Protocolo P2P da Rede de Testes Distribuída

Grupo X

29 de maio de 2025

1 Introdução

Este documento descreve o protocolo de comunicação **P2P** utilizado entre os nós da Rede de Testes Distribuída. O protocolo define como os nós se descobrem, trocam tarefas, monitoram falhas e realizam eleições para recuperação automática.

2 Visão Geral

A comunicação entre nós é realizada via **UDP**, utilizando mensagens serializadas em **JSON**. Cada nó escuta em uma porta UDP específica e envia mensagens diretamente para os peers conhecidos.

3 Tipos de Mensagens

A tabela abaixo lista os principais tipos de mensagens trocadas entre os nós:

Tipo	**Descrição**
CACHE_UPDATE	Atualização do cache de estado da rede
HEARTBEAT	Sinaliza que o nó está ativo
CONNECT	Solicita entrada em uma rede existente
CONNECT_REP	Resposta à solicitação de conexão
TASK_ANNOUNCE	Anúncio de disponibilidade de tarefas
TASK_REQUEST	Solicitação de tarefa
TASK_SEND	Envio de tarefa para execução
TASK_CONFIRM	Confirmação de recebimento de tarefa para execução
PROJECT_ANNOUNCE	Anúncio de novo projeto
RECOVERY_ELECTION	Mensagem de candidatura para eleição de nó de recuperação
EVALUATION_RESPONSIBILITY_UPDATE	Atualização de responsável por projeto/avaliação
RECOVERY_ELECTION_REP	Resposta de participação na eleição de recuperação
RECOVERY_ELECTION_RESULT	Resultado da eleição de recuperação

4 Formato das Mensagens

Todas as mensagens são enviadas em formato JSON, com os seguintes campos principais:

Listing 1: Formato geral de mensagem

```
{
  "cmd": "HEARTBEAT",
  "data": { ... },
  "ip": "192.168.1.10",
  "port": 25000,
  "timestamp": 1710000000.0
}
```

- **cmd**: Tipo da mensagem (ver tabela acima)
- **data**: Dados específicos do comando
- **ip**, **port**: Endereço do nó remetente
- **timestamp**: Momento do envio (em segundos desde epoch)

5 Fluxos de Comunicação

5.1 a) Entrada de um novo nó

1. O nó envia uma mensagem `CONNECT` para um nó conhecido.
2. O nó existente responde com `CONNECT_REP`, informando peers e atribuindo um ID.
3. O novo nó passa a enviar/receber heartbeats e participar da rede.

5.2 b) Distribuição de tarefas

1. O nó com tarefas envia `TASK_ANNOUNCE`.
2. Um nó ocioso responde com `TASK_REQUEST`.
3. A tarefa é enviada via `TASK_SEND`.
4. O recebimento da tarefa é confirmado via `TASK_CONFIRM`.

5.3 c) Detecção de falhas

1. Nós enviam `HEARTBEAT` periodicamente.
2. Se um nó não recebe heartbeat de outro por tempo limite, considera-o falho.

5.4 d) Eleição de nó de recuperação

1. Nós detectam falha e enviam `RECOVERY_ELECTION`.
2. Os peers respondem com `RECOVERY_ELECTION_REP`.
3. O resultado da eleição é divulgado via `RECOVERY_ELECTION_RESULT`.
4. O nó eleito assume as tarefas do nó falho e envia `EVALUATION_RESPONSIBILITY_UPDATE`.

6 Exemplos de Mensagens

HEARTBEAT

```
{
  "cmd": "HEARTBEAT",
  "data": {
    "id": "123456789",
    "peers_ip": ["192.168.1.11:25000"]
  },
  "ip": "192.168.1.10",
  "port": 25000,
  "timestamp": 1710000000.0
}
```

TASK_SEND

```
{
  "cmd": "TASK_SEND",
  "data": {
    "project_name": "meu_projeto",
    "module": "test_mod1.py",
    "api_port": 5001,
    "eval_id": "1234567890"
  },
  "ip": "192.168.1.10",
  "port": 25000,
  "timestamp": 1710000000.0
}
```

RECOVERY_ELECTION

```
{
  "cmd": "RECOVERY_ELECTION",
  "data": {
    "candidate_id": "123456789",
    "failed_node": "987654321",
    "timestamp": 1710000000.0
  },
```

```
  "ip": "192.168.1.10",
  "port": 25000,
  "timestamp": 1710000000.0
}
```

RECOVERY_ELECTION_REP

```
{
  "cmd": "RECOVERY_ELECTION_REP",
  "data": {
    "candidate_id": "123456789",
    "failed_node": "987654321",
    "timestamp": 1710000001.0
  },
  "ip": "192.168.1.11",
  "port": 25000,
  "timestamp": 1710000001.0
}
```

RECOVERY_ELECTION_RESULT

```
{
  "cmd": "RECOVERY_ELECTION_RESULT",
  "data": {
    "winner_id": "123456789",
    "failed_node": "987654321"
  },
  "ip": "192.168.1.10",
  "port": 25000,
  "timestamp": 1710000002.0
}
```

7 Portas e Protocolo

- **Comunicação entre nós**: UDP, portas 8001, 8002 etc (uma por nó)
- **API HTTP**: TCP, portas 5001, 5002, etc (uma por nó)

8 Observações

- Todos os nós implementam o protocolo acima e aceitam mensagens de qualquer peer ativo na rede.
- O sistema é tolerante a falhas e redistribui tarefas automaticamente em caso de falha de um nó.
- Para detalhes de cada campo das mensagens, consulte a implementação em `src/network/message`