UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE ENGENHARIA ELETRÔNICA

UFMG

UNIVERSIDADE FEDERAL
DE MINAS GERAIS

ELT091– TURMA TEE

ESTUDOS DIRIGIDOS REDES TCP/IP

ESTUDO DIRIGIDO EM GRUPO

Guilherme Astolfo Rigacci
Augusto Ribeiro
Matheus Miranda

14 de junho de 2025

Estudos dirigidos redes TCP/IP

Estudo dirigido em grupo

Estudo dirigido 5

Autores:

Guilherme Astolfo Rigacci

Augusto Ribeiro

Matheus Miranda

Prof. Luciano de Errico

14 de junho de 2025

Sumário

1 Questões **1**

1.1 Questão 1 1
1.2 Questão 2 6
1.3 Questão 3 7
1.4 Questão 4 7
1.5 Questão 5 8
1.6 Questão 6 8

1 Questões

1.1 Questão 1

Leiam as Seções 5.2.8 e 5.2.10 do livro-texto e respondam:

a. O TCP é um protocolo que vem evoluindo com o tempo. Um exemplo disso são as extensões do TCP. Listem cada uma das extensões descritas no texto do livro, detalhando que problema ela soluciona e como ela opera.

b. O livro texto discute as decisões de projeto que definiram as características do TCP e cita um outro protocolo de transporte padronizado pelo IETF, o SCTP, que coexiste com o TCP e o UDP. Pesquisem e expliquem as diferenças do SCTP para o TCP e por que ele não é largamente utilizado.

Resposta:

a. i. Janela de Escala (Window Scale)

Problema Solucionado: O cabeçalho TCP original possui um campo de 16 bits para o tamanho da janela de recepção, o que limita o tamanho máximo da janela a 65.535 bytes (2^{16}). Em redes com alto produto de largura de banda e atraso (conhecidas como *long fat networks* ou LFNs), essa limitação impede o aproveitamento de toda a capacidade da rede, pois o transmissor precisa parar de enviar dados e aguardar a confirmação (ACK) com muita frequência.

Como Opera: A extensão de Escala de Janela, definida na RFC 7323, introduz um fator de escala que é negociado durante o *three-way handshake* da conexão TCP. Esse fator, que é um valor de deslocamento de bits, é usado para multiplicar o valor do campo de tamanho da janela. Por exemplo, um fator de escala de 7 significa que o tamanho da janela enviado no cabeçalho TCP deve ser multiplicado por 2^7 (128). Isso permite que a janela de recepção efetiva seja muito maior que 65.535 bytes, otimizando a transmissão de dados em redes de alta performance.

ii. Marcas de Tempo (Timestamps)

Problema Solucionado: Essa extensão aborda dois problemas principais: a medição imprecisa do Tempo de Ida e Volta (Round-Trip Time - RTT) e a ”Numeração de Sequência Envolvida”(Wrapped Around Sequence - PAWS). O RTT é crucial para o cálculo do tempo de retransmissão, e medições imprecisas podem levar a retransmissões desnecessárias ou demoradas. Já o PAWS ocorre em redes de altíssima velocidade, onde os números de sequência de 32 bits podem se repetir rapidamente, causando confusão entre pacotes antigos e novos.

Como Opera: A extensão de Marcas de Tempo adiciona um campo de 32 bits ao cabeçalho TCP. O remetente insere o seu tempo atual nesse campo. O receptor, por sua vez, ecoa esse valor de volta no campo de confirmação. Isso permite ao remetente calcular o RTT com muito mais precisão para cada pacote confirmado. Além disso, as marcas de tempo são monotonicamente crescentes, o que ajuda a distinguir pacotes com o mesmo número de sequência, mas de diferentes ”eras”da conexão, resolvendo o problema do PAWS.

iii. Confirmações Seletivas (Selective Acknowledgments - SACK)

Problema Solucionado: O mecanismo de confirmação original do TCP é cumulativo. Isso significa que, se um pacote for perdido, o receptor só pode confirmar a recepção dos pacotes até o ponto da perda. Consequentemente, o transmissor é forçado a retransmitir todos os pacotes a partir do segmento perdido, mesmo que alguns pacotes subsequentes tenham sido recebidos com sucesso. Isso gera retransmissões desnecessárias e ineficientes.

Como Opera: A extensão SACK permite que o receptor informe ao transmissor quais blocos de dados não contíguos foram recebidos com sucesso. Isso é feito incluindo no cabeçalho TCP a informação dos números de sequência dos blocos recebidos. Com essa informação, o transmissor sabe exatamente quais segmentos de dados se perderam e pode retransmitir apenas os pacotes faltantes, tornando a recuperação de perdas muito mais eficiente. A negociação para o uso do SACK também ocorre durante o handshake inicial da conexão.

iv. Retransmissão Rápida e Recuperação Rápida (Fast Retransmit and Fast Recovery)

Problema Solucionado: Esperar por um timeout (tempo de espera)

para retransmitir um pacote perdido pode introduzir um atraso significativo na comunicação. A Retransmissão Rápida e a Recuperação Rápida são mecanismos que aceleram a detecção e a recuperação de perdas de pacotes.

Como Opera:

- **Retransmissão Rápida:** Em vez de aguardar o esgotamento de um temporizador, a Retransmissão Rápida é acionada quando o transmissor recebe um número específico de ACKs duplicados (geralmente três). Um ACK duplicado é enviado pelo receptor toda vez que ele recebe um pacote fora de ordem. A recepção de vários ACKs duplicados para o mesmo número de sequência é um forte indicativo de que o pacote seguinte foi perdido. Ao detectar isso, o transmissor retransmite imediatamente o pacote presumidamente perdido.
- **Recuperação Rápida:** Após uma Retransmissão Rápida, a Recuperação Rápida entra em ação. Em vez de reduzir drasticamente a taxa de transmissão (como no mecanismo de *slow start*), este algoritmo permite que o transmissor continue a enviar novos pacotes de dados enquanto aguarda a confirmação do pacote retransmitido. Isso ajuda a manter um fluxo de dados mais constante e evita que a conexão fique ociosa, otimizando a utilização da largura de banda disponível.

b. O **SCTP (Stream Control Transmission Protocol)** foi projetado para combinar as melhores características do TCP (confiabilidade, controle de congestionamento) e do UDP (preservação de fronteiras de mensagem), adicionando novos recursos poderosos. No entanto, apesar de suas vantagens técnicas, ele não alcançou a mesma popularidade do TCP.

A seguir, apresentamos as principais diferenças entre os protocolos e os motivos para a baixa adoção do SCTP.

Diferenças: SCTP vs. TCP

Característica	SCTP (Stream Control Transmission Protocol)	TCP (Transmission Control Protocol)
Orientação dos Dados	Orientado a **mensagens**. Preserva as fronteiras das mensagens, entregando-as como unidades completas.	Orientado a **bytes**. Trata os dados como um fluxo contínuo de bytes, sem noção de mensagens individuais.
Múltiplos Fluxos (Multistreaming)	**Suportado**. Permite múltiplos fluxos de dados independentes dentro de uma única conexão (associação).	**Não suportado**. Uma única conexão oferece apenas um fluxo de dados.
Bloqueio ”Head-of-Line”	**Minimizado**. A perda de um pacote em um fluxo não bloqueia a entrega de pacotes nos outros fluxos.	**Problema comum**. A perda de um pacote impede a entrega de todos os pacotes subsequentes até que o perdido seja retransmitido.
Múltiplas Interfaces (Multi-homing)	**Nativo**. Permite que uma única conexão utilize múltiplos endereços IP em cada ponta, oferecendo redundância e tolerância a falhas de rede.	**Não suportado**. Uma conexão é rigidamente definida por um par de endereços IP e portas.
Estabelecimento da Conexão	**Four-Way Handshake**. Utiliza um mecanismo de ”cookie”para validar o cliente antes de alocar recursos, oferecendo proteção contra ataques de inundação SYN.	**Three-Way Handshake**. Aloca recursos após o primeiro passo (pacote SYN), o que o torna vulnerável a ataques de inundação SYN.
Encerramento da Conexão	**Three-Way Handshake**. Garante que ambos os lados fechem a conexão de forma limpa, sem a possibilidade de um estado ”meio-aberto”.	**Four-Way Handshake**. Pode levar a um estado ”meio-aberto”, onde um lado fecha a conexão, mas o outro continua enviando dados.

Por que o SCTP não é Largamente Utilizado?

Apesar de suas vantagens, o SCTP enfrenta barreiras significativas que limitaram sua adoção em larga escala na internet pública.

(a) **Falta de Suporte Nativo nos Sistemas Operacionais**
A principal barreira é a falta de suporte ”out-of-the-box”nos sistemas

operacionais mais populares para desktops, como **Windows e macOS**. Embora esteja presente em sistemas como Linux e FreeBSD, a ausência em plataformas de consumo massivo torna difícil para os desenvolvedores de aplicações adotá-lo, pois isso exigiria a instalação de drivers ou bibliotecas de terceiros pelos usuários.

(b) **Problemas com a Infraestrutura de Rede Existente (Ossificação)**
A internet está repleta de "middleboxes", como firewalls e dispositivos de **NAT (Network Address Translation)**, que são otimizados para TCP e UDP. Muitos desses dispositivos não reconhecem o SCTP e simplesmente bloqueiam seu tráfego. Além disso, o cálculo de checksum do SCTP (usando CRC32) é mais complexo de ser recalculado por um dispositivo NAT do que o do TCP, o que desencorajou a implementação em muitos roteadores domésticos e corporativos.

(c) **Inércia e o Ecossistema do TCP**
O TCP é a base da internet há décadas. Toda a infraestrutura, ferramentas de desenvolvimento, bibliotecas de software e o conhecimento dos engenheiros de rede e desenvolvedores são construídos em torno do TCP. Migrar para um novo protocolo exigiria um esforço monumental e custos significativos, e para a maioria das aplicações (como navegação web e transferência de arquivos), o TCP é considerado "bom o suficiente".

(d) **Soluções Alternativas na Camada de Aplicação**
Muitos dos problemas que o SCTP visa resolver no nível de transporte foram abordados em outras camadas. Por exemplo:

- **Multistreaming:** O **HTTP/2 e o HTTP/3 (QUIC)** implementam o conceito de múltiplos fluxos na camada de aplicação, eliminando o bloqueio "head-of-line"para aplicações web sem a necessidade de substituir o TCP/UDP.
- **Multi-homing:** O **Multipath TCP (MPTCP)** surgiu como uma extensão do próprio TCP para fornecer funcionalidades de multi-homing.

(e) **Aplicações de Nicho**
O SCTP encontrou seu principal caso de uso em nichos específicos, principalmente nas **redes de telecomunicações**. Ele é amplamente utilizado para transportar sinalização em redes 4G e 5G (nos protocolos Diameter e outros). Nesses ambientes controlados, as barreiras de NAT

e a falta de suporte de SO não são um problema. No entanto, para a internet aberta e de uso geral, ele permanece uma solução especializada e pouco difundida.

1.2 Questão 2

Ao fechar uma conexão TCP, por que a expiração do tempo limite de dois tempos de vida do segmento não é necessária na transição de `LAST_ACK` para `CLOSED`?

Resposta:

O Papel do Estado `TIME_WAIT`

Para entender por que a transição de `LAST_ACK` para `CLOSED` é direta, é crucial primeiro entender a função do estado `TIME_WAIT`, que ocorre no lado que inicia o encerramento da conexão (o fechamento ativo). O temporizador de 2 MSL (aproximadamente de 1 a 4 minutos) no estado `TIME_WAIT` tem duas finalidades críticas:

1. **Garantir a Entrega do `ACK` Final:** O estado `TIME_WAIT` garante que o último pacote `ACK` enviado pelo cliente chegue com sucesso ao servidor. Se este `ACK` for perdido, o servidor (que está no estado `LAST_ACK`) não receberá a confirmação, seu temporizador expirará e ele reenviará seu pacote `FIN`. O cliente, ainda no estado `TIME_WAIT`, pode então receber este `FIN` duplicado e reenviar o `ACK` final, permitindo que o servidor feche a conexão graciosamente.

2. **Prevenir Pacotes Duplicados Atrasados:** Uma conexão TCP é definida pela tupla de 4 elementos (IP de origem, porta de origem, IP de destino, porta de destino). Após o fechamento, uma nova conexão com a mesma tupla pode ser criada. O tempo de espera de 2 MSL garante que quaisquer pacotes atrasados (”duplicatas errantes”) da conexão antiga tenham tempo suficiente para expirar e serem descartados da rede, evitando que sejam erroneamente aceitos pela nova conexão.

A Transição de `LAST_ACK` para `CLOSED`

O lado da conexão que entra no estado `LAST_ACK` é o que realiza o fechamento passivo. Ele já recebeu um `FIN` do outro lado, respondeu com um `ACK`, e então

enviou seu próprio FIN. Neste ponto, ele está apenas aguardando um único evento: o ACK final para o seu FIN.

A expiração do tempo limite de 2 MSL não é necessária na transição de LAST_ACK para CLOSED pela seguinte razão:

A responsabilidade pela robustez do encerramento é delegada ao lado em TIME_WAIT.

Uma vez que o lado em LAST_ACK recebe o ACK final, ele tem a confirmação definitiva de que o outro lado recebeu todos os dados e o seu pedido de encerramento (FIN). Neste momento, ele sabe que o outro lado da conexão já fez a transição para o estado TIME_WAIT.

Portanto, é o outro lado que agora assume a responsabilidade de aguardar por 2 MSL para lidar com pacotes perdidos ou duplicados. O lado em LAST_ACK completou sua parte na sequência de encerramento e pode, com segurança, liberar seus recursos e transitar diretamente para o estado CLOSED, pois qualquer problema remanescente na rede será gerenciado pelo temporizador do estado TIME_WAIT no peer. Manter um temporizador semelhante em LAST_ACK seria redundante e um desperdício de recursos.

1.3 Questão 3

Um emissor em uma conexão TCP que recebe uma janela anunciada 0 sonda o receptor periodicamente para descobrir quando a janela se torna diferente de zero. Por que o receptor precisaria de um temporizador extra se ele fosse responsável por informar que sua janela anunciada se tornou diferente de 0 (ou seja, se o transmissor não fizesse a sondagem)?

Resposta:

1.4 Questão 4

O campo de número de sequência no cabeçalho TCP tem 32 bits de extensão, que é grande o suficiente para cobrir mais de 4 bilhões de dados. Mesmo que todos esses bytes nunca sejam transferidos em uma única conexão, por que o número de sequência ainda pode se reiniciar ciclicamente de $2^{32} - 1$ para 0?

Resposta:

1.5 Questão 5

Você foi encarregado de projetar um protocolo de fluxo de bytes confiável que use janela deslizante (como o TCP). Esse protocolo será executado em uma rede de 100 Mbps. O RTT da rede é de 100 ms, e o tempo de vida máximo dos segmentos é de 30 segundos. Quantos bits você incluiria nos campos JanelaAnunciada e NúmeroSeq do cabeçalho do seu protocolo?

Resposta:

1.6 Questão 6

Quando o TCP envia um `<SYN, NúmeroSeq=x>` ou `<FIN, NúmeroSeq=x>`, o ACK correspondente possui Confirmação=x+1; ou seja, SYNs e FINs ocupam uma unidade no espaço do número de sequência. Isso é necessário? Se for, dê um exemplo de uma ambiguidade que surgiria se a Confirmação correspondente fosse x em vez de x+1; se não, explique por quê.

Resposta: