Jogo Multiplayer Adivinhe o Número

Uma implementação prática de comunicação TCP em tempo real com Python

Arquitetura: Três Pilares do Sistema

protocolo.py

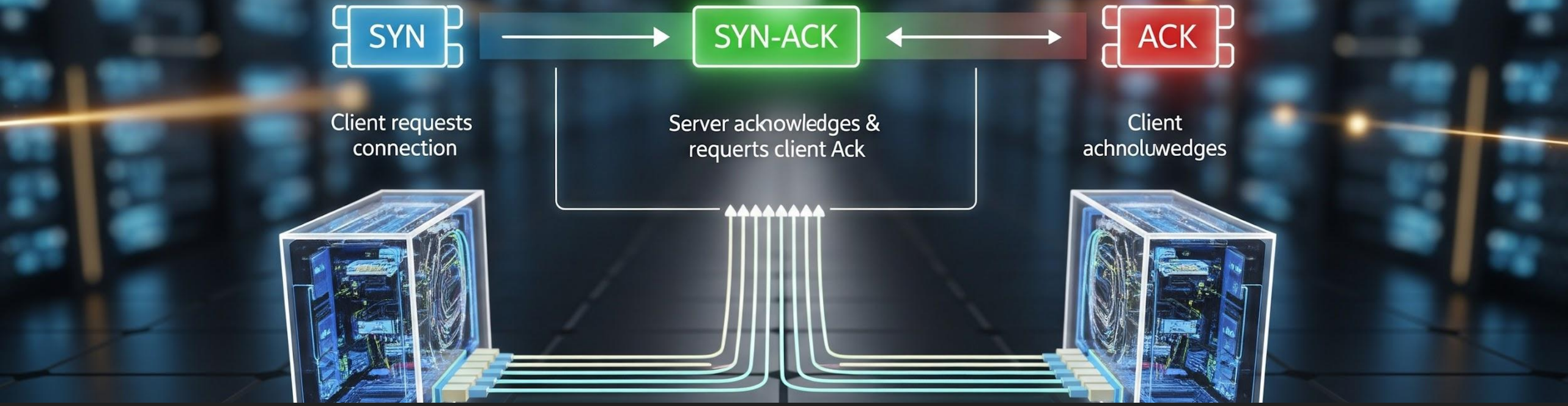
Define a padronização das mensagens trocadas entre servidor e clientes, garantindo interpretação correta dos dados usando formato "COMANDO|DADOS".

servidor.py

Gerencia o estado do jogo, aceita conexões TCP, cria threads para cada cliente e utiliza broadcast para notificação em tempo real.

cliente.py

Conecta-se ao servidor via TCP, envia tentativas codificadas e recebe mensagens através de thread dedicada de escuta.



Socket TCP: A Base da Comunicação

No Servidor

`socket.socket(AF_INET, SOCK_STREAM)` cria socket TCP

`bind()` vincula a porta 8888

`listen()` aguarda conexões

`accept()` recebe novo cliente

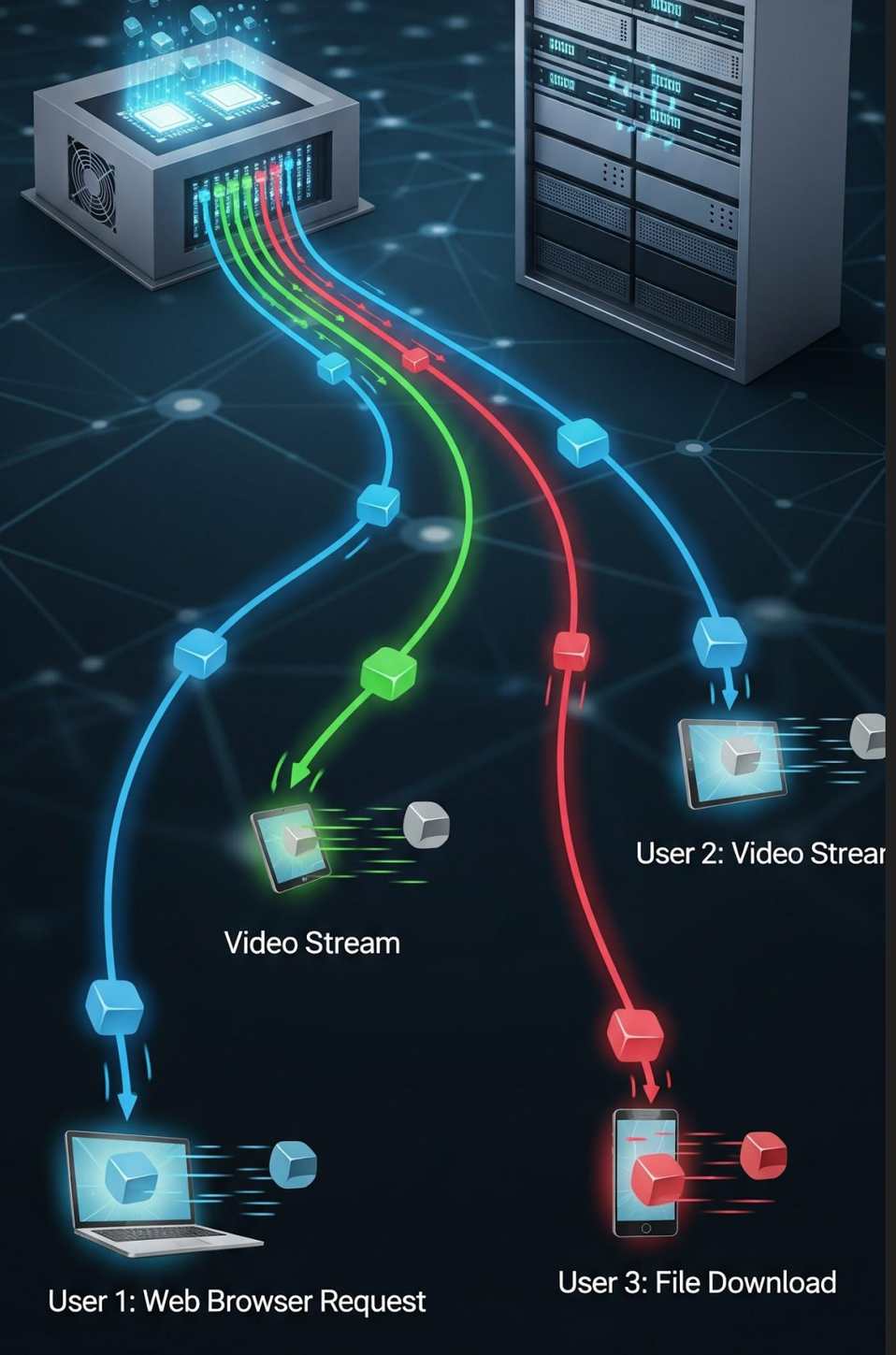
No Cliente

`socket.socket(AF_INET, SOCK_STREAM)` cria socket TCP

`connect()` estabelece conexão com servidor

`send()` transmite tentativas codificadas

`recv()` recebe feedback



Threading: Execução Concorrente

O servidor utiliza **threads** para permitir que múltiplos clientes joguem simultaneamente. Uma thread é um fluxo de execução independente que permite paralelismo dentro do mesmo processo.

0
1

Servidor aguarda conexões

A thread principal fica em loop chamando `accept()`

0
2

Nova conexão chega

Um cliente se conecta e obtém uma tupla (conexão, endereço)

0
3

Thread dedicada criada

`threading.Thread()` inicia função `clientes()` para este cliente

0
4

Paralelismo

Servidor continua aceitando novos clientes enquanto gerencia os anteriores

Protocolo Personalizado: Padronização de Mensagens

A classe `Protocolo` implementa dois métodos estáticos cruciais para garantir que servidor e clientes entendam as mensagens:

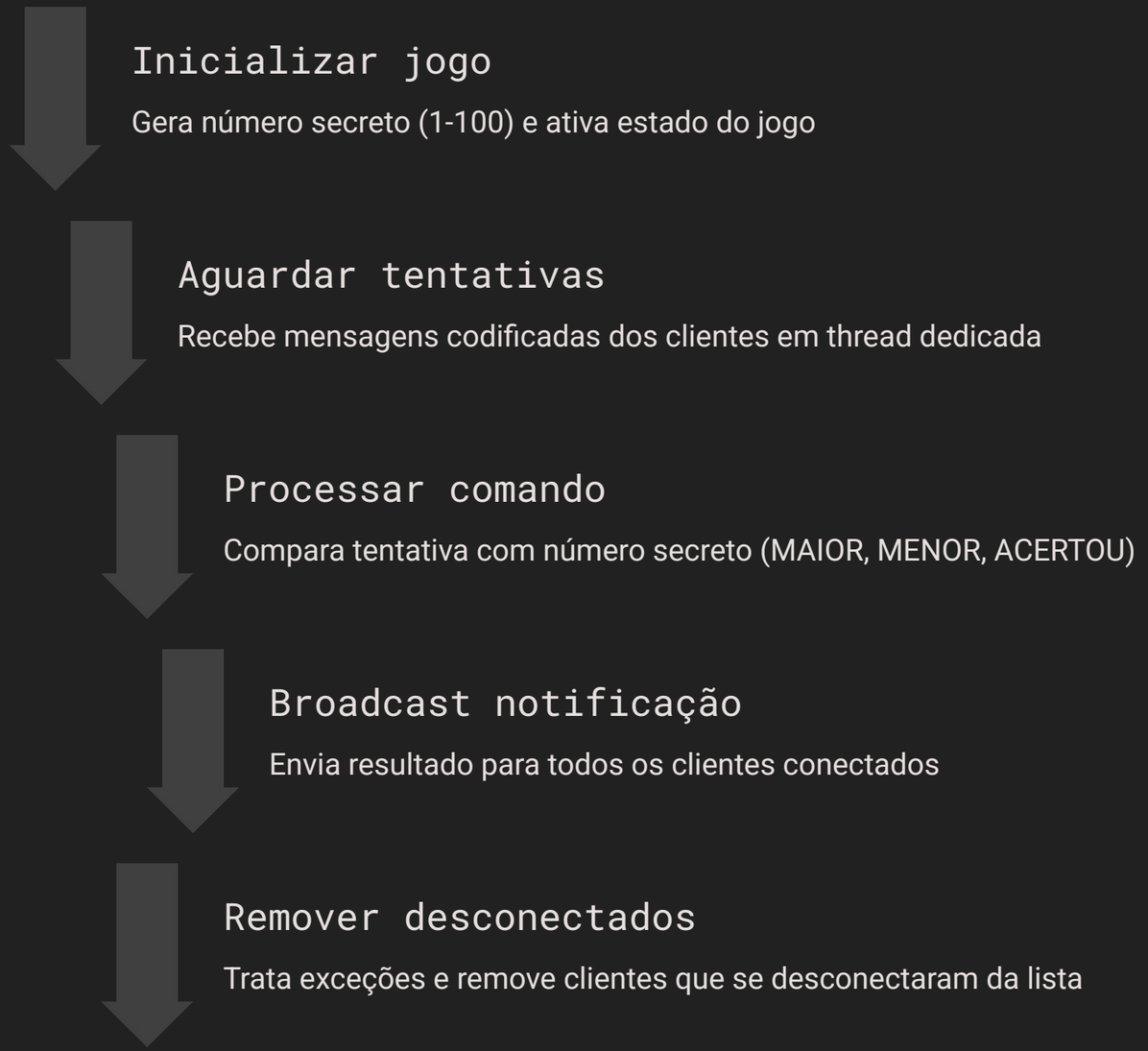
`codificar(comando, dados)`

Formata mensagens como "TENTATIVA|50" ou "MAIOR|O número é menor", usando "|" como separador.

`decodificar(mensagem)`

Divide a string usando `.split("|", 1)` retornando tupla (comando, dados) para interpretação.

Fluxo do Servidor: Gerenciamento Centralizado



Inicializar jogo

Gera número secreto (1-100) e ativa estado do jogo

Aguardar tentativas

Recebe mensagens codificadas dos clientes em thread dedicada

Processar comando

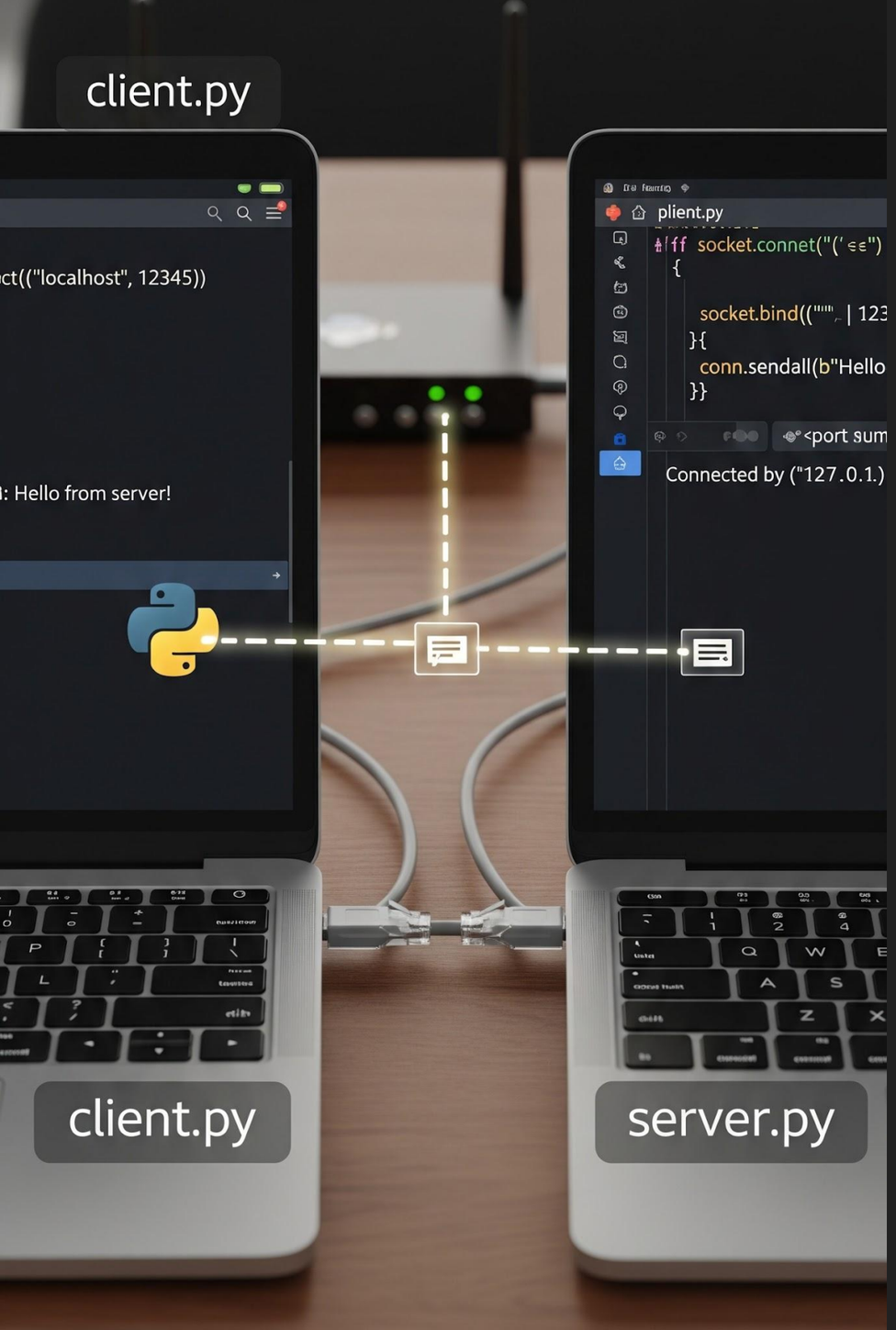
Compara tentativa com número secreto (MAIOR, MENOR, ACERTOU)

Broadcast notificação

Envia resultado para todos os clientes conectados

Remover desconectados

Trata exceções e remove clientes que se desconectaram da lista



Fluxo do Cliente: Interação em Tempo Real

O cliente executa duas operações simultaneamente através de threading:

Thread Principal

Lê entradas do jogador via `input()` e envia tentativas codificadas

Thread Daemon

Escuta continuamente servidor com `recv()`, decodifica mensagens e exibe feedback visual com emojis

A thread daemon (executada como `thread.daemon = True`) permite que o cliente encerre naturalmente quando a thread principal fecha, evitando travamentos.

Conceitos de Redes Demonstrados

Cliente-Servidor

Arquitetura centralizada com servidor gerenciando múltiplos clientes

TCP/IP

Protocolo confiável, orientado a fluxo, garantindo entrega ordenada

Tratamento Erros

Gerenciamento robusto de desconexões e exceções de rede

Broadcast

Servidor envia mensagens para múltiplos clientes simultaneamente

Protocolo Aplicação

Padronização de mensagens garante interpretação consistente

Concorrência

Threading permite múltiplas operações paralelas no mesmo processo