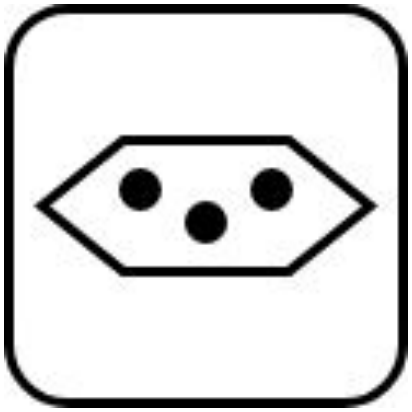
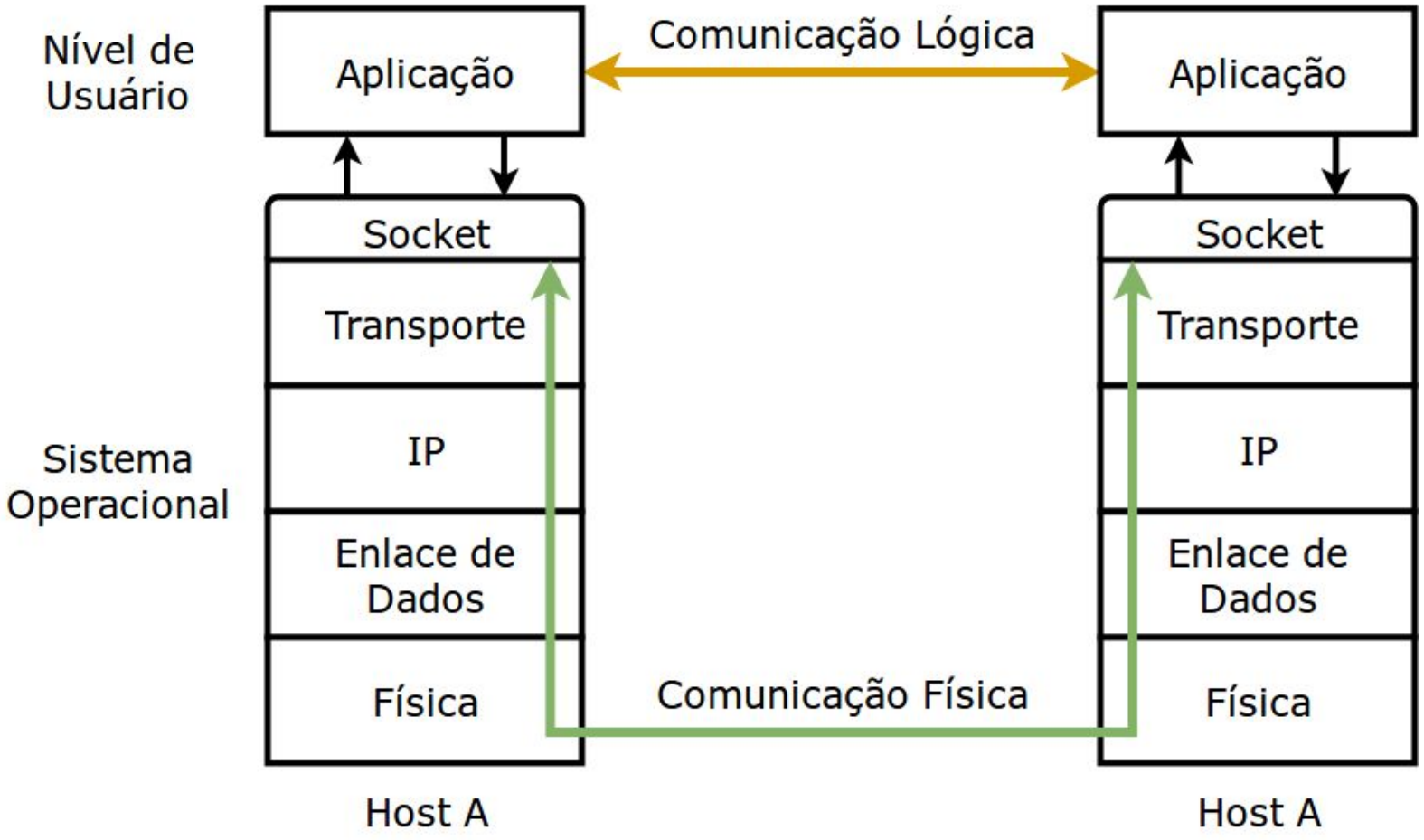
O que são Sockets?

- Permitem a comunicação entre dois processos
  - Local ou remota
- Utilizados em aplicações que demandam troca de dados
  - Ex. Servidor e Cliente
- Interface padrão do Unix
  - Também implementada no Windows (*Winsock*)



Comunicação por Sockets



Principais Tipos de Sockets

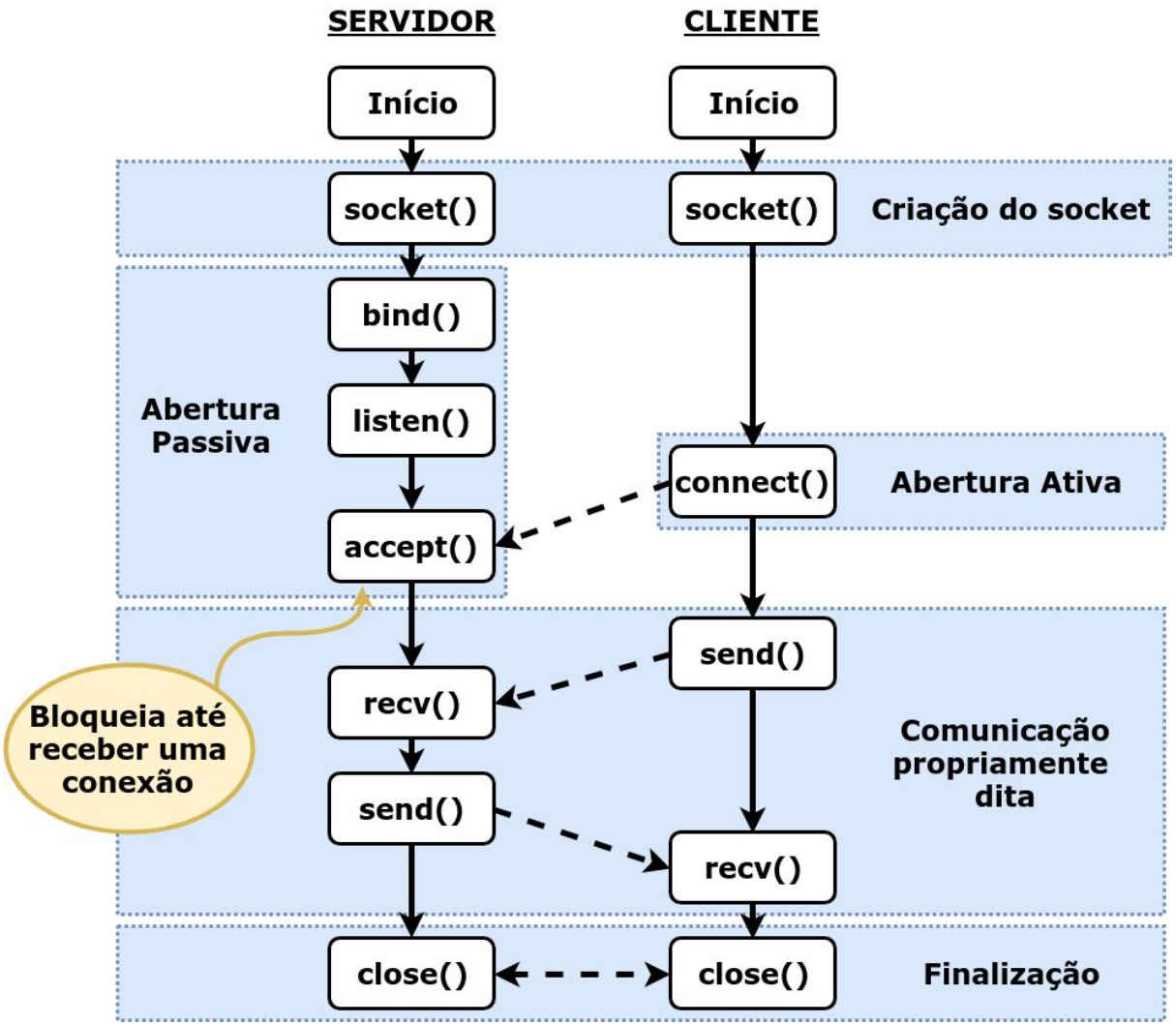
SOCK_STREAM (TCP)	SOCK_DGRAM (UDP)

• Orientado a conexão
• Confiável
  ◦ Sem erros
  ◦ Entrega em ordem
  ◦ Sem mensagens duplicadas
• Fluxo de bytes
	• Orientado a datagrama
	• Sem conexão
	• Não confiável
	• Mensagens com tamanho limitado

Fluxo de Comunicação (com Conexão)

1. Abertura Passiva: Processo servidor se identifica e fica aberto a receber conexões
2. Abertura Ativa: Processo cliente sabe alcançar o processo servidor
3. Abertura Completa: Processo servidor aceita conexão com processo cliente
4. Comunicação entre os processos
5. Finalização da conexão

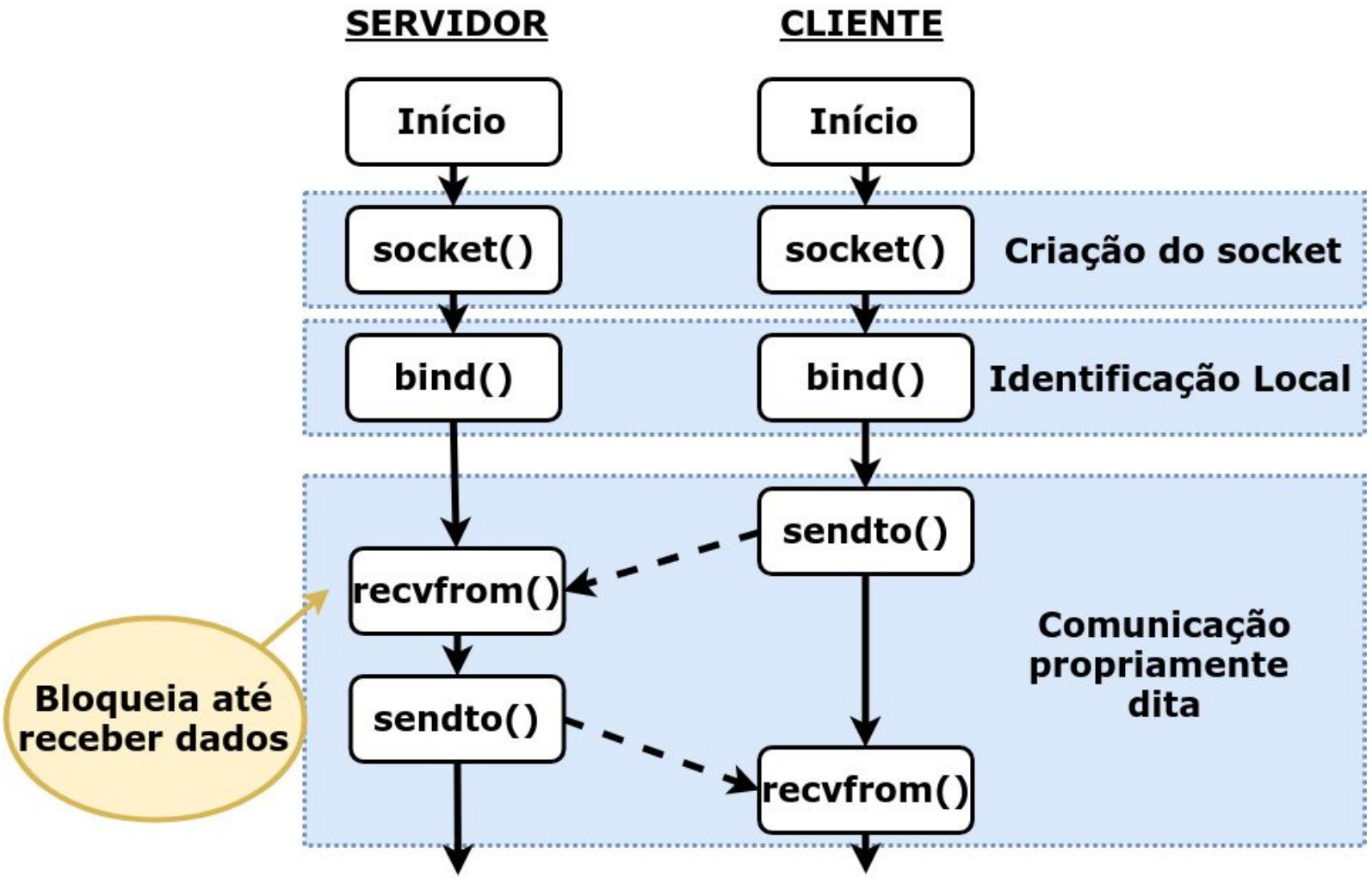
Fluxo de Comunicação (com Conexão)



Fluxo de Comunicação (sem Conexão)

1. Processos servidor e cliente se identificam
2. Comunicação entre os processos

Fluxo de Comunicação (sem Conexão)



Interface – Criação

socket() – Criação de um ponto final de comunicação

Específica família de protocolos
AF_UNIX, AF_INET, AF_INET6, ...

Seleciona protocolo
0 (*default)*

```
int socket(int domain, int type, int protocol)
```

Define tipo de comunicação
SOCK_STREAM, SOCK_DGRAM, ...

Interface – Identificação

bind() – Associa um endereço IP + Porta ao socket

Referência ao socket previamente criado

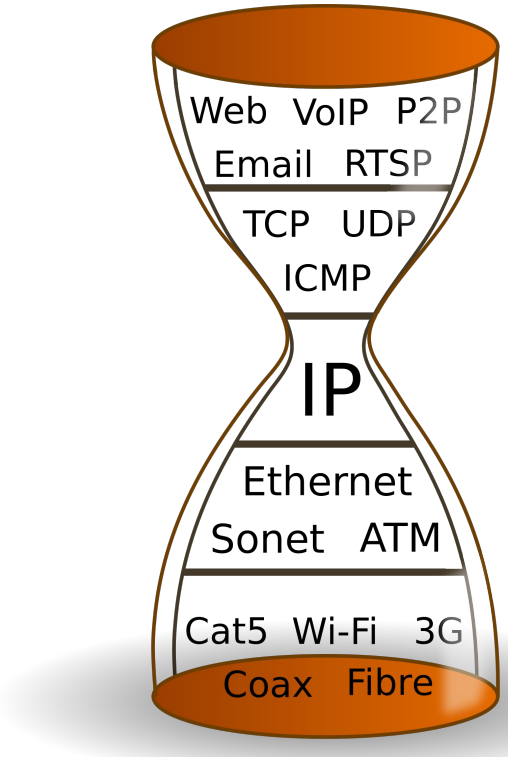
Especificação do endereço a ser associado

```
int bind(int sockfd, const struct sockaddr *addr,
socklen_t addrlen)
```

Tamanho da estrutura apontada por *addr* em bytes

Endereços para Sockets

- Endereço IP + Porta
- Endereços IP
  - Identificam máquinas na Internet
  - 32 bits (IPv4) ou 128 bits (IPv6)
  - Ex.: 127.0.0.1 ou ::1
  - Consultar IP do computador:
    - Linux: ifconfig
    - Windows: ipconfig



Interface – Struct de Endereçamento IPv4

```
struct sockaddr_in {
  sa_family_t    sin_family; /* AF_INET */
  in_port_t      sin_port;   /* Numero da Porta */
  struct in_addr sin_addr;   /* Endereço IPv4 */
};

struct in_addr {
   uint32_t  s_addr;         /* Endereço IPv4 */
};
```

Interface – Struct de Endereçamento IPv6

```
struct sockaddr_in6 {
   sa_family_t     sin6_family;   /* AF_INET6 */
   in_port_t       sin6_port;     /* Numero da Porta */
   uint32_t        sin6_flowinfo; /* Inf. de Fluxo IPv6 */
   struct in6_addr sin6_addr;     /* Endereço IPv6 */
   uint32_t        sin6_scope_id; /* ID do Escopo */
 };

struct in6_addr {
   unsigned char s6_addr[16];     /* Endereço IPv6 */
};
```

Endereços para Sockets

- Portas
  - Identificam aplicações por protocolo (TCP e UDP)
  - Total de 65.536 portas
    - 0 a 1.023: serviços essenciais
    - 1.024 a 49.151: registro a pedido
    - 49.152 a 65.535: privadas (alocação temporária)
  - Ex.: 80 (HTTP) e 22 (SSH)
  - Consultar portas utilizadas:
    - Linux: etc/services
    - Windows: windows\system32\drivers\etc\services

Interface – Abertura Passiva

- listen() – Define um socket como passivo

```
int listen(int sockfd, int backlog)
```

Tamanho máxima da fila de conexões pendentes

- accept() – Aceita uma conexão em um socket passivo

```
int accept(int sockfd, struct sockaddr *addr,
socklen_t *addrlen)
```

Interface – Abertura Ativa

connect() – Inicia uma conexão a um socket

Especificação do endereço ao qual o socket deve ser conectado

```
int connect(int sockfd, const struct sockaddr *addr,
socklen_t addrlen)
```

Interface – Comunicação

send() e sendto() – Enviam mensagens para outro socket

Buffer com a mensagem

```
ssize_t send(int sockfd, const void *buf, size_t len,
int flags)
```

Tamanho da mensagem em bytes

Flags para a função

```
ssize_t sendto(int sockfd, const void *buf, size_t
len, int flags, const struct sockaddr *dest_addr,
socklen_t addrlen)
```

Interface – Comunicação

recv() e recvfrom() – Recebem mensagens de outro socket

Buffer onde a mensagem é armazenada

Tamanho da mensagem recebida em bytes

```
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```

```
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen);
```

Endereço da fonte da mensagem

Interface – Finalização

close() – Fecha o descritor de arquivo

```
int close(int sockfd)
```

Operações especiais

- Conversão de uma string para um endereço IP

```
in_addr_t inet_addr(const char *cp)
```

- Conversão de um endereço IP para uma string

```
char *inet_ntoa(struct in_addr in);
```

Operações especiais

Conversão de inteiros da representação de máquina para a representação da rede e vice-versa

`uint16_t htons(uint16_t hostshort)`	**h**ost **to** **n**etwork **s**hort
`uint32_t htonl(uint32_t hostlong)`	**h**ost **to** **n**etwork **l**ong
`uint16_t ntohs(uint16_t netshort)`	**n**etwork **to** **h**ost **s**hort
`uint32_t ntohl(uint32_t netlong)`	**n**etwork **to** **h**ost **l**ong

Operações Especiais

Definir opções de sockets

```
int setsockopt(int socket, int level, int option_name,
const void *option_value, socklen_t option_len)
```

- SO_RCVTIMEO / SO_SNDTIMEO – *timeout* para recebimento / envio
- SO_RCVBUF / SO_SNDBUF – tamanho do buffer de recebimento / envio