IN4330 2011-2012

Large-Scale Adaptive Systems (LSAS)

Clustering in Mobile Ad-Hoc Networks

SACA - Switch Along Clustering Algorithm

Student: Victor-Lucian Spiridon
(v.spiridon@student.tudelft.nl)

Student no.: 4184599

NetID: vspiridon

1. Explanation of basic mechanism

With SACA we target the grouping of nodes into convex clusters of equal number of nodes. The algorithm assumes that the there is no churn (no nodes enter or leave the system) but some packet loss. We start by initializing the Cluster Heads and a number of nodes with the token (color) of each cluster. Note that the nodes are randomly placed in the system and respect the random walk or random waypoint mobility models.

The clusters will be formed based on local (1-hop) decisions and interactions of any two pair of nodes having different colors. The *target* of any node is to *move* closer to its cluster head. This guarantees that the clusters will have a convex shape. The *moving* relates to exchanging of tokens between the nodes and not to the random mobility model.

The decisions are based on two metrics (each node computes the metric values for itself). The switch is performed when it follows the interest of both the nodes from the pair with respect to the metrics i.e., it will improve at least one of their metrics if the switch is done. The first metric (hop-count) is a combination of two values: the number of hops to its cluster head and the number of hops to the cluster head of its pair. The second metric (neighbor majority) is also a combination of two values: the number of neighbors with his color and the number of neighbors with its pairs color. Therefore, a node tends to minimize the hop-count to its cluster head and to maximize the number of neighbors with the same color as his color.

2. Mechanism implementation details

2.1. Computing the hop-count

We have seen, in the first section that the hop-count is critical for the clustering to emerge. If we choose to take decisions based only on the neighbor majority metric, the nodes will group in several smaller dispersed clusters and not in a big convex cluster.

Each node has its minimum hop-count to every cluster head in the system. This is obtained using a push-pull gossiping mechanism. Each of the two nodes in the gossiping pair will compute the minimum hop-count for each cluster head between its values and the partner's values. Note that the partner's values are incremented by one. That is because if we choose to set our shortest road through the partner, then we must count the hop that separates the pair.

Using flooding (broadcasting) as a dissemination method is tempting but not very energy efficient. Instead experiments showed that gossiping converges fast enough. However, several rounds of gossiping are necessary per tick. Because the network is mobile it is important to

keep the hop-count values updated. This is difficult because the value of the hop-count for a specific cluster head cannot be increased; only changes that minimize it are supported. We overcome this challenge by initiating a new gossiping session every *gradientRoundInterval* ticks. The session is initiated by each cluster head by incrementing its variable *gradientRound*, and by disseminating 1 for the hop-count associated to its color and a very big hop-count for the rest.

Taking into account that we have multiple rounds, when a partner from a pair has values from an older round than the other, it will take the newer values and discard the old ones. By these rules we assure that the hop-count values are up to date.

2.2. Switching decision

At every tick, each node A tries to form a pair with a neighbor of a different color. This starts after it has received the first hop-count gossip. If it succeeds in pairing, A and B will compute four values:

1. Number of neighbors of the same cluster as himself
2. Number of neighbors of the same cluster as his pair
3. Hop-count to his Cluster Head (CH)
4. Hop-count to his pair Cluster Head (CH)

The initiator gets his partner's values and based on them it tries to decide whether they should switch colors or not. Below are the decision steps:

If 1a. A has a bigger hop-count to A's CH than B **and**
1b. A has a smaller or equal hop-count to B's CH than B to A's CH
Else If (2a. A has fewer neighbors of A's color than B **and**
2b. B has fewer neighbors of B's color than B) **or**
(2c. B has no neighbors of B's color **and**
2d. has neighbors of A's color)

If one of the If's is satisfied then the colors are exchanged. Clauses 2c. and 2d. have the task of solving situations when one node, being at the border of the system, is surrounded only by cluster nodes of the same different color. By this rules A makes a decision not in his interest, but necessary for the convergence of the algorithm. Furthermore, the second set of clauses (2*) have the role of accelerating the convergence.

2.3. Migrating the Cluster Head

During the first ticks, right after the initialization, the Cluster Head is surrounded by nodes of different colors, and it can happen that during the clustering time, they become isolated in other clusters. In order to avoid this, the following decisions are made by all the Cluster Heads at every tick:

If I have in range at least one stranger (neighbor who is not from my cluster)

Migrate to a neighboring node from my cluster which has:

1. Fewer strangers than I have **or**
2. More neighbors from our cluster than I have

These clauses are most of the times sufficient to assure that the CH will only be surround by nodes from its cluster.

2.4. Simulating Packet Loss

In simulating the packet loss we assumed that if a packet is lost all the communication session is discarded. Therefore, there is a fraction chance **f** of a session to fail between any two turtles. Note that communication sessions are opened during the gossiping, the switching and the cluster head migration. This is implemented by discarding (refusing) a perfectly viable partner with a fraction **f** chance.

3. Comparison to ASH

Compared to ASH, SACA has a slower convergence time. It needs a higher degree of communication and the cluster borders are not that well defined as with ASH. Furthermore, it needs a special type of initialization: pre-coloring the nodes. On the other hand, SACA can maintain the node count of each cluster during time. It converges faster and generally acts better with higher node mobility. Like ASH, it is self-healing meaning that the moving speed of the clusters is much smaller than the moving speed of the nodes.

4. Limitations of SACA

The local rules that help SACA converge introduce some unwanted behavior in the system. One is that the borders between the clusters tend to be always changing, as if a node at the border is not able to make up his mind what cluster to choose. This behavior increases when more than 30% packet loss is introduced.

In general the bigger the transmission range and the bigger the number of nodes the faster the algorithm converges. However if the network is not dense (few neighbors / node) enough (approx. 3 neighbors / node) but very mobile the algorithm fails to converge.

- The first testing metric (**mean [standard-deviation [clusterId] of in-link-neighbors] of turtles with [(count in-link-neighbors) > 1]**) stays within the values (0.2 , 0.35) after convergence.
- The second testing metric (**standard-deviation clusterSizes / mean clusterSizes**) is always 0.