Efficient Randomized Algorithms for Distributed Systems

Information Dissemination, Distributed Voting, and Plurality Consensus

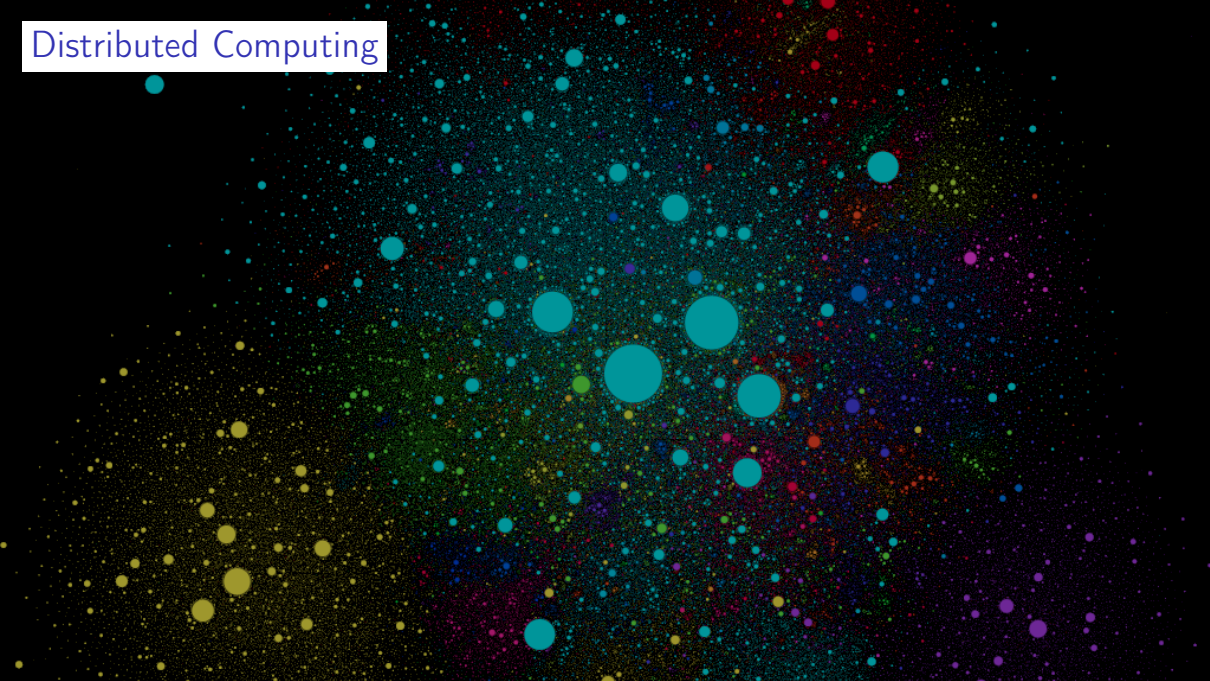
Dominik S. Kaaser

PhD Defense

January 26, 2017

Outline

Distributed Computing



Distributed Computing

Characterizing Distributed Systems

- collection of connected computing devices
- solve suitable subproblems in parallel

Distributed Computing

Characterizing Distributed Systems

- collection of connected computing devices
- solve suitable subproblems in parallel

- improved performance
- resilience against component failure

Distributed Computing

Characterizing Distributed Systems

- collection of connected computing devices
- solve suitable subproblems in parallel

- improved performance
- resilience against component failure

- lack of common memory

Distributed Computing

Characterizing Distributed Systems

- collection of connected computing devices
- solve suitable subproblems in parallel

- improved performance
- resilience against component failure

- lack of common memory
- lack of common clock

Distributed Computing

Characterizing Distributed Systems

- collection of connected computing devices
- solve suitable subproblems in parallel

- improved performance
- resilience against component failure

- lack of common memory
- lack of common clock
- lack of network structure

Distributed Computing

Characterizing Distributed Systems

- collection of connected computing devices
- solve suitable subproblems in parallel

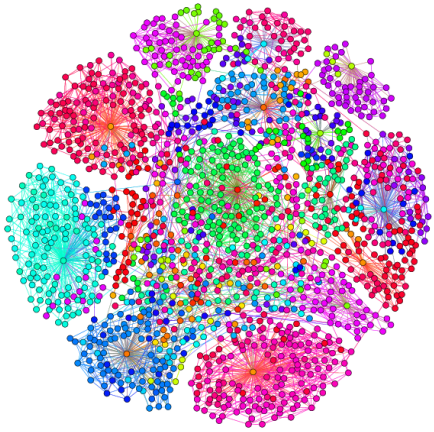
- improved performance
- resilience against component failure

- lack of common memory
- lack of common clock
- lack of network structure
- heterogeneity

Distributed Computing

A Model for Distributed Systems

- graph $G = (V, E)$ with $|V| = n$
- communication with direct neighbors
- algorithms operate in synchronous rounds



Distributed Computing

A Model for Distributed Systems

- graph $G = (V, E)$ with $|V| = n$
- communication with direct neighbors
- algorithms operate in synchronous rounds

Optimize for...

- runtime efficiency
- local memory requirements
- communication overhead
- fault tolerance
- energy consumption
- ...

Information Dissemination

Communication in the Random Phone Call Model

The Random Phone Call Model

- in each round, open a connection to a randomly chosen neighbor
- bi-directional communication over this channel

Demers et al., 1987

Karp, Schindelhauer, Shenker, and Vöcking, 2000

Information Dissemination

Communication in the Random Phone Call Model

The Random Phone Call Model

- in each round, open a connection to a randomly chosen neighbor
- bi-directional communication over this channel

Demers et al., 1987

Karp, Schindelhauer, Shenker, and Vöcking, 2000

Randomization

- efficient randomized algorithm
- solve the problem *with high probability:* $1 - n^{-\Omega(1)}$

Information Dissemination

The Gossiping Problem

- each node has its own initial message

Information Dissemination

The Gossiping Problem

- each node has its own initial message
- goal: distribute all messages to all nodes

Information Dissemination

The Gossiping Problem

- each node has its own initial message
- goal: distribute all messages to all nodes
- $\mathcal{O}(\log n)$ time, $\Omega(n \log n)$ messages

Berenbrink, Czyzowicz, Elsässer, and Gąsieniec, 2010

Information Dissemination

The Gossiping Problem

- each node has its own initial message
- goal: distribute all messages to all nodes
- $\mathcal{O}(\log n)$ time, $\Omega(n \log n)$ messages

  Berenbrink, Czyzowicz, Elsässer, and Gąsieniec, 2010
- gossiping algorithms do extend to sparse graphs

Our Results

Theorem *Elsässer and K., 2015*

The gossiping problem can be solved on a random regular graph with node degree $\Omega(\log^k n)$ for $k \geq 4$ in $\mathcal{O}(\log^2 n / \log\log n)$ time using $\mathcal{O}(n \log n / \log\log n)$ message transmissions, with high probability.

Robert Elsässer and D. K.: *On the Influence of Graph Density on Randomized Gossiping*. In *Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2015.

Our Results

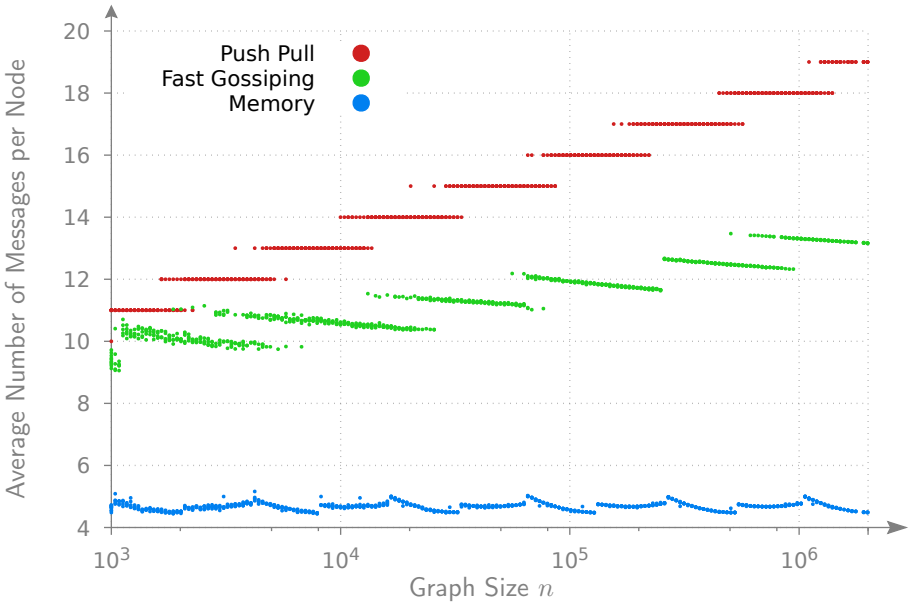
Theorem *Elsässer and K., 2015*

The gossiping problem can be solved on a random regular graph with node degree $\Omega(\log^k n)$ for $k \geq 4$ in $\mathcal{O}(\log^2 n / \log\log n)$ time using $\mathcal{O}(n \log n / \log\log n)$ message transmissions, with high probability.

Theorem *Elsässer and K., 2015*

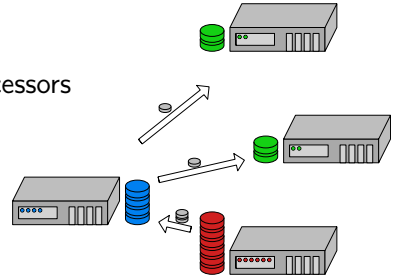
If we may *store* a constant number of connections, the gossiping problem can be solved in $\mathcal{O}(\log n)$ time using only $\mathcal{O}(n)$ message transmissions, with high probability.

Robert Elsässer and D. K.: *On the Influence of Graph Density on Randomized Gossiping*. In *Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2015.



Diffusion Based Load Balancing

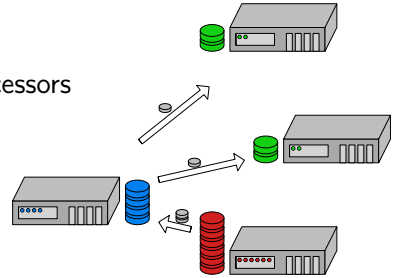
- parallel machine modeled as graph
- different amounts of load generated on processors
- balance load to obtain a substantial benefit for the runtime of the parallel computation



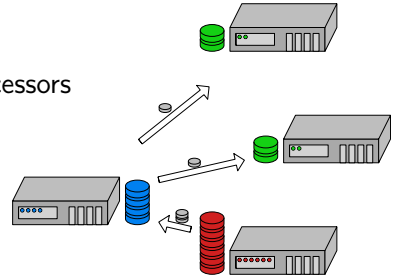
Diffusion Based Load Balancing

- parallel machine modeled as graph
- different amounts of load generated on processors
- balance load to obtain a substantial benefit for the runtime of the parallel computation
- exchange load with direct neighbors
- assume continuous load items

  Diekmann, Frommer, and Monien, 1999



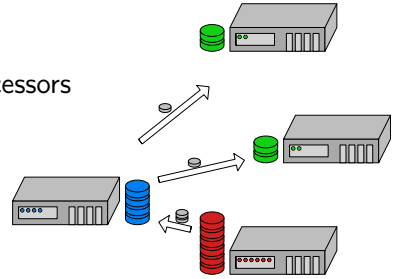
Diffusion Based Load Balancing



- parallel machine modeled as graph
- different amounts of load generated on processors
- balance load to obtain a substantial benefit for the runtime of the parallel computation
- exchange load with direct neighbors
- assume continuous load items
  Diekmann, Frommer, and Monien, 1999
- bound deviation of discrete schemes from continuous schemes
  Rabani, Sinclair, and Wanka, 1998

Diffusion Based Load Balancing

- parallel machine modeled as graph
- different amounts of load generated on processors
- balance load to obtain a substantial benefit for the runtime of the parallel computation
- exchange load with direct neighbors
- assume continuous load items
  Diekmann, Frommer, and Monien, 1999
- bound deviation of discrete schemes from continuous schemes
  Rabani, Sinclair, and Wanka, 1998



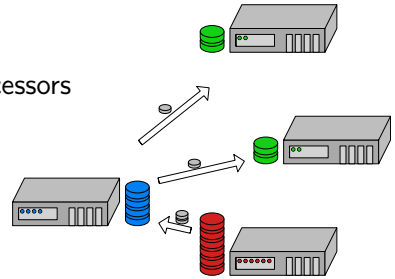
First Order Scheme – FOS

load sent to neighbors depends only on load difference in current round

Muthukrishnan, Ghosh, and Schultz, 1998

Diffusion Based Load Balancing

- parallel machine modeled as graph
- different amounts of load generated on processors
- balance load to obtain a substantial benefit for the runtime of the parallel computation
- exchange load with direct neighbors
- assume continuous load items
  Diekmann, Frommer, and Monien, 1999
- bound deviation of discrete schemes from continuous schemes
  Rabani, Sinclair, and Wanka, 1998



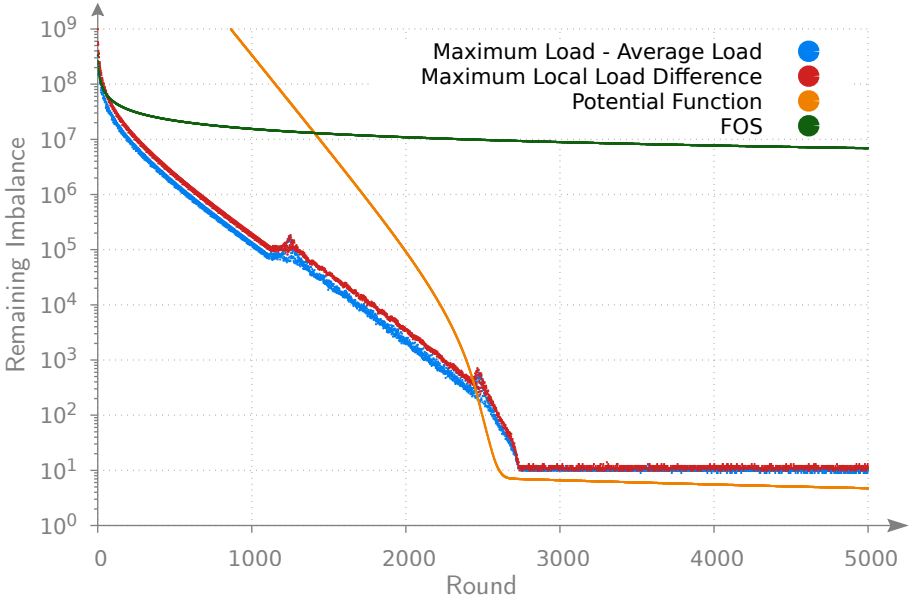
Second Order Scheme – SOS

also take flow from previous round into account

Muthukrishnan, Ghosh, and Schultz, 1998

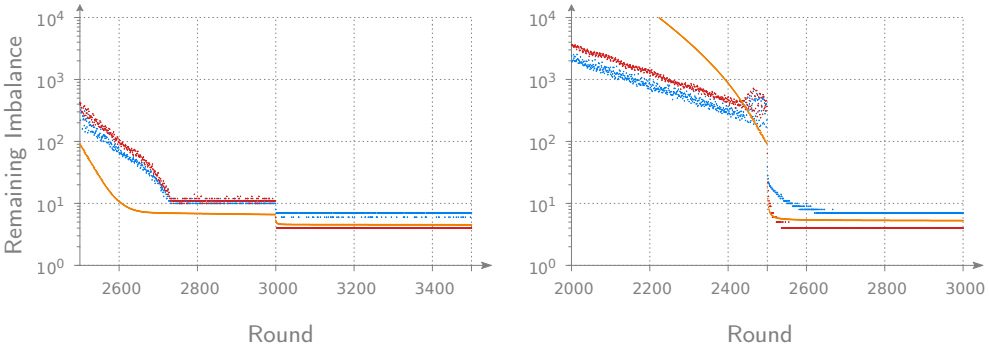
Our Results

Empirical Analysis: 2D Torus (1000×1000)



Our Results

Switch from SOS to FOS



Hoda Akbari, Petra Berenbrink, Robert Elsässer, and D. K.: *Discrete Load Balancing in Heterogeneous Networks with a Focus on Second-Order Diffusion*. In *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2015.

Distributed Voting

The Deterministic Binary Majority Voting Process

- given a graph $G = (V, E)$
- initial opinion assignment $f_0 : V \to \{0, 1\}$
- every node adopts the majority opinion in every round

Known Results

- The process always converges to a two-periodic state. *Goles and Olivos, 1980*
  Poljak and Sûra, 1983

Known Results

- The process always converges to a two-periodic state. *Goles and Olivos, 1980* *Poljak and Sûra, 1983*
- The process converges after at most $\mathcal{O}(|E|)$ rounds. *Winkler, 2008*

Known Results

- The process always converges to a two-periodic state. *Goles and Olivos, 1980* *Poljak and Sûra, 1983*
- The process converges after at most $\mathcal{O}(|E|)$ rounds. *Winkler, 2008*
- These bounds are tight. *Frischknecht, Keller, and Wattenhofer, 2013* *Keller, Peleg, and Wattenhofer, 2014*

Known Results

- The process always converges to a two-periodic state. *Goles and Olivos, 1980*
  Poljak and Sûra, 1983
- The process converges after at most $\mathcal{O}(|E|)$ rounds. *Winkler, 2008*
- These bounds are tight. *Frischknecht, Keller, and Wattenhofer, 2013*
  Keller, Peleg, and Wattenhofer, 2014
- Application: analysis of community-detection algorithms
  Raghavan, Albert, and Kumara, 2007
  Cordasco and Gargano, 2010
  Kothapalli, Pemmaraju, and Sardeshmukh, 2013

Our Results

NP Hardness Result

Definition: voting time decision problem (VTDP)

For a given graph G and an integer k, is there an assignment of initial opinions such that the voting time of G is at least k?

Theorem *K., Mallmann-Trenn, and Natale, 2016*

Given a general simple graph G, VTDP is NP-complete.

D. K., Frederik Mallmann-Trenn, and Emanuele Natale: *On the Voting Time of the Deterministic Majority Process*. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2016.

Our Results

Bounds on the Voting Time

Definition

A set of nodes S is called a *family* if $\forall\ u, v \in S : N(u) \setminus \{v\} = N(v) \setminus \{u\}$.

The *asymmetric graph* $G^{\Delta} = (V^{\Delta}, E^{\Delta})$ is constructed from $G = (V, E)$ by contracting every *family* to one (or two) nodes.

Theorem

K., Mallmann-Trenn, and Natale, 2016

The voting time is at most

$$1 + \min\left\{ |E^{\Delta}| - \frac{|V^{\Delta}_{odd}|}{2},\quad \frac{|E^{\Delta}|}{2} + \frac{|V^{\Delta}_{even}|}{4} + \frac{7}{4} \cdot |V^{\Delta}| \right\} .$$

Our Results

Bounds on the Voting Time

Definition

A set of nodes S is called a *family* if $\forall\ u, v \in S : N(u) \setminus \{v\} = N(v) \setminus \{u\}$.

The *asymmetric graph* $G^{\Delta} = (V^{\Delta}, E^{\Delta})$ is constructed from $G = (V, E)$ by contracting every *family* to one (or two) nodes.

Theorem *K., Mallmann-Trenn, and Natale, 2016*

The voting time is bounded by the voting time in G^{Δ}.

Plurality Consensus

- given a complete graph of n nodes and $k = n^{\epsilon}$ possible opinions

Plurality Consensus

- given a complete graph of n nodes and $k = n^{\epsilon}$ possible opinions

Pull Voting

- open connection to randomly chosen neighbor
- adopt this neighbor's opinion
- requires $\Omega(n)$ rounds to converge

Nakata, Imahayashi, and Yamashita, 1999
Hassin and Peleg, 2001
Cooper, Elsässer, Ono, and Radzik, 2013
Berenbrink, Giakkoupis, Kermarrec, and Mallmann-Trenn, 2016

Plurality Consensus

- given a complete graph of n nodes and $k = n^{\epsilon}$ possible opinions

Pull Voting

- open connection to randomly chosen neighbor
- adopt this neighbor's opinion
- requires $\Omega(n)$ rounds to converge

Two-Choices Process

- sample two nodes and if their opinions coincide, adopt it
- requires a bias of $\Omega\left(\sqrt{n \log n}\right)$ for the plurality opinion A to win
- however, only after $\Omega(k)$ rounds

Cooper, Elsässer, and Radzik, 2014

Cooper, Elsässer, Radzik, Rivera, and Shiraga, 2015

One Bit of Memory

For $\mathcal{O}(\log n)$ *phases* **do at each node** v **in parallel**

1 `Two-Choices Round;`

$\log k$ `Bit-Propagation Rounds;`

Theorem *Elsässer, Friedetzky, K., Mallmann-Trenn, and Trinker*

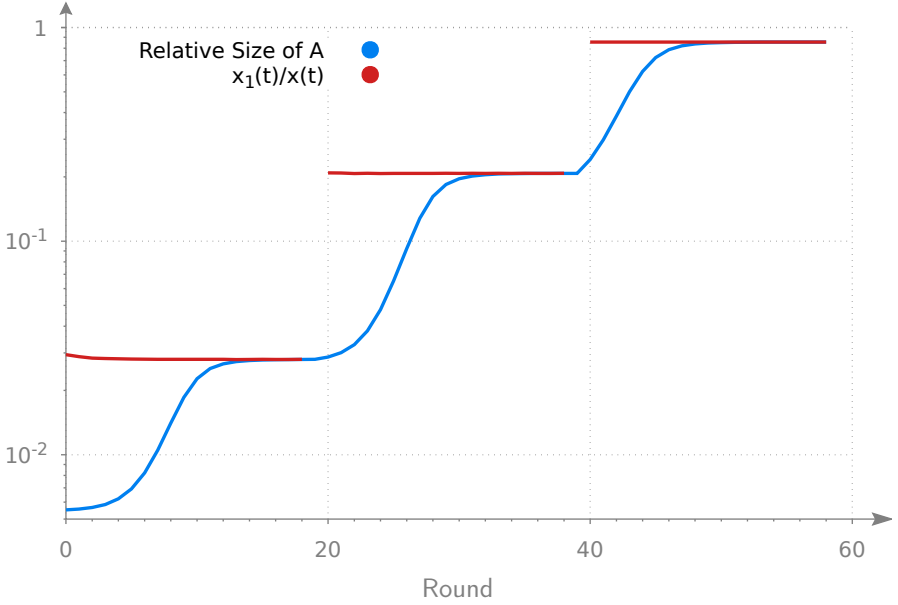
If the bias towards the initial plurality color is $z \cdot \sqrt{n \log^3 n}$ for some constant z, then the `one-bit` process converges to the plurality color within $\mathcal{O}(\log n \cdot \log k)$ rounds, with high probability.

Ghaffari and Parter, 2016

Berenbrink, Friedetzky, Giakkoupis, and Kling, 2016

One Bit of Memory

Simulation Results



Result *Elsässer, Friedetzky, K., Mallmann-Trenn, and Trinker*

After some careful modifications, the results for the `one-bit` process carry over to the asynchronous model.

Asynchronous Model

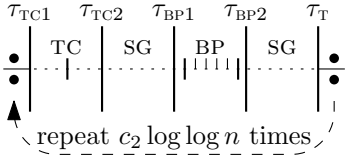
- each node is equipped with a random Poisson clock
- upon activation, nodes perform their action according to the algorithm

Result *Elsässer, Friedetzky, K., Mallmann-Trenn, and Trinker*

After some careful modifications, the results for the `one-bit` process carry over to the asynchronous model.

Asynchronous Model

- each node is equipped with a random Poisson clock
- upon activation, nodes perform their action according to the algorithm

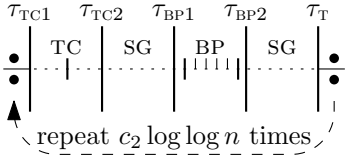


Result *Elsässer, Friedetzky, K., Mallmann-Trenn, and Trinker*

After some careful modifications, the results for the `one-bit` process carry over to the asynchronous model.

Asynchronous Model

- each node is equipped with a random Poisson clock
- upon activation, nodes perform their action according to the algorithm



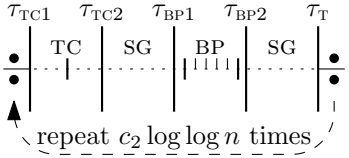
- sub-phases contain long *do-nothing*-blocks

Result *Elsässer, Friedetzky, K., Mallmann-Trenn, and Trinker*

After some careful modifications, the results for the `one-bit` process carry over to the asynchronous model.

Asynchronous Model

- each node is equipped with a random Poisson clock
- upon activation, nodes perform their action according to the algorithm



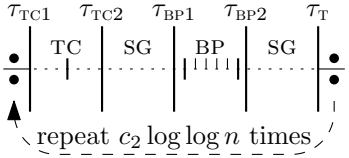
- sub-phases contain long *do-nothing*-blocks
- *Shuffle-Gadget* decouples ticks from opinions

Result *Elsässer, Friedetzky, K., Mallmann-Trenn, and Trinker*

After some careful modifications, the results for the `one-bit` process carry over to the asynchronous model.

Asynchronous Model

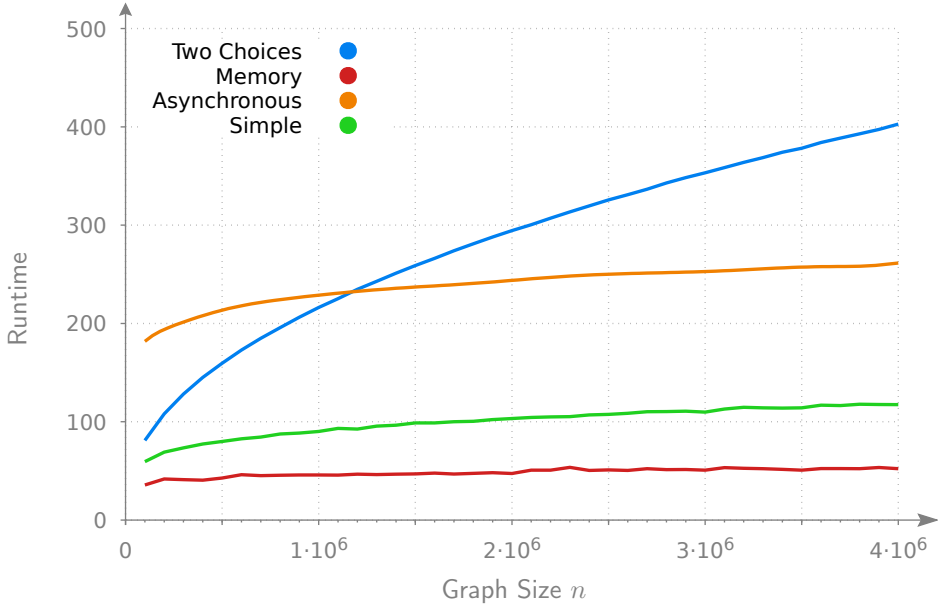
- each node is equipped with a random Poisson clock
- upon activation, nodes perform their action according to the algorithm

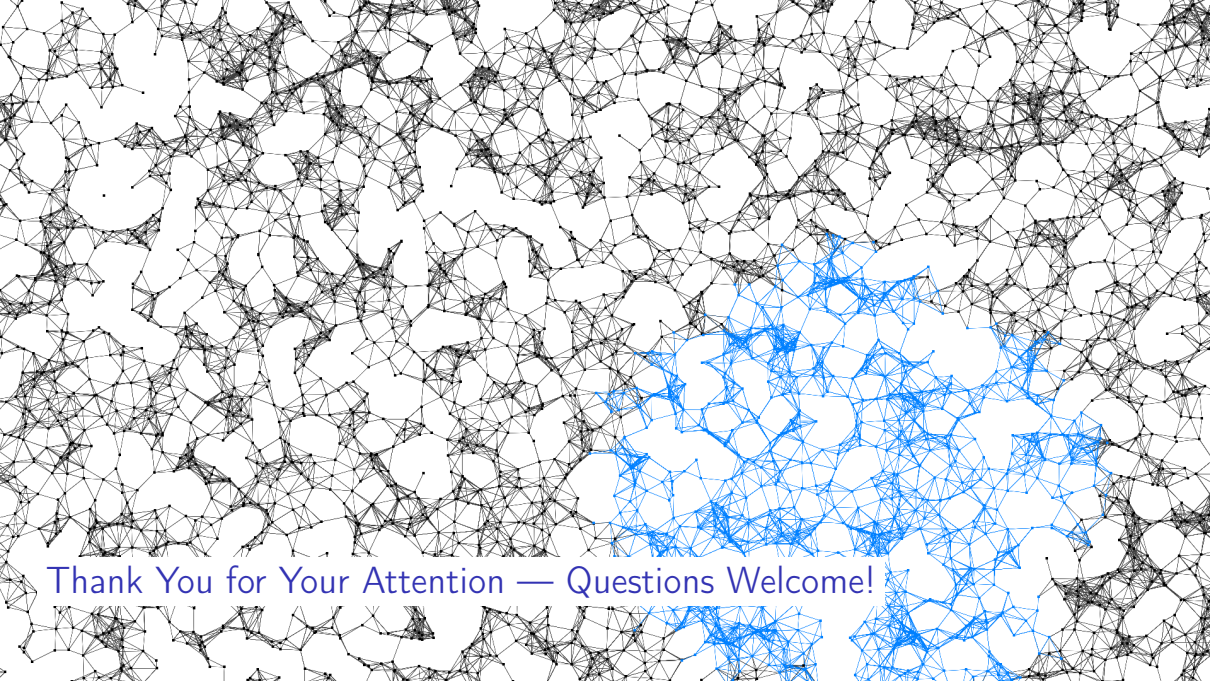


- sub-phases contain long *do-nothing*-blocks
- *Shuffle-Gadget* decouples ticks from opinions
- *Check-Synchronicity-Procedure* disables nodes which are no longer synchronous

Empirical Evaluation

Runtime Comparison




Thank You for Your Attention — Questions Welcome!