



UNIDADE 03:

Comunicação Distribuída entre Processos

Professor: Raimundo Viégas Junior

rviegas@ufpa.br

Créditos: Prof. Josivaldo Araújo

josivaldo@ufpa.br

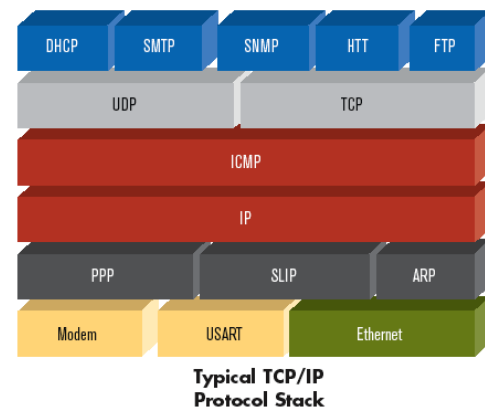
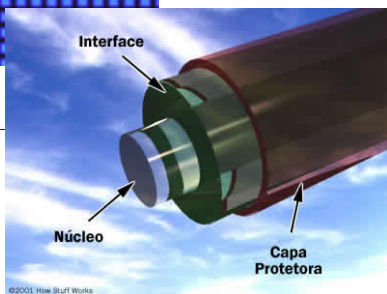
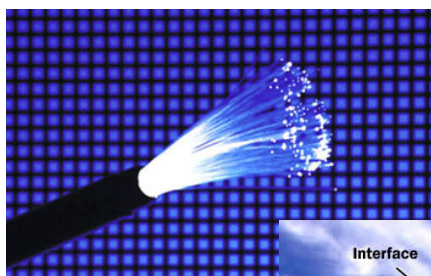
Instituto de Ciências Exatas e Naturais

Faculdade de Computação

Bacharelado em Ciência da Computação

Introdução

As redes de comunicação usadas em sistemas distribuídos são construídas a partir de uma variedade de mídias de transmissão, dispositivos de hardware e componentes de software.



A funcionalidade resultante e o desempenho disponível para o sistema distribuído e para os programas aplicativos são afetados por tudo isso.

Problemas de Interligação em Rede para SD

- As primeiras redes de computadores foram projetadas para atender alguns poucos requisitos de aplicações em rede:
 - ✓ Transferência de Arquivos;
 - ✓ *Login* remoto;
 - ✓ Correio Eletrônico
 - ✓ *Newsgroups*
- O desenvolvimento subsequente de SD, com suporte para programas aplicativos distribuídos, acesso a arquivos compartilhados e outros recursos, estabeleceu um padrão mais elevado.

Problemas de Interligação em Rede para SD

- Surgiram novos requisitos para atender as necessidades das aplicações interativas:
 - ✓ Desempenho ;
 - ✓ Confiabilidade;
 - ✓ Escalabilidade;
 - ✓ Mobilidade;
 - ✓ Segurança
 - ✓ Qualidade de Serviço.

Problemas de Interligação em Rede para SD

DESEMPENHO

- São aqueles que afetam a velocidade com que as mensagens podem ser transferidas entre dois computadores interligados;

LATÊNCIA: A diferença de tempo entre o início de uma transmissão em um processo e o início da recepção da mensagem em outro processo.

TAXA DE TRANSFERÊNCIA DE DADOS: É a velocidade com que os dados podem ser transferidos entre dois computadores em uma rede.

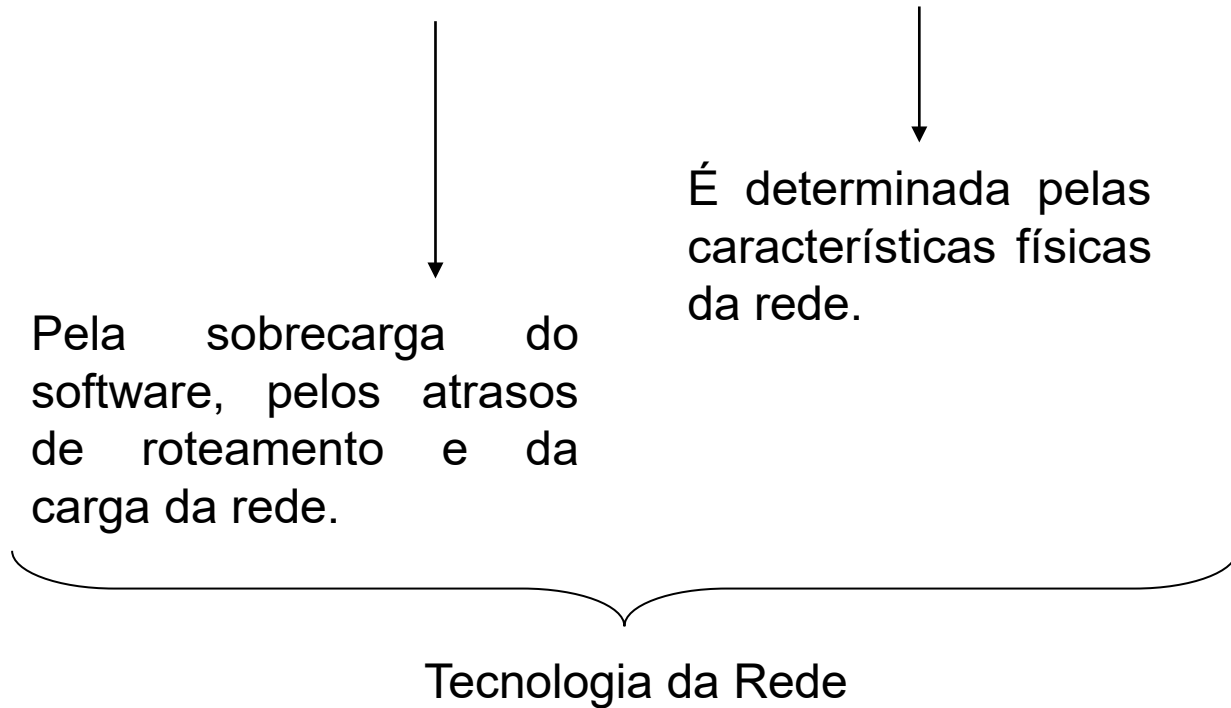
Tempo de Transmissão = latência + largura de banda/taxa de transferência

OBS.: Válido para as mensagens cuja tamanho de pacote não ultrapasse o máximo determinado pela tecnologia da rede empregada para seu envio.

Problemas de Interligação em Rede para SD

DESEMPENHO

Tempo de Transmissão = latência + largura de banda/taxa de transferência

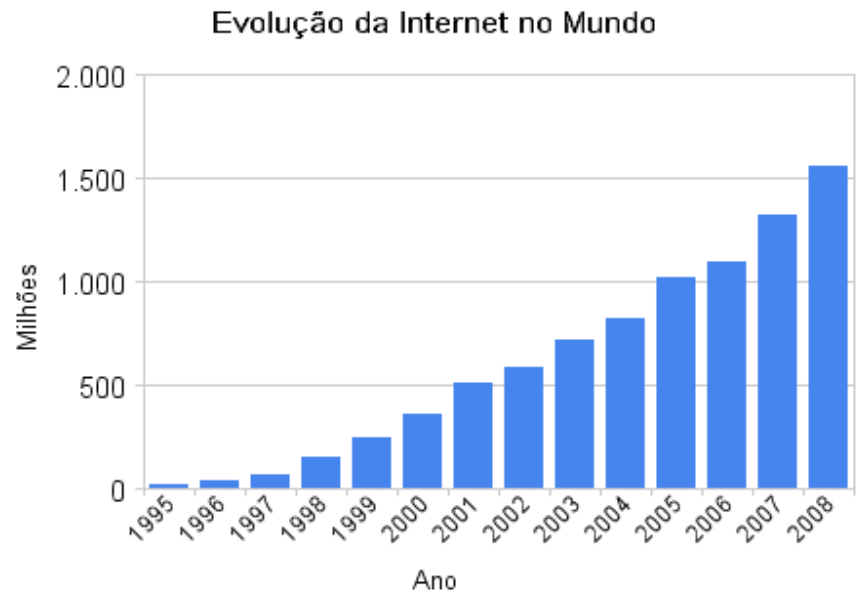


OBS.: Em S.D. é comum a transferência de muitas mensagens de pequeno tamanho:
Latência \geq Taxa de Transferência (importância)

Problemas de Interligação em Rede para SD

ESCALABILIDADE

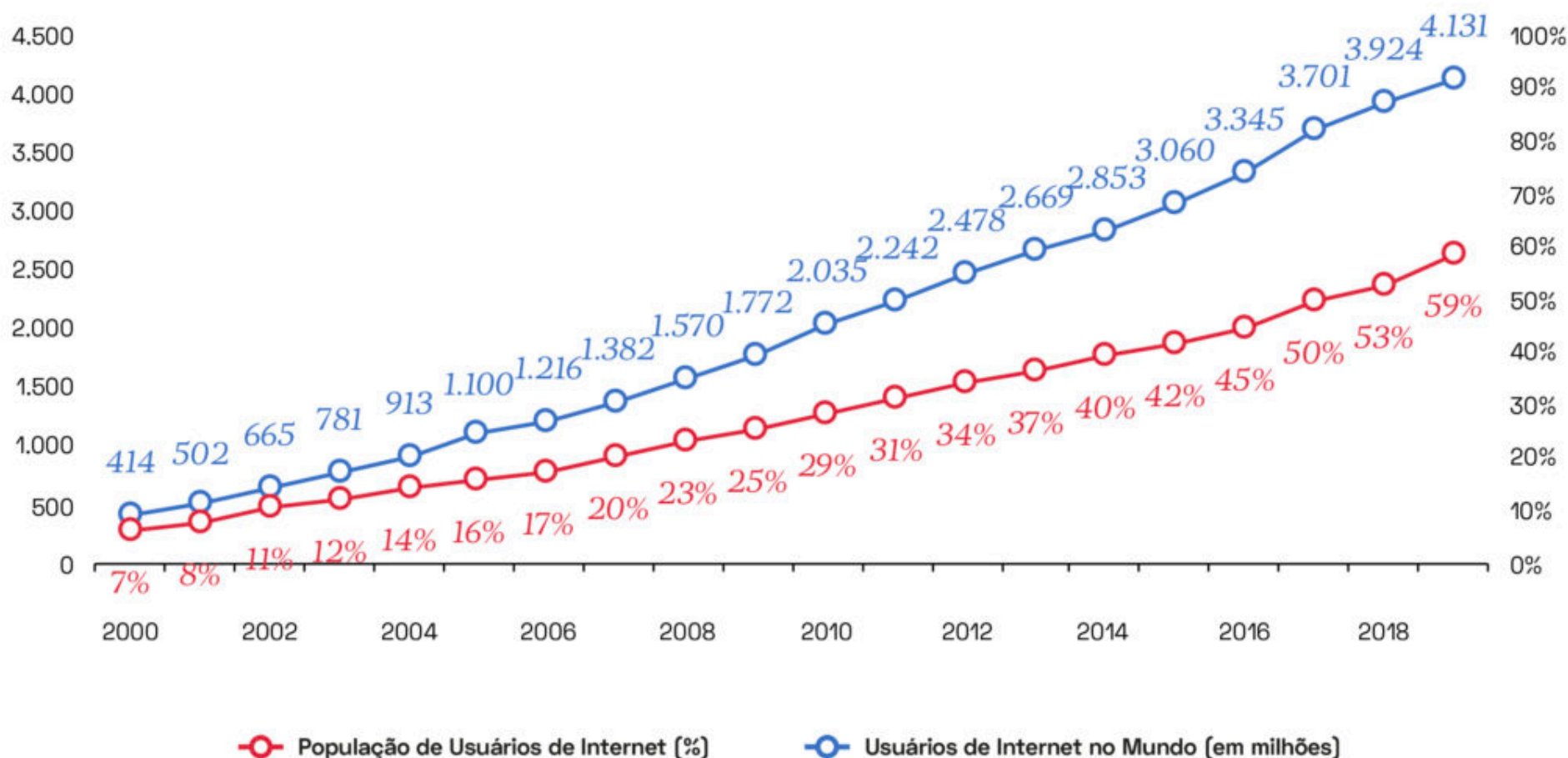
- O potencial futuro tamanho da Internet é proporcional à população do planeta.
- As tecnologias de rede em que ela é baseada não foram projetadas para suportar a atual escala, mas têm funcionado de forma satisfatória.
- Para tratar a próxima fase de crescimento, algumas mudanças substanciais nos mecanismos de endereçamento e roteamento estão em andamento.



Problemas de Interligação em Rede para SD

ESCALABILIDADE ATUALIZADO ATÉ 2018

Fonte: Internet Live Stats ([2020]), Statista (2020), World Bank ([2019]). Elaboração dos autores.





Problemas de Interligação em Rede para SD

CONFIABILIDADE

- O modelo de falhas, descreveu o impacto dos erros de comunicação. A confiabilidade da maior parte da mídia de transmissão física é muito alta;
- Quando ocorrem erros, normalmente devido a falhas no software presente no remetente ou no destinatário, ou devido a estouros de buffer.

SEGURANÇA

- O primeiro nível de defesa adotado pela maioria das organizações, é o *firewall*;
- Uma forma refinada e flexível de segurança pode ser obtida com o uso de técnicas de criptografia (Ex: VPN);
- Exceções incluem a necessidade de proteger componentes de redes como roteadores, enlaces seguros de dispositivos móveis.



Problemas de Interligação em Rede para SD

MOBILIDADE

- Os dispositivos móveis mudam frequentemente de lugar e são reconectados em diferentes pontos da rede, ou mesmo usados enquanto estão em movimento.

QUALIDADE DO SERVIÇO

- Definimos qualidade de serviço como a capacidade de atender prazos finais ao transmitir e processar fluxos de dados multimídia em tempo real;
- Os aplicativos que transmitem dados multimídia exigem largura de banda garantida e latências limitadas para os canais de comunicação que utilizam.

Tipos de Redes (2008)

	<i>Exemplo</i>	<i>Alcance</i>	<i>Largura de Banda (Mbps)</i>	<i>Latência (ms)</i>
<i>Redes Cabeadas:</i>				
LAN	Ethernet	1-2 km	10-1000	1-10
WAN	Roteamento IP	mundial	0.010-600	100-500
MAN	ATM	2-50 km	1-150	10
Interligação em Rede	Internet	mundial	0.5-600	100-500
<i>Redes sem fio:</i>				
WPAN	Bluetooth (802.15.1)	10 - 30m	0.5-2	5-20
WLAN	WiFi (IEEE 802.11)	0.15-1.5 km	2-54	5-20
WMAN	WiMAX (802.16)	5-50 km	1.5-20	5-20
WWAN	GSM, 3G (Redes Telefônicas)	mundial	0.01-02	100-500



Tipos de Redes (Atualizado pelo ChatGPT-4o) (2024)

Explicações das Tecnologias Atualizadas:

- LAN:** Inclui Ethernet Gigabit e 10 Gigabit, com alcance e largura de banda melhorados.
- WAN:** Abrange roteamento IP com um alcance global e uma faixa de largura de banda mais moderna.
- MAN:** Utiliza Ethernet sobre fibra óptica para maiores alcances e larguras de banda.
- Interligação em Rede:** Considera a internet global via fibra óptica, refletindo os avanços na capacidade de largura de banda.
- WPAN:** Atualizado para Bluetooth 5.0, com maior alcance e largura de banda.
- WLAN:** Inclui WiFi 6 (IEEE 802.11ax), que oferece melhorias significativas em alcance e desempenho.
- WMAN:** Atualizado para WiMAX 2.0 (IEEE 802.16m), proporcionando melhores métricas.
- WWAN:** Inclui tecnologias de 4G LTE e 5G NR, oferecendo maior largura de banda e menor latência.

Este quadro reflete as tecnologias mais recentes e amplamente utilizadas atualmente. Se precisar de mais informações ou alterações, sinta-se à vontade para pedir!

Tipos de Redes (Atualizado pelo ChatGPT-4o) (2024)

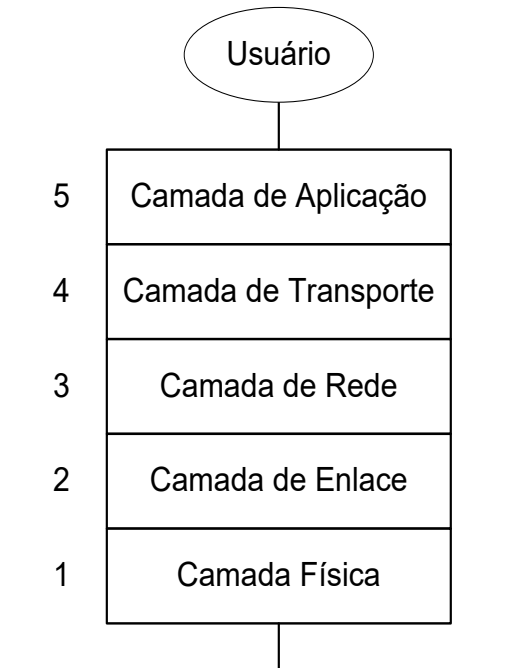
Tipo de Rede	Exemplo	Alcance	Largura de Banda (Mbps)	Latência (ms)
Redes Cabeadas				
LAN	Ethernet (Gigabit, 10 Gigabit)	100 m - 2 km	1000-10000	< 1
WAN	Roteamento IP mundial	Global	10-1000	100-500
MAN	Ethernet sobre fibra óptica	10-80 km	1000-10000	1-10
Interligação em Rede	Internet global (via fibra óptica)	Global	10000-100000	20-200
Redes sem fio				
WPAN	Bluetooth 5.0	50-200 m	2-50	1-10
WLAN	WiFi 6 (IEEE 802.11ax)	0.1-1 km	600-9600	1-10
WMAN	WiMAX 2.0 (IEEE 802.16m)	1-10 km	100-1000	5-50
WWAN	4G LTE, 5G NR	Global	10-10000	1-100

Camadas de Protocolo

- Uma especificação da sequência de mensagens que devem ser trocadas;
- Uma especificação do formato dos dados nas mensagens;



Modelo OSI

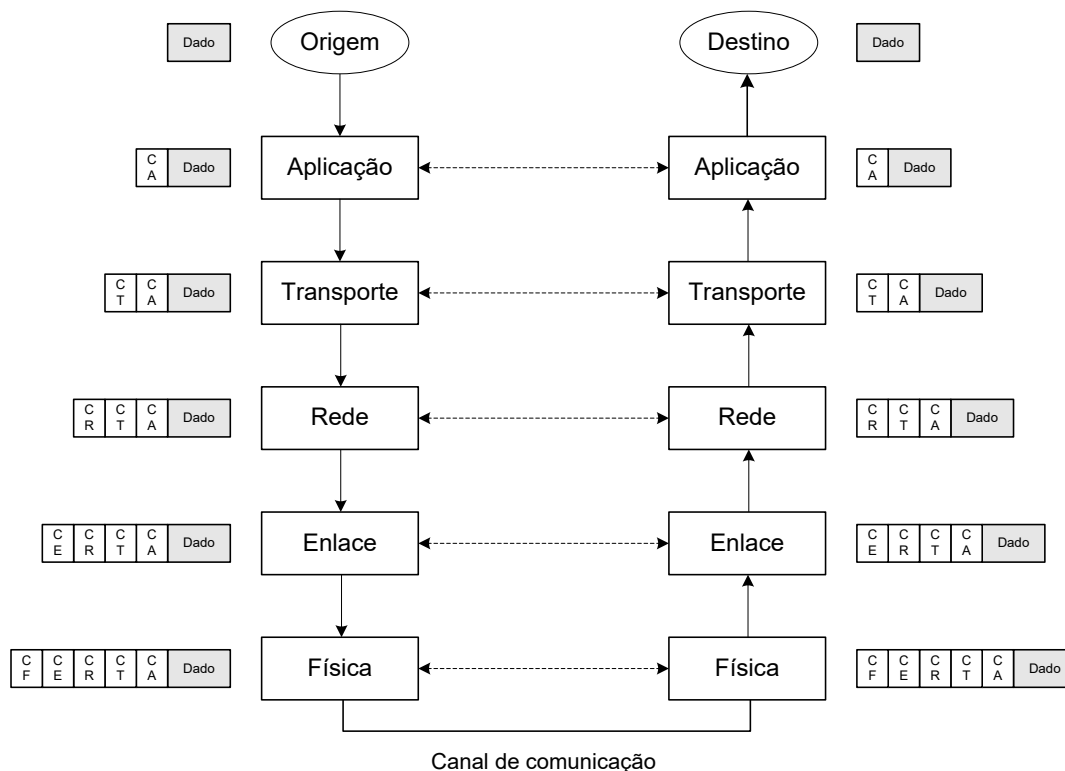


Canal de comunicação

Modelo em SD

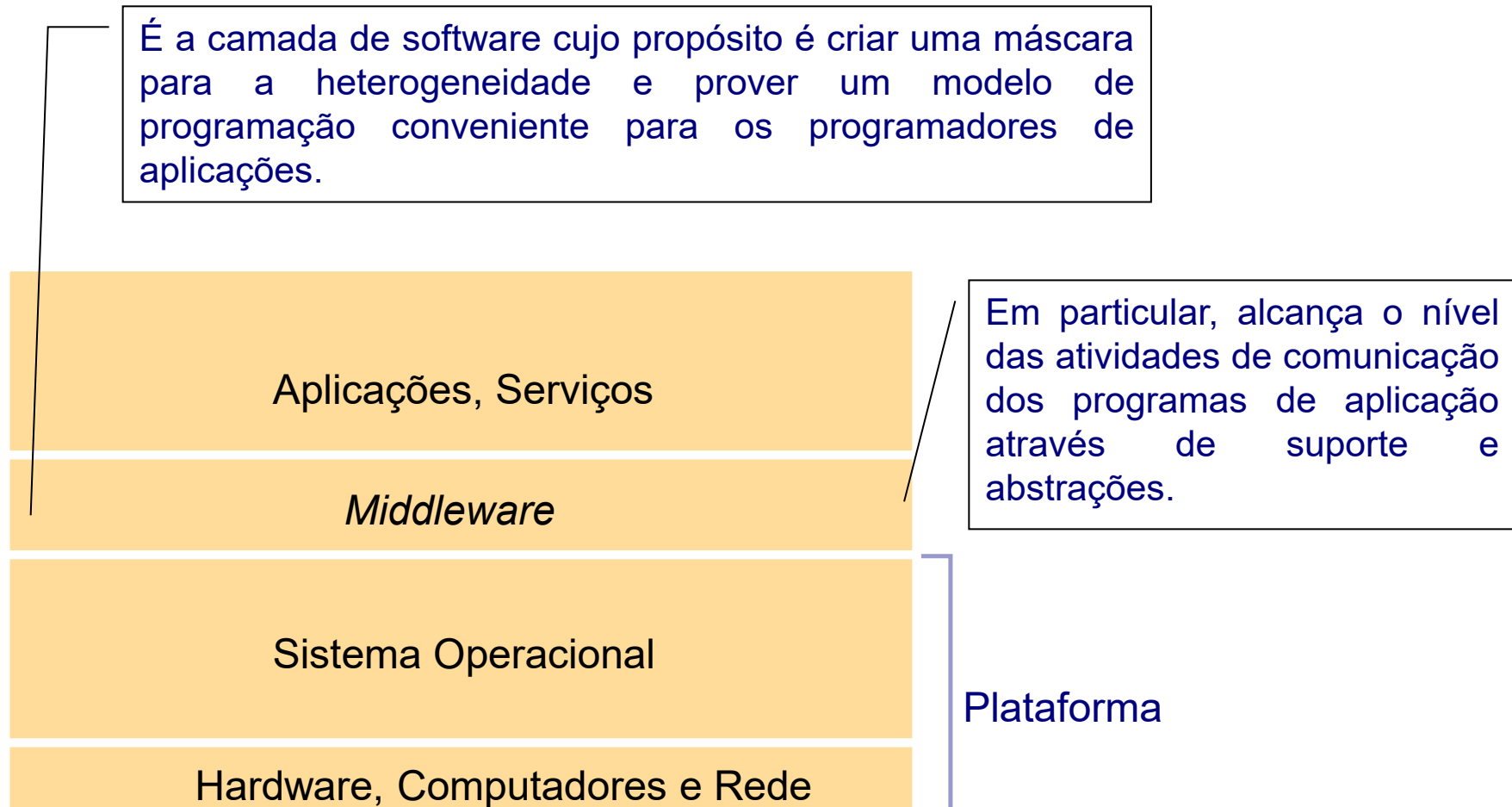
Camadas de Protocolo

- As camadas de Aplicação, Apresentação e Sessão não são claramente distinguidas na pilha de protocolos Internet. Em vez disso, as camadas de Aplicação e Apresentação são implementadas como uma única camada de *middleware*.
- A camada Sessão é integrada com a camada Transporte.



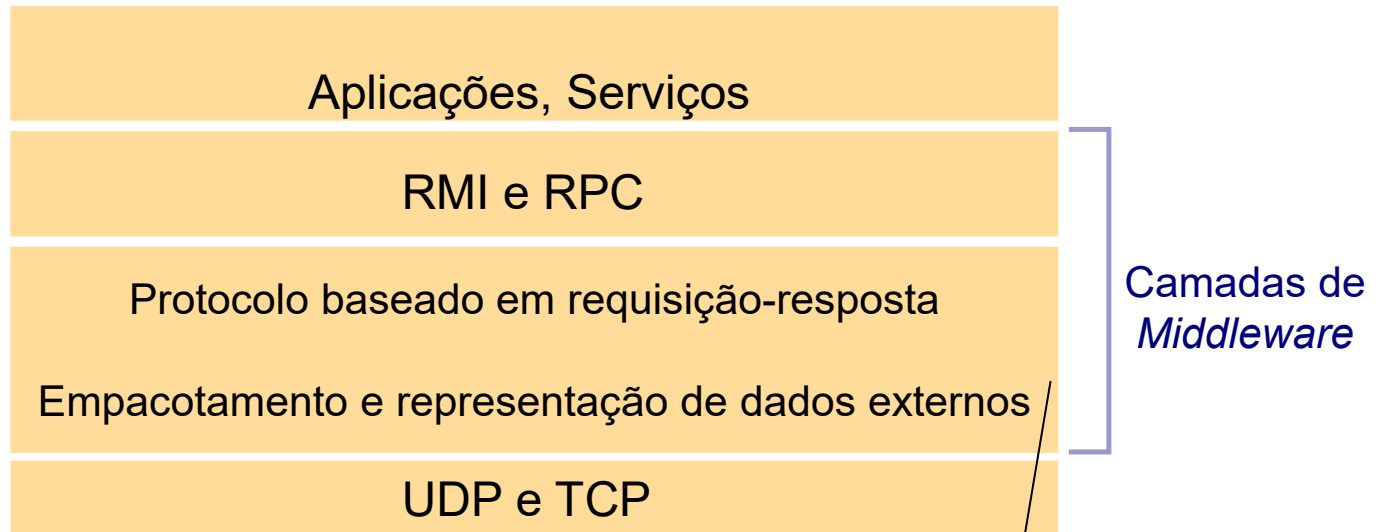
Middleware

Camadas de software e hardware em serviços de Sistemas Distribuídos.



Middleware

Camadas de Middleware.



Operações de passagem de mensagem podem ser usadas para construir protocolos para suportar funções de processo e padrões de comunicação em particular.

Processo Bloqueado e Não Bloqueado

- Em sistemas distribuídos, a comunicação baseada em envio e recebimento de mensagens pode ser classificada como **síncrona ou assíncrona**.
- Essa classificação está diretamente relacionada ao conceito de **bloqueio e não bloqueio** dos processos.

1. Comunicação Síncrona (Bloqueada)

- Na **comunicação síncrona**, o remetente espera até que o destinatário receba a mensagem e confirme o recebimento.
- Aqui, o **processo remetente é bloqueado** até que a operação de comunicação seja concluída.
- Isso significa que o processo **não pode continuar executando outras tarefas** enquanto espera pela confirmação de que a mensagem foi recebida.

1. Comunicação Síncrona (Bloqueada)

- **Envio Bloqueante:** O processo que envia a mensagem não continua sua execução até que tenha certeza de que a mensagem foi recebida pelo destinatário.
- **Recebimento Bloqueante:** O processo que está recebendo a mensagem permanece em espera até que uma mensagem chegue. Ele não pode continuar sua execução até que a mensagem seja efetivamente recebida.

2. Comunicação Assíncrona (Não Bloqueada)

- Na comunicação assíncrona, o remetente envia a mensagem e continua sua execução sem esperar por uma confirmação imediata de recebimento.
- Nesse caso, o processo é não bloqueado e pode continuar executando outras tarefas enquanto a mensagem está sendo entregue ao destinatário.
- Similarmente, o processo receptor pode verificar a chegada das mensagens de tempos em tempos (polling) ou ser notificado através de um mecanismo de interrupção ou callback.

2. Comunicação Assíncrona (Não Bloqueada)

- **Envio Não Bloqueante:** O processo que envia a mensagem pode continuar executando outras tarefas imediatamente após enviar a mensagem, sem esperar pela confirmação de recebimento.
- **Recebimento Não Bloqueante:** O processo que está recebendo a mensagem pode continuar executando outras tarefas e verificar periodicamente se uma mensagem chegou, ou ser notificado quando uma mensagem chega.



Comunicação Síncrona (Bloqueada)

■ Vantagens:

- Simplicidade na implementação e na lógica de controle de fluxo.
- Garantia imediata de entrega ou falha, facilitando o tratamento de erros.

■ Desvantagens:

- Pode levar a ineficiências e desperdício de recursos, pois o processo fica ocioso enquanto espera.
- Menos escalável em sistemas com alta latência ou alta taxa de mensagens.



Comunicação Assíncrona (Não Bloqueada)

■ Vantagens:

- Melhoria na eficiência e na utilização dos recursos, pois o processo pode continuar executando outras tarefas enquanto a comunicação ocorre.
- Melhor desempenho e escalabilidade em sistemas distribuídos de larga escala.

■ Desvantagens:

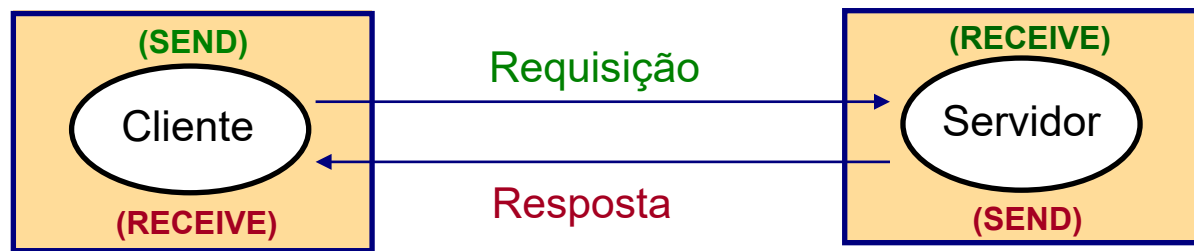
- Complexidade adicional na implementação e na lógica de controle de fluxo.
- Necessidade de mecanismos adicionais para garantir a entrega e tratamento de mensagens.

RESUMO DA COMUNICAÇÃO ENTRE PROCESSOS

- O bloqueio e não bloqueio referem-se ao comportamento dos processos durante a comunicação.
- Em comunicação bloqueada (síncrona), o processo espera até que a operação seja concluída, enquanto em comunicação não bloqueada (assíncrona), o processo pode continuar suas operações independentemente do estado da comunicação.
- A escolha entre comunicação síncrona e assíncrona depende dos requisitos específicos do sistema e do equilíbrio entre simplicidade e desempenho desejado.

Comunicação entre Processos

É definida como a passagem de mensagens entre par de processos, no qual suporta duas operações de comunicação:



DESTINO:

São Identificados pelo par:

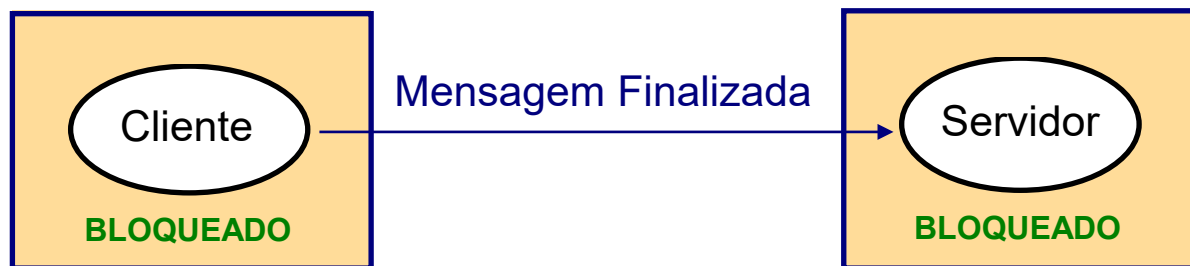
Envio de Mensagem: ENDEREÇO IP + PORTA

Comunicação entre Processos

Comunicação entre processos pode ser:

Comunicação Síncrona:

- ✓ Os processos remetente e destino são sincronizados a cada mensagem.
- ✓ Neste caso, *send* e *receive* são operações que causam bloqueio.



Comunicação entre Processos

Comunicação entre processos pode ser:

Comunicação Assíncrona:

- ✓ O uso da operação *send* é não bloqueante. Já a operação *receive* pode ter variantes com e sem bloqueio.

□ SEM BLOQUEIO:

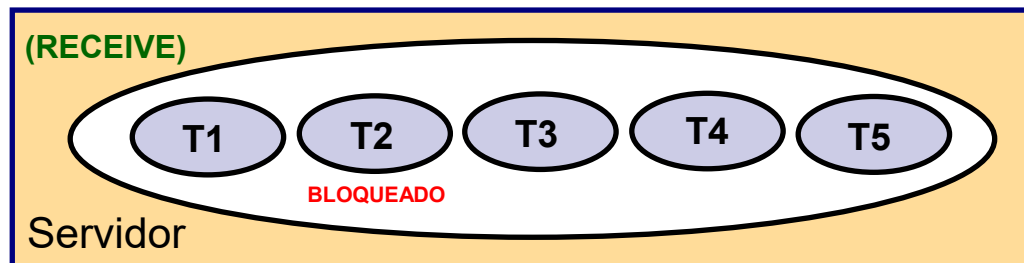


Comunicação entre Processos

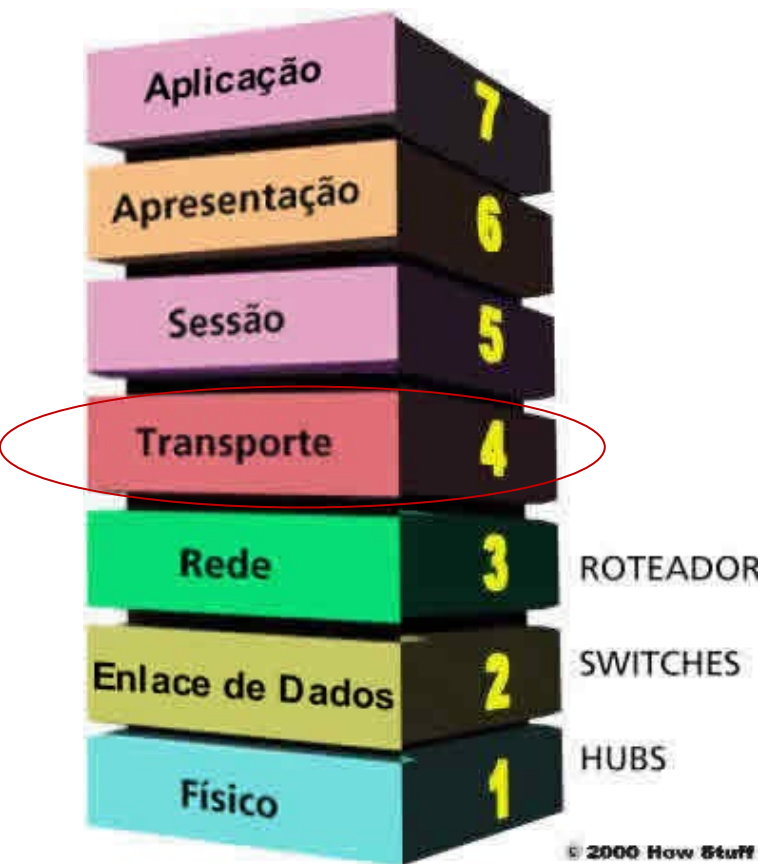
COM BLOQUEIO:



OBS.: Se não forem cuidadosamente projetados, sistemas que utilizam recepção bloqueante podem estar sujeitos a deadlocks, onde uma thread espera indefinidamente por uma mensagem que nunca chega, especialmente em ambientes com múltiplas threads que dependem de respostas entre si.



Comunicação entre Processos



- ✓ A camada de rede é responsável pelo roteamento de datagramas entre dois hosts: comunicação *host-to-host*;
- ✓ A camada de enlace é responsável pela transmissão de frames entre dois nós adjacentes conectados por um enlace físico: comunicação nó-a-nó;
- ✓ A comunicação na Internet não é definida como apenas a troca de dados entre dois nós ou entre dois hosts;
- ✓ A comunicação real ocorre entre dois processos;

Comunicação entre Processos

- ✓ A camada de transporte é responsável pela comunicação entre processos;
 - ✓ A entrega de um pacote, parte de uma mensagem, de um processo para outro;
 - ✓ Dois processos se comunicam em uma relação cliente/servidor;
- ❑ Endereçamento:
- Camada de Enlace: MAC;
 - Camada de Rece: IP;
 - Camada de Transporte: N° de Porta;



Comunicação entre Processos

- ✓ No modelo Internet, os números de porta são inteiros de 16 bits entre 0 e 65.535;
- ✓ Um programa cliente define para si mesmo um número de porta, escolhido de forma aleatória pelo software da camada de transporte;
- ✓ Um processo servidor também deve definir seu número de porta, no entanto, não pode ser escolhido de forma aleatória;
- Faixa de Endereços IANA (*Internet Assigned Number Authority*);
 - Portas Conhecidas: Na faixa de 0 a 1023;
 - Portas Registradas: Na faixa de 1024 a 49.151;
 - Portas Dinâmicas: Na faixa de 49.152 a 65.535;

Comunicação entre Processos

DESTINO FINAL DOS DADOS:

São Identificados pelo par:

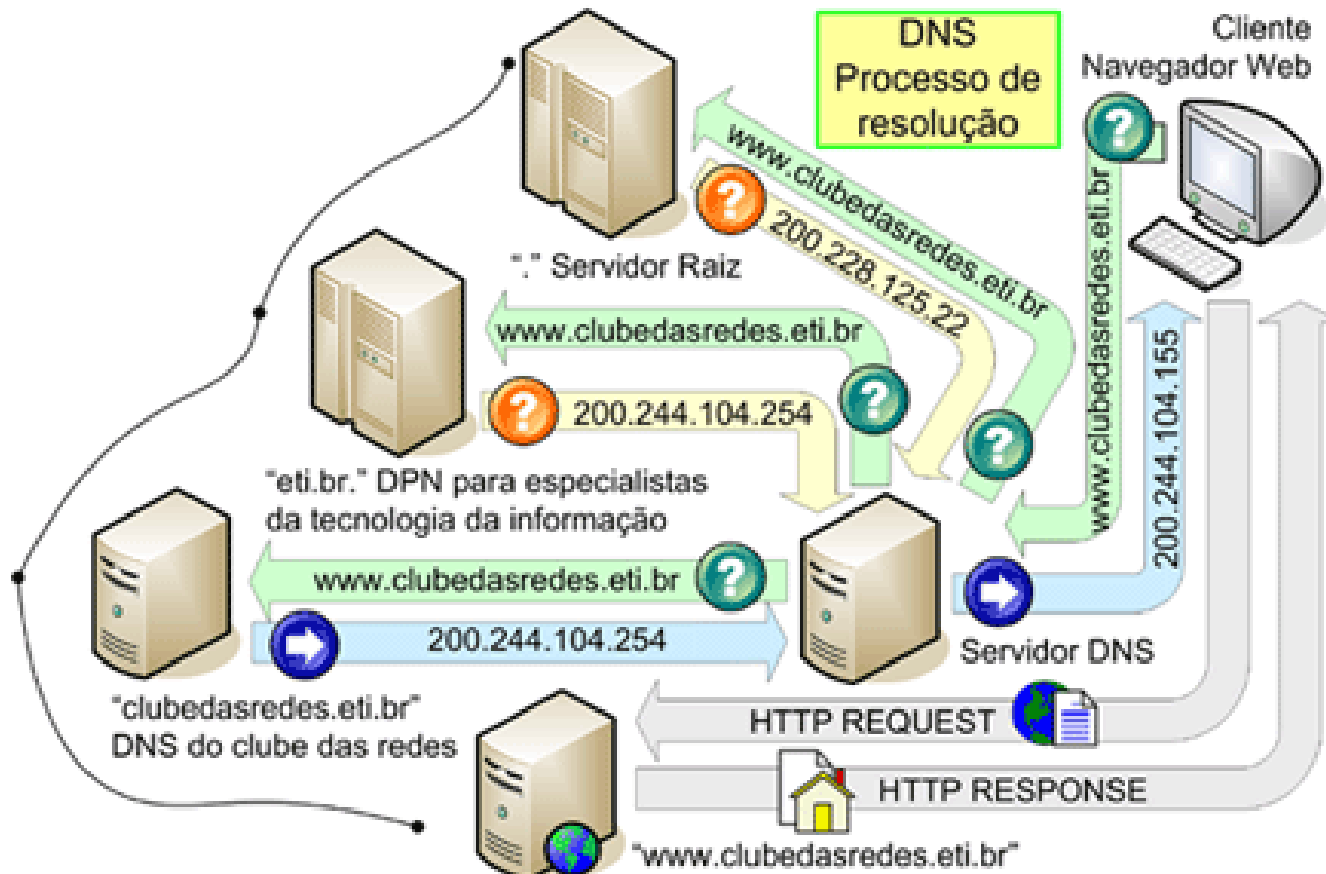
Envio de Mensagem: ENDEREÇO IP + PORTA

Para proporcionar TRANSPARÊNCIA de localização:

1. Os programas clientes se referem aos serviços pelo nome e usam um servidor de nomes (DNS);
2. O sistema operacional fornece identificadores independentes da localização para os destinos das mensagens, mapeando-os em um endereço de nível mais baixo para elas serem entregues nas portas;

Comunicação entre Processos

SERVIDORES DE NOMES (DNS)



Servidor DNS Local

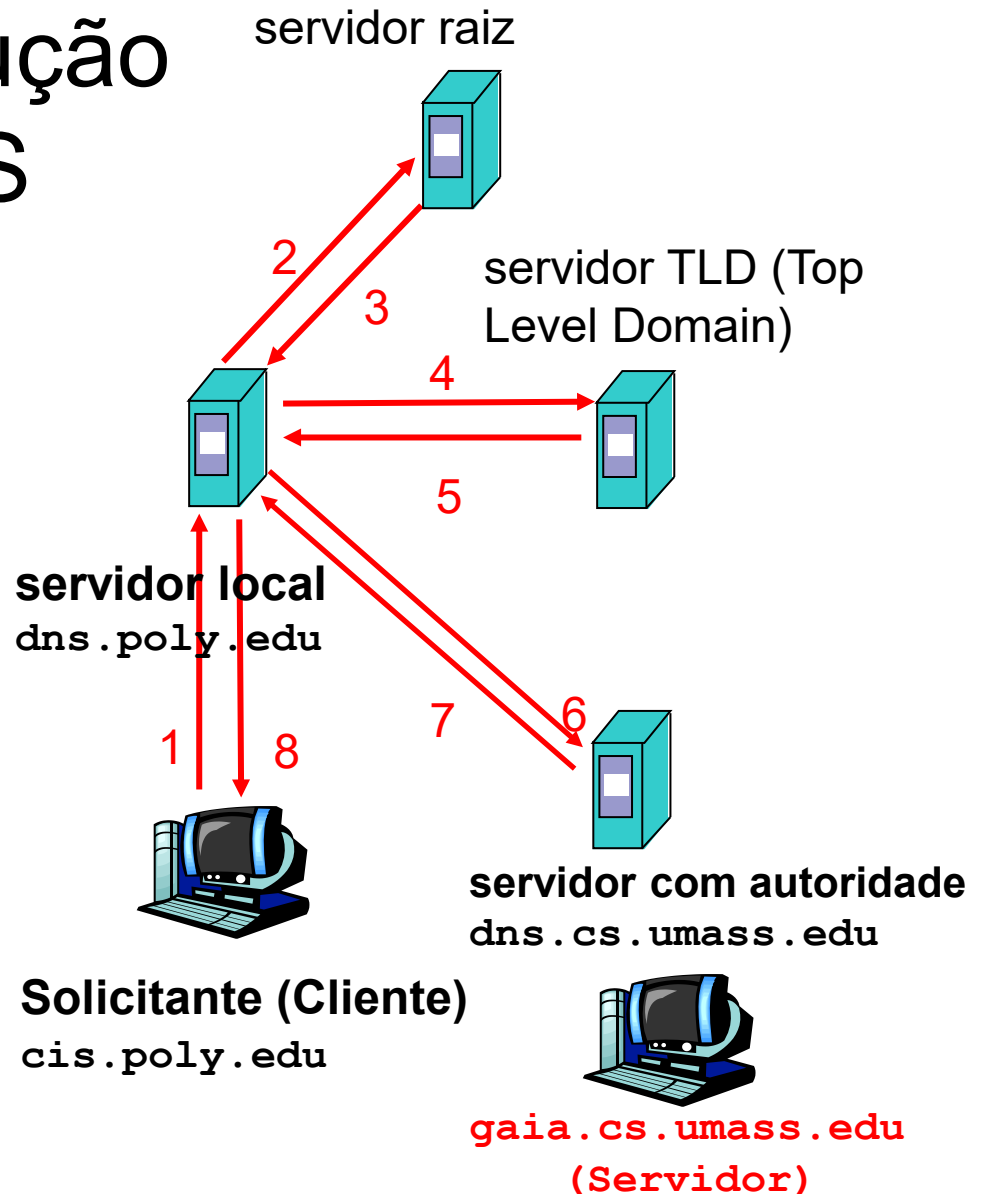
- Não pertence necessariamente à hierarquia
- Cada ISP (**ISP residencial, empresa, universidade**) possui um.
 - Também chamada de “**servidor de nomes default**”
- Quando um hospedeiro faz uma consulta DNS, a requisição é primeiro enviada para o seu servidor **DNS local**
 - Possui uma cache local com pares de tradução nome/endereço recentes (**mas podem estar desatualizados!**)
 - Atua como um intermediário (**proxy**), enviando consultas para a hierarquia.

Exemplo de resolução de nome pelo DNS

- Host cliente em cis.poly.edu quer endereço IP para gaia.cs.umass.edu (server)

Consulta Interativa:

- servidor consultado responde **com o nome de um servidor de contato**
- “Não conheço este nome, mas pergunte para esse servidor”

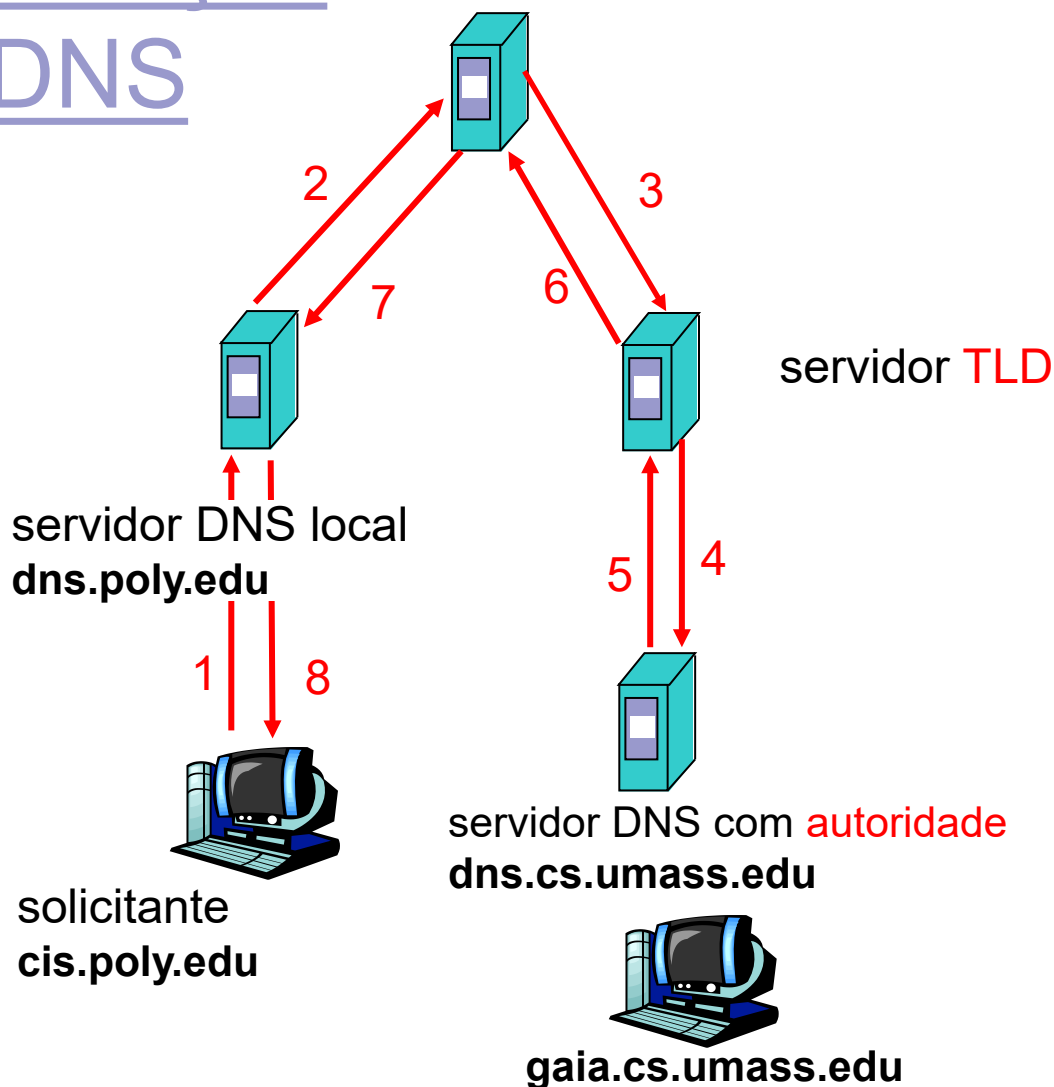


Exemplo de resolução de nome pelo DNS

servidor DNS **Raiz**

Consulta Recursiva:

- transfere a responsabilidade de resolução do nome para o servidor de nomes contatado



DNS: uso de cache, atualização de dados

- uma vez que um servidor qualquer aprende um mapeamento, ele o coloca numa **cache** local
 - entradas na cache são sujeitas a temporização (**desaparecem**) depois de um certo tempo (**TTL**)
- Entradas na cache podem estar **desatualizadas** (tradução nome/endereço do tipo melhor esforço!)
 - Se o endereço IP de um nome de host for alterado, pode não ser conhecido em toda a Internet até que todos os TTLs expirem
- mecanismos de atualização/notificação propostos na **RFC 2136**

Comunicação entre Processos

DESTINO FINAL DOS DADOS:

São Identificados pelo par:

ENDEREÇO IP + PORTA = **ENDEREÇO SOCKET**

Endereço IP	200.23.56.8	69	Número da Porta
Endereço Socket	200.23.56.8	69	

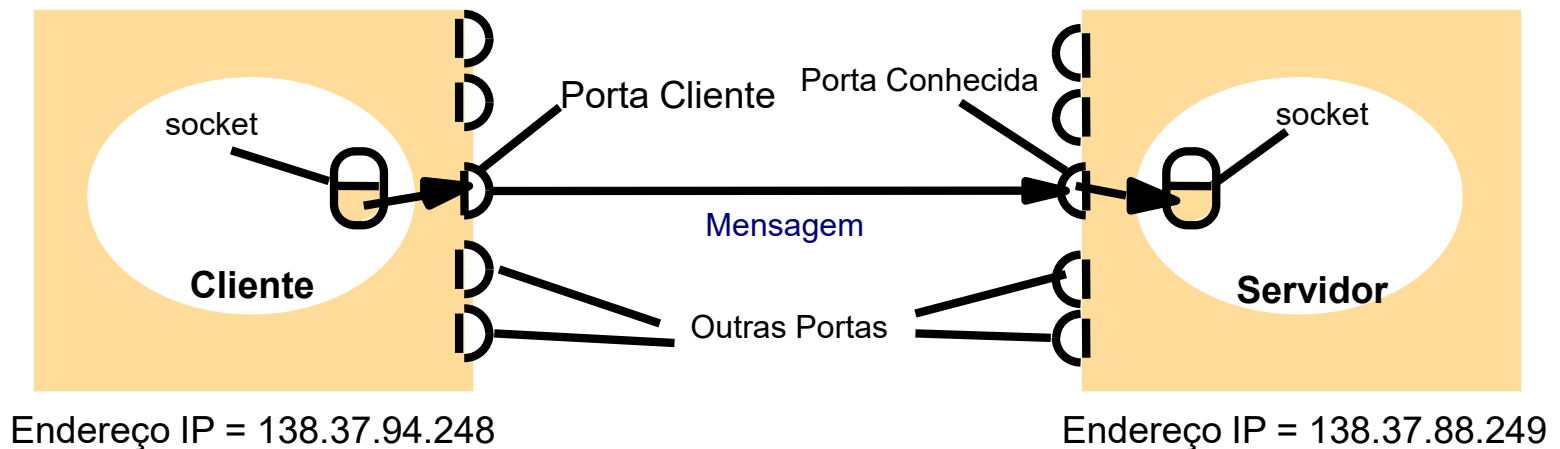
Sockets

- O termo refere-se à API implementada pelo grupo de distribuição de software UNIX da Universidade de Berkeley;
- São definidos como uma extremidade de um canal de comunicação, onde um par de processos ou threads se comunicam em uma rede utilizando um Par de Sockets;
- Um soquete é formado por um endereço IP concatenado com um número de porta.

ENDEREÇO IP + PORTA = **ENDEREÇO SOCKET**

Sockets

- Cada computador tem 2^{16} número de portas disponíveis para envio e/ou recebimento de mensagens;
- Soquete é um mecanismo da arquitetura TCP/IP que permite aplicações distribuídas através da comunicação em rede de processos ou threads.



Sockets

■ Cliente/Servidor:

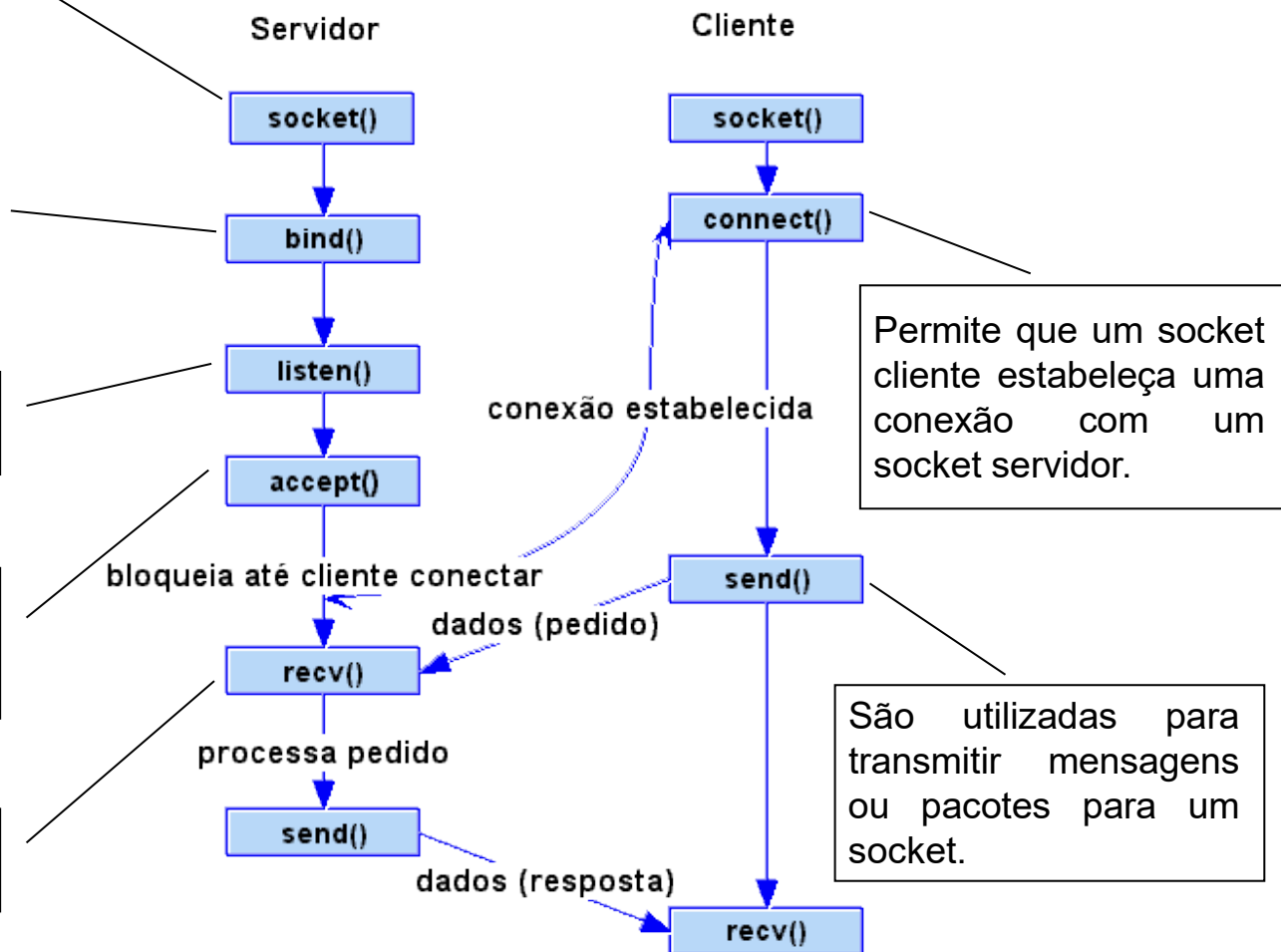
É usada para criar uma interface de comunicação em um domínio específico.

Faz a ligação entre um nome de domínio ou endereço de rede e um socket.

Permite que um socket aceite conexões passivamente.

É usada para criar um socket conectado a partir de uma requisição de conexões.

São utilizadas para a recepção de mensagens ou pacotes.



Permite que um socket cliente estabeleça uma conexão com um socket servidor.

São utilizadas para transmitir mensagens ou pacotes para um socket.

Sockets

- Os servidores que implementam serviços específicos (telnet, ftp, mail e http) utilizam portas bem conhecidas.
- Portas inferiores a 1024 são consideradas portas bem conhecidas, usadas para implementar serviços padrões.
 - **Valores 0 a 1023:** alocados para serviços padronizados pela rede.



Sockets

- As portas no protocolo ****TCP/IP**** são números que permitem que os computadores diferenciem facilmente entre diferentes tipos de tráfego.
- Cada porta está associada a um processo ou serviço específico. Vamos dar uma olhada em algumas das portas mais populares:


Sockets

1. **HTTP** (Hypertext Transfer Protocol): As mensagens HTTP são enviadas para a **porta 80**.
2. **HTTPS** (Hypertext Transfer Protocol Secure): Para comunicações seguras, o HTTPS usa a **porta 443**.
3. **SSH** (Secure Shell): Usada para logins seguros, transferência de arquivos e redirecionamento de porta, o SSH utiliza a **porta 22**.
4. **SMTP** (Simple Mail Transfer Protocol): Para envio de e-mails, o SMTP usa a **porta 25**.

Sockets

5. **SMTPS**: Uma variação segura do SMTP, que utiliza a **porta 465**².
6. **SMTP TLS**: Para criptografia de e-mails, o SMTP TLS usa a **porta 587**².
7. **FTP** (File Transfer Protocol): O FTP tem duas portas associadas:
 - **Porta de comandos**: **21**⁴.
 - **Porta de dados do FTP**: **20**¹.
8. **Telnet**: Comunicação de texto sem criptografia, usando a **porta 23**².
9. **DNS** (Domain Name System): O DNS utiliza a **porta 53** para resolver nomes de domínio em endereços IP¹.

Sockets

Essas são apenas algumas das portas populares no protocolo TCP/IP. Existem muitas outras portas associadas a diferentes serviços e aplicativos. Lembre-se de que as portas fazem parte da ****camada de transporte (camada 4)****, os protocolos de transporte, como ****TCP**** ou ****UDP****, podem indicar para qual porta um pacote deve ser encaminhado². 

Fonte: conversa com o Copilot, 03/06/2024

(1) Protocolos de rede | Portas de rede | Cloudflare. <https://www.cloudflare.com/pt-br/learning/network-layer/what-is-a-computer-port/>.

(2) Portas TCP/IP - Devs Channel. <https://www.devschannel.com/tcp-ip/portas-tcp-ip>.

(3) Lista de portas dos protocolos TCP e UDP.

https://pt.wikipedia.org/wiki/Lista_de_portas_dos_protocolos_TCP_e_UDP.

(4) Portas TCP/IP - CCM.

<https://bing.com/search?q=portas+populares+servi%C3%A7os+protocolo+TCP%2fIP>.

(5) undefined. <https://bing.com/search?q=>.

Sockets

- A comunicação por soquete entre as partes é implementada usando-se passagem de mensagens, utilizando primitivas do tipo *send/receive*.
- A comunicação por soquetes todas as conexões devem ser exclusivas.
- Desta forma, isso garante a integridade da conexão através de um único par de soquetes.



Sockets


■ Tipos de Soquetes:

- **Soquetes Datagrama:** Proporciona um fluxo de mensagens bidirecional não ordenado. **Utiliza o protocolo UDP.**
- **Soquetes Stream:** Proporciona um fluxo sequencial e confiável de dados em ambas as direções, sem limites e sem duplicação de mensagens. **Utiliza o protocolo TCP;**
- **Soquetes Multicast:** Envio para um grupo sem estabelecimento de conexão.

Tipos de Sockets

SOCKETS DATAGRAMA

- ✓ É transmitido de um processo a outro sem o estabelecimento de conexão;
- ✓ Não há a existência de confirmações de recebimento;
- ✓ Não há novas tentativas de envio;
- ✓ Emprega o **protocolo UDP** (*User Datagram Protocol*);
- ✓ Mais rápido que os protocolos baseados à conexão.



O **UDP** (User Datagram Protocol ou Protocolo de Datagrama de Usuário) é um dos protocolos fundamentais na camada de transporte do modelo TCP/IP. Ele é utilizado para enviar mensagens, chamadas de datagramas, na rede.

Diferente do TCP, o UDP é conhecido por ser um protocolo sem conexão, o que significa que ele não estabelece uma sessão antes de enviar dados e não garante a entrega, a ordem ou a integridade dos pacotes enviados¹.

Aqui estão algumas características-chave do UDP:

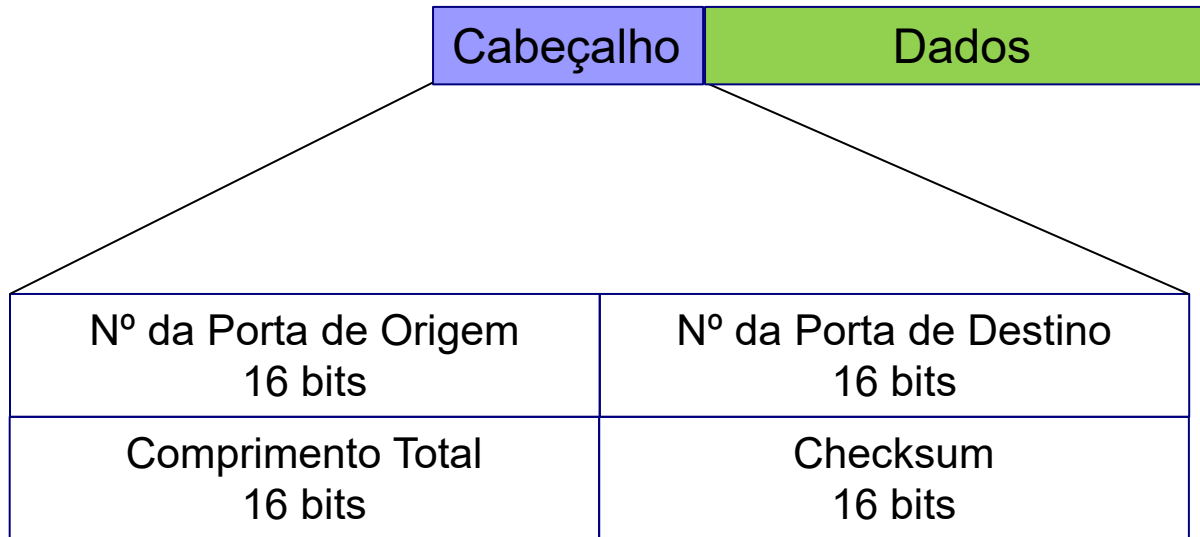
- **Não orientado à conexão:** Não há necessidade de estabelecer uma conexão antes de enviar dados.
- **Sem garantia de entrega:** Não há confirmação de que os pacotes chegaram ao destino.
- **Sem controle de ordem:** Os pacotes podem chegar fora de ordem.
- **Rápido e leve:** Menos sobrecarga do que o TCP, útil para aplicações que requerem velocidade e eficiência, como jogos online ou streaming de vídeo.

Tipos de Sockets

SOCKETS DATAGRAMA

O UDP é frequentemente usado em situações onde a velocidade é mais crítica do que a confiabilidade, como transmissões de vídeo ao vivo ou jogos online, onde a perda ocasional de pacotes é preferível a atrasos causados pela tentativa de retransmissão¹².

8 bytes



Tipos de Soquetes

SOCKETS DATAGRAMA: PROBLEMAS.

❑ Tamanho da Mensagem:

- ✓ O processo destino precisa especificar um vetor de bytes de um tamanho em particular para receber as mensagens;
- ✓ Se a mensagem for grande demais para esse vetor, ela será truncada na chegada;
- ✓ O protocolo IP permite Datagramas de até 64 KB (incluindo cabeçalho e área de dados);
- ✓ A maioria dos ambientes impõem uma restrição de tamanho de 8 Kbytes.

Tipos de Soquetes

SOCKETS DATAGRAMA: PROBLEMAS.

❑ Bloqueio:

- ✓ Os sockets fornecem opções *send* não bloqueante e *receive* bloqueante para comunicação por Datagrama;
- ✓ A operação *send* retornará quando tiver repassado a mensagem para as camadas UDP e IP subjacentes;
- ✓ Se nenhum processo tiver um socket associado à porta de destino, as mensagens serão descartadas.

Tipos de Soquetes

SOCKETS DATAGRAMA:

☐ Modelos de Falhas:

- ✓ **Falhas por Omissão:** Mensagens podem ser descartadas devido a erros de soma de verificação, ou porque não há espaço disponível no buffer;
- ✓ **Ordenamento:** Às vezes, as mensagens podem ser entregues em uma ordem diferente da que foram emitidas.

Tipos de Soquetes

SOCKETS DATAGRAMA:

☐ Emprego do UDP:

- ✓ Os datagramas UDP são escolhas atraentes, pois não sofrem as sobrecargas necessárias a entrega de mensagens garantidas: DNS e VOIP.

☐ Fontes de Sobrecarga:

- ⊕ A necessidade de armazenar informações de estado na origem e no destino;
- ⊕ A transmissão de mensagens extras;
- ⊕ A latência do remetente.



Tipos de Soquetes

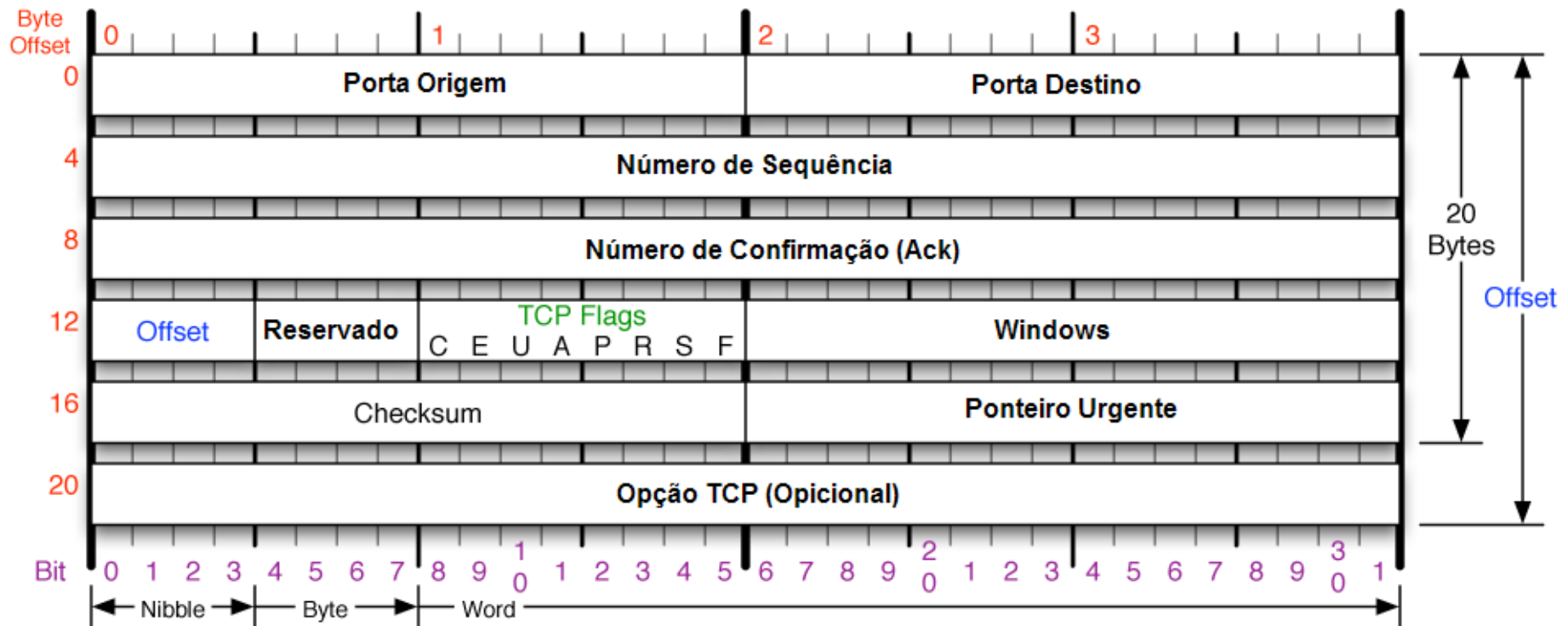
SOCKETS STREAM

- ✓ Se originou do UNIX BSD 4.x.;
- ✓ É transmitido de um processo a outro com o estabelecimento de conexão;
- ✓ Emprega o **protocolo TCP** (*Transmission Control Protocol*);
- ✓ É mais lento que o Datagrama, porém mais confiável;

Tipos de Soquetes

STREAM

TCP





Tipos de Soquetes

STREAM

Características da Comunicação por Stream:

- ✓ **Tamanho das Mensagens:** O aplicativo pode escolher o volume de dados que vai ser transmitido ou recebido;
- ✓ **Mensagens Perdidas:** O protocolo TCP usa um esquema de confirmação;
- ✓ **Controle de Fluxo:** O protocolo TCP tenta combinar a velocidade dos processos que leem e escrevem no fluxo;
- ✓ **Duplicação e Ordenamento de Mensagem;**
- ✓ **Destinos de Mensagens:** *Envolve connect e accept.*

Protocolos Internet

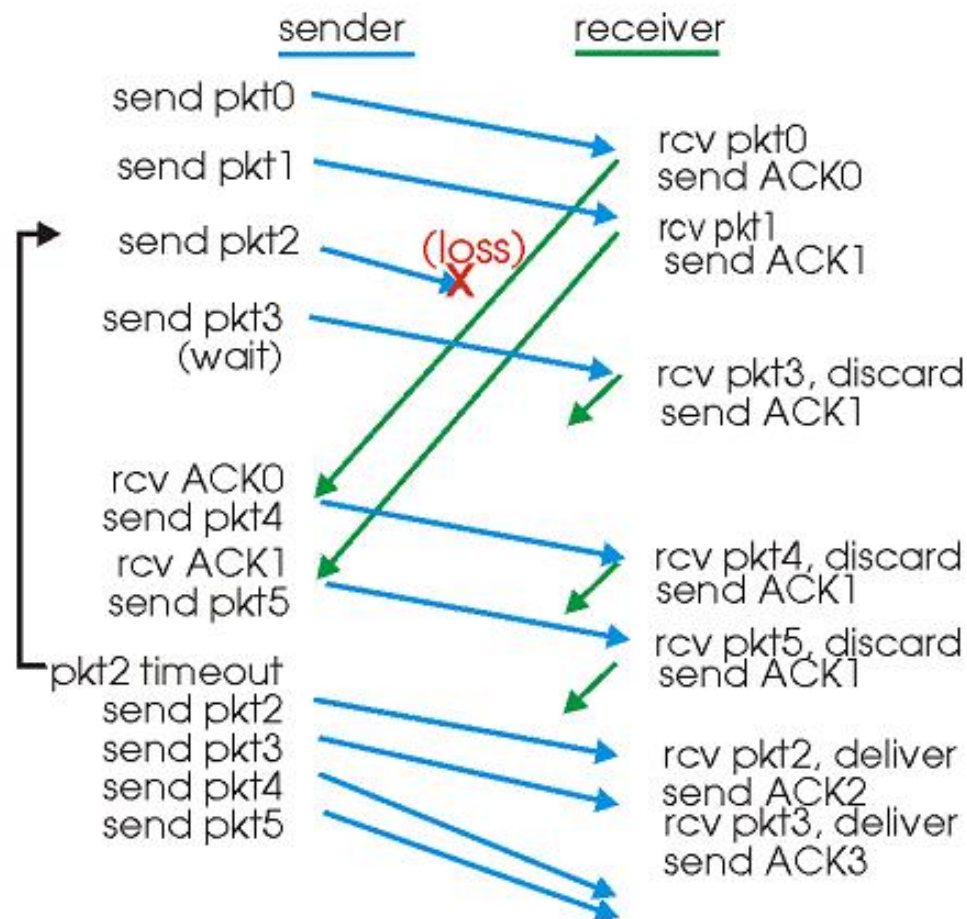
Protocolo TCP (*Transmission Control Protocol*)

CONTROLE DE FLUXO:

O remetente toma o cuidado de não sobrecarregar o destino ou os nós intermediários (obtido por meio de um sistema de confirmação de segmentos).

RETRANSMISSÃO:

Se qualquer segmento não for confirmado dentro de um tempo limite especificado, o remetente o retransmitirá.



Protocolos Internet

Protocolo TCP (*Transmission Control Protocol*)

USO DE BUFFERS:

É utilizado para balancear o fluxo entre o remetente e o destino.

SOMA DE VERIFICAÇÃO:

Se um segmento recebido não corresponde à sua soma de verificação, então ele é eliminado. **“(2 71FA) → 71FA + 2 => 71FC ” => 0111000111111100**

H	e	l	l	o		w	o	r	l	d	.
48	65	6C	6C	6F	20	77	6F	72	6C	64	2E

$$4865 + 6C6C + 6F20 + 776F + 726C + 642E + \text{carry} = 71FC$$

A tabela ASCII (American Standard Code for Information Interchange)

é um sistema de codificação de caracteres utilizado em computadores e dispositivos eletrônicos para representar texto. Ela foi desenvolvida na década de 1960 e é composta por 128 caracteres, incluindo:

- **Caracteres de controle** (não-imprimíveis): Utilizados para controlar dispositivos como impressoras ou para formatar texto, ocupando os códigos de 0 a 31 e o código 127¹.
- **Caracteres imprimíveis:** Incluem letras do alfabeto (maiúsculas e minúsculas), números, sinais de pontuação e alguns símbolos especiais, ocupando os códigos de 32 a 126².

Cada caractere na tabela ASCII é representado por um número decimal, e também pode ser expresso em hexadecimal ou binário. Por exemplo, o caractere “A” tem o código decimal 65, que é 41 em hexadecimal e 1000001 em binário².

A tabela ASCII (American Standard Code for Information Interchange)

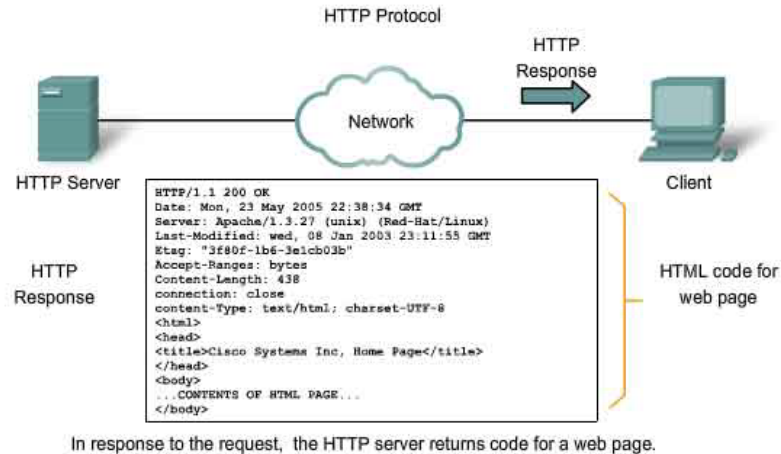
- A tabela ASCII original usa 7 bits para cada caractere, permitindo 128 combinações possíveis.
- Com a evolução dos sistemas de computação, surgiram extensões da tabela ASCII para utilizar 8 bits, permitindo 256 combinações e incluindo mais caracteres, como símbolos gráficos e caracteres de idiomas específicos₁.
- A tabela ASCII é fundamental para a comunicação e processamento de texto em sistemas digitais e continua sendo a base para muitas codificações modernas, como UTF-8₁.

A tabela ASCII (American Standard Code for Information Interchange)

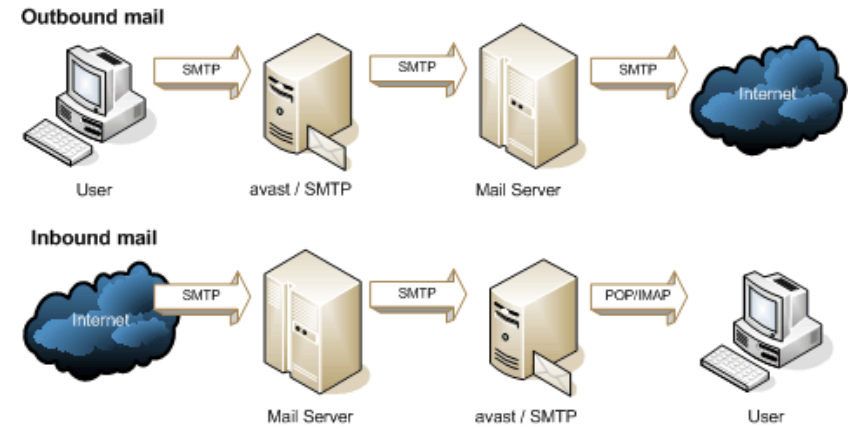
Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SoH	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	SoTxt	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	EoTxt	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EoT	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enq	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Ack	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Bsp	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	HTab	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LFeed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VTab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FFeed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SOut	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SIn	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAck	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Syn	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	EoTB	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Can	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EoM	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Sub	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	Esc	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FSep	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GSep	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RSep	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	USep	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		Delete

Tipos de Soquetes **STREAM** Emprego do TCP:

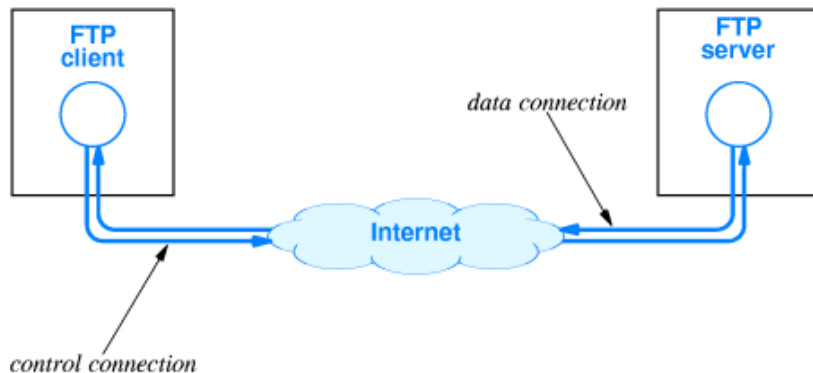
HTTP



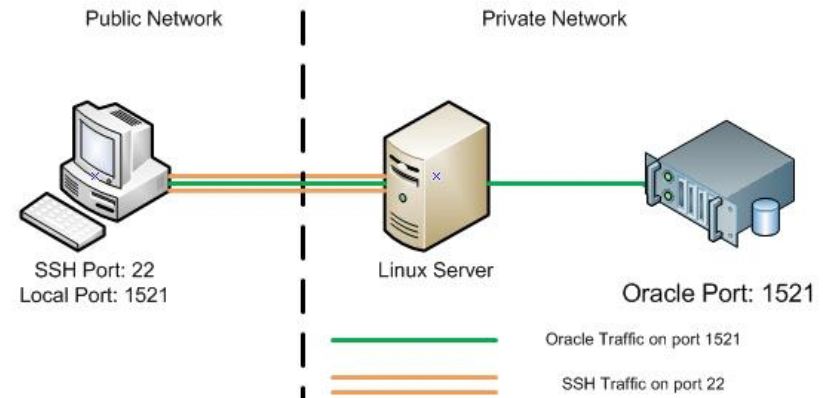
SMTP



FTP



SSH



Tipos de Soquetes

■ Implementação em Java – Soquetes Stream (Servidor)

```
import java.net.*;
import java.io.*;

public class Servidor {
    public static void main(String args[]) {
        try {
            // Cria um servidor
            ServerSocket servidor = new ServerSocket(10001);
            System.out.println("Porta 10001 aberta!");
            System.out.println("Aguardando Conexao...");
            // Aceita uma conexão
            Socket cliente = servidor.accept();
            System.out.println("Nova conexao com o cliente " +
                cliente.getInetAddress().getHostAddress()
            );
            // cria o buffer de leitura
            BufferedReader in = new BufferedReader(
                new InputStreamReader(cliente.getInputStream()));
```

Continuação ----->

```
// lê até o fim

        while(true) {
            String linha = in.readLine();
            if (linha == null) {
                break;
            }
            System.out.println(linha);
        }
        // Fecha tudo
        in.close();
        cliente.close();
        servidor.close();
    } catch (Exception e) {
        //em caso de erro
        System.out.println("Ocorreu um erro na Conexão!");
        e.printStackTrace();
    }
}
```

Tipos de Soquetes

■ Implementação em Python – Soquetes Stream (Servidor)

- `import socket`
- `# Cria um objeto socket`
- `servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`
- `# Associa o socket a um endereço e porta locais`
- `servidor.bind(('localhost', 8080))`
- `# Coloca o socket em modo de escuta`
- `servidor.listen(5)`
- `print("Servidor iniciado e ouvindo na porta 8080...")`
- `# Aceita conexões do cliente`
- `while True:`
- `(cliente, endereco) = servidor.accept()`
- `print(f"Conexão estabelecida com {endereco}")`
- `# Recebe dados do cliente`
- `dados = cliente.recv(1024)`
- `print(f"Dados recebidos: {dados.decode()}")`
- `# Envia uma resposta para o cliente`
- `cliente.sendall("Resposta do servidor".encode())`
- `# Fecha a conexão com o cliente`
- `cliente.close()`

Tipos de Soquetes

- Implementação em Python – Soquetes Stream (Servidor)

Este código inicia um servidor que escuta na porta 8080. Quando um cliente se conecta, o servidor aceita a conexão, recebe dados do cliente, envia uma resposta e fecha a conexão. Lembre-se de que este é um exemplo muito básico e que, em um aplicativo real, você precisaria lidar com exceções, múltiplas conexões de clientes simultâneas (talvez usando threads), e outras questões de rede.

Para mais informações detalhadas e exemplos, você pode consultar a documentação oficial do Python sobre programação de soquetes ou outros guias e tutoriais disponíveis online. Eles oferecem uma visão mais aprofundada sobre soquetes, incluindo soquetes bloqueantes e não bloqueantes, e como trabalhar com eles em diferentes cenários de rede.

Tipos de Soquetes

■ Implementação em Java – Soquetes Stream (Cliente)

```
import java.net.*;
import java.io.*;

public class Cliente {

    public static void main(String args[]) {
        try {
            // Conecta ao servidor
            Socket cliente = new Socket("127.0.0.1",10001);
            System.out.println("O cliente se conectou ao
servidor!");

            // Preparar para a leitura da linha de comando
            BufferedReader in = new BufferedReader(
                new InputStreamReader(System.in)
            );
```

Continuação ----->

```
// Fecha tudo
        cliente.close();

    } catch (Exception e) {
        //em caso de erro
        System.out.println("Ocorreu um erro na
Conexão!");
        e.printStackTrace();
    }
}
```

Tipos de Soquetes

■ Implementação em Python – Soquetes Stream (Cliente)

```
import socket

# Cria um objeto de soquete
# AF_INET refere-se ao endereço da família IP
# SOCK_STREAM indica que será um cliente de soquete
TCP
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Conecta-se ao servidor
# Aqui você precisa substituir 'hostname' pelo endereço do
servidor
# e 'port' pelo número da porta que o servidor está
escutando
s.connect(('hostname', port))

# Agora você pode enviar e receber dados
# Por exemplo, vamos enviar 'Hello, World' para o servidor
s.sendall(b'Hello, World')

# Recebe dados do servidor
data = s.recv(1024)

# Imprime os dados recebidos
print('Received', repr(data))

# Não esqueça de fechar o soquete quando terminar
s.close()
```

Tipos de Soquetes

- Implementação em Python – Soquetes Stream (Cliente)

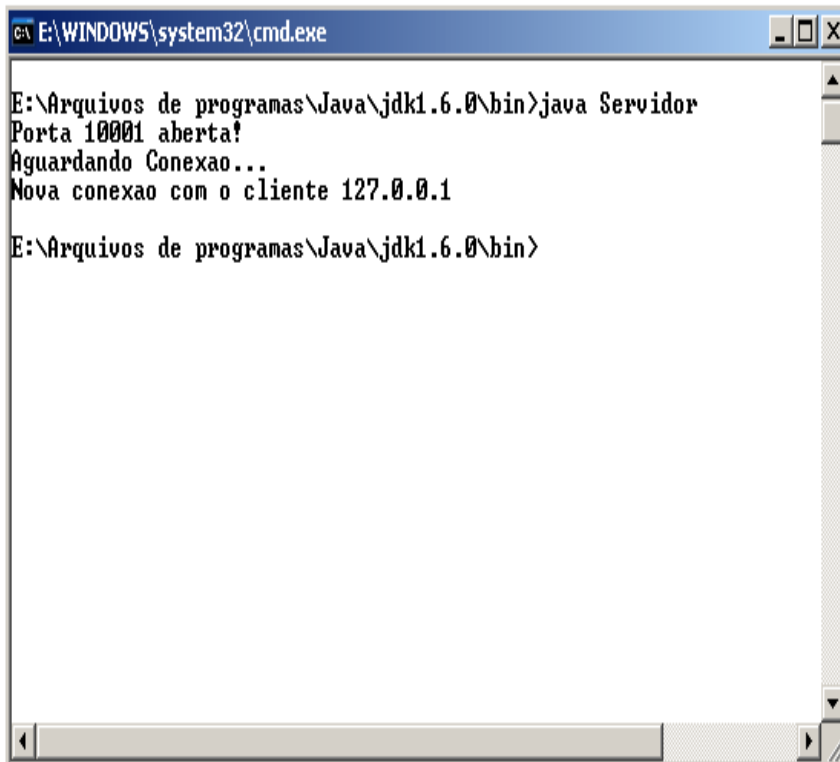
Este é um exemplo muito básico. Na prática, você vai querer adicionar tratamento de exceções e talvez um loop para manter a comunicação com o servidor enquanto necessário.

Para mais informações detalhadas e um guia passo a passo, você pode consultar a [documentação oficial do Python sobre programação de soquetes](#) ou um [guia completo para programação de soquetes em Python](#) disponível online. Esses recursos são ótimos para entender melhor os conceitos de rede e como trabalhar com soquetes em Python.

Tipos de Soquetes (JAVA)

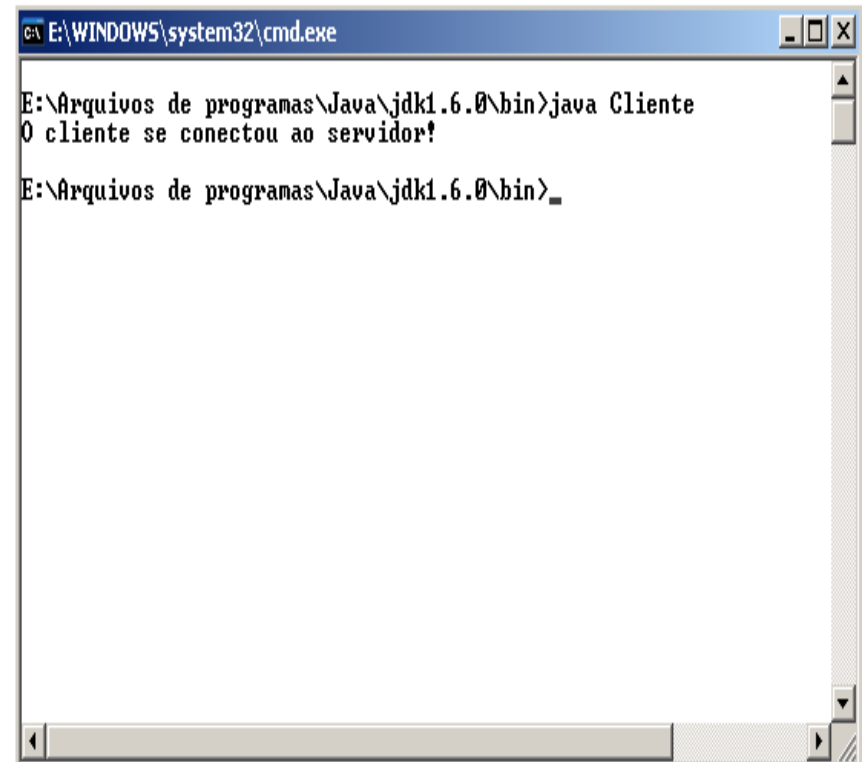
- Socket Stream - Aplicação em Shell (Interpretador de comandos):

SERVIDOR



```
E:\WINDOWS\system32\cmd.exe
E:\Arquivos de programas\Java\jdk1.6.0\bin>java Servidor
Porta 10001 aberta!
Aguardando Conexao...
Nova conexao com o cliente 127.0.0.1
E:\Arquivos de programas\Java\jdk1.6.0\bin>
```

CLIENTE



```
E:\WINDOWS\system32\cmd.exe
E:\Arquivos de programas\Java\jdk1.6.0\bin>java Cliente
O cliente se conectou ao servidor!
E:\Arquivos de programas\Java\jdk1.6.0\bin>
```