Topology Discovery for Autonomic Networks

Background

The ANIMA Working Group [1] of the IETF has specified the operation of nodes in an Autonomic Network (AN). In that specification, autonomic nodes initially have a unique identifier, and (only) link-local communication to adjacent nodes.

An autonomic node is accepted into a domain by the Domain Registrar, based on the node's unique identifier. It acquires a globally routable address (which serves as a more useful identifier), becomes aware of the address of the Registrar, and can exchange messages with it.

Routing is done using RPL, which is a Distance-Vector (DV) routing protocol. RPL builds a Directed Acyclic Graph (DAG) from the initiating node to all other nodes. There can be multiple instances of RPL in a domain. As a result of the DAG structure, packets may go from a source node to the root node and back out to the target node, rather than on a shortest path.

In Link State (LS) routing protocols, each node has full knowledge of the network topology. In Distance Vector routing protocols, knowledge of the full topology is not needed, which makes them more scalable. It also makes them less suitable for a topology-discovery problem.

The ANIMA documents are all at [1]; click on the "Documents" tab.

Goal

To explore and compare various methods of displaying the full topology of the Autonomic Network from the viewpoint of an arbitrary place (node) in the network, using only the information available within the Autonomic Network.

Ideas to explore

1. Centralized algorithm: The Domain Registrar asks each enrolled node for a list of its neighbors, and gets back a list of identifiers and the interface on which it appears.
2. Decentralized algorithm: The discovery of the topology is piggybacked on the construction of the DAG by RPL. Specifically, RPL provides a list of all nodes in the

domain, and this list is used as in approach #1. The difference is that the Registrar is not involved.
3. Decentralized algorithm: The discovery of the topology is done using a new algorithm based on an LS approach. This approach is independent of both the Registrar and RPL.
4. The effect of updates on the algorithms: which one adapts more easily to the dynamic situation?

Dimensions (initial list; will need to be expanded)

1. Number of messages exchanged
2. Time to converge for a static graph
3. Optimal time between refreshes for a dynamic graph (probably depends on the diameter of the graph)
4. ?

Characteristics of the ideal candidate

1. Good software skills
2. Solid theoretical background
3. Good understanding of Human-Computer Interaction
4. Good understanding of layout algorithms for network topologies
5. ?

Available test bed

- A python-based implementation of the communications protocol called GRASP, which was developed by the ANIMA Working Group.
- Some example Autonomic Service Agents (ASAs), also in Python, which serve as test programs and a demonstration of the correctness of the GRASP implementation.
- A test bed with 10 Linux-based PCs, having 10 subnets with distinct address ranges. (This can be made larger at need.)

It should be relatively easy to code an implementation of the thesis ideas as ASAs and place them on the 10 machines, as a demonstration of the correct operation of the proposed algorithm(s). (Note, however, that just doing such an implementation would not be sufficient for a thesis project.)

References

[1] https://datatracker.ietf.org/wg/anima/about/