SERVIDOR TCP

Transmission Control Protocol

Por Sediane Carmem Lunardi Hernandes

AGENDA

- Servidor TCP
- Cliente TCP
- Quando utilizar o TCP

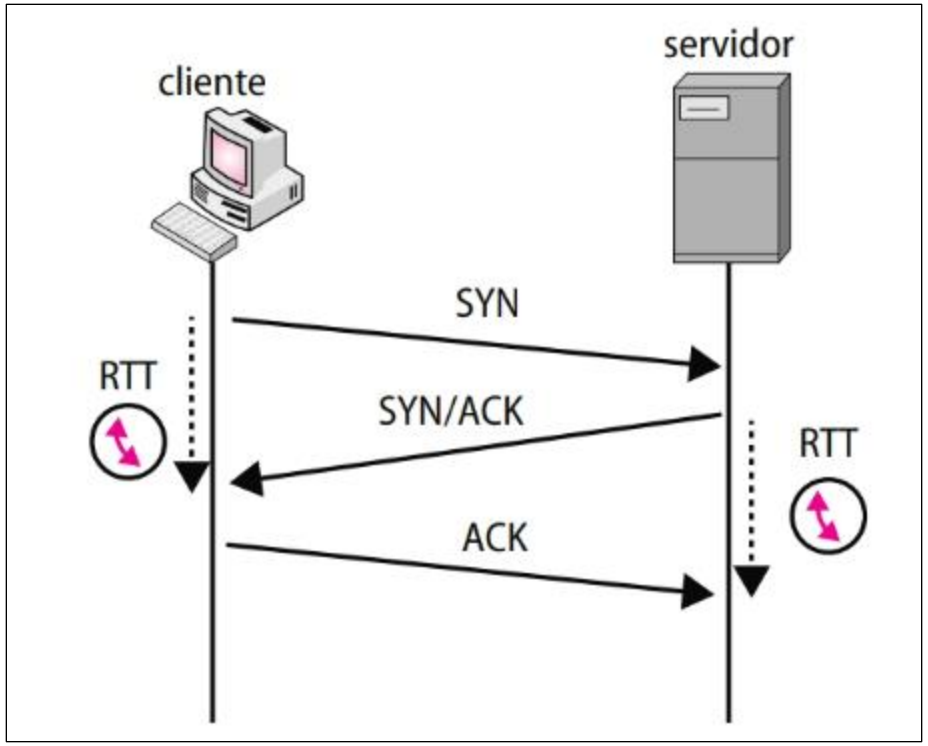
Universidade Tecnológica Federal do Paraná
UTFPR – Câmpus Guarapuava
Autora: **Sediane Carmem Lunardi Hernandes**

SERVIDOR TCP

- Aplicações que utilizam o protocolo de transporte TCP possuem **GARANTIAS** quanto à transmissão de dados:
  1. **Estabelecimento de conexão** entre cliente e servidor
  2. **Controle de fluxo** entre cliente e servidor
  3. **Controle de sequência**
  4. Controle de erros com retransmissão
- O processo de **(1) estabelecimento de CONEXÃO** entre o cliente e o servidor serve para que o cliente certifique-se de que o servidor está ativo e aceita a conexão com o cliente
  - Durante o processo de conexão são trocadas informações importantes
    - o tamanho da área de memória (buffer) de recebimento de mensagens de cada lado (**controle de fluxo**)
    - o tamanho máximo dos pacotes (para que o TCP possa fragmentar as informações da aplicação corretamente)
    - o tempo necessário para que uma mensagem seja enviada e uma resposta recebida (importante para o estabelecimento do tempo de espera máximo de identificação de erros de conexão)

SERVIDOR TCP (CONT.)

Processo de estabelecimento de conexão



Round Trip Time (RTT) é o tempo médio para envio de mensagens

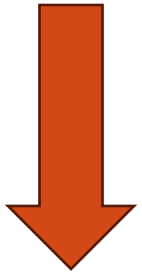
- No estabelecimento da conexão:
  - cliente e servidor permanecem indicando qual o **(1) tamanho de seus buffers** de recebimento de pacotes
    - isto é, informam quantos bytes são capazes de receber
    - com isso, não existe o risco de o cliente enviar mais informações do que o servidor possa processar e vice-versa.
- **Controle de fluxo** permanece durante toda a conexão

SERVIDOR TCP (CONT.)

- O processo de estabelecimento de conexões é realizado por meio do ***three-way handshake***, da seguinte forma:
  - O cliente envia um pacote de estabelecimento de conexão chamado SYN;
  - Caso aceite a conexão, o servidor responde com um pacote de reconhecimento SYN/ACK;
  - O cliente responde com uma confirmação ACK.

SERVIDOR TCP (CONT.)

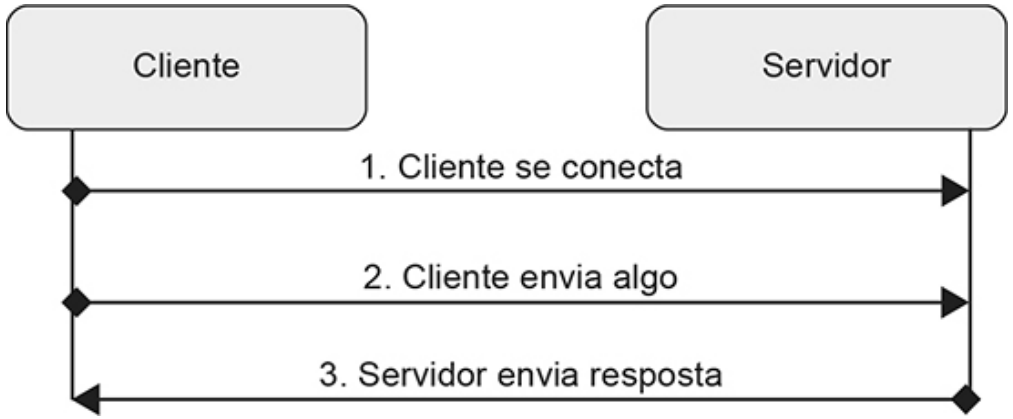
- Pacotes TCP possuem um número de série dentro da conexão
  - Permite identificar pacotes fora de ordem
    - com isso, essa situação pode ser corrigida
  - Quando um pacote não for recebido (identifica-se essa situação pelo número de sequência dos pacotes), ou for recebido com erro, o destinatário envia uma mensagem ao remetente indicando que o pacote deve ser retransmitido.



Controle de sequência
Controle de erros com retransmissão

LEMBRANDO QUE...

- Para a COMUNICAÇÃO com um servidor, o cliente precisa de duas informações:
  - O **endereço IP** do servidor (pode ser um nome, se o mesmo for reconhecido pelo sistema operacional);
  - A **porta** que o servidor está ouvindo (um endereço lógico, que normalmente está relacionada com o tipo de serviço fornecido).



QUANDO UTILIZAR O PROTOCOLO TCP?

- Quando o volume de dados requer vários pacotes e as informações tem que chegar ao destino em ordem e sem perda
  - navegação em páginas web
  - e-mail
  - transferência de arquivos
  - acesso remoto.

E AGORA???

- VAMOS IMPLEMENTAR UM CLIENTE E UM SERVIDOR TCP?
- O QUE SERÁ NECESSÁRIO?

A criação de um servidor TCP deve seguir os seguintes passos:

1. realize uma chamada ao sistema operacional por meio de um comando da API de *sockets* informando que o processo utilizará o protocolo de transporte TCP;
2. vincule o processo à porta por meio de outra chamada ao SO de ligação processo-porta (comando *bind*);
3. realize uma chamada ao sistema operacional colocando a porta em estado "LISTEN", ou seja, aguardando conexões;
4. ao receber uma requisição de conexão, aceite-a por meio de uma chamada ao SO "ACCEPT";
5. aguarde o recebimento da mensagem do cliente;
6. ao receber uma mensagem, imprima e envie uma resposta de reconhecimento ao cliente;
7. feche a conexão;
8. volte ao passo 4.

E o cliente que se conecta neste servidor precisa:

1. realizar uma chamada ao sistema operacional por meio de um comando da API de *sockets* informando que o processo utilizará o protocolo de transporte TCP;
2. realizar uma chamada de *connect* informando o endereço IP do servidor e a porta do servidor;
3. enviar a mensagem ao servidor;
4. receber e imprimir a mensagem de retorno;
5. fechar a conexão.

CÓDIGO CLIENTE

Script 9.1: client-1.py

```
1  #!/usr/bin/env python3
2  import socket
3
4  print("Cliente")
5  # Cria o objeto socket
6  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  # Obtém o nome do host e configura a porta
8  host = socket.gethostname()
9  print ("Host =",host)
10 port = 9999
11
12 # Conecta o socket no host e na porta especificados
13 s.connect((host, port))
14 # Envia algo
15 s.sendall(bytes("texto original",'UTF-8'))
16 # Recebe a resposta
17 rec = s.recv(1024)
18 print("Recebido",rec.decode("utf-8"))
19
20 s.close()
```

Explicação do código

- **linha 2**: Importa a biblioteca que contém a classe socket.
- **linha 6**: Cria o objeto socket. Um socket é a forma mais simples com o qual dois hosts podem se comunicar. Socket. AF_INET e socket.SOCK_STREAM são parâmetros que definem, respectivamente, a família de endereços e o tipo do socket (costumam ser valores padrão nesse tipo de aplicação).
- **linha 8**: Determina o nome do host (ou seja, o nome do computador) em que o script está sendo executado.
- **linha 10**: Atribuímos (arbitrariamente) a porta 9999 para ser usada pelo script.
- **linha 13**: Conecta o cliente ao servidor. Supõe-se que tanto o cliente quanto o servidor estão no mesmo computador! (pois a linha 8 estabelece que o host é o computador que está executando esse script).
- **linha 15**: Envia dados ao servidor.
- **linha 17**: Recebe a resposta do servidor.
- **linha 20**: Fecha a conexão com o servidor.

CÓDIGO SERVIDOR

Script 9.2: server-1.py

```
1  #!/usr/bin/env python3
2  import socket
3  print("Servidor")
4  # Cria o objeto socket
5  serversocket = socket.socket(
6                 socket.AF_INET, socket.SOCK_STREAM)
7  serversocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
8  # Obtém o nome do host e configura a porta
9  host = socket.gethostname()
10 print ("Host =",host)
11 port = 9999
12 # Liga o socket ao host e porta
13 serversocket.bind((host, port))
14 serversocket.listen(1)
15
16 while True:
17     clientsocket,addr = serversocket.accept()
18     #Aguarda a conexão de um cliente
19     recebido = clientsocket.recv(1024)
20     if not recebido: break
21     #transforma byte em string
22     print(recebido)
23     strRecebido = recebido.decode('UTF-8')
24     print(str(host), ":", strRecebido)
25     clientsocket.send( bytes(strRecebido.upper(), 'UTF-8'))
26 clientsocket.shutdown(socket.SHUT_RDWR)
27 clientsocket.close()
```

Explicação do código

- **linhas 5 e 6**: Cria um socket do tipo servidor. O socket servidor basicamente "reserva" a porta e cria um socket cliente para tratar o fluxo de dados.
- **linha 7**: Configura o socket servidor.
- **linha 13**: Informa ao sistema operacional que esse socket servidor vai estar ligado à porta especificada.
- **linha 14**: Aguarda por conexões realizadas ao socket.
- **linha 15**: Aceita uma conexão. Ao se conectar, retorna dois objetos, um socket cliente e o endereço do cliente que se conectou.
- **linha 26**: Termina a conexão.
- **linha 27**: Libera os recursos relacionados com a conexão. Note que o método close() não termina a conexão imediatamente, por isso o comando shutdown é acionado na linha anterior.

Para saber mais sobre conexões TCP consulte https://docs.python.org/3/library/socket.html

MÃOS A OBRA

Vamos executar os códigos cliente e servidor TCP?

Para isso:

- Baixe os códigos do Moodle ou digite os códigos apresentados nos dois slides anteriores
- Abra-os no Visual Studio Code
  - Baixe o interpretador Python
    - O Visual Studio Code irá solicitar para você fazer isso quando você pedir para executar um dos código (responda ok)

REFERÊNCIAS

- SCHMITT, Marcelo A R.; PERES, André; LOUREIRO, César A H. **Redes de computadores**: nível de aplicação e instalação de serviços. (Tekne). Porto Alegre: Grupo A, 2013. E-book. ISBN 9788582600948. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582600948/. Acesso em: 01 ago. 2023.
- NETO, Roberto Fernandes T.; SILVA, Fábio Molina da. **Introdução à Programação para Engenharia**: Usando a Linguagem Python. São Paulo: Grupo GEN, 2022. E-book. ISBN 9788521638346. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521638346/. Acesso em: 03 out. 2023.
- socket — Low-level networking interface. Disponível por www em https://docs.python.org/3/library/socket.html. Acesso em 03 de outubro de 2023.
- Código servidor node.js
  - https://expressjs.com/en/starter/hello-world.html
  - https://expressjs.com/en/starter/installing.html
- Algumas figuras foram retiradas de https://br.freepik.com.