

gRPC

⚡ gRPC

📌 gRPC란?

gRPC(Google Remote Procedure Call)는 **Remote Procedure Call** 프로토콜의 한 종류로, 다른 컴퓨터에 있는 기능을 마치 자신의 기능인 것처럼 실행할 수 있도록 하는 고성능 프레임워크다.

🎯 핵심 특징

- 언어 독립적: 서로 다른 프로그래밍 언어를 사용하는 서버/클라이언트 간 통신 가능
- 높은 성능: HTTP/2 기반으로 빠른 데이터 전송
- 강력한 생태계: 개발자 커뮤니티와 풍부한 도구/라이브러리 제공

🔧 동작 원리

Stub이라고 불리는 객체가 서버의 부두 역할을 담당하여 원격 호출을 중계한다.

🚧 마이크로서비스 환경에서의 gRPC

현재 많은 서비스들이 여러 마이크로서비스들의 연결로 서버를 구성한다.

📚 도서관 시스템 예시



vs gRPC vs REST API

데이터 형식 비교

JSON의 한계

JSON은 간결하지만, 같은 종류의 요청이 자주 반복되면 불필요한 부분(특히 key 부분)이 계속 전송된다.

Protocol Buffer의 장점

gRPC는 이 문제를 해결하기 위해 **Protocol Buffer**를 사용한다.

```

// books.proto 파일 예시
syntax = "proto3";

message Book {
    int32 id = 1;
    string title = 2;
    string author = 3;
    bool available = 4;
}

service BookService {
    rpc GetBook(BookRequest) returns (Book);
    rpc CreateBook(Book) returns (BookResponse);
}
  
```

🚀 Protocol Buffer의 핵심 이점

특징	JSON	Protocol Buffer
형식	텍스트 기반	바이너리 기반
용량	큰 용량	작은 용량
속도	상대적으로 느림	매우 빠름
스키마	없음	강력한 타입 시스템

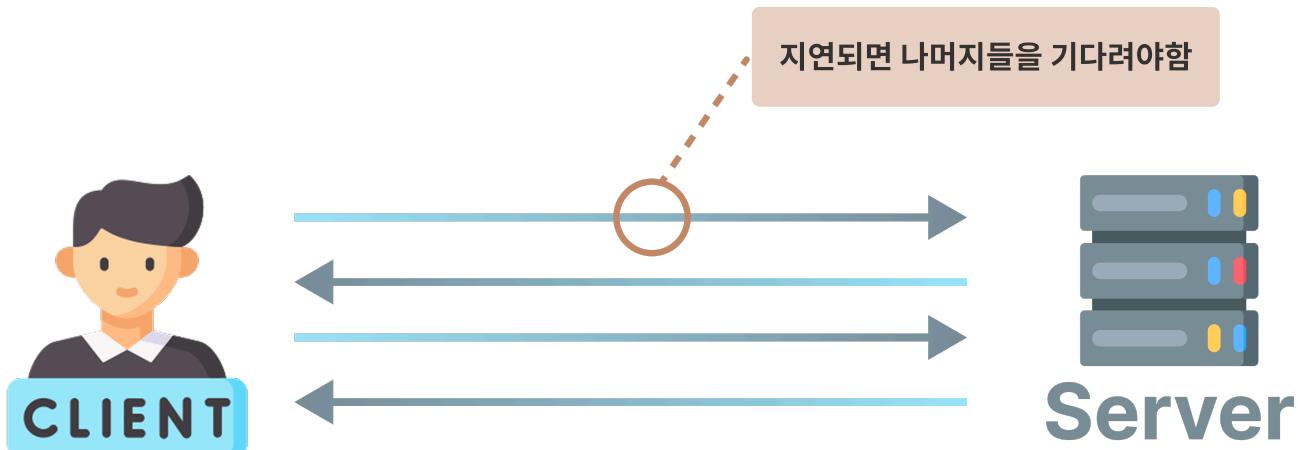
🎯 주요 장점

- 메시지 키 간소화: 반복되는 키 정보 최소화
- 바이너리 직렬화: 텍스트보다 훨씬 작은 용량
- 언어 독립적: 합의된 사양으로 다른 환경 간 매끄러운 소통

🌐 HTTP/2 기반 통신

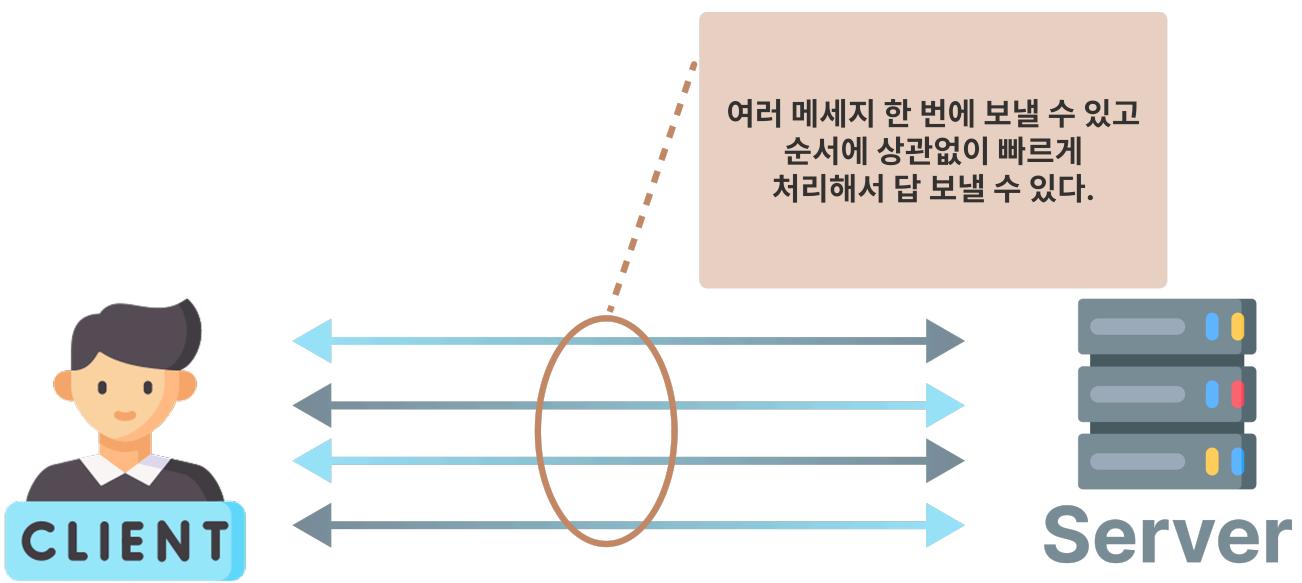
📈 HTTP/1.1 vs HTTP/2

HTTP/1.1 (REST API 주로 사용)



- 순차적 통신: 하나씩 주고받기
- 연결 제한: 동시 처리 어려움

HTTP/2 (gRPC 사용)



- 멀티플렉싱: 양방향 동시 통신
- 스트리밍: 실시간 데이터 교환

⚡ gRPC의 통신 이점

서버와 클라이언트가 보다 빠르게 소통이 가능하여 높은 성능을 제공한다.

🔒 보안 기능

🌐 TLS (Transport Layer Security)

gRPC는 TLS를 통해 암호화하여 전송한다.

🔒 보안 보장사항

- 도청 방지: 다른 누군가 메시지를 훔쳐볼 수 없음
- 변조 방지: 메시지 내용을 변경할 수 없음
- 사칭 방지: 클라이언트 및 서버를 사칭할 수 없음

⚠ 현재 제약사항

🌐 웹 프론트엔드 지원 부족

🚫 브라우저 호환성 문제

- 대부분의 브라우저가 HTTP/2를 지원하지만
- gRPC에 필요한 HTTP/2의 상세 기능은 아직 지원이 부족함
- 웹 프론트엔드에서는 거의 사용되지 않음

🌐 대안 방법

- **gRPC-Web**: 브라우저용 gRPC 프록시
- **REST Gateway**: gRPC를 REST API로 변환

📊 gRPC vs REST API 상세 비교

특징	gRPC	REST API
프로토콜	HTTP/2	HTTP/1.1
데이터 형식	Protocol Buffer (바이너리)	JSON (텍스트)
성능	높음 ⚡	보통
브라우저 지원	제한적 ⚠️	완전 지원 ✅
학습 곡선	가파름 📈	완만함 📉
스트리밍	지원 ✅	제한적
캐싱	어려움	쉬움

🎯 언제 gRPC를 사용할까?

✅ gRPC를 선택해야 하는 경우

- 📱 마이크로서비스 간 통신
- ⚡ 높은 성능이 필요한 서비스
- ⌚ 실시간 스트리밍이 중요한 경우
- 🔒 강력한 타입 시스템이 필요한 경우
- 🌐 다양한 프로그래밍 언어 환경

🚀 실제 사용 사례

- **Netflix**: 마이크로서비스 간 내부 통신
- **Uber**: 실시간 위치 추적 서비스
- **Square**: 결제 시스템의 고성능 통신
- **Dropbox**: 파일 동기화 서비스

✗ REST API를 선택해야 하는 경우

- 🌐 웹 브라우저 직접 통신
- 📱 모바일 앱의 간단한 API
- 🔧 빠른 프로토타이핑

 풍부한 문서화가 필요한 경우

 Public API 제공

gRPC 시작하기

기본 설정 과정

1. **Protocol Buffer 스키마 정의** (.proto 파일)
2. **코드 생성**: 각 언어별 클라이언트/서버 코드 자동 생성
3. **서비스 구현**: 비즈니스 로직 작성
4. **클라이언트 작성**: 서비스 호출 코드 작성

지원 언어

- C++, Java, Python, Go, Ruby, C#, Node.js, PHP, Dart 등

마무리

gRPC는 고성능이 요구되는 마이크로서비스 환경에서 강력한 도구다. 특히 서버 간 통신에서는 REST API 보다 뛰어난 성능을 제공한다.

하지만 웹 프론트엔드 지원 제약과 학습 곡선을 고려하여 프로젝트 특성에 맞는 선택이 중요하다.

내부 서비스 간 통신에는 gRPC를, 공개 API나 웹 서비스에는 REST API를 사용하는 하이브리드 접근법도 좋은 전략이다.

[GraphQL](#)