UNIVERSITY OF LIVERPOOL

LEVERHULME TRUST

Fault Tolerant Network Constructors

Othon Michail, Paul G. Spirakis, Michail Theofilatos
Department of Computer Science, University of Liverpool, UK

21st International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2019)
25 October 2019

The Leverhulme Research Centre for Functional Materials Design

Contents

Network Constructors

Fundamental Problem

Algorithmic distributed construction of an actual communication topology

- Distributed computing model, formed by resource limited mobile *agents*
- Agents (or *processes*) can form/delete connections between them
- on/off case: a connection either exists (active) or not (inactive)
- Initially all connections are inactive

Goal: End up with a desired stable graph

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

Network Constructors

The model

Q: *finite set of node−states*
$q_0 \in Q$: *initial node−state*
$Q_{out} \subseteq Q$: *set of output node−states*
$\delta: Q \times Q \times \{0,1\} \rightarrow Q \times Q \times \{0,1\}$: *the transition function*
In every step, a pair uv is selected by the scheduler and u, v interact according to δ

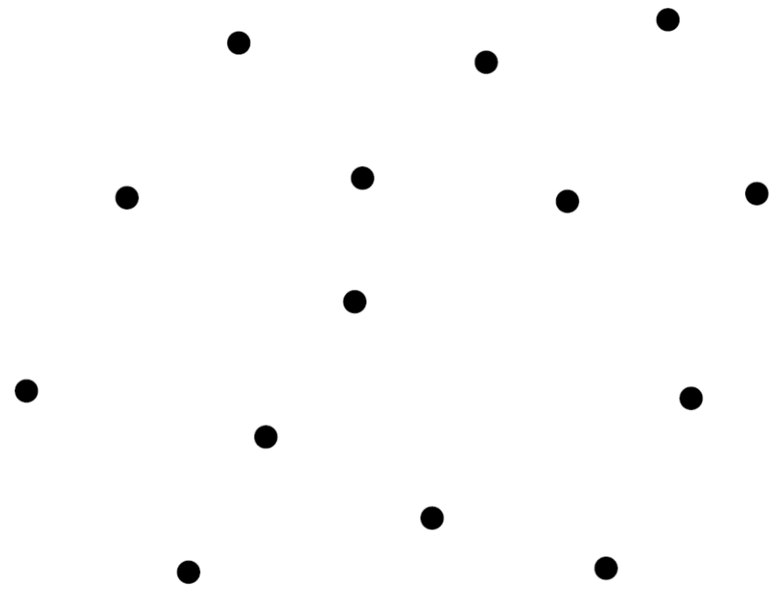
- Fair scheduler: A scheduler is fair if it always leads to fair executions. An infinite execution is fair if for every pair of configurations C and C' such that $C \rightarrow C'$, if C occurs infinitely often, then so does C'
- Output network: nodes that are in output states and edges between them that are active
- Stability: The output network cannot change in future steps

Network Constructors - Example

Spanning Star

- 2 states: black and red
- Initially all black
- Constructs a global star
- Protocol: $(b, b, 0) \rightarrow (b, r, 1)$
  $(r, r, 1) \rightarrow (r, r, 0)$
  $(b, r, 0) \rightarrow (b, r, 1)$

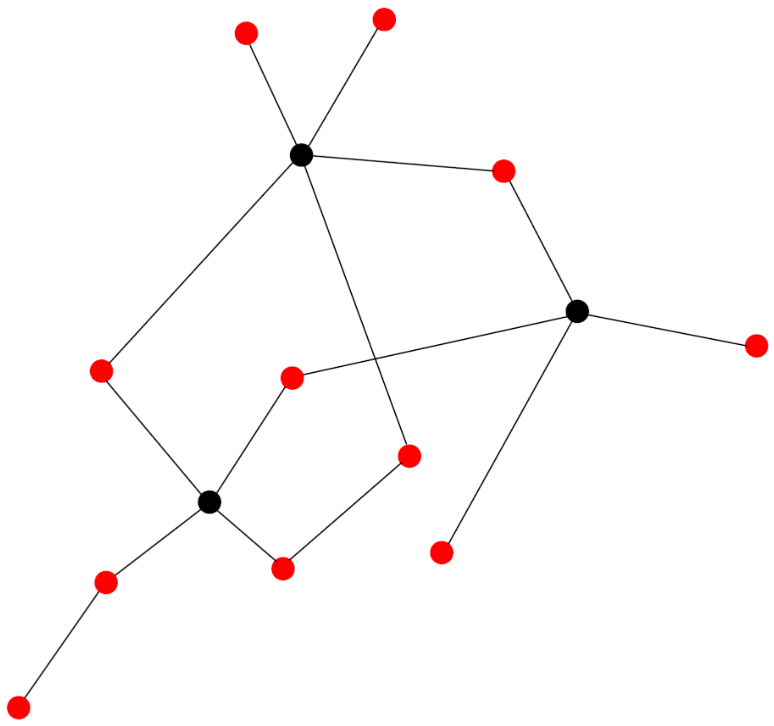
- Space: 2 states
- Time: $O(n^2 \log n)$
- Optimal w.r.t. both



Network Constructors - Example

Spanning Star

- 2 states: black and red
- Initially all black
- Constructs a global star
- Protocol: $(b, b, 0) \to (b, r, 1)$
  $(r, r, 1) \to (r, r, 0)$
  $(b, r, 0) \to (b, r, 1)$

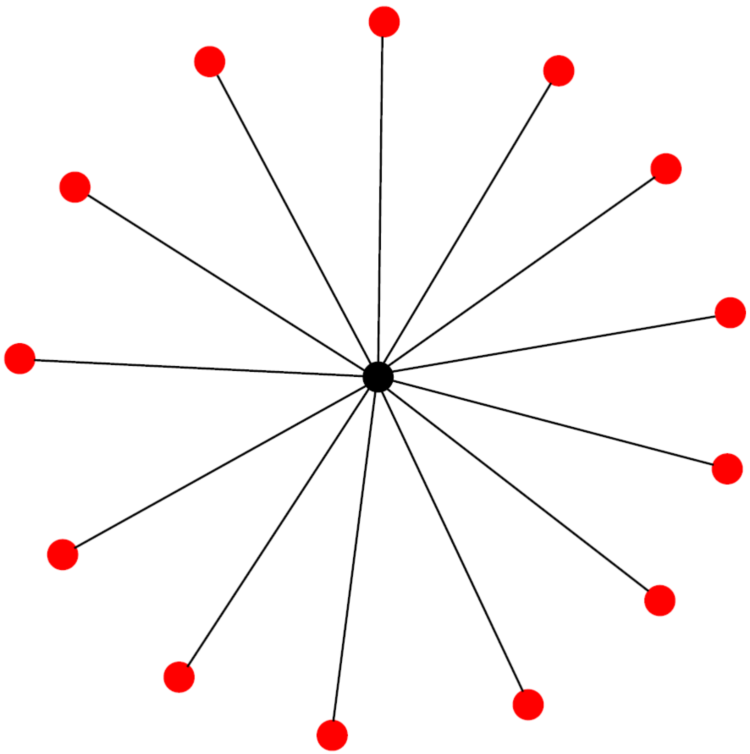


- Space: 2 states
- Time: $O(n^2 \log n)$
- Optimal w.r.t. both

Network Constructors - Example

Spanning Star

- 2 states: black and red
- Initially all black
- Constructs a global star
- Protocol: $(b, b, 0) \rightarrow (b, r, 1)$
  $(r, r, 1) \rightarrow (r, r, 0)$
  $(b, r, 0) \rightarrow (b, r, 1)$



- Space: 2 states
- Time: $O(n^2 \log n)$
- Optimal w.r.t. both

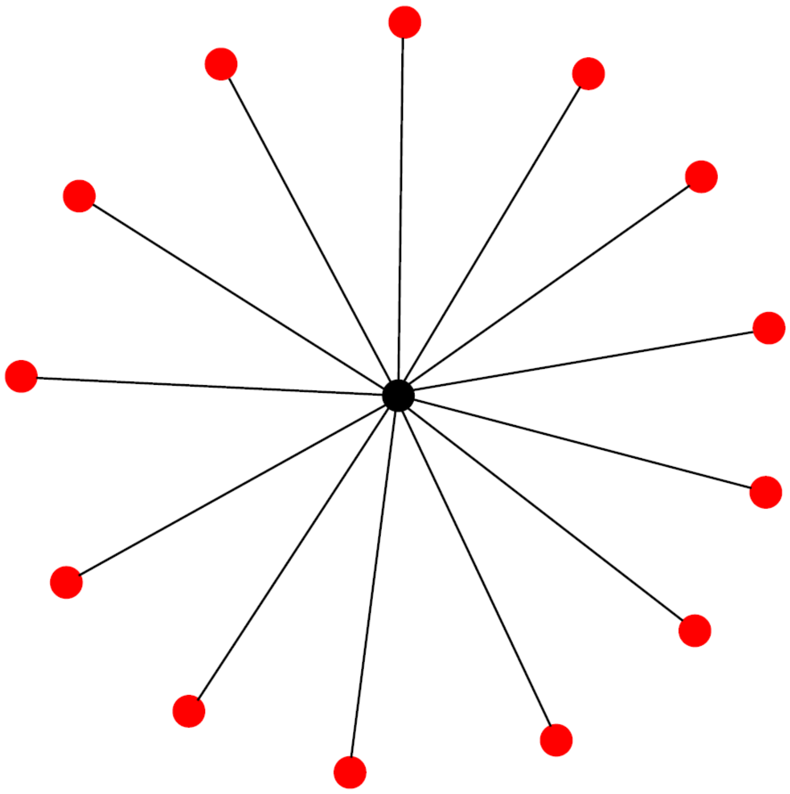
Fault Tolerance

- In each step, either two nodes are selected for interaction, or one node crashes
- During a crash failure, the node and all its edges (active or inactive) are removed from the configuration
- The goal is to find protocols that always re-stabilize to a “correct” graph

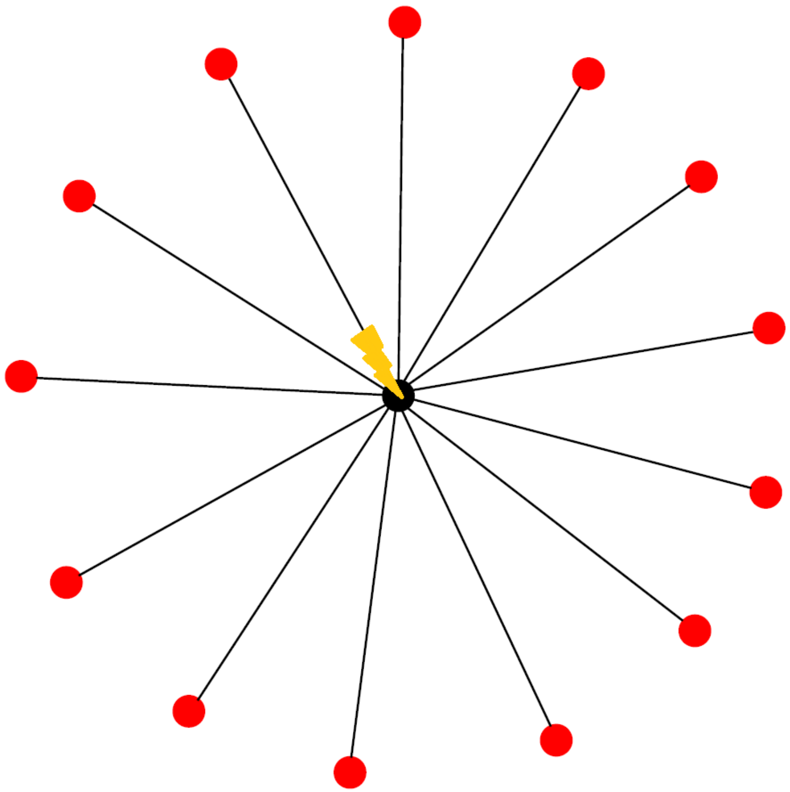
Questions

- If one or more faults can affect the formation process, can we always re-stabilize to a correct graph?
- What is the class of graph languages for which there exist fault-tolerant protocols?
- What are the additional minimal assumptions that we need to make in order to find fault-tolerant protocols for a bigger class of graph languages?

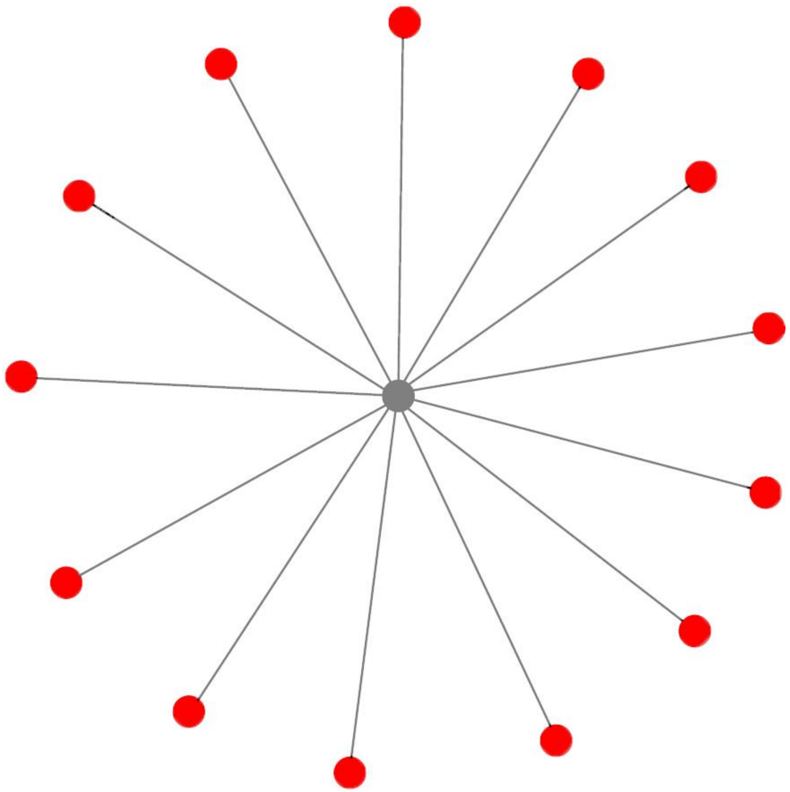
Fault Tolerance - Example



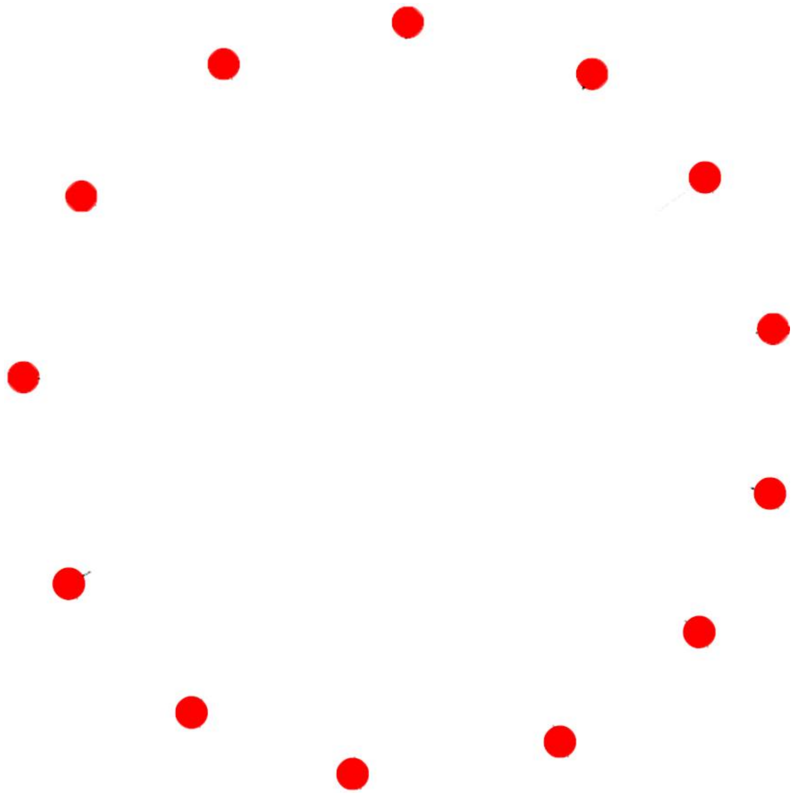
Fault Tolerance - Example



Fault Tolerance - Example



Fault Tolerance - Example



Some Definitions

Constructibility

We say that a protocol Π constructs a graph language L, if:

1. Every execution of Π on n nodes stabilizes to a graph $G \in L$ s.t. $|V(G)| = n$, and
2. $\forall G \in L$ there is an execution of Π on $|V(G)|$ nodes that stabilizes to G.

Partial Constructibility

We say that a protocol Π partially constructs a graph language L, if:

1. (1) from Definition 1 holds, and
2. $\exists G \in L$ s.t. no execution of Π on $|V(G)|$ nodes stabilizes to G.

Some Definitions

Fault-Tolerant Protocol

Let Π be a *NET* protocol that, in a failure-free setting, constructs a graph $G \in L$. Π is called f*-fault-tolerant* if for any population size $n > f$, any execution of Π constructs a graph $G \in L$, where $|V(G)| = n - f$. We also call Π fault-tolerant if the same holds for any number $f \leq n - 2$ of faults.

Constructible language

A graph language L is called constructible (partially constructible) if there is a protocol that constructs (partially constructs) it. Similarly, we call L constructible under f faults, if there is an f-fault-tolerant protocol that constructs L, where f is an upper bound on the maximum number of faults.

Our Results

Constructible languages		
Without notifications		With notifications
Unbounded faults	Bounded faults	Unbounded faults
Only Spanning Clique	Non-hereditary impossibility	Fault-tolerant protocols: Spanning Star, Cycle Cover, Spanning Line
Strong impossibility even with linear waste	A representation of any finite graph (partial constructibility)	Universal Fault-tolerant Constructors (with waste)
	Any constructible graph language with linear waste	Universal Fault-tolerant Restart (without waste)

Our Results

Constructible languages		
Without notifications		With notifications
Unbounded faults	Bounded faults	Unbounded faults
Only Spanning Clique	Non-hereditary impossibility	Fault-tolerant protocols: Spanning Star, Cycle Cover, Spanning Line
Strong impossibility even with linear waste	A representation of any finite graph (partial constructibility)	Universal Fault-tolerant Constructors (with waste)
	Any constructible graph language with linear waste	Universal Fault-tolerant Restart (without waste)

Fault Tolerant Spanning Clique

Transition function:

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

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

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

- The above protocol constructs a spanning clique, tolerating any number of faults
- Spanning Clique is the only constructible graph language in the unbounded-faults case
  - Even if we allow linear waste

Our Results

Constructible languages		
Without notifications		With notifications
Unbounded faults	Bounded faults	Unbounded faults
Only Spanning Clique	Non-hereditary impossibility	Fault-tolerant protocols: Spanning Star, Cycle Cover, Spanning Line
Strong impossibility even with linear waste	A representation of any finite graph (partial constructibility)	Universal Fault-tolerant Constructors (with waste)
	Any constructible graph language with linear waste	Universal Fault-tolerant Restart (without waste)

Non-hereditary Graph Languages

Hereditary Language

A graph language L is called *Hereditary* if for any graph $G \in L$, every induced subgraph of G also belongs to L.

- If there exists a graph $G \in L$, such that after removing any node (crash fault), the resulting graph $G' \notin L$, then there is no protocol that stably constructs L.
- If there was a protocol that changes the configuration in order to "fix" the graph, then this would happen indefinitely and the protocol would never be stabilizing.
- This means that if a graph language in non-hereditary, it is impossible to be constructed under a single fault.

Our Results

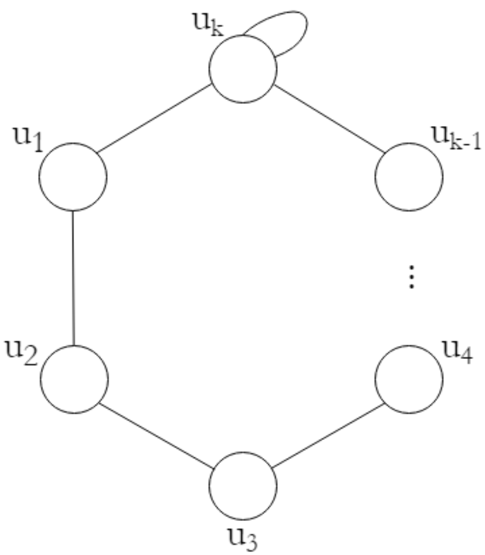
Constructible languages		
Without notifications		With notifications
Unbounded faults	Bounded faults	Unbounded faults
Only Spanning Clique	Non-hereditary impossibility	Fault-tolerant protocols: Spanning Star, Cycle Cover, Spanning Line
Strong impossibility even with linear waste	A representation of any finite graph (partial constructibility)	Universal Fault-tolerant Constructors (with waste)
	Any constructible graph language with linear waste	Universal Fault-tolerant Restart (without waste)

Partial Constructibility

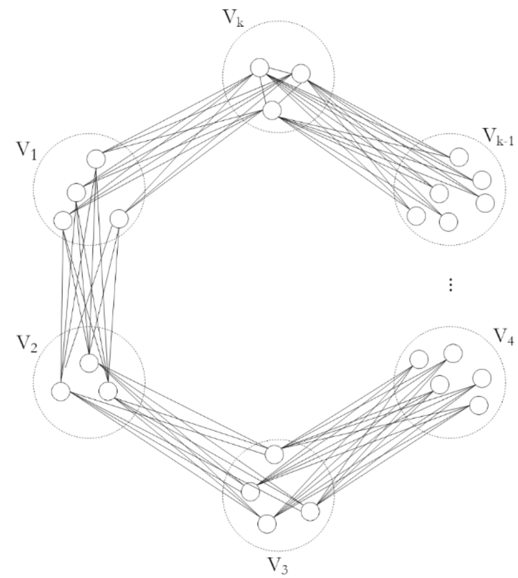
There exists a class of graph languages that is partially constructible in the case of bounded number of faults.

- Class of graph languages $L_{D,f}$
  - $D = ([k], H)$
  - $f < k$ is the finite upper bound on the number of faults
- A graph $G = (V, E)$ belongs to $L_{D,f}$ iff there are k partitions $V_1, V_2, \dots, V_k$ of V s.t. for all $1 \leq i, j \leq k$, $\left| |V_i| - |V_j| \right| \leq f + 1$
- The graph D defines a neighbouring relation between the partitions. For every $(i, j) \in H$, E contains all edges between partitions V_i and V_j.

Partial Constructibility



$D = ([k], H)$



Graph of supernodes $G = (V, E)$

- We provide a protocol which partitions the population into $k = 2^i$ groups.
- It constructs any graph language $L_{D,f}$ (as described before), where $k = 2^i$.
- The partitioning can be used in order to construct any (constructible) graph language on at least $\frac{n}{2f} - f$ nodes, where f is the number of faults

Our Results

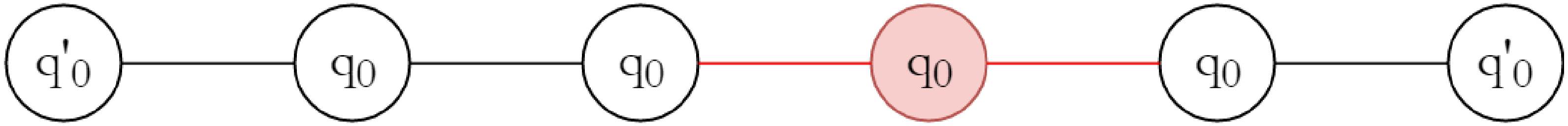
Constructible languages		
Without notifications		With notifications
Unbounded faults	Bounded faults	Unbounded faults
Only Spanning Clique	Non-hereditary impossibility	Fault-tolerant protocols: Spanning Star, Cycle Cover, Spanning Line
Strong impossibility even with linear waste	A representation of any finite graph (partial constructibility)	Universal Fault-tolerant Constructors (with waste)
	Any constructible graph language with linear waste	Universal Fault-tolerant Restart (without waste)

Notified Network Constructors

- We now extend the original model with a minimal form of fault notifications.
- When a node u crashes, all the nodes that maintain an active connection with it at that time, are notified (a fault flag becomes 1).
- If no such nodes exist (i.e., u is isolated), then an arbitrary node is notified.
- In this way, we guarantee that at least one node will “sense” the crash failure.

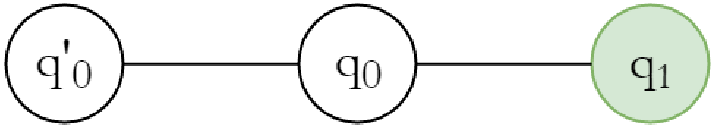
Notified Network Constructors

- We now extend the original model with a minimal form of fault notifications.
- When a node u crashes, all the nodes that maintain an active connection with it at that time, are notified (a fault flag becomes 1).
- If no such nodes exist (i.e., u is isolated), then an arbitrary node is notified.
- In this way, we guarantee that at least one node will “sense” the crash failure.



Notified Network Constructors

- We now extend the original model with a minimal form of fault notifications.
- When a node u crashes, all the nodes that maintain an active connection with it at that time, are notified (a fault flag becomes 1).
- If no such nodes exist (i.e., u is isolated), then an arbitrary node is notified.
- In this way, we guarantee that at least one node will "sense" the crash failure.



Our Results

Constructible languages		
Without notifications		With notifications
Unbounded faults	Bounded faults	Unbounded faults
Only Spanning Clique	Non-hereditary impossibility	Fault-tolerant protocols: Spanning Star, Cycle Cover, Spanning Line
Strong impossibility even with linear waste	A representation of any finite graph (partial constructibility)	Universal Fault-tolerant Constructors (with waste)
	Any constructible graph language with linear waste	Universal Fault-tolerant Restart (without waste)

Notified Network Constructors

- Some otherwise infeasible graph languages are now constructible under any number of faults
  - Spanning Star
  - Cycle Cover
  - Spanning Line

$Q = \{q_0,\ q_1,\ q_2\} \times \{0,1\}$
Initial state: q_0

δ_1 :
$(q_0,\ q_0,\ 0) \rightarrow (q_1,\ q_1,\ 1)$
$(q_1,\ q_0,\ 0) \rightarrow (q_2,\ q_1,\ 1)$
$(q_1,\ q_1,\ 0) \rightarrow (q_2,\ q_2,\ 1)$

δ_2 :
$(q_1, 1) \rightarrow (q_0, 0)$
$(q_2, 1) \rightarrow (q_1, 0)$

Fault Tolerant Cycle-Cover Protocol

$Q = \{b,\ r\} \times \{0,1\}$
Initial state: b

δ_1 :
$(b,\ b,\ 0) \rightarrow (b,\ r,\ 1)$
$(b,\ b,\ 1) \rightarrow (b,\ r,\ 1)$
$(r,\ r,\ 1) \rightarrow (b,\ b,\ 0)$
$(b,\ r,\ 0) \rightarrow (b,\ r,\ 1)$

δ_2 :
$(r, 1) \rightarrow (b, 0)$

Fault Tolerant Spanning Star Protocol

Notified Network Constructors

$Q = \{q_0,\ q_2,\ e_1,\ e_2,\ l_0,\ l_1,\ w,\ w_1,\ w_2\} \times \{0,1\}$
Initial state: q_0

δ_1 :
$(q_0,\ q_0,\ 0) \rightarrow (e_1,\ l_0,\ 1)$
$(l,\ q_0,\ 0) \rightarrow (q_2,\ l_0,\ 1)$
$(l_0,\ l_0,\ 0) \rightarrow (q_2,\ w,\ 1)$

\\\\w nodes perform a random walk on line
$(l_1,\ q_2,\ 1) \rightarrow (e_1,\ w_1,\ 1)$
$(w_i,\ q_2,\ 1) \rightarrow (q_2,\ w_i,\ 1)$
$(w,\ q_2,\ 1) \rightarrow (q_2,\ w,\ 1)$

$(w,\ e_i,\ 1) \rightarrow (w_i,\ e_i,\ 1)$
$(w_i,\ e_i,\ 1) \rightarrow (w_j,\ e_j,\ 1),\ i \neq j$
$(w_i,\ e_j,\ 1) \rightarrow (q_2,\ l_0,\ 1),\ i \neq j$
$(w,\ l_i,\ 1) \rightarrow (w_1,\ e_1,\ 1)$
$(w_i,\ l_i,\ 1) \rightarrow (q_2,\ l_0,\ 1)$

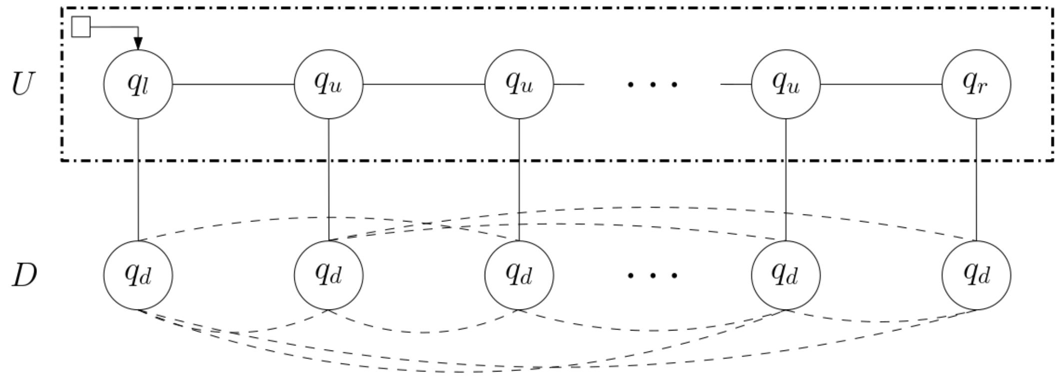
\\\\w nodes eliminate each other, until only one survives
$(w_i,\ w_j,\ 1) \rightarrow (w,\ q_2,\ 1)$
$(w,\ w_j,\ 1) \rightarrow (w,\ q_2,\ 1)$

δ_2 :
$(e_1, 1) \rightarrow (q_0, 0)$
$(e_2, 1) \rightarrow (q_0, 0)$
$(l_0, 1) \rightarrow (q_0, 0)$
$(l_1, 1) \rightarrow (q_0, 0)$
$(q_2, 1) \rightarrow (l_1, 0)$
$(w, 1) \rightarrow (l_1, 0)$
$(w_1, 1) \rightarrow (l_1, 0)$
$(w_2, 1) \rightarrow (l_1, 0)$

Fault Tolerant Spanning Line Protocol

Universal Fault-Tolerant Constructors

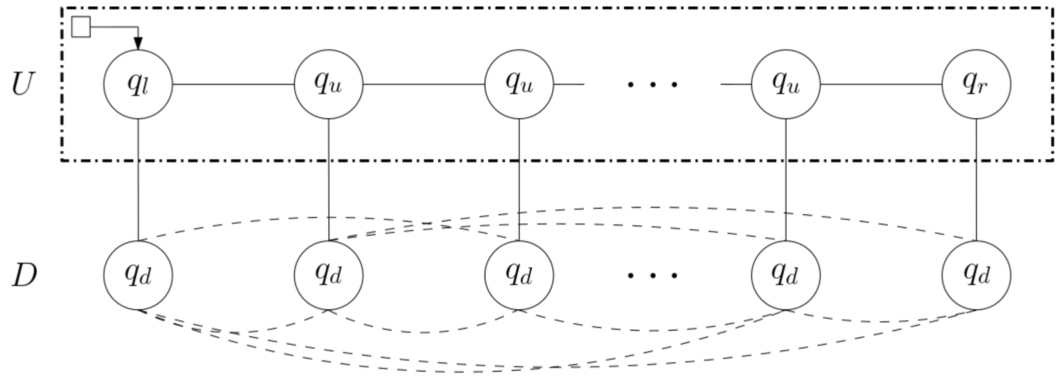
- Is there a generic fault-tolerant constructor capable of constructing a large class of graphs?
- The Fault-Tolerant Spanning Line is capable of simulating a given Turing Machine of space $O(n-k)$, where $0 \leq k < n$ is the number of faults
- We provide a fault-tolerant protocol that splits the population into two groups U and D of equal size
  - U is a spanning line with a unique leader in one endpoint and can eventually simulate a TM
  - Each node of D is connected with exactly one node of U, and vice versa



2-Partition

Universal Fault-Tolerant Constructors

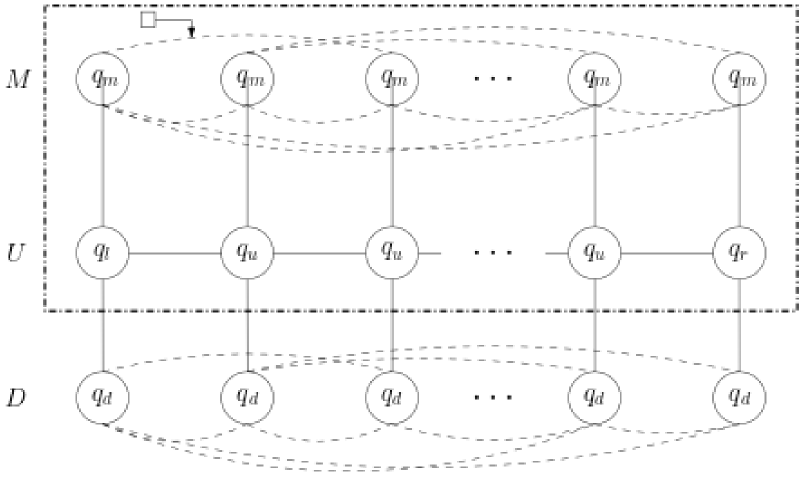
- This protocol *(Partition)* is fault-tolerant, but adds a waste of $2f(n)$, where $f(n)$ is an upper bound on the number of faults.
- We show that for any graph language L that can be decided by a *linear space* TM, there is a protocol that constructs a graph from L in D with waste at most $\min\{\frac{n}{2} + f(n), n\}$.



2-Partition

Universal Fault-Tolerant Constructors

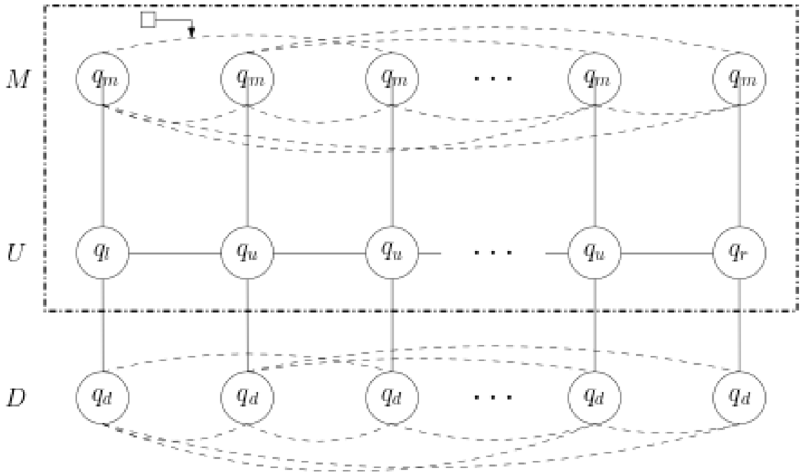
- This idea can be extended in order to increase the memory of the TM, by partitioning the population into three groups U, D and M of equal size.
- We provide a fault-tolerant protocol where
  - U is a spanning line that can eventually simulate a TM
  - Each node in $\mathrm{D} \cup M$ is connected with exactly one node of U
  - Each node of U is connected to exactly one node in D and one node in M.



3-Partition

Universal Fault-Tolerant Constructors

- This protocol is fault-tolerant, but adds a waste of $3f(n)$, where $f(n)$ is an upper bound on the number of faults.
- We show that for any graph language L that can be decided by an $O(n^2)$-*space* TM, there is a protocol that constructs a graph from L in D with waste at most $\min\{\frac{2n}{3} + f(n), n\}$.



3-Partition

Our Results

Constructible languages		
Without notifications		With notifications
Unbounded faults	Bounded faults	Unbounded faults
Only Spanning Clique	Non-hereditary impossibility	Fault-tolerant protocols: Spanning Star, Cycle Cover, Spanning Line
Strong impossibility even with linear waste	A representation of any finite graph (partial constructibility)	Universal Fault-tolerant Constructors (with waste)
	Any constructible graph language with linear waste	Universal Fault-tolerant Restart (without waste)

Fault-Tolerant protocols without Waste

- We increase the memory of each node to $O(logn)$ bits
  - We show that for constant memory, if the nodes can form a function of n connections with other nodes, it is impossible to restart the protocol correctly
- Each node stores two components C_1 and C_2
  - C_1 runs the restart protocol (*leader, phase, fault-flag)*
  - C_2 runs the given PP or NET protocol
- Whenever the fault-flag of a node is raised, all nodes eventually reinitialize their states in C_2
- After any re-initialization, *phase* is increased by one
- Nodes in different phases do not update their C_2 components
- We provide a protocol which guarantees that every node which enters to a new phase, has re-initialized its state correctly (all adjacent edges become inactive)

Future Work

- Are hereditary graph languages constructible if a bounded number of faults is allowed?
- Can we drop the assumption of waste and coin tossing?
- Consider other types of faults such as random, Byzantine, communication/edge faults
- Examination of fault-tolerant protocols for stable dynamic networks in models stronger than NETs.

UNIVERSITY OF LIVERPOOL

Thank You