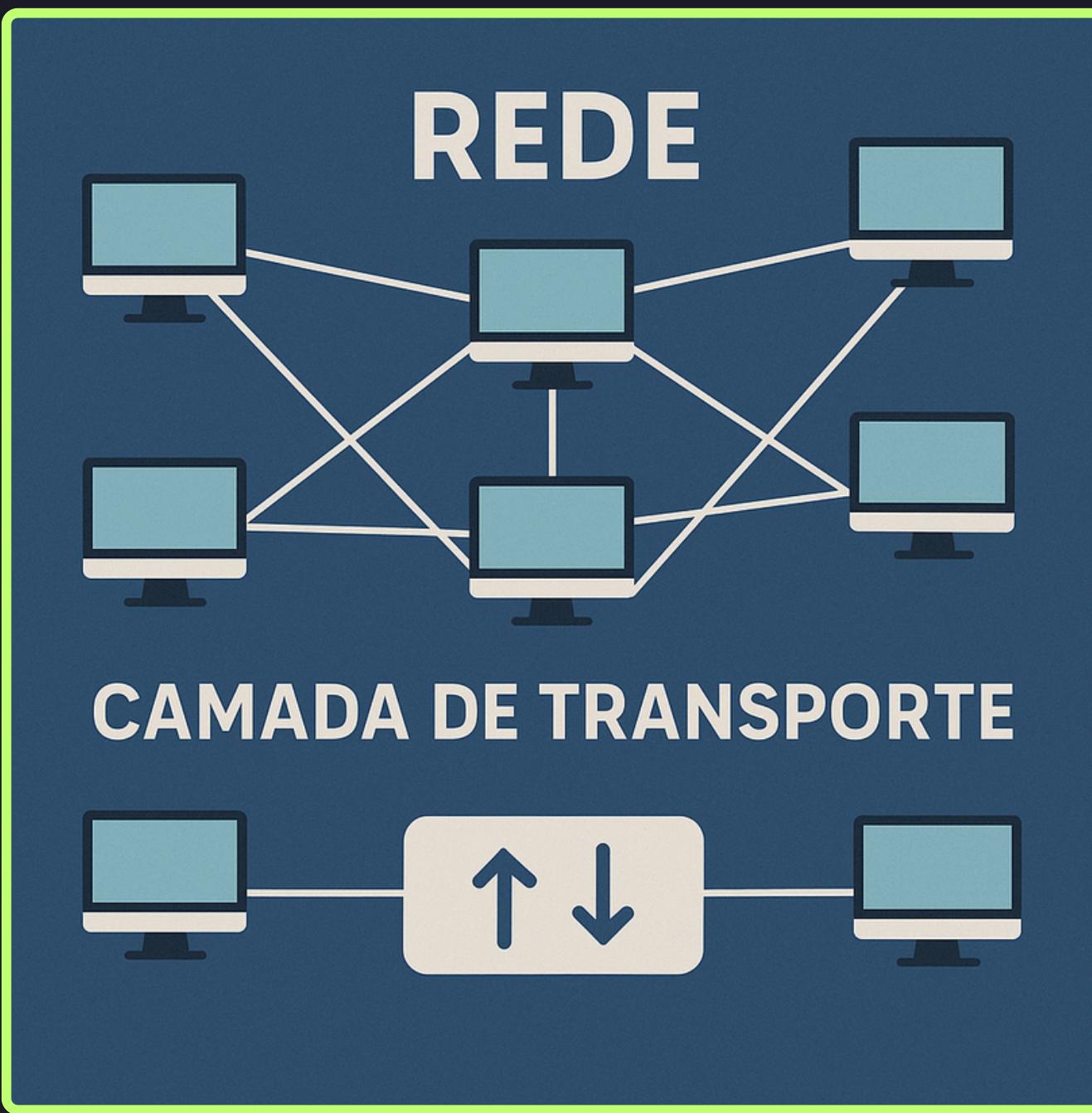


CAMADA DE TRANSPORTE



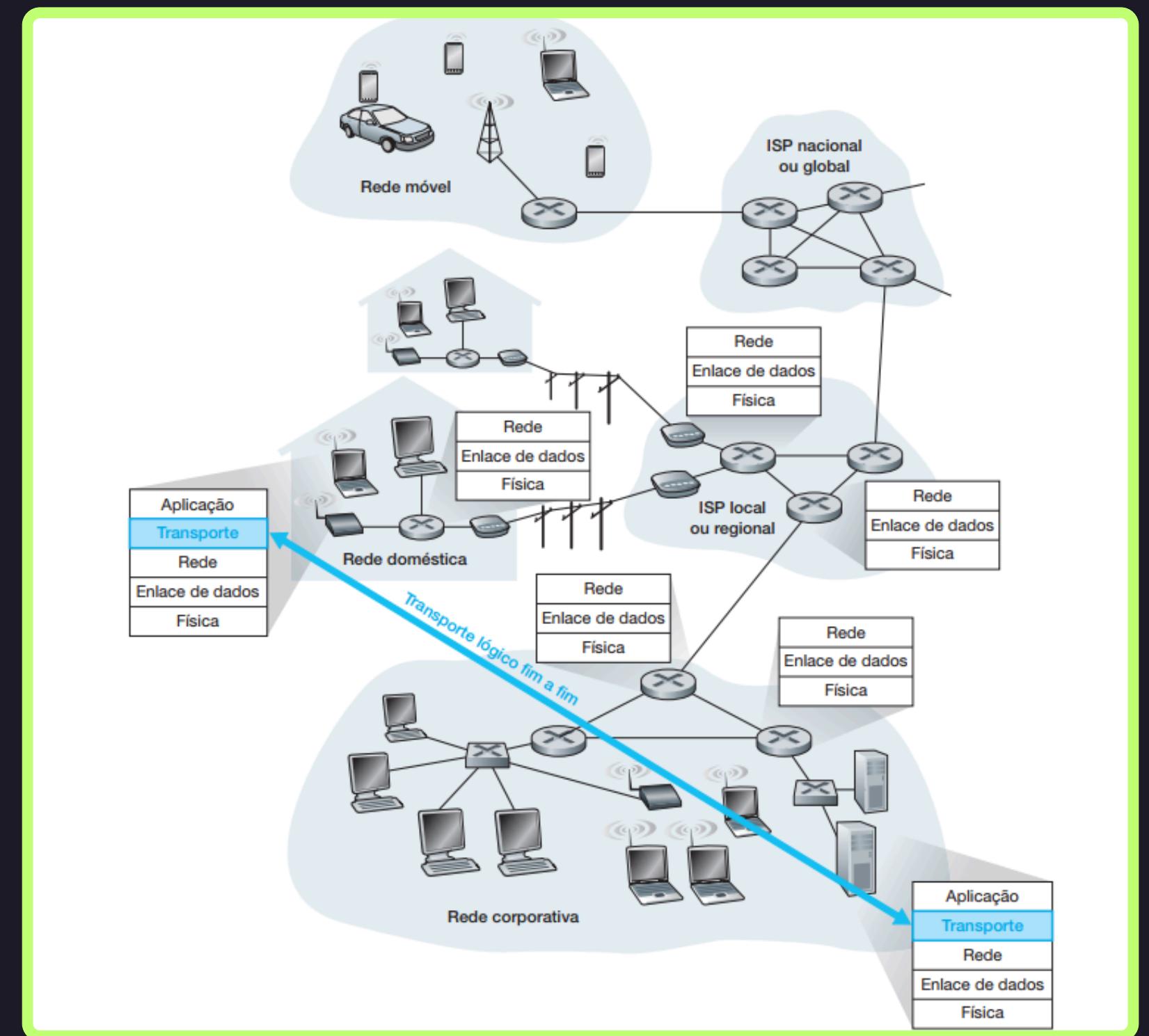
CAMADA DE TRANSPORTE

A camada de **transporte** é uma das camadas fundamentais do modelo OSI (Open Systems Interconnection) e do modelo **TCP/IP**, desempenhando um papel crucial na comunicação entre dispositivos em redes de computadores. Sua principal responsabilidade é garantir a entrega confiável, ordenada e livre de erros dos dados entre aplicações que estão se comunicando em diferentes dispositivos.

Enquanto as camadas inferiores (como a camada de rede e a de enlace de dados) cuidam da movimentação dos pacotes entre dispositivos na rede, a camada de transporte se concentra na comunicação entre **processos (aplicações)** e na gestão da sessão de comunicação, controlando como os dados são enviados e recebidos.

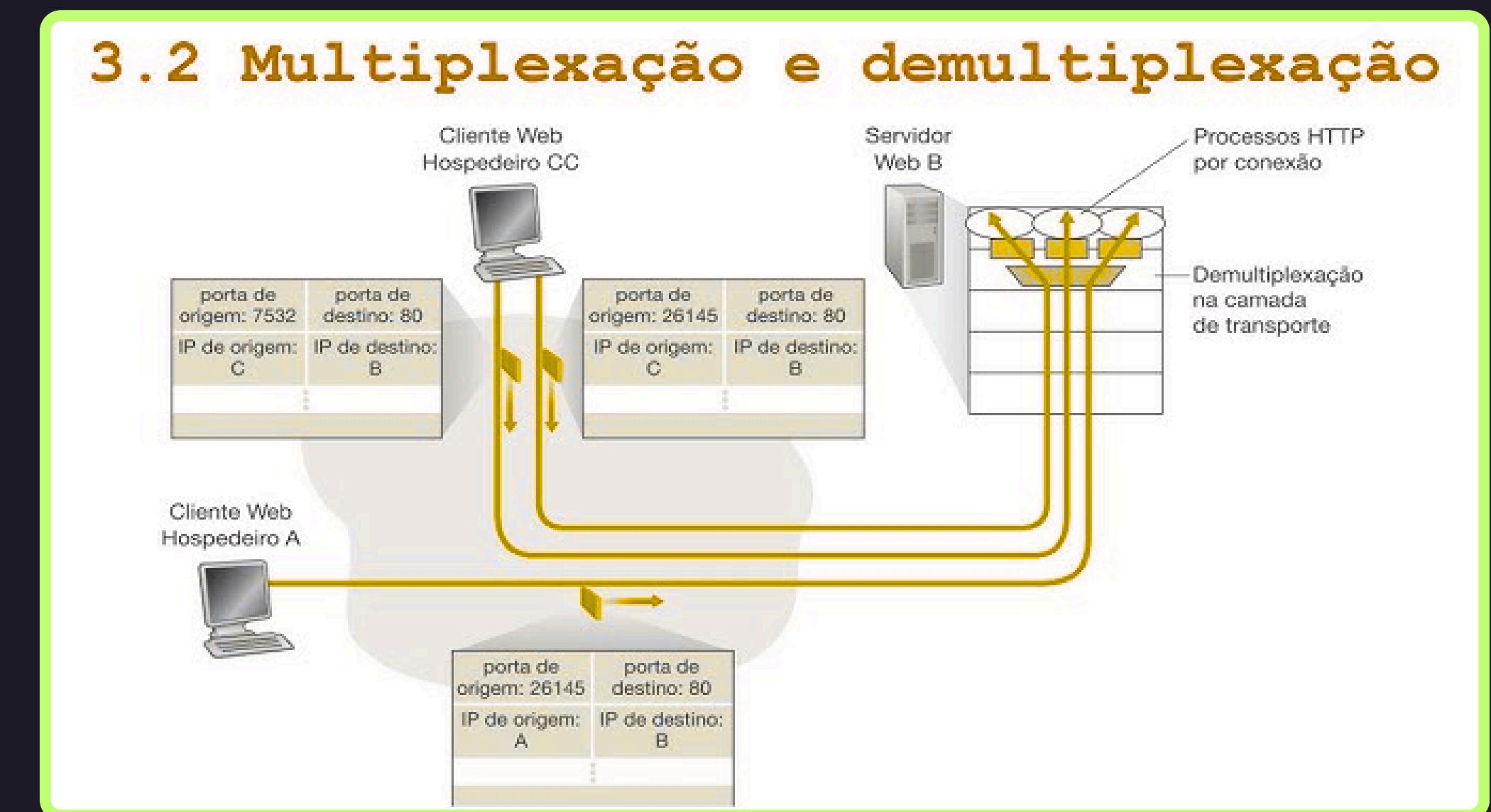
CAMADA DE TRANSPORTE

A camada de transporte fornece comunicação lógica, e não física, entre Processos de aplicações



CAMADA DE TRANSPORTE

A **multiplexação** e a **demultiplexação** são funções fundamentais da camada de transporte, responsáveis por permitir que múltiplas aplicações utilizem simultaneamente os serviços da rede.

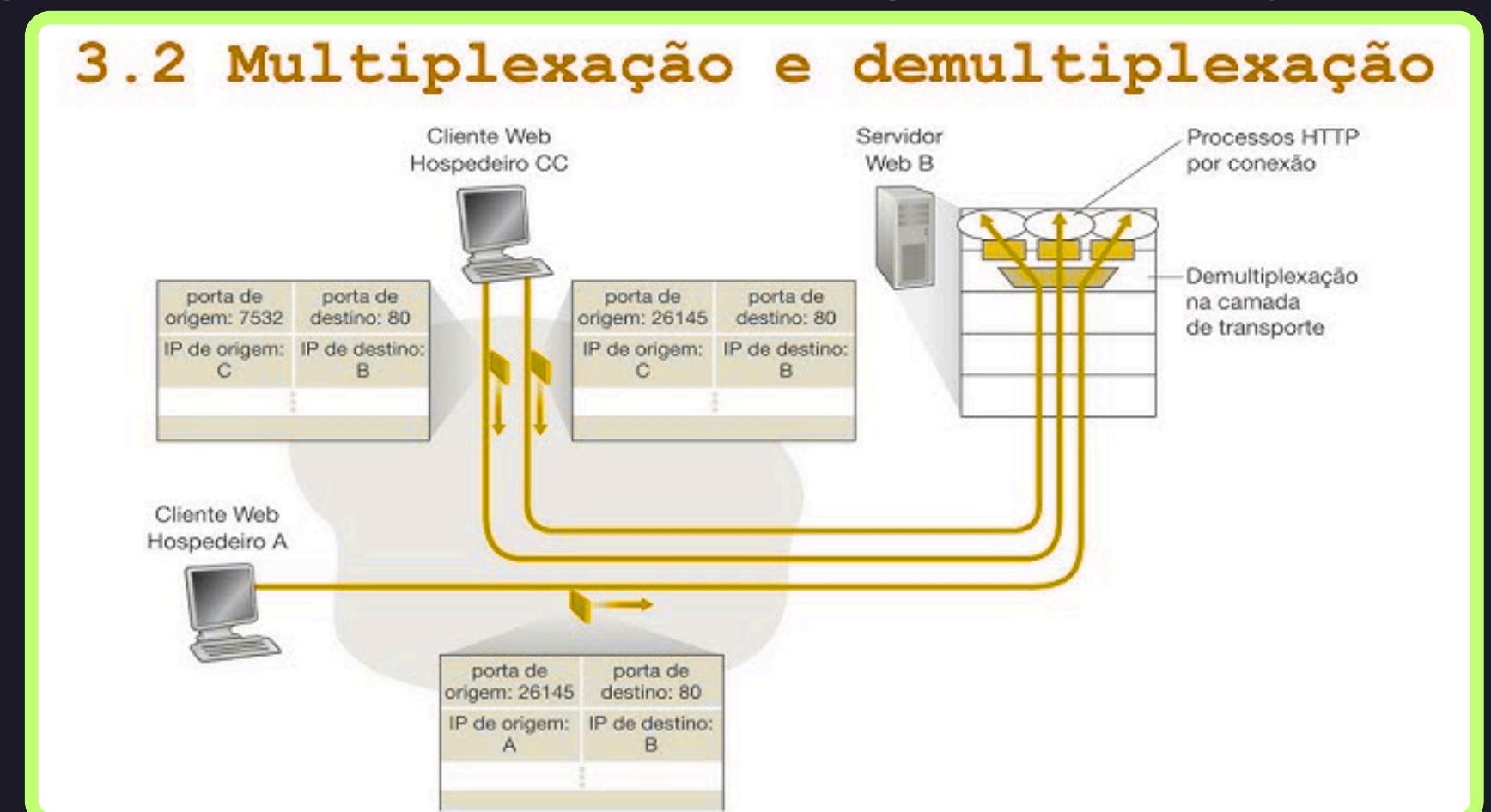


CAMADA DE TRANSPORTE

O que é Multiplexação?

A **multiplexação** ocorre no lado do remetente, onde a camada de transporte recebe dados de diferentes aplicações (como navegador, e-mail, mensageiro, etc.) e os prepara para envio. Cada conjunto de dados é associado a um número de **porta de origem**, permitindo que seja identificado mais tarde.

Assim, mesmo que várias aplicações estejam usando a rede ao mesmo tempo, a camada de transporte consegue organizar os dados de forma que todos sejam enviados corretamente.

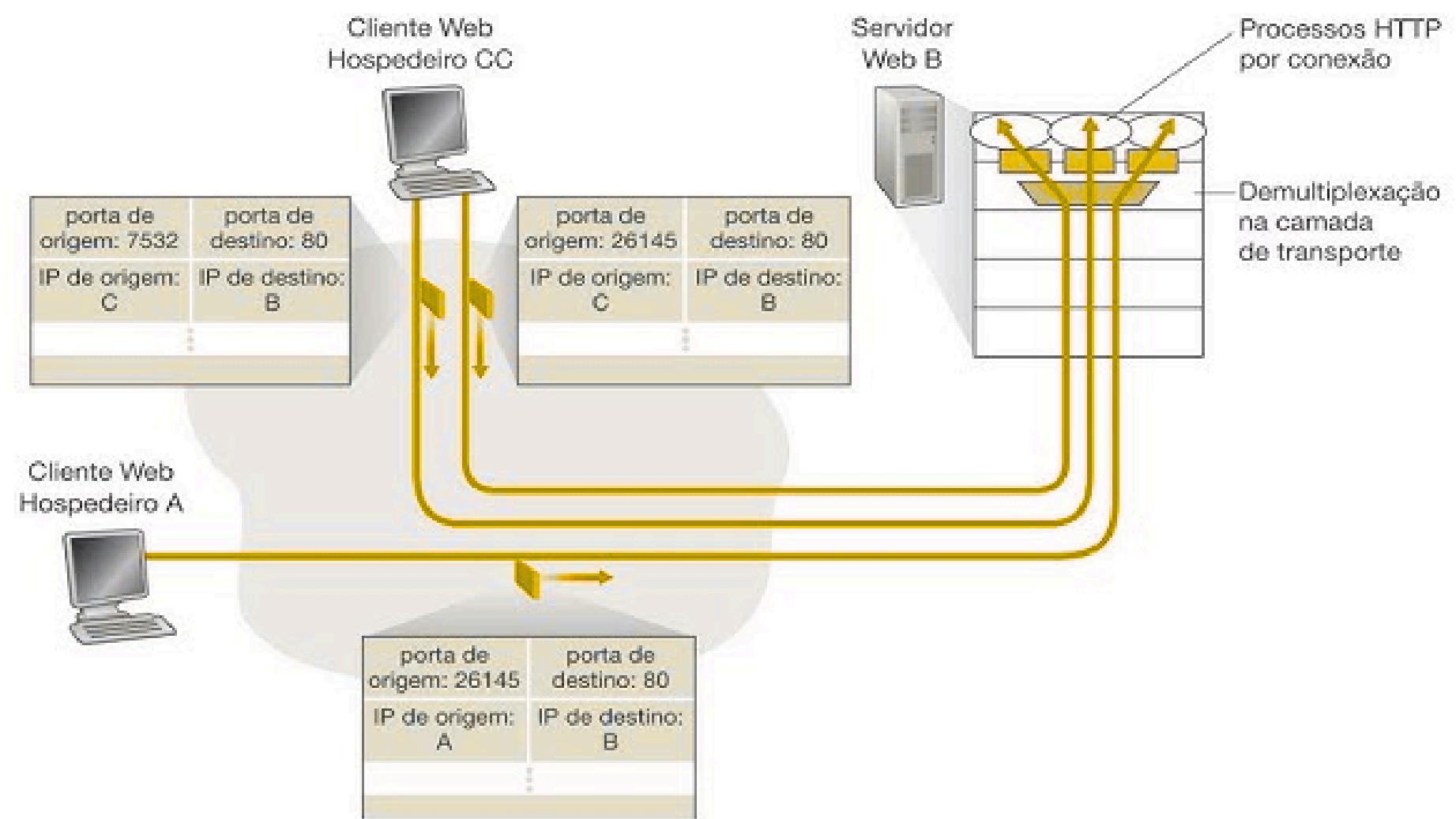


CAMADA DE TRANSPORTE

O que é Demultiplexação?

A **demultiplexação** acontece no lado do **receptor**, onde os dados recebidos da rede são distribuídos para a aplicação correta, com base no número de porta de destino. A camada de transporte examina esses números para identificar qual aplicação deve receber cada segmento de dados.

3.2 Multiplexação e demultiplexação



CAMADA DE TRANSPORTE

Como funciona na prática?

Imagine um computador rodando simultaneamente um navegador (porta **80**), um cliente de e-mail (porta **110**) e um aplicativo de chat (porta **5222**). Quando os dados chegam da rede, a camada de transporte usa os números de porta para demultiplexar os dados e entregá-los corretamente a cada aplicação.

Exemplo com TCP

Aplicativo	Porta de origem	Porta de destino	Protocolo
Navegador Web	49523	80 (HTTP)	TCP
Cliente de E-mail	49524	110 (POP3)	TCP
Aplicativo de Chat	49525	5222 (XMPP)	TCP



CAMADA DE TRANSPORTE

Multiplexação e Demultiplexação Não Orientadas para Conexão

Nos sistemas de comunicação não orientados para conexão, como o **UDP (User Datagram Protocol)**, as funções de multiplexação e demultiplexação ainda são fundamentais, mas ocorrem de maneira mais simples e direta do que nos protocolos orientados à conexão, como o **TCP**.

O que significa "**não orientado para conexão**" ?

Significa que não há uma conexão prévia estabelecida entre o emissor e o receptor antes da troca de dados. Cada pacote (**datagrama**) é enviado de forma independente, sem garantias de entrega, ordem ou integridade.

CAMADA DE TRANSPORTE

Multiplexação no UDP

A multiplexação no **UDP** acontece no lado do remetente. Quando diferentes aplicações desejam enviar dados pela rede, o protocolo UDP **encapsula** esses dados com um número de porta de origem e destino.

Cada aplicação é identificada por uma porta, e o UDP se encarrega de criar datagramas com essas informações, mesmo que o destino seja o mesmo servidor.

Exemplo:

- Um navegador pode enviar dados pela porta de origem **54001** para o servidor DNS na porta **53**.
- Um aplicativo de música pode usar a porta de origem **54002** para enviar dados de streaming para outro servidor.

CAMADA DE TRANSPORTE

Demultiplexação no UDP

No lado do receptor, o UDP executa a **demultiplexação**, verificando o número de porta de destino contido no cabeçalho do datagrama recebido. Com isso, ele entrega os dados diretamente à aplicação correspondente.

Ao contrário do **TCP**, que cria conexões exclusivas baseadas em **4 tuplas** (IP origem, porta origem, IP destino, porta destino), o UDP apenas utiliza a **porta de destino** para entregar os dados, o que torna o processo mais leve, mas menos seguro.

Característica	UDP
Estabelecimento de conexão prévia	✗ Não
Entrega confiável	✗ Não
Ordenação garantida	✗ Não
Identificação de aplicação por porta	✓ Sim
Baixo atraso (baixa latência)	✓ Sim

Aplicações comuns

- DNS (Domain Name System)
- Streaming de áudio/vídeo
- Jogos online
- Chamadas VoIP



CAMADA DE TRANSPORTE

Aplicações Populares da internet e seus Protocolos de transporte subjacentes

No lado do receptor, o UDP executa a **demultiplexação**, verificando o número de porta de destino contido no cabeçalho do datagrama recebido. Com isso, ele entrega os dados diretamente à aplicação correspondente.

Ao contrário do **TCP**, que cria conexões exclusivas baseadas em **4 tuplas** (IP origem, porta origem, IP destino, porta destino), o UDP apenas utiliza a **porta de destino** para entregar os dados, o que torna o processo mais leve, mas menos seguro.

Aplicação	Protocolo da camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP	TCP
Acesso a terminal remoto	Telnet	TCP
Web	HTTP	TCP
Transferência de arquivo	FTP	TCP
Servidor de arquivo remoto	NFS	Tipicamente UDP
Recepção de multimídia	Tipicamente proprietário	UDP ou TCP
Telefonia por Internet	Tipicamente proprietário	UDP ou TCP
Gerenciamento de rede	SNMP	Tipicamente UDP
Protocolo de roteamento	RIP	Tipicamente UDP
Tradução de nome	DNS	Tipicamente UDP



CAMADA DE TRANSPORTE

Protocolos de Transferência Confiável de Dados com Paralelismo

Em **redes de computadores**, a transferência confiável de dados é essencial para garantir que as mensagens enviadas cheguem ao destino completas, na ordem correta e sem erros. Para isso, os protocolos de transporte utilizam diversos mecanismos de controle, como retransmissões, confirmações (**ACKs**) e controle de fluxo.

Uma estratégia que melhora ainda mais o desempenho da comunicação confiável é o uso do paralelismo, por meio de técnicas como janelamento deslizante (sliding window), que permitem o envio de múltiplos segmentos de dados antes de receber confirmações.

Por que usar paralelismo?

Sem paralelismo, um protocolo confiável (como o protocolo **Stop-and-Wait**) envia um segmento e aguarda o **ACK** antes de enviar o próximo. Esse método é simples, mas muito ineficiente em redes de alta latência, pois desperdiça tempo de transmissão.

Com paralelismo, vários segmentos podem ser enviados antes de serem confirmados, aumentando significativamente a eficiência e o aproveitamento da banda.



CAMADA DE TRANSPORTE

Protocolos de Transferência Confiável de Dados com Paralelismo

Protocolos com suporte a paralelismo

- ◆ **Go-Back-N (GBN)**

- Usa uma janela deslizante que permite enviar vários pacotes sem esperar ACK imediato.
- Se um pacote é perdido, todos os subsequentes são retransmitidos.
- Simples, mas pode causar retrabalho.

- ◆ **Selective Repeat (SR)**

- Também usa janela deslizante, mas permite retransmitir apenas os pacotes perdidos ou corrompidos.
- Exige mais complexidade no controle de buffers e no recebimento fora de ordem.
- É mais eficiente que o Go-Back-N.

- ◆ **TCP (Transmission Control Protocol)**

- Protocolo real amplamente utilizado.
- Usa uma forma avançada de janela deslizante, com controle de fluxo (janela do receptor) e controle de congestionamento (janela do emissor).
- Realiza retransmissões seletivas usando o mecanismo SACK (Selective Acknowledgment), quando habilitado.
- Adapta dinamicamente a quantidade de dados enviados em paralelo, conforme as condições da rede.



CAMADA DE TRANSPORTE

Transferência Confiável de Dados em um Canal com Perda e Erros de Bits: Protocolo rdt3.0

Em **redes de computadores**, a transferência confiável de dados é um dos grandes desafios quando há possibilidade de erros de bits e perda de pacotes durante a comunicação. Para lidar com essas falhas, são utilizados protocolos confiáveis, como o **rdt3.0**, projetado especificamente para funcionar em canais não confiáveis.

Contexto do Problema

Canais de comunicação estão sujeitos a diversas falhas:

- Erros de **bits**, causados por interferência elétrica, **ruído** ou distorções.
- Perda de pacotes, que pode ocorrer por congestionamento, falhas nos enlaces ou buffers cheios nos roteadores.

Essas falhas tornam necessário o uso de mecanismos que detectem e corrijam problemas, garantindo que os dados cheguem ao destino de forma correta e completa.

CAMADA DE TRANSPORTE

Transferência Confiável de Dados em um Canal com Perda e Erros de Bits: Protocolo rdt3.0

Antes do **rdt3.0**, versões anteriores lidavam com canais mais simples:

- **rdt1.0**: Funcionava em canais perfeitamente confiáveis, sem erros.
- **rdt2.0 e rdt2.1**: Tratavam canais com erros de bits, utilizando confirmações (ACKs) e negações (NAKs), além de bits de sequência.
- **rdt2.2**: Melhorava a confiabilidade substituindo NAKs por ACKs duplicados.

Porém, essas versões ainda não tratavam perda de pacotes — e é aí que o **rdt3.0** se destaca.

O que é o rdt3.0?

O **rdt3.0 (Reliable Data Transfer 3.0)** é um protocolo de pare e espere (stop-and-wait), que oferece confiabilidade total mesmo em um canal sujeito a erros e perdas. Seu funcionamento baseia-se em:

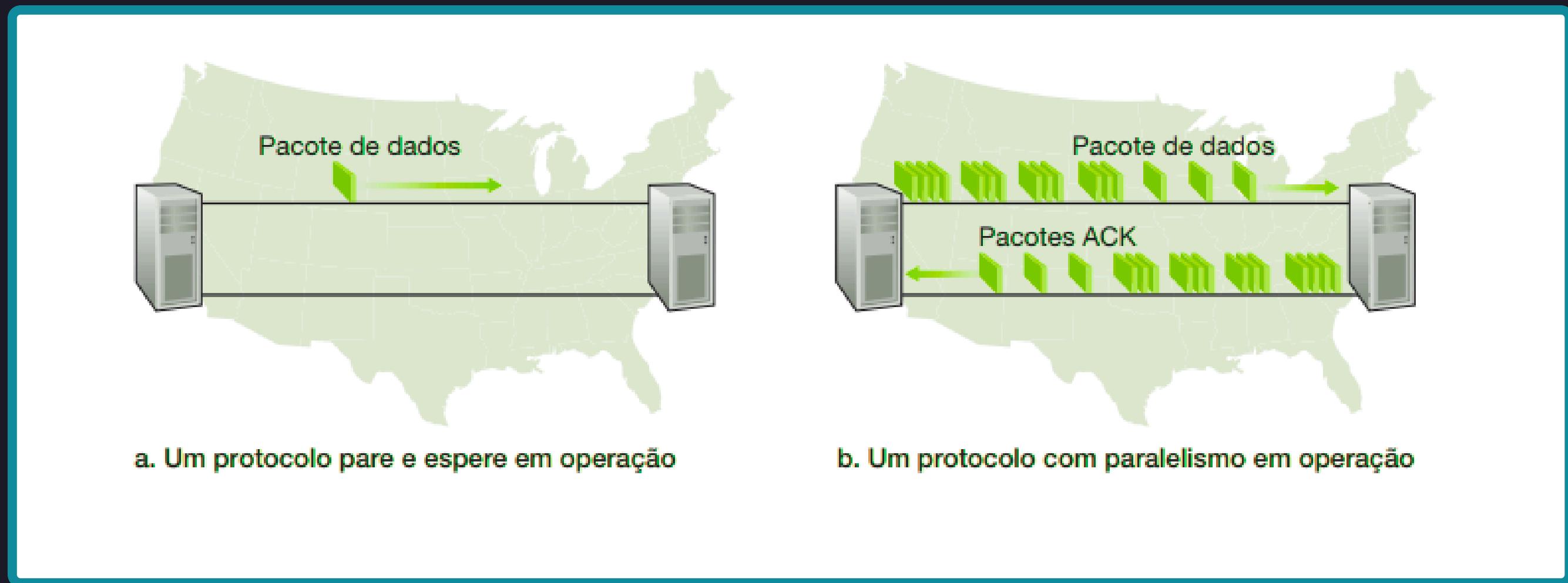
- Envio de um único pacote por vez.
- Esperar uma confirmação (ACK) antes de prosseguir.
- Uso de temporizador para detectar a perda de pacotes ou ACKs.

Se o ACK não chegar dentro de um tempo limite, o remetente retransmite o pacote, assumindo que ele ou sua confirmação foram perdidos.



CAMADA DE TRANSPORTE

Protocolo Pare e espere versus Protocolo com Paralelismo



A solução para esse problema de desempenho em especial é **simples**: em vez de operar em modo pare e espere, o remetente é autorizado a enviar vários pacotes sem esperar por reconhecimentos, se um remetente for autorizado a transmitir três pacotes antes de ter de esperar por reconhecimentos, sua utilização será triplicada.

CAMADA DE TRANSPORTE

Protocolo Go-Back-N (GBN)

Em redes de comunicação de dados, a garantia de entrega confiável de informações é fundamental. No entanto, os meios de transmissão são suscetíveis a ruídos e interferências, o que pode levar à perda ou corrupção de pacotes. Para lidar com esses problemas, diversos protocolos de controle de erros e fluxo foram desenvolvidos.

Um desses protocolos, fundamental para entender os mecanismos de comunicação confiável, é o **Go-Back-N (GBN)**.

O **Go-Back-N** é um protocolo da camada de transporte que utiliza o conceito de janela deslizante para gerenciar a transmissão de múltiplos pacotes antes de receber **acknowledgments (ACKs)**. Ele opera da seguinte maneira:

- **Numeração de Sequência:** Cada pacote transmitido é numerado sequencialmente. Essa numeração permite ao receptor identificar a ordem correta dos pacotes e detectar eventuais perdas ou duplicações.

CAMADA DE TRANSPORTE

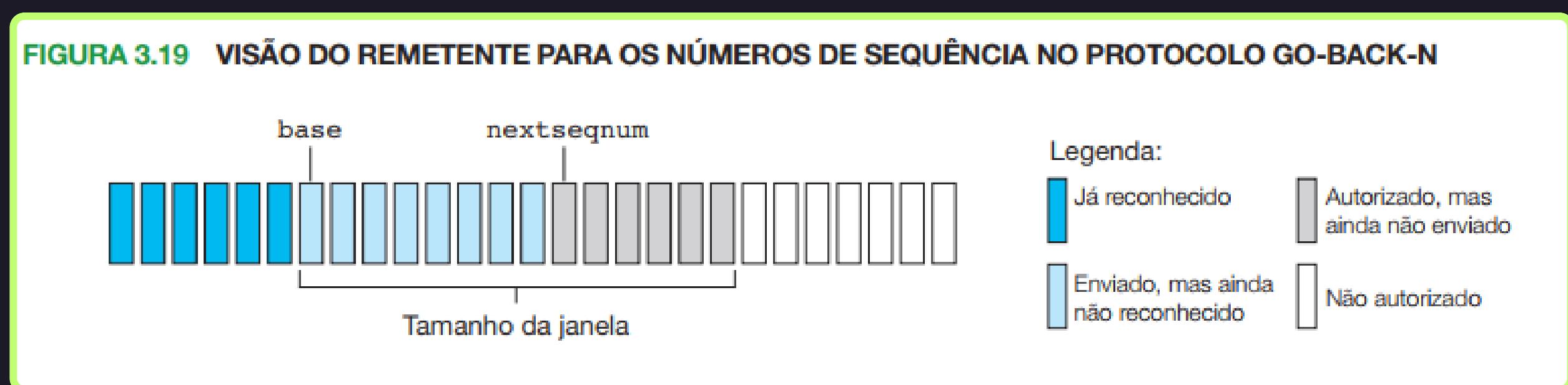
Protocolo Go-Back-N (GBN)

- **Janela de Transmissão:** O transmissor mantém uma janela de transmissão, que define o número máximo de pacotes que podem ser enviados sem receber um ACK correspondente. O tamanho dessa janela (N) é um parâmetro importante do protocolo.
- **Envio Contínuo:** O transmissor envia pacotes dentro de sua janela de transmissão de forma contínua, sem esperar por um ACK para cada pacote individualmente. Isso melhora a eficiência da transmissão, especialmente em redes com alto atraso.
- **Acknowledgments (ACKs):** O receptor envia ACKs para indicar que recebeu um pacote com sucesso. No protocolo GBN, o ACK para um pacote de número de sequência n implica que todos os pacotes até o número n (inclusive) foram recebidos corretamente. Esse tipo de ACK é conhecido como ACK cumulativo.
- **Timer de Timeout:** Para cada pacote enviado, o transmissor inicia um timer de timeout. Se um ACK para um determinado pacote não for recebido antes que o timer expire, o transmissor assume que o pacote (e todos os pacotes enviados posteriormente dentro da janela) foram perdidos.

CAMADA DE TRANSPORTE

Protocolo Go-Back-N (GBN)

- **Retransmissão:** Se ocorrer um timeout, o transmissor volta N pacotes (daí o nome "**Go-Back-N**") e retransmite todos os pacotes não reconhecidos, a partir do pacote que causou o timeout. O receptor, ao receber pacotes retransmitidos, simplesmente descarta aqueles que já foram recebidos (com base no número de sequência) e aceita os pacotes novos e na ordem correta.



CAMADA DE TRANSPORTE

Vantagens do Go-Back-N

- **Implementação relativamente simples:** Em comparação com outros protocolos de controle de erros mais sofisticados, como o Selective Repeat, o GBN possui uma lógica de implementação mais direta, tanto no transmissor quanto no receptor.
- **Bom desempenho em redes com baixa taxa de erros:** Quando a taxa de erros é baixa, o GBN permite uma transmissão eficiente devido ao envio contínuo de pacotes dentro da janela.
- **Garante a entrega ordenada:** O receptor sempre entrega os pacotes para a camada superior na ordem correta dos números de sequência.

CAMADA DE TRANSPORTE

Desvantagens do Go-Back-N

- **Ineficiência na recuperação de erros:** A principal desvantagem do GBN é a sua ineficiência na recuperação de erros. Quando um pacote é perdido, o transmissor precisa retransmitir não apenas o pacote perdido, mas também todos os pacotes que foram enviados posteriormente, mesmo que esses tenham sido recebidos corretamente pelo receptor. Isso pode levar a um desperdício significativo de largura de banda, especialmente em redes com alta taxa de erros ou com conexões de alta latência.
- **Descarte de pacotes recebidos fora de ordem:** O receptor descarta todos os pacotes que chegam fora de ordem (ou seja, pacotes com número de sequência maior que o esperado), mesmo que esses pacotes estejam corretos. Isso também contribui para a ineficiência do protocolo.

CAMADA DE TRANSPORTE

Cenários de Aplicação

O protocolo Go-Back-N pode ser adequado para cenários onde:

- A taxa de erros na rede é relativamente baixa.
- A simplicidade da implementação é uma prioridade.
- O desperdício de largura de banda causado por retransmissões desnecessárias é aceitável.
- A ordenação estrita dos pacotes na entrega é crucial.

CAMADA DE TRANSPORTE

Protocolo Repetição Seletiva (Selective Repeat - SR):

Dando continuidade ao nosso estudo sobre protocolos de controle de erros e fluxo, exploraremos agora o protocolo Repetição Seletiva (**Selective Repeat - SR**).

Como vimos no capítulo sobre o **Go-Back-N**, a retransmissão de múltiplos pacotes após a detecção de uma perda pode ser ineficiente, especialmente em redes com alta taxa de erros ou **alta latência**.

O **Repetição Seletiva** surge como uma alternativa mais sofisticada, buscando otimizar o processo de recuperação de erros ao retransmitir apenas os pacotes que foram efetivamente perdidos.

Neste capítulo, detalharemos os mecanismos de funcionamento do **protocolo SR**, suas vantagens sobre o **Go-Back-N**, suas desvantagens e os cenários onde sua utilização se mostra mais vantajosa.

Ao compreender o Repetição Seletiva, você terá uma visão mais completa das técnicas utilizadas para garantir a comunicação confiável em redes de dados.

CAMADA DE TRANSPORTE

Princípios Fundamentais do Repetição Seletiva

Assim como o **Go-Back-N**, o Repetição Seletiva também utiliza o conceito de janela deslizante e numeração de sequência para gerenciar a transmissão de pacotes. No entanto, a principal diferença reside na forma como o transmissor e o receptor lidam com a perda de pacotes e a retransmissão. Os princípios chave do SR são:

- **Numeração de Sequência:** Cada pacote transmitido é numerado sequencialmente, permitindo a identificação da ordem correta e a detecção de perdas ou duplicações.
- **Janela de Transmissão e Recepção:** Tanto o transmissor quanto o receptor mantêm uma janela. A janela de transmissão define o número máximo de pacotes que o transmissor pode enviar sem receber um ACK específico para cada um. A janela de recepção define o número máximo de pacotes que o receptor está preparado para aceitar fora de ordem. Idealmente, o tamanho da janela de transmissão (N) deve ser no máximo metade do espaço de numeração ($2m$, onde m é o número de bits usados para a numeração de sequência) para evitar ambiguidades em certas situações de perda de ACK.
- **Envio Contínuo:** O transmissor envia pacotes dentro de sua janela de transmissão sem esperar por ACKs individuais.

CAMADA DE TRANSPORTE

Princípios Fundamentais do Repetição Seletiva

- **Acknowledgments (ACKs) Individuais:** Ao contrário do GBN, o receptor envia um ACK individual para cada pacote recebido com sucesso. Esse ACK especifica o número de sequência do pacote recebido.
- **Negative Acknowledgments (NAKs):** Para informar o transmissor sobre um pacote perdido, o receptor pode enviar um Negative Acknowledgment (NAK) especificando o número de sequência do pacote que não foi recebido. O uso de NAKs é opcional, mas pode acelerar o processo de retransmissão.
- **Timers Individuais:** O transmissor mantém um timer individual para cada pacote enviado. Se o timer de um pacote específico expirar antes que um ACK para ele seja recebido, apenas esse pacote é considerado perdido e será retransmitido.
- **Buffer do Receptor:** O receptor mantém um buffer para armazenar os pacotes que chegam fora de ordem, desde que estejam dentro de sua janela de recepção. Quando o pacote faltante (com o número de sequência esperado) chega, o receptor pode então ordenar os pacotes armazenados no buffer e entregá-los à camada superior na sequência correta.

CAMADA DE TRANSPORTE

Princípios Fundamentais do Repetição Seletiva

- **Acknowledgments (ACKs) Individuais:** Ao contrário do GBN, o receptor envia um ACK individual para cada pacote recebido com sucesso. Esse ACK especifica o número de sequência do pacote recebido.
- **Negative Acknowledgments (NAKs):** Para informar o transmissor sobre um pacote perdido, o receptor pode enviar um Negative Acknowledgment (NAK) especificando o número de sequência do pacote que não foi recebido. O uso de NAKs é opcional, mas pode acelerar o processo de retransmissão.
- **Timers Individuais:** O transmissor mantém um timer individual para cada pacote enviado. Se o timer de um pacote específico expirar antes que um ACK para ele seja recebido, apenas esse pacote é considerado perdido e será retransmitido.
- **Buffer do Receptor:** O receptor mantém um buffer para armazenar os pacotes que chegam fora de ordem, desde que estejam dentro de sua janela de recepção. Quando o pacote faltante (com o número de sequência esperado) chega, o receptor pode então ordenar os pacotes armazenados no buffer e entregá-los à camada superior na sequência correta.