A random walk based clustering with local re-computations for mobile ad hoc networks

Kudireti ABDURUSUL ⋆
Alain BUI†
Devan SOHIER†

⋆SysCom, CReSTIC
Université de Reims Champagne-Ardenne, France
†PRiSM (UMR CNRS 8144)
Université de Versailles St-Quentin-en-Yvelines

Plan

Context and Related Works

MANETs

- Decentralized wireless networks.
- Arbitrary Movement .

Motivation

- Large computer networks : dividing them into several disjoint connected parts.
- Managed separately and be coordinated.

Context and Related Works

Related Works

- *1-hop* : each node in the network is the neighbor of its *clusterhead*. (*eg.* LCA (LCA2), DMAC, GDMAC).
- *K-hop* : any node in any cluster is at most k hops away from its *clusterhead* (*eg. Max-min D-hop-cluster*, hierarchical clustering)

Model and hypotheses

- **Model** : an asynchronous message-passing model
- **Hypotheses**
  - Connected network
  - Unique identifier
  - Link bidirectional
  - Detection of Link failure

Context and Related Works

Random walk based distributed algorithm

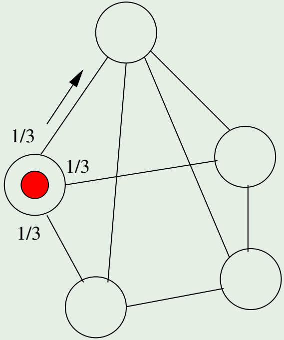
An algorithm involving a particular message, the *token*, that circulates according to a random walk scheme

- At each step, a node possesses the token
- Transmission: choose one neighbor at random, and send the token to it

Properties of random walks:

- Hitting
- Meeting

Random walks scheme



Context and Related Works

Random walk based distributed algorithm

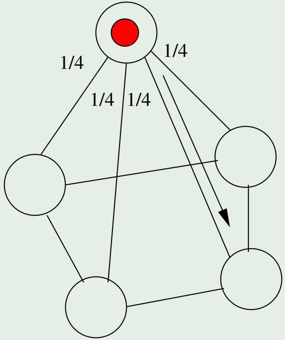
An algorithm involving a particular message, the *token*, that circulates according to a random walk scheme

- At each step, a node possesses the token
- Transmission: choose one neighbor at random, and send the token to it

Properties of random walks:

- Hitting
- Meeting

Random walks scheme



Context and Related Works

Random walk based distributed algorithm

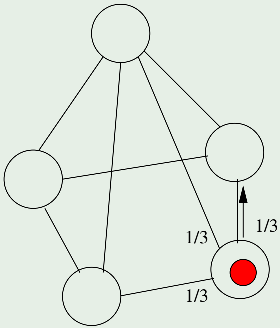
An algorithm involving a particular message, the *token*, that circulates according to a random walk scheme

- At each step, a node possesses the token
- Transmission: choose one neighbor at random, and send the token to it

Properties of random walks:

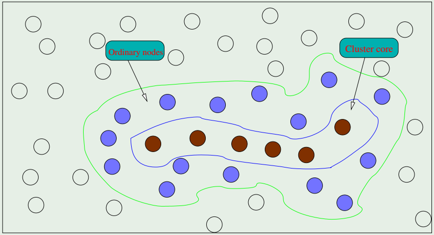
- Hitting
- Meeting

Random walks scheme



Principle

Our cluster

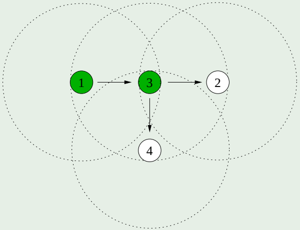


A distributed clustering algorithm based on random walks

- Core Construction
- Core neighbors
- MaxCoreSize
- Complete cluster and Incomplete cluster

Algorithm and Messages

Token message



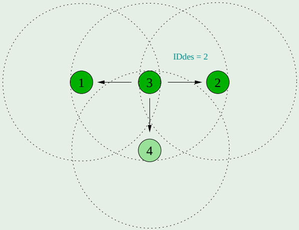
On reception of *Token* message

- Join procedure
- Transmit the token
- Send the Token back

Delete message

Algorithm and Messages

Token message



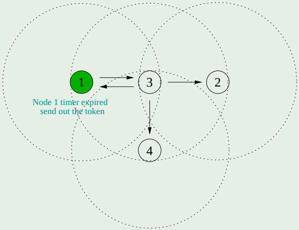
On reception of *Token* message

- Join procedure
- Transmit the token
- Send the Token back

Delete message

Algorithm and Messages

Token message



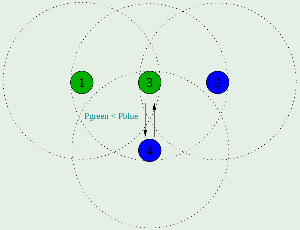
On reception of *Token* message

- Join procedure
- Transmit the token
- Send the Token back

Delete message

Algorithm and Messages

Token message



On reception of *Token* message

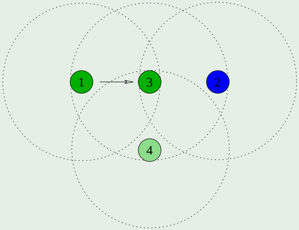
- Join procedure
- Transmit the token
- Send the Token back

Delete message

Algorithm and Messages

Token message

Delete message



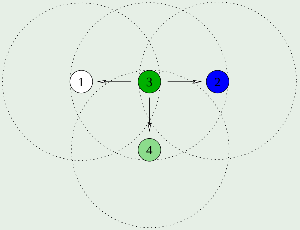
On reception of *Delete* message

if $P_r = P_e$
then Broadcast $Delete(P_r, O_r)$ message, $isCore = false$, reset timer

Algorithm and Messages

Token message

Delete message

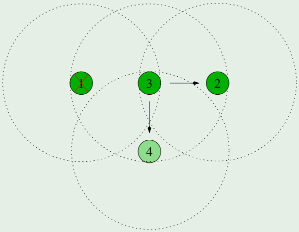


On reception of *Delete* message

if $P_r = P_e$
then Broadcast $Delete(P_r, O_r)$ message, $isCore = false$, reset timer

Algorithm and Messages

Link failure



On detecting a link $(i, j) \in E$ failure on node i

if $(i, j) \in Core \wedge isCore_i = true$ **then**

Delete procedure

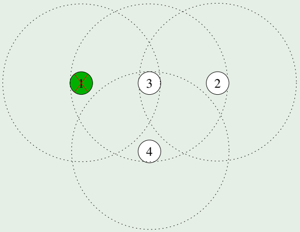
re-initialization

if $isCore_i = false$

re-initialization

Algorithm and Messages

Link failure



On detecting a link $(i, j) \in E$ failure on node i

if $(i, j) \in Core \wedge isCore_i = true$ **then**

Delete procedure

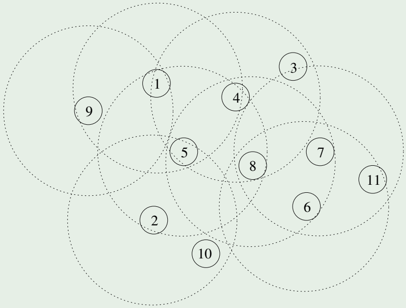
re-initialization

if $isCore_i = false$

re-initialization

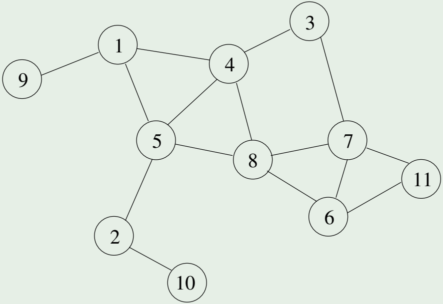
Example execution of the algorithm

Ad-hoc network with 11 nodes



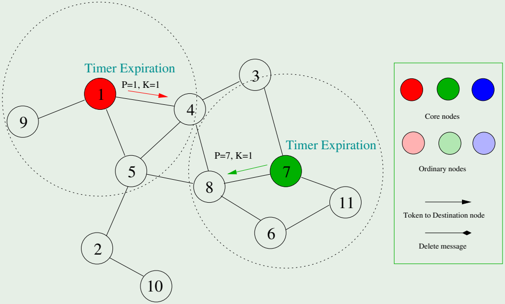
Example execution of the algorithm

The modeling of this ad hoc network of ($MaxCoreSize = 3$)



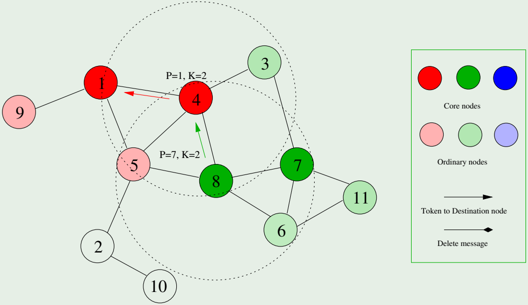
Example execution of the algorithm

Second state



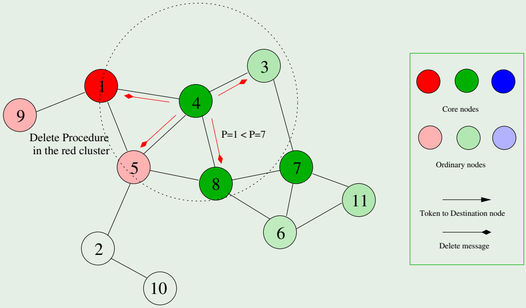
Example execution of the algorithm

Second state



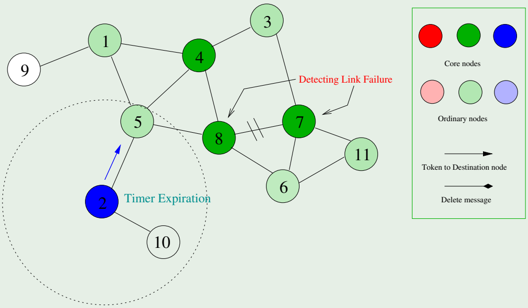
Example execution of the algorithm

Third state



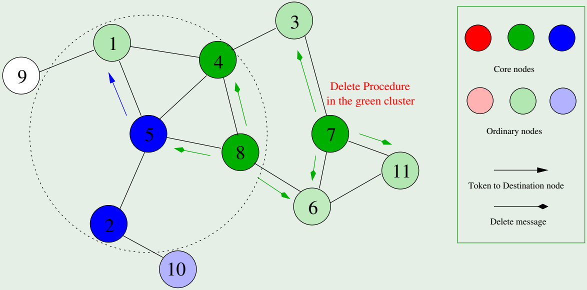
Example execution of the algorithm

Fourth state



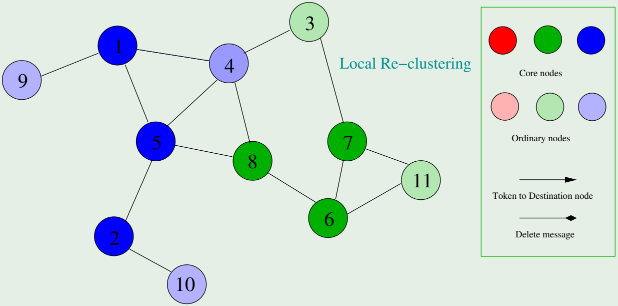
Example execution of the algorithm

Fifth state



Example execution of the algorithm

Steady state



Result

2 clusters with $MaxCoreSize = 3$

Properties

Convergence

- The clusters will eventually stabilize.

Correctness

- Each node eventually belongs to a cluster.
- The cluster is connected in the steady state.

Important properties

Properties

Convergence

Correctness

Important properties

- The core size of any cluster in [2,MaxCoreSize].
- Two adjacent clusters can not both be incomplete in the steady state.
- An incomplete cluster contain only core nodes in the steady state.

Properties

Local re-clustering

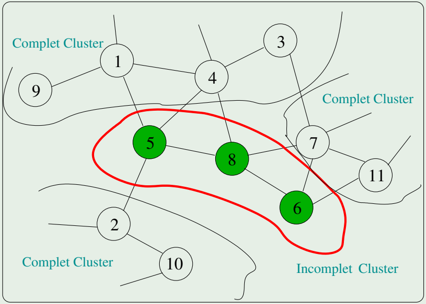
- Allows the scalability of algorithm
- Ensure that the bounded number of nodes have to recompute their cluster
- Avoid the "chain reaction"

Properties

Local re-clustering

- Using 2 important properties
- Effect the deleted cluster
- In the worst case, effect the adjacent clusters.

Different cases

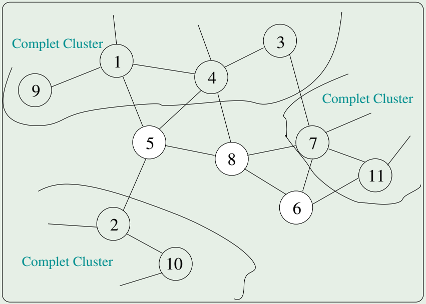


Properties

Local re-clustering

- Using 2 important properties
- Effect the deleted cluster
- In the worst case, effect the adjacent clusters.

Different cases

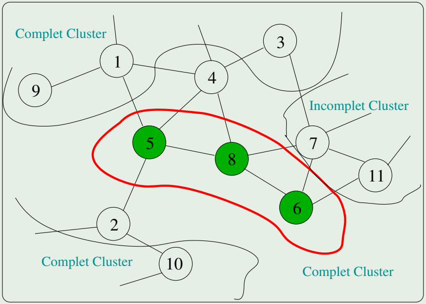


Properties

Local re-clustering

- Using 2 important properties
- Effect the deleted cluster
- In the worst case, effect the adjacent clusters.

Different cases

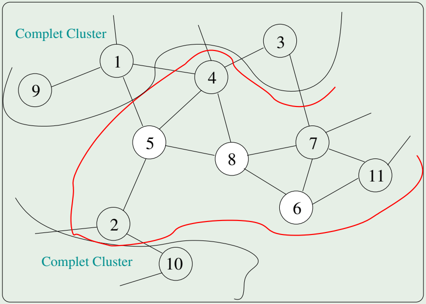


Properties

Local re-clustering

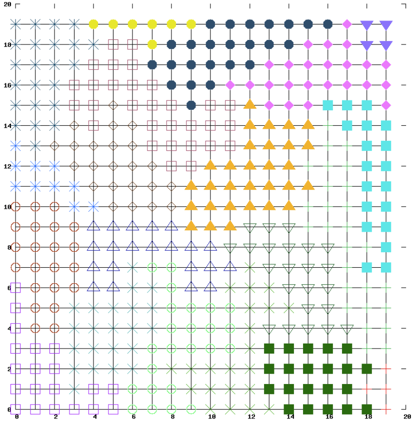
- Using 2 important properties
- Effect the deleted cluster
- In the worst case, effect the adjacent clusters.

Different cases



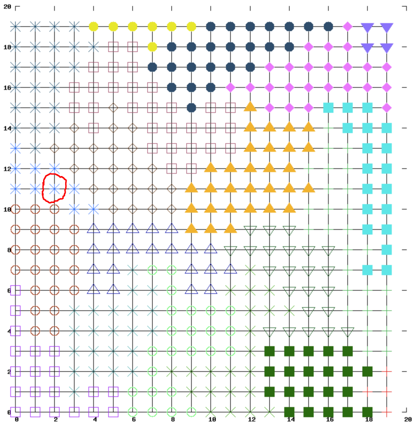
Simulation

- DASOR: a C++ library for discrete event simulation of distributed algorithms
- "Romeo": the high performance computing center of the University of Reims Champagne-Ardenne
- Simulation steps :
  - simulate without any connection or disconnection of nodes
  - starting from the configuration results, adding a node crash-and-restart



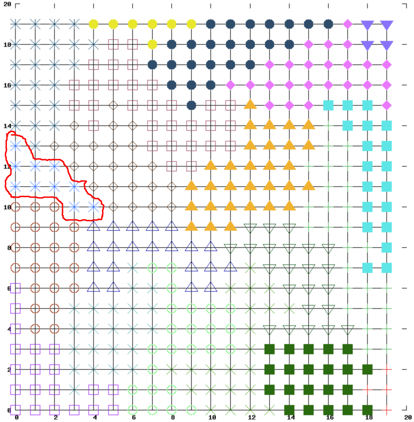
Simulation

- DASOR: a C++ library for discrete event simulation of distributed algorithms
- "Romeo": the high performance computing center of the University of Reims Champagne-Ardenne
- Simulation steps :
  - simulate without any connection or disconnection of nodes
  - starting from the configuration results, adding a node crash-and-restart



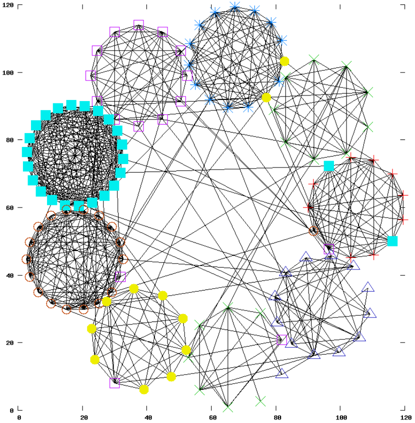
Simulation

- DASOR: a C++ library for discrete event simulation of distributed algorithms
- "Romeo": the high performance computing center of the University of Reims Champagne-Ardenne
- Simulation steps :
  - simulate without any connection or disconnection of nodes
  - starting from the configuration results, adding a node crash-and-restart

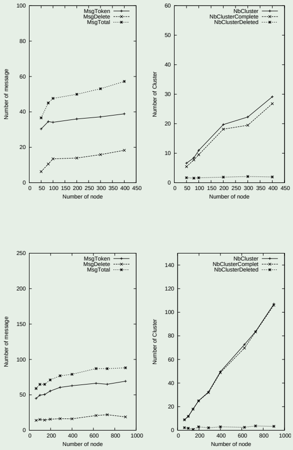


Simulation results analyse

- Cluster sizes
  - few incomplete clusters: 2.56% for *Random graph*, 0.82% for *caveman* graphs.
- few deletion.
- Message complexity increases with the parameter *MaxCoreSize*
- CaveMan graph:
  - $NBclusters = NBcaves$.
  - $98.2\% - 99.5\%$ nodes in each cave belong to one cluster.



Experiment results of re-clustering (managing link failure)

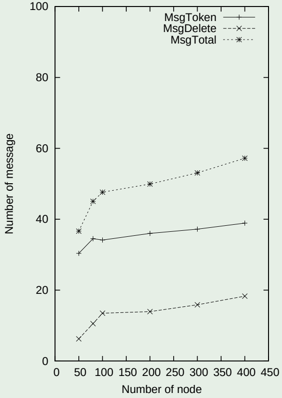


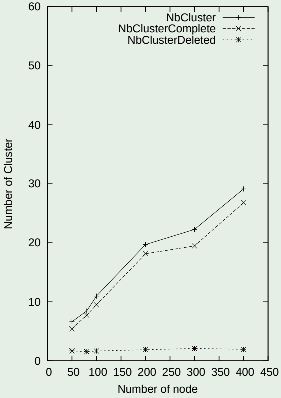
Re-clustering

- re-clustering is much faster than the initial clustering
- number of message grows slowly with the size of the network
- re-clustering takes a bounded (average) number of messages

Experiment results of re-clustering (managing link failure)

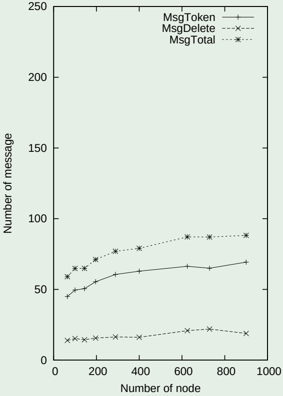
Random graph MaxCoreSize = 6

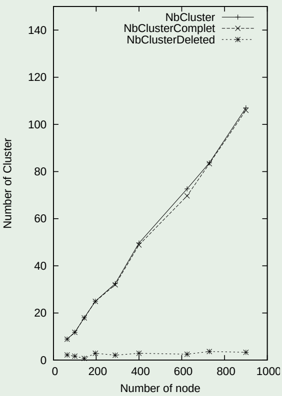




Experiment results of re-clustering (managing link failure)

Grid graph MaxCoreSize = 6





Conclusion and perspectives

Conclusion

- Original algorithm based on random walks
- Requires no assumption on the network topology
- Local re-clustering "Mobility adaptive"
- Simulation of performance of algorithm

Perspectives

- Improvement of adaptability
- Inter cluster management
- Self-stabilization

Questions?

Thanks for your attention !