Network Constructors: A Model for Programmable Matter

Paul G. Spirakis

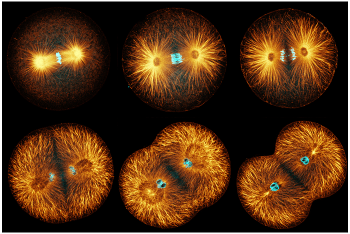
joint work with
Othon Michail

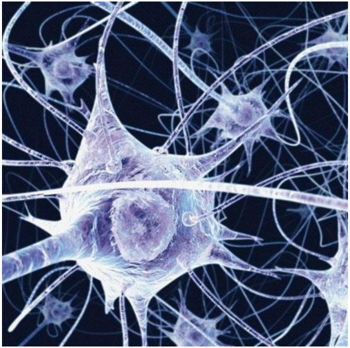
Department of Computer Science, University of Liverpool, UK
Computer Technology Institute & Press "Diophantus" (CTI)

43rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)
January 16-20, 2017
Lero-Limerick, Ireland

General Motivation

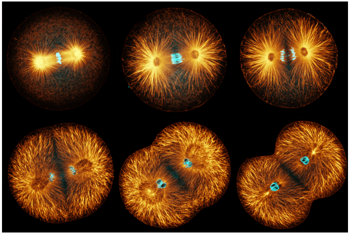
- A wide range of physical/biological systems are governed by algorithmic laws
- Usually collections of very large numbers of simple distributed entities
- Higher-level properties are the outcome of coexistence and constant interaction (cooperative and/or competing) of such entities
- Goal:
  - Reveal the algorithmic aspects of physical systems
  - Develop innovative artificial systems inspired by them

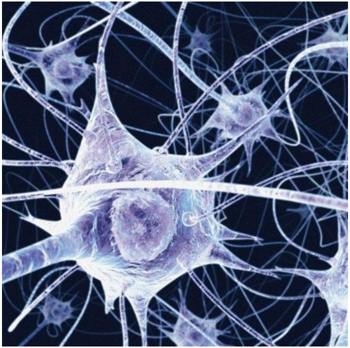




General Motivation

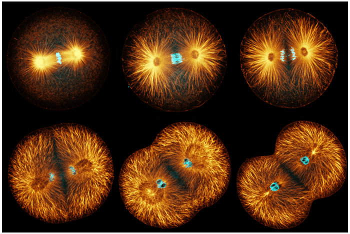
- A wide range of physical/biological systems are governed by algorithmic laws
- Usually collections of very large numbers of simple distributed entities
- Higher-level properties are the outcome of coexistence and constant interaction (cooperative and/or competing) of such entities
- Goal:
  - Reveal the algorithmic aspects of physical systems
  - Develop innovative artificial systems inspired by them

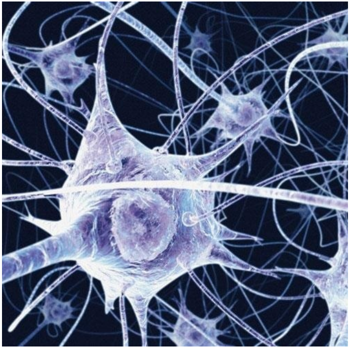




General Motivation

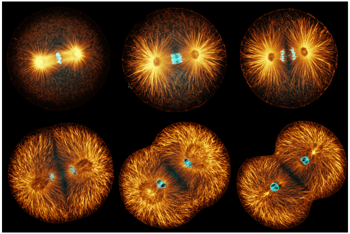
- A wide range of physical/biological systems are governed by algorithmic laws
- Usually collections of very large numbers of simple distributed entities
- Higher-level properties are the outcome of coexistence and constant interaction (cooperative and/or competing) of such entities
- Goal:
  - Reveal the algorithmic aspects of physical systems
  - Develop innovative artificial systems inspired by them

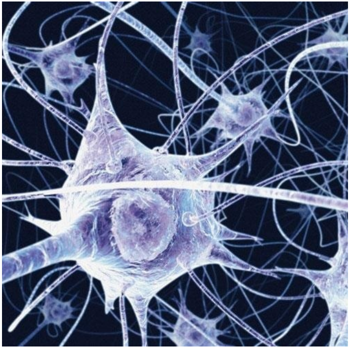




General Motivation

- A wide range of physical/biological systems are governed by algorithmic laws
- Usually collections of very large numbers of simple distributed entities
- Higher-level properties are the outcome of coexistence and constant interaction (cooperative and/or competing) of such entities
- Goal:
  - Reveal the algorithmic aspects of physical systems
  - Develop innovative artificial systems inspired by them





Some Existing Theoretical Approaches

- Cellular Automata model neural activity, self-replication, bacterial growth, ...
- Population Protocols [AADFP, PODC '04] are formally equivalent to Chemical Reaction Networks [Doty, SODA '14]
- Network Constructors [Michail, Spirakis, PODC '14; Michail, PODC '15]: abstract and simple model of distributed network formation
- Algorithmic self-assembly of DNA: DNA tiles binding to other tiles via Watson-Crick complementary sticky ends
- Models of programmable matter equipped with active mobility/ actuation mechanisms

Some Existing Theoretical Approaches

- Cellular Automata model neural activity, self-replication, bacterial growth, ...
- Population Protocols [AADFP, PODC '04] are formally equivalent to Chemical Reaction Networks [Doty, SODA '14]
- Network Constructors [Michail, Spirakis, PODC '14; Michail, PODC '15]: abstract and simple model of distributed network formation
- Algorithmic self-assembly of DNA: DNA tiles binding to other tiles via Watson-Crick complementary sticky ends
- Models of programmable matter equipped with active mobility/ actuation mechanisms

Some Existing Theoretical Approaches

- Cellular Automata model neural activity, self-replication, bacterial growth, ...
- Population Protocols [AADFP, PODC '04] are formally equivalent to Chemical Reaction Networks [Doty, SODA '14]
- Network Constructors [Michail, Spirakis, PODC '14; Michail, PODC '15]: abstract and simple model of distributed network formation
- Algorithmic self-assembly of DNA: DNA tiles binding to other tiles via Watson-Crick complementary sticky ends
- Models of programmable matter equipped with active mobility/ actuation mechanisms

Some Existing Theoretical Approaches

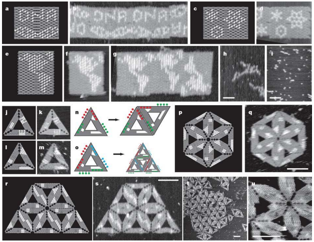
- Cellular Automata model neural activity, self-replication, bacterial growth, ...
- Population Protocols [AADFP, PODC '04] are formally equivalent to Chemical Reaction Networks [Doty, SODA '14]
- Network Constructors [Michail, Spirakis, PODC '14; Michail, PODC '15]: abstract and simple model of distributed network formation
- Algorithmic self-assembly of DNA: DNA tiles binding to other tiles via Watson-Crick complementary sticky ends
- Models of programmable matter equipped with active mobility/ actuation mechanisms

Some Existing Theoretical Approaches

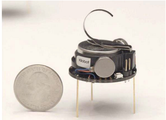
- Cellular Automata model neural activity, self-replication, bacterial growth, ...
- Population Protocols [AADFP, PODC '04] are formally equivalent to Chemical Reaction Networks [Doty, SODA '14]
- Network Constructors [Michail, Spirakis, PODC '14; Michail, PODC '15]: abstract and simple model of distributed network formation
- Algorithmic self-assembly of DNA: DNA tiles binding to other tiles via Watson-Crick complementary sticky ends
- Models of programmable matter equipped with active mobility/ actuation mechanisms

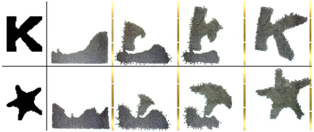
Some Existing Systems

- DNA self-assembly: single-stranded DNA molecules folded into arbitrary nanoscale shapes and patterns [Rothemund, Nature ’06]
- Kilobot [RCN, Science ’14]: programmable self-assembly of complex 2D shapes by a swarm of 1000 simple autonomous robots
- Robot Pebbles: 1cm cubic modules able to form 2D shapes through self-disassembly [GKR10]
- Millimotein: a chain which can fold itself into arbitrary 3D shapes [$\text{KCL}^{+}12$]
- Catoms [GCM05]: Nanotechnology, Intel



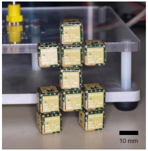
- DNA self-assembly: single-stranded DNA molecules folded into arbitrary nanoscale shapes and patterns [Rothemund, Nature '06]
- Kilobot [RCN, Science '14]: programmable self-assembly of complex 2D shapes by a swarm of 1000 simple autonomous robots
- Robot Pebbles: 1cm cubic modules able to form 2D shapes through self-disassembly [GKR10]
- Millimotein: a chain which can fold itself into arbitrary 3D shapes [KCL$^+$12]
- Catoms [GCM05]: Nanotechnology, Intel

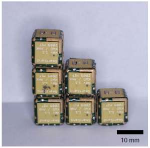




Some Existing Systems

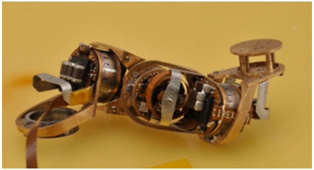
- DNA self-assembly: single-stranded DNA molecules folded into arbitrary nanoscale shapes and patterns [Rothemund, Nature ’06]
- Kilobot [RCN, Science ’14]: programmable self-assembly of complex 2D shapes by a swarm of 1000 simple autonomous robots
- Robot Pebbles: 1cm cubic modules able to form 2D shapes through self-disassembly [GKR10]
- Millimotein: a chain which can fold itself into arbitrary 3D shapes [$KCL^{+}12$]
- Catoms [GCM05]: Nanotechnology, Intel





Some Existing Systems

- DNA self-assembly: single-stranded DNA molecules folded into arbitrary nanoscale shapes and patterns [Rothemund, Nature '06]
- Kilobot [RCN, Science '14]: programmable self-assembly of complex 2D shapes by a swarm of 1000 simple autonomous robots
- Robot Pebbles: 1cm cubic modules able to form 2D shapes through self-disassembly [GKR10]
- Millimotein: a chain which can fold itself into arbitrary 3D shapes [$KCL^{+}12$]
- Catoms [GCM05]: Nanotechnology, Intel



Some Existing Systems

- DNA self-assembly: single-stranded DNA molecules folded into arbitrary nanoscale shapes and patterns [Rothemund, Nature '06]
- Kilobot [RCN, Science '14]: programmable self-assembly of complex 2D shapes by a swarm of 1000 simple autonomous robots
- Robot Pebbles: 1cm cubic modules able to form 2D shapes through self-disassembly [GKR10]
- Millimotein: a chain which can fold itself into arbitrary 3D shapes [KCL+12]
- Catoms [GCM05]: Nanotechnology, Intel

Long-term Vision: Programmable Matter

- Manipulate matter via information-theoretic and computing mechanisms and principles
- Incorporation of information to the physical world
- Plausible future outcome of progress in high-volume nanoscale assembly
- Physical realization of any computer-generated object
- Profound implications for how we think about chemistry and materials
- Materials will become user-programmed, smart, and adaptive
- It will change the way we think about engineering and manufacturing

Long-term Vision: Programmable Matter

- Manipulate matter via information-theoretic and computing mechanisms and principles
- Incorporation of information to the physical world
- Plausible future outcome of progress in high-volume nanoscale assembly
- Physical realization of any computer-generated object
- Profound implications for how we think about chemistry and materials
- Materials will become user-programmed, smart, and adaptive
- It will change the way we think about engineering and manufacturing

Long-term Vision: Programmable Matter

- Manipulate matter via information-theoretic and computing mechanisms and principles
- Incorporation of information to the physical world
- Plausible future outcome of progress in high-volume nanoscale assembly
- Physical realization of any computer-generated object
- Profound implications for how we think about chemistry and materials
- Materials will become user-programmed, smart, and adaptive
- It will change the way we think about engineering and manufacturing

Long-term Vision: Programmable Matter

- Manipulate matter via information-theoretic and computing mechanisms and principles
- Incorporation of information to the physical world
- Plausible future outcome of progress in high-volume nanoscale assembly
- Physical realization of any computer-generated object
- Profound implications for how we think about chemistry and materials
- Materials will become user-programmed, smart, and adaptive
- It will change the way we think about engineering and manufacturing

Long-term Vision: Programmable Matter

- Manipulate matter via information-theoretic and computing mechanisms and principles
- Incorporation of information to the physical world
- Plausible future outcome of progress in high-volume nanoscale assembly
- Physical realization of any computer-generated object
- Profound implications for how we think about chemistry and materials
- Materials will become user-programmed, smart, and adaptive
- It will change the way we think about engineering and manufacturing

Long-term Vision: Programmable Matter

- Manipulate matter via information-theoretic and computing mechanisms and principles
- Incorporation of information to the physical world
- Plausible future outcome of progress in high-volume nanoscale assembly
- Physical realization of any computer-generated object
- Profound implications for how we think about chemistry and materials
- Materials will become user-programmed, smart, and adaptive
- It will change the way we think about engineering and manufacturing

Long-term Vision: Programmable Matter

- Manipulate matter via information-theoretic and computing mechanisms and principles
- Incorporation of information to the physical world
- Plausible future outcome of progress in high-volume nanoscale assembly
- Physical realization of any computer-generated object
- Profound implications for how we think about chemistry and materials
- Materials will become user-programmed, smart, and adaptive
- It will change the way we think about engineering and manufacturing

Population Protocols

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

Distributed model. n computational entities, called nodes

1. interact in pairs
2. cannot control their interactions
   - passive mobility, like particles in a well-mixed solution
   - fair adversary or uniform random scheduler
3. have constant memory (uniformity)
4. do not have unique ids (anonymity)
5. $\delta : Q \times Q \rightarrow Q \times Q$: transition function

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

Distributed model. n computational entities, called nodes

1. interact in pairs
2. cannot control their interactions
   - passive mobility, like particles in a well-mixed solution
   - fair adversary or uniform random scheduler
3. have constant memory (uniformity)
4. do not have unique ids (anonymity)
5. $\delta : Q \times Q \to Q \times Q$: transition function

Population Protocols

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

Distributed model. n computational entities, called nodes

1. interact in pairs
2. cannot control their interactions
   - passive mobility, like particles in a well-mixed solution
   - fair adversary or uniform random scheduler
3. have constant memory (uniformity)
4. do not have unique ids (anonymity)
5. $\delta : Q \times Q \to Q \times Q$: transition function

Population Protocols

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

Distributed model. n computational entities, called nodes

1. interact in pairs
2. cannot control their interactions
   - passive mobility, like particles in a well-mixed solution
   - fair adversary or uniform random scheduler
3. have constant memory (uniformity)
4. do not have unique ids (anonymity)
5. $\delta : Q \times Q \to Q \times Q$: transition function

Population Protocols

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

Distributed model. n computational entities, called nodes

1. interact in pairs
2. cannot control their interactions
   - passive mobility, like particles in a well-mixed solution
   - fair adversary or uniform random scheduler
3. have constant memory (uniformity)
4. do not have unique ids (anonymity)
5. $\delta : Q \times Q \to Q \times Q$: transition function

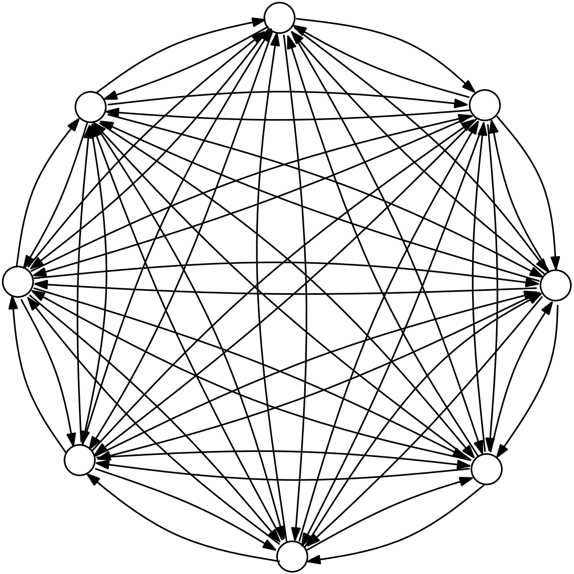
Population Protocols

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

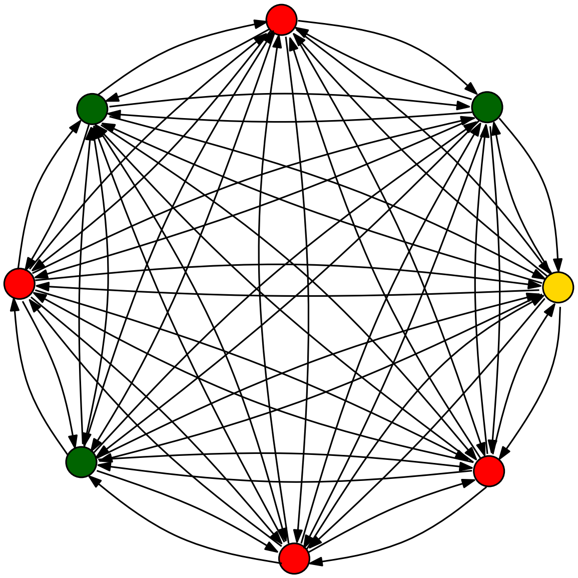
Distributed model. n computational entities, called nodes

1. interact in pairs
2. cannot control their interactions
   - passive mobility, like particles in a well-mixed solution
   - fair adversary or uniform random scheduler
3. have constant memory (uniformity)
4. do not have unique ids (anonymity)
5. $\delta : Q \times Q \rightarrow Q \times Q$: transition function

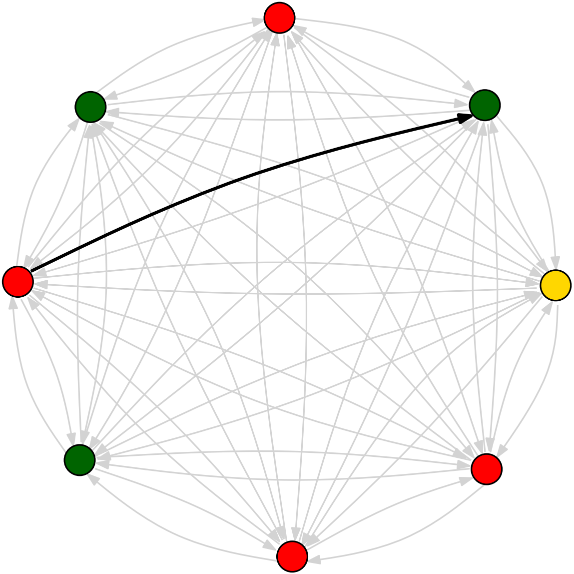
An Illustration



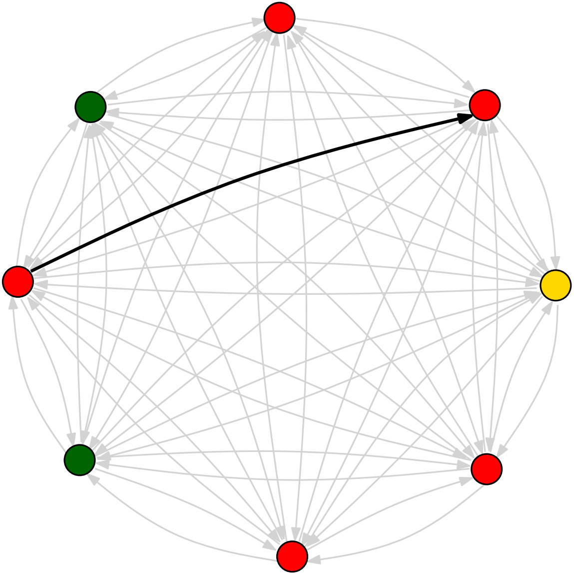
An Illustration



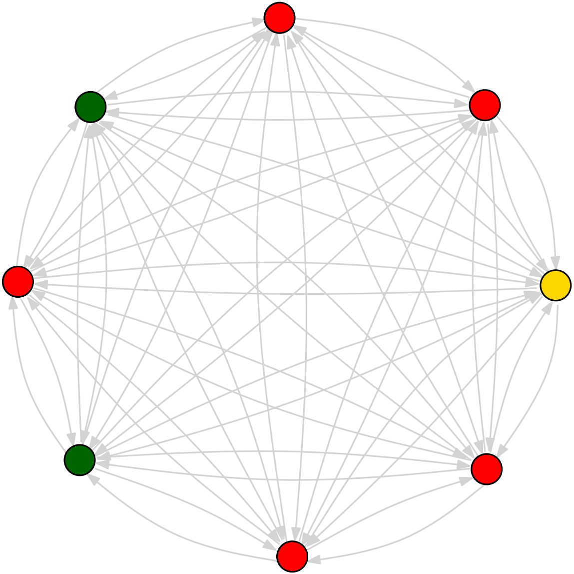
An Illustration



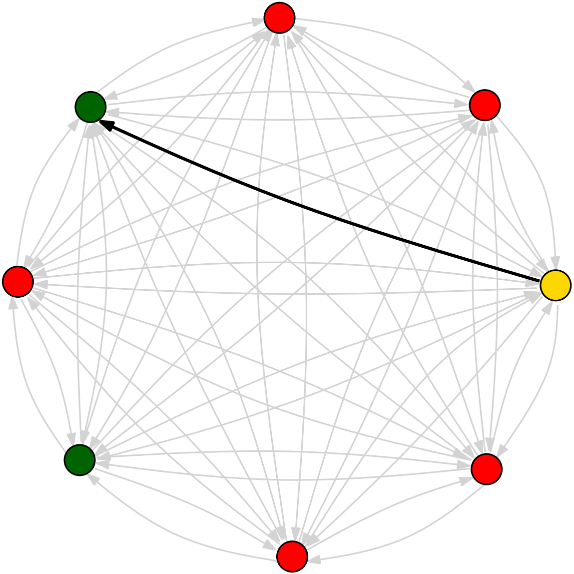
An Illustration



An Illustration



An Illustration



PPs and Natural Processes

- May be viewed as an abstraction of "fast-mixing" physical systems
- Chemical Reaction Networks
  - Finite sets of chemical reactions such as $A + B \to A + C$
  - Promising programming language for molecular control circuitry
  - Some CRNs can simulate space-bounded Turing machines
- PPs are formally equivalent to CRNs [Doty '14]
  - Consequence: bounds for PPs, usually translate to inherent properties of natural systems
  - e.g., time-consuming to generate exact quantities of molecular species quickly [DS15]
- Relations to self-assembly, gene regulatory networks, opinion spreading, and antagonism of species models
- A predicate is computable by PPs iff it is semilinear [AAER '07]

PPs and Natural Processes

- May be viewed as an abstraction of "fast-mixing" physical systems
- Chemical Reaction Networks
  - Finite sets of chemical reactions such as $A + B \rightarrow A + C$
  - Promising programming language for molecular control circuitry
  - Some CRNs can simulate space-bounded Turing machines
- PPs are formally equivalent to CRNs [Doty '14]
  - Consequence: bounds for PPs, usually translate to inherent properties of natural systems
  - e.g., time-consuming to generate exact quantities of molecular species quickly [DS15]
- Relations to self-assembly, gene regulatory networks, opinion spreading, and antagonism of species models
- A predicate is computable by PPs iff it is semilinear [AAER '07]

PPs and Natural Processes

- May be viewed as an abstraction of "fast-mixing" physical systems
- Chemical Reaction Networks
  - Finite sets of chemical reactions such as $A + B \to A + C$
  - Promising programming language for molecular control circuitry
  - Some CRNs can simulate space-bounded Turing machines
- PPs are formally equivalent to CRNs [Doty '14]
  - Consequence: bounds for PPs, usually translate to inherent properties of natural systems
  - e.g., time-consuming to generate exact quantities of molecular species quickly [DS15]
- Relations to self-assembly, gene regulatory networks, opinion spreading, and antagonism of species models
- A predicate is computable by PPs iff it is semilinear [AAER '07]

PPs and Natural Processes

- May be viewed as an abstraction of "fast-mixing" physical systems
- Chemical Reaction Networks
  - Finite sets of chemical reactions such as $A + B \to A + C$
  - Promising programming language for molecular control circuitry
  - Some CRNs can simulate space-bounded Turing machines
- PPs are formally equivalent to CRNs [Doty '14]
  - Consequence: bounds for PPs, usually translate to inherent properties of natural systems
  - e.g., time-consuming to generate exact quantities of molecular species quickly [DS15]
- Relations to self-assembly, gene regulatory networks, opinion spreading, and antagonism of species models
- A predicate is computable by PPs iff it is semilinear [AAER '07]

PPs and Natural Processes

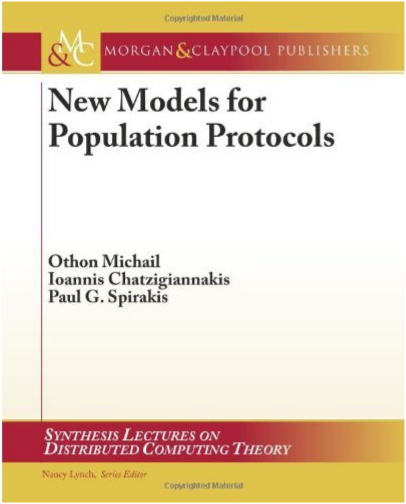
- May be viewed as an abstraction of "fast-mixing" physical systems
- Chemical Reaction Networks
  - Finite sets of chemical reactions such as $A + B \rightarrow A + C$
  - Promising programming language for molecular control circuitry
  - Some CRNs can simulate space-bounded Turing machines
- PPs are formally equivalent to CRNs [Doty '14]
  - Consequence: bounds for PPs, usually translate to inherent properties of natural systems
  - e.g., time-consuming to generate exact quantities of molecular species quickly [DS15]
- Relations to self-assembly, gene regulatory networks, opinion spreading, and antagonism of species models
- A predicate is computable by PPs iff it is semilinear [AAER '07]

PPs and Natural Processes

- May be viewed as an abstraction of "fast-mixing" physical systems
- Chemical Reaction Networks
  - Finite sets of chemical reactions such as $A + B \rightarrow A + C$
  - Promising programming language for molecular control circuitry
  - Some CRNs can simulate space-bounded Turing machines
- PPs are formally equivalent to CRNs [Doty '14]
  - Consequence: bounds for PPs, usually translate to inherent properties of natural systems
  - e.g., time-consuming to generate exact quantities of molecular species quickly [DS15]
- Relations to self-assembly, gene regulatory networks, opinion spreading, and antagonism of species models
- A predicate is computable by PPs iff it is semilinear [AAER '07]

PPs and Natural Processes

- May be viewed as an abstraction of "fast-mixing" physical systems
- Chemical Reaction Networks
  - Finite sets of chemical reactions such as $A + B \rightarrow A + C$
  - Promising programming language for molecular control circuitry
  - Some CRNs can simulate space-bounded Turing machines
- PPs are formally equivalent to CRNs [Doty '14]
  - Consequence: bounds for PPs, usually translate to inherent properties of natural systems
  - e.g., time-consuming to generate exact quantities of molecular species quickly [DS15]
- Relations to self-assembly, gene regulatory networks, opinion spreading, and antagonism of species models
- A predicate is computable by PPs iff it is semilinear [AAER '07]

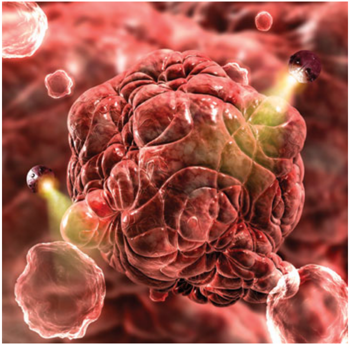

Copyrighted Material
MORGAN & CLAYPOOL PUBLISHERS
New Models for
Population Protocols
Othon Michail
Ioannis Chatzigiannakis
Paul G. Spirakis
SYNTHESIS LECTURES ON
DISTRIBUTED COMPUTING THEORY
Nancy Lynch, Series Editor
Copyrighted Material

Dynamic Environments: A Motivating Example

- n tiny computational devices
- Injected into a human circulatory system for monitoring/treating
- Move and interact passively (blood flow)
- Cooperation: can create bonds with each other
- Self-assemble into a desired global structure/network
- The artificial population evolves greater complexity, better storage capacity, and adapts and optimizes its performance to the specific task to be accomplished

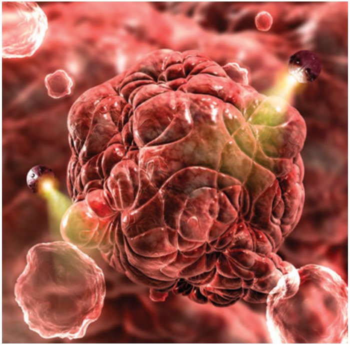
Dynamic Environments: A Motivating Example

- n tiny computational devices
- Injected into a human circulatory system for monitoring/treating
- Move and interact passively (blood flow)
- Cooperation: can create bonds with each other
- Self-assemble into a desired global structure/network
- The artificial population evolves greater complexity, better storage capacity, and adapts and optimizes its performance to the specific task to be accomplished



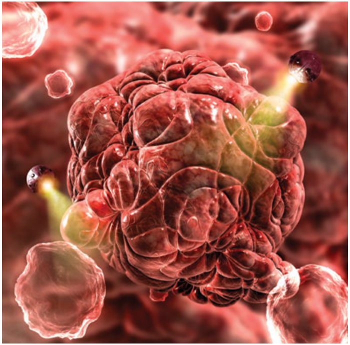
Dynamic Environments: A Motivating Example

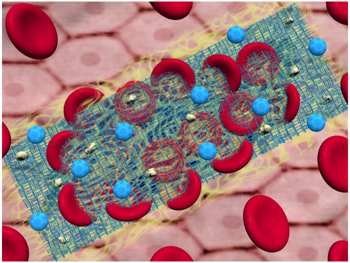
- *n* tiny computational devices
- Injected into a human circulatory system for monitoring/treating
- Move and interact passively (blood flow)
- Cooperation: can create bonds with each other
- Self-assemble into a desired global structure/network
- The artificial population evolves greater complexity, better storage capacity, and adapts and optimizes its performance to the specific task to be accomplished



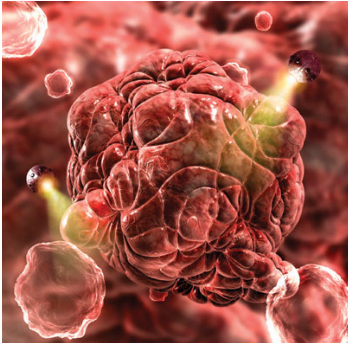
Dynamic Environments: A Motivating Example

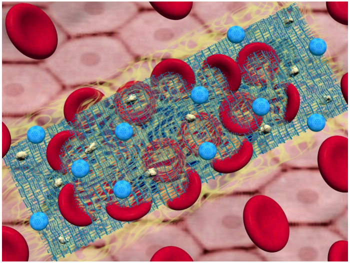
- n tiny computational devices
- Injected into a human circulatory system for monitoring/treating
- Move and interact passively (blood flow)
- Cooperation: can create bonds with each other
- Self-assemble into a desired global structure/network
- The artificial population evolves greater complexity, better storage capacity, and adapts and optimizes its performance to the specific task to be accomplished



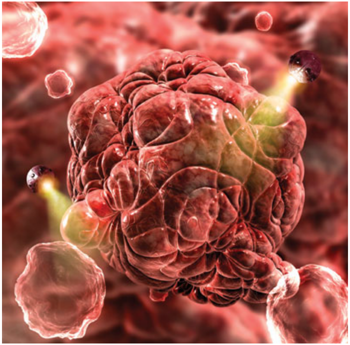


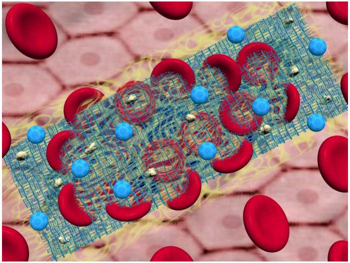
- *n* tiny computational devices
- Injected into a human circulatory system for monitoring/treating
- Move and interact passively (blood flow)
- Cooperation: can create bonds with each other
- Self-assemble into a desired global structure/network
- The artificial population evolves greater complexity, better storage capacity, and adapts and optimizes its performance to the specific task to be accomplished





- n tiny computational devices
- Injected into a human circulatory system for monitoring/treating
- Move and interact passively (blood flow)
- Cooperation: can create bonds with each other
- Self-assemble into a desired global structure/network
- The artificial population evolves greater complexity, better storage capacity, and adapts and optimizes its performance to the specific task to be accomplished





Distributed Network Construction

[Michail and Spirakis, PODC '14 and Distrib. Comput. '16]

Fundamental problem: Algorithmic distributed construction of an actual communication topology

- Processes can form/delete connections between them
- Physical or virtual connections depending on the application
- on/off case: a connection either exists (active) or does not exist (inactive)
- Initially all connections are inactive
- Goal: End up with a desired stable network

Distributed Network Construction

[Michail and Spirakis, PODC ’14 and Distrib. Comput. ’16]

Fundamental problem: Algorithmic distributed construction of an actual communication topology

- Processes can form/delete connections between them
- Physical or virtual connections depending on the application
- on/off case: a connection either exists (active) or does not exist (inactive)
- Initially all connections are inactive
- Goal: End up with a desired stable network

Distributed Network Construction

[Michail and Spirakis, PODC '14 and Distrib. Comput. '16]

Fundamental problem: Algorithmic distributed construction of an actual communication topology

- Processes can form/delete connections between them
- Physical or virtual connections depending on the application
- on/off case: a connection either exists (active) or does not exist (inactive)
- Initially all connections are inactive
- Goal: End up with a desired stable network

Distributed Network Construction

[Michail and Spirakis, PODC '14 and Distrib. Comput. '16]

Fundamental problem: Algorithmic distributed construction of an actual communication topology

- Processes can form/delete connections between them
- Physical or virtual connections depending on the application
- on/off case: a connection either exists (active) or does not exist (inactive)
- Initially all connections are inactive
- Goal: End up with a desired stable network

Distributed Network Construction

[Michail and Spirakis, PODC '14 and Distrib. Comput. '16]

Fundamental problem: Algorithmic distributed construction of an actual communication topology

- Processes can form/delete connections between them
- Physical or virtual connections depending on the application
- on/off case: a connection either exists (active) or does not exist (inactive)
- Initially all connections are inactive
- Goal: End up with a desired stable network

[Michail and Spirakis, PODC '14 and Distrib. Comput. '16]

Fundamental problem: Algorithmic distributed construction of an actual communication topology

- Processes can form/delete connections between them
- Physical or virtual connections depending on the application
- on/off case: a connection either exists (active) or does not exist (inactive)
- Initially all connections are inactive
- Goal: End up with a desired stable network

A Minimal Model

- Processes are finite automata
- Homogeneity: All begin from the same initial state and execute the same finite program
- Adversarial environment: Fair adversary scheduler choosing pairwise interactions (a la population protocols)
- Uniform random scheduler to estimate efficiency
- Complex global behaviour via simple, uniform, anonymous, homogeneous, and cooperative entities

A Minimal Model

- Processes are finite automata
- Homogeneity: All begin from the same initial state and execute the same finite program
- Adversarial environment: Fair adversary scheduler choosing pairwise interactions (a la population protocols)
- Uniform random scheduler to estimate efficiency
- Complex global behaviour via simple, uniform, anonymous, homogeneous, and cooperative entities

A Minimal Model

- Processes are finite automata
- Homogeneity: All begin from the same initial state and execute the same finite program
- Adversarial environment: Fair adversary scheduler choosing pairwise interactions (a la population protocols)
- Uniform random scheduler to estimate efficiency
- Complex global behaviour via simple, uniform, anonymous, homogeneous, and cooperative entities

A Minimal Model

- Processes are finite automata
- Homogeneity: All begin from the same initial state and execute the same finite program
- Adversarial environment: Fair adversary scheduler choosing pairwise interactions (a la population protocols)
- Uniform random scheduler to estimate efficiency
- Complex global behaviour via simple, uniform, anonymous, homogeneous, and cooperative entities

A Minimal Model

- Processes are finite automata
- Homogeneity: All begin from the same initial state and execute the same finite program
- Adversarial environment: Fair adversary scheduler choosing pairwise interactions (a la population protocols)
- Uniform random scheduler to estimate efficiency
- Complex global behaviour via simple, uniform, anonymous, homogeneous, and cooperative entities

Related Work

- PPs/CRNs can only capture the dynamics of molecular counts and not of structure formation
- We aim to capture the stable structures that may occur in a well-mixed solution

- PPs/CRNs can only capture the dynamics of molecular counts and not of structure formation
- We aim to capture the stable structures that may occur in a well-mixed solution

- PPs/CRNs can only capture the dynamics of molecular counts and not of structure formation
- We aim to capture the stable structures that may occur in a well-mixed solution

Our Goal: Determine what stable structures can result in such systems, how fast, and under what conditions (e.g. by what underlying codes/reaction-rules)

Applications

- Principles of algorithmic network formation
- Programmable matter
- Reconfigurable robotics
- Medicine (diagnosis/treatment)
- Smart material (that self-built, self-adjust, ...)
- Biomaterial manufacture
- Modeling and understanding network formation by physical/chemical/biological processes (e.g. molecules reacting in a well-mixed solution)

Applications

- Principles of algorithmic network formation
- Programmable matter
- Reconfigurable robotics
- Medicine (diagnosis/treatment)
- Smart material (that self-built, self-adjust, ...)
- Biomaterial manufacture
- Modeling and understanding network formation by physical/chemical/biological processes (e.g. molecules reacting in a well-mixed solution)

Applications

- Principles of algorithmic network formation
- Programmable matter
- Reconfigurable robotics
- Medicine (diagnosis/treatment)
- Smart material (that self-built, self-adjust, ...)
- Biomaterial manufacture
- Modeling and understanding network formation by physical/chemical/biological processes (e.g. molecules reacting in a well-mixed solution)

Applications

- Principles of algorithmic network formation
- Programmable matter
- Reconfigurable robotics
- Medicine (diagnosis/treatment)
- Smart material (that self-built, self-adjust, ...)
- Biomaterial manufacture
- Modeling and understanding network formation by physical/chemical/biological processes (e.g. molecules reacting in a well-mixed solution)

Applications

- Principles of algorithmic network formation
- Programmable matter
- Reconfigurable robotics
- Medicine (diagnosis/treatment)
- Smart material (that self-built, self-adjust, ...)
- Biomaterial manufacture
- Modeling and understanding network formation by physical/chemical/biological processes (e.g. molecules reacting in a well-mixed solution)

Applications

- Principles of algorithmic network formation
- Programmable matter
- Reconfigurable robotics
- Medicine (diagnosis/treatment)
- Smart material (that self-built, self-adjust, ...)
- Biomaterial manufacture
- Modeling and understanding network formation by physical/chemical/biological processes (e.g. molecules reacting in a well-mixed solution)

Applications

- Principles of algorithmic network formation
- Programmable matter
- Reconfigurable robotics
- Medicine (diagnosis/treatment)
- Smart material (that self-built, self-adjust, ...)
- Biomaterial manufacture
- Modeling and understanding network formation by physical/chemical/biological processes (e.g. molecules reacting in a well-mixed solution)

The Model of Network Constructors

1. Q: finite set of node-states,
2. $q_0 \in Q$: initial node-state,
3. $Q_{out} \subseteq Q$: set of output node-states, and
4. $\delta : Q \times Q \times \{0,1\} \rightarrow Q \times Q \times \{0,1\}$: the transition function

- In every step, a pair uv is selected by the scheduler and u, v interact according to δ
- Fair scheduler: If a configuration is reachable infinitely often then it is eventually reached
- Output network: nodes that are in output states and edges between them that are active
- Stability: The output network cannot change in future steps

The Model of Network Constructors

1. Q: finite set of node-states,
2. $q_0 \in Q$: initial node-state,
3. $Q_{out} \subseteq Q$: set of output node-states, and
4. $\delta : Q \times Q \times \{0,1\} \rightarrow Q \times Q \times \{0,1\}$: the transition function

- In every step, a pair uv is selected by the scheduler and u, v interact according to δ
- Fair scheduler: If a configuration is reachable infinitely often then it is eventually reached
- Output network: nodes that are in output states and edges between them that are active
- Stability: The output network cannot change in future steps

The Model of Network Constructors

1. Q: finite set of node-states,
2. $q_0 \in Q$: initial node-state,
3. $Q_{out} \subseteq Q$: set of output node-states, and
4. $\delta : Q \times Q \times \{0,1\} \to Q \times Q \times \{0,1\}$: the transition function

- In every step, a pair uv is selected by the scheduler and u, v interact according to δ
- Fair scheduler: If a configuration is reachable infinitely often then it is eventually reached
- Output network: nodes that are in output states and edges between them that are active
- Stability: The output network cannot change in future steps

The Model of Network Constructors

1. Q: finite set of node-states,
2. $q_0 \in Q$: initial node-state,
3. $Q_{out} \subseteq Q$: set of output node-states, and
4. $\delta : Q \times Q \times \{0,1\} \to Q \times Q \times \{0,1\}$: the transition function

- In every step, a pair uv is selected by the scheduler and u, v interact according to δ
- Fair scheduler: If a configuration is reachable infinitely often then it is eventually reached
- Output network: nodes that are in output states and edges between them that are active
- Stability: The output network cannot change in future steps

The Model of Network Constructors

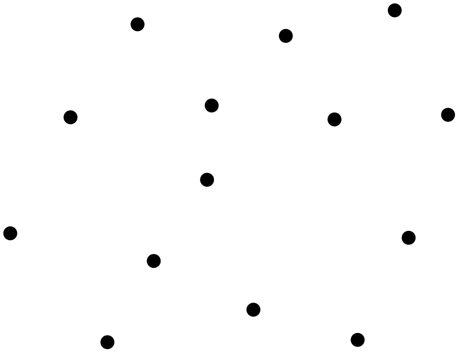
1. Q: finite set of node-states,
2. $q_0 \in Q$: initial node-state,
3. $Q_{out} \subseteq Q$: set of output node-states, and
4. $\delta : Q \times Q \times \{0,1\} \to Q \times Q \times \{0,1\}$: the transition function

- In every step, a pair uv is selected by the scheduler and u, v interact according to δ
- Fair scheduler: If a configuration is reachable infinitely often then it is eventually reached
- Output network: nodes that are in output states and edges between them that are active
- Stability: The output network cannot change in future steps

An Illustration: Spanning Star

- 2 states: black and red
- Initially all black
- Construct a global star
- Program:

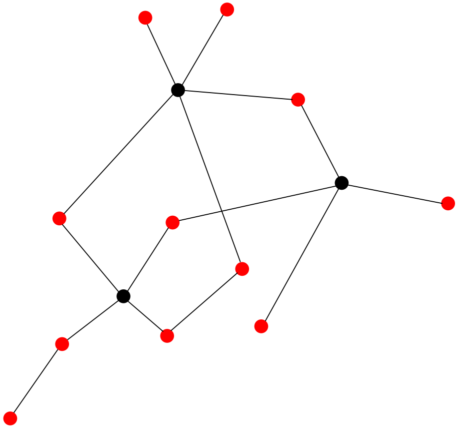
$$(b, b, 0) \rightarrow (b, r, 1)$$
$$(r, r, 1) \rightarrow (r, r, 0)$$
$$(b, r, 0) \rightarrow (b, r, 1)$$



An Illustration: Spanning Star

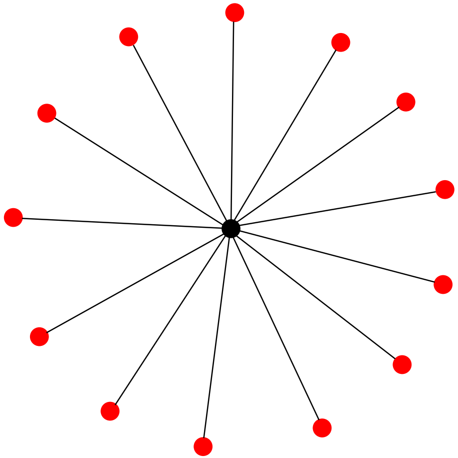
- 2 states: black and red
- Initially all black
- Construct a global star
- Program:

$$(b, b, 0) \rightarrow (b, r, 1)$$
$$(r, r, 1) \rightarrow (r, r, 0)$$
$$(b, r, 0) \rightarrow (b, r, 1)$$



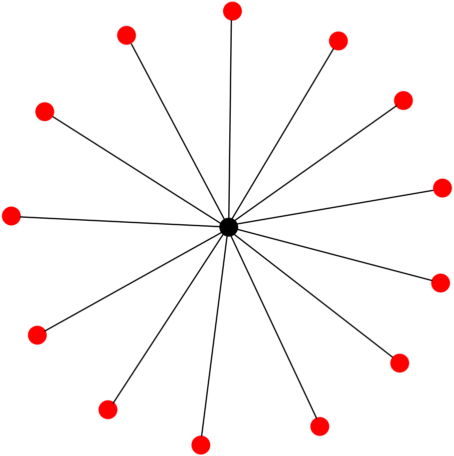
- 2 states: black and red
- Initially all black
- Construct a global star
- Program:

$$(b, b, 0) \rightarrow (b, r, 1)$$
$$(r, r, 1) \rightarrow (r, r, 0)$$
$$(b, r, 0) \rightarrow (b, r, 1)$$



An Illustration: Spanning Star

- Size: 2 states
- Time: $O(n^2 \log n)$
- Optimal w.r.t. both
- Time = # interactions (sequencial)
  - Parallel time is on the average $\frac{1}{n} \cdot$ sequential time



Global Line

Global Line: For all n, the n processes must construct a spanning line

Theorem (Generic Lower Bound)

The expected time to convergence (always under the uniform random scheduler) of any protocol that constructs a spanning network is $\Omega(n \log n)$.

Proof.

Consider the time at which the last edge is activated. By that time, all nodes have some active edge incident to them, thus every node has interacted at least once. The latter can be shown to require an expected number of $\Theta(n \log n)$ steps. □

Theorem (Line Lower Bound)

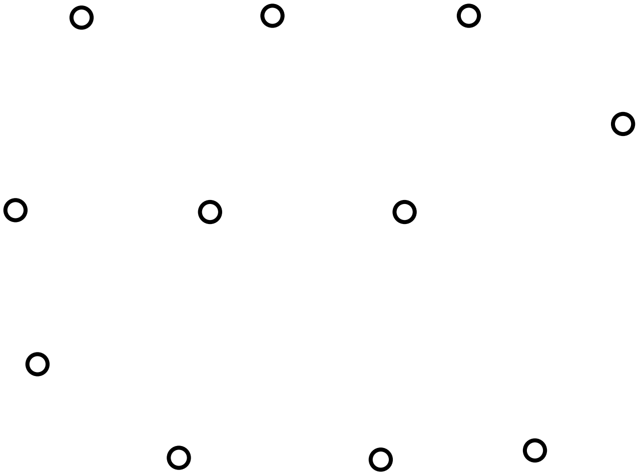
The expected time to convergence of any protocol that constructs a spanning line is $\Omega(n^2)$.

- Protocol *Simple-Global-Line*:

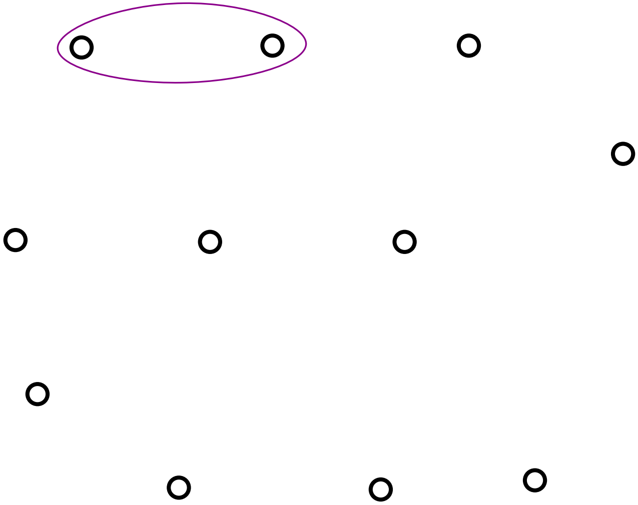
$$(q_0, q_0, 0) \rightarrow (q_1, l, 1)$$
$$(l, q_0, 0) \rightarrow (q_2, l, 1)$$
$$(l, l, 0) \rightarrow (q_2, w, 1)$$
$$(w, q_2, 1) \rightarrow (q_2, w, 1)$$
$$(w, q_1, 1) \rightarrow (q_2, l, 1)$$

  - Every node remembers its degree (q_0, q_1, q_2)
  - Every line has a unique leader (endpoint: l, internal: w)
  - Lines expand towards isolated nodes and merge to other lines via l
  - After merging, w performs a random walk to reach an endpoint

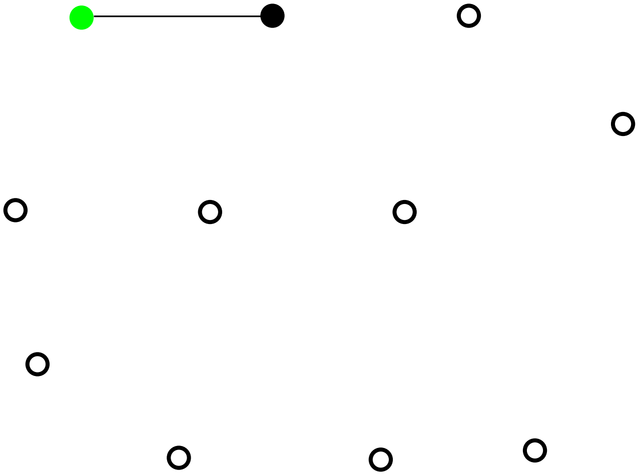
Simple-Global-Line Protocol



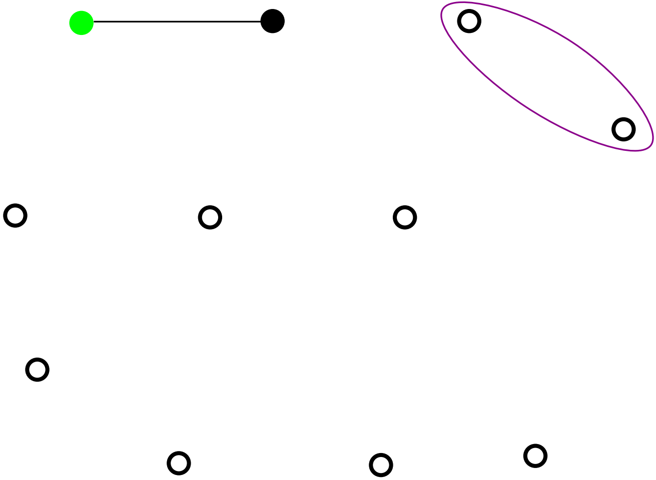
Simple-Global-Line Protocol



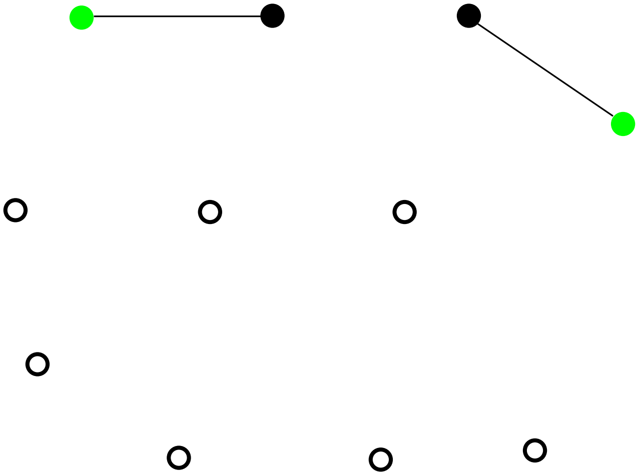
Simple-Global-Line Protocol

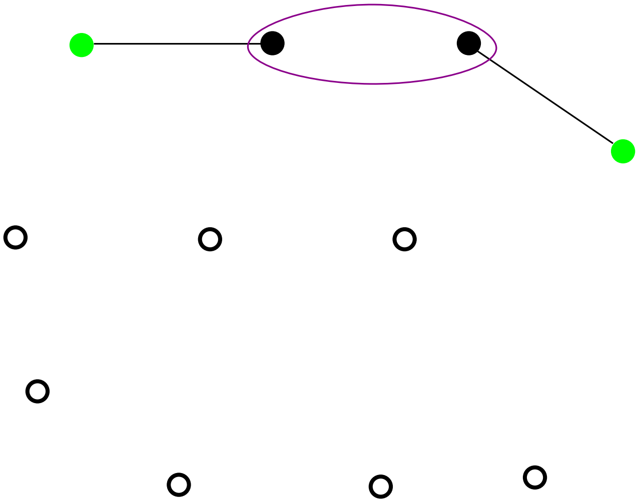


Simple-Global-Line Protocol

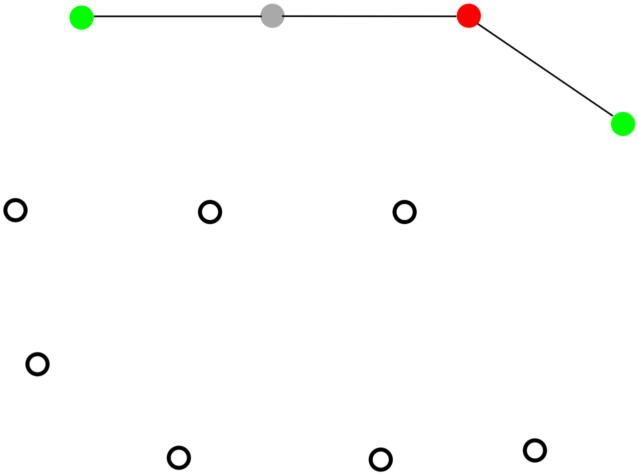


Simple-Global-Line Protocol

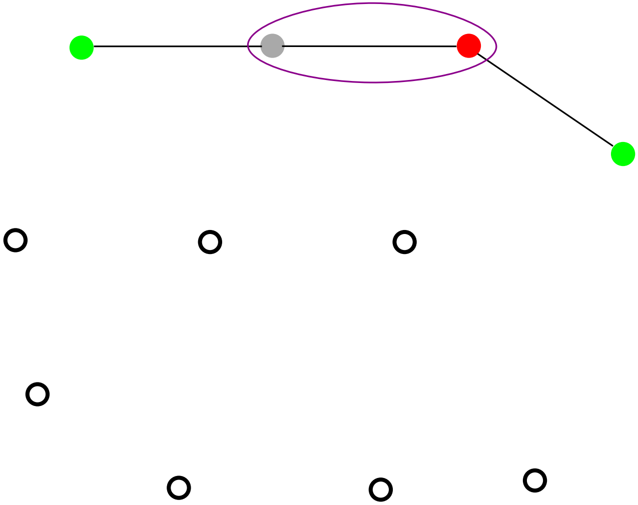




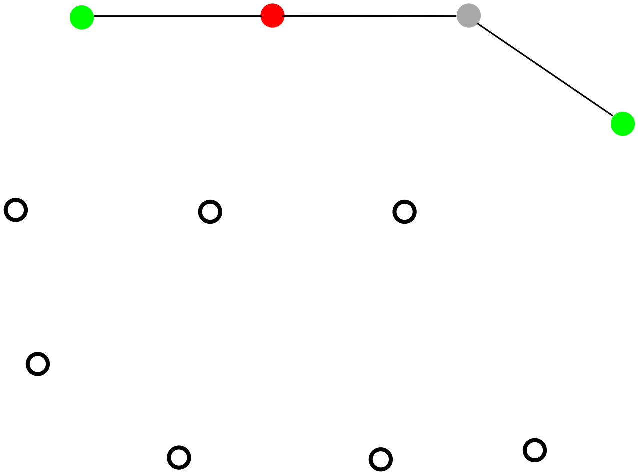
Simple-Global-Line Protocol



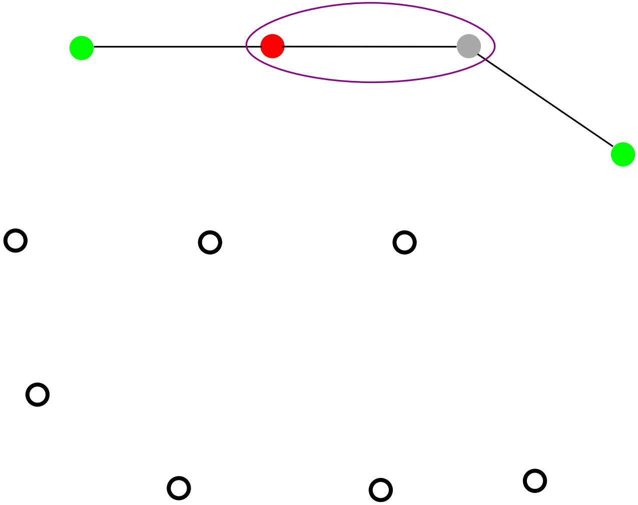
Simple-Global-Line Protocol



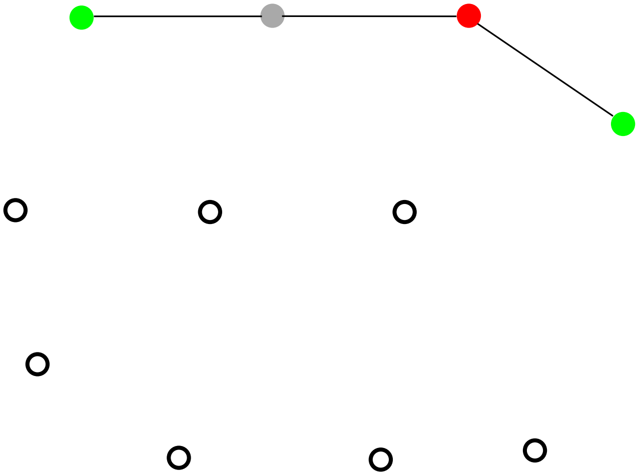
Simple-Global-Line Protocol



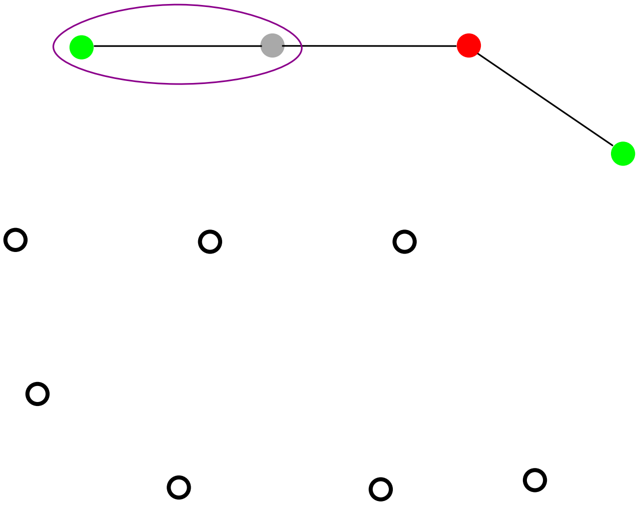
Simple-Global-Line Protocol



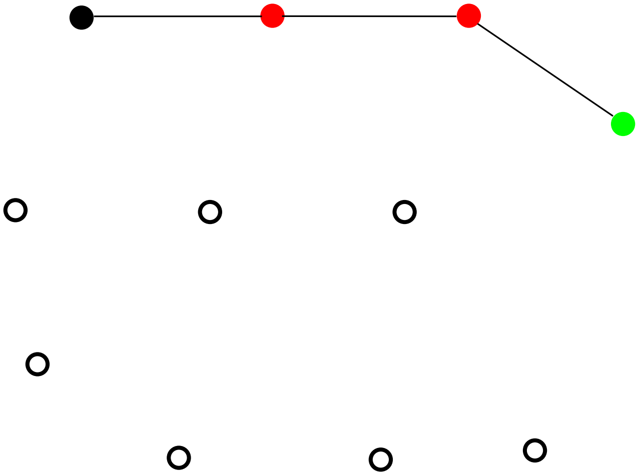
Simple-Global-Line Protocol

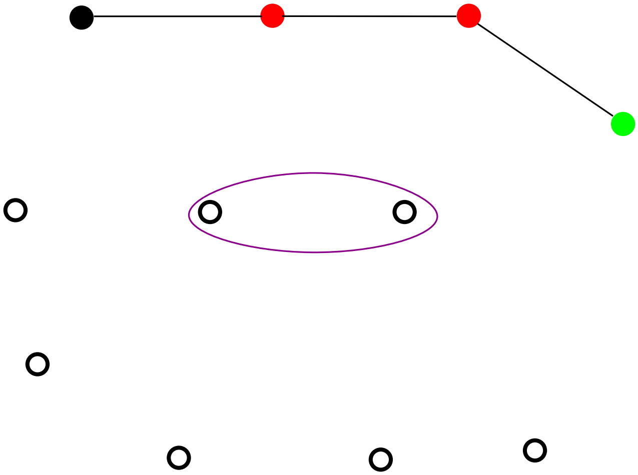


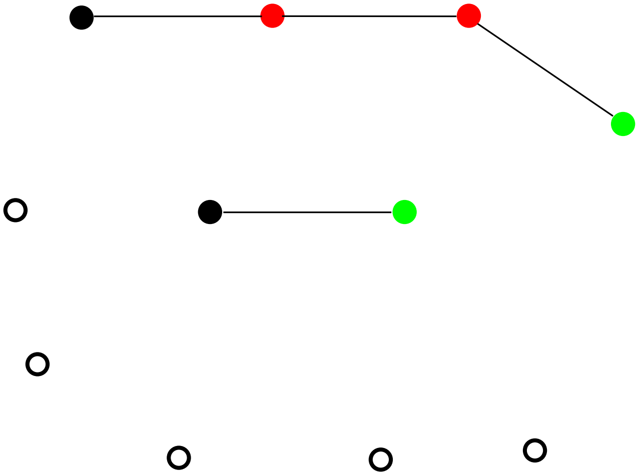
Simple-Global-Line Protocol



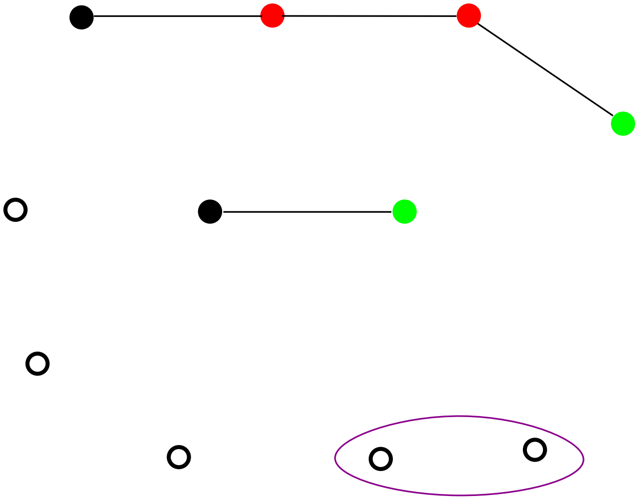
Simple-Global-Line Protocol



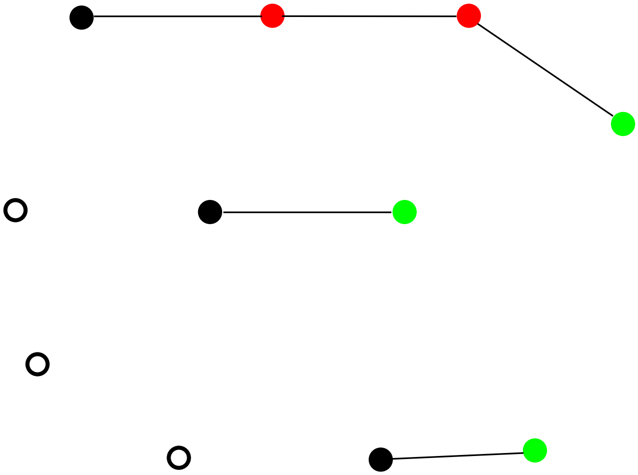




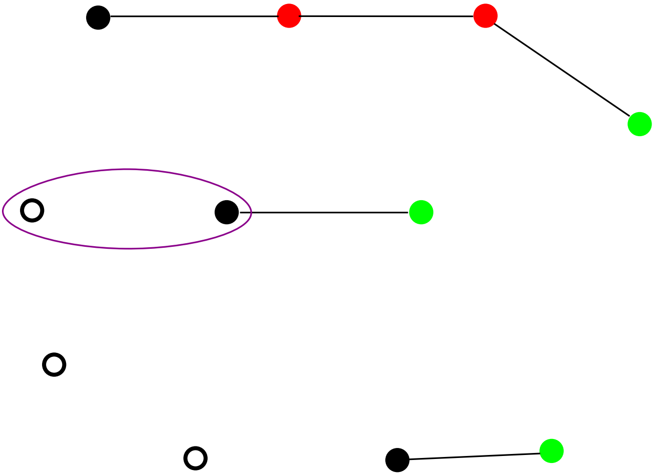
Simple-Global-Line Protocol



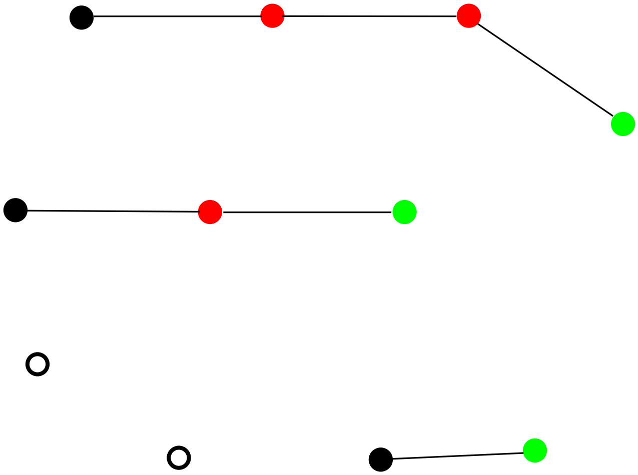
Simple-Global-Line Protocol



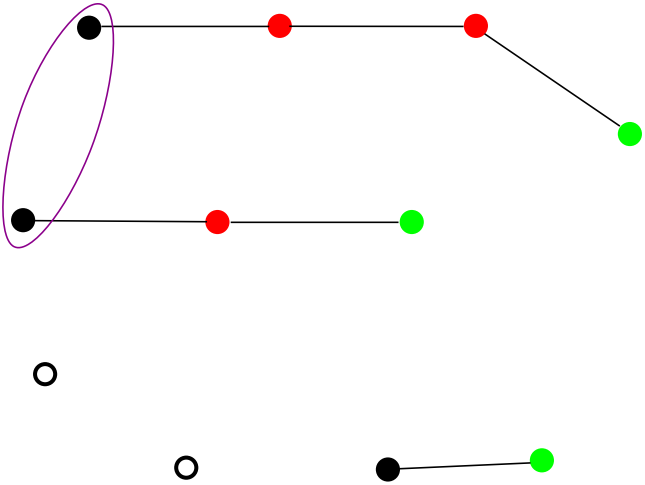
Simple-Global-Line Protocol



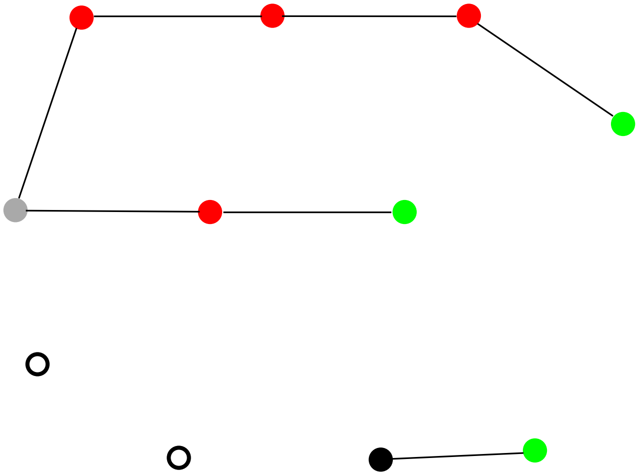
Simple-Global-Line Protocol



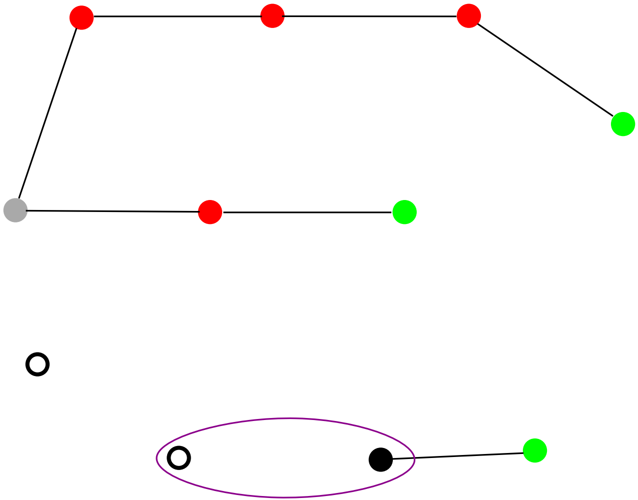
Simple-Global-Line Protocol

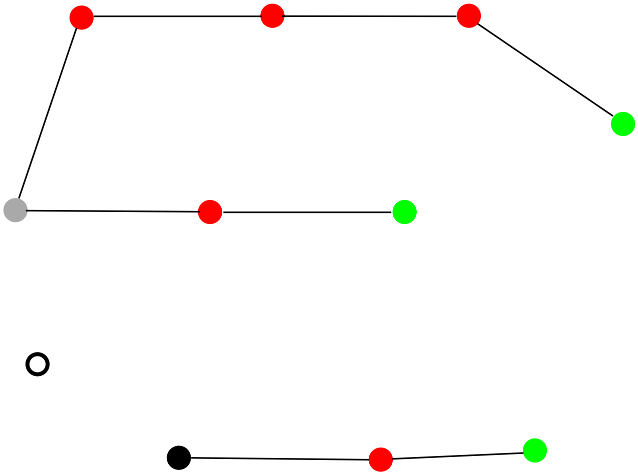


Simple-Global-Line Protocol

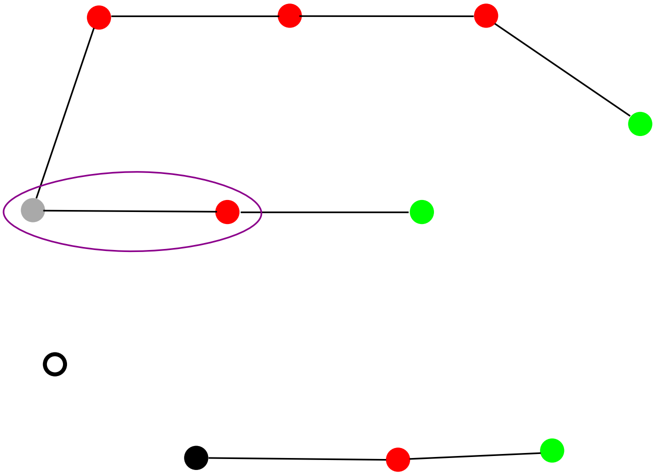


Simple-Global-Line Protocol

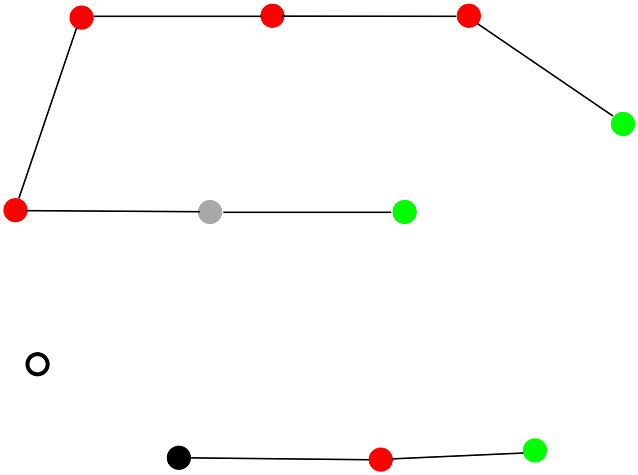




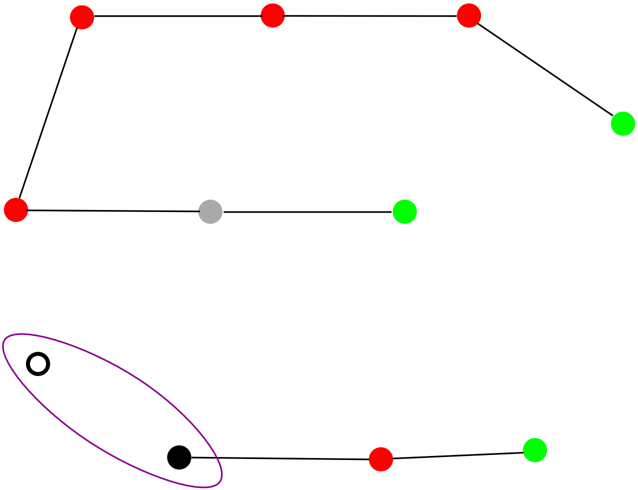
Simple-Global-Line Protocol

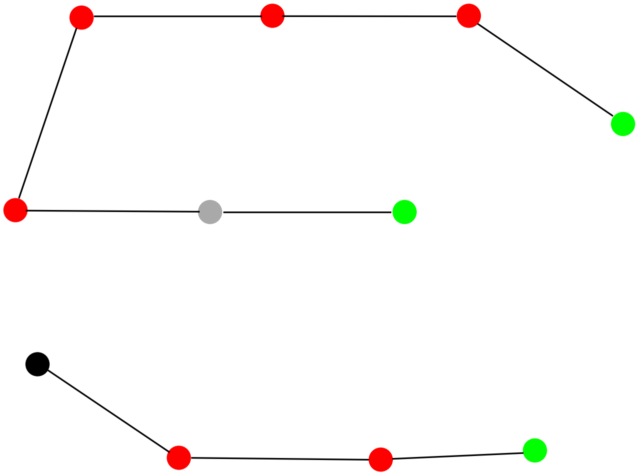


Simple-Global-Line Protocol

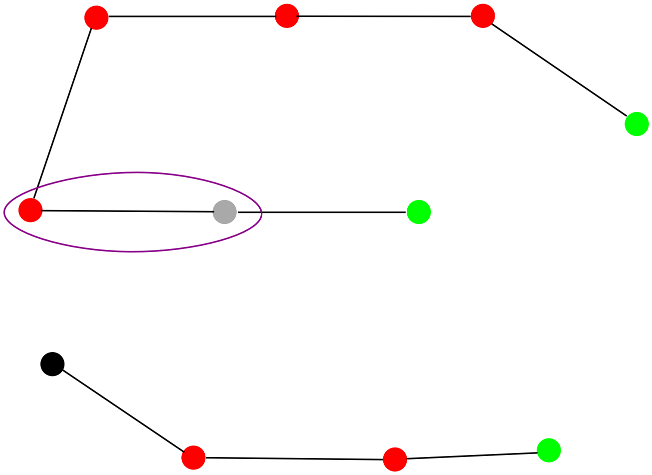


Simple-Global-Line Protocol

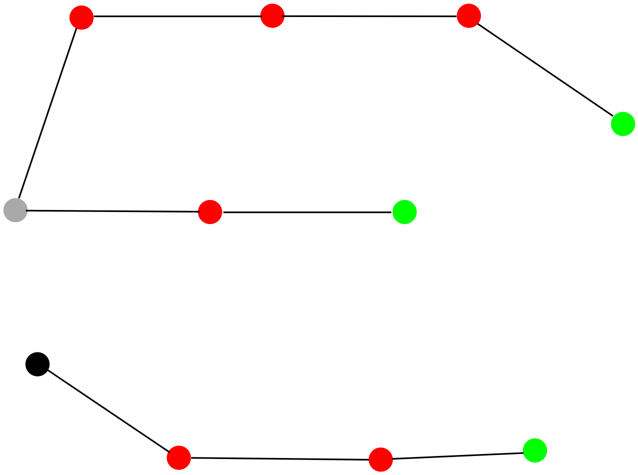




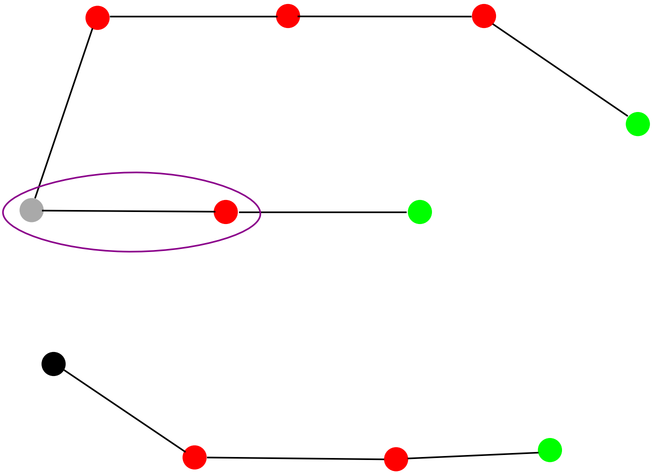
Simple-Global-Line Protocol



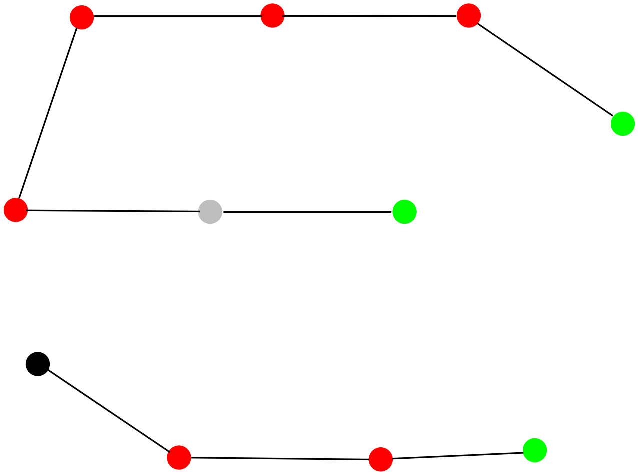
Simple-Global-Line Protocol

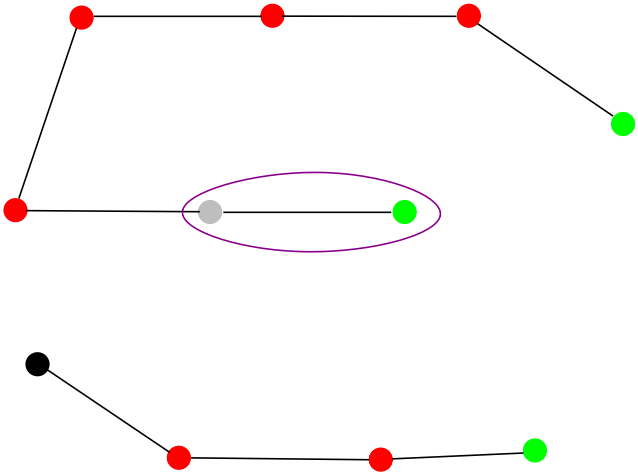


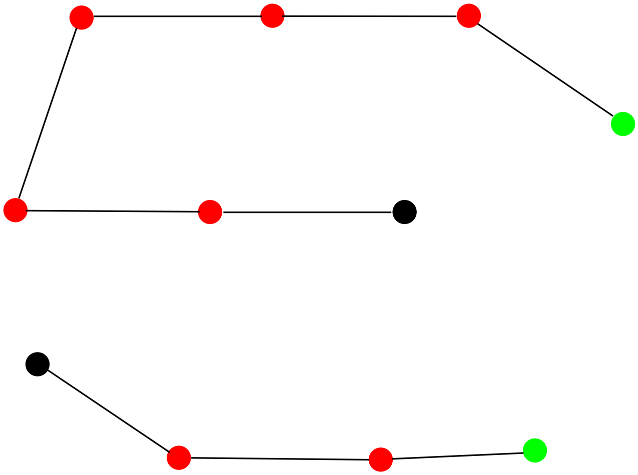
Simple-Global-Line Protocol

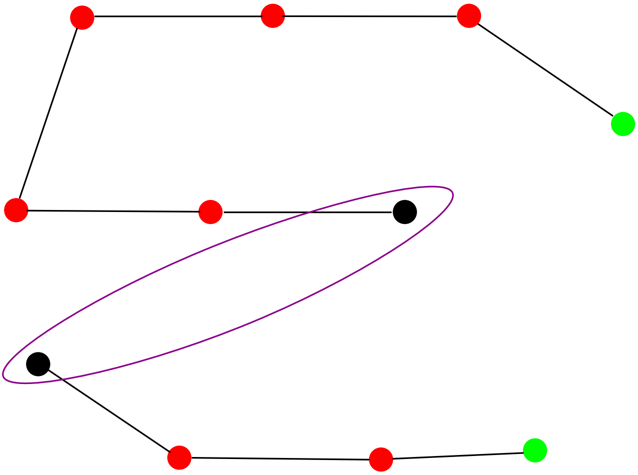


Simple-Global-Line Protocol

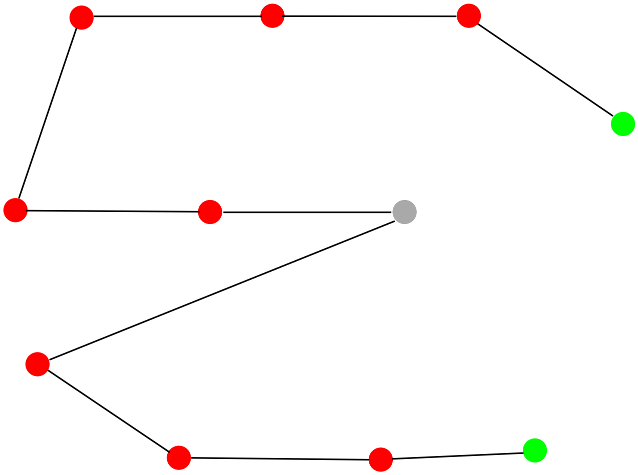




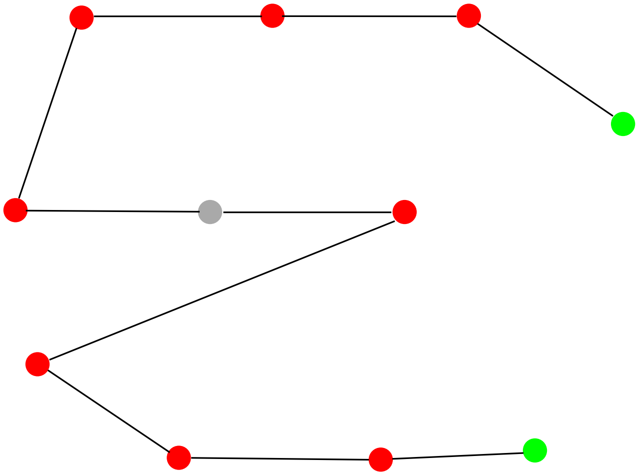




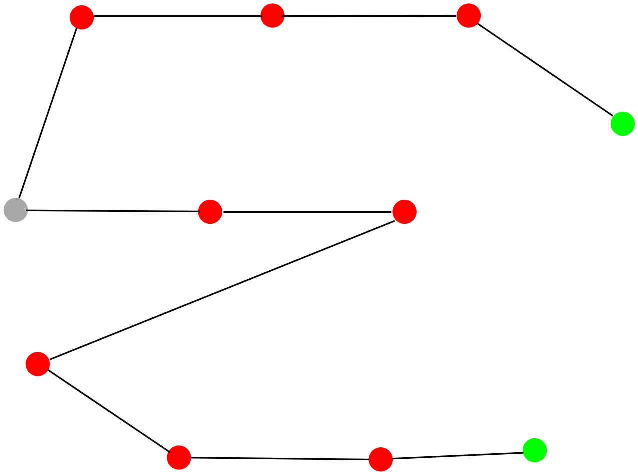
Simple-Global-Line Protocol

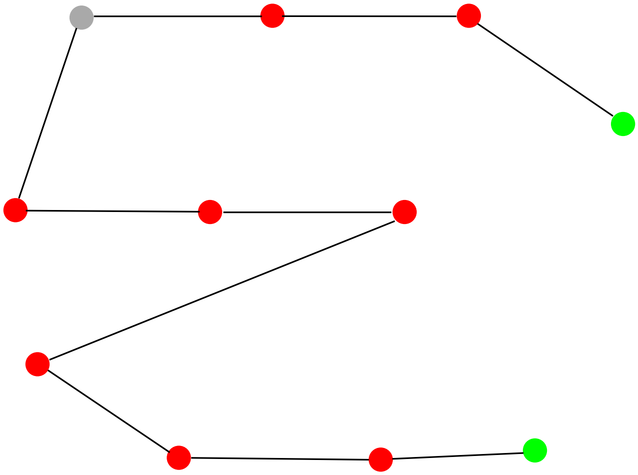


Simple-Global-Line Protocol

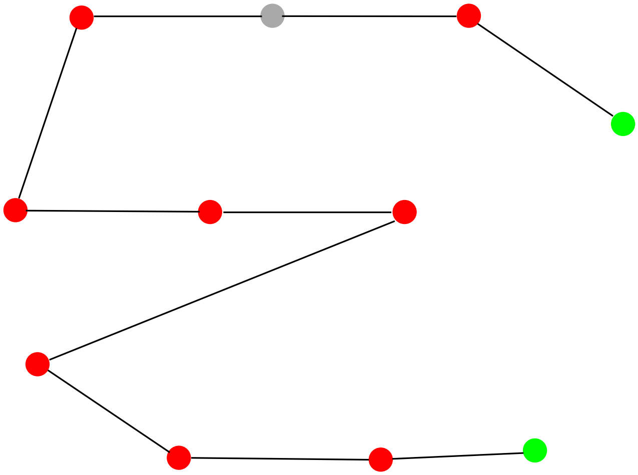


Simple-Global-Line Protocol

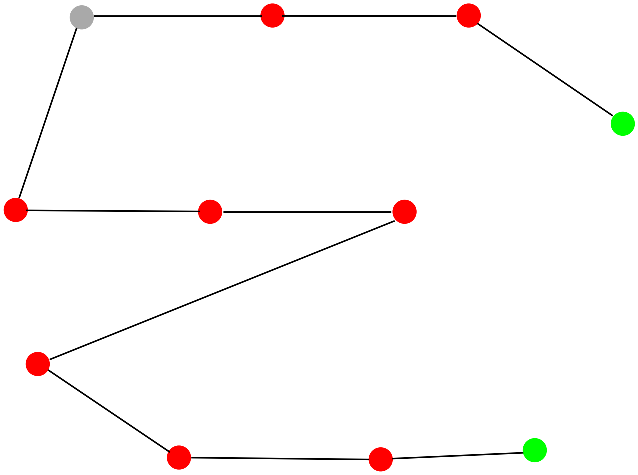




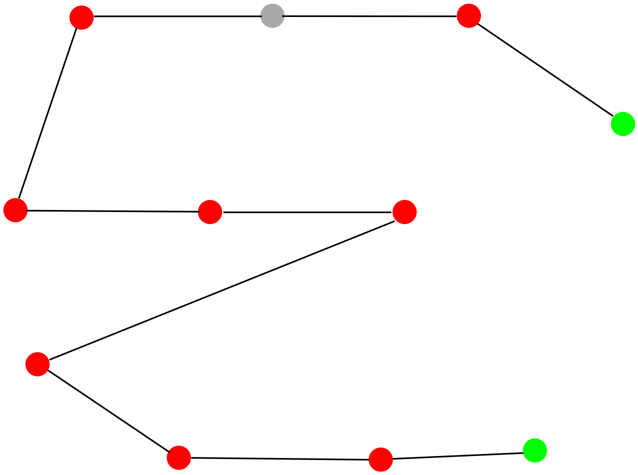
Simple-Global-Line Protocol



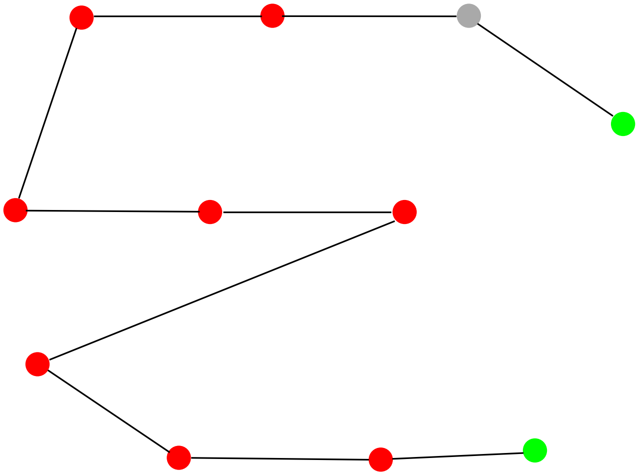
Simple-Global-Line Protocol



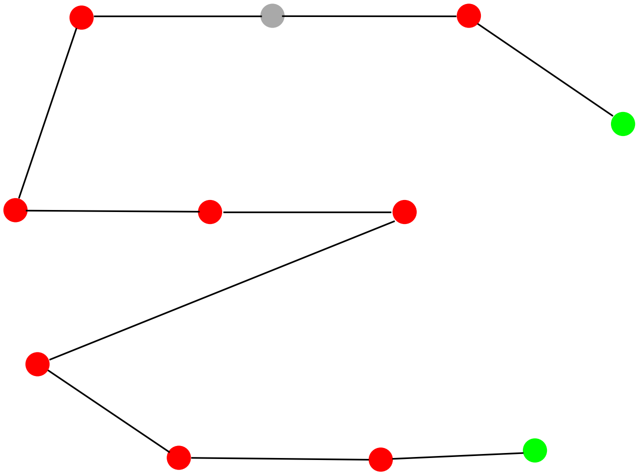
Simple-Global-Line Protocol



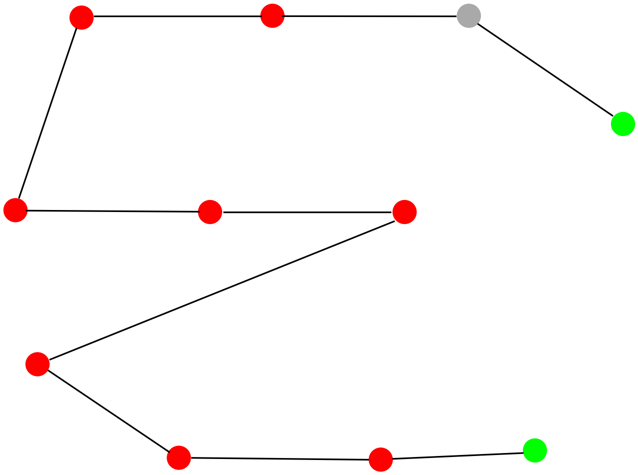
Simple-Global-Line Protocol



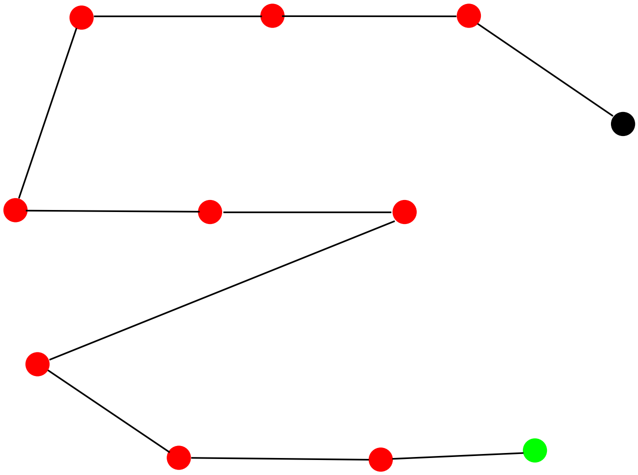
Simple-Global-Line Protocol



Simple-Global-Line Protocol



Simple-Global-Line Protocol



Simple-Global-Line Protocol

Theorem

Protocol Simple-Global-Line constructs a spanning line. It uses 5 states and its expected running time is $\Omega(n^4)$ *and* $O(n^5)$.

Proof

We have to prove two things:

1. *There is a set S of output-stable configurations whose active network is a spanning line*
2. *For every reachable configuration C it holds that $C \rightsquigarrow C_s$ for some $C_s \in S$*

1. *Spanning line, non-leader endpoints in state q_1, non-leader internal nodes in q_2, and unique leader either in l (endpoint) or in w (internal)*
2. *Any reachable C is a collection of active lines with unique leaders and isolated nodes. It is not hard to present a finite sequence of transitions that converts C to a $C_s \in S$.*

Simple-Global-Line Protocol

Theorem

Protocol Simple-Global-Line constructs a spanning line. It uses 5 states and its expected running time is $\Omega(n^4)$ *and* $O(n^5)$.

Proof

We have to prove two things:

1. *There is a set $\mathcal{S}$ of output-stable configurations whose active network is a spanning line*
2. *For every reachable configuration C it holds that $C \rightsquigarrow C_s$ for some $C_s \in \mathcal{S}$*

1. *Spanning line, non-leader endpoints in state q_1, non-leader internal nodes in q_2, and unique leader either in l (endpoint) or in w (internal)*
2. *Any reachable C is a collection of active lines with unique leaders and isolated nodes. It is not hard to present a finite sequence of transitions that converts C to a $C_s \in \mathcal{S}$.*

Simple-Global-Line Protocol

Theorem

Protocol Simple-Global-Line constructs a spanning line. It uses 5 states and its expected running time is $\Omega(n^4)$ *and* $O(n^5)$.

Proof

We have to prove two things:

1. *There is a set* $\mathcal{S}$ *of output-stable configurations whose active network is a spanning line*
2. *For every reachable configuration* C *it holds that* $C \rightsquigarrow C_s$ *for some* $C_s \in \mathcal{S}$

1. *Spanning line, non-leader endpoints in state* q_1*, non-leader internal nodes in* q_2*, and unique leader either in* l *(endpoint) or in* w *(internal)*
2. *Any reachable* C *is a collection of active lines with unique leaders and isolated nodes. It is not hard to present a finite sequence of transitions that converts* C *to a* $C_s \in \mathcal{S}$.

Simple-Global-Line Protocol

Theorem

Protocol Simple-Global-Line constructs a spanning line. It uses 5 states and its expected running time is $\Omega(n^4)$ and $O(n^5)$.

Proof

We have to prove two things:

1. *There is a set $\mathcal{S}$ of output-stable configurations whose active network is a spanning line*
2. *For every reachable configuration C it holds that $C \rightsquigarrow C_s$ for some $C_s \in \mathcal{S}$*

1. *Spanning line, non-leader endpoints in state q_1, non-leader internal nodes in q_2, and unique leader either in l (endpoint) or in w (internal)*
2. *Any reachable C is a collection of active lines with unique leaders and isolated nodes. It is not hard to present a finite sequence of transitions that converts C to a $C_s \in \mathcal{S}$.*

Simple-Global-Line Protocol

Theorem

Protocol Simple-Global-Line constructs a spanning line. It uses 5 states and its expected running time is $\Omega(n^4)$ and $O(n^5)$.

Proof

We have to prove two things:

1. *There is a set $\mathcal{S}$ of output-stable configurations whose active network is a spanning line*
2. *For every reachable configuration C it holds that $C \rightsquigarrow C_s$ for some $C_s \in \mathcal{S}$*

1. *Spanning line, non-leader endpoints in state q_1, non-leader internal nodes in q_2, and unique leader either in l (endpoint) or in w (internal)*
2. *Any reachable C is a collection of active lines with unique leaders and isolated nodes. It is not hard to present a finite sequence of transitions that converts C to a $C_s \in \mathcal{S}$.*

- *Upper bound:* $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- *Lower Bound:*
    - *w.h.p. constructs $\Theta(n)$ different lines of length 1 during its course*
    - *$k = \Theta(n)$ disjoint lines $\Rightarrow$ $k-1 = \Theta(n)$ distinct merging processes*

Simple-Global-Line Protocol

- *Upper bound:* $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- *Lower Bound:*
  - *w.h.p. constructs* $\Theta(n)$ *different lines of length 1 during its course*
  - $k = \Theta(n)$ *disjoint lines* $\Rightarrow$ $k-1 = \Theta(n)$ *distinct merging processes*
    - t_{min}: *the first time at which there is a line L of length $h \geq k/4$*
    - *It holds that* $k/4 \leq h \leq k/2 - 1$
    - *remaining length at least* $k - h \geq k - (k/2 - 1) = k/2 + 1$ *to get merged to L via distinct sequential mergings*
    - d_i: *length of the ith line merged to L*
    - $$\begin{aligned} \mathrm{E}[Y] &= \mathrm{E}[\textstyle\sum_{i=1}^{j} Y_i] = \sum_{i=1}^{j} \mathrm{E}[Y_i] = \sum_{i=1}^{j} n^2(h + d_1 + \ldots + d_{i-1})d_i \\ &\geq n^2 \textstyle\sum_{i=1}^{j} hd_i = n^2 h \sum_{i=1}^{j} d_i \geq n^2 \cdot \frac{k}{4} \cdot (\frac{k}{2} + 1) \\ &= n^2 \cdot \Theta(n) \cdot \Theta(n) \\ &= \Theta(n^4) \end{aligned}$$

$\square$

Simple-Global-Line Protocol

- *Upper bound:* $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- *Lower Bound:*
  - *w.h.p. constructs* $\Theta(n)$ *different lines of length 1 during its course*
  - $k = \Theta(n)$ *disjoint lines* $\Rightarrow$ $k-1 = \Theta(n)$ *distinct merging processes*
    - t_{min}*: the first time at which there is a line L of length* $h \geq k/4$
    - *It holds that* $k/4 \leq h \leq k/2-1$
    - *remaining length at least* $k-h \geq k-(k/2-1) = k/2+1$ *to get merged to L via distinct sequential mergings*
    - d_i*: length of the ith line merged to L*
    - $$\begin{aligned} E[Y] &= E[\textstyle\sum_{i=1}^{j} Y_i] = \sum_{i=1}^{j} E[Y_i] = \sum_{i=1}^{j} n^2(h+d_1+\ldots+d_{i-1})d_i \\ &\geq n^2 \textstyle\sum_{i=1}^{j} hd_i = n^2 h \sum_{i=1}^{j} d_i \geq n^2 \cdot \frac{k}{4} \cdot (\frac{k}{2}+1) \\ &= n^2 \cdot \Theta(n) \cdot \Theta(n) \\ &= \Theta(n^4) \end{aligned}$$

$\square$

Simple-Global-Line Protocol

- *Upper bound:* $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- *Lower Bound:*
  - *w.h.p. constructs* $\Theta(n)$ *different lines of length 1 during its course*
  - $k = \Theta(n)$ *disjoint lines* $\Rightarrow$ $k-1 = \Theta(n)$ *distinct merging processes*
    - t_{min}: *the first time at which there is a line L of length $h \geq k/4$*
    - *It holds that* $k/4 \leq h \leq k/2-1$
    - *remaining length at least* $k-h \geq k-(k/2-1) = k/2+1$ *to get merged to L via distinct sequential mergings*
    - d_i: *length of the ith line merged to L*
    - $E[Y] = E[\sum_{i=1}^{j} Y_i] = \sum_{i=1}^{j} E[Y_i] = \sum_{i=1}^{j} n^2(h+d_1+\ldots+d_{i-1})d_i$

$$\geq n^2 \sum_{i=1}^{j} hd_i = n^2 h \sum_{i=1}^{j} d_i \geq n^2 \cdot \frac{k}{4} \cdot (\frac{k}{2}+1)$$
$$= n^2 \cdot \Theta(n) \cdot \Theta(n)$$
$$= \Theta(n^4)$$

□

Simple-Global-Line Protocol

- *Upper bound:* $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- *Lower Bound:*
  - *w.h.p. constructs* $\Theta(n)$ *different lines of length 1 during its course*
  - $k = \Theta(n)$ *disjoint lines* $\Rightarrow$ $k-1 = \Theta(n)$ *distinct merging processes*
    - t_{min}: *the first time at which there is a line* L *of length* $h \geq k/4$
    - *It holds that* $k/4 \leq h \leq k/2 - 1$
    - *remaining length at least* $k - h \geq k - (k/2-1) = k/2+1$ *to get merged to* L *via distinct sequential mergings*
    - d_i: *length of the* i*th line merged to* L
    - $$\begin{aligned} \mathrm{E}[Y] &= \mathrm{E}[\textstyle\sum_{i=1}^{j} Y_i] = \sum_{i=1}^{j} \mathrm{E}[Y_i] = \sum_{i=1}^{j} n^2(h + d_1 + \ldots + d_{i-1})d_i \\ &\geq n^2 \textstyle\sum_{i=1}^{j} hd_i = n^2 h \sum_{i=1}^{j} d_i \geq n^2 \cdot \frac{k}{4} \cdot (\frac{k}{2}+1) \\ &= n^2 \cdot \Theta(n) \cdot \Theta(n) \\ &= \Theta(n^4) \end{aligned}$$

$\square$

Simple-Global-Line Protocol

- *Upper bound:* $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- *Lower Bound:*
  - *w.h.p. constructs* $\Theta(n)$ *different lines of length 1 during its course*
  - $k = \Theta(n)$ *disjoint lines* $\Rightarrow$ $k-1 = \Theta(n)$ *distinct merging processes*
    - t_{min}: *the first time at which there is a line L of length* $h \geq k/4$
    - *It holds that* $k/4 \leq h \leq k/2 - 1$
    - *remaining length at least* $k - h \geq k - (k/2 - 1) = k/2 + 1$ *to get merged to L via distinct sequential mergings*
    - d_i: *length of the ith line merged to L*
    - $E[Y] = E[\sum_{i=1}^{j} Y_i] = \sum_{i=1}^{j} E[Y_i] = \sum_{i=1}^{j} n^2(h + d_1 + \ldots + d_{i-1})d_i$

$$\geq n^2 \sum_{i=1}^{j} hd_i = n^2 h \sum_{i=1}^{j} d_i \geq n^2 \cdot \frac{k}{4} \cdot (\frac{k}{2} + 1)$$

$$= n^2 \cdot \Theta(n) \cdot \Theta(n)$$

$$= \Theta(n^4)$$

$\square$

- *Upper bound:* $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- *Lower Bound:*
  - *w.h.p. constructs* $\Theta(n)$ *different lines of length 1 during its course*
  - $k = \Theta(n)$ *disjoint lines* $\Rightarrow$ $k-1 = \Theta(n)$ *distinct merging processes*
    - t_{min}*: the first time at which there is a line L of length* $h \geq k/4$
    - *It holds that* $k/4 \leq h \leq k/2 - 1$
    - *remaining length at least* $k - h \geq k - (k/2 - 1) = k/2 + 1$ *to get merged to L via distinct sequential mergings*
    - d_i*: length of the ith line merged to L*
    - $\mathrm{E}[Y] = \mathrm{E}[\sum_{i=1}^{j} Y_i] = \sum_{i=1}^{j} \mathrm{E}[Y_i] = \sum_{i=1}^{j} n^2(h + d_1 + \ldots + d_{i-1})d_i$
      $\geq n^2 \sum_{i=1}^{j} hd_i = n^2 h \sum_{i=1}^{j} d_i \geq n^2 \cdot \frac{k}{4} \cdot (\frac{k}{2}+1)$
      $= n^2 \cdot \Theta(n) \cdot \Theta(n)$
      $= \Theta(n^4)$

$\square$

Simple-Global-Line Protocol

- *Upper bound:* $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- *Lower Bound:*
  - *w.h.p. constructs* $\Theta(n)$ *different lines of length 1 during its course*
  - $k = \Theta(n)$ *disjoint lines* $\Rightarrow$ $k-1=\Theta(n)$ *distinct merging processes*
    - t_{min}*: the first time at which there is a line* L *of length* $h \geq k/4$
    - *It holds that* $k/4 \leq h \leq k/2-1$
    - *remaining length at least* $k-h \geq k-(k/2-1) = k/2+1$ *to get merged to* L *via distinct sequential mergings*
    - d_i*: length of the* i*th line merged to* L
    - $$\begin{aligned} \mathrm{E}[Y] &= \mathrm{E}[\textstyle\sum_{i=1}^{j} Y_i] = \sum_{i=1}^{j} \mathrm{E}[Y_i] = \sum_{i=1}^{j} n^2(h+d_1+\ldots+d_{i-1})d_i \\ &\geq n^2 \textstyle\sum_{i=1}^{j} hd_i = n^2 h \sum_{i=1}^{j} d_i \geq n^2 \cdot \frac{k}{4} \cdot (\frac{k}{2}+1) \\ &= n^2 \cdot \Theta(n) \cdot \Theta(n) \\ &= \Theta(n^4) \end{aligned}$$

□

Fast-Global-Line Protocol

$$
\begin{aligned}
(q_0, q_0, 0) &\to (q_1, l, 1) \\
(l, q_0, 0) &\to (q_2, l, 1) \\
(l, l, 0) &\to (q_2', l', 1) \\
(l', q_2, 1) &\to (l'', f_1, 0) \\
(l', q_1, 1) &\to (l'', f_0, 0) \\
(l'', q_2', 1) &\to (l, q_2, 1) \\
(l, f_0, 0) &\to (q_2, l, 1) \\
(l, f_1, 0) &\to (q_2', l', 1)
\end{aligned}
$$

Fast-Global-Line Protocol

- Avoid mergings as they seem to consume much time
  - even deterministic merging takes time $\Omega(n^3)$ and $O(n^4)$
- When leaders of lines interact, they play a pairwise game
- The winner grows by one towards the other line and the loser sleeps
- A sleeping line cannot increase any more and only loses nodes by lines that are still awake
- A single leader is guaranteed to always win and eventually remain unique and this occurs quite fast
- The leader makes progress (by one) in most interactions and every such progress is in turn quite fast
- Running-time: $O(n^3)$

Fast-Global-Line Protocol

- Avoid mergings as they seem to consume much time
  - even deterministic merging takes time $\Omega(n^3)$ and $O(n^4)$
- When leaders of lines interact, they play a pairwise game
- The winner grows by one towards the other line and the loser sleeps
- A sleeping line cannot increase any more and only loses nodes by lines that are still awake
- A single leader is guaranteed to always win and eventually remain unique and this occurs quite fast
- The leader makes progress (by one) in most interactions and every such progress is in turn quite fast
- Running-time: $O(n^3)$

Fast-Global-Line Protocol

- Avoid mergings as they seem to consume much time
  - even deterministic merging takes time $\Omega(n^3)$ and $O(n^4)$
- When leaders of lines interact, they play a pairwise game
- The winner grows by one towards the other line and the loser sleeps
- A sleeping line cannot increase any more and only loses nodes by lines that are still awake
- A single leader is guaranteed to always win and eventually remain unique and this occurs quite fast
- The leader makes progress (by one) in most interactions and every such progress is in turn quite fast
- Running-time: $O(n^3)$

Fast-Global-Line Protocol

- Avoid mergings as they seem to consume much time
  - even deterministic merging takes time $\Omega(n^3)$ and $O(n^4)$
- When leaders of lines interact, they play a pairwise game
- The winner grows by one towards the other line and the loser sleeps
- A sleeping line cannot increase any more and only loses nodes by lines that are still awake
- A single leader is guaranteed to always win and eventually remain unique and this occurs quite fast
- The leader makes progress (by one) in most interactions and every such progress is in turn quite fast
- Running-time: $O(n^3)$

Fast-Global-Line Protocol

- Avoid mergings as they seem to consume much time
  - even deterministic merging takes time $\Omega(n^3)$ and $O(n^4)$
- When leaders of lines interact, they play a pairwise game
- The winner grows by one towards the other line and the loser sleeps
- A sleeping line cannot increase any more and only loses nodes by lines that are still awake
- A single leader is guaranteed to always win and eventually remain unique and this occurs quite fast
- The leader makes progress (by one) in most interactions and every such progress is in turn quite fast
- Running-time: $O(n^3)$

Fast-Global-Line Protocol

- Avoid mergings as they seem to consume much time
  - even deterministic merging takes time $\Omega(n^3)$ and $O(n^4)$
- When leaders of lines interact, they play a pairwise game
- The winner grows by one towards the other line and the loser sleeps
- A sleeping line cannot increase any more and only loses nodes by lines that are still awake
- A single leader is guaranteed to always win and eventually remain unique and this occurs quite fast
- The leader makes progress (by one) in most interactions and every such progress is in turn quite fast
- Running-time: $O(n^3)$

Fast-Global-Line Protocol

- Avoid mergings as they seem to consume much time
  - even deterministic merging takes time $\Omega(n^3)$ and $O(n^4)$
- When leaders of lines interact, they play a pairwise game
- The winner grows by one towards the other line and the loser sleeps
- A sleeping line cannot increase any more and only loses nodes by lines that are still awake
- A single leader is guaranteed to always win and eventually remain unique and this occurs quite fast
- The leader makes progress (by one) in most interactions and every such progress is in turn quite fast
- Running-time: $O(n^3)$

All Established Bounds

Protocol	# states	Expected Time	Lower Bound
Simple-Global-Line	5	$\Omega(n^4)$ and $O(n^5)$	$\Omega(n^2)$
Intermediate-Global-Line	8	$\Omega(n^3)$ and $O(n^4)$	$\Omega(n^2)$
Fast-Global-Line	9	$O(n^3)$	$\Omega(n^2)$
Cycle-Cover	3	$\Theta(n^2)$ (optimal)	$\Omega(n^2)$
Global-Star	2 (optimal)	$\Theta(n^2 \log n)$ (optimal)	$\Omega(n^2 \log n)$
Global-Ring	9		$\Omega(n^2)$
2RC	6		$\Omega(n \log n)$
kRC	$2(k+1)$		$\Omega(n \log n)$
c-Cliques	$5c-3$		$\Omega(n \log n)$
Graph-Replication	12	$\Theta(n^4 \log n)$	

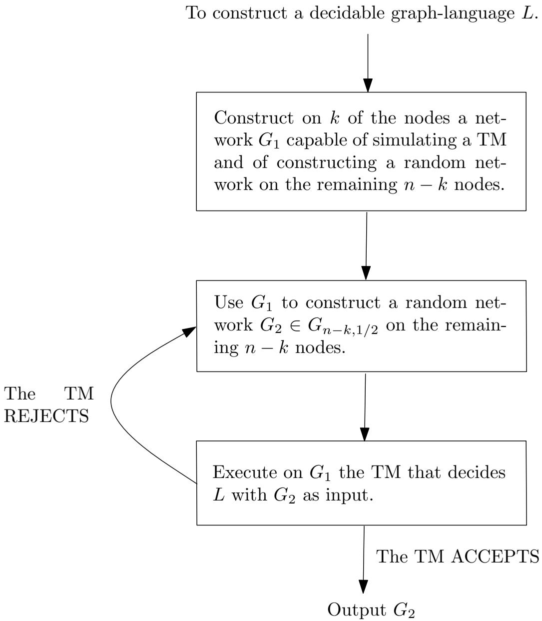
Generic Constructors

Question: Is there a generic constructor capable of constructing a large class of networks?

1. constructors that simulate a Turing Machine
2. a constructor that simulates a distributed system with names and logarithmic local memories

- l: the binary length of the input of a TM
  - In what follows, $l = \Theta(n^2)$
- **DGS**$(f(l))$: the class of graph languages decidable by a TM of (binary) space $f(l)$ (input graph in adjacency matrix encoding)
- **REL**$(g(n))$: the class of graph languages constructible with useful space $g(n)$ (relation or on/off class)
- **PREL**$(g(n))$: (i) allow transitions that with probability $1/2$ give one outcome and with probability $1/2$ another (ii) all graphs must be constructed equiprobably

Generic Constructors: Main Mechanism



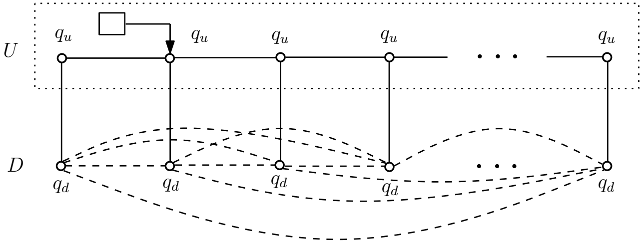
Theorem (Linear Waste-Half)

DGS$(O(n)) \subseteq$ **PREL**$(\lfloor n/2 \rfloor)$. *In words, for every graph language L that is decidable by a $O(n)$-space TM, there is a protocol that constructs L equiprobably with useful space $\lfloor n/2 \rfloor$.*

Proof

- *Partition the population into equal sets U and D and construct an active perfect matching between U and D*
- *Construct a spanning line in U (e.g. Fast-Global-Line)*
- *Organize the line into a TM M*
- *M must compute a graph from L* and *construct it on D*
  - *uniquely identify the nodes of D by their distance from one endpoint*
  - *to modify edge (i, j) mark appropriately the D-nodes at distances i and j from one endpoint*

Linear Waste-Half



- *M computes a graph G from L equiprobably: activate/deactivate each edge of D equiprobably and independently of other edges*
- *Then it simulates on input G the TM that decides L in $\sqrt{l}$ space to determine whether $G \in L$*
- *The TM rejects: M repeats the random experiment to produce a new random graph G'*
- *The TM accepts: release the network, set D-nodes to q_{out}*
- *Reinitialize whenever the global line protocol makes progress* □

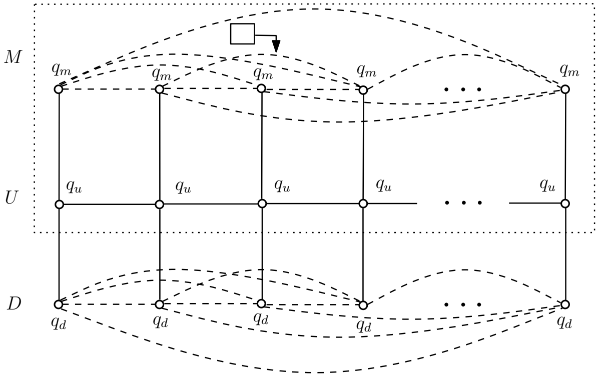
Reinitialization

- Nodes cannot detect termination and cannot sequentially compose subroutines
  - Instead, all subroutines must be executed in parallel and become "sequentialized" by perpetual reinitializations
- Executed whenever a line on U-nodes expands
- The protocol "assumes" that no further expansions will occur, restores the components of the simulation to their original values, ensures that each U-node has a D-neighbor, and starts drawing a new random graph
- The assumption may be wrong several times
- However, eventually the assumption of the protocol will be correct
  - The simulation will be reinitialized and executed for the last time on the correct sets U and D

Later we shall discuss terminating protocols that are correct w.h.p.

Theorem (Linear Waste-Two Thirds)

DGS$(O(n^2) + O(n)) \subseteq$ **PREL**$(\lfloor n/3 \rfloor)$. *That is, for every graph language L that is decidable by a $O(n^2)$-space TM, there is a protocol that constructs L equiprobably with useful space $\lfloor n/3 \rfloor$.*



Logarithmic Waste

Theorem (Logarithmic Waste)

$\mathbf{DGS}(O(\log n)) \subseteq \mathbf{PREL}(n - \log n)$.

Proof.

- Construct a line to count n in binary, i.e. to occupy $\log n$ cells
- Release the constructed counter and reinitialize the other (free) nodes
- So, there is a logarithmic memory with a leader, knowing a good estimate of $n - \log n$
- Construct a random graph on the free nodes:
  - For every free node, let it toss a coin on one after the other its edges to other free nodes
  - To know when to stop, count on the line up to $n - \log n$ (known) □

Supernodes

- A population consisting of n nodes can be partitioned into k *supernodes* each consisting of $\log k$ nodes, for the largest such k
- The internal structure of each supernode is a line, thus it can be operated as a TM of memory logarithmic in the total number of supernodes
- This amount of storage is sufficient for the supernodes to obtain unique names and exploit their names and their internal storage to realize nontrivial constructions
- We are interested in the networks that can be constructed at the supernode abstraction layer

Supernodes

Theorem (Partitioning into Supernodes)

For every network G that can be constructed by k nodes having local memories $\lceil \log k \rceil$ and unique names there is a NET that constructs G on $n = k\lceil \log k \rceil$ nodes.

Network Transformations

[Michail and Spirakis, TCS '16]

- Minimal strengthenings of NETs that can maximize computational power
  - Also gain termination
- Initial configuration: any connected graph spanning V
- Ability to detect small local degrees
  - e.g., a node can detect if its active degree is 0
  - we can now simulate any constructor that assumes an empty initial network
- Pre-elected leader or pairs of nodes able to tell whether they have a neighbor in common

[Michail and Spirakis, TCS '16]

- Minimal strengthenings of NETs that can maximize computational power
  - Also gain termination
- Initial configuration: any connected graph spanning V
- Ability to detect small local degrees
  - e.g., a node can detect if its active degree is 0
  - we can now simulate any constructor that assumes an empty initial network
- Pre-elected leader or pairs of nodes able to tell whether they have a neighbor in common

[Michail and Spirakis, TCS '16]

- Minimal strengthenings of NETs that can maximize computational power
  - Also gain termination
- Initial configuration: any connected graph spanning V
- Ability to detect small local degrees
  - e.g., a node can detect if its active degree is 0
  - we can now simulate any constructor that assumes an empty initial network
- Pre-elected leader or pairs of nodes able to tell whether they have a neighbor in common

[Michail and Spirakis, TCS '16]

- Minimal strengthenings of NETs that can maximize computational power
  - Also gain termination
- Initial configuration: any connected graph spanning V
- Ability to detect small local degrees
  - e.g., a node can detect if its active degree is 0
  - we can now simulate any constructor that assumes an empty initial network
- Pre-elected leader or pairs of nodes able to tell whether they have a neighbor in common

Characterization

- Gives the maximum computational power that one can hope for in this family of models
- Can compute with termination any symmetric predicate computable by a TM of space $\Theta(n^2)$, and no more than this, i.e., it is an exact characterization
- Symmetricity can only be dropped by UIDs or other means of maintaining an ordering of the nodes' inputs
- This power is maximal because the distributed space of the system is $\Theta(n^2)$
- Universal computations are now terminating and not just eventually stabilizing
- The additional mechanisms are minimal. Removing each one leads to
  - impossibility of termination or
  - substantial decrease in the computational power

Characterization

- Gives the maximum computational power that one can hope for in this family of models
- Can compute with termination any symmetric predicate computable by a TM of space $\Theta(n^2)$, and no more than this, i.e., it is an exact characterization
- Symmetricity can only be dropped by UIDs or other means of maintaining an ordering of the nodes' inputs
- This power is maximal because the distributed space of the system is $\Theta(n^2)$
- Universal computations are now terminating and not just eventually stabilizing
- The additional mechanisms are minimal. Removing each one leads to
  - impossibility of termination or
  - substantial decrease in the computational power

Characterization

- Gives the maximum computational power that one can hope for in this family of models
- Can compute with termination any symmetric predicate computable by a TM of space $\Theta(n^2)$, and no more than this, i.e., it is an exact characterization
- Symmetricity can only be dropped by UIDs or other means of maintaining an ordering of the nodes' inputs
- This power is maximal because the distributed space of the system is $\Theta(n^2)$
- Universal computations are now terminating and not just eventually stabilizing
- The additional mechanisms are minimal. Removing each one leads to
  - impossibility of termination or
  - substantial decrease in the computational power

Characterization

- Gives the maximum computational power that one can hope for in this family of models
- Can compute with termination any symmetric predicate computable by a TM of space $\Theta(n^2)$, and no more than this, i.e., it is an exact characterization
- Symmetricity can only be dropped by UIDs or other means of maintaining an ordering of the nodes' inputs
- This power is maximal because the distributed space of the system is $\Theta(n^2)$
- Universal computations are now terminating and not just eventually stabilizing
- The additional mechanisms are minimal. Removing each one leads to
  - impossibility of termination or
  - substantial decrease in the computational power

Characterization

- Gives the maximum computational power that one can hope for in this family of models
- Can compute with termination any symmetric predicate computable by a TM of space $\Theta(n^2)$, and no more than this, i.e., it is an exact characterization
- Symmetricity can only be dropped by UIDs or other means of maintaining an ordering of the nodes' inputs
- This power is maximal because the distributed space of the system is $\Theta(n^2)$
- Universal computations are now terminating and not just eventually stabilizing
- The additional mechanisms are minimal. Removing each one leads to
  - impossibility of termination or
  - substantial decrease in the computational power

Characterization

- Gives the maximum computational power that one can hope for in this family of models
- Can compute with termination any symmetric predicate computable by a TM of space $\Theta(n^2)$, and no more than this, i.e., it is an exact characterization
- Symmetricity can only be dropped by UIDs or other means of maintaining an ordering of the nodes' inputs
- This power is maximal because the distributed space of the system is $\Theta(n^2)$
- Universal computations are now terminating and not just eventually stabilizing
- The additional mechanisms are minimal. Removing each one leads to
  - impossibility of termination or
  - substantial decrease in the computational power

Approach

- Develop protocols exploiting initial connectivity to
  - transform the network to a less symmetric and detectable
  - without ever breaking its connectivity
- Still, very symmetric components cannot be exploited for powerful computations
- Symmetric target-networks may not be detectable for termination
- Our protocols always transform to a spanning line
  - Detectable
  - Can serve as a linear memory
  - Exploit it to count n and to simulate TMs of space $\Theta(n)$ or even $\Theta(n^2)$ (with an additional transformation)

Approach

- Develop protocols exploiting initial connectivity to
  - transform the network to a less symmetric and detectable
  - without ever breaking its connectivity
- Still, very symmetric components cannot be exploited for powerful computations
- Symmetric target-networks may not be detectable for termination
- Our protocols always transform to a spanning line
  - Detectable
  - Can serve as a linear memory
  - Exploit it to count n and to simulate TMs of space $\Theta(n)$ or even $\Theta(n^2)$ (with an additional transformation)

Approach

- Develop protocols exploiting initial connectivity to
  - transform the network to a less symmetric and detectable
  - without ever breaking its connectivity
- Still, very symmetric components cannot be exploited for powerful computations
- Symmetric target-networks may not be detectable for termination
- Our protocols always transform to a spanning line
  - Detectable
  - Can serve as a linear memory
  - Exploit it to count n and to simulate TMs of space $\Theta(n)$ or even $\Theta(n^2)$ (with an additional transformation)

Approach

- Develop protocols exploiting initial connectivity to
  - transform the network to a less symmetric and detectable
  - without ever breaking its connectivity
- Still, very symmetric components cannot be exploited for powerful computations
- Symmetric target-networks may not be detectable for termination
- Our protocols always transform to a spanning line
  - Detectable
  - Can serve as a linear memory
  - Exploit it to count n and to simulate TMs of space $\Theta(n)$ or even $\Theta(n^2)$ (with an additional transformation)

Approach

- Develop protocols exploiting initial connectivity to
  - transform the network to a less symmetric and detectable
  - without ever breaking its connectivity
- Still, very symmetric components cannot be exploited for powerful computations
- Symmetric target-networks may not be detectable for termination
- Our protocols always transform to a spanning line
  - Detectable
  - Can serve as a linear memory
  - Exploit it to count n and to simulate TMs of space $\Theta(n)$ or even $\Theta(n^2)$ (with an additional transformation)

Approach

- Develop protocols exploiting initial connectivity to
  - transform the network to a less symmetric and detectable
  - without ever breaking its connectivity
- Still, very symmetric components cannot be exploited for powerful computations
- Symmetric target-networks may not be detectable for termination
- Our protocols always transform to a spanning line
  - Detectable
  - Can serve as a linear memory
  - Exploit it to count n and to simulate TMs of space $\Theta(n)$ or even $\Theta(n^2)$ (with an additional transformation)

Our Results

- Unique leader + detect local degree 1
  - Time-optimal transformation to spanning line, with running time $\Theta(n^2 \log n)$
  - Implies a full-power TM simulation
- Identical nodes
  - Impossibility: no protocol to transform to an acyclic topology without breaking connectivity
- Common neighbor detection
  - transformation to spanning line, with running time $O(n^3)$
  - Implies a full-power TM simulation

Our Results

- Unique leader + detect local degree 1
  - Time-optimal transformation to spanning line, with running time $\Theta(n^2 \log n)$
  - Implies a full-power TM simulation
- Identical nodes
  - Impossibility: no protocol to transform to an acyclic topology without breaking connectivity
- Common neighbor detection
  - transformation to spanning line, with running time $O(n^3)$
  - Implies a full-power TM simulation

Our Results

- Unique leader + detect local degree 1
  - Time-optimal transformation to spanning line, with running time $\Theta(n^2 \log n)$
  - Implies a full-power TM simulation
- Identical nodes
  - Impossibility: no protocol to transform to an acyclic topology without breaking connectivity
- Common neighbor detection
  - transformation to spanning line, with running time $O(n^3)$
  - Implies a full-power TM simulation

Line-Around-a-Star

- Unique leader in state l and all other nodes in state q_0
- The leader (center) starts connecting with the q_0s converting them to p_0s (peripherals)
- p_0s repel by disconnecting edges between them
- A p_0 with degree 1 becomes p (normal peripheral)
- The leader constructs a line with its left endpoint on the center of the star and expanding over the peripherals until it covers them all
- Terminates when the degree of the center becomes 1 for the first time

Line-Around-a-Star

- Unique leader in state l and all other nodes in state q_0
- The leader (center) starts connecting with the q_0s converting them to p_0s (peripherals)
- p_0s repel by disconnecting edges between them
- A p_0 with degree 1 becomes p (normal peripheral)
- The leader constructs a line with its left endpoint on the center of the star and expanding over the peripherals until it covers them all
- Terminates when the degree of the center becomes 1 for the first time

Line-Around-a-Star

- Unique leader in state l and all other nodes in state q_0
- The leader (center) starts connecting with the q_0s converting them to p_0s (peripherals)
- p_0s repel by disconnecting edges between them
- A p_0 with degree 1 becomes p (normal peripheral)
- The leader constructs a line with its left endpoint on the center of the star and expanding over the peripherals until it covers them all
- Terminates when the degree of the center becomes 1 for the first time

Line-Around-a-Star

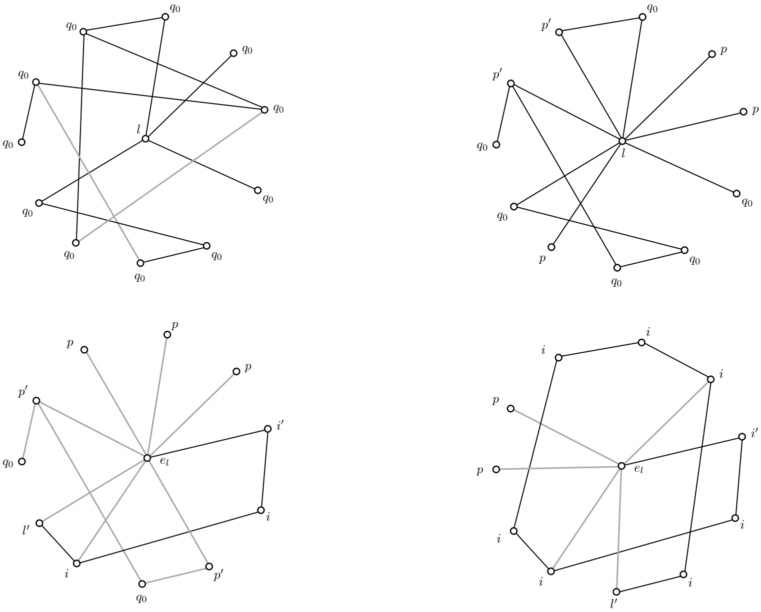
- Unique leader in state l and all other nodes in state q_0
- The leader (center) starts connecting with the q_0s converting them to p_0s (peripherals)
- p_0s repel by disconnecting edges between them
- A p_0 with degree 1 becomes p (normal peripheral)
- The leader constructs a line with its left endpoint on the center of the star and expanding over the peripherals until it covers them all
- Terminates when the degree of the center becomes 1 for the first time

Line-Around-a-Star

- Unique leader in state l and all other nodes in state q_0
- The leader (center) starts connecting with the q_0s converting them to p_0s (peripherals)
- p_0s repel by disconnecting edges between them
- A p_0 with degree 1 becomes p (normal peripheral)
- The leader constructs a line with its left endpoint on the center of the star and expanding over the peripherals until it covers them all
- Terminates when the degree of the center becomes 1 for the first time

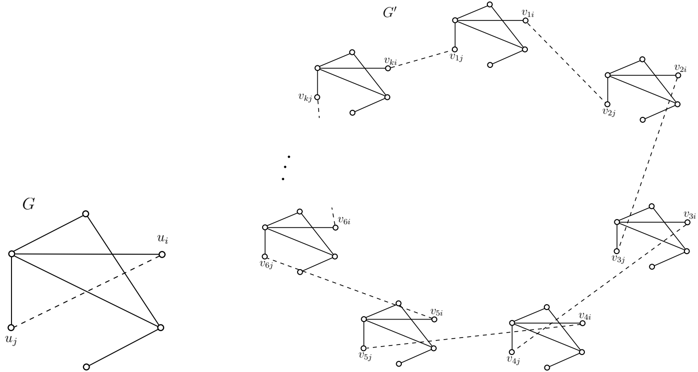
- Unique leader in state l and all other nodes in state q_0
- The leader (center) starts connecting with the q_0s converting them to p_0s (peripherals)
- p_0s repel by disconnecting edges between them
- A p_0 with degree 1 becomes p (normal peripheral)
- The leader constructs a line with its left endpoint on the center of the star and expanding over the peripherals until it covers them all
- Terminates when the degree of the center becomes 1 for the first time

An Example Execution



Theorem (Strong Impossibility)

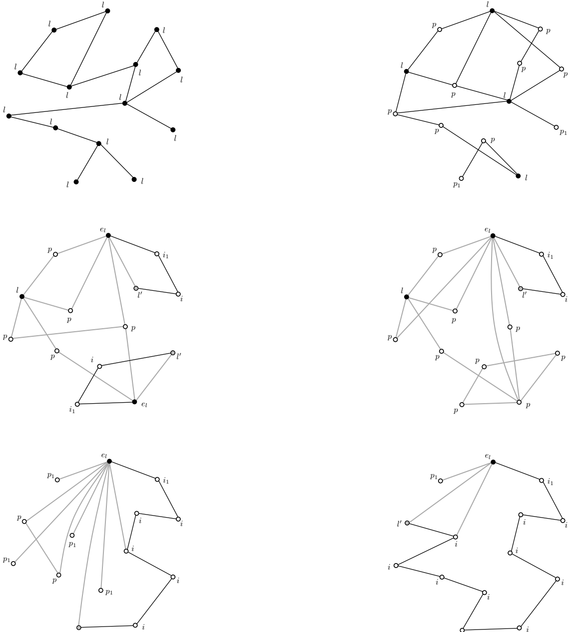
For every connected graph G with at least one cycle, there is an infinite family of graphs $\mathcal{G}$ such that for every $G' \in \mathcal{G}$ every protocol (beginning from identical states on all nodes) that makes G acyclic may disconnect G' in some executions.



Theorem

By assuming that nodes are equipped with a common neighbor detection mechanism and have the ability to detect local degrees 1 and 2, Protocol Line-Transformer solves the Terminating Line Transformation problem in the setting in which all nodes are initially identical. Its running time is $O(n^3)$.

An Example Execution



All Established Bounds

Protocol	Leader	DD	CND	Expected Time	Lower Bound
Online-Cycle-Elimination	Yes	1	No	$\Theta(n^4)$	$\Omega(n^2 \log n)$
Line-Around-a-Star	Yes	1	No	$\Theta(n^2 \log n)$ (opt)	$\Omega(n^2 \log n)$
Line-Transformer	No	1,2	Yes	$O(n^3)$	$\Omega(n^2 \log n)$

- Leader: Whether it makes use of a pre-elected unique leader
- DD: what local degree detection is used
- CND: whether it uses common neighbor detection
- Expected Time: Expected running time under the uniform random scheduler
- Lower Bound: Best known lower bound

A Geometric Version

[Michail, PODC '15]

- Adjust some of the abstract parameters of NETs
- Allowable interactions are geometrically constrained (by already formed structures)
- Each device can connect to other devices only via a limited number of ports (4 in 2D and 6 in 3D)
  - Connections are made at unit distance and are perpendicular to neighboring connections
  - Known universal constructors do not apply in this case

A Geometric Version

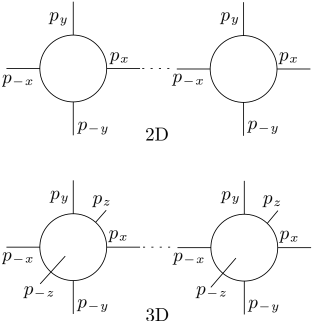
[Michail, PODC '15]

- Adjust some of the abstract parameters of NETs
- Allowable interactions are geometrically constrained (by already formed structures)
- Each device can connect to other devices only via a limited number of ports (4 in 2D and 6 in 3D)
  - Connections are made at unit distance and are perpendicular to neighboring connections
  - Known universal constructors do not apply in this case

A Geometric Version

[Michail, PODC '15]

- Adjust some of the abstract parameters of NETs
- Allowable interactions are geometrically constrained (by already formed structures)
- Each device can connect to other devices only via a limited number of ports (4 in 2D and 6 in 3D)

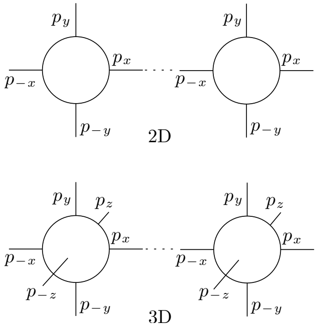


- Connections are made at unit distance and are perpendicular to neighboring connections
- Known universal constructors do not apply in this case

A Geometric Version

[Michail, PODC '15]

- Adjust some of the abstract parameters of NETs
- Allowable interactions are geometrically constrained (by already formed structures)
- Each device can connect to other devices only via a limited number of ports (4 in 2D and 6 in 3D)

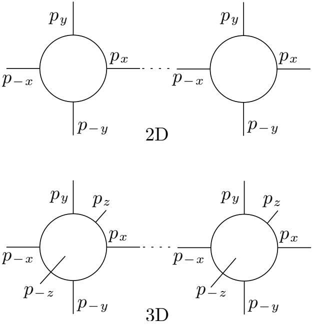


- Connections are made at unit distance and are perpendicular to neighboring connections
- Known universal constructors do not apply in this case

A Geometric Version

[Michail, PODC '15]

- Adjust some of the abstract parameters of NETs
- Allowable interactions are geometrically constrained (by already formed structures)
- Each device can connect to other devices only via a limited number of ports (4 in 2D and 6 in 3D)



- Connections are made at unit distance and are perpendicular to neighboring connections
- Known universal constructors do not apply in this case

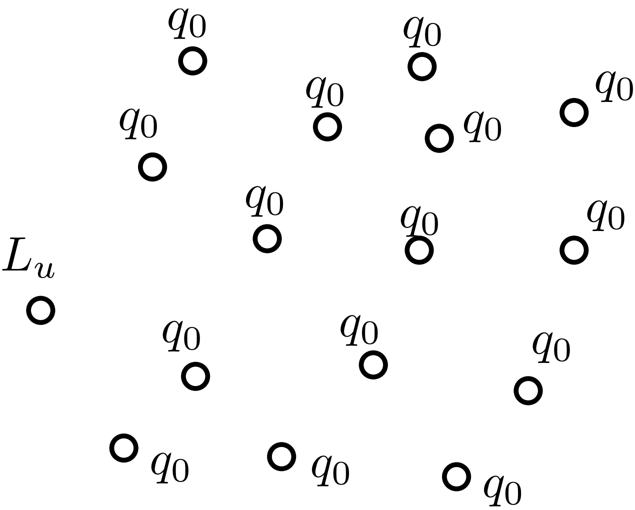
The Model

- The transition function is now:
  $\delta : (Q \times P) \times (Q \times P) \times \{0, 1\} \to Q \times Q \times \{0, 1\}$,
  $P = \{u, r, d, l\}$ is the set of ports in 2D
- In every step, a pair $(v_1, p_1)(v_2, p_2)$ is selected by the scheduler and v_1, v_2 interact via their p_1, p_2 ports according to δ
- Valid configuration: its connected components are subnetworks of the 2D grid network
- Uniform random scheduler: selects independently and uniformly at random from the permitted interactions (leading to valid config.)
- Output shape: nodes that are in output (or halting) states and edges between them that are active

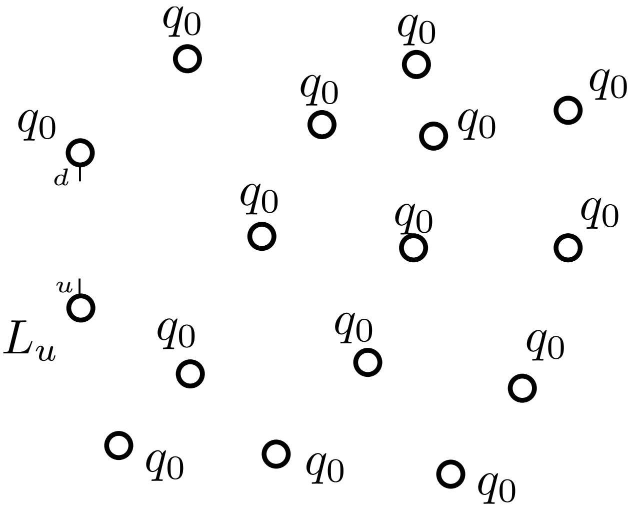
An Illustration: Spanning Square

- δ:

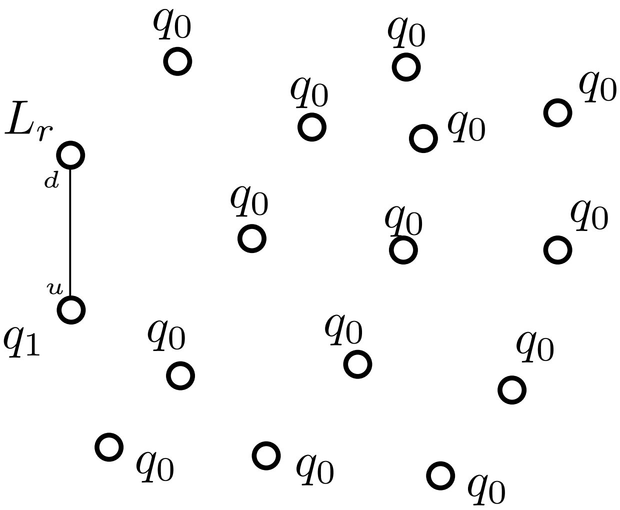
$$
\begin{aligned}
(L_u, u), (q_0, d), 0 &\rightarrow (q_1, L_r, 1) \\
(L_r, r), (q_0, l), 0 &\rightarrow (q_1, L_d, 1) \\
(L_d, d), (q_0, u), 0 &\rightarrow (q_1, L_l, 1) \\
(L_l, l), (q_0, r), 0 &\rightarrow (q_1, L_u, 1) \\
(L_u, u), (q_1, d), 0 &\rightarrow (L_l, q_1, 1) \\
(L_r, r), (q_1, l), 0 &\rightarrow (L_u, q_1, 1) \\
(L_d, d), (q_1, u), 0 &\rightarrow (L_r, q_1, 1) \\
(L_l, l), (q_1, r), 0 &\rightarrow (L_d, q_1, 1)
\end{aligned}
$$

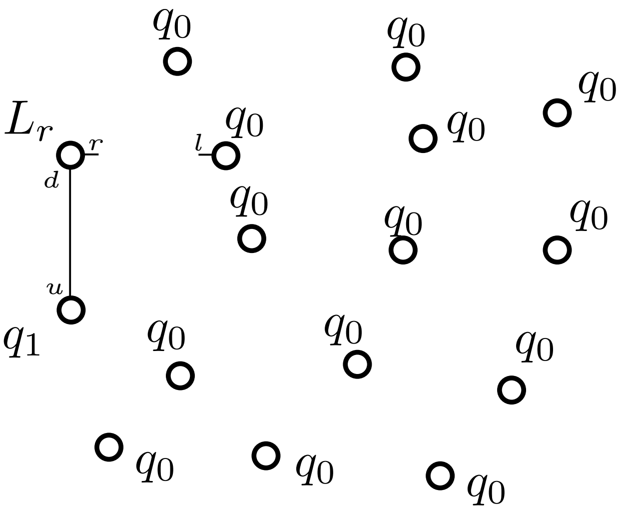

q_0
q_0
q_0
q_0
q_0
q_0
q_0
q_0
q_0
L_u
q_0
q_0
q_0
q_0
q_0
q_0

Spanning-Square Protocol

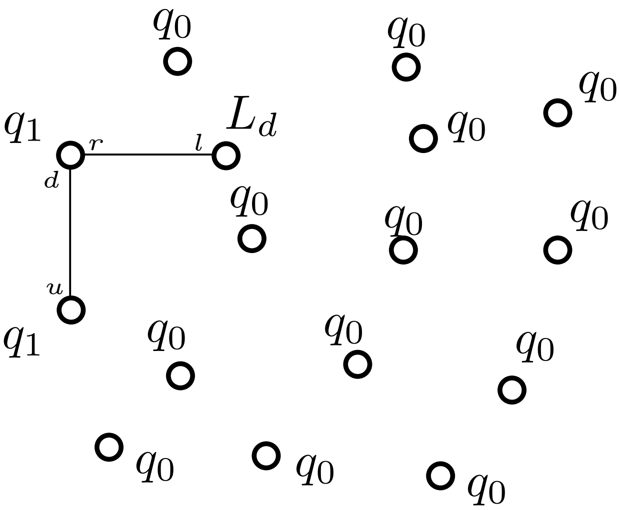


Spanning-Square Protocol



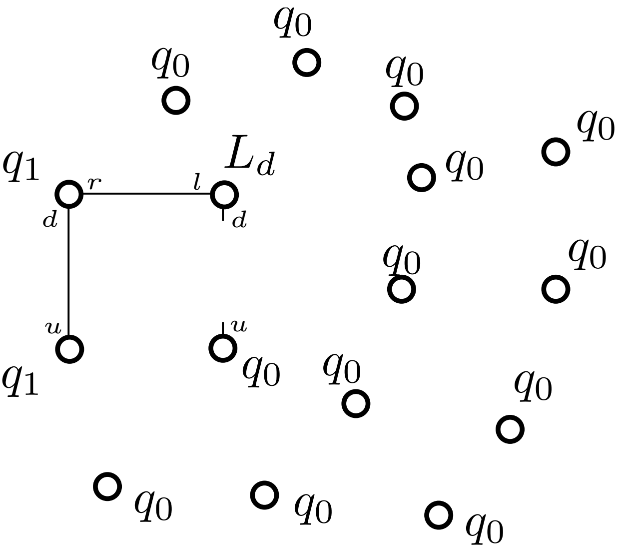


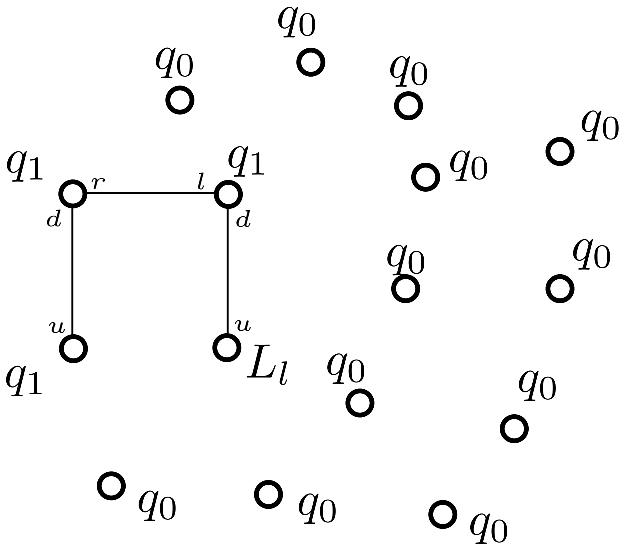
q_0
q_0
q_0
L_r
r
l
q_0
q_0
d
q_0
q_0
q_0
u
q_1
q_0
q_0
q_0
q_0
q_0
q_0

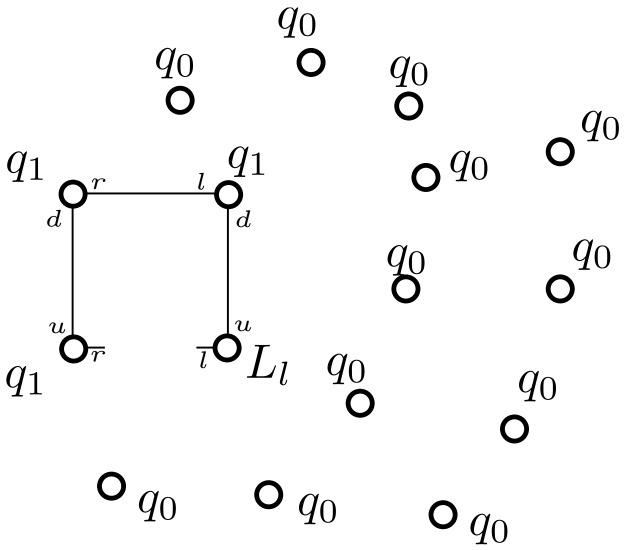


q_0
q_0
q_0
q_1
L_d
q_0
r
l
d
q_0
q_0
q_0
u
q_1
q_0
q_0
q_0
q_0
q_0
q_0

Spanning-Square Protocol

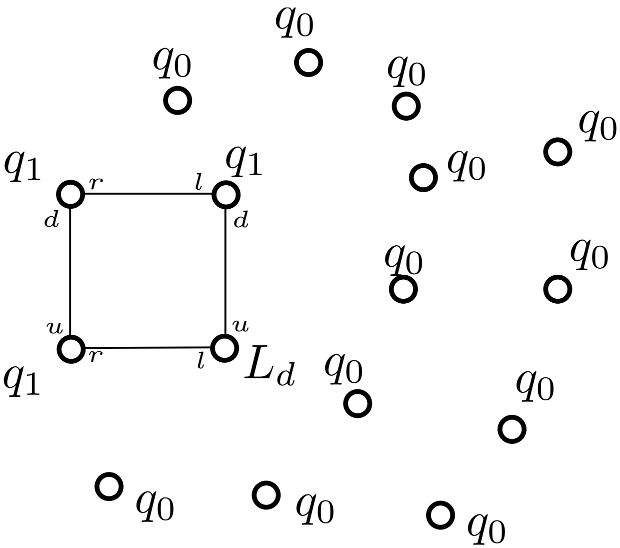



q_0
q_0
q_0
q_0
q_1
q_1
r
l
q_0
d
d
q_0
q_0
u
u
q_1
L_l
q_0
q_0
q_0
q_0
q_0

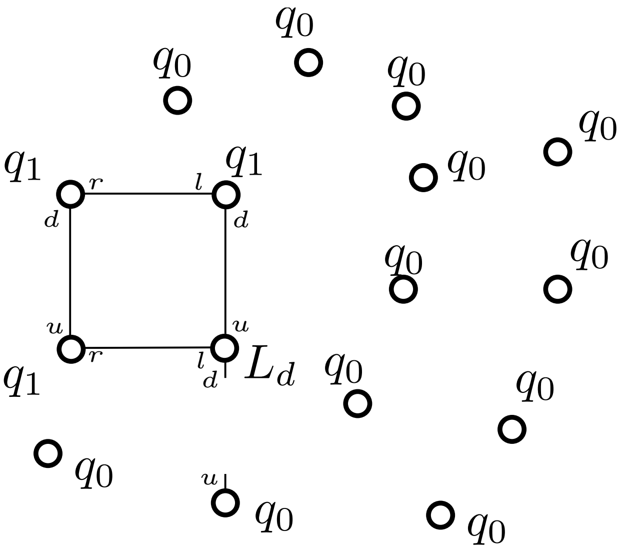


q_0
q_0
q_0
q_0
q_1
r
l
q_1
q_0
d
d
q_0
q_0
u
u
$\overline{r}$
$\overline{l}$
L_l
q_1
q_0
q_0
q_0
q_0
q_0

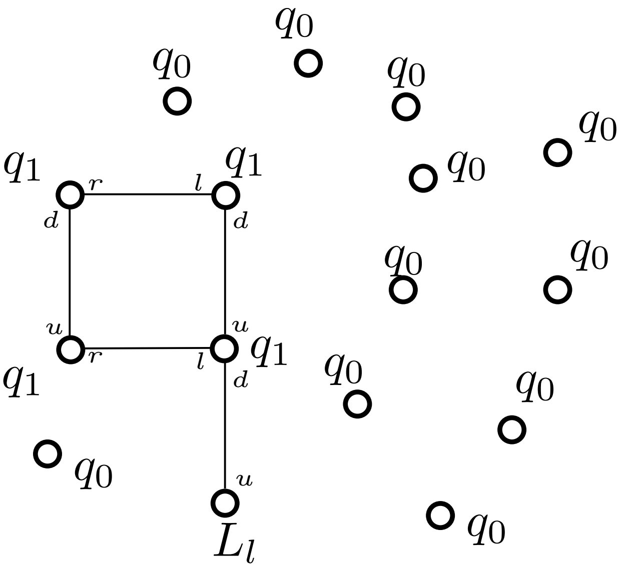
Spanning-Square Protocol



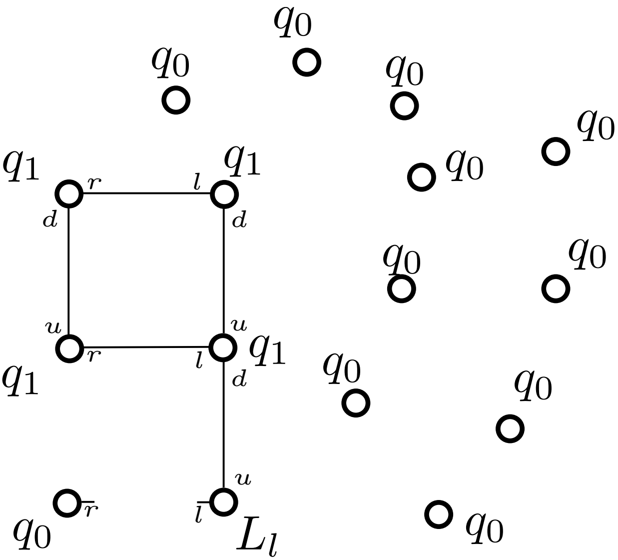
Spanning-Square Protocol



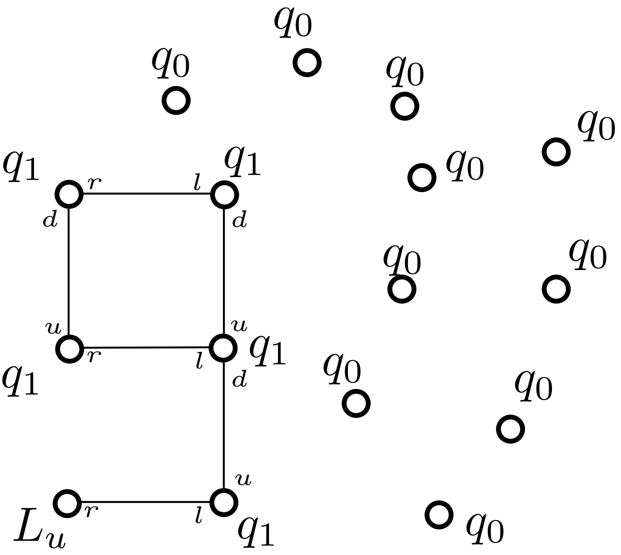
Spanning-Square Protocol



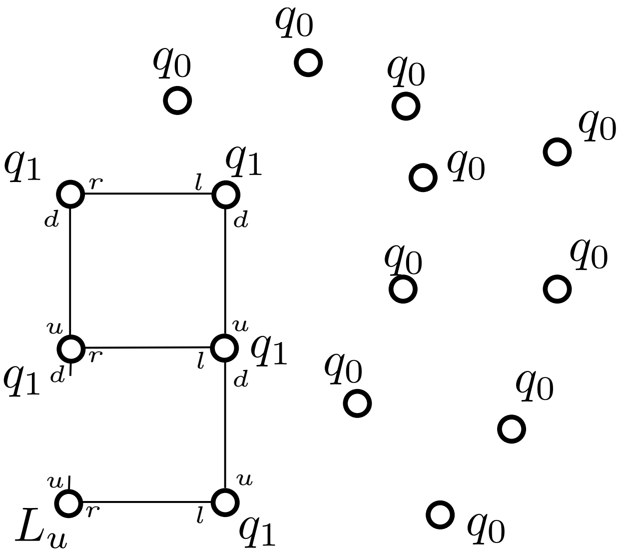
Spanning-Square Protocol



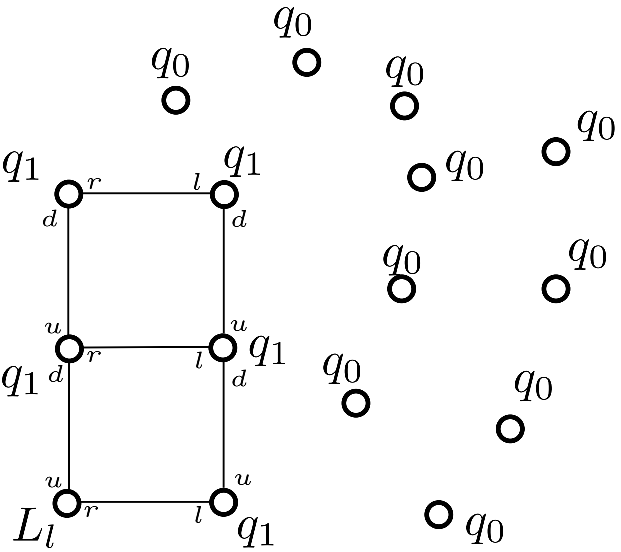
Spanning-Square Protocol

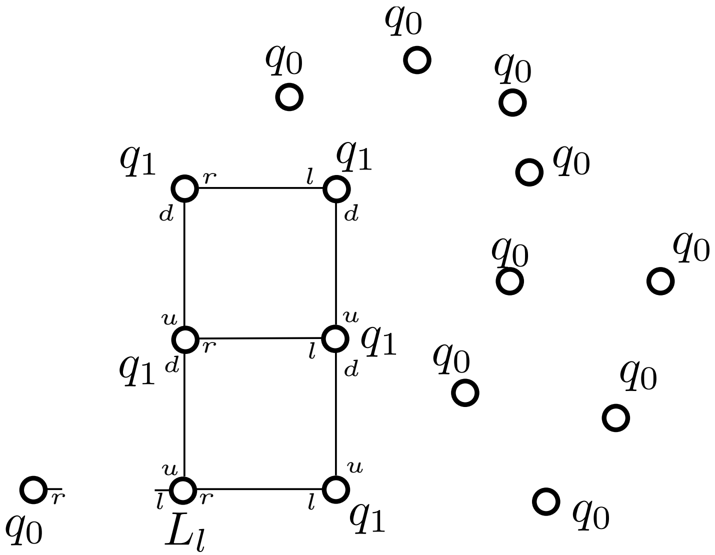


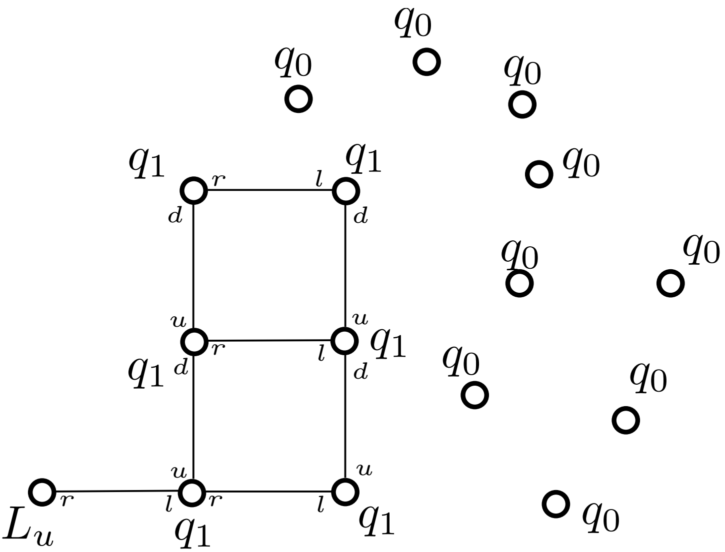
Spanning-Square Protocol

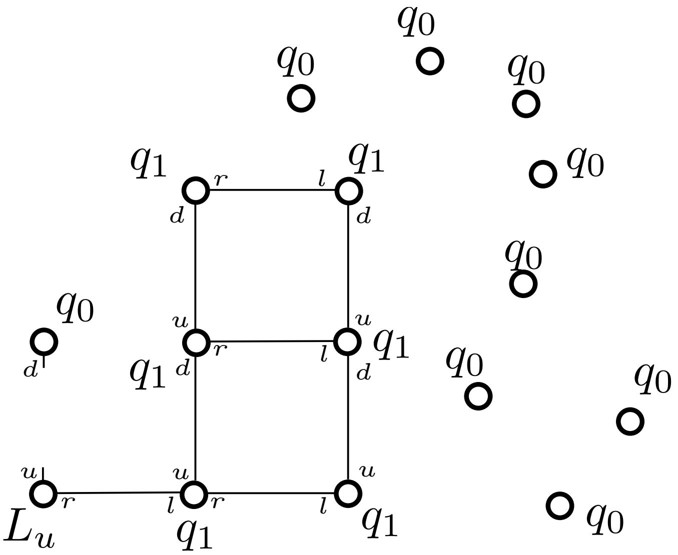


Spanning-Square Protocol



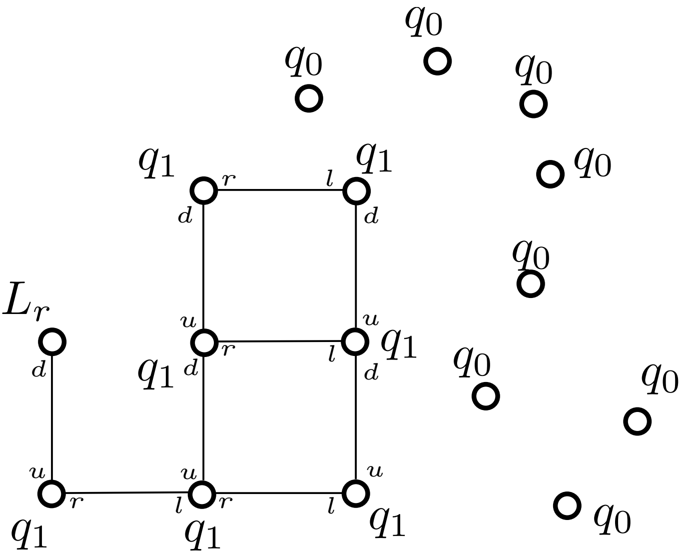

q_0
q_0
q_0
q_1
q_1
q_0
r
l
d
d
q_0
q_0
u
u
q_1
r
l
q_1
d
d
q_0
q_0
u
u
l
r
l
r
q_0
L_l
q_1
q_0


q_0
q_0
q_0
q_0
q_1
q_1
r
l
d
d
q_0
q_0
u
u
q_1
r
l
q_1
d
d
q_0
q_0
u
u
r
l
r
l
L_u
q_1
q_1
q_0

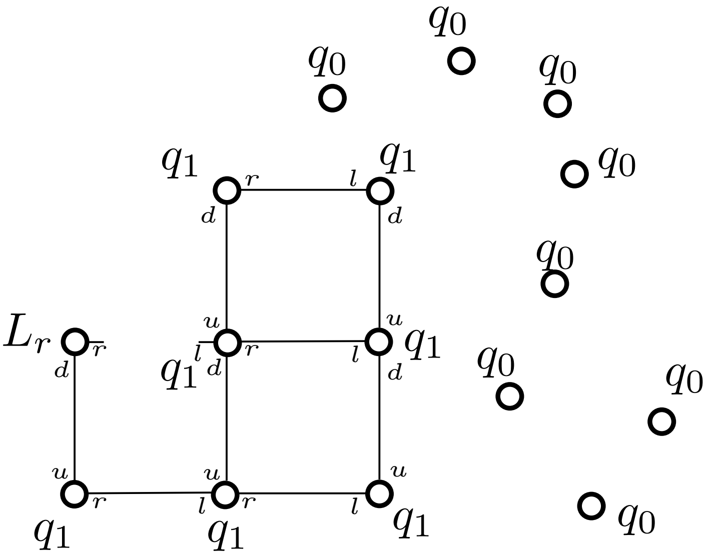


q_0
q_0
q_0
q_0
q_1
q_1
r
l
d
d
q_0
q_0
u
u
q_1
r
l
d
q_1
d
d
q_0
q_0
u
u
u
r
l
r
l
L_u
q_1
q_1
q_0

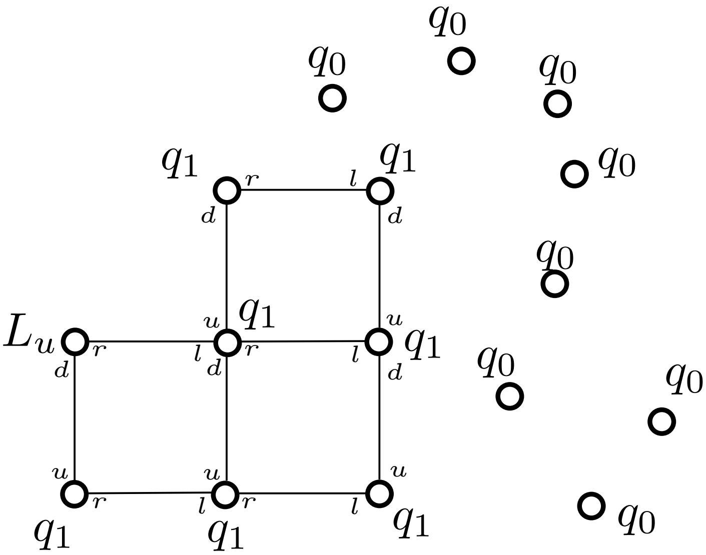
Spanning-Square Protocol



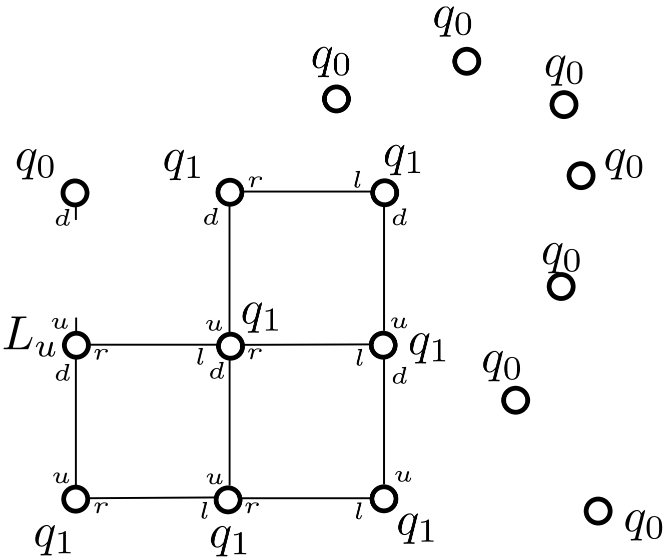
Spanning-Square Protocol



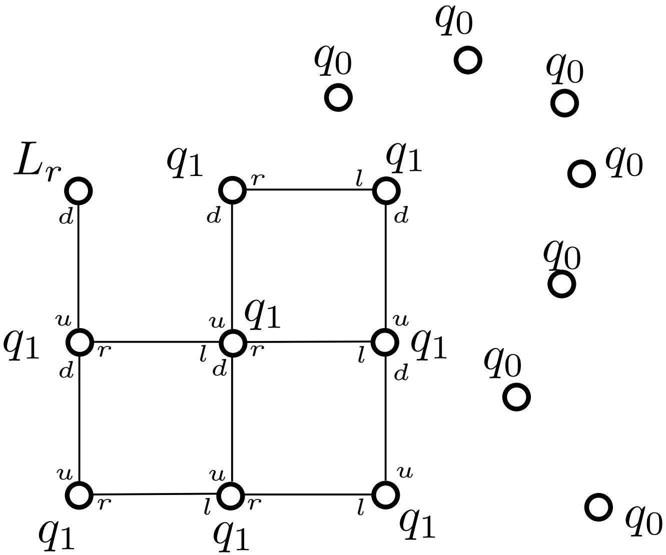
Spanning-Square Protocol

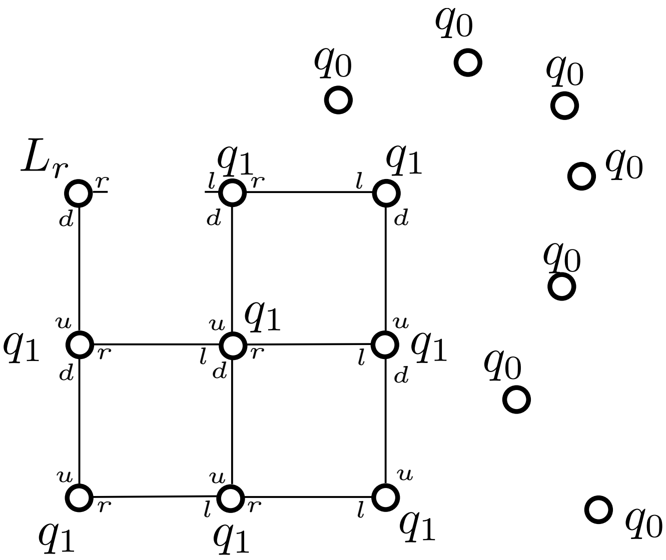


Spanning-Square Protocol



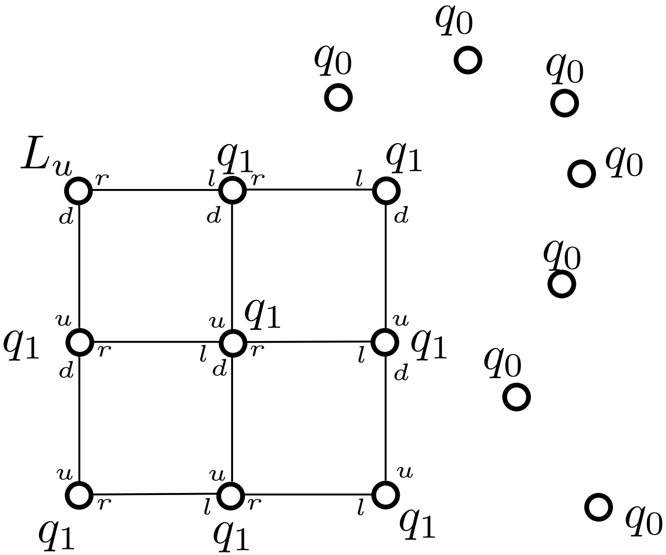
Spanning-Square Protocol

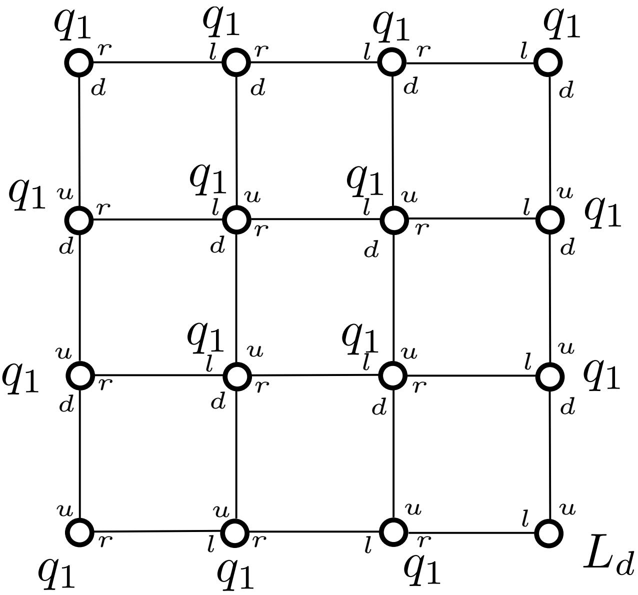




q_0
q_0
q_0
q_0
q_0
q_0
q_0
L_r
q_1
q_1
q_1
q_1
q_1
q_1
q_1
q_1
r
l
d
u

Spanning-Square Protocol




q_1
r
l
d
u
L_d

Probabilistic Counting

- Uniform random scheduler
- Terminating protocol that counts n (or a large fraction of n) w.h.p.
- The best that we can hope for
  - There is no protocol that always terminates and is always correct
- Main subroutine of generic constructors
- Allows for sequential composition and avoids perpetual reinitializations

- Uniform random scheduler
- Terminating protocol that counts n (or a large fraction of n) w.h.p.
- The best that we can hope for
  - There is no protocol that always terminates and is always correct
- Main subroutine of generic constructors
- Allows for sequential composition and avoids perpetual reinitializations

Probabilistic Counting

- Uniform random scheduler
- Terminating protocol that counts n (or a large fraction of n) w.h.p.
- The best that we can hope for
  - There is no protocol that always terminates and is always correct
- Main subroutine of generic constructors
- Allows for sequential composition and avoids perpetual reinitializations

Probabilistic Counting

- Uniform random scheduler
- Terminating protocol that counts n (or a large fraction of n) w.h.p.
- The best that we can hope for
  - There is no protocol that always terminates and is always correct
- Main subroutine of generic constructors
- Allows for sequential composition and avoids perpetual reinitializations

Probabilistic Counting

- Uniform random scheduler
- Terminating protocol that counts n (or a large fraction of n) w.h.p.
- The best that we can hope for
  - There is no protocol that always terminates and is always correct
- Main subroutine of generic constructors
- Allows for sequential composition and avoids perpetual reinitializations

Fast Probabilistic Counting with a Leader

- Unique leader l, all other in q_0 initially
- Disregard for a while ports, geometry, and link activations
- The scheduler selects in every step equiprobably one of the $n(n-1)/2$ possible node pairs
- Assume for simplicity that the leader can store two n-counters in its memory
- Classical PP with an additional leader with linear memory
- Then, the protocol is adjusted to work in the present model
  - Only keep the unique leader assumption (but drop its memory)

Fast Probabilistic Counting with a Leader

- Unique leader l, all other in q_0 initially
- Disregard for a while ports, geometry, and link activations
- The scheduler selects in every step equiprobably one of the $n(n-1)/2$ possible node pairs
- Assume for simplicity that the leader can store two n-counters in its memory
- Classical PP with an additional leader with linear memory
- Then, the protocol is adjusted to work in the present model
  - Only keep the unique leader assumption (but drop its memory)

Fast Probabilistic Counting with a Leader

- Unique leader l, all other in q_0 initially
- Disregard for a while ports, geometry, and link activations
- The scheduler selects in every step equiprobably one of the $n(n-1)/2$ possible node pairs
- Assume for simplicity that the leader can store two n-counters in its memory
- Classical PP with an additional leader with linear memory
- Then, the protocol is adjusted to work in the present model
  - Only keep the unique leader assumption (but drop its memory)

Fast Probabilistic Counting with a Leader

- Unique leader l, all other in q_0 initially
- Disregard for a while ports, geometry, and link activations
- The scheduler selects in every step equiprobably one of the $n(n-1)/2$ possible node pairs
- Assume for simplicity that the leader can store two n-counters in its memory
- Classical PP with an additional leader with linear memory
- Then, the protocol is adjusted to work in the present model
  - Only keep the unique leader assumption (but drop its memory)

Fast Probabilistic Counting with a Leader

- Unique leader l, all other in q_0 initially
- Disregard for a while ports, geometry, and link activations
- The scheduler selects in every step equiprobably one of the $n(n-1)/2$ possible node pairs
- Assume for simplicity that the leader can store two n-counters in its memory
- Classical PP with an additional leader with linear memory
- Then, the protocol is adjusted to work in the present model
  - Only keep the unique leader assumption (but drop its memory)

Fast Probabilistic Counting with a Leader

- Unique leader l, all other in q_0 initially
- Disregard for a while ports, geometry, and link activations
- The scheduler selects in every step equiprobably one of the $n(n-1)/2$ possible node pairs
- Assume for simplicity that the leader can store two n-counters in its memory
- Classical PP with an additional leader with linear memory
- Then, the protocol is adjusted to work in the present model
  - Only keep the unique leader assumption (but drop its memory)

Counting-Upper-Bound Protocol

- $I(r_0, r_1)$: The state of I, where r_0, r_1 are the values of the two counters, $0 \leq r_0, r_1 \leq n$
- Rules:

$$(I(r_0, r_1), q_0) \rightarrow (I(r_0 + 1, r_1), q_1)$$
$$(I(r_0, r_1), q_1) \rightarrow (I(r_0, r_1 + 1), q_2), \text{ and}$$
$$(I(r_0, r_1), \cdot) \rightarrow (halt, \cdot) \text{ if } r_0 = r_1$$

- r_0 counts the number of q_0s in the population
- r_1 counts the number of q_1s in the population
- When a q_0 (q_1) is counted it is converted to q_1 (q_2)
- Terminates when $r_0 = r_1$ for the first time

- $I(r_0, r_1)$: The state of I, where r_0, r_1 are the values of the two counters, $0 \leq r_0, r_1 \leq n$
- Rules:

$$(I(r_0, r_1), q_0) \rightarrow (I(r_0 + 1, r_1), q_1)$$
$$(I(r_0, r_1), q_1) \rightarrow (I(r_0, r_1 + 1), q_2), \text{ and}$$
$$(I(r_0, r_1), \cdot) \rightarrow (halt, \cdot) \text{ if } r_0 = r_1$$

- r_0 counts the number of q_0s in the population
- r_1 counts the number of q_1s in the population
- When a q_0 (q_1) is counted it is converted to q_1 (q_2)
- Terminates when $r_0 = r_1$ for the first time

Counting-Upper-Bound Protocol

- $I(r_0, r_1)$: The state of I, where r_0, r_1 are the values of the two counters, $0 \leq r_0, r_1 \leq n$
- Rules:

$$
\begin{gathered}
(I(r_0, r_1), q_0) \to (I(r_0 + 1, r_1), q_1) \\
(I(r_0, r_1), q_1) \to (I(r_0, r_1 + 1), q_2), \text{ and} \\
(I(r_0, r_1), \cdot) \to (halt, \cdot) \text{ if } r_0 = r_1
\end{gathered}
$$

- r_0 counts the number of q_0s in the population
- r_1 counts the number of q_1s in the population
- When a q_0 (q_1) is counted it is converted to q_1 (q_2)
- Terminates when $r_0 = r_1$ for the first time

Counting-Upper-Bound Protocol

- $I(r_0, r_1)$: The state of I, where r_0, r_1 are the values of the two counters, $0 \le r_0, r_1 \le n$
- Rules:

$$
\begin{aligned}
(I(r_0, r_1), q_0) &\to (I(r_0 + 1, r_1), q_1) \\
(I(r_0, r_1), q_1) &\to (I(r_0, r_1 + 1), q_2), \text{ and} \\
(I(r_0, r_1), \cdot) &\to (halt, \cdot) \text{ if } r_0 = r_1
\end{aligned}
$$

- r_0 counts the number of q_0s in the population
- r_1 counts the number of q_1s in the population
- When a q_0 (q_1) is counted it is converted to q_1 (q_2)
- Terminates when $r_0 = r_1$ for the first time

Counting-Upper-Bound Protocol

- $I(r_0, r_1)$: The state of I, where r_0, r_1 are the values of the two counters, $0 \leq r_0, r_1 \leq n$
- Rules:

$$
\begin{gathered}
(I(r_0, r_1), q_0) \rightarrow (I(r_0 + 1, r_1), q_1) \\
(I(r_0, r_1), q_1) \rightarrow (I(r_0, r_1 + 1), q_2), \text{ and} \\
(I(r_0, r_1), \cdot) \rightarrow (halt, \cdot) \text{ if } r_0 = r_1
\end{gathered}
$$

- r_0 counts the number of q_0s in the population
- r_1 counts the number of q_1s in the population
- When a q_0 (q_1) is counted it is converted to q_1 (q_2)
- Terminates when $r_0 = r_1$ for the first time

Counting-Upper-Bound Protocol

- $I(r_0, r_1)$: The state of I, where r_0, r_1 are the values of the two counters, $0 \leq r_0, r_1 \leq n$
- Rules:

$$(I(r_0, r_1), q_0) \to (I(r_0 + 1, r_1), q_1)$$
$$(I(r_0, r_1), q_1) \to (I(r_0, r_1 + 1), q_2), \text{ and}$$
$$(I(r_0, r_1), \cdot) \to (halt, \cdot) \text{ if } r_0 = r_1$$

- r_0 counts the number of q_0s in the population
- r_1 counts the number of q_1s in the population
- When a q_0 (q_1) is counted it is converted to q_1 (q_2)
- Terminates when $r_0 = r_1$ for the first time

Counting-Upper-Bound Protocol

Theorem

Counting-Upper-Bound halts in every execution. Moreover, if the scheduler is a uniform random one, when this occurs, w.h.p. it holds that $r_0 \geq n/2$.

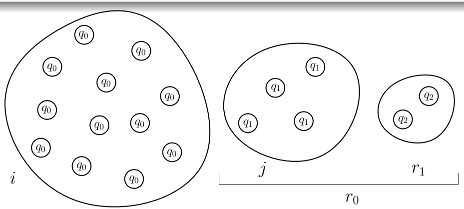
Proof

- $p_{ij} = i/(i+j)$: *probability that an effective interaction is an* (l, q_0)
- $q_{ij} = 1 - p_{ij} = j/(i+j)$: *probability that it is an* (l, q_1)
- *r.w. on a line with* $n+1$ *positions* $0, 1, \ldots, n$
- *a particle begins from position* b, *absorbing barrier at* 0, *and reflecting at* n, *position corresponds to* $r_0 - r_1 = j$

Counting-Upper-Bound Protocol

Theorem

Counting-Upper-Bound halts in every execution. Moreover, if the scheduler is a uniform random one, when this occurs, w.h.p. it holds that $r_0 \geq n/2$.



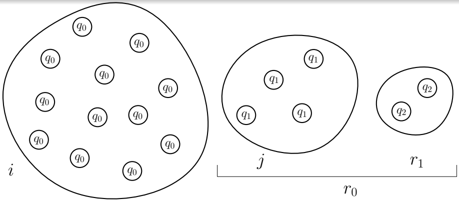
Proof

- $p_{ij} = i/(i+j)$: *probability that an effective interaction is an* (l, q_0)
- $q_{ij} = 1 - p_{ij} = j/(i+j)$: *probability that it is an* (l, q_1)
- *r.w. on a line with* $n+1$ *positions* $0, 1, \ldots, n$
- *a particle begins from position b, absorbing barrier at* 0, *and reflecting at n, position corresponds to* $r_0 - r_1 = j$

Counting-Upper-Bound Protocol

Theorem

Counting-Upper-Bound halts in every execution. Moreover, if the scheduler is a uniform random one, when this occurs, w.h.p. it holds that $r_0 \geq n/2$.



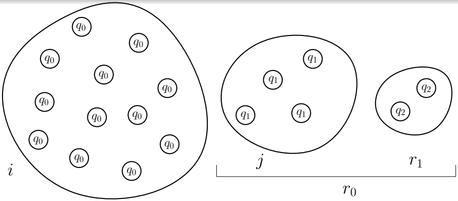
Proof

- $p_{ij} = i/(i+j)$: *probability that an effective interaction is an* (l, q_0)
- $q_{ij} = 1 - p_{ij} = j/(i+j)$: *probability that it is an* (l, q_1)
- *r.w. on a line with* $n+1$ *positions* $0, 1, \ldots, n$
- *a particle begins from position* b*, absorbing barrier at* 0*, and reflecting at* n*, position corresponds to* $r_0 - r_1 = j$

Counting-Upper-Bound Protocol

Theorem

Counting-Upper-Bound halts in every execution. Moreover, if the scheduler is a uniform random one, when this occurs, w.h.p. it holds that $r_0 \geq n/2$.



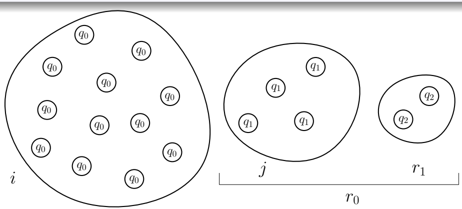
Proof

- $p_{ij} = i/(i+j)$: *probability that an effective interaction is an* (l, q_0)
- $q_{ij} = 1 - p_{ij} = j/(i+j)$: *probability that it is an* (l, q_1)
- *r.w. on a line with* $n+1$ *positions* $0, 1, \ldots, n$
- *a particle begins from position b, absorbing barrier at 0, and reflecting at n, position corresponds to* $r_0 - r_1 = j$

Counting-Upper-Bound Protocol

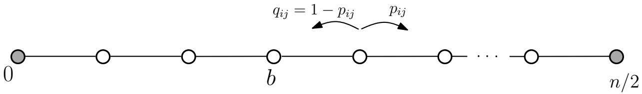
Theorem

Counting-Upper-Bound halts in every execution. Moreover, if the scheduler is a uniform random one, when this occurs, w.h.p. it holds that $r_0 \geq n/2$.



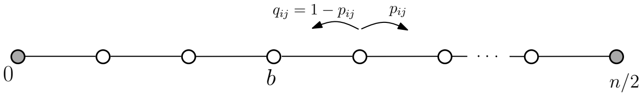
Proof

- $p_{ij} = i/(i+j)$: *probability that an effective interaction is an* (l, q_0)
- $q_{ij} = 1 - p_{ij} = j/(i+j)$: *probability that it is an* (l, q_1)
- *r.w. on a line with* $n+1$ *positions* $0, 1, \ldots, n$
- *a particle begins from position* b, *absorbing barrier at* 0, *and reflecting at* n, *position corresponds to* $r_0 - r_1 = j$



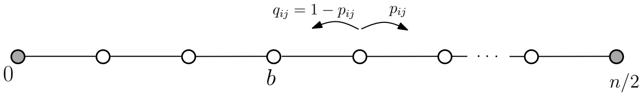
Proof

- *"difficult" r.w.: the transition probabilities depend on the position j and also on $i + j$ which decreases in time*
- *upper bound* $\mathrm{P}[$*failure*$] = \mathrm{P}[$*reach 0 before* $r_0 \geq n/2$ *holds*$] \leq$
- *reduce it to a r.w. that does not depend on* $i + j$
  - *Ehrenfest r.w.*
- $r_0 + r_1 \leq n$ *but* $r_1 \leq r_0 \Rightarrow 2r_1 \leq r_0 + r_1$, *thus*
- $2r_1 \leq n \Rightarrow r_1 \leq n/2 \Rightarrow (n-1)-(i+j) \leq n/2 \Rightarrow i+j \geq (n/2)-1 = n'$



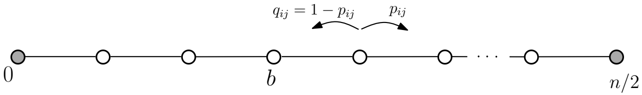
Proof

- *"difficult" r.w.: the transition probabilities depend on the position j and also on $i+j$ which decreases in time*
- *upper bound* $\mathrm{P}[\textit{failure}] = \mathrm{P}[\textit{reach 0 before } r_0 \geq n/2 \textit{ holds}] \leq$
- *reduce it to a r.w. that does not depend on $i+j$*
  - *Ehrenfest r.w.*
- $r_0 + r_1 \leq n$ *but* $r_1 \leq r_0 \Rightarrow 2r_1 \leq r_0 + r_1$, *thus*
- $2r_1 \leq n \Rightarrow r_1 \leq n/2 \Rightarrow (n-1)-(i+j) \leq n/2 \Rightarrow i+j \geq (n/2)-1 = n'$



Proof

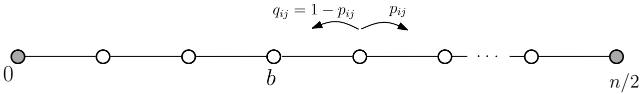
- *"difficult" r.w.: the transition probabilities depend on the position j and also on $i+j$ which decreases in time*
- *upper bound* $\mathrm{P}[\textit{failure}] = \mathrm{P}[\textit{reach 0 before } r_0 \geq n/2 \textit{ holds}] \leq$
- *reduce it to a r.w. that does not depend on $i+j$*
  - *Ehrenfest r.w.*
- $r_0 + r_1 \leq n$ *but* $r_1 \leq r_0 \Rightarrow 2r_1 \leq r_0 + r_1$, *thus*
- $2r_1 \leq n \Rightarrow r_1 \leq n/2 \Rightarrow (n-1)-(i+j) \leq n/2 \Rightarrow i+j \geq (n/2)-1 = n'$



Proof

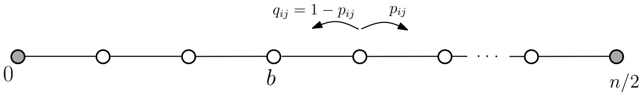
- *"difficult" r.w.: the transition probabilities depend on the position j and also on $i + j$ which decreases in time*
- *upper bound* $\mathrm{P}[\textit{failure}] = \mathrm{P}[\textit{reach 0 before } r_0 \geq n/2 \textit{ holds}] \leq$
- *reduce it to a r.w. that does not depend on $i + j$*
  - *Ehrenfest r.w.*
- $r_0 + r_1 \leq n$ *but* $r_1 \leq r_0 \Rightarrow 2r_1 \leq r_0 + r_1$, *thus*
- $2r_1 \leq n \Rightarrow r_1 \leq n/2 \Rightarrow (n-1)-(i+j) \leq n/2 \Rightarrow i+j \geq (n/2)-1 = n'$

Counting-Upper-Bound Protocol



Proof

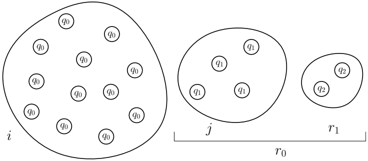
- *"difficult" r.w.: the transition probabilities depend on the position j and also on $i + j$ which decreases in time*
- *upper bound* $\mathrm{P}[\textit{failure}] = \mathrm{P}[\textit{reach 0 before } r_0 \geq n/2 \textit{ holds}] \leq$
- *reduce it to a r.w. that does not depend on $i + j$*
  - *Ehrenfest r.w.*
- $r_0 + r_1 \leq n$ *but* $r_1 \leq r_0 \Rightarrow 2r_1 \leq r_0 + r_1$, *thus*
- $2r_1 \leq n \Rightarrow r_1 \leq n/2 \Rightarrow (n-1)-(i+j) \leq n/2 \Rightarrow i+j \geq (n/2)-1 = n'$



Proof

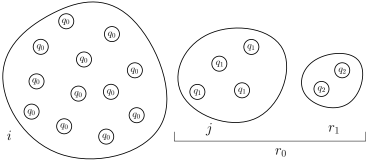
- *"difficult" r.w.: the transition probabilities depend on the position j and also on $i + j$ which decreases in time*
- *upper bound* $P[\textit{failure}] = P[\textit{reach 0 before } r_0 \geq n/2 \textit{ holds}] \leq$
- *reduce it to a r.w. that does not depend on $i + j$*
  - *Ehrenfest r.w.*
- $r_0 + r_1 \leq n$ *but* $r_1 \leq r_0 \Rightarrow 2r_1 \leq r_0 + r_1$, *thus*
- $2r_1 \leq n \Rightarrow r_1 \leq n/2 \Rightarrow (n-1)-(i+j) \leq n/2 \Rightarrow i+j \geq (n/2)-1 = n'$

Counting-Upper-Bound Protocol



Proof

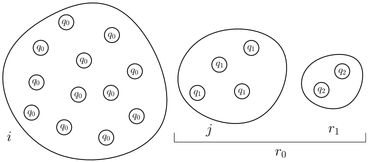
- *If we set $n' = (n/2) - 1$ we have $i + j \geq n'$*
- *When $r_0 + r_1 = n + 1$ we have $n + 1 = r_0 + r_1 \leq 2r_0 \Rightarrow r_0 \geq n/2$*
- *During the first n effective interactions: $i + j \geq n' = (n/2) - 1$*
- *When interaction $n + 1$ occurs: $r_0 \geq n/2$*
- *If the process is still alive after time n, then r_0 has managed to count up to $n/2$*



Proof

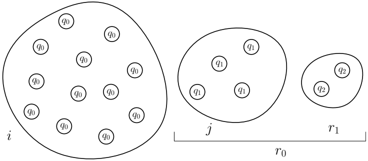
- *If we set $n' = (n/2) - 1$ we have $i + j \geq n'$*
- *When $r_0 + r_1 = n + 1$ we have $n + 1 = r_0 + r_1 \leq 2r_0 \Rightarrow r_0 \geq n/2$*
- *During the first n effective interactions: $i + j \geq n' = (n/2) - 1$*
- *When interaction $n + 1$ occurs: $r_0 \geq n/2$*
- *If the process is still alive after time n, then r_0 has managed to count up to $n/2$*

Counting-Upper-Bound Protocol



Proof

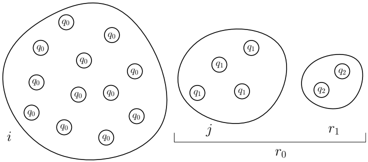
- *If we set* $n' = (n/2) - 1$ *we have* $i + j \geq n'$
- *When* $r_0 + r_1 = n + 1$ *we have* $n + 1 = r_0 + r_1 \leq 2r_0 \Rightarrow r_0 \geq n/2$
- *During the first* n *effective interactions:* $i + j \geq n' = (n/2) - 1$
- *When interaction* $n + 1$ *occurs:* $r_0 \geq n/2$
- *If the process is still alive after time* n*, then* r_0 *has managed to count up to* $n/2$



Proof

- *If we set $n' = (n/2) - 1$ we have $i + j \geq n'$*
- *When $r_0 + r_1 = n + 1$ we have $n + 1 = r_0 + r_1 \leq 2r_0 \Rightarrow r_0 \geq n/2$*
- *During the first n effective interactions: $i + j \geq n' = (n/2) - 1$*
- *When interaction $n + 1$ occurs: $r_0 \geq n/2$*
- *If the process is still alive after time n, then r_0 has managed to count up to $n/2$*

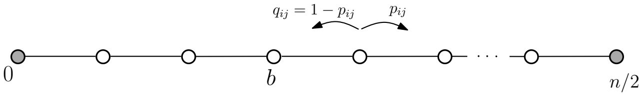
Counting-Upper-Bound Protocol



Proof

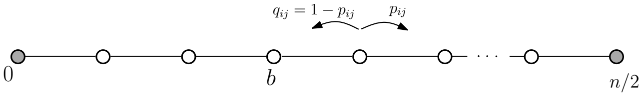
- *If we set $n' = (n/2) - 1$ we have $i + j \geq n'$*
- *When $r_0 + r_1 = n + 1$ we have $n + 1 = r_0 + r_1 \leq 2r_0 \Rightarrow r_0 \geq n/2$*
- *During the first n effective interactions: $i + j \geq n' = (n/2) - 1$*
- *When interaction $n + 1$ occurs: $r_0 \geq n/2$*
- *If the process is still alive after time n, then r_0 has managed to count up to $n/2$*

Counting-Upper-Bound Protocol



Proof

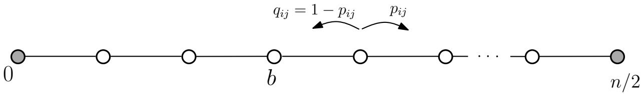
- $i + j \geq n'$ *implies that* $p_j \geq (n' - j)/n'$ *and* $q_j \leq j/n'$
- *Now the probabilities only depend on the position* j
- *Ehrenfest random walk from the theory of brownian motion [EE1907]*
  - *Gas molecules moving randomly in a container, divided into two urns*
- *Prob. counting fails asympt. equivalent to prob. urn I becomes empty in the first n steps, assuming it initially contains b molecules*
- *The mean recurrence time is* $((R+k)!(R-k)!/(2R)!)2^{2R}$ *[Ka47]*
  - *For* $k = -R$, *initial position is* $R + k = 0$ *and gives* $2^{2R} = 2^{n/2}$
  - *In the sequel, we turn this into the desired high probability argument*



Proof

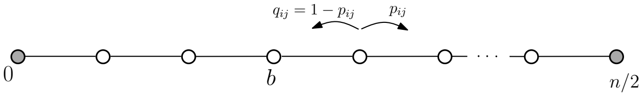
- $i + j \geq n'$ implies that $p_j \geq (n' - j)/n'$ and $q_j \leq j/n'$
- Now the probabilities only depend on the position j
- Ehrenfest random walk from the theory of brownian motion [EE1907]
  - Gas molecules moving randomly in a container, divided into two urns
- Prob. counting fails asympt. equivalent to prob. urn I becomes empty in the first n steps, assuming it initially contains b molecules
- The mean recurrence time is $((R + k)!(R - k)!/(2R)!)2^{2R}$ [Ka47]
  - For $k = -R$, initial position is $R + k = 0$ and gives $2^{2R} = 2^{n/2}$
  - In the sequel, we turn this into the desired high probability argument

Counting-Upper-Bound Protocol



Proof

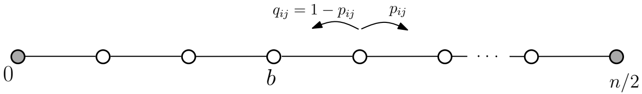
- $i + j \geq n'$ implies that $p_j \geq (n' - j)/n'$ and $q_j \leq j/n'$
- *Now the probabilities only depend on the position j*
- *Ehrenfest random walk from the theory of brownian motion [EE1907]*
    - *Gas molecules moving randomly in a container, divided into two urns*
- *Prob. counting fails asympt. equivalent to prob. urn I becomes empty in the first n steps, assuming it initially contains b molecules*
- *The mean recurrence time is $((R+k)!(R-k)!/(2R)!)2^{2R}$ [Ka47]*
    - *For $k = -R$, initial position is $R + k = 0$ and gives $2^{2R} = 2^{n/2}$*
    - *In the sequel, we turn this into the desired high probability argument*



Proof

- $i + j \geq n'$ implies that $p_j \geq (n' - j)/n'$ and $q_j \leq j/n'$
- Now the probabilities only depend on the position j
- Ehrenfest random walk from the theory of brownian motion [EE1907]
  - Gas molecules moving randomly in a container, divided into two urns
- Prob. counting fails asympt. equivalent to prob. urn I becomes empty in the first n steps, assuming it initially contains b molecules
- The mean recurrence time is $((R+k)!(R-k)!/(2R)!)2^{2R}$ [Ka47]
  - For $k = -R$, initial position is $R + k = 0$ and gives $2^{2R} = 2^{n/2}$
  - In the sequel, we turn this into the desired high probability argument

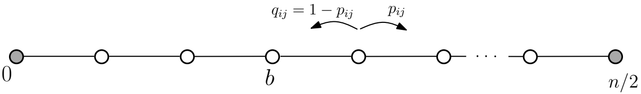
Counting-Upper-Bound Protocol



Proof

- $i + j \geq n'$ implies that $p_j \geq (n' - j)/n'$ and $q_j \leq j/n'$
- *Now the probabilities only depend on the position j*
- *Ehrenfest random walk from the theory of brownian motion [EE1907]*
  - *Gas molecules moving randomly in a container, divided into two urns*
- *Prob. counting fails asympt. equivalent to prob. urn I becomes empty in the first n steps, assuming it initially contains b molecules*
- *The mean recurrence time is $((R+k)!(R-k)!/(2R)!)2^{2R}$ [Ka47]*
  - *For $k = -R$, initial position is $R + k = 0$ and gives $2^{2R} = 2^{n/2}$*
  - *In the sequel, we turn this into the desired high probability argument*

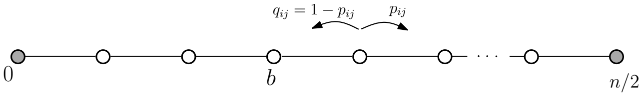
Counting-Upper-Bound Protocol



Proof

- $i + j \geq n'$ implies that $p_j \geq (n' - j)/n'$ and $q_j \leq j/n'$
- Now the probabilities only depend on the position j
- Ehrenfest random walk from the theory of brownian motion [EE1907]
  - Gas molecules moving randomly in a container, divided into two urns
- Prob. counting fails asympt. equivalent to prob. urn I becomes empty in the first n steps, assuming it initially contains b molecules
- The mean recurrence time is $((R + k)!(R - k)!/(2R)!)2^{2R}$ [Ka47]
  - For $k = -R$, initial position is $R + k = 0$ and gives $2^{2R} = 2^{n/2}$
  - In the sequel, we turn this into the desired high probability argument

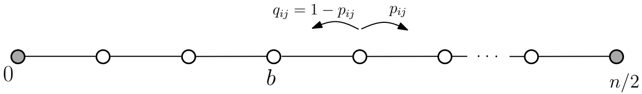
Counting-Upper-Bound Protocol



Proof

- *$i + j \geq n'$ implies that $p_j \geq (n' - j)/n'$ and $q_j \leq j/n'$*
- *Now the probabilities only depend on the position j*
- *Ehrenfest random walk from the theory of brownian motion [EE1907]*
  - *Gas molecules moving randomly in a container, divided into two urns*
- *Prob. counting fails asympt. equivalent to prob. urn I becomes empty in the first n steps, assuming it initially contains b molecules*
- *The mean recurrence time is $((R+k)!(R-k)!/(2R)!)2^{2R}$ [Ka47]*
  - *For $k = -R$, initial position is $R + k = 0$ and gives $2^{2R} = 2^{n/2}$*
  - *In the sequel, we turn this into the desired high probability argument*

Counting-Upper-Bound Protocol



Proof

- $i + j \geq n'$ *implies that* $p_j \geq (n' - j)/n'$ *and* $q_j \leq j/n'$
- *Now the probabilities only depend on the position* j
- *Ehrenfest random walk from the theory of brownian motion [EE1907]*
  - *Gas molecules moving randomly in a container, divided into two urns*
- *Prob. counting fails asympt. equivalent to prob. urn I becomes empty in the first* n *steps, assuming it initially contains* b *molecules*
- *The mean recurrence time is* $((R + k)!(R - k)!/(2R)!)2^{2R}$ *[Ka47]*
  - *For* $k = -R$*, initial position is* $R + k = 0$ *and gives* $2^{2R} = 2^{n/2}$
  - *In the sequel, we turn this into the desired high probability argument*

Counting-Upper-Bound Protocol

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix $[0, b]$ of the line*
- *It holds that $j \leq b$, implying that $p \geq (n' - b)/n'$ and $q \leq b/n'$*
- *Set $p = (n' - b)/n'$ and $q = b/n'$*
    - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on $b - 1$ it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with $b + 1$ positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability $p' = q$, backward with probability $q' = p$, and fails at b*

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix $[0, b]$ of the line*
- *It holds that $j \leq b$, implying that $p \geq (n' - b)/n'$ and $q \leq b/n'$*
- *Set $p = (n' - b)/n'$ and $q = b/n'$*
  - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on $b - 1$ it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with $b + 1$ positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability $p' = q$, backward with probability $q' = p$, and fails at b*

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix* $[0, b]$ *of the line*
- *It holds that* $j \leq b$, *implying that* $p \geq (n' - b)/n'$ *and* $q \leq b/n'$
- *Set* $p = (n' - b)/n'$ *and* $q = b/n'$
    - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on* $b - 1$ *it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with* $b + 1$ *positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability* $p' = q$, *backward with probability* $q' = p$, *and fails at b*

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix* $[0, b]$ *of the line*
- *It holds that* $j \leq b$*, implying that* $p \geq (n' - b)/n'$ *and* $q \leq b/n'$
- *Set* $p = (n' - b)/n'$ *and* $q = b/n'$
  - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on* $b - 1$ *it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with* $b + 1$ *positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability* $p' = q$*, backward with probability* $q' = p$*, and fails at b*

Counting-Upper-Bound Protocol

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix $[0, b]$ of the line*
- *It holds that $j \leq b$, implying that $p \geq (n' - b)/n'$ and $q \leq b/n'$*
- *Set $p = (n' - b)/n'$ and $q = b/n'$*
    - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on $b - 1$ it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with $b + 1$ positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability $p' = q$, backward with probability $q' = p$, and fails at b*

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix $[0, b]$ of the line*
- *It holds that $j \leq b$, implying that $p \geq (n' - b)/n'$ and $q \leq b/n'$*
- *Set $p = (n' - b)/n'$ and $q = b/n'$*
  - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on $b - 1$ it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with $b + 1$ positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability $p' = q$, backward with probability $q' = p$, and fails at b*

Counting-Upper-Bound Protocol

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix $[0, b]$ of the line*
- *It holds that $j \leq b$, implying that $p \geq (n' - b)/n'$ and $q \leq b/n'$*
- *Set $p = (n' - b)/n'$ and $q = b/n'$*
    - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on $b - 1$ it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with $b + 1$ positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability $p' = q$, backward with probability $q' = p$, and fails at b*

Counting-Upper-Bound Protocol

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix* $[0, b]$ *of the line*
- *It holds that* $j \leq b$, *implying that* $p \geq (n' - b)/n'$ *and* $q \leq b/n'$
- *Set* $p = (n' - b)/n'$ *and* $q = b/n'$
  - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on* $b - 1$ *it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with* $b + 1$ *positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability* $p' = q$, *backward with probability* $q' = p$, *and fails at b*

Counting-Upper-Bound Protocol

Proof

- *Reduce the Ehrenfest walk to one in which the probabilities do not depend on j*
- *Further restrict the walk, to the prefix* $[0, b]$ *of the line*
- *It holds that* $j \leq b$, *implying that* $p \geq (n' - b)/n'$ *and* $q \leq b/n'$
- *Set* $p = (n' - b)/n'$ *and* $q = b/n'$
  - *This may only increase the probability of failure*
- *Imagine now an absorbing barrier at 0 and another one at b*
- *Whenever the r.w. is on* $b - 1$ *it will either return to b before reaching 0 or it will reach 0 (and fail) before returning to b*
- *A r.w. with* $b + 1$ *positions, where 0 and b are absorbing*
- *Equivalent: begins from position 1, moves forward with probability* $p' = q$, *backward with probability* $q' = p$, *and fails at b*

Proof

- *Bound* P[*reach b before 0 (when beginning from position 1)*]
- *Probability of winning in the classical ruin problem*
- *If we set* $x = q'/p' = p/q = (n' - b)/b$ *we have that:*

$$P[reach\ b\ before\ 0] = 1 - \frac{x^b - x}{x^b - 1} = \frac{x - 1}{x^b - 1} \leq \frac{x}{x^b - 1} \approx \frac{1}{x^{b-1}} \approx \frac{1}{n^{b-1}}.$$

- *Thus, whenever the original walk is on* $b - 1$*, the probability of reaching 0 before reaching b again, is at most* $1/n^{b-1}$
- *Repeat the above walk n times*
  - *Place the particle on* $b - 1$ *and play the game*
  - *If it returns to b, place the particle on* $b - 1$ *and play the game again*

Counting-Upper-Bound Protocol

Proof

- *Bound* P[*reach b before 0 (when beginning from position 1)*]
- *Probability of winning in the classical ruin problem*
- *If we set* $x = q'/p' = p/q = (n' - b)/b$ *we have that:*

$$\mathrm{P}[\textit{reach } b \textit{ before } 0] = 1 - \frac{x^b - x}{x^b - 1} = \frac{x - 1}{x^b - 1} \leq \frac{x}{x^b - 1} \approx \frac{1}{x^{b-1}} \approx \frac{1}{n^{b-1}}.$$

- *Thus, whenever the original walk is on* $b - 1$*, the probability of reaching 0 before reaching b again, is at most* $1/n^{b-1}$
- *Repeat the above walk n times*
  - *Place the particle on* $b - 1$ *and play the game*
  - *If it returns to b, place the particle on* $b - 1$ *and play the game again*

Counting-Upper-Bound Protocol

Proof

- *Bound* P[*reach b before 0 (when beginning from position 1)*]
- *Probability of winning in the classical ruin problem*
- *If we set* $x = q'/p' = p/q = (n' - b)/b$ *we have that:*

$$\mathrm{P}[reach\ b\ before\ 0] = 1 - \frac{x^b - x}{x^b - 1} = \frac{x-1}{x^b - 1} \leq \frac{x}{x^b - 1} \approx \frac{1}{x^{b-1}} \approx \frac{1}{n^{b-1}}.$$

- *Thus, whenever the original walk is on* $b-1$*, the probability of reaching 0 before reaching b again, is at most* $1/n^{b-1}$
- *Repeat the above walk n times*
  - *Place the particle on* $b-1$ *and play the game*
  - *If it returns to b, place the particle on* $b-1$ *and play the game again*

Counting-Upper-Bound Protocol

Proof

- *Bound* P[*reach b before 0 (when beginning from position 1)*]
- *Probability of winning in the classical ruin problem*
- *If we set* $x = q'/p' = p/q = (n' - b)/b$ *we have that:*

$$\mathrm{P}[reach\ b\ before\ 0] = 1 - \frac{x^b - x}{x^b - 1} = \frac{x-1}{x^b-1} \leq \frac{x}{x^b-1} \approx \frac{1}{x^{b-1}} \approx \frac{1}{n^{b-1}}.$$

- *Thus, whenever the original walk is on* $b-1$*, the probability of reaching 0 before reaching b again, is at most* $1/n^{b-1}$
- *Repeat the above walk n times*
  - *Place the particle on* $b-1$ *and play the game*
  - *If it returns to b, place the particle on* $b-1$ *and play the game again*

Proof

- *Bound* P[*reach b before 0 (when beginning from position 1)*]
- *Probability of winning in the classical ruin problem*
- *If we set* $x = q'/p' = p/q = (n' - b)/b$ *we have that:*

$$\mathrm{P}[\textit{reach } b \textit{ before } 0] = 1 - \frac{x^b - x}{x^b - 1} = \frac{x-1}{x^b - 1} \leq \frac{x}{x^b - 1} \approx \frac{1}{x^{b-1}} \approx \frac{1}{n^{b-1}}.$$

- *Thus, whenever the original walk is on* $b-1$*, the probability of reaching 0 before reaching b again, is at most* $1/n^{b-1}$
- *Repeat the above walk n times*
  - *Place the particle on* $b-1$ *and play the game*
  - *If it returns to b, place the particle on* $b-1$ *and play the game again*

Counting-Upper-Bound Protocol

Proof

- *Bound* P[*reach b before 0 (when beginning from position 1)*]
- *Probability of winning in the classical ruin problem*
- *If we set* $x = q'/p' = p/q = (n' - b)/b$ *we have that:*

$$\mathrm{P}[\textit{reach } b \textit{ before } 0] = 1 - \frac{x^b - x}{x^b - 1} = \frac{x - 1}{x^b - 1} \leq \frac{x}{x^b - 1} \approx \frac{1}{x^{b-1}} \approx \frac{1}{n^{b-1}}.$$

- *Thus, whenever the original walk is on* $b - 1$*, the probability of reaching 0 before reaching b again, is at most* $1/n^{b-1}$
- *Repeat the above walk n times*
    - *Place the particle on* $b - 1$ *and play the game*
    - *If it returns to b, place the particle on* $b - 1$ *and play the game again*

Proof

- *Bound* P[*reach b before 0 (when beginning from position 1)*]
- *Probability of winning in the classical ruin problem*
- *If we set* $x = q'/p' = p/q = (n' - b)/b$ *we have that:*

$$\mathrm{P}[\textit{reach } b \textit{ before } 0] = 1 - \frac{x^b - x}{x^b - 1} = \frac{x-1}{x^b - 1} \leq \frac{x}{x^b - 1} \approx \frac{1}{x^{b-1}} \approx \frac{1}{n^{b-1}}.$$

- *Thus, whenever the original walk is on* $b-1$, *the probability of reaching 0 before reaching* b *again, is at most* $1/n^{b-1}$
- *Repeat the above walk* n *times*
  - *Place the particle on* $b-1$ *and play the game*
  - *If it returns to* b, *place the particle on* $b-1$ *and play the game again*

Counting-Upper-Bound Protocol

- *From Boole-Bonferroni inequality, we have that:*

$$\begin{aligned}\mathrm{P}[\textit{fail at least once}] &\leq \sum_{m=1}^{n} \mathrm{P}[\textit{fail at repetition } m] \\ &\leq \sum_{m=1}^{n} \frac{1}{n^{b-1}} = \frac{n}{n^{b-1}} \\ &= \frac{1}{n^{b-2}}\end{aligned}$$

- *In summary:*
  - *Even if the protocol was restricted to disregard counter differences greater than b*
  - *With probability $\geq 1 - 1/n^c$ (for constant $c = b - 2$) the protocol has not terminated after at least n effective interactions*
  - *Implies that the leader has counted at least half of the nodes*

Counting-Upper-Bound Protocol

- *From Boole-Bonferroni inequality, we have that:*

$$\begin{aligned} \mathrm{P}[\textit{fail at least once}] &\leq \sum_{m=1}^{n} \mathrm{P}[\textit{fail at repetition } m] \\ &\leq \sum_{m=1}^{n} \frac{1}{n^{b-1}} = \frac{n}{n^{b-1}} \\ &= \frac{1}{n^{b-2}} \end{aligned}$$

- *In summary:*
  - *Even if the protocol was restricted to disregard counter differences greater than b*
  - *With probability $\geq 1 - 1/n^c$ (for constant $c = b - 2$) the protocol has not terminated after at least n effective interactions*
  - *Implies that the leader has counted at least half of the nodes* □

Counting-Upper-Bound Protocol

- *From Boole-Bonferroni inequality, we have that:*

$$
\begin{aligned}
P[\mathit{fail\ at\ least\ once}] &\leq \sum_{m=1}^{n} P[\mathit{fail\ at\ repetition\ m}] \\
&\leq \sum_{m=1}^{n} \frac{1}{n^{b-1}} = \frac{n}{n^{b-1}} \\
&= \frac{1}{n^{b-2}}
\end{aligned}
$$

- *In summary:*
  - *Even if the protocol was restricted to disregard counter differences greater than b*
  - *With probability $\geq 1 - 1/n^c$ (for constant $c = b - 2$) the protocol has not terminated after at least n effective interactions*
  - *Implies that the leader has counted at least half of the nodes* □

- *From Boole-Bonferroni inequality, we have that:*

$$\begin{aligned}\mathrm{P}[\textit{fail at least once}] &\leq \sum_{m=1}^{n} \mathrm{P}[\textit{fail at repetition } m] \\ &\leq \sum_{m=1}^{n} \frac{1}{n^{b-1}} = \frac{n}{n^{b-1}} \\ &= \frac{1}{n^{b-2}}\end{aligned}$$

- *In summary:*
  - *Even if the protocol was restricted to disregard counter differences greater than b*
  - *With probability $\geq 1 - 1/n^c$ (for constant $c = b - 2$) the protocol has not terminated after at least n effective interactions*
  - *Implies that the leader has counted at least half of the nodes* □

Counting-Upper-Bound Protocol

- *From Boole-Bonferroni inequality, we have that:*

$$\begin{aligned} P[\textit{fail at least once}] &\leq \sum_{m=1}^{n} P[\textit{fail at repetition } m] \\ &\leq \sum_{m=1}^{n} \frac{1}{n^{b-1}} = \frac{n}{n^{b-1}} \\ &= \frac{1}{n^{b-2}} \end{aligned}$$

- *In summary:*
  - *Even if the protocol was restricted to disregard counter differences greater than b*
  - *With probability $\geq 1 - 1/n^c$ (for constant $c = b - 2$) the protocol has not terminated after at least n effective interactions*
  - *Implies that the leader has counted at least half of the nodes* □

Further Remarks

- Expected running time: $O(n^2 \log n)$ interactions
- Experiments show that in most cases the estimation is closer to $(9/10)n$
- Exact value of n: l waits an additional large polynomial of r_0
- The unique leader seems to be necessary (also some experimental evidence)
  - very interesting open problem
  - w.h.p. all states coexist with $\Theta(n)$ cardinalities [Doty, SODA '14]
  - a node may observe the same as in a fixed population and terminate after meeting a few nodes
- If there is no leader but UIDs we can solve the problem
  - The maximum id can be made to simulate the behavior of a leader

Further Remarks

- Expected running time: $O(n^2 \log n)$ interactions
- Experiments show that in most cases the estimation is closer to $(9/10)n$
- Exact value of n: l waits an additional large polynomial of r_0
- The unique leader seems to be necessary (also some experimental evidence)
  - very interesting open problem
  - w.h.p. all states coexist with $\Theta(n)$ cardinalities [Doty, SODA '14]
  - a node may observe the same as in a fixed population and terminate after meeting a few nodes
- If there is no leader but UIDs we can solve the problem
  - The maximum id can be made to simulate the behavior of a leader

Further Remarks

- Expected running time: $O(n^2 \log n)$ interactions
- Experiments show that in most cases the estimation is closer to $(9/10)n$
- Exact value of n: l waits an additional large polynomial of r_0
- The unique leader seems to be necessary (also some experimental evidence)
  - very interesting open problem
  - w.h.p. all states coexist with $\Theta(n)$ cardinalities [Doty, SODA '14]
  - a node may observe the same as in a fixed population and terminate after meeting a few nodes
- If there is no leader but UIDs we can solve the problem
  - The maximum id can be made to simulate the behavior of a leader

Further Remarks

- Expected running time: $O(n^2 \log n)$ interactions
- Experiments show that in most cases the estimation is closer to $(9/10)n$
- Exact value of n: l waits an additional large polynomial of r_0
- The unique leader seems to be necessary (also some experimental evidence)
  - very interesting open problem
  - w.h.p. all states coexist with $\Theta(n)$ cardinalities [Doty, SODA '14]
  - a node may observe the same as in a fixed population and terminate after meeting a few nodes
- If there is no leader but UIDs we can solve the problem
  - The maximum id can be made to simulate the behavior of a leader

Further Remarks

- Expected running time: $O(n^2 \log n)$ interactions
- Experiments show that in most cases the estimation is closer to $(9/10)n$
- Exact value of n: l waits an additional large polynomial of r_0
- The unique leader seems to be necessary (also some experimental evidence)
  - very interesting open problem
  - w.h.p. all states coexist with $\Theta(n)$ cardinalities [Doty, SODA '14]
  - a node may observe the same as in a fixed population and terminate after meeting a few nodes
- If there is no leader but UIDs we can solve the problem
  - The maximum id can be made to simulate the behavior of a leader

Generic Constructors

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Counting Protocol: constructs w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
    - input: index of pixel and $\sqrt{n}$, in binary
    - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

Generic Constructors

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Counting Protocol: constructs w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
   - input: index of pixel and $\sqrt{n}$, in binary
   - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

Generic Constructors

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Counting Protocol: constructs w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
   - input: index of pixel and $\sqrt{n}$, in binary
   - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Counting Protocol: constructs w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
    - input: index of pixel and $\sqrt{n}$, in binary
    - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

Generic Constructors

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Counting Protocol: constructs w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
   - input: index of pixel and $\sqrt{n}$, in binary
   - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

Generic Constructors

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Counting Protocol: constructs w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
    - input: index of pixel and $\sqrt{n}$, in binary
    - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

Generic Constructors

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Counting Protocol: constructs w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
   - input: index of pixel and $\sqrt{n}$, in binary
   - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

Generic Constructors

- Characterization for the class of constructible 2D shape languages
- Simulate shape-constructing TMs to realize their output-shape in the distributed system

1. Counting Protocol: constructs w.h.p. a line of length $\Theta(\log n)$, containing n in binary
2. The leader exploits knowledge of n to construct a $\sqrt{n} \times \sqrt{n}$ square
3. Simulate the TM on the square n distinct times, one for each pixel
    - input: index of pixel and $\sqrt{n}$, in binary
    - output: on or off (decision for the corresponding pixel)
4. Release the connected shape consisting of the on pixels

Characterization

Theorem

Let $\mathcal{L} = (S_1, S_2, \ldots)$ be a connected 2D shape language, such that $\mathcal{L}$ is TM-computable in space d^2. Then there is a protocol that w.h.p. constructs $\mathcal{L}$. In particular, for all $d \geq 1$, whenever the protocol is executed on a population of size $n = d^2$, w.h.p. it constructs S_d and terminates. In the worst case, when G_d (that is, the shape of S_d) is a line of length d, the waste is $(d-1)d = O(d^2) = O(n)$.

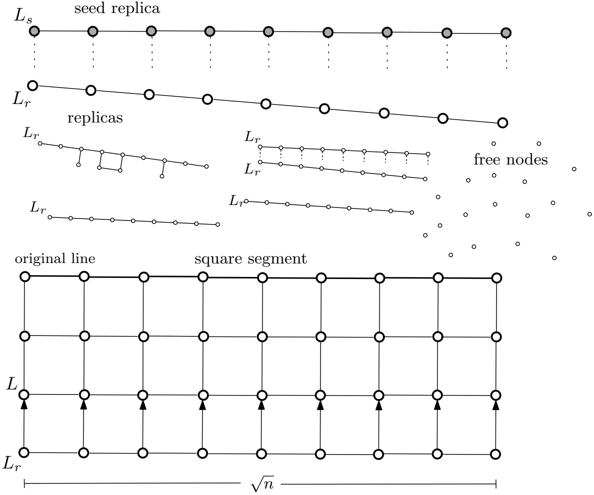
Counting-on-a-line Protocol

- Adapt Counting-Upper-Bound to work in the present model
- The same probabilistic process
- The leader constructs a line that stores the two counters in binary
  - The line grows whenever more space is required

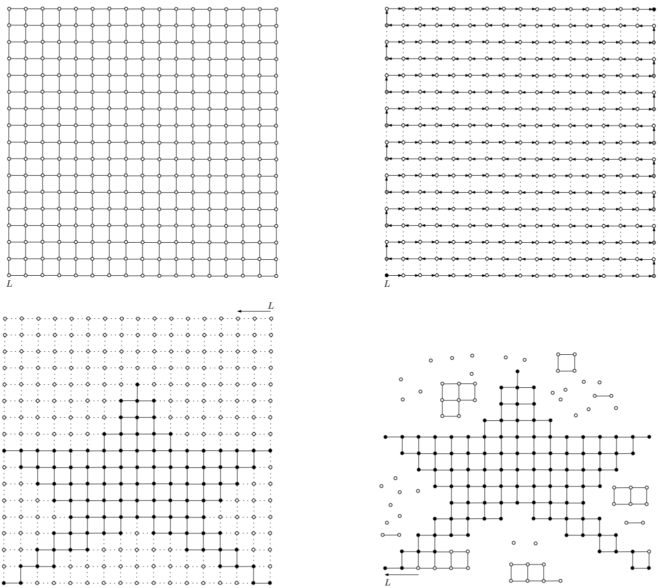
Lemma

Counting-on-a-Line protocol terminates in every execution. Moreover, when the leader terminates, w.h.p. it has formed an active line of length $\log n$ containing n in binary in the r_0 components of the nodes of the line (each node storing one bit).

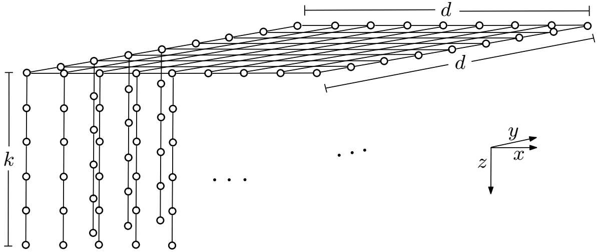
Constructing a $\sqrt{n} \times \sqrt{n}$ Square



Simulating and Releasing



Parallelizing the Simulations



Theorem

Let $\mathcal{L} = (S_1, S_2, \ldots)$ be a TM-computable connected 2D shape language, such that S_d is computable in space $k = f(d)$ and k is computable in space $O(k \cdot d^2)$. Then there is a protocol that w.h.p. constructs $\mathcal{L}$. In particular, for all $d \geq 1$, whenever the protocol is executed on a population of size $n = k \cdot d^2$, w.h.p. it constructs S_d and terminates, by executing d^2 simulations in parallel each with space $O(k)$.

Open Problems

- Complete characterization of constructed networks
- Give a faster than $O(n^3)$ protocol for global line (e.g. $O(n^2 \log n)$)
- Count w.h.p. and terminate if all nodes are initially identical?
- Models of active mobility/actuation (or hybrid active-passive)
- Take other physical considerations into account
  - mass, strength of bonds, rigid and elastic structure, collisions
- Structures that optimize some global property or that achieve a behavior/functionality
- Protocols that efficiently reconstruct broken parts of the structure
- Draw connections to natural processes and to self-assembly and programmable matter models
- We need more real systems-collectives of large numbers of simple interacting devices

Thank You!