Clustering Algorithm for MANETs based on mobility prediction

Vincent Bricard-Vieu, Noufissa Mikou*

*LIRSA, Faculté des Sciences Mirande 9, av A. Savary BP 47870, 21078 Dijon Cedex
vincent.bricard-vieu@u-bourgogne.fr
nmikou@u-bourgogne.fr

Abstract: In mobile ad hoc networks (MANETs), the network topology is autonomously formed and continuously changes, due to the mobility of the nodes. Clustering allows us to organize the topology in a structured manner. The association and dissociation of nodes to and from clusters perturb the stability of the network topology, and hence reconfiguration of the system is often unavoidable. Several existing on-demand clustering algorithms update that topology if needed.

In this paper, we aim to anticipate these updates by predicting the mobility of the nodes, so we propose a new distributed Mobility Prediction-based Weighted Clustering Algorithm based on an on-demand distributed clustering algorithm.

Simulation experiments are conducted to evaluate performances of our algorithm and compare them to those of the weighted clustering algorithm (WCA), which does not consider prediction. Results show that our algorithm performs better than WCA, in terms of updates of the dominant set, handovers of a node between two clusters and average number of clusters in a dominant set.

Keywords: Ad Hoc networks, clustering, mobility prediction.

1 INTRODUCTION

The rapid advancement in mobile computing platforms and wireless communication technology lead us to the development of protocols for easily deployable wireless networks typically termed wireless ad hoc networks. These networks are used where fixed infrastructures are non-existent or have been destroyed. They permit the interconnectivity between workgroups moving in urban or rural area. They can also help in collaborative operations, for example, distributed scientific research and rescue.

A multi-cluster, multi-hop wireless network should be able to dynamically adapt itself with the changing networks configurations. Some nodes, known as cluster-heads, are responsible for the formation of clusters each consisting of a number of nodes (analogous to cells in a cellular network) and maintenance of the topology of the network. The set of cluster-heads is also called Dominant set. A cluster-head is responsible of resource allocation to all nodes belonging to its cluster. Due to the dynamic nature of the mobile nodes, their association and dissociation to and from clusters perturb the stability of the network and thus reconfiguration of cluster-heads is unavoidable.

The paper is organized as follows. Section 2 describes previous clustering algorithms. In section 3 we propose a new distributed Mobility Prediction-based Weighted Clustering Algorithm and compare in section 4, using simulations, its performances to those of the Weighted Clustering Algorithm (WCA) (Chatterjee et al., 2002). Finally, we conclude our study.

For our simulations, we use GloMoSim (Zeng et al., 1998). GloMoSim is a discrete event parallel environment based on PARSEC (PARallel Simulation Environment for Complex systems) (Bagrodia, 1998).

2 RELATED WORKS

Current algorithms for the construction of clusters contained in many routing protocols, as well as clustering heuristics, such as the lowest-identifier (Jiang et al., 1999), the highest-degree (Gerla, 1995) (Hou, 2001) and the Linked-Cluster Algorithm (LCA) (Mitelman, 1999) (Baker, 1981), have proactive strategies. By proactive, we mean that they require a constant refresh rate of cluster dependent information. That introduces a significant background control overhead even if there is no data to send. The major difficulty comes from node mobility, which has an impact on the position of the nodes and on the neighborhood information, which is essential for
clustering. To ensure the correct collection of neighborhood information, existing clustering solutions rely on periodic broadcast of the neighbor list. Mobility causes adjacency relations to change. As well as in Lowest Distance Value (LDV) and the Highest In-Cluster Traffic (ICT) (Habetha et al., 2000), depending on nodes movement and traffic characteristics, the criterion values used in the election process can keep on varying for each node, and hence result in instability.

Proposed by Chatterjee et al (Chatterjee et al., 2002), the Weighted Clustering Algorithm (WCA) works differently of the algorithms described above since it is only invoked on demand by isolated nodes. Moreover, to determine the cluster-head nodes, that algorithm considers the ideal number of nodes that a cluster can handle, the mobility (speed of nodes), the distance between a node and its neighbors and the battery power. WCA assigns weights to these different parameters.

The cluster-head election procedure is invoked at the time of system activation, and also when the current dominant set is unable to cover all the nodes. Every invocation of the election algorithm does not necessarily mean that all the cluster-heads in the previous dominant set are replaced by new ones. If a node detaches itself from its current head-cluster and attaches itself to another cluster-head, then involved cluster-heads update their member list instead of invoking the election algorithm. See detailed description of WCA in section 5 and in (Chatterjee et al., 2002).

After the election, all the nodes are in clusters with a cluster-head in each cluster and each node has a list constituted by its neighbors and the set of all the cluster-heads.

All nodes continuously monitor the signal strength of a Hello messages received from the cluster-head. When the distance between the node and its cluster-head increases, the signal strength decreases. In that case, the mobile has to notify its current cluster-head that it is no longer able to attach itself to it. The node tries to handover to the first neighboring cluster which cluster-head is the first found in its list. If the node goes into a region not covered by any cluster-head, then the WCA election procedure is invoked and a new dominant set is obtained.

Unfortunately, these periodic hello messages induce a high communication overhead.

3 OUR ALGORITHM : MPWCA

As mentioned above, the overhead induced by WCA is very high, since it uses a large part of bandwidth which cannot be used for useful data transmissions.

To avoid that overhead, we propose to increase the duration between two hello messages. Due to nodes mobility, the topology is always changing. Increasing the duration between two Hello messages will lead to link breaking since a node can go out of its cluster without knowing it. So, we propose a distributed mobility prediction-based mechanism using the past movements of the cluster-heads to replace the missing informations given by frequent hello messages.

3.1 Description

Our estimation algorithm starts after the election of the cluster-heads, when the ordinary nodes are monitoring the signal strength of packets from their cluster-head. It works as follow:

1. The cluster-head periodically sends informations about its position and its speed in Hello messages. When ordinary node receives these Hello messages, it stores the informations about its cluster-head into a list named information list. The stored informations are:
   - the position of the cluster-head in Cartesian coordinates (x, y, z).
   - the speed of the cluster-head

2. If an ordinary node has less than two past informations about its cluster-head in its list, during the between two hello messages, it waits for the next Hello message (step 1). Otherwise, it use the past information list to estimate the current position and speed of their cluster-head and store them in a list named prediction list. Since the time interval between two Hello messages can be very large, the ordinary node could make more than one estimation which will be stored in its prediction list. In this case, the prediction list is appended to the past information list, to make other estimations. The computing of an estimated position is detailed in subsection 3.2.

3. Using their prediction, the ordinary node decides if it should stay in its current cluster or not. To this goal, the ordinary node computes the distance to the estimated position of its cluster-head. It compares this distance to the transmission range, which is the same for all nodes. If the distance is less than transmission range, the ordinary node stays in its cluster. Otherwise, it tries to handover to another neighboring cluster-head. Even if it cannot find another cluster-head in its neighborhood, it stays in its
current cluster, waiting for the estimation or the next \textit{Hello} message, thus to avoid updates of the dominant set which are not needed, due to false estimations.

The ordinary nodes make estimations until they receive a new \textit{Hello} message from their cluster-head. After the ordinary nodes have received a new \textit{Hello} message, their prediction list is cleared.

Since an ordinary node estimates position using past information, it needs to take "fresh information". So, the past information list has a finite size (in our experiments, we choose to keep at most 10 positions). When the list is full, the new inserted information drives the oldest information out of the list.

3.2 Estimation Computation

all ordinary nodes need to estimate the position of their cluster-head. To explain the estimation computation, we suppose that a given ordinary node has already estimated positions stored in its prediction list. To estimate the next position of its cluster-head, the ordinary node appends its prediction list to its past information list to form a bigger list \( L \). Let us note \( P = \{ p_i = (x_i, y_i, z_i) \} \) the list of \( N \) last positions of its cluster-head (in Cartesian coordinates) stored in \( L \).

In our experiments we choose \( N \) such as \( 2 \leq N \leq 10 \)

We compute the \( N-1 \) vectors:

\[
p_i p_{i+1} = (x_{j+1} - x_j, y_{j+1} - y_j, z_{j+1} - z_j)
\]

for \( i = 1..(N-1) \).

Then we compute the average moving vector:

\[
\overline{D} = \frac{1}{N-1} \sum_{i=1}^{N-1} p_i p_{i+1}
\]

Finally, the estimated position \( p_{N+1} \) is computed by translating the last position \( p_N \) by the vector \( \overline{D} \)

\[
(p_N p_{N+1} = \overline{D})
\]

The position \( p_{N+1} \) is then stored in the prediction list.

4 PERFORMANCE EVALUATION

Using simulations, we show that our algorithm (MPWCA) performs better than WCA in terms of number of \textit{updates} of a dominant set, number of successful \textit{handovers} of a node in a cluster and average number of \textit{clusters}.

4.1 Simulation study

We simulate two systems of 50 and 100 nodes respectively on a 1000m x 1000m area. The nodes have a transmission range of 100m and 200m. The nodes can randomly move in all possible directions with speed varying uniformly between 0 and one parameter representing the maximum value of the speed.

The cluster-head election takes place at the start of the simulation and when a node can no longer be covered by the dominant set. See details of the election procedure in section 5 and in (Chatterjee et al., 2002).

For this election, we assume that each cluster-head can handle \( \delta=3 \) nodes (ideal degree) in its cluster in terms of resource allocation. We choose arbitrarily \( \delta=3 \) for many reasons. If the transmission range is small and the density of nodes in the network is weak, the cluster-head will have a little connectivity. Furthermore, if the bandwidth is limited, the cluster-head will be unable to handle too many nodes at the same time. Finally, it is the value used by GloMoSim users.

Moreover, in this election, the choice of parameters \( w_i (i = 1..4) \) is done as in (Chatterjee et al., 2002) and as GloMoSim users. The weight \( w_1 \) of the difference between the current degree of the node and \( \delta \), and the weight \( w_2 \) of the sum of distances to the neighbors of the nodes (see section 5) are higher than the others because we want properties of connectivity and promiscuity with neighbors to be more important for a good cluster-head than low mobility and low consumed battery energy. In our experiments, the values used are \( w_1=0.7, w_2=0.2, w_3=0.05 \) and \( w_4=0.05 \).

When all cluster-heads are chosen, they start sending hello messages with a period of 2s, instead of 1s, as usually used in WCA. Then, ordinary nodes start predictions (step 2 to step 3 described in section 3). To reduce the impact of overhead induced by hello messages, we divide their frequency by two. But, that could involve instability of the hierarchy so, we add location estimations between hello messages to replace the missing informations and we show that the hierarchy remains stable.

In our experiments, ordinary nodes make two estimations before receiving the next hello message. After that, the prediction list is cleared and the algorithm is then in step 1.

To measure performances of our system, we consider three metrics:

- the number of \textit{updates} of the dominant set.
- the number of successful \textit{handovers} between two clusters.
- the average \textit{number of clusters} in the dominant set, which characterizes the load of clusters.
These three parameters are studied as a function of the maximum speed of the nodes.

4.2 Simulation results

In our simulation experiments, we choose values 1m/s, 5m/s, 10m/s, 20m/s, 30m/s and 40m/s for the maximum speed of nodes. The lowest value is equivalent to walking speed and the highest value corresponds to a car speed on some highways. The nodes move randomly and uniformly in all possible directions.

Figure 3. number of successful handovers vs maximum speed (50 nodes)

Figure 4. number of successful handovers vs maximum speed (100 nodes)

On Figure 3 and Figure 4, we can see that, for a fixed value of speed, our algorithm allows a higher number of successful handovers than WCA.

We can also observe that the number of successful handovers increase while speed increases. Due to the mobility, nodes do not always stay in the same cluster. But, there are less changes when nodes have a high transmission range or when they move slowly.

Figure 1. number of updates of the dominant set vs maximum speed (100 nodes)

Figure 2. number of updates of the dominant set vs maximum speed (50 nodes)

Figure 1 and Figure 2 show that, for a fixed value of speed, our algorithm gives better results for this metric, since WCA involves more updates of the dominant set than our algorithm, and the cost of these updates is higher in terms of resources allocation such as CPU and bandwidth. As well as for handovers, we can see that the number of updates of the dominant set increases while speed increases, due to mobility.

The higher the number of nodes in a cluster is, the more the dominant set is stable. Nevertheless, each cluster should not be overloaded, because nodes need to load the cluster-head to communicate. In our experiments, the predefined number of nodes that a cluster-head can handle ideally is three nodes (in addition to the cluster-head). This number is defined considering the parameters commonly used in GloMoSim.
Figure 5 and Figure 6 show that, for a fixed value of speed, our algorithm (with an average value of 2.7 nodes per cluster) is closer to this ideal number than WCA (with an average value 2.5 nodes per cluster).

We can also note that the number of clusters increases when speed increases because, due to fast mobility, the dominant set tends to be unstable, but we can notice that with our algorithm, the number of clusters increases slower than with WCA.

CONCLUSION AND FURTHER WORKS

In this paper we propose a new distributed Mobility Prediction-based Weighted Clustering Algorithm (MPWCA). To limit the overhead induced by control messages such as Hello messages, we increase the interval between two messages. During this long time, nodes try to estimate the movement of their cluster-head and then anticipate handovers, to avoid link breaks.

Using GloMoSim (Zeng et al., 1998), we simulate two networks of 50 and 100 nodes respectively, uniformly distributed on a 1000 m x 1000 m area. To compare the performances of our new distributed mobility prediction-based weighted clustering algorithm with the Weighted Clustering Algorithm (WCA), we consider three metrics characterizing the stability of the dominant set: the number of updates of the dominant set, the number of handovers of a node to another cluster and the number of clusters. We show that our algorithm ensures a better stability of the dominant set than WCA.

In future work, we will investigate the performances of our prediction mechanism on a locally-centralized system. By locally-centralized we mean that we will always use a cluster-based architecture but the prediction will work on the cluster-head itself instead of ordinary nodes.

5 APPENDIX A : CLUSTER-HEAD ELECTION IN WCA

The algorithm for the cluster-head election in WCA is the following:

1. Find the set $N(v)$ of neighbors of each node $v$ (i.e., nodes $v'$ such that the distance between $v$ and $v'$ is less than the transmission range of $v$). Set $d_v$, the degree of $v$, the cardinality of $N(v)$.

2. Compute the degree-difference $\Delta_v = |d_v - \delta|$ for each node $v$, where $\delta$ is the number of nodes (pre-defined threshold) that a cluster-head can handle ideally.

3. For every node, compute the sum of the distances $D_v$ with all its neighbors $D_v = \sum_{v' \in N(v)} \text{dist}(v, v')$

4. Calculate the combined Weight ($W_v$) for each node $v$ where $W_v = w_1 \times \Delta_v + w_2 \times D_v + w_3 \times M_v + w_4 \times P_v$

5. Choose that node with the smallest $W_v$ as cluster-head. All neighbors of the chosen cluster-head are no more allowed to participate in the election procedure.

6. Repeat steps 2 to 5 for the remaining nodes which are not yet selected as a cluster-head or assigned to a cluster.

RÉFÉRENCES


