International Journal of Wireless Information Networks, Vol. 9, No. 2, April 2002 (© 2002)

Fuzzy Rule-Based Mobility and Load Management for Self-Organizing Wireless Networks

Jörg Habetha[1] and Bernhard Walke[2]

Mobility management in a cluster-based, multihop ad hoc network is studied. It is shown that the process of clustering the network into groups of stations has similarities to data analysis, in particular, pattern recognition. In data analysis, the term *clustering* refers to the process of unsupervised learning, which also describes the situation in a mobile ad hoc network.

In this paper, existing data-clustering algorithms are first classified into different categories. Some of the most important types of algorithms are afterwards described, and their applicability to the problem of mobility management in an ad hoc network is studied. It is shown that most of the pattern-recognition algorithms are not suited to the application under consideration.

This is why we have developed a new clustering scheme that incorporates some of the ideas of the data classification schemes. The new clustering scheme is based on a rule-based fuzzy inference engine. The main idea consists of the consideration of dynamic clustering events chosen as a consequence of the fuzzy rules. Four types of clustering events are considered.

The performance of the clustering algorithm has been evaluated by computer simulation.

KEY WORDS: Ad hoc network; clustering mobility management; handover.

1. INTRODUCTION

Wireless networks can be divided into infrastructure-based and self-organizing networks. Self-organizing, so-called ad hoc networks were mainly used by the military in the past, but various other applications are foreseen today. Examples are *personal area networks* (PAN) for short-range communication of small user devices, *wireless local area networks* (WLAN) mostly for user and data communication, and *in-house digital networks* (IHDN) for audio, video, and data exchange. The initial communication standards with ad hoc capability have already been completed: Bluetooth, wireless PAN, IEEE802.11a, WLAN and HIPERLAN/2, WLAN and IHDN.

The size of the area covered by ad hoc networks is in general much bigger than the transmission range of the stations. Communication between two stations therefore involves several other stations, which have to forward the data. This means that ad hoc communication results in multihop networks, whereas infrastructure-based communication, typically, uses only one radio hop.

Two classes of ad hoc networks can be distinguished: *decentralized* and *cluster-based* ad hoc networks. In decentralized ad hoc networks, the access scheme as well as the network management is completely decentralized. An example of such a network is the *distributed coordination function* (DCF) of the IEEE 802.11 system. Advantages of decentralized systems are

[1] Philips Research, Weisshausstrasse 2, D-52066 Aachen, Germany. E-mail: joerg.habetha@philips.com

[2] Aachen University of Technology, Chair of Communication Networks, Kopernikusstr. 16, D-52074 Aachen, Germany.

1068-9605/02/0400-0119/0 © 2002 Plenum Publishing Corporation

their relatively low complexity and their robustness against failures. In cluster-based networks, certain functions like the *medium access control* (MAC) or the *routing* are performed by one specific station per cluster, the so-called *cluster head* or *central controller* (CC). These functions do not necessarily have to be carried by the same station all the time. The functions can, of course, be handed over to another station in the same cluster that is able to carry them. Centralized networks ease the provision of quality of service (because centralized polling schemes can be applied), and they allow for a possible reuse of infrastructure-oriented protocols and equipment. Especially due to this last characteristic, cluster-based network architectures are interesting candidates for hybrid networks, in which large parts of the network rely on an installed infrastructure and where ad hoc networks are automatically set up if no preinstalled infrastructure is available.

This paper is concerned with cluster-based networks and especially the mobility management of the stations. Mobility management basically comes down to building and administering clusters of wireless nodes. Several *clustering* algorithms have been proposed so far. Two of the first algorithms proposed have been the *lowest ID* and the *highest connectivity* algorithms [1,2]. In the *lowest ID* (LID) algorithm, all stations have a networkwide unique identifier (ID), which is periodically broadcast by each station and received by the direct neighbors. Each terminal compares the IDs of its neighbors with its own ID and decides to become a CC if its own ID is lower than all other IDs received. As the name implies, the *highest connectivity* algorithm is based on the connectivity, i.e., the number of direct neighbors, of a node. As in the lowest ID algorithm, each terminal periodically broadcasts its own connectivity value to its direct neighbors. The neighbors will compare their own connectivity with the connectivity of all neighboring terminals and become a CC if their own connectivity is higher than all other values received.

Several extensions of these basic algorithms have been proposed. We have ourselves proposed an extension of the LID algorithm, which we have called the *highest ID* algorithm in order to distinguish it from the LID. The aim of the modified algorithm is to take capacity restrictions inside the clusters into account and to open additional clusters not only if another terminal with a lower ID is detected but also if the capacity of a cluster is exhausted [3]. Another algorithm, which could be considered as an extension of the highest connectivity algorithm is our *lowest distance value*, rsp. *highest received signal strength (RSS)* algorithm [4,5]. The idea is to not only take into account the number of neighboring terminals but also the distance to these neighbors. The terminal which has the lowest average distance becomes the CC. Instead of the average distance, the total traffic of a terminal could also be chosen as clustering criterion. With such a decision value it can be expected that the traffic forwarded among the clusters is minimized [4,5].

Other algorithms have been proposed that, for example, take into account the mobility of the stations [6] or the influence of power control on the size of the clusters [7].

To our knowledge, in none of the previous studies has the similarity of the clustering in an ad hoc network to the classification of data objects in the framework of pattern recognition been analyzed. In the area of data analysis, a huge amount of algorithmic solutions to the clustering problem have already been developed. It is the aim of this work to review the available material, to assess its suitability for ad hoc networking, and, finally, to extract some ideas in order to generate a clustering algorithm for mobility management in self-organizing wireless networks.

The remainder of the paper is organized as follows. In Section 2 the considered cluster-based system architecture and a possible realization of the lower protocol layers are described. An overview of algorithms used in the framework of data analysis and classification is given in Section 3. Based on the results of the analysis of the existing algorithms, their suitability for mobility management in a wireless ad hoc network is assessed in Section 4. As none of the algorithms perfectly matches the requirements, a new clustering algorithm is developed next. A performance evaluation of the new clustering scheme is carried out in Section 5. This section also contains a description of the assumed mobility models, routing algorithms, and the simulation scenario. The paper concludes with a summary and an outlook on open research issues.

2. SYSTEM ARCHITECTURE AND LOWER PROTOCOL LAYERS

In this section, we describe one possible realization of a cluster-based ad hoc network on the physical as well as DLC layer. However, it has to be noted that the mobility management algorithms presented in the remainder of the paper are independent of the concrete physical and DLC layer characteristics and apply to the general class of cluster-based systems.

In [5,8] we have presented the concept of a cluster-based multihop ad hoc network based on *frequency division multiplexing* (FDM) among clusters and *time division multiple access* (TDMA) inside the clusters. In the following two sections the network architecture as well as some details of the physical and *data link control* (DLC) layer will be described.

2.1. System Architecture

The system is organized according to the *link-clustered architecture* [9,10], which foresees the formation of clusters of terminals for contention-free operation inside the clusters. In contrast to the system presented in [1,2], where clusters are separated by orthogonal spreading codes, each cluster operates on a different frequency in the system considered. In each cluster one station, called the *central controller* (CC), generates MAC frames and allocates transmission slots to all terminals in its cluster. The clusters are interconnected on the MAC level by so-called *forwarding terminals* (FT), which are located in the overlapping zones of the clusters and participate in the communication of two clusters. This is different from the general link-clustered architecture, where clusters do not necessarily have to overlap and where two terminals can form a so-called *distributed gateway.* In our system, however, interconnection of clusters is achieved by one single FT. This is illustrated in Fig. 1.

Because each cluster operates on a different frequency, the FTs have to switch from one frequency to another and can be present in only one cluster at a time. This mechanism is illustrated in Fig. 2, where the two upper rows of rectangles represent the MAC frame structure in two different clusters and the lowest row the presence times of the FT in cluster 1 and 2, respectively, on frequency f1 and f2. It can be seen that the MAC frames in the two clusters are, in general, not synchronized. Consequently, the FT is not only absent during the frequency switching time T_S but loses also waiting time T_W until the beginning of the next MAC frame.

Mechanisms to improve the efficiency of the forwarding mechanism (e.g., by means of double-transceivers) are discussed, for example, in [11].

A very important characteristic of the concept described is the possibility of reusing existing protocols developed for infrastructure-based systems. Therefore, the system architecture presented is compatible with existing or currently developed WLAN standards for the 5 GHz band.

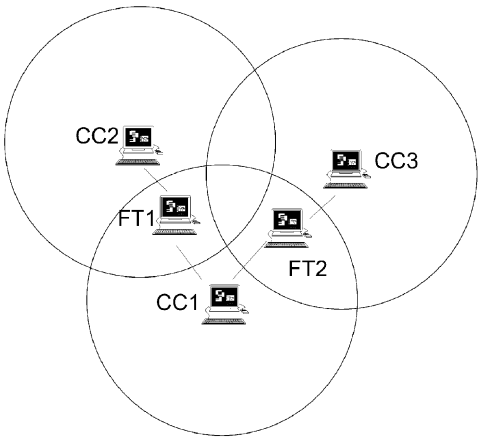


Fig. 1. Cluster-based networking concept.

2.2. Physical and DLC Layer

For our example system we assume operation in the unlicensed 5 GHz band and conformance to the harmonized physical layer of the IEEE 802.11a and the ETSI HiperLAN/2 (HL/2) systems. This physical layer is based on *orthogonal frequency division multiplexing* (OFDM) with 52 subcarriers. Each subcarrier can be modulated with four different modulation schemes (BPSK, QPSK, 16QAM, and 64QAM). Forward error correction is achieved with a convolutional code with code rate 1/2 and constraint length 7. Different code rates (1/2, 9/16, and 3/4) are achieved by the application of puncturing schemes. A combination of a modulation scheme and code rate is called a *PHY-mode*. With the highest PHY-mode (64QAM3/4), a data rate of 54 Mbit/s is achieved.

On the DLC layer either the centralized mode of operation of the 802.11 system, called *point coordination function* (PCF) or *hybrid coordination function* (the latter is currently discussed in the IEEE 802.11e subgroup), or the DLC protocol of the HIPERLAN/2 system could be applied. Taking the example of the HIPERLAN/2 protocol, a CC generates MAC frames with a duration of 2 ms, in which time division multiple access (TDMA) is employed. Short slots (9 bytes payload) are used for the

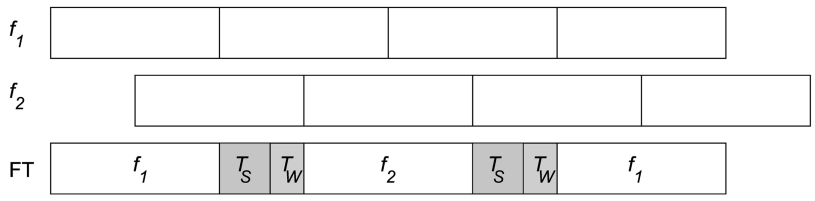


Fig. 2. Absence times of the forwarding terminal.

resource request and ARQ feedback messages of the terminals. Long slots (48 bytes payload) are used for the transmission of user data. The usage of a slot is granted by the CC on the basis of the resource needs of the terminals. More details regarding the IEEE 802.11 and HIPERLAN/2 DLC protocols and a performance evaluation can be found, for example, in [12,13].

As mentioned earlier, the clustering algorithms are also applicable to a variety of other physical and DLC layer protocols.

3. DATA ANALYSIS AND CLUSTERING

The task of grouping terminals into clusters according to certain criteria is similar to what is usually done in the framework of data analysis. The most important scientific means for automatic data analysis is the *classification* of data, also known as *pattern recognition.* The term *data analysis,* with respect to pattern recognition, describes the process of searching the structure in a given set of data [14]. The most important steps in this process are the identification of characteristic attributes of the data objects and the devising of a mechanism to partition the data objects into a number of subgroups according to these attributes.

Commonly, a distinction is made between supervised and unsupervised pattern recognition. Supervised pattern recognition deals with the classification of data objects in the case that the membership values of a set of given objects to a certain number of classes is known. Supervised pattern recognition is therefore mainly concerned with the development of a suitable mechanism (called the *classificator*) in order to classify new objects in the future according to the past data.

In the case of unsupervised pattern recognition, the membership values of data objects to classes are not known in advance. In addition, the number of classes is also generally unknown. According to a commonly used principle, classes of data are formed in such a way that objects inside a class have a high degree of similarity, whereas objects of different classes should be as different as possible. Unsupervised pattern recognition is also known as *clustering,* and the classes are called *clusters* in that case. We will only consider unsupervised clustering algorithms, because the task of forming groups of terminals in a network corresponds to a situation in which no classes are defined a priori.

3.1. Classification of Clustering Algorithms

Classification algorithms can be separated into syntactic (cf. [15,16]) and decision-theoretic algorithms. In this section, we will classify the decision-theoretic algorithms according to the following criteria:

- The underlying mathematical theory
- The characteristics of the algorithm itself
- The behavior of the system, with respect to data objects, in time

In Fig. 3 the three-dimensional classification of the algorithms is illustrated.

Regarding the underlying mathematical theory, we distinguish between *crisp* and *fuzzy* algorithms. The fuzziness can either refer to the attributes of the data objects or to the definition of the data subgroups with respect to *classes* or *clusters.*

Four types of algorithms can be distinguished:

- Graph-theoretic algorithms
- Iterative algorithms
- Knowledge-based algrithms
- Neural networks

Most of the algorithms exist in crisp or fuzzy formulation. In the following, we will consider the iterative, i.e., objective-function based, and the knowledge-based algorithms in fuzzy formulation (as shaded in gray in Fig. 3).

As far as the time behavior of the system, i.e., data set, is concerned, it can be either static or dynamic. Most of the clustering algorithms consider the situation at a given point in time and have to be referred to as static algorithms. Only recently have dynamic clustering algorithms been developed that take the dynamic character of the system, i.e., data objects, into account.

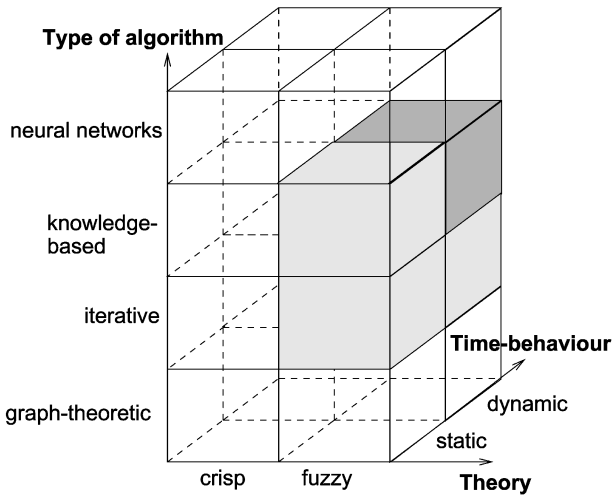


Fig. 3. Classification of pattern recognition algorithms.

3.2. Fuzzy-Set Theory

Before some important fuzzy clustering algorithms are described, we will give in this section a brief overview of the basic fuzzy-set theoretic concepts.

The concept of a fuzzy set was introduced by L. A. Zadeh in 1965 [17]. A fuzzy set is characterized by the fact that its members belong to the set only to a certain degree, called the membership value. The classic membership values of 0 or 1 are generalized to any real number. In most cases, the membership values are normalized to the interval [0,1].

Definition 1 Let X be a set of objects x. A **fuzzy set** $\tilde{S}$ over the set X is the set of all pairs:

$$\tilde{S} = \{(x, \mu_{\tilde{S}}(x)) \mid x \in X\}$$

$\mu_{\tilde{S}}(x)$ represents the degree of membership of element x to the set $\tilde{S}$ and can assume values in [0,1], whereby 1 represents the maximum degree of membership. For arbitrary x we call $\mu_{\tilde{S}}(x)$ the membership function. In Fig. 4, three commonly used membership functions are shown. These functions can be interpreted as "the set of all x roughly equal to x_0."

Similar to classic set theory, set-theoretic operations can be defined on fuzzy sets. Zadeh has defined the following operators for *complement*, *union* and *intersection:*

Definition 2 The **complement** $\tilde{S}'$ of a fuzzy set $\tilde{S} = \{(x, \mu_{\tilde{S}}(x)) \mid x \in X\}$ is defined by the following membership function:

$$\mu_{\tilde{S}'} = 1 - \mu_{\tilde{S}}$$

Definition 3 The **intersection** $\tilde{A} \cap \tilde{B}$ of two fuzzy sets $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$ and $\tilde{B} = \{(x, \mu_{\tilde{B}}(x)) \mid x \in X\}$ is defined by the following membership function:

$$\mu_{\tilde{A} \cap \mu_{\tilde{B}}} = \min\{\mu_{\tilde{A}}, \mu_{\tilde{B}}\}$$

Definition 4 The **union** $\tilde{A} \cup \tilde{B}$ of two fuzzy sets $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$ and $\tilde{B} = \{(x, \mu_{\tilde{B}}(x)) \mid x \in X\}$ is defined by the following membership function:

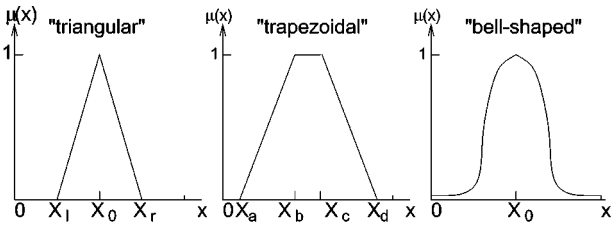


Fig. 4. Membership functions.

$$\mu_{\tilde{A} \cup \tilde{B}} = \max\{\mu_{\tilde{A}}, \mu_{\tilde{B}}\}$$

Other set-theoretic operators have been proposed. The properties of intersection as well as union operators have been defined axiomatically. Operators that fullfil the properties of intersection operators are called *T-norms* (or *triangular norms*). Union operators are part of the class of *S-norms* (or *T-conorms*) [18,19].

3.3. Iterative Algorithms

The class of algorithms considered in this section partition a given set of data objects into c clusters in such a way that an objective function is optimized. It has to be distinguished between a so-called *hard-c partition* and a *fuzzy-c partition* [14]. A hard-c partition consists of disjunctive clusters and membership values of 0 or 1, whereas a fuzzy-c partition can contain overlapping clusters with real-valued membership values μ_{ik} of an object k to a cluster i. The objective function of most of the algorithms has the general form

$$z(\mathbf{U}, \mathbf{v}) = \sum_{k=1}^{n} \sum_{i=1}^{c} \mu_{ik}^{m} \cdot d^2(\mathbf{x}_k, i) \qquad (1)$$

$d(\mathbf{x}_k, i)$ is a distance function between object $\mathbf{x}_k$ and cluster i; m, with $m > 1$, is the so-called *fuzzification* parameter.

The objective function is minimized and therefore represents a generalized form of the *mean square error* technique, commonly used in optimization theory. Instead of $d(\mathbf{x}_k, i)$, mostly $d(\mathbf{x}_k, \mathbf{v}_i)$ is used, where $\mathbf{v}_i$ is the center of cluster i. This is why the algorithms are sometimes also called *prototype-based algorithms*, as the *cluster center* (CC) $\mathbf{v}_i$ can be interpreted as a virtual data object, a prototype of the cluster. In our example system, the *central controller* can be considered as the *cluster center* and is designated by the same acronym, CC.

The optimization of the objective function results in a mathematical expression for the membership values μ_{ik} with given cluster centers $\mathbf{v}_i$. However, the cluster centers are not known a priori. Therefore, the considered algorithms determine the cluster centers based on the membership values of all objects. By iterating this approach, an optimal combination of cluster centers $\mathbf{v}_i$ and membership values μ_{ik} is derived.

Different clustering algorithms can be constructed by a specific choice of the CC expression and distance measure.

The first iterative fuzzy clustering algorithm was the so-called *fuzzy-c means* algorithm [20]. The algorithm is a generalization of the so-called ISODATA algorithm (also *hard-c means* algorithm) [21,22]. With the fuzzy-*c* means, the cluster centers are determined as "centers of gravity" of the given set of data with respect to each of the classes:

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n} \mu_{ik}^m \boldsymbol{x}_k}{\sum_{k=1}^{n} \mu_{ik}^m} \tag{2}$$

As the distance measure, the euclidic norm is chosen:

$$d(\boldsymbol{x}_k, \mathbf{v}_i) = \| \boldsymbol{x}_k - \mathbf{v}_i \| = \sum_{p=1}^{P} (x_{kp} - v_{ip})^2 \tag{3}$$

where p is the index of the attribute coordinate.

It can be proven (e.g., by Lagrange multiplicators), that the objective function 1 subject to the constraint $\sum_{i=1}^{c} \mu_{ik} = 1$ is minimized by the following choice of the values μ_{ik} [23]:

$$\mu_{ik} = \begin{cases} \dfrac{1}{\sum_{j=1}^{n} \left(\dfrac{d^2(\boldsymbol{x}_k, \mathbf{v}_i)}{d^2(\boldsymbol{x}_k, \mathbf{v}_j)} \right)^{\frac{1}{m-1}}} & \text{if } d(\boldsymbol{x}_k, \mathbf{v}_j) \neq 0 \\ \forall j = 1, \ldots, c & \\ 1 & \text{if } d(\boldsymbol{x}_k, \mathbf{v}_i) = 0 \\ 0 & \text{if } \exists\, d(\boldsymbol{x}_k, \mathbf{v}_j) = 0 \\ & \text{with } j \neq i \end{cases} \tag{4}$$

The following iteration is carried out:

1. Initialization of the membership values $\mu_{ik}^{(0)}$ (e.g., with random values)
2. Calculation of the cluster centers $\mathbf{v}_i^{(l)}$ according to Eq. (2) for all clusters $i = 1, \ldots, c$
3. Calculation of new membership values $\mu_{ik}^{(l+1)}$ for all data objects $\boldsymbol{x}_k$ and clusters i according to Eq. (4)
4. Check of stop criterion: $\| \mu_{ik}^{(l+1)} - \mu_{ik}^{(l)} \| \leq \epsilon$. If not fulfilled, goto Step 2.

Most of the other algorithms use a similar iteration mechanism but differ in the distance measure and/or CC formula (e.g., [24,25]).

3.4. Knowledge-Based Algorithms

Another class of clustering algorithms, the knowledge-based algorithms, use expert knowledge in order to come to a clustering decision. The knowledge is stored in an *expert system*, which also contains an *inference engine* where the clustering logic is carried out. There are many different ways of knowledge representation, such as *semantic networks* (e.g., *Petri nets* [26]), *frames* [27], and *rules*. Rule-based logic is probably the most commonly used and will be considered in the following.

Classic expert systems use dual or multivalued logic in the inference engine. There is, however, no principal difference between dual and multivalued logic. A principal difference exists compared to the so-called *fuzzy logic* developed by L. A. Zadeh, which significantly improves the inference capability of an expert system. Fuzzy logic is an extension of the fuzzy set theory described in Section 3.2. The same operators as mentioned in Section 3.2 are used as logic operators.

Of special importance in fuzzy logic is the concept of a *linguistic variable:*

Definition 5 (cf. [28]) A **linguistic variable** x is characterized by a set of linguistic terms $T(x)$. Each linguistic term corresponds to a fuzzy set $\tilde{M}(u)$ over the base variable u with values in U. The base variable is the same for all linguistic terms. For each value in U, the membership function $\mu_{\tilde{M}}(u)$ determines the degree to which a value of u corresponds to the linguistic term T.

As an example, consider the linguistic variable "Truth," characterized by the linguistic terms "true," "false," and "undecided," with possible membership functions as illustrated in Fig. 5.

Taking into account the definition of the *implication* in dual logic, a fuzzy inference operator (i.e., *implication*) could be defined as follows:

$$A \rightarrow B = (\neg A) \vee B = \max\{1 - \mu(A), \mu(B)\} \tag{5}$$

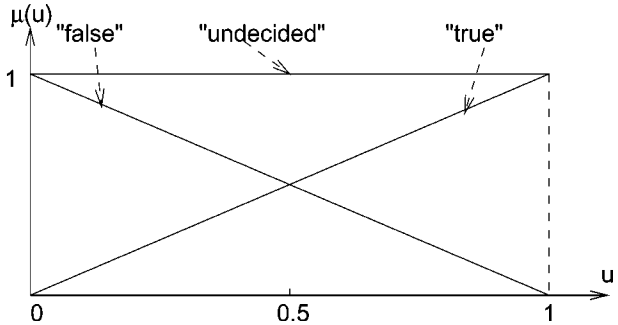


Fig. 5. Membership functions of the linguistic variable "Truth."

where $\mu(A)$ rsp. $\mu(B)$ are the membership functions of the (linguistic) truth values of two statements A and B. Many other fuzzy implication operators have been proposed.

A fuzzy rule has the general form

IF "x is $\tilde{A}$" THEN "y is $\tilde{B}$"

Fuzzy rule-based (plausible) inference, also known as *generalized modus ponens*, can be summarized as

Fact:	x is $\tilde{A}$
Rule:	IF "x is $\tilde{A}$" THEN "y is $\tilde{B}$"
Inference:	y is $\tilde{B}$

With rule-based data analysis, the real-valued attributes of the data objects are transformed into linguistic variables by defining interval terms like "very small," "small," "medium," "big," and "very big." The attribute coordinates will be designated with x_p ($p = 1, \ldots, P$) in the following. A linguistic term of the feature x_p is characterized by a membership function $\tilde{A}_{p,j}$. An attribute x_p can take J_p ($j = 1, \ldots, J_p$) linguistic values.

The classification is carried out according to rules of the form

$$R_r\text{: IF } x_1 \text{ is } \tilde{A}_{1,j(r)} \text{ AND } \ldots \text{ AND } x_P \text{ is } \tilde{A}_{P,j(r)}$$
$$\text{THEN } g_{r,1} = z_{r,1} \text{ AND } \ldots \text{ AND } g_{r,c} = z_{r,c} \quad (6)$$

R_r is the rth rule of the rule base. By the index p, $j(r)$ it is expressed that in each rule an attribute x_p can assume a different linguistic value $\tilde{A}_{p,j}$ and that the index therefore depends on r. The $g_{r,i}$ ($i = 1, \ldots, c$) are functions that characterize the membership of an object in a class i according to rule r. The $z_{r,i}$ are concrete values of these memberships, e.g., normalized to the interval [0,1]. In other words, with such a rule base, each attribute of a data object is checked and, according to the linguistic values of all attributes of the object, one or several rules classify the object into one or several classes.

Three types of rule-based classification can be distinguished [29]:

1. Assignment of an object to a single class. In the model above, this corresponds to values $z_{r,i} \in \{0,1\}$, with all $z_{r,i}$ ($i \neq m$) except one—$z_{r,m} = 1$—taking a value of 0.
2. Assignment of an object to a single class, but with the assignment weighted with a *certainty factor*. This corresponds to values $z_{r,i} \in [0,1]$, where all $z_{r,i}$ ($i \neq m$) except one—$z_{r,m}$—take a value of 0.

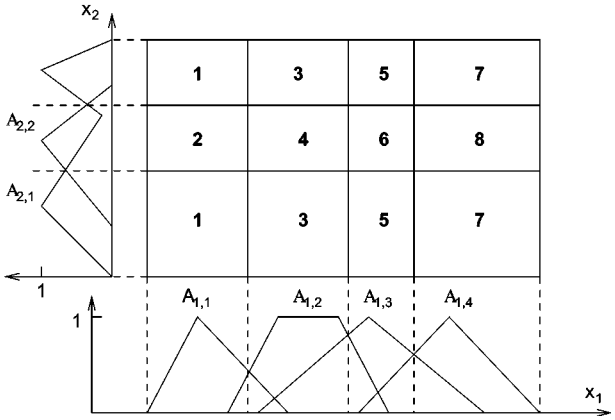


Fig. 6. Cluster-formation with rule-based classification [30].

3. Fuzzy assignment of an object to several classes with a *certainty factor* for each class. This corresponds to arbitrary values $z_{r,i} \in [0,1]$.

Kuncheva has shown that rule-based clustering with crisp output values $z_{r,i} \in \{0,1\}$ corresponds to a division of the feature-space into "hyper-boxes" [30]. This is illustrated in Fig. 6 for a two-dimensional feature-space.

3.5. Dynamic Clustering Algorithms

The dynamics of the system can either refer to the objects or to the clusters, which results in four possible scenarios:

1. Static objects and static cluster structure
2. Static objects and dynamic cluster structure
3. Dynamic objects and static cluster structure
4. Dynamic objects and dynamic cluster structure

The last three scenarios are treated in the framework of dynamic data analysis. We are dealing with the fourth scenario in the following, which matches the clustering problem in an ad hoc network. First, we will describe how the dynamic character of the objects can be taken into account, before the dynamics of the cluster structure are treated.

In Section 3.3, a vector metric like the euclidic distance has been used as objective function of the clustering algorithm. With dynamic objects, the use of a distance metric is also possible; however, a pointwise distance between the trajectories of the objects has to be considered in this case. The similarity $s_{x_1x_2}$ of two trajectories $\boldsymbol{x}_1(t)$ and $\boldsymbol{x}_2(t)$ can be derived from the distance, e.g., by one of the three following relations [31]:

$$s_{x_1x_2} = 1 - d(\boldsymbol{x}_1, \boldsymbol{x}_2) \quad (7)$$

$$s_{x_1x_2} = \frac{1}{1 + d(\boldsymbol{x}_1, \boldsymbol{x}_2)} \tag{8}$$

$$s_{x_1x_2} = \exp^{-d(\boldsymbol{x}_1,\boldsymbol{x}_2)} \tag{9}$$

There is another class of similarity measures of trajectories, is based on certain characteristics of the trajectories, known as *structural similarity.* The degree of structural similarity can be transformed with Eq. (7, 8, or 9) into a distance measure between the trajectories. Afterwards, the data objects could be classified with a static objective-function-based (cf. Section 3.3) clustering algorithm (independently of the fact that pointwise or structural distance measures have been used). By these means, dynamic objects are classified in [32] with the fuzzy-*c*-means algorithm. In [33] the algorithm of Gath and Geva [25] is used.

It should be mentioned that dynamic clustering does not necessarily imply the use of a dynamic similarity measure. The dynamic character of the algorithm first of all refers to the dynamic cluster structure and the clustering process itself.

Regarding the dynamics of the cluster structure, Mann [34] has formulated the following possible topology changes:

- Creation of new clusters: Whenever an object cannot be assigned to a cluster an additional cluster should be created. Dubuisson [35] distinguishes between ambiguity reject and distance reject as possible reasons for the creation of a new cluster.
- *Merging of clusters:* Several clusters should be merged into a single cluster if there are a lot of objects that have a similar degree of membership in all these clusters. The clusters are not clearly separated in this case.
- *Splitting of clusters:* The contrary process to the merging of clusters should be initiated if more than one separate subgroup inside an existing cluster is detected. The cluster is not sufficiently homogeneous in this case.
- *Deletion of clusters:* The deletion of clusters is closely related to the age of the data objects. Old objects might be disregarded in the future classification process, and therefore the number of objects assigned to a cluster might shrink below a certain minimum density, at which the cluster should be deleted.
- *Drift of clusters:* The drift of a cluster refers to a drift of the cluster center. The drift can be caused either by new objects or a change in the membership values of the existing objects.

In conclusion, the dynamics of the cluster-structure are characterized by time-varying cluster centers and membership values of the objects to the clusters.

Regarding the dynamics of the clustering process itself, dynamic clustering is carried out in two phases [33]:

1. A *monitoring process* is used to detect structural changes.
2. During an *adaptation process,* the cluster structure is adapted according to the detected structural changes.

The monitoring of the classification is performed by means of several performance indicators, which measure, for example, the accuracy of the classification, the number of misclassified objects in the past, the unambiguity of the classification, the distribution of the objects among the classes, and the temporal moments of the object attributes.

During the adaptation process the classification can either be rebuilt from scratch or gradually adapted to the detected structural changes. The first solution could, for example, imply a new classification by means of the fuzzy-*c*-means algorithm. A possible solution for a gradual adaptation of the classification could also be based on the fuzzy-*c*-means algorithm, but with a recursive adaptation of the cluster centers [36]:

$$\begin{aligned}\mathbf{v}_i(n+1) &= \frac{\sum_{k=1}^{n} \mu_{ik}^m \boldsymbol{x}_k + \mu_{i(n+1)}^m \boldsymbol{x}_{n+1}}{\sum_{k=1}^{n} \mu_{ik}^m + \mu_{i(n+1)}^m} \\ &= \frac{\mathbf{v}_{Z,i}(n) + \mu_{i(n+1)}^m \boldsymbol{x}_{n+1}}{\mu_{N_i}(n) + \mu_{i(n+1)}^m}\end{aligned} \tag{10}$$

The sums $\mathbf{v}_{Z_i}(n)$ and $\mu_{N_i}(n)$ are built recursively by adding the respective terms of a new object n to the previous sums $\mathbf{v}_{\boldsymbol{Z}_i}(n-1)$ and $\mu_{N_i}(n-1)$. For this purpose, the values $\mathbf{v}_{\boldsymbol{Z}_i}(n)$ and $\mu_{N_i}(n)$ are stored after each time step.

Mikenina [33] uses a similar approach to carry out gradual cluster changes based on the algorithm of Gath and Geva [25]. Furthermore, Mikenina also considers abrupt changes like creation, merging, or splitting of clusters, which are carried out if certain indicators like cluster similarity, cluster overlap, or object density inside a cluster exceed predefined thresholds.

4. FUZZY RULE-BASED CLUSTERING ALGORITHM

After having studied many existing data clustering algorithms, we came to the conclusion that none of them

was suited to the application under consideration. The reasons for this are illustrated in Section 4.1. We therefore developed a new dynamic clustering algorithm, which will be described in the remainder of this chapter.

4.1. Requirements and Characteristics of the Algorithm

As mentioned earlier, the clustering task in an ad hoc network can be characterized as an unsupervised dynamic classification problem. The stations can be considered as data objects, which are characterized by certain attributes like position, speed, traffic, etc. The membership of the stations in the clusters could be defined in a fuzzy way, even though finally stations have to be unambigously assigned to a single cluster (except for the forwarding terminals).

In a clustered ad hoc network the cluster center is represented by an existing station, the central controller. The CC could be interpreted as a cluster prototype. However, prototype-based clustering algorithms in general result in cluster prototypes that are virtual points in the feature space, and not existing data objects.

There are a few additional differences between classical data analysis and the application. A first difference can be seen in the fact that in data analysis the clustering is carried out by a "global observer or classificator," which has total knowledge about all data objects. This will probably not be the case in a realistic ad hoc network, where a single station will not have global knowledge about all stations in the network. The clustering in the ad hoc network will also have to be performed in a distributed way (in contrast to classical data analysis). Another important difference is that no training data are available in our application scenario. Training data are used in data analysis to find appropriate clusters and to verify the accuracy of the clustering result. In most of the dynamic data-clustering algorithms, the underlying dynamics are created by new data objects. In an ad hoc network, the dynamics of the system are mainly caused by movements and traffic fluctuations of the *existing* objects (i.e., stations). Also, of course, in an ad hoc network objects will appear or disappear if users switch on or off their devices. In classic data analysis, data objects that have very extraordinary characteristics are often ignored. This might be useful in order not to worsen the clustering performance for "normal" data objects. In a cluster-based wireless network, however, *all* stations have to be assigned to a cluster.

Taking these considerations into account, the following requirements can be formulated for the clustering algorithm of the ad hoc network:

- The algorithm has to be real-time capable.
- The clusters have to have a certain minimum stability (in the order of 500 ms).
- The clustering has to take the existing constellation into account and cannot rebuild the complete network in a single time step.
- The algorithm has to take "hard" constraints into account.
- *All* objects have to be assigned to a cluster.
- The algorithm has to work without a training-data set.
- The clustering has to build clusters in which the cluster centers are represented by existing (and not virtual) objects.

The necessity of the real-time capability is obvious for a realistic wireless network. A minimum stability is nevertheless needed, because otherwise the signaling for one topology change could not be completed before the next topology change regarding the same object would occur. Taking into account "hard" constraints means that not all conditions can be formulated in a fuzzy way, but that some constraints represent real "hard" upper or lower bounds (like the minimum stability).

Besides the indispensable requirements, some desirable features of the clustering algorithm can be formulated:

- The algorithm should be distributed, that is, carried out in a decentralized way.
- The algorithm should minimize the number of clusters.
- It would be desirable that the algorithm is adaptive and can react to changing conditions.
- The clustering decisions should be understandable by an expert.
- In turn, expert knowledge should be incorporated in the clustering process.

Decentralized execution is on the border between indispensable and desirable features. A fully centralized approach will be technically difficult to realize. However, some partial centralization, e.g., in the sense that the CCs carry out some parts of the algorithm, would certainly be acceptable. Regarding the number of clusters, a minimization would be desirable in order to minimize the forwarding traffic. This is in contrast to clustering in the framework of data analysis, where the number of clusters is not part of the objective function, but only a result of the clustering process.

Based on these assumptions, we will now briefly analyze which type of clustering algorithm might be

suited to our scenario. The decisions invoked by a neural network cannot be fully understood by an expert. Therefore, this type of algorithm has not been considered in our work.

Graph-theoretic algorithms require a scan of the complete graph, which makes a decentralized execution of the algorithm very difficult. Objective-function-based clustering algorithms seem to be well suited for the scenario at first sight, because they build cluster centers and because they have already been used in some dynamic clustering approaches. The problem that the cluster centers are virtual points in the feature space could be solved in such a way that the station that is the nearest neighbor to the virtual cluster center always becomes the central controller. Two other drawbacks to this type of algorithm, however, cannot be resolved. The first problem is its real-time incapability, which is due to its iterative character. The second drawback is the centralization of the objective-function-based algorithms.

The last group of algorithms to consider are the knowledge-based algorithms. Their main advantage is their real-time capability, which has been proven many times in practical applications in the framework of *fuzzy control*. Rule-based algorithms seem to be sufficiently flexible to guarantee a minimum stability of the clusters, i.e., to limit the number of topology changes per time. We will outline in the following sections how this can be achieved. Another advantage of rule-based algorithms is that "hard" constraints can be easily taken into account (by appropriate rules). It is possible to assign *all* objects to a cluster. Furthermore, rule-based algorithms do, in general, not require any training data, if the knowledge is taken from an expert and not from past data.

The knowledge-based algorithms fulfill not only all the indispensable requirements but also most of the desirable features of an ad hoc clustering algorithm. The most important characteristic is the possible decentralized execution of the algorithm, i.e., the rules. Another important advantage of this type of algorithm is that the incorporation of expert knowledge is eased and that, on the other hand, rules can be easily understood by an expert.

4.2. Clustering Algorithm

As the analysis of different types of clustering algorithms has shown, knowledge-based algorithms are the best candidate for the application under consideration. In Section 3.4, it was stated that in classic, rule-based classification algorithms, an object is assigned to one or several classes according to the rules in the rule base. In a *dynamic* classification problem with variable object attributes, the rules would have to be called periodically. The new assignments according to the rules would have to be compared against the current assignments of the objects to the clusters. In the case that a difference is detected, an object would have to change its cluster.

This approach seems to be quite long-winded, and it has the additional disadvantage that the number of clusters is not controlled. We therefore propose a new rule-based clustering scheme, wherein the topology changes are considered instead of the membership degrees of the objects. We consider slightly different topology changes than the ones proposed by Mann [34]:

- *Creation of new clusters*
- *Deletion of clusters*
- *Drift of clusters*
- *Objects changing association to cluster*

In a data analysis application, the merging or splitting of clusters could be part of the new clustering scheme as well. However, in the ad hoc network, the splitting of clusters would be realized by the creation of an additional cluster, followed by some objects associating with the new cluster. In the same way, the merging of two clusters would be realized by some objects leaving their cluster, followed by the deletion of the old cluster of these objects.

In the specific case of a cluster-based ad hoc network with forwarding terminals (FT) in between the clusters, three additional topology changes are considered:

- *Creation of a forwarder*
- *Deletion of a forwarder*
- *Handing over the forwarder function to another station*

The consequences of the clustering rules in the new algorithm are not the degrees of membership in the clusters but the different possible topology changes. The output variables of the rules have the form of *yes/no* decisions regarding the possible topology changes. We therefore consider these output variables as *linguistic variables.* In order to make use of the improved inference capability of fuzzy logic, we also formulate the input variables of the rules in the form of linguistic variables. This will also ease the formulation and understanding of the rules by an expert. The new clustering scheme can be interpreted as a *fuzzy control* approach because input values for the clustering rules are taken from a dynamic process. The output values of the rules trigger topology changes, which represent control ac-

tions in the dynamic system. Because input and output variables are formulated as linguistic variables, the rules are of the Mamdani type (cf. [37]). The selection of the *input* variables as well as the formulation of the rules very much depends on the specific application. We will develop a possible rule base for the ad hoc network in the following sections. The four basic clustering events (*creation of new clusters, deletion of clusters, drift of clusters,* and *objects changing association to cluster*) will probably appear as *output* variables of the rules in most types of applications. Note that our clustering algorithm can not only be used in the framework of ad hoc networking but can be applied to the scope of dynamic data analysis in general. It is especially suited for all clustering problems in which a classification in real time is needed. However, in the following sections we will concentrate on the concrete realization of the algorithm for the purpose of mobility and load management in a wireless ad hoc network.

4.3. Input and Output Variables

The input variables of the rules are very specific to the wireless application. A station could, for example, use the following input variables:

- *Received signal strength* (RSS) of the own CC
- or trajectory of this RSS in the past
- RSSs of neighboring CCs
- Signal quality, i.e., *packet error ratio* (PER) or *bit error ratio* (BER), with which the own CC is received
- Signal quality, i.e., PER or BER, with which the neighboring CCs are received
- Traffic load of the own CC
- Traffic load of the neighboring CCs
- Number of terminals in the own cluster
- Average RSS value of a station (see explanation below)
- Difference of the average RSS values of two stations
- Connectivity, i.e., number of direct neighbors of a station
- Difference of the connectivity of two stations
- Average total traffic of a station
- Difference of the average total traffic values of two stations
- Speed of movement of a station
- Time since the last CC handover, *wireless terminal* (WT) handover, and FT handover
- Speed of change of the RSS of the own CC
- Power supply of a station (battery powered or plugged)

A *station* can assume the three different roles—*CC, FT,* or *WT*. We designate as *WT* all terminals that are neither a CC nor an FT.

The RSS and the PER are the most important indicators in order to decide on the cluster membership of a terminal. A terminal should be assigned as far as is possible to the CC that is received with the highest RSS, i.e., lowest PER. However, for load balancing reasons, a terminal could also be assigned to a different cluster with a very low traffic load. The traffic load is the most important indicator for the creation of additional clusters. On the other hand, the number of terminals inside a cluster is probably the most important parameter to control the deletion of a cluster.

The average over the RSS of all neighboring terminals (which is equivalent to an average distance to the neighbors) and the connectivity of a station as well as its total traffic could be used in order to decide on the drift of a cluster, i.e., a CC handover. When choosing a new CC, other criteria like the speed of a station or its type of power supply could play a significant role in real applications. Even though the speed of a terminal is difficult to determine in practice, a terminal could at least dispose of the information if it is mobile or not, is plugged or not, etc.

In the previous section, it was mentioned that a requirement for the clustering algorithm is to guarantee a minimum stability of the clusters. This can be achieved in a rule-based clustering scheme by incorporating input variables like “time since last CC handover,” “time since last WT handover,” or “time since last FT handover.” A condition can then be formulated that the time since the last clustering event is larger than a certain minimum bound. We will use such a bound for the CC handovers in Section 4.4.

In general, trajectories of all the mentioned values or their derivatives could be used. However, we have decided to consider only a single value of the input variables. Nevertheless, this single value can be a sliding average in order to avoid instabilities in the network. We formulate all input (and output) variables as linguistic variables. For this purpose, we normalize the basic input variables to the interval [0,1] and define the five linguistic terms:

B: Big
MB: Medium Big
M: Medium
MS: Medium Small
S: Small

For these linguistic terms triangular membership functions are used, as illustrated in Fig. 7. Note that the positions of the characteristic points of the triangular functions can be

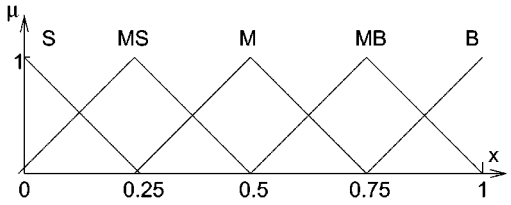


Fig. 7. Membership functions of the input variables.

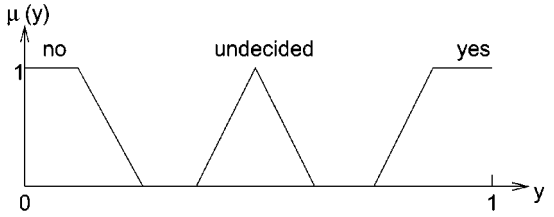


Fig. 8. Membership functions of the output variables (cf. [38]).

chosen independently for each base variable (RSS, traffic load, PER, number of WTs, connectivity, speed, etc.). For bounded variables a normalization can be carried out simply by dividing all input values by the maximum possible value. For infinite base variable domains (like "time since last CC handover" or "number of WTs") a normalization in the following form is carried out:

$$x_{norm} = 1 - \frac{1}{1 + \alpha x} \tag{11}$$

with a scalar α that can be chosen in an appropriate way for each specific variable.

The *output* variables of the clustering algorithm correspond to the allowed topology changes. The following summarizes the topology changes as well as the respective output variables:

- Cluster creation: "CC creation" (yes/no/undecided)
- Cluster deletion: "CC deletion" (yes/no/undecided)
- Cluster drift: "CC handover" (yes/no/undecided)
- Object changing association to cluster: "WT handover" (yes/no/undecided)
- Installation of a new forwarder: "FT creation" (yes/no/undecided)
- Deletion of a forwarder: "FT deletion" (yes/no/undecided)
- Handover of forwarder functionality: "FT handover" (yes/no/undecided)

Each of the output variables is a linguistic variable that can take the linguistic values "yes," "no," or "undecided." Such variables have already been used in [38] to decide on a terminal handover in infrastructure-based networks. We will use the same membership functions as in [38] for each of the output variables (cf. Fig. 8).

A few additional informative output variables could be defined. Similar to [38] we will use an output variable "INDISPENSABLE" (yes/no/undecided) to indicate if the execution of the clustering event was unavoidable or if it occurred for performance optimization purposes. Other output variables indicate the reason for the execution of the clustering event (like "CC-Creation-Reason-Traffic," etc.).

4.4. Fuzzy Rule Base

In this section, we will give an example of a *knowledge-based* formulation of the fuzzy rules. We have formulated the rules in such a way that they can be executed in a decentralized manner by each CC or WT. The rules described here refer to the four basic clustering events: "CC creation," "CC deletion," "CC handover," and "WT handover." As far as the three FT-related clustering events are concerned, we currently are not triggering these events by fuzzy rules. In our current implementation, an FT deletion is only carried out if the current FT can no longer keep contact with one of the two connected CCs. FT creation and FT handover are not triggered by the basic rule-based approach, but by an additional algorithm that will be described in Section 4.5.

A CC can decide on all four possible basic clustering events. We will give an example of a possible CC rule base in the following.

The following CC rules regard the CC creation:

1. IF Traffic-CC = "Big" AND Traffic-Neighbor-CCs = "Big"
   THEN CC-Creation = "yes" AND INDISPENSABLE = "no" AND CC-Creation-Reason-Traffic = "yes"
   This rule foresees that a new cluster is formed if the own cluster as well as the clusters in the neighborhood of the CC run out of capacity.
2. IF Traffic-CC = NOT "Big"
   THEN CC-Creation = "no"
   This rule is the first counterpart to the previous rule.
3. IF Traffic-Neighbor-CCs = NOT "Big"
   THEN CC-Creation = "no"
   This is the second counterpart to the first rule. It must be noted that another rule is formulated later that foresees a forced handoff of some ter-

minals of the cluster of this CC to the neighboring clusters with small load.

The following CC rules regard the CC deletion:

1. IF Traffic-CC = "Small" AND Number-Terminals = "Small" AND "terminals-served"
   THEN CC-Deletion = "yes" AND CC-Deletion-Reason-Number-Terminals = "yes"
   If only a very small number of terminals is associated with the CC and the traffic load in the cluster is low, the cluster is deleted. The condition "terminals-served" is an example of how a crisp condition can be incorporated in the fuzzy rules. The CC deletion is only carried out if all terminals (WTs and FTs) can be taken over by neighboring CCs.
2. IF Traffic-CC = NOT "Small"
   THEN CC-Deletion = "no"
   This is the first counterpart of the previous rule.
3. IF Number-terminals = NOT "Small"
   THEN CC-Deletion = "no"
   This is the second counterpart of the first CC deletion rule.
4. IF NOT "terminals-served"
   THEN CC-Deletion = "no"

   This condition is included in order to guarantee that only in the case that *all* WTs and FTs can be handed over to other clusters is a CC deletion carried out.
   We will now describe CC rules that concern a CC handover:

1. IIF Average-RSS-Difference = "Big" AND Time-since-last-CC-HO = "Big" AND Speed-CC-candidate = "Small" AND RSS-CC-Candidate = ("MEDIUMBIG" OR "Big")
   THEN CC-HO = "yes" AND INDISPENSABLE = "no" AND CC-HO-Reason-RSS = "yes"
   This rule triggers a CC handdover if a CC candidate is found that has a better average RSS value (i.e., a lower average distance to its neighbors) than the current CC. The time condition is inserted in order to guarantee a certain minimum stability of the clusters. The last condition, which requires a small distance between the old and the new CC, has been added because simulations have shown that otherwise unpractical and disadvantageous CC handovers would result
2. IF Average-RSS-Difference = "Big" AND Time-since-last-CC-HO = "Big" AND Speed-CC-candidate = "Small" AND RSS-CC-Candidate-To-Neighbor-CCs = "SMALL"
   Then CC-HO = "yes" AND INDISPENSABLE = "no" AND CC-HO-Reason-RSS = "yes"
   The only difference to the first rule is the fourth condition. The idea is that if the CC candidate is not close to the current CC, it should be at least relatively far away from all other CCs.
3. IF Average-RSS-Difference = NOT "Big"
   THEN CC-HO = "no"
   This rule is the first counterpart to the two previous rules.
4. IF Time-since-last-CC-HO = NOT "Big"
   THEN CC-HO = "no"
   This rule is the second counterpart to the first and second CC handover rules. A certain minimum cluster stability can be guaranteed if μ(NOT "Big") = 1 − μ("Big") assumes a value of 1 below a certain minimum time.
5. IF Speed-Candidate-CC = NOT "Small"
   THEN CC-HO = "NO"
   This rule is the third counterpart to the two first CC handover rules. The reason for this rule is that CCs should be as stationary as possible in order to stabilize the network.
6. IF RSS-CC-Candidate = NOT ("MEDIUMBIG" OR "Big")
   THEN CC-HO = "NO"
   This rule is the last counterpart to the first CC handover rule.
7. IF RSS-CC-Candidate-To-Neighbor-CCs = NOT "SMALL"
   THEN CC-HO = "NO"
   This rule is the last counterpart to the second CC handover rule.

Additional rules similar to the two first CC handover rules can be formulated by replacing the "Average-RSS-Difference" by either the "Total-Traffic-Difference" or the "Connectivity-Difference." For the sake of brevity we will not describe these rules here.

Finally, WT handovers could be initiated by a CC for the purpose of load balancing among the clusters:

1. IF Traffic-CC = "Big" AND Traffic-Neighbor-CCs = "Small" AND "WT-served"
   THEN WT-HO = "yes" AND INDISPENSABLE = "no" AND WT-HO-Reason-Traffic = "yes"
   A WT handover is initiated if the load of the own cluster is high and the load in at least one neighboring cluster is low. The last condition, which had already been used above in a similar way,

guarantees that the WT handover is only carried out if the WT can be taken over by the other CC.

2. IF Traffic-CC = NOT "Big"
   THEN WT-HO = "no"
   This is the first counterpart to the previous rule.
3. IF Traffic-Neighbor-CCs = NOT "Small"
   THEN WT-HO = "no"
   This is the second counterpart to the first WT handover rule.
4. IF NOT "WT-served"
   THEN WT-HO = "no"
   If the WT cannot be taken over by the other CC (e.g., because it is out of range), no WT handover is carried out.

After the description of the CC rules, an examplary WT rule base, stored and executed by all WTs, is outlined in the following. A WT can only initiate its own CC creation as well as its own WT handover. The CC creation rules are again treated first:

1. IF RSS-CC = "Small" AND RSS-Neighbor-CCs = "Small"
   THEN CC-Creation = "yes" AND INDISPENSABLE = "yes" AND CC-Creation-Reason-RSS = "yes"
   This rule guarantees that each terminal is associated with a cluster. If no other CC is in range, the terminal makes itself a CC.
2. IF RSS-CC = NOT "Small"
   THEN CC-Creation = "no"
   This is the first counterpart to the previous rule.
3. IF RSS-Neighbor-CCs = NOT "Small"
   THEN CC-Creation = "no"
   This is the second counterpart to the previous rule.

The following WT rules regard a possible WT handover of the terminal that executes the rules:

1. IF RSS-CC = "Small" AND RSS-Neighbor-CCs = ("Medium" OR "Medium Big" OR "Big")
   THEN WT-HO = "yes" AND INDISPENSABLE = "yes" AND WT-HO-Reason-RSS = "yes"
   If the own CC is received only weakly and another CC is received with at least medium RSS, a WT handover to the other CC should be initiated. The handover is considered indispensable.
2. IF RSS-CC = NOT "Small"
   THEN WT-HO = "no"
   This rule is the first counterpart to the previous rule.
3. IF RSS-Neighbor-CCs = ("Small" OR "Medium Small")
   THEN WT-HO = "no"
   This is the second counterpart to the first WT handover rule.
4. IF PER-CC = "Big" AND PER-Neighbor-CCs = "Small"
   THEN WT-HO = "yes" AND INDISPENSABLE = "yes" AND WT-HO-Reason-PER = "yes"
   This is a rule similar to the first WT handover rule with the RSS value being replaced by the PER.
5. IF PER-CC = NOT "Big"
   THEN WT-HO = "no"
   This is the first counterpart to the previous rule.
6. IF PER-Neighbor-CCs = NOT "Small"
   THEN WT-HO = "no"
   This is the second counterpart to the PER-based WT handover rule above.
7. IF RSS-CC = ("Small" OR "Medium Small" OR "Medium") AND RSS-Difference = "Big"
   THEN WT-HO = "yes" AND INDISPENSABLE = "no" AND T-HO-Reason-RSS-Difference = "yes"
   This rule triggers a WT handover based on the RSS difference between the best-received neighbor CC and the current CC of the WT. If the other CC is received with a much better RSS, a handover should be carried out.
8. IF RSS-CC = ("Small" OR "Medium Small" OR "Medium") AND RSS-Difference = NOT "Big")
   THEN WT-HO = "no"
   This is the counterpart to the previous rule. The cases where the RSS of the own CC is "Medium Big" or "Big" have already been treated in Rule 2.

As has already been mentioned, the input variables RSS, PER, and RSS-Difference are sliding-average values in order to avoid "pingpong" decisions. In [39], a WT handover algorithm has been proposed that is based on a fuzzy average of the RSS-Difference. The fuzzy average of the RSS-Difference (ΔRSS) is determined in the following way [39]:

$$\mu(\Delta RSS_n) = \max\left(0, \mu(\Delta RSS_{n-1}) + \mu_{\tilde{N}}(\Delta RSS_n) - \mu_{\tilde{A}}(\Delta\, RSS_n)\right) \quad (12)$$

$\mu(\Delta RSS_n)$ is the WT handover decision criterion. If this criterion assumes a value above a certain threshold (e.g., 3,0), a WT handover to the respective neighboring cell is executed. $\mu_{\tilde{A}}(\Delta RSS_n)$ and $\mu_{\tilde{N}}(\Delta RSS_n)$ represent the membership values to the fuzzy sets "acceptable" and "unacceptable," as illustrated in Fig. 9. By means of Eq. (12), how often the RSS-Difference consecutively falls

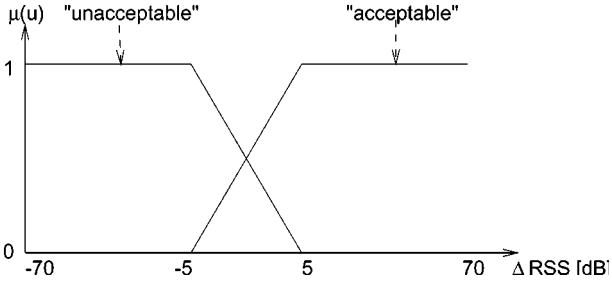


Fig. 9. Fuzzy sets "acceptable" and "unacceptable" [39].

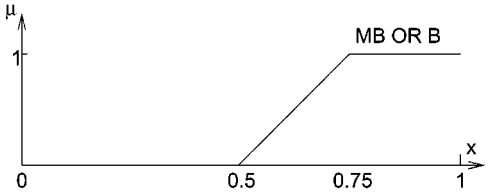


Fig. 10. Membership function of ("MEDIUMBIG" OR "BIG").

below an unacceptable value is measured. The WT handover decision criterion of [39] could be used as an alternative to the last two WT handover rules in the WT rule base. Finally, it should be mentioned that each of the rules above could be weighted with a *certainty factor*. We have currently assigned an equal weight to all the rules.

4.5. Choice of Operators

For the evaluation of the fuzzy rules, the following operators have to be defined:

- The AND operator to combine the antecedents of a rule
- The OR operator inside the antecedents of a rule
- The INFERENCE operator to scale the output fuzzy sets of a rule with the combined membership degree of the antecedents
- The AGGREGATION operator to combine the output of *different* rules
- The DEFUZZIFICATION operator to derive a crisp value out of the resulting membership function of an output variable

For the AND operator, any possible *T-norm* could be used. We are using the minimum operator because it is the least restrictive of all possible T-norms. As the OR operator, the arithmetic sum is used. The reason for this is that very simple and logical membership functions of the input variables will result. Taking the example of the fuzzy set ("MEDIUMBIG" OR "BIG"), the membership function in Fig. 10 will result. Several INFERENCE operators are known from the literature. The two most commonly used in fuzzy control are the so-called *Mamdami inference* and the *scaled inference*, which are defined as follows:

- Mamdami inference: $\mu_M(y) = \min(\mu(\mathbf{x}^*), \mu(y))$
- Scaled inference: $\mu_s(y) = \mu(\mathbf{x}^*) \cdot \mu(y)$

$\mu(\mathbf{x}^*)$ is the combined degree of membership of the antecedents for a given input vector $\mathbf{x}^*$. We are using the scaled inference, which is the faster operation and which is therefore according to [40] most often used in practical fuzzy control systems. The selection of the aggregation and defuzzification operators of the rules are very much related to each other. Many combinations of aggregation and defuzzification operators have been proposed, including *Center-of-area*, *Center-of-sums*, *Height*, *First-of-Maxima*, *Middle-of-Maxima*, etc. (cf. [40]). The two most prominent ones are probably *Center-of-area* and *Center-of-sums*, which are illustrated in Figs. 11 and 12. The difference between the two is that *Center-of-area* uses the max operator as the AGGREGATION operator, whereas the *Center-of-sums* uses the arithmetic sum for this purpose. Therefore, with *Center-of-sums* the dark shaded triangle in Fig. VI is counted twice. Both mechanisms use the *Center-of-gravity* as the defuzzification operator. We are currently using the *Center-of-area* operator for the aggregation of the rules and the defuzzification of the output variables.

As was shown in Fig. VI, the output variables are defined on the interval [0,1]. We therefore choose a value of 0.5 as the decision value for the clustering events. If the defuzzified output variable has a value larger than 0.5, the respective clustering event is carried out; otherwise, not.

4.6. Forwarder Selection Algorithm

Our forwarder selection algorithm was first presented in [3]. We use this algorithm in its distributed version in order to trigger FT creation and FT handover clustering events. An example of a forwarding problem is shown in

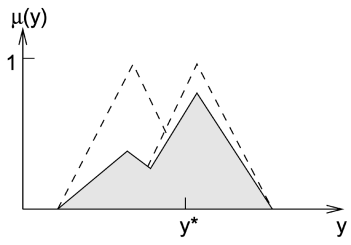


Fig. 11. Center-of-area.

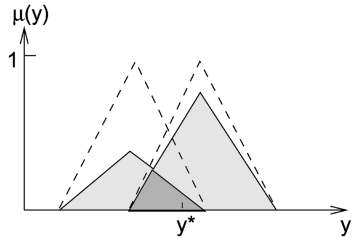


Fig. 12. Center-of-sums.

Fig. 13. It illustrates the optimal forwarding choices for a central controller called CC1. In the situation shown in Fig. 13, terminals T2 and T3 are both possible candidates to interconnect cluster 1 and cluster 3. For the efficiency of the forwarding process, it is preferable that one forwarder interconnects only two clusters. In the constellation considered forwarding between clusters 2 and 3 can only be performed by terminal T2. This means that if terminal T2 was chosen to connect clusters 1 and 3, no terminal would be left for the interconnection of clusters 2 and 3. This example illustrates the importance of the order in which the FTs between the different clusters are installed. In order to avoid, insofar as possible, situations in which no forwarder for the interconnection of two clusters is found, FTs have first to be installed between those clusters where the least number of forwarding candidates exist. Our forwarder selection algorithm allows every CC to determine its optimal forwarding constellation.

In its distributed version, the forwarder selection algorithm is carried out locally by every CC, which also stores locally all necessary information. To prevent confusion, the CC that is performing the algorithm will be called the *processing central controller* (PCC) in the following description.

All possible forwarders are inscribed on a three-dimensional array $F(1 \ldots n, 1 \ldots n, 1 \ldots t)$, with n being the number of CCs known by the PCC and t being the number of possible forwarders. If m different terminals are able to establish a connection between two CCs (i and j, $i < j$), their IDs are entered on the array at $F(i, j, 1 \ldots m)$. The IDs of the possible forwarders are sorted by link quality, so that the ID of the forwarder that offers the best link quality is registered at $F(i, j, 1)$. The array is illustrated in Fig. 14.

The following algorithm is repeated until the matrix is empty:

1. At every step the PCC scans the array for the link ($i, j, i < j$) with the smallest m.
2. The terminal (with ID t_k) registered at $F(i, j, 1)$ is stored in a list, which contains the identified optimum forwarders and the CCs (i and j in this case) that the forwarder should interconnect.
3. All the entries for this link $F(i, j, 1 \ldots m)$ are removed.
4. The ID t_k is searched and removed over the whole array.

In a next step, the PCC scans the list of terminals that have been identified as optimal forwarders and checks each of them to determine whether it is already installed as an FT between the two optimal clusters. If this is the case, the terminal is removed from the optimum forwarder list and no clustering action is required. Also, in the case that the terminal is already installed as an FT but not between the same two CCs, no action is carried out in order not to destabilize the constellation. If the terminal is not yet installed as an FT and also no other FT between the two clusters exists, an FT creation is carried out and the optimal forwarder is installed.

In the case that a different terminal is already installed as an FT between the two respective clusters, an FT handover is initiated in two different situations. Either the optimal forwarder offers a link quality that is higher than the

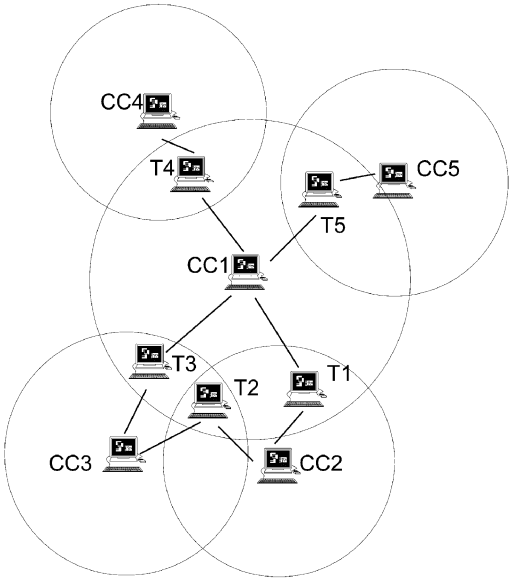


Fig. 13. Example of a forwarding problem.

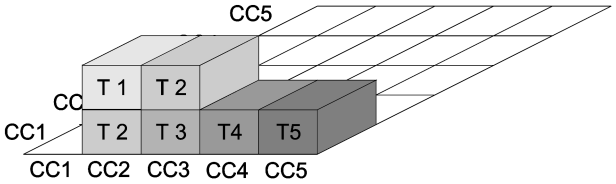


Fig. 14. Three-dimensional forwarder array.

link quality of the current FT by a given (fuzzy) factor, or the current FT is needed as forwarder between two currently unconnected clusters. In the latter situation, by handing over the FT function to another terminal, the old FT is free to take over the FT function for the two other clusters.

This algorithm chooses the best forwarders between the terminals situated within the coverage area of the PCC. A PCC takes its decisions based not only on possible forwarders for its own cluster but also on the interconnections of other clusters. However, a PCC only installs a forwarder—i.e., decides on FT creation or FT handover—if the new FT would serve as a forwarder for its own cluster. This is necessary in order to avoid instabilities in the case that different PCCs do not have the same information about possible forwarding constellations. A situation in which the CCs do not have exactly the same information is not very unrealistic. We have foreseen that each terminal informs all CCs in its hear range, with which CCs it would be able to interconnect. In this case, a CC disposes of only local information. If this information (i.e., the forwarder array) is exchanged among neighboring CCs, a situation with almost global information could be achieved. Note that the algorithm could easily be modified if several FTs between the same two clusters should be installed.

5. PERFORMANCE EVALUATION OF THE CLUSTERING ALGORITHM

5.1. Mobility Model

The mobility model we assume is the *random waypoint* model [41]. It is frequently used for the performance evaluation of ad hoc networks and especially of routing algorithms [42,43]. In this model, each node selects a random destination within the given area, to which it moves on a straight line. The speed of movement is uniformly distributed between 0 and a maximum speed. At each destination, a node pauses for a given constant pause duration before it moves to the next randomly drawn destination.

5.2. Routing Algorithm

We have developed a routing algorithm called *hierarchical time-vector-routing* (HTVR), which is based on routing tables that are stored at the CCs and are periodically updated by message exchanges between the CCs. Each routing table contains one field per terminal in the network as well as the time, T_{up}, at which the table was last updated. Each of the terminal fields has five entries: the identifier of the forwarder to which the data of the respective terminal has to be directed; the path length, *PL;* the maximum transmission rate of the path; a so-called

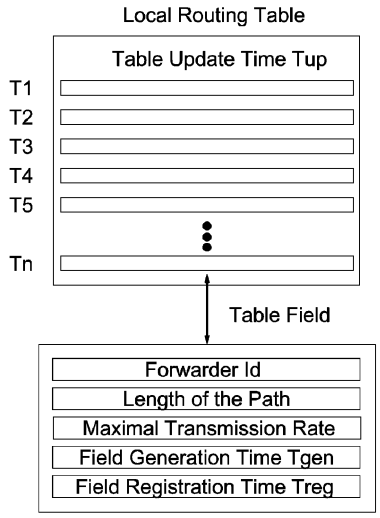


Fig. 15. HTVR routing table.

field generation time, T_{gen}; and a field registration time, T_{reg} (cf. Fig. 5). The entry T_{gen} indicates when the terminal has changed the cluster for the last time, and the entry T_{reg} stores when the field was updated the last time by the CC that stores the respective routing table.

The updating procedure foresees that an updating CC (called UCC in the following) periodically sends its T_{up} to all its neighboring CCs and asks for all their terminal fields that have a generation time $T_{gen} > T_{up}$. The fields received are then updated in the routing table of the UCC, if the following update conditions are fulfilled [44]:

- $t_{gen}^{new} > t_{gen}^{ucc}$
- OR $((t_{gen}^{new} = t_{gen}^{ucc})$ AND $(PL^{new} + 2 < PL^{ucc}))$
- OR $((t_{gen}^{new} = t_{gen}^{ucc})$ AND $(PL^{new} + 2 = PL^{ucc})$ AND $(min(MTR^{new}, MTR^{nch-ucc\,1}) = MTR^{ucc}))$

If the conditions are fulfilled, the following changes are applied:

- The new forwarder ID is set to the ID of the forwarder that connects the CC with the CC from which the response was received.
- $t_{gen}^{ucc} = t_{gen}^{new}$
- $PL^{ucc} = PL^{new} + 2$
- $MTR^{ucc} = min(MTR^{new}, MTR^{ncc-ucc})$

$MTR^{ucc-ncc}$ is the maximum transmission rate between the neighboring CC and the updating CC.

5.3. Simulation Scenario

Our simulation scenario is an exhibition hall of the size 72 m $\times$ 72 m. Inside this area we place 30 devices

equally distributed that are moving according to the *random waypoint* mobility model (see Section 5.1). A radio propagation law according to Eq. (13) is assumed.

$$P_R = P_S \cdot \left(\frac{c_0}{4\pi f}\right)^2 \cdot \frac{1}{d^\gamma} \tag{13}$$

P_R is the received power and d the distance between sender and receiver. The transmitter power P_S of the devices is constant and set to 0.1 W. c_0 is the speed of light and f the carrier frequency, which is set to 5.5 GHz. We have assumed a typical propagation exponent of $\gamma = 3.5$. The sensitivity of the devices is set to -85 dBm, which is, for example, the sensitivity required by the HIPERLAN/2 standard. With these settings the hear range of a station is on the order of 47 m.

We have carried out different types of simulations series. In the first scenario, a constant end-to-end load of 30 Mbit/s has been assumed, and the mobility of the stations has been varied by increasing the average speed of the stations. It is assumed that connections between all terminals are equally probable. In a second scenario, a constant speed of 2 m/s and a constant pause time of 5 s have been assumed, and the end-to-end load has been increased.

5.4. Simulation Results

The results of the simulations with varying speed of the stations are reported in Figs. 16–20.

In Fig. 16, the control of CC handovers by fuzzy rules is compared against an ID-based and a purely connectivity-based CC-handover algorithm. It can be depicted that the fuzzy algorithm is as stable as the *lowest ID* (LID), whereas the *highest connectivity* (HIC) results in frequent CC handovers, especially at higher speed. It should be mentioned that we have implemented the LID and HIC in such a way that only CC handovers to terminals with a lower ID, i.e., higher connectivity, that are *members* of the cluster of the current CC, are carried out. If CC handovers to any terminal in hear range were allowed, the two algorithms would be more unstable.

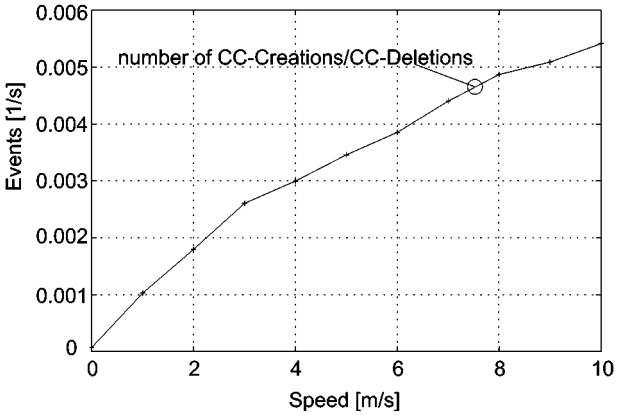


Fig. 17. No. CC creations and deletions vs. speed.

Note that also the ID- and connectivity-based clustering algorithms employed here are fuzzy because the criterion only refers to the CC handover decisions. Cluster creations/deletions and WT handovers are still carried out according to the fuzzy rules because the ID- or connectivity-based algorithms do not deal with these topology changes. Furthermore, the same FT-selection algorithm is applied in all cases. Therefore, the number of CC creations, CC deletions, WT handovers, FT creations, FT deletions, and FT handovers reported in the following is similar for all three CC handover algorithms.

The average number of CC creations and CC deletions per time for a given speed is shown in Fig. 17. It is obvious that in a stationary state the number of CC cre-

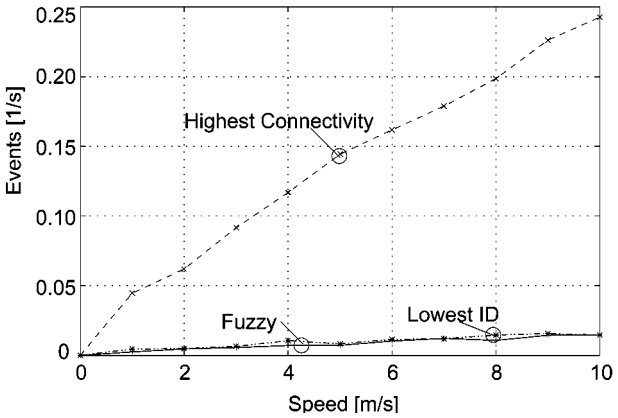


Fig. 16. No. CC handovers vs. speed.

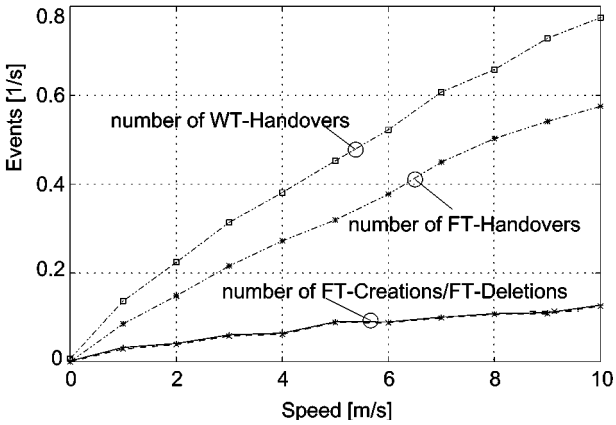


Fig. 18. No. FT creations, FT deletions, and handovers vs. speed.

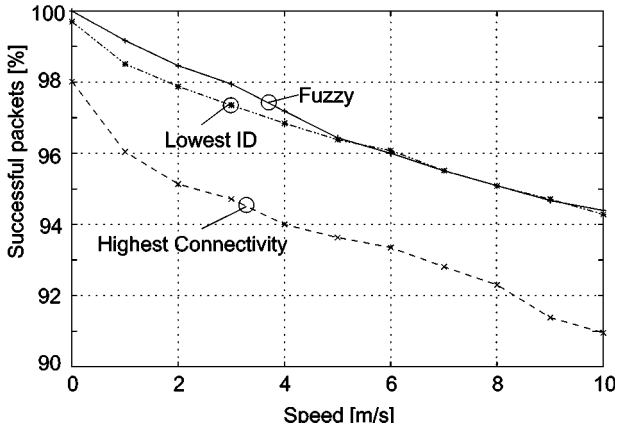


Fig. 19. Percentage of succefully delivered packets vs. speed.

ations and CC deletions have to be equal. The number of cluster creations and deletions depends on the mobility of the terminals, because a situation in which a terminal moves in or out of another cluster occurs more often at higher speed. These topology changes are difficult to avoid if an almost seamless service is aimed at. Alternatively, a terminal that moves out of all existing clusters could just wait until it gets in the range of another CC again.

Fig. 18 contains plots of the number of all FT-related events as well as the number of WT handovers versus the speed of the stations. The number of WT handovers of course very much depends on the mobility of the stations, as illustrated in Fig. VI. Even for stationary terminals, WT handovers occurred, which were obviously initiated for load balancing reasons. The number of FT handovers is influenced by the mobility of the stations because we have chosen as FT selection criterion the RSS value, by which a potential forwarder receives the two CCs to interconnect. There are fewer FT handovers than WT handovers because of the additional WT handovers for the purpose of load balancing among the clusters. The number of FT creations and FT deletions (which have to be equal) also depend on the average speed because it is more probable at higher speeds that an FT cannot keep the connection to the two CCs after a certain time. Finally, in Figs. 19 and 20 the performance of the clustering algorithm is evaluated in terms of the "percentage of successfully delivered packets" and the "average packet delay." The fuzzy CC handover is compared against the identifier- and connectivity-based CC handover algorithms.

Figure 19 shows that success rates in an order of magnitude, typical for ad hoc networks, are obtained. Furthermore, it can be seen that the number of CC handovers has a significant impact on system performance. The differences in the number of CC handovers result in noticeable performance differences. The fuzzy and ID-based CC handover algorithms show a similar behavior. However, the fuzzy CC handover is much more flexible and can be parametrized to be as stable as required. After all, there is a trade-off between stability and fairness regarding the power consumption of the stations.

As far as the average packet delay is concerned, Fig. 20 illustrates that optimizing the topology by clustering pays off in terms of packet delay. The connectivity-based CC handover algorithm results in the lowest delay due to the frequent topology optimizations at the expense of routing path stability. As we have seen in Fig. 19, this instability of the topology results in a high packet loss. In contrast to this very unbalanced behavior of the HIC, the fuzzy and LID CC handover algorithms give a good packet delay and a high percentage of successfully delivered packets.

Figures 21–24 illustrate the dependence of the clustering and overall network performance on the load. With the fuzzy CC handover and the HIC algorithm, the num-

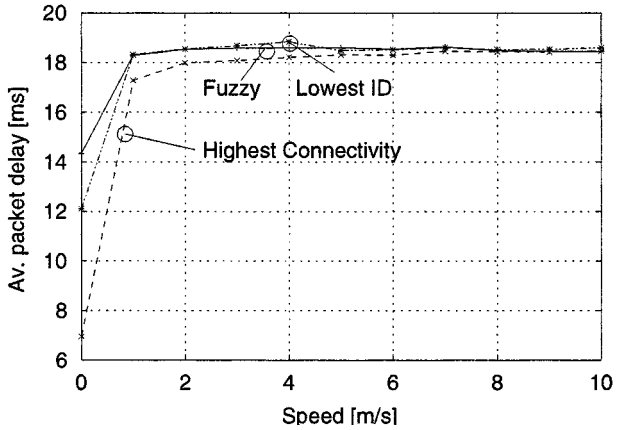


Fig. 20. Average packet delay vs. speed.

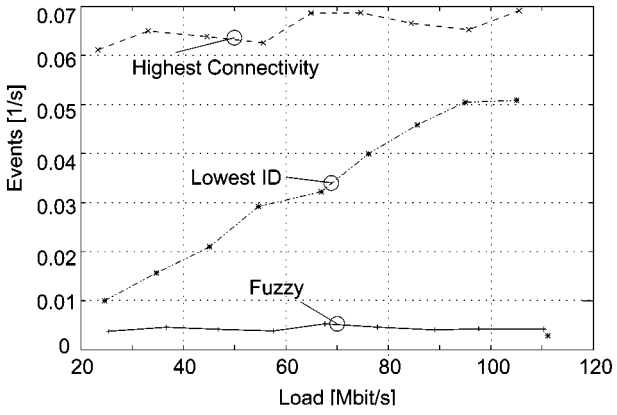


Fig. 21. No. CC handovers vs. load.

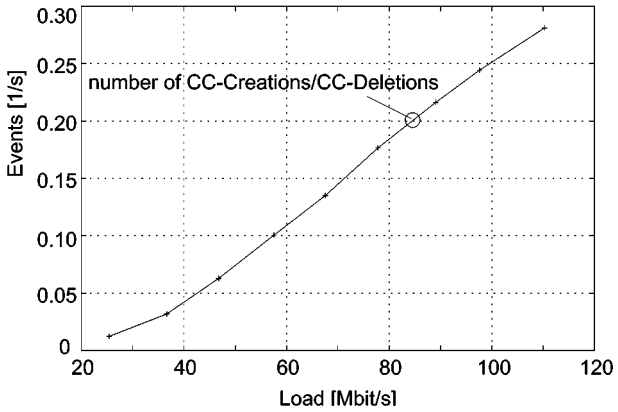


Fig. 22. No. CC creations and deletions vs. load.

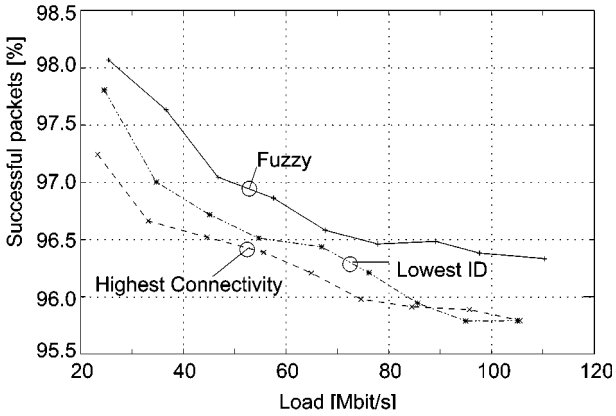


Fig. 24. Percentages of succefully delivered packets vs. load.

ber of CC handovers is not very much affected by a higher network load, as would be expected. However, it is very interesting to note that with the LID, the number of CC handovers increases with the network load. A possible explanation could be the higher number of clusters at higher load, with the result that stations with higher IDs take over the CC function. The higher the network load, the more probable are cluster creations and deletions, as can be seen in Fig. 22. The number of WT handovers also depends on the load, mainly because of the load-related WT handover rules (see Fig. 23). The number of FT handovers is not affected by the load situation because the FT selection is based on an RSS criterion. However, the number of FT creations/deletions increases with the load, which is caused by cluster creations and deletions. The dependence of the percentage of successfully delivered packets on the network load is shown in Fig. 24. The higher network instability and bigger number of clusters/hops at higher load result in a decrease in the percentage of successfully delivered packets. The fuzzy CC handover algorithm copes better with a higher load than the LID and HIC. For the LID, the increasing number of CC handovers (cf. Fig. 21) has a negative effect on the percentage of successfully delivered packets.

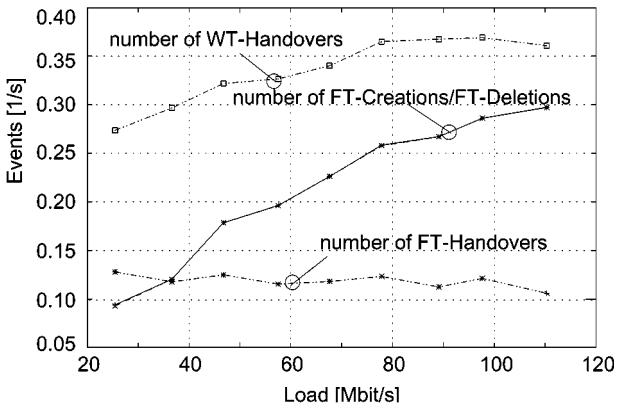


Fig. 23. No. FT creations, FT deletions, and handovers vs. load.

6. CONCLUSIONS

This work studied clustering algorithms used in the framework of data analysis to derive a suitable algorithm for mobility and load management in a cluster-based wireless ad hoc network. The analysis of the data-clustering algorithms has shown that none of the algorithms is perfectly suited to the application considered. However, some useful ideas could be incorporated into a new clustering algorithm that allows for a distributed execution in real time. The new algorithm is based on a fuzzy-inference engine and rule-based knowledge representation. The main characteristic of the algorithm is that several predefined topology changes are triggered by the output variables of the rules. Each output variable indicates in the form of a linguistic variable whether the respective topology change should be carried out or not.

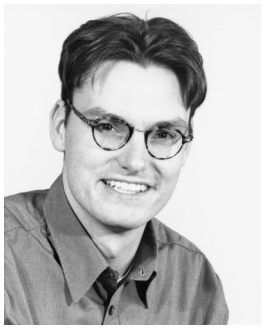
The applicability and stability of the algorithm has been evaluated by computer simulations. The simulation results indicate that the fuzzy clustering algorithm outperformes ID- and connectivity-based CC handover algorithms. Furthermore, in addition to the selection of an appropriate CC, the new algorithm also controls the *number* of clusters (depending on the load).

For each of the allowed topology changes, a signaling procedure has been defined (which we did not report in this paper). Two of them (CC and WT handover) have already been proposed and incorporated into the HIPERLAN/2 standard. Our plan is to propose the remaining (CC creation, CC deletion and FT related) signaling pro-

cedures to ETSI and IEEE 802.11, which underlines the practical relevance of this work.

REFERENCES

1. M. Gerla and J. Tzu-Chieh Tsai, Multicluster mobile, multimedia radio network, *Wireless Networks,* Vol. 1, pp. 255–265, Oct. 1995.
2. C. R. Lin and M. Gerla, Adaptive clustering for mobile wireless networks, *IEEE J. Select. Areas Commun.,* Vol. 15, pp. 1265–1275, Sept. 1997.
3. J. Habetha, M. Nadler, and D. Calvo de No, Dynamic clustering with quality of service guarantees and forwarder selection in wireless ad hoc network. In *Proc. Asia Pacific Conference on Communications,* pp. 31–35, Nov. 2000.
4. J. Habetha, A. Hettich, J. Peetz, and Y. Du, Central controller handover procedure for ETSI-BRAN HiperLAN/2 ad hoc networks and clustering with quality of service guarantees. In *IEEE Annual Workshop on Mobile Ad Hoc Networking & Computing (MobiHOC),* pp. 131–132, Aug. 2000.
5. J. Habetha and M. Nadler, Concept of a wireless centralised multihop ad hoc network. In *Proc. European Wireless,* (Dresden), Sept. 2000.
6. B. McDonald and T. F. Znati, A mobility-based framework for adaptive clustering in wireless ad hoc networks, *IEEE J. Select. Areas Commun.,* Vol. 17, pp. 1466–1487, Aug. 1999.
7. T. J. Kwon and M. Gerla, Clustering with power control. In *Proc. MILCOM,* Nov. 1999.
8. J. Habetha and M. Nadler, Outline of a centralised multihop ad hoc wireless network, *Computer Networks,* Vol. 37, pp. 63–71, 2001.
9. D. A. Baker and A. Ephremides, The architectural organization of a mobile radio network via a distributed algorithm, *IEEE Trans. on Commun.,* Vol. 29, pp. 1694–1701, Nov. 1981.
10. A. Ephremides, J. E. Wieselthier, and D. J. Baker, A design concept for reliable mobile radio networks with frequency hopping signaling, *Proc. of the IEEE,* Vol. 75, pp. 56–73, Jan. 1987.
11. J. Habetha and J. Wiegert, A comparison of new single- and multiple-transceiver data forwarding mechanisms for multihop ad hoc wireless networks. In *Proc. Int. Symposium on Performance Evaluation of Computer and Telecommunication Systems,* (Orlando), pp. 436–443, July 2001.
12. B. Walke, N. Esseling, J. Habetha, A. Hettich, A. Kadelka, S. Mangold, and J. Peetz, IP over wireless mobile ATM—Guaranteed wireless QoS by HiperLAN/2, *Proc. of the IEEE,* Vol. 89, pp. 21–40, Jan. 2001.
13. B. Walke, *Mobile Radio Networks,* 2nd ed., Wiley & Sons Ltd., Chichester, Sussex, UK, 2001.
14. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms,* Plenum, New York, London, 1981.
15. K. S. Fu, *Syntactic Methods in Pattern Recognition,* Academic Press, New York, 1974.
16. K. S. Fu, *Syntactic Pattern Recognition with Applications,* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.
17. L. A. Zadeh, Fuzzy sets, *Inform. Control,* Vol. 8, pp. 338–353, 1965.
18. D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications,* Academic Press, New York, London, Toronto, 1980.
19. H.-J. Zimmermann, *Fuzzy Set Theory—and Its Applications,* 3rd ed. Kluwer, Boston, Dordrecht, London, 1996.
20. J. C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact, well-separated clusters, *J. Cybernetics,* Vol. 3, No. 3, pp. 32–57, 1973.
21. G. H. Ball and D. J. Hall, ISODATA, an iterative method of multivariate analysis and pattern classification, *Behavioural Science,* Vol. 12, pp. 153–155, 1967.
22. R. Duda and P. Hart, *Pattern Classification and Scene Analysis,* Wiley, New York, 1973.
23. F. Höppner, F. Klawonn, and R. Kruse, *Fuzzy-Clusteranalyse: Verfahren für die Bilderkennung, Klassifikation und Datenanalyse,* Vieweg, Braunschweig, 1996.
24. D. E. Gustafson and W. C. Kessel, Fuzzy clustering with a fuzzy covariance matrix. In *Proc. of IEEE CDC* (San Diego, California), pp. 761–766, Jan. 1979.
25. I. Gath and A. B. Geva, Unsupervised optimal fuzzy clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 11, pp. 773–781, July 1989.
26. J. F. Peters, A. Skowron, Z. Suraj, S. Ramanna, and A. Paryzek, Modeling real-time decision-making systems with rough fuzzy Petri Nets. In *Proc. EUFIT'98* (Aachen), pp. 985–989, ELITE Foundation, 1998.
27. L. Minsky, A framework for representing knowledge. In R. Brachman and H. Levesque (eds.), *Readings in Knowledge representation* Morgan Kaufmann, Los Altos, 1985.
28. L. A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 16, No. 3, pp. 28–44, 1973.
29. O. Cordon, M. J. del Jesus, and F. Herrera, A proposal on reasoning methods in fuzzy rule-based classification systems, *Int. J. on Approximate Reasoning,* Vol. 20, No. 1, pp. 21–45, 1999.
30. L. I. Kuncheva, How good are fuzzy if-then classifiers? *IEEE Transactions on Systems, Man and Cybernetics*—Part B: *Cybernetics,* Vol. 30, pp. 501–509, Aug. 2000.
31. H. Bandemer and W. Näther, *Fuzzy Data Analysis,* Kluwer, Dordrecht, Boston, London, 1992.
32. A. Joentgen, L. Mikenina, R. Weber, and H.-J. Zimmermann, Dynamic fuzzy data analysis based on similarity between functions, *Fuzzy Sets and Systems,* Vol. 105, No. 1, pp. 81–90, 1999.
33. L. Mikenina, *Dynamic Fuzzy Pattern Recognition.* Ph.D. thesis, RWTH Aachen, 1999.
34. S. Mann, *Ein Lernverfahren zur Modellierung zeitvarianter Systeme mittels unscharfer Klassifikation.* Ph.D. thesis, Technische Hochschule Karl-Marx-Stadt, 1983.
35. B. Dubuisson, An adaptive decision system using pattern recognition, *Artificial Intelligence in Real-Time Control,* pp. 299–302, 1992.
36. S. Marsili-Libelli, Adaptive fuzzy monitoring and fault detection, *Int. J. of Condition Monitoring and Diagnostic Engineering Management (COMADEM),* Vol. 1, No. 3, pp. 31–37, 1998.
37. E. H. Mamdani and S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, *Int. J. Man-Machine Studies,* No. 7, pp. 1–13, 1975.
38. M. Junius, *Leistungsbewertung intelligenter Handover-Verfahren für zellulare Mobilfunksysteme.* Ph.D. thesis, RWTH Aachen, 1995.
39. G. Edwards, A. Kandel, and S. Ravi, Fuzzy handoff algorithms for wireless communication, *Fuzzy Sets and Systems,* Vol. 110, pp. 379–388, 2000.
40. D. Driankov, H. Hellendoorn, and M. Reinfrank, *An Introduction to Fuzzy Control,* 2nd ed., Springer, Berlin, 1996.
41. D. Johnson and D. A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth (eds.), *Mobile Computing* Kluwer Academic Publishers, Norwell, Massachusetts, 1996.
42. J. Broch, D. A. Maltz, D. B. Johnson, et al., A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM),* pp. 85–97, Oct. 1998.
43. S. Das, C. Perkins, and E. M. Royer, Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. IEEE INFOCOM,* Mar. 2000.
44. J. Habetha and D. Calvo de No, Hierarchical time-vector routing for mobile ad hoc networks. In *Proc. IEEE International Conference on Communications (ICC),* Helsinki, June 2001.



Jörg Habetha received the engineering diploma degree from Ecole Centrale, Paris, France, in 1996 and the diploma degree in electrical engineering from Aachen University of Technology, Aachen, Germany, in 1997 as he participated in the TIME (Top Industrial Managers for Europe) double diploma program among reputed European universities. He also holds a diploma degree in applied economics from Aachen University of Technology, Germany.

Mr. Habetha was a research scientist at the Chair of Communication Networks from 1997 to 2000, working toward the Ph.D. degree in wireless communications. Since October 2000, he has been with the Philips Research Laboratories, Aachen. His research interests include ad hoc and wireless LAN networks, and he is actively participating in the standardization of wireless LAN systems.

Bernhard H. Walke received the diploma and Ph.D. degrees in electrical engineering and communications engineering form the University of Stuttgart, Germany, in 1965 and 1975, respectively.

From 1965 to 1983, he was a researcher and department head in various industrial companies, where he designed computer-based communications networks and evaluated their traffic performance. In 1983, he joined the Department of Electronics Engineering, Fern-University of Hagen, as a Full Professor for Dataprocessing Techniques. In 1990, he moved to Aachen University of Technology, Aachen, Germany, as a Full Professor for Communication Networks. His current research covers air-interface design, protocols, stochastic performance simulation, and services of wireless and cellular radio systems. His scientific work comprises more than 80 scientific papers and five textbooks on modeling and performance evaluation of computer systems and communications networks. His most recent book is *Mobile Radio Networks* (Wiley, New York, 1999).

Prof. Walke is a Senior Member of IEEE as well as a member of ITG/VDE and GI and has served as program committee chairman of European conferences such as EPMCC and EW.