PER : « *Best-effort Group Service in Dynamic Networks* »

- Presented by : Adrien Prouillac, Bilal Amrouche, Mohammed Yessin Neggaz
- Supervisors : Colette Johnen, Serge Chaumette, Arnaud Casteigts
- Paper :

Mobile Ad-Hoc Network

- **Introduction**
  - Spontaneous network
  - Self-configuring
  - Without any infrastructure or central control

- **Characteristics**
  - Dynamic topology
  - Limited capacity (CPU, Storage, Battery power and BandWidth)
  - Radio environment (interference, Unidirectional links)
Mobile Ad-Hoc Network

• Usage areas
  - Military scenarios
  - Rescue operations
  - VANET
  - Sensors networks

Sensor network
http://www.ece.ncsu.edu/wireless/Images/sensor.gif
Objectives

- Organize various mobile devices:
  - Network segmentation in different groups
  - Group maintenance
  - A node can belong to only one group
  - A node have an entire view of its group
- Avoid overloading communication links
- Try to ensure the robustness against network topology changes
- Provide a self-stabilizing solution
Dynamic group approach

- Set of nodes
  - Exchange information
  - Run application

- Dynamic
  - High device mobility
  - Devices frequently leave and join the group
Protocol property

- **Agreement property**: All nodes must agree to add new node in their group
- **Safety property**: Resulting group diameter should not exceed Dmax
- **Maximality property**: Each group must merge with its neighbor groups to reach the maximum diameter
- **Self-Stabilization**
  - We call a self-stabilizing algorithm, the algorithm that, from any state, reaches the expected result and a stable state in a finite time
Best-Effort

• **Continuity**
  - Ensure that the quality of the output increases between successive configurations

• **Best-effort**
  - Ensure the continuity if no “important topology changes” occur
Problems

- Integration of a new node (quarantine period)
- Expulsion of a node (integration priority)
- Wireless communication: asymmetric links between two nodes
GRP Algorithm
GRP Algorithm

- **Links**
  - Communication

- **Node**
  - **Set of received messages,** \( \text{msgSet} \)
  - **List of sets of nodes being processed,** \( \text{list} \)
    - Sets are ordered by distance between nodes
      - \( \text{List}_1 = (\{1\}, \{2,3\}, \{4,5\}) \)
      - \( \text{List}_6 = (\{6\}) \) (stable state)
  - **Set of group nodes,** \( \text{view} \)
    - **Nodes in the same group**
      - \( \text{View}_1 = \{1,2,3,4,5\} \)
      - \( \text{View}_6 = \{6\} \)
Progress

- Timer Tc (t1)
- Timer Ts (t2)

Sending list (broadcast)
Reception (update msgSet)

Tc expiration
Compute

Ts expiration

Bilal AMROUCHE  Adrien PROUILLAC  Mohammed Yessin NEGGAZ
Compute

- $msgSet = \{list_1, list_2, list_3, \ldots\}$
- Each list is processed (checked)
  - Result:
    - Merging with $list$
      - The sender and receiver will be part of the same group
    - Rejection
      - The sender is ignored

Update $list$ and $view$
Symmetry

processing

\( list_u \)

\( list_v \) after processing

\( list_v \) with marked \( v \) \( (\n) \)

\( list_v \)

\( v \) new!
Self-stabilization

- GRP algorithm is self-stabilizing
  - Message passing
  - $r$-operator ant($list1$, $list2$)
    - Calculate the output list
    - Ensure the self-stabilization
  - Reach the final list in a finite time from any state
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization
Self-stabilization

Diagram: A network of nodes labeled 1, 2, 3, 4, 5, and 6, showing connections between them.
Self-stabilization

List ({2},{1,3,5},{4}) View {1,2,3,4,5}
List ({1},{2},{3,5},{4}) View {1,2,3,4,5}
List ({4},{3,1},{2},{1,5}) View {1,2,3,4,5}
List ({5},{2},{1,3},{4}) View {1,2,3,4,5}
List ({2},{1,3,5},{4}) View {1,2,3,4,5}
List ({3},{2,4},{1,5}) View {1,2,3,4,5}
List ({6}) View {6}
D_{\text{max}} = 6
D_{\text{max}} = 6

Merging

Bilal AMROUCHE
Adrien PROUILLAC
Mohammed Yessin NEGGAZ
D_{max} = 6

Merging

({1}, {2,4}, {3})
Dmax = 6

Merging

({1},{2,4},{3})

({5},{1,6,7},{8})
Dmax = 6

Merging

Check list

→ Accept

Dmax = 6

Merging

Bilal AMROUCHE    Adrien PROUILLAC    Mohammed Yessin NEGGAZ
Dmax = 6

Merging

Check list\textsubscript{5} → Accept

\begin{align*}
\text{CompatibleList} & = \{(1),\{2,4\},\{3\}\} \\
\text{resulting group diameter} & < D_{\text{max}} \Rightarrow \text{Accept}
\end{align*}

Bilal AMROUCHE Adrien PROUILLAC Mohammed Yessin NEGGAZ
Dmax = 6

Merging

Check list_5 → Accept

list_1 = ant(list_1, list_5)
= (\{1\}, \{5, 2, 4\}, \{3, 6, 7\}, \{8\})
Dmax = 6

Check list₅ → Accept

list₁ = ant(list₁, list₅)

= \{1\}, \{5, 2, 4\}, \{3, 6, 7\}, \{8\}
Dmax = 6

Merging

Check list_5 → Accept

\[
\{1\},\{2,4\},\{3\},\{5\},\{1,6,7\},\{8\},\{1\},\{5,2,4\},\{3,6,7\},\{8\},\{5\},\{1,6,7\},\{8,4,2\},\{3\}
\]

list_1 = \text{ant}(list_1, list_5) = \{1\},\{5,2,4\},\{3,6,7\},\{8\}

Bilal AMROUCHE    Adrien PROUILLAC    Mohammed Yessin NEGGAZ
D_{max} = 6

CompatibleList ??

resulting group diameter ? 5 < D_{max} Accept

D_{max} = 6

list1 = \text{ant}(list_1, list_5) = \{(1), \{5, 2, 4\}, \{3, 6, 7\}, \{8\})

Check list_5 → Accept

Update view view = \{1, 2, 3, 4, 5, 6, 7, 8\}

Merging

list1 = \text{ant}(list_1, list_5) = \{(1), \{5, 2, 4\}, \{3, 6, 7\}, \{8\})

Bilal AMROUCHE
Adrien PROUILLAC
Mohammed Yessin NEGGAZ
Safety

• Received lists are checked
  − Condition: The diameter of the resulting group ?
  − GRP uses mainly CompatibleList() test

  • Diameter of the resulting group < Dmax → accept → merge

  • Diameter of the resulting group > Dmax → ignore → reject
Dmax = 6

CompatibleList ??
resulting group diameter ?
7 > Dmax
Reject
Ignore 5

Bilal AMROUCHE    Adrien PROUILLAC    Mohammed Yessin NEGGAZ
Accepting two groups

Resulting of a group with diameter > Dmax

Nodes having lists with size = Dmax+2

6 has the priority
4 leaves the group by ignoring 2

Time

Dmax = 4
Agreement

- Accepted nodes should be approved by all the nodes of the group
  - Quarantine period
    - New accepted nodes → Quarantine period = Dmax
    - Quarantine period of existing nodes decreases by 1 every $T_c$ expiration (compute)
    - Nodes with null quarantine period are added to view
Doubts

- CompatibleList (Maximality)
- Separation problem
- Distributed requests
Doubts

- Propagation (separation problem)
Propagating (separation problem)

- Proposed solution
  - `existenceCounter`
  - A node takes the minimum value of the `existenceCounter` of the received node
  - `existenceCounter` of nodes added to the list is incremented by 1
  - If the `existenceCounter` value > `(Dmax * 2) - 1`
    → the node is not added to the list
Problem determination

• High dynamic in the network
  - Concurrent acceptance with high frequency
    → Split of groups
    → Nodes leave their groups
  - Connection and disconnection of nodes when the duration of convergence is elevated
    → No continuity

• The best-effort requirement says that outside of these conditions the algorithm should ensure the continuity and self-stabilizing properties
Evaluation
Implementation

- GNU/linux
- Eclipse IDE
- JBotSim Simulator
Simulation parameters

- Experimental field: the area in which the nodes can move
- Number of nodes: number of node deployed on the experimental field
- Dmax: Max diameter of a group
- Node speed: Distance traveled on time period
Metric of interest

- Stability: the percentage of time when the views are stable
  - Stability = (stableTime/totalTime) * 100
Tests

- Random WayPoint
- Vanet Model
- Group Model
- Student Model
Parameters

- Field = 800*800 distance unit
- Nodes speed = 5 distance unit per X units of time
- Time = 3000 units of time
- X=Speed factor → variable
- Number of node → variable
Random waypoint test
Vanet Model test

Stability

Speed of nodes

Nodes number
Conclusion

- Discussion about GRP
  - Offer a solution without a central node
  - It seems to be more relevant to those ad-hoc networks with a little number of events and relative stable periods

- Additional work
  - Contribute to improve the performance based on the results achieved by our study of the algorithm
  - Design and implement a centralized optimal algorithm for this problem and compare its stability to that of GRP