네트워크

네트워크는 서로 연결된 여러 장치들이 정보를 주고받을 수 있도록 해주는 시스템입니다.

1. 네트워크의 기본 구성 요소

- 장치: 네트워크에 연결된 컴퓨터, 스마트폰, 프린터, 서버 등이 장치에 해당합니다. 이러한 장치들은 네트워크를 통해 서로 소통하고 데이터를 주고받습니다.
- 연결 매체: 장치들을 연결해주는 물리적 매체입니다. 일반적으로 케이블(예: 이더넷 케이블)이나 무선 기술(예: Wi-Fi)을 통해 연결됩니다.

2. 이더넷과 인터넷

이더넷과 인터넷은 각각 다른 네트워크 유형입니다. 이들 사이의 차이를 이해하면 네트워크가 어떻게 구성되는지에 대한 기초를 확립할 수 있습니다.

- 이더넷:
    - 이더넷은 주로 가정이나 사무실에서 사용되는 네트워크 기술입니다. 이더넷 네트워크는 동일한 물리적 네트워크 환경 내에서 장치들이 서로 연결되어 데이터를 교환할 수 있게 합니다.
    - 이더넷 네트워크는 물리적으로 연결된 장치들 간의 통신을 관리합니다. 예를 들어, 집안의 컴퓨터와 프린터가 서로 연결되어 파일을 공유하거나 프린터를 사용하는 경우 이더넷을 사용합니다.
- 인터넷:
    - 인터넷은 전 세계의 네트워크를 연결하는 거대한 네트워크입니다. 인터넷은 다양한 네트워크를 연결하여 정보를 전 세계 어디서든 주고받을 수 있게 합니다.
    - 인터넷은 여러 네트워크 집단을 하나로 연결하여, 이메일을 보내거나 웹사이트에 접속하는 등의 활동을 가능하게 합니다.

3. IP 주소와 도메인 이름

네트워크에서 각 장치는 고유한 IP 주소를 가지고 있습니다. IP 주소는 네트워크에서 장치를 식별하는 숫자 주소입니다. 예를 들어, 192.168.1.2와 같은 IP 주소는 특정 장치를 식별합니다.

도메인 이름:

- 도메인 이름은 사람이 이해하기 쉬운 웹 주소입니다. 예를 들어, www.example.com은 특정 웹사이트를 가리킵니다.
- DNS (Domain Name System)는 도메인 이름을 IP 주소로 변환하여 웹사이트에 접근할 수 있게 합니다. 예를 들어, 사용자가 www.example.com을 입력하면 DNS는 이를 192.0.2.1이라는 IP 주소로 변환하여 웹사이트에 연결합니다.

4. 네트워크 계층

네트워크는 여러 계층으로 구성되며, 각 계층은 특정 역할을 담당합니다.

- 데이터 링크 계층:
  - 이 계층은 물리적으로 연결된 장치들 간의 데이터 전송을 관리합니다. 이더넷이 이 계층에서 작동합니다.
- 네트워크 계층:
  - 이 계층은 데이터를 목적지까지 전달하기 위해 경로를 결정합니다. IP 주소를 사용하여 장치를 식별하고, 데이터가 올바른 경로로 이동하도록 합니다.
- 전송 계층:
  - 이 계층은 데이터 전송의 신뢰성을 보장합니다. 데이터가 제대로 전달되도록 확인하고, 오류가 있으면 수정합니다. TCP와 UDP가 이 계층의 주요 프로토콜입니다.
- 응용 계층:
  - 이 계층은 사용자와 직접 상호작용하는 서비스와 프로토콜을 제공합니다. 예를 들어, HTTP(웹 페이지 보기), FTP(파일 전송), SMTP(이메일 전송) 등이 이 계층에 해당합니다.

주요 네트워크 프로토콜

네트워크에서 데이터 전송과 통신을 위해 여러 가지 프로토콜이 사용됩니다. 각 프로토콜은 특정 기능과 목적을 가지고 있습니다.

- HTTP (Hypertext Transfer Protocol):
    - 웹 브라우징에 사용되는 프로토콜로, 클라이언트(브라우저)와 서버 간의 하이퍼텍스트 전송을 관리합니다. 기본 포트는 80입니다.
- HTTPS (Hypertext Transfer Protocol Secure):
    - HTTP에 SSL/TLS를 적용하여 보안을 강화한 프로토콜입니다. 기본 포트는 443입니다. 암호화를 통해 데이터 전송의 보안을 보장합니다.
- FTP (File Transfer Protocol):
    - 파일 전송을 위한 프로토콜로, 포트 21을 사용합니다. 파일의 업로드 및 다운로드를 지원하며, 제어 연결과 데이터 연결을 분리합니다.
- SMTP (Simple Mail Transfer Protocol):
    - 이메일 전송을 위한 프로토콜로, 포트 25를 사용합니다. 이메일을 서버로 전송하는 데 사용되며, 수신 프로토콜로는 POP3와 IMAP이 있습니다.
- TCP (Transmission Control Protocol):
    - 연결 지향 프로토콜로, 데이터 전송의 신뢰성을 보장합니다. 패킷의 순서 보장, 오류 검출 및 수정, 흐름 제어 기능을 제공합니다.
- UDP (User Datagram Protocol):
    - 비연결 지향 프로토콜로, 빠른 데이터 전송을 중시합니다. 신뢰성 보장 기능이 없으므로 실시간 스트리밍과 같은 애플리케이션에 사용됩니다.

포트와 프로토콜의 상호작용

포트는 네트워크에서 특정 프로세스나 서비스를 식별하는 숫자입니다. IP 주소가 장치를 식별한다면, 포트 번호는 장치에서 실행 중인 특정 애플리케이션을 식별합니다.

- 포트 번호의 범위:
    - 0-1023 (Well-Known Ports): HTTP(80), HTTPS(443), FTP(21), SMTP(25) 등과 같은 표준 서비스에 할당됩니다.
    - 1024-49151 (Registered Ports): 특정 애플리케이션에 등록된 포트입니다. 예를 들어, MySQL 데이터베이스 서버는 포트 3306을 사용합니다.
    - 49152-65535 (Dynamic/Private Ports): 클라이언트 측에서 임시로 할당하여 사용됩니다.

포트와 프로토콜 사용 예시:

1. 웹 브라우징:
    - 사용자가 웹 브라우저에서 http://www.example.com을 입력합니다.
    - 브라우저는 DNS 서버에 www.example.com의 IP 주소를 요청합니다.
    - 브라우저는 서버의 IP 주소와 포트 80(HTTP)으로 TCP 연결을 설정합니다.
    - 브라우저는 서버에 HTTP GET 요청을 보냅니다.
    - 서버는 요청을 처리하고, HTTP 응답을 브라우저로 보냅니다. 응답은 TCP를 통해 전송되며, 브라우저는 이를 사용자에게 표시합니다.
2. 파일 전송 (FTP):
    - 클라이언트는 FTP 클라이언트를 통해 서버의 IP 주소와 포트 21에 연결합니다.
    - FTP 클라이언트는 서버의 IP 주소와 포트 21로 TCP 연결을 설정합니다.
    - 클라이언트는 사용자 이름과 비밀번호를 사용하여 FTP 서버에 로그인합니다.
    - 클라이언트는 파일 업로드 또는 다운로드를 요청합니다.
    - FTP 서버는 요청된 파일을 TCP를 통해 클라이언트로 전송합니다.

네트워크 보안 기초

SSL (Secure Sockets Layer)과 TLS (Transport Layer Security)는 네트워크 통신의 보안을 강화하기 위해 사용됩니다. 이들은 데이터를 암호화하고, 인증하며, 데이터의 무결성을 보장합니다.

- SSL/TLS의 역할:
    - 암호화: 데이터가 도청되는 것을 방지합니다.
    - 인증: 통신 상대방이 신뢰할 수 있는지를 확인합니다.
    - 데이터 무결성: 데이터가 전송 중에 변경되지 않았음을 보장합니다.
- 작동 원리:
    - 클라이언트와 서버 간의 SSL/TLS 핸드셰이크를 통해 보안 연결이 설정됩니다.
    - 클라이언트는 서버의 인증서를 확인하고, 대칭 키 암호화를 위한 세션 키를 교환합니다.
    - 이후 데이터 전송은 암호화된 형태로 이루어집니다.
- HTTPS:
    - HTTPS는 HTTP에 SSL/TLS를 적용하여 보안을 강화한 프로토콜입니다. 기본 포트는 443입니다. 데이터를 암호화하여 보안을 강화합니다.

소프트웨어 정의 네트워크 (SDN)

소프트웨어 정의 네트워크(SDN)는 네트워크의 제어 평면(Control Plane)과 데이터 평면(Data Plane)을 분리하여 네트워크를 중앙에서 소프트웨어적으로 제어하고 관리하는 기술입니다. 이는 네트워크 관리와 구성을 더 유연하고 효율적으로 할 수 있도록 돕습니다.

SDN의 주요 개념

- **제어 평면(Control Plane)**:
    - 제어 평면은 네트워크의 라우팅과 트래픽 흐름을 제어하는 역할을 합니다. SDN에서는 제어 평면이 중앙의 소프트웨어 컨트롤러에 의해 관리됩니다. 이 컨트롤러는 네트워크의 모든 트래픽 흐름을 결정하고, 각 네트워크 장비(스위치, 라우터 등)에 필요한 명령을 전달합니다.
    - 예를 들어, SDN 컨트롤러는 트래픽을 특정 경로로 우회시키거나, 네트워크의 부하를 분산시키는 작업을 수행할 수 있습니다.
- **데이터 평면(Data Plane)**:
    - 데이터 평면은 실제로 데이터를 전송하는 역할을 합니다. SDN에서는 데이터 평면이 다양한 스위치와 라우터를 통해 구현됩니다. 이 장비들은 중앙 컨트롤러로부터 받은 명령에 따라 데이터를 전달하고 처리합니다.
    - 예를 들어, SDN 스위치는 컨트롤러의 지시에 따라 패킷을 적절한 포트로 전송합니다.

SDN의 장점:

- **유연성**: 네트워크의 설정을 소프트웨어적으로 조정할 수 있어 새로운 서비스를 빠르게 배포할 수 있습니다.
- **효율성**: 중앙에서 네트워크를 관리하므로 네트워크 자원의 최적화와 관리가 용이합니다.
- **자동화**: 네트워크 설정 및 관리를 자동화하여 운영 비용을 절감할 수 있습니다.

네트워크 기능 가상화 (NFV)

네트워크 기능 가상화(Network Functions Virtualization, NFV) 는 네트워크 장비의 기능을 하드웨어에서 소프트웨어로 가상화하여, 일반 서버에서 실행하도록 하는 기술입니다. NFV는 네트워크의 유연성과 확장성을 높이기 위해 사용됩니다.

NFV의 주요 개념

- **가상화된 네트워크 기능 (VNF)**:
    - VNF는 방화벽, 라우터, 스위치 등의 네트워크 기능을 소프트웨어로 구현하여 가상 머신(VM) 또는 컨테이너에서 실행합니다. 이는 특정 하드웨어 장비에 의존하지 않고, 표준 서버에서 다양한 네트워크 기능을 수행할 수 있게 합니다.
    - 예를 들어, NFV를 통해 소프트웨어 기반의 가상 방화벽을 구현하여 네트워크 트래픽을 필터링하고 보호할 수 있습니다.
- **NFV 관리 및 오케스트레이션 (MANO)**:
    - MANO는 가상화된 네트워크 기능을 관리하고 조정하는 소프트웨어 플랫폼을 말합니다. 이를 통해 네트워크의 자원을 효율적으로 관리하며, VNF의 배치, 구성, 모니터링 및 확장을 자동화합니다.
    - 예를 들어, MANO 플랫폼은 네트워크의 리소스를 실시간으로 모니터링하고 필요에 따라 가상 네트워크 기능을 동적으로 조정할 수 있습니다.

NFV의 장점:

- **비용 절감**: 하드웨어 장비의 구입과 유지 관리 비용을 절감할 수 있습니다.
- **확장성**: 네트워크 기능을 소프트웨어로 가상화하여 필요에 따라 쉽게 확장할 수 있습니다.
- **빠른 배포**: 새로운 서비스와 기능을 신속하게 배포할 수 있습니다.

클라우드 네트워킹 (AWS, Azure 등)

클라우드 네트워킹은 클라우드 서비스 제공자가 제공하는 네트워크 리소스를 사용하여 애플리케이션과 서비스를 연결하고 관리하는 기술입니다. 이를 통해 물리적 네트워크 장비 없이도 네트워크를 구성할 수 있습니다.

주요 클라우드 서비스 제공자

- **AWS (Amazon Web Services)**:
    - **AWS VPC (Virtual Private Cloud)**: 사용자가 자신의 가상 네트워크를 설정할 수 있게 해주는 서비스입니다. VPC는 공용 및 사설 서브넷을 포함하여, 네트워크 환경을 세밀하게 제어할 수 있습니다.
    - **AWS ELB (Elastic Load Balancer)**: 애플리케이션의 트래픽을 여러 서버에 분산시켜 부하를 균등하게 나눕니다. 이를 통해 애플리케이션의 가용성과 성능을 높일 수 있습니다.
    - **AWS Route 53**: DNS 웹 서비스로, 도메인 이름을 IP 주소로 변환하여 클라우드 기반의 애플리케이션에 접근할 수 있도록 합니다.
- **Azure (Microsoft Azure)**:
    - **Azure Virtual Network**: 클라우드 리소스를 격리된 네트워크 환경에서 운영할 수 있게 해주는 서비스입니다. 사용자는 IP 주소 범위, 서브넷, 라우팅 등을 정의할 수 있습니다.
    - **Azure Load Balancer**: 트래픽을 여러 VM에 분산시켜 애플리케이션의 고가용성을 보장합니다. Azure Load Balancer는 네트워크 레벨에서 작동하여 높은 성능을 제공합니다.

클라우드 네트워킹의 장점:

- **유연성**: 네트워크 인프라를 필요에 따라 쉽게 조정할 수 있습니다.
- **확장성**: 클라우드 기반의 자원을 필요에 따라 확장하거나 축소할 수 있습니다.
- **비용 효율성**: 물리적 장비의 구입과 유지 관리 비용이 없으며, 사용한 만큼만 비용을 지불할 수 있습니다.

API의 이해

API (Aqpplication Programming Interface)는 소프트웨어 간의 상호작용을 가능하게 하는 인터페이스입니다. API를 이해하기 위해 자동차와 고속도로를 비유로 설명할 수 있습니다.

자동차와 고속도로 비유

- **고속도로**: 고속도로는 여러 차량이 안전하고 효율적으로 이동할 수 있도록 설계된 도로입니다. 고속도로는 차량들이 목적지까지 빠르고 원활하게 도달할 수 있는 경로를 제공합니다.
- **자동차**: 자동차는 사람이나 물건을 운송하기 위한 수단입니다. 자동차는 고속도로 위에서 특정 목적지까지 이동할 수 있습니다.
- **API**: API는 고속도로의 도로와 자동차의 관계를 나타냅니다. API는 소프트웨어가 서로 통신할 수 있도록 하는 규칙과 도구입니다. 소프트웨어는 API를 사용하여 서로 데이터를 주고받거나 기능을 호출할 수 있습니다.

API의 작동 원리

1. **요청(Request)**:
    - 사용자가 자동차를 이용하여 고속도로를 통해 특정 목적지로 이동하려고 합니다. 이는 소프트웨어가 API를 통해 요청을 보내는 것과 유사합니다.
    - 예를 들어, 웹 애플리케이션이 서버에 데이터를 요청할 때 API를 사용하여 요청을 보냅니다.
2. **처리(Process)**:
    - 자동차는 고속도로의 규칙에 따라 주행합니다. 마찬가지로, 소프트웨어는 API의 규칙에 따라 요청을 처리합니다.
    - 예를 들어, API가 요청된 데이터를 처리하고 필요한 작업을 수행합니다.
3. **응답(Response)**:
    - 자동차가 목적지에 도착하면, 사용자는 원하는 결과를 얻게 됩니다. 소프트웨어도 API를 통해 처리된 결과를 응답으로 받습니다.
    - 예를 들어, 서버가 데이터베이스에서 정보를 조회한 후, 그 결과를 웹 애플리케이션에 응답으로 보냅니다.