A LIGHTWEIGHT CLUSTERING ALGORITHM UTILIZING CAPACITY HETEROGENEITY

Nicklas Beijar[1], Raimo Kantola[1], and Jose Costa-Requena[2]
[1]Helsinki University of Technology, Espoo, Finland, {Nicklas.Beijar,Raimo.Kantola}@hut.fi;
[2]Nokia Mobile Phones, Helsinki, Finland. Jose.Costa-Requena@nokia.com

Abstract: This paper describes an algorithm for clustering an ad hoc network, in which the devices have highly varying resources. Such a heterogeneous ad hoc network is formed when wireless and fixed consumer devices like laptops, personal data assistants, cellular phones and servers automatically interconnect. The aim of clustering is twofold: to reduce broadcast traffic, which is typical to ad hoc routing protocols, and to concentrate traffic to devices with more available resources and less mobility. Further, the clustering process identifies nodes that are suitable for service provision. The algorithm is lightweight as it operates with a single periodically sent message. We provide simulation results that show the performance of the algorithm.

Key words: clustering; mobile ad hoc network; virtual backbone

1. INTRODUCTION

Ad hoc networks are wireless networks established on temporary basis between a set of mobile nodes. The key feature of an ad hoc network is that there is no infrastructure; hence, there are no wired links and other fixed equipment. Data transmission is based on multi-hop routing, i.e. nodes act as routers by forwarding traffic of other source-destination pairs. Numerous routing protocols have been proposed for ad hoc networks. These can be divided into proactive and reactive protocols. Both approaches utilize flooding for distributing topological information: proactive protocols distribute information about topology changes by broadcasting to all nodes, and reactive protocols broadcast route requests. Several papers[1-5] have shown that the flooding traffic can be reduced by minimizing the number of nodes participating in broadcasting. This can be achieved with clustering or with a virtual backbone spanning the network. The control information is then only forwarded by selected nodes, such as cluster-heads and gateway nodes, or nodes belonging to the virtual backbone.

In scenarios including consumer devices, it can be assumed that a wide range of different types of devices are part of the ad hoc network. The devices have different amounts of available resources, e.g. memory, battery and processing power, and they have different mobility patterns. In this paper, we assume an ad hoc network consisting of commercially available devices of widely different type. Roughly, we can group the devices into two main categories: (i) highly mobile devices with low resources, such as personal data assistants (PDAs) and mobile phones, and (ii) devices with low mobility and high capacity, such as laptops, servers and access points.

Information about the available resources of a device can be reduced into a *preference value* using a *preference function*. Fundamentally, the preference function calculates the preference value by summing the weighted values of battery power, available memory, CPU resources, administrative preference, etc. Both control traffic and packet transmission should prefer a path consisting of nodes with a high amount of available resources. The preference function should additionally observe the level of mobility. Low mobility increases the preference value, while high mobility decreases it.

This paper presents an algorithm for clustering a network and forming a spanning virtual backbone by loosely interconnecting clusters. The aims of the algorithm are (i) to utilize the diversity in resources and mobility among nodes, (ii) to be very lightweight and adapt quickly to changes, and (iii) to provide a platform for service discovery. The protocol uses only a single Hello message for constructing the clusters and connecting them into a virtual backbone, consisting of nodes with relatively high preference.

2. NETWORK MODEL

We represent an ad hoc network using an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges in the graph. The distance $D(v,w)$ between two nodes v and w is the lowest number of edges on a path from v to w. The ith neighborhood $N_i(v)$ of a node v is the set of nodes $N_i(v) = \{w \mid w \in V, D(v,w) \leq i\}$. The ith deleted neighborhood $N'_i(v)$ of node v is the set of nodes $N'_i(v) = N_i(v) / \{v\}$. The nodes in the 1st deleted neighborhood $N'(v) = N'_1(v)$ are said to be the direct neighbors of v. We denote the number of direct neighbors with $|N'(v)|$ and call it the degree of the node. We assume that the nodes in the network share the same communication channel. Every node is assumed to have an identical omnidirectional radio transmitter and receiver, so that a message transmitted by a node is received by every node within a fixed radio-range with the radius Z. Links are assumed bi-directional. A node operating in promiscuous mode is able to receive transmissions directed to other nodes than itself provided that the sender is within radio-range. If the network is partitioned, the clustering algorithm operates separately in each partition, and these are automatically combined as the partitions unite.

3. OBJECTIVE

Each node has a *preference* value, $P(v)$, which indicates the node's suitability as a backbone node. The preference can be calculated based on the stability and resources of the node using a *preference function* used by all nodes in the network. The determination of the preference function is out of the scope of this paper. Generally, a stable node (whose neighbors do not frequently change) with low velocity and high resources should have a high preference value. The algorithm categorizes nodes into three colors, $C(v) \in \{white, green, black\}$, depending on the node's function in the network. *White* nodes are isolated nodes that have no neighbors within radio-range. *Black* nodes are the most stable and resource-rich nodes in the network. The aim is to keep the highest preference nodes as black nodes. *Green* nodes are nodes that have at least one black node as a neighbor. These have the lowest resources and their participation in message forwarding is minimized. The set of black nodes is denoted $V_{black} = \{v \mid v \in V, C(v) = black\}$. Correspondingly the sets of green, and white nodes are denoted V_{green}, and V_{white}, respectively. The colors are ordered so that black is higher than green, and green is higher than white. In addition to the color, a node can have an additional role: a black node can act as a *cluster-head* and both a green and a black node as a *bridge*.

Every node v has an unique address, which we denote $A(v)$. Further, every node v belongs to a *cluster*, $\pi(v)$, which can contain a variable number of nodes. The cluster is identified with the address of its *cluster-head* $\lambda(\pi)$, which is the node with the highest preference of the nodes in the cluster. The distance $hdist(v)$ from a node to its cluster-head is measured in hops. In addition to the color and the cluster, the algorithm assigns every node a *dominator*, $dom(v)$. The dominator can be any of the direct neighbors of the node, or the node itself: $dom(v) \in N_1(v)$.

When a node receives updated neighborhood information through a Hello message, it re-selects its dominator, its color and its cluster so that the following objectives are satisfied. Firstly, a node must always have a dominator, which is either a direct neighbor or the node itself. In the selection, the color order is the primary selection criteria and the preference the secondary selection criteria. Hence, if there are black nodes in the first neighborhood $N_1(v)$ of node v, then the black node w with the highest $P(w)$ is selected as $dom(v) = w$. If there are no black nodes in $N_1(v)$, the green node w in $N_1(v)$ with the highest $P(w)$ is selected. If there are no green nodes, the white node with the highest preference is selected. We denote this combination of color order and preference value as the (color, preference)-order. The reason for first observing the color is to improve stability of the backbone. The link between a node and its dominator is called a *dominator link*. The node w that has selected node v as its dominator ($dom(w) = v$) is called a *dominatee* of node v. A node can have zero or more dominatees but only one dominator.

The color of a node is determined by the neighboring nodes. A node v is white if it has no direct neighbors, i.e. $N'(v) = \varnothing$. If some direct neighbor w has selected v as its dominator ($dom(w) = v$, $w \in N_1(v)$), then the color of v is black. Thus, all dominators are black. Note that the node is also black if it has selected itself as its dominator. If v has neighbors, but none of them has selected v as its dominator, the color of v is green. Consequently, all dominators are black and other connected nodes are green. A node that has selected itself as its dominator takes its own address as its cluster identifier. It becomes the cluster-head of its cluster. In practice, this indicates that it has the highest (color, preference) value within its first neighborhood. Otherwise, the node joins the cluster of its dominator, and obtains the cluster identifier from the Hello message of the dominator.

The black nodes form a backbone within the cluster. Black nodes have the locally highest preference value. Since the preference value represents the resources and stability of the node, black nodes are suitable for distributing and storing information. The cluster-head can easily be located by following the path of dominator links. Hence sending a message to the local head can be implemented as the process of repeated forwarding of the message to the dominator until the head is reached. Because the cluster-head has the highest preference in the cluster, and because of the simple locating, it is suitable for storing persistent information related to the cluster.

The size of the cluster is determined probabilistically. The geographical size of a cluster is limited if the preference value is uncorrelated to the node's location in the network. However, with certain node placements it is possible to create a cluster encompassing all nodes in the network, and the cluster size is practically unlimited. Although these cases are improbable in a real scenario, we propose a method for limiting the cluster size: A node only considers a neighbor eligible as its dominator if the neighbor's distance to the cluster-head $hdist(v)$ is smaller than a given threshold, called the *cluster* *radius limit*, R_{max},. Otherwise, it chooses the dominator from the remaining nodes (including itself).

4. CONTROL MESSAGES

The algorithm uses a single control message. The *Hello Message* is periodically sent to all direct neighbors. The message sent by node v contains the fields $\{A(v), C(v), P(v), dom(v), \pi(v), hdist(v), dv(v)\}$. The fields $A(v)$, $C(v)$, $P(v)$, $dom(v)$, $\pi(v)$ are the address, color, preference, dominator and cluster identifier of the node, respectively. The field $hdist(v)$ is the distance from the node to its cluster-head. The field $dv(v)$ is a distance vector table, as will be described later.

Every node maintains a neighbor table N_v containing the information received in Hello messages from its direct neighbors. When a Hello message is received, the contents is inserted into N_v. Entries in the neighbor table time out if not refreshed after a specified time, whereas they are removed. Because a Hello message is required in most routing protocols, the clustering does not add message overhead, but only increases the Hello message size.

5. CLUSTERING ALGORITHM

When a node v powers on, it sets its color $C(v) = white$, its dominator $dom(v) = A(v)$ and its cluster $\pi(v) = A(v)$. It starts sending and receiving Hello messages. A node continuously listens to Hello messages received from neighboring nodes, and builds up and maintains the neighbor table N_v. When a Hello message is received, the information in the Hello message is added to or updated in the neighbor table. Additionally, the timer for the specific neighbor is restarted. If a Hello message has not been received within a specific timeout period, the entry is removed.

An *attribute determination procedure* is invoked (i) after each received Hello message and (ii) if a neighbor entry times out. The procedure has two phases. It first determines the dominator and cluster (phase 1), and then the color (phase 2) of a node. We provide two versions of the first phase.

When node v performs the determination procedure, it includes every node w of the neighbor table for which $hdist(w)$ is less than the cluster radius limitation (R_{max}) in its dominator-candidate list ($E(v)$). In our first version of this phase, also the node v itself is included:

$$E(v) = \{w \mid w \in N_1(v), hdist(w) < R_{\max}\}$$

In the second version, the node *v* is included only if it is the dominatee of some direct neighbor:

$$E(v) = \begin{cases} \{w \mid w \in N_1(v), hdist(w) < R_{\max}\} & , if\ \exists u \in N_1'(v) : v = dom(u) \\ \{w \mid w \in N_1'(v), hdist(w) < R_{\max}\} & , otherwise \end{cases}$$

The rest is identical for both versions. From the dominator-candidates, it selects the node *w* with the highest (color, preference) order as its dominator:

$$dom(v) = w : w \in E(v) \quad so\ that$$
$$\forall u \in E(v) : C(w) \geq C(u) \vee (C(w) = C(u) \wedge P(w) \geq P(u))$$

After that, it updates its cluster and head-distance attributes:

$$\pi(v) = \begin{cases} A(v) & ,if\ dom(v) = v \\ \pi(dom(v)) & ,if\ dom(v) \neq v \end{cases}$$
$$hdist(v) = \begin{cases} 0 & ,if\ dom(v) = v \\ hdist(dom(v)) + 1 & ,if\ dom(v) \neq v \end{cases}$$

The second phase checks the *dom*(*u*) of every direct neighbor *u* and updates the color *C*(*v*).

$$C(v) = \begin{cases} black, if\ \exists u : u \in N_1'(v) : dom(u) = v \\ green, if\ |N_1'(v)| > 0 \vee \forall u : u \in N_1'(v) : dom(u) \neq v \\ white, if\ |N_1'(v)| = 0 \end{cases}$$

6. CONNECTING CLUSTERS

The described algorithm connects the nodes within a cluster with a backbone. However, the backbone does not connect the clusters. The nodes connected by dominator links therefore form a forest consisting of one tree for each cluster. In this section we present an approach that obtains routes to neighboring clusters without permanently connecting them. Instead it connects a temporary *bridge* between two neighboring clusters.

A node learns through the Hello messages to which cluster each of its direct neighbors belongs. For neighbors in a different cluster than itself, the node also knows the distance to the respective cluster-head since the Hello message contains the *hdist* field. A routing table entry is created for each neighboring cluster and the node with the lowest cluster-head distance becomes the next hop to this cluster. The distance to the neighboring cluster is set as the cluster-head distance plus one. This favors routes leading close to the center of the cluster. In case of equal cluster-head distance, the next hop with the highest preference is selected. A distance vector with entries for all neighboring clusters is included in the Hello message.

The distance vector information is progressed toward the cluster-head in the *dv* field of the Hello messages. A node stores the distance vectors received from a dominatee in its routing table. For each cluster, it selects the neighbor with the lowest distance to the cluster as the next hop. In case of equal distance, the neighbor with the highest preference is selected. Because there is a strict hierarchy within the cluster, distance vector loops cannot appear. Eventually, the cluster-head has routing table entries to all neighboring clusters. A path is available from the cluster-head to each neighboring cluster since every intermediate node has a next hop entry to the given cluster. This mechanism enables sending messages between neighboring clusters through a cluster-head. The route follows the backbone, and consequently consists of nodes with high preference. There may be one or two green nodes acting as bridge nodes between two neighboring clusters.

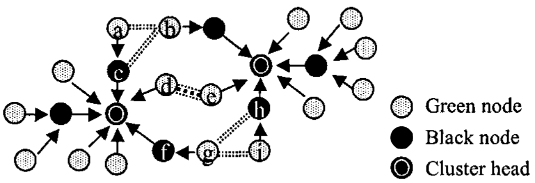

Figure 1. Example of connecting clusters with bridges

Figure 1 illustrates two neighboring clusters connected with bridges. Node *a* and node *c* both have selected node *b* as its bridge to the neighboring cluster, and node *b* has selected *c*. Node *d* and node *e* have selected each other. Node *g* has selected node *i*, while node *i* and node *h* both have selected node *g*. The cluster-heads have selected node *d* and node *e*.

7. SIMULATION

In order to observe the cluster formation and its performance, we performed a series of simulations. We based the simulation on a topology simulator[6] implemented in the Python[7] language. A Hello message is received error free by all nodes within the radio-range *Z* of the sender, and lost by all other nodes. Only network layer functionality is implemented, as we assume that the physical layer operates at a much faster time scale than the network layer. The Hello interval was 1 s and the Hello timeout was 2 s.

Initially the nodes are randomly distributed in the 1000 x 1000 m roaming area. The random waypoint mobility model[8] is used. A node selects a random destination and a speed between 1 m/s and a maximum speed, and moves towards the destination. When the destination is reached, the node pauses for a time randomly chosen between 0 s and 100 s. Five runs of 600 s each were performed for each set of parameters, and the results were averaged. In the simulations presented in this paper, the nodes were assigned a fixed randomly generated preference value.

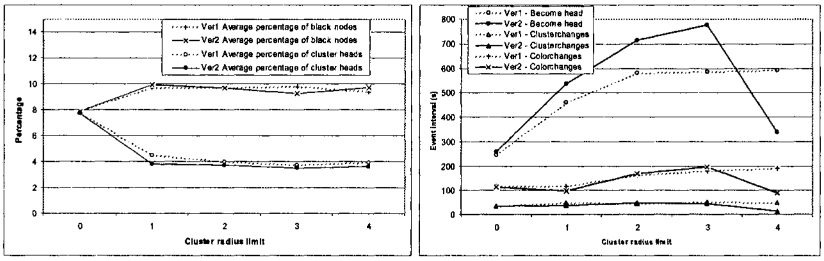

Figure 2. Influence of algorithm version and cluster radius limit

First we compared the two versions of the algorithm. The cluster radius limit parameter was varied, while the number of nodes was 100 and the radio-range was 300 m. The node speed was randomly chosen between 0 and 10 m/s (36 km/h). Figure 2a shows the average percentage of black nodes and cluster-heads for both versions of the algorithm. The number of cluster-heads is equal to the number of clusters since each cluster has exactly one cluster-head. The reason for using the second version is to reduce the number of redundant cluster-heads. As expected, the second version has fewer cluster-heads, but the difference is marginal. The number of black nodes is in practice similar for both versions. Figure 2b shows the average time between color changes, cluster changes and head-selection for the same simulation. A higher value represents more stability. The average interval $\overline{T}$ is calculated as $\overline{T} = (NT)/ N_e$, where N_e is the measured number of events, N is the number of nodes, and T is the simulation time. We see that the changes to become cluster-heads are more infrequent in the second version. The intervals of color and cluster changes are similar in both versions. Because of the ability to form larger clusters and the increased stability of cluster-heads, we will use the second version in further simulations.

In the simulation we varied the cluster radius limit parameter. Some applications and routing protocols may perform better in small clusters. We can see that the largest difference is between values 0 and 1 of the parameter.

When the parameter is 0, all black nodes are cluster-heads and the curves coincide. When the parameter is 1, the number of cluster-heads is halved but some more black nodes are required. Higher values of the parameter have minimal influence on the color distribution. However, stability increases until the maximum at the parameter value 3 and then dramatically decreases. A reason is that a node can keep the same dominator longer as the cluster radius limitation does not force it to change. After a certain distance, mobility causes long branches to break more frequently. We will use a cluster radius limit of one in further simulations.

The influence of node density can be tested by either varying the radio-range or the number of nodes in a fixed size network. As a measurement of density, we use the average number of nodes within the radio-range of a node, i.e. the average degree Δ. Denoting the number of nodes in the network with N, the roaming area size with A and the radio-range of a node with Z, the theoretical average degree Δ_t can be calculated as $\Delta_t = (N/A)\ \pi Z^2$.

Figure 3a shows the percentage of white, green and black nodes when the radio-range is increased from 50 m to 350 m. There are 50 nodes in the network of size 1000 x 1000 m. Because the radio-range Z is increased linearly, the average degree increases exponentially ($\Delta_t \sim Z^2$). The node speed is uniformly random between 0 and 20 m/s (72 km/h).

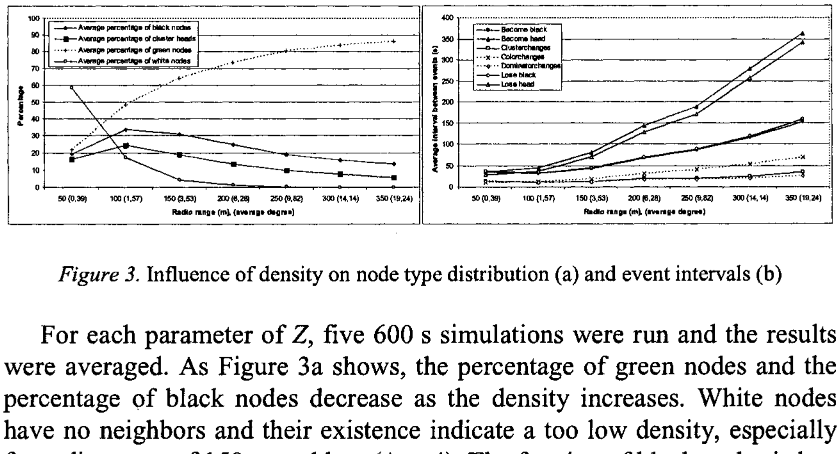

Figure 3. Influence of density on node type distribution (a) and event intervals (b)

For each parameter of Z, five 600 s simulations were run and the results were averaged. As Figure 3a shows, the percentage of green nodes and the percentage of black nodes decrease as the density increases. White nodes have no neighbors and their existence indicate a too low density, especially for radio-range of 150 m and less ($\Delta_t < 4$). The fraction of black nodes is less than 30% in most connected networks and less than 20% in dense ($\Delta_t > 10$) networks. In dense networks, about half of the black nodes are cluster-heads. Figure 3b shows the average interval between various events. We can see that nodes change their dominator at a rate that is relatively independent of the node density (15 – 25 s). Black nodes remain black long – in most connected networks over one minute despite the high mobility. Cluster-heads are stable, with a change interval of over 3 minutes in dense networks.

8. CONCLUSIONS

The simulations give a rough estimation about the performance of the clustering algorithm. Because of paper length limitations, only some simulations were presented. We can see that the algorithm is able to reduce the number of nodes that participate in broadcasting. By allowing an additional layer of black nodes in addition to the cluster head, the stability can be increased. The clustering algorithm has a low overhead with a single periodically sent Hello message. The proposed algorithm creates a clustering structure that respects the different resources of the nodes. It allows concentration of traffic and service provision to nodes with more resources.

ACKNOWLEDGEMENTS

This work has been partly supported by the European Union under the E-Next Project FP6-506869, and by the European Commission under the IST-2001-38113 MobileMAN project. The work has also been partly supported by the TES and Sonera foundations.

REFERENCES

1. B. Das, R. Sivakumar, and V. Bharghavan, Routing in ad hoc networks using a spine, in *Proceedings of Sixth International Conference on Computer Communications and Networks*, 1997, pp. 34-39, September 1997
2. R. Sivakumar, B. Das, and V. Bharghavan, The Clade Vertebrata: Spines and routing in ad hoc networks, in *Proceedings of the Third IEEE Symposium on Computers and Communications*, 1998, ISCC '98, pp. 599-605, June-July 1998
3. Jie Wu, and Hailan Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, in *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, August 1999
4. Stojmenovic, M. Seddigh, and J. Zunic, Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks, in *IEEE Transactions on Parallel and Distributed Systems*, Volume 13, Issue 1, pp. 14-25, January 2002
5. Srivastava, R. K. Ghosh, Cluster based routing using a k-tree core backbone for mobile ad hoc networks, in *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, September 2002
6. Simple Network Layer Simulator for Ad-Hoc Networks, http://www.netlab.hut.fi/tutkimus/MobileMan/snelsan/
7. http://www.python.org/
8. R. Giovanni, S. Paolo, An analysis of the node spatial distribution of the random waypoint mobility model for ad hoc networks, in *Proceedings of the second ACM international workshop on Principles of mobile computing*, pp. 44-50, October 30-31, 2002