
DCC 602 – Sistemas Distribuídos

4 – Comunicação

Sumário

5 Comunicação

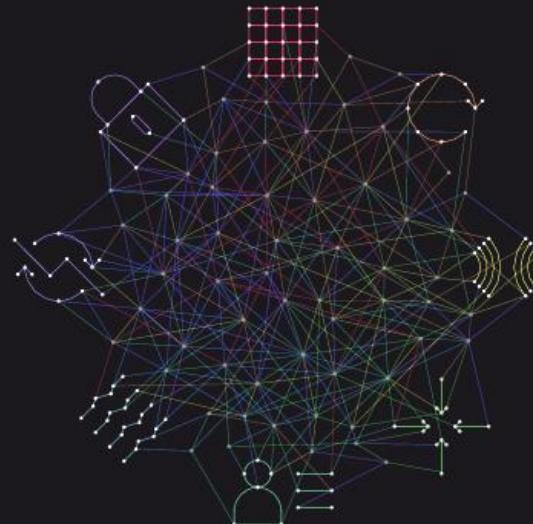
4.1 Fundamentos

4.2 Chamada de Procedimento Remoto

4.3 Comunicação Orientada a Mensagens

4.4 Comunicação Multicast

DISTRIBUTED SYSTEMS



MAARTEN VAN STEEN
ANDREW S. TANENBAUM

4TH EDITION

VERSION 00

4.1 Fundamentos

- Nesse capítulo, discutimos brevemente os **protocolos de comunicação de rede**, pois eles formam a **base de qualquer sistema distribuído**.
- A partir daí, adotamos uma abordagem diferente, **classificando os diferentes tipos de comunicação** que geralmente ocorrem em **sistemas distribuídos**.

4.1.1 Protocolos em Camadas

- Devido à **ausência de memória compartilhada**, toda a comunicação em sistemas distribuídos é baseada no envio e recebimento de mensagens (de baixo nível).
 - Quando o processo **P** deseja se comunicar com o processo **Q**, ele primeiro constrói uma mensagem em seu próprio espaço de endereço.
 - Em seguida, ele executa uma chamada de sistema que faz com que o sistema operacional envie a mensagem **pela rede para Q**.
 - Embora essa ideia básica pareça bastante simples, para evitar o caos, **P e Q precisam concordar com o significado dos bits enviados**

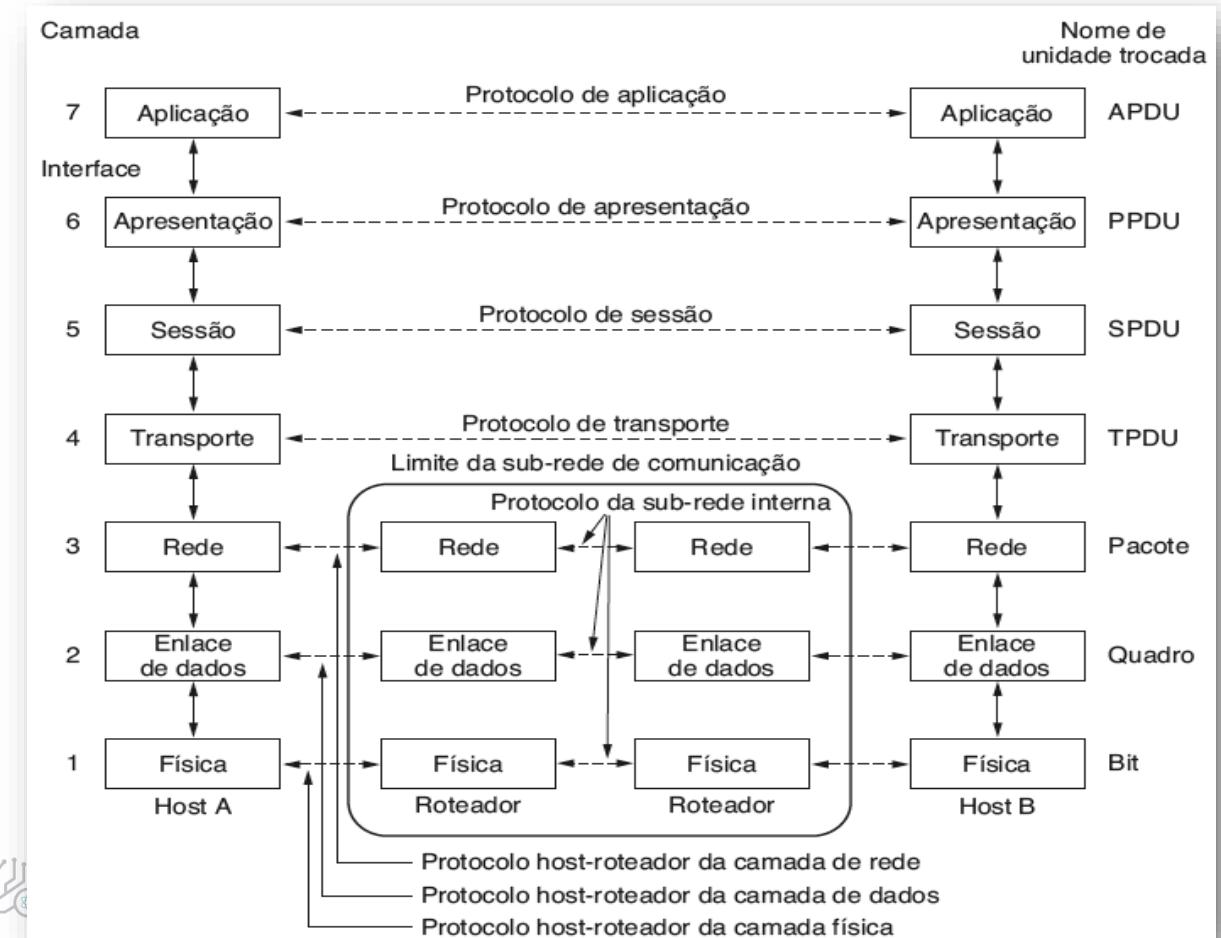
4.1.1 Protocolos em Camadas

- O **Modelo de Referência OSI** (Open Systems Interconnection) foi incluído no padrão **ISO 7489** e lançado em 1984.
- ISO significa Organização Internacional de Normalização



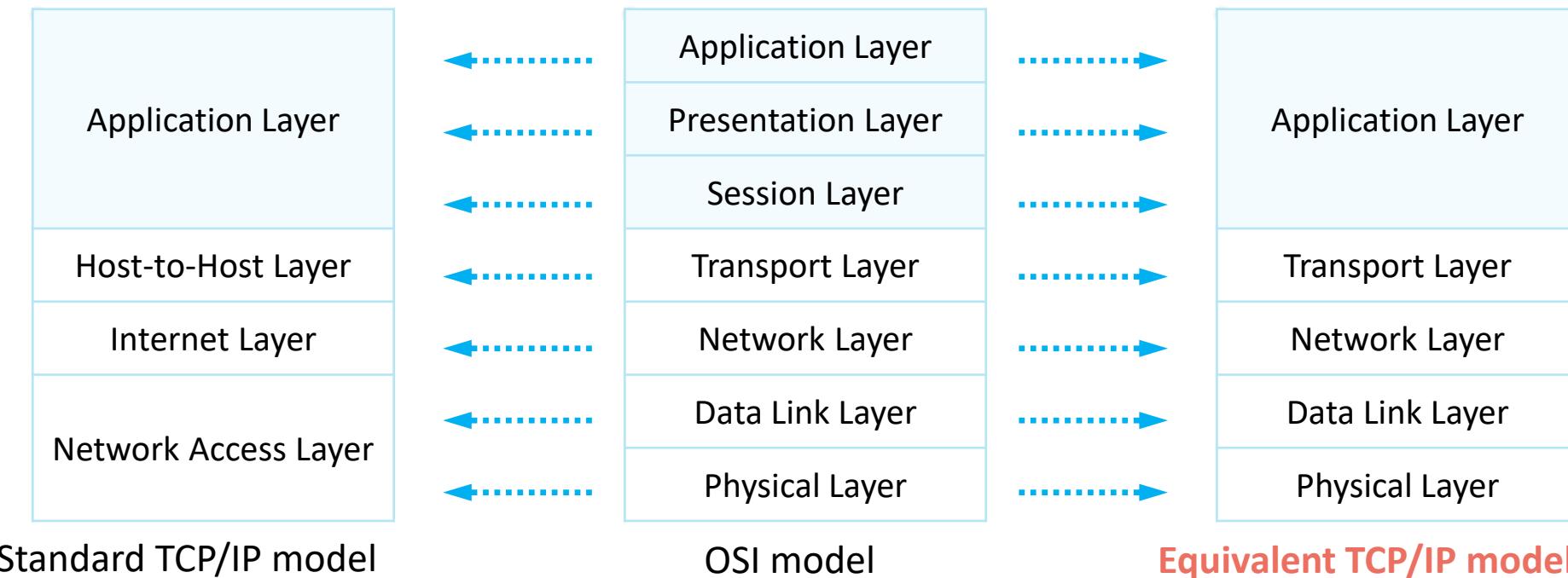
4.1.1 Protocolos em Camadas

- O modelo de referência OSI também é chamado de **modelo de sete camadas**. As sete camadas, de baixo para cima, são as seguintes:



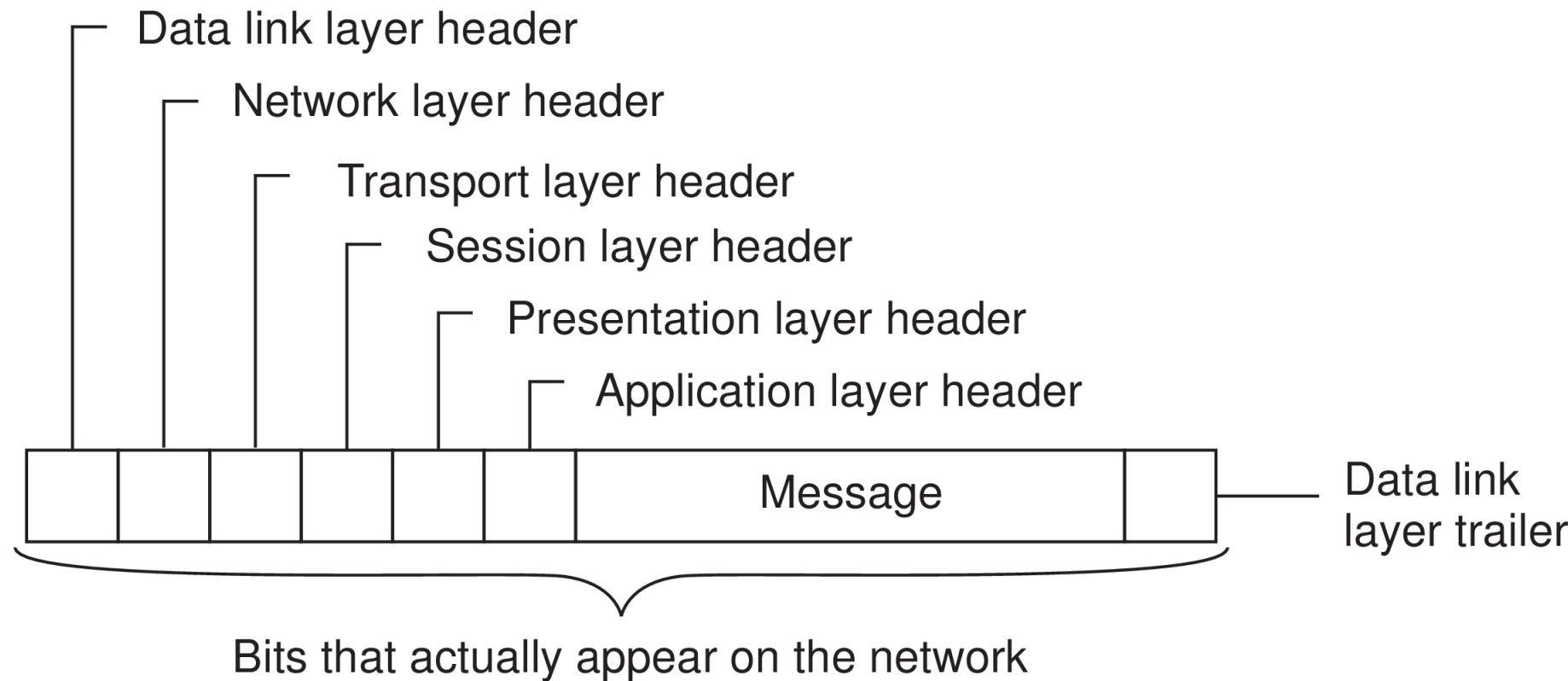
4.1.1 Protocolos em Camadas

- **Modelo TCP/IP.**
- A pilha de protocolos OSI é complexa e os protocolos TCP e IP são amplamente usados na indústria.
 - Portanto, o modelo de referência TCP/IP se tornou o modelo de referência predominante da Internet.

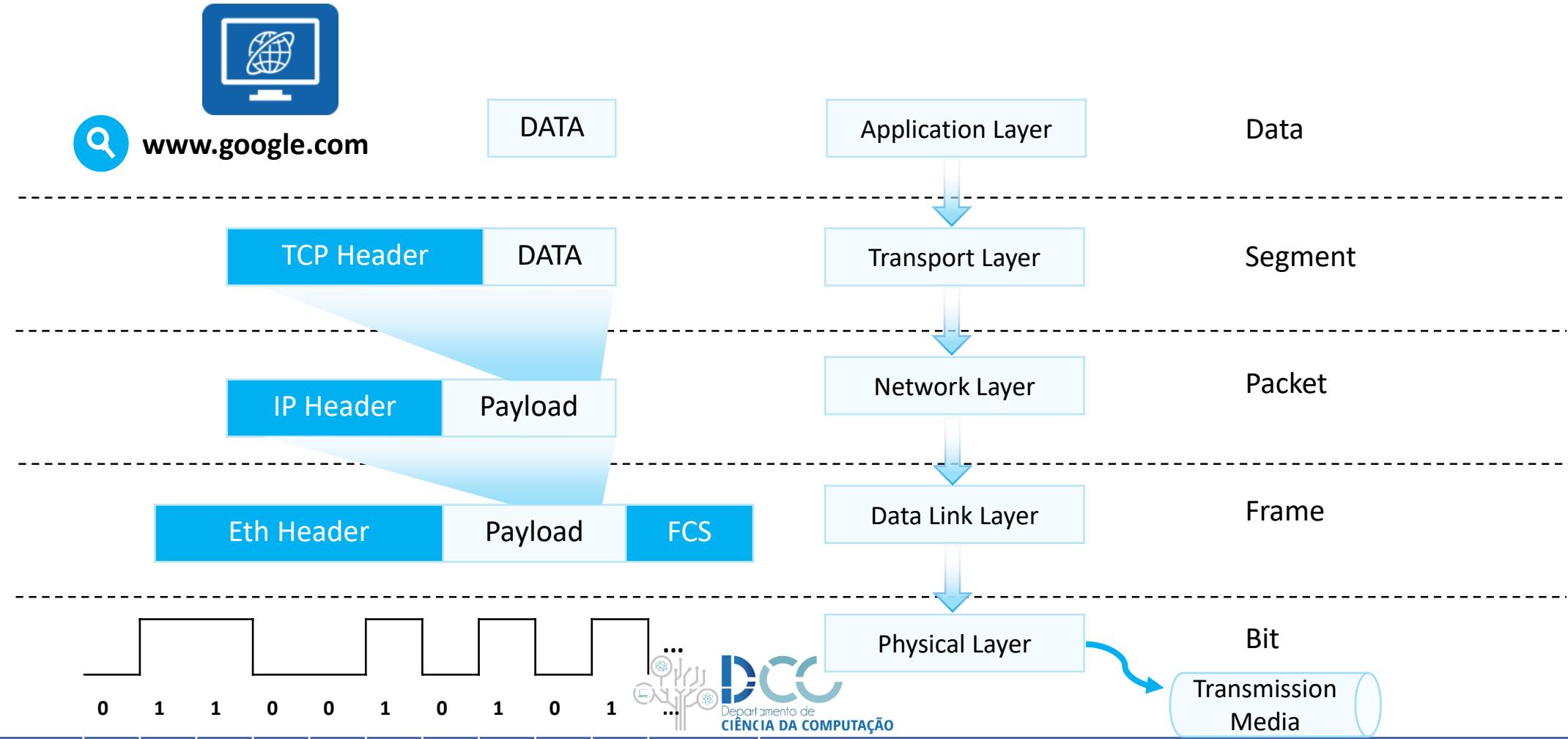


4.1.1 Protocolos em Camadas

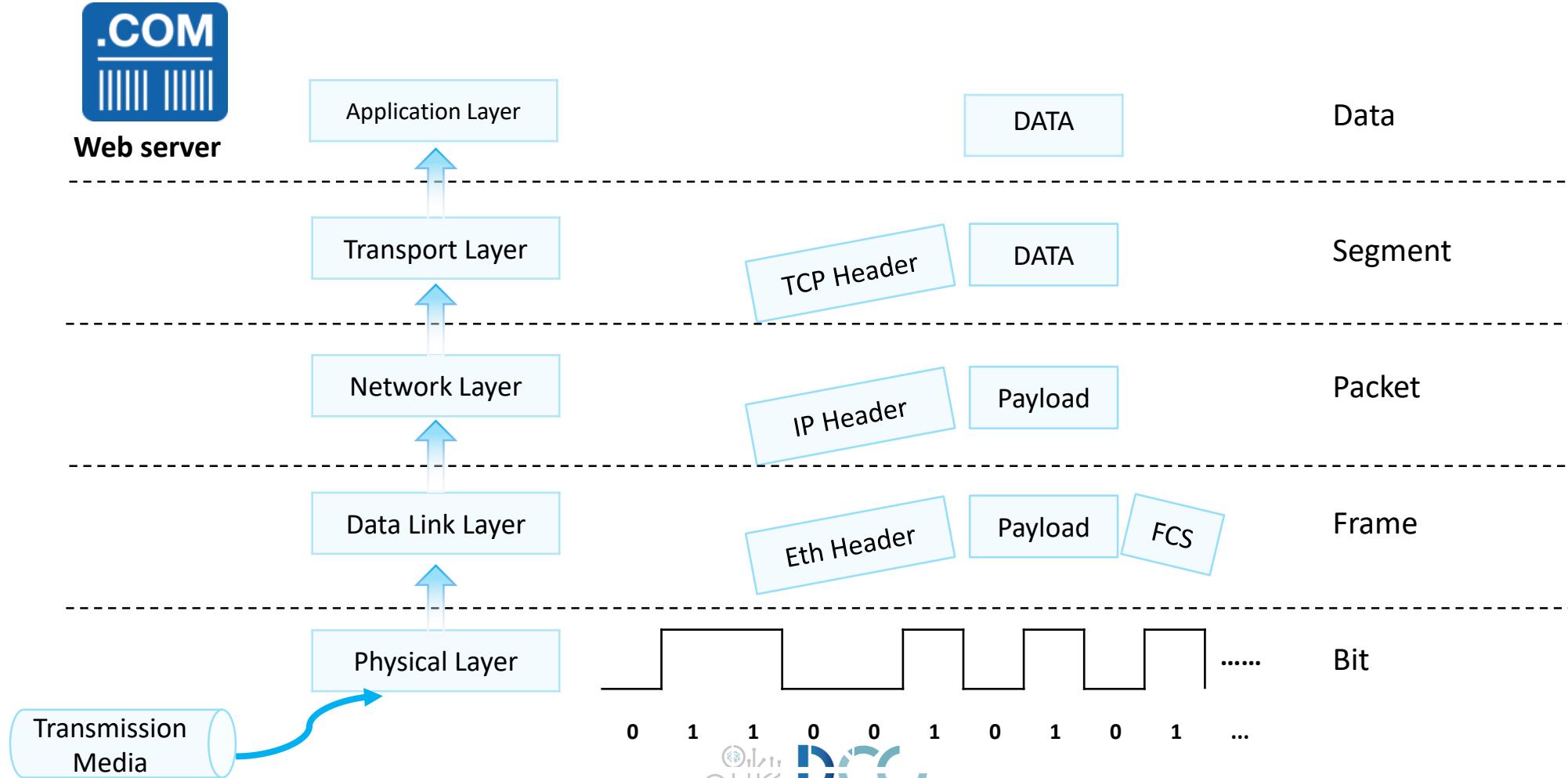
- **Encapsulamento/Desencapsulamento.**



Encapsulamento de Dados - Transmissor



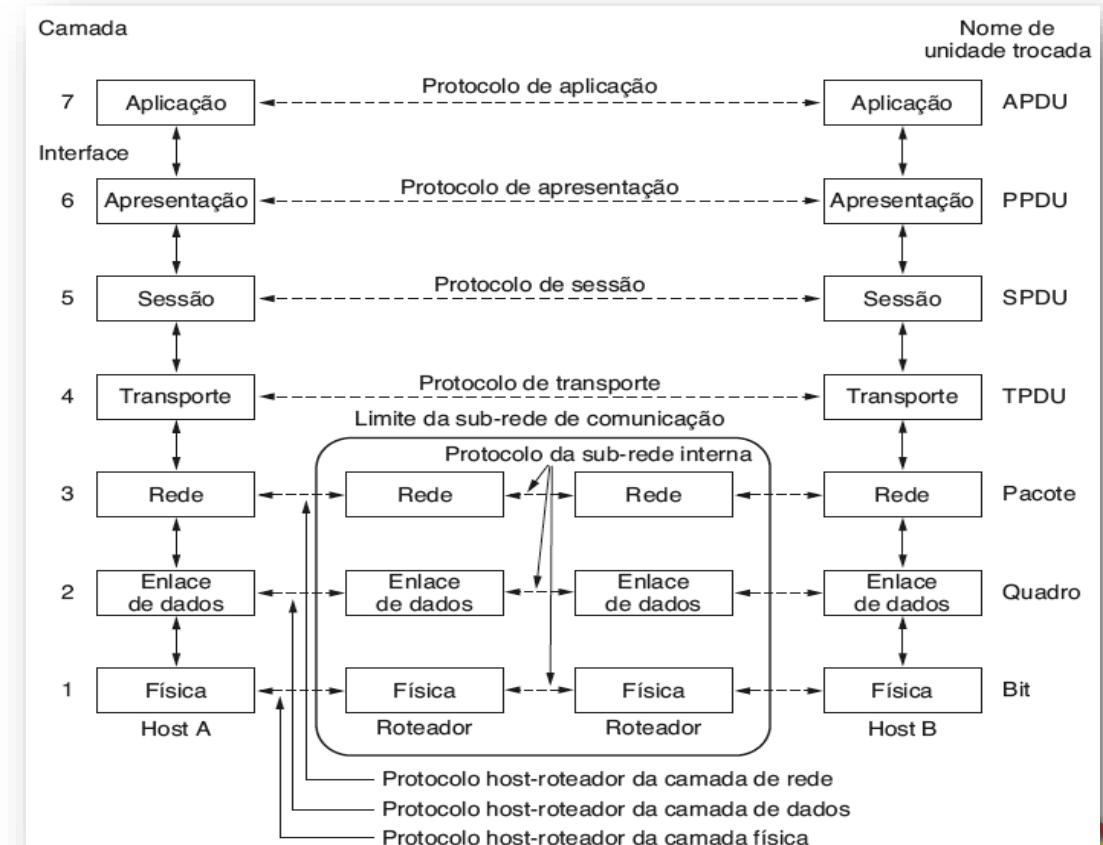
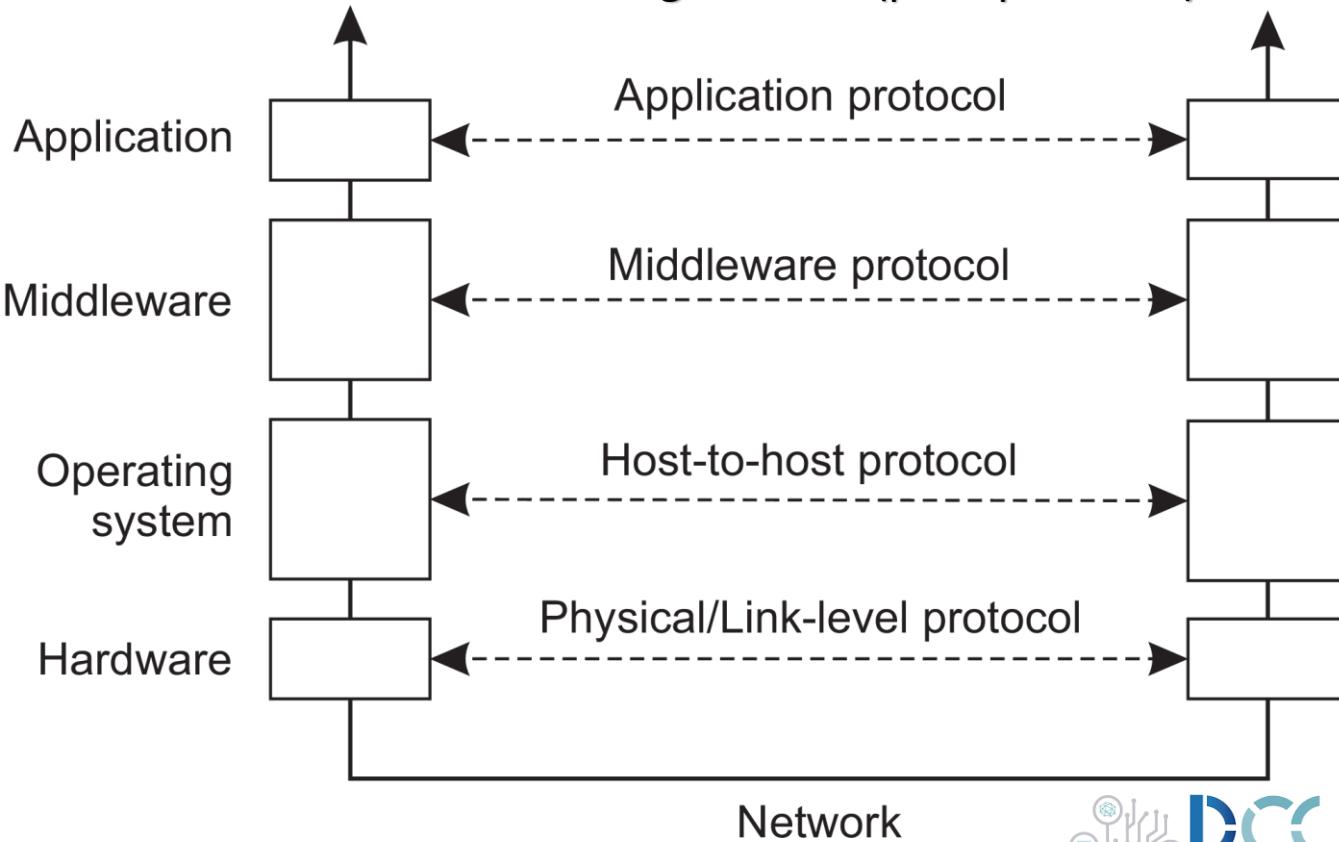
Desencapsulamento de Dados - Receptor



4.1.1 Protocolos em Camadas

- **Protocolos de Middleware**

- Middleware vive logicamente (principalmente) na camada de aplicação OSI



4.1.1 Protocolos em Camadas

- **Protocolos de Middleware: Exemplos**

- O **Domain Name System (DNS)** é um serviço distribuído usado para procurar um endereço de rede associado a um nome.
- Em termos do modelo de referência OSI, o DNS é uma aplicação
 - É logicamente colocado na camada de aplicação.
- No entanto, deve ser bastante óbvio que o DNS está oferecendo **um serviço de uso geral e independente de aplicação**.
- Indiscutivelmente, **faz parte do middleware**.

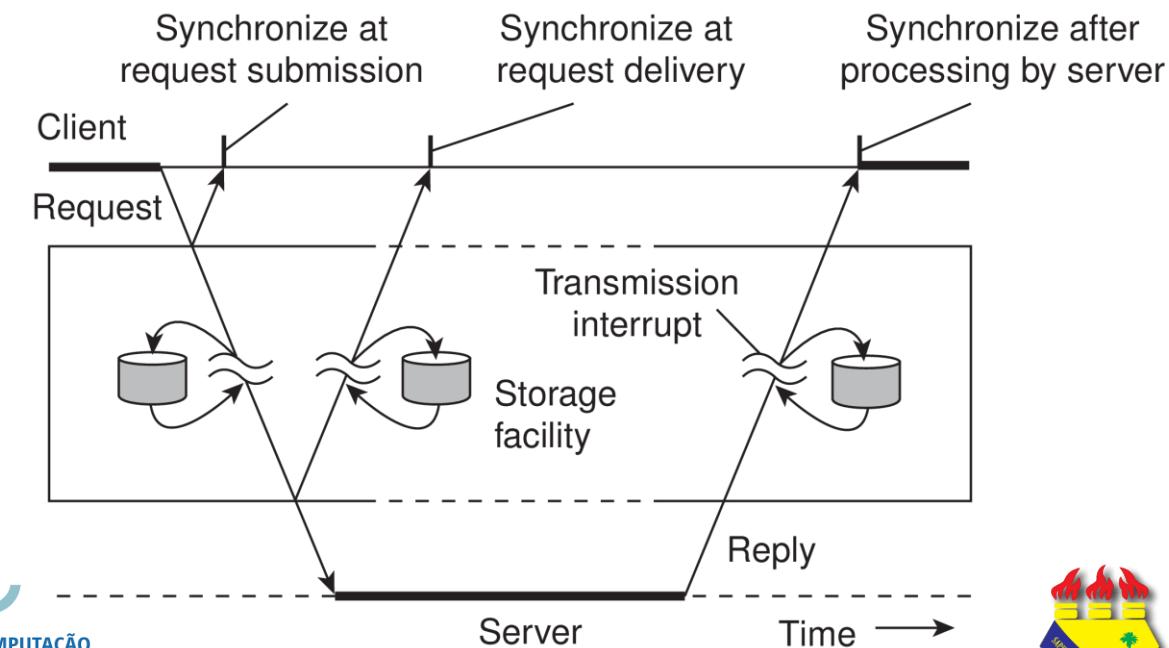
4.1.1 Protocolos em Camadas

- **Protocolos de Middleware: Exemplos**

- Como outro exemplo, existem várias maneiras de estabelecer **autenticação**, ou seja, fornecer prova de uma identidade reivindicada
- Os protocolos de autenticação não estão intimamente ligados a nenhuma aplicação específica
 - Podem ser integrados a um sistema de middleware como um serviço geral.
- Rotulados como **aplicação no modelo de referência OSI**, esses são exemplos claros que **pertencem ao middleware**.

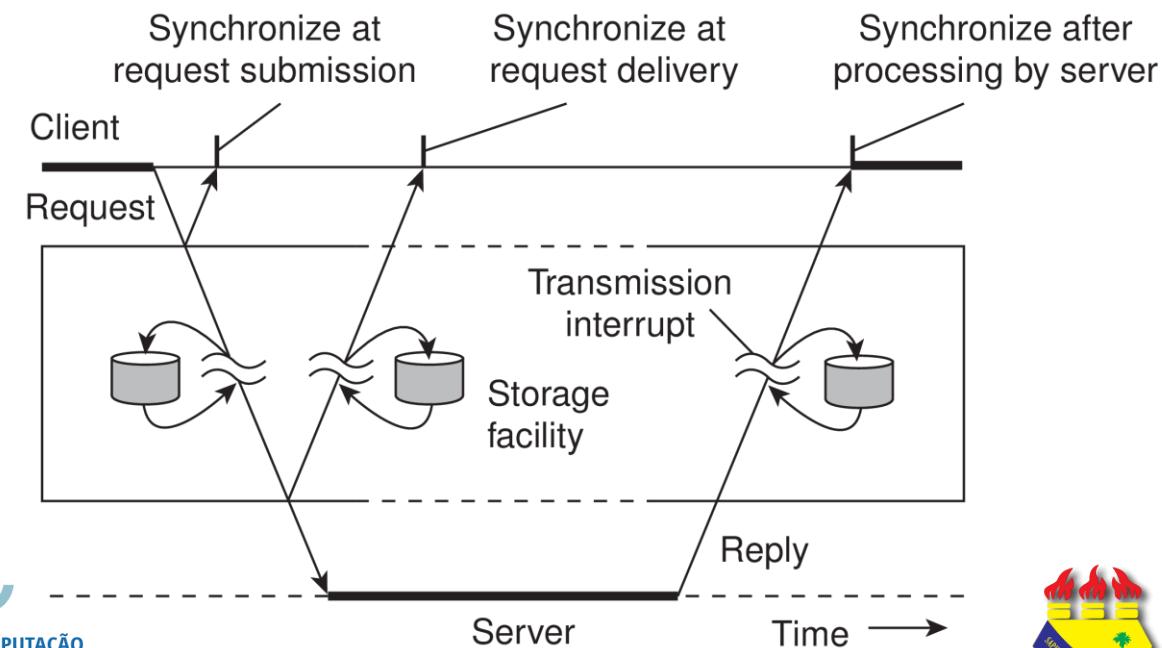
4.1.2 Tipos de Comunicação

- Vamos nos concentrar nos serviços de **comunicação de middleware de alto nível**.
- Para entender as diversas **alternativas de comunicação que o middleware pode oferecer** às aplicações, visualize o middleware como um serviço adicional na **computação cliente-servidor**.
- Por exemplo, um **sistema de e-mail** onde o núcleo do sistema de entrega de correspondência pode ser visto como um **serviço de comunicação de middleware**.



4.1.2 Tipos de Comunicação

- Um sistema de e-mail é um exemplo típico em que a **comunicação é persistente**.
- Com a comunicação persistente, uma mensagem que foi enviada para transmissão é **armazenada pelo middleware** de comunicação **pelo tempo necessário** para entregá-la ao receptor.
 - Nesse caso, o **middleware armazenará** a mensagem em uma ou várias das **instalações de armazenamento**.
 - Não é necessário que a **aplicação remetente continue a execução** após o envio da mensagem.
 - Da mesma forma, o **aplicativo receptor**.



4.1.2 Tipos de Comunicação

- Em contraste, com a **comunicação transitória**, uma mensagem é armazenada pelo sistema de comunicação **apenas enquanto o aplicativo de envio e recebimento estiver em execução**.
- Se o **middleware não puder entregar uma mensagem** devido a uma interrupção de transmissão ou porque o destinatário não está ativo no momento, ele **simplesmente será descartado**.
- Normalmente, todos os níveis de transporte dos serviços de comunicação oferecem apenas comunicação transitória.

4.1.2 Tipos de Comunicação

- Além de **ser persistente ou transitória**, a comunicação também pode ser **assíncrona ou síncrona**.
- **Comunicação assíncrona:** um remetente continua imediatamente após enviar sua mensagem para transmissão.
 - Isso significa que a mensagem é (temporariamente) armazenada imediatamente pelo middleware após o envio. Exemplo a Web.
- **Comunicação síncrona:** o remetente é bloqueado até que sua solicitação seja aceita.
 - Existem essencialmente três pontos onde a sincronização pode ocorrer:
 - O remetente pode ser bloqueado até que o middleware notifique que assumirá a transmissão da solicitação.
 - O remetente pode sincronizar até que sua solicitação seja entregue ao destinatário pretendido.
 - A sincronização pode ocorrer deixando o remetente esperar até que sua solicitação seja totalmente processada, ou seja, até o momento em que o destinatário retorna uma resposta.

Sumário

4 Comunicação

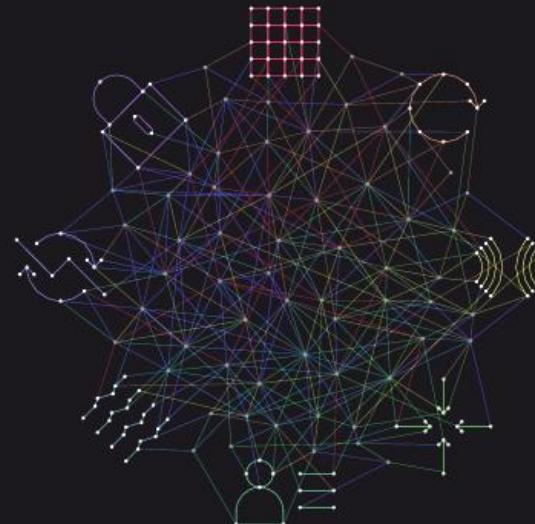
4.1 Fundamentos

4.2 Chamada de Procedimento Remoto

4.3 Comunicação Orientada a Mensagens

4.4 Comunicação Multicast

DISTRIBUTED SYSTEMS



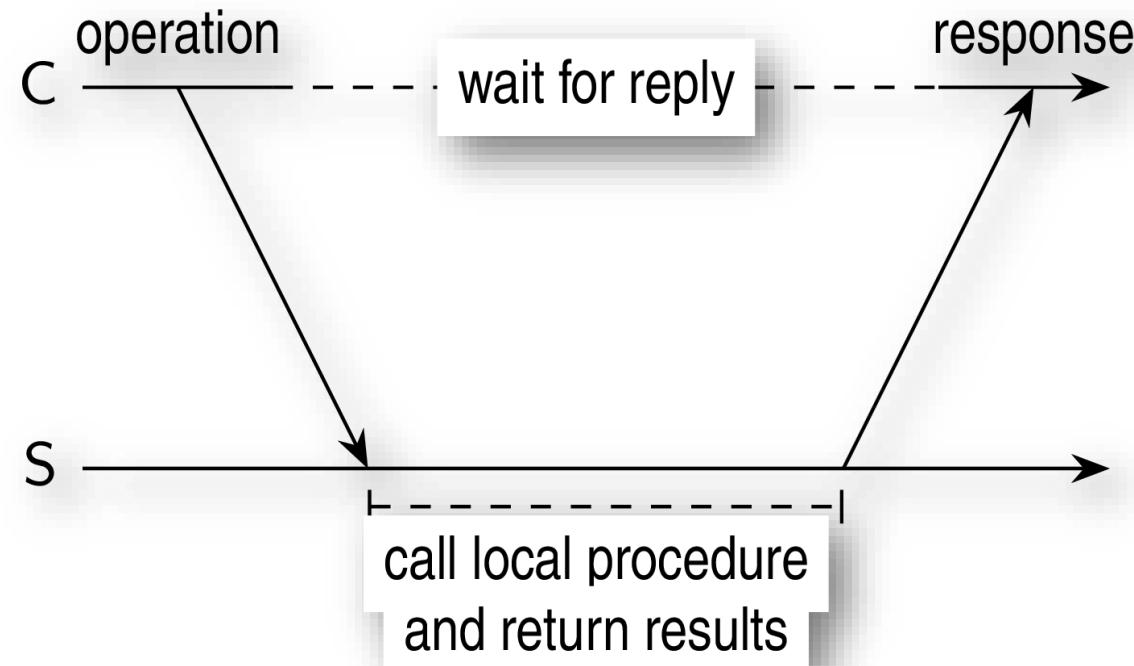
MAARTEN VAN STEEN
ANDREW S. TANENBAUM

4TH EDITION

VERSION 00

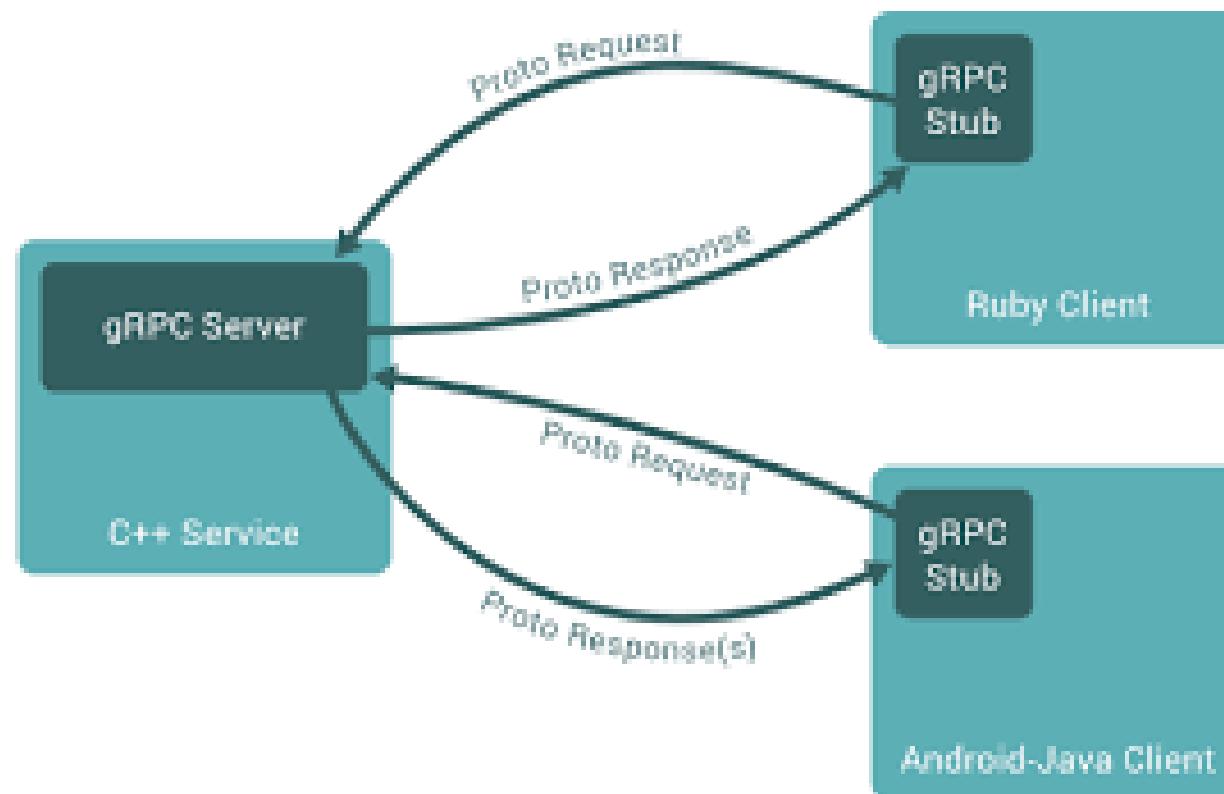
4.2 Chamada de Procedimento Remoto

- **RPC (Remote Procedure Call)** é uma tecnologia utilizada para permitir que um processo em um computador faça uma **chamada de procedimento para um processo** em outro computador, sem a necessidade de que o programador precise implementar detalhes de comunicação de rede.
- O objetivo é tornar a **comunicação entre processos remotos o mais transparente possível**, fazendo com que o **programador não precise se preocupar com os detalhes da comunicação**.



4.2 Chamada de Procedimento Remoto

- Para entender melhor a **arquitetura geral do RPC**, vamos usar como exemplo o **gRPC**, que é uma implementação moderna do RPC. <https://grpc.io/>
- A **arquitetura do gRPC** consiste em três componentes principais: **o servidor, o cliente e o protocolo.**



4.2 Chamada de Procedimento Remoto

- O servidor é responsável por implementar os **procedimentos que serão chamados pelo cliente** e expor esses procedimentos através de uma **interface de serviço**.
- O cliente, por sua vez, **faz chamadas aos procedimentos do servidor**, como se estivesse chamando **um procedimento local**.
- O protocolo define a **estrutura da mensagem** que será enviada entre o cliente e o servidor, incluindo **o formato da mensagem, as informações de cabeçalho e os dados**.

4.2 Chamada de Procedimento Remoto

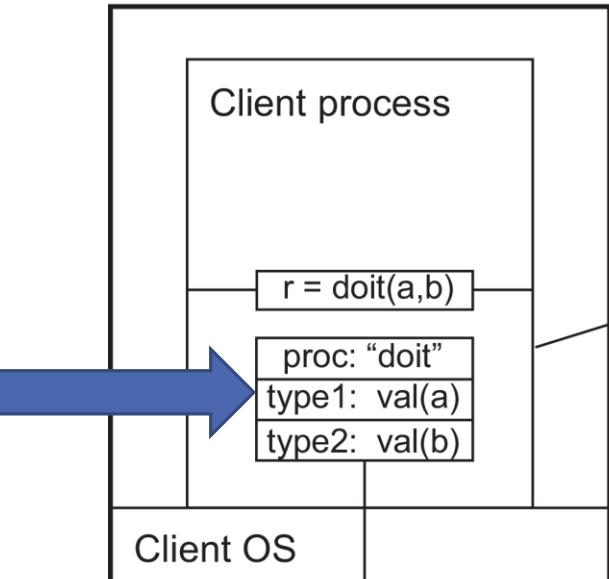
- Fluxo de execução de uma chamada de procedimento usando gRPC:
 1. O procedimento do cliente chama o stub do cliente, que empacota os parâmetros da chamada de procedimento em uma mensagem de solicitação.
 2. O stub do cliente envia a mensagem de solicitação para o servidor usando o protocolo definido.
 3. O servidor recebe a mensagem de solicitação, desempacota os parâmetros e chama o procedimento correspondente.
 4. O procedimento do servidor executa a lógica solicitada e retorna o resultado.
 5. O procedimento do servidor chama o stub do servidor, que empacota o resultado em uma mensagem de resposta.
 6. O stub do servidor envia a mensagem de resposta para o cliente usando o protocolo definido.
 7. O stub do cliente recebe a mensagem de resposta, desempacota o resultado e retorna para o procedimento do cliente.

4.2 Chamada de Procedimento Remoto

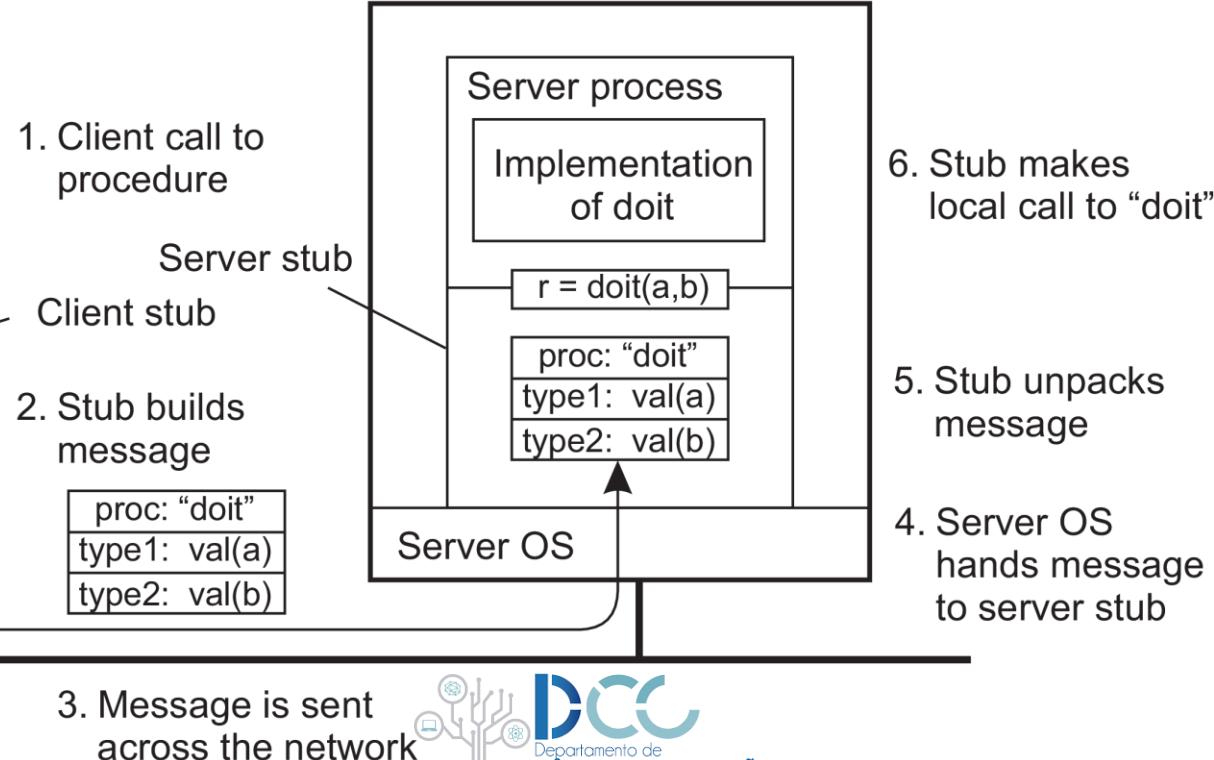
- Fluxo de execução de uma chamada de procedimento usando gRPC:

1. O procedimento do cliente chama o stub do cliente, que empacota os parâmetros da chamada de procedimento em uma mensagem de solicitação.

Client machine

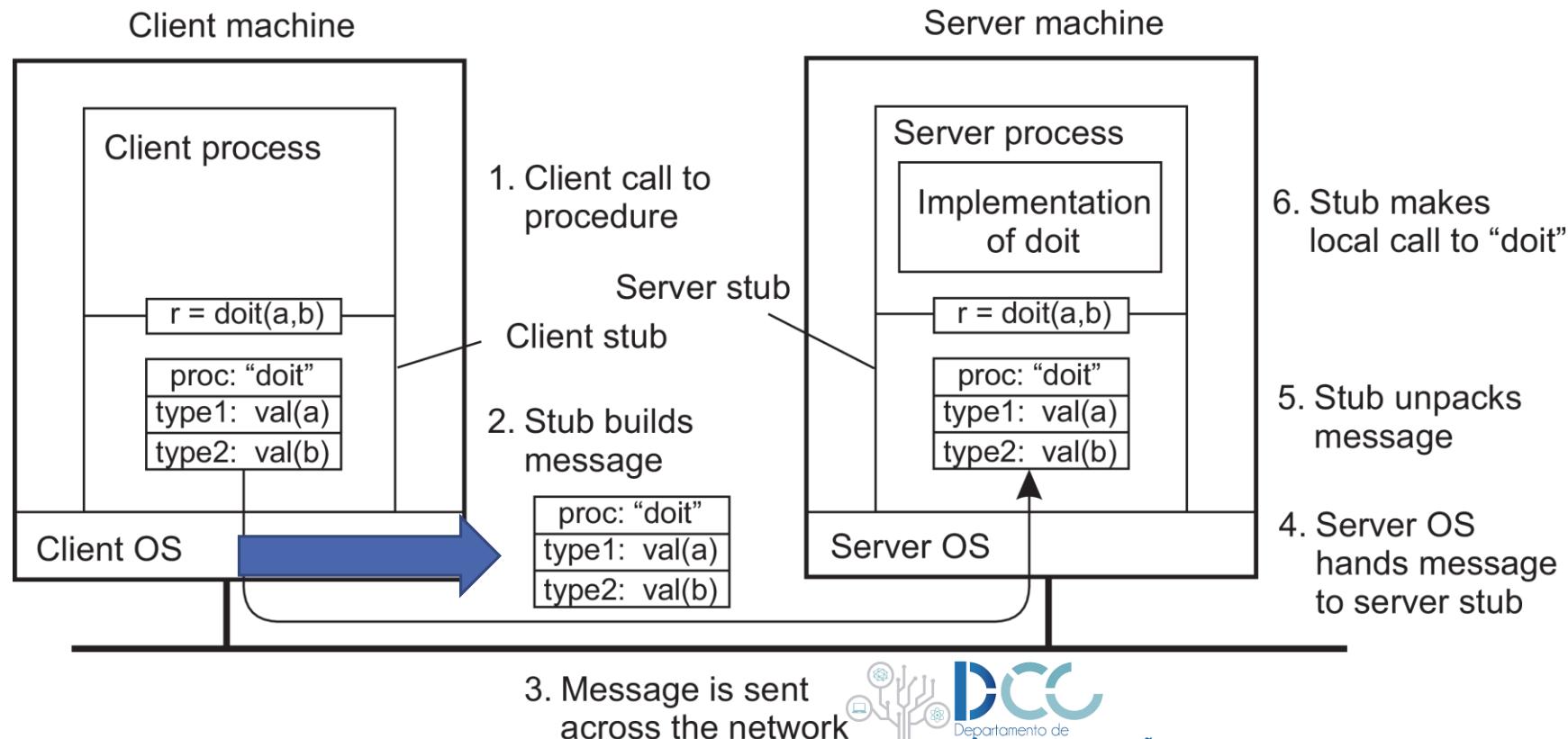


Server machine



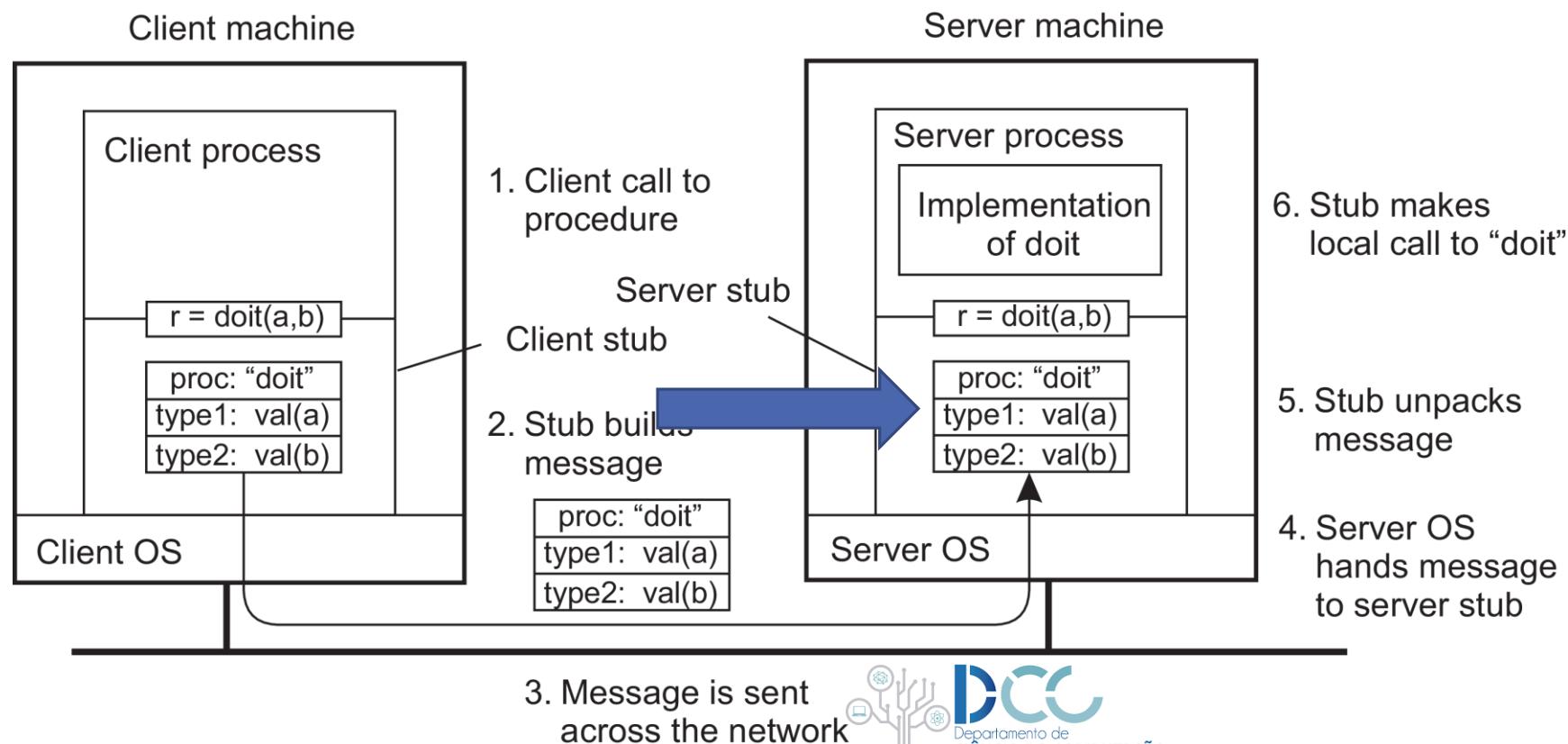
4.2 Chamada de Procedimento Remoto

- Fluxo de execução de uma chamada de procedimento usando gRPC:
 - O stub do cliente envia a mensagem de solicitação para o servidor usando o protocolo definido.



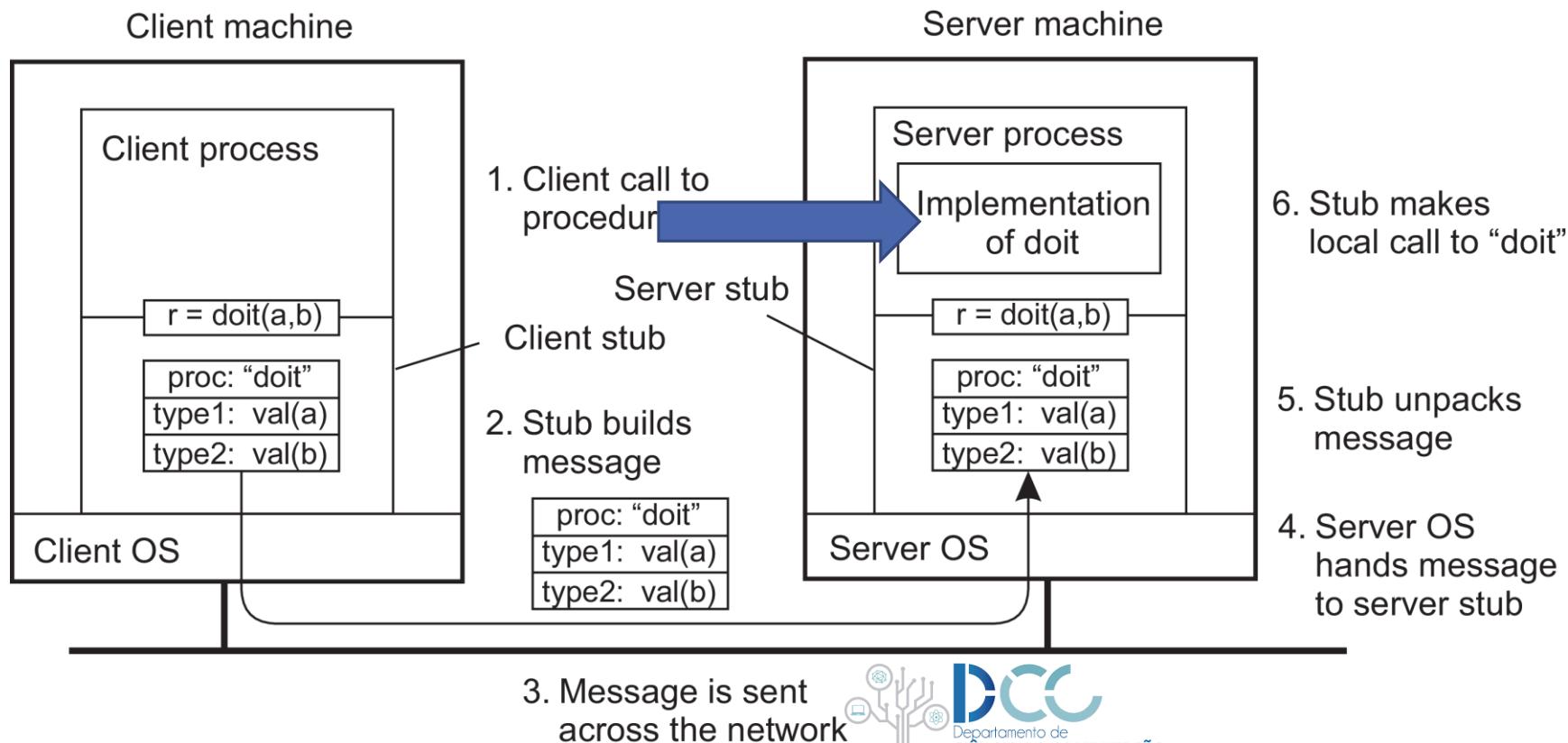
4.2 Chamada de Procedimento Remoto

- Fluxo de execução de uma chamada de procedimento usando gRPC:
 - O servidor recebe a mensagem de solicitação, desempacota os parâmetros e chama o procedimento correspondente.



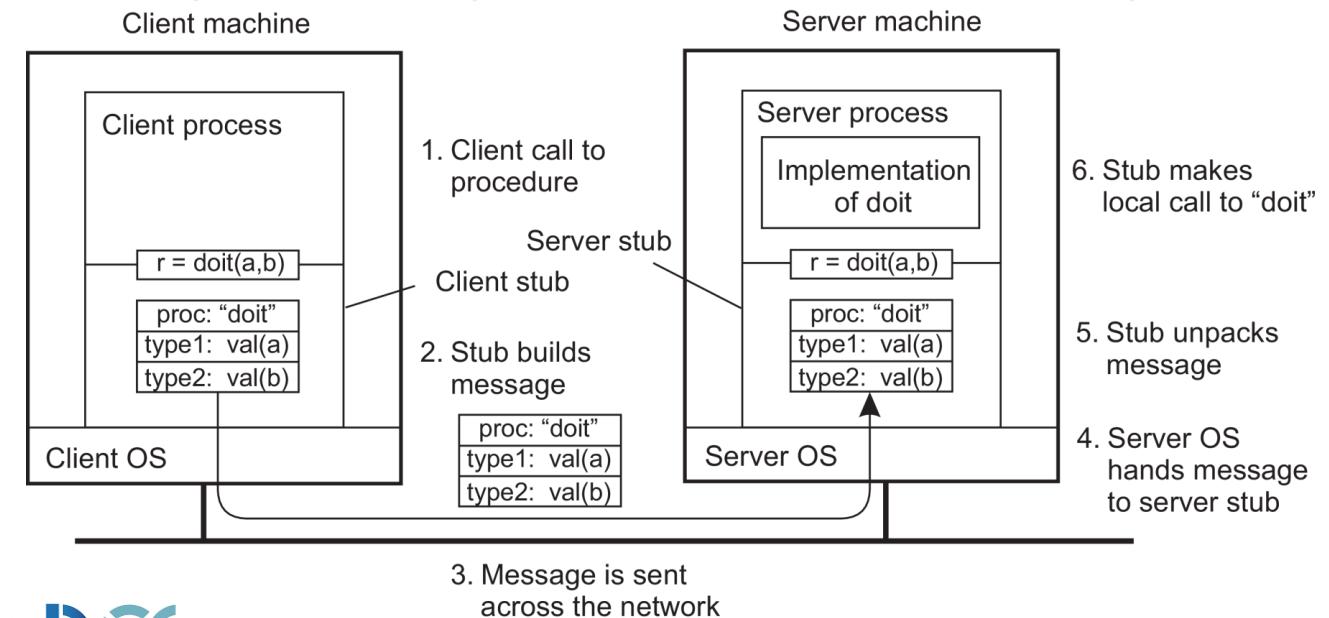
4.2 Chamada de Procedimento Remoto

- Fluxo de execução de uma chamada de procedimento usando gRPC:
 4. O procedimento do servidor executa a lógica solicitada e retorna o resultado.



4.2 Chamada de Procedimento Remoto

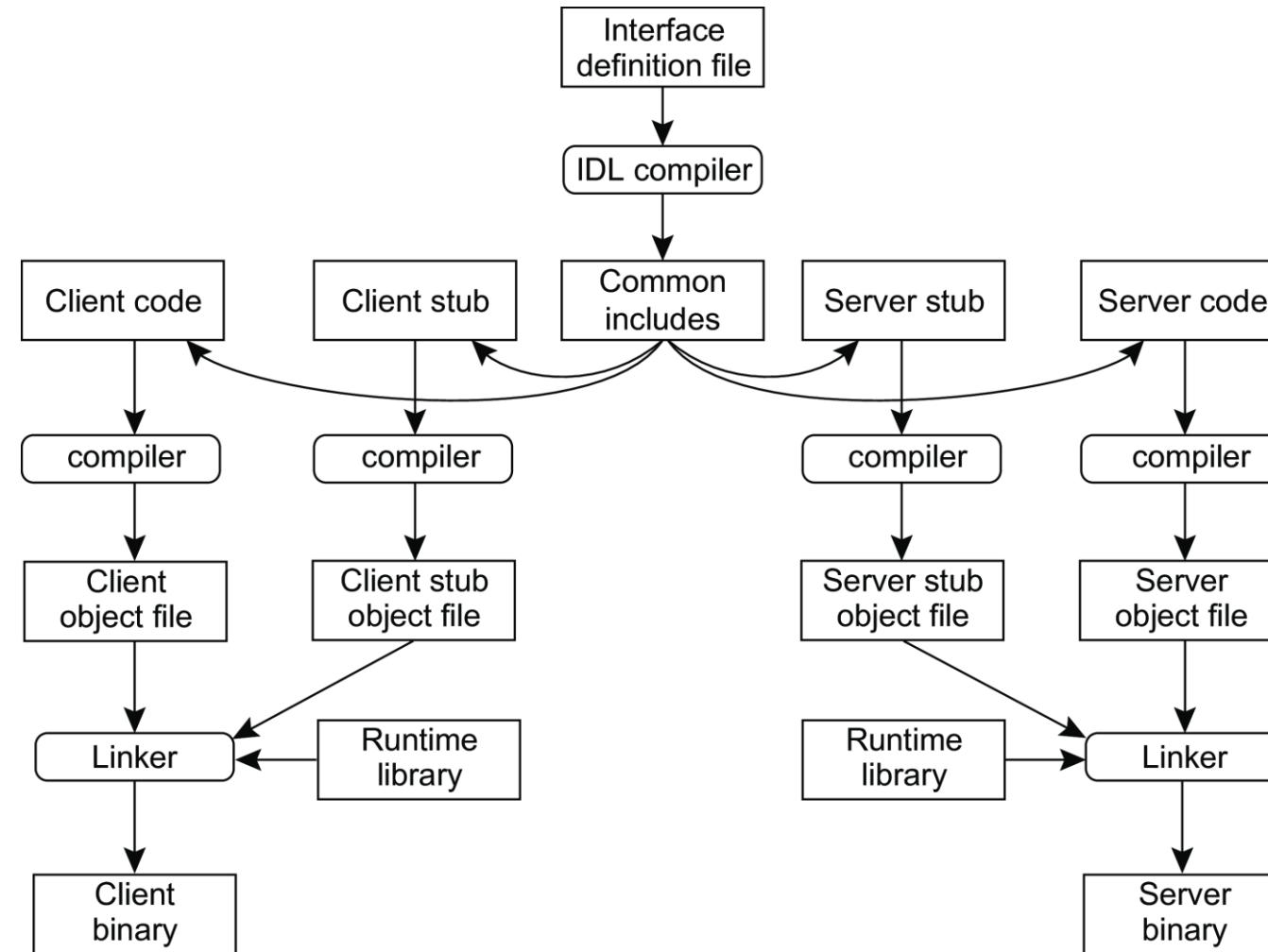
- Fluxo de execução de uma chamada de procedimento usando gRPC:
 - O procedimento do servidor chama o stub do servidor, que empacota o resultado em uma mensagem de resposta.
 - O stub do servidor envia a mensagem de resposta para o cliente usando o protocolo definido.
 - O stub do cliente recebe a mensagem de resposta, desempacota o resultado e retorna para o procedimento do cliente.



4.2 Chamada de Procedimento Remoto

- Na arquitetura RPC, a **IDL (Interface Definition Language)** é uma ferramenta essencial para garantir a interoperabilidade entre diferentes linguagens de programação.
- A IDL é usada para **definir a interface entre o cliente e o servidor**, incluindo as estruturas de dados, as operações suportadas e os tipos de parâmetros que podem ser passados entre o cliente e o servidor.
- A IDL define uma linguagem neutra de programação que é usada para **descrever a interface entre o cliente e o servidor**, permitindo que as implementações de cliente e servidor em **diferentes linguagens de programação possam ser desenvolvidas**.

4.2 Chamada de Procedimento Remoto



Continua!



obrigado!