

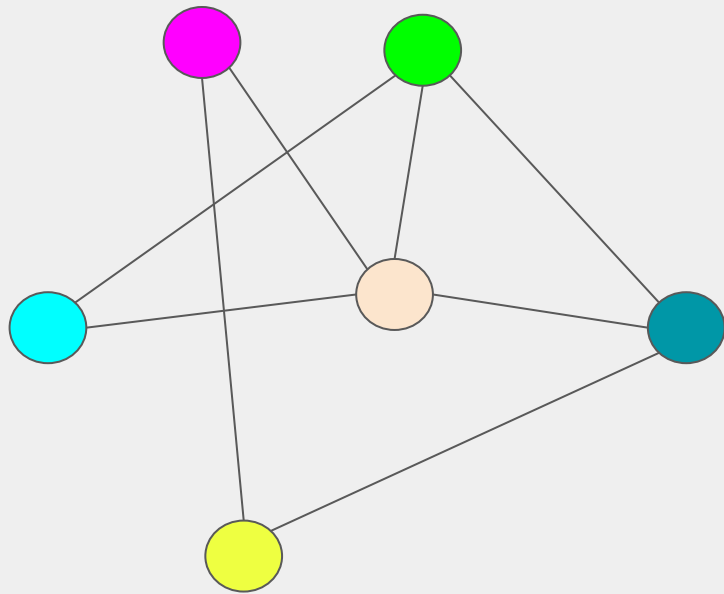
Tornando a Soberania Pessoal
Mais Acessível: Avanços em
Clientes Leves

Agenda

- Por que rodar um node?
- Quais são os desafios em rodar um:
 - Espaço em disco,
 - Longa espera,
 - Necessidade de um servidor dedicado.
- Clientes leves: vantagens e desvantagens
- Reduzindo espaço em disco com *pruning* e *utreeexo*
- Skipping IBD
- Busque o seu balanço privadamente
- Nodes embarcados

Por que rodar um node

- Bitcoin é uma rede p2p
 - Não existem servidores centrais
 - Nem entes centrais que a controlam
- A rede é formada por nodes, que de forma
- Independente participam da mesma
- Caso você não rode seu próprio node, você utilizará o node de terceiros



Por que rodar um node

O resultado é a perda de:


- Segurança: O node que você confia pode mentir
- Privacidade: Ele pode ligar todas as suas transações à sua pessoa
- Descentralização: A rede fica menos resiliente, pois menos pessoas estão validando



Porém, existem desafios

- Rodar um node pode ser custoso, consumir tempo e não ser muito pratico
- Nodes utilizam muito recurso computacional
 - Disco
 - Rede
 - CPU
- Além de serem complexos para usuários leigos
 - Instalar o Bitcoin Core
 - Instalar um *Electrum Server*
 - *Tailscale* ou TOR para acessar de fora de casa

Porém, existem desafios



DELL
Micro Computador Dell 5070: J4105, 8 Giga Ddr4, Ssd 128 Giga

R\$ 999,99


R\$ 899⁹⁹ 10% OFF

em 10x R\$ 90 sem juros

[10% OFF Linha de crédito](#)

Frete grátis

Patrocinado



Raspberry Pi4 Pi 4 Model B 4gb Ram
Novo Com Nota Fiscal

5.0 ★★★★★ (2)

R\$ 575

em 12x R\$ 55,75

[10% OFF Linha de crédito](#)

Frete grátis

Enviado pelo **FULL**



Mini Pc Tecs Amd G-t48e 2gb

R\$ 299

em 9x R\$ 33,22 sem juros

[10% OFF Linha de crédito](#)

Chegará amanhã

Usado

Clientes leves

- Um cliente leve é aquele que consegue entregar um nível satisfatório de privacidade e segurança, sem requerer a mesma quantidade de recursos que um node.
- O tipo de cliente leve mais comum, conhecido como **SPV**, possui alguns problemas
 - Inseguros em caso de um ataque de 51%
 - Não definem como encontrar transações para sua wallet
- Mas possui como vantagens
 - Consumir menos de um GB de dados e disco
 - Quantidade negligível de CPU
 - *Sync* em alguns minutos

Entram clientes leves

- Com o tempo, algumas tecnologias foram desenvolvidas para melhorar esse tipo de cliente, garantindo mais privacidade e segurança
 - *Pruning*
 - *Compact Block Filters*
 - *Utreexo*
 - *PoW Fraud Proofs*
- Ainda existem desafios
 - Reduzir a banda utilizada
 - Disco
 - Tempo de *rescan*

Pruning

- Técnica já estabelecida
- Consiste em descartar dados de transações já confirmadas
- Esta informação é redundante para a validação dos próximos blocos
- Reduz o espaço necessário de ~600GB para meros ~10GB

Compact Block Filters

- Uma forma de verificar se existem transações destinadas a sua carteira, de forma local
- Requer ~15GB de filtros ao invés dos 600GB da *blockchain*
- Ótima privacidade

Utreexo

- O UTXO set, além de relativamente grande, requer muito I/O
- Utreexo reduz o UTXO set para menos de 1kb, e não requer nenhuma forma de I/O
- Blocos e transações precisam de uma **prova de inclusão** para serem validados

Proof of Work Fraud Proofs

- Permite “pular” o *IBD*
- Seguro enquanto ao menos $x\%$ dos mineradores forem honestos
 - X pode ser um valor qualquer, sendo 20% o valor utilizado na prática
- Requer o *download* da cadeia de cabeçalhos.

Implementação - Floresta

- Implementa todas as tecnologias citadas
- Pode ser utilizado como dependência em outros projetos
 - O node roda ao lado da sua aplicação, no mesmo dispositivo
 - Informações são servidas via protocolo *Electrum* ou **JSON-RPC**
 - Carteira somente leitura integrado
- Possível de utilizar em dispositivos de baixo poder computacional, como smartphones
- *Homepage*: <https://github.com/vinteumorg/Floresta>

Implementação - Neutrino

- Focado em *Compact Block Filters*
- Segurança **SPV** clássica
- Também pode ser utilizado como biblioteca
- *Homepage*: <https://github.com/lightninglabs/neutrino>

Implementação - Kyoto

- Similar ao neutrino, porém escrito em *Rust*, com suporte a outras linguagens via bindings
- **WIP** integração com **BDK**
- *Homepage: <https://github.com/rustaceanrob/kyoto>*

Conclusão

- Clientes leves podem levar melhor privacidade e segurança para mais pessoas
- Avanços estão sendo feitos para melhorar a *UX* dos mesmos
- Já existem projetos em ponto de produção que podem ser utilizados por usuários finais e desenvolvedores

Agradecimentos

