CONTROL AND USER PLANE SEPARATION IN AD-HOC NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DOĞANALP ERGENÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2018

Approval of the thesis:

CONTROL AND USER PLANE SEPARATION IN AD-HOC NETWORKS

submitted by **Doğanalp Ergenç** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Ertan Onur _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. İbrahim Körpeoğlu
Computer Engineering Department, Bilkent University _____

Assoc. Prof. Dr. Ertan Onur
Computer Engineering Department, METU _____

Assist. Prof. Dr. Hande Alemdar
Computer Engineering Department, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Doğanalp Ergenç

Signature :

ABSTRACT

CONTROL AND USER PLANE SEPARATION IN AD-HOC NETWORKS

Ergenç, Doğanalp
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Ertan Onur

September 2018, 120 pages

Separation of user (data) plane from control plane in networks helps scale resources independently, increase the quality of service and facilitate autonomy. In ad-hoc networks, the plane-separation through clustering introduces a hierarchy where control functions can be carried out by some designated cluster heads and other nodes perform according to the outcomes of those functions. Therefore, clustered topologies can be considered as a natural consequence of the control and user plane separation (CUPS). Moreover, hierarchical routing protocols, which are constructed upon the clustered topologies, enable the use of CUPS architecture for the end-to-end communication. However, there is no silver bullet to apply clustering algorithms that are directly dependent on the network characteristics, and the routing protocols designed for clustered topologies cannot effectively utilize CUPS since they neglect the role of the nodes in the data plane. This study investigates the application of CUPS architecture in ad-hoc networks by considering clustering and routing protocols holistically. First, the adaptability of the clustering techniques is discussed to satisfy different objectives such as stability, energy efficiency and service quality; and Dependability-

based Clustering Algorithm (DCA) is proposed. DCA is a dynamic clustering algorithm that exploits a cross-layer architecture. Its different parameters are analyzed and optimized using the sensitivity analysis technique, Moment-Independent Delta Analysis. Then, the hierarchical routing protocol CUPS-based Hierarchical Routing Algorithm (CHRA) is proposed for end-to-end communication. In CHRA, the separate functions of the control and data planes are explicitly defined to provide the quality of service and energy efficiency taking advantage of the clustered topology. The overall CUPS-centric framework including DCA and CHRA is implemented in the discrete event-based simulator, OMNeT++. The results show that DCA outperforms its opponents when it is optimized for different scenarios. Besides, the study reveals the significant points that are required to be considered for designing clustering algorithms through the discussion of the optimization process. Finally, CHRA offers a better quality of service and a fair energy consumption thanks to its novel approach that considers the effective use of the data plane as well as the control plane. The complete plane-separated approach is utilized for energy efficiency and the quality of service in ad-hoc networks.

Keywords: cross-layer, clustering, ad-hoc, routing, control and user plane, CUPS

ÖZ

TASARSIZ AĞLARDA KONTROL VE KULLANICI DÜZLEMLERİNİN AYRILMASI

Ergenç, Doğanalp
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ertan Onur

Eylül 2018 , 120 sayfa

Kontrol ve veri düzlemlerinin ayrımı ağlarda kaynakların bağımsız şekilde ölçeklendirilmesine, servis kalitesinin arttırılmasına ve otonomluğun sağlanmasına yardım eder. Tasarsız ağlarda, öbekleme vasıtası ile düzlem ayrımı, kontrol işlevlerinin belirlenmiş bazı öbek başları tarafından yürütüldüğü ve kalan düğümlerin bu işlevler sonucuna göre hareket ettiği sıradüzeni sunar. Bu sebepten, öbekli topolojiler kontrol ve kullanıcı düzlemi ayrımının doğal bir sonucu olarak görülebilir. Dahası, öbekli topolojiler üzerine tasarlanan sıradüzenli yönlendirme protokolleri kontrol ve kullanıcı düzlemi ayrık mimarilerin uçtan uca iletişimde kullanımının önünü açar. Fakat isterleri farklı senaryolara göre belirlenen öbekleme algoritmalarının uygulamasında belirli bir kural olmadığı gibi, öbekli ağlar için tasarlanan yönlendirme protokolleri veri düzlemini oluşturan düğümlerin rolünü ihmal ettiğinden ayrık düzlemlerden etkili şekilde faydalanamamaktadır. Bu çalışma, öbekleme ve yönlendirme protokollerinin bütüncül değerlendirilmesiyle tasarsız ağlarda kontrol ve veri düzlemi ayrımının faydalarını tartışmaktadır. İlk olarak kararlılık, enerji verimliliği ve güvenilirlik is-

terlerine uygun bir öbekleme algoritmasının uygulanabilirliğini araştırıp, katmanlararası mimariyle dinamik olarak farklı isterlere adapte olabilen öbekleme algoritması DCA sunulmuştur. DCA'nın parametreleri hassaslık analiziyle değerlendirilmiş ve eniyilenmiştir. Sonrasında, kontrol ve veri düzlemi işlevlerinin net olarak belirlendiği, servis kalitesi ve enerji verimliliğine yoğunlaşan sıradüzenli yönlendirme protokolü CHRA önerilmiştir. Kontrol ve veri düzlemi ayrımını esas alan bütün mimari, ayrık olay eksenli gerçekleyici OMNeT++ kullanılarak programlanmıştır. Sonuçlar, farklı senaryolar için eniyilendiğinde DCA'nın performansının rakiplerinden daha iyi olduğunu göstermiştir. Ayrıca eniyileme sürecinin tartışılması, benzer algoritmaların tasarımında dikkat edilmesi gereken noktaları gözler önüne sermiştir. CHRA ise yönlendirmede kullandığı yeni yaklaşımlarla servis kalitesini geliştirip adil enerji kullanımı sağlamıştır. Sonuç olarak, bütüncül bir düzlem-ayrımı yaklaşımının tasarsız ağlarda enerji verimliliğini ve servis kalitesini arttırdığı gösterilmiştir.

Anahtar Kelimeler: katmanlar-arası, öbekleme, tasarsız, yönlendirme, kontrol ve kullanıcı düzlemi

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Assoc. Prof. Ertan Onur of the Computer Engineering Department at Middle East Technical University. The door to Prof. Onur office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently steered me in the right direction whenever I needed it. He is an advisor, a mentor and an idol for me through my academic career. Besides, I greatly appreciate the feedback offered by Dr. Tolga Numanoğlu. He significantly helped me to improve numerous technical aspects of my study.

Finally, I must express my very profound gratitude to my parents and to my darling Sude for providing me an unfailing support and a continuous encouragement throughout my years of study and through the process of researching and writing this thesis. I would also like to thank Cem, my concord and fellow, for sharing all the stress and exhaustion through this challenging journey. This accomplishment would not have been possible without them. Thank you all.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

DCA	Dependability-based Clustering Algorithm
CHRA	CUPS-based Hierarchical Routing Algorithm
SDN	Software-defined Network
IETF	Internet Engineering Task
MANET	Mobile Ad-hoc Network
LCA	Linked Cluster Algorithm
LCC	Least Cluster Change
ACA	Adaptive Clustering Algorithm
HEED	Hybrid Energy-Efficient Distributed
LEACH	Low Energy Adaptive Cluster Hierarchy
HCC	High Connectivity Clustering
3-hBAC	3-hop Betweeen Adjacent Clusterheads
PC	Passive Clustering
DDCA	Distributed Dynamic Clustering Algorithm
MCSVANET	Multi-hop Clustering Scheme for Vehicular Ad-Hoc Networks
SBCA	Score Based Clustering Algorithm
WCA	Weighted Clustering Algorithm
WBCA	Weight-based Clustering Algorithm
AMACAD	Adaptable Mobility-Aware Clustering Algorithm based-on Destination
DMCNF	Distributed Multi-hop Clustering Algorithm for vehicular ad-hoc networks (VANETs) based on Neighborhood Follow
CLPSO	Comprehensive Learning Particle Swarm Optimization

MOPSO	Multi-Objective Particle Swarm Optimization
AWCP	Adaptive Weighted Clustering Protocol
NSGA-II	Non-dominated Sorted Genetic Algorithm version 2
OLSR	Optimized Link State Routing Protocol
DSDV	Destination Sequence Distance Vector
AODV	Ad-hoc On-demand Distance Vector
ZRP	Zone Routing Protocol
CBRP	Cluster-based Routing Protocol
CLACR	Core-location Aided Cluster-based Routing
CIDR	Cluster-based Inter-domain Routing
BGP	Border Gateway Protocol
Cross-CBRP	Cross-layer Cluster-based Routing Protocol
ZHLS	Zone-based Hierarchical Link State Routing Protocol
GPS	Global Positioning System
CMDSR	Cluster-Based Multipath Dynamic Source Routing
HOLSR	Hybrid Optimized Link State Routing Protocol
HCR	Hybrid Cluster Routing Protocol
ARAMA	Ant Routing Algorithm for Mobile Ad-hoc Networks
HLANMAR	Hierarchical Landmark Ad-hoc Routing
LANMAR	Landmark Ad-hoc Routing
API	Application Programming Interface
CSMA	Carrier-sense Multiple Access
TDMA	Time-division Multiple Access

CHAPTER 1

INTRODUCTION

Ad-hoc networks are fundamentally predicated on multi-hop communication. Its history can be tracked to 500 BC where Persian King Darius the First had used intermittently located archers for communication instead of mounted scouts [1]. The King decreased end-to-end delay nearly 95% using this method. After more than 1900 years when hop-to-hop communication is born, wireless ad-hoc networks appear to be discussed in Hawaii, with the project ALOHANet. ALOHANet and its posterior PRNET project aimed to design multi-hop wireless communication infrastructure onto relatively larger geographical areas; and formally Internet Engineering Task (IETF) Mobile Ad-hoc Networks (MANET) working group started to develop an open-source standard in this subject.

In wireless communication, there are two major approaches: cellular networks and ad-hoc networks. The fundamental idea behind ad-hoc networks is having no infrastructure. That is, a pre-deployed infrastructure is not required in ad-hoc networks and the network is generally managed distributedly with end-point rather than an infrastructure such as base stations in cellular networks. The differences between cellular and ad-hoc networks are summarized in Table 1.1.

Ad-hoc networks are significantly used in the emergency, disaster and army-tactical scenarios. Packets in such networks are forwarded between nodes themselves due to multi-hop nature of the architecture. In ad-hoc networks with flat topology, scalability is one of the important problems considering such multi-hop communication. Organization and management of the unlimited number of nodes dynamically (i.e.,

Table 1.1: Differences between cellular and ad-hoc networks [1].

Cellular networks	**Ad-hoc networks**
Fixed infrastructure	Infrastructureless
Single-hop links	Multi-hop links
Guaranteed bandwidth	Shared radio channel
Centralized routing	Distributed routing
Reliable connection	Frequent de-linking under mobility
High cost and time of deployment	Quick and cost-effective deployment
Spatial frequency reuse	Dynamic frequency reuse
Easy synchronization	Bandwidth-required synchronization
High cost of network maintenance	Self-organized networks
Scenario-specific architecture	Adaptable architecture

without any central controller) are other major challenges. For instance, the size of a routing table directly depends on the number of nodes in a network. The management of the tables is getting harder with the increasing population of the network and eventually control traffic outnumbers the actual data traffic. Similarly, resource allocation and link scheduling to orchestrate overall communication are not easy to handle in the absence of the centralized control mechanisms. A number of techniques and protocols has been proposed to overcome those challenges for years.

Control and user plane separation (CUPS), on the other hand, has become popular in a near future for next-generation cellular networks [2][3][4] and software-defined networks (SDN) [5][6]. When the networks become more heterogeneous and the number of network-based services has swelled, the flexible network architectures which can be easily scaled and modified to satisfy both networks' infrastructural requirements and user-level agreements have to be considered. In CUPS, the control plane consists of (physically or logically) central control elements that have a manager role over data plane elements. While network configurations and dynamic policies are set via the control plane, the data plane performs befittingly such decisions taken by the control plane. For instance, while forwarding rules and bandwidth limit for a user are regulated in the control plane, forwarding the packets in limited flow rate is actuated by

the data plane. Moreover, since some other critical components of the network such as security and monitoring modules are placed in the control plane, they are able to operate in a wider scope where all data is passing through, the data plane [7].

The separation of the planes in such a way naturally leads to a hierarchy where some components of the network (i.e., data plane elements) perform according to some others' decisions (i.e., control plane elements). In ad-hoc networks, *clustering* directly corresponds to that hierarchical structure. It can be defined as grouping of nodes based on some common properties and is commonly employed for achieving scalability and manageability [8][9]. Nodes in a clustered topology are categorized into different roles according to their functionality. While cluster heads (CHs) are the main nodes that possess the local neighborhoods (or clusters), ordinary nodes are gathered around cluster heads. Cluster heads can manage routing, scheduling, data aggregation etc. [10][11] to orchestrate the group of nodes, and ordinary nodes communicate under this management in the hierarchy. Therefore, CHs form the control plane conducting other nodes in clusters and ordinary nodes reside in the data plane realizing the network communication under CHs', or control plane's leading. Eventually, it is separated as "controller" nodes and "user" or ordinary nodes and it leads us to form the control and user planes where the nodes have different importance and roles with respect to the plane they are logically lying on.

The techniques for end-to-end communication in ad-hoc networks are also directly affected and inspired by the hierarchical structure that is a consequence of the CUPS. Routing algorithms, which are designed for clustered structure, use CHs and gateways for routing discovery and maintenance; and also data forwarding since those are the main nodes that are aware of the topology. However, they overload CHs with not only control plane functions but data plane functions. It eventually imposes a high amount of load on the specific nodes (i.e., CHs and gateways), and also causes losing many other alternative paths that can be defined through ordinary nodes. Therefore, the separation of control and data plane, and using them collaboratively for end-to-end communication may lead to a much more effective routing in terms of energy efficiency and the quality of service.

1.1 Problem Definition

Through the thesis, various problems are addressed from the CUPS perspective. In clustering algorithms, the challenge is generally CH selection to dynamically manage ad-hoc networks. Different algorithms focus on different aspects of nodes such as mobility, energy, location-based requirements of the scenarios for which they are designed. The evaluation of those criteria reveals which node is more suitable for being a CH. That is, they are considered to understand nodes' eligibility to be a group leader. However, clusters themselves are also needed to be evaluated to understand the effectiveness of related clusters, and also to be able to compare them. Therefore, it is important to define some measures to comprehend the effectiveness of clustering and find some techniques to analyze the relative importance of those measures.

CHs in hierarchical structures naturally bring a variety of advantages for management of the network. However, using them to find routes and forward data together exhausts those specially-selected nodes, i.e., CHs and gateways and that is the most important drawback for the most of hierarchical routing algorithms. In those algorithms, many possible routes that can be defined by ordinary nodes are neglected while focusing on CHs. Therefore, the hierarchy in a network is not being used effectively. Moreover, while some of the studies are directly coupled to clustering process, others require some special nodes (e.g., having longer transmission ranges, GPS) or central pre-deployed mechanism (e.g., routing servers and managers) to maintain routing and clustering processes.

1.2 Motivation and Scope

CUPS is popularly studied in different areas such as mobile cellular networks and SDN. It is also really suitable to compensate for the absence of a centralized controller and infrastructure in ad-hoc networks by creating a dynamic and distributed management scheme through clustering. Even though clustering is a primitive application of such design, clustering algorithms are generally designed for very specific purposes

and make lots of assumptions. Moreover, the hierarchical routing algorithms, which are designed coupled with clustering algorithms, cannot use this structure effectively. Therefore, a complete solution for ad-hoc networks exploiting the whole nature of the CUPS is required to be designed and evaluated. In the thesis, the main motivation is to reshape an overall clustering and routing architecture from the CUPS perspective for a flexible, energy-efficient and high-performance ad-hoc network design. It consists of two main parts. The first one is the formation of the CUPS architecture. The technique is a weighted clustering algorithm, Dependability-based Clustering Algorithm (DCA). In that part, DCA is designed and analyzed, also compared with other clustering algorithms. Through the analysis of the algorithm, a complete sensitivity analysis framework is proposed to evaluate the impacts of different parameters in DCA. Therefore, it is the part that the backbone of the CUPS architecture, i.e., construction of the hierarchy and control plane, is discussed in length and breadth. The second part is the establishment of end-to-end communication scheme on the top of the CUPS architecture. The details of the control and user plane-separated routing algorithm, which is CUPS-based Hierarchical Routing Algorithm (CHRA), for clustered ad-hoc networks are presented here. It is a discussion of the effective use of CUPS for the energy efficiency and end-to-end communication performance. Both parts are jointly implemented in the discrete event-based network simulator OMNeT++. Apart from using built-in methods of the simulator, a cross-layer stack is also designed from scratch.

The thesis is organized as follows. In Chapter 2, similar studies for clustering and routing in ad-hoc networks are presented and categorized. Besides, different cross-layer architectures and some example work are given. In Chapter 3, the design artifacts, which are the common terminology and implementation details, used in both DCA and CHRA are presented. In Chapter 4, the details and dynamics of the clustering algorithm DCA are discussed. Chapter 5 presents the fundamentals of CHRA step by step with descriptive figures. In Chapter 6, the overall performance evaluation of DCA and CHRA is given. Besides, the sensitivity analysis for DCA is presented in this chapter. In the end, Chapter 7 presents the general evaluation of the whole design and results, and also a discussion for possible improvements and future work.

1.3 Contributions

The applications and extensions of the CUPS architecture are investigated for (1) the management of ad-hoc networks and (2) establishment of end-to-end communication scheme in CUPS. Eventually, DCA and CHRA are designed jointly to constitute a complete CUPS-centric solution to satisfy (1) and (2). To realize and evaluate such a complete design,

- A new and flexible topological structure, cluster sight area (CSA), is proposed to discover a limited area in the network to be able to find end-to-end routes in the data plane proactively. It is basically a super-structure that is formed by a number of clusters.
- The whole design is implemented in the discrete event-based simulator OMNeT++, and it is compared with opponent clustering and routing algorithms separately.

For the formation of CUPS architecture in ad-hoc networks,

- DCA is proposed considering both nodes' and clusters' benefits. The node and dependability scores are defined to evaluate (a) eligibility of a node to be cluster head and (b) *dependability* of a cluster so that a node can select the best one among neighbor clusters to increase its own chance to get a guaranteed resource with high stability and reliability.
- The term "dependability" is propounded to be able to evaluate and compare the clusters themselves. Basically, it is a cluster-related measure that is considered by nodes to be able to decide the cluster they join to maximize their own benefit.
- The analytic method, Moment-independent Delta Analysis [12], is embodied to evaluate the impacts of weighted metrics that are used to calculate the node and dependability scores to optimize performance metrics in a weighted clustering algorithm. This method is directly used for DCA and proposed as a generic framework to evaluate any other weighted clustering algorithm.

- Through the sensitivity analysis, the significant metrics that need to be considered for designing a weighted clustering algorithm are revealed. Since the advantages of those metrics depend on different use cases, their applicability is discussed considering different goal-based requirements.
- DCA is evaluated for different use cases that aim high-stability, low and fair energy consumption, high quality of service (QoS). Two different versions of DCA and other benchmark clustering algorithms are also implemented for the performance evaluation and comparison.

For the establishment of an end-to-end communication scheme in hierarchical ad-hoc networks,

- A CUPS-centric routing algorithm, CHRA, is presented for ad-hoc networks as a natural extension of the clustering for energy efficiency and quality of service.
- New techniques for route recovery in hierarchical routing are proposed focusing on the communication in the data plane.

1.3.1 Publications

There is a number of outputs of this thesis. Some of them form the major parts of the thesis while others are prepared as results of a continuous thinking and redesign iteration throughout the thesis. They are listed as,

- Ergenc, D., Eksert, L., & Onur, E. (2018). Density-Aware Probabilistic Clustering in Ad Hoc Networks. *In Proc. of the IEEE International Black Sea Conference on Communications and Networking(BlackSeaCom).*, doi:10.1109/blackseacom.2018.8433605
- Ergenc, D., & Onur, E. (2018). Cross-layer Stack Design Framework in OMNeT++. *in Proc. of the 5th OMNeT++ Summit.* (to be appeared)
- Ergenc, D., & Onur, E. (2018). CUPSMAN: Control User Plane Separation Based Routing in Ad-hoc Networks. *ArXiv e-prints.*, eprint:1807.10747

CHAPTER 2

RELATED WORK

In this chapter, the major studies for clustering and routing protocols in ad-hoc networks are presented. Besides, the well-known cross-layer design principles are discussed giving examples from the literature. Each section presents the literature review on the related subject and is summarized with a discussion table.

2.1 Clustering

In the literature, there is a number of clustering algorithms that are designed for different environments and with different decisive purposes. Each algorithm aims to optimize different performance measures such as power consumption and control overhead. Therefore, they are applicable only to some specific types of ad-hoc networks. In this section, some of those studies are selected to cover a wide range of algorithms that represent different approaches.

There is not a common classification scheme for clustering algorithms. In this study, they are divided into five categories that are:

1. **Identifier-based Clustering**: In this approach, each node in the network has a unique identifier (ID) and clusters are formed with respect to those IDs. Most of the cases, identifier-based clustering algorithms are seen as a random clustering technique since the IDs are assigned randomly. Generally, this type of clustering algorithms tends to be light-weight and easy-to-implement. However, since

they ignore many important network characteristics like energy consumption, mobility etc., they do not fit every scenario with different requirements.

2. **Energy-based Clustering**: Energy-based clustering algorithms generally use residual energy as the main metric to select cluster heads. Even though they use the same metric, while some of them focus on fair energy consumption by changing cluster heads with respect to their residual energy, others directly promote the nodes with the highest energy to be CH.

3. **Topology-based Clustering**: Topology-based clustering algorithms take advantage of topology-related information to select CHs and eventually form clusters. In this algorithms, the topology information (e.g., location, formation and neighborhood-related information) is one of the most considered criteria to select more central nodes that can manage high number of nodes as a CH.

4. **Mobility-based Clustering**: In this approach, the main concerns are the formation and the maintenance of the clustered structure in the networks with different mobility characteristics. Ad-hoc networks are self-organized (i.e., no central control mechanism) and such organization becomes harder when nodes are not stable. Therefore, mobility-based clustering algorithms investigate the methods to orchestra mobile ad-hoc networks in a reliable way.

5. **Score-based Clustering**: When there are multiple parameters that need to be focused for an efficient design, clustering algorithms have to take such parameters into consideration at the same time. For example, if one needs to design an energy-efficient algorithm for high-mobility ad-hoc networks, mobility and energy-consumption of nodes may be considered to form clusters. Score-based clustering approach gathers different requirements and calculates a score as a combination of different parameters. It can be seen as a hybrid approach to form clusters where nodes are evaluated with respect to their scores to be CH.

6. **Optimization-based Clustering**: Optimization-based clustering methods transform the clustering process into some optimization problems and attempt to optimize different configuration parameters of clustering algorithms to improve their performance in terms of different measures.

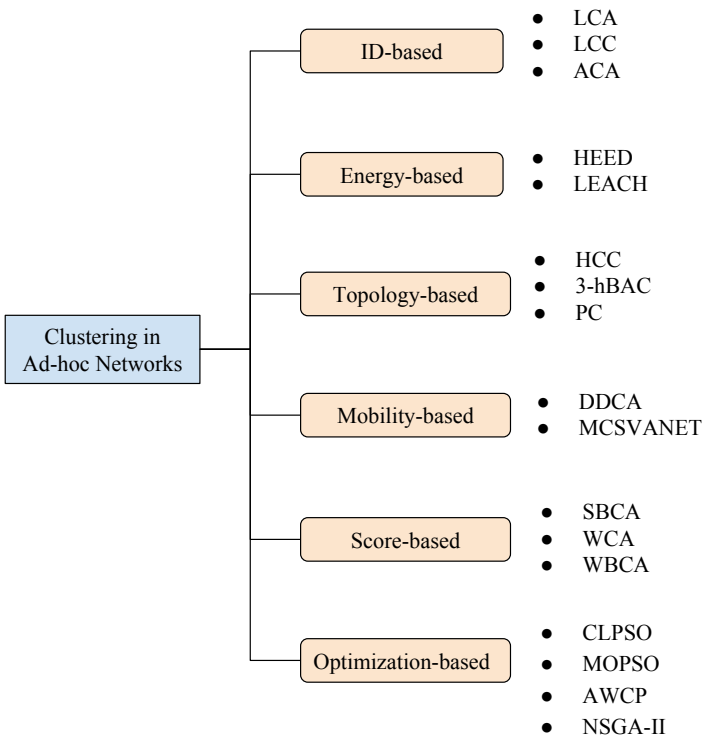


Figure 2.1: The classification of the clustering algorithms

In the rest of this section, some major studies, which fall under those categories, are presented. Figure 2.1 shows the classification of those clustering algorithms.

2.1.1 Identifier-based Clustering Algorithms

Identifier-based clustering algorithms use unique IDs of the nodes as the main parameter to form clusters. Linked Cluster Algorithm (LCA) [13] is one of the most fundamental clustering algorithms. In LCA, each node broadcasts its own and neighbors' IDs periodically and the node with the lowest ID is selected as a CH. While LCA is a quite simple and low-complexity clustering algorithm, since it forces only some particular nodes to be CH, those nodes tend to be left with drained battery quicker than the others.

Least Cluster Change (LCC) [14] has a similar CH selection technique to LCA. However, it also considers if (a) CHs are getting closer to each other, and (b) ordinary nodes are getting further from their cluster heads to avoid frequent reclustering pro-

cesses and prevents ordinary nodes from challenging the cluster heads in some scenarios. It decreases the changes in clustered formation and creates more stable clusters. However, still its maintenance phase depends on node IDs and it tends to change cluster formation any time considering randomly assigned IDs.

Adaptive Clustering Algorithm (ACA) [15] is another alternative to LCA. In ACA, the clusters are formed using the similar lowest-ID method, but any node does not behave as a cluster head. Instead, the clustered structure is used to create cell-like (i.e., internally-organized cells in cellular networks) structure where each one uses different signal frequencies to promote spatial frequency reuse. Even if it is increasing resource efficiency, ACA lacks the hierarchical structure that makes an ad-hoc network manageable.

2.1.2 Energy-based Clustering Algorithms

Hybrid Energy-Efficient Distributed (HEED) clustering [16] is designed for quasi-stational sensor networks in order to aggregate data to a centralized entity through cluster heads. In the algorithm, every node elects itself as a cluster head with a probability depending to the residual energy or joins the cluster head with the strongest received signal strength as a repetitive process.

Low Energy Adaptive Cluster Hierarchy (LEACH) [17] is proposed for enhancing scalability and robustness in the data transmission flow that transferring data from the wireless microsensor nodes to the base station through the cluster heads of each cluster. In each round, nodes acquire the chance of being cluster head role consecutively, leading to an evenly distributed load of cluster head role and additional energy consumption among the homogeneous sensors. However, both HEED and LEACH only focus on energy metric and do not take into consideration any of the performance parameters of communicating nodes such as mobility and signal strength.

2.1.3 Topology-based Clustering Algorithms

High Connectivity Clustering (HCC) [18] basically selects the node with the highest number of neighbors as CH. Its main purpose to decrease the total number of clusters so that the network can be managed with less control overhead. However, since it is significantly affected by topology changes, HCC has a high reclustering overhead in high-mobility ad-hoc networks.

Another topology-based clustering algorithm that selects the node with highest-degree as CH is 3-hop Between Adjacent Clusterheads (3-hBAC) [19]. In 3-hBAC, after clusters are formed considering the connectivity of nodes, new nodes (i.e., nodes freshly participate to the network) and other nodes that do not belong to related cluster are called as "cluster guest". Those nodes can connect to CHs indirectly via an intermediate node, and still can be managed by CHs that they are indirectly connected to. Therefore, nodes can be related to further cluster heads instead of creating new clusters and eventually the number of clusters decreases. However, since there need to be intermediate nodes to maintain clustered structure, 3-hBAC is quite sensitive to mobility.

Passive Clustering (PC) [20] has a different approach to form clusters: instead of performing a clustering process with control packets, nodes piggyback its own cluster-related information. Any node receiving those packets becomes aware of the possible clusters in its neighborhood. In PC, the nodes sending more data packets with piggybacked control information have a higher probability to be cluster head in their neighborhood. Even if it decreases the control overhead, it is quite hard to form and maintain clustered structure in the networks with low communication rate with PC.

2.1.4 Mobility-based Clustering Algorithms

Distributed Dynamic Clustering Algorithm (DDCA) [21] focuses on the mobility of the nodes and creates multi-hop clusters using (α, t) metric. This metric represents if a node has a path through the related cluster head with probability α during t seconds.

While this approach leads to the formation of multip-hop clusters in low-mobility networks, the probability of single-hop cluster existence is much higher in high-mobility networks. Eventually, DDCA provides a stable and mobility-sensitive cluster scheme bringing the maintenance cost for multi-hop clusters.

Multi-hop Clustering Scheme for Vehicular Ad-Hoc Networks (MCSVANET) [22] is another mobility-based clustering algorithm that focuses on relative mobility.
MCSVANET assumes that if a node has different mobility characteristic than the others, it tends to change current cluster formation. Therefore, it tries to form clusters among the nodes that have similar mobility patterns, and one of the least mobile nodes is selected as cluster head. Those clusters may be single or multi-hop. The relative mobility is measured by calculating the difference of signal strength between consecutive control beacons. While depending only on signal strength is not a reliable measure considering different channel conditions, the extra beacons also increase clustering overhead.

A mobility-based approach, MOBIC [23], which has a similar mechanism with identifier based clustering [15], uses relative mobility as the main metric for cluster formation. The node which has the least relative mobility among surrounding nodes announces itself as a cluster head where the other nodes in its vicinity become cluster members.

2.1.5 Score-based Clustering Algorithms

Score-based Clustering Algorithm (SBCA) [24] is a weighted clustering algorithm that uses residual energy, the number of neighbors and node stability to calculate a node score to select cluster heads. It aims to decrease the number of clusters and increase energy efficiency. However, the calculation of score is still ambiguous and need to be optimized to get the targetted results effectively.

Weighted Clustering Algorithm (WCA) [25] aims to create a dominating set of network graph with the cluster heads. WCA elects cluster heads according to the weighted sum of ideal node degree, transmission power, mobility and battery power metrics of

nodes. The reelection of cluster heads is invoked as the relative distance between the cluster head and ordinary nodes changes. In contrast, Weight-based Clustering Algorithm (WBCA) [26] calculates the weighted average of consumed energy and degree difference among the neighboring nodes periodically and chooses the node with the least values among its one-hop neighbors as the cluster head. Weight-based clustering algorithms basically construct the clusters based on one or a combination of performance parameters of the communicating nodes. The weights of these parameters are determined by the user experiences and performance results of the algorithms are measured with fixed and predetermined weights.

2.1.6 Optimization-based Clustering Algorithms

Swarm or evolutionary optimization algorithms are used in order to maximize some quality measures of the network predetermined by the clustering methods. In Comprehensive Learning Particle Swarm Optimization (CLPSO) [27], given a MANET, a set of cluster formation in WCA [25] is generated and the weights in WCA are optimized with a swarm optimization algorithm in order to maximize a single objective, the total score of the cluster heads. In contrast, in [28], Multi-Objective Particle Swarm Optimization (MOPSO) is proposed to optimize the number of clusters in a MANET as well as energy-consumption in nodes in order to provide energy efficiency and reduce the network traffic. Degree difference, energy consumption, mobility, and transmission range are determined as the objectives and the performance of MOPSO is tested and compared to that of WCA and CLPSO. Although MOPSO outperforms WCA and CLPSO by finding relatively more optimal number of clusters, neither of these optimization approaches offer an extensive cluster maintenance mechanism as introduced in the proposed method. Various swarm optimization techniques such as ant colony optimization [29] and grey wolf optimization [30] are applied for VANET clustering as well. Similar to MOPSO, both techniques try to optimize the number of cluster heads in the network and neither of them does not have a distributed cluster head reelection, cluster maintenance, and cluster discovery mechanisms. However, another study introduced in [31] differs from the aforementioned optimization

methods, presenting a comprehensive clustering method, Adaptive Weighted Clustering Protocol (AWCP) in which cluster head election, cluster maintenance, and cluster merging operations are defined. An evolutionary optimization algorithm, Non-dominated Sorted Genetic Algorithm version 2 (NSGA-II) [32], aims to regulate a set of clustering parameters of AWCP in order to optimize three objectives: average cluster lifetime, packet delivery ratio, and control packet overhead. The clustering method is run on a network simulator and simultaneously reconfigured by the optimization algorithm operated on an optimization tool. Although this technique devises a set of clustering parameters and weights similar to the proposed method, the study does not investigate the correlation between any of the clustering parameters and the performance measures. Furthermore, stationary scenarios are not tested in network simulation, preventing to observe the influence of node mobility on the weight values and performance metrics.

There are also other clustering algorithms that do not fall under those categories. Some methods provide cluster head reelection, cluster maintenance, and cluster discovery mechanisms as well as cluster formation as a whole. For instance, Adaptable Mobility-Aware Clustering Algorithm based on Destination positions (AMACAD) [33] uses GPS and destination data in order to form clusters and elect cluster heads whereas Distributed Multi-hop Clustering Algorithm for vehicular ad-hoc networks (VANETs) based on Neighborhood Follow (DMCNF) [34] propagation delay for composition and maintenance of n-hop clusters. In [35], a clustering algorithm is proposed to generate and maintain the clusters as well as to suggest a mechanism of merging clusters for IEEE 802.11p and LTE hybrid network architecture.

Each algorithm in different categories has its own advantages and disadvantages. The summary of all those clustering algorithms are presented in Table 2.1 in terms of a number of different comparison metrics. *Radius* shows the coverage area of clusters and it is inversely proportional to *Number of Clusters*. *Stability* basically represents how much clustered formation tends to change. *Mobility* shows the tolerance of algorithms to the changes stem from moving nodes. While *energy efficiency* represents if the algorithm promotes efficient energy use for nodes, *Load-balancing* shows

if the traffic is distributed fairly between nodes considering intra- and inter-cluster communication. Lastly, the control overhead for clustering process represented with *Overhead*. Since optimization-based clustering algorithms are not directly clustering designs but optimization methods (mostly from different study areas than wireless networking), they are not presented in Table 2.1.

Table 2.1: The comparison of clustering algorithms

Algorithm	**Category**	**Radius**	**Stability**	**Mobility**	**energy efficiency**	**Load balancing**	**Num. of Clusters**	**Overhead**
LCA [13]	Identifier	1-hop	Low	Very low	Very low	Low	High	High
LCC [14]	Identifier	1-hop	Low	Very low	Low	Low	High	High
ACA [15]	Identifier	1-hop	Low	Low	Low	Low	High	High
HEED [16]	Energy	1-hop	Low	Medium	High	Low	Medium	Low
LEACH [17]	Energy	1-hop	Medium	Low	Medium	High	Medium	High
HCC [18]	Topology	1-hop	Very low	Very low	Very low	Low	Low	High
3-hBAC [19]	Topology	2-hop	Low	Medium	Low	Low	Medium	High
PC [20]	Topology	1-hop	Low	High	High	Low	Low	Low
DDCA [21]	Mobility	n-hop	High	High	Low	Low	Low	High
MCSVANET [22]	Mobility	n-hop	High	High	Low	Low	Low	High
SBCA [24]	Score	1-hop	Medium	Medium	Medium	Low	Low	High
WCA [25]	Score	1-hop	Medium	Medium	Medium	Medium	Medium	High
WBCA [26]	Score	1-hop	Medium	Low	Medium	Low	Low	High

2.2 Routing

Routing algorithms in ad-hoc networks are primarily divided into four categories with respect to their route discovery and maintenance techniques. Those categories are,

1. **Proactive (Table-driven) Routing**: In this approach, the nodes, which actively participate to routing, send topology information (or available routes) of all discovered nodes periodically. Therefore, the direct and indirect (one-hop and multi-hop) reachability information of the nodes are kept updated and fresh. Since routes are constantly maintained, any source node becomes able to send packets to a destination node spontaneously without extra delay for route discovery. In contrast, the cost of continuous maintenance in terms of resource occupation is the major drawback of the proactive routing algorithms. Optimized Link State Routing Protocol (OLSR) [36] and Destination Sequence Distance Vector (DSDV) [37] are very well-known examples of this category.

2. **Reactive (On-demand) Routing**: In this approach, nodes send route request packets to their neighbors whenever they need to communicate with other nodes that are further than a single hop. The originator node then evaluates alternative paths which are obtained from route response packets and selects the best option e.g., the shortest one. Even though the control overhead is relatively less than the proactive approach, the end-to-end delay per communication increases due to discovery process being performed right before the traffic demand. Ad-hoc On-demand Distance Vector (AODV) [38] is one of the on-demand routing protocols.

3. **Hybrid Routing**: Hybrid routing algorithms take advantage of both proactive and reactive routing techniques. Zone Routing Protocol (ZRP) [39] is a standard hybrid routing algorithm in the literature.

4. **Hierarchical Routing**: Although it is quite similar to the hybrid approach, hierarchical routing considers different roles of the nodes i.e., hierarchies to manage route discovery and maintenance processes. Cluster-based Routing Protocol (CBRP) [40] is one of the pioneer examples of this approach.

CBRP, which is one of the fundamental hierarchical routing algorithms, might be important to discuss the relationship between clustering and routing. Similar to LCC and LCA, the node with the lowest ID in each group is selected as CH in CBRP. All nodes discover their 2-hop neighborhood. Different nodes take roles as CH and gateway in each cluster, and diversity in the roles directly affects tasks of the nodes in routing. For example, since gateways are connected to multiple CHs by definition, they have a significant role for inter-cluster communication. Similarly, CHs manage intra- and inter-cluster communication via gateways. Eventually, the route discovery process is mostly handled by those nodes with specific roles. Moreover, CHs are responsible for the management of the routes and orchestration of their cluster members [40].

CBRP basically offers a divide-and-conquer approach to deal with the challenges of routing in ad-hoc networks and this is the fundamental idea of all hierarchical routing protocols. Broadcast packets for route discovery are only used in intra-cluster communication. In contrast, CHs and gateways use unicast and multicast packets to find inter-cluster routes. This method helps to detect topology changes locally and increase stability. It also decreases the number of broadcast packets for an efficient use of bandwidth [41].

As most of the ad-hoc networks are lack of pre-deployed infrastructure for the network management, protocol-based solutions are utterly in need to overcome many restrictions. Even though CBRP offers a degree of scalability, requirements and limitations such as power consumption, mobility level, and topology characteristics must be considered for realistic scenarios. Focusing on the main points, hierarchical routing algorithms can be categorized in itself. Figure 2.2 shows the classification of the routing algorithms, emphasizing the hierarchical ones.

2.2.1 Position-based Routing Algorithms

Position-based algorithms mainly consider positions of nodes to find reliable routes. As long as the fixed or relative position of each node is detected, both intra- and inter-cluster routing can be maintained as stable and reliable.

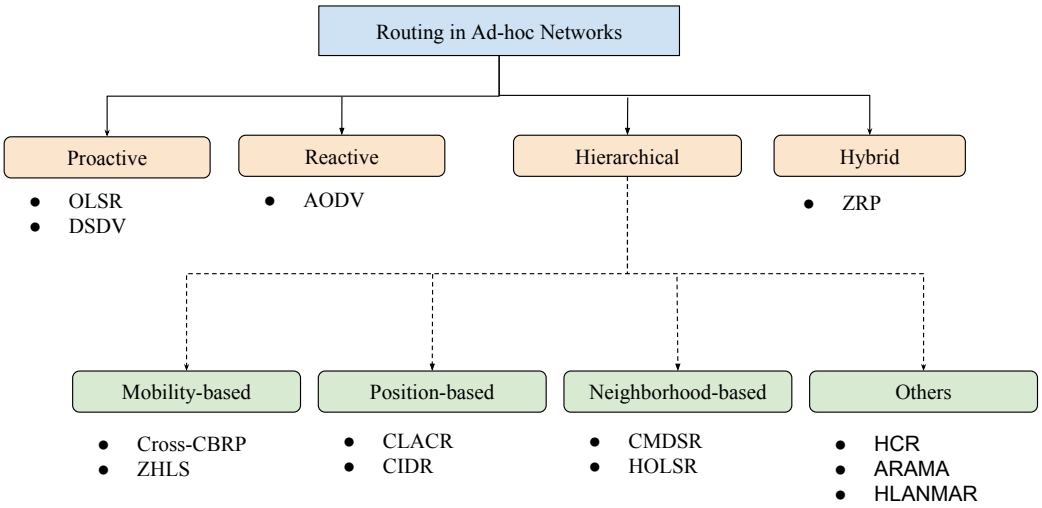


Figure 2.2: The classification of the routing algorithms

In Core-location Aided Cluster-based Routing (CLACR), instead of extra effort for clustering, the whole network is divided into rectangular areas to maximize per-cluster node density and a central node in each group is selected as the CH. The division of areas is conducted by a position manager, i.e., a central server collecting information from each node. New or non-stationary nodes are attended their clusters by this manager. Nodes use Dijkstra's the shortest path algorithm on the topology map obtained from the position manager to find end-to-end routes [42]. This method eases the selection of CH and also route discovery using a central controller mechanism. However, it is not a case in most of the scenarios in ad-hoc networks to have a position manager. Therefore, CLACR has its own cost in terms of the infrastructure.

Rather than position detection of the individual nodes, the nodes moving together are grouped in Cluster-based Inter-domain Routing (CIDR). That is, the nodes which stay close to each other form a cluster considering their relative mobility patterns. Similar to Border Gateway Protocol (BGP) [43] used in the Internet architecture, CHs spread the topology information of their neighborhood, i.e., neighbor domains in BGP, for the discovery of routes in CIDR [44]. In this manner, CIDR is an easily scalable routing algorithm.

Even though position detection by a central server significantly helps to both clustering and routing, it tightly depends on nodes' capabilities, network homogeneity and

environmental conditions. Besides, the effectiveness of position-based clustering and routing is totally up to reliability of the positioning information. Especially in the tough environments with a number of obstacles, understanding node positions might be misleading and it negatively affects the reliability of the routing information [45].

2.2.2 Mobility-based Routing Algorithms

Mobility-based routing algorithms focus on the stability of routing in non-stationary networks. They also aim to establish long-lasting and locally-fixed routes to minimize control overhead even in disorganized mobile networks.

When cluster-based routing algorithms do not consider mobility for CH selection, constantly moving nodes might be selected for the role. It generally leads to the high number of role changes and information lost, since CHs have relatively more information about routing and clustering than ordinary nodes. From this perspective, Cross-layer Cluster-based Routing Protocol (Cross-CBRP) considers different parameters such as mobility and change in signal quality, enhancing with cross-layer optimization. It uses those information to determine the most convenient CH that offers maximum stability in CBRP [46]. Apart from CH selection, consideration of mobility of the intermediary nodes in Cross-CBRP has also increased the reliability of routes which are established by CBRP.

In the networks with high-mobility, cluster expansion, merge and shrinkage, CH selection should be continuously managed to maintain the hierarchical structure. In Zone-based Hierarchical Link State Routing Protocol (ZHLS), the environment is split into non-coincident geographical areas and each node becomes aware of its area using Global Positioning System (GPS). Those areas also represent the clusters, i.e., each area is a different cluster. For intra-cluster communication nodes use their routing tables. In contrast, they need to forward data packets via gateways nodes for inter-cluster communication [47]. Eventually, the maintenance cost due to mobility is minimized using pre-defined areas and GPS information. However, GPS is a quite specific capability for many types of nodes and it exists only a limited number of

scenarios. Even if it is an option to manage clustered mobile networks, ZHLS is far from being a general solution for ad-hoc networks.

2.2.3 Neigborhood-based Routing Algorithms

In neighborhood-based routing algorithms, intra- and inter-cluster neighborhood are considered to find alternative routes. They minimize the possibility of route errors that can frequently occur due to leaving nodes, mobility etc.

Cluster-Based Multipath Dynamic Source Routing (CMDSR) uses the multi-level hierarchical relationship between nodes in addition to their natural positions. By constructing a multi-level clustering (i.e., considering multi-hop distance or multi-role nodes), routing and data forwarding can be separated into those levels for an abstraction. While the first level includes CH and ordinary node communication, the second level is formed between only CHs and routing servers that are defined through the network to effectively share routing information. In this structure, CHs and ordinary nodes, and routing servers and clusters heads are directly connected to each other in a single hop. In contrast, ordinary nodes and routing servers are 2-hop neighbors [48]. This multi-level neighborhood relationship increases topology discovery rates and possible routes that can be found for any end-to-end communication. The existence of the routing servers also leads easy and in-detail discovery of routes. However, having that kind of servers is rarely possible for ad-hoc networks.

Hybrid-OLSR (HOLSR) enhances OLSR by using hierarchy between nodes. It decreases control overhead in scalable networks with the fish-eye technique, which sends routing control messages to further nodes less frequently in comparison to closer ones. This technique also increases bandwidth utilization by decreasing control messages. The hierarchy definition of HOLSR is not related to the roles as in clustered structure, it is represented by the number of hops between nodes instead [49]. HOLSR has similar drawbacks with proactive routing algorithms. However, the hierarchical approach and the fish-eye technique significantly decrease the overhead problem. Figure 2.3 simply shows the hop-hierarchy in HOLSR.

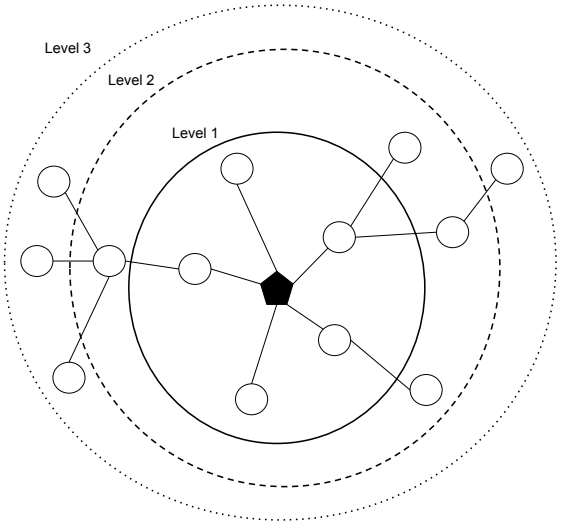


Figure 2.3: Hop-hierarchy in HOLSR

2.2.4 Other Routing Algorithms

Similar to many other clustering algorithms, Hybrid Cluster Routing Protocol (HCR) elects the node with the lowest ID as CH. The main concern in HCR is the different requirements of intra- and inter-cluster communication. In a relatively small area and with a fewer number of nodes, intra-cluster communication is convenient for proactive routing i.e., continuous information sharing between nodes. In contrast, since there is a higher number of stationary and mobile nodes in the whole network, the proactive approach would be quite costly. To avoid this cost, reactive (or on-demand) routing is preferred for inter-cluster communication. Eventually, while it offers a low-delay intra-cluster communication, HCR aims to decrease control overhead and increase bandwidth utilization for inter-cluster communication [50]. However, the clustering approach that HCR takes is not efficient for many other scenarios that are subject to high mobility, limited resource and node-batteries. Therefore, the gains of the hybrid routing are still affected by the underlying clustering algorithm.

Nature has its own solutions for communication between living things. For example, ants left their smell behind so that they can detect the most popular paths considering flavor intensity. The nature-inspired routing algorithm, Ant Routing Algorithm for

Mobile Ad-hoc Networks (ARAMA) uses the battery status and the queue delay of the intermediary nodes as analogous to the ant-smell on a path. The paths containing higher-battery and lower-queue delay nodes become more popular and are tend to be selected. When ZHLS and ARAMA are holistically designed, an increase in packet delivery ratio and a decrease in end-to-end delay observed as presented by the authors of [51].

Hierarchical Landmark Ad-hoc Routing (HLANMAR) is an enhanced version of Landmark Ad-hoc Routing (LANMAR). HLANMAR aims to take advantage of some special roles in heterogeneous networks: the nodes with more transmission power form a control structure so that the communication can be performed in a fewer number of hops. Other than the landmark nodes that are specifically selected in LANMAR, some nodes are also selected for communication backbone in HLANMAR. Routing information of the backbone nodes is managed by the landmark nodes. From this perspective, while landmark nodes are responsible for the routing control packets, the backbone nodes take role in data transfer in long ranges. HLANMAR also proposes a method to detect the optimum size for the backbone nodes [52]. Since it requires special types of nodes with extra transmission power, HLANMAR can be used in limited scenarios with such nodes and energy consumption concern.

Table 2.2: The comparison of hierarchical routing algorithms

Algorithm	**Category**	**Stability**	**Overhead**	**Mobility**	**Convergence**	**Resource**
CBRP [41]	Other	High	Low	Medium	Fast	Medium
CLACR [42]	Position	High	High	High	Very Fast	High
CIDR [44]	Position	Medium	Medium	Very High	Medium	High
Cross-CBRP [46]	Mobility	High	Low	High	Fast	Low
ZHLS [47]	Mobility	High	Medium	Very High	Fast	High
CMDSR [48]	Neighborhood	Very High	High	High	Medium	High
HOLSR [49]	Neighborhood	N/A	Low	High	Fast	Low
HCR [50]	Other	High	Medium	Medium	Fast	High
ARAMA [51]	Other	High	Low	High	Medium	Low
HLANMAR [52]	Other	Very High	High	Medium	Medium	High

Apart from those examples, there are some other routing algorithms that focus on en-

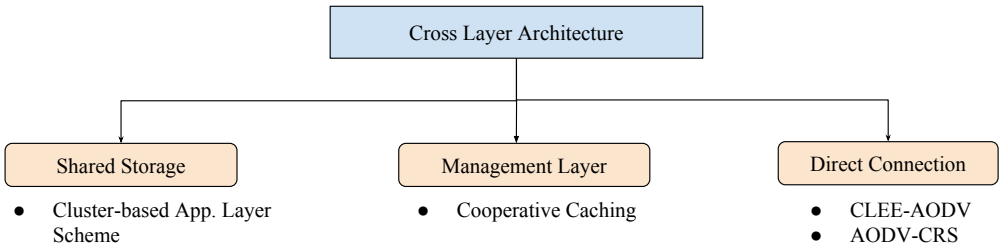


Figure 2.4: The classification of the cross-layer architectures

ergy consumption and a variety of aspects with a weighted average [53]. In Table 2.2, a comparison of the studies presented here is shown in terms of stability, overhead, mobility-tolerance, convergence and resource utilization of them. Stability is directly related to cost and resource utilization since it represents how stable and long-living routes are established by the related algorithm. Mobility-tolerance and convergence are related to each other as well, they show the quickness of the protocols for a low-delay communication.

A significant point that worths underlying is that the existence of CHs and the hierarchical architecture have a great importance in the routing protocol design. Therefore, routing and clustering need to be considered holistically for a complete system. Requirements and characteristics of the networks and scenarios are decisive and reshape the relationship between those two important concepts, routing and clustering.

2.3 Cross-layer Optimization

Cross-layer optimization is considered for a number of problems such as effective routing and clustering, energy conservation and caching. The requirements of those problems are decisive for the cross-layer communication architecture. For instance, the frequency, quantity and direction of information sharing between layers affect the overall design [54]. Figure 2.4 shows the types of different architectures briefly. In this section, the different categories of the cross-layer architecture are discussed.

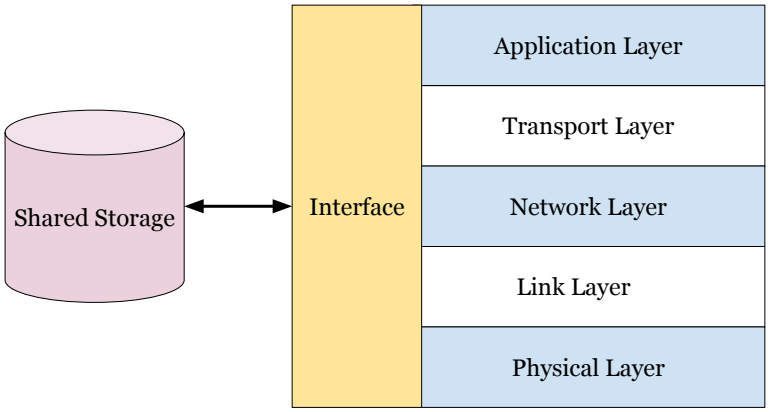


Figure 2.5: Shared storage cross layer architecture

2.3.1 Shared Storage

Shared storage in cross-layer communication is accessed by selected layers to both extract and update commonly-used information. It could be considered as a micro-level database with layer interfaces. Especially when all layers need to share information, defining a single shared storage for common usage is an effective solution. This architecture is shown in Figure 2.5.

Glitho *et al.* propose a clustered topology with cross-layer architecture for multimedia applications named *conferences* [55]. In the algorithm, battery status and measurable distance between nodes are considered for clustering and routing. Taking advantage of the cross-layer design, specific parameters from the network, link, and application layers are sent to the shared storage. Then, each layer continuously exchanges information with the shared storage to update related parameters of itself (or other layers) using predefined interfaces. While the network and link layer use a common interface, application layer has its own interface. In this scenario, the shared storage and the interfaces form the complete architecture of the cross-layer design.

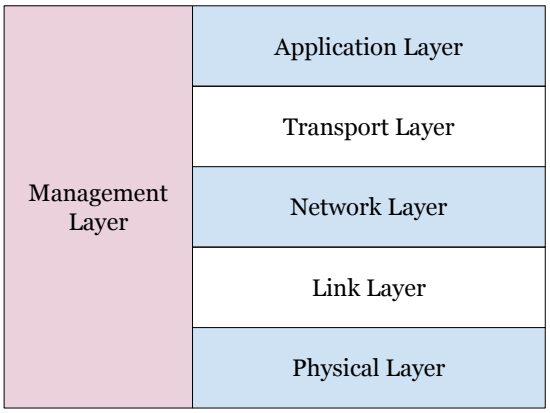


Figure 2.6: Vertical management layer

2.3.2 Management Layer

The management layer is (vertically) placed as a proxy layer between multiple layers and manages different layer-specific parameters interacting with other layers. It is able to make cross-layer asynchronous requests to fetch or update parameters. In some scenarios, it is defined as an integrated shared storage and interfaces block. However, it can actually work as an active and independent layer (or mechanism) that orchestrates a variety of information and takes action when it is required. In Figure 2.6, the management layer vertically covers all other layers as a part of the cross-layer architecture.

Especially in cases which require information processing instead of only sharing raw parameters, the cross-layer architecture with the management layer is priorly preferred. The study conducted by Denko *et al.* is an illustrative example of such scenario [56]. Caching while forwarding enables the clients to fetch data from the closest server (or other edge-point caching hosts) rather than the server that actual data reside. In this study, CHs stores cache-indexing of their member nodes to be aware of the accessible data in related clusters. However, since each member node needs to be updated cache information constantly, there is a heavy control traffic load in uplink, i.e., from nodes to CHs. The information flow through the CHs is performed considering traffic density and clustering information defined in the network layer, current battery status and cache-indexing of a node by the management layer. In this

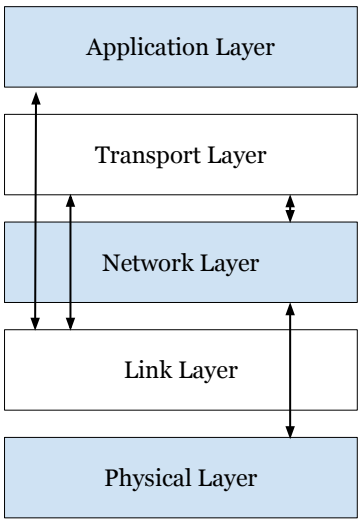


Figure 2.7: Directly connected layers

structure, the layers are not interconnected but the overall system is supervised by the management layer in a cross-layer fashion.

2.3.3 Direct Connection

When a few layers need to work in cooperation in an architecture, connecting them directly could be the least-cost option. Direct connection in a cross-layer schema means that related layers are connected without any intermediary mechanism such as shared storage and the management layer architectures. This architecture can rely on API calls through well-defined interfaces, layer-to-layer messaging with new packet definitions or integration of multiple layers into the single one. Apart from the integration of layers, the other two techniques are performed without consideration of the levels of the cooperated layers.

In the standard AODV routing, a node knows only the next-hop for an end-to-end communication. Therefore, it is enough for a node to be aware of only direct-neighbors to initiate communication via AODV routing. Similarly, in Cross-Layer Energy Efficient Ad-hoc On-demand Distance Vector (CLEE-AODV), the "best" route is not defined as the shortest one: it is the one that costs minimum energy-consumption

instead [57]. Thanks to the cooperation of the link layer and physical layer, each node can identify the minimum transmission power to be able to communicate with its neighbors. Eventually, routes are formed with the least-signal power nodes priorly and overall energy consumption is kept at the minimum. In this architecture, signal power from the physical layer, one-hop neighborhood from the link layer and routing information from the network layer are continuously interchanged.

Since different types of services have a variety of requirements to provide high performance, same routes may not satisfy for each service as expected. In Ad-hoc On-demand Distance Vector Routing Cross-Layer Scheme (AODV-CRS), the authors offer a cross-layer design that covers the network layer and the link layer to obtain service-specific high-performance routes [58]. In the algorithm, the network layer plays an active role in the link scheduling, and the neighbor nodes are informed about the link scheduling scheme continuously. Each node in a neighborhood finds the most convenient route for each service utilizing the channel capacity. For this utilization, the scheduling schemes of the other neighbors are jointly used. The architecture includes a symmetric connection of link and network layers for cross-layer optimization.

CHAPTER 3

DESIGN ARTIFACTS

Before presenting the architectural details of the design, it is better to explain the main design artifacts. The design artifacts are the fundamental terms and implementation details of the overall architecture. In this chapter, first, the two main terms commonly used in DCA and CHRA are presented in Section 3.1. The first one is the *backbone* and it represents the control plane of the architecture, which is constructed by cluster heads and gateways. The other one is the *cluster sight area* and represents the neighborhoods with a certain size measured by a number of clusters. Then, the implementation details of the network elements (i.e., nodes in ad-hoc networks) that embody DCA and CHRA are shown in Section 3.2. Basically, non-parametric details of the simulation design are given in this section.

3.1 Terminology

The backbone and cluster sight area are the core definitions in DCA and CHRA. They construct the control plane of the network and take essential roles in both clustering and routing. The backbone is conceptually similar to structures in other cluster-based routing protocols that use CHs and gateways to maintain both control and data traffic. In this section, these terms are defined for the comprehensibility of the algorithm.

3.1.1 Backbone

Instead of flooding through the whole network, the routing control packets are being forwarded via a specific set of nodes, which are CHs and gateways as shown in Figure 3.1. Except isolated clusters (i.e., when no node in a cluster has a neighbor node from a different cluster), all CHs are connected to the others through gateways. Therefore, CHs and gateways form a complete structure, which is called the backbone, in a fully connected network -in terms of clusters-. Since CHs have topology information of their own clusters, they can easily make routing decisions for an end-to-end communication that is destined to any cluster-member node. Therefore, the maintenance and discovery of the routes are narrowed down to the backbone, which also forms the control plane. From this perspective, control plane and backbone terms can be used interchangeably for both routing and clustering processes.

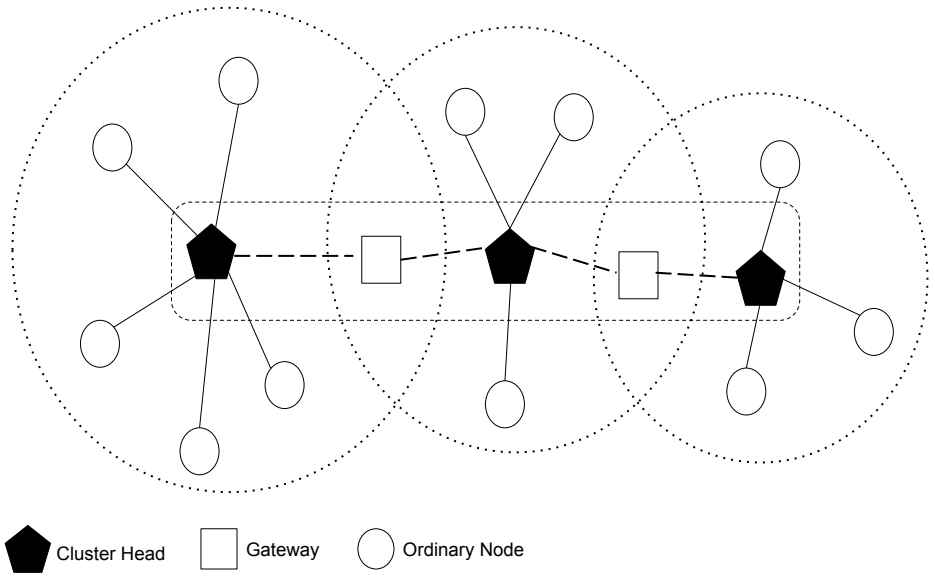


Figure 3.1: The backbone is constructed by cluster heads and gateways. Diamonds, squares and circles represent CHs, gateways and ordinary nodes, respectively.

3.1.2 Cluster Sight Area (CSA)

Apart from the neighborhood of nodes in a flat topology, the neighborhood of cluster heads can be considered in a hierarchical network. Each CH discovers its sight area

considering the clusters in -at most- ***n*-CH-hop** neighborhood as shown in Figure 3.2. A CH-hop represents the distance between two clusters in terms of cluster heads. That is, if two cluster heads can communicate via a single gateway, their clusters are direct neighbors in a 1-CH-hop distance. For instance, Figure 3.2 represents a 2-CH-hop neighborhood where the radius of CSA is 2-CH-hop. In this limited area, the whole topology is known by all CHs, i.e., any link between nodes is identified by each cluster head. Naturally, each cluster head discovers and maintains its own cluster's topology regularly with periodic clustering control packets. Similarly, each CH sends this local topology information with inter-cluster sight area messages (SAM) to its neighbor CHs via gateways. To reduce control overhead for maintenance of CSA, inter-cluster control packets are sent in different periods with a fish-eye approach. That is, while the control packets are sent in every T_{SAM} seconds to 1-CH-hop neighbors, they are sent to n-CH-hop neighbors in nT_{SAM} seconds. Eventually, each CH has more fresh and reliable topology information about closer clusters. In this manner, CSAs are maintained proactively by CHs, as a natural extension of clusters. T_{SAM} is chosen as $3\,\mathrm{s}$ for simulations.

The representation of CSA is a simple adjacency matrix named **visibility matrix**. It does not have to contain complete CSA: a CH stores all nodes that it is aware of in such matrix. Therefore, a visibility matrix is constructed via both clustering control packets and SAMs over time. Since each node is known with its ID, rows and columns of the visibility matrix contains those IDs. If two nodes are neighbor, related cell (i.e., row with ID of first node and column with ID of the second node, and vice-versa) contains 1, else it contains 0. Note that, it basically shows the existence of a link. Alternatively, each cell may contain different information that represents a link. For instance, link quality indicator for each pair of nodes can be used to analyze overall communication quality through a route.

The radius of CSA in terms of CH-hops, *n*, is selected as 2 for simulations. Note that, 2-CH-hop distance is not mandatory but a design issue. The most primal fish-eye approach for inter-cluster information exchange contains (at least) 2-CH-hop distance so that a pivot cluster head is able to discover neighbor topologies with a relative

freshness that is proportional to distance in CH-hops. The upper bound for such structure is the whole network, i.e., sending topology information to all other clusters. In contrast, 2-CH-hop constructs the minimal structure and eventually minimum control overhead for topology discovery. The implicit relationship between CH-hop and regular node neighborhood is also simple. Assuming that cluster heads are connected to each other via gateways (not directly connected), *n*-CH-hop contains $(4n + 2)$-hop paths at most.

The main reason for the construction of CSA is creating a sense of a smaller network that is relatively easy to maintain. Since proactive maintenance of the network-wide routes is costly, a full-discovery only in a smaller area decreases the delay in end-to-end communication and utilizes the control overhead for routing. Therefore, CSA is an effective yet easy-to-maintain structure depending on its size.

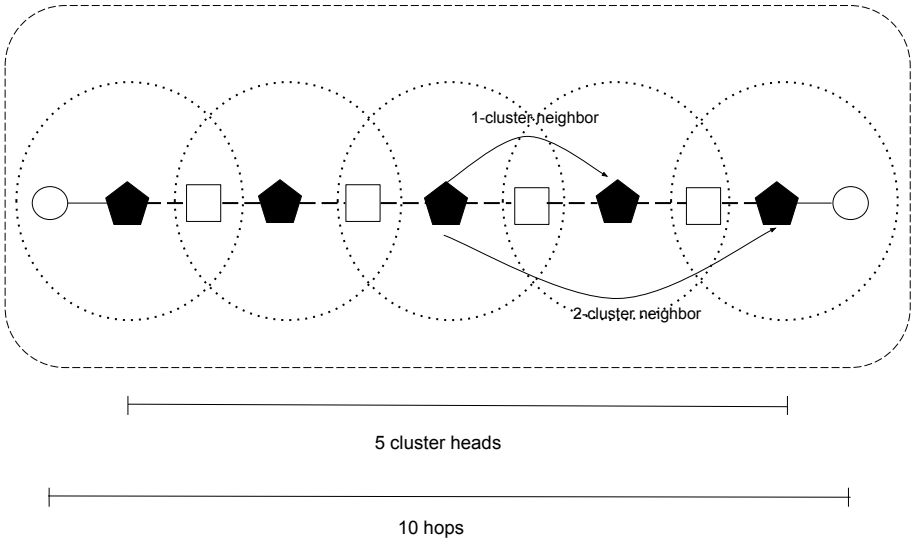


Figure 3.2: Cluster sight area covers maximum 10-hop and cover 2-cluster-hop.

3.2 Fundamentals of the Implementation

The overall architecture, which contains DCA and CHRA, is implemented in the discrete event-based simulator OMNeT++ for the performance evaluation. In the

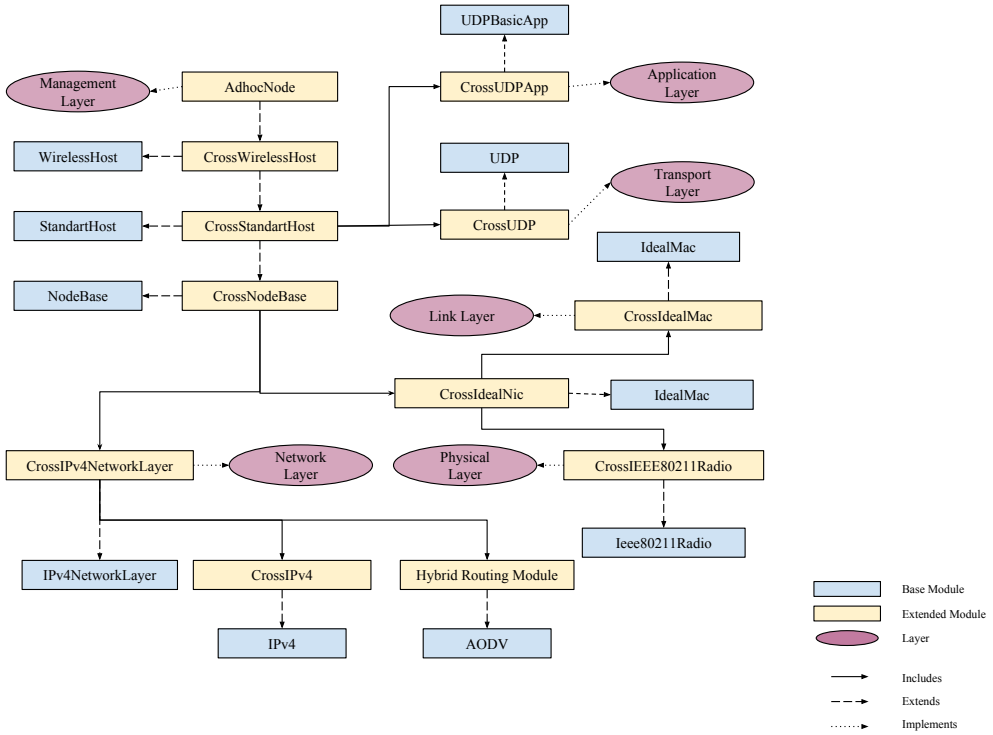


Figure 3.3: The abstract inheritance scheme of the simulation design in OMNeT++.

implementation, each node (i.e., network entities in a homogeneous ad-hoc network) is capable to run DCA and CHRA. The architecture of the nodes is constructed upon the regular TCP/IP stack. All layers in the stack are designed to be controlled by a cross-layer manager named management layer so that clustering is performed in a detailed, flexible and collaborative fashion. The management layer takes the major role in clustering. That is, it gathers all parameters from other layers (e.g., received information from neighbor nodes through the application layer) and analyzes them as presented in Chapter 4 and Chapter 5. The management layer also works as a gateway between layers. In the architecture, there are not directly connected layers (except the management layer): if two layers need to share parameters between, they must (a) request or (b) directly send related parameter to the management layer. The cross-layer architecture mainly depends on the communication interface between layers and the implementation of the management layer. Apart from them, any layer can be changed with another, extended and modified. The extensions of OMNet++ modules for this study is detailedly shown in Figure 3.3.

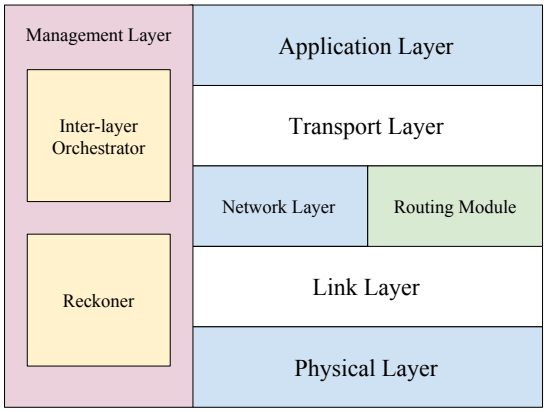


Figure 3.4: The cross-layer architecture implemented in OMNeT++.

Figure 3.4 simply shows this structure. Since the management layer is connected to all layers, it is represented as a vertical module. The interpretation of the different layer-specifics parameters is actually the main definition of the clustering algorithm. The reckoner in Figure 3.4 is the core mechanism that performs clustering and also affects routing. The dynamics of DCA is performed in the reckoner and in this sense, the deployment of different clustering algorithms is very easy in this architecture only by changing the implementation of the reckoner. Therefore, it provides a significant flexibility and modularity while supporting the inherited core architecture.

Lastly, Table 3.1 summarizes the node architecture presenting stack layers and OMNeT++ modules which are used to extend those layers. *CrossPhyLayer* is an extension of *Ieee80211Radio*, and used to calculate SNIR values. It deploys a half-duplex antenna that is able to either send or receive a packet at the same time. Note that, the channel model and physical layer module is different for DCA and CHRA. While the signal quality is considered to understand if nodes are getting closer or further in DCA, the interference and channel errors are omitted in CHRA so that the quality of end-to-end communication can be observed correctly. The physical layer model used in CHRA is extended using *IdealRadio*. For the link layer, *CrossIdealMac*, is designed as an extension of *IdealMac*. In this module, received MAC packets are forwarded to the management layer to count the number of neighbors in 1-hop. *ClusteringApp* is implemented extending *UDPBasicApp* for the application layer to share clustering control packets periodically. Each node broadcasts control packets con-

taining a variety of node and cluster-specific information with a UDP packet. This module also forwards received control packets to the management layer so that it can analyze the information coming from neighbor nodes and realize if there are any clusters in the node's neighborhood. The network layer, *CrossIPv4NetworkLayer*, is implemented as an extension of *IPv4NetworkLayer*. Besides, *CHRA* is the routing module and it is integrated into the network layer. Once the network layer becomes aware of the identifier of the cluster head node, routing control packets for end-to-end communication are started to forward to that node by the network layer. In this sense, *CrossIPv4NetworkLayer* is configured to communicate with related cluster head so that it can orchestrate the communication in a cluster. During network lifetime, the management layer constantly informs *CrossIPv4NetworkLayer* in case of changing cluster. Lastly, *DataApp* is designed to send and receive data packets between randomly-selected nodes. This scheme is given to clarify overall architecture to ease the design of such simulation environment to reimplement this study, and also similar ones required cross-layer structure.

After the presentation of the terminology, it is much easier to understand the fundamentals of DCA and CHRA now. In the next chapter, the formation of CUPS structure is explained through DCA.

Table 3.1: The stack design and corresponding OMNeT++ modules.

Stack	**Module**	**OMNeT++ Class**	**Function**
Layer 5	ClusteringApp	UDPBasicApp	Sending and receiving clustering packets with node information
	DataApp	UDPBasicApp	Sending and receiving random data packets
Layer 4	CrossTransLayer	UDP	Standard UDP protocol connected to managment layer
Layer 3	CrossNetLayer	IPv4	Standard IPv4 protocol hosts routing module
Layer 2	CrossMacLayer	IdealMac	Sending neighborhood information to managment layer
Layer 1	CrossPhyLayer	Ieee80211Radio	Sending SNIR values of related packets. Deploys isotropic a half-duplex antenna.
Routing Module	CHRA	AODVRouting	Implements CHRA
Vertical	ManagementLayer	None	Analyzes all information coming from other layers

CHAPTER 4

DEPENDABILITY-BASED CLUSTERING ALGORITHM

In this chapter, the dynamics and the details of Dependability-based Clustering Algorithm (DCA) are presented. DCA is a novel clustering algorithm that approaches to clustering in ad-hoc networks from a different perspective. It focuses on the dependability of clusters in addition to the reliability of individual nodes as opposed to other proposals in the literature.

This chapter is divided into three sections. Section 4.1 shows the main phases of DCA and also gives a brief overview of the general algorithmic flow. In Section 4.2 and 4.3, a number of metrics used in DCA are given and the dynamics of DCA are discussed, respectively.

4.1 Dynamics of DCA

DCA consists of two sequential phases: *bootstrapping* phase and *maintenance* phase. Bootstrapping phase is an initialization phase to form initial clustered structure quickly. After this phase, DCA-specific parameters (i.e., metrics for node and cluster score evaluation) are started to be considered using clustering control packets that are periodically broadcast. Evaluating the neighborhood information in these packets, (a) cluster head selection and (b) cluster selection are periodically performed in the maintenance phase. Cluster head selection is the process that some nodes are selected as group leaders (i.e., heads of related clusters) with respect to their scores, i.e., node score. On the other hand, cluster selection is another process where each node selects

a cluster to join by comparing the dependability of clusters by dependability score.

Most of the dynamics of DCA are repeated processes such as node and cluster (dependability) score evaluation, cluster head claim, control packet broadcast etc. Therefore, there are a number of periodic tasks in both phases. For the sake of simplicity, those periods are presented in Table 4.1. Each of them is explained in related sections that they are actively used. The analysis of the periods is presented verbally in Table 6.8.

Table 4.1: The periods and time intervals of recurring tasks in DCA.

Parameter	**Symbol**	**Value**
Bootstrap period	t_{boot}	4 sec
Control period	T_{ctrl}	1.5 sec
Claim period	T_{claim}	$5 \times T_{ctrl}$
Tick period	T_{tick}	$3 \times T_{ctrl}$
Cluster period	T_{clstr}	T_{ctrl}
Dependability period	T_{dpnd}	T_{ctrl}

In the rest of this section, the dynamics of DCA are briefly explained. The cluster head selection and cluster selection processes are presented detailedly in Section 4.2 and 4.3.

4.1.1 Bootstrapping Phase

Bootstrapping is the initialization phase of DCA. The main purpose of this phase is creating a clustered network topology where nodes have various roles as soon as the network is initiated. The clustered topology, which is formed at the end of the bootstrapping phase, constitutes a primitive structure where DCA can be performed in the maintenance phase considering a variety of metrics.

The bootstrapping phase is implemented based on lowest-ID-based clustering algorithm proposed by Gerla et al. in [15] with some modifications:

1. There is not an acknowledgment mechanism (or ACK messages) for clustering control packets since the main goal of the phase is creating a primitive clustered topology quickly with a minimum overhead.

2. Each node is initialized as a cluster head. If a node does not get any clustering control packets (due to link or packet failures because of interference or collisions). If a node is isolated, it designates itself as a cluster head by default.

This primitive clustering approach taken in the bootstrapping phase works as follows. Each node broadcasts its unique node identifier (ID $\in \mathbb{N}$) and the node with the lowest ID is selected as a cluster head in each one-hop neighborhood after a quick convergence time. Besides, nodes residing in the coverage area of multiple clusters are designated as gateways and they become a member of the cluster with the lowest-ID cluster head. After the bootstrapping period (which is actually a duration, rather than period) t_{boot}, the maintenance phase begins.

There are two important parameters to be specified in this phase, the bootstrapping period and the control period. The control period, T_{ctrl} is the control packet broadcast period to update cluster formations and the roles of nodes. t_{boot} is the duration of the bootstrapping phase. Those parameters deserve further investigation because they depend on the density of the network and the degrees of nodes, and an optimal value may change by different topologies and scenarios. Since the main objectives of the bootstrapping phase are discovering topology and assigning roles to nodes, this phase can be considered as a "best effort" phase (far from being optimal) that shifts the network from flat- to clustered-topology. The maintenance phase starts at the first control packet transmission cycle, which is a multiple of T_{ctrl}, after t_{boot} seconds.

Basic neighborhood-related information such as centrality, the number of neighbors and neighbor cluster heads is also collected in this phase. In Section 4.2.1, all those effective metrics used in the maintenance phase are explained.

4.1.2 Maintenance Phase

Cluster maintenance is a continuous process that rearranges clustered structure considering a variety of metrics related to nodes and clusters. Therefore, the maintenance phase starts after t_{boot} seconds and proceeds until the network is dismissed. In this phase, nodes start to analyze the information obtained from neighbor nodes and broadcast more comprehensive information at every T_{ctrl} seconds. Eventually, starting from the first control cycle after the bootstrapping phase, nodes elicit some other details about their neighbors. The collected information and the detailed neighborhood-awareness (local topology) are evaluated for two purposes: (1) to select cluster head(s) in a neighborhood and (2) to select a cluster for becoming a member of. Both processes are jointly carried out and form the maintenance phase as presented in the next two sections.

4.2 Cluster Head Selection

Cluster head selection is a repeated process, it could be considered as an attempt to select a suitable cluster head for each cluster. It also triggers cluster splitting and merging. The suitability of a node for being a cluster head is measured by **node score** that is calculated using some metrics presented in Section 4.2.1. The technique for computing node score is explained in Section 4.2.2.

4.2.1 Cluster Head Selection Metrics

Node score is calculated using six different metrics: Energy, Link Quality Improvement Ratio, Self-to-cluster Degree Ratio, Clique-to-degree Ratio, Centrality and Capacity Utilization. Each metric impacts the node score in a certain way that changes the suitability of the node for being a cluster head. In this section, these metrics are explained.

4.2.1.1 Energy

This metric is defined as the ratio of residual energy to nominal energy, which represents the current energy ratio. Each node consumes an amount of energy for transmitting and receiving packets. As the consumption scheme, a state-based power consumption model, which depends on the state of the radio equipment, is employed in this study. In the state-based model, a radio can be in off, sleep, switching, idle, receiving or transmitting states and each state results in consuming various amounts of power depending on the characteristics of the equipment. Such power consumptions per state are defined by the chips in the market. For instance, to get more realistic results, the radio state-based power consumption model is scaled according to well-known chips Microchip RN1810 [59] and SparkLAN WSDB-102GN [60] in this study. Transmitting and receiving states are the active ones that represent signal transmission and reception. Off state represents a deactivated radio. Switch state is also very common when a radio is changing state between receiving and transmitting, or any of those active states to idle state. Idle is a ready-to-go state which consumes relatively low power until a signal reception or transmission is triggered. Generally, while transmission and reception consume relatively higher power; sleep, idle and switch states are less power-consumer as they are basically internal or passive states.

4.2.1.2 Link Quality Improvement Ratio (LQIR)

Nodes calculate the link quality change (LQC) value for each neighbor node. LQC is another important metric that represents link quality variations between two nodes. It is evaluated by comparing the average signal-to-interference-plus-noise ratio (SINR) values of previous clustering control packets and the latest clustering control packet; i.e., it shows how link quality changes over time. Each node stores average link quality for each neighbor and updates it after it receives a clustering control packet. The update process is shown in Algorithm 1. Using this approach, each node may roughly infer whether it is getting closer to or further away from its neighbors.

In Algorithm 1, ϵ represents a threshold value to avoid oscillation in LQC values. It

Algorithm 1 Update Link Quality Change

```
1:  procedure UPDATELQC(node, SINR)
2:      lqc ← 0
3:      n ← neighbors.find(node.ID)              ▷ Find the node in main neighbor list
4:      avgSINR ← n.getSINR()
5:      ϵ ← avgSINR ÷10                          ▷ 10% threshold
6:      if SINR > avgSINR+ϵ then
7:          lqc ← 1
8:      if SINR < avgSINR−ϵ then
9:          lqc ← −1
10:     n.updateSINR(SINR)                       ▷ Update avg. link quality
11:     n.update(lqc)                            ▷ Update link quality change of related node
```

is defined as 10% of the average link quality with related neighbor and can be selected considering channel models, mobility etc. For instance, the frequently changing positions of nodes in high-mobility networks directly affect the density of packet transmissions (and interference) and the distance between nodes in a certain area (and received signal power). Therefore, SINR value of signals between two nodes (i.e., the link quality) is constantly changing even the nodes are moving around a small subarea in long-term. Similarly, the deviation of noise in some particular areas of the network can affect SINR even if nodes are stationary. Therefore, using a threshold value and considering the average SINR value lead to a more reliable comparison. If the latest SINR value is higher than the average of the previous ones more than ϵ, LQC is set as 1: it means links are getting better. If there is no change, i.e., it is between [avgSINR$-\epsilon$, avgSINR$+\epsilon$] the link is stable so LQC is 0. Otherwise, the link quality is decreasing and LQC is set to -1.

LQC evaluation is the significant process to calculate link quality improvement ratio (LQIR). LQIR is the ratio of the number of links that are getting better to the number of all links that a node has. Keeping LQCs updated, it is trivial to calculate link improvement ratio of a node as shown in Algorithm 2.

Note that, LQIR is calculated as a natural extension of clustering maintenance process. That is, there is no link quality indication packets rather than the clustering

Algorithm 2 Link Quality Improvement Ratio

```
1: procedure CALCULATELQIR
2:     conv ← 0
3:     numNegs ← neighbors.size()
4:     for each node n in neighbors do
5:         if n.getLQC() ≥ 0 then        ▷ Check each neighbor if link quality is stable or better
6:             conv = conv + 1
7:     ratio ← conv ÷ numNegs
8:     return ratio
```

control packets. Therefore, LQIR value is updated in every T_{ctrl} seconds.

4.2.1.3 Self-to-Cluster Degree Ratio (SCDR)

SCDR is the ratio of the number of neighbors of a node to the size of CSA that the node resides. Having larger degrees is expected to indicate better adequacy for being cluster heads since connectivity increases with the degree and the number of alternative routes becomes larger.

SCDR is a normalized value in [0,1] because the actual numeric value of the degree is hard to use in a weighted manner for score calculation. The important point in such normalization is that the neighbor nodes have to use a common normalization factor i.e., the size of CSA for a fair comparison.

4.2.1.4 Clique-to-degree Ratio (CDR)

If a network is considered as a graph, it is important to find how strongly a neighborhood is connected since it indicates the possibility of having alternative links and overall connectivity in that neighborhood. Using simple neighbor lists obtained from the neighbor nodes, it is possible to calculate completeness (or density) of connections of a node's neighborhood, which is **cliqueness**. Each non-isolated node involves a clique of at least 2 nodes, and the maximal clique is defined as the largest clique that a node involves. The maximal clique is identified with the ID of the lowest-ID

node in that clique.

The clique-to-degree ratio (CDR) represents the ratio of the maximal clique size in which a node is involved to the degree of the node and is naturally lying between [0,1]. It indicates a stronger link density among its neighborhood and higher CDR represents higher resilience to failures. Algorithm 3 [61] shows the method to find such maximal clique and calculate CDR.

Algorithm 3 Clique-to-degree Ratio [61]

```
1: procedure CALCULATECDR
2:     max_clique ← 1
3:     clique ← {}
4:     for each node n in neigbor do
5:         common ← self.findCommon(n)        ▷ Find common neighbors with node n using its simple neighbor list
6:         for each node c in common do
7:             tmax ← 1
8:             tclique ← {}
9:             for each node d in common do
10:                if d.isNeighbor(c) then        ▷ Check if node c is a neighbor of node d
11:                    tmax ← tmax + 1
12:                    tclique.push(d)
13:            if tmax ≥ max_clique then
14:                max_clique ← tmax        ▷ Find size of max clique gradually
15:                clique ← tclique
16:    clique_id ← clique.findMin()        ▷ Find clique ID by finding the node with lowest-ID in clique
17:    ratio ← max_clique ÷ simple_neighbor.size()
18:    return ratio
```

Note that, Algorithm 3 guarantees that it can obtain all maximal cliques with at least 3 nodes in the network [61]. Even though the message complexity is *O(mn)* (considering the whole network as a graph, *n* is the number of nodes and *m* is the number of edges), since clustering control packets are periodically shared for the maintenance of the clustered structure there is no extra overhead for discovering cliques. In the worst

case, all cliques have the same number of nodes and any adjacent cliques have a common node, *O(n)* rounds are required. For mobile scenarios, the constantly changing topology leads nodes to a continuous cliqueness calculation. However, it is also an integrated process to cluster maintenance throughout the network lifetime.

4.2.1.5 Centrality

Centrality is defined as the ratio of the average degree of all neighbors to the size of CSA and is defined in [0,1]. Since apart from one-hop neighbors of a node, potential connectivity through its neighbors is also quite important for communication with distant nodes. Therefore, centrality represents a different aspect than node degree. Similar to SCDR, it is a normalized value.

4.2.1.6 Capacity Utilization (CU)

A node may be a source, destination or forwarder of a packet traffic. Independent of its role in communication, a higher packet traffic flowing over a node implies that it is located in a more traffic-dense position. This implication also shows how important this node is for end-to-end communication. The capacity utilization (CU) of a node is used to evaluate its effectiveness in communication in terms of the total size of data and control packet it processes. CU is calculated as

$$\sigma(t) = \frac{d_{ctrl} + d_{data}}{\beta t} \tag{4.1}$$

where d_{ctrl} and d_{data} represent the size of all control and data packets processed by a node in bytes respectively, t is the current time and β is the channel capacity or byte rate defined as $\mathrm{B/s}$. Eventually, CU is the ratio of processed data to processing capacity in unit time, in terms of bytes.

CU also shows a location-centrality of a node evaluating its popularity in handling traffic; i.e., nodes that are located at the center of a network tend to be more exposed to traffic. Besides, cluster heads and gateways convey more control traffic. Therefore,

the capacity utilization of those nodes may be higher than ordinary nodes providing them an advantage of re-designation as a cluster head.

4.2.2 Cluster Head Selection Technique

The information that is required to select cluster heads is sent via clustering control packets. Figure 4.1 shows the structure of that type of packet. The fields of the packet are also listed below.

1. Packet Type: This field indicates the packet type. Different packet types are (1) full clustering control packet (FCP), (2) core clustering control packet (CCP), or (3) cluster announcement packet (CAP). FCP and CCP are directly related to cluster head selection, while CAP is used for cluster selection. In CCP packets, only must-fields for cluster head selection are included. In contrast, FCP contains many other fields that help (a) current cluster heads to calculate dependability score and (b) other ordinary nodes to recalculate node score. Packet type occupies $2\,\mathrm{bit}$.

2. Node ID: The ID of the sender node is contained here. In Figure 4.1, this field occupies $T\mathrm{bit}$ where T is $\lceil logN \rceil$ and N is the number of nodes in the network assuming each node has a unique ID.

3. Cluster ID: It is basically node ID of the cluster head and indicates which cluster the related node resides. It occupies $T\mathrm{bit}$.

4. Node Score: It is the score calculated by using all (or some, depending on the metrics that are decided to be used for objective-based design) metrics presented. The calculation of the node score is explained in the rest of this section. Its size directly depends on the precision of score and defined as $10\,\mathrm{bit}$ by default.

5. LQIR**: It is the parameter defined in Section 4.2.1.2. Its size directly depends on the precision of the value and defined as $10\,\mathrm{bit}$ by default.

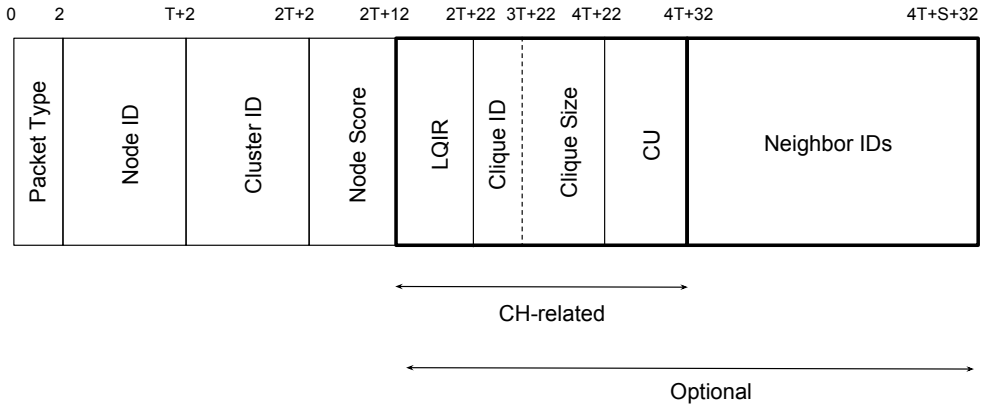


Figure 4.1: The structure of clustering control packet

6. Clique ID**: This field indicates ID of the maximal clique that the node resides. The details for the need of such field are explained in Section 4.2.1.4. Its size is $T\mathrm{bit}$.

7. Clique Size**: The size of the maximal clique is required for cluster heads so that they can calculate cluster score. This value is bounded by *N*, therefore it occupies $T\mathrm{bit}$.

8. CU**: It is the parameter defined in Section 4.2.1.6. Its size directly depends on the precision of the value, and defined as $10\,\mathrm{bit}$ by default.

9. Neighbor IDs*: The IDs of neighbor nodes are also sent with FCP packets. They are required to calculate CDR as explained in Section 4.2.1.4. Its length is indicated with *S* which is NT or $NlogN$.

The fields indicated with (*) are required by other nodes to calculate the node score properly. Others with (**) are required by cluster heads to be able to calculate the dependability score. Both types of fields are optional and do not have to be sent in every clustering control packet. For instance, while nodes send CCP (i.e., not containing optional fields) in every $T_{ctrl}\mathrm{s}$ to maintain current clustered structure, they can send FCP (i.e containing all packets) to trigger score recalculations in every $nT_{ctrl}\mathrm{s}$ where $n > 1$. As seen in Figure 4.1, while the length of CCP packets is $2T + 12\mathrm{bit}$ which is bounded by *O(logN)*, FCP is larger than CCP and takes *O(NlogN)* in bit. Therefore,

switching between those types of packets can increase bandwidth utilization while it negatively affects the freshness and precision of score calculations.

Evaluating all node-specific metrics presented in Section 4.2.1, node n calculates its score δ_n as

$$\begin{aligned} \delta_n &= \sum_{i=1}^{6} \alpha_i n_i, \\ \sum_{i=1}^{6} \alpha_i &= 1, \\ \alpha_i &\geq 0, \end{aligned} \tag{4.2}$$

where n_i represents a cluster head selection metric shown in Section 4.2.1 and α_i is the related weight for each metric. Since all metrics are normalized, δ_n is also defined in [0,1].

The node score, δ_n, basically represents the eligibility of a node to be a cluster head. That is, each node periodically compares its score with its neighbors and the associated cluster head. For any node, there are three conditions to become entitled to claim itself as a cluster head:

1. δ_n should be greater than all its neighbors δ_{n_i} by ϵ where ϵ is a design parameter and defined as 10% of the current cluster head's score to avoid oscillations in cluster head selection. The race conditions are resolved using lower (unique) node identifiers.

2. Node's current residual energy should be at least γ so that it has less risk of a drained battery. Note that, the minimum ratio of energy depends on the devices' capability. However, since it is assumed that the network is homogeneous, it is fixed to 30% for all nodes in the simulations. Alternatively, γ can be defined considering the energy consumption rates of individual nodes. For instance, higher γ values are more convenient for fast-draining nodes, which are more busy or popular in terms of data forwarding.

3. Node's priority (i.e., giving some nodes extra credit to be selected as cluster

head) can be added as an additional restriction. If the designer of the system prefers, priority can be quantized, a weight can be assigned to it and it can be considered in the overall node score.

Note that, each node metric is defined as a ratio normalized in [0,1] and δ_n is the weighted sum of those metrics. The weights are adjustable according to desired importance associated with a metric. δ_n is calculated in each T_{claim} seconds that represents the claim period. The claim period is a preventive factor for oscillations in cluster head selection: if nodes frequently give up being a cluster head and then reclaim again, cluster stability would be significantly violated. However, when the claim period is large, the chances of having more reliable cluster heads may be wasted. Node scores are recomputed at every T_{claim} seconds. It is also decisive for consumed energy for computation.

In case of satisfied conditions, nodes may claim leadership (i.e., being a cluster head) and this situation leads to different scenarios:

1. The former cluster head concedes and the new cluster head continues to orchestrate the cluster. The new one recalculates cluster-specific metrics and dependability score (as explained in the next section) and becomes a member of the control backbone in the network.

2. The former cluster head concedes, however, the new cluster head cannot cover the whole members of the cluster. Then, the nodes that cannot receive information from any cluster head, select a new cluster head among themselves. Eventually, the former cluster is divided into multiple clusters, one with the self-claimed cluster head and the other that is formed by the nodes left without a cluster head.

These scenarios are implicitly handled in the maintenance phase of DCA. For instance, when a node is set free (i.e., no cluster head among one-hop neighbors) as in the second scenario, if there is no neighbor node with a higher node score, it declares itself as a cluster head and a new cluster is constructed. Moreover, isolated nodes are

designed as cluster heads in the network by default as a matter of consistency in the overall network.

Even if δ_n is periodically updated, the liveness of neighbors is also an important issue. To ensure liveness, each node controls the last time it received a packet from its neighbors. A node recalculates the topology-related metrics, which are *SCDR*, *CDR* and *Centrality*, if a neighbor node is absent. Besides, the absence of a CH sets nodes free and triggers the re-selection of CHs; i.e., nodes satisfying claim conditions can declare themselves as CHs. To consider control packet failures, it is important to respite for re-transmission of the missing control packets and the tick period T_{tick} creates a notice time to handle such failures.

4.3 Cluster Selection

When a node joins the network, it also needs to join a cluster (if it is not isolated) to employ common resources. Similarly, mobility and handovers can force nodes to join other clusters. In case of receiving control packets from multiple cluster heads, the node should be able to decide on which cluster it participates in. To be able to compare clusters and to select one, clusters have to declare their dependability scores. The **dependability score** basically represents the dependability of a cluster. The exact definition of the dependability is going to be more clear in the sequel. Fundamentally, the cluster with a dependability score is priorly selected by a node in case of the existence of alternative clusters to join.

The dependability score is calculated using the information collected from the member nodes of a cluster. The context of the collected information is as explained in Section 4.2.2. However, the semantics of those metrics are quite different for computing the dependability score. Instead of using them directly, each cluster head calculates a bunch of cluster-specific metrics that lead to the dependability score calculation.

4.3.1 Cluster Selection Metrics

The cluster-specific metrics, which are Cliqueness, Contraction, Traffic Density and Cluster Degree, are defined in this section.

4.3.1.1 Cliqueness

Since each node shares the identifier of maximal clique it involves and the size of that clique, cluster heads can evaluate the number of different cliques and sizes of the cliques in their cluster. Cliqueness, as a metric, represents the ratio of the average size of the cliques in a cluster to the cluster size. It basically shows the density of connections in different sub-groups of a cluster and is normalized with respect to the cluster size.

4.3.1.2 Contraction

Contraction metric indicates whether or not the member nodes of a cluster tend to get further away from each other. Getting further away means worsening link quality instead of geographically moving away. However, it is still a sign of contraction in a cluster in terms of link quality. This metric is simply defined as the general appearance of link states throughout the whole cluster. It is the average of LQIR values of all nodes in the cluster. Eventually, contracting clusters tend to be more stable and keep their current conditions.

4.3.1.3 Traffic Density

Taking the average of the CU values of each node in a cluster, cluster head evaluates mean capacity utilization ratio inside the cluster. This value implies the operability of a cluster for packet forwarding; a higher traffic density of a cluster shows that related cluster is more popular to forward data.

4.3.1.4 Cluster Degree

Cluster degree is directly related to the number of nodes in a cluster. A cluster with a few nodes brings a certain control overhead and needs resource allocation to manage only a few number of nodes. In contrast, such resource allocation and link scheduling become harder and the number of intra-cluster control packets becomes relatively larger in crowded clusters. For instance, assuming Time-division Multiple Access (TDMA), if nodes randomly access to the channel, the number of collisions increases with the increasing number of nodes. Therefore, it is important to keep the cluster size close to an ideal number of nodes for an optimum resource use and communication quality. Therefore, free nodes should prefer joining in sparse clusters instead of dense clusters without overloading a cluster. On the other hand, it is important to join a cluster with the size that ensures a degree of connectivity. Cluster degree is defined as

$$\sigma(n) = \frac{4n(\mathrm{MAX} - n)}{\mathrm{MAX}^2} \tag{4.3}$$

where n is the number of nodes in a cluster and *MAX* is the maximum limit of nodes a cluster can lodge. In this sense, Cluster Degree metric has a positive impact when the number of nodes in a cluster is closer to the optimal value, which is defined as $MAX/2$ and the cluster with optimal number nodes has the highest cluster degree. A node cannot join in a cluster with *MAX* number nodes. Therefore, *MAX* is a control parameter to avoid clusters from overloading. Figure 4.2 also shows the value of the Cluster Degree as a ratio depending on the *MAX*. Note that, *MAX* is supposed to be adjusted considering the total number of nodes in a network. It is set to 10 for the simulation scenarios.

4.3.2 Cluster Selection Technique

The information which is required to select cluster heads is sent via cluster announcement packets. Figure 4.3 shows the structure of that type of packet. Note that, the sender of this packet is always a cluster head to announce the existence of the cluster and the dependability score of the related cluster. The fields in the packet are listed

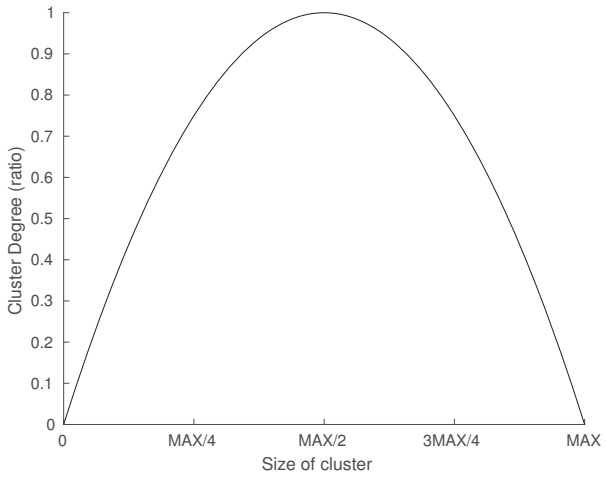


Figure 4.2: The impact of parameter MAX on the cluster degree

below.

1. Packet Type: This field indicates the packet type and marked as CAP. Nodes recognize the field to separate this type of packets from other clustering control packets. Packet type occupies $2\,\mathrm{bit}$.

2. Cluster ID: It is basically node ID of the cluster head (i.e., its own ID) and indicates which cluster the related node resides. It occupies $T\mathrm{bit}$ depending on the number of nodes in the network.

3. Cluster Score: It is the score calculated using the cluster selection metrics. Its size directly depends on the precision of score and defined as $10\,\mathrm{bit}$ by default.

4. CSA Size: It is sent as the normalization degree as explained in Section 4.2.1.3. As any other parameter depending on the network size, its size is $T\mathrm{bit}$.

5. Cluster Size: The role of this field is explained in Section 4.3.1.4. Since the size of a cluster is limited to *MAX*, its length is *log(MAX)*bit which is shown as *M* in Figure 4.3.

Eventually, the size of the cluster announcement packet is bounded by *O(T+S)* or *O(log(NMAX))*.

Similar to the node score, the dependability score δ_c of cluster c is a weighted sum of

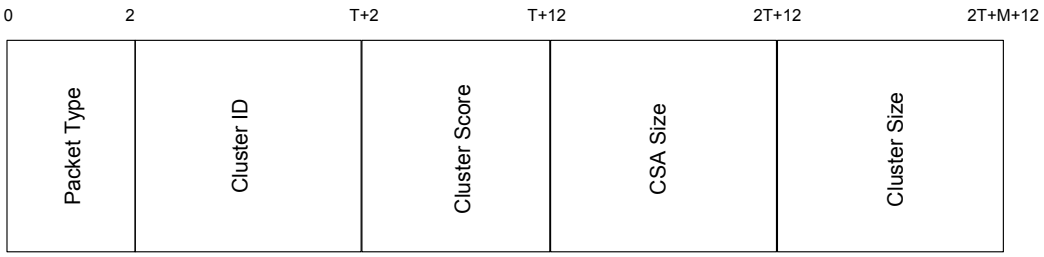


Figure 4.3: The structure of cluster announcement packet

those cluster-specific metrics and reflects the characteristics of them as proportional to metric-weights as

$$\begin{aligned} \delta_c &= \sum_{i=1}^{4} \rho_i c_i, \\ \sum_{i=1}^{4} \rho_i &= 1, \\ \rho_i &\geq 0, \end{aligned} \tag{4.4}$$

where c_i represents a cluster selection metric shown in Section 4.2.1 and ρ_i is the related weight for each metric. Since all metrics are normalized, δ_c is also defined in [0,1].

Similar to CH selection, a node selects the most dependable cluster among many others by comparing their dependability score. A node joins the cluster with the highest dependability score and changes its current cluster if another cluster's dependability score is higher than the current one's by ϵ, which is defined 10% of the current cluster's score. However, the computation and declaration of dependability scores bring some other additional questions that are related to the algorithm design. Dependability score can be announced by employing three different methods:

1. It is announced in every T_{ctrl} seconds piggy-backed to node-specific metrics if the transmitter node is a cluster head.

2. It is announced in every T_{SAM} seconds as an extension to SAM packets to

neighbor cluster heads since cluster-related information has already been shared through those packets.

3. It is announced in every T_{dpnd} seconds which is defined as the dependability period and is independent of other periods and specifically chosen to announce the dependability score. Depending on different topologies and scenarios, dependability period can be defined separately and only dependability scores of the clusters can be announced as a *cluster announcement* message by cluster heads. Besides, when (a) nodes' scores change in a cluster, (b) nodes leave a cluster, (c) nodes join in a cluster, (d) nodes switch off/on their power, the dependability score of the cluster changes. It has to be recalculated before it is shared. It implies that the frequency of announcing the score is also entangled with the frequency of recalculation. Therefore, the dependability period T_{dpnd} needs to be set carefully for energy efficiency. This method is currently being used in the algorithm.

In this chapter, DCA is proposed as the key protocol to form CUPS architecture in ad-hoc networks. In the next chapter, the establishment of a hierarchical end-to-end communication scheme is presented through the routing algorithm, CHRA.

CHAPTER 5

CUPS-BASED HIERARCHICAL ROUTING ALGORITHM

The previous chapter presented the formation process of CUPS architecture. In this chapter, the dynamics and the details of CUPS-based Hierarchical Routing Algorithm (CHRA) are presented as the description of end-to-end communication scheme. In CHRA, two major distance-dependent approaches are taken in the routing process. The first one is **in-area communication** and it represents the communication in short distances (in terms of hops) inside a CSA. In contrast, **long-distance communication** means end-to-end communication outside the CSA where the number of hops between source and destination nodes is relatively higher. Those techniques are explained in Section 5.1 and Section 5.2, respectively.

The hybrid approach in CHRA brings different techniques together taking advantage of the clustered network structure. Table 5.1 summarizes those techniques. Only in the in-area communication, the data plane is used for end-to-end data transfer and the control plane is used to find routes with routing control packets through the backbone. Therefore, an effective use of the plane-separation is observed there. Since the complete topology is discovered in a small area i.e., CSA, the routes including end-to-end paths (EEP) from source to destination can be discovered. Even if CSAs are proactively maintained, routes are still drawn on-demand. Therefore, it is regarded as a semi-proactive technique. In contrast, the long-distance communication is totally up to the backbone for routing and data carriage on-demand. Thus, the type of routing in long-distance communication is reactive. Algorithm 4 briefly shows this hybrid routing process as well.

Table 5.1: Different distance-dependent approaches in CHRA. While in-area communication represents the end-to-end communication inside a CSA, long-distance communication refers the communication between source and destionation nodes that belong to different CSAs.

Attribute	**Type**	
	In-area Communication	**Long-distance Communication**
Plane Separation	Yes	No
Route Type	End-to-end path	Next-hop and distance
Routing Table	EEP table	Distance table
Routing Maintenance	Semi-proactive	Reactive

Algorithm 4 Hybrid route discovery process.

procedure ROUTEDISCOVERY

Source node (SN) sends an RREQ to CH containing destination address

. . . . . .

▷ CH checks;

if Destination node in the visibility matrix **then**

CH sends RREP containing the shortest EEP to RREQ-source

else if Destination node in distance table **then**

CH sends RREP containing next-hop information to RREQ-source

else

CH forwards RREQ to other CHs via gateways

. . . . . .

▷ SN checks;

if No RREP received until timeout **then**

SN initiates route discovery again

else

if RREP contains full EEP **then**

SN updates routing table to keep the shortest EEP

else if RREP contains only next-hop **then**

SN updates distance table

Note that, the in-area and long-distance communication depend on different types of routes to forward data. In-area communication employs EEPs that are defined inside of certain CSAs. On the other hand, only next-hop information is known through forwarding in the long-distance communication similar to the well-known distance vector approach. Therefore, there are different types of routing tables for each approach. In the rest of this section, the distance-dependent approaches are presented considering those differences.

5.1 In-area Communication

It is possible to find an end-to-end path in a CSA, since at least one CH knows the complete topology of that area. The control and the data plane are separately considered for finding a route and forwarding data respectively in this case. That is, an EEP includes a number of intermediary nodes that do not belong to the control plane i.e., not a CH or gateway. Therefore, the discovery and maintenance of such routes require different techniques than the traditional methods that totally depends on the control plane -or the backbone- for both routing and forwarding. In this section, route discovery and maintenance methods for the in-area communication are explained.

Route Discovery: The in-area communication is illustrated without the use of the control plane (i.e., the backbone) for data forwarding in Figure 5.1. In the figure, node (a) and (f) are source and destination nodes, respectively. To find the route going to node (f), node (a) sends a route request (RREQ) to its CH (g) with packet (1) containing connection demand to node (f). Each CH stores its visibility matrix that is proactively formed using the topology information in periodic SAM packets. When the CH receives an RREQ, it first checks its visibility matrix if the destination node (e.g., node (f) in this scenario) is visible, i.e., contained in the matrix. If it were visible, the CH would have run Dijkstra's shortest path algorithm on the visibility matrix and found the shortest path independent from the backbone.

However, node (f) is not in the visibility matrix of node (g) Figure 5.1. Consequently,

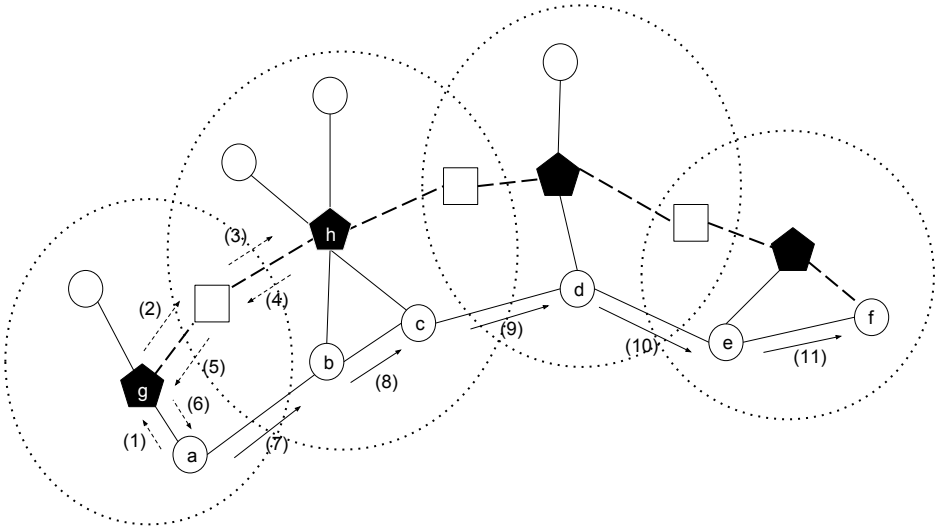


Figure 5.1: In-area communication is performed inside a CSA. CH (*h*) contains both source node (*h*) and destination (*f*) in its visibility matrix and finds an EEP.

RREQ in packet (1) cannot be responded directly. Instead, node (*g*) forwards the RREQ to neighbor CHs via gateways through the backbone. Packets (2)-(3) represent forwarding of an RREQ to the neighbor CH, (*h*). Node (*h*) is aware of the whole topology shown in Figure 5.1 since every node is placed in 2-CH-hop range with respect to node (*h*). Therefore, it can find a complete route from node (*a*) to node (*f*) by running Dijkstra's shortest path algorithm on its visibility matrix. Note that, Dijkstra's shortest path is very well-known and easy to apply on an adjacency matrix in terms of both implementation and time complexity (that is *O(MlogN)* where M is the number of links and N is the total number nodes in CSA). Besides, any other shortest-path algorithm can be considered after constructing CSA as a visibility matrix.

Afterwards, node (*h*) sends back the EEP [*a-b-c-d-e-f*] to node (*g*) with packets (4)-(5). Node (*g*) notifies the source node (*a*) with a routing response (RREP) containing the demanded route and node (*a*) stores this EEP in its **EEP table**. EEP table is a simple routing table that contains the ID of destination node, EEP to destination node and the length of this EEP. It is constructed for only in-area communication. After

route request and response messages have arrived and related EEP is recorded to EEP tables, the process in the control plane finishes. That is, the whole routing process (i.e., finding an end-to-end path on the data plane) is handled in the control plane.

Finally, node (*a*) forwards the data packets tailing EEP to node (*b*), and the forwarding process is continued hop-by-hop through the packets (4)-(7) in the data plane that consists of ordinary nodes. If an intermediary node is not aware of that particular EEP, it caches the path and forwards the packet. Otherwise, it assumes that the EEP is used before and next nodes in this EEP are aware of this path as well, and removes EEP from the data packet before forwarding to decrease the size of the packet. Note that, intermediary nodes can use cached EEPs for only data forwarding. It means that they do not use an indirectly obtained path for initiating an end-to-end communication as a source node. The reason for this restriction is explained in the next part, Route Error.

Route Error: When a broken link exists in the backbone, it is relatively easy to detect since CHs have a periodic message exchange scheme for clustering control packets. However, it is not always possible to detect a broken link between two ordinary nodes. In such cases, any path containing the broken link loses its validity. Other nodes using related invalid routes need to be informed about the broken links using a minimum number of control packets. Therefore, the control and data plane separation requires a route recovery and maintenance mechanism to continuously manage routes in the data plane.

Figure 5.2 shows a routing error scenario that occurs in the data plane. In the scenario, node (*e*) is available anymore due to mobility, or a node crash. Since nodes periodically send keep-alive messages in DCA to maintain the clustered structure as a common nature of clustering algorithms, its neighborhood becomes aware of the loss soon depending on the cluster maintenance scheme.

When node (*a*) sends data using the route that is obtained from its CH with the control packets (1)-(2), data packets are forwarded through node (*d*), and node (*d*) detects that the EEP is actually broken since node (*e*) is off. In this scenario, it deletes any recorded route in which node (*e*) is included and send a route error (RERR) packet to

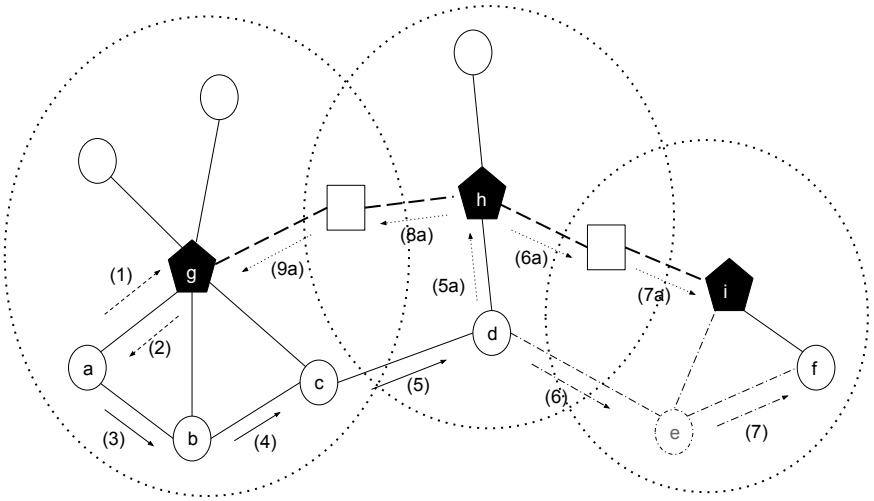


Figure 5.2: Routing error in data plane. Absence of node (*e*) breaks the EEP constructed between node (*a*) and node *f*

.

its cluster head with the control message (5a). The first RERR packet (5a) contains the source of the route (node (*a*)), ID of the lost node (node (*e*)) and a timestamp. When a CH receives an RERR packet, firstly it deletes the lost node from its visibility matrix and all recorded routes containing that node from its routing table(s). Then, construct a list of source nodes (**source notification list**) that requested any of the deleted routes. Note that, the destination nodes of those routes also added to the list since they record reverse routes (i.e., route from destination to source node) as well. Adding the source notification list, it forwards the RERR packet to all neighbor cluster heads. After the first RERR packet, each cluster head applies the same procedure with a difference, they also forward RERRs to the nodes which are in source notification list and update this list if it knows another source node demanding the related broken path before. Eventually, all cluster heads and source/destination nodes are informed about a broken link. Note that, this method is only applicable when intermediary nodes are not allowed to use a cached route to initiate a connection. If they do so,

source nodes for related routes cannot be tracked and RERR messages need to be broadcast frequently, and it creates a significant overhead especially for high-mobility networks.

While a node removes another node (i.e., a lost node) from its visibility matrix, other CHs which are not aware of this loss yet may send a SAM packet containing the lost node. It may lead adding the lost node the visibility matrix again. Therefore, it is necessary to be able to decide the freshness of the information. Each node records the lost node in its **node ban-list** after removing related routes. Each entry in node ban-list contains the ID of a lost node and a timestamp when its loss is detected. In case of a topology update via a SAM packet, a node firstly checks its ban-list if any node in SAM appears in its ban-list. If any exists, it checks the timestamp in related ban-list record to evaluate how long it has been since the node is lost; if more than T_{ban} seconds passed after the loss, then topology update is considered as "fresh" otherwise any topology information related to a lost node is discarded. T_{ban} is directly related to mobility level of a network and in the test scenarios, it is determined as 10 seconds.

When RERR packets are propagated through the network via the backbone, it is possible to get same RERR packets for a node. Because there are multiple paths to access CHs, gateways and source nodes. Assuming there is no isolated cluster, all CHs are connected forming the backbone and it means that all of them would receive a RERR packet at least once. Each CH records the sequence number of RERRs and directly discards duplicates. Besides, since EEPs are defined in maximum $(4n + 2)$-hop (where CSA has a *n*-CH-hop radius), TTL of RERR packets for source nodes is limited to $(4n + 2)$. Eventually, discarding duplicates and the TTL limitation minimize the flooding of RERR packets.

Route Repair: There is also an alternative method to overcome excessive number of RERRs that may be an issue in high-mobility networks, that is route repair. When a node detects a broken link, it is able to repair such link before sending a RERR packet.

In CHRA, there are two types of route repair mechanisms. The first one is **local re-**

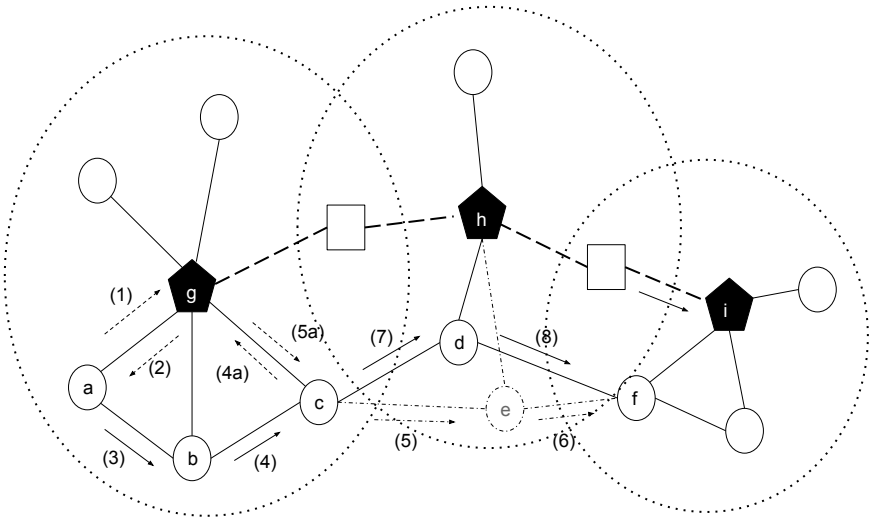


Figure 5.3: Local repair is completed when a single next-hop alternative is found. Node (*c*) detects the absence of node (*e*) and sends a RPREQ to its CH. An alternative node, node (*d*), is found and the EEP is repaired locally.

pair and it aims for minimum control overhead and modification in an existing route for repair. The other one is **global repair** that aims for route reliability with a more controllable approach. Figure 5.3 shows an example of local repair. The main idea behind this type of repair is that, instead of finding an alternative route for the EEP with a broken link or repairing it in an end-to-end fashion, only a single alternative next-hop node is searched to quickly fix the route. In this sense, related route is patched with minimum effort and it is not required to spread RERR packets through the backbone for a lost intermediary node. In Figure 5.3, [*a-b-c-e-f*] is constructed for the communication between node (*a*) and node (*f*). When node (*c*) detects the broken link to node (*e*, it sends a repair request (RPREQ) to its cluster head with (4a)-(5a). Since the CH *g* can observe the whole topology (i.e., in the CSA) presented in the figure, it directly looks for an alternative path going from node (*c*) to node (*f*), instead of from node (*a*) to node (*f*). Note that, looking for an alternative route from node (*c*) to node (*f*) is not for directly finding a partial path to destination, it is finding another next-hop node that completes the original path [*a-b-c-e-f*]. The only update in the route is forwarding through node *d*) instead of node (*e*). Whether node (*a*) is aware of the loss of node (*d*) or not, (*c*) repairs the path without announcing it to the whole network but its cluster head. During repair, the data packets are cached in the node that detects the broken link (node (*c*) in this scenario). Eventually, a minimum number of routing control packets is generated and it leads both higher resource utilization and low delay communication with a quick fix. However, it is not always possible to perform local repair considering an alternative next-hop. The alternative method, global repair, is preferred in such cases.

Figure 5.4 shows the global repair that is performed when a local repair is not possible. Global repair basically tries to find a full sequence of nodes after a broken link. It is different than finding a new EEP since it only completes the path after a broken link. In the figure, when node (*d*) detects the broken link, it sends an RPREQ packet to its cluster head with (5a)-(6a). Since CH (*h*) cannot find a local repair alternative (a single alternative node instead of node (*e*), it draws a totally different path to be replaced with the broken part. For instance, the data packets are forwarded from node (*d*) to node (*f*) through (8)-(10) rather than (6)-(7). Additionally, CH (*h*) sends

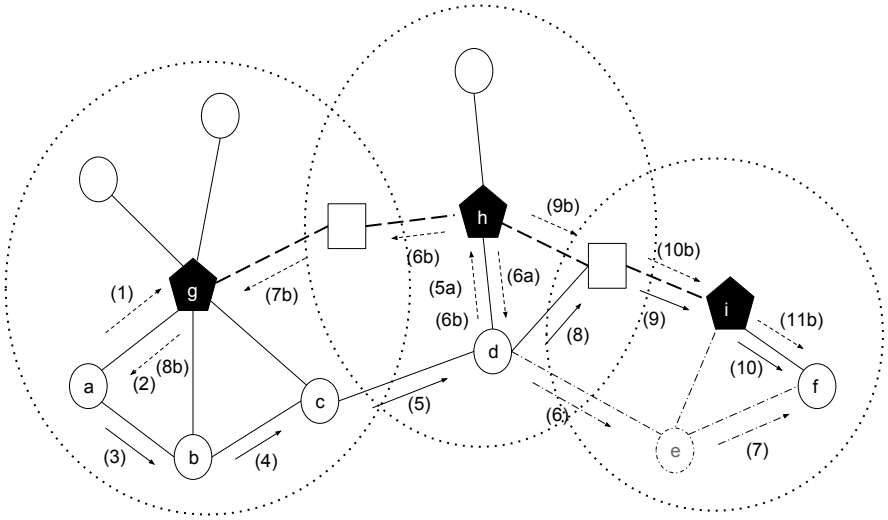


Figure 5.4: Global repair is performed when local repair cannot find an alternative next-hop node instead of the lost node. In the figure, node (*d*) detects a broken link to node (*e*) and asks for repair to its CH. Since local repair fails, a new route is drawn and announced to rest of the network.

RERR packets through the backbone including (a) updated path and (b) the identifier of broken EEP with (6b)-(11b) to announce such update to the source and destination nodes. Note that, while the invalid route is announced to the network in global repair, it is not a case in local repair. Because the maintenance of a one-hop updated path is relatively easier than a multi-hop path. That is, when a global repair is performed, any other repair in formerly-repaired parts leads nodes to maintain multiply-repaired routes without awareness of other nodes including CHs. To avoid side effects in such scenarios and keep the maintenance easier, any changeover in routes is announced to rest of the network via the backbone in global repair.

Lastly, there could be such scenarios where any type of route repair is not possible at all. However, RPREQ packets are sent in any case since repair cannot be performed without CHs that manages the CSAs. Therefore, nodes are waiting for RPREQ response (RPREP) for a limited time, then drops the cached data packets if related

RPREP is not received. RERR packets are triggered by the CHs that receive RPREQ but cannot repair the broken part. Algorithm 5 briefly concludes the procedure of the route recovery.

Algorithm 5 Route repair process for end-to-end paths in the data plane.

procedure ROUTEREPAIR

Node sends an RPREQ to CH containing missing node and invalid EEP identifier

▷ CH checks;

if There is an alternative node to patch the route **then**

CH sends RPREQ to originator node containing patched route

else if A new path exists from originator node to destination node **then**

CH sends RPREQ to originator node containing new path drawn from originator to destination node

CH sends RERR to other CHs and source nodes via backbone

else

CH sends RERR through the backbone containing missing node and invalid path identifier

5.2 Long-distance Communication

For end-to-end communication outside a CSA, there is not a single CH that can find an EEP. Therefore, instead of separating the control and data plane, data packets are forwarded through the backbone. That is, both control and data packets are forwarded via CHs and gateways. In Figure 5.5, the distance between source node (a) and destination node (f) is $p + 1$ where $p \geq 4n + 2$. When node (a) sends an RREQ to its CH, it cannot find an EEP. Therefore, the CH forwards that RREQ to neighbor CHs via gateways. Since no intermediary CH has both source and destination nodes in its visibility matrix, the RREQ is forwarded until it reaches to the cluster where the destination node (f) resides. Packets (1)-(p) represent the routing process through the CH of the destination node's cluster. Afterward, since the CH knows all members of its cluster, it sends an RREP back to the originator of the RREQ. Packets (p+1)-(2p) in Figure 5.5 shows this process. In each packet (p+i), the receiver node records where the packet comes from and its own distance from the destination node to its **distance table**. For instance, the node which receives (p+1). packet becomes aware of that

it can send packets to node (f) through the CH in 2 hops. Similarly, the receiver of RREP (p+i) knows that the destination node is $i + 1$ hops away through the node that sends this RREP. Note that, when an RREP offering a shorter distance to a destination node is received, the distance table is updated with this distance and related next-hop node. In this sense, this approach is quite similar to Ad-hoc On-demand Distance Vector (AODV) routing that is constructed upon the backbone. Eventually, the route, which is going from source to destination through the backbone, is found and node (a) starts to send data via this route forwarding packets to its CH.

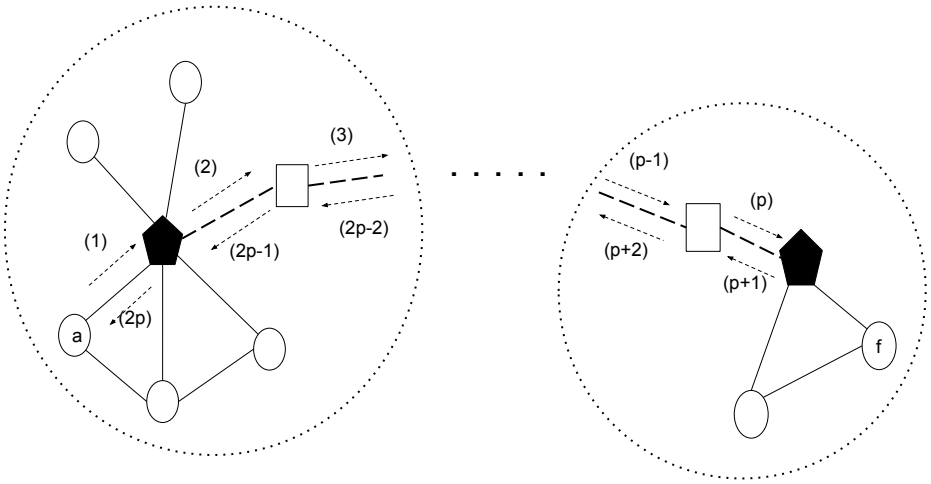


Figure 5.5: Communication in longer distances is constructed on the backbone.

When an RREQ is conducted by any source node, it is not possible to instantly determine if the destination node is in CSA. Besides, RREQ packets are not forwarded through only a single cluster head. That is, even though the source node sends only a single RREQ to its own cluster head, it is then forwarded to all neighbor cluster heads via gateways lying between adjacent clusters. Therefore, it is quite common that multiple routes, which resides inside or outside of a CSA, are obtained. In Figure 5.6, two alternative routes are found using two different methods: packets (2)-(n) leads a backbone-dependent route for long-distance communication. In contrast, packets

(2a)-(3a) provides a plane-separated EEP since the CH (*h*) has both source and destination nodes (*a*) and (*g*) in its visibility matrix. In this case, for both reducing the traffic load on the backbone and selecting the shorter path, the path [*a-b-c-d-e-f-g*] is selected by source node (*a*) to forward the data packets.

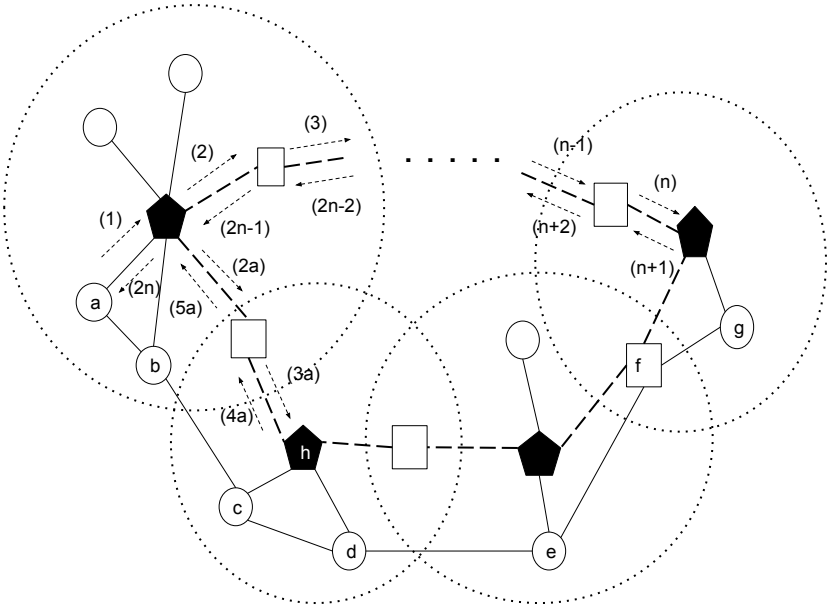


Figure 5.6: End-to-end path in data plane is priorly preferred over the backbone-dependent one.

In the long-distance communication, since the routing process is started on-demand, route errors only appear when a stored route in the distance table is not valid anymore. In this case, similar to AODV, RERR packets are sent by the source node and the routing process is retriggered. When the backbone is directly used for route discovery, the related route would be always discovered unless the cluster that destination node resides is isolated.

After the presentation of CHRA, the formation of CUPS architecture and hierarchical end-to-end communication scheme are completed as a whole design. In the next chapter, the performance evaluation of this design is presented and discussed.

CHAPTER 6

PERFORMANCE EVALUATION AND DISCUSSION

In previous chapters, the details of CUPS-centric network management and end-to-end communication scheme are given through clustering and routing. In this chapter, the performance evaluation and discussion of DCA and CHRA are given. In Section 6.1, the sensitivity analysis for metric-weights is discussed and DCA is compared with the opponent clustering algorithms. Then, the performance evaluation of CHRA in stationary and mobile scenarios with uniformly and nonuniformly distributed networks is given in Section 6.2.

There is a number of measures to evaluate DCA and CHRA. They are mostly related to stability of clusters, energy efficiency and quality of service of the overall design. Those performance measures are listed as,

1. *Average number of role changes per node** shows how frequently clusters restructure since in each role change a cluster head turns into an ordinary node or vice-versa. Therefore, it is a negative indicator for cluster stability.

2. *Average role duration per node** represents how long a node keeps its role and it is strongly-coupled with (1). It is, again, used to measure the stability of the clustered structure.

3. *Average number of cluster changes per node** shows how frequently nodes change their clusters and join a different one. It is not as critical as role change, however, an indicator for cluster stability.

4. *Average duration for staying in the same cluster per node** is similar to (2), and directly proportional to cluster stability.

5. *Number of control packets for cluster convergence** shows the number of effective control packets that triggers an action in clusters, e.g role or cluster changes. This performance measure shows the effective control overhead for clustering.

6. *Packet delivery ratio** represents the success in end-to-end communication and is directly related to the quality of service. A set of random source and destination nodes are selected for each scenario to create a continuous data traffic to measure this one.

7. *Number of control messages for routing*** shows control overhead in terms of the number of routing control packets.

8. *Data-to-All ratio (DAR)*** is defined as the ratio of the total size of successfully delivered data packets to the total size of all packets including CSA overhead for CHRA. To measure the total size of control packets, different aspects and design issues are specifically considered for each algorithm. In CBRP, for the size of control packets flowing through the backbone, AODV packet size is selected as $64\,\mathrm{B}$ which is shown as the optimum size in [62]. The same packet size is also chosen for the standard AODV algorithm. In contrast, even though the size of basic control packets is again 64 bytes in CHRA (i.e., route request, response and repair packets), the cost of topology discovery to form CSA is varying depending on the size of local topology information that SAM packets carry.

9. *Average end-to-end delay**** is an indicator for the quality of service for users or nodes in the scenarios.

10. *Standard deviation in energy consumption**** shows if nodes generally have a fair energy consumption scheme, or just a particular group of nodes is draining. It is important to reveal if only particular nodes such as members of the backbone are exhausted, or energy consumption is fairly distributed.

11. *Average energy consumption per node**** is self-descriptive and shows the energy efficiency. It is jointly used with (10) to show even if nodes consume similar energy on average, it does not mean each of them consumes equally, or fairly. The same power consumption model, state-based energy consumption, is used for the whole architecture.

Note that, the measures indicated with (*) are used to evaluate DCA while (**) represents the measures that are used to evaluate CHRA. Lastly, the measures with (***) are used in common for both DCA and CHRA.

6.1 Performance Evaluation of DCA

In this section, the sensitivity analysis and the performance evaluation are presented for DCA. The sensitivity analysis for the metric-specific weights of DCA is presented in Section 6.1.1. Using the results obtained from the sensitivity analysis, DCA is optimized and compared with other clustering algorithms in stationary and mobile scenarios.

One of the main reasons for designing a weighted clustering algorithm is adaptability: the algorithm can be adapted different scenarios and conditions by changing weights. For performance evaluation, three different goals are defined: stability, energy efficiency and QoS and each of them is measured by the performance measures presented at the beginning of this chapter. The relationship between the performance measures and the goals is shown in Table 6.1. Throughout the sensitivity analysis, it is aimed to find metric-weights that optimize the performance of DCA in terms of such objective-specific measures. Besides, the overall performance evaluation is given in Section 6.1.2 considering the measures that are grouped under different goals.

Table 6.1: The performance measures for different objectives. Those measures are evaluated separately considering changing weights of cluster head selection metrics and cluster selection metrics.

Goal	CH Selection Metric-weights	Cluster Selection Metric-weights
Stability	Num. of role changes	Num. of cluster changes
	Avg. role duration	Avg. duration for staying in same cluster
Energy efficiency	Avg. energy consumption	Avg. energy consumption
	Std. deviation in energy consumption	Std. deviation in energy consumption
QoS	Num. of clustering packets	Packet delivery ratio
		End-to-end delay

6.1.1 Sensitivity Analysis

In this section, the sensitivity analysis for the metric-weights of DCA is presented. The sensitivity analysis (SA) is an analytic approach to determine the effects of different parameters on a dependent result i.e., a multivariate function. There are a number of SA methods in the literature focusing on different aspects of the parameters that are analyzed. For instance, the dependency and uncertainty of the parameters, and the size and the randomness of dataset are fundamental characteristics of the data taken into consideration for the selection of SA method [63]. In this study, Moment-independent Delta Analysis is used as the sensitivity analysis method [12][64]. The main reasons for this selection are,

1. Moment-independent Delta Analysis is able to detect both linear and non-linear relations between parameters and results.

2. It is not a variance-based algorithm. Instead of considering the effect of a single parameter at a time, it takes all sample parameters and related results into consideration to analyze the whole model and cross-relations through all data. This methodology is convenient especially for dependent input parameters i.e., when the sum of all parameters must be 1. Therefore, it is a globally-sensitive analysis algorithm.

3. It is useful for uncertain parameters. The uncertainty in parameters means that

none of the parameters has a restriction, priority or difference from the others as an input. Therefore, all parameters have the same importance for the function initially.

The SA framework consists of four major phases, (a) statistical analysis, (b) numerical analysis, (c) cross-validation and (d) optimization. Figure 6.1 shows all those phases step by step. In phase (a), 5000 random metric-weight sets are generated and used in stationary and mobile scenarios in the simulation environment at step (1) to obtain the performance results in terms of the performance measures presented in Table 6.1. At step (2), Moment-Independent Delta Analysis algorithm is run over those results and it finds a sensitivity indicator δ for each metric-weight. Basically, δ shows the relationship between specific parameters and the performance measures. The most influential metric-weights are identified at step (3) (Section 6.1.1.1). However, the indicator δ does not directly show if a metric-weight has a positive or negative influence on the performance measures. It only indicates that the related metric-weight has a significant impact on the performance measures. Therefore, in phase (b), the simulations are re-run using a specific set of metric-weights at step (4) and the numerical results are collected at step (5) (Section 6.1.1.2). In phase (c), the positive and negative effects of the metric-weights are comprehended by validating statistical and numerical results obtained from phase (a) and phase (b) together (Section 6.1.1.3). Finally, one becomes able to define optimized metric-weights for different objectives in phase (d) using the cross-validation results obtained in phase (c) (Section 6.1.1.4). In the rest of this section, all those phases are discussed presenting their step-by-step outcomes.

6.1.1.1 Statistical Analysis

The statistical analysis is the first phase of the SA framework. Using randomly generated 5000 different metric-weight sets for cluster head selection and cluster selection metrics, the impacts of the metrics on specific goals are found. For example, assume that 6 random weights w_i are assigned to each cluster selection metric and a version of DCA embodying those weights is designed at step (1) of Figure 6.1. Then,

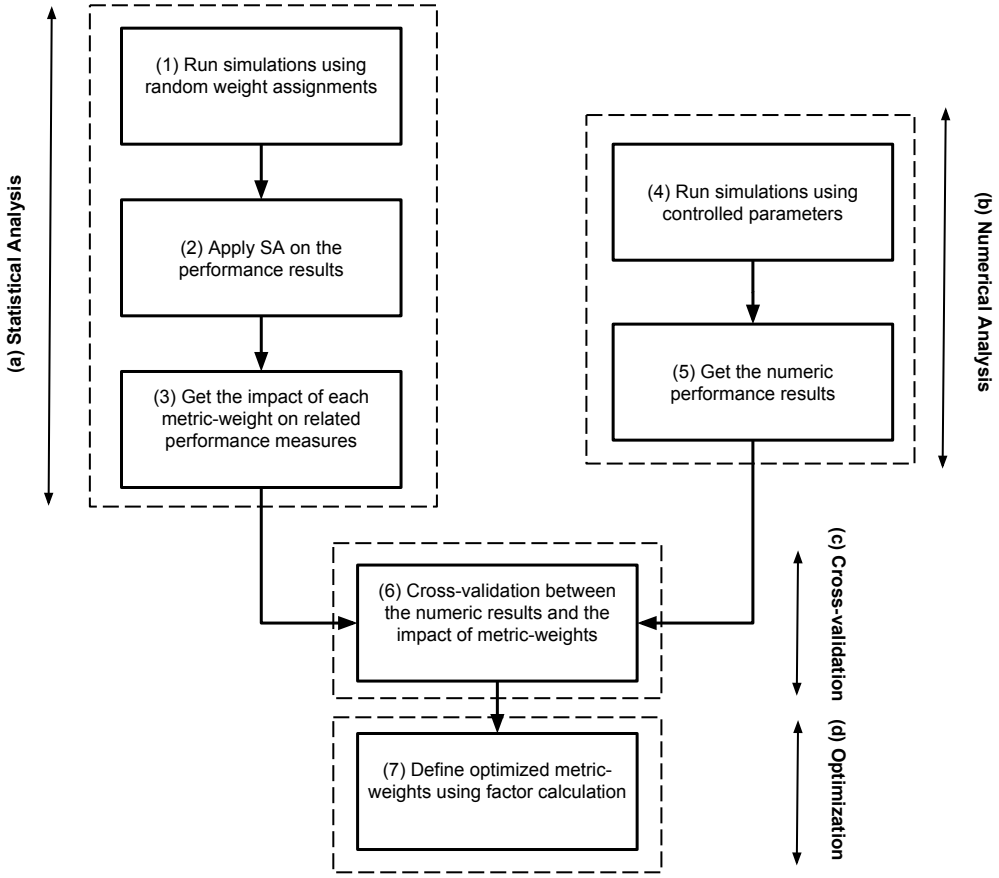


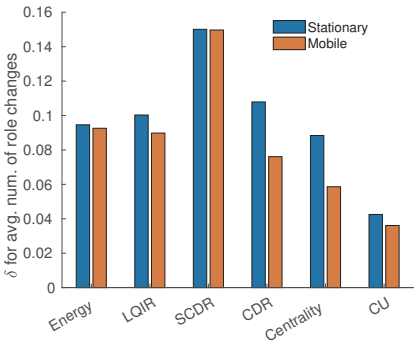
Figure 6.1: The steps for sensitivity analysis. While steps (1), (2) and (3) give the statistical results of the Moment-independent Delta Analysis, steps (4) and (5) are required to understand if a metric has a positive or negative effect on the related performance measure. Step (6) gathers the statistical results and metric-effect to reveal the effects of the impactful parameters. At step (7), the exact weight values are found to optimize DCA using the factor system.

a number of simulations are conducted to obtain performance measures presented in Table 6.1. Such process is repeated for 5000 different set of weights and practically 5000 different versions of DCA are shaped. All those versions are realized in the simulation environment to collect performance results for each. Then, all those performance results and related randomly-defined metric-weight sets are analyzed together to comprehend the effects of changing weights on the performance measures at step (2). Moment-Independent Delta Analysis method returns a δ value that indicates the relationship between a metric-weight and performance measure analyzing all different sets of weights. The interpretation of (δ) is quite straightforward: the metric-weight with higher δ value is more effective on related performance measure. In other words, a performance measure is more sensitive to the metric-weights having higher δ values. At step (3), the results of SA (i.e., indicator δ) are evaluated and the metrics with the most impactful weights are found for stationary and mobile scenarios for each goal.

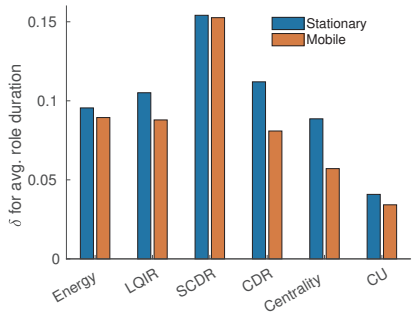
Figure 6.2 shows the sensitivity of all metric-weights in stationary and mobile scenarios to the performance measures for stability. Figure 6.2a shows that the weights of *SCDR*, *CDR*, *LQIR* and *SCDR*, *CDR*, *Energy* are the most impactful ones on the number of role changes in stationary and mobile scenarios respectively. δ values in Figure 6.2b shows nearly the same results for the average role duration in both scenarios. For cluster selection metrics, *Contraction* and *Traffic Density* are the top influential ones as shown in Figure 6.2c and Figure 6.2d.

Figure 6.3 shows the sensitivity of all metric-weights to the performance measures for energy efficiency. In Figure 6.3a, there is not a significant difference between δ values of the metric-weights. However, those values are observably different in Figure 6.3b. In terms of energy efficiency, the weights of *SCDR*, *CDR* and *Energy* are the most impactful ones for stationary and mobile scenarios. For cluster selection metrics, Figure 6.3c and Figure 6.3d show quite similar results. While the weight of *Cliqueness* is dominant for all scenarios, *Traffic Density* and *Contraction* are influential for stationary and mobile scenarios, respectively.

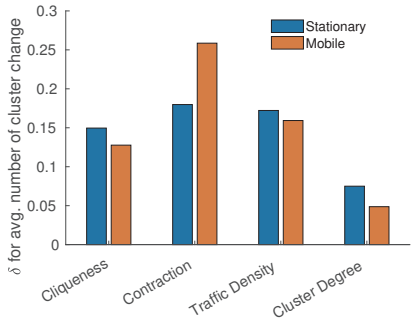
Figure 6.4 represents SA results for QoS-related performance measures. Figure 6.4a



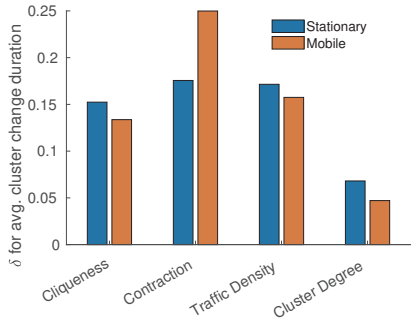
(a) δ of each CH selection metrics on the number of role changes



(b) δ of each CH selection metrics on the average role duration



(c) δ of each cluster selection metrics on the number of cluster changes

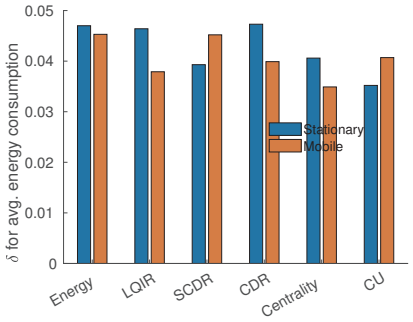


(d) δ of each cluster selection metrics on the average cluster change duration

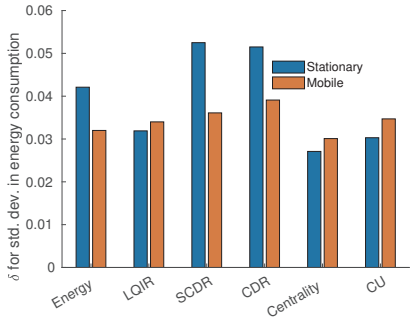
Figure 6.2: The SA indicator δ in stationary and mobile scenarios considering stability. Figure 6.2a and Figure 6.2b show the evaluation of cluster head selection metrics and performance measures to find an optimal metric-weight set to calculate a node score that boosts stability. On the other hand, Figure 6.2c and Figure 6.2d aim to find optimal metric-weight set for stability-boosting dependability score considering cluster selection metrics.

shows that *SCDR*, *CDR* and *Energy* are significantly influential in terms of control overhead. For QoS, the weight of *Contraction* has an observable effect on both packet delivery ratio and end-to-end delay as shown in Figure 6.4b and Figure 6.4c. While the weight of *Cluster Degree* is more significant for stationary scenarios, *Cliqueness* is considered as the second impactful metric for mobile scenarios.

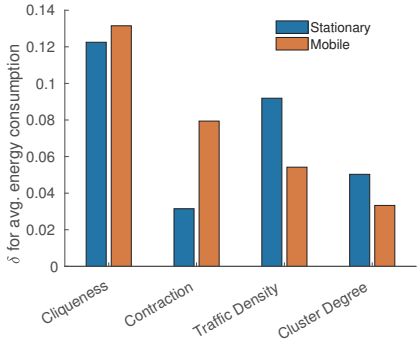
Table 6.2 presents the concrete outcome of the statistical analysis phase. For each scenario and objective, the most impactful metric-weights are shown in this table. The next phase, numerical analysis, is the intermediary step before understanding



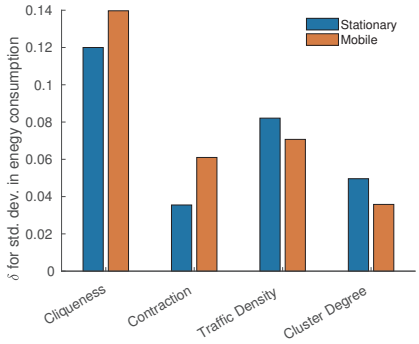
(a) δ of each CH selection metrics on the average energy consumption per node



(b) δ of each CH selection metrics on the standard deviation in energy consumption



(c) δ of each cluster selection metrics on the average energy consumption per node



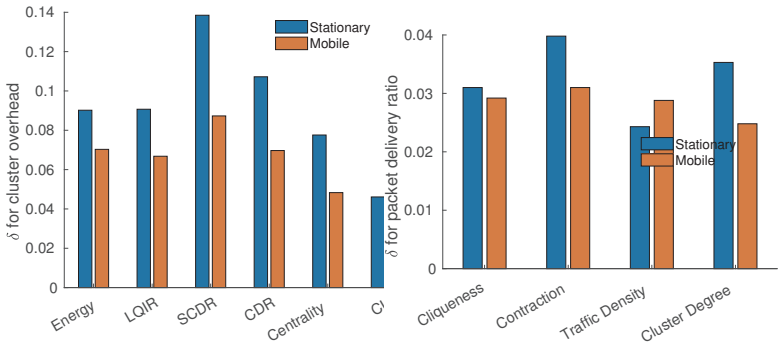
(d) δ of each cluster selection metrics on the standard deviation in energy consumption

Figure 6.3: The SA indicator δ in stationary and mobile scenarios considering energy efficiency. Figure 6.3a and Figure 6.3b show the evaluation of cluster head selection metrics and performance measures to find optimal metric-weight set for node score. Figure 6.3c and Figure 6.3d aim to find optimal metric-weight set for cluster selection metrics.

if the impacts of those metric-weights are positive or negative on the performance measures.

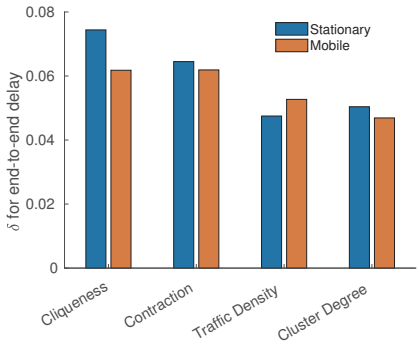
6.1.1.2 Numerical Analysis

After the phase (a), the numeric results are collected in this phase to understand the practical impacts of metrics, i.e., they positively or negatively affect the performance measures. To obtain the numerical results at step (4), the metric-weights are assigned as follows. Concentrating one metric at a time, if metric i is weighted as w_i, others are equally weighted as $(1 - w_i)/(n - 1)$ where n is the total number of metrics.



(a) δ of each CH selection metrics on the clustering control overhead

(b) δ of each cluster selection metrics on the packet delivery ratio



(c) δ of each cluster selection metrics on the end-to-end delay

Figure 6.4: The SA indicator δ in stationary and mobile scenarios considering QoS. Figure 6.4a shows the evaluation of cluster head selection metrics and performance measures to find the optimal metric-weight set to calculate node score. Figure 6.4b and Figure 6.4c aim to find the optimal metric-weight set for cluster selection metrics.

Table 6.3 and Table 6.4 are the presentations of example numeric results for different performance measures in stationary and mobile scenarios. In those tables, each metric has a different weight degree. If a metric is weighted as "All", it means that its weight is $w_i = 1.0$ and others are just $w_{i'} = 0.0$. "High", for example, means that related metric-weight is 0.75 and others are equally distributed as $(1 - 0.75)/5$ for cluster head selection metrics. Note that, since there are 4 metrics for cluster selection, the equal distribution becomes $(1 - 0.75)/3$ when a metric is weighted as "High". Similarly, "Medium" and "Low" are weighted as 0.5 and 0.25 respectively. Having "None" represents $w_i = 0.0$ and it leads $w_{i'} = 0.2$ (or $w_{i'} = 0.33$ for cluster selection metrics) for all other metrics. Such tables are generated for each performance measure for sta-

Table 6.2: The most influential metrics in descending order for stable and mobile scenarios.

Goal	**CH Selection Metric-weights**		**Cluster Selection Metric-weights**	
	Stationary	**Mobile**	**Stationary**	**Mobile**
Stability	SCDR	SCDR	Contraction	Contraction
	CDR	Energy	Traffic Density	Traffic Density
	LQIR	LQIR		
Energy efficiency	SCDR	Energy	Cliqueness	Cliqueness
	CDR	CDR	Traffic Density	Contraction
	Energy	SCDR		
Control overhead/QoS	SCDR	SCDR	Contraction	Contraction
	CDR	CDR	Cluster Degree	Cliqueness
	Energy	Energy		

tionary and mobile scenarios. However, only Table 6.3 and Table 6.4 are shown here to exemplify the technique to keep the results comprehensible.

The numeric results show the actual impacts of the metrics on performance measures. For example, in Table 6.3, the effects of metric-weights on the number of clustering control packets are given. As seen, while the weight of *Energy* is decreasing (going from "All" to "None"), the number of clustering control packets are increasing. It is concluded as the more weight for *Energy* leads to less control overhead, therefore it has a positive (decreasing) impact on control overhead and is represented with the symbol ($\uparrow$). However, the situation is vice-versa for *SCDR* and *CDR*, thus they are marked with ($\downarrow$). Lastly, a regular change cannot be observed in the number of clustering control packets for the other metrics; they are shown as ($\sim$), which implies irregularity. Note that, the interpretation of Table 6.4 is quite similar. After this process is repeated for each performance measure in Table 6.1, the real impact of each metric-weight (for both cluster head and cluster selection) is revealed.

Table 6.3: The numeric results of the number of effective cluster control packets in stationary scenarios. If increasing weight degree decreases the number of control packets then related parameter has a positive impact on the performance measure. *Energy*, here, has such positive impact and in contrast, *SCDR* and *CDR* have a negative impact. In others, irregularities are observed.

Parameters	**Weight Degree**					**Impact**
	All	**High**	**Medium**	**Low**	**None**	
Energy	134.42	143.08	146.94	147.83	149.13	$\uparrow$
LQIR	136.52	138.04	142.43	139.25	150.27	$\sim$
SCDR	156.9	152.80	150.12	149.24	146.65	$\downarrow$
CDR	180.31	159.68	149.84	147.16	146.24	$\downarrow$
Centrality	144.84	143.31	148.09	148.81	146.16	$\sim$
CU	155.93	147.06	151.45	146.11	149.18	$\sim$

Table 6.4: The numeric results of the average cluster change duration in mobile scenarios. If increasing weight degree increases the duration, it implies more stable clusters and related metric has a positive impact on the performance metric. *Contraction* has such impact and *Traffic Density* does not.

Parameters	**Weight Degree**					**Impact**
	All	**High**	**Medium**	**Low**	**None**	
Cliqueness	20.25	17.27	17.077	18.37	18.99	$\sim$
Contraction	22.30	19.82	18.92	18.45	17.32	$\uparrow$
Cluster Degree	18.82	18.09	18.17	18.36	21.16	$\sim$
Traffic Density	16.50	17.72	18.04	18.15	18.31	$\downarrow$

6.1.1.3 Cross-validation

After finding the most influential metrics in phase (a) and identifying their actual effects in phase (b), it is easy to make a cross-validation to get complete results of SA. Table 6.5 represents the concrete outcome of step (6) combining all the information obtained in previous phases. For stationary and mobile scenarios, and focusing on different objectives, almost every case requires different metric-weight assignment considering varying metrics and their impacts. For example, while *CDR* metric is positively affecting energy efficiency in stationary scenarios, its effect is negative on control overhead and QoS in mobile scenarios. The interpretation of the indicators in Table 6.5 is quite similar to the ones in Table 6.3. While ($\uparrow$) symbol means that higher weights of related parameters have a positive effect, ($\downarrow$) implies that lower weights tend to have a positive impact. ($\leftrightarrow$) shows mediocre values of a weight has the most positive effect. Lastly, even if the parameters with ($\sim$) have higher impacts, there is no such regular increasing or decreasing effect of them on the related performance measure. Note that, those indicators are **not** related to actual values of those metrics calculated during the network lifetime; instead they are related to the weights assigned to such parameters. In the rest of this section, the actual effects of the metric-weights found after cross-validation are discussed for each goal.

Stability: According to Table 6.5, the different conditions in stationary and mobile scenarios require to adjust metric-weights considering different goals. In stationary scenarios, for a more stable network in terms of related performance measure presented in Table 6.1, *SCDR*, *CDR* and *Energy* are the most influential (or sensitive) metrics. While *Energy* positively affects the stability, *SCDR* and *CDR* have negative impacts for cluster head selection. Ideally, *SCDR* and *CDR* are not supposed to change during network lifetime in stationary networks. However, the interference in the medium due to random packet traffic (and also triggered routing traffic) between many nodes easily changes the values of those metrics since control packets may be lost in such conditions. Therefore, their possible and frequent changes also trigger the selection of new cluster heads when their weights are higher. Note that, deployment

Table 6.5: The most influential parameters in descending order for stable and mobile scenarios. While the parameters indicated with (↑) has a positive effect, the others with (↓) have a negative effect. (↔) indicates a mediocre value gives the best results and the ones with (∼) do not show a regularity even if they have high impact. The factorial values that are used to calculate actual weights are also given in the latest row.

Goal	**CH Selection Metric-weights**				**Cluster Selection Metric-weights**			
	Stationary		**Mobile**		**Stationary**		**Mobile**	
Stability	SCDR	↓	SCDR	↓	Contraction	↑	Contraction	↑
	CDR	↓	Energy	↑	Traffic Density	∼	Traffic Density	↓
	LQIR	↑	LQIR	↑				
Energy efficiency	SCDR	∼	Energy	↓	Cliqueness	↓	Cliqueness	↓
	CDR	↑	CDR	∼	Traffic Density	↔	Contraction	↔
	Energy	∼	SCDR	↔				
Control overhead/QoS	SCDR	↓	SCDR	↓	Contraction	↑	Contraction	↔
	CDR	↓	CDR	↓	Cluster Degree	∼	Cliqueness	↔
	Energy	↑	Energy	↑				
Factors	(↑): 5 (↔): 3 (∼): 1 (↓): 1							

of a contention-free link layer protocol would probably change the effects of those metrics. *LQIR*, in contrast, positively affects since stronger or stable connections bring more stable neighborhoods. For cluster selection metric-weights, *Contraction* has a positive effect on stability by default since it indicates cluster constancy. *Traffic Density*, in contrast, does not have a regular effect but still have a significant impact on the stability-related performance measures.

In mobile scenarios, *SCDR* has a negative impact on stability since constantly changing node degrees due to mobility lead to frequent cluster head reselections, and unsettle current clustered structure. energy consumption does not dramatically change from node to node and only CHs consume relatively more energy. Therefore, it does not directly affect ordinary nodes but only triggers reselection when a CH's node score decreases dramatically due to high energy consumption. *LQIR* and *Contraction* have positive effects as they do in stationary scenarios. However, focusing on *Traffic Density* negatively affects stability since the randomness in packet traffic can confuse

nodes to change clusters often and violates stability.

Energy Efficiency: Energy efficiency metrics are mostly affected by *SCDR*, *CDR* and *Energy* in stationary scenarios. However, *SCDR* and *Energy* have no regular effects in terms of increasing or decreasing metric-weight values. Note that, selecting nodes with higher energy as cluster head may not lead a better energy efficiency. Even if it decreases the standard deviation in energy consumption (i.e., provides fairer energy consumption between nodes), the total energy consumption does not have to be affected positively. In contrast, *CDR* indicates highly-connected groups and focusing on higher-*CDR* cluster heads have more control over their neighborhood and eventually packet traffic with less routing overhead, and energy consumption. However, while individual *CDR* value has a positive impact on energy efficiency influencing cluster head selection, *Cliqueness* does not show the same effect for cluster selection. The analysis reveals that *Cliqueness*, which is strongly related to *CDR*, decreases energy efficiency. On the other hand, mediocre metric-weight for *Traffic Density* gives the best results. Both high and low traffic clusters are not desired, mediocre values lead to a decent number of known alternative routes, and also low interference inside a cluster.

In mobile scenarios, there is no metric having a dominantly positive impact with a higher metric-weight. Even if *CDR* is the second most impactful parameter, its influence does not show regularity. *Energy* has an observable negative effect when it is considered for cluster head selection. In fact, selecting the highest-energy node as cluster head can easily disorganize a mobile network by forcing more inactive (low-energy consumer) node to be cluster head, and it eventually leads to extra route discovery and clustering processes. This comment is also held for *SCDR*. However, selecting the nodes with higher degrees as cluster heads still saves most of the nodes from triggering routing processes repeatedly. That is, packet traffic is managed by fewer number of cluster heads that cover a larger number of ordinary nodes and this leads to higher energy efficiency. *Contraction* takes higher weight-share. When nodes join clusters which do not tend to dismiss, energy efficiency is eventually increasing due to, for example, using once-determined routes in a stable cluster.

QoS: In terms of control overhead in both stationary and mobile scenarios, the picture is the same and very similar to the results in stability case: negative impacts of *SCDR* and *CDR*, and positive impact of *Energy*. As shown in Table 6.5, *SCDR* is not taking high metric-weight values in nearly any case since small degree changes possibly cause important changes in clustered structure when its weight is increased. Therefore, networks become more sensitive to degree changes. While stability and energy efficiency are decreasing, control overhead is increasing due to the increscent weight of *SCDR*. In contrast, *Energy* has a decreasing role for the control overhead in both scenarios. For cluster selection, *Contraction* has a positive impact as usual, promoting more stable clusters it increases QoS in terms of packet delivery ratio and end-to-end delay. In stationary scenarios, *Cluster Degree* seems effective for QoS but no observable pattern exists for it. On the other hand, *Cliqueness* helps to satisfy QoS by promoting more connected clusters to join.

All in all, after the first 3 phases of SA framework, it is concluded that each case (considering different goals) requires to be focused on different metrics and metric-weight values. Even the sensitivity analysis charts indicate a starting-point to reshape the details of the algorithm, it is still difficult to validate their results with a high-precision. Since the node metric-weights are dependent on each other, a direct inference of the singular effects of them is not possible. Therefore, further investigation with different weight assignments may show different results. Besides, those sensitivity results are directly related to the performance measures. Different performance measures that are related to different goals may reveal more goal-specific results as well.

The final question is how to find some sets of metric-weights that give us the best results for the related performance measures. In the next phase, this question is answered using the outcomes of the cross-validation.

6.1.1.4 Optimization

In this phase, the optimized metric-weights are found for different objectives. The main purpose of the cross-validation (and more generally SA) is to decide the op-

timum weight w_i to assign to each metric. The indicators in the last row of Table 6.5 (i.e., ($\uparrow$), ($\leftrightarrow$), ($\downarrow$), and ($\sim$)) are the concrete representations of weight impacts and need to be correlated among themselves (i.e., finding relative impact of each indicator) to obtain final weight values. Therefore, the row shows this correlation by **factors**. Using those factors, the weight for parameter *i* is calculated as

$$w_i = \frac{f_i}{\sum_{j=1}^{k} f_j}, \tag{6.1}$$

where f_i is the factor of the weight for metric *i*. The factors are assigned to optimize the impact of the weights and get the best result for related performance measure. They are numerical values and represented by indicators as shown in the last row of Table 6.5. While higher factors try to increase the fraction of the metrics in the total score (node or dependability score) by giving them higher weights, lower ones decrease their share. Accordingly, the weights with ($\uparrow$) are assigned with the highest factor 5, ($\leftrightarrow$) have a mediocre factor 3, and ($\downarrow$) and ($\sim$) have the lowest ones as 1. Note that, those factors are the design parameters for the framework and reflect the overall relationship of metric-weights.

j in (6.1) is bounded by [1,*k*] where *k* could be any number less than the total number of metrics. $k = 3$ is defined in the framework. Therefore, only three most influential metrics are taken into consideration as shown in Table 6.5. That is, the metrics that only have the real impact on the actual measurements can be considered. For instance, the top three impactful (i.e., the three highest δ value in sensitivity analysis results) for CH selection weights and the top two for cluster selection weights are selected. Note that, increasing the number of metrics to share between nodes means larger control messages and overhead. Since others (i.e., not considered parameters) have no numerable impact according to the analysis, they are just omitted and concluded as not necessary for particular performance goals.

After step (6) where the cross-validation is performed and Table 6.5 is obtained, one can evaluate the exact metric-weights to optimize DCA with respect to different objectives at step (7) of Figure 6.1. Then, the objective-based weight values for both

Table 6.6: The metric-weights for different objectives and scenarios. Those values indicate the weights for cluster head selection and cluster selection metrics to optimize DCA considering different goals in stationary and mobile scenarios.

Goal	Scenario	CH Selection Metric-weights						Cluster Selection Metric-weights			
		Energy	LQIR	SCDR	CDR	Centrality	CU	Cliq.	Cont.	Cl. Dg.	Tr. Den.
Stability	Stationary	0.00	0.71	0.14	0.14	0.00	0.00	0.00	0.71	0.00	0.29
	Mobile	0.45	0.45	0.09	0.00	0.00	0.00	0.00	0.83	0.00	0.17
Energy efficiency	Stationary	0.22	0.00	0.22	0.56	0.00	0.00	0.25	0.00	0.00	0.75
	Mobile	0.17	0.00	0.50	0.33	0.00	0.00	0.25	0.75	0.00	0.00
Control overhead/QoS	Stationary	0.71	0.00	0.14	0.14	0.00	0.00	0.00	0.71	0.29	0.00
	Mobile	0.71	0.00	0.14	0.14	0.00	0.00	0.50	0.50	0.00	0.00

cluster head and cluster selection metrics are calculated as shown in Table 6.6 using the factors to find a numerical relationship between weights as defined in (6.1). For instance, to optimize DCA for stability in stationary scenarios, one needs to consider the first row of Table 6.6 and assign indicated weight-values to related metrics. Therefore, those are ready-to-use values to employ for designing the objective-oriented weighted clustering algorithm, DCA.

6.1.2 Results

All tests are conducted in OMNeT++ using the implementation presented in Chapter 3. The simulation is fixed to $200\,\mathrm{m} \times 200\,\mathrm{m}$. Considering the number of nodes and node speeds, each distinct case is examined in 200-repetition batches where each repetition simulates $60\,\mathrm{s}$ network lifetime. The average performance measures and confidence intervals of each batch are recorded. For the physical channel, free space path loss is deployed with a fixed $-90\,\mathrm{dB\,m}$ background noise. Besides, YANS error model, which is commonly implemented in most of the network simulators, is used to evaluate errors stem from channel conditions. The state-based radio model is a built-in module in OMNeT++ and implements the state-based radio. In this model, while $150\,\mathrm{mW}$ is consumed for packet transmission, it is $60\,\mathrm{mW}$ and $2\,\mathrm{mW}$ for reception and idle states respectively. Lastly, Random Waypoint mobility model [65] is used to define node behaviors under mobility. In this model, nodes periodically move through a random direction with a specific speed, which changes between $2\,\mathrm{km/h}$-$10\,\mathrm{km/h}$.

30% of the nodes are randomly selected to move with increasing speeds in random directions for mobile scenarios. All parameters are shown in Table 6.7. To measure the success in data transfer and end-to-end delay, the UDP application sends a packet in every 2 s between a randomly created set of source and destination nodes. The size of a data packet is defined as 300 B, and the packets are sent at once (i.e., not fragmented).

Table 6.7: The values of the simulation parameters.

Parameter	**Value**
Area size	$200\,\mathrm{m} \times 200\,\mathrm{m}$
Runs per batch	200
Scenario duration	60 s
Transmission power per node	0.08 mW
Node density	$0.001\,\mathrm{node/m^2}$-$0.0015\,\mathrm{node/m^2}$
Ratio of mobile nodes	30%
Speed of nodes	2 km/h-10 km/h
Background noise	$-90\,\mathrm{dB\,m}$
Path loss model	Free space
Error model	YANS
Power consumption model	150 mW Tx 60 mW Rx 2 mW Idle
Mobility model	Random Waypoint [65]

The results are collected for five different algorithms where two of them are the different configurations of DCA. The first one is optimized DCA (DCA-o) which deploys metric-weights that are obtained as a result of the sensitivity analysis as presented in Section 6.1.1.4. DCA-o is run for each different objective and scenario using the metric-weight sets presented in Table 6.6. The 0-weighted metrics are the omitted ones due to their low impacts. Designing the alternative versions of DCA using the weight sets, it is aimed to show (a) weight optimization with the sensitivity analysis gives better results for different cases and (b) DCA-o shows a better performance than

its opponents being adapted to different goals. The other four algorithms are equal DCA (DCA-e) where metric-weights are equally distributed among metrics of DCA, Lowest-ID Clustering (LI) [13], Highest-degree Clustering (HD) [66] and Highest-energy Clustering (HE) [67]. Last three clustering algorithms are quite popular and widely used for benchmarking. The comparison is performed in two major scenarios, stationary and mobile. In each scenario, different goals are separately evaluated with related performance measures. Note that, even though some of the performance measures are given as a ratio, they are presented and compared scaling to the percentage for the sake of clarity.

Another important issue is the different periods used in DCA. They are presented in Table 4.1 at the beginning and the discussion about those periods is presented in Table 6.8. In the light of this discussion, DCA is performed with different period values and eventually, a manually-optimized set of values is used for the simulations.

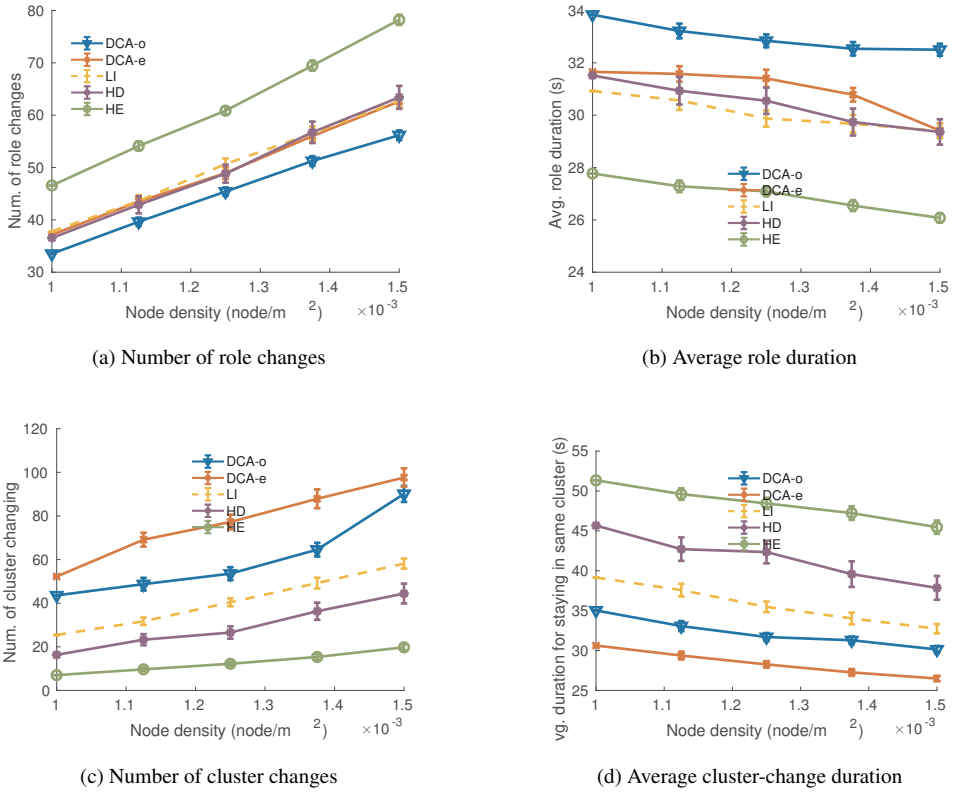
In the rest of this section, the simulation results are discussed for different goals that are stability, energy efficiency and QoS in stationary and mobile scenarios observing the effects of node density and node speed.

Stability: Figure 6.5 shows the results for the stability-related measures in stationary scenarios. DCA shows better performance than all other algorithms in terms of the number of role changes and average role duration. Focusing on *LQIR* and *Contraction*, DCA succeeds in minimizing the number of role changes (i.e., cluster head to ordinary node or vice-versa) and maximizing role duration as shown in Fig 6.5a and Figure 6.5b. On the other hand, since HE mostly considers energy consumption to select cluster heads, nodes become more sensitive to claim themselves as cluster heads with continuously changing residual energy. However, Fig 6.5c and Figure 6.5d show that even if DCA-o is better than DCA-e in terms of cluster changes and duration for staying in the same cluster, there are fewer cluster changes in other algorithms considering those metrics. Because in DCA, a node changes its cluster depending on the dependability score which is affected by different metrics. Since DCA keeps clusters dependable, nodes change their clusters as they aim to be a member of the most

Table 6.8: Possible effects of the related actions in different periods

Parameter	Small Periods	Large Periods
T_{ctrl}	• Freshness of neighborhood info. • Updated scores • More reliable routes • Large number of broadcast messages • Higher reliability but lower stability	• Obsolete information of neighborhood • Obsolete node and dependability scores • Less reliable routes • Fewer broadcast messages • Higher stability but lower reliability
t_{boot}	• Guaranteed convergence • Proper density-orientation in clusters	• Fast-shift to main algorithm • Nearly random clustering
T_{claim}	• Lower stability • Higher number of role changes and longer role duration • Higher oscillation possibility • Fairly distributed energy consumption • Construction onto fresh neighborhood info.	• Higher stability • Fewer number of role changes and longer role duration • Settled backbone • More energy consumption in particular nodes i.e., cluster heads
T_{tick}	• Less tolerance to broken links • Increasing false-negatives for neighbor nodes • Updated neighborhood info. • Decreasing efficiency in routing	• Not convenient for mobile scenarios • Less reliable routes • Obsolete node and dependability scores
T_{SAM}	• Fresh topology info. in a wider range • More reliable hybrid routing • More energy consumption in the backbone	• Obsolete topology info. • Less reliable hybrid routing • Less control overhead through the backbone
T_{dpnd}	• Increasing possibility to join another cluster for nodes • Cluster selection based on fresh info.	• More stability • Less reliable cluster selection

dependable (i.e., cluster with the highest score) and this is not a case any of the other algorithms. Eventually, the stability goal is satisfied keeping clustered formation stable in DCA. However, nodes always try to get into the most dependable cluster and this issue increases inter-cluster node changes.



(a) Number of role changes

(b) Average role duration

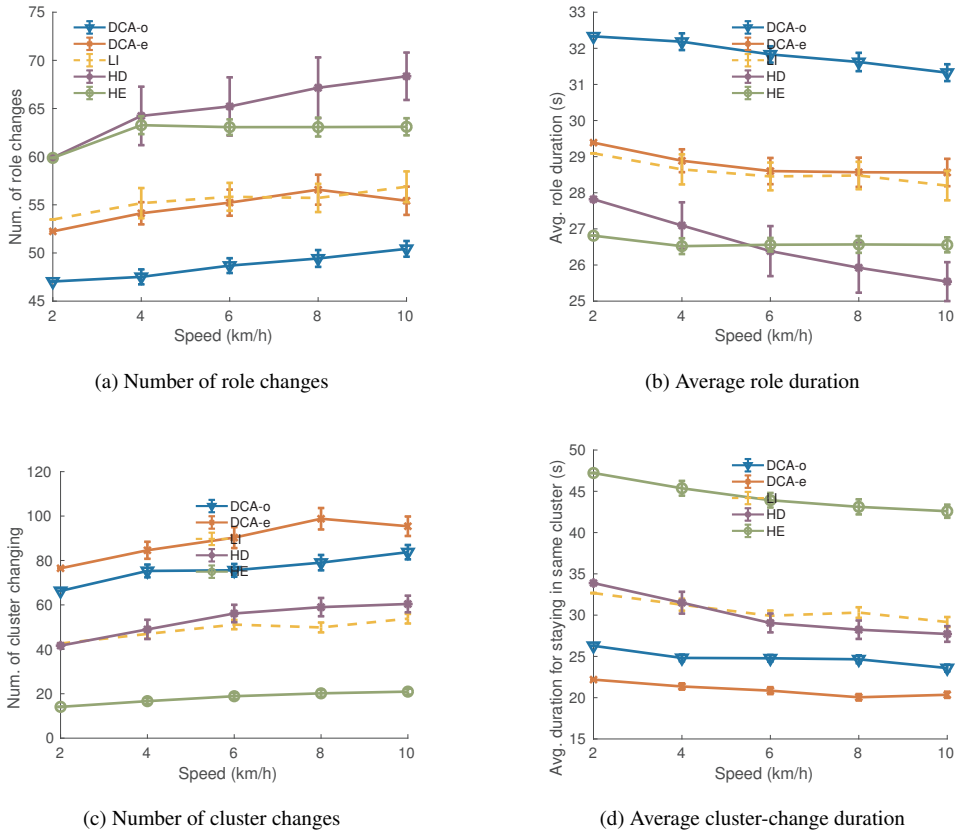
(c) Number of cluster changes

(d) Average cluster-change duration

Figure 6.5: The effects of increasing node density on stability metrics in stationary scenarios

In mobile scenarios, the results are quite similar to the ones in stationary scenarios as shown in Figure 6.6. Figure 6.6a and Figure 6.6b show that DCA-o is much more effective than DCA-e and other algorithms in terms of the number of role changes and average role duration. Differently, HD becomes more sensitive to mobility. Constantly changing neighborhood and node degree negatively affect due to HD's cluster head selection method. Because of the same issue, which is the cluster selection to join a more dependable cluster, inter-cluster node changes are still higher in DCA in terms of the number of cluster changing as shown in Figure 6.6c. Besides, Figure 6.6d shows that DCA-o is better in average duration in the same cluster than DCA-e. However, this measure is higher in other algorithms because they do not have a

particular cluster-changing mechanism. Considering those metrics, HE is more insensitive to mobility. Apart from those, DCA-o outperforms DCA-e considering any performance measure in terms of stability.



(a) Number of role changes

(b) Average role duration

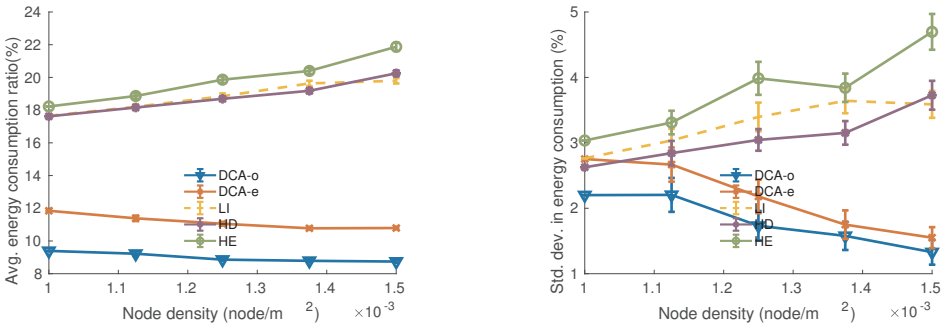
(c) Number of cluster changes

(d) Average cluster-change duration

Figure 6.6: The effects of increasing speed on stability metrics in mobile scenarios

Energy Efficiency: Figure 6.7 and Figure 6.8 show the simulation results in stationary and mobile scenarios, respectively. Figure 6.7a and Figure 6.7b show that DCA-o is better than DCA-e for overall energy consumption and fair energy consumption in stationary scenarios. Smaller standard deviation in energy consumption means that the energy consumption among different nodes is close to each other. Therefore, over-consumption in a specific set of nodes is not an issue in DCA. As presented in Figure 6.7a, while other algorithms are very similar in terms of energy consumption, DCA is significantly better than all of them with nearly 10% less consumption. Besides, DCA is able to keep its performance with increasing number of nodes by selecting cluster heads effectively and leading them to choose dependable clusters. Therefore,

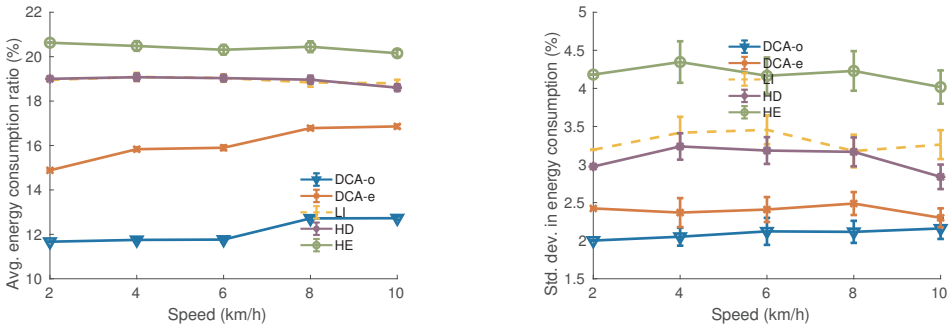
it is concluded that DCA is also a scalable clustering algorithm in terms of energy efficiency.



(a) Average energy consumption ratio (b) Standard deviation in energy consumption

Figure 6.7: The effects of increasing node density on energy efficiency metrics in stationary scenarios

On the other hand, mobility obviously affects energy efficiency negatively. Even if the order of compared algorithms is the same with stationary scenarios, nodes consume more energy to handle reclustering and routing processes due to mobility. Besides, Figure 6.8b shows that in mobile scenarios, the standard deviation in energy consumption for each algorithm is slightly higher than the stationary scenarios.

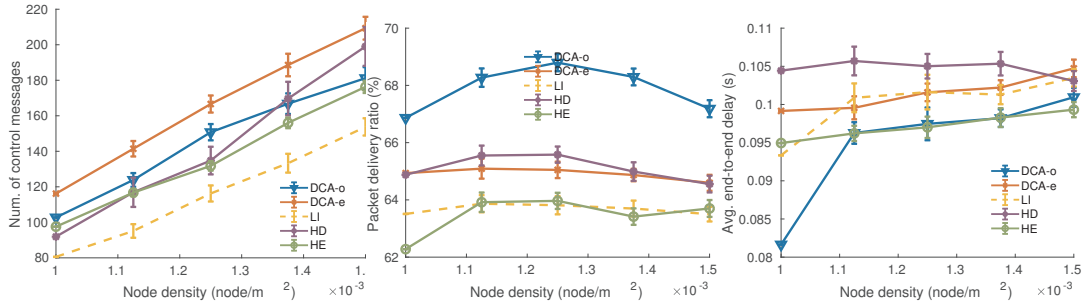


(a) Average energy consumption ratio (b) Standard deviation in energy consumption

Figure 6.8: The effects of increasing speed on energy efficiency metrics in mobile scenarios

QoS: DCA-o outperforms DCA-e in stationary scenarios with less control overhead, a higher PDR and a lower end-to-end delay. For PDR, DCA-o shows 3-6% better performance than all others as shown in Figure 6.9b. In stationary scenarios,

PDR mostly depends on effective cluster head selection and the stability in clustered structure since hop-to-hop packet traffic is performed through cluster heads. Note that, the main reason for packet loss in all algorithms is interference. PDR after a certain node density is starting to drop for all algorithms because of interference. Besides, it is also directly related to stability in clusters. For instance, while the number of role changes is the highest in HE, the packet delivery ratio is the lowest for that algorithm. In terms of end-to-end delay, DCA-o places in the lowest level in Figure 6.9c. The most important reason for the increase in end-to-end delay is repeating the routing process: if cluster heads change frequently, the number of route discoveries naturally increases to find the cluster in which destination node resides and the intermediary cluster heads reaching that cluster. Besides, more dependable clusters (i.e., clusters that can manage packet traffic effectively) promote the packet traffic for nodes as implied from DCA's higher performance. However, Figure 6.9a shows that since cluster changes (i.e., a node joining a different cluster than the current one it resides) are still frequently happening in DCA to stay in the most dependable cluster, DCA (both DCA-o and DCA-e) has slightly higher overhead than the others in stationary networks.



(a) Number of effective control packets (convergence)

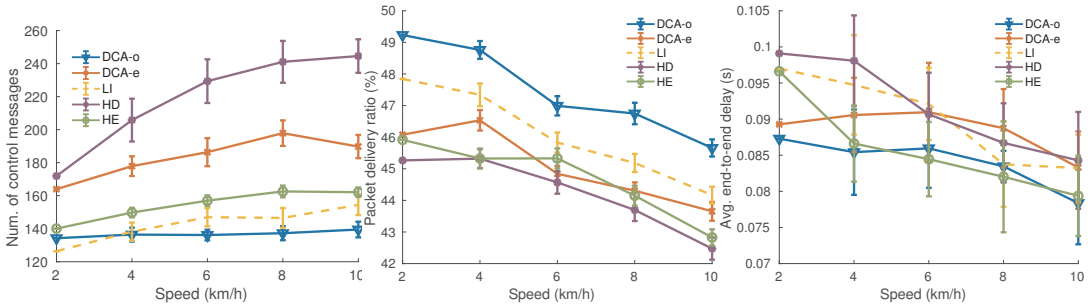
(b) Packet delivery ratio

(c) Average end-to-end delay

Figure 6.9: The effects of increasing node density on QoS metrics in stationary scenarios

Figure 6.10 shows QoS results under mobility. Similar to HE's situation in stationary scenarios, as frequent degree changes due to mobility significantly affect HD, it has the worst performance in terms of the number of effective control packets and PDR in mobile scenarios as shown in Figure 6.10b. Even though differences are more subtle,

DCA still outperforms other its opponents in terms of PDR. In contrast to stationary scenarios, Figure 6.10a shows that DCA-o causes observably less control overhead in comparison to both DCA-e and other algorithms. However, since mobility has an important impact on routes/routing, no algorithm is able to show a dominance for the end-to-end delay in Figure 6.10c.



(a) Number of effective control packets (convergence)

(b) Packet delivery ratio

(c) Average end-to-end delay

Figure 6.10: The effects of increasing speed on QoS metrics in mobile scenarios

Apart from individual node eligibility for being a cluster head, cluster dependability is also considered to maximize nodes' benefits and increase overall network performance. DCA-o performs better than the equally-weighted version in terms of a variety of performance metrics in different scenarios. Therefore, it is shown that the proposed SA framework for DCA practically helps adapting it to different scenarios which require stability, energy efficiency or QoS/low overhead. Note that, it is also applicable to any weighted clustering algorithm with different selection metrics and objectives. Besides, it is shown that DCA-o with goal-specific weight distribution works better than other benchmarking clustering algorithms namely LI, HD, and HE especially in mobile scenarios.

6.2 Performance Evaluation of CHRA

In this section, the performance evaluation of CHRA is presented. The simulation parameters are shown in Table 6.9. To represent other cluster-based routing algorithms which use on the backbone to carry all packets, CBRP is implemented as backbone routing upon clustered structure. Note that, the backbone routing is the common

technique that is primarily used in CBRP and then most of the hierarchical routing algorithms. To validate the advantages of CUPS architecture, CHRA is compared with (a) CBRP which has the major routing method used in almost all hierarchical routing algorithms and (b) AODV on flat topologies. Note that, the main purpose in the simulation design is to show the effects of control-user plane separation on fair energy consumption, energy efficiency and quality of service in terms of end-to-end delay and data-to-all ratio. Therefore, the generic implementation of CBRP is the most important comparison element in this simulation design.

Table 6.9: The values of the simulation parameters

Parameter	**Value**
Area size	$200\,\mathrm{m} \times 200\,\mathrm{m}$
Runs per batch	200
Scenario duration	$200\,\mathrm{s}$
Transmission radius per node	$40\,\mathrm{m}$
Node density	$0.001\,\mathrm{node/m^2}$-$0.0015\,\mathrm{node/m^2}$
Ratio of mobile nodes	30%
Speed of nodes	$2\,\mathrm{km/h}$-$10\,\mathrm{km/h}$
Path loss model	Free space
Power consumption model	$150\,\mathrm{mW}$ Tx $60\,\mathrm{mW}$ Rx $2\,\mathrm{mW}$ Idle
Background noise	$-90\,\mathrm{dB\,m}$
Mobility model	Random Waypoint [65]

Fro this study, while CHRA and CBRP are customarily implemented, the AODV algorithm is taken from built-in OMNeT++ modules. Simulations are conducted in both mobile and stationary scenarios with uniformly and nonuniformly distributed topologies. Triangular distribution is used for nonuniformly distributed network scenarios. It represents the topology where the majority of the nodes tend to gather around an area and some other nodes are spread as outliers. The results of all four cases are presented and discussed in the rest of this section. Even though some of the perfor-

mance measures are given as a ratio, they are presented and compared scaling to the percentage for the sake of clarity.

6.2.1 Stationary Scenarios

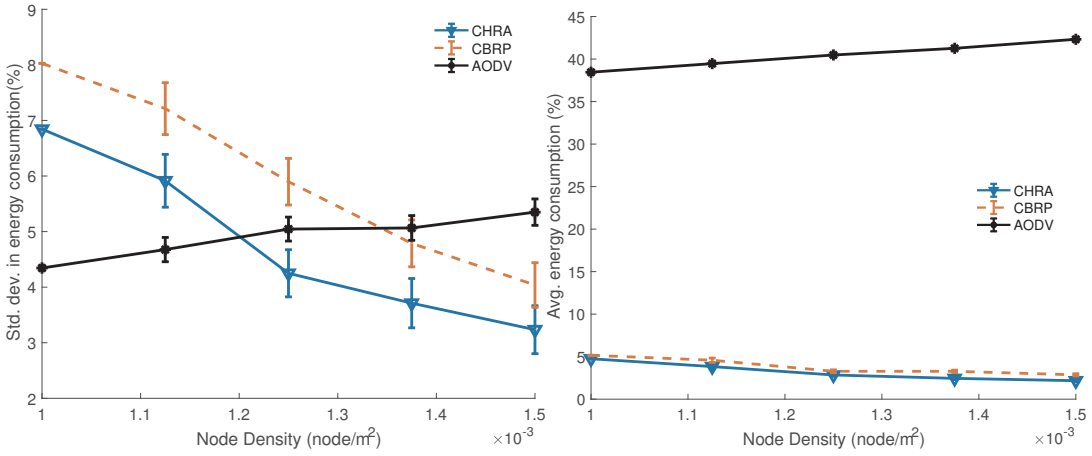
Figure 6.11 and Figure 6.12 show the effects of node density (i.e., increasing number of nodes) in uniform and nonuniform node distributions.

As seen in Figure 6.11b and Figure 6.12b, the energy consumption is nearly the same for CHRA and CBRP. However, there is a huge -and expected- difference between AODV and hierarchical routing methods, which are CBRP and CHRA, in the average energy consumption per node. Since control packets are consequently broadcast (that causes flooding) in AODV, the energy consumption is much higher than the others considering both signal transmission and reception costs. Figure 6.11a and Figure 6.12a reveal that even if the energy consumption is very close to each other, the standard deviation of the consumption in CHRA is lower than that of CBRP in both scenarios. It means that the difference in energy consumption between nodes are observably smaller in CHRA and this is a strong indication of fairer energy consumption. Therefore, the cross-interpretation of the average energy consumption and the standard deviation in energy consumption is quite important to understand the key outcomes of the control and data plane separation in terms of energy efficiency. Figure 6.11a also shows that the standard deviation in energy consumption is minimum in AODV. The reason is that nearly all nodes tend to broadcast control packets due to flooding mechanism and every node consumes similar energy even though it is much higher than the consumption in CBRP and CHRA. In contrast, Figure 6.12a shows that the standard deviation is the highest in AODV. Because, while the frequency of broadcast is much higher in a specific dense area in nonuniform distribution than the rest of the network where isolated nodes are seen. Eventually, the significant difference in standard deviation in energy consumption between such sparse area and dense area results with higher standard deviation. Another point is, while the node density (i.e number of nodes in the same area) is increasing, the standard deviation in energy consumption in CBRP and CHRA are decreasing because it directly increases

the number of alternative paths that can be found in both the data and control plane. Eventually, even fewer number of particular nodes are exhausted due to the preference of the same routes for end-to-end communication. This is not the case for AODV since the flooding of routing control packets still makes the largest proportion of the energy consumption. Therefore, the effects of the existence of alternative routes are hard to comprehend for AODV. Note that, the similar effect to increasing density can be concluded for changing distribution. While the results in Figure 6.11a are lying in 3-8% range, it is in the range of 1-3% as can be seen in Figure 6.12a. In nonuniform scenarios, nodes are gathered in particular areas with higher density creating many alternative routes, as it happens for increasing node density case in stationary scenarios.

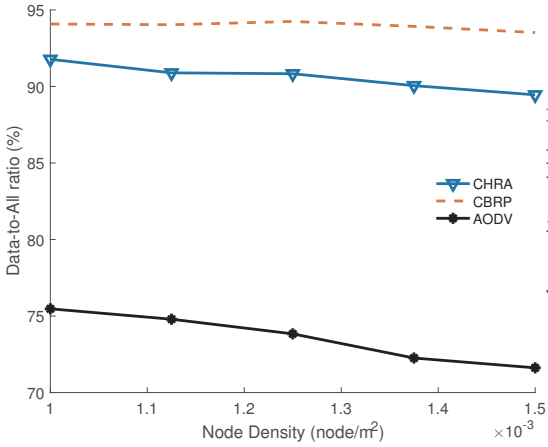
Figure 6.11e and Figure 6.12e show the number of routing control packets including route request and reply packets, and also repair, recovery and CSA packets for CHRA. While the overhead of CHRA is slightly higher than that of CBRP due to repair, recovery and CSA packets, AODV has the highest overhead in terms of the number of the control packets due to flooding control packets through the whole network.

The quality of service for three different algorithms is evaluated in terms of DAR and end-to-end delivery delay. In Figure 6.11c and Figure 6.12c, the DAR generally remains above 90% for CHRA and CBRP while AODV's is much lower, 75% at maximum. As seen in the figures, the DAR of CHRA and CBRP is nearly the same. Since the routing control overhead in CHRA is slightly higher due to CSA maintenance and route recovery, the DAR in CHRA is lower than CBRP around 2%. On the other hand, AODV shows a quite poor performance in terms of DAR due to high control overhead. The case is different for the end-to-end delay: after CSA is constructed, finding the shortest path is trivial for CHRA and both Figure 6.11d and Figure 6.12d show that CHRA has the lowest end-to-end delay. CBRP has very limited alternatives to choose a path which is constructed through cluster heads and gateways. Therefore, it is not easy to find the shortest path for end-to-end communication. Eventually, CHRA outperforms other two algorithms in terms of the end-to-end delay. Moreover, the difference between uniform and nonuniform scenarios is notable. Since nonuni-

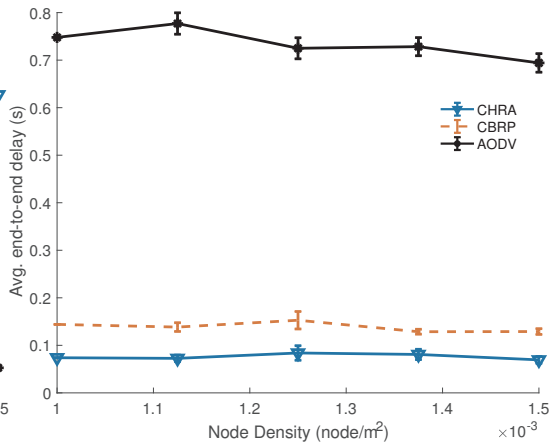


(a) Standart deviation in energy consumption

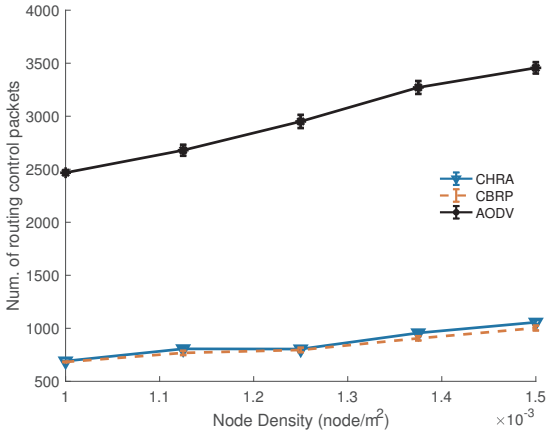
(b) Average energy consumption ratio per node



(c) Data-to-All ratio

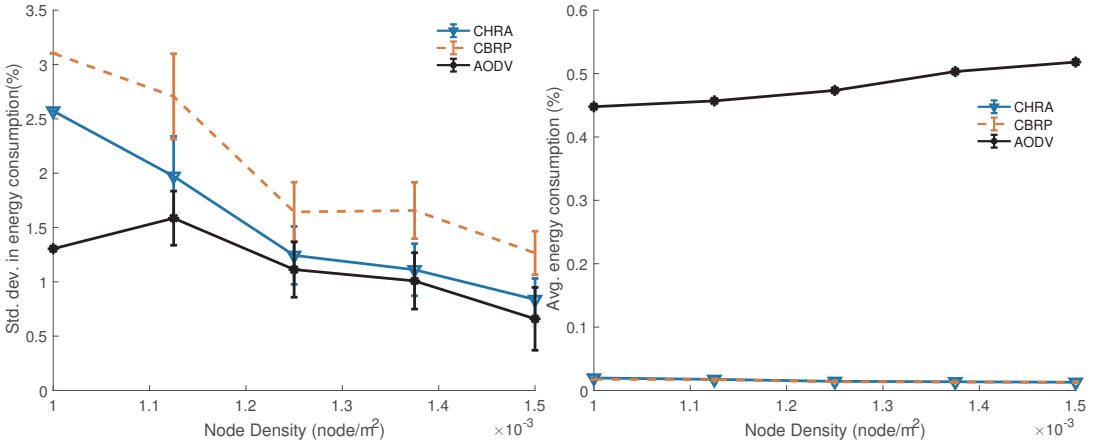


(d) Average end-to-end delay



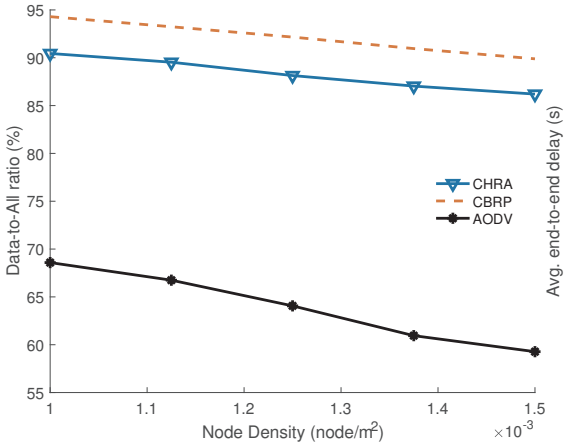
(e) Number of control packets

Figure 6.11: The effects of the network density in uniformly distributed scenarios

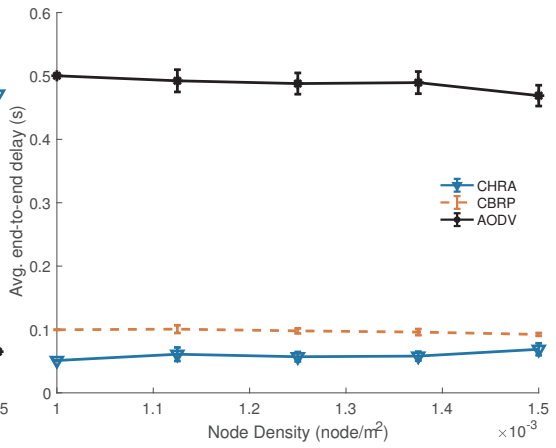


(a) Standart deviation in energy consumption

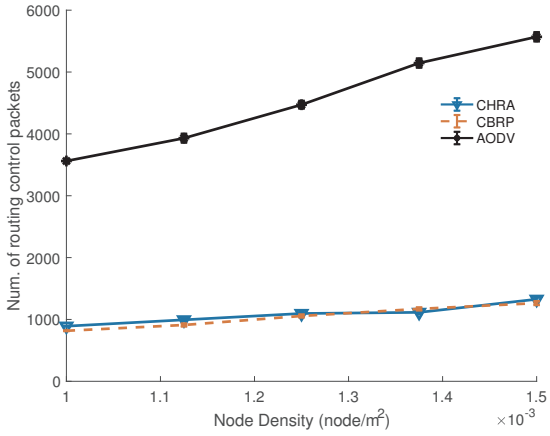
(b) Average energy consumption ratio per node



(c) Data-to-All ratio



(d) Average end-to-end delay



(e) Number of control packets

Figure 6.12: The effects of the network density in nonuniform scenarios

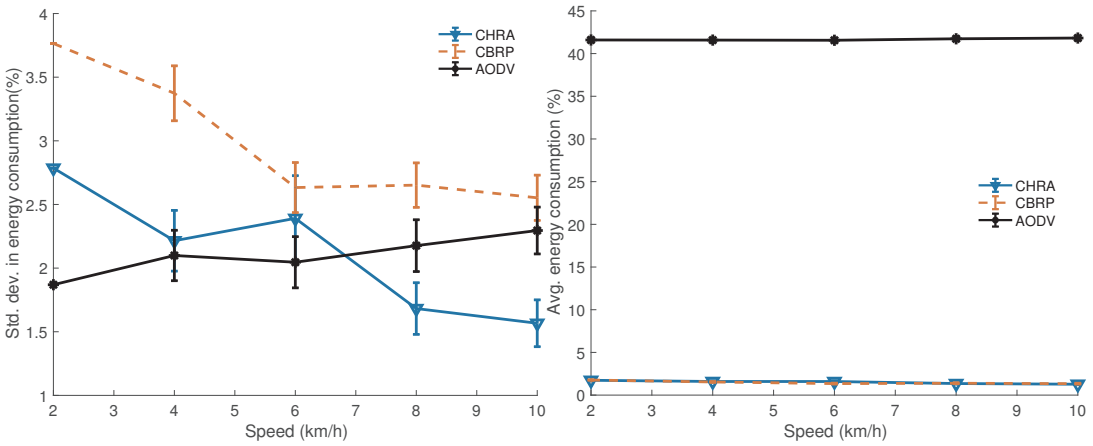
form deployment causes a denser formation, many nodes are placed around a certain area and this topology decreases average end-to-end delay in communication.

6.2.2 Mobile Scenarios

Figure 6.13 and Figure 6.14 show the effects of speed (i.e., increasing speed of nodes) in uniform and nonuniform node distributions. In mobile scenarios, the node density is fixed to $0.001\,25\,\mathrm{node\,m^2}$.

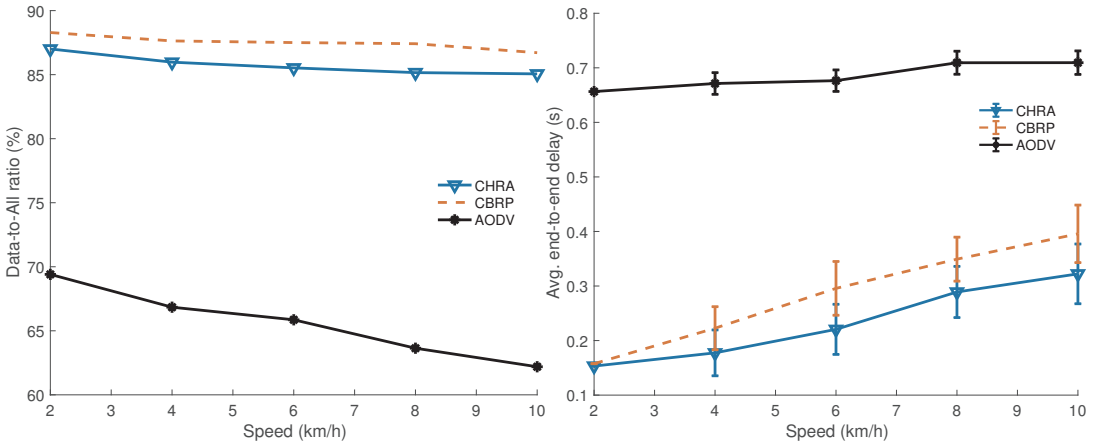
The average energy consumption is slightly higher than the results in stationary scenarios as shown in Figure 6.13b and Figure 6.14b. Because of the mobile nodes, re-routing, recovery and repair processes are more frequent in mobile scenarios. As seen in 6.13a and Figure 6.14a, CHRA has a low standard deviation in energy consumption even in higher mobility with increasing speed. Note that, having different standard deviations in parallel to the same ratio of energy consumption between CHRA and CBRP shows that CHRA promotes fairer energy consumption in mobile scenarios as well. Again, the cross-interpretation of the average energy consumption and the standard deviation in energy consumption reveals such effect of the control and user plane separation. In uniform scenarios, the increasing speed of nodes observably affects the standard deviation in energy consumption since the mobility strongly changes the already-sparse network distribution. In contrast, it is not affected especially for CHRA since the tolerance to mobility in a denser area that is mostly covered by CSAs is much higher. Therefore, routes can be maintained more effectively in CHRA thanks to CSA structure.

Figure 6.13d and Figure 6.14d show that the overall end-to-end delay for all routing algorithms is lower in nonuniform scenarios than uniform scenarios. The ordering between the algorithms is the same, CHRA has the lowest end-to-end delay in any case even if it is increasing with speed of nodes for every algorithm. Interpreting them in Figure 6.13c and Figure 6.14c, it is seen that CHRA decreases the end-to-end delay preserving a high DAR as 85%. The DAR is decreasing with the increasing speed of nodes since the high mobility triggers routing process and eventually in-



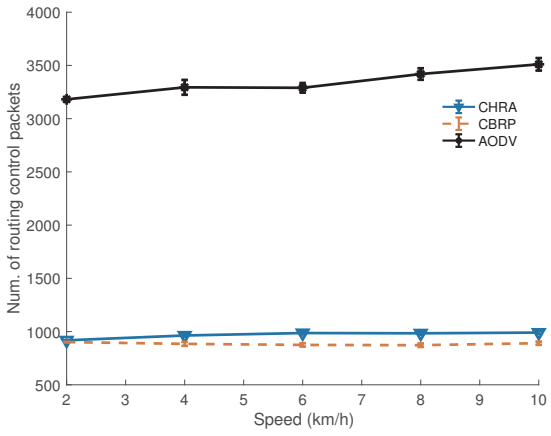
(a) Standart deviation in energy consumption

(b) Average energy consumption ratio per node



(c) Data-to-All Ratio

(d) Average end-to-end delay



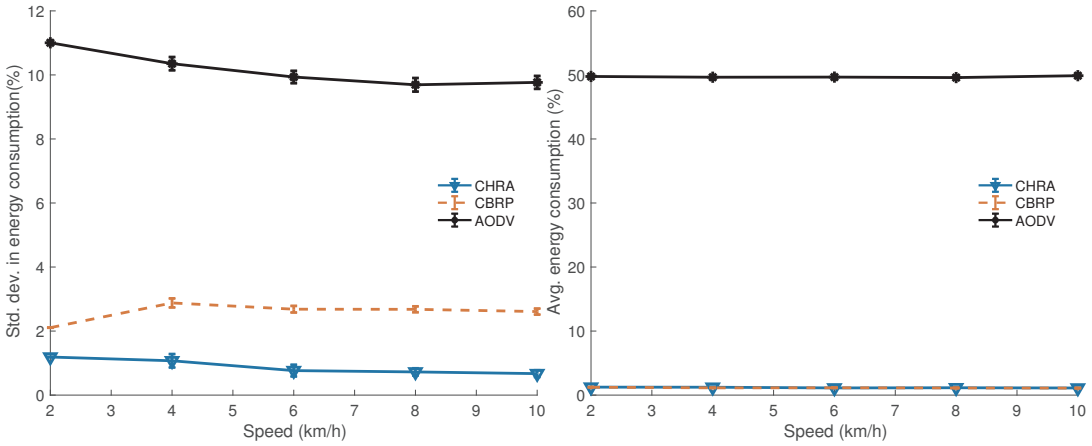
(e) Number of control packets

Figure 6.13: The effects of the speed in uniformly distributed scenarios

crease routing control overhead. Note that, CHRA is affected by mobility more than CBRP because the routes depend on many ordinary nodes which are mobile instead of the backbone (i.e., CHs and gateways) that is relatively easy to fix with during the clustering maintenance. When a new CH is selected, the cluster neighborhood can be recovered easily and the traffic flows through the backbone. However, thanks to route repair and recovery process for the data plane-routes in CHRA, even if it is more sensitive to mobility, there is not a DAR decrease due to increasing speed of nodes. The difference of DAR results between CHRA and CBRP is caused by extra recovery process and CSA maintenance in CHRA. Figure 6.13e and Figure 6.14e show such extra overhead in terms of the number of control packets. Besides, they explain the low-DAR results of AODV that demonstrates a high control overhead.

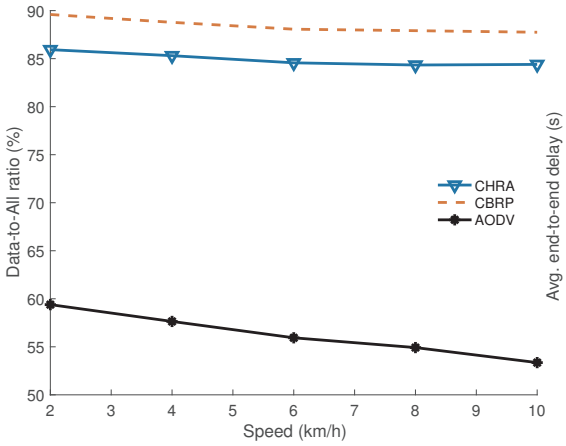
6.2.3 The Maintenance of CSA

At the end of the discussion, the control overhead for CSA maintenance is worth touching. As seen in Figure 6.15, PDR is decreasing with increasing period of SAM packets, T_{SAM}. In contrast, control overhead is getting less with more infrequent SAM packets as expected. The reason is, the infrequent SAM packets directly lead to routing based-on obsolete topology information. In this case, packets cannot be forwarded through destination when the route repair is not possible. In this manner, T_{SAM} need to be decided based on mobility characteristics of the network.

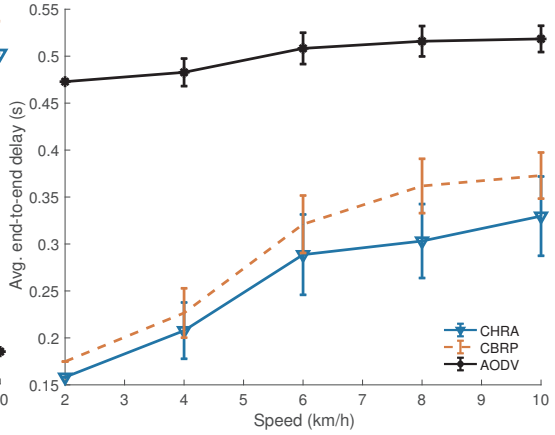


(a) Standart deviation in energy consumption

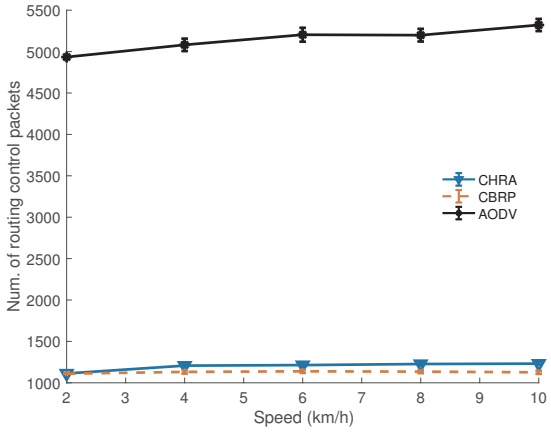
(b) Average energy consumption ratio per node



(c) Data-to-All ratio



(d) Average end-to-end delay



(e) Number of control packets

Figure 6.14: The effects of the speed in nonuniform scenarios

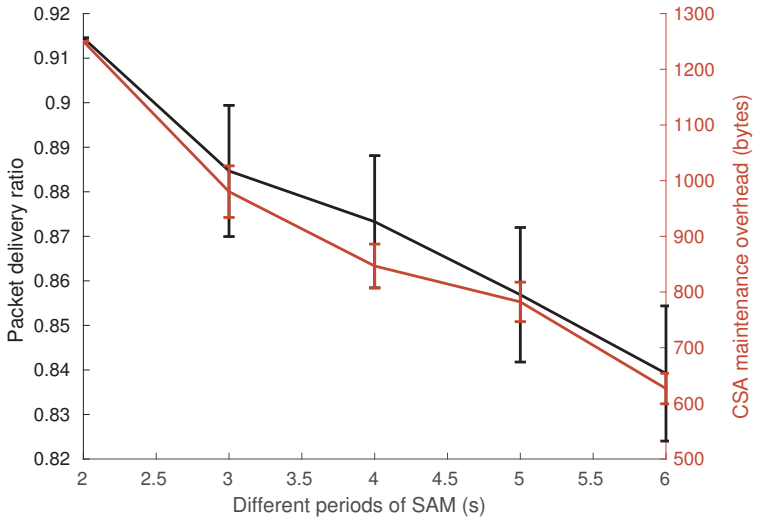


Figure 6.15: The effects of T_{SAM} on packet delivery ratio and overhead

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this chapter, a brief conclusion for the whole design is given. Besides, future work and possible extensions of this study are discussed.

7.1 Conclusion

Through the thesis, first, the plane-separated architecture is built to able to manage and orchestrate ad-hoc networks. The dependability-based clustering technique DCA for ad-hoc networks is presented and a sensitivity analysis framework is offered to evaluate any weighted clustering algorithm. DCA is validated by the discrete event-based simulations. Apart from searching for the optimal weights; the positive, negative and neutral (e.g., irregular or none) effects of different metrics are discussed to be able to design a goal-based optimized clustering algorithm. In the sensitivity analysis, it is also shown that different parameters may have unexpected effects on the objectives. For instance, degree-related metrics such as *SCDR*, *CDR* and *Centrality* do not have observable positive effects on the stability while the metrics which considers changing communication-quality conditions like *LQIR* and *Contraction* are important for stability according to SA results. Moreover, it is presented that focusing on cluster stability, or *Contraction*, is important to provide QoS. To create energy-efficient clusters (i.e., fair and efficient energy-consumption for member nodes), using an energy-dependent metric is not a solution. Instead, *SCDR* and *CDR*, or degree-related metrics, positively affect energy efficiency. The results show that

the optimization with SA leads to a significant performance improvement for DCA in stationary and mobile scenarios considering any performance metric presented in the study. Since the cluster dependability is taken into consideration alongside the node eligibility, DCA has shown better performance in comparison to its rivals. However, it is shown that the weighted algorithm, DCA, requires to find optimal weights for an efficient objective-based design.

After the formation of CUPS architecture, a plane-separated routing algorithm in ad-hoc networks, CHRA, is presented to establish an end-to-end communication scheme in the hierarchical network structure. The separation of the control plane and the user plane leads to finding alternative routes that are not dependent on the backbone in contrast to many other cluster-based routing algorithms. Using those alternatives provides a fair energy-consumption scheme since a significant data forwarding burden is taken from the control plane, and distributed to other nodes which leads to effective use of the user plane. The results also show that using a proper route recovery mechanism and establishing alternative paths in the user plane, CHRA can handle data transfer with a lower end-to-end delay than the technique that purely uses backbone for both routing and forwarding, while maintaining a high-level packet throughput even in mobile scenarios.

The overall study shows that the control and the user plane separation is a quite convenient concept that can be applied in ad-hoc networks. While clustering itself is a CUPS solution by nature, exploiting the hierarchy created by clustered formation is also an opportunity to design more advance CUPS-centric algorithms like CHRA. All in all, DCA and CHRA are presented as a complete framework that takes the plane-separated approach to increase energy efficiency and the quality of service.

7.2 Future Work

There are some possible extensions for the overall design, and also for DCA and CHRA separately. In this study, a CUPS-centric architecture is designed for ad-hoc networks in terms of clustering and routing. However, some other important points

are not touched: resource allocation and link scheduling. When a network is divided into clusters, a huge opportunity for flexible resource allocation (i.e., frequency reuse) shows up. Moreover, intra- and inter-cluster link scheduling considering the dynamically allocated resources are directly complementary for the design presented here. That kind of holistic approach, which is dynamically solving (1) how to organize ad-hoc networks, (2) how to use resource effectively and (3) how to satisfy an end-to-end communication with certain QoS requirements, would be an example architecture for next-generation networks. Apart from that, the deployment of a well-known link scheduling algorithm such as Carrier-sense Multiple Access (CSMA) and TDMA may increase the performance of the overall design since there are many periodic control packets and also random data traffic.

A significant part of Chapter 6 explains the sensitivity analysis framework for DCA. The difference between optimized and non-optimized versions of DCA is also presented there. DCA can be also optimized with different optimization methods mentioned in Chapter 2 to compare them with the framework presented in this thesis. However, no other method proposes a detailed analysis as it is discussed here. Therefore, they are required to be extended to reflect the direct relationship between metrics and different objectives. Moreover, the different parameters of DCA such as periods can be analytically discussed and optimized by the sensitivity analysis framework. An optimization is also applicable to the scenarios with different physical channel conditions, node distributions etc.

CUPS architecture can be used to satisfy many other QoS requirements for CHRA. For example, while continuous traffic e.g., a phone call or any session-based application layer protocol and non-continuous e.g., text messaging have different characteristics. Therefore, various modifications on CHRA to decrease control overhead and end-to-end delay, and increase data-to-all ratio are possible. Moreover, even if CUPS architecture is focused in this study, CHRA can compete with any other hybrid routing algorithm. In the future work, such comparison would be performed. Lastly, CSA directly affects the efficiency of EEPs for the in-area communication. Therefore, the size of CSA will be investigated in more detail possibly for larger networks.

REFERENCES

[1] C Siva Ram Murthy and BS Manoj. *Ad hoc Wireless Networks: Architectures and Protocols*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[2] S. Abe, G. Hasegawa, and M. Murata. Effects of C/U Plane Separation and Bearer Aggregation in Mobile Core Network. *IEEE Transactions on Network and Service Management*, 15(2):611–624, June 2018.

[3] A. Mohamed, O. Onireti, M. A. Imran, A. Imran, and R. Tafazolli. Control-Data Separation Architecture for Cellular Radio Access Networks: A Survey and Outlook. *IEEE Communications Surveys Tutorials*, 18(1):446–465, Firstquarter 2016.

[4] P. Arnold, N. Bayer, J. Belschner, and G. Zimmermann. 5G radio access network architecture based on flexible functional control / user plane splits. In *Proc. of the European Conference on Networks and Communications (EuCNC)*, pages 1–5, June 2017.

[5] F. Eichhorn, M. I. Corici, T. Magedanz, P. Du, Y. Kiriha, and A. Nakao. SDN enhancements for the sliced, deep programmable 5G core. In *Proc. of the 13th International Conference on Network and Service Management (CNSM)*, pages 1–4, Nov 2017.

[6] F. Bannour, S. Souihi, and A. Mellouk. Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Communications Surveys Tutorials*, 20(1):333–354, Firstquarter 2018.

[7] I. H. Abdulqadder, D. Zou, I. T. Aziz, and B. Yuan. Modeling software defined security using multi-level security mechanism for SDN environment. In *Proc. of the IEEE 17th International Conference on Communication Technology (ICCT)*, pages 1342–1346, Oct 2017.

[8] Craig Cooper, Daniel Franklin, Montserrat Ros, Farzad Safaei, and Mehran Abolhasan. A comparative survey of VANET clustering techniques. *IEEE Communications Surveys & Tutorials*, 19(1):657–681, 2017.

[9] Giorgia V Rossi, Zhong Fan, Woon Hau Chin, and Kin K Leung. Stable Clustering for Ad-Hoc Vehicle Networking. In *on Proc. of the WCNC*, pages 1–6, 2017.

[10] Y. C. Chen and C. Y. Wen. Adaptive Cluster-Based Scheduling Management for Wireless Ad-Hoc Sensor Networks. In *Proc. of the Third International Conference on Sensor Technologies and Applications*, pages 256–263, June 2009.

[11] S. Kamble and T. Dhope. Reliable routing data aggregation using efficient clustering in WSN. In *Proc. of the International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pages 246–250, May 2016.

[12] E. Borgonovo. A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6):771 – 784, 2007.

[13] Anthony Ephremides, Jeffrey E Wieselthier, and Dennis J Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, 75(1):56–73, 1987.

[14] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proc. of the IEEE SICON*, volume 97, pages 197–211, 1997.

[15] Chunhung Richard Lin and Mario Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, 1997.

[16] Ossama Younis and Sonia Fahmy. Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on mobile computing*, 3(4):366–379, 2004.

[17] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of the 33rd Annual Hawaii System sciences*, pages 10–pp. IEEE, 2000.

[18] Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, mobile, multimedia radio network. *Wireless networks*, 1(3):255–265, 1995.

[19] JY Yu and Peter HJ Chong. 3hbac (3-hop between adjacent clusterheads): a novel non-overlapping clustering algorithm for mobile ad hoc networks. In *Proc. of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, volume 1, pages 318–321. IEEE, 2003.

[20] Taek Jin Kwon, Mario Gerla, Vijay K Varma, Melbourne Barton, and T Russell Hsing. Efficient flooding with passive clustering-an overhead-free selective forward mechanism for ad hoc/sensor networks. *Proceedings of the IEEE*, 91(8):1210–1220, 2003.

[21] A Bruce McDonald and Taieb F Znati. Design and performance of a distributed dynamic clustering algorithm for ad-hoc networks. In *Proc. of the 34th Annual Simulation Symposium*, pages 27–35. IEEE, 2001.

[22] Zhenxia Zhang, Azzedine Boukerche, and Richard Pazzi. A novel multi-hop clustering scheme for vehicular ad-hoc networks. In *Proc. of the 9th ACM international symposium on Mobility management and Wireless Access*, pages 19–26. ACM, 2011.

[23] Prithwish Basu, Naved Khan, and Thomas DC Little. A mobility based metric for clustering in mobile ad hoc networks. In *Proc. of the International Distributed Computing Systems Workshop*, pages 413–418. IEEE, 2001.

[24] Sahar Adabi, Sam Jabbehdari, Amir Masoud Rahmani, and Sepideh Adabi. Sbca: Score based clustering algorithm for mobile ad-hoc networks. In *Proc. of the 9th International Conference for Young Computer Scientists*, pages 427–431. IEEE, 2008.

[25] Mainak Chatterjee, Sajal K Das, and Damla Turgut. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Cluster computing*, 5.

[26] Wei-dong Yang and Guang-zhao Zhang. A weight-based clustering algorithm for mobile ad hoc network. In *Wireless and Mobile Communications, 2007. ICWMC'07. Third International Conference on*, pages 3–3. IEEE, 2007.

[27] Waseem Shahzad, Farrukh Aslam Khan, and Abdul Basit Siddiqui. Clustering in mobile ad hoc networks using comprehensive learning particle swarm optimization (CLPSO). In *Communication and Networking*, pages 342–349. Springer, 2009.

[28] Hamid Ali, Waseem Shahzad, and Farrukh Aslam Khan. Energy-efficient clustering in mobile ad-hoc networks using multi-objective particle swarm optimization. *Applied Soft Computing*, 12(7):1913–1928, 2012.

[29] Farhan Aadil, Khalid Bashir Bajwa, Salabat Khan, Nadeem Majeed Chaudary, and Adeel Akram. CACONET: ant colony optimization (ACO) based clustering algorithm for VANET. *PloS one*, 11(5):e0154080, 2016.

[30] Muhammad Fahad, Farhan Aadil, Salabat Khan, Peer Azmat Shah, Khan Muhammad, Jaime Lloret, Haoxiang Wang, Jong Weon Lee, Irfan Mehmood, et al. Grey wolf optimization based clustering algorithm for vehicular ad-hoc networks. *Computers & Electrical Engineering*, 2018.

[31] Mohamed Hadded, Rachid Zagrouba, Anis Laouiti, Paul Muhlethaler, and Leila Azouz Saidane. A multi-objective genetic algorithm-based adaptive weighted clustering protocol in VANET. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 994–1002. IEEE, 2015.

[32] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[33] Mildred M Caballeros Morales, Choong Seon Hong, and Young-Cheol Bang. An adaptable mobility-aware clustering algorithm in vehicular networks. In *Proc. of the 13th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–6. IEEE, 2011.

[34] Yuzhong Chen, Mingyue Fang, Song Shi, Wenzhong Guo, and Xianghan Zheng. Distributed multi-hop clustering algorithm for vanets based on neighborhood follow. *Eurasip journal on Wireless communications and networking*, 2015(1):98, 2015.

[35] Seyhan Ucar, Sinem Coleri Ergen, and Oznur Ozkasap. Multihop-cluster-based ieee 802.11 p and lte hybrid architecture for vanet safety message dissemination. *IEEE Transactions on Vehicular Technology*, 65(4):2621–2636, 2016.

[36] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proc. of the IEEE International Multi Topic Conference*, pages 62–68, 2001.

[37] Charles E Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM Computer Communication Review*, volume 24, pages 234–244. ACM, 1994.

[38] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proc. of the Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, Feb 1999.

[39] Zygmunt Haas, Marc R. Pearlman, and Prince Samar. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. Internet-Draft draft-ietf-manet-zone-zrp-04, Internet Engineering Task Force, August 2002. Work in Progress.

[40] Mingliang Jiang, Jinyang Li, and Yong Chiang Tay. Cluster Based Routing Protocol(CBRP) Functional Specification. Internet-Draft draft-ietf-manet-cbrp-spec-01, Internet Engineering Task Force, August 1999. Work in Progress.

[41] J. Y. Yu, P. H. J. Chong, and M. Zhang. Performance of Efficient CBRP in Mobile Ad Hoc Networks (MANETs). In *Proc. of the IEEE 68th Vehicular Technology Conference*, pages 1–7, Sept 2008.

[42] Yi Wang, Liang Dong, Taotao Liang, Xinyu Yang, and Deyun Zhang. Cluster based location-aided routing protocol for large scale mobile ad hoc networks. *IEICE Transactions on Information and Systems*, E92.D(5):1103–1124, 2009.

[43] Yakov Rekhter, Susan Hares, and Tony Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006.

[44] Biao Zhou, Zhen Cao, and Mario Gerla. Cluster-based inter-domain routing (CIDR) protocol for MANETs. In *Proc. of the sixth International Conference on Wireless On-Demand Network Systems and Services (WONS)*, pages 19–26. IEEE, 2009.

[45] Domenico De Guglielmo, Simone Brienza, and Giuseppe Anastasi. IEEE 802.15.4e: A survey. *Computer Communications*, 88:1 – 24, 2016.

[46] Kazem Jahanbakhsh and Marzieh Hajhosseini. Improving performance of cluster based routing protocol using cross-layer design. *arXiv preprint arXiv:0802.0543*, 2008.

[47] M. Joa-Ng and I-Tai Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1415–1425, Aug 1999.

[48] Hui-Yao An, Ling Zhong, Xi-Cheng Lu, and Wei Peng. A cluster-based multipath dynamic source routing in MANET. In *Proc. of the IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob)*, volume 3, pages 369–376. IEEE, 2005.

[49] L. Ming, G. Zhao, G. Xie, and X. Kuang. Holsr: A novel routing scheme of ad hoc wireless networks for pervasive computing. In *Proc. of the 2nd International Conference on Pervasive Computing and Applications*, pages 661–666, July 2007.

[50] X. Niu, Z. Tao, G. Wu, C. Huang, and L. Cui. Hybrid Cluster Routing: An Efficient Routing Protocol for Mobile Ad Hoc Networks. In *Proc. of the IEEE International Conference on Communications*, volume 8, pages 3554–3559, June 2006.

[51] S. Asadinia, Marjan kuchaki Rafsanjani, and Arsham Borumand Saeid. A novel routing algorithm based-on ant colony in Mobile Ad hoc Networks. In *Proc. of*

the 3rd IEEE International Conference on Ubi-Media Computing, pages 77–82, July 2010.

[52] Kaixin Xu, Xiaoyan Hong, and Mario Gerla. Landmark routing in ad hoc networks with mobile backbones. *Journal of Parallel and Distributed Computing*, 63(2):110 – 122, 2003. Routing in Mobile and Wireless Ad Hoc Networks.

[53] Z. El-Bazzal, M. Kadoch, B. L. Agba, F. Gagnon, and M. Bennani. A flexible weight based clustering algorithm in mobile ad hoc networks. In *Proc. of the International Conference on Systems and Networks Communications (ICSNC)*, pages 50–50, Oct 2006.

[54] V. Srivastava and M. Motani. Cross-layer design: a survey and the road ahead. *IEEE Communications Magazine*, 43(12):112–119, 2005.

[55] R. Glitho, C. Fu, and F. Khendek. Cross-layer design for optimizing the performance of clusters-based application layer schemes in mobile ad hoc networks. In *Proc. of the 4th IEEE Consumer Communications and Networking Conference*, pages 239–243, Jan 2007.

[56] M. K. Denko, J. Tian, T. K. R. Nkwe, and M. S. Obaidat. Cluster-based cross-layer design for cooperative caching in mobile ad hoc networks. *IEEE Systems Journal*, 3(4):499–508, Dec 2009.

[57] R. Mehta and D. K. Lobiyal. Energy efficient cross-layer design in manets. In *Proc. of the 4th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 448–453, Feb 2017.

[58] J. Peng, H. Niu, W. Huang, X. Yin, and Y. Jiang. Cross layer design and optimization for multi-hop ad hoc networks. In *Proc. of the IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 1678–1682, March 2017.

[59] Microchip. Microchip rn1810, Jun 2018.

[60] SparkLAN. Sparklan wsdb-102gn, Jun 2018.

[61] Chuanwen Luo, Jiguo Yu, Dongxiao Yu, and Xiuzhen Cheng. Distributed algorithms for maximum clique in wireless networks. In *on Proc. of the 2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, MSN '15, pages 222–226, Washington, DC, USA, 2015. IEEE Computer Society.

[62] Z. Ismail and R. Hassan. Effects of Packet Size on AODV Routing Protocol Implementation in Homogeneous and Heterogeneous MANET. In *Proc. of the Third International Conference on Computational Intelligence, Modelling Simulation*, pages 351–356, Sept 2011.

[63] Yanjun Gan, Qingyun Duan, Wei Gong, Charles Tong, Yunwei Sun, Wei Chu, Aizhong Ye, Chiyuan Miao, and Zhenhua Di. A comprehensive evaluation of various sensitivity analysis methods: A case study with a hydrological model. *Environmental Modelling & Software*, 51:269 – 285, 2014.

[64] Elmar Plischke, Emanuele Borgonovo, and Curtis L. Smith. Global sensitivity measures from given data. *European Journal of Operational Research*, 226(3):536 – 550, 2013.

[65] Esa Hyytiä and Jorma Virtamo. Random waypoint mobility model in cellular networks. *Wireless Networks*, 13(2):177–188, Apr 2007.

[66] Abhay K Parekh. Selecting routers in ad-hoc wireless networks. In *Proc. of the SBT/IEEE International Telecommunications Symposium*, volume 204. Rio de Janeiro (Brazil), 1994.

[67] Yuna Kim, Ki-Young Jung, Tae-Hyung Kim, and Jong Kim. A distributed energy-efficient clustering scheme for deploying ids in MANETs. *Telecommunication Systems*, 52(1):85–96, 2013.