Self-Stabilization and MANET

Franck Petit

UPMC
Firstname.Lastname@lip6.fr
MANET

- Network
  - computing nodes
  - communication links
  - topology $\sim$ graph $G(V, E)$

- Ad hoc network
  - uniform nodes: all equivalent, all routers

- Wireless network
  - mutual exclusion in the neighborhood:
    a message sent by a neighbor is received only if no other neighbor is sending
  - message passing or register model

- Wireless ad hoc networks
Dynamic MANET

- Dynamic network: the topology is not fixed
  \[ G_0(V_0, E_0), G_1(V_1, E_1), G_2(V_2, E_2), G_3(V_3, E_3) \ldots \]
  1. Dynamic links
     - adding links
     - deleting links
  2. Dynamic nodes
     - adding nodes
     - deleting nodes
     - \( \sim \) adds and deletes links
  3. Moving nodes
     - \( \not= \) deleting node + adding node elsewhere
       memory of the node
     - \( \sim \) adds and deletes links temporarily
  4. Dynamic and moving nodes...
Dynamic MANET

Dynamic ad hoc networks

with infrastructure
- wired network
- mobile terminal
- mobile network
- ad hoc networks
- hand-over...
- routers, fixed servers
- Internet, IP

without infrastructure
- virtual structures management (tree...)
- mobile ad hoc networks
- Cellular
- MobileIP
- MANET
- VANET
Metrics for Dynamic

- Percentage of nodes or links affected
- Mean percentage of a neighborhood affected
- Frequency of changes Unit of time?
- Frequency of changes vs. efficiency of the com.
  Nodes could move very fast without impact on the algorithm if the communication protocol is efficient.

Algorithmic metric
  - $\delta$-dynamic system: any node that experiments a neighborhood change is able to send a message to all its neighbors until $\delta$ hops before the next topology change
  - 1-dynamic system: a minimal requirement for allowing local exchange in a dynamic network
Examples of Dynamic MANET

- Large networks are generally dynamic
- Social networks
- Peer-to-peer networks
- Network of laptops
  IEEE working group MANET: Mobile Ad hoc NETworks
- Network of pedestrian with personal devices
- Network of embedded computers
  - Robots networks
  - Vehicular networks
Dynamic networks

Best-effort algorithms

Group service maintenance

Conclusion

VANET

- **Intelligent Transport Systems (ITS)**
  - **infrastructure oriented applications**
    for optimizing the infrastructure management (transit, freeway, freight, emergency organization...)
  - **vehicle oriented applications**
    for increasing the road safety
    (crash prevention, alerts, visibility distance...)
  - **driver oriented applications**
    for improving the road usage
    (traffic jam and road work information, traveler payment, ride duration estimate...)
  - **passenger oriented applications**
    for offering new services on board
    (Internet access, distributed games, chat, touristic information...)
VANET

- Intelligent Transport Systems (ITS)
- Example of scenario

from Car 2 Car Communication Consortium
VANET

- Intelligent Transport Systems (ITS)
- Example of scenario
VANET

- Intelligent Transport Systems (ITS)
- Example of scenario
VANET

- Intelligent Transport Systems (ITS)
- Example of scenario
VANET

- **Intelligent Transport Systems (ITS)**
- **Example of scenario**
Coping Dynamic
Global Data Structure

- Backbones, spanning trees, clusters...
  - using such structures as in non-dynamic networks
  - updating the structures when the topology changes
Coping Dynamic
Global Data Structure

- Backbones, spanning trees, clusters...
  - using such structures as in non-dynamic networks
  - updating the structures when the topology changes

- But:
  - require control messages to be updated
  - when the dynamic increases, too much control messages are required
  - diverge

- Thus:
  - useful only when the dynamic is low
Coping Dynamic Redundancy

- **Important data are replicated**
  - critical system
  - A node’s disappearance is then supported
Coping Dynamic Redundancy

- Important data are replicated
  - critical system
  - A node’s disappearance is then supported

- But:
  - replicated data should be coherent
  - pessimistic replication requires consensus
    - consensus is unsolvable in unreliable asynchronous networks [FLP85]
    - alternative: failure detectors [CT96]
  - optimistic replication will eventually converge
    - working with non up-to-date data

- Thus:
  - strong conditions on the network
  - or weak conditions on the replicas
Coping with Dynamic Self-Stabilization

- Self-stabilizing algorithms:
  - recover after a transient fault affecting a memory, a message
  - neighborhood change ➔ some memories are not up-to-date
  - topology change ⇔ transient fault
Coping with Dynamic Self-Stabilization

▶ Self-stabilizing algorithms:
  ▶ recover after a transient fault affecting a memory, a message
  ▶ neighborhood change
    ⟷ some memories are not up-to-date
  ▶ topology change ↔ transient fault

▶ But:
  ▶ duration of the convergence phase vs. dynamic
  ▶ the system doesn’t know whether the stabilized phase is reached or not

▶ Thus:
  ▶ useful only when the dynamic is low
  ▶ and for non critical applications
Best-effort algorithms

Principle

The dynamic affects the algorithms
When the dynamic increases, it becomes illusory to expect that an application continuously ensures the service for which it has been designed.

- impossibility results?
- weak specifications?
- conditions on the dynamic

What we can only expect from the distributed algorithms is to behave as "the best" as possible, the result depending on the dynamic.

A best effort algorithm fulfills its specifications if the dynamic of the network allows it, and fulfills them few time after the network allows it, otherwise.
Best-effort algorithms
Using self-stabilization?

▶ Self-stabilization can help face to the dynamic Neighborhood change
   ~ some memories do not reflect the neighborhood
   ~ similar to a transient failure
Best-effort algorithms

Using self-stabilization?

- Self-stabilization can help face to the dynamic neighborhood change
  - some memories do not reflect the neighborhood
  - similar to a transient failure

- However, it is implicitly assumed that the convergence time is smaller than the delay between two topology changes
Best-effort algorithms

Using self-stabilization?

- Self-stabilization can help face to the dynamic neighborhood change
  - some memories do not reflect the neighborhood
  - similar to a transient failure

- However, it is implicitly assumed that the convergence time is smaller than the delay between two topology changes

- General case:
  self-stabilization property must be completed
Best-effort algorithms

Definition

- Continuity predicate $\Pi_C$ on successive config.: False if the “quality of successive outputs” decreases depends on the problem

- Topological predicate $\Pi_T$ on successive config.: False if the topological change is “important” depends on the problem

- Best-effort requirement: $\Pi_T \Rightarrow \Pi_C$
  
  While the system is converging to a correct behavior, the result is better and better, as long as the dynamic allows it.
Best-effort algorithms

Enhancing stabilization for dynamic systems

- How to complete self-stabilization?
  - Fault-containing network protocols
    Gosh, Gupta, Pemmaraju, SAC ’97
  - Stabilizing time adaptive protocols
  - Superstabilizing protocols for dynamic distributed systems
    Dolev, Herman. PODC 1995
Best-effort algorithms

Enhancing stabilization for dynamic systems

- How to complete self-stabilization?
  - Fault-containing network protocols
    Gosh, Gupta, Pemmaraju, SAC ’97
  - Stabilizing time adaptive protocols
  - Superstabilizing protocols for dynamic distributed systems
    Dolev, Herman. PODC 1995

- Superstabilization
  - self-stabilization + passage predicate
  - after a legitimate state is reached, if a single topology change occurs, the predicate passage holds until a new legitimate state is reached
Best-effort algorithms

Enhancing stabilization for dynamic systems

- How to complete self-stabilization?
  - Fault-containing network protocols
    Gosh, Gupta, Pemmaraju, SAC ’97
  - Stabilizing time adaptive protocols
  - Superstabilizing protocols for dynamic distributed systems
    Dolev, Herman. PODC 1995

- Superstabilization
  - self-stabilization + passage predicate
  - after a legitimate state is reached, if a single topology change occurs, the predicate passage holds until a new legitimate state is reached
  - but:
    - what before stabilization?
    - important in a dynamic system
Groups service

Requirements: groups in vehicular networks

- Intelligent Transport Systems
  - infrastructure oriented applications
  - vehicle oriented applications
  - driver oriented applications
  - passenger oriented applications

- Some services are based on collaboration
  - driving, diagnostic, perception, infotainment...
  - collaboration $\leadsto$ group

- Vehicular networks: a kind of dynamic networks
Groups service

Requirements: constraints on the groups

- Maintaining the running service
  - the aim is not to optimize the partition of the vehicles into groups
  - it is much more important to not split existing groups
  - keeping the existing groups as long as possible

- Diameter constraint
  - delay vs. number of hops
  - no collaboration with far vehicles
    - either useless (driving, diagnostic, perception...)
    - or inefficient (chat, games...)
  - bound on the diameter depending on the applications
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Example
Groups service for inter-vehicles applications

Specifications

- **Groups**: disjoint subgraphs of $G(V, E)$
  - subgraphs $H_i(V_i, E_i)$ with $V_i \cap V_j = \emptyset$
    - $V_i \subset V$ and $\forall (u, v) \in E, (u, v \in V_i) \Rightarrow (u, v) \in E_i$
  - $\text{view}_v^c$: knowledge of $v$ about its own group at configuration $c$

- **Agreement $\Pi_A(c)$**: views define groups
  - $(u \in V_i$ and $v \in V_i) \Leftrightarrow \text{view}_u^c = \text{view}_v^c = V_i$ $\Omega_v^c = \text{view}_v^c$ if $\Pi_A(c)$ holds, $\emptyset$ else

- **Safety $\Pi_S(c)$**: groups are well formed
  - connected and bounded diameter $(d_{\Omega_v^c}$: distance in $\Omega_v^c)$
    - $\forall v \in V, \max_{x, y \in \Omega_v^c} d_{\Omega_v^c}(x, y) \leq D_{\text{max}}$

- **Maximality $\Pi_M(c)$**: groups cannot merge more
  - $\forall u, v \in V$ with $\Omega_u^c \neq \Omega_v^c$
    - $\exists x, y \in \Omega_u^c \cup \Omega_v^c$ such that $d_{\Omega_u^c \cup \Omega_v^c}(x, y) > D_{\text{max}}$
Groups service for inter-vehicles applications

Best-effort specification

- **Topological predicate** $\Pi_T$:
  The distance between members of a group will remain smaller than $D_{\text{max}}$

\[ \forall v \in V, \max_{x,y \in \Omega_v} d_{\Omega_v}^{c_i+1}(x, y) \leq D_{\text{max}} \]
Groups service for inter-vehicles applications

Best-effort specification

- Topological predicate $\Pi_T$:
  
  The distance between members of a group will remain smaller than $D_{\text{max}}$

  \[ \forall v \in V, \max_{x,y \in \Omega^c_v} \quad d_{\Omega^c_v}^{c_i+1}(x, y) \leq D_{\text{max}} \]

- Continuity property $\Pi_C$:

  No node disappear from a group

  \[ \forall v \in V, \quad \Omega^c_v \subseteq \Omega^c_{v+1} \]
Groups service for inter-vehicles applications

Best-effort specification

- **Topological predicate \( \Pi_T \):**
  The distance between members of a group will remain smaller than \( D_{\text{max}} \)
  \[
  \forall v \in V, \max_{x,y \in \Omega_v^c} d_{\Omega_v^c}^{c_{i+1}}(x, y) \leq D_{\text{max}}
  \]

- **Continuity property \( \Pi_C \):**
  No node disappear from a group
  \[
  \forall v \in V, \Omega_v^c \subset \Omega_v^{c_{i+1}}
  \]

- **Best-effort specification: \( \Pi_T \Rightarrow \Pi_C \):**
  As long as the diameter of a group remains smaller than \( D_{\text{max}} \),
  the algorithm should ensure that no node will disappear
Groups service for inter-vehicles applications

Best-effort specification

- **Topological predicate** $\Pi_T$:
The distance between members of a group will remain smaller than $D_{max}$

$$\forall v \in V, \max_{x,y \in \Omega_v^{c_i}} d_{\Omega_v^{c_i}}^{c_i+1}(x,y) \leq D_{max}$$

- **Continuity property** $\Pi_C$:
No node disappear from a group

$$\forall v \in V, \Omega_v^{c_i} \subset \Omega_v^{c_i+1}$$

- **Best-effort specification**: $\Pi_T \Rightarrow \Pi_C$:
As long as the diameter of a group remains smaller than $D_{max}$, the algorithm should ensure that no node will disappear

- $\rightsquigarrow$ An application can work with the current knowledge of the group (view) even if the convergence did not happen
Groups service for inter-vehicles applications

Summary

- Self-stabilizing algorithm for $\Pi_A \land \Pi_S \land \Pi_M$
  - $\mathcal{C}$: set of all the configurations
  - $\mathcal{L} \subset \mathcal{C}$: set of configurations $c$ satisfying $\Pi_A(c) \land \Pi_S(c) \land \Pi_M(c)$
  - prove, on a fixed topology, that
    - $\mathcal{L}$ is an attractor for $\mathcal{C}$
    - $\mathcal{L}$ is close

- Best-effort requirement
  - assuming a dynamic network sequence of graphs
  - considering two consecutive configurations $c_i, c_{i+1}$ in any execution, prove that $\Pi_T(c_i, c_{i+1}) \Rightarrow \Pi_C(c_i, c_{i+1})$
Example: A Best-Effort Group Service Algorithm

Overview: diffusion of lists

- Candidates for a group: neighbors up to $D_{max}$
Example: A Best-Effort Group Service Algorithm

Overview: diffusion of lists

- Candidates for a group: neighbors up to Dmax
- Lists of close nodes:
  - diffusion at timer expiration
  - merging of the received lists
  - lists of nodes ordered by the distance: 
    \[\{d\}, \{b\}, \{a, c\}\]
  - lists truncated to Dmax
Example: A Best-Effort Group Service Algorithm

Overview: diffusion of lists

- Candidates for a group: neighbors up to $D_{\text{max}}$
- Lists of close nodes:
  - diffusion at timer expiration
  - merging of the received lists
  - lists of nodes ordered by the distance:
    - $\{d\}, \{b\}, \{a, c\}$
  - lists truncated to $D_{\text{max}}$
- Lists filtering:
  - only symmetric links
    - Three-way handshake by marking nodes: $v$
  - malformed lists ignored
  - arrival list accepted only if the diameter remains smaller than $D_{\text{max}}$ after the merge
  - if merging is impossible, the neighbor is double-marked ($v$)
Example: A Best-Effort Group Service Algorithm

Overview: merging lists

- $S$: set of lists of nodes’ sets $({d}, {b}, {a, c}) \in S$
Example: A Best-Effort Group Service Algorithm

Overview: merging lists

- $\mathcal{S}$: set of lists of nodes’ sets ($\{d\}, \{b\}, \{a, c\}) \in \mathcal{S}$

- Operator $\oplus$ on $\mathcal{S}$ that merges two lists while deleting useless members:
  $$(\{d\}, \{b\}, \{a, c\}) \oplus (\{c\}, \{a, e\}, \{b\}) = (\{d, c\}, \{b, a, e\})$$
Example: A Best-Effort Group Service Algorithm

Overview: merging lists

- $\mathcal{S}$: set of lists of nodes’ sets $(\{d\}, \{b\}, \{a, c\}) \in \mathcal{S}$
- Operator $\oplus$ on $\mathcal{S}$ that merges two lists while deleting useless members:
  $$(\{d\}, \{b\}, \{a, c\}) \oplus (\{c\}, \{a, e\}, \{b\}) = (\{d, c\}, \{b, a, e\})$$
- Endomorphism $r$ of $\mathcal{S}$, that inserts an empty set at the beginning of a list
  $$r(\{d\}, \{b\}, \{a, c\}) = (\emptyset, \{d\}, \{b\}, \{a, c\})$$
Example: A Best-Effort Group Service Algorithm

Overview: merging lists

- $S$: set of lists of nodes’ sets ($\{d\}$, $\{b\}$, $\{a, c\}$) $\in S$
- Operator $\oplus$ on $S$ that merges two lists while deleting useless members:
  $$(\{d\}, \{b\}, \{a, c\}) \oplus (\{c\}, \{a, e\}, \{b\}) = (\{d, c\}, \{b, a, e\})$$
- Endomorphism $r$ of $S$, that inserts an empty set at the beginning of a list
  $$r(\{d\}, \{b\}, \{a, c\}) = (\emptyset, \{d\}, \{b\}, \{a, c\})$$
- $l_1 \triangleleft l_2 = l_1 \oplus r(l_2)$, $l_1, l_2 \in S$ strictly idempotent $r$-operator
Example: A Best-Effort Group Service Algorithm

Overview: priorities

- **Conflict:**
  - deciding between far nodes in a too large list
- **Node priority:**
  - oldness of the node in its group
  - local logical clock increased until the node belongs to a group
- **Group priority:**
  - smallest priority of its members
  - Usefull to avoid loops of merging groups
- **Quarantine:**
  - waiting for $D_{\text{max}}$ timers before entering into the view
  - allowing to broadcast its identity in the whole group, and then resolve conflicts (using priorities)
Sketch of Proof

1. lists bounded
2. lists contain only existing nodes
Sketch of Proof

1. lists bounded
2. lists contain only existing nodes
3. propagation until double-marked edges
4. if \( d(u, v) > D\text{max} \), each path from \( u \) to \( v \) contains a double-marked edge
5. if \( d(u, v) > D\text{max} \), \( u \), \( v \) not in the same subgraph
6. Agreement: convergence to similar views inside each subgraph
   \( \rightsquigarrow \) groups
Sketch of Proof

1. lists bounded
2. lists contain only existing nodes
3. propagation until double-marked edges
4. if $d(u, v) > D_{\text{max}}$, each path from $u$ to $v$ contains a double-marked edge
5. if $d(u, v) > D_{\text{max}}$, $u$, $v$ not in the same subgraph
6. Agreement: convergence to similar views inside each subgraph
   $\sim$ groups
7. Safety: group’ diameters smaller than $D_{\text{max}}$
Sketch of Proof

1. lists bounded
2. lists contain only existing nodes
3. propagation until double-marked edges
4. if $d(u, v) > D_{\text{max}}$, each path from $u$ to $v$ contains a double-marked edge
5. if $d(u, v) > D_{\text{max}}$, $u$, $v$ not in the same subgraph
6. Agreement: convergence to similar views inside each subgraph
   $\rightsquigarrow$ groups
7. Safety: group’ diameters smaller than $D_{\text{max}}$
8. The number of external edges does not increase
9. The number of external edges decrease
10. Maximality: if new merge, safety is false
Sketch of Proof

1. lists bounded
2. lists contain only existing nodes
3. propagation until double-marked edges
4. if \( d(u, v) > D_{\text{max}} \), each path from \( u \) to \( v \) contains a double-marked edge
5. if \( d(u, v) > D_{\text{max}} \), \( u, v \) not in the same subgraph
6. Agreement: convergence to similar views inside each subgraph \( \sim \) groups
7. Safety: group’diameters smaller than \( D_{\text{max}} \)
8. The number of external edges does not increase
9. The number of external edges decrease
10. Maximality: if new merge, safety is false
11. Best-effort: a node leaves a group only if the safety becomes false
Experiences

Platform

- **Airplug**: Same tools for simulation, prototyping, implementation
Experiences

▶ On Road: airplug-road
▶ Six vehicles (with Orange R&D)
Experiences

- On Road: airplug-road
- In the lab: airplug-lab
  - Possibility to reuse GPS data recorded on road
  - New scenarios
Experiences

- On Road: airplug-road
- In the lab: airplug-lab
- Simulation: airplug-sim
  - Network Simulator
Conclusion

- **Dynamic ad hoc network**:  
  - a next step in distributed computing?  
  - How to build distributed applications?

- **Best-effort approach**:  
  - algorithms do their best  
  - the result depends on the dynamic

- **Best-effort algorithm**:  
  - self-stabilizing + continuity in the outputs  
  - depending on the dynamic

- **Application**:  
  - **Group Service in vehicular networks** Code and videos:  
    - http://www.hds.utc.fr/~ducourth/airplug  
  - **Unison in vehicular networks**