On Exploiting Diversity for Cluster Formation in Self-Healing MANETs

Ann T. Tai Kam S. Tso
IA Tech, Inc.
923 Euclid Street, #301
Santa Monica, CA 90403

William H. Sanders
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

1 Introduction

Ad hoc wireless network technology, serving as a basis for computation nomadicity, embeddedness, and ubiquity, has been enabling our world to be heterogeneously networked by an invisible global infrastructure [1]. By building and maintaining network hierarchies among mobile wireless devices, large-scale networks can scale up affordably. Various clustering algorithms have been thereby devised as building blocks for the purpose of scalability.

To date, most clustering algorithms are iterative in nature, implying that their performance is dependent of network dynamics. Clustering protocols are typically invoked periodically to re-cluster the nodes which may migrate, fail, or die due to power exhaustion. If the performance cost of a protocol prevents it from being invoked frequently enough, clustering coverage may deteriorate excessively between two clustering points. Consequently, various cluster-maintenance algorithms were proposed to resolve the problem. Nonetheless, cluster-maintenance routines are often cumbersome and costly, potentially defeating the purpose of clustering.

In this paper, we propose a notion called superimposed clustering. A superimposed clustering protocol (SCP) applies two diversified clustering policies simultaneously to form two different sets of clusters which we view as two cluster layers with one on the top of the other. As a result, the superimposed layers enable significantly more nodes to be clustered in a single round. In turn, this enables an SCP to be invoked more frequently for a self-healing mobile ad hoc wireless network (MANET). In the following sections, we present an algorithm for SCP and an analytic evaluation.

2 Superimposed Clustering

2.1 Basics of Clustering

A cluster can be viewed as a unit disk with a radius equal to the center node's transmission range. The unit disk is called a "cluster" with the center node being the *clusterhead (CH)* and with all the non-center nodes being the *cluster members (CMs)*. If node v is 1) a one-hop neighbor of the CHs of two different clusters, or 2) a one-hop neighbor of a CM of another cluster, v can become a direct or internal *gateway (GW)*, respectively. A node that is located outside two clusters but has at least one neighbor (a CM) in each of the clusters can become an external gateway.

After the distributed autonomous cluster formation, CHs and GWs constitute a backbone for inter-cluster communication while CMs may talk to each other either directly or via their CHs. The two-tiered communication architecture, namely, intra-cluster and inter-cluster communications, thereby enables network scalability.

2.2 MaxMin-ID-Based SCP

Among other choices to combine two clustering policies together to realize superimposed clustering, in this paper we choose to combine the well-known maximum ID (Max-ID) and minimum ID (Min-ID) policies to compose an SCP. With this combination, call it *MaxMin-ID*, the nodes that have the largest or smallest IDs within their neighborhoods (determined by their transmission range) will be self-elected as CHs to accommodate the nodes in their neighborhoods as their CMs. Due to the symmetry, the Max-ID and Min-ID clustering policies require exactly the same neighborhood information, implying that the parallel use of the two policies does not entail additional message exchange and thereby does not incur additional performance cost or energy consumption.

More importantly, with superimposed clustering, we do not stress a perfect clustering coverage, rather, we focus on letting the superimposed cluster layers cover significantly more nodes in a single round. To aid description, we introduce the following notation:

$N_1[v]$:	The set that enumerates all the one-hop neighbors of node v (excluding v itself).
$\hat{N}_1[v]$:	The set that enumerates v's one-hop neighbors whose IDs are larger than that of v.
$\check{N}_1[v]$:	The set that enumerates v's one-hop neighbors whose IDs are smaller than that of v.
$G_h[v]$:	A variable which indicates whether v is qualified to be a cluster head.
$G_m[v]$:	A variable which indicates whether v is qualified to be a cluster member.

Based on the definitions of $G_h[v]$ and $G_m[v]$, we can then define the word "clustered" as follows:

Definition 1 *A node v is said "clustered" iff the condition $G_h(v) = 1 \vee G_m(v) = 1$ holds.*

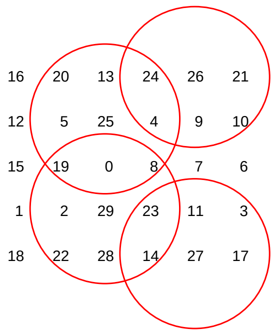
Then with the MaxMin-ID policy stated earlier, v will become a CH if and only if the following condition is true:

$$N_1(v) \neq \emptyset \ \wedge (\hat{N}_1(v) = \emptyset \vee \ \check{N}_1(v) = \emptyset)$$

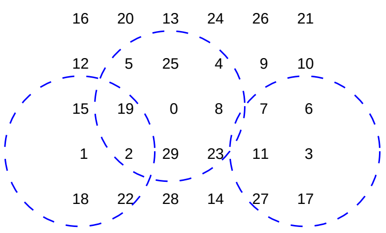
It follows that v will be clustered as a CM if and only if the condition below holds:

$$\exists u,\ u \in N_1(v),\ \hat{N}_1(u) = \emptyset \vee \check{N}_1(u) = \emptyset$$

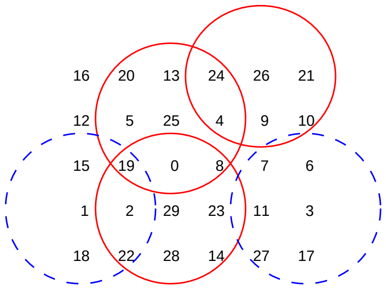
Fig. 1 illustrates the idea of superimposed clustering via an example. For clarity, we let 30 nodes be distributed in a 6×5 matrix. Each node is represented by the ordinal number of its ID, while the placement of the ordinal numbers is done using a random number generator. Fig. 1(a) and Fig. 1(b) show that when Max-ID and Min-ID policies are applied alone, 7 and 11 nodes are left unclustered, respectively, upon the completions of a single round of the corresponding clustering protocols.



(a) By Max-ID Policy



(b) By Min-ID Policy



(c) By MaxMin-ID Policy

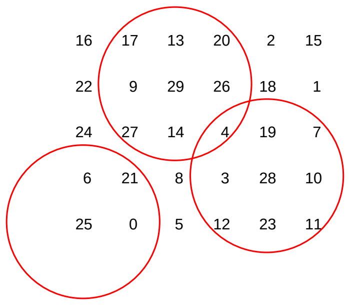
Figure 1: Superimposed Clustering: Example 1

On the other hand, when the two policies are applied in parallel, only two nodes are left unclustered after a single round, as shown in Fig. 1(c), a significant improvement of clustering efficiency.

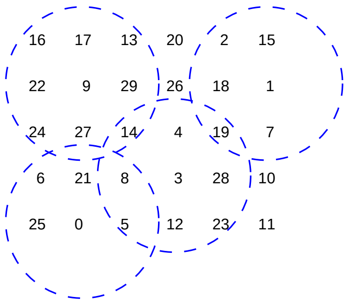
Likewise, Fig. 2 shows that MaxMin-ID achieves a perfect coverage in another randomly generated case in which Max-ID and Min-ID policies leave 9 and 4 nodes unclustered, respectively.

2.3 Algorithm

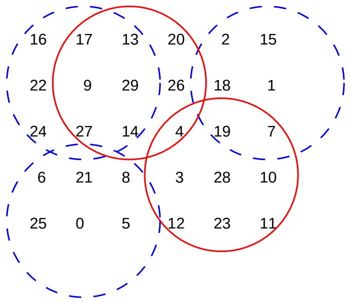
Algorithm 1 enables the MaxMin-ID-based superimposed clustering. As shown, the single-round SCP involves three



(a) By Max-ID Policy



(b) By Min-ID Policy



(c) By MaxMin-ID Policy

Figure 2: Superimposed Clustering: Example 2

message exchanges altogether between a node v and its one-hop neighbors. Specifically, it begins by letting each node broadcast its ID to enable neighborhood probing (line 1); upon the receipt of the messages from the neighbors, each node reports the collected IDs (line 3). Thus by the end of the second message exchange (line 4), v will have the complete knowledge about $N_2(v)$ which is the union of $N_1(v)$ and $N_1(u)$, $\forall u \in N_1(v)$. Based on the knowledge, v will first determine if itself can become a CH candidate per the MaxMin-ID policy (line 5); v will also check if there exist any nodes in $N_1(v)$ that are qualified to be the candidates of CH (line 6).

It is noteworthy that while the two independent CH qualifying policies will never choose the same node to be a CH, it is possible that a cluster overlaps entirely with one or more clusters (of the same or different types). To completely avoid or remove such redundancies using fully distributed algorithms is similar to the NP-complete problem addressed in [2]. In particular, with some overlapping patterns, a potential CH of a cluster C will be unable to determine if any of the clusters overlapping with C will decide to withdraw too (so that C's withdrawal will be unsafe). To circumvent that situation, we limit ourselves to partial redundancy checking and pruning.

Algorithm 1 Superimposed Clustering

```
1: send(nID(v));                                    // ID diffusion
2: $N_1(v) \leftarrow$ receive(integerSet);          // $N_1$ is a set of nIDs of 1-hop neighbors of v
3: send(nID(v),$N_1(v)$);                            // neighborhood info exchange
4: $N_2(v) \leftarrow$ receive(2-tupleSet);          // $N_2$ is a set of $\langle nID, N_1 \rangle$
5: CHstatus(v) $\leftarrow$ MAXorMIN(nID(v),$N_1(v)$);
6: CHs(v) $\leftarrow$ identifyCH($N_2(v)$);
7: CHstatus(v) $\leftarrow$ redundancyChecking(nID(v),CHstatus(v),$N_1(v)$,$N_2(v)$);
8: send(nID(v), CHs(v), CHstatus(v), nil);
9: cRegistry(v) $\leftarrow$ receive(4-tupleSet);
10: CHstatus(v) $\leftarrow$ redundancyPruning(CHstatus(v), cRegistry(v));
11: identifyGW(cRegistry(v));
```

Specifically, the partial redundancy checking by node v (line 7) enables v to 1) avoid forming a cluster that will cause Type-A or B redundancy, and 2) prepare for a subsequent removal of cluster redundancy of Type B. Type A refers to the scenario in which $\{N_1(v) \cup v\} \subset N_1(u)$, where both v and u are potential CHs; while Type B redundancy refers to the scenario in which $N_1(v) \subset \cup_{w \in W} N_1(w), \forall w, N_1(w) \not\subset N_1(v)$, where v is a CH candidate based on the Min-ID policy and W is a nonempty node set consisting of the CH candidates per the Max-ID policy. Note at the end of the second message exchange (line 4), v will have adequate knowledge to evaluate the conditions that will yield Type-A and Type-B redundancies. In addition, it can be mathematically proved that to avoid such redundancies at that point is safe. Accordingly, SCP permits v, which is involved in a Type-A or B cluster redundancy, to determine and report its CH-candidate status prior to and during the third message exchange, respectively.

Finally, Type-C redundancy refers to the scenario in which $N_1(v) \subset \cup_{w \in W} N_1(w), \forall w, N_1(w) \not\subset N_1(v))$, where v is a CH candidate per the Max-ID policy and W is a nonempty node set consisting of the CH candidates that are qualified per the Min-ID policy. Although Type-C redundancy is symmetric to Type-B redundancy, it will not be safe for v to withdraw its CH candidacy upon the detection of the redundancy. This is because before learning the effects of the redundancy-Type-B avoidance decisions from v's peers, v will be unable to determine whether such redundancy will persist.

Hence v's decision will be postponed until after the completion of the third message exchange (line 10). By confirming that the redundancy yet exists at that point, v will remove a Type-C redundancy by downgrading its CH status to "passive CH." A cluster registry, as shown in Table 1, enables v to use the node ID of each neighbor as an index, to access and maintain the knowledge about which potential CHs a neighbor u can hear from, the CH status of u itself, and whether it is eligible to be a GW.

Table 1: Cluster Registry of Node v

nID(u), $u \in N_1(v)$	CHs(u)	CHstatus(u)	GW(u)
2	$\{1, 29\}$	"notaCH"	1
1	$\emptyset$	"CHmin"	0
…	…	…	…

The last step in the cluster formation is gateway discovery. If a node v is a CH, v will use its cluster registry to identify among its CMs who are also the neighbors of the CHs of the neighboring clusters. If v finds u is such a *direct gateway* candidate, v uses nID(u) as an index to access the GW field in that row (see Table 1) and puts a mark "1" there.

When v is a CM that hears only from its own CH, v will still participate in the GW search based on its cluster registry. That is, v looks for a neighbor u that is affiliated with a cluster other than that v is affiliated with. If v finds such a neighbor u, v will mark "1" in the GW field of the row indexed by nID(u). suggesting an inter-cluster connection established by two internal GWs.

As to an unclustered node v, it will search each of its neighbors to see if it is affiliated with a cluster. If such a node, call it u, is identified, v will mark "1" in the GW field in the row of its cluster registry indexed by nID(u). This enables a link between an external GW v and an internal GW u.

When gateway discovery completes, v's cluster registry will contain adequate information to enable route discovery for inter-cluster communication. Furthermore, the GW information so obtained enables the inter-cluster communication layer to adopt any routing protocols. Note also that the algorithm may identify multiple GWs for an inter-cluster connection. That will offer robustness benefits to the scenarios where 1) a GW dies or migrates away, and 2) energy balancing is necessary.

3 Analytic Model

For a quantitative assessment, we evaluate the performability of the MaxMin-ID-based SCP in terms of its clustering coverage, i.e., the probability that a node will be clustered by SCP.

We note the likelihood that a node v will be qualified to be a CH under the SCP depends upon the value rank of nID(v). As node IDs need only to be unique but not necessary to be consecutive, we let each node be associated with an ordinal number that ranks its ID (those ordinal numbers are both unique and consecutive). To take into account the the node population density, we assume nodes are uniformly distributed in a finite-size field. Then, by letting C_2 denote that a node is clustered under the MaxMin-ID-based SCP, our performability measure can be formulated as follows:

$$P(C_2) = \sum_{i=1}^{I_{max}} P(ID(v) = i) \sum_{n=1}^{I_{max}-1} P(\mid N_1(v) \mid = n) \quad (1)$$
$$P(G_h(v) = 1 \;\vee\; G_m(v) = 1 \mid ID(v) = i, \mid N_1(v) \mid = n)$$

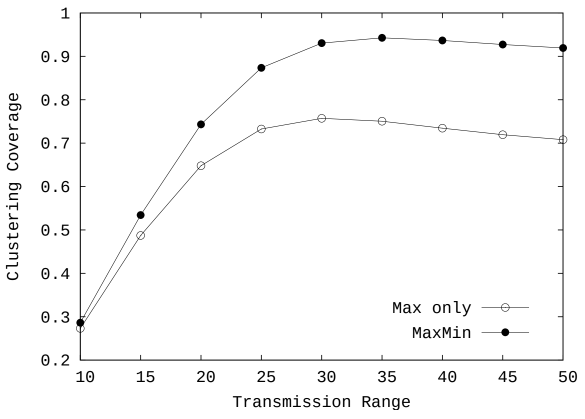
Note that the last term of Eq. (1) is the conditional probability that v will be a CH or CM given that v has n neighbors and ID ordinal number i. To evaluate the constituent probability that v, which has an ID ordinal number i and n neighbors, will be a CH, we translate the probability into the summation of the probability that all the nodes with IDs ranked greater than i will locate outside $N_1(v)$ and the probability that all the nodes with IDs ranked smaller than i will locate outside $N_1(v)$. We then evaluate the probability that v will be a CM via computing the complement probability, i.e., the probability that none of the nodes in $N_1(v)$ will be qualified to be a CH (so that v will not be a CM).

4 Discussion of Evaluation Results

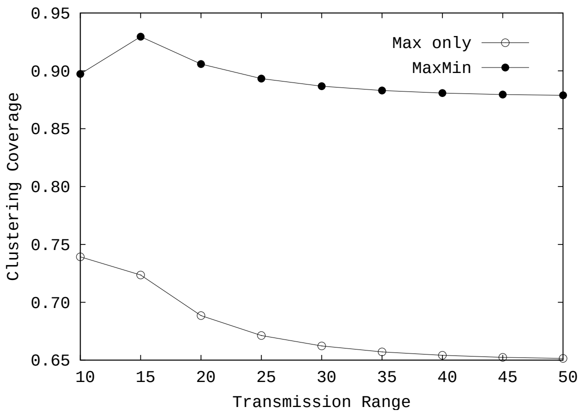
Based on the analytic model described in Sec. 3 and using Mathematica, we evaluate SCP's performability measure $P(C_2)$, as well as $P(C_1)$, which is the same measure but for the baseline Max-ID-based clustering protocol.

Fig. 3 displays the results of $P(C_2)$ and $P(C_1)$ computed as a function of node transmission range r. In the first scenario (Fig. 3(a)) in which 25 nodes are uniformly distributed in a 150×150 square field, we see that when the node transmission range r is small, SCP offers appreciable but limited coverage improvement because many nodes would not have any 1-hop neighbors so that organizing them is beyond the ability of any clustering mechanisms. But when r increases, which means node v is likely to have more 1-hop neighbors, SCP performs clearly better than the Max-ID protocol. Nonetheless, SCP and the Max-ID protocol have slightly dropped coverages after r reaches 35 and 30, respectively. This is due to the fact that a larger population within a unit disk makes it more competitive for a node's ID to be ranked as the maximum or minimum. As a result, the likelihood that v will be clustered decreases.

Fig. 3(b) offers us a related observation. The results displayed there are from an evaluation in which we assume a 100×100 field which accommodate 80 nodes. We see that for such a denser node population, the performability of the Max-ID policy is a monotonic decreasing function of r, while the MaxMin-ID SCP has its peak performance at $r = 15$ and its coverage stays reasonably high and stable throughout the (increasing) range of r.



(a)



(b)

Figure 3: Clustering Coverage as a Function of r

5 Concluding Remarks

We have developed a clustering protocol which simultaneously applies two different CH qualifying policies. Our initial study shows that superimposed clustering yields a significantly better coverage with low performance and energy costs, relative to the traditional clustering protocols that are iterative in nature and the adaptive/hybrid protocols that conditionally apply two policies in sequence. This effort is worthwhile due to at least two reasons: 1) we exploit parallelism and diversity in a novel fashion, and 2) the SCP framework provides mobile hosts with low-cost self-organizing capability which is the key to self-healing MANETs.

Specifically, while traditional clustering protocols allow all the nodes execute a clustering algorithm in parallel, we go beyond that by letting two different layers of clusters be formed simultaneously to compose a significantly better coverage. Indeed a combination of Max-ID (or Min-ID) and maximum node-degree or other judicious choices of policy combination will likewise yield no or minimal performance costs. Moreover, diversity is traditionally applied to validate computation correctness based on the convergence or majority agreement of computation results. In contrast, SCP applies diversified clustering policies to achieve better clustering coverage by taking advantage of result diversity.

Finally, while design diversity is often expensive with respect to the costs of development, maintenance, and performance, the combined use of diversified policies that are appropriately chosen will be always affordable. With the MaxMin-ID-based SCP described in this paper, the parallel application of the two clustering policies does not require any additional message exchange. Moreover, the activities related to the parallel use of clustering policies (e.g., redundancy checking and pruning) are all conducted locally by individual hosts, requiring neither intra- nor inter-cluster communication. And since message exchange is almost always the major drive of performance overhead and energy consumption in MANETs, the superimposed clustering approach can be justified by both its efficiency and affordability.

References

[1] L. Klienrock, "An Internet vision: the invisible global infrastructure," *Ad Hoc Networks*, vol. 1, pp. 3–11, July 2003.

[2] W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *IEEE Trans. Mobile Computing*, vol. 1, pp. 111–122, Apr. 2002.