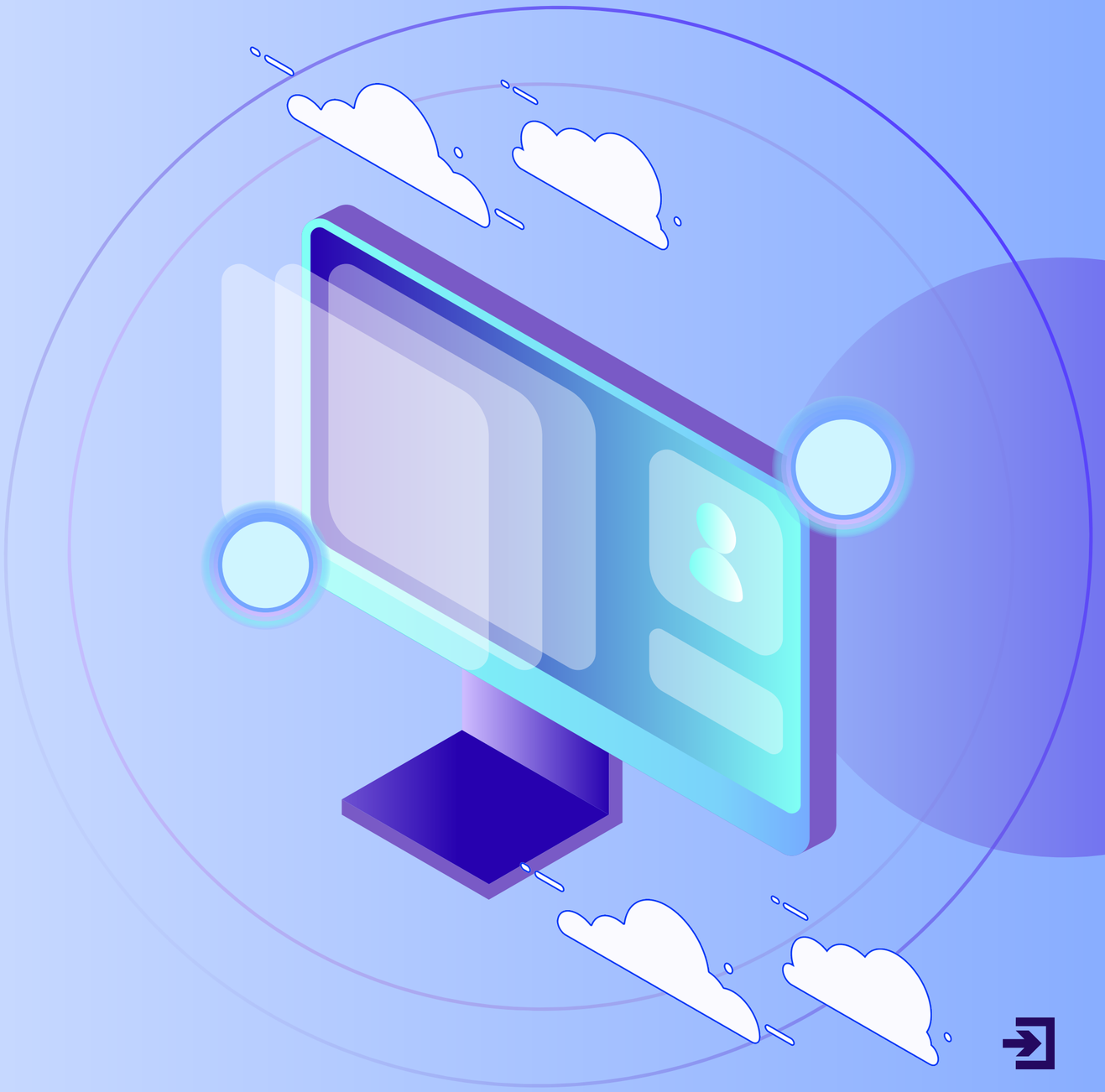
COMUNICAÇÃO ENTRE PROCESSOS

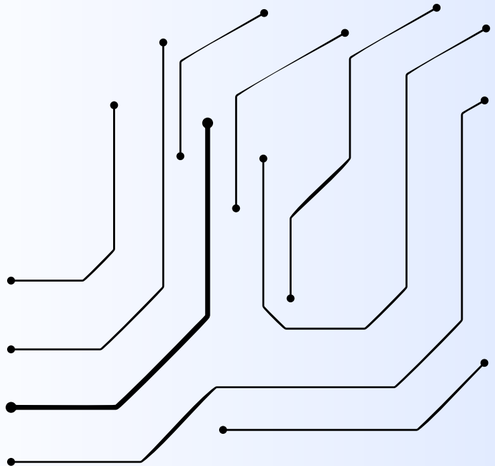
TROCA DE MENSAGENS

Sistemas Operacionais

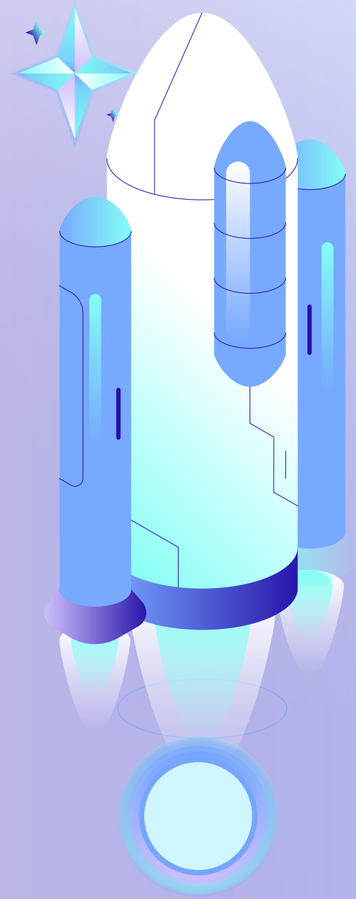


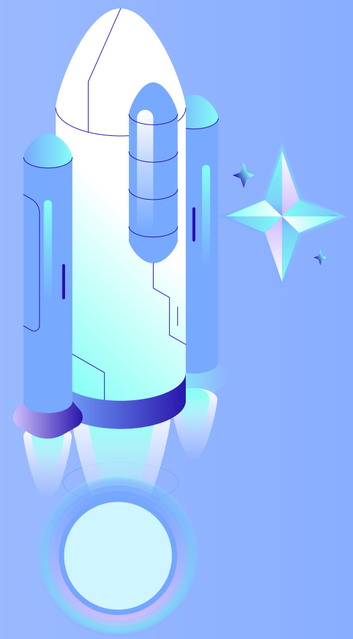
DO QUE SE TRATA ESSA PRÁTICA?

A comunicação entre processos (IPC) por troca de mensagens é um método onde um processo envia dados para outro através de um canal de comunicação, usando primitivas como send (enviar) e receive (receber).



VANTAGENS





1. Baixo acoplamento entre processos;

2. Mais segurança e isolamento;

3. Fácil de usar em sistemas distribuídos;

4. Sincronização embutida e

5. Escalabilidade.

DESVANTAGENS

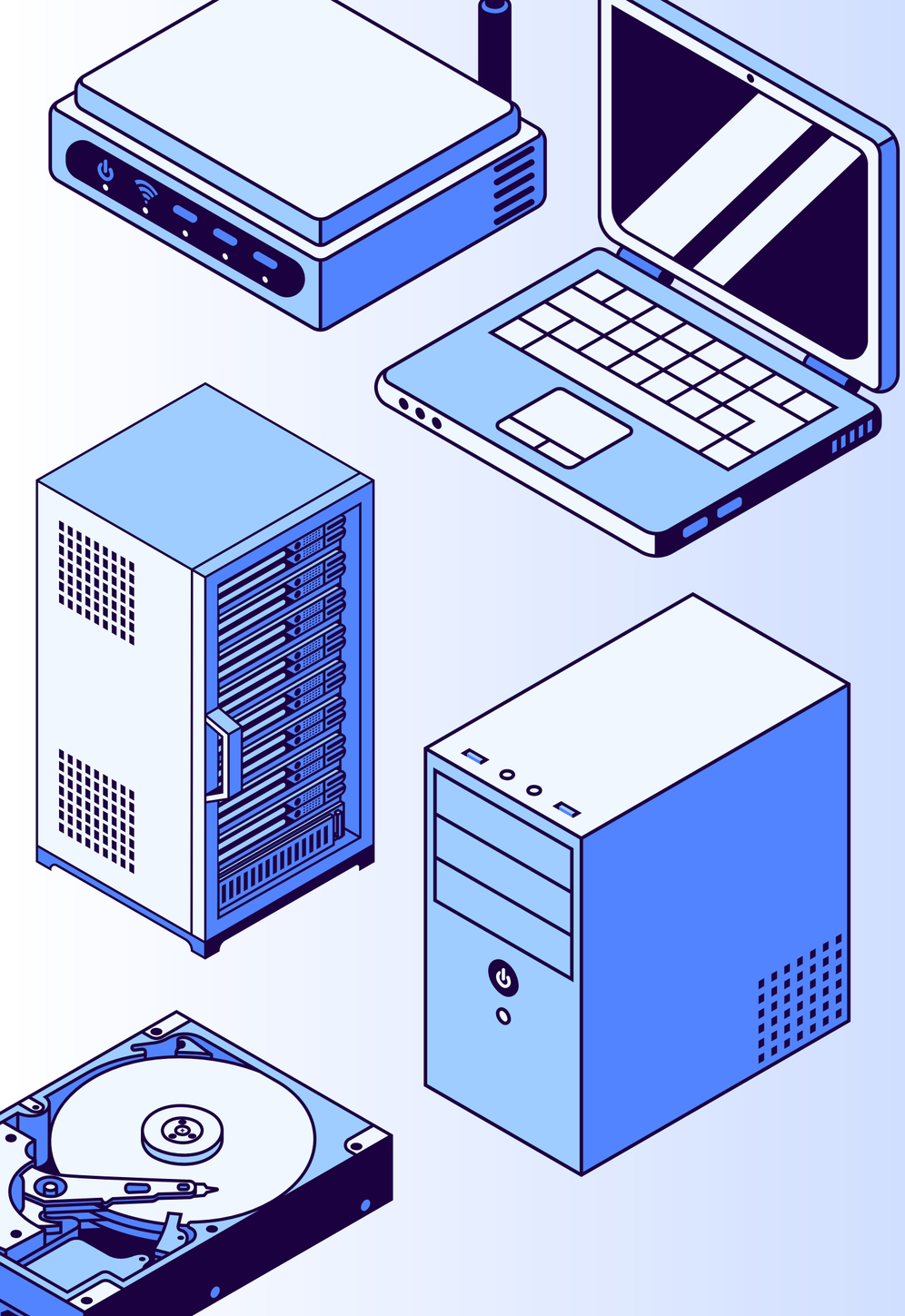
1. Podem gerar overhead (custo extra) ;

2. Complexidade maior em comunicação intensa;

3. Necessidade de protocolos;

4. Possível espera/bloqueio e

5. Depende fortemente do SO ou da rede.



Código

Pseudo código / C

```c
#define N 100

void producer(void) {
  int item;
  message m;
  while (TRUE) {
    item = produce_item();
    receive(consumer, &m);
    build_message(&m, item);
    send(consumer, &m);
  }
}
```

Pseudo código / C

```c
void consumer(void) {
  int item, i;
  message m;
  for (i = 0; i < N; i++);
  while (TRUE) {
    receive(producer, &m);
    item = extract_item(&m);
    send(producer, &m);
    consume_item(item);
  }
}
```

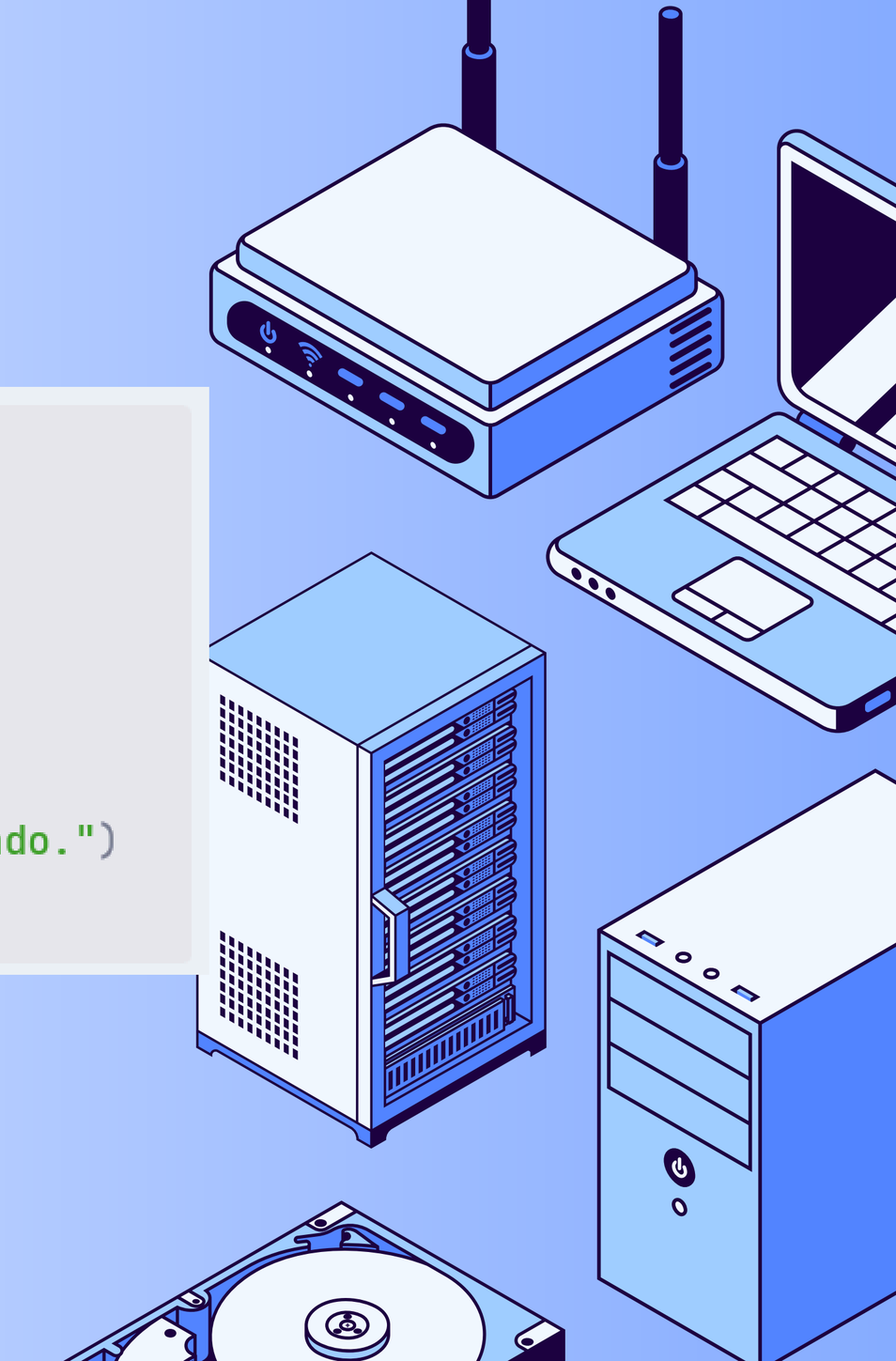
Go

```go
const BUFFER_SIZE = 5
const NUM_ITEMS = 10
```

```go
func producer(ch chan<- int, wg *sync.WaitGroup) {
  defer wg.Done()
  for item := 0; item < NUM_ITEMS; item++ {
    fmt.Println("PRODUTOR: Gerando item", item)
    ch <- item
    fmt.Println("PRODUTOR: Enviou item", item)
    time.Sleep(100 * time.Millisecond)
  }
  close(ch)
  fmt.Println("PRODUTOR: Canal fechado.")
}
```

Go

```go
func consumer(ch <-chan int, wg *sync.WaitGroup) {
  defer wg.Done()
  for item := range ch {
    fmt.Println("CONSUMIDOR: Recebeu item", item)
    time.Sleep(300 * time.Millisecond)
  }
  fmt.Println("CONSUMIDOR: Canal fechado e vazio. Encerrando.")
}
```



Go

```go
func main() {
  ch := make(chan int, BUFFER_SIZE)
  var wg sync.WaitGroup
  wg.Add(2)
  go producer(ch, &wg)
  go consumer(ch, &wg)
  fmt.Println("MAIN: Aguardando o término das goroutines...")
  wg.Wait()
  fmt.Println("MAIN: Todas as goroutines terminaram.")
}
```

```python
import threading
import queue

canal = queue.Queue()   # fila atua como canal de mensagens

def produtor():
    for i in range(5):
        canal.put(f"msg {i}")     # envia mensagem
        print(f"Produtor enviou msg {i}")
    canal.put(None)               # mensagem de encerramento

def consumidor():
    while True:
        msg = canal.get()         # recebe mensagem
        if msg is None:
            break                 # termina
        print(f"Consumidor recebeu {msg}")

threading.Thread(target=produtor).start()
threading.Thread(target=consumidor).start()
```

OBRIGADA!