A DSA Model For Data Access In Self-Organizing Systems

Dhavy Gantsou

Dhavy.gantsou@univ-valenciennes.fr
Self-Organizing System?

- Distributed system with:
  - reactive characteristics
    - Ability to detect unusual behavior
  - self-healing characteristics
    - Ability to carry out corrective actions in order to optimize the system’s behavior

- Example of self-organizing system: Mobile Ad Hoc Network (MANET).
Mobile Ad Hoc Network (MANET)

- Network deployed without pre-existing fixed infrastructure
- Consists of mobile nodes connected by wireless links
- Each node acts both as router and host
- Dynamic topology
- Fully decentralized
MANET characteristics
Dynamic topology

E
B
A
D
C
G
F
Data access requirements

- Learning about data availability
- Accessing those data
Data access issues

- Lack of centralized server
- Dynamic node constellation
- Changing link’s status
Data access issues (1)
Lack of centralized server

- Maintaining ones is
  - unreliable, and
  - expensive because of the unpredictable topology (nodes can join and leave the network without warning)
Data access issues(2)
Dynamic node constellation
Data access issues(3)

Potential change of link’s status

- Connectivity between two nodes may change over time:
  - **Symmetric**
    Node A and B can receive messages from each other.
  - **Asymmetric**
    Node A can receive messages from B, then B may not necessarily receive messages from A.

⇒ Need for a real-time lookup service
Key concepts for the design of the data access API

- Network partitioning
  - Group (group leader, group leader elector)
- Each node maintains a database
  - neighbor set, 2-hop neighbor set, group leader set, group leader elector set
  - status of links between nodes
- Time based database management
- Database consistency through exchange of messages between nodes
network partitioning
DSA access API

Overall architecture

- **Database management (Remote Type)**
  - Time based location tracking
  - Time based connectivity tracking

- **Node implementation** (multithreaded procedure)
  - Thread 1: communication with user
  - Thread 2: Sends requests.

- **Common Interface (Remote Type)**
  - Abstract type
  - Abstract methods

- **Proxy (Remote Call Interface)**
  - Node registry
  - Requests dispatching

- **RPC**
Managing connectivity

Node A Data base: symmetric link to B.

Node B Data base: symmetric link to A

Node C Data base symmetric link to A

A broadcasts a message to its neighbors
DSA data access API services:
Node registry

Database (RT)

Interface : (RT)

Proxy : (RCI)
Uses a protected object for Database remote reference registry
DSA data access API services: group leader election.

Group leader Election

Group leader Election root

Node.

Proxy: (RCI)

Group leader elected
Conclusion

- API implemented

- Works on network of PC running Linux
Future work

- Implementing multicast communication between nodes
- Requires to extend Glade ???.
- Implementing a GUI (GTK does not support tasking)
Thanks