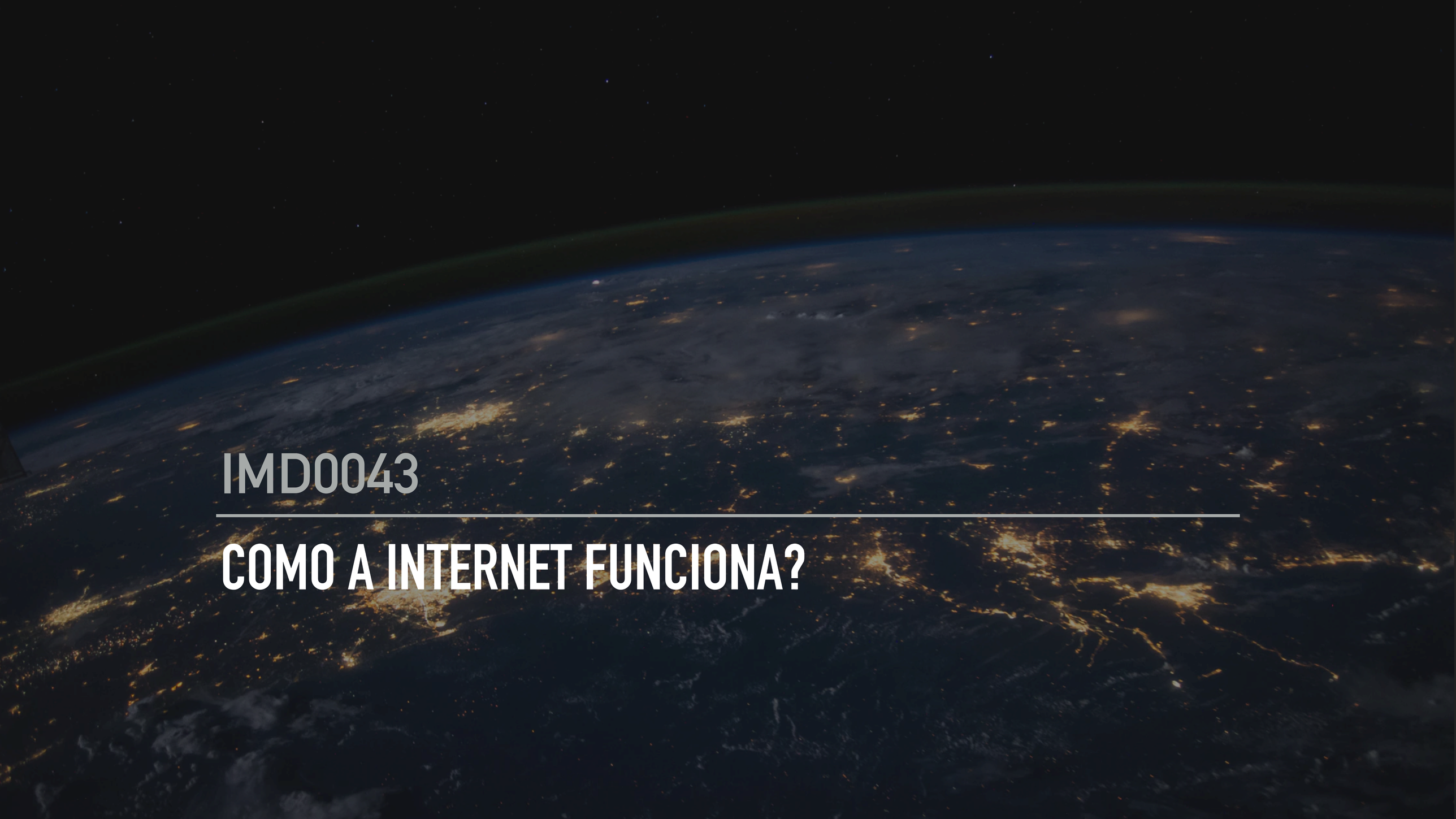


IMD0043

COMO A INTERNET FUNCIONA?

OBJETIVOS

Como a **Internet** funciona?

uma visão fim-a-fim

QUATRO PROBLEMAS FUNDAMENTAIS!

1. **Nomeação e endereçamento:** localizando o destino
2. **Roteamento:** encontrando um caminho para o destino
3. **Encaminhamento:** enviando dados para o destino
4. **Confiabilidade:** lidando com falhas, perda de pacotes, etc.

QUATRO PROBLEMAS FUNDAMENTAIS!

Nomeação, Roteamento, Encaminhamento e Confiabilidade

- Cada um é motivado por uma necessidade clara
- As soluções nem sempre são diretas
- Lembre-se dos problemas que você irá entender as soluções!
- Inicialmente iremos analisar do ponto de vista fim-a-fim

PROBLEMA FUNDAMENTAL #1: NOMEAÇAO E ENDEREÇAMENTO

Endereço de rede: onde o *host* está localizado

- Requer um endereço para o *host* de destino

Nome do *host* (*hostname*): qual *host* é esse

- Por que precisamos de um nome?
- Ex: Quando mudamos um *host* fisicamente (de um prédio para outro)
  - O endereço pode mudar, o nome não muda.
  - Similar ao seu nome e seu endereço!

NOMES VS ENDEREÇOS

Quando você acessa uma página *web*

- Insere URL no navegador (ex: www.imd.ufrn.br)
- Pacotes são enviados para o *website*
- Pacote chega na aplicação no *host* de destino

Como você chega ao *site*?

- URL é nome em nível de usuário (ex: www.imd.ufrn.br)
- Rede precisa de endereço (ex: onde fica www.imd.ufrn.br?)

É necessário mapear nomes a endereços!

- Da mesma forma que fazemos em uma agenda eletrônica

MAPEANDO NOMES A ENDEREÇOS

- Na Internet, nomeamos apenas *hosts* (ou quase isso…)
  - URLs são baseadas no nome de um *host* contendo muitas vezes o conteúdo (www.imd.ufrn.br nomeia um *host*)
- Antes de enviar pacotes para www.imd.ufrn.br, você precisa **resolver** esse nome para o endereço do *host*
- Isso é feito pelo **Domain Name System (DNS)**
  - A origem sabe o nome e mapeia o nome a um endereço usando DNS

PROBLEMA FUNDAMENTAL #2: ROTEAMENTO

Roteando pacotes através de dispositivos de rede (roteadores) até o destino

- Dado um endereço de destino, como cada roteador sabe para onde enviar o pacote para que ele chegue ao destino?
- Quando um pacote chega no roteador
  - uma **tabela de roteamento** determina para qual *link* de saída o pacote será enviado
  - calculado através de **protocolos de roteamento**

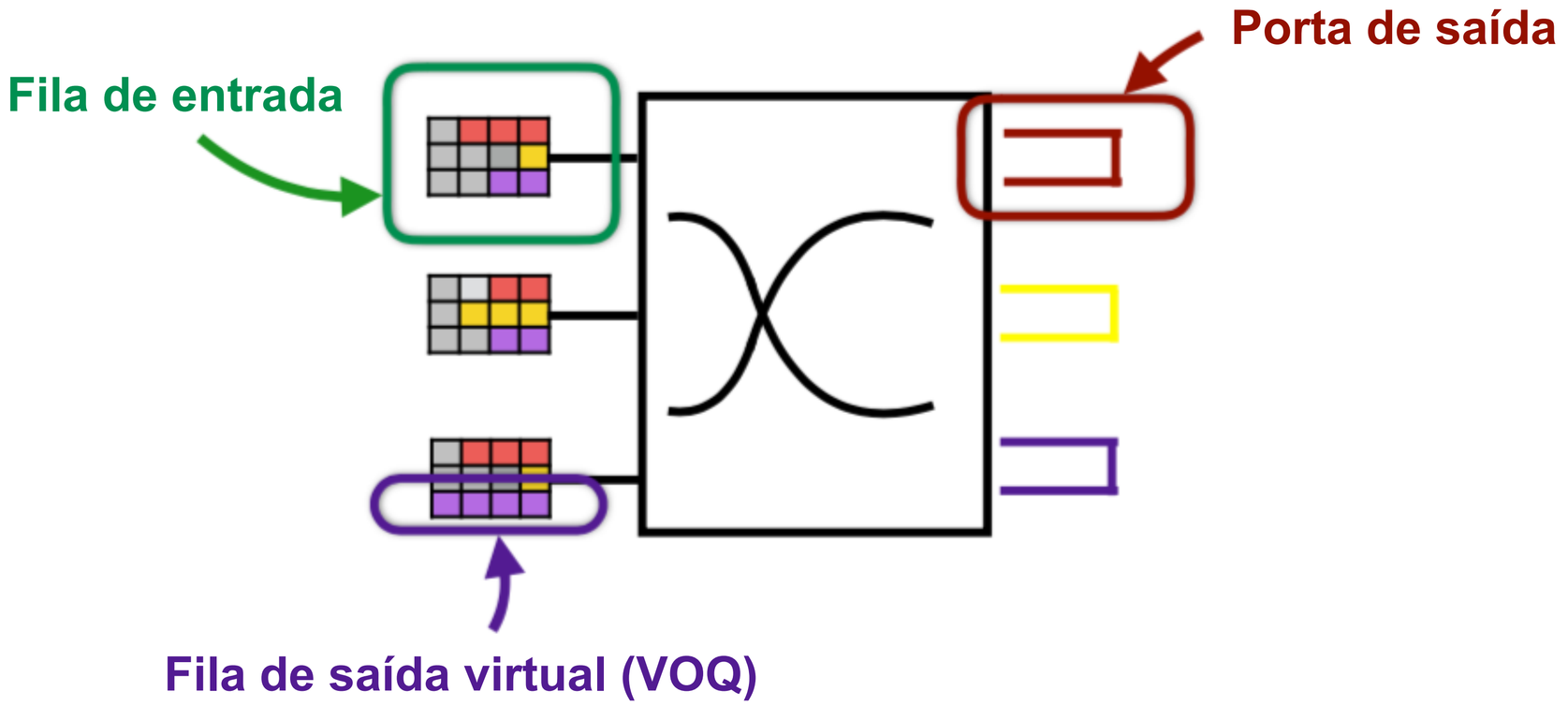
PROTOCOLOS DE ROTEAMENTO (CONCEITUALMENTE)

- Algoritmo distribuído que é executado entre roteadores
  - Distribuído significa que nenhum roteador possui uma visão "completa" da rede
- Troca de mensagens para obter informações "suficientes" sobre a topologia da rede
- Calcula caminhos através dessa topologia
- Armazena informação de **encaminhamento** em cada roteador
  - Se o pacote for destinado para X, envie para o *link* de saída L1
  - Se o pacote for destinado para Y, envie para o *link* de saída L2
  - Pacotes com destinos diferentes podem usar o mesmo *link*?
  - Chamamos isso de **tabela de roteamento**

PROBLEMA FUNDAMENTAL #3: ENCAMINHAMENTO

- **Enfileiramento**: quando um pacote chega, armazena em filas de entrada
  - Cada fila de entrada é dividida em múltiplas filas virtuais de saída
  - Uma saída virtual para cada *link* de saída
  - Quando um pacote chega:
    - Verifica o endereço de destino (*como?*)
    - Encontra o *link* no qual o pacote será encaminhado (*como?*)
    - Armazena o pacote na fila virtual de saída correspondente
- **Encaminhamento**: quando o *link* de saída está disponível
  - Pega um pacote da fila virtual de saída correspondente
  - Encaminha o pacote!

PROBLEMA FUNDAMENTAL #3: ENCAMINHAMENTO



O QUE PACOTES DEVEM CONTER PARA PERMITIR O ENCAMINHAMENTO?

Pacotes devem descrever para onde serão enviados

- Requer um endereço de destino

Pacotes devem descrever de onde estão vindo

- Para lidar com falhas, por exemplo
- Requer um endereço de origem

Pacotes devem conter dados

- Podem ser *bits* de um arquivo, imagem, aplicação, protocolo, etc.

ATRASO DE PROCESSAMENTO E ENFILEIRAMENTO

Atraso de processamento

- Cada *router* precisa decidir onde irá "colocar" o pacote
- Requer conferir o cabeçalho, por exemplo

Atraso de enfileiramento

- Depende de "quantos pacotes estão na minha frente”
- Depende do tráfego da rede
- Quanto mais carga/tráfego, maior o atraso de enfileiramento
- Em casos tráfego extremo, o resultado é perda de pacotes

PROBLEMA FUNDAMENTAL #4: CONFIABILIDADE

Como entregar pacotes de maneira confiável?

- Pacotes podem ser perdidos ("dropados") ao longo do caminho
  - *Overflow* de *buffers* em roteadores
  - Roteadores podem travar enquanto contém pacotes em *buffer*
  - *Links* podem adulterar pacotes
- Como garantir a entrega de pacotes de maneira íntegra em uma rede não confiável?
  - Ou pelo menos saber que chegaram ao destino?
- Quem é responsável por isso? A rede? O *host*? Como é implementado?

TERMINANDO NOSSA HISTORIA...

- Sabemos:
  - o endereço do *website*
  - uma rota/caminho para o destino
  - mecanismos para encaminhar os pacotes em cada *switch*/*router*
- Isso de uma maneira confiável, podemos então **enviar pacotes da origem ao destino**!
- Quando o pacote chega no *host*, o que ele faz com o pacote?
  - Para qual processo (aplicação) esse pacote deve ser enviado?
  - Se o cabeçalho do pacote só contém o endereço de destino, como o *host* sabe para onde enviar os dados?
  - Podem haver múltiplas aplicações naquele destino!

PILHA DE REDE DO DISPOSITIVO FINAL: SOCKETS E PORTAS

- Quando um processo quer acessar a rede, abre um ***socket***, que é associado a uma **porta**
- **Socket:** Mecanismo do SO que conecta processos a pilha de rede
- **Porta:** Número que identifica aquele *socket* em particular
- O número da porta é usado pelo SO para direcionar pacotes que chegam no *host*

IMPLICAÇOES PARA O CABEÇALHO DO PACOTE

Cabeçalho do pacote deve incluir:

- Endereço de destino (usado pela rede)
- Porta de destino (usado pela pilha de rede)
- Endereço de origem (usado pela rede)
- Porta de origem (usado pela pilha de rede)
- e o que mais ?

SEPARAÇAO DAS PREOCUPAÇOES

Rede:

- entrega de pacotes *host*-a-*host*

Pilha de rede (SO):

- entrega de pacotes para o *socket* apropriado (baseado na porta)

Aplicações:

- enviam e recebem pacotes
- entendem o conteúdo da carga útil (corpo) dos pacotes

HISTORIA FIM-A-FIM

- Aplicação abre um ***socket*** que permite se conectar a **pilha de rede**
- Mapeia o **nome** de um destino (ex: *website*) ao seu **endereço** usando **DNS**
- A pilha de rede na origem inclui o endereço e a **porta** de origem e de destino no **cabeçalho do pacote**
- Cada roteador constrói sua **tabela de roteamento** usando um **algoritmo de roteamento**
- Cada roteador utiliza o endereço de destino no cabeçalho do pacote para verificar o ***link* de saída** na tabela de roteamento
  - E quando está livre, **encaminha** o pacote
- Quando o pacote chega no destino, a pilha de rede usa a porta para encaminhar o pacote para a **aplicação** correta