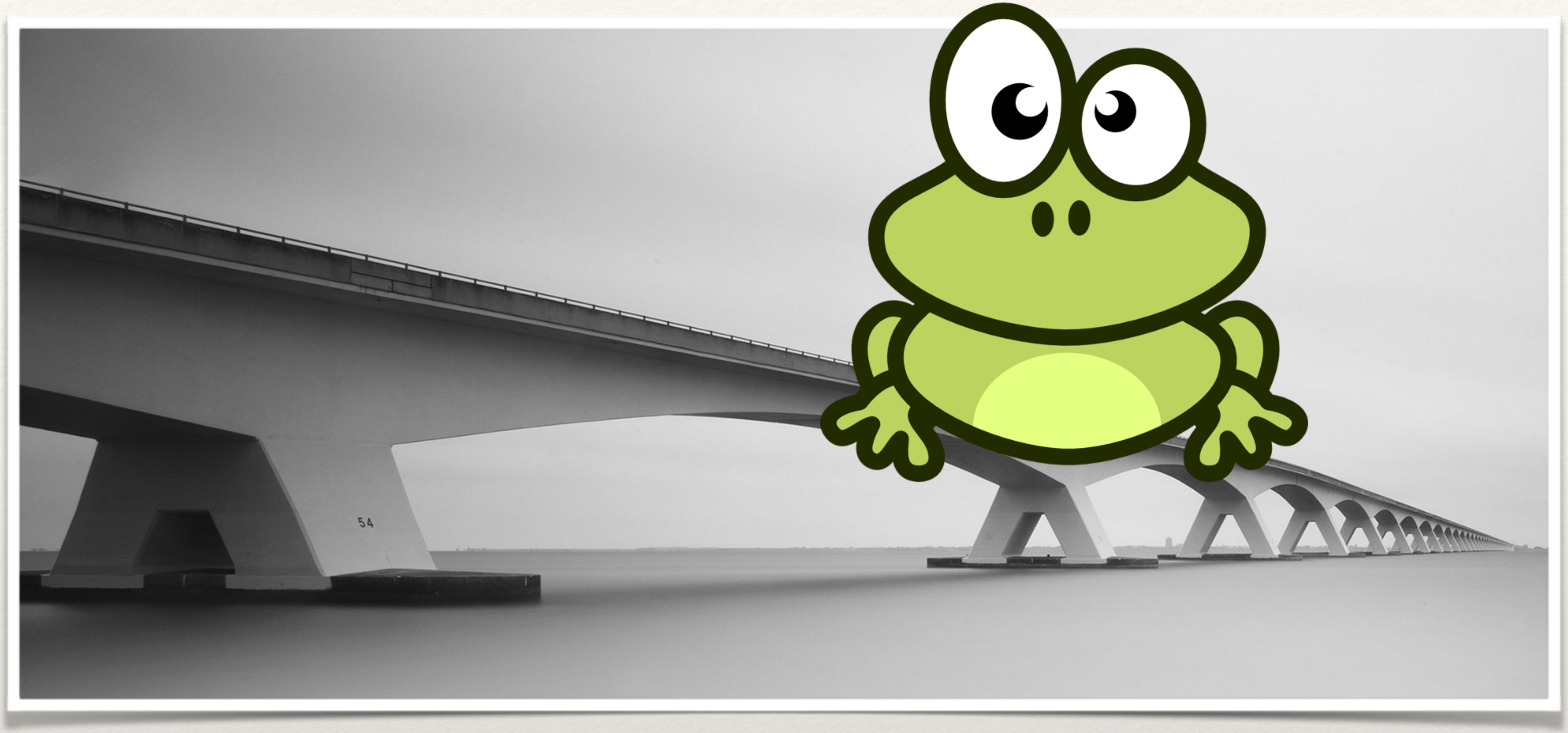


FrogWild!

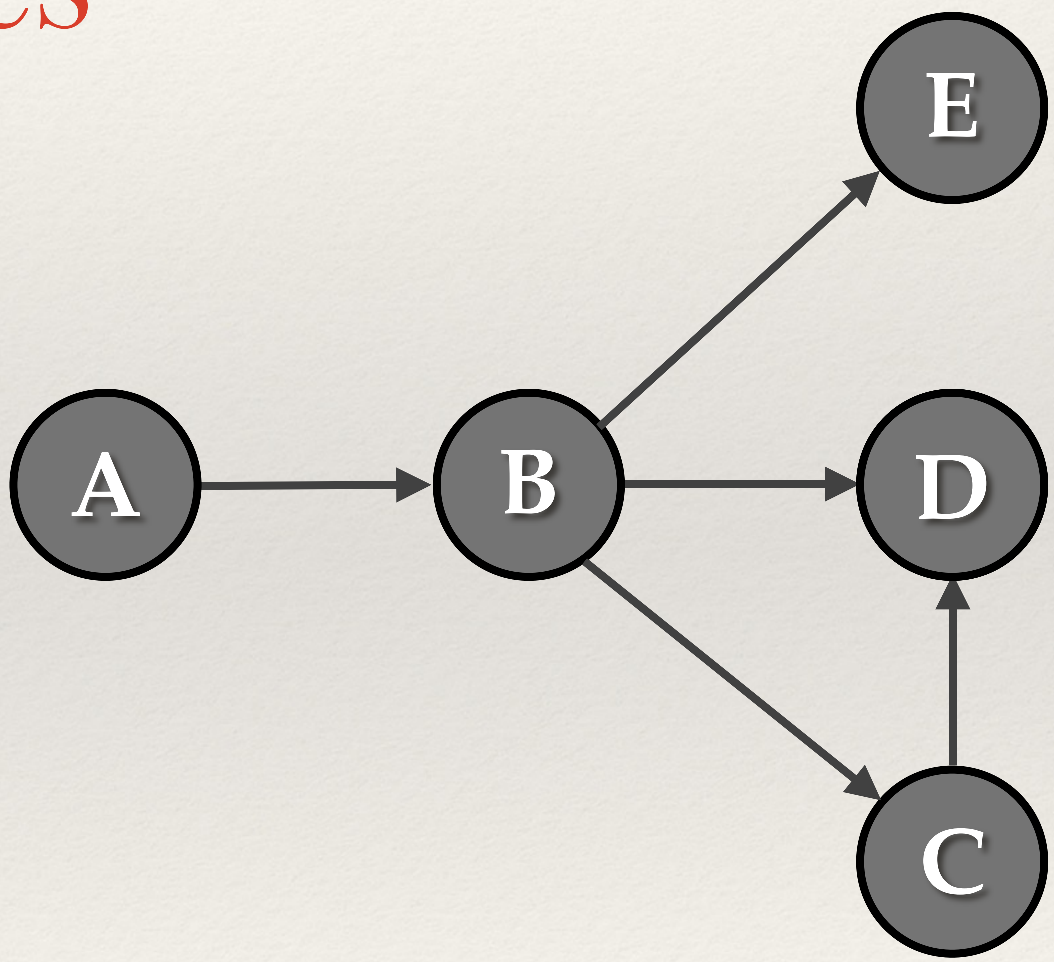
Fast PageRank Approximations on Graph Engines

Ioannis Mitliagkas
Michael Borokhovich
Alex Dimakis
Constantine Caramanis

Web Ranking

Given web graph

Find "important" pages



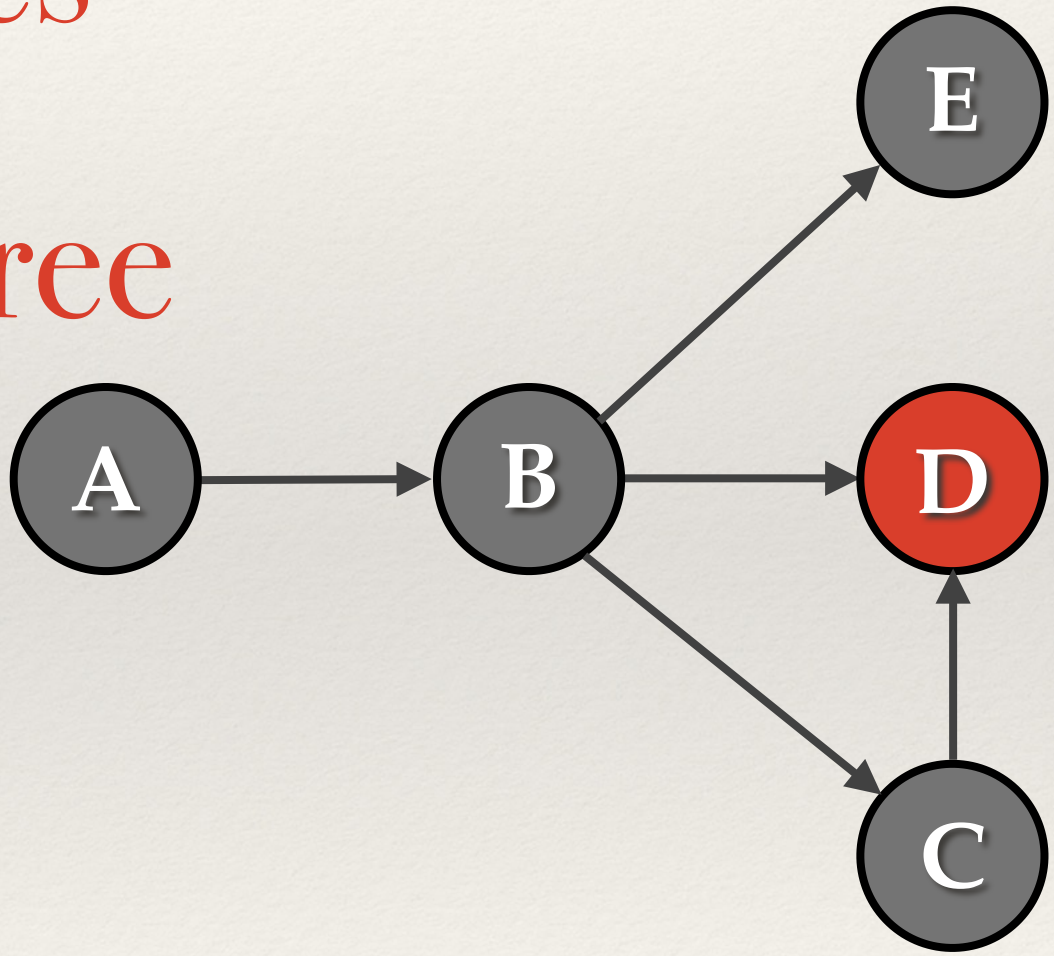
Web Ranking

Given web graph

Find "important" pages

Rank Based on In-degree

Classic Approach



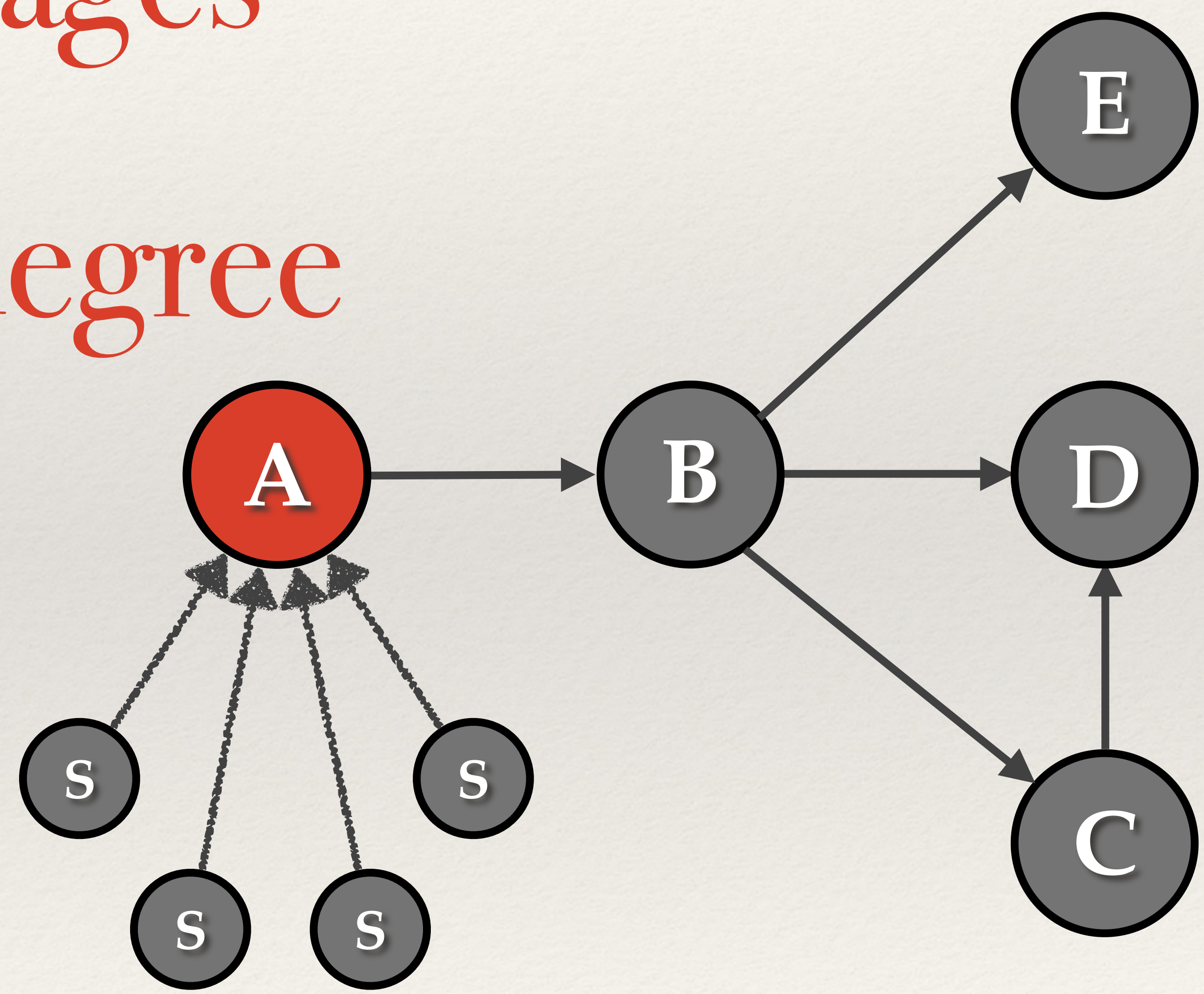
Web Ranking

Given web graph

Find "important" pages

Rank Based on In-degree

Classic Approach



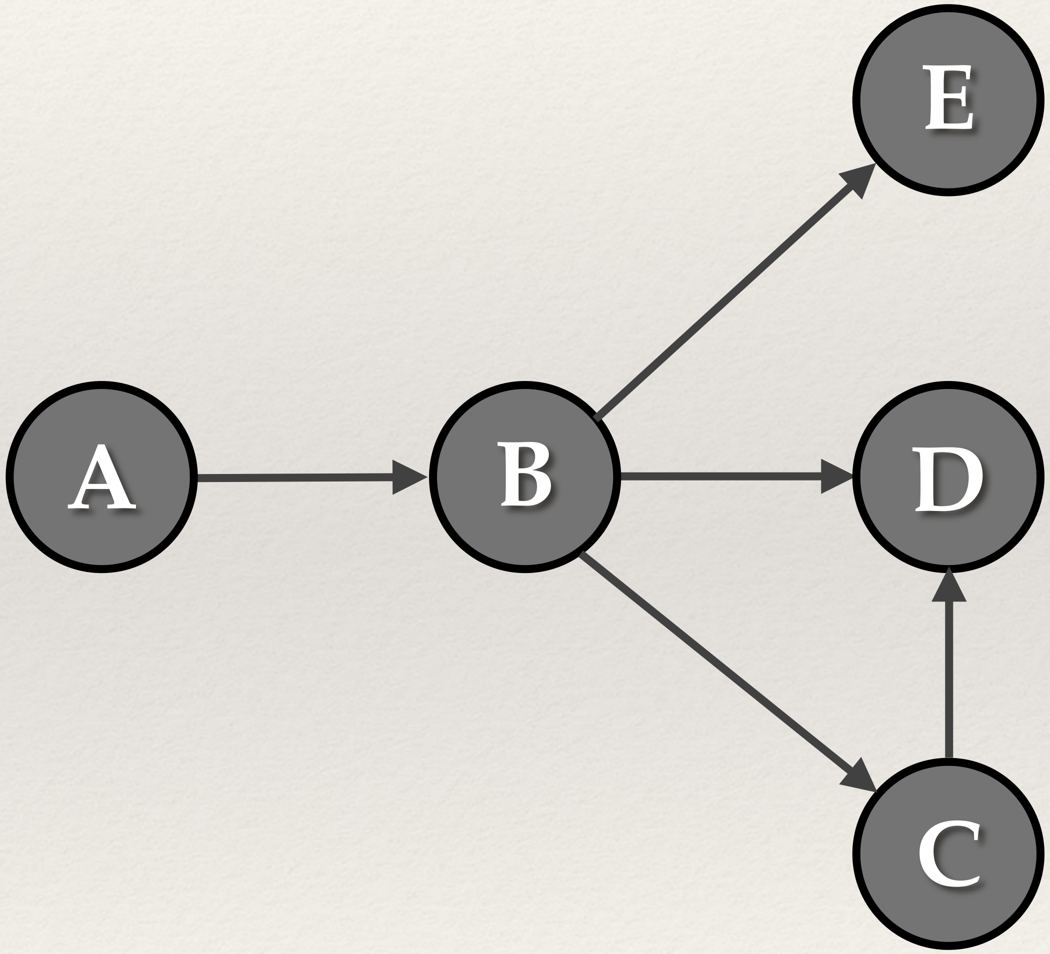
Susceptible

to manipulation by spammer networks

PageRank [Page et al., 1999]

Page Importance

Described by distribution π



PageRank [Page et al., 1999]

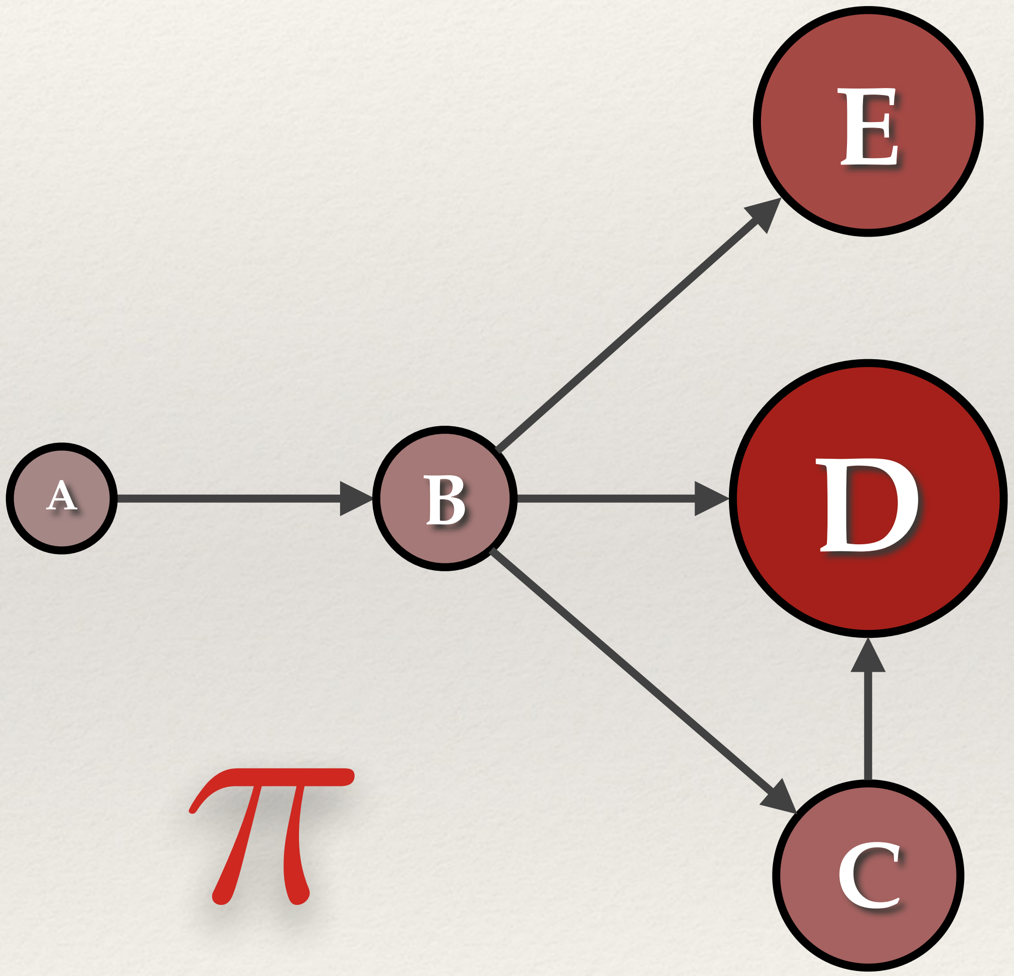
Page Importance

Described by distribution π

Recursive Definition

Important pages are pointed to by

- important pages are pointed to by
  - important pages are pointed to by…



π

PageRank [Page et al., 1999]

Page Importance

Described by distribution π

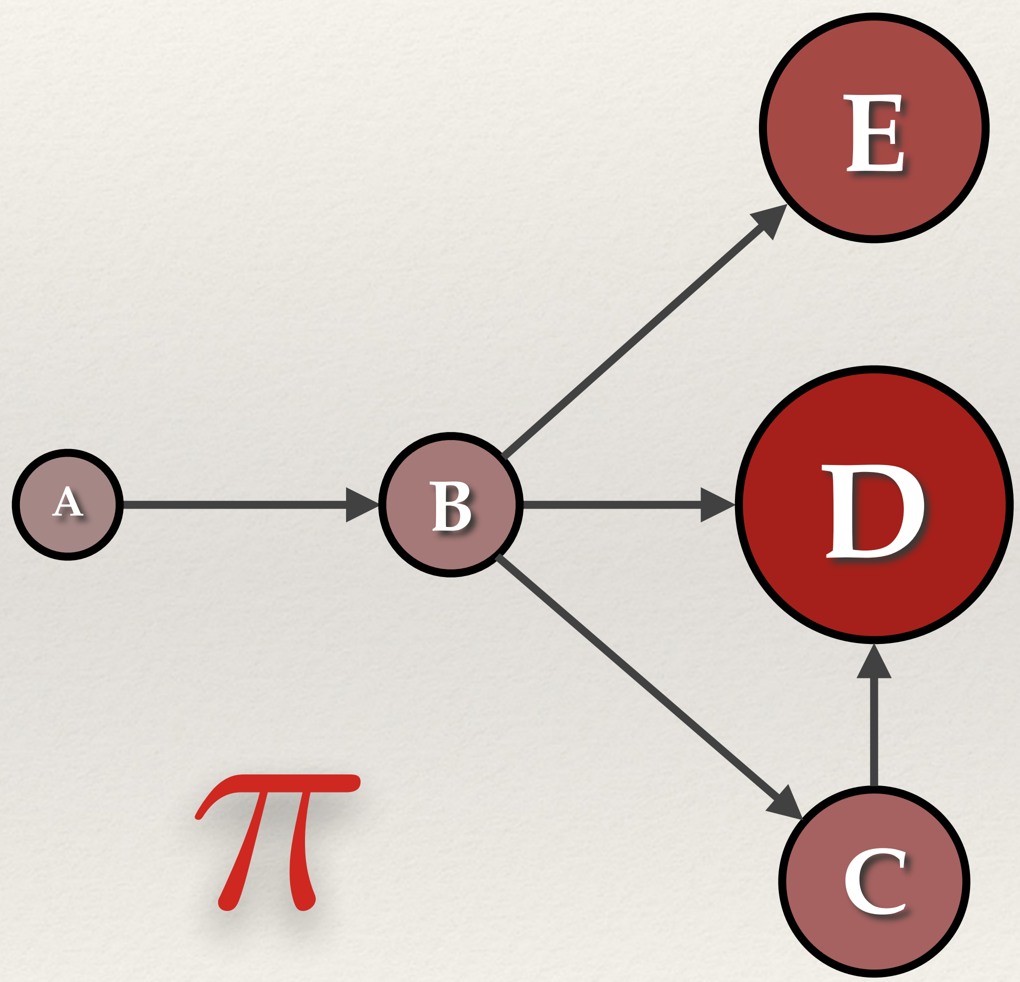
Recursive Definition

Important pages are pointed to by

- important pages are pointed to by
  - important pages are pointed to by…

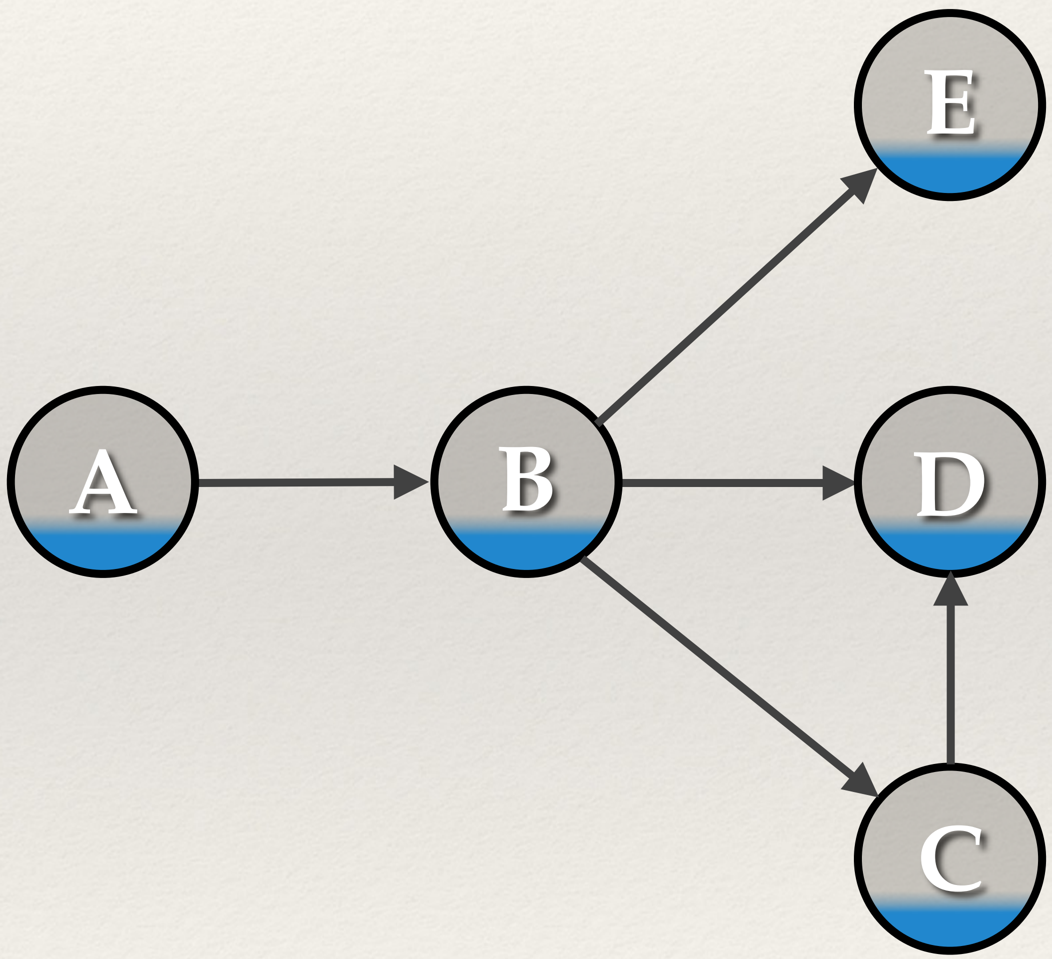
Robust

to manipulation by spammer networks



PageRank - Continuous Interpretation

Start: Gallon of water distributed evenly

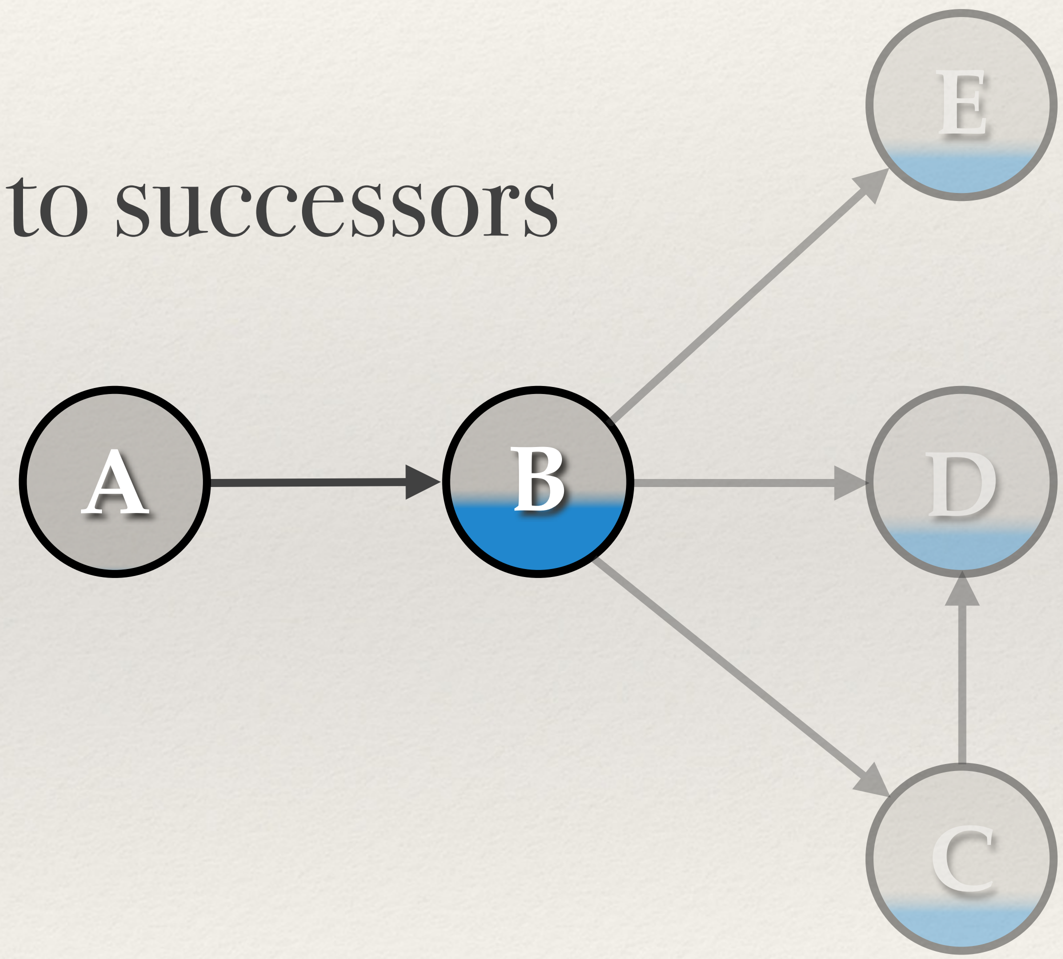


PageRank - Continuous Interpretation

Start: Gallon of water distributed evenly

Every Iteration

Each vertex spreads water evenly to successors

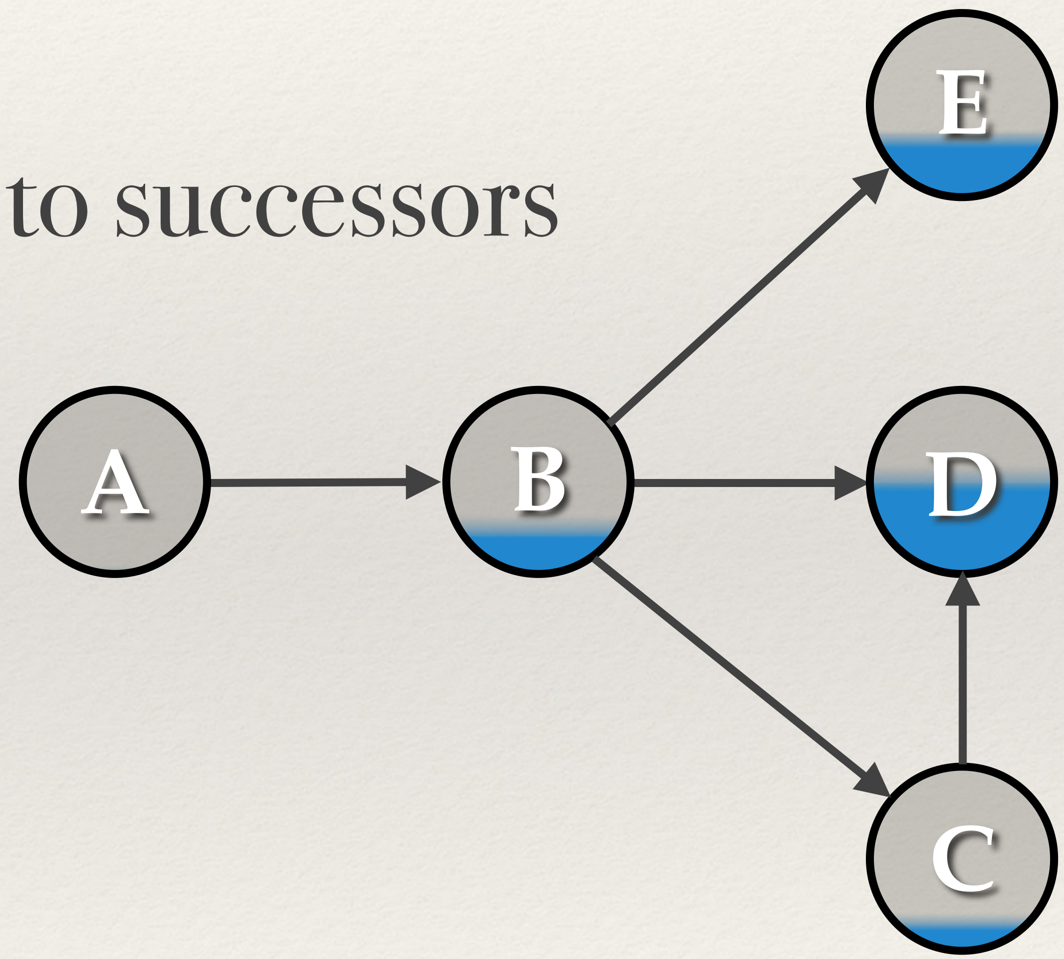


PageRank - Continuous Interpretation

Start: Gallon of water distributed evenly

Every Iteration

Each vertex spreads water evenly to successors



PageRank - Continuous Interpretation

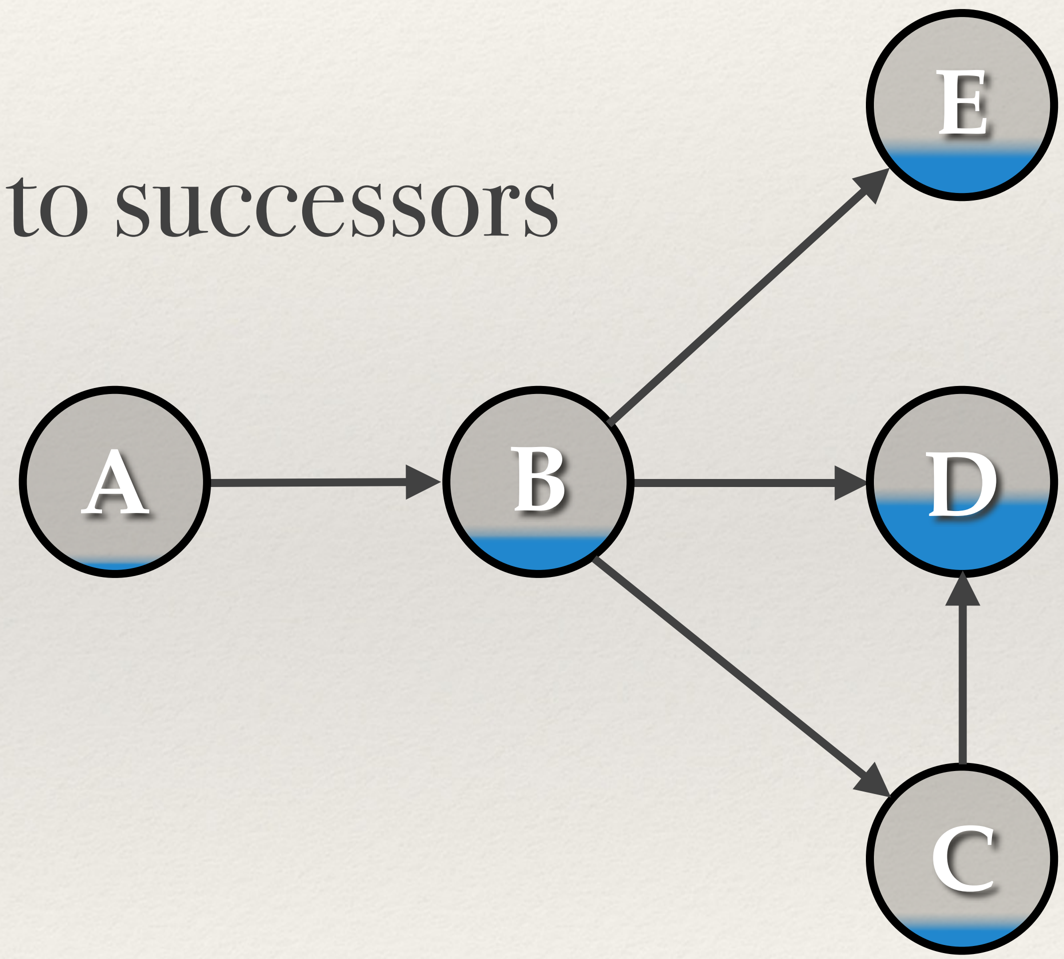
Start: Gallon of **water** distributed evenly

Every Iteration

Each vertex spreads water evenly to successors

Redistribute evenly

a fraction, $p_T = 0.15$, of all water



PageRank - Continuous Interpretation

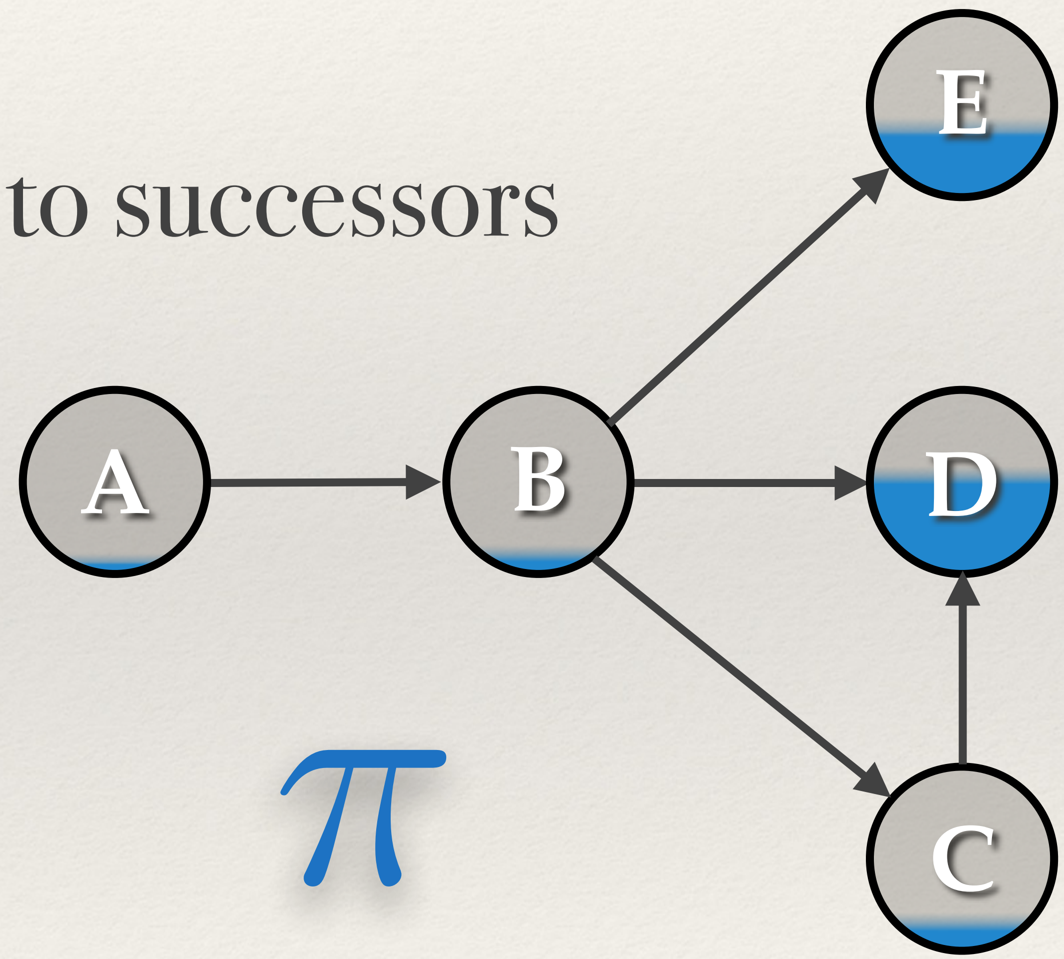
Start: Gallon of water distributed evenly

Every Iteration

Each vertex spreads water evenly to successors

Redistribute evenly

a fraction, $p_T = 0.15$, of all water



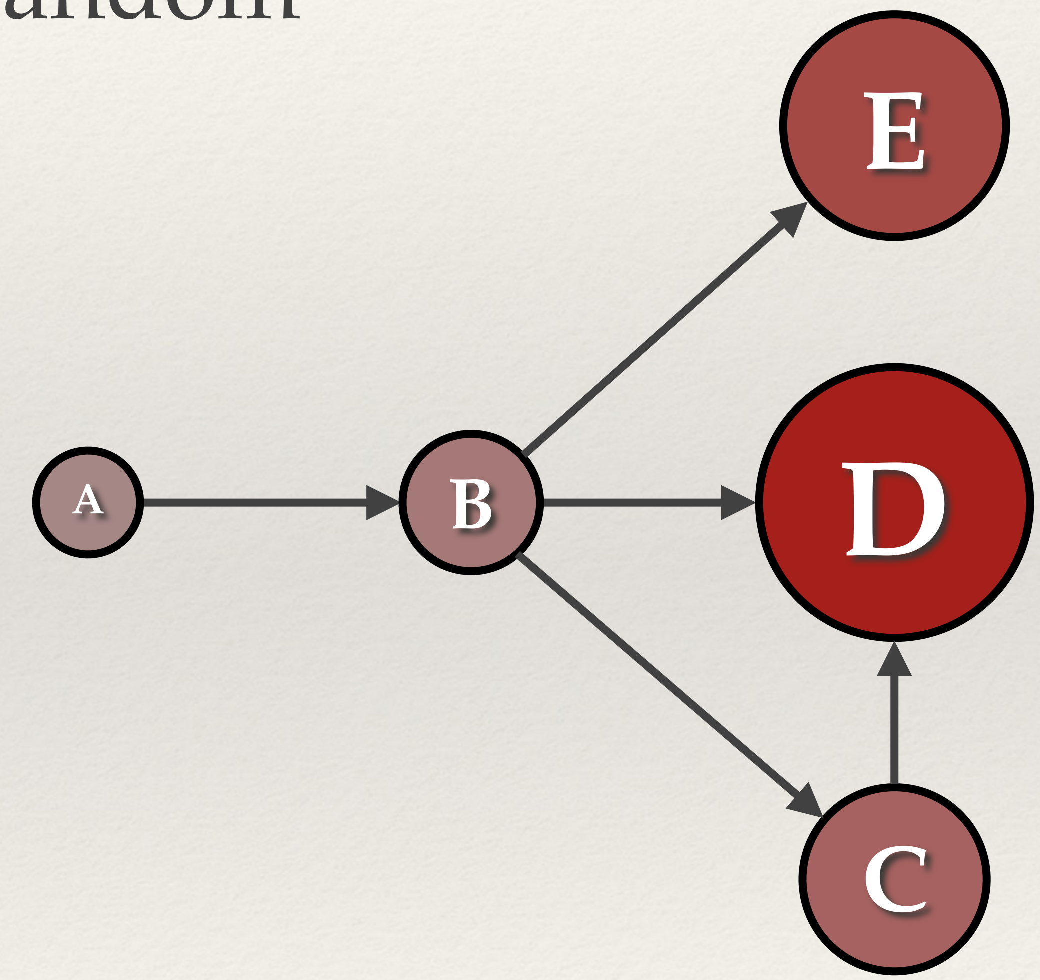
Repeat until convergence

Power Iteration employed usually

Discrete Interpretation

Frog walks randomly on graph

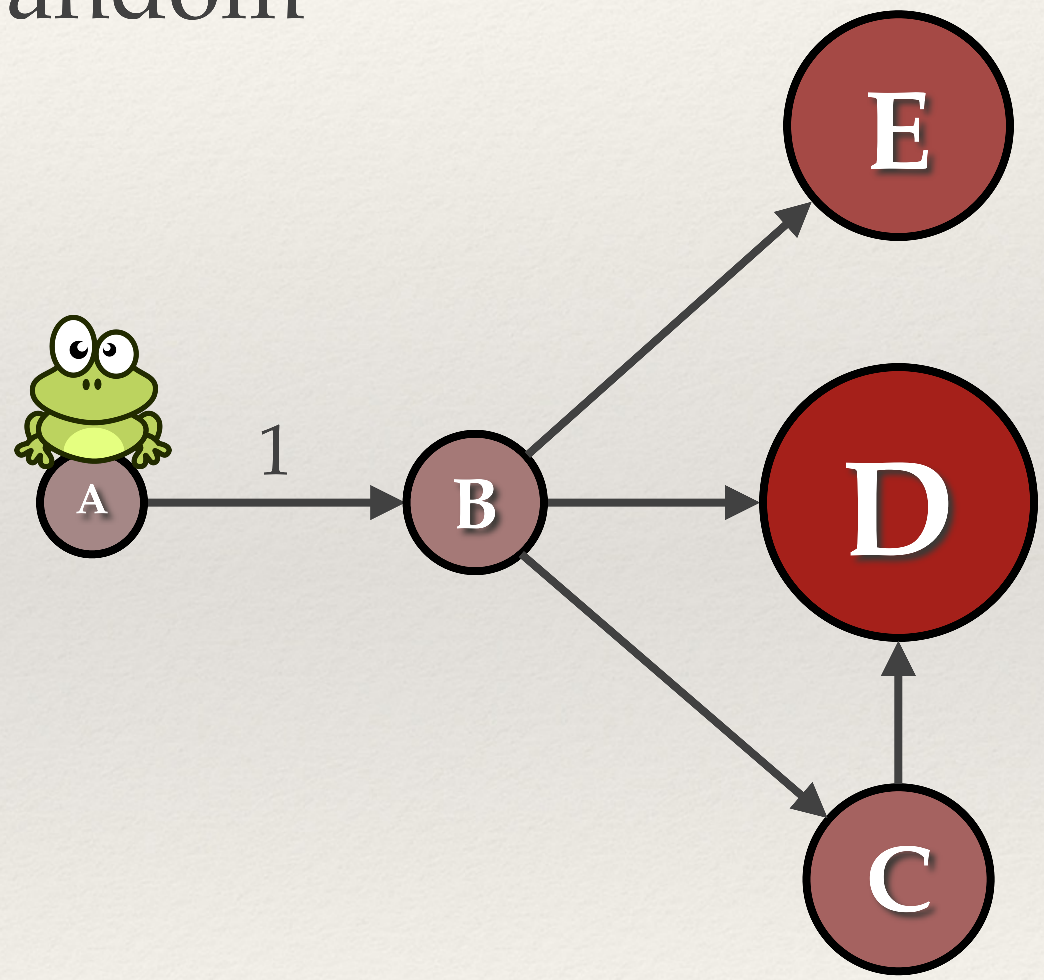
Next vertex chosen uniformly at random



Discrete Interpretation

Frog walks randomly on graph

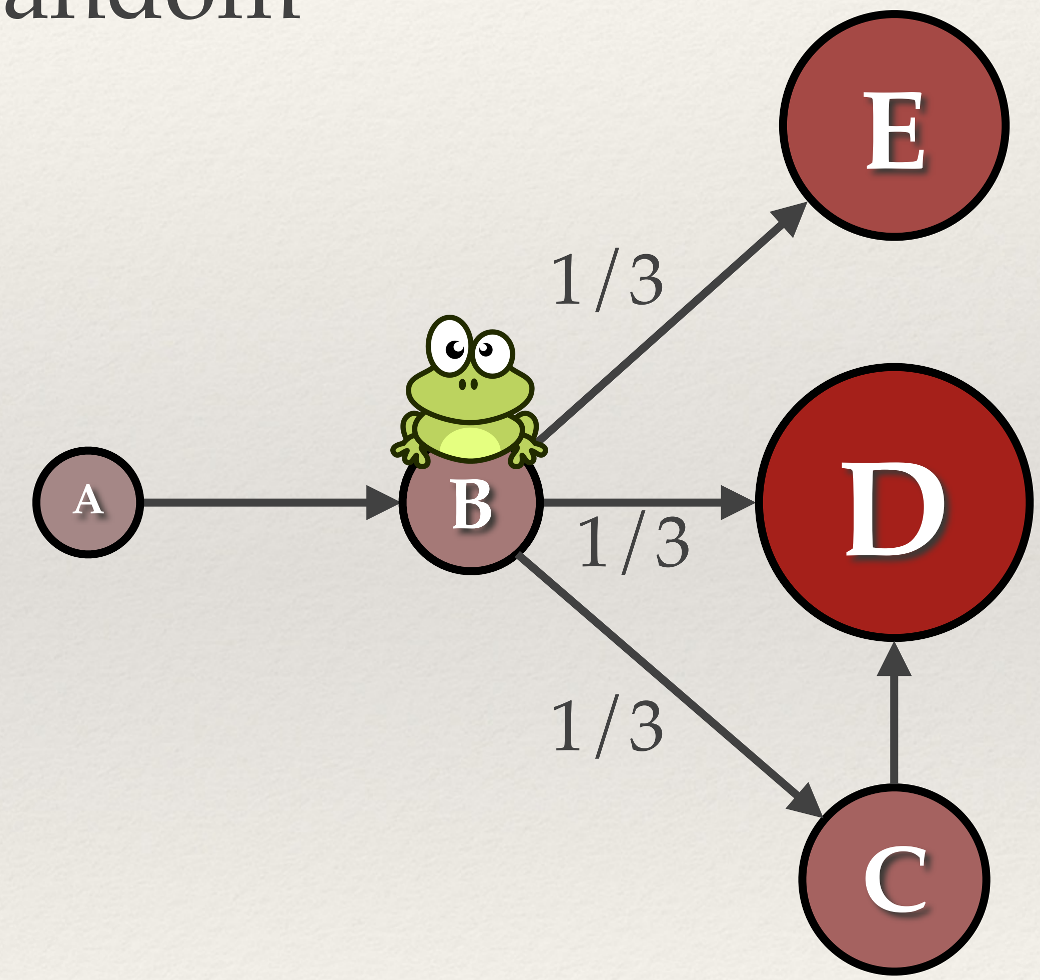
Next vertex chosen uniformly at random



Discrete Interpretation

Frog walks randomly on graph

Next vertex chosen uniformly at random



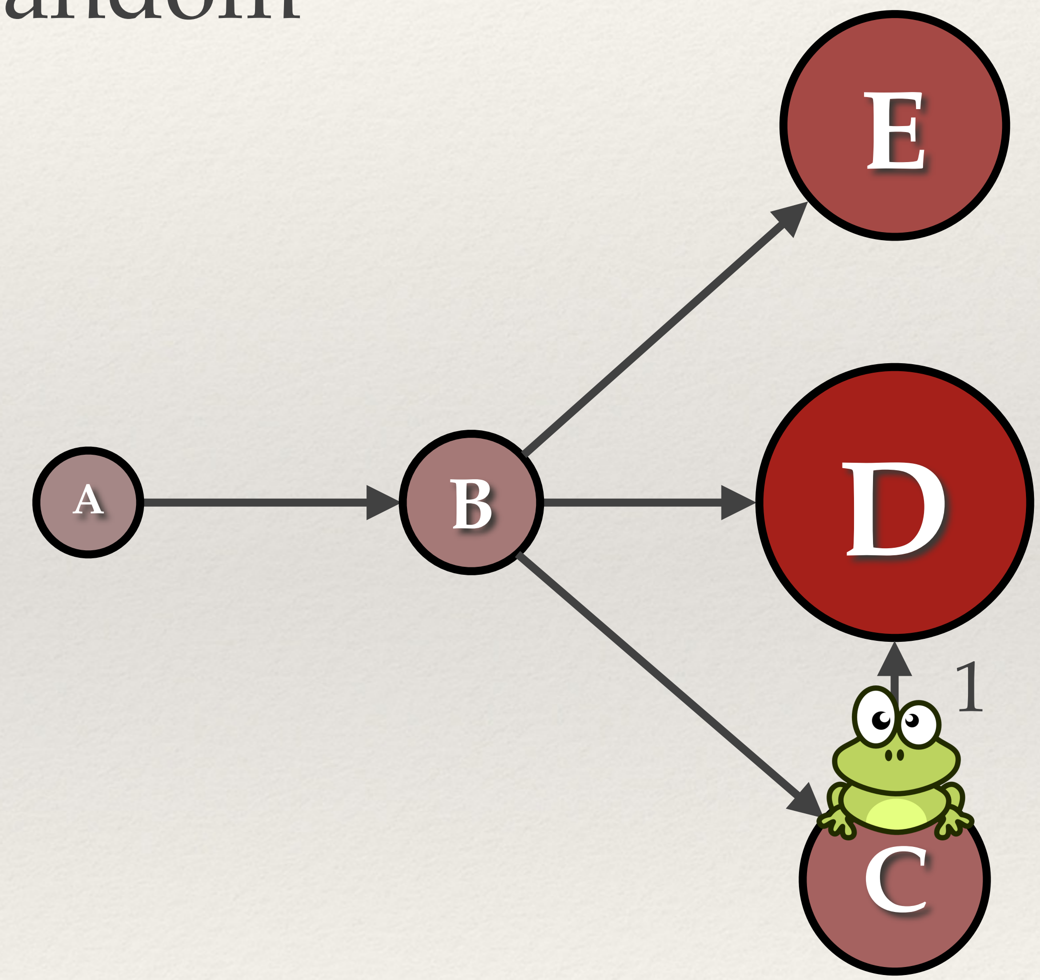
Discrete Interpretation

Frog walks randomly on graph

Next vertex chosen uniformly at random

Teleportation

Every step: teleport w.p. p_T



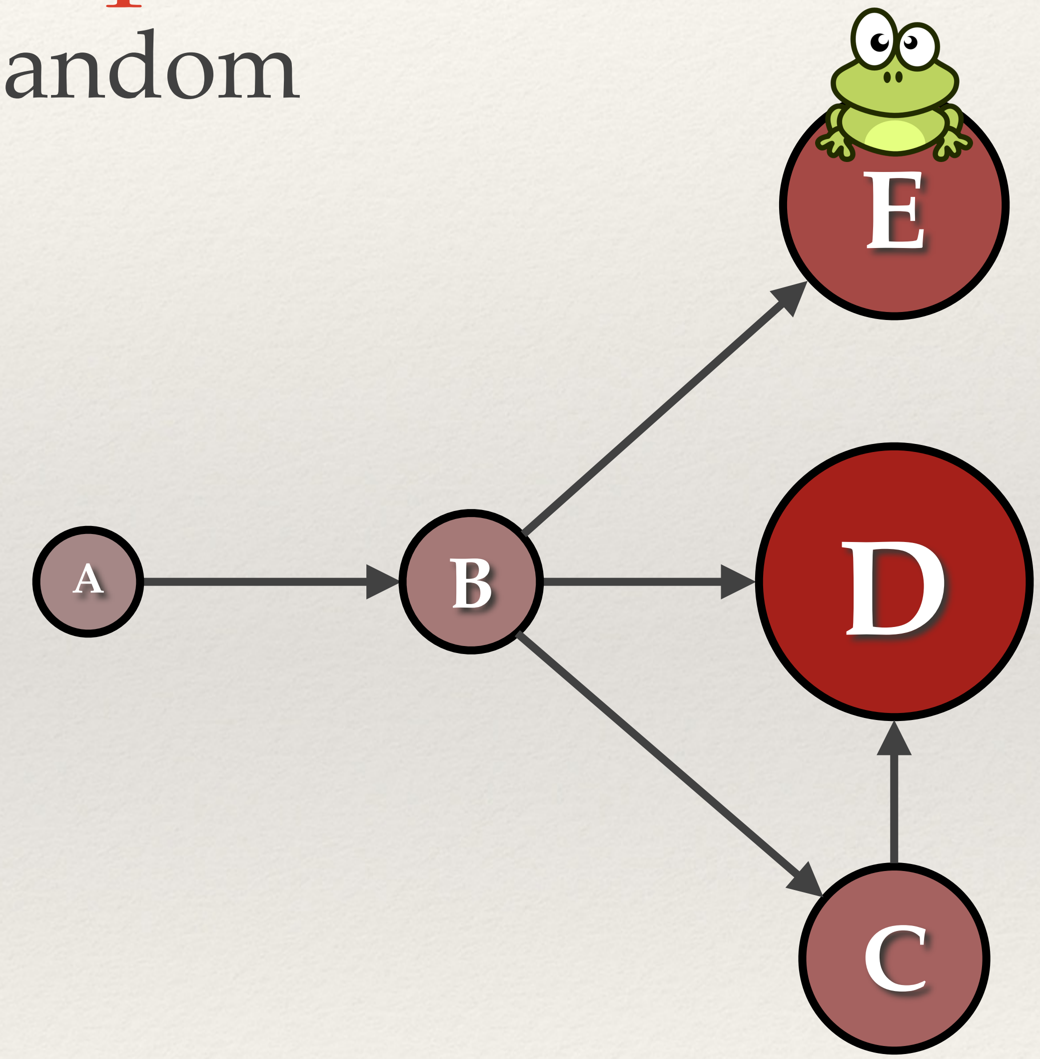
Discrete Interpretation

Frog walks randomly on graph

Next vertex chosen uniformly at random

Teleportation

Every step: teleport w.p. p_T



Discrete Interpretation

Frog walks randomly on graph

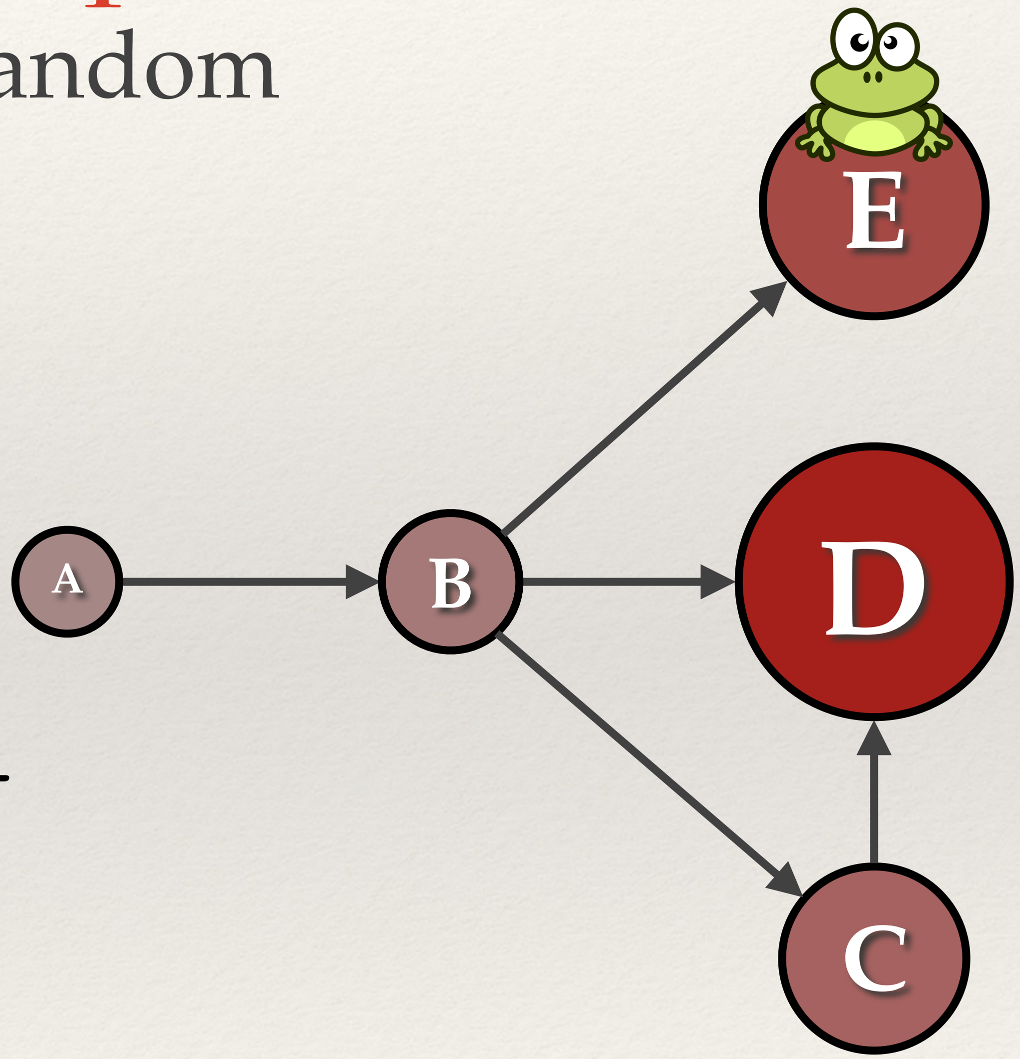
Next vertex chosen uniformly at random

Teleportation

Every step: teleport w.p. p_T

Sampling after t steps

Frog location gives sample from π



Discrete Interpretation

Frog walks randomly on graph

Next vertex chosen uniformly at random

Teleportation

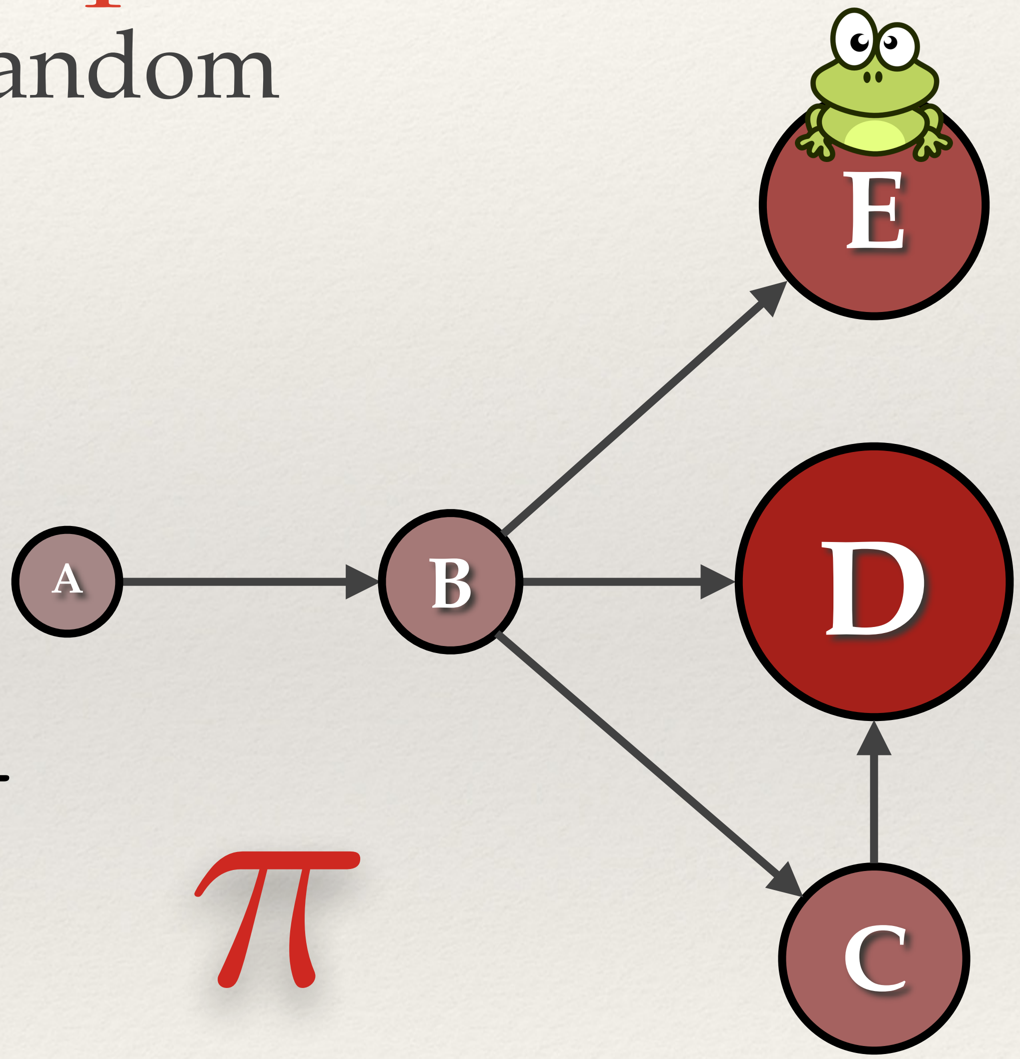
Every step: teleport w.p. p_T

Sampling after t steps

Frog location gives sample from π

PageRank Vector

Many frogs, estimate vector π



PageRank Approximation

Looking for k "heavy nodes"

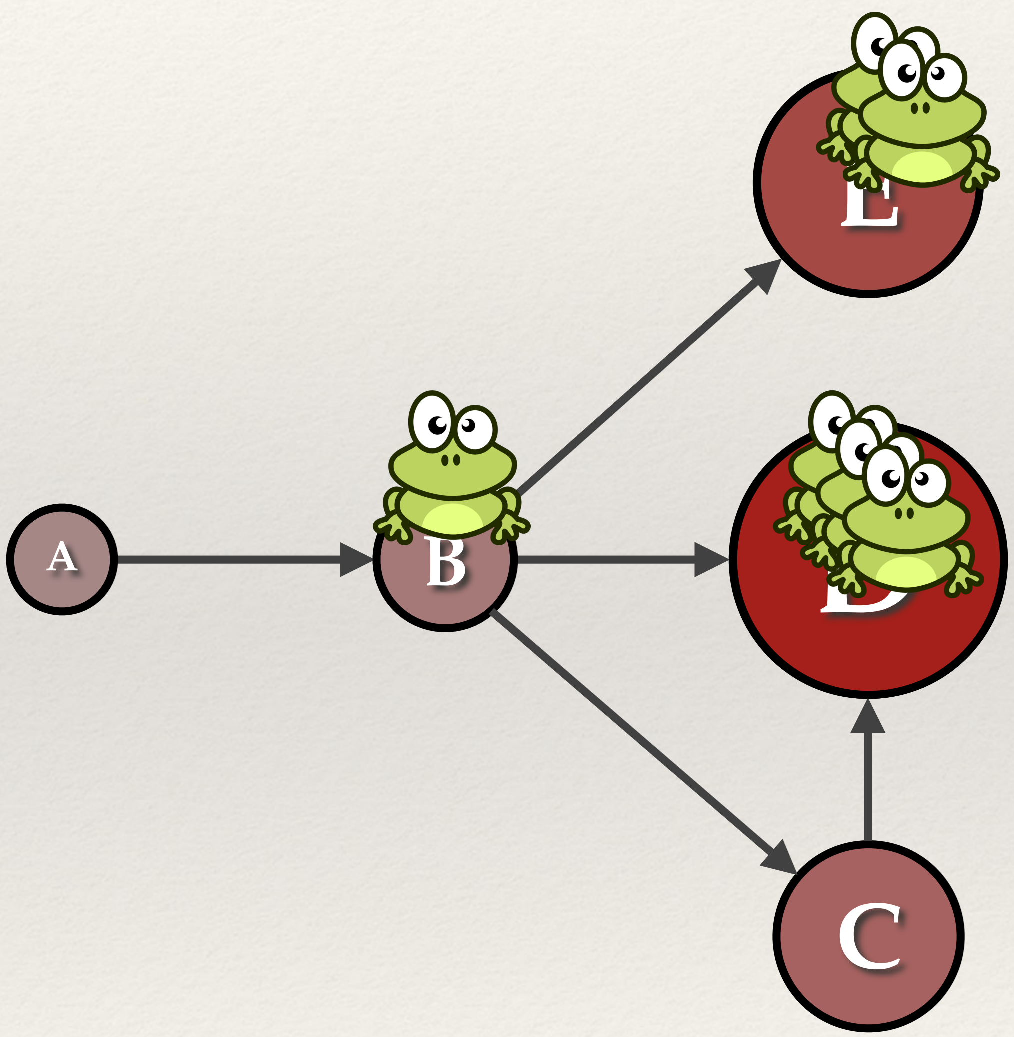
Do not need full PageRank vector

Random Walk Sampling

Favors heavy nodes

Captured Mass Metric

For node set S: $\pi(S)$



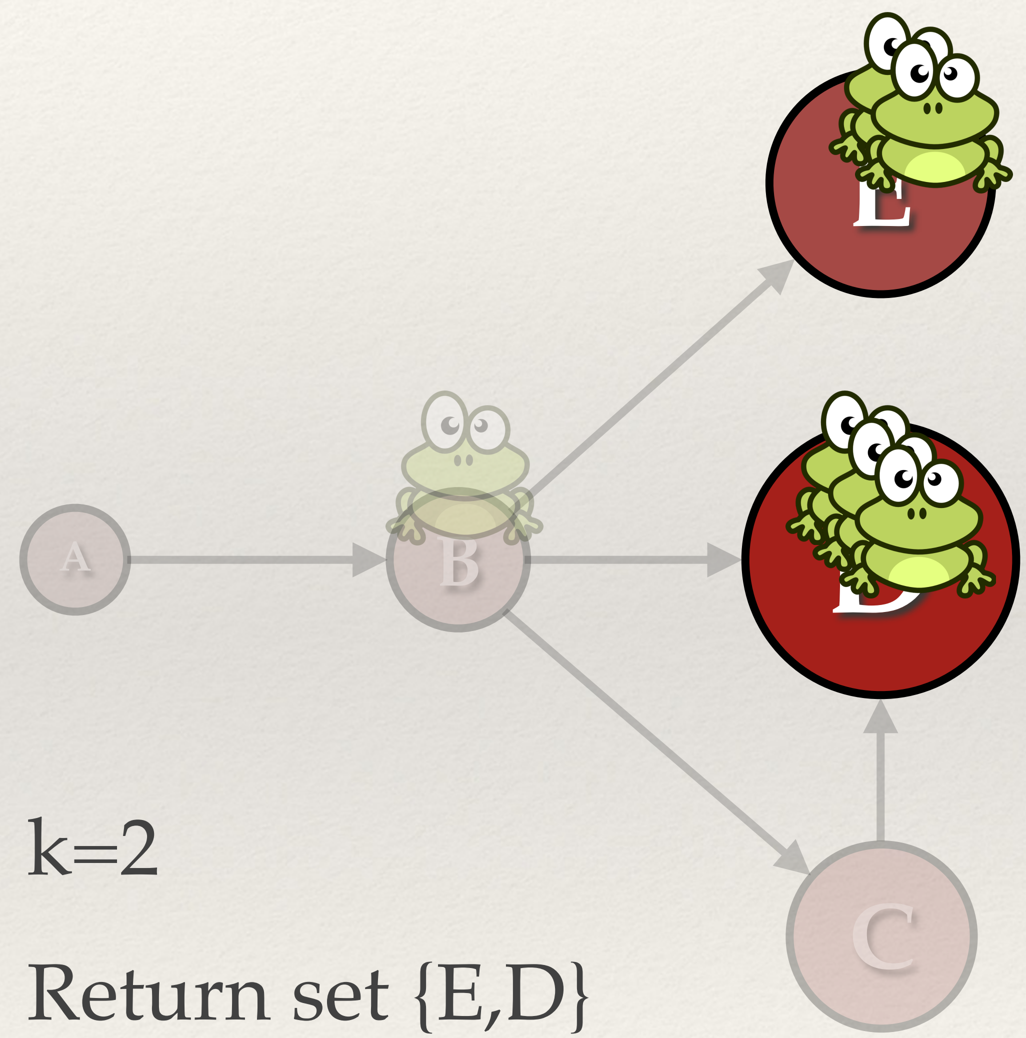
PageRank Approximation

Looking for k "heavy nodes"

Do not need full PageRank vector

Random Walk Sampling

Favors heavy nodes



Captured Mass Metric

For node set S: $\pi(S)$

k=2

Return set {E,D}

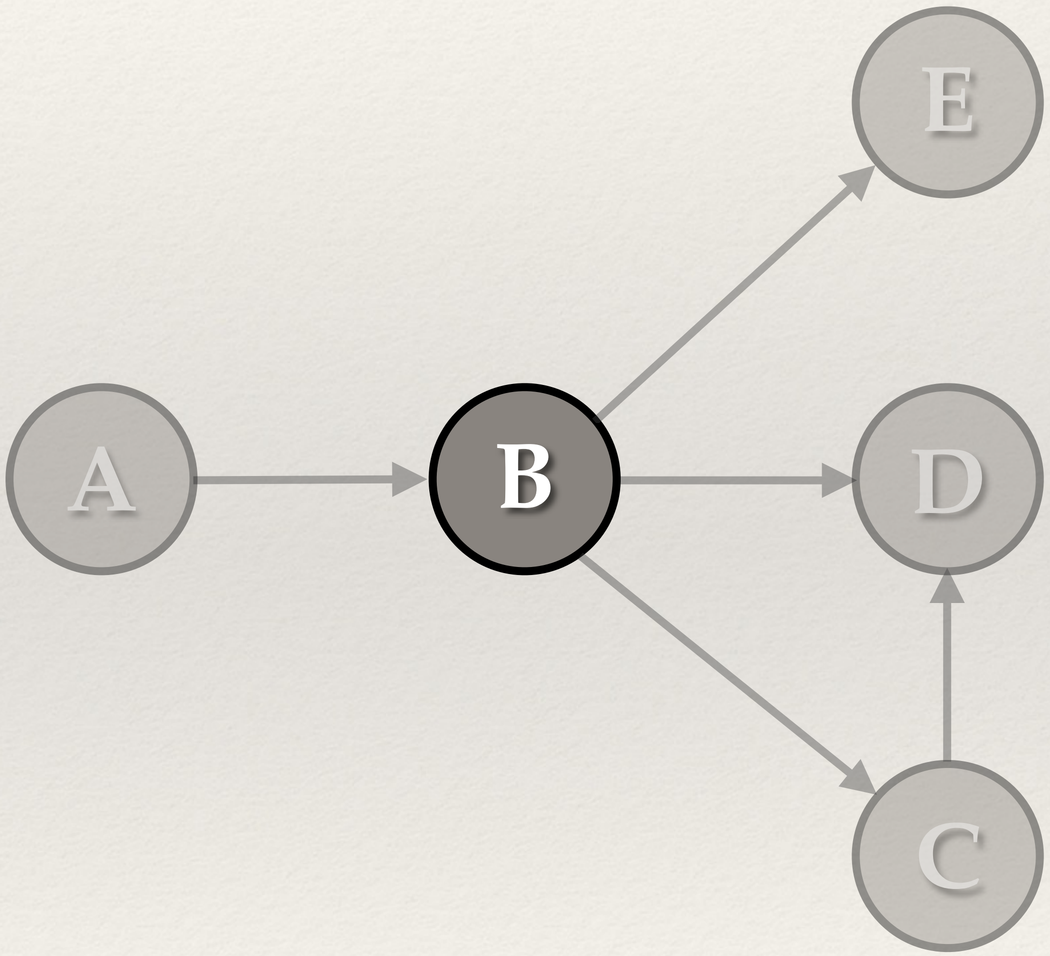
Captured mass $= \pi(\{E,D\})$

Platform

Graph Engines

- Engine splits graph across cluster
- **Vertex program** describes logic

GAS abstraction



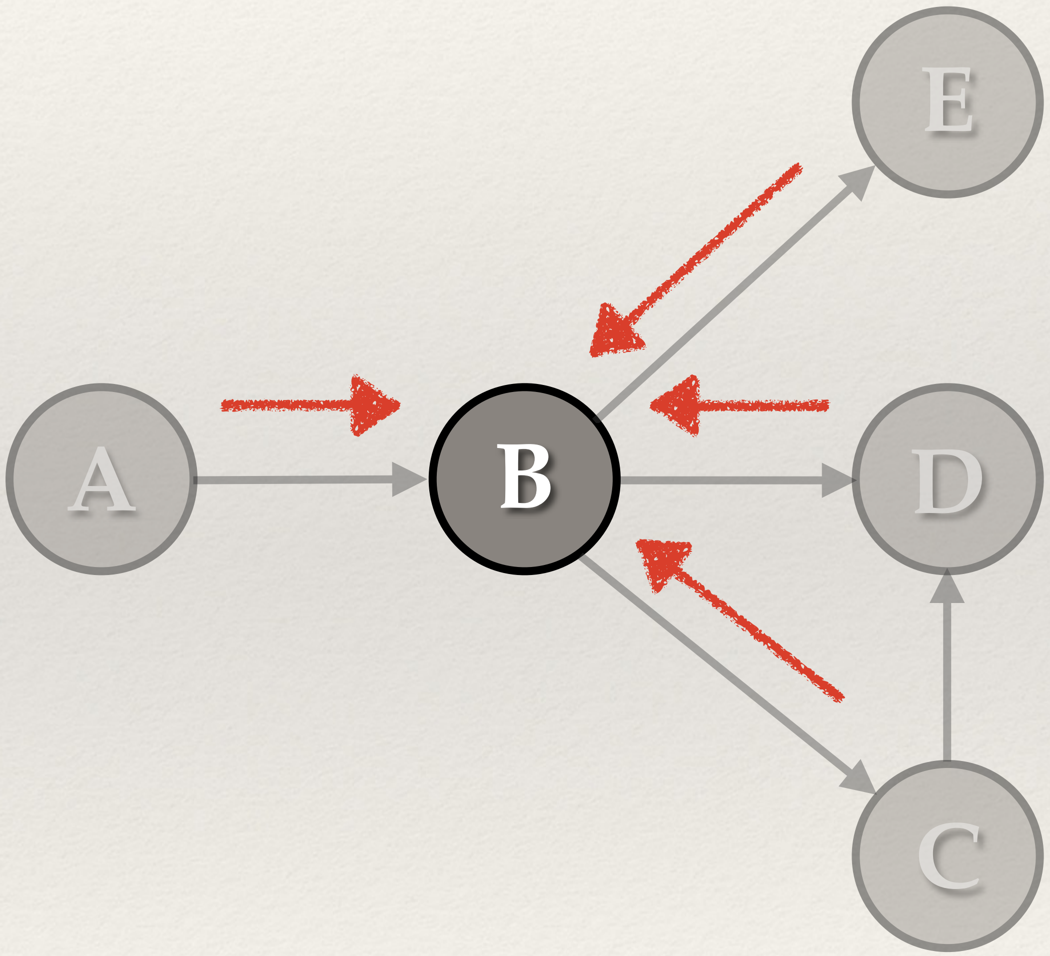
Other approaches:
Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]

Graph Engines

- Engine splits graph across cluster
- **Vertex program** describes logic

GAS abstraction

1. Gather



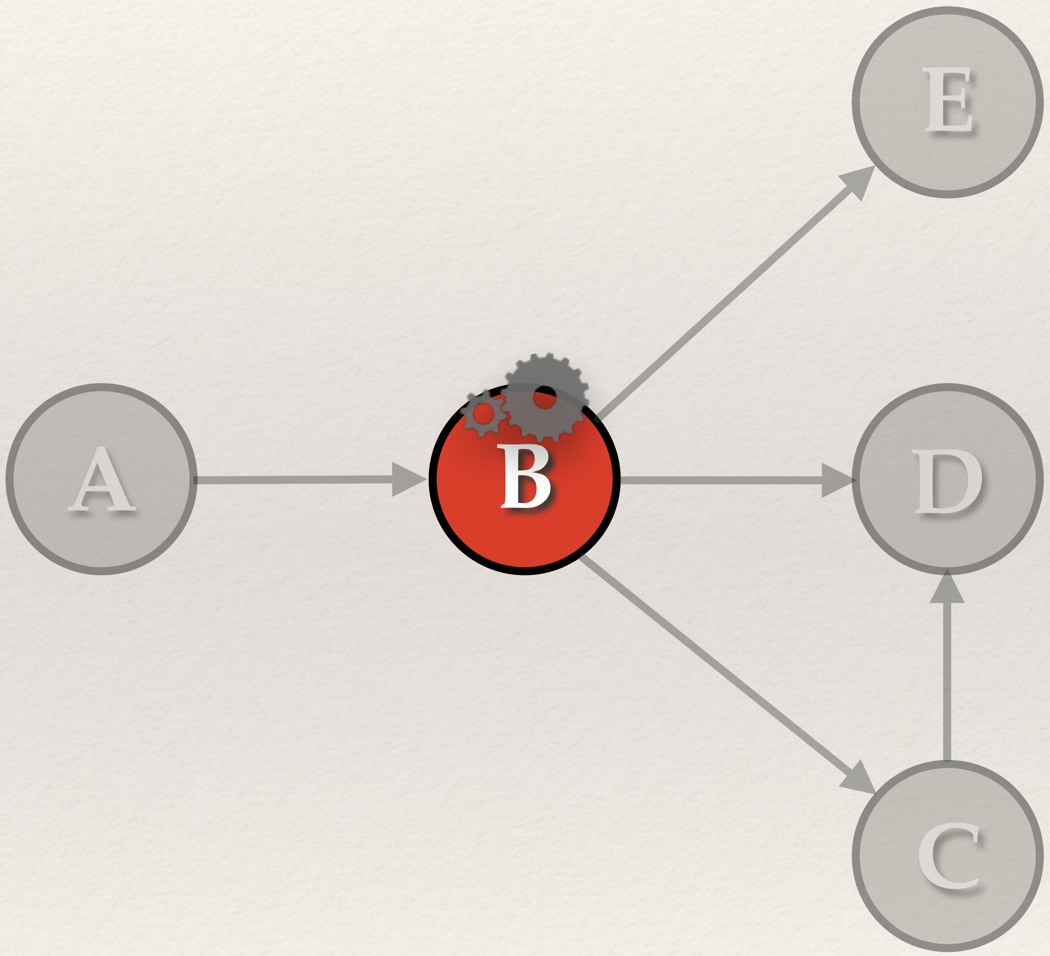
Other approaches:
Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]

Graph Engines

- Engine splits graph across cluster
- **Vertex program** describes logic

GAS abstraction

1. Gather
2. Apply



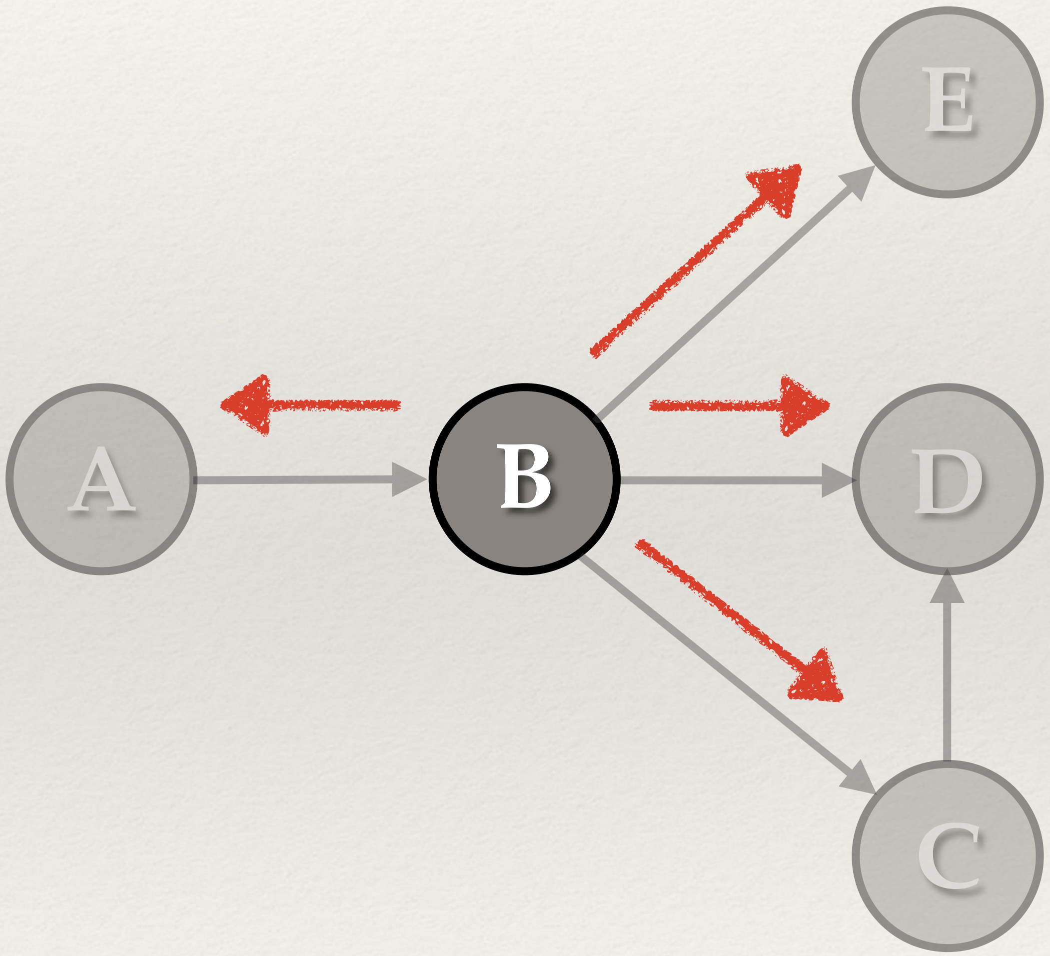
Other approaches:
Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]

Graph Engines

- Engine splits graph across cluster
- **Vertex program** describes logic

GAS abstraction

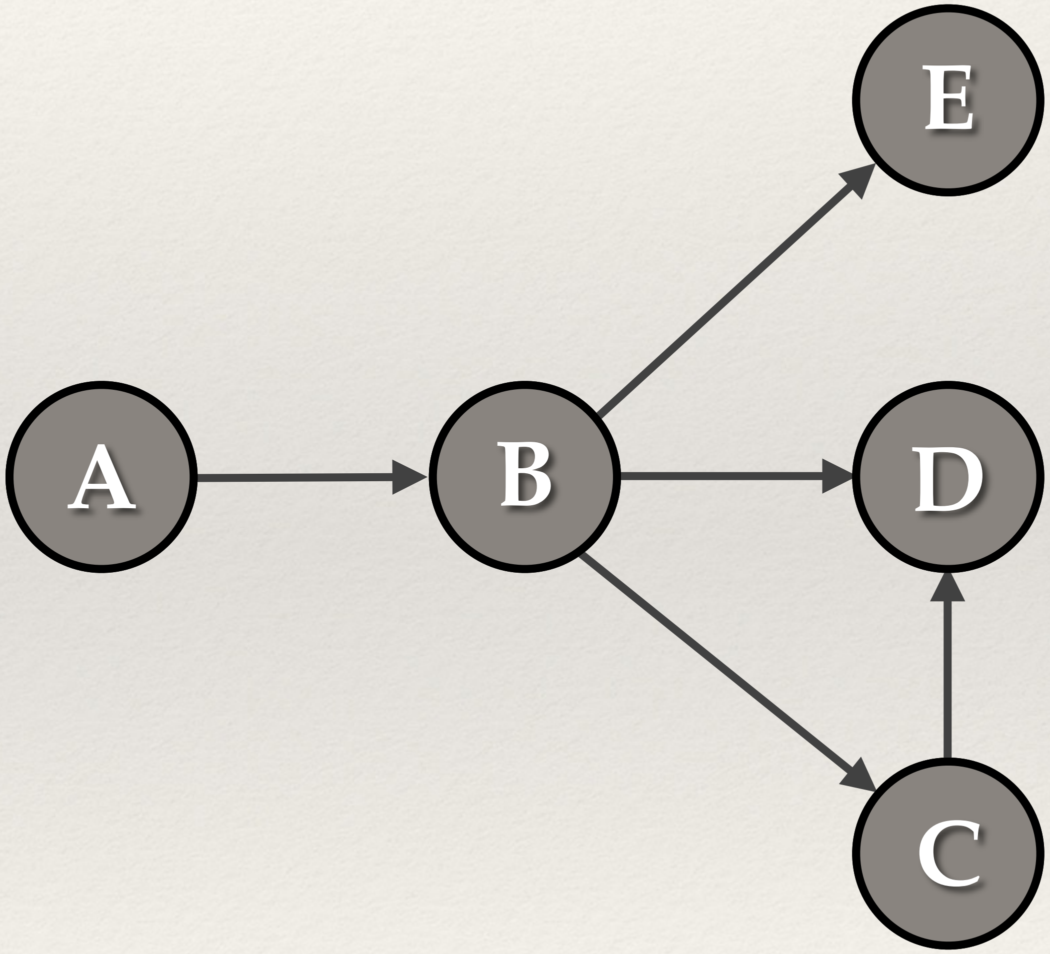
1. Gather
2. Apply
3. Scatter



Other approaches:
Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]

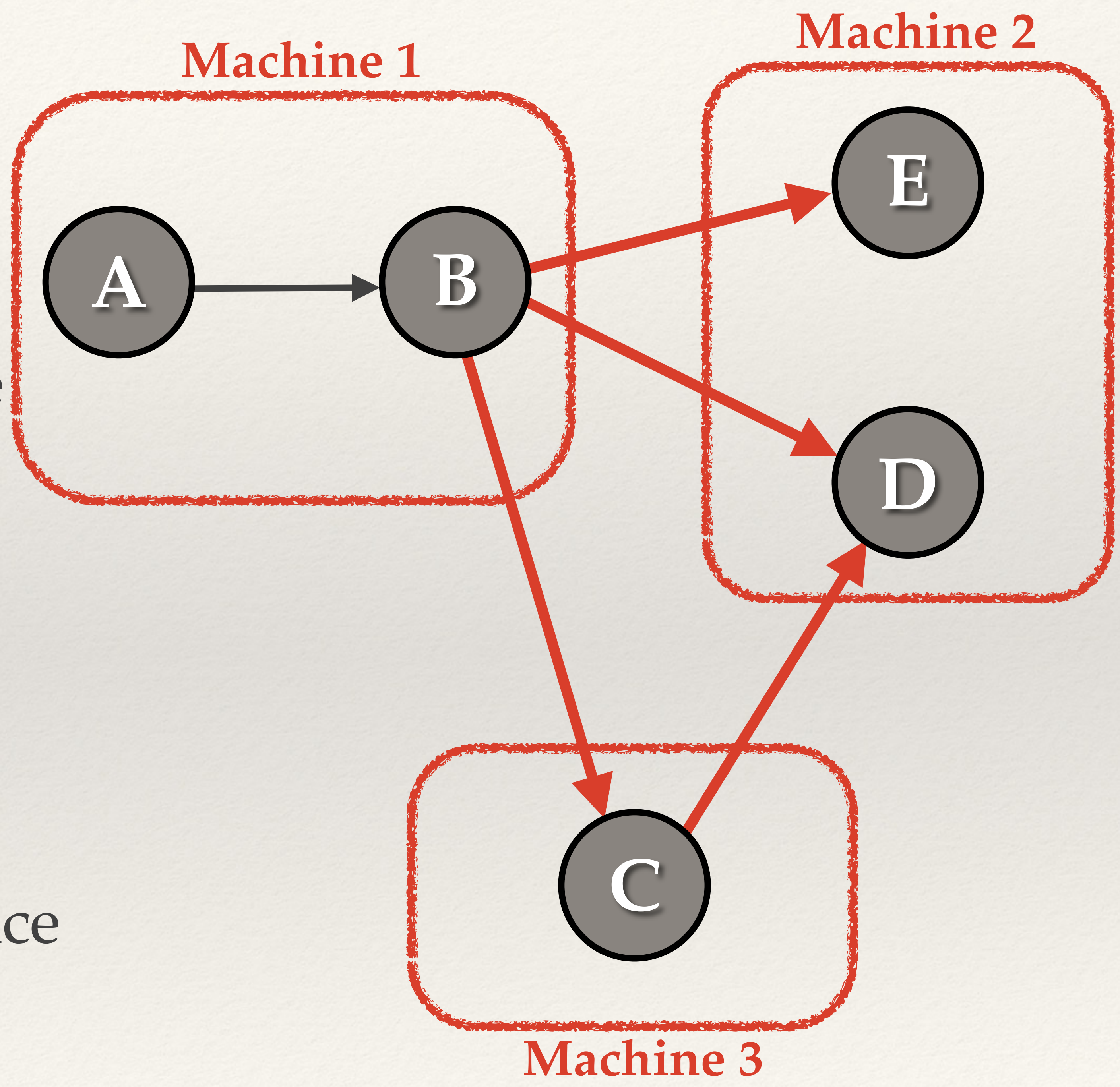
Edge Cuts

- Assign vertices to machines
- Cross-machine edges require network communication
- Pregel, GraphLab 1.0
- High-degree nodes generate large volume of traffic
- Computational load imbalance



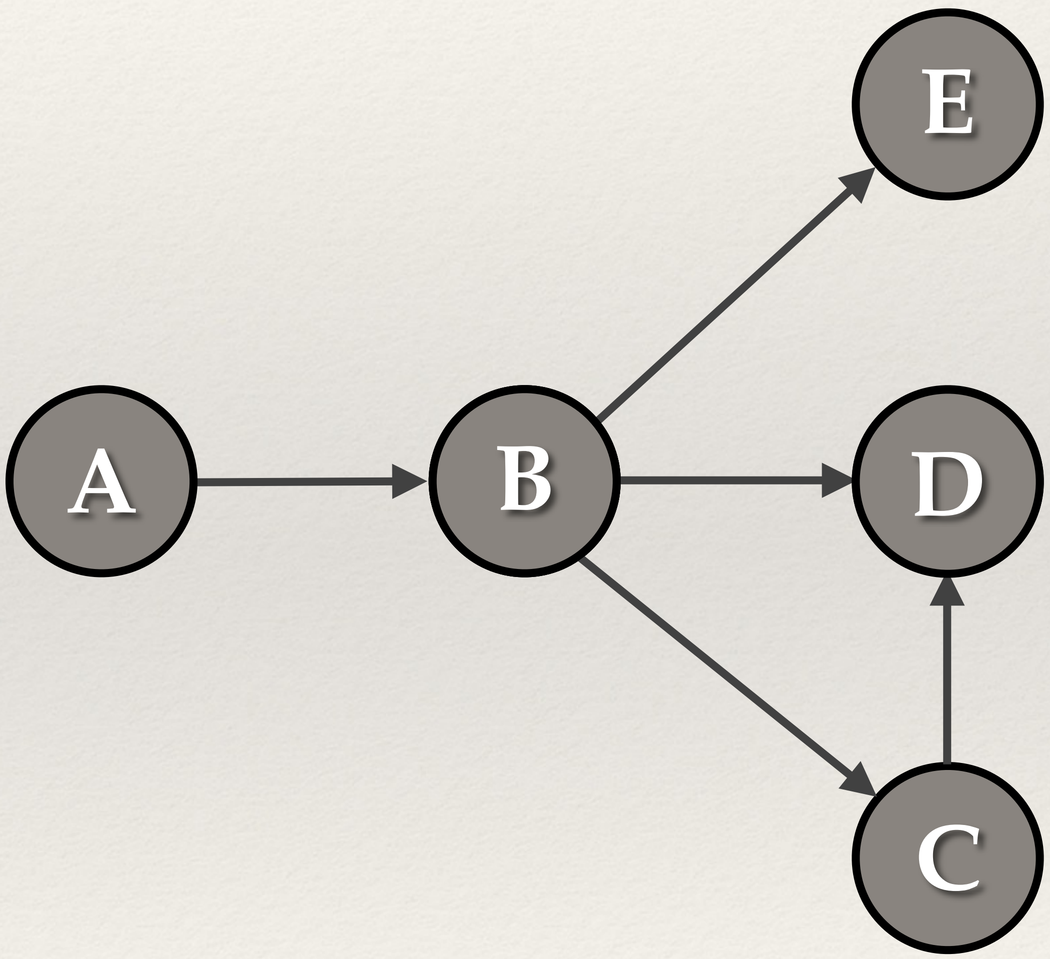
Edge Cuts

- Assign vertices to machines
- Cross-machine edges require network communication
- Pregel, GraphLab 1.0
- High-degree nodes generate large volume of traffic
- Computational load imbalance



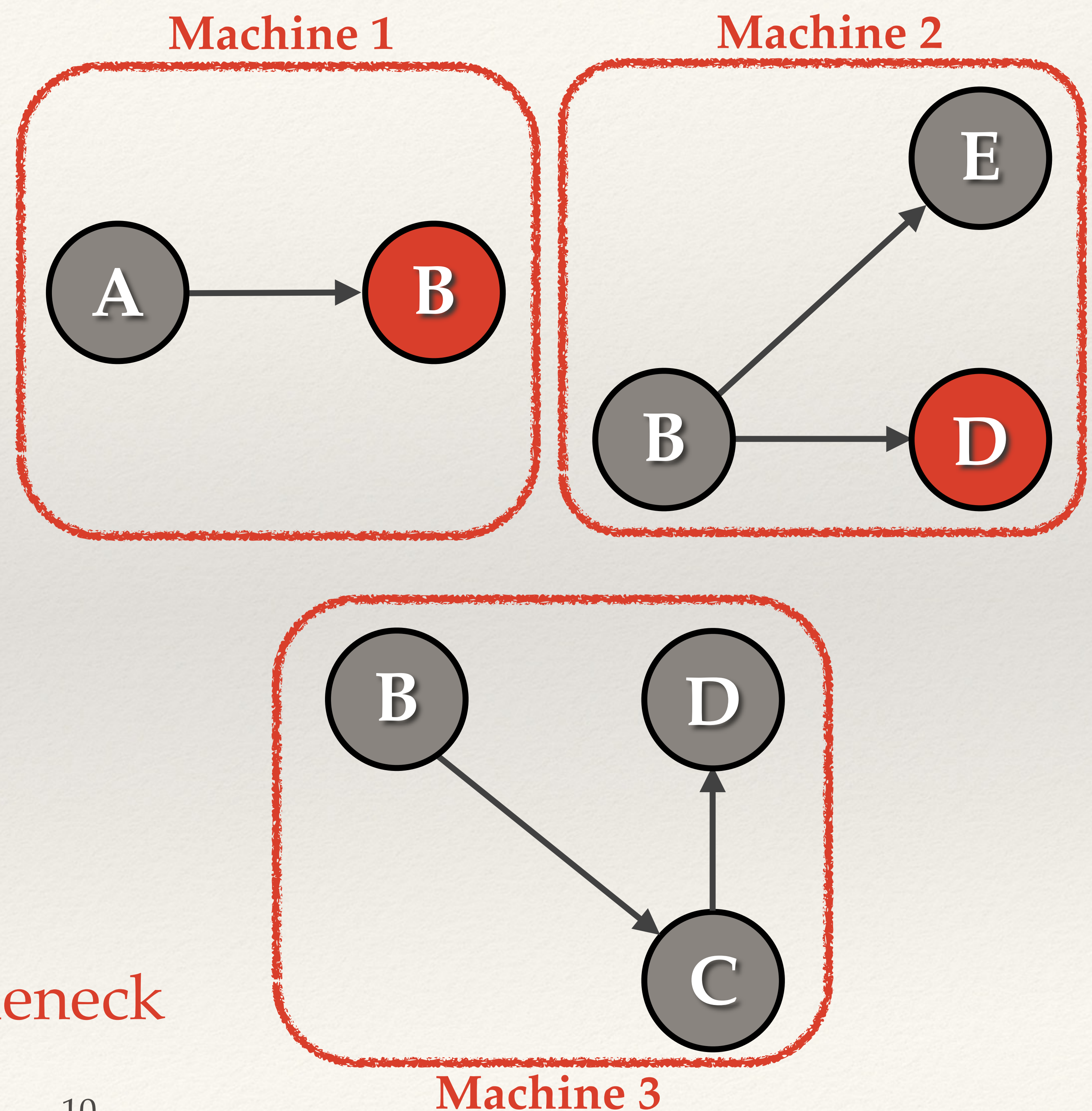
Vertex Cuts

- Assign edges to machines
- High-degree nodes replicated
- One replica designated master
- Need for synchronization
  1. Gather
  2. Apply [on master]
  3. Synchronize mirrors
  4. Scatter
- GraphLab 2.0 - PowerGraph
- Balanced - Network still bottleneck



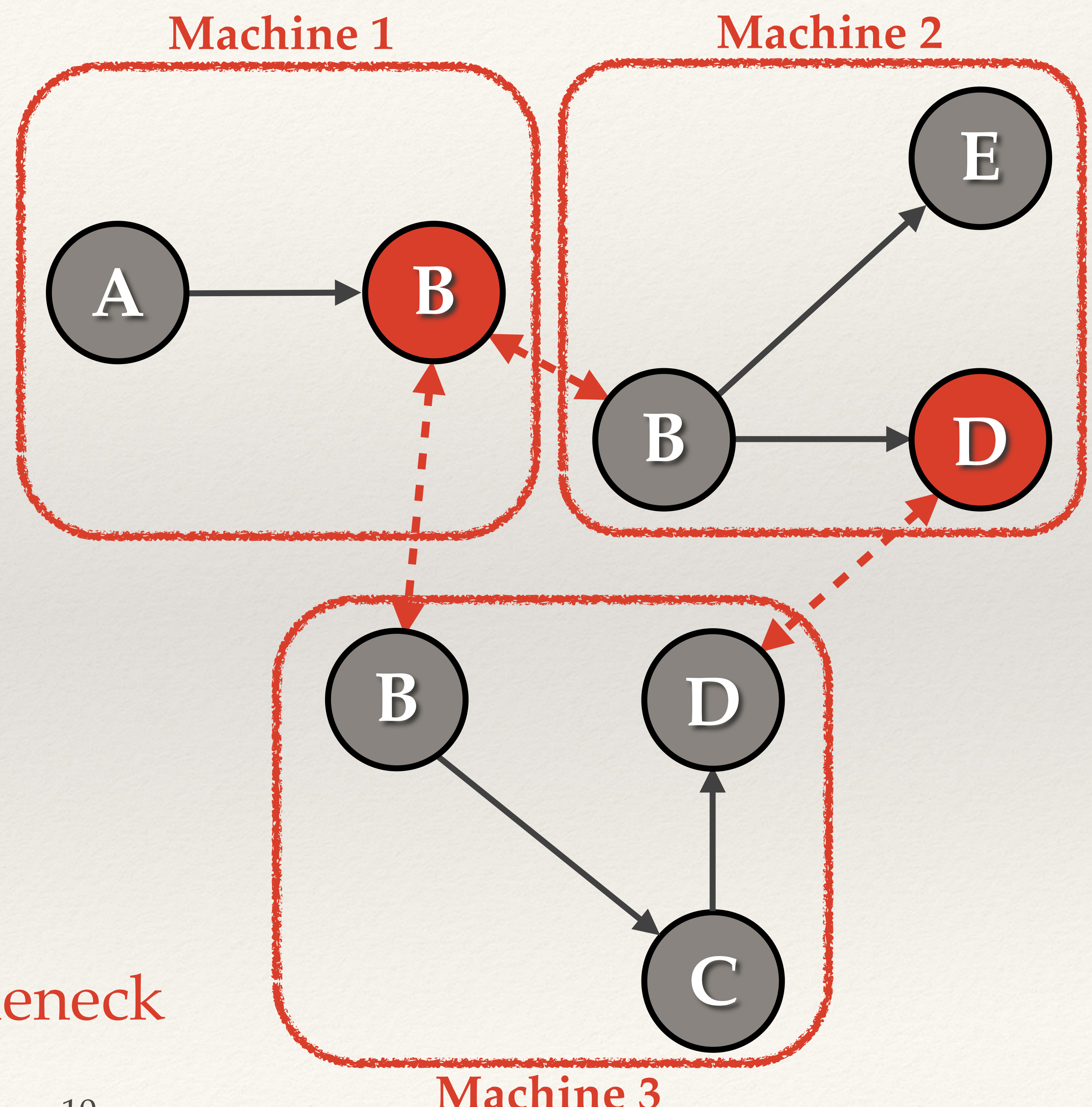
Vertex Cuts

- Assign edges to machines
- High-degree nodes replicated
- One replica designated master
- Need for synchronization
  1. Gather
  2. Apply [on master]
  3. Synchronize mirrors
  4. Scatter
- GraphLab 2.0 - PowerGraph
- Balanced - Network still bottleneck



Vertex Cuts

- Assign edges to machines
- High-degree nodes replicated
- One replica designated master
- Need for synchronization
  1. Gather
  2. Apply [on master]
  3. Synchronize mirrors
  4. Scatter
- GraphLab 2.0 - PowerGraph
- Balanced - Network still bottleneck



Random Walks on GraphLab

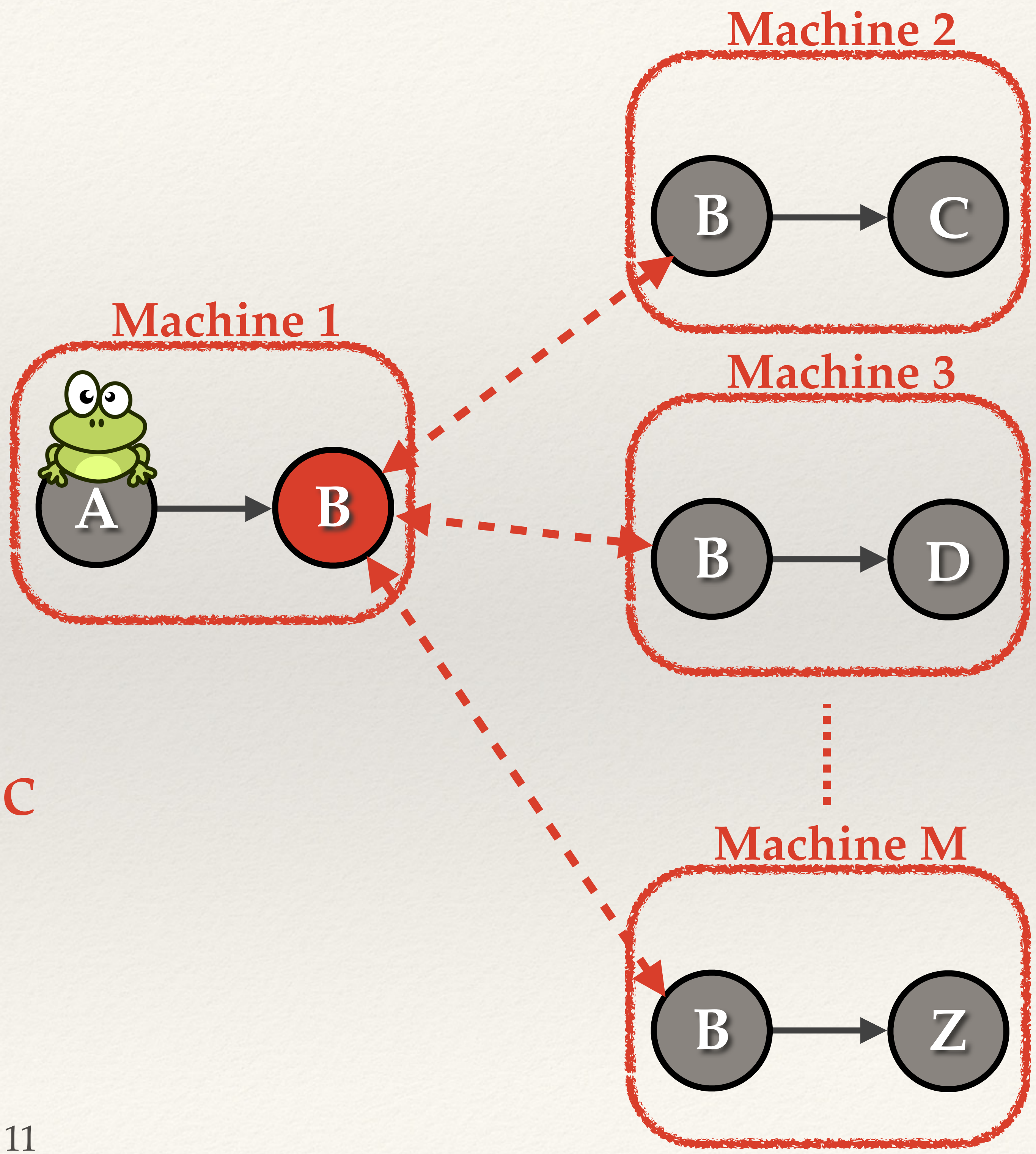
Master node decides step

Decision synced to all mirrors

Only machine M needs it

Unnecessary network traffic

Average replication factor ~8



Random Walks on GraphLab

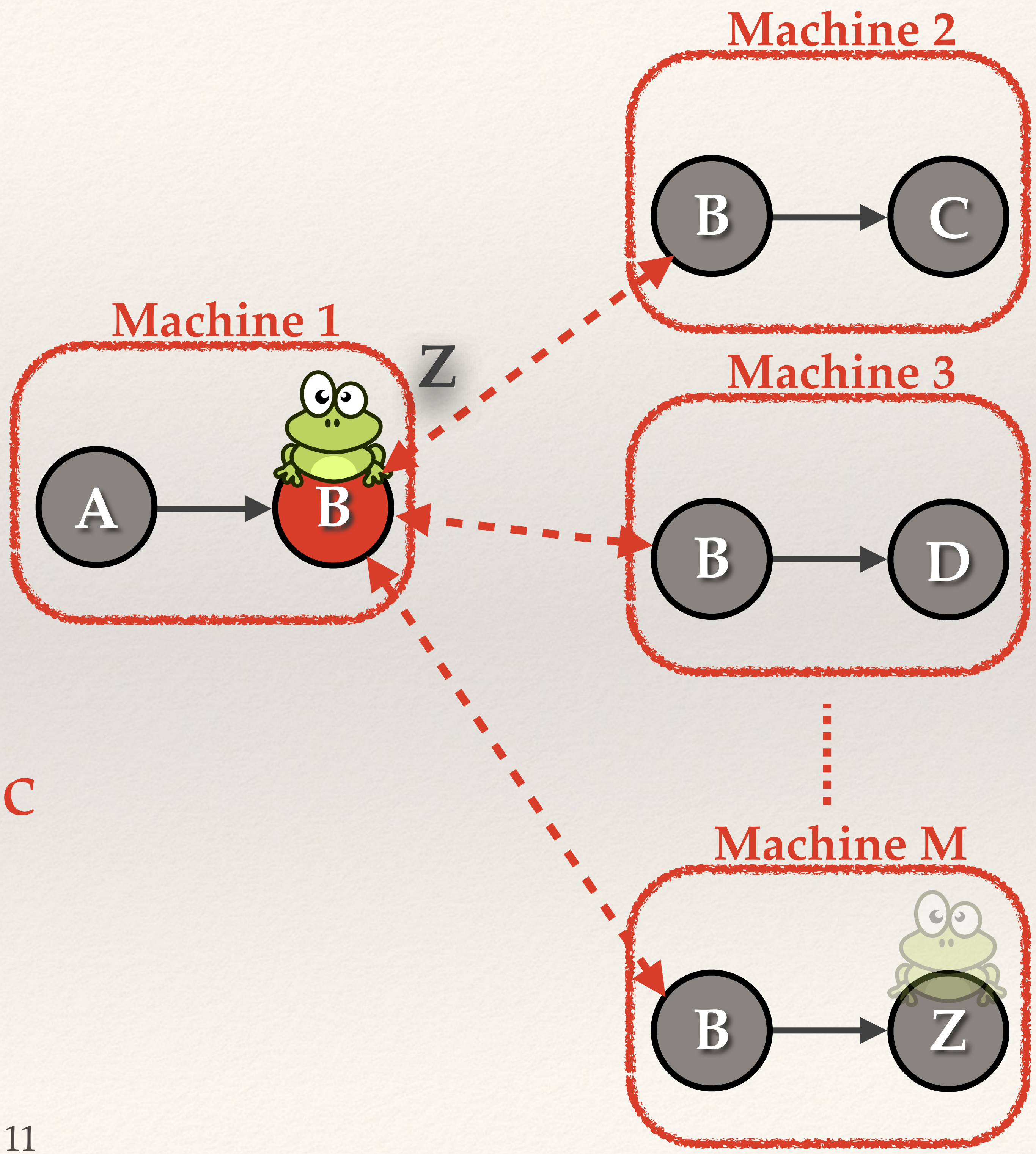
Master node decides step

Decision synced to all mirrors

Only machine M needs it

Unnecessary network traffic

Average replication factor ~8



Random Walks on GraphLab

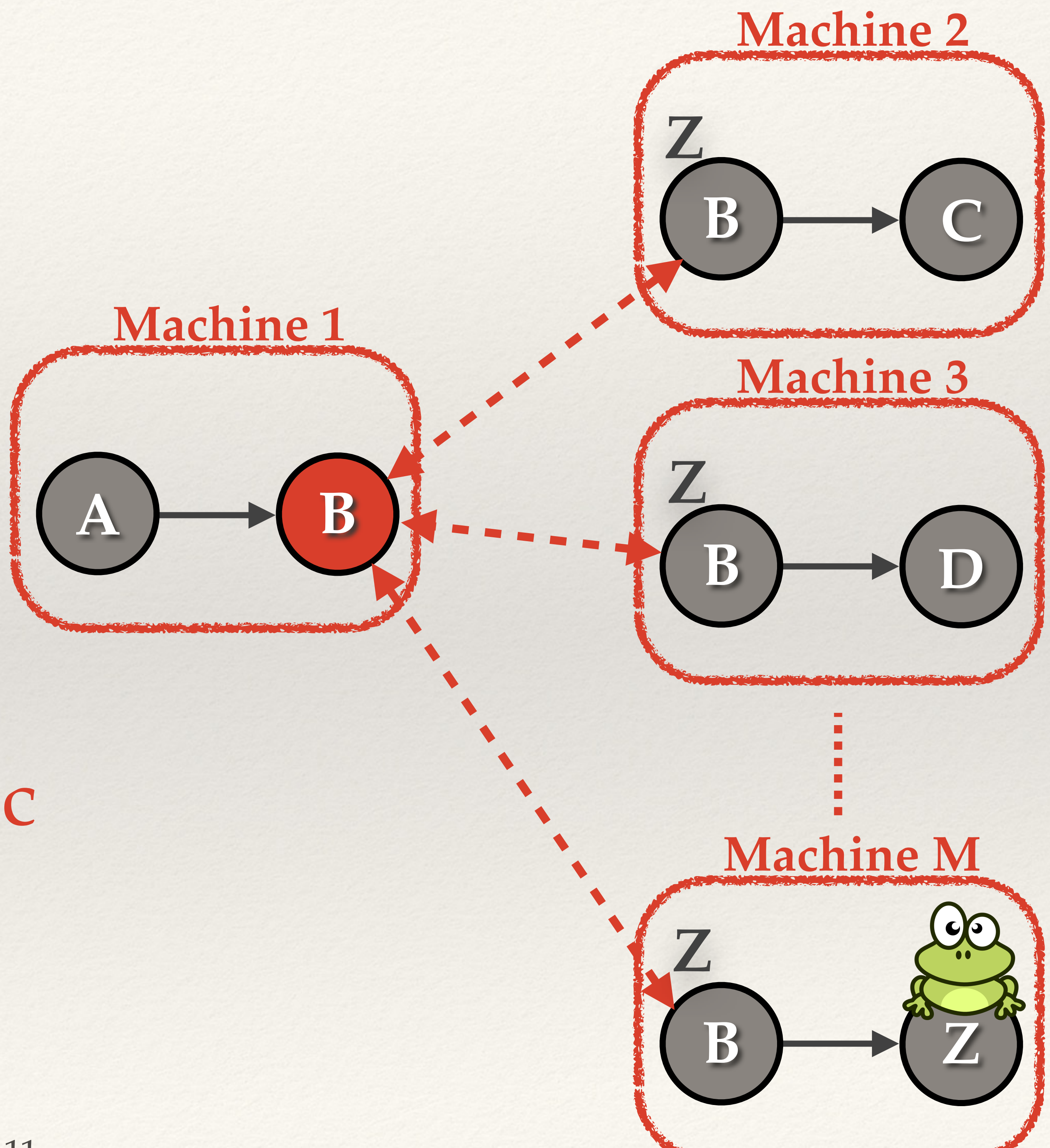
Master node decides step

Decision synced to all mirrors

Only machine M needs it

Unnecessary network traffic

Average replication factor ~8



Objective

Faster PageRank approximation on GraphLab

Idea

Only synchronize the mirror that will receive the frog

Doable, but requires

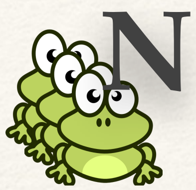
1. Serious engine hacking
2. Exposing an ugly/complicated API to programmer

Simpler

Pick mirrors to synchronize at random!

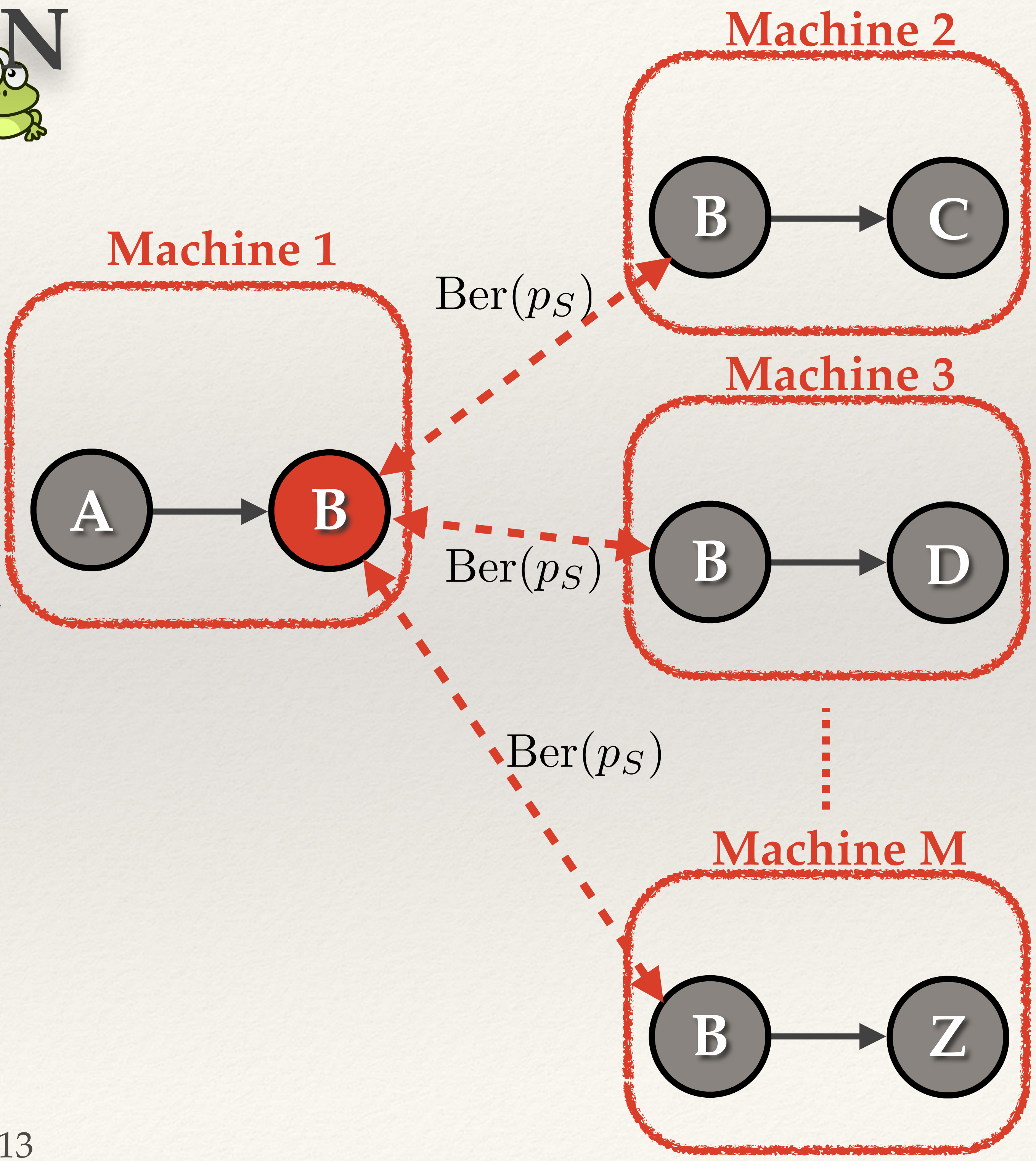
Synchronize independently with probability p_S

FrogWild!

Release N frogs in parallel 

Vertex Program

1. Each frog dies w.p. p_T (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. p_S
3. Spread frogs evenly among synchronized mirrors.

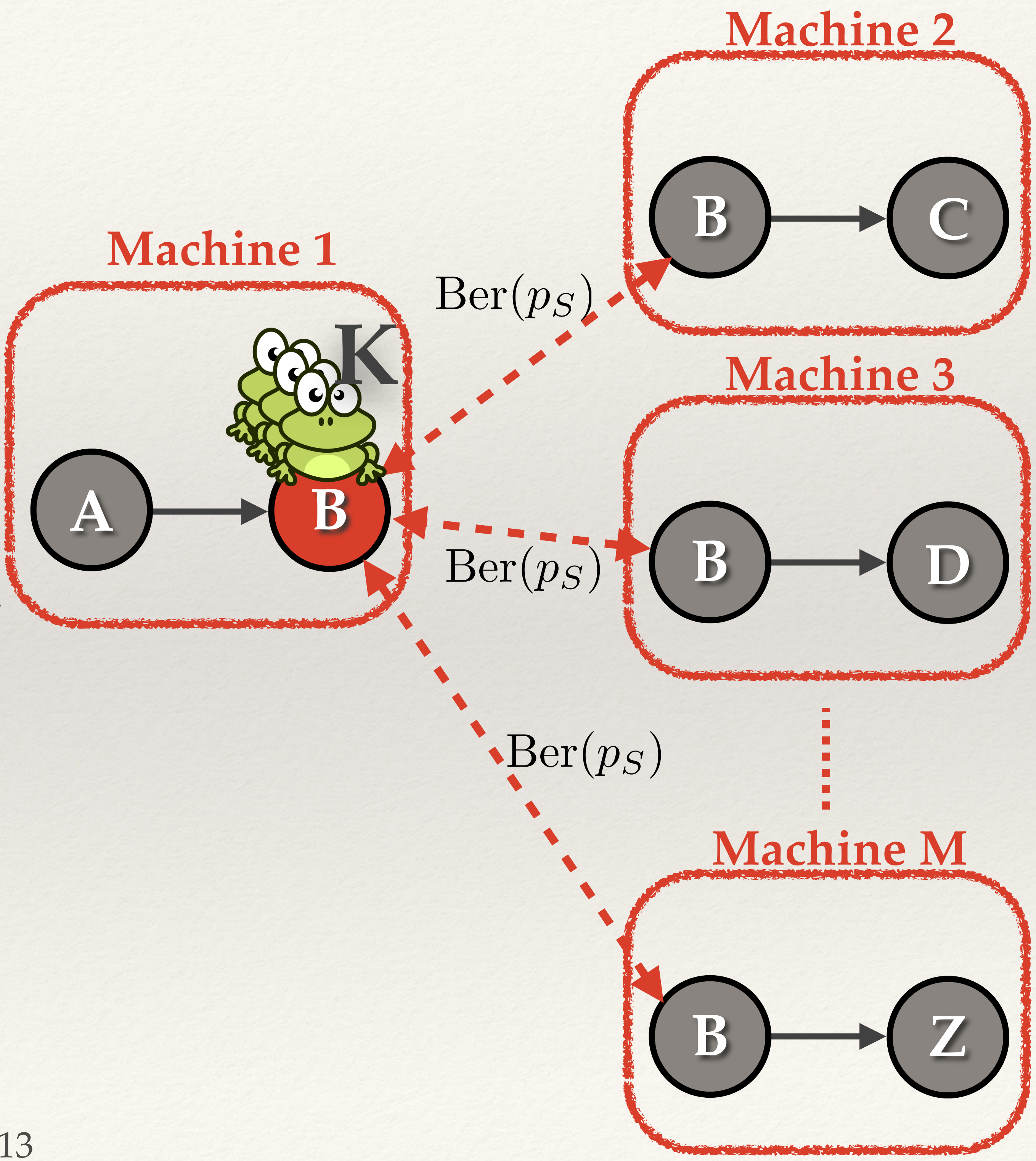


FrogWild!

Release N frogs in parallel

Vertex Program

1. Each frog dies w.p. p_T (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. p_S
3. Spread frogs evenly among synchronized mirrors.

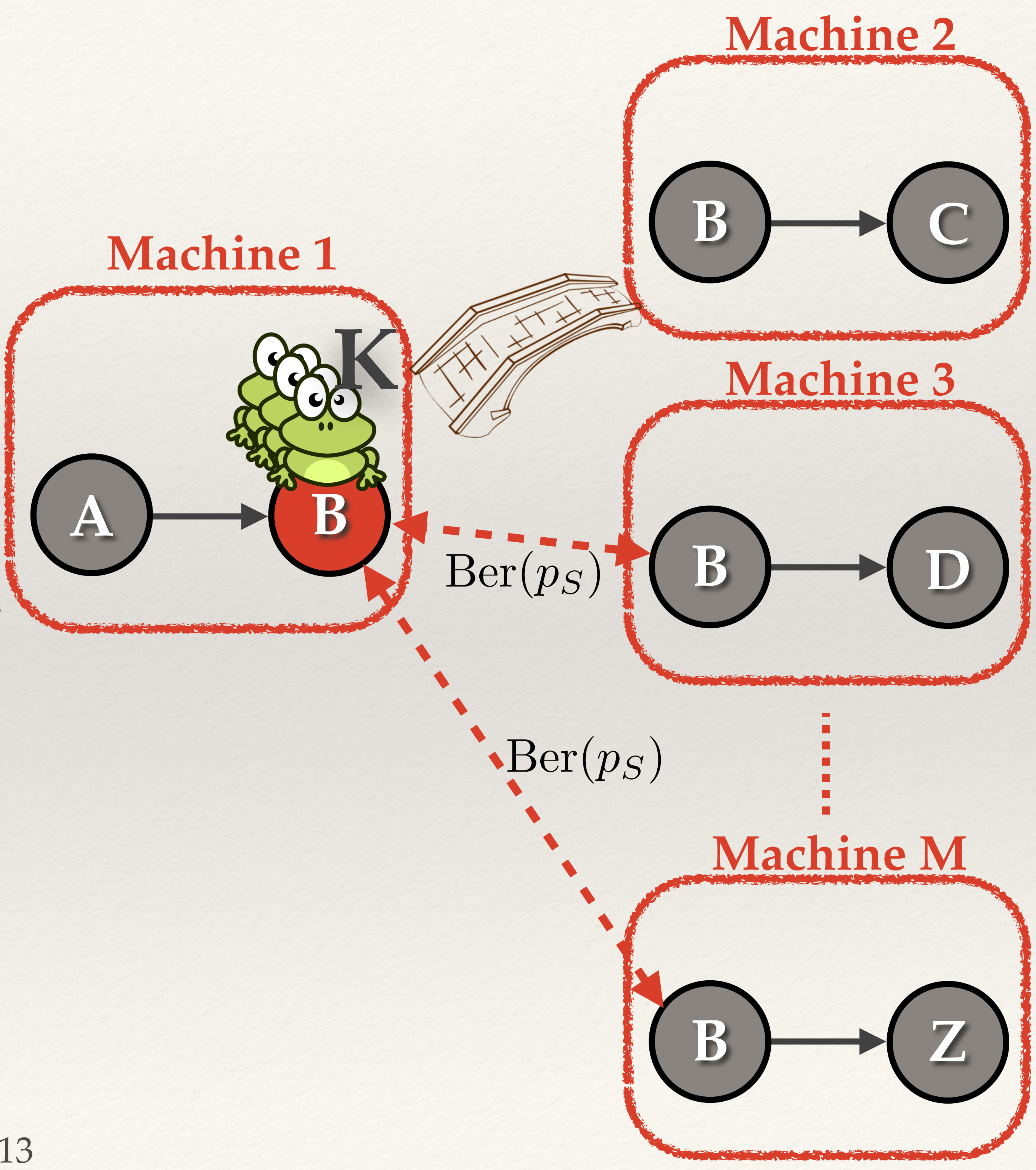


FrogWild!

Release N frogs in parallel

Vertex Program

1. Each frog dies w.p. p_T (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. p_S
3. Spread frogs evenly among synchronized mirrors.

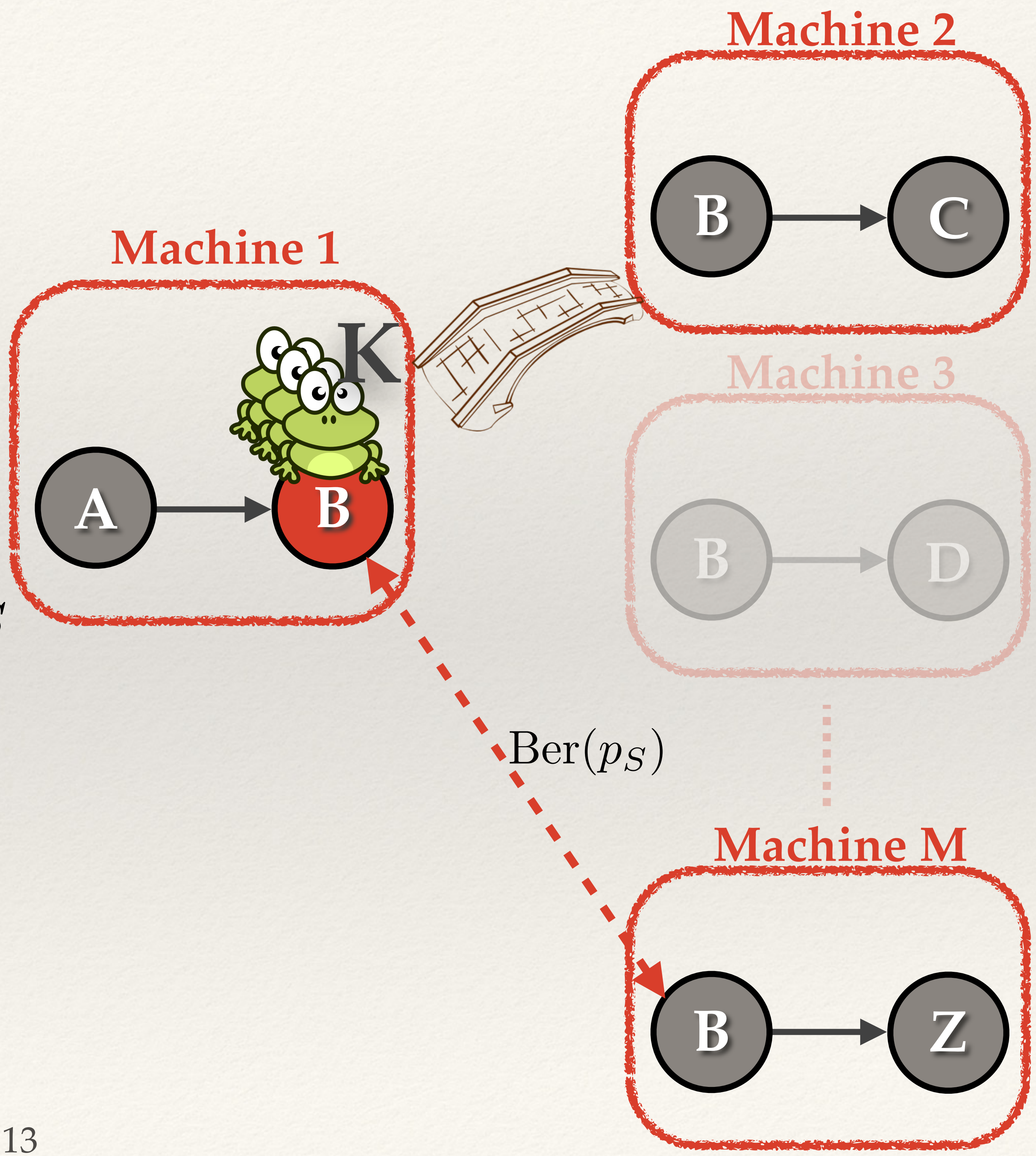


FrogWild!

Release N frogs in parallel

Vertex Program

1. Each frog dies w.p. p_T (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. p_S
3. Spread frogs evenly among synchronized mirrors.

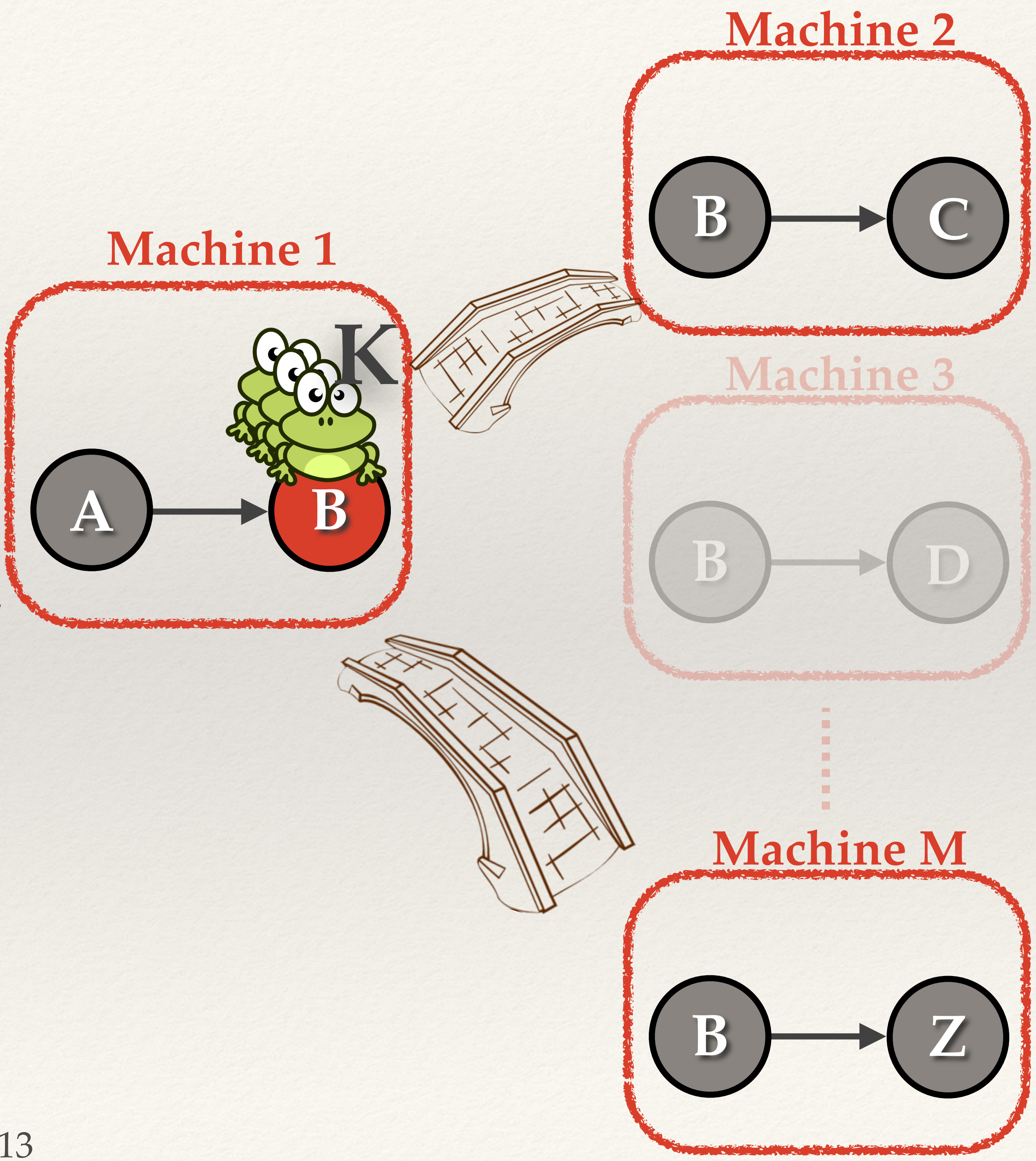


FrogWild!

Release N frogs in parallel

Vertex Program

1. Each frog dies w.p. p_T (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. p_S
3. Spread frogs evenly among synchronized mirrors.

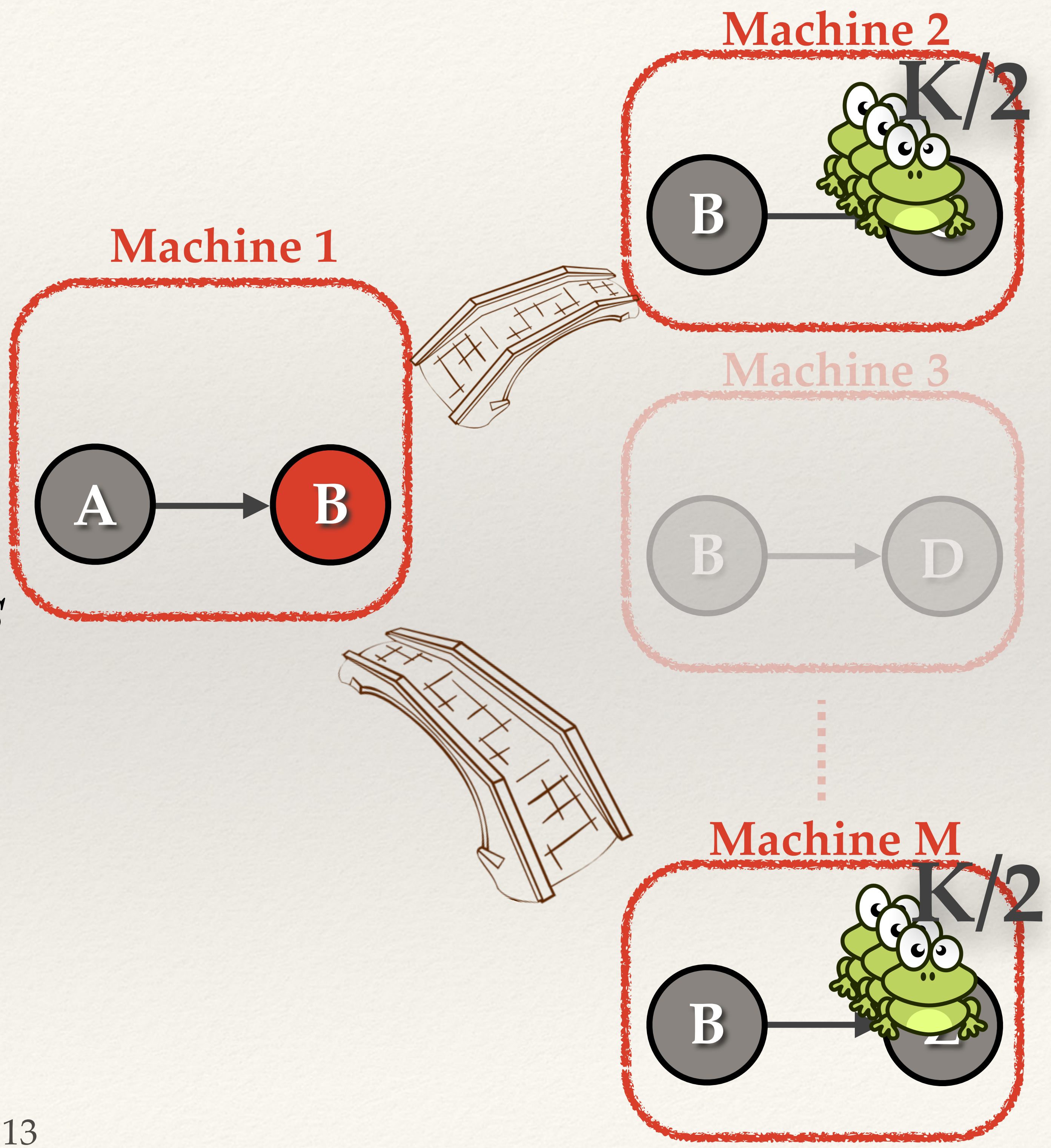


FrogWild!

Release N frogs in parallel

Vertex Program

1. Each frog dies w.p. p_T (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. p_S
3. Spread frogs evenly among synchronized mirrors.

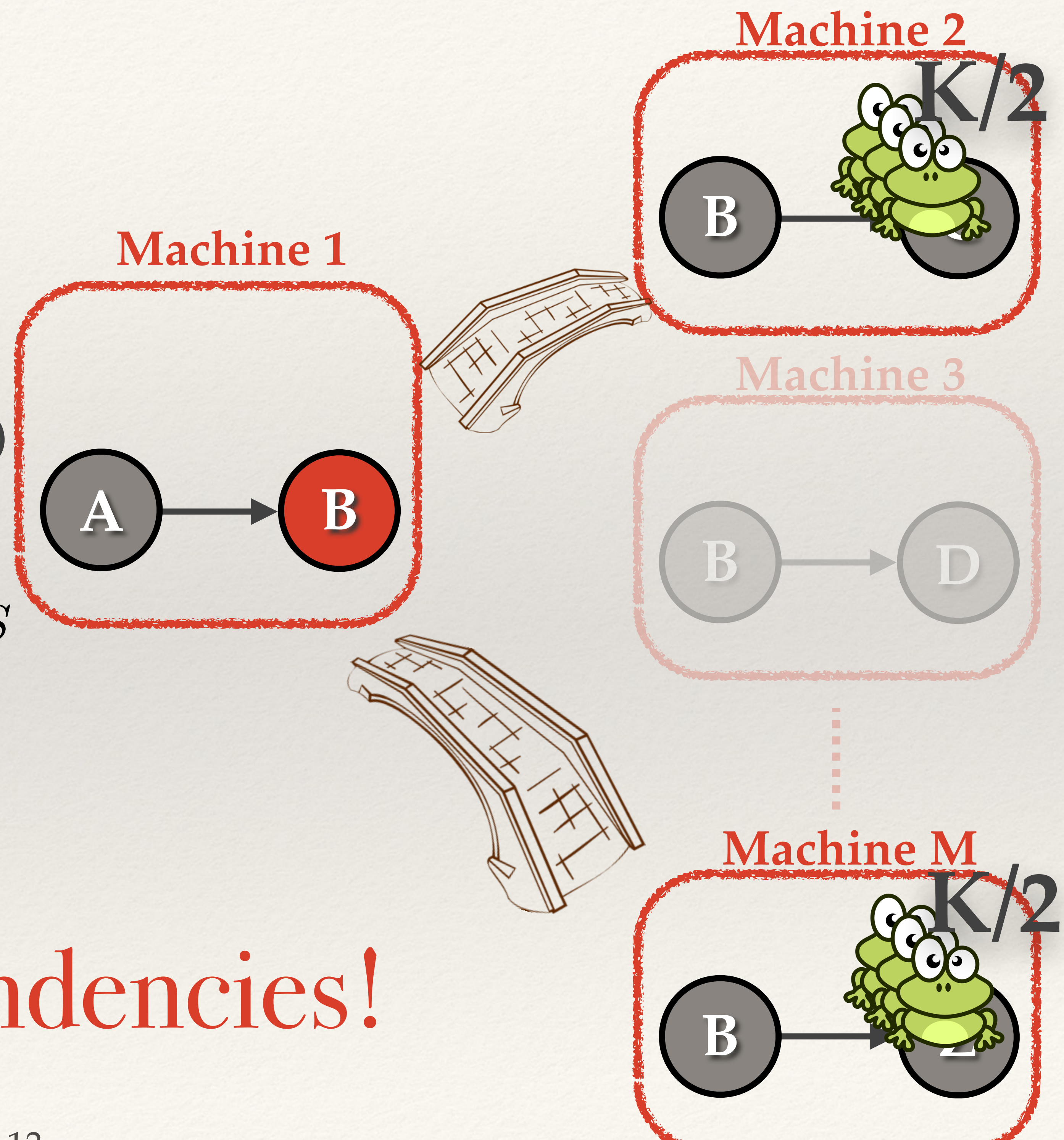


FrogWild!

Release N frogs in parallel

Vertex Program

1. Each frog dies w.p. p_T (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. p_S
3. Spread frogs evenly among synchronized mirrors.



Bridges introduce **dependencies!**

Contributions

1. Algorithm for approximate PageRank
2. Modification of GraphLab
   Exposes very simple API extension (p_S).
   Allows for randomized synchronization.
3. Speedup of 7-10x
4. Theoretical guarantees for solution despite introduced dependencies

Theoretical Guarantee

Mass Captured by top-k set, S, of estimate from N frogs after t steps

$$\pi(S) \geq \mathrm{OPT} - 2\epsilon \quad \text{w.p. } 1-\delta$$

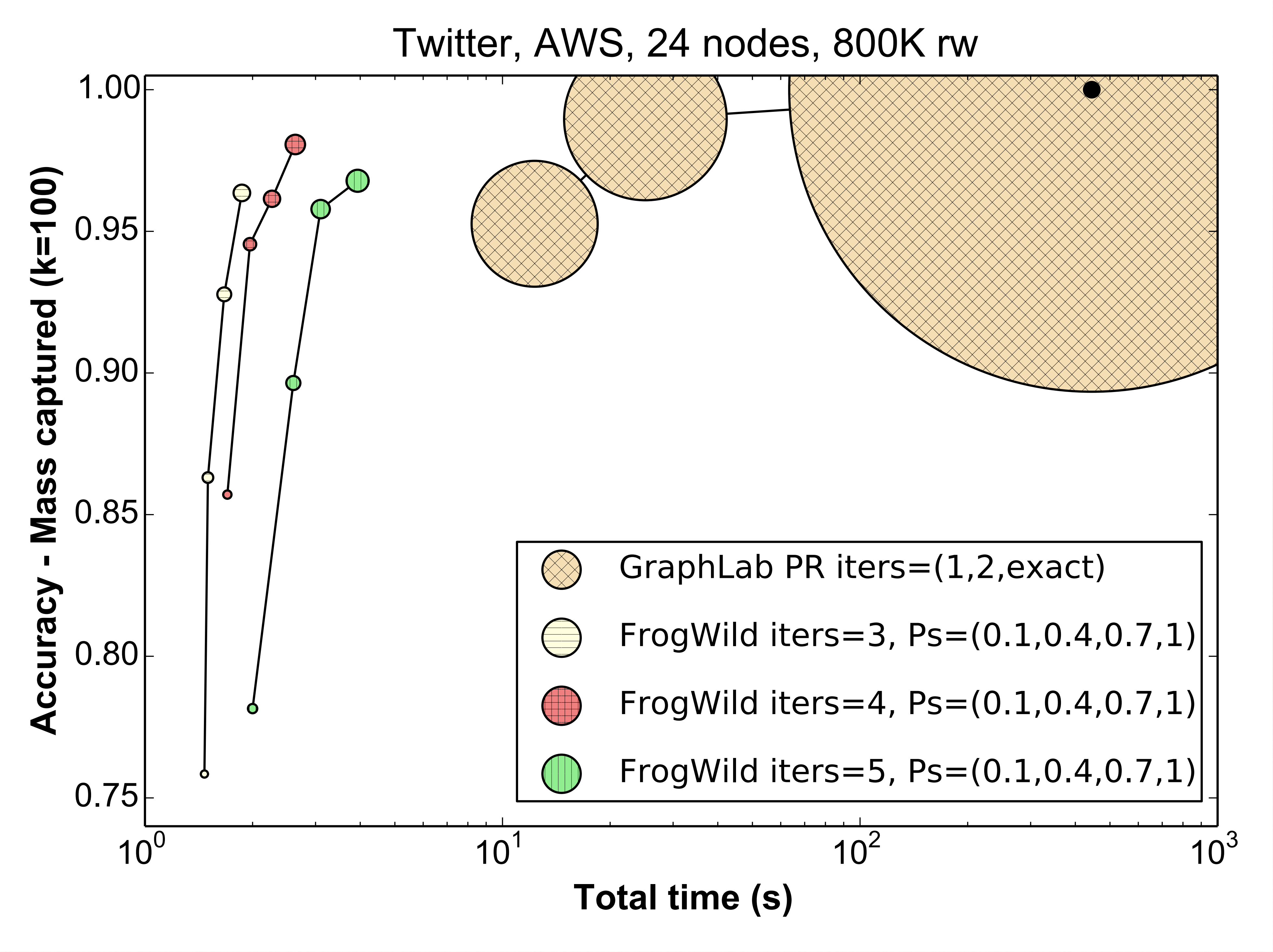
where $$\epsilon < \sqrt{k}\lambda_2^t + \sqrt{\frac{k}{\delta}\left[\frac{1}{N} + (1-p_S^2)p_\cap(t)\right]}$$

probability two Frogs meet at first t steps

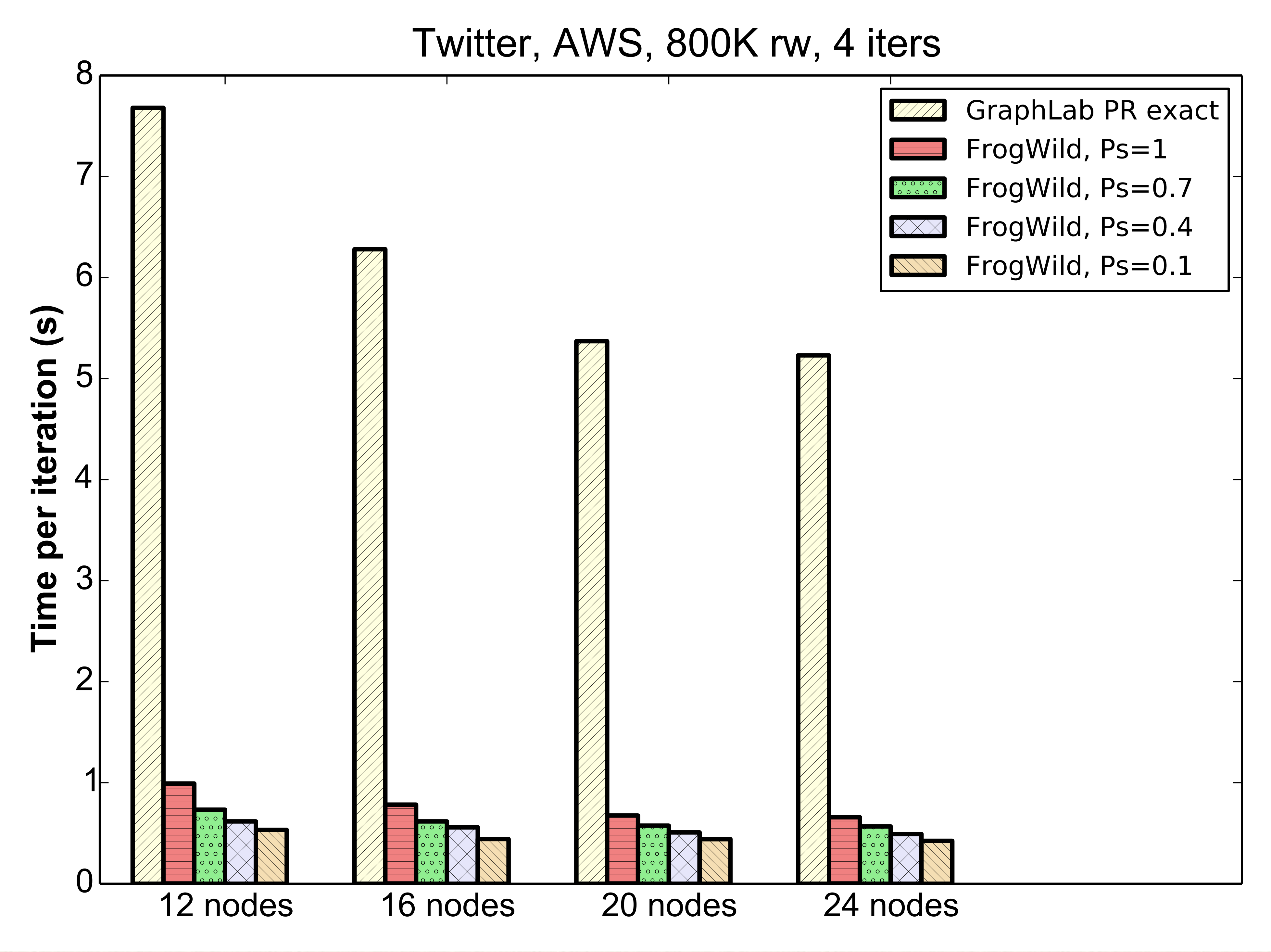
$$p_\cap(t) \leq \frac{1}{n} + \frac{t\|\pi\|_\infty}{p_T},$$

Experiments

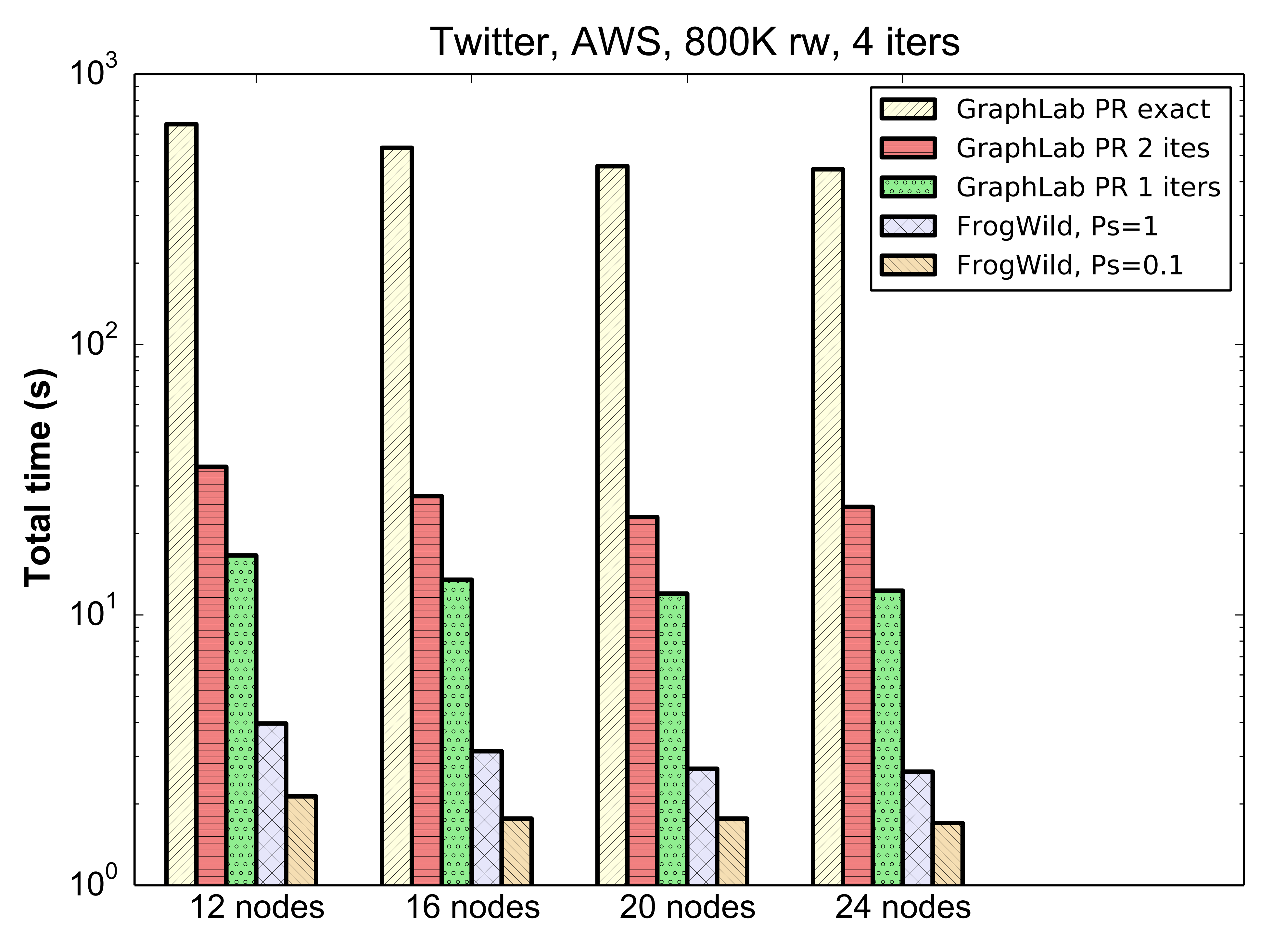
Experimental Results



Experimental Results



Experimental Results



Thank you!

References

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web.

Malewicz, Grzegorz, et al. "Pregel: a system for large-scale graph processing." Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010.

Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., & Hellerstein, J. M. (2010). Graphlab: A new framework for parallel machine learning. arXiv preprint arXiv:1006.4990.

Gonzalez, J. E., Low, Y., Gu, H., Bickson, D., & Guestrin, C. (2012, October). PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs. In OSDI (Vol. 12, No. 1, p. 2).

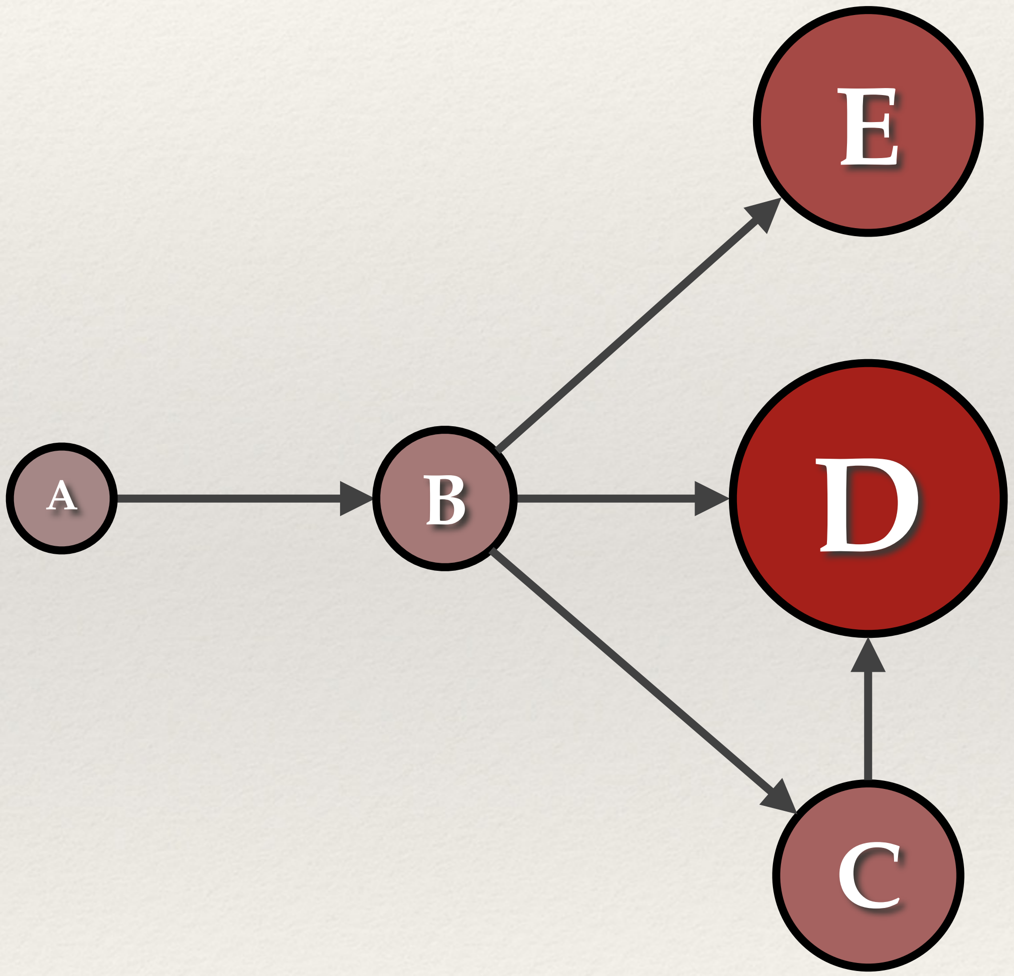
Nguyen, D., Lenharth, A., & Pingali, K. (2013, November). A lightweight infrastructure for graph analytics. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (pp. 456-471). ACM.

Avery, C. (2011). Giraph: Large-scale graph processing infrastruction on Hadoop. Proceedings of Hadoop Summit. Santa Clara, USA:[sn].

Xin, R. S., Gonzalez, J. E., Franklin, M. J., & Stoica, I. (2013, June). Graphx: A resilient distributed graph system on spark. In First International Workshop on Graph Data Management Experiences and Systems (p. 2). ACM.

Backup Slides

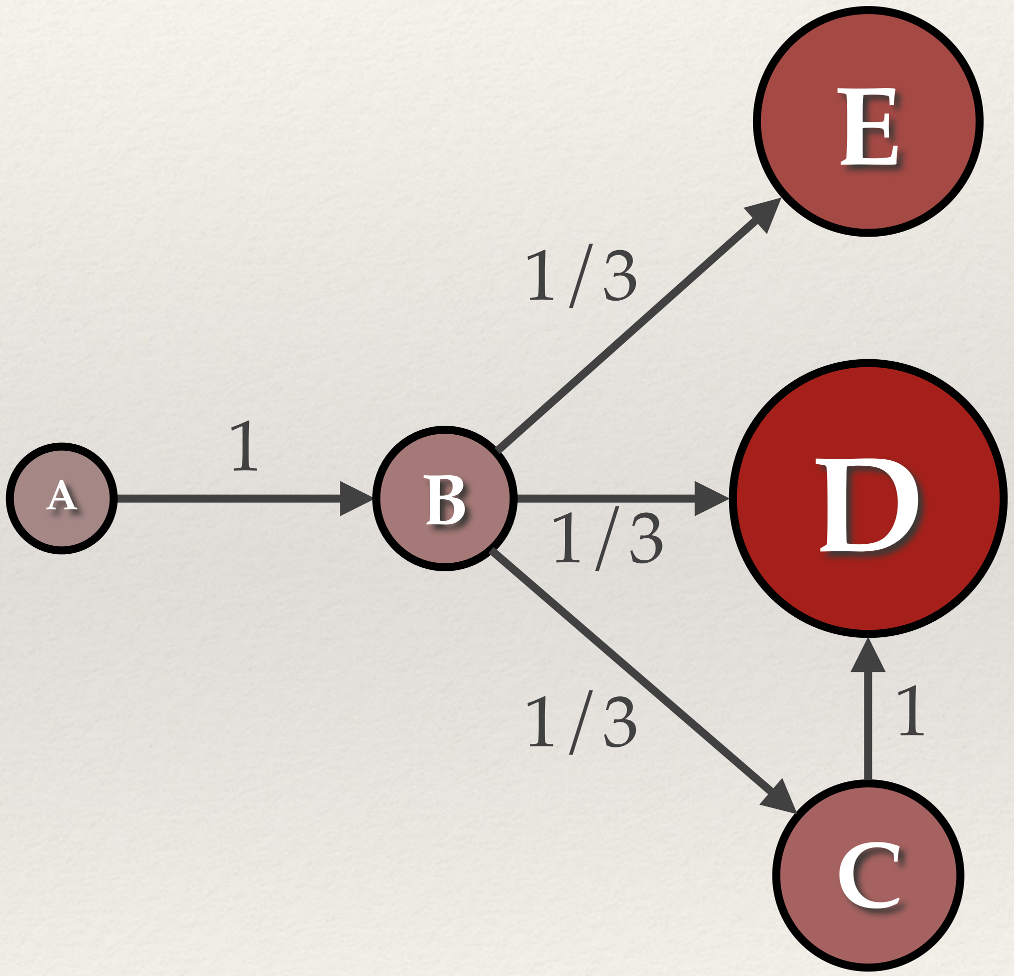
PageRank [Page et al., 1999]



PageRank [Page et al., 1999]

Normalized Adjacency Matrix

$$P_{ij} = \frac{1}{\mathrm{d}_{\mathrm{out}}(j)}, \quad (j, i) \in G$$



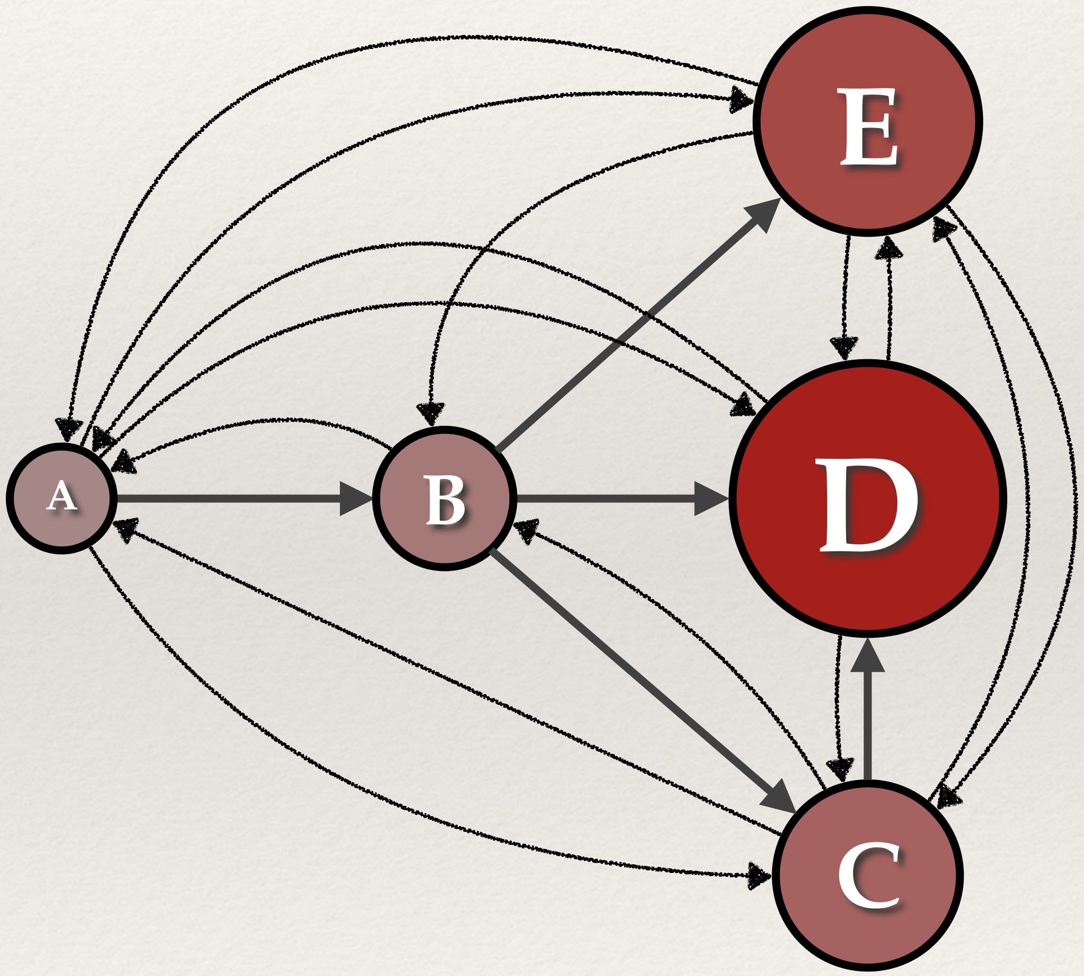
PageRank [Page et al., 1999]

Normalized Adjacency Matrix

$$P_{ij} = \frac{1}{\mathrm{d}_{\mathrm{out}}(j)}, \quad (j, i) \in G$$

Augmented Matrix $p_T \in [0, 1]$

$$Q_{ij} = (1 - p_T) P_{ij} + \frac{p_T}{n}$$



PageRank [Page et al., 1999]

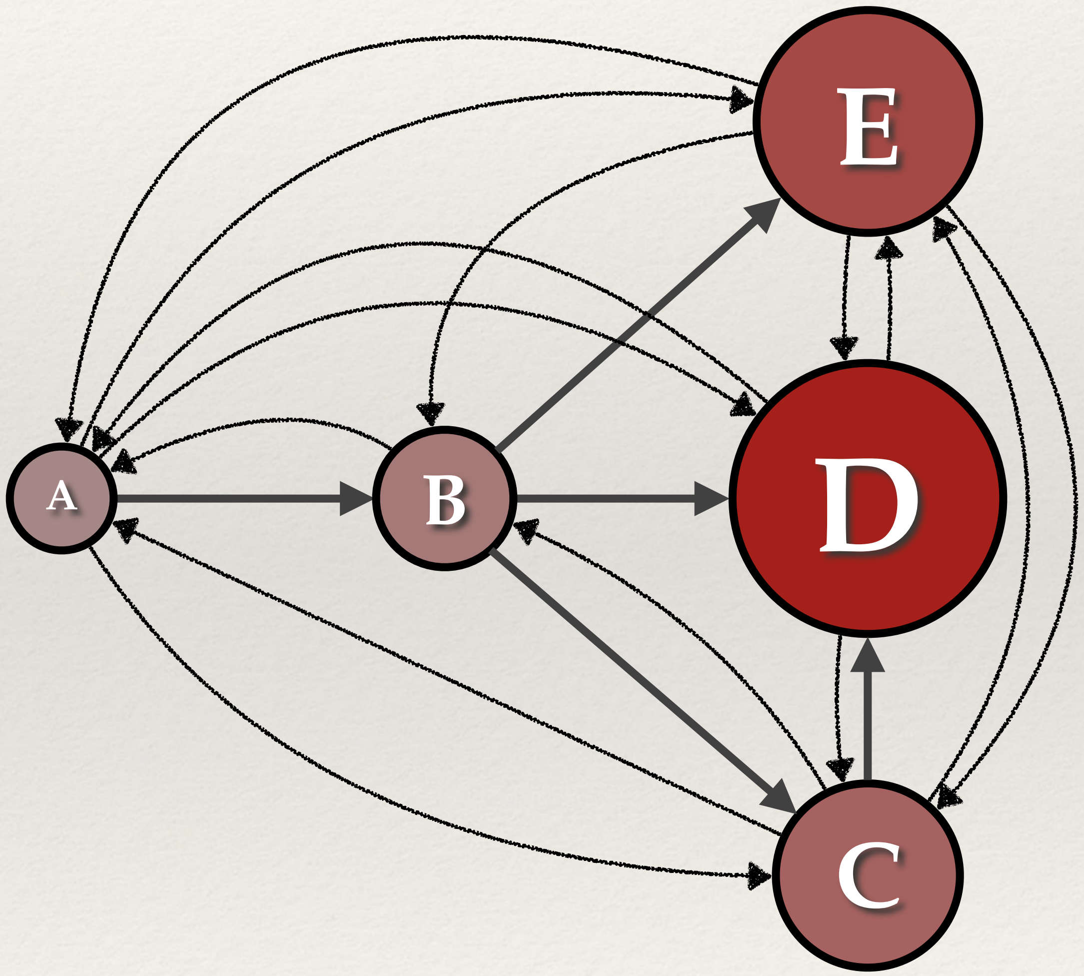
Normalized Adjacency Matrix

$$P_{ij} = \frac{1}{\mathrm{d}_{\mathrm{out}}(j)}, \quad (j, i) \in G$$

Augmented Matrix $p_T \in [0, 1]$

$$Q_{ij} = (1 - p_T) P_{ij} + \frac{p_T}{n}$$

PageRank Vector $\pi = Q\pi$



PageRank [Page et al., 1999]

Normalized Adjacency Matrix

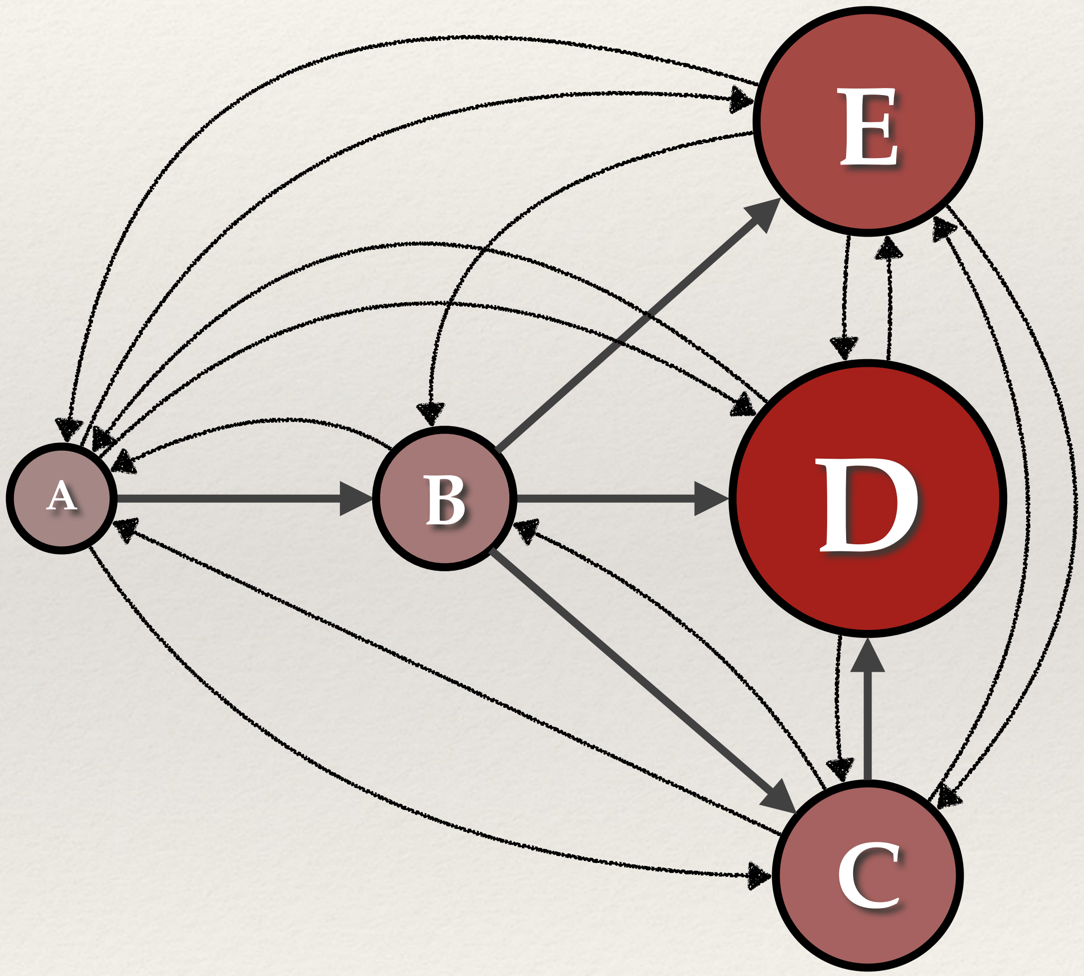
$$P_{ij} = \frac{1}{\mathrm{d_{out}}(j)}, \quad (j, i) \in G$$

Augmented Matrix $p_T \in [0, 1]$

$$Q_{ij} = (1 - p_T)P_{ij} + \frac{p_T}{n}$$

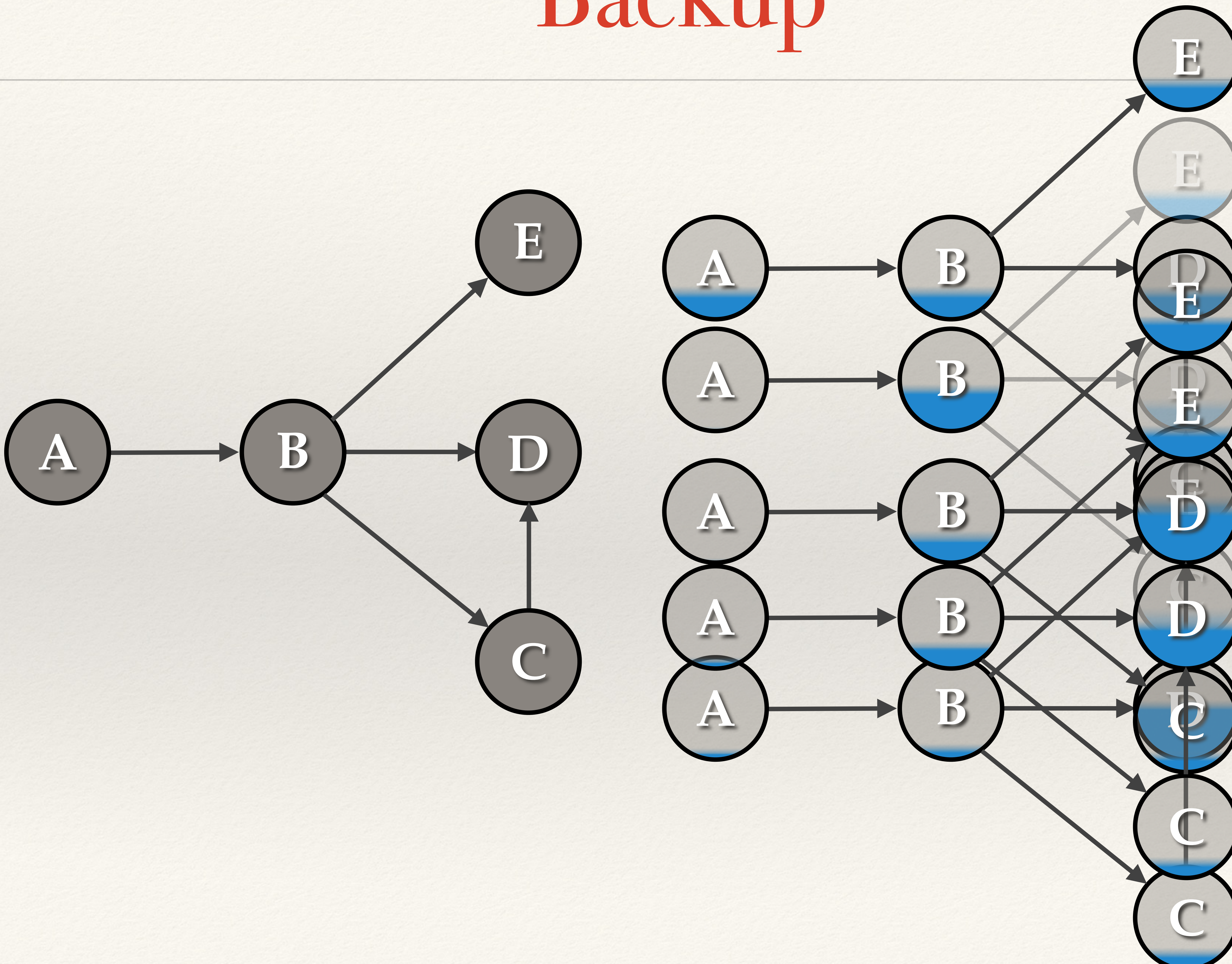
PageRank Vector $\pi = Q\pi$

Power Method $Q^t p^0 \to \pi$



Here be dragons.

Backup



Backup