MiniBit

Trabalho final de Sistemas Distribuídos

SIMPLIFICAÇÕES

- **Divisão de Blocos:** O sistema opera com uma abstração de "blocos numerados" (block_0, block_1, ...), sem a necessidade de um arquivo .torrent para descrever a estrutura do arquivo e seus blocos;

- **Uso de Flask:** Tanto o *tracker.py* quanto o *peer.py* utilizam o microframework *Flask* para implementar as funcionalidades de servidor HTTP. Escolha feita para facilitar o gerenciamento da comunicação de rede.

- **Arquitetura**: https://www.mermaidchart.com/app/projects/8d3c36f1-5150-435b-9bca-f138e535a96f/diagrams/59bedf74-b67c-465b-ae24-74d741ee1738/version/v0.1/edit

TRACKER (rastreador)

- Faz a divisão do “arquivo”;
- Age como seeder no início, distribuindo os blocos para os peers que se registram;
- Gerencia a lista de peers ativos e os blocos que cada um possui;
- Após o registro, não faz mais distribuições de blocos.

TRACKER(rastreador)

ENDPOINT	DESCRIÇÃO
/register	Registro de peer e entrega de blocos iniciais.
/get_peers	Lista de peers ativos (excluindo o solicitante)
/get_block_info	Quais peers possuem determinados blocos
/update_blocks	Atualiza os blocos que o peer possui

PEER (par)

- Faz o trabalho “pesado” da aplicação;
- Registro no tracker e descoberta de outros peers;
- Download de Blocos: baixa blocos de outros peers sob a lógica do "Rarest First";
- Controla o envio de blocos para outros peers através de uma versão simplificada das estratégias "Tit-for-Tat" e "Optimistic Unchoke".
- Transição para Seeding: ao completar o download, ele permanece ativo na rede.