Distributed Computing on the (Fruit) Fly

Yuval Emek

Distributed Computing Group
ETH Zurich

October 2012

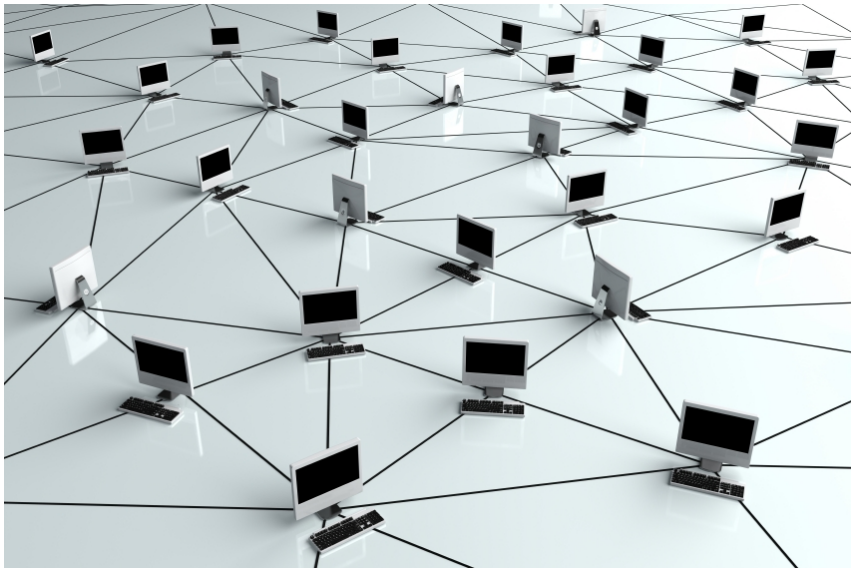
Advances in Distributed Graph Algorithms (ADGA)
Salvador Bahia, Brazil

Synopsis

Distributed computing = power & limitations of computation in networks

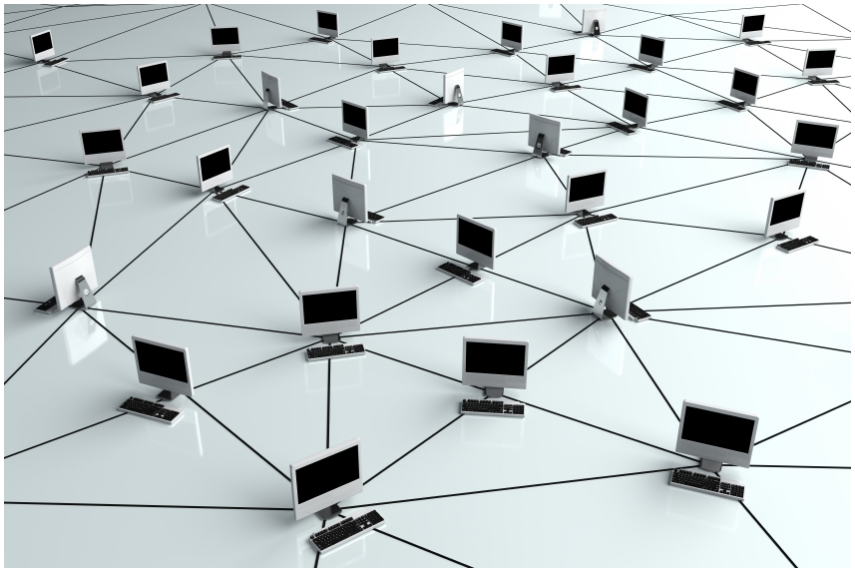
Synopsis

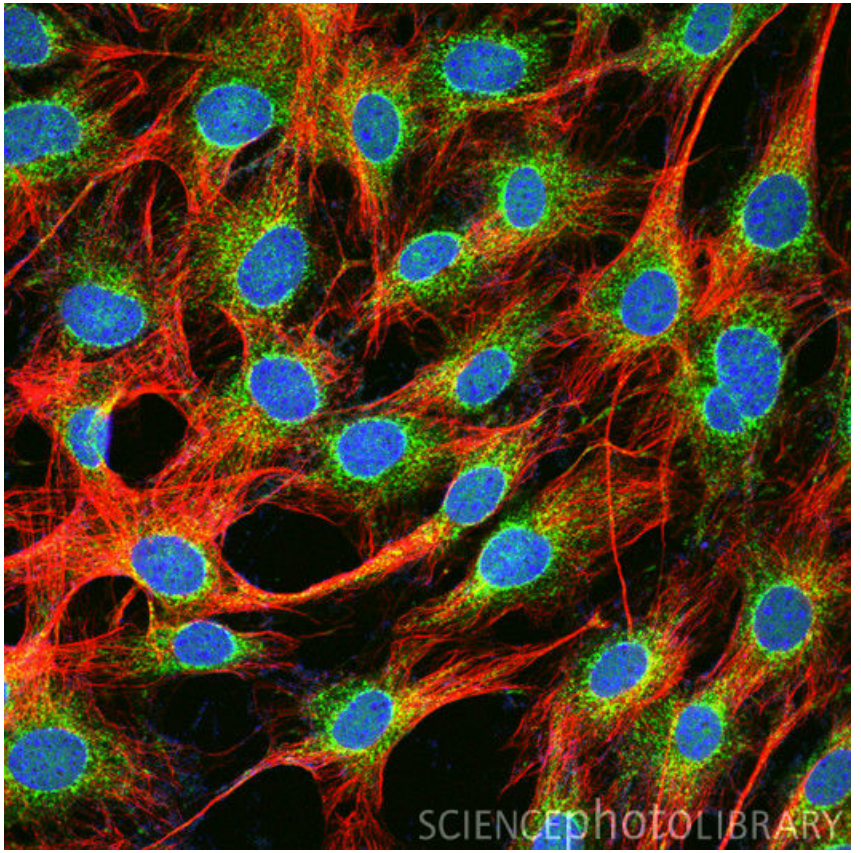
Distributed computing = power & limitations of computation in networks



Synopsis

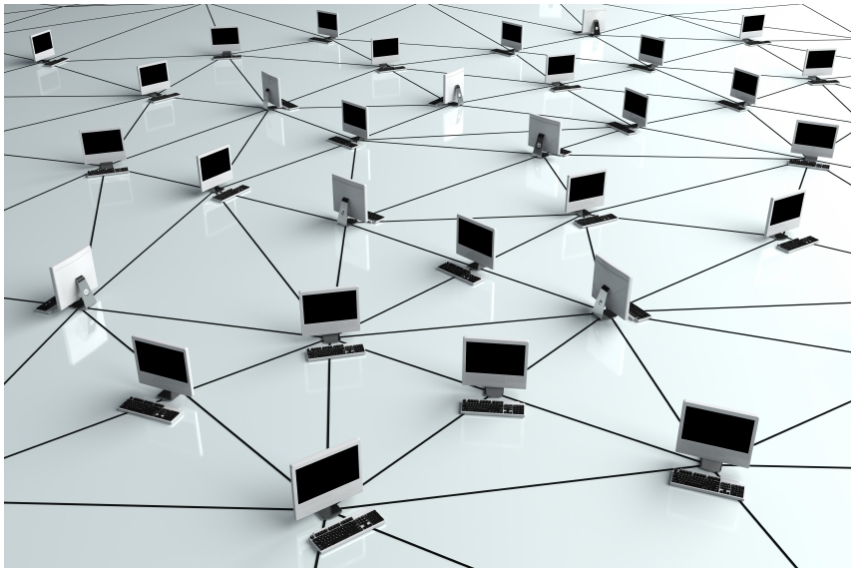
Distributed computing = power & limitations of computation in networks

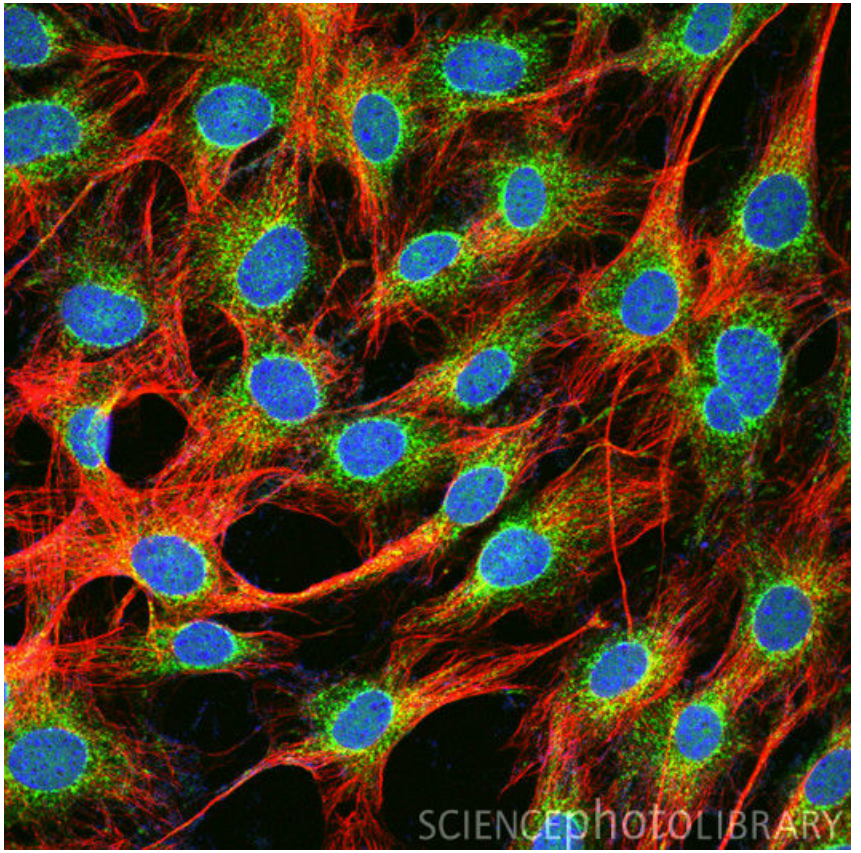




Synopsis

Distributed computing = power & limitations of computation in networks

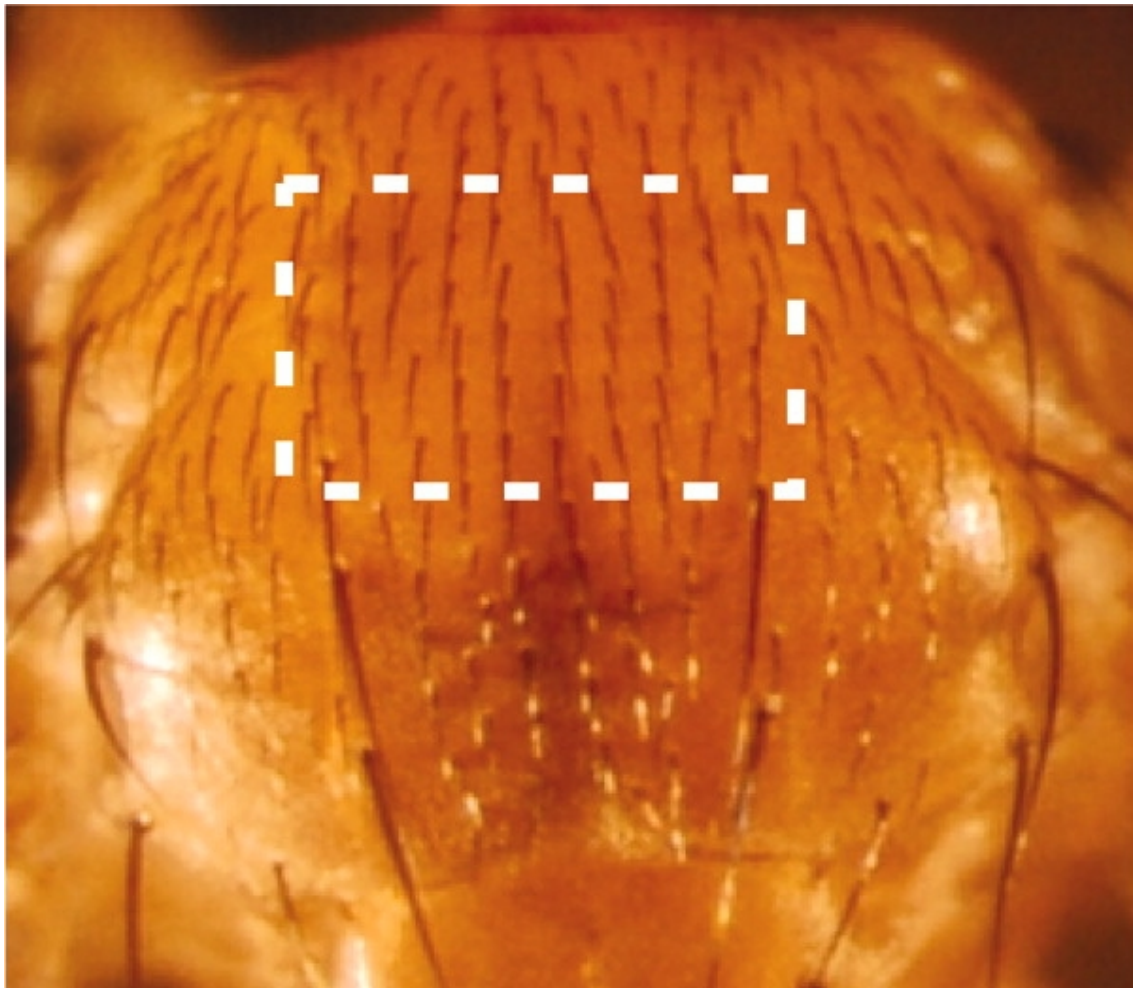




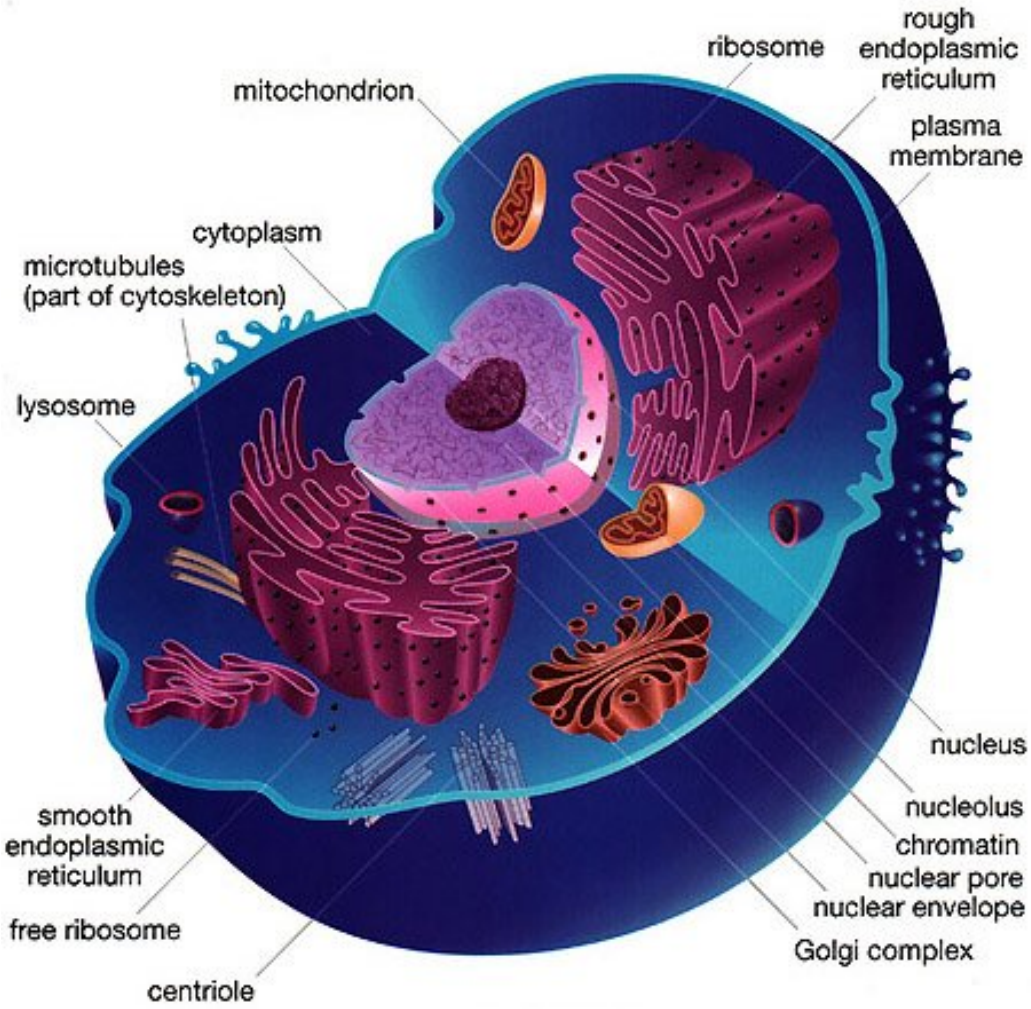
Our mission: distributed computing in biological cellular networks

Motivation

Selection of sensory organ precurser (SOP) cells = solving MIS
[Afek, Alon, Barad, Hornstein, Barkai, Bar-Joseph 11]



The structure of cells (eukaryotes)



The nucleus

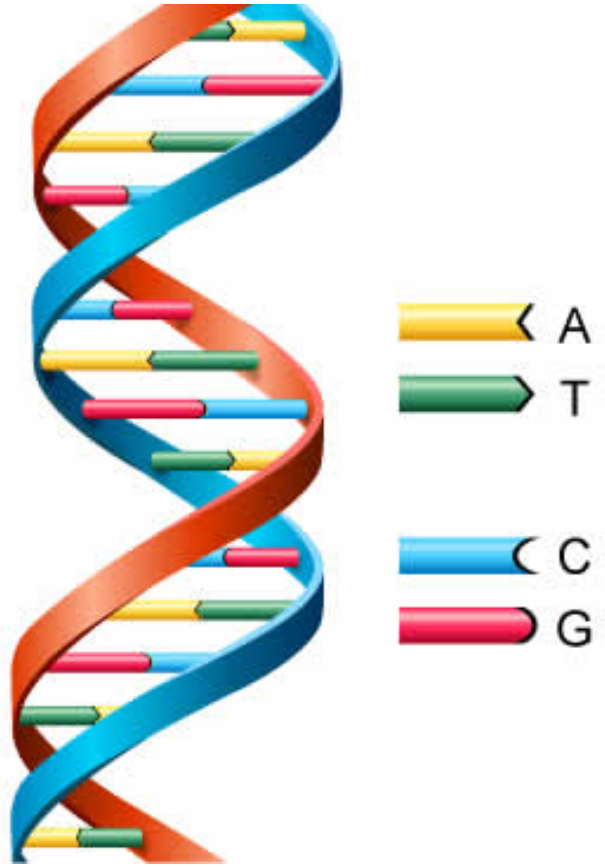
The nucleus

Analogous to central processing unit

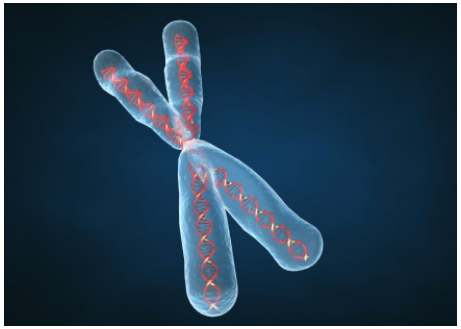
The nucleus

Analogous to central processing unit

- Code = DNA



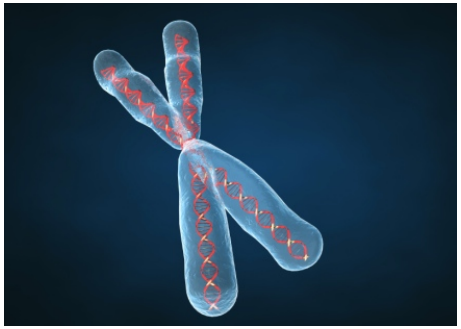
The nucleus



Analogous to central processing unit

- Code = DNA
  - Organized in chromosomes

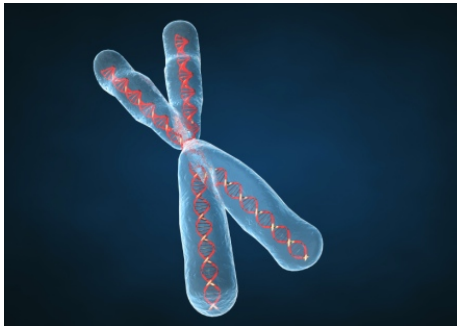
The nucleus



Analogous to central processing unit

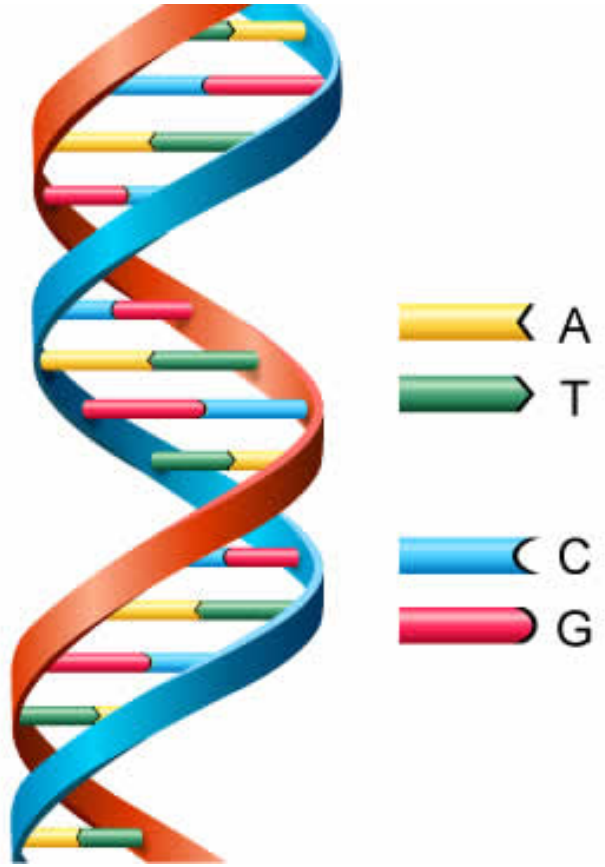
- Code = DNA
  - Organized in chromosomes
  - Strings of nucleotides

The nucleus

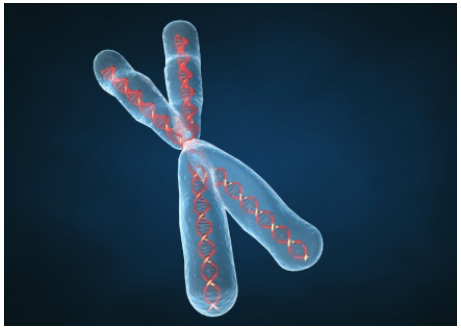


Analogous to central processing unit

- Code = DNA
  - Organized in chromosomes
  - Strings of nucleotides
- Instructions = genes (DNA substrings)



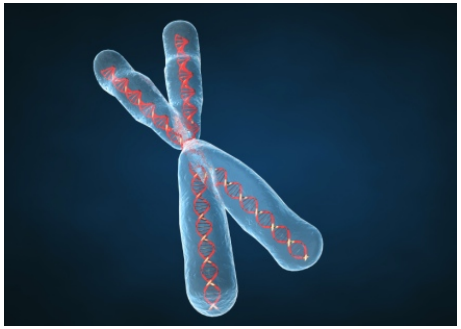
The nucleus



Analogous to central processing unit

- Code = DNA
  - Organized in chromosomes
  - Strings of nucleotides
- Instructions = genes (DNA substrings)
- Execution = gene expression
  - Producing RNA molecules

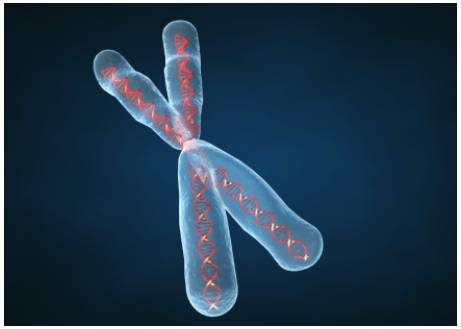
The nucleus

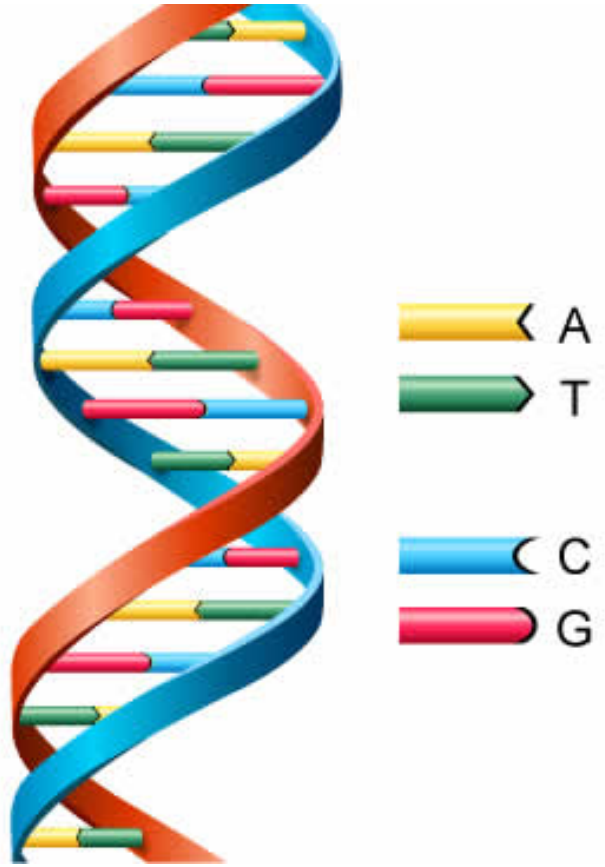


Analogous to central processing unit

- Code = DNA
  - Organized in chromosomes
  - Strings of nucleotides
- Instructions = genes (DNA substrings)
- Execution = gene expression
  - Producing RNA molecules
- Main question:
  which genes are currently expressed?

The nucleus

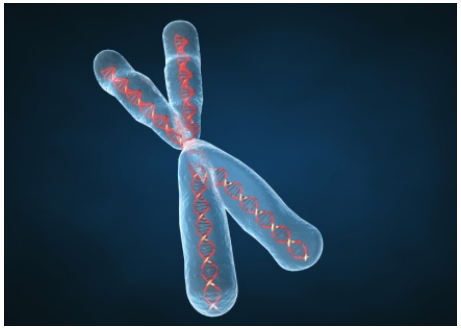


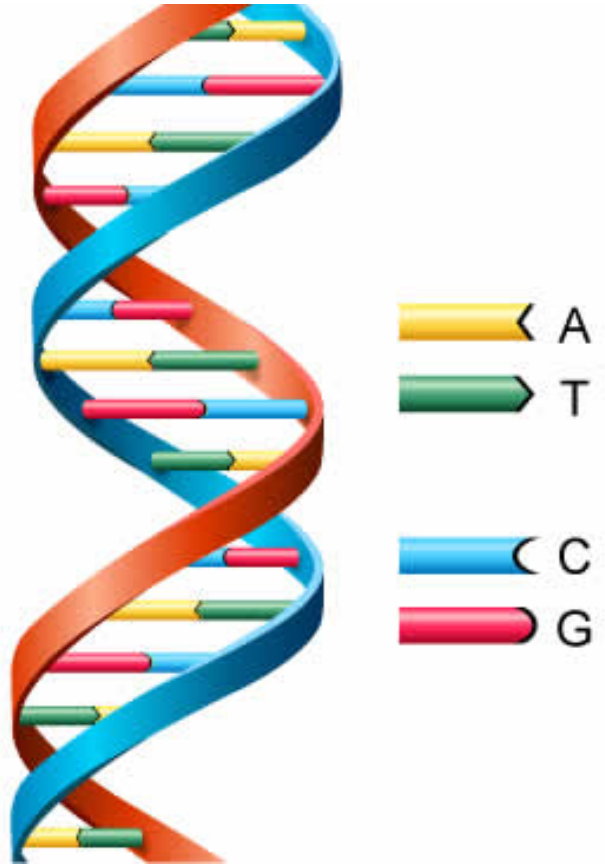


Analogous to central processing unit

- Code = DNA
  - Organized in chromosomes
  - Strings of nucleotides
- Instructions = genes (DNA substrings)
- Execution = gene expression
  - Producing RNA molecules
- Main question:
  which genes are currently expressed?
  - Analogous to CPU's current state

The nucleus





Analogous to central processing unit

- Code = DNA
  - Organized in chromosomes
  - Strings of nucleotides
- Instructions = genes (DNA substrings)
- Execution = gene expression
  - Producing RNA molecules
- Main question:
  which genes are currently expressed?
  - Analogous to CPU's current state
  - Controlled by concentration levels

Cell-to-cell communication

Cell-to-cell communication

- All cells communicate
- Communication exists on all levels (cell types, organisms, species)

Cell-to-cell communication

- All cells communicate
- Communication exists on all levels (cell types, organisms, species)
- Bonnie Bassler on How bacteria "talk"

Cell-to-cell communication

- All cells communicate
- Communication exists on all levels (cell types, organisms, species)
- Bonnie Bassler on How bacteria "talk"

Classify according to communication range:

Cell-to-cell communication

- All cells communicate
- Communication exists on all levels (cell types, organisms, species)
- Bonnie Bassler on How bacteria "talk"

Classify according to communication range:

- Juxtacrine — direct contact
  - respects network's topology

Cell-to-cell communication

- All cells communicate
- Communication exists on all levels (cell types, organisms, species)
- Bonnie Bassler on How bacteria "talk"

Classify according to communication range:

- Juxtacrine — direct contact
  - respects network's topology
- Paracrine, endocrine
  - out of our scope

Juxtacrine communication

Delivery of message m from cell s to cell t

Juxtacrine communication

Delivery of message m from cell s to cell t

1. s produces molecule m

Juxtacrine communication

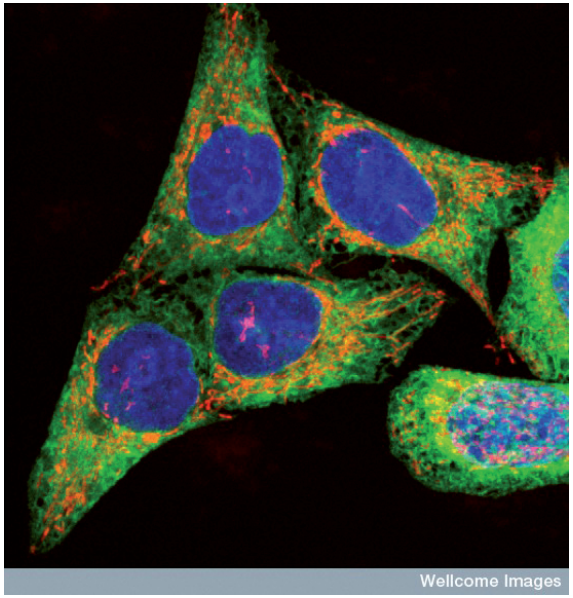
Delivery of message m from cell s to cell t

1. s produces molecule m
2. m crosses from s to t

Juxtacrine communication

Delivery of message m from cell s to cell t

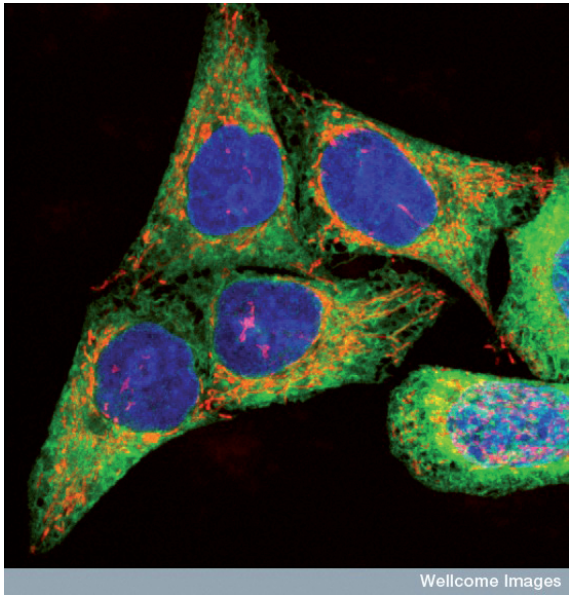
1. s produces molecule m
2. m crosses from s to t
   - gap junction connecting two cytoplasms

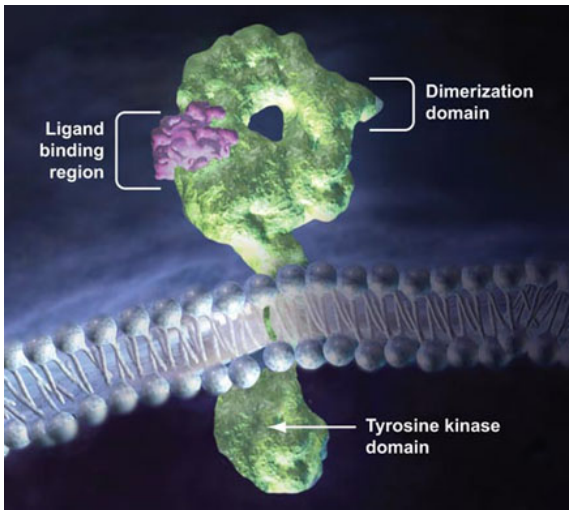


Juxtacrine communication

Delivery of message m from cell s to cell t

1. s produces molecule m
2. m crosses from s to t
   - gap junction connecting two cytoplasms
   - binds to crossmembrane receptor

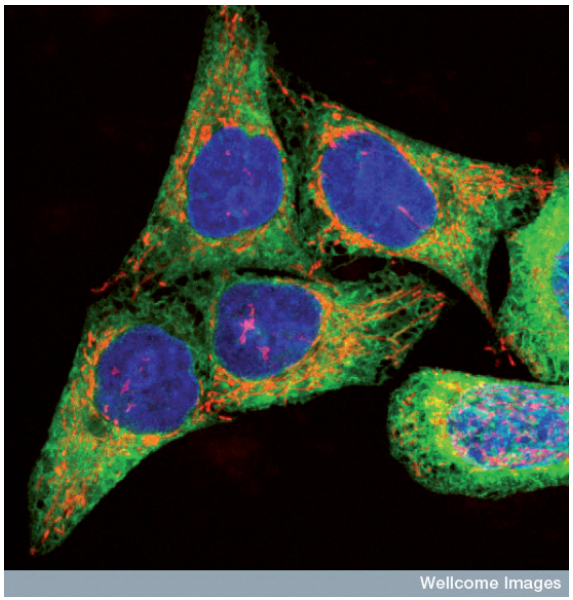


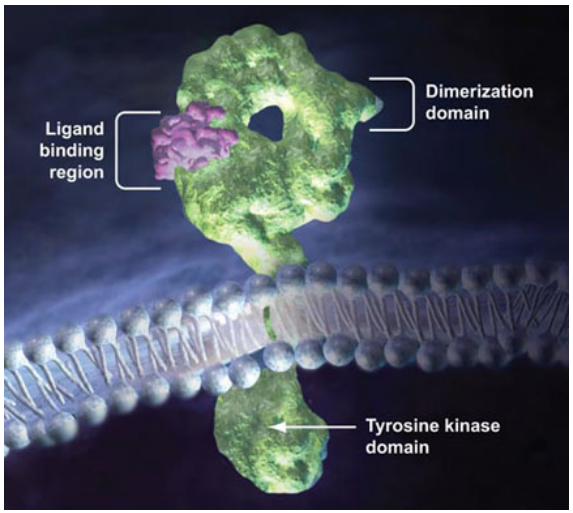


Juxtacrine communication

Delivery of message m from cell s to cell t

1. s produces molecule m
2. m crosses from s to t
   - gap junction connecting two cytoplasms
   - binds to crossmembrane receptor
3. Triggers a signaling cascade inside t

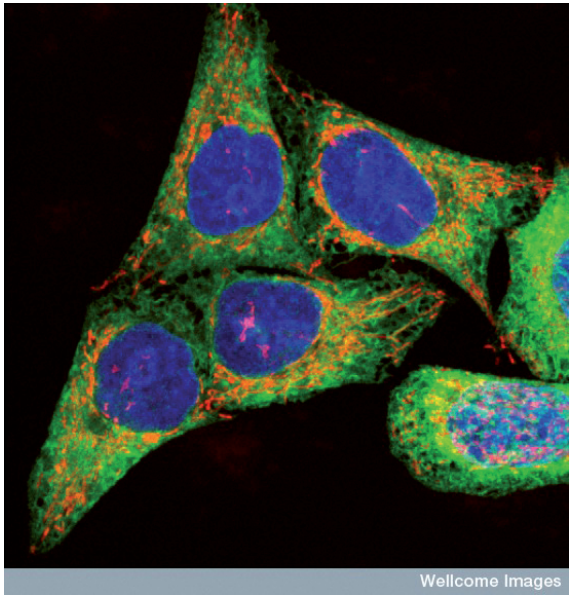


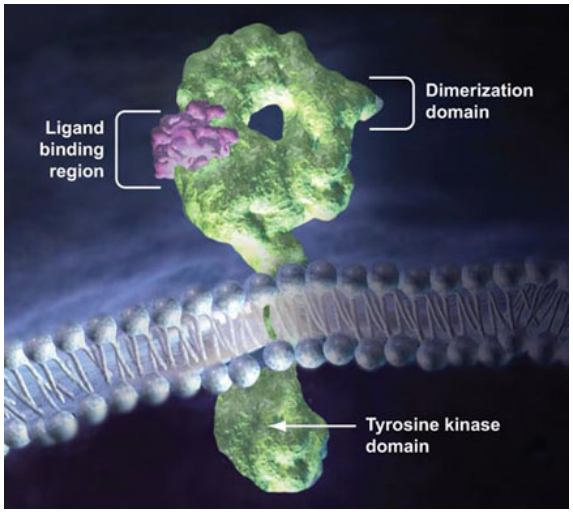


Juxtacrine communication

Delivery of message m from cell s to cell t

1. s produces molecule m
2. m crosses from s to t
   - gap junction connecting two cytoplasms
   - binds to crossmembrane receptor
3. Triggers a signaling cascade inside t
4. Modifies concentration levels in nucleus

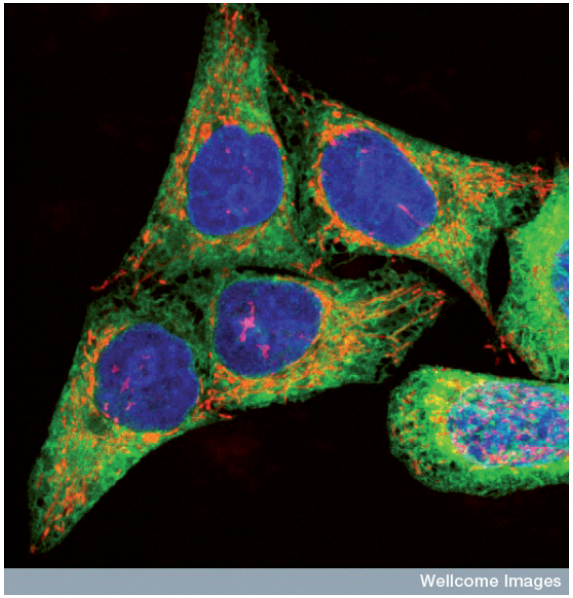


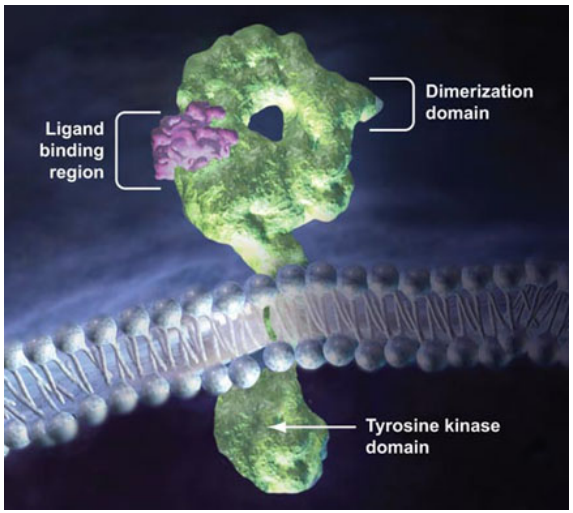


Juxtacrine communication

Delivery of message m from cell s to cell t

1. s produces molecule m
2. m crosses from s to t
   - gap junction connecting two cytoplasms
   - binds to crossmembrane receptor
3. Triggers a signaling cascade inside t
4. Modifies concentration levels in nucleus
5. Affects t's gene expression

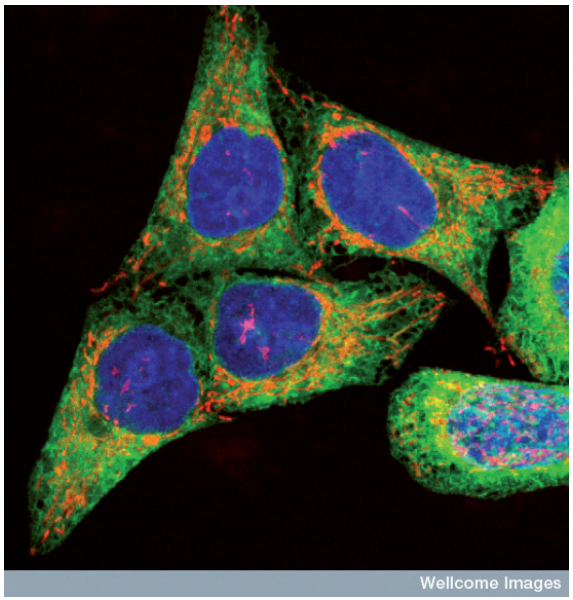




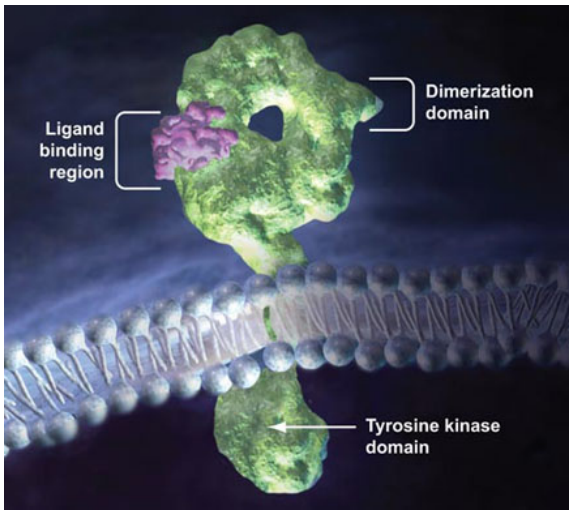
Juxtacrine communication

Delivery of message m from cell s to cell t

1. s produces molecule m
2. m crosses from s to t
   - gap junction connecting two cytoplasms
   - binds to crossmembrane receptor
3. Triggers a signaling cascade inside t
4. Modifies concentration levels in nucleus
5. Affects t's gene expression



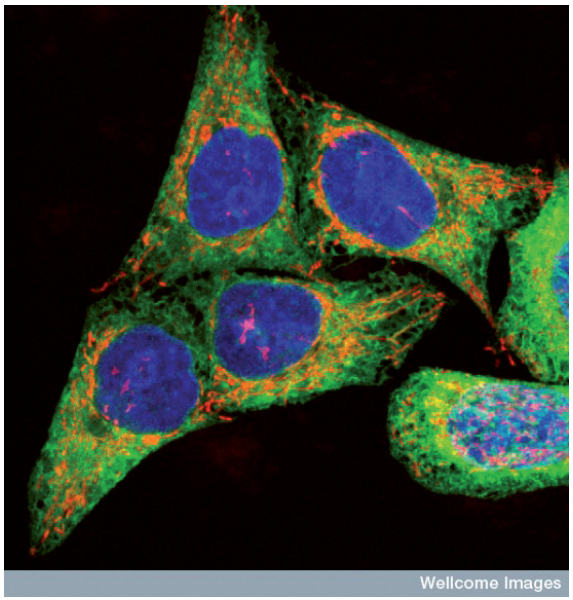
- Gap junction/receptor = port

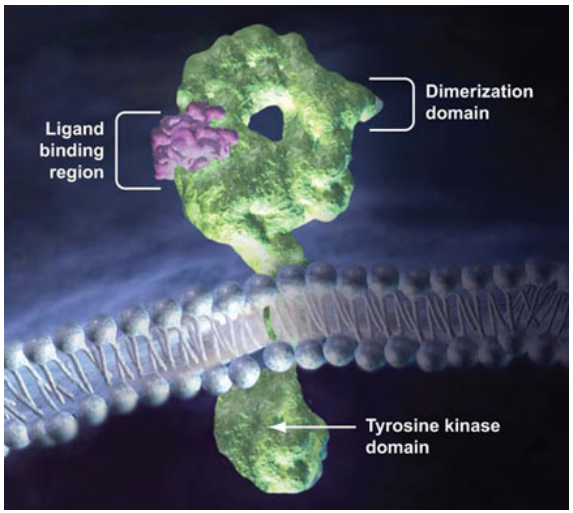


Juxtacrine communication

Delivery of message m from cell s to cell t

1. s produces molecule m
2. m crosses from s to t
   - gap junction connecting two cytoplasms
   - binds to crossmembrane receptor
3. Triggers a signaling cascade inside t
4. Modifies concentration levels in nucleus
5. Affects t's gene expression



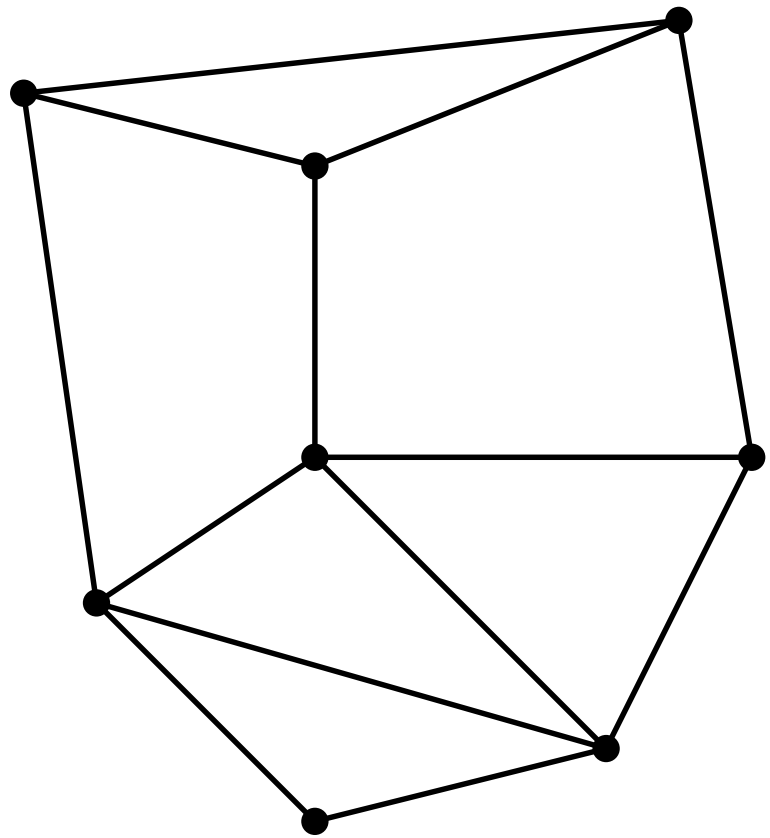


- Gap junction/receptor = port
- No sense of direction
  - all neighbors look the same

Message passing

Message passing

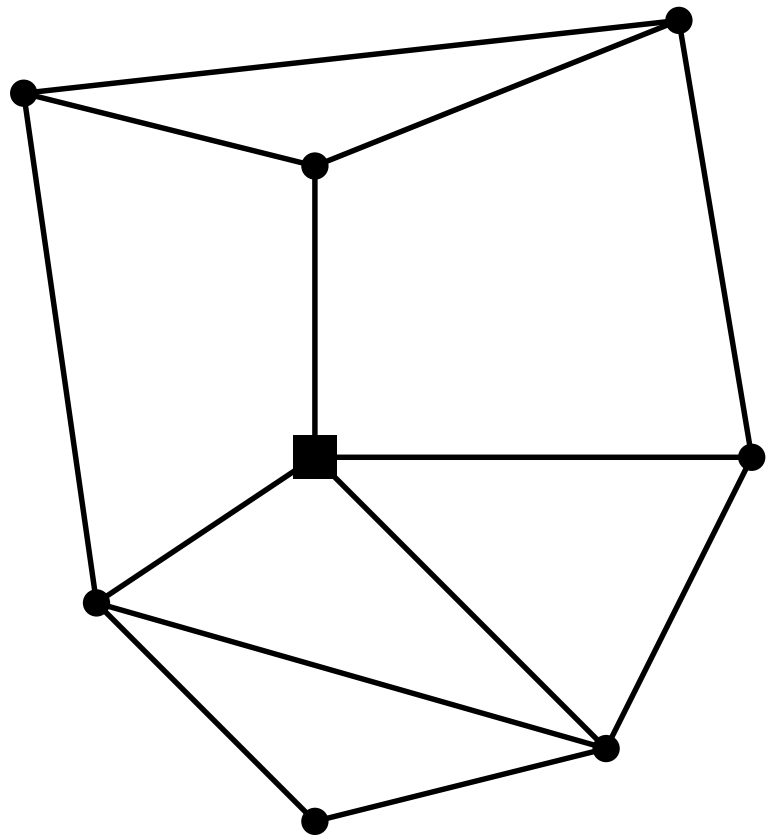
Nodes act locally (don't know global topology)



Message passing

Nodes act locally (don't know global topology)

In each step, node v:

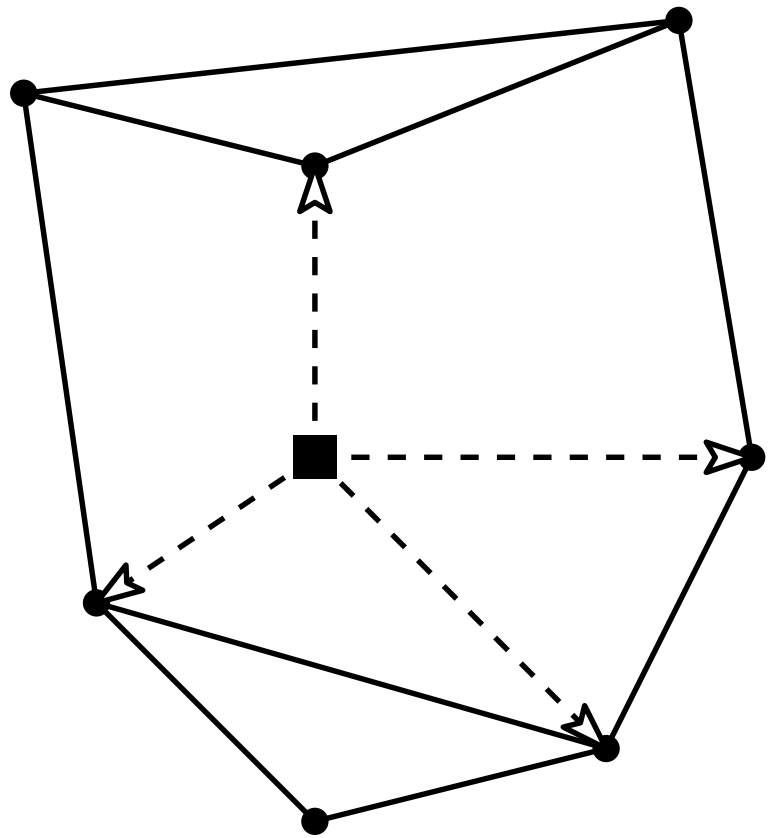


Message passing

Nodes act locally (don't know global topology)

In each step, node v:

- sends messages to $N(v)$

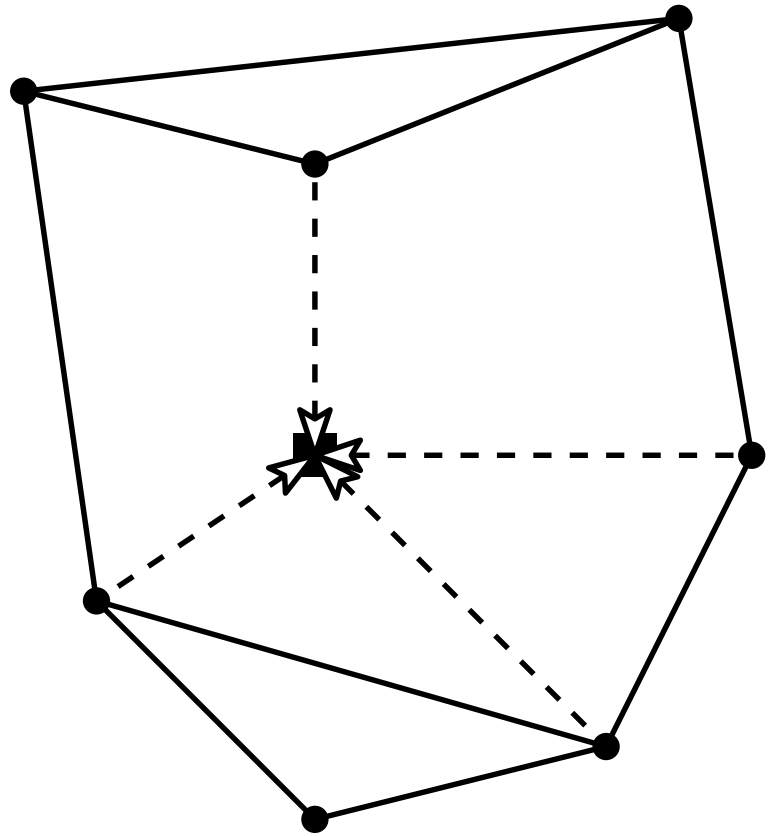


Message passing

Nodes act locally (don't know global topology)

In each step, node v:

- sends messages to $N(v)$
- receives messages from $N(v)$

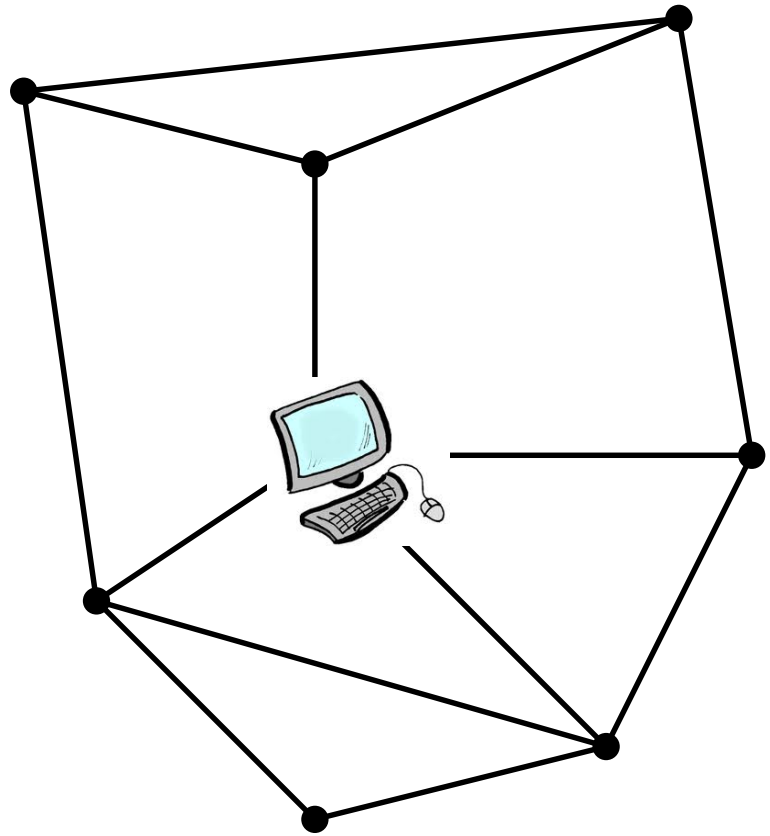


Message passing

Nodes act locally (don't know global topology)

In each step, node v:

- sends messages to $N(v)$
- receives messages from $N(v)$
- performs local computation

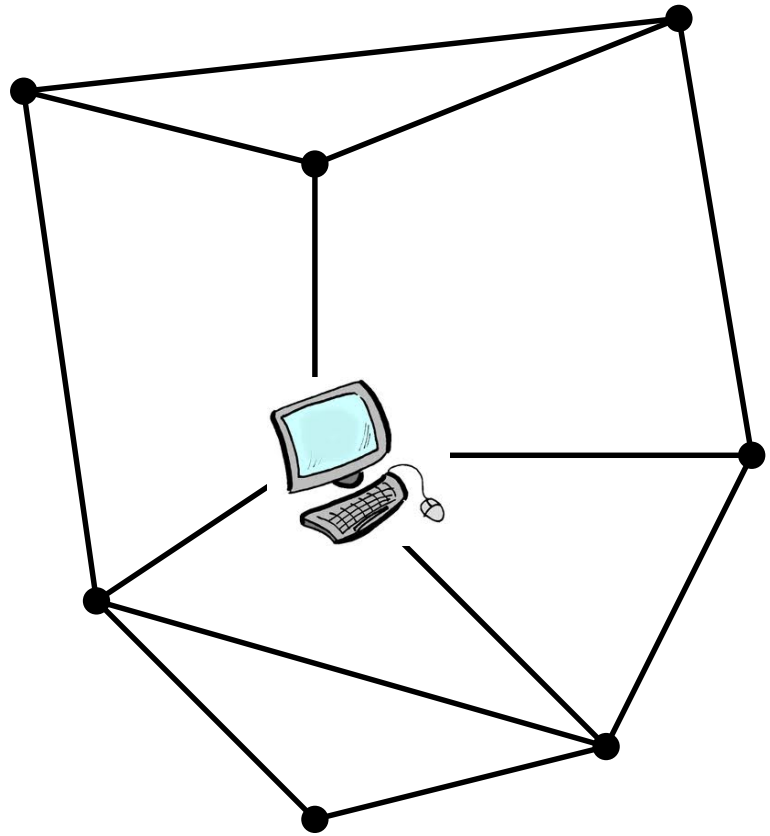


Message passing

Nodes act locally (don't know global topology)

In each step, node v:

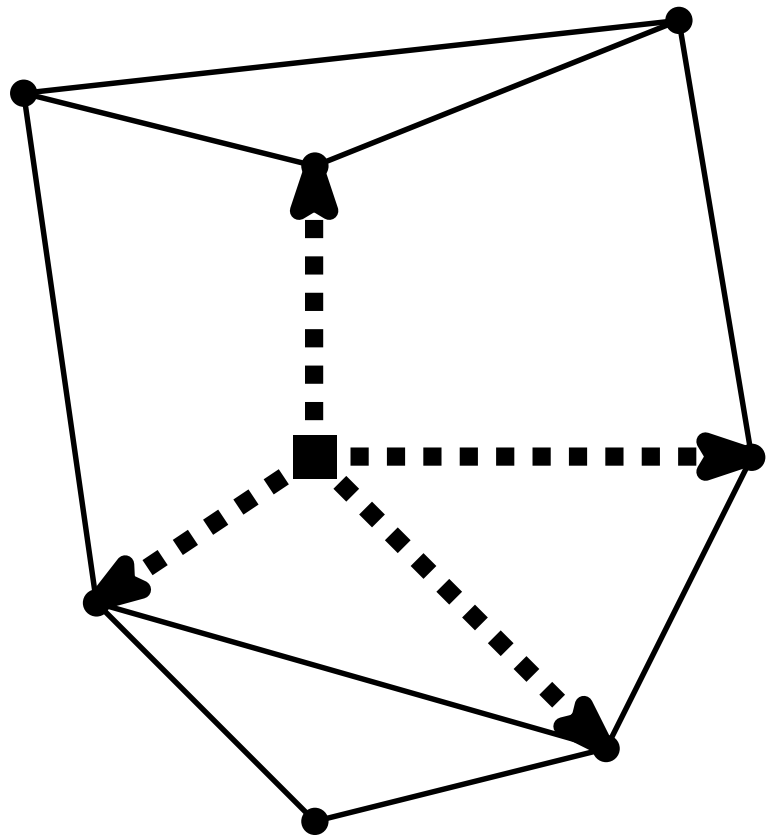
- sends messages to $N(v)$
- receives messages from $N(v)$
- performs local computation



Communication too strong

Message passing

Nodes act locally (don't know global topology)



In each step, node v:

- sends messages to $N(v)$
- receives messages from $N(v)$
- performs local computation

Communication too strong

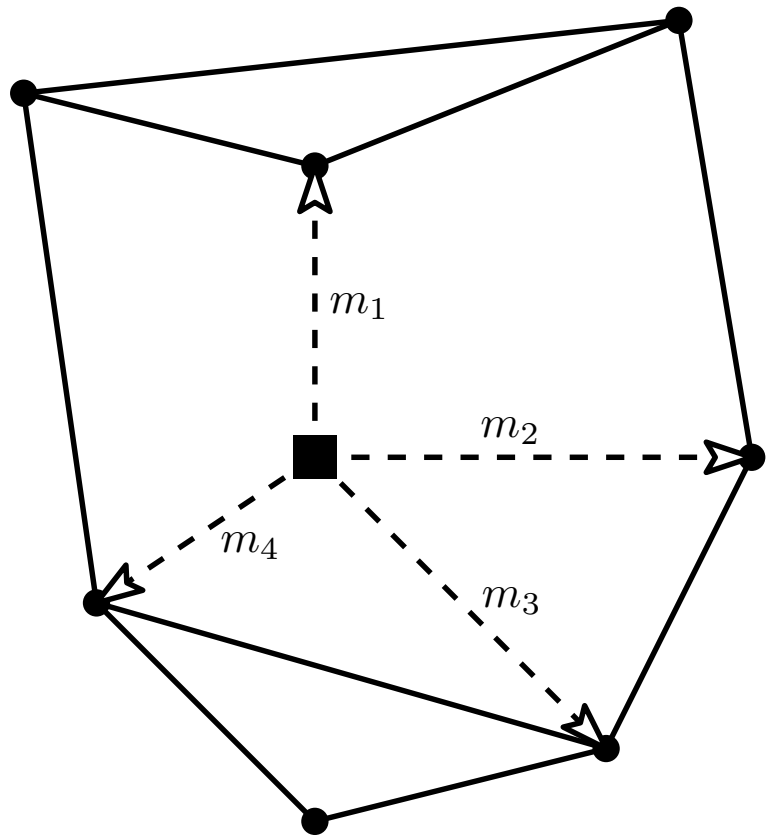
- large messages (size grows with n)

Message passing

Nodes act locally (don't know global topology)

In each step, node v:

- sends messages to $N(v)$
- receives messages from $N(v)$
- performs local computation



Communication too strong

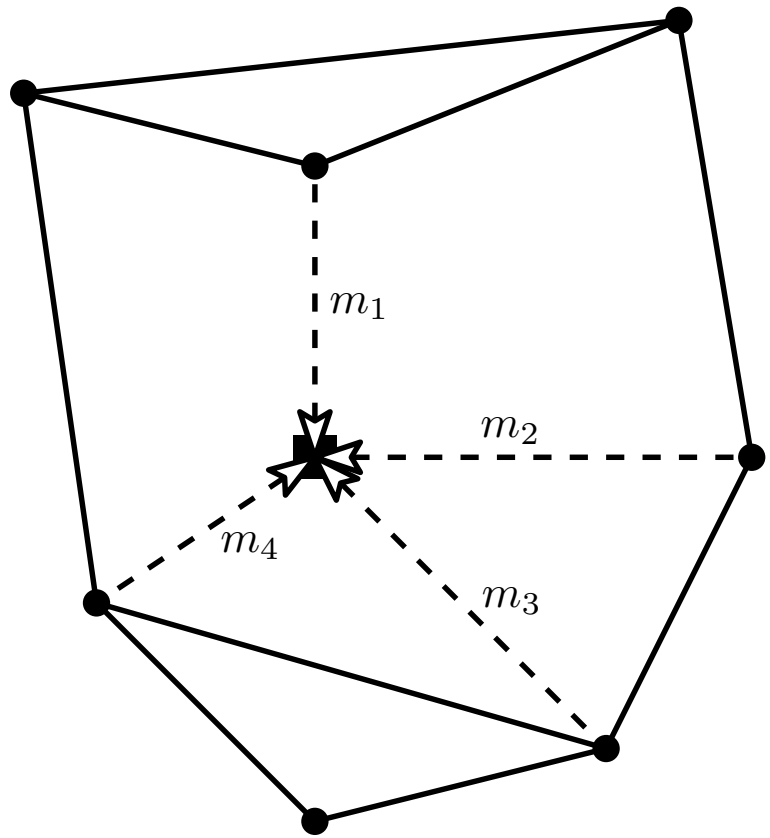
- large messages (size grows with n)
- independent messages to/from each neighbor

Message passing

Nodes act locally (don't know global topology)

In each step, node v:

- sends messages to $N(v)$
- receives messages from $N(v)$
- performs local computation

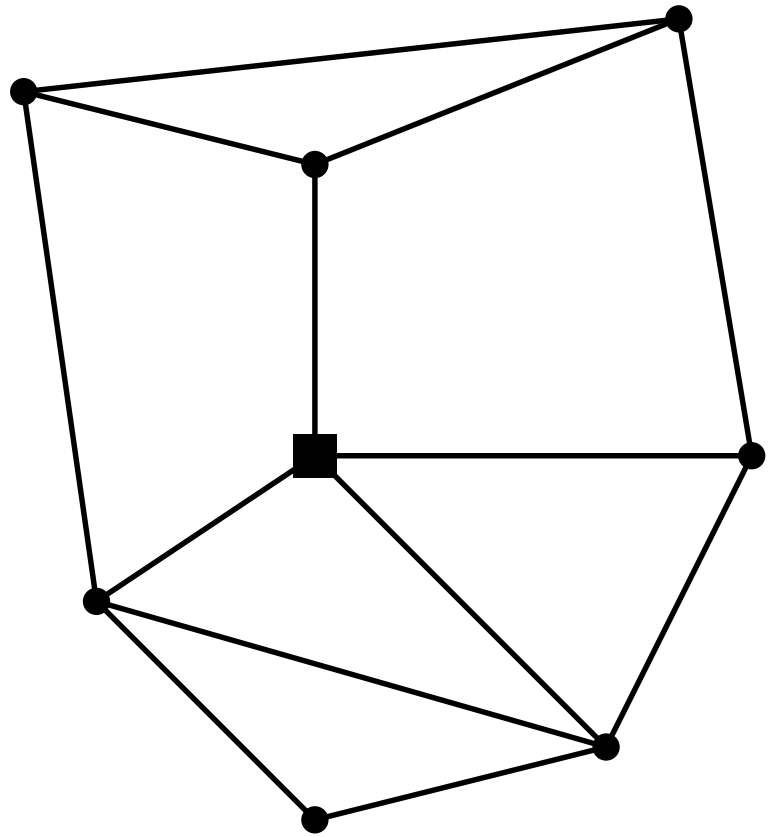


Communication too strong

- large messages (size grows with n)
- independent messages to/from each neighbor

The beeping model

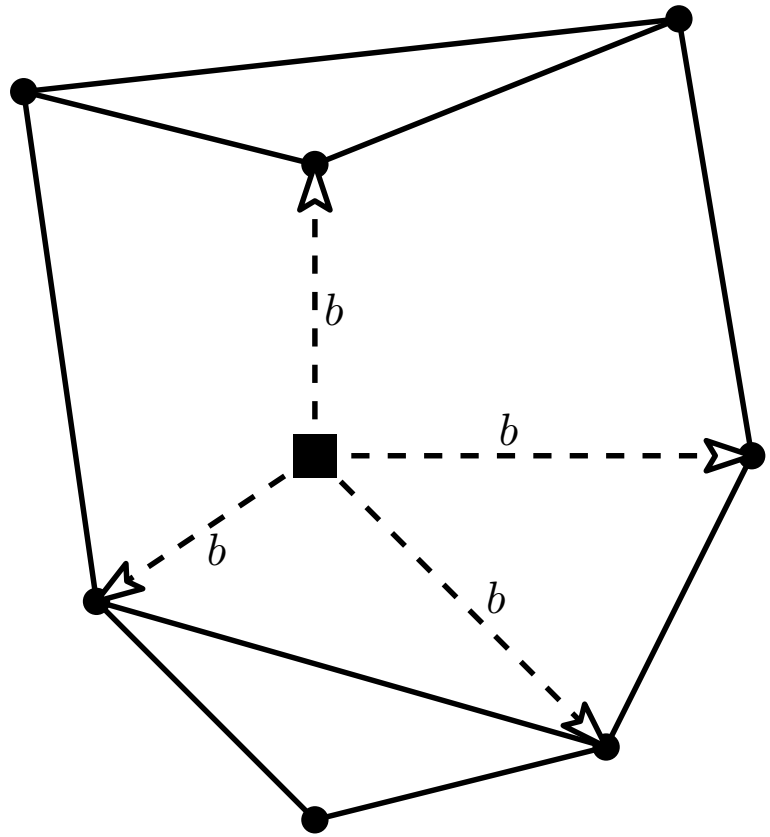
Introduced in [Cornejo, Kuhn 10]



The beeping model

Introduced in [Cornejo, Kuhn 10]

Messages = beeps (no information)

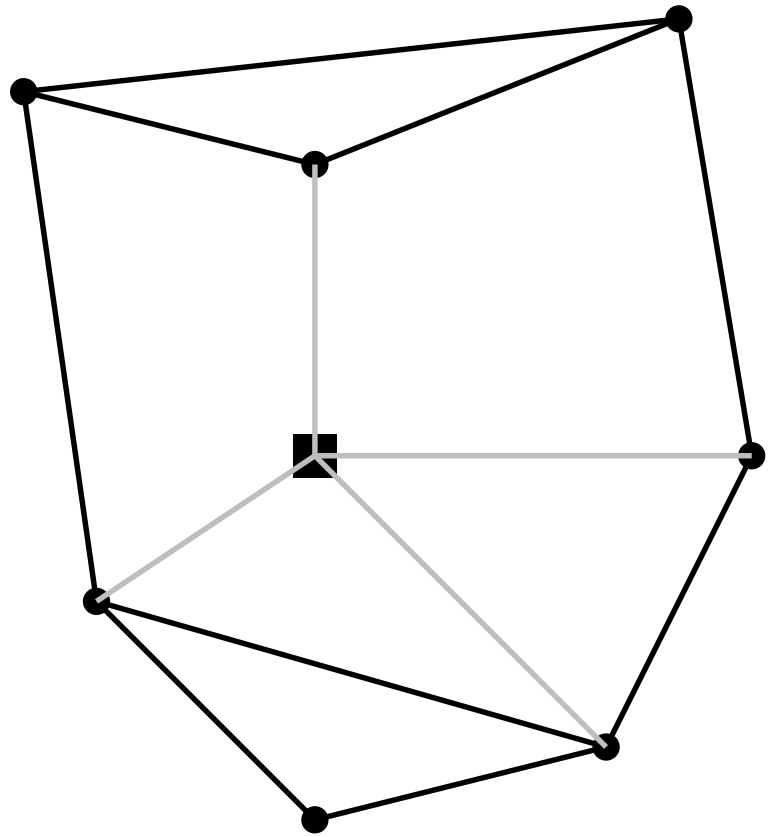


The beeping model

Introduced in [Cornejo, Kuhn 10]

Messages = beeps (no information)

Node distinguishes 0 and ≥ 1 beeps

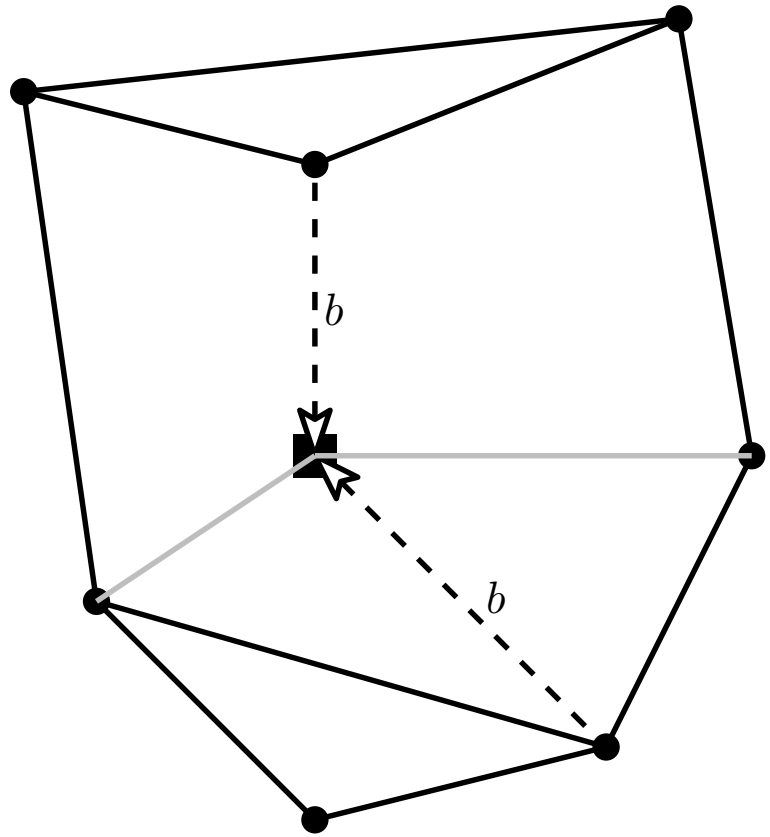


The beeping model

Introduced in [Cornejo, Kuhn 10]

Messages = beeps (no information)

Node distinguishes 0 and ≥ 1 beeps

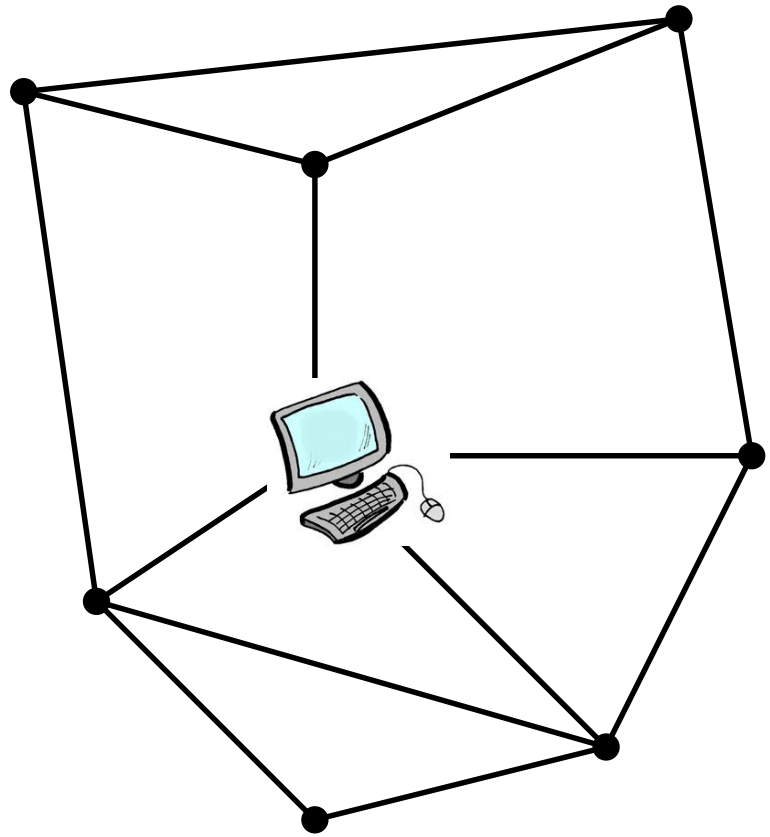


The beeping model

Introduced in [Cornejo, Kuhn 10]

Messages = beeps (no information)

Node distinguishes 0 and ≥ 1 beeps



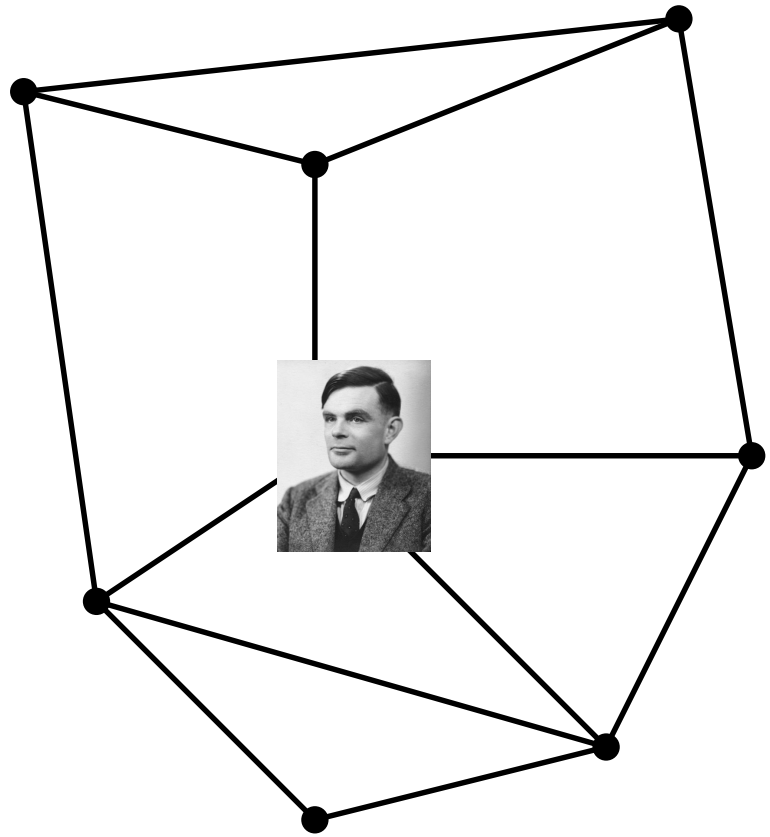
Local computation too strong

The beeping model

Introduced in [Cornejo, Kuhn 10]

Messages = beeps (no information)

Node distinguishes 0 and ≥ 1 beeps



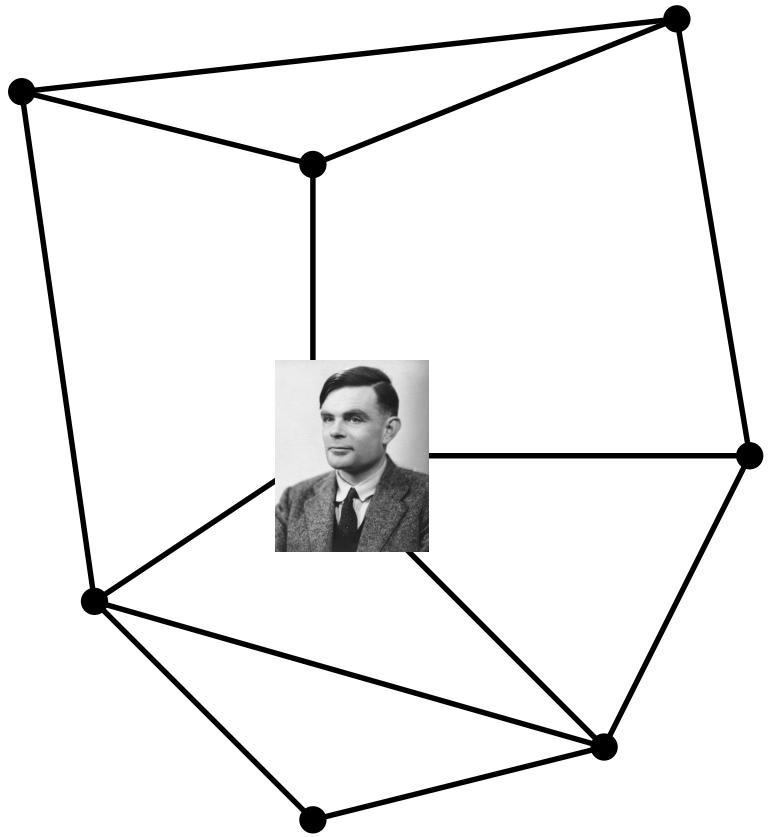
Local computation too strong

The beeping model

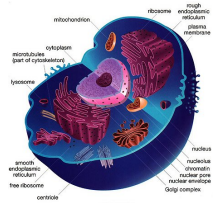
Introduced in [Cornejo, Kuhn 10]

Messages = beeps (no information)

Node distinguishes 0 and ≥ 1 beeps



Local computation too strong



$\ll$

Finite state machines (a.k.a. automata)

Finite state machines (a.k.a. automata)

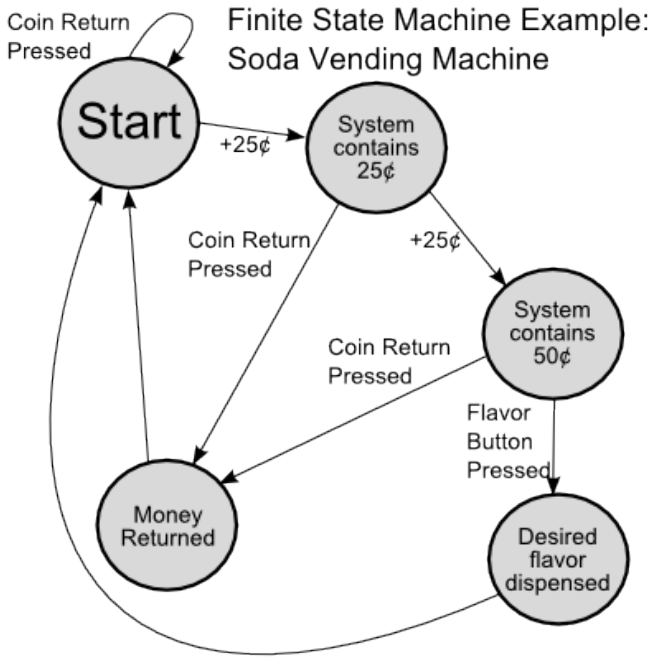
- Finite (fixed) collection of states

Finite state machines (a.k.a. automata)

- Finite (fixed) collection of states
- $q(t+1) \longleftarrow q(t), \text{signals}(t)$

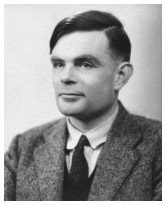
Finite state machines (a.k.a. automata)

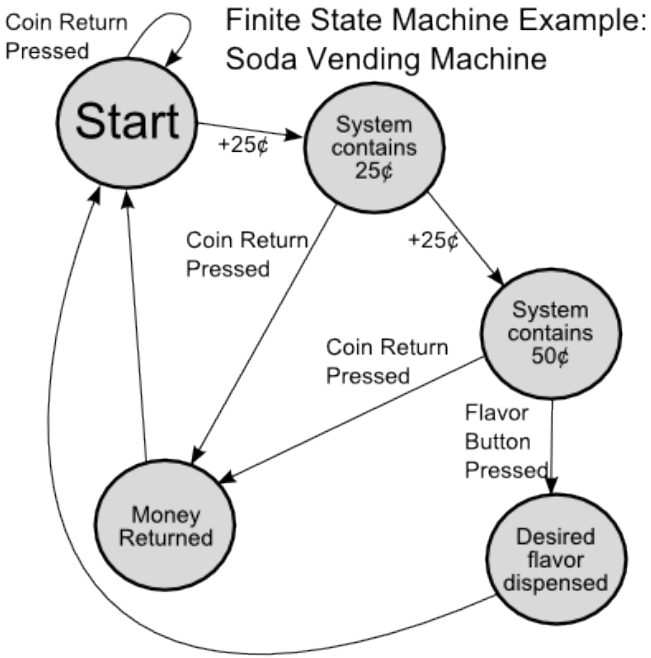
- Finite (fixed) collection of states
- $q(t+1) \longleftarrow q(t), \text{signals}(t)$



Finite state machines (a.k.a. automata)

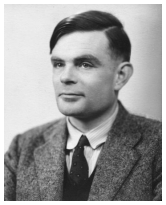
- Finite (fixed) collection of states
- $q(t+1) \longleftarrow q(t), \text{signals}(t)$
- Computational power $\ll$

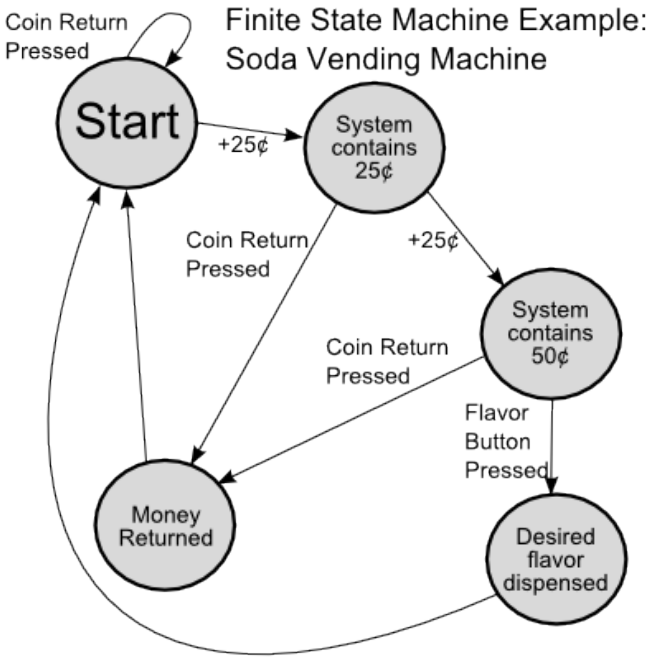




Finite state machines (a.k.a. automata)

- Finite (fixed) collection of states
- $q(t+1) \longleftarrow q(t), \text{signals}(t)$
- Computational power $\ll$

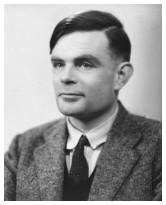


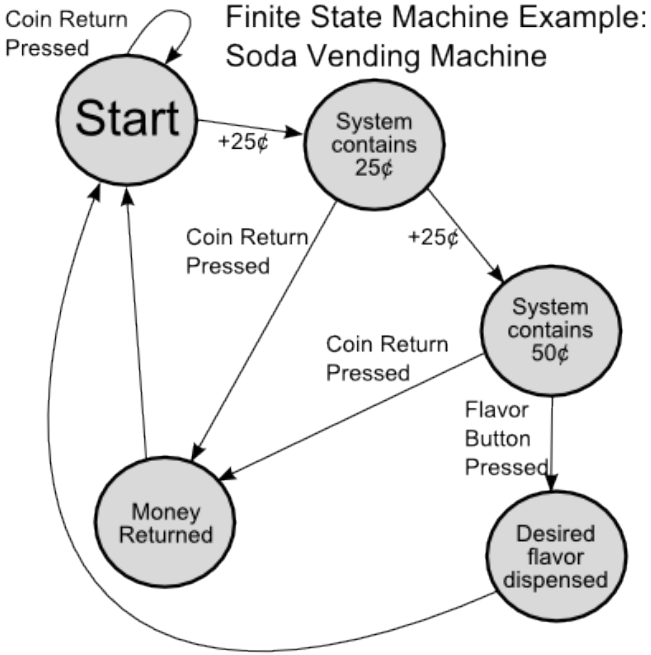


Cell enzymes "programmed" to implement an FSM
[Benenson, Paz-Elizur, Adar, Keinan, Livneh, Shapiro 01]

Finite state machines (a.k.a. automata)

- Finite (fixed) collection of states
- $q(t+1) \longleftarrow q(t), \text{signals}(t)$
- Computational power $\ll$





Cell enzymes "programmed" to implement an FSM
[Benenson, Paz-Elizur, Adar, Keinan, Livneh, Shapiro 01]

Perhaps we should aim for a network of FSMs?

Cellular automata

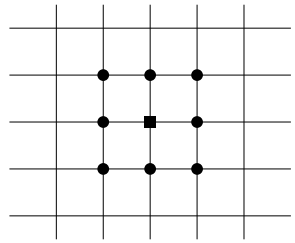
Cellular automata

Infinite grid of FSMs

Cellular automata

Infinite grid of FSMs

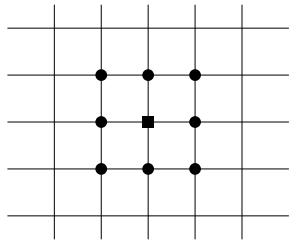
$q_{x,y}(t+1) \longleftarrow q_{x,y}(t), \{q_{x',y'}(t) : \text{grid neighbors } (x', y')\}$



Cellular automata

Infinite grid of FSMs

$q_{x,y}(t+1) \longleftarrow q_{x,y}(t), \{q_{x',y'}(t) : \text{grid neighbors } (x', y')\}$

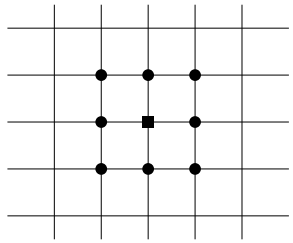


Typical question: How an initial (finite) configuration evolves?

Cellular automata

Infinite grid of FSMs

$q_{x,y}(t+1) \longleftarrow q_{x,y}(t), \{q_{x',y'}(t) : \text{grid neighbors } (x', y')\}$



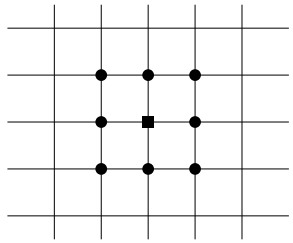
Typical question: How an initial (finite) configuration evolves?

Invented by   (crystal growth, self-replicating systems)

Cellular automata

Infinite grid of FSMs

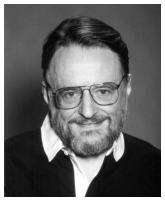
$q_{x,y}(t+1) \longleftarrow q_{x,y}(t), \{q_{x',y'}(t) : \text{grid neighbors } (x', y')\}$



Typical question: How an initial (finite) configuration evolves?

Invented by   (crystal growth, self-replicating systems)

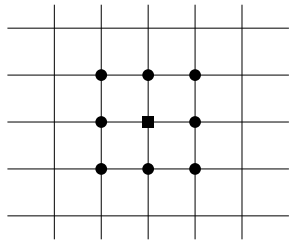
Game of life



Cellular automata

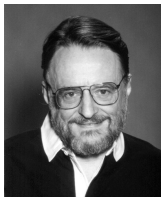
Infinite grid of FSMs

$q_{x,y}(t+1) \longleftarrow q_{x,y}(t), \{q_{x',y'}(t) : \text{grid neighbors } (x', y')\}$



Typical question: How an initial (finite) configuration evolves?

Invented by   (crystal growth, self-replicating systems)

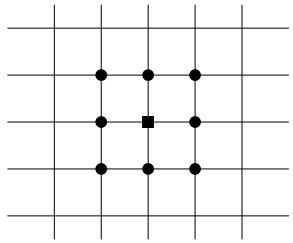
Game of life 

Digital physics  

Cellular automata

Infinite grid of FSMs

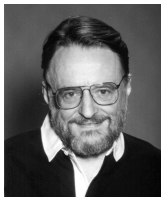
$q_{x,y}(t+1) \longleftarrow q_{x,y}(t), \{q_{x',y'}(t) : \text{grid neighbors } (x', y')\}$



Typical question: How an initial (finite) configuration evolves?

Invented by   (crystal growth, self-replicating systems)

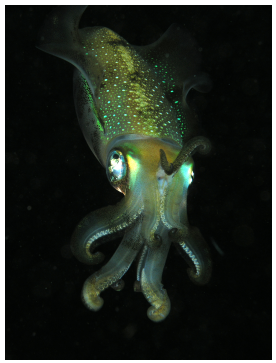
Game of life



Digital physics

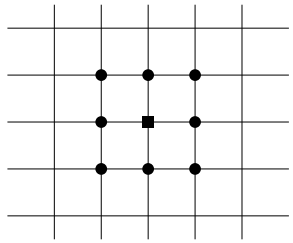
Biological processes

Cellular automata

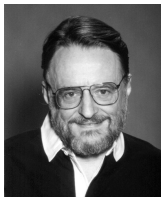
Infinite grid of FSMs

$q_{x,y}(t+1) \longleftarrow q_{x,y}(t), \{q_{x',y'}(t) : \text{grid neighbors } (x', y')\}$

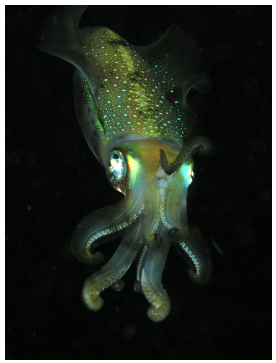


Typical question: How an initial (finite) configuration evolves?

Invented by   (crystal growth, self-replicating systems)

Game of life 

Digital physics  

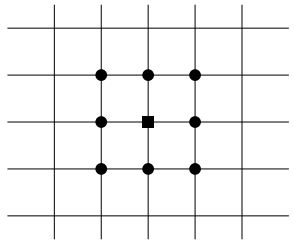
Biological processes  

Highly regular topology

Cellular automata

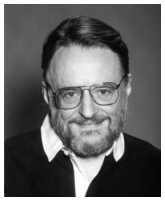
Infinite grid of FSMs

$q_{x,y}(t+1) \longleftarrow q_{x,y}(t), \{q_{x',y'}(t) : \text{grid neighbors } (x', y')\}$

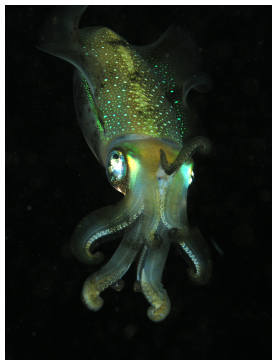


Typical question: How an initial (finite) configuration evolves?

Invented by   (crystal growth, self-replicating systems)

Game of life 

Digital physics  

Biological processes  

Highly regular topology
Synchronous environment

1. Cell biology — a short intro
2. Abstract models
3. Networked finite state machines
   - Results
   - MIS protocol
4. Conclusions

nFSM

nFSM

- Every node is an FSM

nFSM

- Every node is an FSM
- Communication based on transmissions: same message delivered to all neighbors

nFSM

- Every node is an FSM
- Communication based on transmissions: same message delivered to all neighbors
- Constant size messages

nFSM

- Every node is an FSM
- Communication based on transmissions:
  same message delivered to all neighbors
- Constant size messages
  - Message is a letter in a constant-size communication alphabet Σ

nFSM

- Every node is an FSM
- Communication based on transmissions:
  same message delivered to all neighbors
- Constant size messages
  - Message is a letter in a constant-size communication alphabet Σ
- Node u has a port corresponding to each $v \in N(u)$
  - Stores the last message $\sigma \in \Sigma$ delivered from v

nFSM

- Every node is an FSM
- Communication based on transmissions:
  same message delivered to all neighbors
- Constant size messages
  - Message is a letter in a constant-size communication alphabet Σ
- Node u has a port corresponding to each $v \in N(u)$
  - Stores the last message $\sigma \in \Sigma$ delivered from v
- In each step, u decides on next state and which letter to transmit based on its current state and letters currently stored in its ports

nFSM

- Every node is an FSM
- Communication based on transmissions:
  same message delivered to all neighbors
- Constant size messages
  - Message is a letter in a constant-size communication alphabet Σ
- Node u has a port corresponding to each $v \in N(u)$
  - Stores the last message $\sigma \in \Sigma$ delivered from v
- In each step, u decides on next state and which letter to transmit based on its current state and letters currently stored in its ports
- Problem:

nFSM

- Every node is an FSM
- Communication based on transmissions:
  same message delivered to all neighbors
- Constant size messages
  - Message is a letter in a constant-size communication alphabet Σ
- Node u has a port corresponding to each $v \in N(u)$
  - Stores the last message $\sigma \in \Sigma$ delivered from v
- In each step, u decides on next state and which letter to transmit based on its current state and letters currently stored in its ports
- Problem:
  - # possible signals = # port configurations = $|\Sigma|^{\text{degree}(u)}$

nFSM

- Every node is an FSM
- Communication based on transmissions:
  same message delivered to all neighbors
- Constant size messages
  - Message is a letter in a constant-size communication alphabet Σ
- Node u has a port corresponding to each $v \in N(u)$
  - Stores the last message $\sigma \in \Sigma$ delivered from v
- In each step, u decides on next state and which letter to transmit based on its current state and letters currently stored in its ports
- Problem:
  - # possible signals = # port configurations = $|\Sigma|^{\text{degree}(u)}$
  - Should be fixed in an FSM!

nFSM

- Every node is an FSM
- Communication based on transmissions:
  same message delivered to all neighbors
- Constant size messages
  - Message is a letter in a constant-size communication alphabet Σ
- Node u has a port corresponding to each $v \in N(u)$
  - Stores the last message $\sigma \in \Sigma$ delivered from v
- In each step, u decides on next state and which letter to transmit based on its current state and letters currently stored in its ports
- Problem:
  - # possible signals = # port configurations = $|\Sigma|^{\text{degree}(u)}$
  - Should be fixed in an FSM!
  - How does u interpret the content of its ports?

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$
- When in state q, node u cares only about the number π_σ of appearances of σ in its ports (currently)

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$
- When in state q, node u cares only about the number π_σ of appearances of σ in its ports (currently)
- # possible signals $= \mathrm{degree}(u) + 1$

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$
- When in state q, node u cares only about the number π_σ of appearances of σ in its ports (currently)
- # possible signals $= \mathrm{degree}(u) + 1$
- π_σ calculated by the one-two-many principle:

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$
- When in state q, node u cares only about the number π_σ of appearances of σ in its ports (currently)
- # possible signals $= \mathrm{degree}(u) + 1$
- π_σ calculated by the one-two-many principle:
  isolated cultures developed counting systems that don't go beyond 2

Walpiri (Australia)

Piraha (the Amazon)

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$
- When in state q, node u cares only about the number π_σ of appearances of σ in its ports (currently)
- # possible signals $= \mathrm{degree}(u) + 1$
- π_σ calculated by the one-two-many principle:
  isolated cultures developed counting systems that don't go beyond 2

Walpiri (Australia)

Piraha (the Amazon)

- Constant bounding parameter $b \in \mathbb{Z}_{>0}$ (property of the protocol)

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$
- When in state q, node u cares only about the number π_σ of appearances of σ in its ports (currently)
- # possible signals $= \mathrm{degree}(u) + 1$
- π_σ calculated by the one-two-many principle:
  isolated cultures developed counting systems that don't go beyond 2

Walpiri (Australia)

Piraha (the Amazon)

- Constant bounding parameter $b \in \mathbb{Z}_{>0}$ (property of the protocol)
- u can distinguish between $\pi_\sigma = 0, 1, \ldots, b-1$, or $\pi_\sigma \geq b$

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$
- When in state q, node u cares only about the number π_σ of appearances of σ in its ports (currently)
- # possible signals $= \text{degree}(u) + 1$
- π_σ calculated by the one-two-many principle:
  isolated cultures developed counting systems that don't go beyond 2

Walpiri (Australia)

Piraha (the Amazon)

  - Constant bounding parameter $b \in \mathbb{Z}_{>0}$ (property of the protocol)
  - u can distinguish between $\pi_\sigma = 0, 1, \ldots, b-1$, or $\pi_\sigma \geq b$

- Transition function of the FSM:
  $\delta : Q \times \{0, 1, \ldots, b-1, \geq b\} \to Q \times \Sigma$

The one-two-many principle

- Each state $q \in Q$ is associated with a query letter $\sigma = \sigma(q) \in \Sigma$
- When in state q, node u cares only about the number π_σ of appearances of σ in its ports (currently)
- # possible signals $= \text{degree}(u) + 1$
- π_σ calculated by the one-two-many principle:
  isolated cultures developed counting systems that don't go beyond 2

Walpiri (Australia)

Piraha (the Amazon)

  - Constant bounding parameter $b \in \mathbb{Z}_{>0}$ (property of the protocol)
  - u can distinguish between $\pi_\sigma = 0, 1, \ldots, b-1$, or $\pi_\sigma \geq b$

- Transition function of the FSM:
  $\delta : Q \times \{0, 1, \ldots, b-1, \geq b\} \to 2^{Q \times \Sigma}$

Crux of the model

- Applicable to arbitrary network topologies

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM
- All parameters of the protocol are constants, independent of any feature of the input graph (including $\mathrm{degree}(u)$):

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM
- All parameters of the protocol are constants, independent of any feature of the input graph (including $\mathrm{degree}(u)$):
  - number of states

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM
- All parameters of the protocol are constants, independent of any feature of the input graph (including $\mathrm{degree}(u)$):
  - number of states
  - size of alphabet Σ

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM
- All parameters of the protocol are constants, independent of any feature of the input graph (including $\mathrm{degree}(u)$):
  - number of states
  - size of alphabet Σ
  - bounding parameter b

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM
- All parameters of the protocol are constants, independent of any feature of the input graph (including $\mathrm{degree}(u)$):
  - number of states
  - size of alphabet Σ
  - bounding parameter b
  - size of the description of $\delta : Q \times \{0, 1, \ldots, b-1, \geq b\} \to 2^{Q \times \Sigma}$

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM
- All parameters of the protocol are constants, independent of any feature of the input graph (including $\mathrm{degree}(u)$):
  - number of states
  - size of alphabet Σ
  - bounding parameter b
  - size of the description of $\delta : Q \times \{0, 1, \ldots, b-1, \geq b\} \rightarrow 2^{Q \times \Sigma}$
- A genuine FSM!

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM
- All parameters of the protocol are constants, independent of any feature of the input graph (including $\mathrm{degree}(u)$):
  - number of states
  - size of alphabet Σ
  - bounding parameter b
  - size of the description of $\delta : Q \times \{0, 1, \ldots, b-1, \geq b\} \rightarrow 2^{Q \times \Sigma}$
- A genuine FSM!
- The biological angle:

Crux of the model

- Applicable to arbitrary network topologies
- Fully asynchronous environment
- Nodes run the same (randomized) FSM
- All parameters of the protocol are constants, independent of any feature of the input graph (including $\mathrm{degree}(u)$):
  - number of states
  - size of alphabet Σ
  - bounding parameter b
  - size of the description of $\delta : Q \times \{0, 1, \ldots, b-1, \geq b\} \to 2^{Q \times \Sigma}$
- A genuine FSM!
- The biological angle:
  - one-two-many counting $=$
    discrete analogue for detecting different concentration levels

nFSM — computability

nFSM — computability

Theorem

The execution of an nFSM protocol can be simulated by a (randomized) linear-space Turing machine.

nFSM — computability

Theorem

The execution of an nFSM protocol can be simulated by a (randomized) linear-space Turing machine.

Theorem

The execution of a (randomized) linear-space Turing machine can be simulated by an nFSM protocol on a path.

nFSM — computability

Theorem

The execution of an nFSM protocol can be simulated by a (randomized) linear-space Turing machine.

Theorem

The execution of a (randomized) linear-space Turing machine can be simulated by an nFSM protocol on a path.

Corollary (Formal languages)

nFSM protocols do not exceed level 3 (out of 4) in Chomsky's hierarchy.

nFSM — computability

Theorem

The execution of an nFSM protocol can be simulated by a (randomized) linear-space Turing machine.

Theorem

The execution of a (randomized) linear-space Turing machine can be simulated by an nFSM protocol on a path.

Corollary (Formal languages)

nFSM protocols do not exceed level 3 (out of 4) in Chomsky's hierarchy.

Observation (Anonymous networks)

Leader election and consensus are impossible under nFSM.

Performance measure

Run-time:

Performance measure

Run-time:

- # time units until all nodes terminate
  - adversarial delays ≤ 1 time units

Performance measure

Run-time:

- # time units until all nodes terminate
  - adversarial delays ≤ 1 time units
- Las Vegas algorithms

Performance measure

Run-time:

- # time units until all nodes terminate
  - adversarial delays ≤ 1 time units
- Las Vegas algorithms
- Run-time bounds hold in expectation and w.h.p.

Performance measure

Run-time:

- # time units until all nodes terminate
  - adversarial delays ≤ 1 time units
- Las Vegas algorithms
- Run-time bounds hold in expectation and w.h.p.
- Efficient algorithm $= \log^{O(1)} n$ run-time [Linial 92]

Synchronizer

Theorem

Every nFSM protocol designed to operate in a synchronous environment can be simulated in an asynchronous environment with a constant multiplicative run-time overhead.

Synchronizer

Theorem

Every nFSM protocol designed to operate in a synchronous environment can be simulated in an asynchronous environment with a constant multiplicative run-time overhead.

Makes life much easier for the protocol designer

Efficient algorithm: maximal independent set

Efficient algorithm: maximal independent set

Theorem

There exists an nFSM protocol that computes an MIS in any n-node graph in time $O(\log^2 n)$.

Efficient algorithm: maximal independent set

Theorem

There exists an nFSM protocol that computes an MIS in any n-node graph in time $O(\log^2 n)$.

Message passing model:

- $O(\log n)$ [Luby 86], [Alon, Babai, Itai 86]
- $\Omega\left(\sqrt{\log n}\right)$ [Kuhn, Moscibroda, Wattenhofer 04]
- Anonymous networks, constant size messages:
  $\Omega(\log n)$ [Kothapalli, Onus, Scheideler, Schindelhauer 06]
  $O(\log n)$ [Métivier, Robson, Sehab-Djahromi, Zemmari 11]

Efficient algorithm: maximal independent set

Theorem

There exists an nFSM protocol that computes an MIS in any n-node graph in time $O(\log^2 n)$.

Message passing model:

- $O(\log n)$ [Luby 86], [Alon, Babai, Itai 86]
- $\Omega\left(\sqrt{\log n}\right)$ [Kuhn, Moscibroda, Wattenhofer 04]
- Anonymous networks, constant size messages:
  $\Omega(\log n)$ [Kothapalli, Onus, Scheideler, Schindelhauer 06]
  $O(\log n)$ [Métivier, Robson, Sehab-Djahromi, Zemmari 11]

The beeping model:

- $O(\log^2 n)$–$O(\log^3 n)$
  [Afek, Alon, Barad, Hornstein, Barkai, Bar-Joseph 11],
  [Afek, Alon, Bar-Joseph, Cornejo, Haeupler, Kuhn 11]

Efficient algorithm: $(\Delta + 1)$-coloring

Efficient algorithm: $(\Delta + 1)$-coloring

Irrelevant for arbitrary graphs (can't specify output)

Efficient algorithm: $(\Delta + 1)$-coloring

Irrelevant for arbitrary graphs (can't specify output)

Theorem

Given some constant d, there exists an nFSM protocol that $(d+1)$-colors any n-node graph satisfying $\Delta \leq d$ in time $O(\log n)$.

Efficient algorithm: $(\Delta + 1)$-coloring

Irrelevant for arbitrary graphs (can't specify output)

Theorem

Given some constant d, there exists an nFSM protocol that $(d+1)$-colors any n-node graph satisfying $\Delta \leq d$ in time $O(\log n)$.

Message passing model:

- $O(\Delta + \log^* n)$ [Barenboim, Elkin 09], [Kuhn 09]
- $\Omega(\log^* n)$ [Linial 92]
- $O(\log \Delta + \sqrt{\log n})$ [Schneider, Wattenhofer 10]
- Anonymous networks, constant size messages: $\Omega(\log n)$ [Kothapalli, Onus, Scheideler, Schindelhauer 06]

Efficient algorithm: tree 3-coloring

Efficient algorithm: tree 3-coloring

Theorem

There exists an nFSM protocol that 3-colors any (undirected) n-node tree in time $O(\log n)$.

Efficient algorithm: tree 3-coloring

Theorem

There exists an nFSM protocol that 3-colors any (undirected) n-node tree in time $O(\log n)$.

Message passing model:

- Anonymous (undirected) trees, constant size messages: $\Omega(\log n)$ [Kothapalli, Onus, Scheideler, Schindelhauer 06]
- Directed trees:
  - $O(\log^* n)$ [Cole, Vishkin 86]
  - $\Omega(\log^* n)$ [Linial 92]

Efficient algorithm: maximal matching

Efficient algorithm: maximal matching

Theorem

Under a small inevitable modification to the model, there exists an nFSM protocol that computes an MM in any n-node graph in time $O(\log^2 n)$.

Efficient algorithm: maximal matching

Theorem

Under a small inevitable modification to the model, there exists an nFSM protocol that computes an MM in any n-node graph in time $O(\log^2 n)$.

Message passing model:

- $O(\log n)$ [Israeli, Itai 86]
- $\Omega\left(\sqrt{\log n}\right)$ [Kuhn, Moscibroda, Wattenhofer 04]
- Anonymous networks, constant size messages: $\Omega(\log n)$ [Kothapalli, Onus, Scheideler, Schindelhauer 06]

Simulating executions on the square graph

Lemma

For every nFSM protocol Π with bounding parameter $b = 1$, there exists an nFSM protocol Π^2 such that for every graph G, the execution of Π^2 on G simulates the execution of Π on G^2 with a constant multiplicative run-time overhead.

Efficient algorithms: 2-hop variants

Corollary (Maximal 2-hop independent set)

There exists an nFSM protocol that computes a maximal 2-hop independent set for any n-node graph in time $O(\log^2 n)$.

Corollary (2-hop coloring)

Given some constant d*, there exists an nFSM protocol that computes a 2-hop coloring with* (d^2+1) *colors for any n-node graph satisfying* $\Delta \leq d$ *in time* $O(\log n)$.

Efficient algorithms: 2-hop variants

Corollary (Maximal 2-hop independent set)

There exists an nFSM protocol that computes a maximal 2-hop independent set for any n-node graph in time $O(\log^2 n)$.

Corollary (2-hop coloring)

Given some constant d, there exists an nFSM protocol that computes a 2-hop coloring with (d^2+1) colors for any n-node graph satisfying $\Delta \leq d$ in time $O(\log n)$.

Theorem

Maximal k-hop independent set and k-hop coloring are impossible in anonymous networks for any $k \geq 3$.

MIS under nFSM — difficulties

MIS under nFSM — difficulties

- Existing MIS algorithms rely on grouping rounds into phases: u competes with $N(u)$ over joining the MIS

MIS under nFSM — difficulties

- Existing MIS algorithms rely on grouping rounds into phases: u competes with $N(u)$ over joining the MIS
- Require either
  - calculations with super-constant variables
  - independent communication with each neighbor
  - messages of logarithmic size

MIS under nFSM — difficulties

- Existing MIS algorithms rely on grouping rounds into phases: u competes with $N(u)$ over joining the MIS
- Require either
  - calculations with super-constant variables
  - independent communication with each neighbor
  - messages of logarithmic size
- Idea: transmit $O(1)$ bits per round
  - logarithmically long phases

MIS under nFSM — difficulties

- Existing MIS algorithms rely on grouping rounds into phases: u competes with $N(u)$ over joining the MIS
- Require either
  - calculations with super-constant variables
  - independent communication with each neighbor
  - messages of logarithmic size
- Idea: transmit $O(1)$ bits per round
  - logarithmically long phases
- Problem:
  - u must count the rounds in a phase (deciding when it ends)

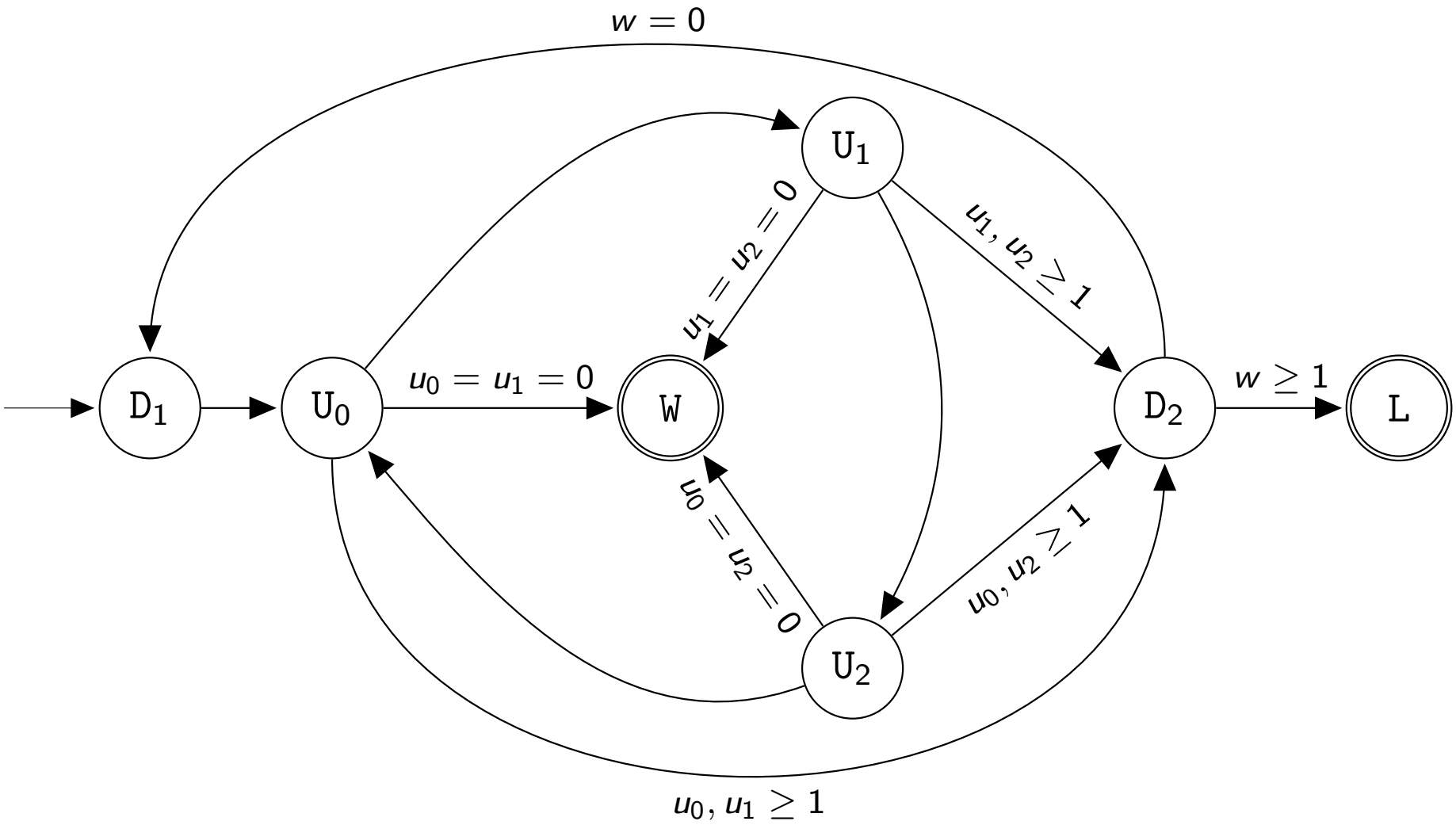
MIS under nFSM — difficulties

- Existing MIS algorithms rely on grouping rounds into phases: u competes with $N(u)$ over joining the MIS
- Require either
  - calculations with super-constant variables
  - independent communication with each neighbor
  - messages of logarithmic size
- Idea: transmit $O(1)$ bits per round
  - logarithmically long phases
- Problem:
  - u must count the rounds in a phase (deciding when it ends)
  - phases must be aligned to guarantee fair competition

MIS under nFSM — difficulties

- Existing MIS algorithms rely on grouping rounds into phases: u competes with $N(u)$ over joining the MIS
- Require either
  - calculations with super-constant variables
  - independent communication with each neighbor
  - messages of logarithmic size
- Idea: transmit $O(1)$ bits per round
  - logarithmically long phases
- Problem:
  - u must count the rounds in a phase (deciding when it ends)
  - phases must be aligned to guarantee fair competition
- How can we decide if u joins MIS without long aligned phases?

MIS under nFSM — solution



MIS under nFSM — solution

- Relax requirement that phase is aligned and of predetermined length

MIS under nFSM — solution

- Relax requirement that phase is aligned and of predetermined length
- Tournament:
  - length determined probabilistically
  - "softly" aligned
  - maintained under nFSM

MIS under nFSM — solution

- Relax requirement that phase is aligned and of predetermined length
- Tournament:
  - length determined probabilistically
  - "softly" aligned
  - maintained under nFSM
- Prove:
  1. Amortized length of a tournament is $O(\log n)$ w.h.p.

MIS under nFSM — solution

- Relax requirement that phase is aligned and of predetermined length
- Tournament:
  - length determined probabilistically
  - "softly" aligned
  - maintained under nFSM
- Prove:
  1. Amortized length of a tournament is $O(\log n)$ w.h.p.
  2. Guarantee fair competition $\Longrightarrow$
     const fraction of the edges is removed with const probability $\Longrightarrow$
     $O(\log n)$ tournaments w.h.p.

Summary

- Abstract model for network of FSMs

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols
- Suitable to biological cellular networks
  - Local computation, communication, asynchrony

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols
- Suitable to biological cellular networks
  - Local computation, communication, asynchrony
  - Also networks of man made nano-devices

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols
- Suitable to biological cellular networks
  - Local computation, communication, asynchrony
  - Also networks of man made nano-devices
- Reasonable constants: $|Q| = |\Sigma| = 7$, $b = 1$.

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols
- Suitable to biological cellular networks
  - Local computation, communication, asynchrony
  - Also networks of man made nano-devices
- Reasonable constants: $|Q| = |\Sigma| = 7$, $b = 1$.
- Independent theoretical interest

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols
- Suitable to biological cellular networks
  - Local computation, communication, asynchrony
  - Also networks of man made nano-devices
- Reasonable constants: $|Q| = |\Sigma| = 7$, $b = 1$.
- Independent theoretical interest
- **Biology through the DC lens**

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols
- Suitable to biological cellular networks
  - Local computation, communication, asynchrony
  - Also networks of man made nano-devices
- Reasonable constants: $|Q| = |\Sigma| = 7$, $b = 1$.
- Independent theoretical interest
- **Biology through the DC lens**

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols
- Suitable to biological cellular networks
  - Local computation, communication, asynchrony
  - Also networks of man made nano-devices
- Reasonable constants: $|Q| = |\Sigma| = 7$, $b = 1$.
- Independent theoretical interest
- **Biology through the DC lens**

Joint works with Jochen Seidel, Jasmin Smula, and Roger Wattenhofer

Summary

- Abstract model for network of FSMs
- Computational power slightly weaker
- Fundamental DC problems still admit efficient protocols
- Suitable to biological cellular networks
  - Local computation, communication, asynchrony
  - Also networks of man made nano-devices
- Reasonable constants: $|Q| = |\Sigma| = 7$, $b = 1$.
- Independent theoretical interest
- **Biology through the DC lens**

Joint works with Jochen Seidel, Jasmin Smula, and Roger Wattenhofer

OBRIGADO