

**Universidade de Brasília**

**Instituto de Ciências Exatas**

**Departamento de Ciência da Computação**

**Redes de Computadores – 2025.2**



## **Trabalho de Programação — Chat P2P**

**Integrantes:**

**Júlia Paulo Amorim - 241039270**

**Rafael Medeiros de Alencar - 241020760**

**Samuel Philipe Domingues de Oliveira - 241002458**

---

Link para a página do repositório: [pyp2p-rdv](https://pyp2p-rdv.github.io)

### **1. Requisitos Mínimos**

- Python 3.10 ou superior
- Conexão com internet (para acessar o servidor Rendezvous)
- Portas TCP disponíveis

---

### **2. Dependências**

- Não há dependências externas - o projeto usa apenas bibliotecas padrão do Python
- Bibliotecas utilizadas:

```
import json          # Para serialização de dados
import socket        # Para comunicação TCP
import threading     # Para concorrência
import logging       # Para logs do sistema
import time          # Para timestamps e delays
import requests      # Para comunicação HTTP com rendezvous
```

---

## 3. Instalação

1. Clone o repositório:

```
git clone https://github.com/leitaonerd/pyp2p-rdv.git
cd pyp2p-rdv
```

2. (Opcional) Crie um ambiente virtual:

```
python3 -m venv venv

# Windows
venv\Scripts\activate

# Linux/Mac
source venv/bin/activate
```

---

## 4. Execução do Cliente

### Método 1: Execução Direta (Recomendado)

```
# No diretório raiz do projeto
python -m src.client.main
```

### Método 2: Com arquivo de configuração

Crie um arquivo config.json:

```
{
  "name": "seu_nome",
  "namespace": "CIC",
  "rendezvous_host": "pyp2p.mfcaetano.cc",
  "rendezvous_port": 8080,
  "listen_port": 6000
}
```

Execute com:

```
python -m src.client.main --config config.json
```

### Método 3: Com nível de log personalizado

```
python -m src.client.main --log-level DEBUG
```

---

## Configurações Disponíveis

Parâmetro	Padrão	Descrição
name	alice	Seu nome de usuário
namespace	CIC	Namespace/sala do chat
rendezvous_host	pyp2p.mfcaetano.cc	Host do servidor Rendezvous
rendezvous_port	8080	Porta do servidor Rendezvous
listen_port	6000	Porta local para receber conexões
ttl_seconds	7200	Tempo de vida do registro (2h)
discovery_interval	15.0	Intervalo de descoberta (segundos)
ping_interval	30.0	Intervalo de PING (segundos)
max_reconnect_attempts	5	Máximo de tentativas de reconexão

---

## Comandos da CLI

Uma vez que o cliente esteja rodando, você verá o prompt `pyp2p>`. Os comandos disponíveis são:

Comando	Descrição	Exemplo
<code>/peers</code>	Lista todos os peers conhecidos	<code>/peers</code>
<code>/peers *</code>	Lista todos os peers	<code>/peers *</code>
<code>/peers #ns</code>	Lista peers de um namespace	<code>/peers #CIC</code>
<code>/msg &lt;peer&gt; &lt;msg&gt;</code>	Envia mensagem direta	<code>/msg bob@CIC Olá!</code>
<code>/pub * &lt;msg&gt;</code>	Broadcast para todos	<code>/pub * Olá a todos!</code>
<code>/pub #ns &lt;msg&gt;</code>	Mensagem para namespace	<code>/pub #CIC Olá CIC!</code>
<code>/conn</code>	Mostra conexões ativas	<code>/conn</code>
<code>/rtt</code>	Mostra latência (RTT)	<code>/rtt</code>
<code>/reconnect</code>	Força reconexão	<code>/reconnect</code>
<code>/log &lt;nível&gt;</code>	Ajusta nível de log	<code>/log DEBUG</code>
<code>/help</code>	Mostra ajuda	<code>/help</code>
<code>/quit</code>	Encerra o cliente	<code>/quit</code>

## Testando a Comunicação no Servidor

### Passo 1: Inicie o cliente

```
python -m src.client.main
```

Você verá algo como:

```
[INFO] Inicializando cliente PyP2P para peer alice@CIC  
[INFO] PeerServer escutando em 0.0.0.0:6000  
[INFO] Registrado no rendezvous como X.X.X.X:6000  
pyp2p>
```

## **Passo 2: Descubra outros peers**

```
pyp2p> /peers
```

Se houver outros peers online, você verá a lista deles.

## **Passo 3: Conecte-se a um peer**

```
pyp2p> /reconnect
```

Isso força a descoberta e tentativa de conexão com peers disponíveis.

## **Passo 4: Envie uma mensagem**

```
pyp2p> /msg bob@CIC Olá Bob, tudo bem?
```

## **Passo 5: Envie broadcast**

```
pyp2p> /pub * Mensagem para todos!
```

## **Passo 6: Verifique conexões e latência**

```
pyp2p> /conn  
pyp2p> /rtt
```

## **Passo 7: Encerre o cliente**

```
pyp2p> /quit
```

# **Testando com Dois Clientes Localmente**

Para testar a comunicação P2P, você pode rodar dois clientes na mesma máquina com portas diferentes:

### **Terminal 1:**

```
# Crie config1.json  
echo {"name": "alice", "namespace": "CIC", "listen_port": 6001} > config1.json  
python -m src.client.main --config config1.json
```

### **Terminal 2:**

```
# Crie config2.json  
echo {"name": "bob", "namespace": "CIC", "listen_port": 6002} > config2.json  
python -m src.client.main --config config2.json
```

Agora ambos estão registrados no servidor Rendezvous. Use `/reconnect` em ambos para que eles se descubram e conectem, então teste com `/msg`.

---

## Troubleshooting

### "Erro de rede com rendezvous"

- Verifique sua conexão com a internet
- Confirme que o servidor `pyp2p.mfcaetano.cc:8080` está acessível
- Tente: `ping pyp2p.mfcaetano.cc`

### "Não foi possível iniciar PeerServer"

- A porta já está em uso. Altere `listen_port` no `config.json`

### "Peer não encontrado"

- O peer pode estar offline ou em outro namespace
- Use `/peers` para ver peers disponíveis
- Use `/reconnect` para atualizar a lista

### Nenhum peer aparece

- Pode não haver outros peers online no momento
- Teste com dois clientes localmente (veja seção acima)

## Logs e Debug

Para ver logs detalhados:

```
python -m src.client.main --log-level DEBUG
```

Ou durante a execução:

```
pyp2p> /log DEBUG
```

Níveis disponíveis: DEBUG, INFO, WARNING, ERROR

---

## Arquitetura final do sistema

```
pyp2p-rdv/  
├── src/  
│   ├── client/  
│   │   ├── main.py           # Inicializa aplicação e logging  
│   │   ├── p2p_client.py     # Lógica principal  
│   │   └── rendezvous_connection.py # Comunicação com servidor Rendezvous
```

peer_connection.py	# Controle das conexões TCP
message_router.py	# Envio e publicação de mensagens
peer_table.py	# Controle de estado e reconexões
peer_server.py	# Controle de conexões inbounds
state.py	# Armazenamento dos peers conhecidos
cli.py	# Interface de linha de comando
config.py	# Configurações do cliente
tools/	
rc_tester.py	# Ferramentas de teste
TASKS.md	# Documentação
pyp2p_Relatório	# Relatório em Jupyter Notebook

---