DevOps – Fundamentos de Redes – Parte 1 - Dia 4

3 – INTRODUÇÃO AO TCP/IP

O modelo TCP/IP é um conjunto de protocolos de comunicação usado para interligar dispositivos em redes, como a internet. Ele é dividido em **quatro camadas**, cada uma com funções específicas:

📌 1. Camada de Aplicação (L7 – Application Layer)

- Essa camada é onde ocorrem as interações entre o usuário e a rede.
- Engloba os protocolos e aplicações que usam a rede para enviar e receber dados.
- Exemplos de protocolos:
  - **HTTP/HTTPS** (navegação na web)
  - **FTP** (transferência de arquivos)
  - **SMTP, POP3, IMAP** (e-mails)
  - **DNS** (resolução de nomes)

✔ **Responsabilidade:** Facilitar a comunicação entre aplicações e a rede.

📌 2. Camada de Transporte (L4 – Transport Layer)

- Garante a entrega confiável ou não dos dados entre dispositivos.
- Controla a fragmentação, envio e recebimento dos pacotes.
- Dois principais protocolos:
  - **TCP (Transmission Control Protocol):** Confiável, com controle de erros, usado em e-mails e navegação.
  - **UDP (User Datagram Protocol):** Mais rápido, mas sem garantia de entrega, usado em streaming e VoIP.

✔ **Responsabilidade:** Gerenciar a conexão entre origem e destino, garantindo ou não a confiabilidade.

📌 3. Camada de Rede (L3 – Network Layer)

- Determina a melhor rota para o envio dos pacotes.
- Usa endereços IP para identificar remetente e destinatário.
- Responsável pelo roteamento e encaminhamento dos pacotes.
- Protocolos importantes:
  - **IP (Internet Protocol):** Define os endereços e roteia pacotes.
  - **ICMP (Internet Control Message Protocol):** Diagnóstico de rede (ping).
  - **ARP (Address Resolution Protocol):** Converte IP em MAC.

✔ **Responsabilidade:** Roteamento e endereçamento dos pacotes na rede.

---

4. Camada de Enlace (L2 – Data Link Layer)

- Garante a transmissão de dados entre dispositivos conectados fisicamente.
- Trabalha com endereços **MAC** para comunicação dentro de uma rede local.
- Controla erros na transmissão dentro de um mesmo segmento de rede.
- Protocolos e tecnologias:
  - **Ethernet (IEEE 802.3)**
  - **Wi-Fi (IEEE 802.11)**
  - **PPP (Point-to-Point Protocol)**
  - **Switches, VLANs**

Responsabilidade: Comunicação dentro da rede local, usando endereços físicos (MAC).

Resumo das Funções por Camada

Camada	Função	Exemplos
Aplicação (L7)	Interface para os usuários	HTTP, FTP, SMTP, DNS
Transporte (L4)	Garante a entrega dos dados	TCP, UDP
Rede (L3)	Define rotas e endereços IP	IP, ICMP, ARP
Enlace (L2)	Comunicação dentro da rede local	Ethernet, Wi-Fi, Switches

4 – A VIDA DE UM PACOTE

O **ciclo de vida de um pacote** em uma rede TCP/IP descreve todas as etapas que ele percorre, desde a origem até o destino. Isso envolve encapsulamento, roteamento, transmissão e decapsulamento. Vamos ver isso em detalhes:

---

1. Criação do Pacote (Camada de Aplicação - L7)

Tudo começa quando um usuário ou uma aplicação deseja enviar dados. Por exemplo:

- Um navegador acessa um site via **HTTP**.
- Um programa envia um e-mail via **SMTP**.
- Um usuário faz download de um arquivo via **FTP**.

Nesse momento, os dados são formatados de acordo com o protocolo de aplicação (exemplo: HTTP, DNS, FTP).

---

2. Encapsulamento (Camada de Transporte - L4)

A camada de transporte recebe os dados da aplicação e adiciona informações de controle:

- Se o protocolo for **TCP**, ele adiciona:
  - Número de sequência e reconhecimento.
  - Controle de erros (checksum).
  - Garantia de entrega.
- Se o protocolo for **UDP**, ele apenas adiciona:
  - Porta de origem e destino.
  - Checksum (opcional).

💡 **Exemplo:**

- O navegador envia uma solicitação **HTTP**.
- O protocolo **TCP** adiciona as informações de controle.

📡 **3. Endereçamento e Roteamento (Camada de Rede - L3)**

Nesta etapa, os pacotes são preparados para viajar na rede:

1. O protocolo **IP** adiciona:
   - **Endereço IP de origem** (do remetente).
   - **Endereço IP de destino** (para onde o pacote vai).
   - **TTL (Time to Live):** Limita o número de roteadores pelos quais o pacote pode passar antes de ser descartado.
2. Se o destino estiver **fora da rede local**, o pacote é enviado para o **gateway (roteador)**.

🔗 **4. Transmissão Física (Camada de Enlace - L2)**

Agora, os pacotes são convertidos em **quadros (frames)** para serem transmitidos:

- O protocolo de enlace (como **Ethernet ou Wi-Fi**) adiciona:
  - **Endereço MAC de origem** (placa de rede do remetente).
  - **Endereço MAC de destino** (pode ser o roteador, se for necessário roteamento).
  - Verificação de erros (CRC - Cyclic Redundancy Check).

💡 **Se for uma rede local (LAN):** O pacote pode ir direto ao destino.
💡 **Se for para outra rede:** Ele passa pelo roteador.

🚦 **5. Roteamento (Passagem por Múltiplos Roteadores)**

Se o destino está em outra rede, o pacote passa por diversos roteadores:

1. O roteador **lê o endereço IP de destino**.
2. **Decide a melhor rota** e envia o pacote para o próximo roteador.
3. A cada roteador, o **endereço MAC muda**, mas o **endereço IP permanece**.

⚠️ Se o TTL chegar a 0 antes de alcançar o destino, o pacote é descartado e um erro **ICMP "Time Exceeded"** é enviado ao remetente.

👁️ 6. Recepção no Destino

Quando o pacote chega ao destino final:

1. A **Camada de Enlace (L2)** verifica se o MAC de destino corresponde ao dispositivo correto.
2. A **Camada de Rede (L3)** verifica se o IP é o certo.
3. A **Camada de Transporte (L4)** reagrupa os pacotes (se for TCP).
4. A **Camada de Aplicação (L7)** lê os dados e exibe para o usuário.

💡 **Exemplo:** O navegador recebe a resposta do site e exibe a página.

⮂ 7. Resposta do Servidor

Se a comunicação precisa de uma resposta (como um site carregando), o servidor cria um **novo pacote** e envia de volta para o cliente, repetindo o mesmo ciclo.

🔄 Resumo do Ciclo de Vida de um Pacote

Etapa	Ação	Camada do Modelo TCP/IP
1. Geração dos dados	O usuário usa um app (ex.: navegador, e-mail)	Aplicação (L7)
2. Encapsulamento	TCP/UDP adiciona portas e controle	Transporte (L4)
3. Endereçamento IP	IP define remetente e destino	Rede (L3)
4. MAC e transmissão	Ethernet ou Wi-Fi adiciona MAC e envia	Enlace (L2)
5. Roteamento	Pacote atravessa a internet via roteadores	Rede (L3)
6. Entrega	O dispositivo final lê os dados e os repassa ao app	Aplicação (L7)
7. Resposta	O servidor gera um novo pacote e envia de volta	Recomeça o ciclo

---

👁️ Exemplo Prático: Acessando um Site

1. Você digita **{HYPERLINK "http://www.google.com" \t "_new"}** no navegador (**Camada de Aplicação**).
2. O **TCP** cria pacotes para a requisição HTTP (**Camada de Transporte**).
3. O **IP** define que os pacotes vão para o servidor do Google (**Camada de Rede**).
4. O **Wi-Fi ou Ethernet** envia os pacotes pela rede (**Camada de Enlace**).
5. O **roteador e os servidores da internet** encaminham os pacotes.
6. O servidor do Google recebe, processa e responde.
7. O ciclo se repete até o site ser carregado.

🔷 5 - HTTP MESSAGE (REQUEST VS RESPONSE)

O protocolo HTTP funciona com **mensagens de requisição (request)** e **mensagens de resposta (response)**.

📩 Request (Requisição)

Quando um cliente (como um navegador) quer acessar um recurso, ele envia uma requisição HTTP para um servidor.

Estrutura da Requisição:

```
GET /index.html HTTP/1.1
Host: www.exemplo.com
User-Agent: Mozilla/5.0
Accept: text/html
```

- **Linha de requisição**: Contém o método (GET), o recurso (/index.html) e a versão do HTTP (HTTP/1.1).
- **Cabeçalhos**: Informações extras como o **host**, **user-agent** e **tipos de conteúdo aceitos**.
- **Corpo (Opcional)**: Usado em POST e PUT para enviar dados.

📩 Response (Resposta)

O servidor recebe a requisição e responde com um código de status e os dados solicitados.

Exemplo de Resposta:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 342

<html>...</html>
```

- **Linha de status**: Mostra o código HTTP (**200 OK**, **404 Not Found**, etc.).
- **Cabeçalhos**: Tipo de conteúdo, tamanho da resposta, etc.
- **Corpo**: O conteúdo real da página ou recurso solicitado.

💡 Resumindo:

- O **cliente envia um request** para o servidor.
- O **servidor processa e responde** com um status e os dados solicitados.

🔷 6 - DEVELOPER MOZILLA: A BÍBLIA DO HTTP

A **Mozilla Developer Network (MDN)** é um dos melhores recursos para aprender sobre HTTP e desenvolvimento web.

🔗 Acesse aqui👉 {HYPERLINK "https://developer.mozilla.org/pt-BR/docs/Web/HTTP" \t "_new"}

📌 O que você encontra lá?

- Estrutura das requisições HTTP
- Códigos de status HTTP
- Métodos (GET, POST, PUT, DELETE, etc.)
- Cabeçalhos HTTP (User-Agent, Content-Type, Authorization, etc.)
- Segurança e HTTPS

Se você quer entender HTTP profundamente, a **MDN é sua melhor 📖fo🔥nte**.

🔷 7 - HTTP PERSISTENT VS NON-PERSISTENT

O HTTP pode ser **persistente** ou **não persistente**, dependendo de como ele gerencia conexões.

🌐 HTTP Não Persistente

Cada requisição HTTP abre uma **nova conexão TCP** e fecha após receber a resposta.
💡 **Problema:** Ineficiente, pois cria conexões repetidamente.

⚡ HTTP Persistente

Mantém a **conexão aberta** para múltiplas requisições, economizando tempo.
💡 **Vantagem:** Reduz latência, pois não precisa abrir uma nova conexão toda vez.

📌 **HTTP/1.0** → Não persistente por padrão.
📌 **HTTP/1.1 e HTTP/2** → Usa conexões persistentes por padrão (Keep-Alive).

🔷 8 - COMO INSTALAR O WIRESHARK

O **Wireshark** é uma ferramenta para capturar e analisar pacotes de rede.

🔷 Instalar no Windows

1. Baixe o instalador no site ofici👉al https://www.wireshark.org/download.html
2. Execute o instalador (.exe) e siga as instruções.
3. Selecione a opção para instalar o **Npcap** (driver necessário).
4. Finalize a instalação e abra o Wireshark.

🔷 Instalar no Linux (Debian/Ubuntu)

sudo apt update
sudo apt install wireshark -y

💡 **Opcional:** Para rodar sem sudo, adicione seu usuário ao grupo

wireshark: sudo usermod -aG wireshark $USER

Depois, **reinicie o computador**.

🔷 Instalar no macOS

Se tiver o **Homebrew**, rode:

brew install wireshark

Agora você pode capturar pacotes e analisar o tráfego da r🌲ede!

🔷 9 - O QUE É E PARA QUE SERVE UM ARQUIVO HAR?

O **HAR (HTTP Archive)** é um arquivo que **registra todas as requisições HTTP** feitas pelo navegador durante a navegação em um site.

🔧 Para que serve?

- Depuração de problemas de carregamento.
- Análise de performance da página.
- Identificação de erros HTTP.
- Diagnóstico de tempo de resposta de APIs.

📥 Como gerar um arquivo HAR no Chrome?

1. **Abra o DevTools**: Pressione F12 ou Ctrl + Shift + I.
2. Vá até a aba **"Network"**.
3. **Recarregue a página** (F5) para capturar as requisições.
4. Clique com o botão direito e selecione **"Save as HAR with Content"**.

💡 **Dica:** Você pode abrir esse arquivo no Wireshark ou ferramentas online para analisar o tráfego da rede.

📝 Resumo Rápido

🔢	Tópico	Explicação
5	**HTTP Request vs Response**	Cliente envia request, servidor responde com status e dados.
6	**Developer Mozilla**	Melhor fonte para aprender HTTP.
7	**HTTP Persistente vs Não Persistente**	Conexão única (Keep-Alive) vs conexão nova a cada requisição.
8	**Como instalar Wireshark**	Ferramenta para capturar pacotes de rede.
9	**Arquivo HAR**	Registro de requisições HTTP para depuração.