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
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
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
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
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

On reception of *Token* message:
- Join procedure
- Transmit the token
- Send the Token back

*Token message*

*Delete message*
Algorithm and Messages

*Token message*

1. Node 1 timer expired
2. Send out the token

At 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

\[
\text{if } P_r = P_e \\
\text{then Broadcast } Delete(P_r, O_r) \\
\text{message, } isCore = false, \text{ reset timer}
\]
Algorithm and Messages

**Token message**

**Delete message**

On reception of *Delete* message

\[
\begin{align*}
&\text{if } P_r = P_e \\
&\text{then Broadcast } \text{Delete}(P_r, O_r) \\
&\text{message, } isCore = \text{false}, \text{ reset timer}
\end{align*}
\]
Algorithm and Messages

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

\[
\begin{align*}
\text{if } (i, j) \in \text{Core} \land \text{isCore}_i &= \text{true} \\
\text{then Delete procedure}\ \\
\text{re-initialization}
\end{align*}
\]

\[
\begin{align*}
\text{if } \text{isCore}_i &= \text{false} \\
\text{re-initialization}
\end{align*}
\]
Algorithm and Messages

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

\[
\begin{align*}
\text{if } (i, j) &\in \text{Core} \land \text{isCore}_i = true \\
\text{then} & \\
\text{Delete procedure} & \\
\text{re-initialization} & \\
\text{if } \text{isCore}_i = false & \\
\text{re-initialization} &
\end{align*}
\]
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

[Diagram showing a network with nodes and links, indicating timer expiration and token to destination node, with labels for core nodes and ordinary nodes.]
Example execution of the algorithm

Second state

- Node 4: P=1, K=2
- Node 5: P=7, K=2
- 'Token to Destination node'
- 'Delete message'

Nodes classification:
- Core nodes: Red
- Ordinary nodes: Green
- Node 1 and 3 are not indicated.
Example execution of the algorithm

Third state

Delete Procedure in the red cluster

P=1 < P=7

Core nodes

Ordinary nodes

Token to Destination node

Delete message
Example execution of the algorithm

Fourth state

- Core nodes
- Ordinary nodes
- Token to Destination node
- Delete message

Timer Expiration
Detecting Link Failure
Token to Destination node
Delete message
Example execution of the algorithm

Fifth state
Example execution of the algorithm

Steady state

![Graph showing a network with nodes and edges indicating core nodes and ordinary nodes.]

Local Re-clustering

Delete message

Result

2 clusters with $\text{MaxCoreSize} = 3$
**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 $[2, \text{MaxCoreSize}]$.
- Two adjacent clusters cannot both be incomplete in the steady state.
- An incomplete cluster contains 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.
Properties

Local re-clustering

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

Different cases

```
1 4
2 5
3
6
7
8
9
10
11
```

Complet Cluster

Kudireti Abdurusul (CReSTIC)

RWCMA
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.
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

![Graph showing experiment results]

- **MsgToken**
- **MsgDelete**
- **MsgTotal**

![Graph showing the relationship between number of nodes and number of messages]

- **NbCluster**
- **NbClusterComplete**
- **NbClusterDeleted**

![Graph showing the relationship between number of nodes and number of clusters]

Kudireti Abdurusul (CReSTIC)
Grid graph MaxCoreSize = 6

Experiment results of re-clustering (managing link failure)
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!