

Fig. 5. The network nodes are partitioned into cluster c_1 , c_2 and c_3 . (a) The nodes $\{v_2, v_3, v_4, v_5\}$ are the border nodes of cluster c_1 after Step II of Section III-B3b. (b) Node v_1 replaces v_2 and v_3 as the border node.

Algorithm 2: Reduce Redundant Border Nodes

```

1 for Each cluster  $c_i$  do
2    $\lfloor$  Set cluster weight as  $1/(|b_i \cup \beta_i|)$ .
3   Calculate MVC on cluster-level topology.
4   if Border node in non-VC cluster then
5      $\lfloor$  Change to the cluster-ID of neighbor VC cluster.
6   Calculate MVC on  $b_i^m \cup \delta_i^m$  as  $\lambda_i$ .
7   Select  $\text{Min}\{|b_i^m|, |\lambda_i|\}$  as the border nodes of  $c_i^m$ .

```

c_2 are $\{v_3, v_5, v_8, v_9\}$, and the cluster border nodes between c_1 and c_3 are $\{v_2, v_3, v_4, v_6, v_7\}$. Secondly, the intersection area between c_1^2 and c_3^1 becomes cluster c_1 with border nodes $\{v_2, v_3, v_4, v_5\}$. Although we select the minimum number of border nodes for c_1^2 and c_3^1 , v_1 can replace v_2 and v_3 as the border node of c_1 as shown in Fig. 5(b) to further decrease the total number of border nodes. We formalize the approach to reduce redundant border nodes as follows. The main operation flow is shown in Alg. 2.

(i) We reset the cluster-ID of all the border nodes selected in Section III-B3b to a minimum number of clusters. We convert it to the Minimum Vertex Cover (MVC) problem [12] as follows. In the first place, we abstract the clusters into a cluster-level topology as shown in Fig. 6, where each cluster-level node represents a cluster. If there exists edges between two clusters as in Fig. 6(a), we connect the two cluster-level nodes. Next, we set the weight of each node in the cluster-level topology. The border nodes in cluster c_i are b_i after Alg. 1, and we call all the other border nodes that have edge connections with cluster c_i as β_i . The nodes set $b_i \cup \beta_i$ represents the maximum set of border nodes in c_i if re-categorizing the cluster-ID of β_i . To concentrate more border nodes in fewer clusters using MVC, we set weight value to each cluster. If the number of all the possible border nodes $|b_i \cup \beta_i|$ is high, we set a low weight value to the cluster c_i . In the implementation, we set the weight of cluster c_i to $1/(|b_i \cup \beta_i|)$. After that, we run the MVC algorithm on the cluster-level topology. If a border node belongs to a non-VC cluster, it changes its cluster-ID to the neighbor VC cluster. To differentiate with the notations before this step, c_i changes to c_i^m after re-categorizing the border nodes, and b_i changes to b_i^m .

(ii) Name δ_i^m as the subset of $c_i^m - b_i^m$, in which each node

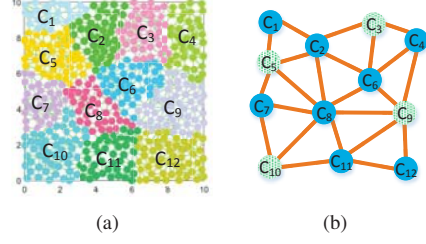


Fig. 6. (a) The network nodes are partitioned into clusters with different colors. (b) The clusters are abstracted into a cluster-level topology. The nodes in solid blue are vertex cover clusters.

has at least a neighbor in b_i^m . To simplify the analysis, we suppose there are not sink nodes at b_i^m or δ_i^m . The nodes in b_i^m and δ_i^m are not in cluster-contained-subnets. Name $\Phi_i = b_i^m \cup \delta_i^m$. Fig. 5(a) illustrates the example Φ_1 of c_1 . Then we calculate MVC on Φ_i as λ_i . We use λ_i as alternative border nodes to b_i^m . Because each edge in Φ_i has at least one endpoint in the MVC nodes λ_i , so monitoring λ_i can capture all the flows passing over Φ_i . The number of λ_i is not necessarily smaller than b_i^m . Therefore, we select $\text{Min}\{|b_i^m|, |\lambda_i|\}$ as the new border nodes of c_i^m . If λ_i are selected as the border nodes of cluster c_i^m , the non-VC nodes in b_i^m do not need to monitor the flow, and their cluster-ID are set to the neighbor cluster.

C. Protocol for Cluster based SD-WSN

CluFlow makes the SDN controller estimate flow among clusters by monitoring the flow at border nodes. The SDN controller controls traffic flow by injecting cluster-level routing rules to the border nodes. The management procedure of the SDN controller and nodes in WSN is shown in Alg. 3.

There are at least two benefits of utilizing SDN control in cluster-level routing. Firstly, compared with SDN management for every node in WSN, CluFlow trades granularity of SDN control for less communication load. Only cluster border nodes communicate with the SDN controller. The number of nodes that communicate with the SDN controller decreases. Secondly, cluster-level routing and local routing are decoupled. The nodes inside the clusters use only distributed local routing and do not need to request flow table entries from the SDN controller. The communication delay caused by requesting flow table entries therefore decreases.

IV. EXPERIMENTAL SETUP AND RESULTS

In this section, we test and evaluate CluFlow in simulation and a real deployed WSN.

A. Benchmark Approaches

To evaluate the performance of CluFlow, three benchmark approaches are implemented to calculate the communication flow among clusters.