📘 TCP/UDP 적용 프로그램 작업내역서_이무준

IP Trade Simulator — 네트워크 구조 구현 작업내역

1. 📡 네트워크 구조 개요

본 프로젝트는 **서버 권위(Server-Authoritative)** 구조를 기반으로 제작되었으며,

- **TCP** : 게임 로직과 관련된 *모든 메시지(채팅, 스냅샷, 상점 정보 등)* 전달
- **UDP** : *게임 플레이에는 관여하지 않고*, 오직 **클라이언트 Ping 측정용**으로 사용

이라는 구조로 설계함.

이 구조는 가벼운 게임 특성상 아래와 같은 점에서 최적이었다.

- 안정성(정합성)을 우선
- 서버의 계산 부담이 매우 낮음
- 클라이언트는 오직 UI만 담당

2. 🧱 TCP 네트워크 프로그래밍 작업내역

✓ 2-1. TCP 기반 채팅 & 스냅샷 시스템 구현

- 서버는 각 클라이언트로부터 입력된 메시지/명령을 수신하고
  이를 가공해 스냅샷 형태로 전체 플레이어에게 배포
- TCP는 메시지가 유실되지 않아 정답 판정/상점 스냅샷에 적합
- 클라이언트는 JSON 기반 DTO로 메시지를 구조화하여 송수신

🔥 2-2. 겪었던 문제 & 해결 과정

🟥 문제 1: JSON 파싱 오류

로그:

```
[UDP] Probe error: JSON parse error: Invalid value.
```

▸ 원인

서버가 보내는 UDP Ping 응답 구조와
클라이언트가 기대하는 JSON 구조가 일치하지 않아
클라이언트에서 파싱 실패 발생.

▸ 해결

- UDP 패킷 포맷을 클라이언트 기대값과 통일
- Ping 응답 DTO를 명확히 구분하도록 서버 코드 수정
- 잘못된 패킷은 무시하고 루프를 유지하는 예외 처리 추가

결과

⚡ Ping 측정 루프가 정상 작동하며, UI에도 Ping이 제대로 반영됨.

🟥 문제 2: 상점 아이템 연출(회전) 중복 실행

스냅샷을 받을 때마다 상점 아이템을 갱신하면서
상점 회전 애니메이션이 매번 재실행되는 버그 발생

▸ 원인

- 기존 코드에서 스냅샷 수신 → 리스트 재구성 → UI 재생성 시
  회전 연출 트리거가 매번 호출됨

▸ 해결

- `ShopManager` 에서 “연출은 UI 최초 생성시에만 실행”하도록 조건 분리
- 스냅샷 적용 시 **데이터만 갱신, 연출은 비활성화**
- `ShopItemView` 컴포넌트 분리 및 책임 분할

결과

🎉 상점이 매 라운드 자연스럽게 갱신되며,
불필요한 애니메이션이 발생하지 않음.

🟥 문제 3: 아이템 아이콘/이름 매칭 오류

아이콘이 랜덤으로 바뀌지 않거나,
아이템 이름이 비정상적으로 “boot” “weapon” 등 단일 단어만 출력되는 현상.

▸ 원인

1. 서버 IconPool과 클라이언트 스프라이트 배열의 인덱스 불일치
2. 이름 생성 규칙(숫자 제거)이
   예상치 못한 문자열까지 줄여버리는 로직 오류

▸ 해결

- 서버 측 `IconPool` 을 스프라이트 파일명과 1:1로 재정의
- 클라이언트 `ShopManager` 의 스프라이트 매핑 배열도 동일한 순서로 재구성
- 이름 생성 로직을
  “숫자만 제거, 문자열은 보존”하는 방식으로 정확히 수정
- `ShopItemView` → 이미지/이름/가격의 책임 분리

결과

🧤 아이템 이미지 & 이름 조합이 시각적으로 정상 출력
재생성 시에도 문제 없음.

3. 🚀 UDP 네트워크 프로그래밍 작업내역

✓ 3-1. UDP 사용 목적 정리

본 프로젝트에서는 UDP를 **게임 플레이용으로 사용하지 않음**
대신 아래 기능에만 사용함:

◆ 클라이언트 Ping 측정

- 클라이언트 → 서버로 작은 UDP Ping 패킷 전송
- 서버는 즉시 같은 값으로 Pong 응답
- 클라이언트는 왕복 시간 기반으로 Ping 계산
- 계산된 Ping을 Player UI에 표시

🔥 3-2. 겪었던 문제 & 해결

🟥 문제 4: Ping이 UI에서 항상 0ms로만 표시

네트워크 루프에서는 Ping이 계산되지만 UI에 반영되지 않음.

▸ 원인

- InfoPanelManager 또는 PlayerView에서 Ping 값 바인딩이 누락됨
- 클라이언트 측에서 Ping 업데이트를 UI에 전달하지 않음

▸ 해결

- Ping 갱신 이벤트를 InfoPanelManager에 전달하도록 구조 리팩터링
- PlayerPrefab에 Ping 텍스트 추가하고 데이터 바인딩
- UI 스레드에서 항상 최신 Ping갱신하도록 동기화

결과

🟢 Ping이 실시간으로 정확하게 UI에 표시됨.

4. 🧩 동기화(Async/Await) 작업내역

✓ 비동기 네트워크 + Unity 메인 스레드 분리 문제

네트워크 통신은 async/await 기반이며,
Unity의 UI는 반드시 메인 스레드에서 실행되어야 하기 때문에
초기 구현에서 UI 갱신 누락/지연 문제 발생 가능성이 있었다.

우리는 이를 다음과 같이 해결함:

해결

- `UnitySynchronizationContext` 를 사용해
  모든 네트워크 콜백 → 메인 스레드 UI 갱신 패턴으로 통일
- UI 갱신 관련 Race Condition 예방
- Ping, 스냅샷, 채팅 등 모든 UI 갱신 처리 안정화

결과

🟦 UI와 네트워크 로직이 완전히 분리된 안정적인 구조 확보.

5. 📄 최종 요약

항목	실제 작업 내용
TCP	채팅, IP 추리, 상점 스냅샷, 플레이어 상태 갱신 등 **모든 핵심 게임 로직 전달**
UDP	**Ping 측정 전용**
구조	서버 권위 모델, 클라 UI 모델
주요 문제	Ping JSON 오류, 상점 연출 중복, 아이콘/이름 매칭 오류, UI 갱신 문제
해결 방식	서버/클라 리팩터링, DTO 정렬, UI 스레드 동기화, 입력 구조 개선
결과	안정적이며 교육용 과제로 적합한 C/S 구조 완성