Redes de Computadores Internet: UDP e TCP - parte1 - Valor 3 pontos da 1º Avaliação

Hector José Rodrigues Salgueiros (Matrícula: 20219003176)
Universidade Federal do Piauí - UFPI
Campus Senador Helvídio Nunes Barros - CSHNB
Prof.: Rayner Gomes

Outubro: 2024

1 Faça um estudo sobre a evolução da Internet desde sua concepção até os dias dehoje. Procure o conceito de patch dentro deste contexto, quais os principais patches encontrados.

A Internet evoluiu desde seus primeiros dias como ARPANET nos anos 1960 até se tornar a rede global que conhecemos hoje. Inicialmente desenvolvida para conectar instituições de pesquisa e o governo dos EUA, ela introduziu inovações como a comutação de pacotes e o protocolo TCP/IP, fundamentais para sua arquitetura. Nos anos 1980, a expansão para universidades e o desenvolvimento do DNS marcaram o crescimento da rede. A popularização na década de 1990 foi impulsionada pela criação do World Wide Web, navegadores como o Netscape, e o surgimento de provedores comerciais. A partir dos anos 2000, a Internet se consolidou com a banda larga, redes móveis, e fenômenos como redes sociais, streaming e a Internet das Coisas.

No contexto da Internet, um patch é uma atualização de software criada para corrigir falhas, otimizar desempenho ou proteger contra vulnerabilidades. Ao longo dos anos, patches se tornaram essenciais para a segurança e funcionalidade da rede, respondendo rapidamente a ameaças como Meltdown e Spectre, Heartbleed, e o ataque WannaCry. Essas correções não apenas protegem sistemas operacionais e navegadores, mas também garantem a estabilidade e a evolução contínua dos serviços conectados globalmente. A implementação de patches é parte vital da manutenção da segurança e desempenho da Internet.

2 Faça um estudo sobre a evolução do protocolo TCP desde sua concepção até os dias de hoje. Para atender as demandas atuais, quais foram as últimas adaptações que o TCP sofreu?

O protocolo TCP foi concebido nos anos 1970 por Vint Cerf e Robert Kahn, como parte do projeto para criar uma rede de comunicação confiável e robusta. Inicialmente, sua função principal era garantir a entrega ordenada de pacotes de dados, mesmo em redes instáveis. Ao longo do tempo, o TCP tornou-se o protocolo padrão para a maior parte do tráfego da Internet, sendo projetado para controlar a fragmentação, o controle de fluxo, e a confiabilidade nas transmissões. Nos primeiros anos, o protocolo se estabeleceu com características fundamentais como o handshake de três vias, a janela deslizante para controle de fluxo, e a capacidade de detectar erros e reenviar pacotes perdidos.

Com o passar dos anos, o TCP passou por várias melhorias para atender às demandas crescentes da Internet, como a explosão de usuários e o aumento de tráfego de dados, especialmente devido a streaming e serviços em tempo real. Dentre as adaptações recentes, uma das mais significativas foi a introdução do TCP Fast Open, que reduz a latência no processo de estabelecimento de conexão, permitindo o envio de dados durante o handshake inicial. Outra importante modificação foi o desenvolvimento de algoritmos de controle de congestionamento mais eficientes, como o TCP BBR (Bottleneck Bandwidth and Round-trip propagation time), criado pelo Google. O BBR ajusta dinamicamente a taxa de envio de dados para maximizar o uso da largura de banda, ao invés de se basear apenas na detecção de perda de pacotes, proporcionando um desempenho mais eficiente em redes de alta velocidade e com alta latência.

3 Se o IP tem o campo QoS, por que o TCP também tem que implementar QoS? Como o TCP implementa QoS?

O IP tem um campo chamado QoS (Quality of Service), especificamente no cabeçalho IPv4 e no IPv6. Esse campo foi criado para permitir que os pacotes de dados fossem tratados de maneira diferenciada nas redes, garantindo diferentes níveis de prioridade, como menor atraso, maior confiabilidade ou maior taxa de transferência. No entanto, o TCP também precisa implementar suas próprias funcionalidades relacionadas à qualidade de serviço, porque o IP e o TCP operam em camadas diferentes e abordam diferentes aspectos do tráfego de rede.

Enquanto o campo QoS do IP se concentra no encaminhamento de pacotes ao longo da rede, o TCP opera na camada de transporte, lidando com a entrega confiável dos dados entre as duas pontas da comunicação. O TCP implementa QoS de maneira indireta por meio de seus mecanismos de controle de fluxo e

congestionamento, que ajustam o envio de pacotes de acordo com as condições da rede. Além disso, o TCP pode ser configurado para dar suporte a diferentes tipos de tráfego, como aplicações que exigem baixa latência como streaming e VoIP ou alta confiabilidade como transferências de arquivos. Alguns métodos que o TCP usa para lidar com QoS incluem:

- **Controle de Congestionamento**: O TCP ajusta a taxa de transmissão com base na condição da rede. Algoritmos modernos como TCP BBR buscam otimizar a largura de banda disponível sem causar congestionamento.
- **Controle de Fluxo**: Garante que o receptor não fique sobrecarregado, ajustando a taxa de envio dos pacotes com base na capacidade do receptor de processar os dados.
- **Priorização de Fluxos TCP**: Em alguns sistemas, é possível associar diferentes conexões TCP a diferentes níveis de prioridade, garantindo que fluxos críticos tenham maior prioridade em relação a fluxos menos importantes.

Assim, mesmo com o campo QoS no IP, o TCP precisa gerenciar seus próprios aspectos de performance e confiabilidade para garantir a entrega adequada dos dados, enquanto o IP lida com o roteamento e encaminhamento dos pacotes pela rede. Ambos trabalham em conjunto para oferecer um nível de serviço adequado para diferentes tipos de tráfego na Internet.

4 Faça um programa para gerar n bits de 16 bits e que faça o cálculo do checksum. Requisitos: A operação do checksum deve ser feita de forma interativa e bit a bit.

```
def calculate_checksum(bits):
    checksum = 0
    for bit in bits:
        checksum += bit
        if checksum > 0xFFFF:
            checksum = (checksum & 0xFFFF) + 1
    checksum = ~checksum & 0xFFFF
    return checksum
```

5 **Modularize a questão 4 e importe como uma biblioteca com a função checksum. Implemente dois processos, o sender e o receiver, para trocar n mensagens aleatórias entre si. O receiver deve receber a mensagem e calcular o checksum da mensagem recebida. Responda: Após as n-ésima sucessivas mensagens houve erro ou não em alguma mensagem enviada? Justifique o motivo da resposta obtida, ou seja, explique o porquê houve ou não houve erro. Os processos devem estar em container separados. Utilize o UDP para comunicação entre os processos.**

Figure 1: Saída da questão

```
sender    | Mensagem 1 enviada: 1110111000100000 com checksum 0xfff8
sender    | Mensagem 2 enviada: 0111101101110001 com checksum 0xfff5
receiver  | Mensagem recebida corretamente: 1110111000100000
sender    | Mensagem 3 enviada: 0110101010101101 com checksum 0xfff6
receiver  | Mensagem recebida corretamente: 0111101101110001
sender    | Mensagem 4 enviada: 0100111000110101 com checksum 0xfff7
receiver  | Mensagem recebida corretamente: 0110101010101101
sender    | Mensagem 5 enviada: 1101011010011111 com checksum 0xfff4
receiver  | Mensagem recebida corretamente: 0100111000110101
receiver  | Mensagem recebida corretamente: 1101011010011111
receiver  | Fim da transmissão
```

6 **Faça um programa para enviar datagramas entre dois processos usando o UDP. Estes processos (o sender e o receiver) devem estar em containers diferentes, interligados por uma topologia de barramento composto por 4 máquinas (ver figura). O transmissor deve gerar uma nova mensagem e enviar sequencialmente. Garanta que ele faça isso por 5 minutos. O Sender encerrará a transmissão com o envio da mensagem "The End".**

Figure 2: Saída da questão

```
receiver-1  | Taxa de recebimento (mensagens/segundo):
receiver-1  | Mínimo: 9.05, Máximo: 10.06, Média: 9.91, Desvio padrão: 0.02

sender-1    | Ack recebido para a mensagem 2956
sender-1    | Ack recebido para a mensagem 2957
sender-1    | Ack recebido para a mensagem 2958
sender-1    | Ack recebido para a mensagem 2959
sender-1    | Ack recebido para a mensagem 2960
sender-1    | Total de mensagens enviadas: 2962
sender-1    | Mensagens recebidas pelo receiver: 2961
sender-1    | Taxa de perda de mensagens: 0.03%
```