Blockchain, Criptomoedas & Tecnologias Descentralizadas

Broadcast P2P: Gossip (“fofoca”)

Prof. Dr. Marcos A. Simplicio Jr. – mjunior@larc.usp.br
Escola Politécnica, Universidade de São Paulo

Objetivos

- Estratégias e distribuição de dados em redes descentralizadas
  - Broadcast eficiente via Gossip
  - Aplicação em Blockchains: o caso do Bitcoin

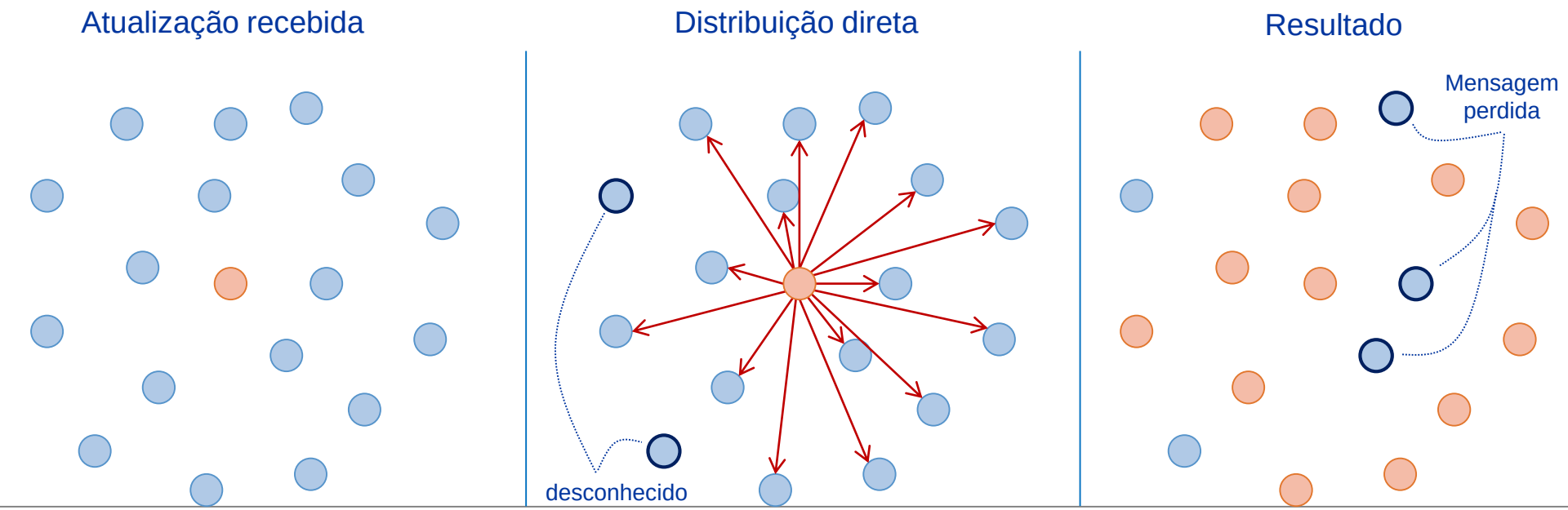
Distribuição de mensagens

- **Problema**: como disseminar informações em sistemas distribuídos sem visão global da rede?
  - Cenário comum em redes P2P: nó conhece alguns vizinhos…
  - Ex.: distribuição de transações em blockchains
- **Inicialização**: nó ingressante descobre alguns nós
  - No Bitcoin: aplicativo vem com endereços de "nós semente", ou pode receber endereços via DNS (e.g., seed.bitcoin.sipa.be)
  - Outros nós podem ser descobertos posteriormente via troca de mensagens (e.g., mensagens addr no Bitcoin)

- **Distribuição**: blocos e transações recebidos por nó são enviados para nós conhecidos
  - Assume-se cenário de **consistência eventual**: nem todos os nós têm a mesma informação pré-consenso

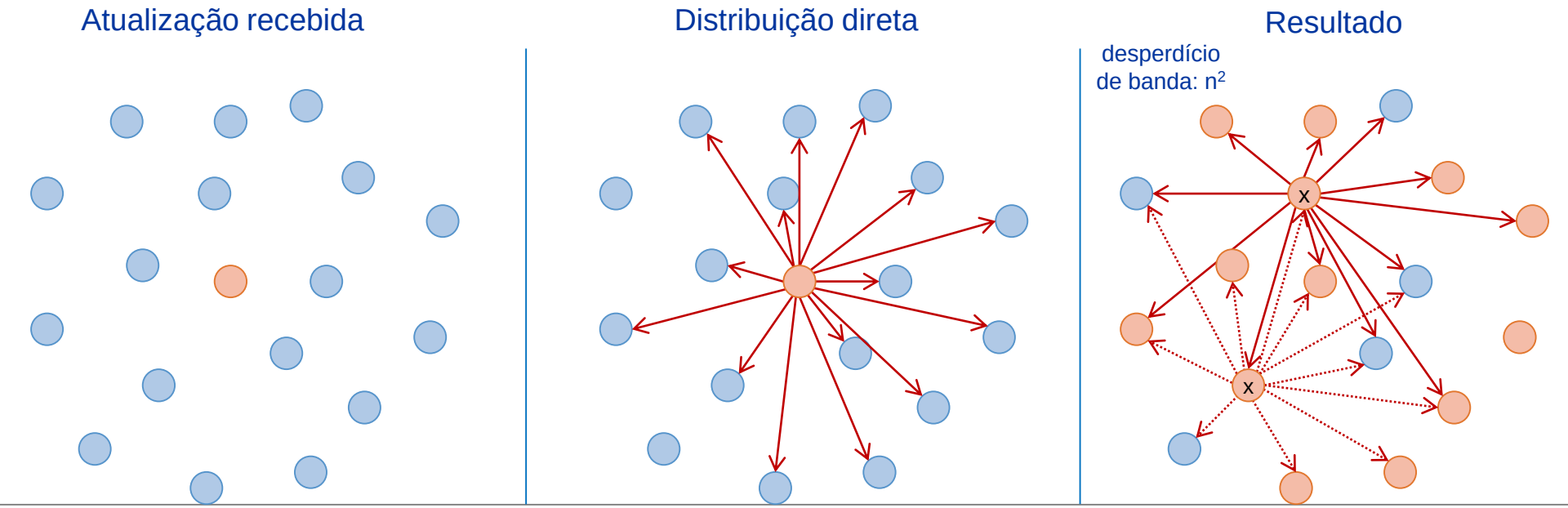
Distribuição de mensagens

- Mecanismo para distribuição epidêmica de dados
- Objetivo: evitar ineficiência de broadcast “clássico”
  - Ex. (distribuição direta): nó notifica todos os outros nós assim que recebe/cria uma atualização



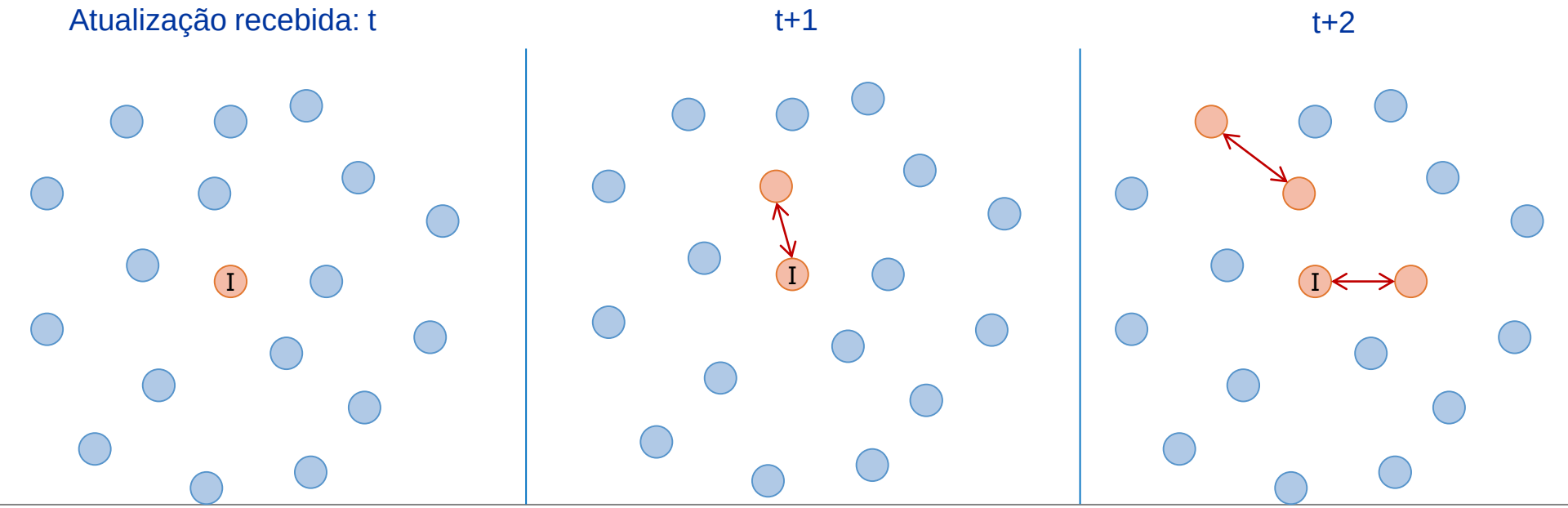
Distribuição de mensagens

- Mecanismo para distribuição epidêmica de dados
- Objetivo: evitar ineficiência de broadcast “clássico”
  - Ex. (inundação): nó notifica todos os outros *n-1* nós assim que recebe/cria uma atualização, e outros nós repetem o processo



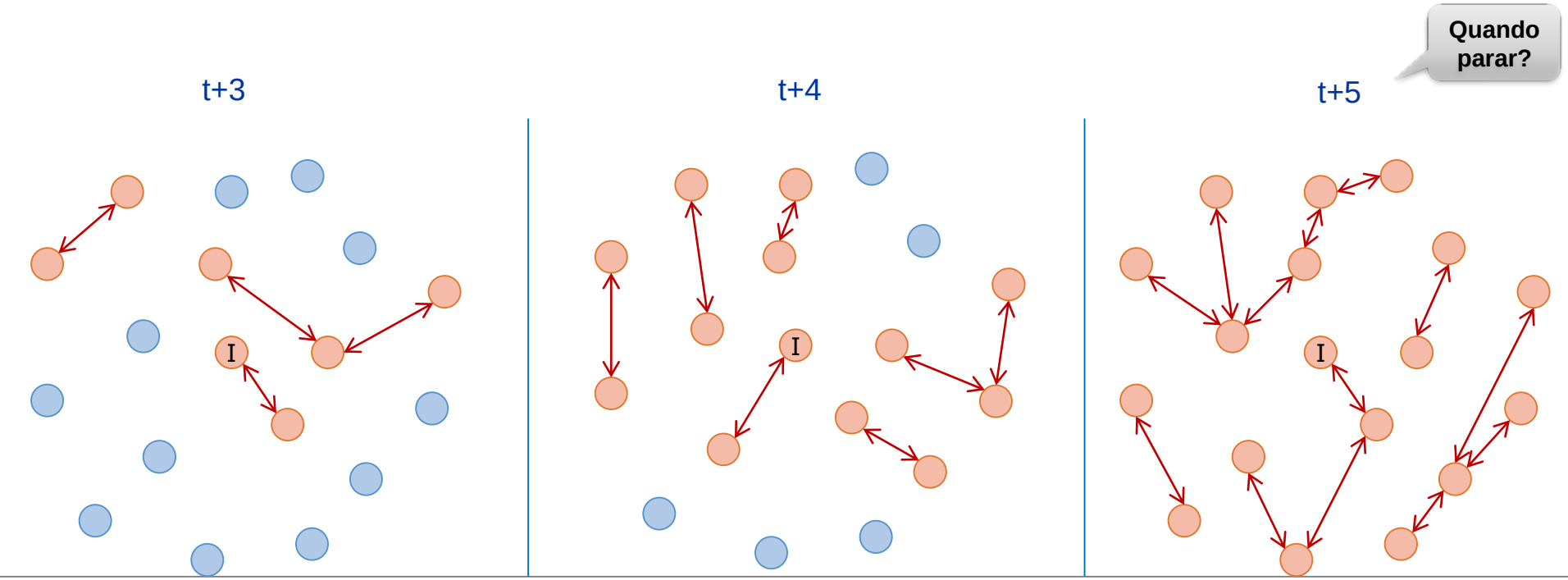
Distribuição de mensagens: gossip

- Mecanismo para distribuição epidêmica de dados
  - Dois estados (SI, ou *anti-entropy*): **suscetível** ou **infectado**
  - Se **push**: infectados informam atualização para nós aleatórios
  - Se **pull**: suscetíveis requisitam atualização de nós aleatórios



Distribuição de mensagens: gossip

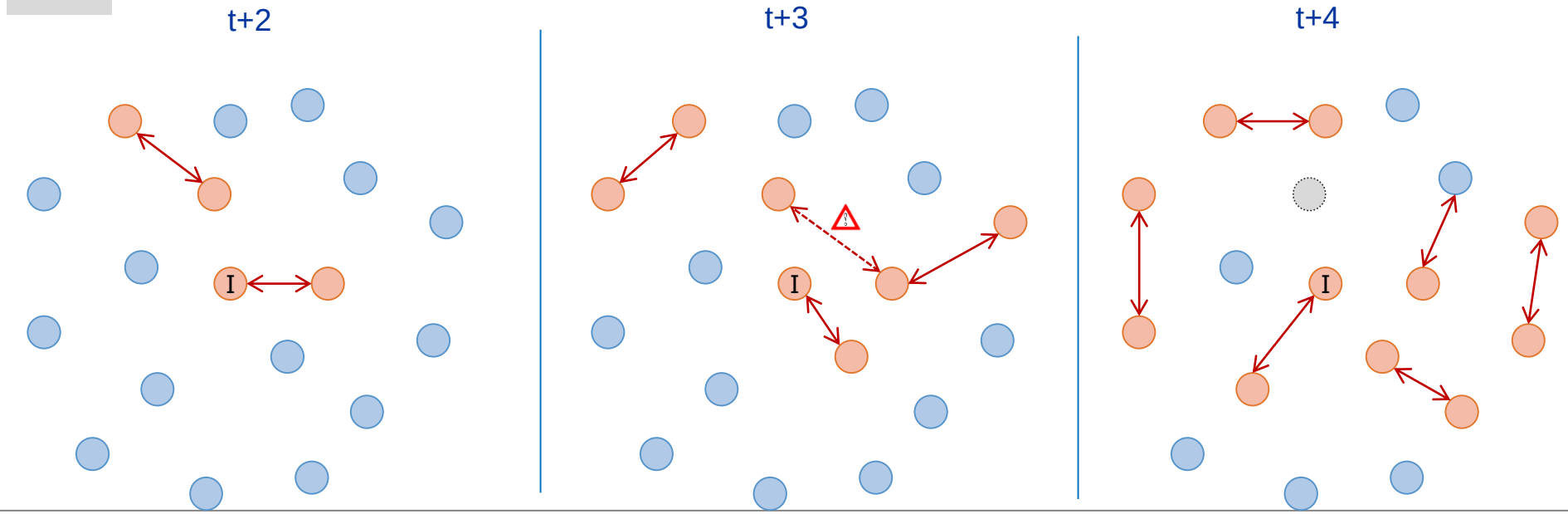
- Mecanismo para distribuição epidêmica de dados
  - Dois estados (SI, ou *anti-entropy*): **suscetível** ou **infectado**
  - Se **push**: infectados informam atualização para nós aleatórios
  - Se **pull**: suscetíveis requisitam atualização de nós aleatórios



Distribuição de mensagens: gossip

- Mecanismo para distribuição epidêmica de dados
  - **Três** estados (SIR, ou *rumor mongering*): **infectado**, **suscetível**, e **removido** (após fofoca "esfriar", para distribuição)
  - Ex.: **k fixo ➔** removido com probabilidade **1/k** após **k falhas**
    - Menor k: convergência mais rápida, maior chance de não-infecção

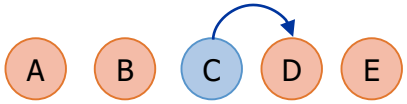
k=1



Distribuição de mensagens: gossip

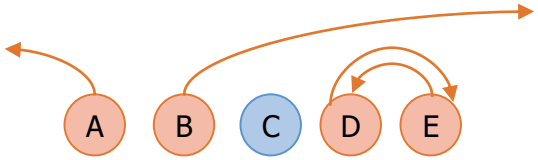
- Algumas observações
  - Push-pull ou pull distribui mais rápido que apenas push
  - Intuição:

(Push-)Pull:



Se apenas C está suscetível: contato com qualquer outro nó certamente leva C a estado infectado!

Push:

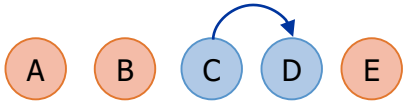


Se apenas C está suscetível: ainda é possível que C permaneça suscetível!

Distribuição de mensagens: gossip

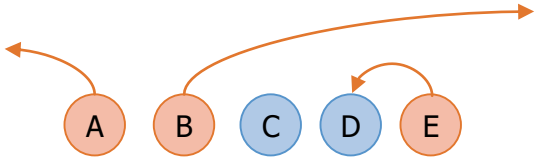
- Algumas observações
  - Push-pull ou pull distribui mais rápido que apenas push
    - Seja P_i a probabilidade de um nó estar suscetível na rodada i: probabilidade cai quadraticamente com pull

(Push-)Pull:



$$P_{i+1} \approx (P_i)^2$$

Push:



$$P_{i+1} \approx P_i(1-\frac{1}{n})^{n(1-P_i)} \approx P_i e^{-1}$$

Na rodada i+1, C só continua suscetível se contactar outro nó D também suscetível (probabilidade P_i)

$(1-\frac{1}{n})$	probabilidade de nó infectado escolher algum nó diferente de C
$n(1-P_i)$	número de nós infectados
n	número total de nós

Fonte: Eugene Bagdasaryan. CS 6410 -- Gossip Protocols. Cornell University, 2016.

Distribuição de mensagens: gossip

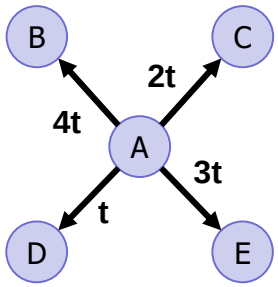
- Algumas observações:
  - Há várias variantes do Gossip aqui apresentado
    - Ex.: estado “removido” após contatar k nós infectados consecutivos
    - Ex.: vários vizinhos contatados (não apenas 1)
    - Ex.: sorteio de nós não considera vizinhos sabidamente infectados
  - Uso também em aplicações que buscam médias

Distribuição de mensagens: Bitcoin

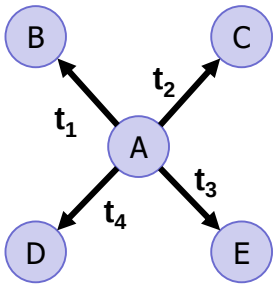
- Bitcoin: **Broadcast (push)**
  - **Hashes** de novas transações ou blocos: `inv` (de "inventário") enviados periodicamente a vizinhos
    - Solicitação de **dados** referentes aos hashes: `getdata`
    - `inv` também fornecido via **pull** (ex.: após reset de nó)
      - `mempool`: IDs de transações pendentes, ainda fora do blockchain
      - `getblocks`: hashes de blocos a partir de ID informado
  - **Blocos** construídos por mineradores: `block`
  - Comumente: 8 nós vizinhos
- Protocolos base:
  - Até 2015: Trickle (seleção aleatória, síncrona)
  - Após 2015: Diffusion (seleção assíncrona)

Distribuição de mensagens: Bitcoin

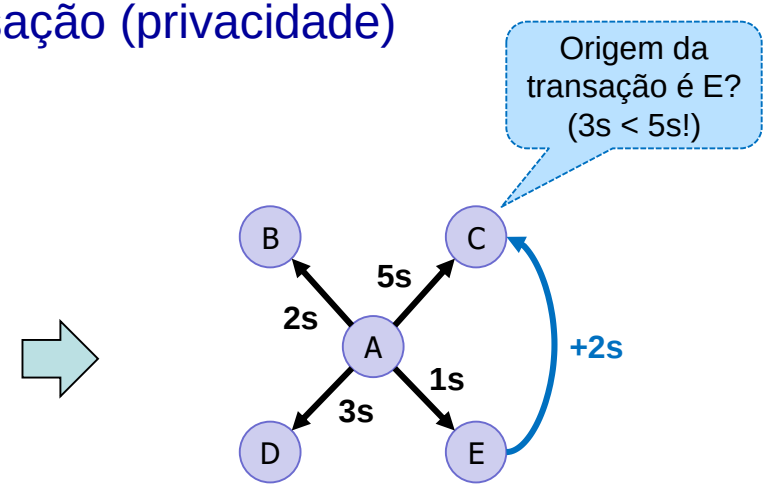
- Bitcoin: **Broadcast (push)**
  - Até 2015: Trickle (seleção aleatória, síncrona)
    - Envio a um vizinho a cada intervalo t
  - Após 2015: Diffusion (seleção assíncrona)
    - Intervalo aleatório exponencial: $t \in \{0, e^{\lambda}\}$
    - Objetivo: disfarçar origem da transação (privacidade)



Trickle

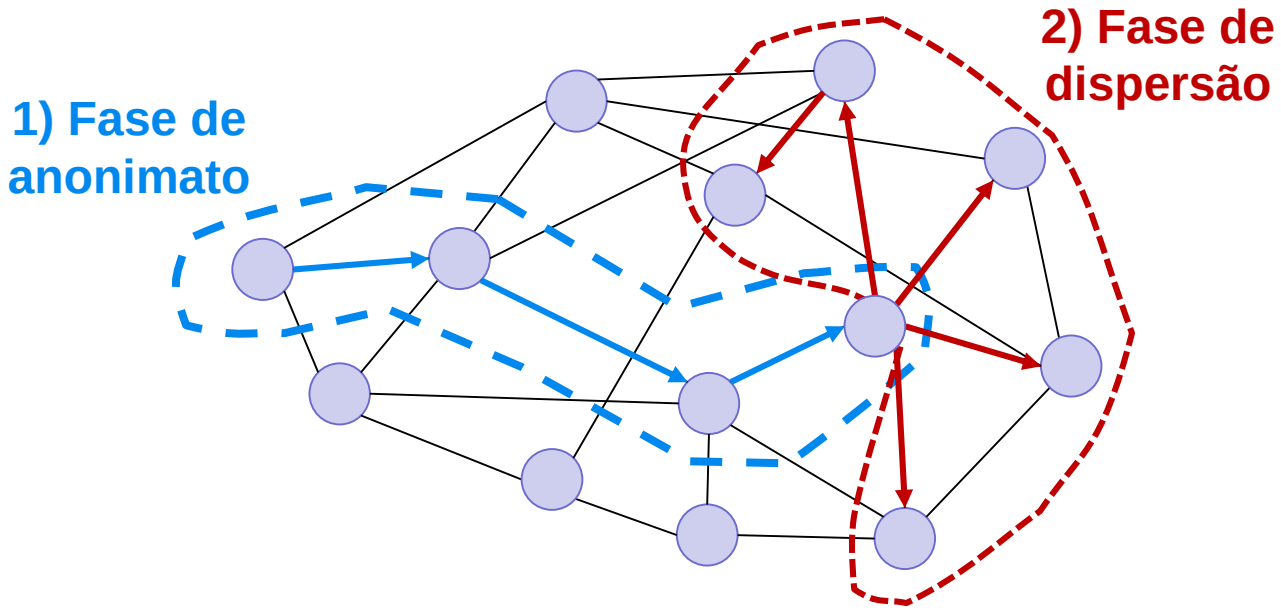


Diffusion



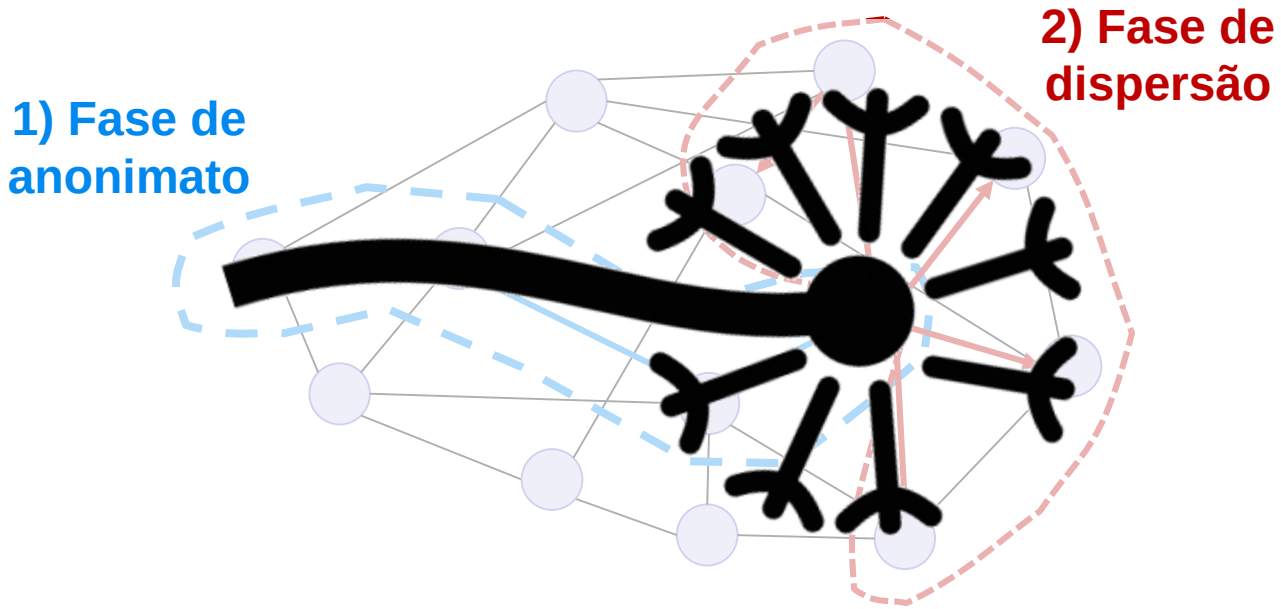
Distribuição de mensagens: Bitcoin

- Bitcoin: **Broadcast (push)**
  - Futuro (???): Dandelion++
    - De forma simplista: "**Tor** integrado a rede blockchain"
    - Usado no Monero desde 2020



Distribuição de mensagens: Bitcoin

- Bitcoin: **Broadcast (push)**
  - Futuro (???): Dandelion++
    - De forma simplista: “**Tor** integrado a rede blockchain”
    - Usado no Monero desde 2020



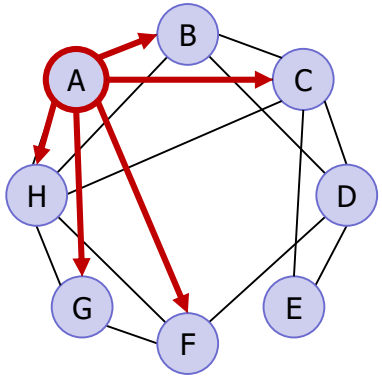
Distribuição de mensagens: Bitcoin

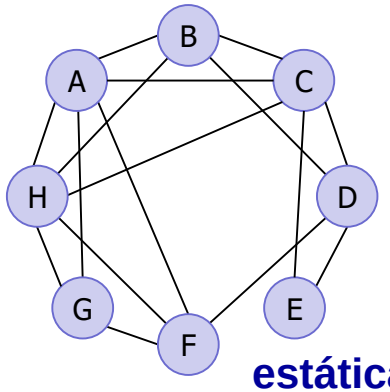
- Dandelion vs. Diffusion

Protocolo de broadcast

Topologia de anonimato

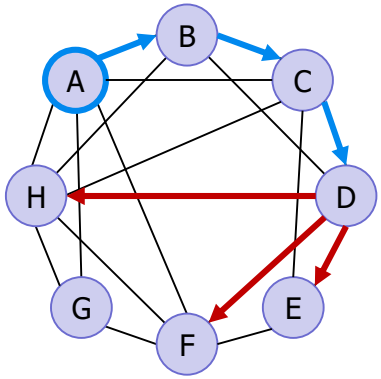
Diffusion:

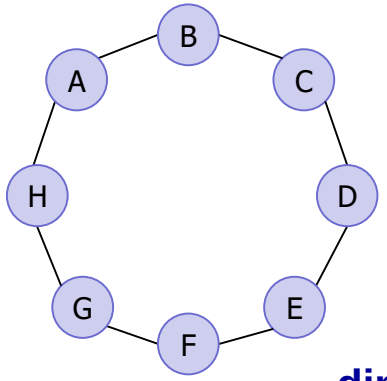


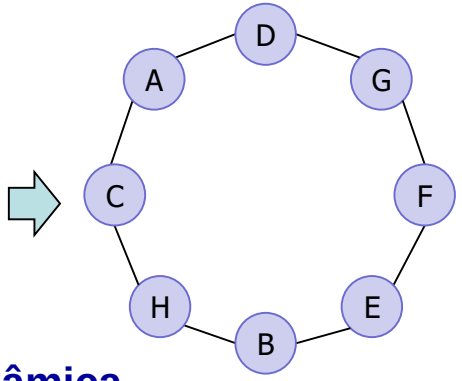


estática

Dandelion:







dinâmica

Blockchain, Criptomoedas & Tecnologias Descentralizadas

Broadcast P2P: Gossip

Prof. Dr. Marcos A. Simplicio Jr. – mjunior@larc.usp.br
Escola Politécnica, Universidade de São Paulo

Apoio financeiro: d24n.org <https://d24n.org>
ripple impact | university blockchain research initiative

USP

Referências

- M. Jelasity (2011). Gossip. In: Self-organising Software. Natural Computing Series. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17348-6_7
- E. Bagdasaryan. CS 6410: Gossip Protocols. Cornell University (2016) URL: https://www.cs.cornell.edu/courses/cs6410/2016fa/slides/19-p2p-gossip.pdf
- G. Fanti, S. Venkatakrishnan, S. Bakshi, B. Denby, S. Bhargava, A. Miller, P. Viswanath (2018). Dandelion++: lightweight cryptocurrency networking with formal anonymity guarantees. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2(2), 1-35.
  - Ver também: https://youtu.be/BwFUUWkp8YQ
- H. Qureshi (2019). Bitcoin's P2P Network. Nakamoto.com. URL: https://nakamoto.com/bitcoins-p2p-network/
- Bitcoin Developer (online). Referência: P2P Network. URL: https://developer.bitcoin.org/reference/p2p_networking.htm l