence



Join Room

Feature

- ☑ **기존 방 참여:** 클라이언트 요청으로 지정된 roomId의 방을 찾아 Room::Enter(player)를 시도.
- ☑ **초기 상태 전달:** 성공 시 응답에 방 정보 + 현재 멤버 목록을 포함해 즉시 동기화.
- ☑ **주변 알림:** 참가가 완료되면 S_JOIN_ROOM_NTF를 BroadcastWithoutSelf로 전파해 다른 멤버에게 참가 사실을 알림..
- ☑ **응답 경로:** MakeSendBuffer<> → session Send() → WSASend(SG).

Execution Snapshot

```
[00:34:55.437][thr:13920][sess:00000193653A63B0]
[player:6] C_JOIN_ROOM_REQ roomId=1
[00:34:55.437][thr:13920][sess:00000193653A63B0]
[player:6][room:1] Enter OK members=2
[00:34:55.437][thr:13920][sess:00000193653A63B0]
[player:6][room:1] Enter OK (WRITE)
[00:34:55.437][thr:13920]
[room:1] BroadcastWithoutSelf snapshot members=1 deliveries=1
 (exclude=6)
[00:34:55.438][thr:13920][sess:00000193653A63B0]
[player:6][room:1] S_JOIN_ROOM_RES success=true
```

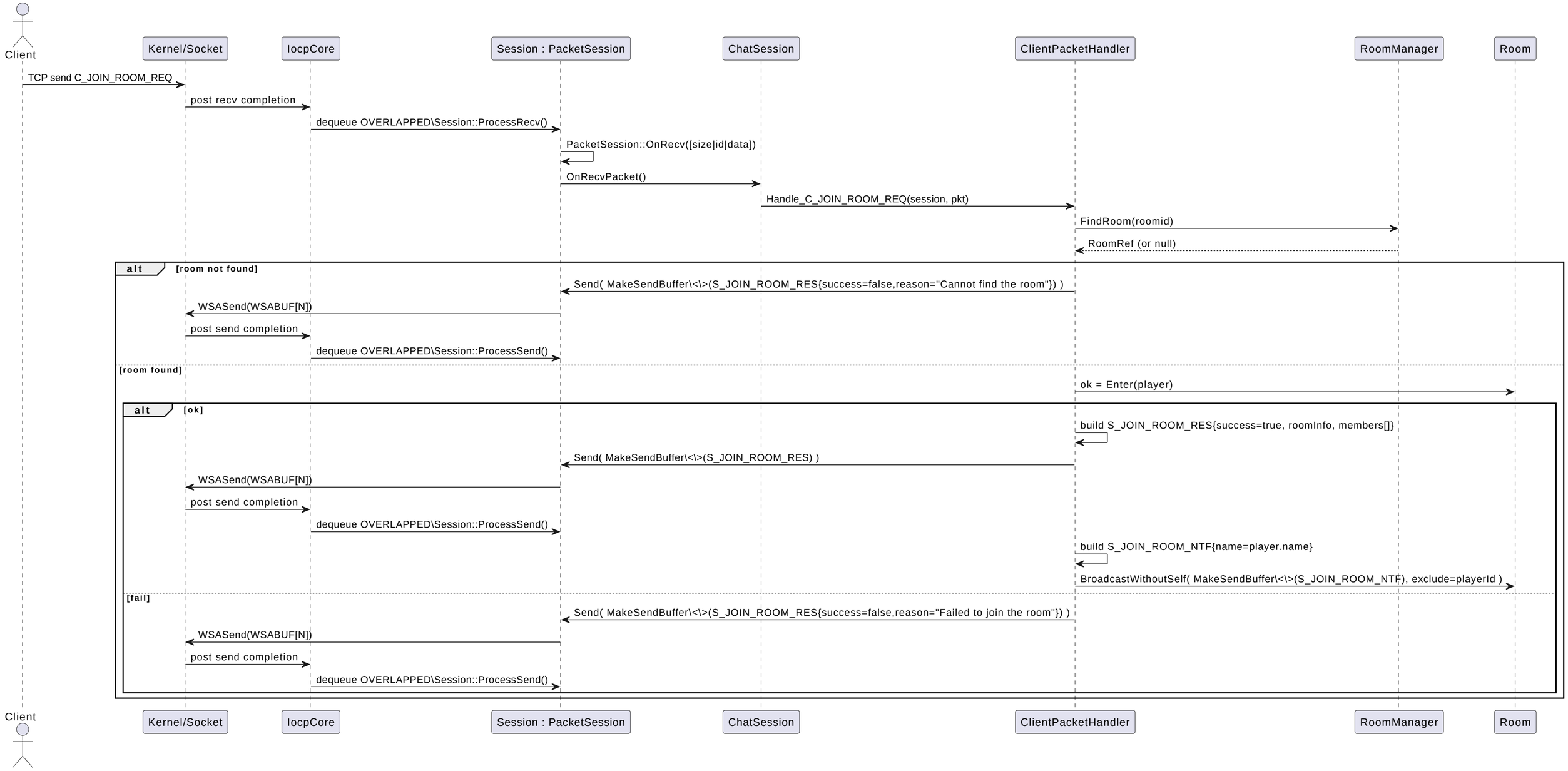
<Server>

```
==========================================
Room created successfully!
Room ID: 1, Room Name: TestRoom
채팅을 시작하세요! (나가려면 /exit 입력)
--- 현재 접속자 ---
- User1 (ID: 5)
==========================================
User2 님이 방에 들어왔습니다!
```

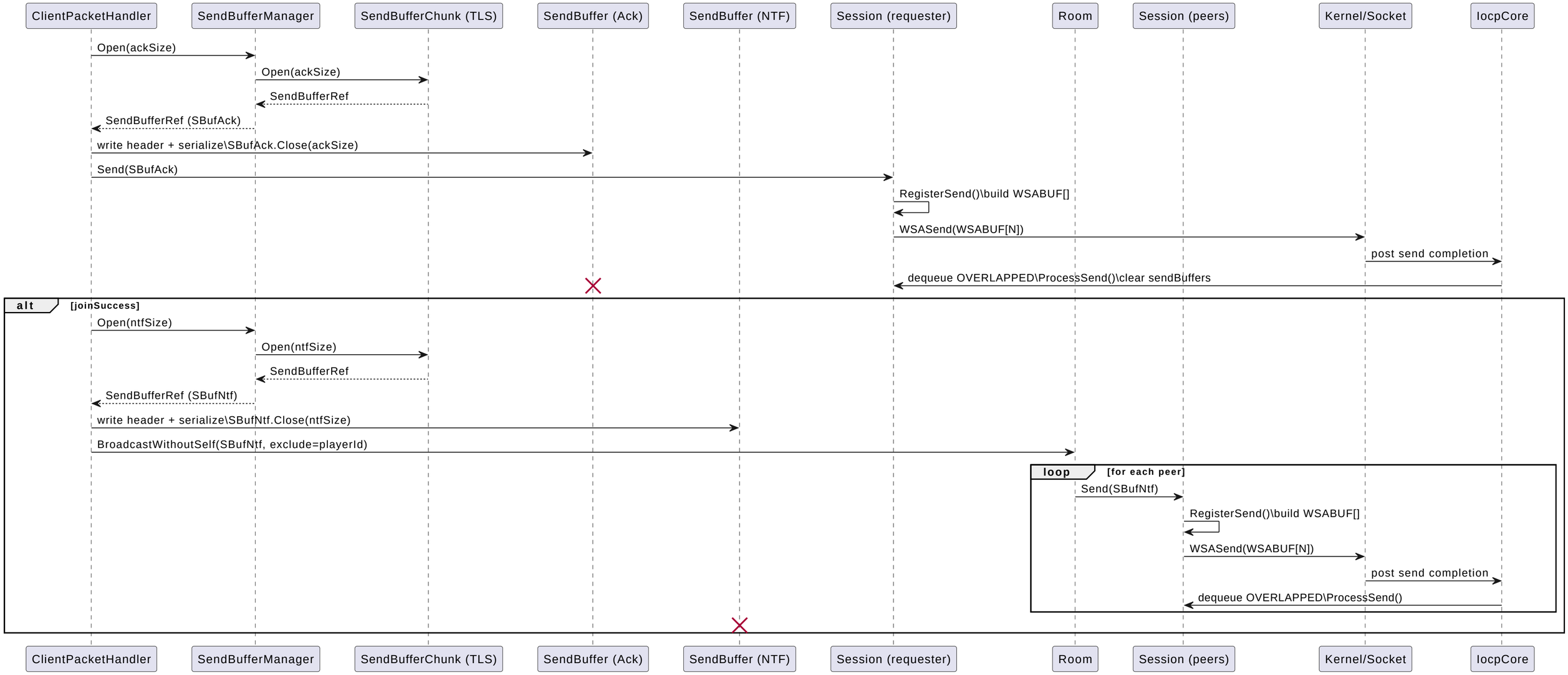
```
[방 생성: 1] [방 입장: 2] -> 2
입장할 방 번호를 입력하세요: 1
방 ID 1 입장 요청을 보냈습니다.

==========================================
[TestRoom] 방 (ID: 1)에 입장했습니다.
채팅을 시작하세요! (나가려면 /exit 입력)
--- 현재 접속자 ---
- User1 (ID: 5)
- User2 (ID: 6)
==========================================
```

<Client>

IocpChatServer Join Room — Packet Sequence



IocpChatServer Join Room — Memory Sequence



Room Chat

Feature

- **방 내 메시지 전달:** 클라이언트가 보낸 채팅을 서버에서 받아 현재 방의 모든 멤버에게 브로드캐스트.
- **간결한 송신 경로:** 패킷은 MakeSendBuffer<>로 직렬화되고, Room::Broadcast()가 멤버 세션에 Send()를 호출해 WSASend(SG) 로 전송
- **로그 기록:** Room::LogChat()(파일) / Room::LogChatToDB()(DB)를 룸 JobQueue에서 비동기로 처리.

Execution Snapshot

```
[room:1] Broadcast snapshot members=2 deliveries=2
[00:36:17.686][thr:11792][sess:00000193653A2270]
[player:5][room:1] C_ROOM_CHAT_REQ bytes=11
[00:36:33.682][thr:11792]
[room:1] Broadcast snapshot members=2 deliveries=2
[00:36:33.682][thr:11792][sess:00000193653A63B0]
[player:6][room:1] C_ROOM_CHAT_REQ bytes=24
```

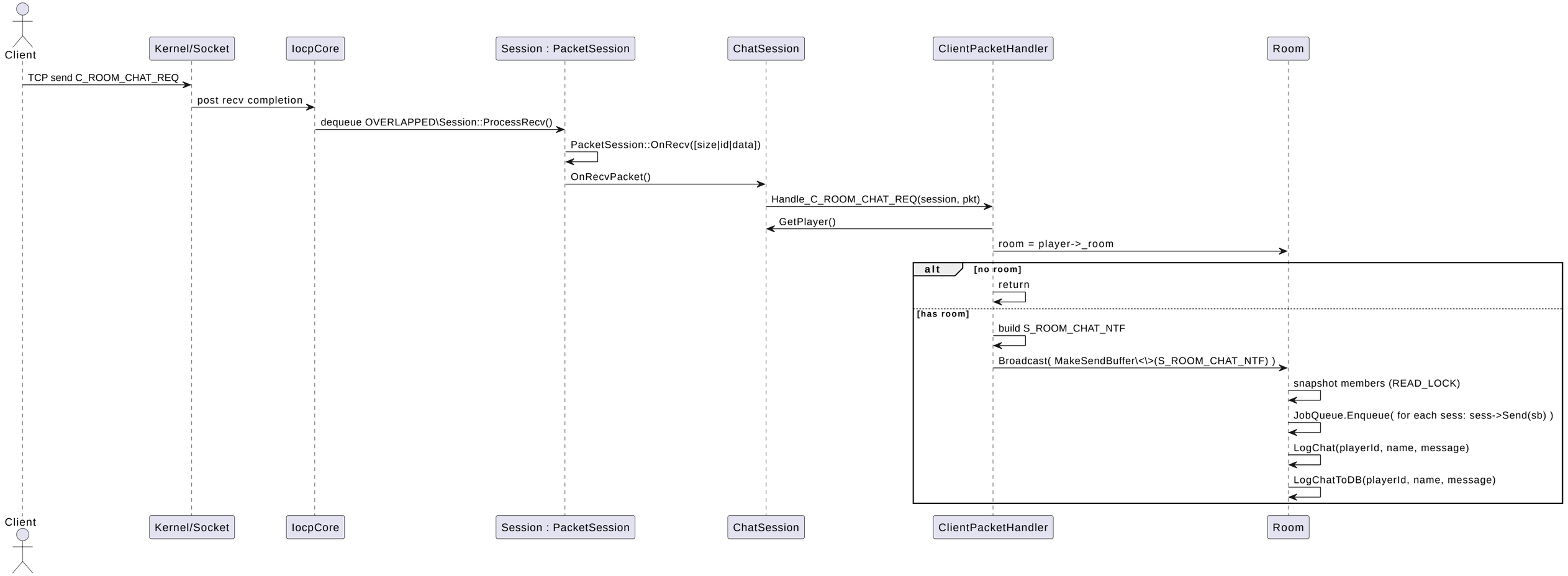
<Server>

```
================================================
User2 님이 방에 들어왔습니다!
Hi Im User1
[Room 1] Player(5): Hi Im User1
[Room 1] Player(6): Hi Im User2 Test is Good
```

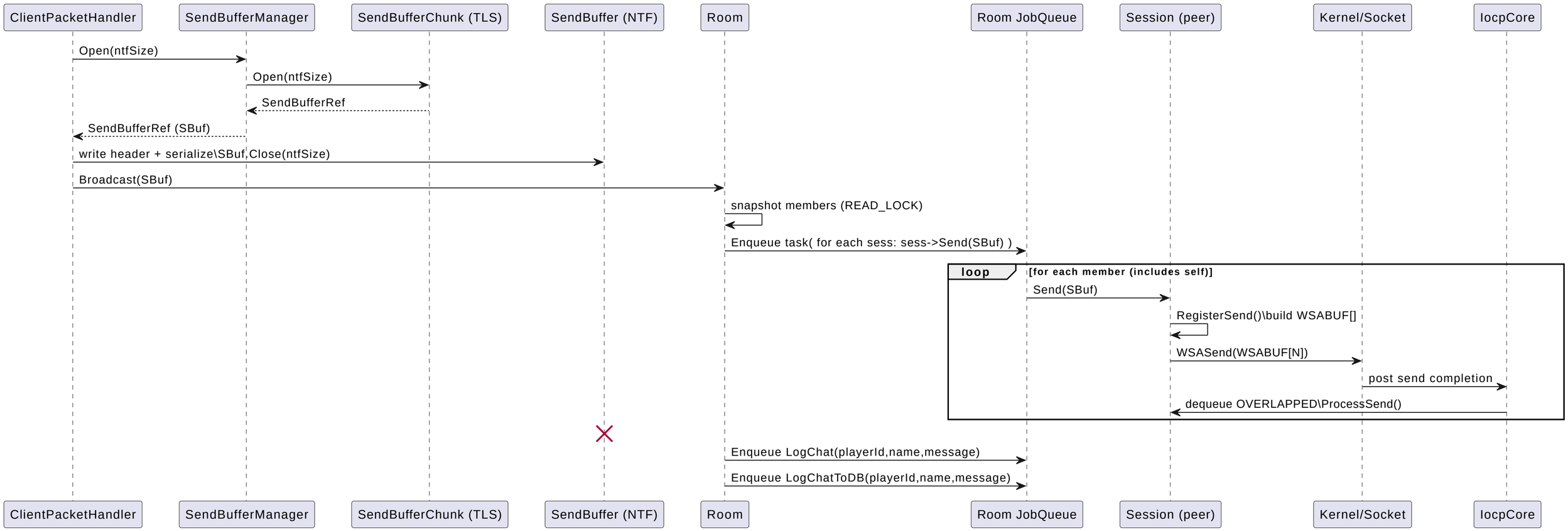
```
================================================
[Room 1] Player(5): Hi Im User1
Hi Im User2 Test is Good
[Room 1] Player(6): Hi Im User2 Test is Good
```

<Client>

IocpChatServer Room Chat — Packet Sequence



IocpChatServer Room Chat — Memory Sequence



Leave Room

Feature

- **방 이탈 처리:** 플레이어가 현재 방에서 안전하게 빠져나가도록 Room::Leave()로 멤버십을 해제하고 세션 상태를 정리.
- **주변 알림:** 이탈 사실을 S_ROOM_CHAT_NTF("... has left the room.")로 다른 멤버에게만 전파(BroadcastWithoutSelf).
- **정리:** 방이 비면 RoomManager.RemoveRoom(roomId)로 방 제거. 응답은 S_LEAVE_ROOM_ACK로 짧게 반환.

Execution Snapshot

```
[00:39:35.560][thr:11792][sess:00000193653A63B0]
[player:6][room:1] C_LEAVE_ROOM_REQ
[00:39:35.560][thr:11792][sess:00000193653A63B0]
[player:6][room:1] Leave OK members=1
[00:39:35.560][thr:11792]
[room:1] BroadcastWithoutSelf snapshot members=1 deliveries=1
 (exclude=6)
[00:39:35.560][thr:11792][sess:00000193653A63B0]
[player:6][room:1] S_LEAVE_ROOM_ACK success=true
```

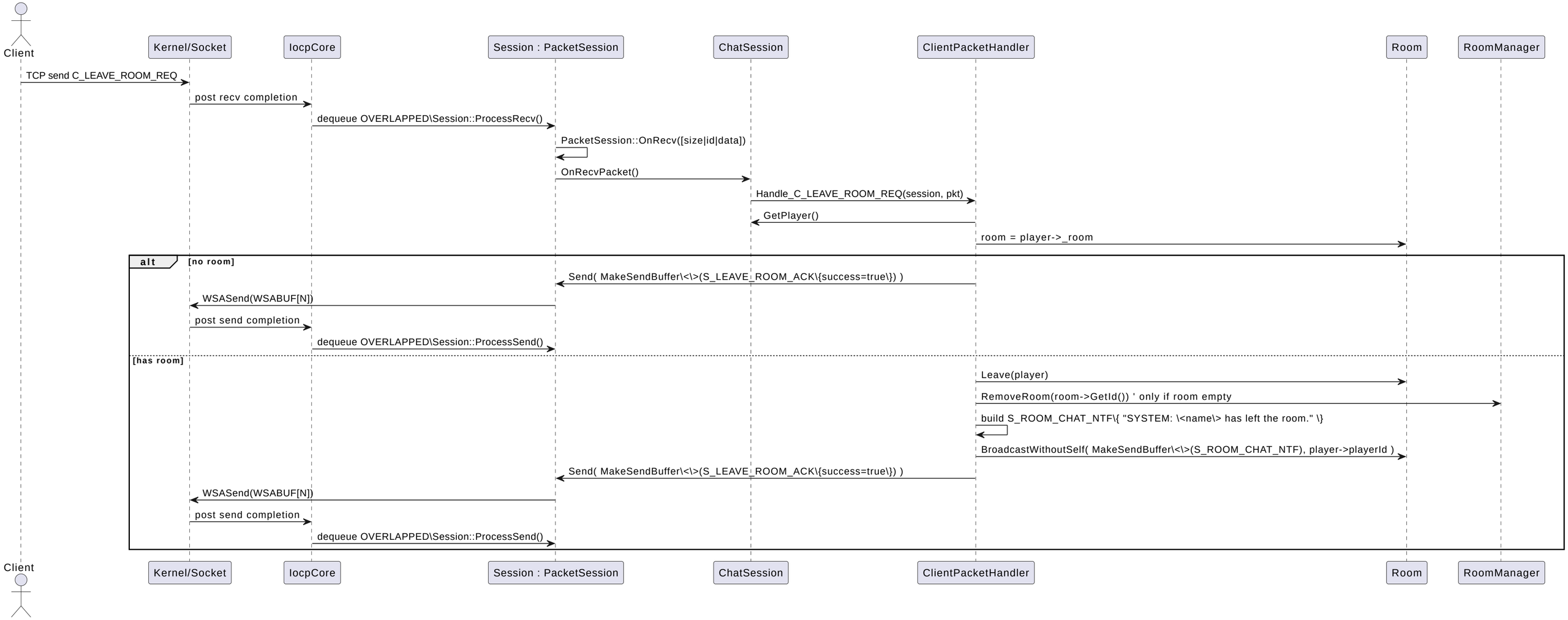
<Server>

```
================================================
User2 님이 방에 들어왔습니다!
Hi Im User1
[Room 1] Player(5): Hi Im User1
[Room 1] Player(6): Hi Im User2 Test is Good
SYSTEM: User2 has left the room.
```

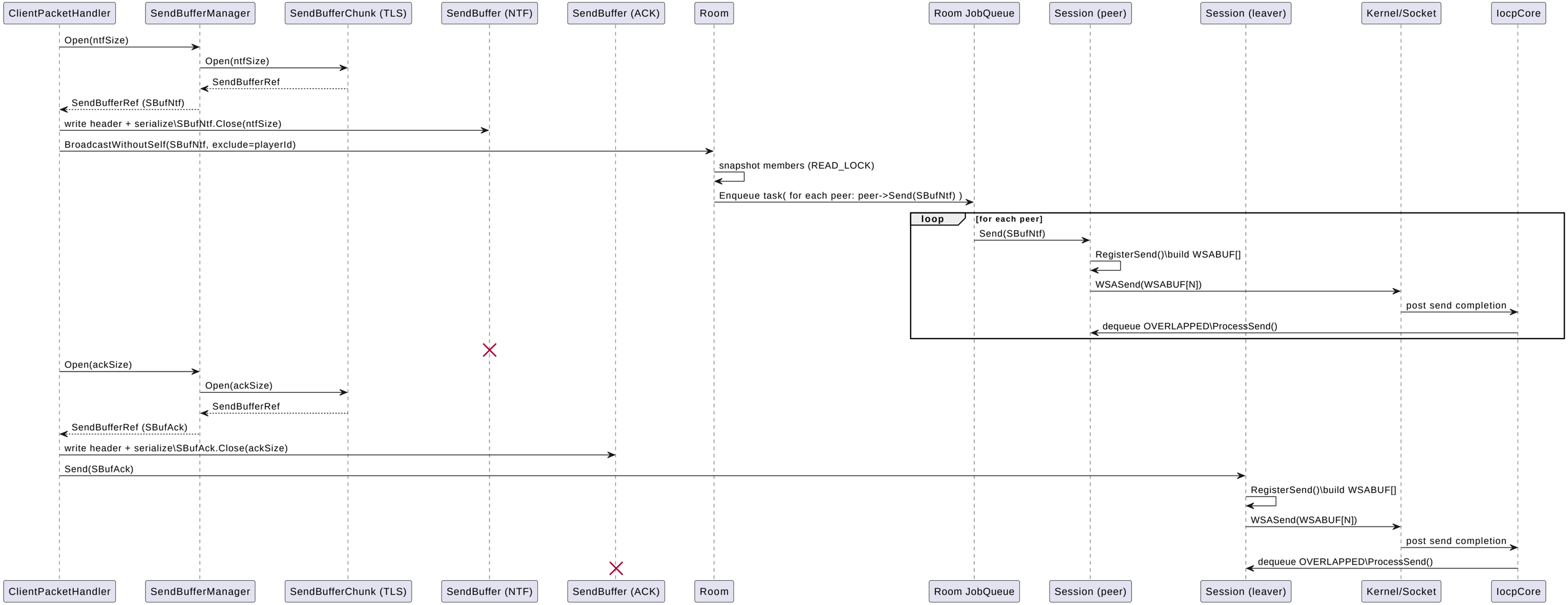
```
================================================
[Room 1] Player(5): Hi Im User1
Hi Im User2 Test is Good
[Room 1] Player(6): Hi Im User2 Test is Good
EXIT
방 나가기 요청을 보냈습니다.

방에서 성공적으로 나갔습니다.
메인 메뉴로 돌아갑니다.

[방 생성: 1] [방 입장: 2] ->
```

<Client>

IocpChatServer Leave Room — Packet Sequence



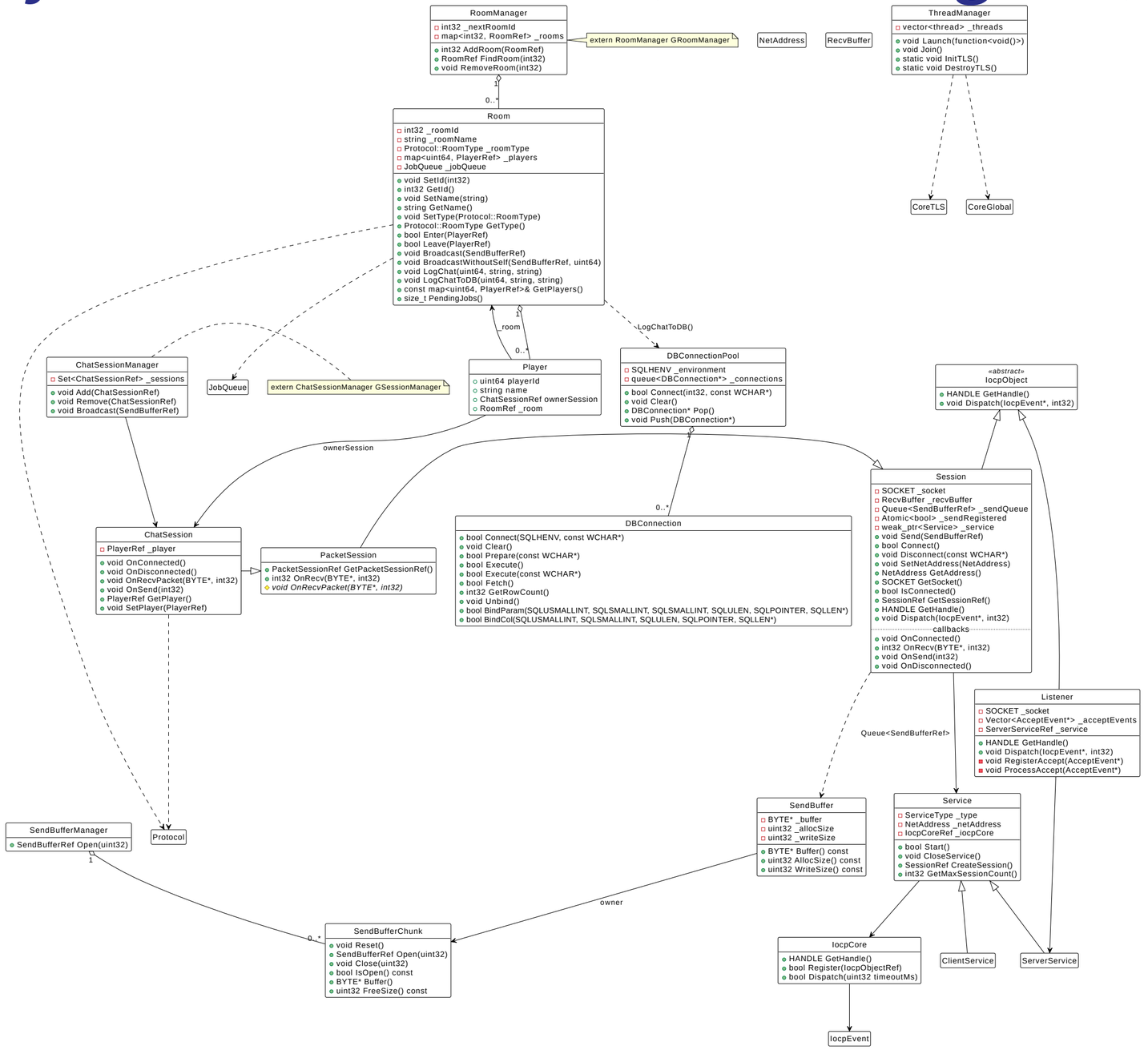
IocpChatServer Leave Room — Memory Sequence



IocpChatServer

- Performance Test Summarry
- System Architecture (Component Diagram)
- Core Operations
  - Login & Session Bind
  - Create Room
  - Join Room
  - Room Chat
  - Leave Room
- **System Architecture (Class Diagram)**
- File Structure Overview
- Performance Test
  - Workload&PerformanceKPIs
  - Queues & Buffers KPIs
  - Performance Analysis

System Architecture (Class Diagram)



File Structure Overview - ServerCore

IOCP 엔진 & 이벤트 디스패치

- IOCP 핸들 생성/등록/대기·디스패치, OVERLAPPED 확장 이벤트로 Accept/Recv/Send 분기 관리.
- Files: IocpCore.h/.cpp, IocpEvent.h/.cpp

소켓/주소 베이스 레이어 (Windows 확장·옵션·주소 유틸)

- AcceptEx/ConnectEx/DisconnectEx 로딩·바인딩, 비동기 소켓 생성 및 옵션 설정, bind/listen, IPv4 문자열↔바이너리·포트 변환.
- Files: SocketUtils.h/.cpp, NetAddress.h/.cpp

세션 I/O 파이프라인 (Recv 링버퍼 + SG-WSASend 배치 전송)

- 세션 생명주기/디스패치, 수신은 링버퍼로 프레이밍 후 상위로 올림, 송신은 세션 Send 큐를 비워 WSABUF 배열 구성 → WSASend 일괄 전송. 락 안=큐 비우기, 락 밖=WSABUF 구성. _sendRegistered로 중복 등록 방지.
- Files: Session.h/.cpp, RecvBuffer.h/.cpp, SendBuffer.h/.cpp

버퍼링 & 메모리/오브젝트 풀 (전역 인프라)

- 사이즈별 슬랩형 MemoryPool과 ObjectPool로 동적 할당 최소화. SendBufferChunk는 thread_local 재사용·슬라이스 오픈/클로즈로 전송 버퍼 재활용.
- Files: MemoryPool.h/.cpp, ObjectPool.h, SendBuffer.h/.cpp

동시성 제어 (커스텀 RW 스핀락)

- 읽기 공유/쓰기 단독 모델, 해제·재진입·양보(백오프)까지 고려한 RW 스핀락으로 룸/매니저 등 자료구조 보호.
- Files: Lock.h/.cpp

DB 연결 & 풀

- ODBC 기반 Prepare/Bind/Execute/Fetch/Unbind 래핑, 커넥션 풀 Pop/Push·종료 처리로 로그인 등 DB 접근 지연 최소화.
- Files: DBConnection.h/.cpp, DBConnectionPool.h/.cpp

File Structure Overview - ChatServer

부팅 & 런타임 구동

- 프로세스 진입점: network core/listener 초기화, IOCP worker threads 준비
- packet handlers 등록, 서버 loop 구동, graceful shutdown 처리
- Files: ChatServer.cpp

세션 수명주기 & 전역 세션 관리

- ChatSession: 접속/해제 콜백, RecvBuffer 프레이밍, 세션별 Send 큐 → WSASend(SG), _sendRegistered로 중복 전송 방지
- ChatSessionManager은 세션 등록/조회/삭제, 전체 세션 대상 보조 연산(안전한 순회/정리)
- Files: ChatSession.h/.cpp, ChatSessionManager.h/.cpp

프로토콜 라우팅 & 직렬화

- 클라→서버 C_* 요청 라우팅(로그인/방 생성·입장·퇴장/채팅)
- 응답/알림 S_* 패킷 빌드(MakeSendBuffer<> 직렬화) 및 세션 송신 경로 트리거
- Files: ClientPacketHandler.h, ClientPacketHandler.cpp

룸 도메인(멤버/브로드캐스트/로그)

- Room: 멤버 추가/제거, Broadcast / BroadcastWithoutSelf(READ 스냅샷 후 session에게 각각 Send())
- 채팅 로그 파일/DB 기록용 room JobQueue 비동기 작업 실행
- Files: Room.h/.cpp, JobQueue.h

룸 디렉터리

- 방 생성/등록/ID 부여, 조회/삭제
- Multi Threads를 안전한 Map으로 접근(생성/삭제 시 WRITE, 조회 시 READ 정책에 맞춰 사용)
- Files: RoomManager.h/.cpp

플레이어 모델

- 플레이어 식별자/이름 등 상태 보관, 현재 room/session 연계에 필요한 최소 정보 제공
- Files: Player.h/.cpp

File Structure Overview - DummyClient

런타임 & 세션 오케스트레이션

- 프로파일(kClients / kRooms / kRps) 설정 → 공용 IOCP 코어/워커 기동 → 세션 생성·연결 → 시작/종료 신호 처리. 접속 직후 로그인 트리거.
- Files: DummyClient.cpp, ServerSession.h

패킷 라우팅 & 직렬화 유틸

- 패킷 ID → 핸들러 테이블 초기화, 공통 파서/분배(HandlePacket) 수행. MakeSendBuffer<>로 C_* 요청(로그인/방생성/입장/퇴장/채팅) 직렬화.
- Files: ServerPacketHandler.h, ServerPacketHandler.cpp

Scenario scheduler & RPS controller

- 로그인 완료 감지 후 일부는 CreateRoom, 나머지는 JoinRoom으로 분산. per-client 페이싱으로 목표 RPS 유지, 랜덤 페이로드 생성해 채팅 전송.
- Files: DummyClient.cpp

State sync & feedback

- 서버 응답(S_LOGIN_RES, S_CREATE_ROOM_RES, S_JOIN_ROOM_RES, S_LEAVE_ROOM_ACK)으로 isLoggedIn / isInRoom / roomId 갱신, 콘솔 피드백. 연결 끊김 시 상태 초기화.
- Files: ServerPacketHandler.cpp, ServerSession.h

IocpChatServer

Performance Test - Workload & PerformanceKPIs

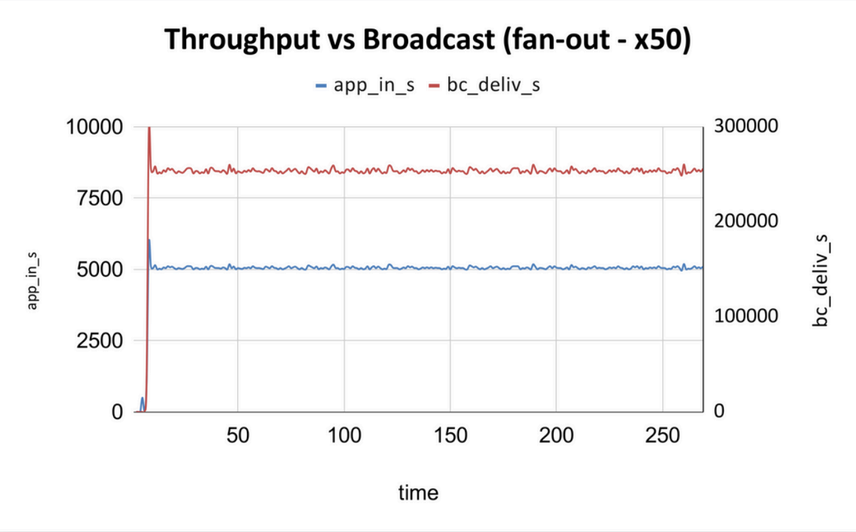
Workload & Test Setup

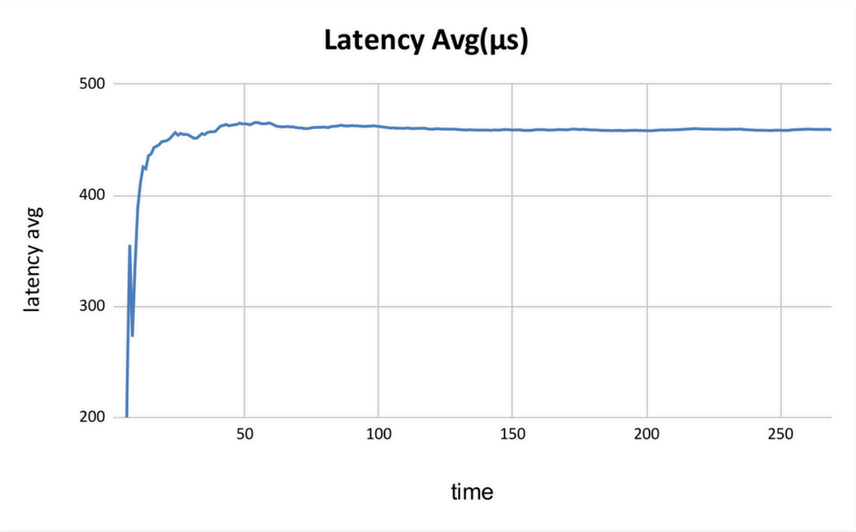
- **Clients / Rooms**: {500clients} / {10rooms}
- **Per-client RPS**: {10rps}
- **Payload mix**: 64B {80}% · 256B {18}% · 1–2KB {2}%
- **Fan-out**: ×{49.75}
- **Environment**: {CPU : Intel Core i5-14400F
  RAM : 32 GB
  OS : Windows 11 (64bit, Release build)
  I/O model=Win32 IOCP (Overlapped I/O)}

sec	io r/s	io s/s	app in/s	app out/s	bytes r/s	bytes s/s	bc-deliv/s	avg(µs)	p50(µs)	p90(µs)	p99(µs)
10s	5,041	117,849	5,041	5,041	106,158	11,660,834	252,961	388	206	1,086	6,733
70s	5,044	120,346	5,044	5,044	105,789	11,891,003	253,144	460	203	985	4,564
130s	5,110	114,104	5,110	5,110	107,355	12,057,805	256,510	459	188	1,014	4,812
190s	5,096	80,294	5,096	5,096	106,625	12,000,661	255,766	458	86	1,400	5,740
250s	5,045	133,973	5,045	5,045	106,339	11,920,759	253,195	458	278	964	4,445

IocpChatServer

Performance Test - Workload & PerformanceKPIs





Headline KPIs

- Throughput (app in/s): {4,948.37}
- Broadcast (bc-deliv/s): {248,153}
- Bandwidth (out MB/s): {11.66 MB/s}
- Fan-out: ×{49.75}
- I/O send ops (io s/s): {96,663}
- SG efficiency (deliveries per I/O): {bc_deliv} / {io_s_s} ≈ {2.567}

Latency(지연)(avg | p50 | p90 | p99)

0.456 ms | 0.142 ms | 1.163 ms | 5.569 ms

<=100µs:205707건 | <=500µs:277936건 | <=1ms:126558건 | <=5ms:61255건 | >50ms:0 (250s) (250s 기준)

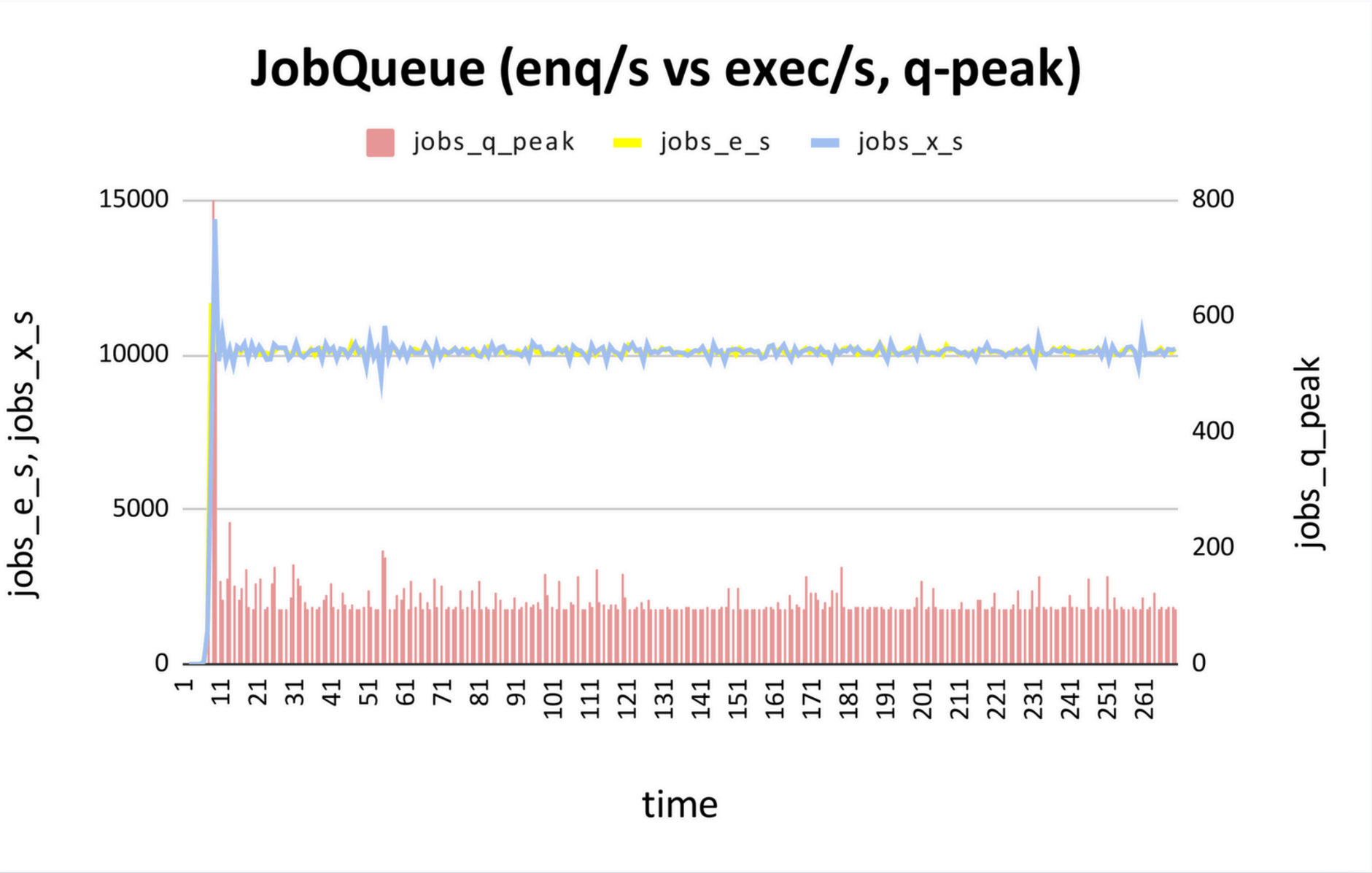
Performance Test - Queues & Buffers KPIs

Workload

sec	job e/s	job x/s	jobs q-peak	sendbuf inuse	sendbuf peak	sendbuf push/s	sendbuf pop/s	mpool inuse	mpool peak	mpool alloc/s	mpool free/s
10s	10,082	9,794	142	16	18	43	43	69,930	70,040	120	122,892
70s	10,088	9,934	5,135	16	18	39	39	71,106	71,222	1	125,390
130s	10,220	9,934	96	16	18	43	43	71,154	71,292	0	119,214
190s	10,192	9,934	96	16	18	40	40	71,196	71,310	0	85,390
250s	10,090	9,934	150	16	19	44	44	71,201	71,301	0	139,018

- **잡 큐(JobQueue) :** enq/s = {9,889.28}, exec/s = {9,888.97}, q-peak = {110.21}.
- **송신 버퍼(Send buffers) :** sendbuf inuse = {15.82} (peak {18.28}), push/s = {39.06}, pop/s = {39.12}.
- **메모리 풀(Memory pool) :** mpool inuse ≈ {69,631.66} (peak {69,742.92}), free/s = {101,641.84}.
- **메모리 풀 재사용 비율 (Memory Pool Reuse ratio) :** reuse% = 100 × (1 − alloc/s ÷ (alloc/s + free/s)) → {98.97}%

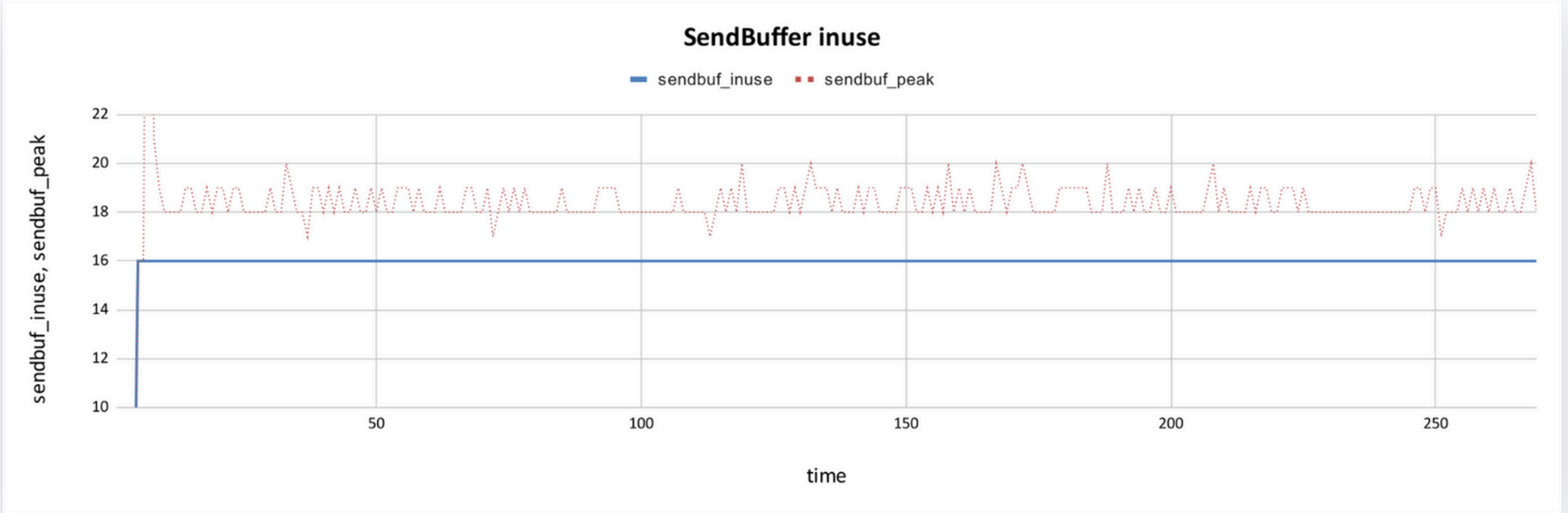
Performance Test - Queues & Buffers KPIs

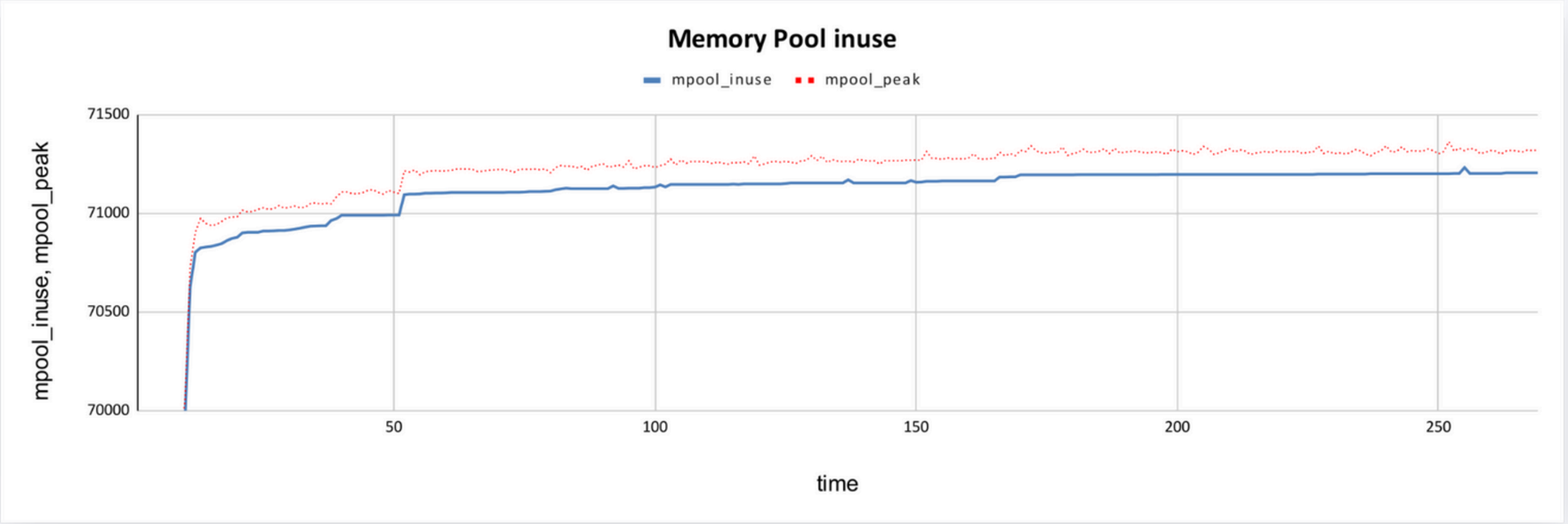


IocpChatServer

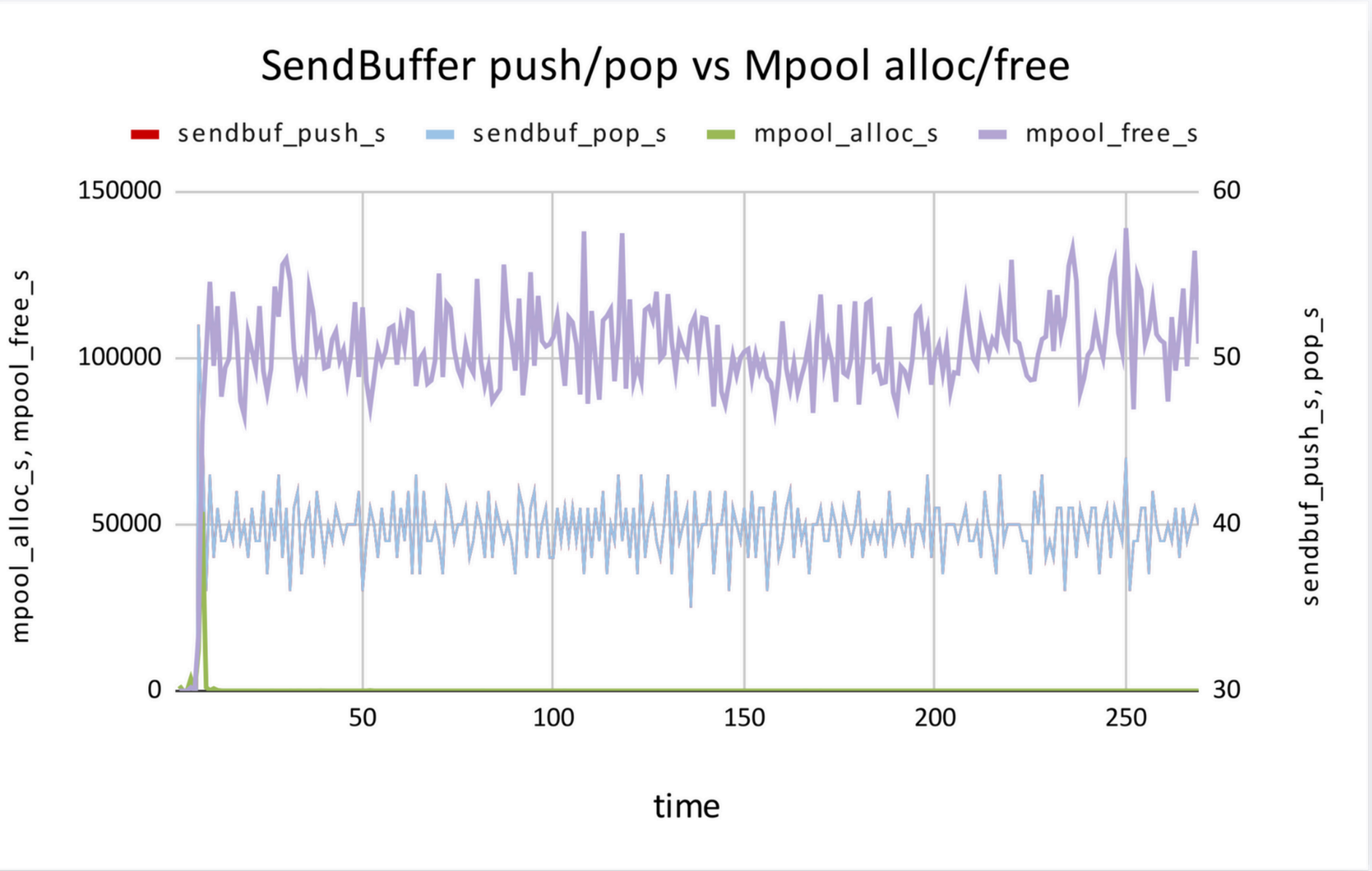
- Performance Test Summarry
- System Architecture (Component Diagram)
- Core Operations
  - Login & Session Bind
  - Create Room
  - Join Room
  - Room Chat
  - Leave Room
- System Architecture (Class Diagram)
- OverView
- **Performance Test**
  - Workload&PerformanceKPIs
  - Queues & Buffers KPIs
  - Performance Analysis

Performance Test - Queues & Buffers KPIs





Performance Test - Queues & Buffers KPIs



IocpChatServer

Performance Test - Performance Analysis

Fan-out & I/O

- ☑ Fan-out: ×{49.75} — 동시 수신자 수의 평균 규모.
  - → fan-out≈×49.75로, 입력 1건당 약 50회 전달이 꾸준히 이루어져 app in/s≈5천 → bc-deliv/s≈25만의 선형 스케일링이 확인됨.
- ☑ I/O send ops (io s/s): {96,663} — WSASend 완료 빈도.
  - → io s/s≈96.7k, bc-deliv/s≈248k → send 1회당 평균 2.6건 전달(SG efficiency)로, fan-out≈×50 환경에서도 시스템 호출 증가가 완만하게 억제됨.
- ☑ SG efficiency: deliveries/I-O = {248,153} / {96,663} ≈ {2.567} → I/O 1회에 평균 {2.567}건 묶음 전송
  - → SG efficiency가 {2.567}로 안정적으로 유지되는 점은, 전송 배치의 묶음 크기·타이밍이 일관되어 send 1회당 평균 {2.567}건 처리가 지속되며, fan-out 확대에도 I/O 호출 증가가 과도하지 않음

Throughput & 대역폭

- ☑ Throughput (app in/s): {4,948.37} / Broadcast (bc-deliv/s): {248,153} / Bandwidth (out): ≈ {11.66} MB/s.
  - → app in/s ≈ app out/s로 내부 큐 정체·드롭 징후 없음. bc-deliv/s ≈ app in/s × fan-out(≈×49.75)가 안정적으로 유지되어 입력 대비 전달량이 선형 확장됨.
  - → out MB/s ≈ 11.66로 관측 구간에서는 링크 사용률이 낮아 네트워크 대역폭이 병목으로 보이지 않으며, 병목이 있다면 NIC보다는 상위 처리 경로(애플리케이션/큐잉/배치/커널 경합 등)에서 나타났을 개연성이 높다.

IocpChatServer

Performance Test - Performance Analysis

JobQueue Health

☑ enq/s = {9,889.28}, exec/s = {9,888.97}, q-peak = {110.21}

➜ enq/s≈9,889과 exec/s≈9,889가 초단위로 밀착해 움직이고, q-peak는 스파이크 후 감쇄하는 패턴이라 큐 정체 신호가 없다고 판단됨

➜ 관측 구간에서 enq가 exec를 지속적으로 상회한 흔적이 없고, q-peak 평균 ≈110은 처리율(≈9.9k/s) 대비 작은 규모라 발생한 버스트도 신속히 해소되는 것으로 보인다.

Send buffers

☑ sendbuf inuse = {15.82} (peak {18.28}), push/s = {39.06}, pop/s = {39.12}

➜ push/s≈pop/s (39.06 vs 39.12)로 거의 일치하고, inuse 평균 ≈15.82가 플래토(peak 18.28만 순간 대응)라 배치·큐잉이 안정적이며 메모리 압력/송신 지연 신호 없음으로 평가됨

➜ 관측 구간에서 push > pop의 지속 패턴이 없고, inuse의 우상향 추세도 미검출되어 송신 백로그 가능성은 낮음.

Memory pool (mpool)

☑ inuse ≈ {69,631.66} (peak {69,742.92}), alloc/s = {274.86}, free/s = {101,641.84}
Reuse ratio: reuse% = 100 × (1 − alloc/s ÷ (alloc/s + free/s)) → {98.97}%

➜ inuse≈69,632(peak 69,743, Δ≈111 ≈0.16%)로 워킹셋이 좁은 범위에서 고정되어 있고, alloc/s 274.86 ≪ free/s 101,641.84로 재사용이 지배적(Reuse ≈ 98.97%)이어서 추가 할당 의존도와 메모리 압력 징후가 낮다.

➜ 부하 상승으로 alloc이 순간 증가하더라도 free가 즉시 우세하고 inuse가 plateau로 복귀하는 패턴으로 보여, 스파이크가 잔류하지 않고 확장 후 정착(plateau) 동작이 유지된다.