1주차 예상 질문

질문 1: TCP와 UDP의 차이점을 설명해주세요.

답변:
TCP는 연결지향적이고 신뢰성을 보장하는 프로토콜입니다. 3-way handshake로 연결을 설정하고, 데이터 순서를 보장하며, 오류 검출과 재전송 기능이 있습니다. 반면 UDP는 비연결지향적이고 신뢰성을 보장하지 않지만 속도가 빠릅니다. TCP는 웹 브라우징, 이메일 등에, UDP는 실시간 스트리밍, 온라인 게임 등에 사용됩니다.

질문 2: HTTP/1.1과 HTTP/2의 주요 차이점은 무엇인가요?

답변:
HTTP/1.1은 한 번에 하나의 요청만 처리할 수 있어 HOL 블로킹 문제가 발생합니다.
HTTP/2는 멀티플렉싱을 지원해 하나의 연결로 여러 요청을 동시에 처리할 수 있고, 헤더 압축으로 오버헤드를 줄였습니다. 또한 서버 푸시 기능으로 클라이언트 요청 없이도 리소스를 미리 전송할 수 있어 성능이 크게 향상되었습니다.

질문 3: OSI 7계층 중 3계층(네트워크 계층)의 역할과 주요 프로토콜을 설명해주세요.

답변:
네트워크 계층은 서로 다른 네트워크 간 경로를 찾아 데이터를 전달하는 라우팅 역할을 담당합니다. IP 주소를 기반으로 최적의 경로를 결정하고, 패킷을 목적지까지 전달합니다. 주요 프로토콜로는 IP, ICMP, ARP가 있으며, 라우터가 이 계층에서 동작합니다. 논리적 주소인 IP 주소를 물리적 주소인 MAC 주소로 변환하는 것도 이 계층의 중요한 기능입니다.

질문 4: HTTPS가 HTTP보다 안전한 이유와 동작 원리를 설명해주세요.

답변:
HTTPS는 HTTP에 SSL/TLS 암호화 계층을 추가한 것입니다. 클라이언트와 서버 간 핸드셰이크 과정에서 CA 인증서로 서버를 검증하고, 대칭키와 공개키 암호화를 조합해 데이터를 암호화합니다. 이로써 데이터 기밀성, 무결성, 인증을 보장하며, 중간자 공격을 방지할 수 있습니다. 또한 SEO 순위에도 긍정적 영향을 줍니다.

질문 5: 로드밸런서의 역할과 L4, L7 스위치의 차이점을 설명해주세요.

답변:
로드밸런서는 클라이언트 요청을 여러 서버로 분산시켜 서버 부하를 줄이고 가용성을 높이는 장비입니다. L4 스위치는 전송 계층에서 IP와 포트 정보로 트래픽을 분산하고, L7 스위치는 애플리케이션 계층에서 URL, 쿠키, HTTP 헤더 등 상세한 정보로 더 정교한 분산이 가능합니다. L7이 더 세밀하지만 처리 비용이 높습니다.

질문 6: DNS의 동작 과정을 단계별로 설명해주세요.

답변:
1단계: 브라우저가 로컬 DNS 캐시 확인
2단계: 캐시에 없으면 로컬 DNS 서버에 질의
3단계: 루트 DNS 서버에서 TLD 서버 정보 응답
4단계: TLD 서버에서 권한 있는 네임서버 정보 응답
5단계: 권한 있는 네임서버에서 실제 IP 주소 응답
6단계: 응답받은 IP로 웹 서버 접속
이 과정을 통해 도메인 이름을 IP 주소로 변환합니다.

질문 7: 쿠키와 세션의 차이점과 각각의 보안 이슈를 설명해주세요.

답변:
쿠키는 클라이언트에 저장되는 작은 데이터로 상태 정보를 유지합니다. 세션은 서버에 저장되어 더 안전합니다. 쿠키는 XSS 공격에 취약하므로 HttpOnly, Secure 플래그를 설정해야 하고, 세션은 세션 하이재킹을 방지하기 위해 HTTPS 사용과 세션 타임아웃 설정이 필요합니다. 일반적으로 중요한 정보는 세션에, 사용자 편의 정보는 쿠키에 저장합니다.

질문 8: CDN의 개념과 이점을 설명해주세요.

답변:
CDN은 콘텐츠를 사용자와 가까운 지역의 서버에 캐싱하여 빠르게 전달하는 네트워크입니다. 주요 이점으로는 응답 속도 향상, 서버 부하 분산, 네트워크 대역폭 절약이 있습니다. 특히 이미지, 동영상 등 정적 콘텐츠 전송에 효과적이며, DDoS 공격 완화 효과도 있습니다. 글로벌 서비스에서 지역별 접속 속도를 개선하는 핵심 기술입니다.

질문 9: NAT의 개념과 필요한 이유를 설명해주세요.

답변:
NAT는 사설 IP 주소를 공인 IP 주소로 변환하는 기술입니다. IPv4 주소 부족 문제를 해결하고, 하나의 공인 IP로 여러 기기가 인터넷에 접속할 수 있게 합니다. 또한 내부 네트워크를 외

부로부터 숨겨 보안성을 높입니다. 다만 P2P 통신이나 특정 애플리케이션에서 연결 문제가 발생할 수 있고, 동시 접속 시 성능 저하가 있을 수 있습니다.

질문 10: 싱글톤 패턴의 개념과 장단점, 그리고 의존성 주입으로 해결하는 방법을 설명해주세요.

답변:
싱글톤 패턴은 하나의 클래스에 오직 하나의 인스턴스만 존재하도록 하는 패턴입니다. 주로 데이터베이스 연결 모듈에 사용되며, 인스턴스 생성 비용을 줄이는 장점이 있습니다. 하지만 모듈 간 결합도가 높아지고 TDD 시 독립적인 테스트가 어렵다는 단점이 있습니다. 의존성 주입을 통해 추상화 레이어를 두고 구현체를 주입하면 이런 문제를 해결할 수 있습니다.

질문 11: 팩토리 패턴과 전략 패턴의 차이점과 각각의 사용 사례를 설명해주세요.

답변:
팩토리 패턴은 객체 생성 부분을 추상화하여 상위 클래스가 뼈대를 결정하고 하위 클래스가 구체적인 객체 생성을 담당하는 패턴입니다. 전략 패턴은 객체의 행위를 바꾸고 싶을 때 캡슐화된 알고리즘을 교체하는 패턴입니다. 팩토리 패턴은 바리스타가 다양한 커피를 만드는 것처럼, 전략 패턴은 결제 시 네이버페이, 카카오페이를 선택하는 것처럼 사용됩니다.

질문 12: 옵저버 패턴의 동작 원리와 MVC 패턴에서의 활용을 설명해주세요.

답변:
옵저버 패턴은 주체가 객체의 상태 변화를 관찰하다가 변화가 있을 때 옵저버들에게 알려주는 패턴입니다. 주체는 상태 변화를 감시하는 관찰자이고, 옵저버는 변화에 따라 추가 작업을 수행하는 객체들입니다. MVC 패턴에서 모델의 변경사항이 발생하면 update() 메서드로 뷰에게 알려주고 컨트롤러가 작동하는 방식으로 활용됩니다.

질문 13: 함수형 프로그래밍과 객체지향 프로그래밍의 주요 특징과 차이점을 설명해주세요.

답변:
함수형 프로그래밍은 순수 함수와 고차 함수를 기반으로 하는 선언형 패러다임입니다. 상태를 변경하지 않고 함수의 연속으로 문제를 해결합니다. 객체지향 프로그래밍은 추상화, 캡슐화, 상속성, 다형성의 특징을 가지며 객체 간의 상호작용으로 프로그램을 구성합니다. 함수형은 부작용이 없고 테스트가 쉽지만, 객체지향은 현실 모델링이 직관적이고 코드 재사용성이 높습니다.

질문 14: SOLID 원칙 중 단일 책임 원칙과 개방 폐쇄 원칙을 설명해주세요.

답변:
단일 책임 원칙(SRP)은 모든 클래스는 각각 하나의 책임만 가져야 한다는 원칙입니다. 개방 폐쇄 원칙(OCP)은 기존 코드를 변경하지 않으면서도 기능을 쉽게 확장할 수 있어야 한다는 원칙입니다. 예를 들어 결제 시스템에서 새로운 결제 방식을 추가할 때, 기존 코드는 수정하지 않고 새로운 결제 클래스만 추가하여 확장하는 것이 OCP를 따르는 것입니다.

질문 15: 프록시 패턴의 개념과 실제 활용 사례를 설명해주세요.

답변:
프록시 패턴은 대상 객체에 접근하기 전에 그 접근을 제어하거나 기능을 추가하는 패턴입니다. 접근을 필터링하거나 수정할 수 있는 계층을 제공합니다. 실제로는 웹에서 프록시 서버가 대표적인 예시입니다. nginx가 Node.js 앞단에서 로드밸런싱을 하거나, CDN이 원본 서버 앞에서 캐싱 역할을 하는 것, 그리고 자바스크립트의 Proxy 객체로 Vue.js에서 반응형 시스템을 구현하는 것도 프록시 패턴의 활용입니다.

질문 16: MVC, MVP, MVVM 패턴의 차이점을 설명해주세요.

답변:
MVC는 Model-View-Controller로 컨트롤러가 사용자 입력을 처리하고 모델과 뷰를 연결합니다. MVP는 컨트롤러가 프레젠터로 바뀌어 뷰와 일대일 관계를 가지며 더 강한 결합을 보입니다. MVVM은 컨트롤러가 뷰모델로 바뀌고 데이터 바인딩을 통해 뷰와 양방향 연결됩니다. Vue.js가 MVVM 패턴의 대표적인 예시로, watch나 computed로 반응형 값을 쉽게 구축할 수 있습니다.

네트워크 관련 추가 정보

TCP/UDP 차이점 (질문 1)

- TCP 헤더는 20바이트, UDP 헤더는 8바이트로 UDP가 더 가벼움
- TCP는 흐름제어, 혼잡제어 기능 제공
- 실무에서 HTTP/HTTPS는 TCP, DNS 조회는 UDP 사용

HTTP/1.1 vs HTTP/2 (질문 2)

- HTTP/2는 바이너리 프레임 사용으로 파싱 효율성 향상
- 요청 우선순위 설정 가능
- HTTP/2는 HTTPS 위에서만 동작 (브라우저 정책)

로드밸런서 (질문 5)

- 로드밸런싱 알고리즘: Round Robin, Least Connection, IP Hash 등
- AWS에서는 ALB(L7), NLB(L4), CLB(Classic) 제공
- 헬스체크 주기와 임계값 설정으로 장애 서버 자동 제외

DNS (질문 6)

- DNS 레코드 타입: A(IPv4), AAAA(IPv6), CNAME(별칭), MX(메일) 등
- TTL(Time To Live)로 캐시 유지 시간 조절
- DNS over HTTPS(DoH)로 보안성 강화 가능

CDN (질문 8)

- Edge Server, Origin Server 개념
- CloudFlare, AWS CloudFront 등 실제 서비스
- 캐시 전략: TTL 설정, Cache-Control 헤더 활용

디자인패턴/프로그래밍 관련 추가 정보

싱글톤 패턴 (질문 10)

- 멀티스레드 환경에서의 Thread-Safe 구현 (Double-Checked Locking)
- 스프링 프레임워크의 기본 빈 스코프가 싱글톤
- 테스트 시 Mock 객체 사용의 어려움

팩토리 패턴 (질문 11)

- Abstract Factory vs Factory Method 차이점
- Spring Framework의 BeanFactory가 팩토리 패턴 사용
- 확장성과 유지보수성 향상 효과

옵저버 패턴 (질문 12)

- JavaScript의 EventListener가 옵저버 패턴
- RxJS의 Observable/Observer 개념

- 메모리 누수 방지를 위한 구독 해제 중요성

함수형 vs 객체지향 (질문 13)

- JavaScript의 map, filter, reduce가 함수형 프로그래밍 예시
- 불변성(Immutability) 개념과 중요성
- 함수형은 병렬처리에 유리, 객체지향은 대규모 시스템 설계에 유리

SOLID 원칙 (질문 14)

- 의존성 역전 원칙(DIP): 고수준 모듈이 저수준 모듈에 의존하지 않아야 함
- 인터페이스 분리 원칙(ISP): 클라이언트가 사용하지 않는 인터페이스에 의존하면 안됨
- 실무에서 Clean Architecture, Hexagonal Architecture에 적용

MVC/MVP/MVVM (질문 16)

- React는 MVC의 V에 해당하는 라이브러리
- Angular는 MVVM 패턴 기반
- 단방향 데이터 플로우 vs 양방향 데이터 바인딩의 장단점