Laboratório 1
Introdução à programação com sockets

Sistemas Distribuídos (MAB-367)
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ — 2 de junho de 2020

Introdução

O objetivo deste Laboratório é introduzir a **programação com sockets** usando a linguagem Python.

O **módulo socket** de Python provê acesso à interface Socket BSD. A função `socket()` retorna um objeto cujos métodos implementam as chamadas de sistema de socket.

API de sockets para conexões TCP: lado ativo

- **socket()**
  - cria um socket usado para comunicação e retorna um descritor
- `connect()`
  - estabelece conexão com o par passivo
- **send()**
  - envia dados para o par remoto da conexão
- **recv()**
  - receve dados da conexão
- **close()**
  - desaloca o socket

API de sockets para conexões TCP: lado passivo

- `bind()`
  - especifica a máquina e porta na qual esse elemento irá esperar por conexões
- `listen()`
  - coloca o socket no modo passivo para torná-lo disponível para aceitar conexões
- `accept()`
  - aceita a primeira solicitação de conexão na fila

Neste laboratório vamos praticar com as funções básicas da API de socket apresentadas acima. Para cada atividade, siga o roteiro proposto.

Atividade 1

Objetivo: Executar uma aplicação distribuída básica que faz uso de sockets.

Roteiro:

1. Abra os arquivos `passivo.py` e `ativo.py` disponíveis na página da disciplina.
2. Entenda como se dá o estabelecimento da conexão entre o par de elementos da aplicação e a comunicação entre eles.
3. Execute o arquivo `passivo.py` no terminal (`python passivo.py`). Em seguida, execute o arquivo `ativo.py` em outro terminal (`python ativo.py`). Veja os resultados impressos na tela.

Atividade 2

Objetivo: Desenvolver uma aplicação distribuída básica usando o modelo de interação **requisição/resposta**, com um processo **servidor** e um processo **cliente**.

Roteiro:

1. Implementar o processo **servidor** que ofereça as seguintes operações:
   (a) Somar dois números;
   (b) Subtrair dois números;
   (c) Multiplicar dois números;
   (d) Dividir dois números.
2. Implementar o processo **cliente** que permita ao usuário interagir com o servidor fazendo várias requisições às operações oferecidas e exibindo os resultados na tela.
3. Experimentar a aplicação com um processo cliente e um processo servidor.